



**Proyecto Final, I, II, III, IV y V Fase – Gestión de Correo Electrónico**

5190-24-6840 Diego Emanuel Musús Reyes

Facultad de Ingeniería en Sistemas de Información y Ciencia de la Computación

Universidad Mariano Gálvez de Guatemala, Campus Villa Nueva

Programación II

Ing. Hernández Gabriel, Sebastián

25 de octubre de 2025

## Tabla de contenido

<b>MANUAL DE USUARIO.....</b>	<b>5</b>
<b>I FASE .....</b>	<b>5</b>
CONEXIÓN A LA BASE DE DATOS .....	5
EXPLORACIÓN DE TABLAS .....	6
CONSULTAS BÁSICAS.....	7
<b>II FASE.....</b>	<b>8</b>
<i>Conexión y despliegue en NetBeans .....</i>	<i>8</i>
<i>Operaciones disponibles.....</i>	<i>8</i>
<i>Pruebas del servicio.....</i>	<i>9</i>
<b>III FASE.....</b>	<b>12</b>
PANTALLA DE INICIO DE SESIÓN .....	12
MENSAJE DE ERROR DE AUTENTICACIÓN .....	12
PANTALLA DE MENÚ PRINCIPAL.....	13
OPCIONES DE USUARIO .....	13
OPCIONES DE REPORTES.....	14
CERRAR SESIÓN.....	14
<b>IV FASE.....</b>	<b>15</b>
<i>Pantalla de Inicio de Sesión (login.jsp).....</i>	<i>15</i>
<i>Menú Principal (menu.jsp) .....</i>	<i>15</i>
<i>Formulario de Inserción (nuevaEmpresa.jsp) .....</i>	<i>16</i>
<i>Pantalla de Mantenimiento (mantenimientoEmpresa.jsp) .....</i>	<i>16</i>
<i>Mensajes de Confirmación (SweetAlert) .....</i>	<i>17</i>
<i>Cerrar Sesión .....</i>	<i>18</i>
<b>V FASE .....</b>	<b>18</b>
<i>Generación de Reportes PDF.....</i>	<i>18</i>
<b>MANUAL TÉCNICO.....</b>	<b>20</b>
<b>I FASE .....</b>	<b>20</b>
HERRAMIENTAS UTILIZADAS.....	20
ARQUITECTURA DEL SISTEMA.....	20
CONEXIÓN A LA BASE DE DATOS .....	20
DIAGRAMAS DEL PROYECTO.....	21
DESPLIEGUE .....	23
<b>II FASE.....</b>	<b>23</b>
<i>Herramientas utilizadas.....</i>	<i>23</i>
<i>Configuración del servidor Payara .....</i>	<i>24</i>
<i>Configuración de la Base de Datos .....</i>	<i>24</i>
<i>Código del servicio JAX-WS.....</i>	<i>24</i>
<b>III FASE .....</b>	<b>25</b>

HERRAMIENTAS UTILIZADAS.....	25
ARQUITECTURA DEL SISTEMA.....	25
CONFIGURACIÓN DEL SERVIDOR.....	26
CONEXIÓN A LA BASE DE DATOS .....	26
<b>IV FASE.....</b>	<b>26</b>
<i>Código Clave .....</i>	26
<b>V FASE .....</b>	<b>30</b>

# Introducción

---

El proyecto “Gestión de Email” se desarrolló como parte del curso de Programación II, con el objetivo de aplicar los conocimientos adquiridos en el desarrollo de aplicaciones web utilizando Java EE, JSP, Servlets y Servicios Web JAX-WS.

Durante las diferentes fases, se construyó un sistema completo que gestiona entidades relacionadas con el envío y registro de correos electrónicos, incluyendo módulos para empresas, usuarios, bandejas, estados de email, destinatarios, recuperación de contraseñas e historial de ingresos.

A lo largo del desarrollo se abordaron los temas fundamentales del ciclo de vida de un sistema: diseño de base de datos, implementación de CRUD, integración con servicios web, configuración de servidor, validaciones, y finalmente la generación de reportes dinámicos en PDF usando iTextPDF 5.5.13.3.

Este manual incluye la descripción técnica del entorno de desarrollo, las herramientas utilizadas, los procedimientos de configuración y la descripción funcional de cada pantalla del sistema, facilitando tanto el uso como el mantenimiento del proyecto.

# Proyecto Gestión de Correo Electrónico

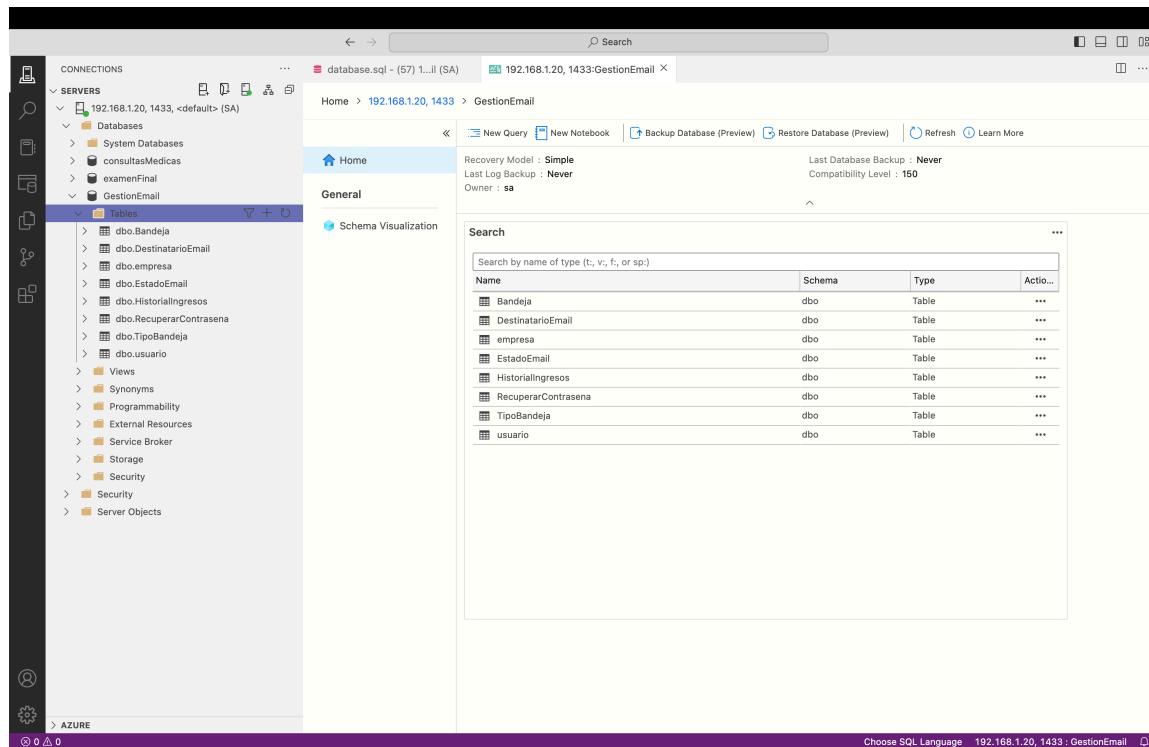
## Manual de Usuario

### I Fase

Este manual describe cómo utilizar la base de datos de Gestión de Correo Electrónico en su fase actual. Actualmente no existe una interfaz gráfica, por lo que todas las operaciones se realizan directamente sobre SQL Server mediante Azure Data Studio.

#### Conexión a la Base de Datos

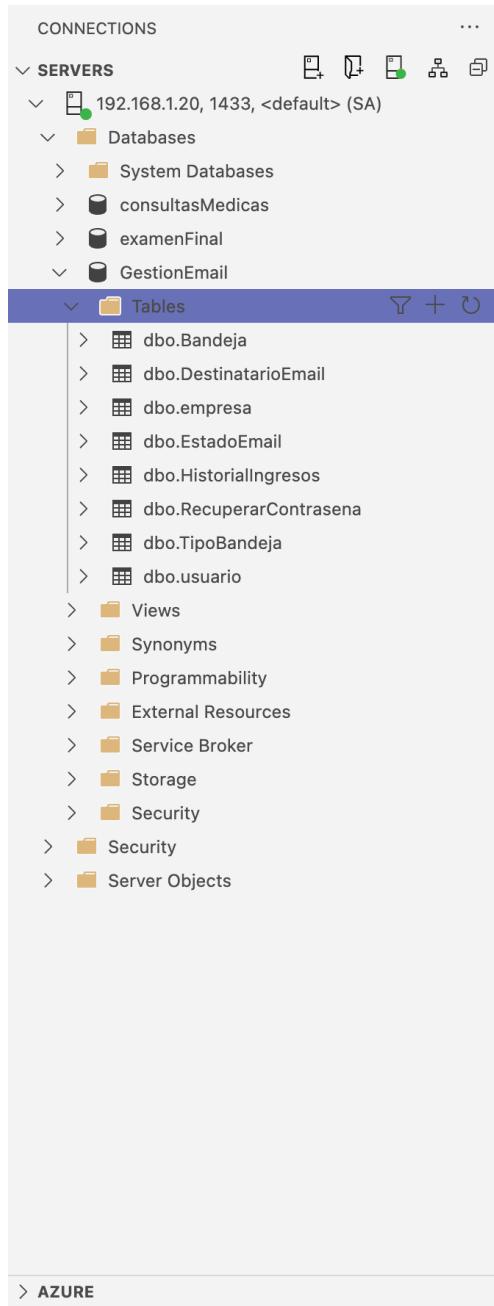
1. Abra Azure Data Studio.
2. En la ventana de conexión, ingrese los siguientes datos:
  - Servidor: 192.168.1.20,1433
  - Tipo de autenticación: SQL Login
  - Usuario: SA
  - Contraseña: Musus123
3. Presione 'Conectar'.
4. Una vez conectado, podrá ver la base de datos 'GestionEmail'.



## **Exploración de Tablas**

Dentro del explorador de objetos podrá visualizar las tablas:

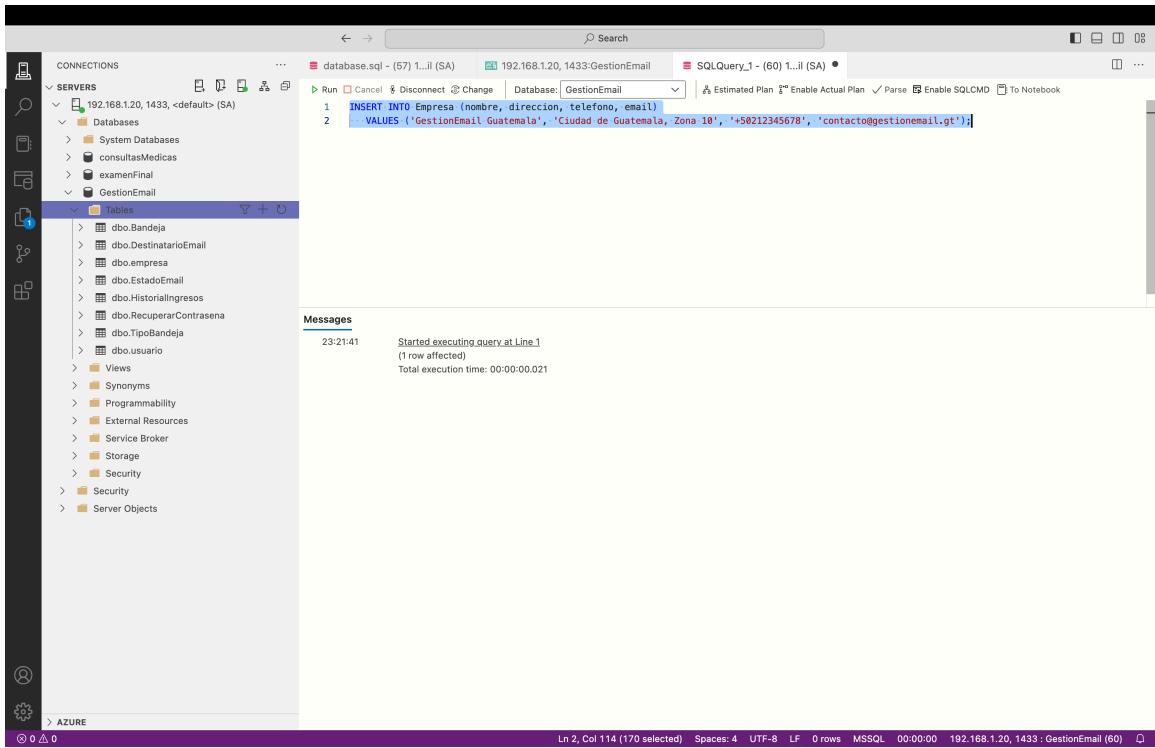
- Empresa
- Usuario
- RecuperarContrasena
- Bandeja
- HistorialIngresos
- DestinatarioEmail
- TipoBandeja
- EstadoEmail



## Consultas Básicas

Ejemplo de cómo registrar una nueva empresa en la base de datos:

```
INSERT INTO Empresa (nombre, direccion, telefono, email)
VALUES ('GestionEmail Guatemala', 'Ciudad de Guatemala, Zona 10', '+50212345678',
'contacto@gestionemail.gt');
```



## II Fase

Cómo utilizar el servicio web publicado en Payara y sus operaciones disponibles.

### Conexión y despliegue en NetBeans

1. Abra Apache NetBeans IDE 26.
2. Cree un proyecto del tipo 'Web Application' y configure Payara Server 5.2022.5 como servidor.
3. Incluya el archivo JDBC 'mssql-jdbc-11.2.0.jre11.jar' en las librerías del proyecto.
4. Genere el servicio web 'srvEmpresa' dentro del paquete 'wsServicios'.
5. Ejecute el proyecto, lo que desplegará el servicio en Payara en el puerto configurado.

URL del servicio:

<http://localhost:50413/wsGestionEmail/srvEmpresa?wsdl>

### Operaciones disponibles

- insertar(nombre, direccion, telefono, email): Registra una nueva empresa en la base de datos.
- actualizar(nombre, direccion, telefono, email, idEmpresa): Modifica una empresa existente.
- eliminar(idEmpresa): Elimina una empresa por su identificador.
- listar(): Retorna todas las empresas registradas.
- listarPorID(idEmpresa): Retorna los datos de una empresa específica.

## Pruebas del servicio

Para probar las operaciones, se puede acceder a la URL del WSDL en el navegador y posteriormente utilizar la opción 'Test Web Service' en NetBeans para enviar parámetros y visualizar la respuesta del servidor.

- URL del WSDL en el navegador.

## - Ejemplo de inserción de datos.

The screenshot shows the Payara Server Console interface with the following details:

**Method parameter(s)**

Type	Value
java.lang.String	Liceo Tecnico VN
java.lang.String	Zona 3, Villa Nueva
java.lang.String	47362837
java.lang.String	liceotecvn@gmail.com

**Method returned**

int : "1"

**SOAP Request**

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<S:Body xmlns:ns2="http://wsServicios/">
<ns2:insertar>
    <nombre>Liceo Tecnico VN</nombre>
    <direccion>Zona 3, Villa Nueva</direccion>
    <telefono>47362837</telefono>
    <email>liceotecvn@gmail.com</email>
</ns2:insertar>
</S:Body>
</S:Envelope>
```

## - Ejemplo de listado de empresas.

The screenshot shows the Payara Server Console interface with the following details:

**Method parameter(s)**

Type	Value
java.lang.String	Liceo Tecnico VN
java.lang.String	Zona 5, Villa Nueva
java.lang.String	66637876
java.lang.String	liceotecvn@hotmail.com
int	1004

**Method returned**

int : "1"

**SOAP Request**

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<S:Body xmlns:ns2="http://wsServicios/">
<ns2:actualizacion>
    <nombre>Liceo Tecnico VN</nombre>
    <direccion>Zona 5, Villa Nueva</direccion>
    <telefono>66637876</telefono>
    <email>liceotecvn@hotmail.com</email>
    <idEmpresa>1004</idEmpresa>
</ns2:actualizacion>
</S:Body>
</S:Envelope>
```

## - Ejemplo de actualización y eliminación.

**listar Method invocation**

**Method parameter(s)**

Type	Value

**Method returned**

```
java.util.List : "[wservicios.Empresa@551c7306, wservicios.Empresa@7d79cd4, wservicios.Empresa@2fc7bd2]"
```

**SOAP Request**

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<S:Body xmlns:ns2="http://wsServicios/">
<ns2:listar/>
</S:Body>
</S:Envelope>
```

**SOAP Response**

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<S:Body xmlns:ns2="http://wsServicios/">
<ns2:listarResponse>
<return>
<direccion>Ciudad de Guatemala, Zona 10</direccion>
<email>contacto@gestionemail.gq</email>
<idEmpresa>1</idEmpresa>
<nombreGestionEmail>Guatemala</nombre>
<telefono>+502212345678</telefono>
</return>
<return>
<direccion>Zona 4, Mz direcccion</direccion>
<email>lauica@gmail.com</email>
<idEmpresa>2</idEmpresa>
<nombreProveedor> La Áñica</nombre>
<telefono>34543</telefono>
</return>
<return>
<direccion>Zona 3, Villa Nueva</direccion>
<email>lizettecv@gmail.com</email>
<idEmpresa>1004</idEmpresa>
<nombreProveedor> LIZETE</nombre>
<telefono>47362837</telefono>
</return>
</ns2:listarResponse>
</S:Body>
</S:Envelope>
```

**eliminar Method invocation**

**Method parameter(s)**

Type	Value
int	1004

**Method returned**

```
int : "1"
```

**SOAP Request**

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<S:Body xmlns:ns2="http://wsServicios/">
<ns2:eliminar>
<idEmpresa>1004</idEmpresa>
</ns2:eliminar>
</S:Body>
</S:Envelope>
```

## III Fase

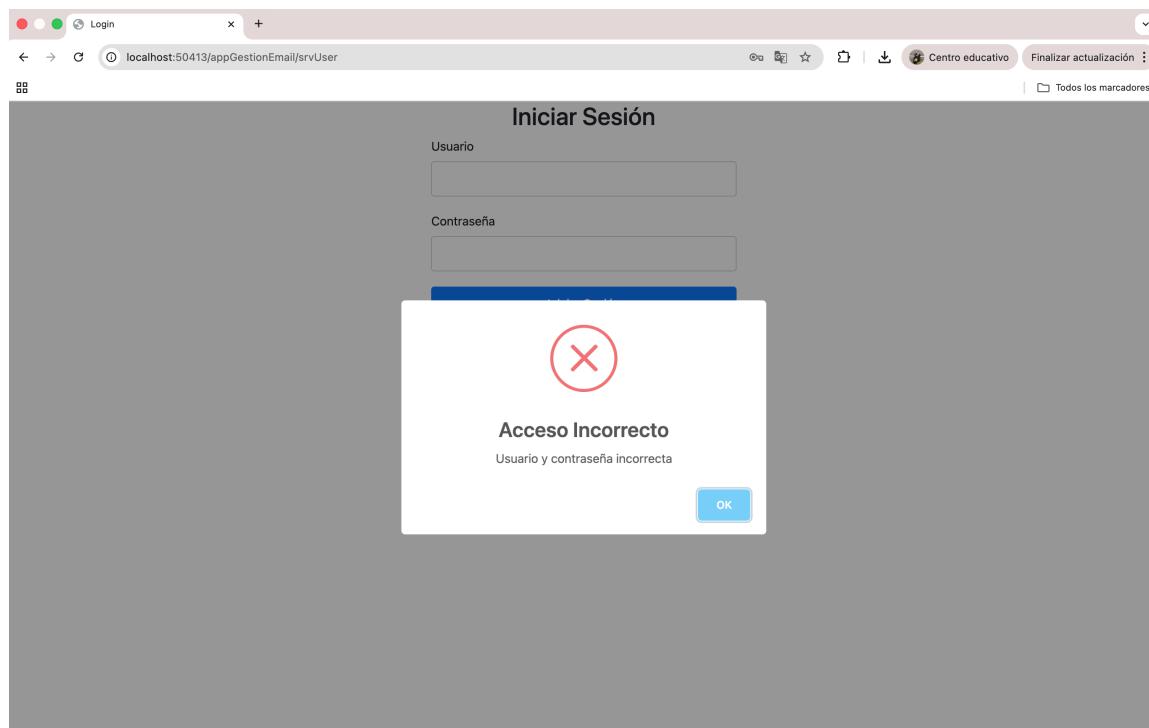
### Pantalla de Inicio de Sesión

El sistema inicia mostrando la pantalla de inicio de sesión (login.jsp). Aquí el usuario debe ingresar su nombre de usuario y contraseña para acceder al sistema.



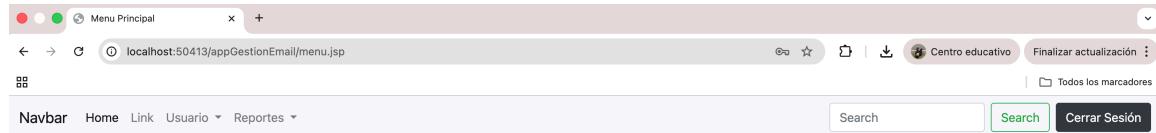
### Mensaje de Error de Autenticación

Si el usuario ingresa credenciales incorrectas, el sistema muestra un mensaje de error mediante SweetAlert.



## Pantalla de Menú Principal

Al iniciar sesión correctamente, se redirige al menú principal (menu.jsp), donde se presentan las opciones de navegación del sistema.



## Opciones de Usuario

Desde el menú 'Usuario' se pueden realizar operaciones de mantenimiento como registrar, actualizar, eliminar o buscar usuarios.



## Opciones de Reportes

En el menú 'Reportes' se encuentran las opciones de consulta y visualización de reportes relacionados con usuarios y el sistema.



## Cerrar Sesión

En la parte superior derecha se encuentra el botón 'Cerrar Sesión', que permite al usuario salir del sistema y regresar a la pantalla de login.

The image contains two screenshots of a web browser. The top screenshot shows the "Menu Principal" page with the "Cerrar Sesión" button highlighted by a red arrow. The bottom screenshot shows the "Login" page with the "Iniciar Sesión" button.

## IV Fase

### Pantalla de Inicio de Sesión (login.jsp)

Permite al usuario ingresar su nombre de usuario y contraseña para acceder al sistema. A continuación, se muestra la interfaz de inicio de sesión.



The screenshot shows a browser window with the title 'Login'. The address bar displays 'localhost:50413/appGestionEmail/'. The main content area is titled 'Iniciar Sesión' and contains two input fields: 'Usuario' and 'Contraseña', followed by a blue 'Iniciar Sesión' button.

### Menú Principal (menu.jsp)

Muestra todas las opciones de navegación del sistema (Empresa, Usuario, Bandeja, etc.). Cada opción permite acceder a los formularios CRUD.



## Formulario de Inserción (nuevaEmpresa.jsp)

Permite registrar nuevos datos en el sistema. Debajo se muestra un ejemplo del formulario utilizado para registrar una empresa.

The screenshot shows a web browser window titled 'Nueva Empresa'. The URL is 'localhost:50413/appGestionEmail/nuevaEmpresa.jsp'. The page header includes 'Gestion de Email' and various dropdown menus. A central modal dialog is titled 'Formulario de Registro'. It contains four input fields: 'Nombre' (Gmail), 'Dirección' (Guatemala), 'Teléfono' (1111-1111), and 'Correo electrónico' (email@gmail.com). Below the fields is a 'Registrar' button.

## Pantalla de Mantenimiento (mantenimientoEmpresa.jsp)

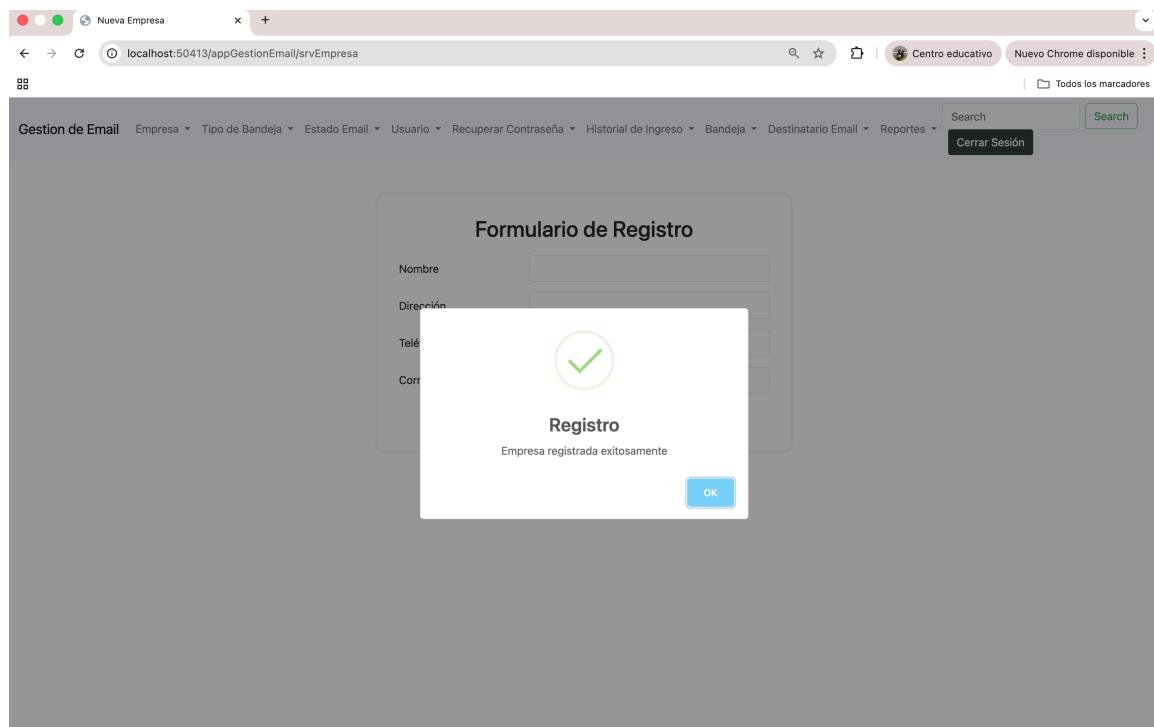
Muestra los registros almacenados con opciones para editar o eliminar cada uno. Facilita la gestión de los datos existentes.

The screenshot shows a web browser window titled 'Mantenimiento de Empresa'. The URL is 'localhost:50413/appGestionEmail/mantenimientoEmpresa.jsp'. The page header includes 'Gestion de Email' and various dropdown menus. A search bar at the top has a placeholder 'ID de la Empresa' and a 'Buscar' button. Below is a table with the following data:

ID	Nombre	Dirección	Teléfono	Email	Acciones
1	GestionEmail	Ciudad de Guatemala, Zona 15	+50212345678	contacto@gestionemail.com	<button>Actualizar</button> <button>Eliminar</button>
1002	Proveedor La Ñanica	Zona 4, VN	34543	launica@gmail.com	<button>Actualizar</button> <button>Eliminar</button>
3003	prueba	1234	111111	prueba@gmail.com	<button>Actualizar</button> <button>Eliminar</button>
4002	Gmail	Guatemala	1111-1111	email@gmail.com	<button>Actualizar</button> <button>Eliminar</button>

## Mensajes de Confirmación (SweetAlert)

Cuando una operación se ejecuta correctamente, se muestra un mensaje de confirmación mediante la librería SweetAlert.



## Cerrar Sesión

El usuario puede cerrar sesión desde el menú superior, lo que invalida la sesión activa y redirige al inicio de sesión.

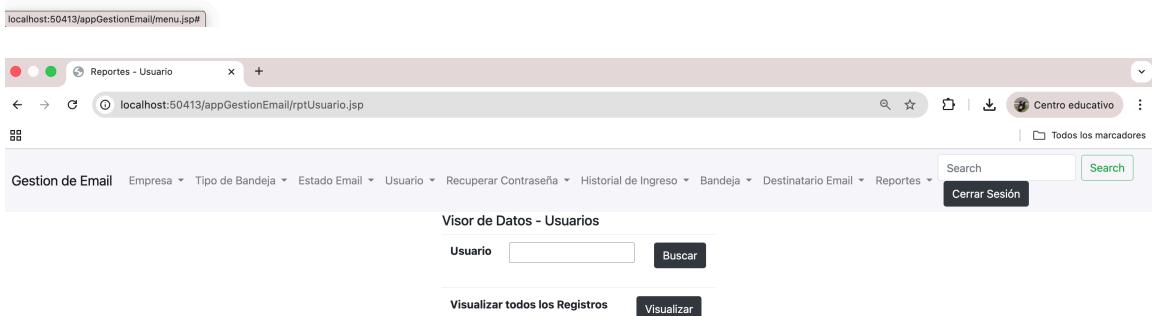
The top screenshot displays a table of company data with columns for ID, Nombre, Dirección, Teléfono, Email, and Acciones. The 'Acciones' column contains 'Actualizar' and 'Eliminar' buttons. A red arrow points to the 'Cerrar Sesión' button in the top right corner of the page header. The bottom screenshot shows a login form with fields for 'Usuario' and 'Contraseña', and a blue 'Iniciar Sesión' button.

## V Fase

### Generación de Reportes PDF

En esta fase se implementó la generación de reportes en formato PDF empleando la librería iTextPDF. Los reportes permiten visualizar la información almacenada en las ocho tablas principales del sistema: Empresa, Usuario, Bandeja, HistorialIngresos, DestinatarioEmail, TipoBandeja, EstadoEmail y RecuperarContrasena. Cada reporte puede ser consultado de forma individual o total desde el menú principal del sistema.

El usuario puede acceder a los reportes desde la opción 'Reportes' del menú principal. Cada sección cuenta con dos formularios: uno para consultar por ID específico y otro para visualizar todos los registros.



Al presionar el botón 'Visualizar' o 'Buscar', el sistema invoca el servlet correspondiente que genera un archivo PDF con los datos solicitados. El archivo puede descargarse e imprimirse directamente desde el navegador.

Usuario	idEmpresa	Nombre	Apellido	Email	Contraseña	Fecha de Registro	Estado
damarisaleta	1	damaris	h	damaris@email.com	234	2025-09-2	activo
dmususr	1	Emanuel	Reyes	dmusus@gmail.com	23	2025-10-0	Activo
musus	1	diego	musus	dmmusus@gmail.com	musus	2025-09-2	Activo

## Manual Técnico

Este manual describe las herramientas, arquitectura y detalles técnicos empleados en la primera, segunda y tercera fase del proyecto de Gestión de Correo Electrónico.

## I Fase

### Herramientas Utilizadas

- Motor de base de datos: SQL Server (en contenedor Docker)
- Cliente: Azure Data Studio
- Diagrama ER: Draw.io
- Diagrama UML: Lucidchart

### Arquitectura del Sistema

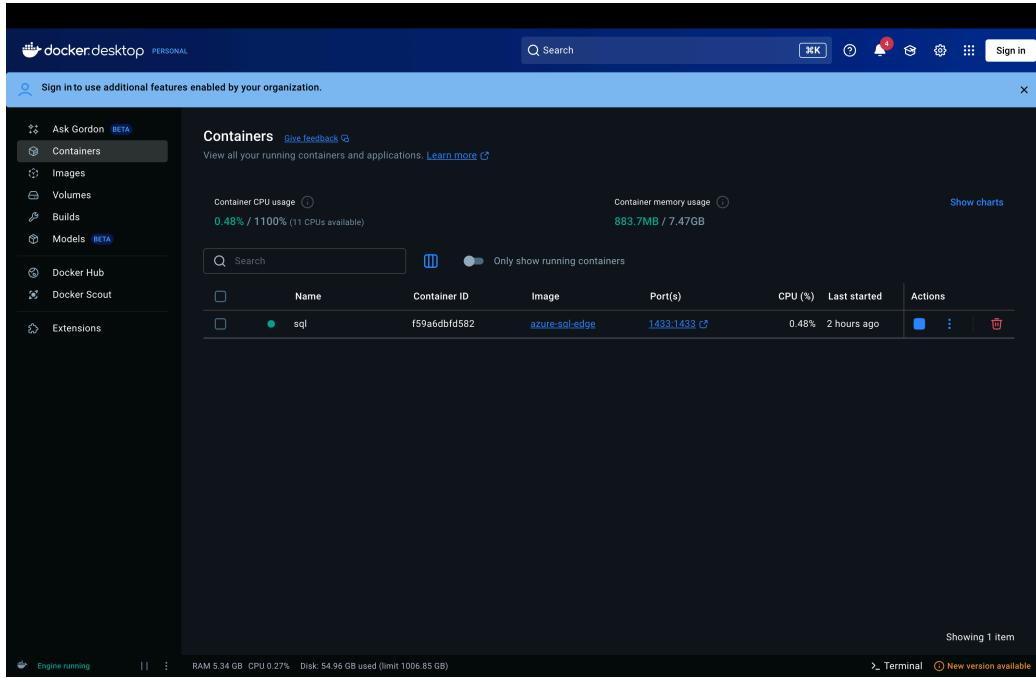
Actualmente, el proyecto solo consta de una base de datos relacional implementada en SQL Server. La aplicación se encuentra en fase inicial.

### Conexión a la Base de Datos

Para conectarse a la base de datos se debe ingresar la IP de la máquina host (Macbook) y el puerto expuesto del contenedor Docker.

Datos de conexión:

- Servidor: IP de la máquina, 1433
- Usuario: SA
- Contraseña: Musus123



## Diagramas del Proyecto

El diseño conceptual y lógico del sistema está representado mediante los diagramas siguientes:

- Diagrama ER (Entidad–Relación) creado en Draw.io

## Gestión de email

**Tablas detalle-PK, FK**

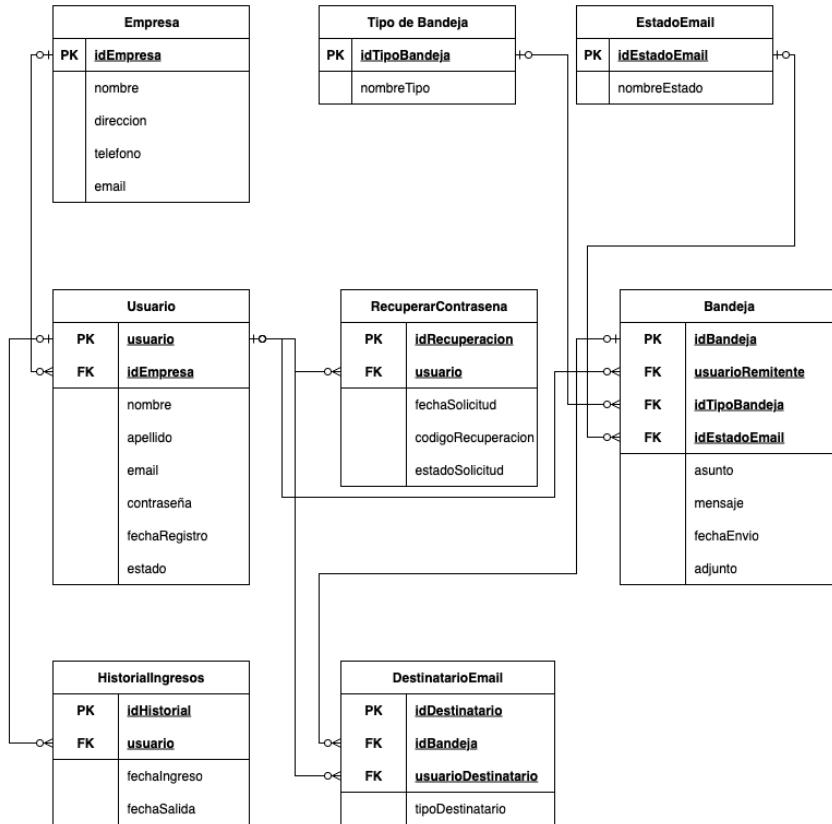
- Usuario (PK empresa)
- Recuperar contraseña (FK usuario)
- Bandeja (FK usuario, tipo de bandeja, estado de email, empresa)
- Historial de registros (FK usuario)
- DestinatarioEmail (FK bandeja, FK usuario)

**Tablas**

Usuario	✓
Recuperar contraseña	✓
Empresa	✓
Tipo de bandeja	✓
Estados de email	✓
Bandeja	✓
Historial de registros	✓
DestinatarioEmail	✓

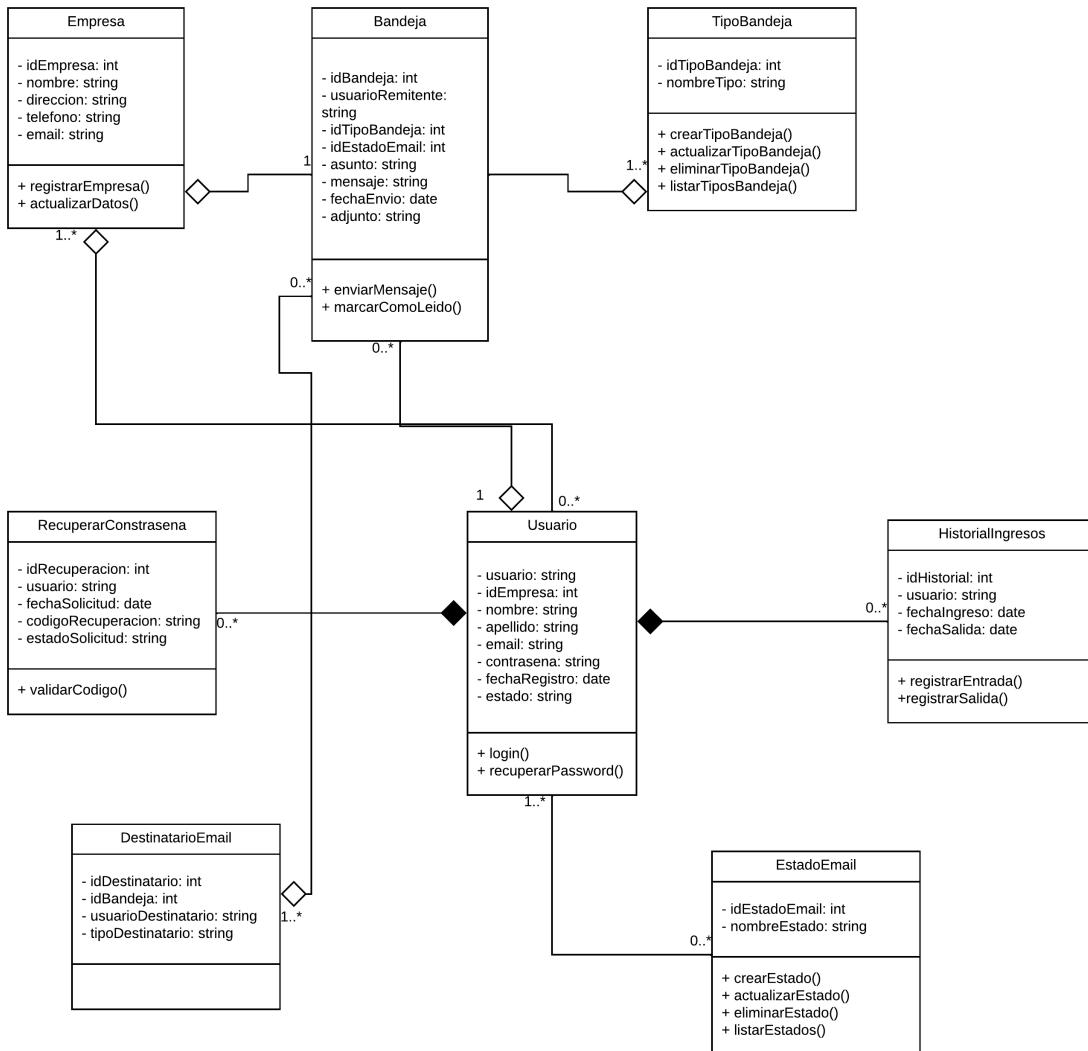
**Catálogos-PK**

- Empresa
- Tipo de bandeja
- Estados de email



PK - PK (uno a uno)  
 PK - FK (uno a varios)  
 FK - FK (varios a varios)

- Diagrama UML de clases creado en Lucidchart



## Despliegue

La base de datos se encuentra actualmente desplegada únicamente en un contenedor Docker local sobre un equipo Macbook. El acceso se limita a la red local mediante la IP y puerto configurados.

## II Fase

### Herramientas utilizadas

- Lenguaje: Java (JDK 11.0.28)
- IDE: Apache NetBeans IDE 26
- Servidor de aplicaciones: Payara Server 5.2022.5

- Base de datos: Microsoft Azure SQL Edge Developer (SQL Server) en contenedor Docker
- Driver JDBC: mssql-jdbc-11.2.0.jre11.jar

### Configuración del servidor Payara

- DAS Port: 50412
- HTTP Port: 50413

El servidor Payara se configuró en NetBeans con los puertos anteriores y se utilizó para publicar el servicio SOAP.

### Configuración de la Base de Datos

Base de datos: GestiónEmail

Tabla: empresa

Campos: idEmpresa, nombre, direccion, telefono, email

Cadena de conexión JDBC:

```
jdbc:sqlserver://localhost:1433;databaseName=GestionEmail;encrypt=false;trustServerCertificate=true
Driver Class: com.microsoft.sqlserver.jdbc.SQLServerDriver
```

### Código del servicio JAX-WS

```
@WebService(serviceName = "srvEmpresa")
public class srvEmpresa {

    @WebMethod(operationName = "insertar")
    public int insertar(String nombre, String direccion, String telefono, String email) {
        csEmpresa e = new csEmpresa();
        return e.insertar(nombre, direccion, telefono, email);
    }

    @WebMethod(operationName = "actualizacion")
    public int actualizar(String nombre, String direccion, String telefono, String email, int idEmpresa){
        csEmpresa e = new csEmpresa();
        return e.actualizar(nombre, direccion, telefono, email, idEmpresa);
    }

    @WebMethod(operationName = "eliminar")
    public int eliminar(int idEmpresa){
        csEmpresa e = new csEmpresa();
        return e.eliminar(idEmpresa);
    }
}
```

```

@WebMethod(operationName = "listar")
public ArrayList<empresa> listar() {
    csEmpresa e = new csEmpresa();
    return e.listarEmpresa();
}

@WebMethod(operationName = "listarPorID")
public empresa listarPorID(int idEmpresa){
    csEmpresa e = new csEmpresa();
    return e.listarEmpresaPorID(idEmpresa);
}

```

## III Fase

### **Herramientas Utilizadas**

- Lenguaje: Java (JDK 11)
- IDE: Apache NetBeans IDE 26
- Servidor de aplicaciones: Payara Server 5.2022.5
- Base de datos: SQL Server en contenedor Docker
- Librerías: Bootstrap, SweetAlert

### **Arquitectura del Sistema**

En la Fase III se extendió la arquitectura MVC con la incorporación de la capa de persistencia y validación:

- JSP: continúa como interfaz de usuario.
- Servlets: además de controlar la lógica, ahora gestionan validaciones de datos antes de enviarlos a la base.
- Servicios Web JAX-WS: se implementaron métodos para consultas por identificador y listados completos.
- Base de datos SQL Server: se consolidó como almacenamiento central de la información con tablas y relaciones activas.

## **Configuración del Servidor**

Se mantiene la ejecución en Payara Server 5.2022.5, pero en la Fase III se añadieron configuraciones adicionales:

- Ajustes para manejar errores y excepciones al comunicarse con la base de datos.
- Pruebas de despliegue enfocadas en operaciones CRUD.

## **Conexión a la Base de Datos**

Nuevas operaciones implementadas:

- Inserción con validación de datos.
- Consulta de registros por ID.
- Listado de todos los registros.
- Actualización con control de errores.

## **IV Fase**

### **Código Clave**

A continuación se muestran fragmentos representativos de código utilizados en la fase IV, incluyendo Servlets, y servlets en el archivo pom.xml.

#### **Servlet srvEmpresa.java:**

```
package Controlador;

import Modelo.empresa.SrvEmpresa_Service;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
/**  
 * Servlet encargado de manejar la inserción de nuevas empresas.  
 * Se comunica con el servicio web (SrvEmpresa_Service) para registrar  
 * una empresa en la base de datos.  
 */  
  
public class srvEmpresa extends HttpServlet {  
  
    /**  
     * Método principal para procesar las peticiones (GET y POST).  
     * Obtiene los datos del formulario, llama al servicio web y reenvía la respuesta.  
     */  
  
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
  
        response.setContentType("text/html;charset=UTF-8");  
  
        try (PrintWriter out = response.getWriter()) {  
            RequestDispatcher rd = null;  
  
            // Declaración de variables para los campos del formulario  
            String nombre, direccion, telefono, email;  
            int respuesta;  
  
            // Captura de datos del formulario
```

```
nombre = request.getParameter("txtNombre").toString();
direccion = request.getParameter("txtDireccion").toString();
telefono = request.getParameter("txtTelefono").toString();
email = request.getParameter("txtEmail").toString();

// No instanciamos un modelo, porque el modelo en este caso es el servicio web
// Instanciamos el servicio web que contiene los métodos de inserción
SrvEmpresa_Service empresa = new SrvEmpresa_Service();

// Llamada al método del servicio web para insertar la empresa
// Devuelve un entero (1 = éxito, 0 = error)
respuesta = empresa.getSrvEmpresaPort().insertar(nombre, direccion, telefono,
email);

// Guardamos la respuesta en un atributo para enviarla a la vista
request.setAttribute("respuesta", respuesta);

// Redirigimos la respuesta a la vista (nuevaEmpresa.jsp)
rd = request.getRequestDispatcher("nuevaEmpresa.jsp");
rd.forward(request, response);
}

}

/** 
 * Maneja el método GET
*/

```

```

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Maneja el método POST
 */
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Información corta del servlet
 */
@Override
public String getServletInfo() {
    return "Servlet para insertar nuevas empresas";
}

```

#### **Servlet en el archivo pom.xml:**

<!-- srvEmpresa -->

```

<execution>
    <id>srvEmpresa-client</id>
    <phase>generate-sources</phase>
    <goals><goal>wsimport</goal></goals>
    <configuration>
        <wsdlUrls>
            <wsdlUrl>http://localhost:50413/wsGestionEmail/srvEmpresa?wsdl</wsdlUrl>
        </wsdlUrls>
        <packageName>Modelo.empresa</packageName>
        <sourceDestDir>${project.build.directory}/generated-sources/jaxws</sourceDestDir>
        <xnocompile>true</xnocompile>
    </configuration>
</execution>

```

## V Fase

Para la implementación técnica de la generación de reportes, se empleó la librería iTextPDF versión 5.5.13.3. Esta librería permite crear documentos PDF dinámicos desde código Java, con control sobre formato, fuentes, tablas y estilos.

El servlet 'srvPdfEstado' se encarga de procesar las solicitudes desde las vistas JSP. Este obtiene los datos mediante el servicio web 'SrvEstadoEmail\_Service', crea un documento PDF con la lista de registros y lo envía como respuesta al navegador.

```

package Controlador;

import Modelo.estadoemail.EstadoEmail;
import Modelo.estadoemail.SrvEstadoEmail_Service;
import com.itextpdf.text.BaseColor;
import com.itextpdf.text.Chunk;
import com.itextpdf.text.Document;
import com.itextpdf.text.Element;
import com.itextpdf.text.Font;
import com.itextpdf.text.FontFactory;
import com.itextpdf.text.PageSize;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.Phrase;
import com.itextpdf.text.pdf.PdfPCell;
import com.itextpdf.text.pdf.PdfPTable;
import com.itextpdf.text.pdf.PdfWriter;

```

```

import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class srvPdfEstado extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        //Tipo de respuesta que necesitamos,
        response.setContentType("application/pdf");
        //Salida de datos, un documento en PDF
        OutputStream out = response.getOutputStream();

        try{
            try{
                Date date = new Date(); //capturamos el dia, mes y año
                SimpleDateFormat dia = new SimpleDateFormat("dd");
                SimpleDateFormat mes = new SimpleDateFormat("MM");
                SimpleDateFormat anio = new SimpleDateFormat("yyyy");

                float left = 10;
                float right = 10;
                float top = 10;
                float bottom = 10;

                //Palabra reservada que permite declarar el documento
                Document documento = new Document(PageSize.LETTER, left, right, top, bottom);
                PdfWriter.getInstance(documento, out);
                documento.open();
                documento.setMargins(left, right, top, bottom); //colocamos el margen del documento

                //Representamos la informacion
                Paragraph par1 = new Paragraph();
                Font titulo = new Font(Font.FontFamily.HELVETICA, 16, Font.BOLD, BaseColor.BLUE);
                par1.add(new Phrase("LISTADO DE ESTADOS DE EMAILS", titulo));
                par1.setAlignment(Element.ALIGN_CENTER);
                par1.add(new Phrase(Chunk.NEWLINE));
                par1.add(new Phrase(Chunk.NEWLINE));
                documento.add(par1);

                Paragraph par2 = new Paragraph();
                Font descripcion = new Font(Font.FontFamily.TIMES_ROMAN, 16, Font.NORMAL, BaseColor.BLACK);
                par2.add(new Phrase("Guatemala, " + dia.format(date) + " del " + mes.format(date) + " de " + anio.format(date),
                descripcion));
                par2.setAlignment(Element.ALIGN_CENTER);
                par2.add(new Phrase(Chunk.NEWLINE));
                par2.add(new Phrase(Chunk.NEWLINE));
                documento.add(par2);

                //Mostramos los datos en una tabla
                PdfPTable tabla = new PdfPTable(2); //asignamos la cantidad de columnas
                PdfPCell celda1 = new PdfPCell(new Paragraph("idEstadoEmail", FontFactory.getFont("Arial", 12, Font.BOLD,
                BaseColor.BLACK)));
                PdfPCell celda2 = new PdfPCell(new Paragraph("nombreEstado", FontFactory.getFont("Arial", 12, Font.BOLD,
                BaseColor.BLACK)));
                //Le agregamos a la tabla cada una de las celdas
                tabla.addCell(celda1);
                tabla.addCell(celda2);

                //Instanciamos el servicio
                SrvEstadoEmail_Service estado = new SrvEstadoEmail_Service();

                //Colocamos una validacion
                if(request.getParameter("btnBuscarIdEstado")!=null){

```

```

EstadoEmail e = new EstadoEmail();
String est = request.getParameter("txtIdEstado");
e = estado.getSrvEstadoEmailPort().listarPorID(Integer.parseInt(est));
tabla.addCell(String.valueOf(e.getIdEstadoEmail()));
tabla.addCell(e.getNombreEstado());
}else{
List<EstadoEmail> lista;
lista = estado.getSrvEstadoEmailPort().listar();

for(int i = 0; i<lista.size(); i ++){
    tabla.addCell(String.valueOf(lista.get(i).getIdEstadoEmail()));
    tabla.addCell(lista.get(i).getNombreEstado());
}
}

documento.add(tabla);
documento.close();

}catch(Exception ex){
    ex.getMessage();
}

}finally{
    out.close();
}
}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
/**
 * Handles the HTTP <code>GET</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Handles the HTTP <code>POST</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}

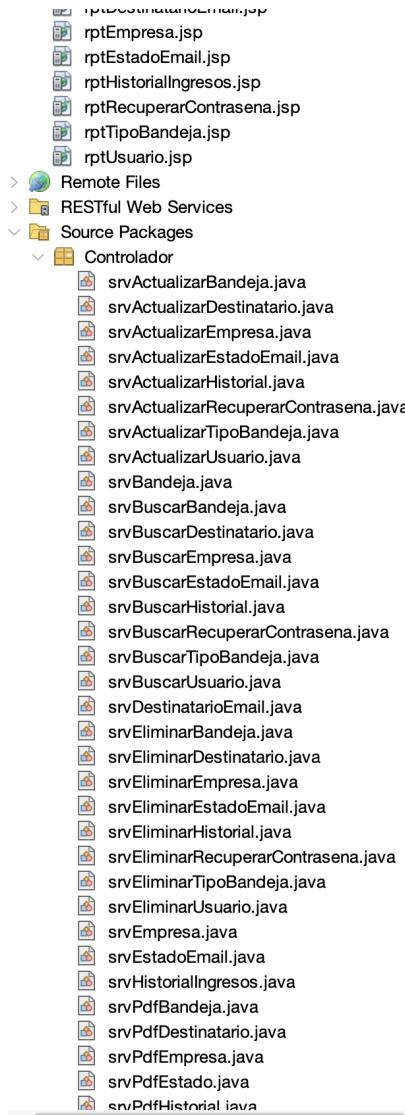
/**
 * Returns a short description of the servlet.
 *
 * @return a String containing servlet description
 */
@Override
public String getServletInfo() {
    return "Short description";
}// </editor-fold>

}

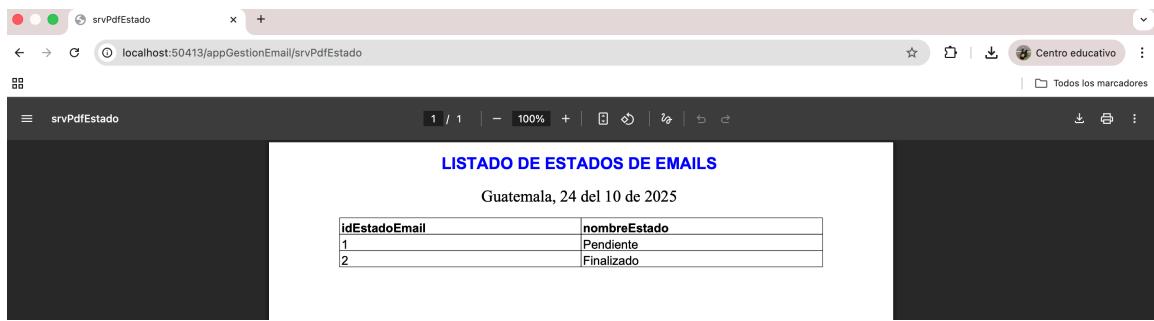
```

El documento PDF incluye encabezado, fecha actual, título centrado y una tabla con los campos correspondientes. El formato de salida es de tamaño carta con márgenes ajustados para una presentación limpia y profesional.

La estructura del proyecto mantiene la división en capas: 'Modelo' para los servicios, 'Controlador' para los servlets y 'Vista' para los archivos JSP. La conexión al servicio JAX-WS se realiza mediante la clase proxy generada en el paquete 'Modelo.estadoemail'.



Durante las pruebas, los reportes se generaron correctamente y pudieron visualizarse o descargarse desde el navegador. Se validó el funcionamiento para todas las entidades del sistema.



## Conclusión

---

A lo largo de las cinco fases del proyecto 'Gestión de Correo Electrónico', se logró construir un sistema web completo y funcional. Desde la definición de la base de datos hasta la integración de reportes PDF, se aplicaron conceptos clave de programación orientada a objetos, desarrollo web y servicios JAX-WS. La experiencia permitió fortalecer habilidades técnicas en el uso de Java EE, SQL Server, iTextPDF y el servidor Payara. Con la culminación de la fase V, el sistema ofrece una solución robusta que centraliza la gestión de información y proporciona reportes dinámicos y exportables, cumpliendo con los objetivos del curso de Programación II.