|  |
| --- |
| Manual de Instruções do Sistema de Apoio à Elaboração de Mapas de AvaliaçãoAplicação FULL STACK Engenharia Informática | 2024-2025| 3º ano  Qualidade de Software | Fátima Leal  Grupo 5 - Gabriel Proença | 47719, João Barbosa | 47713, Miguel Costa | 46997, Igor Oliveira | 47752, Gabriela Almeida | 47719 |

Conteúdo

[Manual de Instruções do Sistema de Apoio à Elaboração de Mapas de Avaliação 2](#_Toc185186643)

[Aplicação FULL STACK 2](#_Toc185186644)

[Manual de Instruções da Aplicação de Gestão do Mapa de Avaliações 4](#_Toc185186645)

[1. Introdução 4](#_Toc185186646)

[2. Descrição dos Requisitos da Aplicação 6](#_Toc185186647)

[Requisitos Funcionais: 6](#_Toc185186648)

[Requisitos Não Funcionais: 7](#_Toc185186649)

[3. Descrição das Tecnologias Utilizadas 7](#_Toc185186650)

[4. Instruções para Utilizadores 9](#_Toc185186651)

[4.1. Master (Gestor dos Diretores) 9](#_Toc185186652)

[4.2. Director (Diretor de Departamento) 9](#_Toc185186653)

[4.3. User (Coordenador de Curso) 9](#_Toc185186654)

# Manual de Instruções da Aplicação de Gestão do Mapa de Avaliações

## 1. Introdução

Esta aplicação foi desenvolvida para simplificar a gestão e organização do mapa de avaliações da Universidade Portucalense. A ferramenta permite que utilizadores com diferentes níveis de acesso (master, director e user) possam, cada um, fazer a sua devida função como vai ser explicado no ponto 4.

A aplicação permite a alocação de uma lista salas em cada avaliação, de forma inteligente, de acordo com as salas disponíveis, numero de alunos para alocar, etc.. Sendo possível, de acordo com as salas atuais da UPT, alocar + de 300 alunos que precisam de salas com computadores na mesma hora, e + de 3000 alunos que não precisam de computadores nessa mesma hora para avaliação, em salas diferentes, sem qualquer repetição.

Através dessa alocação, é possível, posteriormente, ser gerado o PDF(ou visto no próprio site) o mapa de cada semestre de cada CURSO, ou até mesmo, ver o mapa completo de avaliações daquele DEPARTAMENTO naquele semestre, sendo possível também, ver a lista de CURSOS, desse departamento, com geração também, de pdf.

A aplicação foi desenvolvida no Microsoft Visual Code. Para a sua execução serão necessárias as seguintes extensões no VSCode e o próprio MYSQL instalado:

* MYSQL
* Extension Pack for Java
* Spring Boot Extension Pack

Antes da primeira execução, será necessário ajustar a conexão no MYSQL e criar uma DB nova.

1. É necessário clicar nesse “+”, para ser criada uma nova conexão, onde a nossa host foi chamada de “localhost”, e deverá ser configurada de acordo com as informações do MYSQL do utilizador.
2. Uma imagem com texto, captura de ecrã, Tipo de letra, file

   Descrição gerada automaticamenteApós a criação da conexão, deverá ser criada uma DataBase, através da seguinte query que está disponibilizada no .zip do ficheiro com o nome de CREATE\_DB.txt.Uma imagem com texto, captura de ecrã, Tipo de letra, número

   Descrição gerada automaticamente
3. Após a criação da nova DB, é necessária a execução da aplicação, através do “UptWebApplication.java”, para serem criadas as devidas tabelas na DB, de acordo com as classes Entity.
4. Após isso, vão ser criadas as diversas tabelas necessárias, e vamos executar 2 querys, 1 para a inserção do primeiro master(necessário para tudo funcionar, a partir de aí, podemos criar os outros utilizadores [master cria director, director cria coordinator]), e a outra query para a inserção das ROOMS na db. A query da inserção de rooms vai estar disponibilizada em .txt no zip do projeto com o nome ADD\_ROOMS.txt. A inserção do user Master é feita através da query ADD\_MASTER.txt no .zip

Uma imagem com texto, captura de ecrã, software, Tipo de letra

Descrição gerada automaticamente

1. Após a criação do primeiro Master e a inserção das rooms, a aplicação já está pronta para uso. Todos os logins são feitos através do Username e devida Password do user pretendido.

## 2. Descrição dos Requisitos da Aplicação

### Requisitos Funcionais:

1. **Tipos de Utilizadores:**
   * **Master (Gestor dos Diretores):** Gere os diretores e define as salas da UPT para avaliações.
   * **Director (Diretor de Departamento):** Define os anos letivos, semestres e épocas de exames, além de criar utilizadores do tipo user e gerar mapas de avaliação.
   * **User (Coordenador de Curso):** Gere Unidades Curriculares (UCs), gere avaliações em cada UC e organiza os respetivos mapas de avaliações.
2. **Definições Globais:**
   * O director estabelece o início e fim dos semestres e define as épocas de exame (normal, época de exames, recurso e especial).
   * As UCs podem ter dois tipos de avaliação:
     + **Mista:** Com pelo menos dois momentos de avaliação, sendo um obrigatório na época de exames.
     + **Contínua:** Com um mínimo de três momentos de avaliação ao longo do semestre.
3. **Gestão de Avaliações:**
   * Os utilizadores podem criar avaliações com os seguintes tipos:
     + Teste
     + Teste final
     + Trabalho (individual ou em grupo, incluindo apresentação)
     + Monografia
     + Exercício prático individual
     + Pitch
     + Exame final
     + Prova oral
   * Cada avaliação inclui:
     + Tipo de avaliação
     + Peso da avaliação
     + Periodo que será inserida (época de exames, recurso, etc)
     + Data e hora de começo e fim
     + Sala ou indicação de realização no horário de aula
     + Se é necessário computador
     + Nota mínima
4. **Mapas de Avaliação:**
   * O user deve preencher as avaliações das UCs ao longo do semestre.
   * A aplicação assegura que:
     + Evita sobreposição de datas entre avaliações de diferentes anos.
     + Respeita um intervalo de 24 horas entre provas consecutivas do mesmo ano/semestre.
     + Todas as entregas de trabalhos têm data-limite de começo e fim (ex: 23h59).
     + Aloca salas automaticamente, considerando o número de alunos e requisitos específicos (ex.: uso de computadores) e salas disponives, além de apresentar sugestões de outros horários caso não seja um horário inteligente(de acordo com restantes UCS, de outros anos, ou salas ocupadas). Não permite avaliações fora dos intervalos de avaliação selecionados.

### Requisitos Não Funcionais:

1. **Usabilidade:** Interface intuitiva para facilitar a navegação.
2. **Confiabilidade:** Validação automática para reduzir erros de entrada de dados.
3. **Desempenho:** Respostas rápidas para operações de consulta e modificação.

### Validações

1. **Autenticação:**
   * **Login:** Verificação de credenciais (username e password) para autenticação.
   * **Sessão:** Verificação de tipo de usuário (MASTER, DIRECTOR, COORDINATOR) para acesso a funcionalidades específicas.
2. **Anos Acadêmicos:**
   * **Datas:** Validação de datas de início e fim dos semestres e períodos de exames.
   * **Sobreposição de Datas:** Verificação de sobreposição de períodos de semestres e exames.
3. **Coordenadores:**
   * **Campos Obrigatórios:** Nome, curso, duração do curso, username e password são obrigatórios.
   * **Username Único:** Verificação de unicidade do username.
4. **Salas:**
   * **Campos Obrigatórios:** Número da sala, designação, tipo de material, número de assentos e prédio são obrigatórios.
   * **Número da Sala Único:** Verificação de unicidade do número da sala.
5. **Avaliações:**
   * **Campos Obrigatórios:** Tipo de avaliação, peso, período de exame, data e hora de início e fim, nota mínima são obrigatórios.
   * **Peso da Avaliação:** Validação de que o peso está entre 0 e 100.
   * **Nota Mínima:** Validação de que a nota mínima está entre 0 e 20.
6. **Unidades Curriculares:**
   * **Campos Obrigatórios:** Nome da UC, número de estudantes, tipo de avaliação, número de avaliações e ano são obrigatórios.
   * **Número de Estudantes:** Validação de que o número de estudantes é maior que 0.
   * **E muitas outras…**

## 3. Descrição das Tecnologias Utilizadas

A aplicação foi desenvolvida utilizando as seguintes tecnologias:

 **Spring Boot**: Utilizado para estruturar a aplicação de forma simplificada e eficiente, sem necessidade de configurações complexas. O Spring Boot permite a criação de uma aplicação autónoma, com servidor embutido, facilitando o desenvolvimento do backend e a configuração das dependências.

 **Spring Data JPA**: Usado para simplificar o acesso a dados no banco de dados. Com o Spring Data JPA, é possível interagir com o banco sem escrever consultas SQL complexas, utilizando repositórios e a abstração JPA para persistir e recuperar dados de forma eficiente e intuitiva.

 **Spring MVC**: Implementado para separar as responsabilidades de controlo (backend) e visualização (frontend). Utilizando o padrão MVC (Model-View-Controller), o Spring MVC ajuda a estruturar as páginas web, tornando o código mais modular e fácil de manter. Este módulo foi utilizado para gerir o fluxo de dados entre o backend e o frontend.

 **Spring Boot Starter Thymeleaf**: Utilizado para criar páginas HTML dinâmicas no frontend. O Thymeleaf permite integrar dados do backend com as páginas web, permitindo que o conteúdo seja renderizado de forma dinâmica com base nas informações do servidor, como formulários para criação e edição de avaliações.

 **Spring Boot Starter Logging**: Usado para gerir o registo de eventos e mensagens de erro. A aplicação utiliza logs para monitorizar a execução e depurar problemas, facilitando a análise de falhas e o acompanhamento do comportamento do sistema.

 **Jakarta Validation**: Aplicado para validar dados inseridos pelos utilizadores. Por exemplo, ao registar ou editar avaliações, as validações de formato (como datas e campos obrigatórios) são realizadas para garantir que os dados estão no formato correto antes de serem processados ou guardados.

 **Hibernate Validator**: Implementação da Jakarta Validation que foi utilizada no projeto para aplicar regras de validação nos dados. Este componente garante que os dados inseridos nos formulários, como a data e o tipo de avaliação, atendam a restrições específicas definidas no código.

 **MySQL Connector**: Utilizado para estabelecer a conexão entre a aplicação Java e o banco de dados MySQL. Este driver é necessário para que a aplicação possa armazenar e recuperar dados de forma eficiente e segura, com a configuração da base de dados utilizando o Spring Data JPA.

 **H2 Database**: Utilizado para testes, o H2 é um banco de dados em memória que permite realizar testes rápidos e leves sem necessidade de configurar um banco de dados real. Isto facilita o desenvolvimento de testes de integração e a validação de funcionalidades sem a complexidade de um banco de dados completo.

 **JUnit**: Framework de testes utilizado para garantir que as funcionalidades da aplicação estão a funcionar corretamente. Os testes unitários são usados para verificar se as classes e métodos individuais estão a comportar-se como esperado.

 **Mockito**: Utilizado para criar objetos mock em testes unitários, permitindo isolar partes da aplicação para testar funcionalidades específicas sem depender de recursos externos, como o banco de dados. No projeto, foi usado para testar a lógica de negócios sem afetar a persistência de dados.

 **SLF4J**: Usado como a interface de logging para a aplicação. Ele fornece uma camada de abstração sobre diferentes bibliotecas de logging, facilitando a troca de implementações de logs sem alterar o código da aplicação. No projeto, foi utilizado para registar mensagens de erro e comportamento da aplicação.

 **Logback**: Implementação do SLF4J, utilizada para o registo de logs de eventos. O Logback oferece funcionalidades avançadas de registo, como a formatação de mensagens e a gravação em arquivos de log, permitindo um controlo mais eficiente e detalhado dos eventos na aplicação.

 **iText PDF**: Biblioteca utilizada para gerar documentos PDF dentro da aplicação. No contexto do projeto, ela foi utilizada para gerar mapas de avaliações e listas de UCs, permitindo a exportação de informações da aplicação para um formato acessível e portátil.

 **Java 17**: A versão do Java utilizada para desenvolver o backend da aplicação. O Java 17 oferece melhorias de desempenho e novos recursos de linguagem que foram aproveitados no desenvolvimento da aplicação, como melhor suporte para o gerenciamento de memória e novos recursos de linguagem para simplificar o código.

## Instruções para Utilizadores

**4.1.** Master **(Gestor dos Diretores)**  
**Funções Disponíveis:**

* Gerir diretores.
* Gerir as salas da UPT.

**Passos Principais:**

1. Aceder com credenciais de master.
2. Realizar as alterações necessárias e salvar.
   * Verifique se todos os campos obrigatórios estão preenchidos corretamente.
   * Certifique-se de que as salas têm o número de assentos e materiais adequados.

**4.2.** Director **(Diretor de Departamento)**  
**Funções Disponíveis:**

* Criar utilizadores do tipo user.
* Definir anos letivos, semestres e épocas de exames.
* Gerar mapas de avaliação organizados por curso, ano e semestre.

**Passos Principais:**

1. Fazer login como director.
2. Configurar as definições iniciais (anos letivos, semestres, épocas).
   * Certifique-se de que as datas de início e fim dos semestres e períodos de exame não se sobrepõem.
3. Gerir os utilizadores e gerar mapas através da página correspondente.
   * Valide que não há conflitos nas avaliações, como sobreposição de datas e horários.

**4.3.** User **(Coordenador de Curso)**  
**Funções Disponíveis:**

* Criar Unidades Curriculares (UCs) e avaliações.
* Organizar mapas de avaliações e alocar salas automaticamente.
* Preencher informações detalhadas sobre cada avaliação.

**Passos Principais:**

1. Fazer login como user.
2. Navegar até à página de gestão de UCs.
3. Criar ou editar avaliações, assegurando que as informações são inseridas corretamente.
   * Valide que o peso da avaliação está dentro do intervalo permitido (0-100).
   * Certifique-se de que as notas mínimas estão dentro do intervalo correto (0-20).
4. Gerar o mapa final de avaliações do curso.
   * Verifique se a alocação de salas foi feita corretamente e que as avaliações não se sobrepõem.

## Anexos





upt.pt