Relatório de Arquitetura do Projeto (CSS - Fase 2)

GRUPO: fc58176 | fc58195 | fc58236

1. Descrição da Arquitetura do Projeto

1.1. Visão Geral
O diagrama geral da nossa aplicação baseia-se numa arquitetura de Cliente/Servidor, com acesso a Base de Dados e uso de variados Serviços para gestão de chamadas externas, com uma REST API que trata da ligação entre o Cliente e Servidor. Este projeto segue uma arquitetura por camadas (4th Tier Application), onde o Cliente executa uma aplicação JavaFX para login de Alunos e o Servidor usa uma Web Application para o resto dos utilizadores, que podem ser Docentes, Empresas e seus Empresários e Administrador.

2. Escolha e Justificação das Decisões Técnicas na Arquitetura da Aplicação
- A arquitetura da aplicação segue uma abordagem de 4 tiers, onde cada camada desempenha um papel específico na estrutura e funcionalidade do sistema. O uso do Spring Framework junto com Maven facilita a configuração e gerenciamento de dependências, garantindo uma base sólida para o desenvolvimento.
- A implementação de uma REST API permite uma comunicação eficiente entre o cliente JavaFX e o servidor, promovendo escalabilidade e interoperabilidade. O Domain Model, composto essencialmente por Handlers, Services, Repositories e DTOs, encapsula a lógica de negócios de forma modular e coesa, enquanto os Controllers atuam como ponto de entrada para as requisições HTTP, coordenando a interação entre as camadas.
- Decidimos organizar o código por casos de uso, tendo tido como base parte do desenho desenvolvido na fase 1 do projeto, mas com algumas reformulações por necessidade de mudança e alguma falta de tempo.
- No geral, e omitindo alguns passos intermediários, a lógica de aplicação foi desenvolvida com base no seguinte:
Controllers -> Services -> Handlers -> Repositories

3. Escolha e Justificação das Decisões Técnicas no Desenho da Interface Web
- Decidimos seguir um desenho simplista e que permitisse executar todos os casos de uso requeridos. Um exemplo claro desta divisão é a própria página do menu, em que cada botão leva o utilizador, se tiver acesso (faz-se esse check através das tokens), a um certo caso de uso, tal como atribuir um tema ao aluno, por parte do Administrador, entre outros.
- Separação de Preocupações: A arquitetura MVC (Model-View-Controller) separa a lógica de negócio (Model), a lógica de apresentação (View) e a lógica de controle (Controller), facilitando a manutenção e escalabilidade da aplicação.
- Thymeleaf foi escolhido como o motor de templates por ser bem integrado com o Spring e facilitar a criação de views dinâmicas.
- Mensagens de erro e sucesso são fornecidas ao utilizador para informar sobre o status das operações realizadas (ex.: login, registo, submissão de temas).
- Uso de sessões (@SessionAttributes("token")) para gerir autenticação e autorização, garantindo que apenas utilizadores autorizados podem aceder a determinadas funcionalidades.
- Validação de dados nos controladores para prevenir entradas inválidas e garantir a

integridade dos dados (ex.: NumberFormatException para entradas numéricas).
- A utilização de anotações simplifica a configuração e a definição de rotas, como @Controller, @RequestMapping, @PostMapping, etc.
- Uso de redirecionamentos (redirect:/) para manter a consistência na navegação e proporcionar uma experiência de usuário suave.
- Utilização de serviços especializados (TemaService, TeseService, UtilizadorService, EstatisticaService) para encapsular a lógica de negócio, promovendo a coesão e a reutilização de código.

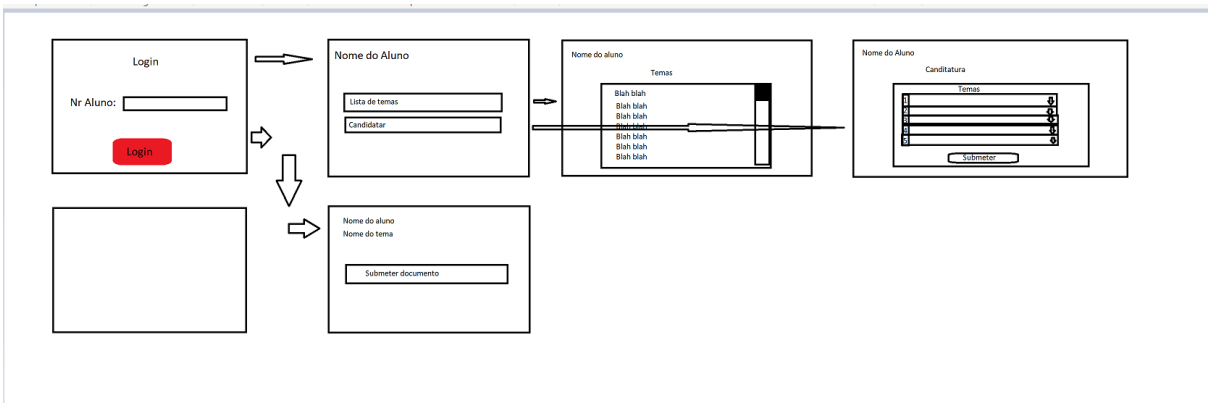
4. Escolha e Justificação das Decisões Técnicas no Desenho da API REST
- O uso do Spring Framework, em particular o Spring Boot, simplifica a configuração e inicialização da aplicação REST, proporcionando um desenvolvimento mais ágil.
- O uso de anotações como @RestController, @GetMapping, @PostMapping, e @DeleteMapping simplifica o mapeamento de endpoints HTTP para métodos Java, reduzindo a complexidade e o código boilerplate.
- Os endpoints da API estão organizados por funcionalidade, o que facilita a compreensão e manutenção do código, seguindo as melhores práticas de design de API.
- A API implementa tratamento de exceções para lidar com cenários inesperados, como alunos não encontrados ou excedendo o número máximo de temas, garantindo uma resposta adequada e consistente ao cliente.

5. Escolha e Justificação das Decisões Técnicas no Desenho da Interface JavaFX
- Decidimos seguir um UI muito simplista e minimalista que permitisse executar todos os casos de uso pedidos, que com a ajuda do SceneBuilder, procurámos construir uma interface coerente e responsiva que fornecesse feedback ao utilizador a cada ação. Além disso, tentamos usar cores de forma a dar '*visual cues*' ao utilizador.
- Simplicidade, consistência e feedback foram os princípios que mais considerámos no design desta interface.
- Com o uso de HttpUrlConnection, conseguimos facilmente estabelecer pedidos rest, para o Rest Api disponibilizado
- Usamos as anotações do JavaFx como o FXMLLoader, que nos permite construir as scene que usamos para mostrar ao nosso utilizador

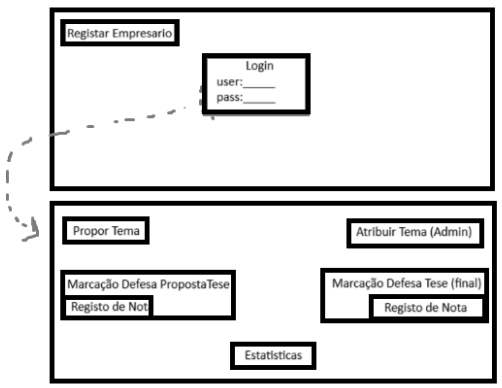
6. Decisões Implementadas
- Usamos Horário como @Embeddable ao invés de @Entity (apenas faz sentido quando falamos no contexto de Defesa; não cria BD)
- Orientador Interno é o Docente da Tese (não colocamos no Júri, desnecessário)
- Não vamos guardar o orientador da tese por questões de simplificação e falta de tempo
- Assumimos que só duas defesas colidem se os horários, para o caso de ser presencial (com sala), tiverem o início e o fim exatamente iguais (falta de tempo, simplificação)
- Assumir que as defesas assim que têm um horário poderão ser logo avaliadas (num contexto real, apenas poderiam ser avaliadas após a data atual ter ultrapassado o horário agendado da defesa)
- Não foi tratada a exceção de meter um número de presidente ou de arguente que não existe, no caso de uso de marcar defesa
- Assumimos que só damos init 1 vez! (não damos clear de nenhum dado automaticamente, por questões de segurança)
- Convém não meter o mesmo número entre utilizadores (falta de tempo, simplicidade)

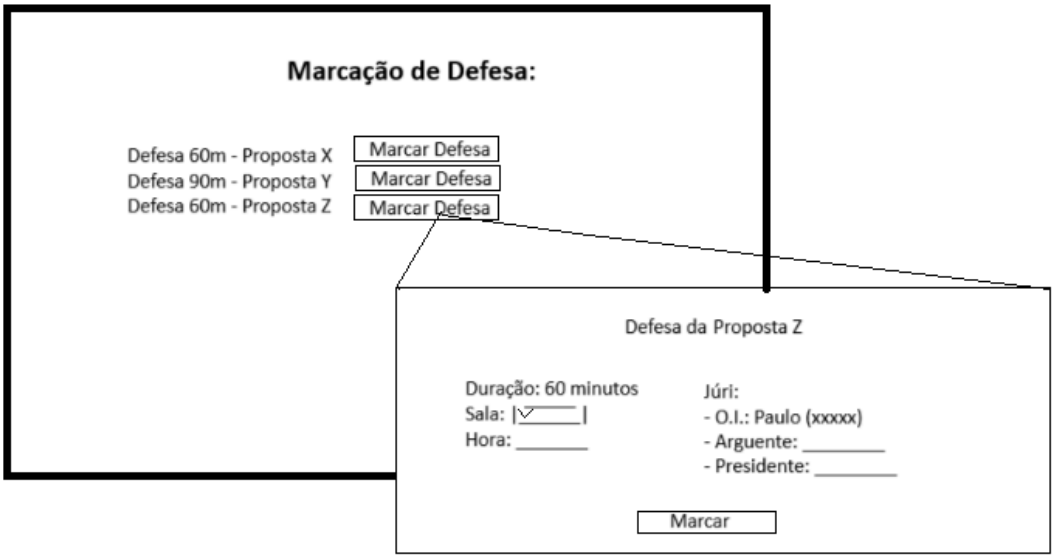
- Decidimos não incluir o orientador da tese devido à simplificação e à falta de tempo.
- Consideramos que duas defesas só colidem se os horários de início e fim forem exatamente iguais, para simplificar o processo.
- Assumimos que as defesas podem ser avaliadas assim que são agendadas, embora, em um contexto real, só poderiam ser avaliadas após a data da defesa.
- Aceitamos a possibilidade de inserir um número inexistente para presidente ou arguente no agendamento da defesa, como uma exceção.
- Presumimos que a inicialização do sistema ocorre apenas uma vez.
- Evitamos a duplicação de números entre usuários para manter a simplicidade e economizar tempo.
- No caso de uso I, optamos por simplificar e deixar a decisão do tema de cada aluno para o administrador, em vez de iterar sobre temas e atribuí-los automaticamente com base na maior nota.
- Assumimos que a atribuição final dos temas aos alunos é sempre feita pelo administrador, pois não há um caso de uso que exija a implementação da atribuição automática por ordem decrescente de notas, o que também simplifica o processo.
- Estatísticas não sofrem alteração com as notas atribuídas às defesas dos alunos (falta de tempo)
- **Mocks:** 3x Aluno, 1x Administrador, 2x Docente, 1x Empresa, 4x Sala, 2x Tema

7. Sketches



Mocks:
1x Aluno
1x Administrador
3x Docente (Orientador Interno + Arguente + Presidente)
1x Empresa
Registro de Empresario:
Empresa <--> Empresario
Propostas de Tema:
Empresario <--> Temas
Docente (O.I.) <--> Temas [assumindo que um dos seus temas é escolhido pelo aluno]
{JavaFX: Aluno --> 5 Temas --> Escolher o primeiro automaticamente --> Tese --> Tema --> O.I. ou O.E.}
Atribuição Manual de Temas:
-> Administrador
Marcação Defesa Proposta de Tese:
-> Docente (O.I.)
-> Empresario (O.E.)
Registo da Nota da Defesa Proposta de Tese:
-> Docente (O.I.)
-> Empresario (O.E.)
Marcação Defesa Tese (final) e nomeação do Júri:
-> Docente (O.I.)
-> Empresario (O.E.)
Registo da Nota da Defesa Tese:
-> Docente (Presidente)
Estatísticas:
**todos*




Marcação de Defesa:
Defesa 60m - Proposta X
Defesa 90m - Proposta Y
Defesa 60m - Proposta Z
Marcar Defesa
Marcar Defesa
Marcar Defesa
Defesa da Proposta Z
Duração: 60 minutos
Sala:
Hora: _____
Júri:
- O.I.: Paulo (xxxxx)
- Arguente: _____
- Presidente: _____
Marcar