

# Universidad de Zaragoza

## Sistemas de Información 23/24

---

### Práctica 5

---



**FGP SHISHAS®**

#### Resumen Proyecto

El sistema de información es una tienda online de cachimbas. En ella se venden cachimbas, elementos de una cachimba y existe la opción de crear una cachimba a su gusto. Además hay un apartado de segunda mano. Los usuarios pueden contactar con la tienda a través del sistema para vender una cachimba. Si la tienda decide comprarla, aparecerá en el apartado de cachimbas de segunda mano.

Pablo Pina Gracia (840020)  
Gonzalo Valero Domingo (842392)  
Flavio Tolosana Hernando (845689)

14 de diciembre de 2023



## Índice

<b>1. Introducción.....</b>	<b>2</b>
<b>2. Objetivo y alcance funcional final de la aplicación.....</b>	<b>3</b>
<b>3. Planteamiento del sistema y Storyboard de la aplicación.....</b>	<b>3</b>
● Iniciar sesión como administrador.....	3
● Registrar a otro administrador.....	4
● Realizar una oferta.....	5
● Añadir un producto.....	5
● Manipular un producto.....	6
● Registrarse como cliente.....	7
● Iniciar sesión como cliente.....	7
● Filtrar por categoría.....	8
● Aceptar o rechazar oferta.....	8
● Ver listado de compras.....	9
● Manipular carrito de compras.....	9
● Vender producto.....	10
● Añadir un producto al carrito.....	11
● Montar Cachimba.....	12
<b>4. Modelo de datos del sistema.....</b>	<b>12</b>
4.1. Diseño de la base de datos.....	12
4.1.1. Modelo E-R.....	12
4.1.2. Modelo relacional.....	14
4.2. Diseño de las clases implementadas en la capa de persistencia de datos.....	15
<b>5. Diferencias entre la primera versión y la final.....</b>	<b>18</b>
<b>6. Procedimiento para el despliegue de la aplicación.....</b>	<b>18</b>
<b>7. Cuestiones necesarias para el uso de la aplicación.....</b>	<b>19</b>
<b>8. Valoración del grupo.....</b>	<b>19</b>



## 1. Introducción

En este trabajo se presenta el diseño, desarrollo e instalación de un sistema de información web con persistencia de datos para la asignatura de Sistemas de Información del Grado de Informática de la Universidad de Zaragoza.

La aplicación web se ha desarrollado siguiendo el paradigma de tres capas: la capa de vista, la capa de modelo y la capa de persistencia. La capa de vista se encarga de la interfaz de usuario y la navegación entre las distintas páginas de la aplicación. La capa de modelo se ocupa de la lógica de negocio y los casos de uso de la aplicación. La capa de persistencia se responsabiliza de la interacción con la fuente de datos que almacena los datos de forma persistente. La fuente de datos utilizada es una base de datos relacional gestionada por un sistema gestor de bases de datos (SGBD).

El objetivo de este trabajo es aplicar los conceptos básicos de programación web con persistencia de datos, así como las buenas prácticas de diseño, desarrollo, pruebas, documentación, seguridad y mantenimiento de sistemas de información web. Para ello, se ha seguido una metodología de trabajo en equipo, basada en la planificación, distribución, coordinación y revisión de las tareas realizadas por los miembros del grupo.

El trabajo se ha dividido en tres bloques: el primero se centra en la configuración del entorno de trabajo y la integración de la capa de persistencia de datos con la capa de modelo; el segundo se enfoca en la completación de la capa de vista y su integración con las demás capas del sistema; y el tercero se dedica a la realización de pruebas de usuario, la implantación del sistema y la generación de la documentación del mismo.



## 2. Objetivo y alcance funcional final de la aplicación

Respecto al objetivo y alcance funcional de la aplicación planteados en la primera práctica, como es obvio ha habido algún cambio, pero tampoco ha habido ningún cambio muy significativo, es bastante parecido a la idea inicial.

Una de las diferencias más grandes, es que en un principio los clientes iban a poder vender sus productos de segunda mano a otros clientes, tipo "Wallapop", pero al final se optó por hacer que nuestra aplicación haga de intermediario, es decir, un usuario vende su producto a la tienda, y después la tienda lo venderá a los usuarios en un apartado de segunda mano. También se ha tomado la decisión de sólo poder comprar si inicias sesión en la web, al añadir al carrito, te exigirá estar registrado. En un principio teníamos pensado que hubiera tres perfiles: administrador cliente registrado y cliente sin registrar. En el caso del cliente sin registrar se guardaba su carrito solo mientras permanecía la ventana abierta, además no iba a poder crear su cachimba personalizada. Al final, concluimos que un usuario sin registrar sólo podía mirar.

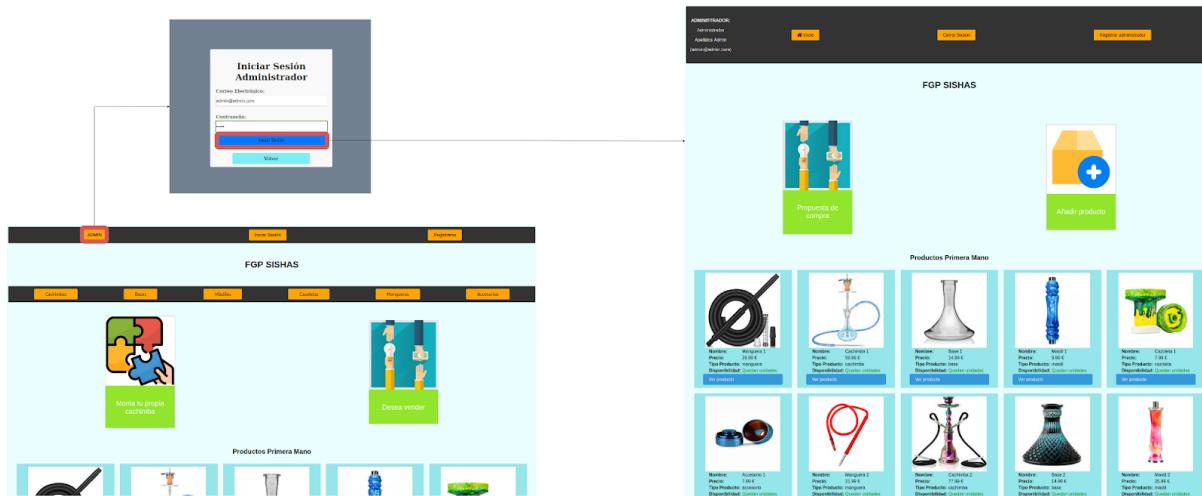
El resto de aspectos, se mantiene fiel a la primera idea.

## 3. Planteamiento del sistema y *Storyboard* de la aplicación

En este apartado se presentarán distintos *Storyboard* de las funcionalidades del sistema. Vamos a dividir los casos de uso en dos bloques bien diferenciados, el primero de ellos serán las actividades a realizar por un administrador:

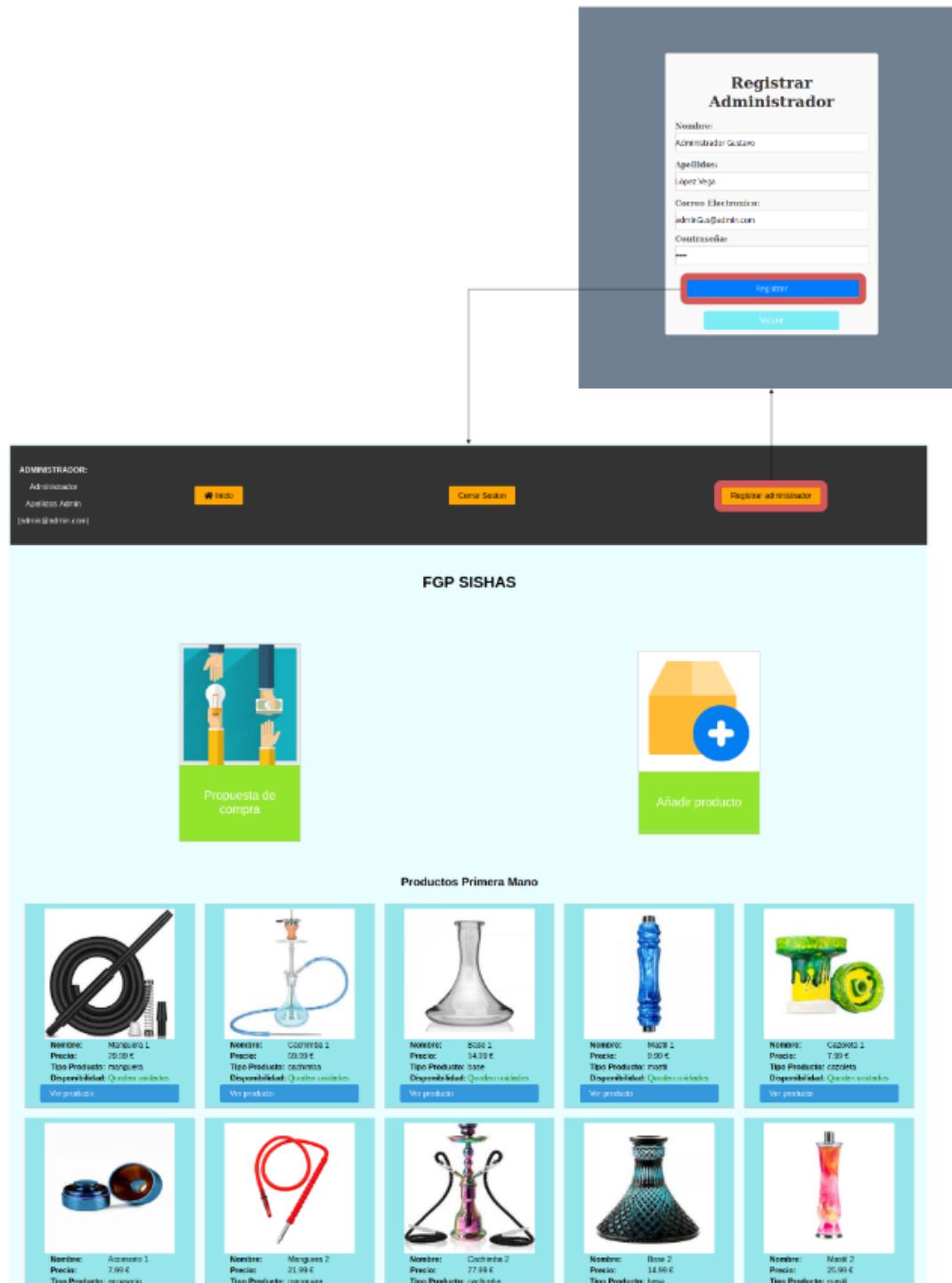
### ● Iniciar sesión como administrador

Para empezar a usar la página como administrador, es esencial iniciar sesión como tal con tu usuario y contraseña.



- Registrar a otro administrador

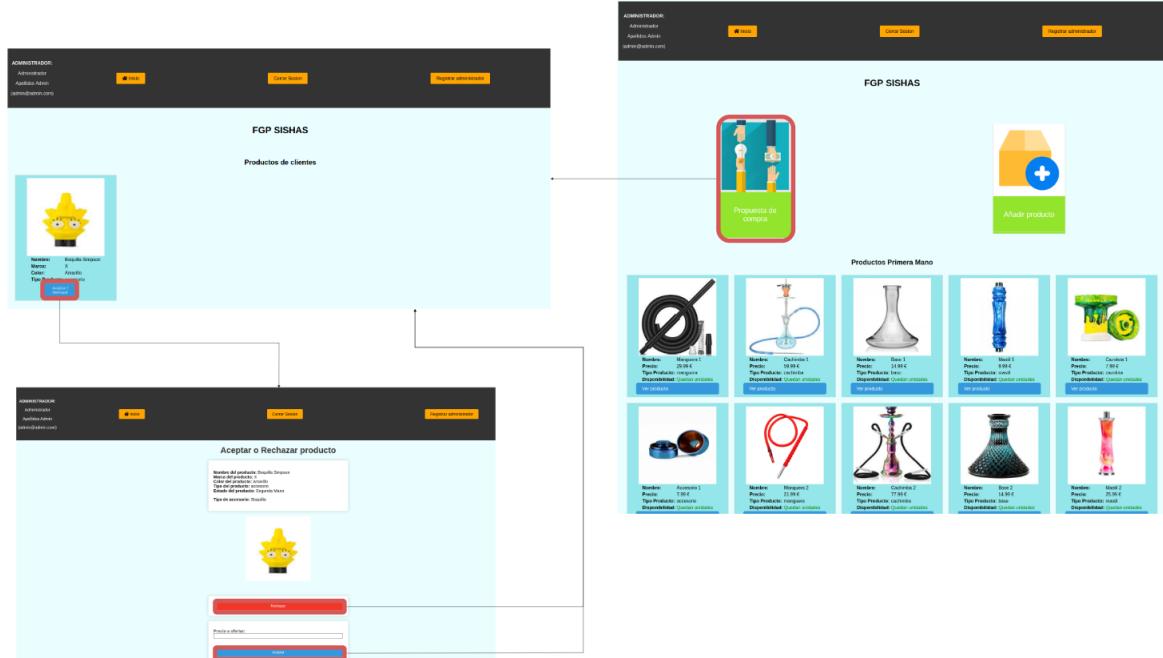
Una de las funcionalidades del administrador es poder crear a otros administradores, para que puedan realizar esas mismas tareas, para ello, obviamente, se debe iniciar sesión como administrador.





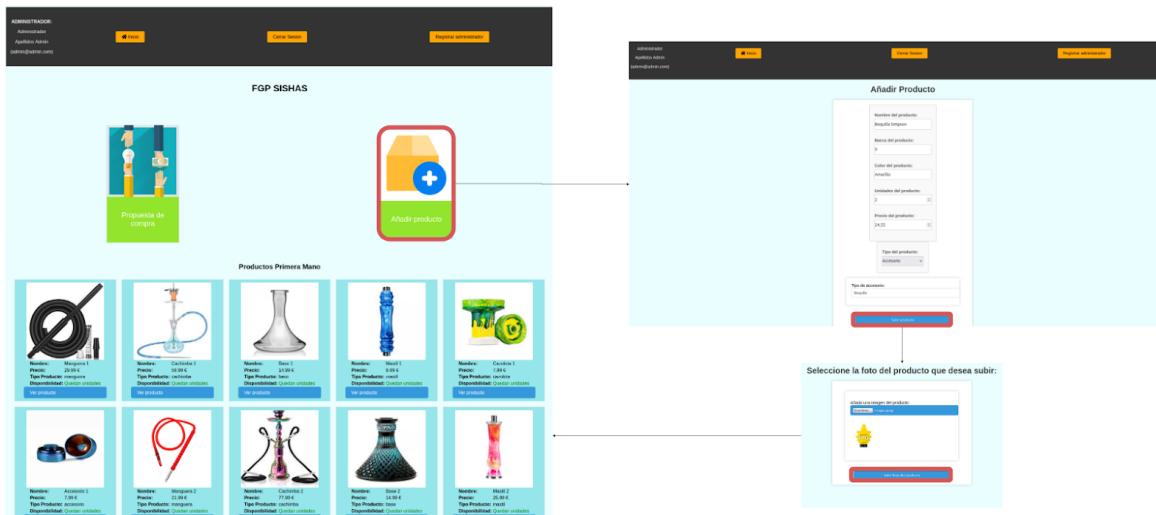
## ● Realizar una oferta

El administrador accede al listado de productos que los clientes quieren vender, posteriormente selecciona uno de los productos. Al administrador se le despliega la información del producto y decide hacerle una oferta (debe insertar un precio) o rechazar el producto.



## ● Añadir un producto

Para añadir un producto el administrador debe clicar en el botón de la pantalla principal llamado “Añadir productos”, tras llenar los datos del producto, como nombre, marca, etc, debe introducir una imagen del producto para finalizar la inserción de este.



## ● Manipular un producto

Al seleccionar un producto del sistema se le desplegará al administrador su información, en este pantalla podrá eliminar el producto del sistema, añadir unas determinadas unidades o cambiarle el precio.

**ADMINISTRADOR:**

Administrador  
Apellidos: Pérez  
Nombre: Juan

[Iniciar](#) [Cerrar Sesión](#) [Regístrate/Iniciar sesión](#)

**FGP SHISHAS**

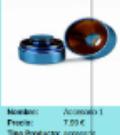


Propuesta de compra



Añadir producto

**Productos Primera Mano**

 <p>Nombre: Manguera 1 Precio: 20.99 € Tipo Producto: manguera Disponibilidad: Quedan unidades <a href="#">Ver producto</a></p>	 <p>Nombre: Cachimba 1 Precio: 35.99 € Tipo Producto: cachimba Disponibilidad: Quedan unidades <a href="#">Ver producto</a></p>	 <p>Nombre: Taza 1 Precio: 21.99 € Tipo Producto: taza Disponibilidad: Quedan unidades <a href="#">Ver producto</a></p>	 <p>Nombre: Model 1 Precio: 15.99 € Tipo Producto: model Disponibilidad: Quedan unidades <a href="#">Ver producto</a></p>	 <p>Nombre: Cachimba 2 Precio: 27.99 € Tipo Producto: manguera Disponibilidad: Quedan unidades <a href="#">Ver producto</a></p>
 <p>Nombre: Accesorio 1 Precio: 1.99 € Tipo Producto: accesorio Disponibilidad: Quedan unidades <a href="#">Ver producto</a></p>	 <p>Nombre: Manguera 2 Precio: 21.99 € Tipo Producto: manguera Disponibilidad: Quedan unidades <a href="#">Ver producto</a></p>	 <p>Nombre: Cachimba 2 Precio: 27.99 € Tipo Producto: cachimba Disponibilidad: Quedan unidades <a href="#">Ver producto</a></p>	 <p>Nombre: Taza 2 Precio: 14.99 € Tipo Producto: taza Disponibilidad: Quedan unidades <a href="#">Ver producto</a></p>	 <p>Nombre: Model 2 Precio: 20.99 € Tipo Producto: model Disponibilidad: Quedan unidades <a href="#">Ver producto</a></p>

**Administrador**  
Apellidos: Pérez  
Nombre: Juan

[Iniciar](#) [Cerrar Sesión](#) [Regístrate/Iniciar sesión](#)

**Información Cachimba 2**

Nombre del producto: Cachimba 2  
Estado del producto: 1  
Color del producto: Roja  
Tipo de producto: cachimba  
Unidades en stock: 10  
Añadir en stock



[Borrar producto](#)

Unidades en stock de producto: 8  
Unidades a añadir:

[Añadir unidades](#)

Precio inicial del producto: 27.99 €  
Precio nuevo:

[Actualizar producto](#)

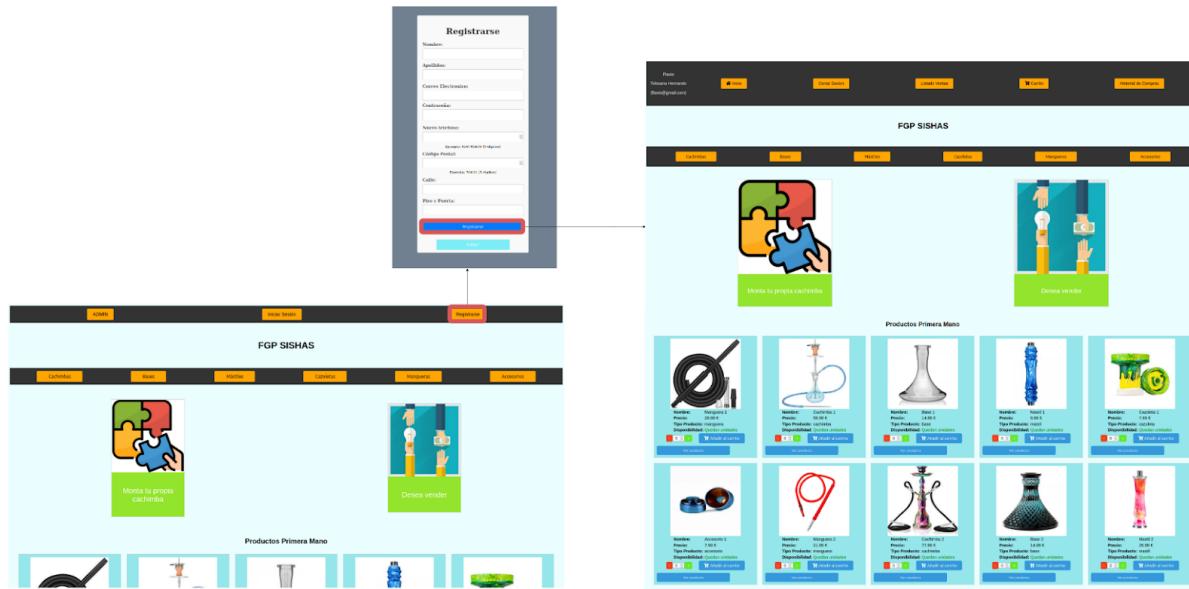
Sistemas de Información 23/24

6

El segundo bloque de casos de usos es el relacionado con el cliente:

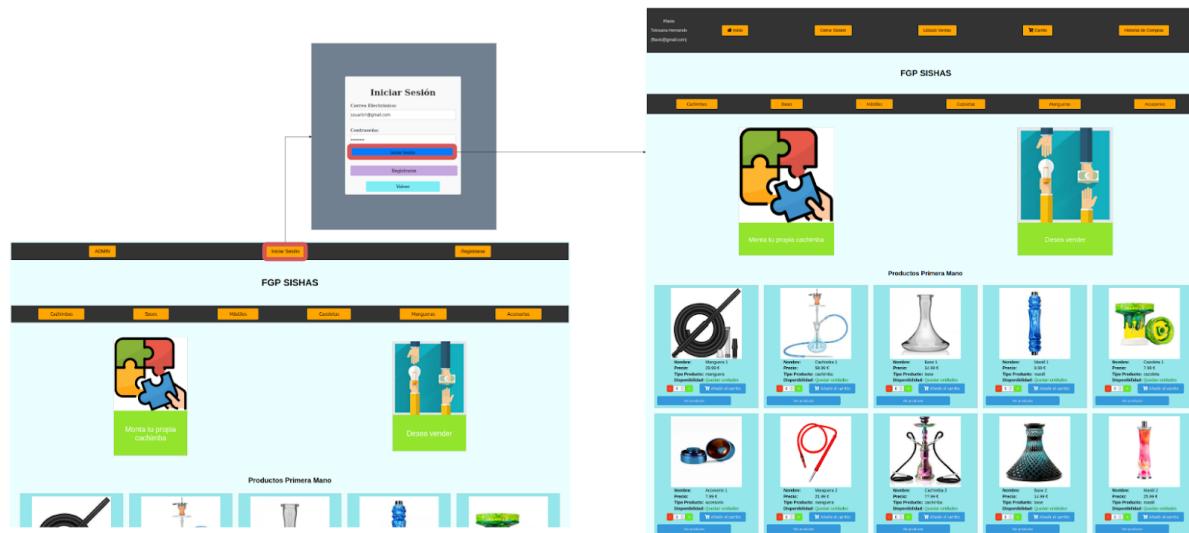
- **Registrarse como cliente**

Para registrarse el cliente debe situarse en la pantalla inicial (sin sesión iniciada) y clicar en el botón “Registrarse” de la parte superior de la pantalla. Tras clicar el botón deberá llenar su información personal y clicar en “Registrarse” para finalizar el registro.



- **Iniciar sesión como cliente**

Una vez registrado el cliente puede clicar en “Iniciar sesión” al lado del botón de registrarse y trás introducir su correo electrónico y contraseña ingresar a la página web con su cuenta.





- Filtrar por categoría

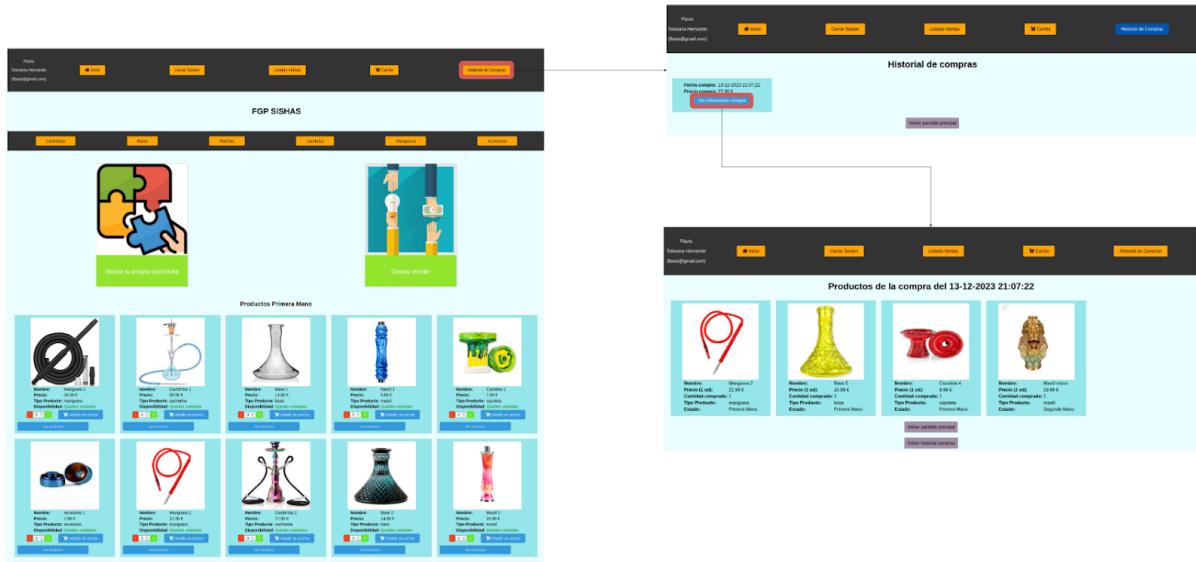
Para filtrar por categoría el cliente debe seleccionar una de las seis categorías representadas como botones debajo del nombre de la página.

- Aceptar o rechazar oferta

El cliente tiene un listado de todos los productos que ha intentado vender, en el caso de que tenga un producto en estado de oferta (el administrador le ha hecho una oferta por su producto) puede rechazar la oferta o aceptarla. En caso de que la acepte debe llenar la información sobre cómo quiere recibir el pago.

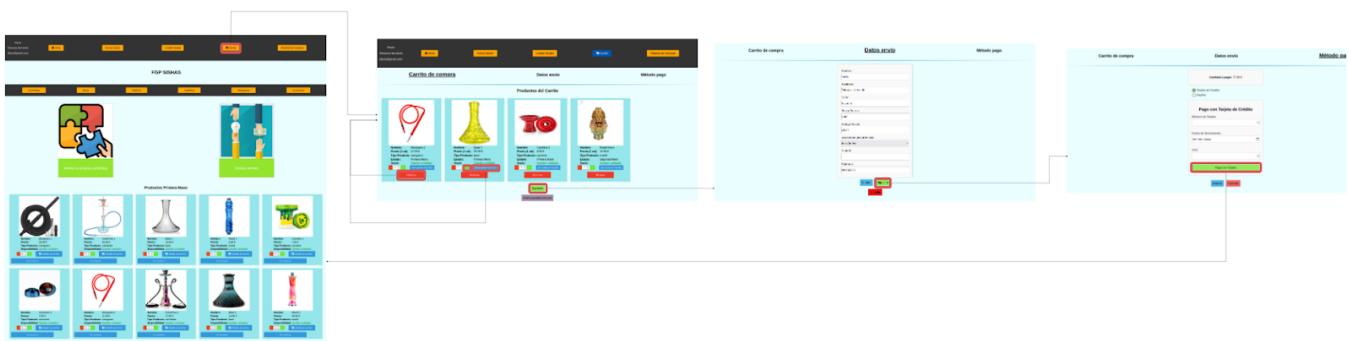
- Ver listado de compras

El cliente tiene un listado de todas las compras realizadas anteriormente, agrupadas por la fecha y horade compras. Puede ver los productos en concreto clicando en cualquier conjunto de compras.



- Manipular carrito de compras

El cliente puede eliminar y añadir unidades del producto desde el carrito. Además puede comprar los productos del carrito clicando en “Siguiente”, tras rellenar o modificar los datos de envío debe rellenar los del pago y clicar en pagar para realizar la compra.





## ● Vender producto

El cliente puede clicar en el botón de “Desea vender” para realizar una venta. Tras esto debe introducir los datos del producto que desea vender y posteriormente añadirle una fotografía. En el momento que la envíe el cliente deberá esperar a recibir una oferta de alguno de los administradores de la página web.

The sequence of screenshots illustrates the process of listing a product for sale:

- Screenshot 1: Product Listing**  
Shows a grid of products under "Productos Primera Mano". One item, a "Cachimba" (Item #1, Price: 20.00 €), has a green "Vender" button highlighted.
- Screenshot 2: Start Selling**  
Shows a large orange "Desea vender" button. A line connects this button to the "Vender" button in Screenshot 1.
- Screenshot 3: Product Data Input**  
A form titled "Introduzca los datos del producto que desea vender:" with fields for Nombre del producto (Product Name), Descripción (Description), Marca del producto (Brand), Color del producto (Color), and Tipo de producto (Product Type). A red "Siguiente paso" (Next Step) button is at the bottom.
- Screenshot 4: Upload Product Photo**  
A form titled "Seleccione la foto del producto que desea vender:" with a placeholder "Añada una imagen del producto" (Add product image) and a note about image requirements. A red "Siguiente paso" (Next Step) button is at the bottom.



## ● Añadir un producto al carrito

En este caso de uso se va a representar el hecho de añadir un producto al carrito, para ello hay dos opciones, o desde el producto así en pequeño fijar la cantidad deseada y darle al botón añadir al carrito, o bien darle a ver producto y en la pantalla del producto añadirlo de la misma manera.

The diagram illustrates the process of adding a product to the cart. It consists of two screenshots of the FGP SHISHAS website:

- Top Screenshot:** Shows the main homepage with a navigation bar at the top. Below the navigation, there are two promotional banners: "Monta tu propia cachimba" (Build your own hookah) and "Desea vender" (Wants to sell). The main content area displays a grid of products under the heading "Productos Primera Mano". Each product card includes a small image, the product name, price, type, availability, and a "Ver producto" (View product) button. A red box highlights the "Añadir al carrito" (Add to cart) button for the first product in the grid.
- Bottom Screenshot:** Shows a detailed product page for "Cachimba 1". The page includes the product name, brand, color, price, type, availability, and a brief description: "Nombre: Cachimba 1 Marca del producto: X Color del producto: Azul Precio del producto: 59.99 Tipo del producto: cachimba Estado del producto: Primera Mano Altura en cm: 48". Below the description is a small image of the hookah. At the bottom of the page is a "Volver pantalla principal" (Return to main screen) button.

- Montar Cachimba

Para montar una cachimba se debe clicar en montar cachimba. Una vez en la página de montar cachimbas se deben añadir los productos que quieras, con la peculiaridad que cuando se añade una base o mástil se filtran las bases y mástiles que se han compatibles con la seleccionada, lo que facilita la compra de una cachimba a partes.

Una vez seleccionados los productos deseados se clica en añadir al carrito y se vuelve a la pantalla principal.



## 4. Modelo de datos del sistema

En este apartado se tratarán los cambios respecto al modelo de datos utilizado para almacenar información en nuestro sistema. Respecto a las herramientas utilizadas no hemos cambiado nada, por lo que no se nombran en este apartado.

### 4.1. Diseño de la base de datos

Como se presentó en la práctica pasada, el diseño de la base de datos ha cambiado notablemente. A continuación se presenta el modelo Entidad-Relación y el modelo relacional.

#### 4.1.1. Modelo E-R

Se han realizado varios cambios respecto a la base de datos anterior, a continuación se adjunta una figura del modelo entidad relación de nuestra base de datos actualizado y se comentarán brevemente alguno de los cambios realizados.

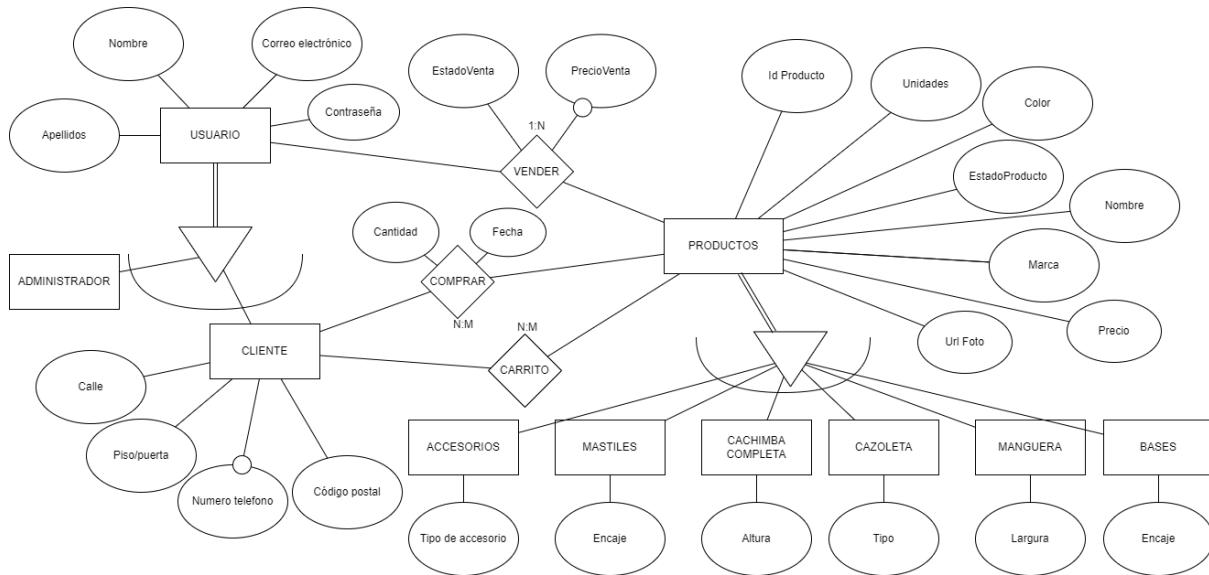
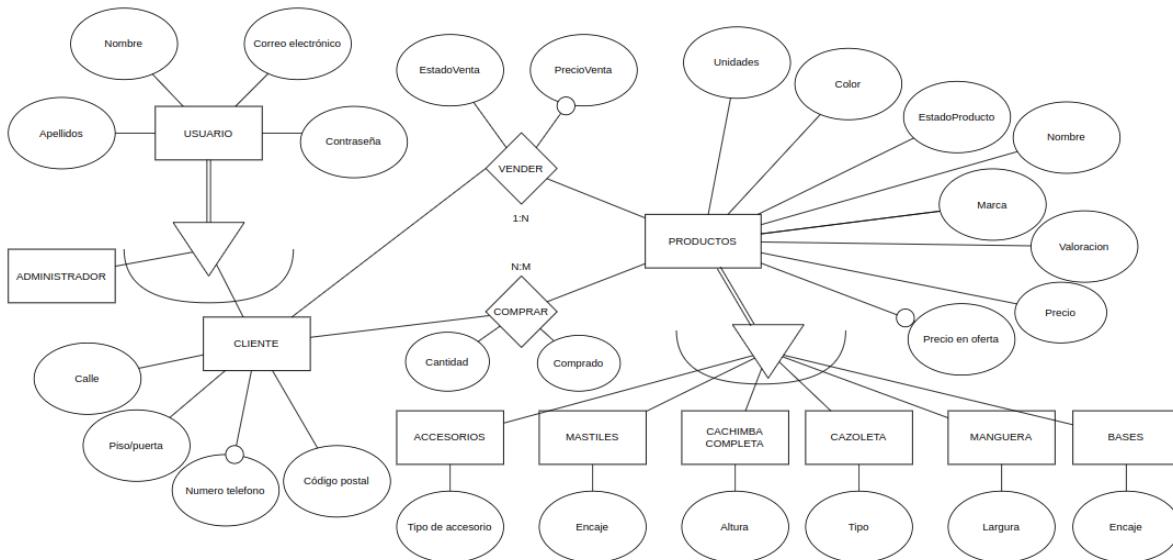


Figura 1: Modelo entidad relación

Se adjunta también una figura con el modelo anterior, para poder compararlos mejor.



Se puede observar, que uno de los mayores cambios, es que se ha añadido una nueva relación "CARRITO" entre "Cliente" y "Productos", en la que se almacenará el carrito que tiene cada cliente preparado para realizar una compra.

Por otro lado, se ha modificado la relación "VENDER", ahora se relaciona "Usuario" con "Producto", en vez de "Cliente" con "Producto". Esta decisión se debe a que de esta forma se almacena en caso de ser un producto de primera mano vendido por la propia empresa se almacena en el correo del vendedor el del propio administrador y en el caso de que sea un producto de segunda mano, se almacena el vendedor real.

Cabe destacar también que se han añadido algunos atributos que no habíamos tenido en cuenta, como puede ser el tema de la fotografía de un producto, además de la eliminación de algún otro atributo, como por ejemplo la valoración o el precio de oferta, que por decisiones internas respecto a la implementación se ha decidido prescindir de ellas.



#### 4.1.2. Modelo relacional

```
Usuario= (
    correolectronico: VARCHAR (50);
    nombre: VARCHAR (20), NO NULO;
    apellidos: VARCHAR (50), NO NULO;
    contrasegna: VARCHAR (20), NO NULO;
    calle: VARCHAR (50);
    pisoPuerta: VARCHAR (20);
    codigopostal: VARCHAR (6);
    numerotelefono: VARCHAR (9);
    esadmin: bool

);

Producto = (
    idproducto: entero;
    nombre: VARCHAR (50);
    estadoProducto: VARCHAR (15);
    marca: VARCHAR(20), NO NULO;
    color: VARCHAR(20), NO NULO;
    precio: entero;
    tipoProducto: entero, NO NULO;
    estadoVenta: VARCHAR (20), NO NULO;
    precioVenta: entero;
    unidades: entero, NO NULO;
    urlfoto: VARCHAR(50), NO NULO;
    usuariovendedorproducto: VARCHAR(50) , NO NULO;

    clave ajena (tipoProducto) referencia a Tipos(idTipo);
);

Tipos= (
    idtipo: entero;
    nombretipo: VARCHAR(20), NO NULO;
);

AtributosTipo = (
    idatributostipo: entero;
    tipoproducto: entero;
    nombreatributo: VARCHAR(20), NO NULO;

    clave ajena (tipoproducto) referencia a Tipos(idtipo);
);
```



```
Atributos = (
    idatributo: entero;
    producto: entero, NO NULO;
    atributotipo: entero, NO NULO;
    valoratributo: VARCHAR (20), NO NULO;

    clave ajena (producto) referencia a Productos(idproducto);
    clave ajena (atributotipo) referencia a AtributosTipo(idAtributostipo);
);

Compras= (
    correoelectrónico: VARCHAR (50);
    idproducto: entero, NO NULO;
    cantidad: entero, NO NULO;
    tiempocompra: VARCHAR(20), NO NULO;
    precio: entero, NO NULO;

    clave ajena (correoelectrónico) referencia a Usuarios(correoelectrónico);
    clave ajena (idproducto) referencia a Producto(idproducto);
);

Carrito= (
    correoelectrónico: VARCHAR (50);
    idproducto: entero, NO NULO;
    cantidad: entero, NO NULO;

    clave ajena (correoelectrónico) referencia a Usuarios(correoelectrónico);
    clave ajena (idproducto) referencia a Producto(idproducto);
);
```

## 4.2. Diseño de las clases implementadas en la capa de persistencia de datos

Respecto a los VO no hay ningún cambio, ya que hay uno por cada tabla creada en la base de datos. Pero en cuanto a los DAO si que se nota un gran cambio, ya que en primera instancia se plantearon un número superior, siendo cada uno más concreto, pero al final se optó por hacer menor cantidad y más generales, quedando uno para gestionar el carrito, otro para manejar los productos, otro para la compra, y por último uno para manejar los usuarios. A continuación se da una breve explicación sobre los distintos DAO implementados para nuestro sistema:

- En primer lugar, se ha implementado el DAO “ManipularUsuarioDAO.java”, en el que como su nombre indica sirve para temas relacionados con los usuarios y consta de los siguientes métodos:



- existeUsuario: devuelve un valor booleano, true si existe un usuario con el correo que le pasas y false si no existe. Comprueba si en la base de datos hay algún usuario con dicho correo.
- validarUsuario: devuelve también un valor booleano. Se le pasa un correo y una contraseña, y lo que hace es ver si existe dicho correo con la contraseña pasada. También se le proporciona un booleano de si es administrador o no, para que busque en una tabla o en otra.
- insertarUsuario: recibe como parámetros un usuario y un booleano al igual que antes para saber si es administrador o no. Devuelve true si se añade correctamente la entrada a la base de datos y false si no se puede añadir porque hay algún tipo de error como que ya exista un usuario con dicho correo por ejemplo.
- obtenerUsuario: recibe como parámetro un correo electrónico y devuelve el usuario correspondiente a dicho correo. Si no existe un usuario con dicho correo, devuelve un usuario totalmente vacío.
- Por otro lado, se ha creado el DAO “ManipularProductoDao.java”, en el que se operará con los productos, para ello contiene los siguientes métodos:
  - listaProductos: recibe un booleano para filtrar según si es de primera mano o de segunda mano, y un string para saber el estado del producto(vendido, oferta, espera, etc.), devuelve una lista con todos los productos que cumplan dichas condiciones.
  - tipoProducto: recibe como parámetro un string con el nombre del tipo de producto (cachimba, cazoleta, accesorio, etc.), y lo que hace es devolver el idTipoProducto asociado a dicho nombre. Si no lo encuentra devuelve -1.
  - tipoAtributo: recibe como parámetro el id de un tipo de producto, y devuelve como entero el tipo de atributo que tiene dicho tipo de producto. Sino, al igual que anteriormente, devuelve -1.
  - insertarProducto: recibe un producto y lo inserta en la base de datos. Si todo va bien devuelve como resultado el id de dicho producto, en caso contrario, devuelve -1.
  - insertarAtributos: recibe un atributo y lo inserta en la base de datos. Si todo va bien devuelve el id de dicho atributo, en otro caso, devuelve -1.
  - modificarEstadoVenta: recibe como parámetros un identificador de producto, un string con el estado de venta deseado y un entero con el precio de venta. Se cambia el estado de venta de dicho producto y el precio de venta con los parámetros pasados.
  - agadirFoto: recibe como parámetros un identificador de producto y un string con la url de la foto que se desea añadir. Añade la foto al producto deseado en la base de datos.
  - agadirUnidades: recibe como parámetros el identificador de un producto y un entero que representa las unidades que se desean añadir. Al ejecutar esto añade las unidades deseadas de este producto a la base de datos, es decir, suma las que ya había a las añadidas mediante el método.
  - eliminarProducto: recibe como parámetro un identificador de producto y elimina la entrada de la base de datos correspondiente a ese producto.



- valorAtributoExtra: recibe como parámetros el identificador de un producto y el identificador de un tipo de atributo. Devuelve el valor del atributo extra asociado a dicho producto, que el atributo variará en función del tipo de producto del que se trate.
- datosProducto: dado un identificador de producto, devuelve el producto entero, con todos sus atributos. Si no existe, devuelve un producto vacío.
- En tercer lugar, nos encontramos con el DAO “ManipularCarritoDao.java”, desde el que se manejará el carrito con los siguientes métodos:
  - listaCarrito: recibe como parámetro un correo electrónico y devuelve una lista que contiene los productos que tiene el usuario asociado a ese correo en su carrito.
  - actualizarCantidadProductoCarrito: recibe un carrito y modifica la cantidad seleccionada por un usuario de un producto de este.
  - eliminarProductoCarrito: recibe un correo electrónico y un identificador de un producto. Elimina del carrito del cliente al que le corresponde el correo el producto deseado.
  - existeProductoEnCarrito: recibe un correo y un identificador de producto al igual que en el caso anterior, y devuelve un booleano, true si dicho producto se encuentra en el carrito del cliente o no.
  - insertarProductoCarrito: recibe un carrito e insertará el producto deseado en el carrito, si ya estaba aumentará la cantidad y sino lo añadirá.
  - possibleCompra: recibe un correo y un identificador de producto. Si el número de unidades que quedan en stock es mayor o igual que la cantidad que hay en el carrito devolverá las unidades restantes, pero si quedan menos de las que hay en el carrito, se devolverá -1.
  - possibleRealizarCompra: recibe un correo electrónico y devuelve true si y solo si todos los productos que tiene en su cesta se pueden comprar, es decir, si el número de unidades de cada producto que tiene en su carrito es menor o igual que el número de unidades que hay en stock de cada producto.
  - sumaPrecioCarrito: recibe un correo y devuelve el precio total del carrito, es decir, la suma de todos los productos que están en el carrito.
  - reducirUnidadesProducto: recibe un identificador de producto y un entero que representa las unidades que se desea restar. Elimina las unidades especificadas de dicho producto.
  - insertarProductoCompra: recibe un correo electrónico, un identificador de producto, una cantidad, la fecha de compra y el precio, y almacena la compra realizada con todos esos datos.
  - realizarCompra: recibe un correo y un string con la fecha de compra, y lo que hace es eliminar las entradas del carrito de este usuario y añadir los productos a la tabla “Compras”.
- Por último, se implementará el DAO “ManipularCompraDAO.java”, para manipular todos los datos relacionados con la tabla “Compras”. Constará de varios métodos pero aún no están implementados.
  - sumaPrecioCompra: recibe como parámetros un correo electrónico y un tiempo de compra, devuelve el precio que le costó al usuario dicha compra.



- listaCompras: recibe como parámetro un correo electrónico y devuelve una lista con las fechas en las que dicho usuario ha hecho compras.
- listaProductosCompra: recibe como parámetros un correo electrónico y un tiempo de compra, devuelve un listado de productos que contiene los productos que compró en dicha operación.

## 5. Diferencias entre la primera versión y la final

Como bien se ha comentado en otros apartados, no hay mucha variación respecto a las funcionalidades, pero sí que existe alguna. Por ejemplo, un cambio que se puede considerar negativo es que no poder realizar compras sin registrarse, pero consideramos que tampoco cuesta tanto identificarse para hacer compras, no es un inconveniente. Puede ser negativo, pero no es tan descabellado. Se tomó esta decisión por el tema de guardar el carrito del usuario.

Otro aspecto que puede ser peor también peor sobre todo a nivel visual es en el apartado de montar tu propia cachimba, que se visualicen todos los elementos que la conforman uno encima de otro para simular la cachimba completa montada, pero nos encontramos con el problema de la dimensión de las imágenes, por lo que se decidió ponerlos en modo de resumen arriba del todo de la pantalla, para que se puedan ver todos juntos, a pesar de que no pueda ser tan visual como en la idea inicial.

En cuanto a una posible mejora, puede ser el tema de crear nuevos perfiles de administrador, que en la idea inicial ni lo contemplamos. Ahora un usuario ya administrador, puede registrar a otro administrador para que él pueda desarrollar las actividades correspondientes.

## 6. Procedimiento para el despliegue de la aplicación

Para desplegar la aplicación se ha utilizado Docker version 24.0.7. Se ha creado una carpeta Docker, en la cual hay dos directorios (postgres y tomcat) y un script .sh para automatizar el despliegue.

En la carpeta postgres hay un Dockerfile que indica que la imagen que se va a utilizar para crear el contenedor es *bitnami/postgresql:16.1.0*. En un principio también estaba una copia de la carpeta main de la base de datos postgresql, pero por problemas de compatibilidad entre la imagen y la base de datos, se optó por una alternativa. En vez de la carpeta main, hay un script .sql en el que están las instrucciones para crear y poblar la base de datos.

Mediante la siguiente instrucción en el Dockerfile, *ADD crear\_BD.sql /docker-entrypoint-initdb.d* se consigue que al arrancar la base de datos de la imagen, se ejecute dicho script.

La carpeta tomcat contiene los archivos *context.xml* y *server.xml* del servidor apache-tomcat-9.0.83. Además está el controlador de la base de datos *postgresql-42.2.5.jar*. La imagen que se ha utilizado para crear el contenedor de tomcat ha sido *bitnami/tomcat:9.0.83*. Y finalmente el archivo .war del proyecto, *ROOT.war*. Se llama así al .war para que pueda ser accesible desde la raíz del puerto 8080 de localhost, es decir, en un navegador en internet se podía ver la página web en *localhost:8080*.



Para desplegar el sistema de información basta con ejecutar el script despliegue.sh. Lo más seguro es que haya que darle permisos de ejecución con chmod u+x despliegue.sh.

## 7. Cuestiones necesarias para el uso de la aplicación

Para usar la aplicación con sus plenas funcionalidades, puedes registrarte con una cuenta nueva o iniciar sesión con una cuenta existente. A continuación se adjunta una tabla con alguno de los usuarios ya creados y sus contraseñas:

Correo electrónico	Contraseña
flavio@gmail.com	flaviopswd
pablo@gmail.com	pablopswd
gonzalo@gmail.com	gonzalopswd

Para los usuarios administradores, se debe iniciar sesión con uno de ellos y si se desea crear un administrador nuevo se deberá crear desde la cuenta de otro administrador, se adjunta una tabla con los administradores creados y sus correspondientes contraseñas:

Correo electrónico	Contraseña
admin@admin.com	admin

## 8. Valoración del grupo

Valorando el trabajo completo una vez terminado, en general estamos satisfechos, ya que después del trabajo llevado a cabo y las complicaciones que han surgido, se ha podido realizar de una manera bastante correcta.

Los dos aspectos que más nos han costado han sido, guardar fotos enviadas por el cliente cuando quiere vender un producto, que nos quedamos atascados hasta que recibimos ayuda de los profesores. Por otro lado, también nos quedamos atascados a la hora del despliegue. El contenedor que contenía la base de datos postgresql daba error al crearlo mediante la carpeta main (copia de la carpeta main de postgresql). Daba un problema de incompatibilidad. Al final se optó por utilizar la imagen de postgresql y un script .sql para crear y poblar la base de datos.

El resto de trabajo ha sido costoso y en algún momento algo complicado debido a la complejidad de estructura y a la inexperiencia realizando proyectos de este tipo.

En cuanto a lo que se ha aprendido con estas prácticas, sobre todo es a entender cómo funciona todo detrás de un sistema de información, en concreto de una tienda online, además de aprender a poner bonito un simple código HTML.

Algo que nos habría gustado implementar y era nuestra idea inicial, era poner más visual el conjunto de las partes de la cachimba al hacerla a tu gusto, pero como bien se ha comentado, se nos planteó el inconveniente de la dimensión de las fotografías, ya que se vería por ejemplo la cazoleta, que es notablemente más pequeña que una base, de un tamaño similar. Por otro lado también nos hubiera gustado implementar el pago real, no sólo guardar unos datos y no hacer nada más. Entender cómo funciona realmente la pasarela de pago a través de internet, aunque entendemos que esos conceptos se darán en alguna otra asignatura más concreta sobre el tema, pero es una cosa que nos llamaba bastante la atención.

La base de datos ha sido diseñada para que en el caso de añadir algún nuevo tipo de producto o atributos a algún tipo ya existente se pueda hacer de una manera simple sin necesidad de alterar la base de datos. Hacer esto hubiera facilitado algún cambio de decisión de último momento sobre los productos que quisiéramos vender.

Cronograma	Flavio Tolosana Hernando (845689)	Gonzalo Valero Domingo (842392)	Pablo Pina Gracia (840020)
Decisión del sistema de información a realizar	4	4	4
Realización del diseño de las pantallas y casos de uso de la aplicación	6	6	6
Diseño base de datos	3	6	6
Programar VOs y DAOs de la aplicación	8	8	8
Implementación pantallas y código java	23	18	16
Despliegue de la aplicación	8	0	0
Redacción de la memoria final	2	3	3