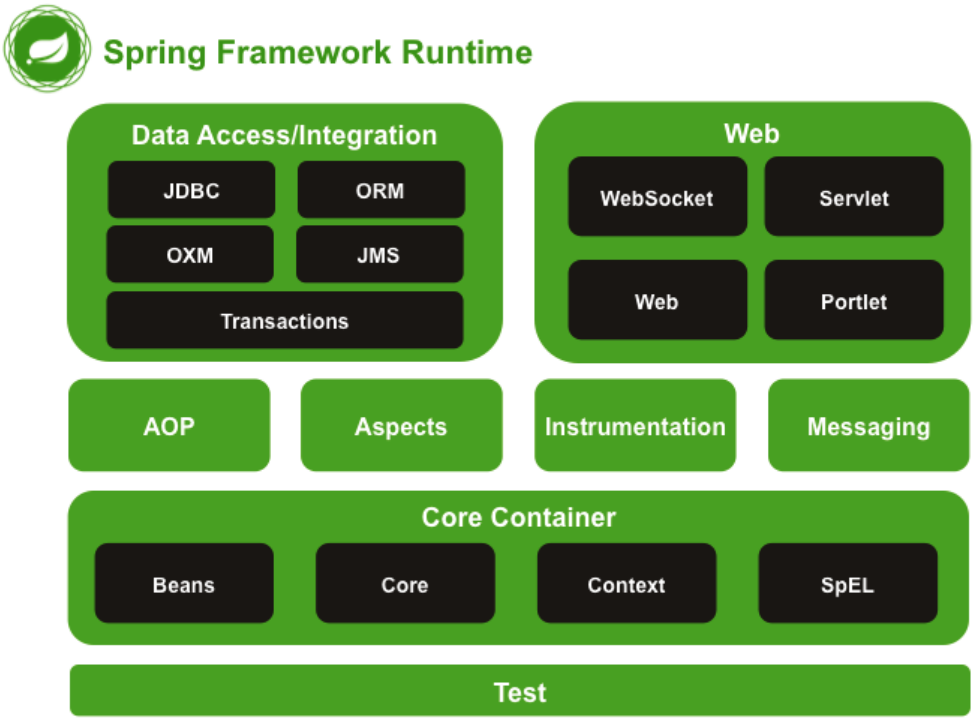
GUSTAVO CORONEL
DESARROLLA SOFTWARE

DESARROLLO WEB
CON SPRING BOOT



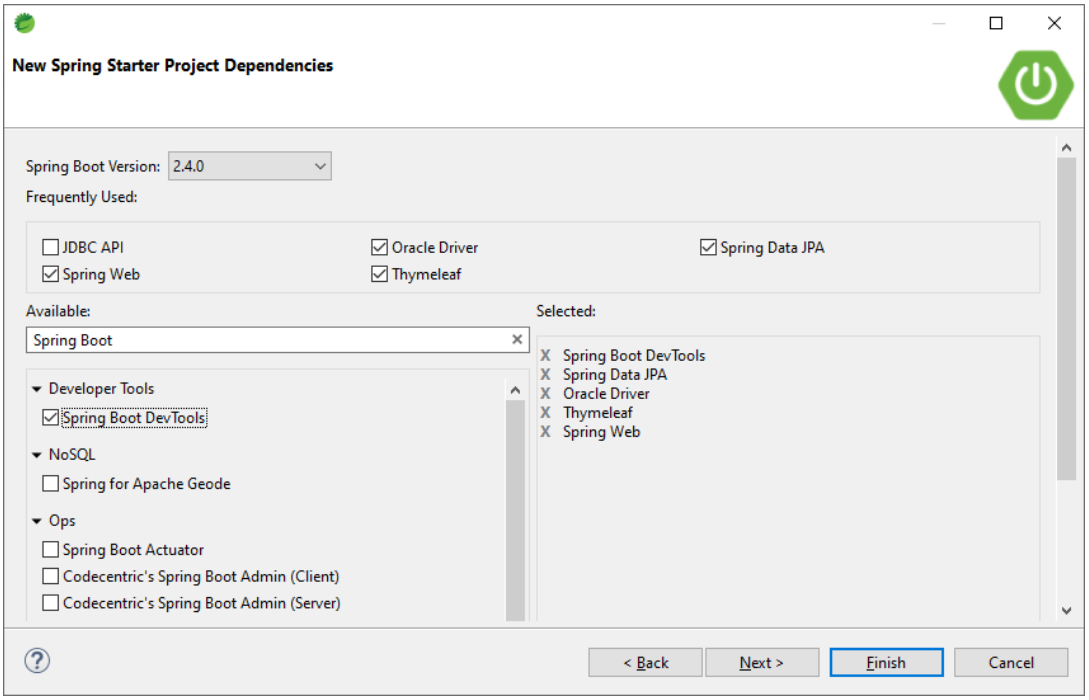
UNIDAD 09
DESARROLLO DE UN CRUD

Eric Gustavo Coronel Castillo
www.desarrollasoftware.com
gcoronelc@gmail.com
I N S T R U C T O R

CONTENIDO

INICIAR PROYECTO

Creación del proyecto



Dependencias

```
<dependencies>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
   </dependency>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-thymeleaf</artifactId>
   </dependency>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
   </dependency>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
```

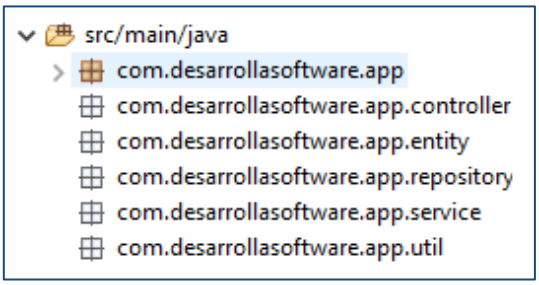
```
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>com.oracle.database.jdbc</groupId>
        <artifactId>ojdbc8</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Parámetros de Conexión

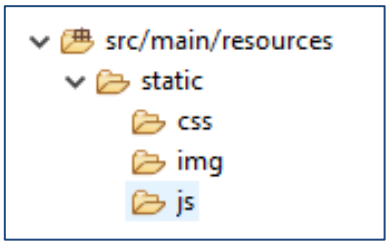
Estos parámetros se deben incluir en el archivo de propiedades (application.properties), también puedes establecer el puerto del servidor.

```
server.port=9090

# Oracle settings
spring.datasource.url=jdbc:oracle:thin:@localhost:1521/XE
spring.datasource.username=ventas
spring.datasource.password=admin
spring.jpa.database-platform=org.hibernate.dialect.Oracle12cDialect
spring.jpa.hibernate.ddl-auto=none
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

Estructura de paquetes



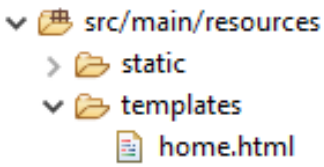
Recursos estáticos



PAGINA DE INICIO

Archivo home.html

Ubicación



Archivo

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Página de Inicio</title>
</head>
<body>
   <h1>Hola desde Spring Boot</h1>
</body>
</html>
```

Controlador

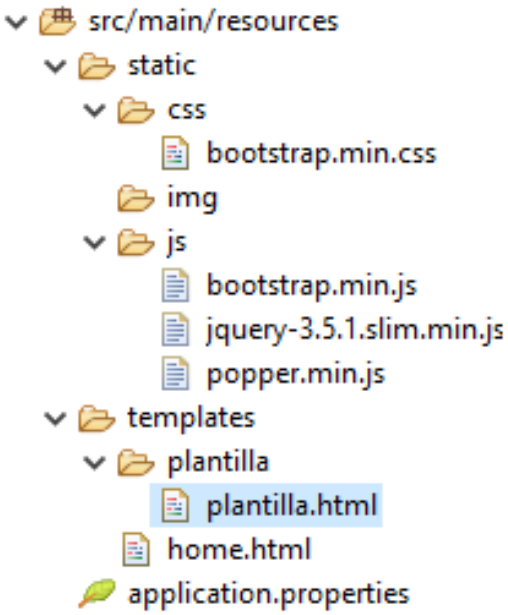
```
package com.desarrollasoftware.app.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class AppController {

   @GetMapping({"/","/home"})
   public String home() {
      return "home";
   }
```

```
}
```

PLANTILLA

Archivo platilla.html

Ubicación



Archivo

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head th:fragment="head">
<meta charset="UTF-8">
<!-- Bootstrap CSS -->
<link rel="stylesheet" th:href="@{/css/bootstrap.min.css}">
<title>APP CRUD</title>
</head>
<body>
   <!-- Barra de navegación -->
   <header th:fragment="header">
      <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
         <div class="container">
```

```
         <a class="navbar-brand" href="#"> <strong>MyLogo</strong>
         </a>

         <button class="navbar-toggler" type="button" data-toggle="collapse"
            data-target="#navbarSupportedContent"
            aria-controls="navbarSupportedContent" aria-expanded="false"
            aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
         </button>

         <!-- ENLACES DEL MENU -->
         <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav mr-auto">
               <li class="nav-item"><a class="nav-link"
                  th:href="@{/home}">Inicio</a></li>
               <li class="nav-item"><a class="nav-link"
                  href="#">Productos</a></li>
            </ul>
         </div>

      </div>
   </nav>
</header>

<!-- Contenido -->
<div class="container"></div>

<!-- Pie de pagina -->
<footer th:fragment="footer"
   class="bg-dark text-center text-white fixed-bottom">

   <div class="container">
      <p>Copyright Desarrolla Software &copy;2020</p>
   </div>

   <script type="text/javascript"
      th:src="@{/js/jquery-3.5.1.slim.min.js}"></script>
   <script type="text/javascript" th:src="@{/js/popper.min.js}"></script>
   <script type="text/javascript" th:src="@{/js/bootstrap.min.js}"></script>

</footer>
```

```
</body>
</html>
```

Archivo home.html

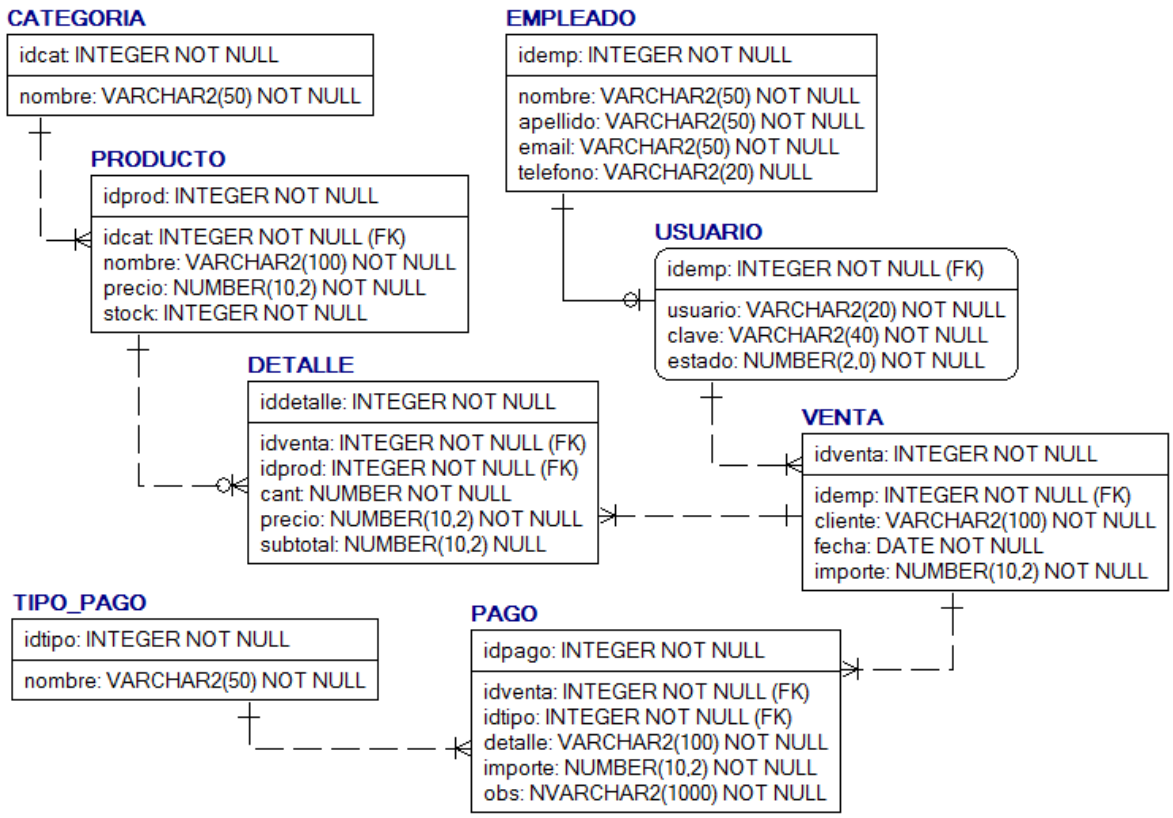
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head th:replace="plantilla/plantilla :: head">
</head>
<body>

   <header th:replace="plantilla/plantilla :: header">
   </header>

   <div class="container">
      <h1>Página principal</h1>
   </div>

   <footer th:replace="plantilla/plantilla :: footer">
   </footer>
</body>
</html>
```

BASE DE DATOS

URL del modelo de dato

```
https://github.com/gcoronelc/databases
```

Modelo de datos VENTAS



Creación del esquema

Proceda a crear el esquema en Oracle:

```
SQL> @D:\databases\Ventas\Oracle\Modelo\Modelo.SQL
```

Tabla CATEGORIA

```
SQL> desc categoria;
 Name                          Null?    Type
 ----------------- -------- ----------------------------
 IDCAT                         NOT NULL NUMBER(38)
 NOMBRE                        NOT NULL VARCHAR2(100)
```

Tabla PRODUCTO

```
SQL> describe producto;
 Name                            Null?    Type
 ---------------------- -------- ----------------------------
 IDPROD                          NOT NULL NUMBER(38)
 IDCAT                           NOT NULL NUMBER(38)
 NOMBRE                          NOT NULL VARCHAR2(100)
 PRECIO                          NOT NULL NUMBER(10,2)
 STOCK                           NOT NULL NUMBER(38)
```

CLASES ENTITY

Clase Categoría

```
package com.desarrollasoftware.app.entity;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "categoria")
public class Categoria implements Serializable {

   private static final long serialVersionUID = 1L;

   @Id
   @Column(name = "idcat")
   private Long id;

   @Column(name = "nombre")
   private String nombre;

   public Categoria() {
   }

   public Long getId() {
      return id;
   }

   public void setId(Long id) {
      this.id = id;
   }

   public String getNombre() {
      return nombre;
   }
```

```
   public void setNombre(String nombre) {
      this.nombre = nombre;
   }

   @Override
   public String toString() {
      return "Categoria [id=" + id + ", nombre=" + nombre + "]";
   }

}
```

Clase Producto

```
package com.desarrollasoftware.app.entity;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;

@Entity
@Table(name = "producto")
@SequenceGenerator(name = "sq_producto", sequenceName = "sq_producto",
allocationSize = 1)
public class Producto implements Serializable {

   private static final long serialVersionUID = 1L;

   @Id
   @Column(name = "idprod")
   @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "sq_producto")
   private Long id;
```

```
    @Column(name = "nombre")
    private String nombre;

    @ManyToOne
    @JoinColumn(name = "idcat")
    private Categoria categoria;

    @Column(name = "precio")
    private Double precio;

    @Column(name = "stock")
    private Long stock;

    public Producto() {
       this.id = 0L;
       this.precio = 0.0;
       this.stock = 0L;
    }

    public Long getId() {
       return id;
    }

    public void setId(Long id) {
       this.id = id;
    }

    public String getNombre() {
       return nombre;
    }

    public void setNombre(String nombre) {
       this.nombre = nombre;
    }

    public Categoria getCategoria() {
       return categoria;
    }

    public void setCategoria(Categoria categoria) {
       this.categoria = categoria;
    }
```

```
  public Double getPrecio() {
     return precio;
  }

  public void setPrecio(Double precio) {
     this.precio = precio;
  }

  public Long getStock() {
     return stock;
  }

  public void setStock(Long stock) {
     this.stock = stock;
  }

  @Override
  public String toString() {
     return "Producto [id=" + id + ", nombre=" + nombre
          + ", categoria=" + categoria + ", precio=" + precio
          + ", stock=" + stock + "]";
  }

}
```

IMPLEMENTACIÓN DE LOS SERVICIOS

Capa Repository

Categoría

```
package com.desarrollasoftware.app.repository;

import org.springframework.data.repository.CrudRepository;

import com.desarrollasoftware.app.entity.Categoria;

public interface CategoriaRepository extends CrudRepository<Categoria, Long>{

}
```

Producto

```
package com.desarrollasoftware.app.repository;

import org.springframework.data.repository.CrudRepository;

import com.desarrollasoftware.app.entity.Producto;

public interface ProductoRepository extends CrudRepository<Producto, Long>{

}
```

Capa Service

Categoría

La interface

```
package com.desarrollasoftware.app.service;

import java.util.List;
import com.desarrollasoftware.app.entity.Categoria;

public interface CategoriaService {

   public List<Categoria> listarTodos();
}
```

La implementación

```
package com.desarrollasoftware.app.service.impl;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.desarrollasoftware.app.entity.Categoria;
import com.desarrollasoftware.app.repository.CategoriaRepository;
import com.desarrollasoftware.app.service.CategoriaService;

@Service
public class CategoriaServiceImpl implements CategoriaService{

   @Autowired
   private CategoriaRepository repository;

   @Override
   public List<Categoria> listarTodos() {
      List<Categoria> lista = (List<Categoria>) repository.findAll();
      return lista;
   }
```

```
}
```

Producto

La interface

```
package com.desarrollasoftware.app.service;

import java.util.List;

import com.desarrollasoftware.app.entity.Producto;

public interface ProductoService {

   public List<Producto> listarTodos();
   public void grabar(Producto producto);
   public Producto buscarPorId(Long id);
   public void eliminar(Long id);

}
```

La implementación

```
package com.desarrollasoftware.app.service.impl;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.desarrollasoftware.app.entity.Producto;
import com.desarrollasoftware.app.repository.ProductoRepository;
import com.desarrollasoftware.app.service.ProductoService;

@Service
public class ProductoServiceImpl implements ProductoService {

   @Autowired
   private ProductoRepository repository;
```

```
    @Override
    public List<Producto> listarTodos() {
        List<Producto> lista = (List<Producto>) repository.findAll();
        return lista;
    }

    @Override
    public void grabar(Producto producto) {
        repository.save(producto);
    }

    @Override
    public Producto buscarPorId(Long id) {
        Producto bean = repository.findById(id).orElse(null);
        return bean;
    }

    @Override
    public void eliminar(Long id) {
        repository.deleteById(id);
    }

}
```

LISTADO DE PRODUCTOS

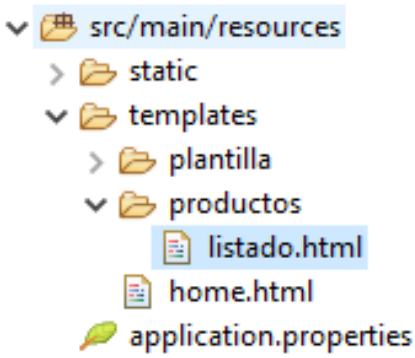
Diseño

Controlador

```
package com.desarrollasoftware.app.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;

import com.desarrollasoftware.app.entity.Producto;
import com.desarrollasoftware.app.service.ProductoService;

@Controller
@RequestMapping("/productos")
public class ProductoController {

   @Autowired
```

```
   private ProductoService productoService;

   @GetMapping({"/","todos"})
   public String listar(Model model) {
      List<Producto> lista = productoService.listarTodos();
      model.addAttribute("titulo", "LISTA DE PRODUCTOS");
      model.addAttribute("productos", lista);
      return "/productos/listado";
   }

}
```

Vista




```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head th:replace="plantilla/plantilla :: head">
</head>
<body>

   <header th:replace="plantilla/plantilla :: header">
   </header>

   <div class="container">
      <h1 th:text="${titulo}"></h1>

      <table class="table">
         <thead class="thead-light">
            <tr>
               <th scope="col">ID</th>
```

```
            <th scope="col">CATEGORIA</th>
            <th scope="col">NOMBRE</th>
            <th scope="col">PRECIO</th>
            <th scope="col">STOCK</th>
          </tr>
        </thead>
        <tbody>
          <tr th:each="pr:${productos}">
            <th scope="row" th:text="${pr.id}"></th>
            <td th:text="${pr.categoria.nombre}"></td>
            <td th:text="${pr.nombre}"></td>
            <td th:text="${pr.precio}"></td>
            <td th:text="${pr.stock}"></td>
          </tr>
        </tbody>
      </table>

    </div>

    <footer th:replace="plantilla/plantilla :: footer">
    </footer>
</body>
</html>
```

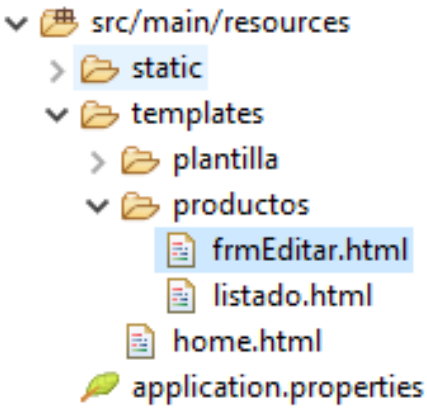
Menú de opciones

Actualizar el menú en el archivo **plantilla.htnl**:

```
    <!-- ENLACES DEL MENU -->
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item"><a class="nav-link"
          th:href="@{/home}">Inicio</a></li>
        <li class="nav-item"><a class="nav-link"
          th:href="@{/productos/todos}">Productos</a></li>
      </ul>
    </div>
```

CREACIÓN DE NUEVOS PRODUCTOS

Diseño del formulario



Controlador para cargar el formulario

```
  @GetMapping("/crear")
  public String crear(Model model) {

     Producto producto = new Producto();
     List<Categoria> listaCategorias = categoriaService.listarTodos();

     model.addAttribute("titulo", "NUEVO PRODUCTO");
     model.addAttribute("producto", producto);
     model.addAttribute("categorias", listaCategorias);

     return "/productos/frmEditar";

  }
```

Formulario frmEditar.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head th:replace="plantilla/plantilla :: head">
</head>
<body>

   <header th:replace="plantilla/plantilla :: header"> </header>

   <div class="container mt-2">
      <form  th:object="${producto}" method="post">
         <div class="card bg-light">

            <div class="card-header bg-info text-white" th:text="${titulo}"></div>

            <div class="card-body">

               <div class="form-group row">
                  <label class="col-md-2" for="categoria">Categoría:</label> <select
                     th:field="*{categoria}"
                     class="form-control form-control-sm col-md-10" id="categoria">
                     <option th:each="categoria:${categorias}"
```

```
                    th:value="${categoria.id}" th:text="${categoria.nombre}" />
               </select>
            </div>

            <div class="form-group row">
               <label class="col-md-2" for="nombre">Nombre:</label> <input
                  type="text" th:field="*{nombre}"
                  class="form-control form-control-sm col-md-10" id="nombre"
                  placeholder="Nombres del producto"> <small> </small>
            </div>

            <div class="form-group row">
               <label class="col-md-2" for="precio">Precio:</label> <input
                  type="text" th:field="*{precio}"
                  class="form-control form-control-sm col-md-10" id="precio"
                  placeholder="Precio del producto"> <small></small>
            </div>

            <div class="form-group row">
               <label class="col-md-2" for="stock">Stock:</label> <input
                  type="text" th:field="*{stock}"
                  class="form-control form-control-sm col-md-10" id="stock"
                  placeholder="Stock del producto"> <small></small>
            </div>

         </div>

         <div class="card-footer bg-light">
            <input type="submit" class="btn btn-primary btn-sm" value="Guardar">
         </div>

      </div>
   </form>
</div>

<footer th:replace="plantilla/plantilla :: footer"> </footer>
</body>
</html>
```

Botón para cargar el formulario

En el archivo listado.html debes agregar el enlace para crear el botón **Nuevo Producto**:

```
<a class="btn btn-primary btn-sm" th:href="@{/productos/crear}"
   th:text="'Nuevo Producto'"></a>
```

Controlador para grabar producto

```
@PostMapping("/grabar")
public String guardar(@ModelAttribute Producto producto) {

   productoService.grabar(producto);
   System.out.println("Producto grabado con exito!");
   return "redirect:/productos/";

}
```

EDITAR Y ELIMINAR PRODUCTO

Controller para editar un producto

```
@GetMapping("/editar/{id}")
public String editar(@PathVariable("id") Long idProd, Model model) {

     Producto producto = productoService.buscarPorId(idProd);
     List<Categoria> listaCategorias = categoriaService.listarTodos();

     model.addAttribute("titulo", "EDITAR PRODUCTO (" + idProd + ")");
     model.addAttribute("producto", producto);
     model.addAttribute("categorias", listaCategorias);

     return "/productos/frmEditar";

}
```

Puedes editar un producto ingresando esta URL:

```
http://localhost:9090/productos/editar/1005
```

Después de modificar los datos de un producto y hacer click en el botón grabar, ¿qué sucede?

Agregar la siguiente línea de código al formulario de edición:

```
  <input type="hidden" th:field="*{id}" id="id"/>
```

Controller para eliminar un producto

```
@GetMapping("/eliminar/{id}")
public String eliminar(@PathVariable("id") Long idProd) {

   productoService.eliminar(idProd);
   System.out.println("Producto eliminado con exito!");
   return "redirect:/productos/";

}
```

Puede eliminar un producto utilizando la siguiente URL:

```
http://localhost:9090/productos/eliminar/1033
```

Botones de edición y eliminación

En la tabla de productos debes incluir dos columnas, una para editar un registro y otra para eliminar un registro.

```
<table class="table">
   <thead class="thead-light">
      <tr>
         <th scope="col">ID</th>
         <th scope="col">CATEGORIA</th>
         <th scope="col">NOMBRE</th>
         <th scope="col">PRECIO</th>
         <th scope="col">STOCK</th>
         <th scope="col">EDITAR</th>
         <th scope="col">ELIMINAR</th>
      </tr>
   </thead>
   <tbody>
      <tr th:each="pr:${productos}">
         <th scope="row" th:text="${pr.id}"></th>
         <td th:text="${pr.categoria.nombre}"></td>
         <td th:text="${pr.nombre}"></td>
         <td th:text="${pr.precio}"></td>
         <td th:text="${pr.stock}"></td>
         <td><a class="btn btn-success btn-sn"
            th:href="@{/productos/editar/} + ${pr.id}" th:text="'Editar'"
            title="Editar producto."></a></td>
         <td><a class="btn btn-danger btn-sn"
            th:href="@{/productos/eliminar/} + ${pr.id}" th:text="'Eliminar'"
            title="Eliminar producto."></a></td>
      </tr>
   </tbody>
</table>
```

Ya puede probar la aplicación, tanto la opción de editar y la opción de eliminar deben funcionar correctamente.

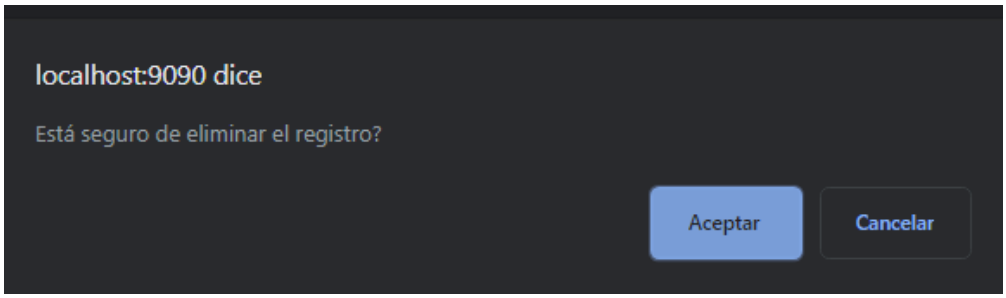
¿Qué notas en las nuevas funcionalidades?

Solicitar la confirmación

En este caso debes modificar el botón eliminar como se ilustra a continuación:

```
<td>
   <a class="btn btn-danger btn-sn"
   th:href="@{/productos/eliminar/} + ${pr.id}" th:text="'Eliminar'"
   title="Eliminar producto."
   onclick="return confirm('Está seguro de eliminar el registro?');"></a>
</td>
```

Ahora, cuando intentes eliminar un registro debes confirmar la acción con una ventana similar a la que se muestra a continuación:



VALIDACIÓN DEL FORMULARIO

Librería

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```

La entidad producto

Atributo nombre

```
@Column(name = "nombre")
@NotEmpty
private String nombre;
```

Atributo precio

```
@Column(name = "precio")
@NotNull
@Min(value = 0)
private double precio;
```

Atributo stock

```
@Column(name = "stock")
@NotNull
@Min(value = 0)
private long stock;
```

Grabar producto

```
@PostMapping("/grabar")
public String guardar(@Valid @ModelAttribute Producto producto,
BindingResult result, Model model) {

   if (result.hasErrors()) {
      System.err.println("Se presentaron errores en el formulario!");
      String titulo = "NUEVO PRODUCTO";
      if (producto.getId() > 0) {
         titulo = "EDITAR PRODUCTO (" + producto.getId() + ")";
      }
      List<Categoria> listaCategorias = categoriaService.listarTodos();
      model.addAttribute("titulo", titulo);
      model.addAttribute("producto", producto);
      model.addAttribute("categorias", listaCategorias);
      return "/productos/frmEditar";
   }

   productoService.grabar(producto);
   System.out.println("Producto grabado con exito!");

   return "redirect:/productos/";
}
```

Campos del formulario

Campo nombre

```
<div class="form-group row">
   <label class="col-md-2" for="nombre">Nombre:</label>
   <div class="col-md-10">
      <input type="text" th:field="*{nombre}"
         class="form-control form-control-sm" id="nombre"
         placeholder="Nombres del producto" /> <small
         class="form-text text-danger"
         th:if="${#fields.hasErrors('nombre')}" th:errors="*{nombre}"></small>
   </div>
</div>
```

Campo precio

```
<div class="form-group row">
   <label class="col-md-2" for="precio">Precio:</label>
   <div class="col-md-10">
      <input type="text" th:field="*{precio}"
         class="form-control form-control-sm" id="precio"
         placeholder="Precio del producto" /> <small
         class="form-text text-danger"
         th:if="${#fields.hasErrors('precio')}" th:errors="*{precio}"></small>
   </div>
</div>
```

Campo stock

```
<div class="form-group row">
   <label class="col-md-2" for="stock">Stock:</label>
   <div class="col-md-10">
      <input type="text" th:field="*{stock}"
         class="form-control form-control-sm" id="stock"
         placeholder="Stock del producto" /> <small
         class="form-text text-danger"
         th:if="${#fields.hasErrors('stock')}" th:errors="*{stock}"></small>
   </div>
</div>
```

En este momento ya puedes probar el formulario:

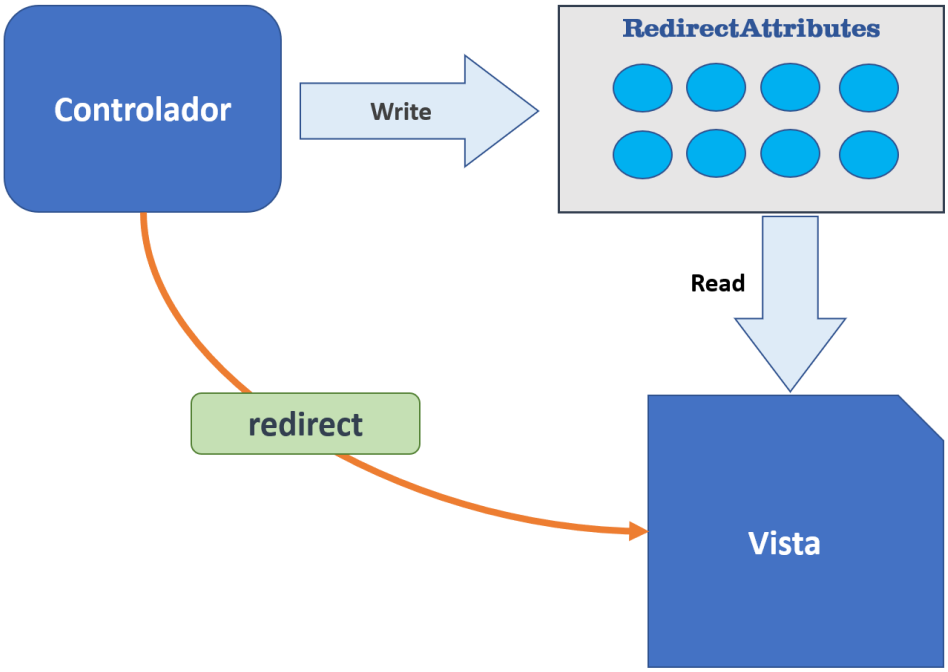
Personalizando los mensajes de error

En la carpeta **resources** debes crear el archivo **messages.properties** con los mensajes respectivos:

```
#Validación del producto
NotEmpty.producto.nombre=El nombre del producto es requerido.
NotNull.producto.precio=El precio del producto es requerido.
NotNull.producto.stock=El stock del producto es requerido.
DecimalMin.producto.precio=El precio no puede tomar valores negativos
Min.producto.stock=El stock no puede tomar valores negativos
typeMismatch.producto.precio=El precio del producto es un numero decimal
typeMismatch.producto.stock=El stock del producto es un numero entero
```

Ahora puedes probar nuevamente el formulario:

MENSAJES DE ALERTA



Estructura

En el controller

```
@RequestMapping("/controlador")
public String controlador(RedirectAttributes atributos) {

   atributos.addFlashAttribute("atributo","mensaje al usuario")

   return "redirect:/vista";
}
```

En la Vista

```
   <etiqueta th:text="${atributo}"></etiqueta>
```

En la plantilla

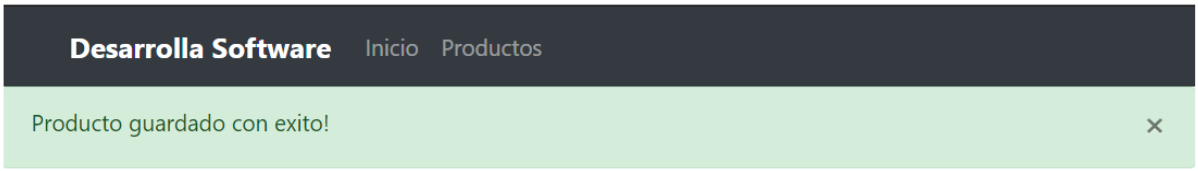
Se incluye las alertas para diferentes tipos de mensajes.

```
<!-- MENSAJES -->

<div class="alert alert-success alert-dismissable"
   th:if="${success != null}">
   <label th:text="${success}"></label>
   <button type="button" class="close" data-dismiss="alert">&times;</button>
</div>

<div class="alert alert-danger alert-dismissable"
   th:if="${error != null}">
   <label th:text="${error}"></label>
   <button type="button" class="close" data-dismiss="alert">&times;</button>
</div>

<div class="alert alert-warning alert-dismissable"
   th:if="${warning != null}">
   <label th:text="${warning}"></label>
   <button type="button" class="close" data-dismiss="alert">&times;</button>
</div>

<div class="alert alert-info alert-dismissable"
   th:if="${info != null}">
   <label th:text="${info}"></label>
   <button type="button" class="close" data-dismiss="alert">&times;</button>
</div>
```

Alerta al grabar el producto

```
@PostMapping("/grabar")
public String guardar(@Valid @ModelAttribute Producto producto,
BindingResult result, Model model, RedirectAttributes atributos) {

   if (result.hasErrors()) {
      System.err.println("Se presentaron errores en el formulario!");
      String titulo = "NUEVO PRODUCTO";
      if (producto.getId() > 0) {
         titulo = "EDITAR PRODUCTO (" + producto.getId() + ")";
      }
      List<Categoria> listaCategorias = categoriaService.listarTodos();
      model.addAttribute("titulo", titulo);
      model.addAttribute("producto", producto);
      model.addAttribute("categorias", listaCategorias);
      return "/productos/frmEditar";
   }

   productoService.grabar(producto);
   System.out.println("Producto grabado con exito!");
   atributos.addFlashAttribute("success", "Producto guardado con exito!");

   return "redirect:/productos/";
}
```

El resultado es el siguiente:



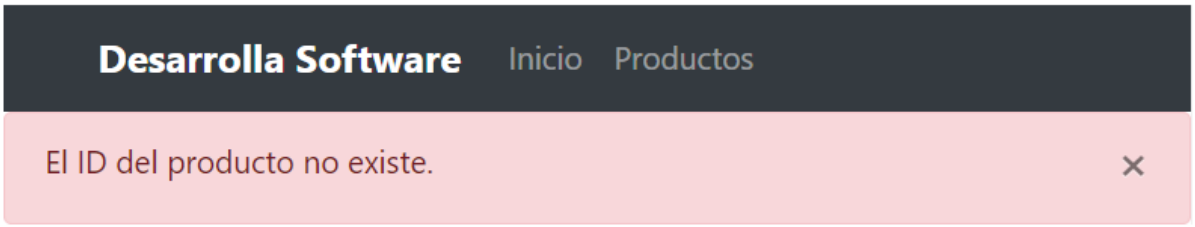
Al editar un producto

```
@GetMapping("/editar/{id}")
public String editar(@PathVariable("id") Long idProd, Model model,
RedirectAttributes atributos) {

   // Id de producto invalido
   if( idProd <= 0) {
      atributos.addFlashAttribute("error", "El id del producto es incorrecto.");
      return "redirect:/productos/todos";
   }

   Producto producto = productoService.buscarPorId(idProd);

   // Id de producto no existe
   if( producto == null ) {
      atributos.addFlashAttribute("error", "El id del producto no existe.");
      return "redirect:/productos/todos";
   }

   List<Categoria> listaCategorias = categoriaService.listarTodos();
   model.addAttribute("titulo", "EDITAR PRODUCTO (" + idProd + ")");
   model.addAttribute("producto", producto);
   model.addAttribute("categorias", listaCategorias);

   return "/productos/frmEditar";

}
```

Para probar su funcionamiento, debes intentar editar un producto que no existe:

```
http://localhost:9090/productos/editar/7777
```

A continuación, tienes el resultado:



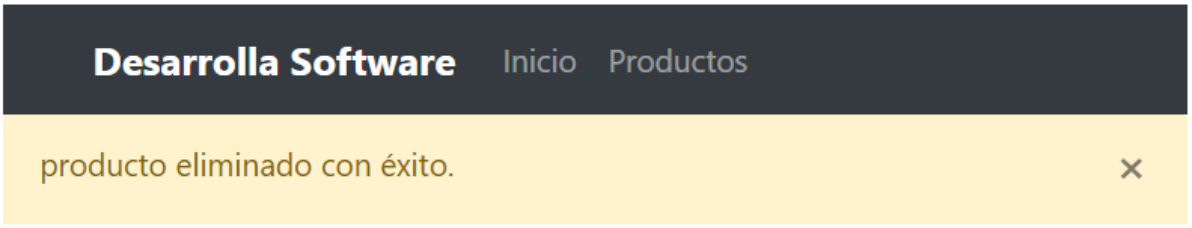
Al eliminar un producto

```
@GetMapping("/eliminar/{id}")
public String eliminar(@PathVariable("id") Long idProd,
RedirectAttributes atributos) {

   // Id de producto invalido
   if (idProd <= 0) {
      atributos.addFlashAttribute("error", "El ID del producto es incorrecto.");
      return "redirect:/productos/todos";
   }

   Producto producto = productoService.buscarPorId(idProd);

   // Id de producto no existe
   if (producto == null) {
      atributos.addFlashAttribute("error", "El ID del producto no existe.");
      return "redirect:/productos/todos";
   }

   productoService.eliminar(idProd);
   System.out.println("Producto eliminado con exito!");
   atributos.addFlashAttribute("warning", "producto eliminado con éxito.");
   return "redirect:/productos/";

}
```

Para probar su funcionamiento, debes intentar editar un producto existente:

```
http://localhost:9090/productos/editar/1020
```

A continuación, tienes el resultado:

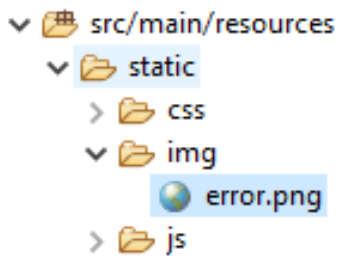


OTROS MENSAJES DE ERROR

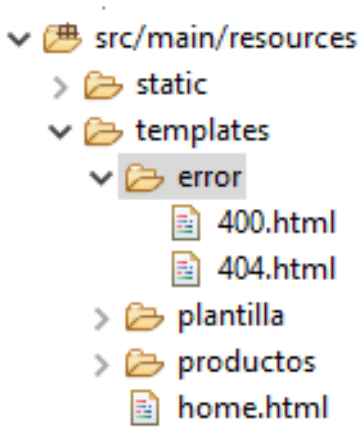
Estructura

Imagen de error

En la carpeta **img** debes agregar tu imagen para las páginas de error:



Estructura de carpetas



Puedes incluir un mensaje en las páginas 400.html y 404.html para probar el funcionamiento de estas páginas.

Error HTTP-400

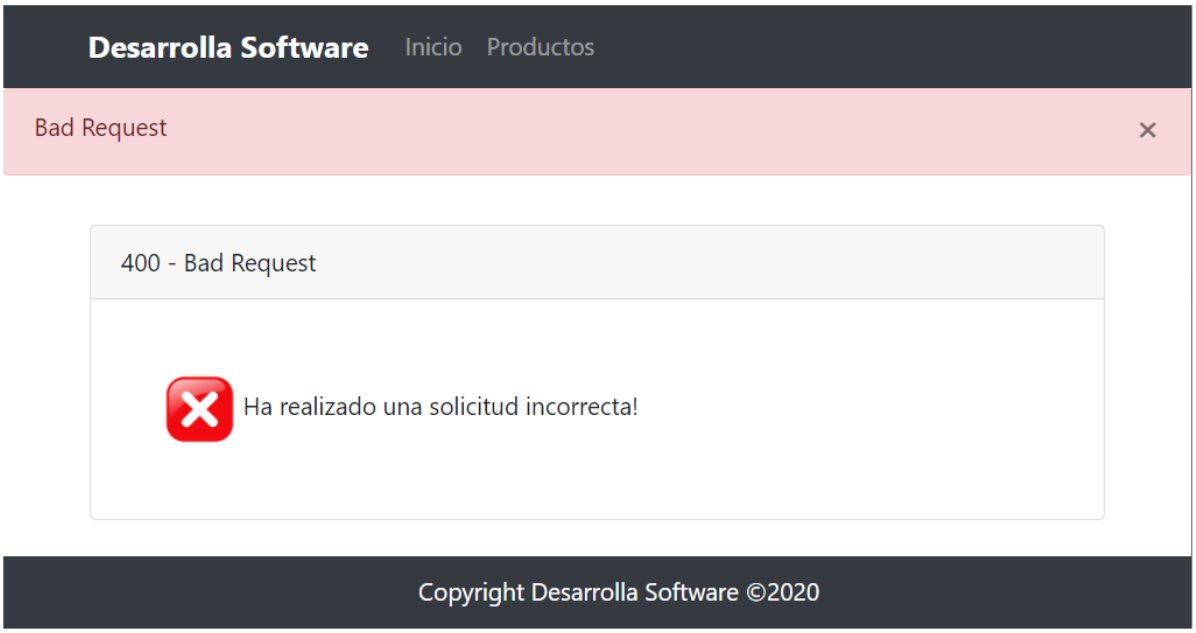
Este tipo de error se presenta cuando se realiza una solicitud incorrecta, por ejemplo:

```
http://localhost:9090/productos/editar/abc
```

A continuación, tienes la codificación de la página:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head th:replace="plantilla/plantilla :: head">
</head>
<body>
   <header th:replace="plantilla/plantilla :: header"></header>

   <div class="container p-3">

      <div class="card">
         <div class="card-header">
              <span th:text="${status}"></span> - <span th:text="${error}"></span>
         </div>
         <div class="car-body p-5">
            <div>
               <img th:src="@{/img/error.png}" width="48" height="48" class="d-
inline"/>
               <p class="card-text d-inline">Ha realizado una solicitud
incorrecta!</p>
            </div>

         </div>
      </div>
   </div>

   <footer th:replace="plantilla/plantilla :: footer"></footer>

</body>
</html>
```

Este sería el resultado:



Error HTTP 404

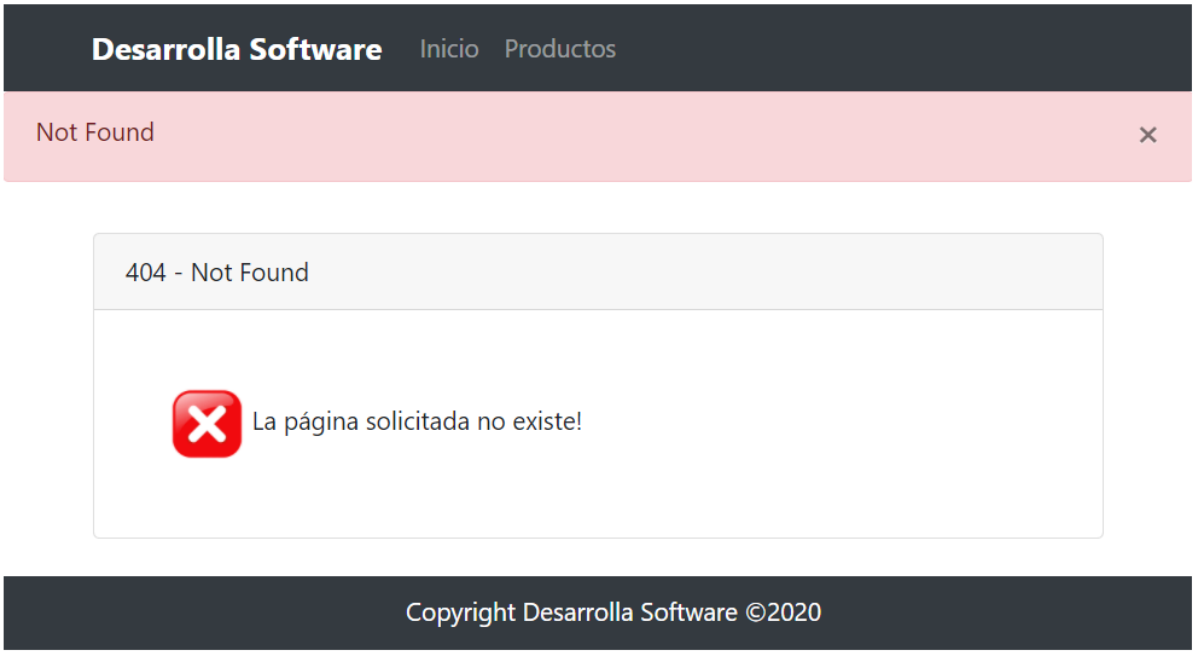
Este tipo de error se presenta cuando se realiza una página que no existe, por ejemplo:

```
http://localhost:9090/abc
```

A continuación, tienes la codificación de la página:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head th:replace="plantilla/plantilla :: head">
</head>
<body>
   <header th:replace="plantilla/plantilla :: header"></header>

   <div class="container p-3">

      <div class="card">
         <div class="card-header">
              <span th:text="${status}"></span> - <span th:text="${error}"></span>
         </div>
         <div class="car-body p-5">
            <div>
               <img th:src="@{/img/error.png}" width="48" height="48"
                  class="d-inline"/>
               <p class="card-text d-inline">La página solicitada no existe!</p>
            </div>

         </div>
      </div>
   </div>

   <footer th:replace="plantilla/plantilla :: footer"></footer>

</body>
</html>
```

Este sería el resultado:



EJERCICIO PROPUESTO

Desarrollar el CRUD de la tabla Empleado o un caso de los sistemas a su cargo en la empresa.