RELATORIO PROJECTO

# Resumo do projecto

O projecto envolve a criação de uma aplicação full-stack para plataformas web e móveis, utilizando conceitos, metodologias e ferramentas modernas.

# Requisitos técnicos

A interface gráfica deve utilizar bibliotecas como Bootstrap ou Reactstrap, ou plataformas de componentes web como Bit, e implementar Responsive Web Design (RWD) para compatibilidade entre dispositivos.  
Deve ser dada preferência a etiquetas HTML semânticas, um design de menu consistente e um esquema de cores coeso.  
A estrutura e a navegação da aplicação devem ser intuitivas e bem planeadas.

Uma imagem promocional otimizada para redes sociais (Facebook e Instagram) e um vídeo de demonstração de 1 minuto, ambos incluindo créditos (nomes dos autores, curso, professor(es), logótipo do ISMT e logótipo do curso de Multimédia).  
Um relatório detalhado abordando as fases do projeto (definição do tema, modelação da base de dados, mapa de navegação, design, etc.), excertos de código, justificações das escolhas, uma página de capa com o título do projeto e detalhes dos alunos (incluindo um URL do GitHub), e secções para conclusões e bibliografia.

# Infra-estrutura técnica escolhida

Para hospedar a base de dados foi escolhido o site <https://freedb.tech> porque é completamente gratuito e disponibiliza serviço de base de dados MySQL/PhpMyAdmin.

Apesar de ter limitações no plano gratuito, as mesmas não têm impacto para este projecto.

A base de dados fica disponível no URL: http://sql.freedb.tech, porta 3306.

As credenciais de acesso são:

Utilizador: freedb\_programacao3

Palavra-chave: g2j5J$PhJ?\*EXqW

Base de dados: freedb\_programacao3

Todo o Código do projecto está disponível no Github, através de um projecto publico, em:

https://github.com/olaf1910/programacao3

# Sumario técnico da aplicação

There are three layers: the manager creates jobs (a single text description suffices) that are placed in a pool of unassigned jobs. The team leaders will consult the pool and based on the job description, availability of the programmers and the technical capabilities of each programmer of his team assigns the job for a specific programmer. The programmer in his turn checks his pool of assigned jobs and record the start and end of each job. A programmer cannot work in two jobs at once. A programmer must be able to assign skills in his profile. There must be security checks for each action: only managers can create update or delete jobs. Assigned jobs can’t be deleted or change its description. Only team leaders can assign jobs. Only programmers can record start and end time. Managers can only see jobs created by him self. Team leaders can see and assign jobs from all managers. Only the admin can create users and assign user role (manager, team leader, programmer)

# Modelo de base de dados

A base de dados do FeedzzTrab foi concebida para gerir utilizadores, os seus papéis, competências, tarefas e atribuições de tarefas de forma eficiente, suportando um sistema de gestão de tarefas para gerentes, líderes de equipa e programadores. O esquema é composto por seis tabelas:

* **Funcoes**: Define os papéis dos utilizadores (admin, gerente, lider\_equipa, programador) utilizando um ENUM para garantir consistência e eficiência. É referenciada pela tabela Utilizadores para impor o controlo de acesso baseado em papéis.
* **Utilizadores**: Armazena os dados dos utilizadores, associando cada utilizador a um papel através de funcao\_id. A restrição ON DELETE RESTRICT assegura que os papéis permaneçam se existirem utilizadores associados, preservando a integridade referencial.
* **Competencias**: Lista competências únicas (por exemplo, "JavaScript") que os utilizadores podem possuir, mantida simples para maior flexibilidade.
* **Competencias\_Utilizadores**: Uma tabela de junção que permite uma relação muitos-para-muitos entre Utilizadorese Competencias, com ON DELETE CASCADE para eliminar associações quando um utilizador ou competência é removido.
* **Tarefas**: Representa as tarefas com um criador (criado\_por) e um estado (estado) que monitoriza o progresso (nao\_atribuida, atribuida, em\_progresso, concluida). A restrição ON DELETE RESTRICT impede a eliminação de tarefas com atribuições associadas.
* **Tarefas\_Atribuicoes**: Regista as atribuições de tarefas, associando tarefas a atribuídos (atribuido\_a) e atribuidores (atribuido\_por), com campos opcionais inicio e fim para registo de tempo. Suporta múltiplas atribuições por tarefa, com ON DELETE RESTRICT para garantir a consistência dos dados.

O desenho utiliza a codificação utf8mb4 para suportar múltiplas línguas e inclui índices (por exemplo, idx\_nome em Utilizadores.nome\_utilizador) para melhorar o desempenho das consultas. As relações foram estruturadas para assegurar a integridade, permitindo ao mesmo tempo uma gestão flexível das tarefas, em conformidade com os requisitos da API FeedzzTrab.

A screenshot of a computer

AI-generated content may be incorrect.

#### 1. **Utilizadores**

Stores all users (managers, team leaders, programmers, and admin) with their roles and authentication details.

CREATE TABLE Utilizadores (

id INT AUTO\_INCREMENT PRIMARY KEY,

nome\_utilizador VARCHAR(50) NOT NULL UNIQUE,

password\_hash VARCHAR(255) NOT NULL,

email VARCHAR(100) NOT NULL UNIQUE,

funcao\_id INT NOT NULL,

criado\_em TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (funcao\_id) REFERENCES Funcoes(id) ON DELETE RESTRICT

)

CHARACTER SET utf8mb4

COLLATE utf8mb4\_unicode\_ci;

CREATE INDEX idx\_nome ON Utilizadores(nome\_utilizador);

#### 2. **Roles Table**

Defines the roles (admin, manager, team leader, programmer) to enforce security constraints.

CREATE TABLE Funcoes (

id INT AUTO\_INCREMENT PRIMARY KEY,

nome ENUM('admin', 'gerente', 'lider\_equipa', 'programador') NOT NULL UNIQUE

)

CHARACTER SET utf8mb4

COLLATE utf8mb4\_unicode\_ci;

#### 3. **Skills Table**

Allows programmers to assign skills to their profiles.

CREATE TABLE Competencias (

id INT AUTO\_INCREMENT PRIMARY KEY,

nome VARCHAR(50) NOT NULL UNIQUE

)

CHARACTER SET utf8mb4

COLLATE utf8mb4\_unicode\_ci;

#### 4. **User\_Skills Table**

A junction table to manage the many-to-many relationship between users and skills (programmers can have multiple skills).

CREATE TABLE Competencias\_Utilizadores (

utilizador\_id INT,

competencia\_id INT,

PRIMARY KEY (utilizador\_id, competencia\_id),

FOREIGN KEY (utilizador\_id) REFERENCES Utilizadores(id) ON DELETE CASCADE,

FOREIGN KEY (competencia\_id) REFERENCES Competencias(id) ON DELETE CASCADE

)

CHARACTER SET utf8mb4

COLLATE utf8mb4\_unicode\_ci;

#### 5. **Jobs Table**

Stores the pool of unassigned and assigned jobs with details and status.

CREATE TABLE Tarefas (

id INT AUTO\_INCREMENT PRIMARY KEY,

descricao TEXT NOT NULL,

criado\_por INT NOT NULL,

criado\_em TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

estado ENUM('nao\_atribuida', 'atribuida', 'em\_progresso', 'concluida') DEFAULT 'nao\_atribuida',

FOREIGN KEY (criado\_por) REFERENCES Utilizadores(id) ON DELETE RESTRICT

)

CHARACTER SET utf8mb4

COLLATE utf8mb4\_unicode\_ci;

#### 6. **Job\_Assignments Table**

Tracks the assignment of jobs to programmers, including start and end times, and ensures a programmer cannot work on two jobs simultaneously.

CREATE TABLE Tarefas\_Atribuicoes (

id INT AUTO\_INCREMENT PRIMARY KEY,

tarefa\_id INT NOT NULL,

atribuido\_a INT NOT NULL,

atribuido\_por INT NOT NULL,

inicio DATETIME,

fim DATETIME,

FOREIGN KEY (tarefa\_id) REFERENCES Tarefas(id) ON DELETE RESTRICT,

FOREIGN KEY (atribuido\_a) REFERENCES Utilizadores(id) ON DELETE RESTRICT,

FOREIGN KEY (atribuido\_por) REFERENCES Utilizadores(id) ON DELETE RESTRICT

)

CHARACTER SET utf8mb4

COLLATE utf8mb4\_unicode\_ci;

### Explanation of the Schema

* **Users and Roles**: The Users table links to Roles to define permissions. The admin creates users and assigns roles, while managers, team leaders, and programmers have restricted actions.
* **Jobs**: Managed by managers, with a status to track progression (unassigned, assigned, completed). Once assigned, jobs cannot be deleted or modified (enforced at the API level).
* **Job\_Assignments**: Tracks assignments, start/end times, and who assigned the job.
* **Skills**: Allows programmers to define their technical capabilities, which team leaders can use for assignment decisions.

### Key Considerations

* **Assignment Process**: Team leaders assign jobs without setting start\_time, leaving it to the programmer to initiate the job.
* **Security**: The schema supports role-based access, but the API must enforce that:
  + Managers can only create/update/delete unassigned jobs.
  + Assigned jobs cannot be deleted or have their description changed.
  + Only team leaders can insert into Job\_Assignments.
  + Only programmers can update start\_time and end\_time.
  + Managers see only their own jobs, team leaders see all jobs, and admin manages users.

# OpenAPI

A especificação OpenAPI 3.0 foi criada com base nos requisitos que definimos para a aplicação e com a estrutura de dados definida.

A API FeedzzTrab, definida na versão OpenAPI 3.0.3, é uma interface programática concebida para gerir atribuições de tarefas no sistema FeedzzTrab, destinada a gerentes, líderes de equipa e programadores. O seu objetivo é suportar a criação, atribuição e monitorização de tarefas num ambiente colaborativo, com controlo de acesso baseado em papéis (admin, gerente, líder de equipa, programador). A API opera em dois servidores: um local (http://localhost:3000/api) para desenvolvimento e outro de produção (https://feedzztrab.example.com/api), utilizando autenticação JWT via Bearer Token para segurança.

A especificação define cinco esquemas principais: Utilizador (dados de utilizadores), Competencia (lista de competências), Tarefa (detalhes das tarefas com estados como "nao\_atribuida" ou "em\_progresso"), Atribuicao (registo de atribuições de tarefas) e Erro (respostas de erro). Os endpoints incluem:

* **Gestão de Utilizadores**: Criação (POST /utilizadores, apenas admin), atualização de palavra-passe (PATCH /utilizadores/{id}/palavra\_passe) e adição de competências (POST /utilizadores/{id}/competencias, apenas programadores).
* **Gestão de Tarefas**: Criação (POST /tarefas, apenas gerentes), listagem (GET /tarefas, diferenciada por papel), atualização (PUT /tarefas/{id}) e eliminação (DELETE /tarefas/{id}, ambas apenas para tarefas não atribuídas por gerentes).
* **Gestão de Atribuições**: Atribuição de tarefas (POST /atribuicoes, apenas líderes de equipa) e atualização de horários (PATCH /atribuicoes/{id}, apenas programadores), com validação de conflitos (409).

A API privilegia respostas consistentes (e.g., 201 para criação, 403 para acesso proibido) e suporta uma estrutura modular que facilita a integração com uma interface gráfica futura, alinhando-se com os objetivos do projeto FeedzzTrab de desenvolvimento full-stack.

# API

O projeto FeedzzTrab é uma aplicação de gestão de tarefas desenhada para facilitar a coordenação de equipas de trabalho, desenvolvida como parte da unidade curricular Programação III do Instituto Superior Miguel Torga. Implementada em Node.js com o framework Express, esta API constitui o backend de uma solução full-stack que suporta a criação, atribuição e monitorização de tarefas num ambiente colaborativo, destinada a utilizadores com papéis distintos: administradores, gerentes, líderes de equipa e programadores.

A arquitetura da aplicação segue um modelo modular, dividindo a lógica em rotas específicas para utilizadores (utilizadores.js), competências (competencias.js), tarefas (tarefas.js) e atribuições (atribuicoes.js), acessíveis sob o prefixo /api. A autenticação é garantida por tokens JWT (JSON Web Tokens), enquanto o controlo de acesso baseado em papéis é implementado através de middlewares (autenticacao.js), assegurando que apenas utilizadores autorizados executem ações específicas (e.g., apenas gerentes criam tarefas). A base de dados MySQL armazena informações sobre utilizadores, competências, tarefas e suas atribuições, utilizando índices e restrições de integridade para otimizar consultas e garantir consistência.

As principais funcionalidades incluem:

* **Gestão de Utilizadores**: Criação de contas (restrita a administradores), autenticação via login e alteração de palavras-passe, com validação rigorosa de dados (e.g., palavras-passe seguras com regex).
* **Competências**: Adição de competências por programadores, com verificação de duplicados.
* **Tarefas**: Criação, listagem (filtrada por papel), atualização e eliminação de tarefas (estas últimas restritas a tarefas não atribuídas), geridas por gerentes.
* **Atribuições**: Atribuição de tarefas por líderes de equipa, com validação de conflitos, e atualização de horários por programadores.

O código utiliza boas práticas como validação de entradas, tratamento de erros detalhado (e.g., mensagens específicas para erros de base de dados como ER\_DUP\_ENTRY), e configuração via variáveis de ambiente (.env). Este backend foi projetado para integração futura com uma interface gráfica em React e React Native, cumprindo os requisitos de modularidade, segurança e escalabilidade definidos para o projeto prático de Programação III,