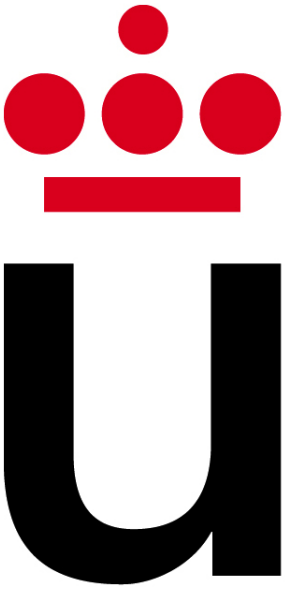
Práctica 1

Diseño de Aplicaciones Web

Implementación de una tienda virtual

GII+GIS: *Germán Alonso Azcutia*
GIS+MAT: *José Ignacio Escribano Pablos*



Móstoles, 28 de marzo de 2015

Índice

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).

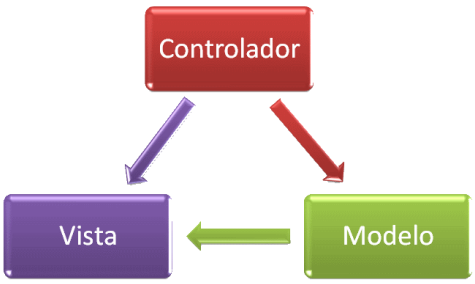


Figura 1: Patrón de arquitectura Modelo Vista Controlador (MVC)

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 carro, 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 ProductWithCuantity, Cart, Product y Order. La descripción de cada clase se puede ver en la Sección 3.

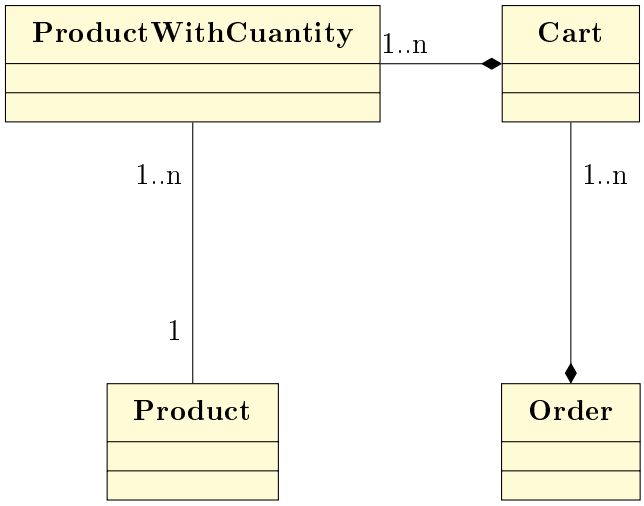


Figura 3: Diagrama de clases de la aplicación

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 adminstrador 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.

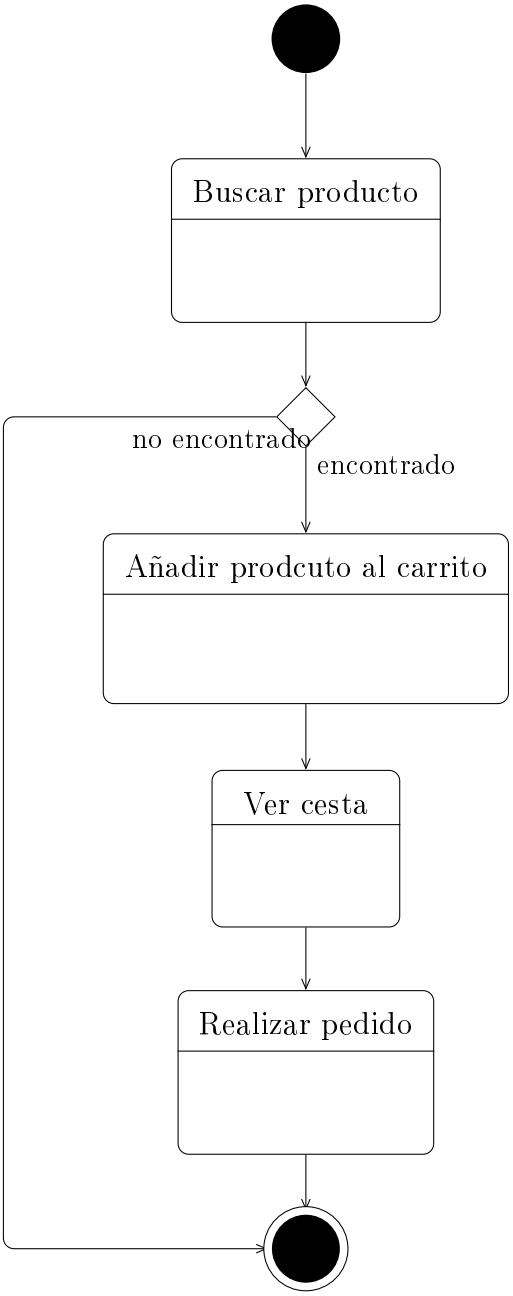


Figura 4: Diagrama de secuencia de comprar producto

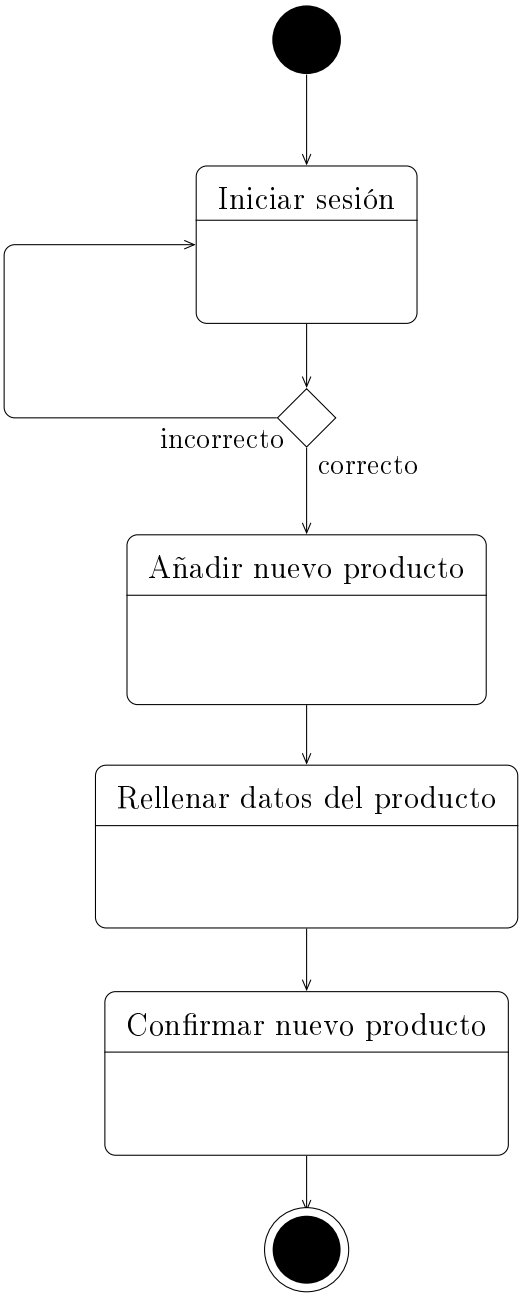


Figura 5: Diagrama de secuencia de añadir producto como administrador

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 ProductWithCuantity

La clase `ProductWithCuantity` 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 `ProductWithCuantity` (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 OrdertRepository

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 de nuestra aplicación, es decir, responderá a los eventos de las diferentes vistas (mayoritariamente acciones del usuario, como pulsar un botón) y hará peticiones al modelo cuando se hace alguna solicitud sobre la información. Un ejemplo podría ser cuando el usuario pulsa sobre la categoría "Videoconsolas", la clase `ProductController` responderá al evento buscando en la base de datos los productos pertenecientes a dicha categoría y devolviendo una vista donde aparezcan estos productos.

3.3. Vista

3.3.1. index.html

Esta vista corresponde a la página principal de nuestra aplicación web.

3.3.2. product.html

En `product.html` encontramos la vista donde se ve un producto seleccionado (botón más información), con la información completa de dicho producto y la posibilidad de añadirlo al carro.

3.3.3. cart.html

En esta vista tenemos nuestro carro de compra, donde podremos aumentar o disminuir la cantidad de los productos que se encuentren en la cesta, eliminarlos y realizar el pedido.

3.3.4. admin.html

Esta vista 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.

3.3.5. new.html

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

3.3.6. edit.html

En esta vista tenemos un formulario que nos permitirá editar los productos de la tienda.

3.3.7. order.html

Esta vista es otro formulario en donde se pedirán los datos al usuario para realizar el pedido.

3.3.8. orders.html

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

4. Recorrido por la aplicación