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.  
Os alunos devem dominar técnicas fundamentais no design, implementação e avaliação de aplicações de complexidade simples a moderada, incluindo modelação de informação.

# 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

# Modelo de base de dados

#### 1. **Users Table**

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

CREATE TABLE Users (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL UNIQUE,

password\_hash VARCHAR(255) NOT NULL,

email VARCHAR(100) NOT NULL UNIQUE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (role\_id) REFERENCES Roles(role\_id)

);

CREATE INDEX idx\_username ON Users(username);

#### 2. **Roles Table**

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

CREATE TABLE Roles (

role\_id INT AUTO\_INCREMENT PRIMARY KEY,

role\_name ENUM('admin', 'manager', 'team\_leader', 'programmer') NOT NULL UNIQUE

);

#### 3. **Skills Table**

Allows programmers to assign skills to their profiles.

CREATE TABLE Skills (

skill\_id INT AUTO\_INCREMENT PRIMARY KEY,

skill\_name VARCHAR(50) NOT NULL UNIQUE

);

#### 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 User\_Skills (

user\_id INT,

skill\_id INT,

PRIMARY KEY (user\_id, skill\_id),

FOREIGN KEY (user\_id) REFERENCES Users(user\_id) ON DELETE CASCADE,

FOREIGN KEY (skill\_id) REFERENCES Skills(skill\_id) ON DELETE CASCADE

);

#### 5. **Jobs Table**

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

CREATE TABLE Jobs (

job\_id INT AUTO\_INCREMENT PRIMARY KEY,

description TEXT NOT NULL,

created\_by INT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

status ENUM('unassigned', 'assigned', 'completed') DEFAULT 'unassigned',

FOREIGN KEY (created\_by) REFERENCES Users(user\_id) ON DELETE RESTRICT

);

#### 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 Job\_Assignments (

assignment\_id INT AUTO\_INCREMENT PRIMARY KEY,

job\_id INT NOT NULL,

assigned\_to INT NOT NULL,

assigned\_by INT NOT NULL,

start\_time DATETIME, -- Set by programmer when they start the job

end\_time DATETIME, -- Set by programmer when they complete the job

FOREIGN KEY (job\_id) REFERENCES Jobs(job\_id) ON DELETE RESTRICT,

FOREIGN KEY (assigned\_to) REFERENCES Users(user\_id) ON DELETE RESTRICT,

FOREIGN KEY (assigned\_by) REFERENCES Users(user\_id) ON DELETE RESTRICT,

CONSTRAINT chk\_no\_active\_job CHECK (

NOT EXISTS (

SELECT 1

FROM Job\_Assignments a2

WHERE a2.assigned\_to = Job\_Assignments.assigned\_to

AND a2.assignment\_id != Job\_Assignments.assignment\_id

AND a2.start\_time IS NOT NULL

AND a2.end\_time IS NULL

)

)

);

### 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.

# API