Macroarquitetura - ToDoAdventure

Visão Geral do Front-end:

O projeto é uma aplicação web desenvolvida em Angular (versão 13) que se comunica com um backend Flask para gerenciamento de missões. A interface do usuário é construída utilizando Angular Material (versão 13).

Tecnologias Front-end:

Bibliotecas e frameworks: Typescript, Angular: 13.0.0, Flask:3.12.0, Vue.js, Karma
Gerenciador de Pacotes: npm

Tecnologias de Estilo: Tailwind (CSS), Angular Material: 13.0.0

Arquitetura Front-end:

1. Estrutura do Projeto:

app.module.ts: Módulo principal da aplicação.
app.service.ts: Serviço principal para comunicação com o backend Flask.
components:
dialog-form-mission.component.ts: Componente para criar e editar missões.
dialog-warnings.component.ts: Componente para exibir avisos no sistema.
app.component.ts: Componente do dashboard de tarefas.

2. Módulo Principal (app.module.ts):

O módulo principal (app.module.ts) contém as declarações dos componentes, módulos importados, e configurações globais.

3. Serviço Principal (app.service.ts):

getMissions(): Observable<Mission[]>: Recupera a lista de missões do backend.
createMission(mission: Mission): Observable<Mission>: Cria uma nova missão no backend.
updateMission(mission: Mission): Observable<Mission>: Atualiza uma missão existente no backend.
deleteMission(id: number): Observable<void>: Exclui uma missão do backend

4. Componentes

DialogFormMissionComponent
O componente (dialog-form-mission.component.ts) é responsável pela criação e edição de missões. Ele utiliza o Angular Material Dialog para fornecer uma interface amigável.

DialogWarningsComponent
O componente (dialog-warnings.component.ts) exibe avisos no sistema. Pode ser chamado a partir de outros componentes para notificar o usuário sobre eventos importantes.

AppComponent
O componente (app.component.ts) representa o dashboard principal da aplicação. Ele exibe as tarefas e interage com os outros componentes conforme necessário.

Visão Geral da Arquitetura Back-end:

- Estilo da Arquitetura: API RESTful
- Linguagem de Programação: Python
- Framework Web: Flask
- Armazenamento de Dados: Arquivo JSON (db.json)
- Interações com o Banco de Dados: E/S direta no arquivo
- Compartilhamento de Recursos de Origem Cruzada (CORS): Ativado

Componentes:

Rotas:

/journeys: Manuseia operações CRUD para viagens
/journeys/<journey_id>: Manuseia atualizações, exclusões e operações de missão para uma viagem específica
/journeys/<journey_id>/missions: Manuseia operações CRUD para missões dentro de uma viagem
/setted_journey: Recupera e define a viagem selecionada no momento

Persistência de Dados:

load_db(): Carrega dados do arquivo JSON para a memória
save_db(): Salva dados da memória de volta para o arquivo JSON

Funções Utilitárias:

get_next_id(): Gera IDs exclusivos para novas viagens e missões

Interações entre Componentes:

As rotas recebem solicitações HTTP e interagem com o modelo de dados na memória.
As alterações no modelo de dados são persistidas no arquivo JSON usando save_db().

Modelo de Dados:

journeys: Uma lista de viagens, cada uma contendo:
id: Um identificador único
missions: Uma lista de missões
setted_journey: O ID da viagem selecionada no momento

Estratégia de Testes:
Testes unitários para componentes críticos.
Testes de integração para garantir a comunicação entre frontend e backend.
É utilizada Karma como ferramenta de testes.