



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.

```
/**
 * 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 listarUsuarios() {
    List<Users> users = userDAO.obtenerTodos();

    final String GETONEWITHROLES = "SELECT users.username, COUNT(authorities.authority) AS num_rols 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_rols 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

```
@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

```
/**
 * 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 listarUsuarios() {
    List<Users> users = userDAO.obtenerTodos();

    final String GETONEWITHROLES = "SELECT users.username, COUNT(authorities.authority) AS num_rols 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

```
/**
 * 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 listarUsuarios() {
    List<Users> users = userDAO.obtenerTodos();

    final String GETONEWITHROLES = "SELECT users.username, COUNT(authorities.authority) AS num_rols 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 X
89
90 @PostMapping("/actualizarUsuario")
91 public ModelAndView actualizarUsuario(@ModelAttribute("userForm") Users userForm,
92                                     @RequestParam(value = "rolesAsignados", required = false) List<String> rolesAsignados,
93                                     BindingResult result) {
94
95     ModelAndView mav = new ModelAndView();
96
97     //VALIDACION DE ERRORES
98     /*if(result.hasErrors()) {
99         mav.setViewName("users/editUsersForm");
100         return mav;
101     }
102     */
103     //ELIMINAR ROLES ANTIGUOS DEL USUARIO
104     authoritiesDAO.eliminarRolesPorUsername(userForm.getUsername());
105
106     //ASIGNAR NUEVOS ROLES AL USUARIO
107     List<Authorities> listaRoles = new ArrayList<>();
108
109     if(rolesAsignados != null) {
110         for(String rol: rolesAsignados) {
111             System.out.println("[ROL] " + rol);
112             //CREAR UN OBJETO POR CADA ROL
113             Authorities authorities = new Authorities();
114             authorities.setUsername(userForm.getUsername());
115             authorities.setAuthority(rol);
116
117             //System.out.println("[ACTUALIZARUSUARIO]: " + authorities.toString()); //DEBUG
118
119             //GUARDAR AUTHORITIES EN LA BD
120             authoritiesDAO.insertar(authorities);
121
122             //AÑADIR CADA OBJETO A LA LISTA
123             listaRoles.add(authorities);
124
125         }
126     }
```

Algoritmo de cálculo y unidades de prueba

7. SENTENCIAS REPETITIVAS

Utilización de
sentencias
repetitivas

```
UsersController.java X
89
90 @PostMapping("/actualizarUsuario")
91 public ModelAndView actualizarUsuario(@ModelAttribute("userForm") Users userForm,
92                                     @RequestParam(value = "rolesAsignados", required = false) List<String> rolesAsignados,
93                                     BindingResult result) {
94
95     ModelAndView mav = new ModelAndView();
96
97     //VALIDACION DE ERRORES
98     /*if(result.hasErrors()) {
99         mav.setViewName("users/editUsersForm");
100         return mav;
101     }
102     */
103     //ELIMINAR ROLES ANTIGUOS DEL USUARIO
104     authoritiesDAO.eliminarRolesPorUsername(userForm.getUsername());
105
106     //ASIGNAR NUEVOS ROLES AL USUARIO
107     List<Authorities> listaRoles = new ArrayList<>();
108
109     if(rolesAsignados != null) {
110         for(String rol: rolesAsignados) {
111             System.out.println("[ROL] " + rol);
112             //CREAR UN OBJETO POR CADA ROL
113             Authorities authorities = new Authorities();
114             authorities.setUsername(userForm.getUsername());
115             authorities.setAuthority(rol);
116
117             //System.out.println("[ACTUALIZARUSUARIO]: " + authorities.toString()); //DEBUG
118
119             //GUARDAR AUTHORITIES EN LA BD
120             authoritiesDAO.insertar(authorities);
121
122             //AÑADIR CADA OBJETO A LA LISTA
123             listaRoles.add(authorities);
124
125         }
126     }
```


Algoritmo de cálculo y unidades de prueba

8. Uso de clases, encapsulamiento, responsabilidad única.

Utilización de clases, encapsulamiento y responsabilidad única

```
11
12 public class Users {
13
14     /**
15      * Miembros de Clase: propiedades
16      */
17     @NotNull
18     @NotBlank(message = "Campo requerido")
19     // @Size(min = 2, message = "Campo requerido")
20     private String username;
21     @NotNull
22     @NotBlank(message = "Campo requerido")
23     private String password;
24     private int enabled;
25     private List<Authorities> authorities;
26
27     /**
28      * Constructor predeterminado
29      */
30     public Users() {
31     }
32
33     /**
34      * Constructor Parametrizado
35      * @param username
36      * @param password
37      * @param enabled
38      * @param authorities
39      */
40     public Users(String username, String password, int enabled, List<Authorities> authorities) {
41         this.username = username;
42         this.password = password;
43         this.enabled = enabled;
44         this.authorities = authorities;
45     }
46
47
48     public String getUsername() {
49         return username;
50     }
51     public void setUsername(String username) {
52         this.username = username;
53     }
54     public String getPassword() {
55         return password;
56     }
57     public void setPassword(String password) {
58         this.password = password;
59     }
60 }
```

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

```
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;  
    }  
}
```

```
@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;  
    }  
}
```

Algoritmo de cálculo y unidades de prueba

10. Convenciones

Convenciones
y estilos de
programación.

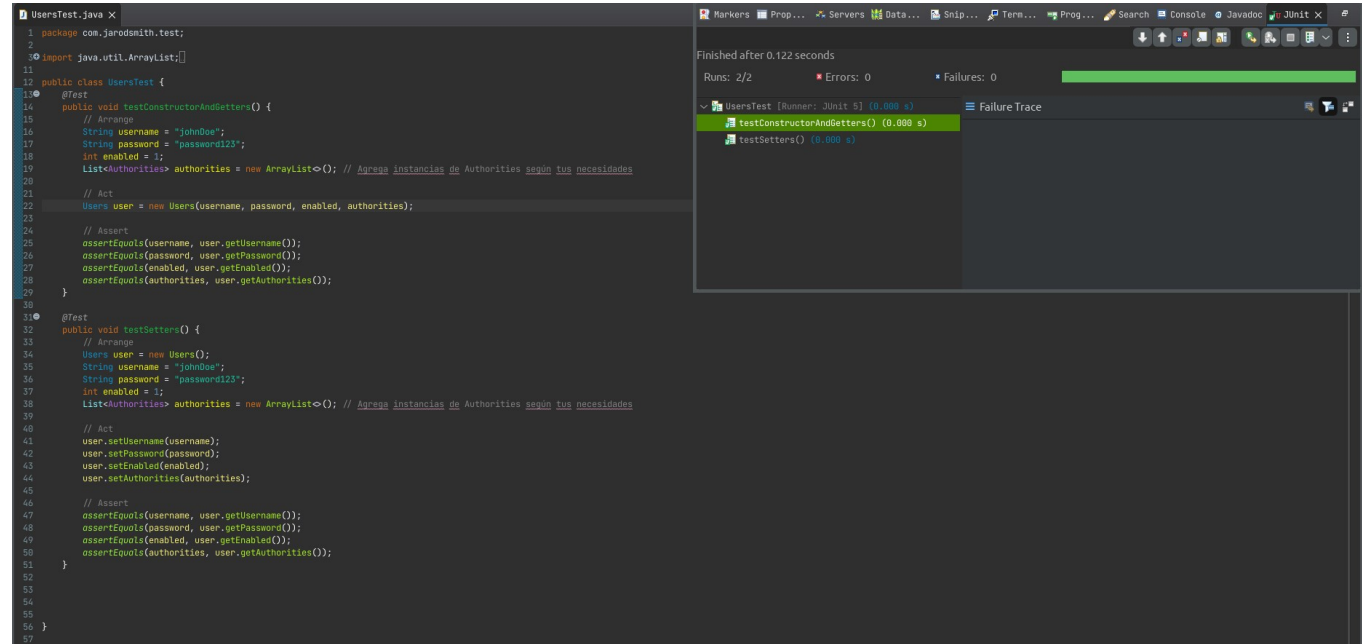
```
public class Cliente extends Base{  
  
    /**  
     * Miembros de Clase: atributos  
     */  
    private int idCliente;  
    private String rut;  
    private String nombre;  
    private String apellido;  
    private String telefono;  
    private String direccion;  
    private String email;  
  
    /**  
     * 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 X  
89  
90 @PostMapping("/actualizarUsuario")  
91 public ModelAndView actualizarUsuario(@ModelAttribute("userForm") Users userForm,  
92 @RequestParam(value = "rolesAsignados", required = false) List<String> rolesAsignados,  
93 BindingResult result) {  
94  
95     ModelAndView mav = new ModelAndView();  
96  
97     //VALIDACION DE ERRORES  
98     /*if(result.hasErrors()) {  
99         mav.setViewName("users/editUsersForm");  
100         return mav;  
101     }  
102     */  
103     //ELIMINAR ROLES ANTIGUOS DEL USUARIO  
104     authoritiesDAO.eliminarRolesPorUsername(userForm.getUsername());  
105  
106     //ASIGNAR NUEVOS ROLES AL USUARIO  
107     List<Authorities> listaRoles = new ArrayList<>();  
108  
109     if(rolesAsignados != null) {  
110         for(String rol: rolesAsignados) {  
111             System.out.println("[ROL] " + rol);  
112             //CREAR UN OBJETO POR CADA ROL  
113             Authorities authorities = new Authorities();  
114             authorities.setUsername(userForm.getUsername());  
115             authorities.setAuthority(rol);  
116  
117             //System.out.println("[ACTUALIZARUSUARIO]: " + authorities.toString()); //DEBUG  
118  
119             //GUARDAR AUTHORITIES EN LA BD  
120             authoritiesDAO.insertar(authorities);  
121  
122             //AÑADIR CADA OBJETO A LA LISTA  
123             listaRoles.add(authorities);  
124         }  
125     }  
126 }
```

Algoritmo de cálculo y unidades de prueba

11. TESTING

Utilización de
sentencias
repetitivas



The screenshot displays an IDE with a Java test class named `UsersTest.java` and its execution results. The code is written in Java and uses JUnit 5 for testing. It includes two test methods: `testConstructorAndGetters()` and `testSetters()`. Both methods use the Arrange-Act-Assert pattern. The `testConstructorAndGetters()` method creates a `Users` object using a constructor and then asserts that the getters return the expected values. The `testSetters()` method creates a `Users` object using a no-argument constructor and then asserts that the setters set the expected values. The execution results show that both tests passed successfully, with a total of 2 runs, 0 errors, and 0 failures. The test `testConstructorAndGetters()` took 0.008 seconds to execute, and `testSetters()` took 0.000 seconds.


```
1 package com.jarodsmith.test;
2
3 import java.util.ArrayList;
4
5
6
7
8
9
10
11
12 public class UsersTest {
13     @Test
14     public void testConstructorAndGetters() {
15         // Arrange
16         String username = "johnDoe";
17         String password = "password123";
18         int enabled = 1;
19         List<Authorities> authorities = new ArrayList<>(); // Agrega instancias de Authorities según tus necesidades
20
21         // Act
22         Users user = new Users(username, password, enabled, authorities);
23
24         // Assert
25         assertEquals(username, user.getUsername());
26         assertEquals(password, user.getPassword());
27         assertEquals(enabled, user.getEnabled());
28         assertEquals(authorities, user.getAuthorities());
29     }
30
31     @Test
32     public void testSetters() {
33         // Arrange
34         Users user = new Users();
35         String username = "johnDoe";
36         String password = "password123";
37         int enabled = 1;
38         List<Authorities> authorities = new ArrayList<>(); // Agrega instancias de Authorities según tus necesidades
39
40         // Act
41         user.setUsername(username);
42         user.setPassword(password);
43         user.setEnabled(enabled);
44         user.setAuthorities(authorities);
45
46         // Assert
47         assertEquals(username, user.getUsername());
48         assertEquals(password, user.getPassword());
49         assertEquals(enabled, user.getEnabled());
50         assertEquals(authorities, user.getAuthorities());
51     }
52
53
54
55
56
57 }
```

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

UsersTest (Runner: JUnit 5) (0.008 s)
testConstructorAndGetters() (0.008 s)
testSetters() (0.000 s)

12. Use Tags HTML

Utilización de tags html, estilos y responsividad


SMOKEY

[INICIO](#)
[ATENCIÓN](#)
[CLIENTES](#)
[PACIENTES](#)
[GESTIÓN USUARIOS](#)
[MENÚ USUARIO](#)

AGREGAR ATENCIÓN

Nombre:

Juan

Apellido:

Pérez

Teléfono:

+56912345678

Nombre del Paciente

Rocky

Fecha Nacimiento:

14 Dic 2022

Genero:

Macho

Color:

Negro/Amarillo

Peso (Kg.):

20.4

SERVICIO A

SERVICIO B

SERVICIO C

Cancelar

Guardar

Guardar

Nombre:

Juan

Apellido:

Pérez

Teléfono:

+56912345678

Nombre del Paciente

Rocky

Fecha Nacimiento:

14 Dic 2022

Genero:

Macho

Color:

Negro/Amarillo

Peso (Kg.):

20.4

SERVICIO A

SERVICIO B

SERVICIO C

Cancelar

Guardar

Guardar

Nombre:

Juan

Apellido:

Pérez

Teléfono:

+56912345678

Nombre del Paciente

Rocky

Fecha Nacimiento:

14 Dic 2022

Genero:

Macho

Color:

Negro/Amarillo

Peso (Kg.):

20.4

SERVICIO A

SERVICIO B

SERVICIO C

Cancelar

Guardar

Guardar

Nombre:

Juan

Apellido:

Pérez

Teléfono:

+56912345678

Nombre del Paciente

Rocky

Fecha Nacimiento:

14 Dic 2022

Genero:

Macho

Color:

Negro/Amarillo

Peso (Kg.):

20.4

SERVICIO A

SERVICIO B

SERVICIO C

Cancelar

Guardar

Guardar

Nombre:

Juan

Apellido:

Pérez

Teléfono:

+56912345678

Nombre del Paciente

Rocky

Fecha Nacimiento:

14 Dic 2022

Genero:

Macho

Color:

Negro/Amarillo

Peso (Kg.):

20.4

SERVICIO A

SERVICIO B

SERVICIO C

Cancelar

Guardar

Guardar

Nombre:

Juan

Apellido:

Pérez

Teléfono:

+56912345678

Nombre del Paciente

Rocky

Fecha Nacimiento:

14 Dic 2022

Genero:

Macho

Color:

Negro/Amarillo

Peso (Kg.):

20.4

SERVICIO A

SERVICIO B

SERVICIO C

Cancelar

Guardar

Guardar

addUserForm.jsp X

1<%
2<%
3<%
4<%
5<!DOCTYPE html>
6<html>
7<head>
8<meta charset="UTF-8">
9<!-- INCRUSTA HEAD -->
10<jsp:include page="../partials/head.jsp" />
11<title>Crear Usuario</title>
12</head>
13<body>
14<header>
15<!-- Incrusta header -->
16<jsp:include page="../partials/navbar.jsp" />
17</header>
18<main class="container">
19

20<h2 class="text-center my-5">Crear Usuario</h2>
21

22<hr>
23

24<div class="form-floating mb-3">
25<input type="text" class="form-control form-control-sm" id="username" name="username" value="{user.username}" placeholder="Nombre de Usuario" required autocomplete="off">
26</div>
27<div class="form-floating mb-3">
28<input type="password" class="form-control form-control-sm" id="password" name="password" value="{user.password}" placeholder="Contraseña" required autocomplete="off">
29</div>
30<div class="form-floating mb-3">
31<input type="password" class="form-control form-control-sm" id="confirm_password" name="confirm_password" value="{user.confirm_password}" placeholder="Confirmar contraseña" required autocomplete="off">
32</div>
33<div class="form-floating mb-3">
34<input type="text" class="form-control form-control-sm" id="email" name="email" value="{user.email}" placeholder="Correo electrónico" required autocomplete="off">
35</div>
36<div class="form-floating mb-3">
37<input type="text" class="form-control form-control-sm" id="phone" name="phone" value="{user.phone}" placeholder="Teléfono" required autocomplete="off">
38</div>
39<div class="form-floating mb-3">
40<input type="text" class="form-control form-control-sm" id="address" name="address" value="{user.address}" placeholder="Dirección" required autocomplete="off">
41</div>
42<div class="form-floating mb-3">
43<input type="text" class="form-control form-control-sm" id="city" name="city" value="{user.city}" placeholder="Ciudad" required autocomplete="off">
44</div>
45<div class="form-floating mb-3">
46<input type="text" class="form-control form-control-sm" id="state" name="state" value="{user.state}" placeholder="Estado" required autocomplete="off">
47</div>
48<div class="form-floating mb-3">
49<input type="text" class="form-control form-control-sm" id="zip" name="zip" value="{user.zip}" placeholder="Código postal" required autocomplete="off">
50</div>
51<div class="form-floating mb-3">
52<input type="text" class="form-control form-control-sm" id="birthdate" name="birthdate" value="{user.birthdate}" placeholder="Fecha de nacimiento" required autocomplete="off">
53</div>
54<div class="form-floating mb-3">
55<input type="text" class="form-control form-control-sm" id="gender" name="gender" value="{user.gender}" placeholder="Género" required autocomplete="off">
56</div>
57<div class="form-floating mb-3">
58<input type="text" class="form-control form-control-sm" id="blood_type" name="blood_type" value="{user.blood_type}" placeholder="Tipo de sangre" required autocomplete="off">
59</div>
60<div class="form-floating mb-3">
61<input type="text" class="form-control form-control-sm" id="weight" name="weight" value="{user.weight}" placeholder="Peso (kg)" required autocomplete="off">
62</div>
63<div class="form-floating mb-3">
64<input type="text" class="form-control form-control-sm" id="height" name="height" value="{user.height}" placeholder="Altura (cm)" required autocomplete="off">
65</div>
66<div class="form-floating mb-3">
67<input type="text" class="form-control form-control-sm" id="blood_pressure" name="blood_pressure" value="{user.blood_pressure}" placeholder="Presión arterial" required autocomplete="off">
68</div>
69<div class="form-floating mb-3">
70<input type="text" class="form-control form-control-sm" id="allergies" name="allergies" value="{user.allergies}" placeholder="Alergias" required autocomplete="off">
71</div>
72<div class="form-floating mb-3">
73<input type="text" class="form-control form-control-sm" id="medical_history" name="medical_history" value="{user.medical_history}" placeholder="Historial médico" required autocomplete="off">
74</div>
75<div class="form-floating mb-3">
76<input type="text" class="form-control form-control-sm" id="family_history" name="family_history" value="{user.family_history}" placeholder="Historial familiar" required autocomplete="off">
77</div>
78<div class="form-floating mb-3">
79<input type="text" class="form-control form-control-sm" id="current_medication" name="current_medication" value="{user.current_medication}" placeholder="Medicación actual" required autocomplete="off">
80</div>
81<div class="form-floating mb-3">
82<input type="text" class="form-control form-control-sm" id="previous_medication" name="previous_medication" value="{user.previous_medication}" placeholder="Medicación anterior" required autocomplete="off">
83</div>
84<div class="form-floating mb-3">
85<input type="text" class="form-control form-control-sm" id="current_vaccination" name="current_vaccination" value="{user.current_vaccination}" placeholder="Vacunación actual" required autocomplete="off">
86</div>
87<div class="form-floating mb-3">
88<input type="text" class="form-control form-control-sm" id="previous_vaccination" name="previous_vaccination" value="{user.previous_vaccination}" placeholder="Vacunación anterior" required autocomplete="off">
89</div>
90<div class="form-floating mb-3">
91<input type="text" class="form-control form-control-sm" id="current_diet" name="current_diet" value="{user.current_diet}" placeholder="Dieta actual" required autocomplete="off">
92</div>
93<div class="form-floating mb-3">
94<input type="text" class="form-control form-control-sm" id="previous_diet" name="previous_diet" value="{user.previous_diet}" placeholder="Dieta anterior" required autocomplete="off">
95</div>
96<div class="form-floating mb-3">
97<input type="text" class="form-control form-control-sm" id="current_exercise" name="current_exercise" value="{user.current_exercise}" placeholder="Ejercicio actual" required autocomplete="off">
98</div>
99<div class="form-floating mb-3">
100<input type="text" class="form-control form-control-sm" id="previous_exercise" name="previous_exercise" value="{user.previous_exercise}" placeholder="Ejercicio anterior" required autocomplete="off">
101</div>
102<div class="form-floating mb-3">
103<input type="text" class="form-control form-control-sm" id="current_stress" name="current_stress" value="{user.current_stress}" placeholder="Estrés actual" required autocomplete="off">
104</div>
105<div class="form-floating mb-3">
106<input type="text" class="form-control form-control-sm" id="previous_stress" name="previous_stress" value="{user.previous_stress}" placeholder="Estrés anterior" required autocomplete="off">
107</div>
108<div class="form-floating mb-3">
109<input type="text" class="form-control form-control-sm" id="current_sleep" name="current_sleep" value="{user.current_sleep}" placeholder="Sueño actual" required autocomplete="off">
110</div>
111<div class="form-floating mb-3">
112<input type="text" class="form-control form-control-sm" id="previous_sleep" name="previous_sleep" value="{user.previous_sleep}" placeholder="Sueño anterior" required autocomplete="off">
113</div>
114<div class="form-floating mb-3">
115<input type="text" class="form-control form-control-sm" id="current_mood" name="current_mood" value="{user.current_mood}" placeholder="Estado de ánimo actual" required autocomplete="off">
116</div>
117<div class="form-floating mb-3">
118<input type="text" class="form-control form-control-sm" id="previous_mood" name="previous_mood" value="{user.previous_mood}" placeholder="Estado de ánimo anterior" required autocomplete="off">
119</div>
120<div class="form-floating mb-3">
121<input type="text" class="form-control form-control-sm" id="current_energy" name="current_energy" value="{user.current_energy}" placeholder="Energía actual" required autocomplete="off">
122</div>
123<div class="form-floating mb-3">
124<input type="text" class="form-control form-control-sm" id="previous_energy" name="previous_energy" value="{user.previous_energy}" placeholder="Energía anterior" required autocomplete="off">
125</div>
126<div class="form-floating mb-3">
127<input type="text" class="form-control form-control-sm" id="current_hair" name="current_hair" value="{user.current_hair}" placeholder="Cabello actual" required autocomplete="off">
128</div>
129<div class="form-floating mb-3">
130<input type="text" class="form-control form-control-sm" id="previous_hair" name="previous_hair" value="{user.previous_hair}" placeholder="Cabello anterior" required autocomplete="off">
131</div>
132<div class="form-floating mb-3">
133<input type="text" class="form-control form-control-sm" id="current_skin" name="current_skin" value="{user.current_skin}" placeholder="Piel actual" required autocomplete="off">
134</div>
135<div class="form-floating mb-3">
136<input type="text" class="form-control form-control-sm" id="previous_skin" name="previous_skin" value="{user.previous_skin}" placeholder="Piel anterior" required autocomplete="off">
137</div>
138<div class="form-floating mb-3">
139<input type="text" class="form-control form-control-sm" id="current_nails" name="current_nails" value="{user.current_nails}" placeholder="Uñas actuales" required autocomplete="off">
140</div>
141<div class="form-floating mb-3">
142<input type="text" class="form-control form-control-sm" id="previous_nails" name="previous_nails" value="{user.previous_nails}" placeholder="Uñas anteriores" required autocomplete="off">
143</div>
144<div class="form-floating mb-3">
145<input type="text" class="form-control form-control-sm" id="current_tattoos" name="current_tattoos" value="{user.current_tattoos}" placeholder="Tatuajes actuales" required autocomplete="off">
146</div>
147<div class="form-floating mb-3">
148<input type="text" class="form-control form-control-sm" id="previous_tattoos" name="previous_tattoos" value="{user.previous_tattoos}" placeholder="Tatuajes anteriores" required autocomplete="off">
149</div>
150<div class="form-floating mb-3">
151<input type="text" class="form-control form-control-sm" id="current_scars" name="current_scars" value="{user.current_scars}" placeholder="Cicatrices actuales" required autocomplete="off">
152</div>
153<div class="form-floating mb-3">
154<input type="text" class="form-control form-control-sm" id="previous_scars" name="previous_scars" value="{user.previous_scars}" placeholder="Cicatrices anteriores" required autocomplete="off">
155</div>
156<div class="form-floating mb-3">
157<input type="text" class="form-control form-control-sm" id="current_hair_color" name="current_hair_color" value="{user.current_hair_color}" placeholder="Color de cabello actual" required autocomplete="off">
158</div>
159<div class="form-floating mb-3">
160<input type="text" class="form-control form-control-sm" id="previous_hair_color" name="previous_hair_color" value="{user.previous_hair_color}" placeholder="Color de cabello anterior" required autocomplete="off">
161</div>
162<div class="form-floating mb-3">
163<input type="text" class="form-control form-control-sm" id="current_hair_length" name="current_hair_length" value="{user.current_hair_length}" placeholder="Longitud de cabello actual" required autocomplete="off">
164</div>
165<div class="form-floating mb-3">
166<input type="text" class="form-control form-control-sm" id="previous_hair_length" name="previous_hair_length" value="{user.previous_hair_length}" placeholder="Longitud de cabello anterior" required autocomplete="off">
167</div>
168<div class="form-floating mb-3">
169<input type="text" class="form-control form-control-sm" id="current_hair_style" name="current_hair_style" value="{user.current_hair_style}" placeholder="Estilo de cabello actual" required autocomplete="off">
170</div>
171<div class

←

→

🔄

📄

localhost:8080/...

🔍

🏠

➤

☰

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 (Kg):
20.4

SERVICIO A

SERVICIO B

SERVICIO C

AGREGAR

QUITAR

Página web html y css

13. Bootstrap

Utilización de
Bootstrap

```
1 <!-- BOOTSTRAP v5.2.3 CDN -->
2 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-rbsA2VBKQhggwzxH7pPCaAq046Mgn0M802W1RWuH610GLWJ3EdK2Kadq2F9CU665" crossorigin="anonymous">
4 <!-- FONTAWESOME CDN v5.2.0 -->
5 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.2.0/css/all.min.css" integrity="sha512-6c4nX2t5Kb0z9Ywpa06kt+cz23BrE1R86fmpcsoN+b/w/euIMuQKNyUoU/nToKN3a8SgN0tPrbW12fug==">
7 <!-- CUSTOM CSS -->
8 <link rel="stylesheet" href="${pageContext.request.contextPath}/resources/css/styles.css">
10 <!-- CUSTOM BOOTSWATCH CSS -->
11 <!--<link rel="stylesheet" href="${pageContext.request.contextPath}/resources/css/bootstrap.min.yeti.css"> -->
12 <link rel="stylesheet" href="${pageContext.request.contextPath}/resources/css/bootstrap.min.lux.css">
```

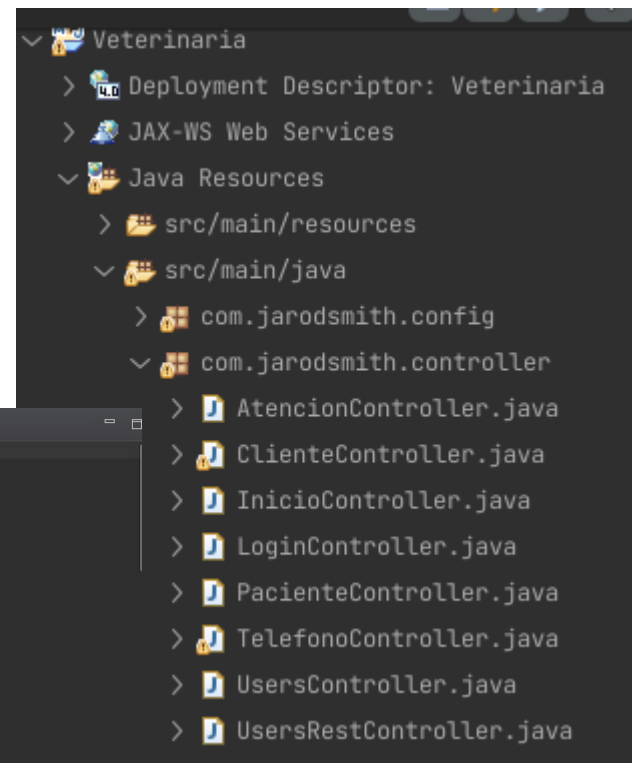
```
1 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2 <%@ taglib prefix="security" uri="http://www.springframework.org/security/tags" %>
3
4 <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
5   <div class="container-fluid">
6     <a class="navbar-brand" href="${pageContext.request.contextPath}/"><i class="fas fa-briefcase-medical"></i> Smokey
7     <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" ar
8       <span class="navbar-toggler-icon"></span>
9   </button>
10   <div class="collapse navbar-collapse" id="navbarSupportedContent">
11     <ul class="navbar-nav me-auto mb-2 mb-lg-0">
12       <li class="nav-item">
13         <a class="nav-link active" aria-current="page" href="${pageContext.request.contextPath}/">Inicio</a>
14       </li>
15
16       <li class="nav-item dropdown">
17         <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">
18           Atención
19         </a>
20         <ul class="dropdown-menu">
21           <li><a class="dropdown-item" href="${pageContext.request.contextPath}/nuevaAtencion">Nueva Atención</a></li>
22           <li><a class="dropdown-item" href="#">Another action</a></li>
23           <li><hr class="dropdown-divider"></li>
24           <li><a class="dropdown-item" href="#">Something else here</a></li>
25         </ul>
26       </li>
27
28       <li class="nav-item dropdown">
29         <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">
30           Clientes
31         </a>
32         <ul class="dropdown-menu">
33           <li><a class="dropdown-item" href="${pageContext.request.contextPath}/cliente/nuevoCliente">Nuevo Cliente</a>
34           <li><a class="dropdown-item" href="${pageContext.request.contextPath}/cliente/listarClientes">Listar Clientes</a>
35           <li><hr class="dropdown-divider"></li>
36           <li><a class="dropdown-item" href="#">Something else here</a></li>
37         </ul>
38       </li>
39     </ul>
40   </div>
41 </div>
```

Spring MVC

14. Controllers

Utilización de
Controllers

```
UsersController.java X
1 package com.jarodsmith.controller;
2
3 import java.util.ArrayList;
4
5
6
7
8
9
10 @Controller
11 @RequestMapping("/usuarios")
12 public class UsersController {
13
14
15
16     @Autowired
17     private UserDaoImpl userDao;
18
19
20     @Autowired
21     private AuthoritiesDaoImpl authoritiesDao;
22
23
24     @Autowired
25     private JdbcTemplate jdbcTemplate;
26
27
28     /**
29      * Devuelve una vista con la lista de usuarios obtenidos desde la base de datos.
30      * @return ModelAndView objeto que representa la vista de la lista de usuarios
31      */
32     @GetMapping("/listarUsuarios")
33     public ModelAndView listarUsuarios() {
34         List<Users> users = userDao.obtenerTodos();
35
36         final String GETONEWITHROLES = "SELECT users.username, COUNT(authorities.authority) AS num_roles FROM users
37             + \" INNER JOIN authorities ON users.username = authorities.username\"
38             + \" GROUP BY users.username\"
39             + \" ORDER BY users.username\";
40
41         List<Map<String, Object>> resumenUsuarios = jdbcTemplate.queryForList(GETONEWITHROLES);
42
43         ModelAndView mav = new ModelAndView();
44         mav.addObject(\"datos\", users);
45         mav.addObject(\"resumen\", resumenUsuarios);
46         mav.setViewName(\"users/listUsersForm\");
47         return mav;
48     }
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
```



Spring MVC

15. Vistas JSP y Taglib

Utilización de vistas
JSP y Taglib

```
addUserForm.jsp
1 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
3 <%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
4
5 <!DOCTYPE html>
6 <html>
7   <head>
8     <meta charset="UTF-8">
9     <!-- INCRUSTA HEAD -->
10    <jsp:include page="../../partials/head.jsp" />
11
12    <title>Crear Usuario</title>
13  </head>
14  <body>
15    <header>
16      <!-- INCRUSTA header -->
17      <jsp:include page="../../partials/navbar.jsp" />
18    </header>
19
20    <main class="container">
21
22      <h2 class="text-center my-5">Crear Usuario</h2>
23
24      <hr>
25
26      <c:if test="${not empty error}">
27        <div class="alert alert-danger alert-dismissible fade show" role="alert" id="alerta">
28          <strong><i class="fas fa-exclamation-circle"></i></strong> ${error}.
29          <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
30        </div>
31      </c:if>
32
33      <a href="${pageContext.request.contextPath}/usuarios/listarUsuarios" class="btn btn-primary btn-sm my-2"><
34
35      <hr>
36
37      <form:form action="${pageContext.request.contextPath}/usuarios/crearUsuario" modelAttribute="userForm" met
38      <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}" />
39
40      <div class="form-floating mb-3">
41        <input type="text" class="form-control form-control-sm" id="username" name="username" value="${user
```


Spring MVC

16. Servicio Spring

Creación Servicio Spring

```
22
23 /**
24  * Clase que configura la aplicación para su funcionamiento con Spring MVC.
25  */
26 @Configuration
27 @EnableWebMvc
28 @ComponentScan(basePackages = "com.jarodsmith")
29 @PropertySource("classpath:persistencia-mysql.properties")
30 public class WebMvcConfig implements WebMvcConfigurer{
31
32     @Autowired
33     private Environment env;
34
35     //--SISTEMA DE LOG PARA REVISIONES --
36     private Logger miLogger = Logger.getLogger(getClass().getName());
37
38
39     /**
40     * Configura el ViewResolver que se encarga de buscar las vistas para el controlador.
41     * @return Un objeto ViewResolver configurado para buscar las vistas en /WEB-INF/view/ con extensión .jsp.
42     */
43     @Bean
44     public ViewResolver viewResolver() {
45         InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
46         viewResolver.setPrefix("/WEB-INF/view/");
47         viewResolver.setSuffix(".jsp");
48         return viewResolver;
49     }
50
51     /**
52     * Configura los ResourceHandlers que se encargan de buscar los recursos estáticos (como CSS, JS, imágenes, etc.).
53     * @param registry El registro de los ResourceHandlers.
54     */
55     @Override
56     public void addResourceHandlers(ResourceHandlerRegistry registry) {
57         registry.addResourceHandler("/resources/**").addResourceLocations("/resources/");
58     }
59
60     //DEFINIR UN BEAN PARA SEGURIDAD
61     @Bean
62     public DataSource seguridadDataSource() {
63
64     }
65 }
```

Spring MVC

17. DAO

Creación DAO acceso a datos

```
1 package com.jarodsmith.dao;
2
3 import java.util.List;
4
5 public interface GenericDAO <T> {
6
7     public T buscarPorId(int id);
8     public T buscarPorNombre(String nombre);
9     public List<T> obtenerTodos();
10    public void insertar(T objeto);
11    public void actualizar(T objeto);
12    public void borrar(int id);
13 }
14
```

```
14 @Repository
15 public class ClienteDAOImpl implements GenericDAO<Cliente> {
16
17     @Autowired
18     private JdbcTemplate jdbcTemplate;
19
20     final String GETONE = "SELECT * FROM cliente WHERE idCliente = ?";
21     final String GETONEFORNAME = "SELECT * FROM cliente WHERE nombre = ?";
22     final String GETALL = "SELECT * FROM cliente order by apellido";
23     final String INSERT = "INSERT INTO cliente(rutCliente, nombre, apellido, telefono, direccion, email) VALUES (?, ?, ?, ?, ?, ?)";
24     final String UPDATE = "UPDATE cliente SET rutCliente = ?, nombre = ?, apellido = ?, telefono = ?, direccion = ?, email = ? WHERE idCliente = ?";
25     final String DELETE = "DELETE FROM cliente WHERE idCliente = ?";
26
27     @Override
28     public Cliente buscarPorId(int id) {
29         Cliente cliente = jdbcTemplate.queryForObject(GETONE, new Object[]{id}, new ClienteRowMapper());
30         return cliente;
31     }
32
33     @Override
34     public Cliente buscarPorNombre(String nombre) {
35         Cliente cliente = jdbcTemplate.queryForObject(GETONEFORNAME, new Object[]{nombre}, new ClienteRowMapper());
36         return cliente;
37     }
38
39     @Override
40     public List<Cliente> obtenerTodos() {
41         List<Cliente> clienteList = jdbcTemplate.query(GETALL, new ClienteRowMapper());
42         return clienteList;
43     }
44
45     @Override
46     public void insertar(Cliente objeto) {
47         Object[]params = {objeto.getRut(), objeto.getNombre(), objeto.getApellido(), objeto.getTelefono(), objeto.getDireccion(), objeto.getEmail()};
48         jdbcTemplate.update(INSERT, params);
49     }
50
51     @Override
52     public void actualizar(Cliente objeto) {
53         Object[]params = {objeto.getRut(), objeto.getNombre(), objeto.getApellido(), objeto.getTelefono(), objeto.getDireccion(), objeto.getEmail(), objeto.getIdCliente()};
54         jdbcTemplate.update(UPDATE, params);
55     }
56 }
```


Spring MVC

18. Creación del proyecto y configuración

```
1 package com.jarodsmith.config;
2
3 import java.beans.PropertyVetoException;
4
5
6 /**
7  * Clase que configura la aplicación para su funcionamiento con Spring MVC.
8  */
9 @Configuration
10 @EnableWebMvc
11 @ComponentScan(basePackages = "com.jarodsmith")
12 @PropertySource("classpath:persistencia-mysql.properties")
13 public class WebMvcConfig implements WebMvcConfigurer {
14
15     @Autowired
16     private Environment env;
17
18     // -- SISTEMA DE LOG PARA REVISIONES --
19     private Logger miLogger = Logger.getLogger(getClass().getName());
20
21     /**
22      * Configura el ViewResolver que se encarga de buscar las vistas para el controlador.
23      * @return Un objeto ViewResolver configurado para buscar las vistas en /WEB-INF/view/ con extensión .jsp.
24      */
25     @Bean
26     public ViewResolver viewResolver() {
27         InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
28         viewResolver.setPrefix("/WEB-INF/view/");
29         viewResolver.setSuffix(".jsp");
30         return viewResolver;
31     }
32
33     /**
34      * Configura los ResourceHandlers que se encargan de buscar los recursos estáticos (como CSS, JS, imágenes, etc.).
35      * @param registry El registro de los ResourceHandlers.
36      */
37     @Override
38     public void addResourceHandlers(ResourceHandlerRegistry registry) {
39         registry.addResourceHandler("/resources/**").addResourceLocations("/resources/");
40     }
41 }
```

Spring MVC

19. Funcionamiento del aplicativo

 SMOKEY

INICIO

ATENCIÓN ▾

CLIENTES ▾



PACIENTES ▾

GESTIÓN USUARIOS ▾

MENÚ USUA




LISTA DE CLIENTES

CREAR CLIENTE

| NOMBRE | APELLIDO | TELÉFONO | EMAIL | ACCIONES |
|----------------------|------------------------|----------|---------------------------|---|
| nombreClienteEjemplo | apellidoClienteEjemplo | 123456 | email ejemplo@ejemplo.com |   |

VETERINARIA
Veterinaria creada con Spring MVC y Bootstrap.

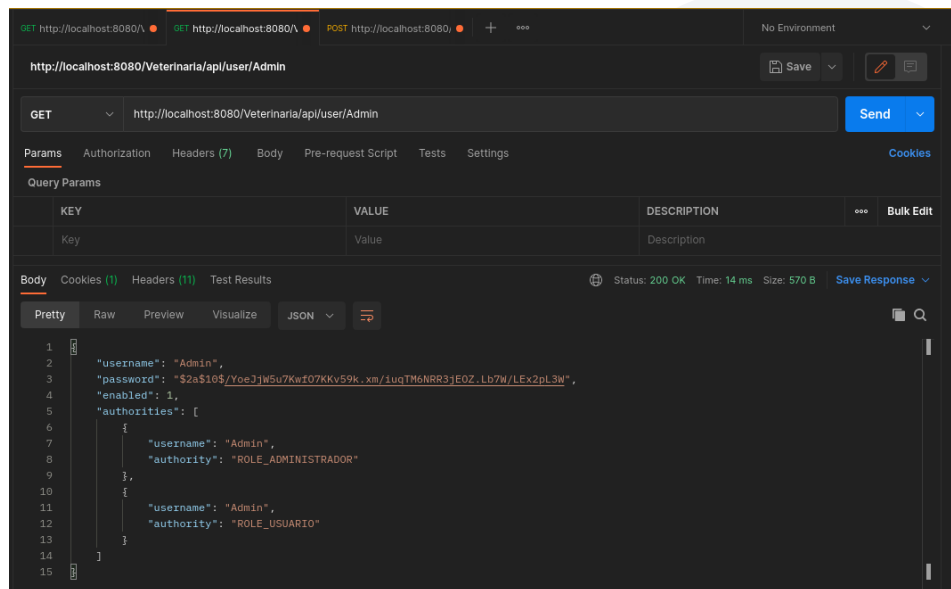
LINKS
[Inicio](#)
[Atención](#)
[Clientes](#)
[Pacientes](#)

CONTACTO
 [Calle Falsa 123 - Quillota](#)
 [+56934567890](#)
 [Github](#)

© 2023 Veterinaria

API REST

20. Creación Servicio REST



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

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

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

| KEY | VALUE | DESCRIPTION |
|-----|-------|-------------|
| Key | Value | Description |

Body Cookies (1) Headers (1) Test Results

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

Pretty Raw Preview Visualize JSON

```
1 {
2   "username": "Admin",
3   "password": "$2a$10$/Yoe3jW5u7KwF07KKv59k.xm/1uqTM6NRR3jEOZ.Lb7W/LEx2pL3W",
4   "enabled": 1,
5   "authorities": [
6     {
7       "username": "Admin",
8       "authority": "ROLE_ADMINISTRADOR"
9     },
10    {
11      "username": "Admin",
12      "authority": "ROLE_USUARIO"
13    }
14  ]
15 }
```

```
1 *UsersRestController.java X
29 public class UsersRestController {
30
31   @Autowired
32   private UserDAOImpl userDAO;
33
34   @Autowired
35   private AuthoritiesDAOImpl authoritiesDAO;
36
37   public ResponseEntity<Users> obtenerSaludo() {}
38
39
40   @PostMapping("/addUser")
41   public ResponseEntity<Users> crearUsuario(@RequestBody Users usuario) {
42
43     // GUARDAR USUARIO EN LA BD
44     userDAO.insertar(usuario);
45
46     return ResponseEntity
47       .status(HttpStatus.CREATED)
48       .body(usuario);
49   }
50
51   @GetMapping("/user/{username}")
52   public ResponseEntity<Users> obtenerUsuarioPorId(@PathVariable String username) {
53     //CREO UN OBJETO TIPO USERS CON LOS DATOS RECUPERADOS DEL DAO
54     Users user = userDAO.buscarPorNombre(username);
55
56     //CREO UNA LISTA CON LOS ROLES RECUPERADOS DEL DAO
57     List<Authorities> listaRoles = authoritiesDAO.buscarRolesPorUsername(username);
58
59     //SETO LA LISTA DE ROLES AL OBJETO
60     user.setAuthorities(listaRoles);
61
62     return ResponseEntity
63       .status(HttpStatus.OK) //CODIGO 200
64       .body(user);
65   }
66 }
```

EDITAR USUARIO

LISTAR USUARIOS

Nombre de Usuario:
Admin

Contraseña:
\$2a\$10\$/YoejW5u7KwF07KKv59k.xm/1uqTM6NRR3jEOZ.Lb7W

Repita su contraseña:
\$2a\$10\$/YoejW5u7KwF07KKv59k.xm/1uqTM6NRR3jEOZ.Lb7W

☒ Habilitado

ROLES DISPONIBLES

ROLE_USUARIO
ROLE_ADMINISTRADOR

ROLES ASIGNADOS

ROLE_ADMINISTRADOR
ROLE_USUARIO