El Paseo Te Lo Lleva

Blanco Valentin Nicolas
Paladino Gabriel Agustín

El Paseo

- Feria con objetivo de contacto entre productor y cliente
- Promueven la Economía popular social y solidaria
- Acercamiento: Planteo al principio de la materia

el paseo
Economía Social y Solidaria
UNLP

Situación

- Anterioridad
  - Página web sin resultados (pandemia)
- Actualidad
  - Venta por whatsapp
  - Planillas a mano

Problemas

- Visibilidad (productores contacto consumidor)
- Limitación Geográfica
- Generar interés
- Ventas online
- Acceso a información importante

Propuesta

- Plataforma de Comercio Electrónico:
  - Automatización de pedidos
  - Automatización de planillas
  - Fácil uso para todo público (administradores y clientes)

Desarrollo de la Aplicacion

- Fue subdividido en 5 etapas
  - Análisis, diseño y bocetado
  - Definición de objetos del modelo
  - Desarrollo de capa de persistencia
  - Desarrollo de capa de servicios
  - Desarrollo de capa de presentación

1er Etapa: Análisis

- En esta etapa se realizaron las siguientes tareas:
  - Historias de Usuario
    - Requisito descrito brevemente utilizando lenguaje común
    - Verificables, Pequeñas e Independientes
  - Maquetado
    - Desarrollado a partir de historias de usuario
    - Boceto de la página web a la que se aspira al finalizar el proyecto
    - Figma para modelado

Ejemplo 1: Historia de usuario junto a maquetado

Perfil administrador y perfil consumidor/visitante

Iniciar sesión

Como usuario registrado quiero ingresar al sitio de compras.

Descripción

El usuario aprieta el botón “Ingresar/Registrarse” y se lo redirigirá a una vista de inicio de sesión donde deberá ingresar su mail y su contraseña. Si los datos ingresados son correctos se lo redirigirá a la página principal y se reemplazará el botón “Ingresar/Registrarse” por su nombre.

Ejemplo 2: Historia de usuario junto a maquetado

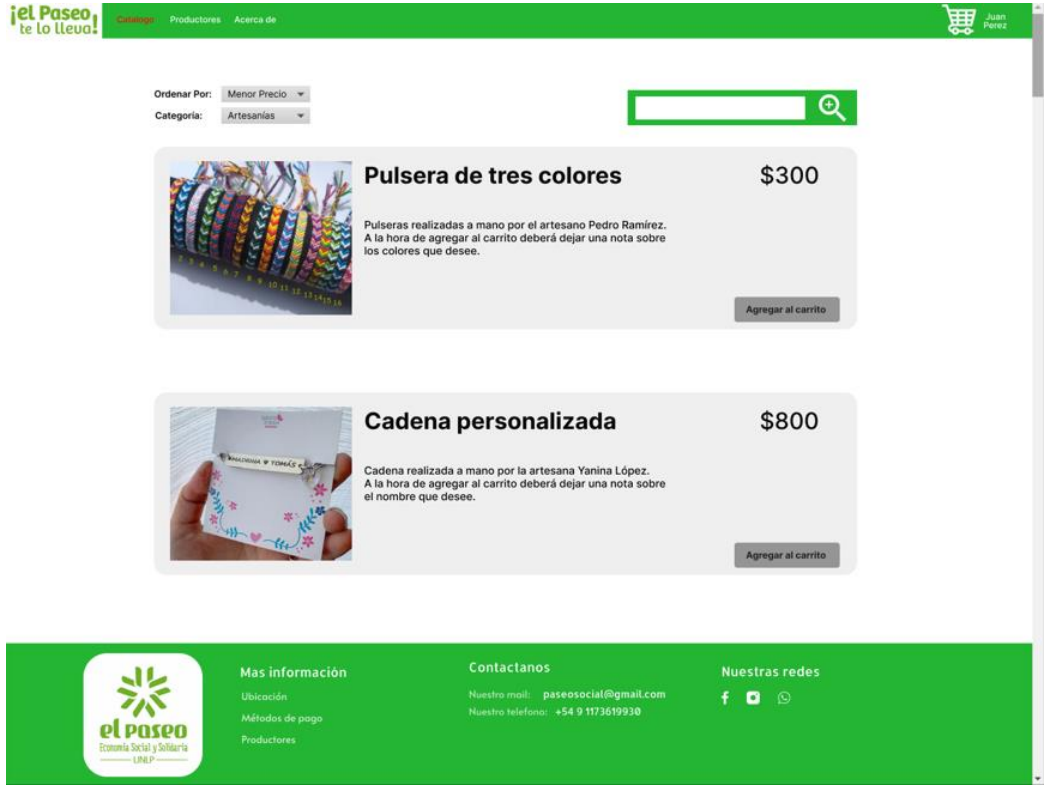
Perfil consumidor/visitante

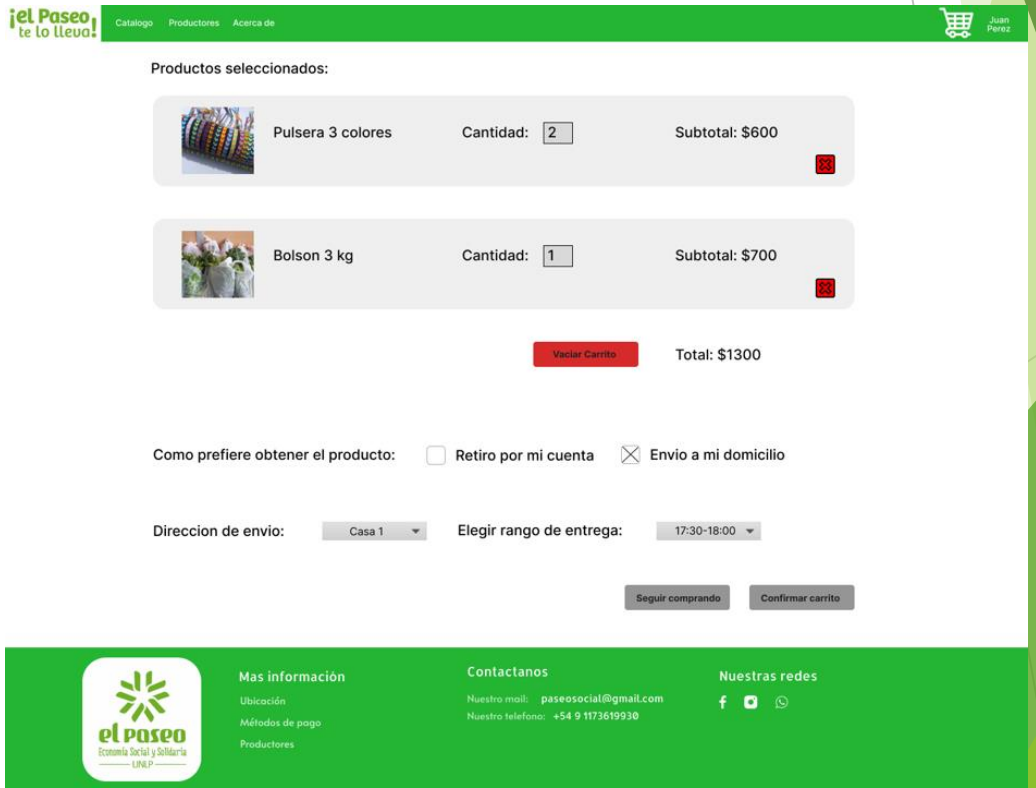
Agregar al carrito

Como cliente, quiero poder agregar productos al carrito de compras para poder comprar varios productos al mismo tiempo.

Descripción

Sobre los productos aparece el botón "Agregar al carrito", haciendo clic en el botón se adjunta el producto al carrito y se actualiza el monto de la compra.

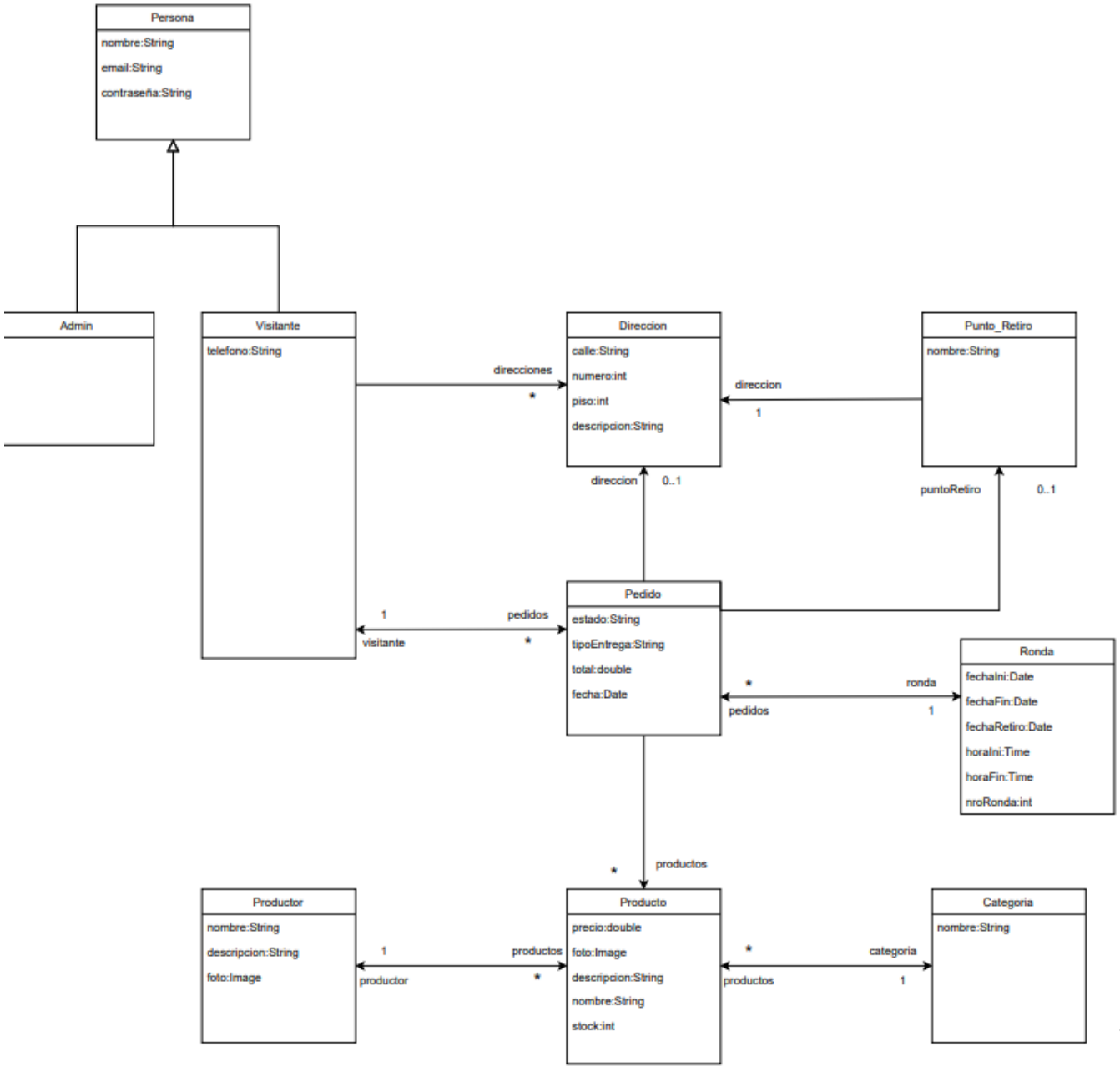




2da Etapa: Definición de Objetos del modelo

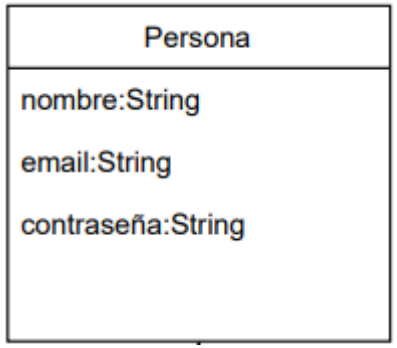
- Se realizaron las siguientes tareas:
  - Creación de diagrama UML
    - Identifica objetos
    - Define relación y atributos de objetos
    - Establece jerarquías
    - Abstracción de objetos junto atributos
  - Creación de clases JAVA
    - Basado en el UML

Diagrama UML resultante



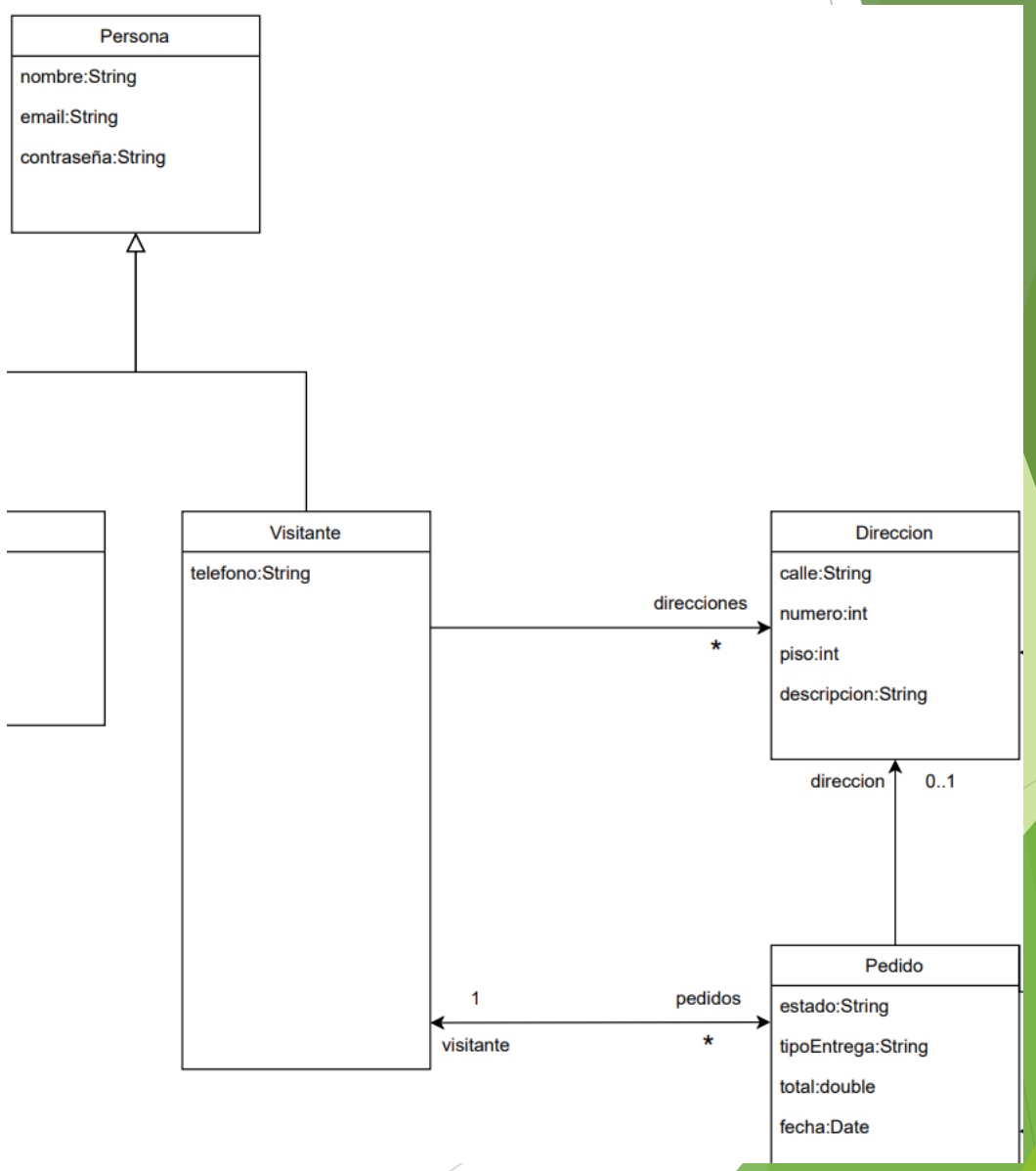
Ejemplo clase Java junto al uml

```java
public class Persona {
    private String nombre;
    private String email;
    private String contraseña;

    public Persona() {
        super();
    }
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getContraseña() {
        return contraseña;
    }
    public void setContraseña(String contraseña) {
        this.contraseña = contraseña;
    }

}
```



Ejemplo clase Java junto al uml

```java
public class Visitante extends Persona {
    private String Telefono;
    private List<Direccion> direcciones;
    private List<Pedido> pedidos;

    public Visitante() {
        super();
    }

    public String getTelefono() {
        return Telefono;
    }

    public void setTelefono(String telefono) {
        Telefono = telefono;
    }

    public List<Direccion> getDirecciones() {
        return direcciones;
    }

    public void setDirecciones(List<Direccion> direcciones) {
        this.direcciones = direcciones;
    }

    public List<Pedido> getPedidos() {
        return pedidos;
    }

    public void setPedidos(List<Pedido> pedidos) {
        this.pedidos = pedidos;
    }
```



3er Etapa: Desarrollo de capa de persistencia

- Las tareas de esta etapa consistieron en:
  - Resolver que clases serán persistidas
    - Se utilizo patrón DAO para las persistencias
  - Definir las relaciones entre las clases mediante anotaciones permitidas a partir de JPA
  - Uso de JPQL para hacer consultas a la base de datos
- Dichas tareas se hicieron para realizar un conjunto de casos de prueba y la alta, baja y modificación de cada una de las clases

Ejemplo Clase con patrón DAO

```java
package grupo2.dao;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

import grupo2.model.Producto;

public class ProductoDAO implements IProductoDAO {

	public List<Producto> getProductos() {
		EntityManager em = EntityManagerSingleton.getEntityManager();
	javax.persistence.Query query = em.createQuery("SELECT p FROM Producto p");
	List<Producto> l=query.getResultList();
	EntityManagerSingleton.CloseEntityManager();
	return l;
	}

	public Producto getProducto(Long id) {
		EntityManager em = EntityManagerSingleton.getEntityManager();
	javax.persistence.Query q = em.createQuery("SELECT p FROM Producto p where(p.id= :id)");
		q.setParameter("id", id);
		Producto d=(Producto)q.getSingleResult();
		EntityManagerSingleton.CloseEntityManager();
	return d;
	}

	public void agregarProducto(Producto p) {
		EntityManager em = EntityManagerSingleton.getEntityManager();
	EntityTransaction tx = em.getTransaction();
	tx.begin();
	em.persist(p);
	tx.commit();
	EntityManagerSingleton.CloseEntityManager();

	}
```

```java
public void actualizarProducto(Producto p){
		EntityManager em = EntityManagerSingleton.getEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();
em.merge(p);

tx.commit();
EntityManagerSingleton.CloseEntityManager();

}

public void eliminarProducto(Producto p) {
		EntityManager em = EntityManagerSingleton.getEntityManager();
		EntityTransaction tx = em.getTransaction();
		tx.begin();
    em.remove(em.merge(p)); // se obtiene una entidad gestionada antes de eliminarla
    tx.commit();
    EntityManagerSingleton.CloseEntityManager();

}
```

Ejemplo clases java utilizando anotaciones JPA

```java
package grupo2.model;
import java.util.*;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
@Entity
public class Producto {
        @Id @GeneratedValue
    @Column(name="PRODUCTO_ID")
    private Long id;
        private double precio;
        private String foto;
        private String descripcion;
        private String nombre;
        private int stock;
        @ManyToOne
        @JoinColumn(name="PRODUCTOR_ID")
        private Productor productor;
        @ManyToOne
        @JoinColumn(name="CATEGORIA_ID")
        private Categoria categoria;
```

```java
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToMany;
@Entity
public class Categoria {
        @Id @GeneratedValue
    @Column(name="CATEGORIA_ID")
    private Long id;
        private String nombre;
        @OneToMany(mappedBy="categoria")
        private List<Producto> productos;
```

```java
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;

import java.awt.Image;
@Entity
public class Productor {
        @Id @GeneratedValue
    @Column(name="PRODUCTOR_ID")
    private Long id;
        private String nombre;
        private String descripcion;
        private String foto;
        @OneToMany(mappedBy="productor")
        private List<Producto> productos;
```

Conexión y configuración de la base de datos

```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
        version="2.1">
        <persistence-unit name="miUP">
                <provider>org.hibernate.ejb.HibernatePersistence</provider>
                <class>grupo2.model.Admin</class>
                <class>grupo2.model.Categoria</class>
                <class>grupo2.model.Direccion</class>
                <class>grupo2.model.Pedido</class>
                <class>grupo2.model.Persona</class>
                <class>grupo2.model.Producto</class>
                <class>grupo2.model.Productor</class>
                <class>grupo2.model.Punto_Retiro</class>
                <class>grupo2.model.Ronda</class>
                <class>grupo2.model.Visitante</class>
                <properties>
<!--        Conexion a base de datos propia
                        <property name="hibernate.hbm2ddl.auto" value="update" />
            <property name="hibernate.connection.driver_class" value="com.mysql.jdbc.Driver" />
            <property name="hibernate.connection.password" value="" />
            <property name="hibernate.connection.url" value="jdbc:mysql://localhost:3306/testpersistencia"/>
            <property name="hibernate.connection.username" value="root" />
            <property name="hibernate.dialect" value="org.hibernate.dialect.MySQL5InnoDBDialect" />-->
                <property name="hibernate.hbm2ddl.auto" value="update" />
                        <property name="hibernate.connection.driver_class" value="com.mysql.jdbc.Driver" />
                        <property name="hibernate.connection.password" value="jyaa2023_pwd3" />
                        <property name="hibernate.connection.url" value="jdbc:mysql://mysql.java.linti.unlp.edu.ar/jyaa2023_bd3?useSSL=FALSE" />
                        <property name="hibernate.connection.username" value="jyaa2023_usr3" />
                        <property name="hibernate.dialect" value="org.hibernate.dialect.MySQL5InnoDBDialect" />
                </properties>
        </persistence-unit>
</persistence>
```

4ta Etapa: Desarrollo de capa de servicios

- Servicio: manera en la que aplicaciones pueden comunicarse entre sí a través de internet.
- Esta etapa consistió en tareas como:
  - Empezar con el desarrollo de servicios para los que se utilizaron las siguientes herramientas
    - Jackson (Permite conversión automática de JSON a clases Java y viceversa)
    - HK2 (Framework que permite la Inversión de Control y la Inyección de Dependencia)
    - Swagger (Permite documentar api REST de manera sencilla)
    - JAX-RS (Framework de java que implementa RESTFul)
      - Dentro de las implementaciones sobre JAX-RS se utilizó Jersey perteneciente a ORACLE

Ejemplo de implementación de servicios de productos

```java
@Path("productos")
@Tags(value= {@Tag(name="Productos",description="Metodos de Productos")})
public class ProductoResource {

	@Context
	UriInfo uriInfo;

	@Context
	Request request;

	private String mensaje;
	@Inject
	private IProductoDAO pdao;

	@GET
	@Path("/todos")
	@Produces(MediaType.APPLICATION_JSON)
	@Operation(summary = "Obtener productos", description = "se listan todos los productos de la base de datos")
	public List<Producto> getAll() {
		return pdao.getProductos();
	}

 @PUT
 @Produces(MediaType.APPLICATION_JSON)
 @Consumes(MediaType.APPLICATION_JSON)
 @Operation(summary = "Editar producto", description = "modifica un producto de la base de datos ")
 @ApiResponses(value= {
		@ApiResponse(responseCode= "200", description = "Producto editado correctamente"),
		@ApiResponse(responseCode= "404", description = "Fallo al encontrar el producto a editar"),
 })
 public Response editar(Producto p) {
	Producto aux = pdao.getProducto(p.getId());
	if ( (aux != null) && (aux.isBorrado()==false)) {
		pdao.actualizarProducto(p);
		return Response.ok().entity(p).build();
	} else {
		return Response.status(Response.Status.NOT_FOUND).entity("[]").build();
	}
 }
```

Ejemplo swagger y jackson

Productores Metodos de Productor

GET `/productores/{id}` Obtener producto por id

DELETE `/productores/{id}` Eliminar productor por id

PUT `/productores` Editar productor

modifica un productor de la base de datos

Parameters Try it out

No parameters

Request body application/json

Example Value | Schema

```
{
  "id": 0,
  "borrado": true,
  "nombre": "string",
  "descripcion": "string",
  "foto": "string"
}
```

Responses

Code	Description	Links
200	Productor editado correctamente	*No links*
404	Fallo al encontrar el productor a editar	*No links*

5ta Etapa: Desarrollo de capa de presentación e integración con servicios

- Capa de Presentación: Interfaz gráfica que permite a los usuarios navegar, consumir contenido y realizar acciones en el sitio.
- Capa realizada utilizando angular junto a HTML, CSS y Bootstrap
- Para esta etapa se realizaron las siguientes tareas:
  - Se crearon las vistas de la página web siguiendo el maquetado
  - Se consumieron los servicios de la 4ta etapa
  - Se verificaron las historias de usuario

Ejemplo integración de capa de servicios

```
import { Injectable, inject } from "@angular/core";
import { Router } from '@angular/router';
import { HttpClient, HttpErrorResponse, HttpParams } from '@angular/common/http';
import { Observable, throwError } from 'rxjs';
import { of } from 'rxjs';
import { HttpHeaders } from '@angular/common/http';
import { catchError, map } from 'rxjs/operators';
import { Producto } from "./producto";

const httpOptions = {
    headers: new HttpHeaders({
    'Content-Type': 'application/json',
    })
};
@Injectable({
    providedIn: 'root'
})
export class ProductoService {
    public productos: Producto[] = [];

  agregarProducto(producto: Producto) {
    this.productos.push(producto);
  }
  obtenerProductos(): Producto[] {
    return this.productos;
  }

  constructor(private http: HttpClient, private router: Router) { }

  public getproductos() : Observable<Producto[]> {
    return this.http.get('/rest/productos/todos')
    .pipe(map((data: any) =>
      data.map((producto: any) =>
        new Producto(producto.id,producto.borrado ,producto.precio, producto.foto, producto.descripcion, producto.nombre, producto.stock, producto
    )
  )
  )

  }
```

Organización del grupo

- Trabajo Realizado en conjunto mediante herramientas tales como:
  - Discord
  - Gitlab
  - Drive
  - Whatsapp

Reflexión

- Acercamiento a experiencia real (Entrevista, documentación, muestra al cliente)
- Experiencia web (Ingeniería en computación no posee este desarrollo)
- Implementación y uso de herramientas y tecnologías como
  - Backend con Java
  - Frontend con Angular
  - Documentacion con Swagger
  - Figma para Maquetado