Leonel Briones Palacios

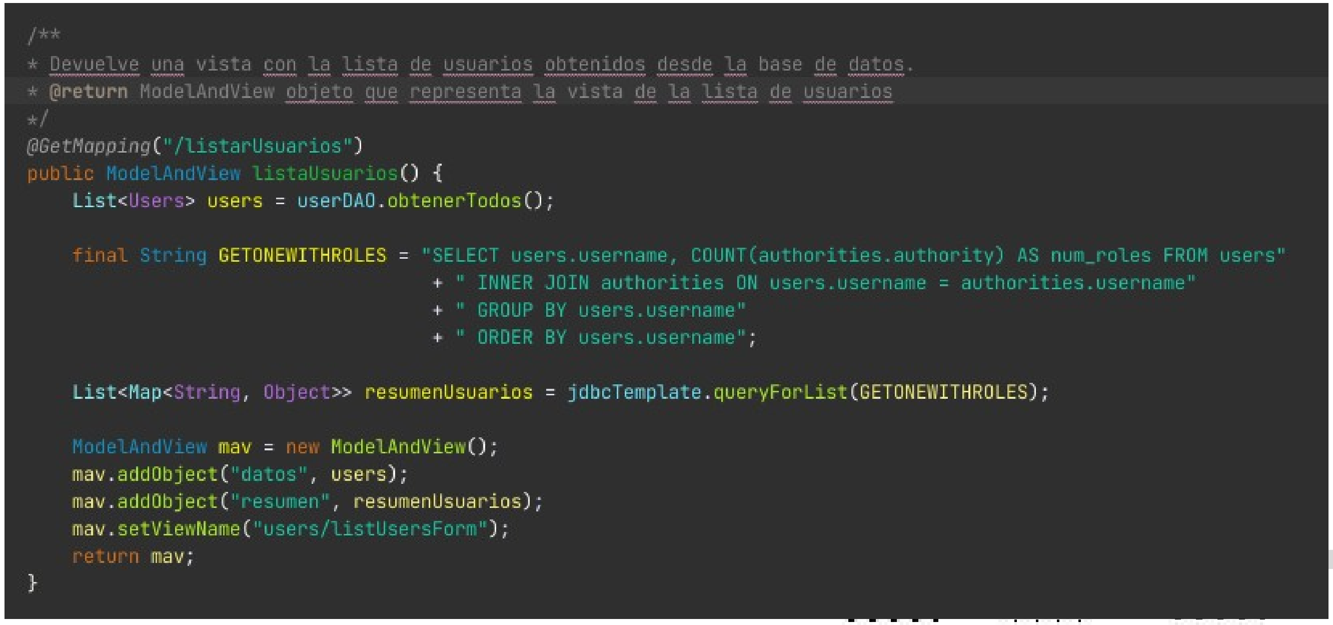
https://github.com/jarodsmdev/Veterinaria

Spring Framework MVC
Spring Security
Java 1.8
JUnit 5
11 Mayo 2023

Consultas a la Base de datos

1. SELECT

Selecciona las columnas requeridas para presentar la información solicitada.

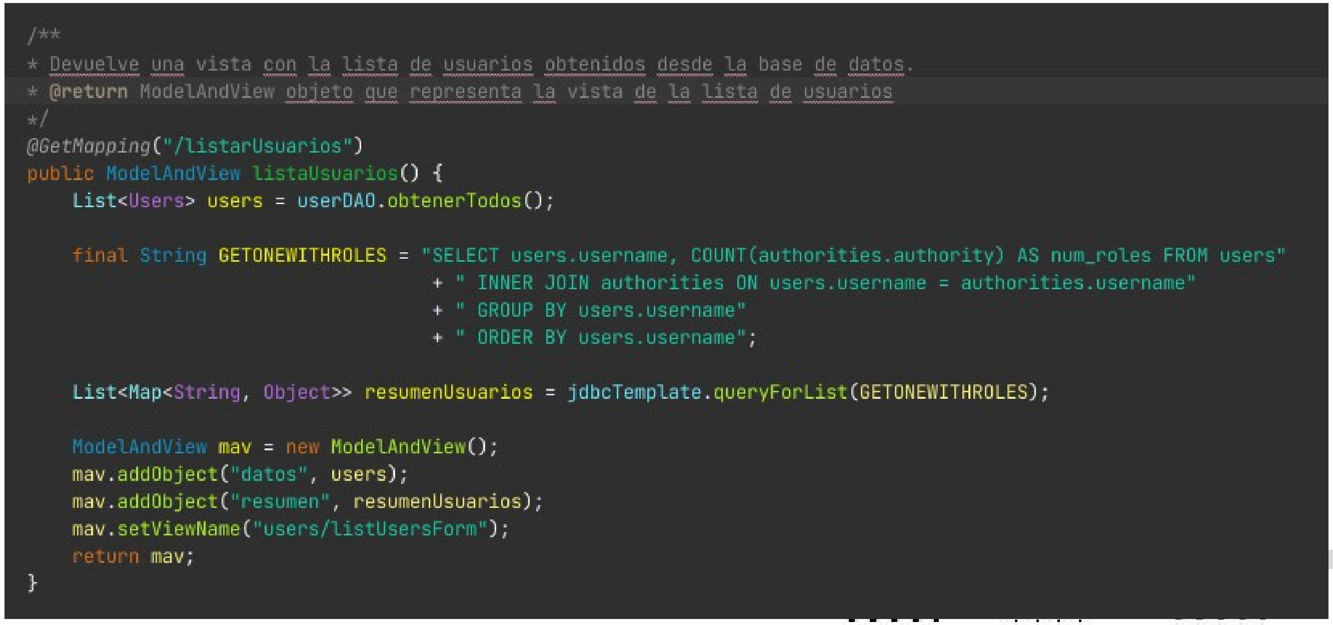


```java
/**
* Devuelve una vista con la lista de usuarios obtenidos desde la base de datos.
* @return ModelAndView objeto que representa la vista de la lista de usuarios
*/
@GetMapping("/listarUsuarios")
public ModelAndView listaUsuarios() {
    List<Users> users = userDAO.obtenerTodos();

    final String GETONEWITHROLES = "SELECT users.username, COUNT(authorities.authority) AS num_roles FROM users"
                                + " INNER JOIN authorities ON users.username = authorities.username"
                                + " GROUP BY users.username"
                                + " ORDER BY users.username";

    List<Map<String, Object>> resumenUsuarios = jdbcTemplate.queryForList(GETONEWITHROLES);

    ModelAndView mav = new ModelAndView();
    mav.addObject("datos", users);
    mav.addObject("resumen", resumenUsuarios);
    mav.setViewName("users/listUsersForm");
    return mav;
}
```

Consultas a la Base de datos

2. JOIN

Utiliza JOIN para relacionar la información de distintas tablas

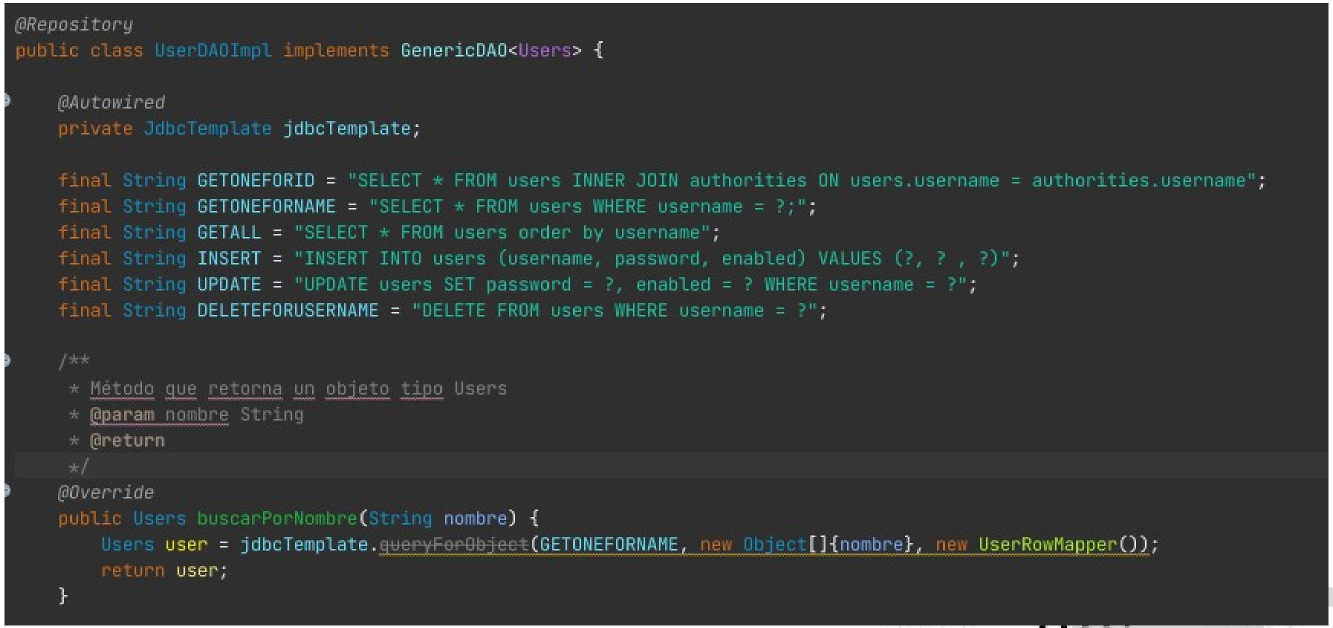


```
/**
* Devuelve una vista con la lista de usuarios obtenidos desde la base de datos.
* @return ModelAndView objeto que representa la vista de la lista de usuarios
*/
@GetMapping("/listarUsuarios")
public ModelAndView listaUsuarios() {
    List<Users> users = userDAO.obtenerTodos();

    final String GETONEWITHROLES = "SELECT users.username, COUNT(authorities.authority) AS num_roles FROM users"
                                   + " INNER JOIN authorities ON users.username = authorities.username"
                                   + " GROUP BY users.username"
                                   + " ORDER BY users.username";

    List<Map<String, Object>> resumenUsuarios = jdbcTemplate.queryForList(GETONEWITHROLES);

    ModelAndView mav = new ModelAndView();
    mav.addObject("datos", users);
    mav.addObject("resumen", resumenUsuarios);
    mav.setViewName("users/listUsersForm");
    return mav;
}
```

Consultas a la Base de datos

3. WHERE

Utiliza WHERE para filtrar información requerida

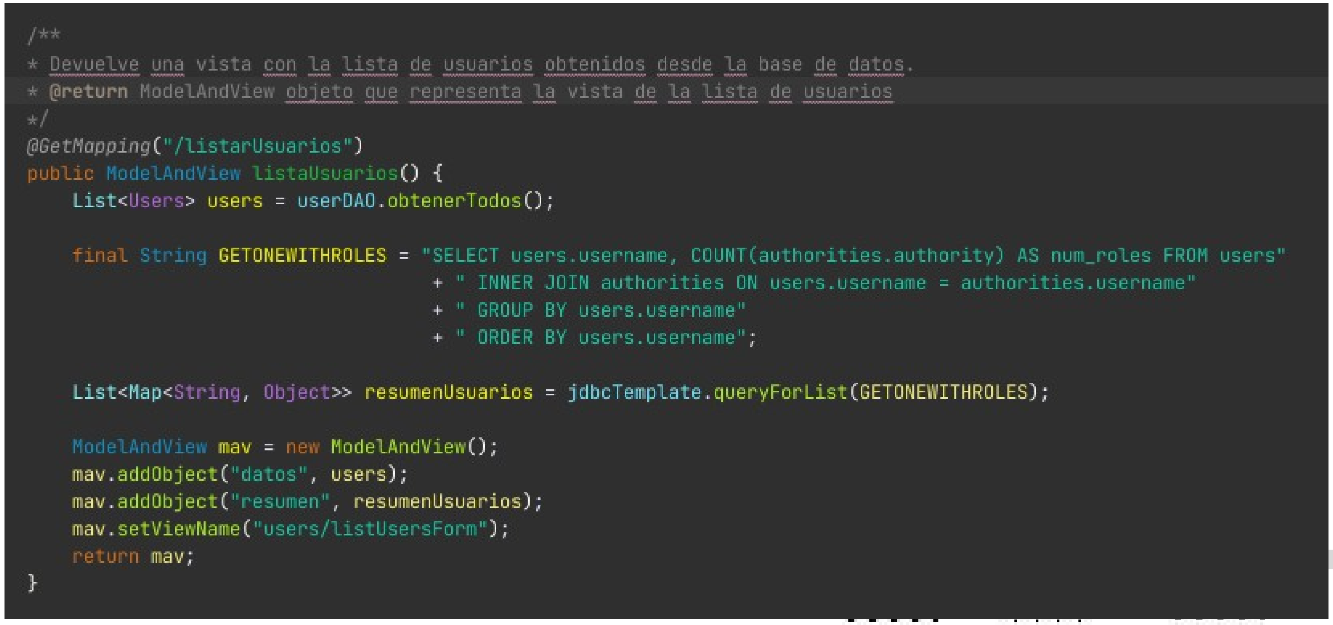


```java
@Repository
public class UserDAOImpl implements GenericDAO<Users> {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    final String GETONEFORID = "SELECT * FROM users INNER JOIN authorities ON users.username = authorities.username";
    final String GETONEFORNAME = "SELECT * FROM users WHERE username = ?;";
    final String GETALL = "SELECT * FROM users order by username";
    final String INSERT = "INSERT INTO users (username, password, enabled) VALUES (?, ? , ?)";
    final String UPDATE = "UPDATE users SET password = ?, enabled = ? WHERE username = ?";
    final String DELETEFORUSERNAME = "DELETE FROM users WHERE username = ?";

    /**
     * Método que retorna un objeto tipo Users
     * @param nombre String
     * @return
     */
    @Override
    public Users buscarPorNombre(String nombre) {
        Users user = jdbcTemplate.~~queryForObject~~(GETONEFORNAME, new Object[]{nombre}, new UserRowMapper());
        return user;
    }
```

Consultas a la Base de datos

4. ORDER BY

Utiliza cláusulas de ordenamiento para presentar la información

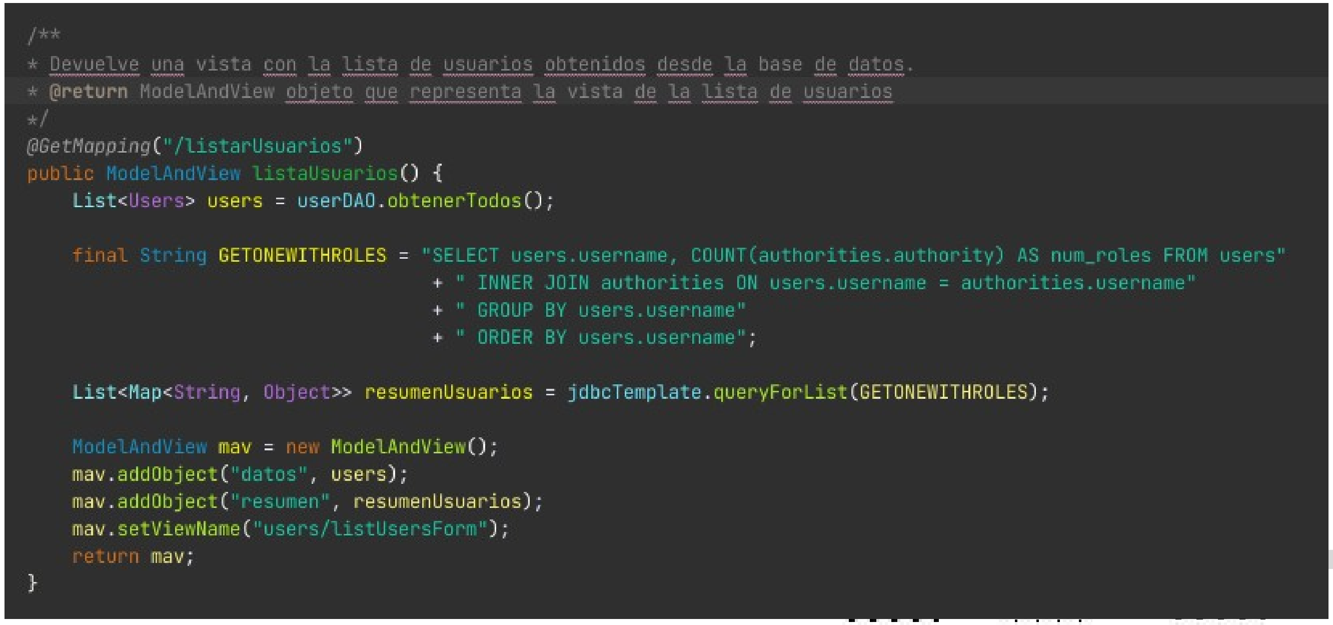


```java
/**
* Devuelve una vista con la lista de usuarios obtenidos desde la base de datos.
* @return ModelAndView objeto que representa la vista de la lista de usuarios
*/
@GetMapping("/listarUsuarios")
public ModelAndView listaUsuarios() {
    List<Users> users = userDAO.obtenerTodos();

    final String GETONEWITHROLES = "SELECT users.username, COUNT(authorities.authority) AS num_roles FROM users"
                                   + " INNER JOIN authorities ON users.username = authorities.username"
                                   + " GROUP BY users.username"
                                   + " ORDER BY users.username";

    List<Map<String, Object>> resumenUsuarios = jdbcTemplate.queryForList(GETONEWITHROLES);

    ModelAndView mav = new ModelAndView();
    mav.addObject("datos", users);
    mav.addObject("resumen", resumenUsuarios);
    mav.setViewName("users/listUsersForm");
    return mav;
}
```

Consultas a la Base de datos

5. GROUP BY

Utiliza cláusulas de agrupación de información para obtener datos agregados

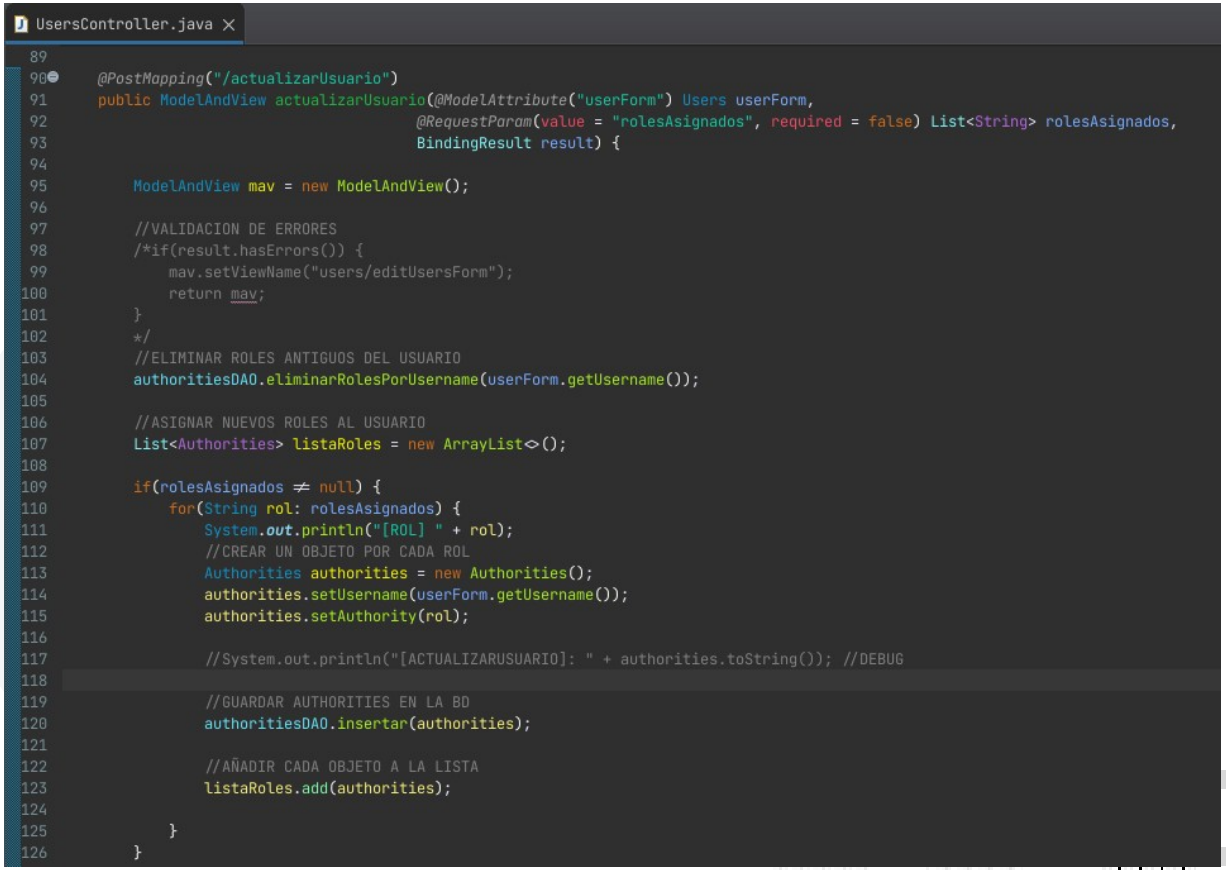


```java
/**
* Devuelve una vista con la lista de usuarios obtenidos desde la base de datos.
* @return ModelAndView objeto que representa la vista de la lista de usuarios
*/
@GetMapping("/listarUsuarios")
public ModelAndView listaUsuarios() {
    List<Users> users = userDAO.obtenerTodos();

    final String GETONEWITHROLES = "SELECT users.username, COUNT(authorities.authority) AS num_roles FROM users"
                                    + " INNER JOIN authorities ON users.username = authorities.username"
                                    + " GROUP BY users.username"
                                    + " ORDER BY users.username";

    List<Map<String, Object>> resumenUsuarios = jdbcTemplate.queryForList(GETONEWITHROLES);

    ModelAndView mav = new ModelAndView();
    mav.addObject("datos", users);
    mav.addObject("resumen", resumenUsuarios);
    mav.setViewName("users/listUsersForm");
    return mav;
}
```

Algoritmo de cálculo y unidades de prueba

6. USO DEL LENGUAJE

Utilización general
del lenguaje, sintaxis,
selección
de tipos de datos,
sentencias lógicas,
expresiones,
operaciones,
comparaciones

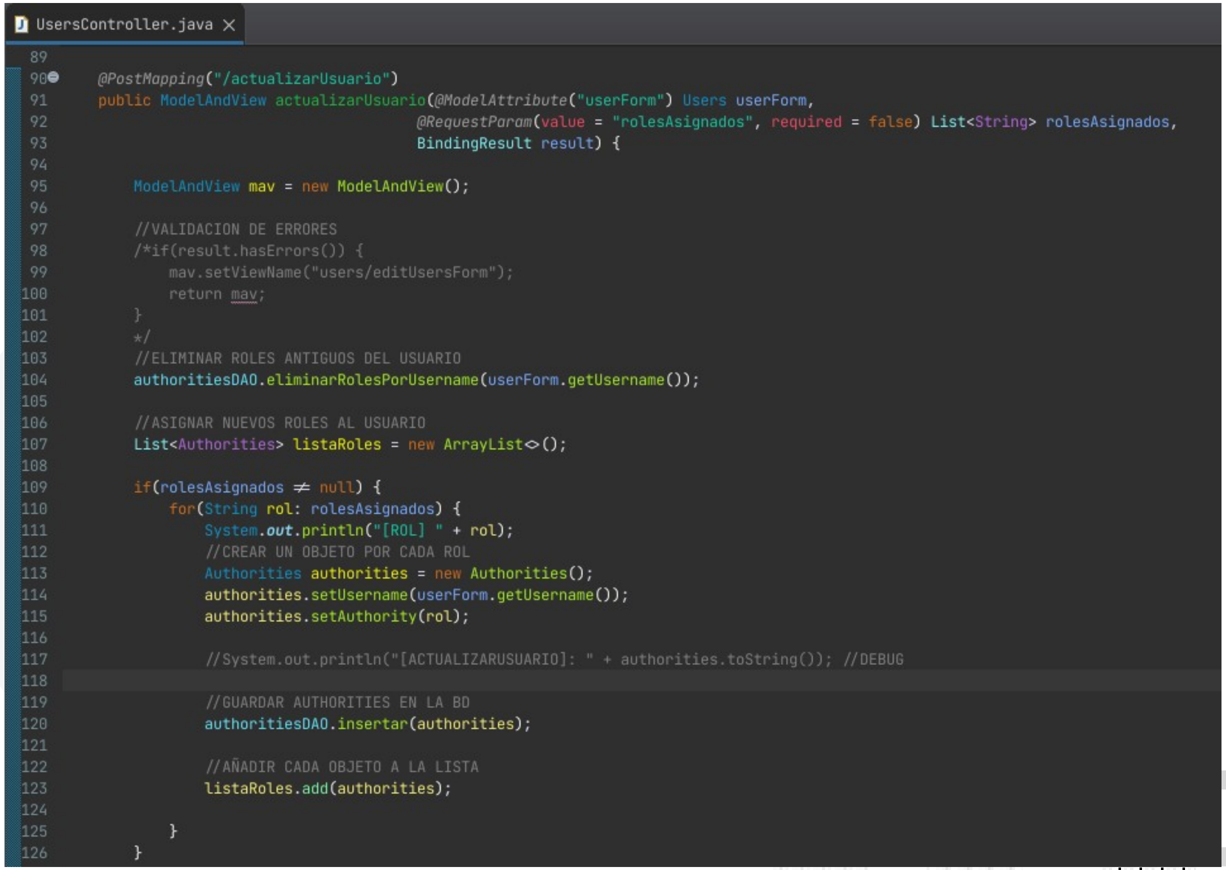

UsersController.java

```java

    @PostMapping("/actualizarUsuario")
    public ModelAndView actualizarUsuario(@ModelAttribute("userForm") Users userForm,
                                          @RequestParam(value = "rolesAsignados", required = false) List<String> rolesAsignados,
                                          BindingResult result) {

        ModelAndView mav = new ModelAndView();

        //VALIDACION DE ERRORES
        /*if(result.hasErrors()) {
            mav.setViewName("users/editUsersForm");
            return mav;
        }
        */
        //ELIMINAR ROLES ANTIGUOS DEL USUARIO
        authoritiesDAO.eliminarRolesPorUsername(userForm.getUsername());

        //ASIGNAR NUEVOS ROLES AL USUARIO
        List<Authorities> listaRoles = new ArrayList<>();

        if(rolesAsignados ≠ null) {
            for(String rol: rolesAsignados) {
                System.out.println("[ROL] " + rol);
                //CREAR UN OBJETO POR CADA ROL
                Authorities authorities = new Authorities();
                authorities.setUsername(userForm.getUsername());
                authorities.setAuthority(rol);

                //System.out.println("[ACTUALIZARUSUARIO]: " + authorities.toString()); //DEBUG

                //GUARDAR AUTHORITIES EN LA BD
                authoritiesDAO.insertar(authorities);

                //AÑADIR CADA OBJETO A LA LISTA
                listaRoles.add(authorities);

            }
        }
```

Algoritmo de cálculo y unidades de prueba

7. SENTENCIAS REPETITIVAS

Utilización de sentencias repetitivas



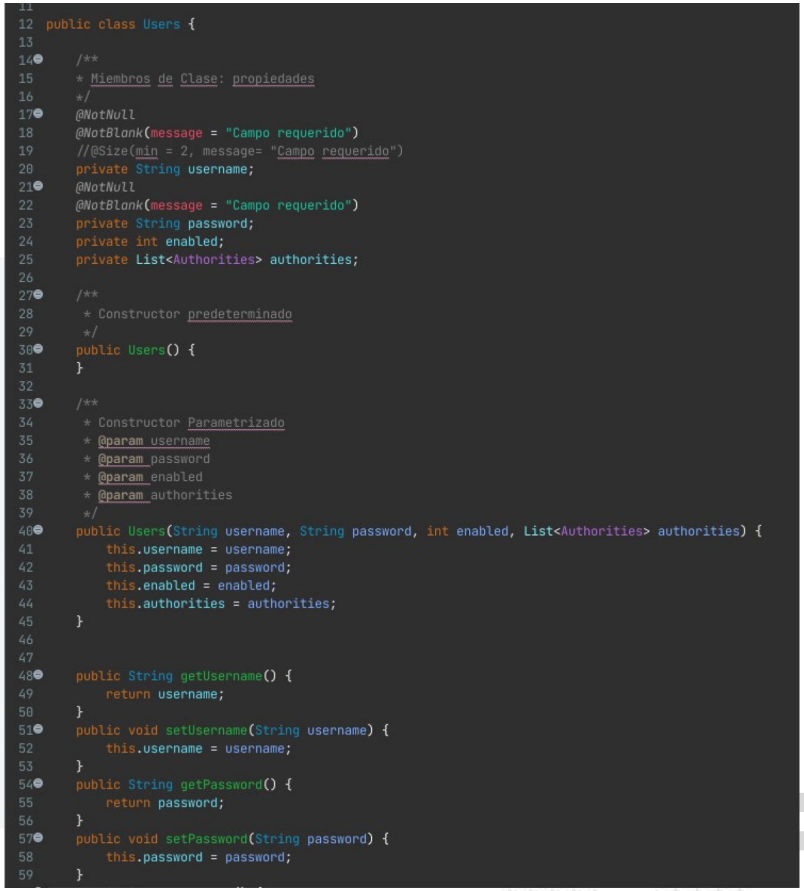
UsersController.java

```java

    @PostMapping("/actualizarUsuario")
    public ModelAndView actualizarUsuario(@ModelAttribute("userForm") Users userForm,
                                          @RequestParam(value = "rolesAsignados", required = false) List<String> rolesAsignados,
                                          BindingResult result) {

        ModelAndView mav = new ModelAndView();

        //VALIDACION DE ERRORES
        /*if(result.hasErrors()) {
            mav.setViewName("users/editUsersForm");
            return mav;
        }
        */
        //ELIMINAR ROLES ANTIGUOS DEL USUARIO
        authoritiesDAO.eliminarRolesPorUsername(userForm.getUsername());

        //ASIGNAR NUEVOS ROLES AL USUARIO
        List<Authorities> listaRoles = new ArrayList<>();

        if(rolesAsignados ≠ null) {
            for(String rol: rolesAsignados) {
                System.out.println("[ROL] " + rol);
                //CREAR UN OBJETO POR CADA ROL
                Authorities authorities = new Authorities();
                authorities.setUsername(userForm.getUsername());
                authorities.setAuthority(rol);

                //System.out.println("[ACTUALIZARUSUARIO]: " + authorities.toString()); //DEBUG

                //GUARDAR AUTHORITIES EN LA BD
                authoritiesDAO.insertar(authorities);

                //AÑADIR CADA OBJETO A LA LISTA
                listaRoles.add(authorities);

            }
        }
```

Algoritmo de cálculo y unidades de prueba

8. Uso de clases, encapsulamiento, responsabilidad única.

Utilización de clases, encapsulamiento y responsabilidad única

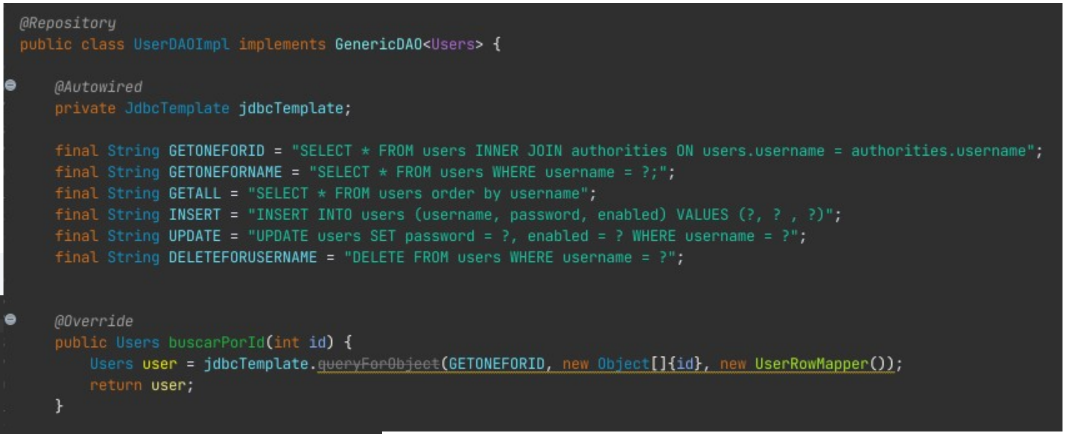


```java
public class Users {

    /**
    * Miembros de Clase: propiedades
    */
    @NotNull
    @NotBlank(message = "Campo requerido")
    //@Size(min = 2, message= "Campo requerido")
    private String username;
    @NotNull
    @NotBlank(message = "Campo requerido")
    private String password;
    private int enabled;
    private List<Authorities> authorities;

    /**
     * Constructor predeterminado
     */
    public Users() {
    }

    /**
     * Constructor Parametrizado
     * @param username
     * @param password
     * @param enabled
     * @param authorities
     */
    public Users(String username, String password, int enabled, List<Authorities> authorities) {
        this.username = username;
        this.password = password;
        this.enabled = enabled;
        this.authorities = authorities;
    }


    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
```

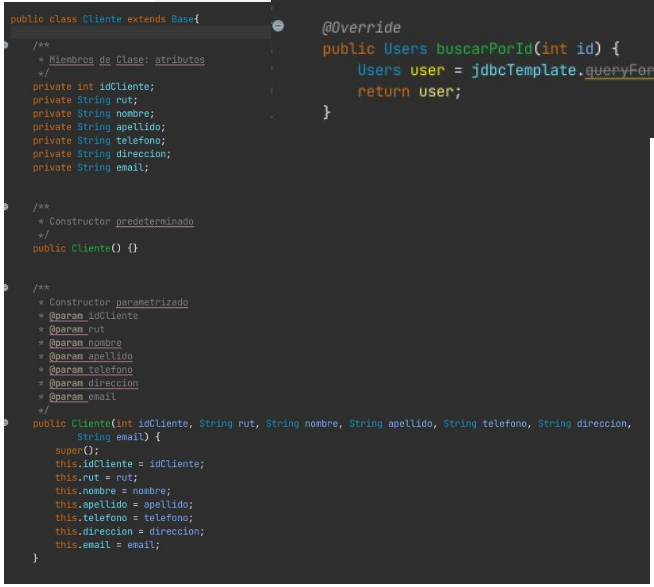
Algoritmo de cálculo y unidades de prueba

9. Uso de clases, encapsulamiento, responsabilidad única.

Se utilizan correctamente interfaces o relaciones de herencia para hacer polimorfismo donde fuese necesario



```java
@Repository
public class UserDAOImpl implements GenericDAO<Users> {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    final String GETONEFORID = "SELECT * FROM users INNER JOIN authorities ON users.username = authorities.username";
    final String GETONEFORNAME = "SELECT * FROM users WHERE username = ?;";
    final String GETALL = "SELECT * FROM users order by username";
    final String INSERT = "INSERT INTO users (username, password, enabled) VALUES (?, ? , ?)";
    final String UPDATE = "UPDATE users SET password = ?, enabled = ? WHERE username = ?";
    final String DELETEFORUSERNAME = "DELETE FROM users WHERE username = ?";


    @Override
    public Users buscarPorId(int id) {
        Users user = jdbcTemplate.~~queryForObject~~(GETONEFORID, new Object[]{id}, new UserRowMapper());
        return user;
    }
```

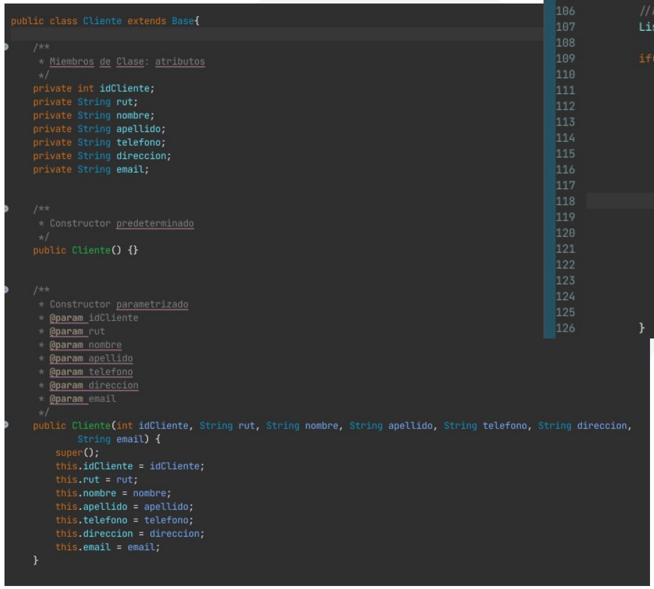


```java
public class Cliente extends Base{

    /**
     * Miembros de Clase: atributos
     */
    private int idCliente;
    private String rut;
    private String nombre;
    private String apellido;
    private String telefono;
    private String direccion;
    private String email;


    /**
     * Constructor predeterminado
     */
    public Cliente() {}


    /**
     * Constructor parametrizado
     * @param idCliente
     * @param rut
     * @param nombre
     * @param apellido
     * @param telefono
     * @param direccion
     * @param email
     */
    public Cliente(int idCliente, String rut, String nombre, String apellido, String telefono, String direccion,
            String email) {
        super();
        this.idCliente = idCliente;
        this.rut = rut;
        this.nombre = nombre;
        this.apellido = apellido;
        this.telefono = telefono;
        this.direccion = direccion;
        this.email = email;
    }
```

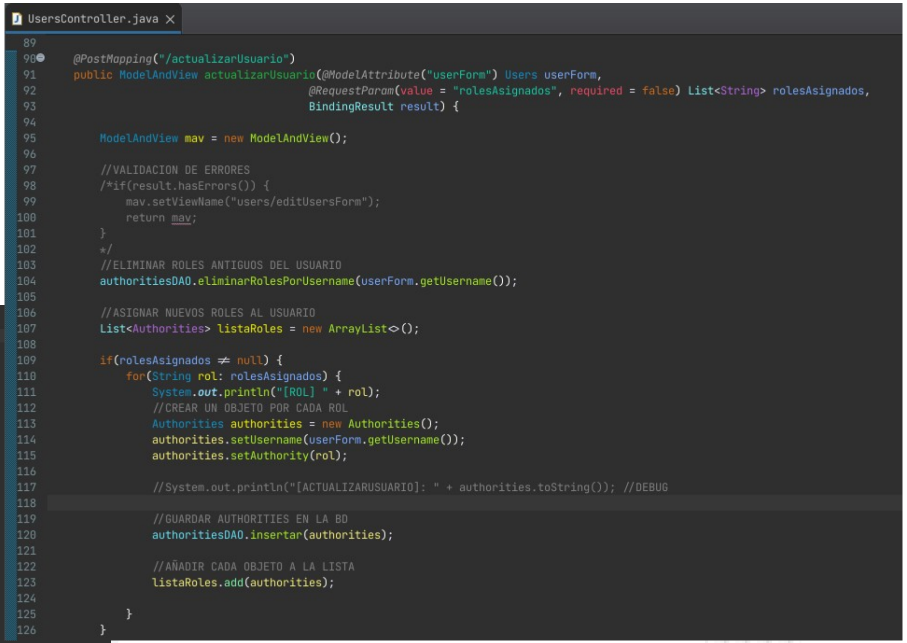
Algoritmo de cálculo y unidades de prueba

10. Convenciones

Convenciones y estilos de programación.



```java
public class Cliente extends Base{

    /**
     * Miembros de Clase: atributos
     */
    private int idCliente;
    private String rut;
    private String nombre;
    private String apellido;
    private String telefono;
    private String direccion;
    private String email;


    /**
     * Constructor predeterminado
     */
    public Cliente() {}


    /**
     * Constructor parametrizado
     * @param idCliente
     * @param rut
     * @param nombre
     * @param apellido
     * @param telefono
     * @param direccion
     * @param email
     */
    public Cliente(int idCliente, String rut, String nombre, String apellido, String telefono, String direccion,
            String email) {
        super();
        this.idCliente = idCliente;
        this.rut = rut;
        this.nombre = nombre;
        this.apellido = apellido;
        this.telefono = telefono;
        this.direccion = direccion;
        this.email = email;
    }
```



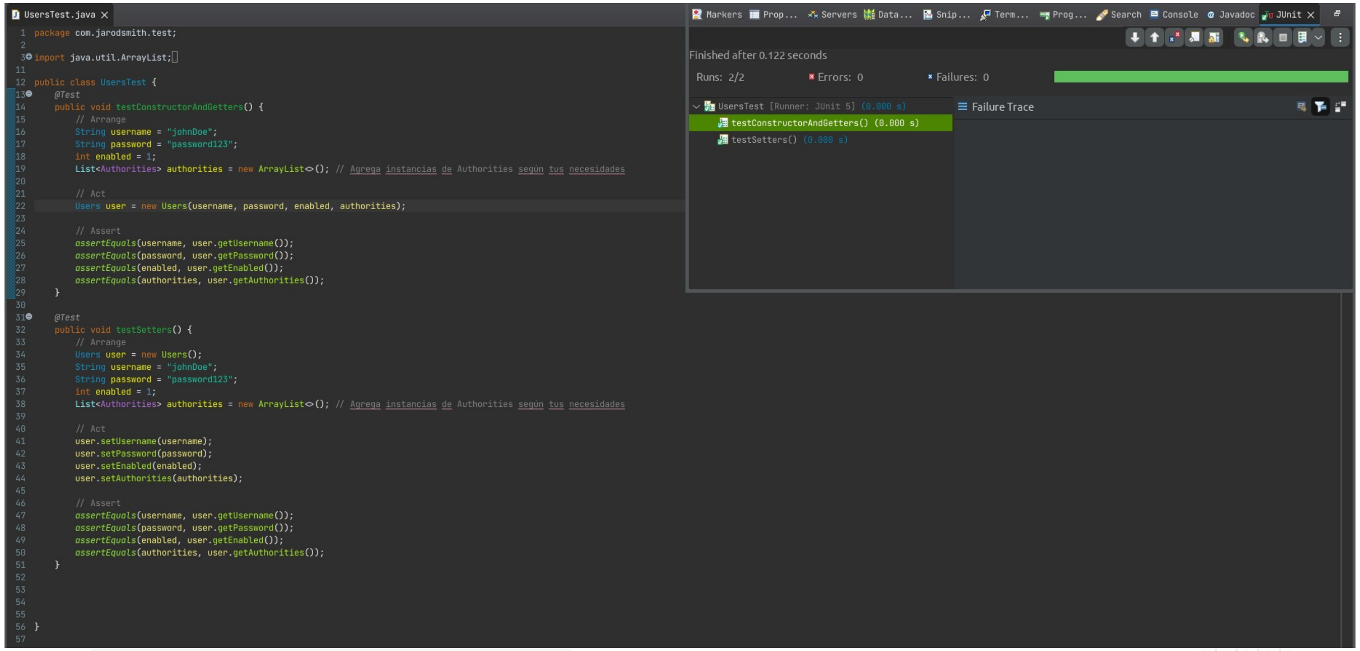
UsersController.java

```java

    @PostMapping("/actualizarUsuario")
    public ModelAndView actualizarUsuario(@ModelAttribute("userForm") Users userForm,
                                          @RequestParam(value = "rolesAsignados", required = false) List<String> rolesAsignados,
                                          BindingResult result) {

        ModelAndView mav = new ModelAndView();

        //VALIDACION DE ERRORES
        /*if(result.hasErrors()) {
            mav.setViewName("users/editUsersForm");
            return mav;
        }
        */
        //ELIMINAR ROLES ANTIGUOS DEL USUARIO
        authoritiesDAO.eliminarRolesPorUsername(userForm.getUsername());

        //ASIGNAR NUEVOS ROLES AL USUARIO
        List<Authorities> listaRoles = new ArrayList<>();

        if(rolesAsignados ≠ null) {
            for(String rol: rolesAsignados) {
                System.out.println("[ROL] " + rol);
                //CREAR UN OBJETO POR CADA ROL
                Authorities authorities = new Authorities();
                authorities.setUsername(userForm.getUsername());
                authorities.setAuthority(rol);

                //System.out.println("[ACTUALIZARUSUARIO]: " + authorities.toString()); //DEBUG

                //GUARDAR AUTHORITIES EN LA BD
                authoritiesDAO.insertar(authorities);

                //AÑADIR CADA OBJETO A LA LISTA
                listaRoles.add(authorities);

            }
        }
```

Algoritmo de cálculo y unidades de prueba

11. TESTING

Utilización de sentencias repetitivas



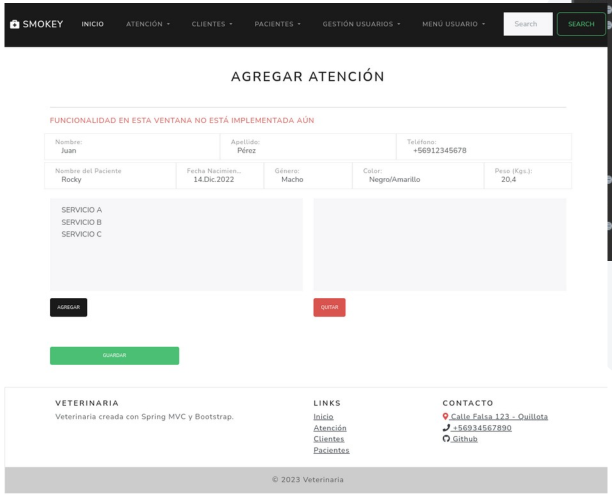
```java
package com.jarodsmith.test;

import java.util.ArrayList;

public class UsersTest {
    @Test
    public void testConstructorAndGetters() {
        // Arrange
        String username = "johnDoe";
        String password = "password123";
        int enabled = 1;
        List<Authorities> authorities = new ArrayList<>(); // Agrega instancias de Authorities según tus necesidades

        // Act
        Users user = new Users(username, password, enabled, authorities);

        // Assert
        assertEquals(username, user.getUsername());
        assertEquals(password, user.getPassword());
        assertEquals(enabled, user.getEnabled());
        assertEquals(authorities, user.getAuthorities());
    }

    @Test
    public void testSetters() {
        // Arrange
        Users user = new Users();
        String username = "johnDoe";
        String password = "password123";
        int enabled = 1;
        List<Authorities> authorities = new ArrayList<>(); // Agrega instancias de Authorities según tus necesidades

        // Act
        user.setUsername(username);
        user.setPassword(password);
        user.setEnabled(enabled);
        user.setAuthorities(authorities);

        // Assert
        assertEquals(username, user.getUsername());
        assertEquals(password, user.getPassword());
        assertEquals(enabled, user.getEnabled());
        assertEquals(authorities, user.getAuthorities());
    }




}
```

```
Finished after 0.122 seconds
Runs: 2/2        Errors: 0        Failures: 0

UsersTest [Runner: JUnit 5] (0.000 s)
    testConstructorAndGetters() (0.000 s)
    testSetters() (0.000 s)
```

Página web html y css

12. Uso Tags HTML

Utilización de tags html, estilos y responsividad



AGREGAR ATENCIÓN

FUNCIONALIDAD EN ESTA VENTANA NO ESTÁ IMPLEMENTADA AÚN

Nombre: Juan	Apellido: Pérez	Teléfono: +56912345678

Nombre del Paciente: Rocky	Fecha Nacimien...: 14.Dic.2022	Género: Macho	Color: Negro/Amarillo	Peso (Kgs.): 20,4

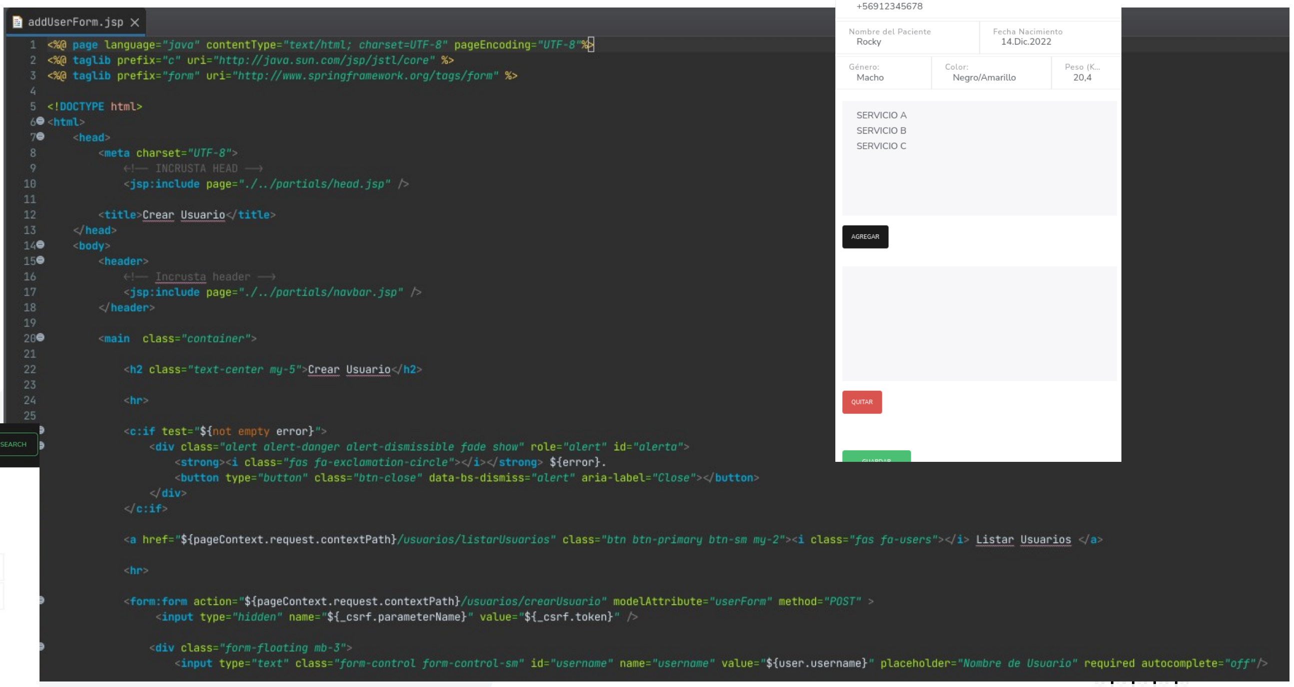
SERVICIO A
SERVICIO B
SERVICIO C

VETERINARIA
Veterinaria creada con Spring MVC y Bootstrap.

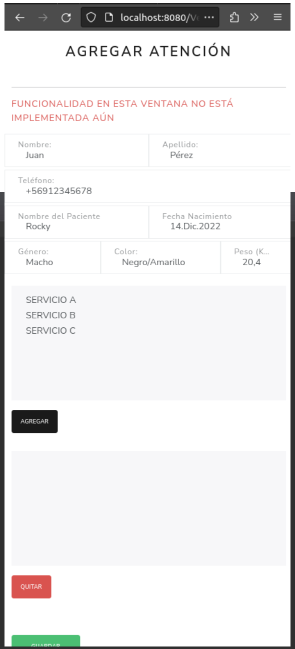
LINKS
Inicio
Atención
Clientes
Pacientes

CONTACTO
Calle Falsa 123 - Quillota
+56934567890
Github

© 2023 Veterinaria



```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>

<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
            <!-- INCRUSTA HEAD -->
            <jsp:include page="./../partials/head.jsp" />

        <title>Crear Usuario</title>
    </head>
    <body>
        <header>
            <!-- Incrusta header -->
            <jsp:include page="./../partials/navbar.jsp" />
        </header>

        <main  class="container">

            <h2 class="text-center my-5">Crear Usuario</h2>

            <hr>

            <c:if test="${not empty error}">
                <div class="alert alert-danger alert-dismissible fade show" role="alert" id="alerta">
                    <strong><i class="fas fa-exclamation-circle"></i></strong> ${error}.
                    <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
                </div>
            </c:if>

            <a href="${pageContext.request.contextPath}/usuarios/listarUsuarios" class="btn btn-primary btn-sm my-2"><i class="fas fa-users"></i> Listar Usuarios </a>

            <hr>

            <form:form action="${pageContext.request.contextPath}/usuarios/crearUsuario" modelAttribute="userForm" method="POST" >
                <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}" />

                <div class="form-floating mb-3">
                    <input type="text" class="form-control form-control-sm" id="username" name="username" value="${user.username}" placeholder="Nombre de Usuario" required autocomplete="off"/>
```



AGREGAR ATENCIÓN

FUNCIONALIDAD EN ESTA VENTANA NO ESTÁ IMPLEMENTADA AÚN

Nombre: Juan	Apellido: Pérez

Teléfono: +56912345678

Nombre del Paciente: Rocky	Fecha Nacimiento: 14.Dic.2022

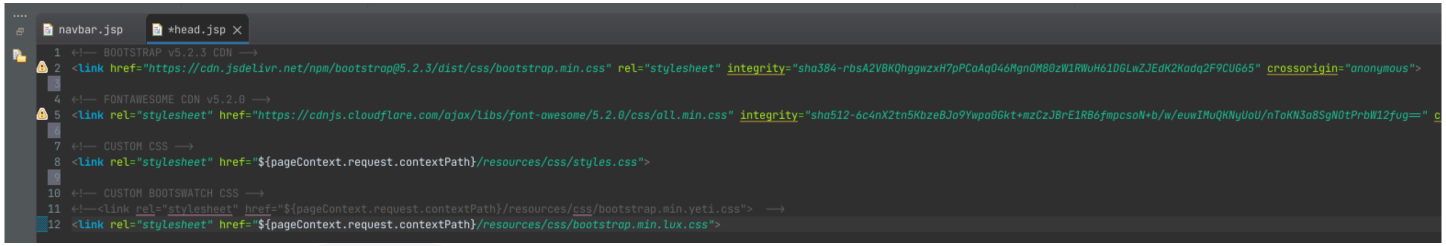
Género: Macho	Color: Negro/Amarillo	Peso (K...: 20,4

SERVICIO A
SERVICIO B
SERVICIO C

Página web html y css

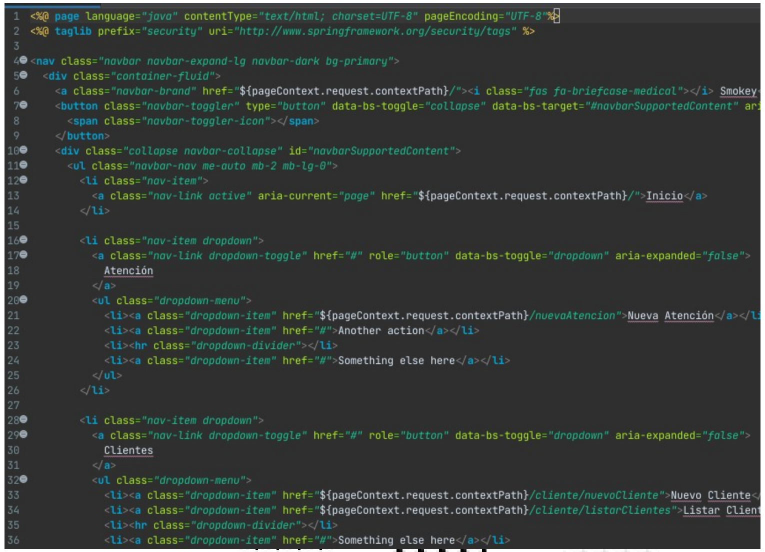
13. Boostrap

Utilización de Bootstrap



navbar.jsp | *head.jsp

```
<!-- BOOTSTRAP v5.2.3 CDN -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65" crossorigin="anonymous">

<!-- FONTAWESOME CDN v5.2.0 -->
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.2.0/css/all.min.css" integrity="sha512-6c4nX2tn5KbzeBJo9YwpaOGkt+mzCzJBrE1RB6fmpcsoN+b/w/euwIMuQKNyUoU/nToKN3a8SgNOtPrbW12fug==" c

<!-- CUSTOM CSS -->
<link rel="stylesheet" href="${pageContext.request.contextPath}/resources/css/styles.css">

<!-- CUSTOM BOOTSWATCH CSS -->
<!--<link rel="stylesheet" href="${pageContext.request.contextPath}/resources/css/bootstrap.min.yeti.css">  -->
<link rel="stylesheet" href="${pageContext.request.contextPath}/resources/css/bootstrap.min.lux.css">
```

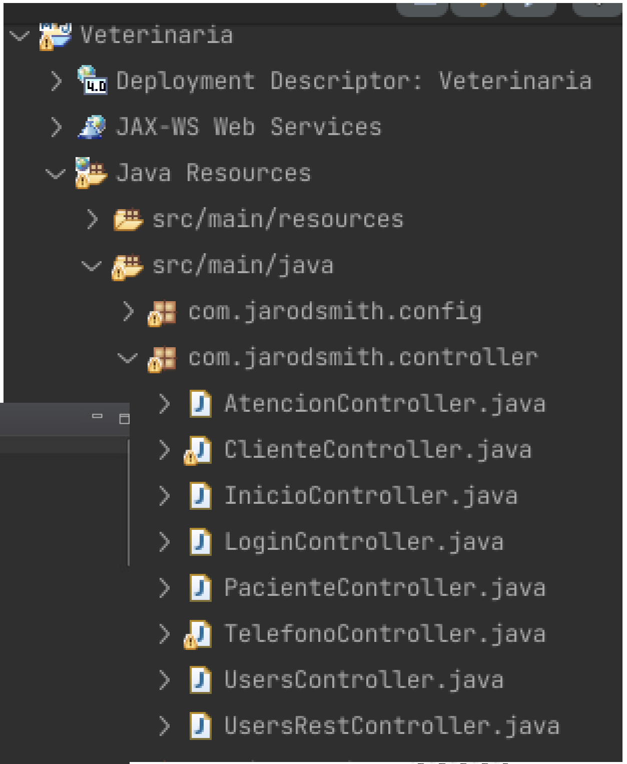


```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="security" uri="http://www.springframework.org/security/tags" %>

<nav class="navbar navbar-expand-lg navbar-dark bg-primary">
  <div class="container-fluid">
    <a class="navbar-brand" href="${pageContext.request.contextPath}/"><i class="fas fa-briefcase-medical"></i> Smokey
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" ari
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="${pageContext.request.contextPath}/">Inicio</a>
        </li>

        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">
            Atención
          </a>
          <ul class="dropdown-menu">
            <li><a class="dropdown-item" href="${pageContext.request.contextPath}/nuevaAtencion">Nueva Atención</a></li
            <li><a class="dropdown-item" href="#">Another action</a></li>
            <li><hr class="dropdown-divider"></li>
            <li><a class="dropdown-item" href="#">Something else here</a></li>
          </ul>
        </li>

        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">
            Clientes
          </a>
          <ul class="dropdown-menu">
            <li><a class="dropdown-item" href="${pageContext.request.contextPath}/cliente/nuevoCliente">Nuevo Cliente</
            <li><a class="dropdown-item" href="${pageContext.request.contextPath}/cliente/listarClientes">Listar Client
            <li><hr class="dropdown-divider"></li>
            <li><a class="dropdown-item" href="#">Something else here</a></li>
```

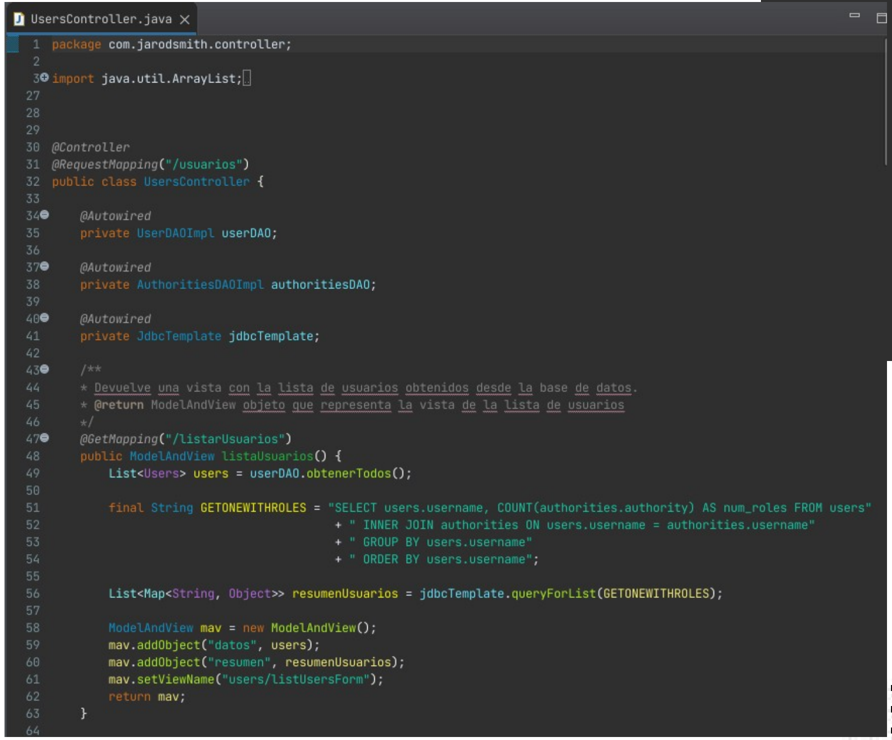
Spring MVC

14. Controllers

Utilización de Controllers



```
Veterinaria
  > Deployment Descriptor: Veterinaria
  > JAX-WS Web Services
  v Java Resources
    > src/main/resources
    v src/main/java
      > com.jarodsmith.config
      v com.jarodsmith.controller
        > AtencionController.java
        > ClienteController.java
        > InicioController.java
        > LoginController.java
        > PacienteController.java
        > TelefonoController.java
        > UsersController.java
        > UsersRestController.java
```



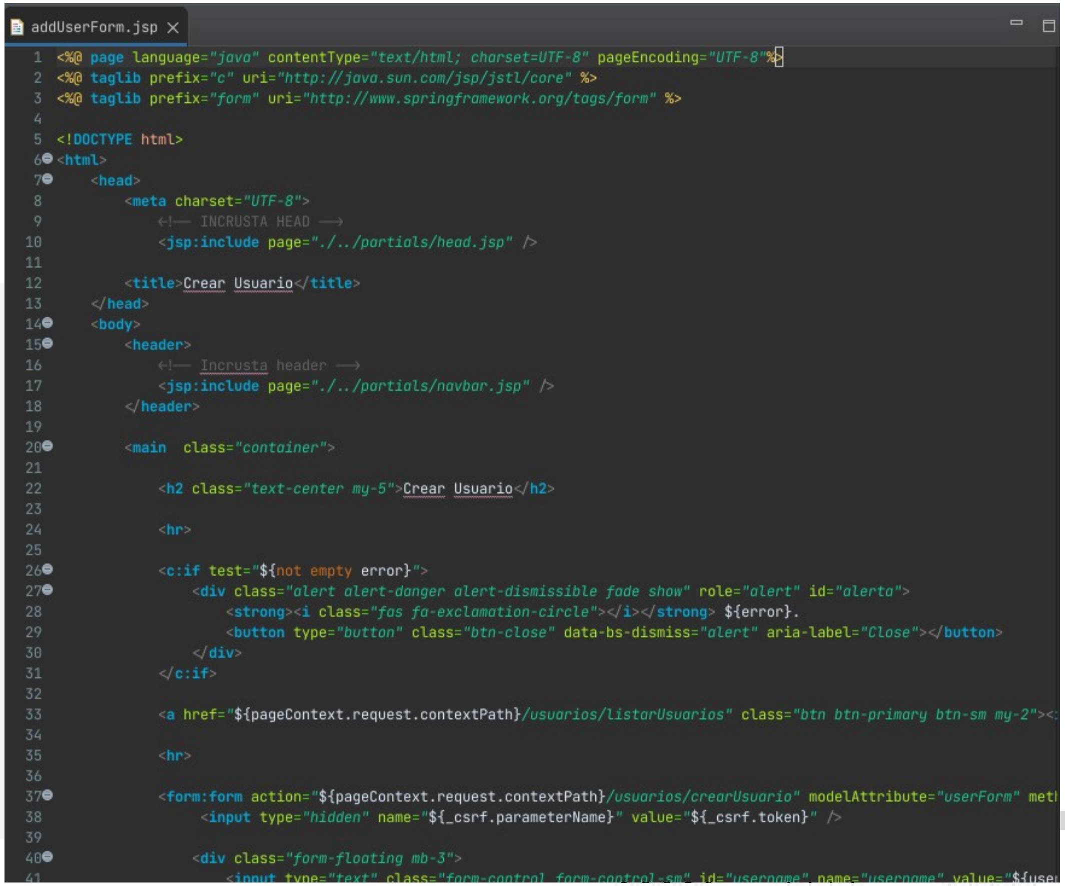
UsersController.java

```java
package com.jarodsmith.controller;

import java.util.ArrayList;



@Controller
@RequestMapping("/usuarios")
public class UsersController {

    @Autowired
    private UserDAOImpl userDAO;

    @Autowired
    private AuthoritiesDAOImpl authoritiesDAO;

    @Autowired
    private JdbcTemplate jdbcTemplate;

    /**
    * Devuelve una vista con la lista de usuarios obtenidos desde la base de datos.
    * @return ModelAndView objeto que representa la vista de la lista de usuarios
    */
    @GetMapping("/listarUsuarios")
    public ModelAndView listaUsuarios() {
        List<Users> users = userDAO.obtenerTodos();

        final String GETONEWITHROLES = "SELECT users.username, COUNT(authorities.authority) AS num_roles FROM users"
                                      + " INNER JOIN authorities ON users.username = authorities.username"
                                      + " GROUP BY users.username"
                                      + " ORDER BY users.username";

        List<Map<String, Object>> resumenUsuarios = jdbcTemplate.queryForList(GETONEWITHROLES);

        ModelAndView mav = new ModelAndView();
        mav.addObject("datos", users);
        mav.addObject("resumen", resumenUsuarios);
        mav.setViewName("users/listUsersForm");
        return mav;
    }

```

Spring MVC

15. Vistas JSP y Taglib

Utilización de vistas JSP y Taglib



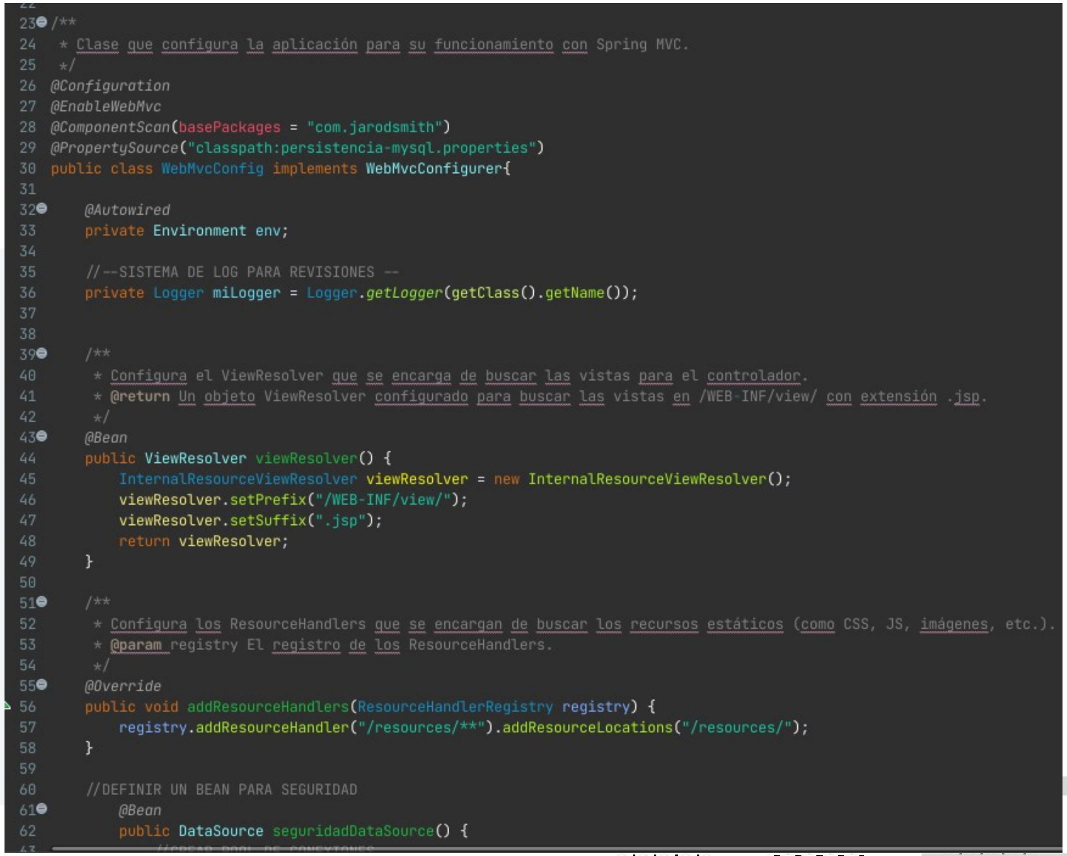
addUserForm.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>

<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
            <!-- INCRUSTA HEAD -->
            <jsp:include page="./../partials/head.jsp" />

        <title>Crear Usuario</title>
    </head>
    <body>
        <header>
            <!-- Incrusta header -->
            <jsp:include page="./../partials/navbar.jsp" />
        </header>

        <main  class="container">

            <h2 class="text-center my-5">Crear Usuario</h2>

            <hr>

            <c:if test="${not empty error}">
                <div class="alert alert-danger alert-dismissible fade show" role="alert" id="alerta">
                    <strong><i class="fas fa-exclamation-circle"></i></strong> ${error}.
                    <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
                </div>
            </c:if>

            <a href="${pageContext.request.contextPath}/usuarios/listarUsuarios" class="btn btn-primary btn-sm my-2"><

            <hr>

            <form:form action="${pageContext.request.contextPath}/usuarios/crearUsuario" modelAttribute="userForm" meth
                 <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}" />

                <div class="form-floating mb-3">
                    <input type="text" class="form-control form-control-sm" id="username" name="username" value="${use
```

Spring MVC

16. Servicio Spring

Creación Servicio Spring

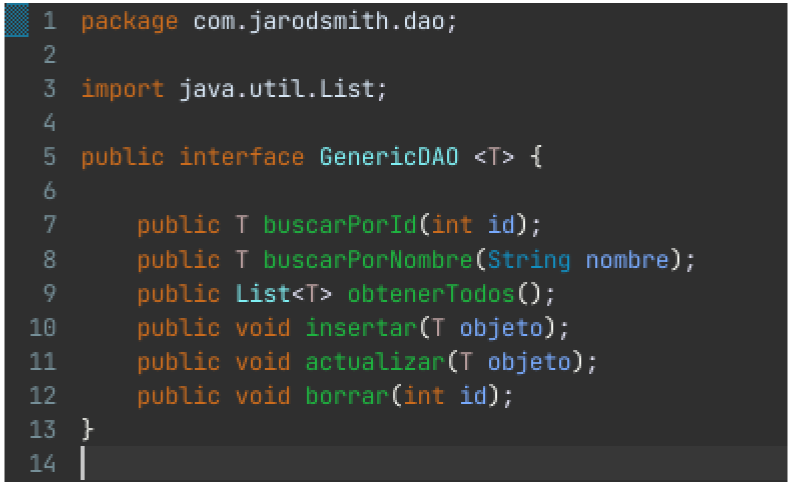


```java
/**
 * Clase que configura la aplicación para su funcionamiento con Spring MVC.
 */
@Configuration
@EnableWebMvc
@ComponentScan(basePackages = "com.jarodsmith")
@PropertySource("classpath:persistencia-mysql.properties")
public class WebMvcConfig implements WebMvcConfigurer{

    @Autowired
    private Environment env;

    //--SISTEMA DE LOG PARA REVISIONES --
    private Logger miLogger = Logger.getLogger(getClass().getName());


    /**
     * Configura el ViewResolver que se encarga de buscar las vistas para el controlador.
     * @return Un objeto ViewResolver configurado para buscar las vistas en /WEB-INF/view/ con extensión .jsp.
     */
    @Bean
    public ViewResolver viewResolver() {
        InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
        viewResolver.setPrefix("/WEB-INF/view/");
        viewResolver.setSuffix(".jsp");
        return viewResolver;
    }

    /**
     * Configura los ResourceHandlers que se encargan de buscar los recursos estáticos (como CSS, JS, imágenes, etc.).
     * @param registry El registro de los ResourceHandlers.
     */
    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/resources/**").addResourceLocations("/resources/");
    }

    //DEFINIR UN BEAN PARA SEGURIDAD
        @Bean
        public DataSource seguridadDataSource() {
```

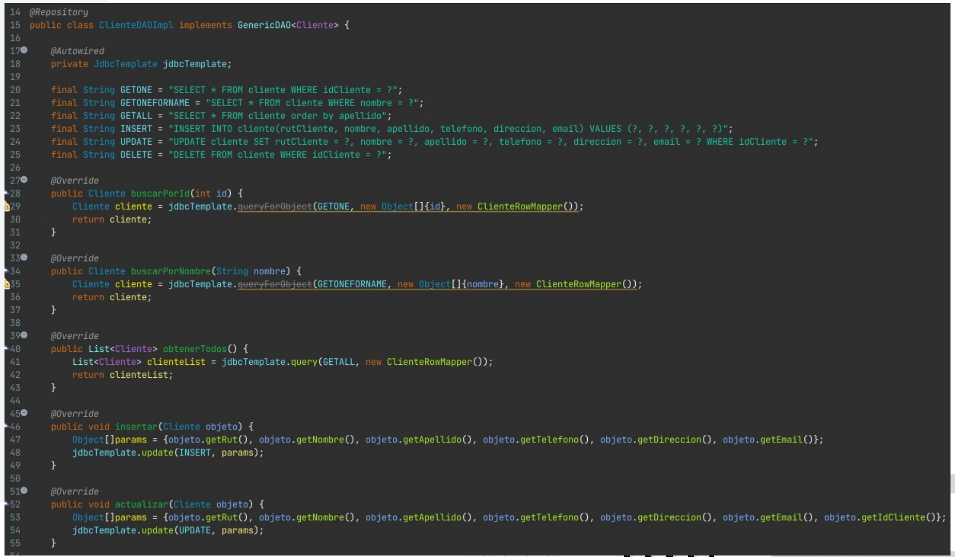
Spring MVC

17. DAO

Creación DAO acceso a datos



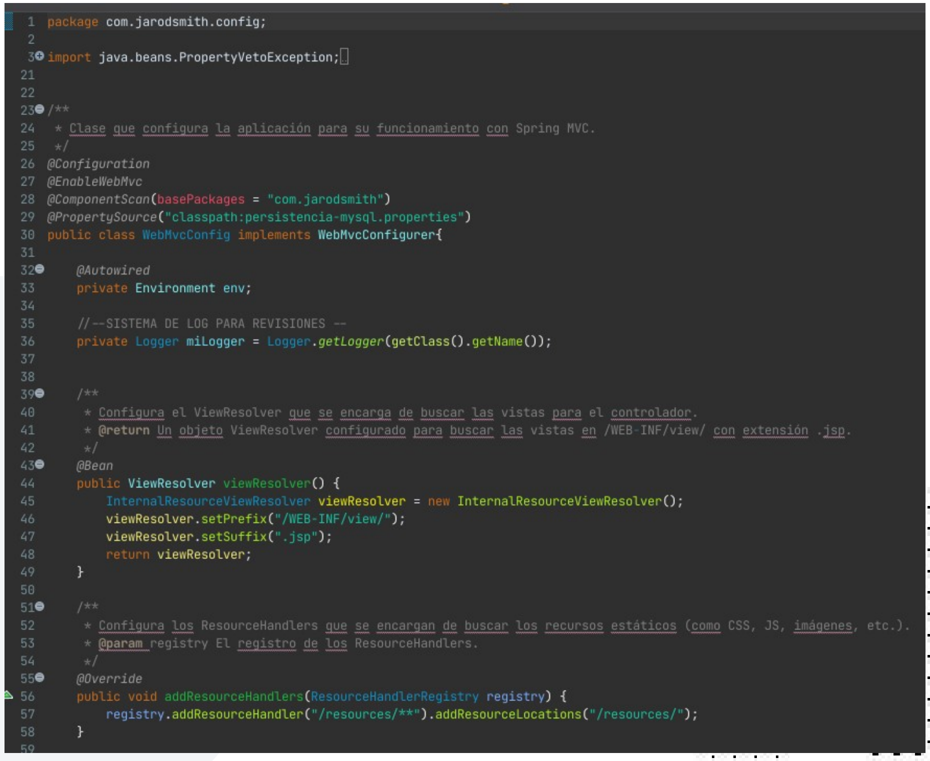
```java
package com.jarodsmith.dao;

import java.util.List;

public interface GenericDAO <T> {

    public T buscarPorId(int id);
    public T buscarPorNombre(String nombre);
    public List<T> obtenerTodos();
    public void insertar(T objeto);
    public void actualizar(T objeto);
    public void borrar(int id);
}
```



```java
@Repository
public class ClienteDAOImpl implements GenericDAO<Cliente> {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    final String GETONE = "SELECT * FROM cliente WHERE idCliente = ?";
    final String GETONEFORNAME = "SELECT * FROM cliente WHERE nombre = ?";
    final String GETALL = "SELECT * FROM cliente order by apellido";
    final String INSERT = "INSERT INTO cliente(rutCliente, nombre, apellido, telefono, direccion, email) VALUES (?, ?, ?, ?, ?, ?)";
    final String UPDATE = "UPDATE cliente SET rutCliente = ?, nombre = ?, apellido = ?, telefono = ?, direccion = ?, email = ? WHERE idCliente = ?";
    final String DELETE = "DELETE FROM cliente WHERE idCliente = ?";

    @Override
    public Cliente buscarPorId(int id) {
        Cliente cliente = jdbcTemplate.~~queryForObject~~(GETONE, new Object[]{id}, new ClienteRowMapper());
        return cliente;
    }

    @Override
    public Cliente buscarPorNombre(String nombre) {
        Cliente cliente = jdbcTemplate.~~queryForObject~~(GETONEFORNAME, new Object[]{nombre}, new ClienteRowMapper());
        return cliente;
    }

    @Override
    public List<Cliente> obtenerTodos() {
        List<Cliente> clienteList = jdbcTemplate.query(GETALL, new ClienteRowMapper());
        return clienteList;
    }

    @Override
    public void insertar(Cliente objeto) {
        Object[]params = {objeto.getRut(), objeto.getNombre(), objeto.getApellido(), objeto.getTelefono(), objeto.getDireccion(), objeto.getEmail()};
        jdbcTemplate.update(INSERT, params);
    }

    @Override
    public void actualizar(Cliente objeto) {
        Object[]params = {objeto.getRut(), objeto.getNombre(), objeto.getApellido(), objeto.getTelefono(), objeto.getDireccion(), objeto.getEmail(), objeto.getIdCliente()};
        jdbcTemplate.update(UPDATE, params);
    }
```

Spring MVC

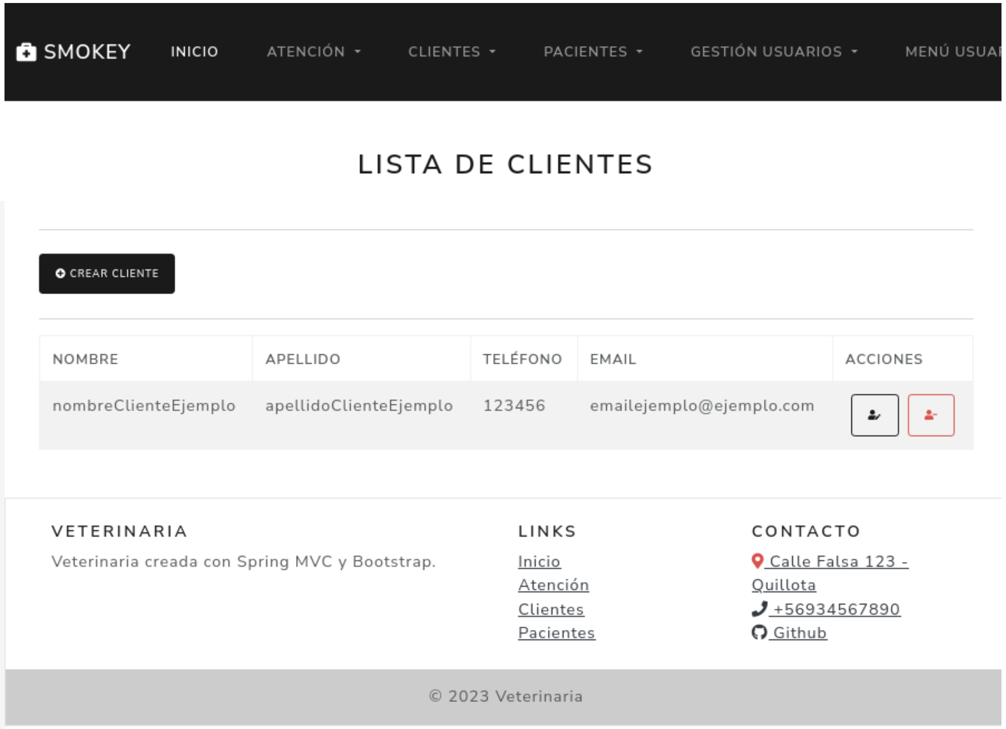
18. Creación del proyecto y configuración



```java
package com.jarodsmith.config;

import java.beans.PropertyVetoException;


/**
 * Clase que configura la aplicación para su funcionamiento con Spring MVC.
 */
@Configuration
@EnableWebMvc
@ComponentScan(basePackages = "com.jarodsmith")
@PropertySource("classpath:persistencia-mysql.properties")
public class WebMvcConfig implements WebMvcConfigurer{

    @Autowired
    private Environment env;

    //--SISTEMA DE LOG PARA REVISIONES --
    private Logger miLogger = Logger.getLogger(getClass().getName());


    /**
     * Configura el ViewResolver que se encarga de buscar las vistas para el controlador.
     * @return Un objeto ViewResolver configurado para buscar las vistas en /WEB-INF/view/ con extensión .jsp.
     */
    @Bean
    public ViewResolver viewResolver() {
        InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
        viewResolver.setPrefix("/WEB-INF/view/");
        viewResolver.setSuffix(".jsp");
        return viewResolver;
    }

    /**
     * Configura los ResourceHandlers que se encargan de buscar los recursos estáticos (como CSS, JS, imágenes, etc.).
     * @param registry El registro de los ResourceHandlers.
     */
    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/resources/**").addResourceLocations("/resources/");
    }
```

Spring MVC

19. Funcionamiento del aplicativo



LISTA DE CLIENTES

NOMBRE	APELLIDO	TELÉFONO	EMAIL	ACCIONES
nombreClienteEjemplo	apellidoClienteEjemplo	123456	emailejemplo@ejemplo.com	

VETERINARIA
Veterinaria creada con Spring MVC y Bootstrap.

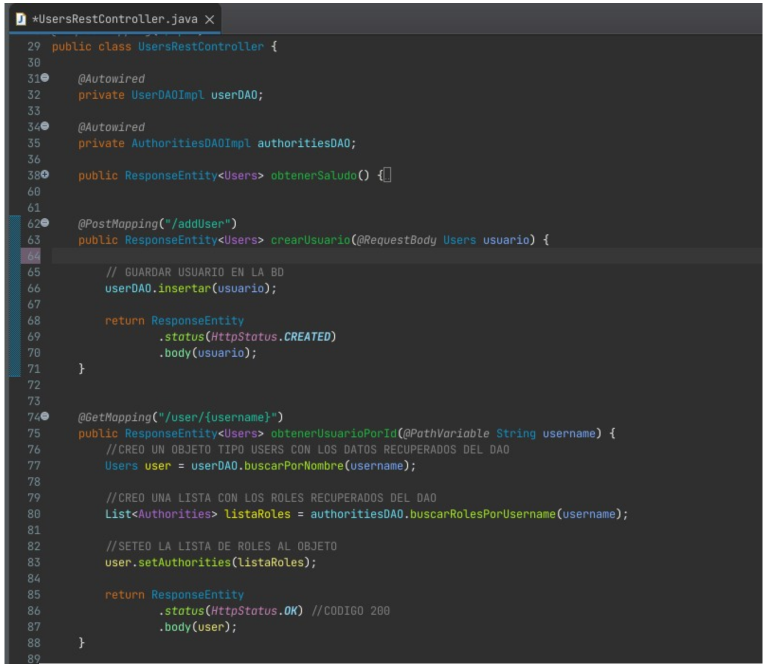
LINKS
Inicio
Atención
Clientes
Pacientes

CONTACTO
Calle Falsa 123 - Quillota
+56934567890
Github

© 2023 Veterinaria

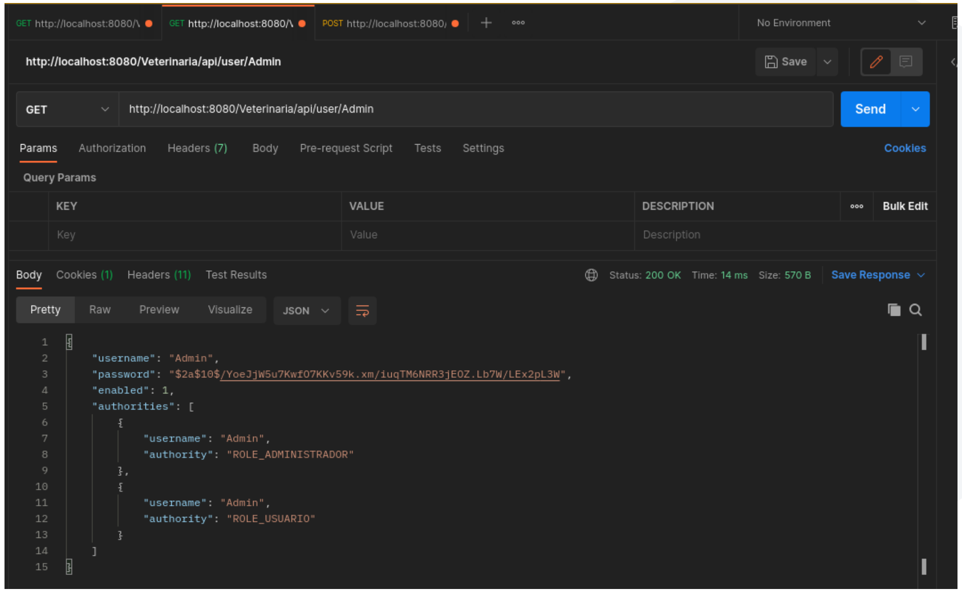
API REST

20. Creación Servicio REST



*UsersRestController.java

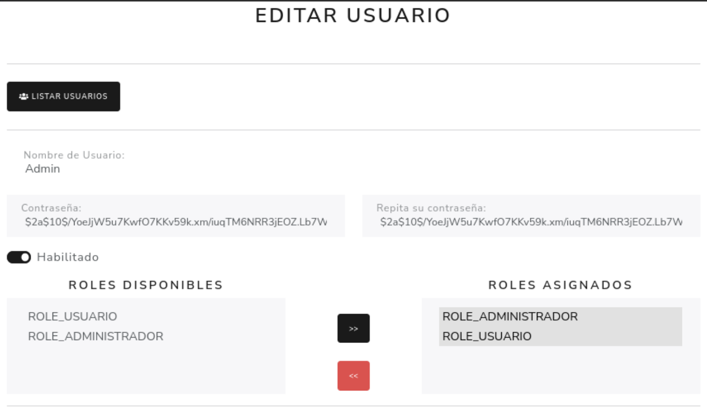
```java
public class UsersRestController {

    @Autowired
    private UserDAOImpl userDAO;

    @Autowired
    private AuthoritiesDAOImpl authoritiesDAO;

    public ResponseEntity<Users> obtenerSaludo() {


    @PostMapping("/addUser")
    public ResponseEntity<Users> crearUsuario(@RequestBody Users usuario) {

        // GUARDAR USUARIO EN LA BD
        userDAO.insertar(usuario);

        return ResponseEntity
                .status(HttpStatus.CREATED)
                .body(usuario);
    }


    @GetMapping("/user/{username}")
    public ResponseEntity<Users> obtenerUsuarioPorId(@PathVariable String username) {
        //CREO UN OBJETO TIPO USERS CON LOS DATOS RECUPERADOS DEL DAO
        Users user = userDAO.buscarPorNombre(username);

        //CREO UNA LISTA CON LOS ROLES RECUPERADOS DEL DAO
        List<Authorities> listaRoles = authoritiesDAO.buscarRolesPorUsername(username);

        //SETEO LA LISTA DE ROLES AL OBJETO
        user.setAuthorities(listaRoles);

        return ResponseEntity
                .status(HttpStatus.OK) //CODIGO 200
                .body(user);
    }

```



GET http://localhost:8080/Veterinaria/api/user/Admin

Status: 200 OK Time: 14 ms Size: 570 B

```json
{
    "username": "Admin",
    "password": "$2a$10$/YoeJjW5u7KwfO7KKv59k.xm/iuqTM6NRR3jEOZ.Lb7W/LEx2pL3W",
    "enabled": 1,
    "authorities": [
        {
            "username": "Admin",
            "authority": "ROLE_ADMINISTRADOR"
        },
        {
            "username": "Admin",
            "authority": "ROLE_USUARIO"
        }
    ]
}
```



EDITAR USUARIO

Nombre de Usuario: Admin

Contraseña: $2a$10$/YoeJjW5u7KwfO7KKv59k.xm/iuqTM6NRR3jEOZ.Lb7W

Repita su contraseña: $2a$10$/YoeJjW5u7KwfO7KKv59k.xm/iuqTM6NRR3jEOZ.Lb7W

Habilitado

ROLES DISPONIBLES	ROLES ASIGNADOS
ROLE_USUARIO	ROLE_ADMINISTRADOR
ROLE_ADMINISTRADOR	ROLE_USUARIO