Programação Avançada em Java
Trabalho Prático 3

Introdução

Objetivo: Neste projeto, pretende-se que os alunos façam uma extensão da aplicação implementada no projeto 2, utilizando persistência de dados do lado do servidor através de uma base de dados *MySQL*. Deve usar JPA e criar endpoints REST para a gestão dos dados. Devem ser capazes de recorrer a tecnologias como JAX-RS e EJBs para desenvolver RESTful Web Services. Os alunos deverão ainda utilizar o sistema de controlo de versões *Git*, bem como a ferramenta de *build Maven.*

Data de Entrega: 3 de Março de 2023

Grupos: o projeto é realizado em grupos que são definidos pelos docentes e são compostos por **dois** elementos.

Arquitetura do Sistema a Desenvolver

Deve ser seguida a arquitetura apresentada na Figura 1, cujos módulos são explicados a seguir.
O módulo **Backend** corresponde a **um** projeto `Maven` que interage com uma base de dados `MySQL` usando `Java Persistence API` com `hibernate`. Os alunos **devem desenhar o modelo de dados** (identificar as tabelas e seus campos, tipos de dados de cada campo, chaves primárias, se um campo é *nullable* ou *unique*, etc.) **antes de começar a fase de implementação**. É possível escolher entre Criteria API e JPQL para fazer consultas dos dados.
O *Backend* deve expor uma API REST (completar a API implementada no projeto 2) que poderá ser consumida por outro sistema, neste caso por endpoints do Postman. **O formato dos dados a serem trocados deve ser `JSON`**.

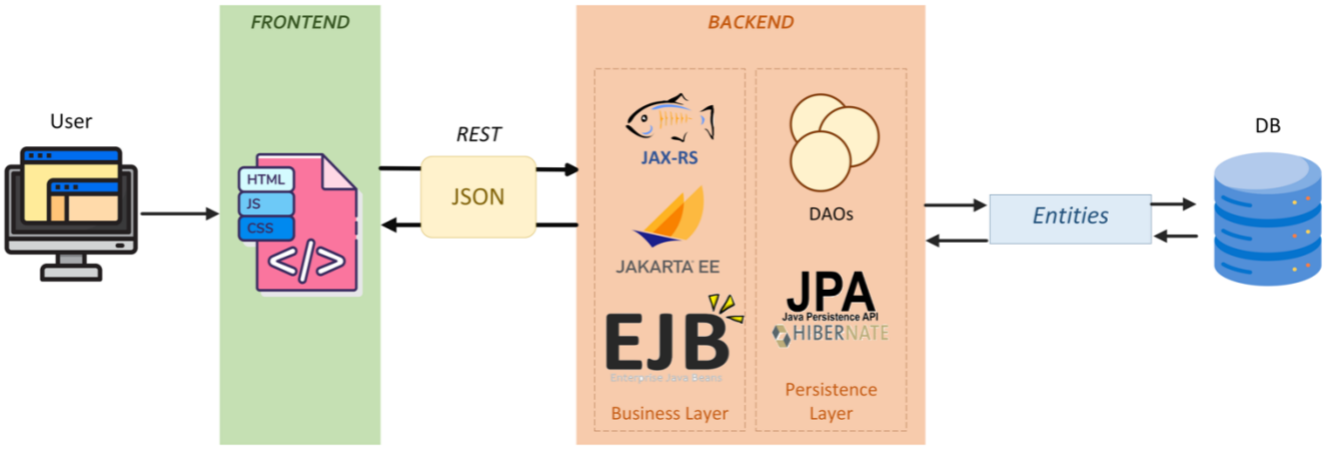

Figura 1. Arquitetura do sistema a implementar.

Especificação

Devemos ter três tipos de utilizadores no sistema, *Product Owner, Scrum Master* e *Developer*. Assim, deve ser possível fazer gestão dos utilizadores, dos perfis pessoais e das tarefas. Os alunos devem criar manualmente um *Product Owner* na base de dados para ser o administrador principal do quadro *Scrum*. Deve haver autenticação e verificação do utilizador para cada pedido, mas neste projeto devem usar um *token* em vez de enviar o *username* e *password* nos pedidos. O servidor gera um *token* único após cada login bem-sucedido. O *token* é usado para provar a identidade de um determinado utilizador durante uma sessão. O objetivo de um *token* é gerar uma senha de uso único (One-Time Password (OTP)) que será validada pelo servidor. **Atenção:** a senha deve ser armazenada na base de dados, numa forma encriptada.

Para cada utilizador, o sistema deve guardar os seguintes dados de forma persistente: primeiro nome, último nome, *username*, *password*, email, número de telemóvel, uma fotografia (pode ser a URL), e também o *token* resultante do processo de autenticação. Para cada tarefa, o sistema deve guardar os seguintes dados de forma persistente: título, descrição, data de início, data de fim, estado (*To Do, Doing, Done*), categoria da tarefa, prioridade (alta, média e baixa), e criador. Por cada categoria de tarefa, basta guardar um título (nome da categoria). Os alunos são livres para adicionar mais campos, se necessário.

Todos os utilizadores podem criar as tarefas. Cada tarefa deve pertencer a uma categoria (exemplo: *design, backend, frontend, test*). As categorias de tarefa podem ser criadas apenas pelo *Product Owner*. **É essencial que o sistema apenas permita realizar funcionalidades compatíveis com o tipo do utilizador**. As funcionalidades por tipo de utilizador são listadas a seguir.

O *Developer* poderá aceder às seguintes funcionalidades:

- **D1.** Alterar o seu perfil e imagem;
- **D2.** Adicionar tarefa;
- **D3.** Editar tarefa de sua autoria;

D4. Alterar estado de qualquer tarefa.

O *Scrum Master* herda todas as funcionalidades do *Developer* e também poderá aceder às seguintes funcionalidades:

S1. Editar tarefa de outros utilizadores;
S2. Apagar tarefas (suas e de outros utilizadores);
S3. Consultar o perfil de um utilizador específico;
S4. Filtrar lista de tarefas criadas por um utilizador;
S5. Filtrar lista de tarefas por categoria;
S6. Consultar tarefas apagadas.

O *Product Owner* herda todas as funcionalidades do *Developer* e *Scrum Master* e também poderá aceder às seguintes funcionalidades:

P1. Alterar a informação do perfil de outros utilizadores;
P2. Adicionar um novo utilizador;
P3. Consultar lista de todos os utilizadores (*Developer*, *Scrum Maste* ou *Product Owner*);
P4. Apagar um utilizador (apagar não implica apagar os dados permanentemente do sistema, mas apenas alterar o estado dos dados para que não sejam visíveis);
P5. Apagar todas as tarefas criadas por um utilizador;
P6. Alterar a role de um utilizador (*developer/scrum master/product owner*);
P7. Deletar permanentemente tarefas e utilizadores;
P8. Adicionar categoria de tarefa;
P9. Editar categoria de tarefas (editar o nome da categoria, ex: *frontend* para *interface*);
P10. Apagar categoria de tarefa (apenas quando não há tarefa criada com essa categoria).

Nota: apagar as tarefas e utilizadores não deve resultar na sua remoção permanente da base de dados. Dessa forma, deve ser possível consultar listas das atividades apagadas na página web. Deve ser possível restaurar as atividades apagadas. No entanto, também deve ser possível apagar as atividades de forma permanente da lista das atividades apagadas. Ao apagar permanentemente um utilizador que tenha tarefas, as tarefas devem identificar que o criador foi excluído.

Testes

Devem **criar endpoints de teste no postman** para responder aos vários requisitos, e também **exportar e colocar no git** esses testes para verificação depois na defesa.
Finalmente, os alunos devem criar ficheiros de testes que devem utilizar para verificar a execução correta das funções implementadas. Para isso devem usar uma **framework de testes JUnit**[1] e devem fazer os testes que acharem necessários de modo a perceber se as vossas funções fazem o que é necessário. Os testes que necessitem da criação, alteração ou remoção de informação **devem usar a framework Mockito**[2] de forma a não alterarem os dados presentes na Base de Dados.

Unit testing - Pelo menos 10 testes para a função principal usando JUnit

[1] https://www.baeldung.com/junit-5 & https://howtoprogram.xyz/2016/09/09/junit-5-maven-example/
[2] http://www.vogella.com/tutorials/Mockito/article.html

API testing - Pelo menos um teste com POSTMAM para cada endpoint
Java script testing - Pelo menos mais 5 testes para a função principal usando Jest (diferentes dos testes do Projeto2)

Os valores indicados são o mínimo para aprender a metodologia. Os grupos podem e devem implementar tantos testes quanto necessários para garantir a qualidade da solução.

Grupo de 3

O Grupo de 3 é responsável por implementar todas as funcionalidades relacionadas à retrospectiva. Os seguintes dados devem ser guardados de forma persistente para cada evento de retrospectiva: data, membros, comentários - texto do comentário e categoria do comentário (Pontos Positivos, Desafios, Sugestões de Melhoria). **Importante**: apenas membros adicionados na retrospectiva podem comentar.

Assim, estabelecemos as seguintes funcionalidades (e tipo de utilizador):

- **G1.** Adicionar comentários na retrospectiva (*Developer*, *Scrum Master* e *Product Owner*);
- **G2.** Editar o próprio comentário na retrospectiva (*Developer*, *Scrum Master* e *Product Owner*);
- **G3.** Criar um evento de retrospectiva (*Scrum Master* e *Product Owner*);
- **G4.** Deletar permanentemente comentários em uma retrospectiva (*Scrum Master* e *Product Owner*);
- **G5.** Adicionar membro na retrospectiva (*Scrum Master* e *Product Owner*).

Em relação aos testes, além dos informados previamente, devem implementar mais testes. Devem implementar testes no **postman** para as funcionalidades da retrospectiva e implementar 3 testes utilizando **JUNIT** e 2 testes usando **Jest**.

Serviço REST

Os alunos são livres de adicionarem **novos** endpoints, se assim o desejarem. No entanto, devem procurar seguir as boas práticas de interfaces RESTful e procurar usar *status codes* adequados em todas as respostas produzidas. Tal como já foi referido anteriormente, todas as mensagens (tanto pedidos como respostas) REST devem usar o formato JSON.
A autenticação deve verificar não só se as credenciais de acesso (i.e., *password* e *username)* estão corretas, mas também se a operação a ser feita (se for uma operação que envolva mudança de estado) afeta apenas dados relativos ao utilizador autenticado. Por exemplo, um utilizador desenvolver (*developer*) não deve conseguir alterar o perfil de outro utilizador. Caso a autenticação falhe, a resposta deve retornar o *status code* 403 (*Forbidden*). Se os dados necessários para a autenticação não tiverem sido enviados (por exemplo, *token* no Header) a resposta deve retornar o *status code* 401 (*Unauthorized*).

Entrega do projeto e avaliação

Entregas: os projetos devem ser guardados num **repositório *Git***. Só será considerado para avaliação o código do **último *commit*** feito na data indicada na introdução deste enunciado.

A avaliação é feita através de um conjunto de critérios que correspondem aos requisitos definidos para o projeto 3. Os alunos devem colocar a tabela de requisitos preenchida por quem fez/tomou responsabilidade (num README.md, Google sheets, ou Excel) no repositório *Git*.

Nota importante: cada elemento de grupo deve obrigatoriamente estar envolvido em pelo menos uma tarefa de cada grupo de requisitos (assinaladas com cores diferentes).

Podem consultar a grelha na figura da página seguinte.

Defesa: As defesas terão uma duração de **45 minutos**, sendo necessário uma inscrição prévia na plataforma Inforestudante, na página da disciplina. Durante a defesa, todo o projeto deve estar a funcionar e deve ser executado sem erros. Se durante a defesa o código apresentar diferenças face ao entregue, será **descontado 25%** da nota. Também é importante garantir que os alunos cheguem à defesa preparados (i.e., código aberto no IDE, web browser com a aplicação pronta a mostrar). Caso contrário será aplicada uma **penalização de 10%**.

Nota importante: qualquer componente implementada mas não demonstrada na defesa recebe **35% de penalização**.

Catergoria	Requisito	Grupos
Interface Funcional - Frontend	D3 - Editar tarefa de sua autoria	Todos
Interface Funcional - Frontend	S1 - Editar tarefa de outros utilizadores	Todos
Interface Funcional - Frontend	S2 - Apagar tarefas (suas e de outros utilizadores)	Todos
Interface Funcional - Frontend	S3 - Consultar o perfil de um utilizador específico	Todos
Interface Funcional - Frontend	S4 - Filtrar lista de tarefas criadas por um utilizador	Todos
Interface Funcional - Frontend	S5 - Filtrar lista de tarefas por categoria	Todos
Interface Funcional - Frontend	S6 - Consultar tarefas apagadas	Todos
Interface Funcional - Frontend	P1 - Alterar a informação do perfil de outros utilizadores	Todos
Interface Funcional - Frontend	P2 - Adicionar um novo utilizador	Todos
Interface Funcional - Frontend	P3 - Consultar lista de todos os utilizadores	Todos
Interface Funcional - Frontend	P4 - Apagar um utilizador	Todos
Interface Funcional - Frontend	P5 - Apagar todas as tarefas criadas por um utilizador	Todos
Interface Funcional - Frontend	P6 - Alterar a role de um utilizador	Todos
Interface Funcional - Frontend	P7 - Deletar permanentemente tarefas e utilizadores	Todos
Interface Funcional - Frontend	P8 - Adicionar categoria de tarefa	Todos
Interface Funcional - Frontend	P9 - Editar as categorias de tarefas	Todos
Interface Funcional - Frontend	P10 - Apagar categoria de tarefa	Todos
Interface Funcional - Frontend	G1. Adicionar comentários na retrospectiva	Grupo de 3
Interface Funcional - Frontend	G2.Editar o próprio comentário na retrospectiva	Grupo de 3
Interface Funcional - Frontend	G3. Criar um evento de retrospectiva	Grupo de 3
Interface Funcional - Frontend	G4. Deletar permanentemente comentários em uma retrospectiva	Grupo de 3
Interface Funcional - Frontend	G5. Adicionar membro na retrospectiva	Grupo de 3
Persistência - Backend	Modelo de dados	Todos
Persistência - Backend	Utilização de MySQL server	Todos
Persistência - Backend	Entities segundo o modelo de dados	Todos
Persistência - Backend	DAOs	Todos
Persistência - Backend	Utilização de Criteria API ou JPQL	Todos
Requisito Funcional - Backend	Token	Todos
Requisito Funcional - Backend	D3 - Editar tarefa de sua autoria	Todos
Requisito Funcional - Backend	S1 - Editar tarefa de outros utilizadores	Todos
Requisito Funcional - Backend	S2 - Apagar tarefas (suas e de outros utilizadores)	Todos
Requisito Funcional - Backend	S3 - Consultar o perfil de um utilizador específico	Todos
Requisito Funcional - Backend	S4 - Filtrar lista de tarefas criadas por um utilizador	Todos
Requisito Funcional - Backend	S5 - Filtrar lista de tarefas por categoria	Todos
Requisito Funcional - Backend	S6 - Consultar tarefas apagadas	Todos
Requisito Funcional - Backend	P1 - Alterar a informação do perfil de outros utilizadores	Todos
Requisito Funcional - Backend	P2 - Adicionar um novo utilizador	Todos
Requisito Funcional - Backend	P3 - Consultar lista de todos os utilizadores	Todos
Requisito Funcional - Backend	P4 - Apagar um utilizador	Todos
Requisito Funcional - Backend	P5 - Apagar todas as tarefas criadas por um utilizador	Todos
Requisito Funcional - Backend	P6 - Alterar a role de um utilizador	Todos
Requisito Funcional - Backend	P7 - Deletar permanentemente tarefas e utilizadores	Todos
Requisito Funcional - Backend	P8 - Adicionar categoria de tarefa	Todos
Requisito Funcional - Backend	P9 - Editar as categorias de tarefas	Todos
Requisito Funcional - Backend	P10 - Apagar categoria de tarefa	Todos
Requisito Funcional - Backend	G1. Adicionar comentários na retrospectiva	Grupo de 3
Requisito Funcional - Backend	G2.Editar o próprio comentário na retrospectiva	Grupo de 3
Requisito Funcional - Backend	G3. Criar um evento de retrospectiva	Grupo de 3
Requisito Funcional - Backend	G4. Deletar permanentemente comentários em uma retrospectiva	Grupo de 3
Requisito Funcional - Backend	G5. Adicionar membro na retrospectiva	Grupo de 3
Testes	Tem de existir pelo menos um teste no postman para cada endpoint REST	Todos
Testes	Pelo menos 10 testes da função principal em java usando Junit	Todos
Testes	Pelo menos 5 testes da função principal em java usando JEST	Todos
Testes	Implementar testes no postman, 3 testes JUNIT e 2 testes Jest para a retrospectiva.	Grupo de 3