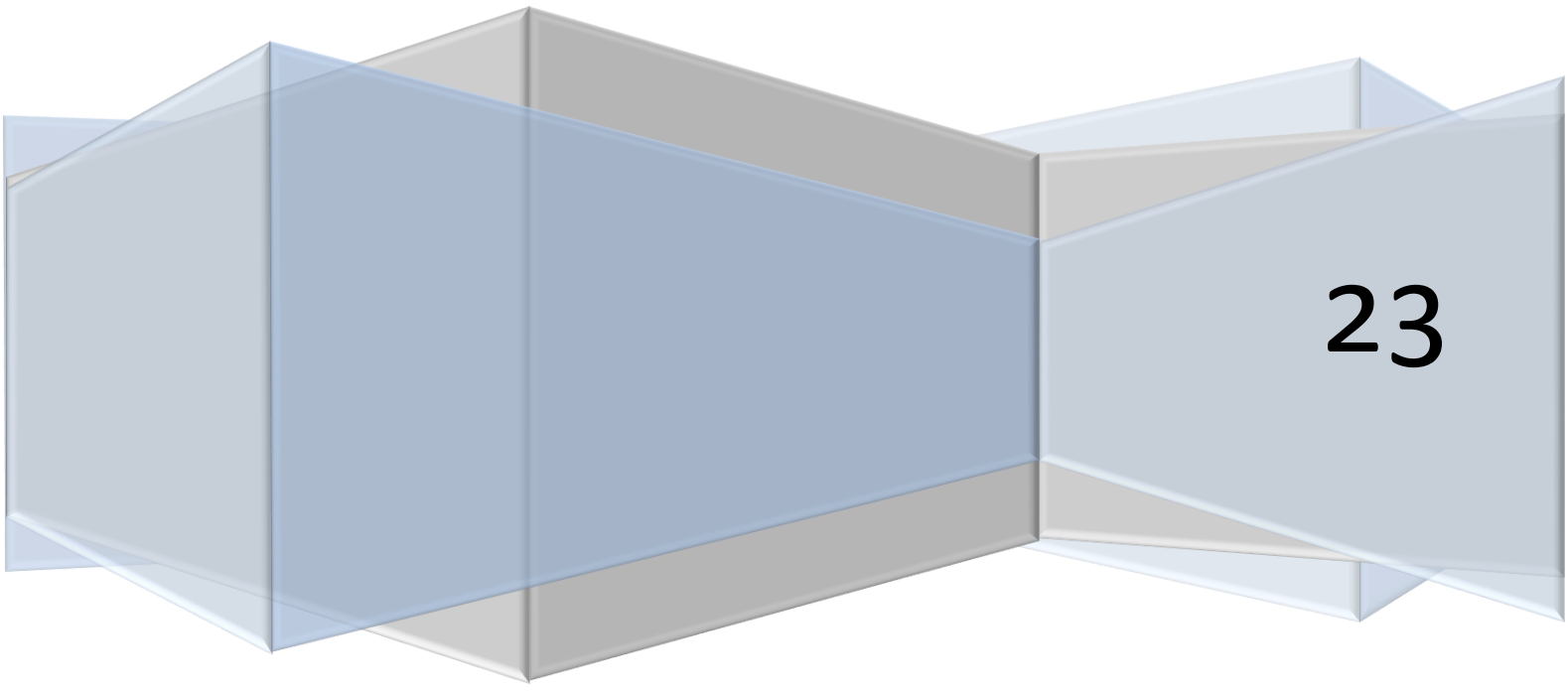
Colegio Salesianos Zaragza

Practica Sprinboot: VIPZone club

Creación de una aplicación que obtiene y muestra datos de una BBDD y realiza operaciones sobre estos datos.

Luis Fernando Moreno



Introducción

La práctica consiste en crear una aplicación en SpringBoot que contenga una conexión a una BBDD. La aplicación debe poder mostrar al usuario los datos de la base de datos y realizar operaciones básicas sobre la BBDD, como añadir, editar y borrar registros. Por último se busca que en la práctica se utilicen los fragments de thymeleaf.

Desarrollo de la práctica

Para la práctica se crean varios ficheros java y html, a continuación se explica el papel de cada uno:

Entidades

Son las representaciones de las tablas en la base de datos, en Springboot se tratan como clases y objetos.

Persona.java

Es la clase que contiene la estructura de la tabla persona “persona” en la BBDD, y es el tipo de objeto que se usa para guardar y cargar los datos de cada registro en la tabla

```java
@Entity
@Table(name = "persona")
public class Persona implements Serializable{

private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "idPersona")
    private Long id;
    @Column(name = "Nombre")
    private String nombre;
    @Column(name = "Apellidos")
    private String apellidos;
    @Column(name = "FechaNac")
    private LocalDate fechaNac;
    @Column(name = "Telefono")
    private String telefono;
    @Column(name = "Provincia")
    private String provincia;
    @Column(name = "Profesion")
    private String profesion;

    public Persona() {
    }

    public Persona(Long id, String nombre, String apellidos, LocalDate fechaNac, String telefono, String provincia,
            String profesion) {
        super();
        this.id = id;
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.fechaNac = fechaNac;
        this.telefono = telefono;
        this.provincia = provincia;
        this.profesion = profesion;
```

VipZone.java

Contiene la estructura de la tabla vip_zone

```java
@Entity
@Table(name = "vipzone")
public class VipZone {

    private static final long serialVersionUID = 3233149207833106460L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false, columnDefinition = "INT(11)")
    private Long id;

    @Column(name = "zona", length = 100)
    private String zona;

    @Column(name = "numPersonas", columnDefinition = "INT(3)")
    private int numPersonas;

    @Column(name = "gasto")
    private double gasto;

    @Column(name = "disponibilidad", columnDefinition = "INT(1)")
    private int disponibilidad;

    public VipZone() {
        super();
    }

    public VipZone(Long id, String zona, int numPersonas, double gasto, int disponibilidad) {
        super();
        this.id = id;
        this.zona = zona;
        this.numPersonas = numPersonas;
        this.gasto = gasto;
        this.disponibilidad = disponibilidad;
    }
```

ClienteVip.java

Contiene la estructura de la tabla cliente_vip

Repositorios

Clases que contienen las herramientas necesarias para comunicarse con la base de datos. Heredan de JpaRepository y son las clases que nos permite conectar y operar sobre la BBDD.

IPersonaDAO.java

Clase que nos comunica con la tabla persona

```java
package com.example.di.blackListClub.dao;

import org.springframework.data.jpa.repository.JpaRepository;

@Repository
public interface IPersonaDAO extends JpaRepository<Persona, Long>{
    @Query("select p from Persona p where p.nombre like ?1 ")
    public Persona findByName(String name);

}
```

IVipZoneDAO.java

Clase que nos comunica con la tabla vip_zone

```java
package com.example.di.blackListClub.dao;

import org.springframework.data.jpa.repository.JpaRepository;

    @Repository
    public interface IVipZoneDAO extends JpaRepository<VipZone, Long>{
        @Query("select vz from VipZone vz where vz.zona like ?1 ")
        public VipZone findByName(String zona);

    }

```

IClienteVipDAO.java

Clase que nos comunica con la tabla cliente_vip

```java
package com.example.di.blackListClub.dao;

import org.springframework.data.jpa.repository.JpaRepository;

public interface IClienteVipDAO extends JpaRepository<ClienteVip, Long>{
    @Query("select cv from ClienteVip cv where cv.dni like ?1 ")
    public ClienteVip findByDni(String dni);

}

```

Interfaces de servicio

Estas interfaces contienen los métodos que el servicio necesita tener

```java
package com.example.di.blackListClub.serv

import java.util.List;

public interface IPersonaService {

    public List<Persona> listAll();
    public void save(Persona persona);
    public Persona get(long id);
    public void delete(long id);
}

```

```java
package com.example.di.blackListClub.se

import java.util.List;

public interface IVipZoneService {

    public List<VipZone> listAll();
    public void save(VipZone vipZone);
    public VipZone get(long id);
    public void delete(long id);

}
```

```java
package com.example.di.blackListClub.service

import java.util.List;

public interface IClienteVipService {

    public List<ClienteVip> listAll();
    public void save(ClienteVip clienteVip);
    public ClienteVip get(long id);
    public void delete(long id);

```

Implementación de servicios

Son clases a las que se les inyecta un repositorio, realizan operaciones sobre la base de datos. Extienden de las interfaces servicio.

```java
@Service
public class PersonaServiceImpl implements IPersonaService{

    @Autowired
    private IPersonaDAO personaDao;

    @Override
    public List<Persona> listAll() {
        return personaDao.findAll();
    }

    @Override
    public void save(Persona persona) {
        personaDao.save(persona);
    }

    @Override
    public Persona get(long id) {
        return personaDao.findById(id).get();
    }

    @Override
    public void delete(long id) {
        personaDao.deleteById(id);
    }

}
```

```java
@Service
public class VipZoneServiceImpl implements IVipZoneService{

    @Autowired
    private IVipZoneDAO vipZoneDao;

    @Override
    public List<VipZone> listAll() {
        return vipZoneDao.findAll();
    }

    @Override
    public void save(VipZone vipZone) {
        vipZoneDao.save(vipZone);
    }

    @Override
    public VipZone get(long id) {
        return vipZoneDao.findById(id).get();
    }

    @Override
    public void delete(long id) {
        vipZoneDao.deleteById(id);
    }
```

```java
@Service
public class ClienteVipServiceImpl implements IClienteVipService{

    @Autowired
    private IClienteVipDAO clienteVipDAO;

    @Override
    public List<ClienteVip> listAll() {
        return clienteVipDAO.findAll();
    }

    @Override
    public void save(ClienteVip clienteVip) {
        clienteVipDAO.save(clienteVip);
    }

    @Override
    public ClienteVip get(long id) {
        return clienteVipDAO.findById(id).get();
    }

    @Override
    public void delete(long id) {
        clienteVipDAO.deleteById(id);
    }

}
```

Controladores

Son clases que contienen los controladores para dar funcionalidad a las opciones que se le muestran al usuario en la aplicación

BlackListController.java

```java
@Controller
public class BlackList1Controller {

    @Autowired
    private IPersonaService personaService;

    @RequestMapping("/")
    public String viewHomePage(Model model) {
        List<Persona> listPersonas = personaService.listAll();
        model.addAttribute("listaPersonas", listPersonas);

        return "index";
    }

    @RequestMapping("/new")
    public String showNewPersonPage(Model model) {
        Persona persona = new Persona();
        model.addAttribute("persona", persona);

        return "new_contact";
    }

    @RequestMapping(value = "/save", method = RequestMethod.POST)
    public String savePerson(@ModelAttribute("persona") Persona persona) {
        personaService.save(persona);

        return "redirect:/";
    }

    @RequestMapping("/edit/{id}")
    public ModelAndView showEditPersonPage(@PathVariable(name = "id") int id) {
        ModelAndView mav = new ModelAndView("edit_contact");
        Persona persona = personaService.get(id);
        mav.addObject("persona", persona);

        return mav;
    }

    @RequestMapping("/delete/{id}")
```

VipZoneController.java

```java
@Controller
public class VipZoneController {

    @Autowired
    private IVipZoneService vipZoneService;

    @RequestMapping("/vipzones")
    public String viewVipZonesPage(Model model) {
        List<VipZone> listVipzones = vipZoneService.listAll();
        model.addAttribute("listaVipzones", listVipzones);

        return "vipzones";
    }

    @RequestMapping("/newVipZone")
    public String showNewVipZonePage(Model model) {
        VipZone vipZone = new VipZone();
        model.addAttribute("vipZone", vipZone);

        return "new_vipZone";
    }

    @RequestMapping(value = "/saveVipZone", method = RequestMethod.POST)
    public String saveVipZone(@ModelAttribute("vipZone") VipZone vipZone) {
        vipZoneService.save(vipZone);

        return "redirect:/vipzones";
    }

    @RequestMapping("/editVipzone/{id}")
    public ModelAndView showEditVipZonePage(@PathVariable(name = "id") int id) {
        ModelAndView mav = new ModelAndView("edit_vipZone");
        VipZone vipZone = vipZoneService.get(id);
        mav.addObject("vipZone", vipZone);

        return mav;
    }

    @RequestMapping("/deleteVipzone/{id}")
```

ClienteVipController.java

```java
@Controller
public class ClienteVipController {

    @Autowired
    private IClienteVipService clienteVipService;
    private IVipZoneService vipZoneService;

    @RequestMapping("/clientesVipzone/{vipzoneID}")
    public String viewClientesVipZonePage(Model model, @PathVariable(name = "vipzoneID") int vipzoneID) {
        List<ClienteVip> listClientesVip = clienteVipService.listAll();
        model.addAttribute("listaClientesVip", listClientesVip);

        VipZone vipZone = vipZoneService.get(vipzoneID);
        model.addAttribute("vipZone", vipZone);

        return "clientesVipZone";
    }

    @RequestMapping("/newVipCliente")
    public String showNewVipClientePage(Model model) {
        ClienteVip clienteVip = new ClienteVip();
        model.addAttribute("clienteVip", clienteVip);

        return "new_ClienteVip";
    }

    @RequestMapping(value = "/saveClienteVip", method = RequestMethod.POST)
    public String saveVipZone(@ModelAttribute("clienteVip") ClienteVip clienteVip,
            @ModelAttribute("vipZone") VipZone vipZone) {
        clienteVipService.save(clienteVip);

        return "redirect:/clientesVipzone/" + vipZone.getId();
    }

    @RequestMapping("/editVipcliente/{id}")
    public ModelAndView showEditVipZonePage(@PathVariable(name = "id") int id) {
        ModelAndView mav = new ModelAndView("edit_clienteVip");
        ClienteVip clienteVip = clienteVipService.get(id);
        mav.addObject("clienteVip", clienteVip);
```

Layout

layout.html es una capa base que se usa como marco para dar uniformidad a las diferentes vistas de la aplicación.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head th:fragment="head">
    <meta charset="UTF-8">
    <title th:text="${titulo}"></title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" in
    <link rel="stylesheet" th:ref="@{/css/estilos.css}">
</head>
<body>
    <header th:fragment="header">
        <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
          <div class="container-fluid">
            <a class="navbar-brand" href="#">
                <img th:src="@{/img/bootstrap-logo.svg}" width="30" height="30" alt="">
              </a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSu
              <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarSupportedContent">
              <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                <li class="nav-item">
                  <a class="nav-link active" aria-current="page" href="#">Home</a>
                </li>
                <li class="nav-item">
                  <a class="nav-link" th:href="@{/}">Contactos</a>
                </li>
                <li class="nav-item">
                  <a class="nav-link" th:href="@{/vipzones}">Vip zones</a>
                </li>
              </ul>
              <form class="d-flex">
                <input class="form-control me-2" type="search" placeholder="Buscar" aria-label="Buscar">
                <button class="btn btn-outline-success" type="submit">Buscar</button>
              </form>
            </div>
          </div>
        </nav>

    </header>
```

Index

index.html es la página principal de la aplicación. Utiliza un fragment para obtener de otro fichero, la tabla html donde se muestran el total de elementos de la tabla personas de la BBDD, y da opciones básicas al usuario para editar, crear y eliminar.

vipzones.html y clientesVipZone.html son similares al index.

show_contacts.html

Este fichero básicamente se usa como un almacén para almacenar los fragments que serán llamados desde otros ficheros html.

```
<table border="1" cellpadding="10" th:fragment="table_contacts1">
    <thead>
        <tr>
            <th>ID Contacto</th>
            <th>Nombre</th>
            <th>Apellidos</th>
            <th>Fecha de nacimiento</th>
            <th>Telefono</th>
            <th>Provincia</th>
            <th>Profesion</th>
        </tr>
    </thead>
    <tbody>
        <tr th:each="persona : ${listaPersonas}">
            <td th:text="${persona.id}">Contacto ID</td>
            <td th:text="${persona.nombre}">Nombre</td>
            <td th:text="${persona.apellidos}">Apellidos</td>
            <td th:text="${persona.fechaNac}">Fecha de nacimiento</td>
            <td th:text="${persona.telefono}">Telefono</td>
            <td th:text="${persona.provincia}">Provincia</td>
            <td th:text="${persona.profesion}">Profesion</td>
            <td>
                <a th:href="@{'/edit/' + ${persona.id}}">Editar</a>
                   
                <a th:href="@{'/delete/' + ${persona.id}}">Borrar</a>
```

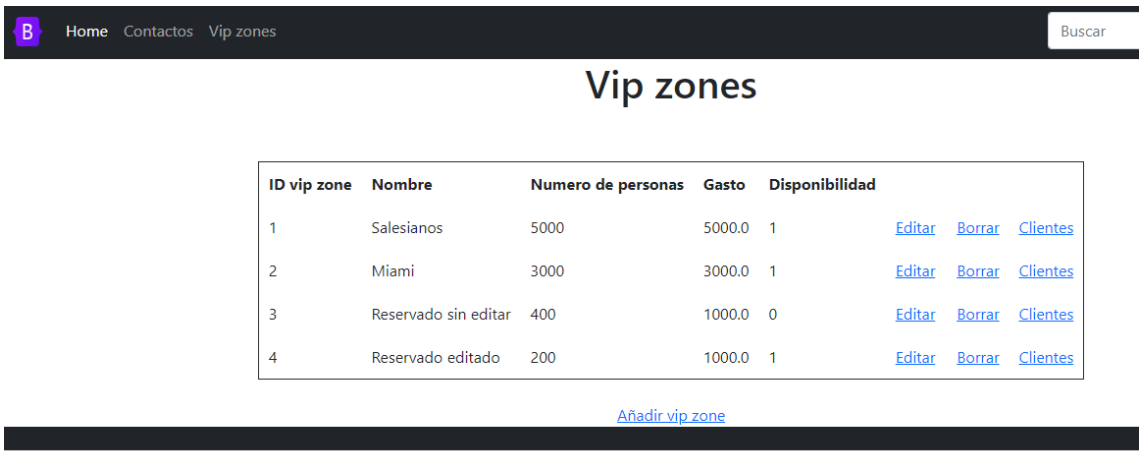
También se crean páginas para crear y editar, cada una de las entidades.

edit_clientes.html, edit_contacts.html, edit_vipZone.html

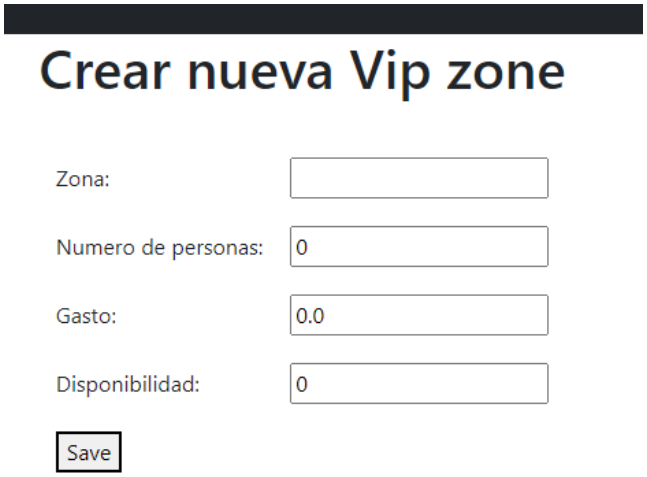
Son formularios que recogen la información de un objeto que se envía desde el modelo, y los muestran en textboxes para facilitarle al cliente su edición

Funcionamiento:

Cuando la aplicación se inicia, carga el index, que nos muestra la lista de contactos. Iremos directamente a las vip zones ya que la parte de los contactos ya la miramos en la práctica pasada.



Podemos añadir una nueva vipzone en el link de abajo



Vip zones

ID vip zone	Nombre	Numero de personas	Gasto	Disponibilidad			
1	Salesianos	5000	5000.0	1	Editar	Borrar	Clientes
2	Miami	3000	3000.0	1	Editar	Borrar	Clientes
3	Reservado sin editar	400	1000.0	0	Editar	Borrar	Clientes
4	Reservado editado	200	1000.0	1	Editar	Borrar	Clientes
6	Vipzone prueba	200	500.0	1	Editar	Borrar	Clientes

Podemos editarla y borrarla:

Editar Vip zone

ID vip zone: 6

Zona: Vipzone prueba

Numero de personas: 200

Gasto: 500.0

Disponibilidad: 1

Save

Vip zones

ID vip zone	Nombre	Numero de personas	Gasto	Disponibilidad			
1	Salesianos	5000	5000.0	1	Editar	Borrar	Clientes
2	Miami	3000	3000.0	1	Editar	Borrar	Clientes
3	Reservado sin editar	400	1000.0	0	Editar	Borrar	Clientes
4	Reservado editado	200	1000.0	1	Editar	Borrar	Clientes
6	Vipzone prueba Editada	300	1000.0	1	Editar	Borrar	Clientes

Para ver los clientes de una vipzone, damos en clientes:

Vip zones

ID vip zone	Nombre	Numero de personas	Gasto	Disponibilidad			
1	Salesianos	5000	5000.0	1	Editar	Borrar	Clientes
2	Miami	3000	3000.0	1	Editar	Borrar	Clientes
3	Reservado sin editar	400	1000.0	0	Editar	Borrar	Clientes
4	Reservado editado	200	1000.0	1	Editar	Borrar	Clientes

Añadir vip zone

VIP Clientes zone: Salesianos

ID Cliente	Dni	Nombre	Edad	Ciudad	Vip zone		
5	12345678C	Raul	14	Alda	1	Editar	Borrar
6	12345678D	Linda	15	kasa	1	Editar	Borrar
7	ajajjajaaj	Pa editar	45	EDITADO	1	Editar	Borrar

Añadir cliente vip Inicio

Igual que con las vipzones, se pueden añadir, editar y borrar clientes:

Crear nuevo Vip cliente

DNI:

Nombre:

Edad: 0

ciudad:

nº Vip Zone: 1

Save

VIP Clientes zone: Salesianos

ID Cliente	Dni	Nombre	Edad	Ciudad	Vip zone		
5	12345678C	Raul	14	Alda	1	Editar	Borrar
6	12345678D	Linda	15	kasa	1	Editar	Borrar
7	ajajjajaaj	Pa editar	45	EDITADO	1	Editar	Borrar
9	99999999A	Nuevo	20	cleinte	1	Editar	Borrar

Añadir cliente vip Inicio

Editar cliente vip

ID cliente vip: 9

DNI: 99999999A

Nombre: Nuevo

Edad: 20

ciudad: EDITADO

nº Vip Zone: 1

Save

VIP Clientes zone: Salesianos

ID Cliente	Dni	Nombre	Edad	Ciudad	Vip zone		
5	12345678C	Raul	14	Alda	1	Editar	Borrar
6	12345678D	Linda	15	kasa	1	Editar	Borrar
7	ajajjajaaj	Pa editar	45	EDITADO	1	Editar	Borrar
9	99999999A	Nuevo	20	EDITADO	1	Editar	Borrar

Añadir cliente vip Inicio

VIP Clientes zone: Salesianos

ID Cliente	Dni	Nombre	Edad	Ciudad	Vip zone		
5	12345678C	Raul	14	Alda	1	Editar	Borrar
6	12345678D	Linda	15	kasa	1	Editar	Borrar
9	99999999A	Nuevo	20	EDITADO	1	Editar	Borrar

Añadir cliente vip Inicio