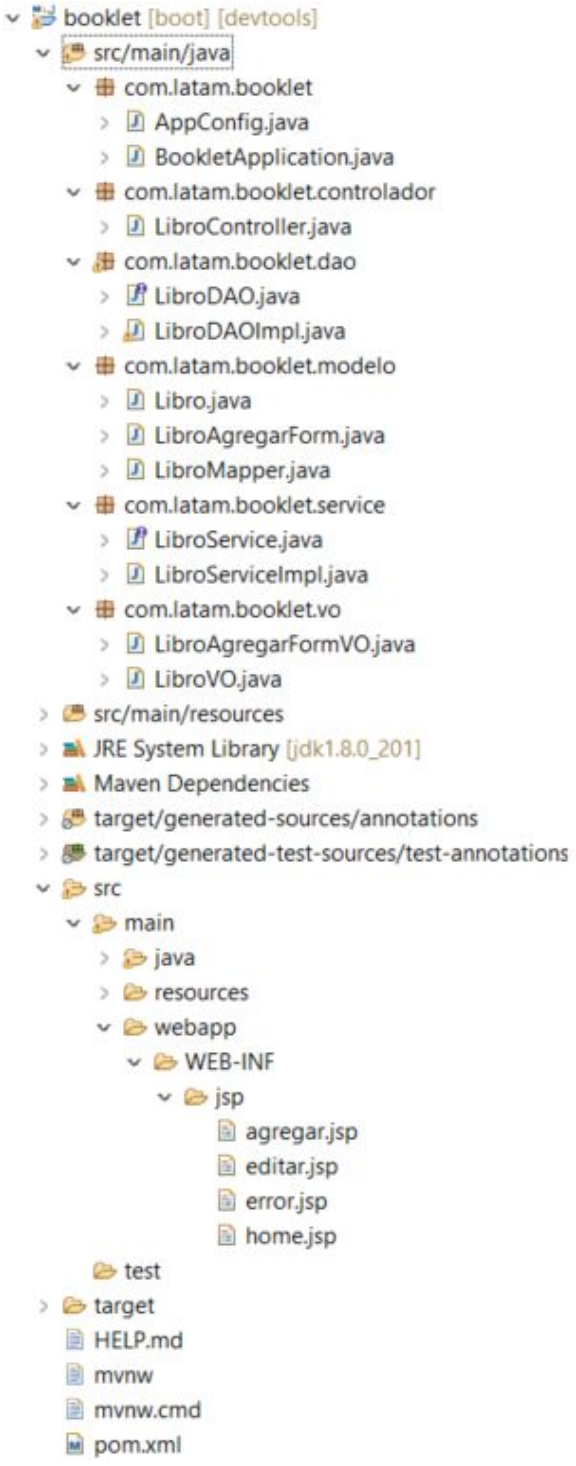
Solución Desafío - Biblioteca Booklet

Estructura del proyecto



Las Capas

Capa de persistencia

Interfaz e implementación con métodos necesarios para realizar todas las operaciones sobre la base de datos

```
package com.latam.booklet.dao;
import java.util.List;
import com.latam.booklet.modelo.Libro;
public interface LibroDAO {
  public List<Libro> findAll();
  public Libro findById(Integer idLibro);
  public int add(Libro libro);
  public int update(Libro libro);
  public int delete(Libro libro);
  List<Libro> findByTituloOrAutor(String textoBuscado);
}
```

```
package com.latam.booklet.dao;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.sql.DataSource;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;
import com.latam.booklet.modelo.Libro;
import com.latam.booklet.modelo.LibroMapper;

@Repository
public class LibroDAOImpl implements LibroDAO {
  JdbcTemplate jdbcTemplate;
  DataSource dataSource;
  @Autowired
  public LibroDAOImpl(DataSource dataSource) {
        this.dataSource = dataSource;
        jdbcTemplate = new JdbcTemplate(dataSource);
  }
  @Override
```

```
	public List<Libro> findAll() {
		return jdbcTemplate.query("select * from LIBRO", new
LibroMapper());
	}

	@Override
	public Libro findById(Integer idLibro) {
		Object[] args = new Object[1];
		args[0] = idLibro;
		return jdbcTemplate.queryForObject("select * from LIBRO
where id_libro = ?", args, new LibroMapper());
	}

	@Override
	public int add(Libro libro) {
		System.out.println(libro);
		return jdbcTemplate.update("insert into libro values
(sq_libro.nextval, ?, ?, ?, ?, ?)", libro.getAnho(),
				libro.getTitulo(), libro.getAutor(),
libro.getImprenta(), libro.getDisponibilidad());
	}

	@Override
	public int update(Libro libro) {
		return jdbcTemplate.update(
				"update libro set anho = ?, titulo = ?,  autor =
?, imprenta = ?, disponible = ? where id_libro = ?",
				libro.getAnho(), libro.getTitulo(),
libro.getAutor(), libro.getImprenta(), libro.getDisponibilidad(),
				libro.getIdLibro());
	}

	@Override
	public int delete(Libro libro) {
		return jdbcTemplate.update("delete from libro where id_libro
= ?", libro.getIdLibro());
	}

	@Override
	public List<Libro> findByTituloOrAutor(String textoBuscado) {
		Object[] args = new Object[2];
		textoBuscado = "%"+textoBuscado+"%";
		args[0] = textoBuscado;
		args[1] = textoBuscado;
```

```
            return jdbcTemplate.query("select * from LIBRO where TITULO
like ? or AUTOR like ?", args, new LibroMapper());
      }
    }
```

Capa de servicio

```
    package com.latam.booklet.service;
    import com.latam.booklet.modelo.Libro;
    import com.latam.booklet.vo.LibroVO;
    public interface LibroService {
      public LibroVO findAll();
      public LibroVO findById(Integer idLibro);
      public LibroVO changeAvailability(Integer idLibro);
      public LibroVO add(Libro usuario);
      public LibroVO update(Libro usuario);
      public LibroVO delete(Libro usuario);
      LibroVO findByTituloOrAutor(String textoBuscado);
    }
```

```
    package com.latam.booklet.service;

    import java.util.ArrayList;
    import org.springframework.beans.factory.annotation.Autowired;
    import org.springframework.stereotype.Service;

    import com.latam.booklet.dao.LibroDAO;
    import com.latam.booklet.modelo.Libro;
    import com.latam.booklet.vo.LibroVO;

    @Service
    public class LibroServiceImpl implements LibroService {

      @Autowired
      LibroDAO dao;

      LibroVO respuesta;

      @Override
      public LibroVO findAll() {
```

```
            respuesta = new LibroVO(new ArrayList<Libro>(), "Ha ocurrido
un error", "101" );
            try {
                  respuesta.setLibros(new ArrayList<>(dao.findAll()));
                  respuesta.setMensaje(String.format("Se ha/n encontrado
%d registro/s", respuesta.getLibros().size()));
                  respuesta.setCodigo("0");
            } catch (Exception e) {
                  System.err.print(e);
            }
            return respuesta;
      }

      @Override
      public LibroVO findById(Integer idLibro) {
            respuesta = new LibroVO(new ArrayList<Libro>(), "Ha ocurrido
un error", "102" );
            try {

                  respuesta.setLibros(new ArrayList<>());
                  respuesta.getLibros().add(dao.findById(idLibro));
                  respuesta.setMensaje(String.format("Se ha/n encontrado
%d registro/s", respuesta.getLibros().size()));
                  respuesta.setCodigo("0");
            } catch (Exception e) {
                  System.err.print(e);
            }
            return respuesta;
      }



      @Override
      public LibroVO add(Libro usuario) {
            respuesta = new LibroVO(new ArrayList<Libro>(), "Ha ocurrido
un error", "103" );
            try {
                  int registrosActualizados = dao.add(usuario);
                  respuesta.setMensaje(registrosActualizados == 1 ? "Se
ha creado el libro correctamente" : "No se ha podido crear el libro");
                  respuesta.setCodigo(registrosActualizados == 1 ? "0" :
"104");
            } catch (Exception e) {
                  System.err.print(e);
```

```
            }
            return respuesta;
      }

      @Override
      public LibroVO update(Libro libro) {
            respuesta = new LibroVO(new ArrayList<Libro>(), "Ha ocurrido
un error", "104" );
            try {
                  int registrosActualizados = dao.update(libro);
                  respuesta.setMensaje(String.format("Se ha/n
actualizado correctamente %d libro/s", registrosActualizados));
                  respuesta.setCodigo("0");
            } catch (Exception e) {
                  System.err.print(e);
            }
            return respuesta;
      }

      @Override
      public LibroVO delete(Libro libro) {
            respuesta = new LibroVO(new ArrayList<Libro>(), "Ha ocurrido
un error", "105" );
            try {
                  int registrosActualizados = dao.delete(libro);
                  respuesta.setMensaje(String.format("Se ha eliminado
correctamente el libro", registrosActualizados));
                  respuesta.setCodigo("0");
            } catch (Exception e) {
                  System.err.print(e);
            }
            return respuesta;
      }


      @Override
      public LibroVO changeAvailability(Integer idLibro) {
            respuesta = new LibroVO(new ArrayList<Libro>(), "Ha ocurrido
un error", "106" );
            try {
                  respuesta.setLibros(new ArrayList<>());
                  Libro libro = dao.findById(idLibro);
                  libro.setDisponibilidad(libro.getDisponibilidad() == 1
? 0 : 1);
```

```
                dao.update(libro);
                respuesta.setMensaje(String.format("Se ha cambiado la
disponibilidad correctamente"));
                respuesta.setCodigo("0");
            } catch (Exception e) {
                System.err.print(e);
            }
            return respuesta;
      }

      @Override
      public LibroVO findByTituloOrAutor(String textoBuscado) {
            respuesta = new LibroVO(new ArrayList<Libro>(), "Ha ocurrido
un error", "107" );
            try {
                respuesta.setLibros(new
ArrayList<>(dao.findByTituloOrAutor(textoBuscado)));
                respuesta.setMensaje(String.format("Se ha/n encontrado
%d registro/s", respuesta.getLibros().size()));
                respuesta.setCodigo("0");
            } catch (Exception e) {
                System.err.print(e);
            }
            return respuesta;
      }

    }
```

El modelo

La clase Libro representa la clase principal del modelo.

```
    package com.latam.booklet.modelo;

    public class Libro {
      Integer idLibro;
      Integer anho;
      String titulo;
      String autor;
      String imprenta;
      Integer disponibilidad;

      public Libro(Integer idLibro, Integer anho, String titulo, String
autor, String imprenta, Integer disponibilidad) {
            super();
            this.idLibro = idLibro;
            this.anho = anho;
            this.titulo = titulo;
            this.autor = autor;
            this.imprenta = imprenta;
            this.disponibilidad = disponibilidad;
      }

      public Libro(LibroAgregarForm libro) {
            super();
            this.idLibro = libro.getIdLibro();
            this.anho = libro.getAnho();
            this.titulo = libro.getTitulo();
            this.autor = libro.getAutor();
            this.imprenta = libro.getImprenta();
            this.disponibilidad = null != libro.getDisponibilidad() ?
(libro.getDisponibilidad() ? 1 : 0) : 1;
      }

      public Libro() {
            super();
      }

      public Integer getIdLibro() {
            return idLibro;
```

```
    }

    public void setIdLibro(Integer idLibro) {
        this.idLibro = idLibro;
    }

    public Integer getAnho() {
        return anho;
    }

    public void setAnho(Integer anho) {
        this.anho = anho;
    }

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public String getAutor() {
        return autor;
    }

    public void setAutor(String autor) {
        this.autor = autor;
    }

    public String getImprenta() {
        return imprenta;
    }

    public void setImprenta(String imprenta) {
        this.imprenta = imprenta;
    }

    public Integer getDisponibilidad() {
        return disponibilidad;
    }

    public void setDisponibilidad(Integer disponibilidad) {
        this.disponibilidad = disponibilidad;
```

```
	}

	@Override
	public String toString() {
		return "Libro [idLibro=" + idLibro + ", anho=" + anho + ",
titulo=" + titulo + ", autor=" + autor
				+ ", imprenta=" + imprenta + ", disponibilidad="
+ disponibilidad + "]";
	}



    }
```

La clase *LibroAgregarForm* se utiliza como clase auxiliar para transformar el *booleano* disponibilidad a *Integer* y viceversa entre la clase Libro y *LibroAgregarForm*

```
    package com.latam.booklet.modelo;

    public class LibroAgregarForm {
      Integer idLibro;
      Integer anho;
      String titulo;
      String autor;
      String imprenta;
      Boolean disponibilidad;

      public LibroAgregarForm(Integer idLibro, Integer anho, String
titulo, String autor, String imprenta, Boolean disponibilidad) {
		super();
		this.idLibro = idLibro;
		this.anho = anho;
		this.titulo = titulo;
		this.autor = autor;
		this.imprenta = imprenta;
		this.disponibilidad = disponibilidad;
	}

      public LibroAgregarForm(Libro libro) {
		super();
		this.idLibro = libro.getIdLibro();
		this.anho = libro.getAnho();
```

```
            this.titulo = libro.getTitulo();
            this.autor = libro.getAutor();
            this.imprenta = libro.getImprenta();
            this.disponibilidad = null != libro.getDisponibilidad() ?
(libro.getDisponibilidad() == 1) : true;
      }

      public LibroAgregarForm() {
            super();
      }

      public Integer getIdLibro() {
            return idLibro;
      }

      public void setIdLibro(Integer idLibro) {
            this.idLibro = idLibro;
      }

      public Integer getAnho() {
            return anho;
      }

      public void setAnho(Integer anho) {
            this.anho = anho;
      }

      public String getTitulo() {
            return titulo;
      }

      public void setTitulo(String titulo) {
            this.titulo = titulo;
      }

      public String getAutor() {
            return autor;
      }

      public void setAutor(String autor) {
            this.autor = autor;
      }

      public String getImprenta() {
```

```
            return imprenta;
      }

      public void setImprenta(String imprenta) {
            this.imprenta = imprenta;
      }

      public Boolean getDisponibilidad() {
            return disponibilidad;
      }

      public void setDisponibilidad(Boolean disponibilidad) {
            this.disponibilidad = disponibilidad;
      }

      @Override
      public String toString() {
            return "Libro [idLibro=" + idLibro + ", anho=" + anho + ",
titulo=" + titulo + ", autor=" + autor
                        + ", imprenta=" + imprenta + ", disponibilidad="
+ disponibilidad + "]";
      }



    }
```

La clase *LibroMapper* se utiliza para mapear los resultados de la capa de persistencia a la clase Libro

```
    package com.latam.booklet.modelo;

    import java.sql.ResultSet;
    import java.sql.SQLException;

    import org.springframework.jdbc.core.RowMapper;

    public class LibroMapper implements RowMapper<Libro> {

      public Libro mapRow(ResultSet resultSet, int i) throws
SQLException {
            Libro libro = new Libro();
            libro.setIdLibro(resultSet.getInt("id_libro"));
            libro.setAnho(resultSet.getInt("anho"));
            libro.setTitulo(resultSet.getString("titulo"));
            libro.setAutor(resultSet.getString("autor"));
            libro.setImprenta(resultSet.getString("imprenta"));
            libro.setDisponibilidad(resultSet.getInt("disponible"));
            return libro;
      }
    }
```

La capa de Visual Objects

Se crea un paquete con *VO* para los diferentes casos posibles.

El primero es *LibroVO*, utilizado en la mayoría de las respuestas del servicio.

```
    package com.latam.booklet.vo;

    import java.util.List;

    import com.latam.booklet.modelo.Libro;

    public class LibroVO {
      List<Libro> libros;
      String mensaje;
      String codigo;

      public LibroVO(List<Libro> libros, String mensaje, String codigo)
{
            super();
            this.libros = libros;
            this.mensaje = mensaje;
            this.codigo = codigo;
      }

      public LibroVO() {
            super();
      }

      public List<Libro> getLibros() {
            return libros;
      }

      public void setLibros(List<Libro> libros) {
            this.libros = libros;
      }

      public String getMensaje() {
            return mensaje;
      }

      public void setMensaje(String mensaje) {
```

```
            this.mensaje = mensaje;
      }

      public String getCodigo() {
            return codigo;
      }

      public void setCodigo(String codigo) {
            this.codigo = codigo;
      }

    }
```

El otro es *LibroAgregarForm*, utilizado para las respuestas relacionadas a las vistas de editar y agregar.

Se hizo necesario este *VO* debido al valor *booleano* disponibilidad (para transformarlo a *Integer* y viceversa)

```
    package com.latam.booklet.vo;

    import java.util.List;

    import com.latam.booklet.modelo.LibroAgregarForm;

    public class LibroAgregarFormVO {
      List<LibroAgregarForm> libros;
      String mensaje;
      String codigo;

      public LibroAgregarFormVO(List<LibroAgregarForm> libros, String
mensaje, String codigo) {
            super();
            this.libros = libros;
            this.mensaje = mensaje;
            this.codigo = codigo;
      }

      public LibroAgregarFormVO() {
            super();
      }
```

```
    public List<LibroAgregarForm> getLibros() {
        return libros;
    }

    public void setLibros(List<LibroAgregarForm> libros) {
        this.libros = libros;
    }

    public String getMensaje() {
        return mensaje;
    }

    public void setMensaje(String mensaje) {
        this.mensaje = mensaje;
    }

    public String getCodigo() {
        return codigo;
    }

    public void setCodigo(String codigo) {
        this.codigo = codigo;
    }

}
```

La capa de controladores

Existe solo un controlador que maneja todas las solicitudes.

```
    package com.latam.booklet.controlador;

    import org.slf4j.Logger;
    import org.slf4j.LoggerFactory;
    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.ModelAttribute;
    import org.springframework.web.bind.annotation.PostMapping;
    import org.springframework.web.bind.annotation.RequestParam;
    import org.springframework.web.servlet.ModelAndView;
    import
org.springframework.web.servlet.mvc.support.RedirectAttributes;

    import com.latam.booklet.modelo.Libro;
    import com.latam.booklet.modelo.LibroAgregarForm;
    import com.latam.booklet.service.LibroService;
    import com.latam.booklet.vo.LibroVO;

    @Controller
    public class LibroController {

      private static final Logger log =
LoggerFactory.getLogger(LibroController.class);

      @Autowired
      private LibroService svc;

      @GetMapping("/editarForm")
      public ModelAndView editarForm(Model model, @RequestParam Integer
idLibro, RedirectAttributes ra) {
            LibroVO respuestaServicio = new LibroVO();
            respuestaServicio.setMensaje("No se pudo cargar la vista de
edición, intente nuevamente.");
            respuestaServicio = svc.findById(idLibro);
            model.addAttribute("mensaje",
respuestaServicio.getMensaje());
```

```
            model.addAttribute("VO", new
LibroAgregarForm(respuestaServicio.getLibros().get(0)));
            return new ModelAndView("editar");
      }

      @PostMapping("/editar")
      public ModelAndView editar(@ModelAttribute LibroAgregarForm libro,
RedirectAttributes ra) {
            LibroVO respuestaServicio = svc.update(new Libro(libro));
            ra.addFlashAttribute("mensaje",
respuestaServicio.getMensaje());
            if (respuestaServicio.getCodigo().equals("0")) {
                  return new ModelAndView("redirect:/home");
            } else {
                  return new ModelAndView("redirect:/editarForm");
            }
      }



      @GetMapping("/agregarForm")
      public String agregarForm(Model model) {
            return "agregar";
      }

      @PostMapping("/agregar")
      public ModelAndView agregarSubmit(@ModelAttribute LibroAgregarForm
libro, RedirectAttributes ra) {
            LibroVO respuestaServicio = svc.add(new Libro(libro));
            ra.addFlashAttribute("mensaje",
respuestaServicio.getMensaje());
            if (respuestaServicio.getCodigo().equals("0")) {
                  return new ModelAndView("redirect:/home");
            } else {
                  return new ModelAndView("redirect:/agregarForm");
            }
      }

      @GetMapping("/eliminar")
      public ModelAndView eliminar(Model model, @RequestParam String
idLibro, RedirectAttributes ra) {
            LibroVO respuestaServicio = new LibroVO();
            respuestaServicio.setMensaje("No se pudo eliminar el libro,
intente nuevamente.");
```

```
            try {
                  Libro eliminado = new Libro();
                  eliminado.setIdLibro(Integer.parseInt(idLibro));
                  respuestaServicio = svc.delete(eliminado);
                  ra.addFlashAttribute("mensaje",
respuestaServicio.getMensaje());
                  return new ModelAndView("redirect:/home");
            } catch (Exception e) {
                  log.error("Error en LibroController eliminar", e);
            }
            ra.addFlashAttribute("mensaje",
respuestaServicio.getMensaje());
            respuestaServicio = svc.findAll();
            ra.addAttribute("VO", respuestaServicio);
            return new ModelAndView("redirect:/home");
      }

      @GetMapping("/cambiarDisponibilidad")
      public ModelAndView cambiarDisponibilidad(Model model,
@RequestParam Integer idLibro,  RedirectAttributes ra) {
            LibroVO respuestaServicio = svc.changeAvailability(idLibro);
            ra.addFlashAttribute("mensaje",
respuestaServicio.getMensaje());
            return new ModelAndView("redirect:/home");
      }

      @PostMapping("/buscar")
      public String agregarSubmit(Model model, @RequestParam String
textoBuscado) {
            System.out.println(textoBuscado);
            LibroVO respuestaServicio =
svc.findByTituloOrAutor(textoBuscado);
            model.addAttribute("mensaje",
respuestaServicio.getMensaje());
            model.addAttribute("VO", respuestaServicio);
            return "home";
      }

      @GetMapping({ "/home" })
      public String home(Model model) {
            model.addAttribute("VO", svc.findAll());
            return "home";
      }
    }
```

La configuración del proyecto

AppConfig

```
    package com.latam.booklet;

    import javax.sql.DataSource;

    import org.springframework.beans.factory.annotation.Autowired;
    import org.springframework.context.annotation.Bean;
    import org.springframework.context.annotation.ComponentScan;
    import org.springframework.context.annotation.Configuration;
    import org.springframework.context.annotation.PropertySource;
    import org.springframework.core.env.Environment;
    import org.springframework.jdbc.datasource.DriverManagerDataSource;

    @Configuration
    @ComponentScan("com.latam.booklet")
    @PropertySource("classpath:database.properties")
    public class AppConfig {

      @Autowired
      Environment environment;

      @Bean
      DataSource dataSource() {
            DriverManagerDataSource driverManagerDataSource = new
DriverManagerDataSource();

driverManagerDataSource.setUrl(environment.getProperty("url"));

driverManagerDataSource.setUsername(environment.getProperty("dbuser"));

driverManagerDataSource.setPassword(environment.getProperty("dbpassword"
));

driverManagerDataSource.setDriverClassName(environment.getProperty("driv
er"));
            return driverManagerDataSource;
      }
    }
```

database.properties

```
driver=oracle.jdbc.driver.OracleDriver
url=jdbc:oracle:thin:@localhost:1521:xe
dbuser=root
dbpassword=root
```

application.properties

```
spring.mvc.view.prefix=/WEB-INF/jsp/
spring.mvc.view.suffix=.jsp
```

BookletApplication

Es la clase principal utilizada para ejecutar correctamente *Spring Boot*.

```
package com.latam.booklet;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.builder.SpringApplicationBuilder;
import
org.springframework.boot.web.servlet.support.SpringBootServletInitialize
r;
@SpringBootApplication
public class BookletApplication extends SpringBootServletInitializer
{
  @Override
    protected SpringApplicationBuilder
configure(SpringApplicationBuilder application) {
        return application.sources(BookletApplication.class);
    }

  public static void main(String[] args) {
        SpringApplication.run(BookletApplication.class, args);
  }
}
```

Las vistas

home.jsp

Es la vista principal que contiene casi todas las operaciones que la aplicación dispone a través de botones, además de mostrar la tabla con libros.

```
    <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
    <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
    <%@page import="com.latam.booklet.modelo.Libro"%>
    <%@page import="com.latam.booklet.vo.LibroVO"%>
    <!DOCTYPE html>
    <html>
    <head>
    <meta charset="charset=ISO-8859-1">
    <script src="/webjars/jquery/jquery.min.js"></script>
    <script src="/webjars/bootstrap/js/bootstrap.min.js"></script>
    <link rel="stylesheet"
      href="/webjars/bootstrap/4.3.0/css/bootstrap.min.css" />
    <title>Booklet</title>
    </head>
    <body>
      <!-- Inicio Header -->
      <nav class="navbar navbar-light bg-light">
            <a class="navbar-brand" href="#">Booklet</a>
      </nav>
      <!-- Fin Header -->

      <!-- Inicio Contenido -->
      <div class="p-3">

            <!-- Inicio Mensajes -->
            <c:if test="${mensaje != null ? true : false}">
                  <div class="alert alert-secondary alert-dismissible
fade show"
                        role="alert">${mensaje}
                        <button type="button" class="close"
data-dismiss="alert"
                              aria-label="Close">
                              <span aria-hidden="true">&times;</span>
```

```
                        </button>
                  </div>
            </c:if>
            <!-- Fin Mensajes -->

            <!-- Botón agregar usuario -->
            <a href="agregarForm" class="btn m-2 btn-success">Agregar
libro</a>


            <form method="POST" action="buscar">
                  <table>
                        <tr>
                              <td class="p-2"><label
for="imprenta">Buscador:</label></td>
                              <td><input class="form-control" type="text"
                                    placeholder="Buscador"
name="textoBuscado" /></td>
                                    <td colspan="2"><input type="submit"
                                    class="btn m-2 btn-primary"
value="Buscar" /></td>
                        </tr>
                  </table>
            </form>

            <!-- Inicio Tabla -->
            <table border="1" class="table table-hover">
                  <thead class="thead-dark">
                        <tr>
                              <th scope="col">Título</th>
                              <th scope="col">Autor</th>
                              <th scope="col">Imprenta</th>
                              <th scope="col">Año</th>
                              <th scope="col">Disponibilidad</th>
                              <th scope="col">Acciones</th>
                        </tr>
                  </thead>
                  <tbody>
                        <c:forEach items="${VO.libros}" var="u">
                              <tr>
                                    <td>${u.getTitulo()}</td>
                                    <td>${u.getAutor()}</td>
                                    <td>${u.getImprenta()}</td>
                                    <td>${u.getAnho()}</td>
```

```
                                    <td>${u.getDisponibilidad() == 1 ?
'Si' : 'No'}</td>

                                    <td><a
href="cambiarDisponibilidad?idLibro=${u.getIdLibro()}"
                                          class="btn btn-warning
                                           btn-sm">Cambiar
Disponibilidad</a>
                                          <a
href="editarForm?idLibro=${u.getIdLibro()}"
                                          class="btn btn-primary
btn-sm">Editar</a> <a

href="eliminar?idLibro=${u.getIdLibro()}"
                                          class="btn btn-danger
btn-sm">Eliminar</a></td>
                                </tr>
                            </c:forEach>

                    </tbody>
                </table>
                <!-- Fin tabla -->
            </div>

            <!-- Fin Contenido -->
        </body>
        </html>
```

agregar.jsp

Es la vista utilizada para agregar libros a la base de datos.

```
    <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
    <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
    <%@page import="com.latam.booklet.modelo.Libro"%>
    <%@page import="com.latam.booklet.vo.LibroVO"%>
    <!DOCTYPE html>
    <html>
    <head>
    <meta charset="charset=ISO-8859-1">
    <script src="/webjars/jquery/jquery.min.js"></script>
    <script src="/webjars/bootstrap/js/bootstrap.min.js"></script>
    <link rel="stylesheet"
      href="/webjars/bootstrap/4.3.0/css/bootstrap.min.css" />
    <title>JSP CRUD</title>
    </head>
    <body>

      <!-- Inicio Contenido -->
      <div class="p-3">

            <h1>Agregar libro</h1>
            <c:if test="${mensaje != null ? true : false}">
                  <div class="alert alert-secondary  alert-dismissible
fade show"
                        role="alert">${mensaje}
                        <button type="button" class="close"
data-dismiss="alert"
                              aria-label="Close">
                              <span aria-hidden="true">&times;</span>
                        </button>
                  </div>
            </c:if>
            <form action="agregar" method="post">
                  <table>
                        <tr>
                              <td class="p-2"><label
for="anho">Año:</label></td>
                              <td><input class="form-control"
```

```
type="number" name="anho"
                                    placeholder="Año"  /></td>
                        </tr>
                        <tr>
                              <td class="p-2"><label
for="titulo">Titulo:</label></td>
                              <td><input class="form-control" type="text"
                                    placeholder="Titulo" name="titulo"
/></td>
                        </tr>
                        <tr>
                              <td class="p-2"><label
for="autor">Autor:</label></td>
                              <td><input class="form-control" type="text"
                                    placeholder="Autor"
name="autor"/></td>
                        </tr>

                        <tr>
                              <td class="p-2"><label
for="imprenta">Imprenta:</label></td>
                              <td><input class="form-control" type="text"
                                    placeholder="Imprenta"
name="imprenta" /></td>
                        </tr>

                        <tr>
                              <td class="p-2"></td>
                              <td>

                                    <input class="form-check-input"
type="checkbox" value="true" id="disponibilidad" name="disponibilidad">
                                      <label class="form-check-label"
for="disponibilidad">

                                        Disponible
                                      </label>

                                    </td>
                        </tr>

                        <tr>
                              <td colspan="2"><input type="submit"
                                    class="btn m-2 btn-success"
value="Agregar" /></td>
```

```
                        </tr>
                    </table>
                </form>
            </div>
        </body>
        </html>
```

editar.jsp

Es la vista utilizada para editar los registros existentes en la base de datos.

```
    <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
    <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
    <%@page import="com.latam.booklet.modelo.Libro"%>
    <%@page import="com.latam.booklet.vo.LibroVO"%>
    <!DOCTYPE html>
    <html>
    <head>
    <meta charset="charset=ISO-8859-1">
    <script src="/webjars/jquery/jquery.min.js"></script>
    <script src="/webjars/bootstrap/js/bootstrap.min.js"></script>
    <link rel="stylesheet"
      href="/webjars/bootstrap/4.3.0/css/bootstrap.min.css" />
    <title>Booklet</title>
    </head>
    <body>
      <div class="p-3">

            <h1>Editar libro</h1>
            <c:if test="${mensaje != null ? true : false}">
                  <div class="alert alert-secondary  alert-dismissible
fade show"
                        role="alert">${mensaje}
                        <button type="button" class="close"
data-dismiss="alert"
                              aria-label="Close">
                              <span aria-hidden="true">&times;</span>
                        </button>
                  </div>
            </c:if>
```

```
            <form action="editar" method="post">
                  <input type="hidden" name="idLibro"
value="${VO.getIdLibro()}" />
                  <table>

                        <tr>
                              <td class="p-2"><label
for="anho">Año:</label></td>
                              <td><input class="form-control"
type="number" name="anho"
                                    placeholder="Año"
value="${VO.getAnho()}" /></td>
                        </tr>
                        <tr>
                              <td class="p-2"><label
for="titulo">Titulo:</label></td>
                              <td><input class="form-control" type="text"
                                    placeholder="Titulo" name="titulo"
value="${VO.getTitulo()}" /></td>
                        </tr>
                        <tr>
                              <td class="p-2"><label
for="autor">Autor:</label></td>
                              <td><input class="form-control" type="text"
                                    placeholder="Autor" name="autor"
value="${VO.getAutor()}" /></td>
                        </tr>

                        <tr>
                              <td class="p-2"><label
for="imprenta">Imprenta:</label></td>
                              <td><input class="form-control" type="text"
                                    placeholder="Imprenta"
name="imprenta" value="${VO.getImprenta()}" /></td>
                        </tr>

                        <tr>
                              <td class="p-2"></td>
                              <td>

                                    <input class="form-check-input"
type="checkbox" checked="${VO.getDisponibilidad()}" id="disponibilidad"
name="disponibilidad">
                                      <label class="form-check-label"
```

```
for="disponibilidad">
                                        Disponible
                                      </label>

                                    </td>
                        </tr>
                        <tr>
                              <td colspan="2"><input type="submit"
                                    class="btn m-2 btn-success"
value="Guardar" /></td>
                        </tr>
                  </table>
            </form>
      </div>

    </body>
    </html>
```

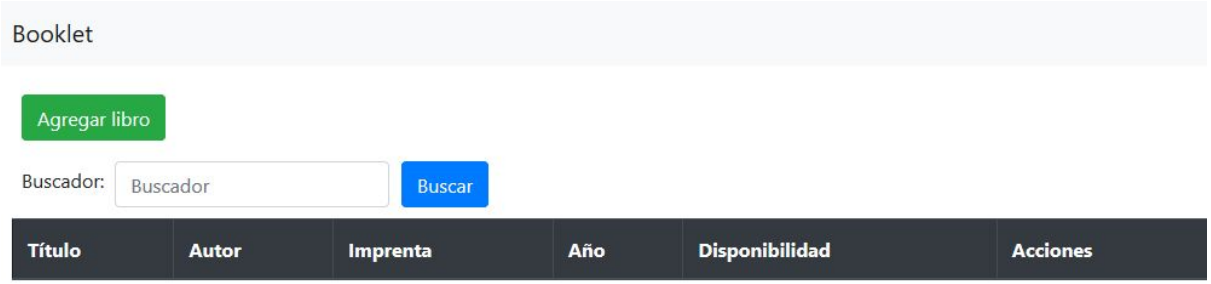
error.jsp

Es la vista a la que se redireccionará si ocurre alguna excepción no controlada en el aplicativo.

```
    <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
    <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
    <!DOCTYPE html>
    <html>
    <head>
      <meta charset="charset=ISO-8859-1">
      <title>Booklet</title>
    </head>
    <body>
      <h1>Ha ocurrido un error</h1>
      <h3>
            <a href="home">volver al inicio</a>
      </h3>
    </body>
    </html>
```

Resultados

Home



Agregando libro

Se despliega el formulario.

Se rellena el formulario y se presiona Agregar.

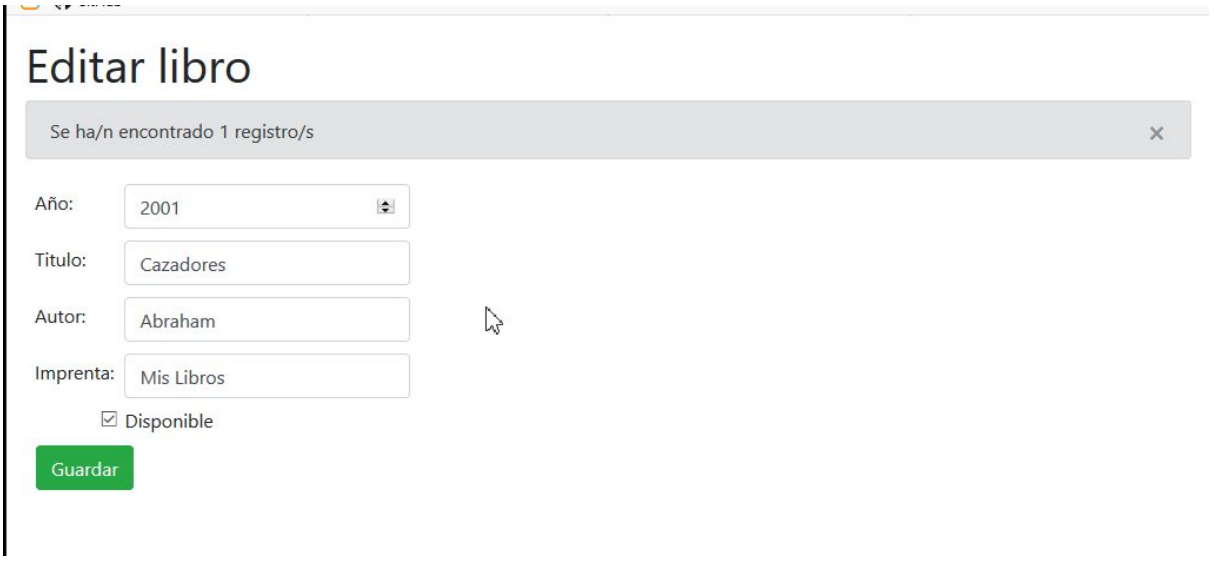
Cambiando la disponibilidad

Se presiona el botón y se despliega el mensaje informativo del resultado.

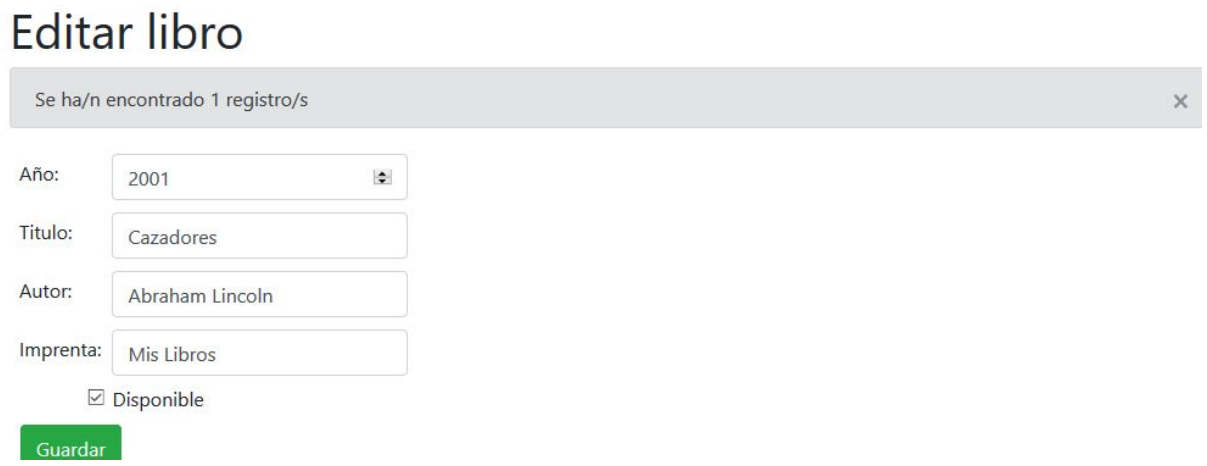
Editando un libro

Se presiona Editar.

Se despliega el formulario



Se modifica el autor del libro de Abraham a Abraham Lincoln.



Se presiona guardar y se muestra el mensaje de resultado.

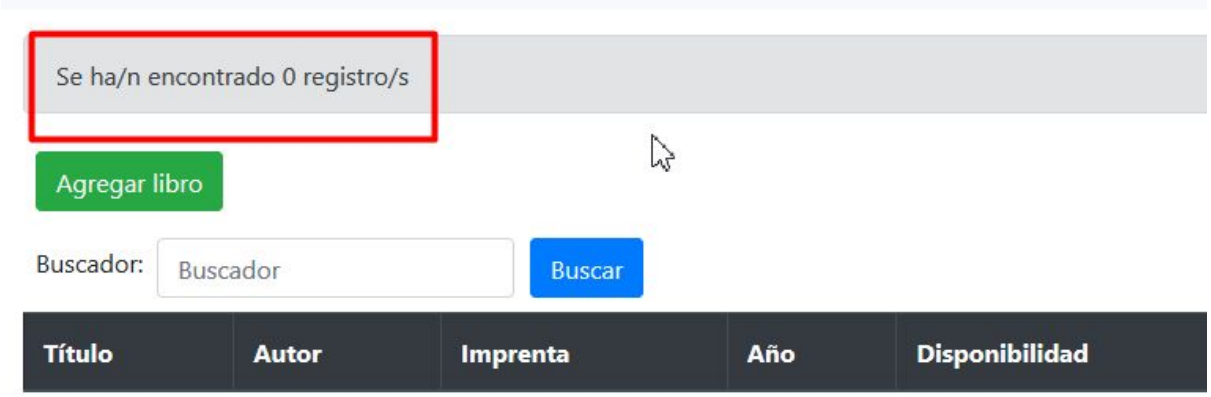
Buscador

Se ingresa el texto Lincoln y se presiona buscar.

Se refresca la lista con los resultados y se despliega mensaje informativo.

Se escribe un texto que no coincida con ningún título ni autor y se presiona buscar.

Se refresca la lista y se despliega mensaje informativo.



Eliminando libros

Se presiona eliminar.

Se refresca la tabla y se muestra mensaje informativo.