



**NOMBRE:** PABLO JESUS ZAMBRANO PATIÑO

**CURSO:** PROGRAMACION 2

**INGENIERO:** ING. FREDDY CATACUAMBA

**FECHA:** 29/03/2021

**PROYECTO**

## INDICE

INTRODUCCION:.....	3
OBJETIVOS:.....	3
• OBJETIVOS GENERALES: .....	3
• OBJETIVOS ESPECIFICOS .....	3
ANTECEDENTES: .....	3
MARCO TEORICO: .....	4
NetBeans:.....	4
PostgreSQL:.....	4
Día: .....	4
Java EE: .....	5
Formato EAR:.....	5
Formato EJB:.....	5
Formato WEB:.....	6
GlassFish: .....	6
DESARROLLO:.....	6
CONCLUSIONES:.....	20
RECOMENDACIONES: .....	20

## INTRODUCCION:

El presente informe tiene como propósito evidenciar el proyecto que se realizó durante esta semana que tuvo el lugar la orden de la persona encargada de dicho curso **“Lenguaje de programación 2”**. En este, se presenta el análisis de lo que sería un avance breve del proyecto que se hará la entrega una semana después de presentar dicho informe. Este informe tiene como objetivo indicar el proceso de creación de cada uno de los componentes aprendidos mediante este curso, como la creación de la estructura base de un sitio web de una heladería.

## OBJETIVOS:

- **OBJETIVOS GENERALES:** El desarrollo de un sitio web XHTML sobre una tienda de helados, el Objetivo general de dicho proyecto es el de implementar los conocimientos aprendidos para el desarrollo de la aplicación en cuestión.
- **OBJETIVOS ESPECIFICOS:** Desarrollar un sitio web alojado en el servidor local con lenguaje de programación Java, en la cual implementaremos varias herramientas y librerías como; el formato de archivo “EAR” que es utilizado por **Java EE** para empaquetar uno o más módulos en un solo archivo. En este caso implementamos el tipo WAR y JPA, para el desarrollo de nuestro sitio web.

## ANTECEDENTES:

Al hacer uso de los conocimientos ya adquiridos en este curso de “Programación 2”. Al desarrollar dicho proyecto ha sido un reto personal, por el simple hecho de ser un tema nuevo para el uso de programación en Java orientada a objetos.

La aplicación fue desarrollada con las funcionalidades requeridas por el ingeniero a cargo de este curso. Que fue el de desarrollar una heladería cumpliendo con los temas ya adquirido que fueron aprendidos mediante se iba transcurriendo el curso. Como el manejo de herramientas como; nuestro IDE de confianza **Netbeans**, nuestra herramienta de confianza para la administración de datos **Postgres** y para el desarrollo de las diagramas de clases UML, nos ayudamos con la herramienta **Dia**.

La aplicación consiste en una tienda online de ventas de Helados donde tiene que haber, una gestión de clientes, gestión de categoría de productos, gestión de productos, gestión de bodega y la funcionalidad de la ventas de los productos de la heladería.

## MARCO TEORICO:

**NetBeans:** Es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las API de NetBeans y un archivo especial (manifest file) que lo identifica como módulo.

## PostgreSQL:

Es un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto, publicado bajo la licencia PostgreSQL,

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre o apoyados por organizaciones comerciales.

## Día:

Es una aplicación informática de propósito general para la creación de diagramas, creada originalmente por Alexander Larsson, y desarrollada como parte del proyecto GNOME . Está concebido de forma modular, con diferentes paquetes de formas para diferentes necesidades.

#### Java EE:

Es una plataforma de programación parte de la Plataforma Java para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java. Permite utilizar arquitecturas de N capas distribuidas y se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una especificación. Similar a otras especificaciones del Java Community Process (JCP), Java EE es también considerado informalmente como un estándar debido a que los proveedores deben de cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE; estandarizado por JCP.

#### Formato EAR:

Es un formato de archivo utilizado por Java EE para empaquetar uno o más módulos en un solo archivo de modo que la implementación de los diversos módulos en un servidor de aplicaciones se realice de manera simultánea y coherente.

#### Formato EJB:

Es posible desarrollar componentes (*enterprise beans*) que luego puedes reutilizar y ensamblar en distintas aplicaciones que tengas que hacer para la empresa.

Su especificación detalla cómo los servidores de aplicaciones proveen objetos desde el lado del servidor. Proporcionan un modelo de componentes distribuido estándar del lado del servidor. El objetivo de los EJB es dotar al programador de un modelo que le permita abstraerse de los problemas generales de una aplicación empresarial (conurrencia, transacciones, persistencia,

seguridad, etc.) para centrarse en el desarrollo de la lógica de negocio en sí. El hecho de estar basado en componentes permite que éstos sean flexibles y sobre todo reutilizables.

#### Formato WEB:

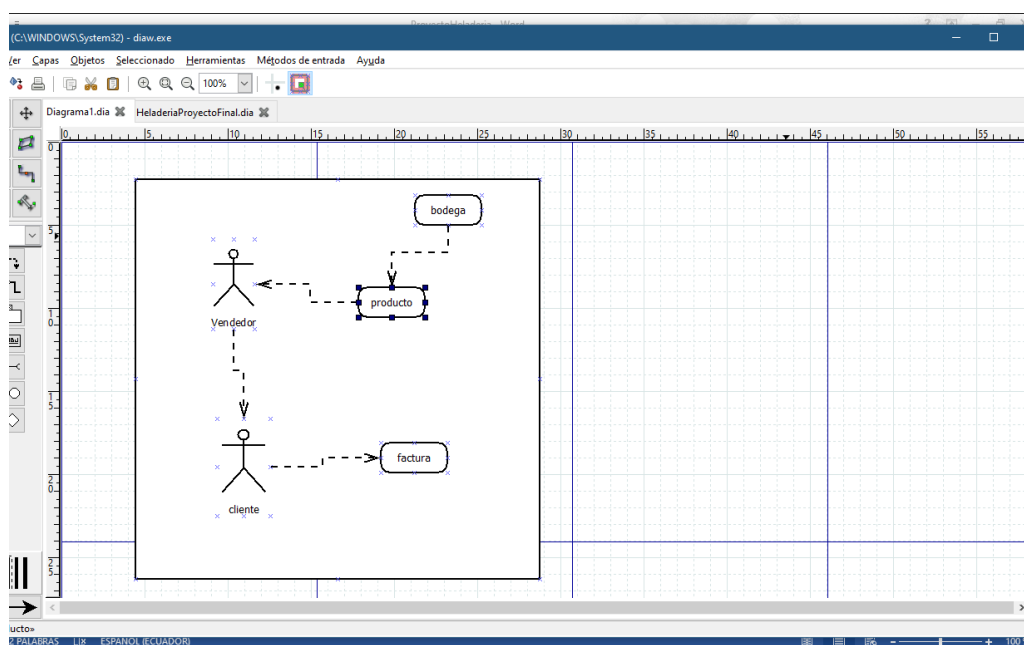
Permite arrancar aplicaciones Java que están en un servidor web de aplicaciones comprobando previamente si el cliente tiene la versión actualizada de dicha aplicación. El arranque de dichas aplicaciones puede ser efectuado mediante enlaces en una página web o bien a través de enlaces en el escritorio cliente. Mediante esta tecnología se asegura que una aplicación es distribuida siempre en su última versión.

#### GlassFish:

Es un servidor de aplicaciones de software libre desarrollado por Sun Microsystems, compañía adquirida por Oracle Corporation, que implementa las tecnologías definidas en la plataforma Java EE y permite ejecutar aplicaciones que siguen esta especificación. Es gratuito, de código libre y se distribuye bajo un licenciamiento dual a través de la licencia CDDL y la GNU GPL.

## DESARROLLO:

Diagrama de clases de uso



Paso 1: Optamos por utilizar el software “Día” para realizar tanto nuestro diagrama de clases de uso y nuestro diagrama de clases, podemos observar que en este avance se encuentra ubicado cuatro clases principales que darán acción a nuestro sitio web de una “Heladería”.

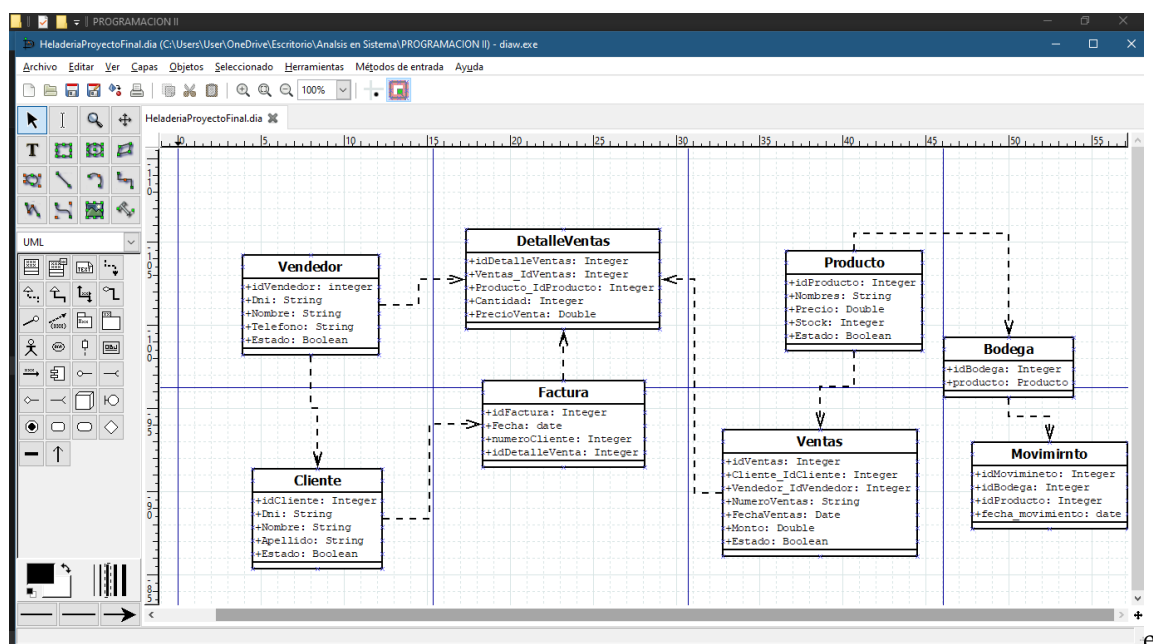
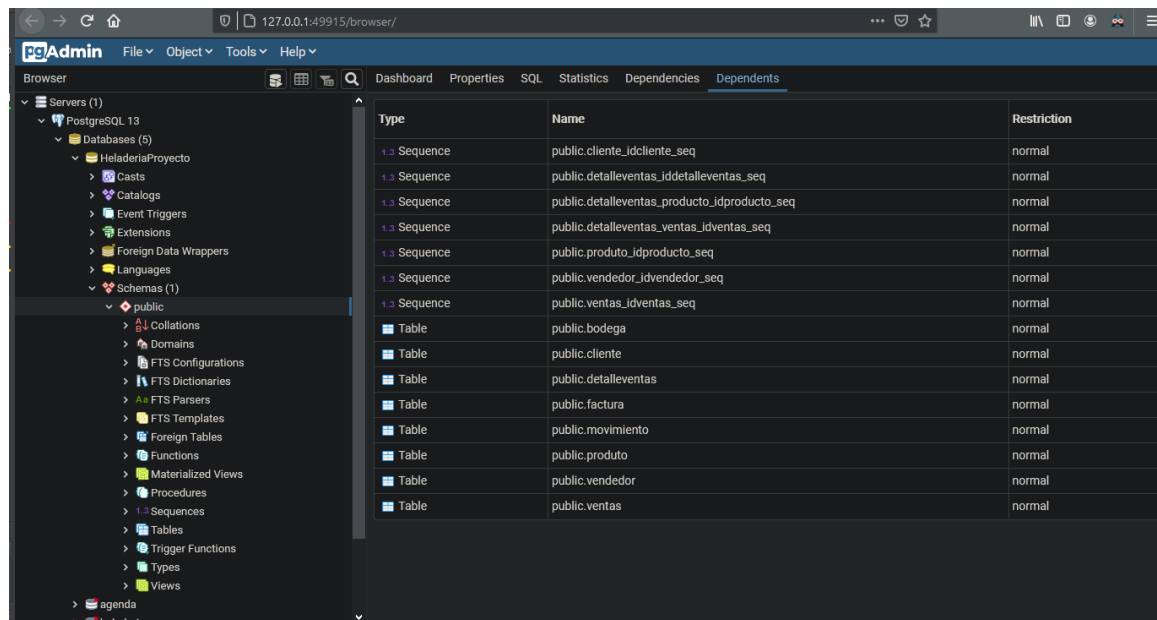


Diagrama de clases

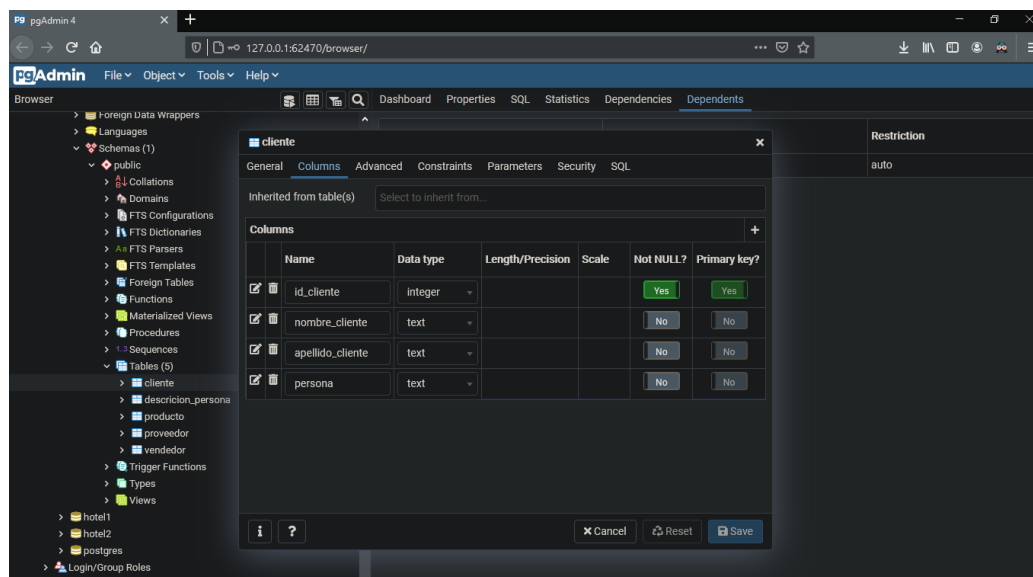
Paso 2: Aquí podemos observar en nuestro diagrama de clases que hemos dado a nuestras clases sus principales objetos principales que serán esenciales para su respectivo desarrollo a futuro.

## Base de datos

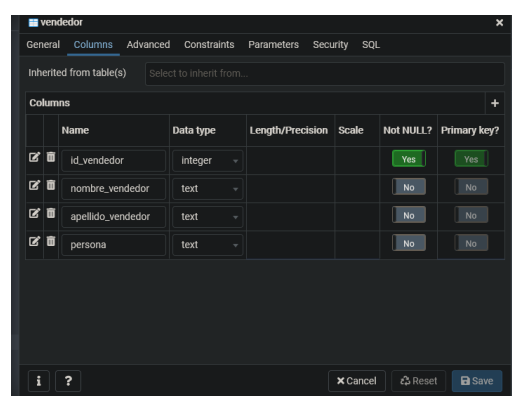
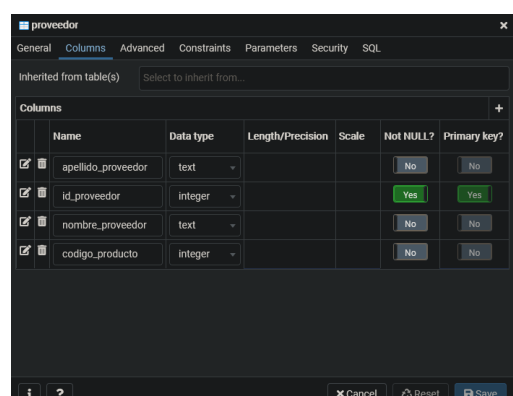
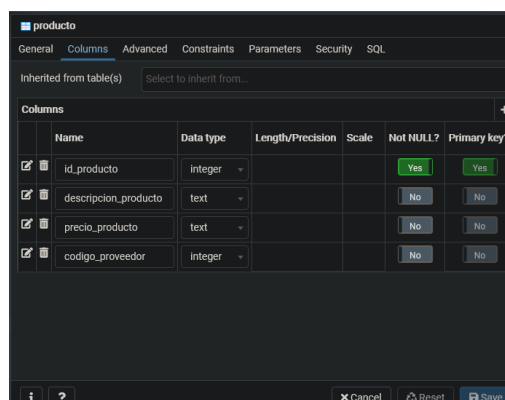
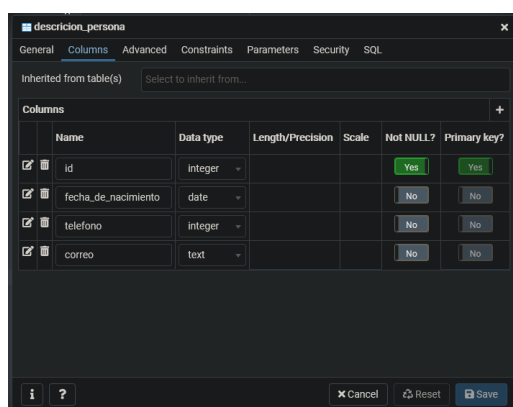


Pasó 3: He utilizado mi software de confianza para administración de datos que es Postgres. Empezamos con la creación de nuestra base de datos y con sus respectivas tablas y relaciones, que anteriormente hemos desarrollamos en nuestros diagramas de clases, que nos servirá como guía. Y a cada creamos objetos con su respetivo tipo.

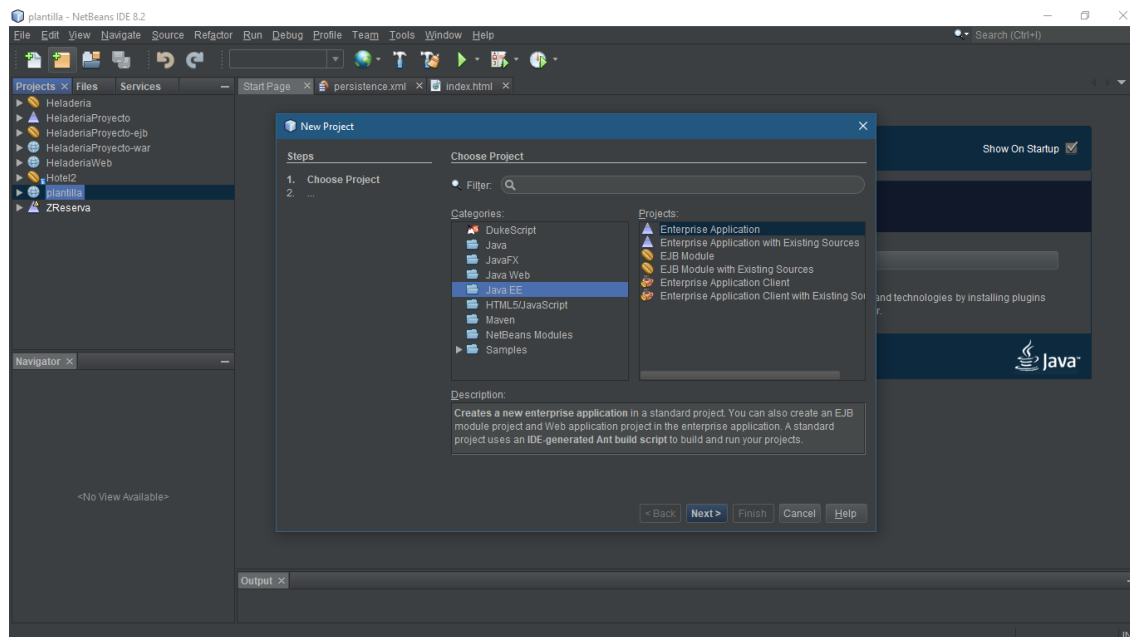




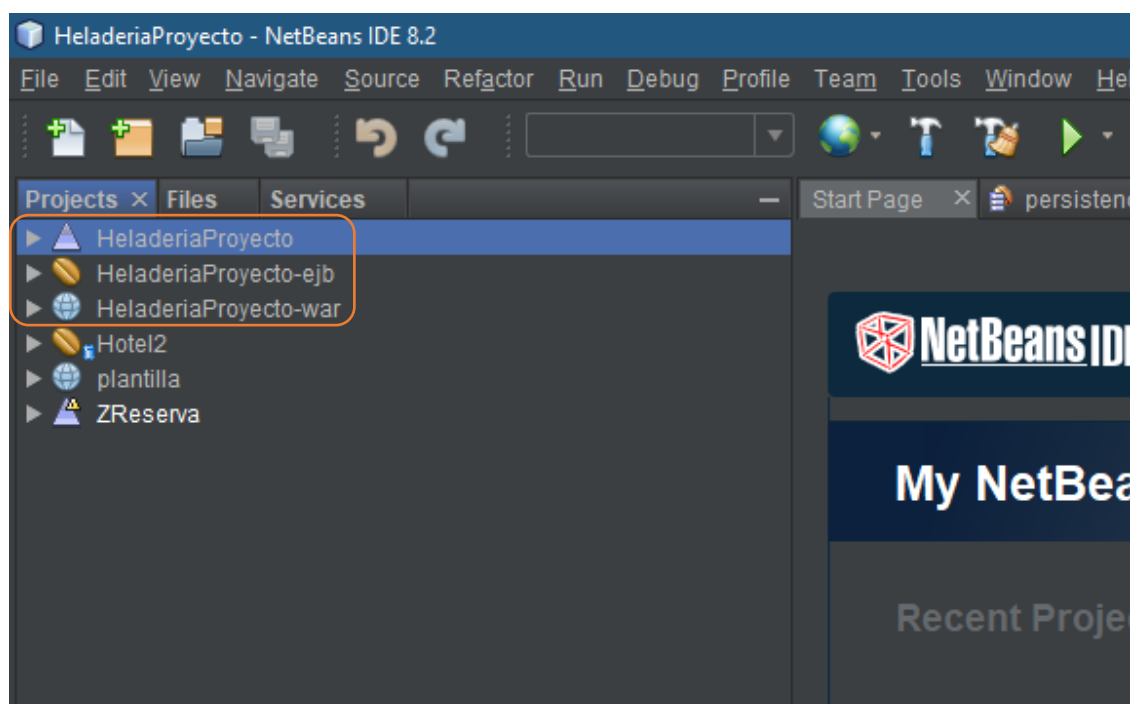
Aquí nos encontramos en las propiedades de una tabla ya creada que es la de “Cliente”. Como podemos observar cada objeto se encuentra con su respectivo tipo, como su “id” de tipo “entero” y el resto de objetos están de tipo “String”, con el fin de que los datos que el usuario ingrese solo sean de ese tipo.



Proyecto EAR con WAR y JPA incluido.

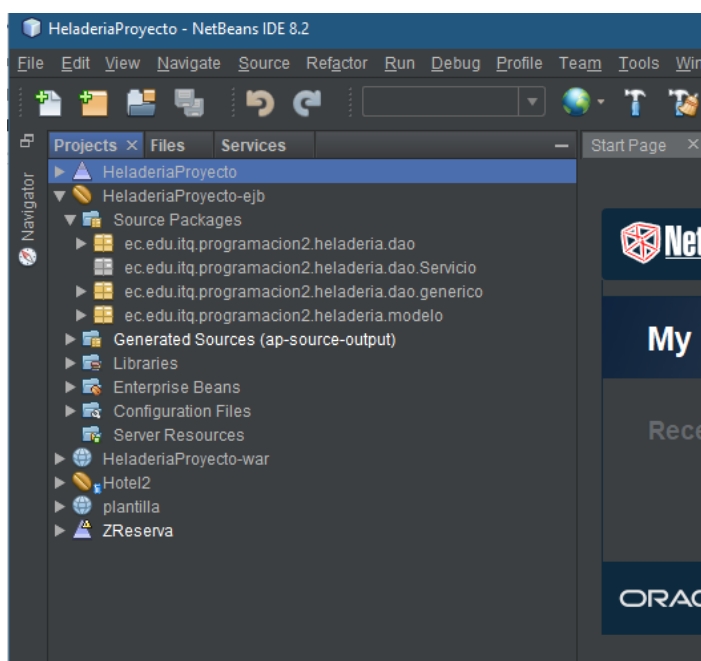


Paso 4: Utilizando nuestra herramienta de ID de confianza. Nos disponemos a crear un nuevo proyecto pero de tipo EAR. Nos dirigimos a crear un nuevo proyecto, y nos dirigimos a la carpeta de “Java EE”, nos ubicamos en “Enterprise Application” y le asignamos un nombre, en este caso le asigne como “HeladeriaProyecto”.

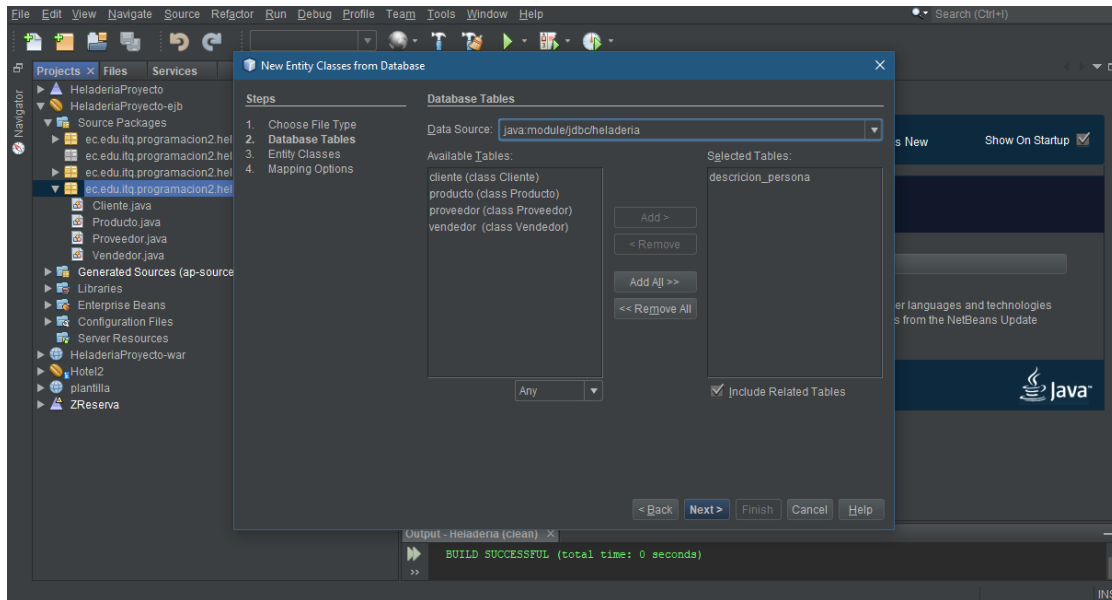


Como podemos visualizar en la parte de proyectos de nuestra herramienta ID, al haber creado un nuevo proyecto EAR, automaticamente se nos crea dos proyectos con el mismo nombre pero de tipo:”EJB” y ”WAR”.

Mapeo relacional de objetos hacia las entidades con JPA



Paso 5: Nos disponemos a crear varios paquetes de tipo “Java”. Siguiendo las reglas de desarrollo en lenguaje de programación java, nombramos a nuestro proyecto especificando el país, su tipo educación, nombre del curso y nombre del proyecto.

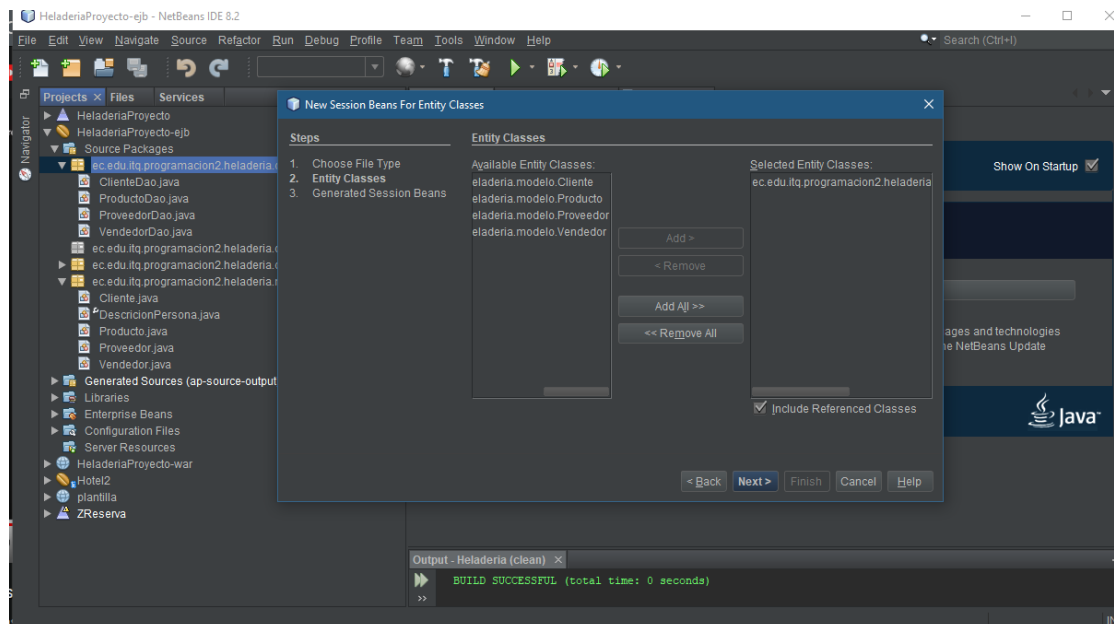


Como el proyecto ya fue creado anteriormente, por motivo para desarrollar este informe recopiló y redactó los pasos esenciales de las creaciones de los diferentes pasos del avance del proyecto.

Aquí podemos visualizar que en el paquete java ya creado anteriormente, damos click derecho en el paquete Java y ponemos “entity manage from data base” para realizar la conexión con nuestra base de datos.

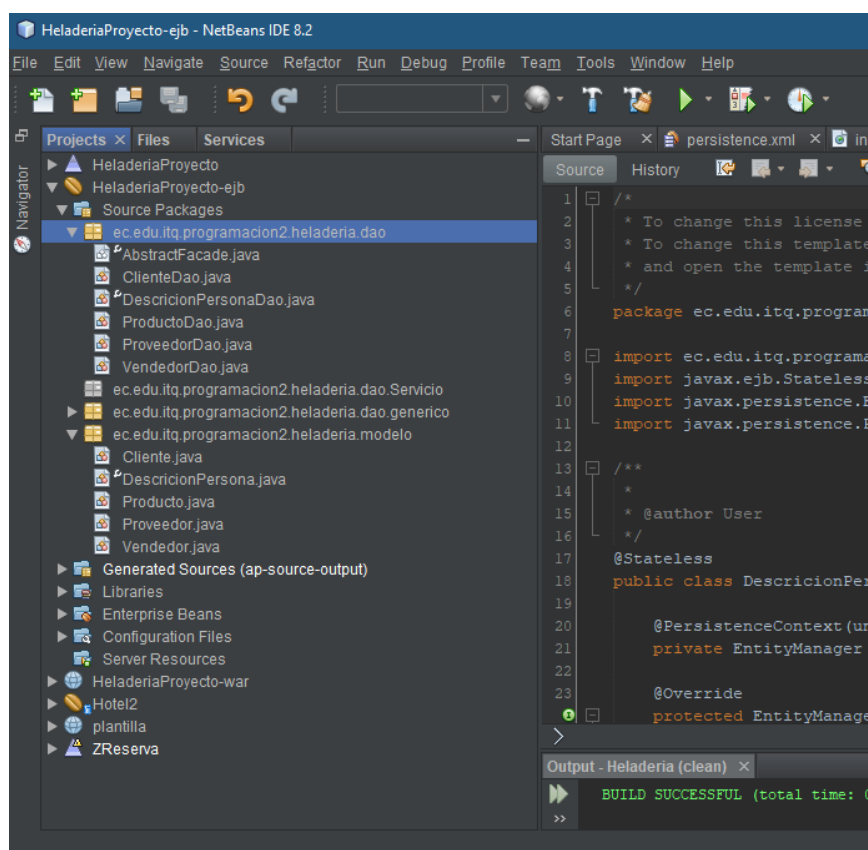
Como podemos visualizar ya está realizada la conexión solo agrego la última tabla ya creada.

Patrón de diseño DAO.



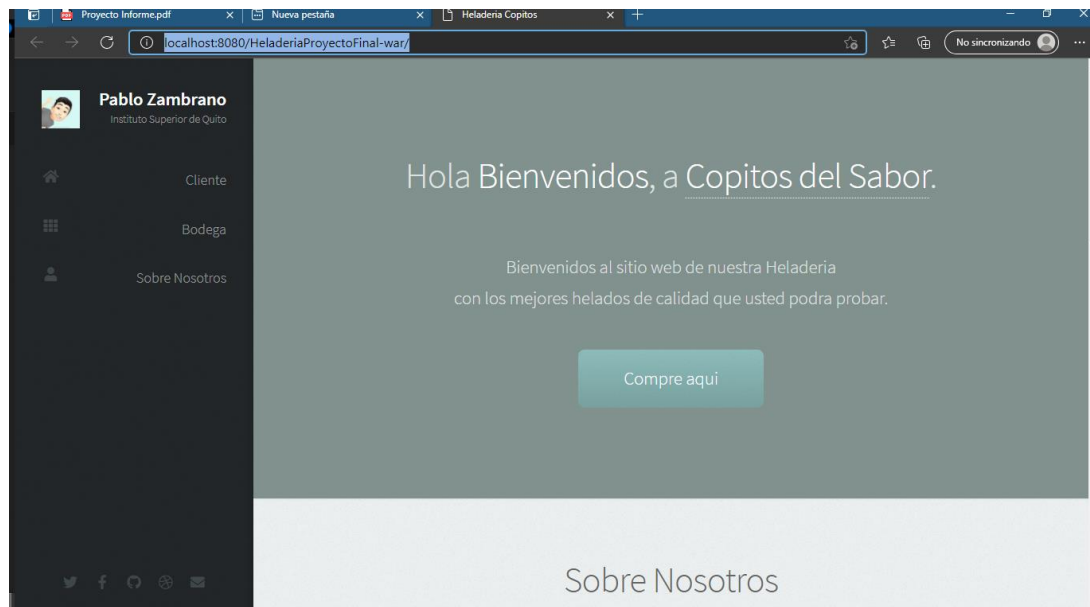
Paso 6: En este último paso de este avance del proyecto hacemos el patrón de diseño DAO.

Nos disponemos a crear otro paquete JAVA. Y dentro del paquete crear un Sesion Beans for Entity Class donde haremos nuestro patrón de diseño DAO.

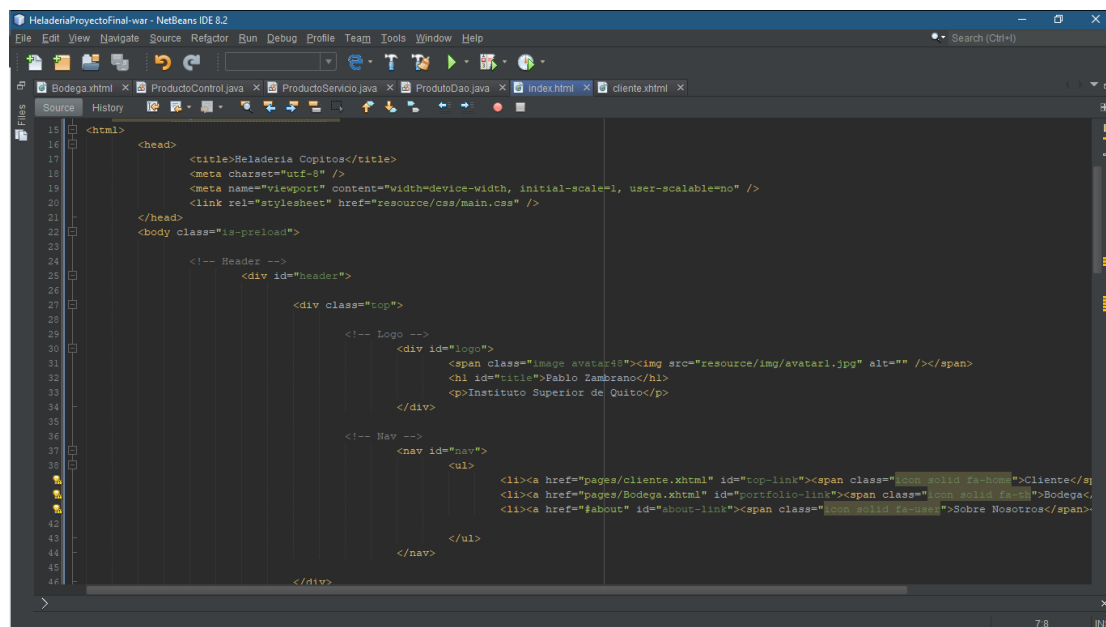


Y nos disponemos a cada clase renombrarlo de Facade a Dao después del nombre de la clase.

Página de inicio Index.html.



Su código



```

48 <div class="bottom">
49
50 <!-- Social Icons -->
51 <ul class="icons">
52 <li><a href="#" class="icon brands fa-twitter"><span class="label">Twitter</span></a></li>
53 <li><a href="#" class="icon brands fa-facebook"><span class="label">Facebook</span></a></li>
54 <li><a href="#" class="icon brands fa-github"><span class="label">Github</span></a></li>
55 <li><a href="#" class="icon brands fa-dribbble"><span class="label">Dribbble</span></a></li>
56 <li><a href="#" class="icon solid fa-envelope"><span class="label">Email</span></a></li>
57 </ul>
58 </div>
59
60 </div>
61
62 <!-- Main -->
63 <div id="main">
64
65 <!-- Intro -->
66 <section id="top" class="one dark cover">
67 <div class="container">
68
69 <header>
70 <h2 class="hi">Hola <strong>Bienvenidos</strong>, a <a href="http://html5up.net/license">C
71 <p>Bienvenidos al sitio web de nuestra Heladeria de
72 con los mejores helados de calidad que usted podra probar.</p>
73 </header>
74
75 <footer>
76 <a href="pages/cliente.xhtml" class="button scrolly">Compre aqui</a>
77 </footer>
78
79




```



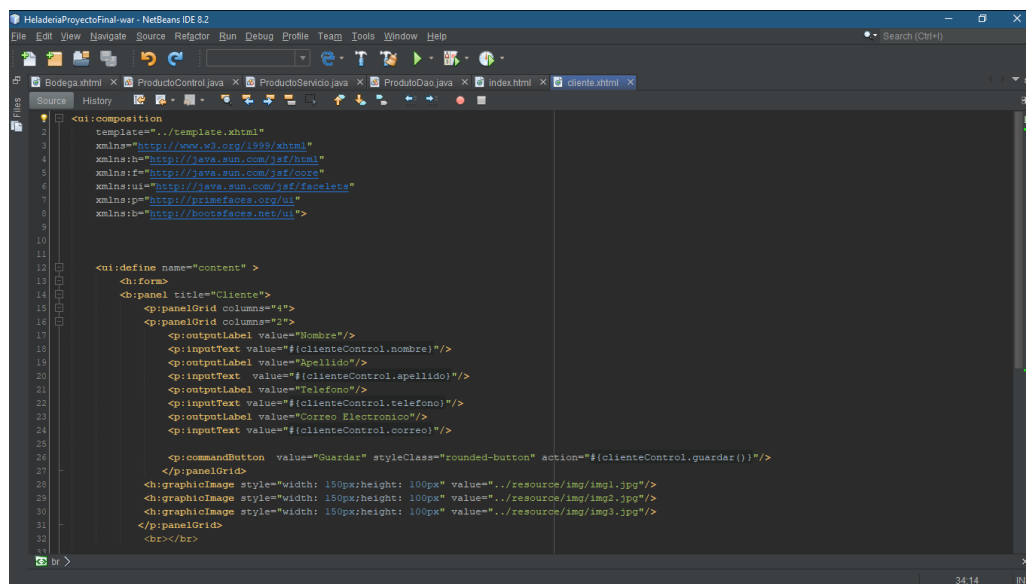
Al visualizar nuestro Index, podemos visualizar un botón de “Compre aqui”, seleccionamos y nos dirigirá a nuestra página xhtml de Cliente donde el usuario podrá ingresar sus datos para luego ser administrados en una base de datos.

Top  
Left

Cliente

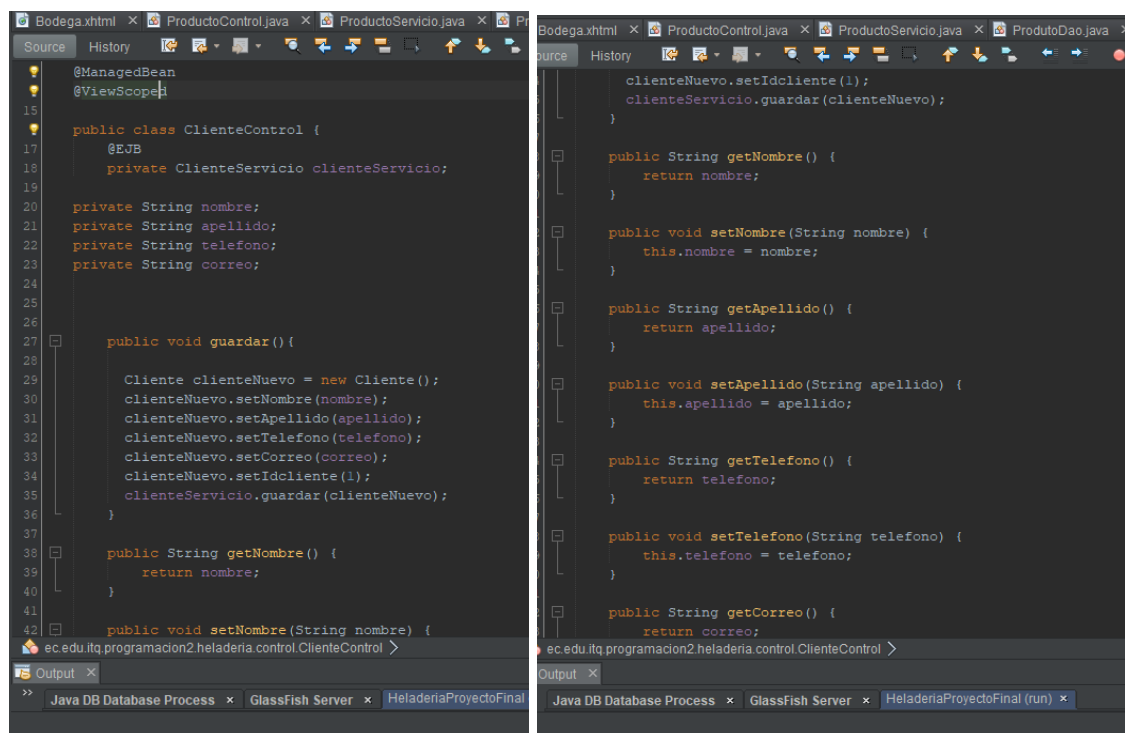
Nombre	<input type="text"/>	  
Apellido	<input type="text"/>	
Telefono	<input type="text"/>	
Correo Electronico	<input type="text"/>	
<input type="button" value="Guardar"/>		

Su código.



En los respectivos Input text damos al usuario a que ingrese sus datos personales y que tenga un valor de guardar en el método ya creado en el controlador.

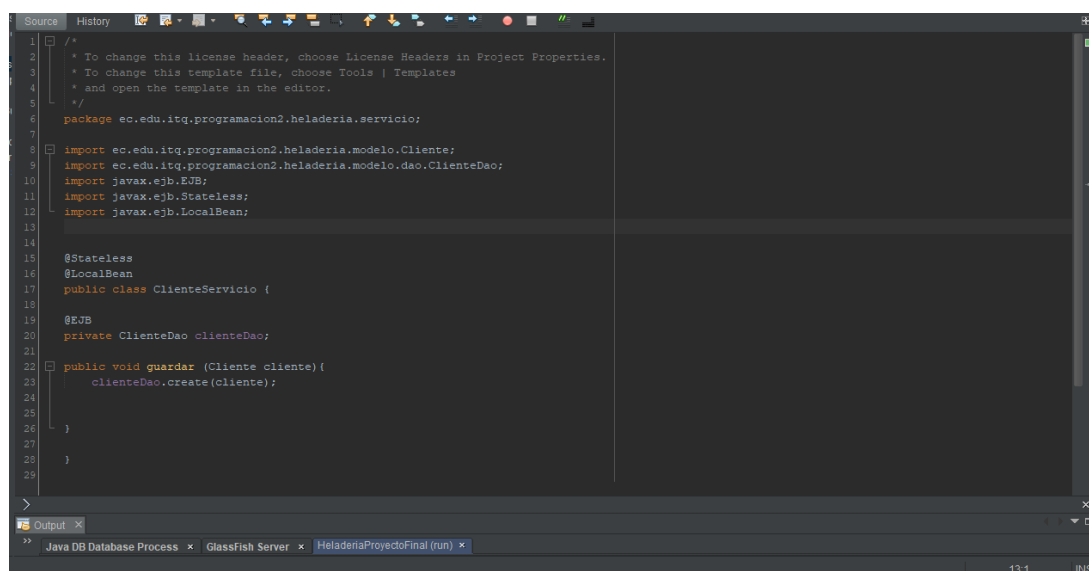
Código de ClienteControl.





Optamos por crear primero las variables en donde se irán alojando los datos del usuario, tomando los mismos nombres que fueron ya creados en las columnas de nuestra base de datos.

Creamos una clase Java Managed bean con el nombre “ClienteServicio”

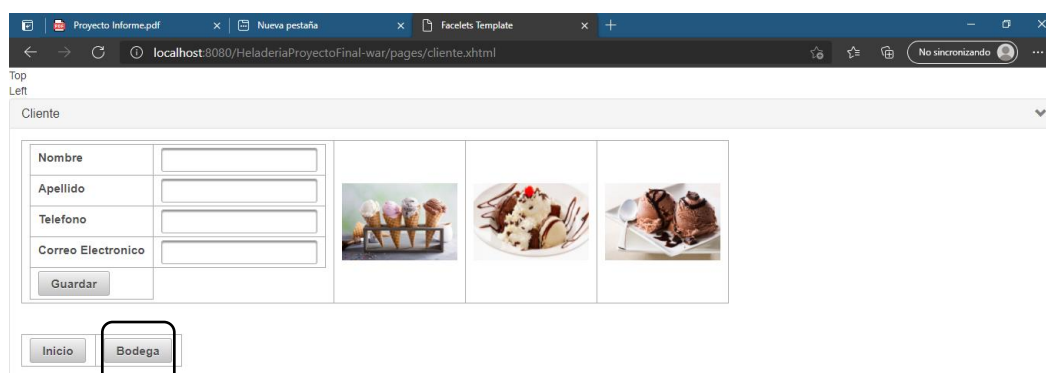


```

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 ec.edu.itq.programacion2.heladeria.servicio;
7
8  import ec.edu.itq.programacion2.heladeria.modelo.Cliente;
9  import ec.edu.itq.programacion2.heladeria.modelo.dao.ClienteDao;
10 import javax.ejb.EJB;
11 import javax.ejb.Stateless;
12 import javax.ejb.LocalBean;
13
14 @Stateless
15 @LocalBean
16 public class ClienteServicio {
17
18     @EJB
19     private ClienteDao clienteDao;
20
21     public void guardar (Cliente cliente){
22         clienteDao.create(cliente);
23     }
24 }

```

Aquí nos disponemos a crear nuestro método guardar donde lo relacionamos con nuestra clase DAO de Cliente, que nos ayudara al ingreso de datos en nuestra tabla cliente.



The screenshot shows a web browser window with the URL `localhost:8080/HeladeriaProyectoFinal-war/pages/cliente.xhtml`. The page title is "Cliente". It contains a form with the following fields:

- Nombre
- Apellido
- Telefono
- Correo Electronico

Below the form is a "Guardar" button. To the right of the form are three images of ice cream treats. At the bottom of the page, there are two buttons: "Inicio" and "Bodega". The "Bodega" button is highlighted with a red rectangle.

Al ingresar en el botón “Bodega” ingresaremos a un listado de todos los productos que nuestra empresa tiene disponible.

```

10 <ui:define name="content">
11 <h:form>
12
13     <p:panel header="Numero">
14         <p:dataTable value="#{productoControl.listaproducto}" var="pr">
15             <p:column headerText="ID" >
16                 <p:outputLabel value="#{pr.idproducto}" />
17             </p:column>
18             <p:column headerText="Nombres" >
19                 <p:outputLabel value="#{pr.nombres}" />
20             </p:column>
21             <p:column headerText="Precio" >
22                 <p:outputLabel value="#{pr.precio}" />
23             </p:column>
24             <p:column headerText="Estado" >
25                 <p:outputLabel value="#{pr.estado}" />
26             </p:column>
27             <p:column headerText="Actualizar">
28                 <p:commandButton type="button" value="Actualizar" styleClass="rounded-button" />
29             </p:column>
30         </p:dataTable>
31     </p:panel>
32
33     <p:panelGrid columns="2">
34         <p:commandButton value="Inicio" action="../index.html" />
35         <p:commandButton value="Clientes" action="cliente.xhtml" />
36     </p:panelGrid>
37 </ui:define>

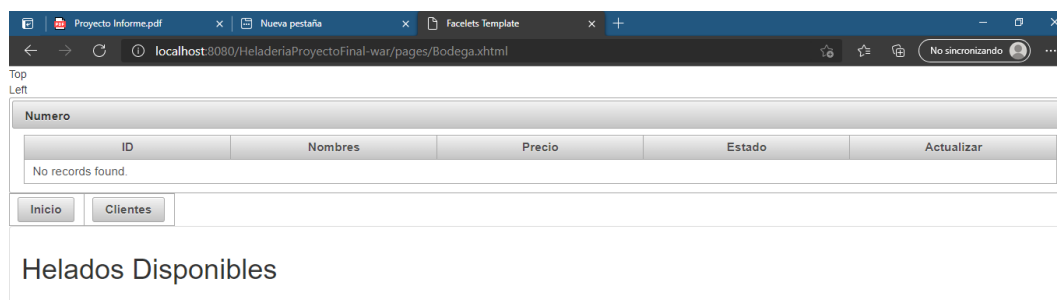
```

Output

Java DB Database Process x GlassFish Server x HeladeriaProyectoFinal (run) x

BUILD SUCCESSFUL (total time: 9 seconds)

8:37 OVR



```

6 @ViewScoped
7 @ManagedBean
8 public class ProductoControl implements Serializable {
9     @EJB
10     private ProductoServicio productoServicio;
11
12     private List<Producto> listaproducto;
13
14     public void init () {
15         buscarProducto();
16     }
17
18     public void buscarProducto () {
19         listaproducto = productoServicio.buscarProducto();
20     }
21
22     public List<Producto> getListaproducto () {
23         return listaproducto;
24     }
25 }

```

Output

Java DB Database Process x GlassFish Server x HeladeriaProyectoFinal (run) x

BUILD SUCCESSFUL (total time: 1 second)

13:37 INS

```

7
8 import ec.edu.itq.programacion2.heladeria.modelo.Producto;
9 import ec.edu.itq.programacion2.heladeria.modelo.dao.ProductoDao;
10 import java.util.List;
11 import javax.ejb.EJB;
12 import javax.ejb.Stateless;
13 import javax.ejb.LocalBean;
14
15 /**
16  *
17  * @author User
18  */
19 @Stateless
20 @LocalBean
21 public class ProductoServicio {
22
23     @EJB
24     private ProductoDao productoDao;
25
26     public List<Producto> buscarProducto() {
27
28         List<Producto> listaProducto = productoDao.listaProducto();
29         return listaProducto;
30     }
31
32 }
33

```

ec.edu.itq.programacion2.heladeria.servicio.ProductoServicio > buscarProducto >

Output

Java DB Database Process x GlassFish Server x HeladeriaProyectoFinal (run) x

BUILD SUCCESSFUL (total time: 1 second)

29.22 INS

```

10 import java.util.List;
11 import javax.ejb.Stateless;
12
13 import javax.persistence.Query;
14
15 /**
16  *
17  * @author User
18  */
19 @Stateless
20 public class ProductoDao extends GenericDao<Producto> {
21
22
23
24     public ProductoDao() {
25         super(Producto.class);
26     }
27
28     public List<Producto> listaProducto() {
29         Query query = getEntityManager().createQuery("SELECT pr FROM Producto pr");
30         List<Producto> listaProducto = query.getResultList();
31         return listaProducto;
32     }
33
34 }
35

```

Hicimos unos cambios en la clase Dao de Producto, donde creando un “query” seleccionamos la clase que está relacionada con nuestra tabla, utilizando dentro del paréntesis lenguaje sql.

## CONCLUSIONES:

En este curso de Programación 2, tuvimos muchas ganancias personales como el de obtener experiencias en el trabajar y conocer como es el entorno de trabajo alrededor del desarrollo de software, siguiendo a cabalidad las peticiones de ingeniero encargado del curso.

Se realiza un sitio web alojado en el servidor local de aplicaciones con programación orientada a objetos “Java”, utilizamos el archivo EAR (Enterprise Archive) que es un formato utilizado en la arquitectura JEE para desplegar de manera coherente y simultanea varios módulos en un servidor de aplicaciones, en este caso hacemos uso de módulos tipo EJB y WEB y un servidor de aplicaciones GlassFish.

## RECOMENDACIONES:

Recomendamos el uso de lenguaje de programación Java para cualquier tipo de desarrollo de software que requiere de intenso análisis informático. Gracias a cuyo proyecto se puede desarrollar múltiples aplicaciones con funcionalidades infinitas gracias a su característica de ser orientada a objetos donde se le puede llevar una organización mucho más ordenada.