

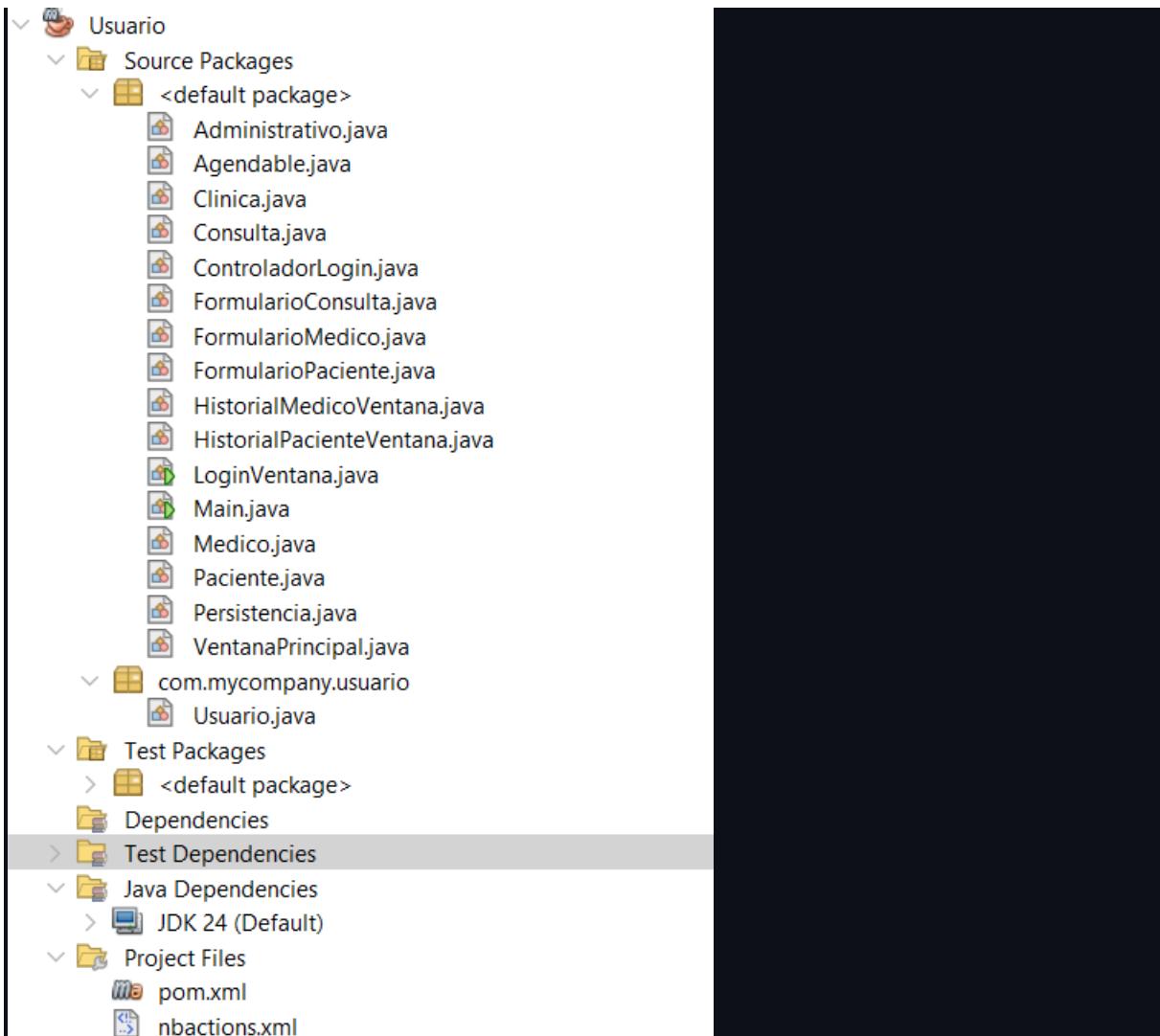
## **Documento de Evidencias: Parcial 2**

**Nombre completo:** Daniel Santiago Camargo Fiagá

**Materia:** Programación Orientada a Objetos

1	Login de usuario médico o administrativo.	✓
2	Registrar nuevos pacientes y médicos.	✓
3	Asignar una consulta a un paciente con un médico y registrar síntomas, diagnóstico y tratamiento.	✓
4	Consultar el historial médico de un paciente.	✓
5	Listar todas las consultas realizadas por un médico.	✓

1. Estructura del Proyecto



El proyecto está organizado en múltiples clases dentro de paquetes. Cada clase tiene una responsabilidad clara:

Paciente, Medico y Administrativo heredan de Usuario.

Clinica centraliza la lógica de manejo de datos.

Persistencia se encarga del guardado/cargado en archivos .txt.

FormularioPaciente, FormularioMedico, FormularioConsulta representan las interfaces gráficas para ingresar datos.

## 2. 2. Clases Base

```

4  /*
5   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
6   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
7   */
8  package com.mycompany.usuario;
9
10 public class Paciente extends Usuario {
11     private String id;
12     private int edad;
13
14     public Paciente(String id, String nombre, String contrasenia, int edad) {
15         super(usuario, contrasenia, nombre); // llamado al constructor de Usuario
16         this.id = id;
17         this.edad = edad;
18     }
19
20     public String getId() { return id; }
21
22     public int getEdad() { return edad; }
23
24     @Override
25     public String getTipo() {
26         return "Paciente";
27     }
28
29     @Override
30     public String getNombreCompleto() {
31         return nombreCompleto;
32     }
33
34 }

```

Define los atributos y métodos específicos para un paciente. Incluye id, nombre, usuario, contraseña y edad. Hereda de Usuario.

```

1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   */
4
5  package com.mycompany.usuario;
6
7  /**
8   * 
9   * @author Daniel Camargo
10  */
11 public abstract class Usuario {
12     protected String usuario;
13     protected String contrasenia;
14     protected String nombreCompleto;
15
16     public Usuario(String usuario, String contrasenia, String nombreCompleto) {
17         this.usuario = usuario;
18         this.contrasenia = contrasenia;
19         this.nombreCompleto = nombreCompleto;
20     }
21
22     public String getUsuario() {
23         return usuario;
24     }
25
26     public String getContrasenia() {
27         return contrasenia;
28     }
29
30     public boolean verificarContraseña(String input) {
31         return contrasenia.equals(input);
32     }
33
34     public String getNombreCompleto() {
35         return nombreCompleto;
36     }
37     public abstract String getTipo();
38 }

```

Esta es una clase abstracta base para pacientes, médicos y administrativos. Contiene usuario, contraseña, nombre completo con sus respectivos getters, método verificarContraseña() para validar los datos que el usuario digita y un método getTipo() para conseguir el tipo de dato que se ingresa. Con ello, esta información será utilizada más adelante.

```
1 import com.mycompany.usuario.Usuario;
2 import java.util.ArrayList;
3 import java.util.List;
4
5 /*
6  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
7  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
8 */
9
10 /**
11  *
12  * @author Daniel Camargo
13  */
14
15
16 public class Medico extends Usuario {
17     private String id;
18     private String nombre;
19     private String especialidad;
20     private List<Consulta> consultasAgendadas;
21
22     public Medico(String id, String nombre, String usuario, String contrasenia) {
23         super(usuario, contrasenia, nombre);
24         this.id = id;
25         this.nombre = nombre;
26         this.usuario = usuario;
27         this.contrasenia = contrasenia;
28         this.especialidad = especialidad;
29         this.consultasAgendadas = new ArrayList<>();
30     }
31
32     public String getId() {
33         return id;
34     }
35
36     public String getNombre() {
37         return nombre;
38     }
39
40     public String getUsuario() {
41         return usuario;
42     }
43
44     public String getContrasenia() {
45         return contrasenia;
46     }
47
48     public String getEspecialidad() {
49         return especialidad;
50     }
51
52     public void agendarConsulta(Consulta consulta) {
53         consultasAgendadas.add(consulta);
54     }
55
56     public List<Consulta> obtenerConsultasAgendadas() {
57         return consultasAgendadas;
58     }
59
60     public String getTipo() {
61         return "Medico";
62     }
63 }
```

```
38 }
39
40     public String getUsuario() {
41         return usuario;
42     }
43
44     public String getContrasenia() {
45         return contrasenia;
46     }
47
48     public String getEspecialidad() {
49         return especialidad;
50     }
51
52     public void agendarConsulta(Consulta consulta) {
53         consultasAgendadas.add(consulta);
54     }
55
56     public List<Consulta> obtenerConsultasAgendadas() {
57         return consultasAgendadas;
58     }
59
60     public String getTipo() {
61         return "Medico";
62     }
63 }
```

Define un médico con su especialidad y una lista de consultas agendadas. También hereda de Usuario. Para ello se usa extends y se llaman a todos los getters con toda la información del médico que se registró; esto con la idea de usar dicha información para poder agendar las consultas necesarias.

### 3. 3. Lógica del Sistema

```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  import com.mycompany.usuario.Usuario;
6  import java.io.BufferedReader;
7  import java.io.FileReader;
8  import java.io.FileWriter;
9  import java.io.IOException;
10 import java.io.PrintWriter;
11 import java.util.*;
12 /**
13 *
14 * @author Daniel Camargo
15 */
16 public class Clinica {
17     private List<Paciente> pacientes;
18     private List<Medico> medicos;
19     private List<Consulta> consultas;
20
21     public Clinica() {
22         pacientes = new ArrayList<>();
23         medicos = new ArrayList<>();
24         consultas = new ArrayList<>();
25     }
26
27     // PACIENTES
28     public void agregarPaciente(Paciente paciente) {
29         pacientes.add(paciente);
30     }
31
32     public Paciente buscarPaciente(String usuario) {
33         for (Paciente p : pacientes) {
34             if (p.getUsuario().equals(usuario)) {
35                 return p;
36             }
37         }
38         return null;
39     }
40 }
```

```
41     public List<Paciente> getPacientes() {
42         return pacientes;
43     }
44
45     // MÉDICOS
46     public void agregarMedico(Medico medico) {
47         medicos.add(medico);
48     }
49
50     public Medico buscarMedico(String usuario) {
51         for (Medico m : medicos) {
52             if (m.getUsuario().equals(usuario)) {
53                 return m;
54             }
55         }
56         return null;
57     }
58
59     public List<Medico> getMedicos() {
60         return medicos;
61     }
62
63     // CONSULTAS
64     public void agregarConsulta(Consulta consulta) {
65         consultas.add(consulta);
66         System.out.println("Consulta agregada con éxito: ");
67         System.out.println("- " + consulta.getPaciente().getUsuario());
68         System.out.println("- " + consulta.getMedico().getUsuario());
69         System.out.println("- " + consulta.getSintomas());
70     }
71     public static void guardarConsultas(String ruta, List<Consulta> lista) {
72         try (PrintWriter pw = new PrintWriter(new FileWriter(ruta))) {
73             for (Consulta c : lista) {
74                 pw.println(
75                     c.getId() + ";" +
76                     c.getPaciente().getUsuario() + ";" +
77                     c.getSintomas() + ";" +
78                     c.getDiagnostico() + ";" +
79                     c.getTratamiento()
80                 );
81             }
82         }
83     }
84 }
```

```
82     } catch (IOException e) {
83         System.err.println("Error al guardar consultas: " + e.getMessage());
84     }
85 }
86
87 public List<Consulta> getConsultas() {
88     return consultas;
89 }
90 public List<Usuario> getTodosLosUsuarios() {
91     List<Usuario> lista = new ArrayList<>();
92     lista.addAll(pacientes);
93     lista.addAll(medicos);
94
95     return lista;
96 }
97
98 public List<Consulta> historialPorPaciente(String idPaciente) {
99     List<Consulta> resultado = new ArrayList<>();
100    for (Consulta c : consultas) {
101        if (c.getPaciente().getId().equals(idPaciente)) {
102            resultado.add(c);
103        }
104    }
105
106    return resultado;
107 }
108 public List<Consulta> historialPorMedico(String usuarioMedico) {
109     List<Consulta> resultado = new ArrayList<>();
110     if (consultas != null) {
111         for (Consulta c : consultas) {
112             if (c != null && c.getMedico() != null && c.getMedico().getUsuario().equals(usuarioMedico)) {
113                 resultado.add(c);
114             }
115         }
116     }
117     return resultado;
118 }
119
```

```
111     for (Consulta c : consultas) {
112         if (c != null && c.getMedico() != null && c.getMedico().getUsuario().equals(usuarioMedico)) {
113             resultado.add(c);
114         }
115     }
116
117     return resultado;
118 }
119
120 // CARGA Y GUARDADO
121 public void cargarTodo() {
122     pacientes = Persistencia.cargarPacientes("data/pacientes.txt");
123     medicos = Persistencia.cargarMedicos("data/medicos.txt");
124     consultas = Persistencia.cargarConsultas("data/consultas.txt", pacientes, medicos);
125 }
126
127 public void guardarTodo() {
128     if (consultas == null) {
129         consultas = new ArrayList<>();
130     }
131     Persistencia.guardarPacientes("data/pacientes.txt", pacientes);
132     Persistencia.guardarMedicos("data/medicos.txt", medicos);
133     Persistencia.guardarConsultas("data/consultas.txt", consultas);
134 }
135
136
137 }
```

Es el controlador principal, maneja listas de pacientes, médicos y consultas. Tiene métodos para agregar elementos, buscar usuarios y guardar toda la información con guardarTodo(). Aquí se incluye el agregar y buscar pacientes y médicos, luego con el uso de listas se consiguen esos datos para implementar la opción de agregar y guardas las consultas realizadas, igualmente a través de una lista y por último se recogen todos los pacientes y médicos que hayan sido registrados hasta ese momento. Por consiguiente, se añade la opción de visualizar el historial por médico y por paciente con la información necesaria para revisar todos los reportes que ha hecho el médico y mostrar un diagnóstico certero y un tratamiento especial para el paciente según corresponda.

```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5  import java.io.*;
6  import java.util.*;
7  /**
8   *
9   * @author Daniel Camargo
10  */
11
12
13  public class Persistencia {
14
15      // ----- PACIENTES -----
16
17      public static List<Paciente> cargarPacientes(String ruta) {
18          List<Paciente> lista = new ArrayList<>();
19          File archivo = new File(ruta);
20
21          if (!archivo.exists()) {
22              System.err.println("Archivo de pacientes no encontrado en: " + ruta);
23              return lista;
24          }
25
26          try (BufferedReader br = new BufferedReader(new FileReader(archivo))) {
27              String linea;
28              while ((linea = br.readLine()) != null) {
29                  String[] partes = linea.split(":");
30                  if (partes.length == 4) {
31                      String id = partes[0];
32                      String contrasenia = partes[1];
33                      String nombre = partes[2];
34                      int edad = Integer.parseInt(partes[3]);
35
36                      lista.add(new Paciente(id, contrasenia, nombre, contrasenia, edad));
37                  } else {
38                      System.err.println("Linea con formato incorrecto: " + linea);
39                  }
40              }
41          } catch (IOException | NumberFormatException e) {
42              System.err.println("Error al cargar pacientes: " + e.getMessage());
43          }
44
45          return lista;
46      }
47
48      public static void guardarPacientes(String ruta, List<Paciente> lista) {
49          try (PrintWriter pw = new PrintWriter(new FileWriter(ruta))) {
50              for (Paciente p : lista) {
51                  pw.println(p.getId() + ";" + p.getNombreCompleto() + ";" +
52                             p.getUsuario() + ";" + p.getContrasenia() + ";" + p.getEdad());
53              }
54              System.out.println("✓ Pacientes guardados en: " + ruta);
55          } catch (IOException e) {
56              System.err.println("✗ Error al guardar pacientes: " + e.getMessage());
57          }
58      }
59
60      // ----- MÉDICOS -----
61
62      public static List<Medico> cargarMedicos(String ruta) {
63          List<Medico> lista = new ArrayList<>();
64          try (BufferedReader br = new BufferedReader(new FileReader(ruta))) {
65              String linea;
66              while ((linea = br.readLine()) != null) {
67                  String[] partes = linea.split(":");
68                  if (partes.length == 4) {
69                      String usuario = partes[0];
70                      String clave = partes[1];
71                      String nombre = partes[2];
72                      String especialidad = partes[3];
73                      lista.add(new Medico(usuario, clave, nombre, especialidad));
74                  }
75              }
76          } catch (IOException e) {
77              System.err.println("Error al cargar médicos: " + e.getMessage());
78          }
79          return lista;
80      }
81  }
```

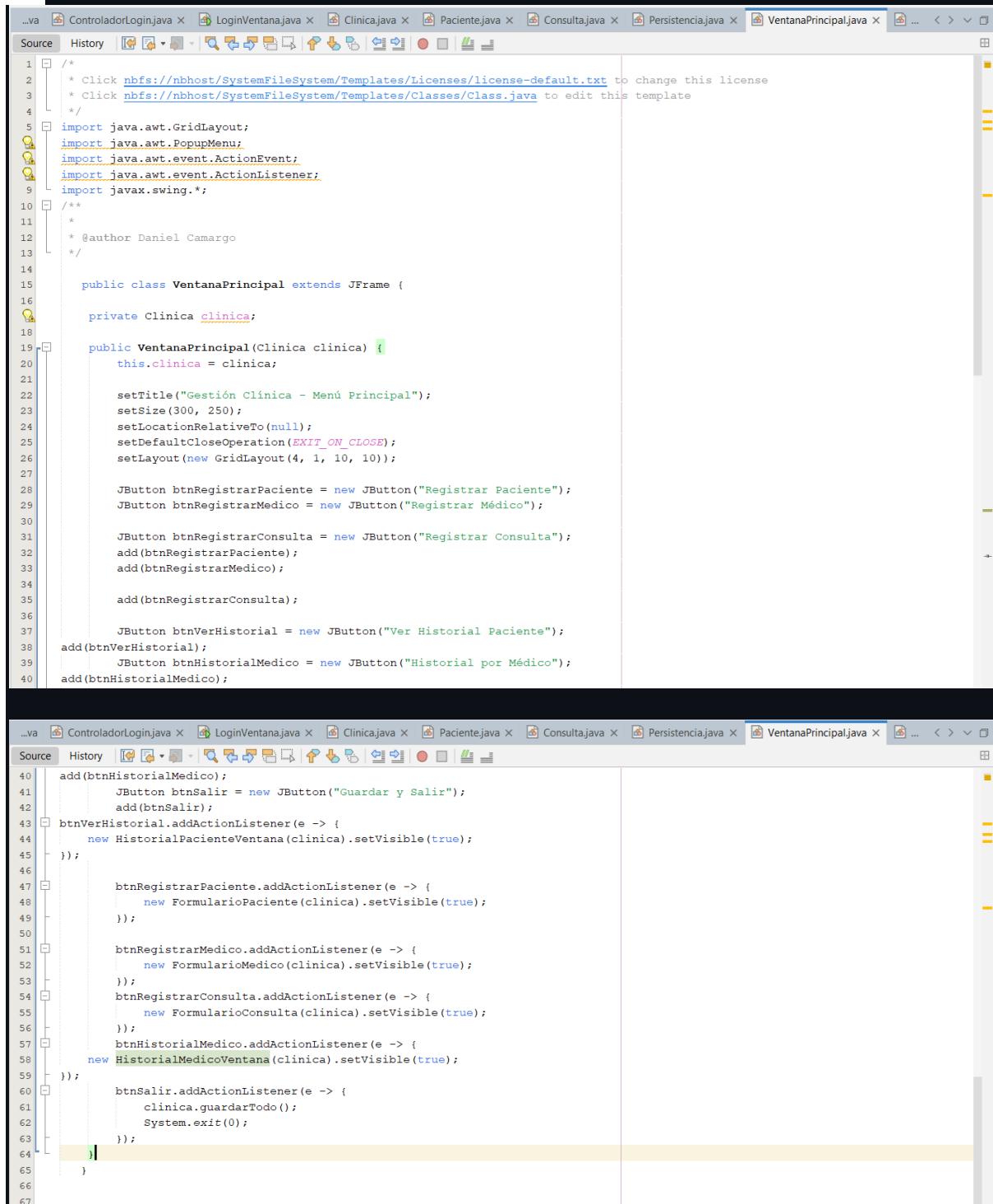
```
81     public static void guardarMedicos(String ruta, List<Medico> lista) {
82         try (PrintWriter pw = new PrintWriter(new FileWriter(ruta))) {
83             for (Medico m : lista) {
84                 if (m != null) { // -> Validar que no sea null
85                     pw.println(m.getId() + ";" + m.getNombre() + ";" + m.getUsuario()
86                               + ";" + m.getContrasenia() + ";" + m.getEspecialidad());
87                 }
88             }
89         } catch (IOException e) {
90             System.err.println("Error al guardar médicos: " + e.getMessage());
91         }
92     }
93
94     // ----- CONSULTAS -----
95
96     public static List<Consulta> cargarConsultas(String ruta, List<Paciente> pacientes, List<Medico> medicos) {
97         List<Consulta> lista = new ArrayList<>();
98         try (BufferedReader br = new BufferedReader(new FileReader(ruta))) {
99             String linea;
100            while ((linea = br.readLine()) != null) {
101                String[] partes = linea.split(";");
102                if (partes.length == 5) {
103                    String idPaciente = partes[0];
104                    String usuarioMedico = partes[1];
105                    String sintomas = partes[2];
106                    String diagnostico = partes[3];
107                    String tratamiento = partes[4];
108
109                    // Buscar paciente y médico en las listas dadas
110                    Paciente paciente = null;
111                    for (Paciente p : pacientes) {
112                        if (p.getId().equals(idPaciente)) {
113                            paciente = p;
114                            break;
115                        }
116                    }
117                }
118            }
119        } catch (IOException e) {
120            System.err.println("Error al cargar consultas: " + e.getMessage());
121        }
122    }
123
124    public static void guardarConsultas(String ruta, List<Consulta> consultas) {
125        try (PrintWriter pw = new PrintWriter(new FileWriter(ruta))) {
126            for (Consulta c : consultas) {
127                pw.println(c.getId() + ";" +
128                           c.getPaciente().getNombre() + ";" +
129                           c.getMedico().getNombre() + ";" +
130                           c.getSintomas() + ";" +
131                           c.getDiagnostico() + ";" +
132                           c.getTratamiento());
133            }
134        } catch (IOException e) {
135            System.err.println("Error al guardar consultas: " + e.getMessage());
136        }
137    }
138
139
140    public static void cargarPacientes(String ruta, List<Paciente> pacientes) {
141        try (BufferedReader br = new BufferedReader(new FileReader(ruta))) {
142            String linea;
143            while ((linea = br.readLine()) != null) {
144                String[] partes = linea.split(";");
145                if (partes.length == 5) {
146                    String idPaciente = partes[0];
147                    String nombre = partes[1];
148                    String apellido = partes[2];
149                    String sexo = partes[3];
150                    String edad = partes[4];
151
152                    Paciente paciente = new Paciente(idPaciente, nombre, apellido, sexo, edad);
153                    pacientes.add(paciente);
154                }
155            }
156        } catch (IOException e) {
157            System.err.println("Error al cargar pacientes: " + e.getMessage());
158        }
159    }
160
161    public static void guardarPacientes(String ruta, List<Paciente> pacientes) {
162        try (PrintWriter pw = new PrintWriter(new FileWriter(ruta))) {
163            for (Paciente p : pacientes) {
164                pw.println(p.getId() + ";" +
165                           p.getNombre() + ";" +
166                           p.getApellido() + ";" +
167                           p.getSexo() + ";" +
168                           p.getEdad());
169            }
170        } catch (IOException e) {
171            System.err.println("Error al guardar pacientes: " + e.getMessage());
172        }
173    }
174
175
176    public static void cargarMedicos(String ruta, List<Medico> medicos) {
177        try (BufferedReader br = new BufferedReader(new FileReader(ruta))) {
178            String linea;
179            while ((linea = br.readLine()) != null) {
180                String[] partes = linea.split(";");
181                if (partes.length == 5) {
182                    String idMedico = partes[0];
183                    String nombre = partes[1];
184                    String apellido = partes[2];
185                    String especialidad = partes[3];
186                    String contrasenia = partes[4];
187
188                    Medico medico = new Medico(idMedico, nombre, apellido, especialidad, contrasenia);
189                    medicos.add(medico);
190                }
191            }
192        } catch (IOException e) {
193            System.err.println("Error al cargar médicos: " + e.getMessage());
194        }
195    }
196
197    public static void guardarMedicos(String ruta, List<Medico> medicos) {
198        try (PrintWriter pw = new PrintWriter(new FileWriter(ruta))) {
199            for (Medico m : medicos) {
200                pw.println(m.getId() + ";" +
201                           m.getNombre() + ";" +
202                           m.getApellido() + ";" +
203                           m.getEspecialidad() + ";" +
204                           m.getContrasenia());
205            }
206        } catch (IOException e) {
207            System.err.println("Error al guardar médicos: " + e.getMessage());
208        }
209    }
210
211
212    public static void cargarConsultas(String ruta, List<Consulta> consultas) {
213        try (BufferedReader br = new BufferedReader(new FileReader(ruta))) {
214            String linea;
215            while ((linea = br.readLine()) != null) {
216                String[] partes = linea.split(";");
217                if (partes.length == 5) {
218                    String idConsulta = partes[0];
219                    String idPaciente = partes[1];
220                    String idMedico = partes[2];
221                    String sintomas = partes[3];
222                    String diagnostico = partes[4];
223
224                    Paciente paciente = null;
225                    Medico medico = null;
226
227                    for (Paciente p : pacientes) {
228                        if (p.getId().equals(idPaciente)) {
229                            paciente = p;
230                            break;
231                        }
232                    }
233
234                    for (Medico m : medicos) {
235                        if (m.getId().equals(idMedico)) {
236                            medico = m;
237                            break;
238                        }
239                    }
240
241                    Consulta consulta = new Consulta(paciente, medico, sintomas, diagnostico, tratamiento);
242                    consultas.add(consulta);
243                }
244            }
245        } catch (IOException e) {
246            System.err.println("Error al cargar consultas: " + e.getMessage());
247        }
248    }
249
250    public static void guardarConsultas(String ruta, List<Consulta> consultas) {
251        try (PrintWriter pw = new PrintWriter(new FileWriter(ruta))) {
252            for (Consulta c : consultas) {
253                pw.println(c.getId() + ";" +
254                           c.getPaciente().getNombre() + ";" +
255                           c.getMedico().getNombre() + ";" +
256                           c.getSintomas() + ";" +
257                           c.getDiagnostico() + ";" +
258                           c.getTratamiento());
259            }
260        } catch (IOException e) {
261            System.err.println("Error al guardar consultas: " + e.getMessage());
262        }
263    }
264
265
266    public static void cargarArchivos() {
267        cargarPacientes("pacientes.txt", pacientes);
268        cargarMedicos("medicos.txt", medicos);
269        cargarConsultas("consultas.txt", consultas);
270    }
271
272    public static void guardarArchivos() {
273        guardarPacientes("pacientes.txt", pacientes);
274        guardarMedicos("medicos.txt", medicos);
275        guardarConsultas("consultas.txt", consultas);
276    }
277
278
279    public static void main(String[] args) {
280        cargarArchivos();
281        guardarArchivos();
282    }
283}
```

```
118
119
120    Medico medico = null;
121    for (Medico m : medicos) {
122        if (m.getUsuario().equals(usuarioMedico)) {
123            medico = m;
124            break;
125        }
126    }
127
128    if (paciente != null && medico != null) {
129        Consulta consulta = new Consulta(paciente, medico, sintomas, diagnostico, tratamiento);
130        lista.add(consulta);
131    }
132
133 } catch (IOException e) {
134     System.err.println("Error al cargar consultas: " + e.getMessage());
135 }
136
137 return lista;
138
139
140 public static void guardarConsultas(String ruta, List<Consulta> consultas) {
141     try (PrintWriter pw = new PrintWriter(new FileWriter(ruta))) {
142         for (Consulta c : consultas) {
143             pw.println(c.getId() + ";" +
144                         c.getPaciente().getNombre() + ";" +
145                         c.getMedico().getNombre() + ";" +
146                         c.getSintomas() + ";" +
147                         c.getDiagnostico() + ";" +
148                         c.getTratamiento());
149         }
150     } catch (IOException e) {
151         System.err.println("Error al guardar consultas: " + e.getMessage());
152     }
153 }
154 }
```

Por su parte, esta clase con métodos estáticos se encarga de guardar y cargar listas de objetos desde archivos .txt. Cada tipo (Paciente, Medico, Consulta) tiene su propio archivo. Allí se implementa el cargarpacientes() y guardarpacientes() que lo que hace es pegar una lista de datos en un archivo de texto verificando que efectivamente ese archivo si vaya a existir. Se

hace exactamente la misma operación con los médicos y las consultas utilizando lectores que verifican que los datos hayan sido digitados de manera correcta y entendible para el sistema.

#### 4. 4. Interfaces Gráficas



```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5  import java.awt.GridLayout;
6  import java.awt.PopupMenu;
7  import java.awt.event.ActionEvent;
8  import java.awt.event.ActionListener;
9  import javax.swing.*;
10 /**
11 *
12 * @author Daniel Camargo
13 */
14
15 public class VentanaPrincipal extends JFrame {
16
17     private Clinica clinica;
18
19     public VentanaPrincipal(Clinica clinica) {
20         this.clinica = clinica;
21
22         setTitle("Gestión Clínica - Menú Principal");
23         setSize(300, 250);
24         setLocationRelativeTo(null);
25         setDefaultCloseOperation(EXIT_ON_CLOSE);
26         setLayout(new GridLayout(4, 1, 10, 10));
27
28         JButton btnRegistrarPaciente = new JButton("Registrar Paciente");
29         JButton btnRegistrarMedico = new JButton("Registrar Médico");
30
31         JButton btnRegistrarConsulta = new JButton("Registrar Consulta");
32         add(btnRegistrarPaciente);
33         add(btnRegistrarMedico);
34
35         add(btnRegistrarConsulta);
36
37         JButton btnVerHistorial = new JButton("Ver Historial Paciente");
38         add(btnVerHistorial);
39         JButton btnHistorialMedico = new JButton("Historial por Médico");
40         add(btnHistorialMedico);
41
42         add(btnHistorialMedico);
43         JButton btnSalir = new JButton("Guardar y Salir");
44         add(btnSalir);
45         btnVerHistorial.addActionListener(e -> {
46             new HistorialPacienteVentana(clinica).setVisible(true);
47         });
48
49         btnRegistrarPaciente.addActionListener(e -> {
50             new FormularioPaciente(clinica).setVisible(true);
51         });
52
53         btnRegistrarMedico.addActionListener(e -> {
54             new FormularioMedico(clinica).setVisible(true);
55         });
56         btnRegistrarConsulta.addActionListener(e -> {
57             new FormularioConsulta(clinica).setVisible(true);
58         });
59         btnHistorialMedico.addActionListener(e -> {
60             new HistorialMedicoVentana(clinica).setVisible(true);
61         });
62         btnSalir.addActionListener(e -> {
63             clinica.guardarTodo();
64             System.exit(0);
65         });
66     }
67 }
```

Es la interfaz principal que muestra los botones necesarios para registrar pacientes, médicos, consultas y salir. Está organizada con GridLayout. Muestra el menú con opciones para registrar pacientes, médicos y consultas con toda la información y datos requeridos.

The screenshot shows the NetBeans IDE interface with the following details:

- Title Bar:** Shows multiple open files: Clinica.java, Paciente.java, Consulta.java, Persistencia.java, VentanaPrincipal.java, FormularioPaciente.java (highlighted in blue), Agendable.java, and Me... .
- Source Tab:** The active tab, displaying the Java code for FormularioPaciente.java.
- Code Content:**

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
/**
 *
 * @author Daniel Camargo
 */
public class FormularioPaciente extends JDialog {
    public FormularioPaciente(Clinica clinica) {
        setTitle("Registrar Paciente");
        setModal(true);
        setSize(400, 300);
        setLocationRelativeTo(null);
        setLayout(new GridLayout(6, 2));

        JTextField txtId = new JTextField();
        JTextField txtNombre = new JTextField();
        JTextField txtUsuario = new JTextField();
        JPasswordField txtContrasenia = new JPasswordField();
        JTextField txtEdad = new JTextField();
        JButton btnGuardar = new JButton("Guardar");

        add(new JLabel("ID:")); add(txtId);
        add(new JLabel("Nombre:")); add(txtNombre);
        add(new JLabel("Usuario:")); add(txtUsuario);
        add(new JLabel("Contrasena:")); add(txtContrasenia);
        add(new JLabel("Edad:")); add(txtEdad);
        add(new JLabel()); add(btnGuardar);

        btnGuardar.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.out.println(">> Se hizo clic en el botón Guardar");
            }
        });
    }
}
```
- Toolbars and Status Bar:** Standard NetBeans toolbars and status bar at the bottom.

```
...va Clinicajava X Pacientejava X Consulta.java X Persistencia.java X VentanaPrincipal.java X FormularioPaciente.java X Agendable.java X Me... < > ...
```

Source History

```
39
40     try {
41         String id = txtId.getText().trim();
42         String nombre = txtNombre.getText().trim();
43         String usuario = txtUsuario.getText().trim();
44         String contrasenia = new String(txtContrasenia.getPassword()).trim();
45         String edadStr = txtEdad.getText().trim();
46
47         if (id.isEmpty() || nombre.isEmpty() || usuario.isEmpty() || contrasenia.isEmpty() || edadStr.isEmpty()) {
48             JOptionPane.showMessageDialog(FormularioPaciente.this, "Todos los campos son obligatorios.");
49             return;
50         }
51
52         int edad = Integer.parseInt(edadStr);
53         Paciente paciente = new Paciente(id, nombre, usuario, contrasenia, edad);
54         clinica.agregarPaciente(paciente);
55         clinica.guardarTodo();
56
57         JOptionPane.showMessageDialog(FormularioPaciente.this, "Paciente registrado con éxito.");
58         dispose();
59
60     } catch (Exception ex) {
61         ex.printStackTrace();
62         JOptionPane.showMessageDialog(FormularioPaciente.this, "Error inesperado: " + ex.getMessage());
63     }
64 }
65 }
66 }
67 }
68 }
```

Consta de un formulario de entrada de datos y cómo se enlaza con la clase Clinica. Permite ingresar datos específicos del tipo de usuario y guarda la información al presionar el botón "Guardar", se verifican todas las operaciones y en caso de presentarse la falta de algún dato específico, se realiza un llamado de atención indicando que es necesario revisar si todos los campos están correctamente diligenciados.

```
...va FormularioConsulta.java X MainJava X HistorialPacienteVentana.java X HistorialMedicoVentana.java X FormularioMedico.java X
Source History 

```

40         add(lblEspecialidad);
41         add(txtEspecialidad);
42         add(new JLabel());
43         add(btnGuardar);

44

45     btnGuardar.addActionListener(e -> {
46         String usuario = txtUsuario.getText().trim();
47         String clave = txtClave.getText().trim();
48         String nombre = txtNombre.getText().trim();
49         String especialidad = txtEspecialidad.getText().trim();
50
51         if (usuario.isEmpty() || clave.isEmpty() || nombre.isEmpty() || especialidad.isEmpty()) {
52             JOptionPane.showMessageDialog(this, "Todos los campos son obligatorios.");
53             return;
54         }
55
56         clinica.agregarMedico(new Medico(usuario, clave, nombre, especialidad));
57         JOptionPane.showMessageDialog(this, "Médico registrado con éxito.");
58         dispose();
59         Medico medico = null;
60
61         clinica.agregarMedico(medico);
62         clinica.guardarTodo();
63     });
64 }
65
66 }
67

```

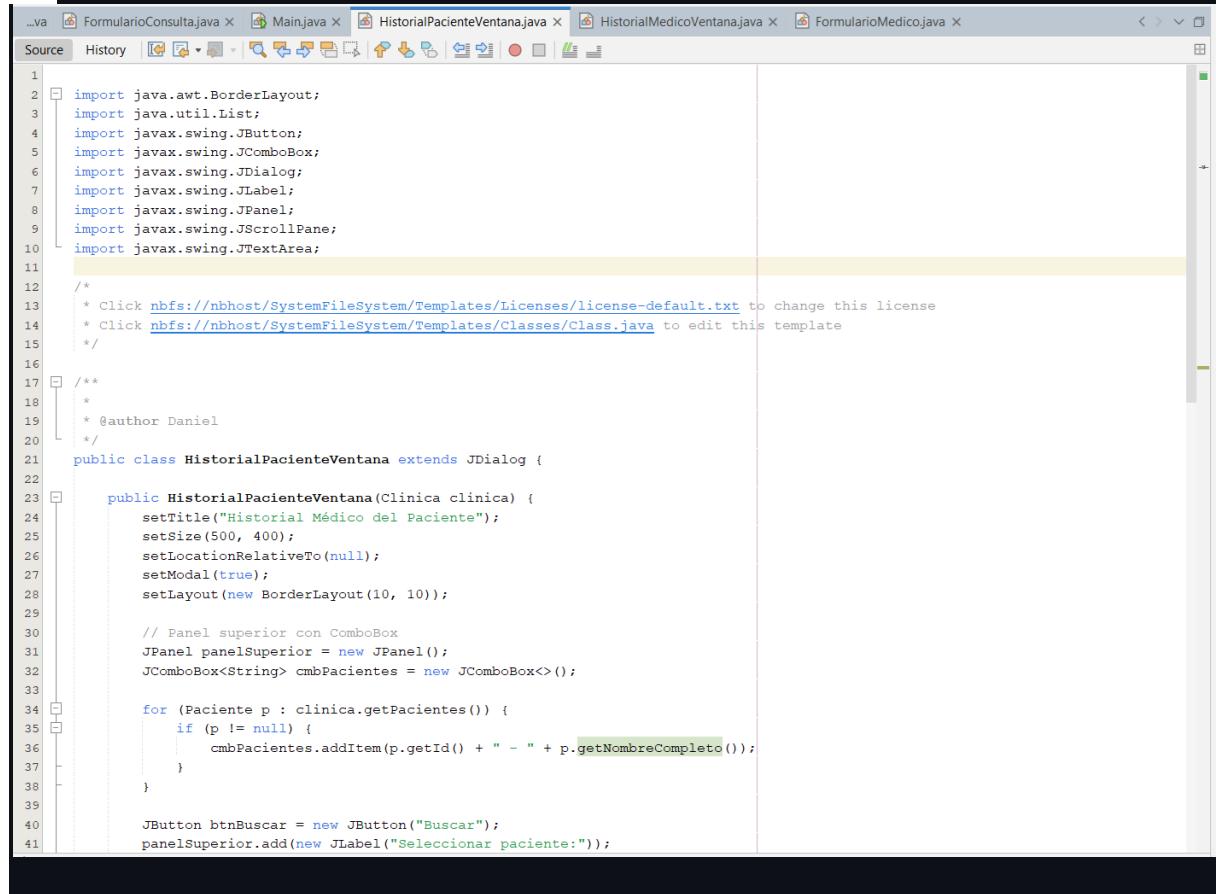

```

Consta de un formulario de entrada de datos y cómo se enlaza con la clase Clinica. Contiene las mismas funciones que el formulario anterior solamente que guarda la información correspondiente a los médicos en este caso.

Consta de un formulario de entrada de datos y cómo se enlaza con la clase Clinica. A diferencia de los anteriores, presenta un cambio, contiene un Combobox con pacientes y médicos, campos de texto, y lógica del botón Guardar. Gracias a esto, Permite seleccionar un

paciente y un médico, ingresar síntomas, diagnóstico y tratamiento, y registrar una nueva consulta.

## 5. 5. Funcionalidades Avanzadas



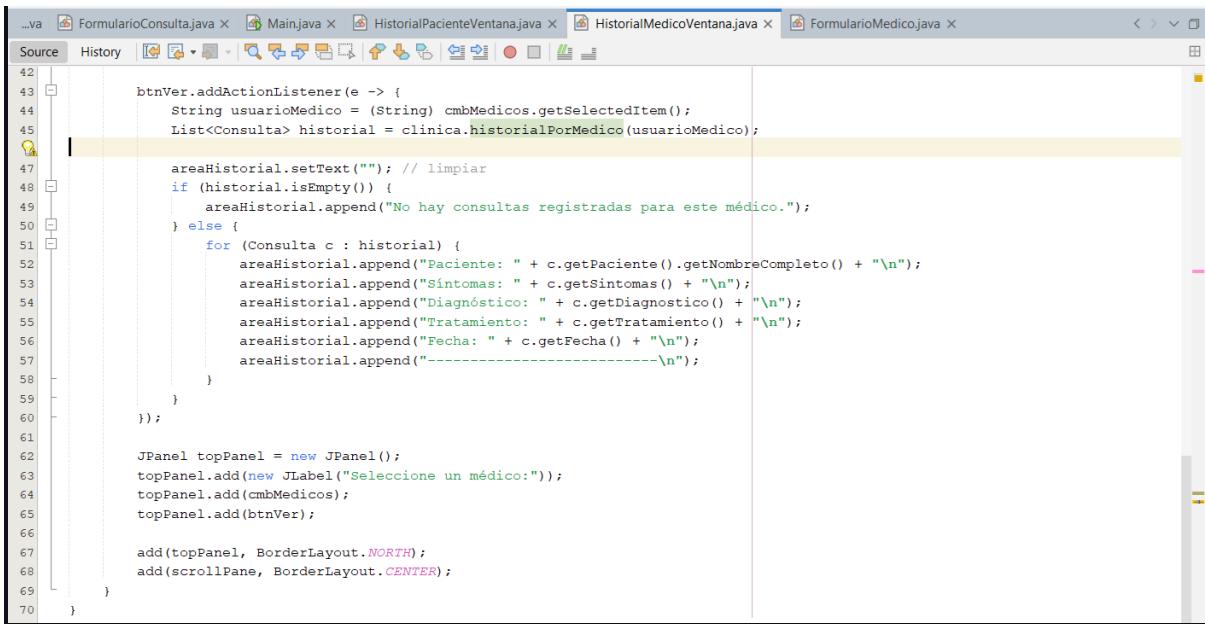
The screenshot shows a Java IDE interface with multiple tabs at the top: ..va, FormularioConsulta.java X, Main.java X, HistorialPacienteVentana.java X, HistorialMedicoVentana.java X, and FormularioMedico.java X. The active tab is 'Source'.

```
1 import java.awt.BorderLayout;
2 import java.util.List;
3 import javax.swing.JButton;
4 import javax.swing.JComboBox;
5 import javax.swing.JDialog;
6 import javax.swing.JLabel;
7 import javax.swing.JPanel;
8 import javax.swing.JScrollPane;
9 import javax.swing.JTextArea;
10
11 /*
12  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
13  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
14  */
15
16 /**
17  * 
18  * @author Daniel
19  */
20 public class HistorialPacienteVentana extends JDialog {
21
22     public HistorialPacienteVentana(Clinica clinica) {
23         setTitle("Historial Médico del Paciente");
24         setSize(500, 400);
25         setLocationRelativeTo(null);
26         setModal(true);
27         setLayout(new BorderLayout(10, 10));
28
29         // Panel superior con ComboBox
30         JPanel panelSuperior = new JPanel();
31         JComboBox<String> cmbPacientes = new JComboBox<>();
32
33         for (Paciente p : clinica.getpacientes()) {
34             if (p != null) {
35                 cmbPacientes.addItem(p.getId() + " - " + p.getNombreCompleto());
36             }
37         }
38
39         JButton btnBuscar = new JButton("Buscar");
40         panelSuperior.add(new JLabel("Seleccionar paciente:"));
41     }
42 }
```

```
40 JButton btnBuscar = new JButton("Buscar");
41 panelSuperior.add(new JLabel("Seleccionar paciente:"));
42 panelSuperior.add(cmbPacientes);
43 panelSuperior.add(btnBuscar);
44
45 // Área para mostrar resultados
46 JTextArea areaResultados = new JTextArea();
47 areaResultados.setEditable(false);
48 JScrollPane scroll = new JScrollPane(areaResultados);
49
50 add(panelSuperior, BorderLayout.NORTH);
51 add(scroll, BorderLayout.CENTER);
52
53 // Acción del botón
54 btnBuscar.addActionListener(e -> {
55     if (cmbPacientes.getItemCount() == 0) {
56         areaResultados.setText("No hay pacientes registrados.");
57         return;
58     }
59
60     String seleccion = (String) cmbPacientes.getSelectedItem();
61     String idSeleccionado = seleccion.split(" - ")[0]; // Extraer el ID antes del guion
62
63     List<Consulta> consultas = clinica.historialPorPaciente(idSeleccionado);
64
65     if (consultas.isEmpty()) {
66         areaResultados.setText("No se encontraron consultas para el paciente seleccionado.");
67     } else {
68         StringBuilder sb = new StringBuilder();
69         for (Consulta c : consultas) {
70             sb.append("Fecha: ").append(c.getFecha()).append("\n")
71             .append("Médico: ").append(c.getMedico().getNombre()).append("\n")
72             .append("Síntomas: ").append(c.getSintomas()).append("\n")
73             .append("Diagnóstico: ").append(c.getDiagnostico()).append("\n")
74             .append("Tratamiento: ").append(c.getTratamiento()).append("\n")
75             .append("-----\n");
76         }
77     }
78     areaResultados.setText(sb.toString());
79 });
80 );
```

Filtró las consultas por ID de paciente. En este caso se implementó un método que recibe el usuario del paciente y retorna una lista con todas las consultas que ha tenido.

```
1 import java.awt.BorderLayout;
2 import java.util.List;
3 import javax.swing.JButton;
4 import javax.swing.JComboBox;
5 import javax.swing.JDialog;
6 import javax.swing.JLabel;
7 import javax.swing.JPanel;
8 import javax.swing.JScrollPane;
9 import javax.swing.JTextArea;
10
11 /*
12  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
13  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
14  */
15
16 /**
17  * 
18  * @author Daniel
19  */
20 public class HistorialMedicoVentana extends JDialog {
21
22     public HistorialMedicoVentana(Clinica clinica) {
23         setTitle("Historial de Consultas por Médico");
24         setModal(true);
25         setSize(500, 400);
26         setLocationRelativeTo(null);
27         setLayout(new BorderLayout());
28
29         JComboBox<String> cmbMedicos = new JComboBox<>();
30         for (Medico m : clinica.getMedicos()) {
31             if (m != null) {
32                 cmbMedicos.addItem(m.getUsuario());
33             }
34         }
35
36         JTextArea areaHistorial = new JTextArea();
37         areaHistorial.setEditable(false);
38         JScrollPane scrollPane = new JScrollPane(areaHistorial);
39
40         JButton btnVer = new JButton("Ver Historial");
41     }
42 }
```

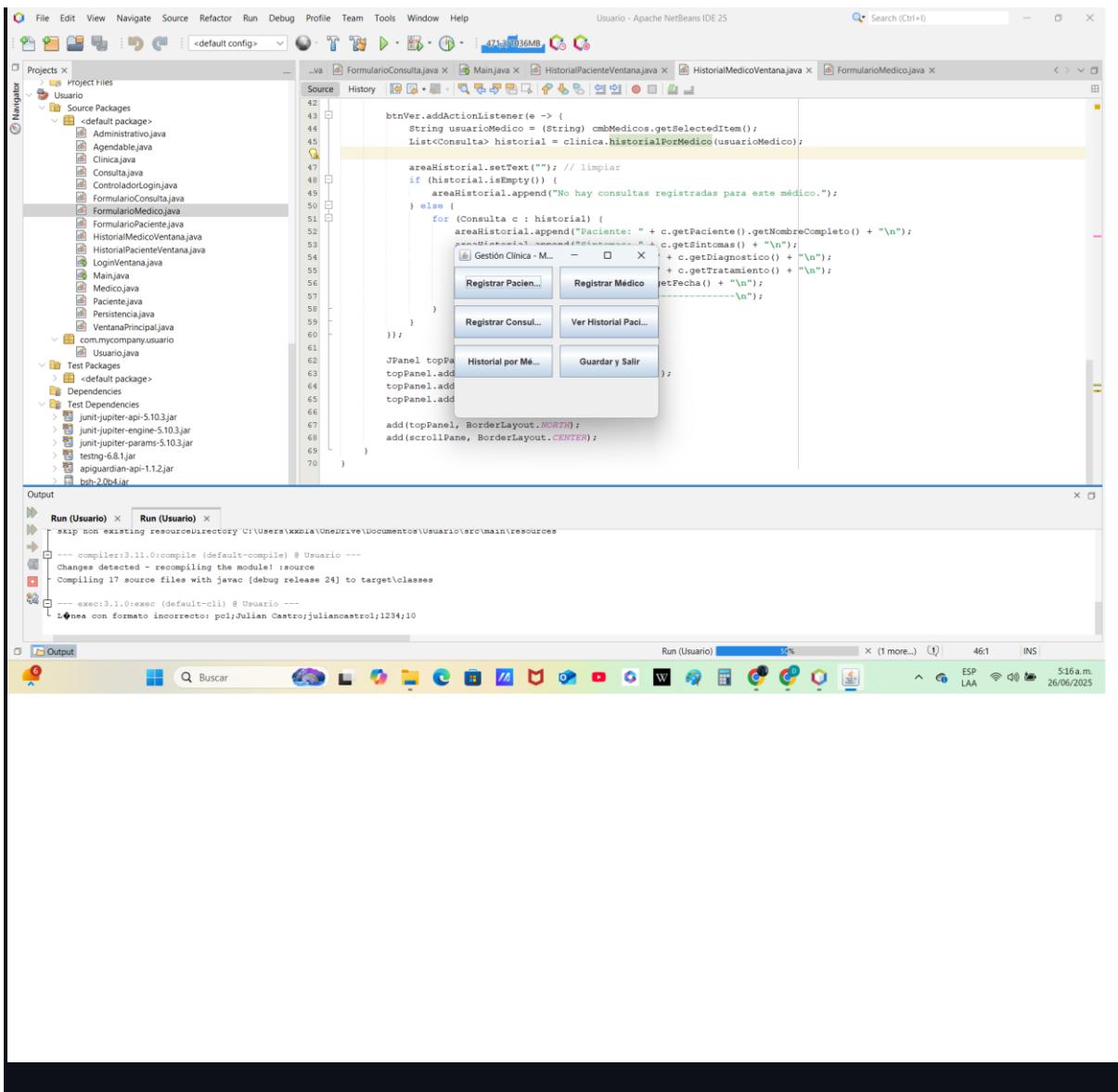


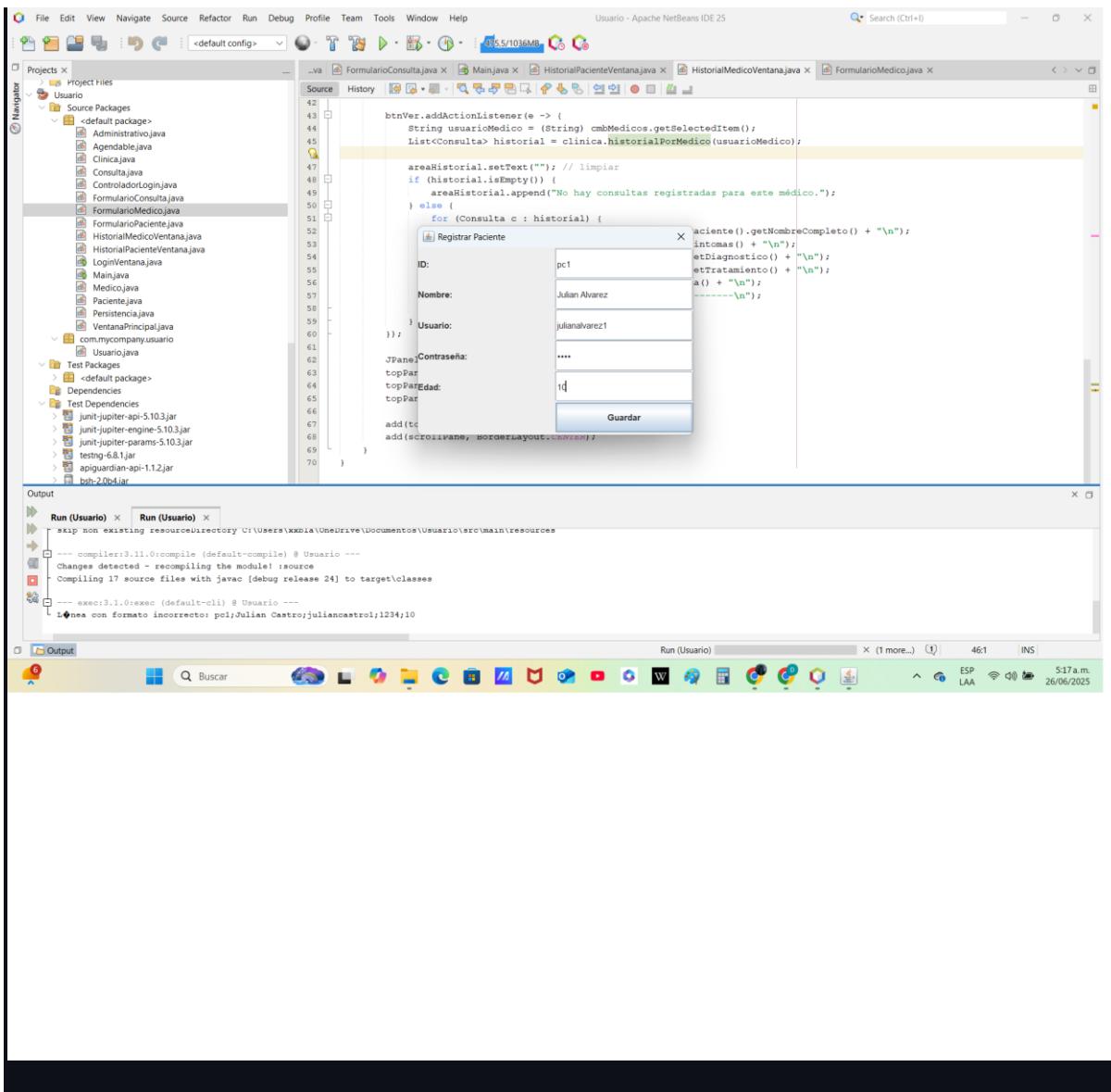
```
42
43     btnVer.addActionListener(e -> {
44         String usuarioMedico = (String) cmbMedicos.getSelectedItem();
45         List<Consulta> historial = clinica.historialPorMedico(usuarioMedico);
46
47         areaHistorial.setText(""); // limpiar
48         if (historial.isEmpty()) {
49             areaHistorial.append("No hay consultas registradas para este médico.");
50         } else {
51             for (Consulta c : historial) {
52                 areaHistorial.append("Paciente: " + c.getpaciente().getNombreCompleto() + "\n");
53                 areaHistorial.append("Síntomas: " + c.getSintomas() + "\n");
54                 areaHistorial.append("Diagnóstico: " + c.getdiagnóstico() + "\n");
55                 areaHistorial.append("Tratamiento: " + c.gettratamiento() + "\n");
56                 areaHistorial.append("Fecha: " + c.getfecha() + "\n");
57                 areaHistorial.append("-----\n");
58             }
59         }
60     });
61
62     JPanel topPanel = new JPanel();
63     topPanel.add(new JLabel("Seleccione un médico:"));
64     topPanel.add(cmbMedicos);
65     topPanel.add(btnVer);
66
67     add(topPanel, BorderLayout.NORTH);
68     add(scrollPane, BorderLayout.CENTER);
69 }
70 }
```

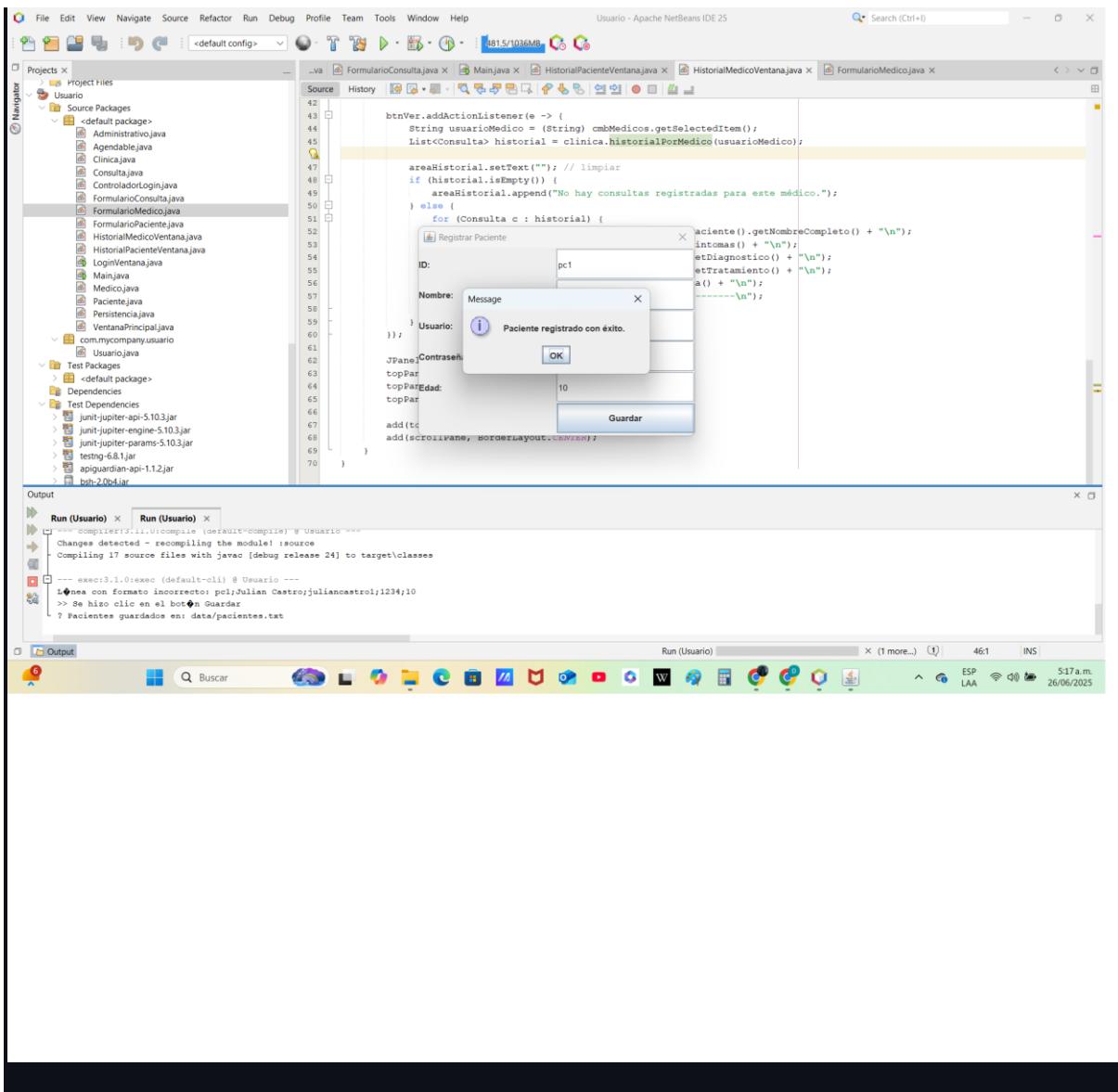
Similar al anterior pero por usuario de médico, filtrando las consultas por el usuario del médico que las atendió.

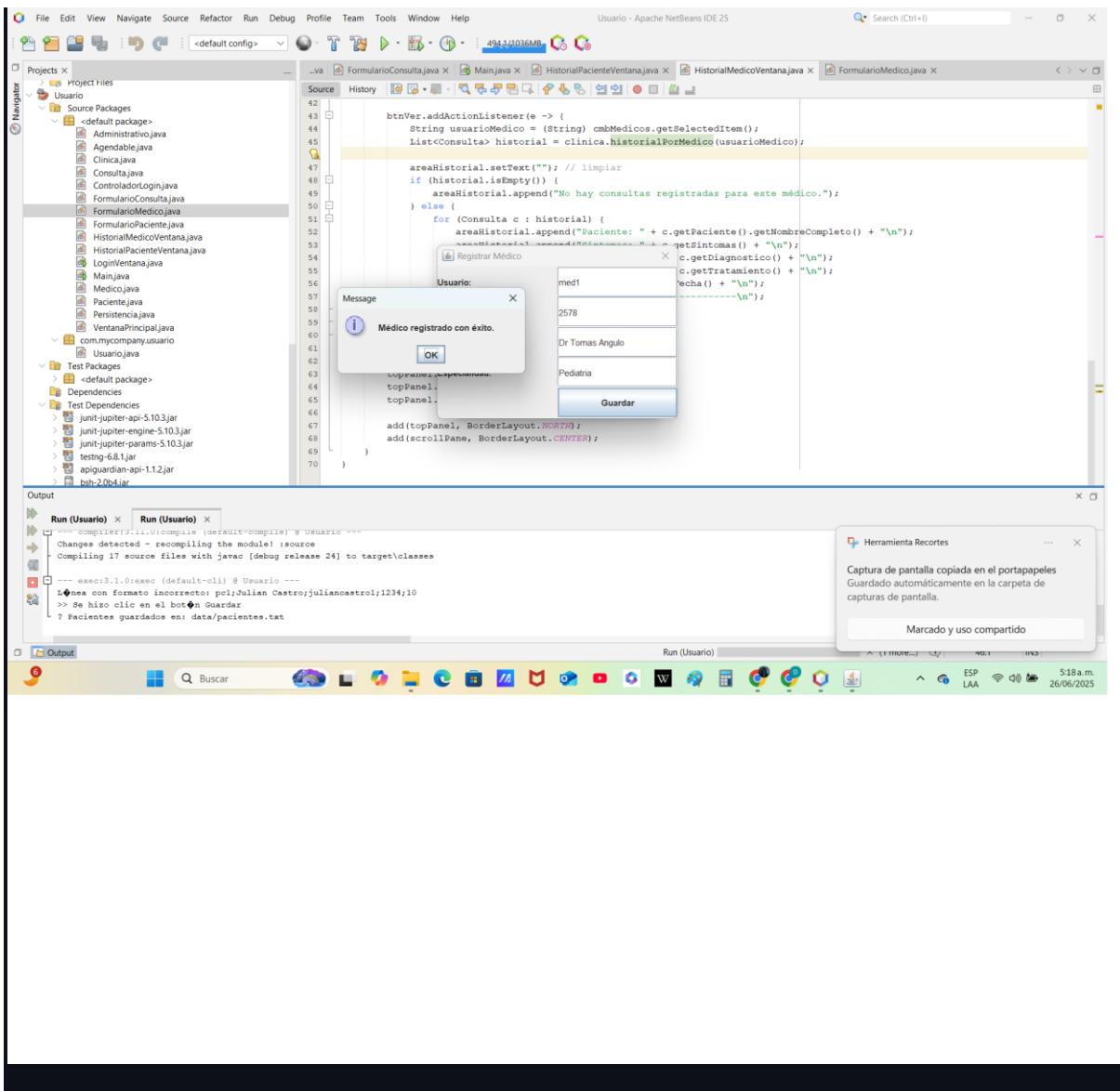
## 6. Pruebas y Resultados

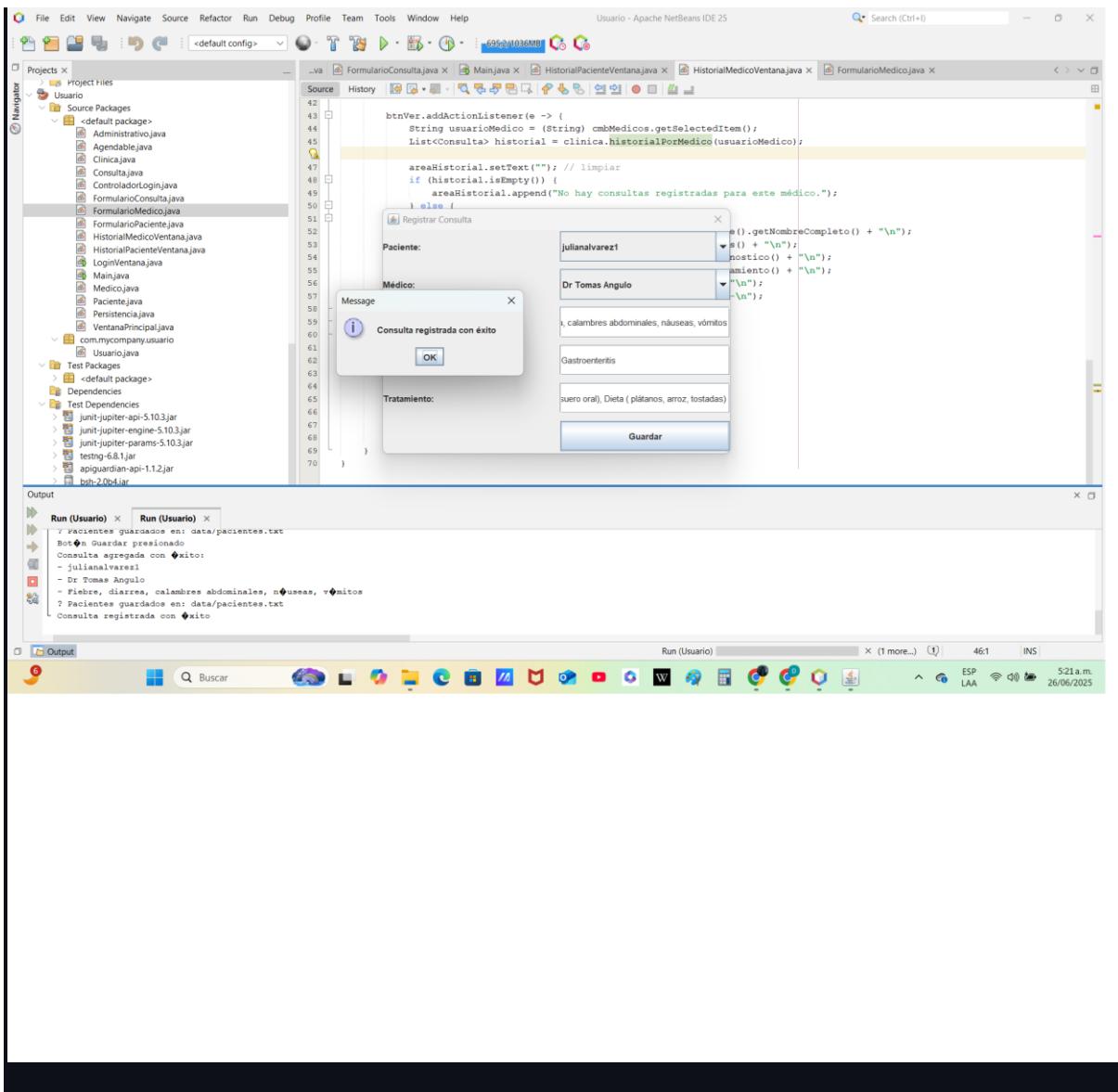
Para probar el sistema, se va a tratar como si fuera una situación real tratando de ser lo más preciso posible de lo que puede llegar a pasar y lo que se tendría que registrar en un caso habitual o común. Incluye logs de consola donde se muestra que los datos se guardan correctamente.

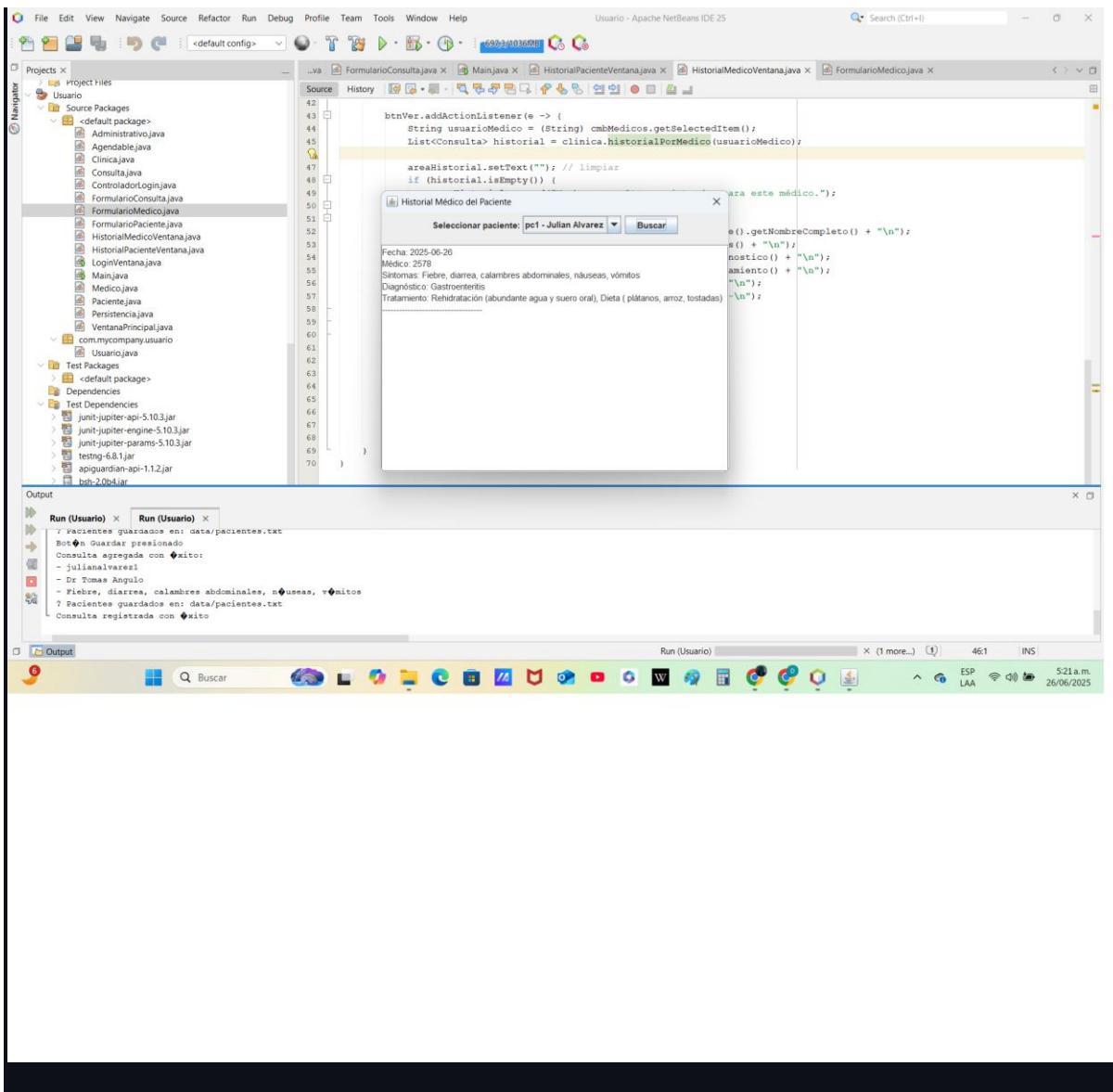


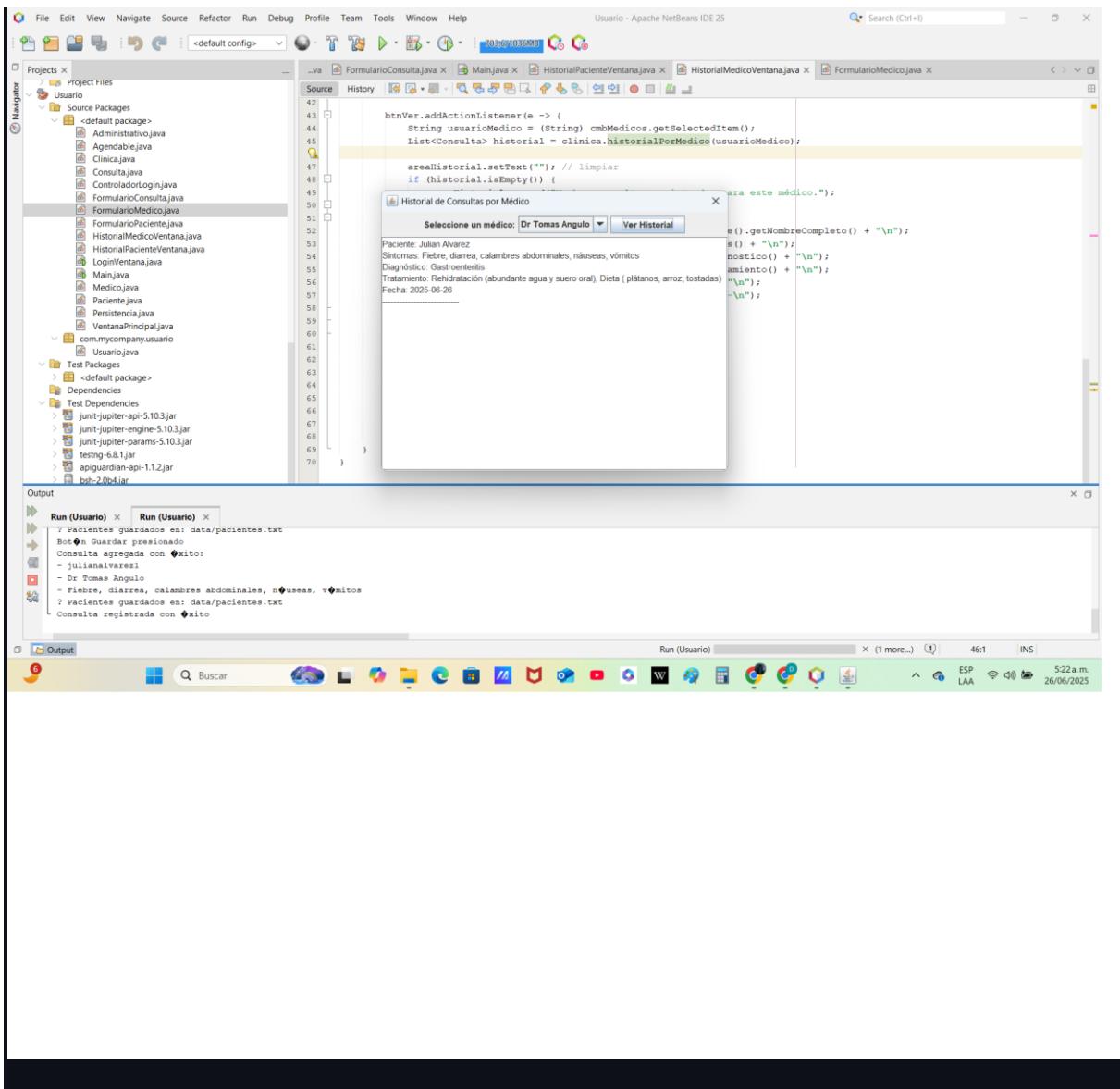


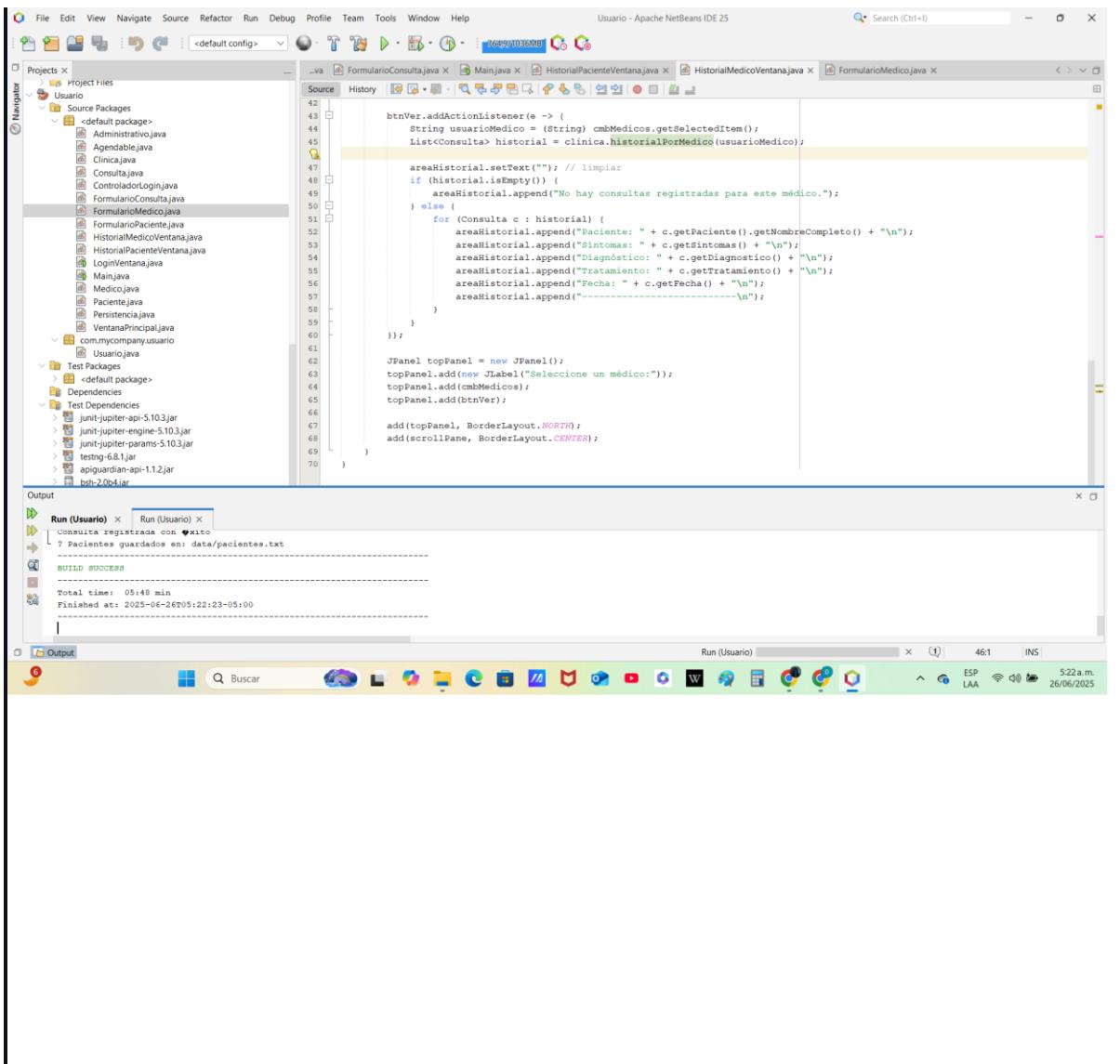












## 7. Resumen del proyecto: ¿Con qué se trabajó? ¿Qué se usó? ¿Cómo se usó?

El proyecto demuestra el uso de la Programación Orientada a Objetos con herencia, encapsulamiento, polimorfismo y modularidad. Se trabajó con archivos de texto para persistencia, diseño de interfaces gráficas con Swing, y se manejaron errores comunes como NullPointerException. Se Implementó un patrón MVVM, Implementación de una clase abstracta e interfaces, Uso de colecciones genéricas, Persistencia de datos en archivos planos, Manejo de excepciones personalizadas, Interfaz gráfica en Jswing, con menús funcionales.