Programação Avançada em Java
Trabalho Prático 5

Introdução

Objetivo: Este Trabalho Prático 5 constrói sobre o Projeto 4, pretendendo introduzir um conjunto de novas funcionalidades e requisitos de qualidade típicos de aplicações do mundo real, garantindo robustez, segurança, boas práticas e uma experiência de utilizador mais fluída.

Ao concluir este projeto, os alunos deverão integrar:

- Confirmação de contas por e-mail e recuperação de password;
- Sistema de mensagens pessoais e notificações em tempo real;
- Interatividade em tempo real
- Internacionalização (i18n) e responsividade;
- Sessões com timeout configurável;
- Dashboard de administrador com estatísticas;
- Logs de atividades

Data de Entrega: 24 de abril de 2025

Grupos: o projeto é realizado **individualmente.**

Novos Requisitos

A aplicação deve respeitar todos os requisitos do Projeto 4:

- Frontend em ReactJS, comunicando com um Backend em Jakarta EE (JAX-RS, EJBs)
- Persistência de dados em PostgreSQL com JPA (Hibernate)
- As mesmas entidades e funcionalidades de base (produtos, utilizadores, categorias, estados do produto, etc.) introduzidas até agora
- Controlo de acesso (diferentes tipos de utilizador)

Contudo, este Trabalho Prático 5 acrescenta as seguintes dimensões:

1. Sistema de Confirmação de Contas
2. Recuperação de Password
3. Página de Perfil do Utilizador (com URL único)
4. Listagem de Utilizadores (filtrável)
5. Mensagens Pessoais e Notificações (com contagem de notificações)
6. Internacionalização e Localização
7. Dashboard do Administrador (estatísticas e gráficos)
8. Atualizações em Tempo Real (WebSockets)
9. Session Timeout
10. Logs
11. Responsividade (Mobile, Tablet, Desktop)
12. Comunicação Eficiente

A seguir detalhamos cada uma destas dimensões.

Sistema de Confirmação de Contas

- **Fluxo de criação:** O Administrador cria contas de utilizador (role Cliente), fornecendo um e-mail.
- **Estado de conta:** A conta fica em estado "aguardando confirmação" até que o utilizador clique no link de confirmação que recebe por e-mail.
- **Link de confirmação:** Ao clicar, o utilizador confirma definitivamente a conta no sistema.
- **Acesso restrito:** Antes de confirmar a conta, o utilizador não pode comprar ou vender produtos. Caso tente, a API deve responder com erro (*e.g.*, 403 Forbidden).
- Para facilitar a implementação, podem enviar um e-mail sempre para a mesma conta.

Recuperação de Password

- **Página "Esqueceu a password?":** Permite ao utilizador solicitar a redefinição de password, introduzindo o seu e-mail.
- **E-mail de reset:** Caso o e-mail exista, o sistema envia um link de redefinição.
- **Token único:** O link deve conter um token único que permita alterar a password apenas uma vez.

- Após redefinição, o token deixa de ser válido.

Página de Perfil do Utilizador

Cada utilizador (Cliente ou Administrador) deve ter uma página de perfil com um URL único (e.g. /users/vitinha).

Conteúdos da página de perfil

- Informações do utilizador: Primeiro nome, último nome, username, e-mail, fotografia.
- Estatísticas de produtos:
    - Número total de produtos associados (produtos que criou).
    - Distribuição desses produtos por estado: “rascunho”, “publicado” (ou “disponível”), “reservado”, “comprado”.
- Botão para editar configurações pessoais (foto de perfil, e-mail, password, etc.).

Regras

- Um utilizador só pode editar o próprio perfil (exceto Administrador, que pode também editar perfis de outros).

Listagem de utilizadores

- Deve existir uma **página que lista todos os utilizadores**, mostrando username, fotografia e e-mail.
- **Filtragem:** Permitir filtrar por username ou e-mail.
- **Ao clicar num utilizador:** ir para a página de perfil desse utilizador.

Sistema de Mensagens pessoais e Notificações

Mensagens Pessoais

- **Escopo:** Dois utilizadores podem trocar mensagens diretas (1 a 1).
- **Página de perfil:** Quando um utilizador (A) visita a página de perfil de outro utilizador (B), deve poder ver o histórico de mensagens trocadas entre A e B (se houver).
- **Marcar como lida:** Ao receber mensagens novas, são inicialmente "não lidas". Clicar numa mensagem faz com que todas as mensagens anteriores também sejam marcadas como lidas. Um sistema que funcione com o "scroll" também é válido (i.e. "se vemos uma mensagem, considera-se que vimos todas as anteriores").
- **Mecanismo:** Deve existir uma API REST para obter a lista de mensagens e também para enviar. Ao mesmo tempo (ver secção abaixo sobre websockets), deve também ser possível a troca de mensagens através de WebSockets. Se a ligação via WebSocket estiver activa, todas as mensagens são trocadas desta forma.

Notificações

- Contador de notificações: Sempre visível, mostrando o número total de notificações não lidas.
- Lista de notificações: Ao clicar no contador, surge a lista de notificações.
- Marcar como lidas: Ao abrir a lista, todas as notificações não lidas são marcadas como lidas.
- Redirecionamento: Ao clicar numa notificação relativa a uma mensagem, o utilizador é redirecionado para o perfil do remetente da mensagem.
- Outras notificações a implementar:
    - Quando um dos produtos do utilizador é vendido
    - Outras – a definir

Observação: Se o utilizador A estiver na página de perfil do utilizador B e receber uma nova mensagem de B, a notificação deve ser marcada automaticamente como lida (visto que A já está a ver a conversa), ou nem sequer ser enviada.

Internacionalização e Localização

- **Mínimo de 2 idiomas** (*e.g.*, PT e EN).
- **Preferências:** Cada utilizador define a sua língua preferida na página de perfil.
- **Texto estático (rótulos, botões, cabeçalhos) deve mudar consoante a língua escolhida**. Conteúdo dinâmico (e.g.: títulos e descrições de produtos) não é traduzido automaticamente (não é preciso traduzir).
- **Estrutura:** Usar ficheiros de tradução (e.g. JSON ou outra ferramenta) no frontend React para mudar texto de UI.

Dashboard do Administrador

Somente o Administrador pode aceder a este dashboard, que contém estatísticas globais:

- Contagem do número total de utilizadores, bem como quantos deles confirmaram a conta.
- Contagem do número de produtos, por estado (“rascunho,” “publicado,” “reservado,” “comprado”).
- Listagem das categorias, ordenadas da mais frequente (mais produtos) para a menos frequente.
- Produtos por utilizador: Tabela que mostre para cada utilizador o total de produtos criados, a distribuição por estado, etc.
- Tempo médio para um produto ser “comprado” (diferença entre data de publicação e data de transição para “comprado”).
- Gráficos:
    - Número de utilizadores registados ao longo do tempo (e.g. gráfico de linhas).
    - Número total de produtos “comprados” ao longo do tempo (gráfico cumulativo).

Atualizações em Tempo Real (WebSockets)

A aplicação deve suportar atualizações em tempo real sem atualização (“refresh”) manual:

- **Quando alguém altera um produto** (título, preço, estado, etc.), essa mudança reflete-se instantaneamente para outros utilizadores que visualizam o mesmo produto ou a lista de produtos.
- **Mensagens pessoais:** Se o utilizador está na conversa, a nova mensagem surge de imediato.
- **Notificações:** Contador de notificações é atualizado em tempo real.
- **Dashboard do administrador:** Sempre que surgem novos utilizadores ou novos produtos, as estatísticas e gráficos são atualizados instantaneamente.
- **Envio de Mensagens:** Para demonstrar a vantagem de comunicação bidirecional, o envio de mensagens pessoais pode (e deve) ser realizado também via WebSockets, em vez de usar sempre REST. Por outras palavras: para além da existência de um endpoint REST

Session Timeout

- Após X minutos de inatividade (configuráveis pelo Administrador), a sessão do utilizador expira automaticamente.
- O utilizador é forçado a efetuar logout e, se tentar usar endpoints, deve receber 401 Unauthorized ou 403 Forbidden.

Logs

- Todas as atividades relevantes (criação e edição de produtos, alterações em perfis, exclusão de utilizadores, etc.) devem ser registadas em logs internos num ficheiro.
- Cada entrada de log inclui data/hora, ID do utilizador ou token, e IP de origem (se disponível).

Responsividade (Mobile, Tablet, Desktop)

- As páginas devem ser responsivas, adaptando-se a ecrãs de telemóvel, tablets e monitores de desktop.
- É aceitável uma abordagem mobile-first, garantindo que:
    - O layout (menus, cards, formulários) não apresenta artefactos visuais estranhos.
    - Elementos ficam corretamente posicionados em diferentes resoluções.

Controlo de acesso

Reforçamos o seguinte:

- Um Cliente não pode editar produtos de outro utilizador, salvo se o projeto tiver uma lógica que o permita.
- Um Cliente não pode ver perfis nem mensagens que não lhe pertençam.
- Só o Administrador pode criar categorias, apagar permanentemente produtos, e gerir contas de utilizador.

Comunicação Eficiente

- Não enviar mais dados que o necessário em cada chamada REST/WebSocket.
- Filtragem e ordenação deve ser feita no backend (evitar filtrar em massa no frontend), exceto em situações em que todos os dados já estão no frontend
- Paginação e/ou limites em listagens extensas (opcional, mas fortemente recomendado).

Arquitetura do Sistema a Desenvolver

A arquitetura do sistema anterior mantém-se, destacando-se o surgimento de websockets para a comunicação em tempo-real.

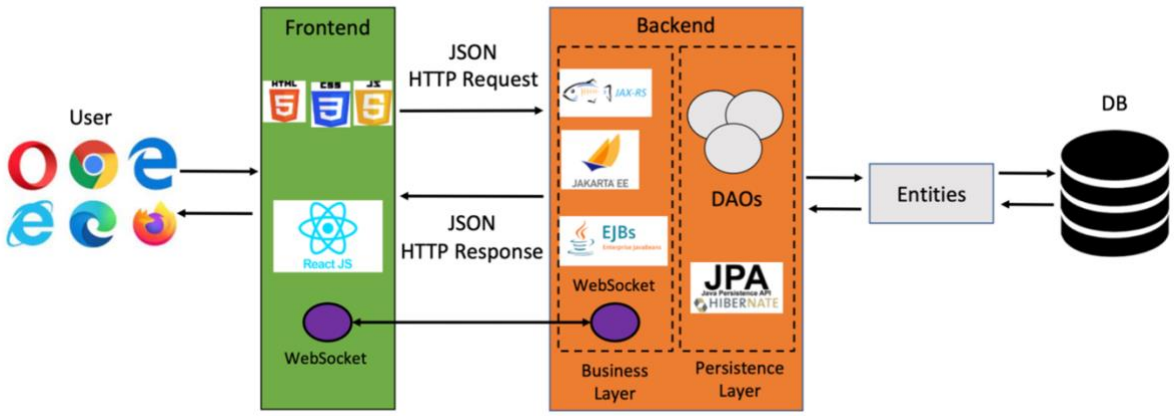

Figura 1. Arquitetura do sistema a implementar.

REST API

Todas as considerações sobre API REST feitas no projeto anterior se mantém.

Gestão de Repositórios GIT e Configuração

De forma a facilitar a organização do backend e do frontend, cada componente deverá ser criada num repositório GIT independente. O processo deverá ser exatamente igual ao do projecto anterior (4), mudando “4” para “5”.

Bibliotecas Externas

Para a comunicação com websockets e para o desenho de gráficos, podem ser utilizadas bibliotecas externas.

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.

Defesa: As defesas terão uma duração de **30 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**.