

MAURÍCIO VIEIRA PEREIRA - RM553748

LUIZ OTÁVIO LEITÃO SILVA - RM553542

VITOR ONOFRE RAMOS - RM553241

Sumário

[Objetivo e escopo do projeto 3](#_Toc167029012)

[Descrição das Funcionalidades 3](#_Toc167029013)

[Protótipos 5](#_Toc167029014)

* [Tela Inicial 5](#_Toc167029015)
* [Tela de Cadastro de Cliente 6](#_Toc167029016)
* [Página de Login 7](#_Toc167029017)
* [Página de Cadastro de Produto 7](#_Toc167029018)

[Modelagem de Dados 8](#_Toc167029019)

[Diagrama de classes 9](#_Toc167029020)

[Tabela de *endpoints* 9](#_Toc167029021)

[Executar aplicação 10](#_Toc167029022)

[Link do repositório 11](#_Toc167029023)

# Objetivo e escopo do projeto

O Projeto SALESFORCE DIRECTIONS tem como objetivo fazer com que o portal seja totalmente utilizável por meio das teclas do teclado, mais especificamente, as teclas de setas e a tecla “Enter”. Essa proposta, fará com que o acesso do site, por pessoas que possuem alguma deficiência e tem dificuldade ou incapacidade de utilizar o mouse, seja mais fluído e natural, dispensando o uso da tecla “Tab” e do atalho “SHIFT+Tab” que já são utilizados, porém são unidirecionais, fazendo com que muitas vezes a utilização seja demorada.

A parte do back-end é uma aplicação Java que utiliza SQL e Maven para gerenciar informações sobre clientes, produtos e endereços. O projeto foi desenvolvido no IntelliJ IDEA 2023.3.3 e é executado em um sistema operacional Windows.

O escopo do projeto inclui a criação, leitura, atualização e exclusão (CRUD) de informações sobre clientes, produtos e endereços. Além disso, o projeto também possui recursos (Resources) que facilitam a conexão com o front-end, permitindo que a lógica de negócios seja separada da interface do usuário.

O projeto é estruturado em várias classes e métodos que interagem com um banco de dados para realizar as operações CRUD. As classes Repository são responsáveis pela lógica de persistência de dados, enquanto as classes Resources expõem essa lógica para o front-end através de uma API RESTful.

Para configurar e executar o projeto, os usuários devem clonar o repositório do GitHub, alterar as configurações do banco de dados no arquivo DatabaseConnection.java, abrir o projeto no IntelliJ IDEA (ou outra IDE de preferência) e executar o arquivo “test.java”.

# Descrição das Funcionalidades

O projeto possui várias funcionalidades, principalmente voltadas para o gerenciamento de clientes, produtos e endereços. Aqui estão as principais funcionalidades:

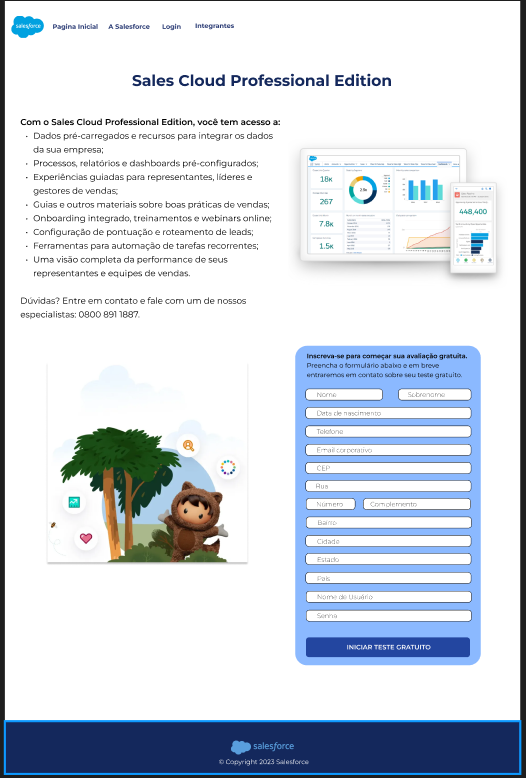
1. Gerenciamento de Clientes: A aplicação permite a criação, leitura, atualização e exclusão (CRUD) de informações do cliente. Isso inclui detalhes como o nome, sobrenome, data de nascimento, número de telefone, email corporativo, nome de usuário e senha do cliente.
2. Gerenciamento de Produtos: Semelhante ao gerenciamento de clientes, a aplicação também permite operações CRUD para produtos. Isso inclui detalhes como nome, descrição do produto, valor, quantidade em estoque e o cliente que o disponibiliza.
3. Gerenciamento de Endereços: A aplicação permite operações CRUD para endereços. Isso inclui detalhes como CEP, logradouro, número, complemento, bairro, cidade, estado, país e o cliente que possui o respectivo endereço.
4. Conexão com o Banco de Dados: A aplicação se conecta a um banco de dados para armazenar e recuperar as informações acima. Os detalhes da conexão podem ser configurados no arquivo `DatabaseConnection.java`.
5. Registro de Log: A aplicação usa a classe `Log4jLogger` para registrar as operações realizadas nos dados.
6. API endpoints: A aplicação expõe suas funcionalidades por meio de uma API RESTful, permitindo que ela interaja com uma aplicação front-end. Os pontos finais específicos e suas funcionalidades não são especificados no contexto fornecido.
7. ResetDatabase: A classe ResetDatabase está localizada no pacote org.fiap.connection. Esta classe é responsável por redefinir o banco de dados e popular as tabelas de países, estados, cidades e bairros. Ela foi criada para lidar com a geração automática de IDs, que pode causar problemas durante os testes. A classe lê um arquivo .sql e executa cada comando SQL individualmente para redefinir o banco de dados. Por favor, note que esta classe é usada apenas durante a fase inicial de testes. Ela lê o arquivo “SQL/FIAP.sql” e executa os comandos SQL contidos nele para redefinir o banco de dados.

Por favor, note que os detalhes específicos das funcionalidades de gerenciamento de produtos e endereços, bem como os pontos finais da API, seriam encontrados nas respectivas classes na base de código do projeto.

# Protótipos

## Tela Inicial

## Tela de Cadastro de Cliente

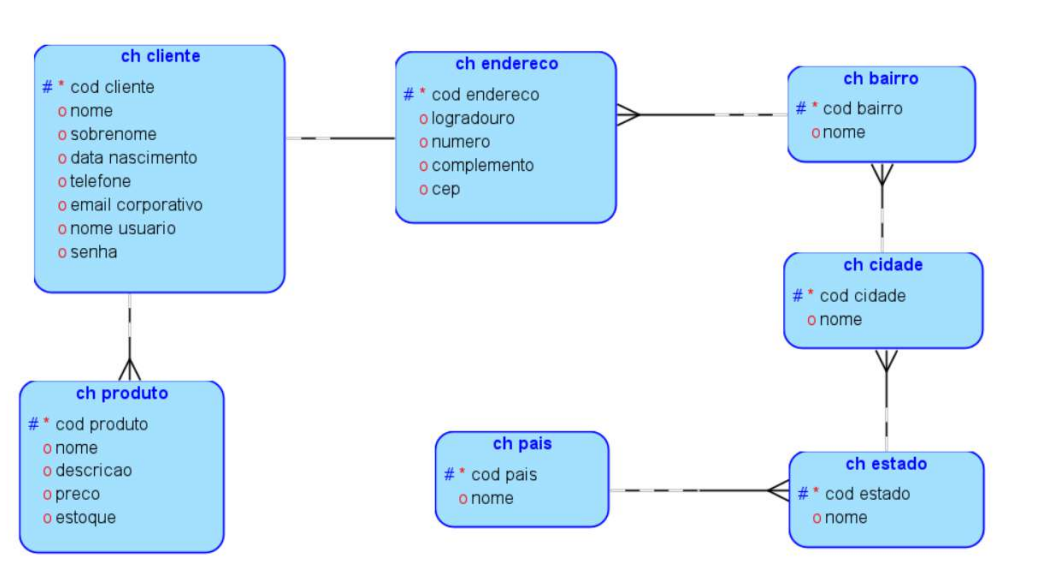


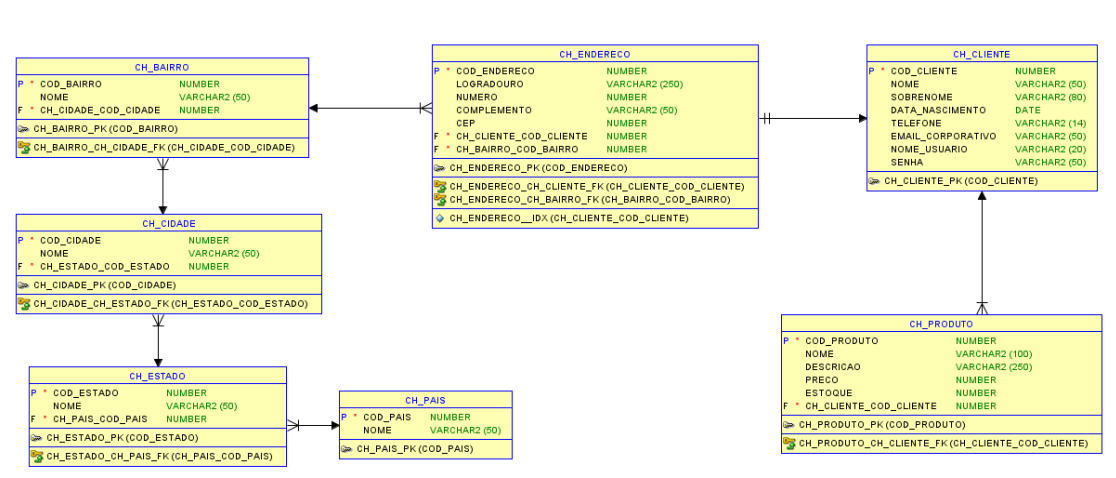
## Página de Login

## Página de Cadastro de Produto

# Modelagem de Dados

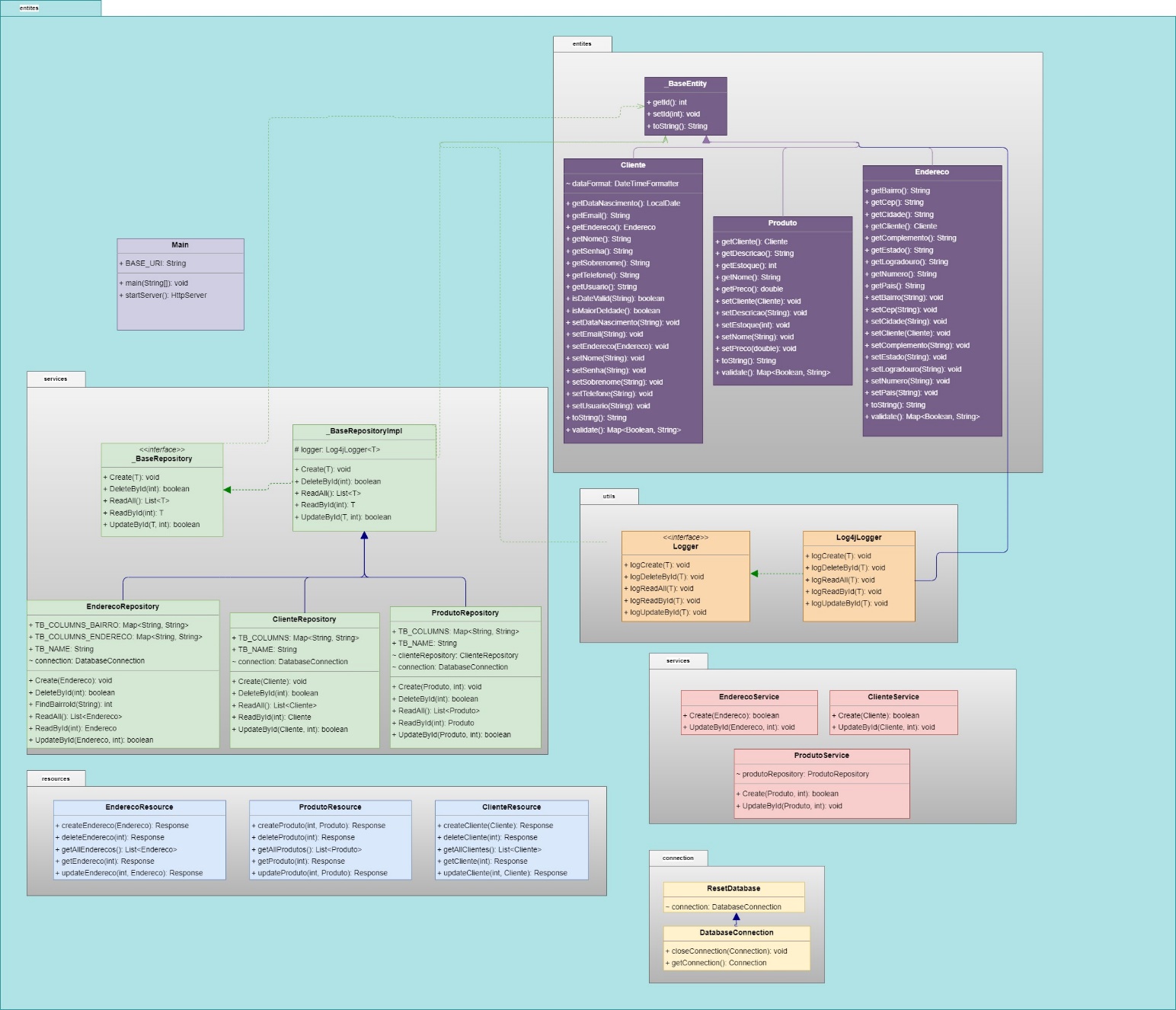
Um arquivo PDF dessa modelagem acompanhará este, para melhor visualização.





# Diagrama de classes

Um arquivo PDF desse diagrama acompanhará este, para melhor visualização.



# Tabela de *endpoints*

O arquivo para teste no POSTMAN está dentro do projeto, na pasta “Postman”.

| **URL** | **Verbo HTTP** | **Descrição** | **Códigos de Status** |
| --- | --- | --- | --- |
| **/salesforcedirections/cliente** | GET | Retorna a lista de clientes | 200 (OK), 500 (Erro Interno do Servidor) |
| **/salesforcedirections/cliente/{id}** | GET | Retorna um cliente específico | 200 (OK), 404 (Não Encontrado), 500 (Erro Interno do Servidor) |
| **/salesforcedirections/cliente** | POST | Cria um novo cliente | 201 (Criado), 400 (Requisição Inválida), 500 (Erro Interno do Servidor) |
| **/salesforcedirections/cliente/{id}** | PUT | Atualiza um cliente existente | 200 (OK), 400 (Requisição Inválida), 404 (Não Encontrado), 500 (Erro Interno do Servidor) |
| **/salesforcedirections/cliente/{id}** | DELETE | Deleta um cliente | 204 (Sem Conteúdo), 404 (Não Encontrado), 500 (Erro Interno do Servidor) |
| **/salesforcedirections/endereco** | GET | Retorna a lista de endereços | 200 (OK), 500 (Erro Interno do Servidor) |
| **/salesforcedirections/endereco/{id}** | GET | Retorna um endereço específico | 200 (OK), 404 (Não Encontrado), 500 (Erro Interno do Servidor) |
| **/salesforcedirections/endereco** | POST | Cria um novo endereço | 201 (Criado), 400 (Requisição Inválida), 500 (Erro Interno do Servidor) |
| **/salesforcedirections/endereco/{id}** | PUT | Atualiza um endereço existente | 200 (OK), 400 (Requisição Inválida), 404 (Não Encontrado), 500 (Erro Interno do Servidor) |
| **/salesforcedirections/endereco/{id}** | DELETE | Deleta um endereço | 204 (Sem Conteúdo), 404 (Não Encontrado), 500 (Erro Interno do Servidor) |
| **/salesforcedirections/produto** | GET | Retorna a lista de produtos | 200 (OK), 500 (Erro Interno do Servidor) |
| **/salesforcedirections/produto/{id}** | GET | Retorna um produto específico | 200 (OK), 404 (Não Encontrado), 500 (Erro Interno do Servidor) |
| **/salesforcedirections/produto/{clienteId}** | POST | Cria um novo produto para um cliente | 201 (Criado), 400 (Requisição Inválida), 500 (Erro Interno do Servidor) |
| **/salesforcedirections/produto/{id}** | PUT | Atualiza um produto existente | 200 (OK), 400 (Requisição Inválida), 404 (Não Encontrado), 500 (Erro Interno do Servidor) |
| **/salesforcedirections/produto/{id}** | DELETE | Deleta um produto | 204 (Sem Conteúdo), 404 (Não Encontrado), 500 (Erro Interno do Servidor) |

# Executar aplicação

Para executar o projeto, siga estas etapas:

1. Abra o projeto no IntelliJ IDEA.
2. Atualize as configurações do banco de dados no arquivo DatabaseConnection.java.
3. Execute o arquivo test.java.

Para informações mais detalhadas sobre o projeto, leia o README.md localizado na pasta raiz.

# Link do repositório

O projeto final de Java está no diretório “Sprint 4/DOMAIN DRIVEN DESIGN/ Challenge Sprint 4”.

* LINK: <https://github.com/Mauricio-Pereira/Challenge-Fiap>