### Exercício 15.2 – Sistema de Gestão de Contactos com Vaadin Flow

Este exercício apresenta a implementação de uma aplicação Vaadin Flow com funcionalidades para gerir contactos, incluindo listagem, adição, edição e eliminação.

### Parte 1: Configuração do Projeto no Spring Initializr

1. **Acessar o Spring Initializr:**
   * Vá ao site [Spring Initializr](https://start.spring.io/).
2. **Configurar o projeto:**
   * **Project:** Maven.
   * **Language:** Java.
   * **Spring Boot:** Última versão estável.
   * **Group:** com.example.
   * **Artifact:** contactmanager.
   * **Name:** contactmanager.
   * **Package Name:** com.example.contactmanager.
   * **Packaging:** Jar.
   * **Java Version:** 21.
3. **Adicionar dependências:**
   * **Spring Web:** Para suporte ao backend.
   * **Vaadin Spring Boot Starter:** Para a interface gráfica com Vaadin Flow.
4. **Gerar o projeto:**
   * Clique em **Generate** para descarregar o projeto como um ficheiro ZIP.
5. **Abrir o projeto no VS Code:**
   * Extraia o ficheiro ZIP.
   * No VS Code, clique em **File > Open Folder** e selecione a pasta do projeto.

### Parte 2: Estrutura do Projeto

1. **Criar as pastas necessárias:**
   * Em src/main/java/com/example/contactmanager, crie:
     + entity para a classe Contact.
     + service para a classe ContactService.
     + views para a classe ContactView.
2. A estrutura deverá ser:

src/  
├── main/  
│ ├── java/com/example/contactmanager/  
│ │ ├── ContactManagerApplication.java  
│ │ ├── entity/  
│ │ │ └── Contact.java  
│ │ ├── service/  
│ │ │ └── ContactService.java  
│ │ └── views/  
│ │ └── ContactView.java  
│ └── resources/  
│ └── application.properties

### Parte 3: Implementação do Sistema de Contactos

1. **Classe Principal (ContactManagerApplication):**

package com.example.contactmanager;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class ContactManagerApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(ContactManagerApplication.class, args);  
 }  
}

1. **Entidade Contact:**

package com.example.contactmanager.entity;  
  
public class Contact {  
 private Long id;  
 private String name;  
 private String email;  
  
 public Contact() {}  
  
 public Contact(Long id, String name, String email) {  
 this.id = id;  
 this.name = name;  
 this.email = email;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
}

1. **Serviço ContactService:**

package com.example.contactmanager.service;  
  
import com.example.contactmanager.entity.Contact;  
import java.util.ArrayList;  
import java.util.List;  
  
public class ContactService {  
 private final List<Contact> contacts = new ArrayList<>();  
  
 public ContactService() {  
 contacts.add(new Contact(1L, "João Silva", "joao.silva@example.com"));  
 contacts.add(new Contact(2L, "Maria Oliveira", "maria.oliveira@example.com"));  
 }  
  
 public List<Contact> getAllContacts() {  
 return contacts;  
 }  
  
 public void saveContact(Contact contact) {  
 if (contact.getId() == null) {  
 contact.setId((long) (contacts.size() + 1));  
 }  
 contacts.add(contact);  
 }  
  
 public void deleteContact(Contact contact) {  
 contacts.remove(contact);  
 }  
}

1. **Vista de Contactos (ContactView):**

package com.example.contactmanager.views;  
  
import com.example.contactmanager.entity.Contact;  
import com.example.contactmanager.service.ContactService;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.formlayout.FormLayout;  
import com.vaadin.flow.component.grid.Grid;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.EmailField;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.Route;  
  
@Route("")  
public class ContactView extends VerticalLayout {  
  
 private final ContactService contactService = new ContactService();  
  
 private final Grid<Contact> contactGrid = new Grid<>(Contact.class, false);  
 private final TextField nameField = new TextField("Nome");  
 private final EmailField emailField = new EmailField("Email");  
 private final Button saveButton = new Button("Salvar");  
 private final Button clearButton = new Button("Limpar");  
  
 private Contact selectedContact = null;  
  
 public ContactView() {  
 setSizeFull();  
  
 // Configuração da grelha  
 contactGrid.addColumn(Contact::getName).setHeader("Nome");  
 contactGrid.addColumn(Contact::getEmail).setHeader("Email");  
 contactGrid.addComponentColumn(contact -> {  
 Button deleteButton = new Button("Eliminar", click -> {  
 contactService.deleteContact(contact);  
 updateGrid();  
 Notification.show("Contacto eliminado!");  
 });  
 return deleteButton;  
 });  
 contactGrid.asSingleSelect().addValueChangeListener(event -> editContact(event.getValue()));  
  
 // Configuração do formulário  
 FormLayout formLayout = new FormLayout();  
 formLayout.add(nameField, emailField);  
  
 saveButton.addClickListener(event -> saveContact());  
 clearButton.addClickListener(event -> clearForm());  
  
 HorizontalLayout buttonsLayout = new HorizontalLayout(saveButton, clearButton);  
  
 // Adicionar componentes ao layout principal  
 add(contactGrid, formLayout, buttonsLayout);  
  
 // Atualizar a grelha inicialmente  
 updateGrid();  
 }  
  
 private void saveContact() {  
 if (nameField.isEmpty() || emailField.isEmpty()) {  
 Notification.show("Todos os campos são obrigatórios!", 3000, Notification.Position.MIDDLE);  
 return;  
 }  
  
 if (selectedContact == null) {  
 Contact newContact = new Contact(null, nameField.getValue(), emailField.getValue());  
 contactService.saveContact(newContact);  
 Notification.show("Contacto adicionado!");  
 } else {  
 selectedContact.setName(nameField.getValue());  
 selectedContact.setEmail(emailField.getValue());  
 Notification.show("Contacto atualizado!");  
 }  
  
 updateGrid();  
 clearForm();  
 }  
  
 private void clearForm() {  
 selectedContact = null;  
 nameField.clear();  
 emailField.clear();  
 }  
  
 private void editContact(Contact contact) {  
 if (contact != null) {  
 selectedContact = contact;  
 nameField.setValue(contact.getName());  
 emailField.setValue(contact.getEmail());  
 } else {  
 clearForm();  
 }  
 }  
  
 private void updateGrid() {  
 contactGrid.setItems(contactService.getAllContacts());  
 }  
}

### Parte 4: Configuração no application.properties

server.port=8080  
vaadin.urlMapping=/\*

### Parte 5: Testar a Aplicação

1. **Iniciar o projeto:**
   * Use o comando mvn spring-boot:run.
2. **Testar as funcionalidades:**
   * Abra <http://localhost:8080> no navegador.
   * Liste, adicione, edite e elimine contactos conforme necessário.