# CRIANDO SOLUÇÕES TECNOLÓGICAS COM A ENGENHARIA DE COMPUTAÇÃO

VOLUME 1

Pascal
Editora

EDILSON CARLOS SILVA LIMA
ELDA REGINA DE SENA CARIDADE
JONATHAN ARAUJO QUEIROZ
MARCOS JOSÉ DOS PASSOS SÁ
WILL RIBAMAR MENDES ALMEIDA
YONARA COSTA MAGALHÃES
(Organizadores)

# CRIANDO SOLUÇÕES TECNOLÓGICAS COM A ENGENHARIA DE COMPUTAÇÃO

VOLUME 1

EDITORA PASCAL
2023

**Editor Chefe:** Prof. Dr. Patrício Moreira de Araújo Filho

**Edição e Diagramação:** Eduardo Mendonça Pinheiro

**Edição de Arte:** Ilgner Mendes Bezerra e Eduardo Mendonça Pinheiro

**Bibliotecária:** Rayssa Cristhália Viana da Silva – CRB-13/904

**Revisão:** Os autores



**Dados Internacionais de Catalogação na Publicação (CIP)**

**M688c**

Coletânea Criando Soluções Tecnológicas com a Engenharia de Computação / Edilson Carlos Silva Lima et al. (Orgs). São Luís - Editora Pascal, 2023.

f. : il.: (Criando soluções tecnológicas com a engenharia de computação; v. 1)

Formato: PDF

Modo de acesso: World Wide Web

ISBN: 978-65-80751-

D.O.I.:

1. . 2. . 3. . 4. . I. Lima, Edilson Carlos Silva. II. Caridade, Elda Regina de Sena. III. Queiroz, Jonathan Araujo. IV. Sá, Marcos José dos Passos. V. Almeida, Will Ribamar Mendes. VI. Magalhães, Yonara Costa. VII. Título

CDU:



**2023**

www.editorapascal.com.br

contato@editorapascal.com.br

# APRESENTAÇÃO

Dedico este livro aos docentes do curso de Engenharia de Computação da Universidade Ceuma, que com muita empolgação ao desconhecido e que ao avançar nessa trajetória de um curso de graduação, descobrem um universo computacional e colocam em prática o ensinamento que buscam criando soluções.

Criar soluções computacionais com boa funcionalidade e aparência é complexo, pois requer dedicação, estudo, aquisição de conhecimentos transdisciplinares e experiência no desenvolvimento dessas soluções que devem focar precipuamente em atender ao usuário.

Este livro é formado por várias soluções desenvolvidas como práticas de disciplinas cursadas pelos alunos durante a graduação, nele temos ideias de negócios que surgem e se apresentam como soluções para as necessidades da rotina de um nicho de usuário. Não tendo o caráter de excluir ou impor tais soluções como únicas, mas de colaborar com o processo de aprendizagem.

Neste livro temos back-ends desenvolvidos com o uso de Framework Spring Boot que é capaz de fornecer os mesmos recursos e informações para diferentes tipos de interface e front-end, que é desenvolvimento de interfaces gráficas de sites e aplicativos, através do uso de frameworks composto pelos por HTML, CSS e JavaScript, para que os usuários possam visualizar e interagir com as soluções desenvolvidas. O que leva a experiência de full Stack profissional multitarefa que utiliza grandes players que agrega tanto no dinamismo, quanto nas funcionalidades de sites e aplicativos.

Os Sites web e aplicações desenvolvidos devem buscar facilidades no acesso e incentivo nos ramos das variadas pesquisas desenvolvidas através de conceito do tipo Single Page Application (SPA). Front-ends elaborados através do framework de JavaScript Vue.Js. Aplicativos desenvolvidos em Flutter, um framework da Google que funciona com a linguagem Dart, onde com apenas um código conseguimos gerar aplicativos nativos para plataformas móveis além de web e desktop, esse framework soluciona o problema do desempenho e trabalha na plataforma como se fosse o próprio código nativo sendo ali compilado.

Os desafios das transformações ao longo do desenvolvimento dos trabalhos foram grandes e, através de uma visão de inovação, a resposta para seus usuários, trouxe maximização nos desenvolvimentos. Uma boa métrica foi estabelecer o que chamamos de startup gap, ou seja, quais e quantas tecnologias em diferentes partes do negócio que utilizamos poderiam ser melhoradas com a aplicação de novas disponíveis no ecossistema de uma solução tecnológica. Tradicionalmente, os trabalhos têm um modelo de atuação flexível, pautado pela tecnologia, suportado por soluções com uso de ferramentas de colaboração. Um dos reflexos desses trabalhos foi a experiência positiva da vivência da necessidade dos usuários focada na intensidade da usabilidade das soluções. Tornar um projeto viável é importante para que ele cresça de forma sustentável, alcance os resultados aguardados pelos envolvidos e potencialize essas soluções.

# ORGANIZADORES

**Edilson Carlos Silva Lima**

Mestrando em Engenharia da Informática com ênfase na área de Sistemas e Tecnologias de Informação na Universidade Fernando Pessoa na Cidade do Porto em Portugal (com previsão de término em 2023), pós-draduado em Análise e Projeto de Sistemas (UFMA, 2009.1), graduado em Sistema de Informação pela Universidade CEUMA (2008.2) e Tecnólogo em Tecnologia de Informática pela Universidade CEUMA (2003.1), atuou como Programador, Analista de Sistema, Gerente de TI, é professor desde 2012.

**LATTES:** https://lattes.cnpq.br/3633743402684029

**ORCID:** https://orcid.org/0000-0002-2301-8006

**Elda Regina de Sena Caridade**

Bacharel em Ciência da Computação pela Universidade Federal do Maranhão (1998) e Mestra em Engenharia de Computação e Sistemas pela Universidade Estadual do Maranhão-UEMA. Atuando na área de docência do ensino superior desde 2002 na Universidade Ceuma e em outras Instituições de Ensino Superior. Atualmente coordena os cursos de Engenharia de Computação e Sistemas de Informação da Universidade CEUMA. Tem experiência na área de Engenharia de Computação, Análise e Desenvolvimento de Sistemas.

**LATTES:** http://lattes.cnpq.br/8833973908569237

**ORCID:** https://orcid.org/0000-0003-2243-0477

**Jonathan Araujo Queiroz**

Possui graduação em matemática licenciatura pela Universidade Federal do Maranhão (2012), especialista em Métodos Estatísticos Aplicados pela Universidade Estadual do Maranhão (2016), mestrado em Engenharia de Eletricidade pela Universidade Federal do Maranhão (2016), doutorado em Engenharia Elétrica pela Universidade Federal do Maranhão (2018) e Pós-Doutorado pela Universidade Federal do Maranhão (2020). É colaborador da Sociedade Brasileira de Engenharia Biomédica e revisor de IEEE Access, IEEE Engineering in Medicine and Biology Society, Journal of Cardiac Disorders and Therapy (JCDT), e Electric Power Components and Systems Journal.

**LATTES:** http://lattes.cnpq.br/7145102625820184

**ORCID:** https://orcid.org/0000-0001-8006-6242

# ORGANIZADORES

**Marcos José dos Passos Sá**

Possui graduação em Tecnólogo em Desenvolvimento de Sistemas pela Universidade Ceuma (2008), Especialização em MBA Governança de TI pela Universidade Ceuma (2013) e Mestrado em Engenharia de Computação e Sistemas pela Universidade Estadual do Maranhão (2020). Atualmente é Professor Mestre da Universidade Estadual do Maranhão e Professor da Universidade Ceuma. Tem experiência na área de Ciência da Computação, com ênfase em Sistemas de Computação, Cloud Computer e Telecomunicações.

**LATTES:** http://lattes.cnpq.br/2111990319894447

**Will Ribamar Mendes Almeida**

Possui graduação em Engenharia Industrial Elétrica pelo Instituto Federal do Maranhão (2002), mestrado em Engenharia de Eletricidade pela Universidade Federal do Maranhão (2004) e doutorado em Engenharia Elétrica pela Universidade Federal de Campina Grande (2009). Atua nos seguintes temas: Automação Residencial, Desenvolvimento de Software de Gestão de TI e Educacional.

**LATTES:** http://lattes.cnpq.br/2668882206079613

**ORCID:** https://orcid.org/0000-0001-5999-7536

**Yonara Costa Magalhães**

É bacharel em Ciência da Computação pela Universidade Federal do Maranhão (1998) e mestre em Engenharia de Eletricidade pela Universidade Federal do Maranhão (2002). É professora titular da Universidade do CEUMA (2011) e professora contratada na Universidade Estadual do Maranhão (UEMA/2022). Foi docente no Centro Universitário Euro-Americano (UNIEURO- DF 2006/2011), tendo atuado como: Pesquisadora Institucional, membro do NDE de Sistemas de Informação e de CST em Redes de Computadores, Coordenadora dos Cursos de Sistemas de Informação, CST em Redes de Computadores e CST em Gestão da Tecnologia da Informação e Coordenadora Pedagógica. Foi Coordenadora do Curso de CST em Análise e Desenvolvimento de Sistemas no CEUMA. É pesquisadora do NuSTI/CnPQ (Núcleo de Pesquisa em Sistemas e Tecnologia da Informação) do CEUMA e membro da Sociedade Brasileira de Computação (SBC).

**LATTES:** http://lattes.cnpq.br/8188763596503654

**ORCID:** https://orcid.org/0000-0001-5502-9634

# SUMÁRIO

# 1

## O USO DA METODOLOGIA SCRUM PARA VALIDAÇÃO DE UM APLICATIVO NO MODELO DE NEGÓCIO DE UMA STARTUP COM FERRAMENTAS COMO CSD, CANVAS, MVP E PROPOSTA DE VALOR

*THE USE OF THE SCRUM METHODOLOGY TO VALIDATE AN APPLICATION IN A STARTUP'S BUSINESS MODEL WITH TOOLS SUCH AS CSD, CANVAS, MVP AND VALUE PROPOSITION*

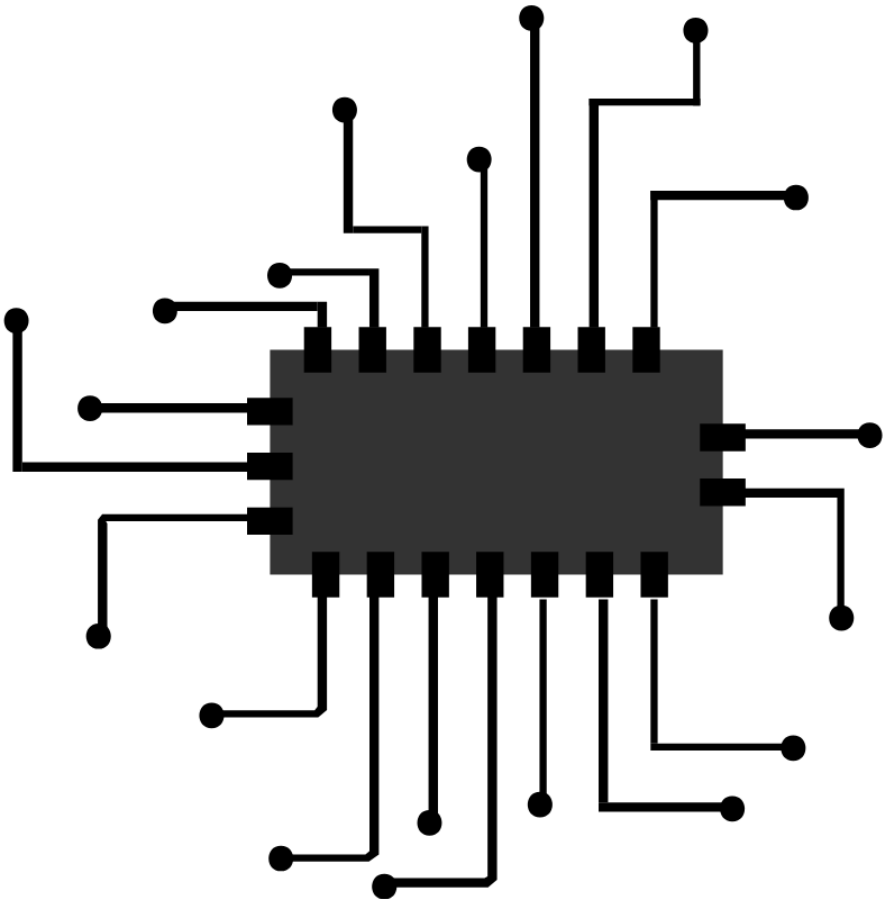
Julyana Corrêa Silva[1]

Edilson Carlos Silva Lima[2]

1 Engenharia da Computação – Universidade Ceuma (UniCEUMA) – São Luís – MA – Brasil

2 Engenharia da Computação – Universidade Ceuma (UniCEUMA) – São Luís – MA– Brasil

{SILVA, Julyana, julyanacorreasilva@gmail.com; LIMA, Edilson Carlos Silva, edilsonlima3@gmail.com.}



**d.o.i.:**

## Resumo

Diariamente ideias de negócios surgem e se apresentam como grandes soluções para as necessidades da rotina. Algumas delas se tornam grandes negócios, já outras não. Não existe uma fórmula certa para alcançar o sucesso, no entanto, a validação é uma etapa fundamental para quem deseja empreender. O *scrum* pertence a uma estrutura leve que é usada em todos os setores para fornecer produtos e serviços complexos e inovadores. Ao se tratar disso, aplicamos no modelo de startup, no qual a metodologia aborda diversas técnicas, dentre elas, oferece o mapa de empatia, a matriz CSD (Certezas, Suposições e Dúvidas), o canvas e proposta de valor, produto mínimo viável (MVP) entre outras metodologias, ou seja, através dessas aplicações, criar a versão mais simples e eficiente de um produto usando, assim com essas técnicas concebemos um guia para um projeto inicial pra agregar melhorias ao longo do processo.


**Palavras-chave:** Scrum, CSD, Canvas, MVP, Proposta de valor.


## Abstract

Daily business ideas arise and present themselves as great solutions for everyday needs. Some of them become big businesses, some don't. There is no right formula to achieve success, however, validation is a fundamental step for anyone who wants to undertake. Scrum belongs to a lightweight framework that is used across industries to deliver complex and innovative products and services. When it comes to this, we apply it to the startup model, in which the methodology addresses several techniques, among them, it offers the empathy map, the CSD matrix (Certainties, Assumptions and Doubts), the canvas and value proposition, minimum viable product ( MVP) among other methodologies, that is, through these applications, create the simplest and most efficient version of a product using, so with these techniques we designed a guide for an initial project to add improvements throughout the process.


**Keywords:** Scrum, CSD, Canvas, MVP, Value proposition.

# 1. INTRODUÇÃO

Um startup é uma empresa concebida para crescer rápido. "A única coisa essencial é o crescimento. Todo o resto que nós associamos com *startups* decorre do crescimento" (GRAHAM, 2012, p.15).

Segundo Ries (2012), uma *startup* é um processo inicial projetada para engendrar um novo produto ou trabalho, em meios incertos. No entanto, é um novo experimento, que serve para deslindar recursos para o problema de seus consumidores (TORRES, 2012, p.15).

Graham (2012) afirma que o principal atributo de uma *startup* é seu desenvolvimento de forma ágil, pela definição de Blank (2012), "é uma corporação temporária elaborada para obter respostas que alcancem a um modelo de negócio recorrente e de grande prontidão".

O ato de comunicar é de extrema importância e serve como um instrumento de conexão, instrução, de troca de informações entre um ou mais seres. E, em um condomínio, essa dificuldade fica em evidente, pois são indivíduos com diferentes vivências e maneiras de vida tendo que habituar-se e no que é melhor para todos. Em um condomínio amplo, o processo de comunicação é mais demorado e difícil. Em vista disso, surge uma problemática que dificulta o gerenciamento para utilizações das devidas áreas comuns e uteis para os condôminos (ex.: Portaria, área de lazer como piscinas, quadras, parquinhos, salão de festa). Como resolver o problema? Quais medidas que auxiliam na resolução do problema entre o gerenciamento e condômino? Em relações a custos-benefícios quais são os necessários para ter uma resolução mais rápida e prática? Com o surgimento de todas essas perguntas, desenvolvemos esse projeto para que as perguntas sejam atendidas. Através de alguns softwares serão gerenciadas desde a situação mais complexas até a mais simples, visando público de grande a pequeno porte de condomínios.

# 2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, vamos mostrar umas metodologias bem comuns utilizadas em *startup* para fazer uma implementação de um aplicativo de segurança dentro de um condomínio e está dividido nos seguintes itens: 2.1 *Srum*, no item 2.2 árvore de problemas, item 2.3 persona, no item 2.4 a mapa de empatia, no item 2.5 a matriz CSD, no item 2.6 *TAM, SAM, SOM*, no item 2.7 o Mercado de Negócio B2B, o item 2.8 Canvas e proposta de valor, 2.9 *Product/Market Fit* (PMF) e por fim 2.10 Produto Mínimo Viável (MVP).

## 2.1 Scrum

Scrum refere-se a uma estrutura leve que é usada em todos os setores para fornecer produtos e serviços complexos e inovadores que despertam interesses em determinados clientes. É uma ferramenta simples de entender, porém, é difícil de dominar. Embora tenha suas raízes no desenvolvimento de software, no entanto, o *Scrum* é usado atualmente em uma variedade de indústrias.

De acordo com o *The Scrum Guidetm* scrum é "um *framework* de grande velocidade que ajuda pessoas, equipes e organizações a gerar um determinado valor, por meio de soluções adaptáveis para problemas" (SCHWABER; SUTHERLAND, 2017). Scrum se assemelha a sistemas evolutivos, adaptativos e autocorretivos. Jeff Sutherland, em seu livro

"*SCRUM*: a arte de fazer o dobro do trabalho na metade do tempo", demonstra vários casos de fracasso.

Com isso, demostramos na figura 1 o passo a passo da aplicação do scrum, no qual foi reunido e organizado no *sprint*.


**Figura 1.** Scrum**.**

**Fonte:** Wrike, 2022.

Sendo assim, organizamos as etapas dos *sprints* em *Product Backlog, Sprint Backlog, Planning, Implementation, Review, Retrospective, Daily Scrum* e *Completed Product.*

- *Product Backlog:* é uma cronologia sistemática de todos os requisitos que se tem noção de que necessitam estar no produto. Assim, ele fica em constante mudança e atualizações.
- *Sprint Backlog:* inicialmente é uma lista de atividades a ser desenvolvida durante os sprints.
- *Planning:*é um novo ciclo de execução, no qual prioriza as atividades que precisam ser estreadas de início.
- *Implementation: a etapa que consiste em implementar todos as modificações.*
- *Review:* ponto que, faz toda a revisão do produto e geralmente acontece nos últimos encontros da reunião.
- *Retrospective:* uma retrospectiva do sprint onde debatem e identificam para descrever o que pode ser melhorado.
- *Daily Scrum:* uma reunião diária para a vez o andamento do processo e tirar os negativos que estão fazendo com que o processo esteja antando devagar.
- *Completed Product*: o produto concluído com as modificações e implementos dos sprints ao decorrer das 4 semanas.

## 2.2 Árvore de Problema

A árvore de problema é um meio que determina o foco e a intervenção, que tem como objetivo uma metáfora [...]. mostrando uma ilustração gráfica com situação-pro-

blematização esta que é representada pelo tronco, e as raízes são as causas e os efeitos desfavoráveis que provocam na população alvo que representam as folhagens e galhos.

A comparação da árvore contribui na visibilidade das etapas de construção dessas ferramentas/instrumentos, embora não se utilize a representação gráfica da árvore, propriamente dita, porém, a sua estruturação é primordial para criação um organograma (BUVINICH, 1999).

Segundo Coral, Ogliari e Abreu (2009, p.109), ele afirma que a árvore de problemas tem a finalidade de identificar o fator e resultados (causa e problema), adequando-se a desenvolver a uma solução do problema.

## 2.3 *Personas*

São modelos de usuários inventados a partir de uma síntese das decorrências de uma observação realizada com um mercado-alvo. Esses modelos podem ser criados sem ferramentas auxiliares, ou baseando-se em diferentes técnicas de modelagem, como Mapa de Empatia e Perfil. Personas auxiliam na licitação de requisitos de forma ágil e no desenvolvimento de uma experiência de uso agradável ao público (MARQUES et al., 2018).

A persona em si, é uma interpretação fictícia do seu cliente ideal. É baseado em informações reais sobre a conduta e a demografia de seus clientes. Personifica também a edificação de um perfil pessoal, necessidades, objetivos, preocupações.

Um bom estabelecimento de persona é se conectar com seu público-alvo. Assim, uma simples análise pode desvendar as características gerais dos potenciais clientes.

## 2.4 Mapa de empatia

Este mapa de compaixão é uma ferramenta para ajudá-lo a entender visualmente cada segmento de cliente. Essa ferramenta faz suposições claros sobre as necessidades comportamentos e outros atributos do cliente (OSTERWALDER; PIGNEUR, 2010).

> Tem campos e ajuda a entender o modelo principal. Um arquétipo de cliente é uma representação de quem é o cliente e quais são suas características, portanto sua definição afeta diferentes aspectos do modelo de negócios, incluindo, além do segmento de clientes, outros blocos do Business Model Canvas, como canais de relacionamento com o cliente e modelo de renda (BLANK; DORF, 2014).

Como tal, é uma estratégia fundamental para as empresas que procuram melhorar a experiência do consumidor aumentar a satisfação do cliente e oferecer os produtos serviços e cuidados que merecem e precisam.

## 2.5 Matriz CSD (Certezas, Suposição e Dúvidas)

Depois de todo esse processo, passamos a utilizar a ferramenta da matriz CSD (Certezas, Suposição e Dúvidas), no qual segundo Ferreira (2021), a matriz CSD é uma ferramenta que visa atrelar assuntos, organizar informações e guiar a construção de um projeto já em exploração de suposições.

Essa ferramenta permite alinhar as certezas do já sabemos junto com as suposições das incertezas que precisaremos ter uma confirmação e as dúvidas das descobertas dos desafios que nossos clientes estudados passam para assim, termos uma melhor qualificação.

## 2.6 TAM, SAM e SOM

Segundo Toledo (2019), alerta sobre a importância de entender se o segmento e o mercado em que uma startup pretende atuar são grandes o suficiente.

Sequoia (2021), indica o uso de três métricas, para avaliar o mercado e o segmento em que uma startup está inserida. Essas métricas são TAM (*Total Addressable Market*), SAM (*Serviceable Addressable market)* e SOM (*serviceable Obtainable Market*)

TAM ou mercado total disponível ou mercado total representa "a demanda total do mercado por um produto ou serviço" SAM ou mercado passível de manutenção ou mercado endereçável é a parcela que uma empresa "realmente tem potencial para alcançar nos próximos anos" SOM ou mercado passível de manutenção ou Mercado Acessível é a parte do SAM que pode ser conquistada dentro de uma "previsão de aquisição realista".

## 2.7 Mercado de Negócio B2B

Gomes (2021), afirma que B2B significar "Business to business" em que referir empresas vendem bens mercadoria serviços para outras empresas. Em outras palavras podemos definir o que é B2B como um modelo de negócios empresa, a empresa onde um é fornecedor e o outro é cliente.

Nos mercados B2B, geralmente não há ponto de venda. O que existe é a construção do relacionamento e a agregação de valor ao produto ou serviço. As vendas geralmente são feitas diretamente por meio de representantes de vendas.

## 2.8 Canvas e proposta de valor

Osterwalder et al. (2014), concebem a proposta de valor por meio de "uma ferramenta que une dois pilares: conhecimento sobre o perfil do cliente com um mapa de valor – no qual é descrito o processo de criação de valor para aquele cliente específico". O conceito inerente à ferramenta está associado ao adequado alinhamento entre a oferta de produtos e serviços de uma empresa com as expectativas e necessidades do consumidor.

O canvas e proposta de valor é um implemento para ajudar a conceber e posicionar propostas de alto valor para produtos ou serviços em torno do que o cliente precisa e anseia. Ou seja, é uma ferramenta organizacional que permite aos empreendedores construir soluções baseadas nas necessidades dos clientes. Aliás, esse método é muito congenial para testar empreendedores digitais de forma mais visual.

## 2.9 Product/Market Fit (PMF)

Segundo Dennehy et al. (2016), "*o product/market* fit representa mais do que apenas o momento em que a solução oferecida ao cliente condiz com o problema identificado." Alcançar o ajuste significa que o mercado está finalmente identificado com certeza, o

modelo de negócios está validado, a tração foi alcançada e o empreendimento está pronto para ser escalado em uma base maior.

Na rota de um empreendimento para o mercado, o ajuste do *product/market fit* constitui um marco no qual o foco muda do aprendizado por meio de pivôs para o crescimento e otimização (MAURYA, 2012).

> A literatura existente baseia-se principalmente em estruturas que mudam iterativamente entre o MVP e o ajuste do produto/mercado até que três critérios centrados no ser humano sobrepostos sejam atendidos: viabilidade, viabilidade e usabilidade/desejabilidade do mercado (DENNEHY et al., 2016).

O verdadeiro desafio em alcançar e validar o *product/market fit* é que não é um momento discreto e absoluto, mas um processo contínuo que precisa ser testado de forma iterativa. A utilidade de monitorar a rota adequada depende inteiramente das métricas implementadas para medi-la.

## 2.10 Produto Mínimo Viável (MVP)

MVP denota produto mínimo viável. Isso significa criar a versão mais simples e eficiente de um produto usando o mínimo de recursos possível para atingir a proposta de valor central da ideia. Você pode construir um MVP antes mesmo da apresentação de um produto. Portanto, uma vez que se tem um problema e um Produto Mínimo Viável definido, pode-se a interagir com o cliente apresentando e oferecendo a solução para seu problema. Neste ponto, testa-se o produto e mede-se a aceitação do mesmo, buscando o ajuste da solução ao problema (COOPER; VLASKOVITS, 2010). Ao final, o produto mínimo útil é uma solução simples. Para ver se sua ideia resolve o problema do público que você quer atingir.

A ideia principal é permitir que clientes em potencial usem versões inacabadas de seu produto ou serviço para avaliar o desempenho. Este mecanismo é amplamente utilizado. relacionado à resposta recebida do usuário. As iniciativas podem revisar muitas das suposições feitas durante o estágio de planejamento.

Outra perspectiva, é o sistema de inicialização pode analisar a interatividade entre o usuário e a solução e coletar informações de desempenho para implementá-la na versão final.

# 3. ESTUDO DE CASO

Neste tópico vamos abordar o desenvolvimento das etapas selecionadas pra a solução do problema citado. Dito isso, no item 3.1 abordaremos a descobertas e validações do problema, no item 3.2, mostraremos a etapa do modelo de negócio, no item 3.3 temos validação e execução pelo cliente.

## 3.1 Descoberta e validação do problema

Quando dizer respeito em gerenciar essa concepção da melhor forma possível e sem gerar grandes resulta a outros projetos, buscamos uma abordagem para gerenciamento desses projetos através da metodologia do *Scrum*. A implementação do *Scrum* no projeto

começou em agosto de 2022, quando toda a equipe inicialmente se concentrou em entender o método Scrum para identificar pontos positivos e negativos.

Primeiramente foi definido o *Scrum Master*, que era o coordenador do projeto da empresa aplicando as práticas e regras, em seguida foi escolhido o *Product Owner*, um dos diretores da empresa que é responsável por manter todas o trabalho em atraso em ordem, garantindo o os marcos são visíveis para todos os membros ,então  o que precisa ser priorizado primeiro,, no qual o Time de Desenvolvimento tomou ênfase pois times também são auto organizáveis fazendo que  gerasse um eficiência no desenvolvimento.

Na primeira reunião, foi apresentado o cenário passo a passo do *Scrum*, indicado o tema a ser utilizado durante o estudo e detalhados os *Sprints* como mostra na Figura 2. Nesta etapa, definiu-se as datas de início e término de cada atividade. Depois de explicar a corrida as atividades, começamos nas primeiras semanas planejando nossa árvore de problema junto com a persona, o mapa de empatia, e o CSD, em que, foi criado todo um cronograma para a divisão dessas etapas e, vendo quais seriam aplicadas com prioridades ao decorrer dos dias.

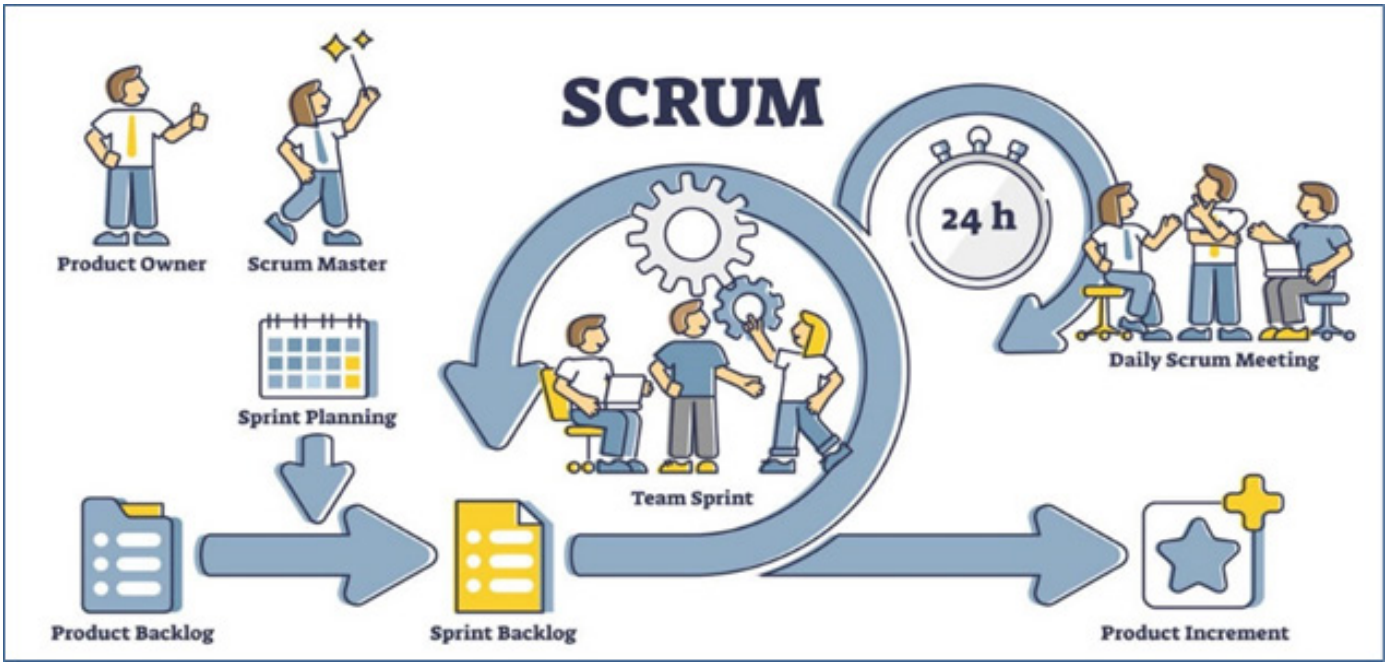



**Figura 2.** Scrum**.**

**Fonte:** GS1 Brasil, 2021.

O objetivo destas tarefas é gerenciar visualmente o progresso do projeto compartilhar informações e facilitar o entendimento durante as reuniões diárias com a equipe *Scrum*. Por questão de segurança as pessoas buscam mais conforto, comodidade e proteção, por isso, as pessoas optam em morar em condomínios. Pois, a maioria condomínios apresentam esses seguintes atrativos.

Ao criamos uma árvore de problemas, ela foi dividida em problema central que nos informa sobre a segurança que os condomínios enfrentam no decorrer do tempo.

As raízes da árvore são concebidas por causas que, sendo a segurança ser um fator questionável, devido à ausência de um controle mais ativo de entrada e saída dos visitantes dentro do condomínio, vem gerando mais um problemático em que cada causa a uma consequência, no qual também virar a falta de eficiência e sem contar o tempo perdido pra entrar. A Figura 3,

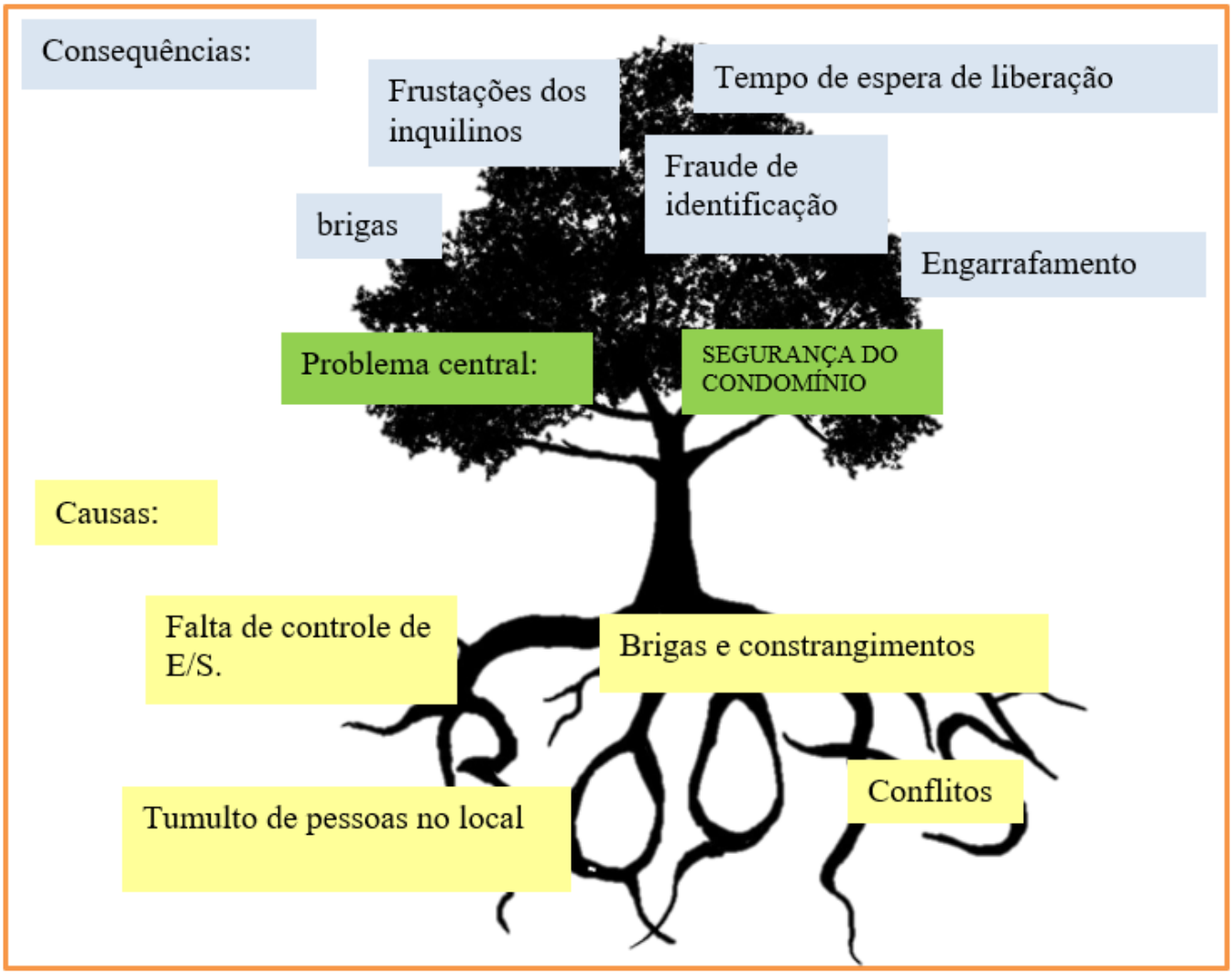



**Figura 3.** Árvore de problemas.

**Fonte:** Autoral, 2022.

Logo após, citamos a ferramenta persona, pois sem uma definição dela, é possível que alguns casos de estratégia acabam sendo frustrados, pois estão sendo ofertadas para os grupos errados, ou seja, um produto que foi proposto à classe A, está sendo mais evidenciado para a classe C.

Sendo assim, enumeramos alguns motivos que provam a importância dos questionamentos para seu negócio:

- Determinar o tipo de conteúdo que você precisa para atingir seus objetivos;
- Definir o tom e o estilo de seu conteúdo;
- Ajudar a desenhar suas estratégias de marketing apresentando o público que deve ser focado;
- Definir os tópicos sobre os quais você deve escrever;
- Entender onde os prospectos buscam suas informações e como eles querem consumi-las

Com isso, ao citarmos a ênfase da persona colocamos o processo do **Mapa da empatia** em jogo. Pois de maneira minuciosa, tem a concepção de apontar como o cliente/usuário comporta-se perante a precisão de um plausível atendimento mediante a liberação na portaria e a segurança que ela tem sobre possíveis entradas sem uma certa docu-

mentação como é apresentada na Figura 4.

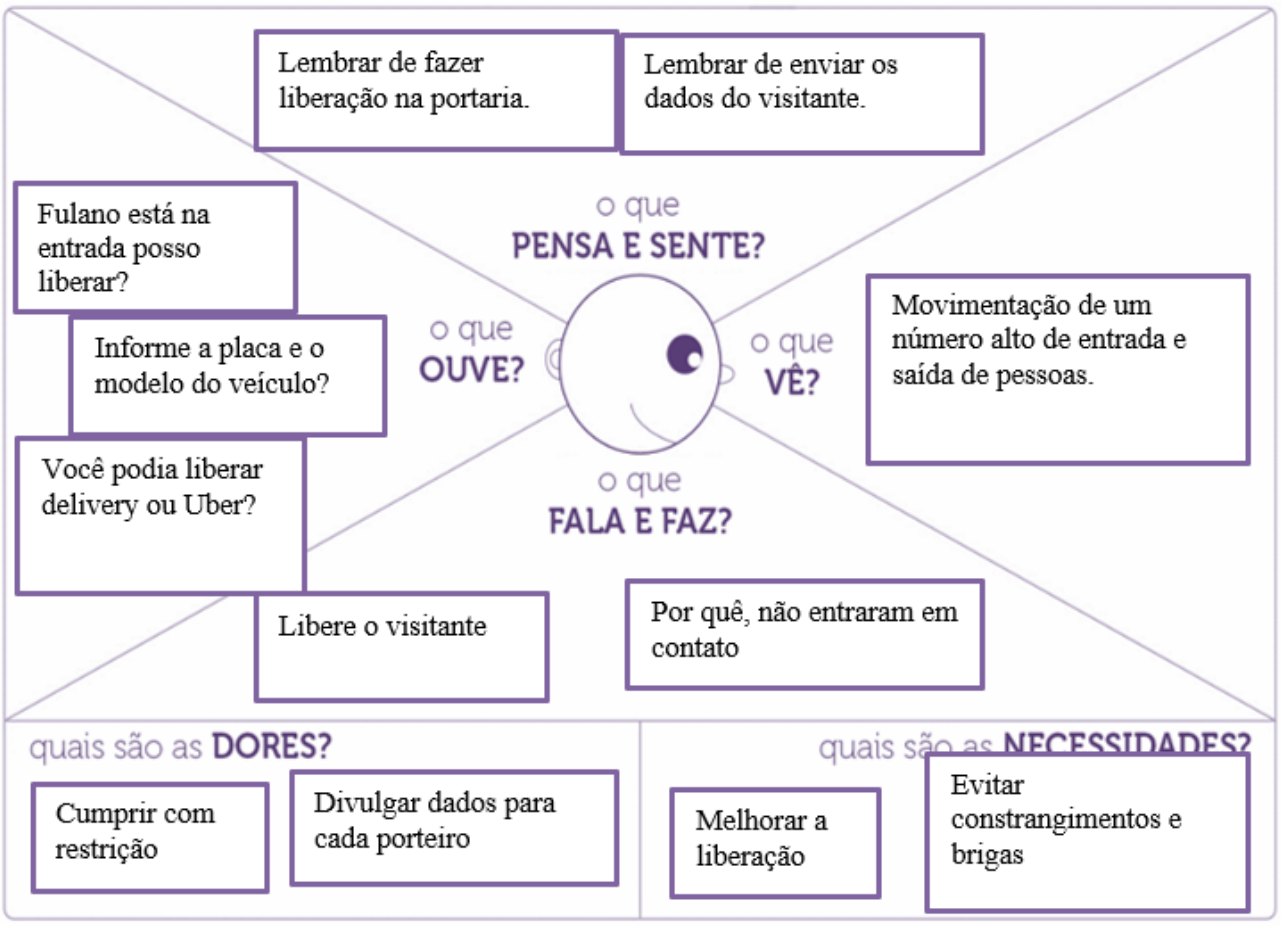



**Figura 4.** Mapa de empatia.

**Fonte:** Autoral, 2022.

Assim, avaliando suas dores, seus pensamentos e fazeres e também, quais seriam suas necessidades como mostra da figura 4, será possível tomarmos as devidas atitudes para solucionar as dificuldades utilizando o sistema de segurança pra identificar se a pessoa é quem diz ser, pois a apenas utilizando um suposto nome familiarizado elas poderiam entrar, que vem sendo uma das preocupações e atrasando a entrada nos condomínios e com essas informações busca fazer esses melhoramentos.

Com a assistência da **matriz CSD** (Certezas, Suposições e Dúvidas), podemos definir precisamente em que devemos focalizar e prestar nossos empenhos no projeto.

Deve ser ativo durante o desenvolvimento do projeto e mostrando as mudanças e os progressos posteriormente quando as dúvidas forem sanadas sobre a aplicação da segurança dos condomínios. A Figura 5, posteriormente mostrada, diz a respeito do que os clientes pensam ao decorrer do problema sobre o lugar onde residem e como gera a o csd.


**Figura 5.** Matriz CSD (Certezas, Suposições e Dúvidas).

**Fonte:** Autoral, 2022.

Analisando isso, obtivemos alguns aspectos das cefaleias de nossos clientes, para que, a partir do CSD, podemos chegar a resultados diferentes com cenários de como pode ser o processo de vida do indivíduo e como ele lida com a tentativa de se livrar desses problemas.

## 3.2 Modelo de Negócio

A partir da segunda semana fomos implementando o modelo de negócio, no qual aborda o *TAM, SAM e SOM*, em que, foi designado nas reuniões dos *sprints* a busca por informações do mercado que consistia em saber sobre o mercado brasileiro e colocando em evidência o crescimento ao decorrer dos tempos, para fazemos a análise sobre o mercado B2B se seria um bom empreendimento sobre essa aplicação com foco em condomínios.

Assim, como apresentado na figura 6, existe *sprints* aplicados, referentes a um País, a uma região e um lugar pra termos dados o suficiente pra tais análises de mercado.

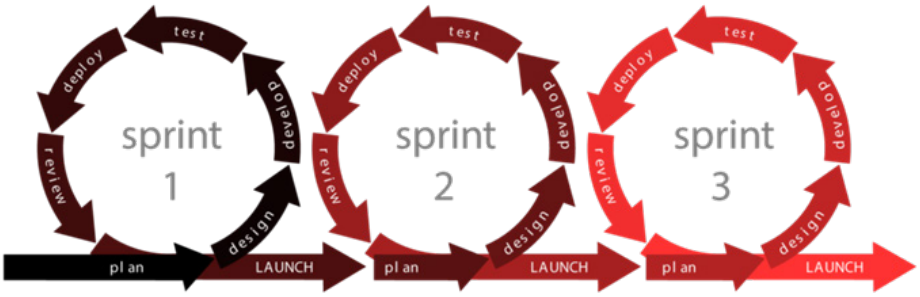



**Figura 6.** Sprints

**Fonte:** MEYE,2022.

Após fazermos o uso dessas ferramentas, buscamos informações para o mercado do **TAM, SAM e SOM** que de acordo com a pesquisa feita pelo infoimóveis, 7,8 milhões de novos apartamentos foram construídos no país nos últimos 35 anos. A apresentação da figura 7 é a quantidades de pessoas que residem em condomínios.

No período de 1984 a 2019, segundo o Instituto Brasileiro de Geografia e Estatística (IBGE) o número de prédios em grandes regiões cresceu 321%. São mais de 68 milhões de pessoas morando em condomínios no Brasil, que movimentam cerca de R$ 165 bilhões anualmente. A maior renda do país está entre os condôminos de casas e apartamentos: média de R$ 6.275.

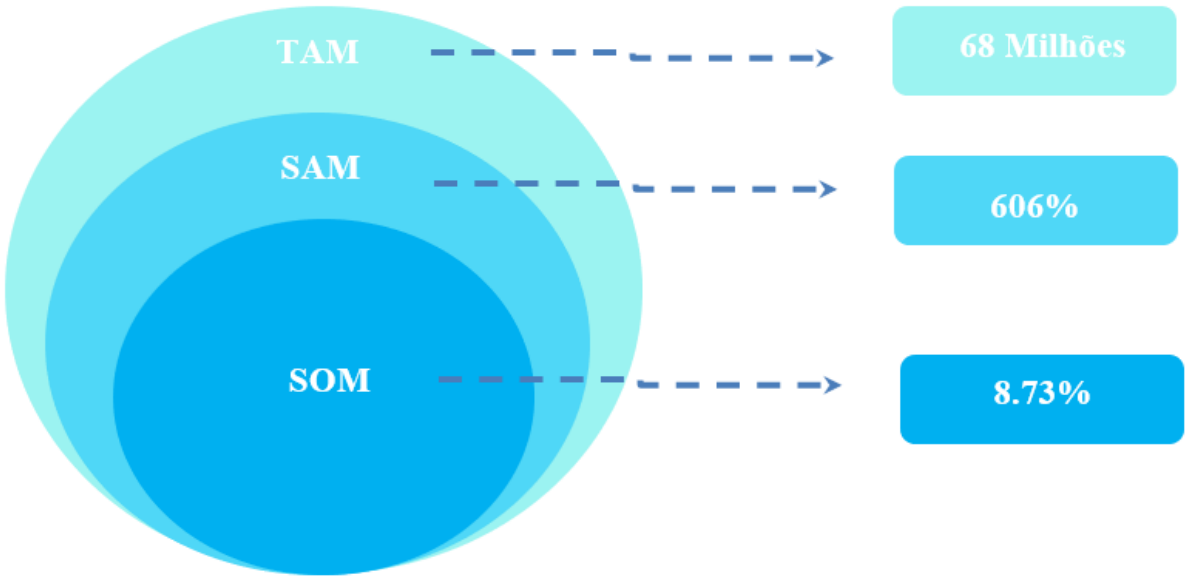



**Figura 7.** TAM, SAM e SOM

**Fonte:** Autoral, 2022.

Todos os dados sobre o crescimento no número de casas, apartamentos e outros tipos de domicílios no Brasil, com base nas pesquisas do IBGE, estão disponíveis no blog do Triider e podem ser acessados na íntegra.

Segundo os dados tirado da empresa Triider em 2021 Região Sudeste foi a que menos cresceu percentualmente no País, ficando atrás do Nordeste (606%), do Centro-Oeste (366,05%) e do Sul (360,92%), além do Norte em 35 anos.

Embora o Norte contabilize o maior crescimento percentual em 35 anos, por lá esse tipo de domicílio corresponde a 6,64% de todos os imóveis, ficando atrás do Nordeste (8,73%), do Sul (14,75%) e do Centro-Oeste (9,24%), além do Sudeste.

Por tanto com base nesses dados retirados do Triider, criamos o aplicativo Tranca voltado a segurança de condomínio, com início da implementação dentro do **mercado B2B**, onde consiste em uma empresa apresentar seu produto a outra empresa

## 3.3 Validação e execução pelo cliente

Na etapa final das reuniões com os integrantes do *Scrum* (figura 8) foram sobrepostas as modificações pra que a questão da problemática segurança se encaixasse dentro do mercado de negócio e por fim, nas 4 semanas fomos modificando e adequando dentro dos *sprints* outras metodologias do scrum, tais como canvas e proposta de valor que foram utilizados dias para fazer sua organização seus valores em relação mercado e seu público

alvo, o *Product/Market Fit* e produto Mínimo Viável.

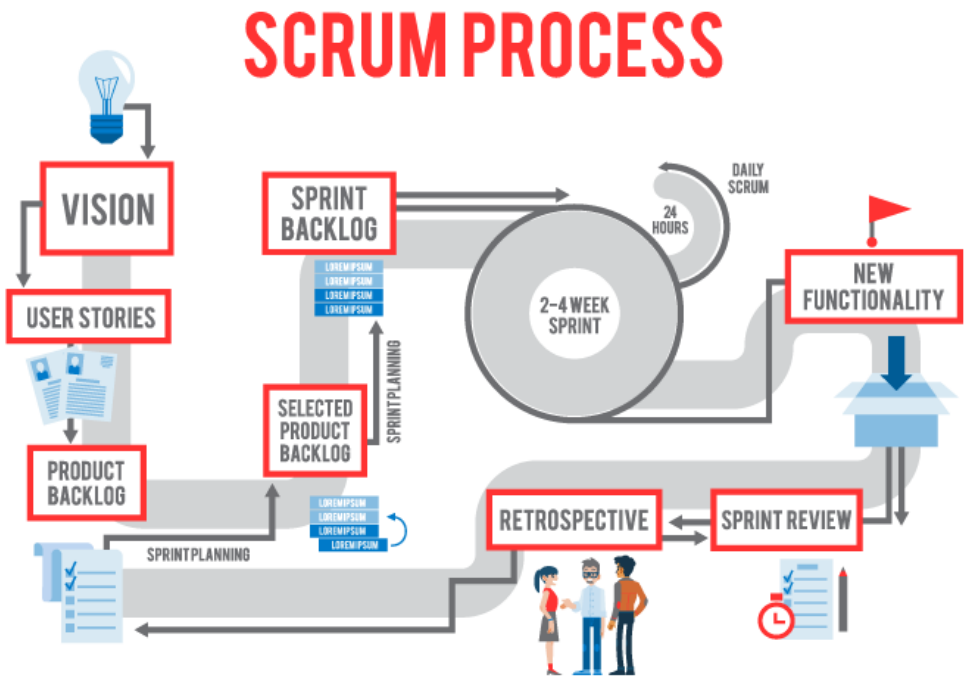



**Figura 8.** *Scrum Process.*

**Fonte:** MEYE, 2022.

Assim de início, prosseguindo com o uso da ferramenta para aprimorar nosso conhecimento dos requisitos do projeto foi o **Canvas de Proposta de Valor**. Cujo objetivo, visa ajudar o empreendedor a alinhar seus negócios de forma mais visual divididos por quadrantes às principais descrições de novos ou negócios existentes.

Ao criar o Canvas, perseguimos encorpar o máximo de lacunas que tivemos como incertezas sobre a interação empresa-consumidor. Por fim, a produção do Canvas não apenas auxilia a estruturar as concepções, mas também novas ideias, como é apresentada na Figura 9.

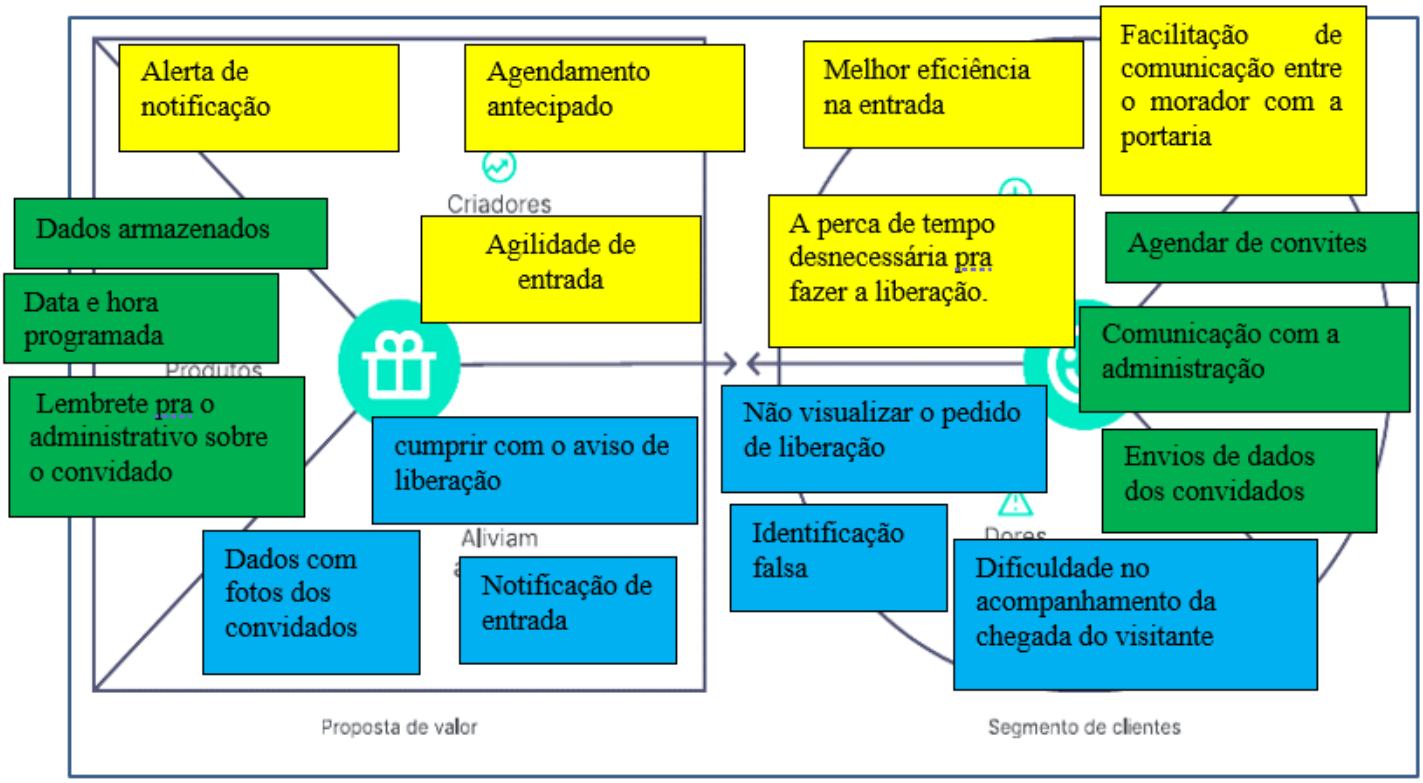



**Figura 9.** Canvas de Proposta de Valor.

**Fonte:** Autoral, 2022.

De forma rápida, utilizando todas as ferramentas acima para conhecer melhor nossos clientes, conseguimos ter maior certeza em alguns pontos, "como conhecer outros que não conhecemos", reforçando nossa prova de valor com eles, promovendo a importância de solucionar tais problemas com o **TRANCA**.

Tranca o aplicativo que aparece no mundo das startups, que busca uma solução para maior conforto e segurança, quer garantir a segurança dos moradores. Releve novos desejos e necessidades da sua alcova de clientes em potencial. Daí a proeminência de sempre colocá-lo à frente e no centro do trabalho

Uma das metodologias utilizadas para tal objetivo é o **Product/Market Fit (PMF**) em que adequação do produto ao marketplace como é mostra na Figura 10.

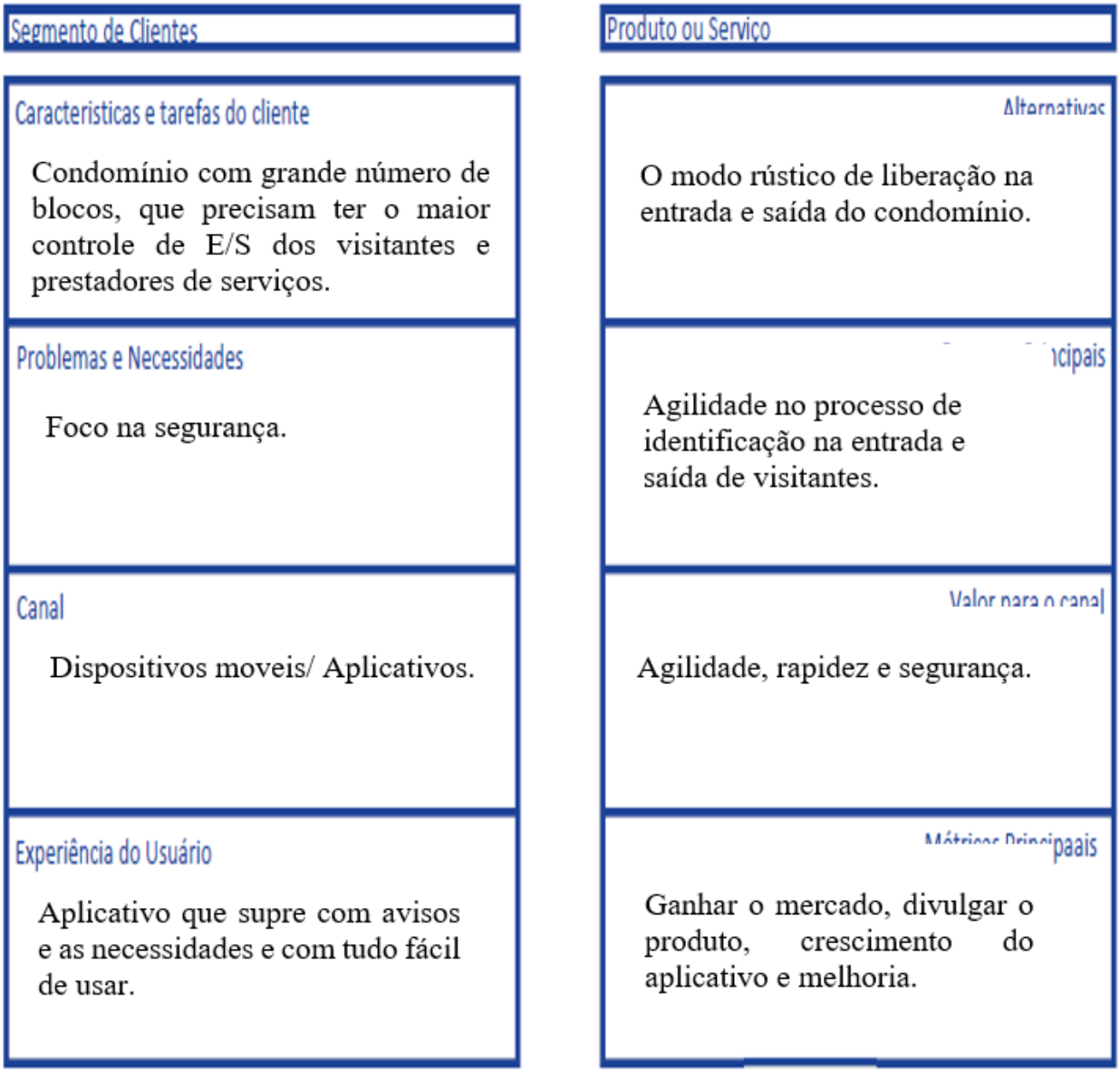



**Figura 10.** Quadro De Encaixe Do Produto No Mercado

**Fonte:** Autoral, 2022.

Assim, podemos dizer que o *Product Market Fit* é uma estratégia utilizada para avaliar e verificar se uma solução é compatível com os problemas e necessidades de um público específico. Na prática também é usado para analisar se existe um mercado com boas perspectivas ou não.

Outra metodologia que aplicamos para melhor visualização foi o **Produto Mínimo Viável (MVP)** que, a partir de uma versão simplificada, é possível validar através da aplicação das telas que foram desenvolvidas, e com isso, é utilizado para mensurar e validar a recepção do público e o potencial de uma determinada ideia gerar retorno financeiro.

O conceito de MVP tem sido utilizado em serviços de segurança para condomínios, onde foi muito útil para a empresa validar uma ideia antes de investir muitos recursos através do feedback de testadores, no qual introduzimos uma versão que não está totalmente preenchida em uma amostra ao público-alvo para testar como é recebida a proposta de valor deste protótipo inicial.

Com base nesse *feedback*, são feitos os ajustes necessários antes que a versão final seja lançada no mercado. Apresentada na Figura 11.


**Figura 11 -** Protótipo inicial

**Fonte:** Autoral, 2022.

Assim foi criada, depois que adquirimos desejos e ideias dos clientes as adaptaçoes foram postas pra mostrar que alem de ser um trabalho pequeno, mas que pode se tornar algo grande atraves de adaptaçoes feitas ao longo dessa tragetoria. Um exemplo disso que foi apresentado na figura 12, foi a adaptação das notificações no aplicativo, no qual iria informar para o morador e pra o porteiro o de designo do visitante.

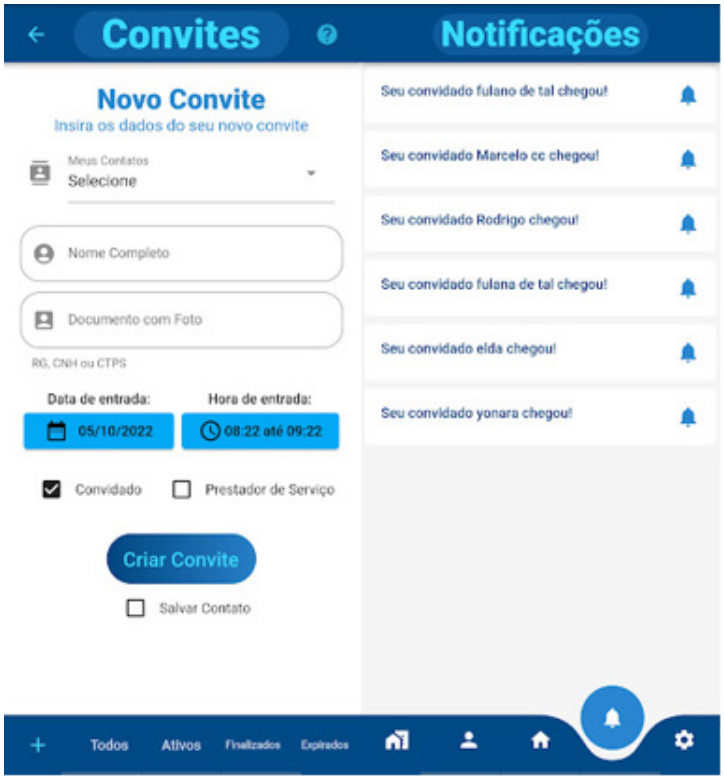



**Figura 12-** Protótipo de notificação.

**Fonte:** autoral, 2022.

A ideia principal é permitir que clientes em potencial usem versões inacabadas de seu produto ou serviço para avaliar o desempenho. Este mecanismo é amplamente utilizado. relacionado à resposta recebida do usuário. As iniciativas podem revisar muitas das suposições feitas durante o estágio de planejamento.

## 4. RESULTADOS E DISCURSÃO

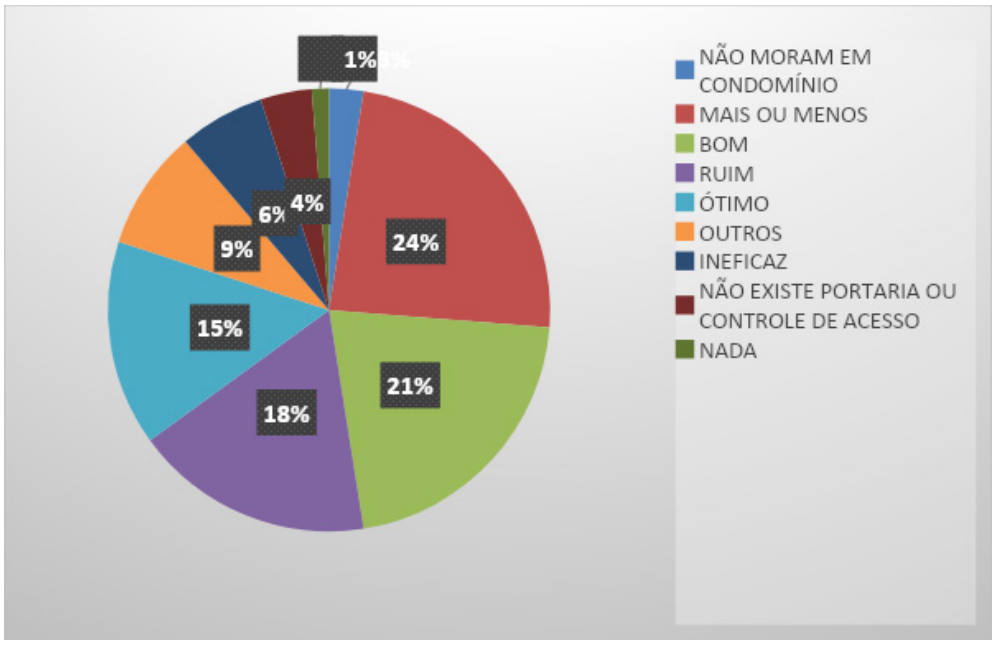
Um condomínio é definido por um espaço que se divide por diversos proprietários/moradores, que compartilham também áreas em comum, cada morador tem seu apartamento privativo e as áreas em comum de acordo com as especificações que foram ditas no momento da compra ou do aluguel do imóvel.

Assim como no problema central que foi mostrado na árvore de problemas, o importante aqui são as estratégias que a gestão de um condômino desenvolve para a melhor segurança dos moradores. Por esse motivo a criação do aplicativo **TRANCA** é muito importante. Para a melhor rapidez no processo de identificação, liberação da entrada e saída de visitantes no condomínio, para mais eficiência na reserva das áreas externas, para melhor desenvolvimento da portaria.

No qual o gráfico 1, apresenta os dados obtidos através da primeira pergunta do questionário feito pelo google forms, desenvolve um problema: "Os condomínios são estruturas de moradias que necessitam de uma gestão de qualidade, pois além dos interesses particulares, também é de extrema importância o bem-estar, os interesses e necessidades de todos os moradores do condomínio. **O que você acha sobre o sistema de entrada e saída de visitantes no seu condomínio?**"

Como apresentado no Gráfico 1, podemos saber o valor em porcentagem das entrevistas feitas em diversos condomínios da cidade de São Luís. Cerca de 21% acham que entrada e saída do seu condomínio é boa, mas como o cliente procura sempre qualidade eles afirmam que seria interessante ter essa aplicação, cerca de mais de 79% dos entrevistados desejariam tem uma entrada com mais eficiência e agilidade, mesmo tento uma boa perspectiva de seu condomínio.

**Gráfico 1.** Primeira pergunta do forms.




**Fonte:** Autoral, 2022.

Comentários do público-alvo:

*1- Acho que seria interessante integrá-los, visto que, o condomínio em que resido não possui um sistema que realize essa comunicação entre portaria e condômino.*

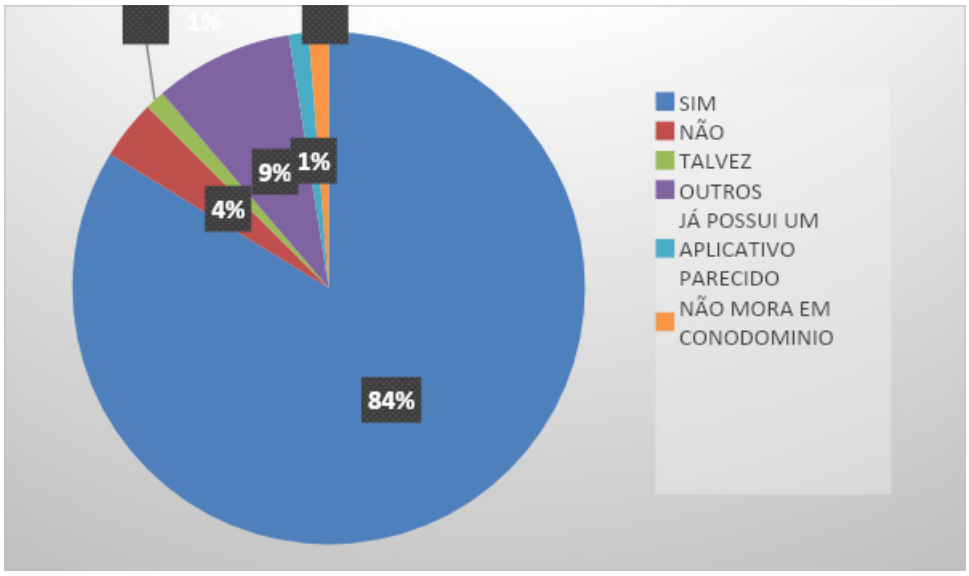
*2- Acho muito boa porque dessa forma não entra qualquer pessoa sem se identificar para saber qual a finalidade que a mesma veio fazer ou que estar querendo.*

Segunda pergunta feita ao público alvo: "O morador pode criar um convite para o visitante no aplicativo, convite esse que facilitaria o processo de entrada de cada visitante, que aumentaria bastante a segurança e o controle do condomínio. Ao entrar em um condomínio, o que você acha sobre a eficácia da portaria? Por exemplo, a velocidade em que você é atendido e é autorizado a entrar. A criação de convites seria útil para você?"

**Quais as opiniões do público alvo com as soluções dadas pelo aplicativo TRANCA? O aplicativo TRANCA seria útil para o público-alvo?**

Como apresentado no Gráfico 2, podemos saber o valor em porcentagem. Cerca de 84% afirmam que, possuem um aplicativo parecido, mas o que torna interessante o aplicativo e sua funcionalidade é a questão de notificar o morador quando seu visitante adentra dentro do seu condomínio. Em se tratar de não eficiência menos de 16 % acha que seria, apenas mais um entre outros, com isso a proposta que foi apresentada na questão posterior mostra outros valores de visão ao público alvo.

**Gráfico 2.** segunda pergunta do Forms




**Fonte:** Autoral, 2022.

Comentários do público alvo:

*1-Seria bem melhor, pois quando se visita pessoas que moram em condomínios, precisarmos esperar alguns minutos para que a pessoa seja avisada da nossa chegada. Porém, com o convite já teríamos um acesso, mais rápido ao ambiente interno do condomínio e o morador a segurança e controle de entradas e saídas de visitantes.*

*2- Acho que seria muito bom para melhoria da segurança dos moradores do condomínio e agilidade da portaria. Sobre a eficácia da portaria está ótima estão desenvolvendo um trabalho perfeito. Seria ótimo o convite com tanto se houvesse um código ou senha entre o agente de portaria e morador do condomínio.*

*3- Muito demorado, pois o porteiro tem que interfonar para o morador e as vezes o interfone não funciona direito. A criação com convite ia facilitar muito a vida do porteiro e*

*do morador, uma vez que todas as informações estão ali no aplicativo.*

*4- Geralmente a entrada nos condomínios é lenta, um tanto "burocrática", um convite por meio do aplicativo facilitaria e tornaria mais ágil o processo.*

*5- Sim, seria muito útil, já que as vezes se perde muito tempo na entrada, esperando a confirmação de entrada, sendo até inconveniente com a visita.*

A terceira pergunta feita ao público alvo: "Após a chegada do visitante o morador em questão receberá uma notificação do aplicativo para avisar que o seu convidado já chegou. **Você como morador, acha importante, o uso desse aplicativo para melhor funcionalidade segurança do seu condomínio?"**

Como apresentado no Gráfico 3, Cerca de 97% afirmam que, seria interessante que o condomínio em que reside tivesse um pouco mais de eficiência e segurando na questão de entrada e os 3 % comentam que a ineficiência seria com o de todos, pois apresentaria obstinação ou que ate mesmo não iria suprir com as suas necessidades, mas também afirmam com dúvidas que seria sim um ótimo aplicativo de segurança, mas que consistisse sempre com a ideologia da proposta.

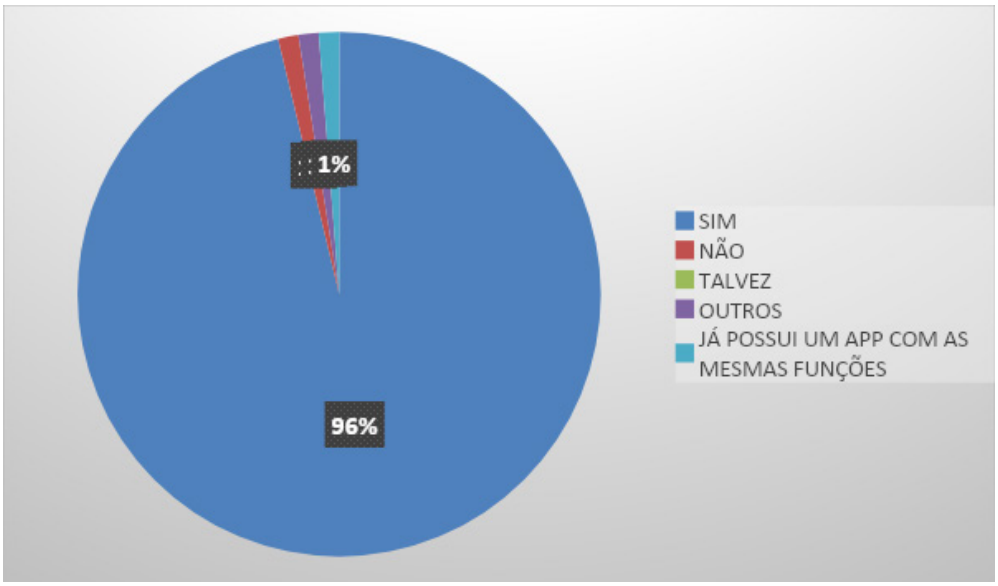
Comentários do público-alvo:

*1 - Sim, pois através das notificações apresentadas pelo aplicativo, é possível ter maior controle de quem entra e sai no condomínio, e até mesmo oportuniza saber quando isso ocorreu, caso seja necessário consultar posteriormente data e horário da visita.*

*2 - Muito bom, pois fica registrado todas as entradas do condomínio e caso tenha alguma fatalidade facilitaria saber o culpado.*

*3 - Sim, tornaria o processo mais seguro e mais rápido, fora que seria mais prático para mim.*

**Gráfico 3.** terceira pergunta do Forms



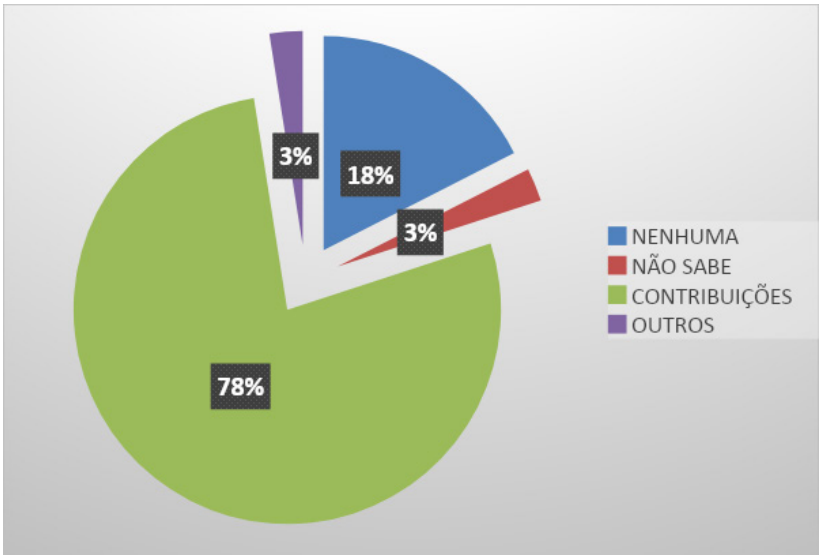

**Fonte:** Autoral, 2022.

A quarta e última pergunta feita ao público alvo: **Se você pudesse dar alguma contribuição para a melhoria do aplicativo, qual seria?**

Assim como demostrado no Gráfico 4, podemos saber o valor em porcentagem em sentido positivo, que cerca de 78% mostram ideia relevantes pra implementações futuras sobre a aplicação do aplicativo dentro de um condomínio, tanto que também consiste que

22% dizem que o aplicativo pra uma versão inicial de implementação está de bom agrado, já que no mínimo avisa ao morador que o visitante adentrou o condomínio.

**Gráfico 4.** Quarta pergunta do Forms




**Fonte:** Autoral, 2022.

Comentários do público alvo:

*1- As perguntas acima deixam apenas 2 itens claros, no caso convite e notificação. Mas de sugestão seria quantidade limitada de aplicativos por morador, por exemplo 2 aplicativos para cada apartamento. Poderia ter uma opção para fazer reserva da área de lazer.*

*2- Uso de uma senha entre o morador e o visitante que apenas os dois saibam e que seria solicitada quando o visitante chegasse, para que em caso de furto ou roubo do celular, quem cometeu o crime não tenha acesso ao condomínio*

*3- Maior interatividade com os outros moradores e uma aba dedicada a reclamações e/ou denúncias que possam ser enviadas diretamente ao síndico*

Assim, através de uma demonstração de uma pesquisa em São Luís sobre alguns condomínios e os seus residentes é notável que a falta segurança em seus condomínios é bem evidente, que com base nessas informações foi desenvolvido o aplicativo tranca, que traz como contribuição diversas contribuições, tais como o armazenamento de dados de visitantes para que não haja fraude de identificação, a facilidade de agendamento e liberação na entrada, os transtornos futuros que teriam que passar pra liberar já que a maioria das portarias demoram pra dá tais retornos ou fazer a liberação por falta de dados , já que o próprio visitante não sabe a localidade do bloco e assim, a diversas outras situações que o aplicativo com bases futuras e atuais iria ser adaptada e implementada pra ter seu próprio diferencial.

## 5. CONCLUSÃO

O objetivo deste trabalho foi descrever e avaliar o procedimento do avanço do modelo de negócios de uma *startup*, reconhecer e decompor as etapas e ferramentas adotadas em seu procedimento de criação, as percepções dos clientes sobre esse processo e a proposta de negócio final da *startup*. Os resultados aqui relatados mostram que os propósitos da pesquisa de opinião foram alcançados, pois foi possível visualizar o processo de

desenvolvimento de negócios realizado pela *startup* de forma estruturada, identificando duas principais diferenças desse processo em relação ao processo teórico proposto pelo modelo desenvolvimento do cliente.

Dessa forma pode-se confirmar que a *startup*, mesmo na primeira fase de concepção, já demonstrou benefícios ao aplicar um processo estruturado para o desenvolvimento de seu modelo de negócios. Uma vez que esta pesquisa se limitou à primeira etapa do modelo de desenvolvimento do cliente como sugestão para designações futuras, propõe-se continuar a análise das seguintes etapas: validação do cliente, geração de necessidades, estruturação do negócio e matriz de concorrência. Com base nisso, propõe-se construir uma avaliação final do modelo.

## Referências


BLANK, Steve e DORF, Bob. **Startup Manual do Empreendedor**. 1º Edição. Rio de Janeiro: Alta Books 2014.

BUVINICH, M. R. Ferramentas para o monitoramento e avaliação de programas e projetos sociais. **Cadernos de Políticas Sociais, série Documentos para Discussão**, n.10,1999.

COOPER, Brant; VLASKOVITS, Patrick. The Entrepreneur's Guide to Customer Development. Paperback, 2010.

CORAL, Eliza; OGLIARI, André; ABREU, Aline França de. **Gestão integrada da inovação:** estratégia, organização e desenvolvimento de produtos. 1.ed. São Paulo: Atlas, 2009.

DENNEHY, D., KASRAIAN L., O'RAGHALLAIGH, P., CONBOY, K., (2016), "Product market fit frameworks for lean product development", in R&D Management Conference 2016, "From Science to Society: Innovation and Value Creation." 3–6 July 2016, Cambridge, UK.

EJCM, Marketing. **Conheça os tipos de MVP mais utilizados por startups**. [*S. l.*], 20 ago. 2021. Disponível em: encurtador.com.br/dgEHT. Acesso em: 28 out. 2022. Disponível em: https://www.triider.com.br/blog/numero-de-apartamentos-no-brasil-cresce/. Acesso em: 25 out. 2022.

FERREIRA, E. (2021). **Matriz CSD:** O que é e como utilizar na sua estratégia? https://www.weme.com.br/blog/matriz-csd. Acesso em: 05 out. 2022.

GOMES, Gustavo. **O que é B2B: Tudo o que você precisa saber sobre o modelo de negócio business to business**. [*S. l.*], 2021. Disponível em: https://www.agendor.com.br/blog/o-que-e-2b/#:~:text=Business%20to%20business., a%20outra%20%C3%A9%20o%20cliente. Acesso em: 24 out. 2022.

GRAHAM, Paul, co-funded da Y Combinator. **Startup = Growth, want to start a Startup?**, set. 2012. Disponível em: <http://www.paulgraham.com/growth.html/>. Acesso em: 18 ago. 2022.

GS1 Brasil. **Scrum:** veja como usar essa abordagem ágil nos negócios. [*S. l.*], 30 set. 2021. Disponível em: https://noticias.gs1br.org/scrum-veja-como-usar-abordagem-nos-negocios/. Acesso em: 2 nov. 2022.

INFOIMÓVEIS (Brasil). **Condomínios movimentam R$ 165 bilhões por ano no Brasil**. [*S. l.*], 20 set. 2021. Disponível em: https://www.infoimoveis.com.br/noticia/condominios-movimentam-r-165-bilhoes-por-ano-no-brasil/2626. Acesso em: 24 out. 2022.

MARQUES, A. B., FIGUEIREDO, R., et al (2018). **Usabilidade e agilidade combinam?** investigando a adoção da modelagem de usabilidade em um projeto de software ágil na indústria.2018, Nova York, NY, EUA.

MAURYA, A. (2012), **Running Lean** (2nd ed.). Sebastopol, CA: O'Reilly Media, Inc.

MEYE, José. **Como o Scrum é organizado?**. [*S. l.*], 2 out. 2022. Disponível em: https://santexgroup.com/blog/how-is-scrum-organized/. Acesso em: 14 nov. 2022.

OSTERWALDER, A., PIGNEUR, Y., BERNARDA, G., SMITH, A., & PAPADAKOS, T. (2014). **Value proposition design:** Como criar produtos que clientes desejam. Hoboken: John Wiley & Sons.

OSTERWALDER, Alexander; PIGNEUR, Yves. **Business Model Generation**. Hoboken: John Wiley & Sons, Inc, 2010.

RIES, Eric. **A Startup Enxuta**. 1º Edição. São Paulo: Lua de Papel, 2012.

SCHWABER, Ken; SUTHERLAND, Jeff. **O scrum guide™ o guia definitivo para o scrum:** as regras do jogo. Recuperado de https://www. academia ágil, scrum-guide-us. pdf, 2017.

SEQUOIA CAPITAL. (2021). **Sequoia - Measuring Product Health**. Sequoia Capital. Recuperado 115 em 08 de março de 2022, de https://www.sequoiacap.cn/china/en/article/measuringproduct-health/.

STARTUP: Transformando ideias em Negócios de Sucesso. **Revista Científica Multidisciplinar Núcleo do Conhecimento. 2016**, v. 1, p. 739-753.

SUTHERLAND, Jeff, **SCRUM:** A Arte de Fazer o Dobro do Trabalho na Metade do Tempo. LeYa,2016.

TOLEDO, Marcelo. (2019). **TAM/SAM/SOM:** qual o tamanho do mercado da sua startup?

TORRES, Joaquim. **O Guia da Startup**. 1º Edição. São Paulo: Casa do Código, 2012.

TRIIDER. **Número De Apartamentos No Brasil Cresce**. [*S. l.*], 25 ago. 2021. Disponível

WRIKE, **Guia para Scrum Sprints**. [S. l.], 2022. Disponível em: https://www.wrike.com/scrum-guide/scrum-sprints/. Acesso em: 11 nov. 2022.

# 2

## ANÁLISE DA UTILIZAÇÃO DO FRAMEWORK SPRING EM SISTEMA WEB PARA O APLICATIVO DE GERENCIAMENTO DE CONDOMÍNIO

*ANALYSIS OF THE USE OF THE SPRING FRAMEWORK IN A WEB SYSTEM FOR THE CONDOMINIUM MANAGEMENT APPLICATION*

Carlos Eduardo Júnior Ferreira[1]

Edilson Carlos Silva Lima[2]

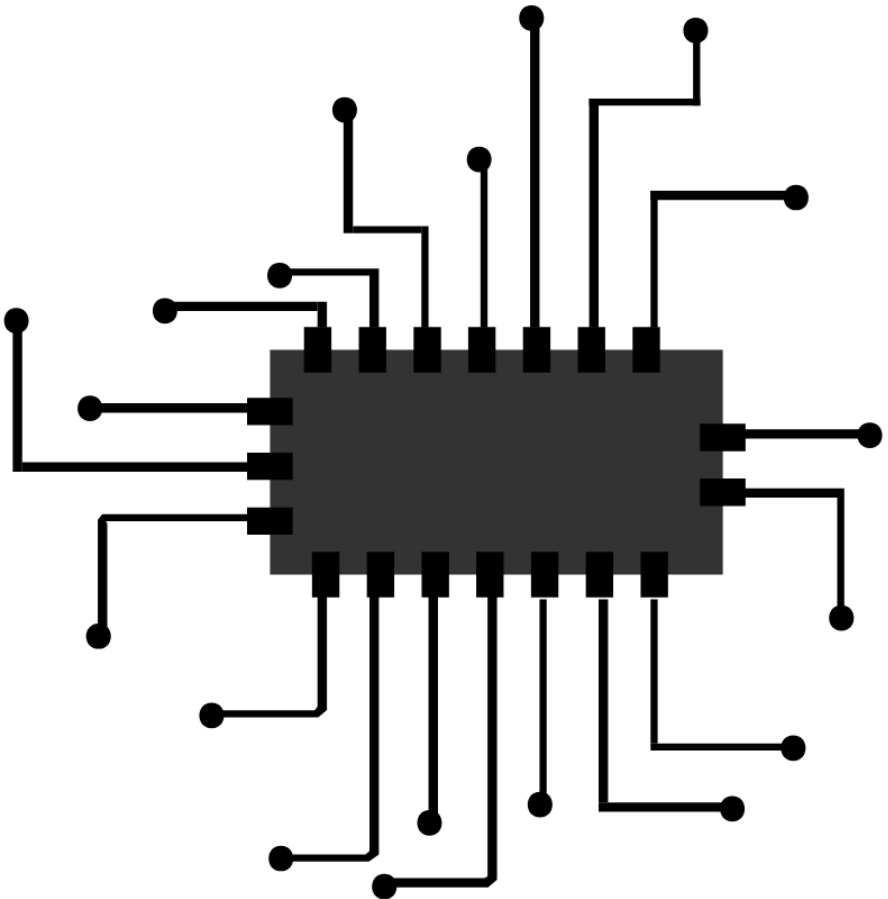
Will Ribamar Mendes Almeida[3]

[1]Engenharia da Computação – Universidade Ceuma (UNICEUMA)– São Luís – MA – Brasil

[2]Engenharia da Computação – Universidade Ceuma (UNICEUMA)– São Luís – MA – Brasil

[3]Engenharia da Computação – Universidade Ceuma (UNICEUMA)– São Luís – MA – Brasil

{FERREIRA, Carlos Eduardo Junior, ceduardoferreiraeng@gmail.com; LIMA, Edilson Carlos Silva, edilsonlima3@gmail.com; ALMEIDA, Will Ribamar Mendes, will.mendes@ceuma.com.br }



**d.o.i.:**

## Resumo

O presente artigo tem como objetivo o desenvolvimento de uma API REST utilizando um Framework Java (Spring) para a criação de um gerenciamento de condomínio. O alvo é ajudar os moradores e funcionários a ter uma harmonia e facilidade na utilização dos serviços como de agendamento de espaços, convites e entre outros serviços que existem no condomínio em si. O sistema contará com criação de usuário de um condomínio, e de condomínios que utilizar a aplicação, espaços do condomínio e também tome de conta os problemas e ações que ocorrer cotidianamente em um condomínio. Tendo essas situações abordadas serão explicadas detalhadamente com na implementação, os dados obtidos no resultado e o que foi revelado através desses dados criando o contexto para a conclusão da pesquisa.


**Palavras-chaves**: Spring, API REST, Java, implementação, gerenciamento


## Abstract

This article aims to develop a REST API using a Java Framework (Spring), in the creation of a condominium management. The aim is to help residents and employees to have harmony and ease in the use of services such as scheduling spaces, invitations and among other services that exist in the condominium itself. The system will have the creation of a condominium user, and condominiums that use the application, condominium spaces and also take care of the problems and actions that occur daily in a condominium. Having these situations addressed will be explained in detail with the implementation, the data obtained in the result and what was revealed through these data creating the context for the conclusion of the research.


**Keywords:** Spring, API Rest, java, implementation, management.

# 1. INTRODUÇÃO

No Brasil, os condomínios residências há um déficit na área de comunicação, entre os moradores e o gerenciador do condomínio. com isso, existe uma problemática que dificulta o acesso, gerenciamento de locais de uso comum (área de lazer, esportiva e etc), segurança e serviços prestados por terceiros. A questão, é. Como se faz para resolver esses problemas? Quais métodos adotasse, pelo fator de ter menos dificuldades, e pelo fato de custo mais acessíveis. Pode ser resolvida? Essas questões fizeram com que esse projeto fosse iniciado. Sendo focado para condomínios, de todos os tipos, apresentando as soluções para resolver esse problema.  E com isso essas empresas buscam no mercado alguns softwares para gerenciar todas as situações que pode ter em um condomínio.

Visando isso a solução para resolver esse problema foi através da linguagem Java, com framework Spring e com banco de dados PostgreSQL com essas ferramentas foram feitas todas as etapas de criação de gerenciamento voltada para o condomínio, sendo um software amigável para o morador, porteiro e administrador. Em etapas de criar ou modificar, usuários, condomínio, espaço, convites, entre outros métodos padrões de um condomínio.

Nesse artigo será explicitados os métodos utilizados no desenvolvimento da aplicação, conceitos de cada parte do projeto, Serão abordados tópicos teóricos que foram essenciais no estudo e pesquisa como o Spring, API REST, UML e entre outros. Serão implementados todos os tópicos teóricos na prática do desenvolvimento. Mostrara os resultados obtidos depois do desenvolvimento da aplicação. como codificações para solucionar os problemas citados aqui na introdução.

# 2. FUNDAMENTAÇÃO TEÓRICA

Neste tópico serão abordados alguns conceitos que serão utilizados na implementação do sistema. Foram abordados assuntos com base no conhecimento adquirido ao longo do curso, e na pesquisa feita para elaboração do projeto. O objetivo é mostrar as teorias de cada tópico e aprofundar o conhecimento nas soluções que são utilizados no mercado de aplicações e sendo implementados no projeto.

## 2.1 Orientação ao objeto

A Programação Orientada a Objetos (OOP) é um modelo de programação de computador que organiza o design de *software* em torno de dados ou objetos, em vez de funcionalidade e lógica (GILLIS, 2021). Os objetos podem ser definidos como campos de dados com propriedades e comportamentos exclusivos.

> A POO se concentra nos objetos que o desenvolvedor deseja manipular, não na lógica necessária para manipulá-los. Este método de programação é adequado para programas grandes e complexos que são mantidos ou atualizados ativamente. Isso inclui programas de fabricação e *design*, bem como aplicativos móveis; por exemplo, OOP pode ser usado em software de simulação de sistema de fabricação (Alexander S. Gillis, 2021).

A organização de programas orientados a objetos também torna a abordagem pro-

pícia ao desenvolvimento colaborativo, onde os projetos são divididos em grupos. Outros benefícios da OOP incluem reutilização de código, escalabilidade e eficiência.

O primeiro passo na OOP é coletar todos os objetos que o programador deseja manipular e determinar os relacionamentos entre eles - um exercício chamado modelagem de dados. Exemplos de objetos podem variar de entidades físicas (como pessoas descritas por atributos como nome e endereço) a pequenos programas de computador (como *widgets*).

Uma vez que um objeto é conhecido, ele é marcado com uma classe de objeto que define o tipo de dados que ele contém e qualquer sequência lógica que possa operar nele. Cada sequência lógica distinta é chamada de método. Os objetos podem se comunicar com interfaces bem definidas chamadas mensagens.

Segundo Sergio Toledo (2019), a OOP é construída em volta de 4 pilares: Herança, Encapsulamento, Abstração e polimorfismo. Conforme são apresentados a seguir:

- **Abstração:** Abstração significa "esconder" parte da implementação de um objeto, expondo apenas uma interface simples para uso. No caso de um forno de micro-ondas, você não precisa entender todos os meandros de como os componentes internos geram ondas e calor, basta apertar um botão ou dois para desfrutar de uma refeição quente. Todas essas complexidades são detalhes de implementação, você não precisa conhecê-las (SERGIO TOLEDO, 2019).
- **Encapsulamento:** Encapsulamento refere-se à construção de um objeto de forma a proteger o acesso direto aos seus dados internos. Quando encapsulamos um objeto, agrupamos propriedades e métodos diretamente relacionados dentro do mesmo objeto, permitindo que essas propriedades sejam acessíveis apenas por meio de métodos públicos. Desta forma, lidamos com questões importantes como a segurança e confiabilidade do estado do objeto (SERGIO TOLEDO, 2019).
- **Herança**: A herança é um método para eliminar a duplicação de código, como o nome sugere, um objeto pode herdar características (ou seja, propriedades e métodos) de outra classe sem substituir essas mesmas características (SERGIO TOLEDO, 2019).
- **Polimorfismo:** Poli significa muitos, *Morphos* significa forma, então polimorfismo significa muitas formas. Na POO, o polimorfismo é caracterizado por duas ou mais classes com métodos de mesmo nome, mas possivelmente implementações diferentes. Portanto, qualquer objeto que implemente o mesmo método pode ser usado sem se preocupar com o tipo do objeto. Na prática, isso nos permite remover várias instruções if ou switch cases do nosso código (SERGIO TOLEDO, 2019).

## 2.2 Spring

Segundo TechTargetContributor (2021), *Spring Framework* (Spring) é uma estrutura de aplicativo de código aberto que fornece suporte de infraestrutura para o desenvolvimento de aplicativos Java. Como uma das estruturas *Java Enterprise Edition* (Java EE) mais populares, o Spring ajuda os desenvolvedores a construir aplicativos de alto desempenho usando POJOs (Plain Old Java Objects).

Uma estrutura é uma coleção de código predefinido ao qual os desenvolvedores podem adicionar código para resolver um problema em um domínio específico. Existem muitos frameworks Java populares, incluindo *Java Server Faces* (JSF), *Maven, Hibernate, Struts e Spring* (TechTargetContributor, 2021)

Lançado por Rod Johnson em junho de 2003 sob a licença Apache 2.0, o *Spring Framework* é hospedado pelo *SourceForge* (TechTargetContributor, 2021)

## 2.3 UML

Segundo Leandro (2012), "UML é um acrônimo para a expressão *Unified Modeling Language.* Pela definição de seu nome, vemos que a UML é uma linguagem que define uma série de artefatos que nos ajuda na tarefa de modelar e documentar os sistemas orientados a objetos que desenvolvemos".

Este possui nove tipos de diagramas para documentar e modelar vários aspectos do sistema. A maioria dos problemas encontrados em sistemas orientados a objetos derivam da construção do modelo, do projeto do sistema conforme o trabalho de Segundo Leandro (2012). Muitas vezes empresas e profissionais não dão muita atenção a essa fase do projeto e acabam cometendo vários erros na análise e modelagem. Isso acontece na modelagem, pois nós profissionais da área sabemos que os projetos geralmente começam na fase de codificação.

### 2.3.1 Diagrama de classe

Na UML, um diagrama de classes é um dos seis tipos de diagramas de estrutura. Os diagramas de classes são a base do processo de modelagem de objetos, modelando a estrutura estática do sistema. Dependendo da complexidade do sistema, você pode usar um único diagrama de classes para modelar todo o sistema ou usar vários diagramas de classes para modelar os componentes do sistema (IBM,2021).

> Um diagrama de classes é uma réplica de um sistema ou subsistema. Você pode usar diagramas de classes para modelar os objetos que compõem seu sistema, mostrando os relacionamentos entre os objetos e descrevendo o que esses objetos fazem e os serviços que eles fornecem (IBM, 2021).

Os diagramas de classes são úteis em muitos estágios do projeto do sistema. Durante a fase de análise, os diagramas de classes podem ajudá-lo a entender as necessidades do domínio do problema e identificar seus componentes. Conforme trabalho de IBM (2021), em projetos de software orientados a objetos, os diagramas de classes criados no início do projeto contêm classes que geralmente são traduzidas em objetos e classes de software reais à medida que você escreve o código. Mais tarde, você pode refinar os modelos analíticos e conceituais anteriores em diagramas de classe mostrando partes específicas do sistema, interface do usuário, implementação lógica etc. O diagrama de classes então se torna um instantâneo que descreve exatamente como o sistema funciona, os relacionamentos entre os componentes do sistema em diferentes níveis e como você planeja implementar esses componentes.

### 2.3.2 Diagrama de caso de uso

De acordo com IBM (2021), na UML, os diagramas de caso de uso modelam o comportamento de um sistema e ajudam a capturar os requisitos do sistema. Os diagramas de caso de uso descrevem a funcionalidade de alto nível e escopo do sistema. Esses diagramas também identificam as interações entre o sistema e seus agentes. Os casos de

uso e atores em um diagrama de casos de uso descrevem o que o sistema faz e como os atores o utilizam, não como o sistema funciona internamente.

> Os diagramas de caso de uso ilustram e definem o contexto e os requisitos de todo o sistema ou de suas partes significativas. Você pode modelar sistemas complexos com um único diagrama de caso de uso ou pode criar muitos diagramas de caso de uso para modelar componentes do sistema. Normalmente, os diagramas de caso de uso são desenvolvidos nos estágios iniciais de um projeto e referenciados durante todo o processo de desenvolvimento (IBM, 2021).

No Diagrama de caso de uso tem elementos que fazem parte como: (LEANDRO,2012):

- **Caso de uso :**Descrevem o papel que o sistema desempenha para atingir os objetivos do usuário. Os casos de uso devem produzir resultados observáveis que sejam valiosos para os usuários do sistema,
- **Agente:** representam funções de usuário que interagem com o sistema que está sendo modelado. Os usuários podem ser pessoas, organizações, máquinas ou outros sistemas externos,
- **Subsistemas:** No modelo UML, um subsistema é um componente prototípico que representa uma unidade independente de comportamento em um sistema. Os subsistemas são usados em diagramas de classe, componente e caso de uso para representar grandes componentes no sistema que está sendo modelado.
- **Relacionamentos em diagramas de casos de uso:** Na UML, os relacionamentos são conexões entre os elementos do modelo. Um relacionamento UML é um elemento de modelo que adiciona semântica a um modelo, definindo a estrutura e o comportamento entre os elementos do modelo.

## 2.4 Swagger

É um aplicativo de código aberto que ajuda os desenvolvedores a definir, criar, documentar e consumir APIs REST. Em poucas palavras, o *Swagger* visa padronizar esse tipo de integração, descrevendo as características que uma API deve ter, como *endpoints*, dados recebidos, dados retornados, códigos HTTP e métodos de autenticação. Ele simplifica o processo de escrever APIs especificando padrões e fornecendo as ferramentas necessárias para escrever APIs seguras, de alto desempenho e escaláveis (ADMINGR1D, 2022).

No mundo do software de hoje, nenhum sistema é executado online sem expor uma API. Passamos de sistemas monolíticos para micros serviços. Todo o design dos micros serviço é baseado em uma API REST (ADMINGR1D, 2022).

APIs REST são frequentemente usadas para a integração de aplicações, seja para consumir serviços de terceiros, seja para prover novos. Para estas APIs, o Swagger facilita a modelagem, a documentação e a geração de código (ADMINGR1D, 2022).

## 2.5 Json Web Token (JWT)

Conforme afirma Autho (2021), *JSON Web Token* (JWT) é um padrão aberto (RFC 7519) que define uma maneira compacta e independente de transmitir informações com segurança entre as partes como um objeto JSON. Essas informações podem ser verifica-

das e confiáveis porque são assinadas digitalmente. Os JWTs podem ser assinados usando um segredo (com o algoritmo HMAC) ou um par de chaves pública/privada usando RSA ou ECDSA.

Um JWT consiste em três partes separadas por pontos (.): *Header*, *Payload* e *Signature*. Conforme o trabalho de Autho (2021). O cabeçalho geralmente consiste em duas partes: o tipo de token, que é o JWT, e o algoritmo de *hash* usado, como HMAC SHA256 ou RSA. Segundo Autho (2021) A segunda parte do token é a carga útil, que contém as declarações. Uma declaração é uma declaração sobre uma entidade (geralmente um usuário) e metadados adicionais. Este é um conjunto predefinido de declarações, que não são obrigatórios, mas são recomendados para fornecer um conjunto útil e interoperável de declarações. Alguns deles são*: iss* (remetente), *exp* (tempo de expiração), *sub* (assunto), a*ud* (público), etc. A assinatura é usada para verificar se o remetente do JWT é quem afirma ser e para garantir que a mensagem não tenha sido alterada no processo. Para criar a parte assinada, você precisa obter o cabeçalho codificado, a carga útil codificada, o segredo, o algoritmo especificado no cabeçalho e assiná-lo (AUTHO, 2021).

## 2.6 API

Uma API é um conjunto de definições e protocolos usados no desenvolvimento e integração de aplicativos segundo RedHat (2020). Uma API às vezes é descrita como um contrato entre um provedor e um consumidor de informações, estabelecendo o que o consumidor precisa (chamadas) e o que o produtor precisa (respostas). Por exemplo, um design de API para um serviço meteorológico pode especificar que o usuário forneça um CEP e o produtor responda em duas partes, com a primeira parte contendo a temperatura mais alta e a segunda parte contendo a temperatura mais baixa.

Em outras palavras, ao interagir com um computador ou sistema para recuperar informações ou executar uma função, a API ajudará a comunicar o que você deseja ao sistema para que o sistema entenda e execute sua solicitação.

> Pense nas APIs como um mediador entre os usuários ou clientes e os recursos ou serviços web que eles querem obter. As APIs também servem para que organizações compartilhem recursos e informações e, ao mesmo tempo, mantenham a segurança, o controle e a obrigatoriedade de autenticação, pois permitem determinar quem tem acesso e o que pode ser acessado (REDHAT, 2020).

Outra vantagem de usar a API é que você não precisa saber todos os detalhes sobre armazenamento em cache, como os recursos são recuperados ou de onde eles vêm.

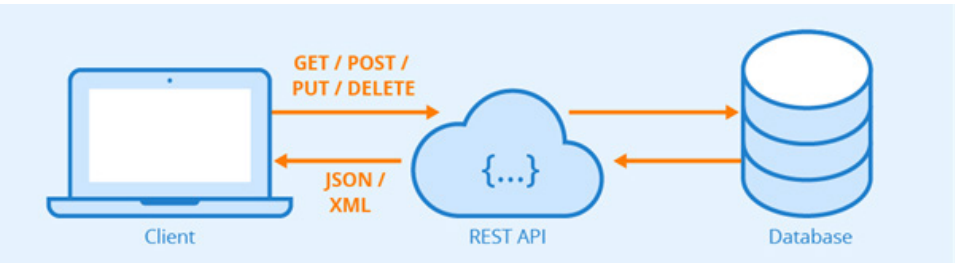



**Figura 1.** imagem representativa das conexões do api com banco de dados e cliente.

**Fonte:** SOMDAY, 2021

## 2.7 Interoperabilidade

A interoperabilidade é quando dois sistemas conseguem operar entre si ou com outros, através de padrões abertos, independente de qual tecnologia (linguagem) utilizam e de onde estão instalados. Segundo Sisqualis (2020), para que seja possível trocar informações entre sistemas de qualquer área, ou da área da saúde, é necessário que haja uma integração de dados- ponto (através da interoperabilidade, integrar sistemas diferentes ou de diferentes tecnologias utilizando protocolos abertos).

Conforme Sisqualis (2020), existem três tipos de interoperabilidade que são: INTEROPERABILIDADE TÉCNICA que é um padrão para comunicação, envio e armazenamento de informações, de uma ou mais entidades. INTEROPERABILIDADE SEMANTICA que As informações recebidas de diferentes fontes possuem significado (semântica) e, para serem lidas e registradas corretamente, requerem o uso de ferramentas que possibilitem a classificação de ontologias de informação. E a INTEROPERABILIDADE POLITICA/HUMANA que A decisão de fornecer informações aos usuários preocupados é muito importante porque um dos principais problemas é considerado a falta de políticas públicas que incentivem o fechamento da exclusão digital. Um bom exemplo da falta de interoperabilidade da informação pode ser visto no número de estudos ou trabalhos que tratam de um mesmo assunto ou ponto setorial.

## 2.8 Arquitetura *Rest*

Segundo Lima (2020), REST (Representational State Transfer) é um modelo de arquitetura, não uma linguagem de programação ou tecnologia, que fornece orientação para sistemas distribuídos se comunicarem diretamente usando princípios e protocolos da *Web* existentes sem a necessidade de SOAP ou outro protocolo complexo

REST requer que o cliente faça uma solicitação ao servidor para enviar ou modificar dados. A solicitação inclui: um verbo ou método HTTP que define o que o servidor fará, um *header* (cabeçalho) com cabeçalhos de solicitação que transmitem informações sobre a solicitação, Informações no corpo da solicitação, essas informações são opcionais (LIMA, 2020).

Na arquitetura REST tem os métodos que são cincos: O método GET é o método mais comumente usados e geralmente é usado para solicitar ao servidor o envio de recursos, o método POST foi projetado para enviar dados de entrada para o servidor. Na prática, é frequentemente usado para dar suporte a formulários HTML, o método PUT edita e atualiza o documento no servidor e método *DELETE*, como o nome sugere, exclui um dado ou uma coleção do servidor (LIMA, 2020).

Cada resposta retornada por um aplicativo REST envia um código que define o status da solicitação. Por exemplo:

> **200 (OK**), a solicitação foi concluída com sucesso, **201 (CREATED**), o objeto ou recurso foi criado com sucesso, **204 (SEM CONTEÚDO)**, o objeto ou recurso foi excluído com sucesso, **400 (BAD REQUEST)**, ocorreu um erro na solicitação (podem ser vários os motivos), **404 (Não Encontrado)**, rota ou coleção não encontrada, **500 (Erro de servidor interno)**, ocorreu algum erro de servidor (LIMA, 2020).

# 3. IMPLEMENTAÇÃO

Aqui será demostrado a parte prática do aplicativo, cada tópicos que foram abordado anteriormente da fundamentação teórica, será aplicada na pratica por etapas na produção da API, como: a descrição do sistema, *framework* utilizado para desenvolvimento do sistema, Modelagens do API no sistema, Implementação do JWT **(**JSON Web token), *Repository* (Camada de repositórios), *Controller* (camada de controle), método para criar dados (POST), método para ler dados (GET), método para atualizar dados (PUT), método para deletar dados (DELETE), *Service* (Regra de negócio dos endpoints) e *ExceptionHandler* (Manipulador de Exceções**).**

## 3.1 Descrição do sistema

O sistema é um Aplicativo que gerencia e monitora condomínio residencial, é responsável por agendamento de espaços (Quadra, piscina, área de evento etc.), controle e solicitação de convites de (visitantes e de serviço), entrada e saída de veículos prezando ajudar o morador a ter uma segurança e facilidade de ter os serviços disponibilizados pelo condomínio. E nisso foi criado esse API que foi feito pelo *framework* Java (SPRING).

## 3.2 Framework utilizado para desenvolvimento do sistema

O Spring foi o *framework* utilizado nesse projeto por ser um ecossistema que facilita na complexidade de desenvolver o aplicativo Java, com isso a criação das classes com os atributos, construtores e métodos que são trabalhosos foram facilitados com dependências como *Spring data JPA* que funciona de suporte para os repositórios de cada classe do projeto, *Validation* (Validação) que ajuda nos *Controllers* (Controles) em caso o usuário passa uma requisição que infrinja as regras colocada no *model*. E entre outros.

Outra parte do ecossistema do *framework Spring* que foi utilizado no projeto foi o *Spring Security*. Que ficou responsável na parte da proteção dos *endpoints* de cada classe do projeto. Com ele, faz com que precise de uma autenticação para poder acessar o CRUD (acrónimo do inglês Create, Read, Update and delete) de uma classe.

Com isso, determinado usuário pode criar, alterar ou excluir dados na API. Sendo assim um auxiliar na ponte entre o API e o banco de dados.

## 3.3 Análise do aplicativo

O processo de planejamento do aplicativo começou através de analisar como seria as classes, atributos e tipos de usuários que o sistema ia se interagir, com isso foram criados a modelagem da UML com dois diagramas, um seria o diagrama de caso de uso, para mostrar detalhadamente as ações de cada usuário teria através das classes. E a outra seria o diagrama de classe, para mostrar as relações entre classes e atributos.

### 3.3.1 Diagrama de caso de uso

O diagrama de caso de uso do projeto consiste em três atores, que são usuários, mas o usuário comum tem restrições, o usuário funcionário tem restrições, mas tem preferencias especiais que o usuário comum, não tem. E o usuário administrador pode controlar tudo.

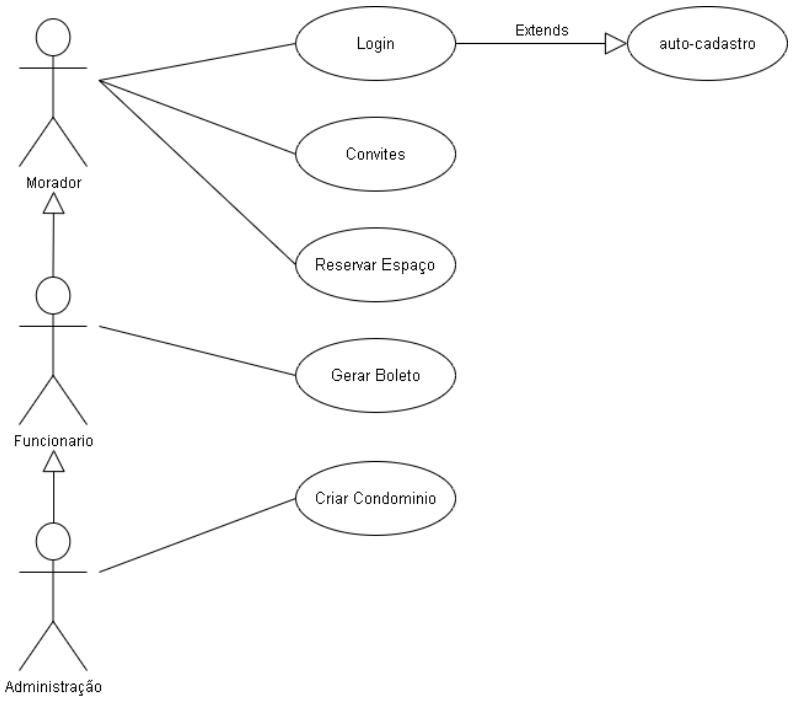



**Figura 2.** Diagrama de caso de uso do aplicativo

**Fonte:** Autoral, 2022.

### 3.3.2 diagramas de classe

Nesse diagrama mostra as classes do projeto, e suas relações, como de *enumeration* (enumerações), quanto de relações entre classes distintas com cardinalidade e tipos de relações entre elas.

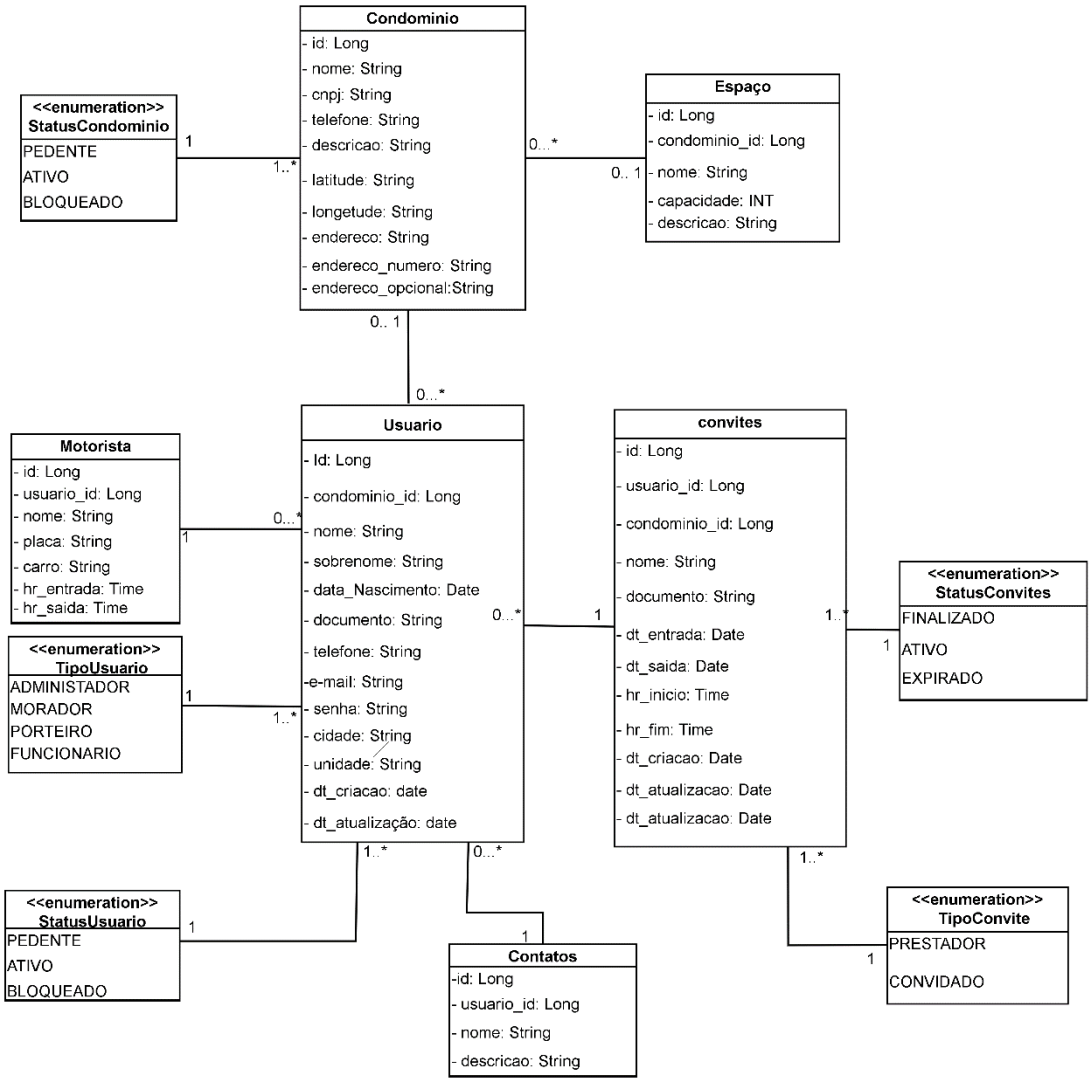



**Figura 3.** Diagrama de classe do aplicativo

**Fonte:** Autoral, 2022.

## 3.4 modelagens do API para o Sistema

A construção da API foi feita através dos controladores na IDE que por ela fazia a criação da CRUD () e nos Controladores utilizamos a injeção de dependência *@autowired* que puxava tanto os *repository* (repositório) de determinada classe para pode fazer as regras através do CRUD. Também utilizava a injeção de dependência dos *services* que será explicado detalhadamente a seguir.

Para facilidade do consumo da API para que outros programadores utilizem essa Aplicação foi utilizada uma documentação chamada *SWAGGER* que ajuda já implementando atributos de cada *endpoint* automaticamente no ponto de fazer os testes de cada um. Foi adicionado a dependência no IDE e feito uma classe para adicionar as suas regras e implementações do sistema ele vai poder utilizar como exemplo abaixo que foram os linhas de código utilizados no projeto, figura 4.

```
@Configuration
public class swaggerConfiguration {

    @Bean
    public Docket trancaApi() {
         return new Docket(DocumentationType.SWAGGER_2)
                    .select()
                    .apis(RequestHandlerSelectors.basePackage("com.project.trancaApp"))
                    .paths(PathSelectors.ant("/**"))
                    .build()
                   // .ignoredParameterTypes(Usuario.class)
                    .globalOperationParameters(Arrays.asList(
                            new ParameterBuilder()
                                    .name("Authorization")
                                    .description("Header para token JWT")
                                    .modelRef(new ModelRef("string"))
                                    .parameterType("header")
                                    .required(false)
                                    .build()));
    }
}
```

**Figura 4.** Classe responsável do Swagger

**Fonte:** Autoral, 2022.

## 3.5 Implementação do JWT

A autenticação utilizada no projeto foi em base do JWT (JSON WEB TOKEN) com finalidade de ter segurança nos endpoints da API. Para isso foi feito um método de login que o usuário precisaria fazer que foi e-mail, e senha. Que foi adicionada no momento que o usuário faz o cadastro na API.com isso gerasse um token com um limite a ser expirado foi colocado um tempo em nanosegundo que a biblioteca só suporta nesse formato de tempo. Fazendo se cálculos para converter em minutos que ficou definido em 30 minutos.

O *header* (Cabeçalho) foi divido em dois, primeiro o tipo de token que foi utilizado (Bearer) e o hash para o JWT o algoritmo HMAC. No sistema foi criado uma classe chamada "*AutenticaçãoViaTokenFilter*" que nela foi modelada em volta do JWT e a segurança e através dessa classe faz todo o sistema por trás quando o usuário for efetuar o *login*. Como validação do token, o ID do usuário que efetuou esse login puxando pelo *repository* da classe usuário, para que tenha uma exceção de que o usuário tem que existir no banco de dados para poder fazer o processo de login.

A classe da filtragem do token vem de uma injeção de dependência do servisse do token chamado "TokenService" e nele tem todas as regras do JWT como: tipo de *hash* (HS256) o que foi abordado anteriormente, tempo de expiração, data do dia do login, status logado do usuário e a data de expiração.

## 3.6 *Repository*

Para que as classes possam ser persistidas, precisamos usar Spring data JPA que é uma dependência do ecossistema *Spring*. Na prática cada classe precisa de uma interface de acesso ao DB (database), a classe dever herdar a classe *JpaRepository* que por sua vez precisa de dois parâmetros. Primeiro a classe que vai ser persistida e a segunda é a chave primaria dessa classe.

Através dessa classe pode-se fazer as operações dos *endpoints* como criar, apagar, editar e buscar. Antes do método colocasse a anotação @repository para as interfaces das

classes funcionem do jeito correto.

## 3.7 *Controller*

Nessa camada fica responsável pelos endpoints da API que são as ações que vão adicionar, deletar ou alterar os dados no Banco de dados, cada classe tem sua respectiva classe de *controller* e com elas pode utilizar os objetos delas.

### 3.7.1 método para criar dados (POST)

O método utiliza o model da classe pegando os objetos que estão nela, e através do *repository*, e do *service* que rodam pelo *back* desse método, e com uma anotação @valid do *bean validation* , e outra anotação *@RequestBody* que nessa anotação é para que na hora de utilizar a APi informa que esse método tem que passar informações no corpo da requisição, após isso retorna para um método da classe *service* para pode salvar esses dados que foram colocados pelo usuário, vão para o *repository* e por sua vez passa para a DB guardar as informações . Em cima do método tem uma anotação POST que é responsável por criação e uma anotação http do *ResponseStatus* que retorna o código (200- ok). Como é mostrado na figura 5.

```
//criar usuario no DB
@PostMapping
@ResponseStatus(HttpStatus.CREATED)
public Usuario adicionar (@Valid @RequestBody Usuario usuario) {
    return catalogoUsuarioService.salvar(usuario);


}
```

**Figura 5.** print do código para criar o método post

**Fonte:** Autoral, 2022.

### 3.7.2 Método para ler dados (GET)

Esse método consiste em ler informações no banco, utilizando o *ResponseEntity* que por sua vez utiliza as informações do Dto (data transfect object) mas também pode usar o *model* genérico da classe, mas no projeto foi utilizado o Dto por filtrar determinados atributos que vão passar no corpo da requisição.

Cada classe tem pelo menos duas ou três métodos de GET, uma é o *ListAll*(Listar todos). Nesse método pega todos os dados de um determinado *endpoint*. Nesse método utiliza a função *list* (Listar) pega essas informações da classe e para isso tem o *repository* da classe, e com o método *findAll* pega todas as informações que o usuário deseja consultar.

Outra aplicação do método de GET é para pesquisar por uma determinada informação e foi usada para poder funcionar esse método, o *responseEntity* e Dto(data transfect object) pelo fato de que pode-se pegar atributos específicos em cada GET. No método tem uma função chamada *opcional* junto com o método. *ispresent* que tem regra de negócio, que é implementado na camada *service*, quando o usuário digitar o tipo de objeto que foi determinado no parâmetro do método, o *Spring* vai passar para o repositório ver se existe essa entidade no banco de dados, e depois que for confirmado, retorna para a API os da-

dos dessa entidade. Caso contrário retorna um *notfound* para o Usuário. Alguns *endpoints* GET utiliza a chave primaria ID ou outros atributos importantes como documentos, para ter um fácil uso do usuário que consumir a API. Como é ilustrado na figura 6.

```
//lista todos os usuarios do DB
@GetMapping
public List<ListUsuario> listar() {

    List<Usuario> usuarios = usuarioRepository.findAll();
    return ListUsuario.converter(usuarios);
}
```

**Figura 6.** Criar o método GETS

**Fonte:** Autoral, 2022.

### 3.7.3 Método para atualizar dados (PUT)

A atualização de dados de uma entidade no DB é utilizando o método PUT que pega o ID e a classe que vai ser persistida no parâmetro do *responseEntity*, e na mesma etapa do GET é utilizada aqui para verificar se o usuário existe através de um If, após isso o id é setado e o usuário faz as alterações passando no corpo todos os atributos da classe, e alterando a que deseja, e seguinte é salvo no *service.* Como é mostrado na figura 7, abaixo.

```
//atualizar usuario no DB
@PutMapping("/{id}")
public ResponseEntity<Usuario> atualizar(@Valid @PathVariable  Long id,
        @Valid @RequestBody Usuario usuario) {
    if(!usuarioRepository.existsById(id)) {
        return ResponseEntity.notFound().build();
    }
    usuario.setId(id);
    usuario  = catalogoUsuarioService.salvar(usuario);
    return ResponseEntity.ok(usuario);
}
```

**Figura 7.** Criar o método PUT

**Fonte:** Autoral, 2022.

### 3.7.4 Método para deletar dados (DELETE)

O método de deletar dados vem através do método *DELETE* que o usuário passa o ID na URL sem passar nada no corpo o método vai verificar se esse ID existe através da camada *service* com um IF, depois vai no catálogo usuário *service*, e após isso excluir do banco a entidade. Como é mostrado na figura 8, abaixo

```
//excluir usuario no DB
@DeleteMapping("/{id}")
public ResponseEntity<Void> remove (@PathVariable Long id) {

    if (!usuarioRepository.existsById(id)) {
        return ResponseEntity.notFound().build();
    }
    catalogoUsuarioService.excluir(id);
    return ResponseEntity.noContent().build();
}
```

**Figura 8.** Criar o método DELETE

**Fonte:** Autoral, 2022.

## 3.8 Service (Regra de negócio dos endpoints)

No projeto em todo foi adicionado uma camada especializada para gerir a regra de negócio do sistema. Essas classes cada um especificamente tem suas regras de como vai funcionar para a hora do consumo da API, A criação dela foi para evitar com que o *RESTController* (camada de controle) ficasse sobrecarregada em organizar os CRUD, e por cima as regras de negócio.

Cada *Service* tinha a injeção de dependência dos *repository* (JPA Repository) para pegar as informações da classe *model* da respectiva entidade. No sistema por exemplo na classe tanto Usuário, quanto Condomínio. Foram colocadas regras para evitar criação de ambas separadamente com Documento (CPF/CNH), e-mail para Usuário. Ou Documento (CNPJ) Condomínio. Isso ajuda na parte do POST la na classe *controller* já barrar o avanço de criação dos dados chegarem ao DB (database). Outra parte seria na busca de dados de uma respectiva entidade, tendo um método para buscar através do *repository,* caso não for encontrado esse dado, entra outra camada responsável por retornar uma mensagem para o consumidor da API, lá no *POSTMAN* que se chama *ExceptionHandler* (Manipulador de Exceções) que vai ser explicado no tópico seguinte.

### 3.8.1 *ExceptionHandler*

Para ajudar o consumidor a identificar os erros na hora de passar o *body* (corpo) da requisição foi criado uma classe responsável em mostrar o erro e não entregar uma poluição de código na tela do usuário, assim fazendo uma filtragem de todo esse código para retornar uma mensagem coloca na produção e o http gerado após enviar a requisição. Essa classe é herdada de uma biblioteca do *Spring* chamado *ResponseEntityExceptionHandler.* Nela tem várias linhas de comando que fica responsável pelas requisições HTTP que é mostrado no response na ferramenta que for utilizada para testar API.

Continuando na classe foi injetado uma dependência chamada "*messagesource* "que fica responsável por tratar os problemas que a API retornar para o Sistema

Na camada *ExceptionHandler* foi criada outra classe para fazer complemento a que foi citada anteriormente nesse tópico, que se chama "Problema" nessa classe fica os atributos que vão auxiliar o consumidor a onde está o problema quando for testar a API. Passando no response o: Status, data e hora dessa requisição, o título, e o campo que é utilizada no *ApiExceptionHandler* com a ajuda do Jackson (JsonInclude) faz uma injeção de dependência de um arquivo chamado "*messages.properties*" e nela tem o que passa

quando a API retornar por exemplo se ela retornar um erro *blank* dentro desse arquivo tem "*blank* = é obrigatório" fazendo com que o usuário perceba que um objeto não pode ser nulo e por isso retornou um erro no response..

## 3.9 Interoperabilidade

No projeto como todo a construção se passou pela técnica de interoperabilidade por vários fatores para que seja organizado e atenta corretamente todos os quesitos das comunicações das entidades e também dos envios de informações necessárias para o usuário

A *web service* foi a essência da interoperabilidade, a criação de todo o projeto pelo Spring utilizando todas as ferramentas para conectar aplicações de diversas linguagens, a API pode ser consumida por vários tipos de linguagem respondendo as requisições HTTP da API, e comunicando com o Banco de dados fazendo se assim uma interoperabilidade Técnica

A parte de que cada *endpoint* de exemplo ***GET***, tem dados distintos, como ter algumas informações especifica no GET por id, e o ***GETALL*** ter todas as informações da classe é entendida como interoperabilidade Relação Política/ humana.

4. RESULTADOS

O back-end do programa está terminado, para utilizar ele bastar ser consumido por um front-end como Javascript puro, *react, VueJs* e entre outros. Como teste foi usado uma classe para ser populado no banco de dados e assim mostrar os resultados obtidos na construção da aplicação, para isso foi usado uma ferramenta HTTP chamada *Postman*, mas também tem a documentação chamada SWAGGER que nela já tem todos os atributos de classe e só no ponto de ser testado.

Percebesse que está sendo ilustrado uma requisição do verbo POST na **URI** http://localhost:8080/usuario que é o endereço do *endpoint*. No *Body* (CORPO) é usada o formato Json que foi a escolhida, e também é a mais usada atualmente. Dentro dela através dos colchetes tem todos os objetos da classe usuário, e foram preenchidas com suas respectivas variáveis. Depois de ser preenchida se dá o *SEND* (enviar) para manda essa solicitação para a API, e seguinte ela retorna à requisição com **201 *CREATED*** dizendo que foi criado esse usuário e foi guardado no DB (database). O response mostra os objetos que foram salvos. Como mostra na figura 9

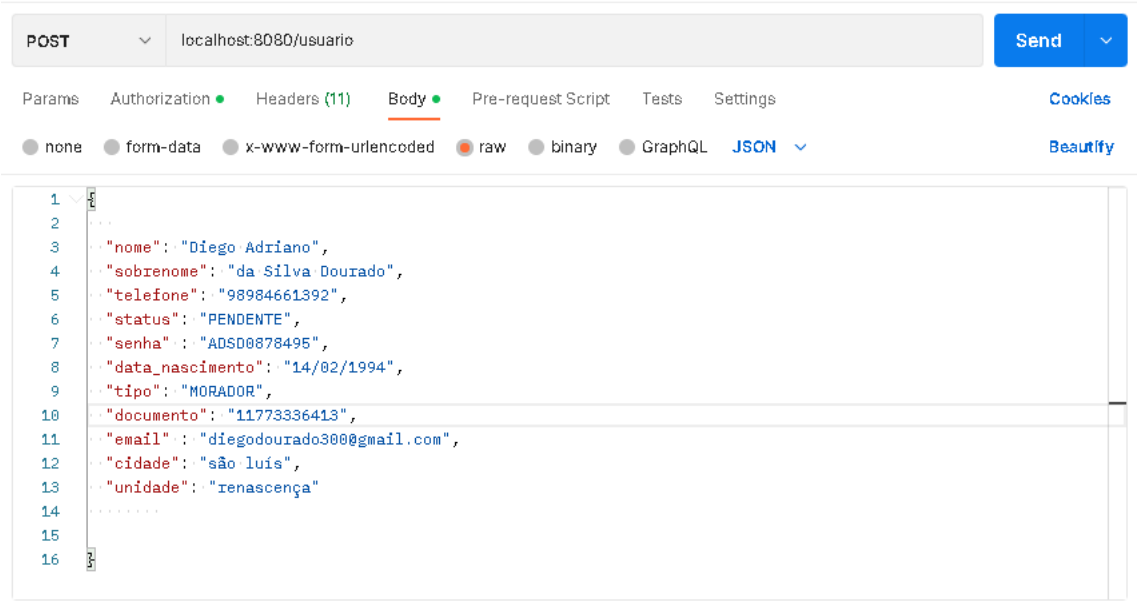



**Figura 9.** Post usuário feito no postman

**Fonte:** Autoral, 2022.

Na figura 9, é apresentado os dados que foram selecionados pelo usuário, e salvo no banco de dados, através do response a API divulga os atributos da classe e as informações que foram colocadas diante dela.

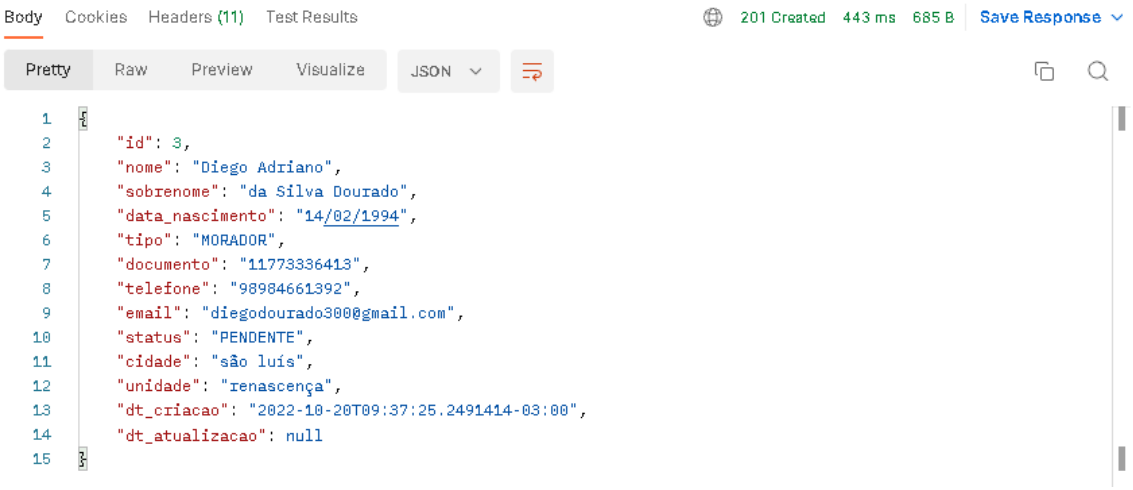



**Figura 10.** O usuário 2 response feito no postman

**Fonte:** Autoral, 2022.

Anteriormente foi demostrado o Cadastro utilizando o *Postman*. Na figura 10 é mostrada a tela de cadastro do aplicativo consumindo a API, nela mostra os atributos e o usuário irá digitar seus dados e cadastrar, e os dados salvos no banco de dados do sistema.


**Figura 11.** Tela de cadastro do usuário no Aplicativo

**Fonte:** Autoral, 2022.

Após a criação do usuário precisa fazer o login para poder gerar o token de autenticação, e assim tem um *endpoint* responsável por isso do verbo POST na URI http://localhost:8080/auth. Esse *endpoint* o usuário passa no corpo em formato JSON o e-mail, e senha que foram digitadas na hora do cadastro do usuário. Depois de preencher ele envia a requisição para API, que vai verificar se existe esse usuário, e retorna no status http **200 OK**, e no response retornar o token. Com o tipo dele "*BEARER*" e com o token criptografado.

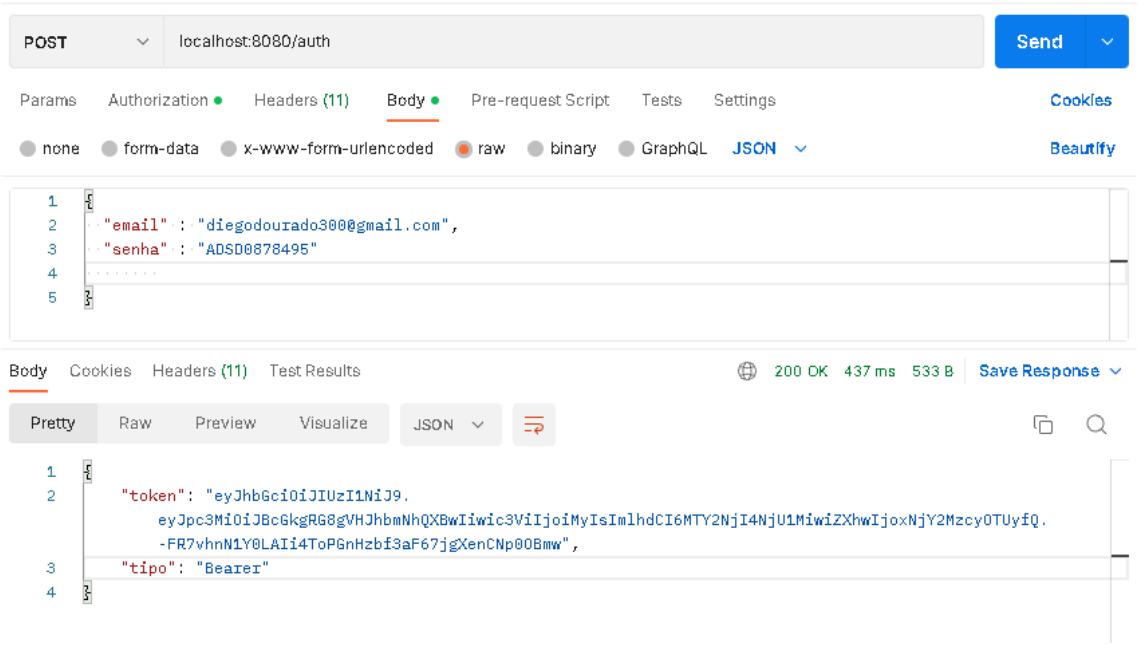



**Figura 12.** Login response feito no postman

**Fonte:** Autoral, 2022.

Na figura 12, mostra como o token pode ser usada para outros *endpoints* como um exemplo na criação de convites o usuário, vai solicitar um convite para o condomínio, com isso utilizando o verbo POST na URI http://localhost:8080/convites. Mas para poder mandar a requisição ele precisa colocar o token que foi gerado no login então tem uma opção para autenticação que seleciona o tipo do token do cabeçalho e o token, colocando

poderá fazer a solicitação que está privada

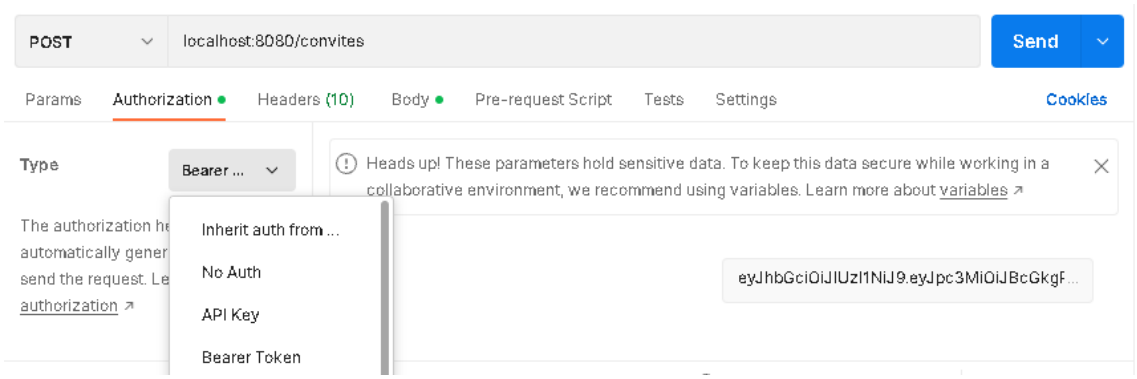



**Figura 13.** Print do post token feito no *postman.*

**Fonte:** Autoral, 2022.

E após colocar o token e preencher os atributos da classe e suas informações em cada uma, envia a solicitação para a API que retorna o status 201 ***CREATED*** e response do *body* retornando as informações que foram salvas no banco de dados, como mostra na figura 13, acima.

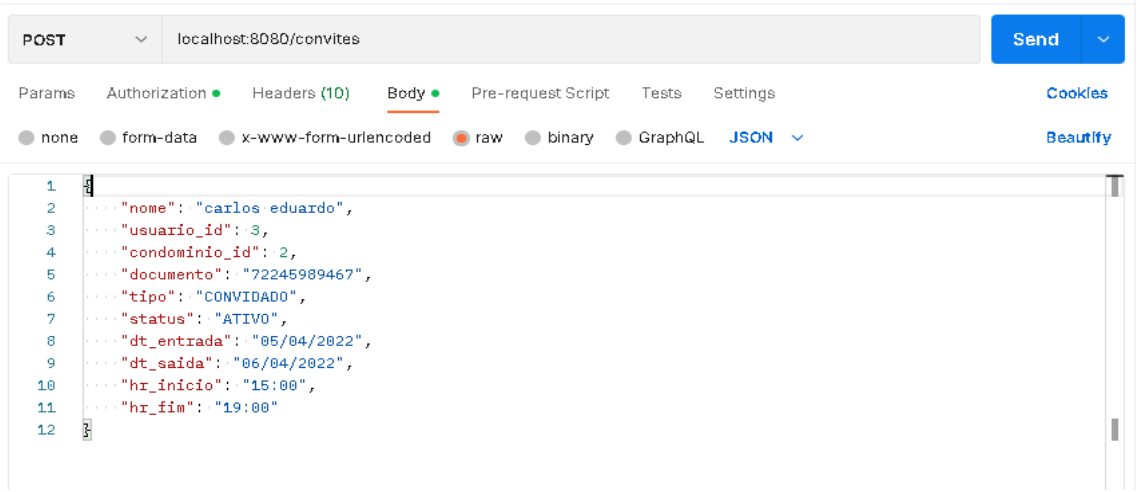



**Figura 14.** Print do POST convites feitos no postman.

**Fonte:** Autoral, 2022.

Na figura 14, foi mostrado a criação do convite através do postman. Para outro exemplo utilizei A criação do convite através do aplicativo, ela consome a API criada e mostra os atributos do endpoint convites, seleciona também um calendário e hora para poder selecionar a data e hora de entrada e saída do convidado. Ao lado mostra a lista de convites geradas que estão ativas até passar a data e hora final do convite e ser expirada. Como é mostrado na figura 15, abaixo.

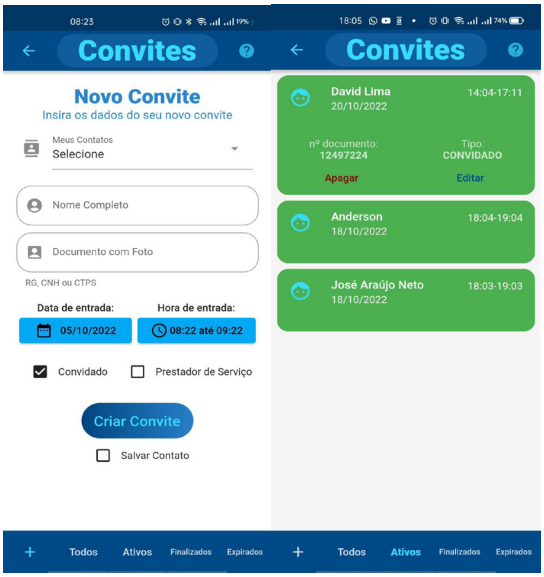



**Figura 15.** Tela do aplicativo fazendo a criação dos convites e listagem.

**Fonte**: Autoral, 2022.

Caso o usuário tente solicitar em qualquer *endpoint* que está privado com Spring Security e JWT (Json Web Token) irá retornar um *ResponseEntity* do tipo 403 *Forbidden* (não autorizado) dizendo que a ação não pode ser feita por não ser autorizado. Alguns *endpoints* tem classe de autorização na qual alguns só quem tem o perfil de administrador pode ter autorização de usar o *endpoint*, e outros os funcionários. Entretendo tem *endpoints* que não precisa do token. Como o de login, e criação do usuário. Exemplo de quando o usuário for tentar solicitar um requerimento, mas não passa ou token, ou não é autorizado. Como é mostrado na figura 16.

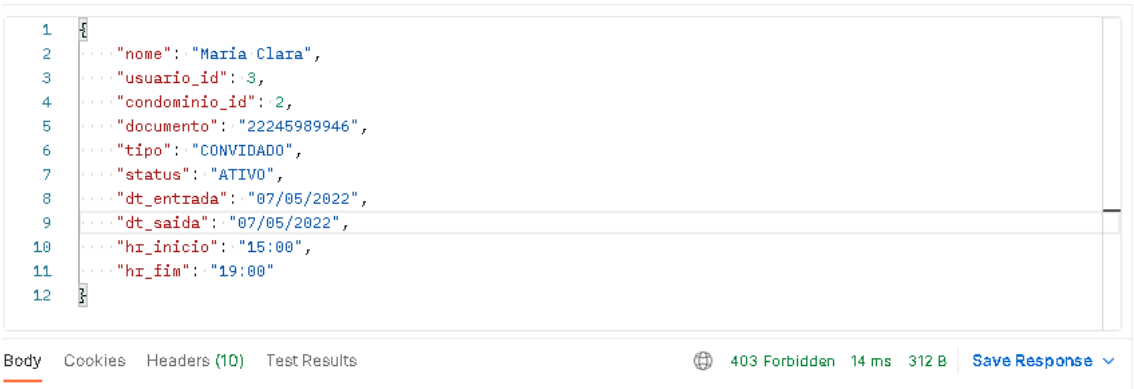



**Figura 16.** Print mostrando o 403 forbidden

**Fonte**: Autoral, 2022.

Alguns exemplos de testes que foram feitos na classe usuário para listar usando o verbo GET para listar todos os usuários, ou para listar usuário específicos com parâmetros específicos como por exemplo (CPF, ou ID).

Com o GET para listar todos, não precisar passar nenhum parâmetro na URI, apenas o endereço http://localhost:8080/usuario. E com o token, consegue fazer a solicitação da requisição para API e retorna todos os usuários que estão guardado no banco de dados como na figura 17 abaixo.

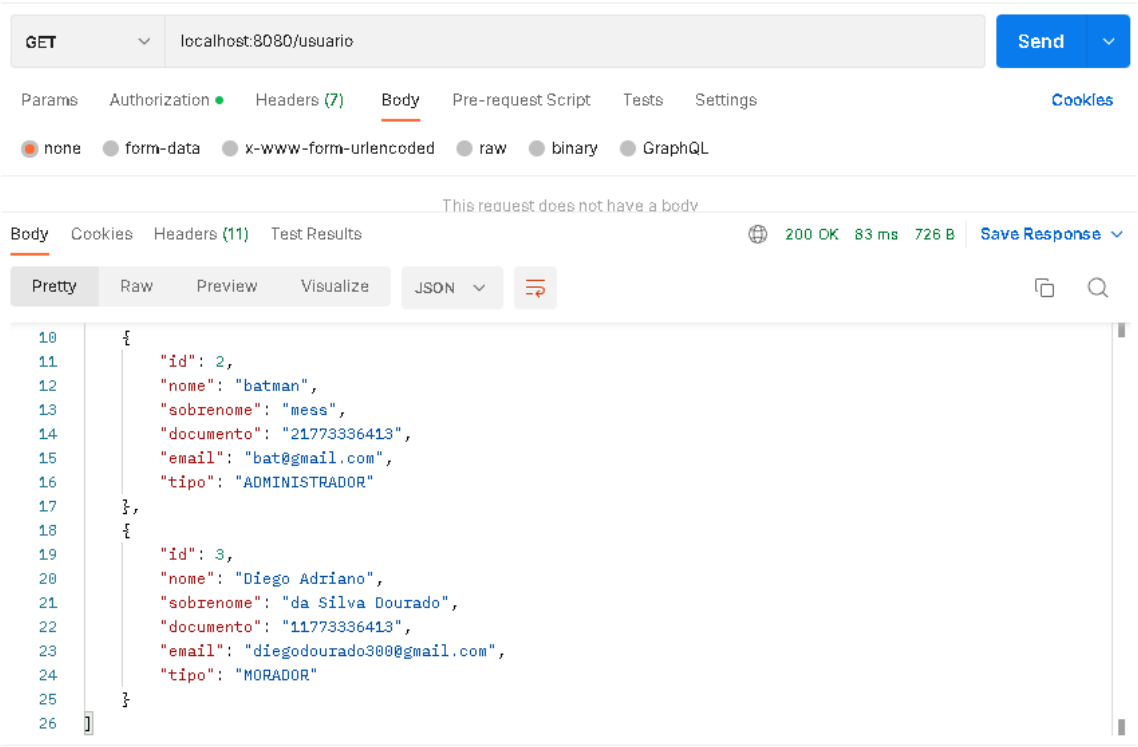



**Figura 17**. Print do GET usuário ALL

**Fonte**: Autoral, 2022.

No *endpoint* GET para procurar o usuário único utilizou-se um método global para todos as classes que é pela chave primaria o **ID,** e nela retorna informações especificas, diferente do *getAll* (Listar todos), por ser alimentadas por **Dto** diferentes. Como é ilustrado na figura 18.

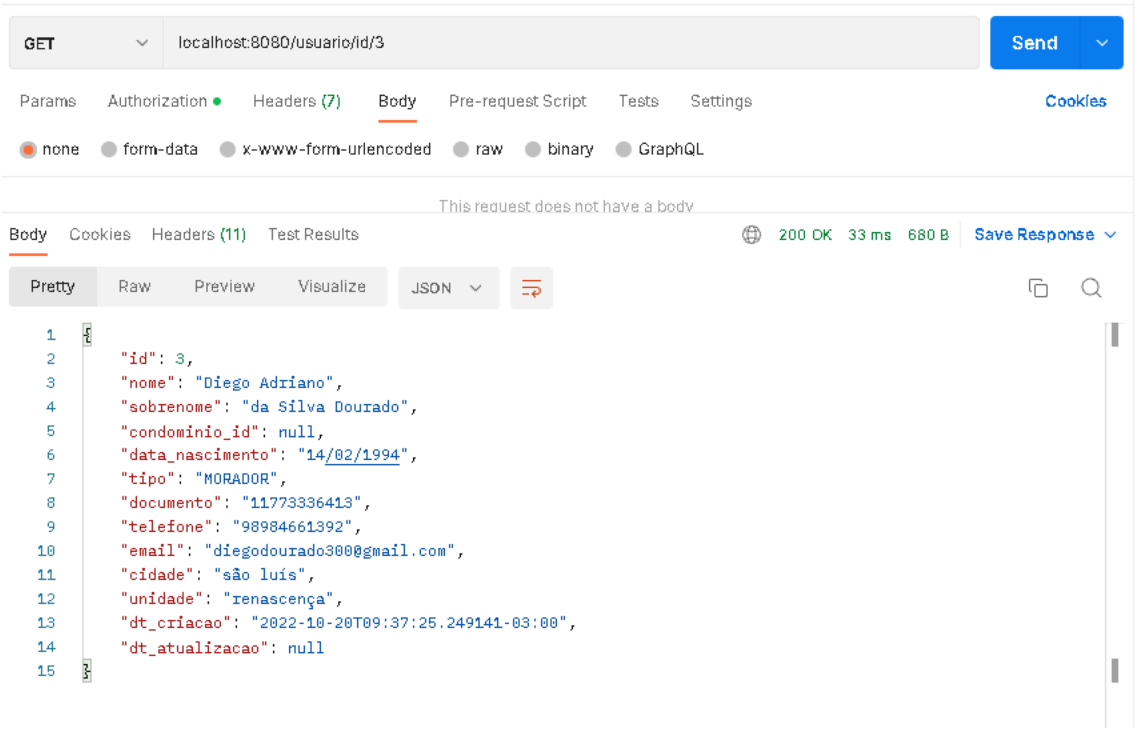



**Figura 18.** Print do GET usuário por id e CPF

**Fonte**: Autoral, 2022.

Na figura 19, é mostrado que para excluir dados, é usado no verbo DELETE, na URI

http://localhost:8080/usuario. O acesso desse endpoint, primeiro o usuário precisa ter status de administrador nisso mesmo que tente passar um token gerado no endpoint login, e não tiver sido indicado com preferencias de administrador é retornado um código **403 forbidden**, entretanto se o usuário for status administrador, é retornado em vez do código **200 ok**, foi preferido que retorne um código **204 *no content*** e com isso é informado para o usuário que o procedimento ocorreu tudo bem**.**

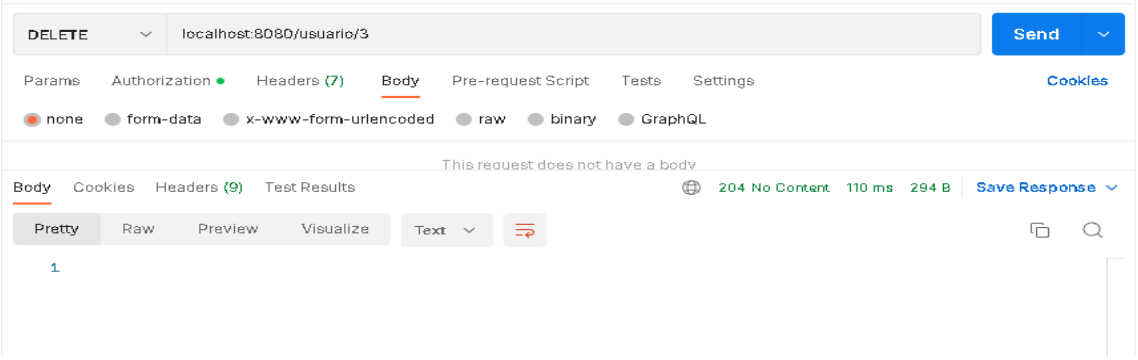



**Figura 19**. Imagem mostrando Requisição Delete

**Fonte**: Autoral, 2022.

Outros métodos também foram testados como de criação de condomínios, que funciona da mesma forma que do usuário, mas com um processo diferente para fazer a ponte entre o usuário e condomínio. Utilizando O autenticador do usuário que foi logado, tem um método POST que faz essa ligação pegando o token gerado com o login do usuário, e o usuário só precisa adicionar na caixa branca o ID do condomínio, que a API faz o resto do trabalho por debaixo dos panos, sem que o usuário precise ficar preocupado de como está funcionando.

## 5. CONCLUSÃO

Após analisar e realizar os testes, foram observados que o uso do framework Spring foi de grande ajudar a resolver o problema de gerenciamento de condomínio, sendo assim muitos condomínios vão adotar a utilização de ferramentas web ou aplicativos para auxiliar nos problemas cotidianos que foram abordados ao decorrer do artigo.

Portanto com o avanço da tecnologia, e de como estão a construção de API pode ajudar mais os condomínios, com outros serviços e fazendo se como um auxiliar em apoiar tanto os moradores quanto os funcionários diminuindo os desgastes e tempo ocioso anteriormente. Acredito que futuramente, o projeto vai se expandir e ter mais opções de serviços para se tornar cada vez mais completo e rico de detalhes, contudo tem que se acompanhar a evolução do Spring para não se deixar a API ficar depreciado.

Com o decorrer do desenvolvimento e estudo, vejo que pode ter melhorias na API para que fique mais robusta e complexa para outros aplicativos a consumir esse sistema. Como uma atualização de contatos frequentes para convidados que visitem frequentemente os visitantes, uma atualização para liberar motorista de aplicativo, um SAC (Serviço de Atendimento ao Cidadão) para que os clientes conversem com a administração do condomínio sobre alguma ocorrência, central de notificação que o funcionário, ou administração do condomínio mande mensagens de aviso aos moradores. E uma Atualização de criação de agendamento de eventos, permitindo ao usuário reservar espaços do condomínio.

## Agradecimento

Agradeço o professor Edison Lima que me orientou e ajudou nas etapas da construção da API, explicando e me guiando em cada problema que ocorreu em determinadas etapas. A minha amiga Julyana Correa que ficou responsável pela startup e ao aluno e amigo Marcelo Correa por desenvolver e consumir a minha API REST.

## Referências

Admingr1d. **Swagger: entenda o que é e como usar**, gr1d,2022. Disponível em: https://gr1d.io/2022/04/15/swagger/.Acesso: 15 out. 2022.

ALEXANDER S. Gillis. **programação orientada a objetos (OOP)**, Techtarget, 2021. Disponível em https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP. Acesso em: 17 out.2022.

AUTHO,Editor. **Introdução aos tokens da Web** JSON.jwt, 2021. Disponível em: https://jwt.io/introduction. Acesso em: 14 out.2022.

IBM, Documentação. **Diagramas de Caso de Uso**, IBM,2021. Disponível em: https://www.ibm.com/docs/pt-br/rsm/7.5.0?topic=diagrams-use-case. Acesso em 18 out.2022

IBM, Documentação. **Diagramas de Classes**, IBM,2021. Disponível em: https://www.ibm.com/docs/pt-br/rsas/7.5.0?topic=structure-class-diagrams. Acesso em 02 nov.2022

LEANDRO. **O que é UML e Diagramas de Caso de Uso: Introdução Prática à UML**. Devmidia,2012. Disponível em: https://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408. Acesso em 29 out. 2022.

LIMA, Guilherme. **REST: Conceito e fundamentos**. Alura, 2020. Disponível em https://www.alura.com.br/artigos/rest-conceito-e-fundamentos. Acesso em: 15 out.2022.

REDHAT,Editor. **O que é API REST?** RedHat, 2020. Disponível em: https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api. Acesso em: 15 out.2022.

SISQUALIS,Editor. **Interoperabilidade o que significa e para que serve**. Sisqualis, 2020. Disponível em: https://sisqualis.com.br/interoperabilidade-o-que-significa-e-para-que-serve/. Acesso em: 16 out.2022.

Somday**. REST API**. Velog,2021. Disponível em: https://velog.io/@somday/RESTful-API-□□. Acesso em 20 out 2022

TECHTARGET,Editor. **Contributor Spring Framework**, Techtarget, 2021. Disponível em:https://www.techtarget.com/searchapparchitecture/definition/Spring-Framework. Acesso em: 14 out.2022.

TOLETO, Sergio. **Os 4 pilares da Programação Orientada a Objetos,** Sergiotoleto,2019. Disponível em:https://www.sergiotoledo.com.br/artigos/os-4-pilares-da-programacao-orientada-a-objetos. Acesso em 18 out 2022

# 3

## O USO DE CLEAN ARCHITECTURE COM MONOREPO PARA O DESENVOLVIMENTO DE UM APLICATIVO NO FLUTTER CONSUMINDO SERVIÇOS DE UMA API REST

*USING CLEAN ARCHITECTURE WITH MONOREPO TO DEVELOP AN APPLICATION IN FLUTTER CONSUMING SERVICES FROM A REST API*

Marcelo da Conceição Correia[1]

Edilson Carlos Silva Lima[2]

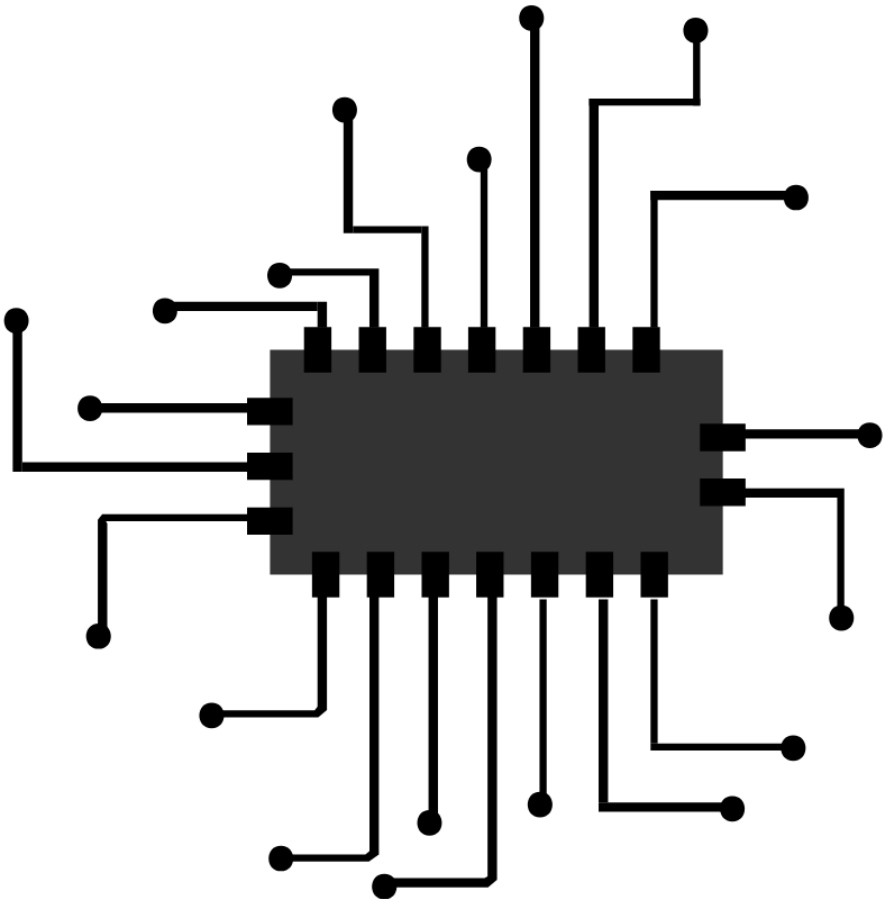
Yonara Costa Mangalhães[3]

1 Engenharia de Computação – Universidade Ceum – São Luís – MA– Brasil

2 Engenharia da Computação– Universidade Ceuma – São Luís – MA– Brasil

3 Engenharia da Computação– Universidade Ceuma – São Luís – MA– Brasil

{CORREIA, Marcelo da Conceição, marcelocc.eng@gmail.com; LIMA, Edilson Carlos Silva, edilsonlima3@gmail.com; MAGALHÃES, Yonara Costa yonara.magalhaes@ceuma.br.}



**d.o.i.:**

## Resumo

Atualmente, segurança é um dos principais motivos que levam as pessoas a morar em condomínios no Brasil. Infelizmente, hoje, existem diversos riscos que demandam cuidado redobrado para evitar importunos nestes locais. Um dos principais fatores de risco em condomínios é a entrada e saída de visitantes, prestadores de serviços e até moradores. Para evitar colocar em risco esses locais é fundamental o controle de acesso e deslocamento dentro desses empreendimentos, fazendo com que todos (portaria, administração do condomínio e moradores) não negligenciam e nem coloquem em risco a vulnerabilidade da segurança de todos. Com o objetivo de desenvolver um aplicativo que facilite e auxilie nesse controle, este artigo mostra o uso do *framework* da Google, o *Flutter*. Utilizando o *Clean Architectures*, o SOLID, entre outras tecnologias, para desenvolver um *Front-End* consumindo uma *API Rest* para auxiliar na solução do problema exposto. A aplicação se mostrou eficaz para facilitar a entrada de visitantes em condomínios e assegurar um maior nível de segurança para esses locais.


**Palavras-chave:** *Flutter, Clean Architecture, SOLID, Front-end, API Rest*.


## Abstract

Currently, security is one of the main reasons that lead people to live in condominiums in Brazil. Unfortunately, today, there are several risks that demand extra care to avoid nuisances in these places. One of the main risk factors in condominiums is the entry and exit of visitors, service providers and even residents. To avoid putting these places at risk, it is essential to control access and displacement within these ventures, making sure that everyone (concierge, condominium administration and residents) does not neglect or jeopardize everyone's security vulnerability. In order to develop an application that facilitates and helps in this control, this article shows the use of Google's framework, Flutter. Using Clean Architectures, SOLID, among other technologies, to develop a Front-End consuming a Rest API to assist in the exposed problem solution. The application proved to be effective in facilitating the entry of visitors into condominiums and ensuring a higher level of security for these locations.


**Keywords:** Flutter, Clean Architecture, SOLID, Front-end, API Rest.

## 1. INTRODUÇÃO

Segurança é um dos fatores mais importantes que as pessoas buscam para morar em um condomínio. A expectativa é sempre de um local mais bem estruturado, organizado e protegido. Por isso, o condomínio deve garantir que isso deve ser oferecido. O controle de acesso de visitantes é, acima de tudo, uma questão de segurança nos condomínios e, como tal, merece atenção por parte da segurança e da administração do condomínio. Afinal, está em jogo a integridade dos moradores, dos funcionários e do próprio patrimônio coletivo. A entrada e a saída de pessoas e veículos do condomínio é um momento crítico, especialmente em grandes centros urbanos, onde o número de roubos e furtos segue elevado. Nesse contexto, o controle de acesso de visitantes é ainda mais importante, uma vez que envolve a presença de pessoas exóticas no condomínio. O trabalho precisa também ser sustentado em práticas adequadas para o controle de entrada de pessoas no condomínio. A autorização para visitantes deve ser dada, primeiramente, pelos condôminos, por meio de comunicação prévia. Todo visitante deve também ser registrado, para que sua presença seja notada quando ele entrar e sair.

Para construir esse artigo foram realizadas pesquisas bibliográficas a fim de desenvolver a aplicação Tranca (nome dado ao aplicativo que auxilia no controle de entrada e saída em condomínios) com o *framework Flutter* (um kit de desenvolvimento de *software* criado pelo Google) consumindo uma *Api Rest* com *Spring Boot*. Tranca utiliza os smartphones dos próprios moradores e porteiros, permitindo uma troca de informações instantânea, onde o morador cria convites para seus visitantes e o porteiro visualiza e confirma a entrada dele.

Neste artigo vamos abordar alguns tópicos importantes, sendo eles: no capítulo 2 discorre-se sobre a metodologia empregada, a fundamentação teórica encontra-se no capítulo 3, no capítulo 4 encontram-se os resultados e discussões, e por fim no tem-se a conclusão deste artigo no capítulo 5.

## 2. METODOLOGIA

Para a realização deste artigo foi utilizado a coleta de informações tendo como objetivo desenvolver a implementação do *Clean Architecture* e *Monorepo* no *Flutter* a fim de construir o *frontend* de uma aplicação bem estruturada respeitando os princípios do SOLID e alcançando de forma eficaz a solução do problema contextualizado. Para isso foi realizado:

- Pesquisa bibliográfica da linguagem *Dart* e o *framework Flutter* através de livros, da própria documentação e artigos na internet;
- Pesquisa bibliográfica sobre *Monorepo* e *Clean Architecture* através de livros e artigos na internet;
- Estudo sobre a construção de arquitetura e acoplamento;
- Diagramação para ajudar a entender o comportamento da aplicação;
- Desenvolvimento da aplicação;
- Pesquisa de satisfação de interface.

# 3. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, vamos abordar uma revisão literária dos assuntos essenciais para a construção de um aplicativo desenvolvido em *Flutter.* Este capítulo está dividido nos seguintes itens: 3.1 SOLID, no item 3.2 Diagramas UML, no item 3.3 o *Front E*nd, no item 3.4 o *framework Flutter*, no item 3.5 temos a *Clean Architecture*, no item 3.6 o *Monorepo* e por fim o item 3.7 com o Consumo de serviços de uma *API Rest*.

## 3.1 SOLID

Para Ugonna Thelma (2020), S.O.L.I.D. é a sigla que representa cinco princípios de programação para garantir uma estrutura de código limpa, legível e com baixo acoplamento, permitindo testes automatizados, reaproveitamento de código e menos bugs. Analisando cada princípio temos:

*Single Responsibility Principle* (Princípio da responsabilidade única): afirma que uma classe, componente ou entidade deve ter somente uma responsabilidade. Como exemplo, temos um robô com diversas funcionalidades (cozinheiro, pintor, jardineiro e motorista), sendo a forma errada de ter uma entidade, onde o certo é ter um robô (entidade) para cada funcionalidade do seu sistema, sendo ele especialista e o melhor nessa função.

*Open-Closed Principle* (Princípio Aberto-Fechado): esse princípio afirma que classes, entidades ou funções devem estar abertos para extensão e fechados para modificação.

*Liskov Substitution Principle* (Princípio da substituição de Liskov): esse princípio afirma que se temos uma classe e a partir dela criamos uma subclasse (via herança), o objeto ou instância resultante dessa subclasse tem que ser capaz de substituir o objeto ou instância da classe principal sem quebrar o código. Na prática, seguir esse princípio obriga o desenvolvedor a usar a abstração da forma correta, pois se existir um objeto ou instância que não respeite a abstração significa que a abstração não serve para ele.

*Interface Segregation Principle* (Princípio da Segregação da Interface): esse princípio afirma que clientes não devem ser forçados a depender de métodos que eles não usam ou seja classes clientes não devem implementar interfaces com "métodos fantasmas". Exemplo: dois robôs (clientes) que devem girar, rotacionar os braços e mexer as antenas, porém somente um robô possui antena. O apropriado é segregar as interfaces, de forma que o cliente que possa executar determinado método estendendo uma interface segregada para isso, respeitando os três primeiros princípios, porém dessa vez para interfaces.

*Dependency Inversion Principle* (Princípio da inversão da dependência): esse princípio afirma que um módulo não deve depender de detalhes de implementação de outro módulo diretamente, deve haver uma abstração(interface) no meio dessa interação.

## 3.2 UML

Modelar um software é essencial na construção de grandes sistemas, como Gilleanes T. A. Guedes (2018) explica no livro sobre a abordagem prática de UML: há uma diferença notável entre construir uma casa e construir um prédio de diversos andares, um pedreiro não conseguiria construir um prédio sem um projeto.

De maneira análoga, todos os softwares precisam de projetos bem definidos para que sejam desenvolvidos da melhor forma possível. A ferramenta mais comum que possibilita isso num nível global é a UML, sigla para *Unified Modeling Language* ou Linguagem

de Modelagem Unificada, pode ser definida da seguinte forma: "uma linguagem visual utilizada para modelar softwares baseados no paradigma de orientação a objetos. É uma linguagem de modelagem de propósito geral que pode ser aplicada a todos os domínios de aplicação." GILLEANES T. A. GUEDES (2018).

A UML é composta por diversos diagramas, cada diagrama é responsável por abordar o sistema (ou parte dele) sob uma perspectiva única. Neste sistema utilizamos um desses diagramas.

Um dos diagramas da UML é o Diagrama de Casos de Uso é o mais simples dentro da UML, isso se deve ao fato de que ele aborda o sistema de forma geral e até mesmo informal. Esse diagrama é geralmente utilizado na fase de análise e levantamento de requisitos, uma das primeiras fases no desenvolvimento de *software*, mas apesar disso ele é consultado durante todo processo de modelagem, podendo servir de base para os outros diagramas (GUEDES, 2018). O objetivo do Diagrama de Casos de Uso é mostrar como o sistema se comporta, identificando os atores (tudo que interage com o sistema), funcionalidades ou serviços que o sistema irá entregar para os atores consumirem, esses serviços são chamados de casos de uso.

## 3.3 *Front End*

*Front End* é a parte visual de uma aplicação, uma interface que nos permite interagir com o sistema, apesar de ser confundido com algo que o profissional de designer faz, o *front end* é obtido através da programação, ou seja, códigos criados com ferramentas específicas para obter tais interfaces da maneira apropriada para o objetivo da aplicação. Ibarra (2021) e colaboradores, em seu artigo, afirmam que:

FrontEnd cuida do estilo da página de forma que possa apresentar informações de forma amigável [dois]. O responsável pelo FrontEnd deve conhecer as técnicas de experiência do usuário proporcionar uma melhor interação entre pessoa e a página que você visita, você também deve ter conhecimento de design de interação e colocar os elementos de tal forma que o usuário possa localizá-los rapidamente e confortável (IBARRA, 2021).

## 3.4 O *Framework Flutter* e Dart

*Flutter* é um *framework* de código aberto robusto criado pela Google para o desenvolvimento de software de interface de usuário (*front end*). Lançado em maio de 2017, o *Flutter* possui uma documentação completa, mesmo sendo uma ferramenta nova, além de permitir a criação de aplicações multiplataforma que podem ser destinadas a *Web*, *Android*, *iOS*, *Windows*, *Mac*, *Linux* e *Google Fuchsia*. Versões de lançamento de aplicativos *Flutter* usam o recurso de antecipação (AOT) compilação em *Android* e *iOS*, tornando o *Flutter*, possível alto desempenho em dispositivos móveis (PRAVEEN, 2015).

Já o Dart é uma linguagem de programação orientada a objeto e fortemente tipada criada pela Google e utilizada para desenvolver o *framework Flutter*. É uma linguagem multiplataforma, além disso oferece segurança *null safetty (nula* sólida), o que significa que os valores não podem ser nulos, a menos que você diga que eles podem ser (Dart, 2017).

## 3.5 *Clean Architecture*

Habitualmente, estamos acostumados a utilizar arquiteturas como MVC (acrônimo de *Model-View-Controller*) e MVVM (*Model, View e View-Model*), mas para sistemas maiores ou que podem se tornar maiores futuramente o interessante é ter a própria arquitetura. O *Clean Architecture* (Arquitetura Limpa), não é uma arquitetura em si, mas sim, são princípios que garantem uma arquitetura mais limpa, ou seja, implementando os conceitos do *Clean Architecture* teremos menos acoplamento e consequentemente mais independência e isolação entre as camadas. Shady Boukhary e Eduardo Colmenares (2019) citam no seu artigo sobre *Clean Architecture* em Flutter: o propósito fundamental é a separação de preocupações e escalabilidade. Dividindo o sistema em camadas que separam a lógica de negócios da implementação específica da plataforma, respeitando os pilares da engenharia de *software* como a estabilidade e a isolação de módulos externos. No geral, a arquitetura de um projeto deve ser bem clara, evidenciando do que se trata o sistema em questão, esse termo é conhecido como arquitetura gritante.

Embora o nome *Clean Architecture* seja intuitivo, vale ressaltar que não existe arquitetura limpa, sempre haverá acoplamento em algum nível. Diante disso, entende-se que se busca sempre por uma arquitetura mais limpa. O MVC por exemplo, não é considerado uma arquitetura limpa, pois possui muitas camadas que tem múltiplas funcionalidades e desrespeita o primeiro princípio da responsabilidade única do SOLID, citado no item 2.1.1 deste artigo. O que pode ser feito para alcançar esse objetivo é criar sua própria arquitetura ou melhorar uma existente, de qualquer forma será algo único para o seu projeto.

## 3.6 *Monorepo*

*Monorepo* é uma forma organizacional para grandes projetos de desenvolvimento de *software* no qual permite manter vários projetos menores ou *packages* que existem dentro de um único repositório, esses *packages* são pacotes de funcionalidades que existem independente do sistema, mas que precisa de um sistema para ser implementado, normalmente esses pacotes são dependências do sistema, um *package* para verificar a conexão com a internet. Voltando ao *monorepo,* em outras palavras é uma estratégia para controle de versão bastante utilizada em grandes empresas que possuem projetos gigantes. Scott (2017) define monorepo como: "um padrão de controle de origem em que todo o código-fonte é mantido em um único repositório. Isso torna super simples fazer com que todos os seus funcionários tenham acesso a tudo de uma só vez. Basta cloná-lo e pronto".

Nem sempre é recomendado utilizar o *Monorepo*, sua implementação depende do tamanho do projeto em curto, médio ou longo prazo. Se uma aplicação será lançada inicialmente como um pequeno projeto, mas poderá futuramente se tornar um grande projeto, então deve-se dar atenção à arquitetura e o *monorepo* se encaixa nisso.

## 3.7 Consumindo serviços de uma API REST no Flutter

A API, do inglês *Application Programming Interface*, é uma interface, implementada via código e que obedece a um conjunto de padrões e rotinas, que é capaz de interligar diferentes tipos de aplicações e sistemas. Desse modo, uma API permite que qualquer cliente que possua 5 conhecimentos em relação a documentação da mesma possa consumi-la, tornando desnecessária a construção de uma aplicação que possua a mesma

lógica. Dessa forma, uma API possui um ou diversos *endpoints*, que fornecem um serviço para ser consumido por outras aplicações clientes (OLIVEIRA, 2018). Na prática, é uma interface entre sistemas, assim como no mundo real as interfaces são feitas para que o usuário consiga utilizar um sistema, a API é uma interface que permite que um sistema utilize outro sistema. Os aplicativos atuais só são possíveis porque usam APIs.

A sigla REST do inglês *Representational State Transfer* e significa Transferência de Estado Representacional. As APIs REST utilizam *Web Service* (serviços de aplicação que utilizam protocolos para a transferência de dados na web) para realizar comunicações através do protocolo Http, herdando métodos do protocolo como *POST*, *GET*, *PUT e DELETE*. Com isso podemos interagir, por exemplo, com uma aplicação de banco de dados, enviando requisições através de *urls* (SAUDATE, 2013).

No *Flutter/Dart*, a forma mais popular de consumir uma API é através de clientes http. Esses clientes são *packages* que utilizam o http e conseguem utilizar os métodos deste protocolo para enviar, alterar, deletar ou buscar dados numa outra aplicação. Os clientes recomendados pela documentação do *flutter* são os *packages* http e Dio, os aspectos de ambos são facilmente encontrados na documentação do *Flutter/Dart*.

## 4. RESULTADOS E DISCUSSÕES

Neste capítulo aborda-se o desenvolvimento da aplicação que visa solucionar o problema elencado. Dito isso, no item 4.1 encontra-se o sistema proposto, no item 4.2 aborda-se sobre a etapa de diagramação, no item 4.3 tem-se a ambientação do projeto *Flutter* com o *monorepo,* no item 4.4 está a base desenvolvida para consumir a API REST, no item 4.5 tem-se o desenvolvimento dos *MicroApps,* no item 4.6 está o aplicativo do porteiro, no item 4.7 encontram-se a simulação do aplicativo e por fim, no item 4.8 tem-se a pesquisa de satisfação de interface.

### 4.1 Sistema Proposto

A idéia proposta foi a criação de um sistema de entrada e saída mais automatizado com o uso da tecnologia, visando acelerar o processo de visitas e prestações de serviços em condomínios, ao mesmo tempo em que guarda dados cruciais sobre moradores e não moradores em um banco de dados robusto para que sejam feitos buscas e relatórios por parte da administração do condomínio, aumentando a segurança no ambiente.

A aplicação tranca foi pensada exatamente para suprir essa necessidade, uma vez que proporcionará performance à portaria, comodidade ao morador, dados consistentes de entrada e saída ao corpo administrativo do condomínio e segurança no contexto geral. Detalhando mais esse sistema, temos dois aplicativos *mobile*, um destinado ao morador e o outro destinado ao porteiro.

No aplicativo destinado ao morador, pode-se inicialmente realizar um cadastro que será enviado para a administração do condomínio, está por sua vez é responsável por verificar a solicitação de cadastro e confirmar se o aquele usuário de fato se trata de um morador daquele condomínio. Após isso o morador pode entrar com suas credenciais (e-mail e senha) e acessar as funcionalidades do sistema, como criar convites e contatos, podendo também gerenciar (com algumas restrições) seus convites e contatos criados(na diagramação isso ficará mais claro). Além disso, pode visualizar informações do seu condomínio e também as suas informações de cadastro na sua tela de perfil.


**Figura 1**. Tela de inicialização do App.

**Fonte**: Autoral, 2022.

O aplicativo do porteiro é muito parecido com o do morador, pois tem telas de login, cadastro, perfil e convites basicamente iguais. O que os difere bastante são as suas funcionalidades, uma vez que o porteiro, inicialmente, pode somente visualizar, checar e confirmar os convites de todos os moradores daquele condomínio.

As aplicações do sistema utilizam uma API desenvolvida em *Spring Boot* para se comunicar com um banco de dados SQL. Neste artigo não vamos nos aprofundar no desenvolvimento dessa API, uma vez que o objetivo é mostrar somente os desenvolvimentos da interface dessas aplicações (*front end*). Para entender como chegamos na primeira versão desses apps, vamos passar pelos alguns itens a seguir.

## 4.2 Diagramação

Foi utilizada a ferramenta de diagramação Draw.io para obter um diagrama de caso de usos que julgou necessário para visualizar o sistema. Esse diagrama foi desenvolvido com o objetivo de termos uma noção inicial do sistema de interação dos atores com as funcionalidades que cada ator teria acesso.

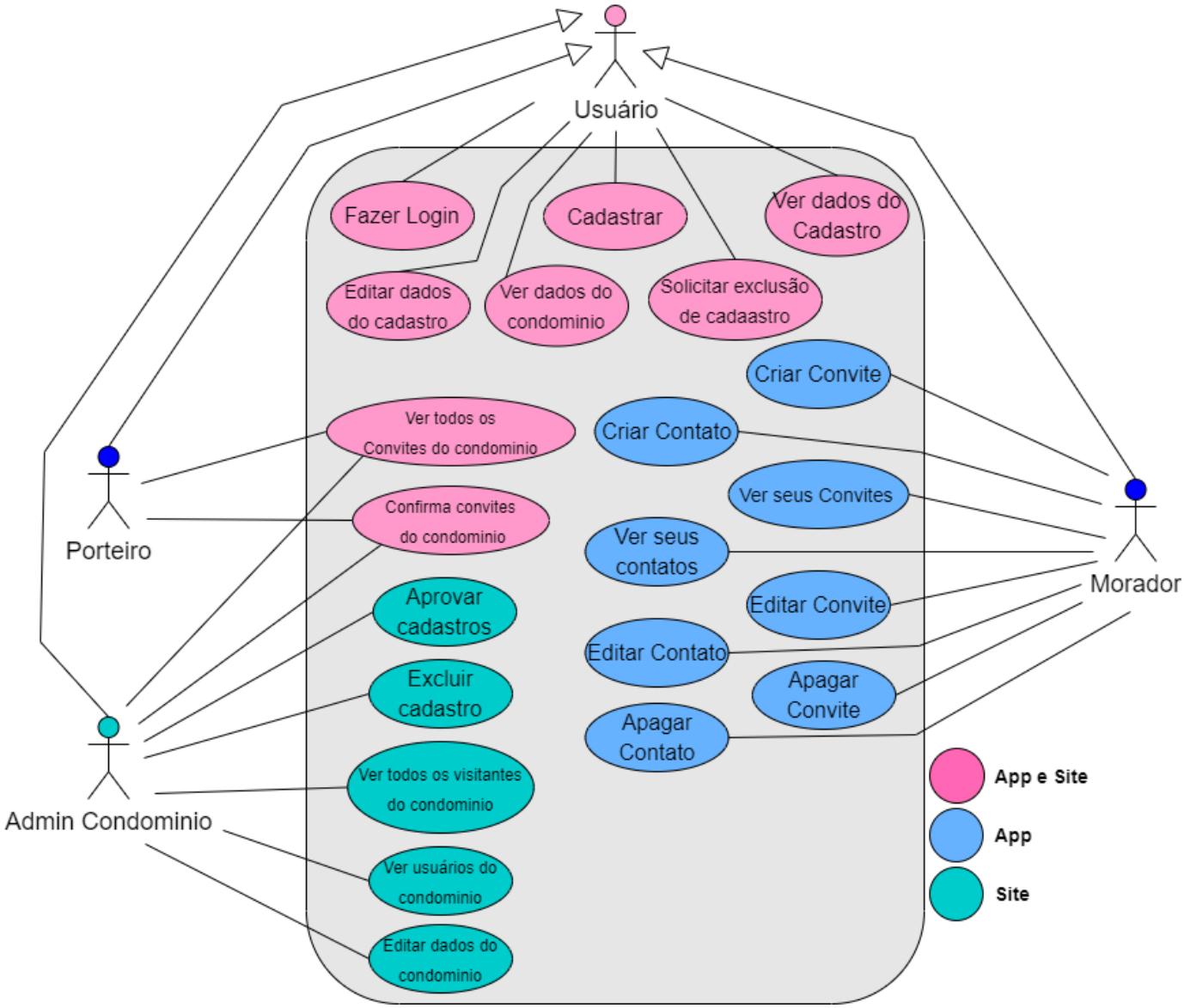



**Figura 4.** Diagrama Caso de Uso

**Fonte**: Autoral, 2022.

Observe que no diagrama temos 4 atores, sendo um o usuário e os outros três são especializações de usuário. Os casos de uso com a cor de tom azul são acessíveis através dos aplicativos, já os de cor com tom verde são acessíveis pelo site e por fim os de tom rosa podem ser acessados tanto pelos aplicativos quanto pelo site.

## 4.3 Ambientando Projeto *Flutter* com *Monorepo*

Apesar de ser um projeto pequeno inicialmente, espera-se que o sistema Tranca torne-se uma aplicação maior e mais robusta futuramente. Para garantir a integridade, escalabilidade e controle de versão foi preciso criar uma arquitetura que atendesse um provável crescimento. Logo, para aplicar uma arquitetura geral de pastas e arquivos eficaz para esse projeto, recorri ao *monorepo*.

A IDE (*Integrated Development Environment*) ou Ambiente de Desenvolvimento Integrado utilizada para o desenvolvimento dos aplicativos foi o *Visual Studio Code* junto com o *Android Studio* (como emulador). Os repositórios estão localizados no *GitHub*, uma

ferramenta bastante conhecida pela comunidade de programadores no geral.

Após criar um projeto *flutter* como casca da aplicação, foram criadas três pastas gerais dentro dele, sendo eles: *Commons, Core e Micro Apps*. Cada pasta define os tipos de *packages* que teremos dentro delas. Esses packages são miniprojetos *Flutter* com seus próprios arquivos de configurações como *pubspec, metadata* e etc, mas não possuem a classe main. dart, uma vez que essa classe se encontra no projeto geral, onde é iniciada a aplicação *mobile*.

A grande vantagem de usar os *packages* é que eles podem ser tratados isoladamente, alcançando um maior nível de desacoplamento e gerência na utilização de dependências (já que cada *package* tem suas próprias dependências).

### 4.3.1 *Commons*

O *commons*, como o próprio nome sugere, é responsável por conter todos os componentes que são comuns no sistema. No nosso caso a pasta *commons* possui um *package* chamado *dependencies*, que contém as dependências da aplicação: como a base da lógica para consumir API (interface de cliente http e https *clients*), que será utilizada em todos os *Micro Apps*. O arquivo *dependencies.dart* exporta os arquivos de serviço de *base_api,* dessa forma, pode-se aproveitar essa base para consumir a api em qualquer lugar dentro do nosso projeto.

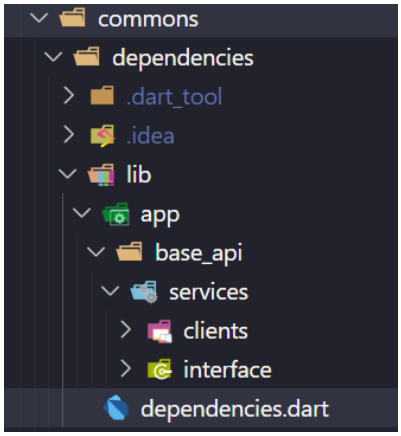



**Figura 5.** Package dependencies

**Fonte**: Autoral, 2022.

### 4.3.2 *Core*

No core está o núcleo da nossa aplicação, nele cria-se o package *micro_core* que contém o sistema gerenciamento de rotas. Esse sistema funciona de forma simples, através de três métodos consegue-se gerar, registrar e injetar rotas. Assim como no *commons*, o arquivo *micro_core.dart* exporta os arquivos que permitem o serviços de rota.

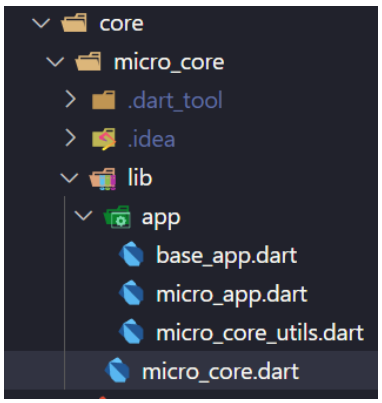



**Figura 6**. Package core

**Fonte:** Autoral, 2022.

No arquivo main, a classe *MyApp* estende do *Base App* do *micro_core,* logo é obrigatório fazer o *Override* do *baseRoutes* para a classe *main* (que direciona para a tela de *Splash*) e da lista de *Micro Apps*.

```
//rota da classe main
@override
Map<String, WidgetBuilderArgs> get baseRoutes => {
      '/splash': (_, __) => const SplashScreen(),
    };

//lista de micro_apps
@override
List<MicroApp> get microApps => [
      MicroAppLoginResolver(),
      MicroAppHomeResolver(),
      MicroAppConviteResolver(),
      MicroAppContatosResolver(),
    ];
```

**Figura 7.** Classe main

**Fonte:** Autoral, 2022.

Cada *Micro App* criado deve ser adicionado na lista de *Micro Apps* na classe *main*, o sistema só vai reconhecer e fazer a rota da forma correta se o *Micro App* estiver nessa lista.

### 4.3.3 *Micro Apps*

Nessa pasta cria-se todos os packages *Micro Apps*. *Micro Apps* são packages com contextos diferentes e não dependem de outros *micro apps*, uma vez que eles não se conhecem. A única forma de acessá-los é através de suas rotas, utilizando a chave global *navigatorKey* definida no *micro_core.* Inicialmente foram criados 4 *micro apps*: *login, home*, convite e contato.

## 4.4 Desenvolvendo base para api

A api desenvolvida permite gerenciar os dados de um banco de dados relacional SQL. Foi utilizado o *Spring Boot* para a criação da api com a linguagem java. A base para consumir uma api foi criada dentro do *package dependencies(Commons)*, uma vez que ela

será utilizada em todo o projeto. Para a construção de uma base com baixo acoplamento, criamos uma interface para um cliente http, essa interface é um contrato que deve ser seguido por qualquer cliente http no nosso sistema.

```dart
abstract class HttpClientInterface {
  Future<dynamic> post(String url, dynamic data);
  Future<dynamic> get(String url);
  Future<dynamic> delete(String url);
  Future<dynamic> put(String url, dynamic data);
}
```

**Figura 8.** Interface Http Client

**Fonte:** Autoral, 2022.

O cliente http escolhido para o sistema foi o Dio, um package citado no item 2.8. A classe *DioClient* implementa a nossa interface e pelo contrato deve ser feito os métodos de *post, get, delete e put*. Veja na figura 6 a implementação do método *get*.

```dart
String token = sp.getString('token')!;
try {
  _dio.options.headers["Authorization"] = "Bearer $token";
  final response = await _dio.get(url);
  if (response.statusCode == 200) {
    return response.data;
  } else {
    return response.statusCode;
  }
} catch (e) {
  String error = e.toString();
  if (error.contains('401')) {
    return 401;//sem autorização
  }
  else if (error.contains('403')) {
    return 403;//proibido
  }
  else if (error.contains('404')) {
    return 404;//não encontrado
  } else {
    return 0;//erro de conexão
  }
}
```

**Figura 9.** Método get com Dio Client

**Fonte:** Autoral, 2022.

No método *get* com Dio Client, injetamos o *token* de usuário para autorização da requisição e logo após é feito a requisição retornando os dados ou código de status.

## 4.5 Desenvolvendo Micro-Apps

Cada micro app tem o *micro-core* e *commons* como dependência no seu arquivo *pubspec,* além de suas dependências específicas. A classe que representa um *micro app* é identificada com o prefixo "*Micro App*" e um sufixo "*Resolver*". Nela define-se o nome do micro app e uma ou mais rotas para aquele micro app e a classe que cada uma dessas rotas representa.

```dart
class MicroAppLoginResolver implements MicroApp {
  @override
  void Function() get injetionRegister => (){};

  @override
  String get microAppName => "micro_app_login";

  @override
  Map<String, WidgetBuilderArgs> get routes => {
    '/login': (_, __)=> const LoginScreen()
  };
}
```

**Figura 10.** Criação de uma classe Micro App

**Fonte:** Autoral, 2022.

A arquitetura dos *Micro Apps* contém quatro camadas que permitem uma ótima estrutura no *front end*, sendo elas:

*Models*: Contém os modelos dos objetos do sistema (modelo de cadastro, modelo de usuário, modelo de condomínio). Nas classes modelos cria-se os atributos (que caracterizam o objeto) e os métodos *toJson* e *fromJson*, que são métodos responsáveis por converter dados de um modelo para ser enviado através da api e também converter dados recebidos de uma api para aquele modelo.

*Services*: nessa camada encontram-se os serviços de regra de negócio, esses serviços contêm classes com métodos (micro serviços) de cada serviço. Alguns exemplos de serviços que vamos encontrar são serviços de autenticação, validadores, serviços de conexão entre outros.

*Controller:* nesta camada tem-se os controladores do sistema. Os controladores conectam os serviços a camada de apresentação (próxima camada) para que os serviços possam ser usados de forma mais maleável, ou seja, o *controller* gerencia a forma como os serviços serão usados, nele controla-se os dados que são recebidos da camada *services* e disponibilizados para a manipulação nas telas.

*View:* enquanto nas camadas anteriores vemos somente códigos *dart*, na camada de apresentação temos a construção das telas de fato com o flutter. Somente aqui constroe-se a interface do usuário manipulando os dados que passaram pelas outras camadas. A implementação dessa camada é como construir um html, mas, ao invés de div tem-se os *Widgets*. No flutter *widgets* são componentes de tela e podem ser botões, linhas, colunas, ícones ou até mesmo telas inteiras. É importante lembrar que as telas e *widgets* são classes que possuem seus próprios métodos e atributos.

Nos próximos itens aborda-se como foi desenvolvido cada *micro app*, citando assuntos que julga-se relevantes para a construção de cada tela.

### 4.5.1 Login

Esse *Micro App* contém tudo relacionado a login e cadastro do usuário. Nele tem-se as classes e métodos responsáveis por buscar, tratar e exibir as informações necessárias para que o usuário possa se autenticar no sistema. Há duas telas principais nesse contexto: a tela de login e a de cadastro. Na tela de login a interface abre um formulário de e-mail e senha, para manipular os dados de entrada e enviar para a API. Na tela de cadastro utiliza-se um controlador que busca a lista de condomínios na api, pois para o usuário se cadastrar é necessário a escolha de um condomínio.

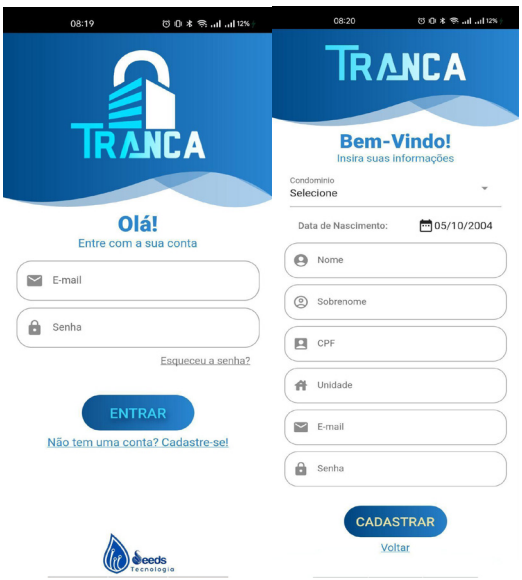



**Figura 11.** Telas de Login e Cadastro

**Fonte:** Autoral, 2022.

### 4.5.2 *Home*

Nesse *Micro App* tem-se tudo relacionado a tela inicial após o login. O contexto geral é composto por cinco telas principais dentro uma tela de navegação com *PageView* (que permite fazer um *scroll* das telas na horizontal) controlada também por uma barra de navegação animada utilizando o *package Curved Navigation Bar*. As cinco telas dentro da tela de navegação são:

- *Home*: mostra as funcionalidades principais do aplicativo, como convites, contatos, e informações sobre cada espaço de um condomínio (academia, ginásio etc).
- Perfil do Usuário: mostra alguns dados do cadastro do usuário, onde o mesmo pode fazer alterações de dados como e-mail e número de telefone.
- Informações do Condomínio: mostra os dados do condomínio do usuário, como localização, descrição, contatos.
- Notificações: mostra as notificações mais recentes, normalmente essas notificações estão relacionadas aos convites do usuário.
- Configurações: aqui encontramos algumas configurações do aplicativo, como alterações do tema, informações sobre o aplicativo e a opção de logout.

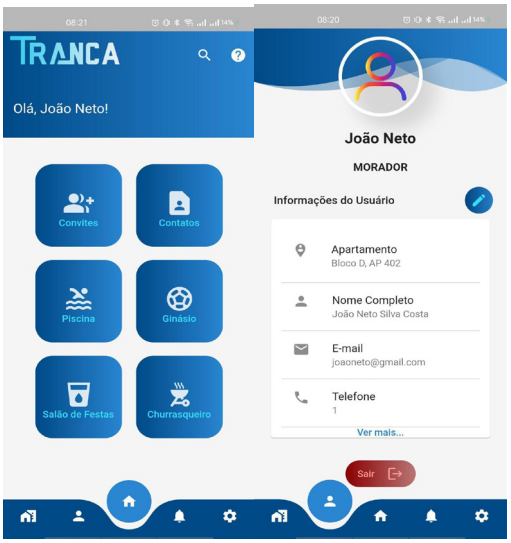



**Figura 12.** Telas de Home e Perfil

**Fonte:** Autoral, 2022.

A navegação entre as cinco telas presentes ficou fluida devido a sincronização entre os *widgets* mencionados neste item. O carregamento dos dados ocorre com um sistema de *cache* de usuário, através de banco de dados local que guarda preferências do usuário e dados não sensíveis retornados pela API.

### 4.5.3 Convites

No *Micro App* de convites tem-se tudo relacionado a convites. O controlador busca convites do usuário e insere convites no banco de dados através dos serviços de convites na camada *services.* Assim como no *Micro App Home*, na camada *view* temos uma *PageView* com uma barra de navegação para controlar cinco telas principais presentes no contexto de convites:

Novo Convite: mostra um formulário de convite requisitando nome completo e número de um documento com foto do convidado. Nessa tela é possível escolher o tipo de convidado (convidado normal ou prestador de serviços) e também salvar o contato. Uma lista de contatos é mostrada se o usuário possuir algum contato salvo, como essa tela exibe e mostra contatos, foi necessário criar serviços e controladores para buscar e salvar contatos pela API.

- Convites Ativos: mostra os convites ativos do usuário, assim que um convite é criado, seu status é de ativo. Somente os convites ativos têm as opções de apagar ou editar.
- Convites Finalizados: mostra os convites finalizados do usuário, assim que o porteiro confirma a chegada do convidado, o status do convite é finalizado.
- Convites Expirados: mostra os convites expirados do usuário, caso o convidado não vá ao condomínio no intervalo de horário definido na criação do convite, o convite expira, tornando-o um convite inválido.
- Todos os Convites: mostra os três tipos de convites anteriores.

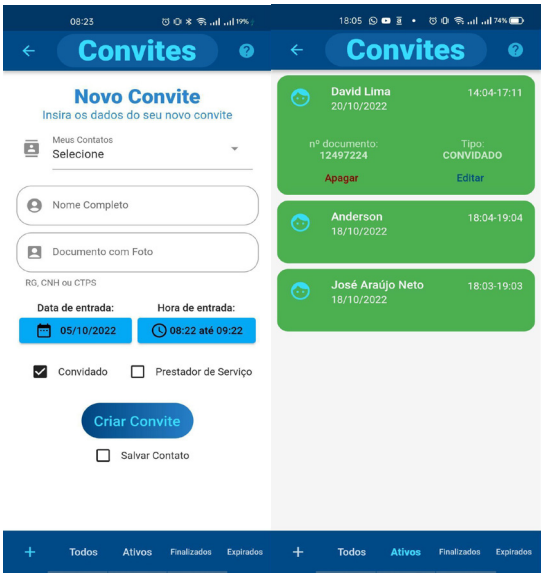



**Figura 13.** Telas de Convite

**Fonte:** Autoral, 2022.

A escolha de cores para representar o status atual de cada convite foi feita pensando em uma interface intuitiva (assim como em toda a aplicação). Cores sólidas foram aplicadas aos convites ativos, finalizados e expirados, sendo elas respectivamente: verde, azul marinho e vermelho.

### 4.5.4 Contatos

Inicialmente, esse é o *Micro App* mais simples, a tela única apenas mostra a lista de contatos salvos, ou seja, os serviços somente buscam, modificam e apagam os dados de contato do usuário.

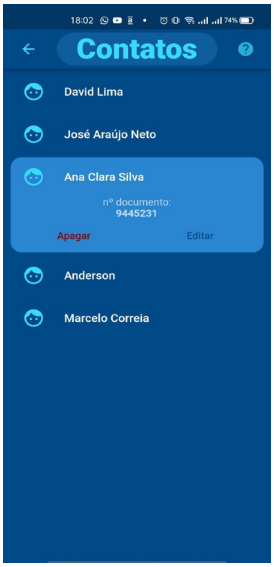



**Figura 14.** Tela de contato

**Fonte:** Autoral, 2022.

## 4.6 Aplicativo do Porteiro

O projeto para a criação do aplicativo do porteiro segue basicamente a mesma estrutura citada nos tópicos anteriores. A diferença é que nesse projeto não se tem os *Micro Apps* de Contato e *Home*, pois como o porteiro irá interagir diretamente com os convites, só precisamos dos *Micro Apps* de Login e Convites. Logo, foram reaproveitados muitos componentes, a refatoração foi simples e rápida: no login temos os mesmos componentes, inclusive de telas. Já nos Convites inserimos a tela de perfil do porteiro invés da tela de criar convite, uma vez que o porteiro não pode criar convites. A visualização dos convites se mantém dividida nas mesmas quatro telas citadas no item 3.5.3, porém o porteiro visualiza todos os convites de todos os usuários daquele condomínio e consegue confirmar (finalizar) os convites com o status de ativo.

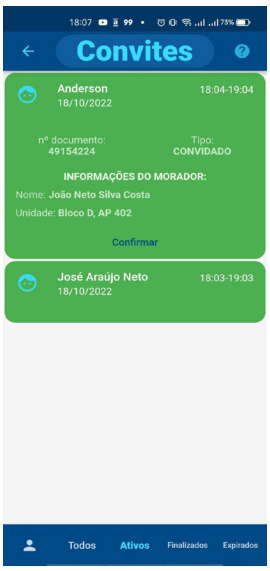



**Figura 15.** Tela de convites dos moradores

**Fonte:** Autoral, 2022.

Conforme vemos na figura, no aplicativo do porteiro vê não só as informações do convidado ou visitantes, mas também de quem o convidou(morador), dessa forma a portaria pode direcionar o convidado para o apartamento desejado.

## 4.7 Simulação

Ao final do desenvolvimento dos dois projetos, foram obtidas as primeiras versões dos aplicativos Tranca. O lançamento em plataformas de aplicativos como a Play Store não foi feito de imediato. Mas o *Flutter* possui um comando que permite criar APKs como uma versão *release.* A partir das versões *release*, consegue-se simular o uso do sistema como se estivesse sendo aplicado em um pequeno condomínio: as simulações duraram 3 dias para testar a estabilidade do sistema. A simulação foi feita com uma base de 30 usuários teste cadastrados, gerenciando convites e contatos. Como resultado tivemos o funcionamento esperado, o sistema se mostrou eficaz na sua proposta, podendo ser testado em condomínios para analisar o comportamento do usuário e satisfação do usuário, tal como o comportamento do sistema com um alto volume de usuários.

## 4.8 Pesquisa de satisfação de interface

Também foi realizada uma pesquisa, tecnicamente chamada de pesquisa de UX, com o objetivo de investigar e medir a interação do usuário com a interface para detectar pontos fortes e fracos com o objetivo de melhorar ou manter determinados aspectos da interface do usuário. A pesquisa contou com 168 entrevistas com o público alvo (pessoas inseridas em condomínios). Foram feitas 4 perguntas dissertativas, sendo as três primeiras acompanhadas pelas telas do aplicativo e a última sobre o contexto geral, veja os detalhes. Questionando se os entrevistados conseguiriam se cadastrar e logar:

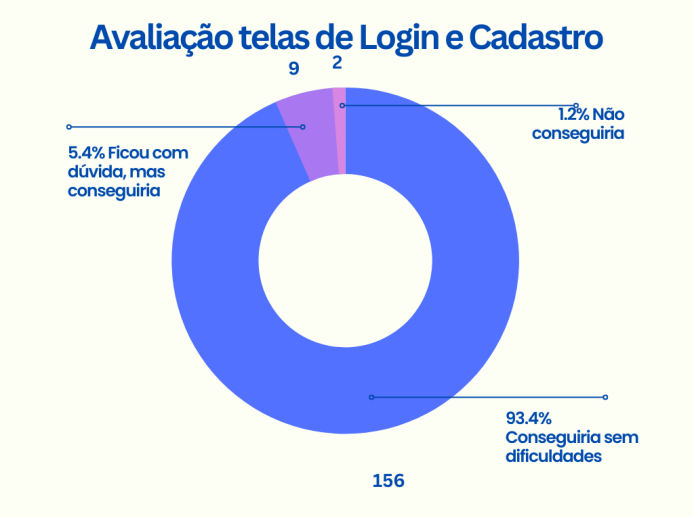



**Figura 16.** Gráfico de rosca sobre as telas de login e cadastro

**Fonte:** Autoral, 2022.

Nesse contexto, a grande maioria (93.4%) consegue utilizar as telas sem nenhuma dificuldade. As dúvidas mais comuns dos 5.4% estão relacionadas à tela de cadastro, especialmente ao campo de "Unidade" e ao uso do calendário para informar a data de nascimento. Já os indivíduos que não conseguiriam (1,2%) devido a dificuldade de leitura e interpretação do formulário de cadastro.

Questionando se os entrevistados conseguiriam acessar as telas iniciais a partir da *home page* e achar as funcionalidades:

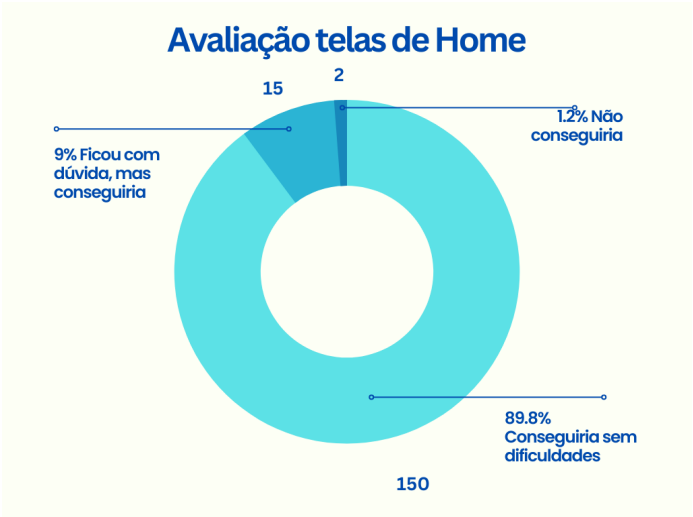



**Figura 17.** Gráfico de rosca sobre as telas de home

**Fonte:** Autoral, 2022.

No contexto das telas presentes na home, a grande maioria(89.8%) consegue utilizar as telas e acessar as funcionalidades do aplicativo sem nenhuma dificuldade. As dúvidas mais comuns dos 9% estão relacionadas a alguns ícones da tela no qual não foram mencionadas suas funcionalidades. Os (1,2%) não conseguiriam utilizar pelo mesmo motivo do gráfico anterior.

Questionando se os entrevistados conseguiriam criar um convite e visualizá-lo navegando nas telas de convites:

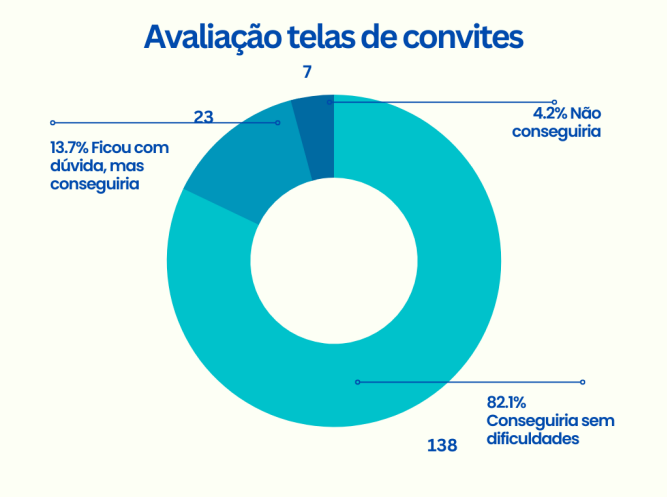



**Figura 18.** Gráfico de rosca sobre as telas de convite

**Fonte:** Autoral, 2022.

Sobre as telas de convite, a maioria dos entrevistados (89.8%) afirmam que conseguiriam utilizá-las facilmente para criação de convites, visualização do histórico de convites, acessando sem dificuldades as funcionalidades  presentes nessas telas. As dúvidas mais comuns dos 13.7% estão relacionadas ao campo de documento do convidado, mesmo sendo explicado na pergunta, alguns entrevistados acharam que teriam que enviar a foto de um documento invés de informar somente o número do documento, uma vez que no formulário está escrito "documento com foto", essa interpretação é tida como normal. Os (1,2%) não conseguiriam utilizar pelo mesmo motivo do gráfico anterior. Questionado sobre o que os entrevistados acharam do designer da interface no geral e sugestões:

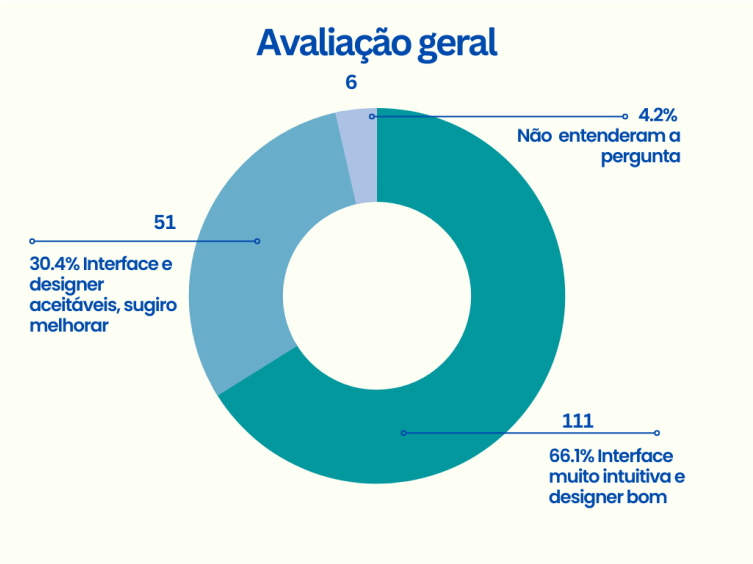



**Figura 19.** Gráfico de rosca sobre a avaliação geral da interface

**Fonte:** Autoral, 2022.

Na avaliação geral da interface, cerca de 2/3 dos entrevistados se mostraram satisfeitos e sem sugestões de melhoria, elogiando a interface com termos do tipo: interativa, intuitiva, bonita, limpa, com boas cores, ótima, entre outros. Quase 1/3 dos entrevistados acharam a interface aceitável para o uso cotidiano, mas sugeriram melhorias do tipo: mudança de paleta de cores, posicionamento de botões, implementação de modo noturno (*dark mode)*e alterações em cores em locais específicos. Os 4.2% não souberam responder a pergunta, em sua maioria por se julgarem com falta de conhecimento para avaliar a experiência de uso com a interface.

## 5. CONCLUSÃO

Conclui-se que as aplicações desenvolvidas estão aptas a se tornarem uma solução eficaz para o controle de entrada e saída em condomínios, otimizando tempo para o morador, o porteiro e o convidado. Além disso, tem a capacidade de guardar dados de visitantes que podem ser usados pela administração do condomínio da forma desejada. A interface dos aplicativos se mostrou majoritariamente eficaz com base nas pesquisas realizadas.

Dessa forma, o sistema contribui para o aumento de segurança, uma vez que a autenticidade de uma entrada de visitante é feita por parte da portaria e do morador, o convite passa a ser um documento que comprova a responsabilidade de um morador por seu convidado e de um porteiro pela análise e liberação dos indivíduos na portaria.

Com base em nossos estudos e na pesquisa realizada, enxergo várias melhorias a serem feitas para deixar o sistema mais completo e a interface mais confortável e eficiente, algumas delas são:

- Uma *feature* (funcionalidades e recursos do sistema pertencentes ao mesmo contexto) para agendamentos, permitindo ao usuário reservar os espaços do condomínio, criando eventos e direcionando convidados aos espaços diretamente no convite.
- Uma *feature* para permitir que seja criado contatos frequentes para convidados que visitem o condomínio frequentemente.
- Uma *feature* para liberação de motorista de aplicativo;
- Um site mais robusto para a administração do condomínio com estatísticas e emissão de relatórios;
- Mudanças na interface como: posicionamento de botões, recursos de acessibilidade, alteração de idioma, alteração de textos para maior clareza após a leitura;
- Ao realizar essas atualizações futuras, o sistema deve se comportar de forma mais eficaz e continuar contribuindo para a solução do problema contextualizado.

## Agradecimentos

Foi necessário muito empenho para que esse artigo fosse possível, não só da minha parte mas também de outras pessoas que me influenciaram positivamente ao longo da vida. Por isso eu gostaria de agradecer:

- Primeiramente a minha família, especialmente a minha mãe, Marly da Conceição, e ao meu pai, Manoel Correia, por todo apoio, amor e carinho.

- Aos meus amigos que participaram da minha formação direta ou indiretamente, pela amizade e companheirismo.
- Aos professores que me orientaram na construção deste artigo, sendo eles o professor Edilson Carlos Silva Lima (Coautor) e a professora Yonara Costa Magalhães (Coautora).
- Devo ressaltar a imensa gratidão às pessoas que ajudaram diretamente a desenvolver o sistema como um todo, à aluna e amiga Julyana Corrêa por fazer a pesquisa de Startup e ao aluno e amigo Carlos Eduardo Ferreira Junior por desenvolver o backend (API REST e banco de dados).

## Referências

ALLEN, Sarah; GRAUPERA, Vidal; LUNDRIGAN, Lee. **Desenvolvimento profissional multiplataforma para Smartphone: Iphone, Android, Windows mobile e Blackberry**. 2012, Alta Books; 1ª edição.

ANICHE, Mauricio. **Orientação a Objetos e SOLID para Ninjas: projetando classes flexíveis**. 2015, Casa do Código, capítulo 1.

BOUKHARY, Shady; COLMENARES, Eduardo. **A Clean Approach to Flutter Development through the Flutter Clean Architecture Package**, 2019, IEEE, CSCI, ISBN 978-1-7281-5584-5/19/$31.00.

DART. **Documentação Oficial**. [online] acessível em <https://dart.dev/overview> (Acessado em 03 de Novembro, 2022).

GUEDES, Gilleanes. **UML 2: Uma Abordagem Prática**. 2018, Novatec Editora, 3ª Edição, capítulo 1.

IBARRA, Suzana; e col. **Ferramentas e tecnologias para desenvolvimento web do FrontEnd ao BackEnd**. XXIII Workshop de Pesquisadores em Ciência da Computação. 15 e 16 de abril de 2021 RedUNCI - UNdeC. ISBN: 978-987-24611-3-3.

OLIVEIRA, Alvaro. **Construção de Aplicações Distribuídas Utilizando-se de APIs REST**. Universidade do Estado do Rio Grande do Norte (UERN), 2018. Acessível em: <https://di.uern.br/tccs2019/html/ltr/PDF/014006456.pdf> (Acessado em 03 de Novembro, 2021).

PRAVEEN, Anagha; e col. **Conference Room Booking Application using Flutter**, 2015, IEEE, ICCSP, ISBN 978-1-7281-4988-2/20/$31.00.

SAUDATE, Alexandre, **REST: Construa APIs inteligentes de maneira simples**, 2013, Casa do Código, capítulos 1 e 2.

SCOTT, Patrick Lee. **Mono-repo ou multi-repo?** [online], acessivel em: <https://medium.com/@patrick-leet/mono-repo-or-multi-repo-why-choose-onewhen-you-can-have-both-e9c77bd0c668> (Accessado: 03 de Novembro, 2022).

THELMA, Ugonna. **The S.O.L.I.D Principles in Pictures**. Acessível em: medium.com/backticks-tildes/the-s-o-l-i-d-principles-in-pictures-b34ce2f1e898. Publicado: maio/2020 (Acessado em 10/10/2022).

# 4

## A METODOLOGIA SCRUM COM AS FERRAMENTAS DE DESENVOLVIMENTO DE UMA STARTUP COMO ESTUDO PARA UM MODELO DE NEGÓCIO DE PUBLICAÇÕES CIENTÍFICAS EM UMA INSTITUIÇÃO DE ENSINO

*THE SCRUM METHODOLOGY WITH THE DEVELOPMENT TOOLS OF A STARTUP AS A STUDY FOR A BUSINESS MODEL OF SCIENTIFIC PUBLICATIONS IN AN EDUCATIONAL INSTITUTION*

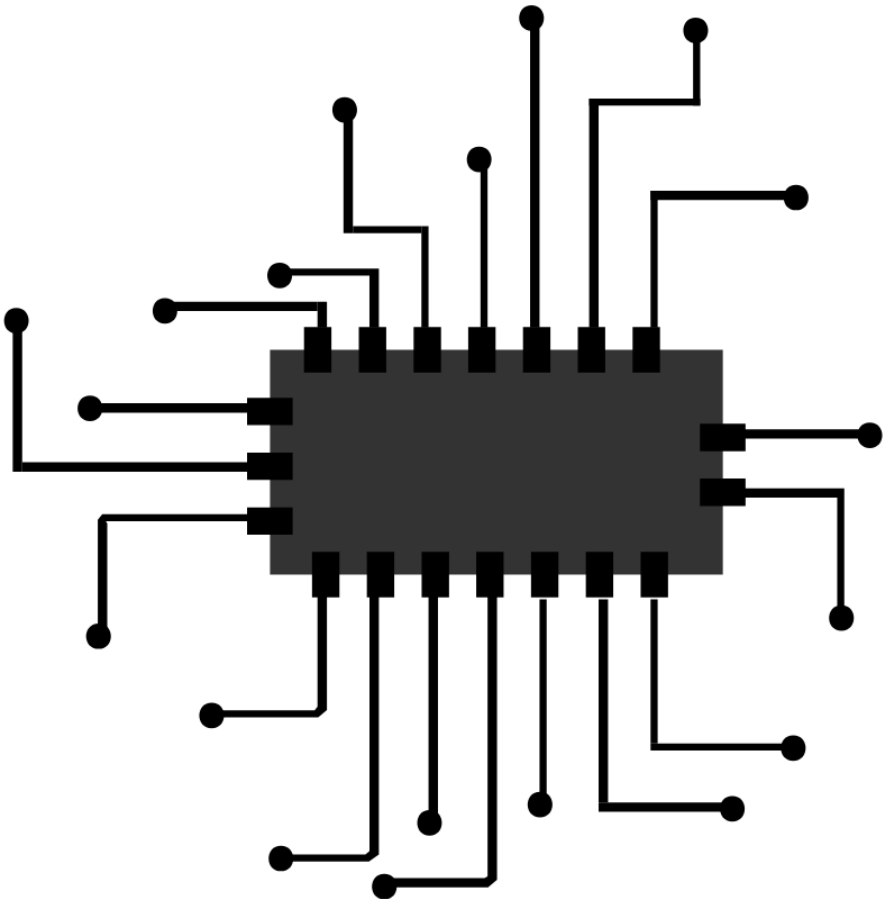
Mariane Soares dos Santos[1]

Edilson Carlos Silva Lima[2]

1 Engenharia da Computação –Universidade Ceuma (UniCEUMA) – São Luís – MA – Brasil

2 Engenharia da Computação– Universidade Ceuma (UniCEUMA) – São Luís – MA – Brasil

{SANTOS, Mariane Soares dos, marianess0310@gmail.com; LIMA, Edilson Carlos Silva, edilsonlima3@gmail.com}



**d.o.i.:**

## Resumo

Um problema latente dentro das instituições de ensino superior no Brasil é a dificuldade de acesso a trabalhos científicos dentro das próprias universidades, o que pode desencorajar e/ou dificultar o desenvolvimento de novas pesquisas. Esse aspecto impacta a ciência e a tecnologia do nosso país. Considerando essa problemática, o presente trabalho apresenta o processo de criação de uma *startup*, com base em metodologia ágil, voltada à construção de um repositório institucional. No processo de desenvolvimento do projeto, foram utilizadas as seguintes ferramentas: o *Scrum*, para o gerenciamento do projeto; a Árvore de Problemas, o Mapa de Empatia, o mapeamento do mercado com *TAM*, *SAM*, *SOM*, o Canvas da proposta de valor e o MVP para validação da plataforma. Como principais referências temos: Ken Schwaber e Jeff Sutherland, Bruno Souza, Douglas Silva, Vitor Leite, Daniel Pereira e Eric Reis.



**Palavras-chave:** *Scrum*, *startup*, árvore de problema, mapa de empatia, canvas, MVP.



## Abstract

A latent problem within higher education institutions in Brazil is the difficulty of access to scientific papers within the universities themselves, which can discourage and/or hinder the development of new research. This aspect impacts science and technology in our country. Considering this problem, this paper presents the process of creating a startup, based on agile methodology, aimed at building an institutional repository. In the project development process, the following tools were used: Scrum, for project management; the Problem Tree, the Empathy Map, the market mapping with TAM, SAM, SOM, the Canvas of the value proposition, and the MVP for platform validation. As main references we have: Ken Schwaber and Jeff Sutherland, Bruno Souza, Douglas Silva, Vitor Leite, Daniel Pereira, and Eric Reis.



**Keywords:** *Scrum*, *startup*, *problem tree, empathy map, canvas,* MVP**.**

## 1. INTRODUÇÃO

A dificuldade de acesso a pesquisas realizadas dentro das próprias instituições de ensino superior (IES) inviabiliza a consulta a produções científicas por discentes e docentes. Isso se dá em razão de essas produções serem publicadas em várias plataformas e revistas diferentes, tornando o sucesso na busca por esses trabalhos praticamente nulos. Várias universidades brasileiras vêm perdendo pontos na avaliação dos cursos pelo Ministério da Educação (MEC) desde que repositórios institucionais próprios se tornaram um critério de avaliação (INEP, 2017). Diante disso, o objetivo do presente artigo é propor uma plataforma que facilite o acesso a publicações acadêmicas nas Universidades, incentivando assim o crescimento e promovendo maior qualidade das produções científicas. Também se busca incentivar o uso de metodologias ágeis e promover a plataforma como um modelo de negócio autossustentável e escalável.

Ao começar um projeto, é muito comum agir somente por paixão e impulso. Estar comprometido é essencial, básico, mas é preciso olhar para a ideia de um ponto de vista mais amplo e prático. No processo de criação de uma *Startup* também é importante se dedicar a pontos como: Faça uma avaliação fria e objetiva de sua ideia; Busque *feedback* sincero; Construa um Mínimo Produto Viável; Construa uma identidade; Crie um plano para aquisição e conquista de clientes, para mostrar se o produto tem aceitação e crescimento no nicho de mercado proposto. Para Reis (2012, p. 26), "uma *startup* é uma instituição humana projetada para criar novos produtos e serviços sob condições de extrema incerteza". Na fase do "Construir", é elaborado o MVP (Mínimo Produto Viável), para testar se a solução proposta será bem recebida pela comunidade e qual a viabilidade real de valor do produto no mercado.

Ainda para Reis (2012, p. 24), o conceito de uma *startup* também pode ser visto como um portfólio de atividades e o desafio do empreendedorismo é equilibrar todas as atividades. Pelo *Framework Scrum*, como metodologia de gerenciamento ágil, e com a aplicação de ferramentas adotadas no desenvolvimento inteligente de uma *startup*, no decorrer deste artigo, iremos colocar em prática os conceitos como: Scrum, Árvore de problemas, Mapa de Empatia, *TAM*, *SAM*, *SOM*, Canvas da proposta de valor, MVP, orientando os esforços efetivamente para a criação de uma *startup* de sucesso. Esse artigo apresenta primeiramente os conceitos das ferramentas citadas acima, em seguida mostra a aplicação de cada uma delas dentro do projeto e por fim faz a validação da solução desenvolvida para a dor em questão. O projeto tem como principais referências: Ken Schwaber e Jeff Sutherland, Bruno Souza, Douglas Silva, Vitor Leite, Daniel Pereira e Eric Reis.

Este artigo fez uso da pesquisa descritiva, a fim de validar a plataforma Guarani como uma solução para a problemática citada. Para sua escolha, foi levado em conta que usar essa técnica possibilita padronizar a coleta de dados (GIL, 2002). Como procedimento metodológico, fez-se uso do estudo de caso, podendo assim explorar esse contexto da vida real, realizando uma investigação, e desenvolver uma análise imparcial dos dados. Fazendo também o levantamento dos principais trabalhos, a fim de obter fundamentação sólida e ampla sobre o assunto.

## 2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, será apresentada uma revisão literária de algumas ferramentas essenciais para o desenvolvimento de uma *startup*, que adote como base as metodologias

ágeis. Divide-se nos seguintes tópicos: 2.1 *Scrum;* 2.2 Árvore de Problemas; 2.3 Mapa de Empatia; 2.4 *TAM*, *SAM*, *SOM;* 2.5 Canvas da Proposta de Valor; e, por fim, 2.6 o MVP.

## 2.1 Scrum

*Scrum* é um *Framework*[1] leve, que ajuda pessoas, times e organizações a gerar valor por meio de soluções adaptativas para problemas complexos. A estrutura do *Scrum* é propositalmente incompleta, apenas definindo as partes necessárias para implementar a teoria *Scrum*. O *Scrum* é construído sobre a inteligência coletiva das pessoas que o utilizam. Ele emprega uma abordagem iterativa e incremental, para otimizar a previsibilidade e controlar o risco. *Scrum* envolve grupos de pessoas que, coletivamente, possuem todas as habilidades e conhecimentos necessários para fazer o trabalho e compartilhar ou adquirir essas habilidades conforme necessário (SCHWABER; SUTHERLAND, 2020).

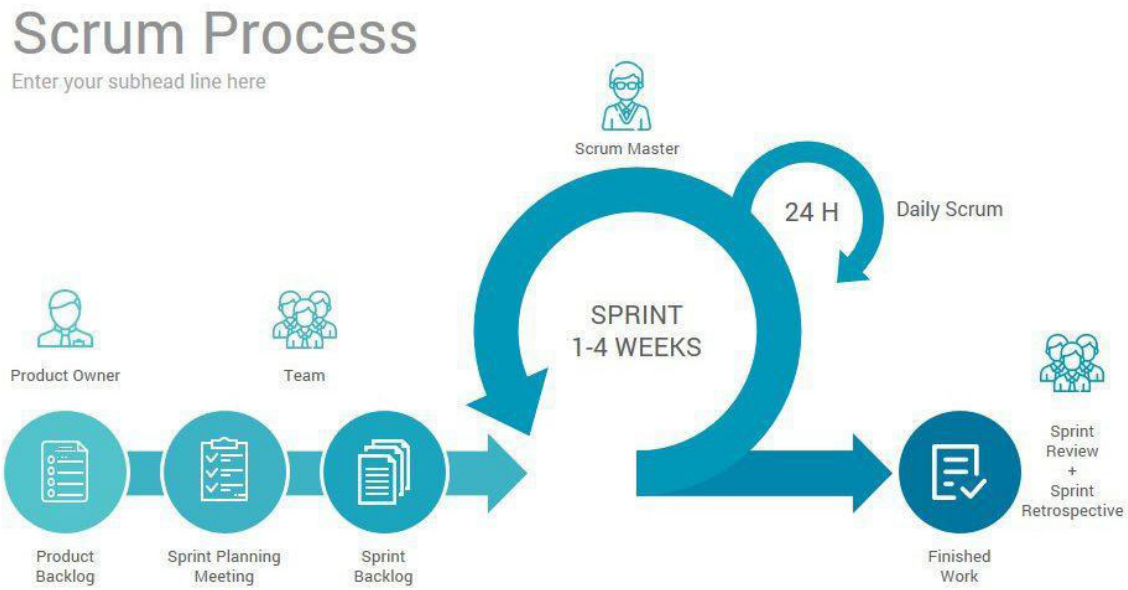



**Figura 1:** processos da metodologia ágil *Scrum*.

**Fonte:** Trainning Education, 2021.

- ***Backlog*:** lista de tarefas que devem ser realizadas pelas equipes para se desenvolver o produto ou serviço desejado (VEYRAT, 2017).
- ***Sprint*:** um período de tempo em que se completam alguns conjuntos de tarefas do *backlog* com o objetivo de consolidar um avanço incremental no produto ou serviço (VEYRAT, 2017).
- **Dono do Produto:** pessoa responsável por defender os interesses dos clientes ou usuários finais, além de definir quais tarefas serão alocadas no *Backlog* (VEYRAT, 2017).
- ***Scrum Master*:** encarregado de ser um guardião da metodologia *Scrum* e se certificar de que ela está sendo seguida corretamente (VEYRAT, 2017).
- ***Daily Scrum*:** reuniões diárias, geralmente pela manhã, em que os membros da equipe falam sobre seus progressos no dia anterior e o que pretendem realizar naquele dia (VEYRAT, 2017).

1 *Framework* - são estruturas compostas por um conjunto de códigos genéricos que permite o desenvolvimento de sistemas e aplicações. Um *framework* funciona como uma espécie de *template* ou modelo que, quando utilizado, oferece certos artifícios e elementos estruturais básicos para a criação de alguma aplicação ou *software*.

- **Retrospectiva:** ao final de cada *Sprint*, a equipe se reúne para avaliar seus resultados, as dificuldades superadas e planejar o próximo *Sprint*, sempre se baseando nos aprendizados do *Sprint* anterior (VEYRAT, 2017).
- **Revisão de *Sprint*:** finalizando cada sprint, a equipe realiza uma revisão para demonstrar as funcionalidades do produto (SANTOS, 2018).
- **Planejamento das *Sprint*:** é a parte em que a equipe se reúne para escolher itens do *backlog* para o próximo *Sprint* e se comprometer com os objetivos traçados (SANTOS, 2018).
- ***Scrum Team*:** *Scrum Team* consiste em um *Scrum Master*, um *Product owne* e *developers*. Dentro de um *Scrum Team*, não há sub-times ou hierarquias. (SCHWABER e SUTHERLAND, 2020).

Por se tratar de uma metodologia simples, o *Scrum* permite identificar a filosofia e como está, se a teoria e estrutura ajudam a chegar às metas traçadas e a obter valor (SCHWABER e SUTHERLAND, 2020), assim promovendo o controle e maior envolvimento da equipe, além de extrair o melhor de cada membro do *Scrum Team*.

## 2.2 Árvore de Problemas

A Árvore de Problemas é uma ferramenta que serve para identificar causas e consequências de uma situação que precisa de soluções. O objetivo dessa ferramenta é encontrar as causas dos problemas para desenvolver projetos que as eliminem (CORAL, 2009). A Árvore de Problemas é classificada no Japão como uma das sete ferramentas gerenciais para o controle da qualidade. Apesar desse caráter aparentemente elevado, ela é uma ferramenta simples, fácil de ser utilizada e apresenta vantagens em relação a algumas das demais metodologias. A Árvore de Problema é uma forma de identificar as causas de um problema (ORIBE, 2012).

Para construir a Árvore de Problemas, podem ser utilizadas várias técnicas. Dentre elas, temos a técnica do *Brainstorming*[2]. Segundo Carlos (2021), *Brainstorming* (ou tempestade de ideias) é uma técnica criativa para grupos, que serve para tentar encontrar uma solução a um problema específico. Isso é feito ao reunir uma lista de ideias apresentadas pelos membros da equipe de maneira espontânea.

Com isso, obtém-se um grande número de possíveis contextualizações, soluções e abordagens, identificando-se causas, consequências e relações entre os setores, áreas e processos. Após essa etapa, a equipe filtra os resultados, eliminando os que não se sustentarem e ampliando os que forem promissores (SOUZA, 2010).

## 2.3 Mapa de Empatia

O Mapa da Empatia é uma ilustração que traz as necessidades e as dores dos clientes e, assim, oferece a visão necessária para que as empresas se coloquem no lugar deles. Criada por Dave Gray, fundador da XPLANE, essa ferramenta tem como base o conceito de *design thinking*[3] e tem como propósito ter uma compreensão profunda de sua perso-

2 *Brainstorming* - é uma técnica utilizada para propor soluções a um problema específico. Consiste em uma reunião também chamada de tempestade de ideias (em português), na qual os participantes devem ter liberdade para expor suas sugestões e debater sobre as contribuições dos colegas.

3 *Design thinking* - é um método para estimular ideação e perspicácia ao abordar problemas, relacionados a futuras aquisições de informações, análise de conhecimento e propostas de soluções.

na em uma situação específica, por meio de reflexões sobre o que o cliente diz, faz, vê, pensa, sente e ouve, como forma de ajudar no desenho do modelo de negócio de uma empresa, para, assim, conseguir oferecer soluções sob medida para ela (SILVA, 2020).

A XPlane desenvolveu o Mapa da Empatia como parte de um conjunto de ferramentas bem específicas de design centrado no ser humano, que eles chamam de *Gamestorming*[4], concebido como uma estrutura para complementar o exercício de desenvolvimento de empatia (PEREIRA, 2017).

## 2.4 *TAM, SAM, SOM*

De acordo com Leite (2021), basicamente, *TAM*, *SAM* e *SOM* são siglas para se referir ao mercado como um todo e às fatias que você consegue atingir com seu negócio. São métricas indicadas por uma das maiores referências em capital de risco do mundo, a Sequoia Capital.

### 2.4.1 TAM

*TAM* (*Total Available Market* ou Mercado Total Disponível) é a fatia do mercado*,* que representa a soma das receitas de empresas que estão num mesmo setor. Apesar de ainda ser um número grande, ele ajuda a entender o tamanho total do setor (LEITE, 2021)

### 2.4.2 SAM

*SAM* (*Seviceable Available Market* ou Mercado Endereçável) é uma fatia do *TAM,* que representa a soma das receitas que um negócio poderia conquistar atendendo a uma categoria específica de um setor. Por ser uma métrica mais específica, ela dá uma dimensão melhor do tamanho do mercado de uma empresa (LEITE, 2021).

### 2.4.3 SOM

*SOM* (*Seviceable Obtainable Market* ou Mercado Acessível) é uma fatia do *SAM,* que representa a receita que seu negócio realmente poderia alcançar. Para isso, devem ser considerados fatores como concorrência, região e canais de aquisição (LEITE, 2021).

## 2.5 Canvas da Proposta de Valor

O Canvas da Proposta de Valor foi criado como uma forma de facilitar um desafio central das empresas em todos os lugares: criar produtos e serviços atraentes e que os clientes desejam comprar (SEBRAE, 2020).

O Canvas da Proposta de Valor é uma ferramenta que pode ajudar a criar e posicionar produtos ou serviços em torno do que o cliente realmente valoriza e precisa. É uma ferramenta que ajuda a encontrar o encaixe do produto no mercado, de forma estruturada. Em outras palavras, ele é um aprofundamento entre duas partes do Canvas: o bloco de segmentos de cliente e a proposta de valor. O Canvas da Proposta de Valor pode ser

4 *Gamestorming* - É uma forma lúdica e criativa de gerar resultados, tomar decisões, desenvolver estratégias e solucionar problemas.

usado quando há necessidade de refinar um produto ou serviço existente ou quando uma nova oferta está sendo desenvolvida do zero (PEREIRA, 2019).

## 2.6 MVP

O MVP (Produto Mínimo Viável) é "a versão de um novo produto que permite à equipe coletar a maior quantidade de informação validadas sobre os clientes, com o mínimo esforço" (RIES, 2009).

O MVP ajuda os empreendedores a começar o processo de aprendizagem o mais rápido possível. É projetado não só para responder a perguntas técnicas ou de design do produto; ele testa hipóteses fundamentais do negócio. Trata-se da maneira mais rápida de percorrer o ciclo construir-medir-aprender (do *Lean Startup*) de *feedback* com o menor esforço possível. Ele permite que uma startup obtenha dados reais para a *baseline* [5]de seu modelo de crescimento – taxas de conversão, taxas de cadastro e período de teste, valor do tempo de vida do cliente etc. (RIES, 2012).

## 3. ESTUDO DE CASO

O estudo de caso foi realizado na empresa Júnior SEEDS do curso de Engenharia de Computação da Universidade Ceuma, por meio da criação de uma *startup*, que oferta valor por meio de uma plataforma de pesquisa de trabalhos científicos. Buscando gerenciar esse projeto de forma rápida e mais produtiva possível, foi-se optado por usar a metodologia *Scrum*, por se tratar de uma ferramenta de alta adaptabilidade e inspeções frequentes. O projeto no qual foi utilizada a estrutura *Scrum* será chamado de "Tupi Guarani" a partir daqui.

O desenvolvimento do projeto foi dividido em *Sprints*, nos quais foram definidos os tópicos a serem desenvolvidos por ordem prioritária. A Figura 2 contém o detalhamento das três *Sprints*. Foi definido o fluxo de tarefas a serem desenvolvidas e em qual momento deveriam ser entregues para a aprovação do cliente.

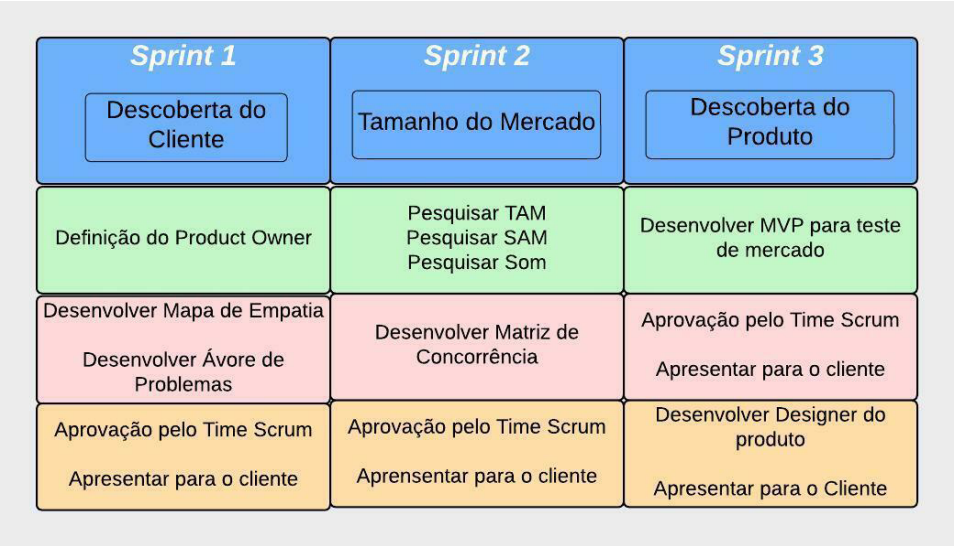



**Figura 2:** *Sprints* do projeto.

**Fonte:** Autoral, 2022.

5 *Baseline* - é a linha de base do projeto, que significa um ponto de partida muito bem definido, ou seja, uma referência composta pelos itens escopo do projeto, cronograma e orçamento.

## 3.1 Descoberta do Cliente

A descoberta do cliente é a etapa em que se identifica o cliente. Essa etapa foi desenvolvida pelo *Product Owner,* que fez o levantamento das necessidades do cliente. Para isso, foram utilizadas as seguintes ferramentas: Árvore de Problemas e Mapa de Empatia. Com a Árvore de Problemas, foi identificada uma necessidade real das universidades, que é a falta de acesso a trabalhos científicos desenvolvidos na própria instituição. Para identificar as dores do cliente, foi utilizado o Mapa de Empatia, pelo qual verificamos que a principal dor foi a falta de repositório institucional, que reúna acesso aos trabalhos desenvolvidos por cada curso. Esta etapa do projeto foi organizada na seguinte *Sprint* (Figura 3):


**Figura 3:** *Sprint* da etapa descoberta do cliente.

**Fonte:** Autoral, 2022.

## 3.2 Tamanho do Mercado

Na etapa de descoberta do tamanho do mercado, foram feitas pesquisas visando a entender o mercado total do setor e qual a fatia acessível para o nosso negócio, a curto prazo. A estratégia utilizada para obtenção da informação acerca do mercado total foi a *Top-Down* (de cima para baixo), por meio de órgãos reguladores, associações e dados de concorrentes, e a estratégia utilizada para obter a informação do mercado acessível foi a *Button-Up* (de baixo pra cima), com coleta de dados, análise de informações e potenciais clientes. A figura 4 ilustra essa etapa do projeto.

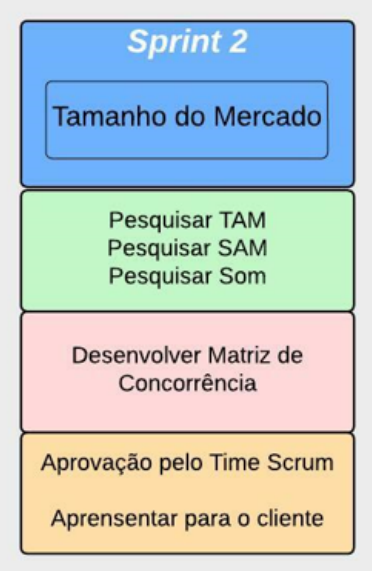



**Figura 4:** *sprint* da etapa descoberta do mercado.

**Fonte:** Autoral, 2022.

## 3.3 Descoberta do Produto

A descoberta do produto é a etapa em que os riscos são mitigados, para que se possa lançar o produto com a maior quantidade de dados relevantes possíveis. Foi feita a prototipação e, em seguida, o MVP, a fim de observar a reação do mercado e a aceitação do público. Nessa etapa, os atores envolvidos foram o *Product Owner* e o Time *Scrum*, realizando o planejamento e desenvolvendo os entregáveis, respectivamente. A figura 5 mostra detalhadamente como se deu o desenvolvimento desta etapa.


**Figura 5:** *sprint* da etapa descoberta do produto.

**Fonte:** Autoral, 2022.

A implementação do *Scrum* no projeto foi iniciada por meio da definição dos personagens do *Scrum*. O *Scrum Master* escolhido foi o Orientador responsável pela empresa Júnior. Ele tem a responsabilidade de garantir que o Time *Scrum* siga todas as regras especificadas pela metodologia.

Para *Product Owner*, foi escolhido um dos estudantes voluntários da empresa, que ficou responsável por manter o *Backlog* do produto visível, para que todos do Time tenham consciência do que há para desenvolver e qual a prioridade de cada etapa

Logo depois, foi definido o Time de desenvolvimento, composto por mais dois alunos voluntários da empresa Júnior. O time é responsável por converter o *Backlog* do produto em funcionalidades, com a maior probabilidade de aceitação pelo cliente em cada *Sprint* do projeto.

A Figura 6 é a representação gráfica de todas as etapas do projeto dentro da empresa júnior. Ao observar o gráfico, destaca-se que em algumas etapas do *Briefing*, quando não há a aprovação do cliente ou da equipe técnica, o projeto retorna à equipe de desenvolvimento, para ser ajustado.

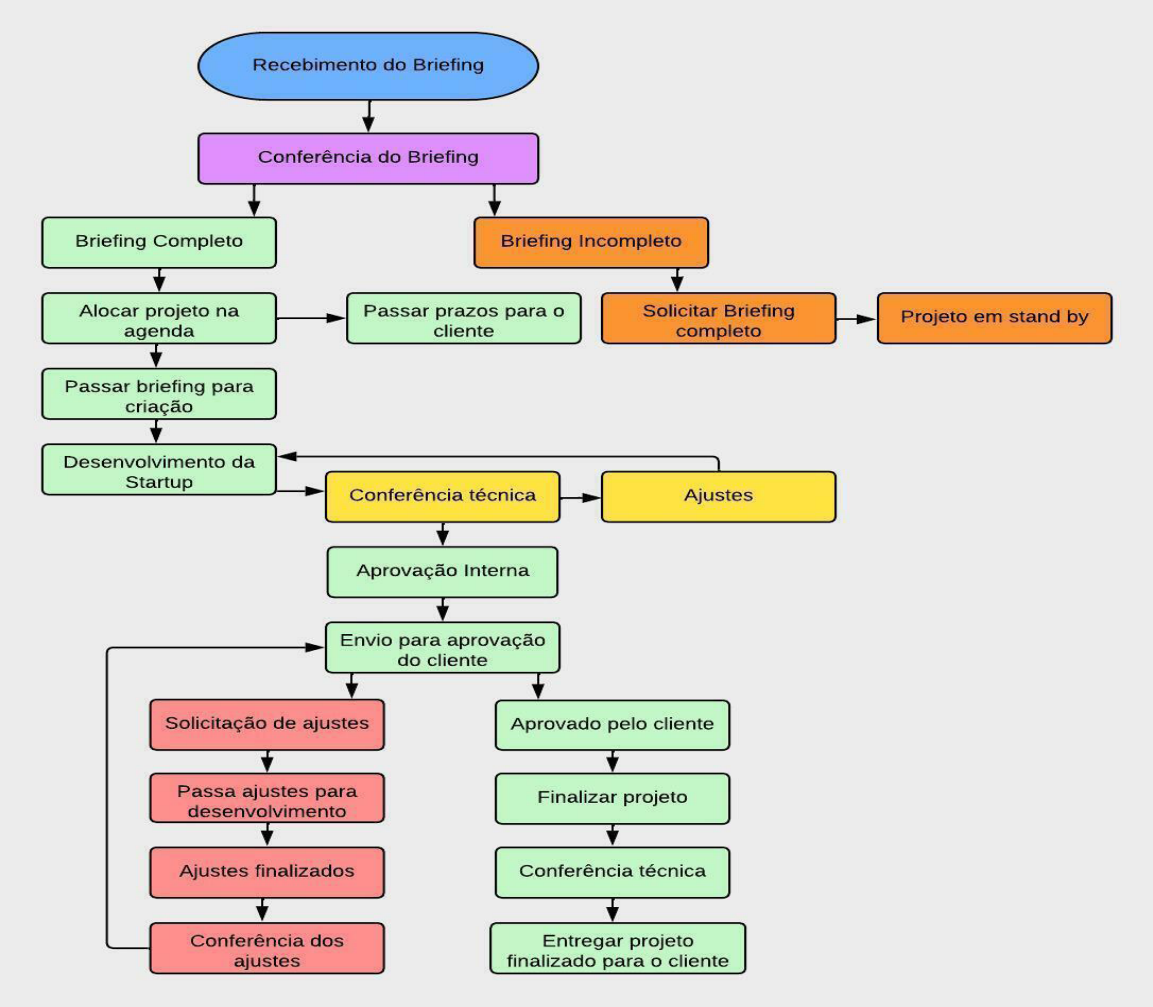



**Figura 6**: fluxograma do projeto.

**Fonte:** Autoral, 2022.

A Figura 7 refere-se à representação gráfica do *Backlog* do projeto. Nele, contém uma lista com tudo que precisa ser desenvolvido para suprir as demandas do projeto.

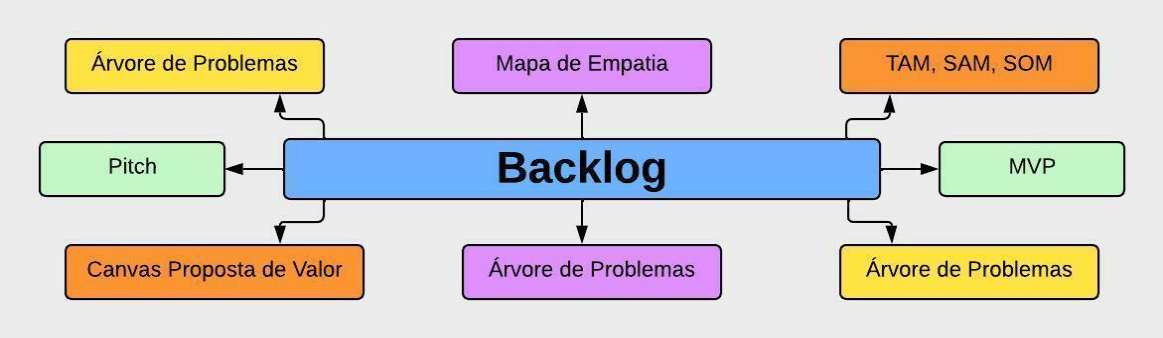



**Figura 7:** *backlog* do projeto.

**Fonte:** Autoral, 2022.

Logo após o detalhamento do projeto utilizando a estrutura *Scrum*, deu-se início ao desenvolvimento das etapas abordadas anteriormente. Começamos com o Mapa de Empatia (figura 8), com o objetivo de entender o que o nosso cliente pensa e como reage diante da falta de acesso a trabalhos científicos em sua instituição de ensino, facilitando assim a tomada de decisão a respeito dessa problemática. O Mapa de Empatia é composto por alguns questionamentos, são eles: "O que ele pensa e sente?", "O que escuta?", "O que fala e faz?", "O que vê?". Esses questionamentos facilitam a compreensão dos clientes, melhoram a comunicação e ajudam a promover conteúdo mais relevantes, promovem oportunidade para melhorar produtos e serviços, entre outros benefícios.

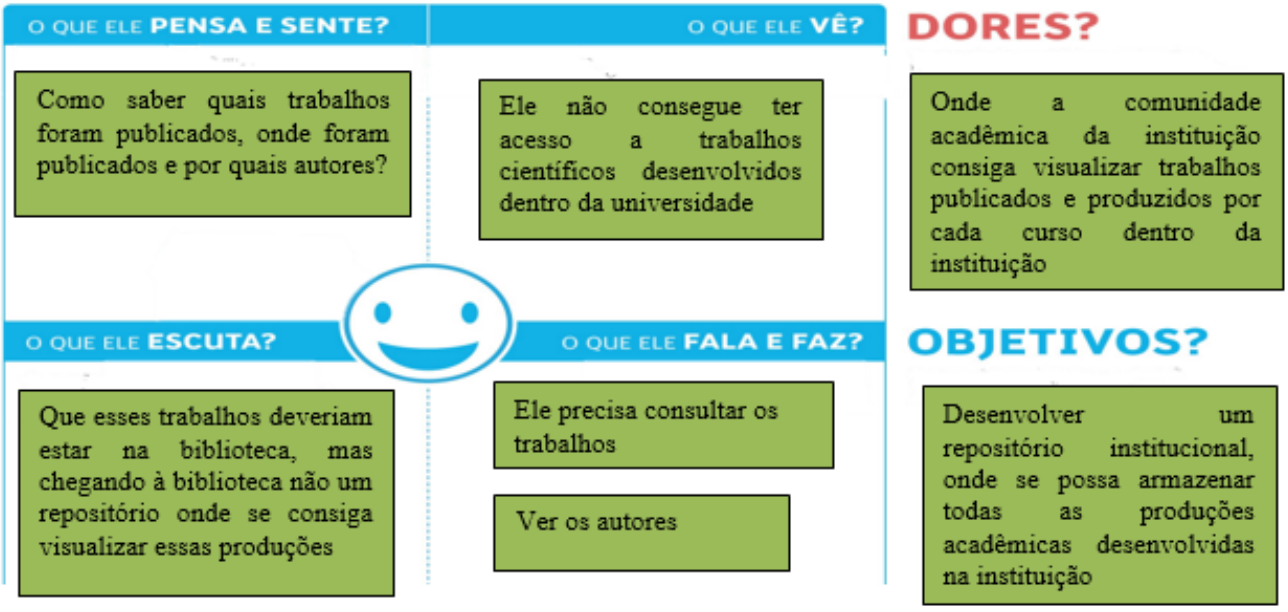



**Figura 8:** mapa de empatia da plataforma guarani.

**Fonte:** Autoral, 2022.

Logo em seguida, colocamos em prática a ferramenta Árvore de Problemas. A ferramenta foi preenchida de forma coletiva e participativa. Pode-se observar no tronco da árvore o problema central, que é a baixa acessibilidade a trabalhos científicos desenvolvidos na própria universidade. As causas deste problema podem ser observadas nas raízes da árvore, que são: a falta de investimento em pesquisa científica e a negligência por parte das IES acerca da divulgação dos próprios trabalhos científicos. Os galhos da árvore trazem as principais consequências dessa problemática, que são: a perda de conhecimento, a privação de informação, a descontinuação de trabalhos desenvolvidos anteriormente e a minimização do acesso a pesquisas, assim como mostra a figura 9.

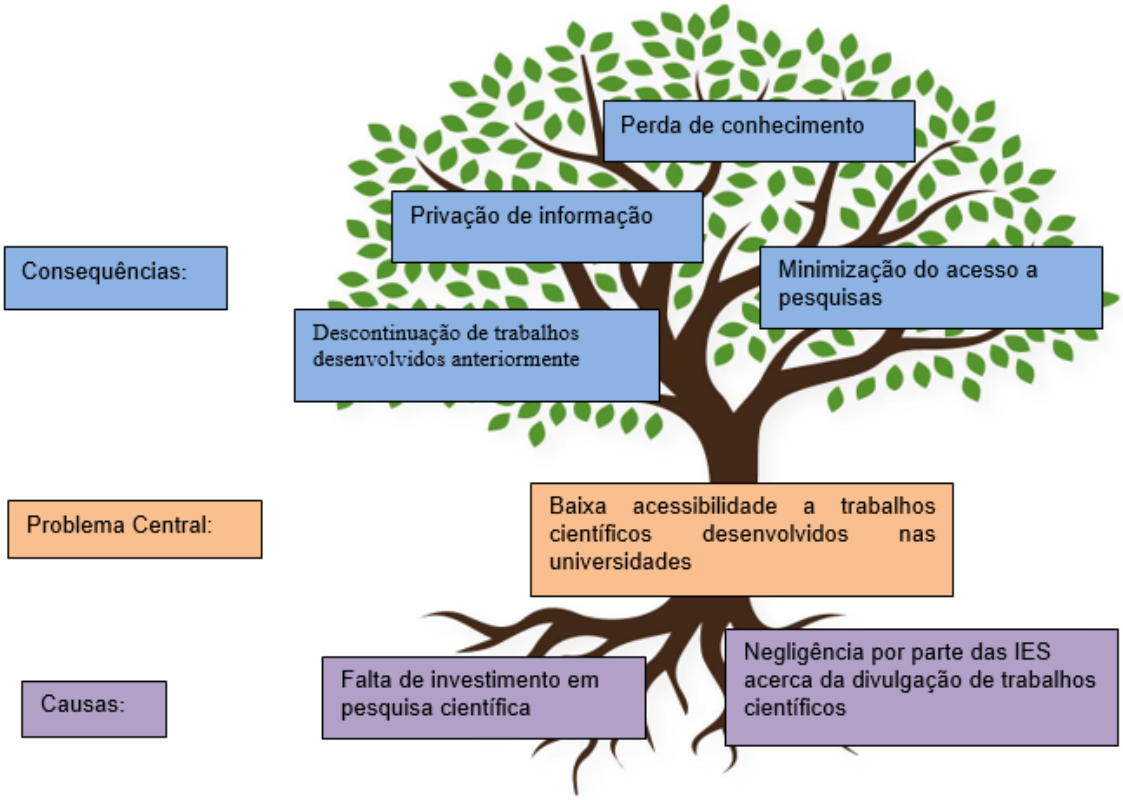



**Figura 9:** árvore de problemas**.**

**Fonte:** Autoral, 2022.

Após a conclusão da Árvore de Problemas foi dado início a pesquisa de mercado com o *TAM, SAM, SOM,* a utilização dessa ferramenta foi de extrema importância no projeto, para maior interação com o público-alvo. Por meio dela, foi possível mapear a necessidade do mercado e descobrir se a solução que estamos desenvolvendo resolve um problema real do mercado e se ela tem um mercado grande o suficiente para justificar o investimento de tempo e o dinheiro dos empreendedores. Verifica-se que esse é um mercado amplo e inexplorado, com grande número de universidades que podem ser clientes (figura 10), revelando um nicho ideal para crescimento de acordo com o modelo de *Startup*, ou seja, um negócio facilmente escalável.


**Figura 10:** tam, sam, som**.**

**Fonte:** Autoral, 2022.

**TAM:**

Em 2020, haviam 2.456 Institutos de Ensino Superior no Brasil, dentre elas 304 são públicas e 2.153 são privadas, ou seja, 87,6% dos institutos de ensino superior são privadas (INEP, 2020).

**SAM:**

Em 2021, o Nordeste possuía 594 instituições de ensino superior que ofertavam cursos presenciais e 159 cursos EAD (eram 132 em 2018, crescimento de 20,5%) (SEMESP, 2021).

**SOM:**

Em 2021, o estado do Maranhão possuía 55 IES que ofertavam cursos presenciais e 61, cursos EAD (SEMESP, 2021).

O mercado para a plataforma se mostra bastante promissor. A maioria das instituições de ensino superior, sobretudo as instituições privadas, não contam com um repositório para o armazenamento e consulta dos trabalhos científicos desenvolvidos dentro da instituição. A meta de nossa Startup é sanar essa demanda do mercado com a nossa plataforma.

A fim de testar a solução da dor principal do nosso cliente, foi desenvolvido o MVP (produto mínimo viável), que se trata de uma plataforma digital que funcionará como repositório institucional. Essa plataforma vem para sanar o problema apresentado, de modo simplificado e intuitivo, e visa a atender a necessidade de se obter referências de qualidade e inspirações para novas publicações. Possibilitando ainda a validação do produto pelo público estratégico, visando a obter *feedbacks* e sugestões de melhorias. Na figura 11, podemos observar a imagem da tela inicial da plataforma Guarani, que conta com *design* moderno e opções de navegação intuitivas.


**Figura 11:** tela inicial da plataforma guarani

**Fonte:** Autoral, 2022.

A imagem da figura 12 apresenta a tela de pesquisa. Um diferencial da plataforma são as opções de pesquisa disponibilizadas ao usuário. Ele pode filtrar os trabalhos acadêmicos, buscando pelo título, pela instituição, pelo orientador, ano de publicação, área de conhecimento, dentre outros.


**Figura 12:** tela de pesquisa da plataforma guarani

**Fonte:** Autoral, 2022.

O MVP é uma ferramenta muito importante no processo de desenvolvimento do negócio, a partir dele se consegue fazer a coleta de feedbacks com o público-alvo. O nosso modelo de negócio é B2B. Os usuários são os alunos das instituições de ensino superior e a comunidade acadêmica em geral. Por meio de nossa plataforma, os alunos terão ciência dos trabalhos que são desenvolvidos dentro da instituição, facilitando a consulta a essas produções e possibilitando a continuação delas.

## 4. RESULTADOS E DISCUSSÕES

A plataforma Guarani é um repositório *on-line* para depósito de trabalhos acadêmicos das instituições de ensino. Essa é uma necessidade cada vez maior, pois promove maior facilidade na busca e agiliza o acesso a referências acadêmicas tão importantes para o desenvolvimento da ciência e tecnologia brasileira. A plataforma vem justamente sanar essa dor, promovendo acesso rápido e simplificado a referências bibliográficas para discentes e docentes, tanto da própria instituição dos autores, quanto de outras que queiram ter acesso.

Visando à validação do produto, o MVP da plataforma foi disponibilizado para 144 universitários, que fizeram o uso da plataforma e responderam a um questionário sobre o contexto do acesso a referências bibliográficas dentro da própria instituição em que estudam e sobre a experiência na plataforma Guarani.

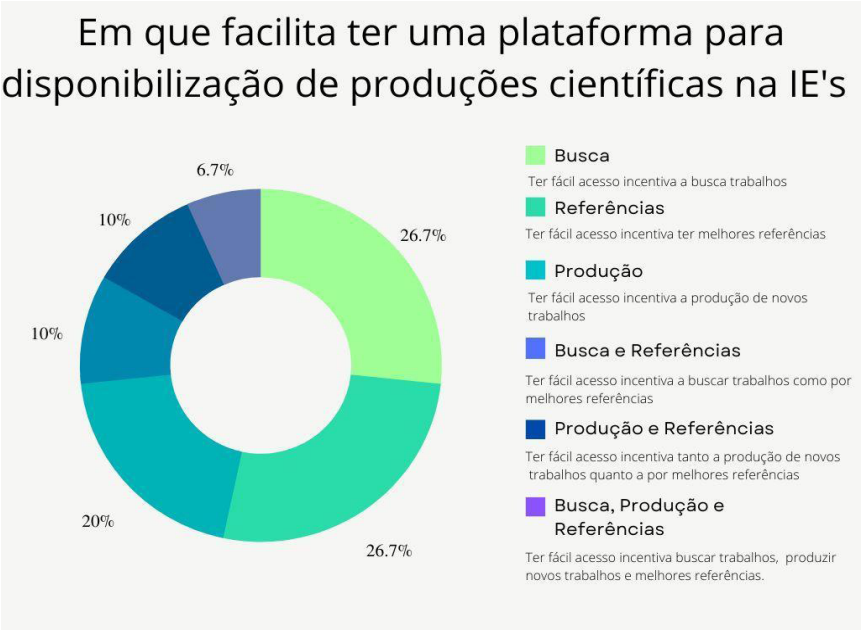
A primeira pergunta feita foi: "Caso você precisasse acessar projetos científicos desenvolvidos dentro de sua instituição de ensino, você encontraria com facilidade? Como?". A grande maioria dos entrevistados respondeu negativamente, sinalizando que não tem acesso *online* a trabalhos desenvolvidos e a outra parcela afirmou que encontra as referências. O gráfico 1 mostra a percepção dos alunos acerca dessa indagação.

**Gráfico 1:** porcentagem de alunos que têm facilidade de acessar trabalhos acadêmicos.


**Fonte:** Autoral, 2022.

Referenciar um trabalho é o que fundamenta uma pesquisa, dando embasamento teórico ao autor. Ter um repositório de fácil acesso às produções científicas facilita a busca e incentiva novas produções acadêmicas. Visto isto, na segunda pergunta do questionário, foi indagado aos estudantes sobre a importância do acesso a trabalhos acadêmicos locais, a partir da produção de cada curso. O gráfico 2 mostra a opinião dos alunos sobre a questão abordada.

**Gráfico 2:** opinião dos alunos sobre as facilidades que um repositório traria para uma IES's.




**Fonte:** Autoral, 2022

Após os questionamentos a respeito do acesso a produções científicas, foi questionado aos entrevistados (comunidade acadêmica) em relação à usabilidade da tela inicial da plataforma Guarani e da tela de pesquisa. Um *design* atrativo e *clean* facilita a busca e otimiza o tempo de pesquisa dos trabalhos acadêmicos na plataforma.

Alguns pontos positivos e negativos citados pelos entrevistados:

1. Pontos positivos:

- "Parece relativamente simples e intuitiva".
- "Sim, se mostra de fácil entendimento".

2. Pontos negativos:

- "Falta da opção de escolher por cursos como uma das principais abas da tela inicial".
- "Creio que precisa de melhoras".

Sobre a tela de pesquisa tivemos como pontos positivos e negativos apontados pelos entrevistados:

1. Pontos positivos:

- "Sim, pois há um detalhamento das informações"
- "Sim, intuitiva e de fácil entendimento"
- "Os campos de pesquisa estão orientando para cada chave de pesquisa que o usuário vai utilizar"

2. Pontos negativos:

- "Falta adicionar o curso, assim a pesquisa ficaria mais assertiva"
- "Uma busca por palavra-chave mais ampla"

Ainda acerca do MVP, foi indagado aos entrevistados quais seriam suas sugestões de melhoria para a plataforma as principais melhorias citadas foram:

- "Criar um sistema de classificação de artigos"
- "A opção de ler on-line e poder fazer o download dos mesmos"
- "Junção da tela de login com a tela de cadastro"
- "Deixar mais aparente a sessão de pesquisa"
- "Aprimorar os botões"
- "Criar um fórum de discussão para a comunidade acadêmica"

Para empresas que fazem uso do *Scrum,* o *feedback* do cliente é também uma ferramenta, que atua na melhoria do produto, no projeto e na atuação, assim possibilitando crescimento ao negócio. Os *feedbacks* recebidos pela plataforma mostraram assertividade no uso das ferramentas propostas, pois conseguiram alcançar dores e objetivos do público. A usabilidade da plataforma, assim como o detalhamento da pesquisa, destacaram-se como fatores diferenciais para os usuários. Havendo também sugestões de melhorias, apontando o interesse e aceitação da plataforma.

## 5. CONCLUSÃO

Os objetivos deste artigo foram os de gerenciar o desenvolvimento de um modelo de *startup,* utilizando uma metodologia ágil de gerenciamento de tarefas, para organizar as atividades propostas a serem concluídas, com o intuito de otimizar o máximo possível o tempo de finalização total do projeto, e desenvolver o MVP do projeto, como proposta

de solução para sanar a dor identificada no processo de elaboração. Com base nos resultados citados acima, obtidos por meio de pesquisa de campo realizada após a finalização do projeto, conclui-se que a plataforma Guarani facilita o acesso dentro das instituições a trabalhos acadêmicos. Possibilitando assim maior comodidade na procura de referências e inspirações para novas publicações. Além disso, mostra que as metodologias ágeis são uma forma de propor e gerenciar o projeto, de maneira mais eficaz. As ferramentas usadas e a divisão em *Sprints* proporcionam maior controle do projeto, sendo possível fazer um gerenciamento detalhado e obter mais engajamento do *Scrum Team.*

Pode-se aferir que, apesar de a *Startup* estar em seu estado de validação, é visível a inclinação do público-alvo à aceitação do produto no mercado. Levando em consideração o atual estado em que se encontra o projeto, o próximo passo é a consolidação da estrutura de negócio, por meio de parceiros investidores, para que seja feito o aprimoramento do MVP, que atualmente conta com um *front-end* e um *back-end* e as principais funcionalidades: cadastro, depósito de trabalhos, consultas com filtros detalhados. Espera-se ter os devidos investimentos em soluções de *marketing*, para auxiliar na divulgação de nossos serviços aos nossos respectivos clientes.

## Referências


CORAL, Eliza; OGLIARI, André; ABREU, Aline França de. **Gestão integrada da inovação: estratégia, organização e desenvolvimento de produtos.** 1.ed. São Paulo: Atlas, 2009;

DOS SANTOS, Virgilio F. M. **Scrum Team: funções e responsabilidades para ser mais eficaz.** 2018. Disponível em: https://www.fm2s.com.br/blog/scrum-team. Acesso em: 14 nov. 2022;

GIL, Antônio Carlos, **Como elaborar projetos de pesquisa.** - 4. ed. - São Paulo: Atlas, 2002 p. 42;

INEP. Ministério da Educação. **Censo da Educação Superior 2020. Notas Estatísticas**. Disponível em: https://download.inep.gov.br/publicacoes/institucionais/estatisticas_e_indicadores/notas_estatisticas_censo_da_educacao_superior_2020.pdf. Acesso em: 14 de nov. 2022;

INEP. Ministério da Educação. **Instrumento de Avaliação de Cursos Presenciais e a Distância.** Disponível em: https://www.gov.br/inep/pt-br/areas-de-atuacao/avaliacao-e-exames-educacionais/avaliacao-in-loco/instrumentos-de-avaliacao. Acesso em 20 de out. 2022;

LEITE, Vitor**. TAM, SAM, SOM: veja como calcular o tamanho de mercado de uma empresa**. maio 2021. Disponível em: https://blog.nubank.com.br/tam-sam-som-veja-como-calcular-tamanho-de-mercado/. Acesso em 25 out. 2022;

ORIBE, Claudemir Y. **Diagrama de Árvore: a ferramenta para os tempos atuais**. Banas Qualidade, São Paulo: Editora EPSE, ano XIII, n. 142, março 2012, p. 78-82;

PEREIRA, Daniel. **Canvas da Proposta de Valor. O Analista de modelos de negócios**. março 2019. Disponível em: https://analistamodelosdenegocios.com.br/canvas-da-proposta-de-valor/. Acesso em 27 out. 2022;

PEREIRA, Daniel. **Mapa de Empatia: O que é. O Analista de modelos de negócios.** [S.I], 2017. Disponível em: https://analistamodelosdenegocios.com.br/mapa-de-empatia-o-que-e/. Acesso em 23 out. 2022;

RIES, Eric. **A startup enxuta: como os empreendedores atuais utilizam a inovação contínua para criar empresas extremamente bem-sucedidas** – São Paulo: Lua de Papel, 2012, p. 24;

RIES, Eric. **A startup enxuta: como os empreendedores atuais utilizam a inovação contínua para criar empresas extremamente bem-sucedidas** – São Paulo: Lua de Papel, 2012, p. 26;

RIES, Eric. **A startup enxuta: como os empreendedores atuais utilizam a inovação contínua para criar empresas extremamente bem-sucedidas** – São Paulo: Lua de Papel, 2012, p. 70;

RIES, Eric. **Minimum Viable Product Guide**. agosto 2009. Disponível em: http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html. Acesso em 28 out. 2022;

SCHWABER e SUTHERLAND. **O Guia Definitivo para o Scrum: As Regras do Jogo**. nov. 2020. Disponível

em: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR-3.0.pdf. Acesso em 22 out. 2022;

SEBRAE. **Canvas da Proposta de Valor: o que é, para que serve e como utilizar.** 2020 Disponível em: https://www.sebraepr.com.br/wp-content/uploads/CANVAS-DA-PROPOSTA-DE-VALOR-2, Acesso em 27 out. 2022;

SEMESP, **Mapa do Ensino Superior**. 2021. Disponível em: https://www.semesp.org.br/mapa/edicao11/regioes/nordeste/#:~:text=A%20regi%C3%A3o%20possui%20594%20institui%C3%A7%C3%B5es,crescimento%20de%2020%2C5%25. Acesso em 29 out. 2022;

SILVA, Douglas. **O que é mapa de empatia e como ele pode ajudar em sua estratégia**. 2020. Disponível em: https://www.zendesk.com.br/blog/o-que-e-mapa-empatia/. Acesso em: 28 out. 2022;

SOUZA, Bruno. C. C. **Gestão da mudança e da inovação: árvore de problemas como ferramenta para avaliação do impacto da mudança.** Revista de Ciências Gerenciais. São Paulo, v. 14, n.19, p. 97-98, 2010. Disponível em: https://revista.pgsskroton.com/index.php/rcger/article/download/2583/2465. Acesso em: 20 out. 2022.

TRAINNING EDUCATION, **Curso de Scrum. Por que estudar?** 2021. Disponivel em: https://blog.trainning.com.br/projetos/curso-de-scrum/ . Acesso em: 10 set. 2022.

VEYRAT, Pierre. **As metodologias Scrum e MVP podem trabalhar juntas?** 2017. Disponível em: https://www.heflo.com/pt-br/inovacao/scrum-mvp/. Acesso em: 14 nov. 2022.

# 5

## IMPLEMENTAÇÃO DE UMA API REST USANDO O SPRING PARA UM SISTEMA DE DIVULGAÇÃO ACADÊMICA E TUTORIA

*IMPLEMENTING A REST API USING SPRING FOR AN ACADEMIC OUTREACH AND TUTORING SYSTEM*

Leonardo Silva Sousa[1]

Edilson Carlos Silva Lima[2]

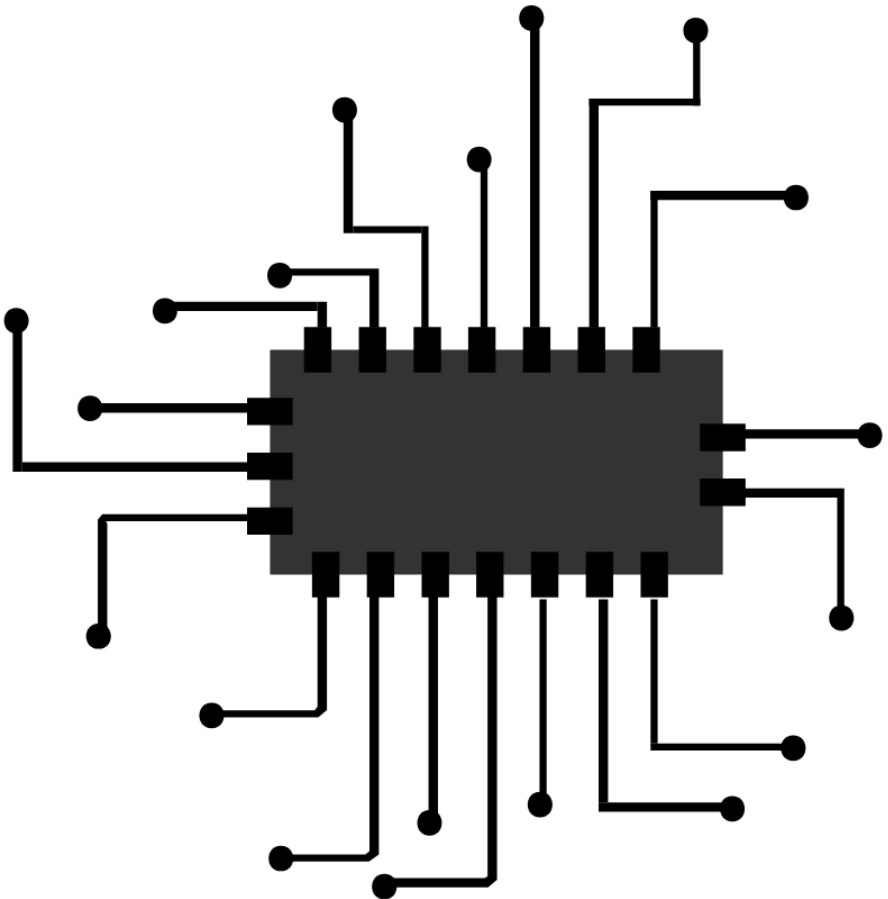
Will Ribamar Mendes Almeida[3]

1Engenharia da computação - Universidade Ceuma (UniCeuma) - São Luís - MA - Brasil

2Engenharia da computação - Universidade Ceuma (UniCeuma) - São Luís - MA - Brasil

3Engenharia da computação - Universidade Ceuma (UniCeuma) - São Luís - MA - Brasil

{SOUSA, Leonardo Silva, leonardosilva01107@gmail.com; LIMA, Edilson Carlos Silva, edilsonlima3@gmail.com; ALMEIDA, Will Ribamar Mendes, will.mendes@ceuma.com.br}



**d.o.i.:**

## Resumo

O objetivo deste artigo é criar um sistema de divulgação de trabalhos acadêmicos e solicitação de tutoria, o propósito é ajudar os pesquisadores a ter um serviço que provê um repositório capaz de centralizar seus trabalhos e com isso facilitar a sua divulgação, o projeto terá uma funcionalidade de tutoria que ajude na busca de pessoas dispostas a compartilhar seu conhecimento, viabilizando assim uma plataforma colaborativa de troca de conhecimento entre os usuários. Para a realização deste projeto, foi utilizado o *framework* do ecossistema *Spring* da plataforma *Java* para o desenvolvimento de um sistema que segue o modelo arquitetural REST. As metodologias utilizadas para a realização deste artigo foram pesquisa bibliográficas e exploratórias, buscou-se construir uma base conceitual para que então houvesse a implementação, a exploração dos temas pesquisados resultou em um projeto que segue os padrões de software. A solução desenvolvida resultou em um sistema que mapeia as publicações e o perfil do usuário, permitindo a criação de uma ferramenta capaz de filtrar de forma mais eficiente o conteúdo.


**Palavras-chave:** API REST, Spring, Web Service, Swagger.


## Abstract

The objective of this article is to create a system for disseminating academic works and requesting tutoring, the purpose is to help researchers to have a service that provides a repository capable of centralizing their works and thus facilitating their dissemination, the project will have a functionality of tutoring that helps in the search for people willing to share their knowledge, thus enabling a collaborative platform for the exchange of knowledge between users. To carry out this project, the Spring ecosystem framework of the Java platform was used to develop a system that follows the REST architectural model. The methodologies used to carry out this article were bibliographic and exploratory research, we sought to build a conceptual basis so that there was the implementation, the exploration of the researched themes resulted in a project that follows the software standards. The solution developed resulted in a system that maps publications and the user's profile, allowing the creation of a tool capable of filtering content more efficiently.


**Keywords**: REST API, Spring, Web Service, Swagger.

# 1. INTRODUÇÃO

Com a popularização da internet tornou-se mais fácil o compartilhamento de informação, independente da área do conhecimento. No entanto, essa grande quantidade de conteúdo, acaba dificultando na busca de trabalhos acadêmicos específicos e escondendo resposta para dúvidas de cunho mais técnico, que se perde na imensidão das informações na web. As questões que ficam é. Como diminuir essa dificuldade? Quais métodos adotar para facilitar a busca desses conteúdos. Este projeto iniciou-se com o intuito de resolver essas questões, oferecendo soluções para resolver estes problemas.

Portanto este trabalho tem como objetivo o desenvolvimento de uma *API REST* para um sistema que tem como função a divulgação de trabalhos acadêmicos e a criação de uma funcionalidade complementar de tutoria que visa a troca de conhecimento entre os usuários. O projeto será desenvolvido na plataforma ***Java*** utilizando o ***framework*** do ecossistema ***Spring***.

O artigo está dividido em 5 capítulos, além da introdução. Ao decorrer do artigo será mostrado uma revisão em elementos teóricos necessários para o desenvolvimento do projeto. Em seguida se sucederá uma análise do sistema e seguirá com a implementação dos conceitos apresentados no desenvolvimento da API. O capítulo 4 mostrará as principais funcionalidades do sistema e os resultados obtidos nos testes do sistema e um exemplo prático do consumo da API REST. E por último será feito a conclusão, onde reúne as considerações finais e algumas sugestões de melhorias.

# 2. FUNDAMENTAÇÃO TEÓRICA

Para o desenvolvimento deste projeto, serão abordados alguns conceitos importantes para a implementação do sistema, os assuntos foram selecionados com base na pesquisa feita para a elaboração desse projeto, o objetivo é aprofundar o conhecimento em soluções que estão sendo utilizadas atualmente no mercado de software e implementá-las no projeto.

## 2.1 *Web Service*

*Web Services* pode ser descrita como "solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas e linguagens diferentes sejam compatíveis" (OLIVEIRA, 2010).

Segundo (MAGRI, 2013), diz que a independência dos sistemas web services dado a plataforma de desenvolvimento, só se tornou possível devido a padronização que foi definida utilizando o protocolo HTTP na transferência de dados, sendo que o formato HTTP (Hypertext Transfer Protocol) e XML (Extensible Markup Language) são tecnologías padronizadas pelo W3C (World Wide Web Consortium) que podem ser usados em qualquer ambiente de programação.

> Os *Web Services* apresentam uma estrutura arquitetural que permite a comunicação entre aplicações. Um serviço pode ser invocado remotamente ou ser utilizado para compor um novo serviço juntamente com outros serviços (HANSEN, 2003, P. 1).

O *web service* permite a reutilização de sistemas já existentes permitindo a reutilização dos mesmos, fazendo com que seja possível a melhoria desses sistemas já existentes.

## 2.2 API

Segundo Silva (2021) Uma API (*application-programming-interfaces*) é um conjunto de definições e protocolos que permitem a integração entre aplicativos cuja função é compartilhar recursos e informações, mantendo-os seguros. Uma API trabalha com um conjunto definido de regras que explicam como computadores ou aplicativos se comunicam entre si. (Normalmente, essas regras são documentadas na especificação da API). Uma API utiliza requisições HTTP para fazer as requisições básicas necessárias para a manipulação dos dados. Algumas requisições básicas são:

- **DELETE**: Excluir informação;
- **GET**: Buscar os dados;
- **POST**:Criar novo dado;
- **PUT**: Atualizar um registro existente.

A API fica entre o aplicativo e o servidor web e atua como uma camada intermediária que trata da transferência de dados entre os sistemas. Segundo (REDHAT, 2020) uma API funciona com mediado entre o usuário ou clientes e os recursos ou serviços web, a API organiza e compartilha seus recursos e informações, e as mantém segura, mantendo um controle obrigatório de autenticação, paro poder determinar quem tem a acesso ao recurso e a quem ele pode ser acessado.

## 2.3 REST

REST ou *Representational State Transfer*, é um estilo de arquitetura que fornece um diretrizes entre sistemas distribuídos na web para que se comuniquem diretamente, facilitando a comunicação entre sistemas (LIMA, 2020).

Conforme o trabalho de (ALCIDES, 2021), Roy Thomas Fielding foi quem idealizou e definiu REST, em sua tese de doutorado Architectural Styles and the Design of *Network-based Software Architectures* que foi apresentada pela *University Of California*, em sua tese Fielding propôs um novo estilo arquitetural, a qual caracterizou "restrições", que foram identificadas em seus estudos iniciais de sua tese.

As restrições restringem como o servidor deve processar e responder às solicitações do cliente, isso permite com que o sistema ganhe funcionalidades não funcionais desejáveis, como desempenho, extensibilidade, simplicidade, escalabilidade, visibilidade, portabilidade e confiabilidade (CODECADEMY).

> Examinando o impacto de cada restrição, como isso é adicionado ao estilo referenciado, pode-se identificar as propriedades induzidas pelas limitações da Web. Restrições adicionais podem então ser aplicadas para formar um novo estilo arquitetural que melhor reflita as propriedades desejadas de uma arquitetura Web (FIELDING, 2000, p.76).

As características selecionadas por Fielding (2000) para compor REST são:

**1. Arquitetura cliente servidor**

A separação da arquitetura em dois ambientes é uma restrição básica para uma aplicação REST, isso permite que o cliente (consumidor do serviço) não tenha preocupação em tarefas tipo: comunicação com banco de dados, gerenciamento de cache, log etc. (PLANSKY, 2014). Fazendo com que o servidor não tenha preocupação com: interface, experiência do usuário, etc.

> O princípio por trás da arquitetura cliente-servidor é o de separação de responsabilidades. Ao separar as responsabilidades de interface das de armazenamento de dados, a utilização da arquitetura cliente-servidor permite que os componentes, o cliente e o servidor, se desenvolvam de forma independente. (RODRIGUES, 2009).

A separação entre cliente e servidor dá mais liberdade e flexibilidade para os desenvolvedores da aplicação, e possibilita com que haja uma evolução independente de ambos.

***2. Stateless* (sem estado)**

Um servidor sem estado não guarda nenhuma informação a respeito das aplicações conectadas a ele, toda informação do estado deve ficar no cliente, e as requisições mandadas pelo cliente devem conter todas as informações necessárias para que o servidor as compreenda e a processe adequadamente. Segundo (FIELDING, 2000, p.79), "Cada pedido do cliente para o servidor deve conter todas as informações necessárias para compreender o pedido".

***3. Cache***

Requisições que são requisitadas repetidas vezes, necessitando dos mesmos recursos, faz com que essas respostas sejam cacheadas, dando a responsabilidade de armazenar para o cliente para posterior utilização, evitando gastos desnecessários de processamento de recursos pelo servidor e aumentando significativamente a performance (PLANSKY, 2014).

**4. Interface uniforme**

Interface uniforme é a adoção de pequenas regras para deixar o componente mais genérico possível, e é constituída por quatro restrições:

**I. Identificação dos recursos**

No REST cada recurso deve conter uma URI (Uniform Resource Identifiers, Identificador Uniforme de Recursos) específica e coesa para poder ser acessada.

II. Representação do recurso

É a forma de como o recurso será retornado para o cliente, podendo elas ser em HTML, XML, JSON, TXT.

**III. Resposta auto-explicativa**

Informações de metadata que vai como resposta na

requisição, Algumas destas informações são: código HTTP da resposta, Host, Content *Content-Type*, etc.

***IV. Hipermídia* como Motor de Estado de Aplicativos**

Atua como um motor de navegação construído dinamicamente e incluído em representações fornecidas pela *web service*. A resposta do *HATEOAS* tem *hiperlinks* além dos dados solicitados, equipado com *auto-links* com informações claras de acesso, além de instruções de navegação entre recursos.

**5. Sistema em camadas**

O sistema deve ser dividido por camadas, isso faz com que o sistema melhore na escalabilidade e modularização, cada camada assume uma função específica no sistema. Cada camada é independente e só podem ver a camada imediata com a qual está interagindo.

**6. Código sob demanda**

Permite a extensão de código sob demanda, isso permite com que parte da lógica do servidor seja estendida para o cliente. Essa restrição é vista como opcional e pode ser exemplificada por *applets Java* e *Scripts Javascript*.

## 2.4 *Json Web Token*

*JSON Web Token* (JWT) é um padrão aberto (RFC 7519) que define uma maneira compacta e independente de transmitir informações com segurança, utilizando objetos JSON entre as partes, essas informações podem ser verificadas e confiáveis porque são assinadas digitalmente. Os JWTs podem ser assinados usando uma chave secreta (usando o algoritmo HMAC) ou usando o par de chaves pública/privada (JWT.IO, 2015).

Segundo o trabalho de Montanheiro (2017), o JSON Web Tokens é usado para transportar informações relacionadas à identidade e características do cliente. Essas informações são assinadas pelo servidor para detectar se foram adulteradas após serem enviadas ao cliente. Isso impedirá que um invasor altere a identidade ou qualquer característica.

Conforme o trabalho de Boscarino (2019), o token é criado durante a autenticação e é validado pelo servidor antes de qualquer processamento. Os aplicativos o usam para permitir que o cliente forneça ao servidor um token representando o "cartão de identidade" do usuário e para permitir que o servidor verifique com segurança a validade e a integridade do token, tudo sem estado e portátil.

## 2.5 *Maven*

Maven é uma ferramenta de construção feita em código aberto, desenvolvida pelo Apache Group para construir, publicar e implantar vários projetos de uma vez para melhor gerenciamento de projetos, essa ferramenta permite que os desenvolvedores criem e documentem estruturas de ciclo de vida (GABA, 2022). O Maven é usado principalmente para projetos baseados em *Java* para ajudar a baixar dependências que fazem referência a bibliotecas ou arquivos JAR, esta ferramenta ajuda a obter o arquivo JAR correto para cada projeto, pois pode haver pacotes separados com versões diferentes (APACHE, [s.d]).

Conforme o trabalho de (GABA, 2022) o *download* de dependências não requer a visita ao site oficial de um software diferente. O site mvnrepository.com é um site usado para encontrar repositórios em diferentes idiomas. A ferramenta também ajuda a criar a estrutura correta do projeto em *struts, servlets*, etc, o que é crucial para a execução.

Segundo Lima (2020), O Maven é escrito em Java é utilizado para construir projetos escritos em *Scala*, C#, Ruby etc. Com base no P*roject Object Model* (POM), que é um ar-

quivo XML que contém todas as informações sobre o projeto e detalhes de configuração. O POM contém a descrição do projeto, detalhes de controle de versão e gerenciamento de configuração do projeto.

## 2.6 *Swagger*

O *Swagger* é uma *framework open source* que auxilia os desenvolvedores nos processos de definir, criar, documentar e consumir APIs REST, ele visa padronizar este tipo de integração, descreve os recursos que uma API possui, como *endpoints*, dados recebidos, dados retornados, código HTTP e métodos de autenticação (SWAGGER, 2011).

Segundo (AQUILES, 2016) O Swagger é um projeto composto por algumas ferramentas que auxiliam o desenvolvedor de APIs REST em algumas tarefas como:

- Modelagem da API
- Geração de documentação (legível) da API
- Geração de códigos do Cliente e do Servidor, com suporte a várias linguagens de programação.

O *Swagger* também permite a geração de bibliotecas para clientes automaticamente em vários idiomas, para sua API explorar outras possibilidades, como testes automatizados. O *Swagger* faz isso solicitando que a API retorne um YAML ou JSON contendo uma descrição detalhada de toda a API.

# 3. ESTUDO DE CASO

Para o desenvolvimento do projeto serão utilizados os conceitos já apresentados de API REST, uma aplicação que seguirá os conceitos de API e da arquitetura REST. Ou seja, uma aplicação que se comunique com outras sem ter a dependência do sistema operacional ou a linguagem de programação, e respeitar as restrições propostas por Fielding.

O projeto será desenvolvido na plataforma java na versão 11, utilizando o *framework* do ecossistema Spring, para a persistência dos dados será utilizado o banco de dados MYSQL. Para o gerenciamento das dependências será utilizado o *Maven Apache*.

A realização dos testes foi feita durante o desenvolvimento, utilizando o programa Postman que é um software capaz de realizar requisições HTTP podendo enviar e receber resposta da API. A documentação da API será feita com o auxílio do *framework Swagger*.

## 3.1 Metodologia

Este artigo se fundamenta em pesquisas bibliográficas e exploratórias, buscando construir uma base conceitual bibliográfica, para que então houvesse a implementação desses conceitos em um projeto, buscando uma solução para resolver os problemas expostos, avaliando a mesma de forma qualitativa.

## 3.2 Arquitetura API REST

Como podemos ver na figura 1, uma API REST serve como uma ponte entre o cliente e o banco de dados, o cliente se comunica utilizando o protocolo HTTP por meio dos ver-

bos (GET, PUT, DELETE ou POST). A API REST é a responsável por receber e interpretar as requisições e fazer a leitura, criação, alteração ou deletar informações em uma fonte remota, tais verbos são chamados de CRUD (Create, Read, Update or Delete), as informações são retornadas para o cliente no formato JSON (JavaScript Object Notation) ou XML.

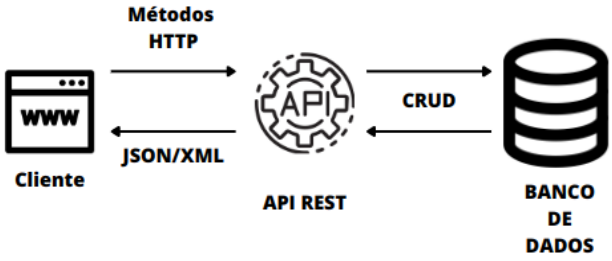



**Figura 1.** Arquitetura do modelo API REST.

**Fonte**: Autoral, 2022.

O modelo API REST permite com que o sistema seja independente da linguagem ou plataforma que o cliente venha a utilizar, isso permite com que a API se comunique com sistemas *Web*, *Mobile* ou até com outras APIs.

## 3.3 Análise da Aplicação

Antes do desenvolvimento do projeto foi feita uma análise da aplicação, onde se estabeleceram as bases e se definiram os requisitos para a construção da API REST. Durante a análise foi levantado uma série de características, tais como funcionalidades global, dados, usuários e outros elementos que são relevantes para o funcionamento do sistema.

A análise será mostrada por meio dos diagramas UML (Unified Modeling Language) que é uma linguagem que se expressa por meio de diagramas, ela pode ser usada para a visualização, especificação e documentação de sistemas complexos.

### 3.3.1 Diagrama de caso de uso

O diagrama de caso de uso é utilizado na modelagem do comportamento de um sistema, com ele é possível visualizar os casos de uso do sistema e seus atores, oferecendo uma visão externa, ele descreve cada caso de uso e como os atores o utilizam, mas não mostra como o sistema funciona internamente.

A figura 2, mostra o diagrama de caso de uso feito durante a análise da aplicação, podemos observar que existem dois tipos de atores que interagem no sistema, o diagrama mostra todas as interações dos atores com cada um dos casos de uso do sistema.

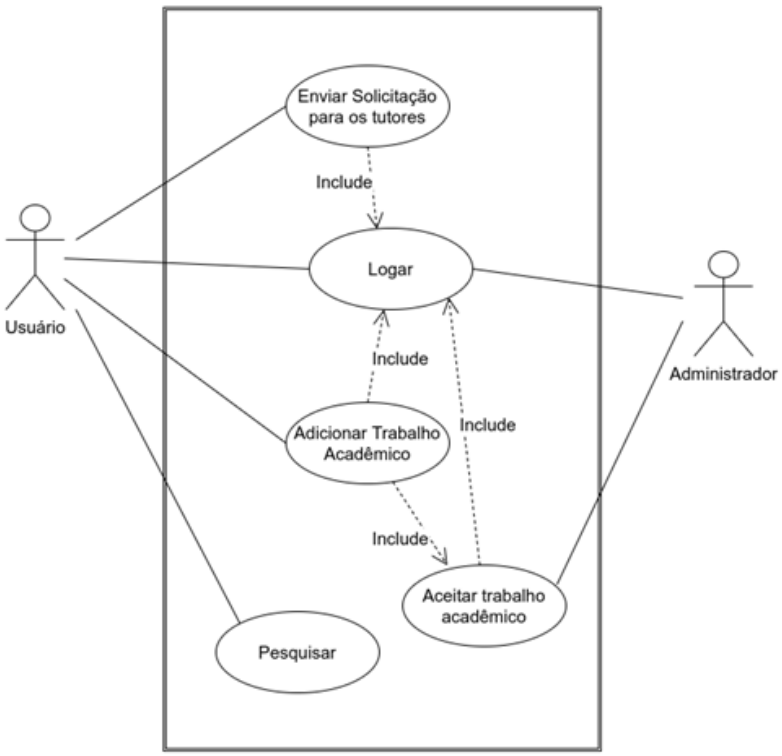



**Figura 2.** Diagrama de caso de uso.

**Fonte:** Autoral, 2022.

Todos os casos de usos que foram modelados para o sistema serão explicados logo a seguir.

1. **Pesquisar:** Este caso de uso permite que o usuário faça buscas dos trabalhos acadêmicos salvos no sistema, uma publicação só entra no mecanismo de pesquisa após passar pela análise do administrador do sistema. Esse caso de uso não requer a autenticação do usuário.
2. **Logar:** É o caso de uso que permite a autenticação do usuário no sistema, ele é pré-requisito para que o usuário execute outros casos de uso. A execução dele requer que o e-mail e a senha sejam informados.
3. **Adicionar trabalho acadêmico:** Este caso de uso permite que o usuário adicione uma nova publicação no sistema, para isso o usuário deverá estar autenticado e sua publicação será publicada com status pendente e será analisada pelo administrador do sistema.
4. **Pesquisa de tutores:** Este caso de uso permite que o usuário autenticado pesquise por tutores com habilidades em um determinado assunto escolhido pelo usuário.
5. **Aceitar trabalho acadêmico:** Este caso de uso só poderá ser acessado pelo administrador do site com devida autenticação, é ela que permite a alteração de status de uma publicação sendo elas pendente e recusada.

Com o diagrama de caso de uso é possível ver claramente as ações e comportamento dos atores que atuam no sistema, conseguimos definir as funcionalidades de cada caso de uso envolvido no sistema.

### 3.3.2 Diagrama de classe do sistema

O diagrama de classe é um tipo de diagrama estrutural utilizado no processo de modelagem de objetos e estrutura do sistema, ele é bastante útil na fase de análise do sistema ajudando a compreender os requisitos do domínio do problema e a identificar seus componentes. Com ele podemos modelar os objetos que compõem o sistema, exibimos os relacionamentos entre os objetos, as suas descrições e os serviços que eles fornecem.

Como é mostrado na figura 3, vemos os conceitos da aplicação representados por 9 classes, também é mostrado os relacionamentos de cada classe e seus atributos. Em seguida, é feita uma breve descrição de cada uma das classes envolvidas no sistema.

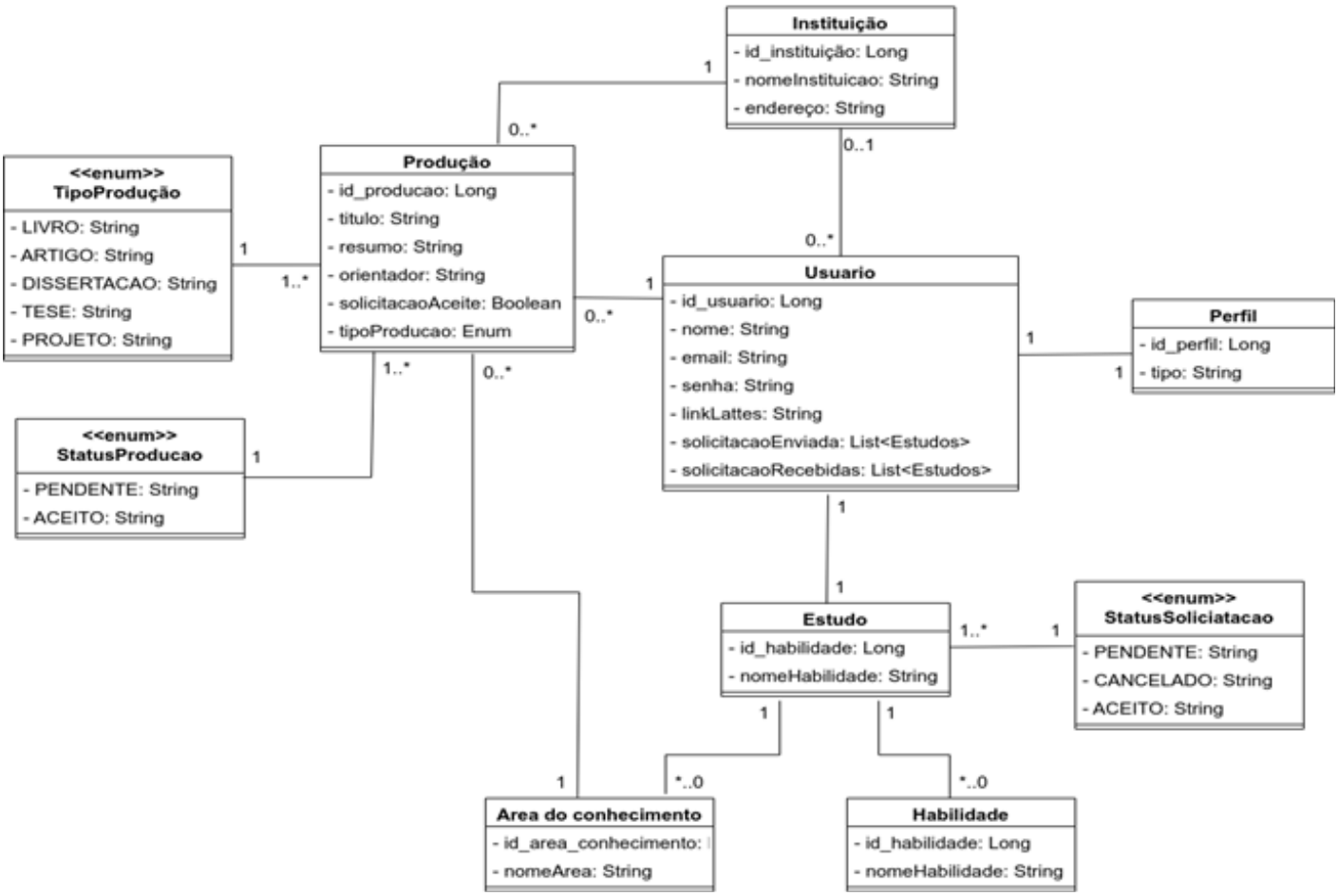



**Figura 3.** Diagrama de classe.

**Fonte**: Autoral, 2022.

A classe "Usuário" contém todos os atributos necessários para formar o perfil do usuário, como e-mail, nome, senha etc.

A classe "Instituição" é responsável por conter as informações das instituições de ensino.

A classe "Perfil" é a classe responsável para definir o nível de permissão que o usuário pode possuir, existem dois tipos de perfil no sistema, administrador e usuário comum.

A classe "Produção" possui informações referente a publicação do trabalho acadêmico no qual podem ser divididos em cinco categorias: LIVRO, ARTIGO, TESE, PROJETO e ARTIGO essas categorias estão no enum "TipoProdução", a produção também possui o enum "StatusProducao" que contém PENDENTE e ACEITO, status necessário para separar as publicações que foram aceitas e já estão aparecendo na pesquisa e as que estão em

análise pelo administrador do sistema.

A classe "AreaConhecimento", são definidas as áreas dos conhecimentos gerais exemplo Ciências Exatas, Ciências Biológicas, e etc. A classe "Habilidade" contém os atributos de habilidades específicas, por exemplo Spring Boot, Vue.js, Java etc.

A classe "Estudo" é responsável por conter a área do conhecimento e habilidade do usuário, é por ela que é feita a função de buscar por tutores, "Estudo" está associada a um enum "StatusSolicitacão", que contém: PENDENTE, CANCELADO e ACEITO, esses status são responsáveis por classificar as notificações de solicitação de tutorias.

Com o diagrama de classe conseguimos formar a estrutura do projeto, montando suas relações e atributos, utilizamos o diagrama de classe durante o desenvolvimento para nos ajudar a compreender os requisitos do sistema.

## 3.4 Definição das camadas

No desenvolvimento de uma API utilizando o Spring, é uma boa prática organizar o projeto separando as classes, colocando as em pacotes que representam camadas que se interconectam, as classes são separadas em pacotes chamados *Controller, Service, Model* e *Repository*.

A figura 4, mostra as camadas em um projeto de arquitetura API REST, onde o cliente envia uma requisição que é interceptada pelo controller e enviada para o service onde é feita a regra de negócio do sistema e por fim chegando ao repository que fica com a responsabilidade de fazer a comunicação com o banco de dados.

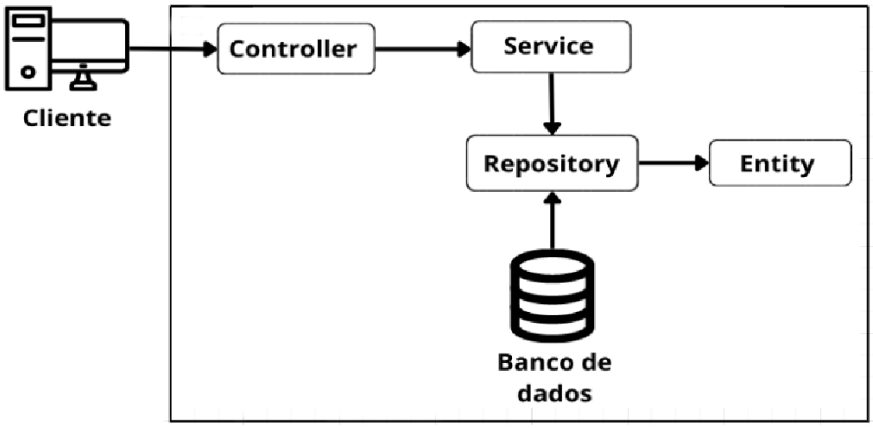



**Figura 4**. Estrutura da API REST.

**Fonte**: Autoral, 2022.

Essa separação permite a construção de um projeto com eficiência e eficácia, também facilita futuras novas implementações e ajuda na manutenção do código, já que é um padrão amplamente conhecido por desenvolvedores de API REST.

### 3.4.1 Camada *controller*

*Controller* é a primeira camada do servidor, é por ela que todas as requisições são recebidas, é nela que são implementados os *endpoints*, essa camada é a porta de entrada para o consumidor da API.

A figura 5, mostra um exemplo de classe controller nela se gerencia as ações dis-

poníveis no projeto. Em um projeto *Spring* faz se uso de anotações, que são funcionalidades introduzidas pelo Spring para o aumento de produtividade no desenvolvimento. A anotação *@RestController* é aplicada a uma classe para marcá-la como um manipulador de solicitação, ela receberá requisições externas através de métodos HTTP (GET, POST, DELETE, PUT etc.). Já a anotação *@RequestMapping* é usada para mapear solicitações da Web para métodos do *Spring Controller*. Cada método possui sua URI. A figura 5 mostra dois exemplos, o método de salvar com a URI "/pesquisadores/criar" e listar usuários "/pesquisadores/listar".

```
@RestController
@RequestMapping("/usuarios")
public class UsuarioController {


    @Autowired
    private UsuarioService usuarioService;

    @PostMapping("/criar")
    @Transactional
    public ResponseEntity<Usuario> cadastrar(@RequestBody @Valid UsuarioCadastro usuarioDto){
        Usuario usuarioSalvo = usuarioService.salvar(usuarioDto.converterCadastro());

        return new ResponseEntity<>(usuarioSalvo, HttpStatus.CREATED);
    }

    @GetMapping("/listar")
    public Page<UsuarioListagemDto> listar(
            @PageableDefault(page = 0, size = 15) Pageable paginacao){

        Page<Usuario> pesquisadores = usuarioService.listaUsuario(paginacao);

        return UsuarioListagemDto.converter(pesquisadores);
    }
```

**Figura 5**. Camada *controller* - *endpoints*.

**Fonte:** Autoral, 2022.

Cada entidade do projeto tem sua classe controller correspondente, e todas elas são diferenciadas pelo seu URI.

## 3.4.2 Camada de *Entity*

A camada de *entity* é responsável pela definição das características de uma classe, é nela que se define os atributos e os relacionamentos entre as entidades do projeto.

A figura 6 mostra como é feita a definição das características de uma aplicação. Todos os relacionamentos de uma tabela são representados como entidades por meio das anotações. A anotação *@Entity* é usada para indicar que a classe representa uma entidade. Além disso, também é colocado *@Id* e *@GeneratedValue* no atributo id_usuário pois se trata de uma chave primária que vai ser gerada automaticamente pelo banco de dados.

```java
@Entity
@JsonIdentityInfo(generator=ObjectIdGenerators.PropertyGenerator.class, property="id_usuario")
public class Usuario implements UserDetails{

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(generator="increment")
    @GenericGenerator(name="increment", strategy="increment")
    private Long id_usuario;

    @NotBlank
    private String nome;
    @NotBlank
    @Email
    private String email;
    @NotBlank
    private String senha;''

    private String linkLattes;

    private String nomeImagem;


    @ManyToMany(fetch = FetchType.EAGER)
    private List<Perfil> perfis = new ArrayList<>();
```

**Figura 6.** Entidade Usuário e suas relações.

**Fonte:** Autoral, 2022.

Cada tabela do banco de dados relacional terá sua entidade, com seus atributos e relacionamentos definidos na classe que a representa.

### 3.4.3 Camada de *Service*

A camada de *Service* é utilizada para colocar as regras de negócio do sistema, é uma ponte entre a camada *controller* e a *repository*, pois não é uma boa prática os *endpoints* terem acesso direto ao repositório, por isso essa classe se faz necessária.

A figura 7 mostra a classe de serviço de usuário nela podemos ver que o repositório de usuário é instanciado nessa classe, a anotação *@Service* indica que essa é uma classe de serviço, a também implementação de dois métodos salvar, listarUsuario. A anotação *@Transactional* demarca transações no banco de dados, ela garante com que a execução ocorra dentro do contexto transacional, e o *rollback* será feito caso haja algum erro durante a transação, por boa prática a utilizamos em operações que fazem transações com o banco de dados exemplo: salvar, alterar e excluir.

```java
@Service
public class UsuarioService {

    private final UsuarioRepository usuarioRepository;

    public UsuarioService(UsuarioRepository usuarioRepository) {
        this.usuarioRepository = usuarioRepository;
    }

    @Transactional
    public Usuario salvar(Usuario usuario) {

        boolean emailEmUso = usuarioRepository.findByEmail(usuario.getEmail())
                .stream()
                .anyMatch(usuarioExistente -> !usuarioExistente.equals(usuario));
        if(emailEmUso) {

            throw new NegocioException("Já existe um cliente cadastrado com esse E-mail");
        }

        return usuarioRepository.save(usuario);
    }


    public List<Usuario> listaUsuario(){
        List<Usuario> pesquisadores = usuarioRepository.findAll();
        return pesquisadores;
    }
```

**Figura 7**. Camada de *service*.

**Fonte**: Autoral.

A camada de *service* deve conter todas as classes que contenham as regras de negócio da aplicação, em um projeto Spring todas as entidades devem ter sua própria classe de serviço.

### 3.4.4 Camada de *Repository*

A camada de *repository* é a responsável por fazer a persistência das classes implementadas no projeto, para que essa persistência seja feita cada entidade deve ter sua interface de acesso ao banco de dados.

O spring utiliza o módulo Spring *Data* que implementa o JPA (Java Persistence API), este módulo conta com vários métodos prontos abstraídos de outra interface. A figura 8 mostra uma uma interface que se estende a classe "*JpaRepository*" e passados dois parâmetros, o primeiro e a classe a ser persistida o segundo é o tipo de sua chave primária.

```java
@Repository
public interface UsuarioRepository extends JpaRepository<Usuario, Long>


    Optional<Usuario> findByEmail(String email);

    List<Usuario> findByNomeContains(String nome);


}
```

**Figura 8.** Camada repository.

**Fonte:** Autoral, 2022.

O uso do Spring *Data* evita o trabalho de criar métodos para operações CRUD em projetos que lidam com bancos de dados, facilitando e agilizando o desenvolvimento da API.

## 3.5 Gerenciamento das dependências do projeto

O gerenciamento do projeto foi feito utilizando o Maven, trata-se de uma ferramenta de código aberto para gestão de dependências, com ele é possível automatizar os processos de obtenção de dependências e de compilação de projetos java. Em um projeto Spring é através do arquivo pom.xml, é o responsável pela configuração do projeto, nele é onde fica listado as as bibliotecas que serão utilizadas pelo projeto. A figura 9, mostra o arquivo pom.xml e nele onde fica as dependências do projeto que então no formato da linguagem de marcação XML.

```
<dependencies>
    <dependency>
        <groupId>org.modelmapper</groupId>
        <artifactId>modelmapper</artifactId>
        <version>3.0.0</version>
    </dependency>
    <dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt</artifactId>
        <version>0.9.1</version>
    </dependency>
```

**Figura** 9. Dependências gerenciadas pelo Maven.

**Fonte:** Autoral, 2022.

A utilização do Maven torna o projeto mais fácil para os desenvolvedores Java desenvolverem relatórios, criarem verificações e testar a configuração de automação. O Maven cuida de baixar automaticamente as bibliotecas e dependências do projeto e as configura dentro do *Build Path/Classpath* da aplicação.

## 3.6 *Endpoints*

*Endpoint* é um recurso usado que permite o consumo da API por clientes externos. Em um serviço web services REST, a exposição de um *endpoint* é feita através de URIs. Uma URI (Universal Resource Identifier), é uma cadeia de caracteres compacta usada para identificar ou denominar um recurso na Internet.

Na tabela 1 abaixo, estão listados alguns exemplos de URIs disponíveis no sistema para serem consumidas pelo cliente. Existem métodos do tipo POST e PUT que vão consumir objetos do tipo JSON, e GET que receberá objetos do tipo JSON como resposta.

| URI | Verbo | Descrição da requisição |
|---|---|---|
| https://localhost/usuarios/criar | POST | Cadastra um novo usuário |
| https://localhost/usuarios/listar | GET | Lista todos os usuários |
| https://localhost/estudos/alter-status/1 | PUT | Editar o status do estudo |
| https://localhost/especialidades/ removerHabilidade/1 | DELETE | Excluir a habilidade do usuário |

**Tabela 1**. Exemplos de requisições HTTP e suas finalidades.

**Fonte**: Autoral, 2022.

O *endpoint* é nada mais do que uma URI mapeada que expõe um serviço da API, cada URI no sistema representa um recurso, que precisa ser identificado de forma única.

## 3.7 Documentação

Os recursos disponíveis na API que são fornecidas por meio de URI foram documentados utilizando o *Swagger*, o *framework* possui várias ferramentas que auxiliam no consumo e visualização dos dados retornados pelos métodos da API.

A Figura 10, ilustra como o swagger exibe os *endpoints* em uma interface criada por ele, as *interfaces* são geradas automaticamente com base nas anotações que estão no código, por ele é possível identificar o tipo de solicitação e a URI correspondente.

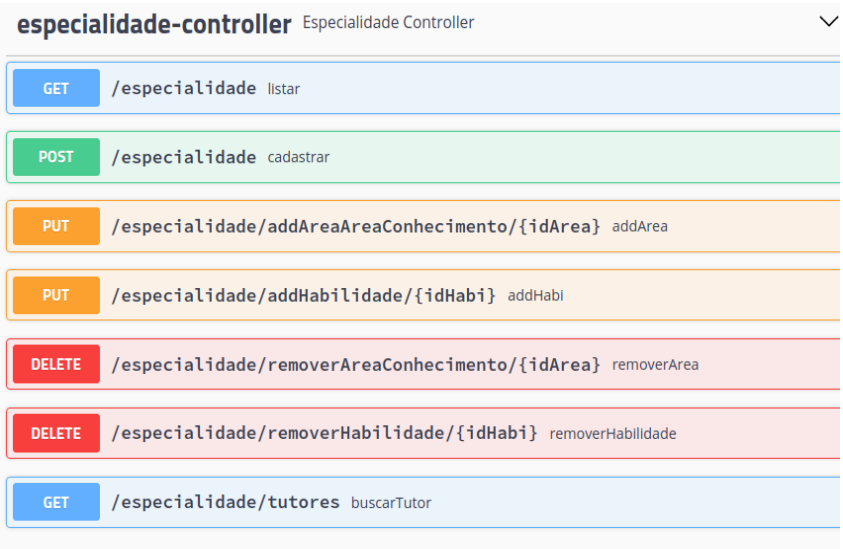



**Figura 10.** Documentação gerada pelo *Swagger*.

**Fonte**: Autoral, 2022.

O uso do *Swagger* permite com que a documentação evolua no mesmo ritmo da aplicação, facilita o entendimento do consumidor da API e exibe toda sua estrutura, mapeando os possíveis parâmetros, respostas e corpo da requisição, a implementação do *swagger* nos ajuda a construir uma documentação robusta e interativa.

# 4. RESULTADOS

Nos resultados, serão apresentadas as principais funcionalidades do sistema de divulgação de trabalhos acadêmicos e de tutoria, como o processo de cadastro de usuário, produção, especialidade etc. Será demonstrado como funciona os *endpoints* de pesquisa dos trabalhos acadêmicos, será mostrado os filtros que foram implementados e como utilizá-los. Também demonstra como é feita a pesquisa por tutores, que também conta com filtros na busca. E por fim, serão mostrados alguns exemplos de um cliente consumindo os serviços da API REST.

## 4.1 Testes

Para a realização dos testes na API REST foi utilizado o programa chamado *Postman*, que é uma ferramenta capaz de realizar requisições HTTP. Esta fase consistirá em testar vários casos nos atores da aplicação, a fim de verificar a resposta recebida, e verificar se os dados estão sendo salvos no banco de dados.

**Cadastro de Usuário:** A figura 11, exibe como é feito um cadastro inicial de um novo usuário é feito, a requisição utilizada é do método POST e tem os campos obrigatórios de nome, e-mail e senha.

```
{
    "nome":"Leo Silva",
    "email": "leosilva@gmail.com",
    "senha": "123"
}
```

**Figura 11.** Json de cadastro de usuário.

**Fonte:** Autoral, 2022.

**Login de usuário:** Na figura 12, mostra um exemplo de login do usuário, a requisição é do método POST, e  que depende do e-mail e da senha do usuário cadastrado.

```
{
    "email": "leosilva@gmail.com",
    "senha": "123"
}
```

**Figura 12**. Json de Login do usuário.

**Fonte**: Autoral, 2022.

**Token JWT:** A figura 13, ilustra um exemplo de saída de um login bem-sucedido, a resposta mostrada é um token do tipo Bearer, a cada login efetuado pelo usuário é gerado um novo, cada token é único e possui uma data de validação que pode ser determinado pelo desenvolvedor. utilizando esse token o usuário poderá ser identificado pela API, o usuário terá que enviá-lo no cabeçalho de autorização dos métodos HTTP.


```
{
    "token": "eyJhbGciOiJIUzI1NiJ9.
        eyJpc3MiOiJBUEkgZG8gcHJvamV0byBHdWFyYW5pIiwic3ViIjoiNyIsImlhdCI6MTY2NjIyNDU5
        MywiZXhwIjoxNjY2MzEwOTkzfQ.nRA9nkJ96iYdx4XgB6S2aX_vHy7UG1c3DP-9iFkw35Y",
    "tipo": "Bearer"
}
```


**Figura 13**. Exemplo de saída de um Login bem sucedido.

**Fonte**: Autoral, 2022.

**Criação de uma nova publicação:** A figura 14, mostra um requerimento do tipo POST para a API de uma nova publicação, a requisição tem que ser enviada junto com o token JWT de validação do usuário, isso permite a identificação do usuário.

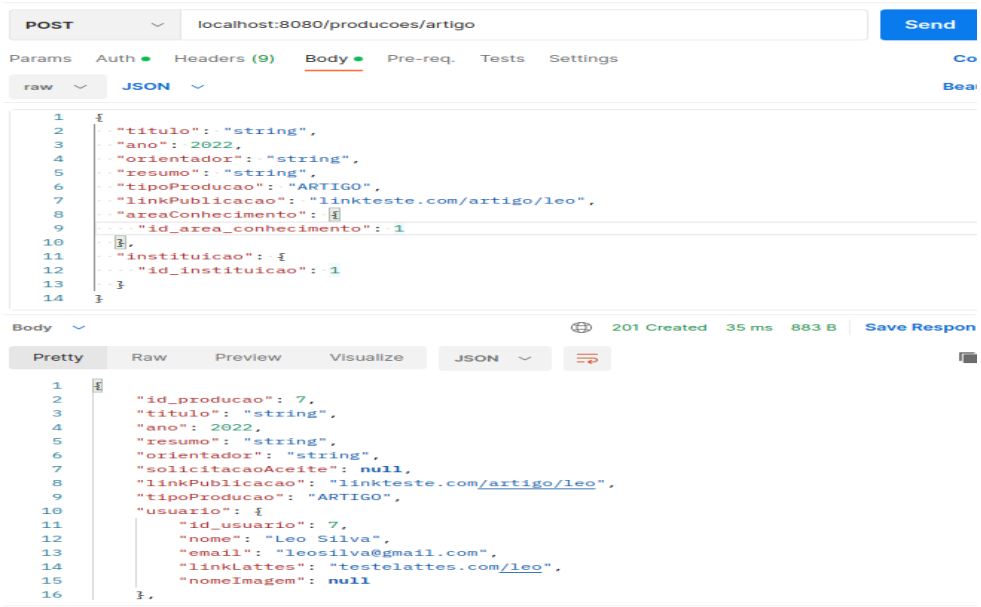



**Figura 14**. Exemplo de Json de cadastro de uma publicação.

**Fonte**: Autoral, 2022.

**Buscando os dados de produções:** A figura 15, ilustra a saída dos dados registrados como artigos, nela vemos como funciona o filtro de busca de produção, dando a possibilidade de filtrar a produção pelo nome do autor, área do conhecimento, título, nome do orientador e instituição.

Todas as *queries* não são obrigatórias e podem ser montadas a critério do usuário, a URI sem filtro "…/artigo/filtro", a primeira *query* é colocada utilizando a interrogação (?), em seguida vem o nome da *query* junto com o igual (=) e por último a variável a ser pesquisada "…/artigo/filtro**?**titulo=Nome do Artigo", as *queries* seguintes é utilizado o "E" comercial no lugar da interrogação "…/artigo/filtro?titulo=Nome do Artigo**&**orientador=Edilson".

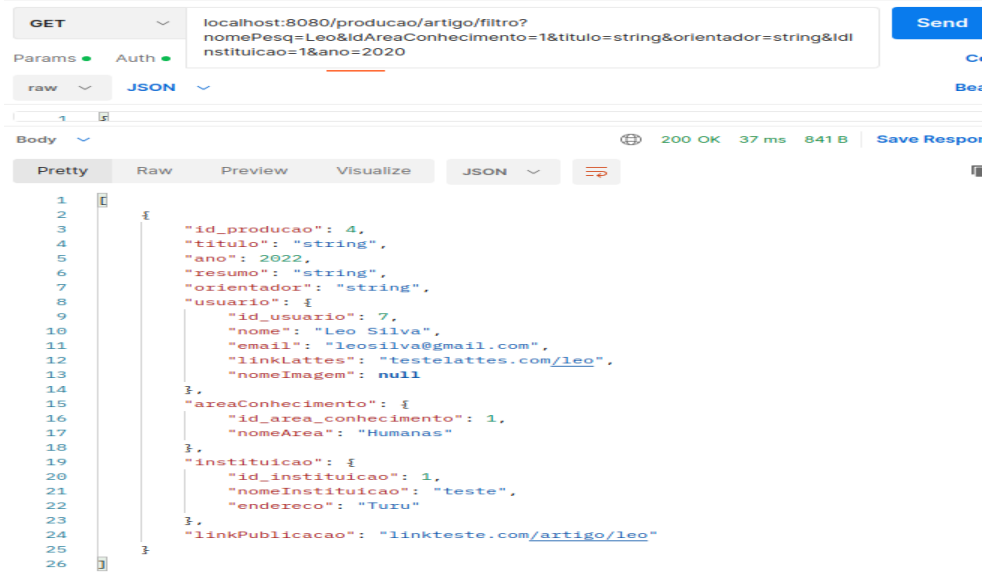



**Figura 15.** Resposta do *endpoint* de pesquisa.

**Fonte**: Autoral, 2022.

**Pesquisa de tutores:** A figura 16, mostra a saída do método GET de especialidade, com esse *endpoint* é possível fazer a busca por tutores, podendo ser filtrado pelo nome do usuário a ser pesquisado,id da área do conhecimento e id de habilidade, todas ela não são

obrigatórias. As *query* são montadas da mesma forma demonstrada no exemplo acima.

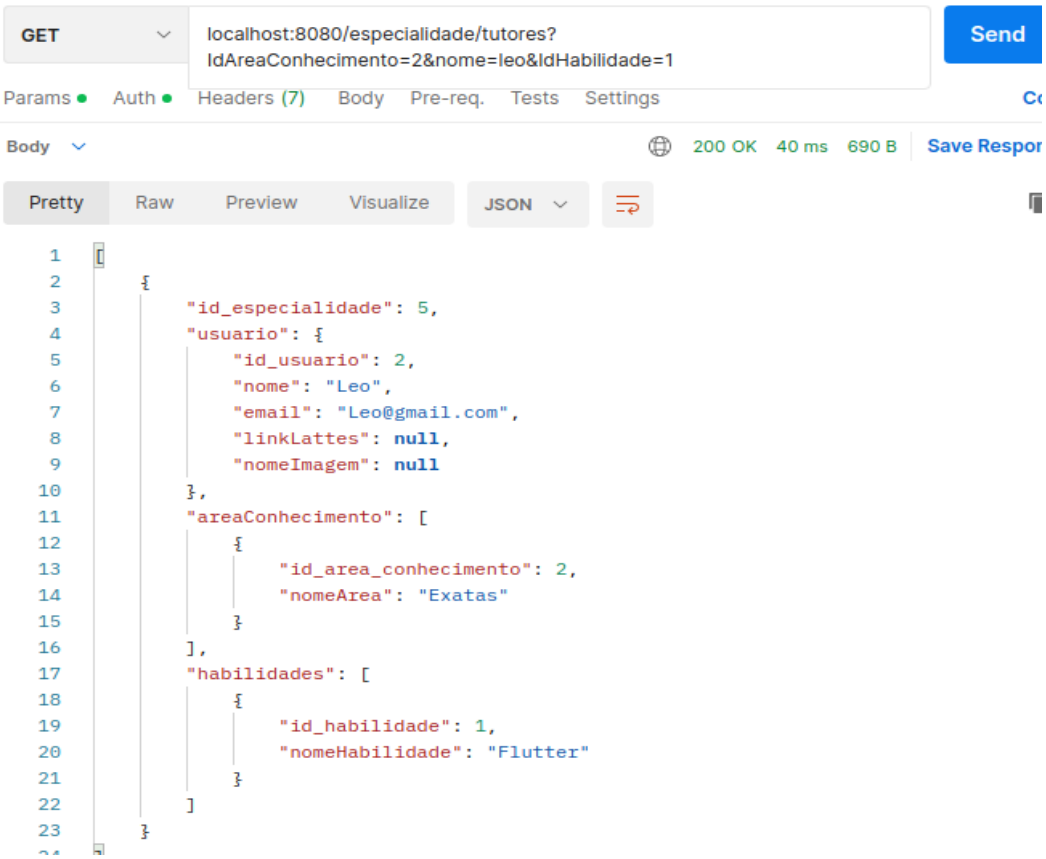



**Figura 16.** *Endpoint* de busca por tutores.

**Fonte:** Autoral, 2022.

**Deletando uma publicação da base de dados:** A figura 17, mostra uma publicação sendo apagada, a produção de id número 4, uma produção só pode ser apagada pelo usuário que a criou ou pelo administrador, que são validados através do token JWT que deve ser enviado no corpo da requisição.

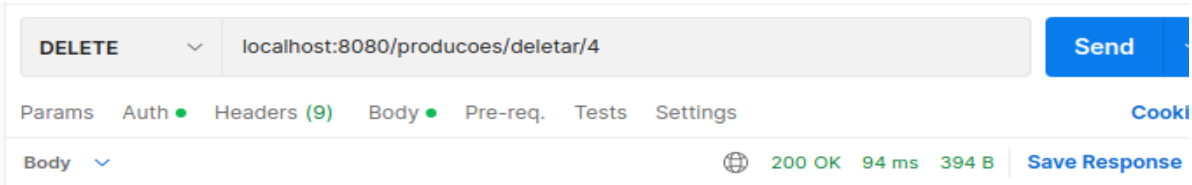



**Figura 17**. Requisição bem sucedida de delete de uma publicação.

**Fonte**: Autoral, 2022.

**Adicionando área do conhecimento:** A figura 18, demonstra como é feita uma requisição PUT, ela adiciona uma nova área do conhecimento, a requisição é feita passando o id da área do conhecimento desejada na URI, no exemplo mostra o id de número 4, que é referente à "Biológicas", para que seja possível a utilização desse *endpoint*, o usuário já deve conter pelo menos uma área do conhecimento ou habilidade cadastrada.

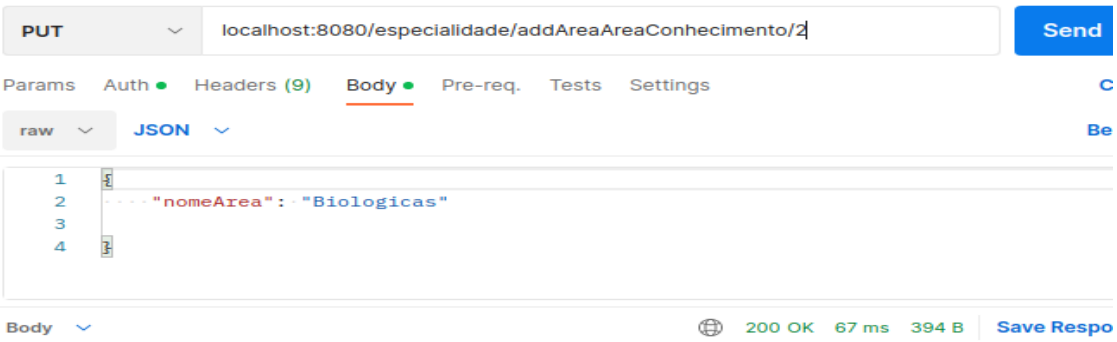



**Figura 18**. Requisição de adição de uma nova área do conhecimento em especialidade.

**Fonte**: Autoral, 2022.

**Solicitação de tutoria:** A figura 19, mostra uma requisição POST para solicitação de tutoria, a realização dessa requisição só é possível com a autenticação do usuário, ou seja, se faz necessário o envio do JWT na requisição.


**Figura 19.** Requisição para criar um nova solicitação de tutoria.

**Fonte**: Autoral, 2022.

A realização dos testes foram realizados durante o desenvolvimento, os testes tinham como objetivo encontrar falhas durante a execução dos *endpoints*, sempre buscando executar várias possibilidades que poderiam causar erros ou falhas durante a execução do projeto, os erros encontrados eram prontamente corrigidos.

## 4.2 Consumo da API REST pelo cliente

Na figura 20 e 21, será ilustrado o consumo da API REST por meio de um site, nos exemplos mostrados, demonstra um site (Cliente) fazendo a pesquisa de um artigo. A figura 21, mostra todos os campos de pesquisa, que serão incrementadas na *query* mostrada no tópico 4.1.5, já a figura 21, mostra o resultado da pesquisa, exibindo os detalhes da pesquisa que foi encontrada na busca.

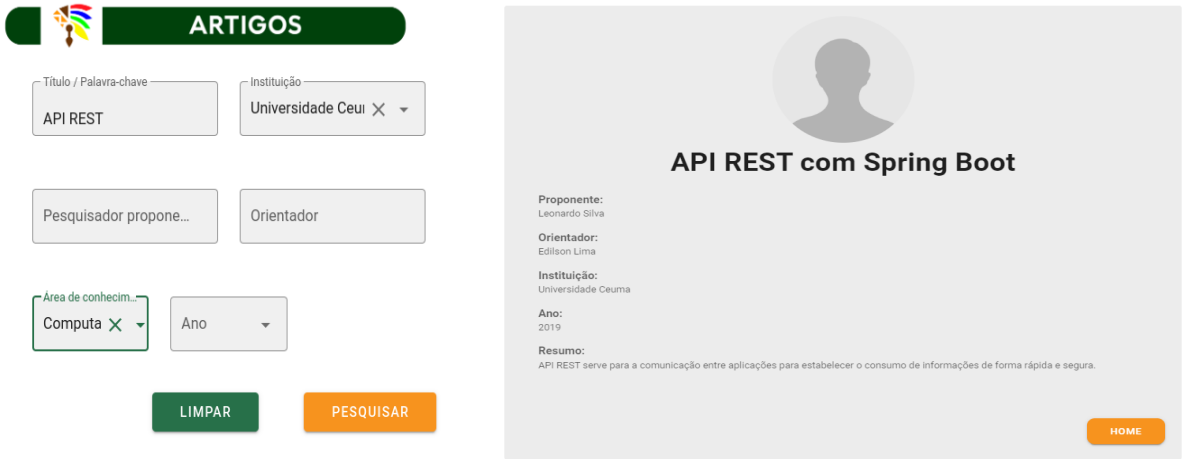



**Figura 20 - 21.** Consumo da API pelo cliente.

**Fonte:** Autoral, 2022.

A API REST desenvolvida oferece uma série de recursos que podem ser utilizadas pelos clientes que vai consumir seus serviços, mas por segurança a API restringe seu uso para apenas sistemas que foram cadastrado de antemão, esse acesso é feito utilizando a URL do site do cliente, caso essa URL constar nas configurações de acesso, esse cliente passará a ter acesso a todos os recursos oferecidos pela API.

## 5. CONCLUSÃO

Neste trabalho foi desenvolvido uma API REST para um sistema de divulgação de trabalhos acadêmicos e de tutoria. O sistema feito nessa pesquisa provê funcionalidades que facilitam a publicação e pesquisa de um trabalho acadêmico no sistema. A funcionalidade de pesquisa foi implementada com vários mecanismos de filtros, visando a fácil busca pelo conteúdo que o usuário procura. O sistema de tutoria permite com que o usuário tire dúvidas com o autor, sobre seu trabalho, ou qualquer assunto relacionado com as habilidades do pesquisador.

Os princípios de API REST empregados na construção do projeto, nos oferecem mecanismos modernos, permitindo com que o sistema funcione em vários tipos de aparelhos ou sistemas operacionais diferentes, diminuindo o custo do desenvolvimento. O Spring demonstrou ser uma ferramenta robusta e poderosa, ela oferece muitos recursos que melhoram a produtividade e como consequência a diminuição do tempo de desenvolvimento, permitindo com que o foco fosse mais na regra de negócio da aplicação.

Como trabalhos futuros fica a oportunidade de criar um sistema de envio de e-mail, para validação do e-mail ou permitir a autenticação de duas etapas. Criar uma funcionalidade de troca de mensagens entre os usuários também seria uma funcionalidade interessante para melhorar a interação entre os usuários.

## Referências

ALCIDES, E. **DESENVOLVIMENTO DE API REST COM SPRING BOOT**. RIOS - Revista Científica do Centro Universitário do Rio São Francisco.unirios. v. 15, n. 29, abril/2021.

Apache . **Welcome to Apache Maven**. 2002–2022. Disponível em: https://maven.apache.org/. Acesso em: 15 out. 2022.

AQUILES, A. **Modelando APIs REST com Swagger**. Alura, 2016. Disponível em: https://www.alura.com.

br/artigos/modelando-apis-rest-com-swagger .Acesso em: 23 out. 2022.

BOSCARINO, A. S. **Por dentro da arquitetura do JWT**. Devmedia, 2019. Disponível em: https://www.devmedia.com.br/por-dentro-da-arquitetura-do-jwt/40281. Acesso em: 22 out. 2022.

CODECADEMY. **What is REST?**. Disponível em: https://www.codecademy.com/article/what-is-rest.Acesso em: 20 out. 2022.

FIELDING, Roy Thomas et al. **Architectural Styles and the Design of Network-based Software Architectures**. 2000. 162 f. Tese (Doutorado) - Curso de Computer Science, Departamento de Computer Science, University Of California, Irvine, 2000.

GABA. I. **What is Maven: Here's What You Need to Know.** Simplilearn, 2022. Disponível em: https://www.simplilearn.com/tutorials/maven-tutorial/what-is-maven. Acesso em: 23 out. 2022.

HANSEN, R. PINTO, S. C. **Construindo ambientes de educação baseada na web através de web services educacionais**. Canoas, 2003. Acesso em: 19 out. 2022.

JWT.IO, **INTRODUCTION to JSON Web Tokens**, 2015. Disponível em: https://jwt.io/introduction. Acesso em: 14 nov. 2022.

LIMA, G. **REST: Conceito e fundamentos**, Alura. 2020. Disponível em: https://www.alura.com.br/artigos/rest-conceito-e-fundamentos. Acesso em: 20 out. 2022.

LIMA, C. **Introdução ao Maven, aprenda como criar e gerenciar projetos Java**, Treinaweb. 2020. Disponível em: https://www.treinaweb.com.br/blog/introducao-ao-maven-aprenda-como-criar-e-gerenciar-projetos-java. Acesso em: 14 nov. 2022.

MAGRI, J. A. **Criando e usando web service**. Augusto Guzzo Revista Acadêmica, São Paulo, n. 11, p. 166-183, junho de 2013. Disponível em: (https://www.fics.edu.br/index.php/augusto_guzzo/article/view/160). Acesso em: 08 nov. 2022.

MONTANHEIRO, L. S. **Utilização de JSON Web Token na Autenticação de Usuários em APIs REST**. 2017. Disponível em: https://www.enacomp.com.br/2017/docs/json_web_token_api_rest.pdf. Acesso em: 14 nov. 2022.

OLIVEIRA, E. M. **Definição do Web Service – Web Service sistema distribuído**. 2010. Disponível em: http://forum-gudo.forumeiros.com/t25-definicao-do-web-service-webservice-sistema-distribuido. Acesso em: 10 out. 2022.

PLANSKY, R. **Definição, restrições e benefícios do modelo de arquitetura REST**. 2014. Disponível em: https://imasters.com.br/desenvolvimento/definicao-restricoes-e-beneficios-modelo-de-arquitetura-rest. Acesso em: 29 out. 2022.

REDHAT. **API REST**. Disponível em: https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api. Redhat, 2020. Acesso em: 19 out. 2022.

RODRIGUES, L. C. R. **Arquitetura REST. In: ARQUITETURA REST**. 2009. UFJF, Juiz de Fora - MG. Disponível em: http://monografias.ice.ufjf.br/tcc-web/exibePdf?id=17. Acesso em: 16 nov. 2022.

SILVA, G. **O que é Rest?**. Coodesh, 2021. Disponível em:https://coodesh.com/blog/dicionario/o-que-e-rest/. Acesso em: 21 out. 2022.

Swagger. **API Documentation.** Disponível em: https://swagger.io/solutions/api-documentation/ .Swagger. 2011. Último acesso: 15 out. 2022.

# 6

## O USO DO VUE.JS NO DESENVOLVIMENTO DE UMA PLATAFORMA WEB PARA A DISPONIBILIZAÇÃO DE TRABALHOS CIENTÍFICOS

*THE USE OF VUE.JS IN THE DEVELOPMENT OF A WEB PLATFORM FOR THE AVAILABILITY OF SCIENTIFIC WORKS*

Ilgner Mendes Bezerra[1]

Edilson Carlos Silva Lima[2]

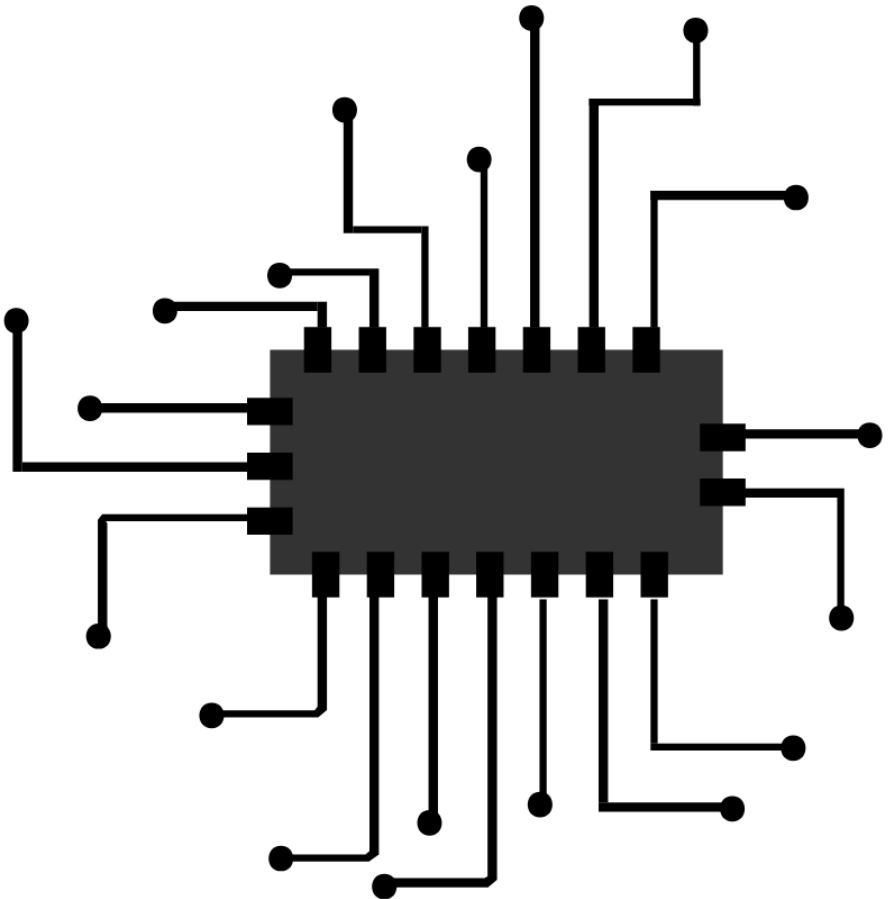
Elda Regina de Sena Caridade[3]

1Engenharia de Computação – Universidade Ceuma (UniCEUMA) 65075-120 – São Luís – MA – Brasil

2Engenharia de Computação – Universidade Ceuma (UniCEUMA) 65075-120 – São Luís – MA – Brasil

3Engenharia de Computação – Universidade Ceuma (UniCEUMA) 65075-120 – São Luís – MA – Brasil

{BEZERRRA, Ilgner Mendes, mendesilgner@gmail.com; LIMA, Edilson Carlos Silva, edilsonlima3@gmail.com; CARIDADE, Elda Regina de Sena, elda18.sena@gmail.com}



**d.o.i.:**

## Resumo

A busca por trabalhos científicos dentro das instituições vem cada vez mais sendo popular, porém ainda existem dificuldades no tocante ao acesso desses trabalhos. O objetivo central deste trabalho é o desenvolvimento de uma aplicação web na qual facilita o acesso e incentivo nos ramos da pesquisa através de conceito do tipo *Single Page Application* (SPA). O *front-end* foi elaborado através do *framework* de *JavaScript* Vue.Js. Já o desenvolvimento *back-end* por meio do *framework* de Java, *Spring Boot* na produção de uma API. Dentro desta ótica, as tecnologias utilizadas nos fornecem uma aplicação web atendendo os problemas visados, além do grande reuso e utilização de componentes no código.


**Palavras-chave:** Framework, Vue.Js, Single Page Application, Spring Boot, API.


## Abstract

The search for scientific works within institutions has been increasingly popular, but there are still difficulties regarding the accessibility of these works. The main objective of this work is the development of a web application that facilitates access and incentives in the fields of research through the concept of the Single Page Application (SPA) type. The front-end was created using the Vue.Js JavaScript framework. Already the back-end development through the Java framework, Spring Boot in the production of an API. optics, as technologies used within an application, which address major problems in the use and utilization of components in the code.


**Keywords:** Framework, Vue.Js, Single Page Application, Spring Boot, API.

## 1. INTRODUÇÃO

Por meio das inúmeras pesquisas acadêmicas, os trabalhos acadêmicos em geral se tornaram cada vez mais acessados, compartilhados e pesquisados. Porém, a realidade por muitas das vezes não é esta, devido a sua grande dificuldade de acesso dos mesmos em diversos cenários. A comunidade científica é formada por uma rede de cientistas de várias disciplinas e entre instituições e interage em termos reais através da ideia ou trabalho que se expressa em artigos científicos. Segundo Haas (1992), o meio científico engloba uma vasta área com conexão entre pessoas interdisciplinares que compreendem que a propagação de ideias estimula o conhecimento. Conforme Yan e Ding (2014), a elaboração e expansão da ciência se deve ao crescente índice de ideias expressadas pela propagação nos meios de comunicação. Onde uma delas é a publicação e divulgação de trabalhos acadêmicos, no qual a ideia apresentada por um autor será desenvolvida por outros cientistas, gerando por sua vez, inovações e novos caminhos a se trilhar.

Logo é visível a necessidade de plataformas que realizam o gerenciamento desses trabalhos científicos. Assim, sustentando uma sequência de vantagens, evitando filas e a facilidade de acesso. Onde qualquer usuário pode acessar, visualizar e pesquisar trabalhos científicos de qualquer lugar apenas com uma conexão à internet. Com base nisso, uma *Single Page Application* (SPA, em português Aplicativo de Página Única) pode ser utilizada, que representa aplicações nas quais suas funcionalidades estão compostas em uma única página. Segundo Oliveira (2017), não há necessidade de recarregar a página inteira quando os usuários interagem com ela. A experiência do usuário final com uma SPA é muito satisfatória, devido ao seu carregamento rápido entre as telas da aplicação, o que nos torna refém do *JavaScript*, porém existem diversos *frameworks* advindos do mesmo, para a construção deste tipo de projeto.

O objetivo deste trabalho é o desenvolvimento de uma SPA através do *framework* Vue.Js para o gerenciamento de trabalhos científicos através das linguagens Java e *JavaScript*. Tal trabalho visa a facilidade de acesso, pesquisa e divulgação de projetos científicos. De acordo com You (2014), em contraste com outros grandes *frameworks*, o Vue.Js foi criado desde o início como um *framework* progressivo que pode ser adotado de forma incremental. É uma estrutura *JavaScript* de código aberto projetada para construir interfaces de usuário. A definição do *framework* utilizado foi por meio da sua fácil escala de aprendizagem e a sua fácil conexão com *API's*, onde utilizaremos o *Spring*. Segundo Souza (2018), ao falarmos de *Spring Boot* temos uma aplicação com rápida produção seguindo as boas normas de desenvolvimento e com características já pré-atribuídas.

## 2. REVISÃO LITERÁRIA

Este tópico irá tratar-se a respeito das áreas de prototipagem, ferramentas de desenvolvimento e conceitos referentes à cada área. É importante salientar a importância de cada uma das etapas a seguir para a construção do projeto. Visto que desde a concepção da prototipação já se fazem definidas interfaces de acordo com a experiência UI/UX[1] do usuário final.

1 São áreas que se completam, a fim de fornecer uma melhor experiência ao usuário. A interação do usuário com o produto juntamente com interfaces mais simples e amigáveis.

## 2.1 Prototipagem

Conforme Palhais (2015), realizar a prototipagem de projetos antes da sua concepção e desenvolvimento nos ajuda a compreender os mais diversos extremos do nosso projeto. Com este auxílio podemos entender de fato a experiência do usuário, visto que a prototipagem simula o produto em uma escala muito próxima do real. Além de permitir a visualização de possíveis erros do projeto.

Visto um projeto voltado a *front-end*, é de suma importância, a relação entre experiência do usuário versus aplicação. Logo foram implementadas vertentes do conceito de *Design Thinking*[2], composto por etapas presentes na figura 1, que por sua vez visam promover o bem-estar na vida das pessoas. Uma forma na qual o *designer* compreende as coisas e age sobre as mesmas, tornando assim novos caminhos.


**Figura 1:** Esquema de etapas do processo de Design Thinking.

**Fonte:** Design Thinking, 2011.

Estas etapas podem não seguir o fluxo linear, como proposto na figura, pois varia com a natureza de cada projeto. Então, cada fase pode ser modelada e implementada da melhor maneira na qual o projeto se adapta.

## 2.2 Figma

Nos últimos anos, a vertente de prototipação antes do desenvolvimento de algo se tornou cada vez mais utilizado, até pelos motivos citados já anteriormente. E, devido a isso, várias vertentes de ferramentas surgiram durante todo esse período. O Figma vem de uma ascensão crescente, e devido a isto se tornou uma das ferramentas de modelagem e prototipagem mais utilizadas. Segundo uma pesquisa do *UX Tools (2021),* o Figma se tornou por si só, a ferramenta mais utilizada por pessoas que trabalham com prototipação, design de experiência do usuário e interface do usuário.

Conforme Villain (2022), podemos criar desde protótipos de websites até aplicações web e mobile. Assim, tornando as etapas mais visíveis, e oferecendo uma visão já futura do resultado final. Deste modo, proporcionando uma infinidade de componentes para a construção desses protótipos. No princípio, apenas designers utilizavam o Figma, porém devido a várias implementações realizadas nas áreas de desenvolvimento como por ex: a iteração entre telas, disponibilização de elementos CSS e HTML e juntamente a um conjunto de *design patterns* sugeridos, os desenvolvedores das mais diversas linguagens adotaram o mesmo para realizar as suas modelagens.

2 Processo iterativo com o foco em compreender os usuários, redefinir problemas e criar soluções inovadoras.

## 2.3 Framework

Segundo Mattsson (1996, 2000), *frameworks* são arquiteturas produzidas com o intuito e conceito de reuso, podendo ser apresentados como classes abstratas e concretas. É justamente por isso que se utiliza de fato, devido ao seu grande reuso, como por exemplo, classes e métodos que possam vir a ser utilizados.

*Frameworks* são classificados de diversos tipos, mas principalmente em dois subgrupos principais: aplicados a orientação a objetos (Fayad et al., 1999a, b; Fayad & Johnson, 2000) e *framework* de componentes (SZYPERSKI, 1997). Os voltados à aplicação, fornecem famílias de aplicações orientadas a objetos, como classes abstratas e/ou interfaces, atribuindo heranças ou implementadas a cada instância de aplicações. Já os de componentes fornecem suporte a componentes que seguem uma determinada forma, possibilitando que instâncias sejam conectadas ao *framework* de componentes.

Os principais benefícios decorrentes da utilização de *frameworks*, segundo Fayad et al. (1999b) e Pinto (2000), advêm da modularidade, reusabilidade, extensibilidade e inversão de controle que os *frameworks* proporcionam.

## 2.4 Vue.Js

De acordo com Incau (2017), a maneira no qual o desenvolvedor *front-end*[3] executava suas tarefas durante as últimas décadas foi modificada. Além de fornecerem todo o leque e apoio aos desenvolvedores, os *frameworks* voltados à reatividade web trouxeram consigo a reutilização de código.

E segundo Sharma (2021), é aqui onde o Vue.Js se destaca, onde ele é um *framework* voltado para o desenvolvimento *front-end,* na linguagem *JavaScript*. Ele se tornou mais utilizado devido a sua fácil integração ao *back-end* não se tornando dependente de sua linguagem. Muitos desenvolvedores se voltaram ao Vue.Js também devido a sua fluidez e reuso de código, tornando o projeto bem mais legível e de fácil entendimento, uma vez que adotamos os *designs patterns*, evitando projetos com inúmeras linhas de código e "sujos", no qual nem o desenvolvedor sabe se situar.

O Vue.Js é composto por tecnologias bases (HTML, CSS e JS), o *framework* basicamente é uma mescla dessas três áreas, porém é de simples compreendimento. Segundo Dornelles (2018), hoje uma das linguagens mais utilizadas na web justamente pela facilidade de interpretação dentre os navegadores é o *JavaScript*, por meio disso os *browsers* conseguem obedecer às definições atribuídas no *script*. Com o uso desta linguagem, podemos realizar toda a manipulação das características do browser em relação à uma página web, nela que ocorre a criação de transições, slides, imagens com vfx (efeitos de vídeo) e tudo que estiver na DOM (*Document Model Object*). E que por sua vez, também é capaz de realizar conexões com *API's*.

Como podemos observar na figura 2, uma vez que o navegador incorpora os dados HTML, CSS e JS e plota isto na página web, obtemos um leque de requisições que são obtidas de acordo com os nossos *endpoints*, referentes à API. Porém com o Vue.Js iremos utilizar uma biblioteca/dependência chamada Axios, que iremos discutir logo em breve.

3 O front-end é a parte do site na qual usuários podem visualizar e interagir, são todas as partes de um aplicativo web que criam a sua aparência.

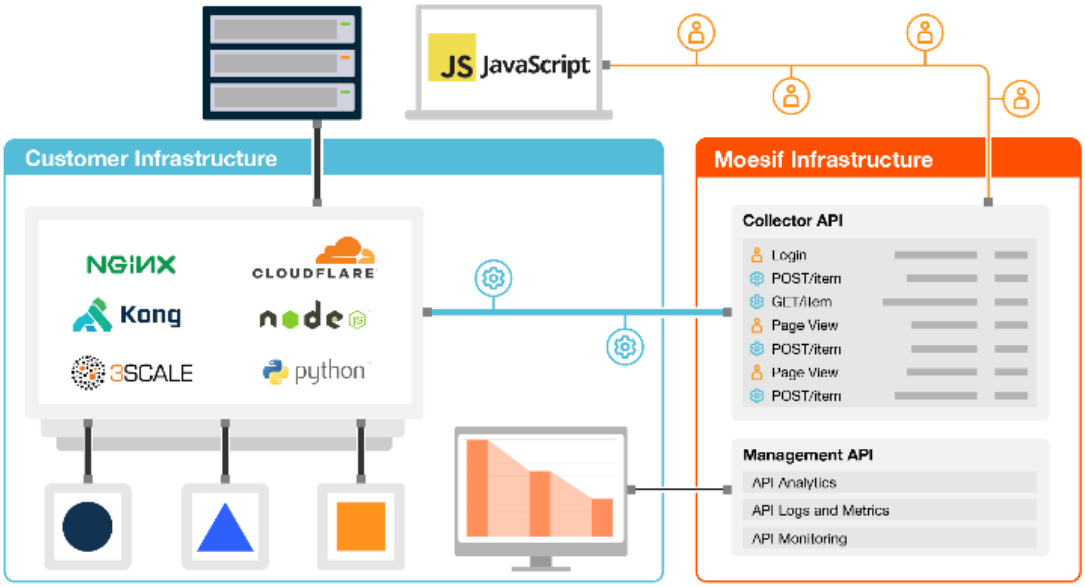



**Figura 2:** Esquema representativo API x navegador

**Fonte:** Moesif, 2022.

O HTML surge para realizar as marcações referentes a cada "texto" exibido, juntamente com o CSS que por sua vez é a apresentação visual do site, o CSS cuida da decoração e estilo referente às camadas do HTML.

Utilizaremos o Vue.Js para o desenvolvimento *front-end* deste projeto, até pela afinidade com os objetivos finais do mesmo, uma vez que iremos trabalhar com SPA e componentes reativos.

## 2.5 Axios

Quando estamos desenvolvendo aplicações web sempre chegamos na situação na qual temos que consumir e exibir dados de uma API. Exclusivamente, a maneira mais importante de realizar isto no Vue.Js é com o Axios. De acordo com Bates (2020), nada mais é que um cliente HTTP baseado em *promises* com suporte para navegador e servidor, utilizaremos o Axios nesse projeto para realizar a integração do *front-end* (Vue.Js) com o *back-end* (Spring Boot) utilizando o padrão HTTP. Na figura 3, temos o esquema visto em uma situação na qual ocorre o consumo de dados provenientes através de requisições da própria API.

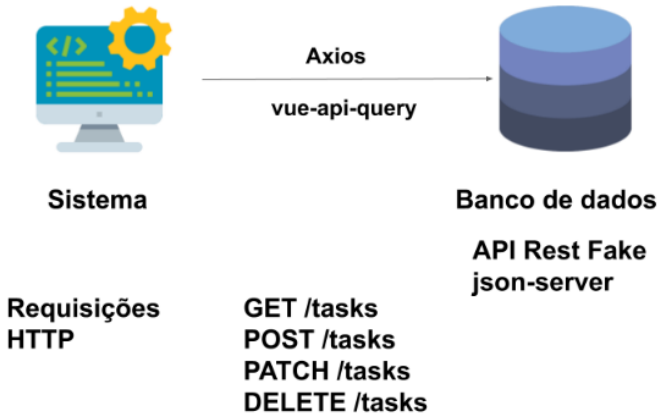



**Figura 3:** Esquema representativo API x navegador

**Fonte:** Package Axios, 2021.

O Axios realiza uma solicitação direto ao banco de dados e retorna um json, que podemos realizar diversas requisições com o próprio Axios, assim podendo enviar, receber e até apagar dados do nosso banco de dados, de acordo com as requisições.

## 2.6 Spring Framework

Por sua vez, chegamos ao *Spring Framework*, que é o responsável pelo *back-end*[4] da nossa aplicação. Mas antes, vamos entender o que seria o Spring, como já explanado na introdução, temos o Spring como um *framework* em Java e também é baseado nos *designs patterns*, que são normas que garantem mais qualidade e segurança à aplicação. De acordo com Weissmann (2014), surgiu para tornar mais ágil a vida do *back-end*, pois antes se tinha configurações muito complexas e imensas para serem definidas, que demoravam até semanas para serem concluídas.

Conforme Afonso (2017), assim surgiu o *Spring* MVC e o seu modelo de acordo com a figura 4, através de camadas que facilitam a configuração inicial para aplicações web que utilizam Java, trazendo diversas melhorias, como a componentização, utilizando apenas componentes necessários para a aplicação. Dentre as principais funcionalidades, destaca-se:

- O Spring MVC
- Suporte para JDBC, JPA
- Injeção de dependências

4 Camada referente ao servidor, é a parte associada a lógica oculta implementada nos aplicativos que os usuários interagem.

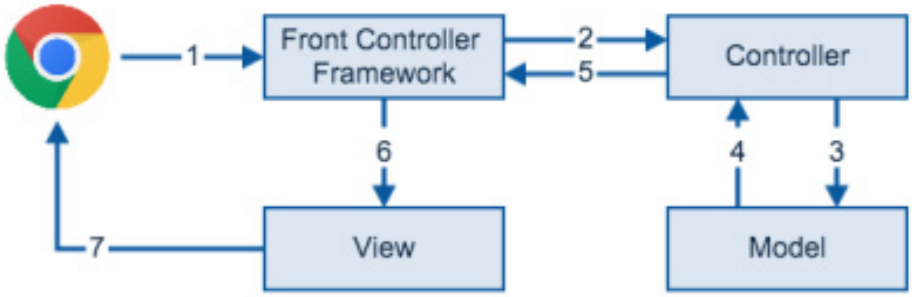



Figura 4: Modelo MVC

Fonte: Algaworks, 2017.

A principal ideia do modelo MVC é realizar a divisão da responsabilidade de cada componente dentro de uma aplicação, facilitando a manutenção do código, iremos observar as etapas para compreender o contexto do Spring MVC.

1. Ao acessar URL o browser enviará a requisição HTTP ao servidor, que é o *controller*.
2. O *controller* irá buscar a classe responsável para esta requisição.
3. O *controller* enviará os dados para o *model*, que processará as regras de negócio.
4. O resultado do *model* é devolvido ao controller.
5. O *controller* devolve a *view* juntamente com os dados necessários para a renderização da página.
6. O *framework* estende-se a *view,* transformando o resultado em HTML.
7. O HTML é retornado ao navegador do usuário final.
8. O *back-end* utiliza o modelo MVC, que nos ajuda em aplicações web robustas, flexíveis e separa as responsabilidades de cada requisição.

## 3. METODOLOGIA

A metodologia abordada no presente artigo é designada quantitativa exploratória de natureza aplicada. A apuração e coleta de dados foi realizada por meio de formulário de pesquisa (online) buscando um público alvo voltado para acadêmicos, entre professores e alunos de diversas instituições de ensino, com questões referente a plataforma desenvolvida, dificuldades dos meios já utilizados e melhorias a serem implementadas. Logo após, foram examinadas e listadas em categorias similares, para assim realizar a construção dos gráficos deste presente trabalho. De tal maneira visando considerar e compreender as dificuldades e necessidades do usuário em busca de trabalhos acadêmicos, aspirando aprimorar e facilitar esta experiência.

## 4. A CRIAÇÃO DO FRONT-END

Sabendo a importância da pesquisa e fomento de iniciativas nas quais fornecem o auxílio à acesso às mesmas, primeiramente fez-se necessário a natureza do problema que deverá ser solucionado. O acesso a esses trabalhos científicos muitas das vezes é

encontrado de difícil acesso, através de solicitações, filas e tempo de espera. Logo, a ideia é tornar isso mais rápido e de fácil acesso por meio de uma plataforma que irá realizar a disponibilização desses trabalhos, onde o usuário apenas realize a busca e rapidamente acesse o conteúdo selecionado. Por isso, escolhemos tecnologias que possuem uma curva de aprendizado rápido, que realizem o reuso de código e componentes visto que a maioria das telas possuem os mesmos componentes e principalmente tecnologias que estão em alta no mercado de trabalho.

## 4.1 Desenvolvimento do Protótipo

Nesta primeira etapa, foi realizado a prototipação e análise do projeto, visando em quais áreas ele iria ser inserido e quais telas seriam prototipadas, como por exemplo: Tela principal da plataforma, tela de pesquisas, resultados e detalhamentos. É, de suma importância, realizar a prototipagem antes em si do desenvolvimento. Isto auxilia na diminuição de problemas e nos fornece uma melhor aproximação do resultado final, todas essas telas utilizadas para criar a relação de protótipo x resultado foram criadas no Figma.

E é importante frisar que, o protótipo não é a versão final do seu projeto, como por exemplo, podemos ver na figura 5 que foi a primeira ideia elaborada no Figma mas que ao longo do tempo foi se tornando mais desenvolvida.


**Figura 5:** Prototipagem Figma

**Fonte:** Autoral, 2022.

É no Figma que ocorre toda a parte voltada para a experiência do usuário, onde vemos a relação de paleta de cores, detalhes visuais da página e até o formato dos campos juntamente com as fontes a serem utilizadas.

## 4.2 Mapa do Site

No próprio Figma foi realizado a análise e criação do mapa do site, compondo todas as etapas/páginas e suas respectivas telas, pois assim podemos verificar e definir quais áreas necessitam de um melhor englobamento e/ou modificação. Como podemos ob-

servar na figura 6, a plataforma é composta pela *Homepage* (página inicial) podendo-se estender as áreas restantes *Login*/Cadastro e Contato. A partir da área de navegação, o usuário pode definir quais tipos de busca ele está procurando, assim chegando nas áreas de busca, efetuando a pesquisa temos a tela de resultados e por fim detalhamentos.

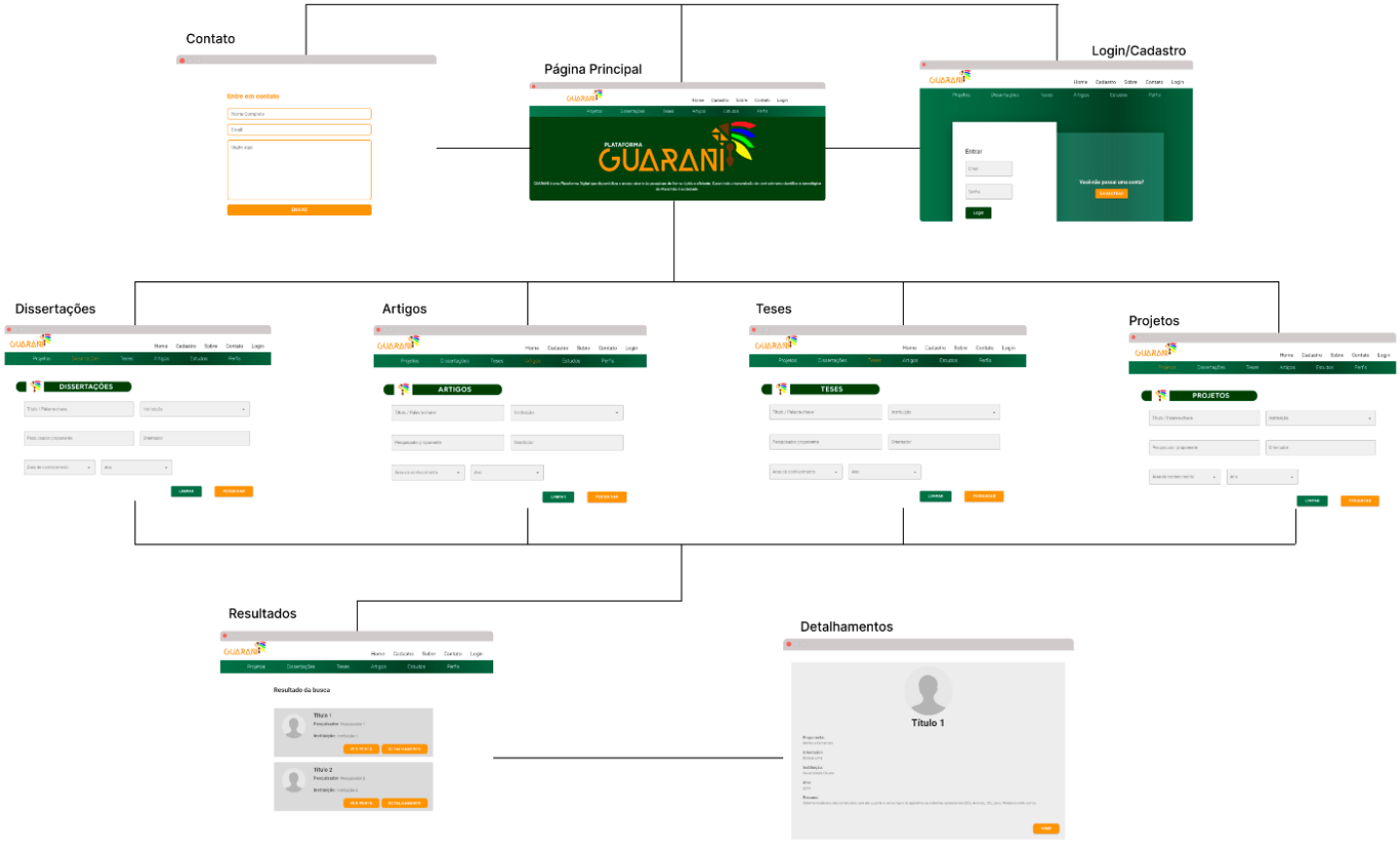



**Figura 6:** Mapa do Site

**Fonte:** Autoral, 2022.

O mapa do site também é importante no SEO da empresa, visto que ajuda os mecanismos de pesquisa a rastrear, indexar e encontrar todo o conteúdo referente a sua página web, eles também informam quais são as páginas do site que são mais importantes ao usuário.

## 4.3 Componentes

Logo após todas estas etapas já percorridas, assim surgiu a dúvida de quais tecnologias (*frameworks)* serão abordadas. É por isso que as etapas anteriores foram de suma importância pois por meio delas se obteve uma visão geral do projeto e observou-se que a grande parte das páginas utilizam o mesmo visual, juntamente com as áreas que possuem os mesmos campos de buscas. Analisando todas estas informações e juntando com os *frameworks* utilizados no mercado, o Vue.Js foi escolhido. Visto que é uma opção que trabalha com conceitos SPA que são sites que reescrevem dinamicamente uma página web com novos dados advindos do servidor *web*. Ou seja, este processo realiza a renderização apenas de uma página e com o conteúdo que o usuário está a solicitar, em vez do método padrão no qual são renderizadas todas as páginas inteiras.

Além disso, é um *framework* leve e trabalha com componentes, estes presentes na figura 7, já que as telas seguem um mesmo padrão. Nota-se que todas as páginas de pes-

quisa utilizam os mesmos segmentos, assim surge a ideia de realizar a síntese de componentes porque assim podemos utilizar o reuso de código, evitando tornar o projeto sujo.

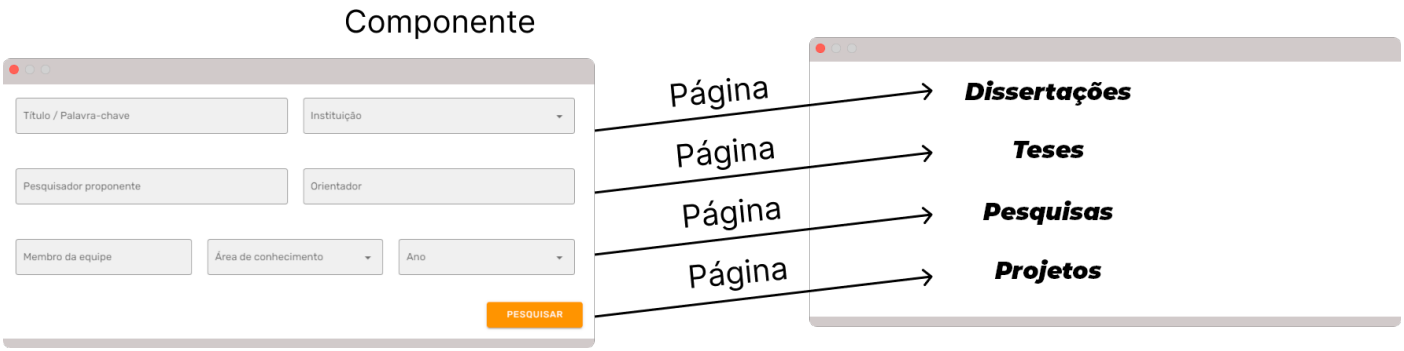



**Figura 7:** Componentes

**Fonte:** Autoral, 2022.

Na figura 8 pode-se perceber o reaproveitamento de componentes em outras páginas. O uso de componentes também ressalta a utilização das boas maneiras, uma vez que o código fica enxuto e reutilizável.

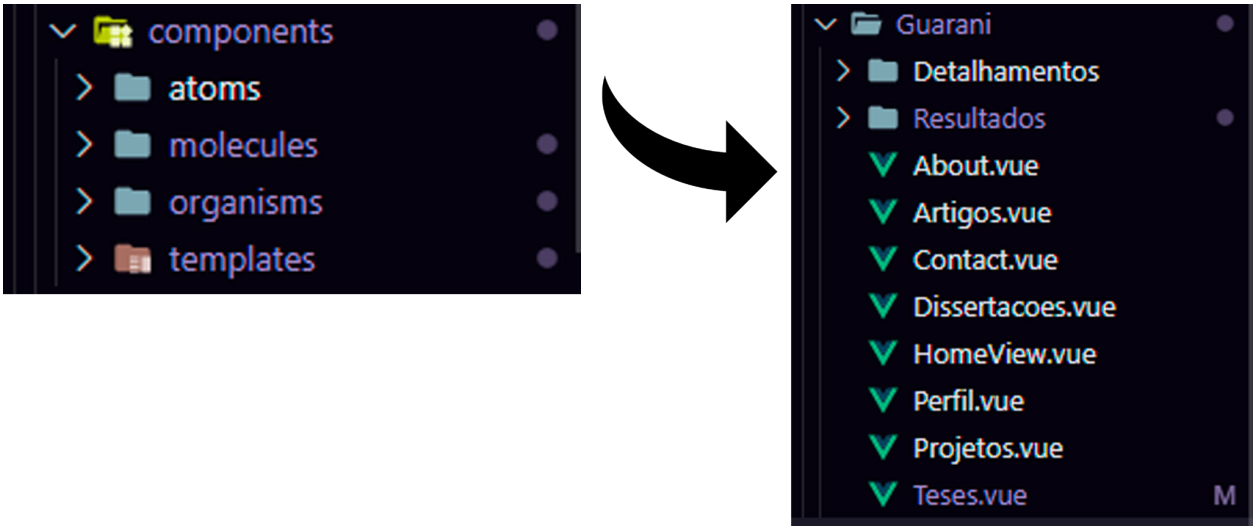



**Figura 8:** Componentes x Páginas

**Fonte:** Autoral, 2022.

Cada componente criado é somente instanciado na página na qual ele é inserido e chamado, na figura acima podemos observar a separação entre componentes e páginas. Componentes formam as páginas e uma vez exportados carregam consigo seus próprios CSS, HTML e SCRIPT.

## 4.4 Consumo da API

Uma vez elaborada a criação de todas essas etapas de estrutura e aparência da plataforma, é chegada a hora da implementação de uma API para o consumo e busca de dados e para isso faz-se necessário o uso de uma dependência do Vue chamada Axios. Antes da elaboração de todos os *endpoints* solicitados pelo *front-end,* é importante a criação de diagramas e arquiteturas para a melhor visualização do projeto como um todo.

Para isto, é possível verificar na figura 9 que cada *endpoint* solicitado pelo *front-end* deve possuir uma rota direta entre tais. O *endpoint* produções abrange todos os trabalhos

científicos disponibilizados pelo site e poderá ser acessado/solicitado através das requisições, como:

- GET /producao/dissertacao
- GET /producao/tese
- POST /producao/livro
- POST /producao/projeto

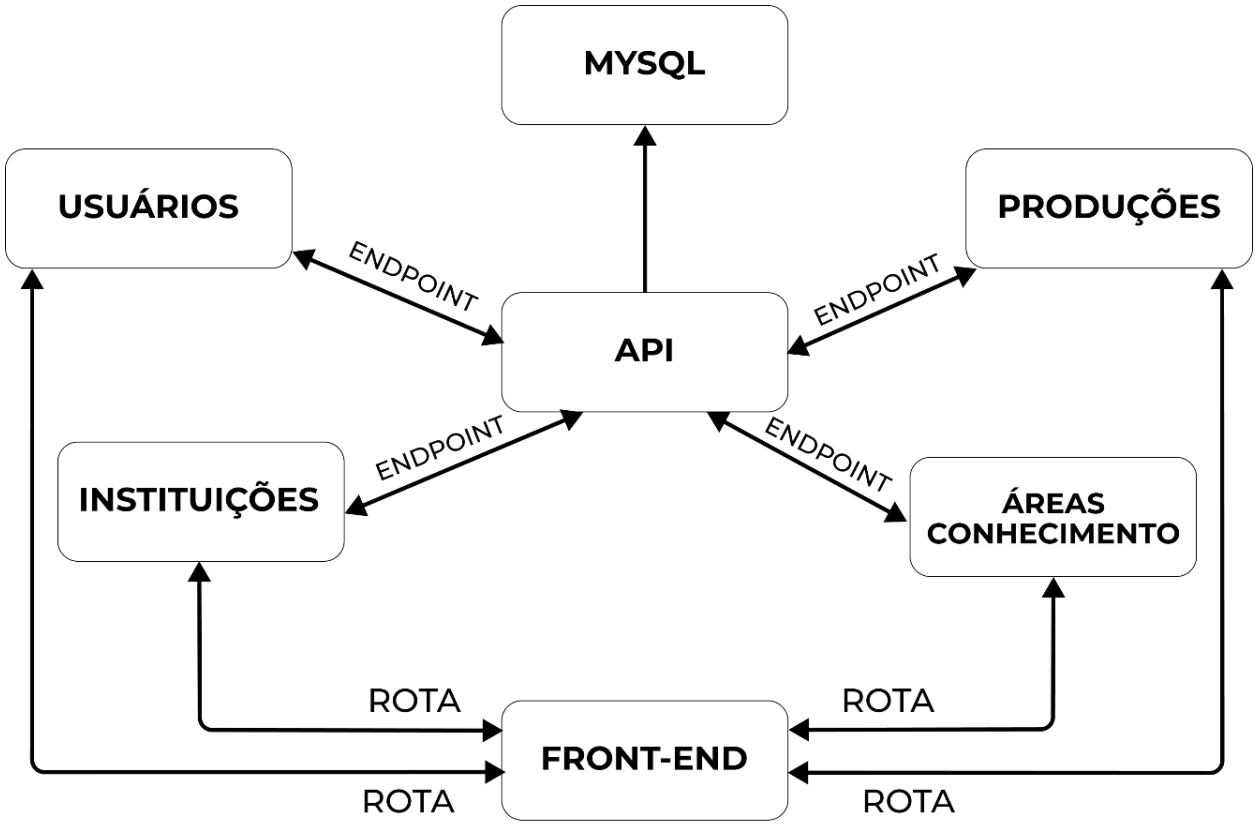



**Figura 9:** Arquitetura Desenvolvida

**Fonte:** Autoral, 2022.

Sabendo isso, é iniciado o consumo da API para a utilização de tais requisições entre o *front-end* e o *back-end.* Um exemplo bem clássico dessa operação é justamente quando o usuário resolve realizar a busca por instituições, onde o campo de selecionar implementa um *autocomplete* que busca no banco informações das instituições cadastradas na plataforma, assim podendo ser exibidas no menu de seleção. Assim, teremos que realizar um *GET* para trazer essas informações do banco de acordo com o nosso *endpoint* e logo após isso exibir esses resultados ao usuário na tela.

Assim que o usuário acessar a tela de pesquisa, todos os campos do modelo de seleção já devem estar com suas respectivas informações, então como vemos na figura 10, existe uma função na qual realiza essa solicitação para a API. É sempre de grande valia nomear os métodos de acordo com as suas ações, assim em um código imenso é possível discernir o que ele está a fazer.

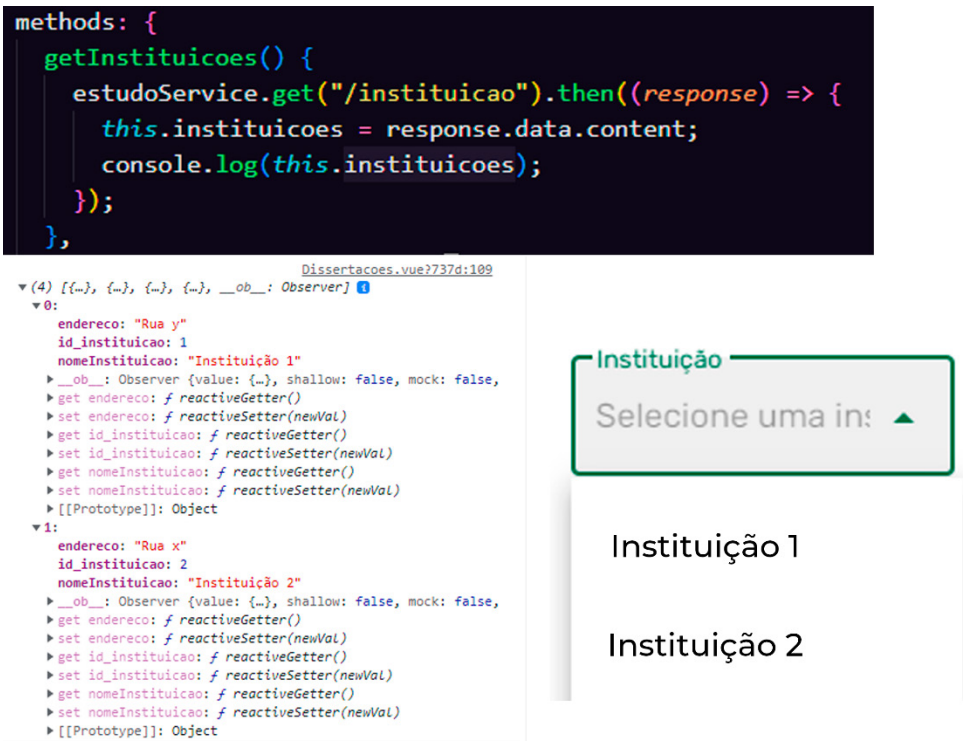



**Figura 10:** GET API

**Fonte:** Autoral, 2022.

É realizada uma solicitação ao *endpoint* /instituicao e a resposta a isto será guardada na variável instituições, porém quando é realizada esta busca não se almeja trazer todos os atributos deste item, como por exemplo: Endereços e outras informações relacionadas à instituição. E por isso deve-se informar somente o conteúdo necessário (*content*) presente na data (*data*) desta requisição, após isso já se obtém todas essas informações desejadas, mas ainda não no campo de seleção e para isso devemos apenas informar a ele o que o mesmo deverá receber, que é a variável na qual guarda as informações da requisição. Uma vez realizado esse procedimento de solicitação e busca diretamente na API, segue o mesmo padrão de requisição em outros campos idênticos.

Após isso, o usuário poderá realizar a pesquisa referente aos seus interesses e deverá trazer os trabalhos científicos especificados pelo mesmo e como já dito anteriormente, como em todas as modalidades de busca de trabalhos científicos existe um mesmo padrão, apenas um componente geral foi definido. Isso ajuda até em relação ao *workflow* do projeto, ao invés de criar outro componente ou outro menu de visualização atribuindo o nome da variável na qual estamos trabalhando e assim podemos reutilizar o mesmo componente com o mesmo resultado de busca. Isso ocorre na maioria das páginas, tanto em componentes de pesquisas quanto nos resultados e detalhamentos trazendo uma harmonia geral para a plataforma como um todo, veremos um pouco a respeito desses componentes citados acima.

No componente abaixo da figura 11, todos os dados são advindos completamente do banco de dados que está sendo consumido por meio da API. Para por exemplo, imprimir essas informações para o usuário devemos acessar os objetos trazidos pelo *GET* e assim definir a sua impressão com as *tags* do próprio HTML.


**Figura 11:** Tela de Resultados

**Fonte:** Autoral, 2022.

O card criado juntamente com as informações da API na figura 12, podem ser acessados por meio da definição de objetos do resultado obtido com as requisições. Por exemplo, o campo título está inserido dentro de um objeto chamado dissertação, devemos acessar primeiro o objeto principal e logo após o atributo solicitado. Realizando assim o mesmo tipo de solicitação para os diferentes campos envolvidos.

```
<v-card color="#D9D9D9" flat class="grid mt-5">
  <div class="container-img">
    <img src="@/assets/hj.jpg" alt="" />
  </div>
  <v-card-title>{{ dissertacao.titulo }}</v-card-title>
  <v-card-subtitle>
    <div class="cargo">
      <h3>Pesquisador:</h3>
      <p>{{ dissertacao.usuario.nome }}</p>
    </div>
    <div class="instituicao">
      <h3>Instituição:</h3>
      <p>{{ dissertacao.instituicao.nomeInstituicao }}</p>
    </div>
  </v-card-subtitle>
  <v-card-actions>
    <v-btn dark color="#F7931E" @click="verPerfilUsuario">Ver perfil</v-btn>
    <v-btn dark color="#F7931E" @click="verDetalhamento">Detalhamento</v-btn>
  </v-card-actions>
</v-card>
```

**Figura 12:** Card Resultados

**Fonte:** Autoral, 2022.

Ao aderir o mesmo princípio, seguiremos consumindo a *API* e trazendo os demais dados referentes aos outros cards, lembrando somente de realizar a troca de objeto principal para os objetos realmente requeridos para esta solicitação.

## 4.5 Redirecionamentos

E antes de encerrarmos esta unidade, é importante ressaltar a implementação das rotas neste projeto, que é realizada pelo Vue Router. Rotas são o redirecionamento para outras áreas dos sites de acordo com as URL's definidas no seu arquivo de rotas. Um exemplo clássico de rota é o clique em algum botão e ele redireciona para uma nova aba de determinado site. Quando o usuário acessa o detalhamento de determinado trabalho científico, o mesmo deverá ser redirecionado para informações referentes àquele trabalho, mas para isso é necessário saber qual id de trabalho está sendo acessado.

Como salientado na figura 13, primeiramente devemos definir o caminho (URL) na qual devemos ser redirecionados, atribuindo um *query* com o nome e o componente que será utilizado para esse redirecionamento. Após isso, foi elaborada uma função para realizar a mudança de página, informando ao *router push* o *endpoint* que solicitamos qualquer id de produção na qual está sendo acessada e finalmente podemos obter o redirecionamento para a página com os detalhes de tal produção.

***ARQUIVO DE ROTA***

```
{
  path: "/dissertacoes/resultados/detalhamentos/:id",
  name: "detalhamentoDissertacoes",
  component: DetalhamentosDissertacoes,
},
```

***FUNÇÃO DE REDIRECIONAMENTO***

```
methods: {
  verDetalhamento() {
    this.$router.push(
      `/dissertacoes/resultados/detalhamentos/${this.dissertacao.id_producao}`
    );
  },
},
```

**Figura 13:** Redirecionamento

**Fonte:** Autoral, 2022.

O funcionamento da plataforma foi esse exemplificado durante este capítulo, onde o usuário pode realizar pesquisas utilizando (ou não) filtros de busca, e acessar os detalhes referentes à cada pesquisa. Utilizando conceitos de consumo de API e principalmente *JavaScript* para a criação de métodos e conexão.

## 5. RESULTADOS E DISCUSSÕES

Com o intuito de entender a realidade dos universitários e também das instituições, foram realizadas pesquisas. Por meio de formulário com perguntas voltadas ao acesso por parte dos acadêmicos a trabalhos científicos disponibilizados pela sua instituição de ensino e sobre a avaliação e experiência do público alvo em relação a plataforma desenvolvida. Desta forma, foram catalogadas 117 respostas e dentre elas foi possível extrair informações que iremos discutir e compreender por meio de gráficos para maior facilidade

e visualização

Em busca de compreender a realidade dentro das instituições, fomos em busca de se saber com base em uma amostra de acadêmicos das mais diversas instituições, o índice de acesso a estes trabalhos seja de maneira virtual (sites) ou presencial (biblioteca). De tal maneira podemos visualizar no gráfico 1 que 53% possuem o acesso, mas de contra-partida 47% relataram que não são privilegiados a isto.

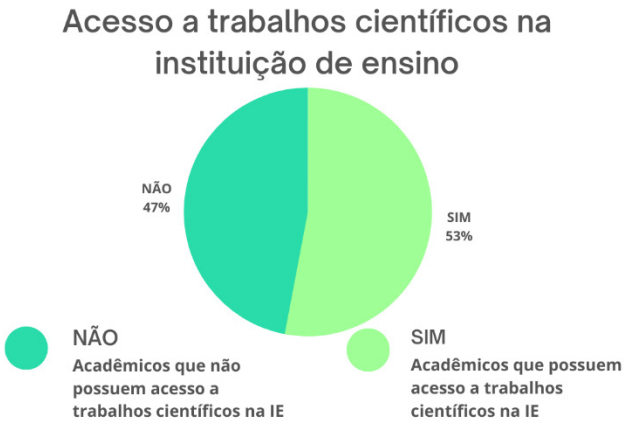



Gráfico 1: Acesso a trabalhos dentro das IE's

Fonte: Autoral, 2022.

O que no século XXI chega a ser até estranho, pois temos todos os adventos tecnológicos e até físicos para o incentivo e disponibilidade de trabalhos científicos, uma vez que isso é um grande passo tanto para a instituição quanto para os alunos residentes na mesma.

Seguindo nesta linha podemos encontrar um dado interessante no gráfico 2, que da mesma amostra citada acima vimos que 53% já possuem acesso a tais trabalhos científicos. Porém a grande maioria (mesmo aqueles que já possuem acesso) usaria algum tipo de ferramenta tecnológica para o acesso a estes trabalhos.

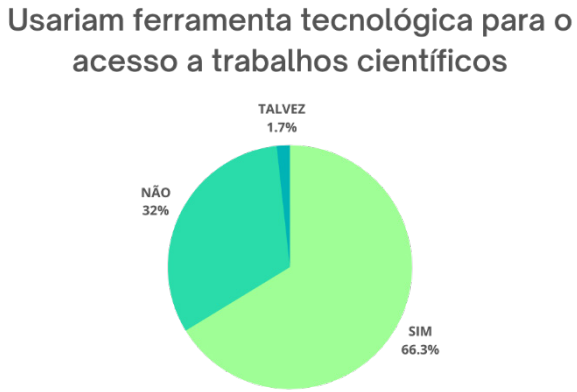



**Gráfico 2:** Uso de ferramenta tecnológica para acesso digital

**Fonte:** Autoral, 2022.

Aqueles que porventura não usariam a ferramenta alegam que os próprios meios já suprem as suas necessidades. Iremos abordar agora quais motivos esses 66,3% entendem que usar a plataforma seria uma boa escolha.

Como citado ao longo deste artigo, os motivos que observamos antes de realizar a

pesquisa eram de facilidade e rapidez, o que após o ato do questionário foi confirmado. Podemos acompanhar no gráfico 3 que, os entrevistados afirmaram que utilizariam a plataforma por motivos de facilidade, rapidez, acesso digital e por ser uma ferramenta tecnológica.

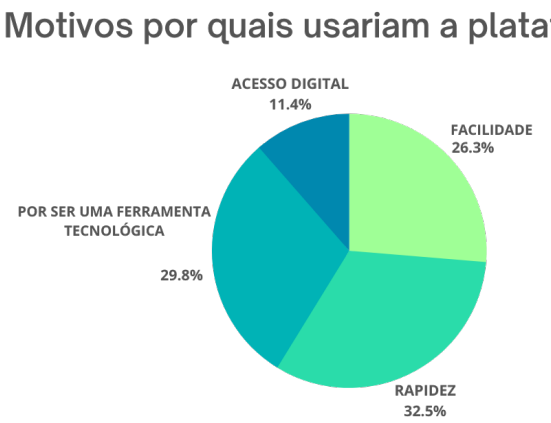



**Gráfico 3:** Motivos de Usabilidade

**Fonte:** Autoral, 2022.

A grande maioria das respostas afirmaram que usariam pelo fato de ser de fácil acesso, oferecendo buscas rápidas, otimizando o tempo de pesquisa (sendo citado a eliminação de filas e requerimentos em maneiras físicas de conseguir estes trabalhos), ofertando uma maior comodidade ao usuário e ampliando o leque de pesquisa assim podendo visualizar diversos trabalhos científicos ao invés de apenas um.

Não poderíamos deixar de solicitar ao usuário aquilo que lhe chamou a atenção e surgiu como um corpo de sugestões para facilitar o uso da plataforma. Analisamos no gráfico 4, que a grande maioria 52,1% avaliaram que a plataforma já se encontra em ótimo estado, entregando aquilo que está como objetivo a ser entregue aos seus usuários. Fora parte, surgiram ideias principalmente em relação a suas funcionalidades (17,1%) e design (19,7%) que iremos abordar com maior atenção na conclusão.

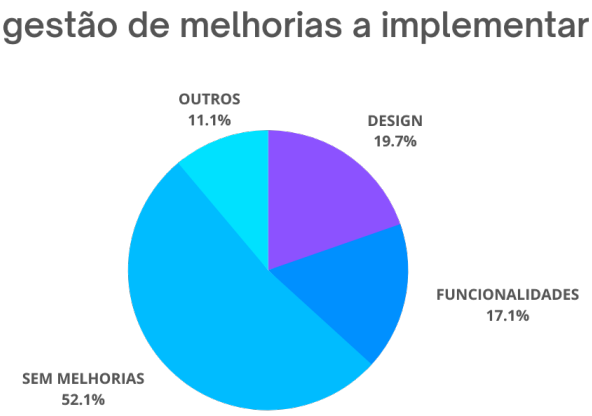



**Gráfico 4:** Melhorias

**Fonte:** Autoral, 2022.

Observamos com todas as descobertas do nosso projeto que a maioria dos acadêmicos realmente se encontra com problemas de acesso. E mesmo aqueles que porventura já se encontram com disponibilidade a trabalhos, também utilizariam a plataforma pois veem nela uma facilidade e rapidez como meio de pesquisa e acesso a trabalhos acadêmicos.

## 6. CONCLUSÃO

Ao longo desse artigo, pode-se compreender como as instituições de ensino tratam os trabalhos acadêmicos e entender as principais dores das pessoas que desejam o acesso a arquivos de pesquisa como os abordados. Após a análise da amostra, juntamente, com outras informações, o projeto da plataforma se torna viável visto que a maioria dos entrevistados afirma que achariam interessante e indispensável a utilização da plataforma como meio tecnológico para a disponibilidade de trabalhos acadêmicos.

Uma vez que os problemas encontrados dentre o público alvo são a dificuldade de acesso, perda de tempo (tempo perdido em filas de bibliotecas e/ou solicitações) e a disponibilidade desses arquivos acadêmicos de maneira digital, acessado em qualquer lugar e a qualquer hora. E nessas lacunas surgem os problemas que seriam sanados justamente pelos motivos que o público alvo usaria a plataforma, que são facilidade, rapidez, acesso digital e por ser uma ferramenta tecnológica permite acesso e divulgação de pesquisas.

O usuário final também nos avaliou e foram recomendadas algumas melhorias onde ao analisar, selecionamos as principais para a implementação em trabalhos futuros onde entram algumas melhorias na plataforma sendo elas separadas em três tópicos, design, funcionalidades e outros (melhorias menos relevantes). As melhorias relacionadas ao design da plataforma foram abordadas características como tornar os botões do site mais chamativos e mínimas mudanças no *front-end* (alinhamentos e posições de itens), melhorias no âmbito das funcionalidades visam uma área específica para o visitante, a adição de mais filtros, mostrar quais professores de determinada instituição estão disponíveis para trabalhar em pesquisas científicas e adicionar uma filtragem de trabalhos mais vistos e avaliados.

## Agradecimentos

Gostaria de fazer alguns agradecimentos e dedicar esse artigo a aqueles que me ajudaram durante toda essa a trajetória:

Em primeiro lugar agradeço a Deus, que permitiu que os meus objetivos durante o curso fossem realizados, pelo conhecimento e apoio permitidos por Ele.

A minha família, minha mãe Vera, meu pai Wagner, minha irmã Ingrid e a minha avó Severa pelo apoio e carinho retribuído durante essa jornada.

A minha namorada Isabela por todo o cuidado e companheirismo comigo durante todo o tempo.

Aos meus amigos da universidade Juliana, Victor, Leonardo, Marcelo, Carlos Eduardo, Julyana e Mariane.

A todos os professores que pude conviver e conhecer.

O meu orientador Edilson Carlos Silva Lima.

Afinal de contas, lamento se por ventura eu esqueci alguém nesta lista de agradecimentos, mas é difícil lembrar de todas as pessoas (amigos e corpo docente) que conheci e realizei conexões, sinta-se incluído aqui.

## Referências


AFONSO, Alexandre. **Produtividade no Desenvolvimento de Aplicações Web com Spring Boot**. São Paulo: Algaworks, 2017. E-book (65p.) Disponível em: <https://cafe.algaworks.com/livro-spring-boot/>. Acesso em: 17 out. 2022.

BATES, Sophie. **What is Axios?**. 2020. Disponível em: <https://codebots.com/docs/what-is-axios>. Acesso em: 15 de set. 2022.

DORNELLES. N. **JavaScript para iniciantes**. 2016. Disponível em: < https://becode.com.br/javascript--para-iniciantes-origens-o-que-e-para-que-serve/>. Acesso em: 27 de set. 2022.

FAYAD, M. E., JOHNSON, R. E. **Domain-specific application frameworks: frameworks experience by industry**. New York: John. Wiley, 2000.

FAYAD, M. E., SCHIMIDT, D. C., JOHNSON, R. E. **Building application frameworks: object-oriented foundations of framework design**. New York: John Wiley, 1999(a).

FAYAD, M. E., SCHIMIDT, D. C., JOHNSON, R. E. **Implementing application frameworks: object-oriented frameworks at work**. New York: John Wiley, 1999(b).

INCAU, C. **Vue.js com suas aplicações incríveis**. Ed. 1. São Paulo: Casa do Código 20 de Abr 2017.

MATTSSON, M. **Evolution and Composition Object-Oriented Frameworks**. [PhD Thesis]. University of Karlskrona/Ronneby, Department of Software Engineering and Computer Science; 2000.

MATTSSON, M. **Object-oriented Frameworks - A survey of methodological issues**. [Licentiate Thesis]. Department of Computer Science, Lund University; 1996.

OLIVEIRA, D. De J. **Uma proposta de arquitetura para Single-Page Applications**. Disponível em: <https://www.cin.ufpe.br/~tg/2017-2/djo-tg.pdf>. Acesso em: 23 de set. de 2022.

P.M. Haas, **Epistemic Communities and International Policy Coordination**, vol. 46, no. 1. Curtin University Library, 1992.

PALHAIS, Catarina Bela Cardoso. **PROTOTIPAGEM: Uma abordagem ao processo de desenvolvimento de um produto**. Orientador: Paulo Parra. 2015. Dissertação (Mestrado) - Curso de Design de Produto, Universidade de Lisboa Faculdade Belas-Artes, Lisboa, 2015. Disponível em: <https://repositorio.ul.pt/bitstream/10451/29163/2/ULFBA_TES_942.pdf>. Acesso em: 19 out. 2022.

PINTO, S. C. C. S. **Composição em Web Frameworks**. [Tese]. Departamento de Informática PUC-Rio, 2000.

SHARMA, Itesh. **WHAT IS VUE JS? THE PROS AND CONS OF VUE.JS**. [S. l.], Out 2021. Disponível em: <https://www.tatvasoft.com/outsourcing/2021/10/what-is-vue-js-and-its-benefits.html>. Acesso em: 11 out. 2022.

SOUZA, Márcio Casale de (org.). **API RESTful com Spring Boot e Java 8.** Manchester. 2018. Disponível em: <https://www.javaavancado.com/ebooks/ebook-api-restful-spring-boot-alex-fernando-egidio.pdf>. Acesso em: 12 out. 2022.

SZYPERSKI, C. **Component Software: Beyond Programming**. Addison-Wesley, 1997.

VILLAIN, Mateus. **Figma: o que é a ferramenta, Design e uso**. 20 out. 2022. Disponível em: <https://www.alura.com.br/artigos/figma>. Acesso em: 30 out. 2022.

WEISSMANN, Henrique Lobo. **Vire o jogo com Spring Framework**. 1. ed. São Paulo: Casa do Código, 2014.

YOU.E. **O que é Vue.js**. [s.d]. Disponível em <https://vuejs.org/v2/guide/> Acesso em: 8 de out. de 2022.

# 7

## APLICANDO PADRÕES DE PROJETO NO DESENVOLVIMENTO DE UMA API REST COM SPRINGBOOT PARA SER CONSUMIDA POR UMA APLICAÇÃO DE CLÍNICAS DE SAÚDE

*APPLYING DESIGN PATTERNS IN THE DEVELOPMENT OF A REST API WITH SPRINGBOOT TO BE CONSUMED BY A HEALTHCARE APPLICATION*

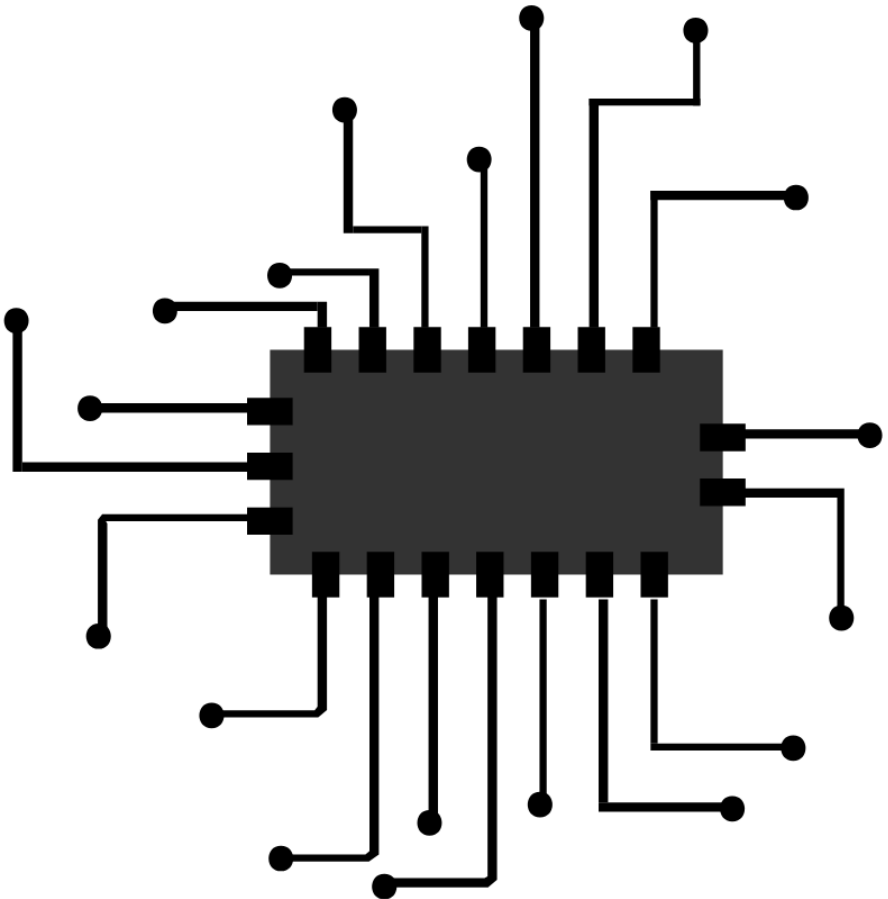
Lucas Yan Costa Matos[1]

Edilson Carlos Silva Lima[2]

1 Engenharia da Computação – Centro Universitário do Maranhão (CEUMA) – São Luís – MA – Brasil

2 Engenharia da Computação – Centro Universitário do Maranhão (CEUMA) – São Luís – MA – Brasil

{MATOS, Lucas Yan Costa, luccas.matos18@gmail.com; LIMA, Edilson Carlos Silva, edilsonlima3@gmail.com}



**d.o.i.:**

## Resumo

Nos dias atuais, vivenciamos uma melhor qualidade de vida com base no avanço da tecnologia. Nesse contexto, podemos inserir os especialistas da área da saúde que estão acompanhando esse progresso de ponta a ponta, e a relação médico/paciente através de agendamento de consultas pode fazer parte dessa evolução. A cada dia, novas tecnologias são criadas e aperfeiçoadas, facilitando o desenvolvimento de softwares, tornando-o mais rápido, seguro e aumentando o seu ciclo de vida. Esse projeto visa atender uma parcela no mercado de especialistas da área da saúde e os pacientes que procuram por profissionais de qualidade. O aplicativo interage com três tipos de usuários: O cliente (paciente), o profissional (médico ou terapeuta) e o usuário administrador, que é responsável por toda parte gerencial, como cadastros de profissionais, departamentos e a disponibilização de horários. O paciente poderá acessar os especialistas e marcar um horário para consultar-se e/ou cancelar uma consulta caso necessário. O projeto será feito no formato API REST, aplicando padrões de projeto e orientação a objetos utilizando o Framework Spring Boot.


**Palavras-chave:** POO. Design Patterns. API REST. Desenvolvimento com Spring Boot.


## Abstract

Nowadays, we experience a better quality of life based on the advancement of technology. In this context, we can include health experts who are following this progress from end to end, and the doctor/patient relationship through appointment scheduling can be part of this evolution. Every day, new technologies are created and improved, facilitating the development of software, making it faster, safer and increasing its life cycle. This project aims to serve a portion of the market of health specialists and patients looking for quality professionals. The application interacts with three types of users: The client (patient), the professional (doctor or therapist) and the administrator user, who is responsible for all managerial aspects, such as registration of professionals, departments and the availability of schedules. The patient will be able to access the specialists and schedule a time to consult and/or cancel an appointment if necessary. The project will be done in the REST API format, applying design patterns and object orientation using the Spring Boot Framework.


**Keywords:** POO. Design Patterns. REST API. Development with Spring Boot.

# 1. INTRODUÇÃO

A cada dia, novas tecnologias são criadas, facilitando ainda mais o desenvolvimento de softwares. Com base nisso, este projeto visa atender o mercado dos especialistas da área da saúde, beneficiando desta forma os pacientes. Ao mesmo tempo que as tecnologias de softwares foram evoluindo, os telefones evoluíram na mesma proporção. Para Steve (2014, p. 141) "Telefones, com o passar dos anos, foram ficando gradualmente mais inteligentes, sugados para dentro das gavetas, cada vez mais parecidos entre si. Mas foi somente com o Grande Salto Adiante que eles finalmente alcançaram a consciência".

O médico e outros profissionais de saúde lidam constantemente com um grande volume e complexidade de conceitos e procedimentos. Este contato diário, através do acesso e manipulação da informação, está relacionado diretamente com a qualidade e a eficácia da assistência ao paciente. Portanto, qualquer solução que ajude esse trabalho passa a ser essencial, e de uso praticamente obrigatório (SABBATINI, 1999, p. 1). Fez-se necessário, por conseguinte, o estudo e implementação de sistemas no intuito de melhorar a comunicação, entendimento e gerenciamento de informações em saúde.

Conforme Costa (2001, p. 29), "a saúde tem um mercado altamente distribuído, que precisa compartilhar informações e isso é feito atualmente, na maioria das vezes, de forma manual, principalmente no Brasil, e na maioria dos casos isso ocorre de forma ineficiente." As clínicas hoje em um número considerável de casos, contam com meios não automatizados para controle de tarefas fundamentais.

Dessa forma, o sistema possibilitará a marcação e gerenciamento de consultas solicitadas centralizado em um único ambiente com todos os detalhes necessários, onde será feito, de forma segura, o armazenamento correto das informações, evitando falhas de comunicação e trazendo uma maior acessibilidade aos clientes e profissionais que a utilizarão.

A metodologia utilizada nessa pesquisa foi o Método Indutivo, onde foi coletado informações baseadas nas experiências de cada usuário, para assim chegarmos a uma conclusão e tomar o pontapé inicial para o desenvolvimento dessa aplicação.

# 2. FUNDAMENTAÇÃO TEÓRICA

Neste tópico será feito uma revisão referencial dos seguintes assuntos: 2.1 Programação Orientada a Objetos, 2.2 MVC, 2.3 Padrões de Projeto, 2.4 Spring Framework, e por fim o item 2.5 API.

## 2.1 Programação Orientada a Objetos

O conceito de orientação a objetos foi introduzido na programação com o objetivo de tornar o desenvolvimento de programas mais simples e natural. Esse conceito é baseado na ideia de que os programas são compostos por objetos que interagem uns com os outros. Cada objeto tem um conjunto de dados (atributos) e de ações (métodos) que podem ser executadas sobre os dados. Uma das principais características é a simplificação do processo de desenvolvimento de um software, tornando o código mais eficiente e legível.

Existem diversas linguagens que é possível utilizar esse paradigma, dentre elas a linguagem de programação Java, que será a linguagem que será utilizada nesse projeto.

Neste paradigma são definidos alguns conceitos importantes a quais serão apresentados a seguir:

- Classes: São coleções de objetos semelhantes, determinados por um grupo de atributos e ações (métodos) de natureza correlata (PUGA; RISSETTI, 2004). Como por exemplo, brigadeiro, trufa e brownie ambos podem ser considerados classes herdadas de doces, porém diferenciam-se em alguns atributos como recheio, tamanho, tipo do chocolate, podendo ser classificado como parte de uma classe superior.
- Objeto: São representações, no domínio da solução, de algum conceito abstrato ou concreto que suporta um incidente (evento ou ocorrência) ou uma interação (transação ou contrato) (PUGA; RISSETTI, 2004).
- Herança: Este é um conceito de reutilizar a classe superclasse ou classe pai herdando seus métodos e atributos. Normalmente, uma classe tem derivações quando, além das características (métodos e atributos) da superclasse, a subclasse possui especificações próprias (DEITEL, 2011).
- Polimorfismo: Este possibilita que um objeto admita o comportamento divergente do que foi estabelecido em sua classe, assim dizendo, a habilidade que o objeto tem de adotar várias formas. Este conceito está relacionado ao de herança (PUGA; RISSETTI, 2004).
- Encapsulamento: Também conhecido como ocultamento de informações, corresponde em esconder detalhes internos de uma classe a um objeto externo. O encapsulamento impede que um programa se torne tão interdependente que uma pequena modificação possa provocar grandes efeitos que se propaguem por todo o sistema (PUGA; RISSETTI, 2004)
- Métodos: São ações que executam tarefas. Eles podem ou não retornar valores e podem ou não receber parâmetros. Regularmente, uma classe possui diversos métodos, que no caso da classe “Atleta” poderiam ser “correr, malhar e descansar” (COAD; YOURDON, 1992).
- Atributos: São as características de um objeto, em outras palavras, a estrutura de dados que vão representar a classe. Os atributos de um objeto representam seu estado, ou seja, o valor de seus atributos em um determinado momento (COAD; YOURDON, 1992).

## 2.2 MVC

O MVC reduz o acoplamento e muda o foco da programação. Lá pelos anos 1970, quando as interfaces de usuários (UIs) eram bem primitivas, um programador decidiu separar os elementos-chave necessários em uma interface gráfica de usuário (GUI) (SANDERS, 2015). Cada uma das partes recebeu uma tarefa específica e cada qual se comunicava com as demais. As partes foram agrupadas em objetos de domínio e objetos de apresentação. Os objetos de domínio eram destinados a modelagem de percepção do mundo real, fornecida ao usuário, e os objetos de apresentação correspondiam ao que era visualizado na tela.

MVC nada mais é que um padrão de arquitetura de software, separando sua aplica-

ção em 3 camadas. A camada de interação do usuário (View), a camada de manipulação dos dados (Model) e a camada de controle (Controller) (ALLAN RAMOS, 2013), a Figura 1 mostra o diagrama de classe do MVC.

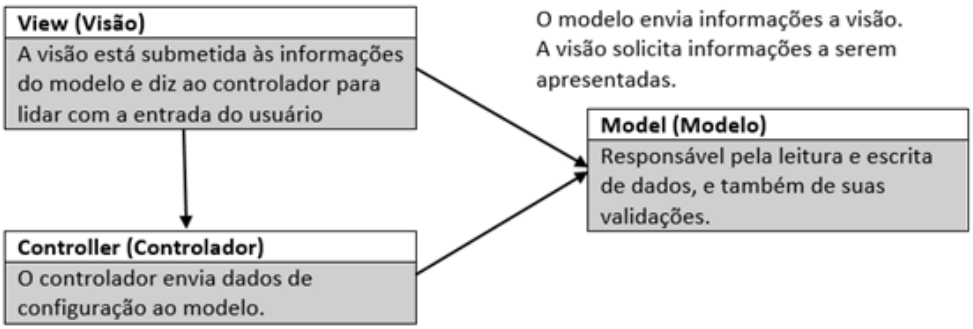



**Figura 1:** Diagrama de Classe do MVC.

**Fonte:** Allan Ramos, 2015.

O MVC parece ter tantas variações quantas são as implementações. Em virtude de sua simplicidade e de sua utilidade, o MVC vem sendo bastante usado, devida e indevidamente. Contudo, permanece a noção de que ele se apresenta como uma estrutura que provê baixo acoplamento entre as partes

## 2.2.1 Model

É a camada responsável pela manipulação dos dados propriamente ditos. As classes que representam as entidades, consultas à banco de dados, métodos de manipulação de dados, entre outros se enquadram nessa camada.

## 2.2.2 View

É a camada com o que os usuários interagem diretamente, ou seja, a interface. No caso de uma aplicação Web, a renderização do HTML final representaria grande parte dessa camada, já em uma aplicação desktop, ela seria representada pela interface como botões, textos, formulários e outros. Basicamente, a camada View é a reposta do processamento efetuado pelas outras camadas do software.

## 2.2.3 Controller

É a camada responsável por efetuar a intermediação entre os **modelos** e a **visualização**, com base na requisição do usuário e/ou outros fatores, decidir qual **Visão** irá usar, fazendo o repasse de dados necessários a ela.

A sua importância reside mais na demonstração de baixo acoplamento do que na funcionalidade direta. Ao separar os diferentes elementos (ou participantes) para realização de uma tarefa, o MVC adicionou uma boa dose de flexibilidade exigida em programas de grande porte. Quanto maior o programa, mais se fazia necessária a flexibilidade de modular no MVC.

## 2.3 Padrões de Projeto

Para Gamma et al (2007), Padrões de projeto, são descrições de objetos e classes comunicantes que precisam ser personalizadas para resolver um problema geral de projeto num contexto particular. Um padrão de projeto nomeia, abstrai e identifica os aspectos-chave de uma estrutura de projeto comum para torná-la útil para a criação de um projeto orientado a objetos reutilizável. O padrão de projeto identifica as classes e instâncias participantes, seus papéis, colaborações e a distribuição de responsabilidades. Cada padrão de projeto focaliza um problema ou tópico particular de projeto orientado a objetos. Ele descreve em que situação pode ser aplicado, se ele pode ser aplicado em função de outras restrições de projeto e as consequências, custos e benefícios de sua utilização.

A classificação dos padrões de projeto seguiu dois critérios, como mostra a Tabela 1. O primeiro critério, chamado finalidade, reflete o que um padrão faz. Os padrões podem ter finalidade de criação, estrutural ou comportamental. Os padrões de criação se preocupam com o processo de criação de objetos. Os padrões estruturais lidam com a composição de classes ou de objetos. Os padrões comportamentais caracterizam as maneiras pelas quais classes ou objetos interagem e distribuem responsabilidades.

| | | Propósito | | |
|---|---|---|---|---|
| | | **De criação** | **Estrutural** | **Comportamental** |
| **Escopo** | **Classe** | Factory Method | Adapter (class) | Interpreter<br>Template Method |
| | **Objeto** | Abstract Factory<br>Builder<br>Prototype<br>Singleton | Adapter (object)<br>Bridge<br>Composite<br>Decorator<br>Façade<br>Flyweight<br>Proxy | Chain of Responsibility<br>Command<br>Iterator<br>Mediator<br>Memento<br>Observer<br>State<br>Strategy<br>Visitor |

**Tabela 1:** O espaço dos padrões de projeto

**Fonte:** Gamma et al (2007).

O segundo critério, chamado escopo, especifica se o padrão se aplica primariamente a classes ou a objetos. Os padrões para classes lidam com os relacionamentos entre classes e suas subclasses. Esses relacionamentos são estabelecidos através do mecanismo de herança, assim eles são estáticos – fixados em tempo de compilação. Os padrões para objetos lidam com relacionamentos entre objetos que podem ser mudados em tempo de execução e são mais dinâmicos.

Os padrões de criação voltados para classes deixam alguma parte da criação de objetos para subclasses, enquanto os padrões de criação voltados para objetos postergam esse processo para outro objeto. Os padrões estruturais voltados para classes utilizam a herança para compor classes, enquanto os padrões estruturais voltados para objetos descrevem maneiras de montar objetos. Os padrões comportamentais voltados para classes usam a herança para descrever algoritmos e fluxo de controle, enquanto os voltados para objetos descrevem como um grupo de objetos coopera para executar uma tarefa que um único objeto não pode executar sozinho.

## 2.4 Framework Spring

O Spring proporciona recursos para aplicativos baseados em Java para qualquer tipo de plataforma. Dispõe de diversas tecnologias, que simplificam o desenvolvimento de código, permitindo que os desenvolvedores se concentrem na construção da regra de negócio. O framework é organizado em módulos que podem ser agrupados em seus principais recursos em Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentação e Teste, conforme apresentado na figura 2.

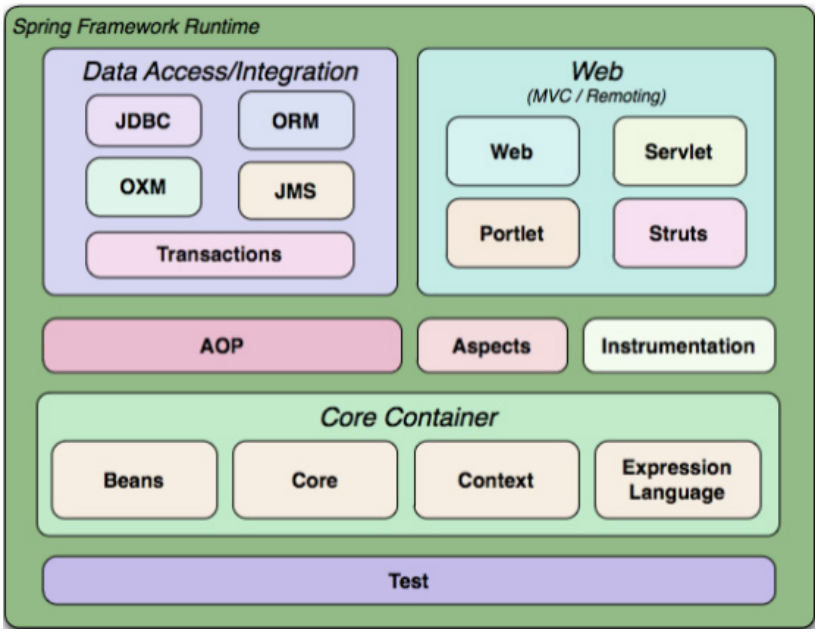



**Figura 2:** Diagrama de Organização Spring

**Fonte:** Pivotal Software (2018).

O objetivo do projeto apresentado é a construção de uma aplicação Web utilizando o módulo Web MVC na qual é estabelecido métodos que recebem requisições HTTP e retornam respostas em JSON através de API RESTful. Fazendo uso, também do Spring Data para fazer a persistência dos dados no MySQL, na qual oferece suporte à Java Database Connectivity (JDBC) e Object Relational Mapping (ORM) com Hibernate e Java Persistence API (JPA).

Outra etapa bastante importante do projeto na qual o spring irá auxiliar será no desenvolvimento de testes unitários para a verificação das funcionalidades de acordo com os requisitos especificados com o Spring Test. O framework permite a utilização da biblioteca JUnit e objetos mocks, que são implementações de abstrações de contexto e ambiente de execução, inclusive com a injeção de dependência.

## 2.5 API

O conceito de API nada mais é do que uma forma de comunicação entre sistemas. Elas permitem a integração entre dois sistemas, em que um deles fornece informações e serviços que podem ser utilizados pelo outro, sem a necessidade de o sistema que consome a API conhecer detalhes de implementação do software (PEREIRA, 2019, n.p.).

Na grande maioria das vezes uma API se comunica com diversos outros códigos interligando diversas funções em um aplicativo. Um exemplo de uma API muito utilizada é a 26 API dos sistemas operacionais que possuem diversos métodos e se comunicam com

diversos processos do sistema operacional.

Já no contexto de desenvolvimento web, uma API é um conjunto definido de mensagens de requisição e resposta HTTP, geralmente expressado nos formatos XML ou JSON. Ainda que esse termo seja um sinônimo para Web Service, a chamada Web 2.0 está aos poucos depreciando o modelo de serviços SOAP para a técnica REST (MEDEIROS, 2012).

De modo prático, os recursos da API são compartilhados através de Web Services e utiliza XML ou JSON como formato de comunicação, ou seja, quando um cliente faz uma requisição, é retornado uma resposta no formato XML ou JSON, quando a API precisa de alguma informação do cliente, o formato enviado também precisa ser o mesmo. Conforme a figura 3, podemos ver o desacoplamento da linguagem de programação, independente do dispositivo ou a linguagem de programação que a aplicação está rodando, a API consegue se comunicar com todos eles, desde que a comunicação seja feita através de um formato em comum, XML ou JSON.


**Figura 3:** Web Service

**Fonte:** (AUGUSTO,2019, n.p.)

Com a informação retornada através desse formato de arquivo, basta que outro sistema consuma suas informações. onde o mesmo pode ser um sistema web, mobile ou desktop. Existem muitos meios de implementar o consumo de uma API. Dentre as ferramentas mais comuns para esse fim, as que são para desenvolvimento web ou mobile são: Angular, React, Vue.JS e etc. Contudo, o consumo desta API não será o foco desse estudo, apenas será apresentado no tópico de desenvolvimento.

## 3. METODOLOGIA

O método indutivo no qual foi utilizado na pesquisa, é responsável por relacionar os dados às teorias, isto é, é um método de abordagem que tem como objetivo fazer conexões com os princípios teóricos. É conhecido como o método de abordagem que segue o "topo para baixo", pois é com base em premissas teóricas que é possível chegar ao particular. Esse método é ideal para conclusões prováveis, que tem como base premissas teóricas.

O método de abordagem é o conjunto de atividades sistemáticas que permite que você alcance os objetivos da pesquisa. É a estratégia de investigação da pesquisa, que segue regras e padrões específicos, para chegar em uma conclusão científica. Em termos mais simples, o método de abordagem refere-se ao conjunto de métodos que vão organizar, de forma lógica, o pensamento para solucionar o problema da pesquisa (BEATRIZ COELHO, 2021).

# 4. RESULTADOS E DISCUSSÕES

Neste tópico iremos apresentar os Resultados obtidos durante todo o processo de desenvolvimento desse projeto, tais como: 4.1 Dados Obtidos na Pesquisa, 4.2 Arquitetura da API desenvolvida e por fim o 4.3 Aplicação, onde mostraremos o funcionamento da API.

## 4.1. Dados Obtidos na Pesquisa

A pesquisa foi realizada por meio de um questionário virtual, disponibilizado em um *link* (por meio de um formulário web elaborado a partir do *Google Forms*, aplicativo do google que permite a criação, compartilhamento e disponibilização de formulário web), com a duração de uma semana para a coleta dos dados, onde no final foi totalizado 30 amostras.

Para aprofundar o tema dessa pesquisa, foram apresentadas 4 questões para os entrevistados, com a finalidade de obter o parecer do objetivo dessa pesquisa.

A primeira questão está relacionada a forma que esses usuários realizam o agendamento de suas consultas médicas: **"Como você costuma agendar as suas consultas atualmente? Se a resposta for online, o agendamento é feito por qual canal de comunicação?"**

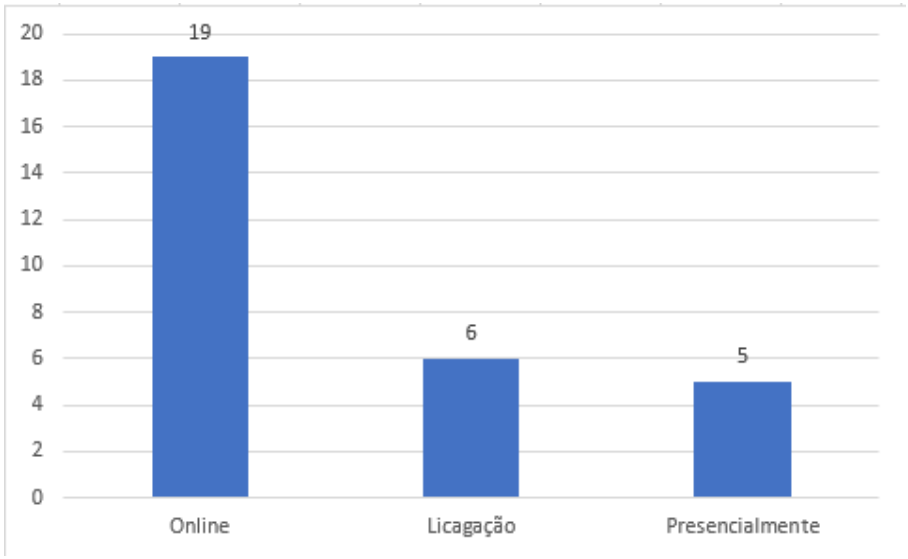



**Gráfico 1.** Análise dos entrevistados quanto a forma de agendar consultas

**Fonte:** Autor (2022)

De acordo com o gráfico 1, a maioria dos entrevistados, ou seja, 63,33% realizam agendamentos para atendimentos clínicos ou hospitalares de forma online, onde, os mesmos relataram que o canal de comunicação utilizada para esses agendamentos é a rede social *WhatsApp.*

Com base nos dados apresentados, pode-se destacar que atualmente as clínicas de saúde ainda não optaram por uma forma mais organizada para os seus controles de atendimento.

Para a segunda questão, está a capacidade de discutir a respeito do histórico de problemas enfrentados por eles quanto ao agendamento de consulta atualmente: **"Você já passou por algum problema em relação ao agendamento de consulta? Se sim, me conte sobre sua experiência".**

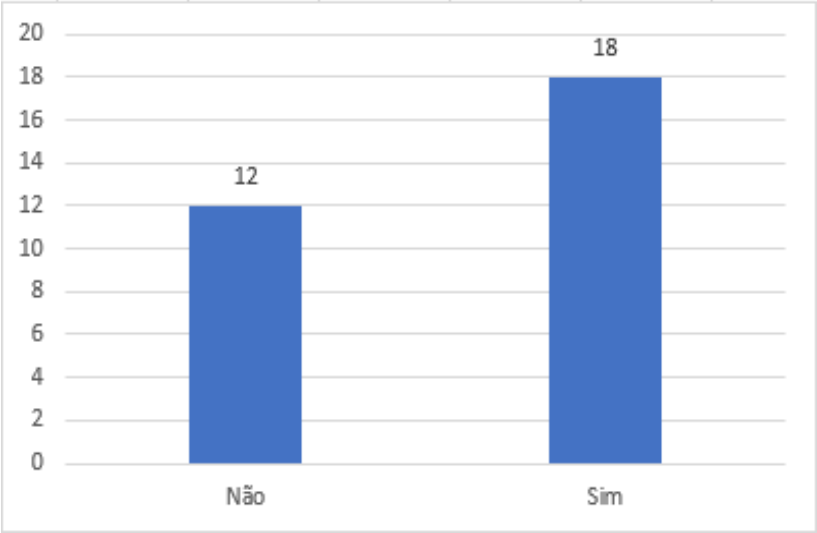



**Grafico 2.** Análise dos entrevistados quanto a problemas com seus agendamentos.

**Fonte:** Autor (2022).

De acordo com o gráfico 2 percebe que a maioria, ou seja 60% dos entrevistados já tiveram problemas com agendamentos de consultas, tendo como destaque o motivo de não terem sido notificados a respeito da remarcação da mesma.

Na terceira questão começamos a abordar sobre a facilidade de termos um aplicativo para o agendamento de consultas tendo como principal característica a opção de buscar pelo profissional de sua preferência: **"Suponhamos que você está precisando agendar uma consulta/exame com uma certa urgência. Opção 1: Utilizaria um aplicativo onde você escolheria o profissional de sua preferência, baseado nas avaliações e no histórico dele / Opção 2: Usaria o próprio canal da clínica/hospital, onde lá eles iriam lhe encaixar na data que você precisa, porém com um profissional aleatório."**

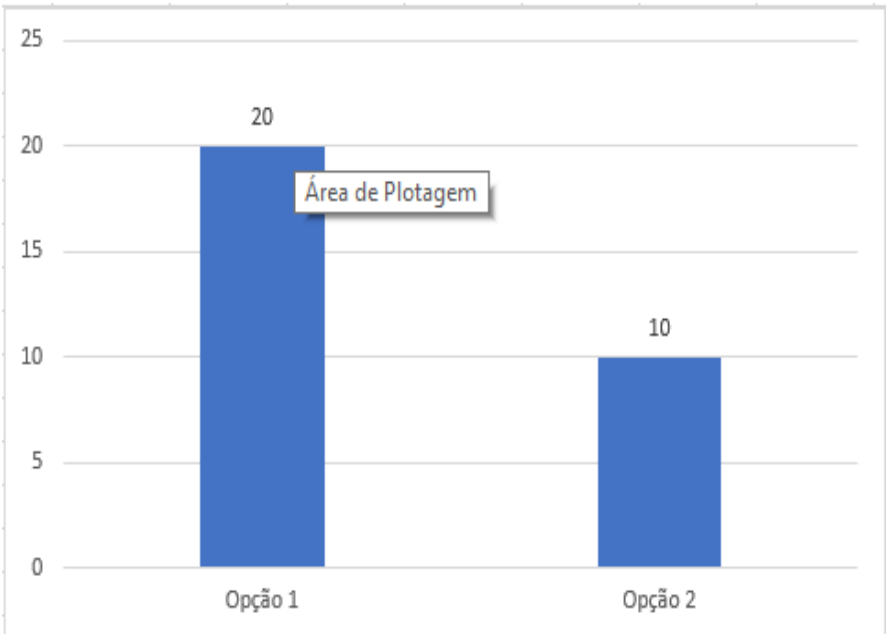



**Gráfico 3.** Análise dos entrevistados sobre a possibilidade de realizar o agendamento através de um app.

**Fonte:** Autor (2022)

No gráfico 3, podemos observar que 66,66% dos entrevistados, ou seja, a grande maioria, optaria pela possibilidade de utilizar um aplicativo onde eles pudessem agendar

as suas consultas. Todos alegaram que, a possiblidade de realizar um agendamento através de um *app*, os trariam mais confiabilidade a respeito do especialista que estão solicitando e por conta da possibilidade de ter um local de segurança para o armazenamento do acompanhamento de sua consulta.

Na quarta questão, partimos pra algo mais técnico, pois queríamos o *feedback* dos nossos entrevistados a respeito usabilidade da *interface* gráfica da aplicação: **"A imagem abaixo é referente a página principal de um aplicativo para agendamento de consultas ou exames online. Você considera essa tela intuitiva? Se você pudesse mudar algo ou acrescentar, o que você faria?"**

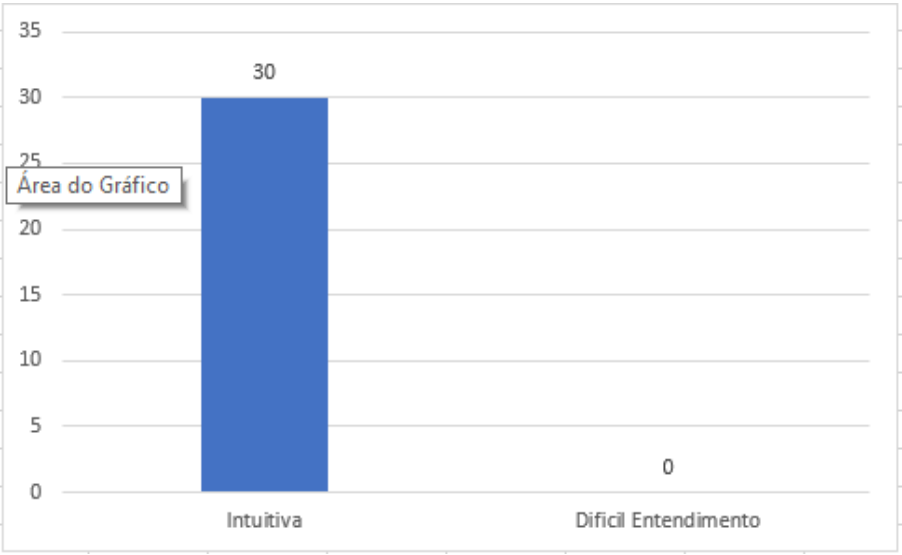



**Gráfico 4.** Análise dos entrevistados sobre a interface gráfica do aplicativo

**Fonte:** Autor (2022)

A partir dos resultados do gráfico 4, observamos que todos os envolvidos na entrevista consideraram a tela do aplicativo intuitiva, ou seja, de fácil manuseio. Obtivemos algumas ressalvas quanto o *layout*, a respeito das cores de alguns botões, dessa forma, iremos pontuar os *feedbacks* obtidos e implementar futuramente.

## 4.2 Aplicação

Baseado na fundamentação teórica, a API foi desenvolvida utilizando o *framework SpringBoot* e para o armazenamento dos dados utilizamos o banco de dados MYSQL.

Com o intuito de simplificar a arquitetura da aplicação. Desenvolvemos um serviço REST API, que é responsável por efetuar todas as operações que serão realizadas no aplicativo, tais como a visualização, atualização, inserção e as exclusões dos registros. Para isso foram criados *endpoints* disponibilizados na API. A figura 4 representa a arquitetura desenvolvida neste projeto.

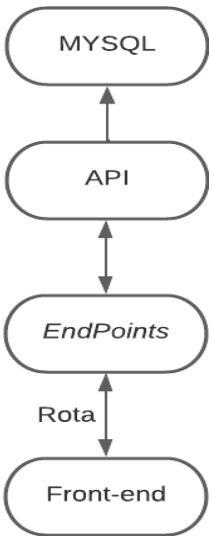



**Figura 4.** Arquitetura do Projeto

**Fonte:** Autor (2022)

Conforme observamos na figura 4, a API desenvolvida será responsável pelo envio e recebimento dos dados do *front-end.*

Nesse tópico, vamos mostrar a funcionalidade de agendamento de consultas. Para isso, iremos utilizar os *endpoints* Usuários e Vendas. Para exibir os resultados das requisições utilizaremos a plataforma Postman, que é uma plataforma de API para desenvolvedores projetar, construir, testar e iterar suas APIs.

### 4.2.1 Cadastro de Usuários

Na API foi desenvolvido o CRUD (Create, Read, Update, Delete), um acrônimo para as maneiras de se operar em informação armazenada, de usuários ou vendas, conforme à regra de negócio estabelecida. Para realizar o armazenamento (Create) as requisições são passadas pelo método POST. O Spring auxilia neste processo, pois basta apenas utilizar a anotação @PostMapping em cima do método desejado e determinar um corpo na requisição através da anotação @RequestBoy seguido do tipo do objeto pretendido no parâmetro do método, conforme na figura 5.

```
@PostMapping
@ResponseStatus(HttpStatus.CREATED)
public Users adicionar(@Valid @RequestBody Users user) {
    return cadUserService.salvar(user);
}
```

**Figura 5:** Método cadastrar usuário

**Fonte:** Autor (2022)

Conforme regra de negócio estabelecida todo usuário que se cadastrar na API poderá selecionar o seu tipo de perfil 1 – Paciente, 2 – Profissional e 3 – Administrador. O usuário poderá ter múltiplos perfis, possibilitando que o mesmo utilize a aplicação para qualquer função. Seja marcar uma consulta como paciente, realizar atendimentos como profissional, ou até mesmo gerenciar sua própria clínica com o perfil Administrador. A figura 6, exibe a realização de uma inserção de um usuário do tipo paciente:

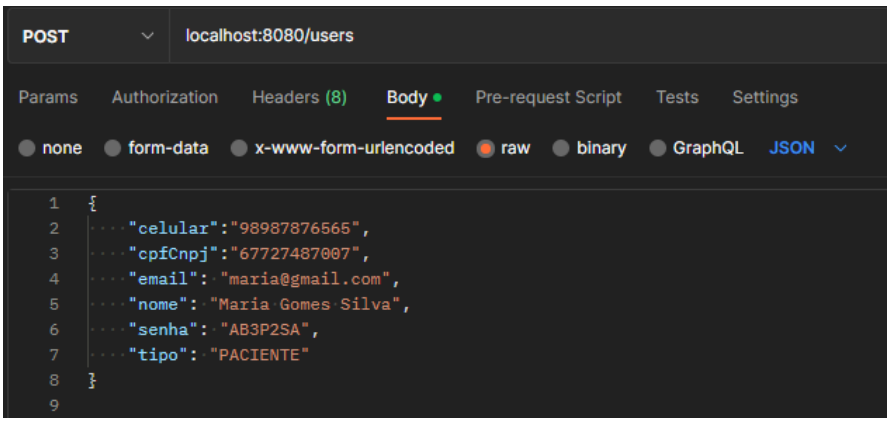



**Figura 6:** Cadastrando um usuário do tipo paciente

**Fonte:** Autor (2022)

Conforme mostrado na figura, primeiro informamos endereço: localhost:8080/users que é a URL do *endpoint* usuários. Depois foi passado em formato JSON os parâmetros de cada atributo necessário para o cadastro de um novo paciente. Na figura 7, podemos observar o status dessa requisição.

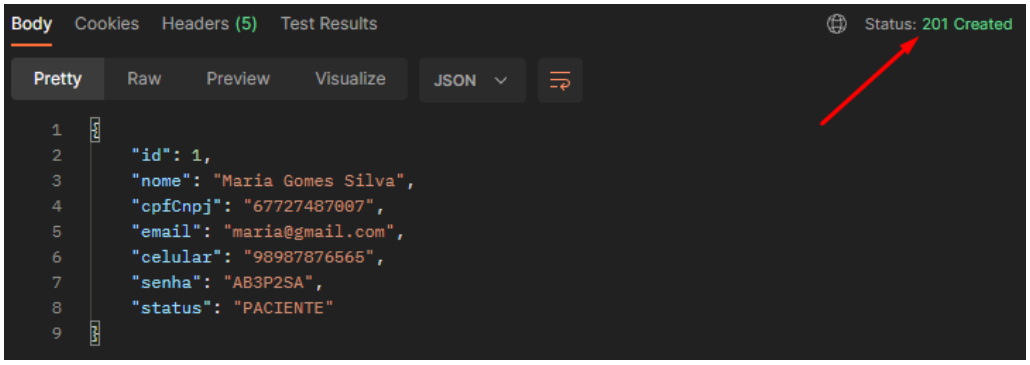



**Figura 7:** Resposta do POST de um novo usuário.

**Fonte:** Autor (2022)

O Postman exibiu no corpo da requisição as informações com todos os dados foram passadas anteriormente. No canto superior o "Status: 201 Created", refere-se a um status HTTP que informa que obtivemos sucesso na nossa resposta. Dessa forma, o usuário já pronto para utilizar o aplicativo a qualquer momento.

## 4.2.2 Acessando os Usuários

Para resgatar a informação dos registros é utilizada a anotação @GetMapping a qual pode ser atribuída o valor, geralmente a variável id, para buscar um objeto em específico, caso contrário, retornará uma lista com todos os objetos salvos como apresentado na figura 8.

```java
@GetMapping
public List<Users> listar() {
    return usersRepository.findAll();

}


@GetMapping("/{userId}")
public ResponseEntity<Users> buscarUsuario(@PathVariable Long userId) {
    Optional<Users> user = usersRepository.findById(userId);

    if(user.isPresent()) {
        return ResponseEntity.ok(user.get());
    }
    return ResponseEntity.notFound().build();

}
```

**Figura 8:** Trecho do código do método GET, para acessar usuários.

**Fonte:** Autor (2022)

A URL que utilizaremos para buscar as informações de todos os usuários será localhost:8080/users, de forma que o Postman nos retornará os seguintes registros, conforme mostrado na figura 9:

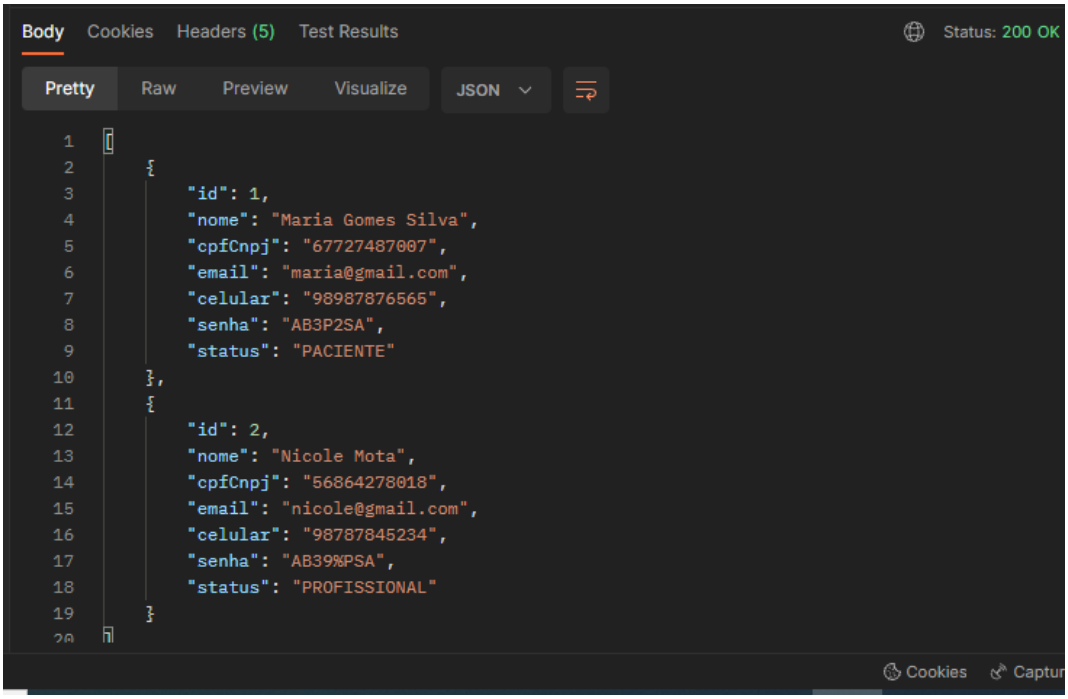



**Figura 9:** Resposta do GET de todos os usuários.

**Fonte:** Autor (2022)

Podemos observar que o status da requisição foi o 200, significando que obtivemos sucesso na utilização desse recurso.

### 4.2.2.1 Consumindo a API

Para realizar a autenticação de um usuário, e o mesmo conseguir logar no aplicativo, é necessário passar o *ID* na URL desse *endpoint*, ficando da seguinte forma: localhost:8080/users/{idUser}. Na figura 10 podemos observar de forma mais clara um exemplo dessa requisição.

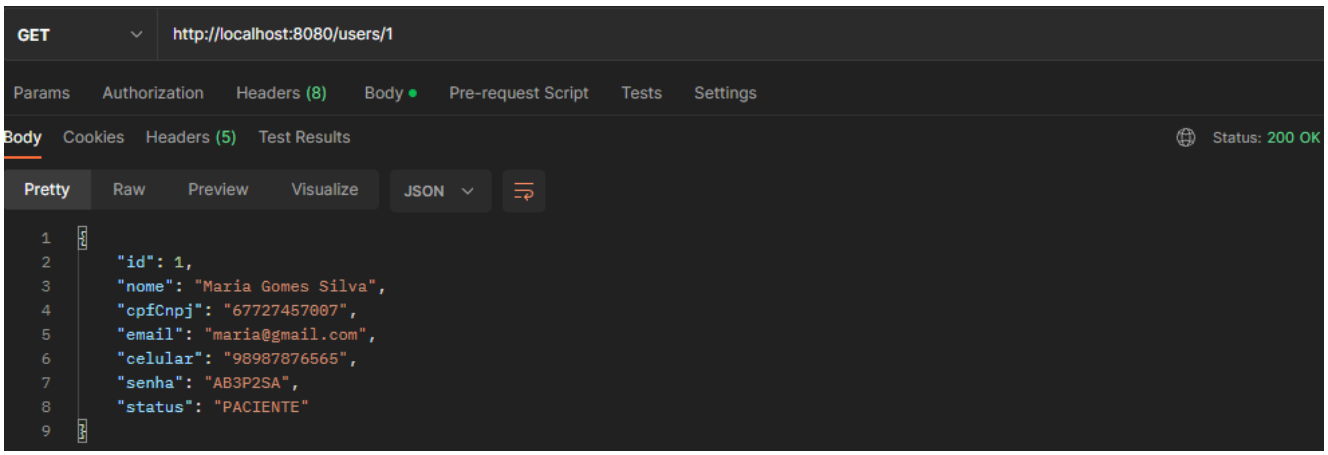



**Figura 10:** Resposta do GET de apenas um usuário.

**Fonte:** Autor (2022)

Com esse recurso implementado, o *front end* da aplicação ficará livre para escolher quais dados serão necessários para ser autenticado e assim, por fim, conseguir acessar de forma prática a aplicação. A figura 11, mostra o exemplo de um usuário logado na página principal do aplicativo.

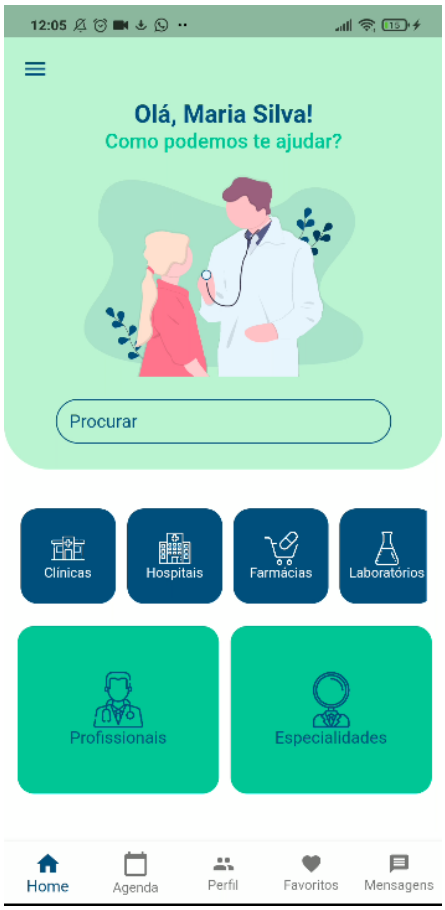



**Figura 11:** Usuário logado

**Fonte:** Autor (2022)

Conforme observamos, foi efetuado o login do usuário cadastrado como exemplo no capítulo 4.2.1. Agora ele está pronto para navegar pelo aplicativo, podendo utilizar todas as funcionalidades disponíveis a ele pelo seu tipo de perfil, como por exemplo, agendar suas consultas.

### 4.2.3 Agendamento de Consulta

Seguindo a mesma ideia do CRUD, método utilizado para as operações da API, conforme abordado no capítulo 4.2.2. A figura 12 mostra a inserção de um registro de venda:

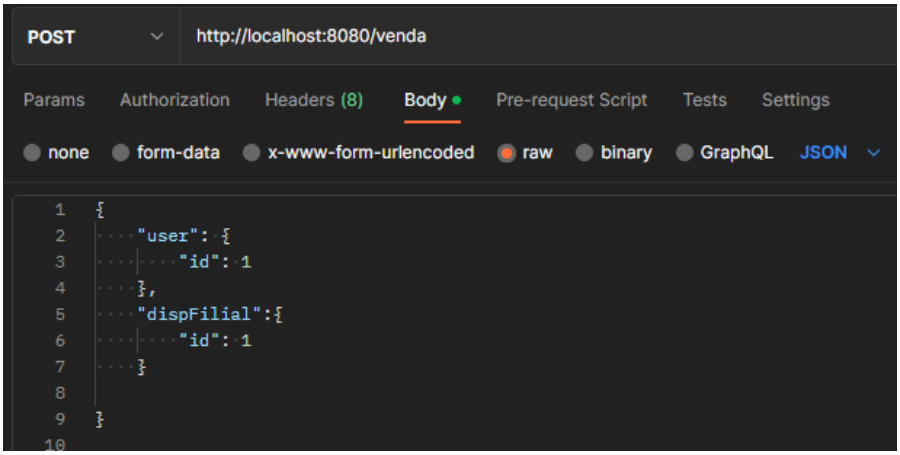



**Figura 12:** Criando um Registro de Venda.

**Fonte:** Autor (2022)

Conforme ilustrado na figura 10, para realizar o POST de uma venda, a URL que utilizamos foi: localhost:8080/venda e é necessário apenar utilizar o *ID* do usuário e o *ID* da disponibilidade da filial (Outro *endpont* criado para controlar os horários disponíveis de um profissional em uma determinada filial). Na figura 13, podemos observar a resposta dessa requisição.

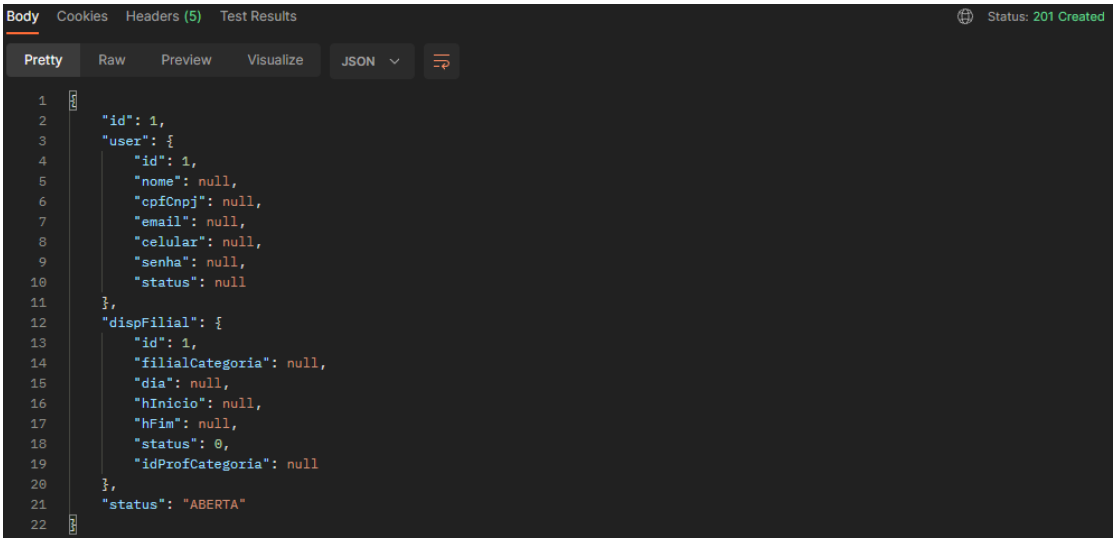



**Figura 13:** Resposta do POST de venda.

**Fonte:** Autor (2022)

Fazendo um GET desse registro criado, poderemos observar todos os seus dados conforme os parâmetros que foram passados. Abaixo na figura 14 temos esse exemplo:


**Figura 14:** Resposta do GET da venda 1, para acessar o registro criado anteriormente.

**Fonte:** Autor (2022)

Como observamos, apenas a partir do *ID* do usuário, e do *ID* da disponibilidade da filial, foi criado uma venda com o status em aberto. Isso foi possível por conta do relacionamento com as outras entidades que o Spring nos proporcionou.

Para atualizarmos o objeto cadastrado é usado o método PUT, @PutMapping, que também receberá um *ID* para identificar o objeto em questão. Neste método pode-se aplicar dois parâmetros: um identificador com a anotação @PathVariable e a @RequestBody para o conteúdo a qual deseja atualizar, seguindo o modelo da figura 15.

```
@PutMapping("/{vendaId}")
public ResponseEntity<Venda> update(@Valid @PathVariable Long vendaId , @RequestBody Venda venda){
    if(!vendasRepository.existsById(vendaId)) {
        return ResponseEntity.notFound().build();
    }

    venda.setId(vendaId);
    regVendaService.salvar(venda);


    return ResponseEntity.ok(venda);
}
```

**Figura 15:** Trecho do código do método PUT, para atualizar um objeto.

**Fonte:** Autor (2022)

Conforme a regra de negócio estabelecida no código na figura à cima, para realizarmos um *update* de algum objeto da nossa entidade de Venda, é necessário passar o *ID* da mesma, conforme ilustrado na figura 63 abaixo.

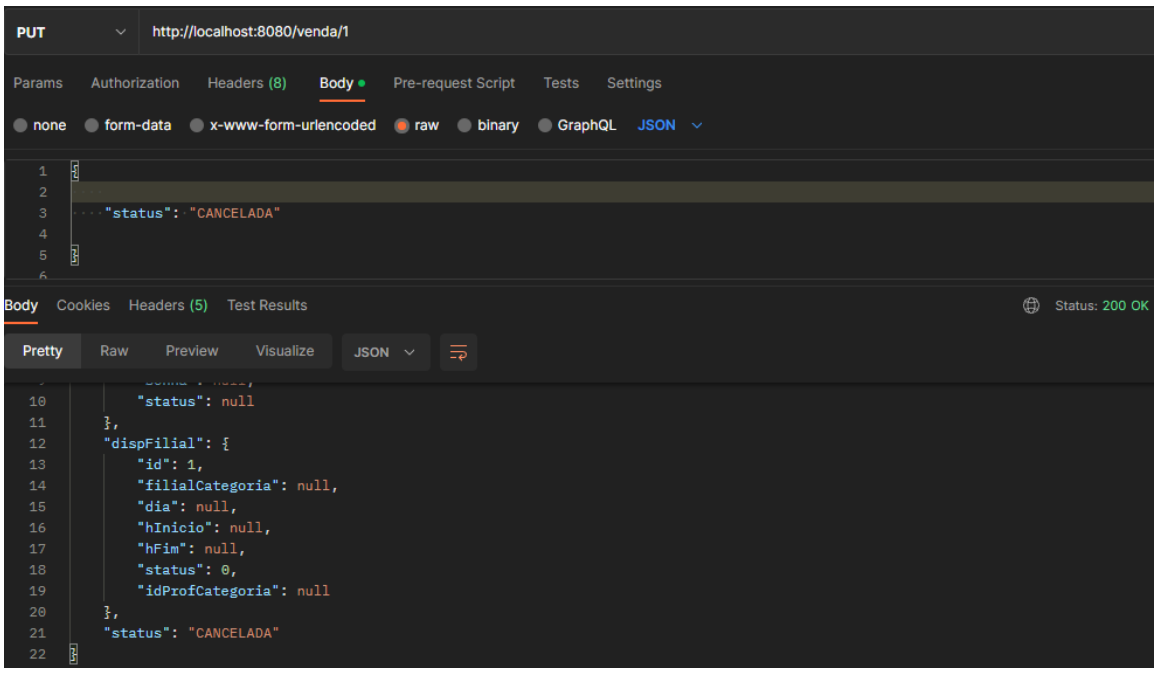



**Figura 16:** Atualizando a venda 1 para o status "cancelada", usando o método PUT

**Fonte:** Autor (2022)

Utilizamos a URL localhost:8080/venda/{idVenda}, para realizar a atualização do status dessa venda, tornando necessário passar apenas o parâmetro que seria alterado.

Para finalizar, vamos deletar essa venda usando o método DELETE. Para isso, deve-se utilizar a anotação @DeleteMapping, atribuindo um valor que se refere ao identificador do objeto e informar este *ID* com a anotação @PathVariable para buscar o recurso alvo, seguindo o modelo da figura 14.

```
@DeleteMapping("/{vendaId}")
public ResponseEntity<Void> delete(@PathVariable Long vendaId){
    if(!vendasRepository.existsById(vendaId)) {
        return ResponseEntity.notFound().build();
    }

    regVendaService.excluir(vendaId);

    return ResponseEntity.noContent().build();
}
```

**Figura 17:** Trecho do código do método DELETE, para excluir uma venda.

**Fonte:** Autor (2022)

Conforme ilustrado na figura, será necessário apenas informar o *ID* do registro de venda, que dessa forma será deletado todos os objetos dessa entidade. Na figura 15 abaixo, temos um exemplo da realização dessa operação.

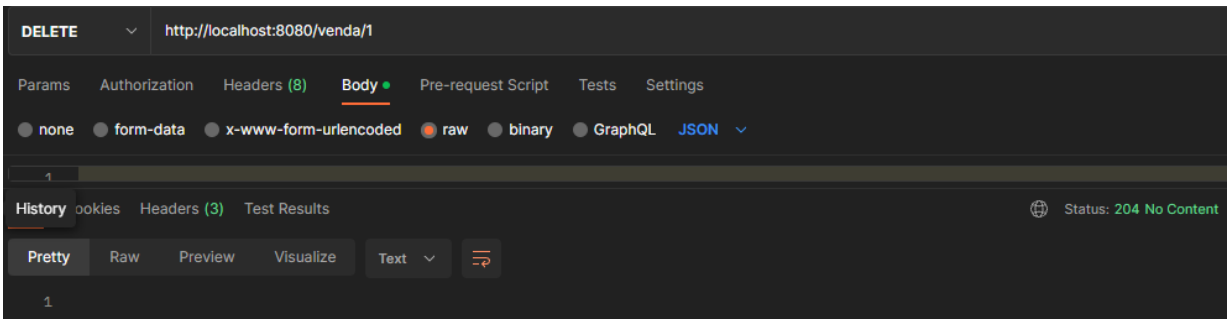



**Figura 17:** Resposta do DELETE, responsável por excluir um registro.

**Fonte:** Autor (2022)

Utilizamos a URL localhost:8080/venda/{idVenda} para identificar o registro que seria excluído, dessa forma, como observado, após realizar a operação, foi retornado o "Status: 204 No Content", status HTTP que indica que obtivemos sucesso na requisição e o nosso registro foi deletado com sucesso.

## 4. CONCLUSÃO

Foi destacado no projeto em questão, que sistemas que são construídos sobre os princípios de API REST oferecem os mecanismos mais modernos para que um sistema rode com um bom funcionamento em variedades de tipos de aparelhos ou sistemas operacionais. Isso se torna vantajoso por conta da variedade de equipamentos, onde cada um funciona com seu sistema operacional, na maioria dos casos não se torna viável construir um sistema para cada aparelho ou SO, pois isso gera custos extraordinários.

O framework spring é uma ferramenta muito poderosa para um desenvolvedor, é usada amplamente na pela comunidade java. Vale destacar sobretudo, a produtividade que foi proporcionada por ele e que, através de metodologias que foram aplicadas sobre os estudos de Padrões de Projeto, MVC, e Programação Orientada a Objetos, o processo de desenvolvimento desse projeto se deu com um curto período de tempo e consequentemente diminuiu os custos do desenvolvimento. O back-end da aplicação produzida está em um repositório no github.

### 4.1 Trabalhos Futuros

Este projeto teve como foco o desenvolvimento Backend de um sistema de gerenciamento de clínicas de saúde e agendamento de consultas online. Posteriormente a API será hospedada no Tomcat, uma plataforma em nuvem para fazer deploy de aplicações e será desenvolvido o Front-end, na qual consumirá a API desenvolvida para executar a regra de negócio e com isso obter o gerenciamento dos serviços.

## Referências

ALLAN RAMOS. **MVC – Afinal, é o quê?** Disponível em http://tableless.com.br/mvc-afinal-e-o-que/Publicado no dia 26 de fevereiro de 2015.

AUGUSTO, Cassio. **Web Services: O que é, como funciona e os protocolos SOAP e REST**. Ninja do Linux, 2019.

COAD, P., YURDON, E. **Análise baseada em objetos**. Rio de Janeiro, 1992 DEITEL, Paul;

COELHO, Beatriz, **Método indutivo: um guia sobre esse método de abordagem**, 19 de março de 2021. Disponível em: <https://blog.mettzer.com/metodo-indutivo/>. Acessado em: 19/11/2022.

DEITEL, Harvey**. Como programar**. 6. ed. São Paulo: Pearson Education, 2011.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Padrões de Projeto: Soluções reutilizáveis de software orientado a objetos.** 1. ed. Porto Alegre: Bookman, 2002.

MEDEIROS, Higor**. Application Programming Interface: Desenvolvendo APIs de Software**: 12 de março de 2012. Disponível em: <http:www.devmedia.com.br/application-programming-interface-desenvolvendo-apis-de-software/30548>.

Módulos Spring Framework. Pivotal Software, Inc, 2018. Disponível em: https://docs.spring.io/spring/docs/3.0.0.M4/reference/html/ch01s02.html. Acessado em: 03/10/2021

PEREIRA, Weberson. **API: conceito, exemplos de uso e importância da integração para desenvolvedores**. Take, 2019.

PUGA, Sandra; RISSETTI, Gerson. **Lógica de programação e estrutura de dados com aplicações em Java.** São Paulo: Pearson Education do Brasil, 2004.

SOMMERVILLE, Ian. **Engenharia de Software**. 10. ed. São Paulo: Pearson Prentice Hall, 2011. Spring Framework. Pivotal Software, Inc, 2018. Disponível em: https://spring.io/.

# 8

## O USO DA MACHINE LEARNING COM O ALGORITMO DE REGRESSÃO LOGÍSTICA PARA REALIZAR ANÁLISE DE DADOS DE DOENÇAS CARDIOVASCULARES DE UM BANCO DE DADOS PÚBLICO DA CC BY 4.0

*THE USE OF MACHINE LEARNING WITH THE LOGISTIC REGRESSION ALGORITHM TO PERFORM DATA ANALYSIS OF CARDIOVASCULAR DISEASES FROM A CC BY 4.0 PUBLIC DATABASE*

Ricardo Santos Silva[1]

Edilson Carlos Silva Lima[2]

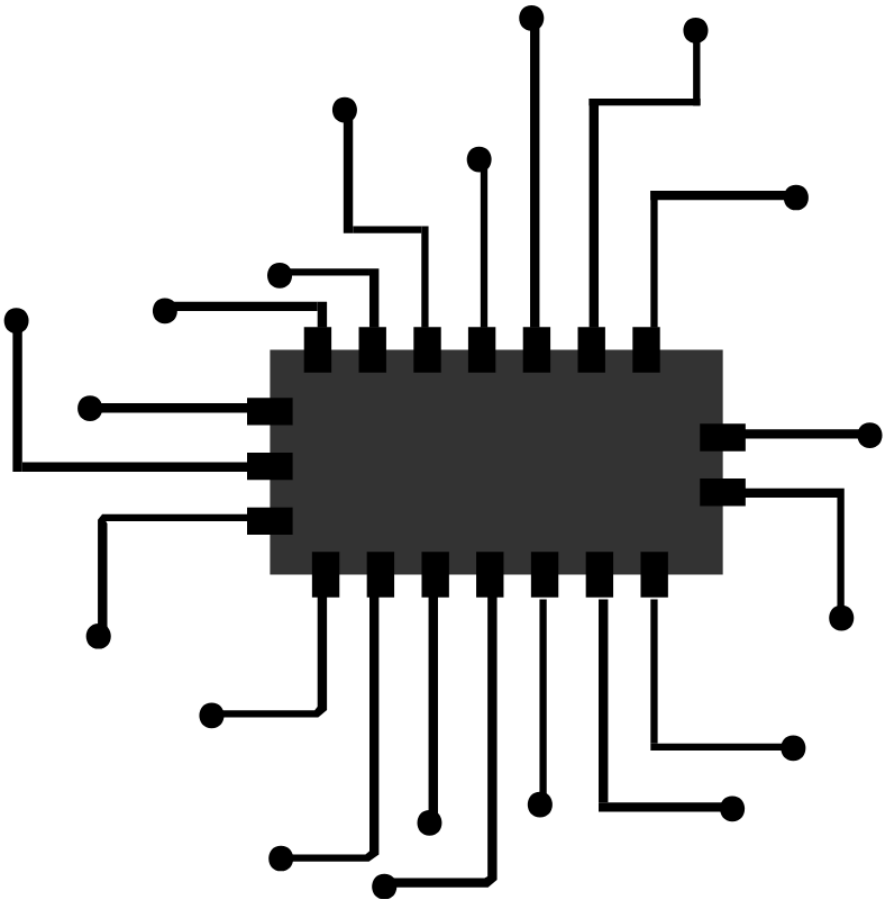
Jonathan de Araújo Queiros[3]

[1]Engenharia de Computação - Universidade Ceuma (UniCEUMA) - São Luís - MA - Brasil

[2]Engenharia de Computação - Universidade Ceuma (UniCEUMA) - São Luís - MA – Brasil

3Engenharia de Computação - Universidade Ceuma (UniCEUMA) - São Luís - MA – Brasil

{SILVA, Ricardo, ricardosantoscomp@gmail.com; LIMA, Edilson Carlos Silva, edilsonlima3@gmail.com; QUEIROS, Jonathan de Araújo, queirozjth@gmail.com}



**d.o.i.:**

## Resumo

A doença cardiovascular (DCV) é uma adversidade que afeta o coração e os vasos sanguíneos, e estima-se que 17,9 milhões de pessoas morreram em 2019 (WHO, 2021). A crescente importância da análise de dados em informática médica despertou o interesse na geração de modelos analíticos em aprendizado de máquina (ML). A previsão de DCV é um desafio complexo e a classificação usando ML desempenhará um papel importante na previsão e investigação de doenças cardíacas para mitigar os efeitos cardíacos e evitar a mortes prematuras. O objetivo deste trabalho é analisar dados treinando com *Machine Learning* e o algoritmo de Regressão Logística para avaliar modelos e aprendizado profundo para prever diagnósticos de doenças cardiovasculares.


**Palavras-chaves**: *Machine Learning*; Algoritmo de Regressão Logística; Análise de Dados.


## Abstract

Cardiovascular disease (CVD) is an adversity that affects the heart and blood vessels, and an estimated 17.9 million people died in 2019 (WHO, 2021). The growing importance of data analysis in medical informatics has sparked interest in generating analytical models in machine learning (ML). CVD prediction is challenging and a classification using ML will play an important research complex in predicting heart disease to mitigate cardiac deaths and prevent premature death. The aim of this work is healthy and to evaluate machine and deep learning models to predict diagnoses of cardiovascular diseases.


**Keywords:** machine learning; cardiovascular diseases; data analysis.

## 1. INTRODUÇÃO

A doença cardiovascular (DCV) é uma adversidade que afeta o coração e os vasos sanguíneos. Também conhecida como doença cardíaca, a incidência dessas doenças se deve principalmente a bloqueios que estreitam as artérias, impedindo o fluxo sanguíneo para o coração ou cérebro. Isso inclui várias doenças, como doença cardíaca coronária, doenças cerebrovasculares, doenças arteriais periféricas, doenças reumáticas e outras doenças. Em 2019, estima-se que as DCV tenham sido a causa de óbito de 17,9 milhões de pessoas, ou seja, 32% das mortes daquele ano (WHO, 2021).

Os fatores de risco para doenças cardiovasculares incluem fatores comportamentais, como tabagismo, dieta não saudável, uso nocivo de álcool e inatividade física, e fatores fisiológicos (metabólicos), incluindo pressão alta e níveis elevados de colesterol e glicose. Devido à enorme influência da informação disponível em várias formas, o papel da análise de dados em informática médica cresceu rapidamente na última década. Isso também levou a um crescente interesse na construção de modelos analíticos baseados em dados de *Machine Learning* (ML, em português, Aprendizado de Máquina) para computação médica.

A previsão de doenças cardiovasculares é um dos desafios mais complexos na análise de dados clínicos. No entanto, a classificação usando ML desempenha um papel importante na previsão de doenças cardíacas e revisão de dados para reduzir o impacto cardíaco e prevenir possível morte prematura. A doença cardiovascular pode ser amenizada através do diagnóstico precoce, reduzindo assim a mortalidade. Portanto, o uso de ML para identificar fatores de risco é uma importante abordagem potencial para a melhora da doença. Vários estudos têm desenvolvido modelos preditivos com o objetivo de prever doenças cardiovasculares. Por exemplo, em Al-Absi et al. (2021), os autores tentaram desenvolver um modelo na base de dados do Qatar Biobank (QBB) para identificar indivíduos saudáveis e pacientes com doença cardiovascular, que pudesse detectar e analisar os principais fatores de risco relacionados à doença no Qatar. Outro exemplo é o estudo desenvolvido por Smigiel et al. (2021), que realizou de diferentes classes diagnosticais de DCV usando monitoramento de eletrocardiogramas (ECG).

Dado o histórico, o principal objetivo do trabalho atual é treinar e avaliar modelos de aprendizado de máquina e aprendizado profundo para prever o prognóstico de doenças cardiovasculares. Como contribuição, este trabalho destaca o uso de Regressão Logística para avaliação de séries temporais e comparação com outros dados já apurados.

O trabalho está organizado da seguinte forma: a seção 2 apresenta alguns trabalhos relacionados a esta proposta de trabalho. A seção 3 apresenta a metodologia e seus passos específicos para orientar este trabalho. Os resultados e análises obtidos são apresentados e discutidos na Seção 4. A Seção 5 conclui com comentários sobre as considerações finais identificadas neste trabalho.

## 2. FUNDAMENTAÇÃO TEÓRICA

O diagnóstico precoce é um passo importante para atingir o objetivo de reduzir o impacto e as consequências das doenças cardiovasculares. Ghosh et al. (2021), semelhante a muitos estudos, tentou classificar as DCV usando ML. Um modelo com 99,05% de precisão foi projetado com bons resultados utilizando Random Forest Packing (RFBM) para classificar doenças cardíacas.

Al-Absi et al. (2021) construíram um modelo de ML para distinguir indivíduos saudáveis de pacientes com doenças cardiovasculares para revelar uma lista de potenciais fatores de risco relacionados à doença. O estudo examinou uma série de medidas biomédicas, incluindo medidas comportamentais, representando as várias medidas biomédicas CVD do Qatar Biobank (QBB). Portanto, os autores listaram o CatBoost como o melhor modelo com 93% de precisão. Além dos fatores de risco conhecidos, como doença renal, função hepática, aterosclerose etc. Outros achados foram a identificação de um novo conjunto de fatores de risco.

Cui et al. (2020) indicaram que níveis elevados de lipídios no sangue são um dos fatores de risco mais importantes para doenças cardiovasculares. Assim, com a previsão precoce de valores anormais de lipídios, a intervenção precoce é possível e reduz o risco de desenvolver dislipidemia, ou seja, colesterol ou lipídios no sangue anormalmente elevados. Portanto, o objetivo do estudo foi prever o risco de desenvolver dislipidemia em trabalhadores de siderúrgicas. Naquele estudo, a rede neural LSTM foi o melhor modelo, com acurácia[1] superior a 95%.

Strodthoff et al. (2020) Uma análise comparativa do conjunto de dados de ECG clínicos, o PTB-XL. Como contribuição, este trabalho implementa e aplica diversos modelos modernos de *Deep Learning* (DL), além de fornecer um banco de dados para mitigar o problema da análise automatizada de ECG. Dentre todos os modelos avaliados, uma rede neural convolucional com arquiteturas *ResNet* e *Inception* tem o melhor desempenho na tarefa proposta.

Este trabalho propõe um método para construção de modelos de redes neurais, usando algoritmos de ML. Para o modelo de treinamento e teste foi utilizado um classificador baseado em Regressão Logística, visto que a primeira proposta feita utilizando Random Forest teve uma acurácia menor que a atual. O modelo de Regressão Logística, é empregado para modelar a relação entre uma variável dependente categórica e um conjunto de variáveis explanatórias.

Os trabalhos apresentados nesta seção são consistentes com o presente trabalho na execução de tarefas de aprendizado de máquina para prever eventos de saúde, particularmente doenças cardiovasculares. Portanto, assim como o presente trabalho, ambos propõem uma tarefa de classificação. O trabalho a ser apresentado de maneira mais detalhada é uma nova contribuição para o estudo de diferentes modelos ML utilizando a base de classificação 

## 2.1 Metodologias

A primeira fase do desenvolvimento da pesquisa consiste na busca e seleção de um bom banco de dados no nicho da saúde, mais especificamente sobre doença cardiovascular estruturado para estudos de aprendizagem supervisionada. O processo de pesquisa de ML requer atividades que incluem pré-processamento e preparação de dados após a aquisição do data set. O objetivo da etapa de pré-processamento é usar técnicas de visualização para entender os dados e fazer insights sobre tendências e qualidade dos dados, além de fazer hipóteses e suposições na análise antes de ir para parte prática. Durante a preparação dos dados, a limpeza dos dados é realizada para otimizar a qualidade dos dados, que visa reduzir as amostras com valores ausentes, atributos com valores nulos e dados discrepantes.

Uma etapa muito importante antes que o modelo preditivo receba os dados é a vi-

1 Acurácia é uma espécie de soma entre exatidão e precisão

sualização e análise dos atributos. Em relação aos atributos, é possível editar atributos deletando, convertendo e buscando novos já fornecidos pelo conjunto de dados, isso se faz necessário quando existe atributos sem valor especificado ou com valor atribuído de forma erronia no data set. Além disso, é possível converter os dados brutos em formatos mais adequados ao processo, o que não foi necessário nesta pesquisa.

## 2.1 Materiais e Métodos

Para atingir os objetivos desejados e testar as hipóteses experimentais formuladas anteriormente, foi necessário decidir quais as ferramentas que seriam usadas para tal processo. Visto que este é um estudo sobre ML, a linguagem de programação é o ponto mais importante a ser decidido, existem diversas linguagens que tem o poder de fazer algoritmos de ML, as mais usadas são Python, Linguagem R, Scala, Java, Julia, C, etc. No presente trabalho, a linguagem de programão utilizada foi Python.

Python é uma linguagem ágil, simples e objetiva, principalmente se tratante da área de análise de dados e apesar da sua simplicidade, ela é uma linguagem muito robusta que exerce seu papel sem apresentar nenhum problema, conta também com uma sintaxe fácil de entender. Python desenvolveu uma comunidade grande e ativa de processamento científico e análise de dados. Nos últimos dez anos, Python passou de uma linguagem de computação científica inovadora, ou para ser usada "por sua própria conta e risco", para uma das linguagens mais importantes em ciência de dados e aprendizado de máquina (MCKINNEY, 2018)

Temos também o Jupyter Notebook, que é uma excelente opção para quem deseja criar códigos em Python, ele é se utiliza de uma interface web que permite prototipagem rápida e compartilhamento de projetos relacionados a dados e foi utilizado para fazer este modelo.

O modelo presente utilizará algoritmos que se enquadrem como ML tradicional de modelo preditivo de regressão logística que faz parte da biblioteca Scikit-Learn. Esta biblioteca se transformou no principal kit de ferramentas para aprendizado de máquina. Além desta se fará o uso também do Pandas e Numpy.

# 3. ESTUDO DE CASO

O trabalho presente utiliza dados do banco de dados CC BY 4.0 fornecido pelo UC *Irvine Machine Learning Repository* (UCI), um projeto com colaboração com Rexa na Universidade de Massachusetts Amherst e apoio financeiro da *National Science Foundation*, onde foi coletado dados de 4 banco de dados distintos (Hospital Universitário de Zurique - Suíça, Instituto Húngaro de Cardiologia – Hungria, Hospital Universitário de Basel - Suíça e VA Medical Center de Long Beach). Todos os dados coletados são reais e disponibilizados para pesquisas e estudos de casos.

## 3.1 Dados e Ferramentas

A primeira etapa do desenvolvimento da pesquisa foi analisar os atributos do banco de dados obtidos para o estudo, ao total são 303 linhas e 14 colunas compondo o Dataset, as linhas se referem a quantidade de pacientes analisados e as colunas ao total de atributos a serem analisados de cada paciente, a fim de gerar um modelo de treinamento e

teste para a otimização de exames reais. Em cada paciente analisado, já vem o resultado se ele porta ou não doença cardiovascular e essa informação prévia será responsável para fazer o treinamento do modelo e calcular sua acurácia.

## 3.2 Bibliotecas Utilizadas

As bibliotecas Python são uma coleção de módulos e funções úteis que reduzem o uso de código de programação e sendo possível processar uma montanha de dados com elas de forma mais automatizadas. A figura 1, mostra quais bibliotecas foram utilizadas neste projeto.

```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import classification_report, confusion_matrix
```

**Figura 1:** Bibliotecas importadas

**Fonte:** Autoral, 2022.

O Numpy tem como função trabalhar operações numéricas, já o Pandas vai fazer a manipulação dos dados de forma geral, as bibliotecas baseadas em Sklearn vai servir para fazer a modelagem estatística, análise e mineração de dados, além de suporte ao aprendizado supervisionado e não supervisionado.

## 3.3 Coleta e Processamento de dados

Esses dados são devidamente preparados e passam por alguns processos de limpeza e ajuste de pré-processamento e seleção de variáveis para que possam ser alimentados a um algoritmo de aprendizado de máquina que faz previsões e verifica novamente até que ponto o resultado está no valor correto. Ajustar os parâmetros usados na previsão ajuda a obter um valor mais adequado.

A biblioteca Pandas terá um trabalho essencial na parte presente, onde será visualizado os dados em conjuntos de forma mais detalhadas. A figura 2, mostra o carregamento dos dados.

```
In [2]: # Carregando os dados csv para o Pandas DataFrame
        heart_data = pd.read_csv('heart_disease_data.csv')
```

**Figura 2:** Carregamento dos dados

**Fonte:** Autoral, 2022.

Todos os dados estão armazenados em um arquivo de informações separadas por vírgulas (CSV) localizado na própria memória do computador e precisa ser puxado através do código descrito na figura acima para, assim, fazer o tratamento e manipulação dos dados.

Agora vamos visualizar as 5 primeiras linhas e as 5 últimas do nosso Data Frame, conforme mostra a figura 3, para ter noção do modelo dos dados de cada atributo apresentado.

```
In [3]: # Imprimir as 5 primeiras linhas do DataFrame
heart_data.head()
```

Out[3]:

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 63 | 1 | 3 | 145 | 233 | 1 | 0 | 150 | 0 | 2.3 | 0 | 0 | 1 | 1 |
| **1** | 37 | 1 | 2 | 130 | 250 | 0 | 1 | 187 | 0 | 3.5 | 0 | 0 | 2 | 1 |
| **2** | 41 | 0 | 1 | 130 | 204 | 0 | 0 | 172 | 0 | 1.4 | 2 | 0 | 2 | 1 |
| **3** | 56 | 1 | 1 | 120 | 236 | 0 | 1 | 178 | 0 | 0.8 | 2 | 0 | 2 | 1 |
| **4** | 57 | 0 | 0 | 120 | 354 | 0 | 1 | 163 | 1 | 0.6 | 2 | 0 | 2 | 1 |

```
In [4]: # Imprimir as 5 ultimas linhas do DataFrame
heart_data.tail()
```

Out[4]:

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **298** | 57 | 0 | 0 | 140 | 241 | 0 | 1 | 123 | 1 | 0.2 | 1 | 0 | 3 | 0 |
| **299** | 45 | 1 | 3 | 110 | 264 | 0 | 1 | 132 | 0 | 1.2 | 1 | 0 | 3 | 0 |
| **300** | 68 | 1 | 0 | 144 | 193 | 1 | 1 | 141 | 0 | 3.4 | 1 | 2 | 3 | 0 |
| **301** | 57 | 1 | 0 | 130 | 131 | 0 | 1 | 115 | 1 | 1.2 | 1 | 1 | 3 | 0 |
| **302** | 57 | 0 | 1 | 130 | 236 | 0 | 0 | 174 | 0 | 0.0 | 1 | 1 | 2 | 0 |

**Figura 3:** Visualização dos dados

**Fonte:** Autoral, 2022.

É notório que eles seguem um padrão, visto que todas essas informações foram coletadas de pessoas reais. Para uma melhor resolução da quantidade de dados presente no Data Frame foram utilizados o comando mostrado na figura 4.

```
In [5]: # Numero de linhas e colunas do dataset
heart_data.shape

Out[5]: (303, 14)
```

**Figura 4:** Quantidade de dados

**Fonte:** Autoral, 2022.

Vemos que, como descrito antes, temos dados de 303 pacientes, outras duas importantes funções apresentadas a seguir para a uma visualização mais profunda é saber quais os tipos de dados foram trabalhados, conforme mostra a figura 5.

```
In [6]: # Obter algumas informações sobre os dados
        heart_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

**Figura 5:** tipo de dados

**Fonte:** Autoral, 2022.

Portanto, é visível que entre os 14 atributos, 13 são do tipo inteiros (INT) e apenas 1 do tipo FLOAT, ou seja, números "quebrados". É de suma importância também verificar se existe algum atributo com valores ausente, o comando para tal finalidade é mostrado na figura 6.

```
In [7]: # Verificando valores ausentes
        heart_data.isnull().sum()

Out[7]: age         0
        sex         0
        cp          0
        trestbps    0
        chol        0
        fbs         0
        restecg     0
        thalach     0
        exang       0
        oldpeak     0
        slope       0
        ca          0
        thal        0
        target      0
        dtype: int64
```

**Figura 6:** verificação dos atributos

**Fonte:** Autoral, 2022.

Dentre todos, nenhum atributo possui valor nulo, como visto na figura acima, isso é muito importante para o projeto, pois poderia acontecer algum erro na hora de treinar

ou testar o modelo. Outro fator importante a ser apresentado, conforme é mostrado na figura 7, é a exibição de quantas pessoas têm ou não doenças cardiovasculares.

```
In [9]: # Verificando a distribuição da Feature Target
        heart_data['target'].value_counts()

Out[9]: 1    165
        0    138
        Name: target, dtype: int64
```

**Figura 7:** identificação dos pacientes

**Fonte:** Autoral, 2022.

Como mostrado na figura acima, o atributo chamado "target" é que mostra qual paciente tem o não doenças cardíacas, e no nosso Data Frame de estudo de caso, 165 pacientes têm e os outros 138 pacientes não têm, ou seja, o número 1 diz que o paciente porta doença e o número 0 que não porta.

## 3.4 Dados de Treinamento e Dados de teste

Uma das bases do aprendizado de máquina são os dados históricos, os algoritmos de aprendizado supervisionado de máquina precisam aprender e, para isso, quanto mais dados usar, melhor será o modelo, por isso acontece essa fase de treinar modelos antes de ir para os testes, conforme mostra a figura 8.

```
In [10]: #Dividindo as Features e Target
         X = heart_data.drop(columns='target', axis=1)
         Y = heart_data['target']

In [11]: print(X)

      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0      63    1   3       145   233    1        0      150      0      2.3
1      37    1   2       130   250    0        1      187      0      3.5
2      41    0   1       130   204    0        0      172      0      1.4
3      56    1   1       120   236    0        1      178      0      0.8
4      57    0   0       120   354    0        1      163      1      0.6
..    ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
298    57    0   0       140   241    0        1      123      1      0.2
299    45    1   3       110   264    0        1      132      0      1.2
300    68    1   0       144   193    1        1      141      0      3.4
301    57    1   0       130   131    0        1      115      1      1.2
302    57    0   1       130   236    0        0      174      0      0.0

     slope  ca  thal
0        0   0     1
1        0   0     2
2        2   0     2
3        2   0     2
4        2   0     2
..     ...  ..   ...
298      1   0     3
299      1   0     3
300      1   2     3
301      1   1     3
302      1   1     2

[303 rows x 13 columns]
```

**Figura 8:** separação dos atributos

**Fonte:** Autoral, 2022.

O primeiro passo antes de treinar o modelo é separar o atributo "target" de todos os

outros, para assim, podermos saber se ele está avaliando corretamente, mas, então vemos a variável "X" recebendo todos os 13 atributos de treinamento e a variável "Y" recebe o atributo mencionado, segue a figura 9.

```
In [12]: print(Y)

0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

**Figura 9:** separação dos atributos 2

**Fonte:** Autoral, 2022.

Dados de treinamento são dados fornecidos a um algoritmo de aprendizado de máquina para construir um modelo. Esses dados geralmente representam cerca de 70% dos dados, por isso foram separados como mostra a figura 8 e figura 9. Segue a figura 10.

**Dividindo os dados em dados de treinamento e dados de teste**

```
In [13]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)

In [14]: print(X.shape, X_train.shape, X_test.shape)

(303, 13) (242, 13) (61, 13)
```

**Figura 10:** dados teste e dados treinamento

**Fonte:** Autoral, 2022.

Agora precisamos entender como funciona o a modelagem inicial do projeto, pois é aqui que o modelo será dividido para ser treinado, onde:

X_train - Isso inclui todas as suas variáveis independentes, elas serão usadas para treinar o modelo, também como especificamos o test_size = 0.2, esses 80% de seus dados completos será usado para treinar/ajustar o modelo e os 20% restantes serão usados para testar o modelo.

X_test - Esta é a parte restante dos 20% das variáveis independentes dos dados que não serão usados na fase de treinamento e serão usados para fazer previsões para testar a precisão do modelo.

Y_train - Esta é sua variável dependente que precisa ser prevista pelo modelo, isso inclui rótulos de categoria contra suas variáveis independentes, precisamos especificar nossa variável dependente enquanto treinamos/ajustamos o modelo.

Y_test - Esses dados têm rótulos de categoria para os dados de teste, esses rótulos serão usados para testar a precisão entre as categorias reais e previstas.

## 3.5 Algoritmo de Aprendizagem

Os algoritmos de atividades preditivas podem ser classificados como sendo de regressão ou classificação, algoritmos de classificação tentam classificar objetos ou amostras com base nas propriedades observadas do operador, enquanto algoritmos de regressão tentam classificar a relação entre variáveis usadas para prever valores. Segue o esquema (figura 11).

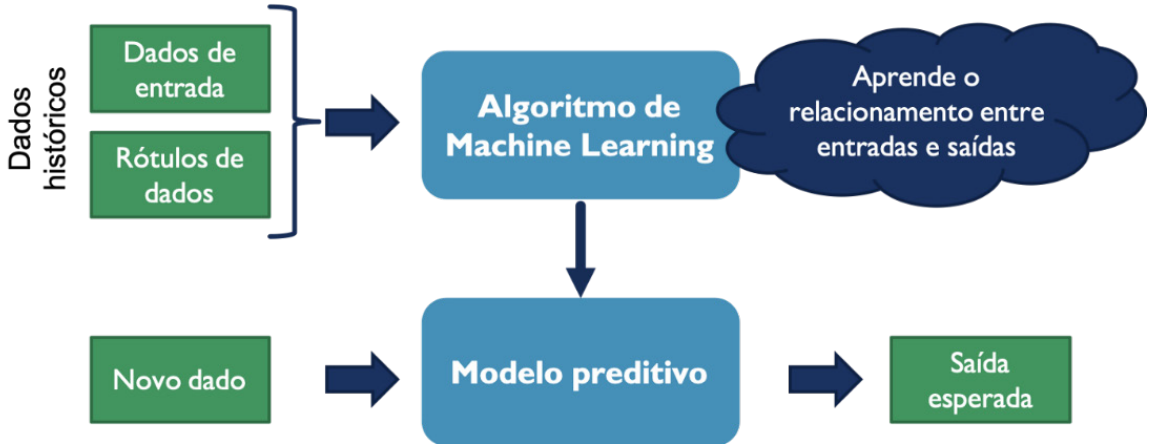



**Figura 11:** esquema do aprendizado

**Fonte:** Tatiana Escovedo, 2022.

Podemos concluir que este é um modelo supervisionado, pois o modelo de entrada é construído através de um conjunto de dados que foi recebido, e são apresentados como pares ordenados (entrada - saída). Dizemos que foram rotulados, porque conhecemos os detalhes de cada entrada e saída com antecedência.

Normalmente, os dados de entrada (rotulados) são divididos em dois conjuntos: um conjunto de treinamento, que é usado para construir o modelo, e um conjunto de teste, que também é chamado de literatura de conjunto de validação.

Só podemos treinar uma máquina usando algoritmos de aprendizado de máquina. Em suma, os algoritmos de aprendizado de máquina são conjuntos organizados de regras, comandos e instruções. Ao combinar algoritmos com dados, eles obtêm um resultado sólido, como mostra a figura 12.

**Regressão Logística**

```
In [15]: model = LogisticRegression()

In [16]: # treinando o modelo LogisticRegression com dados de treinamento
         model.fit(X_train, Y_train)

         C:\Users\ricar\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge
         (status=1):
         STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

         Increase the number of iterations (max_iter) or scale the data as shown in:
             https://scikit-learn.org/stable/modules/preprocessing.html
         Please also refer to the documentation for alternative solver options:
             https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
           n_iter_i = _check_optimize_result(

Out[16]: LogisticRegression()
```

**Figura 12:** algoritmo usado

**Fonte:** Autoral, 2022.

Existem duas classificações principais de algoritmos de aprendizado de máquina:

1. Aprendizado supervisionado: Os algoritmos de aprendizado de máquina supervisionados são aqueles que exigem entrada humana para manipular os dados.

Nesse caso, já existe uma implementação do resultado correto (exemplos de rótulos), ou seja, o algoritmo já recebe determinados dados para realizar determinada operação.

2. Aprendizado não supervisionado: Neste caso, as informações especificadas não são inseridas, portanto, o sistema não possui uma resposta correta. Assim, um algoritmo de aprendizado de máquina não supervisionado tem resultados variáveis e imprevisíveis que geram novos padrões e filtros.

O algoritmo escolhido já citado para o presente projeto, que foi o Algoritmo de Regressão Logística, a figura 13, retrata como ele se apresenta.

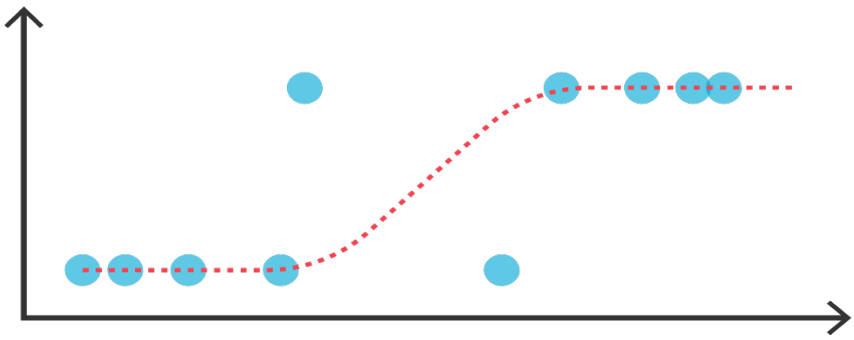


**Figura 13:** curva da regressão logística

**Fonte:** TIBCO Software, 2022.

Este tipo de algoritmo de aprendizado de máquina trabalha com questões estatísticas e probabilísticas e lida com problemas de classificação. Para isso, a regressão logística calcula as variáveis e as classifica da melhor forma. Ao contrário da regressão linear, a regressão logística forma um padrão em S e apresenta valores entre 0 e 1. Na regressão logística a variável dependente tem apenas duas categorias. Em geral, a ocorrência do evento de interesse é codificada como "1" e a ausência como "0". Lembrando que a codificação altera o sinal dos coeficientes e, portanto, sua interpretação substantiva.

A regressão logística difere de outras técnicas de mineração, principalmente pelo fato de sua variável dependente ser categórica, e mesmo quando ela não é dicotômica, é possível torná-la dicotômica, com a finalidade de aplicar esta técnica.

## 4. RESULTADOS E DISCUSSÕES

Como já foi citado anteriormente, o objetivo de um modelo de classificação de dados é fazer uma previsão com base em eventos passados. Para fazer isso, o modelo usa um conjunto de dados com entradas e atributos. Além disso, é necessário conhecer a saída esperada desse conjunto de dados.

Todas essas informações são usadas para treinar um modelo, que é usado para prever os resultados esperados de novos dados futuros. Ao treinar este modelo, foi usado um conjunto de dados (não aquele usado no treinamento) para testar o quão bem o modelo está correto. No entanto, não basta contar os sucessos do seu modelo, se foi bom ou não.

Nesta avaliação, diferentes métricas devem ser utilizadas dependendo do problema sob investigação. Mas antes de introduzir essas métricas, precisamos entender alguns conceitos de classificação binária: as classes que os dados previstos podem assumir.

Se falarmos especificamente sobre validação, então acurácia refere-se a quão próximos os resultados encontrados por meios automatizados ou soluções de inteligência artificial estão da realidade. Então, segue abaixo a figura 14, mostrando os resultados encontrados por meio do modelo criado.

```
In [17]: # Acurácia nos dados de treinamento
         X_train_prediction = model.predict(X_train)
         training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

In [18]: print('A acurácia nos dados de treinamento é de: ', training_data_accuracy)

         A acurácia nos dados de treinamento é de:  0.8512396694214877

In [19]: # Acurácia nos dados de teste
         X_test_prediction = model.predict(X_test)
         test_data_accuracy = accuracy_score(X_test_prediction, Y_test)

In [20]: print('A acurácia nos dados de teste é de: ', test_data_accuracy)

         A acurácia nos dados de teste é de:  0.819672131147541

In [21]: predictions = model.predict(X_test)

In [22]: print(classification_report (Y_test, predictions))
         print(confusion_matrix(Y_test, predictions))

                       precision    recall  f1-score   support

                    0       0.79      0.82      0.81        28
                    1       0.84      0.82      0.83        33

             accuracy                           0.82        61
            macro avg       0.82      0.82      0.82        61
         weighted avg       0.82      0.82      0.82        61

         [[23  5]
          [ 6 27]]
```

**Figura 14:** acurácia do modelo

**Fonte:** Autoral, 2022.

Vemos que a Acurácia do modelo de treinamento está equivalente a 85% e a acurácia do modelo de teste está a 82% arredondado, ou seja, a cada 10 indivíduos avaliados pelo modelo construído, cerca de 8 indivíduos foram diagnosticados corretamente, vale ressaltar que um projeto com acurácia acima de 70% já é considerado bom.

Nesta mesma figura, podemos observar algumas variáveis importantes e seus valores:

- Acurácia indica o desempenho geral do modelo, quantas de todas as classificações o modelo classificou corretamente.
- Precisão: indica as classificações de classe positivas feitas pelo modelo estão corretas.
- Recall indica de todas as situações da categoria positiva, como valor esperado, quantas estão corretas
- F1 score é simplesmente uma maneira de rastrear apenas uma métrica em vez de duas (precisão e recall) em determinadas situações. É a média harmônica dos dois, que está muito mais próxima dos valores mais baixos do que a média aritmética simples. Ou seja, se a pontuação da F1 for baixa, indica que a precisão ou a recuperação são baixas.

Temos também a Matriz de Confusão mostrada no final da Figura 13, uma maneira

simples de apresentar os resultados de um método de classificação de dados é com a chamada matriz de confusão, ela mostra o número de ocorrências que o programa teve de cada uma das quatro categorias.

Para finalizar o projeto apresentado até aqui, foi construído um sistema preditivo, como mostra a figura 15, com objetivo de receber os valores dos atributos de pessoas para serem avaliadas individualmente, como visto anteriormente, a acurácia está a cerca de 80%.

**Construindo um Sistema Preditivo**

In [23]:
```
input_data = (62,0,0,140,268,0,0,160,0,3.6,0,2,2)

# Alterar os dados de entrada para uma matriz numpy
input_data_as_numpy_array= np.asarray(input_data)

# remodelar a matriz numpy como estamos prevendo apenas na instância
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
  print('A pessoa não tem uma doença cardíaca')
else:
  print('A pessoa tem uma doença cardíaca')
```

```
[0]
A pessoa não tem uma doença cardíaca
```

```
C:\Users\ricar\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but LogisticR
egression was fitted with feature names
  warnings.warn(
```

**Figura 15:** sistema preditivo para avaliação

**Fonte:** Autoral, 2022.

O primeiro passo é colocar todos os dados de cada atributo, precisa serem preenchidos todos na seguinte ordem:

1- Age (idade)

2- Sex (sexo)

0: feminino

1: masculino

3- Cp (tipo de dor torácica)

1: angina típica

2: angina atípica

3: dor não anginosa

4: assintomático

4- Trestbps (pressão arterial de repouso)

5- Chol (colesterol sérico em mg/dl)

6- Fbs (glicemia em jejum > 120 mg/dl)

0: falso

1: verdadeiro

7- Restecg (resultados eletrocardiográficos em repouso)

0: normal

1: com anormalidade da onda ST-T (inversões da onda T e/ou elevação ou

depressão do segmento ST > 0,05 mV)

2: mostrando provável ou definitiva hipertrofia ventricular esquerda pelos critérios de Romhilt-Estes

8- Thalach (frequência cardíaca máxima alcançada)

9- Exang (angina induzida por exercício)

0: não

1: sim

10- Oldpeak (depressão ST induzida pelo exercício em relação ao repouso)

11- Slope (a inclinação do pico do exercício segmento ST)

1: ascendente

2: plano

3: downsloping

12- Ca (número de vasos principais (0-3) coloridos por fluorosopia)

13- Thal (3 = normal; 6 = defeito corrigido; 7 = defeito reversível)

O indivíduo usado como teste para o sistema preditivo foi diagnosticado sem doença cardiovascular. Lembrando que o algoritmo avalia de acordo com os dados apresentados a ele, os dados que foram submetidos anteriormente, tiveram finalidade apenas de criar o modelo te treinamento e fazer testes.

## 5. CONCLUSÃO

As doenças cardiovasculares são as principais causas de mortes no mundo. Neste trabalho, o modelo de aprendizado de máquina para classificação multe classes de diagnósticos de doenças cardiovasculares foi proposto para apoiar a tomada de decisão, promover a pesquisa de materiais e fornecer direcionamento para diagnósticos de reconhecimento e, assim, ajudar na indicação de tratamento.

Foi escolhido um modelo de algoritmo de *Machine Learning* para construir o modelo de aprendizado de máquina de classificação, que é: Regressão Logística; um método usado para prever variáveis de resultado categóricas a partir de variáveis preditoras categóricas ou contínuas. As variáveis de resultado são muitas vezes referidas como dependentes e os preditores como independentes. A previsão de uma categoria específica pode ser chamada de classificação.

Em trabalhos futuros, pretende-se avaliar diferentes estratégias de balanceamento, explorar a otimização de mais hiper parâmetros para os modelos e realizar um pré-processamento que defina melhor as classes de amostra com base na probabilidade de diagnósticos implícitos nos dados.

Também pretende-se criar outros modelos de predição e classificação baseado em novos tipos de algoritmos de machine Learning, tais como Regressão Linear, Arvores de Classificação e regressão, Naive Bayes, KNN, Random Forest, etc. Visando buscar a melhor acurácia possível.

## Referências

A.M. AGUILERA, M. ESCABIAS, M.J. VALDERRAMA, Using principal components for estimating logistic regression with high-dimensional multicollinear data. **Computational Statistics & Data Analysis**, 55 (2006), 1905-1924.

AL-ABSI, H. R. H., REFAEE, M. A., REHMAN, A. U., ISLAM, M. T., BELHAOUARI, S. B., and ALAM, T. (2021). Risk factors and comorbidities associated to cardiovascular diseasein qatar: A machine learning based case-control study. **IEEE Access**, 9:29929–29941

MCKINNEY, Wes. **Python for Data Analysis:** tratamento de dados com pandas, numpy e ipython. O'Reilly, 2018. ISBN: 9781491957660.

CHOLLET, F. (2017). **Deep Learning with Python**. Manning.

CUI, S., LI, C., CHEN, Z., WANG, J., AND YUAN, J. (2020). Research on risk predictionof dyslipidemia in steel workers based on recurrent neural network and lstm neuralnetwork. **IEEE Access**, 8:34153–34161

LIAN, Suyun; GUAN, Lixin; PENG, Zhongzheng; ZENG, Gui; LI, Mengshan; XU, Yin. Retrieval of leaf chlorophyll content in Gannan navel orange based on fusing hyperspectral vegetation indices using machine learning algorithms. **Rural Science**, [S.L.], v. 53, n. 3, p. 6-10, out. 2022. FapUNIFESP (SciELO)

MOHAN, S., THIRUMALAI, C., and SRIVASTAVA, G. (2019). Effective heart disease predictionusing hybrid machine learning techniques. **IEEE Access**, 7:81542–81554.

RASCHKA, S. and MIRJALILI, V. (2019). **Python Machine Learning**. Packt Publishing, Bir-mingham, UK, 3 edition

GHOSH, P., AZAM, S., JONKMAN, M., KARIM, A., SHAMRAT, F. M. J. M., IGNATIOUS, E., SHUL-TANA, S., BEERAVOLU, A. R., and DE BOER, F. (2021). Efficient prediction of cardiovas-cular disease using machine learning algorithms with relief and lasso feature selectiontechniques. **IEEE Access**, 9:19304–19326.

RAVI, D., WONG, C., DELIGIANNI, F., BERTHELOT, M., ANDREU-PEREZ, J., LO, B., and YANG,G.-Z. (2017). Deep learning for health informatics. **IEEE Journal of Biomedical and Health Informatics**, 21(1):4–21.

SMIGIEL, S., PAŁCZY´NSKI, K., and LEDZI´NSKI, D. (2021). Ecg signal classification using deeplearning techniques based on the ptb-xl dataset. **Entropy**, 23(9).

STRODTHOFF, N., WAGNER, P., SCHAEFFTER, T., and SAMEK, W. (2020). **Deep learning for ECGanalysis:** Benchmarks and insights from PTB-XL. CoRR, abs/2004.13701.

WAGNER, P., STRODTHOFF, N., BOUSSELJOT, R., KREISELER, D., LUNZE, F. I., SAMEK, W., ANDSCHAEFFTER, T. (2020). Ptb-xl, a large publicly available electrocardiography dataset. **Scientific Data**, 7.

WHO, W. H. O. (2021). **Cardiovascular diseases (cvds)**. Disponivel em: https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds). Acesso em: 29 jun. 2022

WORLD HEALTH ORGANIZATION. (2018). **Technical package for cardiovascular disease management in primary health care:** healthy-lifestyle counselling. World Health Organization.

# 9

# ANÁLISE DE UMA APLICAÇÃO DESENVOLVIDA COM POWERAPPS PARA AUTOMAÇÃO DE RELATÓRIOS DO SAP UTILIZANDO FRAMEWORK SELENIUM COM PYTHON NO ESCRITÓRIO DE UMA ORGANIZAÇÃO

*ANALYSIS OF AN APPLICATION DEVELOPED WITH POWERAPPS FOR AUTOMATION OF SAP REPORTS USING SELENIUM FRAMEWORK WITH PYTHON IN THE OFFICE OF AN ORGANIZATION*

Juliana Serra Portela[1]

Edilson Carlos Silva Lima[2]

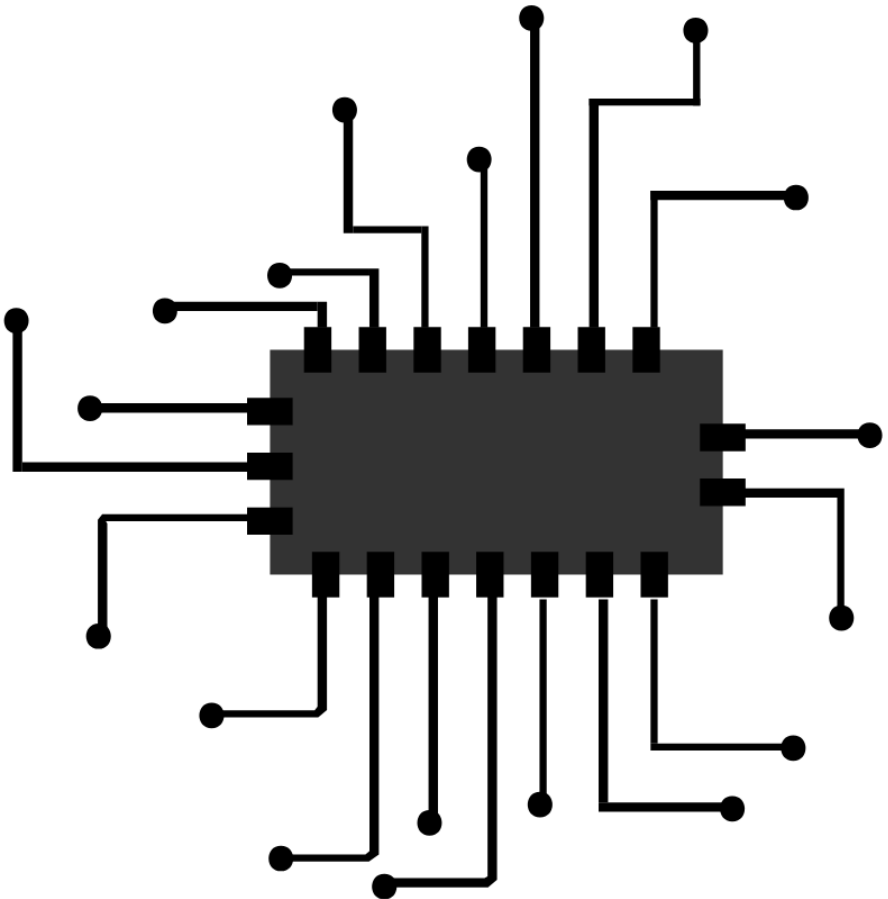
Jonathan de Araújo Queiros[3]

1Engenharia de Computação – Universidade Ceuma (UniCEUMA) – São Luís – MA – Brasil

2Engenharia de Computação – Universidade Ceuma (UniCEUMA) – São Luís – MA – Brasil

3Engenharia de Computação – Universidade Ceuma (UniCEUMA) – São Luís – MA – Brasil

{PORTELA, Juliana Serra, juliana-portela@outlook.com; LIMA, Edilson Carlos Silva, edilsonlima3@gmail.com; QUEIROS, Jonathan de Araújo, queirozjth@gmail.com}



**d.o.i.:**

## Resumo

Este trabalho teve por objetivo apresentar uma proposta de melhoria de gerenciamento da informação interna visando a otimização de relatórios; este artigo contém resultados de um estudo de caso feito em um escritório de uma organização em São Luís do Maranhão. O processo envolve etapas de determinação da necessidade do setor de administração que realiza relatórios sobre as requisições de compras realizadas no sistema SAP. Foi realizado um diagrama de caso de uso para determinar os atores e os casos que fariam parte do aplicativo que foi desenvolvido, além de fazer um fluxo de eventos. Por fim, foi feito um levantamento exploratório-descritivo, classificado como um estudo survey com os colaboradores da organização para saber se a solução desenvolvida atendeu as necessidades destes usuários. A solução desenvolvida utilizou as tecnologias: PowerApps, juntamente com as Listas do Sharepoint que é alimentada por uma rotina utilizando o framework Selenium Webdriver, que de 1h em 1h, que roda no SAP e baixa os dados brutos. Constatou-se que o sistema SAP é fundamental para o processo de compra, mas é preciso ferramentas auxiliares para desenvolver relatórios mais precisos.


**Palavras-chave:** SAP, SAP GUI for HTML, Selenium Webdriver, PowerApps, Automação Selenium.


## Abstract

This paper aimed to present a proposal to improve internal information management aiming at the optimization of reports; this paper contains results from a case study done in an office of an organization in São Luís do Maranhão. The process involves steps of determining the needs of the administration sector that performs reports on the purchase requisitions performed in the SAP system. A use case diagram was made to determine the actors and cases that would be part of the application that was developed, in addition to making an event flow. Finally, an exploratory-descriptive survey was done, classified as a survey study with the organization's employees to find out if the solution developed met the needs of these users. The solution developed used the technologies: PowerApps, along with Sharepoint Lists that is fed by a routine using the Selenium Webdriver framework, which every 1h runs in SAP and downloads the raw data. It was found that the SAP system is fundamental to the purchasing process, but auxiliary tools are needed to develop more accurate reports.


**Keywords:** SAP, SAP GUI for HTML, Selenium Webdriver, PowerApps, Selenium Automation.

# 1. INTRODUÇÃO

A empresa de terceiro setor para qual foi feito o trabalho pretendia otimizar as elaborações dos seus relatórios referentes aos processos de compras. O SAP já fornece dados sobre a compra que pode ser baixado e analisado, porém o que ele fornece não é suficiente para ter o controle que o setor de administração pretendia.

As informações que o SAP fornece sobre a compra são uma breve descrição sobre o que vai ser comprado, o valor do produto, nome do solicitante, se aquela compra é referente a algum projeto ou se é de algum setor de apoio. Porém também é preciso saber de que cidade aquela compra foi feita, qual o tipo de serviço será empregado naquela compra. Pois para cada tipo de serviço e localidade é designado um colaborador diferente para que ele prossiga com o processo de compra.

O presente trabalho demonstra o desenvolvimento de uma aplicação na plataforma *PowerApps* integrada com uma automação feita em Selenium retirando dados diretamente do SAP. Esta solução foi pensada devido a uma necessidade de ter as informações de forma mais centralizadas. Os relatórios serão de grande importância para ver o quanto foi gasto durante um ano, qual área gastou mais, onde pode ser economizado e ter previsões de quanto irá gastar nos próximos anos.

No capítulo 2 deste artigo está a fundamentação teórica, foi abordado assuntos como: SAP, sua arquitetura e o modelo de implementação SAP GUI for HTML. No capítulo 3 o estudo de caso, com o processo de compras e o fluxo dos aplicativos e como eles funcionam, capítulo 4 os resultados e discussões e por fim no capítulo 5 a conclusão e trabalhos futuros da pesquisa.

# 2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será demonstrada toda a fundamentação teórica que serviu como base de estudo e auxiliou no entendimento e no desenvolvimento deste trabalho assim como na construção da aplicação proposta. Começamos conceituando o sistema SAP e como ele funciona no tópico 2.1. Seguimos falando sobre a programação do SAP que tem uma linguagem de programação própria chamada ABAP que se encontra no tópico 2.2 deste artigo. Logo em seguida os *frameworks* que foram utilizados para construir a aplicação como o *Selenium*, que se encontra no tópico 2.3 deste artigo. Por fim ao *PowerApps* que foi a plataforma utilizada para desenvolver a aplicação resultante.

## 2.1 Sistema SAP

A empresa SAP destaca-se como a principal produtora mundial de software para gestão de processos de negócios, desenvolvendo soluções que facilitam o processamento eficaz de dados e fluxo de informações entre as organizações (SAP DOCUMENTATION, 2022).

Essa plataforma é desenvolvida em uma linguagem de programação de alto nível criada pela própria SAP chamada ABAP, que tem sua sintaxe similar à do COBOL. A ABAP é usada para a criação de aplicativos de negócios dentro do ambiente SAP.

O SAP, é um sistema cliente-servidor com uma arquitetura técnica de três camadas (Three Tier Architecture), esses modelos derivam do modelo "n" camadas e caracterizam-se assim por ter sido retirada a camada, *application server* ou de negócio do lado

do cliente, o desenvolvimento é mais demorado, mas em compensação o retorno é muito mais rápido e o controlo do crescimento do sistema é maior (FARIA, 2020, p.34).

A arquitetura SAP possui três camadas: apresentação, aplicação e a camada dos servidores. A camada de apresentação que consiste em qualquer dispositivo de entrada que possa ser usados para controlar o sistema SAP. Pode ser um navegador WEB, dispositivo móvel etc. Todo processamento central ocorre no servidor de aplicação que não é apenas um sistema em si, mas pode ser várias instâncias do sistema de processamento. O servidor interage com a camada do banco de dados, que geralmente são armazenados em servidores separados. Principalmente por razões de eficiência, mas também por razões de segurança. A comunicação ocorre entre cada camada do sistema, da camada de apresentação ao banco de dados, e então é feito o backup da cadeia (KOCH, 2018).

A SAP desenvolve soluções de softwares que são utilizadas de pequenas empresas a multinacionais. A plataforma coleta e processa os dados para que os funcionários consigam acompanhar o processo do seu projeto em todas as etapas. As soluções SAP podem ser instaladas "on premise" na localização do usuário que se chama SAP GUI for Windows ou utilizadas a partir da nuvem conhecido como SAP GUI for HTML (SAP DOCUMENTATION, 2022).

O SAP GUI é a camada do SAP que interage com o usuário. Este sistema fornece uma API (*Application Programming Interface*) para a criação de scripts utilizando-se da camada SAP GUI que pode ser configurada para a criação de testes personalizados. O SAP GUI emula telas das transações SAP dinamicamente (SARMENTO; KRONBAUER; CAMPOS, 2020).

### 2.1.1 Modelo de Implementação SAP GUI for HTML

O SAP GUI for HTML emula dinamicamente telas de transações SAP através do navegador Web com uma interface gráfica semelhante à usada na SAP GUI for Windows (FARIA, 2020).

Para utilizar o SAP em um navegador basta que seja instalado o SAP GUI para software de servidor HTML. Assim, o SAP conseguirá executar as transações. Para cada tela do sistema SAP, o SAP GUI for HTML gera dinamicamente uma página HTML semelhante à do SAP GUI for Windows.

### 2.1.2 Requisitos para ter o SAP GUI for HTML rodando no sistema

Para utilizar o SAP GUI for HTML os requisitos são:

- Do lado do cliente: não são necessários componentes adicionais. O computador só precisará acessar a internet e ter um navegador Web para isso.
- Do lado do servidor: é preciso instalar o *Internet Transaction Server*, pois é a interface entre a internet e o sistema SAP.
- Para executar o *Internet Transaction Server*, o sistema deve satisfazer os seguintes requisitos: Sistema Operacional como o Windows SERVER e um servidor WEB. A figura 2 mostra os requisitos para ter o SAP GUI for HTML rodando no sistema.

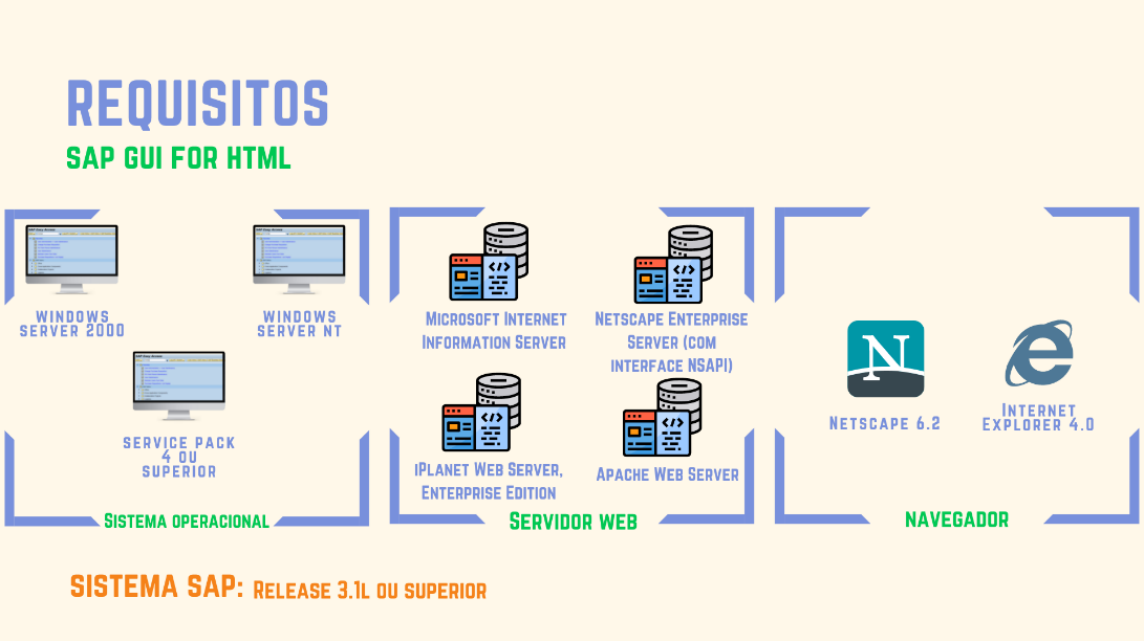



**Figura 1:** Requisitos SAP GUI for HTML

**Fonte:** Autoral, 2022.

Os recursos de máquina para ter o SAP GUI for HTML rodando do lado do cliente são mínimos. Basta apenas um computador com um navegador, mesmo que simples, para emular as telas do sistema SAP dinamicamente.

## 2.1.3 Arquitetura SAP GUI for HTML

O SAP GUI for HTML é impulsionado pelo *Internet Transaction Server* que se comunica com os navegadores Web por ciclos de solicitação/resposta. O esquema a seguir mostra essa comunicação:

A figura 2, mostra a arquitetura do SAP GUI for HTML.

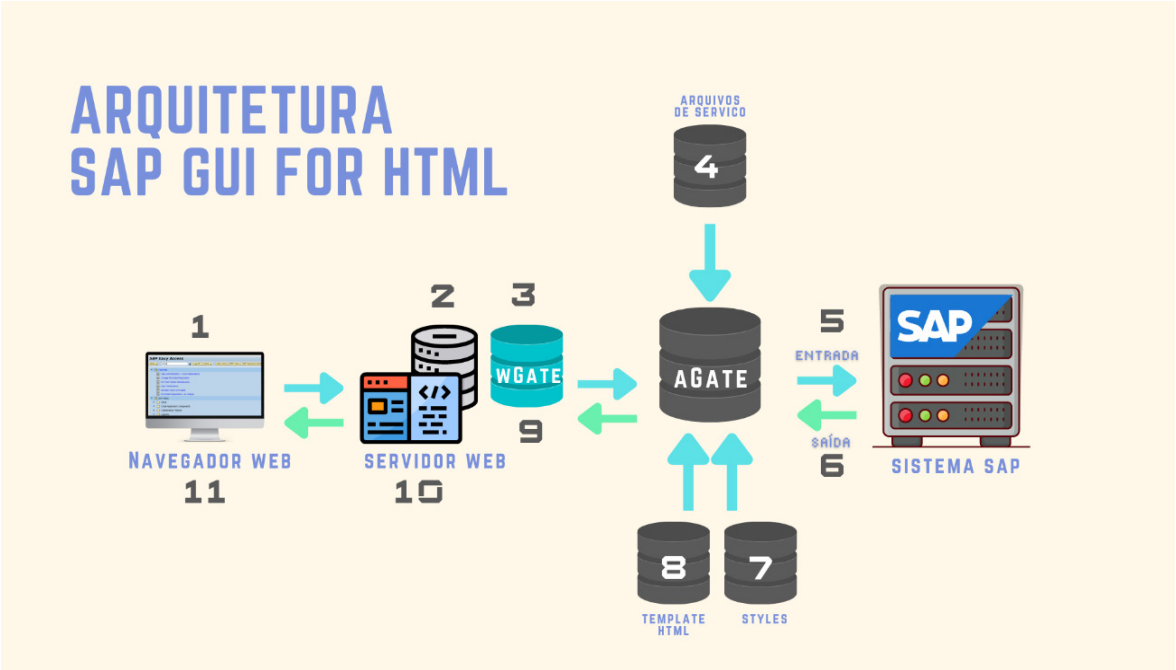



**Figura 2:** Arquitetura SAP GUI for HTML

**Fonte:** Autoral, 2022.

Quando um usuário inicia uma interação com o sistema SAP, o ciclo de solicitação/resposta compreende nos seguintes passos (Tabela 1):

**Tabela 01:** Arquitetura SAP GUI for HTML

| |
|---|
| 1. O navegador Web faz uma solicitação para o Servidor Web |
| 2. O Servidor Web carrega o WGate (Gateway Web), que é a extensão do servidor Web que vincula o ITS ao Servidor Web |
| 3. O WGate envia os dados de solicitação através de uma conexão TCP/IP para o AGate (gateway de Aplicativo), que é o componente de processamento principal do ITS |
| 4. O AGate carrega o arquivo de serviço apropriado e usa as informações armazenadas lá para estabelecer uma conexão com o sistema SAP |
| 5. O Sistema SAP responde enviando a tela de logon de volta para o AGate |
| 6. O AGate usa estilos padrão ou um modelo HTML do cliente para gerar uma página HTML. Um estilo é um conjunto configurável de regras que definem como cada elemento de interface do usuário encontrado em uma tela de transação SAP é mapeado para HTML |
| 7. O AGate envia a página HTML gerada de volta para o WGate |
| 8. O WGate passa a página HTML para o Servidor Web |
| 9. O Servidor Web passa a página para o Navegador Web do usuário, que exibe o resultado |

Autor: SAP DOCUMENTATION, 2022.

O *AGate e WGate* que respectivamente são: *Gateway* de Aplicativo e *Gateway Web*. Cada um tem o seu papel na imagem mostrada. O Gateway de Aplicativo é um balanceador de carga do tráfego da Web que permite que você gerencie o tráfego para os seus aplicativos Web. O Gateway de Aplicativo oferece suporte a SSL / TSL no gateway, o que significa que o tráfego geralmente é enviado sem criptografia para o servidor interno. (“Habilitar o TLS de ponta a ponta no Gateway de Aplicativo Azure”) Esse recurso permite que os servidores Web fiquem livre da sobrecarga da criptografia e descriptografia dispendiosa. Em certos momentos, a comunicação não criptografada com os servidores é uma escolha inconveniente. O aplicativo em questão pode ter requisitos de segurança que impossibilitem essa comunicação sem a criptografia. Para tais aplicações o Gateway de Aplicativo dá suporte à criptografia SSL/TSL de ponta a ponta (AZURE DOCUMENTATION, 2022).

WGate ou Web Gateway, é uma solução de segurança que controla o acesso à rede de uma empresa, controla o acesso à internet etc. Protege os dados da empresa e aplica políticas de segurança. Filtram conteúdo inseguro do tráfego web para deter ameaças cibernéticas (CLOUDFIRE DOCUMENTATION, 2022).

Para manter o contexto do usuário no sistema SAP, o *AGate* (Gateway de Aplicativo) mantém a conexão com o sistema SAP durante toda a sessão SAP GUI para HTML, mas o link entre o navegador da Web e o servidor da Web é fechado após cada ciclo de solicitação/resposta. Para cada solicitação subsequente após a sessão ter sido estabelecida, o navegador da Web usa cookies para enviar um identificador de sessão com a solicitação HTTP que identifica exclusivamente a conexão ao sistema SAP.

Em cada etapa de diálogo subsequente no navegador da Web, o usuário altera um conjunto de campos de tela. O navegador da Web envia esses campos para o servidor web e Wgate, que os passa para o AGate. Graças ao identificador de sessão exclusivo enviado nos dados de solicitação, o AGate encontra a conexão com o sistema SAP e envia os

campos de tela alterados para o sistema SAP, juntamente com um código. O sistema SAP retorna uma tela SAP alterada que é convertida em HTML usando os estilos padrão ou o próprio modelo HTML de um cliente.

## 2.2 ABAP

ABAP[1] é uma linguagem orientada a objetos desenvolvida pela SAP para o desenvolvimento de aplicativos no ambiente SAP. Essa linguagem suporta o modelo de programação orientado a objetos baseados em classes e interfaces além de também suportar um modelo procedural baseados em módulos de função e sub-rotinas.

Uma linguagem de programação orientada a objetos tem características que a difere de uma linguagem estruturada. Uma delas é a classe. A classe define as propriedades e atributos que um objeto terá. A definição de uma classe também define o seu comportamento, ou seja, quais funcionalidades podem ser aplicadas a objetos de classes. Essas duas características podem ser chamadas de métodos e atributos de uma classe (RICARTE, 2001).

Uma classe abstrata, por exemplo, não pode ser instanciada, ela apenas pode ser herdada e não conseguimos criar objetos a partir dela. Segundo Douglas Rocha Mendes (2009), A herança está fundamentada na definição de uma classe com base em outra, ou seja, uma classe é criada a partir de outra classe. O polimorfismo é quando dois objetos de duas classes diferentes possuem um mesmo método, mas implementado de forma diferente ("polimorfismo" vem do grego, *poli* = muitas, *morfos* = forma). E por fim o encapsulamento que consiste em transformar os métodos e atributos privados. Como o próprio nome já sugere, esses métodos e atributos ficam protegidos para que eles não fiquem visíveis fora da classe em que se encontram (RICARTE, 2001). A abstração, a herança, o polimorfismo e o encapsulamento, são os pilares da orientação a objeto.

ABAP é uma linguagem de programação orientada a objetos e no servidor de aplicativos, o pré-requisito para o uso dessa linguagem é a instalação de um Application Server ABAP. Com o cliente-servidor contendo três camadas: apresentação, aplicação e banco de dados (SAP DOCUMENTATION, 2022).

## 2.3 Automatização com Selenium Webdriver

*Selenium* é um framework que utiliza variadas ferramentas e bibliotecas que permitem e suportam a automação de navegadores da Web. Selenium Webdriver permite controlar navegadores Web usando diferentes linguagens de programação: Java, Python, C#, Klotin, Ruby e Javascript. Em testes end-to-end, as chamadas das APIs do Selenium Webdriver são tipicamente incorporadas em casos de teste usando uma estrutura de teste de unidade (SELENIUM DOCUMENTATION, 2022).

O Webdriver manipula o navegador nativamente como um usuário faria. Todas as instruções usadas em qualquer linguagem de programação são feitas pelo Webdriver como se fosse um passo que o usuário estivesse dando dentro do navegador. O Selenium Webdriver é o componente mais usado no conjunto de ferramentas do Selenium (GARCÍA; KLOOS; ALARIO-HOYOS; MUNOZ-ORGANERO, 2022).

1 ABAP: Advanced Business Application Programming

### 2.3.1 Arquitetura do Selenium WebDriver

*Selenium* é um projeto de código aberto dedicado a fornecer automação do navegador. Foi lançado pela primeira vez por Jason Huggins e Paul Hammant em 2004. A primeira versão do Selenium foi uma biblioteca Javascript que já naquela época já se preocupavam em imitar as ações dos usuários em aplicações Web (BURNS, 2009).

Jason Huggins e Simon Stewart fundiram o projeto Selenium e o projeto Webdriver em um novo projeto chamado Selenium Webdriver. É necessário incluir um arquivo intermediário para controlar um navegador. Esse arquivo é chamado de *"driver"*. No navegador Chrome você utiliza o chromedriver, por exemplo. Ele é um arquivo binário dependente da plataforma que recebe comandos do Selenium Webdriver API e os traduz para alguma linguagem específica do navegador. Inicialmente, a comunicação entre o script do Selenium e os drivers era feita usando mensagens JSON sobre HTTP no chamado Protocol Wire JSON. (RAGHAVENDRA, 2021) Recentemente essa comunicação não acontecerá mais a partir do Protocolo Wire JSON na versão 4 do Selenium, ela será feita a partir do W3C Webdriver (GARCÍA; KLOOS; ALARIO-HOYOS; MUNOZ-ORGANERO, 2022).

O driver do navegador usa servidores HTTP para obter solicitações HTTP. Quando o driver do navegador obtém a URL, ele passa a solicitação ao seu navegador por HTTP e acionará o evento de execução das instruções do Selenium no navegador (SHETH, 2020).

O Webdriver W3C é o protocolo recém-introduzido na versão 4 do Selenium. (GARCÍA; KLOOS; ALARIO-HOYOS; MUNOZ-ORGANERO, 2022) No Selenium 4, o protocolo W3C substitui o protocolo JSON Wire. Isso significa essencialmente que você não precisa mais codificar e decodificar a solicitação da API. O protocolo W3C faz com que os testes se comuniquem diretamente com o navegador da Web. As informações serão transferidas enviando e recebendo solicitações HTTP (SHETH, 2020).

A arquitetura do Selenium Webdriver utilizando o W3C Protocol, há uma troca direta de informações entre o cliente e o servidor, sem a necessidade do JSON Wire Protocol. Como o Selenium Webdriver e os navegadores da Web usam o mesmo protocolo, o teste automatizado do Selenium executará testes de forma mais consistente entre diferentes navegadores (STEWART; BURNS, 2020).

## 2.4 Powerapps

O PowerApps fornece uma plataforma para desenvolvimento rápido para criar aplicativos de acordo com a sua necessidade. Você consegue conectar seus dados armazenados na plataforma de dados subjacente ou em várias fontes de dados online e locais como o SharePoint, Microsoft 365, SQL Server entre outros. Essa plataforma se destaca pela característica do Low Code. Com apenas poucas linhas de comando você consegue publicar um aplicativo na plataforma. Isso porque o canvas do PowerApps é totalmente intuitivo e você precisa apenas atribuir uma função a um determinado elemento. A linguagem utilizada é o Power FX (MICROSOFT DOCUMENTATION, 2022).

O *PowerFX* é uma linguagem Low Code fortemente tipada, declarativa e funcional. Ela é baseada nas fórmulas do Excel porque ela vincula objetos com fórmulas declarativas semelhante a planilhas (AZURE DOCUMENTATION, 2022). Além disso, o Power FX oferece uma lógica imperativa quando necessário. As planilhas geralmente não têm botões para enviar alterações no banco de dados, já nos aplicativos essa função existirá.

Essa linguagem de Low Code é concisa, porém poderosa. Toda lógica de um botão, por exemplo, pode ser escrita em apenas uma linha. O objetivo é que o usuário do Excel

consiga usar o PowerApps sem nenhum esforço a mais.

## 3. METODOLOGIA

A metodologia adotada se refere ao tipo quantitativa exploratória de natureza aplicada. O instrumento de coleta de dados utilizado foi um formulário online da plataforma Google Forms que foi enviado para os colaboradores que utilizam o SAP para solicitar compras. O questionário contou com três perguntas discursivas acerca do sistema SAP, da aplicação desenvolvida e quais melhorias esta aplicação poderia ter. Nesse passo, foram analisadas todas as respostas de cada questão. Respostas com a mesma linha de raciocínio foram colocadas nas mesmas categorias e assim foi possível fazer os gráficos que se encontram na sessão Resultados e Discussões deste presente trabalho. Dessa forma se pretende analisar o comportamento do solicitante de compras para otimizar esses processos.

## 4. ESTUDO DE CASO

A Organização para qual foi feito esse trabalho é um empresa do terceiro setor que está espalhada por mais de 70 países e que aqui no Brasil categoriza-se como médio porte por ter em média 200 funcionários pelo país. A gerente do setor de administração percebeu que, para ter relatórios mais detalhados, e esses detalhes são de que cidade está vindo o pedido, saber se o pedido tem um plano de compras, qual escritório está demandando mais um determinado serviço, precisaria das informações mais centralizadas. Essas são algumas das informações que se tinha que buscar em diversas fontes diferentes e às vezes com dados duplicados. Isso pode dar uma certa inconsistência no resultado do relatório.

Esta empresa tem um grande fluxo de transações no SAP diariamente, principalmente quando os projetos estão no início e precisa ter um planejamento do que precisa ser comprado. Além das compras que às vezes não estão programadas. Todo esse fluxo de dados o SAP armazena. Para tirar um melhor proveito para a análise desses dados, pensou-se em uma solução que resultou em um aplicativo que irá alimentar bases de dados, que futuramente, poderá ser usada para fazer relatórios mais precisos pelo setor de administração da empresa. O aplicativo "Solicitante de Compra" traz informações do SAP que o solicitante já preencheu e fornece novos campos para o solicitante preencher fazendo com que todas as informações que o setor de administração precise para os seus relatórios fique centralizado.

Essa solução foi pensada justamente para facilitar o gerenciamento de cada pedido. Utilizando a ferramenta do *PowerApps*, que é bastante amigável para o usuário final, assim, o solicitante consegue acompanhar os status da sua compra.

Para isso foi pensado em uma rotina que roda diariamente no intervalo de 1h em 1h no SAP e segue essas etapas (tabela 2).

**Tabela 2:** Automação dentro do SAP

| Etapas | Passos da Automação |
|---|---|
| Etapa 01 | Faz o login no sistema SAP |
| Etapa 02 | Acessa a transação: me5a, também chamada Purchase Requisitions: List Display, que significa Lista de Requisições de Compras. Nessa transação estará todas as transações feitas por todos os usuários |
| Etapa 03 | Acessa as requisições do dia em duas tabelas:<br>a. Basic Lists<br>b. Sources of Suply |
| Etapa 04 | Baixa as duas tabelas |
| Etapa 05 | Consolida essas tabelas em apenas uma |
| Etapa 06 | Salva em uma base de dados no SharePoint |

**Fonte:** Autoral, 2022.

Essa rotina, mostrada, fica constantemente alimentando a base de dados no SharePoint para que possamos manipulá-la extraindo relatórios ou usando para criar aplicativos corporativos no PowerApps.

Após a construção do Diagrama de Caso de Uso, ficou simples desenvolver o aplicativo na plataforma do PowerApps, usando as Listas do Sharepoint; local onde a automação está salvando os dados sobre as requisições.

Juntamente com o diagrama de também foi feito um Fluxo de Eventos que consiste em um fluxo principal e um fluxo alternativo:

**Fluxo principal:**

- Após o ciclo da automação a RBS aparece no aplicativo do solicitante
- Solicitante verifica se existe plano de compras
- Se houver plano de compras; solicitante seleciona o plano de compras referente a RBS
- Se não houver plano de compras, o solicitante insere os dados necessários para adicionar um plano de compras
- Solicitante continua para a tela onde ele adicionará o tipo de serviço, a localidade, departamento e a descrição do pedido feito (RBS)
- Salvar as informações inseridas
- Aparecerá uma mensagem perguntando se a data limite da entrega satisfaz
- Satisfazendo: solicitante clica em SIM, finalizando o caso de uso
- Não satisfazendo: solicitante clica em NÃO, escolhe uma nova data (que será analisada pelo coordenador do setor administrativo da organização) e finalmente salva as alterações feitas, finalizando assim, o caso de uso

**Fluxo alternativo:**

- Solicitante cancela as alterações a qualquer momento, pressionando o botão Voltar, exibido na tela
- Não serão realizadas alterações na RBS do solicitante
- Caso de uso é finalizado

O Diagrama de Caso de Uso está descrito na Figura 3 a seguir.

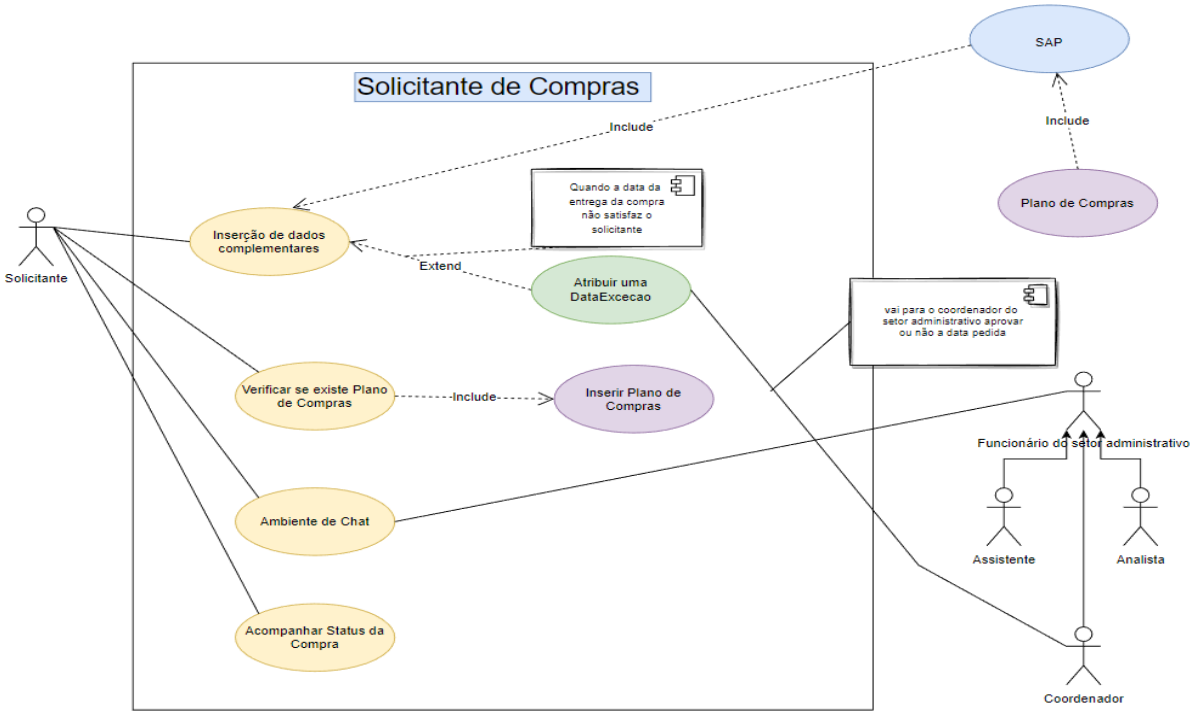



**Figura 3:** Diagrama Caso de Uso

**Fonte:** Autoral, 2022.

As informações citadas anteriormente que o solicitante da compra precisa informar se torna necessária porque algumas informações não são disponibilizadas pelo SAP e que são de grande importância para o andamento da compra. Muitas lacunas são respondidas para o setor administrativo com esse aplicativo como: Saber se as RBS possuem um plano de compras; De onde veio aquele pedido e qual foi o tipo de serviço daquela requisição. Porque para cada tipo de serviço tem uma média de dias para aquela RBS ser entregue.

Como mencionado anteriormente neste artigo, o projeto foi desenvolvido na plataforma PowerApps, que utiliza uma linguagem fortemente tipada chamada *PowerFX* e se assemelha com as fórmulas do Excel. O intuito do *PowerFX* é fazer com que os usuários do Excel não sintam dificuldades na hora de desenvolver os seus aplicativos. A Figura 4 a seguir mostrará um trecho de código que declara uma variável chamada varFiltroEstadoRBS responsável por filtrar as RBS, de um determinado solicitante, com os status: *0. Aguardando envio* e *11. RBS a completar* e logo após navegará para a tela onde o solicitante completará as informações das RBS feitas no SAP.

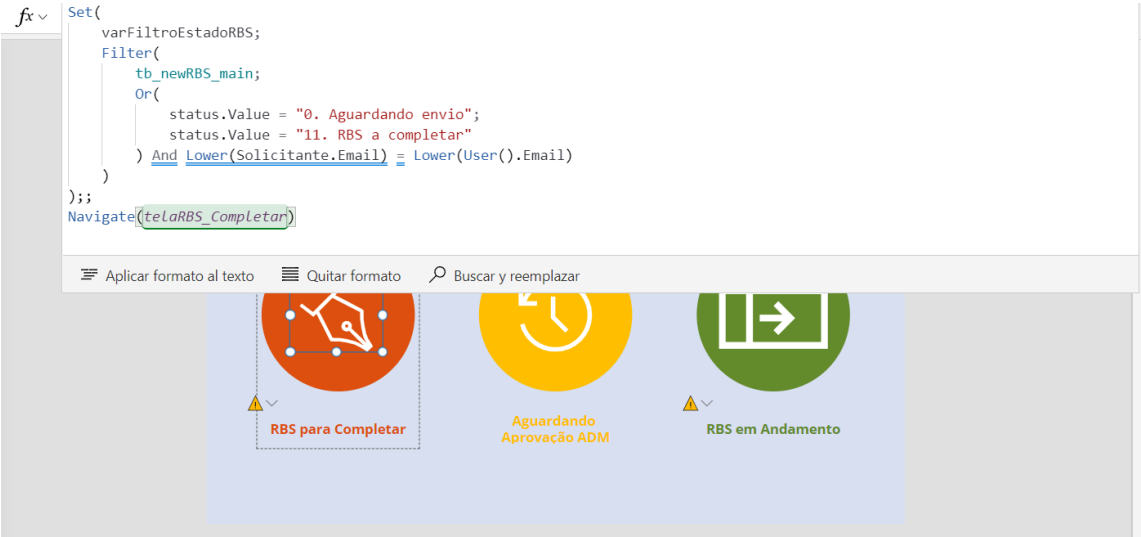



**Figura 4:** Trecho de Código em *PowerFX*

**Fonte:** Autoral

Quando o solicitante clica no botão *RBS para completar,* ele é levado para uma tela onde poderá ter ou não uma lista de RBS que precisam ter as suas informações completadas. Esta tela: *telaRBS_Completar* contém RBS que acabaram de chegar do SAP com o status inicial *0. Aguardando envio;* ou RBS que o setor administrativo da organização já teve contato e percebeu que ainda faltava alguma informação e essa RBS teve que voltar para o solicitante corrigir, alterar ou adionar mais informações. E ela volta com o status *11.RBS a completar.*

## 5. RESULTADOS E DISCUSSÕES

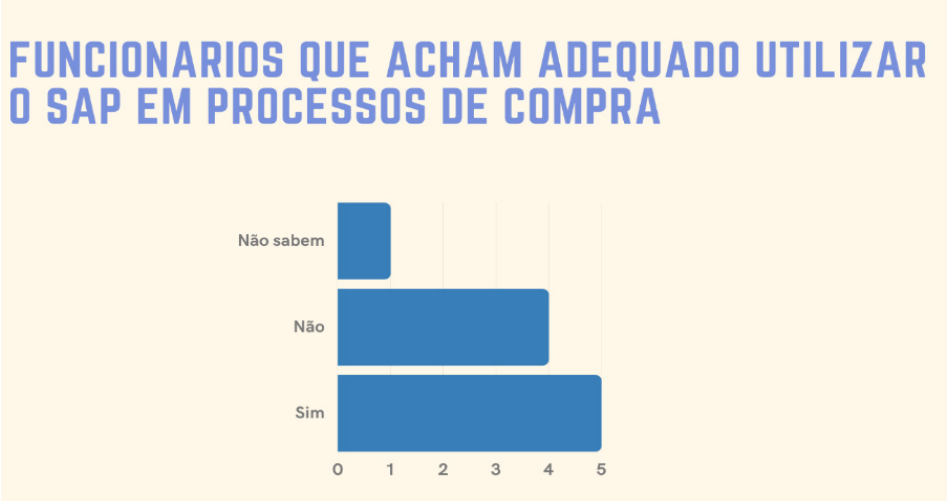
A empresa pesquisada pertence ao terceiro setor, que é formado por organizações sem fins lucrativos. Atualmente ela tem escritórios pelo Brasil, mas a pesquisa foi focada apenas na cidade de São Luís no Maranhão. Que tem uma média de 40 colaboradores, mas desse número, apenas 10 colaboradores utilizam o SAP com a finalidade de solicitar pedidos de compras.

O questionário[2] aplicado dentro da organização contou com três questões discursivas. A primeira questão perguntava para o colaborador se ele considerava o SAP como a ferramenta mais adequada para o processo de compras. A segunda questão perguntou de que forma o novo aplicativo facilitará na comunicação entre o solicitante e o comprador. E a terceira questão perguntou o que poderia ser melhorado na aplicação.

O primeiro item analisado, foi se os funcionários consideram o SAP como a ferramenta mais apropriada para o processo de compras. Foi observado que 5 funcionários, ou seja, 50% dos funcionários entrevistados consideram o SAP como a ferramenta mais apropriada para esse processo; 40% disseram que não é a ferramenta mais apropriada e 10% não soube responder. Um ponto em comum entre as pessoas que deram as respostas opostas sobre o SAP ser a ferramenta mais apropriada, foi a reclamação sobre o sistema ser complexo e confuso ou que a interface dificulta a usabilidade dos usuários. Outro entrevistado que respondeu como o SAP não sendo a ferramenta mais ideal para essa atividade de compras; deu o motivo como sendo um sistema muito rígido e passível a erros. O gráfico a seguir explicita o que foi relatado acima:

2 O questionário pode ser encontrado em: https://forms.gle/2bPiuqGc8YGMc9Mm9

**Gráfico 01:** Funcionários que acham adequado utilizar o SAP em processos de compra




**Fonte:** Autoral, 2022.

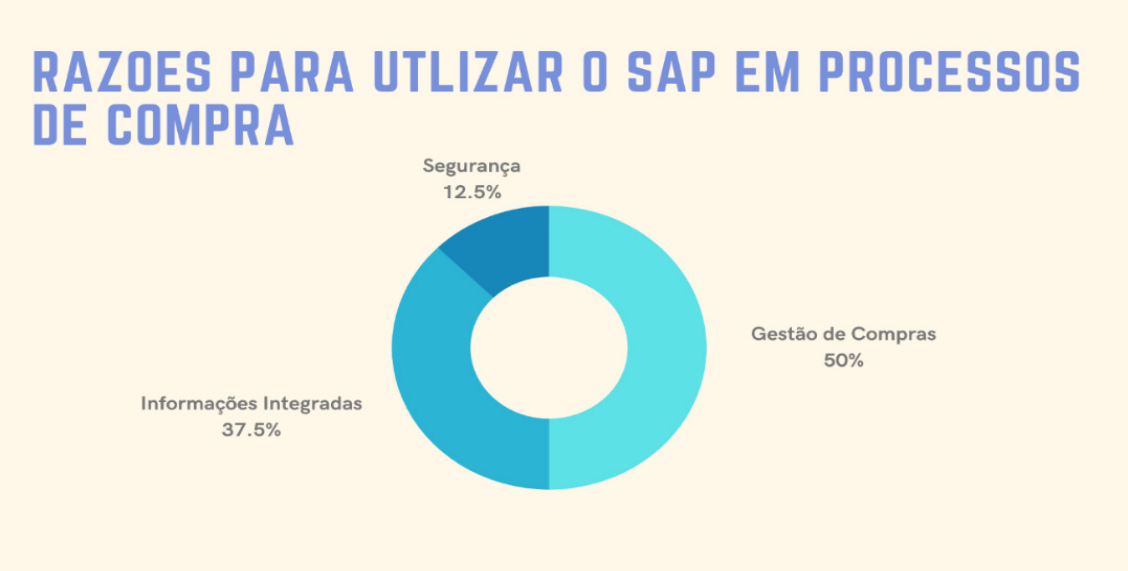
Ainda sobre o Gráfico 1, é importante mostrar a opinião de quem não soube dizer o porquê do SAP ser adequado para o processo de compra. Um dos entrevistados relatou o seguinte: "Por desconhecer outros Sistemas de Compras, não consigo avaliar a adequação dele dentro de um comparativo em relação a outros, contudo pelas experiências de uso, penso que poderia melhorar alguns processos no sistema." Mais uma vez, que a experiência do usuário em relação ao sistema pesa na aceitação do mesmo. Até mesmo as pessoas que julgam o SAP ser o sistema mais adequado para esse processo, ressaltam que a dificuldade está na experiência do usuário, que não é um sistema intuitivo.

Sobre o primeiro item analisado, ainda foram extraídas algumas informações. Diante de análise das perguntas respondidas dos questionários, destrinchei os motivos que os usuários acham adequado a utilização do sistema SAP. Três categorias ressaltaram nessa investigação: a Segurança, com 12,5%; Informações Integradas, com 37,5% e Gestão de Compras, com 50%.

Como ilustra o Gráfico 2, um dos motivos para um usuário aprovar a utilização do SAP é a Segurança que o sistema fornece. Por ser um sistema consolidado no mercado e entender sobre o processo de uma compra, ele se torna uma ferramenta adequada para tal uso. Um dos entrevistados relatou o seguinte: "[...]. O SAP é bem seguro e obedece às etapas necessárias do processo de compra, mas penso que poderia funcionar de forma mais dinâmica e visual". Apesar das constantes reclamações sobre o ambiente de trabalho do SAP não ser intuitivo, essa sentença mostra que ele é um sistema confiável. Um segundo ponto levantado foram as Informações Integradas; um entrevistado respondeu: "Melhoria da gestão de compras e integração total de informações."; o SAP fornece para a pessoa que solicita a compra uma lista das requisições já feitas por esse solicitante, mas o que foi percebido durante essa pesquisa; algumas das pessoas que utilizam esse sistema não consultam o SAP para verificar essas compras já feitas, elas alimentam uma planilha ou um outro documento para fazer esse controle e não necessariamente compartilham com a organização ou com seu superior direto. Serve apenas como um controle pessoal. A organização perde muito em questões de análises de dados com esse comportamento. O terceiro ponto levantado foi a Gestão de Compras; um dos entrevistados relatou: "Por parte do solicitante, o SAP é um sistema importante como ferramenta de comunicação entre o que é solicitado, comprado e executado (pago), no entanto é precário em gerar alguns relatórios de monitoramento e gerenciamento dos processos"; essa é uma sentença que mostra claramente que o solicitante acha que o SAP é adequado para a tarefa de

solicitação de compra e cumpri com as etapas necessárias para essa compra acontecer, porque uma RBS é solicitada, o superior direto desse solicitante aprova e os compradores prosseguem com o processo de compra.

**Gráfico 2:** Razões para utilizar o SAP em processos de compra

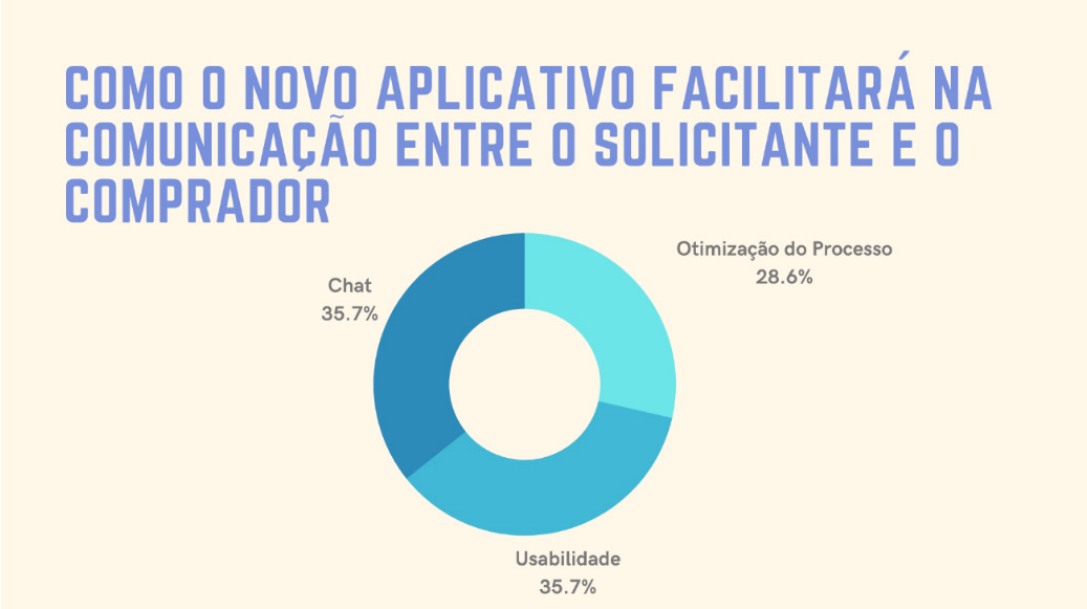



**Fonte:** Autoral, 2022.

O segundo item a ser analisado refere-se ao aplicativo e abordou como essa nova aplicação facilitará na comunicação entre o solicitante e o comprador (setor de administração) no processo de solicitação de alguma compra. Alguns pontos foram levantados, os mais recorrentes dentre as respostas dos entrevistados: 28,6% falaram sobre a Otimização do Processo; 35,7% ressaltaram que ter um ambiente com Chat facilitará consideravelmente e 35,7% falaram sobre a Usabilidade.

O Gráfico 3, elucida as razões pela qual o aplicativo facilitará no processo de compras, um dos entrevistados relatou: "O aplicativo é super intuitivo e faz um link de comunicação entre o SPA, Solicitação e Plano de Compras, facilitando a comunicação entre os setores, outro ponto que favorece a comunicação é o fato de poder ser enviado mensagem para dialogar sobre o processo, ficando esse registro dentro do aplicativo, processo esse que era realizado de diversas formas como por e-mail, Teams e até mesmo pelo WhatsApp, e ter o dialogo acontecendo todo dentro do aplicativo, torna o processo mais transparente com todas as evidências das tomadas de decisão. Outro ponto de melhoria foi poder acompanhar o status do processo sem necessariamente precisar enviar uma mensagem, mas apenas verificando no sistema." Uma questão levantada no processo de desenvolvimento do projeto foi que o aplicativo também conversasse com outros sistemas que já são usados dentro da organização como ressaltou o entrevistado acima; e isso é um fator que otimiza o processo, em que 28,6% constataram que o aplicativo fará isso.

**Gráfico 3:** Como o novo aplicativo facilitará na comunicação entre o solicitante e o comprador




**Fonte:** Autoral, 2022.

Durante a pesquisa a comunicação entre o solicitante e o comprador sempre foi um ponto importante abordado tanto do lado do requisitante quanto do lado do comprador, havia essa preocupação dos dois lados, essa foi uma funcionalidade adicionada ao aplicativo que visou solucionar essa questão; como mostra a opinião de um outro entrevistado: "A junção de informações que sempre buscamos em outras ferramentas, ter a comunicação direta sobre tal solicitação. Me chamou muito atenção positivamente o processo de bate papo sobre a requisição.". Um segundo colaborador também ressaltou: "Ele é uma ferramenta muito prática, pois nos poupará de inserir dados porque já tem a informação lincada a outro sistema que alimenta automaticamente. Isso faz com que se dê mais celeridade aos processo e também exclui a inserção de informação errada, pois já estão armazenados".

Outro ponto tocado foi a Usabilidade com 35,7% das pessoas falando sobre ela: "Status das solicitações poderão ser acompanhado de maneira semelhante a que estamos acostumados a fazer em sites de e-commerce, reduzindo fortemente a incerteza do andamento dos "pedidos"." Um dos colaboradores entrevistados ressaltou sobre essa semelhança a sites de e-commerce onde o requisitante da compra consegue acompanhar os status do seu produto. Em outras respostas também mostra como os colaboradores acharam o aplicativo fácil e bastante intuitivo.

Por fim, foi perguntado aos funcionários o que poderia ser melhorado na aplicação, 30% dos entrevistados disseram que estavam satisfeitos com o aplicativo e que não precisava de nenhuma melhoria e 70% dos entrevistados pediram para adicionar alguma funcionalidade.

Segundo o gráfico 4, a melhoria que mais foi pedida foram as notificações com 40% dos colaboradores afirmando que o aplicativo precisaria adicionar esta funcionalidade. Todos os entrevistados pediram que a notificação viesse por e-mail e alguns especificaram para que notificasse a pessoa que aprova as RBS como mostra essas duas respostas:

- Colaborador A: "Se o app puder avisar por e-mail as pessoas responsáveis por aprovar os processos em sistema de avisos (em uma determinada periodicidade) acho que seria muito útil e daria muita celeridade aos processos de compras".

- Colaborador B: “Para agora, diante da apresentação acredito que ter informação sobre aprovação pela Gerente”.

Percebe-se que esta é uma dificuldade que os solicitantes enfrentam, 10% dos colaboradores pediram que quantificasse as RBS, que fosse gerada uma lista de requisições que aquele solicitante já fez. Outros 10% pediram que fosse possível uma consulta de compras semelhantes, para melhor entendimento deixarei o comentário do entrevistado: “Se não existe, poderia ter algum tipo de consulta de valores praticados em compras semelhantes. Por exemplo, quanto foi pago por camisa no último processo pra eu ter uma base atualizada quando for solicitar camisas pro meu projeto? Ou mesmo o tempo recomendado pra processos parecidos? Enfim, históricos de compras anteriores”. Com essa informação o colaborador deseja que suas próximas compras sejam feitas de forma mais efetivas.

**Gráfico 4:** Melhorias para o aplicativo do solicitante de compras


Fonte: Autoral, 2022.

Durante a apresentação do aplicativo para os colaboradores, foi explicada sobre a rotina que acontecerá no SAP de 1h em 1h e 10% dos entrevistados pediram para reavaliar sobre esse tempo, pediram que, se fosse possível, diminuísse o tempo de cada ciclo da rotina.

## 6. CONCLUSÃO

Com essa pesquisa exploratória e com a análise do estudo de caso, foi possível observar que os colaboradores que utilizam o SAP com a finalidade de requisição de compras sofriam com a falta de informação dos seus pedidos. Os funcionários sempre tinham que entrar em contato com os compradores por ferramentas paralelas como Teams ou até mesmo o WhatsApp para ter um acompanhamento pouco efetivo dos seus pedidos. Com a aplicação, o setor administrativo irá trocar os status e o solicitante poderá acompanhar essa mudança de status sem precisar entrar em contato diretamente com o comprador. E caso precise entrar em contato, cada RBS terá seu ambiente de comunicação.

Por parte dos compradores, a aplicação responderá perguntas fundamentais para que o fluxo dessa compra possa ocorrer. Perguntas essas que o SAP não consegue responder. Agora quando o setor administrativo receber um pedido de compra eles vão saber qual

será o tipo de serviço, de qual localidade esse pedido está vindo e se aquela RBS tem um plano de compras. Estas são algumas das informações que antes o comprador precisaria entrar em contato com o solicitante por aplicativos como Teams ou Whatsapp e que agora basta verificar nos detalhes de cada RBS para saber qual fornecedor eles irão acionar.

Metade dos entrevistados acham que o SAP é a ferramenta mais adequada para o processo de compras e dentre os diversos motivos para isso, destacaram-se três motivos: segurança, gestão de compras e informações integradas. O que muito se reclamou dos dois publicos divergentes foi a dificuldade de usar o sistema.

Para futuros trabalhos, pretende-se adicionar as funcionalidades que foram sugeridas, foram elas: notificações por e-mail, diminuição do tempo da rotina, quantificar as RBS e a consulta de compras semelhantes feitas anteriormente.

## Referências


AZURE DOCUMENTATION. Microsoft Power **Fx: a linguagem de programação de nível baixo e software livre está em versão prévia pública.** [*S. l.*], 2022. Disponível em: https://azure.microsoft.com/pt-br/updates/microsoft-power-fx-the-opensource-lowcode-programming-language-is-in-public-preview/. Acesso em: 14 out. 2022.

AZURE DOCUMENTATION. Recursos do Gateway de Aplicativo do Azure. [*S. l.*], 30 out. 2022. Disponível em: https://learn.microsoft.com/pt-br/azure/application-gateway/features. Acesso em: 14 out. 2022.

BURNS, David et al. **Selenium 2 testing tools beginner's guide**. Packt, 2012.

CLOUDFARE DOCUMENTATION. **O que é um gateway de web seguro (SWG)?.** [*S. l.*], 2022. Disponível em: https://www.cloudflare.com/pt-br/learning/access-management/what-is-a-secure-web-gateway/. Acesso em: 14 out. 2022.

FARIA, Victor. Ferramenta de geração de documentação a partir de um sistema SAP. Orientador: Leonel Domingos Telo Nóbrega. 2020. 91 f. Tese (Mestrado em Engenharia Informática) - Universidade da Madeira, [*S. l.*], 2020.

GARCÍA, Boni; KLOOS, Carlos; ALARIO-HOYOS, Carlos; MUNOZ-ORGANERO, Mario. **Selenium-Jupiter: A JUnit 5 extension for Selenium WebDriver**. The Journal of Systems & Software, [s. l.], n. 12, 2022.

KOCH, Markus. SAP **Architecture Overview.** [*S. l.*], 2018. Disponível em: https://people.redhat.com/mkoch/training/1806-vienna/presentations/08%20-%20SAP%20Architecture.pdf. Acesso em: 11 jul. 2022.

MENDES, Douglas. **Programação Java com Ênfase em Orientação a Objetos.** 1.ed.São Paulo: Novatec Editora LTDA, 2009.

MICROSOFT DOCUMENTATION. **Overview of creating apps in Power Apps.** [*S. l.*], 2022. Disponível em: https://learn.microsoft.com/en-us/power-apps/maker/. Acesso em: 14 out. 2022.

MICROSOFT DOCUMENTATION. **Visão geral do Microsoft Power Fx.** [*S. l.*], 2022. Disponível em: https://learn.microsoft.com/pt-br/power-platform/power-fx/overview. Acesso em: 14 out. 2022.

RAGHAVENDRA, S., 2021. **Introduction to selenium. In: Python Testing with Selenium.** Springer, pp. 1–14.

RICARTE, Ivan. **Programação Orientada a Objetos: Uma abordagem com Java.** n. 118, 2001.

SAP DOCUMENTATION. **ABAP Programming Language - Overview.** [*S. l.*], 2022. Disponível em: https://help.sap.com/doc/abapdocu_752_index_htm/7.52/en-us/abenabap_overview.htm#@@ITOC@@ABENABAP_OVERVIEW_1. Acesso em: 27 jul. 2022.

SAP DOCUMENTATION. **ITS Implementation Models.** [*S. l.*], 2022. Disponível em: https://help.sap.com/doc/saphelp_autoid2007/2007/en-US/48/35d6ca4abf11d18a0f0000e816ae6e/frameset.htm. Acesso em: 19 jul. 2022.

SAP DOCUMENTATION. **SAP GUI for HTML Architecture.** [*S. l.*], 2022. Disponível em: https://help.sap.com/doc/saphelp_autoid2007/2007/en-US/34/b20d38df9e9010e10000009b38f8cf/frameset.htm. Acesso em: 19 jul. 2022.

SAP DOCUMENTATION. **SAP GUI for HTML Requirements.** [*S. l.*], 2022. Disponível em: https://help.sap.com/doc/saphelp_autoid2007/2007/en-US/34/b20d38df9e9010e10000009b38f8cf/frameset.htm. Acesso em: 19 jul. 2022.

SAP DOCUMENTATION. **SAP GUI for HTML.** [*S. l.*], 2022. Disponível em: https://help.sap.com/doc/saphelp_autoid2007/2007/en-US/34/b20d38df9e9010e10000009b38f8cf/frameset.htm. Acesso em: 19 jul. 2022.

SARMENTO, Lívia; KRONBAUER, Arthur; CAMPOS, Jorge. SAP Test Express: Uma Nova Ferramenta de Testes para Aplicações SAP. 2020. 20 f. Artigo (Mestre em Sistemas e Computação) - Companhia de Processamento de Dados do Estado da Bahia, Salvador, 2020.

SELENIUM DOCUMENTATION. **The Selenium Browser Automation Project.** [*S.l.*], 2022. Disponível em: https://www.selenium.dev/documentation/. Acesso em: 25 jul. 2022

SHETH, Himanshu. **Selenium 4 Is Now W3C Compliant: All You Need To Know**. [S. l.], 2020. Disponível em: https://www.lambdatest.com/blog/selenium4-w3c-webdriver-protocol/. Acesso em: 9 set. 2022.

STEWART, Simon; BURNS, David. **WebDriver WC working draft**. 2020.

# 10

## EFICIÊNCIA E SUSTENTABILIDADE EM DATA CENTERS

*EFFICIENCY AND SUSTAINABILITY IN DATA CENTERS*

Daniel Santos Lopes[1]

Edilson Carlos Silva Lima[2]

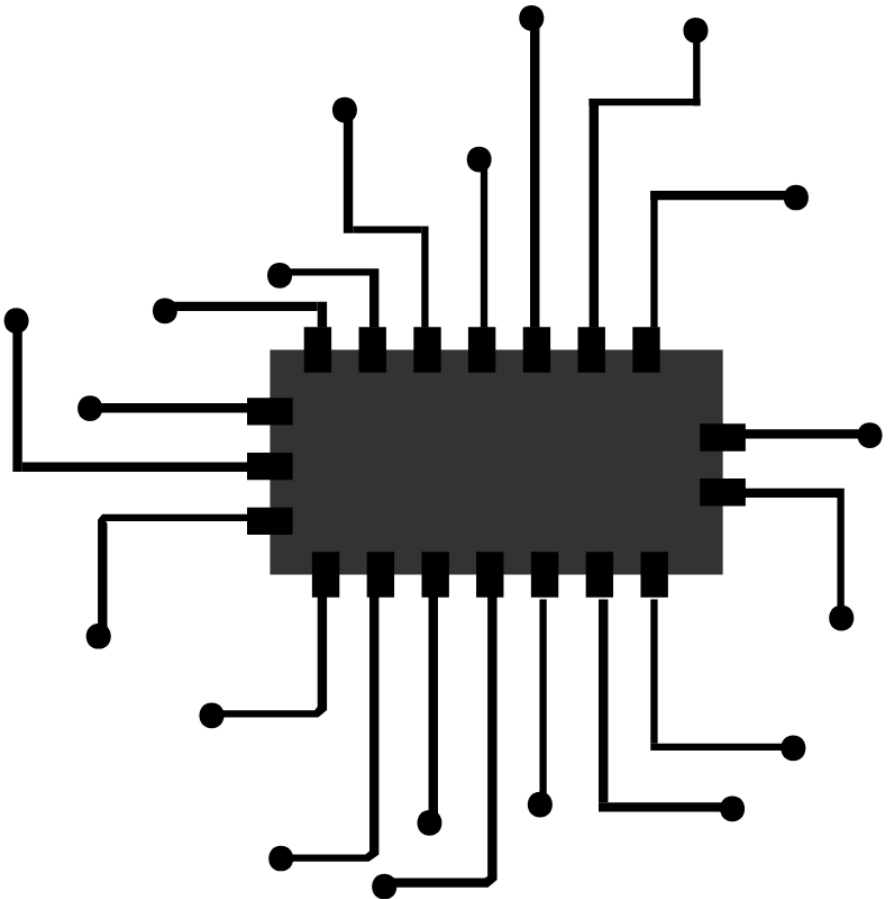
Marcos José dos Passos Sá[3]

1 Engenharia da Computação– Universidade Ceuma (UniCEUMA) – São Luís – MA– Brasil

2 Engenharia da Computação– Universidade Ceuma (UniCEUMA) – São Luís – MA– Brasil

3 Engenharia da Computação– Universidade Ceuma (UniCEUMA) – São Luís – MA– Brasil

{LOPES, Daniel, daniel_lopes-@outlook.com; LIMA, Edilson Carlos Silva, edilsonlima3@gmail.com; SÁ, Marcos José dos Passos, sa.marcos@gmail.com}



**d.o.i.:**

## Resumo

O consumo de energia é intenso dentro de um *Data Center*, uma vez que tal infraestrutura depende de *nobreaks* de grande porte UPS *Uninterruptible Power supply*, de linhas robustas de alta tensão e de geradores a diesel, além de grandes coolers e de outros equipamentos, gerando impactos ambientais durante o seu funcionamento. O objetivo deste artigo é apresentar propostas aos centros de dados independente do seu tamanho, a melhorar a sua eficiência energética, térmica e o controle ambiental afim de reduzirmos a emissão de carbono na atmosfera terrestre, com isto, deixá-los mais ecologicamente corretos e sustentáveis.


**Palavras-chave:** Data Center, Carbono, Sustentável.


## Abstract

The energy consumption is intense within a Data Center, since such infrastructure depends on large nobreaks (UPS) Uninterruptible Power supply, robust lines of high voltage and diesel generators, and large coolers and other equipment, generating environmental impacts during its operation. The objective of this article is to present proposals to data centers independently of their size, to improve their energy efficiency, thermal efficiency and also environmental control in order to reduce the emission of carbon in the earth's atmosphere, thus making them more environmentally friendly and sustainable.


**Keywords***: Data Center, Carbon, Sustainable.*

# 1. INTRODUÇÃO

Ao longo da vida humana alguns hábitos sempre geraram degradação ambiental para a construção de seu *habitat*. com isso, o ritmo acelerado do crescimento da população e sua cisma por inovações tecnológicas tem gerado alta demanda de processamento computacional nos *data centers* para atender vários usuários ao redor do globo, sendo assim, a Tecnologia da Informação (daqui em diante, TI) tornou-se a responsável por uma parte da emissão de dióxido de carbono na atmosfera terrestre ($CO_2$), a TI vem trazendo as estratégias voltadas a sustentabilidade.

Os Centros de dados são as estruturas de TI que mais emitem este gás poluente, porque precisam manter o seu funcionamento constante. De um modo geral, os *Data Centers* oferecem o suporte que um mundo cada vez mais digitalizado exige. Com isso a alta carga de processamento dos equipamentos precisam de uma boa refrigeração, em consequência disto, consomem diretamente uma alta potência elétrica, além de suas fontes de alimentação secundária, construída por grandes geradores movido a diesel, que por sua vez é um grande emissor do gás poluente. Se não houver uma refrigeração eficiente, estes equipamentos terão o ciclo de vida reduzida indo para o lixo. Com isso, aliado a má fiscalização, estes equipamentos podem vir a prejudicar o solo, gerando problemas de saúde a sociedade.

Segundo Hamann (2021), o problema encontrado segundo pesquisas realizadas são:

> Os computadores e servidores consomem alta carga elétrica, entretanto, um sistema de refrigeração não eficiente pode gerar desempenho abaixo do esperado e/ou a perda total dos equipamentos no *Data Hall*. Este problema em conjunto com geradores de alimentação secundaria corroboram causando impactos ambientais, e auxiliando na emissão de gases poluentes.

Em virtude deste cenário, objetiva-se apresentar propostas de intervenções eficientes e sustentáveis de modo a melhorar não apenas, a qualidade de vida da população como também, a qualidade ambiental. As propostas visam a eficiência energética, térmica e controle ambiental, e cada uma destas propostas apresentam subtópicos com o intuito de melhorar a sustentabilidade dos centros de dados.

Eficiência, sustentabilidade e TI, andam juntas e são de extrema importância no mundo globalizado onde vivemos, no que diz respeito a gerenciar eficientemente os ativos de uma empresa, ou seja, tudo o que gera algum retorno benéfico, para que visam a suprir a necessidade de uma sociedade, diminuindo os impactos ambientais e não prejudicando as gerações futuras. Sendo assim, a sustentabilidade está totalmente em série com o desenvolvimento econômico material e tecnológico, servindo-se dos recursos terrestres inteligentemente.

A metodologia aplicada a pesquisa é de origem bibliográfica e qualitativa, sendo baseada em livros e artigos científicos. Em conjunto com um estudo de caso de caráter exploratório. Neste artigo mostra os parâmetros de melhoria sustentáveis em conjunto com os resultados e discussões, este de caráter exploratório e qualitativo. Em seguida mostra o cálculo do PUE "*Power Energy Effectiveness"* Eficácia no Uso da Energia, para analisar se os locais são sustentáveis por um valor obtido, caso contrário o gerente do centro de dados analisara a tabela de potência de consumo de modo a buscar uma alternativa de diminuir este valor.

## 2. REFERENCIAL TEÓRICO

Toda a evolução tem o seu preço a partir do momento em que demandamos uma alta carga de processamento, o mesmo vai consumir bastante potência elétrica. A partir disto, este artigo buscar abordar soluções através de pesquisas para favorecer as empresas que detenha centro de dados a levarem o termo sustentabilidade como sua prioridade principal. Neste capítulo será demonstrada toda a fundamentação teórica que serviu como base de estudo e auxiliou no entendimento para o desenvolvimento deste trabalho.

### 2.1 Sustentabilidade

De acordo com o conceito Sousa (2022), pode-se entender que o termo sustentabilidade é buscar o equilíbrio entre o suprimento das necessidades humanas e a preservação dos recursos naturais.

Quando se fala em sustentabilidade temos 3 aspectos importantes: pessoas, planeta e lucro, que formam o tripé da sustentabilidade (veja Figura 1). Estes três conceitos devem estar em pleno equilíbrio, de forma que todos saiam ganhando. Deste modo os negócios devem gerar ganhos, e estes lucros deve gerar benefícios para as pessoas, impactando minimamente o meio ambiente (ATADEMO, 2021).


**Figura 1:** Tripé da Sustentabilidade.

**Fonte:** Larissa Berlato, 2018.

Todavia, devemos sempre buscar o equilíbrio do meio ambientes descartando de forma correta o que pode agredir o mesmo. Por isso a TI verde está presente para auxiliar na projeção e descarte eficiente destes equipamentos. Segundo Murugesan (2008), TI Verde corresponde a soma da economia de energia com gestão de recursos desde a cadeias produtivas, e todo o ciclo que vai da extração de matéria-prima até o final da vida útil do equipamento, incluindo seu descarte. Tem como foco diminuir o desperdício e ser mais eficiente nos processos e recursos da organização.

Os ambientes da tecnologia da informação que visam o segmento de ser sustentável implementam a filosofia da TI verde, tendo como benefícios pessoas saudáveis e dispostas, a ter um planeta airoso e disposto de recursos.

Desde a sua criação, a indústria de TI se concentrou no desenvolvimento e implantação de equipamentos e serviços capazes de atender às crescentes demandas dos clientes empresariais. Menos atenção foi dada às questões de infraestrutura que incluem consumo de energia, refrigeração e espaço para data centers, uma vez que se supunha que eles

estivessem sempre disponíveis e acessíveis. "Ao longo da última década, essas questões tornaram-se fatores limitantes na determinação da viabilidade de implantação de novos sistemas de TI, enquanto o poder de processamento está amplamente disponível e acessível" (POHLE; HITTNER, 2008). Portanto, a ênfase tem sido no poder de processamento e nos gastos dos sistemas.

## 2.2 Sustentabilidade na TI

Ao termo Sustentabilidade na área de tecnologia segundo o conceito de (PRADO, 2008) é de tal compreensão de que as empresas devem gerir seus ativos tecnológicos eficientemente mantendo o equilíbrio com o meio ambiente e a sociedade.

Os data centers são ativos, e o seu lucro advém de benefícios a sociedade, tais como armazenamento de dados pessoais, banco de dados, organização e proteção de dados pessoais. Sobretudo, há uma demanda energética enorme para manter esta arquitetura em funcionamento 24 horas, 7 dias por semana.

"É estimado que o custo de montagem da infraestrutura de um data center, incluindo sistemas de arrefecimento, cabeamentos, energia e segurança variam de 80 mil até 150 mil dólares" (BINI, 2015).

E parte dessa energia é desperdiçada em forma de energia térmica, prejudicando os equipamentos de TI presentes nos raques. Aumentando a umidade da sala e ocasionando oxidação nas placas de circuito (também conhecida como PCBs, em inglês) e ferrugem na sua superfície, podendo gerar a perda total deste equipamento de alto valor.

"TI sustentável" e especialmente "serviços de TI sustentáveis" são termos que estão se tornando sinônimos de uma segunda onda emergente de inovação em computação verde. Estratégias de TI sustentáveis, estão levando a sustentabilidade além do uso de energia e considerações de produtos. Essa abordagem mais ampla da sustentabilidade corporativa exigirá o redesenho da organização de TI e, de fato, da própria empresa para que os benefícios estratégicos da computação verde sejam realizados. Essa segunda onda abrangerá a adoção de estratégias ecológicas que redefinirão os mercados, estimularão a inovação tecnológica e levarão a mudanças no processo, comportamento e cultura organizacional que integrarão modelos de negócios com responsabilidade ambiental e social (HARMON ROBERT, 2009).

Essas mudanças estão sendo impulsionadas pelas mudanças em evolução nos requisitos dos clientes de uma ênfase única no custo-benefício tangível do uso reduzido de energia para benefícios verdes cada vez mais intangíveis e questões culturais motivadas por preocupações com o aquecimento global e as mudanças climáticas (SENGE; SMITH, 2008).

Contudo, a sustentabilidade na tecnologia da informação se torna uma necessidade para as organizações, beneficiando empresas tais como: economia de recursos, manutenção do meio ambiente e melhora na imagem diante da sociedade.

## 2.3 TI verde

A TI Verde pode ser definida como o conjunto de práticas sustentáveis que reduzem ou minimizam os prejuízos do uso da tecnologia da informação. Seu conceito se desenvolveu com a premissa de que a tecnologia da informação está presente como ferramenta de produtividade em todos os campos da sociedade (MORAIS et al., 2015).

A questão é que quando discorremos sobre sustentabilidade e de tal importância comentar sobre a tecnologia e inovação verde.

A TI Verde pode apresentar como base a utilização de novas tecnologias que concedam equipamentos com menos ou nenhum risco ao meio ambiente, proporcionando a redução no consumo de energia, papel e o descarte, a reutilização e a reciclagem dos mesmos (VELTE, T.; VELTE, A.; ELSENPETER, 2008).

Em grandes centros de dados, principalmente os que estão com mais tempo de funcionamento no mercado tendem a ter equipamentos obsoletos, com isso estes apresentam eficiência energética muito abaixo dos padrões atuais, gerando assim um consumo de alta carga energética com um processamento inferior a de um produto atual com maior processamento e melhor eficiência.

As organizações brasileiras há tempos buscam soluções comprovadas por pesquisas no apoio às práticas Verde, são elas: economia de energia, virtualização de servidores e desktops, videoconferência, economia de papel, descarte e reciclagem de equipamentos eletrônicos (PINTO; SAVOINE, 2011).

Organizações que não se preocupam com a busca e a atualização de conhecimentos referentes à sustentabilidade de suas atividades tendem a aumentar os custos de produção, em virtude dos investimentos de capital e custos de operação, prejudicando o resultado financeiro da organização (KIM; KO, 2010).

Sendo assim vemos que a TI verde busca a evolução tecnológica com o foco na proteção do meio ambiente e a mínima utilização dos recursos energéticos, buscando junto a sustentabilidade o avanço das organizações sem denegrir as gerações futuras, com o uso racional dos recursos já escassos do planeta.

## 2.4 Data Centers

"Um centro de dados é central para as operações de TI de uma empresa. É um repositório para a maioria dos sistemas críticos de negócios, onde a maioria dos dados de negócios é armazenada, processada e divulgada aos usuários" (IBM Cloud Education, 2020).

Segundo a (IBM Cloud Education, 2020) Data centers ou centro de processamento de dados (em português), é uma instalação de um ou mais edifícios que abriga equipamentos de TI tais como (switchers, gatways, roteadores) para processamento e armazenamento de dados. Por isso é considerado o sistema nervoso de empresas, bem como processar uma quantidade enorme de informações.

A pandemia contribuiu para amplificar o uso de soluções virtuais, tanto que houve um aumento expressivo no tráfego de internet mundo afora.

No Brasil, de acordo com um relatório da Agência Nacional de Telecomunicações (Anatel), foram registrados 7,39 milhões de acessos móveis adicionais em 2020 (em comparação com 2019).

Os data centers são de suma importância para a sociedade e para a economia na totalidade, pois são fundamentais em vários setores tais como: sistemas de segurança, banco de dados, saúde pública, entretenimento e muitos outros. Assim, esse fluxo cada vez mais elevado implica, também, em uma maior demanda por água, para arrefecimento das máquinas, e por energia, uma vez que alimentar um Data Center, que opera 24 horas por dia, requer grandes quantidades dos recursos (O DATA, 2022).

> Em 2005, o American National Standards Institute (ANSI) e a Telecommunications Industry Association (TIA) publicaram o padrão ANSI/TIA-942, "Padrão de Infraestrutura de Telecomunicações para Data Centers" que define quatro níveis de data centers por vários níveis de confiabilidade ou resiliência (GIGABYTE, 2022).

Como exemplo, um data center Tier 1 é um nível básico para atender o funcionamento dos equipamentos de TI, e não disponibiliza de equipamentos redundantes na sua infraestrutura, entretanto um data center Tier 4 oferece subsistemas redundantes e alta segurança tais como proteção em quedas bruscas de energia.

### 2.4.1 Infraestrutura de um Data Center

Baseado em pesquisas feitas pelo Gartner (LERNER, 2014), empresa de consultoria Americana, o custo do tempo de parada do data center (downtime) é em torno de 300 mil dólares por hora. Dada sua natureza crítica de operação, as características mais importantes de um data center são: confiabilidade, disponibilidade e redundância. Por isso, esses são os principais índices utilizados para classificá-lo.

Os equipamentos de TI são montados em armários de metal chamados de raques, onde os mesmos possuem trilhos em suas lateiras fazendo com que os equipamentos entrem como se fosse uma gaveta (figura 2).

**Figura 2:** Um Equipamento de TI sendo colocado no raque

**Fonte:** The NBP, 2021.

Os chamados "servidores em raque" recebem o nome dos gabinetes que os abrigam. eles executam as mesmas funções que outros tipos de servidores, e a única coisa que muda é o método de pedido que traz consigo um grande número de vantagens. A largura dos servidores em raque e dos gabinetes e estruturas do raque é padronizada em 19 polegadas (48 cm). Graças a isso, servidores de diferentes fabricantes podem ser integrados no mesmo gabinete. Os armários com medidas sempre têm a mesma largura, mas sua altura e profundidade são variáveis, para permitir maior customização da instalação (SANCHEZ, 2016).

Na sala onde reside estes equipamentos possuí sistemas de proteção contra incêndio, além de sistemas de refrigeração, para manter uma temperatura desejável de operação, para que os equipamentos de TI não sofram superaquecimento e reduzam sua carga de trabalho.

## 2.5. Sustentabilidade em Data Centers

A evolução da internet no fim do século XX, trouxe consigo mudanças estruturais na sociedade, fazendo com que um número crescente de dispositivos acesse, simultaneamente, grandes volumes de dados em altas velocidades. Em 1992, o tráfego de dados na internet era de aproximadamente 100 GB por dia; apenas 10 anos depois, em 2002, esse tráfego global já era de 100 GB por segundo. Atualmente existem mais de 3 bilhões de usuários de internet no mundo e é possível fazer estimativas realistas sobre seus hábitos na rede (CISCO, 2015).

Este tema vem sendo abordado devido aos grandes custos energéticos dos equipamentos e também a maior demanda por eletricidade atrelado às preocupações ambientais. Devido ao elevado valor de acesso simultâneo tendência crescente do uso da Cloud computing.

Para as empresas terem data center mais eficientes precisamos considerar o investimento ao longo prazo. Então pode se dizer que o retorno dos investimentos na infraestrutura de data center é demorada, mas, é benéfica não apenas para o meio ambiente, como também para a sociedade.

O PUE "Power Utilization Effectiveness" Efetividade na utilização da Energia (em português). É a métrica mais utilizada para o cálculo de eficiência em Data centers que foi definida pelo Green Grid como a razão entre o consumo total de energia do data center e o total de energia consumida pelos equipamentos de TI.

$$PUE = \frac{PDC}{PTI}$$

Onde:

1. PUE = Power Utilization Effectiveness.
2. PDC = Potência total do Data Center.
3. PTI = Potência total dos equipamentos de TI.

No resultado desta equação teremos um resultado que quanto mais próximo do 1,00 melhor será a eficiência de um Data Center. PUE ideal é 1,0 onde toda a energia fornecida a um data center é consumida pelo equipamento de TI, entretanto, isto é impossível visto que, os equipamentos presentes na data hall geram calor, então é obrigatório um sistema de refrigeração para melhor funcionamento do mesmo. Com isso é necessário também sistema de iluminação nestas salas, impactando no aumento do resultado do PUE.

## 3. ESTUDO DE CASO

Para melhor rendimento dos data centers antigos e principalmente novos, com relação a sua sustentabilidade precisamos considerar os seguintes tópicos: Eficiencia Energetica; Eficiencia Termica e Controle Ambiental.

## 3.1 Eficiência energética

Segundo a ABESCO (Associação Brasileira das Empresas de Serviços de Conservação de Energia), "a eficiência energética consiste em usar eficientemente a energia para se obter um determinado resultado, é uma atividade que melhorará o uso das fontes de

energia. O incentivo à eficiência energética é um dos principais objetivos das políticas de energia, visto que contribui para a melhoria da gestão dos recursos, reduzindo assim os impactos ambientais e o consumo de energia."

Para melhorar a eficiência energética em data centers, dois tipos de energia de resfriamento precisam ser considerados. O primeiro tipo está associado aos custos de geração do ar refrigerado, ou seja, o custo bruto total dos produtos que fazem este ar gelado (centrais de água gelada "CAG" ou líquido refrigerante). E o segundo tipo está associado à entrega do ar frio, em outros termos o custo total para a produção dos canais de transporte fazendo com que este ar chegue a sala.

Uma empresa global que presta serviços financeiros efetuou melhorias na eficiência energética em seus ambientes de data center legados no Reino Unido (TOZER; FLUCKER, 2015). O índice de eficiência no uso de energia da instalação (PUE) foi de 2,3 no início do programa reduzido em 34% para 1,49, resultando em reduções significativas de custos operacionais e de energia e densidades de raque aprimoradas e capacidades do sistema. Isso foi alcançado por meio de uma avaliação de energia e pesquisa de temperatura do ar do data hall, implementação de melhorias no gerenciamento de ar, otimização do controle dos ventiladores da unidade de resfriamento, aumento gradual nos pontos de ajuste de temperatura do ar e da água gelada e instalação de um circuito de arrefecimento livre.

## 3.2 Eficiência térmica

Podemos caracterizar os data centers por um conjunto de métricas extraídas de medições em tempo real. Como existem várias layout estruturais de data centers espalhadas pelo mundo, então o desempenho de cada centro de dados é baseado em vários fatores, como sua estrutura, número de raques dentro do data hall, número de equipamentos de TI e as unidades de ar-condicionado. Com isso um conjunto de métricas comuns pode ser definido de modo a melhorar a eficiência térmica dos centros de dados. São elas: Os Data centers de confinamento, Aumento do fluxo de ar e aumentar a altura do teto.

### 3.2.1 Data centers de confinamento CAC/HAC

Os modelos de data center modernos que visam melhorar sua eficiência são os confinamentos do data center. Existem dois modelos o CAC- *cold aisle containment* confinamento do corredor frio, e a outra solução HAC- *hot aisle containment* confinamento do corredor quente. Ambos os modelos com sua proposta, sendo de responsabilidade do gerente do centro de dados escolher a opção que melhor se adeque as políticas da empresa.

#### 3.2.1.1 Confinamento corredor frio

No corredor frio a solução é evitar que o ar frio fornecido pelo sistema de ar-condicionado (CRAC) tome outra direção, a única maneira dor ar retornar ao sistema de refrigeração é através dos equipamentos de TI instalados no raques (figura 3).

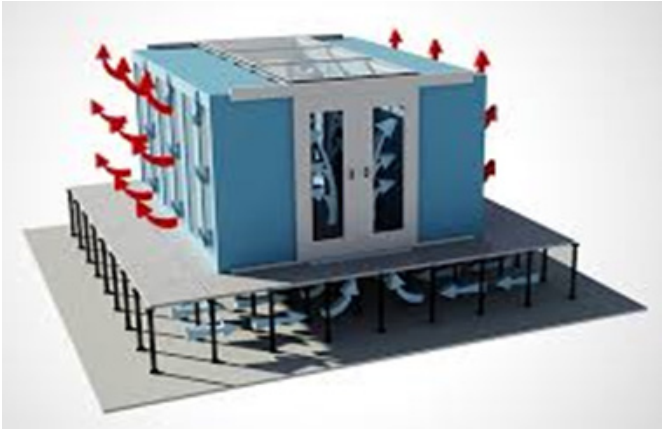


**Figura 3**:Confinamento corredor frio

**Fonte:** Infra News Telecom, 2021

Este tipo de confinamento tem como suas principais características: uma maior uniformidade na temperatura do corredor frio, fácil aplicação para raques padronizados e menos volume de ar frio na sala, pois haverá apenas nos corredores, porém os restos da sala é quente, o que seria inviável se caso precisar instalar os equipamentos fora de raques.

### 3.2.1.2 Contenção corredor quente

Esta solução, evitamos que o ar quente retorne aos servidores criando uma espécie de "duto" no teto entre o corredor quente e o sistema de ar-condicionado evitando que esse ar quente não circule pelos equipamentos de TI (figura 4).

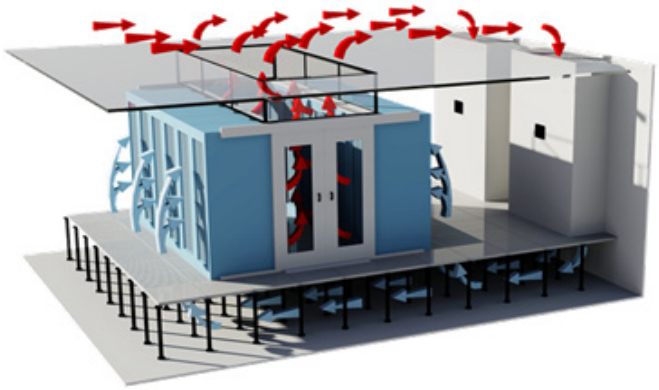


**Figura 4**:Confinamento corredor quente

**Fonte:** Infra News Telecom, 2021

As principais características deste confinamento são: um maior volume de ar frio (no caso de toda a sala sendo facultativo o uso do piso elevado), permite a instalação de equipamentos "*santd alone*" sem se preocupar com superaquecimento, fácil de aplicar quando os racks não são padronizados. Porém o corredor tem temperatura elevada, o que é um problema se algum técnico ficar  por muito tempo.

### 3.2.2 Aumento do fluxo de ar

O piso perfurado localizado nos corredores quentes fornece ar frio aos servidores e aos equipamentos de TI, e a porcentagem de abertura e pressão de subfluxo estão controlando a quantidade total de fluxo de ar. Os ladrilhos que são as furacões localizado no piso elevado e que permite a entrada de ar para a data hall (Figura 5).

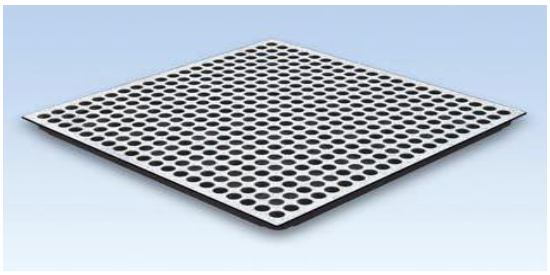


**Figura 5:** piso perfurado com ladrilhos redondos

**Fonte:** archiexpo, 2022

Podemos frisar que quanto menor a área de abertura dos ladrilhos menor é o fluxo de ar na sala e maior será a resistência enfrentada pelas Unidades de ar condicionado "ACUS" (em inglês), em consequência disto, menor será a vazão total alcançada. O que torna necessário a troca deste piso por um com ladrilhos com um diâmetro de abertura maior possibilitando uma maior passagem do ar frio.

A figura 6, mostra do lado direito o piso com ladrilhos de diâmetros muito pequenos, e a direita temos os ladrilhos com um diâmetro grande, possibilitando um melhor fluxo de ar.

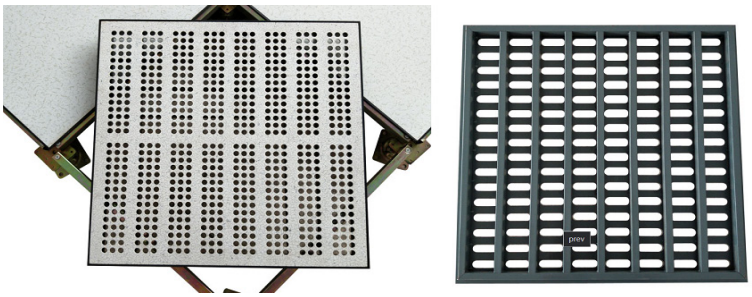


**Figura 6:** A direita piso com ladrilhos de pequeno diametro e a direita com maior

**Fonte:** basicfloor.biz, Ano:2022

Contudo, ladrilhos maiores permite um melhor fluxo de ar para a sala dos servidores e equipamentos de TI, permitindo uma troca de calor mais eficiente.

### 3.2.3 Aumentar a altura do teto

A região acima dos raques, entre a superfície e o teto atuam como um pleno para que o ar quente da exaustão possa passar até chegar as ACU "unidades de ar condicionado", um teto baixo pode ocasionar a retificação desse ar quente em um local específico em vez de serem puxados para as ACUS da data hall, entretanto se o teto for alto demais pode forçar as unidades de ar-condicionado a trabalharem o dobro da capacidade permitida pela fabricante, ocasionando negativamente a durabilidade do equipamento. Estudos de modelagem CFD realizados como parte do projeto atual indicam que aumentar a altura do teto de 22,23 cm para 28 cm, pode resultar em uma redução de 2,3 °C na temperatura

do raque na região do ponto quente.

## 3.3 Controle ambiental nos data centers

Os equipamentos de TI presentes nos data centers trabalham a 100% de sua carga de operação ou bem próximo, (depende muito de fabricante) os mesmos têm um período de vida, que pode oscilar para mais ou para menos dependendo da eficiência de usabilidade das empresas. Com isso é necessário com que as empresas busquem soluções para o descarte sustentável destes equipamentos, então devemos considerar.

- Descarte eficiente dos equipamentos defeituosos
- Utilização de energia de fontes renováveis para alimentação secundaria.

### 3.3.1 Descarte eficiente dos equipamentos defeituosos

Como mencionado no tópico 3.3, naturalmente, o trabalho excessivo dos servidores para compensar a falta de um sistema eficiente de climatização gera fadiga e desgaste. Em consequência disto, você poderia estar trocando o hardware que deveria dura 5 anos, tendo que substituir os mesmos em 3 anos, ocasionado pela falta da implementação dos métodos de eficiência e sustentabilidade mencionados neste artigo.

**Figura 7:** Sucata dos equipamentos de TI

**Fonte:** fotografia e companhia

O descarte com uma fiscalização ruim os componentes químicos presentes nos circuitos eletrônicos que moldam os equipamentos de TI e os servidores no data center podem contaminar o solo, em contrapartida, geraram prejuízo de usabilidade para usuários, uma vez que os mesmos queiram acessar seus arquivos ou qualquer outro serviço que este data center que apresentou equipamento com defeito possa oferecer. Por outro lado, temos prejuízos ambientais gerado pela reciclagem irregular de algumas empresas, os componentes químicos presentes nestes equipamentos uma vez em contato com o solo não só pode gerar doenças aos seres humanos como também degradar a terra daquela região, contaminando a colheita pecuária ou a impossibilidade da mesma.

## 3.3.2 Utilização de energia de fontes renováveis pra alimentação secundaria

De fato, que o consumo de energia em grandes centros de dados é grande, isso se dá não apenas pelos equipamentos de TI, trabalharem a quase toda sua potência, mas também aos sistemas de iluminação e principalmente o sistema de refrigeração, estes funcionando 24h por dia, 7 dias por semana.

Os data centers dependem de pelo menos duas fontes de energia elétrica, a primeira é a companhia elétrica da região e a secundaria são grandes geradores que funcionam a base de óleo diesel, gerando grandes emissões de $CO_2$ na atmosfera terrestre. Como proposta de intervenção em relação à fonte de energia secundaria, este artigo sugerem as fontes de energia renováveis. Como a eólica e a solar fotovoltaica.

### 3.3.2.1 Energia Eólica

A energia eólica é uma energia sustentável em que transforma a força do vento (energia cinética) em energia elétrica, sua geração acontece no interior dos aerogeradores, grandes estruturas que se assemelham a moinhos. Girando sua hélice e ativando o rotor eólico onde as pás estão conectadas (figura 7).

**Figura 8:** Campo de geradores Eólicos

**Fonte**: Sitesustentavel, 2022.

Considerada uma energia limpa e barata em comparação as outras fontes de energia, seria ideal para grandes data centers, entretanto, a desvantagem é para alimentar um data center em caso de queda da fonte primaria é necessária uma grande potência, assim os aerogeradores precisariam ocupar uma faixa de terra enorme.

### 3.3.2.2 Energia Solar

A outra fonte de energia seria a solar fotovoltaica, é uma fonte alternativa renovável e sustentável de energia que provém da radiação eletromagnética (luz e calor) emanada diariamente pelo sol (figura 8).

**Figura 9:** Painéis fotovoltaicos captando a luz e o calor do sol

**Fonte:** epocanegocios, 2022.

A sua desvantagem é a dependência do sol como "combustível" para a geração de energia elétrica, há países que não possui a presença da luz solar o ano inteiro ficando esta possibilidade aos países tropicais como exemplo o Brasil. Sendo uma alternativa sustentável e viável a pequenos e médios data centers. Já em grandes data centers é necessário um terreno exclusivo para as instalações dos painéis fotovoltaicos afins de obter a potência necessária de alimentação.

## 4. RESULTADOS E DISCURSÕES

Antes de começar o tópico de resultados e discursões é importante salientar que o método de chegada aos resultados do tópico 4.1 ao 4.4 foram de pesquisas bibliográficas de aptidão explicativa.

### 4.1 Eficiência energética: resultados

No caso da empresa de serviços financeiros, 80% do ar CRAH estava sendo desviado e cerca de 20% do ar de entrada do servidor era ar quente recirculado. A disponibilidade de fluxo foi 4. Todas as temperaturas de entrada de ar do servidor medidas estavam entre 15 °C e 27 °C.

As seguintes medidas foram implementadas para melhorar a gestão do ar:

1. Instalação de placas cegas dentro dos raques.
2. Vedação de lacunas no piso elevado, incluindo recortes de cabos do gabinete e lacunas em torno das bases da unidade de distribuição de energia (PDU)
3. Colocação revisada das grades do piso onde o resfriamento é requerido.

### 4.2 Eficiência térmica: resultados

As propostas para melhorar a eficiência energética/térmica dentro do data hall citados no tópico 3.0: O aumento do fluxo de ar melhora o rendimento em até 10% permitindo o desempenho de processamento além de prolongar a vida útil do equipamento. A instalação de cortinas tem como resultado a separação dor ar quente com o frio, impedindo a proliferação de umidades nos equipamentos. Aumentar a altura do teto como mencionado no tópico 3.2.3, estudos em modelagem CFD mostra um caso de um data center com a altura entre a superfície do raque e o teto de 22,23 cm, quando aumentaram para 28 cm ouve uma redução de 2,3° C da temperatura do corredor quente, o que possibilitou as ACUS não trabalharem acima da sua potência permitida, evitandos gastos excessivo com manutenções e até mesmo evitando o risco de incêndios.

## 4.3 Controle ambiental: resultados

Os data centers precisam de uma fonte de energia reserva para caso a fonte principal caia, neste caso grande parte utiliza grandes geradores movidos a diesel (um grande poluente). A proposta de utilizar as energias renováveis mencionadas nos tópicos 3.3.2.1 a 3.3.2.2 é uma alternativa para que os data centers de pequeno e médio e até mesmo grande de porte diminuem a emissão de carbono assim se tornando mais sustentáveis e eficientes.

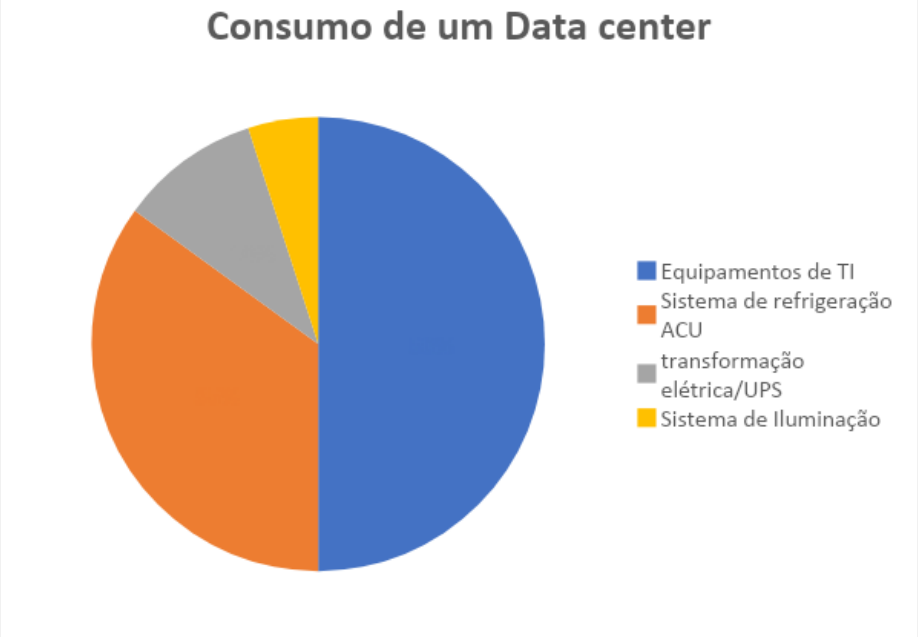
Em relação à reciclagem dos equipamentos defeituosos e de tal responsabilidade da empresa a fiscalização do ponto de coleta até o ponto de descarte das peças. Infelizmente existem empresas que não tem esta visão de sustentabilidade, e pensado apenas no lucro bruto, e não o quanto a empresa reduziu o seu impacto ambiental. Sendo assim é necessário que a justiça em conjunto com o governo crie leis que punem estas empresas com multa diária conforme o volume de equipamentos descartados. Contudo, isto deve ajudar as empresas a colocarem a sustentabilidade como pauta no seu portfólio.

## 4.4 Método PUE: resultados

Já realizado todas as propostas de intervenção mostradas neste artigo, podemos calcular o método PUE para analisarmos a eficiência de um data center. Considere a seguinte situação.

No gráfico abaixo (Gráfico 1), temos um exemplo de um data center relacionado ao seu consumo onde 50% da energia é consumida pelos equipamentos de TI (em azul), 35% do consumo do sistema de refrigeração (em vermelho), 10% do consumo vai para os transformadores, que transforma a tensão da concessionária geralmente de 10 a 50 mil volts (ou mais) para 220 ou 110v, e também os UPS que podemos caracterizá-los como grandes nobreaks de armazenamento de energia (em verde), por fim 5% do gasto com o sistema de iluminação (em roxo).

**Gráfico 1:** Consumo de um data center.




**Fonte:** Autoral, 2022.

Agora vamos montar a tabela (Tabela 1) com os respectivos valores em kilo watts (Kw) de consumo individualmente atrelado a cada um dos itens.

**Tabela 1**: Dados em kw de um data center

| Classe | Porcentagem | Consumo em kw |
|---|---|---|
| Equipamentos de TI | 50% | 2200 kw |
| Sistema de Refrigeração | 35% | 1100kw |
| Transf. Elétrica /UPS | 10% | 330kw |
| Sistema de Iluminação | 5% | 70kw |
| TOTAL | 100% | 3700kw |

**Fonte:** Autoral, 2022.

Ultilizando a fórmula citada no topico 2.5 e com o levantamento dos dados podemos calcular:

$$PUE = \frac{3700Kw}{2200kw} => PUE = 1{,}68$$

O valor do PUE ideal será (1,00), onde toda a energia fornecida seria consumida apenas pelos equipamentos de TI. Porém, é impossível visto que, precisamos dos outros sistemas para que o data center funcione corretamente.

Contudo, o resultado foi de 1,68. Então para melhorar a sua eficiência de modo que fique próximo do valor de 1,00, podemos citar o data center do Facebook, em Prineville, OR, que é uma das infraestruturas de centro de dados mais eficiente do mundo desde que se tornou operacional, com seu PUE de 1,07 em plena carga que foi verificado durante o comissionamento.

Deve se fazer um estudo para entender o que está causando um consumo excessivo, por exemplo: pode ser que os equipamentos de TI já estejam ultrapassados ou o sistemas de refrigeração e/ou iluminação estão sendo ineficientes. Este artigo apresentou as propostas de intervenção visando torna os data centers mais eficientes e sustentáveis.

## 5. CONCLUSÃO

Pela evolução tecnológica que tivemos ao logo de tempo o mundo se automatizou, entretanto, isso acarretou um consumo de energia muito alto e junto a isso a emissão de dióxido de carbono ($CO_2$) na atmosfera tem gerado impactos ambientais consideravelmente alto. Os centros de dados são estrutura vitais para a manutenção e disponibilidade de serviços em uma sociedade, e para estarem disponíveis interruptamente depende do constante fornecimento não apenas de uma alta potência elétrica como também de uma abundância de água para o sistema de resfriamento, assim mantendo os equipamentos operantes.

O estudo abordado neste artigo, sugere propostas de intervenções de modo a melhorar os data centers independente do seu tamanho, assim com a TI verde, tornando-os eficientes e sustentáveis. Sabemos que a renda de um data center a devem de seus serviços a sociedade como exemplo: segurança e fornecimento de dados e arquivos. Porém, quaisquer propostas para melhorar a sua eficiência seja energética, térmica ou ambiental será benéfico não apenas, para a sociedade como também ao planeta.

## Referências


ATADEMO, Robert. **ENTENDA OS TÊS PILARES DA SUSTENTABILIDADE**. 2021, Dísponivel em:<terra-anbiental.com.br> Acesso em:27/08/2022.

BINI, W. Tecnologia Sustentável: **O DESENVOLVIMENTO DE DATA CENTERS ECOLÓGICOS**, Disponível em:. Acesso em: 28/08/2022

CISCO. **THE ZETTABYTE ERA: TRENDS AND ANALYSIS**. 2015. Disponível em: <http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-indexvni/VNI_Hyperconnectivity_WP.html>, Acesso em 27/09/2022.

HAMANN Hendrik; MARIANNO Fernando; KLEIN Levete. **ENERGY EFFICIENCY ASSESSMENT OF DATA CENTERS USING MEASUREMENT AND MANAGEMENT TECHNOLOGY**. 2021. Disponível em: <https://ieeexplore.ieee.org/document/9821731/authors#authors> Acessado em: 20/08/2022.

HARMON, R. R; Auseklis, N. Sustainable IT services. **ASSESSING THE IMPACT OF GREEN COMPUTING PRACTICES**. PICMET, 2 a 7 de agosto 2009. Acesso 25/08/2022.

IBM CLOUD EDUCATION. **DATA CENTER**. Disponivel em: <https://www.ibm.com/cz-en/cloud/learn/data--centers>. Publicado em 2020. Acesso em:28/08/2022.

Kim, Y.; Ko, M. (agosto, 2010). **IDENTIFYING GREEN IT LEADERS WITH FINANCIAL AND ENVIRONMENTAL PERFORMANCE INDICATORS**. Proceedings of the Americas Conference on Information Systems, Lima, Peru, 16. Acesso 29/08/2022.

LERNER, Andrew. **THE COST OF DOWNTIME**. 2014. Disponível em: <Khttp://blogs.gartner.com/andrew--lerner/2014/07/16/the-cost-of-downtime/> Acesso em 30/08/2022.

MARCELO, BARBOZA. **CONFINAMENTO DE CORREDORES EM DATA CENTERS**. Ano: 2021, disponível em: <infranewstelecom.com.br>, Acesso em: 23/10/2022.

MORAES, Samuel de Barros; LANGHI, Celi; TEIXEIRA, Elisabeth Pelosi. **Tecnologia da Informação sustentável (Green IT) - O que é relevante para as empresas brasileiras?.** X WORKSHOP DE PÓS-GRADUAÇÃO E PESQUISA DO CENTRO PAULA SOUZA, SÃO PAULO – SP. Acesso em: 31/08/2022

MURUGESAN, S. (2008). **Harnessing green IT: Principles and practices**. IT professional, 10(1), 24–33. Acesso em: 01/09/2022

O DATA. **INFRAESTRUTURA DE DATA CENTERS,** (setembro, 2022), Disponível em: <https://odatacolocation.com/data-center/> Acesso em: 28/08/2022.

POHLE, G. and J. Hittner. **ATTAINING SUSTAINABLE GROWTH THROUGH CORPORATE SOCIAL RESPONSIBILITY**. IBM Institute for Business Value, White paper, 20 pages, 2008, Disponivel em:<www.ibm.com>. Acesso em: 27/08/2022.

PRADO, Alex. **SUSTENTABILIDADE EM TI.** Disponível em: <https://administradores.com.br/artigos/sustentabilidade-em-ti>. Publicado em 2008. Acesso em: 29/08/2022.

PINTO, T. M. da C; SAVOINE, M. M. **Estudo sobre TI Verde e sua aplicabilidade em Araguaína.** Revista Científica do ITPAC. 2011, v.4, n.1, p. 11-12, 2011.

SANCHEZ C., Pilar. **QUÉ ES UN SERVIDOR RACK**. 2016. Disponivel em: http://pcs-box.blogspot.com/2011/07/que-es-un-servidor-rack.html Acesso em:30/08/2022.

SENGE, P., B. Smith, N. Kruschwitz, J. Laur, and S. Schley, The ecessary Revolution: **HOW INDIVIDUALS AND ORGANIZATIONS ARE WORKING TO CREATE A SUSTAINABLE WORLD**. New York: Double Day, 2008. Acesso em: 27/08/2022.

SOUSA, Rafaela. **SUSTENTABILIDADE**. Brasil Escola. Disponível em: https://brasilescola.uol.com.br/educacao/sustentabilidade.htm. Publicado em mar/2022. Acesso em 26 de agosto de 2022.

TAURION, C. **O QUE É CLOUD COMPUTING?** Disponível em: http://www.oficinadanet.com.br/artigo/1008/o_que_e_cloud_computing. Publicado em 2008. Acesso em: 22/07/2022.

TOZER R, FLUCKER, S. **DATA CENTER ENERGY-EFFICIENCY IMPROVEMENT CASE STUDY**. ASHRAE 2015;121. Acesso em: 03/09/2022.

VELTE, T.; VELTE, A.; ELSENPETER, R. **Green IT: reduce your information systems environmental impact while adding to the bottom line**. New York: McGraw-Hill, 2008.

# AUTORES

## BEZERRA, Ilgner Mendes

Graduado em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), maranhense, desenvolvedor mobile, entusiasta em fotografia, "astrônomo de quintal" e nas horas vagas goleiro de futsal. Filho de dona Marly e seu Manoel, Marcelo nasceu em 2000 no interior do Maranhão, trabalhou na roça quando criança e ingressou na faculdade em 2019 com o sonho de se tornar um profissional de TI, tendo dedicado seu período acadêmico à programação de aplicativos e suas interfaces dentro da empresa Júnior SEEDS Tecnologia.

## CORREIA, Marcelo da Conceição

Graduado em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), Brasil.

## FERREIRA, Carlos Eduardo Júnior

Graduado em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), Brasil.

## LOPES, Daniel Santos

Graduado em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), BRASIL, possui curso técnico na área de informática e em montagem e manutenção de micro.

## MATOS, Lucas Yan Costa

Graduado em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), Brasil.

## PORTELA, Juliana Serra

Graduado em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), formação técnica em Mecatrônica pela Netcom Treinamento e Soluções Tecnológicas.

## SANTOS, Mariane Soares dos

Graduada em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), Brasil.

## SILVA, Julyana, Corrêa

Graduada em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), possui uma breve experiência com o desenvolvimento de sites pelo *Joomla*, e conhecimentos com o *Mysql*.

## SILVA, Ricardo, Santos

Graduado em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), possui experiência na área de Análise de Dados com linguagem Python.

## SOUSA, Leonardo Silva

Graduado em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), possui experiência no desenvolvimento de sistemas Web, atuando principalmente no desenvolvimento do *back-end* utilizando o *framework Spring Boot*.


Pascal
Editora