

ESCUELA SUPERIOR FRANCISCANA ESPECIALIZADA / AGAPE



MINISTERIO
DE EDUCACIÓN,
CIENCIA Y
TECNOLOGÍA



CARRERA

Técnico en Ingeniería de Desarrollo de Software

MÓDULO

Creación de Aplicaciones en Java Enterprise Edition.

TÍTULO

Construcción de los Controladores del Proyecto.

ESTUDIANTE(S)

Daniel Alberto Hernández Masin
Metzelder Norberto Marroquín Ruiz
Armando José Jiménez Ochoa
Crisia Valeria Reyes Romero
Alejandro Ernesto Juárez Argumedo

DOCENTE

Luis Roberto García Zaña

FECHA DE PRESENTACIÓN

22/08/2025

CICLO

02

Capturas

1. Modelos

- Categoría

```
1 package org.esfe.modelos;
2
3 import jakarta.persistence.*;
4 import jakarta.validation.constraints.NotBlank;
5
6 @Entity 14 usages AlejandroJuarez002
7 @Table(name="categorias")
8 public class Categoria {
9     @Id 2 usages
10     @GeneratedValue(strategy = GenerationType.IDENTITY)
11     private Integer id;
12
13     @NotBlank(message="El nombre del Categoria es requerido") 2 usages
14     private String nombre;
15
16 > public String getNombre() { return nombre; }
17
18
19 > public void setNombre(String nombre) { this.nombre = nombre; }
20
21
22
23
24 > public Integer getId() { return id; }
25
26
27
28 > public void setId(Integer id) { this.id = id; }
29
30
31 }
```

- Departamento

```
Departamento.java x
1 package org.esfe.modelos;
2
3 import jakarta.persistence.*;
4 import jakarta.validation.constraints.NotBlank;
5
6 @Entity 14 usages AlejandroJuarez002
7 @Table(name="departamentos")
8 public class Departamento {
9     @Id 2 usages
10    @GeneratedValue(strategy = GenerationType.IDENTITY)
11    private Integer id;
12
13    @NotBlank(message="El nombre del Departamento es requerido") 2 usages
14    private String nombre;
15
16    > public Integer getId() { return id; }
17
18
19
20    > public void setId(Integer id) { this.id = id; }
21
22
23
24    > public String getNombre() { return nombre; }
25
26
27
28    > public void setNombre(String nombre) { this.nombre = nombre; }
29
30
31 }
```

- DestinoTuristico

```
9  @Entity 25 usages 1 DanielMasin
0  @Table(name = "destinoTuristicos")
1  public class DestinoTuristico {
2      @Id 2 usages
3      @GeneratedValue(strategy = GenerationType.IDENTITY)
4      private Integer id;
5
6      @NotBlank(message = "El nombre es requerido") 2 usages
7      private String nombre;
8
9      @NotBlank (message = "La descripcion es requerida") 2 usages
10     private String descripcion;
11
12     @NotBlank (message = "La ubicacion es requerida") 2 usages
13     private String ubicacion;
14
15     @NotBlank (message = "El horario es requerido") 2 usages
16     private String horario;
17
18     // Relación con Imagen (foránea en Imagen)
19     @OneToMany(mappedBy = "destinoTuristico", cascade = CascadeType.ALL, orphanRemoval = true) 2 usages
20     private List<Imagen> imagenes;
21
22     > public Integer getId() { return id; }
23
24     > public void setId(Integer id) { this.id = id; }
25
26     > public String getNombre() { return nombre; }
27
28     > public void setNombre(String nombre) { this.nombre = nombre; }
29
30     > public String getDescripcion() { return descripcion; }
31
32     > public void setDescripcion(String descripcion) { this.descripcion = descripcion; }
33
34     > public String getUbicacion() { return ubicacion; }
35
36     > public void setUbicacion(String ubicacion) { this.ubicacion = ubicacion; }
37
38     > public String getHorario() { return horario; }
39
40     > public void setHorario(String horario) { this.horario = horario; }
41
42     > public List<Imagen> getImagenes() { return imagenes; }
43
44     > public void setImagenes(List<Imagen> imagenes) { this.imagenes = imagenes; }
45 }
```

- Rol

```
7
8 @Entity 19 usages AlejandroJuarez002
9 @Table(name="roles")
10 public class Rol {
11     @Id 2 usages
12     @GeneratedValue(strategy = GenerationType.IDENTITY)
13     private Integer id;
14
15     @NotBlank(message = "El nombre del rol es requerido") 2 usages
16     private String nombre;
17
18     @OneToMany(mappedBy = "rol", fetch = FetchType.LAZY) 2 usages
19     private List<Usuario> usuarios; // Muchos usuarios pueden tener este rol
20
21     > public Integer getId() { return id; }
22
23     > public void setId(Integer id) { this.id = id; }
24
25     > public String getNombre() { return nombre; }
26
27     > public void setNombre(String nombre) { this.nombre = nombre; }
28
29     > public List<Usuario> getUsuarios() { return usuarios; }
30
31     > public void setUsuarios(List<Usuario> usuarios) { this.usuarios = usuarios; }
32 }
33
34
35
```

- Usuario

```

9  @Entity 31 usages AlejandroJuarez002
10 @Table(name="usuarios")
11 public class Usuario {
12     @Id 2 usages
13     @GeneratedValue(strategy = GenerationType.IDENTITY)
14     private Integer id;
15
16     @NotBlank(message = "Su nombre es requerido") 2 usages
17     private String nombre;
18
19     @NotBlank(message = "Su apellido es requerido") 2 usages
20     private String apellido;
21
22     @NotBlank(message = "El nombre de usuario es requerido") 2 usages
23     private String nombreUsuario;
24
25     @NotBlank(message = "La contraseña es requerida") 2 usages
26     private String clave;
27
28
29     private int status; 2 usages
30
31     @ManyToOne(fetch = FetchType.EAGER) 2 usages
32     @JoinColumn(name = "rol_id")
33     private Rol rol;

```

```

26 //private int status;
27
28 > public Integer getId() { return id; }
31
32 > public void setId(Integer id) { this.id = id; }
35
36 > public String getNombre() { return nombre; }
39
40 > public void setNombre(String nombre) { this.nombre = nombre; }
43
44 > public String getApellido() { return apellido; }
47
48 > public void setApellido(String apellido) { this.apellido = apellido; }
51
52 > public String getNombreUsuario() { return nombreUsuario; }
55
56 > public void setNombreUsuario(String nombreUsuario) { this.nombreUsuario = nombreUsuario; }
59
60 > public String getClave() { return clave; }
63
64 > public void setClave(String clave) { this.clave = clave; }
67 }

```

2. Repositorios

- CategoriaRepository

```
1 package org.esfe.repositorios;
2
3 import org.esfe.modelos.Categoria;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 public interface ICategoriaRepository extends JpaRepository<Categoria, Integer> { 2 usages AlejandroJuarez002
7 }
```

- DepartamentoRepository

```
1 package org.esfe.repositorios;
2
3 > import ...
4
5
6 public interface IDepartamentoRepository extends JpaRepository<Departamento, Integer> { 2 usages AlejandroJuarez002
7 }
```

- DestinoTuristicoRepository

```
7 public interface IDestinoTuristicoRepository extends JpaRepository<DestinoTuristico, Integer> { 4 usages DanielMasin
8
9     // Buscar por coincidencia parcial en el nombre (ignora mayúsculas/minúsculas)
10    List<DestinoTuristico> findByNombreContainingIgnoreCase(String nombre); 2 usages DanielMasin
11
12    // Opcional: si también quieres buscar en descripción o ubicación
13    List<DestinoTuristico> findByNombreContainingIgnoreCaseOrDescripcionContainingIgnoreCaseOrUbicacionContainingIgnoreCase(
14        String nombre, String descripcion, String ubicacion
15    );
16 }
17
```

- RolRepository

```
I RolRepository.java x
1 package org.esfe.repositorios;
2
3 import org.esfe.modelos.Rol;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 public interface IRolRepository extends JpaRepository<Rol, Integer> { 2 usages AlejandroJuarez002
7 }
```

- UsuarioRepository

```
I UsuarioRepository.java x
1 package org.esfe.repositorios;
2
3 > import ...
7
8 public interface IUsuarioRepository extends JpaRepository<Usuario, Integer> { 2 usages AlejandroJuarez002
9     Page<Usuario> findByStatus(int status, Pageable pageable); 1 usage AlejandroJuarez002
10 }
```


3. Servicios

➤ Implementaciones

- CategoriaService

```

1 package org.esfe.servicios.implementaciones;
2
3 import org.esfe.modelos.Categoria;
4 import org.esfe.repositorios.ICategoriaRepository;
5 import org.esfe.servicios.interfaces.ICategoriaService;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.data.domain.Page;
8 import org.springframework.data.domain.Pageable;
9 import org.springframework.stereotype.Service;
10
11 import java.util.List;
12 import java.util.Optional;
13
14 /**
15  * Se implementan los metodos de ICategoriaService
16  */
17 @Service no usages AlejandroJuarez002
18 public class CategoriaService implements ICategoriaService {
19     @Autowired 5 usages
20     private ICategoriaRepository categoriaRepository;
21
22     /**
23      * Devuelve registros de manera paginada
24      * @param pageable
25      * @return
26      */
27     @Override no usages AlejandroJuarez002
28     public Page<Categoria> buscarTodosPaginados(Pageable pageable) { return categoriaRepository.findAll(pageable); }
29
30     /**
31      * Devuelve una lista
32      * @return
33      */
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

```

```

35     */
36     @Override no usages AlejandroJuarez002
37     public List<Categoria> obtenerTodos() { return categoriaRepository.findAll(); }
38
39
40
41     @Override no usages AlejandroJuarez002
42     public Optional<Categoria> buscarPorId(Integer id) { return categoriaRepository.findById(id); }
43
44
45
46     /**
47      * Devuelve metodo save
48      * @param categoria
49      * @return
50      */
51     @Override no usages AlejandroJuarez002
52     public Categoria crearOEditar(Categoria categoria) { return categoriaRepository.save(categoria); }
53
54
55
56     @Override no usages AlejandroJuarez002
57     public void eliminarPorId(Integer id) { categoriaRepository.deleteById(id); }
58
59
60 }

```

- DepartamentoService

```
DepartamentoService.java x
1 package org.esfe.servicios.implementaciones;
2
3 import org.esfe.modelos.Departamento;
4 import org.esfe.repositorios.IDepartamentoRepository;
5 import org.esfe.servicios.interfaces.IDepartamentoService;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.data.domain.Page;
8 import org.springframework.data.domain.Pageable;
9 import org.springframework.stereotype.Service;
10
11 import java.util.List;
12 import java.util.Optional;
13
14 /**
15  * Se implementan los metodos de IDepartamentoService
16  */
17 @Service no usages AlejandroJuarez002
18 public class DepartamentoService implements IDepartamentoService {
19     @Autowired 5 usages
20     private IDepartamentoRepository departamentoRepository;
21
22     /**
23      * Devuelve registros de manera paginada
24      * @param pageable
25      * @return
26      */
27     @Override no usages AlejandroJuarez002
28     public Page<Departamento> buscarTodosPaginados(Pageable pageable) {
29         return departamentoRepository.findAll(pageable);
30     }
```

```
31
32     /**
33      * Devuelve una lista
34      * @return
35      */
36     @Override no usages AlejandroJuarez002
37     public List<Departamento> obtenerTodos() { return departamentoRepository.findAll(); }
38
39
40
41     @Override no usages AlejandroJuarez002
42     public Optional<Departamento> buscarPorId(Integer id) { return departamentoRepository.findById(id); }
43
44
45     /**
46      * Devuelve metodo save
47      * @param departamento
48      * @return
49      */
50
51     @Override no usages AlejandroJuarez002
52     public Departamento crearOEditar(Departamento departamento) { return departamentoRepository.save(departamento); }
53
54
55
56     @Override no usages AlejandroJuarez002
57     public void eliminarPorId(Integer id) { departamentoRepository.deleteById(id); }
58 }
59
```

- DestinoTuristicoService

```
1 package org.esfe.servicios.implementaciones;
2
3 > import ...
13
14 @Service no usages DanielMasin
15 public class DestinoTuristicoService implements IDestinoTuristicoService {
16     @Autowired 4 usages
17     private IDestinoTuristicoRepository destinoTuristicoRepository;
18
19     @Override no usages DanielMasin
20     public Page<DestinoTuristico> buscarTodosPaginados(Pageable pageable) {
21         return destinoTuristicoRepository.findAll(pageable);
22     }
23
24     @Override no usages DanielMasin
25     public List<DestinoTuristico> ObtenerTodo() { return destinoTuristicoRepository.findAll (); }
28
29     @Override no usages DanielMasin
30     public DestinoTuristico createOEdit(DestinoTuristico destinoTuristico) {
31         return destinoTuristicoRepository.save (destinoTuristico);
32     }
33
34     @Override DanielMasin
35     public void eliminarPorId(Integer id) { destinoTuristicoRepository.deleteById (id); }
38 }
```

- RolService

```
RolService.java x
1 package org.esfe.servicios.implementaciones;
2
3 > import ...
12
13 @Service no usages AlejandroJuarez002
14 public class RolService implements IRolService {
15     @Autowired 4 usages
16     private IRolRepository rolRepository;
17
18     @Override no usages AlejandroJuarez002
19 > public Page<Rol> buscarTodosPaginados(Pageable pageable) { return rolRepository.findAll(pageable); }
22
23     @Override no usages AlejandroJuarez002
24 > public List<Rol> obtenerTodos() { return rolRepository.findAll(); }
27
28     @Override no usages AlejandroJuarez002
29 > public Rol crearOEditor(Rol rol) { return rolRepository.save(rol); }
32
33     @Override AlejandroJuarez002
34 > public void eliminarPorId(Integer id) { rolRepository.deleteById(id); }
37 }
```

- UsuarioService

```
UsuarioService.java x
1 package org.esfe.servicios.implementaciones;
2
3 > import ...
13
14 @Service no usages AlejandroJuarez002
15 public class UsuarioService implements IUsuarioService {
16     @Autowired 5 usages
17     private IUsuarioRepository usuarioRepository;
18
19     @Override no usages AlejandroJuarez002
20 > public Page<Usuario> buscarTodosPaginados(Pageable pageable) { return usuarioRepository.findAll(pageable); }
23
24     @Override no usages AlejandroJuarez002
25 > public List<Usuario> obtenerTodos() { return usuarioRepository.findAll(); }
28
29     @Override no usages AlejandroJuarez002
30 > public Optional<Usuario> buscarPorId(Integer id) { return usuarioRepository.findById(id); }
33
34     @Override no usages AlejandroJuarez002
35 > public Usuario crearOEditor(Usuario usuario) { return usuarioRepository.save(usuario); }
38
39     @Override AlejandroJuarez002
40 > public void eliminarPorId(Integer id) { usuarioRepository.deleteById(id); }
43 }
```

➤ Interfaces

- ICategoriaService

```
1 package org.esfe.servicios.interfaces;
2
3 import org.esfe.modelos.Categoria;
4 import org.springframework.data.domain.Page;
5 import org.springframework.data.domain.Pageable;
6
7 import java.util.List;
8 import java.util.Optional;
9
10 public interface ICategoriaService { 2 usages 1 implementation AlejandroJuarez002
11     Page<Categoria> buscarTodosPaginados(Pageable pageable); no usages 1 implementation AlejandroJuarez002
12
13     List<Categoria> obtenerTodos(); no usages 1 implementation AlejandroJuarez002
14
15     Optional<Categoria> buscarPorId(Integer id); no usages 1 implementation AlejandroJuarez002
16
17     Categoria crearOEditar(Categoria categoria); no usages 1 implementation AlejandroJuarez002
18
19     void eliminarPorId(Integer id); no usages 1 implementation AlejandroJuarez002
20 }
```

- IDepartamentoService

```
IDepartamentoService.java x
1 package org.esfe.servicios.interfaces;
2
3 import org.esfe.modelos.Departamento;
4 import org.springframework.data.domain.Page;
5 import org.springframework.data.domain.Pageable;
6
7 import java.util.List;
8 import java.util.Optional;
9
10 public interface IDepartamentoService { 2 usages 1 implementation AlejandroJuarez002
11     Page<Departamento> buscarTodosPaginados(Pageable pageable); no usages 1 implementation AlejandroJuarez002
12
13     List<Departamento> obtenerTodos(); no usages 1 implementation AlejandroJuarez002
14
15     Optional<Departamento> buscarPorId(Integer id); no usages 1 implementation AlejandroJuarez002
16
17     Departamento crearOEditar(Departamento departamento); no usages 1 implementation AlejandroJuarez002
18
19     void eliminarPorId(Integer id); no usages 1 implementation AlejandroJuarez002
20 }
```

- IDestinoTuristicoService

```
IDestinoTuristicoService.java x
1 package org.esfe.servicios.interfaces;
2
3
4 import org.esfe.modelos.DestinoTuristico;
5 import org.springframework.data.domain.Page;
6 import org.springframework.data.domain.Pageable;
7
8 import java.util.List;
9
10 public interface IDestinoTuristicoService { 2 usages 1 implementation DanielMasin
11
12     Page<DestinoTuristico> buscarTodosPaginados(Pageable pageable); no usages 1 implementation DanielMasin
13
14     List<DestinoTuristico> obtenerTodo(); no usages 1 implementation DanielMasin
15
16     DestinoTuristico createOEdit (DestinoTuristico destinoTuristico); no usages 1 implementation DanielMasin
17
18     void eliminarPorId(Integer destinoTuristico); 1 implementation DanielMasin
19 }
```

- IRolService

```
IRolService.java x
1 package org.esfe.servicios.interfaces;
2
3 import org.esfe.modelos.Rol;
4 import org.springframework.data.domain.Page;
5 import org.springframework.data.domain.Pageable;
6
7 import java.util.List;
8
9 public interface IRolService { 2 usages 1 implementation AlejandroJuarez002
10     Page<Rol> buscarTodosPaginados(Pageable pageable); no usages 1 implementation AlejandroJuarez002
11
12     List<Rol> obtenerTodos(); no usages 1 implementation AlejandroJuarez002
13
14     Rol crearOEditar(Rol rol); no usages 1 implementation AlejandroJuarez002
15
16     void eliminarPorId(Integer id); 1 implementation AlejandroJuarez002
17 }
```

- IUusuarioService

```
IUsuarioService.java x
1 package org.esfe.servicios.interfaces;
2
3 > import ...
4
5
6
7
8
9
10 public interface IUsuarioService { 4 usages 1 implementation AlejandroJuarez002
11     Page<Usuario> buscarTodosPaginados(Pageable pageable); 1 implementation AlejandroJuarez002
12
13     List<Usuario> obtenerTodos(); no usages 1 implementation AlejandroJuarez002
14
15     Optional<Usuario> buscarPorId(Integer id); 1 implementation AlejandroJuarez002
16
17     Usuario crearOEditar(Usuario usuario); 1 implementation AlejandroJuarez002
18
19     void eliminarPorId(Integer id); 1 implementation AlejandroJuarez002
20
21     Page<Usuario> buscarPorStatus(int status, Pageable pageable); 2 usages 1 implementation AlejandroJuarez002
22 }
```


➤ Controladores

- CategoriaController

```
CategoriaController.java x
1 package org.esfe.controladores;
2
3 import org.esfe.modelos.Categoria;
4 import org.esfe.servicios.interfaces.ICategoriaService;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.data.domain.Page;
7 import org.springframework.data.domain.PageRequest;
8 import org.springframework.data.domain.Pageable;
9 import org.springframework.stereotype.Controller;
10 import org.springframework.ui.Model;
11 import org.springframework.validation.BindingResult;
12 import org.springframework.web.bind.annotation.*;
13 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
14
15 import java.util.List;
16 import java.util.Optional;
17 import java.util.stream.Collectors;
18 import java.util.stream.IntStream;
19
20 @Controller no usages  ⚠ r24004
21 @RequestMapping("/categorias")
22 public class CategoriaController {
23
24     @Autowired 6 usages
25     private ICategoriaService categoriaService;
26
27     @GetMapping no usages  ⚠ r24004
28     public String index(Model model, @RequestParam("page") Optional<Integer> page, @RequestParam("size") Optional<Integer> size) {
29         int currentPage = page.orElse(0) - 1; // por defecto página 0
30         int pageSize = size.orElse(5); // por defecto tamaño de página = 5
31         Pageable pageable = PageRequest.of(currentPage, pageSize);
32
33         Page<Categoria> categorias = categoriaService.buscarTodosPaginados(pageable);
34         model.addAttribute("categorias", categorias);
35
36         int totalPages = categorias.getTotalPages();
37     }
38 }
```

```
CategoriaController.java x
22 public class CategoriaController {
23     public String index(Model model, @RequestParam("page") Optional<Integer> page, @RequestParam("size") Optional<Integer> size) {
24
25         int totalPages = categorias.getTotalPages();
26         if (totalPages > 0) {
27             List<Integer> pageNumbers = IntStream.rangeClosed(1, totalPages).boxed().collect(Collectors.toList());
28             model.addAttribute("pageNumbers", pageNumbers);
29         }
30
31         return "categoria/index";
32     }
33
34     @GetMapping("/create") no usages  ⚠ r24004
35     public String create(Categoria categoria) { return "categoria/create"; }
36
37     @PostMapping("/save") no usages  ⚠ r24004
38     public String save(Categoria categoria, BindingResult result, Model model, RedirectAttributes attributes) {
39         if (result.hasErrors()) {
40             model.addAttribute(categoria);
41             attributes.addFlashAttribute("error", "No se pudo guardar debido a un error");
42             return "categoria/create";
43         }
44
45         categoriaService.crearOEditar(categoria);
46         attributes.addFlashAttribute("msg", "Categoria creada correctamente");
47         return "redirect:/categorias";
48     }
49
50     @GetMapping("/details/{id}") no usages  ⚠ r24004
51     public String details(@PathVariable("id") Integer id, Model model) {
52         Categoria categoria = categoriaService.buscarPorId(id).orElse(null);
53         model.addAttribute("categoria", categoria);
54         return "categoria/details";
55     }
56 }
```



```

1  public class CategoriaController {
2
3      @GetMapping("/details/{id}") no usages &rr24004
4      @
5      public String details(@PathVariable("id") Integer id, Model model) {
6          Categoria categoria = categoriaService.buscarPorId(id).orElse( other: null);
7          model.addAttribute( attributeName: "categoria", categoria);
8          return "categoria/details";
9      }
10
11      @GetMapping("/edit/{id}") no usages &rr24004
12      @
13      public String edit(@PathVariable("id") Integer id, Model model) {
14          Categoria categoria = categoriaService.buscarPorId(id).orElse( other: null);
15          model.addAttribute( attributeName: "categoria", categoria);
16          return "categoria/edit";
17      }
18
19      @GetMapping("/remove/{id}") no usages &rr24004
20      @
21      public String remove(@PathVariable("id") Integer id, Model model) {
22          Categoria categoria = categoriaService.buscarPorId(id).orElse( other: null);
23          model.addAttribute( attributeName: "categoria", categoria);
24          return "categoria/delete";
25      }
26
27      @PostMapping("/delete") no usages &rr24004
28      @
29      public String delete(Categoria categoria, RedirectAttributes attributes) {
30          categoriaService.eliminarPorId(categoria.getId());
31          attributes.addFlashAttribute( attributeName: "msg", attributeValue: "Categoria eliminada correctamente");
32          return "redirect:/categorias";
33      }
34  }

```

- DepartamentoController

```

1  package org.esfe.controladores;
2
3  import org.esfe.modelos.Departamento;
4  import org.esfe.servicios.interfaces.IDepartamentoService;
5  import org.springframework.beans.factory.annotation.Autowired;
6  import org.springframework.data.domain.Page;
7  import org.springframework.data.domain.PageRequest;
8  import org.springframework.data.domain.Pageable;
9  import org.springframework.stereotype.Controller;
10 import org.springframework.ui.Model;
11 import org.springframework.validation.BindingResult;
12 import org.springframework.web.bind.annotation.*;
13 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
14
15
16 import java.util.List;
17 import java.util.Optional;
18 import java.util.stream.Collectors;
19 import java.util.stream.IntStream;
20
21 @Controller no usages &rr24004
22 @RequestMapping("/departamentos")
23 public class DepartamentoController {
24     @Autowired 6 usages
25     private IDepartamentoService departamentoService;
26
27     @GetMapping no usages &rr24004
28     @
29     public String index(Model model, @RequestParam("page") Optional<Integer> page, @RequestParam("size") Optional<Integer> si
30         int currentPage = page.orElse( other: 1) - 1; //si no esta seteado se asigna 0
31         int pageSize = size.orElse( other: 5); //tamano de la pagina se asigna 5
32         Pageable pageable = PageRequest.of(currentPage, pageSize);
33     }
34 }

```

```

CategoriaController.java  DepartamentoController.java x
23 public class DepartamentoController {
28     public String index(Model model, @RequestParam("page") Optional<Integer> page, @RequestParam("size") Optional<Integer> si
32
33         Page<Departamento> departamentos = departamentoService.buscarTodosPaginados(pageable);
34         model.addAttribute(attributeName: "departamentos", departamentos);
35
36         int totalPages = departamentos.getTotalPages();
37         if (totalPages > 0) {
38             List<Integer> pageNumbers = IntStream.rangeClosed(1, totalPages) .boxed() .collect(Collectors.toList());
39             model.addAttribute(attributeName: "pageNumbers", pageNumbers);
40         }
41
42         return "departamento/index";
43     }
44
45     @GetMapping("/create") no usages 2 rr24004
46     public String create(Departamento departamento) { return "departamento/create"; }
47
48     @PostMapping("/save") no usages 2 rr24004
49     public String save(Departamento departamento, BindingResult result, Model model, RedirectAttributes attributes) {
50         if (result.hasErrors()) {
51             model.addAttribute(departamento);
52             attributes.addFlashAttribute(attributeName: "error", attributeValue: "No se pudo guardar debido a un error");
53             return "departamento/create";
54         }
55
56         departamentoService.crearOEditar(departamento);
57         attributes.addFlashAttribute(attributeName: "msg", attributeValue: "Grupo creado correctamente");
58         return "redirect:/departamentos";
59     }
60
61 }
62
63

```

```

CategoriaController.java  DepartamentoController.java x
23 public class DepartamentoController {
63
64
65     @GetMapping("/details/{id}") no usages 2 rr24004
66     public String details(@PathVariable("id") Integer id, Model model) {
67         Departamento departamento = departamentoService.buscarPorId(id).get();
68         model.addAttribute(attributeName: "departamento", departamento);
69         return "departamento/details";
70     }
71
72     @GetMapping("/edit/{id}") no usages 2 rr24004
73     public String edit(@PathVariable("id") Integer id, Model model) {
74         Departamento departamento = departamentoService.buscarPorId(id).get();
75         model.addAttribute(attributeName: "departamento", departamento);
76         return "departamento/edit";
77     }
78
79
80     @GetMapping("/remove/{id}") no usages 2 rr24004
81     public String remove(@PathVariable("id") Integer id, Model model) {
82         Departamento departamento = departamentoService.buscarPorId(id).get();
83         model.addAttribute(attributeName: "departamento", departamento);
84         return "departamento/delete";
85     }
86
87     @PostMapping("/delete") no usages 2 rr24004
88     public String delete(Departamento departamento, RedirectAttributes attributes) {
89         departamentoService.eliminarPorId(departamento.getId());
90         attributes.addFlashAttribute(attributeName: "msg", attributeValue: "Departamento eliminado correctamente");
91         return "redirect:/departamentos";
92     }
93 }
94

```

- DestinoTuristicoController

```

1 package org.esfe.controladores;
2
3
4 import org.esfe.modelos.DestinoTuristico;
5 import org.esfe.modelos.Imagen;
6 import org.esfe.modelos.ImagenDTO;
7 import org.esfe.repositorios.IDestinoTuristicoRepository;
8 import org.esfe.servicios.interfaces.IDestinoTuristicoService;
9 import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.boot.Banner;
11 import org.springframework.data.domain.Page;
12 import org.springframework.data.domain.PageImpl;
13 import org.springframework.data.domain.PageRequest;
14 import org.springframework.stereotype.Controller;
15 import org.springframework.ui.Model;
16 import org.springframework.validation.BindingResult;
17 import org.springframework.web.bind.annotation.*;
18 import org.springframework.data.domain.Pageable;
19 import org.springframework.web.multipart.MultipartFile;
20 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
21
22 import java.io.IOException;
23 import java.util.*;
24 import java.util.stream.Collectors;
25 import java.util.stream.IntStream;
26
27 @Controller no usages & DanielMasin
28 @RequestMapping("/destinoTuristicos")
29 public class DestinoTuristicoController {
30
31     @Autowired 6 usages
32     private IDestinoTuristicoService destinoTuristicoService;
33
34     @Autowired 2 usages
35     private IDestinoTuristicoRepository destinoTuristicoRepository;
36

```

```

29 public class DestinoTuristicoController {
30
31     private IDestinoTuristicoRepository destinoTuristicoRepository;
32
33     @GetMapping no usages & DanielMasin
34     @RequestMapping("/index")
35     public String index(Model model,
36                         @RequestParam("page") Optional<Integer> page,
37                         @RequestParam("size") Optional<Integer> size,
38                         @RequestParam(value = "q", required = false) String q) {
39
40         int currentPage = page.orElse(1) - 1;
41         int pageSize = size.orElse(5);
42         Pageable pageable = PageRequest.of(currentPage, pageSize);
43
44         Page<DestinoTuristico> destinos;
45
46         if (q != null && !q.isEmpty()) {
47             // Si hay búsqueda, no paginar (opcional: se puede implementar paginación también)
48             List<DestinoTuristico> resultados = destinoTuristicoRepository.findByNombreContainingIgnoreCase(q);
49             destinos = new PageImpl<>(resultados, pageable, resultados.size());
50         } else {
51             destinos = destinoTuristicoService.buscarTodosPaginados(pageable);
52         }
53
54         // Convertir imágenes a Base64
55         Page<Map<String, Object>> destinosDTO = destinos.map(destinoTuristico -> {
56             List<ImagenDTO> imagenesDTO = destinoTuristico.getImagenes().stream().map(img -> {
57                 new ImagenDTO(
58                     img.getId(),
59                     img.getBytesImage() != null ?
60                         Base64.getEncoder().encodeToString(img.getBytesImage()) : null
61                 );
62             }).collect(Collectors.toList());
63
64             Map<String, Object> dto = new HashMap<>();
65             dto.put("destino", destinoTuristico);
66             dto.put("imagenes", imagenesDTO);
67
68             return dto;
69         });
70

```

```
DestinoTuristicoController.java x
29 public class DestinoTuristicoController {
35     private IDestinoTuristicoRepository destinoTuristicoRepository;
36
37     @GetMapping no usages & DanielMasin
38     @ public String index(Model model,
39         @RequestParam("page") Optional<Integer> page,
40         @RequestParam("size") Optional<Integer> size,
41         @RequestParam(value = "q", required = false) String q) {
42
43         int currentPage = page.orElse(1) - 1;
44         int pageSize = size.orElse(5);
45         Pageable pageable = PageRequest.of(currentPage, pageSize);
46
47         Page<DestinoTuristico> destinos;
48
49         if (q != null && !q.isEmpty()) {
50             // Si hay búsqueda, no paginar (opcional: se puede implementar paginación también)
51             List<DestinoTuristico> resultados = destinoTuristicoRepository.findByNombreContainingIgnoreCase(q);
52             destinos = new PageImpl<>(resultados, pageable, resultados.size());
53         } else {
54             destinos = destinoTuristicoService.buscarTodosPaginados(pageable);
55         }
56
57         // Convertir imágenes a Base64
58         Page<Map<String, Object>> destinosDTO = destinos.map( DestinoTuristico dest -> {
59             List<ImagenDTO> imagenesDTO = dest.getImagenes().stream().stream<Imagen>
60                 .map( imagen img -> new ImagenDTO(
61                     img.getId(),
62                     img.getBytesArrayImage() != null ?
63                         Base64.getEncoder().encodeToString(img.getBytesArrayImage()) : null)) Stream<ImagenDTO>
64                 .collect(Collectors.toList());
65
66             Map<String, Object> dto = new HashMap<>();
67             dto.put("destino", dest);
68             dto.put("imagenes", imagenesDTO);
69             return dto;
70         });
71     }
72 }
```

```
DestinoTuristicoController.java x
29 public class DestinoTuristicoController {
90     public String save(@ModelAttribute DestinoTuristico destinoTuristico,
101
102     try {
103         List<Imagen> listaImagenes = new ArrayList<>();
104         for (MultipartFile file : imagenFiles) {
105             if (!file.isEmpty()) {
106                 Imagen img = new Imagen();
107                 img.setBytesArrayImage(file.getBytes());
108                 img.setDestinoTuristico(destinoTuristico);
109                 listaImagenes.add(img);
110             }
111         }
112         destinoTuristico.setImagenes(listaImagenes);
113
114     } catch (IOException e) {
115         attributes.addFlashAttribute("error", "Error al procesar las imágenes.");
116         return "destinoTuristico/create";
117     }
118
119     destinoTuristicoService.createOrUpdate(destinoTuristico);
120     attributes.addFlashAttribute("msg", "Destino turistico creado correctamente.");
121     return "redirect:/destinoTuristicos";
122 }
123
124 @GetMapping("/details/{id}") no usages & DanielMasin
125 @ public String details(@PathVariable("id") Integer id, Model model){
126     DestinoTuristico destinoTuristico = destinoTuristicoService.buscarPorId(id).get();
127     model.addAttribute("destinoTuristico", destinoTuristico);
128     return "destinoTuristico/details";
129 }
130
131 @GetMapping("/edit/{id}") no usages & DanielMasin
132 @ public String edit(@PathVariable("id") Integer id, Model model){
133     DestinoTuristico destinoTuristico = destinoTuristicoService.buscarPorId(id).get();
134     model.addAttribute("destinoTuristico", destinoTuristico);
135     return "destinoTuristico/edit";
136 }
137 }
```

```

DestinoTuristicoController.java
29 public class DestinoTuristicoController {
130
131     @GetMapping("/edit/{Id}") no usages & DanielMasin
132     @ public String edit(@PathVariable("Id") Integer Id, Model model){
133         DestinoTuristico destinoTuristico = destinoTuristicoService.buscarPorId (Id).get();
134         model.addAttribute ( attributeName: "destinoTuristico", destinoTuristico);
135         return "destinoTuristico/edit";
136     }
137
138     @GetMapping("/remove/{Id}") no usages & DanielMasin
139     @ public String remove(@PathVariable("Id") Integer Id, Model model){
140         DestinoTuristico destino = destinoTuristicoService.buscarPorId(Id).get();
141         model.addAttribute( attributeName: "destino", destino);
142         return "destinoTuristico/delete";
143     }
144
145     @PostMapping("/delete") no usages & DanielMasin
146     @ public String delete(DestinoTuristico destino, RedirectAttributes attributes){
147         destinoTuristicoService.eliminarPorId(destino.getId());
148         attributes.addFlashAttribute( attributeName: "msg", attributeValue: "Destino eliminado correctamente");
149         return "redirect:/destinoTuristicos";
150     }
151
152     @GetMapping("/search") no usages & DanielMasin
153     @ public String search(@RequestParam("q") String query, Model model) {
154         List<DestinoTuristico> resultados = destinoTuristicoRepository.findByNombreContainingIgnoreCase(query);
155         model.addAttribute( attributeName: "destinos", resultados);
156         model.addAttribute( attributeName: "q", query);
157         return "destinoTuristicos/index"; // usa la misma vista que el listado normal
158     }
159
160 }
161

```

- HomeController

```

HomeController.java
1 package org.esfe.controladores;
2
3 import org.esfe.modelos.DestinoTuristico;
4 import org.esfe.modelos.ImagenDTO;
5 import org.esfe.servicios.interfaces.IDestinoTuristicoService;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Controller;
8 import org.springframework.ui.Model;
9 import org.springframework.web.bind.annotation.GetMapping;
10
11 import java.util.Base64;
12 import java.util.HashMap;
13 import java.util.List;
14 import java.util.Map;
15 import java.util.stream.Collectors;
16
17 @Controller no usages & DanielMasin +1
18 public class HomeController {
19
20     @Autowired 1 usage
21     private IDestinoTuristicoService destinoTuristicoService;
22
23     @GetMapping("/{", "/home"}) no usages & DanielMasin +1
24     @ public String index(Model model) {
25         List<DestinoTuristico> destinos = destinoTuristicoService.ObtenerTodo();
26         List<Map<String, Object>> destinosDTO = destinos.stream().map( DestinoTuristico dest -> {
27             List<ImagenDTO> imagenesDTO = dest.getImagenes().stream() Stream<Imagen>
28                 .map( Imagen img -> new ImagenDTO(
29                     img.getId(),
30                     img.getBytesArrayImage() != null ?
31                         Base64.getEncoder().encodeToString(img.getBytesArrayImage()) : null) Stream<ImagenDTO>
32                 .collect(Collectors.toList());
33             Map<String, Object> dto = new HashMap<>();
34             dto.put("destino", dest);
35             dto.put("imagenes", imagenesDTO);
36             return dto;
37         }).collect(Collectors.toList());
38         model.addAttribute( attributeName: "destinoTuristicos", destinosDTO);
39         return "home/index";
40     }
41
42

```

- UsuarioController

```
1 package org.esfe.controladores;
2
3 import org.esfe.modelos.Rol;
4 import org.esfe.modelos.Usuario;
5 import org.esfe.servicios.interfaces.IRolService;
6 import org.esfe.servicios.interfaces.IUsuarioService;
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.data.domain.Page;
9 import org.springframework.data.domain.PageRequest;
10 import org.springframework.data.domain.Pageable;
11 import org.springframework.security.core.Authentication;
12 import org.springframework.security.core.context.SecurityContextHolder;
13 import org.springframework.security.crypto.password.PasswordEncoder;
14 import org.springframework.stereotype.Controller;
15 import org.springframework.ui.Model;
16 import org.springframework.validation.BindingResult;
17 import org.springframework.web.bind.annotation.*;
18 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
19
20 import java.util.List;
21 import java.util.Optional;
22 import java.util.stream.Collectors;
23 import java.util.stream.IntStream;
24
25 @Controller no usages AlejandroJuarez002
26 @RequestMapping("/usuarios")
27 public class UsuarioController {
28     @Autowired 13 usages
29     private IUsuarioService usuarioService;
30
31     @Autowired 1 usage
32     private IRolService rolService;
33
34     @Autowired 1 usage
35     private PasswordEncoder passwordEncoder;
36 }
```

```
27 public class UsuarioController {
36
37     @GetMapping no usages AlejandroJuarez002
38     private String index(Model model,
39         @RequestParam("page") Optional<Integer> page,
40         @RequestParam("size") Optional<Integer> size) {
41         int currentPage = page.orElse(1) - 1; //Si no esta seteado se asigna 0
42         int pageSize = size.orElse(5); //Tamaño de la pagina, se asigna 5
43         Pageable pageable = PageRequest.of(currentPage, pageSize);
44
45         Page<Usuario> usuarios = usuarioService.buscarPorStatus(status: 1, pageable); //Solo usuarios activos
46         model.addAttribute(attributeName: "usuarios", usuarios);
47
48         int totalPages = usuarios.getTotalPages();
49         if (totalPages > 0) {
50             List<Integer> pageNumbers = IntStream.rangeClosed(1, totalPages).boxed().collect(Collectors.toList());
51             model.addAttribute(attributeName: "pageNumbers", pageNumbers);
52         }
53         return "usuario/index";
54     }
55
56     /**
57      * Accion CREATE
58      */
59     @GetMapping("/create") no usages AlejandroJuarez002
60     public String create(Usuario usuario) { return "usuario/create"; }
61
62     @PostMapping("/save") no usages AlejandroJuarez002
63     public String save(Usuario usuario, BindingResult result, Model model, RedirectAttributes attributes) {
64
65         if (result.hasErrors()) {
66             model.addAttribute(usuario);
67             attributes.addFlashAttribute(attributeName: "error", attributeValue: "No se pudo guardar debido a un error.");
68             return "usuario/create";
69         }
70     }
71 }
```

```

@ UsuarioController.java x
27 public class UsuarioController {
67     public String save(Usuario usuario, BindingResult result, Model model, RedirectAttributes attributes) {
71         attributes.addFlashAttribute( attributeName: "error", attributeValue: "No se pudo guardar debido a un error.");
72         return "usuario/create";
73     }
74
75     /**
76      * Validar que el rol exista si se asignó uno
77      */
78     if (usuario.getRol() == null) {
79         Rol rol = rolService.buscarPorId(1)
80             .orElseThrow(() -> new IllegalArgumentException("Rol no encontrado"));
81         usuario.setRol(rol);
82     } else {
83         attributes.addFlashAttribute( attributeName: "error", attributeValue: "Debe seleccionar un rol válido.");
84         return "usuario/create";
85     }
86
87     String password = passwordEncoder.encode(usuario.getClave());
88
89     usuario.setStatus(1);
90     usuario.setClave(password);
91     usuarioService.crearOEditar(usuario);
92     attributes.addFlashAttribute( attributeName: "msg", attributeValue: "Usuario creado correctamente.");
93     return "redirect:/usuarios";
94 }
95
96 /**
97  * Accion DETAILS
98  */
99 @GetMapping("/details/{id}") no usages AlejandroJuarez002
100 @ public String details(@PathVariable("id") Integer id, Model model) {
101     Usuario usuario = usuarioService.buscarPorId(id).get();
102     model.addAttribute( attributeName: "usuario", usuario);
103     return "usuario/details";
104 }

```

```

@ UsuarioController.java x
27 public class UsuarioController {
100     public String details(@PathVariable("id") Integer id, Model model) {
104     }
105
106     /**
107      * Accion EDIT
108      */
109     @GetMapping("/edit/{id}") no usages AlejandroJuarez002
110 @ public String edit(@PathVariable("id") Integer id, Model model) {
111     Usuario usuario = usuarioService.buscarPorId(id).get();
112     model.addAttribute( attributeName: "usuario", usuario);
113     return "usuario/edit";
114 }
115
116 /**
117  * Accion DESACTIVATE
118  */
119 @GetMapping("/desactivate/{id}") no usages AlejandroJuarez002
120 @ public String showDesactivateView(@PathVariable("id") Integer id, Model model) {
121     Usuario usuario = usuarioService.buscarPorId(id)
122         .orElseThrow(() -> new IllegalArgumentException("Usuario no encontrado"));
123
124     // Usuario logeado
125     Authentication auth = SecurityContextHolder.getContext().getAuthentication();
126     String username = auth.getName();
127
128     Usuario usuarioLogeado = usuarioService.buscarPorNombreUsuario(username)
129         .orElseThrow(() -> new IllegalArgumentException("Usuario logeado no encontrado"));
130
131     // Se agrega bandera si es el mismo
132     boolean esMismoUsuario = usuarioLogeado.getId().equals(usuario.getId());
133
134     model.addAttribute( attributeName: "usuario", usuario);
135     model.addAttribute( attributeName: "esMismoUsuario", esMismoUsuario);
136
137     return "usuario/desactivate";

```

```

27 public class UsuarioController {
120 public String showDesactivateView(@PathVariable("id") Integer id, Model model) {
138 }
139
140 @PostMapping("/desactivate/{id}") no usages & AlejandroJuarez002
141 public String desactivate(@PathVariable("id") Integer id, RedirectAttributes attributes) {
142 // Obtiene el usuario autenticado desde Spring Security
143 Authentication auth = SecurityContextHolder.getContext().getAuthentication();
144 String username = auth.getName();
145
146 // Busca en base de datos el usuario logeado
147 Usuario usuarioLogeado = usuarioService.buscarPorNombreUsuario(username)
148 | .orElseThrow(() -> new IllegalArgumentException("Usuario logeado no encontrado"));
149
150 // Valida que no pueda desactivar el usuario logeado
151 if (usuarioLogeado.getId().equals(id)) {
152 attributes.addFlashAttribute( attributeNames: "error", attributeValue: "No puedes desactivar tu propio usuario.");
153 return "redirect:/usuarios";
154 }
155
156 // Desactivación de otro usuario
157 Usuario usuarioADesactivar = usuarioService.buscarPorId(id)
158 | .orElseThrow(() -> new IllegalArgumentException("Usuario no encontrado"));
159
160 usuarioADesactivar.setStatus(0); // Se pasa a inactivo
161 usuarioService.crearOEditar(usuarioADesactivar); // Se guarda
162
163 attributes.addFlashAttribute( attributeNames: "msg", attributeValue: "Usuario desactivado correctamente.");
164 return "redirect:/usuarios"; // Redirige al listado de usuarios activos
165 }
166
167 /**
168 * Accion REACTIVE USERS
169 */
170 @GetMapping("/inactive") no usages & AlejandroJuarez002
171 public String desactivarInactivo(Model model) {

```

```

27 public class UsuarioController {
120 public String showDesactivateView(@PathVariable("id") Integer id, Model model) {
138 }
139
140 @PostMapping("/desactivate/{id}") no usages & AlejandroJuarez002
141 public String desactivate(@PathVariable("id") Integer id, RedirectAttributes attributes) {
142 // Obtiene el usuario autenticado desde Spring Security
143 Authentication auth = SecurityContextHolder.getContext().getAuthentication();
144 String username = auth.getName();
145
146 // Busca en base de datos el usuario logeado
147 Usuario usuarioLogeado = usuarioService.buscarPorNombreUsuario(username)
148 | .orElseThrow(() -> new IllegalArgumentException("Usuario logeado no encontrado"));
149
150 // Valida que no pueda desactivar el usuario logeado
151 if (usuarioLogeado.getId().equals(id)) {
152 attributes.addFlashAttribute( attributeNames: "error", attributeValue: "No puedes desactivar tu propio usuario.");
153 return "redirect:/usuarios";
154 }
155
156 // Desactivación de otro usuario
157 Usuario usuarioADesactivar = usuarioService.buscarPorId(id)
158 | .orElseThrow(() -> new IllegalArgumentException("Usuario no encontrado"));
159
160 usuarioADesactivar.setStatus(0); // Se pasa a inactivo
161 usuarioService.crearOEditar(usuarioADesactivar); // Se guarda
162
163 attributes.addFlashAttribute( attributeNames: "msg", attributeValue: "Usuario desactivado correctamente.");
164 return "redirect:/usuarios"; // Redirige al listado de usuarios activos
165 }
166
167 /**
168 * Accion REACTIVE USERS
169 */
170 @GetMapping("/inactive") no usages & AlejandroJuarez002
171 public String desactivarInactivo(Model model) {

```



```

27 public class UsuarioController {
171 public String inactiveUsers(Model model,
183     List<Integer> pageNumbers = IntStream.rangeClosed(1, totalPages).toIntStream()
184     .boxed() .stream<Integer>()
185     .collect(Collectors.toList());
186     model.addAttribute("pageNumbers", pageNumbers);
187 }
188
189     return "usuario/inactive_index"; // la vista para listar usuarios inactivos
190 }
191
192 @GetMapping("/activate/{id}") no usages & AlejandroJuarez002
193 @ public String showActivateView(@PathVariable("id") Integer id, Model model) {
194     Usuario usuario = usuarioService.buscarPorId(id)
195     .orElseThrow(() -> new IllegalArgumentException("Usuario no encontrado"));
196     model.addAttribute("usuario", usuario);
197     return "usuario/activate"; // vista de confirmación de reactivación
198 }
199
200 @PostMapping("/activate/{id}") no usages & AlejandroJuarez002
201 @ public String activate(@PathVariable("id") Integer id, RedirectAttributes attributes) {
202     Usuario usuario = usuarioService.buscarPorId(id)
203     .orElseThrow(() -> new IllegalArgumentException("Usuario no encontrado"));
204
205     usuario.setStatus(1); // Se pasa a activo
206     usuarioService.crearOEditar(usuario); // Se guarda
207
208     attributes.addFlashAttribute("msg", "Usuario reactivado correctamente.");
209     return "redirect:/usuarios/inactive"; // Vuelve al listado de inactivos
210 }
211 }

```