- Crie uma classe Contato.java que tenha os atributos CPF, Nome e Idade.

- Crie nesta classe os métodos get e set para CPF, Nome e Idade.

- Crie um construtor que receba como parametro o CPF o nome e a idade e faça a atribuição dos valores no objeto.

- Crie o método toString para realizar a conversão dos dados do objeto para um texto simples contendo CPF: xxxxxx, Nome: xxxx, Idade

- Crie uma classe App.java que tenha um ArrayList para armazenar os objetos Contatos criados.

- Crie as rotas abaixo:

Endereço: /contato/criar Método: Post

Endereço: /contato/listarTodos Método: Get

Endereço: /contato/obterPorCpf/:cpf Método: Get

Endereço: /contato/obterPorIdade/:iddade Método: Get

Endereço: /contato/atualizar/:cpf Método: Put

Endereço: /contato/excluir/:cpf Método: Delete

Use como exemplo o código abaixo:

**App.java**

package br.com.mesttra;

import java.util.ArrayList;

import spark.Spark;

import spark.Request;

import spark.Response;

import spark.Route;

public class App {

    // Aqui definimos a lista contatos como variaveis globais

    // pois já estamos utilizando a orientação a objetos para ter uma melhor

    // separação não sendo necessário termos tanta proteção às variaveis do

    // programa principal

    private static ArrayList<Pessoa> listaContatos = new ArrayList<Pessoa>();

    public static void main(String[] args) {

        Spark.port(8080);

        Spark.get("/usuario/listar", listarUsuario());

        Spark.post("/usuario/cadastrar", incluirContato());

        Spark.get("/usuario/cadastrar/:nome/:email/:telefone", incluirContatoURL());

        Spark.delete("/usuario/deletar/:codigo/:nome", excluirContato());

        Spark.post("/usuario/cadastrar/:nome", incluirContatoURL2());

        //Spark.delete(/caminho/da/rota/:codigo,metodo); //rota para excluir um valor

        //Spark.put(/caminho/da/rota/:codigo,metodo);   //rota para atualizar um valor

    }

    private static Route incluirContato() {

        return new Route() {

            @Override

            public Object handle(Request request, Response response) throws Exception {

                String name = request.queryParams("nome");

                String email = request.queryParams("email");

                String telefone = request.queryParams("telefone");

                Pessoa novaPessoa = new Pessoa(name, telefone, email);

                listaContatos.add(novaPessoa);

                response.status(201);

                return "Contato incluído com sucesso!";

            }

        };

    }

    private static Route incluirContatoURL() {

        return new Route() {

            @Override

            public Object handle(Request request, Response response) throws Exception {

                String name = request.params("nome");

                String email = request.params("email");

                String telefone = request.params("telefone");

                Pessoa novaPessoa = new Pessoa(name, telefone, email);

                listaContatos.add(novaPessoa);

                response.status(201);

                return "Contato incluído com sucesso!";

            }

        };

    }

    private static Route incluirContatoURL2() {

        return new Route() {

            @Override

            public Object handle(Request request, Response response) throws Exception {

                String name = request.params("nome");

                String email = request.queryParams("email");

                String telefone = request.queryParams("telefone");

                Pessoa novaPessoa = new Pessoa(name, telefone, email);

                listaContatos.add(novaPessoa);

                response.status(201);

                return "Contato incluído com sucesso!";

            }

        };

    }

    private static Route listarUsuario() {

        return new Route() {

            @Override

            public Object handle(Request request, Response response) throws Exception {

                String resposta = "";

                // metodo isEmpty verifica se a lista esta vazia

                if (listaContatos.isEmpty()) {

                    resposta += "Nenhum contato cadastrado.";

                } else {

                    resposta += "\n--- Lista de Contatos ---";

                    for (Pessoa pessoa : listaContatos) {

                        resposta += "\n" + pessoa.toString();

                    }

                }

                return resposta;

            }

        };

    }

    // Método para lidar com a rota de excluir usuário

    private static Route excluirContato() {

        return new Route() {

            @Override

            public Object handle(Request request, Response response) throws Exception {

                int id = Integer.parseInt(request.params(":codigo"));

                if (listaContatos.isEmpty()) {

                    response.status(404); // 404 Not Found

                    return "Não existem contatos na base.";

                } else {

                    for (Pessoa pessoa : listaContatos) {

                        if (pessoa.getId() == id){

                            listaContatos.remove(pessoa);

                            response.status(200);

                            return "Usuario com " + id + " foi excluido com sucesso!";

                        }

                    }

                    response.status(404); // 404 Not Found

                    return "Contato com id: " + id + "não encontrado.";

                }

            }

        };

    }

**Pessoa.java**

package br.com.mesttra;

public class Pessoa {

    public static int contador;

    private int id;

    private String nome;

    private String telefone;

    private String email;

    //construtor da classe pessoa

    public Pessoa(String nome, String telefone, String email) {

        contador++;

        this.id = contador;

        setNome(nome);

        setTelefone(telefone);

        setEmail(email);

    }

    public int getId() {

        return id;

    }

    public void setNome(String nome) {

        //aqui implementariamos as validações necessárias

        //antes de inserir o nome

        this.nome = nome;

    }

    public String getNome() {

        return nome;

    }

    public void setTelefone(String telefone) {

        //aqui implementariamos as validações necessárias

        //antes de inserir o telefone

        this.telefone = telefone;

    }

    public String getTelefone() {

        return telefone;

    }

    public void setEmail(String email) {

        //aqui implementariamos as validações necessárias

        //antes de inserir o email

        this.email = email;

    }

    public String getEmail() {

        return email;

    }

    //cria o método to string utilizado para converter o objeto para string

    //quando for necessário imprimir os dados do objeto na tela por outra parte do nosso

    //programa

    @Override

    public String toString() {

        return "ID: " + id + ", Nome: " + nome + ", Telefone: " + telefone + ", Email: " + email;

    }

}

**Arquivo pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>br.com.mesttra</groupId>

    <artifactId>api</artifactId>

    <version>1.0-SNAPSHOT</version>

    <properties>

        <maven.compiler.source>17</maven.compiler.source>

        <maven.compiler.target>17</maven.compiler.target>

    </properties>

    <dependencies>

        <dependency>

            <groupId>com.sparkjava</groupId>

            <artifactId>spark-core</artifactId>

            <version>2.9.4</version>

        </dependency>

    </dependencies>

</project>