

Gestão de Contatos (Comércio S.A)

Descrição

Este projeto é uma aplicação simples para a gestão de contatos utilizando Java com Spring Boot no back-end e HTML, CSS e JavaScript no front-end. O sistema permite o cadastro, atualização, exclusão e listagem de clientes, com seus respectivos contatos (como telefone e e-mail). A comunicação entre o frontend e o backend é feita através de uma API REST.

Funcionalidades

- **Cadastro de Clientes e Contatos:** Permite cadastrar um cliente com seu nome, CPF, data de nascimento, endereço e seus contatos (tipo e valor).
- **Edição de Clientes:** Permite editar as informações do cliente e seus contatos.
- **Exclusão de Clientes:** Permite excluir um cliente e seus respectivos contatos.
- **Listagem de Clientes:** Exibe uma tabela com todos os clientes cadastrados e seus contatos.

Estrutura do Projeto

Frontend (HTML, CSS, JavaScript)

- **index.html:** A página principal onde o formulário de cadastro e a lista de contatos são exibidos.
- **style.css:** Estilos para tornar a interface do usuário mais amigável e organizada.
- **script.js:** Lógica para interagir com a API backend (fazendo requisições para cadastrar, editar, excluir e listar clientes).

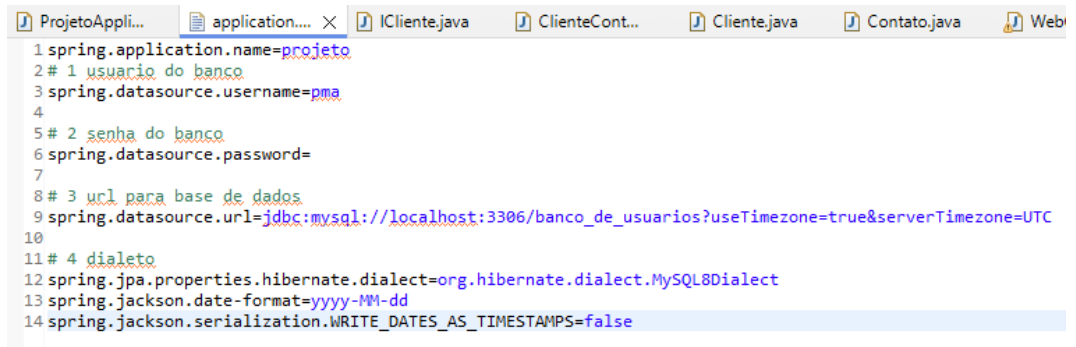
Backend (Java Spring Boot)

- **Modelo de Cliente e Contato:** Entidades Cliente e Contato mapeadas para as tabelas no banco de dados.
- **Controller ClienteController:** Controlador que lida com as requisições da API (POST, GET, PUT, DELETE) para a criação, listagem, atualização e exclusão de clientes.
- **Repositório:** Interface ICliente, que estende CrudRepository, para interação com o banco de dados.

Como Rodar o Projeto

1. Backend:

- Clone o repositório e navegue até a pasta do backend.
- Certifique-se de ter o Java 11+ instalado.
- O servidor será iniciado em <http://localhost:8080>.
- Em src/main/resources vá em application.properties e configure o caminho (endereço) que corresponde ao banco.

A screenshot of an IDE showing the application.properties file. The file contains 14 lines of configuration for a Spring application. The tabs at the top show 'ProjetoAppli...', 'application....', 'ICliente.java', 'ClienteCont...', 'Cliente.java', 'Contato.java', and 'Web...'. The code in the editor is as follows:

```
1 spring.application.name=projeto
2 # 1 usuario do banco
3 spring.datasource.username=pma
4
5 # 2 senha do banco
6 spring.datasource.password=
7
8 # 3 url para base de dados
9 spring.datasource.url=jdbc:mysql://localhost:3306/banco_de_usuarios?useTimezone=true&serverTimezone=UTC
10
11 # 4 dialeto
12 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
13 spring.jackson.date-format=yyyy-MM-dd
14 spring.jackson.serialization.WRITE_DATES_AS_TIMESTAMPS=false
```

- Inicie o projeto

2. Frontend:

- Navegue até a pasta do frontend (onde o index.html está localizado).
- Abra o arquivo index.html em seu navegador.
- A interface será carregada e você poderá interagir com a API para cadastrar e listar contatos.

3. Banco de dados:

Execute o ddl no seu MySQL Workbench:

```
CREATE DATABASE banco_de_usuarios;
USE banco_de_usuarios;
CREATE TABLE cliente (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    cpf VARCHAR(14) NOT NULL UNIQUE,
    data_nascimento DATE NOT NULL,
    endereco VARCHAR(255)
);

CREATE TABLE contato (
    id INT AUTO_INCREMENT PRIMARY KEY,
    cliente_id INT NOT NULL,
    tipo ENUM('Telefone', 'E-mail') NOT NULL,
    valor VARCHAR(100) NOT NULL,
    observacao VARCHAR(255),
    FOREIGN KEY (cliente_id) REFERENCES cliente(id) ON DELETE CASCADE
);
```

Tecnologias Utilizadas

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** Java, Spring Boot
- **Banco de Dados:** MySQL)
- **API:** Spring Boot

Endpoints da API

GET /clientes

Retorna todos os clientes cadastrados com seus respectivos contatos.

POST /clientes

Cadastra um novo cliente com contatos. O corpo da requisição deve ser um JSON no seguinte formato:

```
json
CopiarEditar
{
  "nome": "Nome do Cliente",
  "cpf": "123.456.789-00",
  "dataNascimento": "2000-01-01",
  "endereco": "Rua X, 123",
  "contatos": [
    {
      "tipo": "Telefone",
      "valor": "(11) 1234-5678",
      "observacao": "Contato principal"
    }
  ]
}
```

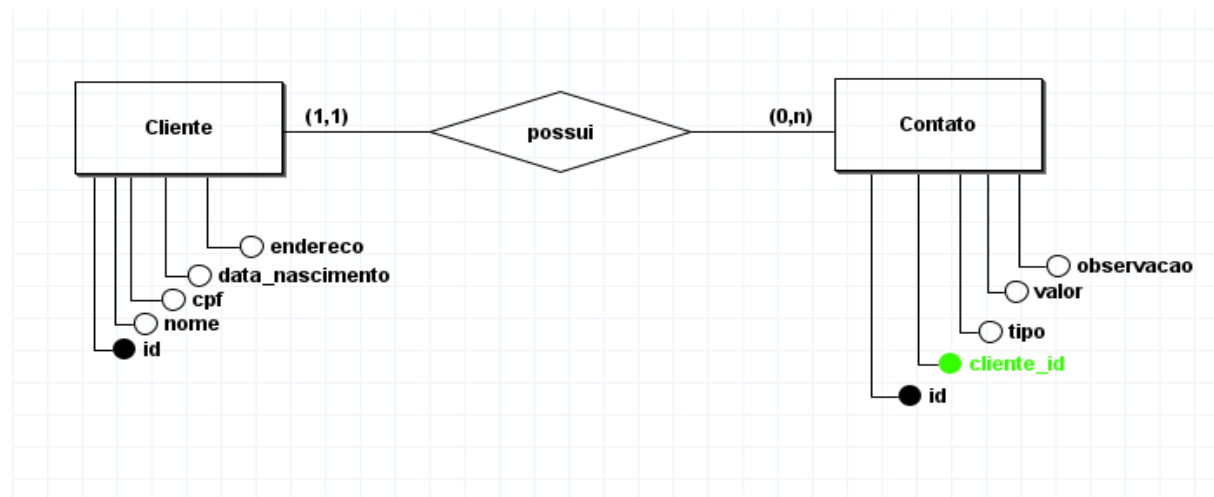
PUT /clientes/{id}

Atualiza os dados de um cliente e seus contatos.

DELETE /clientes/{id}

Exclui um cliente e seus contatos.

Diagrama de Entidade e Relacionamento:



DDL:

```
CREATE DATABASE banco_de_usuarios;
USE banco_de_usuarios;
CREATE TABLE cliente (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    cpf VARCHAR(14) NOT NULL UNIQUE,
    data_nascimento DATE NOT NULL,
    endereco VARCHAR(255)
);

CREATE TABLE contato (
    id INT AUTO_INCREMENT PRIMARY KEY,
    cliente_id INT NOT NULL,
    tipo ENUM('Telefone', 'E-mail') NOT NULL,
    valor VARCHAR(100) NOT NULL,
    observacao VARCHAR(255),
    FOREIGN KEY (cliente_id) REFERENCES cliente(id) ON DELETE CASCADE
);
```