ThesisMan

João Santos 56380
João Costa 58226
Rafael Ferreira 57544

Introdução

Neste relatório apresentamos a nossa implementação para o projeto **ThesisMan**, sistema de gestão de Teses de Mestrado. Nesta última fase, utilizamos uma **interface web com renderização server-side** para lidar com os utilizadores empresariais/docentes e a framework JavaFX para interface gráfica dos alunos através de uma API REST. Tal como na 1ª fase, concretizando a base de dados relacional usando JPA com Spring Data para persistir os objetos.

Casos de uso

Pretendemos suportar os **casos de uso** referidos no enunciado, aqui repetidos por conveniência:

A. Login com autenticação da universidade (Nota: vamos fazer mock desta funcionalidade e qualquer palavra passe será aceite)
B. Registo de utilizadores empresariais
C. Login de utilizadores empresariais
D. Submissão de temas por parte dos docentes
E. Submissão de temas por parte dos utilizadores empresariais
F. Listar os temas disponíveis neste ano lectivo, por parte dos alunos
G. Candidatura a um tema (limite de 5), por parte dos alunos
H. Cancelamento da candidatura a um tema
I. Atribuição dos temas aos alunos (da parte do administrador)
J. Submissão do documento de proposta de tese, por parte dos alunos
K. Marcação da Defesa da proposta de tese, por parte do orientador da tese
L. Registo da nota da defesa da proposta de tese, por parte do orientador da tese
M. Submissão do documento final de tese, por parte dos alunos
N. Marcação da Defesa de tese, por parte do orientador da tese, incluindo a nomeação do júri.
O. Registo da nota da defesa da proposta de tese, por parte do presidente do júri.
P. Recolha de estatísticas sobre taxa de sucesso dos alunos.

Requisitos não Funcionais

Pretendemos cumprir os requisitos não funcionais referidos no enunciado, aqui repetidos por conveniência:

• A autenticação deverá ser feita com o sistema de autenticação da FCUL.
• Toda a informação deverá ser armazenada numa base de dados relacional.
• A plataforma deverá ser implementada em Spring Boot, de forma a ter um custo de desenvolvimento baixo, e a usar a linguagem Java, que é a linguagem para a qual é mais fácil de contratar engenheiros.
• A camada de dados deverá usar JPA e Spring Data.
• A aplicação deverá lidar com pedidos concorrentes, sem criar inconsistências.
• A camada de negócios deverá usar o Domain Model, com meta-dados suficientes para que o JPA faça o mapeamento para a base de dados.
• A interface deverá ser através da web, num momento inicial.
• Deverá ser exposta uma API REST que os clientes (web ou mobile) poderão usar para interagir com a aplicação.
• O repositório deverá aceitar apenas código com o nível de qualidade aceite pela equipa (usando o pre-commit).
• O projecto deverá correr dentro de um docker, que será a forma como vai ser deployed no servidor de produção.
• O controlo da participação de cada membro da equipa será feita através da actividade no repositório git.
• As decisões técnicas deverão estar justificadas num relatório técnico.

Interface web

Na interface web temos 3 grandes partes: login, dashboard e ações. Quando acedemos ao website temos a opção de login e de registo (para consultores/orientadores externos).

O login é feito a partir do WebSecurity, que faz a codificação das passwords para serem guardadas na BD e implementa uma função que permite a certos URLs o acesso público ao controlador (como por exemplo o caminho /api/**), todos os outros terão de se autenticar para ter acesso aos serviços.

Depois do login entramos no ecrã principal onde temos botões para submeter um tópico de tese, marcar defesas e atribuir as respectivas notas. Posteriormente podemos ver algumas estatísticas sobre a taxa de sucesso dos alunos.

Quando um professor administrador acede ao website, existe um botão para este atribuir tópicos não associados a estudantes.

No botão “teses que eu oriento” aparecem 4 tabelas. Na primeira aparecem todas as teses que um determinado professor orienta, para agendar a sua defesa. Na segunda aparecem todas as “primeiras” defesas para este dar a nota. Na terceira estão todas as teses que vão para defesa final. Na quarta estão todas as defesas finais para atribuir a sua nota. No entanto, só ao presidente do júri da defesa de tese é que irá aparecer o botão para submeter a nota da defesa final.
Esta nota será objeto de uma verificação (se está entre 0 e 20) quando for submetida.

JavaFX

A aplicação **JavaFX** permite aos estudantes visualizar os tópicos de tese disponíveis e realizar candidaturas aos mesmos (até um máximo de 5). Cada aluno apenas tem acesso aos tópicos de tese disponíveis para alunos do seu mestrado. Permite também que os alunos visualizem as suas execuções de tese, submetam os documentos das suas teses (propostas e final) e consultem as suas defesas.

Seguimos o padrão **MVC**. As views da aplicação foram definidas em ficheiros **FXML**, com exceção de um overlay que é criado em runtime para cobrir o ecrã inteiro. A aplicação tem 1 stage principal, onde são mostradas 2 scenes: Login, e Main. O Login apenas serve para autenticar o utilizador com o seu ID e carregar a Main. A Main tem uma barra de botões fixos, que permitem ao utilizador selecionar que conteúdo ver e fazer logout. Ao clicar nestes botões, o conteúdo é “fetched” da API REST do servidor em formato JSON, que é “parsed”. Posteriormente é mostrada numa lista. Para além desta, o utilizador tem um painel que lhe permite ver informações e interagir com o item selecionado na lista. As listagens do conteúdo são feitas a partir de um único **FXML** e controlador genérico, que permitem mostrar (e selecionar) qualquer modelo de dados. Para os paineis, temos um **FXML** e controlador para cada um, já que tem funcionalidades e dados diferentes.

Ao submeter/cancelar candidaturas e submeter documentos, o sistema mostra um pop-up para confirmar a ação. Estes popups são mostrados através de um Stage, criado e destruído após o término da mensagem.

API Rest

A **API Rest** foi feita para acesso através do **JavaFX**. Funciona a base do ID do utilizador que está a aceder à aplicação. Como pedido no enunciado, fizemos mock à autenticação: ao fazer login, o cliente envia à API o username do utilizador. Caso ele esteja registado, a API retorna o seu ID. Caso contrário cria-se um novo user e é-lhe atribuído um mestrado. Para o processamento das interacções é retornado o ID desse novo utilizador. Fizemos isto para facilitar o teste do sistema. A partir daí, os pedidos feitos à **API** são sempre acompanhados pelo ID deste utilizador. Assim, torna-se fácil retornar os tópicos / candidaturas / execuções / defesas a que o utilizador tem acesso.

Na criação de candidaturas e submissão de ficheiros, o servidor verifica se são cumpridos os requisitos (max. 5 candidaturas, 1 documento final apenas).

Templates

Os templates foram feitos para ligar a interface gráfica do browser ao backend do **JPA**. Utilizando um **WebController** que faz o mapeamento dos endpoints até ao utilizador.
Para fazer qualquer operação o utilizador acede a um URL fazendo um pedido **REST** que vai ser processado pelo controller. Este adiciona ao model os atributos necessários e retorna a página HTML correspondente ao pedido do utilizador. Dependendo do tipo de pedido estes métodos podem chamar funções dos repositórios para fazer operações na BD.
Utilizamos vários ficheiros HTML para fazer a interface, e ir navegando entre elas conforme acedemos aos serviços do website.
Nesta parte foi utilizado a framework **TailwindCSS** para fornecer estilos. Facilita a leitura dos dados e torna a aplicação do browser mais atrativa e agradável ao utilizador.

Como correr o Projeto

Estando na raiz do **thesisman**:
mvn clean install

Para compilar o **Server**:
$cd server
$bash run.sh

Para compilar **Client** (O Server tem de estar correr):
$cd client
$mvn clean javafx:run

Limitações do trabalho

No ficheiro application.properties, partimos do princípio de que *spring.jpa.hibernate.ddl-auto=create-drop*, este seria o comando predefinido. Correndo o risco da BD entrar em loop.

Quando o presidente do júri atribui a nota a defesa final de uma tese, não há verificação de que o valor está entre 0 e 20. Esta apenas é feita na nota da primeira defesa.