Documentação técnica parcial

Introdução

Este documento tem como objetivo apresentar uma visão geral da arquitetura e funcionamento parcial do aplicativo Instacar, incluindo suas principais funcionalidades, estrutura de navegação e componentes técnicos já implementados.

O projeto está em fase de desenvolvimento e visa oferecer uma plataforma eficiente e intuitiva para conectar usuários que desejam oferecer ou buscar caronas, promovendo mobilidade colaborativa de forma segura e prática.

Essa documentação será expandida e atualizada conforme o avanço do desenvolvimento, incorporando novos módulos, integrações com a API e melhorias baseadas em feedbacks e testes.

Tecnologias utilizadas

Frontend (Flutter):

- Flutter: Framework utilizado para o desenvolvimento multiplataforma do aplicativo mobile.
- Dart: Linguagem de programação principal utilizada no Flutter.

Backend (Node.js com TypeScript):

- Node.js: Ambiente de execução JavaScript para o servidor.
- TypeScript: Superset do JavaScript com tipagem estática, usado para maior segurança no desenvolvimento.
- Express: Framework web utilizado para criação de rotas e APIs REST.
- Swagger: Utilizado para documentação das rotas e testes de API.
- JWT (JSON Web Tokens): Autenticação e autorização segura entre cliente e servidor.

Outros

- Docker (futuramente): Potencial uso para empacotar o backend e facilitar o deploy.
- Postman: Utilizado para testes das rotas da API durante o desenvolvimento.

Descrição da Arquitetura

1. Aplicativo Flutter (instacar)

A arquitetura do aplicativo Flutter segue uma organização modular, com separação clara entre rotas, modelos, serviços, páginas e widgets. A estrutura é dividida da seguinte forma:

- inject.dart e main.dart: pontos de entrada da aplicação. O main.dart inicializa o app, enquanto inject.dart cuida da injeção de dependências.
- core/:
    - app_router.dart: gerenciamento centralizado de rotas usando go_router.
    - constants.dart: constantes globais da aplicação.
    - models/: define as estruturas de dados principais como RideCard e MessageModel.
    - services/: serviços como chat_service.dart que abstraem a comunicação com backend ou lógica de negócios.

- presentation/:
    - pages/: divide-se em subpastas como auth, chat, e main, organizando as telas da aplicação conforme contexto de uso (autenticação, chat e área principal).
    - widgets/: componentes reutilizáveis como BottomNavigationBar, RideListWidget e navbar, que facilitam a composição da UI.

2. API em TypeScript (instacar-api)

A API segue uma arquitetura MVC (Model-View-Controller) com boas práticas de separação de responsabilidades e padronização. A estrutura é organizada da seguinte forma:

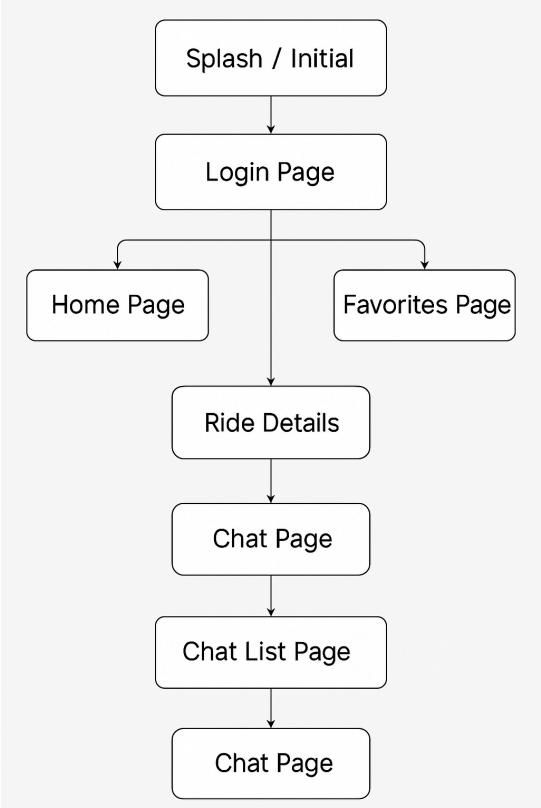
- app.ts e server.ts: ponto de inicialização da aplicação Express.
- config/: configurações de banco de dados (db.ts) e integração com Swagger (swagger.ts).
- controllers/: controladores responsáveis por lidar com as requisições HTTP e interagir com os serviços correspondentes (auth, carona, chat, user).
- interfaces/: define contratos para os serviços e modelos, auxiliando na manutenção da tipagem forte.
- middlewares/: como o Auth.ts, responsável por validação e autenticação de tokens JWT.
- models/: define os esquemas dos dados utilizados na API, como User, Carona, Chat.
- routes/: arquivos que agrupam e registram as rotas de forma organizada.
- services/: camada de lógica de negócio que implementa as regras e integrações de cada funcionalidade (ex: auth.service.ts, carona.service.ts).
- types/ e utils/: tipagens globais (index.d.ts) e utilitários reutilizáveis, como validadores e mensagens de erro.

Comunicação entre app e API

O app Flutter se comunica com a API por meio de requisições HTTP (REST), utilizando serviços como o chat_service.dart. Os dados são trafegados em formato JSON, respeitando os contratos definidos nos modelos e interfaces da API.

Fluxo de Navegação

O app Instacar segue um fluxo de navegação baseado em rotas nomeadas com GoRouter, facilitando a transição entre telas de autenticação, funcionalidades principais e chat.



Próximos Passos

Backend

- Implementar autenticação com tokens de renovação.
- Adicionar testes automatizados para os principais endpoints.
- Integrar envio de emails para notificações e confirmações.
- Adicionar suporte a upload de imagens para perfis de usuários.

Frontend

- Integrar formulários de login e cadastro com a API.
- Exibir caronas com dados reais do backend.
- Implementar sistema de favoritos com persistência.
- Ativar funcionalidade de chat em tempo real.

- Melhorar responsividade e padronização visual.

Documentação

- Finalizar documentação da API com Swagger.
- Criar diagrama das entidades do banco de dados.
- Adicionar instruções de uso e implantação do sistema.

Outros

- Estudar e configurar pipeline de CI/CD.