

# Tecnologias Web

Licenciatura em Engenharia Informática

## 2º Trabalho

Aplicação Web - Room Rent



**André Gonçalves, 58392 | André Zhan, 58762**

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Arquitetura da Aplicação (em Camadas)</b>	<b>2</b>
2.1	Presentation Layer . . . . .	2
2.2	Control Layer . . . . .	2
2.3	Service Layer . . . . .	2
2.4	Data Access Layer . . . . .	2
2.5	Database Layer . . . . .	3
<b>3</b>	<b>Aspetos Omissos</b>	<b>3</b>
3.1	Propósito dos Dados de Pagamento . . . . .	3
3.2	Uso de REST API . . . . .	3
3.3	Dashboard Pessoal Completo . . . . .	3
<b>4</b>	<b>Balanco Crítico</b>	<b>3</b>

# 1 Introdução

Este projeto consiste numa aplicação web full-stack desenvolvida em Java/Spring Boot que funciona como plataforma de gestão de anúncios de alojamento. O sistema permite que utilizadores publiquem anúncios de oferta ou procura de quartos/apartamentos, comuniquem com os anunciantes, e efetuem pagamentos de taxas de publicação via referências Multibanco.

A aplicação atende aos requisitos especificados no enunciado do trabalho e implementa um modelo de negócio com três tipos de acesso:

- Público: visualização de anúncios e registo;
- Autenticado: criação de anúncios, envio de mensagens, pagamentos;
- Administrativo: aprovação de contas e gestão de anúncios.

## 2 Arquitetura da Aplicação (em Camadas)

### 2.1 Presentation Layer

Agrupar todos os templates HTML renderizados pelo Thymeleaf, os formulários de interação do utilizador e as páginas da aplicação. Esta camada é responsável por apresentar a interface ao utilizador final.

### 2.2 Control Layer

É composta pelos Controllers da aplicação (HomeController, AdController, etc.). Esta camada funciona como intermediária entre as requisições HTTP e a lógica de negócio, mapeando cada URL para o método correspondente, validando os dados de entrada e populando o Model com os dados necessários para renderizar a resposta.

### 2.3 Service Layer

Contém toda a lógica de negócio da aplicação.

### 2.4 Data Access Layer

É composta pelos Repositories, que abstraem o acesso aos dados através de interfaces do Spring Data JPA. Estes repositories contêm as queries customizadas e as operações de CRUD (Create, Read, Update, Delete) contra a base de dados.

## 2.5 Database Layer

Consiste no PostgreSQL propriamente dito, com as tabelas relacionais e integridade referencial através de chaves estrangeiras.

## 3 Aspetos Omissos

### 3.1 Propósito dos Dados de Pagamento

Como não estava explícito no enunciado, decidimos que o propósito dos dados de pagamento Multibanco obtidos através do serviço externo corresponderiam à taxa de publicação do anúncio que o proprietário teria que pagar para o anúncio ser ativado pelo administrador, com uma taxa fixa de 5,99€.

### 3.2 Uso de REST API

Decidimos usar o RestTemplate do Spring para HTTP requests, mais especificamente para obter os dados de pagamento MB através do serviço externo fornecido.

### 3.3 Dashboard Pessoal Completo

Optámos por implementar um dashboard pessoal para os utilizadores simples autenticados, onde os mesmos podem visualizar todos os seus anúncios presentes na base de dados, ativos ou inativos. Além disso podem consultar as mensagens associadas a cada anúncio (se houverem), assim como os dados de pagamento MB de cada anúncio.

## 4 Balanço Crítico

O projeto atende completamente os requisitos do enunciado com uma arquitetura clara e bem organizada em camadas. A escolha de Spring Boot, Java 17 e PostgreSQL é apropriada e bem executada. A segurança implementada (BCrypt, CSRF, autorização, validação) é robusta.

No entanto, existem limitações, como a ausência de testes automatizados que compromete a fiabilidade da APP e o Thymeleaf server-side resulta numa interface básica comparada com SPAs modernas.

Para um trabalho académico, a nosso ver, o resultado é sólido com boas decisões técnicas justificadas. Para produção, seria necessário investimento em testes frontend moderno e um frontend mais moderno.