



Daniel Díaz-Delgado Menéndez

TRABAJO FIN DE FP DAW: PROYECTO

Instituto Tecnológico Edix

INDICE

- Introducción.
- Módulos Formativos.
- Objetivos del proyecto.
- Herramientas y lenguajes utilizados.
- Fases del Proyecto.
- Conclusiones y mejoras.
- Usabilidad.

INTRODUCCION

Uno de los motivos por los que creo que todo el mundo debería realizar alguna formación sobre internet es debido a la grande demanda de gente con conocimientos de internet de cómo podemos acercarnos a los demás por medio de rrss, de negocio en línea o de vender y de darse a conocer a través de internet, uno de los negocios que más han cambiado es de la venta online, vemos a empresas como amazon, zalando, incluso zara, que han tenido que evolucionar para llegar a todo el mundo y poder adaptarse al nuevo mercado de ventas.

El trabajo de Fin de Desarrollo de Aplicaciones Web que he seleccionado está basado en un sector en auge que permite el comercio internacional y nacional de una manera mucho más visible.

Con el desarrollo de internet, se han ido desarrollando nuevas modalidades de negocio, permitiendo que cualquier persona pueda vender sus productos o servicios o comprar desde cualquier parte del mundo, esto se ha visto reflejado en las tiendas online y permitiendo que productos que antes eran más difíciles de conseguir sean más accesibles para cualquier ciudadano con conexión a la red.

Viendo que es un sector en auge y la cantidad de productos que podemos ofrecer en España, me he decantado por desarrollar una tienda con productos de la tierra, típicos de este país, como es el jamón serrano, el queso o el lomo, que gracias al género que tenemos se ha convertido en una marca a nivel internacional que nos hace sobresalir sobre el resto de países del mundo.

Lo que permite esta aplicación es que cualquier ganadero, apicultor o cualquier persona de una zona rural, poner a disposición del mundo sus productos de una manera sencilla, da igual donde viva y así generar más conocimiento sobre sus zonas y sus costumbres, ganando público y tanto para su negocio como para su tierra.

He seleccionado tres productos, aunque en un futuro se podrían ampliar a mas, como el vino o como la miel, esta elección se basa en la gran demanda y oferta que hay, dando así la posibilidad de que puedan adquirir género con diferentes precios y calidades.

MODULOS FORMATIVOS

Los módulos usados en este trabajo en relación con este trabajo son:

Empresas:

Esta web está enfocada en un nicho de mercado que quiere productos ibéricos de este país y una de las maneras de hacer este acercamiento posible es a través de una pagina o aplicación web, las personas en la actualidad no se acercan a Mercadona o alCampo a por sus productos de calidad culinaria que sobresalga al resto, compran productos healthy sin que exista mucha manipulación en los alimentos y que sean asequibles desde el sofá de su casa.

Este modelo de negocio permite parametrizar las ventas y lanzar campañas de marketing de una manera mas personalizada, diferenciadno por franjas de edad, sexo, lugar de residencia, etc.

El gasto por persona en comercio electrónico esta subiendo cada año como podemos ver en el siguiente grafico

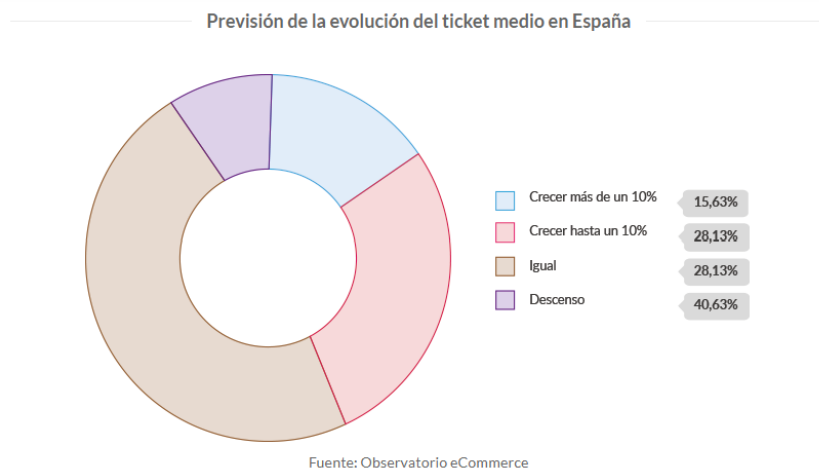


Grafico de <https://ticnegocios.camaravalencia.com/servicios/tendencias/tendencias-del-comercio-electronico-hacia-donde-vamos/>

Como se puede ver, es un mercado en alza que presiona a negocios de toda la vida a que se actualicen y mezclen su modelo de negocio de toda la vida con la tecnología, como ha tenido que realzar la marca Osborne:

MADRID, 15 (EUROPA PRESS)

Cinco Jotas, la firma de jamón 100% ibérico de bellota del Grupo Osborne, impulsa su 'ecommerce' y su expansión internacional con el lanzamiento de su tienda 'online' en Estados Unidos.

Así como hemos comentado, los negocios están cambiando y el coste de montar un ecommerce es menor al de montar una tienda física, quitándonos ese precio del local y de tener a una persona en la tienda sin hacer nada (algunas horas) .

El coste aproximado de montar un ecommerce es el hosting y el dominio y sobre todo la publicidad que quieras darle, no necesitas locales ni un desembolso inicial elevado para poder crearte tu negocio y ver si funciona tu idea, esto es lo que nos facilita la tecnología, el empezar algo con un coste ínfimo.

También necesitas una gestoría para llevarte los impuestos de la web y dinero para realizar un pedido mínimos para poder hacer frente a los primeros pedidos.

Aquí podemos ver unos datos aproximados del coste de montar una tienda online :

Tienda online con Dropshipping

Dominio: 10 euros al año

Hosting: 50-80 euros al año

Plantilla responsive: 50 euros

Programación: 300-500 euros

Creación de contenidos: 200-300 euros

Herramientas SEO: 100 euros

Publicidad online: 200 euros

Inversión para trabajar en una herramienta de email marketing: 30 euros al mes (desde 0 € / mes si trabajas con Mailrelay)

Creación de una estrategia de mailing: 300 euros

► **Inversión total para empezar: 1.240 euros**

Tienda online con productos propios

Dominio: 10 euros al año

Hosting: 50-80 euros al año

Plantilla responsive: 50 euros

Programación: 300-500 euros

Creación de contenidos: 200-300 euros

Compra de productos: 500 euros

Publicidad online: 200 euros

Herramientas SEO: 100 euros

Inversión para trabajar en una herramienta de email marketing: 30 euros al mes ([desde 0 € / mes si trabajas con Mailrelay](#))

Creación de una estrategia de mailing: 300 euros

► **Inversión total para empezar: 1.740 euros**

Tienda online con productos de afiliados

Dominio: 10 euros al año

Hosting: 50-80 euros al año

Plantilla responsive: 50 euros

Programación: 300-500 euros

Creación de contenidos: 200-300 euros

Herramientas SEO: 100 euros

Publicidad online: 200 euros

Inversión para trabajar en una herramienta de email marketing: 30 euros al mes ([desde 0 € / mes si trabajas con Mailrelay](#))

Creación de una estrategia de mailing: 300 euros

Datos de <https://blog.mailrelay.com/es/2018/11/02/costes-lanzar-ecommerce>

Son tres modelos de negocio de una tienda online en los que se modifica el origen de los productos.

Ahora veremos un análisis DAFO del



Como debilidades

- Desconfianza en la venta online que aun se mantiene en el tiempo.
- Cultura de salir a comprar la comida.
- Necesidad de logística.
- Conocimientos de informática.

Como fortalezas

- Ahorro en infraestructuras físicas.
- Relación mas personalizada con el cliente.
- Igualdad entre empresas, da igual que los vena mercadona o una tienda del barrio al poder venderlo de la misma manera que las grandes marcas.

Como amenazas:

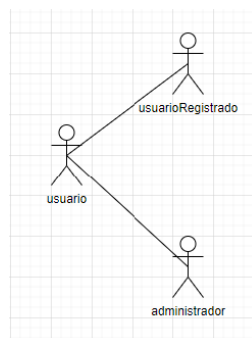
- Dependemos del uso de internet para que nos conozcan.
- Capacidad alta de encontrar competencia al ser 'fácil' la creación de una tienda.
- Ingresos volubles sin un fijo mensual.

Como oportunidades

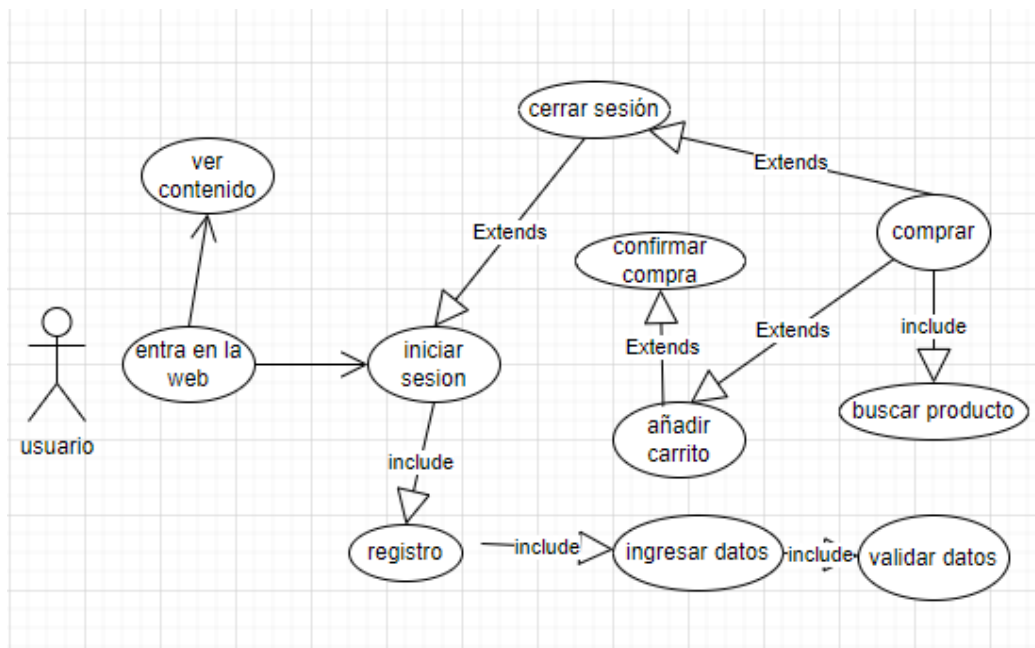
- Ingreso de nuevos mercados a través de una web.
- Captación de nuevos clientes.
- Emisión de presupuestos vía online.
- Marketing online.

Diseño de interfaces web.

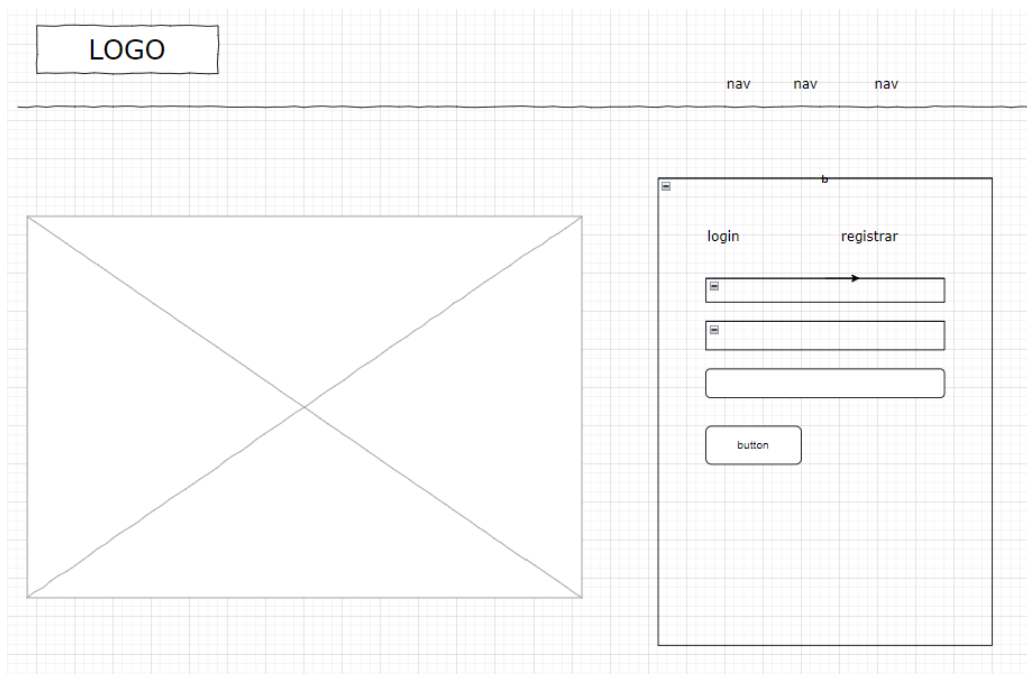
Vamos a crear un diagrama de uso de la web, los actores son:



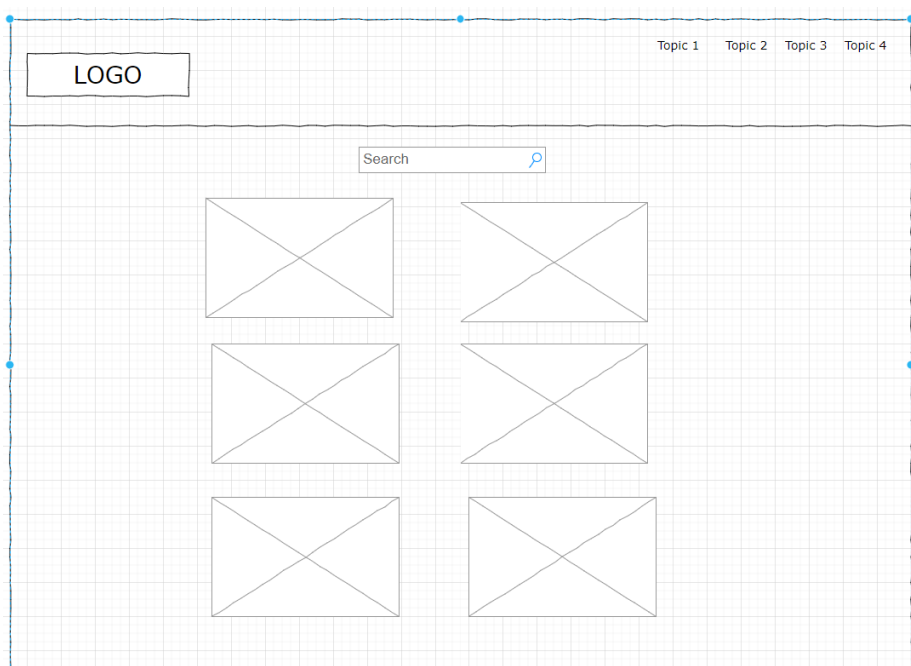
Es uso de web quedaría del siguiente modo:



La página de inicio es:



La página de productos, donde se verán los diferentes productos y el usuario podrá bucar los mismos desde la barra de buscar



La siguiente página que tenemos es la de ‘about’, donde se muestra el origen de la compañía y una descripción detallada del origen de cada producto a fin de dar a conocer la calidad de los mismos.

Base de datos

A fin de tener un stock de la empresa, guardar usuarios registrados, etc etc se ha desarrollado una base de datos en MySQL .

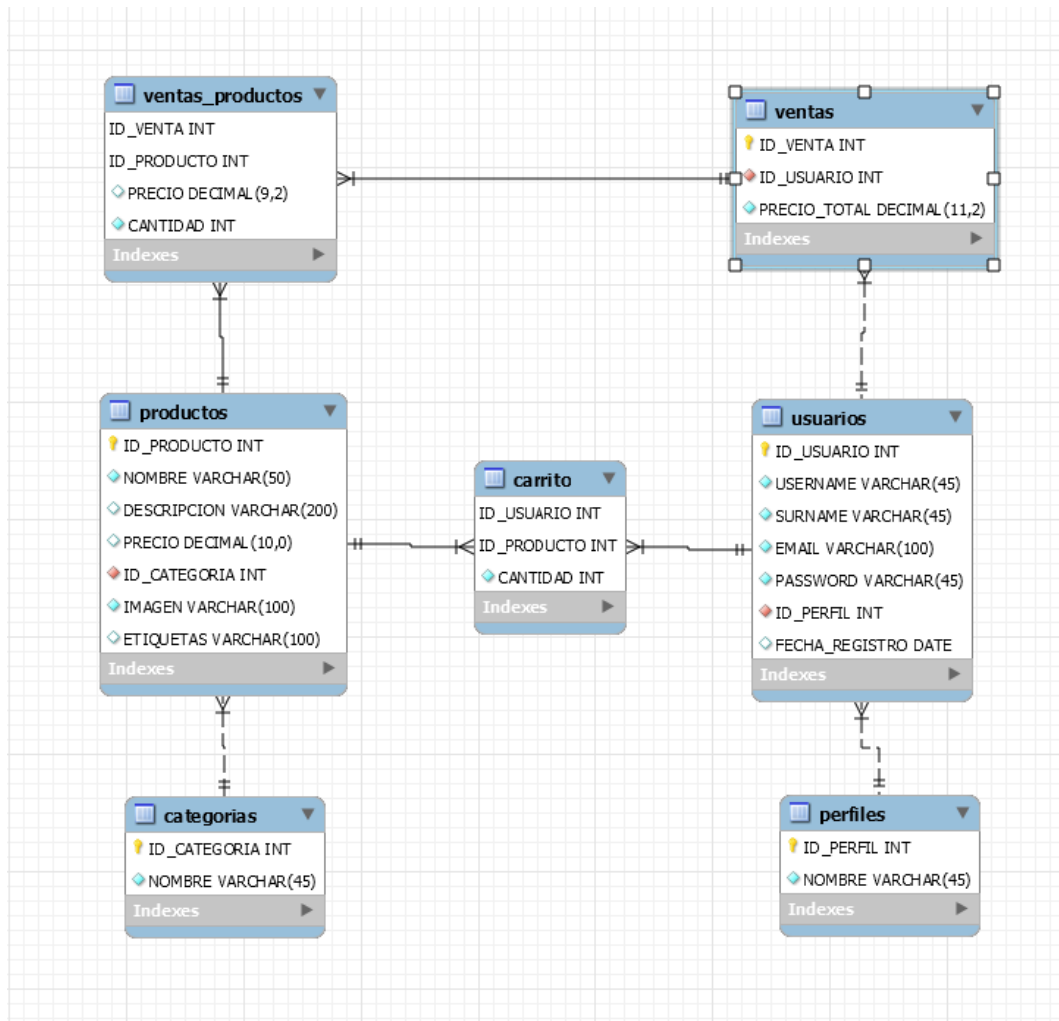
En total salen 7 tablas de esta tienda online:

- Usuario.
- Carrito.
- Perfiles.
- Productos.
- Categorías.
- Ventas.
- Ventas-productos.

El usuario puede loguearse o registrarse, hay dos perfiles de los mismos, uno administrador y otro normal, desde el usuario normal puedes seleccionar productos y pasarlos a la tabla carrito, una vez confirmada la venta, la tabla carrito de vacía a fin de

trasladar la información a la tabla venta y venta-producto, el usuario Administrador puede acceder a esa información y obtener las ventas de cada usuario, el gasto total y los productos más vendidos.

En la tabla productos se almacenan las imágenes de cada producto, su precio, su nombre y descripción y en la tabla categorías se ven las diferentes categorías de cada producto, esto nos permite ampliar el catálogo en un futuro a nuevos productos.



BBDD.sql

Adjuntamos el script de la BBDD.

En la relación nos salen 2 tablas renacidas, la de carrito y la de ventas_productos, esto es para evitar la relación de muchos a muchos que se daría entre usuarios y productos y entre ventas y productos.

A la hora de hacer búsquedas en la BBDD, se realizan a través de JPA, estas consultas llevan por debajo la consulta JDBC.

```
sql = "SELECT p FROM Producto p WHERE p.nombre =:cod1 AND p.descripcion =:cod2"; //consulta sql
```

Así, y de una manera más sencilla y simplificada, obtenemos, modificamos y actualizamos datos en la base de datos desde el modelo.dao.

Programación

Hemos realizado la aplicación utilizando java como lenguaje de programación, alguna de las características de este lenguaje es:

1. Es simple: es mas imple que el lenguaje C++ , se puede ejecutar en una maquina pequeña , el tamaño admitido y la biblioteca de clases es de aproximadamente 40kb.
2. Orientado a objetos: trabaja con objetos e interfaces, soporta las características de:
 - a. Encapsulación.
 - b. Herencia.
 - c. Polimorfismo.
3. Distribuido: Existen librerías y herramientas para que los programas puedan ser distribuidos, para que este en varias máquinas interactuando.
4. Robusto: realiza comprobaciones de problemas tato en tiempo de ejecución como en tiempo de compilación.
5. Arquitectura neutral: genera formatos de archivo que se pueden ejecutar en muchos procesadores
6. Lenguaje seguro: el código pasa muchos test antes de interpretarse en una máquina, se pasa a través de un verificador de bytecodes que comprueba el formato de los fragmentos de código y aplica un probador de teoremas para evitar acceso a objetos de código ilegal,
7. Portable: el tamaño y el algoritmo del tipo están estrictamente regulados por lo que son portátiles.

Entorno de desarrollo

La aplicación estar subida en GitHub a fin de ser publica y poder ser accesible desde cualquier ordenador (si se permite) , de esta manera s se quieren realizar cambios solo habría que subir la nueva versión del proyecto a GitHub y así poder tener en la nube el proyecto sin depender del borrado del mismo en el ordenador .

Por medio de comandos, podemos subir o bajarnos un proyecto desde la web o aportar valor a otros proyectos colaborando con los mismos.

Lenguaje de marcas



En esta web se ha usado HTML, CSS a fin de dar estilo y estructura a nuestras webs, por medio del HTML (HyperText Markup Language)se ha dado un esquema a cada página del proyecto, este permite a los navegadores que interpreten la web, a fin de estructurar las diferentes partes de la misma.

HTML se compone de etiquetas, cuando se crear una, se crea un nodo o un objeto del DOM, esto permite que otros lenguajes accedan a ellos asignando diferentes métodos o atributos, esto lo hacen lenguajes como JavaScript.



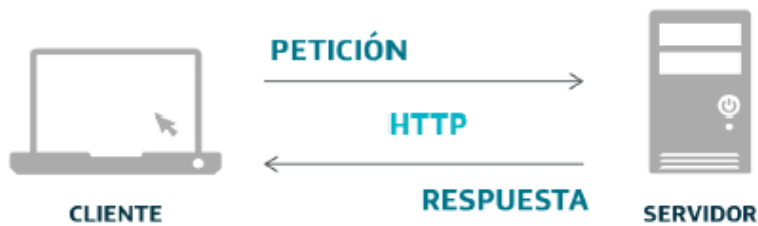
A fin de definir el formato de los elementos, usaremos CSS y aplicaremos estilos a los objetos de nuestra pagina web.

Los estilos son estáticos, no se pueden modificar en tiempo de carga o ejecución, si varios elementos tienen atributos iguales, se definen para cada uno de ellos.

Mediante alguna etiqueta como: hover o :Focus , se puede generar algo de dinamismo al documento.

Desarrollo web en entorno cliente.

Mediante la presente asignatura, aprendimos a generar cambios en el DOM, generan una interacción entre el cliente y la web .



El lenguaje aprendido es JavaScript, lenguaje de programación de scripting (interpretado), normalmente embebido en un documento HTML y definido y orientado a objetos, tiene características dinámicas, se utiliza principalmente del lado del cliente mediante el intérprete del navegador web.

Los lenguajes de script no se usan solamente en las páginas web, sino que están integrados en muchas aplicaciones de uso cotidiano como en google docs.

Se ha usado en la página, a fin de mostrar opciones en el <nav> dependiendo de si el usuario es o no es administrador, o a fin de enseñar los productos que están en la base de datos, esto se hace alterando el DOM de la página web.

O mediante una llamada del cliente, escribiendo en la barra de búsqueda de productos, poder visualizar los mismos de una manera más sencilla.



También se ha usado en el formulario de registro o de loguearse, se muestra uno u otro dependiente de si el usuario clicla en un botón u otro

Login Register cambia, clicamos en uno u otro

Introduce un Nombre

Nombre

Introduce un Apellido

Apellido

Correo electronico

Enter email

Introduce una Contraseña

Contraseña

Registrese

Correo electronico

Contraseña

Enter email

Password

Entrar

Se integra en el documento HTML igual que el CSS de la web, mediante un script:

Así el código queda más limpio y nos permite utilizarlo en otras partes o en otros HTMLS

```
<script type="text/javascript" src="acciones_1.js"></script>
```

Desarrollo web en entorno servidor

La aplicación web tiene una estructura de tres capas, donde el cliente a través del navegador, se comunica con la capa intermedia a través del protocolo HTTP, la aplicación de ejecuta en el servido de la capa intermedia, la respuesta la recibe el cliente en HTML .

Para realizar la aplicación usamos java/java ee, especificación de Oracle que tiene librerías y un conjunto de tecnologías específicas de Oracle.

- Servlets: estos atienden las peticiones que llegan desde la capa cliente y generan una página de respuesta.
- Java Server Pages: estos son archivos de texto con extensión jsp, permite que se combinen bloques de HTML con código Java, se parece a un servlet en funciones

pero con la especificación de que esta más encaminado a generar una respuesta por el cliente que en atender peticiones.

- Enterprise JavaBeans: implementan la lógica del negocio de una aplicación web.

La aplicación java EE necesita la existencia de un software conocido como servidor de aplicaciones para ejecutarse.

En el caso de esta aplicación hemos usado Tomcat, que se desarrolla por el grupo apache, Apache Tomcat es un contenedor Java Servlet, o contenedor web, que proporciona la funcionalidad extendida para interactuar con Java Servlets, al tiempo que implementa varias especificaciones técnicas de la plataforma Java: JavaServer Pages (JSP), Java Expression Language (Java EL) y WebSocket.

También hay funciones similares: Jetty, Resin, Websphere, weblogic, JBoss, Glassfish, GonAS, etc. Su cuota de mercado es la siguiente: Puede ver que Tomcat es el servidor de aplicaciones Java WEB más popular:

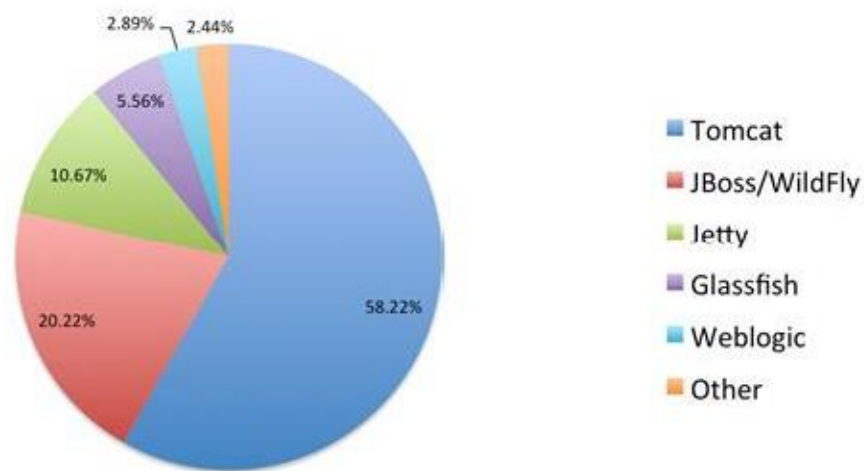


Imagen de <https://programmerclick.com/article/64651871358/>

Tomcat está escrito en lenguaje Java y debe ejecutarse en la máquina virtual Java, por lo que generalmente es necesario instalar el JDK primero para proporcionar un entorno de ejecución.

En los JPS hemos utilizado expresiones de lenguaje (EL), debido a que estas expresiones son más simples e intuitivas que las instrucciones javas, la sintaxis de uso será:

`${objeto.propiedad}`

También hemos incorporado al proyecto la librería JSTL (JavaServer Pages Standard Tag Library), esta se puede añadir o directamente o a través de MAVEN:

[%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %](http://java.sun.com/jsp/jstl/core)

JSTL es una librería de tags estándar que encapsula, en forma de tags, muchas funcionalidades comunes en aplicaciones JSP, de forma que, en lugar que tener que recurrir a varias librerías de tags de distintos distribuidores, sólo necesitaremos tener presente esta librería que, además, por el hecho de ser estándar, funciona de la misma forma en cualquier parte, y los contenedores pueden reconocerla y optimizar sus implementaciones.

Esta forma parte de Java Enterprise, tiene cinco grupos de acciones:

- Core: se utiliza para funciones de propósito general (manejo de expresiones, sentencias de control de flujo, etc).
- Formato
- Sql: sirve para acceder y manipular bases de datos relacionales.
- Xml : se emplea para procesamiento de ficheros XML.
- Function.

En la siguientes tablas se muestran las áreas cubiertas por JSTL (cada una con una librería):

Librería EL:

AREA	URI	PREFIJO
Core	http://java.sun.com/jsp/ jstl/core	c
XML	http://java.sun.com/jsp/ jstl/xml	x
Internacionalización (I18N)	http://java.sun.com/jsp/ jstl/fmt	fmt
SQL	http://java.sun.com/jsp/ jstl/sql	sql

Librería RT:

AREA	URI	PREFIJO
Core	http://java.sun.com/jsp/ jstl/core_rt	c_rt
XML	http://java.sun.com/jsp/ jstl/xml_rt	x_rt
Internacionalización (I18N)	http://java.sun.com/jsp/ jstl/fmt_rt	fmt_rt
SQL	http://java.sun.com/jsp/ jstl/sql_rt	sql_rt

En nuestro proyecto hemos usado la librería Core, a fin de recorrer la lista del carrito he imprimir por pantalla los valores que queríamos al cliente o al administrador, mediante un foreach:

```
<c:forEach [var="variable"] items="conjunto"
[varStatus="variableEstado"] [begin="comienzo"]
[end="final"] [step="incremento"]>
    codigo
</c:forEach>
```

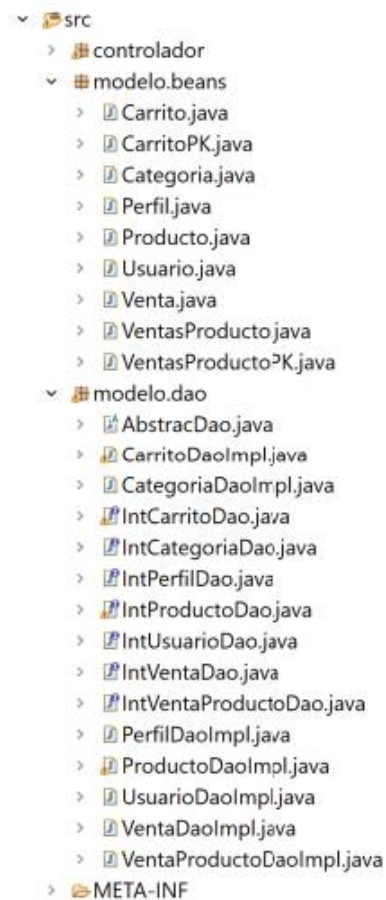

En nuestra aplicación hemos usado un patrón de Modelo Vista Controlador, el uso de patrones en el diseño tiene algunas ventajas:

1. Rapidez de desarrollo.
2. Optimización de código.
3. Menos errores en el código.
4. Aplicaciones escalables.

Mediante el modelo vista controlador, se separan las tareas de presentación y captura de datos de usuario, de la manipulación de los mismos, para interactuar con los dos elementos, utilizamos un tercero, el controlador que coordina todo el proceso.

El modelo se encarga del acceso, modificación, o eliminación de la información, contiene la lógica del negocio de la aplicación, en este aspecto, se ha dividido la presente aplicación en :

- Modelo.beans
- Modelo.dao.



La vista se ha realizado en el Web/Content, en el cual añadimos css, JavaScript, las imágenes y las paginas jsp que sean las que muestren al usuario la información o le soliciten la misma.

Y también disponemos del controlador de la aplicación, el cual responde a eventos o acciones del usuario, invocando al modelo cuando se hace alguna solicitud sobre la información, hace de intermediario entre la vista y el modelo.

Un resumen visual de este patrón puede ser el siguiente:



OBJETIVOS DEL PROYECTO

El objetivo del proyecto es mezclar parte de mis conocimientos adquiridos a lo largo de mis años de estudiante, al haber realizado un master de comercio internacional, empezaron las inquietudes sobre cómo se mueve ese sector y las posibilidades que tenían, viendo casos de éxito como ALIBABA o AMAZON grandes empresas, que vieron la posibilidad de mezclar el comercio de toda la vida con las nuevas tecnologías, superando así las barreras fronterizas e internacionales de los países y dando un servicio de más calidad, llegando a todas las partes del mundo y recibiendo los mismo desde la comodidad de tu casa.

Esta inquietud por internacionalizar mis conocimientos y las posibilidades casi infinitas a tener conocimientos de programación, me llevo a la realización de la presente formación a fin de dirigir mi futuro profesional fuera de las fronteras de nuestro país, adquiriendo

unos conocimientos y una forma de pensar que me permitiesen estar actualizado en el mercado laboral pudiendo crear valor desde cualquier parte del mundo y monetizar la misma.

Del estudio o admiración de las empresas antes mencionadas, opte por hacer el presente proyecto a fin de usarlo en un futuro con alguna empresa o persona que necesite la realización de una web de ventas, este proyecto da la posibilidad de aplicarlo a cualquier tienda online, solo habría que modificar pequeñas cosas como las fotos o el diseño de la misma, pero contendría la base de cualquier negocio en el que el usuario necesite un servicio o producto, y da la posibilidad a cualquier persona que quiera obtener valor económico de los productos que genera de publicitarlos en internet a un coste asumible para todos o casi todos los bolsillos, sin tener que alquilar ningún sitio físico y sin tener que moverse de casa, solo necesitando una conexión a internet para llevar/gestionar su negocio.

También contiene un resumen de los conocimientos adquiridos en cada módulo estudiado en esta formación profesional de desarrollo de aplicaciones web, generando así una base para futuros proyectos e inquietudes y para seguir con la formación en este mundo IT, lleno de posibilidades ilimitadas y en constante desarrollo.

HERRAMIENTAS Y LENGUAJES UTILIZADOS

En este proyecto se ha utilizado Visual Studio Code (un editor de código construido sobre el framework Electron) a fin de realizar toda la parte del HTML, CSS y JavaScript.

Se ha utilizado a fin de realizar el UML y pintar las clases de la base de datos draio:

<https://app.diagrams.net/>

Se ha utilizado MySQL Workbench, es una herramienta de diseño de base de datos que integra desarrollo de software, administración de base de datos, se usa sobre todo para bases de datos relacionales.

Y sobre todo se ha utilizado Eclipse, un entorno de desarrollo integrado a fin de evitar el trabajo por consola del programador.

Hemos usado JPA (Java Persistence API) como framework, proporciona un mecanismo para gestionar la persistencia y la correlación relacional de objetos y funciona desde las especificaciones EJB 3.0.

JPA se basa en el modelo de programación Java que se aplica a los entornos Java EE (Java Enterprise Edition) pero JPA puede funcionar en un entorno Java SE para probar las funciones de las aplicaciones.

JPA crea un estándar para la importante tarea de la correlación relacional de objetos mediante la utilización de anotaciones o XML para correlacionar objetos con una o más tablas de una base de datos. Para simplificar aún más el modelo de programación de persistencia:

- La API EntityManager puede actualizar, recuperar, eliminar o aplicar la persistencia de objetos de una base de datos contiene todos los métodos necesarios para acceder a los datos de la CRUD
- La API EntityManager y los metadatos de correlación relacional de objetos manejan la mayor parte de las operaciones de base de datos sin que sea necesario escribir código JDBC o SQL para mantener la persistencia.
- JPA proporciona un lenguaje de consulta, que amplía el lenguaje de consulta EJB independiente, conocido también como JPQL, el cual puede utilizar para recuperar objetos sin grabar consultas SQL específicas en la base de datos con la que está trabajando.

JPA utiliza los siguientes elementos:

Unidad de persistencia

Define un modelo relacional de objeto que correlaciona las clases Java (entidades + estructuras de soporte) con una base de datos relacional. EntityManagerFactory utiliza estos datos para crear un contexto de persistencia al que se puede acceder mediante EntityManager.

EntityManagerFactory

Se utiliza para crear un EntityManager para las interacciones de base de datos. Los contenedores del servidor de aplicaciones normalmente proporcionan esta función, pero es necesario EntityManagerFactory si se utiliza la persistencia gestionada por las

aplicaciones JPA. Una instancia de EntityManagerFactory representa un contexto de persistencia.

Contexto de persistencia

Define el conjunto de las instancias activas que la aplicación está manipulando actualmente. Puede crear el contexto de persistencia manualmente o mediante inyección.

EntityManager

Gestor de recursos que mantiene la colección activa de objetos de entidad que está utilizando la aplicación. EntityManager maneja la interacción y metadatos de bases de datos para las correlaciones relacionales de objetos. Una instancia de EntityManager representa un contexto de persistencia. Una aplicación en un contenedor puede obtener el EntityManager mediante inyección en la aplicación o buscándolo en el espacio de nombres del componente Java. Si la aplicación gestiona su persistencia, el EntityManager se obtiene desde EntityManagerFactory.

Objetos de entidad

Clase Java simple que representa una fila en una tabla de base de datos con su formato más sencillo. Los objetos de entidades pueden ser clases concretas o clases abstractas. Mantienen estados mediante la utilización de propiedades o campos.

Una vez obtenido EntityManager podemos utilizar métodos para acceder a la capa de persistencia.

Persist => void persist(Object entidad, inserta en el motor del registro y añade a la BBDD información obtenida del del usuario.

Merge => Actualiza la base de datos, T merge(Object entidad).

Remove => elimina la entidad de la base de datos, void remove(Object entidad).

Find => realiza una busueda de una entidad a partir de una clave primaria, T find(Class<T> tipo, Object clave).

Para programar hemos usado Java que ya se ha comentado en puntos anteriores, lenguaje EL, y JSTL.

El proyecto de subirá a GitHub : <https://github.com/daniddm>.

FASES DEL PROYECTO

La primera fase del proyecto fue el diseño de las páginas webs, su estructura, definir donde va ir cada cosa, cada nav, cada foto, el formulario, el pie de página que estilo va a tener, que las paginas no estén muy sobrecargadas, que tengan un diseño limpio, para eso se diseñaron los wireframe que se ven en la página ocho y nueve del proyecto. Viendo como están las tendencias del mercado y los diferentes modelos de posicionar los productos o clasificar los mismos.

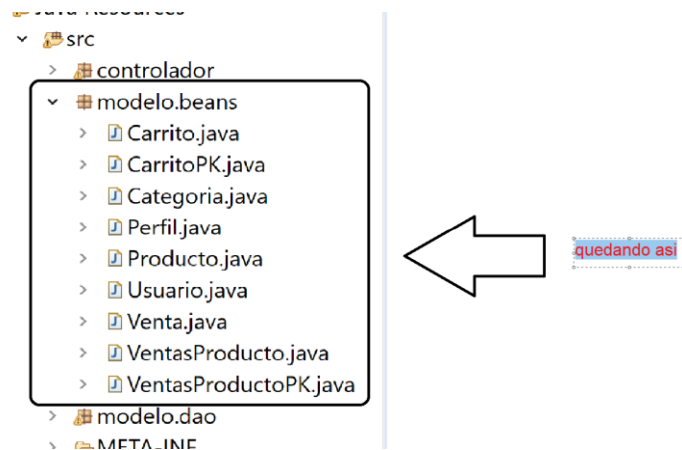
También el formulario que es limpio, viendo de un primer vistazo a fin de que el usuario se registre o haga login sin tener que bajar al fondo de la página.

En Visual Studio Code se realizó todo lo que tenía que ver con la visualización de la web, todo lo referente a HTML, CSS y JavaScript, todos esto aplicando los conocimientos de las asignaturas de lenguaje de marcas y de entorno de cliente.

La segunda fase del proyecto fue diseñar la base de datos, para ellos usamos el UML a fin de ver la usabilidad que se iba a mostrar en la página web, de ellos se realizó una BBDD que luego fue ejecutada en MySQL, dando lugar a 7 tablas.



En la tercera fase de este proyecto de genero el Dynamic Web Project en eclipse, generando el proyecto en jpa y realizando la conexión con la BBDD mediante el driver, metiendo el jar del mysql y después importamos las clases de la base de datos y generamos las asociaciones que hay entre las mismas en el modelo.bean.



Se generan dos clases de más debido a la relación de muchos a muchos y que en esas tablas hay una columna más que hace que se cree dos clases más como la de CarritoPK y la de VentasProductoPK. Con @Embeddable la PK y @Entity la que no termina en PK, poniendo en esta ultima un @EmbeddedId id

```

10 *
11 /**
12  * @Entity
13  * @Table(name="ventas_productos")
14  * @NamedQuery(name="VentasProducto.findAll", query="SELECT v FROM VentasProducto v")
15  * public class VentasProducto implements Serializable {
16  *     private static final long serialVersionUID = 1L;
17  *
18  *     @EmbeddedId
19  *     private VentasProductoPK id;
20  *
21  *     private int cantidad;
22  *
23  *     private BigDecimal precio;
24  *
25  *     //uni-directional many-to-one association to Producto
26  *     @ManyToOne
27  *     @JoinColumn(name="ID_PRODUCTO")
28  *     private Producto producto;
29  *
30  *     //uni-directional many-to-one association to Venta
31  *     @ManyToOne
32  *     @JoinColumn(name="ID_VENTA")
33  *     private Venta venta;
34  *
35  *     public VentasProducto() {
36  *     }

```

```

8 *
9 /**
10 * @Embeddable
11 * public class VentasProductoPK implements Serializable {
12 *     //default serial version id, required for serializable classes.
13 *     private static final long serialVersionUID = 1L;
14 *
15 *     @Column(name="ID_VENTA", insertable=false, updatable=false)
16 *     private int idVenta;
17 *
18 *     @Column(name="ID_PRODUCTO", insertable=false, updatable=false)
19 *     private int idProducto;
20 * }


```

Generamos los getters y setters y los constructores de las clases y realizamos la conexión a la bbdd a fin de modificar la misma, introducir datos nuevos o borrar los mismos.

A la clase que realiza la conexión la llamo AbstractDao.java, teniendo está en su interior lo siguiente:

```
1 package modelo.dao;
2
3 import javax.persistence.EntityManager;
4
5 public abstract class AbstractDao {
6     protected EntityManagerFactory emf; //de la librería persistence
7     protected EntityManager em; //el EntityManager que llevará los métodos del p
8     protected EntityManager tx; //hereda también del java persistence para la
9     protected Query query; //para las queries
10    protected String sql; //para las jpql el string, que llamaremos sql
11
12    public AbstractDao() {
13        //llamamos a la clase persistence y su método, al que le pasaremos un Str
14        emf = Persistence.createEntityManagerFactory("Ecommerce_SanTierno"); //le
15        em = emf.createEntityManager(); //crea el entityManager y lo devuelve
16        tx = em.getTransaction(); //Devuelve la transacción.
17    }
18 }
```

En el mismo paquete se generará la clase Int que será una interface que después desde las clases DaoImpl se implementaran los métodos y el extends que extendiendo la funcionalidad de la clase padre la hija.

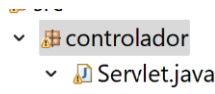
 CarritoDaoImpl.java  IntCarritoDao.java

Estas clases estarán en la carpeta modelo.dao:

```

└─ modelo.dao
  ├─ AbstractDao.java
  ├─ CarritoDaoImpl.java
  ├─ CategoriaDaoImpl.java
  ├─ IntCarritoDao.java
  ├─ IntCategoriaDao.java
  ├─ IntPerfilDao.java
  ├─ IntProductoDao.java
  ├─ IntUsuarioDao.java
  ├─ IntVentaDao.java
  ├─ IntVentaProductoDao.
  ├─ PerfilDaoImpl.java
  ├─ ProductoDaoImpl.java
  ├─ UsuarioDaoImpl.java
  ├─ VentaDaoImpl.java
  ├─ VentaProductoDaoImpl
  └─ META-INF
```


Y por último para relacionar las vistas o front del negocio se crea un Servlet, controlador, que será el encargado de relacionar las dos partes de la aplicación.



CONCLUSIONES Y MEJORAS

En este proyecto habría que poner una pasarela de pagos en la que el usuario pudiese abonar la compra y así facturar por los productos vendidos, introduciendo la dirección donde viva el mismo a fin de remitir el producto.

Sería interesante añadir una función para que el administrador pudiera añadir productos a la base de datos y la cantidad de los mismos, dependiendo del stock que tenga en ese momento a fin de que cuando el usuario vaya comprando los mismos y se quede son producto salga un aviso comunicando que no hay existencias.

También sería interesante crear métodos a fin de que el administrador sepa que productos se han comprado y cuando se ha realizado la compra, que usuario hace compras periódicas a fin de ofrecer una suscripción.

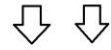
Crear un boot para ver que necesita el usuario y dar posibles respuestas, a fin de obtener información del mismo.

USABILIDAD

Vamos a comentar un poco la usabilidad de la aplicación y el funcionamiento de la misma, al entrar el usuario se registra y se hace login,



[Inicio](#) [Producto](#) [Quienes somos](#) [Conta](#)



[Login](#) [Registro](#)

Correo electrónico

Enter email

Contraseña

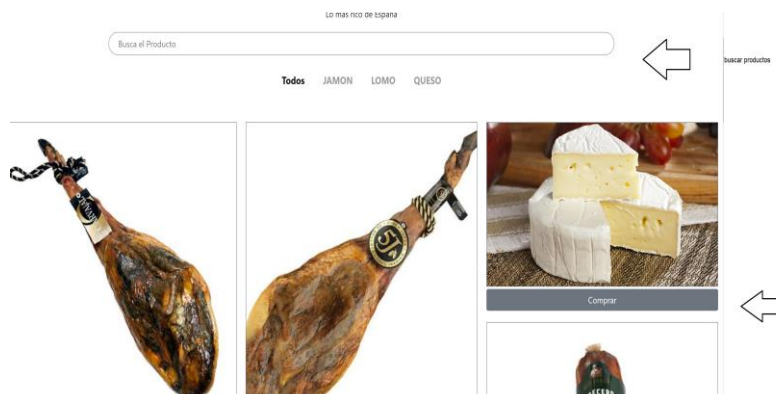
Password

[Entrar](#)



Si hay algún error en sesión se mostraría un mensaje por pantalla a fin de mostrarlo.

Una vez dentro clicarí en la página de productos, estos tendrían una opción para comprar los diferentes productos que hay en la tienda.



Podría darle a comprar producto o buscarlos por categoría en la barra de búsqueda, haciendo una clasificación, una vez seleccionados, daríamos en la pestaña de carrito y aparecerían todos los productos que hay en el carrito del usuario.

Carrito

Eliminar	Producto	Descripción	Precio	Cantidad
Eliminar	JAMON	5 JOTAS	18	1
Eliminar	LOMO	CABECERO	7	1

[Finalizar compra](#)

[Volver a productos](#)

Dándole a finalizar compra, se vaciaría la tabla carrito y se introduce la información en la tabla venta_productos y en la tabla venta, así podría ver la información global el usuario admin.

Ventas totales	
Usuario	Gasto Total
Almudena	22.00
Mercedes	34.00
Almudena	37.00
Almudena	50.00
Almudena	41.00

Productos vendidos			
Cliente	Producto	Precio Producto	Cantidad
Almudena	QUESO	11.00	2
Mercedes	JAMON	12.00	1
Mercedes	QUESO	11.00	2
Almudena	JAMON	18.00	1
Almudena	QUESO	11.00	1
Almudena	QUESO	8.00	1
Almudena	JAMON	12.00	2
Almudena	QUESO	8.00	1
Almudena	LOMO	18.00	1
Almudena	JAMON	18.00	1
Almudena	QUESO	11.00	1

CODIGO MAS PECULIAR.

Al tener varias tablas renacidas, estas con una primarykey Pk se me hizo complejo la función de eliminar productos del carrito, aquí adjunto esa consulta a la base de datos a fin de eliminar una línea del carrito:

```
public void borrarProductoCarrito(int idUsuario, int idProducto) {  
  
    // Creamos la query con los parámetros necesarios.  
    sql = "DELETE FROM Carrito c WHERE c.usuario.idUsuario=:cod1 AND c.producto.idProducto=:cod2";  
    query = em.createQuery(sql);  
    query.setParameter("cod1", idUsuario);  
    query.setParameter("cod2", idProducto);  
  
    // Iniciamos la transacción.  
    tx.begin();  
    // Ejecutamos la consulta.  
    int result = query.executeUpdate();  
    // Hacemos commit (confirmamos la consulta).  
    tx.commit();  
  
}
```

El login de usuario lo realice con un booleano.

```
@Override
public Boolean checkLogin(String email, String password) {
    sql = "SELECT u FROM Usuario u WHERE u.email =:cod1 AND u.password =:cod2";
    query = em.createQuery(sql);
    query.setParameter("cod1", email);
    query.setParameter("cod2", password);
    return query.getResultList().isEmpty() ? false : true;
}
```

Como podemos ver se ha realizado una aplicación web básica que contiene los estudio durante la FP DE DAW, creando así la base para lanzar una tienda e-commerce completa.