UNIVERSIDADE DE FORTALEZA
VICE REITORIA DE GRADUAÇÃO
CENTRO DE CIÊNCIAS TECNOLÓGICAS
TECNÓLOGO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

MAYARA MEDINO MAIA
GABRIEL MARFIM MENDES SILVA
FRANCISCO LEVI DANTAS GOMES
LIVIA CAVALCANTE BARROS RODRIGUES
LUIZ EDUARDO BASTOS LIMA

DOCUMENTAÇÃO TÉCNICA

Sistema TRAVO de Turismo e Eventos

FORTALEZA
2025

DOCUMENTAÇÃO TÉCNICA

Sistema TRAVO de Turismo e Eventos

Este documento contém a documentação técnica do Sistema TRAVO de Turismo e Eventos desenvolvido na componente curricular N393 - Projeto Aplicado Multiplataforma como requisito para obtenção de nota.

Supervisor: Prof. Bruno Lopes, Me

FORTALEZA

2025

SUMÁRIO

1. INTRODUÇÃO

1.1.

1.2. CONTEXTO E JUSTIFICATIVA

O turismo urbano e a participação em eventos culturais, gastronômicos e de entretenimento são atividades que movimentam significativamente a economia local. No entanto, turistas e até mesmo moradores enfrentam dificuldades para descobrir quais eventos estão acontecendo próximos de sua localização, além de encontrar informações consolidadas sobre locais de interesse, cardápios, avaliações e benefícios promocionais.

Atualmente, essas informações encontram-se fragmentadas: parte delas está em redes sociais, outras em sites específicos de eventos ou mesmo dependem de indicações boca a boca. Isso gera perda de oportunidades tanto para os usuários, que deixam de aproveitar experiências relevantes, quanto para os estabelecimentos, que perdem visibilidade e potenciais clientes.

*O **TRAVO** surge como uma solução mobile que centraliza essas informações em um só lugar, proporcionando ao usuário uma experiência simplificada e prática. Com o TRAVO, o cliente poderá visualizar os eventos acontecendo ao seu redor, explorar locais próximos, consultar avaliações, cardápios e ainda utilizar cupons de desconto exclusivos.*

*Assim, o projeto busca atender tanto os **usuários finais (clientes)** que desejam explorar opções de lazer de forma prática, quanto os **estabelecimentos** que buscam ampliar sua divulgação e atrair novos consumidores.*

1.3. OBJETIVOS

Desenvolver um aplicativo mobile que centralize informações sobre eventos e locais turísticos próximos ao usuário, oferecendo uma experiência personalizada que integre avaliações, cardápios, favoritos e cupons de desconto.

Dito isso, os objetivos específicos do projeto são:

- *Implementar um sistema de autenticação com login, cadastro e recuperação de senha.*
- *Desenvolver um dashboard que exiba mapa com locais próximos, eventos em destaque e informações personalizadas para o usuário.*
- *Criar tela de visualização de locais, contendo resumo, cardápio, avaliações, redes sociais e cupons vinculados ao estabelecimento.*
- *Implementar uma área de cupons com listagem geral e funcionalidade de resgate.*
- *Desenvolver funcionalidade de favoritos para que o usuário possa salvar locais e acessá-los rapidamente.*
- *Criar tela de perfil para gerenciamento de dados pessoais e configurações básicas.*
- *Garantir feedback em tempo real ao usuário (mensagens de erro, confirmações de ações, carregamento dinâmico).*
- *Promover integração futura com clubes e descontos exclusivos para usuários assinantes.*

1.4. ESCOPO E DELIMITAÇÃO

- ***Escopo:***
    - ***Tela de Login****: autenticação com validação de credenciais e mensagens de erro.*
    - ***Tela de Cadastro****: formulário para criação de conta com campos essenciais.*
    - ***Tela de Recuperar Senha****: fluxo para redefinição de senha.*
    - ***Tela Dashboard****: exibição de mapa interativo, locais próximos, eventos em destaque e mensagem personalizada de boas-vindas.*
    - ***Tela de Local****: informações sobre estabelecimentos (resumo, cardápio, avaliações, cupons disponíveis).*
    - ***Tela de Cupons****: listagem completa de cupons de todos os estabelecimentos, com opção de resgate.*
    - ***Tela de Favoritos****: locais favoritados pelo usuário para acesso rápido.*
    - ***Tela de Perfil****: edição de informações pessoais (nome, e-mail, foto de perfil, etc.).*

    - ***Configurações básicas****: logout, notificações e preferências gerais.*

- ***Delimitação (Fora do Escopo):***
    - *O aplicativo não incluirá gerenciamento financeiro, emissão de notas fiscais ou integração com sistemas de contabilidade.*
    - *Não será desenvolvido um módulo para cadastro direto de estabelecimentos (nesta versão, o foco é no cliente final).*
    - *Não haverá integração inicial com meios de pagamento ou reservas online.*
    - *O app não fará gestão logística ou controle de entregas.*

2. ENGENHARIA DE REQUISITOS

2.1. REQUISITOS FUNCIONAIS (RFs)

ID	Nome do Requisito	Descrição
RF01	Autenticação de Usuário	O aplicativo deve permitir que um usuário se autentique com e-mail e senha. Após o login bem-sucedido, o app deve manter a sessão ativa até que o usuário escolha sair.
RF02	Cadastro de Usuário	O aplicativo deve permitir que novos usuários se cadastrem informando nome, e-mail, telefone e senha.
RF03	Recuperação de Senha	O aplicativo deve permitir que o usuário recupere a senha através do e-mail cadastrado.
RF04	Dashboard Interativo	O aplicativo deve exibir um dashboard inicial com mapa mostrando locais próximos à localização do usuário, eventos em destaque e mensagem de boas-vindas personalizada.
RF05	Localização	O aplicativo deve utilizar a localização do dispositivo para recomendar locais e eventos próximos.
RF06	Visualização de Locais	O aplicativo deve exibir informações de cada local, incluindo resumo, cardápio, avaliações e cupons disponíveis.
RF07	Favoritar Locais	O aplicativo deve permitir que o usuário favorite locais para acessá-los rapidamente depois.
RF08	Visualização de Favoritos	O aplicativo deve exibir a lista de locais favoritos salvos pelo usuário.
RF09	Listagem de Cupons	O aplicativo deve exibir todos os cupons disponíveis para o usuário.
RF10	Resgate de Cupom	O aplicativo deve permitir que o usuário resgate cupons e exibir confirmação de sucesso.
RF11	Avaliações de Locais	O aplicativo deve exibir avaliações feitas por outros usuários sobre os locais.
RF12	Perfil do Usuário	O aplicativo deve permitir que o usuário edite

		informações pessoais como e-mail, senha e foto de perfil.
RF13	Logout	O aplicativo deve permitir que o usuário encerre sua sessão manualmente.
RF14	Configurações Básicas	O aplicativo deve permitir acesso às configurações, como notificações e preferências gerais.
RF15	Busca e Filtros no Mapa	O usuário deve poder buscar locais/eventos pelo nome e aplicar filtros (ex: tipo de local, categoria do evento).

2.2. REQUISITOS NÃO FUNCIONAIS (RNFs)

Desempenho

- **RNF01**: O tempo de inicialização do aplicativo não deve exceder **3 segundos** em dispositivos compatíveis.
- **RNF02**: O mapa no dashboard deve ser carregado em até **2 segundos** após a inicialização da tela.
- **RNF03**: A navegação entre telas deve ocorrer em menos de **1 segundo**.

Usabilidade

- **RNF04**: A interface deve seguir as diretrizes de **Material Design 3 (Android)**, garantindo uma experiência consistente e intuitiva.
- **RNF05**: O aplicativo deve oferecer feedback imediato para todas as ações do usuário (ex: mensagens de erro, confirmações, loaders).

Compatibilidade

- **RNF06**: O aplicativo deve ser totalmente funcional em dispositivos Android a partir da versão **9.0 (API 28)**.
- **RNF07**: O app deve se adaptar automaticamente a diferentes tamanhos de tela (smartphones e tablets).

Segurança

- **RNF08**: O token de autenticação do usuário deve ser armazenado de forma segura utilizando **EncryptedSharedPreferences (Android Jetpack)**.

- **RNF09**: Todas as comunicações com o backend devem ser realizadas via **HTTPS (TLS 1.2 ou superior)**.
- **RNF10**: As senhas devem ser armazenadas de forma criptografada no servidor (ex: **bcrypt**).

Consumo de Recursos

- **RNF11**: O aplicativo deve minimizar o consumo de dados móveis utilizando cache local para imagens e informações já carregadas.

3. PROJETO E ARQUITETURA DO SOFTWARE

3.1. ARQUITETURA GERAL

Padrão adotado: Clean Architecture + MVVM nas camadas de apresentação. Motivação: Clean Architecture promove separação de responsabilidades, independência de frameworks e fácil testabilidade. MVVM (ViewModel + StateFlow) organiza o estado da UI e se integra bem com coroutines e Jetpack components.

Camadas e responsabilidades

1. Presentation (UI)

- Implementação: Jetpack Compose (recomendado) ou XML.
- Componentes: Screens/Composables, UI State classes, efeitos de UI.
- Responsabilidade: renderizar dados e capturar interações do usuário; enviar intents/ações ao ViewModel.

2. Domain

- Implementação: Use-cases / Interactors, modelos de domínio (Kotlin data classes).
- Responsabilidade: lógica de negócio independente de plataforma.

3. Data

- Implementação: Repositórios, DataSources (Remote, Local), DTOs e mappers.
- Responsabilidade: fornecer a fonte única da verdade, orquestrar chamadas à API e persistência local, aplicar estratégia de cache e sincronização.

4. Frameworks / External

- Implementação: Retrofit/OkHttp, Room, Coil/Glide, Firebase (opcional para push), EncryptedSharedPreferences, WorkManager.
- Responsabilidade: bibliotecas concretas para rede, persistência, imagens, notificações e background jobs.

Comunicação entre camadas

- A View observa o ViewModel (StateFlow/LiveData).
- O ViewModel chama UseCases (domain) que orquestram repositórios.
- Os Repositórios consultam fontes remota (API) e local (Room), aplicando a estratégia “Single Source of Truth” (UI observa dados do local; atualização da rede atualiza o local).

3.2. PROJETO DE DADOS: INTEGRAÇÃO COM API E PERSISTÊNCIA LOCAL

Estratégia geral

- **Fonte da verdade:** API REST do backend.
- **Cache local:** Room — app **lê sempre do banco local** e o Repositório atualiza o banco após respostas da API. Assim a UI é reativa e funcional offline.
- **Padrão:** Single Source of Truth (SSOT).
- **Fluxo padrão**:

    1. ViewModel solicita dados ao UseCase.
    2. UseCase chama Repository.getX().
    3. Repository retorna `Flow` observado do banco local (Room). Simultaneamente dispara request à API (se online) para atualizar o banco local.
    4. Ao chegar a resposta, Repository salva (ou faz merge) em Room; a UI observa alterações e atualiza automaticamente.

Mecanismos e políticas

- **Cache invalidation / TTL:** para coleções dinâmicas (eventos/cupons), use timestamps `lastFetched` e política TTL (ex.: 10–30 min) antes de forçar refresh de rede.
- **Paginação:** quando listar locais e eventos, use Paging 3 (suporta integração Room + Retrofit) para rolagem eficiente.
- **Imagens:** usar *Coil* (Kotlin-first) com cache em disco e placeholders.
- **Sincronização offline:** ações do usuário (favoritar, resgatar cupom) escrevem em Room imediatamente e geram uma *fila de sync* (tabela `PendingActions`) que WorkManager processa quando houver conectividade.
- **Conflito de dados:** na sincronização, priorizar o servidor como fonte da verdade; quando conflito local vs remoto for crítica (ex.: resgate de cupom), usar lógica otimista + verificação de resposta do servidor e feedback ao usuário.

3.3. CONSUMO DA API E FLUXO DE NAVEGAÇÃO

Exemplo de endpoints

- `POST /auth/login` — body: `{ email, password }` → retorno: `{ accessToken, refreshToken, user }`
- `POST /auth/register` — body: `{ name, email, password }`
- `POST /auth/refresh` — body: `{ refreshToken }`
- `GET /users/{id}` — obter perfil
- `GET /places?lat={}&lng={}&radius={}` — listar locais próximos
- `GET /places/{id}` — detalhes do local
- `GET /places/{id}/menu` — cardápio (se aplicável)
- `GET /places/{id}/coupons` — cupons do local
- `GET /events?lat={}&lng={}&from={}&to={}` — eventos próximos/por período
- `POST /coupons/{id}/redeem` — resgatar cupom
- `POST /favorites` — body: `{ userId, placeId }`
- `DELETE /favorites/{id}` — remover favorito

Fluxo de rede típico (ex.: abrir Dashboard)

1. App inicia Dashboard → ViewModel solicita `getDashboardData()`.
2. Repository retorna `Flow` observando o banco local (places/events cached). UI rende dados existentes (se houver).
3. Repository verifica TTL e se necessário faz chamadas:
    - `GET /places?lat,lng,radius`
    - `GET /events?lat,lng,from,to`
4. Ao receber respostas, Repository persiste em Room. UI recebe atualização automaticamente.
5. A cada ação do usuário (ex.: favoritar), escrever em Room e adicionar `PendingAction` para enviar ao servidor com WorkManager.

4. TECNOLOGIAS E FERRAMENTAS

4.1. STACK DE TECNOLOGIAS

O sistema Travoo foi construído com uma arquitetura moderna e escalável, utilizando as seguintes tecnologias:

- Linguagem (Mobile): Kotlin, a linguagem oficial para o desenvolvimento de aplicativos Android. Foi escolhida por sua segurança (null-safety) e concisão, que agiliza o processo de desenvolvimento.

- Backend: Node.js, um ambiente de execução JavaScript, utilizado para o desenvolvimento da plataforma web e, possivelmente, de funções serverless. Sua arquitetura não-bloqueante é ideal para aplicações com alta demanda de I/O.

- Banco de Dados e Backend as a Service (BaaS): Supabase, que funciona como um backend completo, fornecendo um banco de dados PostgreSQL robusto, autenticação, APIs instantâneas e armazenamento de arquivos. A utilização do Supabase simplifica a gestão do backend, permitindo que a equipe se concentre no desenvolvimento das funcionalidades.

- Arquitetura (Mobile): Android Architecture Components, como ViewModel e LiveData, para garantir uma arquitetura robusta e reativa, separando a lógica de negócios da interface do usuário.

- UI Toolkit (Mobile): Jetpack Compose, para construir a interface de usuário do aplicativo de forma declarativa e moderna, com menos código e maior velocidade.

- Consumo de API (Mobile): Retrofit 2 e OkHttp 3, utilizados para realizar chamadas de rede à API do Supabase de forma eficiente e segura.

4.2. FERRAMENTAS DE DESENVOLVIMENTO

Para apoiar o processo de desenvolvimento do Travoo, a equipe utilizou um conjunto de ferramentas que otimizaram a colaboração e a gestão do projeto:

- *IDE (Mobile): Android Studio, a IDE oficial do Google, que oferece todas as ferramentas necessárias para o desenvolvimento de aplicativos Android.*

- *IDE (Backend): Visual Studio Code, que serviu como a IDE padrão para o desenvolvimento com Node.js e outras tarefas, devido à sua leveza, extensibilidade e vasto ecossistema de plugins.*

- *Controle de Versão: Git, com o repositório hospedado no GitHub. Adotamos o fluxo de trabalho GitFlow, com branches separadas para develop, features e main, garantindo um histórico de versões organizado.*

- *Gerenciamento de Projeto: Git Backlog (Projeto) foi a ferramenta utilizada para gerenciar as tarefas da equipe. Criamos um quadro Kanban com as colunas "A Fazer", "Em Andamento", "Em Teste" e "Concluído".*

- *Ferramenta de Teste de API: Insomnia ou Postman, utilizados para testar os endpoints da API durante o desenvolvimento, assegurando que o backend e o frontend se comunicassem corretamente.*

5. IMPLEMENTAÇÃO E RESULTADOS

5.1. TELAS DO SISTEMA

Figura 1: Tela de Login

A tela de login é o ponto de entrada do sistema TRAVO Mobile.

Ela permite que o usuário acesse o aplicativo informando seu e-mail e senha.

Além disso, disponibiliza a opção “Esqueceu sua senha?” para recuperação de acesso e dois botões de ação:

- Cadastrar: direciona o usuário para a tela de cadastro de nova conta.

- Entrar: valida as credenciais e autentica o usuário no sistema.

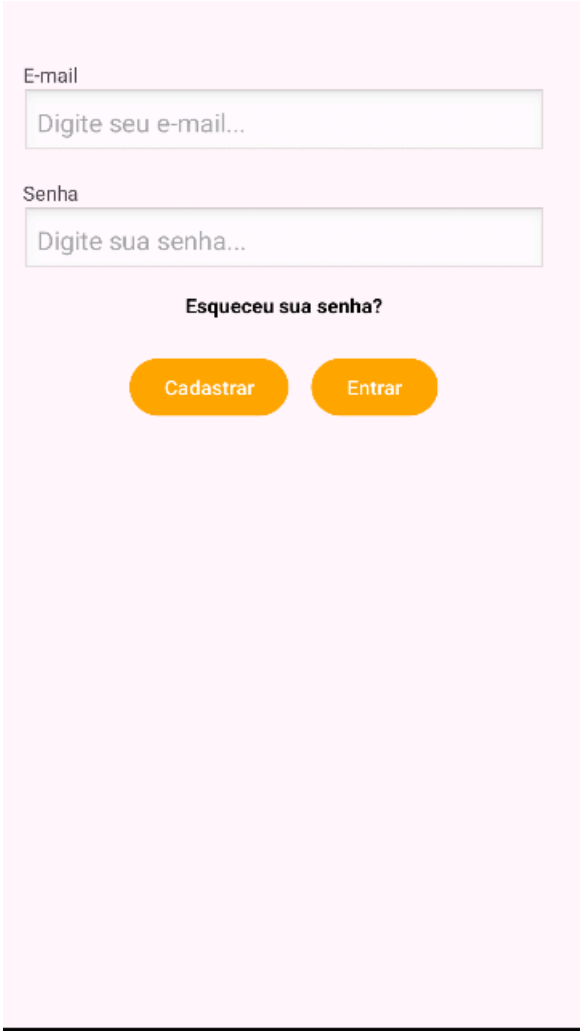


Figura 2: Tela de Cadastro

A tela de cadastro permite que novos usuários criem uma conta no TRAVO Mobile.

Ela solicita informações essenciais para o registro, incluindo nome completo, e-mail, telefone (opcional) e senha.

Após o preenchimento, o usuário deve clicar em Cadastrar para concluir o processo.

Também há a opção "Já tem conta? Entrar", que redireciona o usuário para a tela de login.

Figura 3: Tela de Recuperação de Senha

A tela de recuperação de senha permite que o usuário redefina o acesso à sua conta TRAVO Mobile.

Para isso, basta inserir o e-mail cadastrado e clicar no botão Recuperar, que enviará instruções de redefinição.

A interface também oferece a opção "Voltar para Login", facilitando o retorno à tela inicial de autenticação.

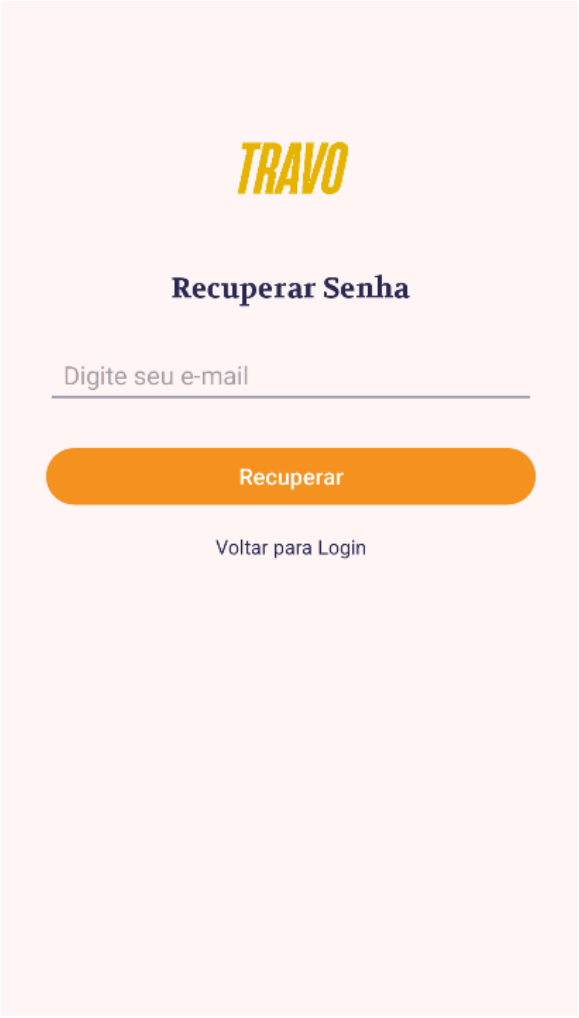


Figura 4: Tela de Configurações

A tela de recuperação de senha permite que o usuário redefina o acesso à sua conta TRAVO Mobile.

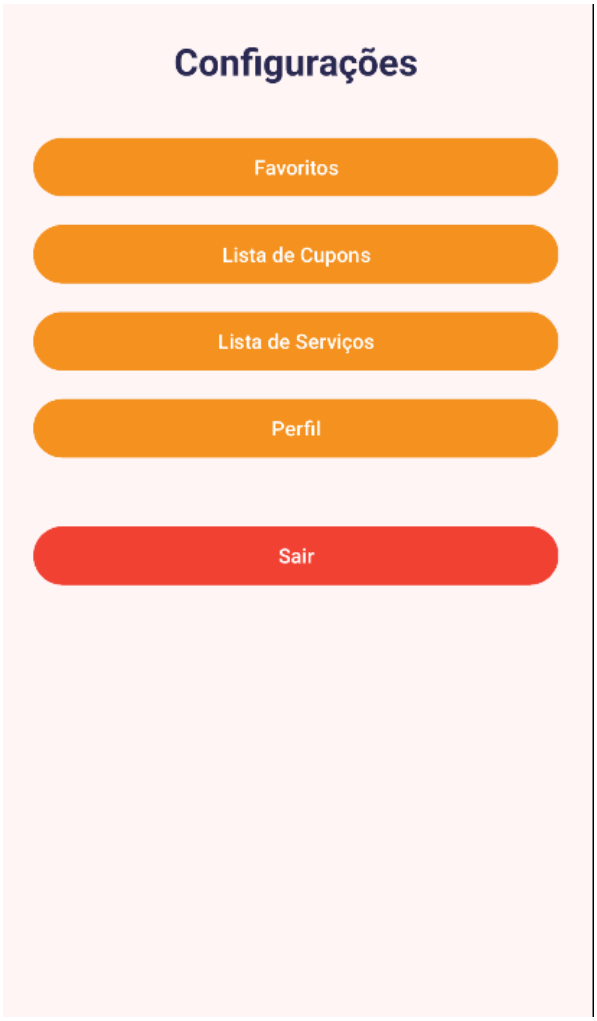


Figura 5: Tela de Detalhes do Local

A tela de detalhe do local apresenta informações completas sobre um estabelecimento específico no TRAVO Mobile.

Exibe dados como nome, endereço, tipo, horário de funcionamento e um resumo descritivo do local.

Além disso, disponibiliza cupons de desconto ativos e uma seção de avaliações de usuários, que inclui comentários, notas e percentuais de satisfação.

A navegação inferior permite alternar rapidamente entre as seções Descontos, Home e Configurações, garantindo uma experiência fluida dentro do aplicativo.

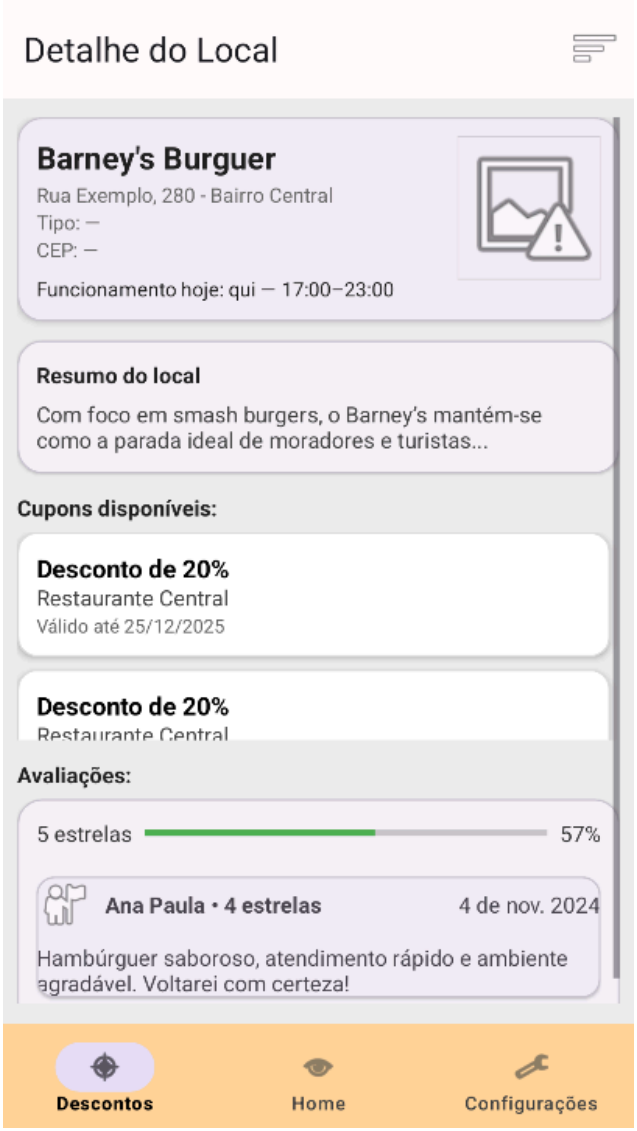


Figura 6: Tela de Favoritos

A tela de favoritos apresenta informações completas sobre um estabelecimento específico que o usuário marcou como favorito.

A navegação inferior permite alternar rapidamente entre as seções Descontos, Home e Configurações, garantindo uma experiência fluida dentro do aplicativo.

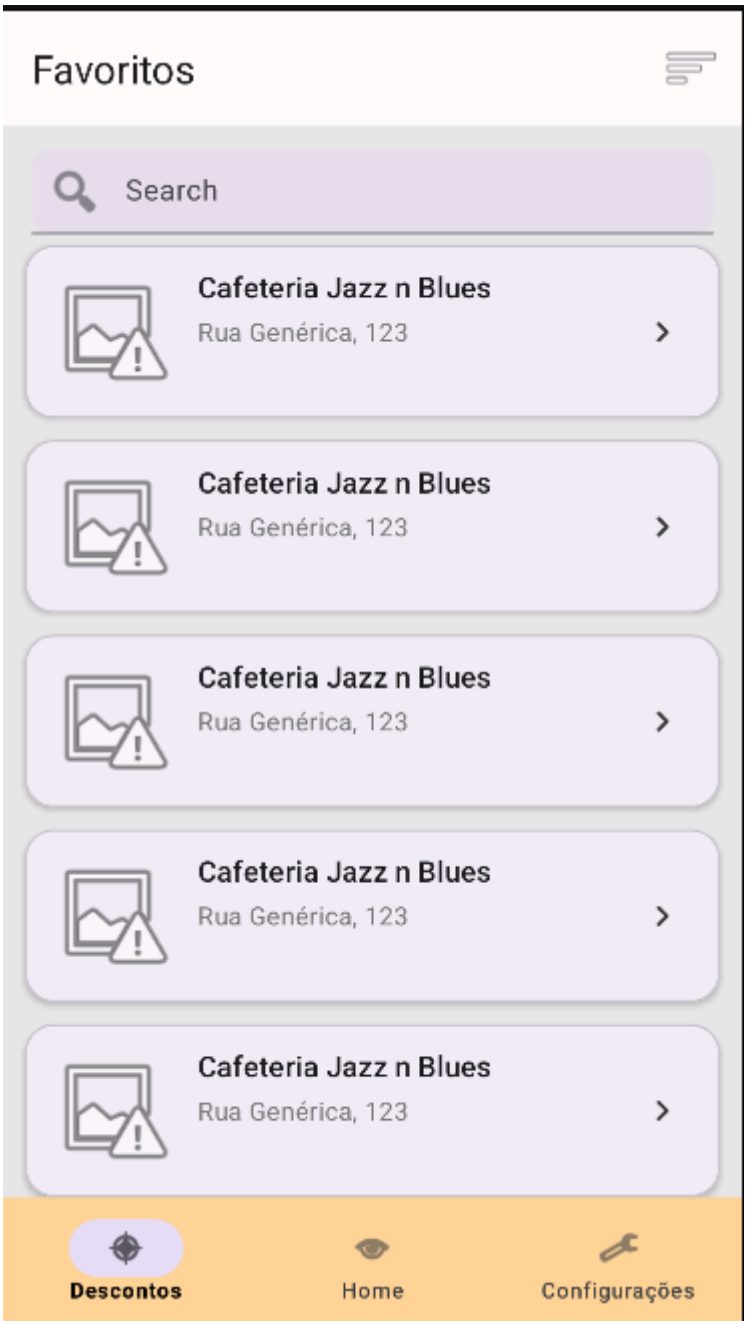


Figura 7: Tela de Listagem de Serviços

A tela de listagem de serviços permite que o usuário veja de forma rápida e em forma de listagem os estabelecimentos que estão cadastrados no app.

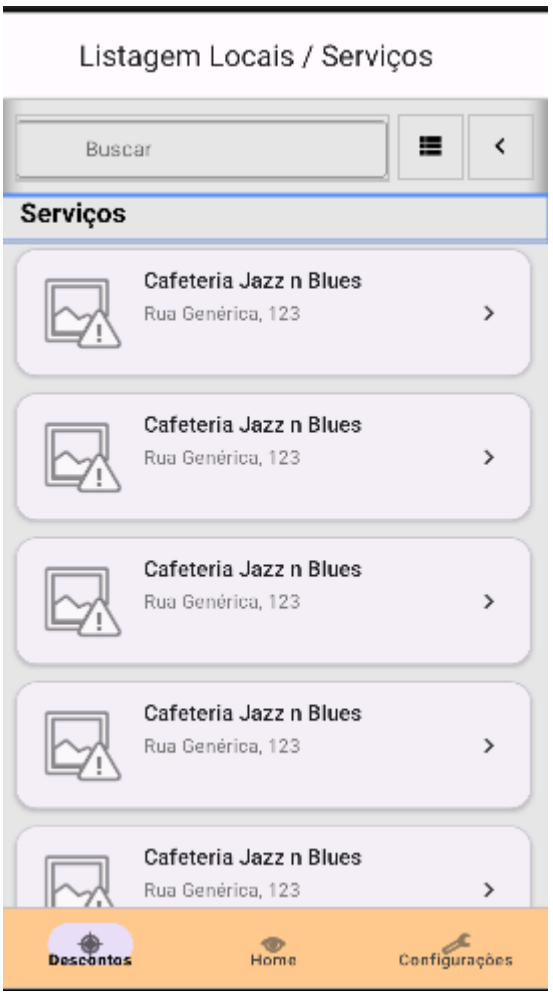


Figura 8: Tela de Listagem de Cupons

A tela de listagem de cupons apresenta os cupons que estão disponíveis para os usuários e a quais estabelecimentos cadastrados no app os cupons se referem.

Figura 9: Tela de Perfil do Usuário

A tela de perfil do usuário exibe as principais informações pessoais cadastradas no TRAVO Mobile.

Apresenta foto do usuário, nome, biografia curta e e-mail. Também são mostrados dados complementares como data de nascimento e telefone.

O botão Editar permite que o usuário acesse a tela de edição de perfil para atualizar seus dados.

Na parte inferior, há uma lista de locais visitados, representada por miniaturas e nomes dos locais, reforçando a interação do usuário com o aplicativo.

A barra de navegação inferior possibilita o acesso rápido às seções Descontos, Home e Configurações, mantendo a consistência visual entre as telas do aplicativo.

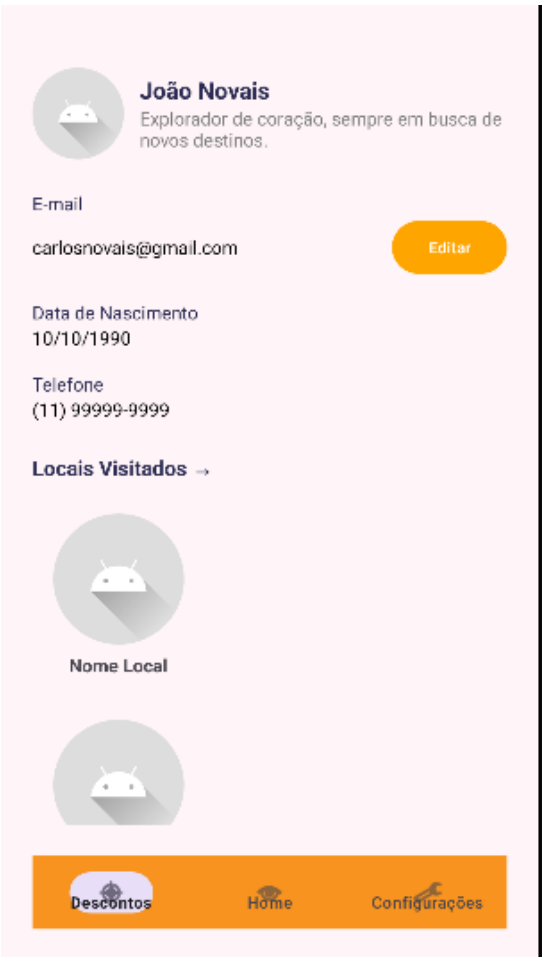


Figura 10: Tela de Edição de Perfil do Usuário

A tela de edição de perfil oferece ao usuário a possibilidade de atualizar suas informações pessoais de forma simples e intuitiva.

Contém campos para nome, bio e telefone, além de permitir a alteração da foto de perfil.

O botão Salvar garante que as mudanças realizadas sejam registradas.

A seção Alterar senha possibilita redefinir a senha, exigindo o preenchimento da senha antiga, nova senha e confirmação da nova senha, reforçando a segurança dos dados.

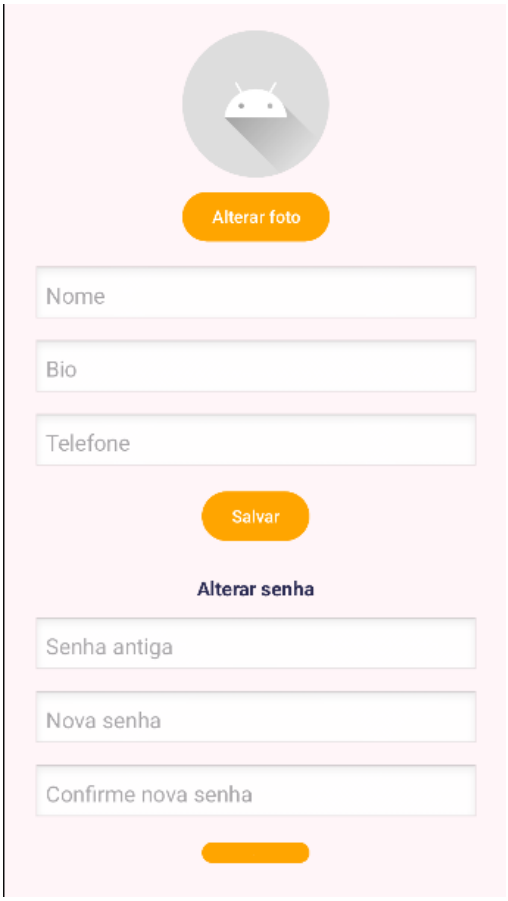
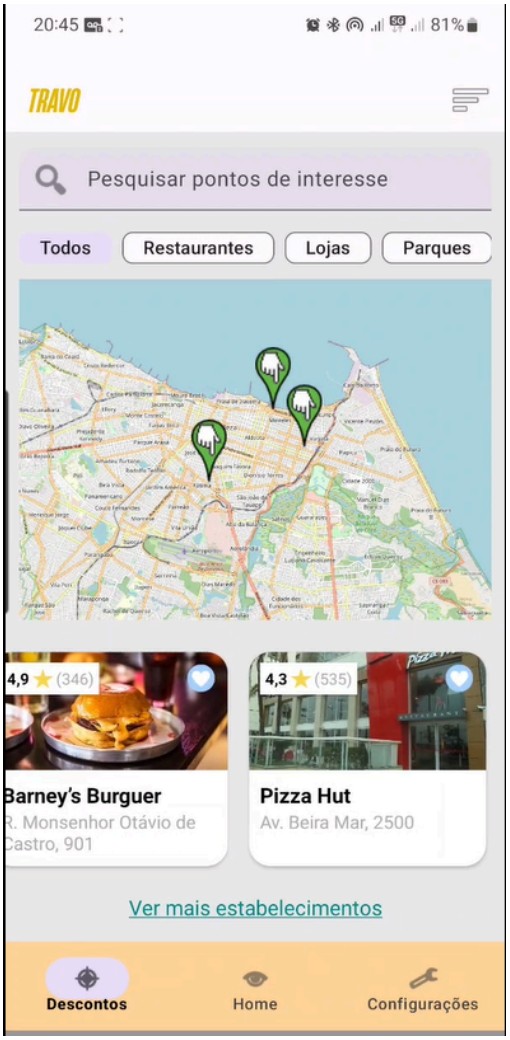


Figura 11: Tela de Home

A tela de home é a tela que ao usuário logar no sistema ela parecerá em seguida, mostrando sua localização e os locais que estão perto.



6. AMBIENTE E GUIA DE GERAÇÃO (BUILD)

Esta seção descreve o ambiente necessário e os passos para compilar o código-fonte e gerar o arquivo de instalação (.apk) do aplicativo TRAVO Mobile.

6.1. REQUISITOS DO AMBIENTE

Para garantir a compilação correta do projeto, recomenda-se utilizar as seguintes versões e ferramentas:

- ***IDE: Android Studio Narwhal 4 Feature Drop 2025.1.4***
- ***Gradle: a versão que já vem no app quando baixado***
- ***JDK: embutido no Android Studio***
- ***Sistema Operacional: Windows 10/11, macOS Monterey ou Linux Ubuntu 22.04***
- ***Emulador/Dispositivo: Android 10 ou superior***

6.2. PROCESSO DE GERAÇÃO DE APLICATIVO

Para gerar a versão release do aplicativo TRAVO Mobile, siga as instruções abaixo conforme seu sistema operacional:

1. Clone o repositório do projeto:

```
git clone https://github.com/medinodev/travo-mobile
```

2. Acesse o diretório do projeto:

```
cd travo-mobile
```

3. Gere o arquivo de instalação (.apk):

No Windows:

```
gradlew.bat assembleRelease
```

No Mac:

./gradlew assembleRelease

4. Após a conclusão, o arquivo de instalação será criado em:

app/build/outputs/apk/release/app-release.apk

7. CONCLUSÃO

7.1. TRABALHOS FUTUROS

Embora o TRAVO Mobile apresente uma base sólida de funcionalidades, ainda existem diversas oportunidades de melhoria que podem tornar o aplicativo mais completo, robusto e útil para os usuários.

- ***Melhoria do Layout:*** *Há uma grande oportunidade de deixar a navegação mais agradável, intuitiva e visualmente harmoniosa, proporcionando uma experiência mais fluida em todas as telas.*

- ***Integração de meios de pagamento e reservas online:*** *A possibilidade de integrar meios de pagamento ou sistemas de reservas online, permitindo que o usuário finalize compras ou agendamentos diretamente pelo aplicativo.*

- ***Recomendações Personalizadas:*** *A implementação de recomendações personalizadas, baseadas no histórico e nas preferências do usuário, oferecendo locais e eventos mais relevantes.*

- ***Gamificações:*** *Para aumentar o engajamento, funcionalidades como gamificação, acúmulo de pontos por check-in, avaliações e resgate de cupons podem ser integradas.*

7.2. LIÇÕES APRENDIDAS

Durante o desenvolvimento do TRAVO Mobile, enfrentamos diversos desafios técnicos que contribuíram significativamente para o nosso aprendizado. Optamos por utilizar Kotlin por já termos experiência prévia com a linguagem, porém o TRAVO exigiu um nível de complexidade muito maior do que havíamos trabalhado antes. Com isso, foi necessário pesquisar constantemente, testar diferentes abordagens e, muitas vezes, reconstruir partes inteiras até encontrar soluções eficientes e consistentes para cada funcionalidade essencial do aplicativo.

A integração com a API do backend foi um dos pontos mais desafiadores e, ao mesmo tempo, mais enriquecedores do projeto. Por ser uma experiência nova para a equipe, exigiu grande dedicação para compreender o funcionamento das requisições, estruturar corretamente os endpoints e lidar com respostas e erros de forma adequada. Esse processo foi fundamental para permitir o avanço das demais partes do sistema e contribuiu muito para nossa evolução técnica.

No que diz respeito ao trabalho em equipe, sempre tivemos uma postura muito unida e colaborativa. Apesar das dificuldades técnicas, nunca houve problemas de comunicação. Mantínhamos um fluxo constante de alinhamento, compartilhando o que já havia sido desenvolvido, o que ainda precisava ser concluído e quais obstáculos estavam surgindo. Um dos grandes pontos fortes da nossa equipe foi a perseverança: mesmo com rotinas pessoais e profissionais intensas, todos se mantiveram comprometidos em dar o seu melhor e em ajudar uns aos outros. Essa união foi decisiva para que pudéssemos concluir o projeto com êxito.

Se tivéssemos a oportunidade de fazer algo diferente, gostaríamos de ter tido mais tempo disponível para nos dedicar ao desenvolvimento. Não se trata do tempo da disciplina, que foi muito bem planejado, mas sim de tempo pessoal para aperfeiçoar ainda mais o aplicativo, explorar funcionalidades adicionais e aprofundar aspectos técnicos com mais calma.

Essa jornada, apesar dos desafios, foi extremamente valiosa e nos proporcionou um crescimento significativo tanto individual quanto em equipe.