

# Global Programación

Tema: Documentos

Integrantes: Ariza, Cosentino

## Explicación del problema:

El cliente (empresa) solicita realizar un sistema para el área de administración de documentos.

- La empresa maneja documentos que se envían al exterior. De cada documento se conoce la fecha de creación, el autor, el destinatario, y las palabras clave que permiten identificarlo.
- De cada persona que trabaja en la empresa se conoce el nombre, dirección, teléfono, fecha de ingreso y el cargo que ocupa.
- Los documentos se generan para ser enviados a algún ente externo, y por lo tanto deben ser despachados por correo. De los entes externos se conoce el nombre, la dirección y el teléfono.
- La empresa trabaja con varios correos, conociendo de cada uno el nombre de la empresa, la dirección, el teléfono, y el nombre de la persona encargada para los contactos.
- Cuando se despacha un correo externo se registra con qué empresa se envía, el número de seguimiento que entrega la misma, y se marca el estado como “enviado”.
- A medida que las empresas de correo devuelven las constancias de entrega se registra el estado correspondiente en cada documento.

El cliente exige que el sistema incluya un buscador de documentos por palabra clave; que pueda determinar el empleado que más documentos ha confeccionado; que se registre los documentos que han sido enviados y que aún no se entregan, registrándolos como “en espera”.

## Clase Main:

```
1 package com.mycompany;
2
3 import com.mycompany.controllers.Controlador;
4 import com.mycompany.controllers.ControladorEnviarDocs;
5 import com.mycompany.models.Documento;
6 import com.mycompany.models.EnteCorreo;
7 import com.mycompany.models.Envio;
8 import com.mycompany.models.Persona;
9 import com.mycompany.models.Services.service;
10 import com.mycompany.models.repository.Repository;
11 import com.mycompany.view.BuscarPorPalabraClave;
12 import com.mycompany.view.CantidadEnEspera;
13 import com.mycompany.view.DatosCorreo_Empresa;
14 import com.mycompany.view.EmpleadoQueMasCofeccionoDocs;
15 import com.mycompany.view.EnviarDocs;
16 import com.mycompany.view.Menu;
17
18 public class ParcialProgramacion {
19
20     public static void main(String[] args) {
21         //Vistas
22         Menu vista = new Menu();
23         DatosCorreo_Empresa enviarDocs2 = new DatosCorreo_Empresa();
24         EnviarDocs enviarDocs = new EnviarDocs();
25         BuscarPorPalabraClave buscar = new BuscarPorPalabraClave();
26         CantidadEnEspera cantidad = new CantidadEnEspera();
27         EmpleadoQueMasCofeccionoDocs emp = new EmpleadoQueMasCofeccionoDocs();
28
29         service modelo = new service();
30
31         ControladorEnviarDocs controladorEnviarDocs = new ControladorEnviarDocs(enviarDocs, enviarDocs2, modelo, vista);
32
33         Controlador controlador = new Controlador(vista, modelo, buscar, cantidad, emp);
34         controlador.iniciar();
35         vista.setVisible(true);
36     }
37
38 }
```

Modelo:

Documento:

```
1 package com.mycompany.models;
2
3 import java.util.ArrayList;
4 import javax.naming.InvalidNameException;
5 import java.util.List;
6 import java.util.Optional;
7 import java.util.stream.Collectors;
8 import com.mycompany.models.repository.Repository;
9 import java.util.Date;
10
11 public class Documento {
12
13     private String destinatario;
14     private Date fecha_creacion;
15     private String autor;
16     private List<String> palabraClave;
17     private ArrayList<Envio> enviado = new ArrayList<>();
18     public List<EnteCorreo> se_envia;
19     private List<Persona> trabaja;
20
21     public Documento() {
22         this.palabraClave = new ArrayList<>();
23         this.enviado = new ArrayList<>();
24         this.se_envia = new ArrayList<>();
25         this.trabaja = new ArrayList<>();
26     }
27
28     public String getDestinatario() {
29         return destinatario;
30     }
31
32     public Date getFecha_creacion() {
33         return fecha_creacion;
34     }
35
36     public String getAutor() {
37         return autor;
38     }
39 }
```

```
40 public List<String> getPalabra_clave() {
41     return palabraClave;
42 }
43
44 public void setDestinatario(String destinatario) {
45     this.destinatario = destinatario;
46 }
47
48 public void setFecha_creacion(Date fecha_creacion) {
49     this.fecha_creacion = fecha_creacion;
50 }
51
52 public void setAutor(String autor) {
53     this.autor = autor;
54 }
55
56 public void setPalabraClave(List<String> palabraClave) {
57     this.palabraClave = palabraClave;
58 }
59
60 public List<Envio> getEnviado() {
61     return enviado;
62 }
63
64 public void setEnvio(Envio enviado) {
65     this.enviado.add(enviado);
66 }
67
68 public void setSe_envia(EnteCorreo se_envia) {
69     this.se_envia.add(se_envia);
70 }
71
72 public void setTrabaja(Persona trabaja) {
73     this.trabaja.add(trabaja);
74 }
75
76 }
```

EmpresaCorreo:

```
1 package com.mycompany.models;
2
3 public class EmpresaCorreo {
4
5     private String nombre;
6     private String direccion;
7     private String telefono;
8     private String encargado;
9
10    public String getNombre() {
11        return nombre;
12    }
13
14    public void setNombre(String nombre) {
15        this.nombre = nombre;
16    }
17
18    public String getDireccion() {
19        return direccion;
20    }
21
22    public void setDireccion(String direccion) {
23        this.direccion = direccion;
24    }
25
26    public String getTelefono() {
27        return telefono;
28    }
29
30    public void setTelefono(String telefono) {
31        this.telefono = telefono;
32    }
33
34    public String getEncargado() {
35        return encargado;
36    }
37
38    public void setEncargado(String encargado) {
39        this.encargado = encargado;
40    }
41
42 }
```

EnteCorreo:

```
2   package com.mycompany.models;
3
4   import java.util.ArrayList;
5   import java.util.List;
6
7   public class EnteCorreo {
8
9       private String nombre;
10      private String direccion;
11      private int telefono;
12      private String encargado;
13
14      public void setNombre(String nombre) {
15          this.nombre = nombre;
16      }
17
18      public void setDireccion(String direccion) {
19          this.direccion = direccion;
20      }
21
22      public void setTelefono(int telefono) {
23          this.telefono = telefono;
24      }
25
26      public void setEncargado(String encargado) {
27          this.encargado = encargado;
28      }
29
30  }
```

Envio:

```
2  package com.mycompany.models;
3
4  public class Envio {
5
6      private int nro_seguimiento;
7      private boolean estado_enviado;
8
9      public void setNro_seguimiento(int nro_seguimiento) {
10         this.nro_seguimiento = nro_seguimiento;
11     }
12
13     public void setEstado_enviado(boolean estado_enviado) {
14         this.estado_enviado = estado_enviado;
15     }
16
17     public boolean isEstado_enviado() {
18         return estado_enviado;
19     }
20
21     public int getNro_seguimiento() {
22         return nro_seguimiento;
23     }
24 }
25
```

Persona:

```
2  package com.mycompany.models;
3
4  import java.util.Date;
5
6  public class Persona {
7
8      private String nombre;
9      private String direccion;
10     private String telefono;
11     private Date fecha_ingreso;
12     private String cargo;
13
14     public Persona() {
15     }
16
17
18     public void setNombre(String nombre) {
19         this.nombre = nombre;
20     }
21
22     public void setDireccion(String direccion) {
23         this.direccion = direccion;
24     }
25
26     public void setTelefono(String telefono) {
27         this.telefono = telefono;
28     }
29
30     public void setFecha_ingreso(Date fecha_ingreso) {
31         this.fecha_ingreso = fecha_ingreso;
32     }
33
34     public void setCargo(String cargo) {
35         this.cargo = cargo;
36     }
```



```
38  public String getNombre() {
39      return nombre;
40  }
41
42  public String getDireccion() {
43      return direccion;
44  }
45
46  public String getTelefono() {
47      return telefono;
48  }
49
50  public Date getFecha_ingreso() {
51      return fecha_ingreso;
52  }
53
54  public String getCargo() {
55      return cargo;
56  }
57
58  }
```

Servicios para BD:

```
1  package com.mycompany.models.Services;
2
3  import java.util.List;
4  import java.util.Optional;
5  import javax.naming.InvalidNameException;
6  import com.mycompany.models.Documento;
7  import com.mycompany.models.EmpresaCorreo;
8  import com.mycompany.models.EnteCorreo;
9  import com.mycompany.models.Envio;
10 import com.mycompany.models.Persona;
11 import com.mycompany.models.repository.Repository;
12 import com.mycompany.view.Menu;
13
14 public class service {
15
16     private Repository repository;
17     private Documento documentos;
18     private Envio envio;
19     private Persona persona;
20     private EnteCorreo enteCorreo;
21     private EmpresaCorreo empresa;
22     private Menu vista;
23
24     public service() {
25         this.documentos = new Documento();
26         this.envio = new Envio();
27         this.persona = new Persona();
28         this.enteCorreo = new EnteCorreo();
29         this.empresa = new EmpresaCorreo();
30         this.repository = new Repository();
31     }
32
33     public Documento getDocumento() {
34         return documentos;
35     }
```

```

37 public Envio getEnvio() {
38     return envio;
39 }
40
41 public Persona getPersona() {
42     return persona;
43 }
44
45 public EmpresaCorreo getEmpresa() {
46     return empresa;
47 }
48
49 public EnteCorreo getEnteCorreo() {
50     return enteCorreo;
51 }
52
53 public List<Documento> documentoQueIncluyen(String palabra) throws InvalidNameException {
54     List<Documento> documentos = repository.buscarPorPalabraClave(palabra);
55     if (documentos.isEmpty()) {
56         throw new InvalidNameException("No se encontraron documentos con la palabra clave proporcionada.");
57     }
58     return documentos;
59 }
60
61 public String autorMasProductivo() {
62     return repository.autorMasProductivo();
63 }
64
65 public Optional<Integer> cantidadEnEspera() {
66     return repository.cantidadEnEsperaConsulta();
67 }
68
69
70

```

```

72 public boolean guardarEnBD(Documento documento, Persona persona, Envio envio, EmpresaCorreo empresa) {
73     //Llamamos al repository para guardar los objetos en base de datos
74     if (this.repository.insertarValoresDocumentosBD(documentos, persona, envio, empresa)) {
75         return true;
76     }
77     return false;
78 }
79
80

```

## Conexión a BD:

```
1 package com.mycompany.models.repository;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import javax.swing.JOptionPane;
6
7 public class Conexion {
8
9     public static final String usuario = "root";
10    public static final String contrasenia = "prisma18";
11    public static final String bd = "bd_documento";
12    public static final String ip = "localhost";
13    public static final String puerto = "3306";
14
15    public static final String cadena = "jdbc:mysql://" + ip + ":" + puerto + "/" + bd + "?allowPublicKey
16
17    public Connection establecerConexion() {
18        Connection conectar = null;
19
20        try {
21            Class.forName("com.mysql.cj.jdbc.Driver");
22            conectar = DriverManager.getConnection(cadena, usuario, contrasenia);
23            System.out.println("conexion a base de datos");
24        } catch (Exception e) {
25            JOptionPane.showMessageDialog(null, "Ocurrio un error: " + e.toString());
26        }
27        return conectar;
28    }
29
30 }
```

## Repositorio para BD:

```
1 package com.mycompany.models.repository;
2
3 import com.mycompany.models.Documento;
4 import com.mycompany.models.EmpresaCorreo;
5 import com.mycompany.models.Envio;
6 import com.mycompany.models.Persona;
7 import javax.swing.*;
8 import java.sql.Connection;
9 import java.sql.PreparedStatement;
10 import java.sql.ResultSet;
11 import java.sql.SQLException;
12 import java.sql.Statement;
13 import java.text.ParseException;
14 import java.text.SimpleDateFormat;
15 import java.util.ArrayList;
16 import java.util.List;
17 import java.util.Optional;
18
19 public class Repository extends Conexion {
20
21    public boolean insertarValoresDocumentosBD(Documento documento, Persona persona, Envio envio, EmpresaCorreo empresa) {
22        Connection conexion = establecerConexion();
23
24        try {
25            conexion.setAutoCommit(false);
26
27            // Insertar en Empleados
28            String sqlPersona = "INSERT INTO Empleados (nombre, direccion, telefono, fecha_ingreso, cargo) VALUES (?, ?, ?, ?, ?)";
29            PreparedStatement psPersona = conexion.prepareStatement(
30                sqlPersona,
31                Statement.RETURN_GENERATED_KEYS
32            );
33            psPersona.setString(1, persona.getNombre());
34            psPersona.setString(2, persona.getDireccion());
35            psPersona.setString(3, persona.getTelefono());
```

```

36      //Date a String
37      SimpleDateFormat sdf = new SimpleDateFormat("dd-mm-yyyy");
38
39      String fechaCadenaPersona = sdf.format(persona.getFecha_ingreso());
40
41      psPersona.setString(4, fechaCadenaPersona);
42      psPersona.setString(5, persona.getCargo());
43
44      int personaResultado = psPersona.executeUpdate();
45      if (personaResultado == 0) {
46          conexion.rollback();
47          return false;
48      }
49
50      // Obtener el ID generado para la tabla Empleados
51      ResultSet personaKeys = psPersona.getGeneratedKeys();
52      int personaId = 0;
53      if (personaKeys.next()) {
54          personaId = personaKeys.getInt(1);
55      } else {
56          throw new SQLException("No se generó el ID de Empleados");
57      }
58
59      // Insertar en Envio
60      String sqlEnvio = "INSERT INTO Envio (estado_enviado, nro_seguimiento) VALUES (?, ?)";
61      PreparedStatement psEnvio = conexion.prepareStatement(
62          sqlEnvio,
63          Statement.RETURN_GENERATED_KEYS
64      );
65      psEnvio.setBoolean(1, envio.isEstado_enviado());
66      psEnvio.setInt(2, envio.getNro_seguimiento());
67
68      int envioResultado = psEnvio.executeUpdate();
69      if (envioResultado == 0) {
70          conexion.rollback();
71          return false;
72      }

```

```

74      // Insertar en EmpresaCorreo
75      String sqlEmpresa = "INSERT INTO EmpresaCorreo (nombre, direccion, telefono, encargado) VALUES (?, ?, ?, ?)";
76      PreparedStatement psEmpresa = conexion.prepareStatement(
77          sqlEmpresa
78      );
79      psEmpresa.setString(1, empresa.getNombre());
80      psEmpresa.setString(2, empresa.getDireccion());
81      psEmpresa.setString(3, empresa.getTelefono());
82      psEmpresa.setString(4, empresa.getEncargado());
83
84      int empresaResultado = psEmpresa.executeUpdate();
85      if (empresaResultado == 0) {
86          conexion.rollback();
87          return false;
88      }
89
90      // Insertar en Documento
91      String sqlDocumento = "INSERT INTO Documento (autor, destinatario, fecha_creacion, palabra_clave, id_empleado)";
92      PreparedStatement psDocumento = conexion.prepareStatement(
93          sqlDocumento
94      );
95      psDocumento.setString(1, documento.getAutor());
96      psDocumento.setString(2, documento.getDestinatario());
97
98      String fechaCadDoc = sdf.format(documento.getFecha_creacion());
99
100      psDocumento.setString(3, fechaCadDoc);
101      psDocumento.setString(4, convertirListaAJson(documento.getPalabra_clave()));
102      psDocumento.setInt(5, personaId);
103
104      int documentoResultado = psDocumento.executeUpdate();
105      if (documentoResultado == 0) {
106          conexion.rollback();
107          return false;
108      }

```

```

110         // Confirmar la transacción
111         conexion.commit();
112         return true;
113
114     } catch (SQLException e) {
115         try {
116             if (conexion != null) {
117                 conexion.rollback();
118             }
119         } catch (SQLException ex) {
120             ex.printStackTrace();
121         }
122         e.printStackTrace();
123         return false;
124     } finally {
125         try {
126             if (conexion != null) {
127                 conexion.setAutoCommit(true);
128                 conexion.close();
129             }
130         } catch (SQLException ex) {
131             ex.printStackTrace();
132         }
133     }
134 }

```

```

136 public List<Documento> buscarPorPalabraClave(String palabra) {
137     List<Documento> documentos = new ArrayList<>();
138     Connection conexion = establecerConexion();
139     if (conexion == null) {
140         System.err.println("No se pudo establecer conexión con la base de datos.");
141         return documentos;
142     }
143     try {
144
145         String palabraJson = "\"" + palabra + "\"";
146
147         PreparedStatement ps = conexion.prepareStatement("SELECT * FROM Documento WHERE JSON_CONTAINS(palabra_clave, ?)");
148         SimpleDateFormat sdf = new SimpleDateFormat("dd-mm-yyyy");
149
150         ps.setString(1, palabraJson);
151         ResultSet rs = ps.executeQuery();
152         System.out.println(rs);
153         while (rs.next()) {
154             Documento doc = new Documento();
155             doc.setAutor(rs.getString("autor"));
156             doc.setDestinatario(rs.getString("destinatario"));
157
158             try {
159                 doc.setFecha_creacion(sdf.parse(rs.getString("fecha_creacion")));
160             } catch (ParseException e) {
161             }
162
163             String palabraClaveJson = rs.getString("palabra_clave");
164             List<String> palabraClaveList = convertirJsonALista(palabraClaveJson);
165             System.out.println(palabraClaveList);
166             doc.setPalabraClave(palabraClaveList);
167
168             documentos.add(doc);
169         }
170     } catch (SQLException e) {
171         e.printStackTrace();
172     }

```

```

173         return documentos;
174     }
175
176     public String autorMasProductivo() {
177
178         Connection conexion = establecerConexion();
179
180         String sql = "SELECT e.id_empleado, e.nombre, COUNT(d.id_documento) AS cantidad_documentos "
181             + "FROM Empleados e "
182             + "JOIN Documento d ON e.id_empleado = d.id_empleado "
183             + "GROUP BY e.id_empleado, e.nombre "
184             + "ORDER BY cantidad_documentos DESC "
185             + "LIMIT 1";
186
187         try {
188             PreparedStatement ps = conexion.prepareStatement(sql);
189             ResultSet rs = ps.executeQuery();
190             if (rs.next()) {
191                 int idEmpleado = rs.getInt("id_empleado");
192                 String nombreEmpleado = rs.getString("nombre");
193                 int cantidadDocumentos = rs.getInt("cantidad_documentos");
194
195                 String resultado = "Nombre: " + nombreEmpleado + "\n"
196                     + ", Cantidad de documentos confeccionados: " + cantidadDocumentos;
197                 return resultado;
198             }
199         } catch (Exception e) {
200         }
201
202         return "No se encontraron resultados";
203     }

```

```

205     public Optional<Integer> cantidadEnEsperaConsulta() {
206         String sql = "SELECT COUNT(*) FROM Envio WHERE estado_enviado = true";
207         Connection conexion = establecerConexion();
208         try {
209             PreparedStatement ps = conexion.prepareStatement(sql);
210             ResultSet rs = ps.executeQuery();
211
212             if (rs.next()) {
213                 int cantidad = rs.getInt(1);
214                 return Optional.of(cantidad);
215             } else {
216                 return Optional.empty();
217             }
218         } catch (SQLException e) {
219             e.printStackTrace();
220         }
221         return Optional.empty();
222     }
223
224     private static String convertirListaAJson(List<String> lista) {
225         StringBuilder json = new StringBuilder("[");
226         for (int i = 0; i < lista.size(); i++) {
227             json.append("[\"").append(lista.get(i)).append("\"]");
228             if (i < lista.size() - 1) {
229                 json.append(",");
230             }
231         }
232         json.append("]");
233         return json.toString();
234     }

```

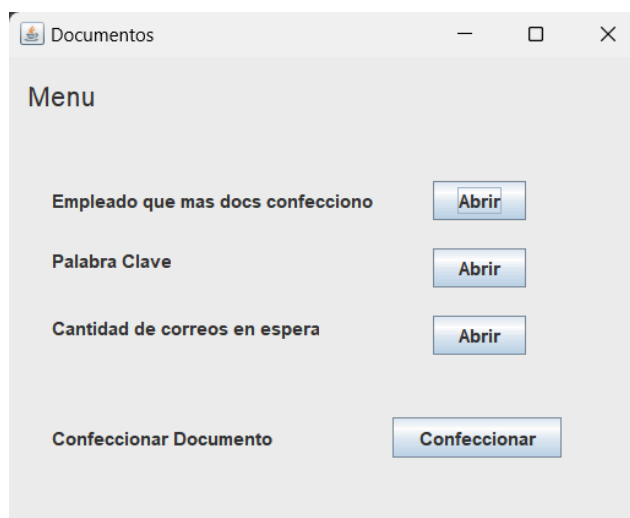
```

236     private List<String> convertirJsonALista(String json) {
237         json = json.replace("[", "").replace("]", "").replace("\"", "");
238         String[] items = json.split(",");
239         List<String> list = new ArrayList<>();
240         for (String item : items) {
241             list.add(item.trim());
242         }
243         return list;
244     }
245 }

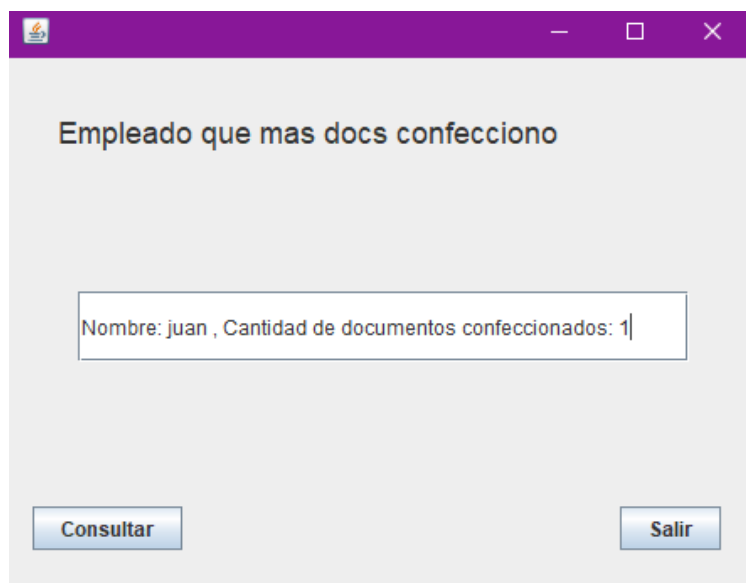
```

Vista:

Menú:

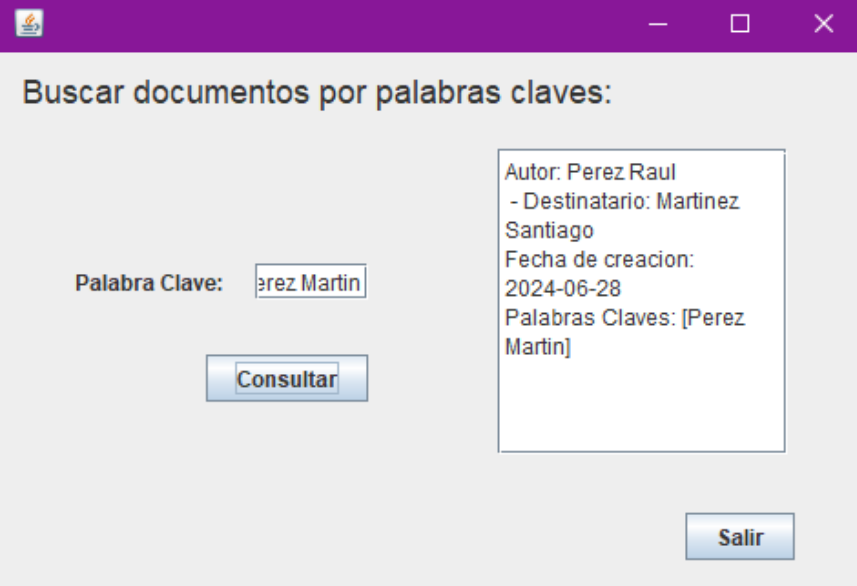


Método empleado que más confeccionó:





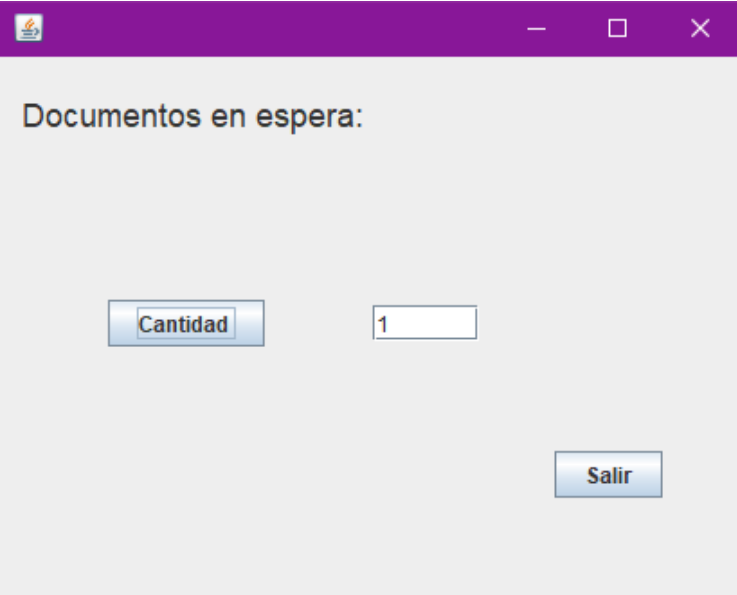
Método buscar por palabra clave:



The screenshot shows a web application window with a purple title bar. The main content area is light gray and contains the following elements:

- Title:** "Buscar documentos por palabras claves:"
- Form:** A label "Palabra Clave:" is followed by a text input field containing "Perez Martin".
- Buttons:** A blue "Consultar" button is positioned below the input field. A blue "Salir" button is located in the bottom right corner.
- Preview Box:** A white box with a thin border on the right side displays the following text:  
Autor: Perez Raul  
- Destinatario: Martinez Santiago  
Fecha de creacion: 2024-06-28  
Palabras Claves: [Perez Martin]

Método documentos en espera:



The screenshot shows a web application window with a purple title bar. The main content area is light gray and contains the following elements:

- Title:** "Documentos en espera:"
- Form:** A blue button labeled "Cantidad" is followed by a text input field containing the number "1".
- Buttons:** A blue "Salir" button is located in the bottom right corner.

Enviar Documentos:



—

□

×

## Enviar Documentos

Datos Trabajador:


<input type="text" value="juan"/>	<input type="text" value="26169488"/>	<input type="text" value="Barrio Jar"/>	<input type="text" value="28/6/2024"/>	<input type="text" value="Encagado"/>
Nombre	Telefono	Dirreccion	Fecha de ingreso	Cargo

Datos Documento:                      Datos ente externo:

Autor	<input type="text" value="Perez Ra"/>	Nombre	<input type="text" value="Martinez"/>
Destinatario	<input type="text" value="Martinez"/>	Direccion	<input type="text" value="Godoy cr"/>
Palabras Claves	<input type="text" value="Perez Ma"/>	Telefono	<input type="text" value="2615346"/>

Limpiar

Siguiente



—

□

×

## Datos de la empresa de correo

Nombre	<input type="text" value="Correos :"/>
Encargado	<input type="text" value="Rodrigue"/>
Direccion	<input type="text" value="Mendoza"/>
Telefono	<input type="text" value="2615234"/>

## Empresa encargada del envio

Nombre empresa	<input type="text" value="Andreani"/>
Num° Seguimiento	<input type="text" value="4321"/>
Estado Envio	<input type="text" value="Enviado"/>

Terminar

Controlador:

Clase Controlador:

```
1 package com.mycompany.controllers;
2
3 import com.mycompany.models.Documento;
4 import com.mycompany.view.Menu;
5 import javax.naming.InvalidNameException;
6 import java.awt.event.ActionEvent;
7 import java.awt.event.ActionListener;
8 import java.util.List;
9 import java.util.Optional;
10 import com.mycompany.models.Services.service;
11 import com.mycompany.view.BuscarPorPalabraClave;
12 import com.mycompany.view.CantidadEnEspera;
13 import com.mycompany.view.EmpleadoQueMasCofeccionoDocs;
14 import javax.swing.JOptionPane;
15
16 //Controlador para los metodos, BuscarPorPalabraClave, CantidadEnEspera, EmpleadoQueMasCofeccionoDocs
17 public class Controlador implements ActionListener {
18
19     private Menu vista;
20     private BuscarPorPalabraClave buscar;
21     private CantidadEnEspera cantidadEspera;
22     private EmpleadoQueMasCofeccionoDocs empMasConfecciono;
23     private service modelo;
24
25     public Controlador(Menu vista, service modelo, BuscarPorPalabraClave buscar, CantidadEnEspera cantidadEspera, EmpleadoQueMasCofeccionoDocs empMasConfecciono) {
26
27         this.vista = vista;
28         this.buscar = buscar;
29         this.cantidadEspera = cantidadEspera;
30         this.empMasConfecciono = empMasConfecciono;
31         this.modelo = modelo;
32         //Buscar boton
33         vista.consultarPorPalabraButtonModal.addActionListener(this);
34         //Cantidad en espera boton
35         vista.cantidadEnEsperaButtonModal.addActionListener(this);
36         //Boton para el empleado que mas confecciono docs
37         vista.cantidadConfeccionoButtonModal.addActionListener(this);
38     }
39 }
```

```

38      //Boton para buscar por palabras claves
39      buscar.consultarPorPalabraButton.addActionListener(this);
40      //Botones para abrir las ventanas externas
41      buscar.salirButton.addActionListener(this);
42      empMasConfecciono.salirButtonEmpleado.addActionListener(this);
43      cantidadEspera.salirButtonEspera.addActionListener(this);
44  }
45
46  public void iniciar() {
47      vista.setTitle("Documentos");
48      vista.setLocationRelativeTo(null);
49  }
50
51  //Usamos el objeto ae para saber que boton se presiona
52  @Override
53  public void actionPerformed(ActionEvent ae) {
54
55      // Ventana para buscar por palabra ingresada
56      if (ae.getSource() == vista.consultarPorPalabraButtonModal) {
57          vista.dispose();
58          buscar.setVisible(true);
59      } else if (ae.getSource() == buscar.consultarPorPalabraButton) {
60          try {
61              String palabra = buscar.palabraClaveTextField.getText();
62              List<Documento> resultados = modelo.documentoQueIncluyen(palabra);
63              //Si la palabra no se encuentra mostramos error
64              if (resultados.isEmpty()) {
65                  System.out.println("No se encontraron documentos con esa palabra clave");
66                  limpiarCajasBuscar();
67                  //sino se muestra en la vista el resultado
68              } else {
69                  buscar.mostrarDocumentos(resultados);
70              }
71          } catch (InvalidNameException e) {
72              JOptionPane.showMessageDialog(vista, "Nombre inválido: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
73          }
74      }

```

```

74      //Boton salir
75      } else if (ae.getSource() == buscar.salirButton) {
76          buscar.dispose();
77          vista.setVisible(true);
78      }
79
80      // Ventana cantidad en espera
81      if (ae.getSource() == vista.cantidadEnEsperaButtonModal) {
82          vista.dispose();
83          cantidadEspera.setVisible(true);
84          Optional<Integer> cantidadDocs = modelo.cantidadEnEspera();
85          if (cantidadDocs.isPresent()) {
86              String docs = String.valueOf(cantidadDocs.get());
87              cantidadEspera.mostrarCantDocs(docs);
88          } else {
89              cantidadEspera.mostrarError("No se encontraron documentos en espera");
90          }
91          //Boton salir
92      } else if (ae.getSource() == cantidadEspera.salirButtonEspera) {
93          cantidadEspera.dispose();
94          vista.setVisible(true);
95      }
96
97      // Método empleado que más confeccionó documentos
98      if (ae.getSource() == vista.cantidadConfecccButtonModal) {
99          vista.dispose();
100          empMasConfecciono.setVisible(true);
101          empMasConfecciono.cantdocsConfecccTextArea.setText(modelo.autorMasProductivo());
102          //Boton salir
103      } else if (ae.getSource() == empMasConfecciono.salirButtonEmpleado) {
104          empMasConfecciono.dispose();
105          vista.setVisible(true);
106      }
107
108  }

```

```

110     public void limpiarCajasBuscar() {
111         buscar.palabraClaveTextField.setText("");
112     }
113 }

```

Controlador para EnviarDocs:

```

1  package com.mycompany.controllers;
2
3  import com.mycompany.models.Documento;
4  import com.mycompany.models.EmpresaCorreo;
5  import com.mycompany.models.EnteCorreo;
6  import com.mycompany.models.Envio;
7  import com.mycompany.models.Persona;
8  import com.mycompany.models.Services.service;
9  import com.mycompany.view.DatosCorreo_Empresa;
10 import com.mycompany.view.EnviarDocs;
11 import com.mycompany.view.Menu;
12 import java.awt.event.ActionEvent;
13 import java.awt.event.ActionListener;
14 import java.text.ParseException;
15 import java.text.SimpleDateFormat;
16 import java.util.Date;
17 import java.util.List;
18 import javax.swing.JOptionPane;
19
20 public class ControladorEnviarDocs implements ActionListener {
21
22     private service modelo;
23     private EnviarDocs enviarDocs;
24     private DatosCorreo_Empresa enviarDocs2;
25     private Menu vista;
26
27     public ControladorEnviarDocs(EnviarDocs enviarDocs, DatosCorreo_Empresa enviarDocs2, service modelo, Menu vista) {
28         this.enviarDocs = enviarDocs;
29         this.enviarDocs2 = enviarDocs2;
30         this.modelo = modelo;
31         this.vista = vista;
32
33         // Agregar ActionListeners
34         vista.EnviaDocsFlotButton.addActionListener(this);
35         enviarDocs.siguieteButton.addActionListener(this);
36         enviarDocs2.TerminarButton.addActionListener(this);
37     }

```

```

40 public void actionPerformed(ActionEvent ae) {
41     try {
42         //Si se preciona el boton enviar en el menu
43         if (ae.getSource() == vista.EnviaDocsFlotButton) {
44             vista.dispose();
45             enviarDocs.setVisible(true);
46             //Si se preciona el boton siguiente
47         } else if (ae.getSource() == enviarDocs.siguienteButton) {
48             if (validarCamposPanel1()) {
49                 guardarDatosPanel1();
50                 enviarDocs.dispose();
51                 enviarDocs2.setVisible(true);
52             }
53             //Si se preciona el boton Terminar
54         } else if (ae.getSource() == enviarDocs2.TerminarButton) {
55             //validamos que no haya campos vacios
56             if (validarCamposPanel2()) {
57                 guardarDatosPanel2();
58                 //Guardamos en base de datos
59                 if (modelo.guardarEnBD(modelo.getDocumento(), modelo.getPersona(), modelo.getEnvio(), modelo.getEmpresa())) {
60                     JOptionPane.showMessageDialog(null, "Datos guardados correctamente.");
61                 } else {
62                     JOptionPane.showMessageDialog(null, "Ocurrió un error al guardar los datos.");
63                 }
64                 enviarDocs2.dispose();
65                 vista.setVisible(true);
66             }
67         } catch (ParseException e) {
68             JOptionPane.showMessageDialog(null, "Error al parsear la fecha: " + e.getMessage());
69         } catch (Exception e) {
70             JOptionPane.showMessageDialog(null, "Ocurrió un error: " + e.getMessage());
71         }
72     }
73 }

```

```

75 private boolean validarCamposPanel1() {
76     if (enviarDocs.destinatarioField.getText().isEmpty()
77         || enviarDocs.autorField.getText().isEmpty()
78         || enviarDocs.palabrasClavesField.getText().isEmpty()
79         || enviarDocs.nombreTextField.getText().isEmpty()
80         || enviarDocs.telefonoTextField.getText().isEmpty()
81         || enviarDocs.dirreccionTextField.getText().isEmpty()
82         || enviarDocs.CargoTextField.getText().isEmpty()) {
83         JOptionPane.showMessageDialog(null, "Por favor, complete todos los campos del primer panel.");
84         return false;
85     }
86     return true;
87 }
88
89 private void guardarDatosPanel1() throws ParseException {
90     //Damos formato a la fecha
91     SimpleDateFormat objSDF = new SimpleDateFormat("dd-mm-yyyy");
92     Date objDate = new Date();
93
94     Documento documentos = modelo.getDocumento();
95     Persona persona = modelo.getPersona();
96     EnteCorreo enteCorreo = modelo.getEnteCorreo();
97
98     documentos.setDestinatario(enviarDocs.destinatarioField.getText());
99     documentos.setAutor(enviarDocs.autorField.getText());
100     documentos.setFecha_creacion(objSDF.parse(objSDF.format(objDate)));
101     documentos.setPalabraClave(stringAList(enviarDocs.palabrasClavesField.getText()));
102
103     persona.setNombre(enviarDocs.nombreTextField.getText());
104     persona.setTelefono(enviarDocs.telefonoTextField.getText());
105     persona.setFecha_ingreso(objSDF.parse(enviarDocs.fechIngreTextField.getText()));
106     persona.setDireccion(enviarDocs.dirreccionTextField.getText());
107     persona.setCargo(enviarDocs.CargoTextField.getText());
108
109     enteCorreo.setNombre(enviarDocs.nombreEnteDocField.getText());
110     enteCorreo.setDireccion(enviarDocs.dirreccionEnteDocField.getText());
111     enteCorreo.setTelefono(Integer.parseInt(enviarDocs.telefonoEnteField.getText()));

```

```

115 private boolean validarCamposPanel2() {
116     if (enviarDocs2.NombreEmpresaField.getText().isEmpty()
117         || enviarDocs2.EmpresaDireccionField.getText().isEmpty()
118         || enviarDocs2.EmpresaTelefonoField.getText().isEmpty()
119         || enviarDocs2.EncargadoCorreoField.getText().isEmpty()
120         || enviarDocs2.numSegTextField.getText().isEmpty()
121         || enviarDocs2.nombreEmpresaEnviTextField.getText().isEmpty()
122         || enviarDocs2.jComboBox.getSelectedIndex() == null) {
123         JOptionPane.showMessageDialog(null, "Por favor, complete todos los campos del segundo panel.");
124         return false;
125     }
126     return true;
127 }
128
129 private void guardarDatosPanel2() {
130     EmpresaCorreo empresaCorreo = modelo.getEmpresa();
131
132     empresaCorreo.setNombre(enviarDocs2.NombreEmpresaField.getText());
133     empresaCorreo.setEncargado(enviarDocs2.EncargadoCorreoField.getText());
134     empresaCorreo.setDireccion(enviarDocs2.EmpresaDireccionField.getText());
135     empresaCorreo.setTelefono(enviarDocs2.EmpresaTelefonoField.getText());
136
137     Envio envio = modelo.getEnvio();
138
139     envio.setEstado_enviado("Enviado".equals((String) enviarDocs2.jComboBox.getSelectedIndex()));
140     envio.setNro_seguimiento(Integer.parseInt(enviarDocs2.numSegTextField.getText()));
141
142     Documento documentos = modelo.getDocumento();
143     EnteCorreo enteCorreo = modelo.getEnteCorreo();
144     Persona persona = modelo.getPersona();
145
146     documentos.setEnvio(envio);
147     documentos.setSe_envia(enteCorreo);
148     documentos.setTrabaja(persona);
149 }
150
151
152 //funcion para convertir string a list
153 private List<String> stringAList(String s) {
154     //Convertimos la palabra a lista, sacandole la coma y el espacio en blanco
155     List<String> lst = List.of(s.split("\\s*,\\s*"));
156     return lst;
157 }
158 }

```

Resultado en la base de datos:

Tabla de Documento:

Result Grid						
Filter Rows:		Edit:		Export/Import:		Wrap Cell Content:
	id_documento	id_empleado	autor	destinatario	fecha_creacion	palabra_clave
▶	1	1	Perez Raul	Martinez Santiago	2024-06-28	["Perez Martin"]
*	NULL	NULL	NULL	NULL	NULL	NULL

## Tabla de Empleados en SQL:

Result Grid					
Filter Rows:					
Edit: Export/Import: Wrap Cell Content:					
	id_empleado	nombre	direccion	telefono	fecha_ingreso
▶	27	Juan	Barrio Jardin	26 169 488	2024-06-28
✱	NULL	NULL	NULL	NULL	NULL

## Tabla de envíos en SQL:

Result Grid			
Filter Rows:			
Edit: Export/Import: Wrap Cell Content:			
	id_envio	estado_enviado	nro_seguimiento
▶	20	1	4321
✱	NULL	NULL	NULL