

TALLER Crear Módulo CRUD con Jakarta EE y JSF

Santiago Castro González

Universidad Manuela Beltrán

INGENIERÍA WEB II

Carlos Eduardo Mujica Reyes

13 de septiembre del 2025

Manual de Construcción y Despliegue de la Aplicación Web crud-empleados

1. Introducción  
    Este manual describe la construcción, configuración y despliegue de la aplicación web crud-empleados, un sistema CRUD para gestionar empleados, utilizando tecnologías modernas y estándares de la industria: Jakarta EE 10, JSF 4, JPA 3, CDI 4, WildFly y PostgreSQL en contenedor Docker.
2. Metodología y Tecnologías Seleccionadas

| **Tecnología** | **Uso** |
| --- | --- |
| Jakarta EE 10 | Plataforma base para desarrollo de aplicaciones empresariales Java |
| JSF 4 / Facelets | Framework web para interfaces de usuario |
| JPA 3 / Hibernate | Persistencia de datos y mapeo objeto-relacional |
| CDI 4 | Inyección de dependencias y gestión de ciclo de vida de componentes |
| WildFly 26+ | Servidor de aplicaciones Jakarta EE |
| PostgreSQL 15 (Docker) | Base de datos relacional |
| Apache Maven | Gestión de dependencias y compilación |
| NetBeans | IDE con soporte Maven y Jakarta EE |

1. Configuración del Entorno de Desarrollo

* **JDK**: Java 23. Verificación: java -version
* **NetBeans**: IDE para desarrollo y edición.
* **Maven**: Integrado en NetBeans. Verificación: mvn -v
* **WildFly**: Servidor de aplicaciones Jakarta EE.
* **Docker PostgreSQL**:

docker run --name postgres-container \

-e POSTGRES\_USER=miusuario \

-e POSTGRES\_PASSWORD=clave123 \

-e POSTGRES\_DB=erp\_hr\_db \

-p 5432:5432 -d postgres:15

1. Estructura del Proyecto crud-empleados

crud-empleados/

├─ Web Pages/

│ └─ WEB-INF/

│ └─ employees.xhtml

├─ Source Packages/

│ └─ com.unimanuela.erp.hr.app/

│ ├─ Employee.java

│ ├─ EmployeeBean.java

│ └─ EmployeeService.java

├─ src/main/resources/

│ └─ META-INF/persistence.xml

├─ Dependencies/

│ └─ Java Dependencies

├─ Project Files/

│ ├─ pom.xml

│ └─ nb-configuration.xml

1. Configuración de Persistencia (JPA)

* **Employee.java**

@Entity

@Table(name = "employees")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name="first\_name", nullable=false)

private String firstName;

@Column(name="last\_name", nullable=false)

private String lastName;

@Column(nullable=false, unique=true)

private String email;

private String position;

@Column(name="hire\_date", nullable=false)

private LocalDate hireDate;

// Getters y Setters

}

* **persistence.xml**

<persistence-unit name="erp-hr-pu" transaction-type="JTA">

<jta-data-source>java:jboss/datasources/ErpHrDS</jta-data-source>

<properties>

<property name="hibernate.hbm2ddl.auto" value="update"/>

<property name="hibernate.show\_sql" value="true"/>

</properties>

</persistence-unit>

1. Configuración de DataSource en WildFly

* Name: ErpHrDS
* JNDI Name: java:jboss/datasources/ErpHrDS
* Connection URL: jdbc:postgresql://localhost:5432/erp\_hr\_db
* Username: miusuario
* Password: clave123

1. Lógica de Negocio (CDI + JPA)

* **EmployeeService.java**

@ApplicationScoped

public class EmployeeService {

@PersistenceContext(unitName = "erp-hr-pu")

private EntityManager em;

public List<Employee> getAllEmployees() {

return em.createQuery("SELECT e FROM Employee e ORDER BY e.lastName", Employee.class).getResultList();

}

@Transactional

public void saveEmployee(Employee employee) {

if (employee.getId() == null) em.persist(employee);

else em.merge(employee);

}

@Transactional

public void deleteEmployee(Long employeeId) {

Employee emp = em.find(Employee.class, employeeId);

if (emp != null) em.remove(emp);

}

}

1. Interfaz de Usuario y Backing Bean

* **EmployeeBean.java**

@Named("employeeBean")

@ViewScoped

public class EmployeeBean implements Serializable {

@Inject

private EmployeeService service;

private Employee selectedEmployee;

private List<Employee> employeeList;

@PostConstruct

public void init() {

selectedEmployee = new Employee();

employeeList = service.getAllEmployees();

}

public void save() { ... }

public void delete(Long id) { ... }

public void prepareNew() { selectedEmployee = new Employee(); }

}

* **employees.xhtml**

<h:form>

<h:panelGrid columns="2">

<h:outputLabel for="firstName" value="Nombre:"/>

<h:inputText id="firstName" value="#{employeeBean.selectedEmployee.firstName}" required="true"/>

...

</h:panelGrid>

<h:commandButton value="Guardar" action="#{employeeBean.save}"/>

</h:form>

<h:dataTable value="#{employeeBean.employeeList}" var="emp">

<h:column>#{emp.id}</h:column>

...

</h:dataTable>

1. Compilación y Despliegue

* Compilar y empaquetar WAR: mvn clean package
* Desplegar en WildFly: copiar crud-empleados.war a standalone/deployments/
* Probar: http://localhost:8080/crud-empleados/employees.xhtml

1. Conclusión  
    La aplicación integra un stack moderno Jakarta EE con persistencia en PostgreSQL mediante contenedor Docker. Cumple con las mejores prácticas y estándares de la industria.