



CURSO DE SISTEMAS DE INFORMAÇÃO

SISTEMA DE INFORMAÇÃO MÓVEL MULTIPLATAFORMA COM SUPORTE A GEOLOCALIZAÇÃO PARA VENDA DE LOTES

DANIEL FERREIRA RODRIGUES

Palmas

2023



CURSO DE SISTEMAS DE INFORMAÇÃO

SISTEMA DE INFORMAÇÃO MÓVEL MULTIPLATAFORMA COM SUPORTE A GEOLOCALIZAÇÃO PARA VENDA DE LOTES

DANIEL FERREIRA RODRIGUES

Projeto apresentado ao Curso de Sistemas de Informação da Fundação Universidade do Tocantins - UNITINS como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação, sob a orientação do professor Me. Silvano Maneck Malfatti.

Palmas

2023

**ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO DO CURSO DE SISTEMAS
DE INFORMAÇÃO DA FUNDAÇÃO UNIVERSIDADE ESTADUAL DO TOCANTINS - UNITINS**

Aos **23** dias do mês de **Junho** de **2023**, reuniu-se na Fundação Universidade Estadual do Tocantins, Câmpus Palmas, Bloco B, às **08:20 horas**, sob a Coordenação do Professor **Silvano Maneck Malfatti** a banca examinadora de Trabalho de Conclusão de Curso em Sistemas de Informação, composta pelos examinadores Professor **Silvano Maneck Malfatti** (Orientador), Professor **Márcia Maria Savoine** e Professor **Marco Firmino de Sousa**, para avaliação da defesa do trabalho intitulado **“Sistema de Informação Móvel Multiplataforma com Suporte a Geolocalização para Vendas de Lotes”** do acadêmico **Daniel Ferreira Rodrigues** como requisito para aprovação na disciplina Trabalho de Conclusão de Curso II (TCC II). Após exposição do trabalho realizado pelo acadêmico e arguição pelos Examinadores da banca, em conformidade com o disposto no Regulamento de Trabalho de Conclusão de Curso em Sistemas de Informação, a banca atribuiu a pontuação: **6,5**.

Sendo, portanto, o Acadêmico: Aprovado Reprovado

Assinam esta Ata:



Documento assinado digitalmente

SILVANO MANECK MALFATTI
Data: 29/06/2023 11:42:20-0300
Verifique em <https://validar.itl.gov.br>

Professor Orientador: _____



Documento assinado digitalmente

MARCA MARIA SAVOINE
Data: 04/07/2023 19:23:02-0300
Verifique em <https://validar.itl.gov.br>

Examinador: _____



Documento assinado digitalmente

MARCO ANTONIO FIRMINO DE SOUSA
Data: 29/06/2023 22:37:50-0300
Verifique em <https://validar.itl.gov.br>

Examinador: _____

Silvano Maneck Malfatti
Presidente da Banca Examinadora

Coordenação do Curso de Sistemas de Informação

**Dados Internacionais de Catalogação na Publicação
(CIP) Sistema de Bibliotecas da Universidade Estadual
do Tocantins**

R696s

RODRIGUES, Daniel Ferreira
SISTEMA DE INFORMAÇÃO MÓVEL
MULTIPLATAFORMA COM SUPORTE A
GEOLOCALIZAÇÃO PARA VENDA DE LOTES.
Daniel Ferreira Rodrigues. - Palmas, TO, 2023

Monografia Graduação - Universidade Estadual do
Tocantins – Câmpus Universitário de Palmas - Curso de
Sistemas de Informação, 2023.

Orientador: Silvano Maneck Malfatti Maneck Malfatti

1. Aplicativo multiplataforma. 2. Arquitetura de
Desenvolvimento. 3. API. 4. Geolocalização.

CDD 610.7

TODOS OS DIREITOS RESERVADOS – A reprodução total ou parcial, de qualquer forma ou por
qualquer meio deste documento é autorizado desde que citada a fonte. A violação dos direitos do
autor (Lei nº 9.610/98) é crime estabelecido pelo artigo 184 do Código Penal.

**Elaborado pelo sistema de geração automática de ficha catalográfica da UNITINS com os
dados fornecidos pelo(a) autor(a).**

Este trabalho é dedicado à minha família e amigos. Obrigado por todo apoio.

Agradecimentos

Primeiramente agradeço a Deus pela graça imerecida. Por todo esse caminho Deus sempre esteve comigo.

Agradeço muito aos meus pais que sempre incentivaram e acreditaram em mim. Minha família que com seu afeto me deu suporte para encarar todas as dificuldades que surgiram.

Agradeço ao meu orientador Silvano Malfatti, por acompanhar este projeto desde a concepção da ideia e finalização de cada fase.

Aos meus amigos que me acompanharam por todo esse tempo.

*“O SENHOR firma os passos do homem bom e no seu caminho se compraz;
se cair, não ficará prostrado, porque o SENHOR o segura pela mão.”
(Salmos 37: 23-24)*

Resumo

A sociedade cada vez mais está imersa nas tecnologias, ou seja conectada a tudo e a todos. Sendo tudo, destacamos aqui os aplicativos/dispositivos móveis com funções direcionadas a rotinas cotidianas, como a rotina da geolocalização. Percebendo a importância dos aplicativos geográficos no cotidiano da sociedade, este trabalho tem por objetivo demonstrar a criação e utilização de um aplicativo que facilite a comercialização de terrenos/lotes, de modo que o proprietário não fique atrelado às consultorias imobiliárias. A pesquisa apontou que, existe sim, a problemática de um domínio total do framework Flutter para finalização de todas as funcionalidades do aplicativo. Todavia o projeto é concluído com uma documentação objetiva, um API que atende às necessidades básicas e um aplicativo que se aproxima do objetivo final.

Palavras-chave: Sistema de Informação, Aplicativo, Geolocalização, Terrenos Imobiliários.

Abstract

Society is increasingly immersed in technologies, that is, connected to everything and everyone. That being all, we highlight here the applications / mobile devices with functions aimed at everyday routines, such as the geolocation routine. Realizing the importance of geographic applications in society's daily life, this work aims to demonstrate the creation and use of an application that facilitates the commercialization of land/lots, so that the owner is not tied to real estate consultancies. The research pointed out that, yes, there is the problem of a full domain of the Flutter framework to finalize all the functionalities of the application. However, the project is completed with objective documentation, an API that meets the basic needs and an application that approaches the final goal.

Key-words: Information System, Application, Geolocation, Land Real Estate.

Lista de ilustrações

Figura 1 – Arquitetura Flutter.	18
Figura 2 – Arquitetura MVC.	22
Figura 3 – API.	23
Figura 4 – Bases do modelo Scrum.	27
Figura 5 – Materiais.	29
Figura 6 – Etapas e atividades do projeto.	31
Figura 7 – Diagrama de Caso de Uso	35
Figura 8 – Diagrama de Classes	37
Figura 9 – Arquitetura da Aplicação	38
Figura 10 – Definição das rotas da API.	39
Figura 11 – Código da classe controladora do Usuário.	40
Figura 12 – Código da classe modelo do Usuário.	41
Figura 13 – Aplicativo To No Lote - Cadastro de Usuário	42
Figura 14 – Aplicativo To no Lote - Autenticação	43
Figura 15 – Dados para requisiçāoo de acesso ao aplicativo.	43
Figura 16 – Resposta da API a requisição de acesso ao aplicativo.	44
Figura 17 – Aplicativo To No Lote - Cadastro do Lote	44
Figura 18 – Aplicativo To No Lote - Lotes à venda	45
Figura 19 – Prototipação 001	49
Figura 20 – Prototipação 002	49
Figura 21 – Prototipação 003	50
Figura 22 – Prototipação 004	50
Figura 23 – Prototipação 005	51

Lista de abreviaturas e siglas

API - Application Programming Interface

HTTP - Hyper Text Transfer Protocol

JSON - JavaScript Object Notation

SO - Sistema Operacional

SOAP - Simple Object Access Protocol

UI - User Interface

URI - Uniform Resource Identifier

URL - Uniform Resource Locator

WADL - Web Application Description Language

WSDL - Web Service Description Language

XML - eXtensible Markup Language

APP - Aplicativo.

AOT - Ahead Of Time.

JIT - Just In Time.

MV - Máquina Virtual.

UI - User Interface.

GPS - Geographic Position System.

GUI - Graphical User Interface.

UML - Unified Modeling Language.

SDK - Software Development Kit.

Sumário

1	INTRODUÇÃO	13
1.1	Objetivos	14
1.1.1	Objetivo Geral	14
1.1.2	Objetivos Específicos	14
2	REFERENCIAL TEÓRICO	15
2.1	Vendas pela internet	15
2.2	Aplicações móveis	15
2.2.1	Desenvolvimento de Aplicações Nativa	16
2.2.2	Desenvolvimento de Aplicações Multiplataforma	17
2.3	Flutter	17
2.3.1	Dart	19
2.3.2	Widgets	19
2.4	Laravel e Arquitetura MVC	20
2.4.1	Arquitetura MVC	21
2.5	Web Services	22
2.6	Geolocalização e Here	23
3	METODOLOGIA	26
3.1	Intervenção Teoria/Prática	30
4	RESULTADOS	33
4.0.1	Levantamento de Requisitos	33
4.0.2	Diagramas	34
4.0.2.1	Diagramas de Caso Uso	34
4.0.2.2	Diagrama de Classe	35
4.0.3	Arquitetura	38
4.0.4	API	38
4.0.5	Aplicativo To No Lote	41
5	CONCLUSÃO	46
5.1	Trabalhos Futuros	46
	REFERÊNCIAS	47
6	APÊNDICE A PROTOTIPAÇÃO	49

7	APÊNDICE B REQUISITOS FUNCIONAIS	52
8	APÊNDICE C ESCOPO DO PROJETO (BACKLOG)	53

1 Introdução

Estamos discutindo a respeito de um processo mais comum em nossa sociabilidade. Isto é, vamos falar da compra ou venda de um lote, terreno, casa e afins. Comprar um lote em algum momento da vida faz parte das metas de nós, homens e mulheres. Todos buscam o conforto e posteriormente a realização dessa meta de usufruir da “casa própria”. Essa vivência comum dos brasileiros mantém o mercado de lotamentos sempre movimentado e aquecido. Quanto a esse contexto econômico nacional do seguimento de loteamento, há números expressivos. Conforme levantamento da Brain¹ e CBIC de 2021 para 2022 ocorreu uma queda de 8,6%, respectivamente. Todavia houve recuperação dessa taxa, como mostra a mais recente pesquisa realizada pela própria Brain em conjunto com ADIT Brasil para projetar o ano de 2023, o resultado mostra que 62% deste setor está otimista e com boas perspectivas de crescimento.

Ao optar por adquirir um lote por meio de uma imobiliária, o cliente tem a opção de efetuar o pagamento à vista ou parcelado. No caso do pagamento parcelado, surgem duas situações comuns que podem ocasionar ocorrências: o distrato (devolução do lote) ou o próprio cliente vender a parte já quitada até aquele momento para um novo interessado.

Havendo o distrato, este cumpre a Lei dos Distratos² (Nº 13.786) editada e sancionada em 2018. Esta lei permite que a imobiliária responsável pelo loteamento retenha 10% do valor pago e que a restituição do saldo possa ser feito em 12 parcelas mensais para o cliente. Na situação em que o cliente deseja se desfazer do lote por conta própria ele precisa encontrar um novo comprador. Ambos precisam chegar a um acordo de venda e realizar a troca para o novo responsável do lote junto a imobiliária.

A decisão do cliente pela opção do distrato muitas vezes vai parar na justiça e demora anos para ser julgada. Tornando o processo de devolução lento e podendo aumentar os custos com a contratação de advogados. Todo esse transtorno agrava na decisão do cliente em optar por muitas vezes vender o que já foi pago a imobiliária por um valor abaixo, para um outro interessado.

Nesse momento o cliente acaba enfrentando uma série de contratemplos. Ele precisa ter informações recentes sobre as constantes mudanças de evolução no loteamento, realizar negociações a qualquer momento do dia com os interessados e ter que divulgar o anúncio da venda para o maior número de pessoas. Nota-se que a venda de lotes é uma tarefa, recorrentemente observada, como algo simples, todavia é complexa e trabalhosa para todos os envolvidos. Além disso o seu meio de realização tradicional pode tomar muito tempo e

¹ <https://brain.srv.br/>

² https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13786.htm

demandar conhecimentos específicos.

Portando esse trabalho projeta o desenvolvimento de uma aplicação móvel, multiplataforma, com acesso livre e que possibilite a criação de anúncios para a venda de lotes. A ferramenta irá contribuir com informações que agregue valor ao bem e auxilie o anunciante a estimar o valor de venda. A saber o trabalho, se organiza, a partir desta breve introdução, depois contextualiza a noção do aplicativo, filtra os objetivos deste estudo, avança no referencial teórico, apontando as principais tecnologias e conceitos envolvidos no processo de desenvolvimento do trabalho proposto.

Por consequinte explica a metodologia utilizada para resolução do objeto e objetivo do estudo apresentado, além de como os testes foram realizados; por fim, apresentam-se os resultados e considerações finais, respectivamente.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver um aplicativo multiplataforma com geolocalização para mapear terrenos/lotes a serem vendidos, por seus próprios donos, sem interferência de agentes imobiliários, facilitando esse tipo de serviço neste setor.

1.1.2 Objetivos Específicos

- Modelar e especificar a arquitetura necessária operação do aplicativo.
- Desenvolver os Requisitos Funcionais e Não Funcionais, Diagrama de Caso de Uso e Diagrama de Classes.
- Construir a prototipação para o desenvolvimento do aplicativo.
- Construir um *Web Service* para uso do aplicativo.

2 Referencial Teórico

Neste capítulo estão apresentados o conceitos e contextualização das aplicações e ferramentas utilizadas no desenvolvimento deste trabalho.

2.1 Vendas pela internet

Os consumidores estão cada vez mais habituados a fazerem suas compras pela internet. A evolução da tecnologia, o alcance dos aplicativos práticos, e com confiabilidade e segurança, contribuem efetivamente para essa rotina. Conforme pesquisa da ABCOMM¹ (Associação Brasileira de Comércio Eletrônico) ocorreram aproximadamente 368,7 milhões de pedidos online no ano de 2022. As compras pela internet, só crescem, logo ainda com base nos dados da ABBCOMM neste ano, as vendas podem chegar em 395 milhões de pedidos online.

Com base nos estudos de Moura e Faria (2019), quando utilizam os aplicativos de compra e venda os consumidores buscam informações que disponibilizam conhecimento amplo em relação ao que pretendem adquirir. Dessa forma os aplicativos com visualização de parâmetros permitem qualificar os produtos ou serviços. Isso garante ao consumidor poder de decisão e segurança na efetivação das suas compras. O consumidor deseja ter acesso a aplicativos que ofereçam boas condições para suas compras. Pontos importantes como praticidade, segurança e objetividade nas ações dentro dos aplicativos devem sempre ser atendidos.

2.2 Aplicações móveis

Zammetti (2020), explica que “criar” aplicações para dispositivos móveis que fornecem aos usuários praticidades e funcionalidades que favoreçam uma utilização otima dos recursos disponíveis pela plataforma - é uma tarefa complexa. Usuários buscam aplicativos com interfaces intuitivas, de boa integração dos recursos como câmera, localização e internet.

Nesse sentido o autor Lecheta (2015), reflete que os usuários qualificam a importância de mobilidade com segurança para sistematizar positivamente suas tarefas do dia a dia. O principal desafio é combinar o desenvolvimento de interfaces agradáveis, associadamente com a melhor utilização das funcionalidades disponíveis no *hardware* do dispositivo móvel.

¹ <https://abcomm.org/noticias/category/numeros-do-e-commerce/>

Com o lançamento do iOS SDK pela Apple em 2008 e do Android SDK pela Google em 2009, era natural criar um código para cada plataforma.

É importante ressaltar que ao longo do tempo, apesar do domínio do mercado de dispositivos móveis por essas duas plataformas, surgiram outros concorrentes que acabaram por desaparecer os conhecidos sendo o *Windows Phone*, *BlackBerry* e *Symbian*. Durante sua existência, era necessário criar um código específico para que a aplicação pudesse ser utilizada em cada uma dessas outras plataformas. Desse modo “criar”, segundo [Zammetti \(2020\)](#) para cada plataforma do mercado é uma alternativa para alcançar melhor usabilidade e desempenho da aplicação móvel. Mas isso significa desenvolver a mesma coisa por mais de uma vez gerando custos e desperdício de tempo. Portanto é necessário que o programador, compreenda, além de conhecer quanto a “analisar” qual a melhor solução para o desenvolvimento da aplicação.

Dessa forma analisaremos os dois principais métodos de desenvolvimento de soluções móveis: nativo e multiplataforma.

2.2.1 Desenvolvimento de Aplicações Nativas

O autor [Constantin N. O. \(2016\)](#), explixa que o desenvolvimento nativo é constituído basicamente com a criação de aplicações para determinados sistemas operacionais. Consequentemente esse tipo de solução define as ferramentas e a linguagem de programação apropriada para cada plataforma . Ainda nessa direção [Zammetti \(2020\)](#) comenta que o desenvolvimento de aplicações nativas para o iOS SDK é na linguagem Objective-C, já está sendo preterida pela linguagem Swift. No desenvolvimento de aplicações para plataforma Android a linguagem principal é Java, seguida por Kotlin e C#. Destacamos que [Redda \(2012\)](#) alerta que as aplicações que precisam ter acesso as funcionalidades como câmera, GPS, acelerômetro, bússola, contatos, e diversas outras, tendem a adotar o desenvolvimento nativo.

As aplicações nativas são arquivos binários executáveis que após instalada no dispositivo não há necessidade da camada de abstração para acesso as funcionalidades e recursos do hardware. Nesse sentido [EL-KASSAS \(2015\)](#) comenta que a aplicação é executada e interage de forma rápida e eficiente com os recursos do dispositivo proporcionando melhor experiência de usabilidade para o usuário.

Desenvolver uma aplicação nativa de acordo a cada plataforma torna-se um processo caro, devido a formulação “escrever” um mesmo código para cada linguagem apropriada, ressalta [Charland A. e Leroux \(2011\)](#). Aspectos, a saber, como o tempo e equipe de desenvolvimento com conhecimento especializado abrangendo a linguagem específica de cada plataforma devem ser bem analisados na adoção desse tipo de solução. Outro ponto que [Charland A. e Leroux \(2011\)](#) trata de ser o grande desafio quando a necessidade é

atingir maior número de usuários estando a aplicação disponível em uma única plataforma. Dessa forma é válido acompanhar a transformação e evolução dos sistemas operacionais mais utilizados nos dispositivos móveis, que são Android e iOS.

A plataforma GS Statcounter², em 2023, disponibiliza quais sistemas operacionais estão mais presentes nos dispositivos móveis. No Brasil o Android está operando em 85% dos aparelhos enquanto o iOS representa aproximadamente 14% desse mercado.

2.2.2 Desenvolvimento de Aplicações Multiplataforma

Aplicações multiplataformas são soluções criadas para serem executadas em diferentes sistemas operacionais e dispositivos. Essa abordagem oferece benefícios significativos, como a economia de tempo e recursos ao desenvolver e manter um único código-fonte para diferentes plataformas. A abordagem de desenvolvimento de aplicações multiplataforma têm o potencial de atingir um público mais amplo, sem a restrição de dispositivo, sejam eles smartphones, tablets ou computadores. El-Kassas et al. (2017), relata que, tornando-se uma opção valiosa para empresas e desenvolvedores, talvez, faltando uma eficiência maior, todavia em busca de alcance mais amplo. Durante a compilação das aplicações multiplataformas acontece a interpretação da linguagem desenvolvida para a linguagem nativa da plataforma.

Outro estudo do tema PONTES (2017), afirma que, o código e o aplicativo são gerados como nativo para ser executado na plataforma escolhida. Uma arquitetura multiplataforma adota o uso de camadas abstratas e interfaces para encapsular as funcionalidades específicas da plataforma. Com isso, o código principal é compartilhado entre as plataformas, enquanto as camadas de interface e adaptação lidam com as especificidades de cada sistema operacional.

A respeito desse movimento Corazza (2018), destaca que, por ocorrer a complexidade do desenvolvimento nativo os frameworks multiplataformas saem em vantagem na escolha como solução. Principalmente por permitir desenvolvimento de um único código base e acesso aos recursos nativos das plataformas, além de reduzir os custos, tempo e conhecimento da solução adotada.

2.3 Flutter

Conforme a literatura estudada, Flutter³ é um framework de desenvolvimento de aplicativos multiplataforma criado pelo Google. Com o Flutter, os desenvolvedores podem criar aplicativos nativos para iOS, Android, web e desktop usando uma única base de

² <https://gs.statcounter.com/os-market-share/mobile/brazil/#monthly-202205-202205-bar>

³ <https://flutter.dev/>

código. Outrossim usa a linguagem de programação Dart, que oferece um desempenho rápido, juntamente com recursos avançados de desenvolvimento, comenta [Zammetti \(2020\)](#).

O Flutter oferece uma vasta coleção de widgets personalizáveis e uma arquitetura reativa, permitindo a criação de interfaces de usuário bonitas e fluidas. Flutter apresenta um próprio mecanismo de renderização, assim, não é necessário uma ponte específica para cada sistema operacional executar a aplicação. Por isso afirma [Wu \(2018\)](#), que a aplicação final é compilada para Android usando o NDK (*Native Development Kit*) e para iOS o LLVM (*Low Level Virtual Machine*). Todo o código desenvolvido em Dart é compilado para código nativo por AOT (*ahead of time*) em tempo de desenvolvimento. Vejamos a figura abaixo:

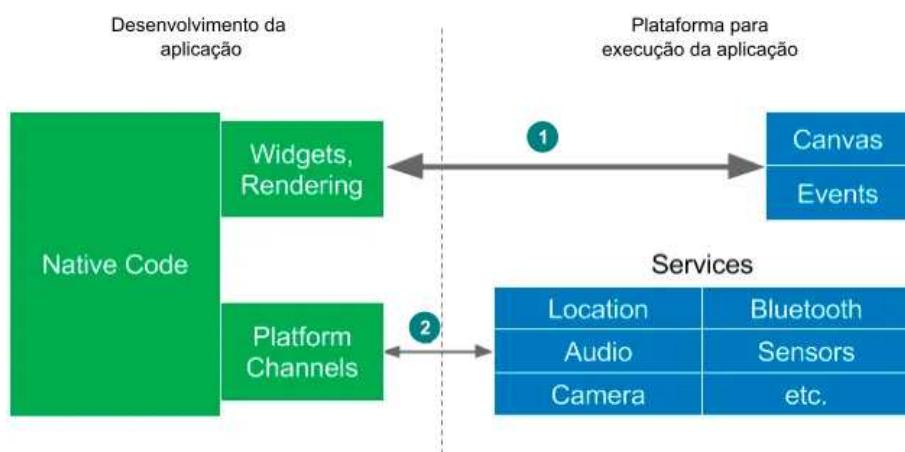


Figura 1 – Arquitetura Flutter.

Fonte: Autor, 2023

A [Figura 1](#) mostra de forma resumida a interação do que está sendo desenvolvendo para ser executado na plataforma escolhida. O ponto 1 indica que os componentes de interface (Widgets, Rendering) são desenhados na tela (*Canvas*) e recebem interações (*Events*) sem necessidade de um intermediador. O mesmo indica o ponto 2, onde o acesso aos recursos nativos na plataforma (*Services*) são feitos de forma direta.

Frameworks similares com intermediários de código para a plataforma enfrentam desafios como tráfego intenso de dados e mudanças de contexto. [Lima \(2019\)](#), em seus estudos destacam que, apesar das melhorias nas pontes desses *frameworks*, problemas podem surgir ao renderizar animações personalizadas, lidar com grandes arquivos e dados JSON, e exibir muitos dados em um mapa.

Além disso, o Flutter possui uma comunidade ativa. Assim como uma extensa biblioteca de pacotes, facilitando a implementação de recursos adicionais nos aplicativos. Com sua abordagem de desenvolvimento rápido, o Flutter passou a ser uma escolha popular para desenvolvedores que desejam criar aplicativos multiplataforma modernos e de alta qualidade.

2.3.1 Dart

Dart é uma linguagem de programação desenvolvida pelo Google, que foi projetada para criar aplicativos web, móveis e de desktop. Ela possui uma sintaxe familiar e amigável para os desenvolvedores. O Dart oferece recursos avançados, como tipagem estática opcional, o que permite detectar erros em tempo de compilação e melhorar a qualidade do código. [Zammetti \(2020\)](#), explica que o Dart possui um mecanismo de coleta de lixo eficiente, que gerencia automaticamente a alocação e liberação de memória. Fornece também suporte para ferramentas populares de desenvolvimento com *Visual Studio Code* e *IntelliJ*.

O Dart foi criado com características próximas ao padrão do *JavaScript*, utilizando métodos como "*async*" e "*await*". Na intenção de atrair os grupos de desenvolvedores adotou uma sintaxe semelhante à do Java. A saber, quando pensados na sua velocidade de execução e facilidade de uso, o Dart acaba sendo uma escolha popular para o desenvolvimento de aplicativos multiplataforma, especialmente quando combinado com o *framework* Flutter.

2.3.2 Widgets

Widgets no Flutter são os blocos de construção fundamentais para construir interfaces de usuário. Eles representam elementos visuais e funcionais, como botões, caixas de texto, imagens. Os *widgets* no Flutter são altamente personalizáveis e reutilizáveis, permitindo a criação de interfaces dinâmicas e interativas. [Lima \(2019\)](#) comentam em seus textos que são elementos essenciais em um aplicativo Flutter, isto é, passam a ter função pela aparência e interação direta com o usuário. São eles responsáveis pelas ações do usuário, controlando e respondendo os comandos. Dessa forma é fundamental que os *widgets* correspondam com desempenho rápido na renderização e animação, garantindo uma experiência fluída e responsiva.

Percebendo que existem classes, [Zammetti \(2020\)](#), indica que o *widgets* será estendido de uma classe padrão fornecida pelo Flutter, determinando o tipo de widget em questão. Em geral, existem dois tipos principais: *StatelessWidget* e *StatefulWidget*. O primeiro indica que não há mudança de estado, enquanto o segundo indica mudanças de estado conforme a interação com o *widgets*. O autor ainda acrescenta que ao formular aplicativos utilizando esses componentes, é criada uma hierarquia denominada "árvore de widgets". A maioria dos *widgets* atua como contêineres (pais) e pode conter um ou mais componentes internos (filhos). Esses filhos, por sua vez, podem ter seus próprios componentes internos, ampliando ainda mais essa estrutura hierárquica.

Sendo assim o Flutter oferece uma vasta coleção de *widgets* prontos para uso, facilitando o desenvolvimento de aplicativos com uma aparência nativa. Com sua abordagem baseada em *widgets*, o Flutter simplifica o processo de criação de interfaces de usuário intuitivas e responsivas para aplicativos.

2.4 Laravel e Arquitetura MVC

Gabardo (2017) em sua pesquisa comenta que Laravel⁴ é um popular framework de desenvolvimento web em PHP, com destaque para eficiência e facilidade de uso. Ele segue os princípios da arquitetura MVC (*Model-View-Controller*), o que facilita a organização e manutenção do código. Ao criar um projeto utilizando o framework Laravel uma estrutura padrão de pastas é criada. Isso serve para organizar e estruturar o código-fonte do aplicativo. Cada pasta tem um propósito específico e abriga diferentes componentes do projeto, como modelos, controladores, visualizações, configurações, rotas e recursos. O autor explica, além, que o Laravel oferece uma ampla gama de recursos e bibliotecas poderosas, incluindo autenticação, gerenciamento de bancos de dados, roteamento e cache. Com seu sistema de *templating Blade*, o Laravel permite a criação de interfaces de usuário dinâmicas e reutilizáveis.

O Laravel adota o padrão MVC para a construção de sua arquitetura de projeto e para gerenciar o fluxo interno das aplicações desenvolvidas. Ao receber uma requisição HTTP, a comunicação ocorre por meio das seguintes camadas: A) *Routes* (rotas): mapeia a requisição e encaminha a mesma para o *controller* correspondente; B) *Controllers* (controladores): recebem as requisições identificadas nas rotas, processam de acordo com as regras de negócio da aplicação. Os dados processados nessa camada são encaminhados para os modelos; C) *Models* (modelos): são os objetos definidos dentro da aplicação. Dessa forma os dados são carregados conforme sua estrutura (atributos e métodos) e encaminhados para a camada de visão; D) *Views* (visão): nessa camadas os resultados das requisições identificadas nas rotas, processadas nos *controllers* e carregadas nos *models* para serem apresentados.

Ainda constam: E) App: pasta onde está localizado o projeto e toda sua estrutura interna de pastas; F) artisan: este arquivo é executado a partir do diretório raiz (pasta app) com o comando "*php artisan*" e que oferece uma série de comandos embutidos. Exemplos: criar códigos de controladores, modelos; Utilizar o sistema de migração para criar e gerenciar banco de dados; Executar o seeders para popuar bancos testes ou os dados inciais; G) composer.json: aquivo para gentenciar as dependências do projeto (pacotes e bibliotecas); H) config: arquivo oara cinfurações bases. Exemplo: dados de acesso ao baco de dados; database: pasta onde estão localizadas as classes para interações com banco de dados (migrations, factories e seeds); I) .env: arquivo usado para armazenar as configurações específicas para o desenvolvimento da aplicação. É carregado automaticamente na inicialização da aplicação e possui flexibilidade no ajustes para ambiente de desenvolvimento, produção e teste. E por fim J) storage: pasta usada para armazenar dados gerados pela aplicação, como uploads de usuários, caches, logs e sessões. Local centralizado para gerenciar esses dados.

⁴ <https://laravel.com/>

O Laravel possui uma comunidade ativa e uma documentação abrangente, o que torna mais fácil para os desenvolvedores aprenderem e colaborarem. Por isso o Laravel se destaca como uma escolha excelente para a construção de APIs (*Application Programming Interfaces*), em decorrência da facilidade de uso, recursos integrados e suporte a boas práticas de desenvolvimento de API.

2.4.1 Arquitetura MVC

Quando estudamos a Arquitetura MVC em função dsta pesquisa, implica dizer que temos por base os autores [Arcos-Medina, Menéndez e Vallejo \(2018\)](#). Estes inferem que os padrões de projetos são soluções para problemas comuns no desenvolvimento de aplicações. Tem por objetivo definir soluções reutilizáveis para problemas recorrentes no desenvolvimento de aplicações. Fornecem abordagens testadas e comprovadas para resolver desafios recorrentes no desenvolvimento. Os padrões de projeto visam melhorar a estrutura, flexibilidade, modularidade e manutenibilidade de um sistema.

Outrossim destacam que, os padrões são representados por meio de templates para garantir uma comunicação clara e efetiva para toda a equipe de desenvolvimento da aplicação, sem gerar espaço para dúvidas e desperdício de trabalho. Estes padrões exemplificam, elementos como nome, tipo, objetivo, motivação, aplicabilidade, estrutura, participantes, colaborações, consequências, implementação, código, usos conhecidos e padrões relacionados.

[Thakur e Pandey \(2019\)](#), argumentam que a arquitetura MVC (*Model - View - Controller*) é um padrão de projeto amplamente utilizado no desenvolvimento de aplicações. Essa arquitetura oferece um framework para desenvolvimento de aplicações proporcionando uma arquitetura eficiente e automação do código. O MVC é considerado uma das principais arquiteturas de desenvolvimento, facilitando a criação de aplicações bem estruturados e gerenciaveis.

A arquitetura separa a lógica de negócio (*Model*), a interface de usuário (*View*) e o controle das interações (*Controller*). Além do que, essa arquitetura é compreendida por [Silval, Flores e Viegas \(2019\)](#) como uma solução definida para separar o projeto em partes distintas e reduzir suas dependências. Utilizar um padrão de desenvolvimento pode trazer como benefícios o aumento da produtividade, facilitar a documentação, reduzir o tempo de desenvolvimento e diminuir a complexidade do código.

A arquitetura MVC é dividida em três camadas, que são a model, view e controller. Abaixo estão descritas as responsabilidades de cada uma das camadas do modelo MVC. Vejamos as camadas: A) Model (modelo) é onde estão as classes que modelam os dados para o sistema. De forma mais técnicas são as regras de negócio que o sistema deve obedecer; B) View (visão) é a representação visual dos dados para que o usuário consiga

interagir com aplicativo sem perceber o que acontece nas outras camadas e C) Controller é a responsável por gerir as ações que acontecem no aplicativo. Nessa camada é onde ocorrem as validações dos dados do modelo.

Analizando a [Figura 2](#) temos a representação da divisão das camadas e como ocorrem interações na arquitetura MVC. As ações realizadas na aplicação (*View*) pelo usuário são processadas pelo controlador (*Controller*) utilizando-se das regras de negócio e objetos definidos pelo modelo (*Model*).

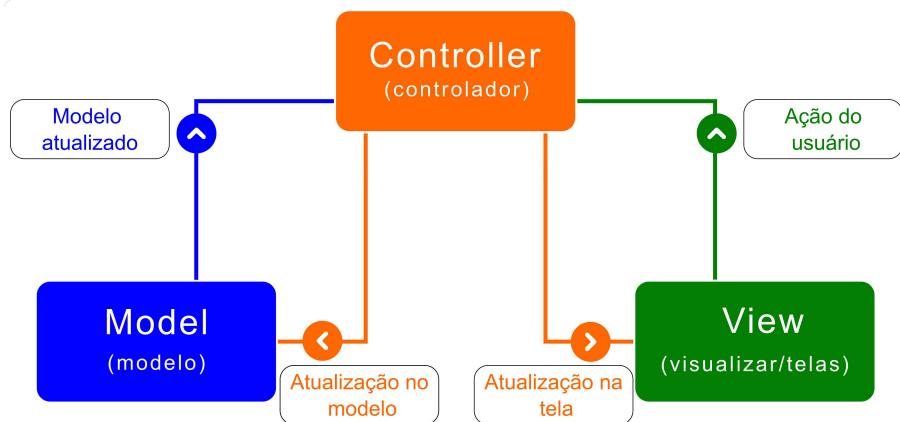


Figura 2 – Arquitetura MVC.

Fonte: Autor, 2023

2.5 Web Services

Sobre este elemento o autor [Lecheta \(2015\)](#) diz que Web Service é uma solução que simplifica a interoperabilidade entre as diferentes plataformas. Ele é utilizado por uma questão de segurança e padronização no acesso aos dados. É aplicável para os diferentes sistemas pois independe da linguagem de programação que o servidor ou cliente estão trabalhando. Embora semelhantes, as APIs (Application Programming Interface) e os web services têm diferenças. Web services são softwares que se auto-disponibilizam na internet e padronizam a comunicação, enquanto as APIs são especificamente componentes de software que interagem uns com os outros usando o protocolo HTTP.

[Verma \(2019\)](#), destaca que, Web services são considerados menos frágeis e mais acoplados, enquanto as APIs proporcionam uma comunicação mais leve e independente entre cliente e servidor. Ambas as soluções servem como meios de comunicação e suportam o envio e recebimento de dados em formato JSON.

[Verma \(2019\)](#), acrescenta que isso resulta em um acoplamento mais fraco e maior flexibilidade entre os sistemas cliente e servidor, que podem ser implementados em diferentes linguagens e trocar cargas de dados em formatos como JSON ou XML. Essa abstração e flexibilidade tornam as APIs da web mais robustas e versáteis. O conceito

de web services têm sido adotado para criar APIs e facilitar a integração de sistemas em diferentes plataformas e tecnologias. Usando o protocolo HTTP permitindo que o cliente utilize comandos como "GET" para solicitar dados sem precisar conhecer os detalhes de implementação no servidor.

A [Figura 3](#) será utilizada para explicar o comportamento de uma API web e sua composição. A API é composta por um banco de dados (ponto 1) e pela própria API (ponto 2). O ponto 1 indica o local de armazenamento dos dados. O ponto 2 tem como papel gerenciar as solicitações chegadas via internet pelo protocolo HTTP (*Hypertext Transfer Protocol*). Na API as requisições (*request* e *response*) são tratadas e entregues para a aplicação ou executadas para manipulação do banco de dados. A aplicação faz solicitações a API e processa por ela um arquivo XML (*Extensible Markup Language*) ou JSON (*JavaScript Object Notation*).

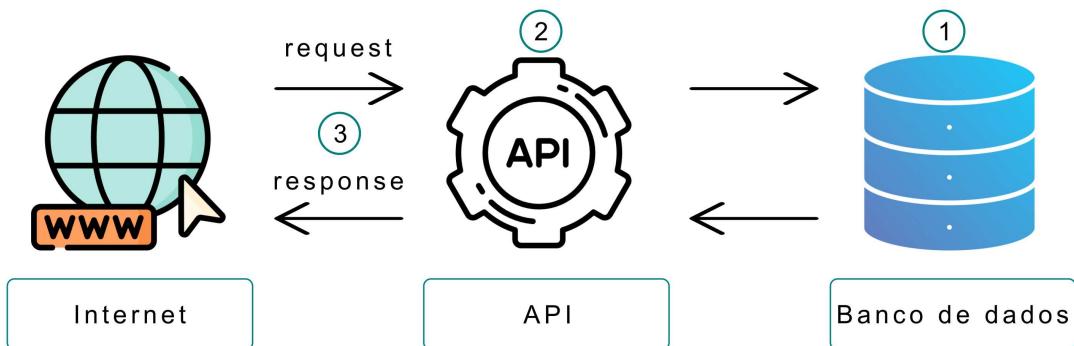


Figura 3 – API.

Fonte: Autor, 2023.

O formato do arquivo retornado pelo web service dependerá da arquitetura utilizada para seu desenvolvimento. As duas mais utilizadas pelas aplicações são a WSDL (*Web Services Description Language*) e REST (*Representational State Transfer*). Segue a arquitetura: A) WSDL É escrito em XML e utiliza HTTP para trafegar o dados com o protocolo SOAP. Essa arquitetura normalmente não é aplicada em aplicações com conexões instáveis (ex.: 3G). O arquivo XML é muito extenso o que pode causar muito consumo de dados e lentidão na rede; B) REST Baseia-se no protocolo HTTP e por isso possui métodos bem definidos - GET, DELETE, POST e PUT. Seu retorno pode ser em um arquivo JSON, XML ou RSS, sendo o desenvolvedor responsável por utilizar o mais adequado.

2.6 Geolocalização e Here

Com base em [Mittelstadt \(2018\)](#), a geolocalização é uma tecnologia presente na maioria dos dispositivos móveis para fornecer a posição geográfica. Um dos objetivos da geolocalização é disponibilizar informações em tempo real para os usuários. Essas

informações podem ser a situação do trânsito, encontrar serviços ou lojas, localizar o próprio usuário, e outras funcionalidades. A geolocalização gera dados, solicita permissões para monitoramento da movimentação dos dispositivos, o que favorece a interpretação de hábitos e costumes dos usuários. Abaixo, na [Tabela 1](#), são apresentados métodos e como eles funcionam para capturar a geolocalização dos dispositivo.

Tabela 1 – Métodos de captação de Geolocalização

Método	Descrição
GPS	Processo ligado a circunstâncias do tempo, possui localização realizada por satélite, captando no mínimo o sinal de três satélites.
AGPS	Aperfeiçoamento do GPS, também recebe dados (posição dos satélites) de uma antena de telefonia celular. Também é conhecido como GPS Assistido.
Radiofrequência	Tem a localização executada através de ondas de rádio e utiliza as informações fornecidas pelas operadoras móveis. Nesse processo, mesmo com o GPS desativado é possível localizar o dispositivo, desde que ele esteja ligado e com sinal.
Wi-fi	Geolocalização indoor pode ser captada por meio das redes wi-fi públicos e privados.

Fonte: Autor, 2023

Quanto as Here Rest APIs⁵ são um conjunto de interfaces de programação de aplicativos que permitem o acesso e a integração dos serviços de localização e mapeamento fornecidos pela HERE Technologies. Essas APIs oferecem recursos como roteamento, geocodificação, cálculo de distâncias e tempos de viagem, além de informações de tráfego e pontos de interesse. Com as Here Rest APIs, os desenvolvedores podem criar aplicativos que aproveitam os recursos de localização avançados para fornecer experiências ricas e personalizadas aos usuários. Abaixo na [Tabela 2](#) serão listados alguns conceitos extraídos da documentação que são fundamentais para trabalhar com a Here Rest APIs.

Tabela 2 – Conceitos e métodos Here Rest APIs

Termo	Definição
Chaves API	Credenciais de segurança para usar no aplicativo com APIs HERE.
Geolocalização	Uma localização específica na superfície da terra, normalmente expressa em latitude, longitude e elevação em relação ao nível médio do mar.
GPS	Sistema de Posicionamento Global de propriedade dos Estados Unidos.
OAuth	Estrutura de autorização aberta. Protocolo padrão da indústria para autorização.

Fonte: adaptado de ([HERE, 2019](#)).

⁵ <https://developer.here.com/develop/rest-apis>

Conforme documentação oficial da [Here \(2019\)](#), O HERE SDK é uma ferramenta que utiliza dados da plataforma HERE e adota princípios de design modernos, como microserviços e componentes modulares. Ele oferece suporte para três plataformas populares: Android e iOS. Com o HERE SDK, os desenvolvedores têm acesso a recursos avançados de mapeamento e localização para criar aplicativos ricos e personalizados em diferentes sistemas operacionais móveis.

3 Metodologia

Para este trabalho foi implementada a metodologia de desenvolvimento Ágil Sacrum. [STOPA e RACHID \(2019\)](#), explicam que as metodologias ágeis para desenvolvimento de software surgiram como uma alternativa às metodologias tradicionais, também conhecidas como "metodologias pesadas". Essas abordagens ágeis tem por objetivo dar resposta às mudanças constantes no ambiente empresarial e aos avanços tecnológicos em ritmo acelerado. [Velásquez et al. \(2019\)](#) acrescenta que as metodologias tradicionais exigem um planejamento minucioso e detalhado de todo o trabalho antes de iniciar o desenvolvimento do produto. Na abordagem do Método Cascata um ponto de destaque é o prazo de entrega que costuma ser maior em comparação com outras abordagens. Seu método sequêncial exige que cada fase do processo de desenvolvimento deve ser concluído antes de passar para a próxima.

Acrescenta [Soares \(2004\)](#) o Scrum é uma metodologia ágil de gerenciamento de projetos que enfatiza a colaboração, a entrega iterativa e a adaptação contínua. É baseado em equipes auto-organizadas, objetivos curtos para serem alcançados e reuniões diárias de alinhamento do projeto.

O Scrum divide o projeto em incrementos menores chamados de sprints e promove uma abordagem flexível para lidar com requisitos em constante evolução. Dessa forma o Scrum é uma metodologia usada para gerenciar o trabalho desenvolvido em busca de um produto. Ainda com base em [STOPA e RACHID \(2019\)](#), este interage como um framework estrutural, permitindo a utilização de diferentes processos e técnicas. Essa metodologia é dividida em três aspectos fundamentais: transparência, inspeção e adaptação.

A [Figura 4](#) apresenta os pontos de atenção da abordagem da metodologia Scrum. A transparência envolve tornar as atividades visíveis e estabelecer uma definição clara e compartilhada para que todos compreendam rapidamente o que está acontecendo. A fase de inspeção requer inspeções regulares para acompanhar o progresso da Sprint e identificar variações indesejadas, sem prejudicar o andamento das atividades. Já a fase de adaptação visa corrigir imediatamente desvios inaceitáveis em um processo ou produto, minimizando variações e garantindo resultados aceitáveis.



Figura 4 – Bases do modelo Scrum.

Fonte: Autor, 2023.

[STOPA e RACHID \(2019\)](#) comentam que o Scrum trabalha geralmente com pequenas equipes que são flexíveis e de rápida aptaçao a mudanças de componentes. As equipes são compostas pelos seguintes elementos: A) Scrum master: garantir que as regras e boas práticas do Scrum sejam utilizadas durante o projeto, respeitando seus valores; B) Product owner: representa as partes interessadas, define as funcionalidades do produto e faz a gestão do backlog utilizando métodos ágeis; C) Time: são as equipes composta por profissionais que realizam seu trabalho de acordo suas áreas. São responsáveis pelas entregas das funcionalidades definidas dentro do backlog.

A metodologia Scrum é comumente utilizada no gerenciamento de equipes em projetos. Dessa forma neste trabalho aplicamos somente características específicas da metodologia de forma a alcançar os objetivos propostos.

A primeira característica da metodologia Scrum utilizada foi a definição de objetivos que levariam mais tempo para serem alcançadas (sprints). Nessa fase foram definidas tempo para estudo e apresendizado dos frameworks, leituras dos materiais bibliográficos e contrução do escopo do projeto. A partir dessas definições mais amplas as atividades internas de cada um desses objetos passam ser tratados de forma específica e pontual, quase sempre diariamente.

Quanto a segunda característica, se trata de reuniões rápidas/objetivas (daily scrum) que não ocorriam diariamente e sim nos encontros de orientações. Dessa forma, verificava-se encontrados problemas pontuais e indicados soluções para continuadade das atividades.

Outra característica importante da metodologia Scrum utilizada foi a definição do documento que apresenta uma visão geral do projeto. Esse documento é flexível e poderia sofrer alterações conforme o desenvolvimento avançava.

É importante pontuar que as alterações nesse documento são agregadoras para que o documento fosse sempre sólido e confiável. Ressaltando [STOPA e RACHID \(2019\)](#), o backlog do produto é o ponto central, consistindo nos registros dos requisitos do produto. Esses requisitos são coletados em reuniões com as partes interessadas (aluno

autor e professor orientador), priorizando as funcionalidades e necessidades da aplicação. O backlog no Scrum é uma lista priorizada de requisitos, funcionalidades e melhorias. O product owner refina e prioriza os itens considerando o valor de negócio e as necessidades dos interessados. O backlog é revisado e ajustado constantemente para garantir a entrega dos itens mais importantes primeiro.

Para garantia produtiva das fases desta pesquisa foram usados alguns softwares. São softwares de codificação, editores de texto, edição de imagens, criação de diagramas e para testes conforme os avanços do projeto. Elas permitem uma abordagem mais eficiente e organizada na elaboração do trabalho.

A figura [Figura 5](#) mostra o fluxo de avanço do projeto e quais ferramentas estavam inseridas naquele momento. A partir da entrada com anotações de reuniões o projeto avança paralelamente em três linhas de trabalho: linha de trabalho Aplicativo, linha de trabalho da API e linha de trabalho do TCC. As reuniões mais amplas foram trabalhadas na fase de prototipação, tanto na modelgame visual (figma) como na modelagem para codificação (astah).

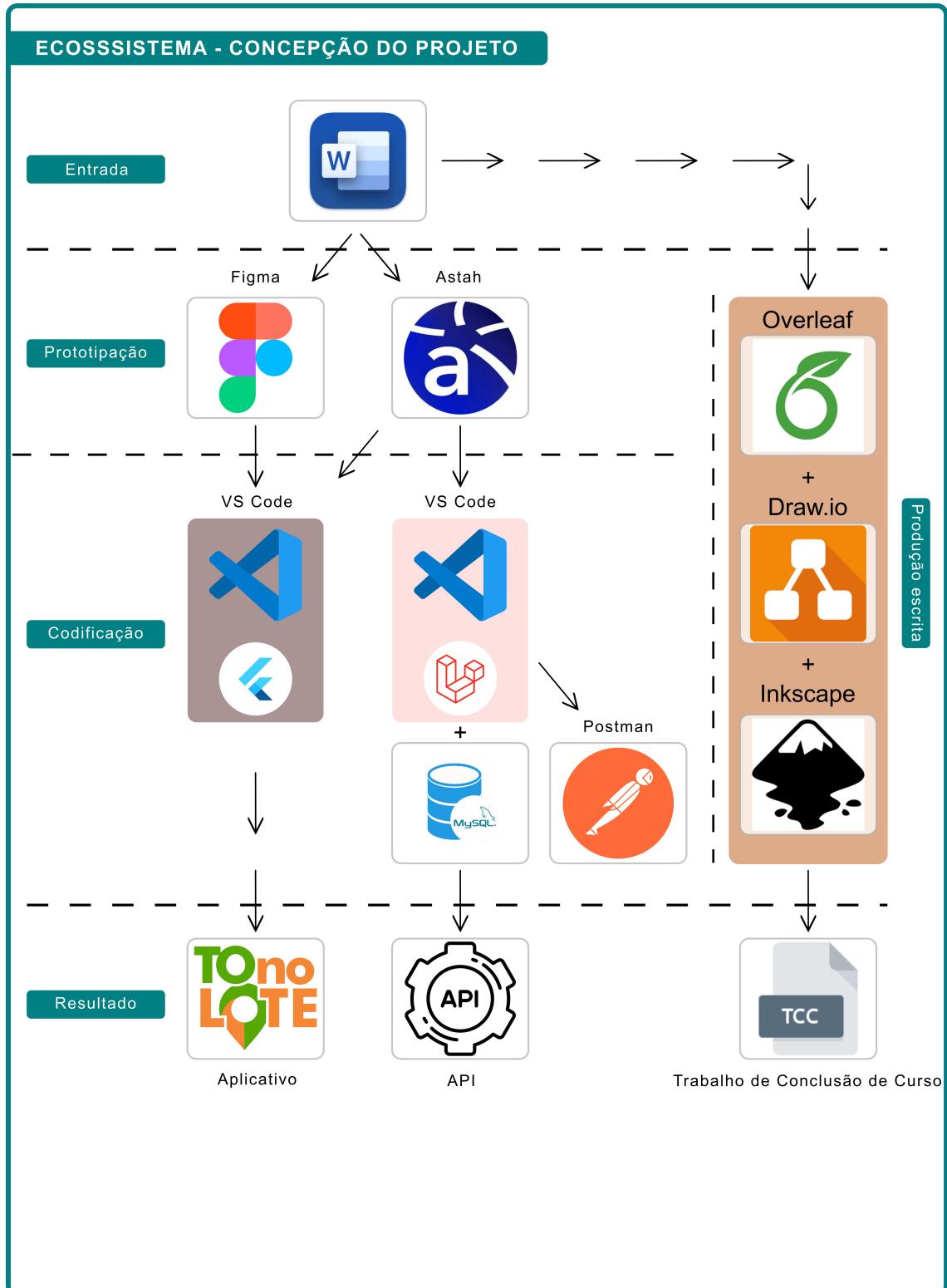


Figura 5 – Materiais.

Fonte: Autor, 2023.

Nessa fase de prototipação é definido o comportamento de navegação das telas e as estruturas internas para codificação que suporte as interações do aplicativo com a API. Na fase de codificação (utilizada como ferramenta base o VS Code) eram implementadas

as atividades internas das fases mais amplas definidas pela metodologia deste trabalho.

Como exemplo: criação da tela de autenticação, criação das telas de cadastros, criação da tela de visualização do mapa, indicar no mapa um lote por alfine e suas características. Paralelo a codificação das telas a API (backend) era construída para receber e tratar as interações feitas por cada tela do aplicativo. O processo de codificação ocorrer de forma paralela cooperando no ganho de tempo e realização de testes, pois dessa forma as linhas de trabalho Aplicativo e API não necessitaram a finalização total de uma para o começo da outra.

Para elaboração das documentações e tratamento dos resultados obtidos relacionadas ao projeto do aplicativo foram utilizados os softwares e suas versões descritas a seguir. Vejamos: A) Microsoft Word: processador de texto. Versão 2304 - Microsoft 365. Utilizado para anotação das orientações recebidas nas reuniões e dados do projeto; B) Figma: editor gráfico de vetor e prototipagem de projetos. Utilizado para prototipação do aplicativo que auxilia no processo de codificação; C) Astah: ferramenta de modelagem UML. Astah UML 9.0.0/1778f1 (Evaluation). Utilizado para criação do diagrama de classe e diagram de casos de uso que também auxiliam no processo de codificação; D) Visual Studio Code: editor de código-fonte desenvolvido pela Microsoft. Versão 1.79.2. Utilizado para codificação do aplicativo e da API; E) MySQL: sistema de gerenciamento de banco de dados. Versão 8.0.33. Utilizado como banco de dados para a API;

Acrescenta-se ainda esse: F) Postman: plataforma de API para desenvolvedores projetarem, construirão, testarem e iterarem APIs. Versão 10.15. Utilizado para testar os endpoints da API; G) Overleaf: editor LaTeX usado para escrever, editar e publicar documentos científicos. Utilizado para criação do documento científico TCC; H) Draw.io: software de desenho gráfico. Utilizado para criação das etapas de composição do desenvolvimento do projeto e I) Inkscape: software para edição de imagens e documentos vetoriais. Versão 1.2. Utilizado para criação e edição de imagens.

3.1 Intervenção Teoria/Prática

O projeto passou por uma sequência de etapas com objetivo de avanços incrementais, a fim de deixar evidente os avanços alcançados e destravar barreiras encontradas. Como apresentado na [Figura 6](#) as etapas conectam-se e avançam conforme há desenvolvimento e conclusão das atividades.

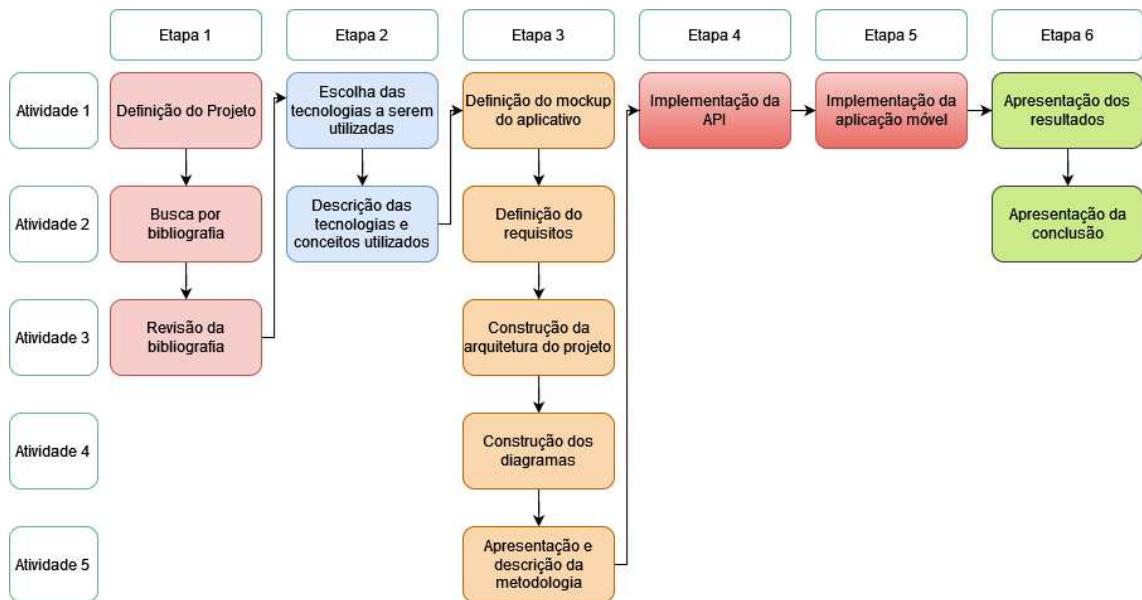


Figura 6 – Etapas e atividades do projeto.

Fonte: Autor, 2023.

A Etapa 1 consiste em três atividades que visam estabelecer a base bibliográfica do projeto, garantindo que os objetivos sejam alcançáveis e não extrapolando limites. Na Etapa 2, inicia-se a produção de material e documentos necessários para o projeto, incluindo a escolha e descrição das tecnologias/ferramentas a serem utilizadas. O foco é reconhecer os recursos utilizados no desenvolvimento do projeto. A Etapa 3 compreende diversas atividades, com o objetivo principal de produzir documentos claros e objetivos que serão utilizados nas próximas etapas. A qualidade e solidez do material produzido têm um impacto direto na continuidade do projeto.

As Etapas 4 e 5 envolvem a codificação do material desenvolvido na Etapa 3, utilizando frameworks para a construção das aplicações (API e Aplicativo), conforme a documentação previamente elaborada. A Etapa 6 resume-se na entrega do projeto, incluindo a apresentação dos documentos produzidos e do aplicativo como resultado final. O objetivo é garantir que o projeto atenda às especificações definidas e construídas ao longo de cada etapa. As etapas descritas foram composta por pequenas reuniões para sanar dúvidas, mudar rotas e redefinir os objetivos de cada atividade.

Abaixo são apresentadas tabelas das pautas de alguns objetivos definidos dentro das reuniões e implementadas neste projeto.

Tabela 3 – Reunião 1

Objetivo	Responsável
Definição da bibliografia para a escrita do projeto	Daniel Ferreira Rodrigues
Indicar materiais	Prof. Me. Silvano Maneck

Tabela 4 – Reunião 2

Objetivo	Responsável
Escrever referencial teórico	Daniel Ferreira Rodrigues
Avaliar definição das tecnologias	Prof. Me. Silvano Maneck
Avaliar escrita do referencial	Prof. Me. Silvano Maneck e Convidado

Tabela 5 – Reunião 3

Objetivo	Responsável
Avaliar mockup do projeto	Prof. Me. Silvano Maneck e Daniel Ferreira Rodrigues
Avaliar e corrigir requisitos	Prof. Me. Silvano Maneck e Daniel Ferreira Rodrigues

Tabela 6 – Reunião 4

Objetivo	Responsável
Desenvolvimento da metodologia	Daniel Ferreira Rodrigues
Avaliar e corrigir metodologia	Prof. Me. Silvano Maneck e Convidado

4 Resultados

Os resultados estão organizados de maneira clara, utilizando recursos visuais como imagens, tabelas e protótipos, quando adequado. Além disso, os resultados serão contextualizados, ressaltando suas implicações e contribuições na busca pelos objetivos estabelecidos.

4.0.1 Levantamento de Requisitos

Considerando o levantamento de requisitos é uma etapa muito importante no desenvolvimento de software, na qual são identificadas e documentadas as necessidades e expectativas dos usuários. É por meio do levantamento de requisitos que se obtém uma compreensão clara dos objetivos do projeto, dos recursos necessários e das restrições envolvidas.

Um levantamento de requisitos eficiente contribui para o sucesso do projeto, garantindo que as soluções desenvolvidas atendam às demandas dos usuários finais. O levantamento de requisitos é parte importante da composição do backlog pois é documento em que o desenvolvedor recorre com frequência para dar segurança no desenvolvimento correto das regras de negócio.

Tabela 7 – Requisitos Funcionais

ID	Descrição
RF001	Gerenciar usuário (cadastrar / editar / deletar)
RF002	Gerenciar lote (cadastrar / editar / deletar)
RF003	Permitir o usuário sem autenticação acessar o mapa com os lotes, porém com informações resumidas do lote
RF004	Permitir ao usuário acessar todas as informações do anúncio de venda do lote somente após autenticação no aplicativo
RF005	Permitir recuperação da senha somente através de confirmação de código enviado para o e-mail cadastrado pertencente ao usuário
RF006	Mostrar mapa da cidade e indicar os lotes através de alfinetes no mapa
RF007	Mostrar o mapa com filtro inicial por cidade apenas
RF008	Cadastrar lote com a indicação da coordenada longitude e latitude

Tabela 8 – Requisitos não Funcionais

ID	Descrição
RFN001	A aplicação móvel deve ser desenvolvida com <i>Framework</i> multiplataforma Flutter
RNF002	A API REST deve ser construída utilizando-se do <i>Framework</i> Laravel
RFN003	O arquivo de comunicação utilizado pela API deve ser do tipo JSON
RFN004	O banco de dados deve ser implementado com o MySQL

4.0.2 Diagramas

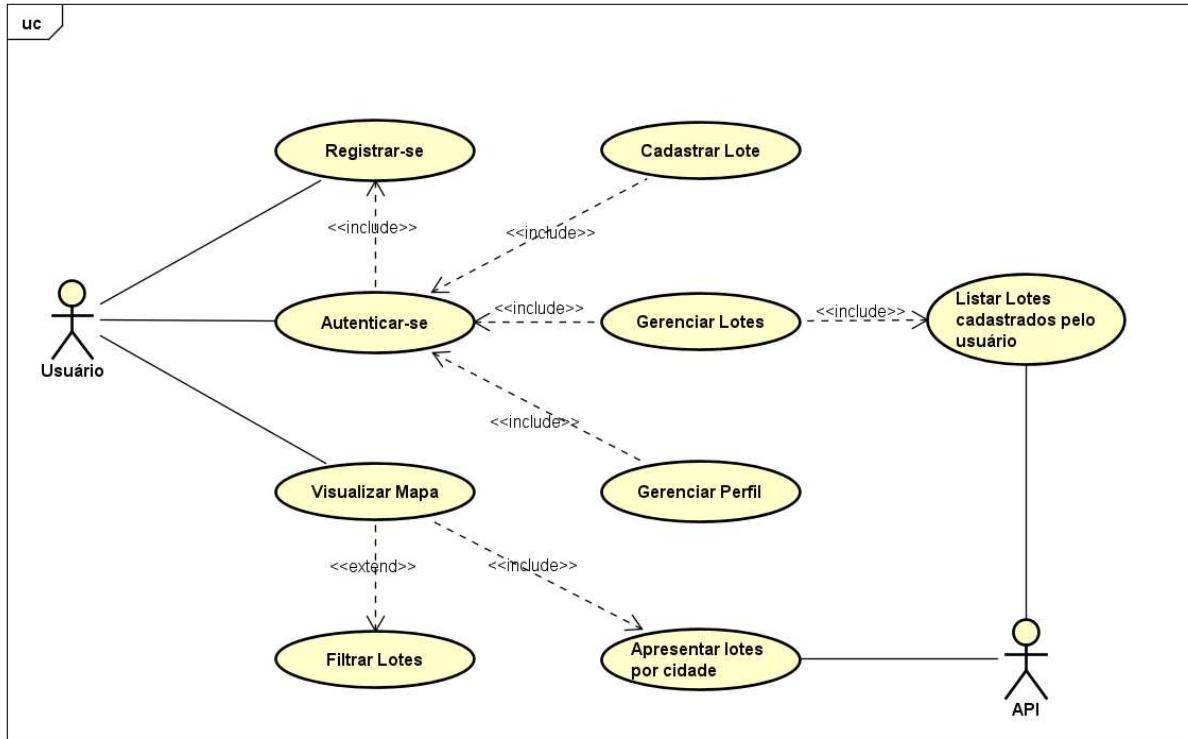
Quanto aos resultados de diagramas, percebemos através destes como deve ser o funcionamento do aplicativo a ser desenvolvimento e fornecer documentação consistente para consulta no momento da codificação.

4.0.2.1 Diagramas de Caso Uso

O diagrama de casos de uso UML é uma representação gráfica que descreve as interações entre atores e o sistema de software. Ele ajuda a identificar e visualizar as funcionalidades e os requisitos do sistema, mostrando os diferentes cenários de uso. O diagrama de casos de uso é uma ferramenta eficaz para capturar e comunicar os requisitos funcionais do sistema, auxiliando na compreensão dos fluxos de trabalho e na definição dos papéis dos usuários envolvidos.

Destaca-se que na [Figura 7](#) os atores presentes são o próprio usuário do aplicativo e algumas funções disponibilizadas pela API. O usuário pode acessar o mapa sem autenticação. O usuário cadastrado e autenticado pode fazer a gestão dos seus dados pessoais, além de cadastrar e alterar os dados dos lotes para anúncio de venda. A propriedade *include* indica que um caso de uso pode ser executado somente se caso de uso incluído a ele for executado. A propriedade *extend* indica que um caso de uso pode ser executado somente se outro caso de uso ligado a ele tiver sido executado.

Figura 7 – Diagrama de Caso de Uso



Fonte: Autor, 2023.

4.0.2.2 Diagrama de Classe

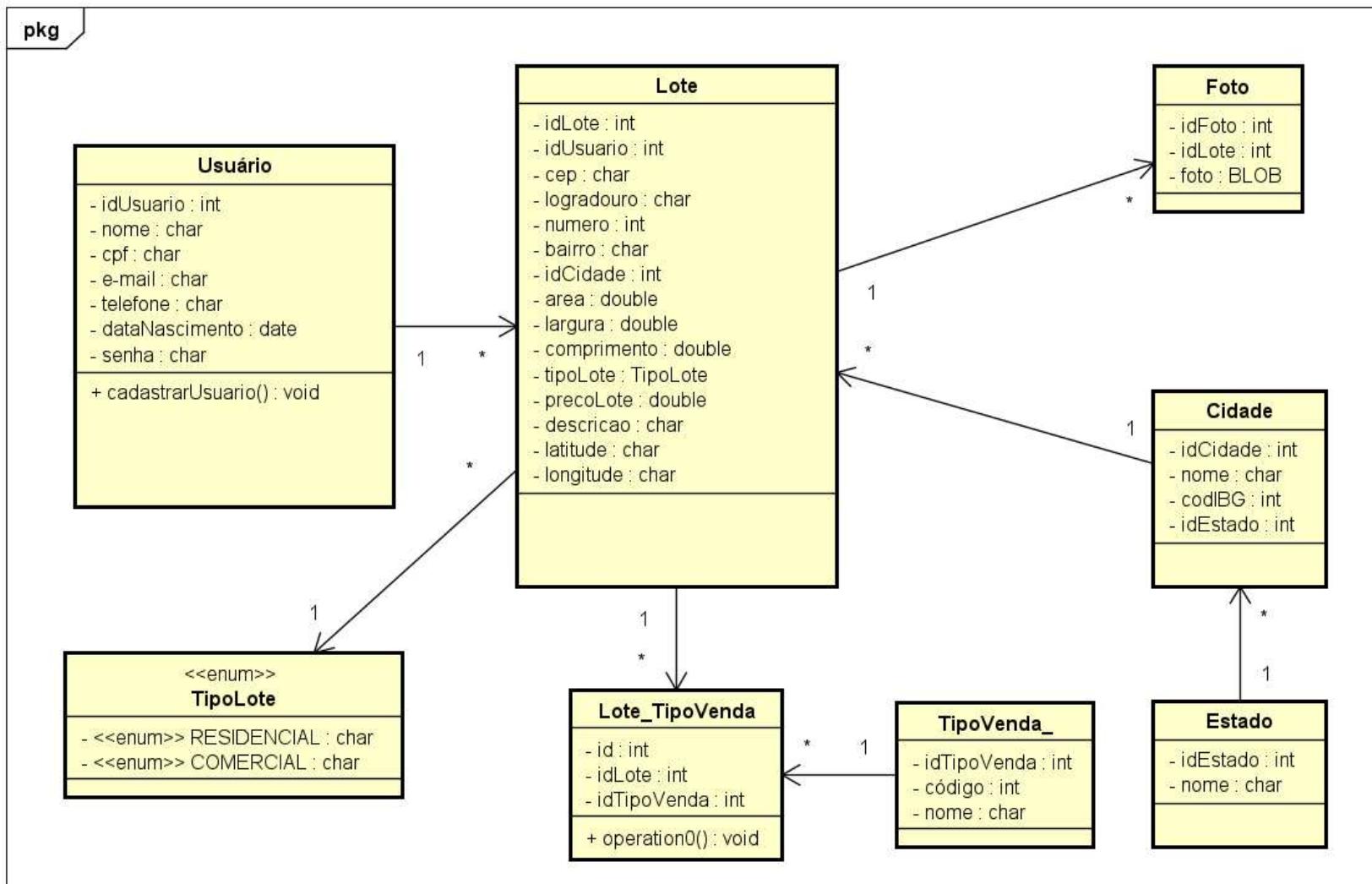
Os resultados também mostram o Diagrama de Classe. Vejamos. O diagrama de classes UML (*Unified Modeling Language*) é uma representação visual das classes, seus atributos, métodos e relacionamentos em um sistema orientado a objetos. Ele fornece uma visão estruturada e organizada da arquitetura do sistema, facilitando o entendimento e a comunicação entre os membros da equipe de desenvolvimento. O diagrama de classes UML é uma poderosa ferramenta para modelagem e design de sistemas, permitindo a análise e a visualização das relações e interações entre as classes.

Na [Figura 8](#) percebemos os elementos: A) Usuário: está classe contém as informações básicas do usuário. Os atributos e-mail e senha serão para autenticação e recuperação de senha; B) Lote: classe principal do aplicativo, contém todas as informações para construção do anúncio de venda do lote. Os atributos longitude e latitude serão trabalhados para agrupar informações para composição do anúncio; C) Foto: classe que contém as imagens cadastradas para cada o lote; D) Tipo Lote: este ENUM representa a classificação do lote, podendo ser comercial ou residencial.

Soma-se a esses os demais a seguir: E) Lote_Tipo Venda: esta classe representa a ligação da classe Lote com a classe Tipo de Venda. Um lote pode ter mais de um Tipo de Venda e o Tipo de Venda pode estar presente em mais de um Lote; F) Tipo Venda: esta classe representa o tipo de venda aceito para negociação do lote. Podendo ser informado

um ou mais de um tipo de venda, sendo eles: À VISTA, ÁGIO e TROCA; G) Cidade: esta classe armazena todas as cidades registradas no Brasil pelo Instituto Brasileiro de Geografia e Estatística, o objetivo é facilitar buscas de lotes em cada cidade no mapa e por fim, H) Estado: esta classe apenas indica os estados e quais cidades pertencem a ele, seguindo informações do registradas pelo IBGE.

Figura 8 – Diagrama de Classes

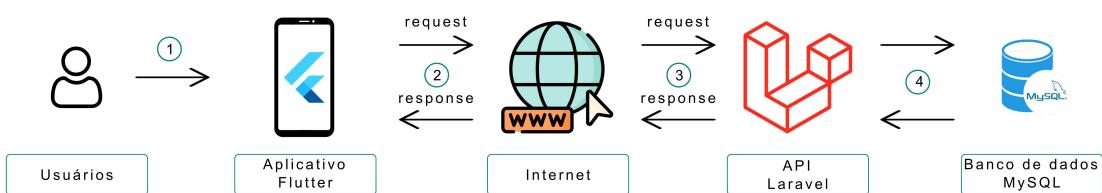


Fonte: Autor, 2023.

4.0.3 Arquitetura

Em relação aos resultados referentes a Arquitetura de Aplicação percebeu-se que após realizar o levantamento de requisitos e diagramas, deu-se a construção da arquitetura base para funcionamento do aplicativo proposto. A [Figura 9](#) apresenta a composição da arquitetura necessária a ser desenvolvida para atender as necessidades de todo o sistema. A interação entre os componentes da arquitetura ocorre em 4 etapas, que serão descritas em sequência na figura.

Figura 9 – Arquitetura da Aplicação



Fonte: Autor, 2023.

O passo 1 indica o uso do aplicativo pelo usuário. Ao utilizar o aplicativo o usuário executa ações (autenticação, cadastros, edição, visualização) clicando nos componentes do aplicativo (botões, textos, imagens). O aplicativo foi desenvolvido utilizando o framework Flutter. Os passos 2 e 3 mostram por onde ocorre a comunicação das requisições (request) e respostas (response) entre o aplicativo e API. As requisições e respostas trafegam pela internet utilizando os métodos GET, POST, PUT e DELETE do protocolo HTTP. Os dados são manipulados utilizando arquivos do tipo JSON (JavaScript Object Notation). O passo 4 mostra a comunicação da API desenvolvida com framework Laravel para manipulação dos dados (consultar, inserir ou deletar) armazenados no banco de dados MySQL.

4.0.4 API

A saber em relação aos resultados quanto a operação da API desenvolvida utilizando o framework Laravel, podemos citar que, este framework utiliza os princípios do padrão MVC e desta forma será explicado o fluxo de comunicação entre as camadas da API.

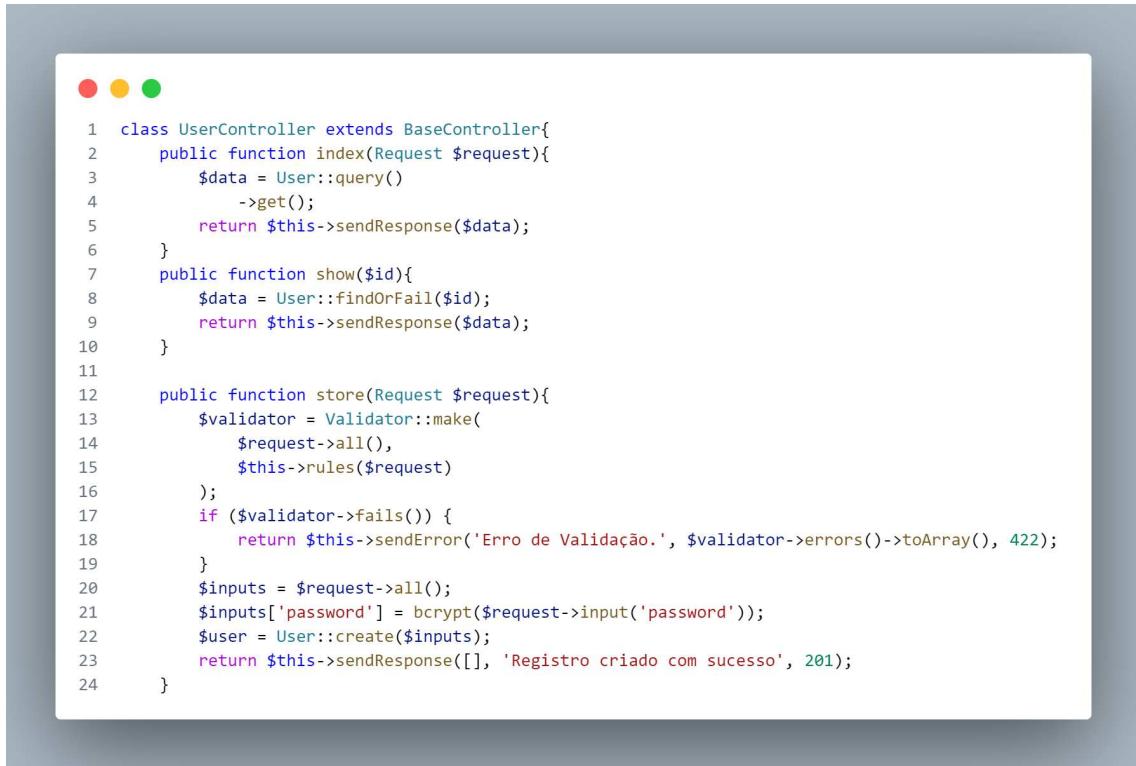


```
● ○ ●
1 Route::prefix('v1')->group(function () {
2     Route::group(['prefix' => 'auth'], function () {
3         Route::apiResource('users', App\Http\Controllers\API\UserController::class)->only(['index', 'show', 'store']);
4         Route::post('login', [App\Http\Controllers\API\SessionController::class, 'store']);
5         Route::middleware('auth:sanctum')->group(function () {
6             Route::delete('logout', [App\Http\Controllers\API\SessionController::class, 'destroy']);
7             Route::apiResource('users', App\Http\Controllers\API\UserController::class)->only(['update']);
8             Route::post('change-password', App\Http\Controllers\API\ChangePasswordController::class);
9             Route::apiResource('lots', App\Http\Controllers\API\LotController::class)->only(['store', 'update', 'destroy']);
10        });
11        Route::post('password/email', App\Http\Controllers\API\ForgotPasswordController::class);
12        Route::post('password/code/check', App\Http\Controllers\API\CodeCheckController::class);
13        Route::post('password/reset', App\Http\Controllers\API\ResetPasswordController::class);
14    });
15    Route::apiResource('lots', App\Http\Controllers\API\LotController::class)->only(['index', 'show']);
16    Route::apiResource('cities', App\Http\Controllers\API\CityController::class)->only(['index', 'show']);
17 });
});
```

Figura 10 – Definição das rotas da API.

Fonte: Autor, 2023.

Na [Figura 10](#) é apresentado como são definidas as rotas da API. Na linha 3, o título de exemplo, é definida a rota padrão para manipular os dados do Usuário através da classe UserContoller. As funções *index*, *show* e *store* indicam qual o tipo de ação: A) *index*: retornar uma lista de todos os usuários registrados no sistema; B) *show*: retornar detalhes de um usuário específico com base no seu ID; C) *store*: valida os dados fornecidos pelo cliente (aplicação), cria um novo objeto/modelo com esses dados e, em seguida, o armazena no banco de dados. Nessa direção observemos a próxima figura.



```
1 class UserController extends BaseController{
2     public function index(Request $request){
3         $data = User::query()
4             ->get();
5         return $this->sendResponse($data);
6     }
7     public function show($id){
8         $data = User::findOrFail($id);
9         return $this->sendResponse($data);
10    }
11
12    public function store(Request $request){
13        $validator = Validator::make(
14            $request->all(),
15            $this->rules($request)
16        );
17        if ($validator->fails()) {
18            return $this->sendError('Erro de Validação.', $validator->errors()->toArray(), 422);
19        }
20        $inputs = $request->all();
21        $inputs['password'] = bcrypt($request->input('password'));
22        $user = User::create($inputs);
23        return $this->sendResponse([], 'Registro criado com sucesso', 201);
24    }
}
```

Figura 11 – Código da classe controladora do Usuário.

Fonte: Autor, 2023.

O código da [Figura 11](#) mostra a partir da linha 2 que a função *index* consulta todos os usuários registrado no banco de dados e retorna os dados na variável *\$data*. A partir linha 7 indica o mesmo processo porém baseando no parâmetro id para retornar somente um único usuário. A partir da linha 14 a função *store* valida os dados fornecidos pelo cliente, cria um novo objeto/modelo com esses dados e, em seguida, o armazena no banco de dados.



The screenshot shows a code editor window with a dark theme. At the top left are three colored circular icons: red, yellow, and green. The main area contains the following PHP code:

```
1 class User extends Authenticatable{
2     use HasApiTokens, HasFactory, Notifiable, IsActive;
3     protected $fillable = [
4         'name',
5         'nif',
6         'phone',
7         'email',
8         'birthdate',
9         'password'
10    ];
11    protected $hidden = [
12        'password',
13        'remember_token',
14    ];
15    protected $casts = [
16        'email_verified_at' => 'datetime',
17    ];
18 }
```

Figura 12 – Código da classe modelo do Usuário.

Fonte: Autor, 2023.

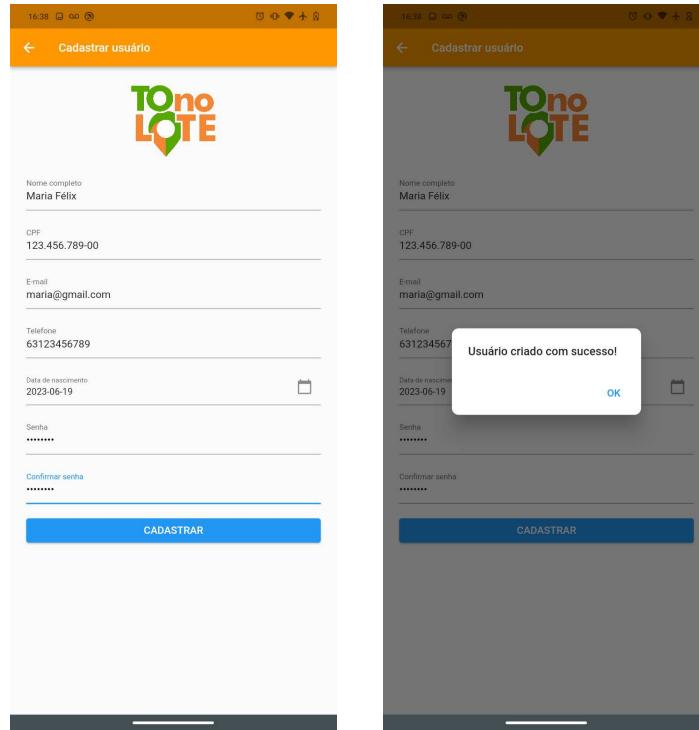
O código da [Figura 12](#) indica a estrutura de composição do modelo do Usuário. A partir da linha 3 são indicados seus atributos. A linha 11 indica, através da propriedade *hidden*, os atributos do modelo que devem ser ocultados ao serializar o objeto em formato JSON. Definindo esses atributos como ocultos, eles não serão visíveis nos resultados da API, oferecendo uma camada adicional de proteção aos dados do usuário.

4.0.5 Aplicativo To No Lote

As considerações quanto ao Aplicativo To No Lote, mostram que as suas principais funcionalidades, proporcionaram uma apresentação do produto minimamente viável. As telas serão descritas juntamente com as interações do usuário.

A [Figura 13](#) mostra o preenchimento das informações pessoais nos campos solicitados para realizar cadastro no aplicativo. Somente após cadastro o usuário tem acesso as funcionalidades de cadastro e visualização do lote com suas informações complementares.

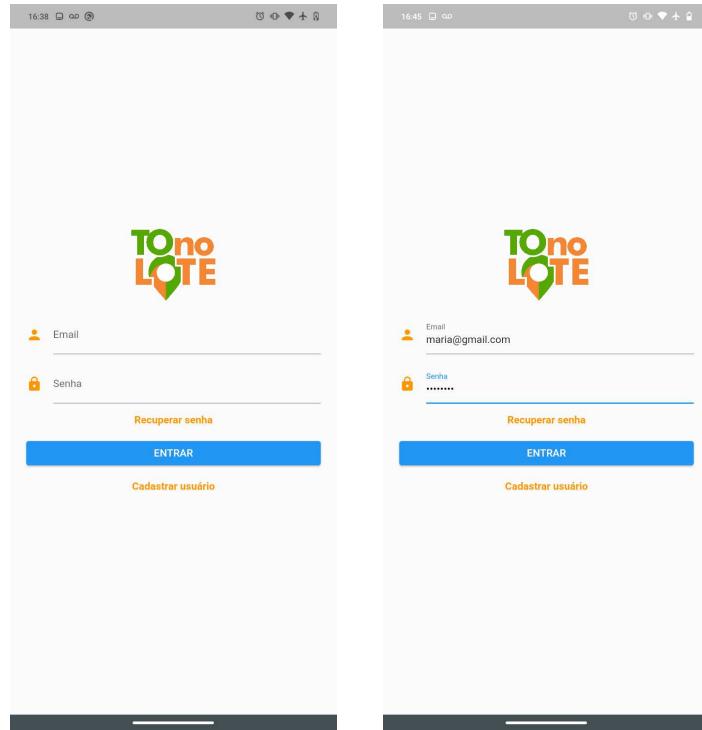
Figura 13 – Aplicativo To No Lote - Cadastro de Usuário



Fonte: Autor, 2023.

A Figura 14 mostra o processo de autenticação onde o usuário deve informar as credenciais e-mail e senha. Dentro do projeto do aplicativo o modelo da classe Usuário utiliza a mesma estrutura de atributos definido na API, porém agora há adição do atributo *token*. O usuário informa as credências e-mail e senha, a API valida e responde com os dados do usuário mais o token que permanecerá válido durante o acesso ao aplicativo.

Figura 14 – Aplicativo To no Lote - Autenticação



Fonte: Autor, 2023.

A Figura 15 mostrar a requisição feita pelo aplicativo a API com os dados de acesso a serem validados.

```

1  {
2      "email": "maria@gmail.com",
3      "password": "12345678"
4  }
5

```

Figura 15 – Dados para requisiçāoo de acesso ao aplicativo.

Fonte: Autor, 2023.

Conforme a API classifica os dados enviados na requisição de acesso ao aplicativo como válidos a mesma retorna o conjunto de dados pertencentes este usuário, como mostra a Figura 16.



```

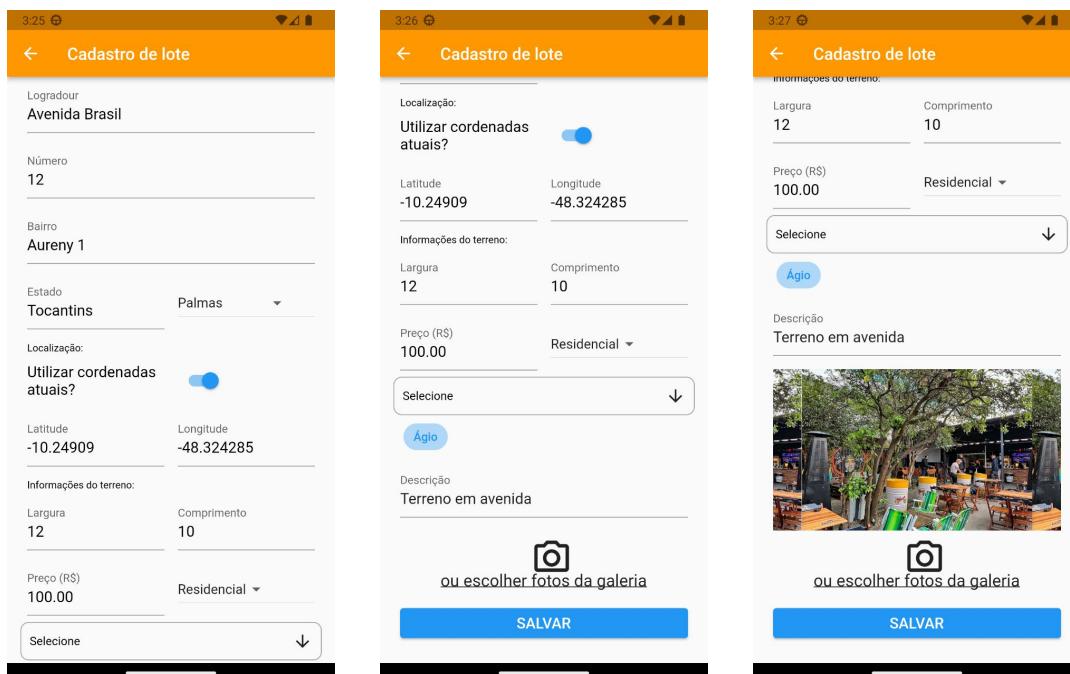
1  {
2      "message": "Login successfully.",
3      "data": {
4          "user": {
5              "id": 6,
6              "name": "Maria Felix",
7              "nif": "12345678900",
8              "phone": "63123456789",
9              "email": "maria@gmail.com",
10             "birthdate": "2023-06-19",
11             "email_verified_at": null,
12             "created_at": "2023-06-19T19:38:37.000000Z",
13             "updated_at": "2023-06-19T19:38:37.000000Z"
14         },
15         "token": "23|V17o7yH8Fjml8mjeCIDg0Jitb4jbYDl1Sn6Q4IHg"
16     }
17 }
```

Figura 16 – Resposta da API a requisição de acesso ao aplicativo.

Fonte: Autor, 2023.

A partir do usuário autenticado é possível realizar o cadastro do lote. Para este cadastro há a necessidade do preenchimento de uma série de dados, como mostra a Figura 17. As credenciais latitude e longitude serão capturadas de forma automática utilizando a localização do dispositivo ou inseridas pelo próprio usuário. Essas credenciais serão utilizadas no complemento das informações do lote na sua apresentação dentro do mapa.

Figura 17 – Aplicativo To No Lote - Cadastro do Lote

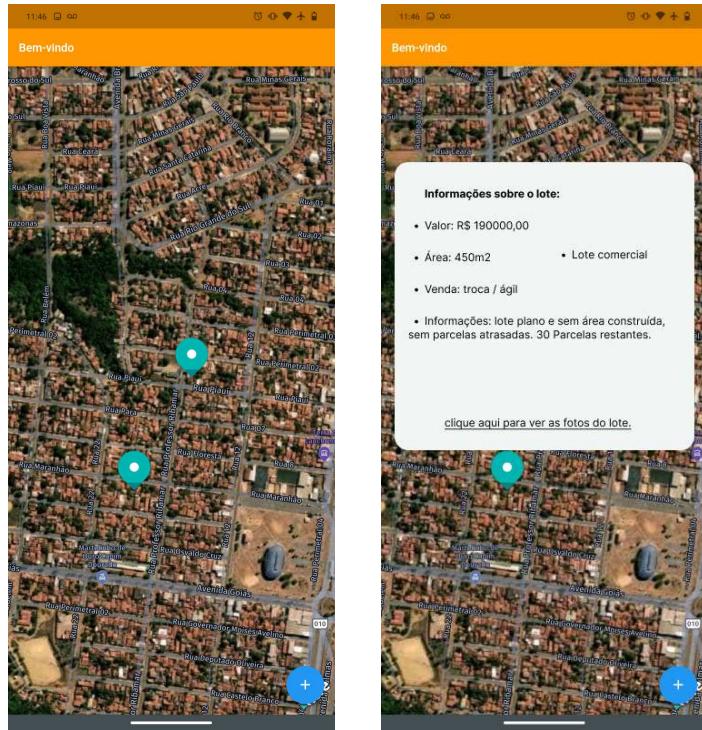


Fonte: Autor, 2023.

A Figura 18 mostra como os lotes à venda serão apresentados no mapa do aplicativo

conforme identificação da cidade da localização do dispositivo. Dessa forma a partir da navegação no mapa mais lotes cadastrados serão apresentados.

Figura 18 – Aplicativo To No Lote - Lotes à venda



Fonte: Autor, 2023.

5 Conclusão

Como notas conclusivas deste estudos, destacamos mesmo que construindo uma documentação objetiva e sem muitas funcionalidades complexas através da aplicação da metodologia ágil houve problemas no processo de desenvolvimento. O principal problema foi chegar a um domínio total do framework Flutter para finalização de todas as funcionalidades do aplicativo. Ainda assim, o projeto é concluído com uma documentação objetiva, uma API que atende às necessidades básicas e um aplicativo que se aproxima do objetivo final, com potencial para melhorias futuras.

5.1 Trabalhos Futuros

Fazemos as seguintes sugestões para próximos trabalhos/estudos/pesquisas neste campo de investigação, tais como:

- Criar um sistema de *backoffice* para fornecer funcionalidade de inserir uma lista de lotes para vendas. Esta funcionalidade é pensado na atividade dos corretores que possuem uma quantidade considerável de lotes sob sua responsabilidade para venda.
- Dar andamento na conclusão das funcionalidade: A)exibir média de preço por cidade; B)exibir informações específicas de distância do lote para hospital, escola, universidade, shopping, supermercado; criar filtro personalizável para buscas.

Referências

- ARCOS-MEDINA, G. et al. Comparative study of performance and productivity of mvc and mvvm design patterns. *KnE Engineering*, p. 241–252, 2018.
- CHARLAND A. E LEROUX, B. Mobile application development: web vs. native. *Communications of the ACM*, v. 54, n. 5, 2011.
- CONSTANTIN N. O., R.-G. M. e. C. N. M. Comparison of hybrid cross-platform mobile applications with native cross-platform applications. *Journal Of Computer Science And Control Systems*, v. 9, n. 2, 2016.
- CORAZZA, P. V. Um aplicativo multiplataforma desenvolvido com flutter e nosql para o cálculo da probabilidade de apendicite. 2018.
- EL-KASSAS, W. S. et al. Taxonomy of cross-platform mobile applications development approaches. *Ain Shams Engineering Journal*, v. 8, n. 2, p. 163–190, 2017. ISSN 2090-4479. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2090447915001276>>.
- EL-KASSAS, W. S. e. a. *Taxonomy of Cross-Platform Mobile Applications Development Approaches*. 2015. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S2090447915001276>>.
- GABARDO, A. C. *Laravel para ninjas*. [S.l.]: Novatec Editora, 2017.
- HERE, D. *Here Rest APIs*. 2019. Disponível em: <<https://developer.here.com/develop/rest-apis>>. Acesso em: Abril. 2023.
- LECHETA, R. R. *Google Android-5ª Edição: Aprenda a criar aplicações para dispositivos móveis com o Android SDK*. [S.l.]: Novatec Editora, 2015.
- LIMA, F. F. d. Avaliação de frameworks para o desenvolvimento de aplicações híbridas. Universidade Federal do Pampa, 2019.
- MITTELSTADT, R. S. Bluepath: Sistema de localização indoor. 2018.
- MOURA, M. A.; FARIA, A. F. Produção, busca e compartilhamento de informação no contexto do comércio móvel. *Universidade Federal de Minas Gerais*, 2019.
- PONTES, S. W. P. Desenvolvimento de aplicativos móveis híbridos: um estudo de caso sobre o rmiíase. Universidade Federal do Maranhão, 2017.
- REDDA, Y. A. *Cross platform Mobile Applications Development: Mobile Apps Mobility*. Dissertação (Mestrado) — Institutt for datateknikk og informasjonsvitenskap, 2012.
- SILVAL, A. S. da S. et al. Asp net mvc e sites comerciais. *REFAQI-REVISTA DE GESTÃO EDUCAÇÃO EE TECNOLOGIA*, v. 6, n. 2, 2019.
- SOARES, M. dos S. Metodologias ágeis extreme programming e scrum para o desenvolvimento de software. *Revista Eletrônica de Sistemas de Informação*, v. 3, n. 1, 2004.

- STOPA, G. R.; RACHID, C. L. Scrum: Metodologia ágil como ferramenta de gerenciamento de projetos. *CES Revista*, v. 33, n. 1, p. 302–323, 2019.
- THAKUR, R. N.; PANDEY, U. A study focused on web application development using mvc design pattern. *Int. Res. J. Eng. Technol.*, v. 6, n. 08, 2019.
- VELÁSQUEZ, S. M. et al. Una revisión comparativa de la literatura acerca de metodologías tradicionales y modernas de desarrollo de software. *Revista Cintex*, v. 24, n. 2, p. 13–23, 2019.
- VERMA, S. *APIs versus web services*. 2019. Disponível em: <<https://blogs.mulesoft.com/dev-guides/apis-versus-web-services/>>. Acesso em: Abril. 2023.
- WU, W. React native vs flutter, cross-platforms mobile application frameworks. Metropolia Ammattikorkeakoulu, 2018.
- ZAMMETTI, F. *Flutter na prática: Melhore seu desenvolvimento mobile com o SDK open source mais recente do Google*. NOVATEC, 2020. ISBN 9788575228227. Disponível em: <<https://books.google.com.br/books?id=UKTJDwAAQBAJ>>.

6 Apêndice A Prototipação

Figura 19 – Prototipação 001

The figure shows two wireframe prototypes for a real estate platform.

Login Screen:

- Header: TO no LOTE
- Form fields: E-mail, Senha (Senha), Entrar button, and a link to recover password.
- Link at the bottom: Clique aqui para cadastrar-se!

Cadastro Screen:

- Header: Preencha os dados abaixo
- Form fields: Nome completo, CPF, E-mail, Telefone, Data de nascimento, Senha, Confirmar senha, and a Cadastrar button.
- Link at the bottom: Clique aqui para cadastrar-se!

Figura 20 – Prototipação 002

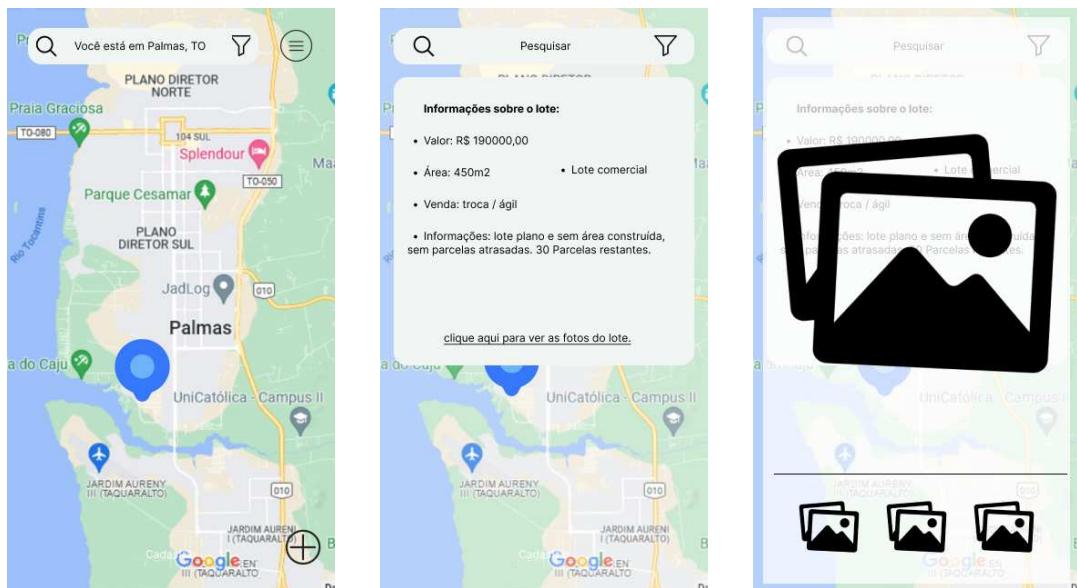


Figura 21 – Prototipação 003

Preencha com os dados do lote

Clique aqui para indicar localização do lote no mapa

CEP

Logradouro Número

Bairro

Estado Cidade

Latitude Longitude

Largura Comprimento

Área (m²) Tipo de lote

Valor (R\$) Tipo de venda

Indique no mapa a localização exata do lote, dessa forma o aplicativo poderá apresentar informações precisas que valorizam sua venda como: distância para supermercados, escolas, universidades, ponto de ônibus, delegacias, praças e hospitais.

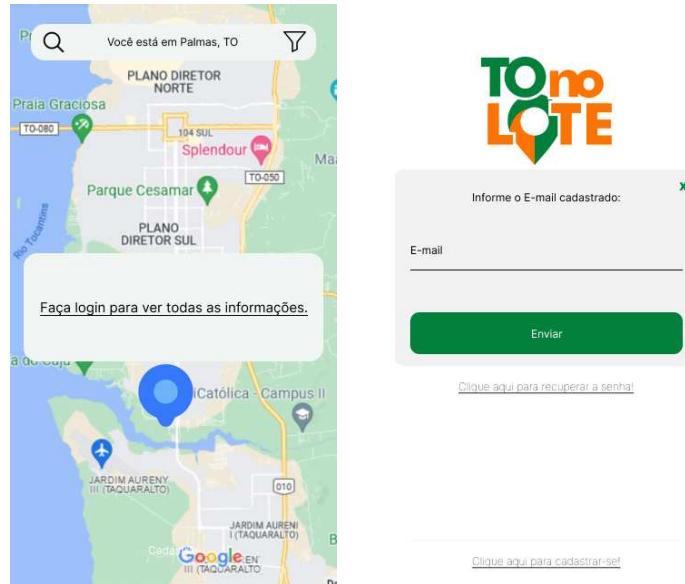
Cadastrar

Figura 22 – Prototipação 004

Lotes cadastrados

<ul style="list-style-type: none"> Lote 001 Valor: R\$ 190000,00 Venda: troca / ágil Cidade: Palmas 		
<ul style="list-style-type: none"> Lote 002 Valor: R\$ 200000,00 Venda: troca / ágil Cidade: Porto Nacional 		
<ul style="list-style-type: none"> Lote 003 Valor: R\$ 150000,00 Venda: troca / ágil Cidade: Paraíso do Tocantins 		

Figura 23 – Prototipação 005



7 Apêndice B Requisitos Funcionais

Tabela 9 – RF001 | CRUD de Usuário

RF001	CRUD de Usuário
Definição	Gerenciar usuário (cadastrar / editar / deletar linguagens)
Sumário	Deve permitir gerenciar os dados relacionados ao usuário
Pré-condições	Ter acesso à API para manipular dados no banco
Descrição	-
Autores	Usuário
Exceção	-

Tabela 10 – RF002 | CRUD de Lote

RF002	CRUD de Lote
Definição	Gerenciar lotes (cadastrar / editar / deletar lições)
Sumário	Deve permitir gerenciar os dados relacionados aos lotes cadastrados pelo usuário
Pré-condições	Ter acesso à API para manipular dados no banco
Autores	Usuário
Descrição	-
Exceção	-

Tabela 11 – RF007 | Apresentar lotes no mapa

RF007	Apresentar lotes no mapa
Definição	Mostrar o mapa com filtro inicial por cidade apenas
Sumário	Identificar automaticamente a cidade do usuário e apresentar os lotes desta cidade.
Pré-condições	Ter acesso à API para manipular dados no banco
Autores	API
Descrição	-
Exceção	-

8 Apêndice C Escopo do Projeto (Backlog)

Tabela 12 – Identificação do Projeto

Projeto Nº.	Descrição	Data
00001	Sistema de informação móvel multiplataforma com suporte a geolocalização para venda de lotes	

Tabela 13 – Objetivos do Projeto

Descrição
Sistema de informação móvel (aplicativo) multiplataforma para venda de lotes. Os lotes devem ser cadastrados por usuário autenticado. Os lotes cadastrados no aplicativo devem ser apresentados no mapa. Junto com as informações cadastradas o aplicativo poderá fornecer dados para agregar o anúncio, como: distância para estabelecimentos (hospital, escola, supermercado, shopping, ponto de ônibus) e valor médio da região.

Tabela 14 – Entregas do Projeto

Entrega	Descrição
API	API Rest para servir ao aplicativo
Aplicativo To no Lote	Aplicativo conforme mockup portotipação das telas (mockup)
Documentação	Trabalho de Conclusão de Curso

Tabela 15 – Lista de Tarefas API

Tarefa Nº.	Descrição
1. Criar endpoint para usuário	Criar CRUD (POST, GET, PUT e DEL)
2. Criar endpoint para lote	Criar CRUD (POST, GET, PUT e DEL)
3. Criar endpoint para autenticação	Criar login e logout
4. Criar endpoint para recuperação de senha	Esqueci minha senha, checagem do código de recuperação via e-mail, redefinição da senha

Tabela 16 – Lista de Tarefas Aplicativo

Tarefa Nº.	Descrição
1. Criar cadastro de usuário	Tela cadastro de usuário
2. Procedimento de autenticação	Tela de autenticação
3. Criar cadastro de lotes	Tela cadastro de usuário
4. Exibir mapa com lotes	Tela do mapa deve apresentar os lotes de acordo a cidade de localização do dispositivo ou pesquisa por cidade

Atributos com caractere "*" (asterisco) indicam obrigatoriedade.

Tabela 17 – Tabela Usuário

Atributo	Tipo	Tamanho	Opções	Observação
IDUsuario	Int	8		
Nome*	C	40		
CPF*	C	11		
Telefone*	C	11		
Email*	C	50		
DataNascimento*	Date	8		
Senha*	C	3		
Data Criação	DateTime			Gravação automática
Data Alteração	DateTime			Gravação automática

Tabela 18 – Tabela Lote

Atributo	Tipo	Tamanho	Opções	Observação
IDLote	Int	8		
IDUsuario*	C	8		Id do Usuário autenticado no aplicativo
CEP	C	8		
Logradouro	C	30		
Número	C	6		
Bairro	C	40		
IDCidade*	Int	8		
Latitude*	Decimal			
Longitude*	Decimal			
Largura	Decimal			
Comprimento	Decimal			
Área	Decimal			Calculador automaticamente, fórmula: Área = Largura * Comprimento
Preço*	Decimal			
Tipo*	C	1	«enum» Residencial ou Comercial	5
Descrição	C	300		Caixa de texto
Data Criação	DateTime			Gravação automática
Data Alteração	DateTime			Gravação automática

Tabela 19 – Tabela Foto

Atributo	Tipo	Tamanho	Opções	Observação
IDFoto	Int	8		
IDLote	Int	8		
Foto				
Data Alteração	DateTime			Gravação automática

Tabela 20 – Tabela Tipo de Venda

Atributo	Tipo	Tamanho	Opções	Observação
IDTipoVenda	Int	8		
Código*	Int	8		
Nome*	C	20		
Data Criação	DateTime			Gravação automática
Data Alteração	DateTime			Gravação automática

Tabela 21 – Tabela Lote_TipoDeVenda

Atributo	Tipo	Tamanho	Opções	Observação
ID	Int	8		
IDLote*	Int	8		
IDTipoVenda*	Int	8		