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. Neste projeto, deve ser utilizada persistência de dados do lado do servidor através de uma base de dados *MySQL*. Deve ser usado JPA e criados 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: 5 de Março de 2023

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

Especificação

Neste projeto deve ser possível ter várias listas de atividades para cada utilizador. Cada lista de atividades pertence a uma categoria específica de atividade (trabalho, família, compras).

Devemos ter dois tipos de utilizadores no sistema, *users* e admins. Assim, deve ser possível fazer gestão dos utilizadores, dos perfis (admin e user), e das atividades. Os alunos devem criar manualmente um *admin* na base de dados para ser o administrador principal do sistema. 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 que a senha deve ser armazenada no base de dados numa forma encriptado.

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, e também o *token* resultante do processo de autenticação. Para cada atividade, o sistema deve guardar os seguintes dados de forma persistente: categoria de atividade, título, uma descrição, data de início (data e hora, se nenhuma hora for introduzida, 00:00:01 deve ser considerada como hora de início da atividade), data de fim, estado (finalizado ou não), se é necessário um lembrete (se sim, devemos também guardar o momento de alerta - minutos antes, horas antes, dias antes). Por cada categoria de atividade basta guardar um título. Os alunos são livres de adicionar mais campos, se necessário.

O *user* poderá aceder às seguintes funcionalidades:

U1- Alterar o seu perfil e imagem

U2- Adicionar uma categoria de atividades
U3- Alterar uma categoria de atividades
U4- Apagar uma categoria de atividades (o apagar de uma categoria resulta no apagar de todas as atividades incluídas na mesma)
U5- Adicionar atividades dentro das categorias
U6- Alterar atividades (todos os campos podem ser alterados)
U7- Apagar atividade

Nota: apagar as atividades e categorias de atividades da lista não deve resultar na sua remoção permanente da base de dados. 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.

Um *admin* poderá aceder às seguintes funcionalidades:

A1. Alterar a informação do perfil de outros utilizadores
A2. Adicionar um novo utilizador
A3. Consultar lista de todos os utilizadores
A4. Apagar um utilizador e as suas atividades do sistema (apagar não implica apagar os dados permanentemente do sistema, mas apenas alterar o estado dos dados para que não sejam visíveis.)
A5. Consultar o perfil de um utilizador específico
A6. Consultar lista de atividades de um utilizador
A7. Alterar a role de um utilizador (user/admin)
A8. Adicionar outro *admin* no sistema
A9. Consultar lista de todos os *admines*
A10. Apagar outro *admin*

Neste projeto **não é necessário desenvolver as páginas para a parte de gestão dos dados (e.g., alterar ou criar)**, portanto, **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.

Arquitetura do Sistema a Desenvolver

Deve ser seguida a arquitetura apresentada na Figura 1, cujos módulos são explicados em baixo.
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,

[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

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`**.

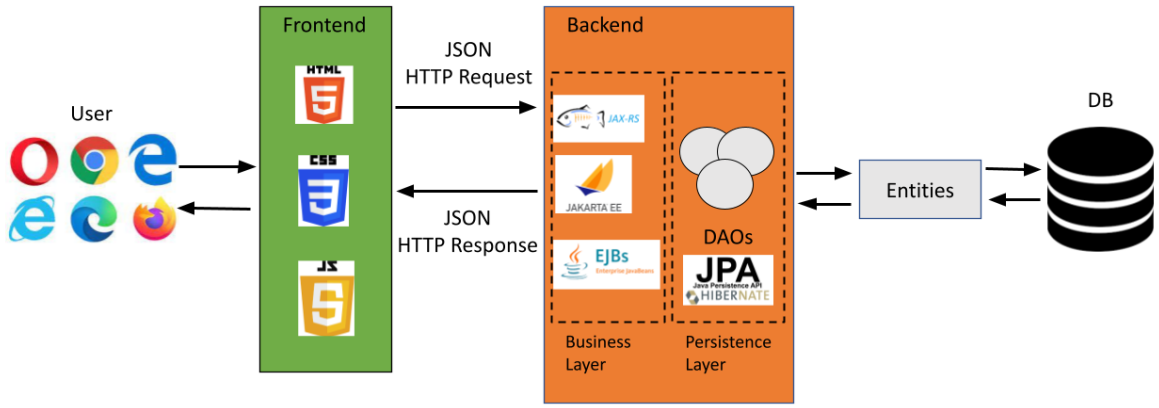

Figura 1. Arquitectura do sistema a implementar.

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 normal (user) 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**.

Grupo	Requisito
Interface Funcional - Frontend	listar as categorias de atividades
Interface Funcional - Frontend	listar atividades de cada categoria
Interface Funcional - Frontend	consultar detalhe de cada atividade
Interface Funcional - Frontend	listar as atividades e categorias apagadas
Interface Funcional - Frontend	apagar o sessionStorage na altura de logout
Interface Funcional - Postman	U2-Adicionar uma categoria de atividades
Interface Funcional - Postman	U3-Alterar uma categoria de atividades
Interface Funcional - Postman	U4-Apagar uma categoria de atividades
Interface Funcional - Postman	U5-Adicionar atividades em cada categoria
Interface Funcional - Postman	U6-Alterar atividades
Interface Funcional - Postman	U7-Apagar atividade
Interface Funcional - Postman	A1. Enquanto admin alterar a informação do perfil de outros utilizadores
Interface Funcional - Postman	A2. Adicionar um novo utilizador
Interface Funcional - Postman	A3. Consultar lista de todos os utilizadores
Interface Funcional - Postman	A4. Apagar um utilizador e as suas atividades do sistema
Interface Funcional - Postman	A5. Consultar o perfil de um utilizador específico
Interface Funcional - Postman	A6. Consultar lista de atividades de um utilizador
Interface Funcional - Postman	A7. Alterar a role de um utilizador (user/admin)
Interface Funcional - Postman	A8. Adicionar um admin no sistema
Interface Funcional - Postman	A9. Consultar lista de todos os admins
Interface Funcional - Postman	A10. Apagar um admin
Persistência - Backend	Modelo de dados
Persistência - Backend	Utilização de MySQL server
Persistência - Backend	Entities segundo o modelo de dados
Persistência - Backend	DAOs
Persistência - Backend	Utilização de Criteria API ou JPQL
Requisitos funcionais - Backend	Token
Requisitos funcionais - Backend	U2-Adicionar uma categoria de atividades
Requisitos funcionais - Backend	U3-Alterar uma categoria de atividades
Requisitos funcionais - Backend	U4-Apagar uma categoria de atividades
Requisitos funcionais - Backend	U5-Adicionar atividades em cada categoria
Requisitos funcionais - Backend	U6-Alterar atividades
Requisitos funcionais - Backend	U7-Apagar atividade
Requisitos funcionais - Backend	A1. Enquanto admin alterar a informação do perfil de outros utilizadores
Requisitos funcionais - Backend	A2. Adicionar um novo utilizador
Requisitos funcionais - Backend	A3. Consultar lista de todos os utilizadores
Requisitos funcionais - Backend	A4. Apagar um utilizador e as suas atividades do sistema
Requisitos funcionais - Backend	A5. Consultar o perfile de um utilizador específico
Requisitos funcionais - Backend	A6. Consultar lista de atividades de um utilizador
Requisitos funcionais - Backend	A7. Alterar a role de um utilizador (user/admin)
Requisitos funcionais - Backend	A8. Adicionar um admin no sistema
Requisitos funcionais - Backend	A9. Consultar lista de todos os admins
Requisitos funcionais - Backend	A10. Apagar um admin