Universidade da Beira Interior

Departamento de Informática

Nº T5 – 2023: IMOBILIÁRIA

Elaborado por:

Israel Torres e Ivanildo Miti Paulo

Orientadora:

Professora Doutora Paula Prata

28 de maio de 2023

Integrantes do grupo T5 – Imobiliária:

Israel Torres - 40715
Ivanildo Miti Paulo – 36558

Índice

Capítulo 1

Introdução

1.1 Enquadramento

O presente projeto surge no âmbito da unidade curricular Sistemas Distribuídos do curso de engenharia informática. Este projeto de forma resumida visa no desenvolvimento de uma aplicação *web* de uma imobiliária, utilizando a *framework Spring* e baseada em *Jakarta EE*. O objetivo dessa aplicação é fornecer serviços de gerenciamento de imobiliaria oferecendo funcionalidades para anúncios, consultas, vendas, aluguel e estatísticas relacionadas ao funcionamento da empresa.

1.2 Motivação

A motivação deste projeto é desenvolver uma aplicação web para uma empresa imobiliária, que permita gerir eficientemente os serviços oferecidos por essa empresa.

A escolha do tema imobiliária se deu pela importância desse setor no mercado e pela necessidade de uma solução eficiente para o gerenciamento de imóveis, consultas, vendas e aluguéis. A aplicação visa facilitar o processo de anúncio de imóveis, contratação de serviços relacionados à locação e fornecer informações estatísticas para ajudar os anunciantes a tomar decisões estratégicas.

Através dessa aplicação, proprietários de imóveis podem cadastrar suas propriedades para venda e aluguel, especificando detalhes como cidade, descrição, tipo de anúncio e preço. Os clientes, por sua vez, podem realizar consultas filtrando por critérios como cidade, tipo de imóvel e faixa de preço, facilitando a busca por um imóvel adequado às suas necessidades.

1.3 Objetivos

Com base nas recomendações deixadas para implementação deste projeto, os objetivos são:

- Gestão de Serviços: Permitir a criação, visualização, atualização e exclusão de registros relacionados aos serviços oferecidos pela empresa imobiliária. Isso inclui a gestão de imóveis, contratos, consultas e estatísticas.

- Consultas de Venda e Aluguéis: Possibilitar aos clientes realizar consultas de imóveis com base em critérios específicos, visualizar detalhes dos imóveis disponíveis e efetuar solicitações de venda ou aluguel. Além disso, permitir à empresa gerenciar essas solicitações e realizar o processo de venda ou aluguel de forma eficiente.

- Estatísticas: Fornecer informações estatísticas sobre o funcionamento da empresa imobiliária, como o número de moradias vendidas, o número de moradias arrendadas, o valor total das moradias vendidas, o valor total (mensal) das moradias arrendadas, o número de clientes diferentes, e o número de cidades diferentes com moradias ativas. Essas estatísticas auxiliam a empresa a tomar decisões estratégicas e melhorar seus serviços.

- Arquitetura e Funcionalidades: Desenvolver uma arquitetura robusta e escalável para a aplicação, seguindo as boas práticas da *framework Spring*. Implementar funcionalidades que atendam às necessidades da empresa imobiliária, levando em consideração os requisitos do negócio e a experiência do usuário.

- Manual de Configuração e Instalação: Criar um manual detalhado que descreva o processo de configuração e instalação da aplicação, permitindo que outras pessoas possam replicar o ambiente de desenvolvimento e utilizar a aplicação de forma adequada.

1.4 Organização do Documento

1. O primeiro capítulo – I**ntrodução** – contextualiza o projeto, a motivação para sua escolha, o seu enquadramento, os objetivos a serem alcançados e a organização do documento.

2. O segundo capítulo – **Modelo de Dados** – descreve o modelo de dados utilizado na aplicação. Apresenta as entidades envolvidas e os relacionamentos entre elas.

3. O terceiro capítulo – **Descrição da Arquitetura e Funcionalidades da Aplicação** – descreve a estrutura do sistema e o funcionamento das principais funcionalidades do projeto.

4. O quarto capítulo – **Manual de Configuração e Instalação** – descreve os passos necessários para configurar e instalar a aplicação.

5. O quinto capítulo – **Conclusão** – apresenta as principais ideias que foram concluídas através da realização do projeto e algumas propostas para o trabalho futuro.

Capítulo

2

Modelo de Dados

2.1 Introdução

O modelo de dados da aplicação de imobiliária é projetado para fornecer uma estrutura organizada para armazenar e gerenciar informações relacionadas a clientes, imóveis, contratos e consultas. O objetivo principal é permitir que a imobiliária gerencie eficientemente os serviços oferecidos, como a realização de contratos de vendas e aluguéis.

2.2 Entidades

O modelo de dados para o projeto de imobiliária envolve a representação das entidades e relacionamentos necessários para o gerenciamento de imóveis, consultas, vendas, aluguéis e estatísticas da empresa. A seguir, apresentamos as principais entidades do modelo:

1. **Anunciante:**

Tabela: Anunciante

Campo	**Tipo**	**Descrição**
aId	Integer	Chave primária autoincrementada
name	String	Nome do anunciante
email	String	Email do anunciante
salt	String	Valor do salt utilizado para criptografar a senha
psw	String	Senha do anunciante (utilizada

		temporariamente e não guardada na base de dados)
salt_Psw_Hash	String	Hash da senha do anunciante com o salt
housesSold	Integer	Contador de casas alugadas
rentedHouses	Integer	Contador de casas alugadas
totalArrendamento	Integer	Total de valor arrecadado com aluguéis
contracts_Ids	List of Integers	Chaves estrangeiras que referenciam os contratos
comes_Ids	List of Integers	Chaves estrangeiras que referenciam as moradias
totalSold	Integer	Total de valor arrecadado com vendas

Relacionamentos:

- Um Anunciante pode ter vários *Home* associados a ele (relacionamento *One-to-Many*). A tabela *Home* possui uma coluna *responsibleUser* que faz referência ao *aId* do Anunciante.

- Um Anunciante pode ter vários *Contract* associados a ele (relacionamento *One-to-Many*). A tabela Contract possui uma coluna *responsibleUser* que faz referência ao *aId* do Anunciante.

Esse modelo de dados representa a estrutura da tabela Anunciante e os relacionamentos com as tabelas *Home e Contract.* Os campos representam as informações relevantes sobre o anunciante, como nome, email, senha (criptografada) e contadores relacionados a vendas e aluguéis. Os relacionamentos permitem associar as casas e contratos ao anunciante correspondente.

2. Client:

Tabela: Client

Campo	Tipo	Descrição
cId	Integer	Chave primária autoincrementada
name	String	Nome do cliente
email	String	Email do cliente
salt	String	Valor do salt utilizado para criptografar a senha
contracts_Ids	List of Integers	Chaves estrangeiras que referenciam os contratos
homes_Id	List of Integers	Chaves estrangeiras que referenciam as moradias
psw	String	Senha do cliente (utilizada temporariamente e não guardada na base de dados)
salt_Psw_Hash	String	Hash da senha do cliente com o salt

Relacionamentos:

- Um *Client* pode ter vários *Home* associados a ele (relacionamento *One-to-Many*). A tabela *Home* possui uma coluna *clientUser* que faz referência ao *cId* do *Client.*
- Um Client pode ter vários *Contract* associados a ele (relacionamento *One-to-Many*). A tabela *Contract* possui uma coluna *clientUser* que faz referência ao *cId* do *Client*.

Esse modelo de dados representa a estrutura da tabela *Client* e os relacionamentos com as tabelas *Home e Contract*. Os campos representam as informações relevantes sobre o cliente, como nome, email, senha (criptografada) e *salt* utilizado para criptografar a senha. Os relacionamentos permitem associar as casas e contratos ao cliente correspondente.

3. Contract:

Tabela: Contract

Campo	Tipo	Descrição
cId	Integer	Chave primária autoincrementada
home_id	Integer	Chave estrangeira que referencia a casa
description	String	Descrição do contrato
responsibleUser_id	Integer	Chave estrangeira que referencia, anunciante responsável
clientUser_id	Integer	Chave estrangeira que referencia o cliente associado
status	String	Status do contrato (pendente, assinado, disponível)

Relacionamentos:

- Um *Contract* está associado a uma única *Home* (relacionamento *One-to-One*). A coluna *home_id* na tabela *Contract* faz referência à chave primária *id* na tabela *Home*.
- Um Contract possui um único anunciante responsável (Anunciante) (relacionamento *Many-to-One).* A coluna *responsibleUser_id* na tabela *Contract* faz referência à chave primária *aId* na tabela Anunciante.
- Um *Contract* está associado a um único cliente (*Client*) (relacionamento *Many-to-One).* A coluna *clientUser_id* na tabela *Contract* faz referência à chave primária *cId* na tabela *Client*.

Esse modelo de dados representa a estrutura da tabela *Contract* e os relacionamentos com as tabelas *Home*, Anunciante e *Client*. Os campos representam as informações relevantes sobre o contrato, como descrição e *status*. Os relacionamentos permitem associar a casa, o anunciante responsável e o cliente ao contrato correspondente.

4. Home:

Tabela: Home

Campo	Tipo	Descrição
hId	Integer	Chave primária autoincrementada
type	String	Indica o tipo da casa, como Venda ou Arrendamento
responsibleUser_Id	Integer	Chave estrangeira que referencia, anunciante responsável
description	String	Descrição da casa
Contract_Id	Integer	Chave estrangeira que referencia o Contract
city	String	Indica a cidade onde a casa está localizada
cost	Integer	Indica o preço da moradia
minCost	Integer	representa o custo mínimo para uma pesquisa
maxCost	Integer	representa o custo máximo para uma pesquisa
status	String	Status da casa, como Disponível, Arrendada ou Comprada.
clientUser_Id	Integer	Chave estrangeira que referencia o Client

Relacionamentos:

- Relacionamento com a classe Anunciante (*responsibleUser):*
- Chave Estrangeira: *responsibleUser_aId*
- Tipo de Relacionamento: Muitas casas podem ter um único anunciante responsável (*Many-to-One*)
- Um anunciante pode ser responsável por várias casas

- Relacionamento com a classe *Contract* (*contract*):
- Chave Estrangeira: *contract_cId*
- Tipo de Relacionamento: Uma casa tem um contrato associado (*One-to-One*)
- Um contrato está associado a uma única casa

- Relacionamento com a classe *Client* (*clientUser*):
- Chave Estrangeira: *clientUser_cId*
- Tipo de Relacionamento: Muitas casas podem ter um único cliente (*Many-to-One*)
- Um cliente pode estar associado a várias casas

Esses relacionamentos são representados nas colunas da tabela *Home* mencionadas anteriormente, que atuam como chaves estrangeiras para as tabelas Anunciante, *Contract e Client*. Eles estabelecem as associações entre as entidades.

Com esse modelo de dados, é possível estabelecer as relações entre as entidades e realizar as operações de criação, leitura, atualização e exclusão de dados para a gestão dos serviços fornecidos pela imobiliária.

Capítulo 3

Descrição da Arquitetura e Funcionalidades da Aplicação

3.1 Introdução

A aplicação imobiliária foi desenvolvida com o objetivo de facilitar e aprimorar a gestão dos serviços fornecidos por uma empresa imobiliária. Ela permite o gerenciamento eficiente de imóveis, contratos, consultas e estatísticas relevantes para o negócio.

3.2 Arquitetura da Aplicação

A aplicação de imobiliária é desenvolvida utilizando a *framework Spring* e adota uma arquitetura em camadas (ou arquitetura MVC - *Model-View-Controller)* para organizar e separar as responsabilidades dos diferentes componentes.

1. Camada de Modelo (*Model*):
    - Responsável pela representação das entidades de negócio, implementadas como classes Java, refletindo o modelo de dados.
    - Inclui as classes como "Anunciante", "Client", "Contract" e "Home".
    - Utiliza anotações do Spring para mapeamento de objetos para o banco de dados, como a anotação *@Entity*.

2. Camada de Visualização (View):
    - Responsável pela interface com o usuário.
    - Utiliza arquivos *HTML, CSS e JavaScript* para a criação das páginas web.
    - O *Thymeleaf*, um mecanismo de *template* do *Spring*, é utilizado para renderizar as páginas *HTML* e permitir a interação com os dados do modelo.

3. Camada de Controle (*Controller*):
    - Responsável pelo gerenciamento das requisições *HTTP* e pela lógica de negócio da aplicação.
    - Utiliza classes Java anotadas com *@Controller* para receber as requisições e processá-las.
    - Interage com a camada de serviço para executar as operações necessárias nos dados.

4. Camada de Serviço (*Service*):
    - Responsável por implementar a lógica de negócio da aplicação.
    - Contém classes Java que realizam operações mais complexas, como validações, cálculos e consultas avançadas.
    - Utiliza interfaces e classes anotadas com *@Service* para definir e implementar os serviços disponíveis.

5. Camada de Acesso a Dados (*Data Access*):
    - Responsável pela comunicação com o banco de dados.
    - Utiliza o *Spring Data JPA* para facilitar o acesso aos dados e realizar operações de persistência.
    - Define interfaces de repositório que estendem a interface *JpaRepository* para acessar os dados das entidades.

3.2 Funcionalidades da Aplicação

- Gerenciamento de Imóveis: Permite cadastrar, visualizar, atualizar e excluir imóveis. Os usuários podem adicionar informações como cidade, descrição, tipo, preço e status dos imóveis.
- Consultas: Os usuários podem realizar consultas em imóveis disponíveis, visualizando detalhes e entrando em contato com os responsáveis pelos imóveis.
- Contratos: Possibilita a criação, visualização e gestão de contratos de aluguel ou venda de imóveis. Os usuários podem fechar contratos, definindo datas de início e término, valores e outras informações relevantes.
- Estatísticas: A aplicação disponibiliza um conjunto de estatísticas sobre o funcionamento da empresa, como o númro de moradias arrendadas, o número de moradias vendidas, o valor total adquirido com ambos os tipos, a quantidade de clientes diferentes, e a quantidade de cidades diferentes com moradias ativas.

Capítulo

4

Manual de Configuração e Instalação

Este manual descreve os passos necessários para configurar e instalar o projeto de Imobiliária em um ambiente local. Certifique-se de ter cumprido os pré-requisitos antes de prosseguir.

Pré-requisitos:

1. *Java Development Kit (JDK)* 8 ou superior instalado em sua máquina.
2. Um servidor de banco de dados suportado, como *MySQL*, *PostgreSQL* ou *H2*.
3. Um ambiente de desenvolvimento integrado (*IDE*) como Eclipse ou *Netbeans*

Passo 1: Configuração do Banco de Dados

1. Instale e configure o servidor de banco de dados de sua escolha.
2. Crie um novo banco de dados para a aplicação imobiliária.
3. Anote as informações de conexão do banco de dados, como URL, nome do banco de dados, nome de usuário e senha.

Passo 2: Clonar o repositório do projeto

Passo 3: Configuração do Projeto

1. Abra o projeto em seu ambiente de desenvolvimento.
2. Localize o arquivo *application.properties* na pasta *src/main/resources*.
3. Abra o arquivo *application.properties* e atualize as configurações do banco de dados com as informações anotadas no Passo 1.

Passo 4: Compilação e Execução

1. Certifique-se de que sua *IDE* esteja configurada para utilizar o *JDK* adequado.
2. Compile o projeto para verificar se não há erros de compilação.
3. Execute o projeto. Isso iniciará o servidor de aplicação embutido.

Passo 5: Acessando a Aplicação

1. Abra o navegador da *web* de sua preferência.
2. Acesse o seguinte endereço: *http://localhost:8080* ou a porta configurada para o servidor de aplicação embutido.
3. A aplicação imobiliária será carregada e estará pronta para uso.

Observação: Certifique-se de que o banco de dados esteja em execução antes de iniciar a aplicação.

Agora você pode começar a explorar e utilizar as funcionalidades da aplicação imobiliária.

Lembre-se de consultar a documentação do projeto e o código-fonte para obter mais detalhes sobre as funcionalidades específicas e personalizar a aplicação de acordo com suas necessidades.

Capítulo

5

Conclusão

Neste projeto de Imobiliária, desenvolvemos uma aplicação web baseada na *framework Spring*, que permite gerir os serviços fornecidos por uma empresa imobiliária. Através da aplicação, os usuários podem realizar operações de manipulação de dados, consultar, vender e alugar propriedades, além de obter estatísticas sobre o funcionamento da empresa.

Durante o desenvolvimento do projeto, foi criado um modelo de dados adequado para representar as entidades relacionadas ao negócio imobiliário, como Anunciante, Client, Contract e Home. Utilizamos um servidor de banco de dados para persistir esses dados.

A arquitetura da aplicação foi estruturada de acordo com os princípios do *Spring*, utilizando a injeção de dependência e a separação das camadas de controle, serviço e persistência. Isso permitiu um código mais modular, reutilizável e de fácil manutenção.

Durante o desenvolvimento do projeto, enfrentamos desafios comuns, como a integração com o banco de dados, o tratamento de exceções e a validação dos dados de entrada. No entanto, com a ajuda da *framework Spring* e de boas práticas de desenvolvimento, conseguimos superar esses desafios e entregar uma aplicação funcional e de qualidade.