

# 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”.

## Captura de excepciones:

Excepción en la clase Controlador para el menú buscar por palabra clave:

```
// Ventana para buscar por palabra clave ingresada
if (ae.getSource() == vista.consultarPorPalabraButtonModal) {
    vista.dispose();
    buscar.setVisible(true);
} else if (ae.getSource() == buscar.consultarPorPalabraButton) {
    try {
        String palabra = buscar.palabraClaveTextField.getText();
        List<Documento> resultados = modelo.documentoQueIncluyen(palabra);
        //Si la palabra no se encuentra mostramos error
        if (resultados.isEmpty()) {
            System.out.println("No se encontraron documentos con esa palabra clave");
            limpiarCajasBuscar();
            //sino se muestra en la vista el resultado
        } else {
            buscar.mostrarDocumentos(resultados);
        }
    } catch (InvalidNameException e) {
        JOptionPane.showMessageDialog(vista, "Nombre inválido: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
}
```

Excepción en la clase ControladorEnviarDocs para el menú enviar documento:

```
try {
    //Si se presiona el boton enviar en el menu
    if (ae.getSource() == vista.EnviarDocsFlotButton) {
        vista.dispose();
        enviarDocs.setVisible(true);
        //Si se presiona el boton siguiente
    } else if (ae.getSource() == enviarDocs.siguienteButton) {
        if (validarCamposPanel1()) {
            guardarDatosPanel1();
            enviarDocs.dispose();
            enviarDocs2.setVisible(true);
        }
        //Si se presiona el boton Terminar
    } else if (ae.getSource() == enviarDocs2.TerminarButton) {
        //validamos que no haya campos vacios
        if (validarCamposPanel2()) {
            guardarDatosPanel2();
            //Guardamos en base de datos
            if (modelo.guardarEnBD(modelo.getDocumento(), modelo.getPersona(), modelo.getEnvio(), modelo.getEmpresa())) {
                JOptionPane.showMessageDialog(null, "Datos guardados correctamente.");
            } else {
                JOptionPane.showMessageDialog(null, "Ocurrió un error al guardar los datos.");
            }
            enviarDocs2.dispose();
            vista.setVisible(true);
        }
    }
} catch (ParseException e) {
    JOptionPane.showMessageDialog(null, "Error al parsear la fecha: " + e.getMessage());
} catch (Exception e) {
    JOptionPane.showMessageDialog(null, "Ocurrió un error: " + e.getMessage());
}
```

```

private void guardarDatosPanel1() throws ParseException {
    //Damos formato a la fecha
    SimpleDateFormat objSDF = new SimpleDateFormat("dd-mm-yyyy");
    Date objDate = new Date();

    Documento documentos = modelo.getDocumento();
    Persona persona = modelo.getPersona();
    EnteCorreo enteCorreo = modelo.getEnteCorreo();

    documentos.setDestinatario(enviarDocs.destinatarioField.getText());
    documentos.setAutor(enviarDocs.autorField.getText());
    documentos.setFecha_creacion(objSDF.parse(objSDF.format(objDate)));
    documentos.setPalabraClave(stringAList(enviarDocs.palabrasClavesField.getText()));

    persona.setNombre(enviarDocs.nombreTextField.getText());
    persona.setTelefono(enviarDocs.telefonoTextField.getText());
    persona.setFecha_ingreso(objSDF.parse(enviarDocs.fechIngreTextField.getText()));
    persona.setDireccion(enviarDocs.dirreccionTextField.getText());
    persona.setCargo(enviarDocs.CargoTextField.getText());

    enteCorreo.setNombre(enviarDocs.nombreEnteDocField.getText());
    enteCorreo.setDireccion(enviarDocs.dirreccionEnteDocField.getText());
    enteCorreo.setTelefono(Integer.parseInt(enviarDocs.telefonoEnteField.getText()));
}

```

Excepción en la clase Service para buscar en la BD los documentos con la palabra clave ingresada:

```

public List<Documento> documentoQueIncluyen(String palabra) throws InvalidNameException {
    List<Documento> documentos = repository.buscarPorPalabraClave(palabra);
    if (documentos.isEmpty()) {
        throw new InvalidNameException("No se encontraron documentos con la palabra clave proporcionada.");
    }
    return documentos;
}

```

Excepción en la clase Conexión para intentar la conexión con la BD:

```

public Connection establecerConexion() {
    Connection conectar = null;
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        conectar = DriverManager.getConnection(cadena, usuario, contrasenia);
        System.out.println("conexion a base de datos");
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Ocurrio un error: " + e.toString());
    }
    return conectar;
}

```

Excepción en la clase Repository para cargar los valores del documento en la BD:

```
public boolean insertarValoresDocumentosBD(Documento documento, Persona persona, Envio envio, EmpresaCorreo empresa) {
    Connection conexion = establecerConexion();

    try {
        conexion.setAutoCommit(false);

        // Insertar en Empleados
        String sqlPersona = "INSERT INTO Empleados (nombre, direccion, telefono, fecha_ingreso, cargo) VALUES (?, ?, ?, ?, ?)";
        PreparedStatement psPersona = conexion.prepareStatement(
            sqlPersona,
            Statement.RETURN_GENERATED_KEYS
        );
        psPersona.setString(1, persona.getNombre());
        psPersona.setString(2, persona.getDireccion());
        psPersona.setString(3, persona.getTelefono());
        //Date a String
        SimpleDateFormat sdf = new SimpleDateFormat("dd-mm-yyyy");

        String fechaCadenaPersona = sdf.format(persona.getFecha_ingreso());

        psPersona.setString(4, fechaCadenaPersona);
        psPersona.setString(5, persona.getCargo());

        int personaResultado = psPersona.executeUpdate();
        if (personaResultado == 0) {
            conexion.rollback();
            return false;
        }

        // Obtener el ID generado para la tabla Empleados
        ResultSet personaKeys = psPersona.getGeneratedKeys();
        int personaId = 0;
        if (personaKeys.next()) {
            personaId = personaKeys.getInt(1);
        } else {
            throw new SQLException("No se generó el ID de Empleados");
        }
    }

    // Insertar en Envio
    String sqlEnvio = "INSERT INTO Envio (estado_enviado, nro_seguimiento) VALUES (?, ?)";
    PreparedStatement psEnvio = conexion.prepareStatement(
        sqlEnvio,
        Statement.RETURN_GENERATED_KEYS
    );
    psEnvio.setBoolean(1, envio.isEstado_enviado());
    psEnvio.setInt(2, envio.getNro_seguimiento());

    int envioResultado = psEnvio.executeUpdate();
    if (envioResultado == 0) {
        conexion.rollback();
        return false;
    }

    // Insertar en EmpresaCorreo
    String sqlEmpresa = "INSERT INTO EmpresaCorreo (nombre, direccion, telefono, encargado) VALUES (?, ?, ?, ?)";
    PreparedStatement psEmpresa = conexion.prepareStatement(
        sqlEmpresa
    );
    psEmpresa.setString(1, empresa.getNombre());
    psEmpresa.setString(2, empresa.getDireccion());
    psEmpresa.setString(3, empresa.getTelefono());
    psEmpresa.setString(4, empresa.getEncargado());

    int empresaResultado = psEmpresa.executeUpdate();
    if (empresaResultado == 0) {
        conexion.rollback();
        return false;
    }
}
```

```

// Insertar en Documento
String sqlDocumento = "INSERT INTO Documento (autor, destinatario, fecha_creacion, palabra_clave, id_empleado) VALUES (?, ?, ?, ?, ?)";
PreparedStatement psDocumento = conexion.prepareStatement(
    sqlDocumento
);
psDocumento.setString(1, documento.getAutor());
psDocumento.setString(2, documento.getDestinatario());

String fechaCadDoc = sdf.format(documento.getFecha_creacion());

psDocumento.setString(3, fechaCadDoc);
psDocumento.setString(4, convertirListaAJson(documento.getPalabra_clave()));
psDocumento.setInt(5, personaId);

int documentoResultado = psDocumento.executeUpdate();
if (documentoResultado == 0) {
    conexion.rollback();
    return false;
}

```

```

// Confirmar la transacción
conexion.commit();
return true;

} catch (SQLException e) {
    try {
        if (conexion != null) {
            conexion.rollback();
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
    e.printStackTrace();
    return false;
} finally {
    try {
        if (conexion != null) {
            conexion.setAutoCommit(true);
            conexion.close();
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}
}

```

Excepción en la clase Repository para buscar por palabra clave en la BD:

```
public List<Documento> buscarPorPalabraClave(String palabra) {
    List<Documento> documentos = new ArrayList<>();
    Connection conexion = establecerConexion();
    if (conexion == null) {
        System.err.println("No se pudo establecer conexión con la base de datos.");
        return documentos;
    }
    try {
        String palabraJson = "\"" + palabra + "\"";
        PreparedStatement ps = conexion.prepareStatement("SELECT * FROM Documento WHERE JSON_CONTAINS(palabra_clave, ?)");
        SimpleDateFormat sdf = new SimpleDateFormat("dd-mm-yyyy");
        ps.setString(1, palabraJson);
        ResultSet rs = ps.executeQuery();
        System.out.println(rs);
        while (rs.next()) {
            Documento doc = new Documento();
            doc.setAutor(rs.getString("autor"));
            doc.setDestinatario(rs.getString("destinatario"));
            try {
                doc.setFecha_creacion(sdf.parse(rs.getString("fecha_creacion")));
            } catch (ParseException e) {
            }
            String palabraClaveJson = rs.getString("palabra_clave");
            List<String> palabraClaveList = convertirJsonALista(palabraClaveJson);
            System.out.println(palabraClaveList);
            doc.setPalabraClave(palabraClaveList);
            documentos.add(doc);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return documentos;
}
```

Excepción en la clase Repository para determinar el autor más productivo:

```
public String autorMasProductivo() {

    Connection conexion = establecerConexion();

    String sql = "SELECT e.id_empleado, e.nombre, COUNT(d.id_documento) AS cantidad_documentos "
        + "FROM Empleados e "
        + "JOIN Documento d ON e.id_empleado = d.id_empleado "
        + "GROUP BY e.id_empleado, e.nombre "
        + "ORDER BY cantidad_documentos DESC "
        + "LIMIT 1";

    try {
        PreparedStatement ps = conexion.prepareStatement(sql);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            int idEmpleado = rs.getInt("id_empleado");
            String nombreEmpleado = rs.getString("nombre");
            int cantidadDocumentos = rs.getInt("cantidad_documentos");

            String resultado = "Nombre: " + nombreEmpleado + "\n"
                + ", Cantidad de documentos confeccionados: " + cantidadDocumentos;
            return resultado;
        }
    } catch (Exception e) {
    }

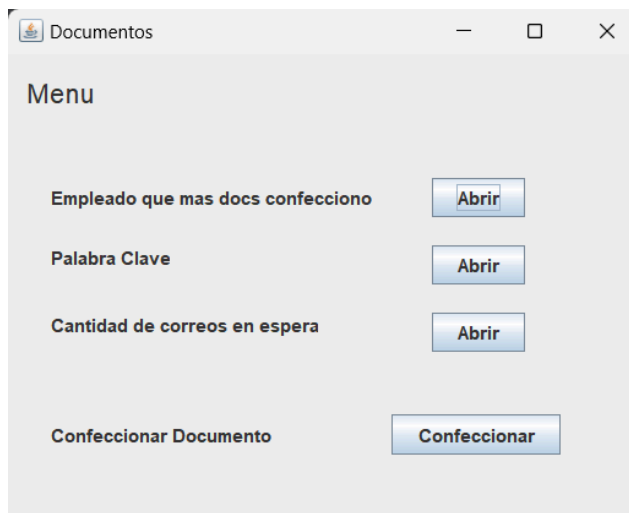
    return "No se encontraron resultados";
}
```

Excepción en la clase Repository para consultar los documentos que se encuentran en espera:

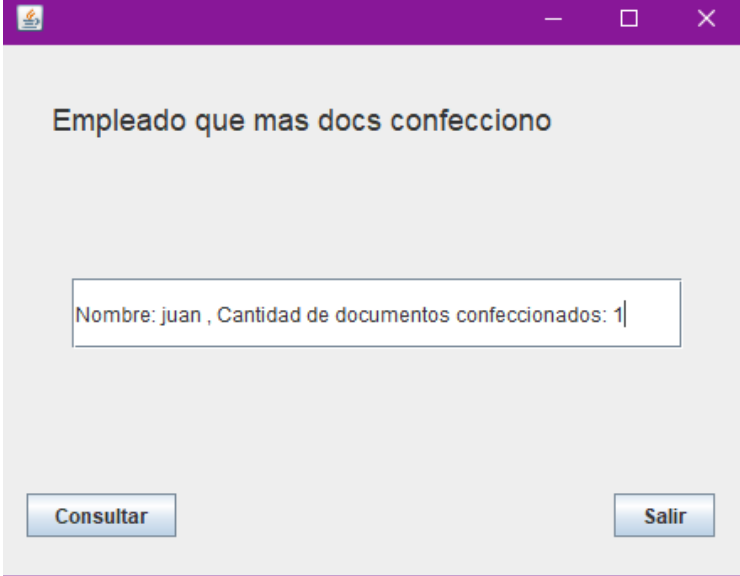
```
public Optional<Integer> cantidadEnEsperaConsulta() {  
    String sql = "SELECT COUNT(*) FROM Envio WHERE estado_enviado = true";  
    Connection conexion = establecerConexion();  
    try {  
        PreparedStatement ps = conexion.prepareStatement(sql);  
        ResultSet rs = ps.executeQuery();  
  
        if (rs.next()) {  
            int cantidad = rs.getInt(1);  
            return Optional.of(cantidad);  
        } else {  
            return Optional.empty();  
        }  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
    return Optional.empty();  
}
```

Vista:

Menú:

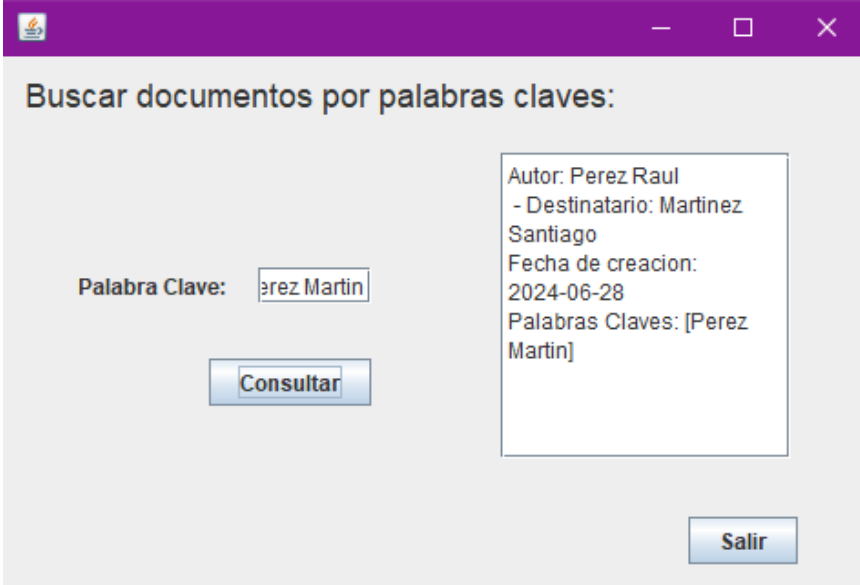


Método empleado que más confeccionó:



A screenshot of a software window with a purple title bar. The window title is "Empleado que mas docs confecciono". Inside the window, there is a text input field containing the text "Nombre: juan , Cantidad de documentos confeccionados: 1". Below the input field, there are two buttons: "Consultar" on the left and "Salir" on the right.

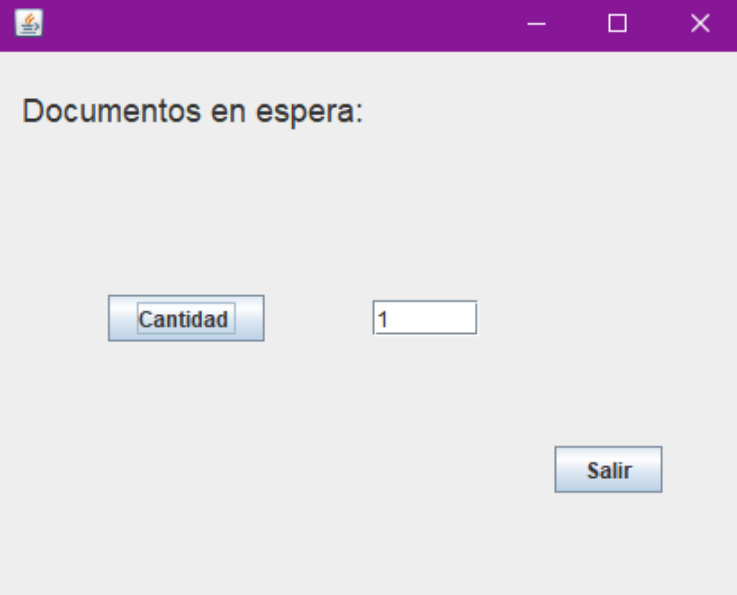
Método buscar por palabra clave:



A screenshot of a software window with a purple title bar. The window title is "Buscar documentos por palabras claves:". Inside the window, there is a text input field labeled "Palabra Clave:" containing the text "Perez Martin". Below the input field, there is a button labeled "Consultar". To the right of the input field, there is a text area displaying the following information: "Autor: Perez Raul", "- Destinatario: Martinez Santiago", "Fecha de creacion: 2024-06-28", and "Palabras Claves: [Perez Martin]". At the bottom right of the window, there is a button labeled "Salir".



Método documentos en espera:



A screenshot of a software window titled "Documentos en espera:". The window has a purple title bar with standard Windows controls. The main area is light gray. It contains a label "Cantidad" next to a text input field containing the number "1". At the bottom right, there is a button labeled "Salir".

Enviar Documentos:



A screenshot of a software window titled "Enviar Documentos". The window has a purple title bar. The main area is light gray and contains two sections: "Datos Trabajador:" and "Datos Documento:". The "Datos Trabajador:" section has five input fields with labels below them: "Nombre" (containing "juan"), "Telefono" (containing "26169488"), "Dirreccion" (containing "Barrio Jar"), "Fecha de ingreso" (containing "28/6/2024"), and "Cargo" (containing "Encagado"). The "Datos Documento:" section has three input fields: "Autor" (containing "Perez Ra"), "Destinatario" (containing "Martinez"), and "Palabras Claves" (containing "Perez Ma"). To the right of this section is the "Datos ente externo:" section with three input fields: "Nombre" (containing "Martinez"), "Direccion" (containing "Godoy cr"), and "Telefono" (containing "2615346"). At the bottom right, there are two buttons: "Limpiar" and "Siguiente".

Datos de la empresa de correo

Nombre

Correos :

Encargado

Rodrigue

Direccion

Mendoza

Telefono

2615234

Empresa encargada del envio

Nombre empresa

Andreani

Num° Seguimiento

4321

Estado Envio

Enviado

Terminar

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:

TA

	id_empleado	nombre	direccion	telefono	fecha_ingreso	cargo
▶	27	juan	Barrio Jardin	26169488	2024-06-28	Encagado
*	NULL	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