

# INTRODUCCIÓN A JSF

# Desarrollo de una

# App Empresarial con

# JEE



INTEGRANTES:

CARLOS SANCHEZ ALZATE

YESENIA BABILONIA MORALES

ARQUITECTURA DE SOFTWARE

UNIVERSIDAD DE ANTIOQUIA

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE SISTEMAS

2019-02

---

# ÍNDICE

<b>INTRODUCCIÓN</b>	<b>3</b>
<b>OBJETIVOS</b>	<b>4</b>
OBJETIVO GENERAL	4
OBJETIVOS ESPECÍFICOS	4
<b>HERRAMIENTAS IMPLEMENTADAS</b>	<b>4</b>
<b>CONCEPTOS</b>	<b>5</b>
<b>INSTRUCCIONES</b>	<b>5</b>
<b>DESARROLLO</b>	<b>6</b>
CREACIÓN DE LA BASE DE DATOS	6
CONFIGURACIÓN DE CONEXIÓN A LA DB	12
CREANDO POOL DE CONEXIONES A LA DB	15
CREACIÓN DE APLICACIÓN WEB	18
CREAR ENTITY CLASS DESDE UNA BASE DE DATOS EXISTENTE	21
Añadir algunos métodos de negocio en el Facade	29
<b>CREAR PARTE WEB</b>	<b>33</b>
Se generaron los getter y Setter de CustomerMBean.	41
Creación de la Unidad de Persistencia	42
Despliegue	44
<b>CONCLUSIONES.</b>	<b>47</b>
<b>ANEXOS.</b>	<b>47</b>

---

## **INTRODUCCIÓN**

El presente informe, pretende ilustrar los pasos ejecutados para el desarrollo de CRUD utilizando JSF, PrimeFaces y Backing Beans por medio de la plataforma JEE.

- Create o Insert: Inserta datos .
- List: Lista los datos de la información generada.
- Update: Actualizar información
- Remove o Delete: Elimina datos

---

## **OBJETIVOS**

### **OBJETIVO GENERAL**

Comprender el uso de PrimeFaces en el desarrollo de aplicaciones Java Web.

### **OBJETIVOS ESPECÍFICOS**

- Comprender el uso de Backing Beans para apoyar las transacciones complejas que se solicitan desde la vista.
- Conocer y utilizar las definiciones de PrimeFaces que nos permiten representar componentes gráficos de una manera más intuitiva y simple.
- Conocer el funcionamiento y diferencias de las anotaciones de tipo Scoped que provee PrimeFaces para representar sus Managed Beans.
- Aplicar y asimilar las funciones que ofrece el IDE NetBeans para la generación de componentes gráficos a partir de Entity Beans.

## **HERRAMIENTAS IMPLEMENTADAS**

- NetBeans 11.2.
- Web Browser (Firefox, Chrome).
- Java EE 8 Web, plataforma de desarrollo.
- JavaServer Faces.(JSF)
- PrimeFaces.
- Enterprise Javabeans.
- GlassFish Server 5 (servidor de aplicaciones).
- Mysql 8
- JPA

---

## CONCEPTOS

**CRUD:** En computación CRUD es el acrónimo de Crear, Leer, Actualizar y Eliminar (del original en inglés: Create, Read, Update and Delete). Se usa para referirse a las funciones básicas en bases de datos o la capa de persistencia en un software.

**JSF (Java Server Faces):** Es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE. JSF usa JavaServer Pages (JSP) como la tecnología que permite hacer el despliegue de las páginas, pero también se puede acomodar a otras tecnologías como XUL (acrónimo de XML-based User-interface Language, lenguaje basado en XML para la interfaz de usuario)

**JPA (Java Persistence API):** Es un framework del lenguaje de programación Java que maneja datos relacionales en aplicaciones usando la Plataforma Java en sus ediciones Standard (Java SE) y Enterprise (Java EE). El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos (siguiendo el patrón de mapeo objeto-relacional), como sí pasaba con EJB2, y permitir usar objetos regulares (conocidos como POJOs).

**MySQL:** Es un sistema de administración de bases de datos (Database Management System, DBMS) para bases de datos relacionales. Así, MySQL no es más que una aplicación que permite gestionar archivos llamados de bases de datos

**JSF Manager Beans:** Managed Bean es una clase Java Bean regularmente registrada con JSF. En otras palabras, Managed Beans es un Java Bean gestionado por el marco JSF.

## INSTRUCCIONES

# DESARROLLO

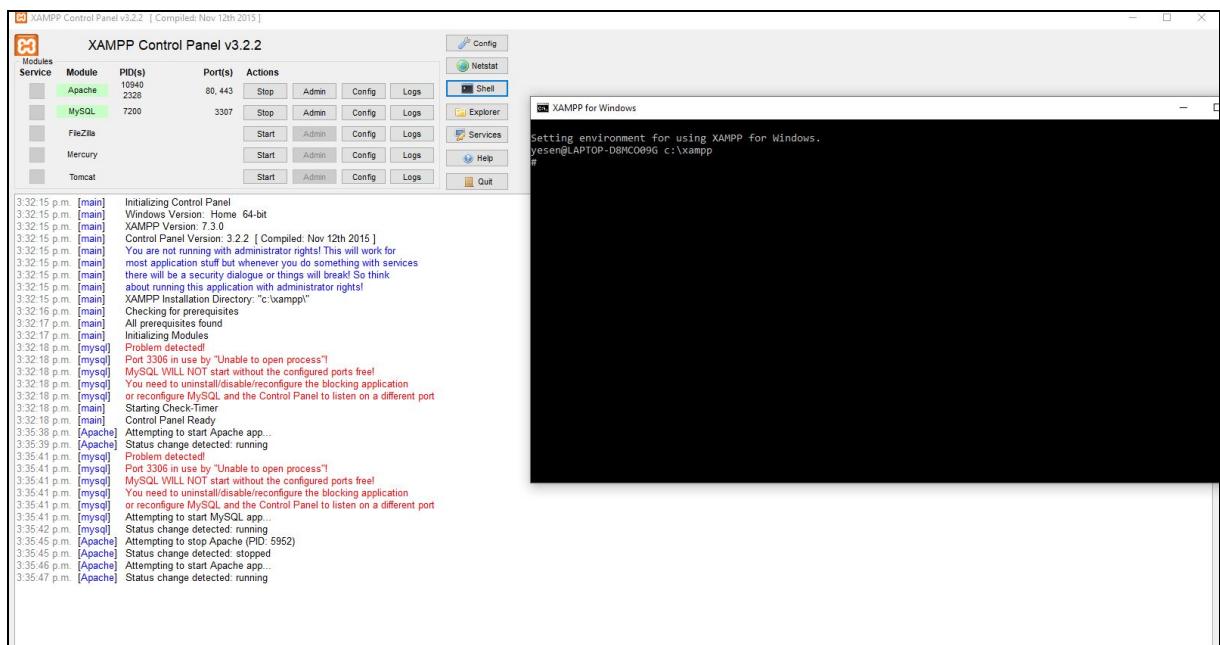
## I. CREACIÓN DE LA BASE DE DATOS

Ante todo se descarga de la página oficial de Mysql ([MySQL :: Sakila Sample Database :: 4 Installation](#)), la DB Sakila, así como se detalla a continuación, luego se descomprime

Title	Download DB	HTML Setup Guide	PDF Setup Guide
employee data (large dataset, includes data and test/verification suite)	<a href="#">GitHub</a>	<a href="#">View</a>	US Ltr   A4
world database	<a href="#">Gzip   Zip</a>	<a href="#">View</a>	US Ltr   A4
world_x database	<a href="#">TGZ   Zip</a>	<a href="#">View</a>	US Ltr   A4
•sakila database	<a href="#">TGZ   Zip</a>	<a href="#">View</a>	US Ltr   A4
menagerie database	<a href="#">TGZ   Zip</a>		

Figura.1.

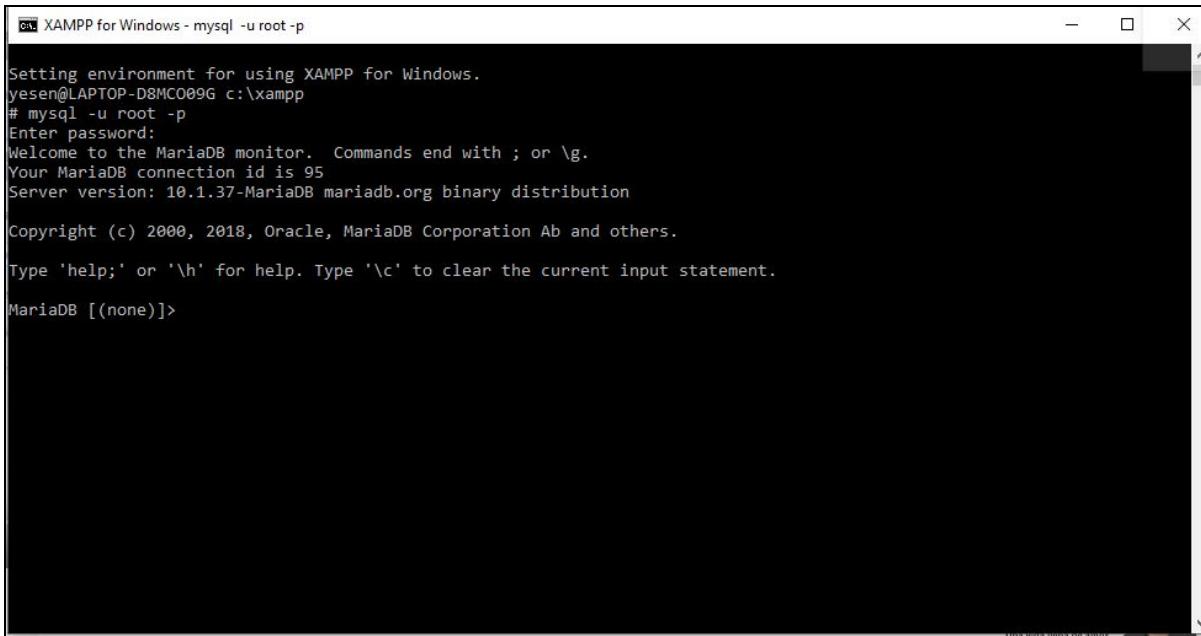
inicia en este caso XAMPP, luego activamos MySQL y posteriormente nos dirigimos al shell, como se ilustra en la figura 2



```
3:32:15 p.m. [main] Initializing Control Panel
3:32:15 p.m. [main] Windows Version: Home 64-bit
3:32:15 p.m. [main] XAMPP Version: 7.3.0
3:32:15 p.m. [main] Control Panel Version: 3.2.2 [ Compiled: Nov 12th 2015 ]
3:32:15 p.m. [main] You are not running with administrator rights! This will work for
3:32:15 p.m. [main] most application stuff but whenever you do something with services
3:32:15 p.m. [main] there will be a security dialogue or things will break! So think
3:32:15 p.m. [main] about running this application with administrator rights!
3:32:15 p.m. [main] XAMPP Install Dir: "c:\xampp"
3:32:15 p.m. [main] Checking for prerequisites
3:32:17 p.m. [main] All prerequisites found
3:32:17 p.m. [main] Initializing Modules
3:32:18 p.m. [mysql] Problem detected!
3:32:18 p.m. [mysql] Port 3306 in use by "Unable to open process"
3:32:18 p.m. [mysql] MySQL WILL NOT start without the configured ports free!
3:32:18 p.m. [mysql] You need to uninstall/disable/reconfigure the blocking application
3:32:18 p.m. [mysql] or reconfigure MySQL and the Control Panel to listen on a different port
3:32:18 p.m. [main] Starting Check-Timer
3:32:18 p.m. [main] Control Panel Ready
3:35:38 p.m. [Apache] Attempting to start Apache app...
3:35:39 p.m. [Apache] Status change detected: running
3:35:41 p.m. [mysql] Problem detected!
3:35:41 p.m. [mysql] Port 80 in use by "Unable to open process"
3:35:41 p.m. [mysql] MySQL WILL NOT start without the configured ports free!
3:35:41 p.m. [mysql] You need to uninstall/disable/reconfigure the blocking application
3:35:41 p.m. [mysql] or reconfigure MySQL and the Control Panel to listen on a different port
3:35:41 p.m. [mysql] Attempting to start MySQL app...
3:35:42 p.m. [mysql] Status change detected: running
3:35:45 p.m. [Apache] Attempting to stop Apache (PID: 5952)
3:35:45 p.m. [Apache] Status change detected: stopped
3:35:46 p.m. [Apache] Attempting to start Apache app...
3:35:47 p.m. [Apache] Status change detected: running
```

Figura.2.

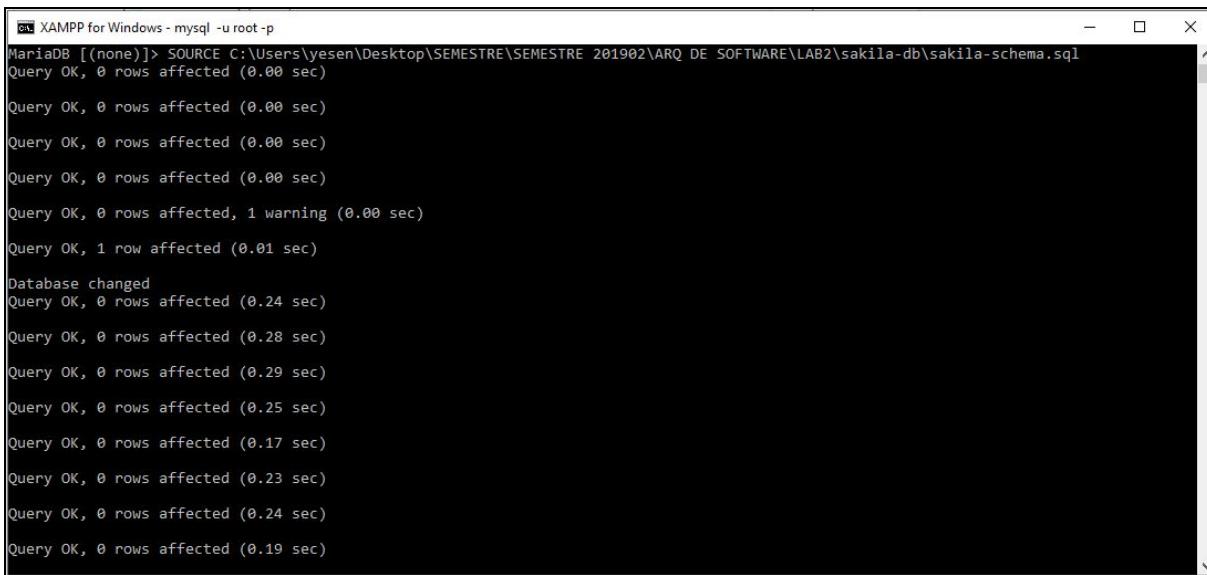
Luego nos logueamos como administrador(*mysql -u root -p*) e insertamos la contraseña, que en este caso no hay, es un campo vacío



```
Setting environment for using XAMPP for Windows.  
yesen@LAPTOP-D8MC009G c:\xampp  
# mysql -u root -p  
Enter password:  
Welcome to the MariaDB monitor. Commands end with ; or \g.  
Your MariaDB connection id is 95  
Server version: 10.1.37-MariaDB mariadb.org binary distribution  
  
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
MariaDB [(none)]>
```

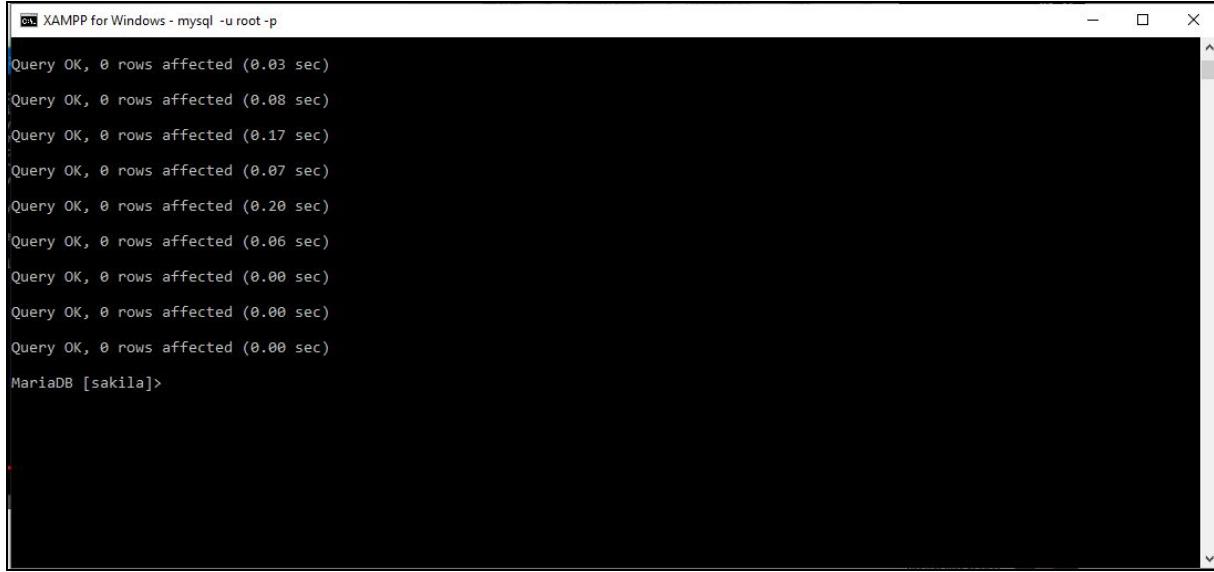
Figura.3.

Ahora decimos que en la siguiente dirección está el esquema de la DB : **SOURCE C:\Users\yesen\Desktop\SEMESTRE\SEMESTRE 201902\ARQ DE SOFTWARE\LAB2\sakila-db\sakila-schema.sql**



```
MariaDB [(none)]> SOURCE C:\Users\yesen\Desktop\SEMESTRE\SEMESTRE 201902\ARQ DE SOFTWARE\LAB2\sakila-db\sakila-schema.sql  
Query OK, 0 rows affected (0.00 sec)  
Query OK, 0 rows affected, 1 warning (0.00 sec)  
Query OK, 1 row affected (0.01 sec)  
  
Database changed  
Query OK, 0 rows affected (0.24 sec)  
Query OK, 0 rows affected (0.28 sec)  
Query OK, 0 rows affected (0.29 sec)  
Query OK, 0 rows affected (0.25 sec)  
Query OK, 0 rows affected (0.17 sec)  
Query OK, 0 rows affected (0.23 sec)  
Query OK, 0 rows affected (0.24 sec)  
Query OK, 0 rows affected (0.19 sec)
```

Figura.4



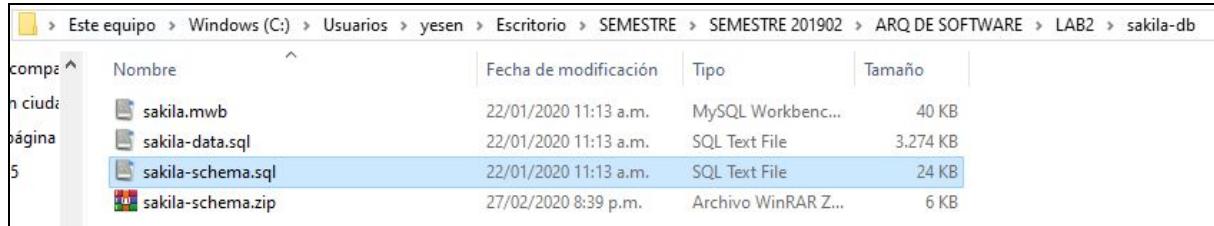
```
XAMPP for Windows - mysql -u root -p

Query OK, 0 rows affected (0.03 sec)
Query OK, 0 rows affected (0.08 sec)
Query OK, 0 rows affected (0.17 sec)
Query OK, 0 rows affected (0.07 sec)
Query OK, 0 rows affected (0.20 sec)
Query OK, 0 rows affected (0.06 sec)
Query OK, 0 rows affected (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
Query OK, 0 rows affected (0.00 sec)

MariaDB [sakila]>
```

Figura.5

NOTA: Recuerde la carpeta donde se descargó Sakila desde la página oficial de Mysql, observe la figura 6



Este equipo > Windows (C:) > Usuarios > yesen > Escritorio > SEMESTRE > SEMESTRE 201902 > ARQ DE SOFTWARE > LAB2 > sakila-db				
Nombre	Fecha de modificación	Tipo	Tamaño	
sakila.mwb	22/01/2020 11:13 a.m.	MySQL Workbench Data	40 KB	
sakila-data.sql	22/01/2020 11:13 a.m.	SQL Text File	3.274 KB	
sakila-schema.sql	22/01/2020 11:13 a.m.	SQL Text File	24 KB	
sakila-schema.zip	27/02/2020 8:39 p.m.	Archivo WinRAR Zip	6 KB	

Figura.6

Verificamos en el administrador de nuestra DB (PhpMyAdmin) y efectivamente observamos (Figura.7) la DB Creada con sus respectivas entidades

Figura.7

Seguidamente se insertan los datos almacenados en sakila, corresponde al archivo “**sakila-data.sql**”(figura 8), con el comando: **SOURCE**  
**C:\Users\yesen\Desktop\SEMESTRE\SEMESTRE 201902\ARQ DE SOFTWARE\LAB2\sakila-db\sakila-schema.sql**, así como se evidencia en la figura 9

Nombre	Fecha de modificación	Tipo	Tamaño
sakila.mwb	22/01/2020 11:13 a.m.	MySQL Workbenc...	40 KB
<b>sakila-data.sql</b>	22/01/2020 11:13 a.m.	SQL Text File	3.274 KB
sakila-schema.sql	22/01/2020 11:13 a.m.	SQL Text File	24 KB
sakila-schema.zip	27/02/2020 8:39 p.m.	Archivo WinRAR Z...	6 KB

Figura.8

```
mysql [sakila]> SOURCE C:\Users\yesen\Desktop\SEMESTRE\SEMESTRE 201902\ARQ DE SOFTWARE\LAB2\sakila-db\sakila-data.sql
Query OK, 0 rows affected (0.00 sec)

Database changed
Query OK, 0 rows affected (0.00 sec)

Query OK, 200 rows affected (0.03 sec)
Records: 200  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.05 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 603 rows affected (0.14 sec)
Records: 603  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.07 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 16 rows affected (0.00 sec)
Records: 16  Duplicates: 0  Warnings: 0
```

Figura.9

Observemos en la *figura 10*, indica que ya han sido insertado los datos, para ello, procedemos a visualizar una vez más a phpMyAdmin (*figura 11*)

```
mysql [sakila]> SOURCE C:\Users\yesen\Desktop\SEMESTRE\SEMESTRE 201902\ARQ DE SOFTWARE\LAB2\sakila-db\sakila-data.sql
Query OK, 0 rows affected (0.59 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 2 rows affected (0.26 sec)
Records: 2  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.18 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 2 rows affected (0.55 sec)
Records: 2  Duplicates: 0  Warnings: 0

Query OK, 0 rows affected (0.04 sec)

Query OK, 0 rows affected (0.00 sec)

MariaDB [sakila]>
```

Figura.10

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1 > Base de datos: sakila > Tabla: actor
- SQL Query:** SELECT \* FROM `actor`
- Table Data:**

	actor_id	first_name	last_name	last_update
1	1	PENELOPE	GUINNESS	2006-02-15 04:34:33
2	2	NICK	WAHLBERG	2006-02-15 04:34:33
3	3	ED	CHASE	2006-02-15 04:34:33
4	4	JENNIFER	DAVIS	2006-02-15 04:34:33
5	5	JOHNNY	LOLLOBRIGIDA	2006-02-15 04:34:33
6	6	BETTE	NICHOLSON	2006-02-15 04:34:33
7	7	GRACE	MOSTEL	2006-02-15 04:34:33
8	8	MATTHEW	JOHANSSON	2006-02-15 04:34:33
9	9	JOE	SWANK	2006-02-15 04:34:33
10	10	CHRISTIAN	GABLE	2006-02-15 04:34:33
11	11	ZERO	CAGE	2006-02-15 04:34:33
12	12	KARL	BERRY	2006-02-15 04:34:33
13	13	UMA	WOOD	2006-02-15 04:34:33
14	14	VIVIEN	BERGEN	2006-02-15 04:34:33
15	15	CUBA	OLIVIER	2006-02-15 04:34:33
16	16	FRED	COSTNER	2006-02-15 04:34:33
17	17	HELEN	VOIGHT	2006-02-15 04:34:33
18	18	DAN	TORN	2006-02-15 04:34:33
19	19	BOB	FAWCETT	2006-02-15 04:34:33

Figura.11

Otra forma de evidenciar que efectivamente nuestras tablas fueron creadas es con los comandos:

**use sakila** : ingresamos a la DB

```
MariaDB [sakila]> use sakila
Database changed
```

Figura.12

Una vez dentro, pedimos que nos proyecte las tablas: **show tables;**

```
ca XAMPP for Windows - mysql -u root -p
MariaDB [sakila]> show tables;
+-----+
| Tables_in_sakila |
+-----+
| actor
| actor_info
| address
| category
| city
| country
| customer
| customer_list
| film
| film_actor
| film_category
| film_list
| film_text
| inventory
| language
| nicer_but_slower_film_list
| payment
| rental
| sales_by_film_category
| sales_by_store
| staff
| staff_list
| store
+-----+
23 rows in set (0.00 sec)
```

Figura.13

## II. CONFIGURACIÓN DE CONEXIÓN A LA DB

Ahora se procede a establecer conexión

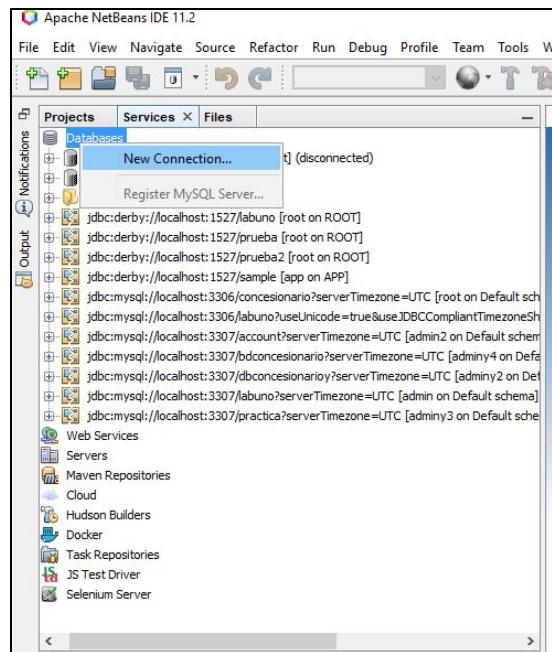


Figura.14

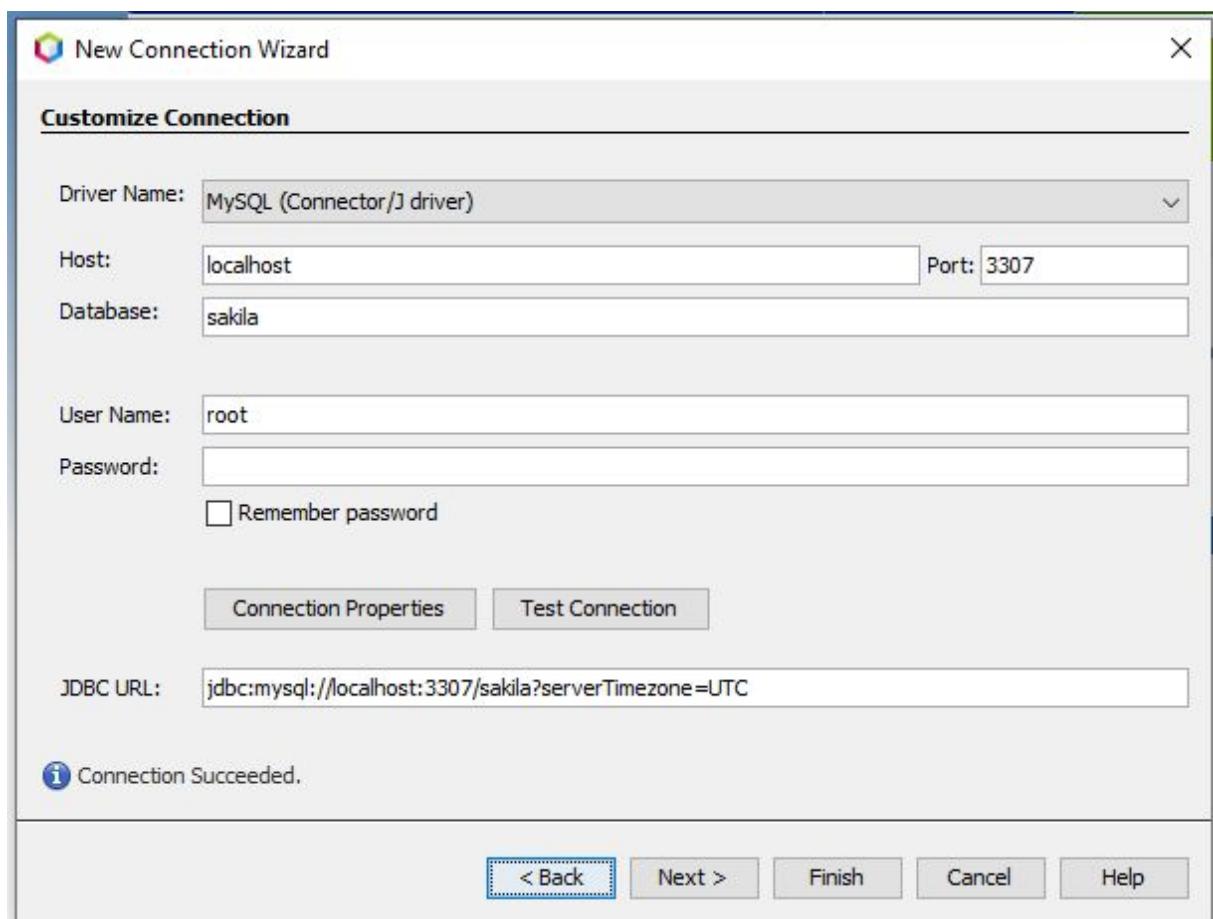


Figura.15

Exitosamente se logra la conexión

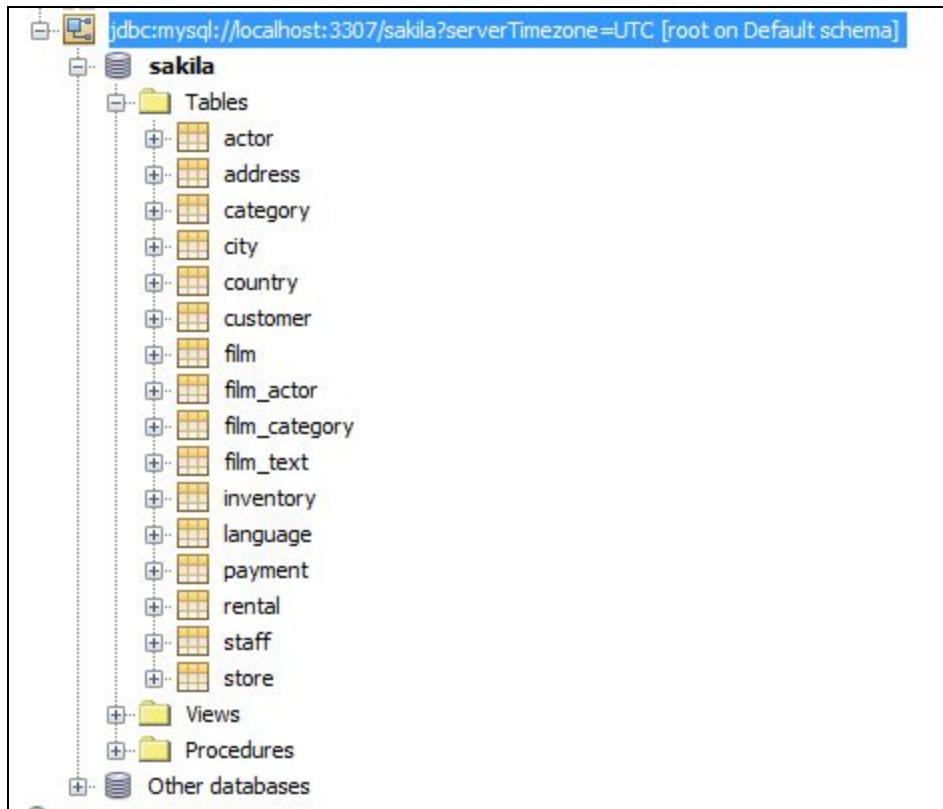


Figura.16

## CREANDO POOL DE CONEXIONES A LA DB

Se necesitaba crear un Pool de Conexiones para crear un recurso JDBC desde el servidor de aplicaciones a MySQL. Para esto se accedió al separador **Servers** del proyecto. Luego se hizo click en el nodo **Glassfish Server** y luego se hizo click derecho y se seleccionó **Start** (Figura 17). Una vez iniciado el servidor se seleccionó la opción **View Domain admin Console**. (Figura 18)

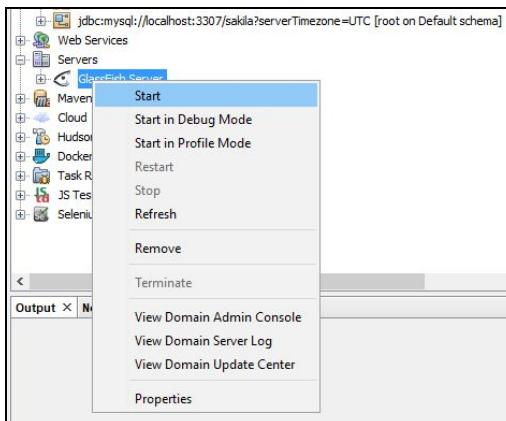


Figura.17

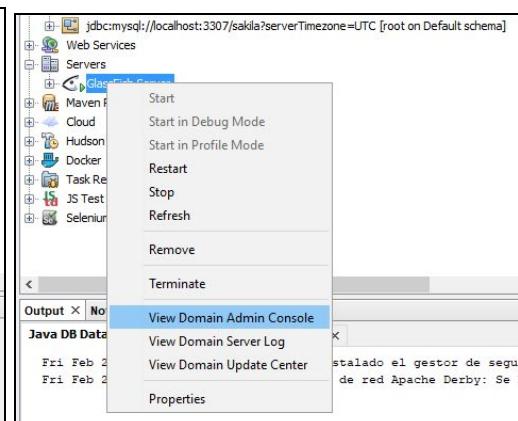
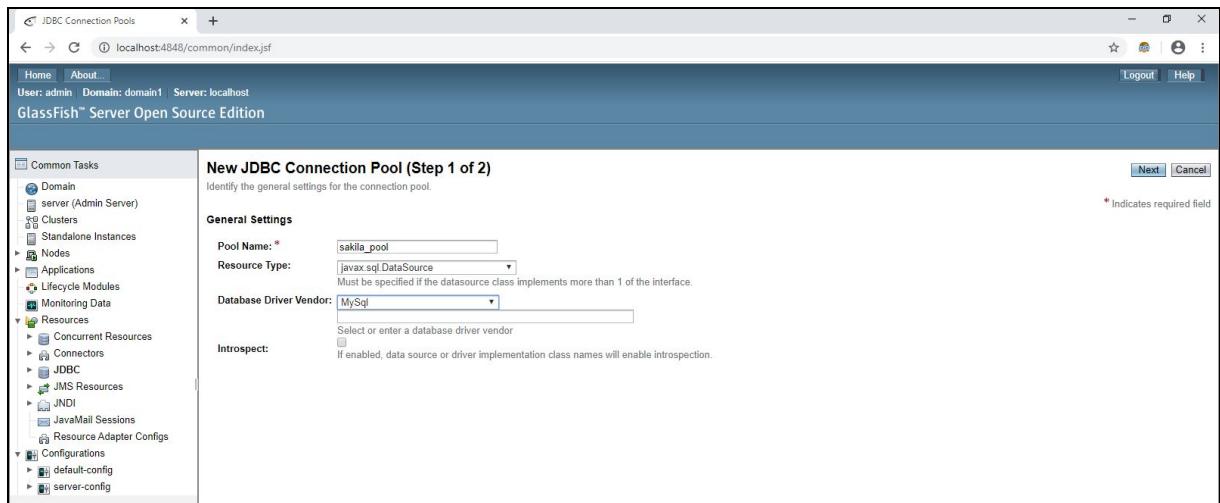


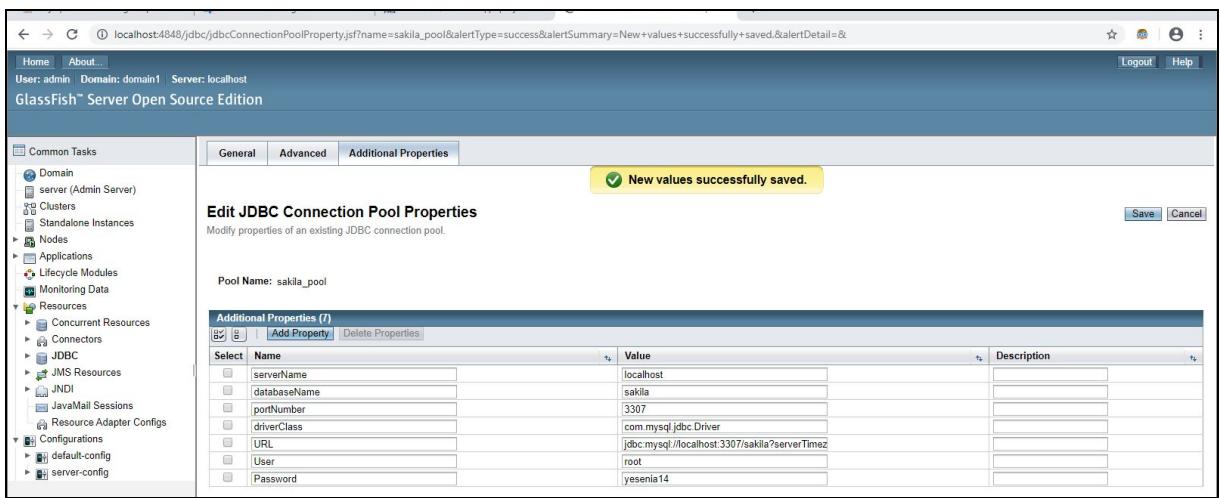
Figura.18

Se iniciará la consola de administración principal de Glassfish en el navegador que tenga por defecto. Dentro de todas las opciones que aparecen nos interesan los nodos JDBC Resources y JDBC Connection Pools. Por medio de la interfaz de glassfish existe la alternativa fácil de crear el recurso, a continuación se observa. Procedemos a añadir la información respectiva, así como aparece en la *figura 19*



*Figura.19*

Ya creado el pool de conexiones , se procede a añadir las propiedades ( *Figura.20*)



*Figura.20*

Seguidamente se crea en Recurso, así como se indica en la figura 21

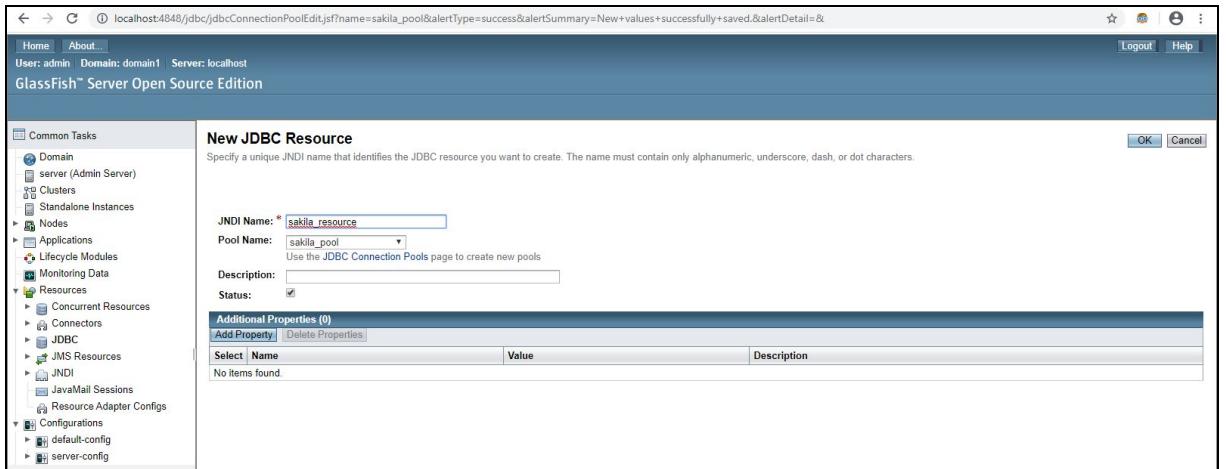


Figura.21

## Finalmente comprobamos la conexión

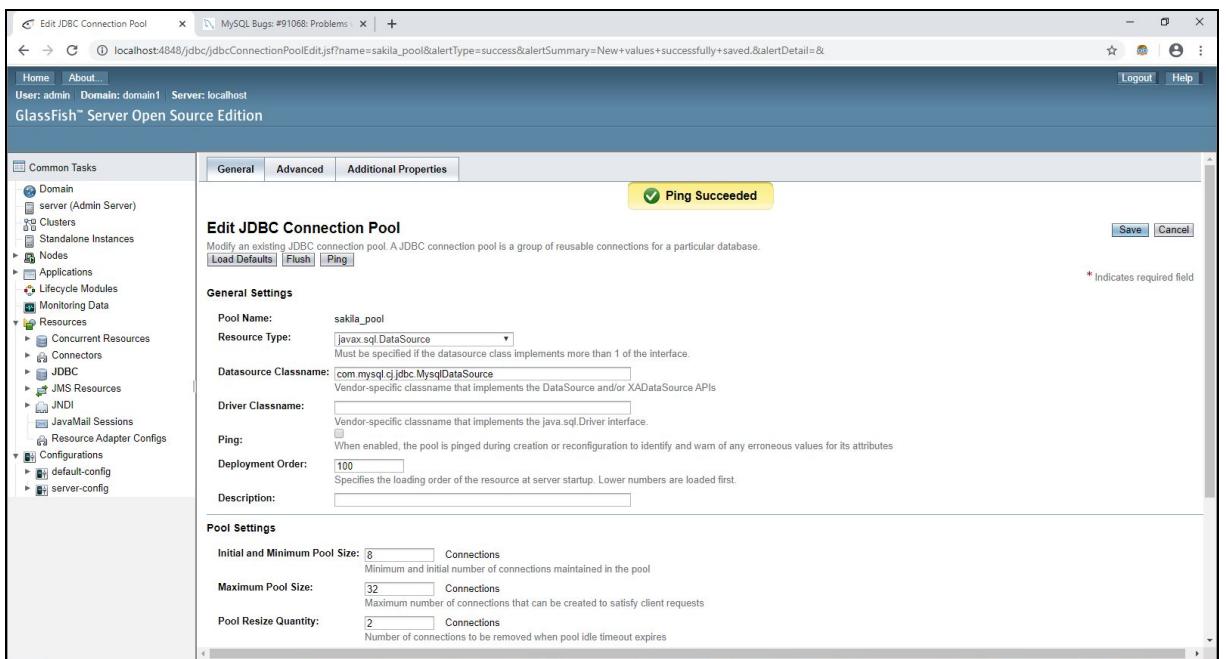


Figura.22

## CREACIÓN DE APLICACIÓN WEB

Es una clase especial de proyecto hecho en módulos diferentes: módulos EJB, módulos web, etc. Todos estos subproyectos son agrupados en un solo proyecto tipo empresarial, que se comporta como un solo contenedor. El Java EE es un archivo .ear (ear meas Enterprise ARchive), y hacer el despliegue más fácil

Usar Netbeans **menu File/New Project→ "Java EE" → "Enterprise Application"**.

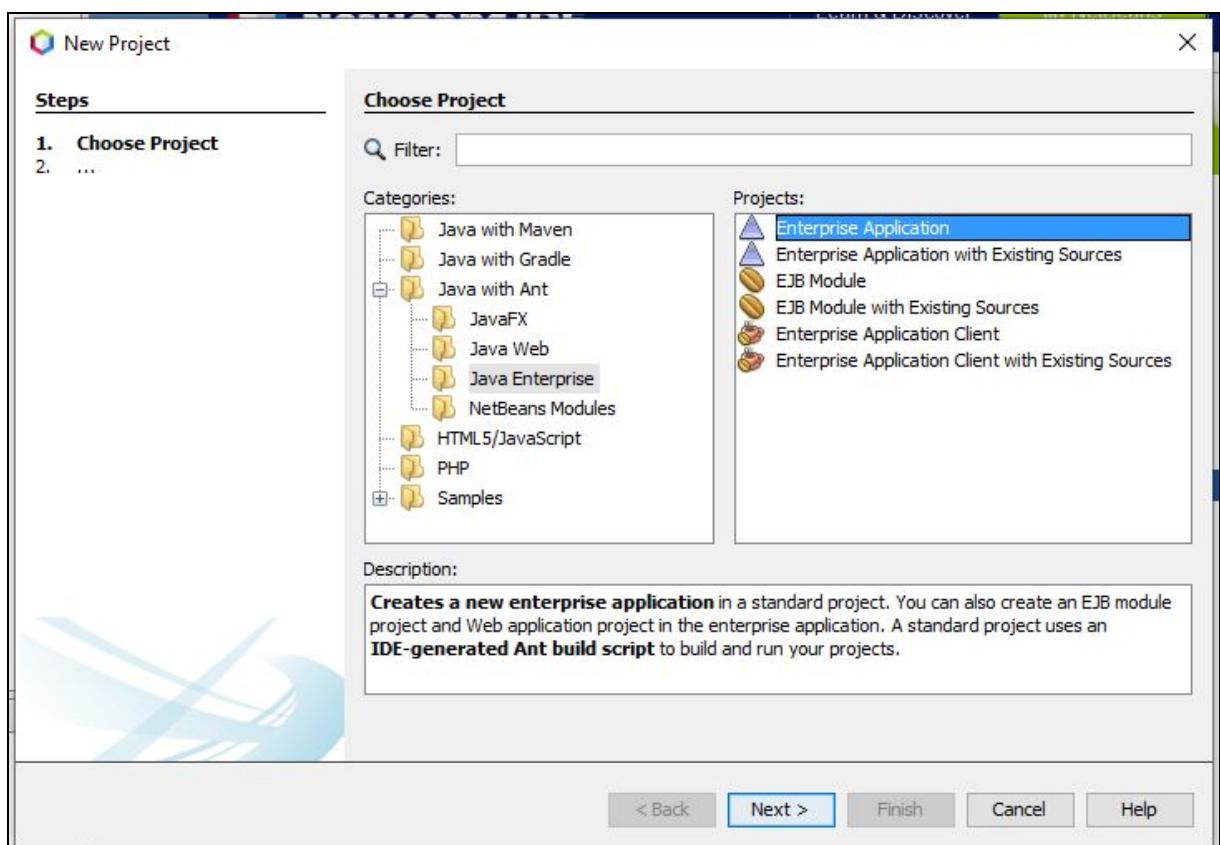


Figura.23

Luego continuamos presionando en "Next", se debe dar un nombre al proyecto, evitar caracteres extraños. Para este proyecto se utilizó el nombre Laboratorio2, como se muestra en la imagen.

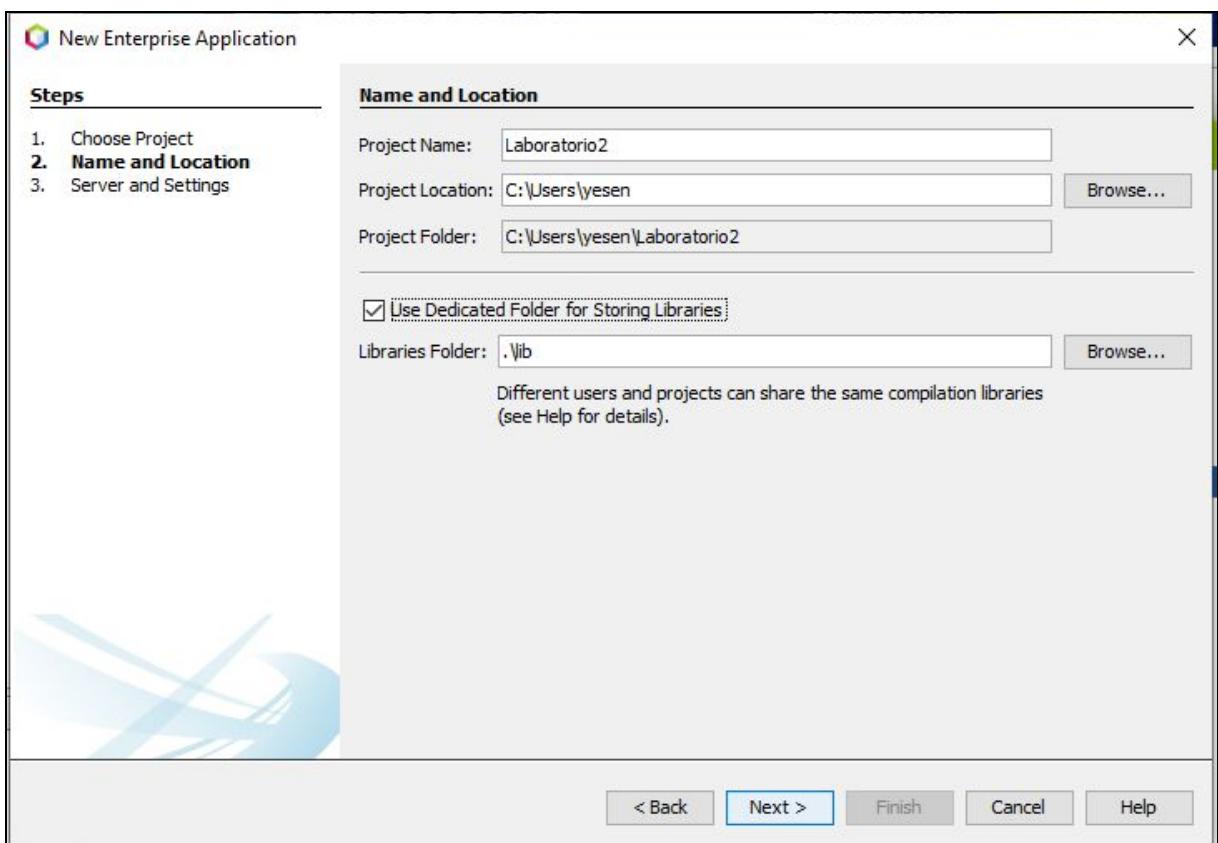


Figura.24

En la siguiente ventana, se debe elegir el servidor de aplicaciones, la versión del Java EE, en este caso corresponde a la 8; tomar todos los valores por defecto.o olvidar dar click en la casilla "**Enable Context and Dependency Injection**", de lo contrario JSF no va a funcionar.

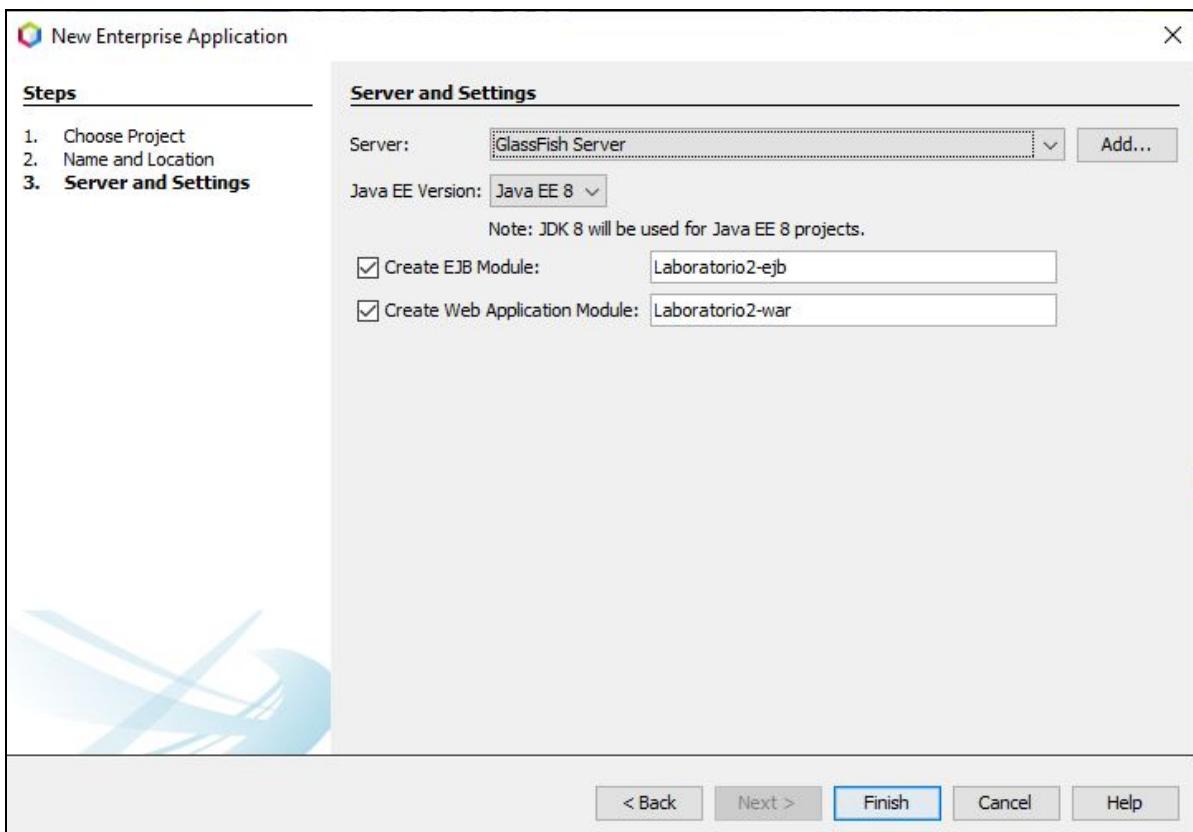


Figura.25

Clic en finalizar. Normalmente se verán 3 proyectos agregados a Netbeans:

- Un proyecto con un triángulo azul ( Laboratorio2 ), que sirve como contenedor para los dos proyectos siguientes, este es el proyecto .ear.
- Un proyecto con un ícono en forma de java bean o “grano de café” ( Laboratorio2-ejb ), es donde se llevará a cabo la parte de la lógica de negocio de la aplicación, en este caso EJBs y Entity class. Todos son "Javabeans": clases con un constructor por defecto, propiedades, etc.
- Un proyecto con un ícono en forma de mundo, este es el proyecto “Web ( Laboratorio2-war )” que llevará a cabo la parte HTTP de la aplicación, en este caso el JSF beans, la página JSF, el archivo CSS

Nuestro proyecto debe quedar luciendo como muestra la figura, se puede apreciar los tres nodos del proyecto, el contenedor principal (ícono del triángulo) con los módulos back-end (ícono grano de café) y front-end (ícono mundo) que acabamos de activar.

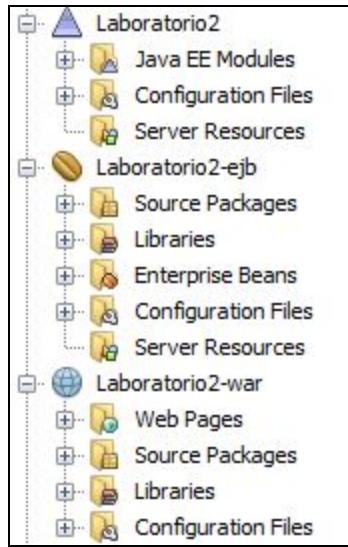


Figura.26

## CREAR ENTITY CLASS DESDE UNA BASE DE DATOS EXISTENTE

Java EE contiene un framework llamado JPA (Java Persistence API), en este framework las clases especiales llamadas "Entity class" corresponde a tablas en una base de datos. El mapeo entre estas clases y tablas se realiza usando herramientas ORM externas como Hibernate o EclipseLink, estas herramientas proporcionan algunos medios para generar automáticamente las Entity class desde la tabla modelo, esto es lo que se va a utilizar.

En el proyecto EJB -el que tiene un ícono de java bean- hacer click derecho en el nombre del proyecto y click en *New/Entity class from database*.

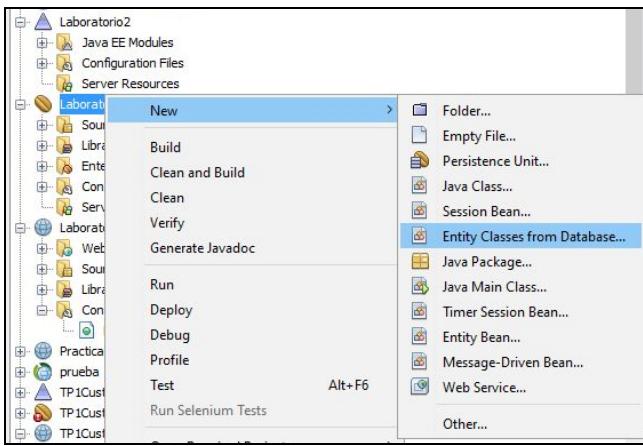


Figura.27

Se pedirá el nombre JNDI (Java Naming and Directory Interface) de la base de datos. Elegir "sakila\_resource", que es una base de datos que viene en la base de datos JavaDB incluida en glassfish 3. Por lo tanto no hay necesidad de ejecutar un servidor de Base de Datos, ni crear a mano alguna base de datos, tablas, etc.

Normalmente después de que se elige el nombre de la Base de Datos, la lista de la izquierda aparecerá mostrando las tablas disponibles:

Elegir "**customer**" y dar click en el botón "**Add**", se agregarán dos tablas: **customer**, **address**, **city**, **country**, **staff** y también **store**, ya que hay una relación entre las dos tablas.

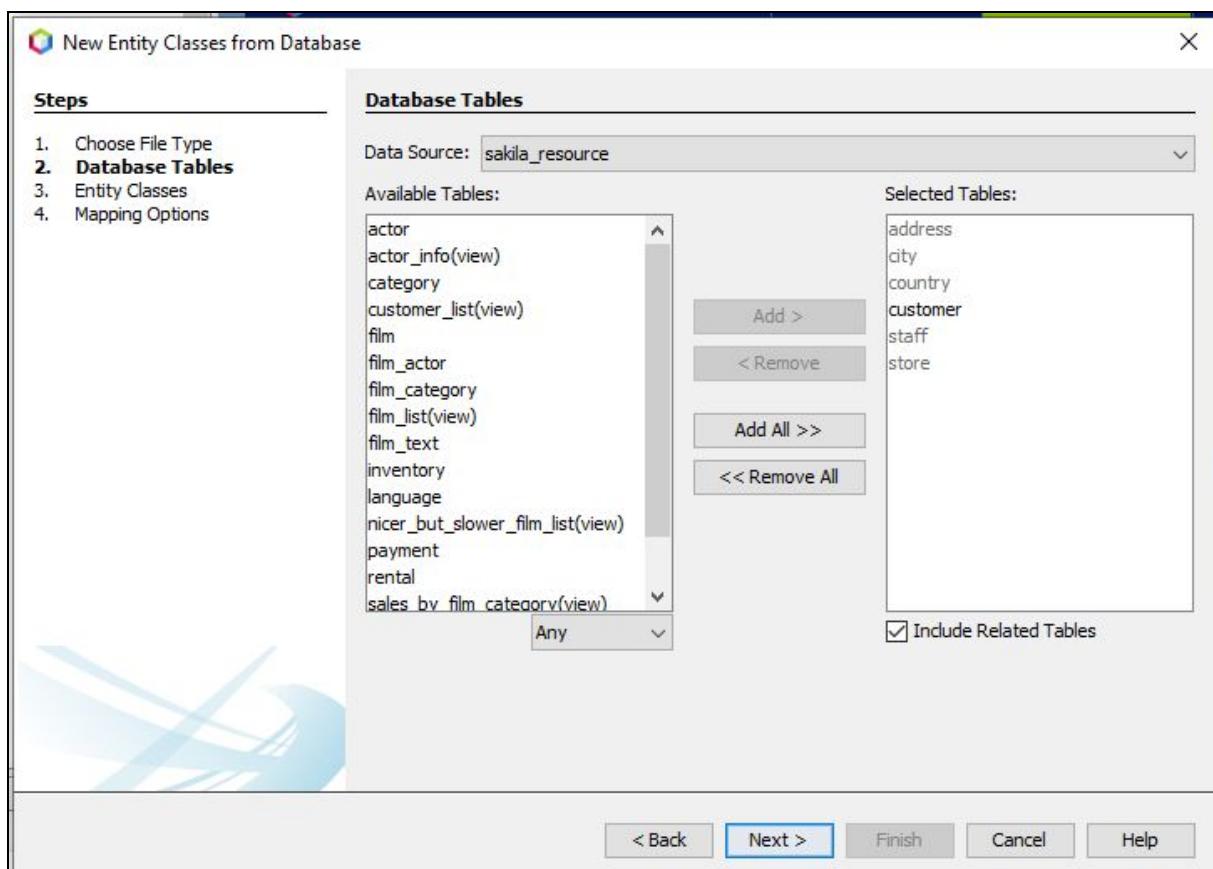


Figura.28

click en "Next", ahora se puede cambiar los nombres de las Java entity classes que se generarán, es recomendable dejar todos los valores por defecto. Una buena práctica consiste en ponerlas en un paquete llamado "entities", como se muestra en la figura 29

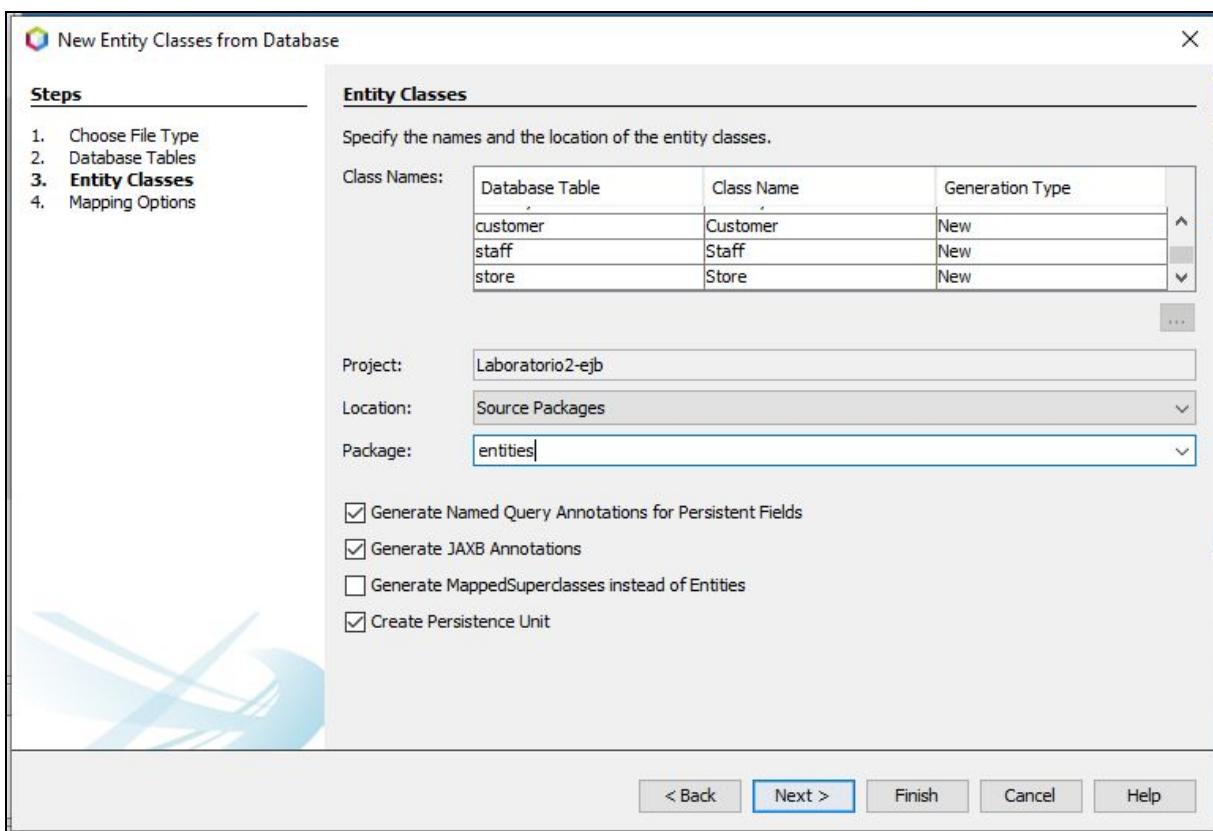


Figura.29

click en "Next". Dejar todos los valores por defecto

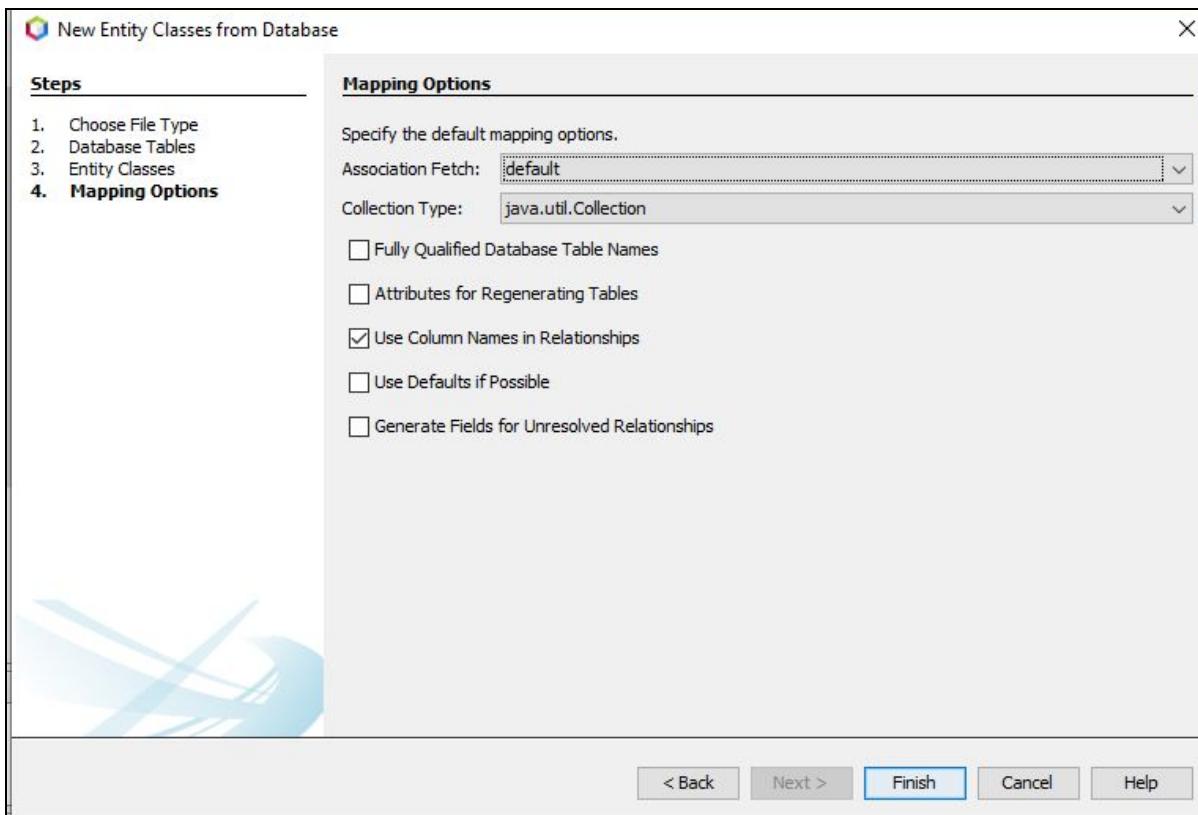


Figura.30

Clic en “finish”. Esto generará las clases Java en un nuevo paquete llamado “entities”

Además, un archivo llamado persistence.xml que aparece dentro de “configuration file”, el cual corresponde a la unidad de persistencia del proyecto.

La unidad de persistencia es la responsable de todas las operaciones de la base de datos, en este objeto se generará todo el código SQL que está por debajo. En la clase customer.java se deben agregar/verificar las siguientes líneas de código



Figura.31

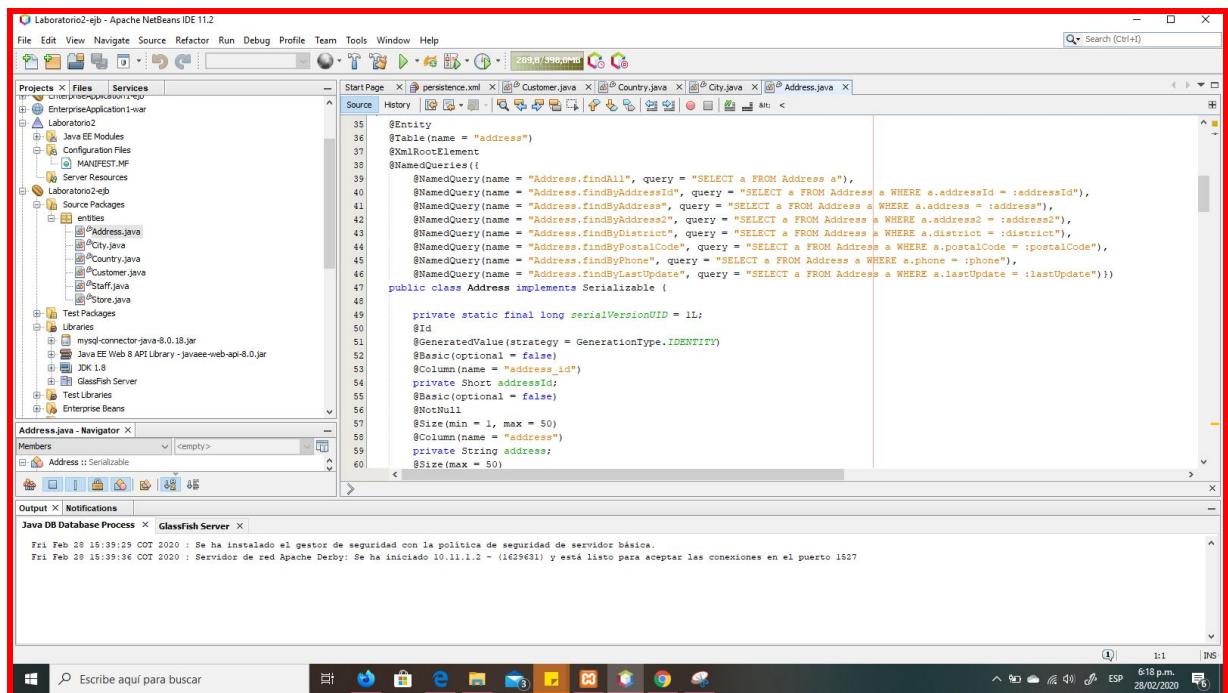
```

<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
    <persistence-unit name="Laboratorio2-ejbPU" transaction-type="JTA">
        <jta-data-source>sakila_resource</jta-data-source>
        <exclude-unlisted-classes>false</exclude-unlisted-classes>
        <properties/>
    </persistence-unit>
</persistence>

```

Figura.32

Una vista del código generado por nuestro IDE, donde podemos apreciar las queries que ha generado y las anotaciones necesarias para que los atributos de la clase sean mapeados a los campos de la tabla de la DB



Doble click en persistence.xml, aparece la siguiente ventana:

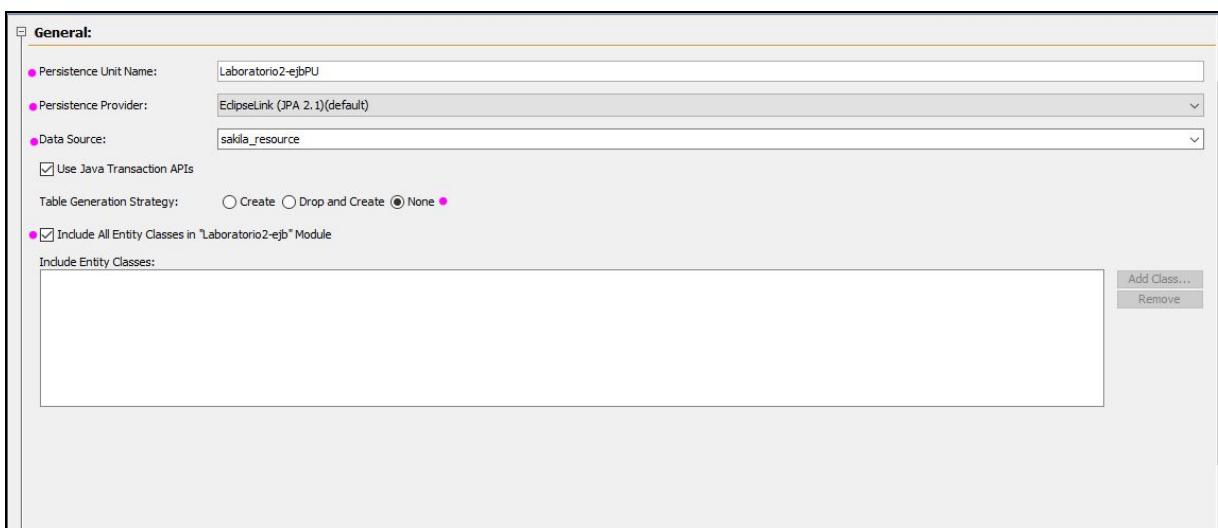


Figura.33

Para ver en detalles, se encuentra que:

1. No se autorizó la aplicación para crear nuevas tablas (None)
2. Se utilizarán las transacciones acceder/modificación de datos en la base de datos
3. Esta unidad de persistencia trabaja con la base de datos sakila, usa EclipseLink como proveedor de persistencia y mapeara todas las entity class desde las tablas de la BD mediante el uso de un ORM (Mapeo Objeto Relacional).

## CREACIÓN DE UN CLIENTE FACADE (Fachada)

Se añadirá al proyecto un **DAO / Facade** que se hará cargo de todas las operaciones CRUD sobre las entidades del cliente (Customer entities), si se quiere insertar, actualizar, eliminar o buscar entidades en la base de datos sakila tendrá que utilizar este facade. Este facade se implementa como un stateless session bean EJB

Click derecho en el proyecto EJB New/Session Bean. (Figura 34 )

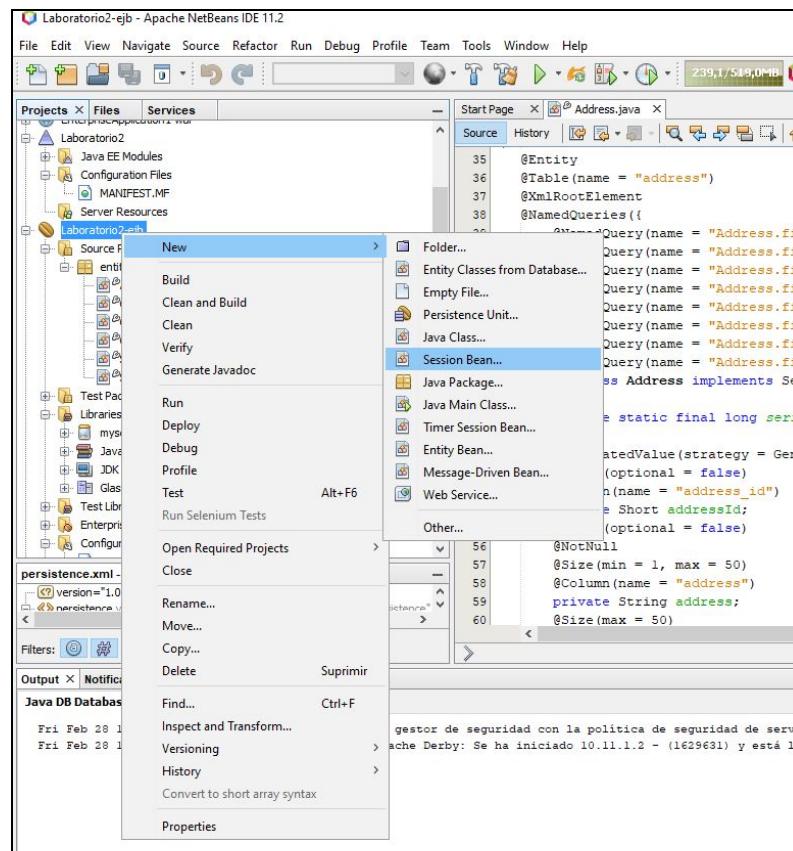


Figura.34

Dar un nombre al Facade, en este caso se nombró CustomerManager y se pone en un paquete llamado "session" con el fin de diferenciarlo dejamos la opción sesión type como Stateless. Es opcional activar la creación de interfaces pues en esta práctica no las usaremos. Damos clic sobre el botón finish..(Figura 35)

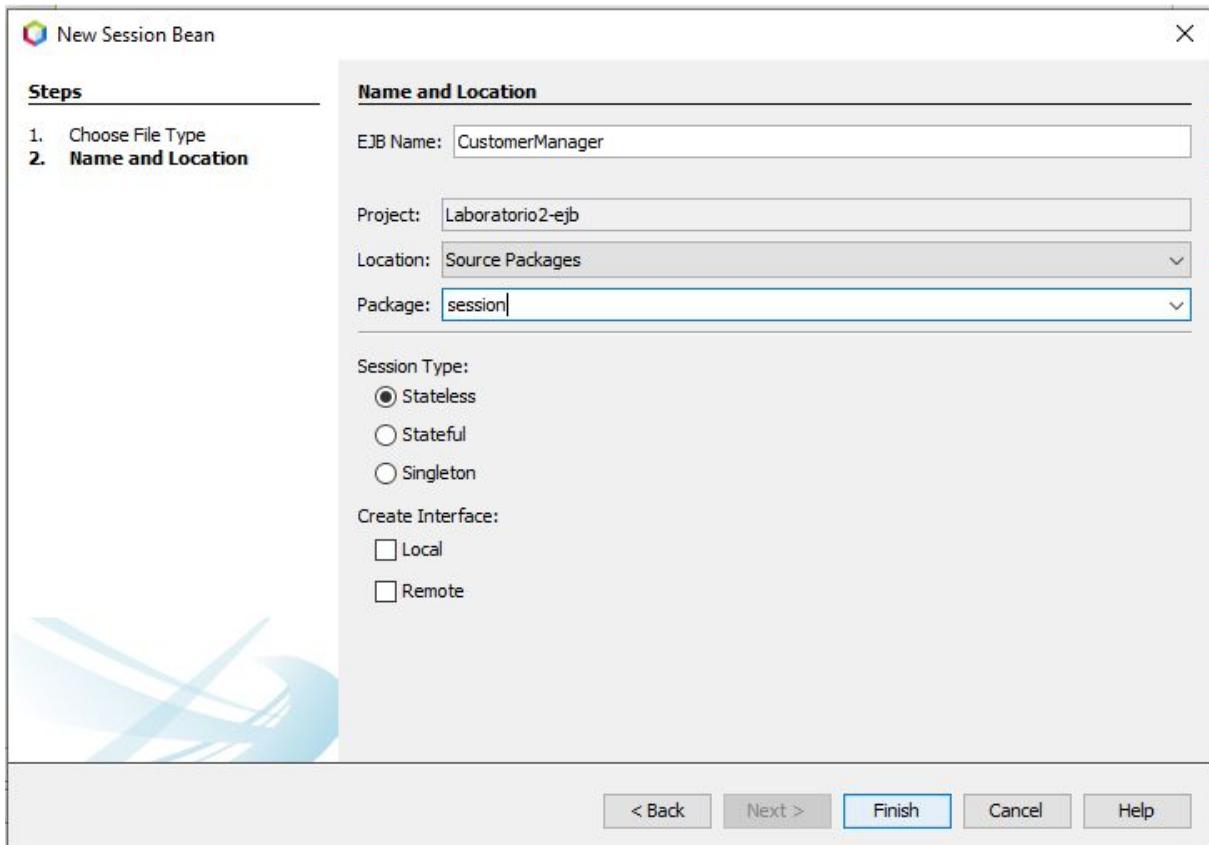


Figura.35

## Añadir algunos métodos de negocio en el Facade

Doble click en "**CustomerManager.java**" en el proyecto de manera que muestre el código fuente en el editor. click en algún lugar de la clase, después de las llaves "{", se da click derecho / **insert code**, en el menú emergente elegimos la opción **Add Bussiness Method** para agregar automáticamente los métodos de negocio



Figura.36

---

Agregamos el método **getAllCustomers** el cual será encargado de retornar la lista de todos los clientes registrados en el sistema por lo tanto su tipo de retorno es `List<Customer>` y no necesita parámetros..

De la misma forma que el paso anterior, agregamos el método **updateCustomer** configurado que retorna un objeto de tipo **Customer** que representa el Customer que ha sido modificado con el método, recibe como parámetro un objeto de la clase **Customer**.

Luego unimos los dos paquetes que se han generado hasta el momento (**session y entity**) por medio de un contexto de persistencia, es decir usar nuestro **entity manager**, hacemos click derecho sobre un espacio vacío de la clase elegimos **Insert Code** y elegimos la opción **Use Entity Manager**.

Ahora el código luce de esta manera (*figura 37*), observamos que se ha agregado al código la anotación `@PersistenceContext` y se ha creado un EntityManager el cual nos permitirá acceder a los métodos que permiten gestionar la persistencia de nuestra aplicación (métodos CRUD a la DB).

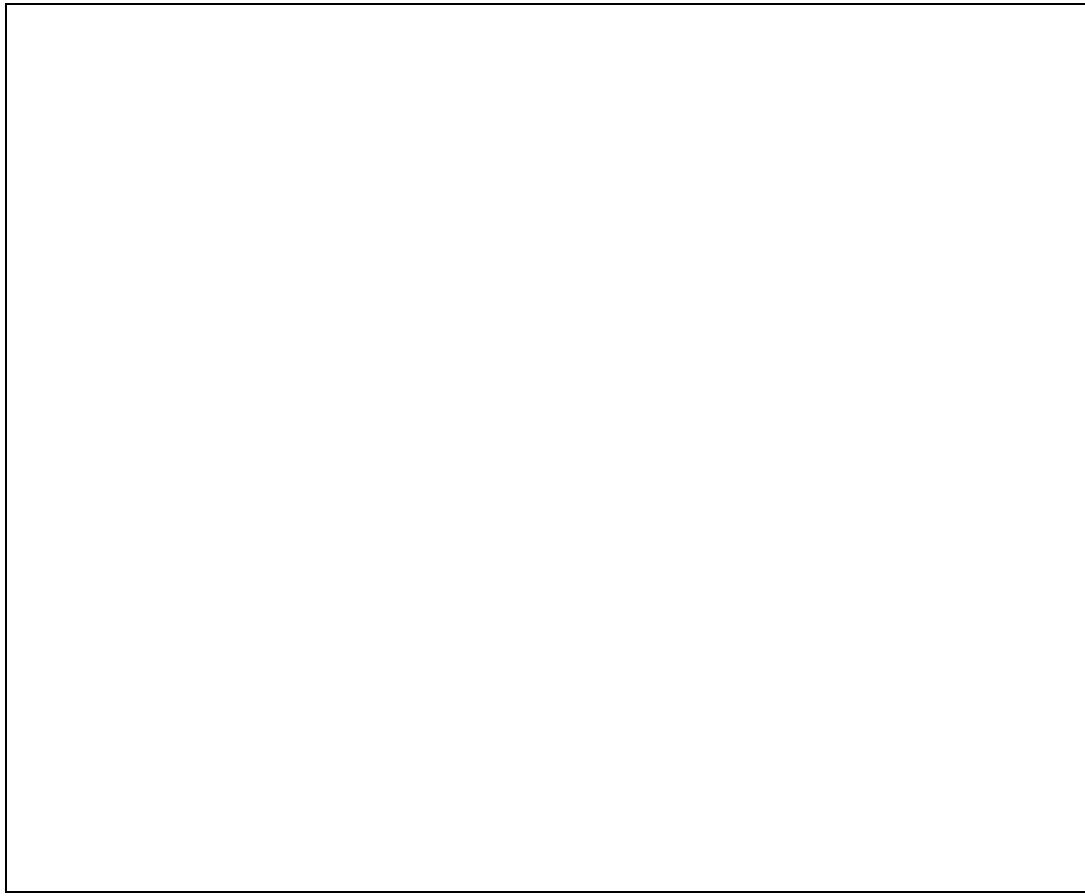
```

11 import javax.ejb.LocalBean;
12 import javax.persistence.EntityManager;
13 import javax.persistence.PersistenceContext;
14
15 /**
16 *
17 * @author yesen
18 */
19 @Stateless
20 @LocalBean
21 public class CustomerManager {
22
23     // Add business logic below. (Right-click in editor and choose
24     // "Insert Code > Add Business Method")
25     @PersistenceContext(unitName = "Laboratorio2-ejbPU")
26     private EntityManager em;
27
28
29     public List<Customer> getAllCustomers() {
30         return null;
31     }
32
33     public Customer updateCustomer(Customer customer) {
34         return null;
35     }
36
37     public void persist(Object object) {
38         em.persist(object);
39     }
40 }

```

Figura.37

A continuación modificaremos los métodos que hemos agregado anteriormente para hacer las consultas que deseamos con cada uno, haremos uso de losNamedQuery que aparecen al principio de la clase Customer, en particular usaremos **"Customer.findAll"**.



*Figura.38*

Los dos métodos que aparecen en la figura anterior se implementaron.

En el primero se ejecuta una "consulta con nombre", cuyo nombre es "Customer.findAll" (esta consulta se encuentra al principio del archivo Customer.java), esto es equivalente a un select \* from CUSTOMER. Pero para este caso como se está trabajando con objetos se debe retornar una lista de clientes (Customer) y no tuplas.

En el Segundo método update(Customer customer)... se actualiza el contenido de la Base de Datos mediante el objeto cliente que se pasa como parámetro; El em.merge(customer) hace precisamente eso: "toma este objeto en memoria, mira si existe en la Base de Datos, a continuación actualiza la columna en la BD.. Los objetos tiene una clave primaria (en este caso, mirar el atributo id en la clase Customer.java).

## CREAR PARTE WEB

Se utilizará el framework JSF 2.2 MVC (Modelo Vista Controlador), para añadirlo se hace clic derecho en el Proyecto Web / propiedades y click en framework. *Add Java Server Faces*.

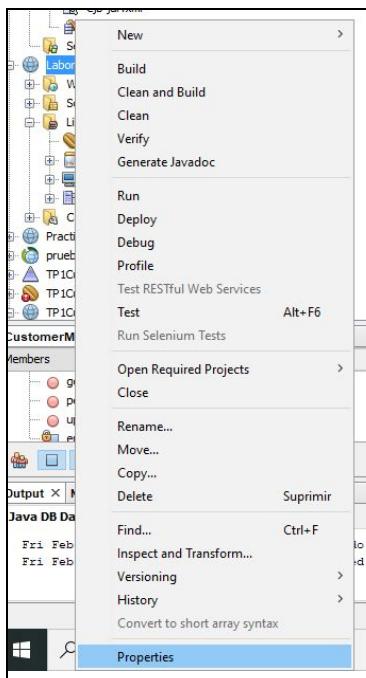


Figura.39

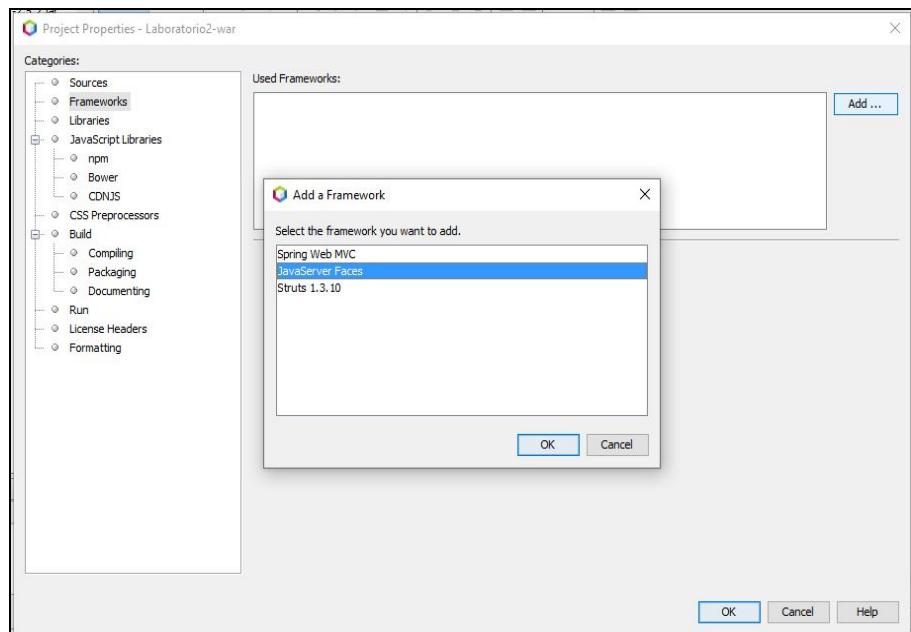
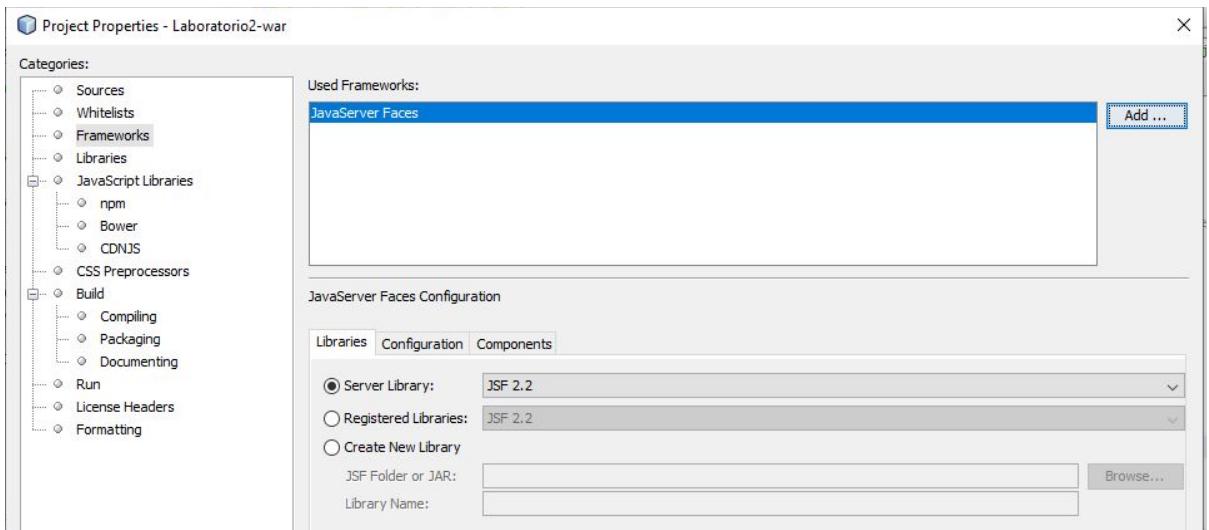
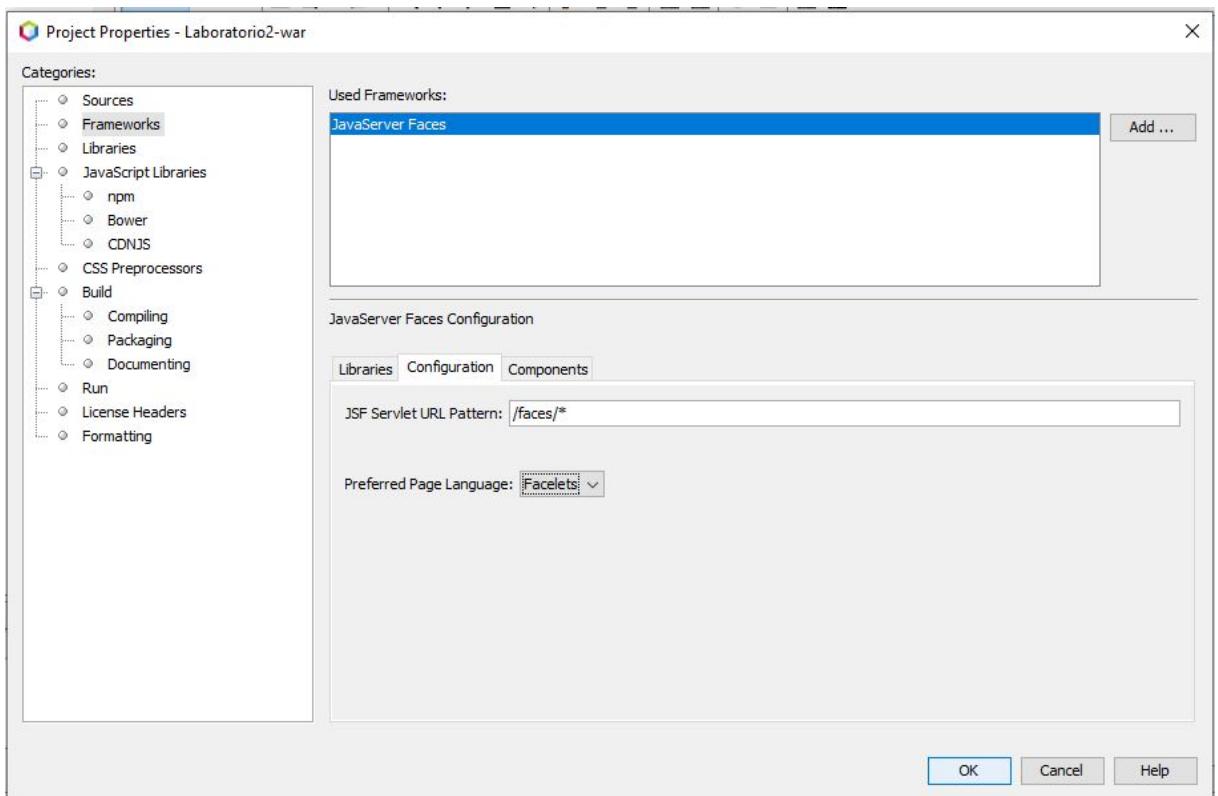


Figura.40

Chequear que se esté utilizando JSF 2.2.

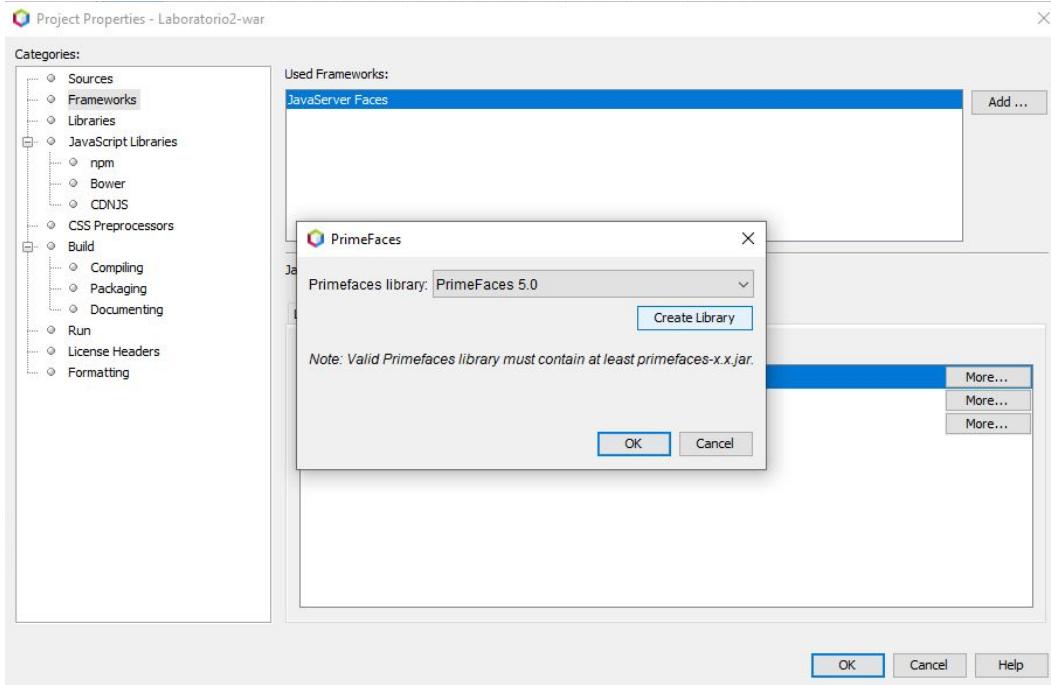


En la pestaña Configuración chequear que esté seleccionado el lenguaje de página FACELETS, este es el tipo de lenguaje (etiquetas) que serán tomadas como válidas dentro de un archivo JSF.

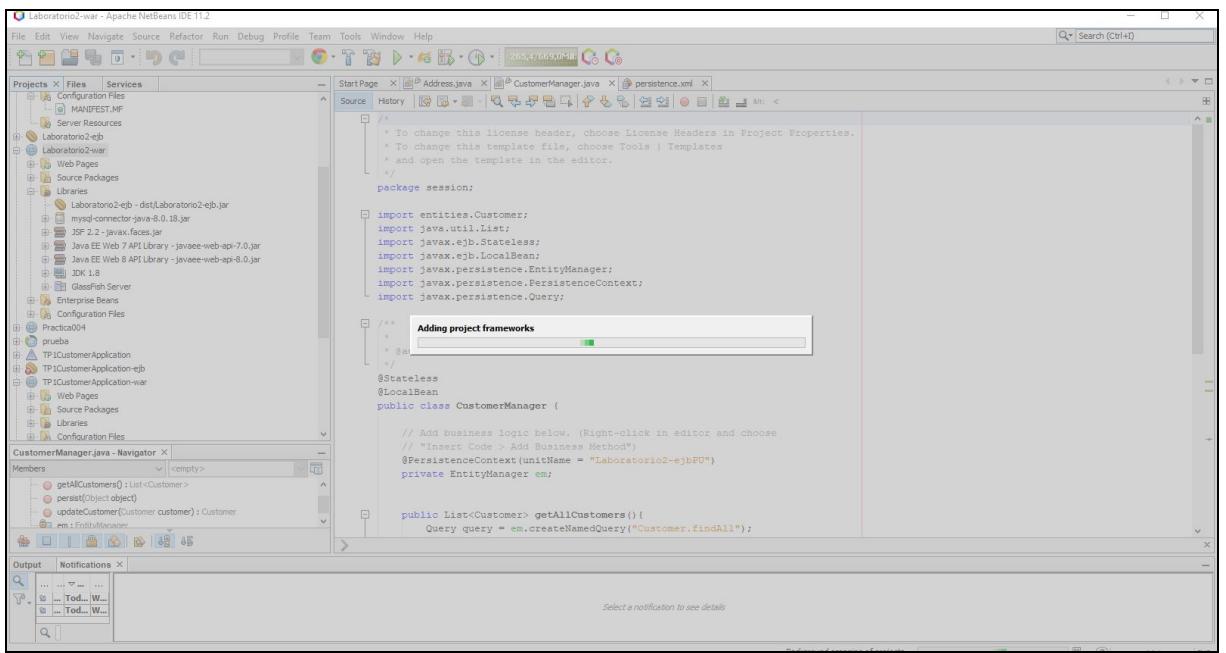


En la pestaña Componentes elegir Primefaces, dado que es el componente básico que necesitamos, los otros son opcionales.

Aparece un error: SF library PrimeFaces not set up properly: Searching valid Primefaces library. Please wait..



Se queda cargando

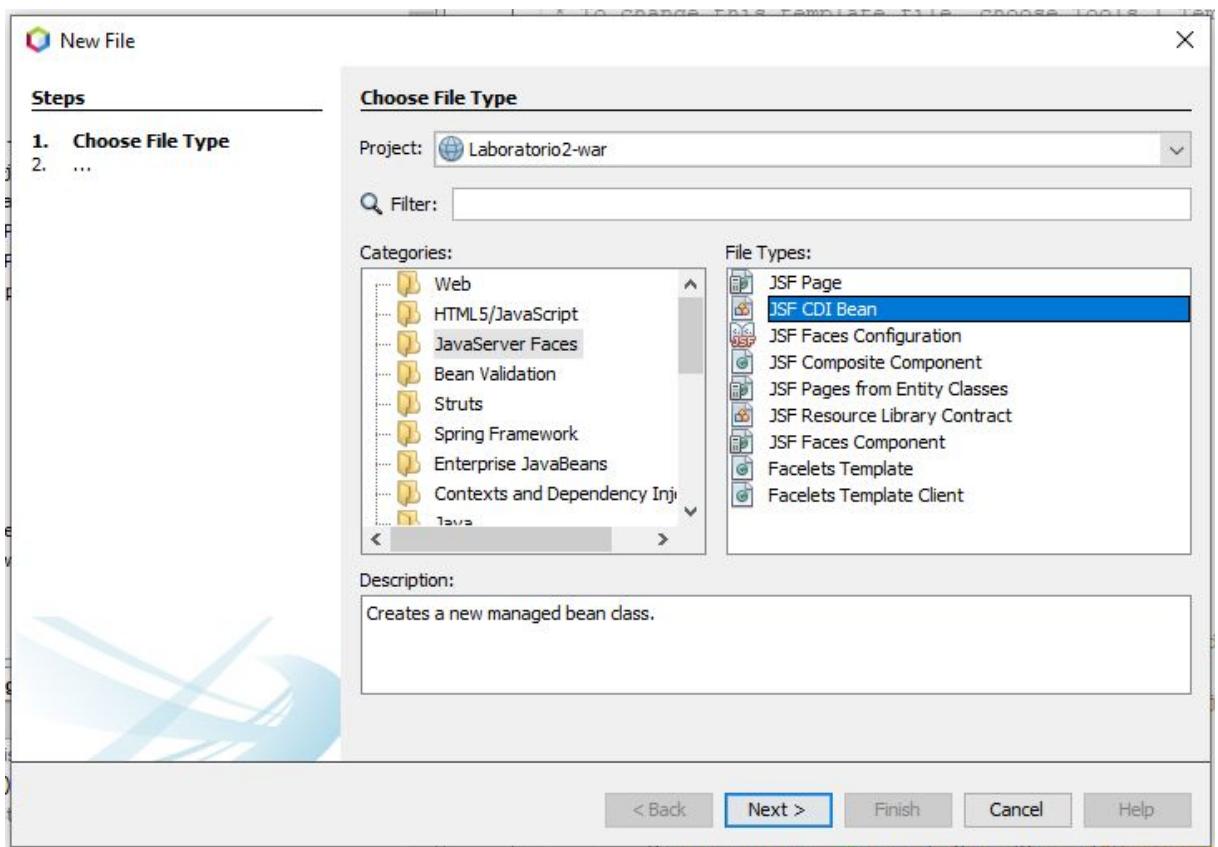


## Creación de un JSF managed bean como un control

Click derecho en el proyecto y **add/java server faces / jsf managed bean**.

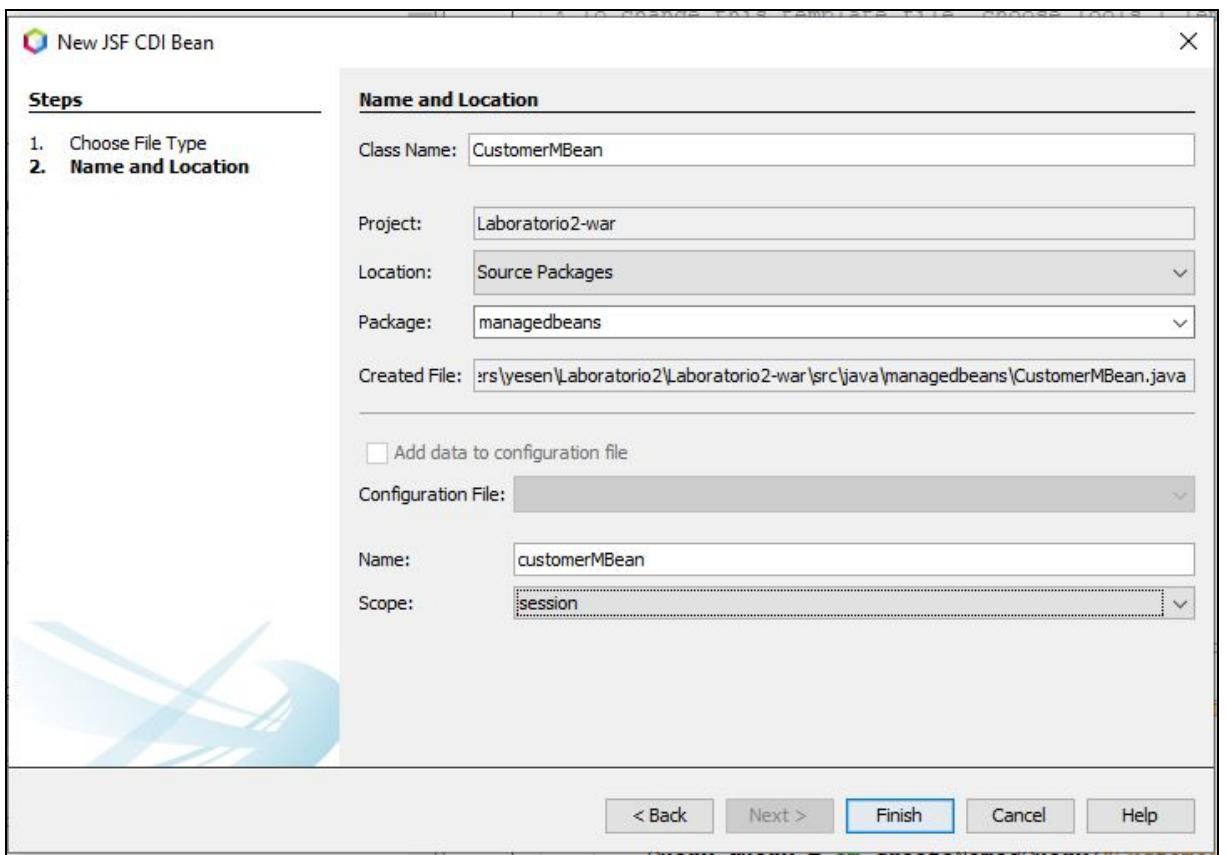
<https://www.apuntesdejava.com/2013/08/javaee-7-adios-managedbean-bienvenido.html>

En este caso, por tener en JDK 8 y JEE 8 , los ManagedBean ( o también llamados Bean Administrados) ya no se usarán. Estos beans se usaban como Backend de las páginas JSF. En su lugar, se usarán inyección de dependencias, ya que se instancian de la misma manera como lo hacía los ManagedBean.



Colocar un nombre al JSF bean, este se encargará de ser el controlador de la interfaz gráfica de usuario GUI para los clientes, en este ejemplo recibe el nombre de CustomerMBean y será añadido al paquete llamado managedbeans. Además se eligió que el ámbito será la sesión HTTP y en el alcance (scope) se eligió la opción "session".

Para hacerlo con anotaciones se debe dejar con la configuración que aparece en la siguiente figura



Se debe tener un CustomerMBean.java que se parezca al siguiente:

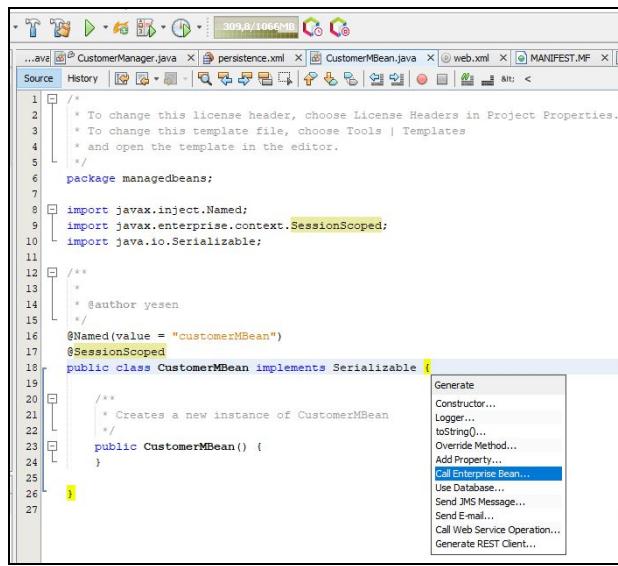
```
6  package managedbeans;
7
8  import javax.inject.Named;
9  import javax.enterprise.context.SessionScoped;
10 import java.io.Serializable;
11
12 /**
13  *
14  * @author yesen
15  */
16 @Named(value = "customerMBean")
17 @SessionScoped
18 public class CustomerMBean implements Serializable {
19
20     /**
21      * Creates a new instance of CustomerMBean
22      */
23     public CustomerMBean() {
24
25     }
26 }
```

Anotación @SessionScoped, esta anotación significa que:

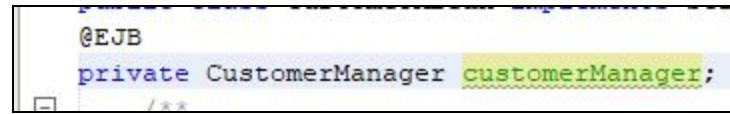
1. El bean durará lo que dure la sesión HTTP
2. Sus atributos/propiedades se almacenarán en la sesión HTTP, se va a utilizar las propiedades de esta clase como modelo para la visualización y modificación en la página JSF

### Añadir funcionalidades al JSF bean

Nos ubicamos en la clase CustomerMBean y agregamos implements Serializable para que la clase sea serializable. A continuación haremos la conexión entre el back-end y el front-end por medio de un Enterprise Java Bean. Hacemos click derecho sobre un espacio vacío de la clase elegimos Insert Code y elegimos la opción Call Enterprise Bean , y aparecen estas dos líneas:



```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6 package managedbeans;
7
8 import javax.inject.Named;
9 import javax.enterprise.context.SessionScoped;
10 import java.io.Serializable;
11
12 /**
13  * @author yesen
14  */
15 @Named(value = "customerMBean")
16 @SessionScoped
17 public class CustomerMBean implements Serializable {
18
19     /**
20      * Creates a new instance of CustomerMBean
21      */
22     public CustomerMBean() {
23
24     }
25
26 }
27
```



```
@EJB
private CustomerManager customerManager;
```

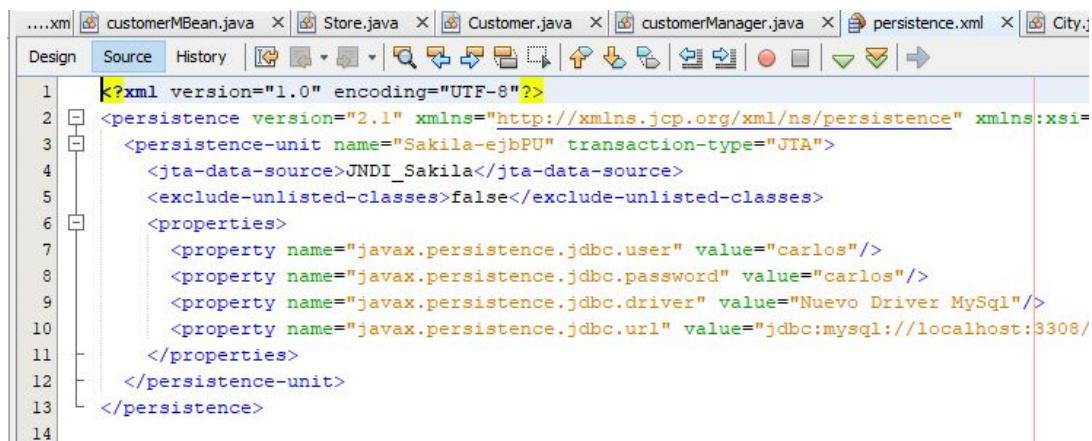
Comenzamos a modificar esta clase para poder manejar las acciones que se generan en las vistas. Además agregamos los métodos para retornar como String las acciones que se generan en las vistas, también agregamos un método refresh() que nos permite refrescar clientes

The screenshot shows the NetBeans IDE interface with the following details:

- Project Structure:** The left pane displays a tree view of the project structure, with nodes for 'Customer' and 'Customer' under 'new'.
- Code Editor:** The main editor area contains Java code for a CustomerMBean class. The cursor is positioned at the end of the line `customer_id(int id){`.
- Code Completion:** A context menu is open over the cursor, listing various code generation options:
  - Generate
  - p Constructor...
  - Logger...
  - } Getter...
  - p Setter...
  - Getter and Setter...** (This option is highlighted with a blue selection bar.)
  - } equals() and hashCode()
  - toString()
  - p Delegate Method...
  - Override Method...
  - } Add Property...
  - p Use Entity Manager...
  - Call Enterprise Bean...
  - } Use Database...
  - Send JMS Message...
  - Send E-mail...
  - Call Web Service Operation...
- Toolbars:** The top toolbar includes icons for Source, History, and various navigation and search functions.
- Tab Bar:** The tabs at the top are labeled ...htm, CustomerMBean.java, build-impl.xml, persistence.xml, and persistence.xml.

Se generaron los getter y Setter de CustomerMBean.

## Creación de la Unidad de Persistencia

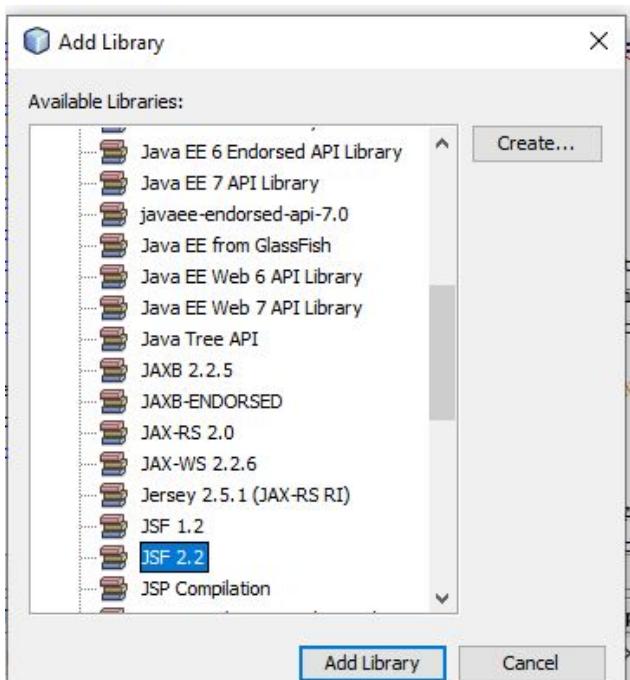


The screenshot shows a Java IDE interface with multiple tabs at the top: ....xml, customerMBean.java, Store.java, Customer.java, customerManager.java, persistence.xml, and City. The persistence.xml tab is active. Below the tabs is a toolbar with various icons. The main area displays the XML code for the persistence unit:

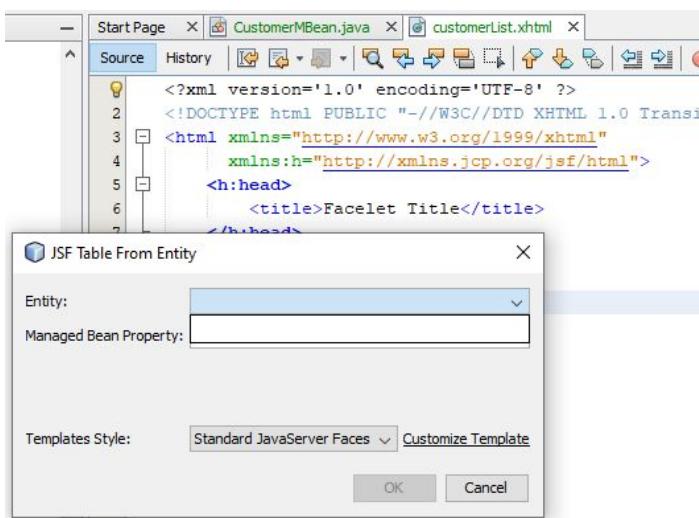
```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi=
  <persistence-unit name="Sakila-ejbPU" transaction-type="JTA">
    <jta-data-source>JNDI_Sakila</jta-data-source>
    <exclude-unlisted-classes>false</exclude-unlisted-classes>
    <properties>
      <property name="javax.persistence.jdbc.user" value="carlos"/>
      <property name="javax.persistence.jdbc.password" value="carlos"/>
      <property name="javax.persistence.jdbc.driver" value="Nuevo Driver MySql"/>
      <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3308/"/>
    </properties>
  </persistence-unit>
</persistence>
```

La unidad de persistencia es la responsable de todas las operaciones de la base de datos, en este objeto se generará todo el código SQL que está por debajo.

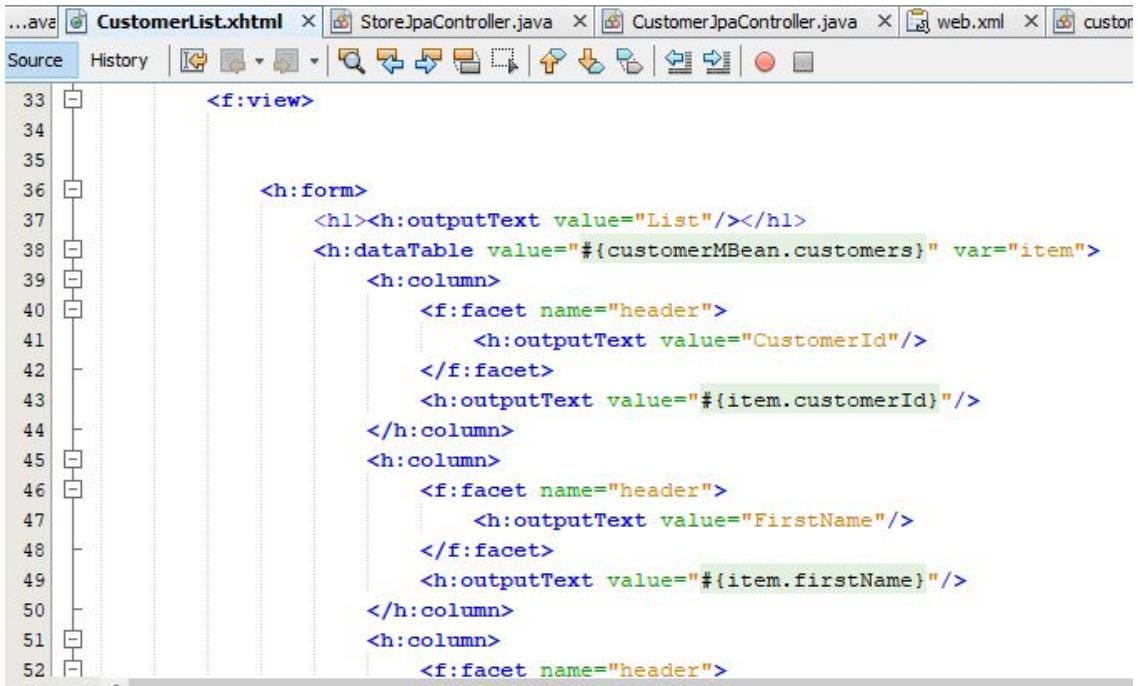
## Se agregó la librería JSF 2.2



## CREACIÓN DEL JSF DATA TABLE:



Ingresar a cliente (Customer) como la entidad (entity) y seleccionar customerMBean.customers como el nombre de las propiedades en el JSF bean que corresponde a este entity en el JSF bean.



```
33 <f:view>
34
35
36     <h:form>
37         <h1><h:outputText value="List"/></h1>
38         <h:dataTable value="#{customerMBean.customers}" var="item">
39             <h:column>
40                 <f:facet name="header">
41                     <h:outputText value="CustomerId"/>
42                 </f:facet>
43                     <h:outputText value="#{item.customerId}"/>
44             </h:column>
45             <h:column>
46                 <f:facet name="header">
47                     <h:outputText value="FirstName"/>
48                 </f:facet>
49                     <h:outputText value="#{item.firstName}"/>
50             </h:column>
51             <h:column>
52                 <f:facet name="header">
```

Se genera el código automáticamente para la tabla JSF

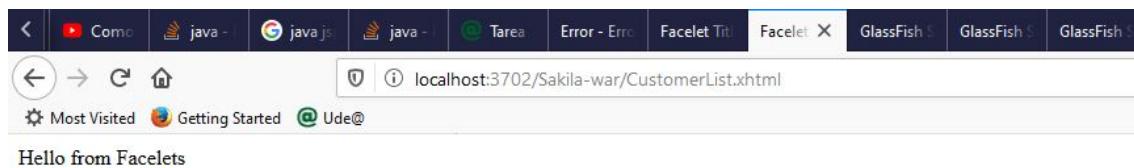
## Despliegue

Click derecho en el proyecto EAR, el que tiene un icono de triángulo azul, y seleccione "Ejecutar"- "Run". Esto debería mostrar la página "index.jsp" por defecto:



Hello from Facelets  
[Primefaces welcome page](#)

Al cambiar la URL por: <http://localhost:3702/Sakila-war/CustomerList.xhtml>



La aplicación se ejecuta mostrando la vista pero no muestra los datos de la base de datos. Se procede a buscar información en internet sobre el problema y a realizar depuración del proyecto pero no se determinó cual era el problema.

---

## CONCLUSIONES.

JPA es un API que proporciona una manera de manejar la capa de persistencia en el lenguaje JAVA, provee una funcionalidad con lógica de objeto y no de datos.

Las anotaciones en las clases JAVA para el manejo ORM nos ahorran tener que mantener archivos .xml en nuestros proyectos.

El trabajo con los Beans empresariales de Java ahorra tiempo de codificación, siendo así más productivos y generando una mayor probabilidad de éxito en los proyectos de software manejados con la plataforma JEE.

El uso de librerías en el front-end nos permite generar vistas con una muy buena interacción para el usuario.

Con la realización de este proyecto hemos estudiado las tecnologías y herramientas existentes para el desarrollo de aplicaciones empresariales.

Sin embargo, presentamos inconvenientes a la hora de listar, por tanto se puede concluir que a pesar de haber dispuesto los conocimientos adquiridos a lo largo de documentarnos, existen falencias

## ANEXOS.

se toma como alternativa subir el proyecto a GitHub, en la que se evidencia fecha, hora, comentarios y demás, ruta:

[https://github.com/carlossen1978/Laboratorio2\\_Arq.git](https://github.com/carlossen1978/Laboratorio2_Arq.git)

## REFERENCIAS

[1]. <https://www.youtube.com/watch?v=E6fCNPWg9JA>

[2].<https://www.youtube.com/watch?v=m8xPbnmrdrk>

[3].<https://dev.mysql.com/doc/index-other.html>

[4].<https://dev.mysql.com/doc/sakila/en/sakila-installation.html>

