Documentação de Arquitetura do Projeto "To Do Adventurer"

1. Visão Geral do Sistema:

- 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).

2. 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.

3. Módulo Principal (app.module.ts):

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

4. 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

5. 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.

6. Back-end Flask

O backend Flask contém uma única main que fornece endpoints para as operações CRUD (GET, PUT, DELETE, UPDATE) relacionadas às missões.

- Rota: /task

GET
Método: GET
Descrição: Recupera todas as tarefas disponíveis, em andamento e concluídas.
Resposta Bem-Sucedida:
json
Copy code

```
{
  "disponiveis": [
    {
      "id": "1",
      "nome": "Nome da Tarefa",
      "descricao": "Descrição da Tarefa",
      "status": "disponiveis"
    }
  ],
  "emAndamento": [
    {
      "id": "2",
      "nome": "Outra Tarefa",
      "descricao": "Descrição da Outra Tarefa",
      "status": "emAndamento"
    }
  ],
  "concluidas": [
    {
      "id": "3",
      "nome": "Tarefa Concluída",
      "descricao": "Descrição da Tarefa Concluída",
      "status": "concluidas"
    }
  ]
}
```

- POST
    Método: POST
    Descrição: Cria uma nova tarefa na lista de tarefas disponíveis.
    Corpo da Requisição:
    json
    Copy code
    {
      "nome": "Nome da Nova Tarefa",
      "descricao": "Descrição da Nova Tarefa"
    }
    Resposta Bem-Sucedida:
    json
    Copy code
    {
      "message": "Tarefa criada com sucesso!",
      "id": "4"
    }

- Rota: /task/{id}
    PUT
    Método: PUT
    Descrição: Atualiza os detalhes de uma tarefa específica.
    Parâmetros de Rota: {id} - ID da tarefa a ser atualizada.
    Corpo da Requisição:
    json
    Copy code
    {
      "nome": "Novo Nome da Tarefa",
      "descricao": "Nova Descrição da Tarefa",
      "status": "emAndamento"
    }
    Resposta Bem-Sucedida:
    json
    Copy code
    {
      "message": "Tarefa atualizada com sucesso!",
      "task": {
        "id": "2",
        "nome": "Novo Nome da Tarefa",
        "descricao": "Nova Descrição da Tarefa",
        "status": "emAndamento"
      }
    }

- DELETE
    Método: DELETE
    Descrição: Exclui uma tarefa específica.

Parâmetros de Rota: {id} - ID da tarefa a ser excluída.
Resposta Bem-Sucedida:
json
Copy code
{
  "message": "Tarefa excluída com sucesso!"
}

7. Estratégia de Testes:

- Testes unitários para componentes críticos.
- Testes de integração para garantir a comunicação entre frontend e backend.

8. Dependências

- Angular: 13.0.0
- Angular Material: 13.0.0
- Flask:3.12.0