

Manual técnico – Módulo de Recursos Humanos

Carlos Felipe Avilán Payares

Laura Katherine Aviles Reina

Juan Diego Pinzón Fontecha

Ingeniería de Software

Universidad Manuela Beltrán

Ingeniería Web II

Carlos Eduardo Mujica Reyes

2025

1. **Información General**

El módulo de Recursos Humanos forma parte de un sistema ERP (Enterprise Resource Planning) desarrollado para la gestión de empleados. Permite registrar, actualizar, consultar y eliminar información del personal mediante una interfaz web construida con JSF y una base de datos relacional en PostgreSQL.

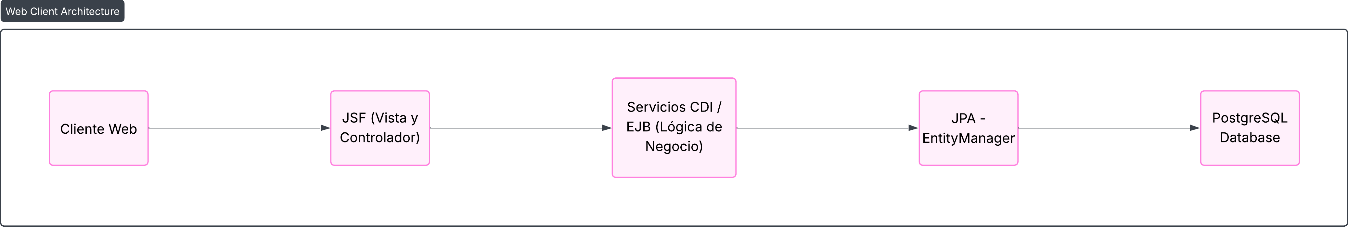
**2. Arquitectura del Sistema**

Arquitectura utilizada: Modelo multicapa (Presentación – Negocio – Persistencia).

Componentes principales:

* Capa de Presentación: Implementada con Jakarta Server Faces (JSF 4.0) y Facelets.
* Capa de Negocio: Implementada con CDI (Contexts and Dependency Injection) y clases de servicio.
* Capa de Persistencia: Implementada con Jakarta Persistence (JPA 3.1) conectada a PostgreSQL.
* Servidor de aplicaciones: WildFly 30 (compatible con Jakarta EE 10).
* Base de datos: PostgreSQL 15.
* Gestor de construcción: Apache Maven.

Diagrama general:



1. **Requisitos del Sistema**

Requisitos de hardware:

* CPU: mínimo 2 núcleos
* Memoria RAM: 4 GB (recomendado: 8 GB)
* Espacio en disco: 2 GB disponibles

Requisitos de software:

* JDK: Java 17 o superior
* Maven: Versión 3.8 o superior
* WildFly: Versión 30 o superior
* PostgreSQL: Versión 15
* IDE recomendado: IntelliJ IDEA Ultimate o Eclipse IDE for Enterprise Java Developers
* Sistema operativo: Windows, Linux o macOS

1. **Instalación y Configuración del Entorno**

**1. Instalar el JDK (Java Development Kit):**

Descargar desde Eclipse Temurin (OpenJDK).

Verificar la instalación con: java -version

**2. Instalar Apache Maven:**

Descargar desde la página oficial de Maven.

Agregar la ruta /bin a la variable de entorno PATH.

Verificar con: mvn -v

**3. Instalar WildFly:**

Descargar el ZIP desde el sitio oficial.

Descomprimir y ejecutar el servidor con: standalone.bat o ./standalone.sh

**4. Instalar PostgreSQL:**

Configurar el usuario postgres y su contraseña.

Crear la base de datos erp\_hr\_db.

Crear la tabla employees

1. **Configuración del Proyecto**

1. Creación del proyecto con Maven:

2. Estructura de paquetes:

3. Archivo de configuración persistence.xml

4. Configuración del DataSource en WildFly:

Nombre: ErpHrDS

JNDI: java:jboss/datasources/ErpHrDS

URL: jdbc:postgresql://localhost:5432/erp\_hr\_db

Usuario: postgres

Contraseña: (la definida en la instalación)

1. **Componentes Principales del Código**

**Entidad (JPA)**

Employee.java – Representa la tabla employees.

**Servicio de Negocio (CDI)**

EmployeeService.java – Gestiona la lógica de persistencia y transacciones.

**Controlador / Backing Bean**

EmployeeBean.java – Interactúa entre la interfaz JSF y el servicio.

**Interfaz de Usuario (JSF)**

index.xhtml – Permite realizar operaciones CRUD sobre empleados.

1. **Compilación y Despliegue**

1. Compilar y empaquetar

2. Desplegar en WildFly

3. Acceso a la aplicación

1. **Pruebas y Verificación**

Pruebas básicas de funcionamiento:

* Crear empleado → Verificar registro en la tabla.
* Editar empleado → Confirmar actualización en la base de datos.
* Eliminar empleado → Confirmar eliminación.
* Visualización de la lista → Validar renderizado dinámico JSF.

1. **Mantenimiento y Actualización**

1. Para actualizar el sistema, modificar el código fuente y ejecutar:

mvn clean package

Luego reemplazar el archivo .war en WildFly.

2. Para hacer copias de seguridad, exportar la base de datos:

pg\_dump erp\_hr\_db > backup\_erp\_hr.sql

1. **Control de Versiones**

Repositorio: GitHub

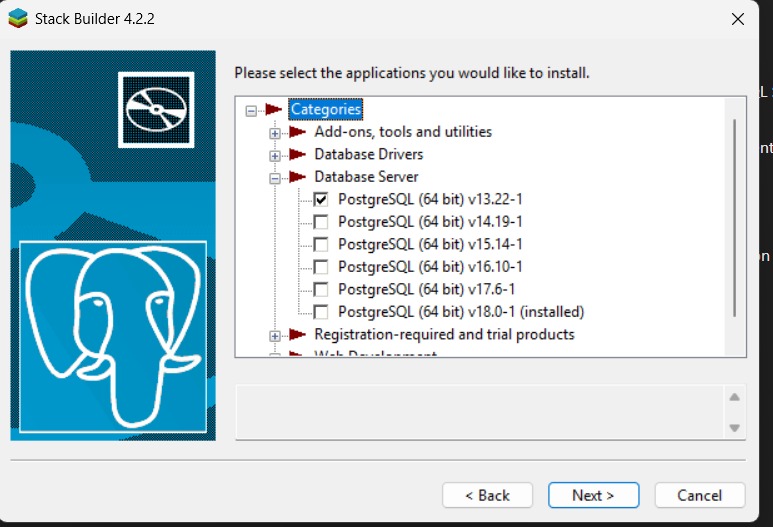
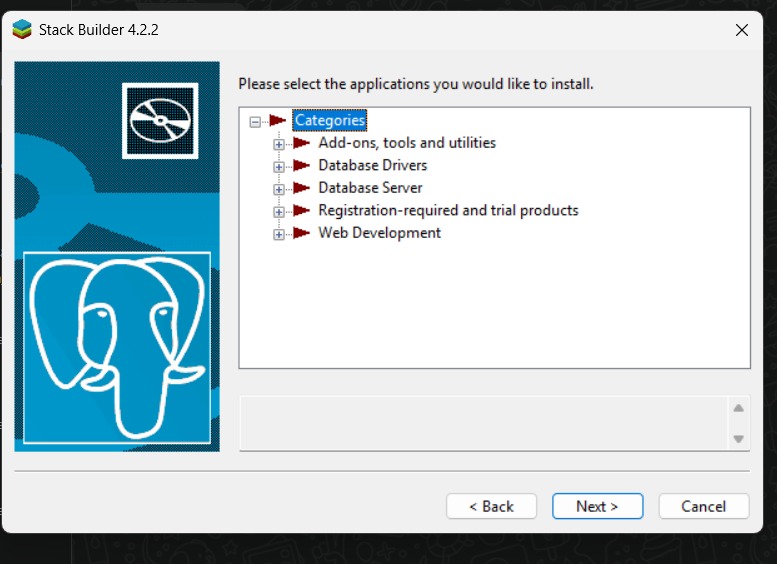
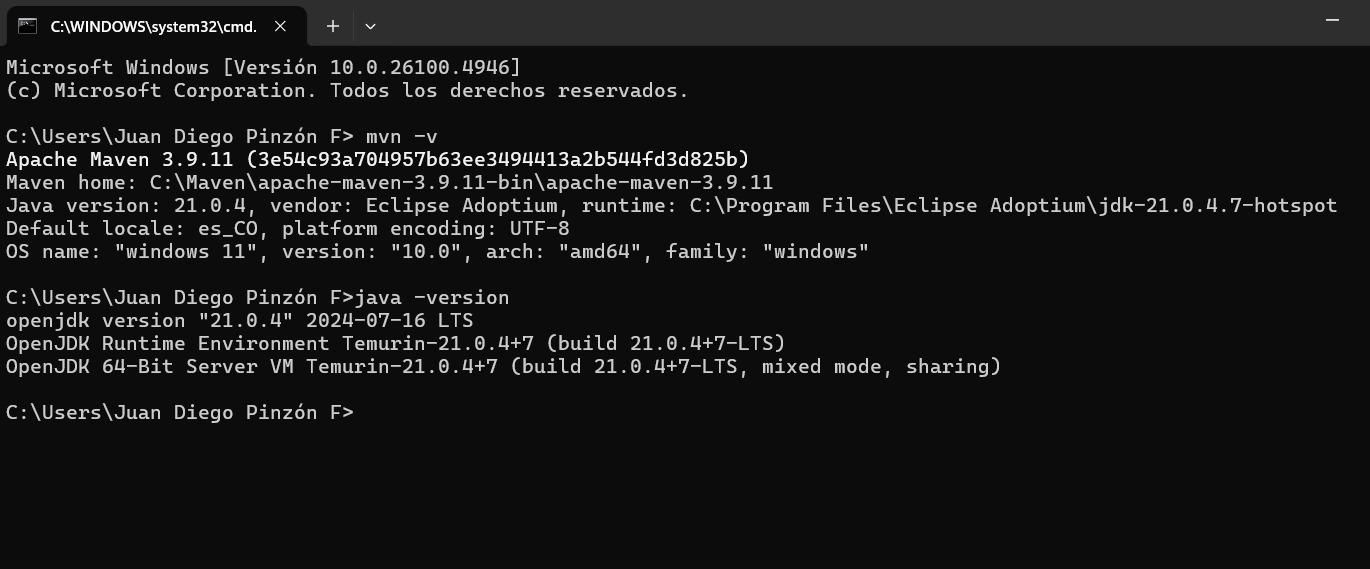
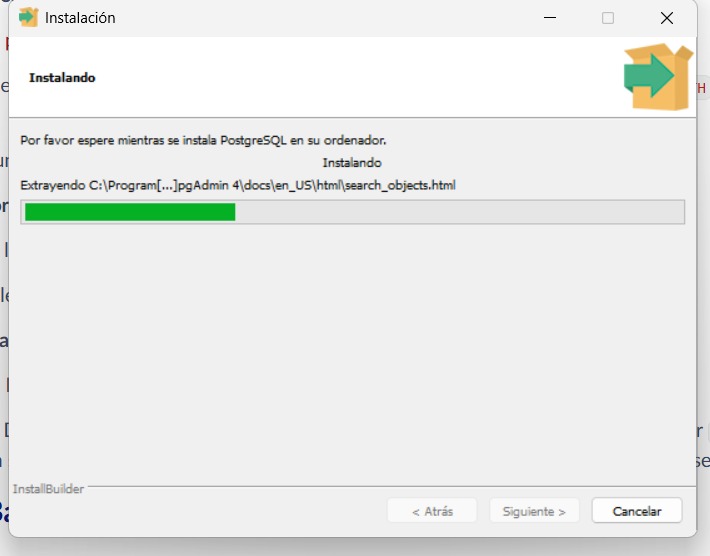
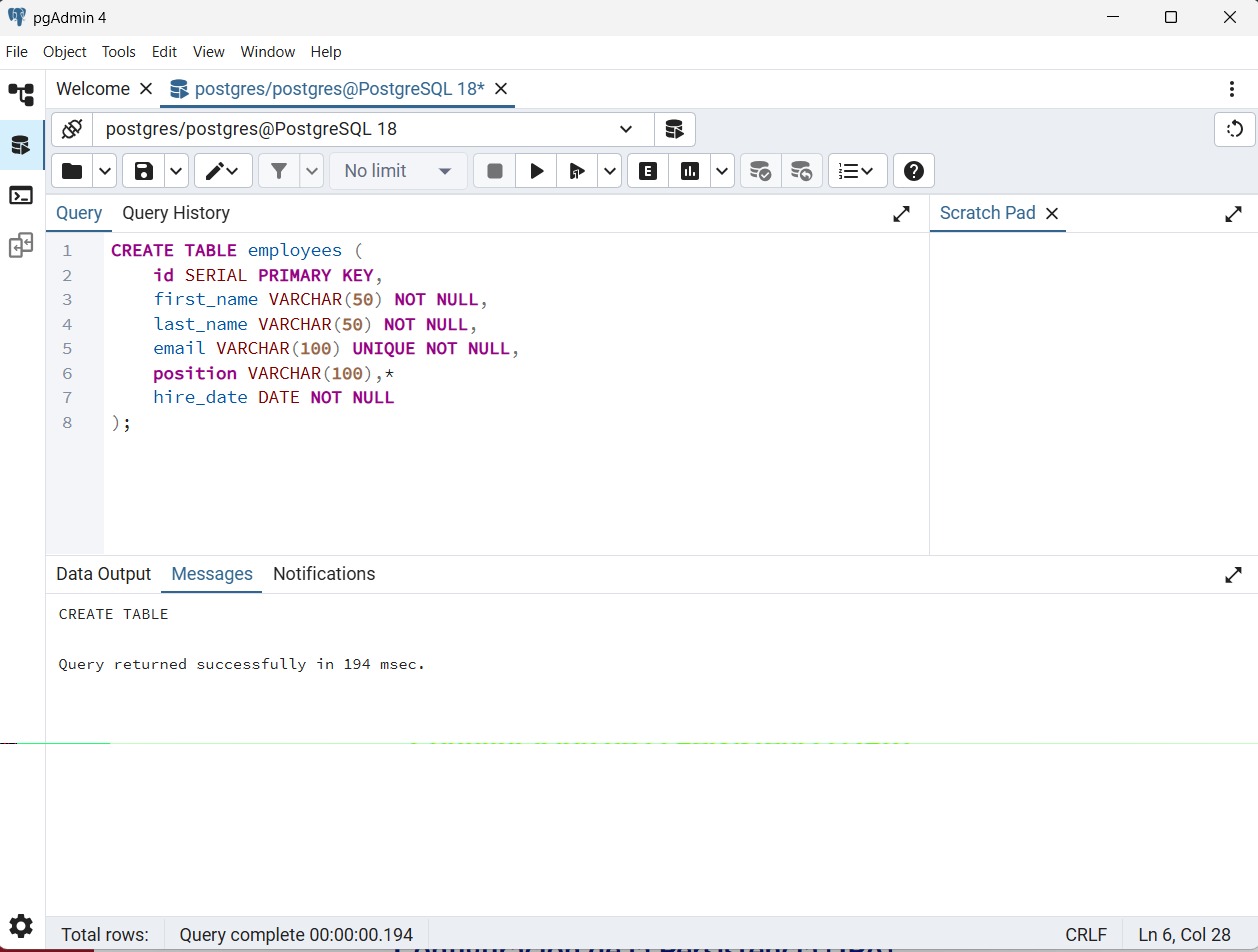
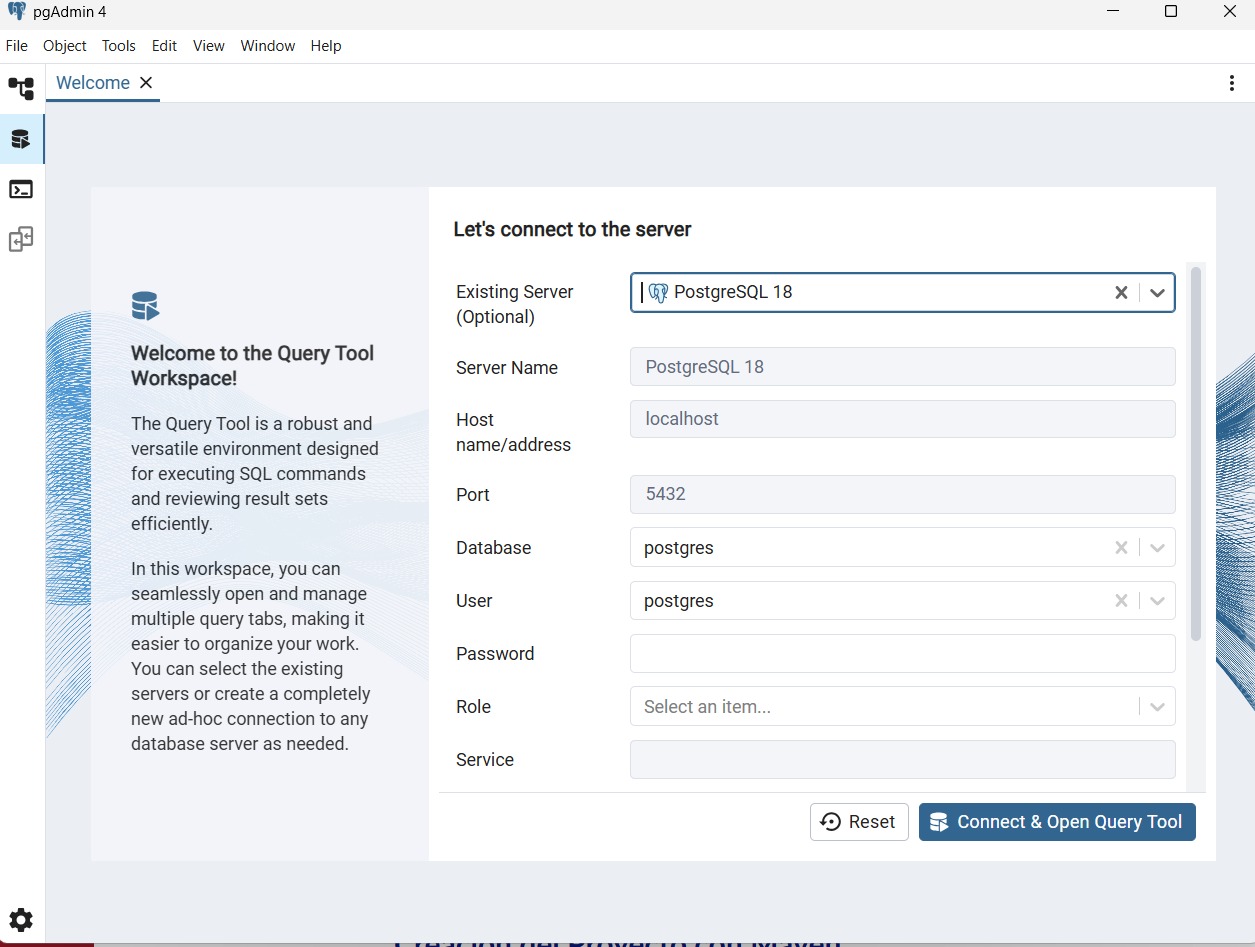
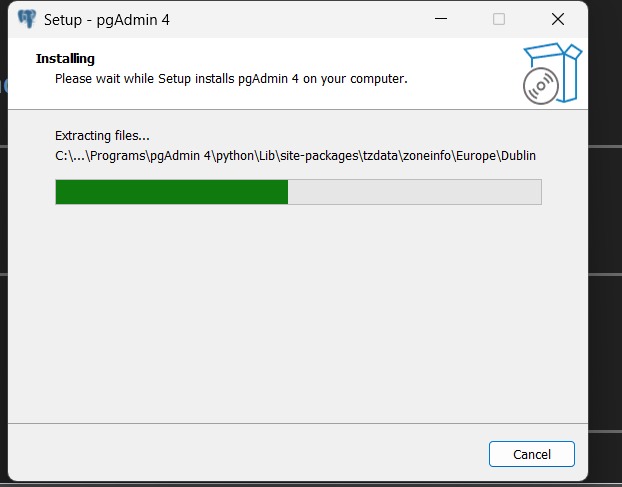
Comandos básicos:

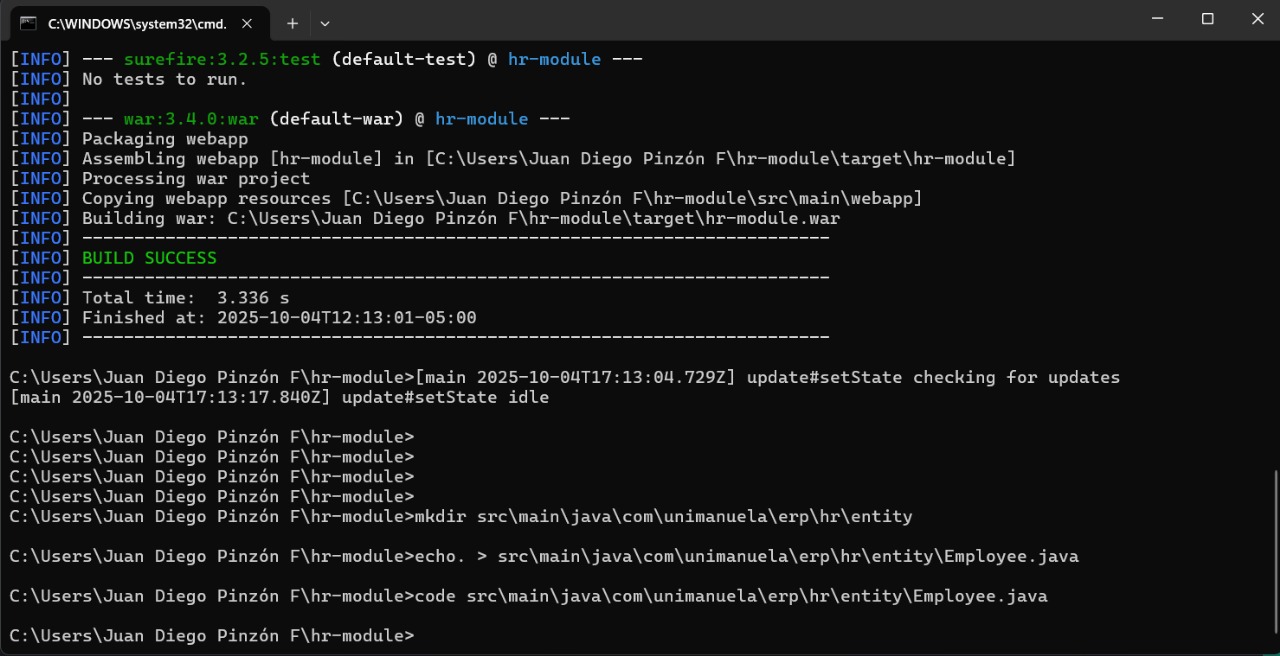
git init

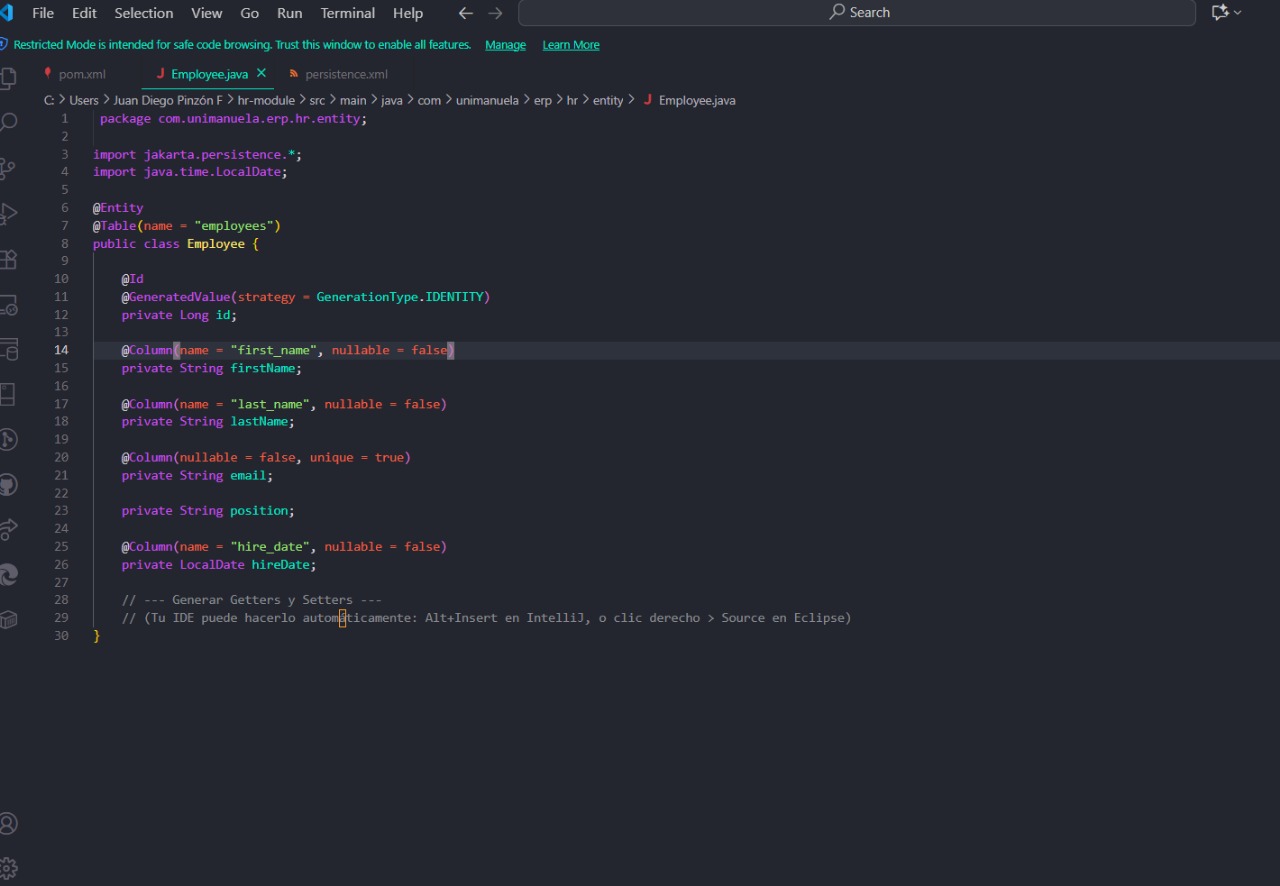
git add .

git commit -m "Versión inicial del módulo HR"

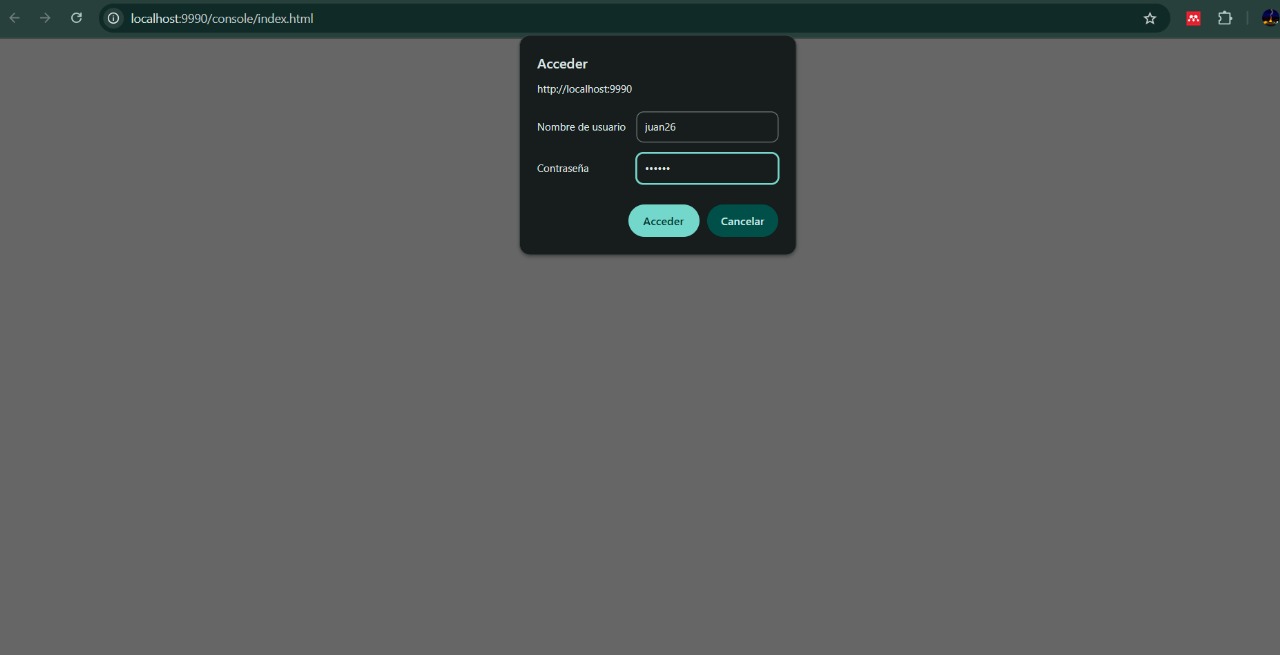
1. **Anexos**

****Se está ejecutando el comando mvn archetype:generate para crear un proyecto Java con Maven. El comando incluye parámetros como groupId, artifactId, archetypeArtifactId, y archetypeVersion. Maven está descargando dependencias y plugins desde el repositorio central (repo.maven.apache.org), como maven-install-plugin y maven-plugins, para completar la configuración del proyecto.

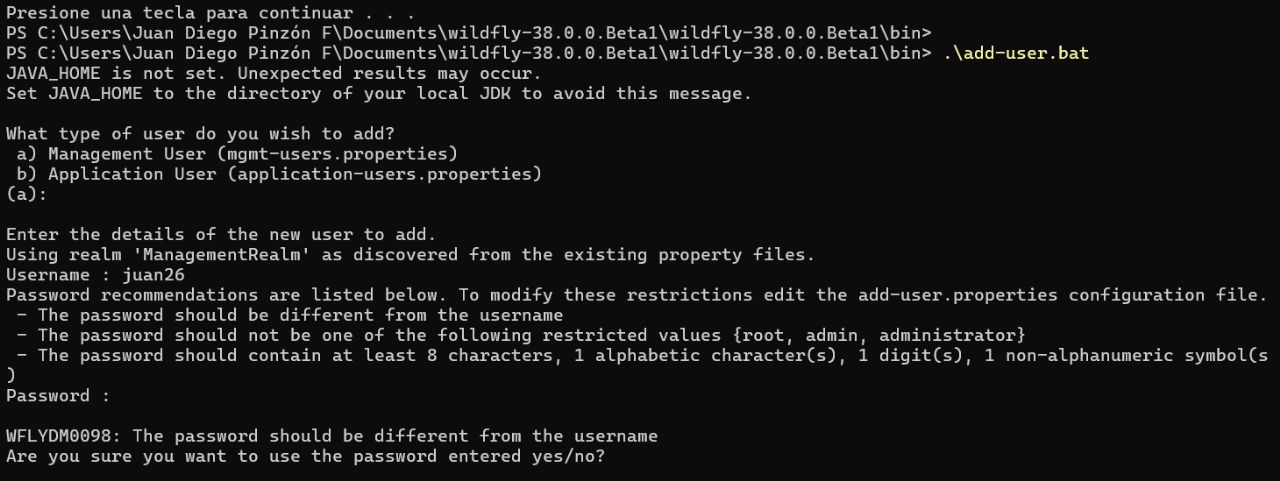


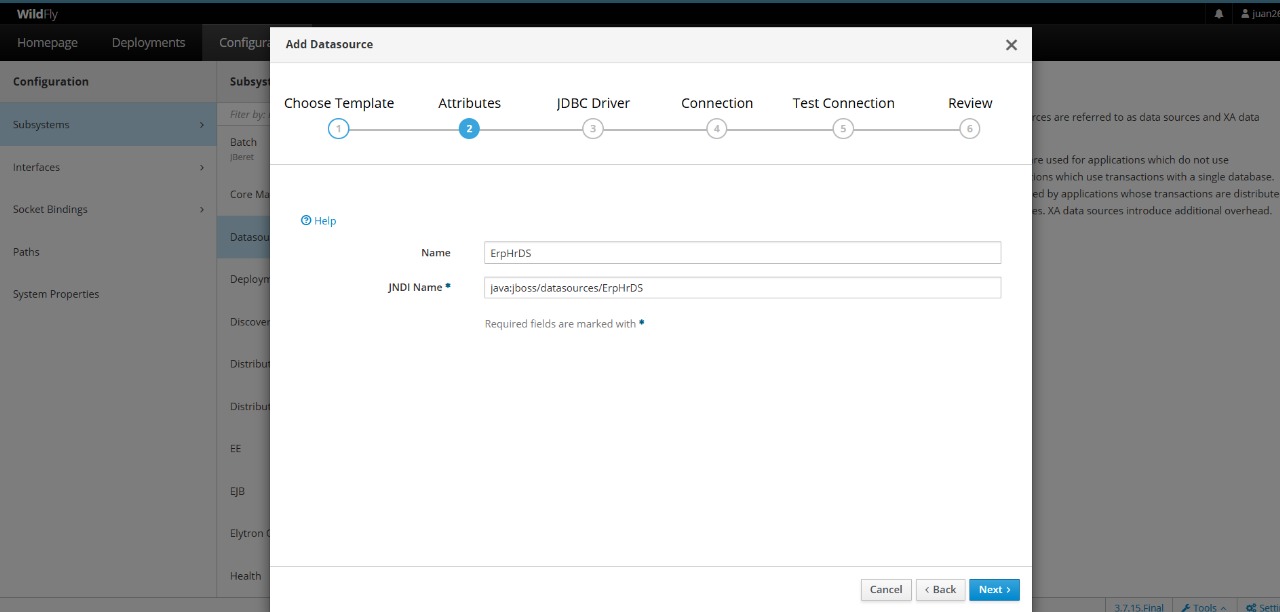
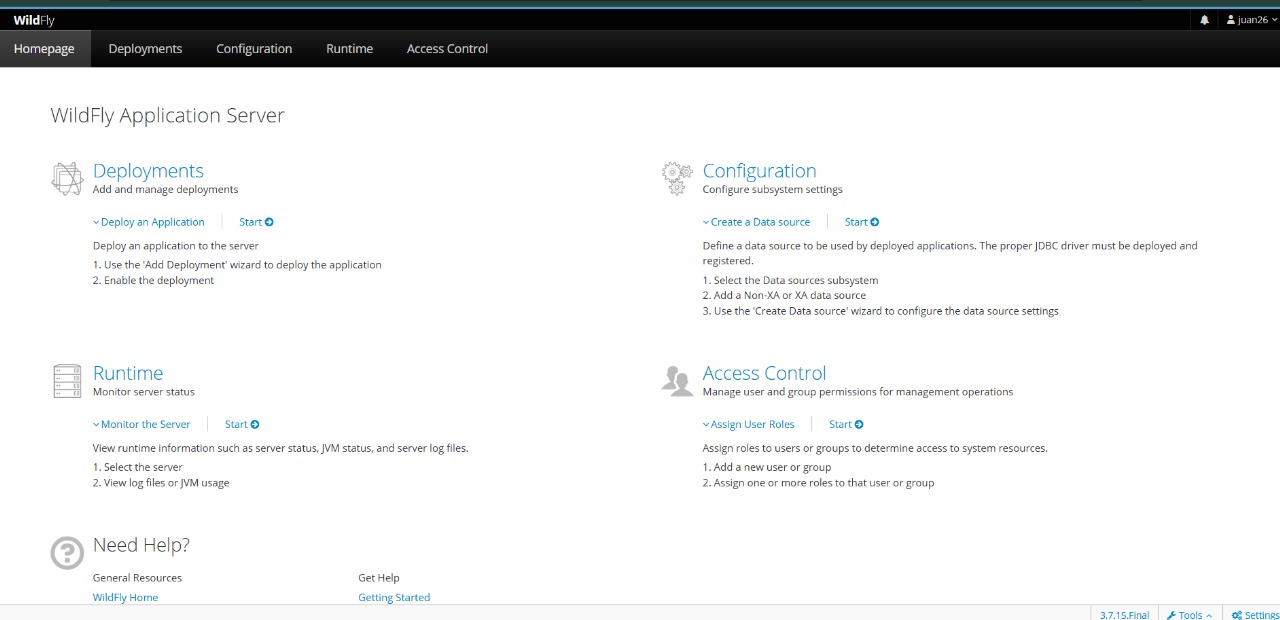
El proyecto Maven se ha compilado correctamente. Se observa el mensaje BUILD SUCCESS, lo que indica que la construcción fue exitosa. El proceso incluyó tareas como empaquetar el proyecto en un archivo WAR (para aplicaciones web), copiar recursos y generar el archivo final en la carpeta target. Después del build, se ven comandos para navegar por directorios y abrir el archivo Employee.java dentro del paquete com.unimanuel.erp.hr.entity.

Se está trabajando en una clase llamada Employee anotada con @Entity, lo que indica que es una entidad JPA. Contiene atributos como id, firstName, lastName, email, position y hireDate, cada uno con anotaciones @Column que especifican detalles como nombre de columna y restricciones (nullable = false, unique = true). También se observa la anotación @Id y @GeneratedValue(strategy = GenerationType.IDENTITY) para la clave primaria.



Se está accediendo a una interfaz web local en http://localhost:9990/console/index.html. Aparece un cuadro de diálogo para iniciar sesión con campos para Nombre de usuario (valor: juan26) y Contraseña (oculta). Hay dos botones: Acceder y Cancelar. Esto indica que se está intentando entrar al panel de administración o consola de una aplicación desplegada localmente.



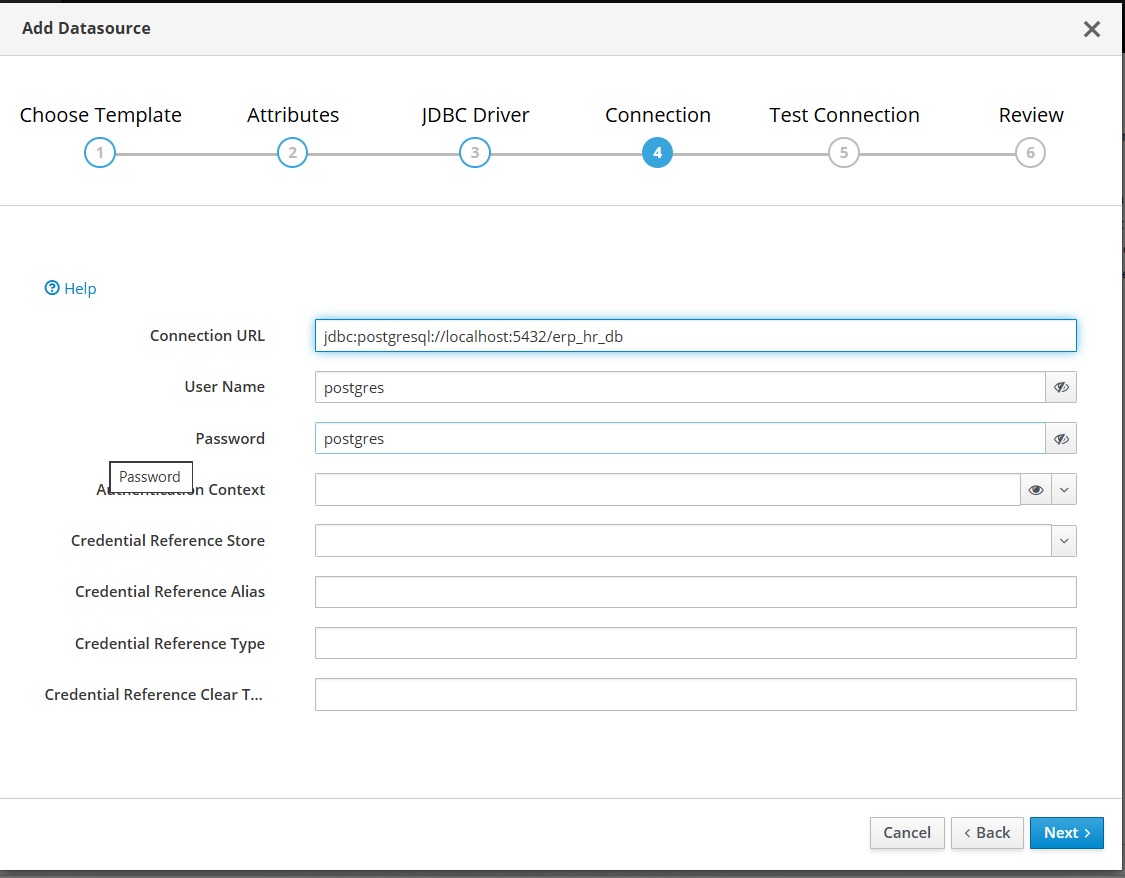
Se está accediendo a una interfaz web local en http://localhost:9990/console/index.html. Aparece un cuadro de diálogo para iniciar sesión con campos para Nombre de usuario (valor: juan26) y Contraseña (oculta). Hay dos botones: Acceder y Cancelar. Esto indica que se está intentando entrar al panel de administración o consola de una aplicación desplegada localmente.

Se está agregando una nueva fuente de datos (Datasource) en el servidor WildFly. El asistente está en el paso Attributes, donde se define:

Name: ErpHDS

JNDI Name: java:/jboss/datasources/ErpHDS

Este proceso forma parte de la configuración para conectar la aplicación a una base de datos mediante JDBC. Los siguientes pasos incluyen seleccionar el driver JDBC, establecer la conexión, probarla y revisar la configuración antes de finalizar.



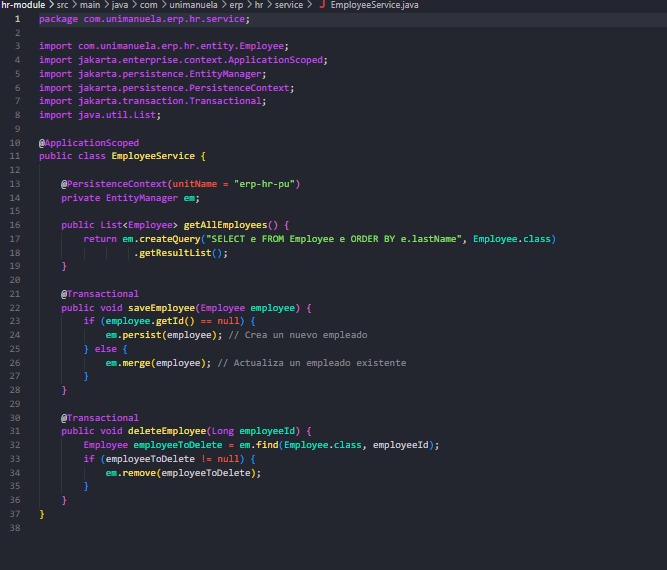
Se está definiendo la conexión a la base de datos PostgreSQL. Los campos completados son:

Connection URL: jdbc:postgresql://localhost:5432/erp\_hr\_db

User Name: postgres

Password: postgres

Esto indica que la aplicación se conectará a una base de datos llamada erp\_hr\_db en el puerto 5432 del servidor local, usando las credenciales del usuario postgres. El siguiente paso será probar la conexión (Test Connection) y luego revisar la configuración antes de finalizar.



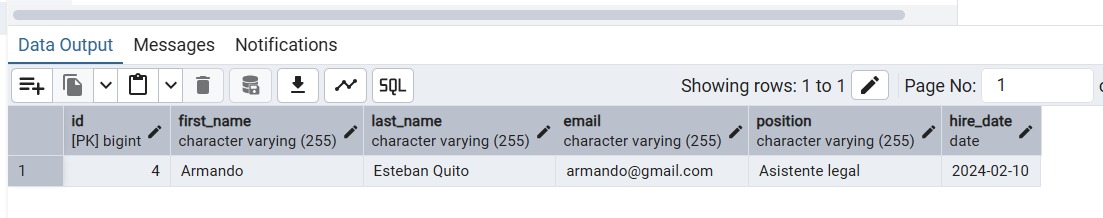
Se trata de una clase de servicio para manejar operaciones sobre la entidad Employee. Está anotada con @ApplicationScoped y utiliza un EntityManager inyectado mediante @PersistenceContext con la unidad de persistencia erp-hr-pu.

Incluye tres métodos principales:

getAllEmployees(): Ejecuta una consulta JPQL para obtener todos los empleados ordenados por apellido.

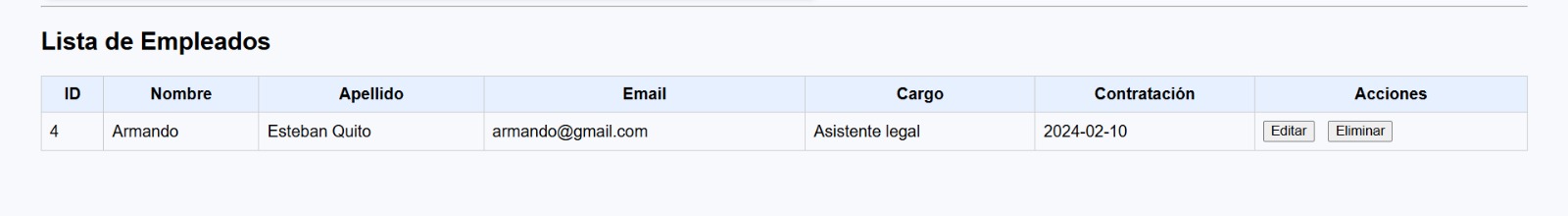
saveEmployee(Employee employee): Guarda un nuevo empleado si no tiene ID, o actualiza uno existente usando persist y merge.

deleteEmployee(Long employeeId): Busca un empleado por ID y lo elimina si existe.



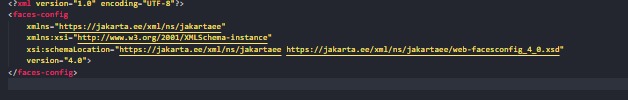
Se visualiza una tabla con los datos de la entidad Employee

Esto confirma que la aplicación logró persistir datos en la base de datos PostgreSQL, validando la configuración del datasource y la lógica implementada en el servicio.



Se visualiza un registro de empleado.

Además, hay botones para Editar y Eliminar, lo que indica que la aplicación permite operaciones CRUD desde la interfaz.



Este archivo define la configuración para JSF en la aplicación. Contiene:

Declaración XML con versión y codificación.

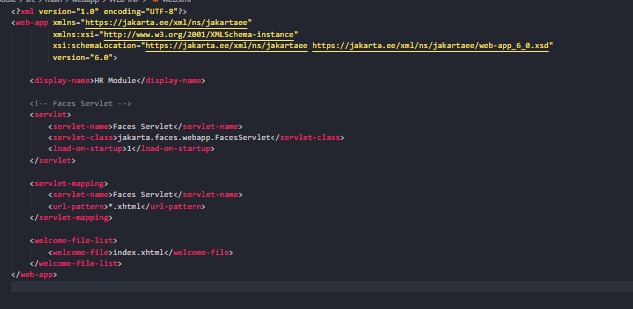
Etiqueta <faces-config> con atributos:

xmlns apuntando a https://jakarta.ee/xml/ns/jakartaee.

xsi:schemaLocation indicando la ubicación del esquema para validación.

version="4.0" que corresponde a Jakarta Faces 4.0.

Este archivo es esencial para habilitar y configurar el ciclo de vida de JSF en la aplicación web.



Este archivo contiene:

Declaración XML con versión y codificación.

Etiqueta <web-app> con atributos para el esquema de Jakarta EE versión 6.0.

display-name: HR Module.

Configuración del Faces Servlet:

<servlet-name>: Faces Servlet.

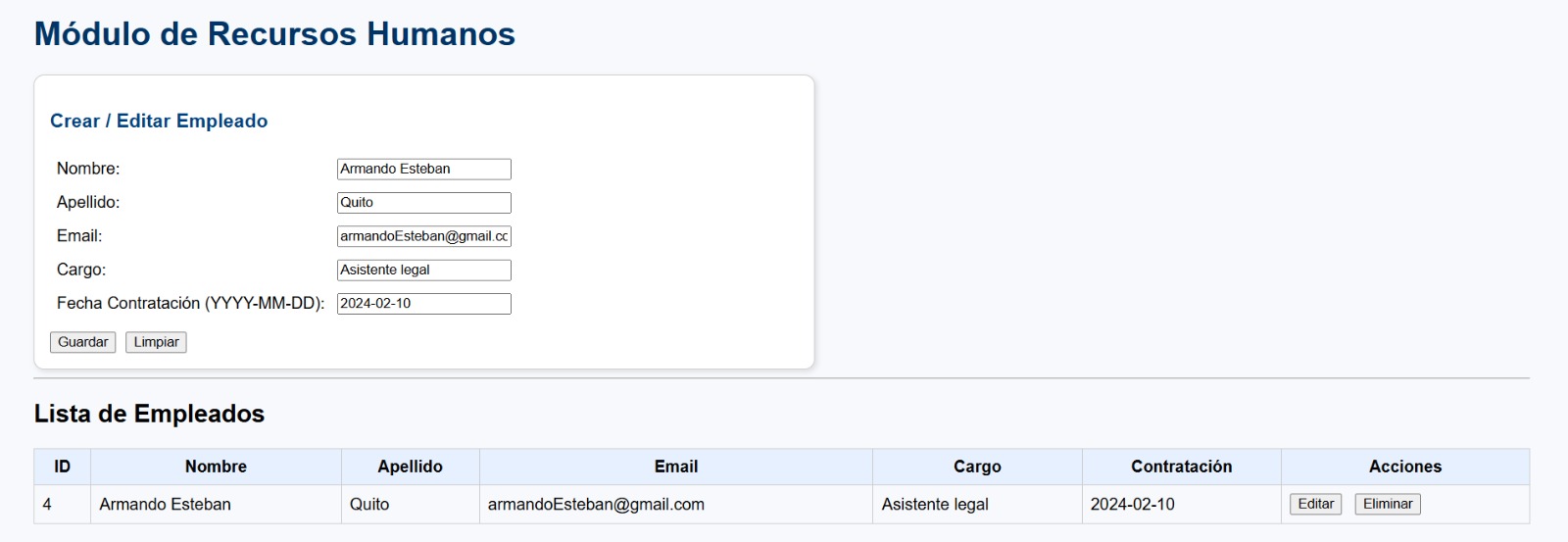
<servlet-class>: jakarta.faces.webapp.FacesServlet.

load-on-startup: 1 (inicia al arrancar la aplicación).

Servlet Mapping: Asocia el patrón \*.xhtml al Faces Servlet.

Welcome File: index.xhtml como página inicial.

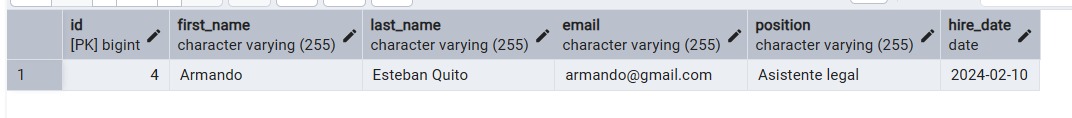
Esto indica que la aplicación usa Jakarta Faces (JSF) para la capa de presentación y está correctamente configurada para manejar vistas .xhtml.

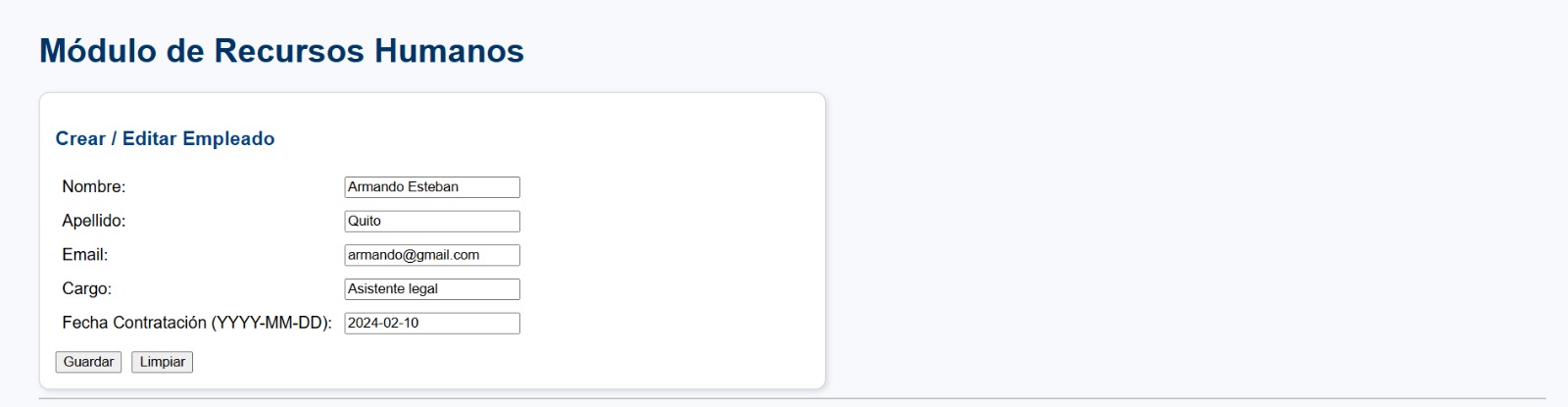


Parte superior: Formulario para crear/editar empleado.

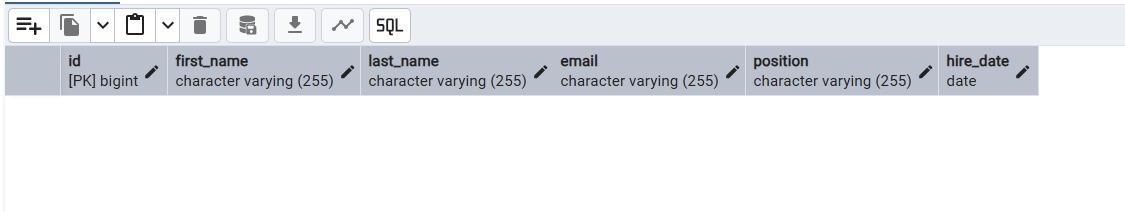
Parte inferior: Tabla Lista de Empleados con el mismo registro y botones Editar y Eliminar.

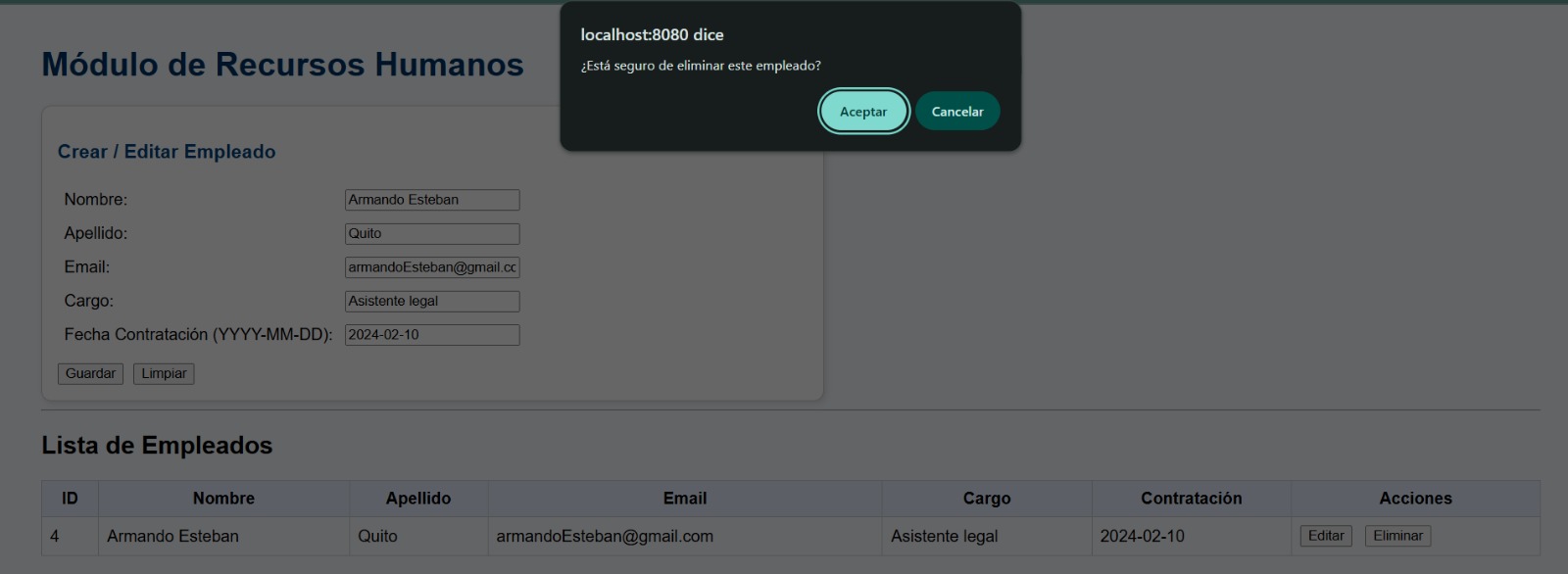
Esto confirma que la aplicación implementa CRUD completo desde la interfaz JSF, conectado a la base de datos PostgreSQL.





Este formulario confirma la funcionalidad CRUD en la interfaz JSF, conectada a la lógica JPA y la base de datos PostgreSQL.





Emergente: Mensaje de confirmación “¿Está seguro de eliminar este empleado?” con opciones Aceptar y Cancelar. Esto confirma la funcionalidad Eliminar dentro del CRUD implementado en la aplicación.