CRIANDO SOLUÇÕES TECNOLÓGICAS COM A ENGENHARIA DE COMPUTAÇÃO

VOLUME 2

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 2

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)

L732c

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

179 f. : il.: (Criando soluções tecnológicas com a engenharia de computação; v. 2)

Formato: PDF

Modo de acesso: World Wide Web

ISBN: 978-65-80751-66-2

D.O.I.: 10.29327/5192626

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

CDU: 004.41

2023

www.editorapascal.com.br

contato@editorapascal.com.br

APRESENTAÇÃO

O progresso das tecnologias traz grandes desafios para o Engenheiro de Computação, pois esse profissional deve integrar conhecimentos múltiplos, necessários para desenvolver soluções tecnológicas para diversas áreas. Esse livro apresenta um conjunto de experiências sobre o processo de concepção, projeto, implementação e operação que intencionam colaborar com os estudantes no planejamento e no desenvolvimento de softwares e hardwares.

Os trabalhos apresentados nos 10 capítulos apresentam soluções propostas pelos estudantes de Engenharia de Computação da Universidade Ceuma, cujas habilidades e competências foram aprimoradas ao longo do curso por meio de atividades e experiências vivenciadas dentro e fora de sala de aula.

Tais vivências possibilitaram mobilizar competências e conhecimentos na busca de soluções para problemas diversos e explorar diferentes caminhos de concepção, projeto, implementação e operação de sistemas, utilizando frameworks, IoT e desenvolvendo APIs, front-end, back-end e aplicativos para diversos sistemas operacionais.

A velocidade das inovações tecnológicas exige do profissional de Engenharia de Computação atualização contínua, adaptabilidade e versatilidade. Neste contexto, o mercado tem procurado por profissionais que tenham habilidades que vão além do conhecimento em tecnologia e, isto, impõe grandes desafios aos discentes, que devem desenvolver habilidades para resolver problemas complexos. Por isso, o desenvolvimento do pensamento computacional para propor soluções criativas, inovadoras e integradas e em como implementá-las, tem sido cada vez mais valorizado e se tornado imprescindível.

O Engenheiro de Computação pode assim atuar e exercer diversas funções colaborando com a sociedade na construção do progresso tecnológico.

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

API REST EM JAVA WEB PARA HOTÉIS: CONSULTA, RESERVA E CADASTRO DE HOTÉIS NO APLICATIVO WEB DA MATRIP

JAVA WEB API FOR HOTELS: CONSULTATION, BOOKING AND REGISTRATION OF HOTELS ON THE WEB MATRIP APP

Sam Helson Nunes Diniz[1]

Edilson Carlos Silva Lima[2]

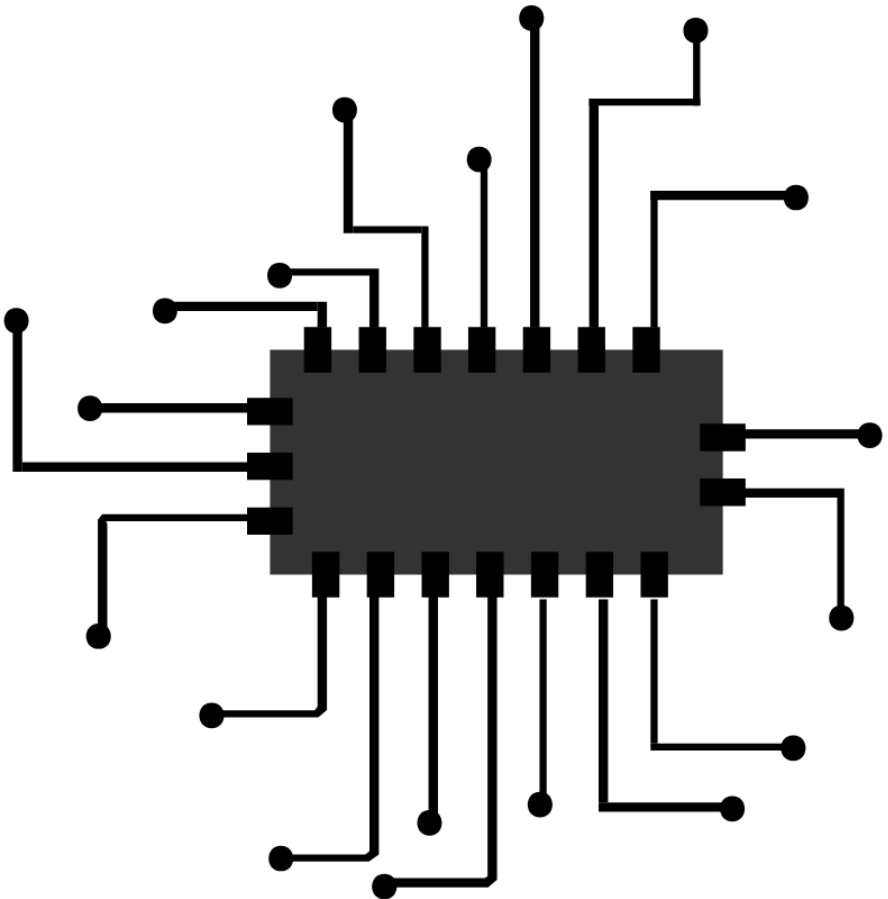
Jonathan de Araújo Queiroz[3]

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

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

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

{DINIZ, Sam Helson Nunes, sam04hel@gmail.com; LIMA, Edilson Carlos Silva, edilsonlima3@gmail.com; QUEIROS, Jonathan de Araújo, queirozjth@gmail.com}



d.o.i.: 10.29327/5192626.1-1

Resumo

O turismo é uma das principais formas de movimentar o mercado, gerar emprego e aumentar a renda. O Maranhão é um grande polo turístico no Brasil a ser descoberto, com suas belezas naturais, culinárias e históricas e atrai diversos turistas o ano inteiro. A cultura encanta com seus festejos e festas populares, seu folclore e sua rica história. O estado recebe milhares de turistas nacionais e internacionais que buscam atrações, conhecimento e experiencias únicas e inovadoras, porém ao chegar na cidade alguns não sabem quais hotéis, passeios e a rica gastronomia que podem conhecer e desfrutar. Então buscam uma agência que ofereçam opções de hospedagem e lazer a esses clientes que são fonte de renda nesse ramo e são importantes para o desenvolvimento econômico. Contudo, o cliente busca descoberta e ele criar seus roteiros turísticos (hospedagem, passeios, gastronomia), apesar dos serviços oferecido pelas agências, o turista quer construir seus sua rotina de lazer de acordo com seu perfil, de forma atrativa e barata, com opções fora de pacote das agências. Este artigo descreve os processos utilizados na criação de uma API REST em java web utilizando o *framework* do ecossistema *Spring Boot, utilizando o Swagger-UI, Heroku* para a realização de consultas, reservas e cadastro de hotéis no aplicativo web no Aplicativo da Matrip.

Palavras-chave: API, Spring boot, Swagger-UI, Heroku.

Abstract

Tourism is one of the main ways to move the market, generate jobs and increase income. Maranhão is a major tourist hub in Brazil to be discovered, with its natural, culinary and historical beauties and attracts many tourists throughout the year. Culture enchants with its celebrations and popular festivals, its folklore and its rich history. The state receives thousands of national and international tourists who seek attractions, knowledge and unique and innovative experiences, but when arriving in the city, some do not know which hotels, tours and the rich cuisine they can discover and enjoy. So they look for an agency that offers accommodation and leisure options to those clients who are a source of income in this field and are important for economic development. However, the client seeks discovery and he creates his tourist routes (accommodation, tours, gastronomy), despite the services offered by the agencies, the tourist wants to build his leisure routine according to his profile, in an attractive and cheap way, with options out of agency package. This article describes the processes used in the creation of a REST API in java web using the Spring Boot ecosystem framework, using Swagger-UI, Heroku to carry out queries, reservations and registration of hotels in the web application in the Matrip Application.

Keywords: API, Spring boot, Swagger-UI, Heroku.

1. INTRODUÇÃO

O turismo é uma das principais formas de movimentar o mercado, gerar emprego e aumentar a renda. O Maranhão possui uma gama de lugares turísticos, com suas arquiteturas naturais, culinárias e histórias, a cultura encantadora e festejos, enriquecem sua história, e acarretam atrair turistas para sua região.

A disponibilidade e conhecimento sobre hospedagem fica restrita apenas para quem conhece a região, e para turistas que desconhecem, ao ficar no mesmo lugar por vários dias, que contenham muitos passeios que não podem ser executados todos no mesmo dia, a decisão do hotel, baseado na localização dos passeios que ele vai escolher é de extrema importância para agilização e barateamento dos processos, e entretenimento do cliente.

Portanto este artigo tem como objetivo o desenvolvimento de uma *API REST* para um sistema que tem como função consulta, reserva e cadastro de Hotéis na aplicação da Matrip. Também a inclusão do Hotel em conjunto com o passeio todos em um pacote único, para pagamento único. O projeto será desenvolvido na plataforma Java utilizando o *framework* do ecossistema *Spring Boot*.

O artigo está dividido em 5 capítulos, além da introdução. O capítulo 2 visa uma revisão em elementos teóricos necessários para o desenvolvimento do projeto. O capítulo 3 mostrará 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. No capítulo 5 será feito a conclusão, onde reúne as considerações finais e algumas sugestões de melhorias.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será abordado alguns conceitos que serão utilizadas na implementação do sistema, os assuntos foram selecionados com base no conhecimento que foram adquiridos ao longo do curso de engenharia da computação e 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 *Framework Spring Boot*

Neste projeto foi utilizado o *framework Spring Boot*. O *Spring Boot* é uma forma de criar aplicações baseadas no *framework Spring* de forma simples e rápida. Nelas, já existe um contêiner *Web*, que pode ser o *Tomcat* ou o Jetty, e a aplicação é executada com apenas um *run*, diferentemente de quando é necessário primeiro instalar e configurar um contêiner, gerar um arquivo WAR (*Web Application Resource*) e por fim implantá-lo no contêiner. Caio Costa (2021) afirma:

> Desde que o Spring surgiu, acredito que o intuito dos criadores sempre foi facilitar nossa vida e com o Spring Boot não é diferente, ele nos entrega uma forma simples e rápida de se criar uma aplicação Standalone, visto que traz consigo "embutido" um Apache TomCat, um WebServer muito bem difundido no mercado há muitos anos (COSTA, 2021).

O *Spring Boot* também facilita a configuração das aplicações através de arquivos *properties* ou diretamente no código, não sendo necessário o uso de arquivos XML.

2.2 REST

REST (Representational State Transfer) é uma descrição técnica de como a *World Wide Web* (WWW) funciona. Especificamente, REST é como a *Web* atinge sua grande escala. Se pode dizer que a *Web* tem um "sistema operacional", seu estilo de arquitetura é REST. Mark (2011) diz:

> "Para muitos de nós, projetar uma API REST às vezes pode parecer mais como uma arte do que uma ciência".

Uma interface de programação de aplicativos REST (API REST) é um tipo de servidor *web* que permite que um cliente, seja operado pelo usuário ou automatizados, para acessar recursos que modelam os dados de um sistema e funções.

2.3 API

A API foi construída utilizando o modelo *Web RESTful*, que por sua vez, entrega informações via verbos HTTP, operações para transferência de representações entre clientes e servidores, no formato JSON, como GET, POST, PUT, PATCH e DELETE. Dan Woods (2012) afirma:

> "As APIs são um grande negócio, e estão ficando cada vez maiores. Empresas pioneiras como Google, Facebook, Apple e Twitter têm exposto ao público soluções tecnológicas incríveis, transformando negócios existentes e criando novas indústrias".

Há diversos métodos HTTP, porém os mais utilizados neste projeto foram os citados abaixo:

1. GET: Retorna dados específicos.
2. POST: Adiciona ou criar novos dados.
3. PUT: Atualiza dados existentes.
4. PATCH: atualiza parcialmente um dado específico
5. DELETE: deleta os dados específicos.

2.3 *Web Service*

Web Service é o manuseamento de dados através de protocolos de comunicação, podendo ser utilizados em diferentes plataformas, independente da linguagem de programação. Uma *Web Service* possui uma interface, que oculta os detalhes de execução para que possa ser usado independentemente da plataforma de *hardware* ou *software* em que está implementado e independentemente da linguagem de programação em que está, ou vai ser escrito. Allamaraju (2010) cita:

> "Em Web Services RESTful, seu principal objetivo, deve ser manter a visibilidade na medida do possível".

Ela também permite a reutilização de sistemas já existentes permitindo a reutilização dos mesmos, fazendo com que seja possível a melhoria desses sistemas já existentes.

2.5 *Swagger-UI*

O *Swagger* é um *framework open source* que auxilia os desenvolvedores nos processos de definir, criar, documentar e consumir API's 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. Allamaraju (2010) cita:

> "Swagger é uma área que eu sinto que não deveria ser tudo tão complicado. No entanto, vi muitas pessoas que precisavam tomar decisões que afetaria muitos outros".

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.

2.6 Maven

Apache Maven é um projeto de código aberto baseado em padrões estruturais de gerenciamento que simplificam a construção, o teste, relatórios e o empacotamento de projetos. Essa ferramenta permite que os desenvolvedores criem e documentem estruturas de ciclo de vida.

As raízes iniciais do Maven foram no projeto *Apache Jakarta Alexandria* que ocorreu no início 2000. Posteriormente, foi usado no projeto *Apache Turbine*. No livro "*Introducing Maven: A Build Tool for Today's Java Developers*" Balaji Varanasi afirma:

> "O Maven se tornou um dos softwares de código aberto mais usados em programas de software em empresas em todo o mundo. E há várias razões pelas quais o Maven é tão popular".

O Maven é escrito em Java é utilizado para construir projetos escritos em Scala, C#, Ruby etc. Sua utilização torna o projeto mais fácil para os desenvolvedores Java desenvolverem relatórios, criarem verificações e testar a configuração de automação).

A versão 1.0 foi lançada em 2004, seguida pela versão 2.0 em 2005, atualmente o Maven se encontra na versão 3.8.6 (11/22).

2.7 Heroku

O Heroku é uma plataforma que pode ser considerada como um sistema operacional da *Web*, que nos permite hospedar códigos e não se preocupar com a disponibilidade, escala e infraestrutura da aplicação. Middleton cita:

> "Como um desenvolvedor, você pode escrever seu aplicativo da mesma forma que um game-designer faria, você não precisa se preocupar com os detalhes, Heroku é uma plataforma que cuida de todas essas coisas e permite que você se integre a ele como quiser".

Ela é mais utilizada para aplicações de *back-end*, como as desenvolvidas em NodeJS, Ruby, Java, PHP, Python, Go, entre outras.

2.8 *Endpoints*

Em um serviço *web services* REST, *Endpoints* são recursos que permitem que o cliente consuma a api por meio de uma URI (*Universal Resource Identifier*), uma cadeia de caracteres compacta usada para identificar ou denominar um recurso na Internet. Éder Santos, no site Yssy afirma:

> "Os endpoints especificam onde as APIs ingressam, tendo papel fundamental para o correto funcionamento do software com que está interagindo".

A seguir, estão listados exemplos de URIs disponíveis no sistema para serem consumidas pelo cliente.

Usando o Método GET http://localhost:8080/hotels lista todos os hotéis cadastrados na matrip.

Usando o Método GET http://localhost:8080/findHotelsByDestaques lista os hotéis que estão cadastrados na categoria de destaques.

Usando o Método GET http://localhost:8080/QueryBy_DataEntry_DataOut_AmountPeople_TotalPrice faz uma pesquisa de acordo com a data de entrada e de saída, a quantidade de pessoas, e exibe os preços baseados nas informações que o cliente forneceu.

Usando o Método GET http://localhost:8080/find/hotelsByName pesquisa hotéis por nome.

Usando o Método POST http://localhost:8080/booksHotel o cliente faz uma reserva no hotel desejado.

2.9 Metodologia

A metodologia abordada é um estudo de pesquisa bibliográfica, e a implementação de uma api de hotéis na aplicação web e mobile da matrip, que compreendeu pesquisas dos últimos dez anos, nos idiomas inglês e português. A busca, coleta de dados, seleção e leitura foram conduzidos por apenas um pesquisador responsável pelo estudo de forma independente.

3. ESTUDO DE CASO

Neste capítulo vamos visualizar o processo de desenvolvimento da aplicação que visa solucionar o problema elencado. Dito isso, no item 3.1 abordaremos a solução realizada para resolver a problemática, no item 3.2 vem a etapa de diagramação de classe, no item 3.3 temos a descrição da anotação *Entity* e sua utilização, no item 3.4 apresenta-se a

descrição da anotação *Repository*, no item 3.5 dispõe-se acerca do *Controller*.

3.1 Solução para o problema

Buscar um guia turístico que ofereça opções de pacotes que incluem passeios turísticos, junto à hospedagem, é uma opção que facilita muito para o cliente, já que são quem sustentam esse ramo e são importantes para o desenvolvimento econômico.

Contudo, apesar dos serviços oferecido pelas agências e guias, o turista quer construir seus pacotes de maneira personalizada que possam incluir tanto passeios, quanto hospedagem de forma atrativa e barata, com opções fora de pacote das agências, como passeios fotográficos nas localidades de visitações.

Na aplicação, o usuário poderá ver as opções de hotéis disponíveis na plataforma da Matrip para fazer reserva ou a montagem de seu pacote, incluindo passeios, hotéis, gastronomia para melhor atender o cliente.

3.2 API REST hospedada no Heroku

Neste tópico é informado como a api está hospedada online, e sendo consumida por um front, e utilizando um banco de dados na nuvem**.**

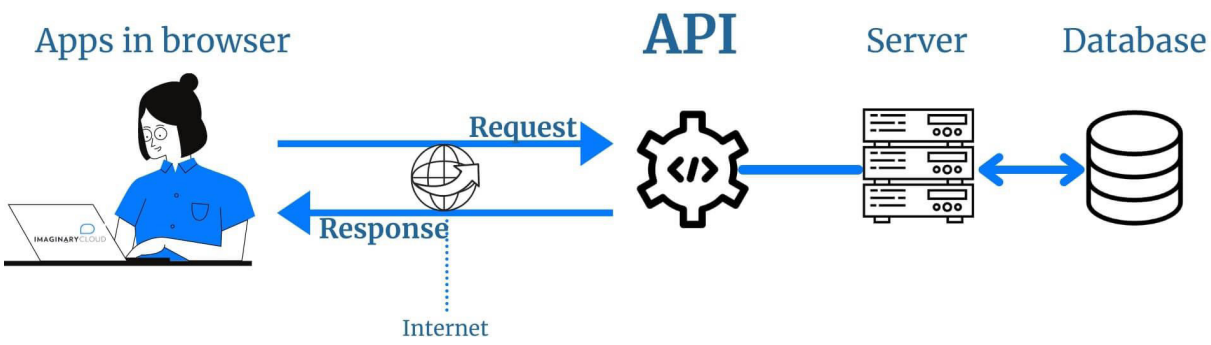


Figura 1. Interação entre cliente, api, servidor e banco de dados

Fonte: Berga e Santos, 2021.

Na figura 1, mostra a API hospedada em um servidor online, o *Heroku*, que também hospeda um banco de dados na nuvem para o comportamento dos dados, e assim permitindo que o front-end consuma seus valores. O *Heroku* permite que o cliente faça requisições através do *response* e *request*, onde o cliente faz uma requisição ao servidor, e o servidor retorna com as respostas dos dados da api.

3.2 Diagrama de caso de uso

O diagrama de caso de uso mostra as ações possíveis de serem feitas no sistema pelo usuário atuando na parte de hotéis da matrip.

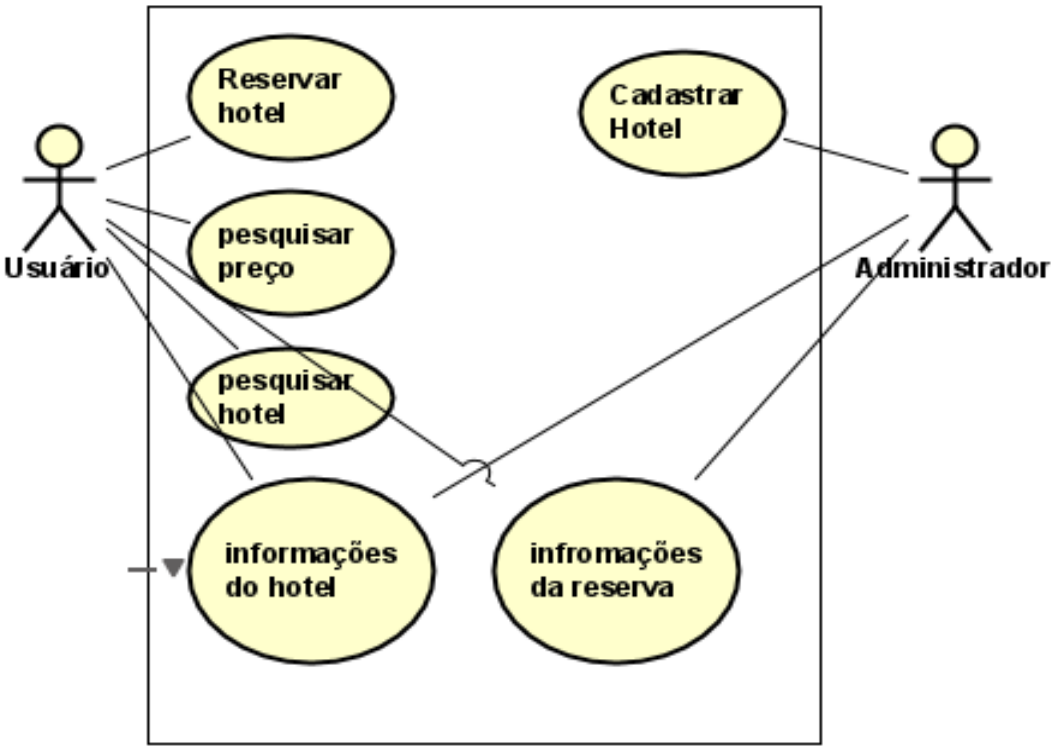


Figura 2. Diagrama de caso de uso criado no Astah

Fonte: Autoral, 2022.

Na figura 2, o diagrama de caso de uso, que representa a parte de hotéis, o usuário pode efetuar uma reserva, pesquisar por um preço de hotel, pesquisar pelo hotel, as informações da reserva que ele efetuou e ter as informações do hotel. O administrador pode cadastrar um novo hotel, editar as informações do hotel e da reserva.

3.2 Diagrama de classe

No diagrama de classe mostra as classes que o sistema compõe, a modelagem dos objetos que compõem o sistema, para exibir os relacionamentos entre os objetos e para descrever o que esses objetos fazem e os serviços que eles fornecem para suas devidas associações.

O diagrama da figura 3 representa como funciona a parte busca por hotéis, que está listado como:

Classe "Categoria": contém informações sobre hotéis e passeios para o cliente escolher. A Classe "Estado": Contém informações de hotéis por estado selecionado. A Classe "Cidade": Contém informações de hotéis por cidade selecionada. Está vinculada a classe estado por uma chave estrangeira. A Classe "Hotels": Contém todas as informações sobre os hotéis cadastrados na Matrip. Possui uma chave estrangeira de cidade, que possui uma de estado. A Classe "Hotel_Precos": Contém informações sobre os preços dos hotéis, que está vinculada com o hotel cadastrado por meio de uma anotação do *Spring* chamada *embedded*. A Classe "Reservar_Hotel": Contém os dados necessários para que o cliente efetue sua reserva de hotel.

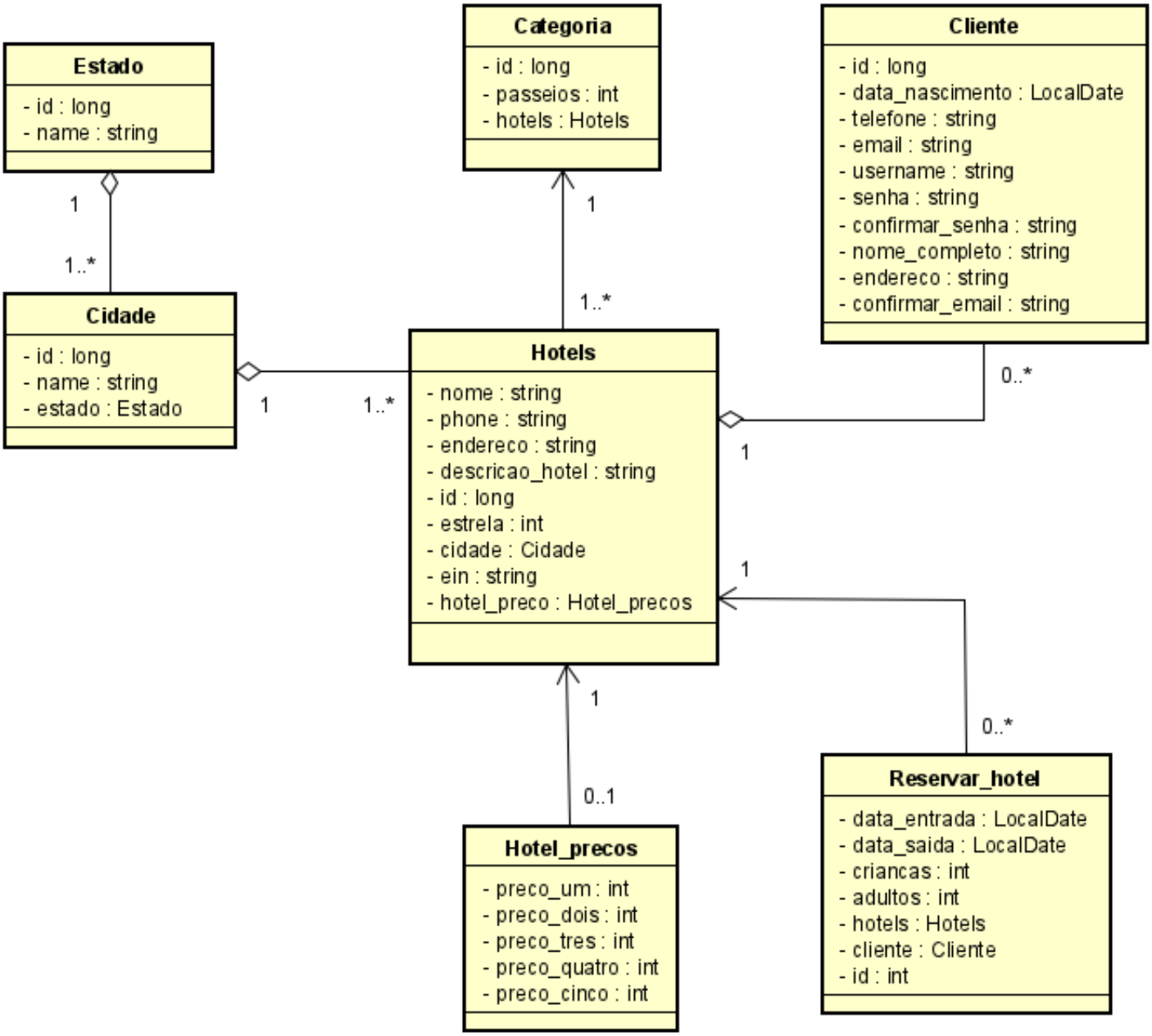


Figura 3. Diagrama de classe criado no Astah

Fonte: Autoral,2022.

Por esse meio, o cliente vai consultar os hotéis que estão cadastrados no banco de dados, seja por nome, cidade, estado ou pela filtragem de preço, e vai escolher o que melhor se encontra próximo aos seus passeios e incluí-los nos pacotes para a realização da compra do mesmo. Também terá a disponibilidade de pacotes já prontos, que disponibilizam Hotéis que já possuem parcerias com determinados passeios, para maior agilização na compra dos pacotes.

3.3 Entity

As entidades são responsáveis pela definição das características de uma classe, onde são definidos os atributos e os relacionamentos entre as entidades do projeto. Brevemente as tabelas do banco de dados são criadas a partir da classe entidade (veja a figura 4).

```java
SamInst
@Entity
@Table (name = "tb_hotels")
public class Hotels {
    5 usages
    @Id
    @GeneratedValue (strategy = GenerationType.SEQUENCE)
    private Long id;

    2 usages
    @ManyToOne
    @JoinColumn(name = "categoria")
    private Categoria categoria;

    2 usages
    @Column(name = "name")
    private String name;
    2 usages
    @Column(name = "employee_identification_number")
    private String ein;
    2 usages
    @Column(name = "address")
    private String Address;

    2 usages
    private String phone;

    2 usages
    @Column(name = "hotel_description")
    private String hotelDescription;

    2 usages
    @ManyToOne
    @JoinColumn(name = "city_id")
    private Cidade city;
```

Figura 4. Descrição da anotação *Entity*

Fonte: Autoral, 2022.

Na figura 4, o mapeamento do objeto-relacional se dá pelas anotações *do Spring Boot*: "*@Entity*" é usado para indicar que a classe representa uma entidade. "*@Table*" descreve o nome da tabela no banco de dados. "*@Column*" é a anotação que define o nome das colunas da tabela. A anotação "*@Id*", indica que a variável privada se trada de um id, e "*@GeneratedValue*" com o valor "sequence" no atributo id, se trata de uma chave primária que vai ser gerada automaticamente pelo banco de dados, com números inteiros, sempre respeitando uma sequência. "@ManyToOne" se trata de uma chave estrangeira com o relacionamento de muito para um, de cidade a qual o hotel está vinculado.

Na figura 5, é criado no banco de dados a tabela e as colunas com os respectivos nomes setados na classe entidade, com suas respectivas chaves primárias e estrangeiras. Os dados acima são apenas ilustrativos.

	id [PK] bigint	address character varying (255)	employee_identification_number character varying (255)	hotel_description character varying (255)
1	1	Hotel Street Test numb...	63.634.901/0001-22	Tv, Wifi, Breakfast
2	2	Hotel Street Test numb...	63.634.901/0001-22	Tv, Wifi, Breakfast
3	3	Hotel Street Test numb...	63.634.901/0001-22	Tv, Wifi, Breakfast
4	4	Hotel Street Test numb...	63.634.901/0001-22	Tv, Wifi, Breakfast
5	5	Hotel Street Test numb...	63.634.901/0001-22	Tv, Wifi, Breakfast

Figura 5. Tabela criada no banco a partir da classe entidade

Fonte: Autoral, 2022.

O Mapeamento Objeto-Relacional (ORM) é uma técnica utilizada para converter dados entre bancos relacionais e linguagens orientadas a objeto. Pode ser usado para acelerar o processo de desenvolvimento eliminando código repetitivo, mapeando campos de resultados de consultas para membros de objetos e vice-versa.

3.4 *Repository*

A camada de entidade é definida pelas das características da classe, é nela que são definidos os atributos, e os relacionamentos entre as entidades do projeto.

Na figura 6, a anotação "@Repository" é uma interface que faz a persistência das classes implementadas. Para que essa persistência seja feita, cada entidade deve ter sua interface de acesso ao banco de dados.

```
11 usages  SamInst
@Repository
public interface HotelRepository extends JpaRepository<Hotels, Long> {
    1 usage  SamInst
    @Query(value = "select u from Hotels u where upper(trim(u.name)) like %?1%")
    List<Hotels> findByName (String name);
    1 usage  SamInst
    List<Hotels>findHotelsByCityId(Long id);
    1 usage  SamInst
    @Query(value = "select u from Hotels u join u.city  where upper(trim(u.city.name)) like %?1%")
    List<Hotels> findHotelsByCity_Name(String name);
```

Figura 6. Descrição da anotação Repository

Fonte: Autoral, 2022.

É implementado também a interface do JPA (Java Persistence API). Ele facilita consultas no banco de dados, podendo ser da maneira do JPA, que é iniciando o nome do método com "find (nome da classe) By (argumento)", ou de maneira nativa utilizando a anotação "@Query" passando a consulta como parâmetro.

3.5 *Controller*

O *Controller* é responsável pelas requisições que são recebidas, e é nela que são im-

plementados os *endpoints* por meio de url's, para que o cliente acesse a API.

A figura 7, mostra o *Controller* utilizando as anotações: @GepMapping, @PostMapping, @PutMapping e @DeleteMapping. O @RestController recebe requisições através dos métodos listados.

```
SamInst
@GetMapping
public List<Hotels> list() { return hotelRepository.findAll(); }

SamInst
@ResponseStatus(HttpStatus.CREATED)
@PostMapping //-----------------------------------------------------------------------------------------
public Hotels add(@RequestBody Hotels hotels) { return hotelRegistrationService.add(hotels); }

SamInst
@PutMapping("/{hotelId}") //----------------------------------------------------------------------------
public Hotels update(@PathVariable Long hotelId, @RequestBody Hotels hotels) {
    Hotels hotels1 = hotelRepository.findById(hotelId).get();
        BeanUtils.copyProperties(hotels, hotels1, ...ignoreProperties: "id","city","ein","categoria");
       return hotelRepository.save(hotels1);
    }
SamInst
@DeleteMapping("/{hotelId}") //-------------------------------------------------------------------------
public ResponseEntity<Hotels> remove(@PathVariable Long hotelId) {
    try {
        hotelRegistrationService.exclude(hotelId);
        return noContent().build();
    } catch (EntityNotFound e) {
        return notFound().build();
    } catch (EntityInUse e) {
```

Figura 7. Controller utilizando as anotações

Fonte: Autoral, 2022.

A anotação "*@RestController*" é aplicada a uma classe para marcá-la como um controlador de uri's, ela receberá requisições externas através de métodos HTTP (GET, POST, DELETE, PATCH, PUT). Já a anotação "*@RequestMapping*" é usada para mapear solicitações da Web para métodos do *Spring Controller*. Cada método possui sua própria URI.

4. RESULTADOS

Neste capítulo serão apresentadas as principais funcionalidades do sistema de Hotéis, que será utilizado em uma plataforma online e no aplicativo mobile da Matrip, como o processo de pesquisa de hotel, pesquisa por nome e cidade, cadastro dos hotéis etc.

Será demonstrado como funciona os *endpoints* de cadastro de hotel, os filtros de pesquisa que foram implementados e como utilizá-los.

4.1 Testes

O teste da API é uma maneira de avaliar a qualidade da aplicação e reduzir o risco de falha em operação. Para a realização dos testes na *API Rest* foi usado *o Postman*. Esta fase consistirá em testar vários casos nos atores do serviço web como o cadastro dos hotéis por um administrador no tópico 4.1.1, as informações do hotel no tópico 4.1.2, a bus-

ca pelo nome do hotel no tópico 4.1.3, o cliente efetuando uma reserva no tópico 4.1.4.

4.1.1 Cadastro do Hotel

O cadastro do Hotel vai ser realizado pelo administrador do sistema, viabilizando as regras de negócio, com as informações do hotel e também para definir a categoria que o hotel vai pertencer, se vai pertencer ao campo de destaques.

Na figura 8, o "id" é atribuído automaticamente, a categoria terá valor 1 para hotéis, e informações complementares como nome, CNPJ, telefone, descrição do hotel, a cidade a qual está vinculado, os preços entre 1 e 5 pessoas, a quantidade de quartos, para atualizações futuras.

```
{
    "categoria": {
        "id": 0
    },
    "name": "String ",
    "ein": "String",
    "phone": "String",
    "hotelDescription": "String",
    "city": {
        "id": 0
    },
    "hotelPrices": {
        "priceOne": 0,
        "priceTwo": 0,
        "priceThree": 0,
        "priceFour": 0,
        "priceFive": 0
    },
    "rooms": {
        "numberRooms": 0
    },
    "star": 0,
    "address": "String ",
    "destaque": 0
}
```

Figura 8. Cadastramento de hotel

Fonte: Autoral, 2022.

O sistema fazer a separação pela quantidade de pessoas em cada quarto, a qualidade em estrelas de nível 1 a 5, o endereço, e destaque tendo valor entre 0 e 1, onde 0 significa que o hotel não entra na categoria de destaques, e 1, entra na categoria destaques e são exibidos.

4.1.2 Informações do Hotel

No site serão exibidas as informações do hotel, como descrição, localização, telefone, a qualidade em estrelas de nível 1 a 5, e preços, de acordo com a quantidade de pessoas.

Na figura 9, é mostrado um exemplo de todos os campos que precisarão ser fornecidos pelo proprietário do hotel, para o administrador poder cadastrar os hotéis na matrip. Os dados são ilustrativos para exemplificar.


```
{
    "name": "Pousada 1 ",
    "ein": "63.634.901/0001-22",
    "phone": "(98)984508897",
    "hotelDescription": "Tv, Wifi, Breakfast",
    "city": {
        "city": "Barreirinhas",
        "state": "Maranhão"
    },
    "address": "Hotel Street Test number one ",
    "daily": {
        "one_Person": 120.0,
        "two_Persons": 200.0,
        "three_Persons": 350.0,
        "four_Persons": 420.0,
        "five_Persons": 500.0
    },
    "star": 4
}
```


Figura 9. Exemplo das informações do hotel

Fonte: Autoral, 2022.

O Json retorna os valores, do nome, CNPJ, número de telefone, a descrição do hotel, a cidade a qual o hotel está vinculado, o seu endereço, os preços das diárias, variando entre 5 pessoas (brevemente poderão ser adicionados mais preços por pessoa, depende do hotel e quarto que vão ser cadastrados na Matrip), e a qualidade que vai ser definida pelos usuários em níveis de estrela de 1 a 5. Esses dados vão ser exibidos pelo front, da melhor maneira para que o cliente seja atraído a consumir o produto.

4.1.3 Pesquisa de Hotel por nome

Um dos principais campos de pesquisa é a pesquisa por nome do Hotel, assim o cliente pode conseguir localizar o hotel desejado mais rápido.

Na Figura 10, o Postman mostra a consulta que foi feita, usando como chave de pesquisa “name”, e retornou todos os valores do hotel com o nome informado, como: localização (cidade e estado), as informações do estabelecimento, o endereço cadastrado, o preço da diária e a qualidade prestada pelo hotel em níveis de estrela entre 1 a 5.

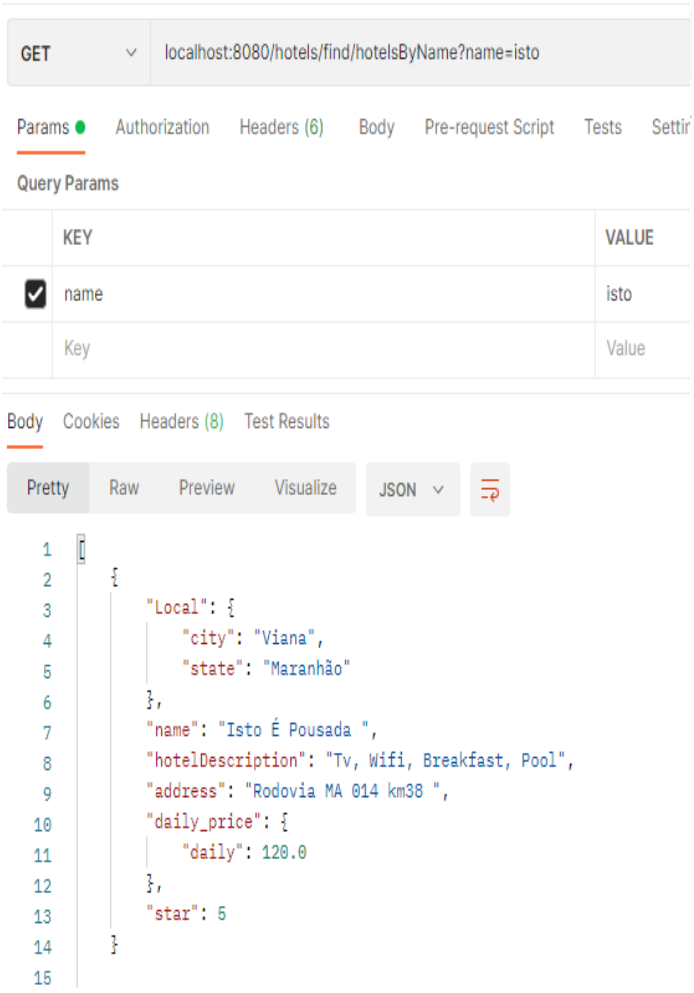


Figura 10. Resposta do Postman em JSON para pesquisas por nome de hotel

Fonte: Autoral, 2022.

O Json retorna o hotel cadastrado no banco de dados com o nome "isto", e exibe as informações da localização, descrição, endereço, a qualidade do hotel e de acordo com a quantidade de pessoas, ele exibe o valor da diária.

4.1.4 Reservar um Hotel

Na reserva de hotel, o cliente realiza a escolha do hotel, que será setado pelo id, no qual, seus dados são pegos do id do cliente no banco de dados, o usuário informa a data de entrada e a data de saída, assim como, a quantidade de adultos e crianças que irão hospedar-se.

Na figura 11, os valores do "id" do hotel e "id" do cliente, precisam ser setados, também a data de entrada e de saída, a quantidade de adultos e crianças, pois a api precisa de todos esses campos para realizar a consulta adequadamente.

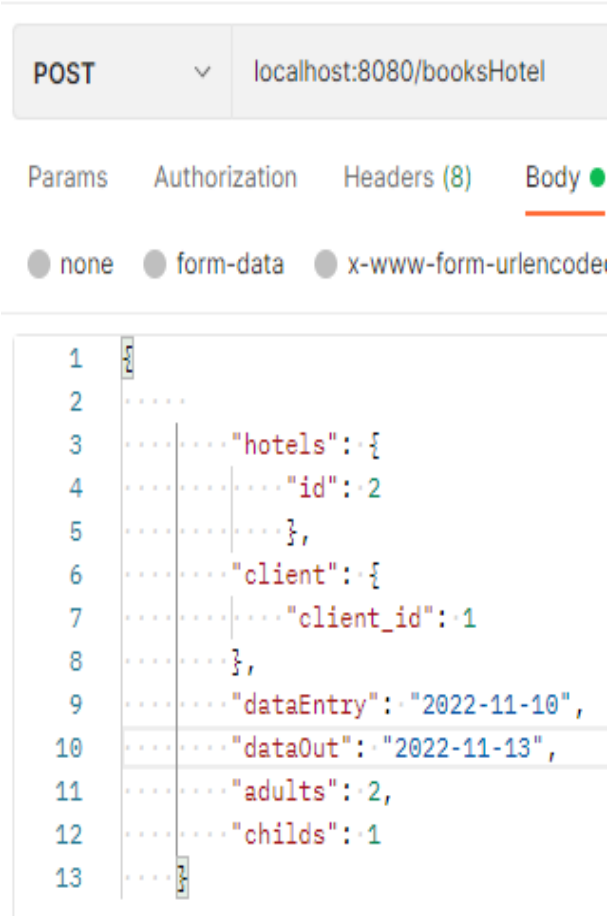


Figura 11. Postman demonstrando a efetuação de uma reserva

Fonte: Autoral, 2022.

Ao inserir os valores do "id" do hotel e "id" do cliente, o banco de dados retorna automaticamente os dados do id fornecido e ao efetuar uma reserva, o cliente apenas insere a data de entrada e a de saída, a quantidade de adultos e crianças, e o Postman já retorna a reserva com os valores cadastrados no banco.

```
{
    "hotelName": "Pousada 2 ",
    "city": {
        "cityName": "São Luís",
        "stateName": "Maranhão"
    },
    "client": {
        "name": "saaaam",
        "cpf": "080.362.532-08",
        "phone": "(98)98452-8497",
        "email": "sam11@gmail.com",
        "address": "Street test number eleven",
        "dataEntry": "2022-11-10",
        "dataOut": "2022-11-13"
    },
    "amount_adults": 2,
    "amount_childs": 1,
    "prices": {
        "days": 3.0,
        "daily": 180.0,
        "total_price": 540.0
    }
}
```


Figura 12. Informações da reserva do hotel

Fonte: Autoral, 2022.

O Postman retorna um JSON com as informações da reserva, os dados do hotel, dados do cliente, a quantidade de pessoas, e nos preços, informa a quantidade de dias, a diária já fica pré-definida pela quantidade de pessoas, e faz a multiplicação pela quantidade de dias, gerando o valor total.

4.2 Documentação

O Swagger é o framework open source utilizado para o desenvolvimento nos processos de definir, criar, documentar e consumir a API. Ele padroniza este tipo de integração, descreve os recursos que uma API possui, endpoints, dados recebidos, dados retornados, código HTTP e métodos de autenticação.

```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-ui</artifactId>
    <version>1.6.9</version>
</dependency>
```

Figura 13. Dependência do Swagger.

Fonte: Autoral, 2022.

A hospedagem do site se encontra no Heroku, uma plataforma de escalonamento e gerenciamento de aplicativos. Ela suporta diversas linguagens, incluindo a Java que foi utilizada neste projeto. O Heroku executa aplicativos por meio de contêineres virtuais conhecidos como Dynos. O Banco de Dados também é hospedado no Heroku, assim facilitando para que o Front-end consuma valores já cadastrados da API.

A API do Hotel está hospedada no Heroku. Na figura 14, monstra as url's que a pesquisa de hotéis vai consumir:

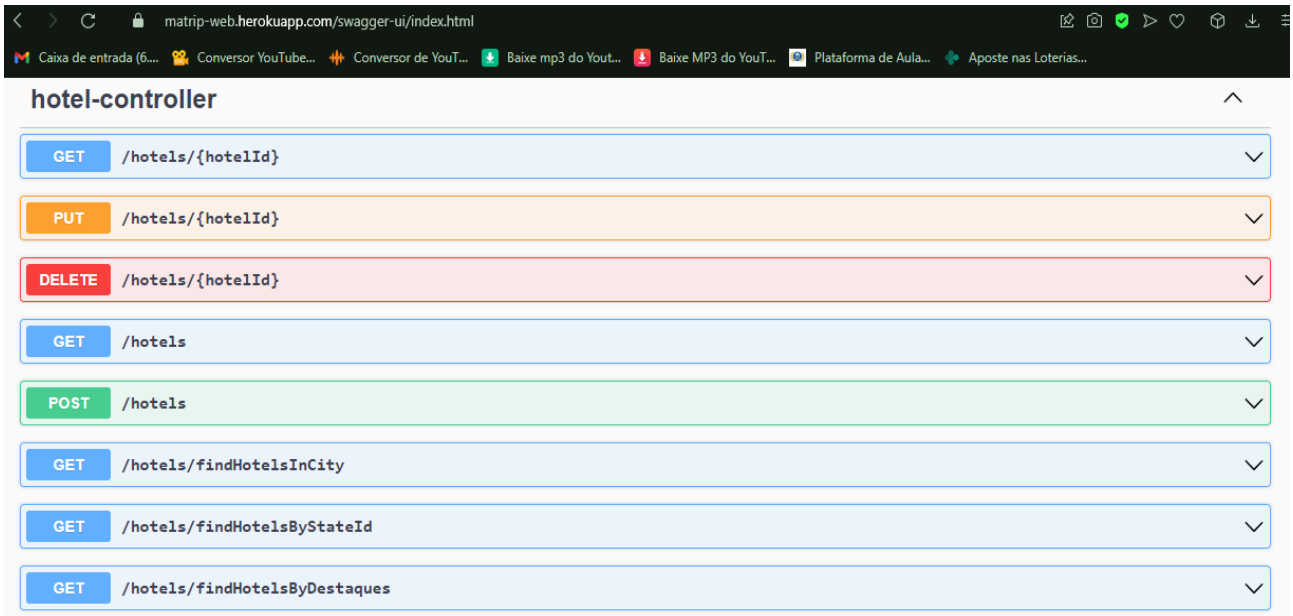


Figura 14. Front consumindo o banco de dados

Fonte: Autoral, 2022.

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 serviços da API.

No projeto foi utilizado a dependência do Swagger UI para documentar as url's criadas do controller.

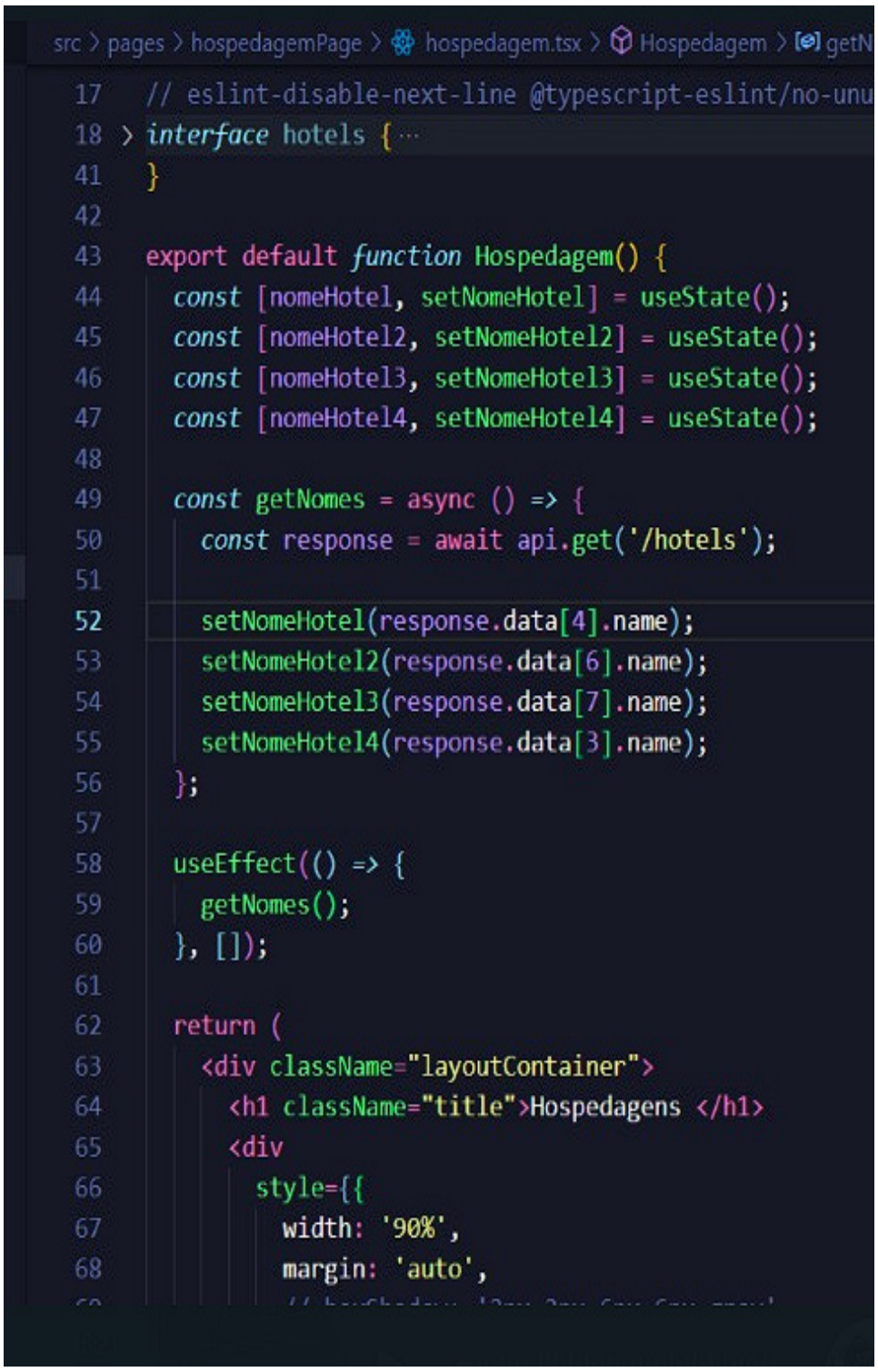


Figura 15. Documentação da API

Fonte: Autoral, 2022.

No código da figura 15, mostra como são consumidos os dados que estão armazenados no banco no Heroku para serem exibidos nas telas.

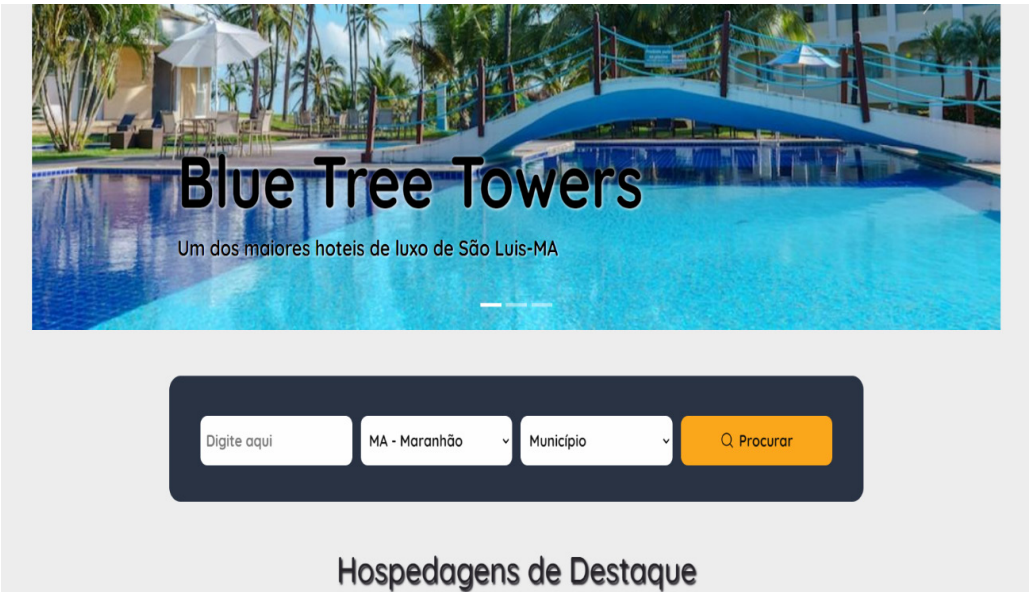


Figura 16. API sendo consumida pelo front

Fonte: Autoral,2022.

Na figura 16, há um campo de pesquisa que pode ser utilizado para procurar hotéis pelo nome, estado ou cidade, assim garantindo mais facilidade para o cliente encontrar o hotel desejado.

4.3 Resultados e Discussões

Ao final do desenvolvimento do projeto, foi obtido a primeiras versões da api de hotel da matrip. A IDE IntelliJ, junto ao *framework Spring* possui um ambiente integrado que permite testar a API. As anotações "@*Test"*, foram utilizadas para teste de cadastro e resposta de informações, e o IntelliJ possui um ambiente de *datasource*, para gerenciamento sem precisar acessar diretamente o banco de dados. A partir disso, conseguimos simular o uso do sistema como se estive cadastrado e pesquisando hotéis de verdade.

As simulações foram feitas durante dias para testar o comportamento do sistema. Ao criar um projeto com o Spring, automaticamente é feito uma pasta para a realização dos testes. Na figura 17, foi utilizando a anotação do Spring "@Test", que permite cadastrar informações dos hotéis que já existem, porém, ainda não cadastrados na matrip, também as consultas por nome de hotel e pesquisa por preço, também gerenciando informações sobre os mesmos.

```
SamInst *
@SpringBootTest
class HotelApplicationTests {
    SamInst
    @Test
    void contextLoads() {
    }
    new *
    @Test
    public void addHotel(){
        Hotels hotels = new Hotels();
        hotels.setId(1L);
         hotels.setName("Hotels Test");
         hotels.setHotelDescription("Description Test");
    }
}
```

Figura 17. Ambiente de teste do IntelliJ

Fonte: Autoral, 2022.

As tabelas foram gerenciadas pelo *datasource* do IntelliJ, que atua diretamente no console. Na figura 18, são criadas tabelas de modo mais rápido e eficiente, pois ele possui a ferramenta de autocompletar os dados inseridos junto às tabelas já existentes, facilitando muito o trabalho com o banco de dados, também permitindo a inserção e remoção de dados nas tabelas para testes futuros.

```
console ×
Tx: Auto
1  insert into tb_states (id, state_name) VALUES (1, 'Maranhão');
2  insert into tb_states (id, state_name) VALUES (2, 'Ceará');
3
4  insert into tb_city (name_city,state_id) values ('Barreirinhas', 1);
5  insert into tb_city (name_city,state_id) values ('São Luís', 1);
6  insert into tb_city (name_city,state_id) values ('Viana', 1);
```

Figura 18. Console do IntelliJ

Fonte: Autoral, 2022.

. Como resultado o funcionamento da API apresentou algumas falhas no começo, porém, posteriormente o sistema retornou todos os valores corretamente de acordo com cada função chamada, como por exemplo, a busca por nome, ou preço preferido pelo cliente. Assim se mostrou eficiente no seu propósito, podendo ser testado na plataforma Heroku, o qual está hospedado para cadastrar hotéis, pesquisar preços, fazer e obter as informações da reserva, tal como o comportamento do sistema se encontra funcional e operante.

5. CONCLUSÃO

Diante do exposto, o intuito deste projeto é fazer com que as pessoas busquem os respectivos Hotéis, que se encontram na melhor localização para a execução dos passeios e para a inclusão dos mesmos nos pacotes de viagens, pois elas têm a possibilidade de comprar pacotes de passeios junto com os pacotes de hotéis da região maranhense, que são feitos por empresas de turismo em conjunto com empresas de Hotéis e o cliente, que compra estes pacotes, além de fazer o trabalho de divulgar o turismo, os Hotéis e a cultura do estado do maranhão.

Isso tudo feito através de uma plataforma web, que será desenvolvida com as funcionalidades de: segurança; possibilidade do usuário de criar um pacote de passeio e hospedagem personalizado, incluindo horários, dias e serviços; parcerias com empresas de hotéis, com comissão para a equipe da Matrip para cada pacote vendido.

Referências

ALLAMARAJU, Subbu. **RESTful Web Services Cookbook: Solutions for Improving Scalability and Simplicity**, 2010.

BERGA, Mariana; SANTOS, André. **gRPC vs REST: comparando estilos de arquitetura de APIs**. Disponível em <https://www.imaginarycloud.com/blog/grpc-vs-rest/>. Publicado em jun/2021. Acessado em nov/2022

CLARK, Jessica. **O que é o Heroku?** Back4App, 2022. Disponível:https://blog.back4app.com/pt/o-que-e--o-heroku/. Acesso em 02/11/2022.

COSTA, Caio. **Spring Boot: Microsserviços na prática**, 2021.

JACOBSON, Daniel; BRAIL, Greg; WOODS, Dan**. APIs: A Strategy Guide: Creating Channels with Application Programming Interfaces,** 2012.

LEITE, Thiago. **Orientação a objeto: aprenda seus conceitos e suas aplicabilidades de forma efetiva**, 2016.

M.F., Ribeiro. **Web Services REST: Conceitos, análise e implementação**, 18 f. Artigo. Instituto Federal Goiano, 2016

MASSE, Mark. **REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces,** 2011.

PONELAT, Joshua S**;** ROSENSTOCK, Lukas L**. Designing APIs with Swagger and Openapi**

RED HAT. **API REST**. Red Hat, 2020. Disponível em: https://www.redhat.com/pt-br/topics/api/what-is-a--rest-api. Acesso em 25/10/22.

RGV HOTELARIA. Hotelaria como alavanca para o desenvolvimento econômico local e regional. **RGV Hotelaria, 2022. Disponível em:** https://rgvhotelaria.com.br/investimento/hotelaria-como-alavanca-para--o-desenvolvimento-economico-local-e regional/#:~:text=De%20acordo%270com%20a%20pesquisa,como%20ao%20consumir%20bens%20industriais. Acesso em: 02/11/2022.

ROTA COMBO. **Os 7 principais pontos turísticos no Maranhão em 2020.** Rota Combo Blog, 2020. Disponível em: https://rotacombo.com/blog/principais-pontos-turisticos maranhao/?utm_source=google&utm_medium=cpc&utm_campaign=pareto.in. gsn. dsads. %5bPiaui_ceara_Maranhao%5d&gclid=-Cj0KCQjwhsmaBhCvARIsAIbEbH46hBAlAI8PsV41_SSsD_SVc4iQl0bEBzRZxR9UsSipOT2py2wiknYaAoQ-EALw_wcB. Acesso em: 22/10/2022.

SANTOS, Éder. **Endpoints: o que é, exemplos e importância da segurança de endpoints**. Disponível em: https://yssy.com.br/update/artigos/endpoints/ Acesso em 28/10/22

SAUDATE, Alexandre**. *REST: Construa APIs inteligentes de maneira simples,*** 2013.

VARANASI, Balaji. **Introducing Maven: A Build Tool for Today's Java Developers**, 2019.

VIEIRA, Ronaldo. Artigo. **World Wide Web: Terra Encantada onde tudo se encontra?** Educação, Ges-

tão e Sociedade: revista da Faculdade Eça de Queirós, ISSN 2179-9636, Ano 4, novembro de 2014.

ZUCHER, Vitor. **O que é padrão MVC? Entenda arquitetura de softwares! Le Wagon**, 2020. Disponível em: <https://www.lewagon.com/pt-BR/blog/o-que-e-padrao-mvc>. Acesso em: 23/10/22.

2

DESENVOLVIMENTO DE UM APP E O USO DE IOT COM RFID PARA DEMANDA EM ESCOLA DE TEMPO INTEGRAL COM BAIXO CUSTO

DEVELOPMENT OF AN APP AND THE USE OF IOT WITH RFID FOR DEMAND IN A FULL-TIME SCHOOL AT LOW COST

Afonso Jansen de Mello Farias[1]

Will Ribamar Mendes Almeida[2]

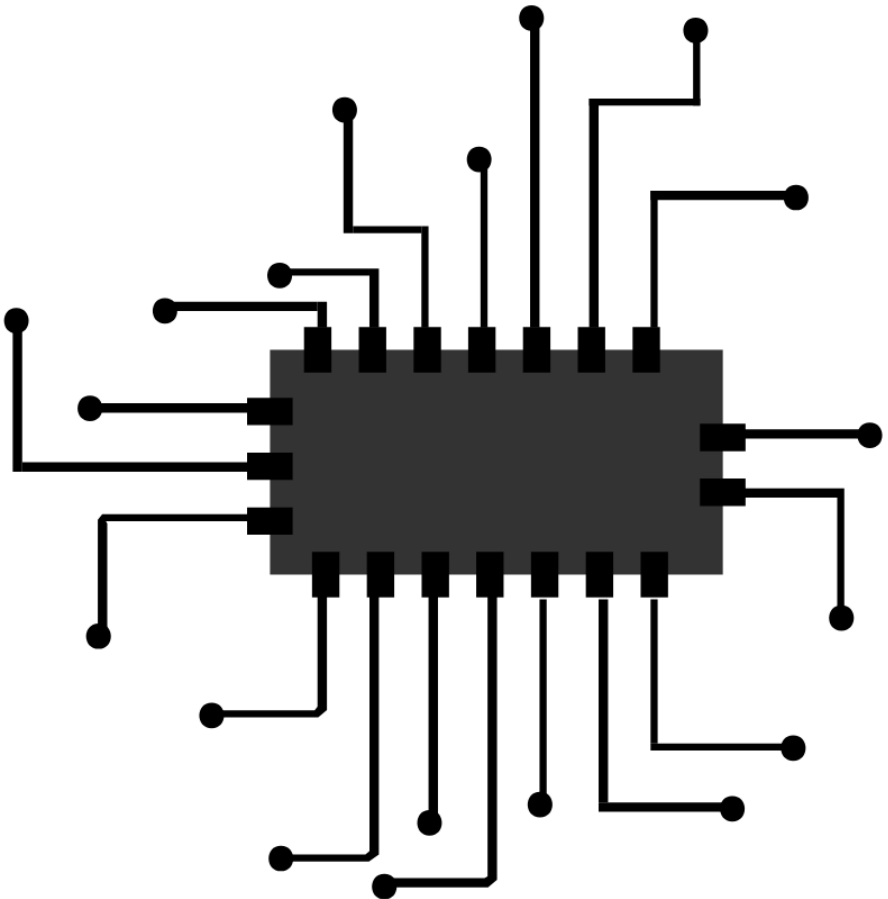
Edilson Carlos Silva Lima[3]

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

{ FARIAS, Afonso Jansen de Mello, afonsojansen@gmail.com; ALMEIDA, Will Ribamar Mendes, will75@gmail.com; LIMA, Edilson Carlos Silva, edilsonlima3@gmail.com.}



d.o.i.: 10.29327/5192626.1-2

Resumo

O uso de radiofrequência para automatizar processos está presente em nossa vida. Na conexão Wi-Fi, cartões de crédito, cartão de passagem, controle de acesso, controle remoto, alarmes entre outros. No colégio em questão, por ser um colégio integral, detectou-se a necessidade de uma solução buscando auxiliar no preparo diário das diversas refeições que ocorrem na instituição. Há uma variação gritante na quantidade do preparo diário das refeições e ele também muda entre as refeições o que leva muitas vezes ao desperdício ou falta no preparo destas refeições. Buscando uma solução de baixo custo partiu-se do pressuposto em que a grande maioria dos alunos utilizam transporte público ou possuir algum cartão em posse que utiliza a tecnologia *RFiD*, parte do material necessário para que o controle ocorra já se encontra em posse dos envolvidos, mesmo que venha a ser necessário a aquisição de alguns cartões, eles possuem baixo custo. Usando um microcontrolador Raspberry Pi, Leitor de *RFiD*, display, BD MARIADB e Wi-Fi, será implementado um algoritmo que possibilite a leitura e armazenamento dos dados dos alunos associando aos seus respectivos identificadores *RFiD* sejam eles quais forem. Desta forma, fica disponibilizado um totem com o equipamento em um local comum e de fácil acesso aos alunos para que estes possam registrar sua inclusão na lista da refeição oferecida pelo refeitório desta instituição. Solucionando o problema em pouco tempo com o mínimo de treinamento necessário para sua utilização.

Palavras-chave: *RFiD, Tags, Raspberry, BD, APP, React Native.*

Abstract

The use of radiofrequency to automate processes is present in our lives. In the Wi-Fi connection, credit cards, ticket card, access control, remote control, alarms among others. In the IEMA school, as it is an integral school, it was detected the need for a solution seeking to assist in the daily preparation of the various meals that occur in the institution. There is a glaring variation in the amount of daily meal preparation and it also changes between meals, which often leads to waste or lack of preparation of these meals. Looking for a low-cost solution for all those involved, where I assumed that the vast majority of students use public transport or have a card in possession that uses RFID technology, part of the material necessary for the control to occur is already in possession of those involved, even if it becomes necessary to purchase some cards, they have a low cost. Using a Raspberry Pi microcontroller, RFiD Reader, display, MARIADB DB and Wi-Fi, an algorithm will be implemented that allows the reading and storage of student data associating their respective RFiD identifiers, whatever they may be. In this way, a totem with the equipment is made available in a common passageway for students so that they can register that they want to be included in the meal list offered by the cafeteria. Solving the problem in a short time with the minimum training necessary for its use.

Keywords: RFiD, Tags, Raspberry, BD, APP, React Native.

1. INTRODUÇÃO

Em instituições de ensino públicas de período integral, é comum sofrer atraso e ou falta no fornecimento das refeições periódicas guarnecidas pela instituição. O mesmo se dá por falta de controle da demanda a ser atendida em cada período. Por vezes tentando evitar a recorrência deste problema, costumam produzir uma quantidade maior por segurança, trazendo outro problema, o desperdício. Qualquer um dos casos é de fundamental importância ser evitado, seja por motivos financeiros, fisiológicos, sociais ou educacionais.

Após compreender a situação junto à diretoria institucional, elaborou-se um plano de controle daqueles que usufruem deste benefício oferecido na instituição, de forma automatizada, isentando os responsáveis pela produção das refeições, tenham de interagir diretamente na contagem.

A solução empregada utiliza tecnologia *IoT*, agregando componentes como Raspberry Pi 3B, leitora de *RFiD*, banco de dados, cartão de memória e aplicativo mobile, entre outros itens essenciais para o funcionamento da solução encontrada que será abordada com mais detalhes nos tópicos a seguir.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, ocorrerá uma revisão literária dos assuntos essenciais para o desenvolvimento de um sistema embarcado e um aplicativo desenvolvido em *React Native* e está dividido nos seguintes itens: *IoT*, Raspberry Pi 3B, Leitora *RFiD* 2.4 o *Banco de Dados*, o Cartão SD e *API*.

2.1 IoT

Seja em casa, no trabalho, nos meios de transportes ou entre os mais diversos e improváveis lugares existe *IoT* (*Internet of Things*), mesmo que não perceba. Uma forma de definição segundo a empresa Oracle (2022):

> "A Internet das Coisas (IoT) descreve a rede de objetos físicos incorporados a sensores, software e outras tecnologias com o objetivo de conectar e trocar dados com outros dispositivos e sistemas pela internet. Esses dispositivos variam de objetos domésticos comuns a ferramentas industriais sofisticadas".

O *IoT* tem o seu funcionamento partindo da coleta de dados e como será usado, para a Amazon Web Services (2022):

> "Um sistema IoT tem três componentes: Dispositivos inteligentes, aplicação de IoT e uma interface gráfica do usuário".

É quase impossível imaginar uma casa ou carro nos tempos atuais que não tenha nada de *IoT*. Sua sigla tem origem do inglês, *Internet of Things* (Internet das Coisas).

2.2 Raspberry Pi 3B

A definição de um Raspberry Pi 3B segundo Gonçalves (2018):

> "Raspberry Pi é um computador de baixo custo e que tem aproximadamente o tamanho de um cartão de crédito. Foi desenvolvido no Reino Unido pela Fundação Raspberry Pi. Para usá-lo, basta plugar um teclado e um mouse padrão a ele e conectar tudo isso a um monitor ou a uma televisão com entrada HDMI. Os modelos custam entre US$ 25 e US$ 35, é claro que aqui no brasil o valor é um pouco mais salgado (tanto para importar como para comprar diretamente aqui), mas ainda assim vale a pena".

Um microcomputador portátil de baixo custo, se destaca por possuir as principais características que encontramos em um notebook. Equipado com processador Quad Core 1.2GHz Broadcom BCM2837 64bit, Memória RAM 1GB, porta ethernet, hdmi, wifi, bluetooth, 4 portas USB e o seu grande diferencial, 40 pinos GPIO por onde podemos integrar diversos módulos, sensores entre outros periféricos. Possui sistema operacional próprio o Raspberry Pi OS baseados em Linux.

2.3 Leitora *RFiD*

Módulo de Leitora *RFiD* é capaz de ler diversos modelos de cartões *RFiD* trabalha dentro da frequência de operação 13.56MHz e tensão 3,3 volts, é um modulo versátil compatível com diversos microcontroladores. Para a empresa Patrimônio E Avaliações (2021), um leitor de identificação por radiofrequência (leitor de *RFID*) é um dispositivo usado para coletar informações de uma etiqueta *RFID*, que é usada para rastrear ativos. As ondas de rádio são usadas para transferir dados da etiqueta para um leitor. A tecnologia *RFID* é semelhante em teoria aos códigos de barras. No entanto, a etiqueta *RFID* não precisa ser lida diretamente, nem exige linha de visão para o leitor: para ser lida, ela precisa estar dentro do alcance de um leitor *RFID*, que varia de 90 a 900 centímetros.

Os tipos de leitores se subdividem quanto a sua mobilidade, podendo ser fixos, integrados e móveis. Quantidade de antenas, frequência e sua conectividade. O tipo abordado neste artigo será um modulo fixo de antena interna, com conectividade via pinos e trabalha na frequência de 13.56Mhz.

2.4 Banco de Dados

Um banco tem como sua principal função o armazenamento de dados, de forma que possa ser consultado estes dados trazendo informações que sejam pertinentes ao seu uso, segundo Machado (2020), Banco de dados pode ser definido como um conjunto de dados devidamente relacionados. Dados são os objetos conhecidos que podem ser armazenados e que possuem um significado implícito, porém o significado do termo banco de dados é mais amplo do que simplesmente a definição dada anteriormente. Um banco de dados possui as seguintes propriedades: É uma coleção lógica coerente de dados com um significado inerente; uma disposição desordenada dos dados não pode ser referenciada como banco de dados. Ele é projetado, construído e preenchido com valores de dados para um propósito específico; um banco de dados possui um conjunto predefinido de usuários e de aplicações. Ele representa algum aspecto do mundo real, o qual é chamado de minimundo; qualquer alteração efetuada no minimundo é automaticamente refletida no

banco de dados.

Exemplificando, uma empresa possui funcionários e clientes. Deverá haver uma tabela "pessoa", uma "funcionario" e uma "cliente". Um cliente possui sua chave primaria e recebe uma chave estrangeira que liga ele à tabela pessoa, a tabela "funcionario", também possui uma chave primaria e uma chave estrangeira que referência a ele na tabela "pessoa". Dessa forma, economiza espaço na tabela já que "cliente" e "funcionario" são pessoas e tem dados em comum para ambos. Assim gerando um armazenamento conciso e sem dificuldades de encontrar o que pertence a quem.

2.5 Cartão SD

Cartões SD estão presentes em celulares, computadores, câmeras fotográficas, filmadoras entre outros equipamentos como forma de armazenamento. Este tipo de mídia de armazenamento ganhou força no fim dos anos 90 quando começou a se popularizar devido ao surgimento das câmeras digitais e mp3 players. para a SD Association (2022), o padrão SD é utilizado em vários segmentos de mercado da indústria de armazenamento portátil, incluindo telefones celulares, câmeras digitais, MP3 players, computadores pessoais, tablets, impressoras, sistemas de navegação para carros, livros eletrônicos e muitos outros dispositivos eletrônicos de consumo.

O cartão SD foi passando por diversas evoluções em seu formato, velocidade e capacidade. Hoje é um item essencial em qualquer tipo de equipamento portátil que use algum armazenamento.

2.6 API

Segundo Amazon (2022), API significa Application Programming Interface (Interface de Programação de Aplicação). A palavra aplicação refere-se a qualquer software com uma função distinta. Sua interface pode ser pensada como um contrato de serviço entre duas aplicações. Esse contrato define como as duas se comunicam usando solicitações e respostas conforme referenciado na figura 1.

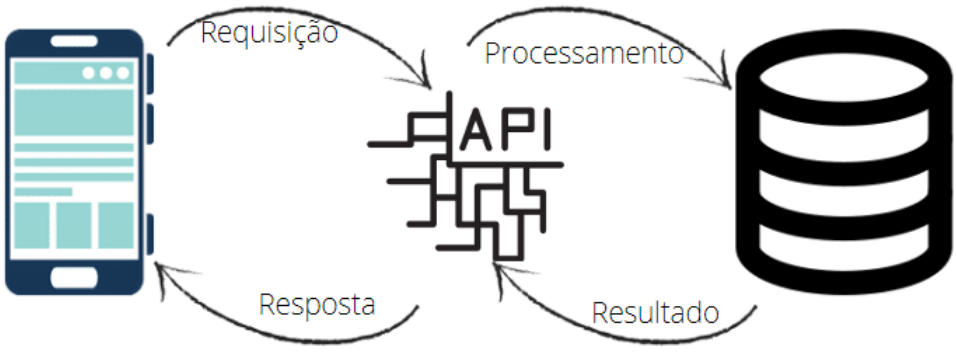


Figura 1. Representação de um API.

Fonte: Autoral, 2022.

A figura 1 detalha o esquema de funcionamento de uma API, podemos imaginá-la como intermediador entre sistemas, ela fornece métodos de comunicação que abstraem o funcionamento dos sistemas que necessitam de comunicação, cria métodos de acesso e manipulação dos dados por meio de requisições e respostas que passam por ela.

3. DESENVOLVIMENTO

Utilizando-se da metodologia pesquisa-ação, onde observamos o problema da instituição, foi identificada a falha no controle quantitativo durante a produção dos alimentos por falta de informação para que a equipe trabalhasse com dados mais precisos. Visto que a instituição não disponibiliza de recursos financeiros, chegou-se à conclusão de que deveria achar uma solução reutilizando o que já tinha e obtendo o mínimo de equipamento, para não elevar o seu custo. Com isso foi idealizado o uso de Raspberry pi 3b que substitui perfeitamente um desktop de entrada e tem 1/5 do seu custo e a reutilização da coisa mais comum entre todos os alunos e funcionários em uma escola pública, as suas respectivas carteiras de ônibus. Agregando um banco de dados no Raspberry, com um modulo leitor de *RFiD* e um software para exibir o quantitativo de leituras de cartões.

Nesta seção, vamos abordar uma revisão literária dos assuntos essenciais para o desenvolvimento de um sistema embarcado e um aplicativo desenvolvido em *React Native* e está dividido nos seguintes itens: 3.1 Idealizando a solução, no item 3.2 Definindo o Hardware e o Software, item 3.3 Raspberry Pi 3B, 3.4 Módulo *RFID* RC522 13,56MHz, no item 3.5 o Cartão de Memória MicroSD 8GB, no item 3.6 temos Jumpers Fêmea-Fêmea e no item 3.7 temos Aplicação React Native no item 3.8 temos Análise da Aplicação no item 3.9 temos o Banco de dados MariaDB e no item 3.10 temos Desenvolvendo o API.

3.1 Resolução

O botão Cadastro, como demonstrado na figura 9, dá acesso à tela onde pode ser realizado consulta das pessoas já cadastradas no banco de dados. Na tela a seguir pode ser realizado consultas, alterar os dados e nível de usuários, este botão é de uso exclusivo para o usuário de nível administrador, demonstrado na figura 2.

Figura 2. Tela manutenção de cadastros.

Fonte: Autoral, 2022

Com base no problema encontrado, foi proposto um método de contagem de pessoas que deverão usufruir do refeitório. Sabe-se que o recurso financeiro destas instituições é limitado, bem como dos alunos. Com isso em mente foi idealizado o uso de cartões *RFiD* já que são presentes nas carteiras de meia passagem de ônibus e todo estudante tem o direito de adquirir sem custos a sua devida 1ª via da carteira.

3.2 Definindo o Hardware e o Software

Pensando no melhor custo/benefício possível, optou-se por usar os seguintes componentes: Raspberry Pi 3B, modulo de leitor *RFiD*, cartão de memória MicroSD 8gb, 7 jumpers fêmea-fêmea, fonte micro usb 5V 2,1A.

O software foi desenvolvido em React Native, as API's foram criadas em Python e por fim o banco de dados em MariaDB.

3.3 Raspberry Pi 3B

Para o projeto foi escolhido o Raspberry Pi 3B, devido a suas capacidades computacionais e compatibilidade com diversos módulos populares no mercado, fazem dele um computador completo, essencial na automação que pode ser visto na figura 3.

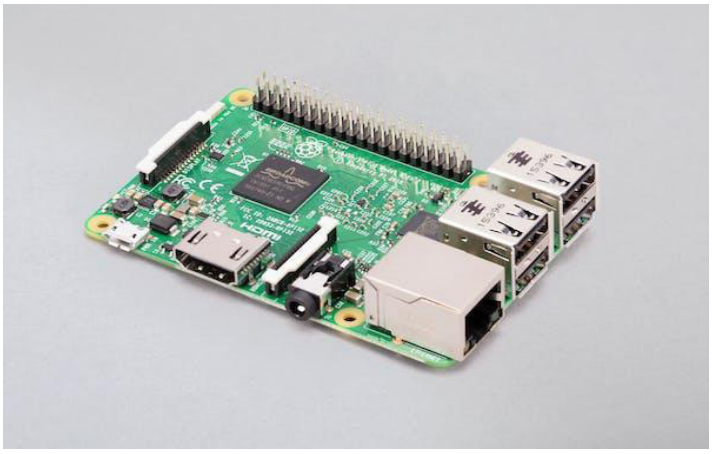


Figura 3. Raspberry Pi 3B.

Fonte: https://www.raspberrypi.com/products/raspberry-pi-3-model-b/

Diferente de microcontroladores "concorrentes" como Arduino e Esp32 ele possui seu próprio sistema operacional baseado em Linux, que tornou possível armazenar localmente todo o banco de dados e API, já que a instituição não possui internet, inviabilizando o armazenamento em nuvem nesta solução. A conexão entre hardware e software ocorre via wifi, conectando o app diretamente ao Raspberry Pi que realiza o intermédio com o modulo *RFiD* RC522 onde ocorre a leitura de cartões a serem consultados pelo aplicativo.

3.4 Módulo *RFID* RC522 13,56MHz

O modulo foi configurado utilizando a linguagem python onde foram importadas as bibliotecas "RPi.GPIO" e "SimpleMFRC522". A conexão entre o modulo e o Raspberry ocorre por meio dos pinos no modulo e as portas GPIO no Raspberry. Foram utilizados os seguintes pinos do modulo MFRC522, 3.3V no pino 1; RESET no pino 22; GROUND no pino

6; MISO no pino 21; MOSI no pino 19; SCK no pino 23; SDA no pino 23, que seguem um padrão comum no meio da prototipagem deste modulo. conforme descrito na tabela 1:

Tabela 1. Pinagem utilizada

PINO MFRC522	PINO RASPBERRY PI
3.3V	Pino 1 – 3.3V
RST	Pino 22 (GPIO25)
GND	Pino 6 – GND
IRQ	Não usado
MISO	Pino 21 (GPIO9)
MOSI	Pino 19 (GPIO10)
SCK	Pino 23 (GPIO11)
SDA	Pino 24 (GPIO8)

Fonte: Autoral, 2022.

Na tabela 1 foi demonstrado como deve ser conectado os pinos entre o modulo MFRC522 e o Raspberry Pi.

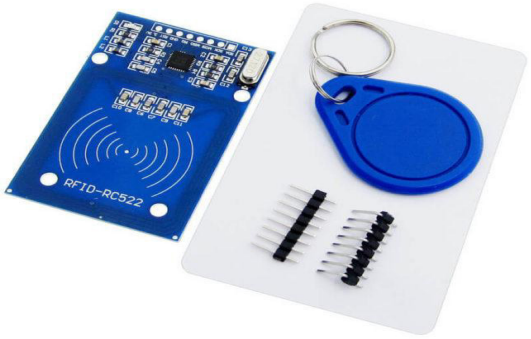


Figura 4. Módulo *RFID* RC522 13,56MHz Mifare.

Fonte: https://www.institutodigital.com.br/produto/modulo-leitor-*RFiD*-rc522-13-56mhz/

Na figura 4 temos o modulo que possibilita o uso das carteiras de meia passagem de ônibus uma vez que são cartões *RFiD* e pode-se utilizar o mesmo cartão em diversas aplicações já que o mesmo possui um token (ID). Nesta aplicação o mesmo deve ser vinculado a uma pessoa (CPF) registrado a um banco de dados que será detalhado mais adiante.

3.5 Cartão de Memória MicroSD 8GB

Este formato de armazenamento se destaca pela sua velocidade de leitura e gravação além de sua portabilidade já que seu tamanho e peso são ínfimos. É um tipo de memória flash. Está presente em nossos smartphones, câmeras e outros eletrônicos que usam algum tipo de armazenamento que pode ser visto na figura 5.

Figura 5. Cartão de Memória Sandisk Micro sd 8GB.

Fonte: https://www.shoptime.com.br/produto/4326303061?pfm_carac=cartao-de-memoria-8gb-sandisk

Neste projeto ele armazenará o sistema operacional Raspberry Pi OS, o banco de dados desenvolvido em MARIADB e as API's desenvolvidas em Python já que o servidor será um local host afim de reduzir custos com armazenamento.

3.6 Jumpers Fêmea-Fêmea

Jumpers são fios metálicos isolados com plugues em ambas as extremidades, podendo variar entre o tipo macho e fêmea. Comumente utilizado em prototipagem e placas de circuito impresso. Destaca-se por sua mobilidade e capacidade de reaproveitamento entre projetos exibido na figura 6.

Figura 6. Jumpers fêmea-fêmea.

Fonte: Autoral, 2022

Como visto na figura 4, foi aplicado o uso de jumpers do tipo fêmea-fêmea, para fazer a ligação do módulo *RFiD* RC522 com as GPIO do Raspberry Pi 3B.

3.7 Aplicação React Native

A aplicação foi criada utilizando React Native com auxílio do Expo. onde foram criadas as rotas dentro da pasta "screens" conforme a figura 7:

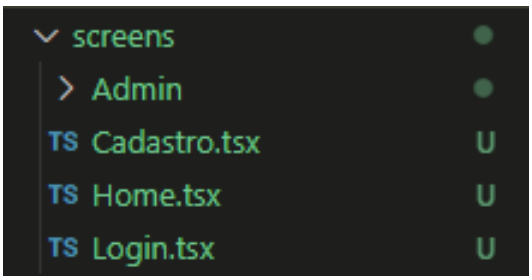


Figura 7. Rotas principais da aplicação.

Fonte: Autoral, 2022

Na rota "Cadastro", o usuário tem a possibilidade de se cadastrar para ter acesso à aplicação, todos os usuários que se cadastram nesta tela, recebem o nível de usuário padrão. A figura 8, mostra a tela de cadastro que vinculada a rota "Cadastro".

Figura 8. Tela de cadastro.

Fonte: Autoral, 2022

Figura 9. Tela Home.

Fonte: Autoral, 2022

Na rota "Home", o usuário de nível padrão tem acesso apenas à tela onde possui o contador de *check-in* e ao botão sair que realiza o *logout*, porém não possui acesso ao botão cadastro. Conforme pode ser percebido na figura 9.

A tela da figura 10 mostra a rota "Admin" e suas sub-rotas: "Create", "List" e "Update". Responsáveis pelo CRUD da aplicação.

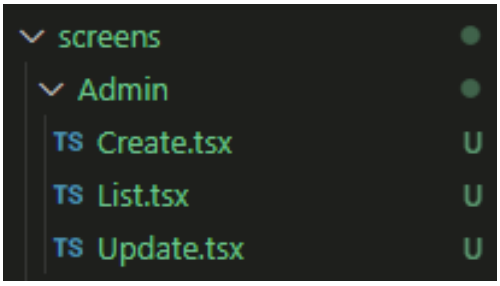


Figura 10. Sub-rotas da aplicação.

Fonte: Autoral, 2022

A sub-rota “Create”, “List” e “Update”, são utilizadas na manutenção da tela “cadastro” tela 9.

3.8. Análise da Aplicação

No artigo em questão, com o intuito de especificar os dados necessários para a realização do projeto, foi utilizado UML (Linguagem Unificada de Modelagem) para auxiliar no levantamento e análise de requisitos, usados para tomada de decisão mostrando diferentes perspectivas sobre quais dados coletar, definir o tipo de dado a ser coletado e os requisitos. Para facilitar o entendimento de como o aplicativo deveria funcionar foi criado um diagrama de caso e um diagrama de classe que seguem nos itens 3.8.1 e 3.8.2 respectivamente.

3.8.1 Diagrama de Caso de Uso

A figura 11, tem como intuito ambientar de forma inicial sobre o uso do sistema por meio dos atores e suas permissões. Representado pelo administrador e o usuário padrão. O usuário padrão tem permissão para: fazer login, cadastrar-se e ver total da quantidade de confirmações para a produção de refeições.

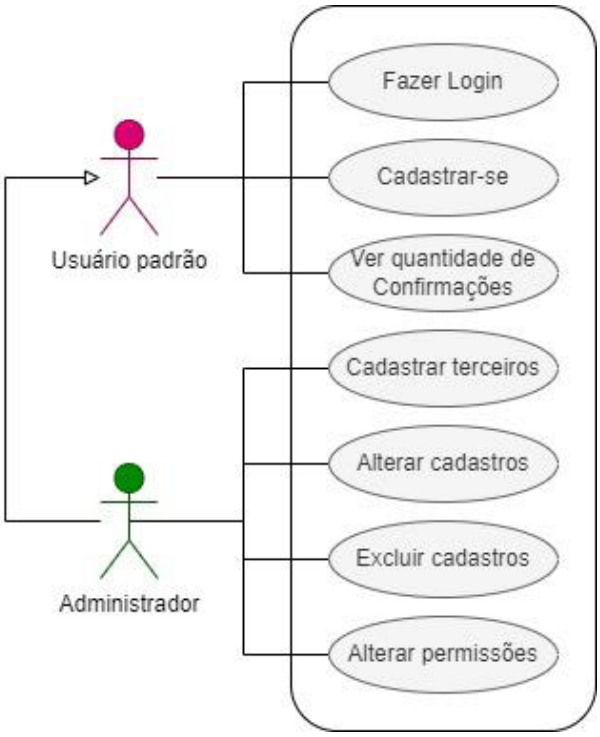


Figura 11. Diagrama de caso de uso.

Fonte: Autoral, 2022.

O administrador, possui todas as permissões cedidas ao usuário padrão. Estas são herdadas pelo administrador. Além dessas permissões, ele possui suas permissões exclusivas, como: cadastrar terceiros, alterar cadastros, excluir cadastros, e alterar permissões.

3.8.2 Diagrama de Classe

A figura 12, demonstra de forma breve, a relação entre os objetos do sistema. Partindo da tela de *login* pode-se logar ou realizar um novo cadastro, somente pessoas cadastradas possuem um login logo não é possível logar-se sem um cadastro prévio.

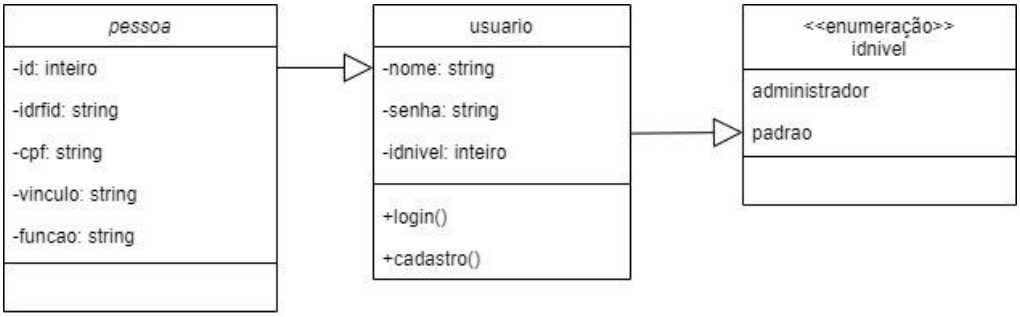


Figura 12. Diagrama de classe.

Fonte: Autoral, 2022.

Todo novo cadastro recebe por padrão o nível de acesso padrão. Somente um usuário que possua previamente o nível administrador, pode realizar a alteração de nível de outros usuários.

3.9 Banco de dados MariaDB

Foi criado o banco de dados utilizando MariaDB por meio da linguagem SQL. Conforme a figura 13, as tabelas criadas foram nível, pessoa e ponto.

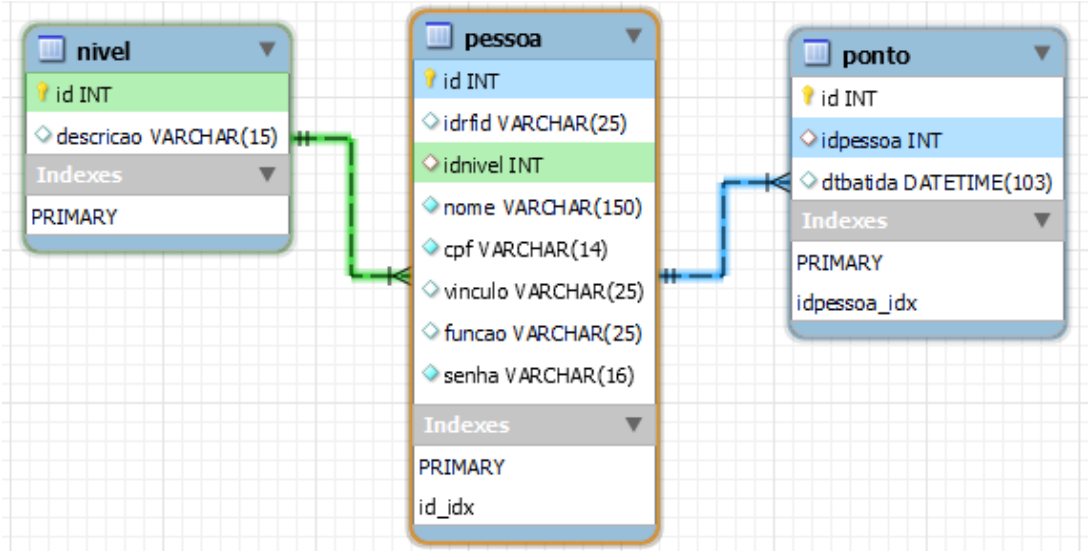


Figura 13. Relacionamento de tabelas.

Fonte: Autoral, 2022.

Em todas as tabelas, foi idealizado que suas chaves primarias se chamariam "id". A tabela nível possui apenas itens: id e descrição, na tabela pessoa possui as colunas: id, id*RFiD*, idnivel, nome, cpf, vinculo, função e senha. O campo idnivel remete a chave estrangeira da tabela nível que associa qual o nível de acesso que o usuário terá. A tabela ponto possui as colunas: id, idpessoa e dtbatida. O campo idpessoa remete à chave estrangeira da pessoa identificando qual pessoa bateu ponto e em qual data e hora referente às tabelas visualizadas na figura 13.

3.10 Desenvolvendo o API

Foram criados arquivos de extensão .py com funcionalidades independentes que são chamados pelo app em cada campo conforme sua necessidade em *localhost*.

Por exemplo, ao clicar em login o sistema chama a funcionalidade do arquivo loginCheck.py onde pega os dados no campo usuário e senha e realiza a demonstrada na figura 14.

```
SELECT
    NOME,
    SENHA
FROM
    PESSOA
WHERE
    PESSOA.NOME = camponome
    AND
    PESSOA.SENHA = camposenha;
```

Figura 14. Consulta e comparação de dados.

Fonte: Autoral, 2022.

Se os dados retornados forem iguais aos digitados então o usuário terá acesso a tela de bem-vindo.

Algo semelhante ocorre ao clicar no campo cadastro onde o *select* irá verificar o nível do usuário registrado na tabela pessoa a qual o solicitante pertence. Se o valor retornado for 1 o usuário é um administrador e terá acesso aos cadastros, se o valor retornado for 2 o usuário é um usuário padrão, abrindo um pop-up contendo a informação: Você não permissão para acessar os cadastros!

4. RESULTADOS E DISCUSSÕES

Com o intuito de solucionar o problema referente a quantidade de refeições produzidas no refeitório, onde ocorria por diversas vezes a falta e outras a sobra de alimento, o que causava transtorno para os alunos e membros da instituição que atrasavam suas atividades enquanto aguardavam a produção de mais alimento, assim como, evitou o gasto com desperdício, tempo e gás já que agora será produzida a quantidade com assertividade. Trazendo economia de tempo e dinheiro.

Durante o período de implementação, foram cadastrados 1327 usuários, dentre eles alunos, professores, corpo administrativo e demais funcionários. Para aqueles que não possuíam o cartão de meia passagem, optou-se por usar algum cartão *RFiD* de outras origens. Pois o cartão *RFiD* em questão não armazena dados e sim funciona como um token que identifica que aquele cartão corresponde àquela pessoa, e esta pessoa é quem realiza a batida de ponto na leitora.

Criou-se a regra, em que as pessoas que iriam fazer uso do refeitório naquele dia deveriam ir até o leitor *RFiD* situado próximo à entrada da instituição e fazer o seu *check-in*. Podendo ser realizado até as 08:00, após este horário o sistema não adiciona ao contador de refeições novos *check-in's* e não aceita 2 *check-in* do mesmo cartão em um mesmo dia. Às 12:30 o sistema volta aceitar novos *check-in's*, já que existem casos de pessoas

que só comparecem na instituição no turno vespertino. Nessa regra existem 2 exceções, são, 2 cartões previamente cadastrados. Que não estão sujeitos à regra de horário nem quantidade de check. Além disso, eles possuem funcionalidades diferentes entre si, um deles realiza apenas *check-in* e o outro apenas *check-out*.

Dessa maneira, caso tenha algum retardatário no *check-in* a pessoa da administração responsável por estes 2 cartões irá realizar o *check-in* o mesmo se aplica no caso de alguém por algum motivo vá sair da instituição antes do horário da refeição, o responsável pelo cartão deverá usar o cartão de *check-out* uma vez, reduzindo a confirmação de uma pessoa no contador de refeições. De qualquer maneira se criou uma margem de segurança na quantidade produzida de dez refeições a mais, visando que sempre existem imprevistos.

Em conversa com a administração da instituição os resultados obtidos pela solução durante o período de testagem foram satisfatórios. Inicialmente houve diversos esquecimentos e atrasos de *check-in*. Mas em pouco tempo tornou-se um hábito.

5. CONCLUSÃO

O sistema empregado se mostrou de extrema eficiência e simplicidade no seu uso, necessitando de pouco treinamento. A incidência de desperdício e falta de refeições na instituição durante o período de aplicação da solução demonstrou bons resultados garantindo economia de recursos financeiros com a redução obtida no desperdício de comida. Tendo a proposta do sistema cumprida. Trabalhando dentro da margem de segurança.

Em contribuições futuras pode ser implementado uma estrutura mais forte para fixar os componentes da leitora evitando imprevistos, como queda, e exposição direta do hardware, o que impediria possíveis danos ao mesmo. Também foi percebido que o sistema poderia integrar outras funcionalidades agregando mais usos da automação que foi empregada, como chamada para os alunos, batida de ponto para os funcionários e registro de entrada e saída de pessoas.

As possibilidades na utilização de *RFiD* no meio educacional são diversas. Desde controle de estoque liberação de acesso a áreas restritas e até mesmo compras armazenando créditos adquiridos previamente no caixa da instituição para que seja utilizado em lanchonete ao longo da semana. Tudo depende da necessidade e recursos para implementação na instituição.

AGRADECIMENTOS

Agradeço primeiramente a minha esposa Danielle Farias e filhas Sophia Eneida Farias e Nicole Noêmia Farias, por seu apoio e paciência.

A minha mãe, que sempre me deu apoio e incentivou a estudar e sempre insistia para que obtivesse a minha formação superior quando em vida.

Agradeço também aos meus companheiros de estudo Gabriel Mouta e Victor Freire que sempre estiveram presentes durante todo o curso ajudando nas dúvidas e problemas recorrentes ao longo do curso. Sem suas contribuições não seria possível concluir este e outros projetos.

E por último, mas não menos importante aos professores que lecionaram ao longo do curso, estes tiveram fundamental importância na transferência de conhecimentos e compartilhamento de experiências essenciais na formação, em especial aos professores Will Almeida e Edilson Lima que também são coautores deste artigo.

Referências

ABREU, A. et al. ***RFID* technology applied to students' backpacks**. 2011 IEEE International Conference on *RFID*-Technologies and Applications, set. 2011.

AMAZON WEB SERVICES, Aws. O que é IoT?: Guia de Internet das Coisas para iniciantes. AWS, 2022. Disponível em: https://aws.amazon.com/pt/what-is/iot/. Acesso em: 09 nov. 2022.

BAKHT, K. et al. **Design of an Efficient Authentication and Access Control System Using *RFID***. 2019 3rd International Conference on Energy Conservation and Efficiency (ICECE), out. 2019.

BARLATI DE MATOS, W. G. et al. **Inventory control with *RFID* integration**. 2015 IEEE Brasil *RFID*, out. 2015.

BRASIL, Oracle. **O que é IoT?**. Oracle Brasil, 2022. Disponível em: https://www.oracle.com/br/internet-of-things/what-is-iot/. Acesso em: 09 nov. 2022.

GONÇALVES, Eduardo. **O que é o Raspberry Pi 3**. Mundo TI Brasil, 2018. Disponível em: https://www.mundotibrasil.com.br/o-que-e-o-raspberry-pi-3-rpi3/. Acesso em: 09 nov. 2022.

MACHADO, Felipe Nery Rodrigues. **BANCO DE DADOS: PROJETO E IMPLEMENTAÇÃO**. Digital: Saraiva Educação S.A., 2020.

MIJIC, D. et al. **An Improved Version of Student Attendance Management System Based on RFID**. 2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH), mar. 2019.

PATRIMÔNIO E AVALIAÇÕES, Afixcode. **O QUE É PARA QUE SERVE UM LEITOR DE RFID**. Mundo TI Brasil, 2021. Disponível em: https://www.afixcode.com.br/blog/leitor-de-*RFiD*/. Acesso em: 09 nov. 2022.

SD, Association. **Visão geral do padrão SD: Uma introdução aos padrões SD**. SD Association, 2022. Disponível em: https://www.sdcard.org/developers/sd-standard-overview/. Acesso em: 11 nov. 2022.

SONAR, P. M.; WALKE, S. S.; BANE, R. R. **Student Smart Card**. Disponível em: <http://ieeexplore.ieee.org/iel7/8581980/8596764/08597202.pdf>. Acesso em: 18 maio. 2022.

3

DESENVOLVIMENTO DE UM APLICATIVO PARA PACIENTES EM UTIS: API REST UTILIZANDO SPRING BOOT PARA UM FRONT END DESENVOLVIDO EM FLUTTER

DEVELOPMENT OF A REST API USING SPRING BOOT FOR A FRONT END DEVELOPED IN FLUTTER

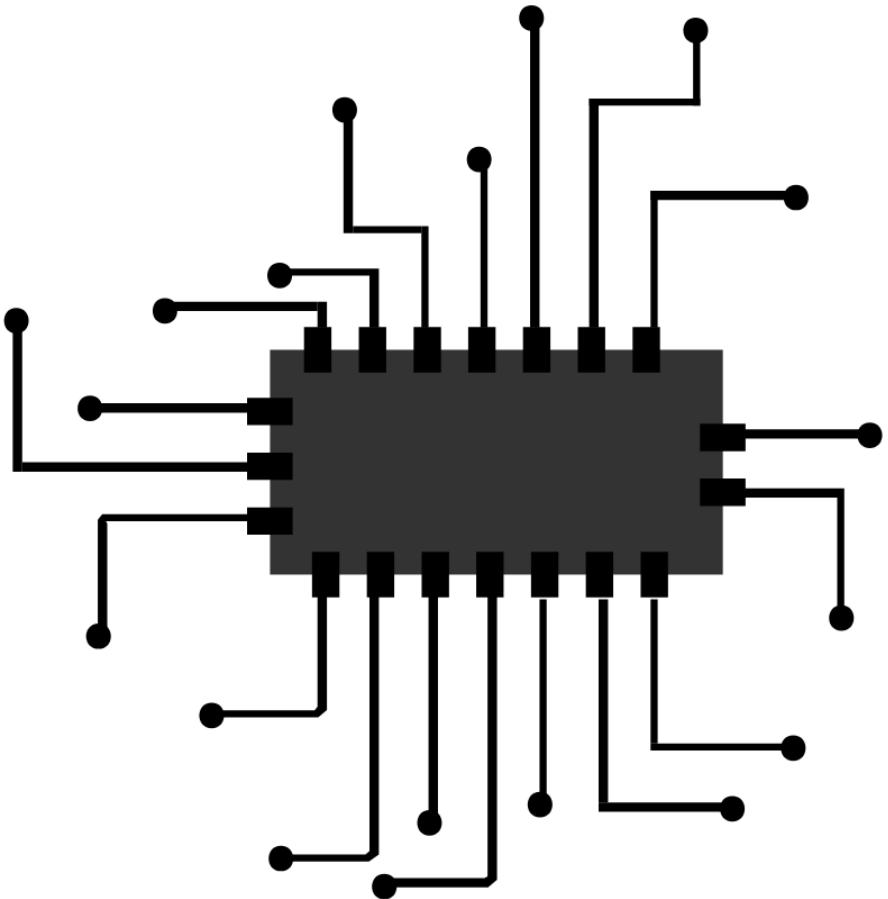
Allicia Sousa da Silva[1]

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, Allicia Sousa da, allicia.ass33@gmail.com; LIMA, Edilson Carlos Silva, edilsonlima3@gmail.com.}



d.o.i.: 10.29327/5192626.1-3

Resumo

Pacientes que se encontram nas Unidades de Terapia Intensiva (UTI), necessitam de acompanhamento periódico. E para acompanhar a evolução do quadro clínico, alguns profissionais usam questionários para fazer uma avaliação de todos os pacientes que estão internados, a fim de fazer comparativos das respostas para avaliar o quadro. O presente artigo, apresenta o desenvolvimento de uma API Spring boot, com API e *Design Patterns* que será consumida por uma aplicação *flutter* que fará cadastro dos pacientes, cadastro de leitos existentes na uti, e armazenará os questionários respondidos por cada paciente. Após a revisão da literatura foi realizado uma pesquisa do tipo aplicada com o uso de um questionário de quatro questões para fins de análise qualitativa.

Palavras-chave: *API Spring boot, Design Patterns, Flutter.*

Abstract

Patients who are in Intensive Care Units (ICU) need periodic monitoring. And to follow the evolution of the clinical picture, some professionals use questionnaires to make an evaluation of all the patients who are hospitalized, in order to make comparisons of the answers to evaluate the situation. This article presents the development of a Spring boot API and with API, Design Patterns that will be consumed by a flutter application that will register patients, register existing beds in the ICU, and store the questionnaires answered by each patient. After reviewing the literature, an applied survey was carried out using a questionnaire with four questions for the purpose of qualitative analysis.

Keywords: API Spring boot, Design Patterns, Flutter.

1. INTRODUÇÃO

Sabe-se que nos últimos anos a tecnologia tem feito avanços surpreendentes na medicina. Mas é evidente que ainda existem processos que precisam ser automatizados, tanto pela importância da tarefa quanto pelo tempo gasto para fazê-la. Uma simples entrevista realizada em pacientes internados nas UTIs para avaliação da evolução do quadro clínico é um exemplo de processo a ser automatizado.

Para pacientes neurológicos é comum o uso dessas entrevistas para identificar problemas e possíveis tratamentos, e no caso de pacientes neurológicos internados em unidades de terapia intensiva, essa entrevista também é utilizada, porém não mais para identificar a doença, mas sim para avaliar a evolução do estado clínico do paciente.

Para isso, tomamos mão de ferramentas como o *framework Spring boot* para desenvolvimento da *Api Rest*, princípios de UML, utilizando os diagramas de caso de uso, diagrama de classe e *Design Patters*, e para demonstrar o consumo da *API* foi desenvolvido um aplicativo Flutter visando as boas práticas de desenvolvimento.

Desta forma o presente artigo está disposto em 5 capítulos, começando com a introdução demonstrada neste capítulo, no capítulo 2 uma Fundamentação Teórica onde será tratado sobre todas as ferramentas e tecnologias utilizadas para desenvolver este artigo, no capítulo 3 O Procedimento Metodológicos onde descreve - se como as tarefas e etapas deste artigos foram desenvolvidas e onde cada tecnologia foi utilizada, no capítulo seguinte, os Resultados e Discussões, é mostrado por meio de uma tabela em quais resultados se chegou baseado nas respostas de usuários sobre a viabilidade e utilização da aplicação, e por fim no capítulo 5, a Conclusão que terá planos dos trabalhos futuros para dar continuidade e melhoria no que será descrito neste artigo.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será falado sobre as ferramentas e tecnologias utilizadas para desenvolvimento da API. No tópico 2.1 o assunto será sobre o conceito de manifesto ágil e no tópico seguinte sobre a metodologia XP utilizada para desenvolver este projeto. No tópico 2.3 conceituaremos orientação a objetos e seus 4 paradigmas, e os demais tópicos são sobre MVC, UML, *Design Patterns*, spring boot, api rest, diagrama de caso de uso e diagrama de classes.

2.1 XP

O *eXtreme Programming* é uma metodologia ágil de desenvolvimento de software voltada para times de pequeno a médio porte, no qual os requisitos são vagos e mudam frequentemente (WILDT, 2015, p.16).

Segundo Macedo e Carvalho (2012, p.151), para que projetos de software sejam feitos com qualidade, a XP propõe uma formação de equipe com alguns papéis bem definidos. São eles o gerente de projeto, coach, desenvolvedor, analista de testes e o redator técnico. A escolha da XP se deu por se tratar de uma equipe pequena e por se tratar de uma metodologia com processos simples. Nossos requisitos não estavam bem definidos e a entrega seria feita por partes funcionais, e a metodologia daria suporte para isso.

O *Extreme Programming* encapsula algumas regras práticas dentro de 4 atividades

padronizadas do modelo. Segundo Kent, essas 4 atividades são denominadas: planejamento, projeto, codificação e teste.

2.2 MVC

Segundo GAMMA et al. (2007, p.20), a abordagem MVC é composta por três tipos de objetos. O Modelo é o objeto de aplicação, a Visão é a apresentação na tela e o Controlador é o que define a maneira como a interface do usuário reage às entradas do mesmo.

Para quem usa o *Spring boot* como ferramenta de desenvolvimento de API deve estar familiarizado com o termo MVC. Um dos grandes benefícios desse padrão é a portabilidade, pois uma API desenvolvida na arquitetura desse modelo pode ser utilizada na web e mobile.

Apesar de muitas pessoas considerarem essa sigla como um padrão de design de interface, na verdade ele é um padrão de arquitetura de software responsável por contribuir na otimização da velocidade entre as requisições feitas pelo comando dos usuários (ZUCHER, 2020).

2.3 UML

De acordo com Guedes (2011, p.7), UML ou Linguagem de Modelagem Unificada, é uma linguagem visual utilizada para modelar softwares baseados no paradigma de orientação a objetos.

Quando falamos em modelagem, a primeira coisa que vem na cabeça é desenho, e a UML é basicamente isso, ela modela os projetos através de desenhos como o diagrama de classe e o diagrama de classe de uso, que também foram utilizados no desenvolvimento desse projeto.

Segundo Fabris E Catarino (2017, p.13), existem alguns tipos de diagramas que compõem a UML, sendo eles do tipo diagramas estruturais, diagramas comportamentais e de interação. E de acordo com Melo (2002), essas categorias são descritas da seguinte maneira:

- Diagramas estruturais: responsáveis pelo tratamento de aspectos estruturais, sob o ponto de vista de um sistema ou de suas classes. Seu objetivo é a visualização, especificação, construção e documentação de aspectos estáticos do software;
- Diagramas comportamentais: responsáveis pela descrição e modelagem do aspecto dinâmico do software; 14 U1 - Diagramas estruturais da UML
- Diagramas de interação: representa o subgrupo dos diagramas comportamentais, usados para mostrar a interação entre elementos de uma modelagem e da aplicação.

2.3.1 Diagrama de Caso de Uso

Caso de uso Segundo Corkburn (2005, p.21) "Caso de uso descreve o comportamento dos sistemas sob diversas condições conforme o sistema responde a uma requisição de um *steakholdes*". Requisição é uma solicitação de serviço, seja uma consulta ou cadastro de produto por exemplo, os stakeholders por sua vez são os atores no nosso diagrama eles são as pessoas de interesse na aplicação.

2.3.2 Diagrama de Classe

De acordo com Guedes (2011, p.19), diagrama de classes, define a estrutura das classes utilizadas pelo sistema, determinando os atributos e métodos que cada classe tem, além de estabelecer como as classes se relacionam e trocam informações entre si.

Todo desenvolvedor depois de estabelecer os requisitos do sistema, constrói um diagrama de classe da sua aplicação, e é de lá que parte o desenvolvimento da API. Pois nelas contém a descrição das tabelas do banco, suas características e tipos de dados.

2.4 *Design Patterns*

Segundo Gamma et al. (2007) um padrão de projeto tem 4 elementos essenciais: O nome do padrão, o problema, a solução e as consequências. Elementos esses que vão dar base ao padrão de projeto. Desenvolver projetos seguindo padrões, torna o processo mais claro e prático, pois você usa recursos que já foram utilizados por outras pessoas, e faz com que seu projeto seja entendido por outros desenvolvedores. os padrões são divididos em três tipos, os de criação, estruturais e comportamentais.

Os padrões de projetos são divididos em 3 categorias: padrões de criação, padrões estruturais e padrões comportamentais. Os padrões de criação segundo Serapião e Ribeiro (2004, p.32) abstraem o processo de instanciação, eles ajudam a tornar um sistema independentemente de como seus objetos são criados, compostos e representados. Alguns exemplos de padrão de criação e o padrão *singleton* e o *factory method.*

Os padrões estruturais se preocupam com a forma como classes e objetos são compostos para formar estruturas maiores. Os padrões estruturais de classes utilizam a herança para compor interfaces ou implementações (SERAPIÃO e RIBEIRO, 2004, p.45). alguns exemplos de padrão estrutural são o *decorator* e o *adpter*. E o pôr fim os padrões do tipo comportamentais exemplo dele são o *observer* e o *templet method.*

2.5 *Spring Boot*

As vantagens de se utilizar um *framework* são diversas, como otimização de tempo e redução de trabalho. Segundo Kriger (2022), ao utilizar um *framework*, as chances que você tem de se deparar com erros frequentes são consideravelmente reduzidas. Dessa forma, quando um projeto é iniciado pelo profissional, esse pacote de códigos prontos é um suporte que facilita o trabalho, evitando a necessidade de iniciar o site do zero.

O Spring não é um framework apenas, mas um conjunto de projetos que resolvem várias situações do cotidiano de um programador, ajudando a criar aplicações Java com simplicidade e flexibilidade (AFONSO, 2017, p.12).

2.6 *Api Rest*

Uma API (*Application Programming Interface*, traduzindo para português, Interface de Programação de Aplicação) pode ser definida como um conjunto de padrões que permite a construção de aplicativos, onde ele conecta aplicações, podendo ser utilizada nos mais variados tipos de negócios (GUEDES, 2019).

Segundo Fabro (2020) API é um conjunto de normas que possibilita a comunicação entre plataformas através de uma série de padrões e protocolos. A figura 1, demonstra

de forma simples o seu funcionamento, o usuário faz uma requisição, que pode ser a consulta de um produto em um site de compras por exemplo, a API faz a consulta no banco de dados e retorna uma resposta para o usuário, que é o produto pesquisado ou uma mensagem caso o produto não tenha sido encontrado.

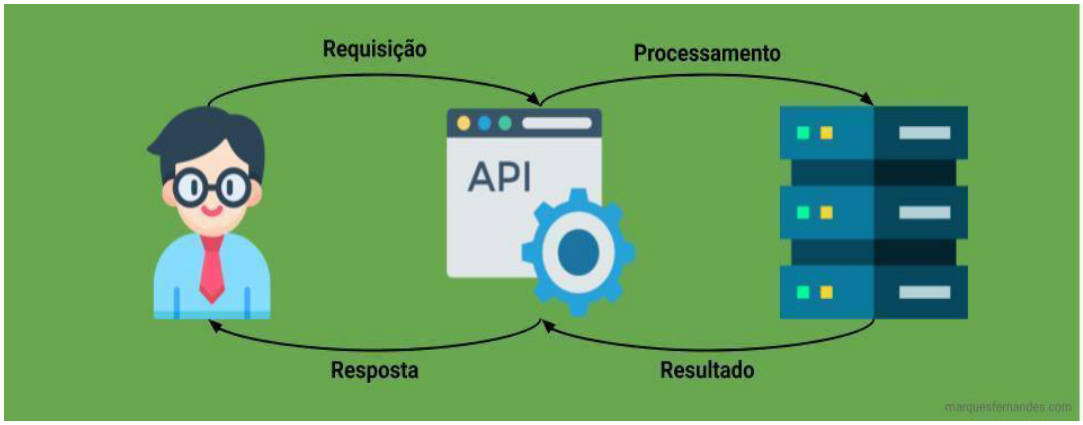


Figura 1: O que é uma api?

Fonte: Guedes, 2019.

Falando especificamente de um sistema que utiliza API Spring boot, a requisição é um serviço oferecido pela aplicação. Em um sistema de biblioteca, as requisições são: cadastrar livros, consultar acervo, e verificar status do livro, por exemplo. Para cada serviço existe um método HTTP para realizá-lo.

Segundo Melo (2021) HTTP "é um protocolo de comunicação, ou seja, uma convenção de regras e padrões que controla e possibilita uma conexão e troca de dados entre dois sistemas computacionais". E eles podem ser do tipo GET, POST, PUT, DELETE e outros.

3. PROCEDIMENTOS METODOLÓGICOS

Na escrita deste artigo foi realizado uma pesquisa aplicada, pois esta é dirigida a gerar soluções de problemas, ou seja, o resultado dessa metodologia é algo para ser aplicado de forma imediata. Essa metodologia foi utilizada com abordagem qualitativa, onde foi utilizado como ferramenta um questionário com 4 perguntas sobre o uso do aplicativo dando ênfase nas suas funcionalidades.

Para isso foi desenvolvido uma API Spring Boot para ser consumida por uma aplicação mobile multiplataforma afim de solucionar problemas de falta de dinamismo e acúmulo de papel na realização de entrevistas em pacientes neurológicos internados em UTI's.

Nos tópicos seguintes será demostrado e explicado de forma detalhada como se deu o desenvolvimento da aplicação.

3.1 Aplicação

Visando substituir completamente o método de avaliação de pacientes neurológicos internados nas unidades de terapias intensivas por meio de questionário impressos, o presente artigo apresenta o desenvolvimento de uma *API Spring boot* para ser consumida por uma aplicação *Flutter*, que faz o cadastro do usuário que irá realizar a avaliação, cadastro de pacientes e de leitos, e armazenará os questionários respondidos de cada paciente.

O desenvolvimento da API iniciou - se com as especificações dos requisitos funcionais, e com essa informação foi criado o modelo de dados e os diagramas, nos tópicos seguintes cada um desses elementos será mostrado.

3.2 Análise da Aplicação

A fase de análise da aplicação foi fundamental para o entendimento dos requisitos básicos e entendimento das funcionalidades básicas do sistema. A partir desta análise foi possível construir os diagramas que fundamentam todo o desenvolvimento. O primeiro diagrama a ser construído foi o diagrama de caso de uso, demostrado logo abaixo na figura 2.

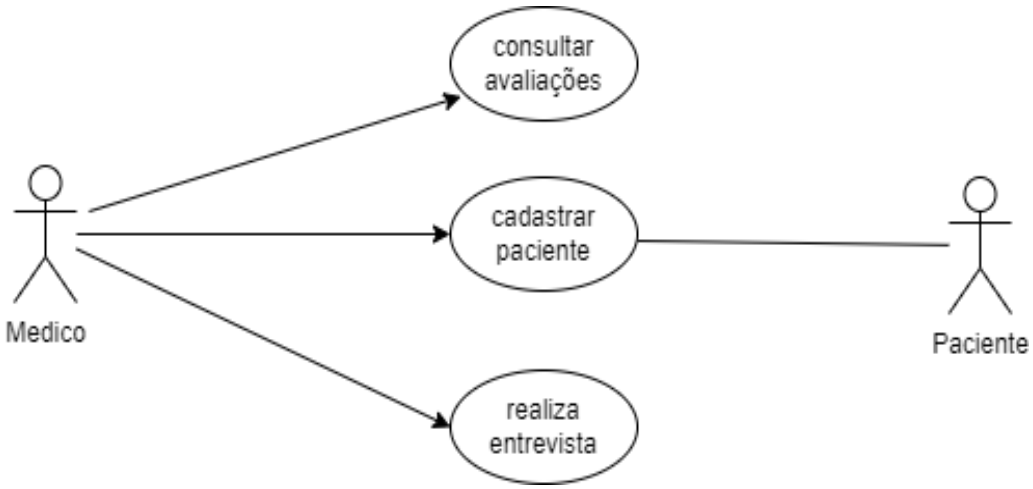


Figura 2: Diagrama de caso de uso.

Fonte: Autoral, 2022.

Este diagrama serve para demostrar como o usuário (ator), representados pelos bonecos, interagem com o sistema, isso significa dizer que o diagrama de caso de uso serve para demostrar as ações dos atores no sistema.

O outro diagrama é o de classes, que serve para descrever a estrutura do sistema, assim como suas classes, métodos, atributos e relacionamento. Isso é demostrado logo abaixo na figura 3.

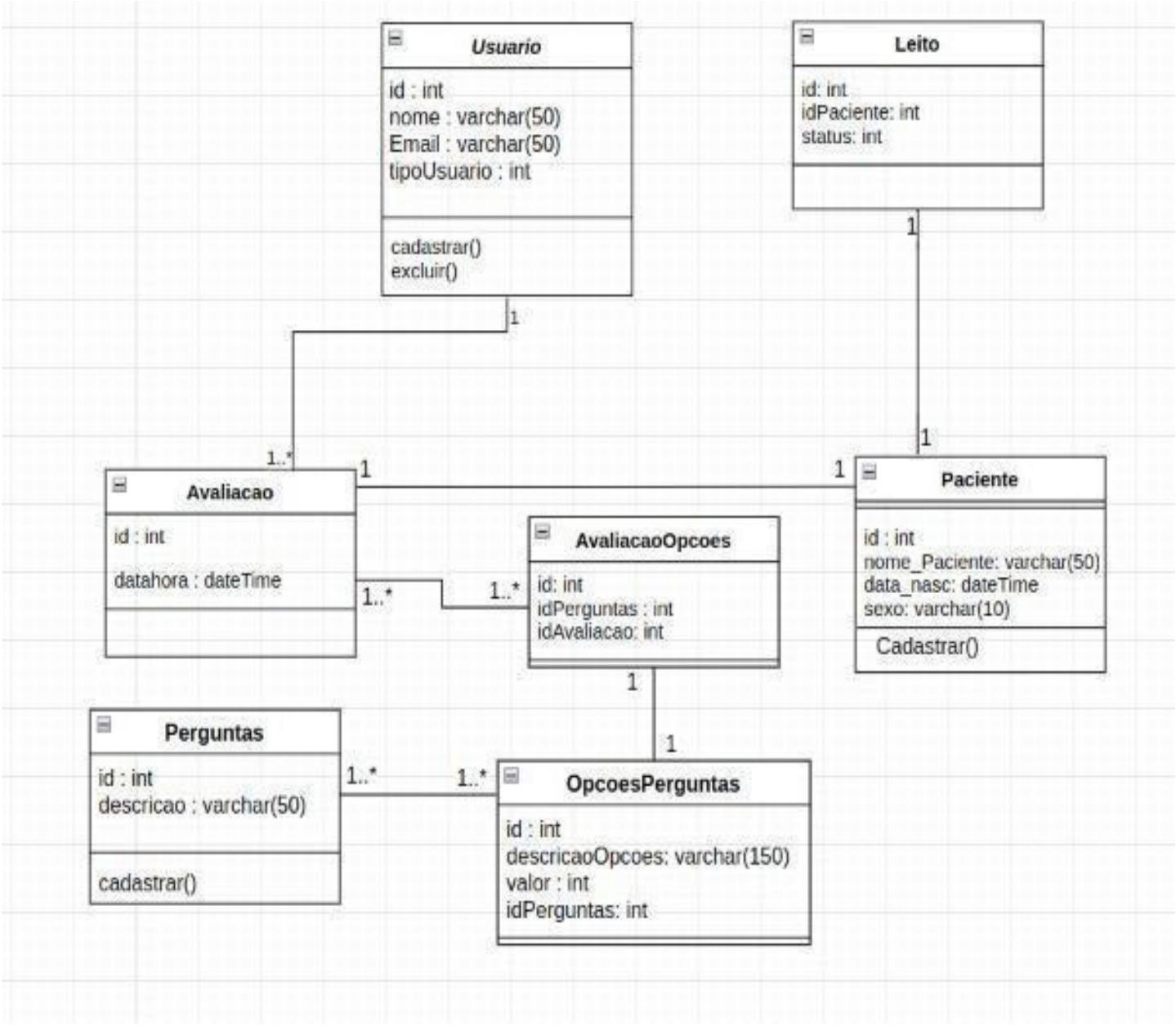


Figura 3: diagrama de classe.

fonte: Autoral, 2022.

O diagrama acima foi desenvolvido tendo como base os requisitos do sistema, ele tem 7 classes, sendo uma delas a classe Usuário que fará todas as ações dentro da aplicação.

O *framework Spring boot* que é um ambiente de desenvolvimento orientado a objetos muito utilizado por sua facilidade e praticidade de desenvolvimento foi utilizado para desenvolver a API, usando os conceitos de padrões de projetos e MVC para organizar as classes e pacotes.

3.3 Desenvolvendo a API

A API foi desenvolvida no framework *Spring* boot, tendo como base o diagrama de classe. Na imagem 4 mostramos o trecho de código da classe UsuarioController.

UsuarioController.java | AvaliacaoController.java

```
41
42  @GetMapping("/{userId}")
43  public ResponseEntity<Usuario> buscarUsuario(@PathVariable Long userId) {
44      Optional<Usuario> user = usuarioRepository.findById(userId);
45
46      if(user.isPresent()) {
47          return ResponseEntity.ok(user.get());
48      }
49      return ResponseEntity.notFound().build(); // RETORNA O ERRO 404 CASO O USER
50
51  }
52
53  @GetMapping
54  public List<Usuario> listaUsuarios(){
55      return  usuarioRepository.findAll();
56  }
57
58  @PostMapping
59  @ResponseStatus(HttpStatus.CREATED)
60  public void CadastrarUsuario(@RequestBody  @Valid Usuario usuario) {
61
62      usuarioRepository.save(usuario);
63  }
64
65  @DeleteMapping("/{id}")
66  public void DeletarUsuario(@PathVariable Long id) {
67
68      usuarioRepository.deleteById(id);
69
70  }
71
```

Figura 4: trecho de código

fonte: Autoral, 2022

A classe UsuarioControler é responsável por todas as requisições da classe Usuário, e para isso temos os métodos *GetMapiping*, *PostMapping*, *PutMapping* e *DeleteMapping*, o método DeleteMapping por exemplo é responsável por executar requisições de exclusão de usuários. Vale ressaltar que todo o código foi desenvolvido seguindo as boas práticas da programação orientadas a objetos.

Para o desenvolvimento da API, inicialmente foi construído o diagrama de classe na plataforma *Diagrams.net* conforme imagem 5, e foi baseado nos requisitos da aplicação que foram definidos no início do desenvolvimento do projeto.

3.4 Usando a API

Para consumir a API foi desenvolvido uma aplicação em Flutter completa, com todas as telas, desde a tela de login de usuário até a tela de realização do formulário que é o foco deste artigo. Para mostrar a API sendo consumida vou utilizar o *postman,* que funciona como uma API cliente é amplamente utilizado para testar API.

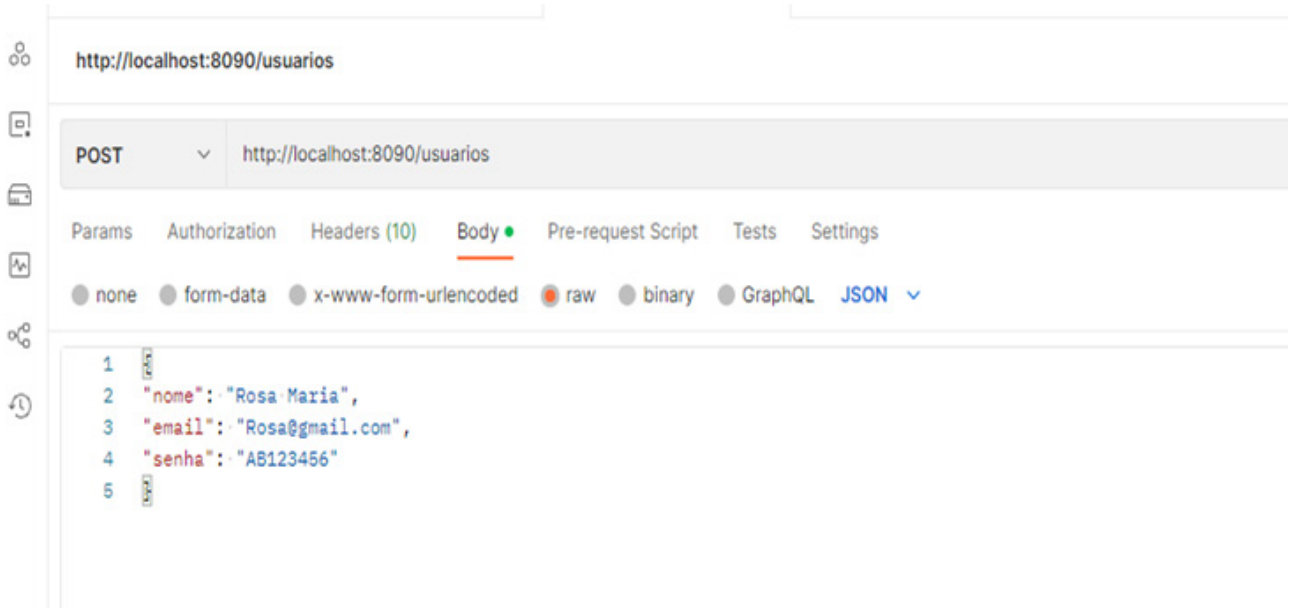


Figura 5: print postman-método POST.

Fonte: Autoral, 2022

Na figura 5 é demonstrado a utilização do método POST da classe usuário, esta requisição foi utilizada para cadastrar um novo usuário no banco através da URI http://localhost:8090/usuarios que é o endereço do *endpoints*. Para isso é necessário fornecer todos os dados do usuário conforme foi especificado na API.

Em *Flutter* foi desenvolvido em parceria com outra estudante, as telas que fazem conexão com a API. Abaixo na figura 6, temos a tela de cadastro que consome a API assim como foi demostrado no *print* do *postman* na figura 5.

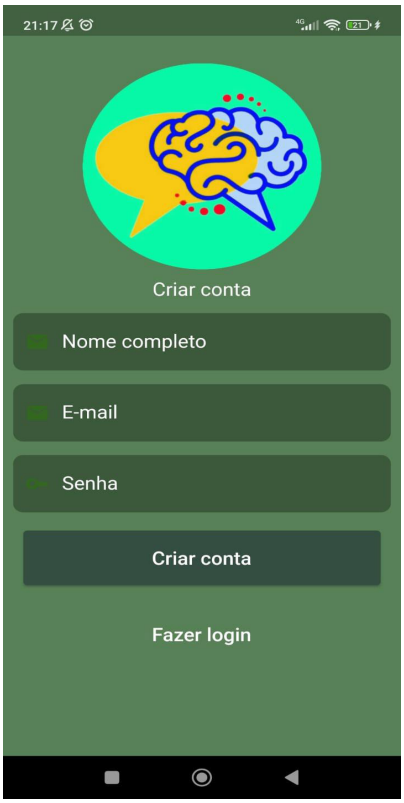


Figura 6: Tela de cadastro de usuário

Fonte: Karoline, 2022.

Ao cadastrar um novo usuário, as informações são armazenadas no banco assim como foi demostrado na requisição na figura 5. O trecho de código responsável por executa tal tarefa será mostrado logo abaixo na figura 7.

```
58 @PostMapping
59 @ResponseStatus(HttpStatus.CREATED)
60 public void CadastrarUsuario(@RequestBody  @Valid Usuario usuario) {
61
62     usuarioRepository.save(usuario);
63 }
64
```

Figura 7: Método POST.

Fonte: Autoral,2022.

Este é o trecho de código retirado da classe UsuarioController, que é a classe responsável pelas requisições. E para ficar ainda mais claro vamos usar o *postman* novamente, só que dessa vez será para demostrar que o usuário foi realmente cadastrado usando o método GET.

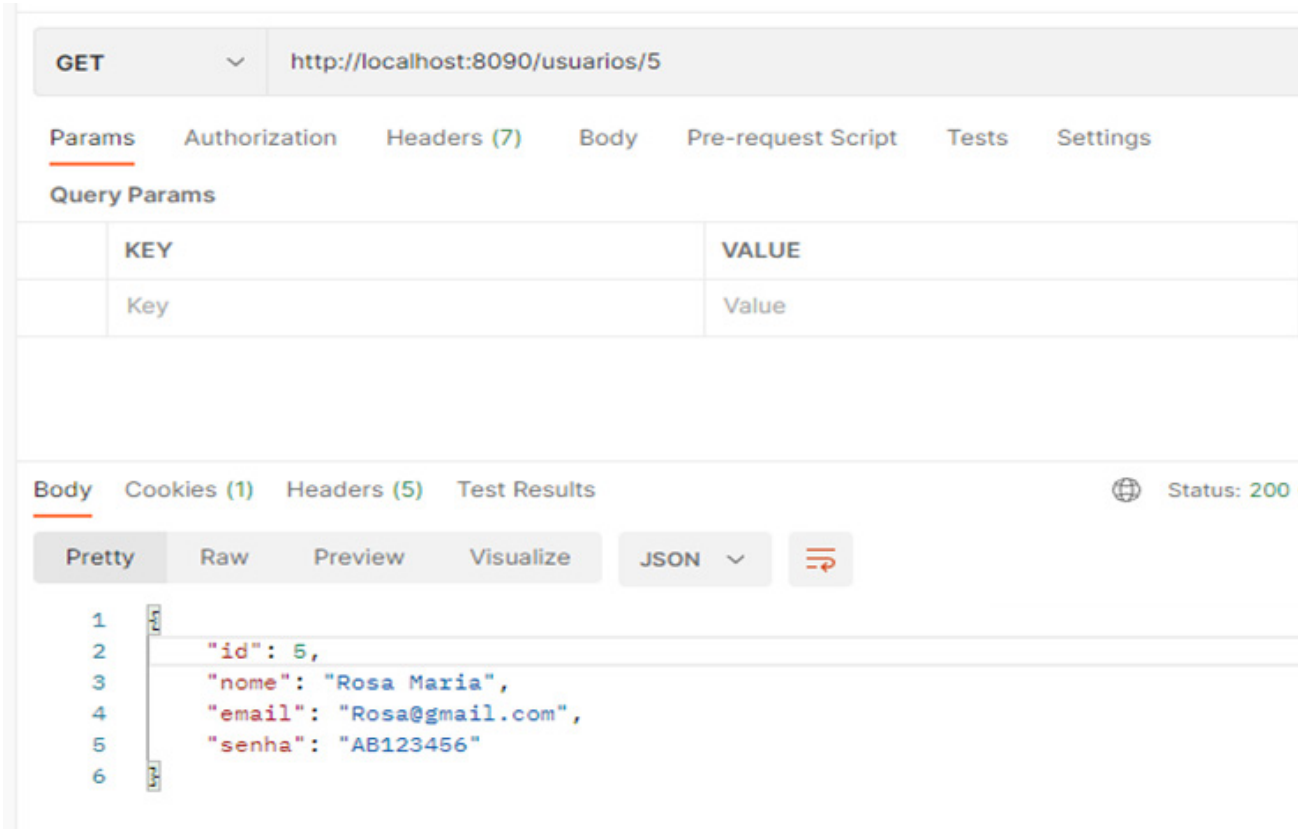


Figura 8: print postman-método GET.

Fonte: Autoral, 2022.

O método GET é utilizado para criar uma sequência de itens do banco, nesse caso, a intenção era lista o usuário com o Id número 5, que foi o usuário cadastrado na figura 5, se caso nossa intenção fosse listar todos os usuários, bastaria tirar o /5 da URI.

4. RESULTADOS E DISCUSSÕES

Neste capítulo será apresentado os resultados obtidos depois do desenvolvimento da API, será usado uma tabela e um gráfico para demonstrar de forma mais clara os resultados e discutiremos os pontos mais relevantes.

A tabela a seguir contém o formulário de pesquisa utilizado para avaliar a experiência do usuário, e para fins de discussão de resultados será apresentado as duas respostas com mais ocorrência.

Tabela 1: Questionário de entrevista de usuários

perguntas	**Resposta 1**	**Resposta 2**
O que você espera de uma aplicação direcionada para a área da saúde?	que seja segura e resolutiva.	que ela consiga cumprir com seus objetivos.
você conseguiria realizar o questionário sem dificuldades?	sim	sim
Você considera a tela home intuitiva e de fácil acesso?	sim	sim
Na tela de cadastro de pacientes, as informações que são pedidas são suficientes para obter os dados necessários para haver uma ficha do paciente? falta algo?	Não, o nome da mãe serve para diferenciar um paciente de outro paciente.	Sim, telefone e endereço são essenciais para o cadastro.

Fonte: Autoral, 2022

Com base nas respostas de alguns usuários, percebe - se que a aplicação resolve o problema proposto, mas alguns usuários levantaram pontos em relação às informações sobre os pacientes, informações estas que eu considero bastante relevante, mas o paciente já terá um cadastro no sistema do hospital no qual está internado, os dados pedidos dentro do aplicativo serviriam apenas para identificá-lo dentro do hospital.

Vale ressaltar que as perguntas do questionário de avaliação do paciente que estão na aplicação, foram fornecidas por um profissional que já o usa como forma de avaliação, mas como ponto de melhoria, futuramente será possível o cadastro de novas perguntas, para o profissional formular um questionário que melhor lhe atenda.

No geral, com base nos resultados podemos perceber que a aplicação é funcional e supri as necessidades dos médicos e profissionais envolvidos, assim como melhora o processo de entrevistas dos pacientes, porém alguns pontos estéticos devem ser adequados como forma de melhoria.

5. CONCLUSÃO

A tecnologia é usada em várias áreas como ferramenta de trabalho, e na medicina não é diferente, pensar como a tecnologia irá revolucionar atividades simples do cotidiano médico é pensar no futuro que está logo a frente. E um dos principais objetivos da aplicação era dar praticidade a tarefa de realizar a entrevista do paciente, e resolvendo esse problema consequentemente resolveríamos outras duas questões, que era o acesso e armazenamento de tais documentos.

E visando tornar o processo de avaliação mais rápido e eficiente, garantindo os padrões já estabelecidos, desenvolvemos uma API *Spring boot* para ser consumida por uma aplicação *flutter* que faz a avaliação em pacientes neurológicos em UTI's, e armazena todas as avaliações no banco de dados, facilitando o acesso do profissional para fins de comparação e análise clínica. A aplicação provou-se muito útil na organização dos formulários médicos, e prático na hora de realiza-lo, sem complicações ou acumulo de papelada, e sem ter a preocupação de onde e como armazenar tais documentos para consultas futuras, pois tudo fica a dois cliques de distância.

Ao final deste artigo percebe - se que a aplicação satisfaz os usuários, em relação a método antigo de avaliação, contribuindo assim para a organização das informações, e praticidade no acesso a elas. Com isso é pretendido fornecer ao usuário a opção de cadastrar novas perguntas, para permitir ao profissional personalizar o formulário conforme a sua necessidade. E com base nos feedbacks dos usuários que participaram da entrevista de coleta de dados, será adicionado o nome da mãe do paciente no formulário de cadastro, pois isso de fato contribui para identificação e evitar possíveis coincidências.

Referências

BECK, K. **Programação Extrema (XP) Explicada: Acolha as Mudanças**. Porto Alegre: Bookman, 2004.

CADENHEAD, Rogers; LEMAY, Laura. **Aprenda em 21 dias Java 2**. 4. ed. São Paulo: Campus, 2005.

CARVALHO, M.; MACEDO, P. **Metodologias ágeis engenharia de software sob medida**. 1.ed. São Paulo: Érica, 2012

CARVALHO, Thiago. **Orientação a objetos.** Casa do código, 2020.

CARVALHO, V.; TEIXEIRA, G. **Programação orientada a objetos**. Espírito Santo: e-Tec Brasil, 2012

COCKBURN, Alistair. **Escrevendo Casos de Usos Eficazes: Um guia prático para desenvolvedores de software**. São Paulo: Bookman Editora, 2005.

FABRIS, P.; CATARINO, I. **Análise orientada a objetos II.** Londrina: Editora e Distribuidora Educacional, 2017.

FABRO, Clara**. O que é API e para que serve?** Cinco perguntas e respostas. Disponível em: <https://www.techtudo.com.br/listas/2020/06/o-que-e-api-e-para-que-serve-cinco-perguntas-e-respostas.ghtml>. Acesso em: 03/06/2022.

GAMMA, Erich et al. **Padrões de Projetos:** soluções reutilizáveis de softwares orientados a objetos. Porto Alegre: Bookman, 2007.

GUEDES, Gilleanes. **UML 2 uma abordagem prática.** 2ed. São Paulo: Novatec, 2011.

GUEDES, Marylene. **O que é Spring Framework?** Disponível em: **<**https://www.treinaweb.com.br/blog/o-que-e-Spring-framework> Acesso em 12/10/2022.

GUEDES, Marylene. **O que é uma API?** Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-uma-api?gclid=Cj0KCQjw1tGUBhDXARIsAIJx01nvW7efzh_LASFQl-vzQYaI-IoTemjz-BtwTHFVC-tFM0q2-YEJqcXkaAgZnEALw_wcB>. Acessado em: 30/05/2022

KRIGER, Daniel. **O que são frameworks e por que são importantes para os devs.** Disponível em: <ht-

tps://kenzie.com.br/blog/framework/>. Acesso em 22/05/2022

MELO, A. C. **Desenvolvendo aplicações com UML 2.0 do conceitual à implementação**. 2 ed. Rio de Janeiro: Brasport, 2002.

MELO, Diego. **O que é HTTP?** Disponível em: <https://tecnoblog.net/responde/o-que=-e-http/#:~:text-O%20protocolo%20HTTP%20define%20oito,mais%20utilizados%20em%20aplica%C3%A7%-C3%B5es%20web> acessado em 01/11/2022.

RIBEIRO, H.; SERAPIÃO, T. **Padrões de projeto: design patterns.** TCC (ciência da computação) - Faculdade de Ciência da Computação da Universidade do Vale do Paraíba, Jacareí, 2004.

SOMMERVILLER, Ian. **Engenharia de software**. 9.ed. São Paulo: Person, 2011.

WILDT, Daniel et al. **eXtreme Programming**: Práticas para o dia a dia no desenvolvimento ágil de software. casa do código, 2015.

ZUCHER, Vitor. **O que é padrão MVC? Entenda arquitetura de softwares!** Disponível em:< https://www.lewagon.com/pt-BR/blog/o-que-e-padrao-mvc> Acessado em:13/11/2022.

4

IMPLEMENTAÇÃO DE UMA API REST USANDO NODE.JS E PRISMA PARA UM SITE DE APRESENTAÇÃO DE UM COWORKING

IMPLEMENTATION OF A REST API USING NODE.JS AND PRISMA FOR AN ACADEMIC SPACE PRESENTATION WEBSITE

Gabriel Mendes Mouta[1]

Edilson Carlos Silva Lima[2]

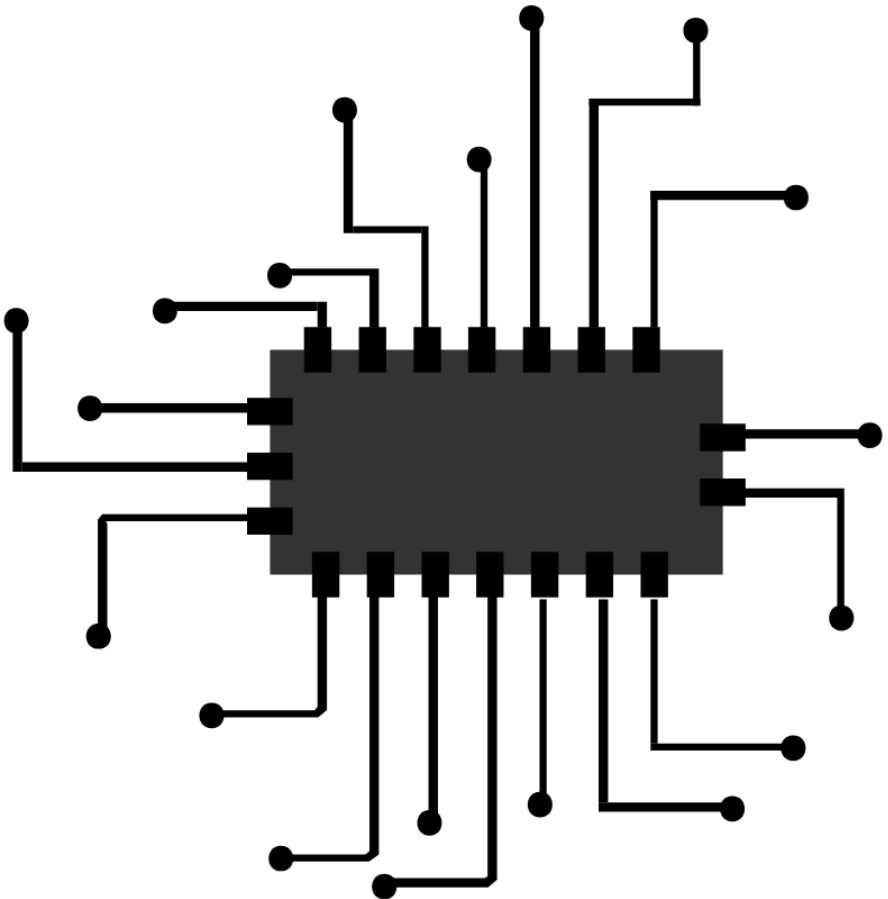
Will Ribamar Mendes Almeida[3]

1Engenharia de 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

{MOUTA, Gabriel Mendes,gabrielmouta06032001@gmail.com; LIMA, Edilson Carlos Silva, edilsonlima3@gmail.com; ALMEIDA, Will Ribamar Mendes, will.mendes@ceuma.com.br.}



d.o.i.: 10.29327/5192626.1-4

Resumo

Para demonstrar a iniciativa do coworking de uma instituição fez–se necessário criação de um site, este com objetivo de alcançar o público-alvo e expor projetos feitos por empresas junior que dele fazem parte. Para tal fim pode se usar várias tecnologias, mas no artigo em questão, para compor este objetivo será demonstrado a realização de um backend utilizando-se do *Framework Express* e da Plataforma *Node JS* para implementar uma *API REST* que será usada para compor um *Frontend*.

Palavras-chave: API REST, NodeJS, Prisma.

Abstract

To demonstrate the coworking initiative of an institution, it was necessary to create a website, with the objective of reaching the target audience and exposing projects made by junior companies that are part of it. For this purpose, several technologies can be used, but the article in question will demonstrate the achievement of this objective using the Express Framework and the Node JS Platform to implement an API REST that will be used to compose a Frontend.

Keywords: REST API, NodeJS, Prisma.

1. INTRODUÇÃO

O *coworking* desta instituição tem a iniciativa de ser um espaço voltado para formentar o empreendedorismo nos alunos e possibilitar um local agradável e estruturalmente adequado para aproximar o mercado de trabalho dos alunos. Como ferramentas indispensáveis para este propósito têm - se a divulgação e a criação de um local de alcance digital, que são necessários para apresentar o ambiente, as empresas envolvidas e os projetos já realizados para que sejam abertas novas possibilidades para que o público-alvo das empresas seja alcançado. Para a criação do acervo que reúne o material que envolve o *coworking* da instituição e para que ele seja encontrado em buscas onlines, faz-se necessário a criação de um site para a realização do intermédio digital entre as empresas juniors e os interessados.

Para o desenvolvimento do site do *coworking,* e como tópico de abordagem principal do artigo, foi realizada a construção de uma *API REST*, utilizando JavaScript como linguagem de programação, o *ORM* Prisma para modelagem do banco*,* a plataforma *NodeJS* e o *framework Express* para o a construção dos *endpoints* que precisam ser usados pelo *Frontend*.

Neste artigo vamos abordar alguns tópicos importantes, sendo eles: a fundamentação teórica na seção 2. A seção 3 a metodologia. A seção 4 mostrará as principais funcionalidades do sistema e os resultados obtidos nos testes do sistema. Na seção 5 será feito a conclusão.

2. FUNDAMENTAÇÃO TEÓRICA

Nesta seção será apresentada uma gama de conceitos que são necessários para o entendimento do processo de desenvolvimento do *backend* do site em questão, com o objetivo de embasar os conteúdos que serão citados ao longo do artigo, os tópicos abordados estão divididos da seguinte forma: no item 2.1 UML, no 2.2 *Backend*, no 2.3 NodeJS, no 2.4 ORM Prisma, no 2.5 JavaScript e no 2.6 ExpressJS.

2.1 UML

A *Unified Modeling Language*, ou Linguagem Unificada de Modelagem é, como o nome indica, uma linguagem de notação utilizada para modelar e documentar as diversas fases do desenvolvimento de sistemas orientados a objetos (NOLETO,2020).

Para a modelagem de sistema, o UML disponibiliza vários diagramas responsáveis por demonstrar diferentes etapas do projeto. Neste sistema será utilizado o diagrama de classes e caso de uso.

2.1.1 Diagrama de Caso de Uso

Esse diagrama documenta o que o sistema faz do ponto de vista do usuário. Em outras palavras, ele descreve as principais funcionalidades do sistema e a interação dessas funcionalidades com os usuários do mesmo sistema (LEANDRO, 2012).

Este é um diagrama bem simples onde se demonstra o comportamento do sistema

por meio do uso de atores que representam os usuários do sistema que são interligados com as funções na qual podem interagir.

2.1.2 Diagrama de Classe

Um diagrama de classes ilustra um conjunto de classes, interfaces, colaborações e respectivas relações, em geral de dependência, generalização e de associação (ALBERTO; CARLOS, 2001)

Este diagrama faz parte dos modelos de tipo estruturais e é essencial para os outros diagramas, pois o diagrama de classes é responsável por esquematizar as classes com seus respectivos métodos, atributos e o tipo de relação entre as classes.

2.2 *Backend*

Backend é toda a parte da programação voltada ao funcionamento interno de um *software* O *backend* serve para que os sites, aplicativos, *softwares* ou outros tipos de sistemas de informação tenham todas as suas funcionalidades operando de maneira efetiva e cumprindo seus objetivos (ROVEDA, 2021).

Em alguns casos o *backend* pode ser responsável por algumas atividades no *frontend*, mas em termos de web quando em foco o *backend* cuida somente da parte funcional, implementando API's para que seja possível realização da correta comunicação entre as ferramentas do site com o banco de dados, implementar arquiteturas ou utilizar de plataformas para lidar com escalabilidade de requisições e entre outros.

2.2.1 Api

Existem lindas definições do que é uma API, e elas variam de acordo com o contexto, mas uma das definições mais claras que tenho é que se uma interface de um sistema é criada para que um usuário possa usa-la, a api é desenvolvida para que um sistema possa usar as funcionalidades de outro sistema (WEBER, 2018).

Considerando o descrito por Weber, podemos imaginar a API como uma ferramenta mediadora, que abstrai a entrega de dados de um outro sistema e permite que um conjunto de dados possa ser usado independentemente da plataforma, assim criando um ambiente de desenvolvimento que possibilita reutilização de recursos já criados, por vários tipos de clientes.

Durante a construção de API's, um programador modela e estabelece formas de acesso e funcionalidades, bem como a segurança para tal, sua estrutura varia de acordo com o contexto e necessidade, mas de modo geral é feito uso de arquiteturas que visam criar padrões de criação para que realmente seja possível utilizar a ferramenta pelas diferentes tecnologias presentes no mercado, de exemplo temos o REST, SOAP e entre outros.

2.2.2 Rest

Segundo Souza (2020), REST, ou *Representational State Transfer*, é um estilo arquitetônico aplicado para fornecer padrões entre sistemas de computador na web, facilitando a comunicação entre eles.

Esse estilo arquitetônico foi introduzido por Roy T. Fielding em sua dissertação de doutorado. Como definição de REST, segundo Fielding (2000, p.76, tradução nossa), podemos considerar que é um estilo híbrido derivado de vários estilos arquiteturais baseados em rede, que combinados com restrições adicionais e definem uma interface única para conexão. Segue abaixo listagem das restrições a se seguir no modelo REST.

É necessário que adotado uma única forma para que seja realizada conexões ou requisições entre o cliente e servidor, segundo Allamaraju (2010, p.3), é usado uma interface que consiste dos métodos *OPTIONS, GET, HEAD, POST, PUT, DELETE* e *TRACE*, onde cada método da interface opera com apenas um recurso, não mudando seu comportamento, nem sua sintaxe independentemente da aplicação ou do recurso que está utilizando o método.

Estabelecer uma conexão livre de estados onde todas as requisições tenham os dados necessários para se completar as chamadas

Segundo REDHAT (2022), ter um sistema em camadas que organiza os tipos de servidores (responsáveis pela segurança, pelo carregamento de carga e assim por diante) envolvidos na recuperação das informações solicitadas em hierarquias que o cliente não pode ver. Segundo ROSA (2022), a interface de usuário do site da web/aplicação deve ser separada da solicitação/armazenamento dos dados, de modo que cada parte possa ser dimensionada individualmente.

2.2.3 Consumo de API

As APIS têm estruturas que variam de acordo com a necessidade, mas geralmente seguem um esquema com construção parecida com a da figura 1:

Figura 1 – esquema de uma API

Fonte: Autoral, 2022

Na figura 1, acima temos um esquema que demonstra as relações de uma API simples, para melhor entendimento podemos imaginar uma seguinte situação, o computador faz o papel de um site que precisa de dados presente num banco de dados que tem acesso regido por *endpoint* numa API, cada acesso é feito por uma requisição em uma URI com o método HTTP e supondo que seja uma API REST, podemos imaginar que num cenário que só precisa – se dos dados fornecidos pela API, usa-se o método *GET* para obter os dados no banco, e como segundo Maniero (2022) define, para consumir uma API basta escrever um código que faça uma requisição na própria api, ou seja o fato de simplesmente realizar uma requisição funcional via código no formato da API, podemos dizer que já estamos consumindo-a.

2.3 NodeJS

Segundo Pessoa (2022), JavaScript nasceu para atender demandas voltadas ao *Front* e como as necessidades aumentam de acordo com o crescimento tecnológico, surgiu a

ideia de utilizar uma mesma linguagem no lado do cliente e do servidor para otimizar processos e serviços. Dessa forma, o Node.JS aparece como uma alternativa viável para programação *backend* por se tratar de um ambiente para desenvolvimento utilizando a linguagem JavaScript.

2.4 Prisma ORM

Segundo PET (2022), *Object-Relational Mapping* (ORM), em português, mapeamento objeto-relacional, é uma técnica para aproximar o paradigma de desenvolvimento de aplicações orientadas a objetos ao paradigma do banco de dados relacional. O uso da técnica de mapeamento objeto-relacional é realizado através de um mapeador objeto-relacional que geralmente é a biblioteca ou *framework* que ajuda no mapeamento e uso do banco de dados.

O Prisma substitui os ORMs tradicionais e as camadas de acesso a dados personalizados por uma abstração de banco de dados universal usada por meio do cliente Prisma (ELSANGEDY, 2022).

2.5 JavaScript

O JavaScript é uma linguagem de programação de computadores dinâmica usada nos navegadores da web para controlar o comportamento de páginas web e interagir com utilizadores. Permite a comunicação assíncrona e pode atualizar partes de uma página web ou mesmo substituir o conteúdo por inteiro de uma página web. Poderá o JavaScript ser usado para visualizar informação de data e hora efetuar animações num site web, validar *input* de formulários, sugerir resultados à medida que o utilizador escreve na caixa de pesquisa, e muitos mais (DIMES, 2015).

O JavaScript quebrou o padrão de criação de projetos web, pois implementou a dinamicidade nos sites o que aumentou a interatividade com o usuário e modernizou os sites que temos hoje, pois anteriormente tinha-se somente páginas estáticas.

2.6 ExpressJS

Para Santos (2022), o *Express*.js é um *framework node* que pode ser comparado com o Laravel para PHP, ele cria abstrações de rotas, *middlewares* e muitas outras funções para facilitar a criação tanto de API's quanto SPA's. Um ótimo exemplo de uso dele é a exposição de uma API simples de *get* que pode ser feita com poucos cliques em menos de 10 minutos.

3. METODOLOGIA

Para construção do site foi necessário a criação da API REST, onde buscou-se métodos para criação rápida e eficiente da mesma para o correto funcionamento das requisições. Como forma de demonstrar processos utilizados, usou-se de pesquisa exploratória com a aplicação de um estudo de caso da API expondo processos da mesma com uma abordagem qualitativa, onde será feita comprovação de inserção e de resposta de dados da api por meio de testes de consumo, no estudo de caso foi usado UML para levantar dados para o banco e para imaginar situações de uso, como representação das rotas criadas

os *endpoints*, e como forma de fazer a análise qualitativa de dados manipulados por api, foram realizados testes de consumo pelo *postman*.

4. ESTUDO DE CASO

Nesta seção será abordado o desenvolvimento do *backend* do site para solução do problema, abordando a implementação da *API REST* para compor o *frontend* que consome os dados armazenados no banco, com auxílio do prisma e plataforma NodeJS para conseguir fazer o *backend* com o JavaScript.

4.1 Análise da Aplicação

Para continuar o desenvolvimento do projeto foi necessário levantar os requisitos necessários, fazendo uma sondagem por características e dados relevantes para a *API REST*. Durante esse processo foi realizado uso de diagramas *UML* e os mesmos serão demonstrados no próximo tópico.

4.1.2 Diagrama de Caso de Uso

Como uma forma de desenhar o comportamento entre os agentes que interagem com o sistema, foi usado um diagrama de caso de uso, que está na figura 1 a seguir:

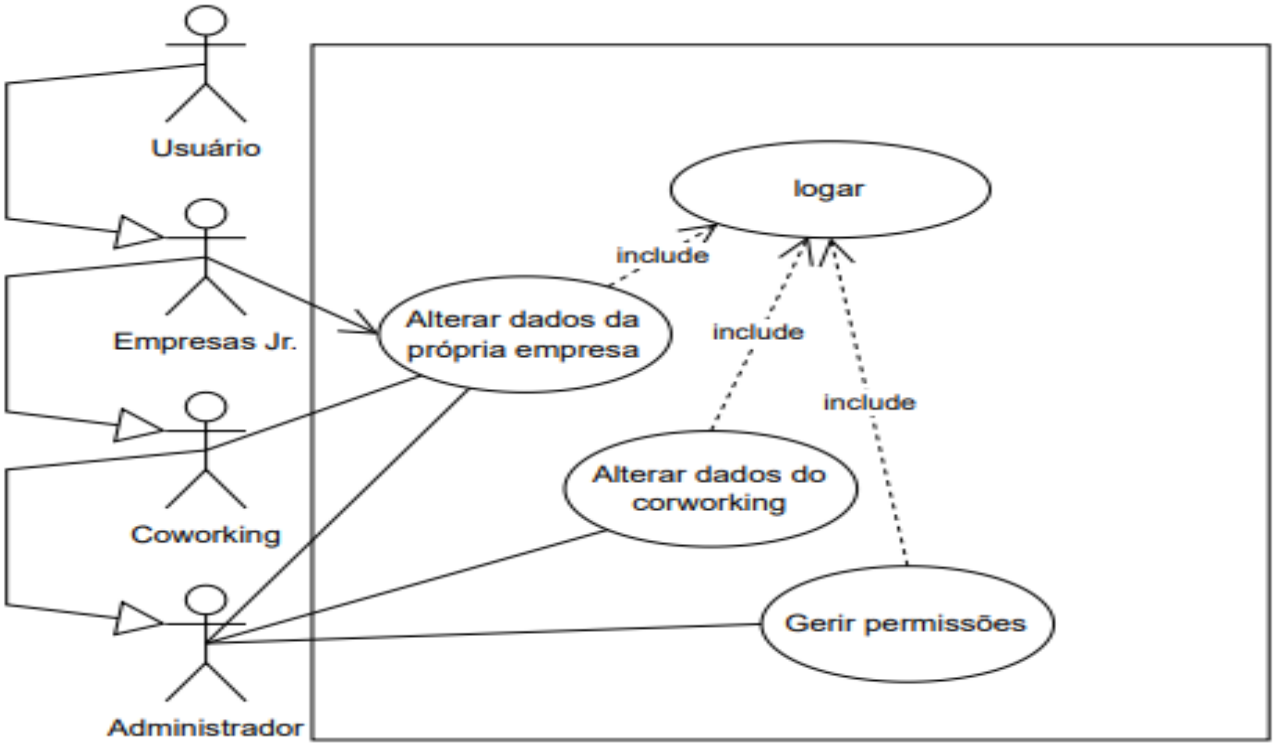


Figura 2 - Diagrama de caso de uso

Fonte: Autoral, 2022

Com o intuito de detalhar cada caso de uso especificado na figura 1, abaixo segue descrição dos autores e suas respectivas interações:

Usuário: Um usuário pode navegar normalmente no site, e não precisa estar logado

para realizar esta ação.

- EmpresasJR: Este ator estende de usuário, ou seja, tem todas as características que ele, adicionado de a possibilidade de poder editar informações da própria empresa jr. Tendo como requisito obrigatório estar logado para visualizar as informações do site.
- Coworking: Este ator herda todas as funções e restrições dos atores anteriores, contendo o adicional de poder editar todas as informações exibidas na página.
- Administrador: Assim como os demais atores, este herda funções e restrições adicionando a ação de definir as permissões que definem a hierarquia no poder de edição do site atribuída aos usuários.

4.1.2 Diagrama de Classe do Sistema

A figura 2 a seguir mostram as classes do sistema, contendo seus atributos e também suas relações. Há um total de 7 classes que após a figura serão descritas.

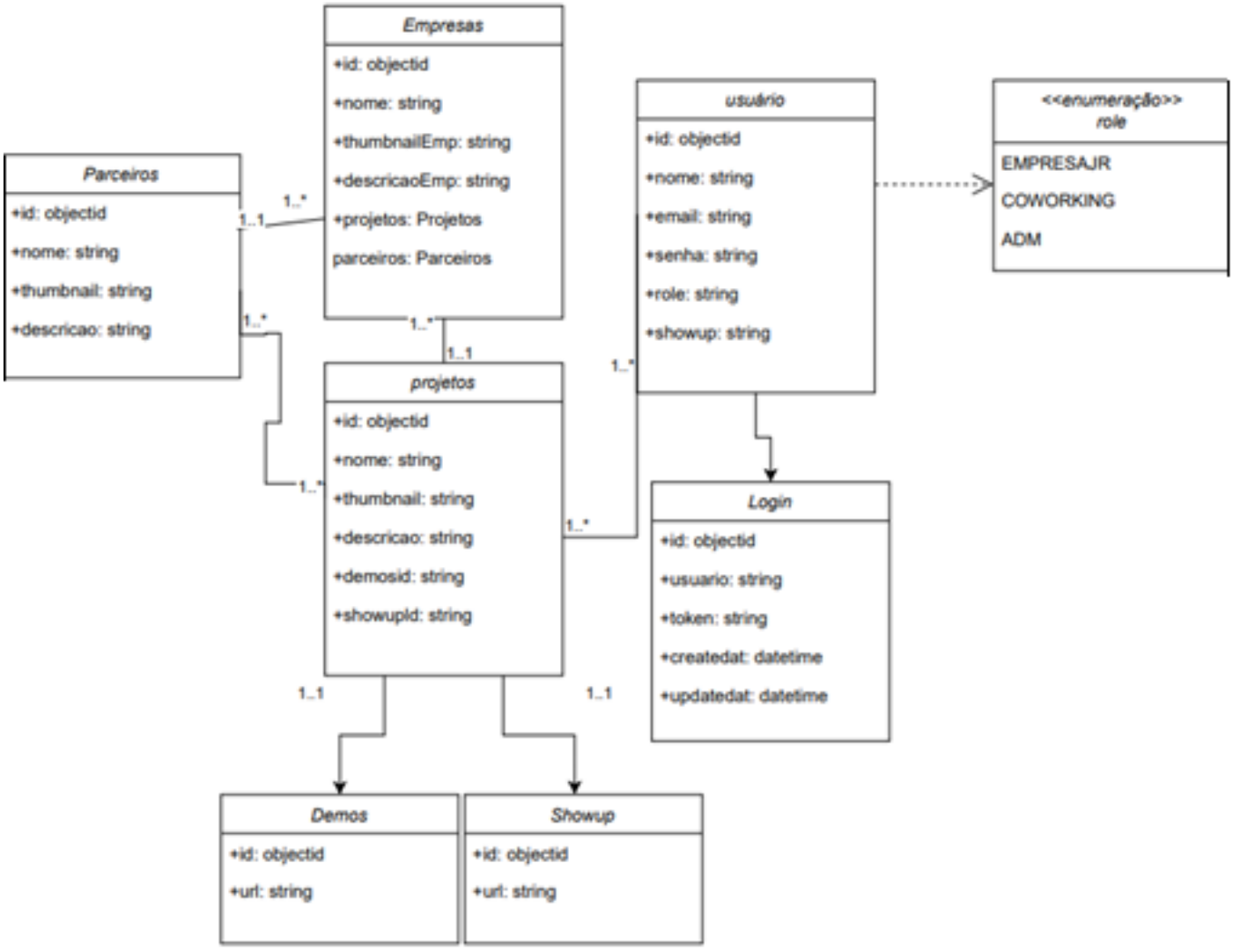


Figura 3 - Diagrama de classe

Fonte: Autoral, 2022

A classe "Usuário" contém todos os atributos necessários para o uso dos dados do usuário, como o usuário conseguir fazer login no *frontend*, criar conta e etç...

A classe "Projetos" contém a formatação dos dados dos projetos das empresas juniors e como é possível ver na figura 1, há uma associação entre a classe projetos e a as classes "Demos" e "ShowUp"

A classe "*Login*" possui informações necessárias para guardar cada *token* do usuário para permitir que o usuário consiga logar em diferentes plataformas, sistemas e diferentes locais.

A classe "Parceiros" contém os dados que são necessários para demonstrar parceiros do coworking no Frontend.

A classe "Empresas" possui informações referentes as empresas que participam do coworking.

4.2 Utilizando o ORM prisma

Este ORM propõe uma camada grande de abstração, e como já citado ele faz um intermédio entre bancos de dados e a orientação objeto, e não se faz necessário criação manual do banco, sua representação de classes se dá por um modelo que mapeia os dados para as tabelas num banco. Com intuito de economizar tempo usando os recursos mencionados para uma rápida criação da API, foi utilizado seu modelo para criação das tabelas nos bancos de dados MongoDB, e elas foram feitas de acordo com os seguintes esquemas:

```
model Projetos {
  id         String   @id @default(auto()) @map("_id") @db.ObjectId
  nome       String
  descricao  String
  thumbnail  String
  demos      Demos[]
  showUp     ShowUp[]
  empresaId  String?   @db.ObjectId
  empresa    Empresas? @relation(fields: [empresaId], references: [id])
}
```

Figura 4 – Esquema Projetos

Fonte: Autoral, 2022

No exemplo da figura 4, temos a criação do modelo Projetos que recebe um campo único de id que é a chave primária, nesse caso no modelo foi definido que o MongoDB irá gerar os ids automaticamente, dentro do modelo temos relações com outras tabelas, sendo essa relação dada pelo atributo empresaId.

```
model Empresas {
  id           String      @id @default(auto()) @map("_id") @db.ObjectId
  nome         String
  descricaoEmp String
  thumbnailEmp String
  projetos     Projetos[]
  parceiros    Parceiros[]
}
```

Figura 5 – Esquema Empresas

Fonte: Autoral, 2022

No exemplo da figura 5, temos a criação do modelo Empresas que recebe três campos do tipo String: nome, descricaoEmp, thumbnailEmp. Também há dois campos com coleções de dados, são eles: projetos e parceiros.

```
model Parceiros {
  id         String   @id @default(auto()) @map("_id") @db.ObjectId
  nome       String
  thumbnail  String
  descricao  String
  empresa    Empresas? @relation(fields: [empresaId], references: [id])
  empresaId String?   @db.ObjectId
}
```

Figura 6 – Esquema Parceiros

Fonte: Autoral, 2022

Na figura 6, temos a criação do modelo Parceiros que recebe três campos do tipo String: nome, descriçao, *thumbnail*. Nesse caso, há uma relação com a tabela empresa através do atributo empresaId. Assim como, o modelo da figura 4.

```
model Usuario {
  id     String   @id @default(auto()) @map("_id") @db.ObjectId
  nome   String
  email  String
  senha  String
  sessao Sessao[]
  Role   Role     @default(EMPRESAJR)
}
```

Figura 7– Esquema Usuário

Fonte: Autoral, 2022

A figura 7 mostra a criação do modelo Usuário que recebe três campos do tipo String: nome, e-mail e senha. Também recebe uma coleção assim como na figura 5, com o diferencial de uma enumeração que é responsável pela permissão inicial do usuário cadastrado no site de acordo com os dados da figura 8.

```
enum Role {
  ADM
  USER
  EMPRESAJR
  COWORKING
}
```

Figura 8 – Esquema Role

Fonte: Autoral, 2022

Na figura 8, tem-se a criação da enumeração role, com os possíveis valores de permissões dos usuários do site.

4.3 *Endpoints*

Os *endpoints* são caminhos ou URI'S utilizados para acessar diferentes rotas e requisições para a API, como os exemplos da tabela 1, utilizando do formato representativo JSON e XML para que seja possível entendimento dos dados com requisições do tipo PUT ou POST que necessitam obrigatoriamente que seja passado um *body*, utilizando também uma rota de identificação que segue a seguinte ordem: protocolo, domínio e rota.

Tabela 1. Exemplos de requisições HTTP e suas finalidades.

URI	Requisição	Descrição da requisição
http://domain/api/empresas	POST	Cadastra uma empresa
http://domain/api//projetos	GET	Lista os projetos cadastrados
http://domain/api//parceiros/:id	PUT	Editar os parceiros cadastrados
http://domain/api//parceiros/:id	DELETE	Excluir um parceiro cadastrado

Fonte: Autoral, 2022.

Os *endpoints* que foram exemplificados na tabela 1 fazem parte das possíveis requisições que podem ser chamadas durante a chamada da API por outro sistema, o que resultará no consumo da mesma.

4.4 Testes de Consumo da API

Com o intuito de demonstrar a realização das requisições para atestar funcionamento da api, e como os dados serão exibidos no site, neste tópico será usado o programa postman para realização do uso de protocolos HTTP para fazer as chamadas ou inserções de dados no *endpoint* referente a projetos, como o site exibe as informações cadastrados no banco via API, será feito requisições que mostrarão o retorno no formato JSON e será relacionada com a exibição da informação no *frontend* que sempre mostrará o conteúdo visualmente por meio de GET'S.

A aba de projetos na rota de api contém os métodos de acesso e manipulações de dados da mesma, realizando um *get* no *endpoint* referente a esta função, obtemos os dados mostrados na figura 9 que segue abaixo:

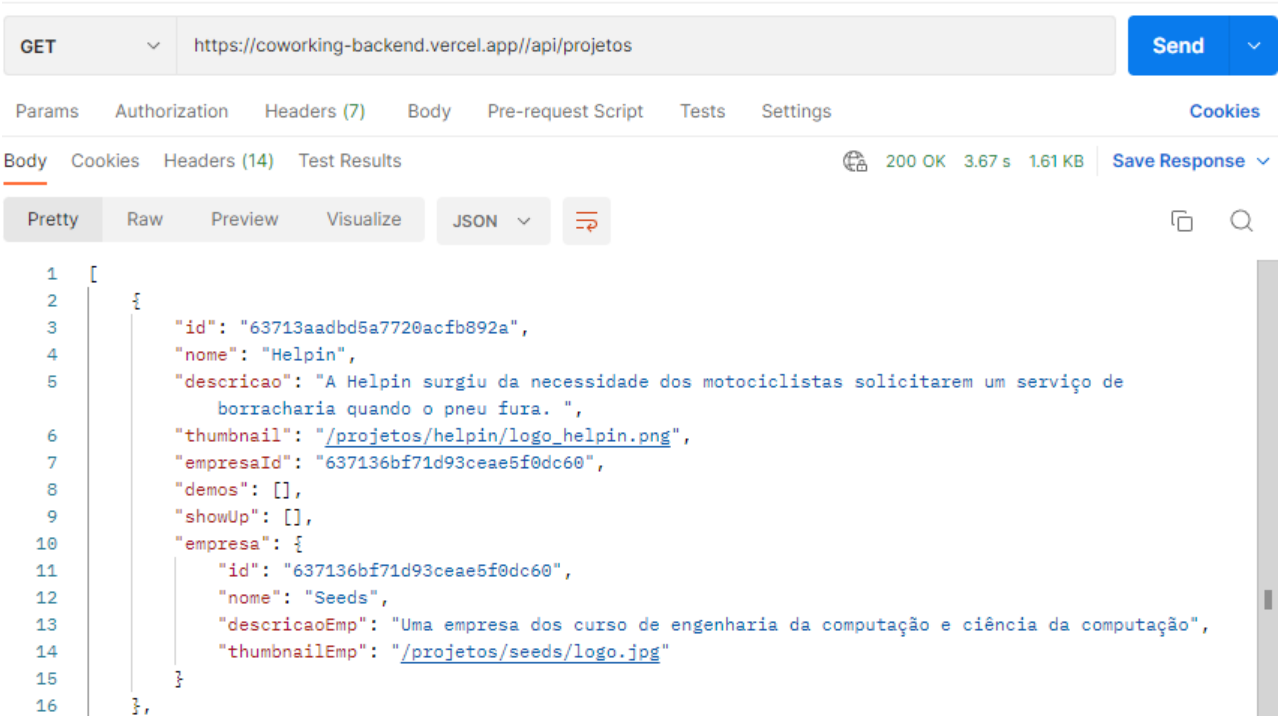


Figura 9 – Representação de requisição GET em projetos

Fonte: Autoral, 2022

Na figura 9, podemos visualizar os dados de projetos cadastrados no banco por meio do método *GET*, e confirmando a exibição de como o dado é exibido no ato do consumo da api por utilização do *endpoint*, temos a representação na figura 10:

Figura 10 – Frontend mostrando dados em projetos

Fonte: Autoral, 2022

A figura 10, mostra o consumo da API pelo *frontend* dos dados dos projetos cadastrados. É possível notar também que há um nome e uma descrição, e é possivel notar também que há a imagem da empresa dona do projeto e o nome dela, mostrando visualmente o relacionamento que há entre projetos e empresas que foi explicitado na figura 4.

Após realizar uma requisição POST, e obter os dados enviados pela requisição por meio de um GET, temos a representação na figura 11 a seguir:

```
"id": "63724678f1ba3d492581c76b",
"nome": "Nome de Exemplo",
"descricao": "O exemplo é um teste para demostração da funcionalidade dos projetos. ",
"thumbnail": "/teste.png",
"empresaId": "637136bf71d93ceae5f0dc60"
```


Figura 11 – Resposta da requisição do GET de um projeto criado

Fonte: Autoral, 2022

Ao realizar a requisição POST obtém-se como retorno os dados do projeto que foram criados no banco de dados, essa criação pode ser vista no *frontend* do site como a figura 13 mostra.

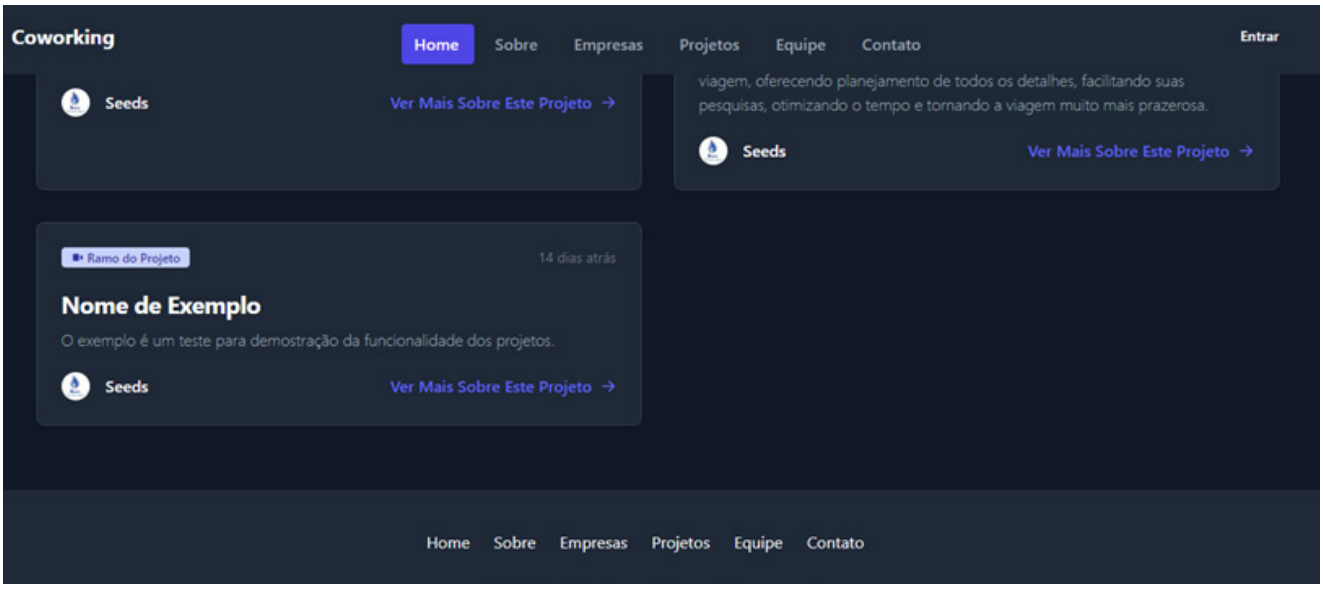


Figura 13 – Front-end com o novo projeto criado

Fonte: Autoral, 2022

Na figura 13 podemos observar que foi criado e exibido um novo projeto no *frontend* com os dados cadastrados pela requisição POST.

Com o intuito de realizar uma alteração, foi realizado um método PUT no projeto, criado com a requisição da figura 11, segue abaixo resposta obtida no ato de empregar o método http para edição na figura 14:


```
"nome": "Nome de Exemplo (Alterado)",
"descricao": "O exemplo é um teste para demostração da funcionalidade dos projetos. ",
"thumbnail": "/teste.png",
"empresaId": "637136bf71d93ceae5f0dc60"
}
```


Figura 14 – Resposta da requisição do PUT projetos

Fonte: Autoral, 2022

Na figura 14 é feito uma edição que adiciona ao nome a palavra alterado para demonstrar que realmente foi feita a mudança que pode ser vista no *frontend* como mostra a figura 15.

Figura 15 – Representação da alteração no *frontend*

Fonte: Autoral, 2022

Após uso do PUT realizado como visto na figura 14, podemos ver na figura 15 que foi realizada mudança na exibição do nome do projeto como bem foi especificado na requisição.

5. CONCLUSÃO

Neste trabalho foi concluída uma API Rest para um site de coworking de um espaço acadêmico. De acordo com os testes realizados, os dados demonstraram que foi possível realizar os métodos de requisição http com sucesso e fazendo uma criação simples proporcionada pelo ORM e o *framework express*.O site feito durante o artigo provê alcance para a iniciativa e pôs se presente edição de dados a ser exibidos diretamente pela comunicação com as funções implementadas pela API. O Sistema apresentará o *coworking* e proporcionará maior possibilidade de crescimento de ofertas para as empresas por meio da exposição do ambiente, bem como do contato para com os envolvidos que também foi feito pelo site.

Dos conhecimentos usados durante a graduação para realização deste projeto destacam -se os conceitos relacionados a gestão de projetos, engenharia de software, orientação a objetos e metodologia científica.

Como trabalho futuro fica aberto a possibilidade do próprio usuário administrador ter mais liberdade de criação dentro do site para apresentar empresas como bem desejado.

AGRADECIMENTOS

Agradeço primeiramente a minha mãe Marinalva por todo o incentivo, apoio e forças passadas durante a escrita do trabalho, bem como minha namorada Milena que assim também fez, ao professor Edilson pela orientação, apoio e confiança. E não poderia deixar de mencionar meus grandes agradecimentos aos amigos Victor Freire e Afonso Jansen, que não só me auxiliaram no projeto, são amigos que ajudaram durante toda a faculdade.

Referências

ALLAMARAJU, S. **RESTful Web Services Cookbook**. Sebastopol, CA, USA: O'Reilly Media, 2010. Acesso em: 13 nov. 2022.

Cadenhead, Rogers; LEMAY, Laura**, Aprenda em 21 dias Java 2. 4. ed. São Paulo**: Campus, 2005.

CIRIACO, Douglas. O que é API?. **Tecmundo,** 2009. Disponível em: https://www.tecmundo.com.br/programacao/1807-o-que-e-api-.htm. Acesso em: 12 nov. 2022.

com/lessons/whatisrest.html#> Acessado em: Fev 2013.

DIMES, T. **JavaScript: Um Guia para Aprender a Linguagem de Programação JavaScript**. [s.l.] Babelcube Inc., 2015.

ELSANGEDY, M. A. **Primeiros passos com Prisma - Codengage - Medium**. Disponível em: <https://medium.com/codengage/primeiros-passos-com-prisma-df6633464417>. Acesso em: 13 nov. 2022.

FREDRICH, T. **REST API Tutorial**. Disponível em: <http://www.restapitutorial.com> Acesso em: 13 nov. 2022.

LEANDRO. **O que é UML e Diagramas de Caso de Uso: Introdução Prática à UML**. Disponível em: <https://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408>. Acesso em: 10 nov. 2022.

LEITE, Thiago. **Orientação a objeto: aprenda seus conceitos e suas aplicabilidades de forma efetiva**. 2016, capítulos 4.2 e 6.1.

MANIERO. **O que significa o termo "Consumir uma API"?** Disponível em: <https://pt.stackoverflow.com/questions/164407/o-que-significa-o-termo-consumir-uma-api>. Acesso em: 13 nov. 2022.

MANUEL, A.; ALBERTO, C. **UML, metodologia e ferramentas CASE**. 2001.

NOLETO, Cairo. **UML: o que é, para que serve e quando usar essa linguagem de notação?** Disponível em: <https://blog.betrybe.com/tecnologia/uml/>. Acesso em: 9 nov. 2022.

O que é back end, para que serve e como aprender em 2021. Disponível em: <https://kenzie.com.br/blog/back-end/>. Acesso em: 9 nov. 2022.

O que é um diagrama UML? Disponível em: <https://www.lucidchart.com/pages/pt/o-que-e-uml>.

PARR, K. **The Four Pillars of Object-Oriented Programming**. Disponível em: <https://www.freecodecamp.org/news/four-pillars-of-object-oriented-programming/>.

PESSOA, C. **Node.JS: definição, características, vantagens e usos possíveis**. Disponível em: <https://www.alura.com.br/artigos/node-js-definicao-caracteristicas-vantagens-usos>. Acesso em: 10 nov. 2022.

PET. **O que é e para que serve um ORM?** Disponível em: <https://www.ufsm.br/pet/sistemas-de-informacao/2022/05/23/orm/#:~:text=O%20Prisma%20%C3%A9%20um%20ORM,modelagem%20e%20migra%C3%A7%C3%A3o%20de%20dados.>. Acesso em: 13 nov. 2022.

Programação orientada a objetos : Curso técnico de informá tica / Victorio Albani de Carvalho, Giovany Frossard Teixeira. – Colatina: IFES, 2012.

ROSA, D. **Tutorial de APIs REST – client REST, serviços REST e chamadas de API explicados com exemplos de código**. Disponível em: <https://www.freecodecamp.org/portuguese/news/tutorial-de-apis-rest-client-rest-servicos-rest-e-chamadas-de-api-explicados-com-exemplos-de-codigo/>. Acesso em: 13 nov. 2022.

SANTOS, L. **Pra que server o expressjs?** Disponível em: <https://pt.stackoverflow.com/questions/149296/pra-que-server-o-expressjs>. Acesso em: 13 nov. 2022.

SOUZA, Ivan . Saiba o que é REST (Representational State Transfer) e como usá-lo neste tutorial. **Rock Content,** 2020. Disponível em: https://rockcontent.com/br/blog/rest/. Acesso em: 12 nov. 2022.

WEBER, Vanessa; FROÉS, Gabriel . API // Dicionário do Programador. **Código Fonte TV,** 2018. Disponível em: https://www.youtube.com/watch?v=vGuqKIRWosk&ab_channel=C%C3%B3digoFonteTV. Acesso em: 12 nov. 2022.

5

O USO DO Next.js NA CRIAÇÃO DE UM FRONTEND PARA UM COWORKING CONSUMINDO UMA API REST

USING Next.js TO CREATE A FRONTEND TO A COWORKING CONSUMING A REST API

Victor Rodrigues Freire[1]

Edilson Carlos Silva Lima[2]

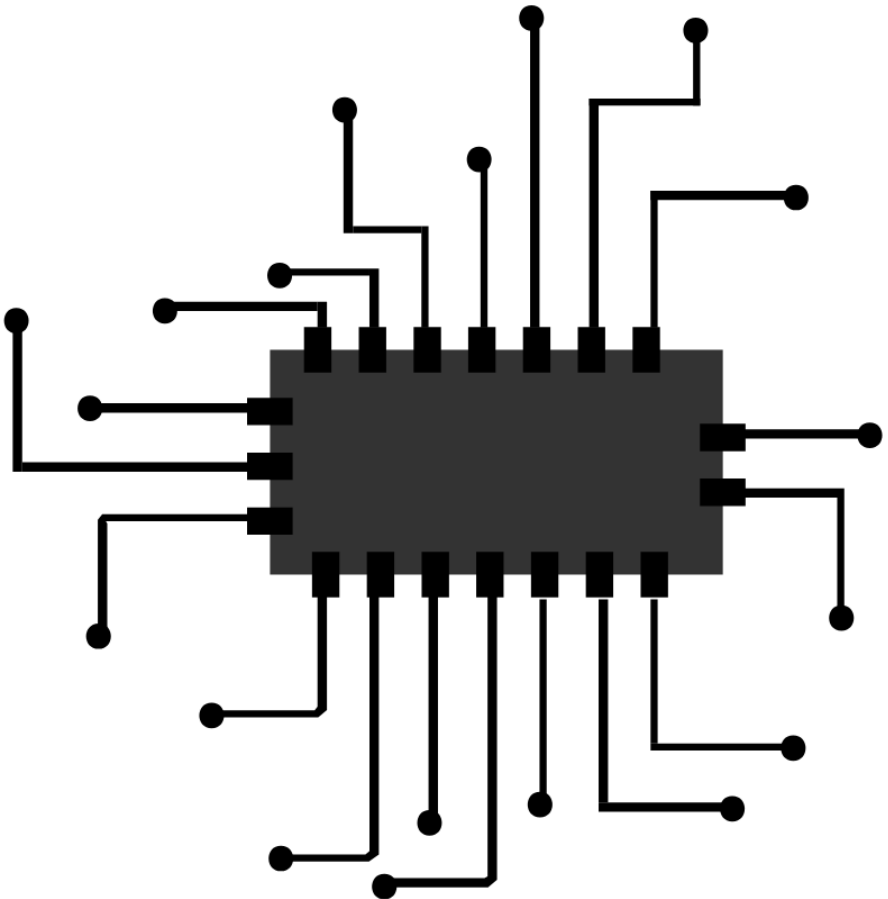
Will Ribamar Mendes Almeida[3]

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

{FREIRE, Victor Rodrigues, vic.frei2@icloud.com; LIMA, Edilson Carlos Silva, edilsonlima3@gmail.com. ALMEIDA, Will Ribamar Mendes, will.mendes@ceuma.com.br.}



d.o.i.: 10.29327/5192626.1-5

Resumo

Um coworking gostaria de divulgar os seus projetos e empresas. Com isso, houve a necessidade de utiliza-se uma ferramenta que cumprisse com esse propósito. Por tanto, foi iniciado a construção do site para mostrar as empresas júniores e os projetos de cada empresa do *coworking*, com o objetivo de obter mais notoriedade e visibilidade dos mesmos, e assim resolver o problema. Com isto, esse artigo tem como objetivo mostrar o uso do *framework* do React chamado Nextjs, junto com o estilizador Tailwind junto a componentização do Flowbite, utilizando-se da métodologia "*Atomic Design*" e utilizando-se de um backend baseado na *API REST*, e com isso a união de todas as tecnologias junto com o método utilizado obter a resolução do problema proposto.

Palavras-chave: *Atomic Design, Web, Hardware, Software, Framework, API REST, Nextjs.*

Abstract

A coworking space would like to publicize its projects and companies. With that, there was a need to use a tool that fulfilled this purpose. Therefore, the construction of the website was started to show the junior companies and the projects of each coworking company, with the objective of obtaining more notoriety and visibility of them, and thus solving the problem. With this, this article aims to show the use of the React framework called Nextjs, along with the Tailwind styler along with Flowbite componentization, using the "Atomic Design" methodology and using a backend based on the REST API, and with that the union of all the technologies together with the used method obtains the resolution of the proposed problem.

Keywords: *Atomic Design, Web, Hardware, Software, Framework, API REST, Nextjs.*

1. INTRODUÇÃO

Um site pode divulgar e mostrar diversas informações para diferentes pessoas, o *coworking* em necessidade de divulgar as suas empresas juniores e os seus trabalhos (projetos) de cada empresa junior, necessitou de um site para esse propósito, o site é uma das principais ferramentas de divulgação, assim como as redes sociais, e diferente de uma rede social tem a possibilidade de obter um maior alcance nas indexações em sites de busca, como o google, se bem divulgado. Diante disso, um site para o *coworking* foi idealizado com o intuído de resolver o problema de divulgação das empresas juniores e seus projetos.

Para o desenvolvimento do site do *coworking* foram utilizadas tecnologias com Nextjs (Um *framework* do React), utilizando a linguagem Javascript, sendo utilizado como *backend* uma API REST. O site tem o propósito de mostrar os projetos e as empresas juniores, para melhor divulgação e notoriedade dos mesmos.

Neste artigo vamos abordar alguns tópicos importantes, sendo eles: a fundamentação teórica e metodológica no capítulo 2 deste artigo, no capítulo 3 discorreremos do site web que foi desenvolvido para auxiliar na problemática escrita na introdução, no capítulo 4 falaremos sobre os resultados e discussões, no capítulo 5 sobre a conclusão e trabalhos futuros, no capítulo 6 ressalto meus agradecimentos e por fim no capítulo 7 encontram-se as referências utilizadas para a construção deste artigo.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, vamos abordar uma revisão literária dos assuntos essenciais para o desenvolvimento de um site com o desenvolvido tendo ênfase em Nextjs e está dividido nos seguintes itens: 2.1 Web, 2.2 Javascript, no item 2.3 React, no item 2.4 Front End

2.1 Web

A transformação digital é irreversível. Em 20 anos, a internet mudou o mundo mais rápido do que todo o século XX foi capaz de fazer (MACCEDO, 2017).

Nos últimos 10 anos, o processo de mudança acelerou ainda mais: em 2007 o primeiro iPhone foi lançado. Wi-fi era uma raridade na época. iPad só surgiu em 2010. Banda larga era cara demais e disponível em pouquíssimos lugares. Algo como 3G ou 4G não faria o menor sentido há 10 anos (MACCEDO, 2017).

A noção de cloud (nuvem de dados) ainda era incipiente e ninguém imaginava acessar a sua conta do banco através de um aplicativo (MACCEDO, 2017).

A Web pode ser também espaço de autopromoção, de organização de círculos sociais, de entretenimento, sendo incrível seu alcance, seja a nível regional ou mundial. Apesar de estar cada vez mais sofisticada, com recursos gráficos poderosos, sons, vídeos, efeitos visuais, a Web nem sempre foi assim (SEHN, 2018).

2.2 Javascript

Se você já conhece outras linguagens de programação, talvez ajude saber que Ja-

vascript é uma linguagem de alto nível, dinâmica, interpretada e não tipada, conveniente para estilos de programação orientados a objetos e funcionais (FLANAGAN, 2013).

JavaScript é a linguagem de script do lado do cliente mais amplamente utilizada. Ela é realmente poderosa e dinâmica. Outras populares linguagens de script do lado do cliente incluem VBScript e Python. Linguagens de script do lado do cliente são usadas para tornar seus sites mais interativos, animados e responsivos (PRESCOTT, 2016).

A sintaxe de Javascript é derivada da linguagem Java, das funções de primeira classe de Scheme e da herança baseada em protótipos de Self. Mas não é preciso conhecer essas linguagens nem estar familiarizado com esses termos para utilizar este livro e aprender Javascript (FLANAGAN, 2013).

2.3 React

O React é uma biblioteca para construção de UIs – ela ajuda você a definir a UI de uma vez por todas. Então, quando o estado da aplicação mudar, gastar metade do corpo de sua função procurando-nos do DOM; tudo que você deve fazer é manter o estado sua aplicação (com um velho objeto Javascript comum. Conhecer o React é um bom negócio, você pode criar aplicações nativas com desempenho e controles nativos (controles realmente nativos, e não cópias com aparência nativa) usando as mesmas ideias (STEFANOV, 2019).

2.3.1 Nextjs

O desenvolvimento web mudou muito nos últimos anos. Antes do advento das estruturas Javascript modernas, a criação de aplicativos da Web dinâmicos era complexa e exigia muitas bibliotecas e configurações diferentes para que funcionassem conforme o esperado. Angular, React, Vue e todos os outros frameworks permitiram que a web evoluísse muito rapidamente e trouxeram algumas ideias muito inovadoras para o desenvolvimento web front-end (RIVA, 2022).

O Nextjs é um aplicativo de página única implementa essa arquitetura para clientes da Web: o aplicativo JavaScript é iniciado a partir de uma página da Web e, em seguida, executado inteiramente no navegador. Todas as mudanças visuais no site acontecem como reação às ações do usuário e aos dados recebidos das APIs remotas (KONSHIN, 2018).

2.4 Front End

O frontend é o nome dado ao desenvolvimento da “frente”, ou seja, a parte visual do desenvolvimento, esta parte é a parte mais simples e inteligível de um desenvolvimento essa área de desenvolvimento é considerada uma das mais familiares. De acordo com Eis(2014), Foquei-me em desenvolvedores front-end porque é a área no qual os profissionais possuem maior familiaridade quando desejam entrar no mercado de web. Mesmo assim, se você é um apenas um curioso sobre o assunto, mas quer entender melhor sobre o tema, talvez este livro possa ajudar. Este guia vai apresentar os assuntos mais básicos e importantes, tentando auxiliar aqueles que já adentraram ou querem adentrar, agora, o mercado de web.

2.4.1 Api Rest

O REST estabelece um conjunto de padrões que permite fazer isso de forma eficiente e interoperável (coisas que, considerando cenários de micros serviços, por exemplo, se tornam especialmente interessantes de se possuir" (SAUDATE, 2021).

O padrão de desenvolvimento de uma API REST trabalha em cima do conceito de criação e manipulação de recursos. Esses recursos basicamente são entidades da aplicação que são utilizadas para consultas, cadastros, atualização e exclusão de dados, ou seja, tudo é baseado em manipular os dados de um recurso (RIBEIRO PEREIRA, 2016).

2.4.2 Axios

Axios é um cliente HTTP baseado-em-promessas para o node.js e para o navegador. É isomórfico (pode rodar no navegador e no node.js com a mesma base de código). No lado do servidor usa o código nativo do node.js - o modulo http, enquanto no lado do cliente (navegador) usa XMLHttpRequests (AXIOS, 2020).

2.4.3 Tailwind

Tailwind é um conjunto de classes utilitárias reutilizáveis de baixo nível que podem ser usadas como blocos de construção para criar praticamente qualquer projeto que possamos imaginar. Essa estrutura que prioriza o utilitário cobre as propriedades CSS mais importantes, mas pode ser facilmente estendida de várias maneiras. Ele pode ser usado para prototipagem rápida ou para criar projetos completos (GERCHEV, 1978).

3. DESENVOLVIMENTO DO SITE

Neste capítulo, será abordado como o site foi desenvolvido e como cada tecnologia foi utilizada para chegar ao objetivo de criar um site para o *coworking*, por tanto, os tópicos estão nas seguintes ordens: no item 3.1 relata sobre o uso do Nextjs como um facilitador para a criação de rotas, utilizando como base o React. No item seguinte o 3.2 relata sobre a organização visual do site e sua estilização utilizando um Postcss (uma folha de estilo pre-processada) chamado Tailwind e Flowbite e obtendo dados da API REST através do módulo chamado Axios.

3.1 O uso do Nextjs (Framework do React)

A metodologia empregada foi "*Atomic Design"* essa uma metodologia que separa cada parte do desenvolvimento em mini-pedaços, esse método foi utilizado em conjunto com o *framework* do React, o Nextjs, dentro do universo do React esses mini-pedaços são chamados de componentes. Com essas informações, o desenvolvimento ocorreu da seguinte forma: O Nextjs foi utilizado por fornecer uma forma mais rápida de criação de site e para a criação dos componentes presentes no desenvolvimento, esses componentes também forem combinados com os componentes do Flowbite no objetivo de desenvolver o site mais rapidamente. A escolha do Nextjs foi por causa que ele cria automaticamente as rotas baseados nas pastas que forem criadas dentro de uma pasta chamada "*Pages*", conforme a figura 1.

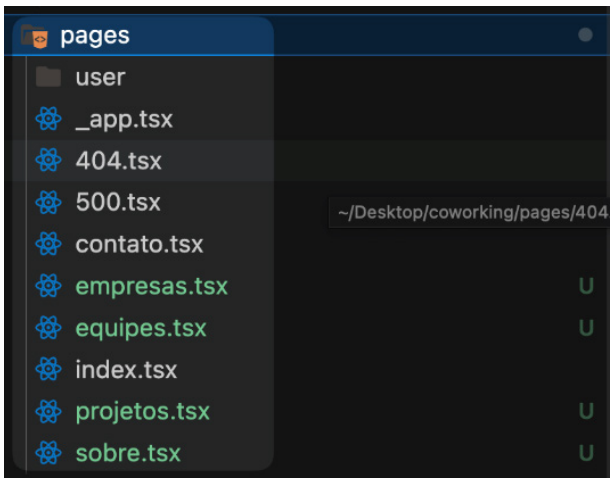


Figura 1. Pasta "*Pages*"

Fonte: Autoral, 2022

A figura 1, mostra que as rotas principais presentes no site são "user", e a rota raiz que seria "/", a rota raiz é a rota padrão do site, exemplo: https://host.com.br (rota raiz), e uma rota não raiz seria: https://host.com.br/user (rota não raiz).

É possível observar as sub-rotas do site através da figura 2:

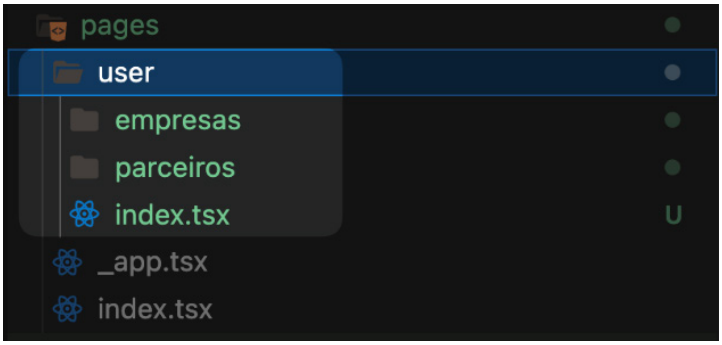


Figura 2. Sub-rotas de user

Fonte: Autoral, 2022

As sub-rotas presentes dentro da rota "user" são as seguintes, todas elas foram separadas em componentes, utilizando-se da metodologia *"Atomic Design",* um exemplo disso é que todas as rotas contém componentes, como *"_form", "create","edit","index" e "update"* cada componente tem um função especifica e juntando eles temos um layout. Essas observações podem ser observadas na figura 3, 4 e 5.

Empresas: Possuí todo o método CRUD (Create, Read, Update, Delete) para a criação de cadastro de novas empresas, edição de cada empresa, remoção de cada empresa e alteração de valores de cada empresa, como mostra a figura 3.

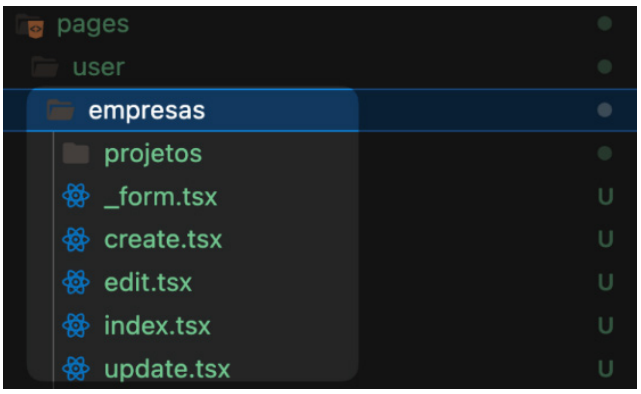


Figura 3. Rotas de empresas

Fonte: Autoral, 2022

Parceiros: Contém todo o método CRUD (Create, Read, Update, Delete) para a criação de cadastro de novos parceiros, edição de cada parceiro, remoção de cada parceiro e alteração de valores de cada parceiro, como mostra a figura 4.

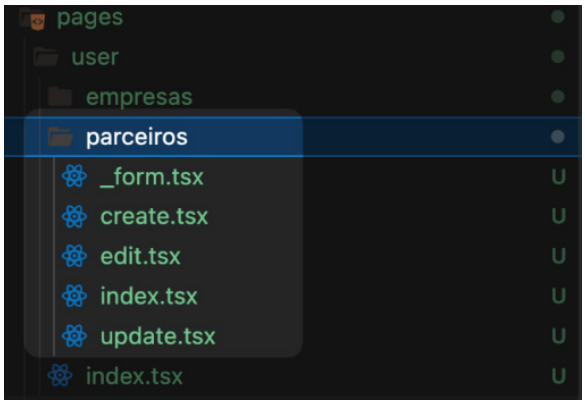


Figura 4. Rotas de parceiros

Fonte: Autoral, 2022

Todas as sub-rotas de "user" possuem um arquivo chamado "_form" o que é o arquivo onde possuí o formulario padrão para cada rota, que será utilizado para mostrar os valores nas telas dos sites e permitir que os usuários façam modificações ou inserções de valores.

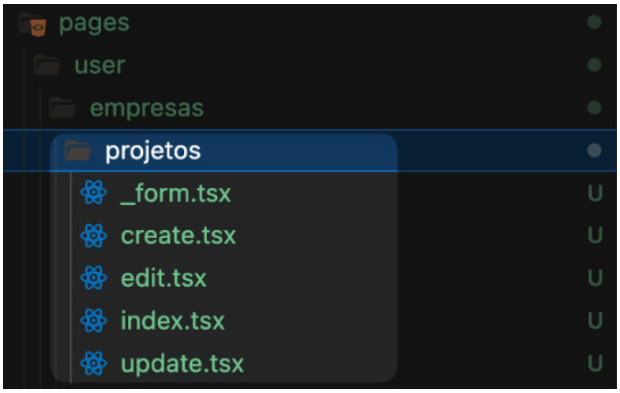


Figura 5. Rotas de parceiros

Fonte: Autoral, 2022

A rota empresa possui uma sub-rota chamada "projetos" que segue o mesmo padrão da rota "empresas" e da rota "parceiros", a figura 5 mostra esse padrão.

3.2 Tailwind e Flowbite para criação visual

O Tailwind junto com o Flowbite foram utilizados para criar o Css das páginas, o uso do Tailwind foi devido a sua praticidade em estilizar as páginas, que se dar através do uso de classes, o Tailwind ajuda também a manter um *design system*, que seria um padrão visual da página ou em outras palavras um *design* padronizado, para auxiliar e agilizar o desenvolvimento foi utilizado também o flowbite que são conjuntos de estilos pré-definidos que são chamados de componentes.

As páginas foram estruturadas, organizadas e desenvolvidas utilizando-se do *"Atomic Design"*, assim como foi mostrado as organizações de pastas e arquivos, na página do site (que é uma *single page aplication*) ela possuiu componentes, estes são os exemplos de componentes: o header, o botão, o campo de chamada para ação, outro para mostrar os parceiros e outros componentes. A página está da seguinte forma com os componentes unidos formando o layout, ficaram da seguinte forma:

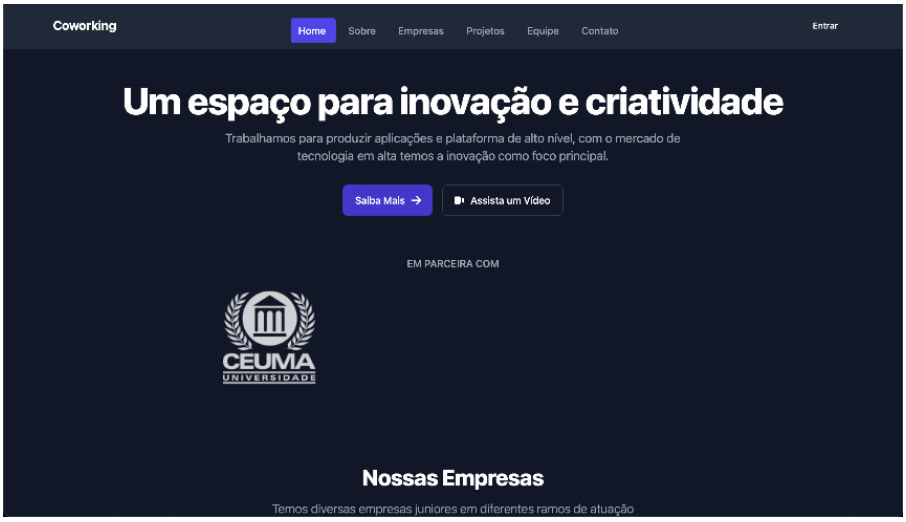


Figura 6. Home - Coworking

Fonte: Autoral, 2022

A página inicial possuí uma parte para mostrar o lema do coworking, a sua visão e missão, há também um botão para ver mais detalhes sobre o *coworking*, como por exemplo onde o ele está localizado, os seus integrantes, as fotos do *coworking* e outros detalhes.

Figura 7. Home - Parceiros

Fonte: Autoral, 2022

Além da página inicial mostrar sobre o *coworking* um pouco mais abaixa é mostrado os parceiros das empresas juniores. Os dados mostrados nessa página são dinâmicos e estão sendo obtidos através da *API REST*, utilizando um módulo do react chamado Axios, o Axios possibilita fazer requisições *HTTP* já tendo tratativas (*exeptions*) implementadas

por ele, ele também possibilita uma requisição mais segura pois já contem proteções contra ataques hackers. A requisição utilizada para obter os parceiros é o método *GET*.

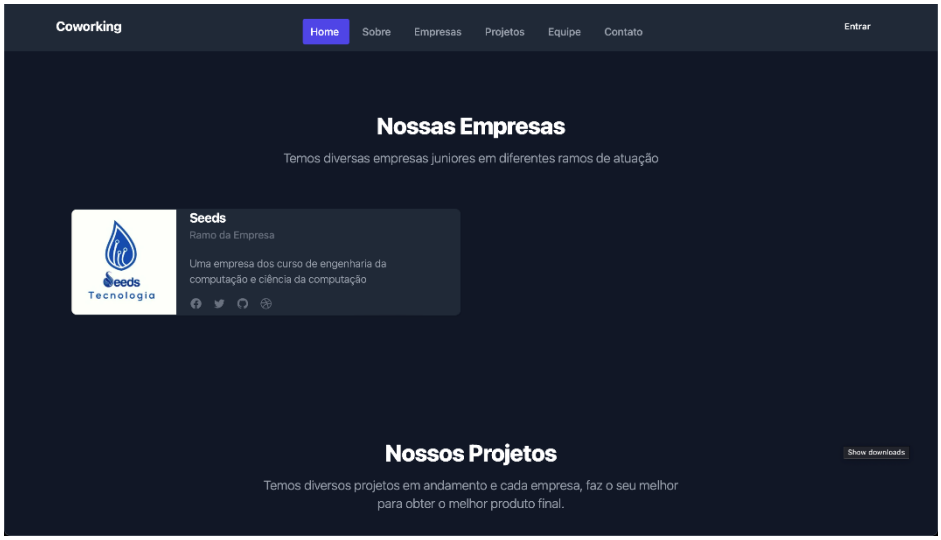


Figura 8. Home - Empresas

Fonte: Autoral, 2022

Um pouco mais abaixa dos parceiros, há também mostrando todas as empresas pertencentes ao *coworking*, o usuário tem a possibilidade de ver as redes sociais de cada empresa junior ou no próprio site ver mais sobre a empresa específica. A figura 8 mostra a listagem das empresas juniores, que está sendo obtida da *API REST*.

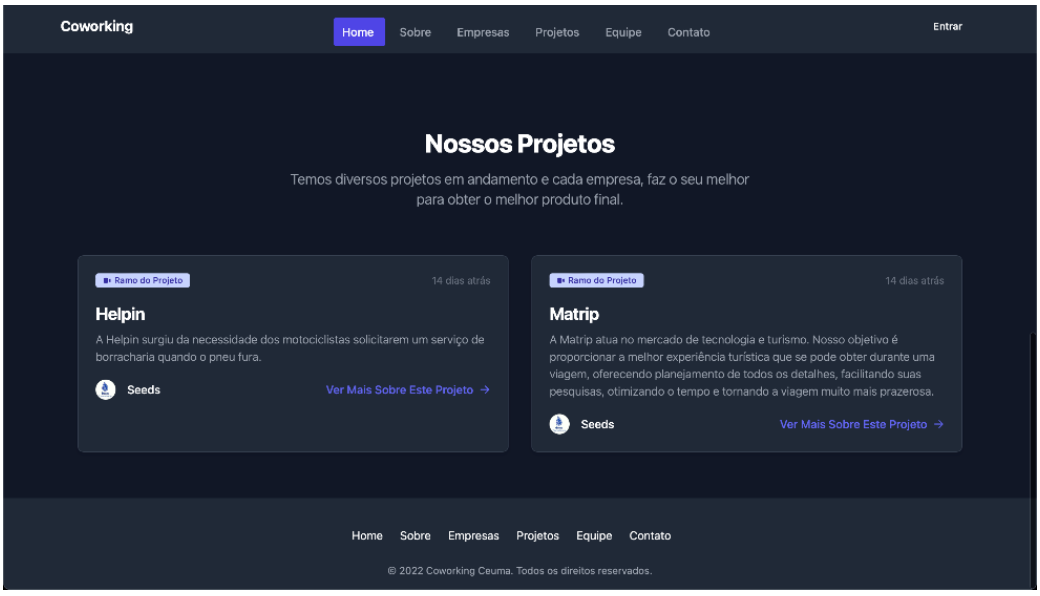


Figura 9. Home - Nossos Projetos

Fonte: Autoral, 2022

Abaixo das empresas na página inicial, há uma parte que mostra os projetos que o *coworking* desenvolve e a empresa que desenvolve ou desenvolveu o projeto, nessa parte o usuário tem a capacidade de ver sobre a empresa específica de cada projeto ou ver sobre o projeto especifico. A figura 9 mostra a listagem dos projetos.

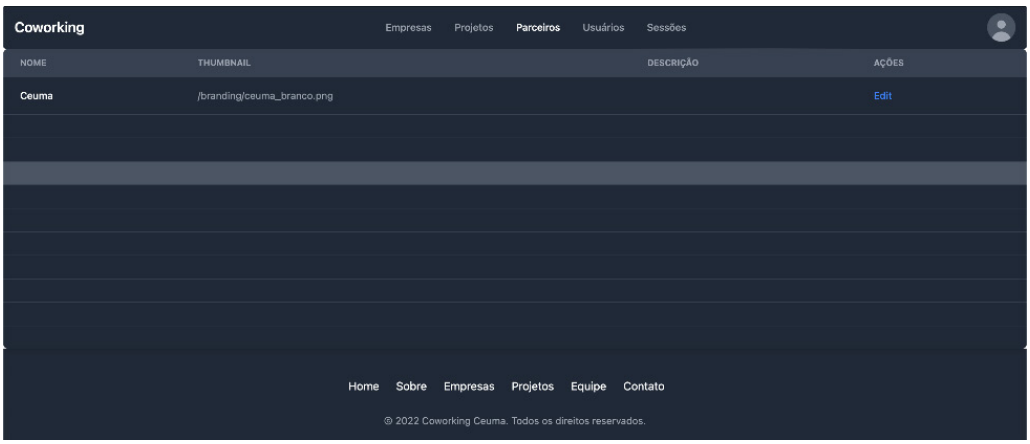


Figura 10. Studio - Parceiros

Fonte: Autoral, 2022

Com o usuário logado na página de parceiros, se o usuário possuir uma das seguintes permissões: administrador ou coworking, ele poderá adicionar um novo parceiro, editar um parceiro, excluir um parceiro ou ver todos os parceiros cadastrados. A figura 10 mostra a listagem de todos os parceiros.

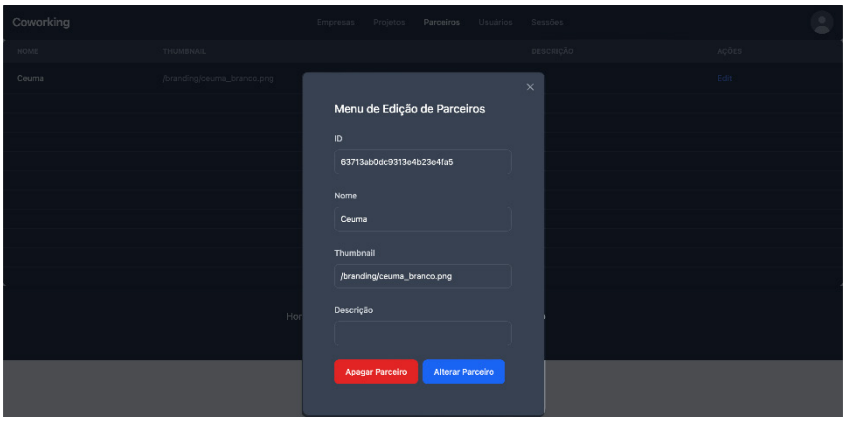


Figura 11. Studio – Menu de Edição

Fonte: Autoral, 2022

Na mesma página, o usuário pode abrir o menu de edição, no menu de edição ele pode realizar todas as operações citadas anteriormente. A figura 11 mostra o menu de edição.

Esse mesmo padrão de página se mantém, nas outras páginas como a página de empresas, projetos e usuários. Porém, somente o usuário administrador possuí acesso a página de usuários, nessa página o administrador pode editar as permissões do usuário, assim como desativar ou ativar uma conta.

4. RESULTADOS E DISCUSSÕES

Ao finalizar o desenvolvimento do projeto, foi possível notar que objetivo de divulgação das empresas e os projetos do coworking foram satisfatórios. Foi possível utilizar o site para testes e os testes também foram satisfatórios.

Também foi realizada uma pesquisa, com o objetivo de obter informações sobre a usabilidade e a interação do usuário com a interface com o objeto de identificar falhas ou melhorias que poderiam ser realizadas. A pesquisa obteve 150 entrevistados com o

público-alvo, os alunos e pessoas relacionadas ao Ceuma. Foram feitas 4 perguntas dissertativas, sendo as três primeiras acompanhadas pelas imagens do site e a última sobre quais melhorias poderiam ser realizadas, veja os resultados:

Questionando se os entrevistados têm algumas dificuldades em utilizar sites:

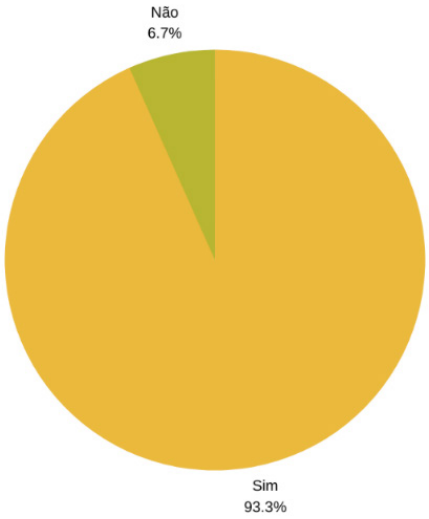


Figura 12. Gráfico de pizza sobre as dificuldades de utilizar sites.

Fonte: Autoral, 2022.

Os resultados obtidos mostraram que a grande maioria (93.3%) consegue utilizar sites sem problemas e que uma pequena minoria (6,7%) já possuem dificuldades para utilizar sites. Os resultados mostram que a maioria dos entrevistados não possuem dificuldades para utilizar sites em geral, dessa forma essa maioria já possuí habilidades bem definidas para utilizar sites.

A próxima pergunta, questiona aos usuários se eles seriam capazes de navegar pelo site que foi desenvolvido sem dificuldades:

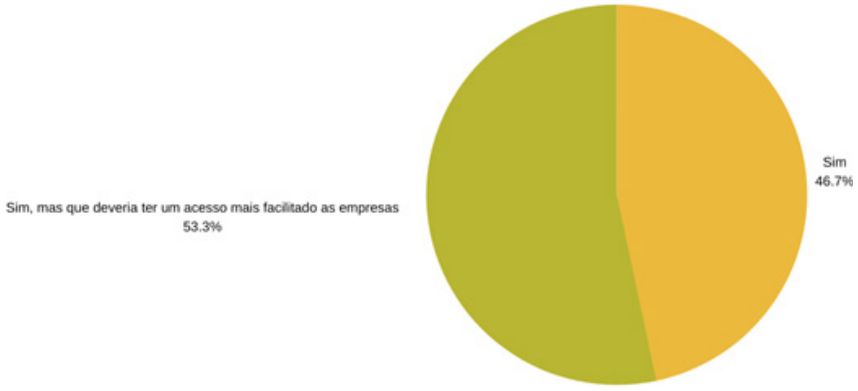


Figura 13. Gráfico de pizza sobre a capacidade de navegar pelo site

Fonte: Autoral, 2022.

Com essa pergunta foi possível notar que a maioria dos usuários (53,3%), afirmaram que deveria haver um melhor acessa as informações das empresas e os outros usuários (46,7%) afirmaram que conseguiriam utilizar o site tranquilamente. Foi possível notar através das respostas, que houve um equilíbrio entre as pessoas que conseguiriam acessar as informações presentes no site e as pessoas que acharam que deveria haver algum outro método de acessar essas informações.

Perguntando aos entrevistados se conseguiriam navegar entre os projetos e as em-

presas, foi possível ter a seguinte conclusão:

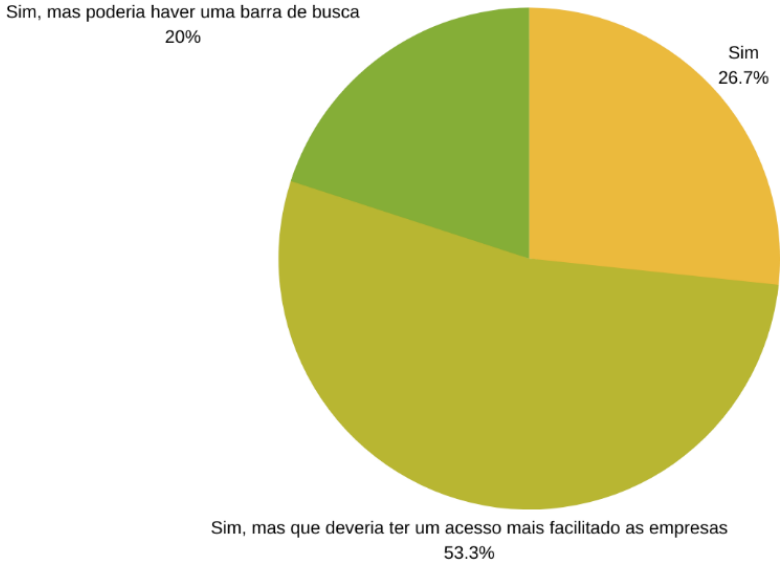


Figura 14. Gráfico de rosca sobre as telas de convite

Fonte: Autoral, 2022.

Sobre a navegação entre empresas e projetos, 26,7% dos entrevistados disseram que conseguiram navegar tranquilamente entre empresas e projetos, 20% dos entrevistados acham que a navegação seria mais fácil se houvesse uma barra de pesquisa e 53,3% acreditam que as empresas estão um pouco escondidas no site, tendo um acessa mais dificultado as informações delas. No entanto, a maioria dos entrevistados demostraram que deveria haver um acesso mais facilitado as informações do site, como por exemplo a inclusão de uma barra de pesquisa.

Perguntando aos entrevistados o que poderia haver de melhorias no site, os resultados foram o seguinte:

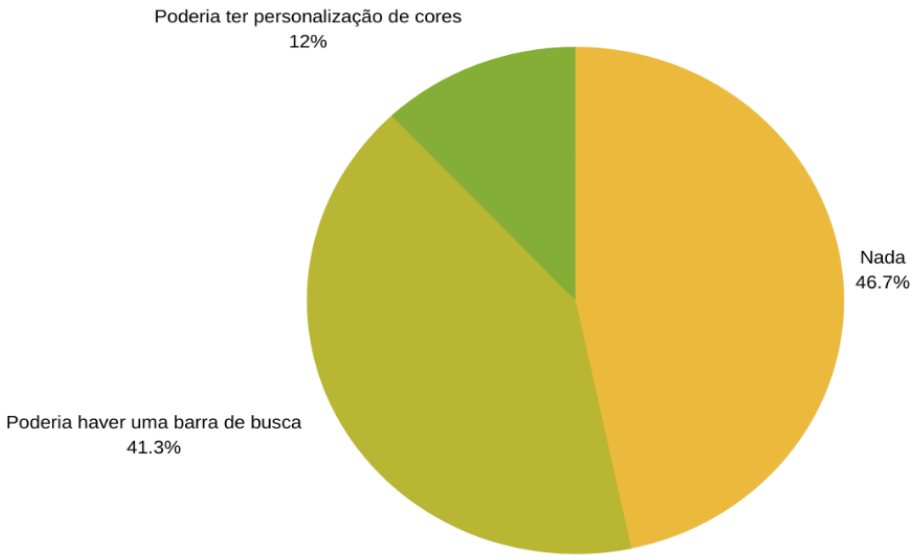


Figura 15. Gráfico de pizza sobre melhorias no site

Fonte: Autoral, 2022.

No último questionamento, 46,7% entrevistados não sugeriram nada. 41,3% dos

entrevistados sugeriram a inclusão de uma barra de pesquisa que levasse direto para empresas ou projetos de interesse e 12% disseram que poderia haver uma opção de personalização de cor, como um botão que mudasse do modo escuro para o modo claro. A maioria dos entrevistados sugeriram que deveria haver uma barra de pesquisa para melhor acesso as informações e que deveria haver uma personalização de cores no site, como ir ter um modo *dark* e um *light*.

5. CONCLUSÃO

As pesquisas mostram que o site obteve um excelente desempenho de função, mesmo que alguns entrevistados tenham solicitado algumas mudanças, como a inclusão de uma barra de pesquisa. Apesar disso, o objetivo desse projeto foi alcançado.

Com isso, o site cumpre o papel de ser um site auxiliar ao *coworking* trabalhando na divulgação das empresas e seus projetos, o gerenciamento ocorre pelos integrantes do *coworking* fazendo com que o site esteja sempre com dados recentes e atualizado.

As pesquisas ajudaram a verificar melhorias que poderiam ser integradas ao site, como uma barra de pesquisa para melhorar na navegação, a opção de mudar do modo escuro para o claro e vice-versa, e a inclusão de mais botões de acesso rápido as informações.

AGRADECIMENTOS

Agradeço primeiramente a Deus pelo meu dom de aprender com facilidade, agradeço meus pais por terem me proporcionado uma ótima instrução acadêmica e agradeços aos meus amigos Gabriel e Afonso pela ajuda quando tive dificuldades e pela possibilidade de aprender também mesmo quando eu os ensinava algo.

Referências

AXIOS, Axios. React. **Introdução ao Axios,** 2020. Disponível em: https://axios-http.com/ptbr/docs/intro. Acesso em: 08 nov. 2022.

Conceitos. **Leonardo.dev,** 2022. Disponível em: https://docs.leunardo.dev/reactjs. Acesso em: 12 nov. 2022.

Create a Next.js App**. Next.js,** 2022. Disponível em: https://nextjs.org/learn/basics/create-nextjs-app. Acesso em: 12 nov. 2022.

EIS, Diego. **Guia Front-End**: O caminho das pedras para ser um dev front-end. Digital: Casa do Código, 2014.

FERREIRA, Rodrigo. REST: Princípios e boas práticas. **Alura,** 2017. Disponível em: https://www.alura.com.br/artigos/rest-principios-e-boas-praticas?gclid=CjwKCAiA68ebBhB-EiwALVC-NpXoNhwbom4GnfdBdXV9r-vVr0irf7UXIrW0fNrlDM55HJCt_dEEMHBoC1DAQAvD_BwE. Acesso em: 13 nov. 2022.

FLANAGAN, David. **Javascript**: O guia definitivo. 6. ed. Porto Alegre: Bookman, 2013.

FLOWBITE, Tailwind. Flowbite - Tailwind CSS component library. **Flowbite,** 2022. Disponível em: https://flowbite.com/docs/getting-started/introduction/. Acesso em: 12 nov. 2022.

GERCHEV, Ivaylo. **Tailwind CSS**: Craft Beautiful, Flexible, and Responsive Designs. Austrália: SitePoint, 1978.

HAT, Red. API REST. **Red Hat,** 2020. Disponível em: https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api. Acesso em: 10 nov. 2022.

Introdução. **React,** 2022. Disponível em: https://pt-br.reactjs.org/docs/getting-started.html. Acesso em: 12 nov. 2022.

KONSHIN, Kirill. **Next.js Quick Start Guide**: Server-side rendering done right. Birmingham: Packt Publishing Ltd, 2018.

LIMA, Matheus. O Guia Completo do React e o seu Ecossistema. **Medium,** 2022. Disponível em: https://medium.com/tableless/o-guia-completo-do-react-e-o-seu-ecossistema-b31a10ecd84f. Acesso em: 12 nov. 2022.

PRESCOTT, Preston. **Programação em JavaScript**. Digital: Babelcube Inc., 2016.

RIVA, Michele. **Real-World Next.js**: *Build scalable, high-performance, and modern web applications using Next.js, the React framework for production.* 1. ed. BIRMINGHAM: Packt Publishing Ltd, 2022.

RIBEIRO PEREIRA, Caio. **Construindo APIs REST com Node.js**: Caio Ribeiro Pereira. Digital: Editora Casa do Código, 2016.

SAUDATE, Alexandre. **APIs REST**: Seus serviços prontos para o mundo real. Digital: Casa do Código, 2021.

SEHN, Leandro Roberto. **Web Design**: Conceitos Introdutórios. Porto Alegre: Simplissimo Livros Ltda, 2018.

STEFANOV, Stoyan. **Primeiros passos com React**: Construindo aplicações web. 1. ed. São Paulo: Novatec, 2019.

6

A OTIMIZAÇÃO DE UM BACK END COM WEB SERVICE REST API DESENVOLVIDO COM SPRING BOOT

OPTIMIZING A BACK END WITH WEB SERVICE REST API POWERED BY SPRING BOOT

Lucas Pereira Saraiva Carneiro[1]

Edilson Carlos Silva Lima[2]

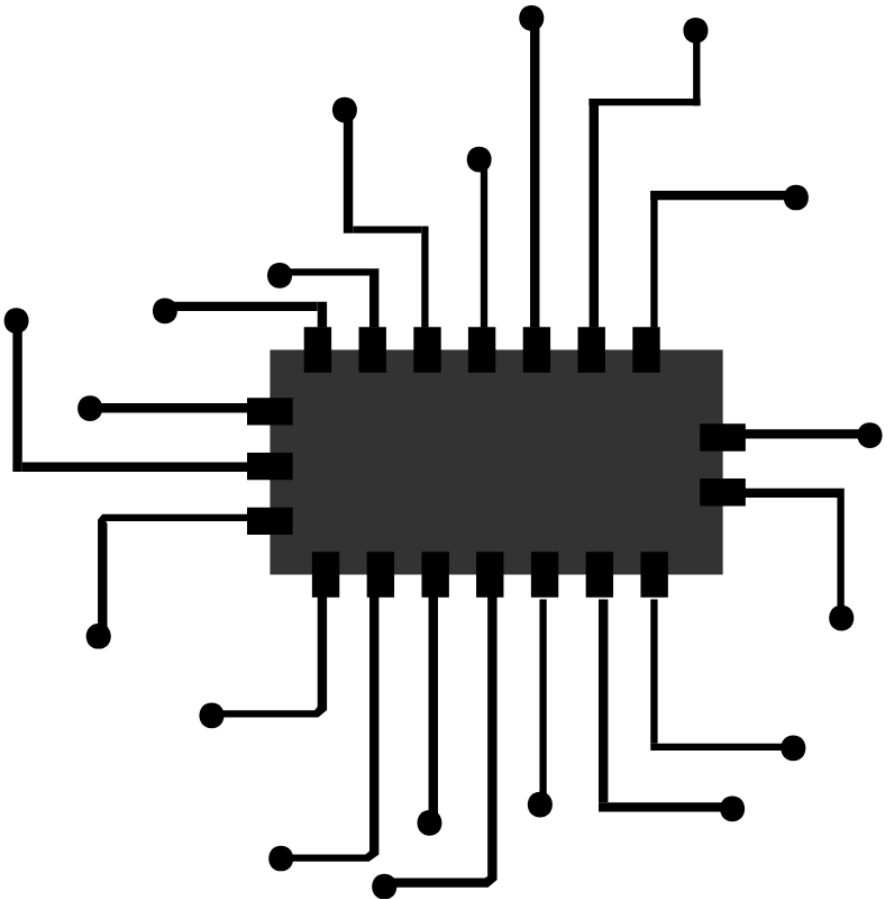
Elda Regina de Sena Caridade[3]

[1]Engenharia 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

{CARNEIRO, Lucas Pereira Saraiva, lucaspscarneiro@gmail.com; LIMA, Edilson Carlos Silva. Edilsonlima3@gmail.com; CARIDADE, Elda Regina de Sena. elda18.sena@gmail.com}



d.o.i.: 10.29327/5192626.1-6

Resumo

Hoje em dia, a pessoas estão se preocupando mais com sua saúde e das pessoas que amam e sempre buscam por informações médicas no seu dia-dia. Pensando em trazer uma maior acessibilidade aos serviços médicos foi criado a *MEDIWAY*, um *WEB* Service que tem o objetivo de fornecer informações confiáveis sobre saúde. Nesse artigo apresenta-se um *Back End* da plataforma *MEDIWAY* sendo detalhado também a construção de *BACK-END* com API-REST utilizando o ecossistema Spring, e permitindo que o sistema se dívida de acordo com os seus recursos e seja aplicado em outros sistemas. Primeira parte do artigo temos uma fundamentação teórica resumindo as funções e métodos de cada parte do sistema, logo no capítulo seguinte temos o estudo de como foi desenvolvido e como será consumido essa API. Todas as informações adquiridas para elaboração deste trabalho, se deram através de buscas bibliográfica em trabalhos acadêmicos já publicados. Portanto aqui será descrito desde as características da arquitetura de desenvolvimento, a arquitetura REST. Observando várias tendências (por exemplo, suporte JSON generalizado, documentação gerada por software), constatou-se resultados que podem ajudar os profissionais a desenvolver diretrizes e padrões para projetar serviços de maior qualidade e também entender as deficiências nos serviços atualmente implantados na área.

Palavras-chave: API, BACKEND, Arquitetura REST, REST API

Abstract

Nowadays, people are more concerned about their health and that of the people they love and are always looking for medical information in their daily lives. Thinking about bringing greater accessibility to medical services, MEDIWAY was created, a WEB Service that aims to provide reliable information about health. This article presents a Back End of the MEDIWAY platform, also detailing the construction of a BACK-END with API-REST using the Spring ecosystem, and allowing the system to debt according to its resources and be applied in other systems. The first part of the article has a theoretical foundation summarizing the functions and methods of each part of the system, then in the next chapter we have the study of how this API was developed and how it will be consumed. All information acquired for the elaboration of this work was given through bibliographical searches in academic works already published. Therefore, here it will be described from the characteristics of the development architecture, the REST architecture. Looking at various trends (e.g., widespread JSON support, software-generated documentation), we found results that can help practitioners develop guidelines and standards for designing higher quality services and also understand deficiencies in services currently deployed in the field.

Keywords: API, BACKEND, REST Architecture, REST API

1. INTRODUÇÃO

A preocupação com a saúde e bem-estar deve estar em primeiro lugar sempre, e atualmente, o surgimento de algumas doenças tem causado um enorme impacto na população mundial. Além da velocidade em que se proliferam, elas se deparam também com a falta de acessibilidade a informações e tratamento que atinge uma parte grande da sociedade de diversos países. E pensando em levar um suporte a essa necessidade da população que estamos oferecendo o serviço MEDIWAY, um site que fornece informações para a manutenção da sua saúde, sendo um importante recurso pra quem necessita de tratamentos ou medicamentos, oferecendo suporte de profissionais médicos para auxiliar cada paciente em diversas especialidades.

1.1 Metodologia

O sistema foi desenvolvido com a implantação de softwares com códigos pré-programados, tornando se uma base para esse projeto WEB. Para desenvolvimento desse *Back End da MEDIWAY* foi feita o consumo de uma API REST, utilizando o *SPRING BOOT* e seus módulos em JAVA. Esse tipo de aplicação permite o *Back End* se comunicar com vários *Front end's*, reduzindo tempo e custo de desenvolvimento, essa separação de responsabilidade sobre o sistema é devido a sua comunicação *STATELESS*, o que facilita que cada componente desse sistema possa de forma independente evoluir e incluir novas funcionalidades sem que a outra camada do sistema sofra com suas implementações. Através do *TOKEN* de autorização será possível gerenciar os códigos por meio da plataforma do *POSTMAN* permitindo a troca de informações entre o *cliente-server*.

O Resultado do consumo dessa API foi de êxito sendo necessário apenas a conexão com o *FRONT END*, possibilitando o *INTERFACE* com o sistema.

Neste artigo vamos abordar alguns tópicos importantes, sendo eles: a fundamentação teórica e metodológica no capítulo 2 , no capítulo 3 discorreremos o aplicativo desenvolvido para auxiliar na problemática escrita na introdução, no capítulo 4 sobre a conclusão e, no capítulo 6 ressalto meus agradecimentos e por fim no capítulo temos as referências utilizadas para a construção deste artigo.

1.2 Justificativa

Desta forma a motivação para o desenvolvimento deste projeto é a busca por soluções capazes de integrar e fornecer os serviços de informações referente ao MEDIWAY.

Com surgimento de vírus e doenças altamente contagiosas ficou clara a importância de estar bem informado, porem nem sempre as informações divulgadas em outros serviços são confiáveis. O principal objetivo da MEDIWAY é fornecer de forma confiável, informações a respeito a saúde de seus clientes.

Pelo fato de existirem diversos tipos de tecnologias e frameworks que possibilitam a criação de *backends*. Acabei investigando quais os Frameworks mais usados e eficazes devido a cada tecnologia possuir suas próprias particularidades e escolhi o SPRING INITIALIZER, pois já tenho uma breve noção da implementação de aplicações JAVA.

2. FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo, será abordado uma revisão literária dos assuntos essenciais para o Desenvolvimento de uma Web Service API REST sendo dividido em partes, são elas: API, POSTMAN, Arquitetura REST, MYSQL, Desenvolvimento com SPRING BOOT.

2.1 API e Arquitetura REST

A API (Application Progamming Interface, em português, Interface de Programação de Aplicação), define um meio de acesso a determinada informação por uma aplicação. Para Gouveia (2016):

> "API é uma ferramenta que realiza comunicação entre aplicações que desejam compartilhar suas rotinas, ferramentas, padrões e protocolos".

Uma API gera uma interface entre duas aplicações sem que elas compartilhem algum código entre si. É uma forma de comunicação entre sistemas, permitindo interação e compartilhamento de serviços e informações entre si.

> "API é como um garçom. O cliente nesse caso é a aplicação que deseja receber os serviços, recebe do garçom o menu com todos os itens daquele restaurante. Após a escolha, o garçom leva o pedido até a cozinha, aplicação da API, onde os cozinheiros que são os serviços compartilhados pela aplicação, realizam o pedido como foi descrito pelo cliente. Ao concluir o pedido o cozinheiro avisa o garçom, este entrega o pedido ao cliente completando o processo de exemplificação de uma aplicação de API" (GOUVEIA, 2016).

Um exemplo: Uma API para solicitação de previsão do tempo, tem um endereço para acessar que esperaria um ou mais parâmetros para identificar o local de consulta. Quando receber o pedido, irá processar somente as informações internas necessárias, retornando apenas o resultado final da solicitação. A aplicação requisitante trata o retorno da melhor forma.

Ou seja, trata-se de uma pergunta e uma resposta, sem necessidade de interação no processamento intermediário. Outro exemplo é o sistema de geolocalização (*Google Maps*) onde muitos sistemas clientes usam seus serviços, como em aplicativos de trânsito, transporte, delivery..., onde nenhum sabe de detalhes de sua implementação.

Existem diversas maneiras de interação do cliente com a API, onde a API não é diretamente associada a uma linguagem de programação, por exemplo, podendo assim ser escrita na linguagem JAVA e consumida na linguagem PHP.

Na figura 1, temos a demonstração da estrutura de interação de uma *WEB service*.

Figura 1: Web Service.

Fonte: AUGUSTO, 2019.

Nesse estudo as interações utilizaram a internet por meio de protocolo-HTTP e suas requisições ou também conhecidas como *WEB Services*. Segundo AUGUSTO, 2019:

> [...] é uma tecnologia que permite a comunicação entre aplicações independente a linguagem de programação e o sistema operacional [...]. Os WEB Services são componentes que permite que a aplicação envie e receba dados geralmente no formato XML ou JSON (AUGUSTO, 2019).

Representational State Transfer (REST) é uma arquitetura que se baseia nas regras do protocolo-HTTP, que a define como um conjunto de CONSTRAINTS que tem o objetivo de definir uma forma de fracionar um sistema. A arquitetura REST foi introduzida no início do século XX (2000) por Roy Thomas Fielding. De acordo com os princípios REST, as interfaces REST dependem exclusivamente de *Uniform Resource Identifiers* (URI), para detecção e interação de recursos, e geralmente no *Hypertext Transfer Protocol,*(HTTP) para transferência de mensagens. Um URI de serviço REST fornece apenas a localização e o nome do recurso, que serve como um identificador de recurso exclusivo. Os verbos HTTP predefinidos são usados para definir o tipo de operação que deve ser executada no recurso selecionado

> [...] um conjunto de princípios que definem como os padrões da Web, como HTTP e URI's, devem ser usados. A promessa é que, se você aderir aos princípios do REST ao projetar seu aplicativo, terá um sistema que explora a arquitetura WEB em seu benefício (TILKOV, 2007).

Essa Arquitetura nos proporciona a vantagem das suas partes possam progredir sem comprometa o sistema.

As URI's servem para identificarmos os recursos. Com a determinação desses recursos conseguimos executar ações na aplicação.

Nos *WEB Services* sua execução é feita pelo HTTP através dos verbos GET: recuperar dados; *POST*: enviar; *PUT*: atualizar; *DELETE*: remover

2.2 POSTMAN

É uma plataforma que só pode ser utilizado em API, ela se conecta ao MYSQL fazendo com que o banco de dados preencher todos os formulários de maneira rápida e simples, seja para construir ou testar a API, esse programa foi criado pelo Engenheiro de Software *Abhinav Astana* (2012). Sua ideia era criar essa ferramenta que auxilia outros programas implementarem seus projetos com mais facilidade. Os produtos que armazenam documentos, orçamentos, entre outros, após alguns comandos já pode usar os métodos *POST e GET* para testar seus sistemas.

O POSTMAN possui suas ferramentas que ajuda as API's. São elas de acordo com MOZILLA (2000):

- *GET*: É uma solicitação de recurso específico que pode retornar somente dados.
- *POST*: É utilizado para mudança de estado de recurso no servidor (banco de dados).
- *DELETE*: É utilizado para remover dados.
- *TOKEN*: Autenticação que solicita autoriza o para de criptografar.
- *URI*: Sistema da listagem do cliente.

Existe também os códigos do status de retorno do protocolo de cada ação do servidor, de acordo com Mozilla (2000) são eles:

- 200 *OK*: O código status está estabilizado e funcionando.
- 204 *NO content*: Não tem conteúdo disponível para enviar resultado usando *POST*.
- 404 *NOT found*: Servidor não foi encontrado.
- 500 *INTERNAL Server Error*: Servidor não foi encontrado.

2.3 *Client Server*

É a comunicação entre componentes SERVIDOR, onde estão as informações, e o CLIENTE, que é quem requisita essas informações.

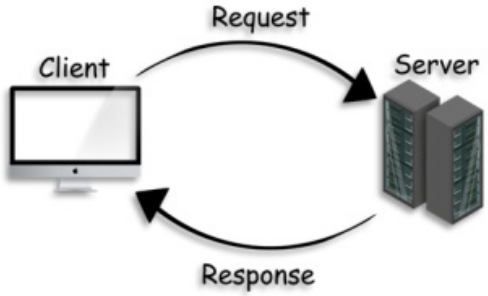


Figura 2: Comunicação *Client-Server*.

Fonte: Leite, 2016.

Na figura 2, temos um sistema distribuído em funcionamento com dois componentes, quem solicita algo *client-side* (lado cliente) e quem escuta e responde, *sever-side* (lado server).

Um exemplo, o YOUTUBE onde você como cliente SOLICITA o vídeo e o servidor LER

a requisição e te mostra uma cópia do vídeo.

Segundo Thomas Fielding (2000), essa arquitetura tem o objetivo separar as funções do *BACK-END*, encarregado de todo lado servidor, implantando regras de negócio, e o CRUD no banco de dados. E do *FRONT-END,* que interage com o cliente através de interface gráfica. Ele acredita que essa arquitetura possa progredir de forma independente, permitindo incluir novas funções ao sistema sem que um prejudique o outro.

2.4 *Stateless*

Segundo Thomas Fielding (2000), a comunicação entre servidor e cliente deve estar sem estado, todas requisições direcionadas ao servidor devem ter informações necessárias para a compreensão da solicitação por ele. Fazendo que o sistema até perca empenho na REDE, mas acaba ganhando mais confiabilidade, visibilidade e escalabilidade.

- Confiabilidade: pois, facilita todo processo de recuperação de falhas parciais.
- Escalabilidade porque o servidor não armazena o estado entre as requisições, permitindo que libere o recurso rapidamente.
- Visibilidade pois o sistema apenas precisa de a requisição para identificar toda sua natureza.

A Figura 3. demonstra a simulação do pedido em uma loja virtual, o cliente envia uma lista com tudo que deseja, seus dados e pagamento. Assim o servidor possui em uma requisição todas as informações necessárias para finalizar o pedido.

Figura 3: Exemplo da aplicação *Stateless*

Fonte: Leite, 2016.

Devida a Arquitetura REST não permite armazenar informação no servidor, FIELDING solucionou isso propondo que cada requisição seja tratada como a primeira q o servidor "recebeu" daquele usuário. É utilizado TOKENS e COOKIES que armazenam o estado CLIENTE, parecendo que já o conhece.

2.5 Cache

De acordo com os estudos de Fielding (2000), essa *constraints* tem a função de melhorar o desempenho da rede, assim sendo rotulada toda resposta do servidor sendo ela armazenável ou não em CACHE.

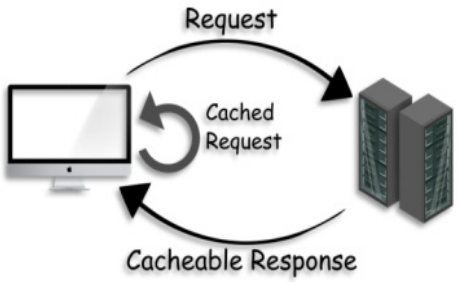


Figura 4: Função Cache.

Fonte: Leite, 2016

Se uma resposta pode ser armazenada em CACHE, ela fica salva em CACHE, quando o cliente fizer a requisição, a informação ser novamente utilizada, a menos que ela tenha sido alterada no servidor. Conforme na figura 4.

2.6 *Uniform Interface*

Na Arquitetura REST sua ênfase na INTERFACE UNIFORME entre as aplicações é para Fielding (2000) a sua grande diferença dos outros modelos, sabendo que o servidor não diferencia o acesso aos dados pelo sistema operacional ou pelo dispositivo do cliente, tornando o acesso aos *ENDPOINTS* do serviço WEB igual para qualquer dispositivo que tenha acesso, como Smartphones, Laptops, Notebooks e outros.

Conforme na figura 5. O servidor se comunica de maneira igualitária entre os dispositivos;

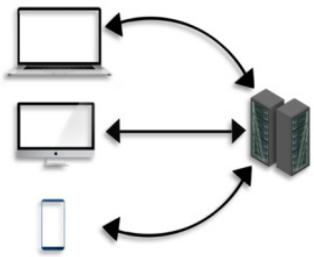


Figura 5: Interface Uniforme.

Fonte: Leite, 2016.

Sendo feitos através do JSON e XML manipulando a informação seja qual for o dispositivo, conforme mostrado no tópico sobre API.

2.7 *Layred System*

Traduzido como sistemas de camadas e segundo Fielding (2000) o estilo no qual a Arquitetura REST é construída, seguindo uma sequência hierárquica onde um componente só consegue visualizar a camada imediata a qual ele está interagindo. Como ilustrado na Figura 6, os sistemas são dispostos em camadas.

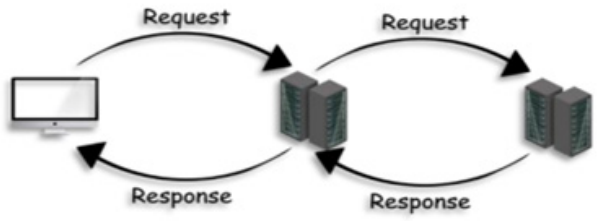


Figura 6: Estrutura do SISTEMA EM CAMADAS

Fonte: Leite, 2016.

Um exemplo, aplicativos de entrega(*delivery*) possuem seus clientes que acessam seus sites e aplicativos, *ENDPOINTS* que se integram a outros serviços como o *GOOGLE-MAPS* para realizar a entrega. Ou seja, o usuário não precisa acessar o *GOOGLEMAPS* diretamente, pois o sistema acessa a *ENDPOINT* no aplicativo de entrega que acessa a *ENDPOINT* do aplicativo de geolocalização.

3. ESTUDO DE CASO

Com a função de intermediar o contato do paciente com médicos e medicamentos e também levar acessibilidade de diversas pessoas a informações preciosas para os tratar e combater as doenças que foi criado o site MEDIWAY, uma plataforma onde você pode se manter informado sobre doenças, tratamentos e medicamentos com segurança e qualidade. Se você tiver um cadastro na plataforma, você terá a função de publicar informações que potencialize nossa aplicação no segmento da saúde.

O sistema foi desenvolvido com a implantação de softwares com códigos pré-programados, para a construção desse sistema foi usado mecanismos do SPRING INITIALIZER gerando através dos princípios REST API, todas as dependências para a base do projeto tornando compatível com a maioria dos sistemas.

Foi também executado um código de autenticação para gerar segurança e controle ao acesso ao sistema possibilitando o consumo dele junto ao POSTMAN, utilizado para gerenciar e modificar a aplicação de forma prática e segura.

O Desenvolvimento *BACK END* da aplicação esta finalizado, necessitando apenas consumir em uma aplicação *FRONT END* através do JAVA Script.

3.1 Desenvolvimento do *Back End*

Esse sistema foi construído sobre os princípios de REST API onde é fornecido mecanismos atuais para o suporte em vários tipos de sistemas operacionais. O que o torna uma vantagem pois com a vários equipamentos, é inviável criar um sistema para cada aparelho ou SO devido aos custos, REST API é cada vez mais requisitadas nas organizações, como forma de separar os sistemas, permitindo múltiplas integrações de dispositivos a um único sistema.

Foi utilizado o *SPRING BOOT* para gerar uma API, nela foi implementado uma classe com a função de popular nosso banco de dados e cadastrado dados para realizar testes na aplicação através da ferramenta HTTP chamada “POSTMAN”, onde é possível realizar as operações ENDPOINTS.

3.1.1 Desenvolvendo a API com Spring Boot

No SPRING INITIALIZER é onde seleciona-se as dependências iniciais, gerando uma API base para download. O SPRING é fundamental para desenvolvedores, e é muito usado pelos adeptos do JAVA, destacando a produtividade elevada proporcionada por ele, diminuindo tempo e custos no desenvolvimento. Para criar o projeto, seleciona-se as dependências, descompactua e importe o projeto no IDE escolhida. Os recursos serão colocados na pasta >*src/main/resources*>, usados na configuração de arquivos com banco de dados, tumb, imagens e outros. Sendo fundamental nos projetos MAVEN, o arquivo <porn.xml> serve para a configurar o projeto, fazendo o *dowload* automático de todas as configurações listadas nas dependências do projeto ao desenvolvedor.

No arquivo >*aplication.properties*> deve ser colocado todos os comandos de aplicação no banco de dados.

3.1.2 Classes do Sistema

Devido a possibilidade de colaboração de outros autores ao serviço, devemos organizar o sistema em camadas numa pasta "src/main/JAVA", conforme mostra a figura 7, vemos as camadas do sistema *MEDIWAY.*

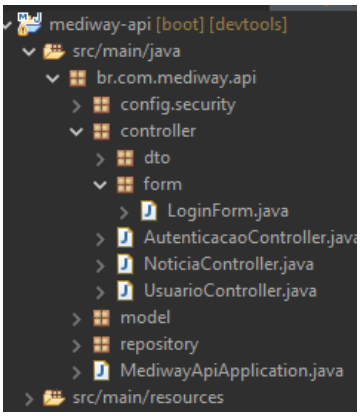


Figura 7: Classes do Sistema.

Fonte: Autoral, 2022.

E no pacote da aplicação criamos outro pacote com o nome "entidades" onde será colocado as classes do nosso sistema. Todos os códigos JAVA estão na pasta de origem >src/*main*/JAVA> que são as classes, a lógica do sistema e as camadas de organização. Os pacotes de códigos da aplicação serão colocados junto a pasta.

A classe do sistema Usuario Controller.java possui tributos como >ID< >Autenticaçao< >Noticia< >Usuario< contem anotações JPA (Java Persitence API), na função de saber como interagem as entidades dentro do banco de dados, uma técnica boa para persistência de dados. Em cada classe o sistema implantara seu modelo físico, segundo seus atributos e relações definidas.

3.1.4 Camada Repositório

Para memorizar as classes usaremos o SPRING Data, modulo SPRING que implantara o JPA164, assim, cada classe deverá ter uma interface de acesso para banco de dados. Devemos estender a classe "JPArepository" através da interface e mostrar dois parâmetros genéricos, que são eles a classe persistida e o tipo de sua chave primaria, conforme mostra a figura 8.

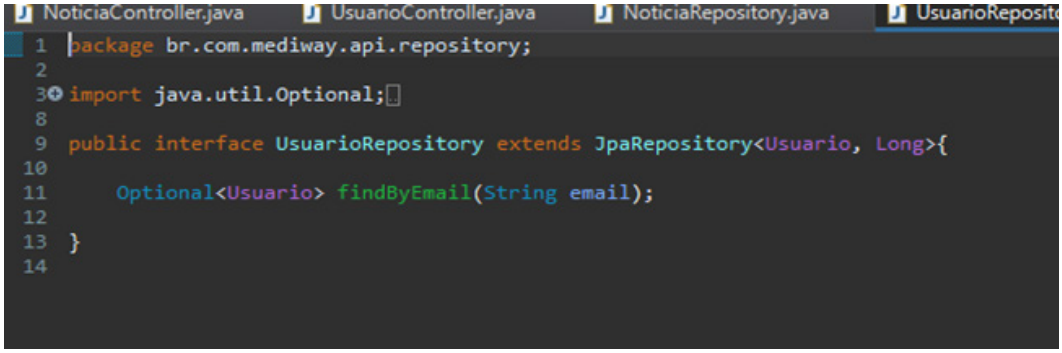


Figura 8: Criação Camada de repositório

Fonte: Autoral, 2022.

A partir daí crie um pacote chamado "repository" e também as suas interfaces, que permite usar recursos sem que os detalhes sejam conhecidos e implementados em classes abstratas.

3.1.5 Camada de Serviço

Todas REGRAS DE NEGOCIO do sistema estão colocadas em um pacote na camada "serviços". Já que os ENDPOINTS da API não terão acesso direto a camada repositório, é criado uma camada que faz a ponte entre a ENDPOINT e a camada repositor. Na camada de serviço deve ser criada em cada classe, sua classe de serviço, por exemplo, para a classe Postagem sua classe serviço se chamara "model", seguindo todas esse raciocínio.

Na figura 9, é ilustrado a classe Model:

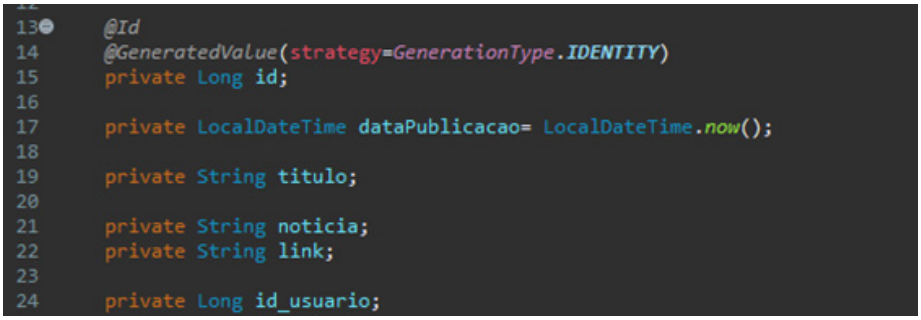


Figura 9: Classe de serviços da API

Fonte: Autoral, 2022.

As classes de serviços terão esses métodos: SALVAR, DELETAR, EDITAR, BUSCAR.

3.1.5.1 Método BUSCAR

No método BUSCAR é implementado um método para retornar a lista de notícias, fazendo uma busca geral utilizando camada repositório através da interface que definimos, bastando usar o método "*findALL*()". como está ilustrado na figura 10.

```
@GetMapping("/buscar")
public List<Noticia> buscar() {

    List<Noticia> todosNoticia = noticiaRepository.findAll();

    return todosNoticia;
}
```

Figura 10: Método buscar

Fonte: Autoral, 2022.

3.1.5.2 Método SALVAR e DELETAR

O método SALVAR, que serve para depositar a informação no sistema. No método SALVAR recebemos o parâmetro de uma classe já estanciada tipo Noticia, e utiliza suas estancias para salvar.

```
@PostMapping("/salvar")
public Noticia salvar(@RequestBody Noticia noticia, @AuthenticationPrincipal Usuario usuario) {
    noticia.setId_usuario(usuario.getId_usuario());
    Noticia noticiaSalvo = noticiaRepository.save(noticia);

    return noticiaSalvo;
}
```

Figura 11: Criação do método SALVAR Noticia.

Fonte: Autoral, 2022.

O método DELETAR serve para remover dados já estanciados no sistema, ele recebe o identificador da Postagem, verifica a existência dela através do método "BUSCAR" e o deleta.

Na figura 12, ilustrado o método DELETAR por ID;

```
48
49    @DeleteMapping("/deletar/{id}")
50    public void apagarPorId(@PathVariable Long id) {
51
52        noticiaRepository.deleteById(id);
53
54
55    }
56
```

Figura 12: Criação do método DELETAR id.

Fonte: Autoral, 2022.

onde será removido todas as informações depositadas no sistema pelo usuário selecionado.

3.1.5.3 Método EDITAR

No método EDITAR é necessário implementar um outro método auxiliar, devido os objetos estarem monitorados pela JPA, Para EDITAR a postagem é necessário BUSCAR

ela, inserir as modificações e por fim, atualizar ela.

```
19 @RestController
20 @RequestMapping("/usuarios")
21 public class UsuarioController {
22
23
24     @Autowired
25     private UsuarioRepository usuarioRepository;
26
27
28     @PostMapping("/salvar") //salva um novo usuario
29     public Usuario salvar(@RequestBody Usuario usuario) {
30
31         Usuario usuarioSalvo = usuarioRepository.save(usuario);
32
33         return usuarioSalvo;
34     }
```

Figura 13: Método EDITAR

Fonte: Autoral, 2022.

Conforme a figura 13, O SPRING Data utiliza o método "save" para cadastrar, e para editar. Ele consegue diferenciar a ação do método pela chave primaria da entidade, exemplo, se o ID for nulo, irá cadastrar, se não for nulo, irá tentar editar. Então o método recebera um objeto monitorado e irá chamar o método "save" do seu repositório.

3.2 Camada de Recurso

Na camada de recurso implementamos os ENDPOINTS, que funciona como entrada pra quem utiliza a API. Ao iniciar devemos criar um pacote "recurso". Conforme abaixo na Figura 14.

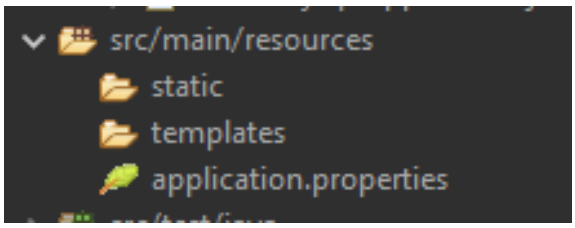


Figura 14: Pacote de recursos

Fonte: Autoral, 2022.

Nesse pacote deve ser criado para cada classe, suas classes de recurso, contendo os ENDPOINTS. por exemplo, a classe de recurso de postagem, nome atribuído foi "RecursoNoticias", nela será estabelecido uma conexão entre camada de serviço e os métodos criados.

3.3 Consumindo serviço da API com o POSTMAN

Foi implementado uma classe com a função de popular nosso banco de dados e cadastrado dados para realizar testes na aplicação através da ferramenta HTTP chamada "POSTMAN", onde é possível realizar as operações ENDPOINTS.

POSTMAN é muito útil para praticar a forma mais segura para modificar sua aplicação, com ele você pode gerenciar os dados de sua aplicação de forma prática pois possuem uma série de métodos que automatiza o processo de codificação. É necessário usar um *Token* que é uma autenticação de permissão do desenvolvedor para de criptografar

sua informação para o usuário, como podemos ver na figura 15. Esse *TOKEN* serve como permissão para que o usuário possa programar cada método.

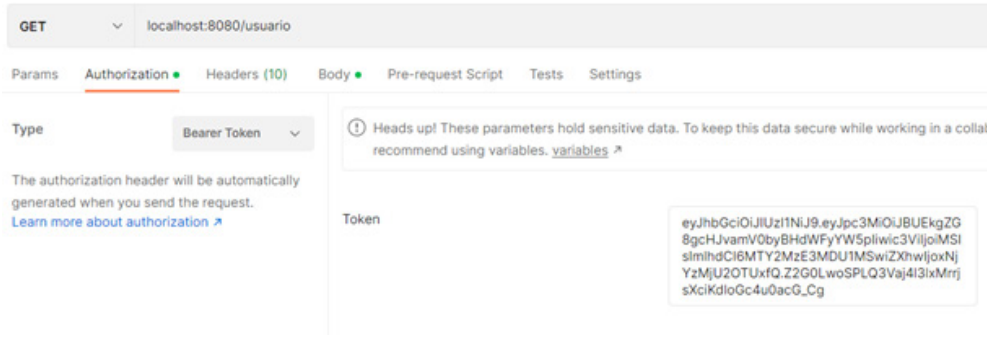


Figura 15: Token de autenticação no POSTMAN

Fonte: Autoral, 2022.

Para cadastrar os usuários implementamos os outros métodos como: GET, POST, DELETE.

1. GET: Solicita um recurso específico que pode retornar somente dados.

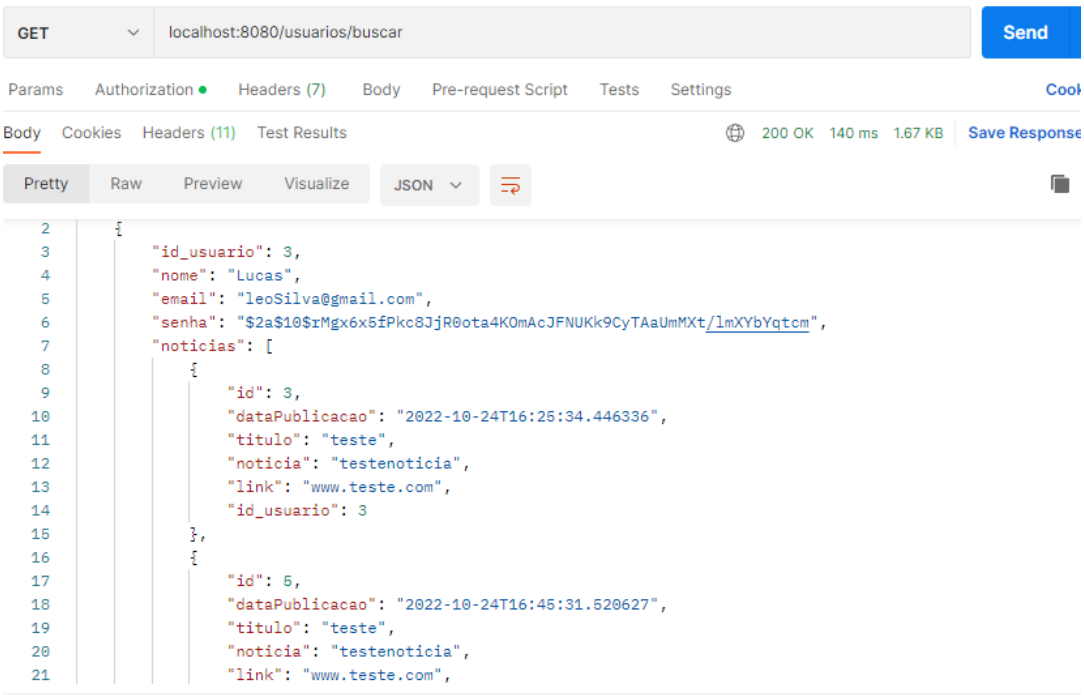


Figura 16: Método *GET* executado no POSTMAN

Fonte: Autoral, 2022

Na figura 16, vemos uma busca a um determinado usuário a partir de seu número de identificação (ID).

2. POST: É utilizado para mudança de estado de recurso no servidor, salvando as informações no banco de dados. Conforme acima na figura 17, podemos verificar as informações de um usuário que deseja depositar uma notícia ao sistema.

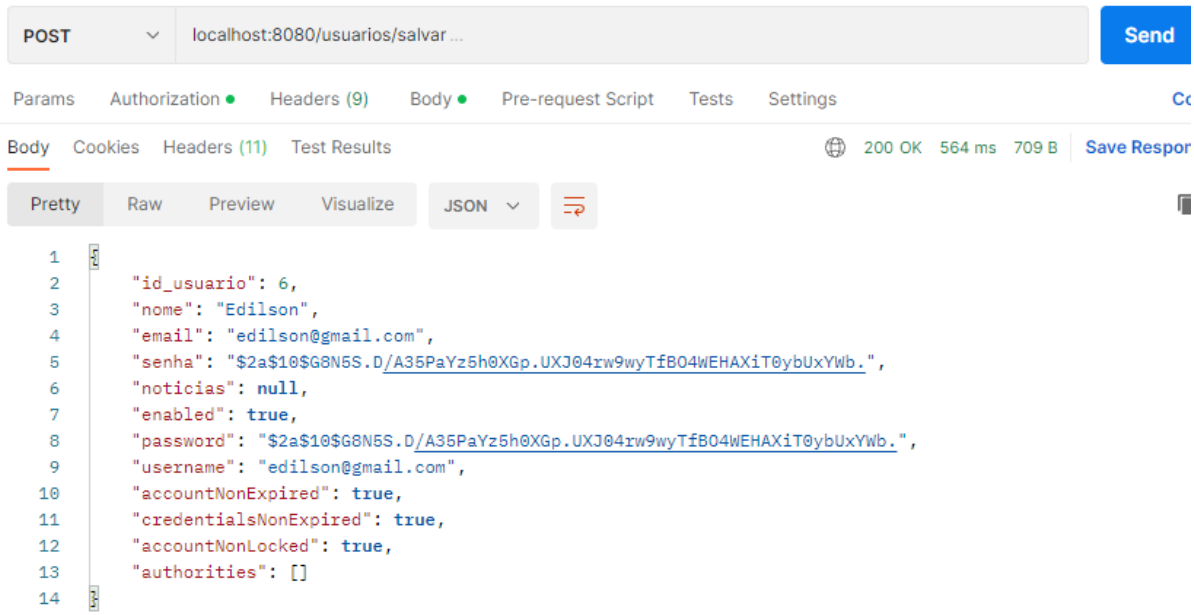


Figura 17: Método *POST* executado no POSTMAN

Fonte: Autoral, 2022.

3. DELETE: É utilizado para remover dados; Excluindo a informação ou o cadastro de forma permanente do sistema. na figura18, vemos o método deletando informações do ID2

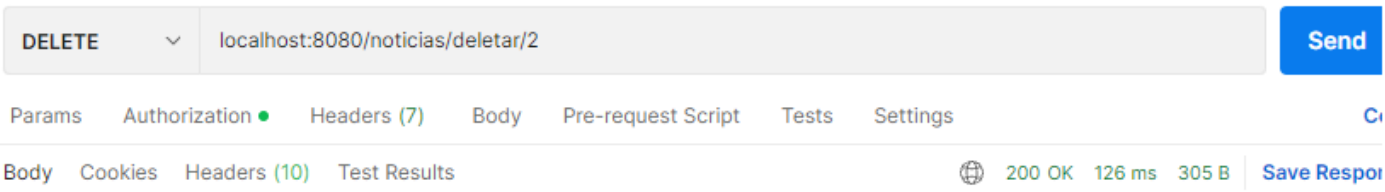


Figura 18: Método DELETE executado no POSTMAN

Fonte: Autoral, 2022.

Note que o código de retorno 200 OK, mostrando que o método foi executado com sucesso.

4. CONCLUSÃO

Concluímos que as aplicações desenvolvidas estão aptas a se tornarem uma solução eficaz para ajudar pacientes quanto aos seus tratamentos, horários de medicações e auxílio com informações uteis para o combate a vários tipos de doenças, além disso, tem a capacidade de guardar dados de pacientes que desejam se cadastrar proporcionando a eles uma melhor interação com os serviços.

A proliferação de doenças tem tido taxas elevadas ultimamente, sendo de fundamental importância um sistema que preste estes serviços a sociedade. Dessa forma, o sistema contribui para o aumento da acessibilidade de qualquer pessoa a informações e tratamentos médicos, uma vez que todas as informações depositadas são de responsabilidade de

seus autores, todos treinados e capacitados na área médica.

Com base nos estudos, algumas melhorias futuras que podem serem feitas para deixar o sistema ainda mais completo e eficiente para os clientes. Algumas delas são: 1 Comunicação com outras API's como as de transporte e entrega, podendo fazer a ligação entre as empresas que oferecem consultas e remédios e empresas que possam levar os pacientes aos determinados locais ou fazer a entrega dos produtos. 2 Conexão dos serviços dessa API com serviços públicos, implementando essa tecnologia e gerando acessibilidade a informações de necessidade públicas. 3 Gerenciamento e cadastro a planos de tratamento e consultas. 4 Adotar estratégias que transmitam os sistemas para outras plataformas visando atingir uma maior quantidade de pessoas.

AGRADECIMENTOS

Agradeço a Deus primeiramente, pela vida e toda benção que recebo dele, agradeço aos meus Pais, Sra. ALDYLENA MARIA PEREIRA SARAIVA e Sr. GEORGE MARCELO SILVA CARNEIRO por sempre batalharem por mim e sempre depositarem seu amor e esperança na minha pessoa, queria agradecer a minha avó materna, Sra. MARIA MADALENA PEREIRA, minha inspiração e motivação diária, obrigado por toda sabedoria e suporte em minha vida.

Agradeço também aos meus amigos e companheiros de faculdade, que prestam toda atenção e sempre tentam ajudar uns aos outros nos momentos de agonia. Ressalto também minha gratidão aos Professores da minha instituição, pois além de altamente capacitados, são pessoas de coração bom e que sempre agem em acordo ao que é de melhor ao aluno.

Este Artigo Acadêmico integra o curso de ENGENHARIA DA COMPUTAÇAO, na disciplina de TOPICOS INTEGRADORES ministrada por Edilson Lima.

Referências

FIELDING, Thomas. **Design of Network and Softwares Architetures**. University of Calfornia,2000. Acessado em 25 outubro de 2022.

GOUVEIA, Alexandre. **API, o que é**. Universidade Positivo, 2016 Acessado em 2 de novembro de 2022

LEITE, Thiago. **Orientação a objeto: aprenda seus conceitos e suas aplicabilidades de forma efetiva**. 2016 Acessado em 25 outubro de 2022

MOZILLA. **Métodos de Requisição HTTP.** MDN web doc 2020. Acessado em 2 de novembro de 2022.

7

A UTILIZAÇÃO DO SPRING BOOT PARA DESENVOLVIMENTO DE UMA API REST PARA CONSUMIR SERVIÇO EM UMA APLICAÇÃO

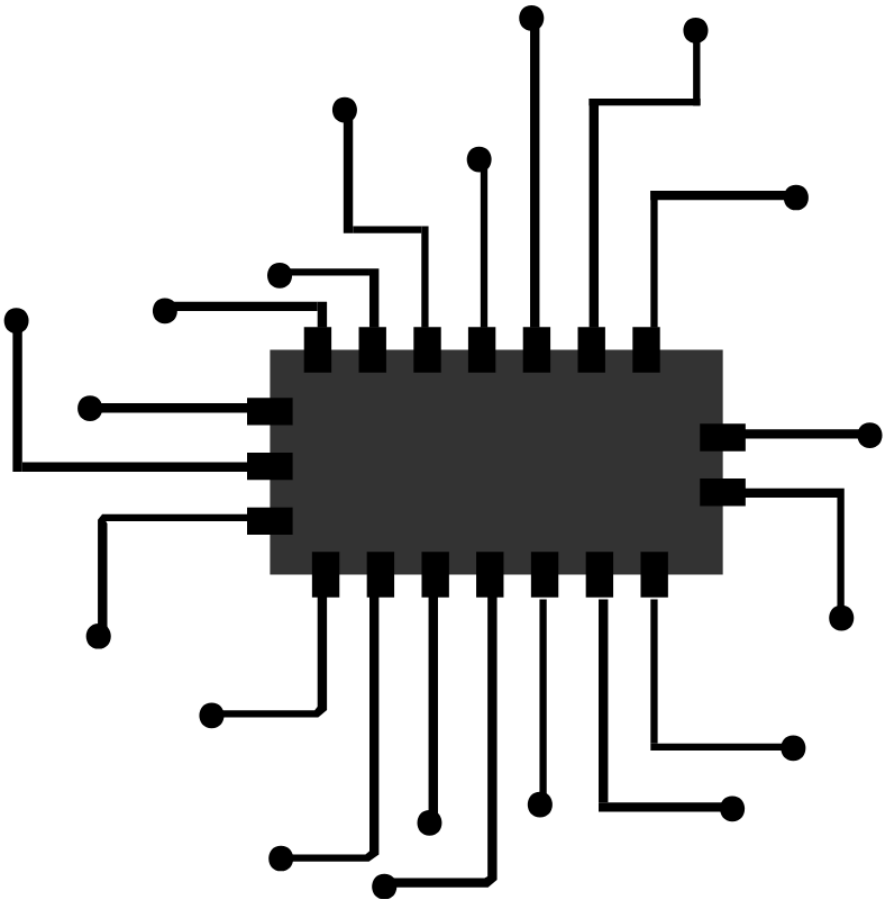
USING SPRING BOOT TO DEVELOP A REST API TO CONSUME SERVICE IN AN APPLICATION

Kauã Pereira Veras[1]

Edilson Carlos Silva Lima[2]

Yonara Costa Magalhães[3]

1Engenharia da Computação – Universidade Ceuma – 65.075-120 – São Luís – CEUMA – Brasil
2Engenharia da Computação – Universidade Ceuma – 65.075-120 – São Luís – CEUMA – Brasil
3Engenharia da Computação – Universidade Ceuma – 65.075-120 – São Luís – CEUMA – Brasil
{VERAS[1], Kauã, kauaveras@gmail.com; LIMA[2], Edilson Carlos Silva, edilsonlima3@gmail.com}



d.o.i.: 10.29327/5192626.1-7

Resumo

Com um dia a dia mais atarefado, as empresas e pessoas buscam agilizar suas tarefas, e quando esta tarefa está ligada ao transporte, os centros urbanos enfrentam dificuldades por causa de trânsitos caóticos, pela grande concentração de veículos, ao se deparar com um problema necessitam de um socorro emergencial, diante dessa problemática o objetivo desse trabalho é desenvolver o API REST utilizando Spring boot o framework para Java bastante popular que pode ser a construção de aplicação web tradicionais, será demonstrado como foi feito o desenvolvimento da API e colocado na pratica, o projeto foi separado as Stacks em front-end e back-end, utilizando framework Spring boot da plataforma Java e os dados serão utilizado em banco de dados relacionado ao Postman e MySQL. Este trabalho consiste em descrever o processo de construção de back-end conforme API REST utilizando o ecossistema Spring permitindo que um único sistema seja particionado conforme seus recursos.

Palavra-chave: *Back End*, *Spring Boot*, *API Rest*, ORM.

Abstract

With a busier day-to-day, companies and people seek to streamline their tasks, and when this task is linked to transport, urban centers face difficulties due to chaotic traffic, the large concentration of vehicles, when faced with a problem they need to of an emergency rescue, in the face of this problem, the objective of this work is to develop the REST API using Spring boot, the very popular framework for Java that can be used to build traditional web applications, it will be demonstrated how the API development was done and put into practice, the project was separated into front-end and back-end Stacks, using the Spring boot framework of the Java platform and the data will be used in a database related to Postman and MySQL. This work consists of describing the back-end construction process according to REST API using the Spring ecosystem, allowing a single system to be partitioned according to its resources.

Keywords: Back End, Spring Boot, API Rest, ORM.

1. INTRODUÇÃO

Este estudo consiste em demonstrar o desenvolvimento de API REST utilizando o ecossistema Spring, ou seja, uma série de módulos construídos para auxiliar desenvolvedores Java em seu trabalho.

O sistema precisa armazenar as informações dos usuários no banco de dados em MySQL, de forma remota para que o gestor ou colaborador consiga a partir de ter acesso ao sistema de sua organização e tomar sua decisão.

API REST é uma solução baseada em termo que define o endpoint e os métodos que têm permissão de acesso/envio de dados ao servidor. Falaremos sobre isso em mais detalhes abaixo. Outras alternativas são o GraphQL, o JSON e o oData.

Para resolver esse problema foi utilizado o framework Spring boot, o *Postman* e o *MySQL*, com essas ferramentas foram feitas com todas as etapas de camada de criação como *Back End* e um aplicativo desenvolvido em *Flutter* (*Front End*).

Este artigo está escrito em 4 capítulo, no capítulo 1 foi exclamado a problematização na Introdução, no capítulo 2 os tópicos necessários para o desenvolvimento do *back end* como referencial teórico, no capítulo 3 o estudo de caso, onde mostraremos como desenvolvemos o projeto e aplicamos o referencial teórico na prática e por fim no capítulo 4 com a conclusão.

2. FUNDAMENTO TEÓRICO

Neste capítulo foi realizada uma pesquisa com os assuntos de suma importância para o trabalho e foram estudados no item 2.1 Serviço *Web* para a comunicação entre as aplicações, *front end* e *back end*, no item 2.2 o *framework Spring boot* muito utilizado em projetos ágeis e organizados, no item 2.3 *API Rest* que é a interface de programação de aplicações (*API* ou *API web*) que está em conformidade com as restrições do estilo de arquitetura *REST*, permitindo a interação com serviços *web*, no item 2.4 *Postman* uma plataforma de *API* para desenvolvedores projetar, construir, testar e iterar *APIs*, no item 2.5 *Back End* que fica responsável pelos bastidores de uma aplicação, ou seja, os códigos que permitem a parte visual funcionar corretamente, no item 2.6 *Front End* onde o usuário pode interagir , e pôr o item 2.7 *MySQL* onde está todo o sistema de gerenciamento de banco de dados.

2.1 Serviço Web

Para Erl (2013), *Web Services* (daqui em diante tratado como Serviços *Web* ou *WS*) é uma 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 diferentes sejam compatíveis. Os *Web Services* são componentes que permitem às aplicações enviar e receber dados. Cada aplicação pode ter a sua própria «linguagem», que é traduzida para uma linguagem universal, um formato intermediário como XML, JSON, CSV etc.

Para as empresas, os *Web Services* podem trazer agilidade para os processos e eficiência na comunicação entre cadeias de produção ou de logística. Toda e qualquer comunicação entre sistemas passa a ser dinâmica e principalmente segura, pois não há

intervenção humana (TANENBAUM, 2015).

O objetivo dos *Web Services* (COULOURIS, 2003) é a comunicação de aplicações através da Internet. Esta comunicação é realizada com intuito de facilitar a *EAI* (*Enterprise Application Integration*) que significa a integração das aplicações de uma empresa, ou seja, interoperabilidade entre a informação que circula numa organização nas diferentes aplicações como, por exemplo, o comércio electrónico com os seus clientes e seus fornecedores. Esta interação constitui o sistema de informação de uma empresa. E para além da interoperabilidade entre as aplicações, a *EAI* permite definir um *workflow* entre as aplicações e pode constituir uma alternativa aos *ERP* (*Enterprise Resource Planning*). Com um *workflow* é possível otimizar e controlar processos e tarefas de uma determinada organização (WIKIWAND, 2022).

2.2 Spring boot

Spring é um *framework* usado para criar aplicações *web* em linguagem Java. Ele foi criado como uma alternativa às aplicações de programação que utilizam a tecnologia *EJB (Enterprise Java Beans)*, que não é muito apreciada pelos programadores. A primeira versão do *Spring Framework* foi criada em 2002 e, no entanto, seu desenvolvimento continua até hoje. Sua principal vantagem é que apesar de ser composto por muitos módulos, todos eles não precisam ser usados, o programador pode escolher os módulos que deseja usar, o que não sobrecarrega a aplicação com outros módulos desnecessários. Isso tem um impacto significativo no desempenho do aplicativo e na quantidade de recursos usados. (CÁSSIO MURILO, 2022).

O *Spring boot* foi mais utilizável para quaisquer programadores que pode fazer várias interações de atualização de arquitetura de software, no fim de reduzir o acoplamento de produtos de software e surgiu vários *frameworks* de micros serviços, e também é utilizado com linguagem Java.

2.3 API Rest

APIs abstraem, encapsulam e, idealmente, definem uma forma ergonômica de interagir com um módulo, independentemente de seu funcionamento interno (JOHNSON & FOOR, 1988).

Assim, os APIs são padrões que permitem desenvolver, elas podem criar as soluções como novas funcionalidades para criadores de aplicação disponível. O objetivo do API é fazer a integração do processo com facilidade para permitir utilizar a solução que pode ter atenção e satisfação do seu cliente. Várias bibliotecas e estruturas de software fornecem uma variedade de APIs para oferecer suporte a codificação segura.

Uma API RESTful funciona através da manipulação de recursos e representações. Essas representações são trocadas entre os usuários e o servidor através de uma interface padronizada e de um protocolo de comunicação específico — geralmente o HTTP (BARRO, 2022).

Para Liew (2021), quando um usuário deseja usar uma funcionalidade da aplicação, seu dispositivo envia uma solicitação via HTTP ao servidor. O servidor localiza o recurso e comunica a representação do estado dele na resposta ao usuário através do mesmo protocolo, como mostra a figura 1. E são essas representações que podem ser feitas em diversos formatos.

Figura 1. REST API.

Fonte: Liew, 2021.

As solicitações da aplicação requerem a execução de alguma função padrão no banco de dados do sistema. Por exemplo, a API RESTful envia ao servidor, via HTTP.

2.4 Postman

Postman é uma plataforma que só pode ser utilizado em API, ela tem uma conexão com SQL e faz o banco de dados preencher o formulário rápido e simples como construir e testar o seus APIs, esse programa começou do projeto paralelo de engenharia de softwares, Abhinav Asthana o criador do programa Postman. Ele teve a brilhante ideia de criar o *Postman* para ajudar outros programas a implementarem seus projetos com mais facilidade, os produtos são que usuário faça armazenamento dos documentos, orçamentos entre outros, após fazer programação já pode testar utilizando *POST* e *GET* para mostrar algo que realmente está funcionando (LOZOVEI, 2018).

A plataforma possui várias ferramentas que ajudam a acelerar as APIs com grande facilidade e a descoberta, e ela possui criar, compartilhar, testar e documentar APIs, para facilitar a criação para melhor *REQUESTS* como recurso ou funcionalidade para documentos APIs.

2.5 Back End

O *back-end* é escrito em linguagem Java e a Maven é usado para gerenciamento de projetos. O *framework usa spring boot+mybatis*. A razão pela qual a inicialização por mola é usada é que simplifica a configuração complicada. Se você configurar um projeto da web de mola de acordo com o horário usual precisará configurar web.xml, carregar mola e mola mvc, configurar conexão de banco de dados, configurar transação de mola e configurar arquivo de configuração de carregamento. Leia, abra anotações, configure arquivos de log... implemente a depuração do *tomcat* após a configuração, etc., no entanto, se você usar o *spring boot*, você só precisa configurar o pom e o arquivo de configuração do aplicativo (EQUIPE TOTVS, 2022).

Back End é linguagem para banco de dados e gerenciamento de informação como cofre para usa transação de informação como pagamentos, enviar documentos, armazenar dados pessoais, é responsável para fazer termo de qualquer implementação de APIs, MySQL e *Postman* (AUTORAL, 2022).

Não precisa se preocupar com o navegador do cliente, pois o código é armazenado em uma única máquina que pode configurar qualquer desenvolvimento, tem uma estrutura de operação do sistema como banco de dados e APIs que ele cuida de qualquer aplicação.

2.6 Front End

No desenvolvimento de grandes sistemas de software hoje, o *Front End* é responsável para desenvolver em *web* como interface gráfica, sem o *back* fica em desvantagem de fazer uns aplicativos ou web sem o banco de dados para armazenar informação dos usuários de forma mais prática, é a parte de um serviço web ou aplicativo da web que usuário ver diretamente sem ter tipo de controle como botão de iniciar, por isso é preciso de *back-end* para ter armazenamento de dados e botões que interagem com software de *Back End*, podem aprender ativar tecnologia como linguagem para melhorar sua habilidade de codificação de campo de atuação (QUEIRÓS, 2018)

O *Front-end* trabalha em foco de maiores camadas das interfaces como aplicação para usuários tocarem na tela de web site ou aplicativos móveis, o objetivo é fazer interfaces gráficas para os usuários que irão interagir diretamente, como site, aplicativos, softwares e entre outros.

2.7 MySQL

As duas abordagens amplamente utilizadas para modelagem de dados, a aplicação de *Unified Modeling Language (UML)* e *Object Role Modeling (ORM)* são comparadas. Como um dos propósitos da modelagem é apresentar e esclarecer o projeto do sistema, os principais critérios de comparação foram expressividade, ou seja, apresentação clara e fácil percepção das informações armazenadas e substancialidade, ou seja, capacidade de armazenar muitas informações. Foram escolhidas algumas situações características que frequentemente surgem na prática de modelagem de dados, e os exemplos mostram claramente como a mesma informação pode ser percebida de forma muito diferente em modelos ORM e UML (CARVALHO, 2015)

Assim, o artigo pode ajudar na seleção do framework de modelagem para um determinado projeto pois o MySQL são criação de bancos de dados para armazenar e manipular os dados pessoais em cada tabelas e pode ser aplicada no servidor como solicitação aparece no cliente.

2.8 ORM

O *Object Relational Mapper* significa Mapeamento Objeto-Relacional, são a mesma parceira com MySQL que faz mapeamento de bancos essa é a ideia de poder escrever consultas e também é uma técnica que permite fazer a relação com dados que mesmo representam e eles pode fazer muito desenvolvedores não sentirem vontade em escrever código SQL e por isso que existe ORM com Hibernate, NHibernate, Entity framework e entre outro. (AUTORAL, 2022).

Pois o ORM é justamente responsável pelo banco de dados de todas as classes, é ele quem vai permitir que você armazene os seus objetos no banco de dados com permissão de desenvolvedores para facilitar com mais segurança entre os *frameworks*.

3. ESTUDO DO CASO E RESULTADO E DISCUSSÃO

Neste capítulo, será mostrado o desenvolvimento da API, será colocado em prática os conceitos que foram abordados no capítulo 2. O projeto foi desenvolvido separando as

stacks em *Front-end* e *Back-end*. O presente artigo abordará o desenvolvimento do *Back--end* do projeto, que será desenvolvido utilizando o *Framework Spring Boot* da plataforma java, e os dados serão persistidos utilizando o banco de dados relacional Mysql.

3.1 O *App*

O uso de motocicletas é vantajoso, por ser um veículo leve, tem faixas exclusivas nas vias urbanas e transitam de forma mais rápida nas ruas. Quando esse veículo apresenta algum tipo de problema e precisa de um socorro emergencial, é necessário que se chame um prestador mais próximo que possa atender a esse usuário.

O aplicativo é uma ferramenta onde o motorista pode solicitar serviços emergenciais, onde parceiros do aplicativo recebem essa solicitação e aceitam o serviço de socorro, podendo resolver o problema de forma rápida e segura, pois os parceiros passam por avaliações dos serviços prestados, agilidade, cordialidade, rapidez entre outros.

3.2 Desenvolvendo a API Rest

Uma API Rest permite a comunicação entre aplicações, ela estabelece um conjunto de protocolos e definições para a integração de uma aplicação. Uma API é capaz de enviar requisições *HTTP*, por meio dos métodos, é possível realizar operações com a finalidade de alterar e ler as informações do banco de dados, os métodos mais comuns são *POST, GET, PUT e DELETE,* como mostra a figura 2.

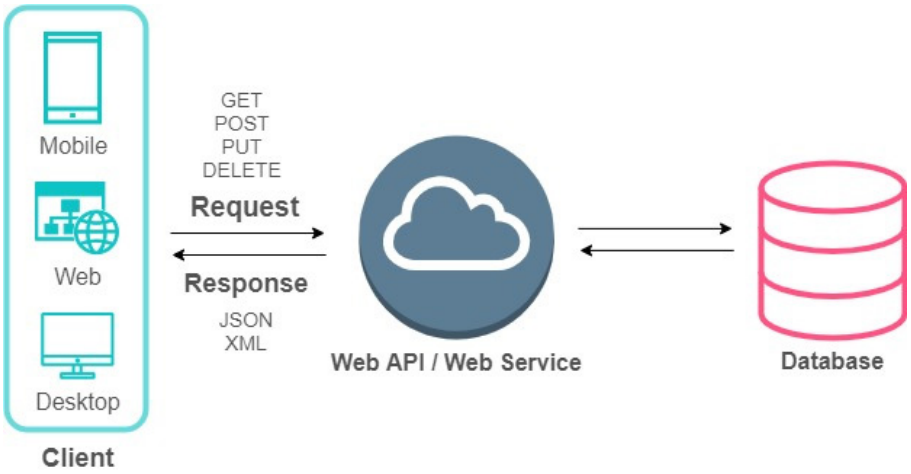


Figura 2: codificação segura existente em JAVA.

Fonte: Autoral, 2022.

A implementação de API Rest permite com que a aplicação seja multiplataforma, permitindo o acesso a várias plataformas, tanto web quanto *mobile*, isso só é possível devido a separação entre o *front-end* e *back-end*, isso permite com que o desenvolvedor foque na regra de negócio da aplicação.

3.3 Banco de Dados

O banco de dados tem como finalidade armazenar e organizar informações do sistema, em um projeto Spring o banco de dados é criado e manipulado pelo *Hibernate,* que é uma ferramenta de mapeamento objeto relacional para a plataforma java, fornecendo

funcionalidades para persistir o banco de dados.

Foi utilizado o *Framework* (Spring Boot) com MySQL para desenvolver as classes de recurso dos dados, utilizamos MySQL (figura 3), para fazer os dados de usuário, atendimentos e motorista.

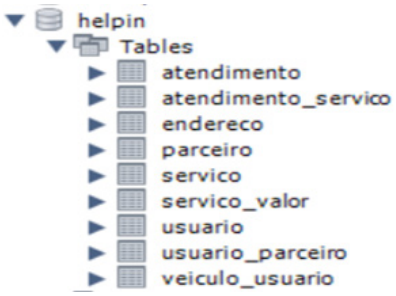


Figura 3: Todas as tabelas do Banco de dados.

Fonte: Autoral, 2022.

A figura 4, mostra a tabela de "atendimento", ela contém as informações de atendimento do cliente, onde se armazena o local de encontro, lugar a qual será realizado o atendimento, também é armazenado o veículo utilizado pelo parceiro para a realização do atendimento.

Result Grid | Filter Rows: | Edit:

id_atendimento	encontro	id_usuario	veiculo
1	Rua afonso pena, centro	1	uno
2	Rua sucupira	2	celta
3	praça maria aragão	3	palio

Figura 4: Tabela atendimento.

Fonte: Autoral, 2022.

A figura 5, mostra a tabela de serviço, nela se armazena as informações referentes ao serviço prestado pelo parceiro, nela contém o nome do parceiro e a chave estrangeira referente ao valor e serviço.

Result Grid | Filter Rows:

id_servico	idservico_valor	nome
1	1	gabriel
2	2	Manoel
3	3	Victor

Figura 5: Tabela serviço.

Fonte: Autoral, 2022.

Na figura 6, mostra tabela de usuário, que é o cliente da aplicação, nela contém as informações essenciais para a construção do perfil do usuário.

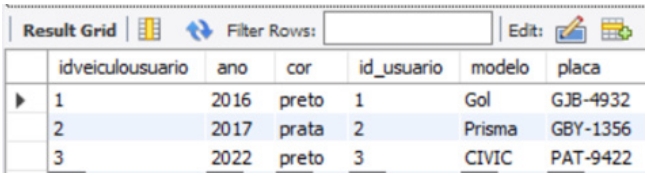
Result Grid | Filter Rows: | Edit: | Export/Import:

id_usuario	cpf	celular	email	nome
1	2894763901	98986749312	esmeralda@gmail.com	Esmeralda veras
2	93784673291	98985643872	leandro@gmail.com	Leandro da silva
3	01735897310	98984631441	kauaveras@gmail.com	kauã veras
NULL	NULL	NULL	NULL	NULL

Figura 6: tabela usuário.

Fonte: Autoral, 2022.

A imagem 7, é a tabela veículo-usuário nela contém informações do carro do cliente, essas informações serão úteis para o parceiro pois ela servirá de referência para a manutenção do veículo.



Result Grid | Filter Rows: | Edit:

idveiculousuario	ano	cor	id_usuario	modelo	placa
1	2016	preto	1	Gol	GJB-4932
2	2017	prata	2	Prisma	GBY-1356
3	2022	preto	3	CIVIC	PAT-9422

Figura 7: tabela veículo-usuário.

Fonte: Autoral, 2022.

O banco de dados do projeto é gerado a partir das classes criadas e declaradas como modelo, o Spring utiliza o módulo Spring Data, para fazer a persistência com o banco de dados.

3.4 Camada de aplicação

Os modelos de cada classe, possui sua função de ID para deixar usuário ligado com banco de dados, que são 3 funções básicas usado em java:

- ***Controller*:** o controlador não deve criar nada, ele usa uma fábrica para adquirir o objeto que você precisa
- ***Model*:** deve pegar informação do usuário da solicitação
- ***Repository*:** É para separar os objetos de domínios dos mapeadores de dados.

3.4.1 Camada Repository

A classe *Repository* são armazenamento de dados no *back-end* que suportam a criação da página de dados e classificação quando apropriadamente, como dados pessoais na interface. Na classe é estendida uma biblioteca JPARepository que por ele passa dois parâmetros, um é a classe *model* (modelo) que vai ser utilizada no *repository* e a *PrimaryKey* dessa classe que é do tipo *Long.*

UsuarioRepository.java

```
1  package com.project.Helpin.repository;
2
3  import org.springframework.data.jpa.repository.JpaRepository;
6
7  public interface UsuarioRepository extends JpaRepository<Usuario , Long> {
8
9  }
```

Figura 8: Classe *Repository*.

Fonte: Autoral, 2022.

3.4.2 Camada Controller

Na figura 9 cada objeto tem a função de desenvolver a classe, *Controller* tem cada objeto tem o codigo de dados retornados para cada método gravado diretamente no cor-

po da resposta em vez de executar um modelo, a classe *Repository* tem injetado pelo construção de *Controller* porque o mapeamento de dados, cada operação de mapeamento tem: *@GetMapping*, *@PostMapping* e *@DeleteMapping* que corresponde a Postman *GET, POST* e *DELETE*.

*UsuarioController.java

```
1  package com.project.Helpin.Controller;
2
3
4
5  import java.util.List;
21
22
23 @RestController
24 @RequestMapping("/usuario")
25 public class UsuarioController {
26
27     @Autowired
28     private UsuarioRepository usuarioRepository;
29     @PostMapping
30     @ResponseStatus(HttpStatus.CREATED)
31     public Usuario adicionar(@RequestBody Usuario usuario) {
32         return usuarioRepository.save(usuario);
33     }
34     @DeleteMapping("/{id}")
35     public ResponseEntity<Usuario> remover(@PathVariable Long id){
36
37         Optional<Usuario> opcional = usuarioRepository.findById(id);
38         if (opcional.isPresent()) {
39             usuarioRepository.deleteById(id);
40
41             return ResponseEntity.ok().build();
42         }
43
44         return ResponseEntity.notFound().build();
45     }
46     @GetMapping
47
48     public List<Usuario> listar(){
49         return  usuarioRepository.findAll();
50     }
51
52 }
```

Figura 9: Classe *Controller.*

Fonte: Autoral, 2022.

Sobre a solicitação de mapeamento como *PostMAPPING*, *GetMAPPING* e *DeleteMAPPING* no *spring* são anotação para mapear todas as URLs de solicitação HTTP recebidas para os métodos do controlador correspondentes como:

3.4.2.1 Método GET

Na figura 10, é usado método do verbo GET é usado para fazer a listagem de determinada classe, ele pega os dados do banco através do *repository*, usando a função findAll que puxará todas as informações dessa classe para o método. em cima do método deve usar o ***@GetMapping.***

Existe outro método do verbo GET, esse responsável por buscar dados de um usuário específico através de um parâmetro, normalmente usado o ID do usuário, usando o DTo para filtrar determinados atributos que deseja obter através desse método.

```
@GetMapping

public List<Usuario> listar(){


    return  usuarioRepository.findAll();
}
```

Figura 10: utilizando método GET.

Fonte: Autoral, 2022.

3.4.2.2 Método POST

@PostMapping é o métodos anotados nas classes *@Controller* tratam como solicitação HTTP *POST* que corresponde uma determinada expressão de URI fornecida para inserir novos dados na API, que ele indica os valores que iremos adicionar, como o exemplo como escrever métodos de controlador mapeados com anotações *@PostMapping*, esse método usa a anotação @RequestBody que é responsável para que esse método tem que passar um atributo no corpo da requisição, e depois que o usuário passar os dados, irá retornar o status 201 Created, e salvar no repository.

```
@PostMapping
@ResponseStatus(HttpStatus.CREATED)
public Usuario adicionar(@RequestBody Usuario usuario) {
    return usuarioRepository.save(usuario);
}
```

Figura 11: utilizando método POST.

Fonte: Autoral, 2022.

3.4.2.3 Método Delete

Esse método que tem por finalidade a exclusão de dados da API, O usuário passar na requisição o parâmetro do tipo ID do objeto que vai ser excluído. antes disso o Spring vai verificar se esse ID existe no banco de dados, com o método *Optional.isPresent* e se esse ID existe, será excluído e voltará um status *NotFound* dizendo que esse objeto não pode ser mais encontrado.

```
@DeleteMapping("/{id}")
public ResponseEntity<Usuario> remover(@PathVariable Long id){


    Optional<Usuario> opcional = usuarioRepository.findById(id);
    if (opcional.isPresent()) {
        usuarioRepository.deleteById(id);

        return ResponseEntity.ok().build();
    }

    return ResponseEntity.notFound().build();
}
```

Figura 12: utilizando método DELETE.

Fonte: Autoral, 2022.

3.5 *Postman*

Postman é uma ferramenta que é utilizada em API, ela faz a uma conexão com o banco de dados para que os usuários possam testar o seus APIs, esse programa começou do projeto paralelo de engenharia de softwares, Abhinav Asthana o criador do programa *Postman*. Ele teve a brilhante ideia de criar o *Postman* para ajudar outros programas a implementarem seus projetos com mais facilidade, os produtos são que usuário faça armazenamento dos documentos, orçamentos entre outros, após fazer programação já pode testar utilizando *POST* e *GET* para mostrar algo que realmente está funcionando.

As ferramentas Postman:

- ***GET***: É uma solicitação de um recurso específico que pode retornar apenas dados.
- ***POST***: É utilizado para mudança de estado do recurso ou efeito no servidor (banco de dados).
- ***DELETE***: Remover os dados.
- ***TOKEN***: Autenticação de autorização para descriptografar.
- **URI**: Sistema de listagem de clientes.

Pelo exemplo da figura 13 uma API tem a requisição de *HTTP* com verbo de *GET* na URI "http://localhost/auto", isso faz similar qualquer implementação do API em desenvolvimento.

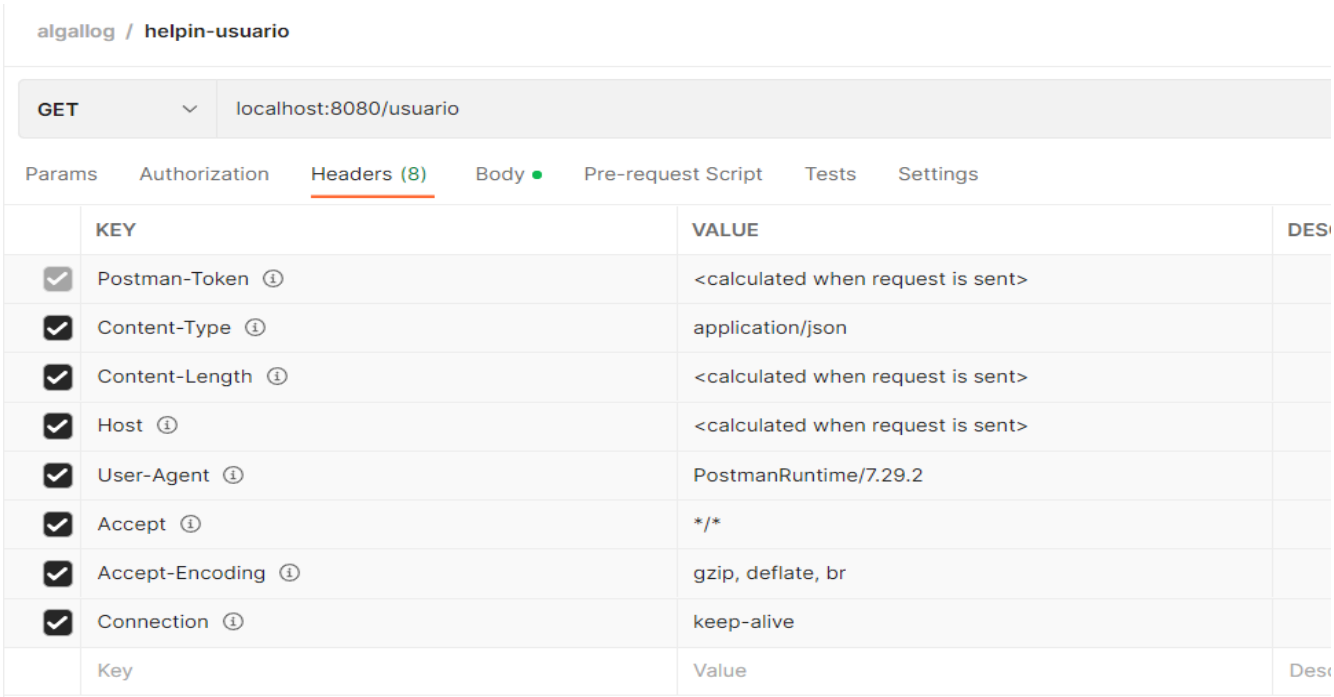


Figura 13: Postman API cliente.

Fonte: Autoral, 2022.

Na autenticação, o usuário precisa fazer um login para poder gerar o *token*, através dele pode se acessar ou realizar determinadas ações do endpoints. com isso o usuário faz o login, e é gerado um token que tem um determinado tempo, depois é colocado no *authorization* e no cabeçalho é selecionado o tipo do *token* "*Bearer*" que por sua vez aparece o espaço para adicionar o token, e o JWT (Json Web Token) vai fazer uma verificação se o token é válido ou não. se for válido o usuário pode fazer a ação que deseja. Como é demonstrado na figura 14.

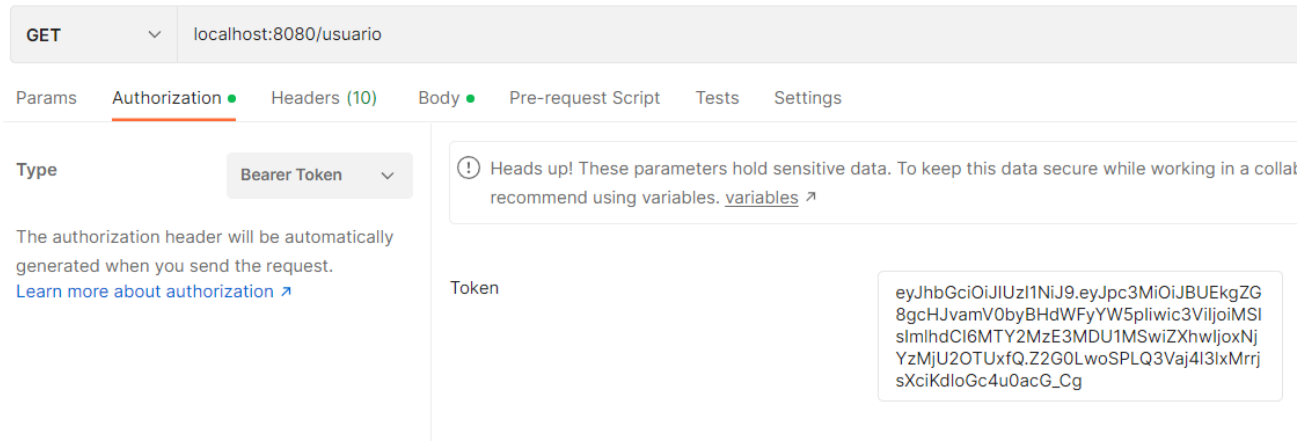


Figura 14: Postman TOKEN.

Fonte: Autoral, 2022.

Na figura 15, é demonstrado a utilização do verbo POST para criação dos dados do usuário, esse método foi usado na URI localhost:8080/usuário, mas para utilizar esse método não precisa de token, por ser um endpoint público. O usuário preenche seus dados nos atributos e depois envia a requisição para o DB, que por sua vez, se ocorrer tudo certo, irá retornar um HTTP do status ***201 Created.***

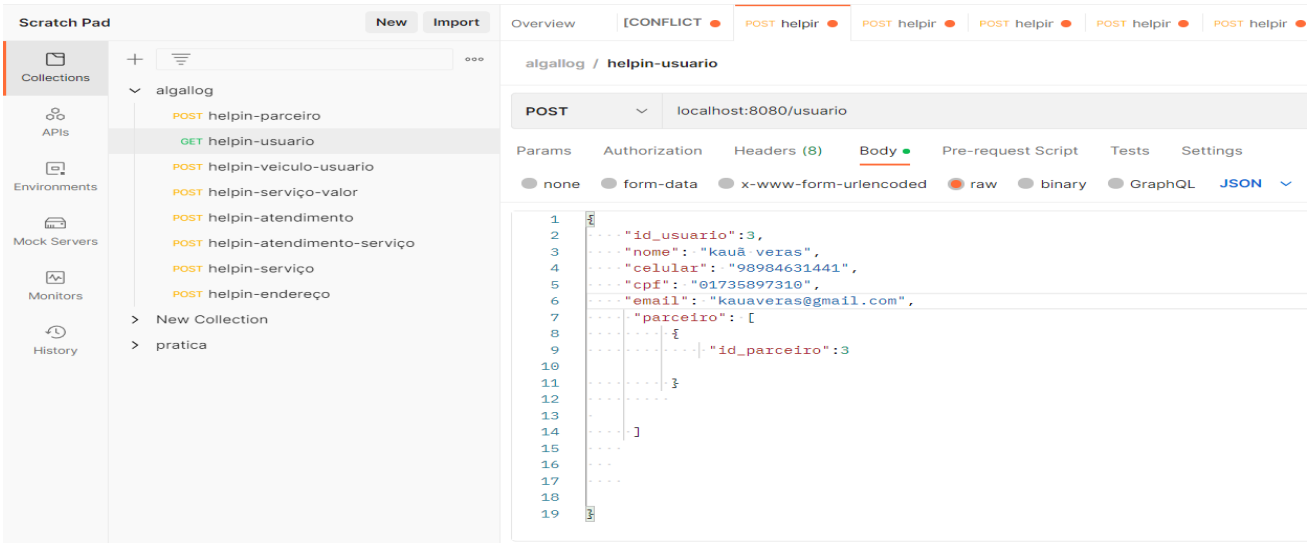


Figura 15: Postman utilizando POST.

Fonte: Autoral, 2022.

No método do verbo GET é mostrado na figura 16, os dados que foram cadastrados no DB (database) nesse método foi solicitado para mostrar todas as informações que está guardado no banco *GetAll* e por esta puxando através do model, em vez de um DTo, o *Postman* pega todas as relações que está junto da classe model usuário.

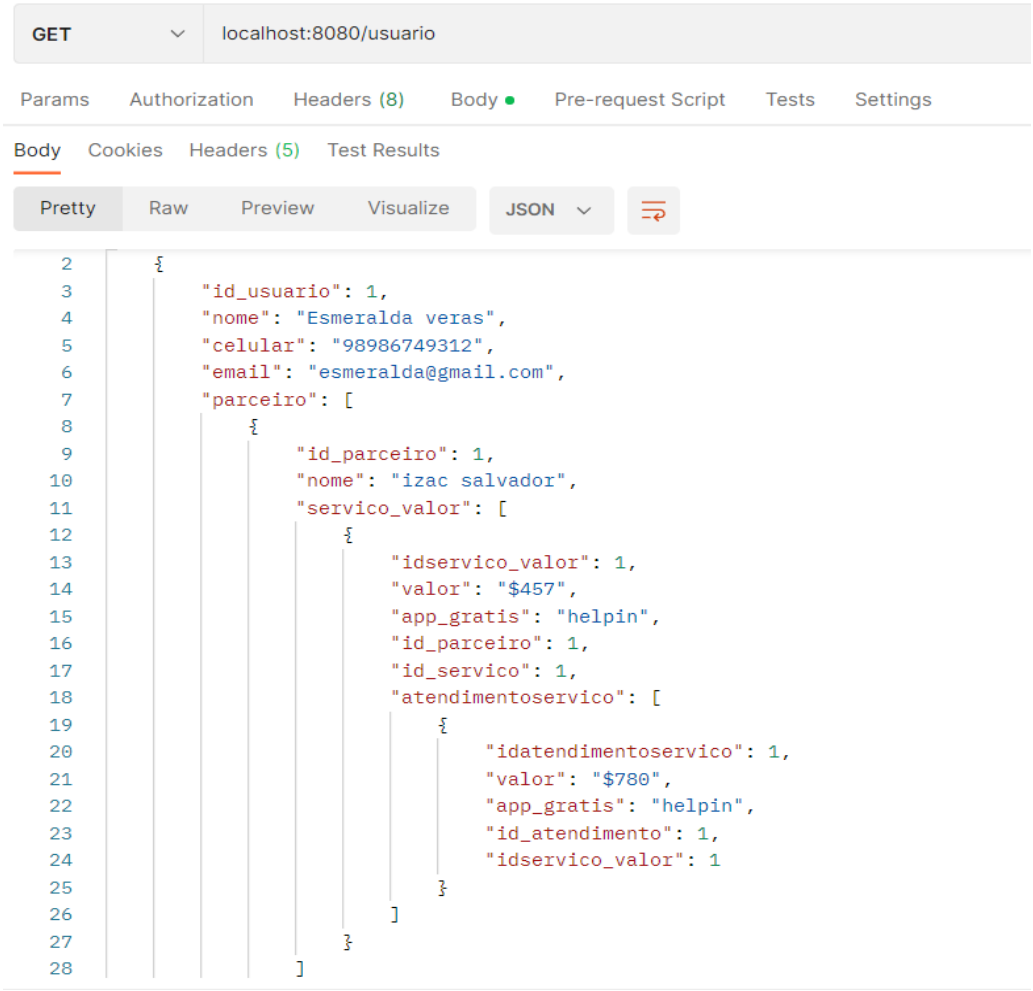


Figura 16: Postman utilizando Get.

Fonte: Autoral, 2022.

A solicitação Postman *DELETE* exclui um recurso já presente no servidor. O método *DELETE* envia uma solicitação ao servidor para excluir a solicitação mencionada no endpoint. Assim, é capaz de deletar o dado no servidor.

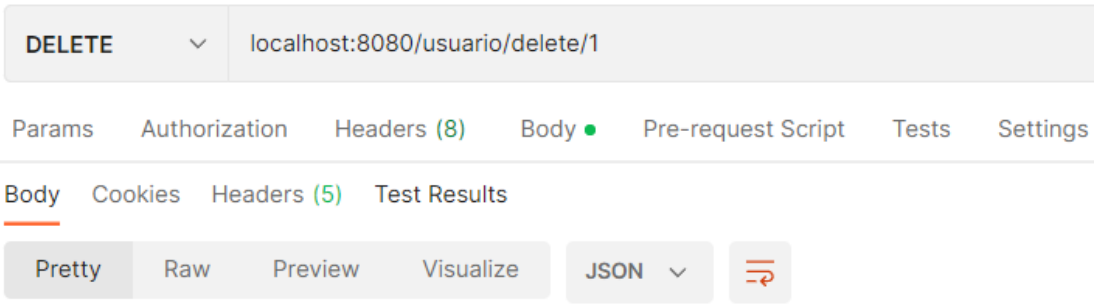


Figura 17: Postman utilizando DELETE

Fonte: Autoral, 2022

3.6 Consumindo serviço da API Rest através do aplicativo

API Rest em flutter utiliza a busca de dados, usando um arquivo JSON. Em um aplicativo é muito comum o seu uso, são capazes de exibir os dados, que fornece métodos avançados para realizar várias operações. A API Rest usa chamadas HTTP simples para se

comunicar com dados JSON, pois o HTTP usa o recurso de espera, fornece vários métodos e realiza requisição web.

```
ListTile(
  leading: Icon(Icons.person_sharp),
  title: Text("Nome Completo"),
  subtitle: Text(
      "${widget.userInfo['Name']}"), // Text
), // ListTile
ListTile(
  leading: Icon(Icons.email),
  title: Text("E-mail"),
  subtitle: Text(widget.userInfo['email']),
), // ListTile
ListTile(
  leading: Icon(Icons.phone),
  title: Text("Telefone"),
  subtitle: Text(widget.userInfo['telefone']),
), // ListTile
Visibility(
  visible: show,
  child: ListTile(
    leading: Icon(Icons.person_pin),
    title: Text("Data de Nascimento"),
    subtitle: Text(widget.userInfo['data']),
```

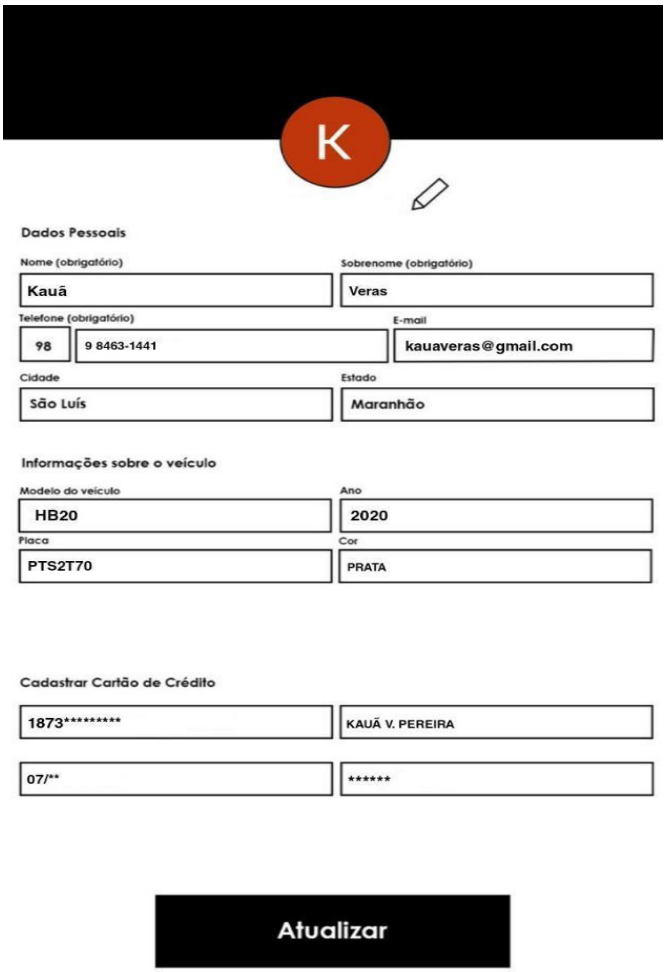


Figura 18-19: Trecho de código e perfil do usuário.

Fonte: Autoral, 2022.

O API Rest foi desenvolvido na base do flutter utilizando o URL da API do Spring com link de localhost:8080/swagger.ui para que o API fica online usando heroku, e também utilizando esse trecho de código para armazenar os dados de usuário como nome, identidade, endereço, cartão e entre outro.

O *Heroku* é uma outra plataforma que nos permite hospedar código e não se preocupar muito com a disponibilidade, escala e infraestrutura da aplicação. Ela é mais utilizada para aplicações de *back-end*, como as desenvolvidas em Node. js, Ruby, Java, PHP, Python, Go, entre outras, para rodar o API online para outra máquina do *Front-end*.

O Spring *Initializr* é uma ferramenta baseada na Web que gera a estrutura do projeto Spring Boot . O erro ortográfico em *initializr* é inspirado em *initializr* . Os IDEs modernos integraram o Spring *Initializr*, que fornece a estrutura inicial tornando assim mais fácil para os desenvolvedores selecionar a configuração necessária para seus projetos. A ferramenta Spring Initializr cuida da seguinte configuração para qualquer projeto baseado em Spring. Ao iniciar o *Spring Boot* é preciso criar o arquivo, e a primeira coisa é ir no site do *Spring boot* <https://start.spring.io/>

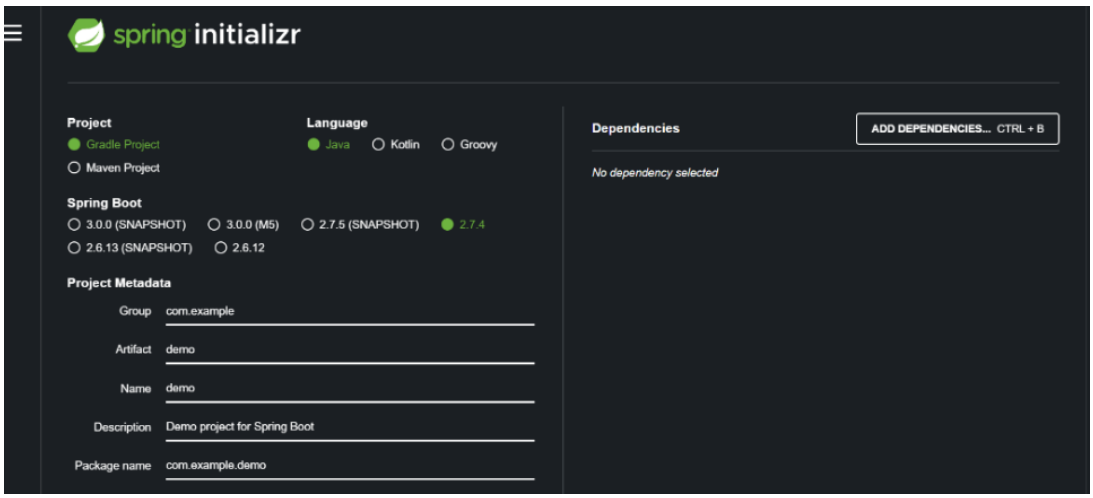


Figura 20: site Spring Boot initialize.

Fonte: Autoral, 2022.

4. CONCLUSÃO

Neste artigo foi desenvolvido uma API para a criação de um banco de dados para produtos de borracharia em uma loja. A funcionalidade foi implementada com mecanismo de Postman e MySQL em busca dos dados de usuários. Esse aplicativo faz com que o usuário possa solicitar serviços de mecânicos de qualquer lugar.

O ecossistema Spring é uma ferramenta poderosa para um desenvolvedor, é usada amplamente pela comunidade Java. Vale destacar sobretudo a produtividade proporcionada por ele, diminuindo o tempo e consequentemente os custos de desenvolvimento.

A criação de um banco de dados de usuários foi importante para realizar o armazenamento de recursos provenientes do aplicativo, as funcionalidades do projeto e a conexão entre clientes e usuários.

Referências

BARANCELLI, Gaspar, **Banco de dados MySQL com Spring Boot**, 2021. Disponível em:<https://gasparbarancelli.com/post/banco-de-dados-mysql-com-spring-boot?lang=pt>. Acesso em 19 de out de 2022.

BARRO, Bruna B. **API REST**. Publicado em maio/2022. Disponível em: https://www.hostinger.com.br/tutoriais/api-restful. Acesso em: 13 de out de 2022.

BOAGLIO, Fernando. **Spring Boot, Acelere o desenvolvimento de microsserviços**,2017. Disponível em:<https://www.casadocodigo.com.br/products/livro-spring-boot>. Acesso em: 19 de out de 2022.

CARVALHO, Vinícius, MySQL **Comece com o principal banco de dados open source do mercado**, 2015. Diponivel em:<https://www.docsity.com/pt/vinicius-carvalho-mysql-comece-com-o-principal-banco-de-dados-open-source-do-mercado-1-casa-do-codigo-2015/4932470/> Acesso em: 21 out de 2022.

Equipe TOTVs. **O que é back-end e qual seu papel na programação?**, 2020. disponível em: https:<//www.totvs.com/blog/developers/back-end/>. Acesso em 28 de out de 2022

ERL, Thomas. **SOA – Princípios de Design de Serviços.** São Paulo: Editora Pearson, 2013.

FERREIRA, Rodrigo: **API REST com o framework Spring Boot**, 2022. Disponível em:<https://www.alura.com.br/curso-online-spring-boot-api-rest> Acesso em 17 de out de 2022.

JOHNSON, Ralph E.; FOOTE, Brian. **Designing Reusable Classes.** Publicado em 1988. Disponível em: https://www.researchgate.net/publication/215446177_Designing_Reusable_Classes. Acesso em 17 de out de 2022.

LOZOVEI, Julio. **Indo além com Postman, 2018**. disponível em: https:<//medium.com/trainingcenter/indo-além-com-postman-3f95726e0bb4>. Acesso em 29 de out de 2022.

LIEW, Zell. **Understanding And Using REST APIs**. Publicado em maio/2021. Disponível em: https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/. Acesso em: 13 de out de 2022.

MURILO, Cássio. **O que é um Framework?**. ALURA, 2022. Disponível em: https://www.alura.com.br/artigos/spring-conheca-esse-framework-java. Acesso em 28 de out de 2022

QUEIRÓS, Ricardo. **Introdução ao Desenvolvimento Moderno Para a Web. Do Front-End e Back-End. Uma visão Global!,** 2018. Disponível em:<https://static.fnac-static.com/multimedia/PT/pdf/9789727229154.pdf> Acesso em: 19 de out de 2022.

SAUDATE, Alexandre. **APIs REST, Seus serviços prontos para o mundo real**,2021. Disponível em:<https://www.casadocodigo.com.br/products/livro-apis-rest>. Acesso em: 19 de out de 2022.

TANENBAUM, Andrew S. **Distributed Operating Systems**. Pearson universidade, 2015.

WOJCIECHOWSKI, Nicolas, **Criando back-end API com Spring boot 2.0**, 2020. Disponível em:<https://kbase.com.br/2020/09/16/criando-back-end-api-com-spring-boot-2-0/>. Acesso em 13 de out de 2022.

Wikiwand, **Web service**, 2015. Disponivel em : <https://www.wikiwand.com/pt/Web_service>. Acesso em 28 de out de 2022.

8

APP DE VENDAS: APLICAÇÃO EM FRAMEWORKS PARA MELHORAR O SISTEMA DE CONTROLE DE PEDIDOS UTILIZANDO FLUTTER E API

APP OF SALES: APLICATION IN FRAMEWORK TO IMPROVE THE ORDER CONTROL SYSTEM USING FLUTTER AND API

João Victor Vieira Beckman[1]

Edilson Carlos Silva Lima[2]

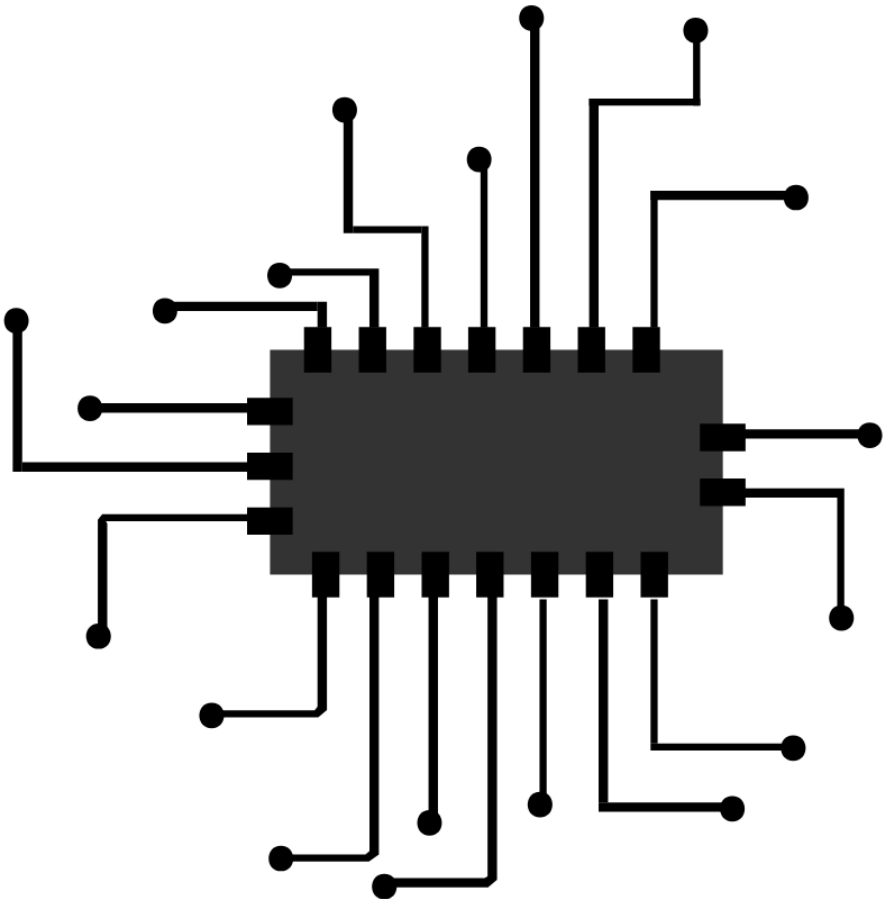
Yonara Costa Magalhães[3]

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

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

31Engenharia de Computação – Universidade Ceuma (UniCEUMA) – São Luís – MA – Brasil

{BECKMAN, João Victor Vieira, jvvbeckman123@gmail.com; LIMA, Edilson Carlos Silva, edilsonlima3@gmail.com.}



d.o.i.: 10.29327/5192626.1-8

Resumo

Este artigo aborda um desenvolvimento móbile, que tem como foco principal o estudo do flutter e as altas demandas de produtos para suprir a necessidades de clientes, assim como os bons negócios e as relações das empresas. Todos sendo vistos, para manter uma estrutura e qualidade boa para os diversos ambientes de serviços, que venham tendo mais necessidades de se reinventarem e aprimorar de forma para satisfazer o cliente e prover resultados efetivos, na qual o aplicativo tem como função facilitar, oferecer mais recursos para atender toda a mercadoria, e desenvolver um app de gerenciamento de vendas, para melhorar todos os trabalhos da empresa e tarefas para os clientes como: solicitação de pedidos, compras dos produtos, localização e ofertas.

Palavras-chave: Flutter, API, Framework, UML, Front-end, Back-end.

Abstract

This article template covers a mobile develepment, whose main focus is the study of flutter and the high product demands to meet customer needs, as well as good business and company relationships. All being seen, to maintain a good structure and quality for the different service environments, that have more needs to reinvent themselves and improve in order to satisfy the customer and provide effective results, in which the application has the function of facilitating, offer more features to meet all the merchandise, and develop a sales management app, to improve all company jobs and task for customers like: order request, product purchases, localization and offers.

Keywords: Flutter, API, Framework, UML, Front-end, Back-end.

1. INTRODUÇÃO

Por meio de vários estudos, a programação se tornou nos últimos anos, uma das maiores funções e profissões no mercado de trabalho para muitas empresas, tendo solução para diversos problemas, que atualmente são bem mais simples de resolver usando a tecnologia. Foram criados muitos aplicativos, para ajudar no desenvolvimento de altas demandas de serviços, e visto que nos dias de hoje, a aplicação dos celulares é uma ferramenta obrigatória para atender e resolver qualquer problema, principalmente para priorizar o ambiente de trabalho e aprimorar um novo método de comunicação.

Inúmeras cidades e até estados vem sofrendo com a baixa e alta demanda de água e gás, pois são os produtos que são muito importantes no consumo do dia a dia. Uma das soluções é fazer com que essas demandas venham sendo ajustadas, de uma forma mais organizada e estruturada para as empresas que vendem os produtos, assim como os clientes que compram e necessitam dos produtos. Na maioria dos casos, muitos serviços que são realizados sem nenhuma informação clara ou até mesmo uma desinformação, custam caro para os clientes, e resulta até mesmo em maus negócios para os vendedores.

O principal objetivo é realizar um estudo sobre os problemas do controle de gerenciamento de pedidos das empresas, levando em consideração análises e recursos por meio da quantidade e qualidade, que serão utilizados e explicados para mostrar resultados em meio ao condicionamento do cliente e dos vendedores. Por meio do projeto, a função é fazer com que seja desenvolvido uma aplicação, de forma para priorizar as tarefas e as demandas dos serviços das empresas, e para os clientes que realizam os pedidos, logo tem-se de apresentar funcionalidades, mecanismos e novas estruturas utilizando o aplicativo de vendas.

2. REFERENCIAL TEÓRICO

No mundo da tecnologia toda a aplicação construída ou criada, possui uma estrutura que é desenvolvida por diversas linguagens de programação e frameworks, nos quais fornecem as ferramentas essenciais para serem usadas no aprimoramento de vários sistemas como: Nativos (Android ou IOS), Híbrido (Web ou Móbile) e WebApp (Sites ou Dispositivos Móveis).

Todos os recursos apresentados são utilizados em diferentes ambientes, o projeto da aplicação é focado nos Nativos, que são trabalhados os programas para a construção dos apps. Por meio desse sistema, é necessário conhecer alguns dos recursos e linguagens, que visam mostrar como cada estrutura de um aplicativo funciona de forma eficiente. Dentre eles são: Framework, Spring, Rest, API, Flutter, UML, Front-end e Back-End.

2.1 Framework

Framework significa uma estrutura, que apresenta soluções para um conjunto de códigos, que vem sendo implementado em basicamente todos os sistemas em linguagens bem específicas, no caso para desenvolvimentos móbiles, como java e kotlin. Além disso, o uso dos frameworks no processo de projetos de software, fez parte para contribuição de alta qualidade na realização de programas avançados, que tiverem destaques nos tempos atuais, mostrando não só uma evolução, como uma inovação que vem cada vez mais tornando

o ambiente organizado, seguro e estruturado (SOMMERVILLE, 2007; PRESSMAN, 2011).

De acordo com Tomhave (2005), o framework é definido como uma forma básica de um software, que mostra uma fundamentação que estabelece pressupostos, conceitos, valores e práticas que na própria diretriz ocorre para a execução de um sistema completo da aplicação. Diante disso, as técnicas, os métodos e as ferramentas de todo o código do projeto, conseguem obter resultados que auxiliam na melhoria por meio do framework, não só dos apps, como os sites, programas móbiles, webs e até um sistema de criação de jogos.

Além dos princípios e diretrizes, o framework possui no uso da maior parte da estrutura do código fonte do sistema, em que o mesmo está determinado no ambiente ativo no app, e promove o reuso de conteúdo do domínio de um software ou encontrar formas para minimizar os problemas que são complexos e difíceis de serem localizados nos cenários do sistema (GIMENES; HUZITA, 2005). O reuso proporcionado com frameworks não se restringe à apenas implementação, busca-se permite uma análise direta do projeto de software, que adiciona elementos a serem estudados como os constructos e as suas relações com o ambiente da programação.

Nos próximos resultados a serem discutidos sobre o framework, é como ele pode ser trabalhado e funcional para os softwares implementados no projeto, tendo um desenvolvimento por meio de modelo que esteja padrão a vista do usuário (ROCHA; BARANAUSKAS, 2003).

2.2 Spring

O Spring é um framework para a plataforma Java e trabalha para simplificar as aplicações. O mais importante sobre este programa, é que de forma opcional é das ferramentas que realizar tarefas por base dos conceitos chamado de convenções sobre as configurações, ou seja, significa que é muito efetivo, para melhorar e seguir um caminho para desenvolver seu código fonte de projeto, utilizando muito recursos e as configurações necessárias para o funcionamento da app (RODERICK, 2002).

Através de estudos e análises do Spring, é a peça fundamental para o desenvolvimento de uma estrutura back-end. É descrito por Rod Johnson (2002), como um framework que fornece todos os recursos guiados no ecossistema do software, e é ideal para o Java Corporativo, uma das linguagens de programação principais, que possui resultados oficialmente bons, para diversos projetos de configurações e aprimoramento de sistemas, cuja a relação, é manter o máximo de dedicação ao código feito pelo programador, e prover uma experiência com uso dos dados já criados e implementá-los no aplicativo.

As ações para realizar um projeto framework Spring, pode ser considerado um ponto alto, porém exige que o ambiente esteja preparado, caso venha ser feito uma aplicação ou programa de software do zero, existe a plataforma que oferece os recursos disponíveis e é trabalhada com o chamado Spring Boot. Podendo se basear em modelos pré-prontos, a plataforma é essencial para fazer os testes e verificar se o programa está correlacionado e ativo, para que assim esteja no padrão, sem ocorrer erros na inicialização do sistema e adicionar funcionalidades dando vida ao projeto do código fonte (RODERICK, 2002).

Antes da preparação do código fonte, veremos a seguir outros recursos que são o suporte obrigatório na construção de um projeto software, e os benefícios que traz para a função back-end e o uso do banco de dados.

2.3 Rest

O rest é um modelo de arquitetura de computadores, que tem como base os serviços que são solicitados por uma API (interface de programação de aplicações). Diante dessa medida, o rest não se trata de uma linguagem ou tecnologia de programação, seus princípios passam por diretrizes e protocolos existentes na web, que puxam ter relações para se comunicar diretamente com um sistema servidor. A ligação que o rest possui com a API, é que o gerenciamento de vários projetos pode ser implementado na arquitetura do rest de forma relativa ou maneiras variadas (FIELDING, 2000).

Além do rest ser focado como serviços que atendem as demandas dos servidores, a API cumpri com o papel de transferir uma representação dos recursos ao solicitante, ou seja, o serviço que é chamado de Restful, que priorizam as demandas do cliente. Em outras ocasiões, para que haja uma conexão estabelecida, a rest por sua vez, tem a efetividade fundamental do protocolo HTTP, pelo motivo que é usado para estruturar uma comunicação entre os dados (MOCKUS; FIELDING; HERBSLEB, 2000).

Um exemplo que se pode considerar uma rest feita para um site, que trabalha com uma API alocada, contendo algumas informações associadas a um documento, cujo arquivo é mostrador como XML. Veja na figura 1.

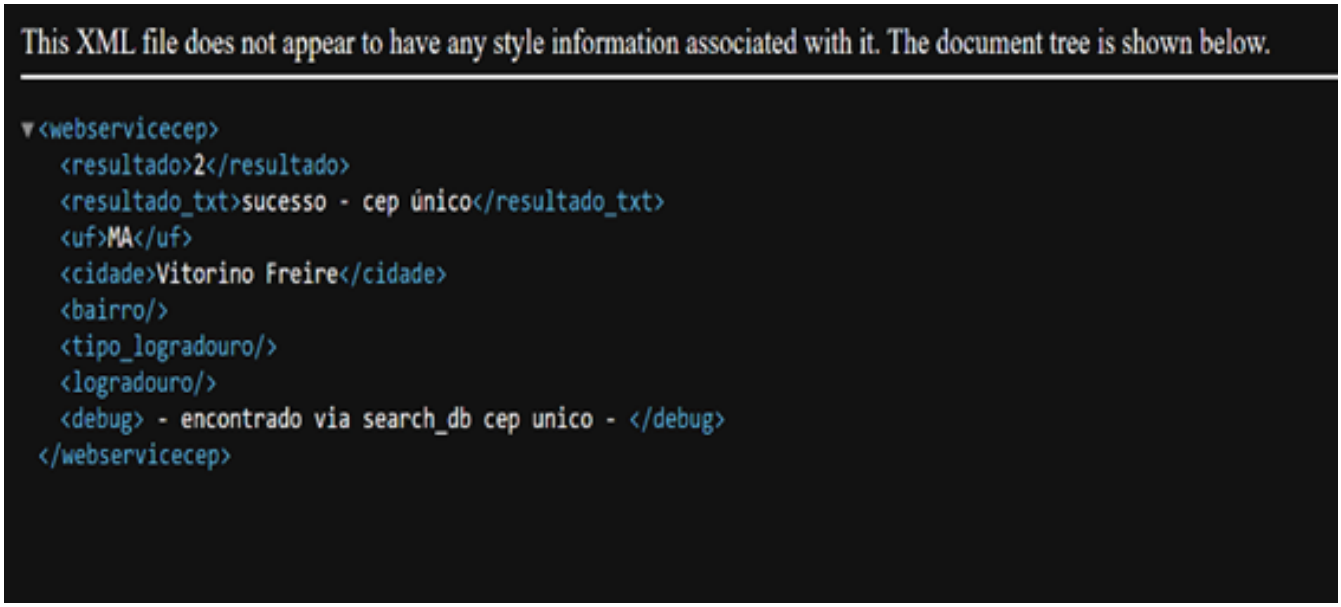


Figura 1. Exemplo API rest

Fonte: Autoral, 2022.

Na imagem acima temos alguns dados descrito em php, trata-se de um site que verifica as informações necessárias, para enviar diretamente nos correios, como os dados que incluem: cep, nome, sigla uf "MA", cidade "Vitorino Freire", bairro e tipo logradouro. Cada um dos métodos possui uma função, para ser realizada nas comunicações com a empresa de encomendas, e está associada pelo site que fornece os recursos necessário para o funcionamento da API.

2.4 API

As APIs são basicamente webs, e é definida um conjunto de padrões que faz a ligação entre os aplicativos ou vários sites, e possuem muitos benefícios que ajudam nos negócios, tanto para o proprietário quanto ao usuário, que são importantes para funcionamen-

to e testes realizados na conexão das informações estabelecidas nas APIs. Dentre seus recursos, a forma de como as APIs são utilizadas, influencia muito nos trabalhos com o Rest, pois é o sistema que gerenciar as transferências de informações e mantém a comunicação segura entre o software e a aplicação (PAUTASSO; SCHREIER 2016).

2.5 Flutter

Flutter é um framework, que tem como foco desenvolver plataformas para sistemas Android, IOS, Windows, Mac, sendo compilado e executado em Desktop, mobile e WEB. Foi criado em 2018, e mostrou-se bastante eficaz no uso da programação moderna, e vem se destacando muito com as implementações e bibliotecas desde códigos simples e outros complexos. A linguagem básica utilizada no flutter é o Dart, que foi lançada pela em 2011 e assim como o framework que são mantidas pela Google e podendo destacar várias multifuncionalidades, que se relacionam com banco dados de qualquer projeto programado (ZAMMETTI, 2020).

A função de todo projeto de programação em flutter é apresentar mecanismos que possam ajudar a facilitar problemas ou eventos recorrentes do mundo atual, isso devido a todas as tarefas que se tornam muito cansativas e acumulativas. Por isso, existe uma estrutura que todas as aplicações seguem para organizar e solucionar de forma prática e sem dificuldades, garantindo um bom aproveitamento serviços, sem ter que recorrer a medidas ineficientes para resolver erros comuns que acontecem na sociedade (MARINHO, 2020).

Muitos aplicativos que foram criados pelo flutter tem como base, trazer benefícios para quem precisa de suporte, tendo esforço para manter integridade em ambientes onde há extrema necessidade, ou seja, é um método preparado para dar total apoio ao que a população sempre obteve dificuldades. Por causo do impacto de muitas aplicações, o que mais tem de resultados são as formas estratégicas de profissionais que visão sempre entender as demandas como um ponto central das análises dos sistemas de softwares, e usam o modelo das interfaces do programa, como flutter, para prover o mínimo de recurso possível (MARINHO, 2020).

É de total entendimento que para buscar diversas ferramentas e maiores compatibilidades com o flutter, exige certos níveis e etapas que todo o sistema utiliza para aprender a desenvolver de forma conjunta e padronizada. O ambiente, a linguagem, a lógica, o algoritmo, a estrutura, e muitos outros, são em partes a peça fundamental para obter todos os recursos e a construção ideal da aplicação. Dito isso, tudo é feito por fases, seja elas trabalhosas ou não, o que vai realmente mudar é de como pode ser analisado e estudado, para que assim, seja feito o projeto e inovar mais ainda os programas softwares (ZAMMETTI, 2020).

2.6 UML

A UML idealizada em 1997 para diagramação de design de software, é uma linguagem para especificação, visualização, construção e documentação dos programas. Visto que são feitos para montar a estrutura de um projeto inicial e que são modelados por parâmetros, que fornece representação gráfica, para elementos essenciais do paradigma de objetos como: Classes, atributos, objetos, troca de mensagens, entre outros... Pode gerar uma visão intermediária entre cliente, o analista, o programador e demais envolvidos no desenvolvimento do software (BOOCH, 1994).

O propósito geral da UML, privilegia a descrição de um sistema segundo três perspectivas, os Dados (Diagrama de Classes), Operações (Diagrama de Caso de Uso) e Eventos (Diagramas de Sequência, atividades, de Transição do Estado). Cada um deles, tem uma relação de projeto diferente e um formato padrão criado a partir do proprietário, que defini os diagramas e adiciona os seus dados que vão ser ligados ao sistema do código (JACOBSON, 1992).

2.6.1 Diagrama de Classe

O diagrama de classe é fundamental, para modelos de objeto e estrutura estática de um sistema. Independente do sistema ser complexo ou simples, pode construi-lo inteiro ou vários diagramas de classe para modelar um os componentes de um sistema. Uma classe descreve um conjunto de objetos com as mesmas propriedades, o mesmo comportamento, os mesmos comportamentos, os mesmos relacionamentos com outros objetos e a mesma semântica (GUEDES, 2018).

Em um projeto de software orientado a objeto, os diagramas de classe são criados durante os estágios iniciais do projeto, contêm classes que normalmente são convertidas em na linguagem do software quando está gravado em um código. Basicamente, são feitas as análises e os modelos conceituas utilizadas no diagrama de classe que mostrem as partes específicas do sistema, interfaces com os usuários, as implementações lógicas e assim por diante (BENZERRA, 2015).

2.6.2 Diagrama de Casos de Uso

O diagrama de caso de uso, em resumo, é a relação entre os usuários e os serviços ou empresas, com a interação do sistema. É representado por uma forma oval rotulada, sendo assim, são criados bonecos de palitos que neste caso, são os atores no processo, e a participação do ator do sistema é modelada com a linha entre o ator e o caso de uso. Em geral, são muito utilizados para construir um cenário em que o sistema ou aplicativo interage com o cliente, e ajuda com as entidades (atores) a atingir o escopo do sistema (BENZERRA, 2015).

2.7 Front-End

O front-end é responsável por desenvolver por meio de código uma interface gráfica e, normalmente, com tecnologias que tem como base a Web: HTML, CSS e JavaScript. É trabalhado aplicação na interação em relação ao usuário, e por isso é importante que quem está programando para prover uma experiência para cliente, e destacar as ferramentas de linguagens de programação, principalmente para desenvolver os móbiles, que tem como função, criar apps para soluções de diversos problemas, e melhorar o sistema de operações do software para que seja realizado vários projetos para o futuro (VILARINHO, 2021).

Os softwares relacionados ao front-end, tem como finalidade possibilitar elementos de design e de interface para o cliente, que foca tanto na funcionalidade quanto na aparência. Comparado ao back-end, que prioriza os "bastidores" de um site, como protocolos de seguro, e armazenamento de dados e outras funções. Outro ponto importante que é utilizado no software do front-end é projetar e garantir manutenções na parte visual de um site ou aplicativo (VILARINHO, 2021).

2.8 Back-End

O back-end é toda a estrutura das operações que gera o funcionamento do sistema de software, visto que na própria palavra "back-end", traz a ideia daquilo que tem por trás de uma aplicação. Um exemplo que está no dia a dia, sobre a aplicação relacionada ao back end, é o Facebook que contém os dados (informações) do seu perfil, amigos e publicações que estão salvas em algum local e são processados a partir do local criado para ser chamado e armazenado em um banco de dados (SANTANA, 2021).

Um dos princípios que faz com que o back-end seja preciso para o front-end, é que tem como função, trabalhar em boa parte dos casos fazendo a ponte entre os dados que vem no navegador rumo ao banco de dados, sempre tendo que manter as regras de negócios, validações, e garantias num ambiente restrito e com segurança para o acesso comuns em sites ou aplicações utilizados pelo cliente (SANTANA, 2021).

3. RESULTADOS E DISCUSSÃO

3.1 Estudo de Caso

Através de vários estudos e treinamentos para aprender a realizar um projeto de aplicação, e desenvolver um sistema de softwares, para ajudar para as facilitar altas demandas e minimizar vários problemas que são recorrentes do dia a dia. Portanto, as próximas seções iram abordar todo o projeto realizado, desde sua estrutura até a sua aplicação real que tenha como funcionalidade todos os requisitos para os clientes e a empresas.

3.2 Fase de análise e Diagrama da aplicação

Os estudos e análises feitas para entender o funcionamento do aplicativo, tem como ponto principal, as formas e estruturas de modelos que são trabalhados na aplicação. Com base no diagrama de classe (figura 2), e o diagrama de uso (figura 3), os métodos e informações contidas em cada parte do diagrama vai ser gerenciada um sistema diferente, em que todas as funções vão ter tarefas que dependerão da relação e da comunicação entre ambas as tabelas.

Na figura 2, o diagrama de classe do projeto, aborda um modelo padrão da aplicação que apresenta os dados, que são oferecidos por meio tanto para o cliente quanto ao vendedor, sendo assim tem-se pedido, empresas parceiras, compras e pagamentos. Todos os dados, vão ser gerenciados de forma organizada e baseado no modelo do diagrama de classe, para que assim seja representado na aplicação.

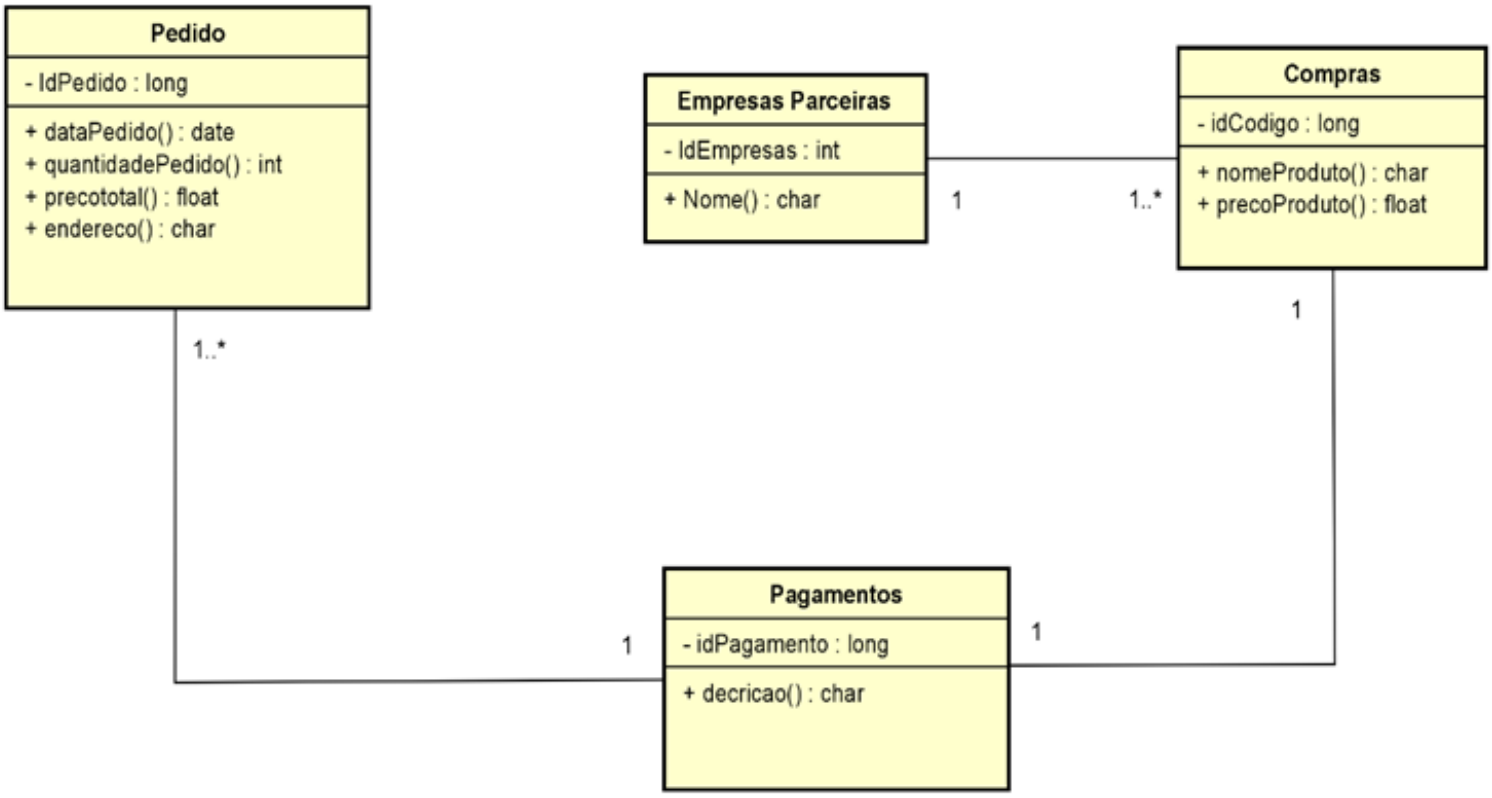


Figura 2. Diagrama de Classe

Fonte: Autoral, 2022.

Na figura 3, aborda o diagrama de caso de uso, que tem como modelo, a forma do gerenciamento da aplicação, de como vai ser realizado os procedimentos, visto que os dados apresentados estão relacionados ao cliente e o vendedor. Logo, o sistema de gerenciamento do app irá funcionar seguindo todos os métodos e etapas com base no diagrama de caso de uso, que tem como prioridade: as empresas parceiras, os pedidos, o produto e a compra.

A principal função do gerenciamento do app, é estabelecer a conexão dos dados, e passar as informações e comunicar o cliente e o vendedor sobre os procedimentos dos serviços que o aplicativo está realizando, de acordo com o que irá ser apresentado na parte do back-end e do front-end.

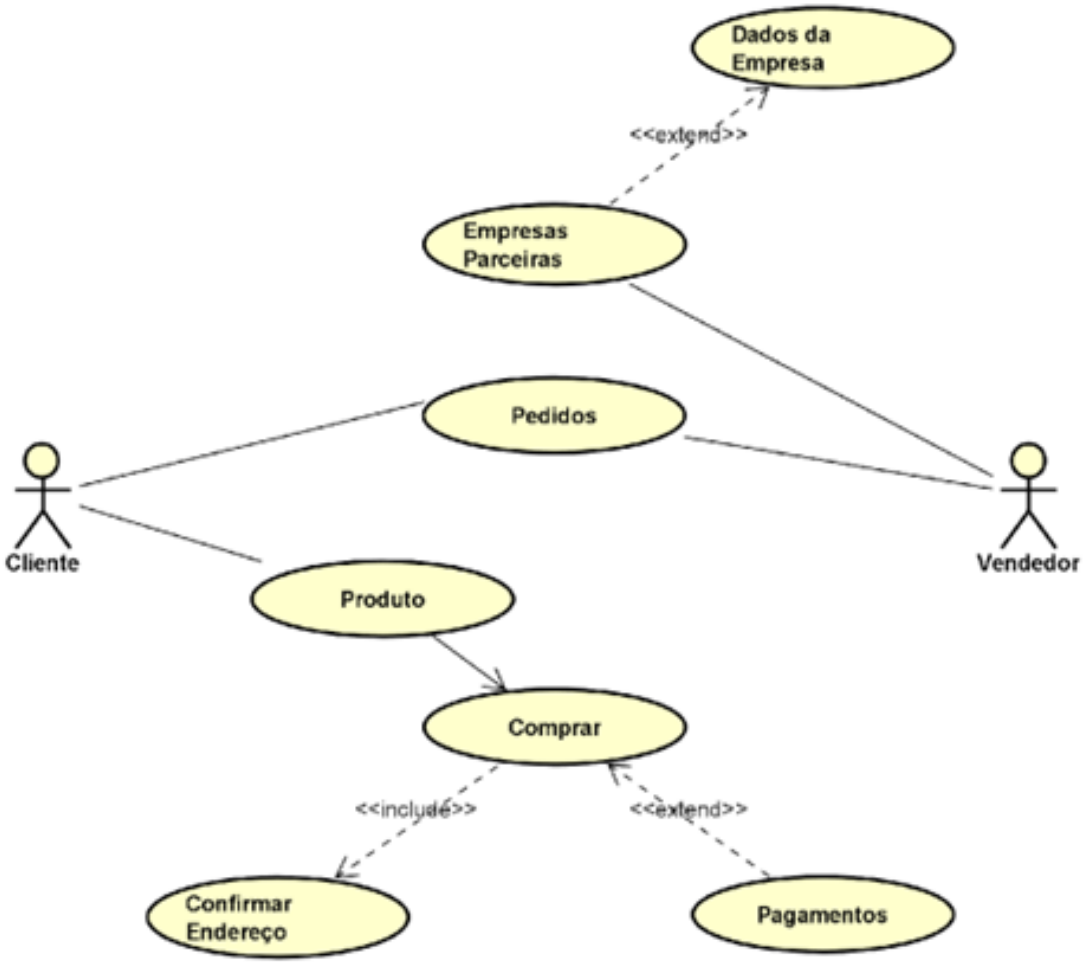


Figura 3. Diagrama de Caso de Uso

Fonte: Autoral, 2022.

Vendo os modelos acima, a estrutura é baseada na aplicação do projeto, um exemplo de como vai ser feito e relacionado para cada tabela, com suas funções, dentre elas: a tela de compras, tela de pagamentos, tela de pedidos e a tela de empresas parceiras. Com relação ao banco dados, todos eles estão sendo armazenados dentro do mysql, para ser usados dentro da API, de forma diferente, o tipo que vai ser apresentado é definido pelo Spring, e puxado do banco de dados, para gerenciar todo o sistema de compras dos produtos, pagamentos, pedidos e as empresas.

3.3 Back-end da Aplicação

O projeto teve como base o framework spring, para realizar o sistema de dados e a criação da API e também mostrar o funcionamento dos serviços que vão ser gerador e passados na aplicação. Vão ser apresentados, todo o procedimento do back-end da aplicação, desde os métodos simples até os mais complexos, todos sendo gerenciados em diferentes funções, com a interação do banco de dados que são gravados no mysql.

O primeiro gerenciamento são os produtos que vão ser solicitados na tela de compras, para o cliente é importante saber que seu produto terá um registro, de acordo com a empresa que vai atender o chamado e encaminhar diretamente os produtos. Olha a figura 3, os dados que vão ser utilizados para a tela de compras.

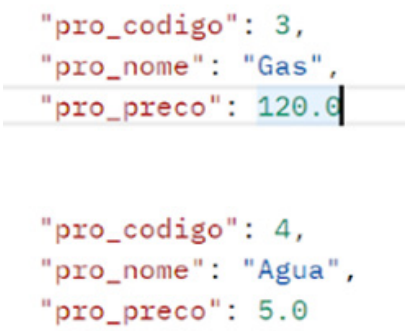


Figura 4. API da tela de compras

Fonte: Autoral, 2022.

Veja que os dados são o código do produto, o nome e o preço, que vão ser utilizados para gerenciar o sistema de compras, que busca as interações para que o cliente possa realizar o seu pedido sem nenhum erro.

O segundo gerenciamento é a tela de pagamentos que vão ser identificados, os produtos e as formas de pagamentos, dentro do que foi fornecido para que o cliente finalize sua compra, utilizando o cartão, dinheiro ou o pix, para pagar o pedido, que foi solicitado para empresa. Veja na figura 4, o sistema de pagamentos que vai ser usado para o gerenciar as compras do cliente, assim como notificar as empresas da solicitação do produto.

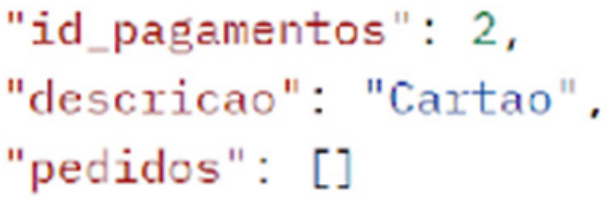


Figura 5. API da tela de pagamentos

Fonte: Autoral, 2022.

Perceba no sistema da API que para a tela de pagamentos, vai ter o id, a descrição e o pedido, onde o modelo padrão é apenas finalizar a compra dos produtos solicitados, tendo como relação a tela de compras, com os dados já obtidos para encerrar os procedimentos das duas telas listadas na aplicação.

O terceiro gerenciamento, vai se basear na tela de pedidos, onde vai ter todo o histórico da compra do produto, visto que esses dados ajudarão bastante o cliente a saber o andamento do pedido, e também algumas informações necessárias que poderão ajudar no auxílio para o cliente. Na figura 5, vai mostrar o funcionamento e a relação dos dados que serão fornecidos a tela de histórico de pedidos, sendo efetivo na segurança e em alguns mecanismos da aplicação.

```
"id_pedido": 3,
"ped_data_elaboracao": "2022-11-13T19:08:55.265621",
"ped_quantidade": 2,
"ped_preco_total": 125.0,
"endereco": "Ribamar",
"produtos": [
    {
        "pro_codigo": 1,
        "pro_nome": "Água",
        "pro_preco": 5.0
    },
    {
        "pro_codigo": 2,
        "pro_nome": "Gás",
        "pro_preco": 120.0
    }
],
"fk_pagamento": 1
```


Figura 6. API da tela de Histórico de Pedidos

Fonte: Autoral, 2022.

A tela de pedidos é uma das mais importantes, pois irá mostrar todos os dados que o cliente precisa, para acompanhar e receber com segurança o seu pedido. Os dados acima, contém informações que vai ajudar no procedimento, como: código do pedido, o nome do produto, o preço, o id do pedido, a quantidade, a data e o endereço. Todos esses recursos, garante uma segurança para usar o aplicativo sem nenhum problema.

Todos os sistemas têm como função ajudar nas configurações, e melhorar o gerenciamento das compras, pagamentos e pedidos, tudo relacionado ao cliente, pois o principal objetivo, é garantir que o app de vendas, cumpra com suas tarefas e seja feito cada um dos procedimentos, para que assim possa obter resultados positivos e benéficos não só para o cliente, mais também para as empresas que recebem pelo produto vendido no mercado.

3.4 Front-end da Aplicação

Foi desenvolvido um aplicativo de vendas de água e gás para o sistema operacional Android e IOS, e contém telas que foram criadas utilizando front-end. O flutter foi framework essencial para realizar em seis partes as telas que vão ser mostradas nas figuras abaixo, e foram configuradas pela API, funcionando sem problemas. Dos procedimentos que foram aplicados para usar no App, possui uma tela de empresas parceiras, compras, histórico de pedidos e pagamentos, sendo gerenciadas e padronizadas por ordem de forma organizada e dividido, cada uma com função diferente e contendo os recursos para serem usados no app.

A primeira etapa do aplicativo leva em consideração as empresas que são parceiras,

e estão fornecendo os serviços e incluindo novidades dos produtos, que estão sendo vendidos no aplicativo, e tendo como base as informações necessárias para a compra dos produtos, e também escolher manter contado com a empresa que distribuirá e vender conforme a solicitação feita pelo cliente. Veja a figura 6, as empresas que colaboram com o sistema do app e que atendem as demandas do cliente.

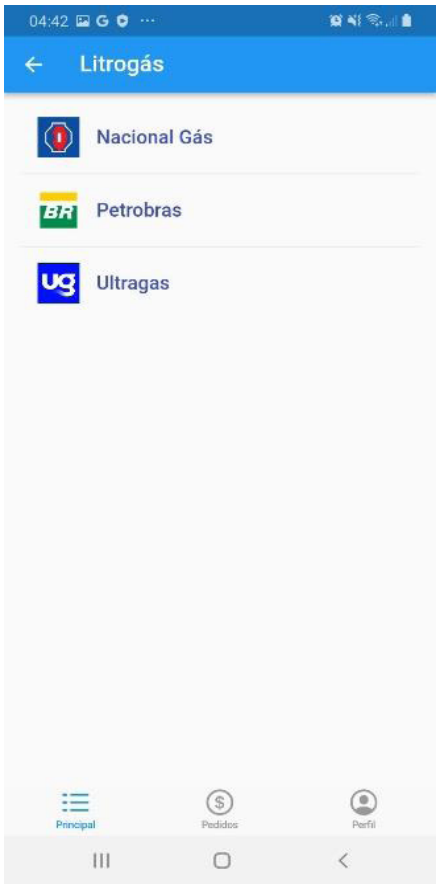


Figura 7. Tela de Empresas Parceiras

Fonte: Autoral, 2022.

Perceba que são empresas muito conhecidas, e que se destacam por ter bons negócios para atribuir e progredir na venda dos produtos de maior porte. A tela das empresas parceiras, é o ponto chave para que sejam aplicados os bons resultados e que sejam efetivos para a opção dos clientes, pois teria mais recursos e ofertas que as empresas concederiam com segurança a venda dos produtos, ou seja, teria muitos benefícios para obter no aplicativo, e melhorar ainda mais as boas avaliações com os clientes.

O que mais impacta as relações das empresas com o cliente, é justamente a qualidade e a forma organizada do ambiente de serviço, e o ótimo trabalho da equipe e cuidados com o produto, e isso gera um rendimento ainda maior para a criação da tela da figura 4, pois é preciso buscar sempre o que tem de melhor de todas as empresas, seja para alcançar pontos altos ou com lidar com alguns prejuízos, mais de forma estruturada, tendo que melhorar a cada nível e progredir para que tenha bons desfechos.

A tabela utilizada para as demandas das empresas é:

- IdEmpresasParceiras: identificação e informações.
- NomeEmpresa: nome da empresa.

A segunda etapa tem como finalidade os clientes escolherem a quantidade ou o produto desejado para realizar a compra de uma forma simples e efetiva. O carrinho de com-

pras, é o um dos principais serviços que o aplicativo vai ter que priorizar, pois o cliente está realizando um pedido, e é obrigatório ter o maior cuidado e também gerenciar com alta segurança a compra desses produtos, tendo uma relação de confiança e principalmente para a empresa que está vendendo os produtos. Veja abaixo a figura 7, a tela de compras e como é a interface e o procedimento padrão que ela vai seguir quando o cliente for fazer o pedido do produto, sendo eles água e gás.

Figura 8. Tela de Compras

Fonte: Autoral, 2022

A tela de compras tende a ser muito detalhada, visto que tem até mesmo o endereço para confirmar a localização da entrega, pois quanto mais recursos que sejam para assegurar a compra feita pelo cliente, melhor vai ser os benefícios que o aplicativo pode oferecer para as empresas, focando não só na venda no produto como também na qualidade e em requisitos altamente necessários para realizar a compra de produtos como água e gás que são obrigatórios para suprir a necessidades dos clientes no dia a dia.

A tabela feita na tela de comprar é dada por:

- IdProduto: identificação do produto, podendo ser água ou gás.
- Fk-Quantidade: o número de produtos.
- Preco: o valor do produto.
- Fk-Endereço: o local que será feito a entrega.

A terceira etapa consiste na tela de pagamentos, onde o cliente que após escolher o produto água e gás, pode realizar seu método de forma organizada, para comprar o produto, seja no dinheiro, pix ou até no cartão de crédito/débito. Veja abaixo na figura 8, modelo da interface, e através das análises, a tela de pagamentos, tem como foco os dados que foram complementados na tela de compras, sendo assim finalizando as opções

para realizar o pedido.

Figura 9. Tela de Pagamentos**.**

Fonte: Autoral, 2022.

A tela de pagamentos busca a principal etapa da aplicação, pois trata-se de uma funcionalidade que tende a ser bastante utilizada em vários outros aplicativos e em sites de lojas famosas, é importante, pois é através dos pagamentos que os clientes vão solicitar o produto e aguardar na tela de pedidos, pra saber o andamento e o histórico da compra que foi feita no aplicativo.

A tabela feita na tela de compra é dada por:

- IdPagamentos: identificação dos pagamentos.
- Descrição: que seria as identificações dos tipos de compras, podendo ser cartão, dinheiro ou pix.

A quarta etapa tem como principal foco, a listagem de todos os pedidos realizados na tela de pagamentos, aqui irá mostrar um histórico de tudo que o cliente solicitou, podendo acompanhar e ser informado pelo app. Os dados que são registrados, estão na figura 9, mostrando sua funcionalidade e tendo como foco a principal relação da tela de pagamentos, podendo levar em consideração que ambas as telas passam a comunicação até finalizar o pedido para o cliente.

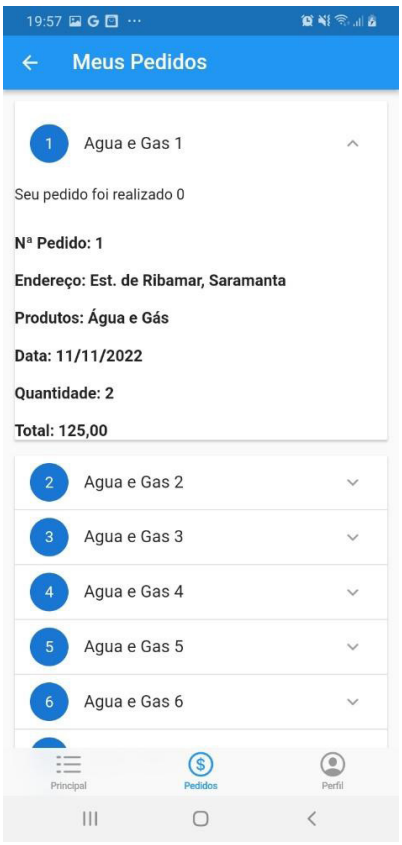


Figura 10. Tela de Histórico de Pedidos**.**

Fonte: Autoral, 2022.

A tela de histórico de pedidos é muito importante para que o cliente saiba que o produto está a caminho para o endereço que foi fornecido, e também por questão de segurança, para ficar sempre informado, e se não houve erros no procedimento, podendo ser visto como algo que afete não só as empresas, como também os clientes que pode correr riscos de perder seu pedido e não receber o produto a tempo.

A tabela feita na tela de pedidos é dada por:

- IdPedido: Número de identificação do pedido.
- NomeProduto: Identificar o produto que solicitado.
- Endereço: A localização que foi comprado.
- Data: A data do pedido.
- PrecoTotal: Quanto custou todos os produtos.
- Quantidade: Quantos produtos foram comprados.

Esses procedimentos, são todos definidos e armazenados no local onde tem o banco de dados 'Litrogas', e a API é utilizada pelo Framework Spring, onde estão alocadas para atender as necessidades e fornece os serviços e suportes ao cliente quando estiver utilizando o aplicativo.

Com base no que foi visto sobre a interface e funcionalidades do aplicativo, foi feito um estudo e uma pesquisa para acompanhar todo o progresso sobre o uso do app de vendas. O modelo que foi necessário para as pesquisas foi importante para o desenvolvimento da aplicação, principalmente na parte de design e alguns pontos que reforçam o uso das telas, para melhorar e adicionar alguns recursos que propõem ao cliente, um ambiente mais organizado e com boas estruturas para ser utilizado de forma efetiva e simples.

3.5 Pesquisa de Campo

A pesquisa sobre o uso da aplicação, foi feito através de entrevistas utilizando redes sociais, anúncios e eventos, tendo como foco principal os jovens e adultos para responder 4 perguntas e falar sobre o app, dando sugestões e melhorias sobre o que o aplicativo precisa para se torna importante para os clientes e as empresas, que vão utilizar de forma com que facilite as altas demandas de pedidos e o número grande de consumo obrigatório dos produtos água e gás.

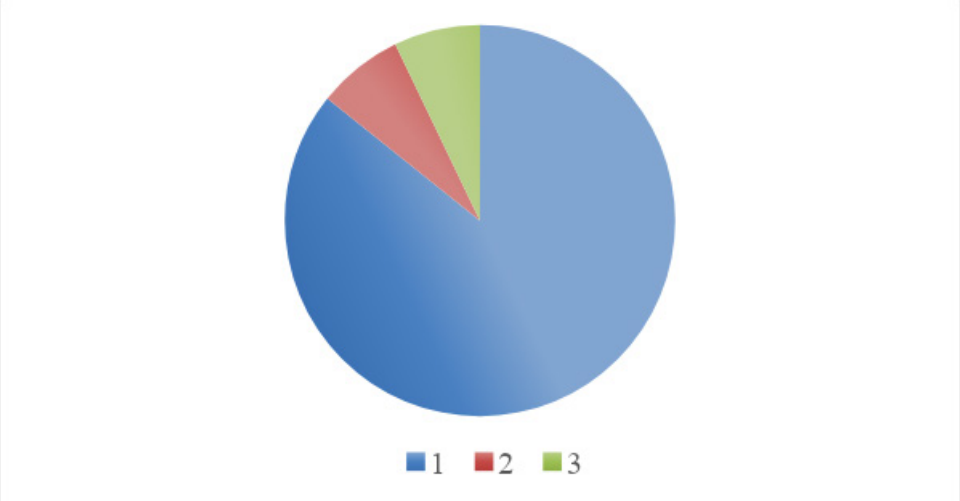
Foram entrevistados 60 clientes, utilizando um questionário para as perguntas e respostas, sobre as altas demandas e recursos que o próprio cliente que já usa para fazer os pedidos dos produtos, assim como os problemas recorrentes. As perguntas foram divididas em duas etapas, as duas primeiras sobre a situação, problemas e recursos, e as duas últimas sobre as interfaces principais, a tela de empresas parceiras e a tela de compras, e também para deixar sugestões e melhorias sobre o app.

3.6 Análise Gráfica das Entrevistas

No gráfico 1, aborda a primeira pergunta do questionário, em que indaga a forma de contato com o vendedor, **"Na compra de produtos como água e gás, quais meios você usuária para conseguir entrar com vendedor ou empresa, e esse método é sempre efeito? Como?"**.

Assim como apresentado no gráfico 1, pode-se ver que foi apresentado em porcentagem que cerca de mais de 85,7% dos entrevistados, dizem que utilizam telefones para fazer ligações ou enviar mensagens nas redes sociais e que o método é efetivo e seguro para fazer o pedido. Cerca de 14,3% dos entrevistados, dizem manter contato com algumas empresas, porém não são muito efetivos, e acham inseguro e ruim para realizar as compras.

Gráfico 1. Primeira pergunta



Fonte: Autoral, 2022.

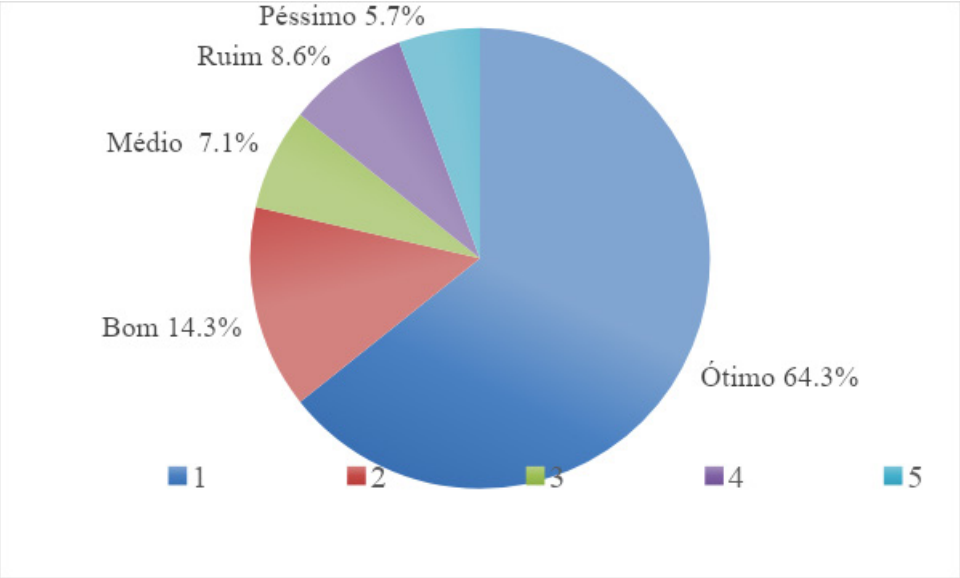
Respostas de alguns entrevistados:

1. Ligação, mensagem WhatsApp, esses métodos costumam ser bem efetivos
2. Via telefone, mais nem sempre é eficaz, pois as vezes está ocupado, e demoram atender.

No gráfico 2, aborda a segunda pergunta do questionário, que leva em consideração ao uso de aplicativos, **"O que você acharia se os vendedores recomendassem utilizar um aplicativo que contém mais recursos e informações dos produtos para realizar as compras?"**.

As informações coletadas no gráfico 2, sobre a segunda pergunta, teve opinião diversas, cerca de 78.6% dos entrevistados, disseram que é uma boa e ótima ideia para usar aplicativos para realizar os pedidos dos produtos nas empresas. Cerca de 7.1% dos entrevistados, usariam o aplicativo apenas em questão de sua necessidade ou dependendo da demanda. Cerca de 14.3% dos entrevistados, não utilizariam o aplicativo, pois mantém contato com o vender e que não há necessário para vender os produtos através do app.

Gráfico 2. Segunda Pergunta



Fonte: Autoral, 2022.

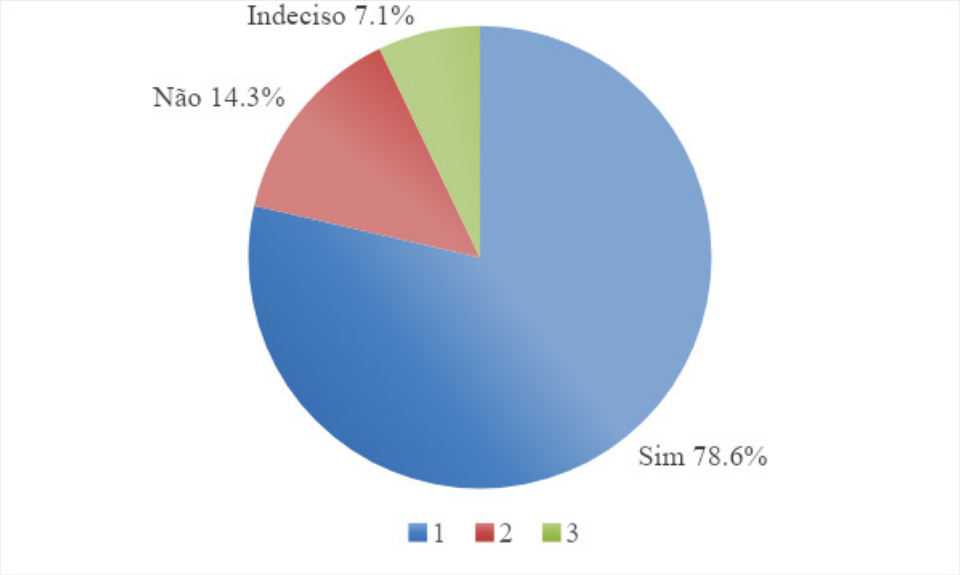
Respostas da segunda pergunta de alguns entrevistados:

1. Bom, achariam muito bom, por não ter que entrar em contato com o vendedor.
2. Sim, mas utilizaria apenas algumas vezes.
3. Ruim, pois já fiz pedidos de compras online e não tive uma boa experiência.

No gráfico 3, aborda a terceira pergunta do questionário, onde fala sobre o uso da tela de compra do aplicativo, "**Na tela de compras, possui dois produtos para a venda. Tendo todos esses requisitos, você utilizaria esse app para realizar a compra dos produtos? Dê sua opinião"**.

Olhando os dados no gráfico 3, na terceira pergunta, muitos tiveram opiniões com relação aos recursos e também a forma de como os pedidos vão ser realizado na compra, cerca de 78.6% dos entrevistados, disseram que a tela é viável e com bastante recursos para fazer os pedidos dos produtos. Cerca de 14.3%, não acharam viável, e que teria que ter de melhorar um pouco mais na interface, e com mais recursos a segurança. Cerca de 7.1%, disseram que teria que usar o aplicativo para poder saber como funciona, e realizar os testes de compras utilizando as demais ferramentas da tela de compras.

Gráfico 3. Terceira Pergunta



Fonte: Autoral, 2022.

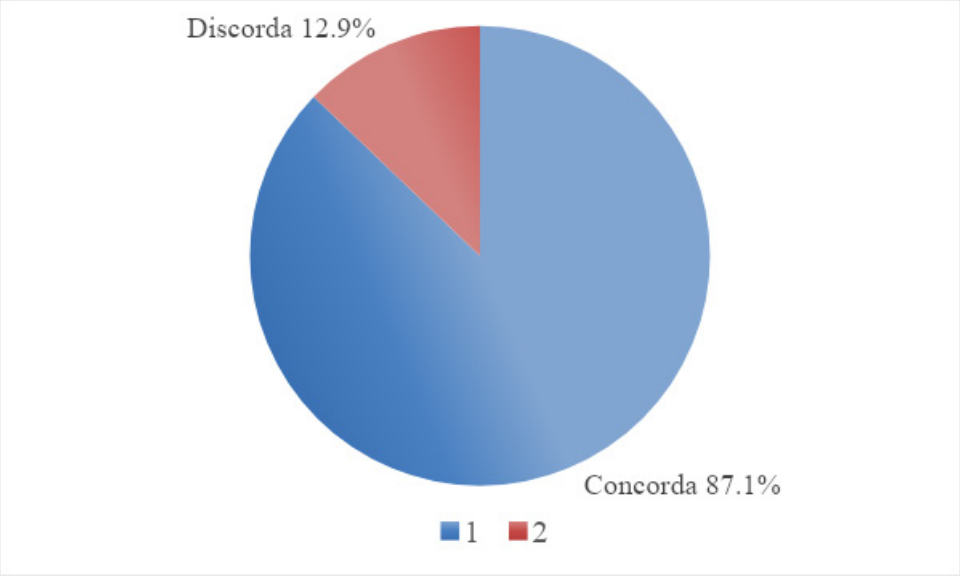
Resposta da terceira pergunta de alguns entrevistados:

1. Sim, a tela é bem simples e fácil de usar.
2. Não, pois a tela precisa de mais recursos como mostrar a avaliação do cliente e informações confiáveis da empresa para assegurar a compra dos produtos.
3. Não sei responder, pois só comprei em sites, mais experimentaria.

No gráfico 4, aborda sobre a última pergunta do questionário, onde apresenta a tela do aplicativo, que seria a interface das empresas parceiras, que relata a conexão das empresas com o cliente, "**A tela abaixo apresenta as Empresas Parceiras do aplicativo. Em sua opinião, esse seria um modelo eficiente para receber as novidades e gerar bons negócios?"**.

Os dados mostrados no gráfico 4, foram analisadas as empresas tendem a ser a peça fundamental para oferecer as novidades e ofertas do produto. Cerca de 87.1% dos entrevistados, concordam que é ótimo para as empresas gerar bons negócios, assim como receber as novidades, e apresentar serviços que são importantes para os clientes. Cerca de 12.9%, discordam, disseram que não haveria necessidade de receber novidades ou ofertas, sendo que seria o suficiente apenas para as funcionalidades do aplicativo, visto a necessidade do cliente que irá utilizar no aplicativo.

Gráfico 4. Quarta pergunta



Fonte: Autoral, 2022.

Respostas da quarta perguntas de alguns entrevistados:

1. Sim, beneficiaria não só as empresas, como também notificaria os clientes de receberam ofertas e novidades dos produtos, que ajudaria até mesmo ao realizar a compra em diferentes locais ou em outras empresas dependendo do custo e da qualidade do produto.
2. Discordo, não vejo necessidade de receber ofertas e nenhuma notificação.

Ao final de todos os dados e informações coletadas, vários entrevistados sugeriram algumas melhorias para o aplicativo, como: no design, colocar marcas dos produtos, informar os pagamentos, conhecer mais sobre app, ver a taxa de entrega, divulgação, comunicação, entre vários outros requisitos do aplicativo.

4. CONCLUSÃO

Tudo que foi estudado e trabalhado durante todo projeto, precisa de várias melhorias, para que assim, a aplicação ofereça maior recurso, e que seja mais eficiente, para que os clientes fiquem seguros de usar a aplicação. Tendo todas as funcionalidades, o projeto mostrou-se que pode obter boas soluções, a fim de facilitar as altas demandas dos produtos citados na aplicação, visto que para a maioria dos clientes e empresas, tem a necessidade dos aplicativos na atualidade, como benefícios para diversos serviços e oferecer novos mecanismos que ajudam a suprir a necessidade dos produtos.

Além de propor benefícios para os serviços das empresas, o aplicativo também mostra de forma organizada, um ambiente limpo para que possam ser adicionados mais ferramentas no futuro e aprimorando ainda mais as funções do aplicativo. Logo, baseando no que foi feito nas entrevistas, as sugestões foram muito importantes para desenvolver as tarefas e as informações sobre as vendas dos produtos água e gás, vendo que seria necessário algum método efetivo, que tornariam app de vendas como uma solução para melhorar o sistema de gerenciamento de pedidos.

Dentre as altas demandas e os trabalhos corridos, que as empresas e até mesmo os clientes têm recorrentemente nos dias atuais, o app pode gerenciar um sistema de controle para manter as demandas organizadas, e prover sempre novidades, e recursos para melhorar não só o sistema, mais também inovar o que mais pode impactar no mercado,

no uso da tecnologia para mudar e aumentar boas relações e comunicação entre as empresas e os clientes.

Referências

BEZERRA, EDUARDO. (2015) **Princípio de Análise e Projeto de Sistemas com UML**. Rio de Janeiro: Campus

BOOCH, GRADY; (1994) **Object-Oriented Analysis and Design with Applications** 3. ed. Addison Wesley New York

FIELDING, ROY THOMAS (2000). **Architectural Styles and Design of Network-based Software Architectures** Dissertation. University of California, Irvine

GIMENES, I.M. S; HUZITA, E. H. M. (2005) **Desenvolvimento Baseado em Componentes**. Rio de Janeiro: Ciência Moderna.

GUEDES, GILLEANES T.A. UML2: **Uma Abordagem Prática.** 2 ed. São Paulo: NOVATEC, 2018.

JACOBSON, IVAR (1992) **Object-Oriented Software Engineering** 1. ed. Addison Wesley New York

JOHNSON, RODERICK; HOELLER, JUERGEN; ARENDSEN, ALEF; RISBERG, THOMAS; SAMPALEANU, COLIN; (2002) **Desenvolvimento Java Profissional com Spring Framework**

MARINHO, LEONARDO H. (2020) **Iniciando com Flutter Framework: Desenvolva aplicações móveis no Dart Side** 1. ed. Casa do Código

MOCKUS, A.; FIELDING, R. T.; HERBSLEB, J. (2000) **A Case Stuty of Open Source Software development**. Proceeding os 22nd international conference on Software engineering – ICSE

PAUTASSO, C; IVANCHIKJ, A; SCHREIER, S. (2016) **A pattern language for RESTful, conversations. In Proceedings of the 21st European Conference on Pattern Languages os Programs** – EuroPlop ’16, ACM Press.

PRESSMAN, Roger S; LOWE, (2009) David. **Engenharia Web**. Tradução de Daniel Vieira. Rio de Janeiro: LTC.

PRESSMAN, Roger. S. (2011) **Engenharia de Software**. Tradução de Ariovaldo Griesi e Mario Moro Fecchio. 7. Ed. Porto Alegre: AMGH.

ROCHA, H. V; BARANAUKAS, M. C. C. (2003) **Design e Avaliação de Interfaces Humano-Computador**. Campinas: NIED/UNICAMP.

SANTANA, EDUARDO F. Z. (2021) **Back-end Java: Microsserviços, Spring Boot e Kubernetes** 1. ed. Casa do Código

SOMMERVILLE, Ian. (2007) **Engenharia de software**. Tradução de Selma Shin Shimizu Melnikoff, Reginaldo Arakaki e Edílson de Andrade Barbosa. 8. ed. São Paulo: Pearson.

VILARINHO, LEONARDO. (2021) **Front-end com Vue.js: Da teoria à prática sem complicações** 1. ed. Casa do Código

ZAMMETTI, FRANK (2020) **Flutter na Prática: Melhore seu Desenvolvimento Mobile com SDK Open Source Mais Recente do Google** 1. ed. Novatec Editora~

9

O USO DO FRAMEWORK MAKER PARA O DESENVOLVIMENTO DE SISTEMA BASEADO EM JAVA, QUE AMPARA INSTITUIÇÕES ENCAMINHAREM ALUNOS E PROFESSORES PARA TRATAMENTOS PSICOPEDAGOGO

THE USE OF FRAMEWORK MAKER FOR THE DEVELOPMENT OF A JAVA-BASED SYSTEM, WHICH SUPPORTS INSTITUTIONS TO REFER STUDENTS AND TEACHERS TO PSYCHOPEDAGOGICAL TREATMENTS

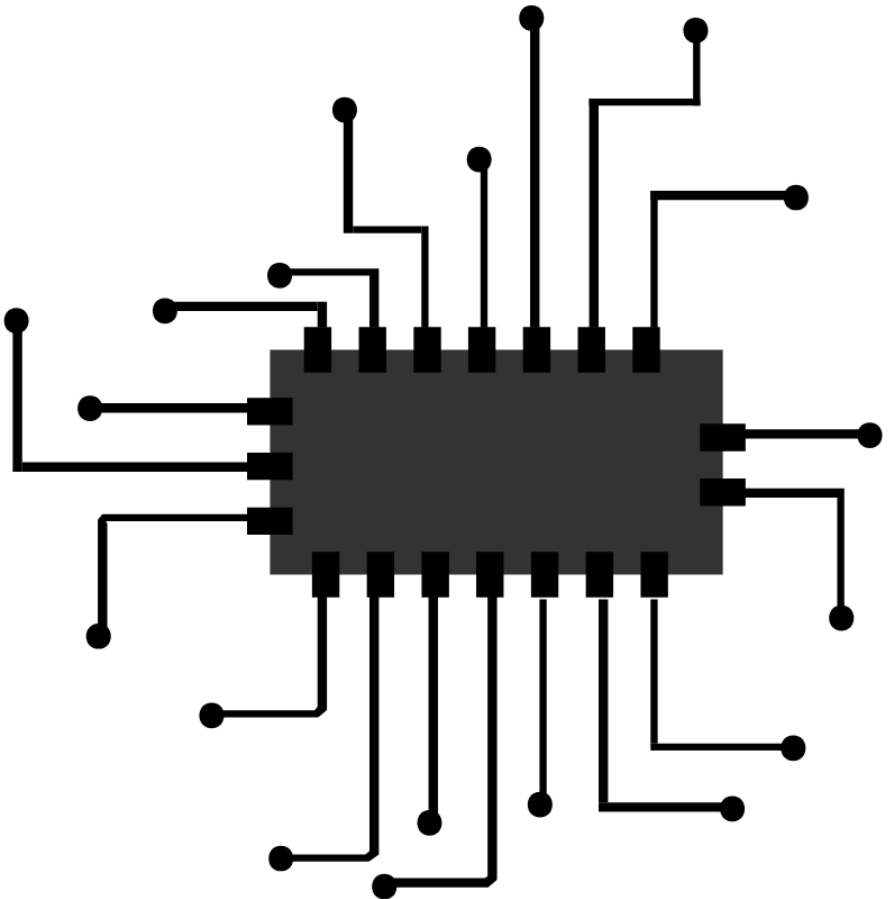
Gustavo dos Santos Cidreira

Edilson Carlos Silva Lima

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

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

{CIDREIRA, Gustavo, gustavo.cidreira2@hotmail.com; LIMA, Edilson Carlos Silva, edilsonlima3@gmail.com.}



d.o.i.: 10.29327/5192626.1-9

Resumo

O trabalho apresentado, tem como objetivo de descrever um sistema desenvolvido em web utilizando programação orientada a objeto, MVC e *framework* Maker, para facilitar o encaminhamento pedagógico e psicossocial de instituições a suas devidas clínicas credenciadas na instituição. Assim desenvolvendo um sistema que facilita a comunicação de informações importantes e formalizadas de seus alunos, organizando e gerenciando esses dados para enviar documentações necessárias para o atendimento da clínica.

Palavras-chave: POO, Maker, MVC.

Abstract

The work presented aims to describe a system developed on the web using object-oriented programming, Design Patterns, MVC and Maker framework, to facilitate the pedagogical and psychosocial referral of institutions to their appropriate clinics accredited in the institution. Thus developing a system that facilitates the communication of important and formalized information from its students, organizing and managing this data to send the necessary documentation for the service of the clinic.

Keywords: OOP, Maker, MVC.

1. INTRODUÇÃO

A instituição tinha dificuldade de encaminhar alunos e professores para as clínicas conveniadas. Além disso, havia a grande quantidade de perda de dados de alunos e professores que solicitavam este encaminhamento. De ante disso, a instituição teve a necessidade de ter um site para gerenciar alunos e professores e controlar os encaminhamentos para as empresas conveniadas.

Para o desenvolvimento do site foi utilizado a linguagem de programação orientado a objeto JAVA, foi utilizado também o *framework Maker*, para a construção do site e o backend.

O objetivo desse trabalho foi desenvolver uma solução para o atendimento psicopedagógico de uma instituição, para encaminhar os alunos e professores a clínicas conveniadas de forma rápida e organizada.

Foi construindo um sistema capaz de cadastrar alunos e professores utilizando dados necessários e obrigatórios para encaminhar a clínicas conveniadas, obtendo um melhor gerenciamento desses dados.

O estudo foi realizado diretamente em uma instituição no setor pedagógico, que necessitava de um aprimoramento, foi feito um levantamento para avaliar necessidades encontradas na instituição, foram realizadas reuniões necessárias para entender o problema deste setor, assim desenvolvendo uma solução, após coletar informações suficientes da coordenação de pedagogia , foi descoberto uma deficiências de demandas enviadas a clínicas conveniadas a instituição, com isso foi desenvolvido uma solução sistemática para sanar o problema do setor, estes resultados foram desenvolvidos.

Nesse artigo iremos abordar alguns tópicos importantes, como: o capítulo 2 mostra a fundamentação teórica, no capítulo 3 mostra o desenvolvimento do site para resolução do problema, no capítulo 4 é sobre a conclusão e o trabalho, o capítulo 5 encontra-se as referências utilizadas nesse artigo.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo vamos abordar uma revisão literário dos assuntos essenciais para o desenvolvimento de um site e um *backend* utilizando o *Maker* e está desposto da seguinte maneira: no tópico 2.1 Análise, no item 2.2 UML, 2.3 Metodologia Ágil - *Scrum*, 2.4 *Framework Maker*, 2.5 MVC.

2.1 Análise

A análise de um sistema, é algo essencial para realizar um planejamento de sistema, pois é onde se encontra toda as lógicas positivas e negativas do produto, sem ocorrer um desperdício de recurso. Essa análise é um suporte para as necessidades da empresa, a parte em que vai ser projetado o sistema, construído e entregue ao usuário.

> Isso provavelmente parecerá muito simples. Entretanto, no mundo real, isso não é tão fácil. Em 2010, estima-se que foram gastos US$ 2,4 trilhões em hardware, software e serviços de TI, pelas organizações e governos, em todo o mundo. Projetou-se para 2011 aumento desse nível de gasto em 3,5%.

> 1 Infelizmente, um estudo realizado em 2008 encontrou que o sucesso é "improvável" em 68% dos projetos de tecnologia. Atualmente, tanto as empresas como os governos enfrentam erros embaraçosos e onerosos em seus sistemas de informações. Eis um exemplo de apenas algumas poucas falhas notáveis de software que ocorreram em 2010 (DENNIS, 2014).

Se tornando essencial para um desenvolvimento de software, é uma das partes mais importantes para esse desenvolvimento, pois é onde obtém as informações necessárias para desenvolvimento do software.

2.1.1 Orientação a Objeto

Muitos profissionais consideram um tanto complexo o conceito de paradigma de orientação a objetos. No entanto, esse conceito é apenas diferente do enfoque procedural ao qual estão acostumados. Na realidade o ser humano no início de sua infância, aprende a pensar de uma maneira orientada a objeto, representando seu conhecimento por meio de abstrações e classificações (na verdade continuamos fazendo isso mesmo quando adulto, mas desenvolvemos outras técnicas que também utilizamos em paralelo) (GUEDES, 2011).

2.1.2 Abstração

A ideia geral de abstração em computação não é diferente das abstrações que usamos diariamente em línguas naturais. Por exemplo, a palavra cachorro denota um fenômeno que tem características de gato. Se alguém disser, "veja o gato", usará uma abstração de gato (ou seja, a palavra gato) para indicar uma instância concreta de um animal com características semelhantes às de um gato. Ao usar as palavras gato e coelho, podemos diferenciar um de outro e representar vários tipos diferentes de gatos e coelhos. No entanto, se usarmos a palavra "gato" para indicarmos um coelho, é provável que sejamos corrigidos. "Não é um gato. É um coelho". Desse modo, embora sejam genéricos, as abstrações são especificas o suficiente para que possamos diferenciar instancias especificas (SANDERS, 2013).

Booch (2006) apresenta uma definição clara de abstração, que a sintética de modo bastante adequado:

> "Uma abstração denota as características essenciais de um objeto que o distingue de todos os demais tipos de objetos e, desse modo, proporciona limites conceituais claramente definidos em relação à perspectiva de quem visualiza."

A abstração é importante porque permite aos programadores agrupar e classificar objetivos. Em certa medida, todas as classes são abstrações de um conjunto de operações sobre dados. Tenha a seguinte ideia em mente, no que diz respeito às abstrações:

> A abstração é a ferramenta principal para lidar com a complexidade. Quanto mais complexo for um problema, mais abstrações serão necessárias para solucioná-lo.

Pense no paradoxo a seguir: abstrações são métodos concretos para lidar com a complexidade. Nós agrupamos semelhanças no mundo real (abstraímos semelhanças concretas) para que possamos lidar mais facilmente com elas. Desse modo, em vez de dizer "meu amigo leal, corajoso, peludo, que abana a calda, lambe meu rosto, tem nariz úmido e que se chama. ErroDeSintaxe", posso dizer "meu cachorro".

2.1.3 Encapsulamento

Quando ler a respeito do *encapsulamento*, com frequência você se deparará com a expressão *ocultar informações*. A expressão não é imprecisa, mas fará mais sentido quando você entender o que é encapsulamento; começar com o conceito de ocultar nem sempre ajuda a explicar o encapsulamento. Em vez disso, um ponto de partida mais adequado está na imagem de *compartimentos.* Booch (2006) fornece a seguinte descrição:

> Encapsulamento corresponde ao processo de compartimentar os elementos de uma abstração, os quais constituem sua estrutura e seu comportamento; o encapsulamento serve para separar a interface contratual de uma abstração de sua implementação.

Após um problema grande e complexo ter sido modularizado, transformando-o em subproblemas solucionáveis, o encapsulamento consiste em uma maneira de tomar as abstrações menores e compartimentá-las.

Em programação, encapsulamento é o que faz com que um objeto seja *um objeto.* Um objeto possui determinadas características que o cercam, de modo que o acesso à sua funcionalidade é controlado pela estrutura do programa – da mesma maneira que a estrutura de seu carro controla o acesso às várias partes do mesmo. Uma classe é encapsulada por meio de acesso limitado a seus métodos e às suas propriedades. Você não vai querer que influências externas assumam o controle das propriedades das classes, usando-as ou alterando seu estado, a não ser que seja por meio de caminhos específicos – assim como você não vai querer que alguém assuma o controle da direção do seu carro. Portanto, quando você encontrar a expressão *ocultar informações* no contexto de encapsulamento, isso significa que os detalhes de um módulo poderão estar ocultos para que o módulo possa ser usado por meio de canais apropriados de acesso, e não por meio dos detalhes do módulo.

2.1.4 Getters e setters

Para preservar o encapsulamento, ao mesmo tempo em que provê acessibilidade, os projetos com POO sugerem o uso de *getters* e *setters* (também chamados de *acessors* e *mutators*, respectivamente) (SANDERS, 2013). Em vez de acessar uma classe diretamente e obter ou alterar valores de uma propriedade por meio de atribuição direta, a função getter/setter faz isso para você. Em geral, o uso de *getters* e *setters* deve ser feito de modo criterioso; o excesso pode violar o encapsulamento.

As funções getter/setter são privados, de modo que o acesso é encapsulado. Além disso, nessa implementação o valor de setter está dentro da classe, de modo que ele funciona como um grande armazenador de dados.

Ao lidar com dados em sistemas orientados a objetos, Allen Holub, em Holub on Patterns (2003), sugere o seguinte:

- Não peça as informações de que você necessita para executar a tarefa;
- Peça ao objeto que tem as informações para que faça a tarefa para você.

2.1.5 Herança

A herança, em seu nível mais básico, é um conceito simples. Uma classe que estender outra classe terá todas as propriedades e métodos de outra classe. Isso permite aos desenvolvedores criar novas classes que estendam a funcionalidade de outras classes (SOMMERVILLE, 2003).

2.1.6 Polimorfismo

Basicamente, polimorfismo refere-se a várias formas, mas isso não é muito útil, a menos que você o entenda no contexto da POO. O verdadeiro valor do polimorfismo está no fato de que objetos com a mesma interface podem ser chamados a realizar diferentes tarefas (SOMMERVILLE, 2003). Em uma estrutura maior e mais complexa (um programa verdadeiramente de grande porte), adições e alterações podem desestruturar seu programa, a menos que ele tenha uma interface comum (de uma classe pai ou de uma interface).

2.2 UML

A UML é uma linguagem flexível e extensível, independente de processos ou linguagens de programação, garante a liberdade do desenvolvedor adotar qualquer processo, metodologia ou linguagem de programação, sem expressar-se claramente para seus usuários e outros desenvolvedores, utilizando uma notação padrão comum em qualquer ambiente de empresa, tendo suporte e alcance mundial, sendo uma linguagem completa e cheia de recursos permitindo a captura de informações e expressá-las com sintaxe clara e objetiva (LIMA,2018).

Essa linguagem é atualmente a linguagem-padrão de modelagem adotada internacionalmente pela indústria de engenharia de software. Uma linguagem que auxilia os engenheiros de software a definirem as características do sistema, como seus requisitos, seu comportamento, sua estrutura lógica, a dinâmica de seus processos e até mesmo suas necessidades físicas em relação ao equipamento sobre o qual o sistema deverá ser implantado. (GUEDES, 2018).

A linguagem UML (*Unified Modeling* Language, em português Linguagem de Modelagem Unificada) é uma linguagem que é expressado diagramas que definem as realizações de cada ação utilizada pelo usuário dentro do sistema, especificando o detalhe da estrutura do sistema, e os detalhes do comportamento do sistema, tornando visualmente um sistema que ainda não foi projetado mais apresentável para o programador realizar as aplicações.

2.2.1 Diagrama de caso de uso

Costuma ser utilizado principalmente nas fases de elicitação e análise de requisitos do sistema, embora venha a ser consultado durante todo o processo de modelagem e possa servir de base para diversos outros diagramas. Procura apresentar uma linguagem simples e de fácil compreensão para que os usuários possam ter uma ideia geral de como o sistema vai se comportar. Procura identificar os atores (usuários, outros sistemas ou até mesmo algum hardware especial) que utilizarão, de alguma forma, o software, bem como os serviços, ou seja, as funcionalidades que o sistema disponibilizará a esses atores, conhecidas nesse diagrama como casos de uso (GUEDES, 2018).

O diagrama de uso é onde se encontra a funcionalidade do ambiente que irá ser projetado para o usuário, é uma ferramenta que ajuda no levamento de requisito do sistema, descrevendo as principais interações dos usuários com o ambiente irá ser programado usando conjuntos de símbolos e conectores, assim ajudara a equipe em discursões e melhorias de seu produto.

2.2.2 Diagrama de classe

Seu principal enfoque está em permitir a visualização das classes que comporão o sistema com seus respectivos atributos e métodos, bem como em demonstrar como as classes do diagrama se relacionam, complementam e transmitem informações entre si. Esse diagrama apresenta uma visão estática de como as classes estão organizadas, preocupando-se em como definir a estrutura lógica delas (GUEDES, 2018).

O diagrama de classes é o esquema de como se comporta o sistema, serve para modelar os objetos que compõem o sistema, descrevendo para que servem esses objetos e quais ações esses eles realizam dentro do sistema, são muito utilizados na documentação da arquitetura do software. Sendo assim o diagrama de classes é o mais importante e utilizado na UML.

2.3 Metodologia Ágil - Scrum

O Scrum. Trata-se de uma mudança radical em relação às metodologias prescritivas e hierarquizadas empregadas no passado no gerenciamento de projetos. Ao contrário delas, o Scrum se assemelha a sistemas evolucionários, adaptativos e autocorretivos. Desde sua concepção, a estrutura do Scrum se tornou a maneira como o setor de tecnologia cria novos softwares e produtos (SUTHERLAND, 2019).

A metodologia scrum trata-se de uma aplicação no gerenciamento de tempo que permite organizar e agilizar o processo do desenvolvimento da produção, de maneira que o produto final seja entregue ao usuário com excelência, otimizando dinamicamente a execução de projetos complexos com melhor aproveitamento dos recursos com o tempo adequado para a finalização da tarefa.

2.4 *Framework Maker*

O *framework Maker*, é uma ferramenta baseada em java , que facilita o desenvolvimento do programador, aumentando sua produtividade em realizações de projetos com mais rapidez por conta da ferramenta facilitar a criação de telas de forma no-code, sem

precisar escrever uma linha de código, pois a ferramenta já faz isso pra você, e se ocorrer alguma implementação que o framework não existe, pode está sendo implementada pelo programador utilizando linhas de códigos, sendo assim se tornando bem útil para produções mais rapidez e de ótima manutenção por ser baseada em java uma linguagem simples para qualquer programador entender e realizar as devidas correções ou adicionar algo novo ao ambiente. Sendo assim uma plataforma mundialmente conhecida mundialmente para o desenvolvimento de softwares e aplicações corporativas obtendo uma alta facilidade de programação e edição de fluxogramas para criações de soluções web.

2.5 MVC

O Padrão MVC (Model-View-Control) ele realizar a separarão de logística, o controle e a visualização da tela, sendo a view responsável pelo a chamada do usuário e seus dados utilizados na tela, o controle são as classes utilizadas, e serão utilizadas como controle, e parte da view sendo a mais importante contendo seus códigos, e a porte model, onde seus modelos iram ficar separados para que outras aplicações utilizarem (SHVETS, 2021).

O MVC (Model-View-Controller) que significa a trindade entre modelo, visão e controlador é responsável pela comunicação, realizando a troca de informação do banco de dados e usuário, de forma bem ágil e dinâmica, sendo um dos padrões mais utilizados pelos os desenvolvedores, equilibrando os recursos e distribuindo resultados sem sobrecarregamentos, assim os recursos iram ser bem distribuídos pela arquitetura do sistema, otimizando o tempo de resposta com o banco de dados e ações do usuário.

3. ESTUDO DE CASO

Neste capitulo, vai ser citado como foi o desenvolvimento utilizando cada tecnologia, para o resolver o problema, os tópicos abordados são:

3.1 A metodologia

A instituição que trabalha com setor pedagógico, oferecem um acompanhamento de alunos e professores de forma individual e coletiva para suas melhorias em atividades acadêmicas e profissionais, existem demandas especiais que necessitavam ser atendidas de forma emergencial.

Com isso foi encontrado a problemática do setor, que ocorria uma grande demora desses alunos e professores serem cadastrados pelas clínicas conveniadas.

Então foi desenvolvido a implementação do sistema baseado nesses fatores, foi realizado uma reunião com o líder do setor da pedagogia da instituição, para demonstrar como eles realizavam o procedimento atualmente, assim foi feito um levantamento de requisitos, para achar a melhor forma de organizar e enviar os dados necessários para o encaminhamento do aluno ou professor.

3.2 Análise do Sistema

Este sistema, foi desenvolvido para solucionar a problemática inicial de um setor institucional, que possuem dificuldades de realizar envio de documentação, pois ocorriam perda de dados, ocasionando muito retornos e atrapalhando a produtividade do atendimento, neste sistema foi desenvolvido uma forma de resolver este problema.

Com isso, foi utilizado a metodologia scrum como mostra a figura 1, para desenvolver o sistema de maneira ágil e atendendo todas as necessidades possíveis da instituição, o scrum master (autor do artigo) desenvolveu orientações para a criação deste sistema, como: separar e ordenar as etapas e finalizações do problema, e foi realizada sprints para modelagem do projeto.

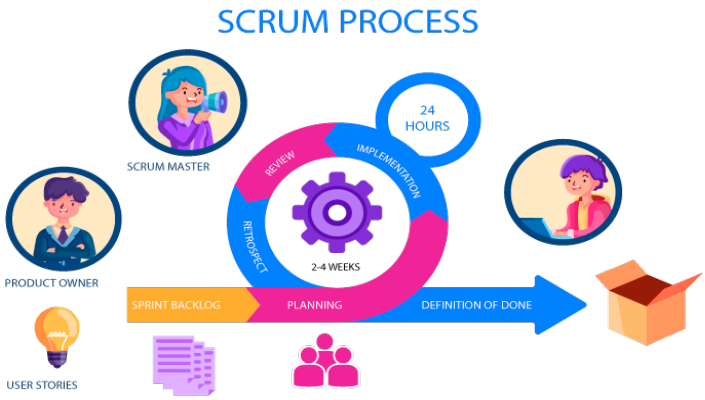


Figura 1. Processo Scrum.

Fonte: Freepik, 2020.

Como observamos na figura 1, foi realizado as orientações estudada para ser entrada para um sprint, ocorreu um planejamento para o alinhamento das variáveis do problema, foi realizado os diagramas e revisados para ser entregue para segunda sprint de desenvolvimento do produto.

O projeto foi criado para resolver o problema do encaminhamento dos dados do setor da instituição, utilizando campos obrigatórios para inserir todos os dados suficientes para o envio, sem perder dados e obtendo uma melhor organização, evitando o retorno do atendimento.

Na primeira sprint foi desenvolvido diagramas para realizar o desenvolvimento do sistema e do banco de dados. Como mostra a figura 2, o diagrama de caso de uso.

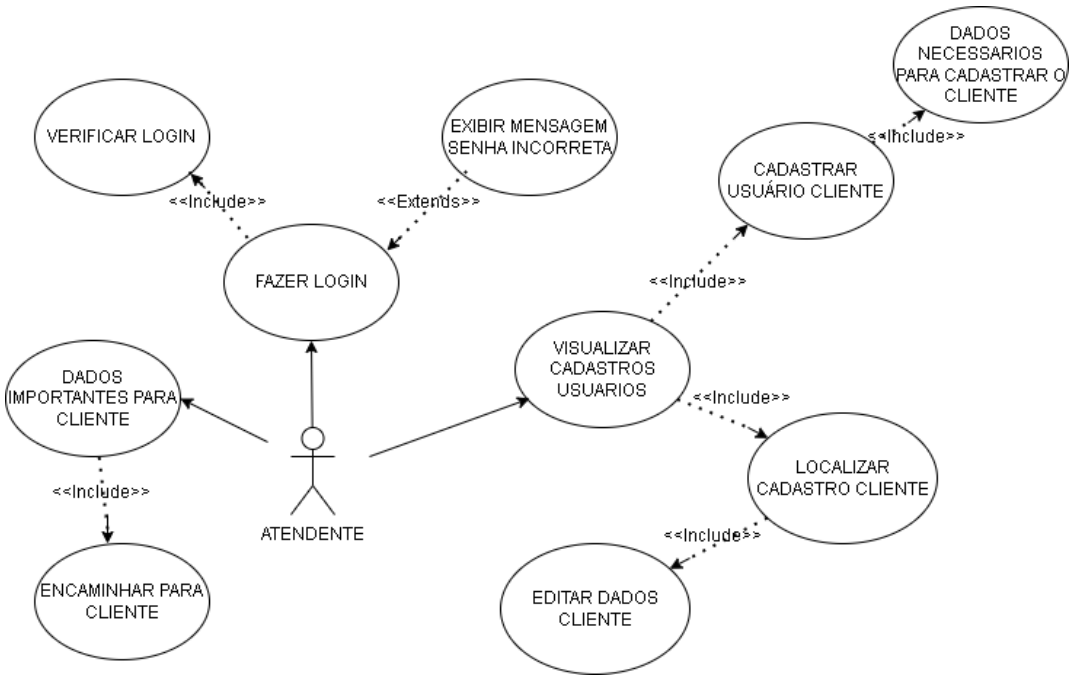


Figura 2. Diagrama de Caso de Uso.

Fonte: Autoral, 2022.

A figura 2, mostra como irá ser realizado a usabilidade do usuário atendente, que irá fazer os procedimentos, como acessar o login, se estiver autenticado corretamente, irá acessar o ambiente do sistema, tendo a opção de visualizar usuários, cadastrar o usuário com os dados obrigatórios utilizados no campo, pode localizar e editar o cliente, e pode encaminhar esses dados para clínica (cliente).

Já na figura 3, mostra o diagrama de classes desenvolvido para modelar os objetos do sistema.

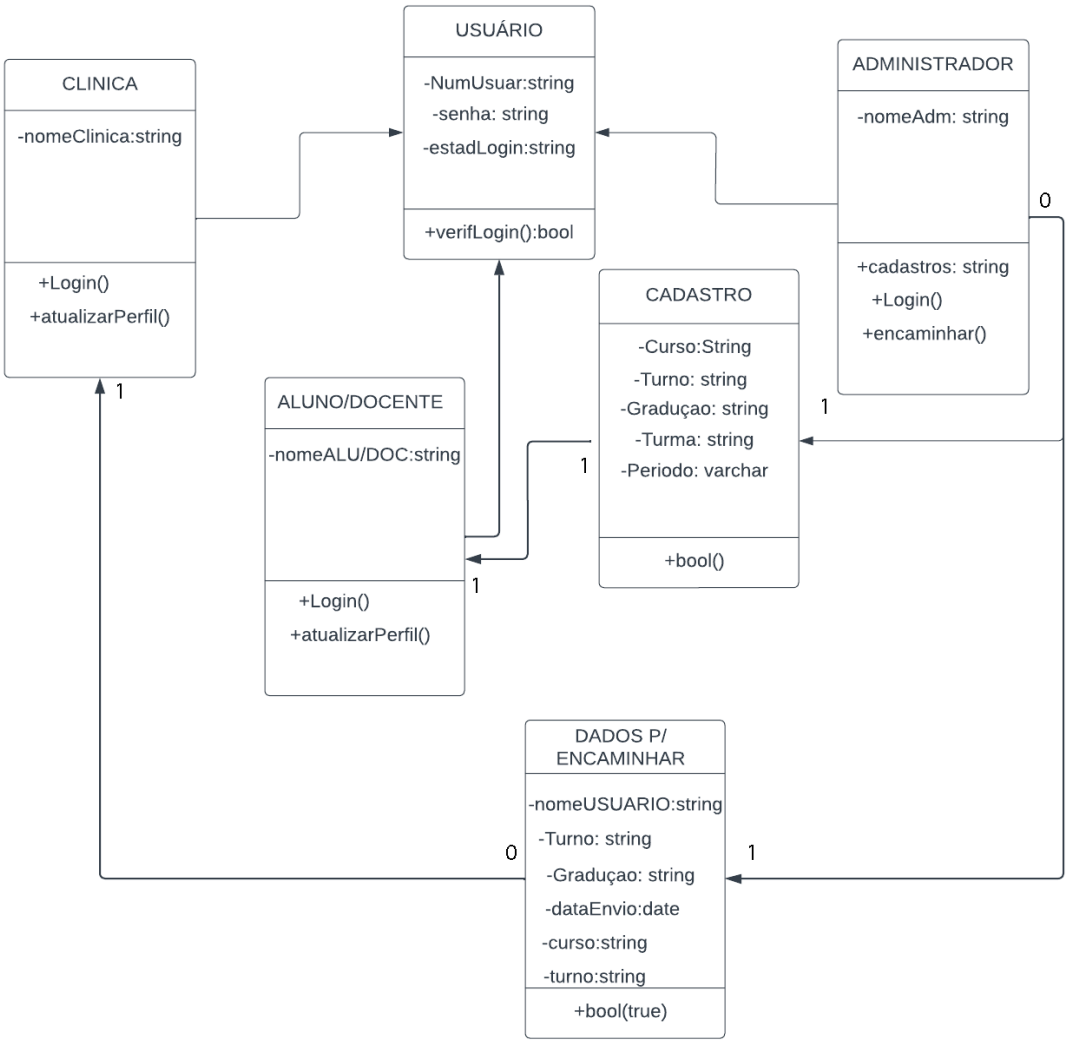


Figura 3. Diagrama de classes.

Fonte: Autoral, 2022.

Os diagramas são muito importantes, para entender de forma geral de como vai funcionar o sistema, e suas funcionalidades e ações feitas por usuários, entendendo melhor sua aplicação para uma melhor implementação na estrutura do sistema, como mostra na figura 3 , o aluno/docente, clínica e administrador, poderá entrar nos sistema com suas autenticações corretas, o administrador gerencia os dados, o mesmo faz os cadastros das informações necessários do aluno/docente, para ser encaminhado para a clínica.

3.3 Desenvolvendo a Solução

Na segunda scrum, foi desenvolvida as telas do ambiente do sistema, com as informações obtidas e demonstradas em forma de diagramas, foi avaliado e feito a aplicação dessas telas dentro do sistema, também utilizando a orientação do scrum master (figura 4) avaliando o caminho para realizar o planejamento das telas, realizando a sprint, planejando o fluxo de trabalho, para ser feita as devidas implementações com o melhor aproveitamento de tempo, para finalização das telas de forma ágil, obtendo as telas com uma boa funcionalidade e ser fácil para o entendimento do usuário.

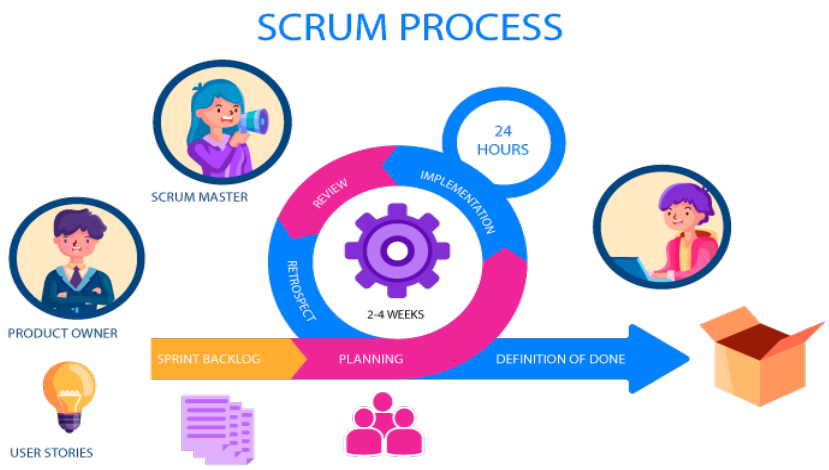


Figura 4. Processo Scrum.

Fonte: Freepik, 2020

Como observamos na figura 1 , foi realizado as orientações estudada de acordo com a primeira sprint pelo scrum master (autor), para a entrada da segunda sprint, colocando ela na fase de planejamento, encontrando as dificuldades, e dividindo o processo de acordo com a dificuldade do produto, produzindo cada situação por tempo determinado, para um melhor desempenho em tempo ágil, depois foram realizadas as implementações e visualizadas e testadas, quando ocorria erros eram voltadas a fase de planejamento foram voltadas para a fase de planejamento, assim realizando o ciclo novamente até ser entregue o resultado do produto.

A primeira tela desenvolvida foi o cadastro dos discentes, que representa a figura 5, tendo a opção de cadastrar o usuário, seu devido curso, a turma, turno e o período e outros dados obrigatórios.

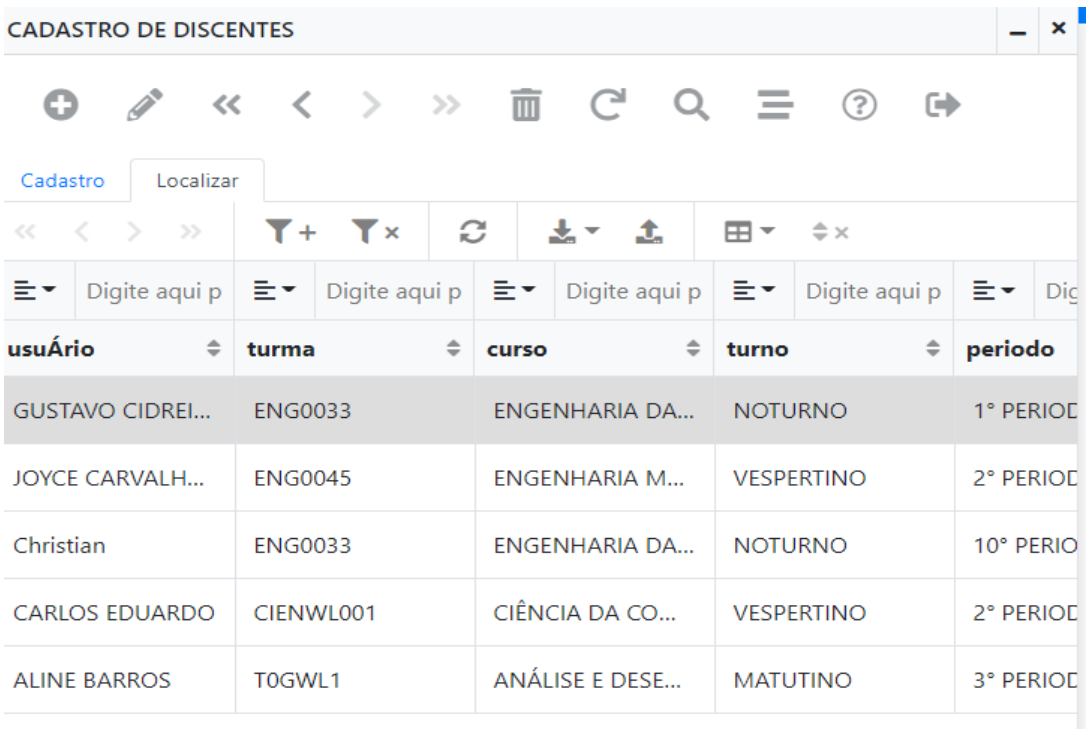


Figura 5. Cadastro de Discentes.

Fonte: Autoral, 2022.

Já na segunda tela mostra, a tela de cadastro do aluno que será encaminhado para a clínica conveniada, marcando a opção de caixa o aluno será enviado para clínica com os dados cadastrados, como mostra a figura 6.

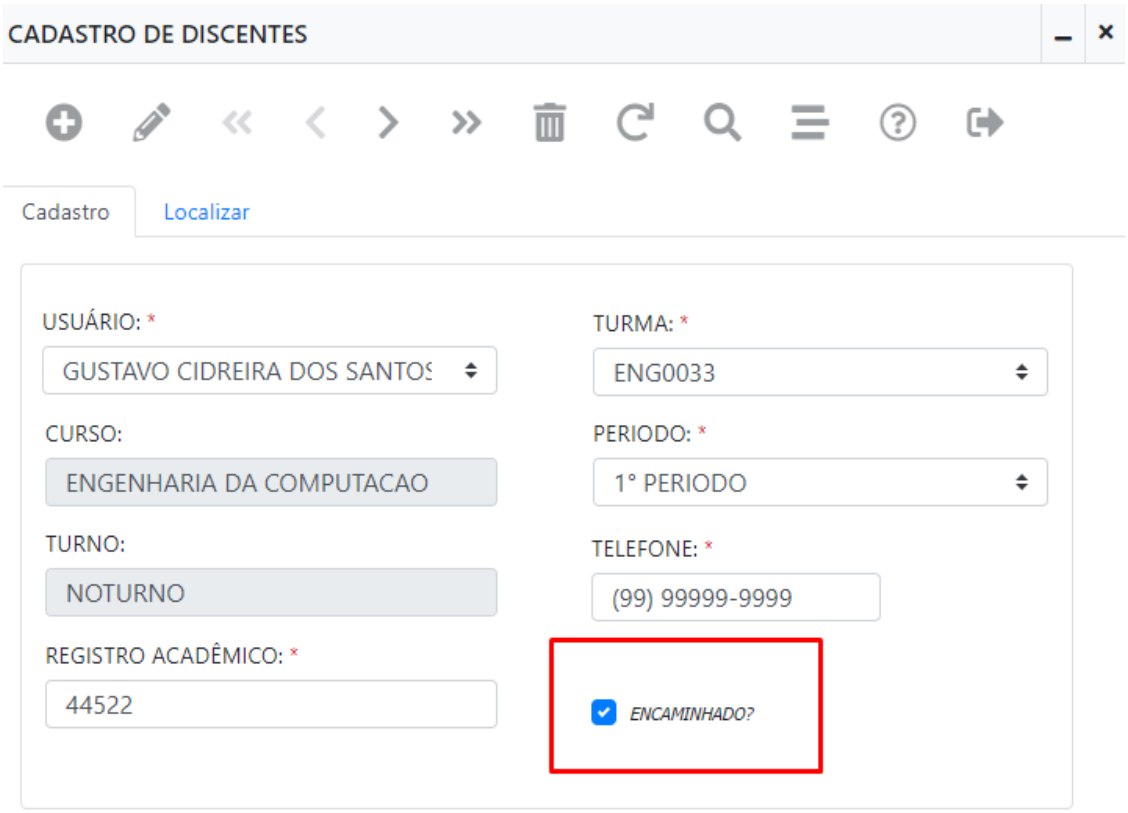


Figura 6. Cadastro de Discentes.

Fonte: Autoral, 2022.

Já na terceira tela mostra, a ficha com os dados dos discentes e docentes enviados para as clínicas conveniadas a instituição, como mostra a figura 7.

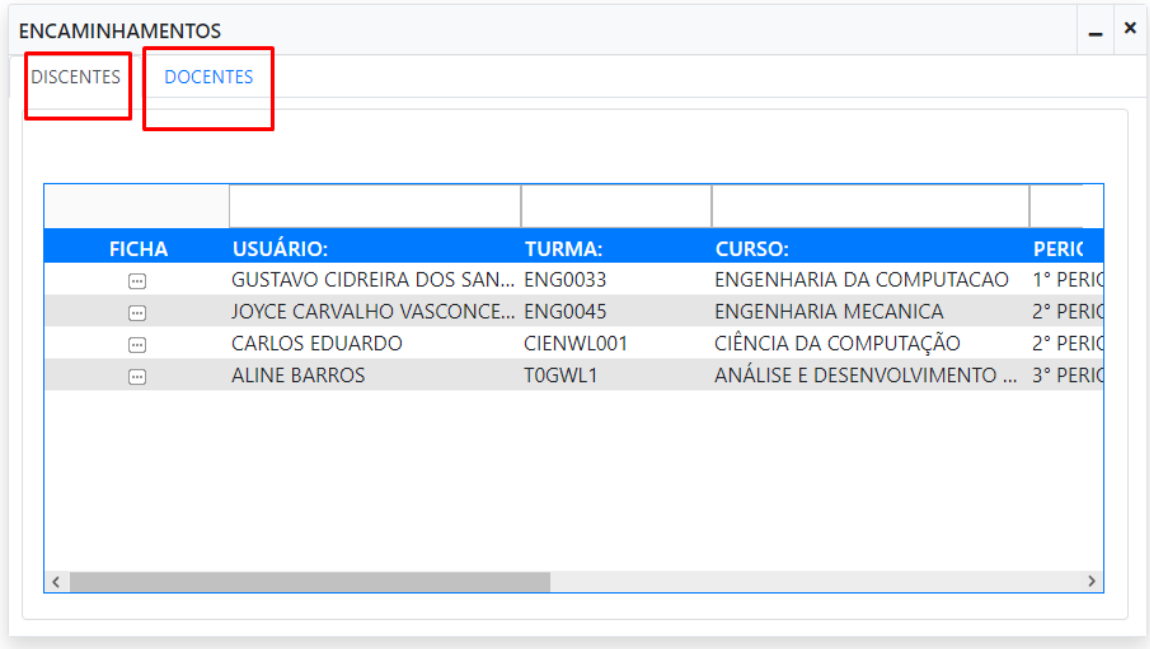


Figura 7. Cadastro de Discentes.

Fonte: Autoral, 2022.

Esses encaminhamentos são todas as informações de cadastro de discentes e docentes, informando o usuário, turma, curso e seu período.

O sistema foi desenvolvido utilizando o maker com a linguagem java, o maker foi utilizado pela ferramenta desenvolvendo as telas de forma rápida utilizando de uma maneira de programação chamada de no-code, uma nova maneira de criação de sistemas mais produtiva para criações e implementações para o sistema sem muito esforço, como mostra a figura 8.

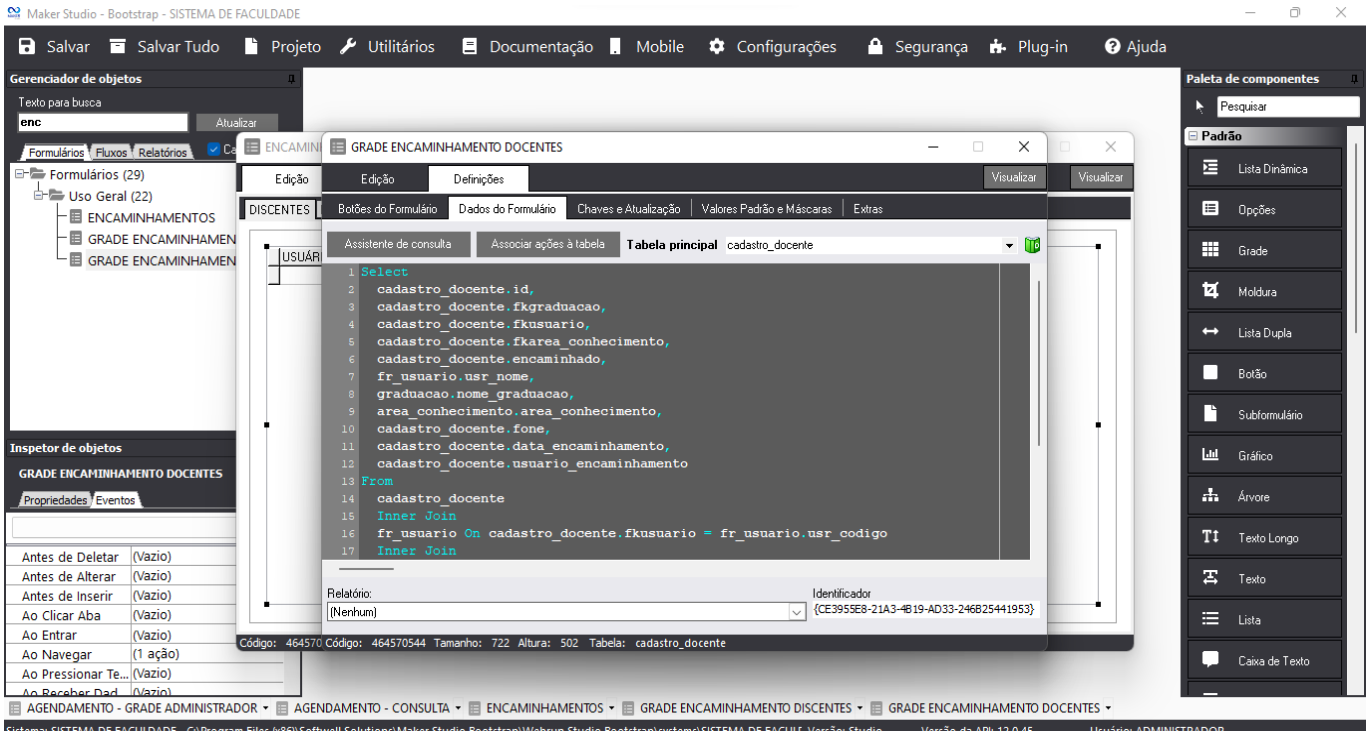


Figura 8. Tela do maker que obtém todos os dados cadastrados.

Fonte: Autoral, 2022.

Um dos padrões de projetos e muito utilizado dentro da maioria dos desenvolvimentos de software é o MVC, Model-View-Control, que nos permite realizar implementações e manutenção de maneiras mais fácil, pela facilidade e organização da separação desses dados, é onde acontece tudo sem que o usuário saiba o que está ocorrendo por trás.

Como mostra a imagem do funcionamento e divisões nas funcionalidades dentro do Maker, na figura 9.

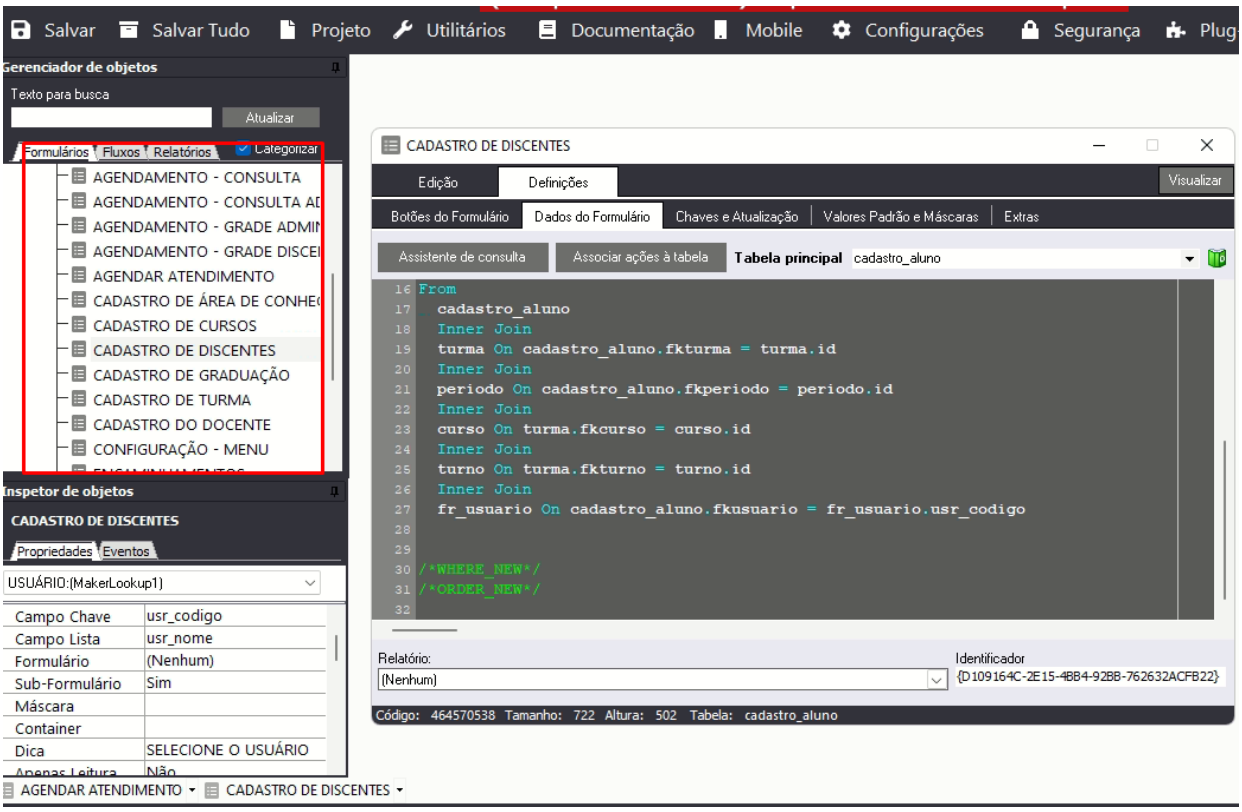


Figura 9. Tela do maker que obtém todos os dados cadastrados.

Fonte: Autoral, 2022.

Este tipo modelagem facilita ao desenvolvedor identificar, o projeto para a modificação e manutenção, e com a conexão ao banco de dados, que mostrar a busca desses dados referentes aos projetos e cada realizar cada funções dentro do sistema, como mostra a figura 10, que representa o banco de dados.

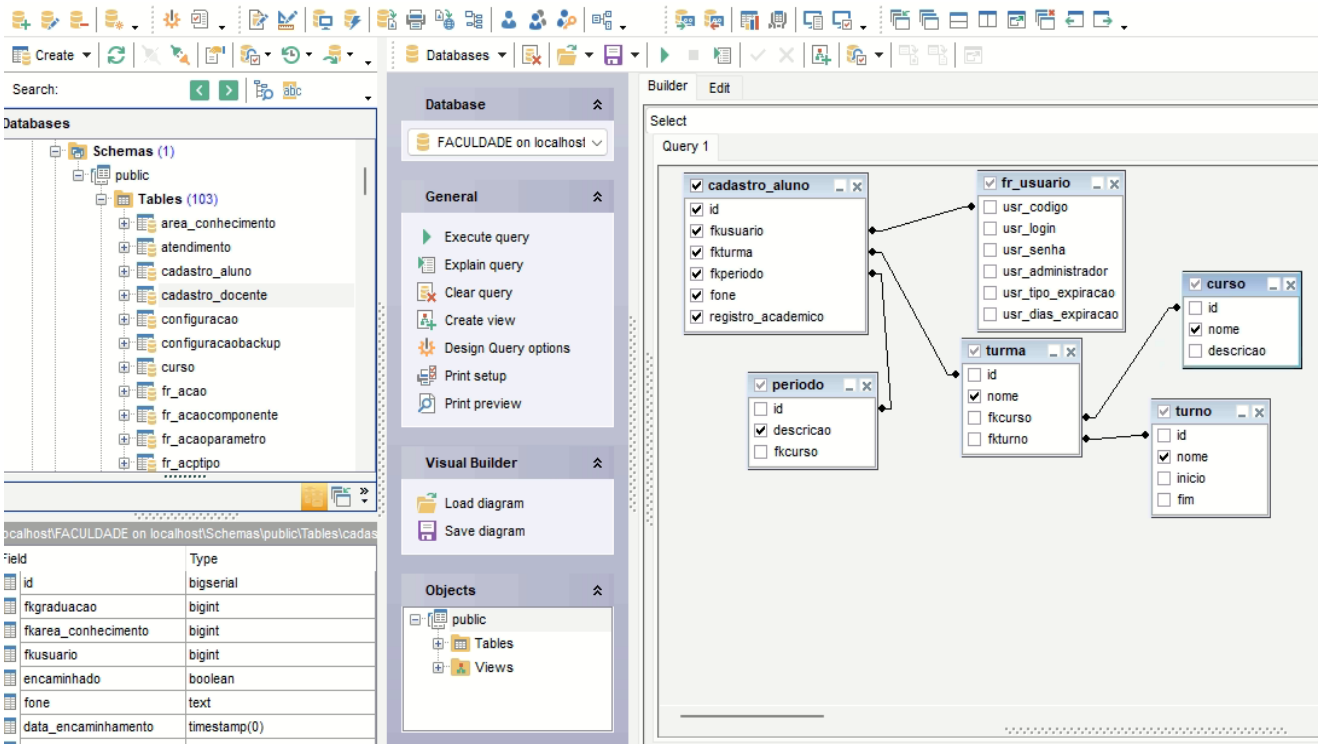


Figura 10. Tela do maker que obtém todos os dados cadastrados.

Fonte: Autoral, 2022.

Assim realizando a conexão entre eles, para desenvolver uma funcionalidade, modelados e organizados, para uma melhor manutenção desses dados, sem que o usuário saiba como é realizado suas ações dentro do sistema.

Para desenvolver um sistema, deve-se ter uma boa orientação de objeto para torna um o sistema bem construído e de fácil manutenção e manipulação para realizar implementações. como a figura 11. mostra uma função criado para o sistema, para a criação de regras, suas várias e parâmetros que serão utilizados para a construção do ambiente.

```
import java.sql.* ;

public class Asmmontarparametros
extends wfr.com.systems.system_fac.rules.WebrunFunctions {

  public Asmmontarparametros() {
    init();
  }

  public Asmmontarparametros(WFRRuleManager manager,
  DBConnection connection, WFRRuleClient client, WFRFormData fields) {
    super(manager, connection, client, fields);
    init();
  }

  /**
   * Define o nome da regra, as variáveis e os parâmetros de entrada.
   */
  private void init() {
    metaData.setReturnType("Letras");
    metaData.add("Variante");
    this.variableNames = new String[]{"posicao", "resultado"};
    this.inputNames = new String[]{"lista de parametros"};
    this.ruleName = "AsmMontarParâmetros";
  }
```

Figura 11. Função para parametrização.

Fonte: Autoral, 2022.

A figura 11, mostra uma função criado para o sistema, para a criação de regras, variáveis e seus parâmetros que serão utilizados para a construção do ambiente.

Com este trabalho foi desenvolvido orientação de objeto, padrões e MVC, que garante uma organização operacional, tornando o sistema fácil a ser entendido e maleável para realizar implementações no ambiente, e ganho operacional tornando bem estruturado ao uso pelo usuário.

4. CONCLUSÃO

O trabalho realizado neste artigo, resolve um grande problema de envio de documentação do setor pedagógico da instituição, proporcionando uma organização em seu setor, e evitando o retorno de falta de informações das clínicas conveniadas, assim administrando os docentes e discentes que necessitam ser atendidos pelas suas clínicas conveniadas.

Ainda existem melhorias a serem acrescentadas no sistema, como mais interações nas telas, e deixando bem mais claro de como é feito os cadastros obrigatórios de dados da instituição para serem realizados os cadastrados corretamente dentro do sistema.

Um dos principais destaques de grandes empresas é ser ágil e ter um bom serviço até atingir o seu produto final obtendo sua integridade. O desenvolvimento deste sistema soluciona uma demanda de encaminhamento de documentos, como dados obrigatórios

para determinadas ações corporativas de uma empresa.

Com esse desenvolvimento, pode ser implementados outras funções de acordo com necessidade do cliente para melhorar e crescer cada vez mais o sistema.

Referências

BOOCH, Grady; RUMBAUCH, James; JACOBSON, Ivair. **UML – Guia do usuário.** Rio de Janeiro: Campus, 2006. ISBN: 978-85-352-1784-1

DENNIS, Alan; WIXOM, Barbara; ROTH, Robetar. **Análise e Projeto de Sistemas.** Editora LTC. 2014.

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. **Padrões de Projetos: Soluções Reutilizáveis de Software Orientados a Objetos.** Editora Bookman. 2000.

GUEDES, Gilleanes T. A. **UML 2 – Uma abordagem prática.** Editora Novatec. 2018.

SANDERS, William**. Aprendendo Padrões de Projeto em PHP.** Publicado em 2013. Editora Novatec.

SOMMERVILLE, Ian. **Engenharia de software.** 6.ed. São Paulo: Addison Wesley, 2003.

SUTHERLAND, Jeff. **Scrum: A arte de fazer o dobro do trabalho na metade do tempo.** Editora GMT.2019.

LIMA, Adilson. **UML 2.5 - do Requisito a Solução.**Editora Érica.2018.

Softwell Maker: **Manual do Maker**. Disponível em < https://documentation.help/softwell.manual.maker.2/documentation.pdf > Acesso em: 15 out 2022.

GUERRA, Eduardo. **Design Patterns com Java. Projeto Orientado a Objetos Guiado por Padrões.** Editora Casa do Código. 2012.

SHVETS, Alexander. **O Mergulho nos Padrões de Projeto.** High Dev. 2021

10

UTILIZANDO A METODOLOGIA SCRUM PARA DESENVOLVIMENTO DE UM SISTEMA MODELADO EM UML EMPREGANDO DESIGN PATTERNS EM JAVA COM HIBERNATE

USING THE SCRUM METHODOLOGY TO DEVELOP A SYSTEM MODELED IN UML USING DESIGN PATTERNS IN JAVA WITH HIBERNATE

Matheus Costa Vaz Souza[1]

Edilson Carlos Silva Lima[2]

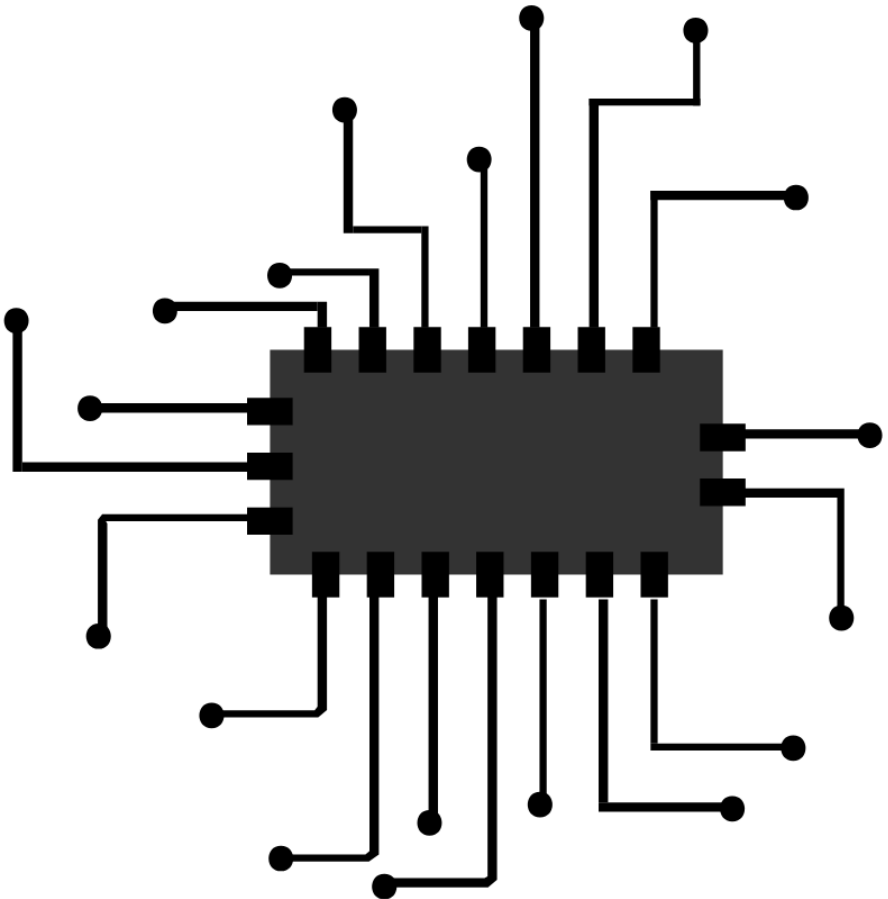
Jonathan de Araújo Queiroz[3]

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

{SOUZA, Matheus Costa Vaz, matheuscostav@outlook.com; LIMA, Edilson Carlos Silva, edilsonlima3@gmail.com; QUEIROS, Jonathan de Araújo, queirozjth@gmail.com}



d.o.i.: 10.29327/5192626.1-10

Resumo

A falta de um gerenciamento em vendas e controle de um produto em um pequeno comércio pode implicar diretamente na perda de clientes e má gestão financeira do negócio. Dessa maneira, a gestão em vendas e controle no estoque vem com o intuito de prevenir que essa falta ocorra. Através de questionamentos sobre qual o volume adequado de estoque, quanto comprar e com que frequência, entre outras. E, levando em consideração tais questionamentos, o presente artigo apresenta um estudo de caso que teve o objetivo o desenvolvimento de um sistema utilizando a metodologia ágil *SCRUM*, o projeto de elaboração a Linguagem de Modelagem Unificada (UML - *Unified Modeling Language*), utilizando *design patterns*, o padrão *Data Access Object* (DAO) que é um padrão estrutural que permite isolar a camada de aplicativo/negócio da camada de persistência e a integração de *frameworks* de código aberto *Hibernate* e o banco de dados *MySQL.*

Palavra-chave: *Design Patterns*; DAO; Hibernate; UML.

Abstract

The lack of sales management and product control in a small business can directly imply the loss of customers and poor financial management of the business. In this way, sales management and stock control comes with the aim of preventing this shortage from occurring. Through questions about the appropriate volume of stock, how much to buy and how often, among others. And, taking into account such questions, this article presents a case study that aimed to develop a system using the agile SCRUM methodology, the project to develop the Unified Modeling Language (UML - Unified Modeling Language), using design patterns, the Data Access Object (DAO) pattern which is a structural pattern that allows you to isolate the application/business layer from the persistence layer and the integration of Hibernate open source frameworks and the MySQL database.

Keywords: Design Patterns; DAO; Hibernate; UML.

1. INTRODUÇÃO

Segundo com os dados divulgado pela SEBRAE (2016), cerca de 99% das empresas no Brasil são micros e pequenas empresas. Na área de vendas a maior dificuldade é a gestão de movimentação das vendas e estoque, onde a administração da maioria das vezes em pequenas empresas é feita de uma forma manual, em que pode ocorrer erros. Para alcançar uma gestão de estoque eficiente, as empresas devem fazer o planejamento da demanda e monitorar cuidadosamente o estoque (RIBEIRO, 2020). Visto esta problemática, a criação de um software eficaz e acessível ajuda muito no dia-dia de um pequeno empreendedor.

Visando esse incomodo, foi utilizado metodologia de pesquisa aplicada, na qual tem como objetivo resolver um problema específico. Na qual o pesquisador toma um cuidado especial para identificar um problema desenvolver uma hipótese de pesquisa e passar a testar essas hipóteses por meio de um experimento. Em muitos casos, essa abordagem de pesquisa usa uma abordagem empírica para resolver problemas práticos. Existem 3 tipos de pesquisa aplicada, Trata-se de pesquisa de avaliação, pesquisa e desenvolvimento e pesquisa-ação.

O *software* foi desenvolvido utilizando a técnica ágil *SCRUM,* os diagramas em UML (*Unified Modeling Language*), design patterns para proporcionar o reaproveitamento de soluções dentro do sistema, o *frameworks Hibernate* e utilizado o banco de dados *MySQL*.

Este artigo tem por objetivo apresentar como foi desenvolvido um sistema *desktop* que gerencia o estoque e vendas de uma pequena empresa. Ela está dívida em 5 Capítulo: 2 – Revisão Bibliográfica, 3 – Estudo de Caso, 4 – Resultado e 5 – Conclusão.

2. REVISÃO BIBLIOGRÁFICA

Neste capítulo será demonstrada toda a fundamentação teórica que serviu como base de estudo e auxílio no entendimento para o desenvolvimento deste artigo. No item 2.1 *Scrum*, 2.2 OO, 2.3 UML, 2.4 *Design Patterns,* 2.5 DAO, 2.6 *Frameworks Hibernate* e por fim no item 2.7 Banco de Dados – MySQL.

2.1 *Scrum*

Scrum é um *framework* de gerenciamento de projeto, surgiu após seu criador Jeff Sutheraland ver vários processos em seus diferentes empregos ou projetos ter métodos antiquado e furado, na qual notou que essas metodologias estava longe de ser ideal, por exemplo quando trabalhou na FBI em que os relatórios eram escritos em papeis e quando aprovados recebiam um determinado número escrito em papel, na qual esse método recebeu culpa pelo que aconteceu em 11 de setembro (SUTHERLAND, 2014).

O *framework Scrum* é uma técnica ágil de software projetado para aumentar o dinamismo, foco e transparência às equipes de projeto que desenvolvem sistemas de software (SUTHERLAND, 2007). É uma estrutura leve projetada para ajudar equipes pequenas e unidas de pessoas a desenvolver produtos complexos.

Nesse projeto foi utilizado a ideais do *Scrum* para o gerenciamento da criação sistema em *desktop* de gestão em vendas e controle no estoque. Segundo SUTHERLAND (2014) os principais conceitos do *framework* retirados para o projeto foi:

- **Dono do Produto**: é responsável pela visão do que você vai fazer ou alcançar. Julgue os riscos e benefícios, o que é possível e o que pode ser feito.
- **Pendência do Produto:** é o responsável por tratar de uma lista detalhada de tudo o que você precisa fazer ou edificar para dar vida à sua visão. Esses tópicos existem e evolui durante o desenvolvimento do produto. Em cada etapa do projeto são a visão única e definitiva de tudo o que deve fazer em todos os momentos por ordem de prioridade.
- **Planejamento do Sprint**: consiste em que o **Dono do Produto** se reúne para planejar o Sprint, na qual esses Sprint são um ciclo de trabalho, em que temos um objetivo e tempo para resolver.
- **Revisão do Sprint:** esta é a reunião que mostra o que foi realizado durante o Sprint.
- **Retrospectiva do Sprint:** é uma reunião recorrente dedicada a discutir o que deu certo e o que pode ser aprimorado em um *Sprint*. Também lhe dá a chance de se recuperar de um *Sprint* e se preparar para o próximo. Com uma **Retrospectiva do Sprint**, você pode tornar cada *Sprint* mais fácil e mais bem-sucedido do que o anterior.

2.2 POO (Paradigma de Orientação a Objetos)

O Paradigma de Orientação a Objetos (POO) é um modelo de programação de computador que organiza o projeto de software em torno de dados ou objetos, em vez de funções e lógica. Um objeto pode ser definido como um campo de dados com atributos e comportamento únicos (HOURANI, WASMI e ALRAWASHDEH, 2019). De acordo com os principais conceitos por trás da POO (Figura 1) é a abstração, encapsulamento, herança e polimorfismo e segundo a ISO/IEC/IEEE (2010) define:

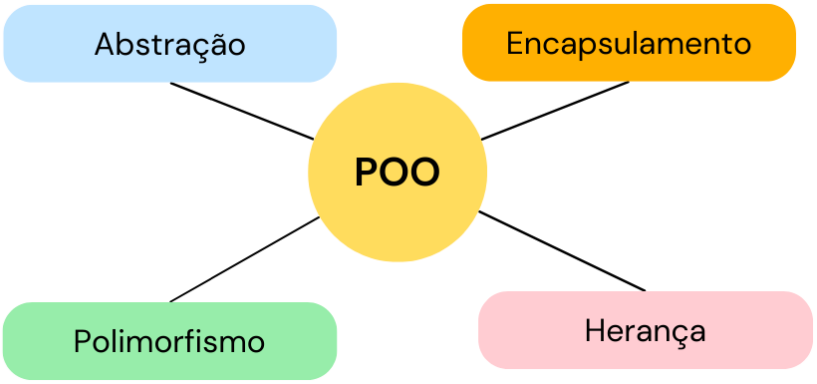


Figura 1 - Pilares do POO

Fonte: autoral, 2022.

- Abstração: o processo de acobertar os detalhes de implementação de um objeto para que possa ser usado sem entender como o objeto funciona. Ele permite que você crie código que é fácil de usar e manter.
- Encapsulamento: uma técnica de desenvolvimento de software que consiste em isolar uma função do sistema ou um conjunto de dados e operações nesses dados dentro de um módulo e fornecer especificações precisos para o módulo. É importante porque ajuda a manter seus dados seguros. Além disso, você pode alterar a

implementação do seu código sem afetar o restante da base.

- Herança: um mecanismo que permite que uma nova classe seja criada a partir de uma classe existente, aproveitando as características existentes da classe que está sendo estendida.
- Polimorfismos: Implementar o mesmo método ou operador de maneiras diferentes. Isso é útil porque permite criar um código mais flexível e adaptável e também podem ser usados com vários tipos de objetos.

2.3 UML

UML - Linguagem de Modelagem Unificada – é uma linguagem visual utilizada para modelar softwares baseados no paradigma de orientação a objetos (BRAUDE, 2017). Outro destaque para a UML na qual o BOOCH (2006) afirma que:

> UML proporciona uma forma-padrão para a preparação de planos de arquitetura de projetos de sistemas, incluindo aspectos conceituais tais como processos de negócios e funções do sistema, além de itens concretos como as classes escritas em determinada linguagem de programação, esquema de bancos de dados e componentes de software reutilizáveis.

Cada diagrama UML analisa um sistema ou parte de um sistema de uma determinada perspectiva. Como se o sistema fosse modelado hierarquicamente, alguns diagramas focam de forma mais geral no sistema, apresentando uma visão externa, que é o objetivo dos diagramas de casos de uso, enquanto outros diagramas fornecem uma visão mais profunda do sistema. As falhas podem ser encontradas usando vários gráficos, reduzindo a probabilidade de erros futuros.

Os diagramas UML são divididos em diagramas de estrutura e diagramas de comportamento, conforme mostrado na Figura 2. Este último também possui subseções representadas por diagramas de interação.

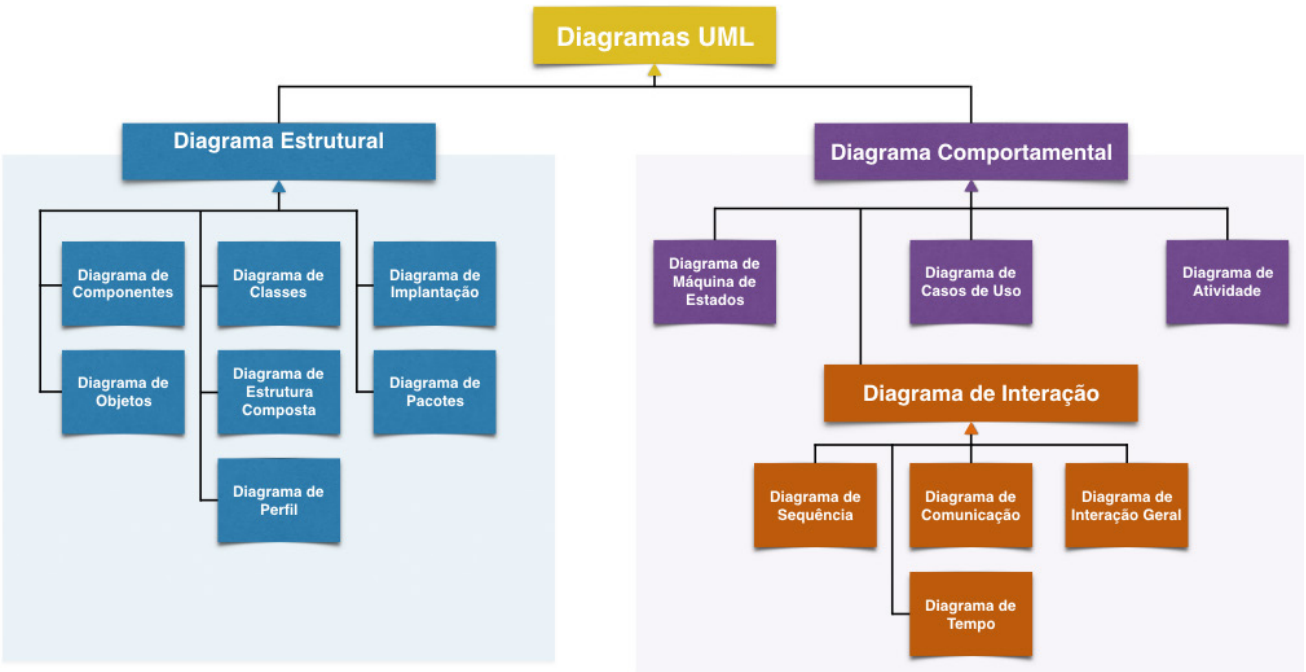


Figura 2 - Divisão dos Diagramas da UML

Fonte: Ed Wilson Jr., 2020.

Para Bezerra (2003), na UML, o diagrama de caso de uso resume os detalhes dos usuários do seu sistema (também conhecidos como atores) e as interações deles com o sistema. Para criar um, use um conjunto de símbolos e conectores especializados. Um bom diagrama de caso de uso ajuda sua equipe a representar e discutir:

- Cenários em que o sistema ou aplicativo interage com pessoas, organizações ou sistemas externos
- Metas que o sistema ou aplicativo ajuda essas entidades (conhecidas como atores) a atingir
- O escopo do sistema

Segundo Lima (2009), o diagrama de classes é uma representação estática utilizada na área da programação para descrever a estrutura de um sistema, apresentando suas classes, atributos, operações e as relações entre os objetos.

Este tipo de representação é bastante útil no desenvolvimento de sistemas e de *softwares* de computação, pois define todas as classes que o sistema precisa ter e serve de base para a construção de outros diagramas que definem o tipo de comunicação, sequência e estados dos sistemas. O diagrama de classes representa as principais finalidades da UML, tendo a função de separar os elementos de design da codificação do sistema. Esta linguagem ajuda a modelar diversos subconjuntos de diagramas, incluindo diagramas de comportamento, interação e estrutura. Normalmente, ela é utilizada por engenheiros para documentar a arquitetura dos *softwares*.

2.4 *Design Patterns*

Design Patterns – Padrões de projeto – fornecem uma maneira eficiente de criar software orientado a objetos mais flexível, complexo e reutilizável. Portanto, sua aplicação correta no projeto do sistema tem um grande impacto na reutilização, flexibilidade, extensibilidade e manutenção do sistema (LIU, 2009).

Um dos maiores desafios na detecção de padrões de projeto é a identificação precisa da simetria entre a definição de um modelo de projeto e as instâncias do modelo de projeto em sistemas de software. Ao considerar mais variantes, os métodos de reconhecimento tornam-se mais capazes de determinar a simetria entre as instâncias do padrão de projeto que se aplicam especificamente a essas variantes e a definição do padrão de projeto. Consequentemente, aumentar o número de variantes de projeto cobertas por um método de detecção desempenha um papel importante no aumento de sua acurácia (KOULI e RASOOLZADEGAN, 2022).

Os padrões de projeto podem acelerar o processo de desenvolvimento provendo um paradigma de desenvolvimento comprovado. O design de software eficaz requer a consideração de questões que podem não se tornam aparentes até mais tarde na implementação. A reutilização de padrões de projeto ajuda a evitar problemas sutis que podem causar sérios problemas e melhora a legibilidade do código para programadores e arquitetos experientes em padrões.

MVC é um padrão de arquitetura de software, separando sua aplicação em 3 camadas. A camada de interação do usuário (View ou Visão), a camada de manipulação dos dados (Model ou Modelo) e a camada de controle (Controller ou Controle), a Figura 3 mostra a organização MVC.

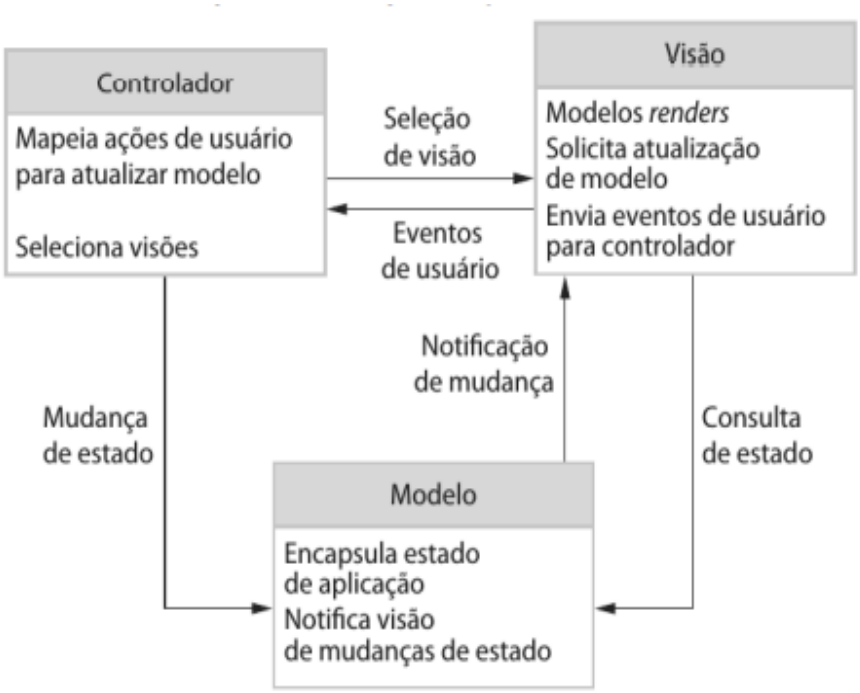


Figura 3 - Organização do MVC

Fonte: SOMMERVILLE, 2011, 2022.

A camada **Model** (modelo) é responsável pela leitura e escrita de os dados bem como verificar a exatidão. A camada **View** (visão) é responsável por usar as informações disponíveis para produzir qualquer interface de apresentação que seu aplicativo possa precisar. E a camada **Controller** (controlador) é responsável por retornar uma resposta com a ajuda das camadas Model e View.

2.5 DAO

O *Data access object (DAO)* que tem como intenção separar a lógica de negócio da persistência do banco de dados, para que ambas as partes possam evoluir e ser alteradas independentemente (ANICHE, 2014).

A abordagem DAO isola o desenvolvedor de muitas complexidades de programação de banco de dados enquanto fornece objetos, propriedades e métodos de acesso a dados de alto nível.

2.6 *Frameworks Hibernate*

O *Hibernate* é um MOR (mapeamento objeto-relacional) e é um middleware de código aberto que fornece serviços de banco de dados. Na qual deixa a comunicação com o banco de dados mais simples. A estrutura do software pode ser dividida em uma ou mais camadas. Na fase de criação da aplicação, frequentemente é utilizada a arquitetura de duas camadas, ou seja, modelo cliente/servidor (YAN, 2018). E conforme o LIU (2009) o *Hibernate* não apenas cuida do mapeamento de classes Java para tabelas de banco de dados, mas também fornece recursos de consulta e recuperação de dados e pode reduzir significativamente o tempo de desenvolvimento gasto com manipulação manual de dados em SQL e JDBC.

2.7 Banco de Dados - MySQL

Sistema de Gerenciamento de Banco de Dados uma coleção de programas que permite aos usuários acessar bancos de dados e gerenciar, manter, gerar relatórios e associar dados. O SGBD é frequentemente usado para reduzir a redundância de dados. Troca de dados controlada e reduzir problemas de integridade de dados. O SGBD não é um sistema de informação, mas é apenas um software (MILANI, 2007).

MySQL é um servidor e gerenciador banco de dados relacional de código aberto convencional. De acordo com a Trybe (2020) o MySQL é o terceiro banco de dado mais utilizado no mundo, por ser robusto, rápido e sua licença é gratuita tanto para afins acadêmicas como para negócio.

3. ESTUDO DE CASO

Neste artigo é detalhado alguns passos dados para o resultado obtido no programa, em que teve como base o gerenciamento de projeto *Scrum*, na qual algumas ideias aprendidas foram colocadas em prática para o desenvolvimento do software. No item 3.1 O Sistema, 3.2 Análise de Negócio, 3.3 Desenvolvimento e Implementação, 3.4 Evolução do Sistema.

3.1 O sistema

Para iniciar o desenvolvimento do sistema de gestão e controle de vendas de produto primeiramente foram levantados os requisitos junto ao cliente onde nesta etapa foi realizada uma entrevista com o dono da empresa foi solicitada a funcionalidade básica do software e para o desenvolvimento utilizamos o *framework* de gerenciamento de projeton *Scrum.*

Figura 4 - Processo do Scrum

Fonte: Autoral, 2022.

O *Scrum* como foi citado anteriormente, existe principais conceitos, no qual foi se-

parado e constituído. A primeira divisão foi achar que seria o **Dono do Produto** que é responsável pela visão do que você vai fazer ou alcançar. Julgue os riscos e benefícios, o que é possível e o que pode ser feito para fazer essa construção das ideias, conforme na Figura 4, o projeto foi planejado com o Processo do *Scrum*.

Logo após essa divisão constituímos o **Planejamento do Sprint** que consiste através de uma reunião feita pelo **Dono do produto** para planejar o que irá ser produzido através dos dias, na qual esses Sprint são um ciclo de trabalho, em que temos um objetivo e tempo para resolver. E para o **Planejamento do Sprint** foi feito o conceito principal para criação do software foi uma entrevista com o microempreendedor, na qual informou o que precisava do programa. Na qual suas principais necessidades foram o (1) administração de vendas, (2) estoque dos produtos e (3) relatório de vendas. E essas necessidades foram separas em Sprints conforme a Figura 4.

Logo após essa divisão constituímos o **Planejamento do Sprint** que consiste através de uma reunião feita pelo **Dono do produto** para planejar o que irá ser produzido através dos dias, na qual esses Sprint são um ciclo de trabalho, em que temos um objetivo e tempo para resolver, aderimos também, a **Pendência do Produto**, que se trata de uma lista detalhada de tudo o que você precisa fazer ou edificar para dar vida à sua visão.

Figura 5 - Planejamento do Sprint

Fonte: Autoral, 2022.

Esses tópicos existem e evolui durante o desenvolvimento do produto. Em cada etapa do projeto são a visão única e definitiva de tudo o que deve fazer em todos os momentos por ordem de prioridade. E por final dentro de uma das metodologias do Scrum foi a **Demonstração do Sprint**, no qual na reunião mostra o que foi realizado durante o Sprint e assim podendo da continuidade nos desenvolvimentos e tornando mais fácil e rápido de identificar em que segmento está como prioridade a sem desenvolvido.

3.2 Análise do Negócio

Para ter um desenvolvimento do *software* de uma forma clara e eficaz foi utilizado a criação dos diagramas da UML, o Diagrama de casos de uso (Figura 6).

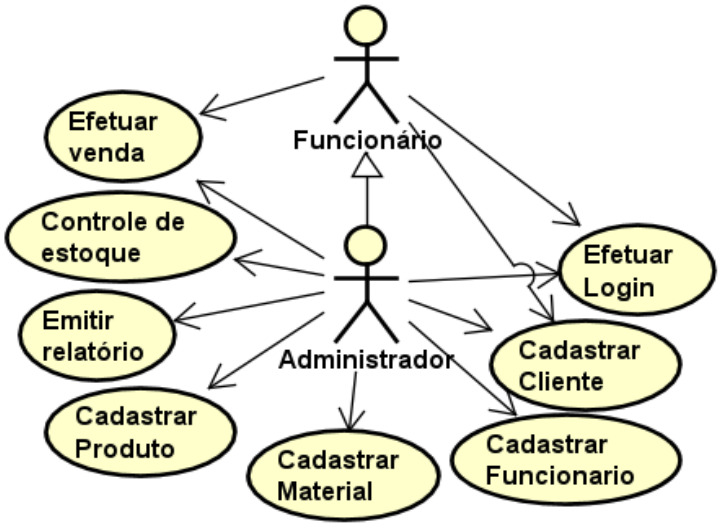


Figura 6 - Diagrama de Casos de Uso.

Fonte: Autoral, 2022.

Em que consiste no ator **Administrador** e **Funcionário,** em que o **Administrador** tem associação a casos de uso de efetuar login, efetuar venda, Controle de Estoque, emitir relatório, cadastrar produto, cadastrar material, cadastrar funcionário e cadastrar cliente. Já o ator **Funcionário** tem apenas associação de casos de uso em efetuar login, cadastrar cliente e efetuar venda.

O diagrama de Casos de Uso Cadastrar Funcionário do Ator Administrador está ilustrado na Figura 7.

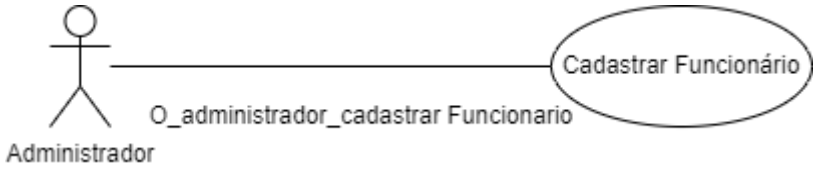


Figura 7 - Diagrama de Caso de Uso - Cadastrar Funcionário

Fonte: autoral, 2022.

E a Tabela 1 mostra a especificação do Caso de Uso Cadastrar Funcionário.

Nome do Caso de Uso	Cadastrar Funcionário
Caso de Uso Geral	
Ator Principal	Administrador
Ator Secundário	
Resumo	Esse caso de uso descreve as etapas percorridas pelo administrador para cadastrar um Funcionário.
Pré-Condições	
Pós-Condições	
Fluxo Principal	
Ações do Ator	Ações do sistema
1. Clicar no Menu Cadastrar Funcionário	
	2. Mostra a tela Cadastro de Funcionário

3. 3. Preencher corretamente os dados do formulário	
4. Clicar em Salvar	
	5. Sistema salva as informações e o formulário é resetado.
Restrições/Validações	1. Para Cadastrar o Funcionário o usuário deve informar todos os dados do formulário.
Fluxo Alternativo – Manutenção do cadastro de Funcionário	
Ações do Ator	Ações do sistema
1. O usuário deve selecionar a Funcionário deseja alterar e clica em Editar	2. O sistema Salva as informações alteradas pelo Usuário.
Fluxo Alternativo – Excluir Funcionário	
Ações do Ator	Ações do sistema
1. O usuário deve selecionar a Funcionário deseja excluir e clica em Excluir	2. O sistema exclui as informações do Funcionário.
Fluxo de Exceção – Informações do Formulário	
Ações do Ator	Ações do Sistema
	1.O usuário deve informar obrigatoriamente todas as informações da Loja, sem exceção.
	2. Caso não preencha recusar informações.

Tabela 1 - Especificação do Caso de Uso Cadastrar Funcionário.

Fonte: autoral, 2022.

E com a utilização da ferramenta *MySQL Workbech*, foi feito a modelagem lógica do banco de dados (Figura 8).

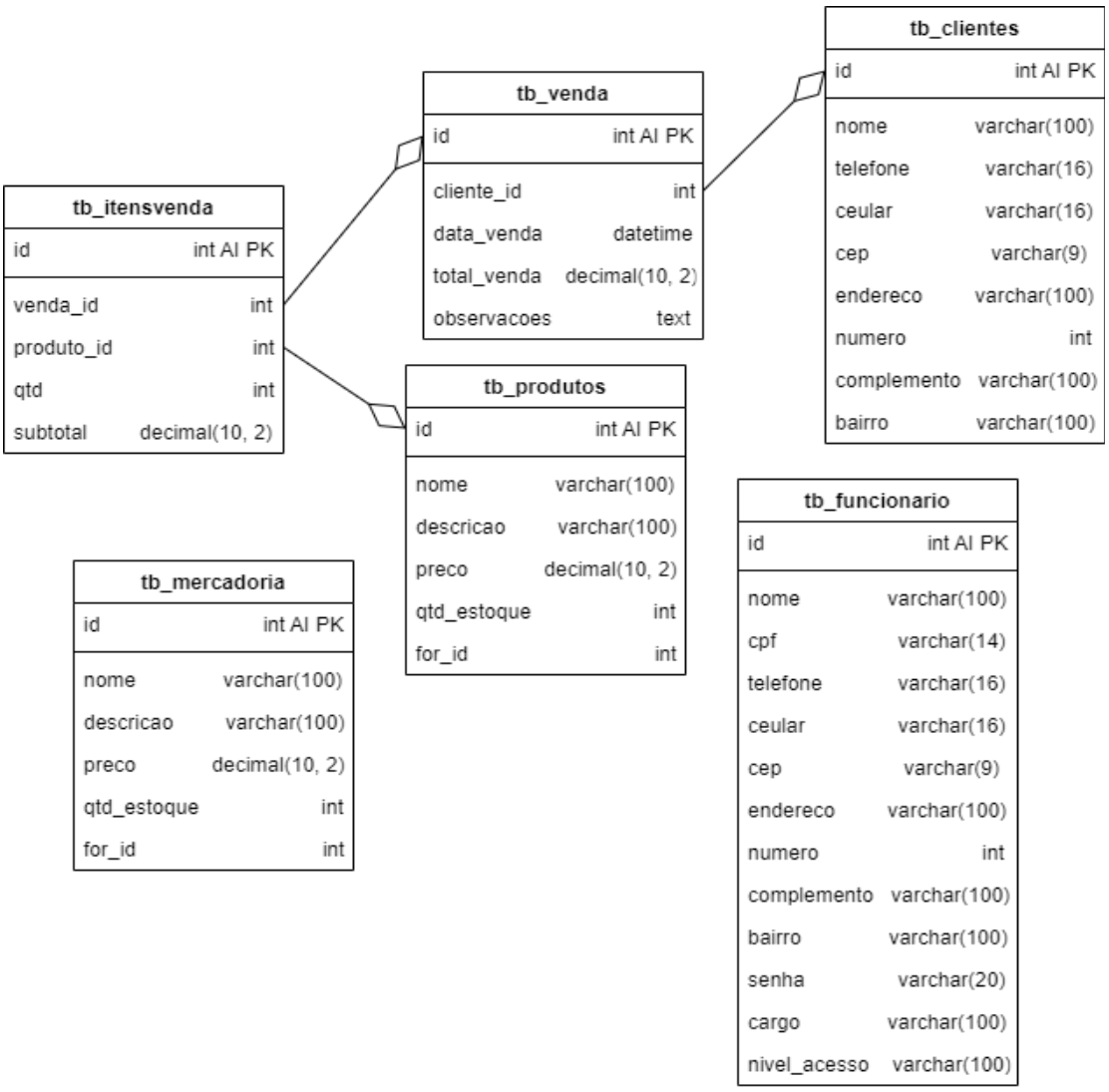


Figura 8 - Modelagem Relacional do Banco de Dados

Fonte: autoral, 2022.

3.3 Desenvolvimento e Implementação

Criação do software — para o desenvolvimento do programa, utilizamos o *NetBeans IDE,* seguindo as normas do padrão MVC (Figura 9) para o projeto e para o banco de dados o padrão DAO. Primeiro foi mapeado as classes, com o uso do JPA *Annotations* do *Hibernate.*

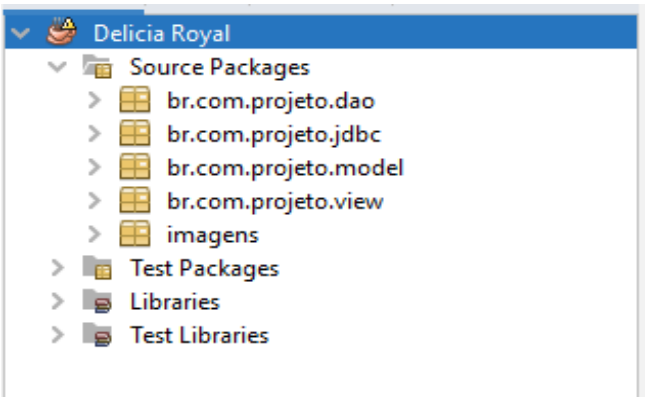


Figura 9 - Padrão de projeto MVC

Fonte: autoral, 2022.

O primeiro pacote foi o Diretório *View,* em que é a responsável por interagir com o usuário, famosa "telinha", na qual foi seguido um padrão para o uso fácil (Figura 10), neste exemplo temos a tela de cadastro de funcionário, na qual apenas o **Administrador** tem acesso, e coloca o formulário seguindo a Tabela 1.

Figura 10 - Tela de Cadastro de Funcionário

Fonte: autoral, 2022.

Em seguida temos o Diretório Model (Figura 9), na qual tem como propósito encapsular os dados da camada *View*, por exemplo neste modelo é pego os atributos do model Cliente e estendido no model *Funcionários*. E adicionados alguns atributos.

```
public class Funcionarios extends Clientes{

    //Atributos
    private String senha;
    private String cargo;
    private String nivel_acesso;

    //Getters e Setters
    public String getSenha() {
        return senha;
    }

    public void setSenha(String senha) {
        this.senha = senha;
    }

    public String getCargo() {
        return cargo;
    }

    public void setCargo(String cargo) {
        this.cargo = cargo;
    }

    public String getNivel_acesso() {
        return nivel_acesso;
    }

    public void setNivel_acesso(String nivel_acesso) {
        this.nivel_acesso = nivel_acesso;
    }
```

Figura 11 - Classe de Modelo de Funcionário.

Fonte: autoral, 2022.

Logo após é criado o Diretório *DAO,* onde é um objeto que interage com o banco de dados *MySQL,* ele tem métodos em que pega os objetos da camada *Model* e passa para o banco de dados (Figura 10), neste exemplo temos um método excluir funcionário.

```
//Metodo excluir funcionario
public void excluirFuncionarios(Funcionarios obj) {
    try {
        //Criar o comando sql
        String sql = "delete from tb_funcionarios where id=?";

        PreparedStatement stmt = con.prepareStatement(sql);
        stmt.setInt( parameterIndex: 1,  x: obj.getId());

        //Executar o comando sql
        stmt.execute();
        stmt.close();

        JOptionPane.showMessageDialog( parentComponent: null,  message: "Excluido com Sucesso!");
    } catch (SQLException erro) {
        JOptionPane.showMessageDialog( parentComponent: null, "Erro ao Excluir. Erro:" + erro);
    }
}
```

Figura 12 - Camada DAO como o método Excluir Funcionário

Fonte: autoral, 2022.

E para se conectar ao banco de dados foi feito um Diretório **jdbc,** e criado um método *Java Connection* que é da biblioteca *jdbc (Java Database Connectivit)* (Figura 11) em que usamos a classe *DriverManager* que conecta com o banco de dados.

```
public class ConnectionFactory {

    public Connection getConnection(){

        //Tratamento de erro
        try {

            //Conecta de verdade é o DriverManager
            return DriverManager.getConnection(
                    url: "jdbc:mysql://localhost/[illegible]",
                    user: "[illegible]",
                    password: "[illegible]");

        } catch (Exception erro) {
            throw new RuntimeException( cause: erro);
        }
    }

}
```

Figura 13 - método Java Connection

Fonte: autoral, 2022.

Entrega — após a criação e teste com o programa, foi mostrado ao empreendedor em que o mesmo fez seus testes e aprovou e implementou em sua empresa.

4. RESULTADO

A proposta dada a este projeto foi desenvolver um sistema *desktop* utilizando como base principal a linguagem Java, e nele ajudar uma pequena empresar gerenciar suas vendas e estoque. Com a linguagem utilizada, que é um desenvolvimento claro e organizado, foi facilmente criado o *software.*

Durante o processo de criação do sistema, foi utilizada uma metodologia de pesquisa aplicada, que tem como foco o desenvolvimento de novos produtos e serviços a partir das necessidades do mercado-alvo. Ele se concentra na coleta de informações sobre as necessidades de mercadologia e no descobrimento de maneiras de melhorar um produto existente ou desenvolver novos produtos que atendam às necessidades identificadas. A pesquisa aplicada também ajuda os empregadores a identificar e atender às necessidades de produtividade dos funcionários. O método aplicado para coletar dados foi a entrevista, que é um método de coleta de dados qualitativos envolvendo interações individuais ou discordâncias com sujeitos a pesquisa, para coletar dados relevantes que podem ser usados como dados empíricos. Outrossim, destaca-se o estudo de caso, que procura temas que já foram estudados por outros pesquisadores. Ora, se este fosse um caso especial, possuir sempre um aspecto distintivo que o distinguia dos demais, permitindo uma concepção diferente do objeto de estudo.

Com base nas ideias de organização de projeto *Scrum* o desenvolvimento foi bem orgânico, ordenado e projetado. E com o uso dos diagramas da UML foi possível entender as necessidades e ajustar a modelagem do programa antes mesmo de codificar. A sua compilação foi feita no ambiente *NetBeans* e com a utilização das normas do padrão MVC o projeto foi organizado para a facilitar a progressão e futuras manutenções e implementações. E para a sua comunicação facilitada com o banco de dados *MySQL* o padrão de projeto DAO.

O *software* foi implementado conforme exigido pelo cliente e atendeu suas necessidades, onde lhe ajudou no seu dia a dia de forma mais rápida, eficiente e fácil.

5. CONCLUSÃO

Com a utilização da linguagem Java, banco de dados *MySQL, framework Hibernate,* gerenciamento de projetos de metodologia agil *Scrum* e os padrões de projeto durante o procedimento de desenvolvimento foi adquirido muito conhecimento dessas tecnologias e padrões utilizados, e ao decorrer da sua criação ocorreu alguns empecilhos em que aprimorou a sua criação.

Futuramente será desenvolvido algumas melhorias como a realização de impressão de relatório, na qual conseguira ver de uma forma mais ampla o seu desenvolvimento.

Referências

"ISO/IEC/IEEE International Standard - Systems and software engineering -- Vocabulary," in *ISO/IEC/IEEE 24765:2010(E)*, vol., no., pp.1-418, 15 Dec. 2010.

ANICHE, Maurício F.; OLIVA, Gustavo A.; GEROSA, Marco A. **Are the methods in your data access objects (DAOs) in the right place? A preliminary study.** In: 2014 Sixth International Workshop on Managing Technical Debt. IEEE, 2014. p. 47-50.

Bezerra, E. **Princípios de Análise e Projeto com a UML**, ed. CampusElsevier. 2003.

BOOCH, Grady. **UML: guia do usuário**. Elsevier Brasil, 2006.

BRAUDE, Eric. Incremental UML for agile development: embedding UML class models in source code. In: **2017 IEEE/ACM 3rd International Workshop on Rapid Continuous Software Engineering (RCoSE)**. IEEE, 2017. p. 27-31.

GUEDES, Gilleanes TA. **UML 2-Uma abordagem prática**. Novatec Editora, 2018.

HOURANI, Hussam; WASMI, Hiba; ALRAWASHDEH, Thamer. A code complexity model of object oriented programming (OOP). In: **2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)**. IEEE, 2019. p. 560-564.

KOULI, Mariam; RASOOLZADEGAN, Abbas. A Feature-Based Method for Detecting Design Patterns in Source Code. **Symmetry**, v. 14, n. 7, p. 1491, 2022.

LIMA, Adilson da Silva. **UML 2.5 do requisite a solução.** 2009. Editora Saraiva.

LIU, Shuang; CHEN, Peng. **Developing java EE applications based on utilizing design patterns.** In: 2009 WASE International Conference on Information Engineering. IEEE, 2009. p. 398-401.

MILANI, André. **MySQL-guia do programador**. Novatec Editora, 2007.

RIBEIRO, Paulo Pinto. **Avaliação da gestão de estoque em uma microempresa de autopeças utilizando a curva abc como ferramenta de apoio.** Revista Cereus, v. 12, n. 2, p. 130-146, 2020.

SEBRAE. **Pequenos negócios em números.** Disponível em https://www.sebrae.com.br/sites/PortalSebrae/ufs/sp/sebraeaz/pequenos-negocios-em-numeros,12e8794363447510VgnVCM1000004c00210aRCRD. Publicado em Setembro de 2016. Acessado em Outubro de 2022.

SUTHERLAND, Jeff et al. **Distributed *Scrum*: Agile project management with outsourced development teams.** In: 2007 40th annual Hawaii international conference on system sciences (HICSS'07). IEEE, 2007. p. 274a-274a.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011. ISBN 978-85-7936-108-1.

SUTHERLAND, Jeff. ***SCRUM*: A arte de fazer o dobro de trabalho na metade do tempo**. Leya, 2014.

Trybe. **Banco de dados: Tipos o que é e suas diferenças!.** Disponivel em https://blog.betrybe.com/tecnologia/bancos-de-dados/. Publicado em Maio de 2020. Acessado em novembro de 2022.

YAN, Liang. **Construction and Testing of Modern Distance Learning Platform System Based on Struts and Hibernate Framework.** In: 2018 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS). IEEE, 2018. p. 72-74.

AUTORES

BECKMAN, João Victor Vieira

Graduado em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), Brasil.

CARNEIRO; Lucas Pereira Saraiva Carneiro

Graduado em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), Brasil.

CIDREIRA, Gustavo dos Santos

Graduado em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), Brasil.

DINIZ, Sam Helson Nunes

Graduado em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), Brasil.

FARIAS, Afonso Jansen de Mello

Graduado em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), atualmente trabalha como analista de banco de dados no NTI da Faculdade CEST, Desenvolvedor *Web* e de *App* na *GeekApps*, atua como técnico de computadores e notebooks nas horas vagas e *hobista* na área de automação e robótica.

FREIRE, Victor Rodrigues

Graduado em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), Brasil, possui formação técnica em Eletrônica pelo IFMA (2016.

MOUTA, Gabriel Mendes

Graduado em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), Brasil.

SILVA, Allicia Sousa da

Graduada em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), atuando hoje na área de desenvolvimento de software na Pulse, maior software *house* do Maranhão.

SOUZA, Matheus Costa Vaz

Graduado em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), Brasil.

VERAS, Kauã, Pereira

Graduado em Engenharia de Computação pela Universidade Ceuma (UniCEUMA), Brasil.

ISBN: 978-65-80751-66-2