

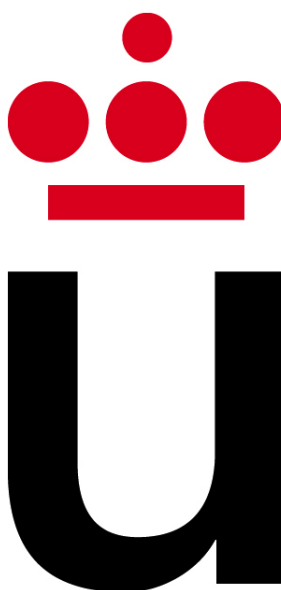
PRÁCTICA 1

Diseño de Aplicaciones Web

IMPLEMENTACIÓN DE UNA TIENDA VIRTUAL

GII+GIS: Germán Alonso Azcutia

GIS+MAT: José Ignacio Escribano Pablos



MÓSTOLES, 29 DE MARZO DE 2015

Índice de figuras

1.	Patrón de arquitectura Modelo Vista Controlador (MVC)	1
2.	Diagrama de casos de uso de la aplicación	2
3.	Diagrama de clases de la aplicación	3
4.	Diagrama de secuencia de comprar producto	4
5.	Diagrama de secuencia de añadir producto como administrador	5
6.	Vista index.html	8
7.	Vista product.html	9
8.	Vista cart.html	10
9.	Vista order.html	10
10.	Vista admin.html	11
11.	Vista new.html	12
12.	Vista edit.html	12
13.	Vista orders.html	13
14.	Vista orders.html (actualizar estado)	13

Índice

1. Introducción	1
2. Diagramas UML	2
2.1. Diagrama de casos de uso	2
2.2. Diagrama de clases	3
2.3. Diagramas de actividad	3
3. Descripción de las clases	6
3.1. Modelo	6
3.1.1. Clase Product	6
3.1.2. Clase ProductWithQuantity	6
3.1.3. Clase Cart	6
3.1.4. Clase Order	6
3.1.5. Interfaz ProductRepository	6
3.1.6. Interfaz OrderRepository	6
3.2. Controlador	7
3.2.1. Clase ProductController	7
3.2.2. Clase OrderController	7
3.2.3. Clase AdminController	7
3.2.4. Clase StartController	7
3.3. Vista	7
4. Recorrido por la aplicación	8
4.1. index.html	8
4.2. product.html	9
4.3. cart.html	9
4.4. order.html	9
4.5. admin.html	11
4.6. new.html	11
4.7. edit.html	11
4.8. orders.html	13

1. Introducción

Para la realización de la práctica vamos a usar el patrón de arquitectura de software **Modelo-Vista-Controlador(MVC)**, que como su propio nombre indica, define la organización independiente del Modelo (información y lógica de negocio), la Vista (interfaz con el usuario) y el Controlador (intermediario entre modelo y vista).

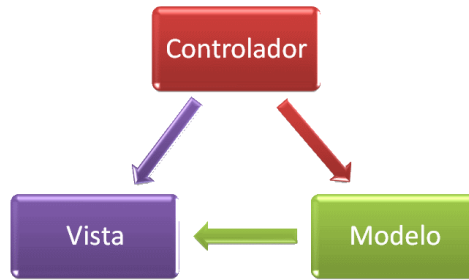


Figura 1: Patrón de arquitectura Modelo Vista Controlador (MVC)

Para llevar a cabo la explicación de la responsabilidad de cada clase, vamos a dividir dicha explicación en las tres partes del patrón.

Pero antes, veamos los diagramas UML de nuestra aplicación web.

2. Diagramas UML

2.1. Diagrama de casos de uso

El diagrama de casos de uso de la Figura 2 muestra que la aplicación tiene dos roles distintos dentro de la aplicación: usuario y administrador. El primero de ellos puede ver productos, añadirlos al carrito, ver su carrito, y el segundo puede añadir productos al catálogo de la tienda, modificarlos, ver los pedidos, así como todas las acciones derivadas de las anteriores.

Esto supone que hay que distinguir casos distintos casos, ya sea para un usuario o para el administrador.



Figura 2: Diagrama de casos de uso de la aplicación

2.2. Diagrama de clases

El diagrama de clases muestra todas las clases que manejarán la lógica de la aplicación. Estas clases son ProductWithCantidad, Cart, Product y Order. La descripción de cada clase se puede ver en la Sección 3.

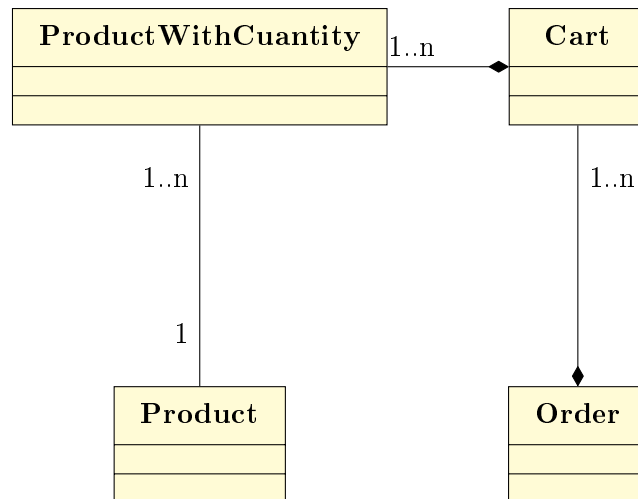


Figura 3: Diagrama de clases de la aplicación

2.3. Diagramas de actividad

Como hemos comentado anteriormente, debemos distinguir tareas para un usuario normal y el administrador. Esto se refleja en los diagramas de actividad: en el caso de un usuario normal no es necesario iniciar sesión (en el enunciado de la práctica se especifica que no hay usuarios, salvo el administrador), mientras que en el administrador sí que es necesario.

Un ejemplo de lo anterior se puede ver en las Figuras 4 y 5 donde no es necesario iniciar sesión para comprar un producto de la tienda, y, por el contrario, sí que lo es para añadir un producto al catálogo de la tienda.

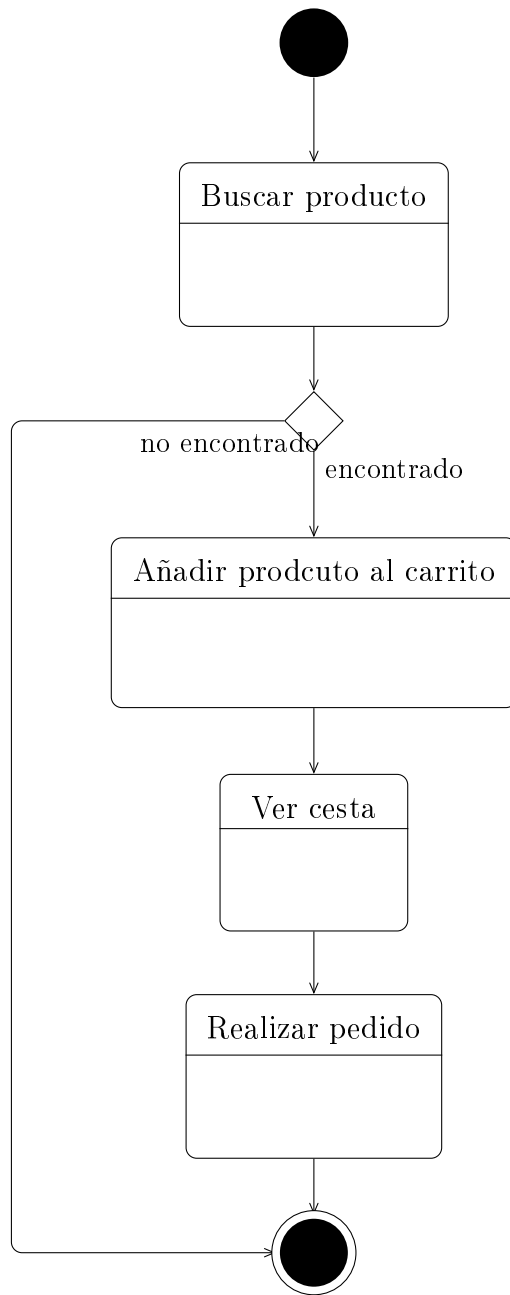


Figura 4: Diagrama de secuencia de comprar producto

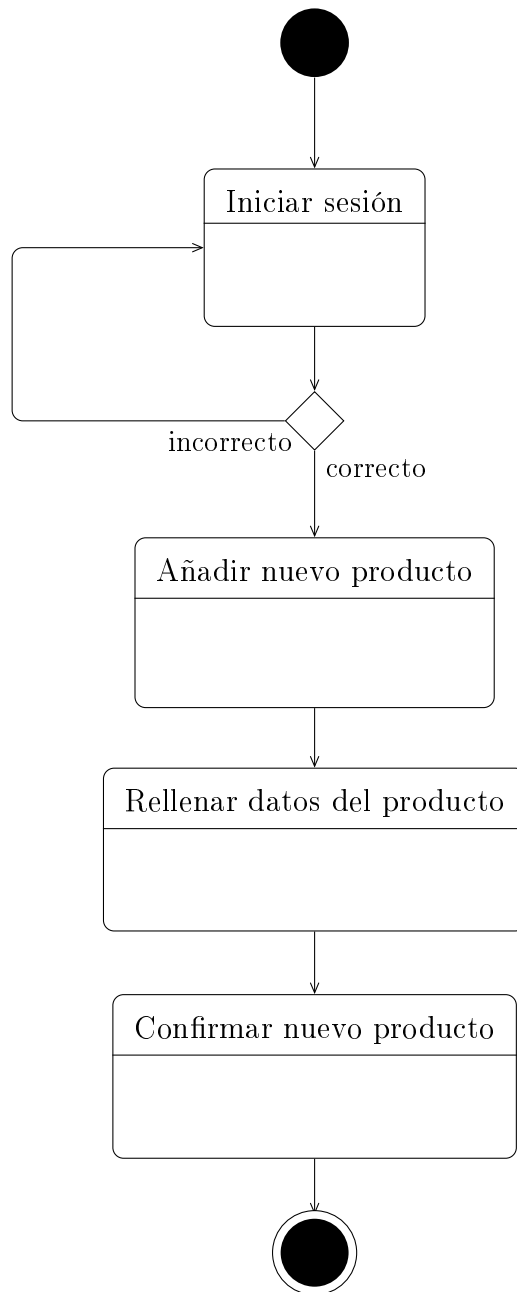


Figura 5: Diagrama de secuencia de añadir producto como administrador

3. Descripción de las clases

3.1. Modelo

3.1.1. Clase Product

Esta clase define el objeto producto y las propiedades y métodos de los que dispondrán los productos de nuestra web.

3.1.2. Clase ProductWithQuantity

La clase `ProductWithQuantity` define un producto que estará en el carro de compra y la cantidad de dicho producto almacenada en el carro. Por tanto, se compone de un producto y su cantidad.

3.1.3. Clase Cart

Esta clase define el carro de compra de cada usuario. Está compuesto por un `ArrayList` de `ProductWithQuantity` (que serán el total de productos en la cesta) y el precio total del carro.

3.1.4. Clase Order

La clase `Order` define el pedido del usuario, es decir, la confirmación de querer comprar su carro de compra. Por ello, esta clase contiene un atributo carro, además del estado del pedido (pendiente o preparado) y los datos correspondientes del usuario que ha realizado el pedido.

3.1.5. Interfaz ProductRepository

Es una clase que extiende de `CrudRepository`, interfaz genérica que ofrece operaciones tipo CRUD (Create-Read-Update-Delete) para acceder a los datos de la base de datos. Nosotros lo utilizaremos en esta clase para realizar consultas sobre los productos en la BD.

3.1.6. Interfaz OrderRepository

Al igual que `ProductRepository`, esta clase extiende de `CrudRepository`, pero esta vez la clase nos servirá para realizar consultas a la BD sobre los pedidos.

3.2. Controlador

3.2.1. Clase ProductController

Esta clase es el controlador que se encarga de gestión todas las acciones relacionadas con los productos del catálogo. Ejemplo de esto puede ser mostrar información de productos, listar todos los productos, añadir productos al carrito de la compra, buscar (tanto por categorías, por nombre o precio).

Este controlador recoge las peticiones del repositorio y lleva la información proporcionada por el repositorio hasta la vista correspondiente a la acción requerida.

3.2.2. Clase OrderController

Análogo al anterior, pero gestionando los pedidos. Realiza acciones tales como mostrar pedidos, confirmarlos o pedir los datos (nombre y apellidos) del pedido.

3.2.3. Clase AdminController

Este controlador es el encargado de gestionar las peticiones desde la sección de administrador como añadir nuevo producto, borrar productos, editar productos, cambiar estados de los pedidos, etc.

3.2.4. Clase StartController

Esta clase es un controlador que implementa la interfaz `CommandLineRunner`, que permite ejecutar algunos comandos al iniciar la aplicación. Se encarga de introducir algunos productos y pedidos de prueba que sirvan para probar el correcto funcionamiento de la aplicación. En la versión de producción, este controlador se eliminaría.

3.3. Vista

La interfaz de nuestra aplicación se divide en las siguientes vistas: `\index.html`, `\product.html`, `\cart.html`, `\admin.html`, `\new.html`, `\edit.html`, `\order.html` y `\orders.html` que se explicarán en la sección de Recorrido por la aplicación

4. Recorrido por la aplicación

4.1. index.html

Esta es la vista principal de nuestra aplicación web. Como se puede observar en la Figura 6, en la parte superior los usuarios pueden buscar productos por categoría y por nombre, y el administrador podrá loguearse con el menú desplegable. Estas opciones también estarán disponibles en las demás vistas.

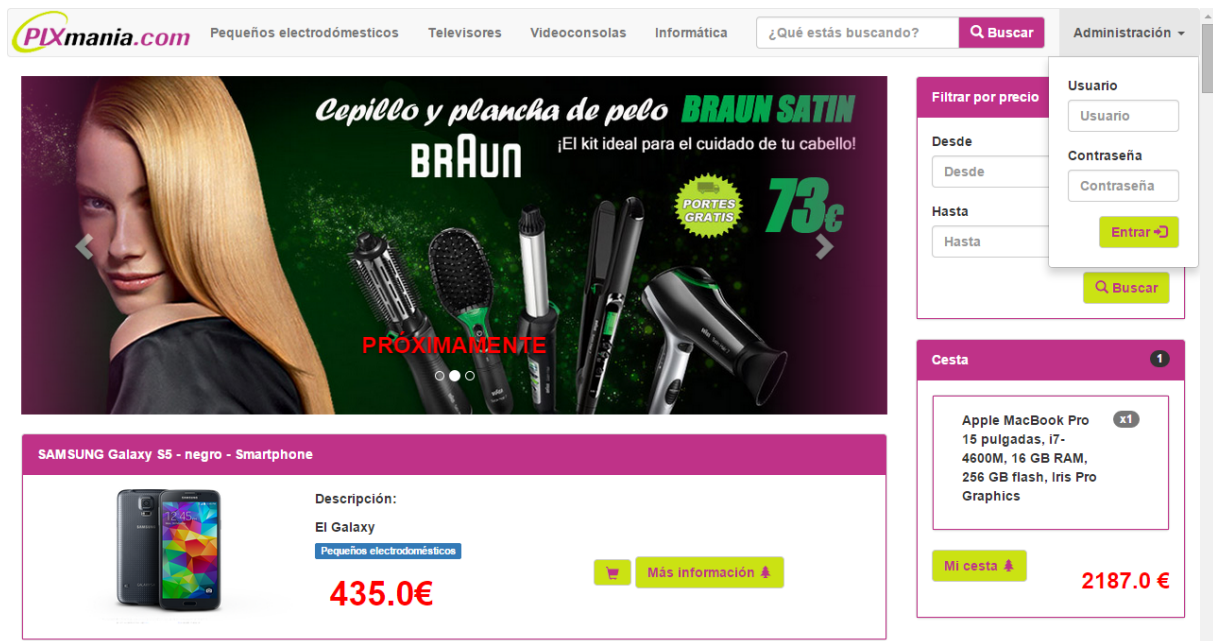


Figura 6: Vista index.html

En el cuerpo de la vista encontramos en la parte izquierda el listado de los productos de la web, y si han realizado una búsqueda, aparecerán los productos que concuerden con dicha búsqueda. Para cada producto aparecen 2 botones, el botón **Carro** que añadirá el producto a la cesta y el botón **Más información**, que nos llevará a la vista `/product.html` para ver en detalle dicho producto.

Por último, en la parte derecha tenemos la opción de filtrar los productos por precio y el carro de compra, donde se pueden ver los productos añadidos y un botón para acceder a la vista `/cart`, que contendrá el carro de compra (Figura 7).

4.2. product.html

En esta vista (Figura 7) se puede observar el producto seleccionado con la descripción completa y la opción de añadirlo al carro.



Figura 7: Vista product.html

4.3. cart.html

En esta vista tenemos (Figura 8) todos los productos que han sido añadidos a la cesta de compra. En ella podemos aumentar la cantidad de un producto o, si lo deseamos, borrar un producto de la cesta. Cuando ya esté lista la cesta, se podrá pulsar el botón de hacer pedido, que nos llevará a la vista `order.html` con un mensaje confirmando el pedido.

4.4. order.html

Esta vista (Figura 9) como se puede observar, podemos ver el carro de compra que hemos hecho y tenemos un pequeño formulario para rellenar antes de proceder a confirmar el pedido (botón realizar pedido). Al realizar el pedido, se volverá a la vista `/index.html`

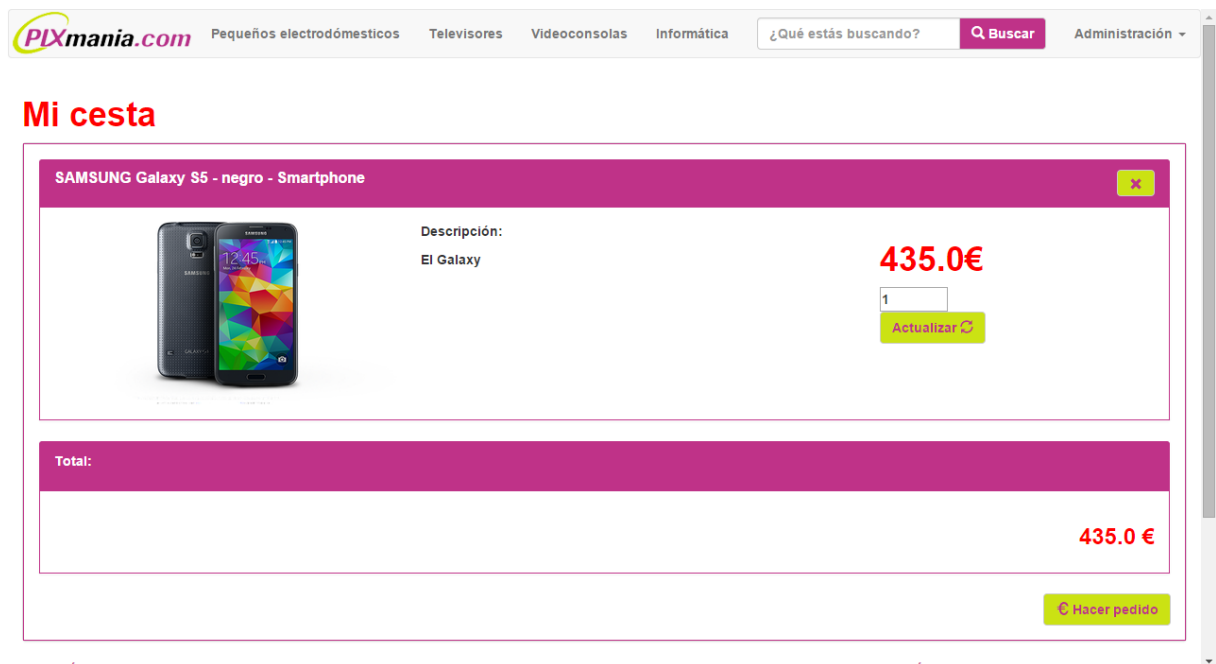


Figura 8: Vista cart.html

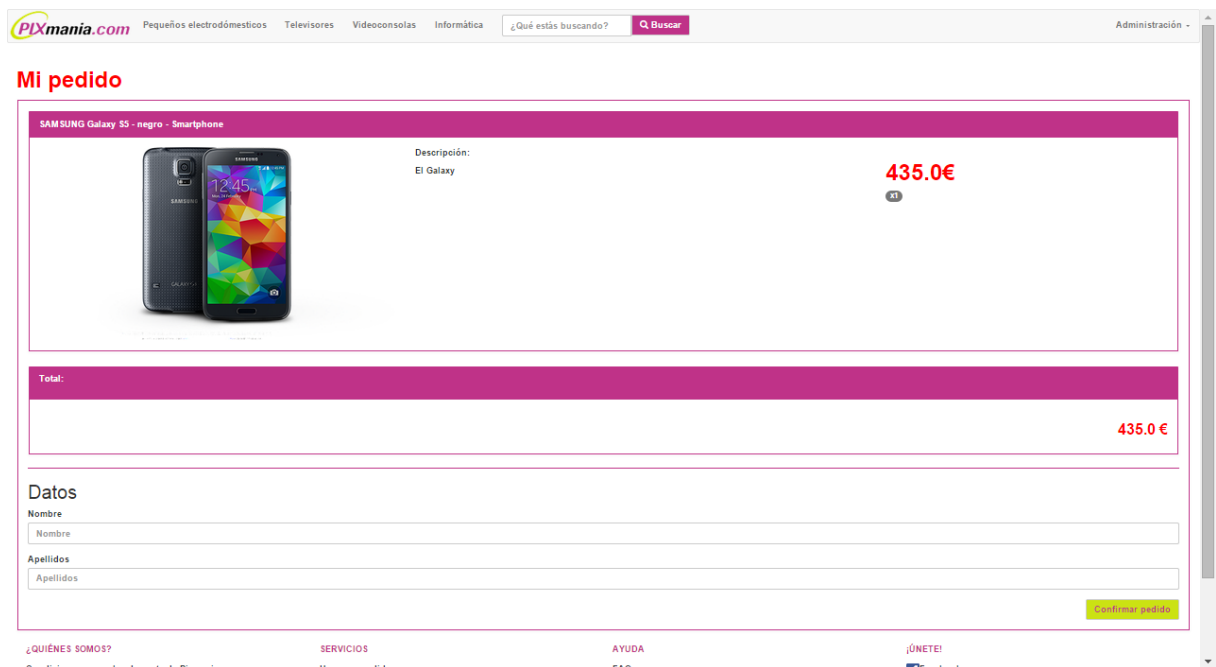


Figura 9: Vista order.html

4.5. admin.html

Esta vista (Figura 10) es la pantalla principal del administrador, en ella aparecen los distintos productos de la web, donde tenemos las opciones de añadir nuevos productos, borrar o editar productos existentes, filtrar los productos o ver los pedidos.




Figura 10: Vista admin.html

4.6. new.html

En esta vista (Figura 11) tenemos un formulario que nos permitirá añadir nuevos productos a la tienda.

4.7. edit.html

En esta vista (Figura 12) tenemos otro formulario que nos permitirá editar los productos de la tienda.

Pequeños electrodomésticosTelevisoresVideoconsolasOtras cosas

¿Qué estás buscando?

Buscar

Cerrar sesión

Nuevo producto

Nombre del producto

Nombre del producto

Categoría

Pequeños electrodomésticos

Precio

Precio

Imagen


Seleccionar archivo

Ningún archivo seleccionado

Descripción

Añadir producto

Figura 11: Vista new.html

Pequeños electrodomésticosTelevisoresVideoconsolasOtras cosas

¿Qué estás buscando?

Buscar

Cerrar sesión

Editar producto

Nombre del producto

SAMSUNG Galaxy S5 - negro - Smartphone

Categoría

Pequeños electrodomésticos

Precio

435,0

Imagen

Seleccionar archivo

Ningún archivo seleccionado

Descripción

El Galaxy

Editar producto

Figura 12: Vista edit.html

4.8. orders.html

En `orders.html` (Figura 13) se mostrarán los distintos pedidos de la tienda donde tendremos la opción de cambiar su estado (pendiente o preparado) (Figura 14). Además, se podrán filtrar los pedidos mediante su estado

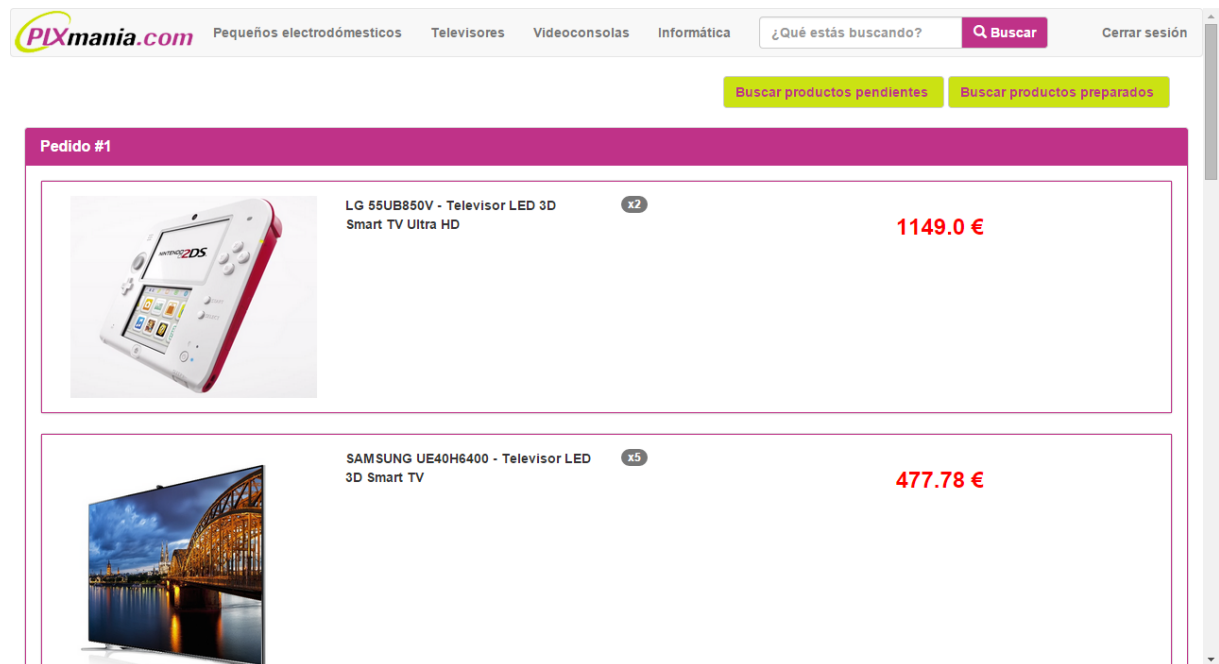


Figura 13: Vista orders.html



Figura 14: Vista orders.html (actualizar estado)