



**PUC
GOIÁS**

Pontifícia Universidade Católica de Goiás
Análise e Desenvolvimento de Sistemas

DOCUMENTAÇÃO COMPLETA PROJETO FINAL

**MOISES LOPES DE MATOS,
DIEGO BARBOSA BRANDÃO ROCHA,
LUIZ FERNANDO FERREIRA DA SILVA**

GOIANIA

2025

PROJETO INTEGRADOR – 5º PERÍODO

Curso: Análise e Desenvolvimento de Sistemas – PUC Goiás

- **Diego Barbosa Brandão Rocha**
- **Luiz Fernando Ferreira da Silva**
- **Moisés Lopes de Matos**

1. APRESENTAÇÃO E SUMÁRIO

Este documento tem como objetivo apresentar a quarta entrega do Projeto Integrador do 5º período do curso de Análise e Desenvolvimento de Sistemas da PUC Goiás. O projeto propõe o desenvolvimento de um aplicativo mobile multiplataforma, utilizando Flutter, voltado à compra e venda de veículos usados. Essa entrega detalha a estrutura de classes, pacotes, padrões de projeto e interfaces aplicadas no frontend da aplicação.

SUMÁRIO

1.	Apresentação	pág. 2
2.	Informações Gerais	pág. 3
2.1	Escopo do Projeto	pág. 3
2.2	Requisitos do Projeto	pág. 4
2.2.1	Diagrama de Casos de Uso	pág. 5
2.3	Arquitetura de Software	pág. 5
2.4	Criação do Banco de Dados	pág. 9
2.4.1	Relacionamentos entre as Entidades	pág. 10
2.4.2	Chaves Estrangeiras (FKs)	pág. 11
2.5	Estrutura Analítica do Projeto (EAP)	pág. 13
2.6	Cronograma de Entregas	pág. 14
2.7	Descrição do Software Desenvolvido	pág. 15
2.7.1	Fluxo de Uso com Telas	pág. 16
2.8	Especificação de Objetivos e Requisitos	pág. 23
2.9	Design de Software	pág. 25
2.10	Ciclo de Vida do Software	pág. 27
2.11	Project Model Canvas	pág. 28
3.	Critérios de Avaliação	pág. 28
4.	Atribuição das Notas por Disciplina	pág. 28
5.	Considerações Finais	pág. 29
5.1	Limitações e Melhorias Futuras	pág. 29
6.	Link do GitHub	pág. 29

2. INFORMAÇÕES GERAIS

Tema: Aplicativo Mobile para Compra e Venda de Veículos Usados

Tecnologia utilizada: Flutter (Frontend), Spring Boot (Backend RESTful)

2.1. Escopo do Projeto

O projeto propõe o desenvolvimento de um aplicativo mobile multiplataforma, utilizando o framework Flutter, com o objetivo de intermediar negociações entre compradores e vendedores de veículos usados. A proposta visa criar uma solução acessível, intuitiva e funcional que resolva problemas comuns no mercado de veículos seminovos, como a fragmentação de informações, a dificuldade de comunicação entre as partes e a ineficiência na busca por veículos com critérios específicos.

O escopo do projeto contempla as seguintes funcionalidades principais:

- **Catálogo de Veículos:** Visualização de uma listagem de veículos disponíveis com informações completas, como marca, modelo, ano, preço, quilometragem, combustível e câmbio, além de imagens ilustrativas.
- **Busca Avançada:** Ferramenta de filtragem que permite aos usuários encontrar veículos com base em múltiplos critérios, otimizando a experiência de navegação.
- **Sistema de Anúncios:** Interface amigável para criação, edição e exclusão de anúncios por parte dos vendedores. Cada anúncio pode conter descrição detalhada, fotos e vídeos, dados técnicos do veículo e valor de venda.
- **Integração com a Câmera e Galeria:** Recurso que permite aos vendedores tirar fotos diretamente do aplicativo ou selecionar imagens salvas no dispositivo para ilustrar os anúncios.
- **Chat Integrado:** Módulo de comunicação em tempo real entre compradores e vendedores, possibilitando o envio e recebimento de mensagens diretamente dentro do app, com notificações e histórico de conversas.
- **Geolocalização via Mapa:** Os usuários poderão visualizar a localização aproximada dos veículos anunciados por meio de integração com o Google Maps, facilitando a busca por opções próximas.

- **Cadastro e Login de Usuários:** Sistema de autenticação que diferencia perfis de comprador, vendedor particular e concessionária, com gerenciamento de dados pessoais e controle de acesso. *Persistência de login com SharedPreferences*

O escopo abrange apenas as funcionalidades voltadas para a experiência no aplicativo mobile, sendo o backend responsável pelo armazenamento de dados e lógica de negócios por meio de uma API RESTful desenvolvida em Java com Spring Boot. Funcionalidades como pagamentos, inspeções físicas, transferência de documentos ou qualquer tipo de mediação financeira estão fora do escopo desta aplicação.

2.2. Requisitos do Projeto

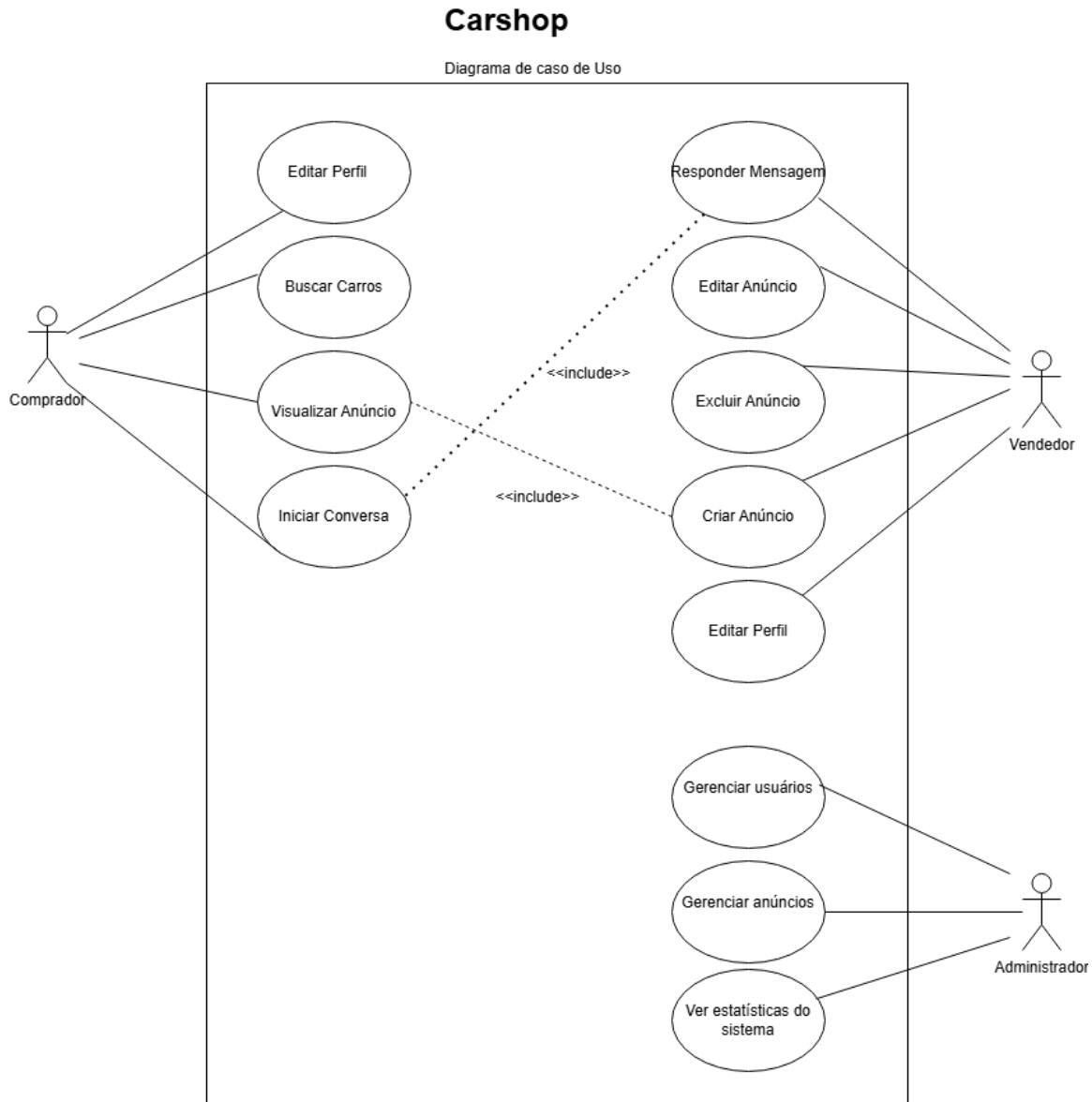
Requisitos Funcionais:

- - Cadastro/login de usuários (comprador, vendedor, concessionária)
 - Cadastro e gerenciamento de anúncios
 - Busca com filtros
 - Upload de fotos (image-video picker)
- - Localização GPS no
 - Chat integrado
- - Perfis e Customização.
 - Notificações

Requisitos Não Funcionais:

- - Interface responsiva e intuitiva
 - Conformidade com LGPD
 - Tempo de resposta de até 3 segundos
 - Suporte para Android 7+ e iOS 12+
-

2.2.1 Diagrama de Casos de Uso



2.3. Arquitetura de Software

Descrição da Arquitetura

A arquitetura adotada para o projeto é **em camadas com base no modelo cliente-servidor**, estruturada para possibilitar escalabilidade e reutilização dos serviços desenvolvidos.

O sistema é composto por três grandes componentes:

1. **Frontend Mobile (Flutter):**

- Interface voltada ao usuário final (compradores e vendedores).
- Desenvolvido com Flutter, segue arquitetura modular, separando:
 - models: entidades do domínio como Carro
 - screens: telas de interface do usuário
 - services: comunicação com a API RESTful
 - dwidgets: componentes visuais reutilizáveis

2. Backend RESTful (Spring Boot):

- Responsável por toda a lógica de negócios, regras, e persistência dos dados.
- Desenvolvido em Java com Spring Boot.
- Exposição de endpoints REST que permitem operações CRUD sobre entidades como Carro, Usuário, Anúncio, etc.
- Comunicação via HTTP/JSON com autenticação básica e segurança de dados.
- Camadas: Controller → Service → Repository → Database.

3. Interface Gráfica Administrativa (JavaFX):

- Aplicação desktop voltada para administradores ou gerentes da plataforma.
- Realiza consumo da mesma API RESTful que o app mobile.
- Permite visualizar, cadastrar, editar e remover dados do sistema.
- Desenvolvida em Java com JavaFX, utilizando bibliotecas como HttpClient, Gson ou Jackson para comunicação e manipulação de dados JSON.

Padrão Arquitetural

O padrão adotado é **em camadas e multicliente (multiplataforma)**.

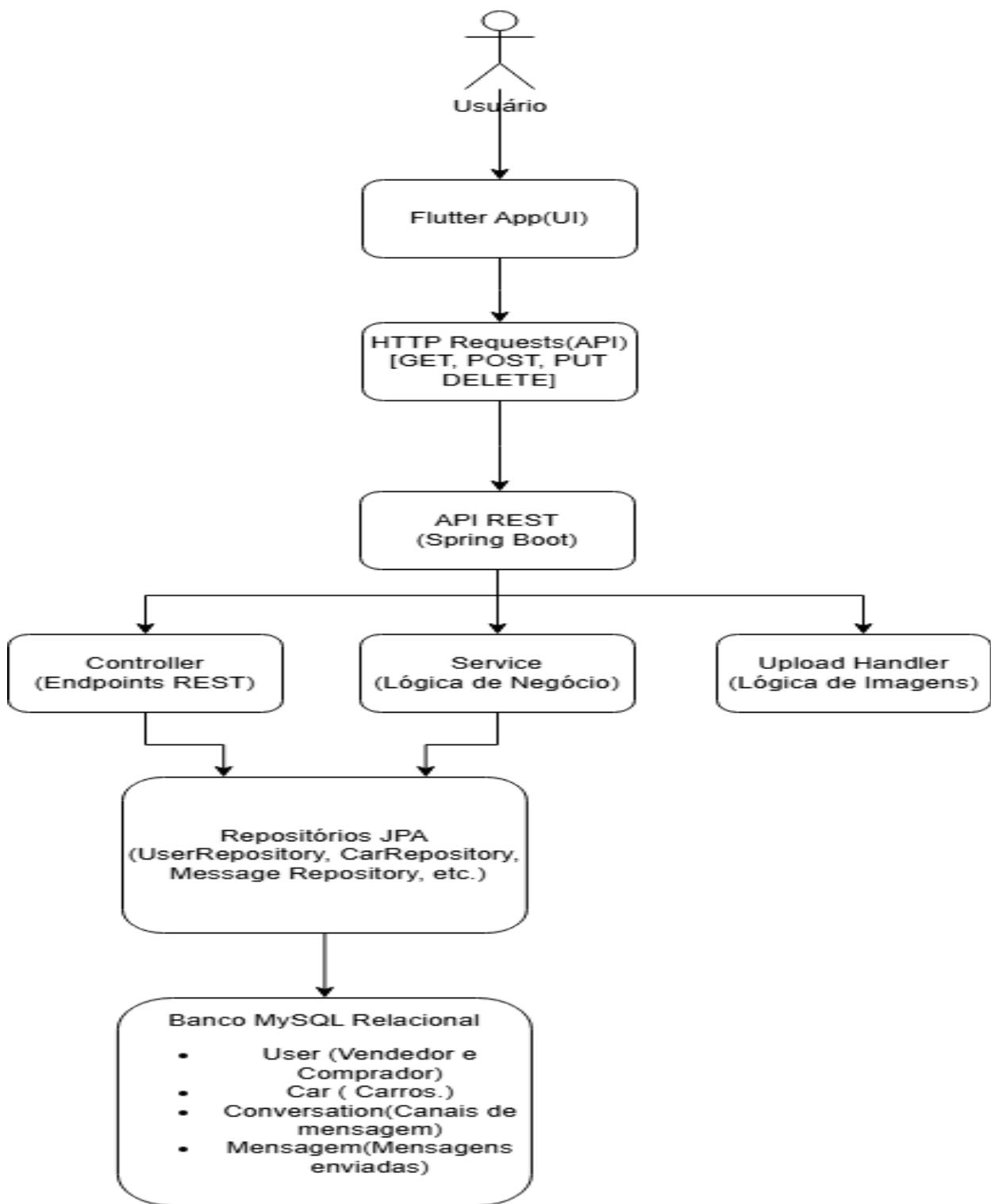
Cada camada possui responsabilidades bem definidas, permitindo manutenção e testes isolados. A API REST atua como intermediária entre as interfaces (Flutter e JavaFX) e o banco de dados.

Diagrama de Módulos do Flutter

```
/lib
|
|--- models/    # Classes como Car, User
|--- screens/   # Telas da interface (home, chat, mapa, etc.)
|--- services/  # Classe CarService para API
|--- widgets/   # CarCard e outros widgets
└--- main.dart  # Entrada do aplicativo
```

Diagrama de Arquitetura de Software

□



2.4. Criação do Banco de Dados

TABELAS DAS ENTIDADES

◊ Tabela user

Coluna	Tipo	Descrição
id	BIGINT	Chave primária
name	VARCHAR(255)	Nome do usuário
email	VARCHAR(255)	Email do usuário
password	VARCHAR(255)	Senha (criptografada)
birth_date	VARCHAR(255)	Data de nascimento
city	VARCHAR(255)	Cidade
latitude	DOUBLE	Latitude da localização
longitude	DOUBLE	Longitude da localização
profile_image_url	VARCHAR(255)	Caminho da imagem de perfil

◊ Tabela car

Coluna	Tipo	Descrição
id	BIGINT	Chave primária
title	VARCHAR(255)	Título do anúncio
price	VARCHAR(255)	Preço do carro
year	VARCHAR(255)	Ano do carro
mileage	VARCHAR(255)	Quilometragem
variant	VARCHAR(255)	Versão do carro
image_url	VARCHAR(255)	Caminho da imagem
meeting_lat	DOUBLE	Latitude do ponto de encontro
meeting_lng	DOUBLE	Longitude do ponto de encontro
created_at	DATETIME(6)	Data/hora da criação do anúncio
seller_id	BIGINT (FK)	Chave estrangeira para user(id)

◊ Tabela conversation

Coluna	Tipo	Descrição
id	BIGINT	Chave primária
item_id	BIGINT (FK)	FK para car(id) (o carro anunciado)
buyer_id	BIGINT (FK)	FK para user(id) (usuário que está comprando)
seller_id	BIGINT (FK)	FK para user(id) (usuário que está vendendo)

last_message_at	DATETIME(6)	Data/hora da última mensagem
-----------------	-------------	------------------------------

◊ *Tabela message*

Coluna	Tipo	Descrição
id	BIGINT	Chave primária
conversation_id	BIGINT (FK)	FK para conversation(id)
sender_id	BIGINT (FK)	FK para user(id) (quem enviou a mensagem)
content_text	LONGTEXT	Texto da mensagem
content_blob	LONGBLOB	Conteúdo de vídeo (opcional)
type	VARCHAR(255)	Tipo de conteúdo (texto ou vídeo)
created_at	DATETIME(6)	Data/hora da mensagem

◊ **2. RELACIONAMENTOS ENTRE AS ENTIDADES**

- **User e Car:**
Um usuário pode vender vários carros, mas cada carro pertence a **um único vendedor**.
(Relacionamento 1:N → `user.id` → `car.seller_id`)
- **User e Conversation:**
Um usuário pode participar de várias conversas, tanto como comprador quanto como vendedor.
(Relacionamento 1:N → `user.id` → `conversation.buyer_id` e `conversation.seller_id`)
- **Car e Conversation:**
Um carro pode ter várias conversas associadas (interessados diferentes), mas cada conversa está ligada a **um único carro**.
(Relacionamento 1:N → `car.id` → `conversation.item_id`)
- **Conversation e Message:**
Cada conversa pode conter várias mensagens, mas cada mensagem pertence a **uma única conversa**.
(Relacionamento 1:N → `conversation.id` → `message.conversation_id`)
- **User e Message:**
Um usuário pode enviar várias mensagens, mas cada mensagem tem **um único remetente**.
(Relacionamento 1:N → `user.id` → `message.sender_id`)

CHAVES (FKs)

◊ Tabela car

- seller_id → REFERENCES user(id)

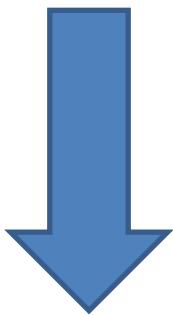
◊ Tabela conversation

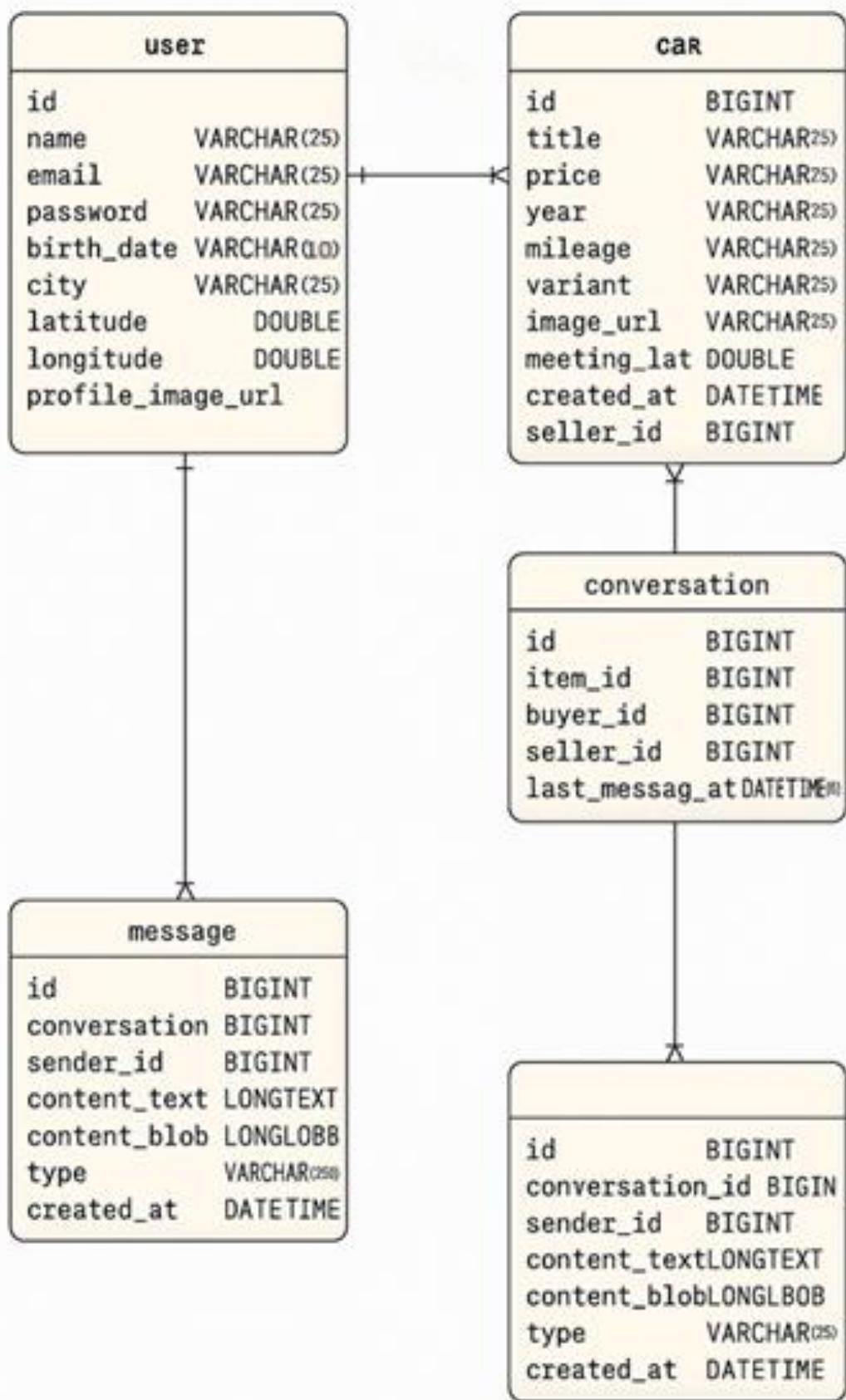
- item_id → REFERENCES car(id)
- buyer_id → REFERENCES user(id)
- seller_id → REFERENCES user(id)

◊ Tabela message

- conversation_id → REFERENCES conversation(id)
- sender_id → REFERENCES user(id)

DIAGRAMA (VISUAL)





2.5. Estrutura Analítica do Projeto (EAP)

A Estrutura Analítica do Projeto (EAP) define as fases e etapas necessárias para o desenvolvimento do aplicativo mobile proposto. A EAP está organizada em **três níveis de hierarquia**, permitindo o acompanhamento detalhado das atividades, facilitando o planejamento, controle e execução do projeto.

EAP – Níveis Hierárquicos:

Nível 1 – Projeto Integrador

→ Representa o escopo total do projeto, desde a concepção até a entrega final.

Nível 2 – Fases do Projeto

- 1. Planejamento**
- 2. Desenvolvimento**
- 3. Testes**
- 4. Documentação**
- 5. Apresentação Final**

Nível 3 – Atividades Detalhadas

1. Planejamento

- 1.1 Formação da equipe
- 1.2 Escolha do tema e definição de escopo
- 1.3 Escolha das tecnologias (Flutter, Spring Boot, JavaFX)
- 1.4 Definição do cronograma de entregas
- 1.5 Estudo e modelagem inicial dos requisitos

2. Desenvolvimento

- 2.1 Criação de wireframes e protótipos
- 2.2 Implementação das telas principais no Flutter
- 2.3 Criação dos serviços (camada de comunicação com a API)
- 2.4 Desenvolvimento do backend RESTful em Spring Boot
- 2.5 Implementação do CRUD de entidades (ex: Carro, Usuário)
- 2.6 Desenvolvimento da interface administrativa em JavaFX
- 2.7 Integração entre o Flutter e a API
- 2.8 Integração entre JavaFX e a API

3. Testes

- 3.1 Testes unitários no Flutter
- 3.2 Testes de integração no backend
- 3.3 Testes de comunicação entre frontend/backend
- 3.4 Testes com usuários (usabilidade e experiência)

4. Documentação

- 4.1 Redação da documentação funcional
- 4.2 Elaboração da arquitetura e diagramas
- 4.3 Documentação da API (Documentação manual via código e testes com Insomnia/Postman)
- 4.4 Preparação do relatório final para submissão

5. Apresentação Final

- 5.1 Preparação do pitch/apresentação
- 5.2 Demonstração do sistema funcionando
- 5.3 Entrega do relatório e código-fonte

2.6. Cronograma de Entregas

- - Início: Março/2025
- MVP: Abril/2025
- Entrega Final: Junho/2025

2.7. Descrição do Software Desenvolvido

Visão Geral do Software

O software desenvolvido é um aplicativo mobile multiplataforma, construído com Flutter, que tem como objetivo facilitar a compra e venda de veículos usados por meio de uma interface amigável, acessível e funcional. O app permite que vendedores anunciem veículos com fotos, vídeos e informações técnicas detalhadas, enquanto compradores podem realizar buscas com filtros avançados, visualizar detalhes e entrar em contato com os anunciantes via chat integrado.

Os dados apresentados no aplicativo são fornecidos por uma **API RESTful**, desenvolvida separadamente com Java e Spring Boot, que centraliza toda a lógica de negócio, persistência e controle de segurança. Essa API garante que as operações de cadastro, consulta, edição e exclusão de dados sejam feitas de forma segura, confiável e escalável. Além disso, uma interface desktop administrativa foi desenvolvida com JavaFX, permitindo o gerenciamento do sistema por administradores através do consumo da mesma API.

Funcionalidades Implementadas

- Cadastro e login de usuários com autenticação segura.
- Criação, edição e exclusão de anúncios de veículos.
- Upload de fotos e vídeos diretamente do dispositivo.
- Sistema de busca com filtros.
- Tela de exibição de detalhes do veículo, com todas as informações relevantes.
- Chat integrado para contato direto entre comprador e vendedor.
- Notificações de novas mensagens e atividades no app.
- Visualização no mapa da localização aproximada dos veículos anunciados.
- Perfil do usuário com gerenciamento de dados e histórico de anúncios.
- Interface JavaFX para administradores (consumo REST).

Desafios Enfrentados

Durante o desenvolvimento, os principais desafios técnicos envolveram:

- Integração segura e eficiente entre o aplicativo Flutter e a API RESTful.

- Implementação do sistema de chat e notificações em tempo real.
- Gerenciamento de múltiplas fotos e vídeos em dispositivos móveis.
- Adaptação da interface para diferentes tamanhos de tela e dispositivos.
- Garantia da conformidade com as práticas de segurança e proteção de dados (LGPD).
- Estruturação e consumo de endpoints REST tanto por Flutter quanto por JavaFX.

Tecnologias Utilizadas

- **Frontend Mobile:** Flutter (Dart)
- **Backend RESTful:** Java + Spring Boot
- **Interface Administrativa:** JavaFX
- **Banco de Dados:** PostgreSQL (ou outro banco relacional, conforme definição da equipe)
- **Comunicação entre camadas:** Protocolo HTTP com payloads JSON
- **Hospedagem (planejada):** AWS ou servidor local para testes
- **Controle de Versão:** Git + GitHub
- **Gerenciamento de Projeto:** Trello
- **Outras ferramentas:** Insomnia, Flutter Test

2.7.1 Fluxo de Uso com Telas (com prints)

[Print da tela de login]

8:30 ☰ 🛡️ 📱 📈

DEBUG

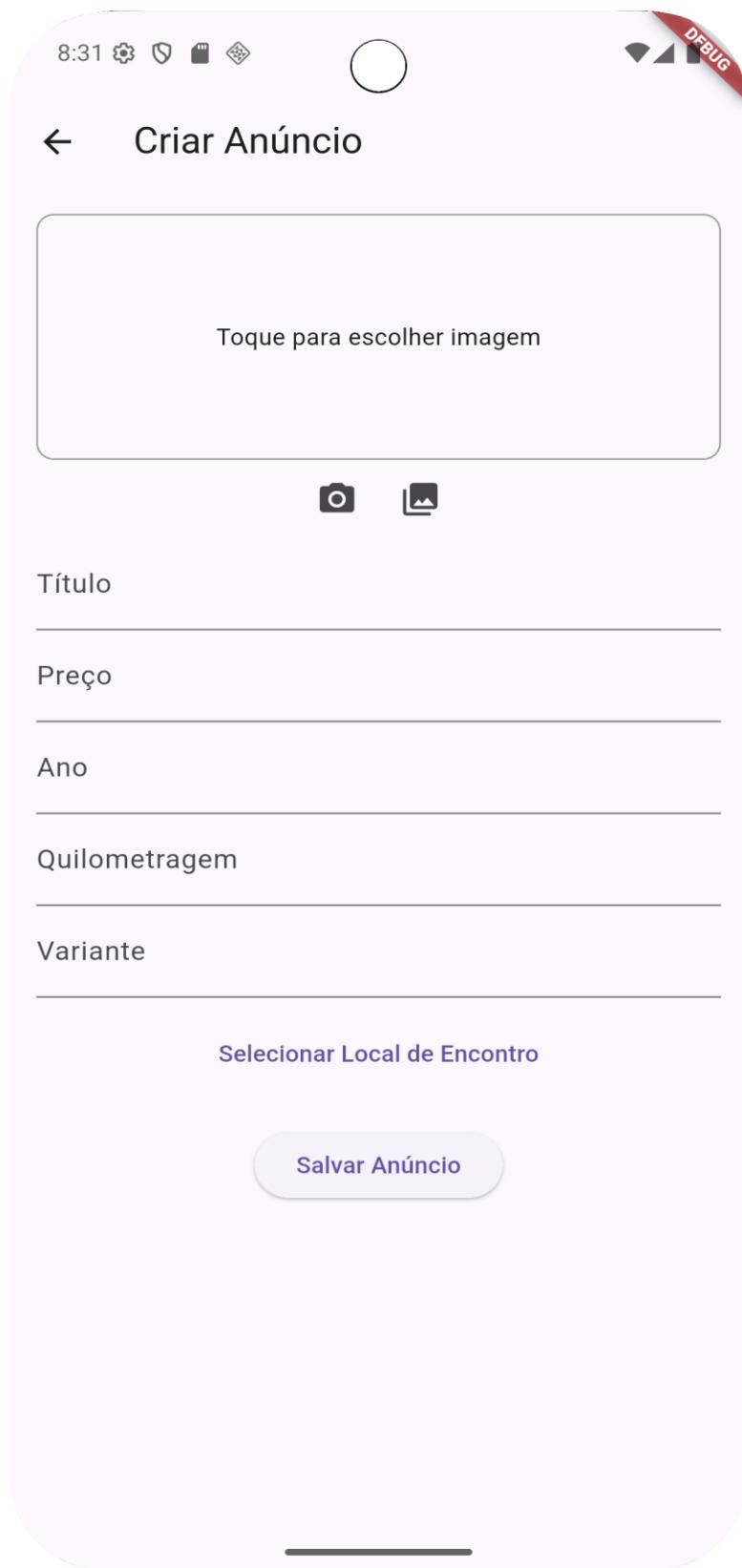
Login

E-mail

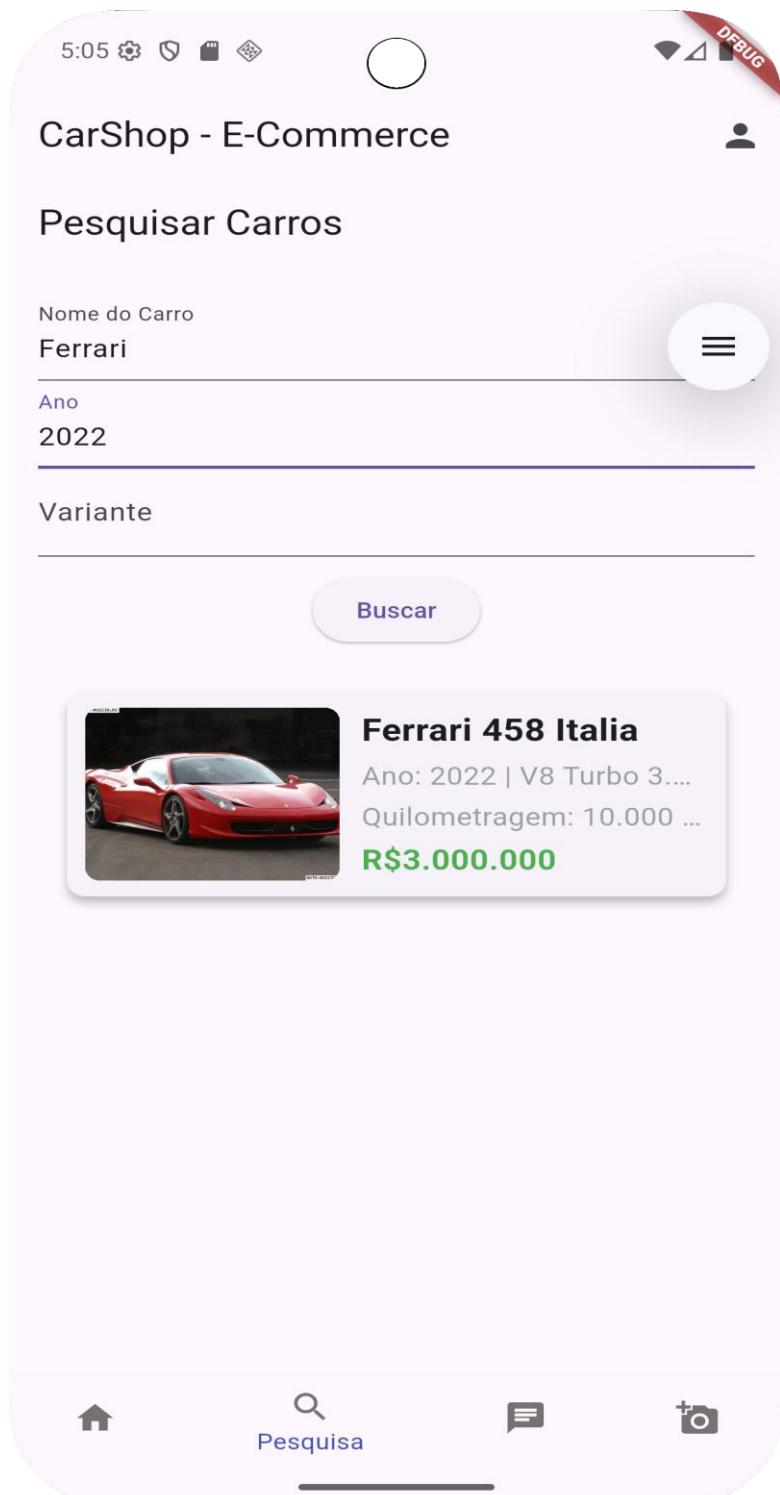
Entrar

Não tem conta? [Cadastre-se](#)

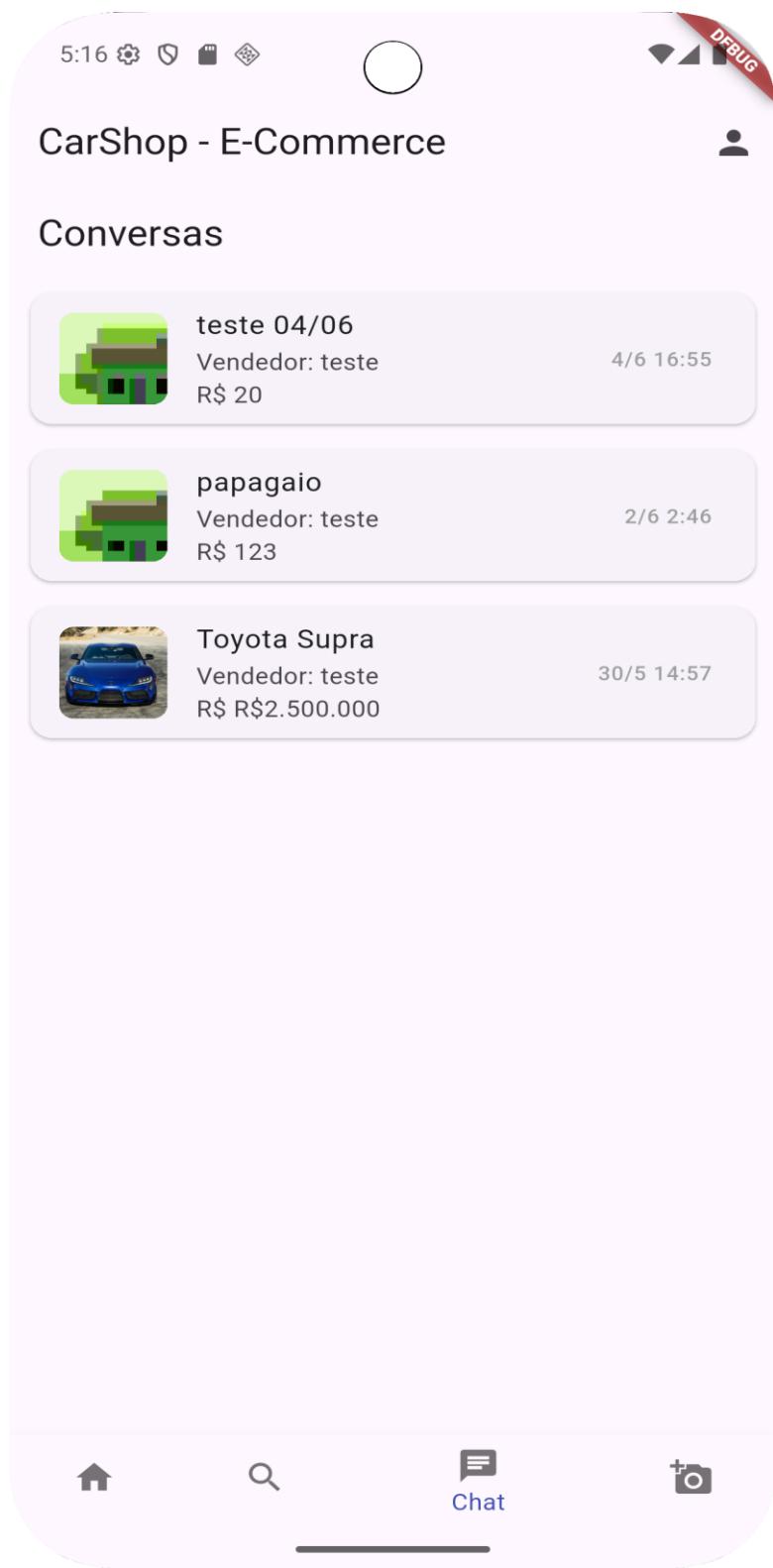
[Print da tela de criação de anúncio]



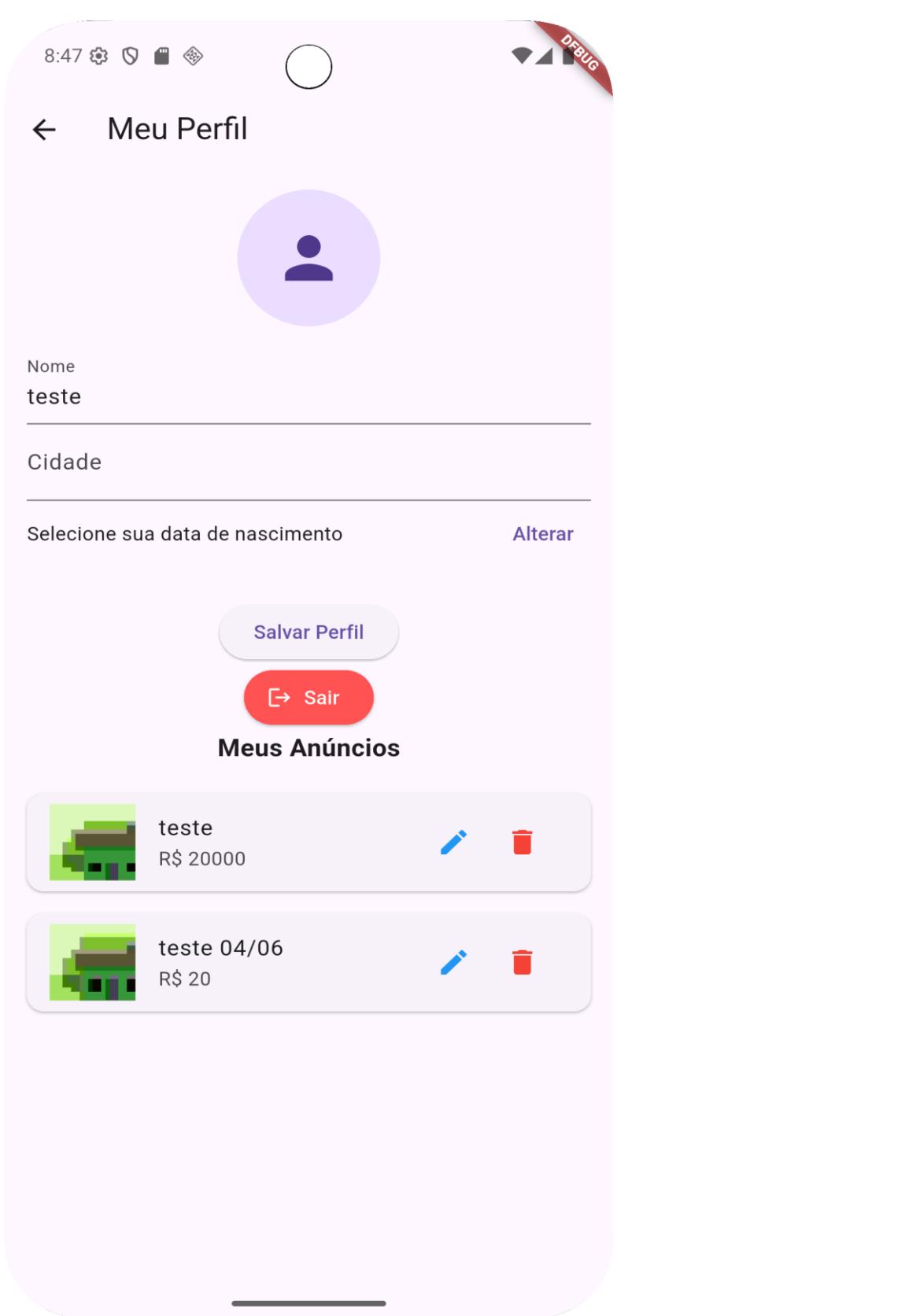
[Print da tela de pesquisa]



[Print do chat e perfil]







2.8. Especificação de Objetivos e Requisitos

Objetivo Geral

Desenvolver uma plataforma mobile para facilitar a compra e venda de veículos usados de maneira eficiente, prática e segura, utilizando uma interface intuitiva e acessível, com funcionalidades modernas e integração com uma API RESTful para persistência de dados.

Atores Envolvidos

- **Comprador:** Usuário interessado em adquirir veículos.
 - **Vendedor Particular:** Usuário que deseja anunciar seus próprios veículos.
 - **Concessionária:** Empresa que anuncia múltiplos veículos.
 - **Administrador:** Usuário com permissões avançadas para gerenciamento e moderação do sistema (via JavaFX).
 - **API RESTful:** Intermediário entre as interfaces e o banco de dados.
-

Casos de Uso e Requisitos Relacionados

Caso de Uso 01 – Cadastro de Usuário

- **Atores:** Comprador, Vendedor, Concessionária
 - **Requisitos Funcionais:**
 - RF01: O sistema deve permitir o cadastro de usuários.
 - RF02: O sistema deve validar dados antes de salvar.
 - **Requisitos Não Funcionais:**
 - RNF01: Os dados devem ser criptografados.
 - RNF02: O sistema deve responder em até 2 segundos.
-

Caso de Uso 02 – Login e Autenticação

- **Atores:** Todos os usuários
 - **Requisitos Funcionais:**
 - RF03: O sistema deve realizar login com email e senha.
 - RF04: O sistema deve permitir recuperação de senha.
 - **Requisitos Não Funcionais:**
 - RNF01: A autenticação deve ser segura e compatível com LGPD.
-

Caso de Uso 03 – Criar Anúncio

- **Atores:** Vendedor, Concessionária
 - **Requisitos Funcionais:**
 - RF05: O sistema deve permitir a criação de anúncios com múltiplas fotos.
 - RF06: O sistema deve validar os campos do veículo.
 - **Requisitos Não Funcionais:**
 - RNF03: A interface deve ser responsiva.
 - RNF04: O sistema deve permitir upload de imagens em até 5 segundos.
-

Caso de Uso 04 – Buscar Veículos

- **Atores:** Comprador
 - **Requisitos Funcionais:**
 - RF07: O sistema deve permitir buscas com filtros (modelo, preço, ano etc.).
 - RF08: O sistema deve ordenar os resultados por relevância ou preço.
 - **Requisitos Não Funcionais:**
 - RNF05: O tempo de resposta da busca não deve exceder 3 segundos.
-

Caso de Uso 05 – Visualizar Detalhes

- **Atores:** Comprador
 - **Requisitos Funcionais:**
 - RF09: O sistema deve exibir todas as informações técnicas e fotos do veículo.
 - **Requisitos Não Funcionais:**
 - RNF06: A experiência visual deve se adaptar ao dispositivo (responsividade).
-

Caso de Uso 06 – Enviar e Receber Mensagens (Chat)

- **Atores:** Comprador e Vendedor
 - **Requisitos Funcionais:**
 - RF10: O sistema deve permitir troca de mensagens entre usuários.
 - RF11: O sistema deve armazenar o histórico de conversas.
 - **Requisitos Não Funcionais:**
 - RNF07: O chat deve funcionar em tempo real com baixo consumo de rede.
-

Caso de Uso 07 – Gerenciar Perfil

- **Atores:** Todos os usuários
 - **Requisitos Funcionais:**
 - RF12: O sistema deve permitir editar informações pessoais e visualizar histórico.
 - **Requisitos Não Funcionais:**
 - RNF08: As informações devem ser persistidas com segurança.
-

2.9. Design de Software

Padrões de Design Utilizados

Durante o desenvolvimento do sistema, foram adotados diversos padrões de design com o objetivo de estruturar o código de forma limpa, coesa e de fácil manutenção:

- **MVC (Model-View-Controller):** Utilizado principalmente no frontend Flutter para organizar o código separando responsabilidades.
 - Model: Contém as classes que representam as entidades (ex: Carro, Usuário).
 - View: Representada pelas screens, que exibem a interface gráfica ao usuário.
 - Controller: Substituído por services e gerenciadores de estado (quando aplicável), que cuidam da lógica e da comunicação com a API.
 - **Injeção de Dependência:** Aplicada na organização dos services (como CarService) para desacoplar as funcionalidades e facilitar testes e manutenção.
 - **Interface/Abstract Class:** Utilizadas para definir contratos claros entre a interface e a implementação (ex: ICarService), possibilitando a troca de implementações sem afetar a aplicação como um todo.
 - **Builder Pattern (parcial):** Utilizado na construção dos objetos Car, facilitando a criação de instâncias com dados provenientes da API JSON.
-

Decisões de Design

As principais decisões de design foram orientadas pelos seguintes princípios:

- **Separação de Responsabilidades:** Cada parte da aplicação foi organizada por função (modelo, tela, serviço, widget), garantindo modularidade e reusabilidade.
 - **Baixo Acoplamento e Alta Coesão:** Os módulos foram desenhados para depender o mínimo possível uns dos outros, mantendo foco em uma única responsabilidade por classe ou função.
 - **Escalabilidade e Testabilidade:** A estrutura do projeto permite o crescimento do código sem se tornar difícil de entender ou manter. O uso de interfaces e serviços injetáveis facilita a criação de testes unitários e mocks.
 - **Aderência à Arquitetura Limpa:** Sempre que possível, adotou-se a ideia de camadas bem definidas (dados, domínio, apresentação), inspirada na Clean Architecture, para manter o controle sobre o fluxo de dados.
 - **Facilidade de Manutenção e Evolução:** As decisões de design visam facilitar futuras modificações, adição de novas funcionalidades e correções de bugs, sem comprometer a estrutura existente.
-

2.10. Ciclo de Vida do Software

O ciclo de vida do software adotado no projeto foi baseado na **metodologia ágil Scrum**, adaptada para o contexto acadêmico. Essa abordagem permitiu maior flexibilidade, entregas iterativas, revisões contínuas e maior envolvimento dos membros da equipe.

O Scrum foi escolhido por sua capacidade de dividir o projeto em partes menores e entregáveis, chamadas **sprints**, promovendo ciclos de desenvolvimento curtos e focados. Isso viabilizou uma gestão mais eficaz do tempo e permitiu rápidas correções e adaptações com base no progresso real do projeto.

As práticas principais adotadas foram:

- Reuniões de planejamento de sprint (início de cada semana)
 - Reuniões rápidas de acompanhamento (daily meetings)
 - Reuniões de revisão e retrospectiva ao final de cada sprint
-

Etapas do Ciclo de Vida

1. Concepção e Planejamento

- Definição do tema e escopo do projeto
- Formação da equipe
- Escolha das tecnologias
- Levantamento dos requisitos e elaboração inicial do cronograma

2. Modelagem e Arquitetura

- Modelagem de dados e entidades principais
- Definição da arquitetura do sistema (camadas, API REST, clientes Flutter e JavaFX)
- Criação dos primeiros diagramas e estrutura de pacotes

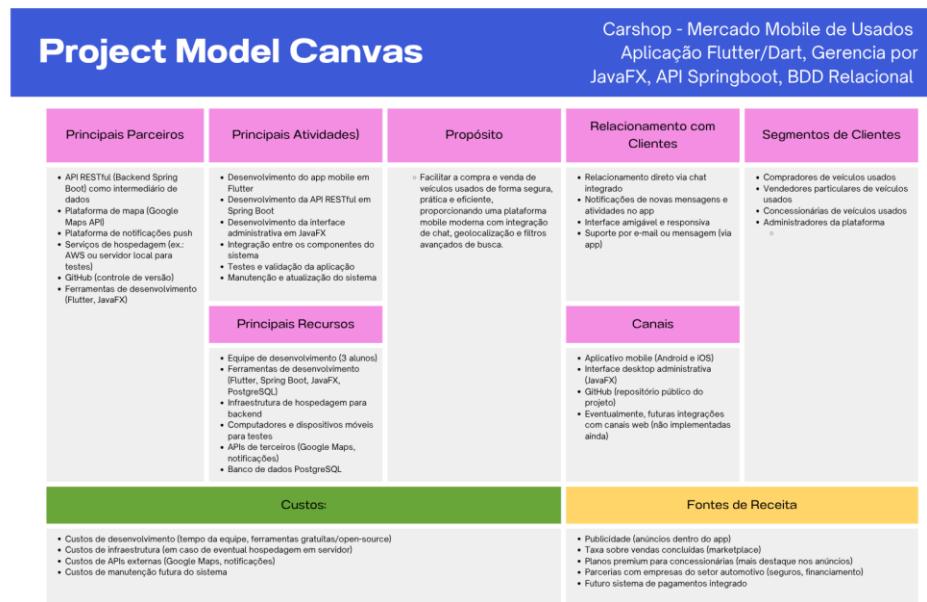
3. Desenvolvimento

- Desenvolvimento do backend RESTful com Java e Spring Boot
- Criação da interface administrativa com JavaFX
- Implementação da aplicação mobile com Flutter
- Integração entre os componentes do sistema

4. Testes

- Testes manuais e automatizados nos principais fluxos do sistema
 - Verificação de funcionamento dos endpoints da API
 - Testes de usabilidade no app
-

2.11 Project Model Canvas



3. CRITÉRIOS DE AVALIAÇÃO

- Completude dos artefatos: 25%
 - Qualidade técnica: 25%
 - Organização e clareza: 25%
 - Engajamento e proatividade: 25%
-

4. ATRIBUIÇÃO DAS NOTAS POR DISCIPLINA

ADS1251 – FERRAMENTAS VISUAIS: 20%

ADS1252 – MOBILE: 20%

ADS1253 – POO/BANCO: 20%

ADS1254 – GERÊNCIA DE PROJETOS: 20%

CMP1024 – GOVERNANÇA (EaD): Extra

5. CONSIDERAÇÕES FINAIS

A equipe aplicou os conhecimentos adquiridos no curso para modelar, implementar e documentar um sistema completo e funcional, com potencial real de uso no mercado de veículos usados.

5.1 Limitações e Melhorias Futuras

Limitações: Sem sistema de pagamento, sem painel web (Aplicação web diferente.)

Melhorias futuras: Avaliações, pagamento, mensagens de voz, login por redes sociais.

6. LINK DO GITHUB

Link do Repositório do Projeto: [GitHub - Lopeslz/carshop-mobile-rafael](https://github.com/Lopeslz/carshop-mobile-rafael)