PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

PROJETO E DESENVOLVIMENTO DE APLICATIVO PARA AUXILIAR EM COMPRAS TECNOLÓGICAS

PAULO HENRIQUE LEÃO DE OLIVEIRA

GOIÂNIA
2024

PAULO HENRIQUE LEÃO DE OLIVEIRA

PROJETO E DESENVOLVIMENTO DE APLICATIVO PARA AUXILIAR EM COMPRAS TECNOLÓGICAS

Trabalho de Conclusão de Curso apresentado à Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador (a):

Prof. Me. André Luiz Alves

Banca examinadora:

Profª. Ana Flávia Marinho

Prof. Vicente Paulo de Camargo

GOIÂNIA

2024

PAULO HENRIQUE LEÃO DE OLIVEIRA

PROJETO E DESENVOLVIMENTO DE APLICATIVO PARA AUXILIAR EM COMPRAS TECNOLÓGICAS

Trabalho de Conclusão de Curso aprovado em sua forma final pela Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, para obtenção do título de Bacharel em Engenharia de Computação, em _____/_____/_____.

_____
Orientador (a): Prof. Me. André Luiz Alves

_____
Profª. Ana Flávia Marinho

_____
Prof. Vicente Paulo de Camargo

GOIÂNIA
2024

RESUMO

À medida que a tecnologia avança, sua complexidade aumenta, tornando essencial entender não apenas como manuseá-la, mas também identificar o que realmente atende às nossas necessidades. Nesse contexto, surge a demanda por soluções que auxiliem pessoas menos familiarizadas com tecnologia na hora de adquirir produtos eletrônicos, como eletrodomésticos, de forma eficaz. Para abordar esse desafio, foi projetado e desenvolvido um aplicativo nomeado "MaaTECH". Baseado nos princípios da Engenharia de *Software*, esse programa inteligente utiliza *APIs (Application Programming Interface)* para buscar informações na *internet* e orientar os usuários na escolha dos melhores produtos, de acordo com suas preferências e requisitos. Além de facilitar a pesquisa, o *software* permite aos usuários gerenciar lista de compras personalizadas, adaptando-as conforme necessário. Com funcionalidades que simplificam o processo de compra, como a possibilidade de cadastro e autenticação para salvar listas, o “MaaTECH” destaca-se pela sua usabilidade intuitiva e pelo tempo economizado em grandes demandas.

Palavras-Chave: *APIs,* Eletrodomésticos, Engenharia de *Software,* Lista de Compras e “MaaTECH”.

ABSTRACT

As technology advances, its complexity increases, making it essential to not only understand how to use it but also to identify what truly meets our needs. In this context, there is a growing demand for solutions that assist individuals less familiar with technology in effectively acquiring electronic products, such as home appliances. To address this challenge, an application named "MaaTECH" was designed and developed. Based on Software Engineering principles, this intelligent program utilizes APIs (Application Programming Interfaces) to retrieve information from the internet and guide users in selecting the best products according to their preferences and requirements. In addition to streamlining the search process, the software enables users to manage personalized shopping lists, adapting them as needed. With features that simplify the purchasing process, such as registration and authentication for saving lists, "MaaTECH" stands out for its intuitive usability and the time it saves in managing extensive demands.

*Keywords: APIs, Household Appliances, Software Engineering, Shopping List e "*MaaTECH*".*

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface* (Interface de Programação de Aplicações)
CSS	*Cascading Style Sheets* (Folhas de Estilo em Cascata)
DBA	*Database Administrator* (Administrador de Banco de Dados)
HTML	HyperText Markup Language (Linguagem de Marcação de Hipertexto)
IA	Inteligência Artificial
IBM	*International Business Machines*
IBGE	Instituto Brasileiro de Geografia e Estatística
JSON	*JavaScript Object Notation* (Notação de Objeto JavaScript)
JVM	*Java Virtual Machine* (Máquina Virtual Java)
MaaTech	Me ajuda *Technology*
MVC	*Model-View-Controller* (Modelo-Visão-Controlador)
MVVM	*Model-View-ViewModel* (Modelo-Visão-Modelo de Visão)
REST	*Representational State Transfer* (Transferência de Estado Representacional)
RF	Requisito Funcional
RNF	Requisito Não Funcional
UML	*Unified Modeling Language* (Linguagem de Modelagem Unificada)
WEB	*World Wide Web* (Rede de Alcance Mundial)

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Atualmente, estamos em uma constante evolução no quesito da tecnologia, e os eletrodomésticos não ficam de fora. O site Escolhendo Bem é uma plataforma que orienta consumidores na escolha de eletrodomésticos e outros produtos para o lar, oferecendo dicas sobre comparações de preços, especificações técnicas e avaliações de clientes, além de sugerir boas práticas para compras online (Escolhendo Bem, 2024). De acordo com Rafael Costa, no portal Escolhendo Bem (Escolhendo Bem, 2024), cita que “Os eletrodomésticos são aparelhos que facilitam a nossa vida em casa. Eles surgiram ao longo da história, acompanhando o desenvolvimento da tecnologia”. No cenário atual, os eletrodomésticos estão com integração de microprocessadores e microcontroladores, tornando-os mais um aparelho próximo a um "computador". Tendo em vista essa evolução, que está bastante rápida, muitos que não são da área de tecnologia, ou até mesmo da área, estão com dificuldades de escolher o melhor aparelho para lhe atender.

Os eletrodomésticos têm as suas vantagens e características que tornam a vida das pessoas mais fácil. Eles trazem conforto nas atividades rotineiras mais práticas, como cozinhar e lavar. Segundo o Instituto Brasileiro de Geografia e Estatística (IBGE), em 2018, a geladeira estava presente no domicílio de 98,3% dos brasileiros (Exame, 2019). Esses aparelhos antes não tinham o fácil acesso até mesmo para pessoas que residiam na zona rural.

Entende-se que com a rápida evolução da tecnologia e a implementação delas nos eletrodomésticos, tornou-se mais complexo a descrição técnica dos aparelhos. Um exemplo, e referenciando novamente a geladeira, antes era utilizada apenas para gelar e manter frio os alimentos para não perder ou apenas refrescar algo. O portal Olhar Digital, é uma plataforma dedicada a divulgar os artigos sobre tecnologia. Em seu artigo “Geladeiras com tecnologia IA: para que servem?”, a escritora Ramana Rech cita sobre a tecnologia presente na geladeira *Bespoke AI Famly Hub* que possui uma câmera interna com inteligência artificial. Através desta inteligência artificial é possível indicar para os usuários, produtos com vencimento, criar lista de compras e lembretes dos alimentos armazenados (OlharDigital, [2024]).

Paralelamente a esse cenário, existe uma necessidade de adquirir esses eletrodomésticos inteligentes. Sendo assim, a proposta de projetar o desenvolvimento do *software* “MaaTECH”, para o propósito de aproximar o máximo possível a pessoa

ao seu interesse. Encontrar o aparelho doméstico inteligente que está buscando sem gastos desnecessários com funções que nunca serão utilizadas.

Este projeto de *software* apresenta os requisitos funcionais e não funcionais do *software*, bem como os testes de validação para os casos de uso. O programa foi desenvolvido usando a linguagem de programação *Java* com arquitetura *REST* utilizando o *framework SpringBoot* para o *backend*. No *frontend*, utilizamos *JavaScript* com o *framework React* para a interface *web* e a biblioteca de componentes *chakra-ui* para trabalhar com responsividade e estilização dos componentes, que foi construída em *HTML*.

O capítulo 2 abordará aspectos gerais sobre Engenharia de *Software*, trazendo um entendimento breve sobre essa devasta área para proceder com o projeto do *software*.

Em seguida, no terceiro capítulo, o levantamento do problema que gerou esse projeto de software e o levantamento de dados que validam o problema.

No quarto capítulo, é aplicado a teoria vista no segundo capítulo atendendo as solicitações feitas pelo “cliente”. É descrito toda a estrutura deste projeto de aplicação, desde a definição do banco de dados, até o *framework* utilizado no *frontend* para renderizar e criar as páginas.

No quinto capítulo, são expostas as considerações finais, abordando tanto os pontos positivos quanto os desafios enfrentados ao longo do desenvolvimento e implementação do projeto. Esse capítulo proporciona uma visão ampla do contexto geral, ressaltando os aprendizados adquiridos e sugerindo possíveis aprimoramentos para trabalhos futuros.

Por fim, o Capítulo 6 apresenta a conclusão, sintetizando as principais descobertas e os resultados alcançados durante o desenvolvimento do aplicativo MaaTech. Esse capítulo encerra o trabalho de forma coesa, destacando a relevância do projeto para a área de estudo e apontando caminhos para pesquisas futuras e potenciais melhorias.

Portanto, ao longo deste trabalho, é possível ver as etapas de um projeto de *software,* enxergando as etapas do projeto e as formas como são feitas para chegar ao resultado esperado.

2. ENGENHARIA DE SOFTWARE

De acordo com Ian Sommerville, *“engenheiros fazem as coisas funcionarem”* (Sommerville, 2013, p. 5), e não seria diferente para um engenheiro de *software*. Nesta engenharia, como o próprio nome já diz, pertence ao que se diz respeito de *software*. No entanto, ela pode se destrinchar em outras funções além do desenvolvimento de *software* em si. Em um projeto de *software* existe a necessidade de gerenciar o projeto, definir ferramentas que serão utilizadas, gerir uma equipe dependendo da complexidade do projeto.

Para entender melhor o que é *software*, o engenheiro de *software* Roger Pressman (Pressman, 2016), diz:

> Software consiste em: (1) instruções (programas de computador) que, quando executadas, fornecem características, funções e desempenho desejados; (2) estruturas de dados que possibilitam aos programas manipular informações adequadamente; e (3) informação descritiva, tanto na forma impressa como na virtual, descrevendo a operação e o uso dos programas.

Os engenheiros de software também sabem que devem atender sempre às diretrizes da empresa para qual irão realizar o projeto. A engenharia de software se preocupa em obter qualidade nos resultados requeridos, atendendo o prazo e o orçamento. Para se aproximar desta característica antes mencionada, existem várias formas de definir o que deve ser projetado. A mais usada em engenharia é o levantamento de requisitos.

2.1. Engenharia De Requisitos

Os requisitos são as partes mais importantes do projeto, conforme Sommerville (Sommerville, 2013), “Os requisitos são a base do desenvolvimento de sistemas de software”. Pressman (Pressman, 2016) reforça que “requisitos mal definidos ou incompletos são uma das causas principais de falhas em projetos de software”. É nele que será definido a funcionalidade total do produto. Por meio dele, desenvolve-se o que é importante para o produto atender as necessidades do usuário, o que é importante para o cliente, funcionalidades que devem ser implementadas para que seja atendido o pedido do cliente, quais funções a mais o programa deve ter para que funcione com melhor performance, e muitas outras características de suma importância para o projeto.

De acordo com Pressman (Pressman, 2016, p. 162/163), a engenharia de requisitos é importante para que todos os envolvidos no projeto, tenham o mesmo entendimento do problema descrito. Para Sommerville, no que diz ser engenharia de requisitos, que também pode ser conhecido como especificação de *software*, *"é o processo de compreensão e definição dos serviços requisitados do sistema e identificação de restrições relativas à operação e ao desenvolvimento do sistema"* (SOMMERVILLE, 2013, p. 24).

Nesse sentido, conforme descrito por Ian Sommerville (Sommerville, 2013), podemos identificar quatro etapas principais relacionadas à engenharia de requisitos:

a) **Estudo de viabilidade:** Busca resultados que se aproximem da satisfação das necessidades do usuário com tecnologias atuais de software e hardware. Aqui obtém-se o relatório de viabilidade.

b) **Elicitação e análise de requisitos:** Este processo é crucial, pois é nele que deriva os requisitos do sistema através da observação de outros sistemas existentes, também é possível obter esses requisitos através de discussões com potenciais usuários e compradores, análise de tarefas, entre outros métodos. Vale ressaltar que se for feito de malfeito, todo projeto será comprometido. Através dessa etapa, é possível obter os modelos de sistemas.

c) **Especificação de requisitos**: Neste processo, que depende totalmente da elicitação e análise de requisitos, é feito a tradução obtidas no processo anterior. Nesta etapa, internamente, será dividido em dois tipos de requisitos. Requisitos que são definidos pelos usuários e que são requisitos abstratos do sistema para o cliente e usuário, chamados de requisitos do usuário. E por fim, os requisitos do sistema que são observações mais detalhadas sobre as funcionalidades a serem providas. A partir desta etapa, obtém os requisitos de usuários e de sistemas.

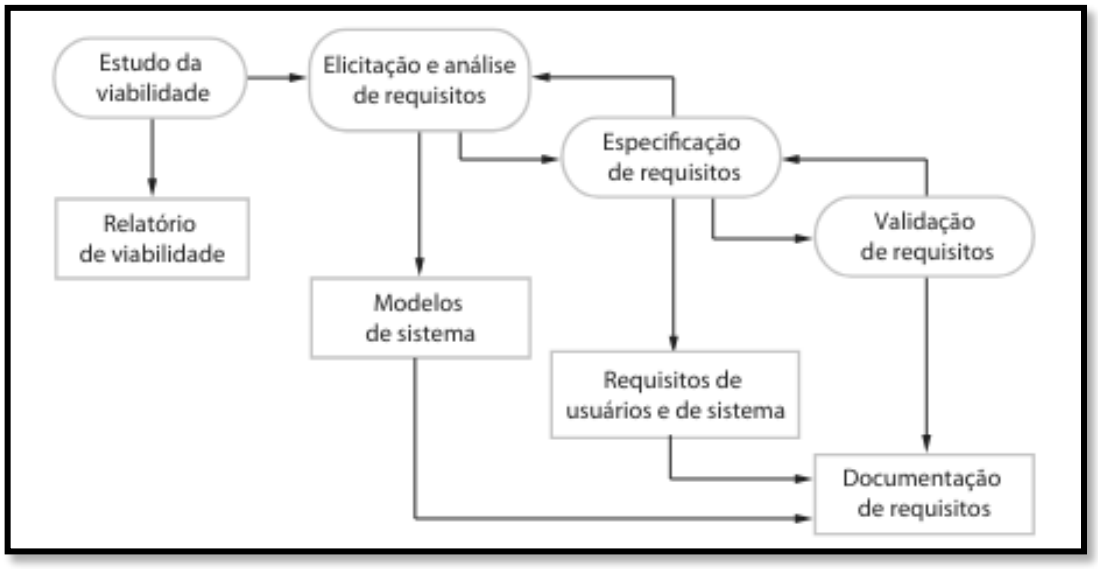
d) **Validação de requisitos**: Nesta etapa os requisitos são testados e então validados caso passem nos testes. No pior caso, deve corrigir para que atenda o requisito em que tenha falhado. Este processo é o último que fomenta o documento final que é a Documentação de requisitos.

A figura 1 demonstra como é feita a atividade do processo. Ele começa no estudo da viabilidade, com esse estudo é possível obter o relatório de viabilidade. Depois, o processo vai para Elicitação e análise de requisitos, que por meio deste têm os modelos de sistema que influenciam na documentação de requisitos.

O próximo passo é a especificação de requisitos que gera os requisitos de usuários e de sistemas, outro que influencia na documentação de requisitos. E para finalizar, a validação de requisitos, que é o último item que compõe a documentação de requisitos.

Da etapa de licitação de análise de requisitos até a validação de requisitos, é possível voltar e entrar em um ciclo até os requisitos ficarem bem formulados. Afinal, a parte mais importante em um projeto e que também vai determinar o restante, é o levantamento de requisitos e sua documentação.

Figura 1 – Atividades do Processo



Fonte: SOMMERVILLE, 2013

Na engenharia de software, o processo de levantamento de requisitos é organizado em dois tipos principais: **requisitos funcionais**, que detalham as funcionalidades específicas que o sistema deve executar para atender às necessidades dos usuários, e **requisitos não funcionais**, que estabelecem restrições e parâmetros de qualidade, como desempenho, segurança, usabilidade e compatibilidade, garantindo que o sistema opere de forma eficiente e confiável (SOMMERVILLE, 2013).

i. Requisitos Funcionais

Os requisitos funcionais, são aqueles que devem constar no programa e que fazem o *software* funcionar, atendendo o que se foi pedido. Muitas vezes é confundido o requisito funcional com o não funcional por serem parecidos. Mas é importante entender que o requisito funcional é o que vai definir se o programa foi feito corretamente.

Embora comentando anteriormente, o requisito funcional pode ser derivado em requisito de usuário e requisito de sistema. De acordo com Sommerville (Sommerville, 2013, p. 58), *"os requisitos de usuário foram feitos para serem compreendidos pelos usuários do sistema, por essa razão, eles são mais abstratos."* Já os requisitos de sistema funcionais são mais detalhados. Eles são feitos de forma que devem ser compreendidas suas funções nos sistemas, suas entradas e saídas, exceções etc.

A tabela 1 é um requisito funcional que está presente no apêndice A. Por meio deste exemplo fornecido, é possível ter uma noção de como foi feito e separado cada requisito funcional. O campo "Nrº do Requisito" faz referência ao número do requisito que está sendo descrito na tabela, a "Classificação" define qual requisito, se é funcional, não funcional ou outros tipos que podem ser incluídos futuramente, "Descrição" que é o detalhamento do que é aquele requisito, "Justificativa" o motivo de existir este requisito, "Origem do requisito" de onde surgiu a necessidade do requisito, "Pré-requisito" que são os requisitos devem ser avaliados antes do presente requisito, "Prioridades" define a importância do requisito e o impacto que pode causar no projeto, "Conflitos" quais impactos causam no projeto.

Exemplos de Tabela de Requisitos Funcionais:

Tabela 1 – Exemplo de Requisitos Funcionais

Nrº do Requisito:	RF004
Classificação:	Requisito Funcional (RF)
Descrição:	Após a autenticação, o sistema deve permitir que o usuário visualize e edite suas informações de perfil, como nome, e-mail e preferências.
Justificativa:	Facilitar a gestão de dados pessoais e preferências do usuário.
Origem do requisito:	Necessidade de personalização e gerenciamento de contas.
Critério de Aceitação:	O usuário autenticado deve poder acessar e editar suas informações de perfil sem dificuldade.
Pré-requisito:	RF003
Prioridades:	Média
Conflitos:	

Fonte: autor

ii. Requisitos Não Funcionais

Os requisitos não funcionais desempenham um papel importante no desenvolvimento de software, pois determinam como o sistema deve operar, complementando os requisitos funcionais, que descrevem o que o sistema deve fazer. De acordo com Sommerville (2013), os requisitos não funcionais se referem a restrições que afetam o funcionamento do sistema, como desempenho, segurança e confiabilidade, sem estarem diretamente ligados a funcionalidades específicas. A ausência de atenção a esses requisitos pode comprometer a utilidade do sistema. Por exemplo, um requisito funcional pode falhar, mas uma falha em um requisito não funcional pode tornar o sistema completamente inutilizável.

Ian Sommerville define que os requisitos não funcionais *"não estão diretamente relacionados com os serviços específicos oferecidos pelo sistema a seus usuários"* (Somerville, 2013, p. 60). De acordo com ele, tais requisitos podem afetar a arquitetura geral de um sistema em vez de apenas os componentes. Isso ocorre em razão deles serem mais críticos do que os requisitos funcionais individuais. Os requisitos não funcionais *"podem estar relacionados às propriedades emergentes do sistema, como confiabilidade, tempo de resposta e ocupação de área"* (Sommerville, 2013, p. 60).

A tabela 2 é um requisito não funcional que está presente no apêndice A. Por meio deste exemplo fornecido, é possível ter uma noção de como foi feito e separado cada requisito funcional. A tabela segue o mesmo modelo descrito para a tabela 1. As informações descritas na tabela são importantes para entender o que o sistema espera quanto aos seus requisitos não funcionais. Desta forma, tanto a tabela 1 quanto a tabela 2 seguem esse padrão para não deixar despercebido nenhum dado. Assim, quando o leitor estiver estudando a documentação dos requisitos ele consiga identificar sem dificuldade alguma os mesmos detalhes do requisito. Tornar um padrão no projeto de forma que seja possível identificar sem dificuldades os dados necessários, é um ponto interessante. Nesse sentido, os dados mais importantes permanecem na tabela, separando sua titulação a esquerda e sua descrição a direita.

Se realizar uma análise minuciosa quanto as duas tabelas, será identificado apenas diferenças de texto, ou melhor, serão identificados apenas as descrições dos campos distintos de um tipo de requisito (requisito funcional) para outro requisito (requisito não funcional).

Exemplo de Tabela de Requisito não Funcional:

Tabela 2 – Exemplo de Requisitos Não Funcionais

Nrº do Requisito:	RNF001
Classificação:	Requisito Não-Funcional (RNF)
Descrição:	O sistema deve seguir padrões de usabilidade que garantam facilidade de uso, mesmo para pessoas com pouca experiência em tecnologia.
Justificativa:	Facilitar o uso do sistema por todos os tipos de usuários, independentemente de sua familiaridade com tecnologia.
Origem do requisito:	Necessidade de acessibilidade e simplicidade no uso do sistema.
Critério de Aceitação:	O sistema deve ser facilmente navegável e intuitivo para todos os usuários, com feedback positivo em testes de usabilidade
Dependências:	RF001, RNF003.
Prioridades:	Alta
Conflitos:	

Fonte: autor

2.1.1. Levantamento de Requisitos

Para realizar o levantamento dos requisitos de um *software*, existem diversas formas de como pode ser feito. Desde entrevistas, questionários, prototipagem e outros. No caso, o ideal seria definir um modelo de sistema que o projeto de software vai seguir. Conforme Sommerville (Sommerville, 2013), os modelos são quase sempre baseados em notações de UML do inglês *Unified Modeling Language*). Esses modelos auxiliam no processo de engenharia de requisitos para extrair os requisitos do sistema.

Para este projeto de *software*, foi feito um protótipo e através deste, fez-se o levantamento de requisitos. Para realizar o levantamento foi necessário criar um protótipo que por meio dele, gerou a sensibilidade das necessidades existentes para o software. O protótipo, gerado a partir de uma história, possibilitou classificar os requisitos funcionais e não funcionais.

2.2. Projeto de software

Um projeto de *software* é a projeção de como será desenvolvido um *software* solicitado pelo cliente para o qual o engenheiro foi contratado. Para este caso, é um projeto de aplicação para desenvolvimento de um aplicativo que auxilia nas compras tecnológicas.

> “Modelagem de sistema é o processo de desenvolvimento de modelos abstratos de um sistema, em que cada modelo apresenta uma visão ou perspectiva, diferente do sistema” (SOMMERVILLE, 2013, p. 82).

Todo projeto de aplicativo exige um processo de *software*, processo que justamente leva a produção do produto do *software*. Diante disso, Sommerville relata:

> Um processo de software é um conjunto de atividades relacionadas que levam à produção de um produto software. Essas atividades podem envolver o desenvolvimento de software a partir do zero em uma linguagem padrão de programação como Java ou C (SOMMERVILLE, 2013, p. 18).

Existem diversas etapas em um processo de *software*, são eles: especificação de *software*, projeto e implementação de *software*, validação de *software* e evolução de *software*.

No projeto de software, é designada também a metodologia que será utilizada para o desenvolvimento. Através desta metodologia, e se for bem aplicada, ela facilitará a produção da aplicação. Dependendo da metodologia, ela será aplicada a projetos mais complexos ou a projetos mais simples, mas não perde a sua importância. A definição clara da metodologia é necessária, pois ela pode agilizar a produção do software, identificar erros e auxiliar os membros da equipe no desenvolvimento conjunto do projeto. Como ressaltado por Sommerville, a escolha da metodologia correta tem um papel crucial no sucesso do projeto, pois ela pode influenciar diretamente a eficiência do processo e a qualidade do produto final (Sommerville, 2013).

iii. Metodologia de Desenvolvimento Adotada

Embora o desenvolvimento de software possa parecer simples, não é apenas uma questão de abrir uma interface de desenvolvimento e começar a escrever um algoritmo do zero. Antes de mais nada, é crucial definir qual metodologia será adotada. Existem várias opções, e algumas das mais conhecidas são o modelo em cascata, o modelo espiral e o modelo incremental, cada uma com suas próprias características

e adequações a diferentes tipos de projeto. O modelo cascata, por exemplo, segue uma sequência linear de fases, enquanto o modelo espiral envolve ciclos de desenvolvimento com foco em avaliação de riscos. Já o modelo incremental se baseia na entrega do sistema em partes, com funcionalidades prioritárias sendo entregues nas primeiras versões (Sommerville, 2013).

Em concordância com o ditado popular brasileiro, “cada caso é um caso”, salienta-se que para cada tipo de projeto existe um modelo de processo mais adequado. Não existe o “modelo melhor de todos”, mas sim o “modelo melhor para este projeto de software”. Cada modelo possui vantagens e desvantagens que devem ser consideradas ao ser adotado, dependendo das necessidades específicas do projeto (Sommerville, 2013).

Após uma avaliação com outros métodos, chegamos à conclusão de que o modelo espiral é o ideal para este projeto. As vantagens que ele apresenta são superiores às desvantagens.

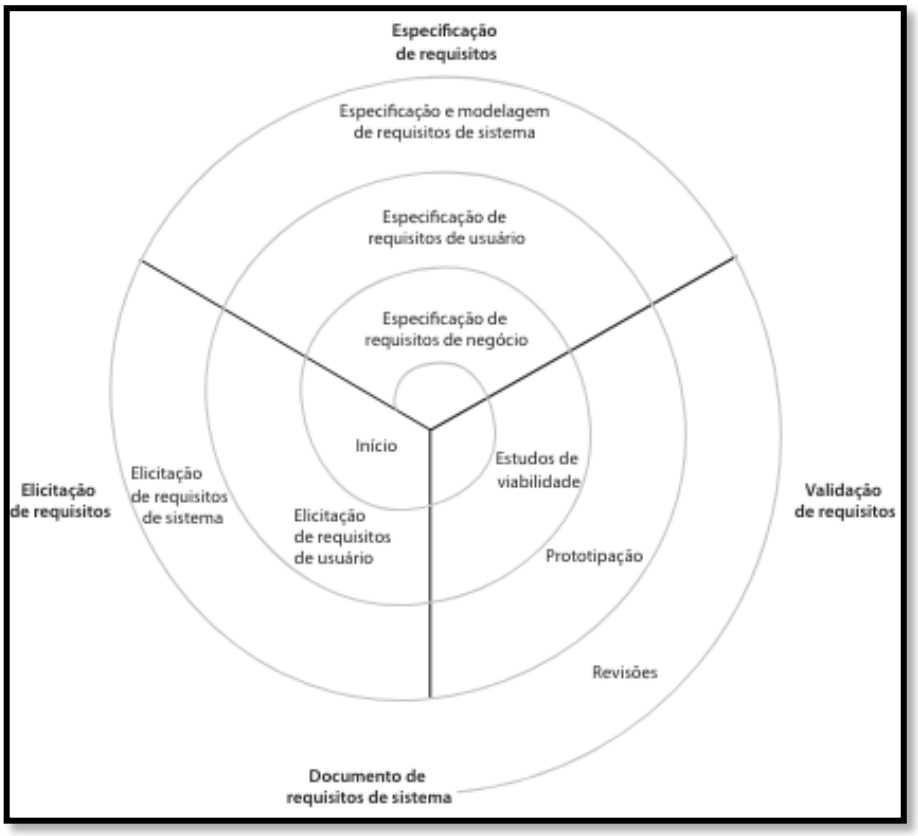
A figura 2 é uma breve representação do que seria um modelo espiral. Entende-se que na primeira etapa existe a definição dos objetivos. Estes podem ser levantados conforme a necessidade, ou então o desenvolvimento de um novo requisito. A avaliação e redução de riscos ocorre depois que os objetivos são elaborados.

Vale destacar que esses passos ocorrem todas as vezes que o espiral retornar ao quadrante. Se tudo ocorrer bem, vai para a implementação e validação do que foi levantado. Este é o terceiro passo. Por último, os planejamentos e as especificações do que foi implementado. De acordo com o modelo espiral de desenvolvimento de software descrito por Sommerville (Sommerville, 2013), as fases são iterativas e os passos são repetidos a cada ciclo da espiral. O terceiro passo envolve a validação das implementações, enquanto o último consiste no planejamento para o próximo ciclo, assegurando que o projeto evolua de maneira contínua e ajustada às necessidades.

O modelo espiral, descrito por Sommerville (2013), é uma abordagem iterativa que combina desenvolvimento incremental e análise de riscos, destacando-se por sua adaptabilidade. Cada ciclo inclui planejamento, análise, implementação e validação, garantindo que ajustes sejam feitos conforme necessário. De acordo com o autor, essa metodologia é ideal para projetos de grande porte, pois permite verificar e validar os resultados em cada iteração. Comparado ao modelo cascata, que é linear, e ao incremental, que entrega partes funcionais em etapas, o modelo espiral se mostra

mais completo. Além disso, por incluir elementos do incremental, ele oferece a flexibilidade de adaptar diferentes modelos conforme os riscos e requisitos emergem (Sommerville, 2013). No projeto em questão, essa característica foi essencial, sendo possível utilizar o modelo incremental em uma etapa específica, maximizando sua eficácia.

Figura 2 – Representação do Modelo Espiral



Fonte: SOMMERVILLE, 2013.

Embora o modelo especificado seja melhor para sistemas de grande porte, ele é um dos melhores em questão de verificação e validação. Outra opção seria o modelo incremental, mas como o espiral possui também o incremental nele, não tem o porquê de escolher ele.

Para o processo de implementação e validação, como o modelo espiral é uma evolução do modelo incremental e que também é uma evolução do modelo em cascata, nesta fase deve-se escolher qual melhor modelo a ser utilizado nesta etapa do modelo espiral. Por exemplo, fez-se a primeira “volta” no modelo espiral, voltamos então às soluções dos novos riscos. Qual modelo seria ideal para solucionar este problema?

No nosso caso, o modelo escolhido foi o modelo incremental, tendo em vista que o sistema permanece em funcionamento e é possível gerenciar qual etapa estamos no próprio modelo através de versões.

2.3. Construção de *Software*

A construção do software começa desde a concepção da ideia do produto. Após o processo de idealização, levantamento dos requisitos e definição do modelo, inicia-se a codificação do software. Esta fase é especificamente destinada aos desenvolvedores, analistas e outros profissionais que utilizam linguagens de programação para implementar o sistema solicitado pelo cliente. Durante essa etapa, a equipe de desenvolvimento transforma os requisitos e o modelo definido em um código funcional, dando início à criação do produto final. De acordo com Sommerville (Sommerville, 2013), o processo de desenvolvimento de software é iterativo e evolutivo, onde as fases como a especificação, projeto, implementação e validação podem ser repetidas à medida que os requisitos se desenvolvem durante o ciclo de vida do software.

Deve-se levar em consideração que o código pode ser desenvolvido por uma ou mais de uma pessoa, podendo ser uma equipe de desenvolvimento.

Uma breve ideia de como será o software em sua funcionalidade pode ser feita através de um protótipo. Desta forma, o cliente tem sua visão crítica antes de iniciar o processo de desenvolvimento e facilita na ideia do que desenvolver e por onde começar a desenvolver.

Além disso, a fase de construção deve considerar boas práticas de codificação, como reutilização de código, modularidade e clareza, que são fundamentais para facilitar a manutenção e a evolução do software. De acordo com Sommerville (2013), essa etapa exige uma abordagem disciplinada, garantindo que o código desenvolvido esteja alinhado aos requisitos e padrões de qualidade estabelecidos. Sommerville (2013) também destaca que ferramentas como o Git desempenham um papel importante no controle de versão, permitindo o gerenciamento eficaz de mudanças no código, especialmente em equipes maiores. Essas ferramentas promovem a colaboração entre os desenvolvedores e oferecem rastreabilidade das alterações realizadas. Dessa forma, a construção do software se torna não apenas uma etapa

de execução, mas um processo estratégico para garantir a sustentabilidade do sistema ao longo do tempo.

A figura 3 é um exemplo de como poderá ser a tela inicial do *software* MaaTECH.

Figura 3 – Tela Inicial MaaTECH

Fonte: Autor

3. ESTUDO DE CASO

O problema de compras acontece quando pessoas tentam encontrar o produto ideal. Muitas vezes, quando se trata de um produto onde ela não sabe nada a respeito, ou não sabem por onde começar a buscar, ou então, não sabem se o mínimo produto o atende.

3.1. Causa

As causas para a dificuldade de compra englobam diversos fatores, tais como desconhecimento do produto ou de suas tecnologias. A *Orbit Data Science* é uma empresa especializada na análise de dados sociais, utilizando inteligência artificial e ciência de dados para gerar *insights* sobre o comportamento dos consumidores. Segundo a empresa, ela "coleta dados das redes sociais e outras fontes digitais, aplicando modelos quantitativos para identificar tendências e entender melhor as percepções e preferências do público" (*Orbit Data Science*, 2024). No artigo “Motivos pelos quais Brasileiros Deixam de Comprar certos Produtos/Marcas”, publicado pela *Orbit,* diz que foram capturados 3532 *tweets* alegando insatisfação quanto a algum produto. *Twitter*, atualmente chamado de X, é uma rede social onde as pessoas interagem através de *tweets*. São mensagens rápidas compartilhando uma ideia ou alguma novidade. Esse conteúdo capturado pela *Orbit* em 2023, relacionava todos os resultados com palavras como “não compro mais”, “parei de comprar”, “não compre” e “não comprem”. De acordo com o artigo *“52% das pessoas acima de 60 têm dificuldades de achar produtos”* publicado pela Amanda Schnaider, no blog meio&mensagem, comenta sobre a dificuldade das pessoas mais velhas em adquirir produtos generalizados.

> *Mais da metade deste público — 52% dos entrevistados — afirma sentir dificuldade em encontrar produtos e serviços que atendam às suas necessidades, enquanto 72% dizem que as lojas não estão preparadas para lidar com a longevidade. (SCHNAIDER, 2019)*

O problema enfrentado pelos usuários é definido pela falta de conhecimento técnico sobre as especificações dos produtos, incertezas quanto às necessidades reais versus os recursos oferecidos, dificuldade em comparar produtos de diferentes marcas e categorias, entre outros fatores. De acordo com o artigo "*Mais de 10 milhões*

de brasileiros não acessam a internet por não saberem usar a tecnologia, diz IBGE", de Guilherme Gama, publicado no portal da CNN, "*em 2023, cerca de 12% da população brasileira com idade a partir de 10 anos não usava internet – o equivalente a 22,4 milhões de pessoas*". Nesse cenário, a ausência de conhecimento técnico sobre as especificações dos produtos tende a se tornar um obstáculo significativo para esses usuários.

Essa limitação interfere diretamente na capacidade de identificar se os recursos de um produto atendem ou não às suas necessidades reais. Muitas vezes, o consumidor não compreende que um produto pode suprir suas expectativas ou, ao contrário, ser insuficiente para suas demandas específicas.

Além disso, existe uma dificuldade em comparar produtos de diferentes marcas e categorias. Por exemplo, um produto pode oferecer as mesmas funcionalidades de outro, mas a um custo mais acessível. Dennis Ventapane, no portal Escolha Ideal (escolhaideal, 2024), site EscolhaIdeal.org que fornece análises detalhadas de plataformas de ensino online, ajudando os usuários a comparar diferentes opções, como Udemy e Alura, para decidir qual delas oferece a melhor relação custo-benefício para suas necessidades de aprendizagem e desenvolvimento profissional, cita sobre essa dificuldade em compras online “Hoje, a maior praticidade do e-commerce promoveu um aumento das lojas, abrindo espaço para novas marcas. Com isso, dificilmente você encontra um segmento monopolizado por uma única empresa, sendo possível contar com uma boa variedade de opções “. Entretanto, devido à falta de compreensão, muitas pessoas acabam adquirindo itens que apresentam menor durabilidade ou não correspondem ao custo-benefício esperado.

3.2. Análise Técnica

A análise técnica dos produtos é feita, majoritariamente, por profissionais da área de tecnologia. São acionados para fazer uma recomendação dos produtos e até mesmo cotados para realizar orçamentos e compras.

3.3. Possíveis soluções

Para escolher um produto, sendo uma pessoa leiga quanto ao conhecimento da mercadoria interessada, existem algumas técnicas que é realizada até mesmo pelos próprios técnicos, mas que levam um tempo até obter o resultado.

a) **Buscar *feedbacks* em vídeos no *YouTube*:** Esse é um método realizado por todos, pois nele temos um contato verbal com o influenciador que está apresentando o seu ponto de vista. O problema é quando o avaliador está sendo patrocinado e pode acabar não sendo leal à sua avaliação.

b) **Notas de avaliações**: As notas de avaliações são essenciais para destacar se a compra compensa ser processada ou não. Através dessa nota, é possível, mesmo sem ter um conhecimento muito alto, considerar a compra ser suficiente ou não.

c) **Pesquisas em sites de tecnologia**: As pesquisas em sites de tecnologia são mais para o público leigo. Esses buscam os sites para se inteirarem melhor do assunto e assim conseguir definir ao certo o que precisa procurar. O problema nessas pesquisas é que demandam bastante tempo e na maioria das vezes acabam procurando a opinião de um técnico.

O referente trabalho não dispensa o serviço de um técnico, tampouco diminui sua competência. Recomenda-se para quem esteja manuseando este programa, caso não tenha conhecimento do produto, procurar um profissional para um melhor aconselhamento. O objetivo principal deste trabalho é otimizar o tempo de busca dos produtos que esteja interessado.

3.4. Estratégia para auxiliar na procura de mercadorias que necessitam de um maior conhecimento técnico

Existem formas de realizar essa busca e chegar ao seu resultado esperado. Mas para isso, exige-se um tempo de busca para entender o que o produto oferece, as especificações daquele produto, qual a vantagem de um e qual a vantagem de outro, tempo que muitos as vezes não possuem.

3.5. Implementação das técnicas no software “MaaTech”

No contexto do aplicativo “MaaTech”, a implementação das técnicas como, busca de *feedbacks*, busca de produtos mais recentes e ou novos e busca de avaliações, serão paralelamente integradas. Os usuários poderão realizar buscas dos

produtos interessados e obter diversos produtos, mas que o atendam de forma mais precisa.

Essa abordagem possibilita otimizar o tempo de busca quando o usuário já sabe qual produto quer e então buscar mais sobre o produto específico. A técnica realizada pelo “MaaTech”, nada mais será com o auxílio de IA (Inteligência Artificial) para realizar essa busca.

4. PROPOSTA DE SOLUÇÃO

O desenvolvimento da aplicação web MaaTech foi concebido para oferecer suporte especializado a indivíduos que enfrentam dificuldades na busca por produtos tecnológicos ou que precisam adquirir itens fora do seu conhecimento habitual. Reconhecemos que a área de tecnologia está em constante evolução, e, portanto, o MaaTech precisa ser alinhado com as tendências mais recentes para proporcionar uma experiência otimizada ao usuário.

Dentro desse contexto, uma das inovações implementadas no MaaTech é a utilização de Inteligência Artificial para otimizar a busca por produtos. Em vez de realizar a busca manualmente, o processo é terceirizado para a IA, que se encarrega de refinar as opções de produtos, priorizando os mais atualizados e que atendem às expectativas do usuário. Esse processo é facilitado por um formulário que o usuário preenche, enviando as informações para uma API de IA. A partir disso, o usuário recebe uma lista de produtos com as características e especificações detalhadas, permitindo uma melhor compreensão dos itens, de forma personalizada e eficiente.

No desenvolvimento da aplicação, a arquitetura foi pensada para garantir uma navegação intuitiva e a eficiência do sistema. A estrutura da aplicação, com foco na usabilidade, foi planejada para fornecer uma experiência otimizada, destacando os elementos da interface e funcionalidades essenciais para o bom funcionamento do MaaTech. Além disso, foram integrados recursos específicos para o gerenciamento da plataforma, assegurando que as necessidades dos usuários fossem atendidas de forma precisa e eficiente.

Este capítulo fornece uma visão abrangente sobre o processo de desenvolvimento do MaaTech, discutindo as decisões estratégicas tomadas durante o desenvolvimento *fullstack* da aplicação e destacando como a combinação de *design*, usabilidade e IA contribui para uma experiência mais satisfatória para os usuários.

4.1. Prototipação

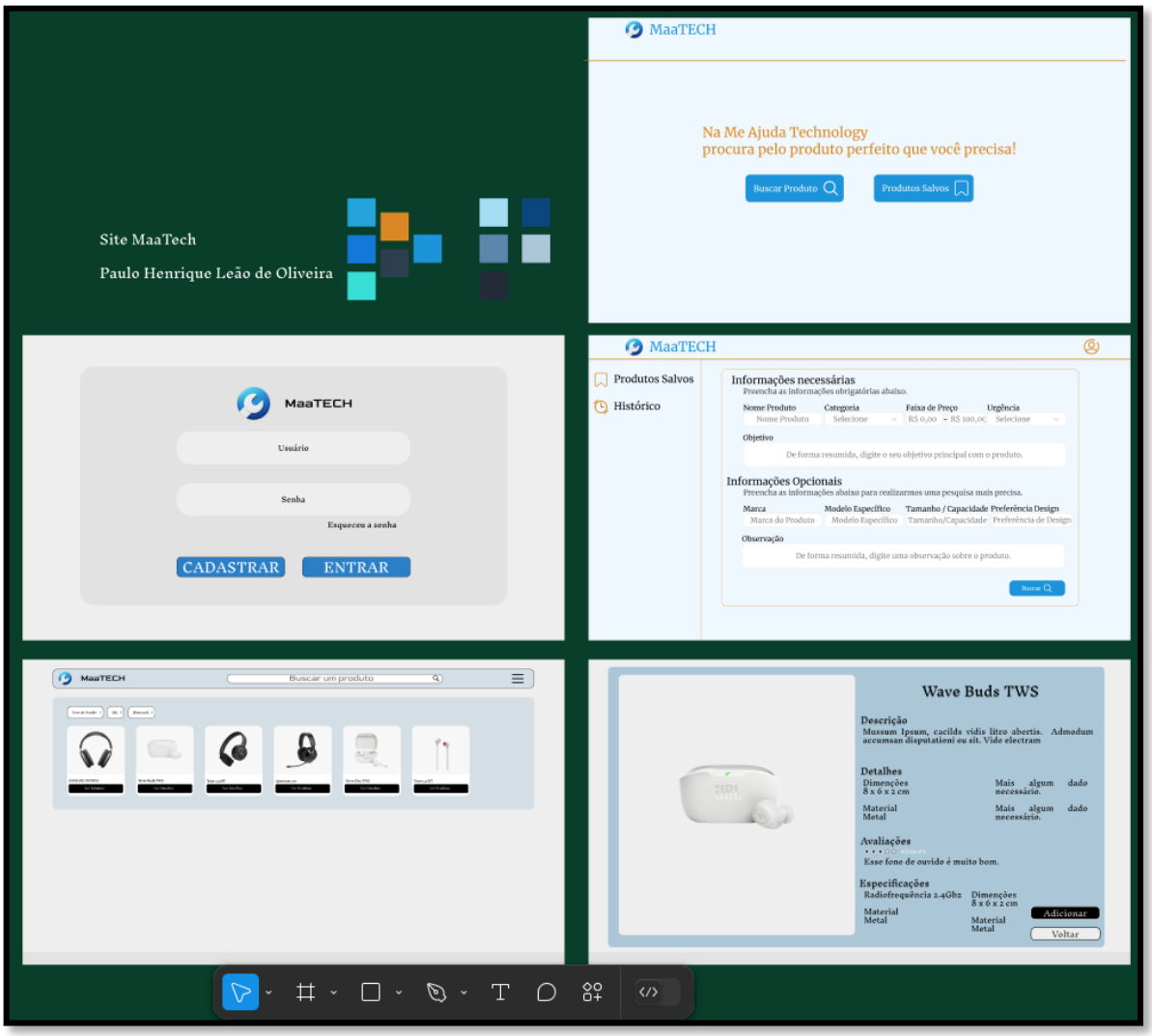
O Figma® é uma ferramenta colaborativa baseada na *web*, que oferece recursos avançados para criar interfaces de usuário e protótipos interativos. De acordo com o portal do próprio Figma, se auto descrevem como uma ferramenta de design acessada através da *web* com funcionalidade de aplicação nativa. Pode ser acessado

através do *link* www.figma.com.br. Focada na colaboração em tempo real, a plataforma permite que *designers* e equipes trabalhem simultaneamente, independentemente da localização geográfica.

Durante o desenvolvimento da interface do usuário, os protótipos desempenham um papel fundamental na concretização e avaliação das ideias propostas. Criados com a ferramenta de *design* Figma® (Figma, 2023), esses protótipos são versões interativas e visuais das interfaces planejadas. Eles não apenas ilustram o layout e o design, mas também permitem simular a experiência do usuário por meio de interações específicas.

A Figura 4 apresenta os protótipos inicialmente projetados para a aplicação MaaTech, ilustrando as representações interativas das interfaces propostas.

Figura 4 – Ferramenta Figma



Fonte: Autor

O protótipo foi criado simulando a necessidade do cliente. Criou-se uma história como se um cliente estivesse solicitando o programa e definindo como deveria ser a visualização e as necessidades do cliente.

O Figma permite que os *designers* criem e editem *designs* diretamente no navegador, sem a necessidade de instalar *software* adicional. A plataforma é compatível com o design de interfaces para diversos dispositivos, como *desktop*, *mobile* e *web*, facilitando a adaptação de projetos a diferentes tamanhos de tela e contextos de uso.

Uma das principais vantagens do Figma é a sua capacidade de permitir que múltiplos usuários colaborem em um projeto simultaneamente, visualizando e editando *designs* em tempo real. Isso torna o trabalho em equipe mais ágil, favorecendo a comunicação eficiente e a revisão contínua do projeto.

Além disso, o Figma oferece recursos avançados de prototipagem, permitindo aos *designers* criar protótipos interativos que demonstram o fluxo e a funcionalidade de aplicativos ou sites. Esses protótipos podem ser compartilhados e testados por usuários para coletar *feedback*.

Com sua abordagem inovadora e recursos robustos, o Figma tornou-se uma ferramenta popular entre *designers* e equipes de design que buscam criar interfaces atraentes, protótipos interativos e colaborar de forma eficiente durante todo o processo de design (Figma, 2023).

4.2. Desenvolvimento de Aplicações web

A ideia é permitir que o usuário acesse a plataforma de qualquer lugar, sem dificuldades. Com isso em mente, optamos por desenvolver o MaaTech como uma aplicação *web*, evitando problemas de compatibilidade com sistemas operacionais, como *Android*, e garantindo o acesso por qualquer dispositivo com conexão à internet e um navegador. Após definir o formato da aplicação, escolhemos as ferramentas que seriam utilizadas.

A aplicação MaaTech foi desenvolvida utilizando duas linguagens de programação: *Java* e *TypeScript*. Cada uma foi designada para atender a diferentes aspectos do projeto.

4.2.1. Backend

O *Java* foi escolhido para o desenvolvimento do *backend*, por ser uma linguagem de alto nível com uma comunidade ampla e ativa, que oferece suporte e recursos valiosos. Além disso, é uma linguagem em constante evolução, consolidada como uma das mais utilizadas no mundo.

A DevMedia é uma plataforma online que oferece uma ampla gama de conteúdos educacionais voltados para o aprendizado de programação e desenvolvimento de software, incluindo cursos, artigos e tutoriais para profissionais de TI e estudantes da área. De acordo com o artigo "Por que *Java*?" publicado no portal *DevMedia*, *Java* vai além de ser apenas uma linguagem de programação, sendo considerado também uma plataforma de desenvolvimento. Sua versatilidade e integração com diversas interfaces específicas tornam o desenvolvimento mais prático. Além disso, o *Java* possui bibliotecas e ferramentas que facilitam a criação de código, como o *Maven*.

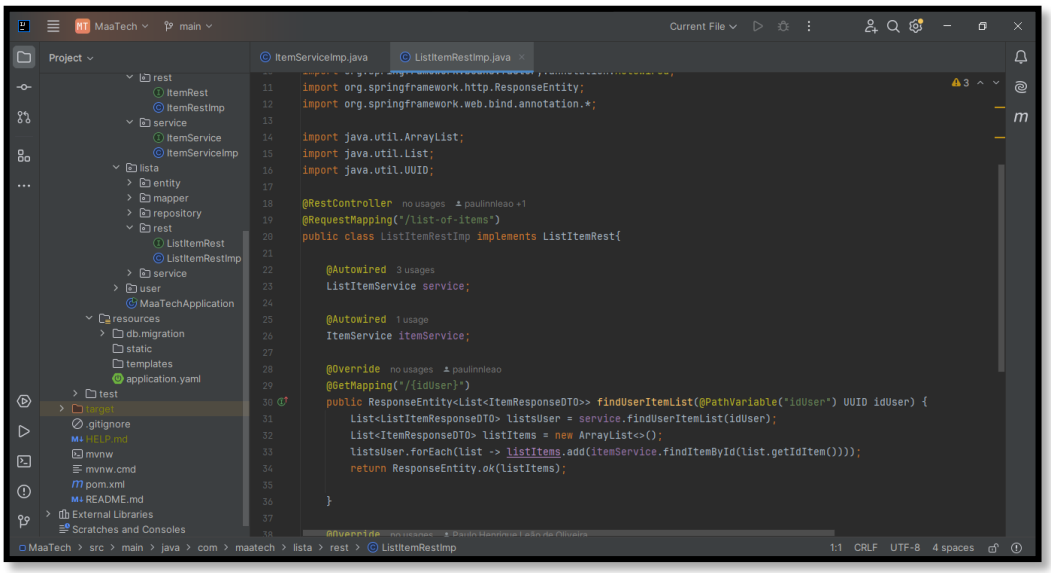
O *Maven* é um gerenciador de dependências para projetos *Java*. No portal *iMasters*, plataforma online que oferece conteúdos e recursos técnicos voltados para profissionais de tecnologia, promovendo o compartilhamento de conhecimento e a troca de experiências na comunidade de tecnologia da informação, Diego Pacheco escreveu o artigo "Dez motivos para você usar *Maven*", que, embora não seja recente, permanece relevante. O Maven é uma ferramenta crucial no processo de construção de aplicações, sendo responsável pela automação da construção do *software*. Ele permite a integração e configuração de parâmetros, além de gerenciar eficientemente a injeção de dependências, garantindo que todas as bibliotecas necessárias sejam baixadas e organizadas corretamente. Isso facilita a padronização no ambiente de desenvolvimento, assegurando que o projeto seja construído de forma consistente em diferentes plataformas sem que o desenvolvedor precise se preocupar manualmente com as dependências (SOMMERVILLE, 2013; PRESMAN, 2016).

Para finalizar a composição do *backend*, é necessário definir qual *framework* realizará a autoconfiguração, buscar as dependências e instalá-las ao projeto. A IBM (*International Business Machines Corporation*) é uma das maiores empresas de tecnologia do mundo, reconhecida por suas inovações em inteligência artificial, computação em nuvem, *blockchain* (tecnologia de registro distribuído que organiza dados em blocos interligados) e consultoria em tecnologia da informação. No site da

IBM eles definem *spring* como “Um *framework* bastante conhecido de nível empresarial, de *software* livre, para criar aplicativos independentes de nível de produção que são executados na *Java Virtual Machine* (JVM)”. No entanto, existem outros que entregam a mesma funcionalidade, por exemplo o *Quarkus,* A escolha do *spring* foi devido ao fato que o *Quarkus* é mais recente e muitas bibliotecas já foram desenvolvidas para o *spring*.

A figura 5 ilustra o *backend* com *Java.* É possível ver na imagem as anotações que fazem parte do configurações feitas no *backend* e a separação/organização das entidades, modelos de serviços, controle e manipulação no banco de dados. Em específico, o trecho abordado é um exemplo de como é feita a lógica de busca da lista de produto de um usuário. Espera-se o identificador do usuário “idUser” para seguir a lógica de busca presente no código.

Figura 5 – *Backend Java*



Fonte: Autor

4.2.2. Frontend

Para o *frontend* foi escolhido *TypeScript* que basicamente é a linguagem *JavaScript* com tipagem, o que facilita o desenvolvimento evitando erros inesperados. “*TypeScript* é basicamente uma versão turbinada do bom e velho *JavaScript*” diz Bruno Braga no portal *Orango.dev*. Ele aponta que *TypeScript* é um superconjunto de *JavaScript.* Ou seja, tudo que existe em *JavaScript,* existem também em *TypeScript*. Quando ocorre um erro em JavaScript, você o recebe durante o tempo de execução

pois se trata de uma linguagem interpretada. Vantagem que o *TypeScript* traz, pois ele compila o código antes de transpilar (o ato de converter uma linguagem para outra, mas para linguagens do mesmo nível de abstração) para *JavaScript*, evitando assim erros que poderiam ser vistos durante a execução do programa. Como *TypeScript* compila, caso haja algum erro semântico, ele identifica e apresenta o erro.

O motivo principal para utilizar *TypeScript* é devido a sua fácil manipulação. No artigo Noções básicas de *JavaScript*, do portal D*eveloper.mozilla,* plataforma de recursos e documentação voltada para desenvolvedores web, fornecendo tutoriais, guias e referências sobre tecnologias como HTML, CSS, JavaScript e APIs web*,* diz que *JavaScript* é uma linguagem de programação que adiciona interatividade ao site, relativamente compacto, mas muito flexível, criação dinâmica de *HTML* e definição de estilos *CSS*, *APIs* de terceiros que permitem aos desenvolvedores incorporar funcionalidades em sites. Enfim, é uma poderosa linguagem de programação.

Para completar esse time de ferramentas utilizados no *frontend* o *REACT* foi selecionado minuciosamente devido a sua alta compatibilidade com outras versões. Uma desvantagem que existe no *Angular*. Esses são *frameworks* que desenvolvedores utilizam para criar telas responsivas e dinâmicas.

Trabalhar com *REACT* gera uma facilidade devido a gama de bibliotecas que existem. Escolhido a dedo, *Chakra-ui* comporta uma configuração opcional de temas escuros e claros, facilitando o desenvolvimento de suas telas e aplicação a todos os componentes que são renderizados.

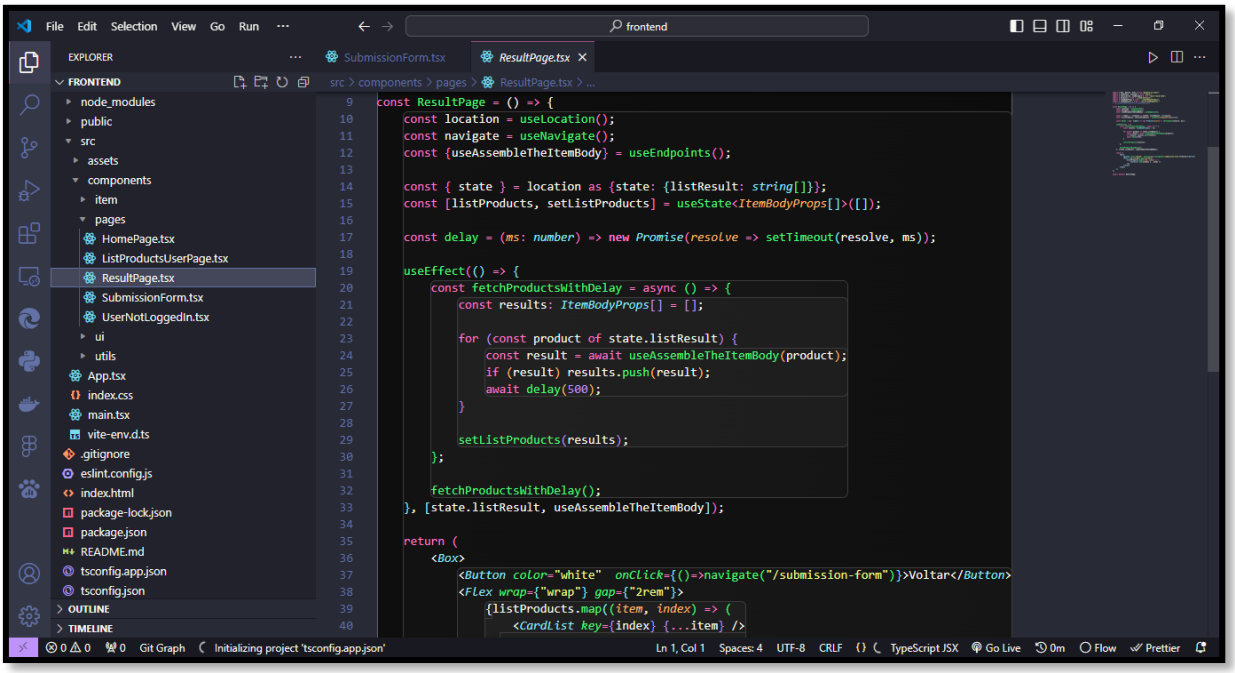
Embora não tenha sido levantado nenhum requisito quanto a temas e estilização, pode ser visto como uma futura evolução no projeto de *software* quando começar mais uma volta no ciclo espiral.

Com base nos conceitos propostos por Sommerville (Sommerville, 2013), *React* e *Angular* podem ser entendidos como ferramentas que ajudam na implementação da camada de apresentação (*frontend*) de um software. Essas ferramentas se alinham com as práticas de modularidade e reutilização, que são enfatizadas pelo autor. *Frameworks* como *Angular* ou bibliotecas como *React* são úteis para promover a reutilização de componentes, simplificar o desenvolvimento de interfaces e melhorar a manutenção do código.

A figura 6 tem um exemplo de código utilizando *TypeScript* com *REACT.* Em específico, a figura 6 retrata uma parte do código onde envia a requisição para obter os detalhes do produto que fora buscado. Ele realiza um atraso entre as requisições

dos produtos para não receber nenhum erro do *backend* quando realizar as requisições, de forma que não trave a aplicação sobrecarregando o servidor com requisições.

Figura 6 – *Frontend TypeScript* com *React*



Fonte: Autor

4.2.3. Testes

Conforme Ian Sommerville (Sommerville, 2013), a implementação robusta exige uma atenção cuidadosa durante o processo de testes e depuração. O autor destaca que, após o desenvolvimento do código, é essencial realizar uma série de testes com o objetivo de descobrir e corrigir o maior número possível de erros antes da entrega do software. Ele enfatiza a importância de criar testes que tenham alta probabilidade de identificar falhas de forma eficiente. Além disso, destaca a necessidade de se adotar uma abordagem de depuração estruturada, como o rastreamento para localizar a origem dos problemas e a eliminação cuidadosa das causas. Neste texto, detalharemos as estratégias utilizadas para testes unitários, testes de componentes e depuração do aplicativo, assegurando que o "MaaTech" cumpra os padrões de qualidade esperados.

No contexto do aplicativo "MaaTech", foram realizados diversos casos de teste, cada um associado diretamente a um caso de uso específico. O objetivo desses testes é garantir que as funcionalidades do aplicativo estejam completamente alinhadas com

os requisitos definidos, proporcionando uma experiência confiável e eficiente para os usuários.

Cada caso de teste é estruturado com informações detalhadas, incluindo identificação, objetivo, data e horário de execução, responsável pela execução, procedimento inicial, passos a serem seguidos durante o teste, resultado esperado e resultado real. Essa abordagem metodológica busca oferecer uma visão clara e detalhada da validação das diversas funcionalidades do "MaaTech".

Os casos de teste descritos estão disponíveis no Apêndice D - Testes. Nesse apêndice, os casos de teste relacionados a cada caso de uso estão organizados de forma sistemática e detalhada. Essa seção apresenta uma visão abrangente do processo de teste realizado em todas as etapas do desenvolvimento do aplicativo.

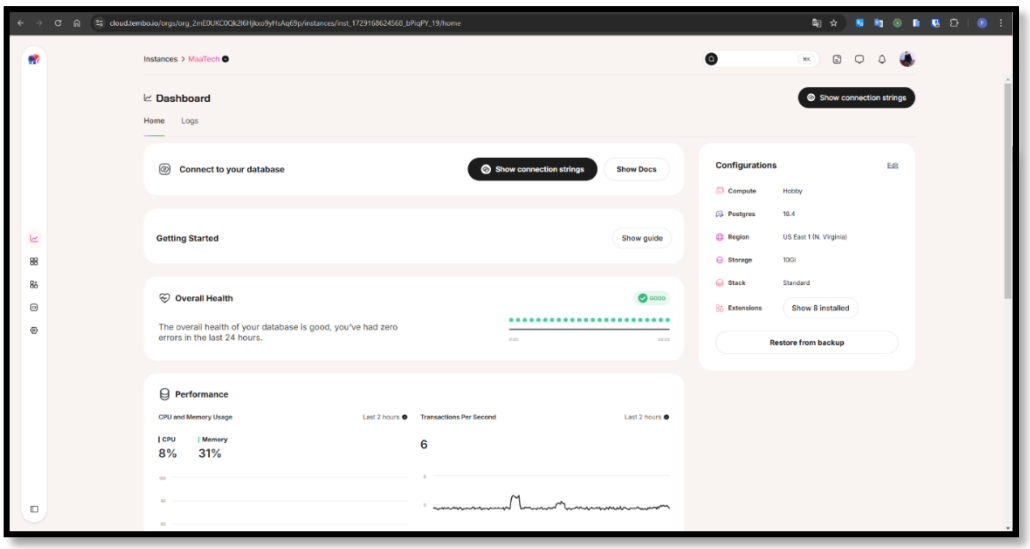
4.2.4. Banco de Dados

Em meio a tantos bancos de dados existentes, definir um é um desafio. Como este projeto de *software* tem como um de seus objetivos entregar uma aplicação mais tecnológica possível, optamos em escolher o *PostgreSQL como* o *database.* É um banco de dados robusto, seguro e extensível e principalmente por possuir um ecossistema rico de ferramentas disponíveis. No portal da *Azure*, você consegue ter acesso a informações precisas sobre *PostgreSQL* e sua documentação atualizada. Outro motivo que se destaca é que não possui um custo de licenciamento. Logo, para primeiro momento, ele se destaca dos demais por entregar o mesmo que outros pagos.

Para hospedar o banco, foi escolhido o site *Tembo.io,* A *Tembo.io* é uma plataforma que facilita o desenvolvimento de aplicações de inteligência artificial (IA), integrando-se ao PostgreSQL para otimizar o armazenamento e a busca de vetores de palavras diretamente no banco de dados, além de oferecer integrações via *SQL* e *HTTP*, permitindo o uso de grandes modelos de linguagem para criar soluções de forma simples e eficiente. Nele, é possível hospedar de forma gratuita e ainda gerenciar o banco de dados, possibilitando acessar de qualquer lugar usando o usuário e a senha do suposto *DBA*.

A figura 7 retrata o banco de dados *PostgreSQL* hospedado no *Tembo.io.* Nessa tela é possível ver os status e se caso tiver algum erro, é alertado na tela.

Figura 7 – Imagem do Banco de Dados *PostgreSQL* hospedado no *Tembo.io*



Fonte: Autor

Após a configuração da instância, é possível acessar, gerenciar e manipular bancos de dados, tabelas e colunas utilizando diversas ferramentas especializadas em administração e consulta, como *DBeaver*, *Azure Data Studio* e outras soluções compatíveis com *PostgreSQL*. Essas ferramentas oferecem interfaces gráficas intuitivas que facilitam a interação com a plataforma Tembo.io, permitindo consultas avançadas, ajustes no esquema e otimização do gerenciamento de dados de forma eficiente.

Na figura 8, é possível ver o acesso ao banco de dados através da ferramenta DBeaver. De acordo com o próprio portal do DBeaver, "DBeaver Community é uma ferramenta de banco de dados multiplataforma gratuita para desenvolvedores, administradores de banco de dados, analistas e todos que trabalham com dados" (DBeaver, 2024).

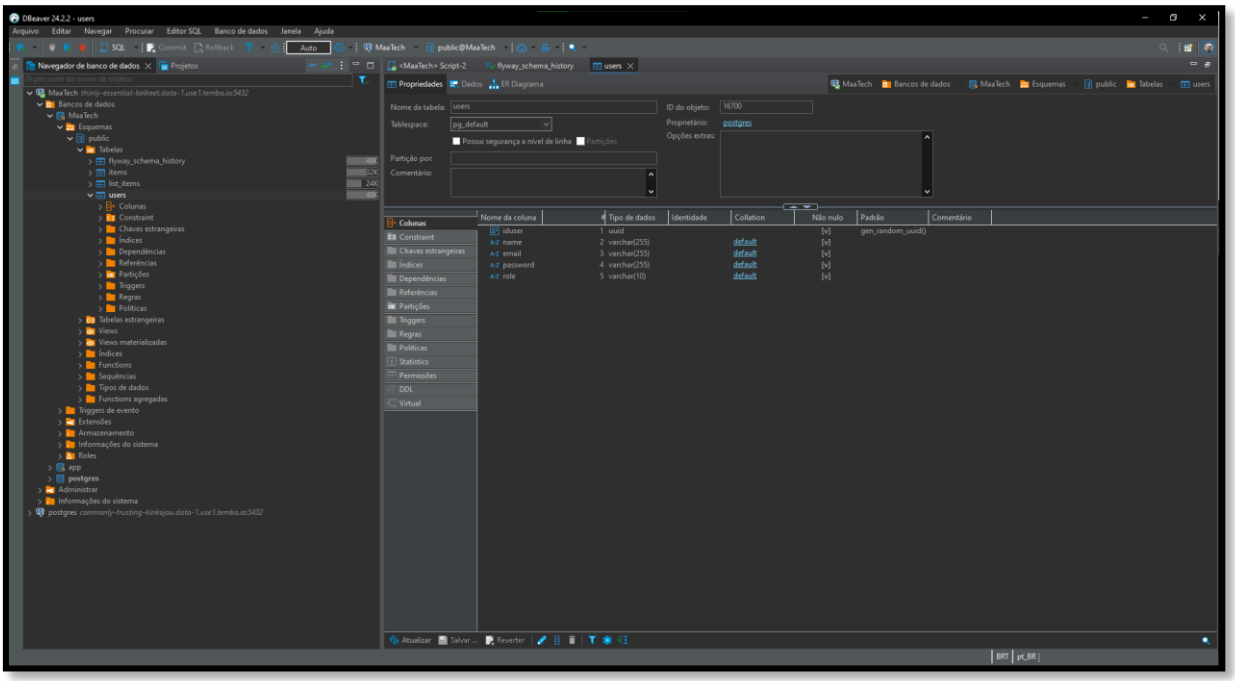
De acordo com Sommerville (2013), a seleção cuidadosa de tecnologias é um passo essencial para alcançar sistemas eficientes e de alta qualidade. O PostgreSQL, amplamente reconhecido por sua robustez, segurança e flexibilidade, é uma escolha estratégica, destacando-se por ser de código aberto e, consequentemente, sem custos de licenciamento. Além disso, sua extensibilidade permite a integração com tecnologias modernas, como inteligência artificial.

A opção pela hospedagem no Tembo.io complementa essa escolha, viabilizando o gerenciamento remoto do banco de dados e a integração direta com modelos avançados de linguagem. Pressman (2019) destaca que ferramentas que

promovem eficiência no ciclo de vida do software, como o Tembo.io, são fundamentais para aumentar a produtividade e reduzir os custos de desenvolvimento.

Combinando um banco de dados versátil a uma plataforma inovadora, a solução proposta segue práticas consolidadas da engenharia de software, atendendo plenamente às demandas do projeto.

Figura 8 – Acessando o banco através do *Dbeaver*



Fonte: Autor

Da mesma forma, pode ser acessado na aplicação e fazer o relacionamento das entidades com as tabelas através de algum *framework*.

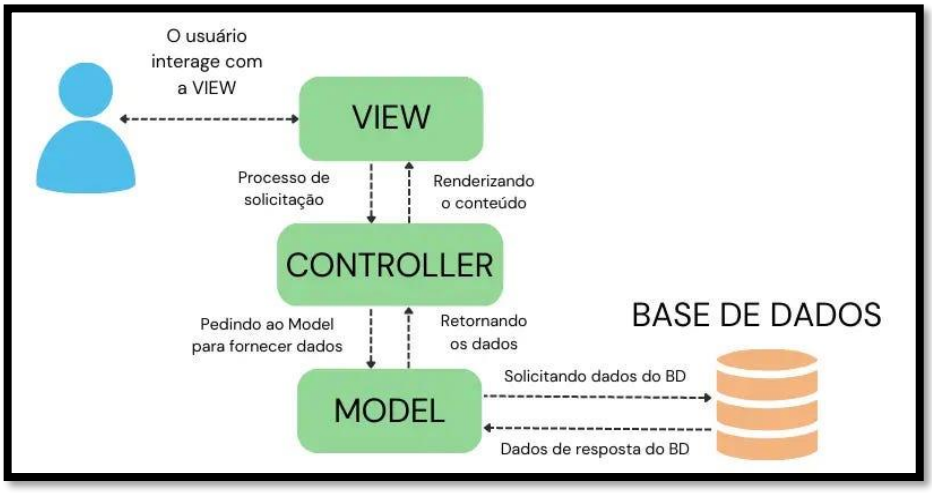
4.2.5. Arquitetura do projeto de desenvolvimento de aplicativo

Existe diversas formas de estruturar uma aplicação. Estamos falando da arquitetura de como o *software* funcionará. Para o referente projeto de *software*, foi utilizado a arquitetura *MVC (Model-View-Controller)* e *MVVM (Model-View-ViewModel)* No portal da *awari,* plataforma voltada para o desenvolvimento de carreira, oferecendo recursos como mentorias e experiências personalizadas, com o objetivo de ajudar os usuários a aprimorar suas habilidades profissionais e alcançar seus objetivos de carreira, informa que *"*Essas arquiteturas permitem uma separação clara entre a camada de apresentação (*frontend*) e a camada de lógica de negócio (*backend*)". A comunicação com o *backend* é feito através de *APIs*, desta forma a

separação de lógica e visual fica mais fácil de ser realizada. É alguma regra de negócio?! Então deve-se desenvolver no *backend*. É algo que deve ser renderizado na página?! Então é um dever do *frontend*.

A figura 9 é uma representação de como funciona a arquitetura MVC de forma ilustrativa e intuitiva. Por meio dela, é possível ver a separação no que se diz respeito a Modelo, Visualização e Controle.

Figura 9 – Exemplo da arquitetura MVC

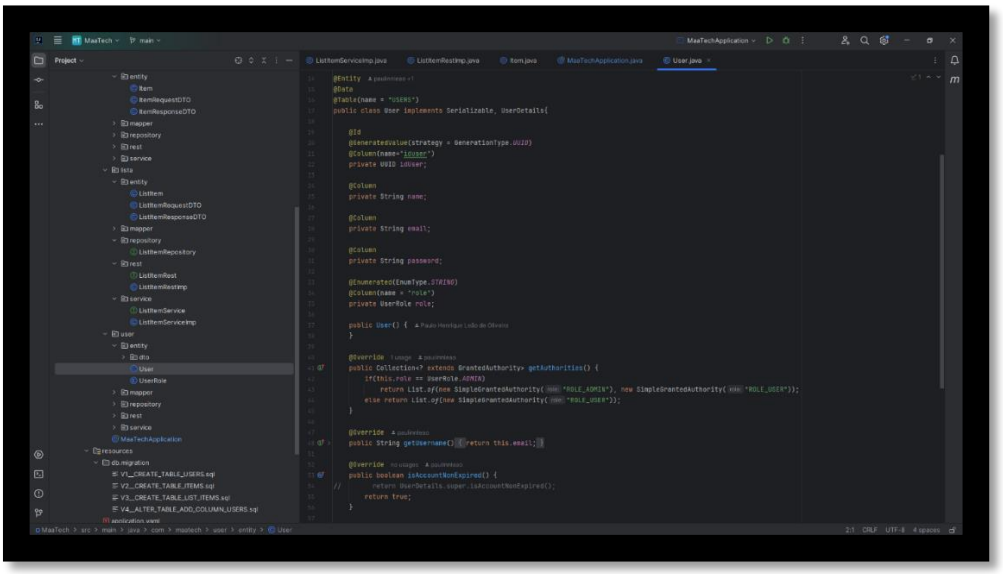


Fonte: *Célio Normando, blog medium*

Basicamente, essa estruturação funciona em diversos projetos e é usualmente utilizada por desenvolvedores em todo o mundo. A arquitetura funciona bem e é legível. Sua separação em modelo, visualização e controle facilita o entendimento de cada parte do código. Basicamente, o *backend* tem o *model* que é a entidade que representa a tabela do banco de dados, o *repository* que faz o relacionamento da tabela com o banco realizando a parte lógica de busca, inserção, deleção e alteração, o *service* onde é implementada a regra de negócio e toda lógica deve ser desenvolvida e por fim o *controller* também conhecido como *rest* por ser um *endpoint* de uma *API*, ele funciona como uma *API* recebendo os parâmetros e chamando o serviço para implementar a parte lógica, ao final, devolve a resposta caso exista.

A figura 10 é um exemplo da arquitetura sendo aplicada no projeto. Na lateral esquerda da imagem, é possível ver a árvore do projeto onde as pastas separam os modelos de Visualização, Controle e Modelo. Através dessa imagem notasse a importância deste modelo de arquitetura aplicado que é a melhor visualização do projeto e entendimento.

Figura 10 – Exemplo da arquitetura no *Backend*



Fonte: Autor

No *frontend*, o conceito de "*model*" refere-se, essencialmente, à resposta que o *backend* envia para o *frontend* por meio de um *JSON*. Como estamos utilizando *TypeScript*, é possível tipar essa resposta ao convertê-la para o tipo esperado, garantindo maior segurança e previsibilidade no uso dos dados. Com o objeto agora tipado, a *view* corresponde aos componentes que são renderizados como scripts no *HTML*, permitindo a apresentação dinâmica da interface para o usuário. Esse fluxo facilita a comunicação entre o *backend* e o *frontend*, além de assegurar a integridade dos dados durante a manipulação nas interfaces (SOMMERVILLE, 2013; PRESMAN, 2016).

4.2.6. Segurança do backend

Como *backend* é uma *API* que pode ser acessada por qualquer outra fonte externa, e no projeto de software foi levantado como requisitos a possibilidade de realizar login, foi desenvolvido a autenticação com o e-mail e senha que são enviados no cadastro. Ao realizar o cadastro, a senha é encriptada e persistida no banco com o seu *HASH*, trazendo uma segurança à mais para o usuário, não expondo sua senha a qualquer outro que tenha acesso ao banco de dados da aplicação. Ao realizar o login, a senha enviada é encriptada e comparada com a senha que está no banco, caso o *HASH* seja equivalente ao existente no banco, é retornado um *token* que ficará disponível por 2 horas para realizar as demais requisições ao *backend*.

Como o *backend* foi feito utilizando o *spring*, utilizamos então o próprio *spring security* para importar o algoritmo de encriptação e para monitorar as rotas das *APIs*.

5. CONSIDERAÇÕES FINAIS

Desenvolver este projeto foi um grande desafio, mas também uma experiência muito enriquecedora. Aplicar o conhecimento de engenharia de *software* na prática nunca é simples, mas seguir a teoria e colocá-la em ação foi essencial para tornar o processo mais organizado e até mesmo prazeroso.

Criar o aplicativo MaaTech foi uma oportunidade de aprendizado que trouxe contato com novas tecnologias, além de explorar ferramentas e conceitos que, embora já existentes, foram novidade. Esse processo ampliou a visão sobre desenvolvimento de software e proporcionou um crescimento significativo.

Trabalhar em um projeto como este exige equilíbrio entre teoria e prática, além de uma boa dose de dedicação. A cada etapa, desde a concepção inicial da ideia até a construção da última tela, é possível aplicar diferentes métodos e ferramentas da engenharia de software, sempre com o objetivo de entregar um resultado final que refletisse o melhor possível do que foi planejado.

5.1. Resultados Obtidos

Como esperado, ao seguir o que foi definido no capítulo 2, obtivemos um aplicativo conciso e objetivo. O projeto de *software* que fora apresentado, caso seja seguido, resultará em um aplicativo que contempla todos os requisitos e regras de negócios levantados durante a sua projeção.

5.2. Dificuldades e Aprendizado

Assim como todo projeto existe uma dificuldade, este projeto de *software* não foi diferente. Dificuldades para implementar algumas regras de negócios ou requisitos foram encontrados durante o desenvolvimento.

Embora tenha sido claro durante a projeção e a definição da aplicação, neste meio tempo as dificuldades de desenvolvimento foram durante o período de teste do formulário. A IA que retornava a lista de produtos ou descrição técnica do produto, não seguia um padrão. Mesmo criando o formulário para evitar textos duvidosos e que a IA interpretasse aquele texto de forma equivocada, a Inteligência Artificial designada para o trabalho, hora ou outra devolvia um texto com formato inesperado ou com

textos desnecessários. Para corrigir esse processo, foi necessário criar um corpo de texto que recebia os parâmetros do formulário e enviasse o mesmo padrão de texto sempre. Desta forma, o texto recebido de resposta era o mesmo.

Outra dificuldade, foi durante o desenvolvimento do *frontend.* Embora não tenha sido dificil, o *frontend* é a parte do projeto onde nunca acaba o trabalho. Existe sempre algo que pode ser melhorado, já que vai ser utilizado pelo usuário. Nesse cenário, o problema maior foi a atualização da biblioteca que foi utilizada no projeto. Chakra-ui teve uma atualização onde as janelas flutuantes definidas no projeto ficaram bagunçadas.

5.3. Importância do Trabalho

Este projeto de software tem sua importância no quesito de aplicar toda a teoria vista no curso de Engenharia de Computação. Ele engloba os mais diversos temas apresentados no curso, desde a engenharia de *software* e requisitos, até mesmo em sistemas distribuídos e criptografia aplicada.

5.4. Sugestões e Perspectivas Futuras

Embora este projeto de software tenha um protótipo de desenvolvimento *web*, essa é uma aplicação que deve ser implementada em futuras versões. O desenvolvimento do *frontend* utilizando o *framework React* foi intencional nesse sentido. Mesmo com a biblioteca *chakra-ui* que já possui uma pré-renderização do formato responsivo, ainda assim espera-se que seja mais bem definido os seus componentes. Outra evolução seria que esse aplicativo fosse implementado em outras aplicações.

CONCLUSÃO

O presente trabalho, teve como alvo projetar o desenvolvimento de um *software* implementando métodos que são usados durante a projeção de um aplicativo. Um projeto de *software* que teve um início, meio e fim de forma que apresentasse ao final uma aplicação com bibliotecas e *frameworks* atualizados. Um projeto que fez a separação dos requisitos, objetivos e regras de negócios, uma arquitetura legível e uma segurança tanto para acessar as *APIs* que foram criadas, quanto para visualizar os dados do banco de dados.

O aplicativo desenvolvido “MaaTech” auxilia pessoas em busca de produtos atualizados de forma que elas não se sintam lesadas e realizem uma busca precisa quanto as suas necessidades, sem precisar dar voltas e voltas em sites e vídeos. Podendo fazer sua pesquisa quanto ao produto selecionado de forma mais precisa e objetiva.

Em comparação com outros *softwares* que entregam a mesma coisa, MaaTech tem o seu diferencial ao realizar essa busca com Inteligência Artificial, retornando produtos que atendem a pessoa e que descreva de maneira mais legível do que em termos técnicos.

O presente projeto pode ser utilizado em pesquisas futuras ou ser agregado também a outros projetos cujos quais precisam de um software inteligente para realizar pesquisas de produtos. Como o projeto possui APIs que realizam todo o trabalho de pesquisar o produto, suas imagens e características, pode ser usado até mesmo para ser uma extensão de algum software de compras online.

REFERÊNCIAS BIBLIOGRÁFICAS

AWARI. Como separar frontend e backend: dicas essenciais para desenvolvedores brasileiros. Disponível em: < https://awari.com.br/como-separar-frontend-e-backend-dicas-essenciais-para-desenvolvedores-brasileiros/ >. Acesso em: 24 nov. 2024.

AZURE. O que é PostgreSQL? Disponível em: < https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-postgresql#:~:text=O%20PostgreSQL%20serve%20como%20o,Object%20Notation)%20e%20dados%20relacionais >. Acesso em: 24 nov. 2024.

DEV MEDIA. Por que Java? Disponível em: < https://www.devmedia.com.br/por-que-java/20384 >. Acesso em: 18 nov. 2024.

DIAS, Ricardo. O modelo em cascata. O Medium, 2019. Disponível em: < https://medium.com/contexto-delimitado/o-modelo-em-cascata-f2418addaf36 >. Acesso em: 17 abr. 2024.

DIAS, Ricardo. O modelo espiral. O Medium, 2019. Disponível em: < https://medium.com/contexto-delimitado/o-modelo-em-espiral-de-boehm-ed1d85b7df >. Acesso em: 17 abr. 2024.

DIAS, Ricardo. O modelo incremental. O Medium, 2019. Disponível em:< https://medium.com/contexto-delimitado/o-modelo-incremental-b41fc06cac04 >. Acesso em: 17 abr. 2024.

DIAS, Ricardo. Os processos de software. O Medium, 2019. Disponível em: < https://medium.com/contexto-delimitado/os-processos-de-software-56a2e70fddfb >. Acesso em: 17 abr. 2024.

EXAME. 35,7% dos brasileiros vive sem esgoto, mas 79,9% tem internet, diz IBGE. Exame, 2019. Disponível em: < https://exame.com/brasil/357-dos-brasileiros-vive-sem-esgoto-mas-799-tem-internet-diz-ibge/ >. Acesso em: 26 abr. 2024.

GRANCURSOS. Processo de software - Engenharia de Software. Disponível em: < https://blog.grancursosonline.com.br/processo-de-software-engenharia-de-software/ >. Acesso em: 17 abr. 2024.

IBM. Java Spring Boot. Disponível em: < https://www.ibm.com/br-pt/topics/java-spring-boot >. Acesso em: 24 nov. 2024.

I MASTERS. Dez motivos para você usar Maven. Disponível em: < https://imasters.com.br/back-end/dez-motivos-para-voce-usar-maven >. Acesso em: 18 nov. 2024.

LOJAS ERP VAREJO. Mercado de eletrodomésticos no Brasil: tendências para o segmento em 2023. Disponível em: <

https://www.mannesoftmaislojas.com.br/blog/mercado-de-eletrodomesticos-no-brasil-tendencias-para-o-segmento-em-2023 >. Acesso em: 26 abr. 2024.

MDN. Fundamentos de JavaScript. Disponível em: < https://developer.mozilla.org/pt-BR/docs/Learn/Getting_started_with_the_web/JavaScript_basics >. Acesso em: 24 nov. 2024.

ORANGO. TypeScript vs JavaScript: vantagens e casos de uso. Disponível em:< https://orango.dev/typescript-vs-javascript-vantagens-e-casos-de-uso/#:~:text=Uma%20das%20grandes%20vantagens%20do,vari%C3%A1veis%20antes%20de%20utiliz%C3%A1-las >. Acesso em: 24 nov. 2024.

ORBIT DATA SCIENCE. Motivos para brasileiros deixarem de comprar produtos de marcas. Disponível em: < https://www.orbitdatascience.com/artigos/motivos-para-brasileiros-deixarem-de-comprar-produtos-marcas >. Acesso em: 17 nov. 2024.

PRESSMAN, Roger S.; MAXIM, Bruce R. Engenharia de Software: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.

PRESSMAN, Roger S. Engenharia de Software: Uma Abordagem Profissional. 7. ed. São Paulo: McGraw-Hill, 2011.

SOMMERVILLE, Ian. Engenharia de Software. 9ª ed. São Paulo: Pearson, 2013.

VVERNER. A importância de fazer o levantamento de requisitos para um sistema. Disponível em: < https://vverner.com/a-importancia-de-fazer-o-levantamento-de-requisitos-para-um-sistema/#:~:text=Entrevistas%20s%C3%A3o%20a%20forma%20mais,ser%20estruturados%20ou%20n%C3%A3o%20estruturadas >. Acesso em: 26 abr. 2024.

MEIO E MENSAGEM. 52% das pessoas acima de 60 têm dificuldade de achar produtos. Disponível em: < https://www.meioemensagem.com.br/marketing/52-das-pessoas-acima-de-60-tem-dificuldade-de-achar-produtos >. Acesso em: 17 nov. 2024.

CNN BRASIL. Mais de 10 milhões de brasileiros não acessam internet por não saberem usar a tecnologia, diz IBGE. Disponível em: < https://www.cnnbrasil.com.br/nacional/mais-de-10-milhoes-de-brasileiros-nao-acessam-internet-por-nao-saberem-usar-a-tecnologia-diz-ibge/ >. Acesso em: 17 nov. 2024.

OLHAR DIGITAL. Geladeiras com tecnologia IA: para que servem? 2024. Disponível em: < https://olhardigital.com.br/2024/06/06/reviews/geladeiras-com-tecnologia-ia-para-que-servem/ >. Acesso em: 11 dez. 2024.

FIGMA. Sobre. Disponível em: < https://www.figma.com/pt-br/about/ >. Acesso em: 11 dez. 2024.

DBEAVER. Página inicial. Disponível em: < https://dbeaver.io/ >. Acesso em: 11 dez. 2024.

ESCOLHENDO BEM. Descubra a fascinante evolução dos eletrodomésticos. Disponível em: < https://escolhendobem.com.br/descubra-a-fascinante-evolucao-dos-eletrodomesticos/ >. Acesso em: 11 dez. 2024.

ESCOLHA IDEAL. Como saber se um produto é bom e vale a pena. Disponível em: < https://escolhaideal.org/como-saber-se-um-produto-e-bom-e-vale-a-pena/. >Acesso em: 11 dez. 2024.

LUCIDCHART. Página inicial. Disponível em: < https://www.lucidchart.com/. > Acesso em: 11 dez. 2024.

APÊNDICE A
DOCUMENTO DE REQUISITOS DO APLICATIVO

Documento de Requisitos –
MaaTECH

Versão 1.0.1

Informações do Documento de Requisitos

Título do documento	Documento de Requisitos do MaaTech.
Autor	Paulo Henrique Leão de Oliveira (PHLO)
Comentários	
Nome do arquivo	MaaTech.doc

HISTÓRICO DE REVISÕES

Revisão	**Data**	**Descrição**	**Autor**
01	10/06/24	Elaboração da primeira versão do documento.	PHLO
02	02/09/24	Elaboração da segunda versão do documento.	PHLO
03	24/11/24	Elaboração da terceira versão do documento.	PHLO
04	25/11/24	Elaboração da quarta versão do documento.	PHLO

SUMÁRIO – APÊNDICE A

Documento de Requisitos

1. Introdução

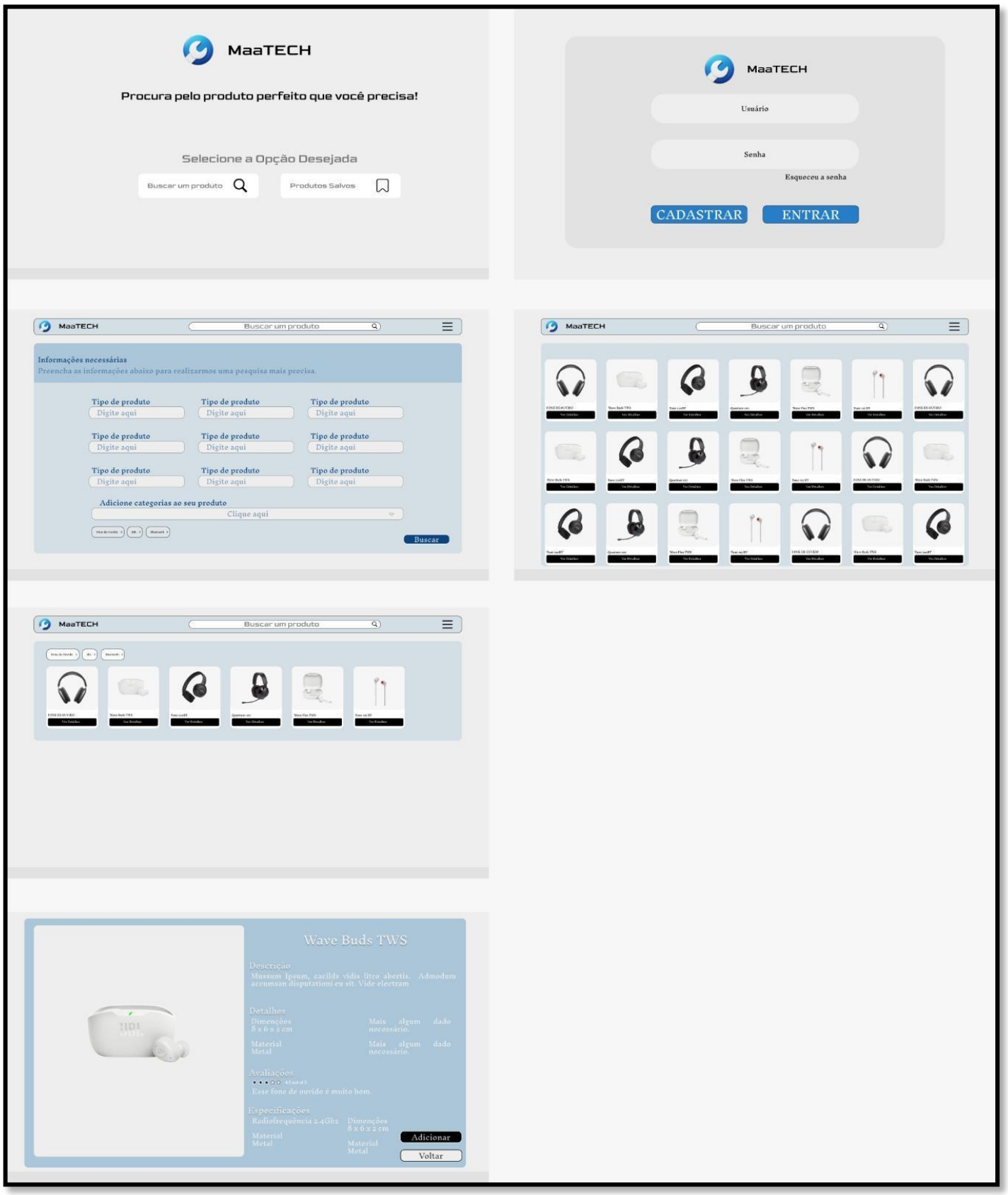
Os eletrodomésticos têm as suas vantagens e características que tornam a vida das pessoas mais fácil. Eles trazem conforto nas atividades rotineiras mais práticas, como cozinhar e lavar. Segundo o Instituto Brasileiro de Geografia e Estatística (IBGE), em 2018, a geladeira estava presente no domicílio de 98,3% dos brasileiros (EXAME, 2019). Esses aparelhos antes não tinham o fácil acesso até mesmo para pessoas que residiam na zona rural.

1.1 Finalidade

A finalidade deste documento é definir as funcionalidades e características necessárias para desenvolver o aplicativo MaaTECH. O *software* orienta pessoas leigas no quesito de tecnologia a comprarem não mais do que o necessário. Fazendo assim com que evite gastos desnecessários ou compras frustradas que não atendem as expectativas.

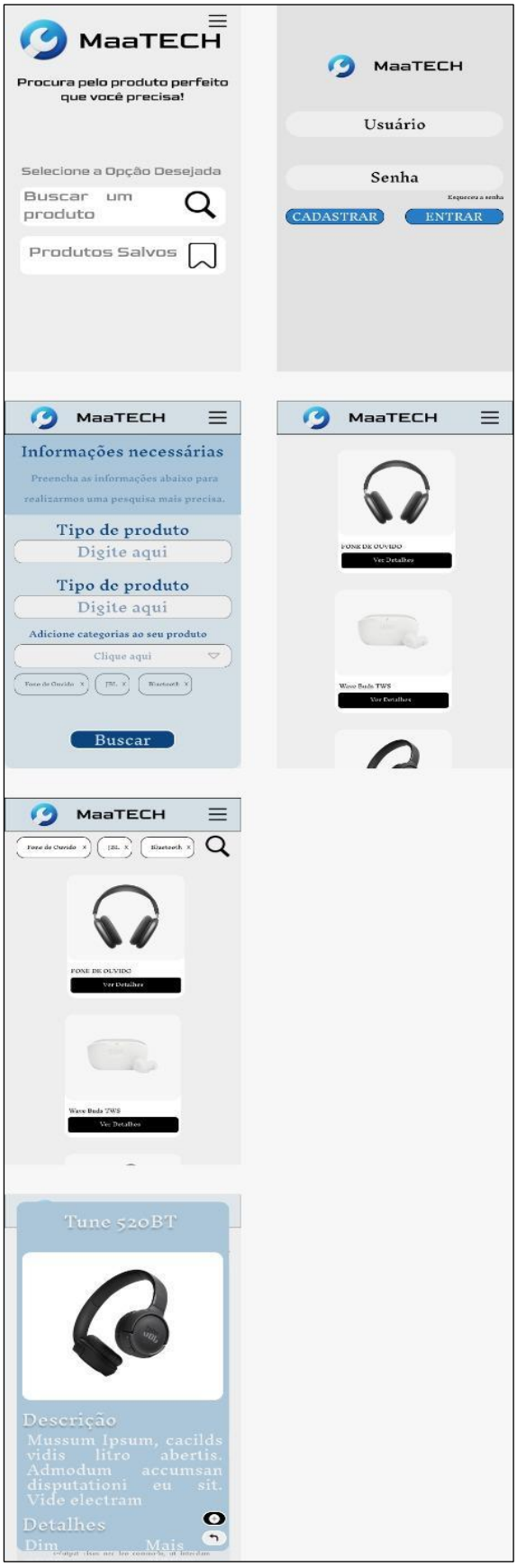
Abaixo está o protótipo do MaaTECH tanto para ambiente desktop, quanto para ambiente mobile.

Figura 1 – MaaTech Web tamanho Desktop



Fonte: Autor

Figura 2 – MaaTECH Web tamanho Mobile



Fonte: Autor

1.2 Escopo

O *software* deverá ser desenvolvido para web, e deve incluir as seguintes funcionalidades:

- Buscar produtos mais atualizados;
- Filtrar conforme a necessidade do cliente;
- Descrever as finalidades do produto;
- Destacar os diferenciais do produto;
- Salvar em uma lista de desejos;

Também deve ser de fácil compreensão e interativo com o usuário. Os recursos deverão ser baseados em produtos já lançados no mercado e que estejam atualizados. O software tem a finalidade de auxiliar os usuários, mas não dispensa também o auxílio de um técnico da área responsável.

1.3 Visão Geral

Este documento contém as seções e uma breve descrição do conteúdo de cada uma:

- Seção A2 - Descrição Geral do Sistema: O objetivo desta seção é delinear o escopo do sistema e seus usuários de forma ampla e genérica.
- Seção A3 - Análise dos Requisitos: Detalha as prioridades e interdependências entre os requisitos. A subseção de Requisitos Funcionais específica todos os relacionados às funcionalidades planejadas para o sistema.
- Seção A4 - Requisitos Não-Funcionais: Foca em especificar todos os requisitos que não estão relacionados diretamente às funcionalidades da primeira iteração do sistema.
- Seção A5 - Diagrama de Casos De Uso: Apresenta um resumo das relações entre os casos de uso que atendem aos requisitos levantados, incluindo os fluxos de eventos, entradas e saídas.

1.4 Padronização

1.4.1 Identificação dos Requisitos

Vamos utilizar a seguinte notação para especificar os requisitos. [Tipo][Número] Nome. O campo “Tipo” deve ser preenchido com uma das seguintes siglas: “RF” (Requisitos Funcionais), “RNF” (Requisitos Não-Funcionais). O campo “Número” será preenchido com um número que reflete a ordem de identificação de cada requisitos. Esta abordagem é essencial para a correta categorização e ordenação dos requisitos, permitindo uma análise eficaz e o acompanhamento detalhado de cada item durante o processo de desenvolvimento do sistema.

1.4.2 Prioridade dos Requisitos

Cada requisito será classificado de acordo com sua prioridade, definindo o que é essencial, importante e desejável para o software. As categorias de prioridade são as seguintes:

- **ALTO:** Requisitos fundamentais e insubstituíveis, sem o qual o sistema não operará.
- **MÉDIO:** Requisitos necessários, mas que, se não implementado, ainda permitirá que o sistema funcione, embora de forma incompleta.
- **BAIXO:** Requisito que oferece um diferencial ao sistema e pode ser adicionado em futuras iterações.

2. Descrição Geral

Buscar produtos tecnológicos que atendam perfeitamente às necessidades pode ser um desafio para muitas pessoas, especialmente sem o conhecimento necessário para evitar escolhas equivocadas. Pensando nisso, o MaaTech foi criado para ser um aliado nesse processo, ajudando os usuários a encontrar as melhores opções de forma eficiente e segura. O objetivo do software é simplificar a busca, economizando tempo e garantindo que os usuários façam escolhas bem-informadas.

Com uma série de recursos pensados para atender a essas necessidades, o MaaTech se destaca por sua praticidade e foco na experiência do usuário. Ao longo deste documento, você encontrará uma explicação detalhada de cada requisito, mostrando como cada funcionalidade contribui para alcançar os objetivos do software.

3. Requisitos Funcionais

Nesta seção, você encontrará uma explicação detalhada dos requisitos funcionais do sistema, com uma visão geral das funcionalidades que precisam ser implementadas para garantir que tudo funcione corretamente. É importante destacar que uma análise cuidadosa desses requisitos é essencial para o sucesso do projeto. Compreendendo claramente o que o sistema precisa, conseguimos criar soluções mais eficazes e atender melhor às expectativas dos usuários.

3.1.1 - [RF001] - Acesso ao Botão “Buscar Produtos”

Requisito que permitirá o usuário acessar o botão de buscar produtos.

Nrº do Requisito:	RF001
Classificação:	Requisito Funcional (RF)
Descrição:	Qualquer usuário pode acessar a funcionalidade de busca na tela principal, independentemente de estar autenticado.
Justificativa:	Facilitar o acesso à busca para todos os usuários, sem a obrigatoriedade de login imediato.
Origem do requisito:	Usabilidade para usuários não autenticados.
Critério de Aceitação:	Usuários autenticados e não autenticados devem conseguir acessar a busca na página principal.
Dependências:	
Prioridades:	Alta
Conflitos:	

Fonte: Autor.

3.1.2 - [RF002] - Redirecionamento para Login ao Acessar "Produtos Salvos"

Requisito que direciona o usuário para realizar login caso tente acessar produtos salvos.

Nrº do Requisito:	RF002

Classificação:	Requisito Funcional (RF)
Descrição:	O sistema deve redirecionar para a página de login quando um usuário não autenticado tenta acessar "Produtos Salvos".
Justificativa:	Garantir segurança e privacidade para usuários.
Origem do requisito:	Necessidade de proteção de dados dos usuários.
Critério de Aceitação:	O sistema deve redirecionar o usuário para a página de login ao acessar "Produtos Salvos" sem autenticação.
Dependências:	RF003
Prioridades:	Alta
Conflitos:	

Fonte: Autor.

3.1.3 - [RF003] - Funcionalidade de Login e Cadastro

O sistema deve permitir que os usuários se cadastrem e realizem login.

Nrº do Requisito:	RF003
Classificação:	Requisito Funcional (RF)
Descrição:	O sistema deve permitir que os usuários se cadastrem e façam login para salvar e gerenciar suas listas de compras personalizadas.
Justificativa:	Garantir que os usuários possam salvar seus produtos e gerenciar suas preferências.
Origem do requisito:	Necessidade de gerenciamento de dados pessoais.
Critério de Aceitação:	Usuários devem conseguir se cadastrar e fazer login para acessar listas personalizadas.
Dependências:	
Prioridades:	Alta
Conflitos:	

Fonte: Autor.

3.1.4 - [RF004] - Visualização de perfil

Visualizar e editar perfil cadastrado.

Nrº do Requisito:	RF004
Classificação:	Requisito Funcional (RF)
Descrição:	Após a autenticação, o sistema deve permitir que o usuário visualize e edite suas informações de perfil, como nome, e-mail e preferências.
Justificativa:	Facilitar a gestão de dados pessoais e preferências do usuário.
Origem do requisito:	Necessidade de personalização e gerenciamento de contas.
Critério de Aceitação:	O usuário autenticado deve poder acessar e editar suas informações de perfil sem dificuldade.
Dependências:	RF003
Prioridades:	Média
Conflitos:	

Fonte: Autor.

3.1.5 - [RF005] - Salvar Histórico de Busca

Permitir que o usuário salve um item em sua lista.

Nrº do Requisito:	RF005
Classificação:	Requisito Funcional (RF)
Descrição:	O sistema deve salvar automaticamente o histórico de busca para usuários autenticados. Para usuários não autenticados, o histórico deve ser salvo localmente no navegador.
Justificativa:	Melhorar a usabilidade, permitindo que os usuários acessem suas buscas anteriores facilmente.
Origem do requisito:	Necessidade de uma funcionalidade de histórico para aumentar a eficiência do processo de busca.
Critério de Aceitação:	O sistema deve permitir acesso ao histórico de busca tanto para usuários autenticados quanto para não autenticados.

Dependências:	RF003, RF013
Prioridades:	Média
Conflitos:	

Fonte: Autor.

3.1.6 - [RF006] - Validação de Campos Obrigatórios no Formulário de Busca

O usuário deve preencher os campos obrigatórios do formulário para proceder com a busca.

Nrº do Requisito:	RF006
Classificação:	Requisito Funcional (RF)
Descrição:	O sistema deve garantir que todos os campos obrigatórios, como "Nome do Produto", "Categoria", e "Faixa de Preço", estejam preenchidos antes de executar uma busca.
Justificativa:	Aumentar a precisão dos resultados de busca, exigindo informações essenciais.
Origem do requisito:	Necessidade de precisão nas buscas.
Critério de Aceitação:	O sistema deve impedir que a busca seja realizada se os campos obrigatórios estiverem vazios.
Dependências:	RF007
Prioridades:	Alta
Conflitos:	

Fonte: Autor.

3.1.7 - [RF007] - Preenchimento de Campos Opcionais

O usuário pode preencher os campos não obrigatórios do formulário para fazer uma busca mais precisa.

Nrº do Requisito:	RF007
Classificação:	Requisito Funcional (RF)

Descrição:	O sistema deve permitir o preenchimento de campos opcionais no formulário de busca, como "Marca", "Modelo Específico", e "Tamanho/Capacidade", para uma busca mais detalhada.
Justificativa:	Proporcionar resultados mais personalizados e adequados às preferências dos usuários.
Origem do requisito:	Necessidade de detalhamento nas buscas.
Critério de Aceitação:	Usuários devem poder preencher os campos opcionais para obter resultados mais precisos.
Dependências:	RF006
Prioridades:	Média
Conflitos:	

Fonte: Autor.

3.1.8 - [RF008] - Listagem de Produtos com Descrição

O sistema deve listar os produtos com uma breve descrição.

Nrº do Requisito:	RF008
Classificação:	Requisito Funcional (RF)
Descrição:	O sistema deve exibir uma lista de produtos que correspondam à busca do usuário, com uma breve descrição de cada item.
Justificativa:	Facilitar a decisão de compra ao fornecer informações essenciais de forma clara.
Origem do requisito:	Necessidade de uma apresentação clara dos resultados de busca.
Critério de Aceitação:	A lista de produtos deve ser exibida com descrições sucintas e informações relevantes.
Dependências:	RF006, RF007
Prioridades:	Alta
Conflitos:	

Fonte: Autor.

3.1.9 - [RF009] - Visualização de Detalhes dos Produtos

Quando o usuário clicar no produto, ele tem acesso a mais detalhes do produto.

Nrº do Requisito:	RF009
Classificação:	Requisito Funcional (RF)
Descrição:	O usuário deve poder visualizar informações detalhadas sobre cada produto listado, com a opção de ajustar os filtros de busca para refinar os resultados.
Justificativa:	Proporcionar uma experiência de compra informada e personalizada.
Origem do requisito:	Necessidade de detalhamento nos resultados de busca.
Critério de Aceitação:	Os usuários devem poder visualizar os detalhes de cada produto e ajustar os filtros de busca.
Dependências:	RF008
Prioridades:	Alta
Conflitos:	

Fonte: Autor.

3.1.10 - [RF010] - Adicionar Produto à Lista de Favoritos

Caso o usuário esteja autenticado e tenha interesse em favoritar o produto.

Nrº do Requisito:	RF010
Classificação:	Requisito Funcional (RF)
Descrição:	O sistema deve permitir que usuários autenticados adicionem produtos à sua lista de favoritos. Usuários não autenticados devem ser redirecionados para a página de login ao tentar adicionar produtos.
Justificativa:	Facilitar a organização de produtos de interesse para futuras consultas ou compras.
Origem do requisito:	Necessidade de gerenciamento de favoritos.

Critério de Aceitação:	O sistema deve permitir a adição de produtos à lista de favoritos, com redirecionamento para login quando necessário.
Dependências:	RF003
Prioridades:	Média
Conflitos:	

Fonte: Autor.

3.1.11 - [RF011] - Visualizar e Gerenciar Produtos Favoritos

O usuário tem acesso a lista de favoritos caso esteja autenticado, podendo excluir ou visualizar o produto.

Nrº do Requisito:	RF011
Classificação:	Requisito Funcional (RF)
Descrição:	O sistema deve permitir que os usuários visualizem e gerenciem seus produtos favoritos, incluindo a opção de remover produtos da lista.
Justificativa:	Facilitar o gerenciamento de produtos de interesse.
Origem do requisito:	Necessidade de gerenciamento de favoritos.
Critério de Aceitação:	Usuários devem poder visualizar e gerenciar seus produtos favoritos de forma eficiente.
Dependências:	RF010
Prioridades:	Média
Conflitos:	

Fonte: Autor.

3.1.12 - [RF012] - Recuperação de Senha

Caso o usuário tenha registrado no MaaTech, e tenha esquecido a senha. Ele deve conseguir recuperar a senha.

Nrº do Requisito:	RF012
Classificação:	Requisito Funcional (RF)
Descrição:	O sistema deve oferecer uma funcionalidade de recuperação de senha, enviando um link para redefinição ao e-mail cadastrado.
Justificativa:	Facilitar o acesso ao sistema caso a senha seja esquecida.
Origem do requisito:	Necessidade de acessibilidade contínua ao sistema.
Critério de Aceitação:	O usuário deve receber um link de redefinição de senha em seu e-mail dentro do tempo estipulado.
Dependências:	RNF002
Prioridades:	Alta
Conflitos:	

Fonte: Autor.

3.1.13 - [RF013] - Integração com IA para Busca de Produtos

A busca solicitada pelo usuário deve ser feita por uma IA (Inteligência Artificial).

Nrº do Requisito:	RF013
Classificação:	Requisito Funcional (RF)
Descrição:	O sistema deve utilizar uma Inteligência Artificial para realizar buscas de produtos, personalizando os resultados com base nas preferências e histórico de compras do usuário.
Justificativa:	Fornecer resultados de busca mais personalizados e precisos, com base no perfil do usuário e em suas interações anteriores com o sistema.
Origem do requisito:	Necessidade de personalização e melhoria na experiência de compra.
Critério de Aceitação:	O sistema deve apresentar resultados de busca baseados nas preferências dos usuários e melhorar a cada interação.
Dependências:	RF001, RF005

Prioridades:	Alta
Conflitos:	

Fonte: Autor.

4. Requisitos Não-Funcionais

Nesta parte são apresentados detalhadamente os requisitos não funcionais do sistema, igualmente importantes para o seu bom funcionamento. Cada requisito descrito aqui é essencial para garantir que o sistema atenda às expectativas do usuário e tenha um desempenho satisfatório em diversos cenários de uso. É importante notar que, mesmo que os requisitos não funcionais não estejam diretamente relacionados com a funcionalidade do sistema, eles são cruciais para a sua qualidade e eficácia. Portanto, uma análise cuidadosa destes requisitos é essencial para o desenvolvimento de um sistema confiável e eficiente.

4.1.1. [RNF001] Conformidade com Padrões de Usabilidade

Nrº do Requisito:	RNF001
Classificação:	Requisito Não-Funcional (RNF)
Descrição:	O sistema deve seguir padrões de usabilidade que garantam facilidade de uso, mesmo para pessoas com pouca experiência em tecnologia.
Justificativa:	Facilitar o uso do sistema por todos os tipos de usuários, independentemente de sua familiaridade com tecnologia.
Origem do requisito:	Necessidade de acessibilidade e simplicidade no uso do sistema.
Critério de Aceitação:	O sistema deve ser facilmente navegável e intuitivo para todos os usuários, com feedback positivo em testes de usabilidade
Dependências:	RF001, RNF003.
Prioridades:	Alta

Conflitos:	

Fonte: Autor.

4.1.2. [RNF002] Segurança de Dados

Nrº do Requisito:	RNF002
Classificação:	Requisito Não-Funcional (RNF)
Descrição:	O sistema deve garantir a proteção das informações dos usuários, utilizando criptografia para dados sensíveis, como senhas, e assegurando que apenas usuários autenticados possam acessar listas de favoritos e históricos de busca.
Justificativa:	Proteger os dados pessoais e garantir privacidade, cumprindo normas de segurança digital.
Origem do requisito:	Necessidade de segurança e conformidade com leis de proteção de dados.
Critério de Aceitação:	Os dados devem ser criptografados e acessíveis apenas por usuários autenticados. Testes de segurança devem confirmar a invulnerabilidade do sistema.
Dependências:	RF002, RF012.
Prioridades:	Alta
Conflitos:	

Fonte: Autor.

4.2. Requisito de Interface

4.2.1. [RNF003] Interface de Usuário Clara e Intuitiva

O programa deve ser de fácil uso.

Nrº do Requisito:	RNF003
Classificação:	Requisito Não-Funcional (RNF)

Descrição:	A interface deve ser simples, organizada e intuitiva, permitindo que usuários menos experientes possam navegar facilmente pelas funcionalidades do sistema.
Justificativa:	Melhorar a experiência do usuário, especialmente para aqueles com pouca familiaridade com tecnologia.
Origem do requisito:	Necessidade de usabilidade para todos os públicos, com foco em usuários leigos.
Critério de Aceitação:	O sistema deve ser testado por usuários com pouca experiência em tecnologia, que devem ser capazes de completar tarefas sem ajuda.
Dependências:	RF001, RF003, RF004.
Prioridades:	Alta
Conflitos:	

Fonte: Autor.

4.3. Requisito de Desempenho

4.3.1. [RNF004] Tempo de Resposta da Interface

Nrº do Requisito:	RNF004
Classificação:	Requisito Não-Funcional (RNF)
Descrição:	A interface do sistema deve ser responsiva, com o tempo de carregamento das páginas e ações, como redirecionamento para login ou exibição de resultados de busca, não excedendo 2 segundos.
Justificativa:	Garantir uma experiência de usuário eficiente e fluida.
Origem do requisito:	Necessidade de otimização da experiência do usuário.
Critério de Aceitação:	Páginas e funcionalidades devem carregar dentro do tempo estipulado, sem causar atrasos perceptíveis.
Dependências:	RNF001, RNF002.
Prioridades:	Alta

Conflitos:	

Fonte: Autor.

4.3.2. [RNF005] Tempo de Resposta para Salvamento do Histórico de Busca

O sistema precisa salvar rapidamente o que o usuário solicitar.

Nrº do Requisito:	RNF005
Classificação:	Requisito Não-Funcional (RNF)
Descrição:	O sistema deve salvar o histórico de busca dos usuários autenticados em menos de 1 segundo.
Justificativa:	Garantir que os usuários possam acessar suas buscas anteriores rapidamente, sem interrupções.
Origem do requisito:	Necessidade de resposta rápida para aumentar a satisfação do usuário.
Critério de Aceitação:	O histórico deve ser salvo instantaneamente sem comprometer a experiência do usuário.
Dependências:	RF005
Prioridades:	Alta
Conflitos:	

Fonte: Autor.

4.4. Requisito de Usuário

4.4.1. [RNF006] Facilidade de Uso para Usuários Leigos

Nrº do Requisito:	RNF006
Classificação:	Requisito Não-Funcional (RNF)
Descrição:	O sistema deve ser intuitivo e fácil de usar, mesmo para usuários com pouca experiência em tecnologia.

Justificativa:	Garantir que o sistema seja acessível a todos, independentemente de sua familiaridade com tecnologia.
Origem do requisito:	Necessidade de acessibilidade para uma base de usuários mais ampla.
Critério de Aceitação:	Testes com usuários leigos devem confirmar a facilidade de uso do sistema.
Dependências:	RF001, RNF003.
Prioridades:	Alta
Conflitos:	

Fonte: Autor.

4.4.2. [RNF007] Personalização de Resultados

Nrº do Requisito:	RNF007
Classificação:	Requisito Não-Funcional (RNF)
Descrição:	O sistema deve permitir que os usuários ajustem suas preferências de busca, como marca e faixa de preço, e que a IA personalize os resultados com base em seus interesses.
Justificativa:	Fornecer resultados mais alinhados às preferências e necessidades dos usuários.
Origem do requisito:	Necessidade de personalização e adequação aos perfis dos usuários.
Critério de Aceitação:	O sistema deve ajustar os resultados de busca com base nas preferências fornecidas pelos usuários.
Dependências:	RF007, RF013.
Prioridades:	Alta
Conflitos:	

Fonte: Autor.

4.4.3. [RNF008] Orientação e Feedback ao Usuário

Nrº do Requisito:	RNF008
Classificação:	Requisito Não-Funcional (RNF)
Descrição:	O sistema deve fornecer orientações claras durante o uso, como dicas de preenchimento de formulários e feedback após a execução de ações.
Justificativa:	Facilitar o uso do sistema e garantir que os usuários compreendam o resultado de suas ações.
Origem do requisito:	Necessidade de melhorar a experiência do usuário.
Critério de Aceitação:	O sistema deve exibir mensagens claras e feedback após cada interação.
Dependências:	RF006
Prioridades:	Média
Conflitos:	

Fonte: Autor.

4.4.3. [RNF009] Orientação e Feedback ao Usuário

Nrº do Requisito:	RNF009
Classificação:	Requisito Não-Funcional (RNF)
Descrição:	O sistema deve permitir que o usuário configure e altere suas preferências e histórico de compras, para que as recomendações sejam sempre relevantes e atualizadas.
Justificativa:	Proporcionar uma experiência personalizada e garantir que as recomendações evoluam com o tempo e preferências do usuário.
Origem do requisito:	Necessidade de personalização contínua.
Critério de Aceitação:	Usuários devem poder alterar suas preferências e ver as recomendações atualizadas.
Dependências:	RF013, RF004.
Prioridades:	Alta
Conflitos:	

Fonte: Autor.

A Tabela 1 é uma matriz de rastreabilidade para facilitar a visualização dos requisitos de forma que a navegação entre ele seja mais fácil.

Tabela 1 – Matriz de Rastreabilidade dos Requisitos

Matriz de Rastreabilidade																						
Projeto MaaTech																						
Requisito	RF1	RF2	RF3	RF4	RF5	RF6	RF7	RF8	RF9	RF10	RF11	RF12	RF13	RNF1	RNF2	RNF3	RNF4	RNF5	RNF6	RNF7	RNF8	RNF9
RF1																						
RF2			X																			
RF3																						
RF4			X																			
RF5			X										X									
RF6							X															
RF7						X																
RF8						X	X															
RF9								X														
RF10			X																			
RF11										X												
RF12															X							
RF13	X				X																	
RN1	X															X						
RNF2		X										X										
RNF3	X		X	X																		
RNF4														X	X							
RNF5					X																	
RNF6	X															X						
RNF7							X						X									
RNF8						X																
RNF9				X									X									

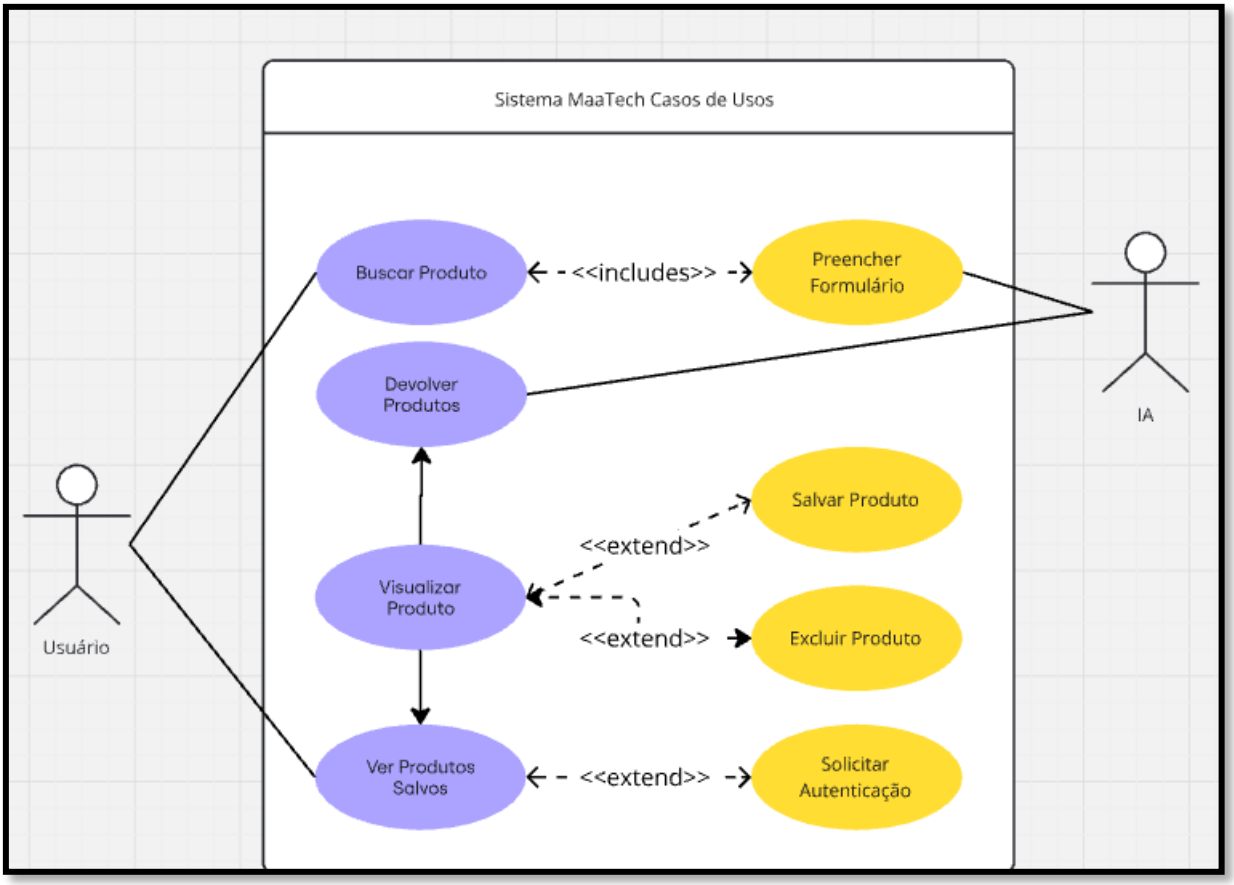
Fonte autor.

APÊNDICE B
ARTEFATOS DO PROJETO

O projeto possui diversas etapas e definições, tais como casos de uso, diagramas de relacionamento. Essas separações e definições servem para facilitar o desenvolvimento da aplicação. Um projeto bem separado e definido tende a ser um projeto limpo e bem-feito. Quando se tem más separações, a aplicação final pode ser cheia de gambiarras e erros inesperados, mais conhecidos como *bugs.*

1. Casos de Usos

Figura 1 – Casos de Usos



Fonte – Autor

O caso de uso é uma técnica de modelagem usada para descrever as interações do sistema com o usuário ou outros sistemas. Através do caso de uso, o projeto de *software* fornece uma visão clara das funcionalidades do sistema, servem

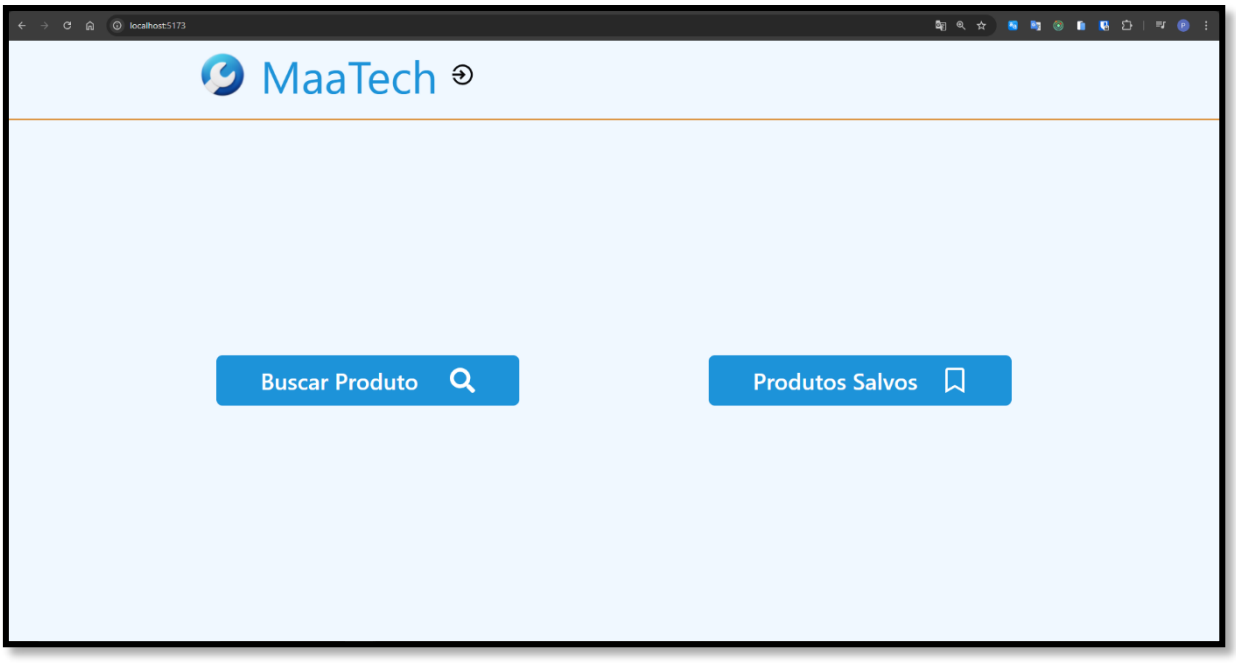
como base para identificar requisitos funcionais, facilita a comunicação entre os clientes, desenvolvedores e analistas e são uteis para validar os testes.

Caso de Uso Buscar Produto

Identificação: (CSU01) Buscar Produto
Escopo: Acesso via web
Descrição do propósito: Permitir que o usuário busque produtos preenchendo um formulário
Ator primário: Usuário
Interessados: Usuário
Pré-condições:
Pós-condições:
Fluxo normal: 1. O usuário acessa na página inicial a opção de buscar produto 2. A página apresenta um formulário a ser preenchido pelo usuário 3. O sistema envia para a API da Gemini a requisição com o formulário 4. O resultado é listado para o usuário

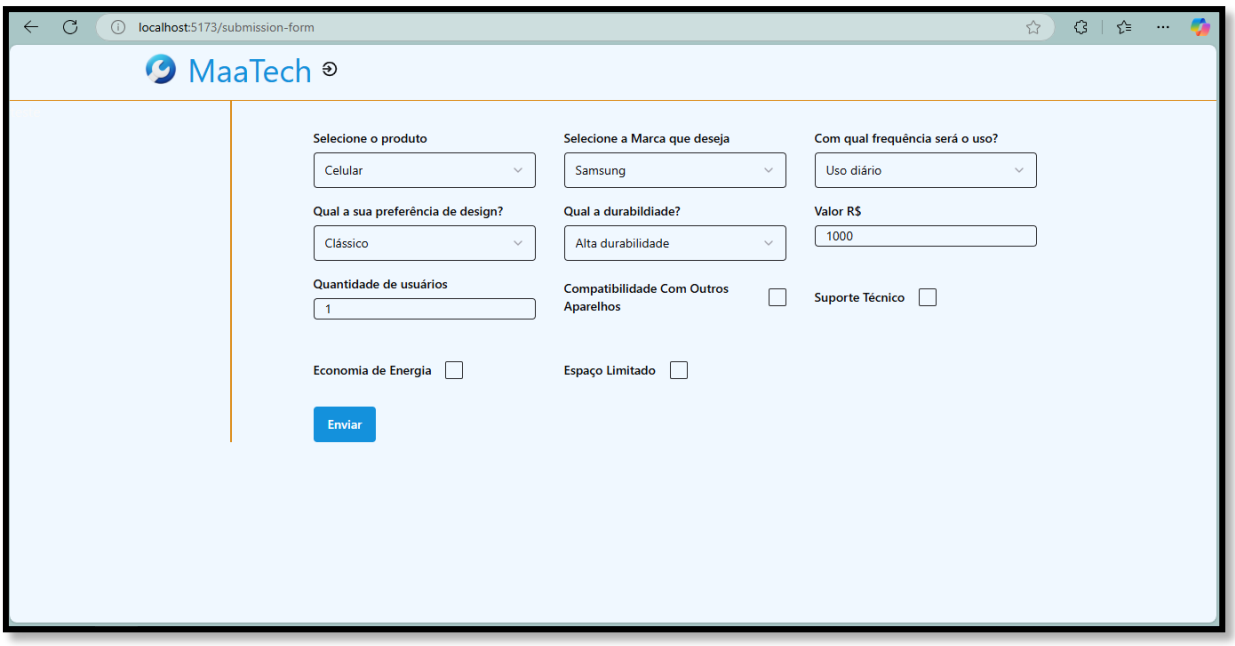
Fonte: Elaborado pelo próprio autor.

Figura 2 – Tela Inicial



Fonte – Autor

Figura 3 – Tela do Formulário



Fonte – Autor

Caso de Uso Visualizar Produto

Identificação: (CSU02) Visualizar Produto
Escopo: Acesso via web
Descrição do propósito: Permitir que o usuário visualize os produtos retornados pela API da Gemini
Ator primário: Usuário
Interessados: Usuário
Pré-condições:
Pós-condições:
Fluxo normal: 1. Com a listagem de produtos, o usuário clica no botão visualizar 2. Abre um modal com descrições do produto

Fonte: Elaborado pelo próprio autor.

Caso de Uso Visualizar Produto

Identificação: (CSU03) Visualizar Produto
Escopo: Acesso via web
Descrição do propósito: Permitir que o usuário visualize os produtos retornados pela API da Gemini
Ator primário: Usuário
Interessados: Usuário
Pré-condições:
Pós-condições:
Fluxo normal: 1. Com a listagem de produtos, o usuário clica no botão visualizar 2. Abre um modal com descrições do produto

Fonte: Elaborado pelo próprio autor.

Caso de Uso Salvar ou Deletar o Produto

Identificação: (CSU04) Salvar Produto
Escopo: Acesso via web
Descrição do propósito: Permitir que o usuário salve o produto visualizado ou delete caso já tenha na sua lista de produtos
Ator primário: Usuário
Interessados: Usuário
Pré-condições: Usuário estar logado
Pós-condições:
Fluxo normal: 1. Com o produto aberto, o usuário pode adicioná-lo a sua lista

Fonte: Elaborado pelo próprio autor.

Caso de Uso Ver Lista de Produtos Salvos

Identificação: (CSU05) Ver Lista de Produtos Salvos
Escopo: Acesso via web
Descrição do propósito: Permitir que o usuário visualize a lista de produtos salvos
Ator primário: Usuário
Interessados: Usuário
Pré-condições: Usuário estar logado
Pós-condições:
Fluxo normal: 1. Na tela inicial o usuário pode ir direto para seus produtos salvos

Fonte: Elaborado pelo próprio autor.

Caso de Uso Cadastro

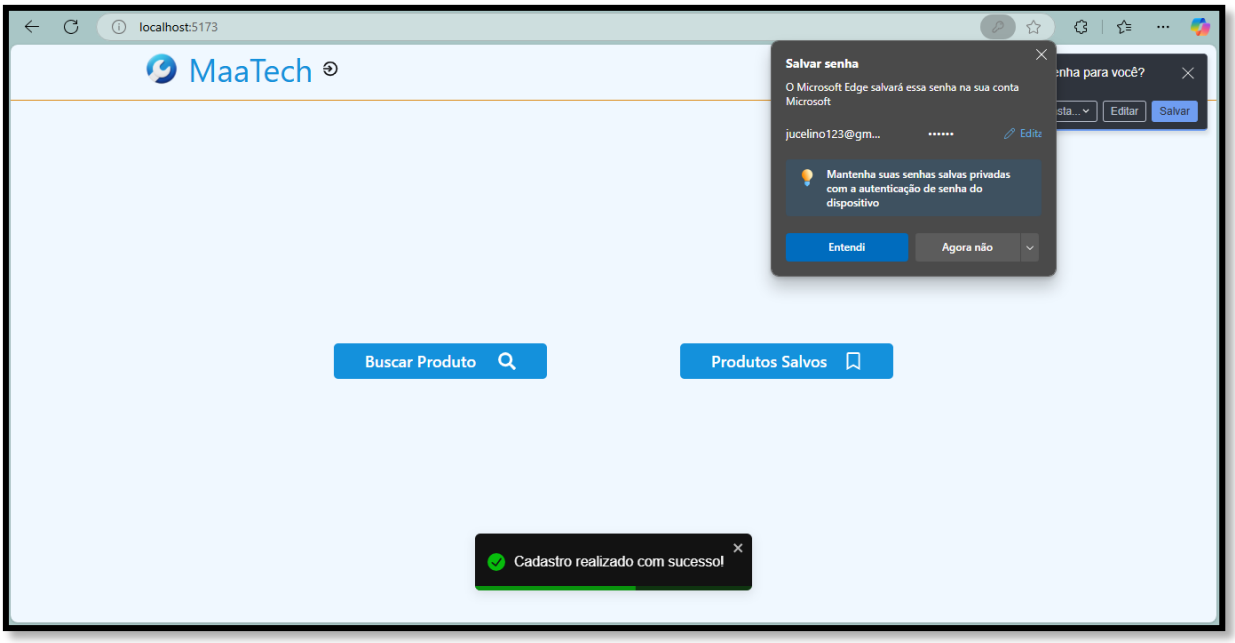
Identificação: (CSU06) Cadastro
Escopo: Acesso via web
Descrição do propósito: Permitir o usuário realizar cadastro
Ator primário: Usuário
Interessados: Usuário
Pré-condições: Não existir email já cadastrado
Pós-condições:
Fluxo normal: 1. Quando o usuário clicar em ver lista, é redirecionado para a tela de login. 2. Seleciona a opção de cadastrar caso não tenha nenhuma conta.

Fonte: Elaborado pelo próprio autor.

Figura 4 – Cadastro de Usuário

Fonte – Autor

Figura 5 – Cadastro de Usuário Com Sucesso



Fonte – Autor

Caso de Uso Cadastro

Identificação: (CSU07) Login
Escopo: Acesso via web
Descrição do propósito: Permitir o usuário realizar login
Ator primário: Usuário
Interessados: Usuário

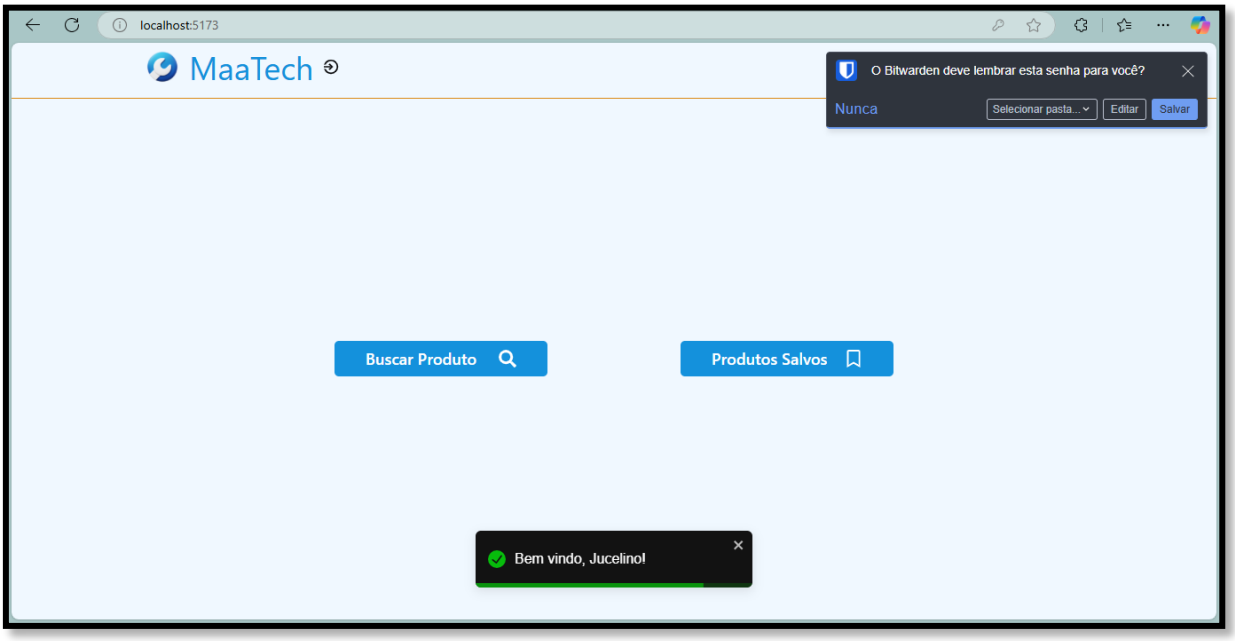
Pré-condições: Existir email já cadastrado
Pós-condições:
Fluxo normal: 1. Quando o usuário clicar em ver lista, é redirecionado para a tela de login. 2. Preenche os dados para realizar o login

Fonte: Elaborado pelo próprio autor.

Figura 6 – Realizando login

Fonte – Autor

Figura 4 – Realizando login com sucesso

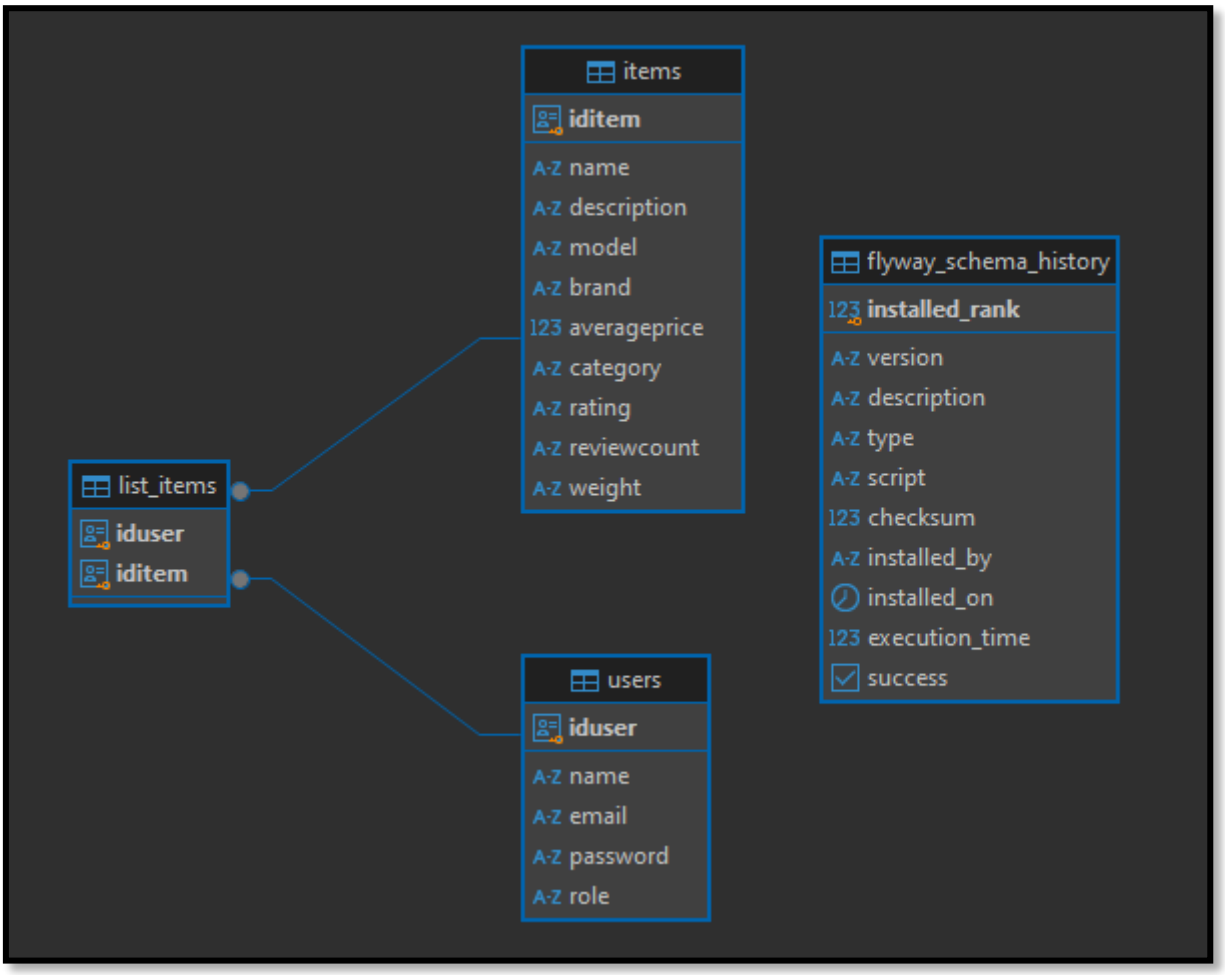


Fonte – Autor

2. Diagrama de Relacionamento com Entidade

A figura 4 representa o diagrama de relacionamento das entidades no banco de dados. Através desta imagem é possível ter um melhor entendimento de como as tabelas se enxergam e se relacionam.

Figura 4 – Diagrama de Relacionamento

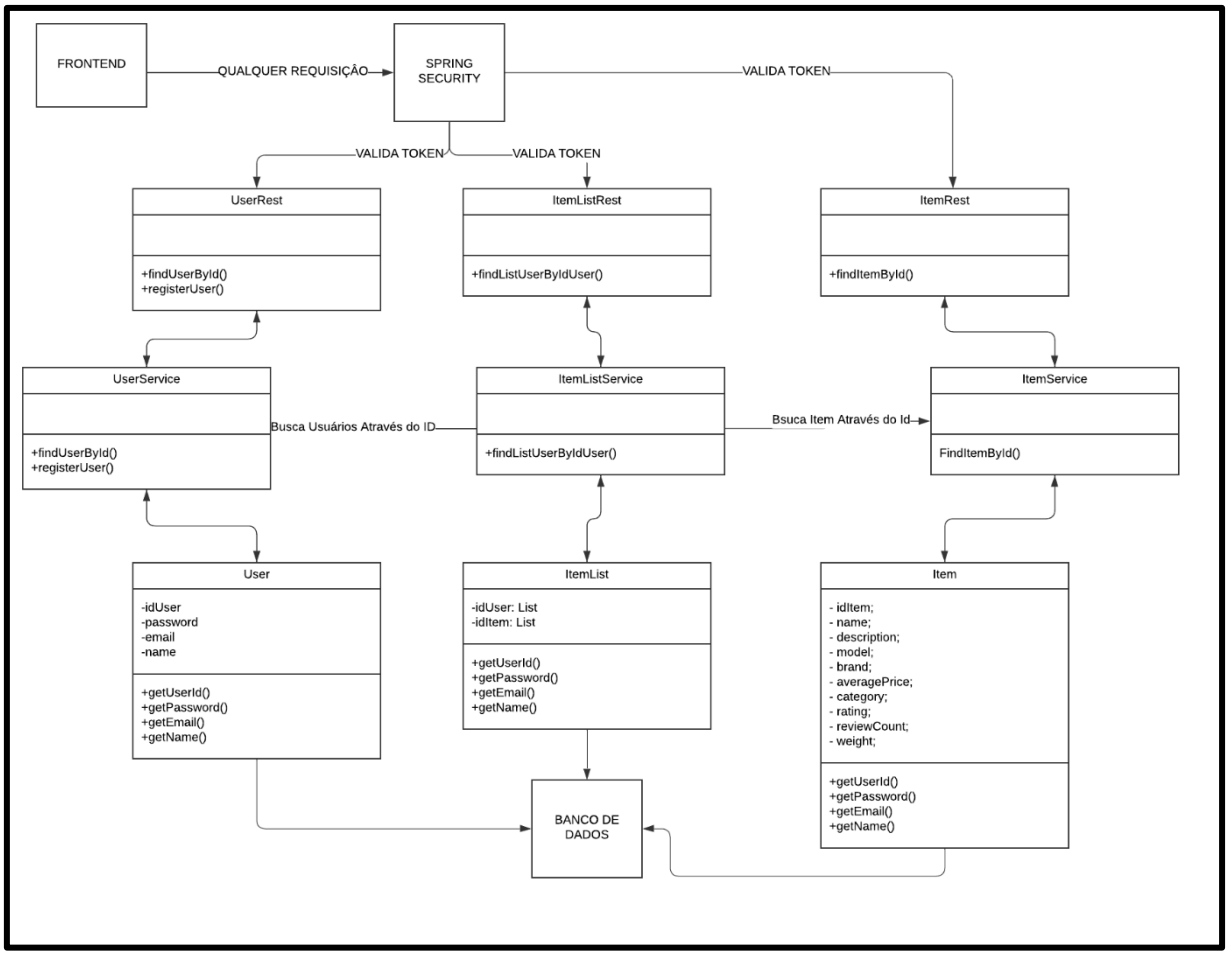


Fonte – Autor

3. Diagrama UML

A figura 5 é um simples modelo de UML feito para entender o relacionamento que as classes tem no código. Como no DER o relacionamento é feito através de uma tabela terceira, da mesma forma são as classes no código. O relacionamento ocorre através da classe *listItem* que enxerga as outras duas classes *user* e *item*

Figura 5 – Diagrama de Relacionamento



Fonte – Autor

Tanto o diagrama UML quanto o diagrama de classe de uso forma feitos através da plataforma luciddart. Lucidchart é uma ferramenta online de criação de diagramas e visualizações gráficas que permite a criação de fluxogramas, organogramas e diagramas de processos, facilitando a colaboração em tempo real entre usuários. A plataforma se integra com diversas outras ferramentas, como Google Drive e Microsoft Office, e é amplamente utilizada em áreas como análise de sistemas e planejamento de processos de negócios (Lucidchart, 2024). A ferramenta foi escolhida devido sua facilidade de disponibilizar os componentes e conectá-los uns com outros. Além de ser uma plataforma gratuita, o que facilita o seu acesso.

APÊNDICE C
CONSTRUÇÃO DO SOFTWARE

Partes importantes do código e considerados essenciais para o funcionamento da aplicação, estão presentes neste apêndice.

1. APIs de IA

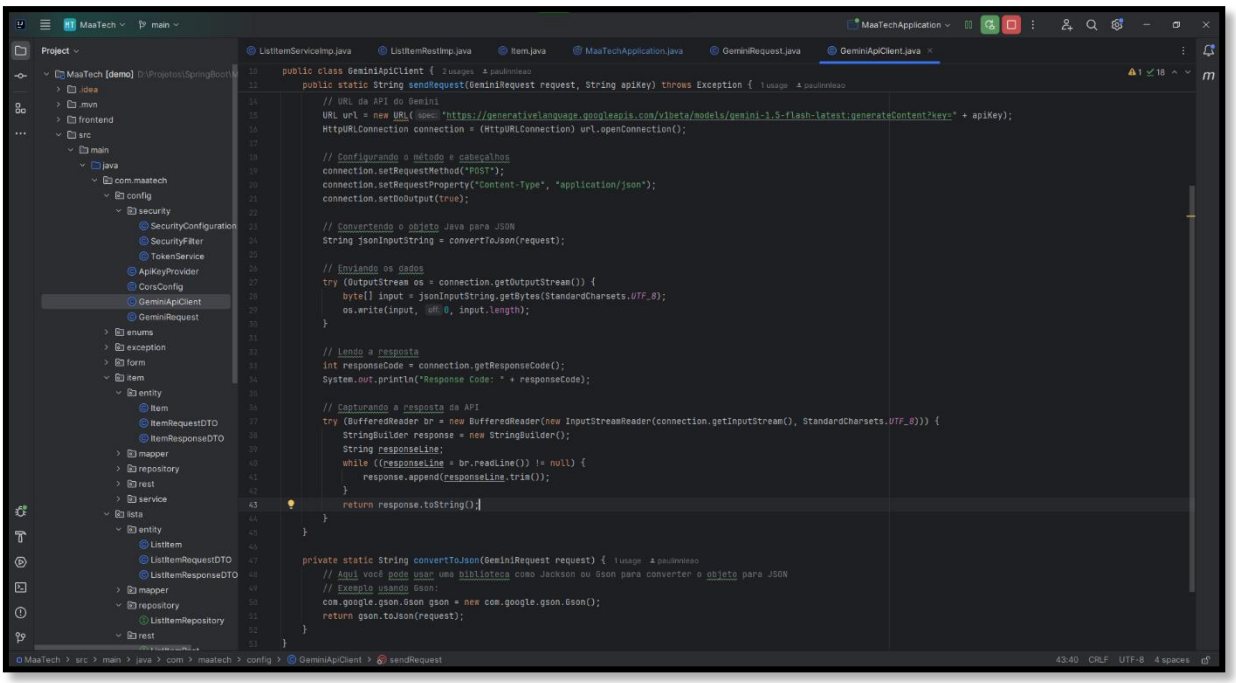
A função principal da aplicação que é realizar a busca com Inteligência Artificial foi feita através de uma API da Gemini. É enviado um texto formatado para a API solicitando que ela liste 10 itens do que fora solicitado pelo cliente, ou seja, do que foi preenchido no formulário. Posteriormente, é enviado outra requisição para a mesma API solicitando mais dados de cada produto que ela retornou. Como são partes lógicas da aplicação, toda essa requisição ficou sobre responsabilidade do *backend.*

A requisição é feita para o *backend* que envia uma requisição para a Gemini. Foi feito uma chave de autenticação na Gemini e configurada para ser usada para este projeto. A requisição é enviada como JSON e a chave como cabeçalho de autorização da requisição.

As imagens que também foram cruciais para a visualização do produto no sistema, tiveram de ser buscadas através de outra API. A API da *WallMart* tem sua função no código buscando imagens atualizadas dos produtos de forma que o usuário tenha acesso aos últimos detalhes que existem no produto. Embora outras APIs tenham um custo para utilizá-las ou sejam muito burocráticas no seu uso, a API da *WallMart* prestou um bom suporte permitindo 20 requisições ao mês, o que foi suficiente para realizar testes e definir se essa seria a API ideal para teste projeto.

Através da figura 1 é possível ver como foi implementado o método de busca com as APIs. A chave token cadastrada realiza o processo de busca, montando o corpo da requisição que será enviada e assim validada.

Figura 1 – Classe de configuração da GEMINI API e Wallmart

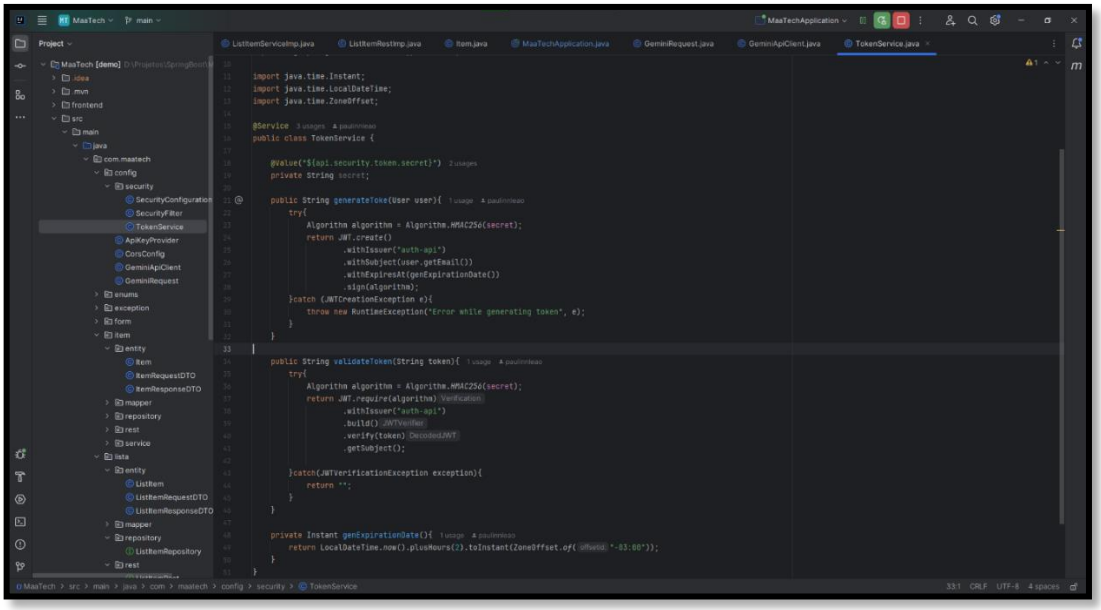


Fonte – Autor

2. Configuração do *SpringSecurity*

Outro ponto principal da aplicação é a segurança que tem para acessar o *backend*. O usuário até pode enviar o formulário sem se autenticar, mas para acessar outros *endpoins* da API do *backend* o usuário deve estar autenticado com uma conta válida. Ele recebe um *token* que é verificado a cada requisição. Caso o token seja válido, validação que é feita pelo próprio *spring security,* o sistema não retorna erro algum. Caso contrário, retorna um erro do *spring security,* alegando que o token é inválido, solicitando que o usuário ou se cadastre para ter o token ou realize o login. Veja a classe de serviço do token na imagem 2.

Figura 2 – Autenticação com *spring security*


MaaTech
main
MaaTechApplication
Project
ListItemServiceImp.java
ListItemRestImp.java
Item.java
MaaTechApplication.java
GeminiRequest.java
GeminiApiClient.java
TokenService.java
MaaTech [demo]
idea
mvn
frontend
src
main
java
com.maatech
config
security
SecurityConfiguration
SecurityFilter
TokenService
ApiKeyProvider
CorsConfig
GeminiApiClient
GeminiRequest
enums
exception
form
item
entity
Item
ItemRequestDTO
ItemResponseDTO
mapper
repository
rest
service
lista
entity
ListItem
ListItemRequestDTO
ListItemResponseDTO
mapper
repository
ListItemRepository
rest
import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
@Service
public class TokenService {
@Value("${api.security.token.secret}")
private String secret;
public String generateToke(User user){
try{
Algorithm algorithm = Algorithm.HMAC256(secret);
return JWT.create()
.withIssuer("auth-api")
.withSubject(user.getEmail())
.withExpiresAt(genExpirationDate())
.sign(algorithm);
}catch (JWTCreationException e){
throw new RuntimeException("Error while generating token", e);
}
}
public String validateToken(String token){
try{
Algorithm algorithm = Algorithm.HMAC256(secret);
return JWT.require(algorithm)
.withIssuer("auth-api")
.build()
.verify(token)
.getSubject();
}catch(JWTVerificationException exception){
return "";
}
}
private Instant genExpirationDate(){
return LocalDateTime.now().plusHours(2).toInstant(ZoneOffset.of("-03:00"));
}
}
MaaTech > src > main > java > com > maatech > config > security > TokenService
33:1
CRLF
UTF-8
4 spaces

Fonte – Autor

APÊNDICE D
TESTES – CASOS DE TESTES

Identificação	**CT01 – Cadastro**
Objetivo:	Verificar se é possível cadastrar um novo usuário no sistema.
Data – Hora:	25/11/2024 - 14h12
Responsável	Paulo Henrique Leão de Oliveira
Procedimento Inicial:	Clicar no botão de login ao lado da logo ou clicar em Produtos Salvos, em seguida clicar em cadastrar.
Passos:	Digitar no campo “Nome Completo”: Jucelino Digitar no campo “Email”: jucelino@gmail.com Digitar no campo “Senha”: “123456” Digitar no campo “Confirmar Senha”: “123456” Clicar no botão “Cadastrar”
Resultado Esperado:	Espera-se que as validações feitas no front com *Yup* sejam efetivas no quesito de encriptação da senha visualmente e validação de e-mail. Ao enviar, caso não exista nenhum e-mail cadastrado igual ao e-mail enviado, o sistema automaticamente faz o login do usuário, armazena a token do usuário no *storage* e redireciona para tela de lista do usuário.
Resultado Real:	O sistema não identificou nenhum e-mail e redirecionou para a tela principal.
Evidências	Capturas de tela mostrando a tela de registro preenchida, a mensagem de confirmação de registro e a tela de login subsequente

Identificação	**CT02 – Login**
Objetivo:	Realiza a tentativa de login do usuário
Data – Hora:	25/11/2024 - 14h42
Responsável	Paulo Henrique Leão de Oliveira
Procedimento Inicial:	Clicar no botão de login ao lado da logo ou clicar em Produtos Salvos
Passos:	Digitar no campo “Email”: jucelino123@gmail.com Digitar no campo “Senha”: “123456” Clicar no botão “Entrar”
Resultado Esperado:	Espera-se que as validações feitas no front com *Yup* sejam efetivas no quesito de encriptação da senha visualmente e validação de e-mail. Ao enviar, caso exista um e-mail igual ao e-mail enviado e a senha quando encriptada for o mesmo *Hash* que está no banco de dados, o sistema faz o login do usuário, armazena a token do usuário no *storage* e redireciona para tela principal.

Resultado Real:	O sistema não identificou nenhum e-mail e redirecionou para a tela de lista do usuário.
Evidências	Capturas de tela mostrando a tela de registro login preenchida e a mensagem de login com sucesso.

Identificação	**CT03 – Buscar Produtos**
Objetivo:	Preenche um formulário para buscar os produtos específicos
Data – Hora:	25/11/2024 - 14h54
Responsável	Paulo Henrique Leão de Oliveira
Procedimento Inicial:	Clicar no botão de Buscar Produtos
Passos:	Selecionar a opção no campo “Selecione o produto”: Celular Selecionar a opção no campo “Selecione a Marca que deseja”: Samsung Selecionar a opção no campo “Com qual frequência será o uso?”: Uso diário Selecionar a opção no campo “Qual a sua preferência de design?”: Clássico Selecionar a opção no campo “Qual a durabilidade?”: Alta durabilidade Digitar no campo “Valor R$”: 1000 Digitar no campo “Quantidade de usuários”: 1 Selecionar a opção “Compatibilidade com outros aparelhos”: Não Selecionar a opção “Suporte Técnico”: Não Selecionar a opção “Economia de Energia”: Não Selecionar a opção “Espaço limitado”: Não Clicar no botão “Enviar”
Resultado Esperado:	Deve enviar o formulário e então ser encaminhado para a página de listagem de produtos com o resultado do formulário.
Resultado Real:	O sistema carregou os produtos em cards.
Evidências	Capturas de tela mostrando a tela com os produtos nos *cards.*

Identificação	**CT04 – Visualizar o Produto**
Objetivo:	Visualizar os detalhes e características do produto
Data – Hora:	25/11/2024 - 15h00
Responsável	Paulo Henrique Leão de Oliveira
Procedimento Inicial:	Clicar no botão Visualizar
Passos:	
Resultado Esperado:	Um modal será aberto com detalhes e a imagem do produto.
Resultado Real:	O modal foi aberto com detalhes do produto.

Evidências	Capturas de tela mostrando a tela com o produto e sua descrição

Identificação	**CT05 – Visualizar Lista de Produtos do Usuário**
Objetivo:	Visualizar os produtos que estão salvos para o usuário logado.
Data – Hora:	25/11/2024 - 15h10
Responsável	Paulo Henrique Leão de Oliveira
Procedimento Inicial:	No menu inicial, clicar no botão produtos salvos
Passos:	
Resultado Esperado:	Caso tenha algum produto salvo, uma lista de produtos deve ser carregada
Resultado Real:	Uma lista com os produtos do usuário foi apresentada com produtos em *cards*.
Evidências	Capturas de tela mostrando a tela com os produtos nos *cards.*

Os casos de testes executados e presente no documento foram feitos para validar todo o sistema no geral. Testes de caixa cinza foram aplicados para verificar tanto a funcionalidade do sistema em seu objetivo principal, quanto para o seu desempenho.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 ● Setor Universitário
Caixa Postal 86 ● CEP 74605-010
Goiânia ● Goiás ● Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br ● reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante Paulo Henrique Leão de Oliveira do Curso de Engenharia de Computação, matrícula 2019100330072-1, telefone: 62994670109 e-mail aluno.paulo.leao@gmail.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado Projeto e desenvolvimento de software para auxiliar em compras tecnológicas, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto (PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 10 de Dezembro de 2024.

Assinatura do(s) autor(es): _____

Nome completo do autor: Paulo Henrique Leão de Oliveira

Documento assinado digitalmente
gov.br
ANDRE LUIZ ALVES
Data: 12/12/2024 18:11:51-0300
Verifique em https://validar.iti.gov.br

Assinatura do professor-orientador: _____

Nome completo do professor-orientador: André Luiz Alves