Licenciatura em Engenharia Informática e de Computadores

Organização de Projeto
TASA - Theater Auto Silence App

Projeto e Seminário 2024/2025

João Marques, n.º 48297, e-mail: a48297@alunos.isel.pt,tel.: 925934664

Gonçalo Ribeiro, n.º 48305, e-mail: a48305@alunos.isel.pt, tel.: 966559629

Orientador: Artur Ferreira, e-mail: artur.ferreira@isel.pt

Julho de 2025

1 Repositório do Projeto

Toda a implementação e documentação do projeto **TASA** estão disponíveis publicamente no **GitHub**. Disponível neste link.

2 Organização do Projeto

A raiz do projeto está organizada em quatro diretorias principais:

- `docs/` - Documentação.
- `frontend/` - Implementação do *frontend*.
- `jvm/` - Implementação do *backend*.
- `scripts/` - Ficheiros de *script*.

2.1 `/docs` - Documentação e Recursos

Contém a documentação do projeto, diagramas e materiais relacionados:

- `imgs/` - Recursos gráficos (imagens).
- `problems/` - Diretoria que contém os tipos de erros retornados pela API. Cada ficheiro representa um tipo específico de erro.

2.2 `/frontend` - Aplicação Cliente

Contém a implementação da aplicação *Android* e os componentes da interface gráfica.

`/app/src` - Código-Fonte Principal

- `main/java/com/tasa` — Pacote raiz:
  - `activity/` — Gestor de reconhecimento de atividade e recetor de eventos.
  - `alarm/` — Agendamento de alarmes com *AlarmManager*.
  - `domain/` — Classes de domínio.
  - `geofence/` — Gestor de *geofencing* e recetor de eventos.
  - `infrastructure/` — Repositório *DataStore*.
  - `location/` — Acesso à localização, *geofencing* e serviço em primeiro plano.
  - `repository/` — Repositórios da aplicação.
  - `service/` — Serviço para comunicação com a *API*.
  - `silence/` — Lógica de "Não Incomodar"(DND) e controlo do áudio do sistema.
  - `storage/` — Acesso a dados locais com *Room* (*DAOs* e entidades).
  - `ui/` — Ecrãs construídos com Jetpack Compose e lógica de interface:
    - `screens/` — Organizados por funcionalidade (ex: `homepage`, `map`).
    - `components/` — Composables reutilizáveis (ex: diálogos, botões, contentores).
  - `utils/` — Classes utilitárias (ex: permissões, constantes).
  - `workers/` — Agendamento de tarefas com *WorkManager*.
- `res/` — Recursos:
  - `drawable/`, `layout/`, `xml/`, `values/`, etc.
  - Inclui ícones do mapa, elementos gráficos vetoriais, temas e traduções.
- `AndroidManifest.xml` — Declara permissões, serviços, recetores e atividades.

2.3 `/jvm` - Aplicação *Backend*

O diretório `jvm` contém a implementação do *backend*, e tem a seguinte organização:

- `domain/` - Camada de domínio da aplicação.
- `host/` - Configuração da aplicação e infraestrutura.
- `http-api/` - Pontos de acesso e controladores da API HTTP.
- `http-pipeline/` - Pipeline de processamento de pedidos e respostas HTTP.
- `repository-jdbi/` - Implementações específicas de repositórios com JDBI.
- `repository/` - Abstrações da camada de acesso a dados.
- `service/` - Serviços da aplicação e lógica de negócio.

A documentação da *API* está disponível neste link.

2.4 `/scripts` - ficheiros de *script*

3 Implantação e Utilização

A implantação da API e a geração do ficheiro APK foram realizadas com sucesso utilizando *Docker*. O processo está devidamente documentado e inclui um guia completo para realizar o *deploy* local da aplicação através de contentores Docker, bem como instruções detalhadas de configuração e execução e credenciais de utilizadores já armazenados na base de dados. Esse guia está disponível em Guia de Instalação.

4 Recursos

- Proposta de Projeto
- Apresentação de Progresso
- Repositório GitHub
- Relatório
- Cartaz
- Documentação da *API*
- Diagrama do percurso do utilizador na aplicação (*User journey*)