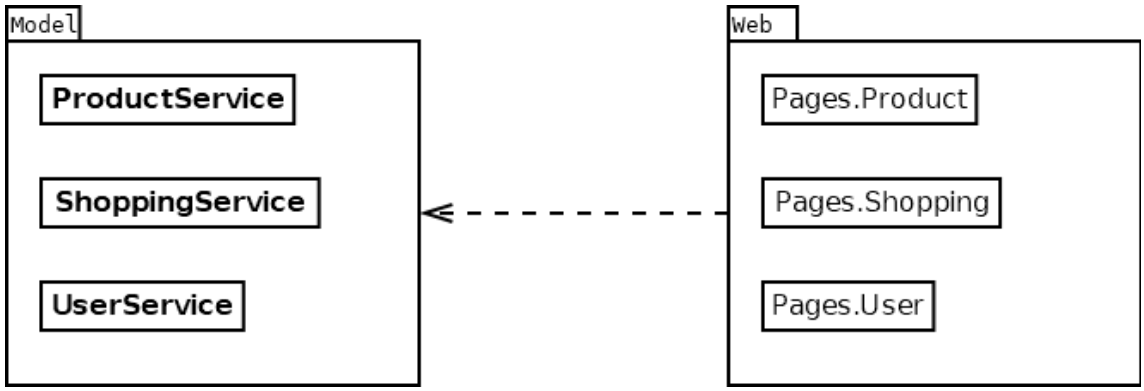
Memoria MaD

Alejandro Pereira Carballo

Sara García Senra

1. Arquitectura global

La estructura de la práctica se basa en la arquitectura en Web Forms de ASP.NET. Es un estilo de programación basado en eventos y controladores, donde distinguimos dos grandes capas: modelo y web.

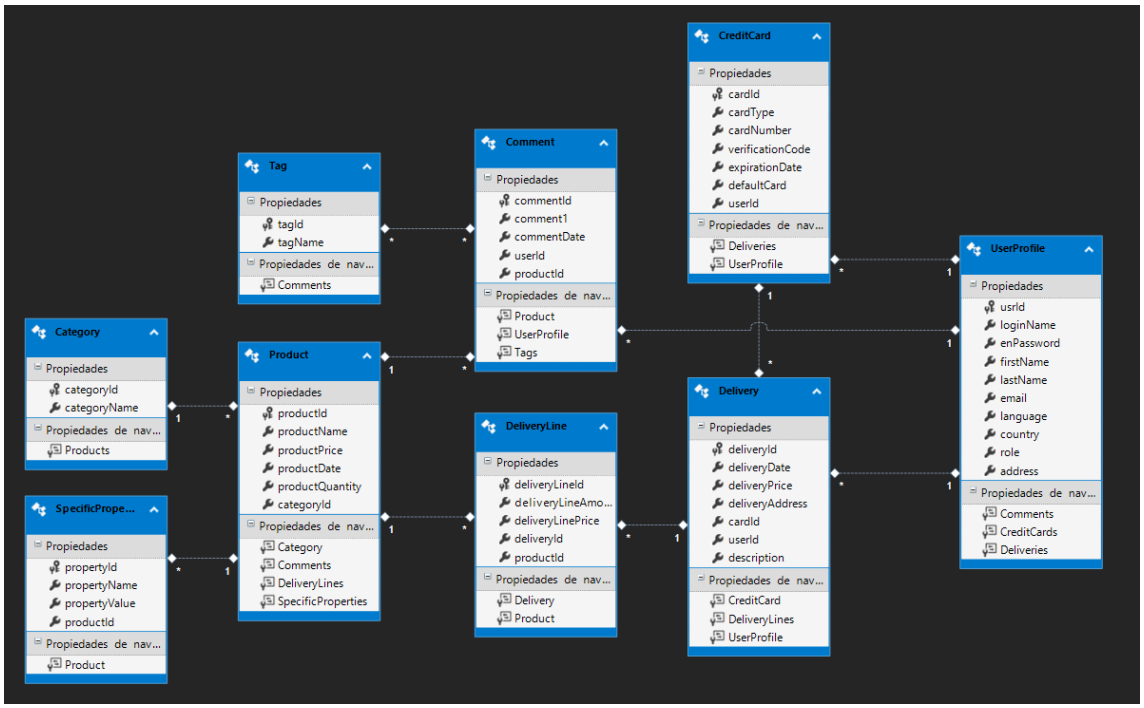


La capa modelo se encarga de manejar y convertir los datos que entran y salen de la base de datos. Ofrece tres servicios distintos para controlar las operaciones en función de la naturaleza de sus datos: “ProductService” para el manejo de datos de los productos de la tienda, “ShoppingService” para el control de compras e historiales, y “UserService” para la gestión de usuarios de la plataforma. Estos servicios, a su vez, implementan su funcionalidad mediante clases de menor nivel, los DAOs, que ofrecen comunicación directa con la base de datos.

La capa web se apoya en la capa modelo para mostrar dichos datos de manera comprensible para el usuario, mediante formularios web. Dividimos las distintas páginas de manera similar a como lo hicimos en la capa modelo: tenemos una serie de formularios dedicados a la gestión del perfil de usuario, otros que manejan los historiales de compras y las nuevas operaciones, y otros que hacen lo propio con el stock y detalles de los productos ofrecidos en la tienda.

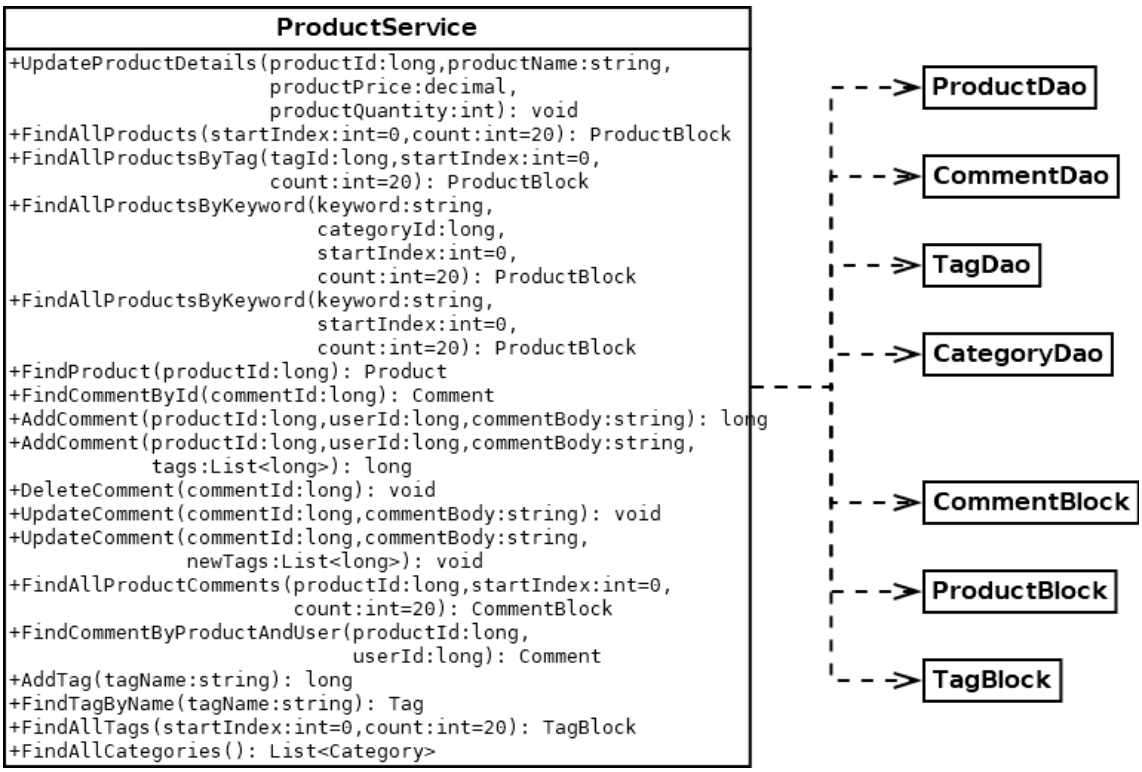
2. Modelo

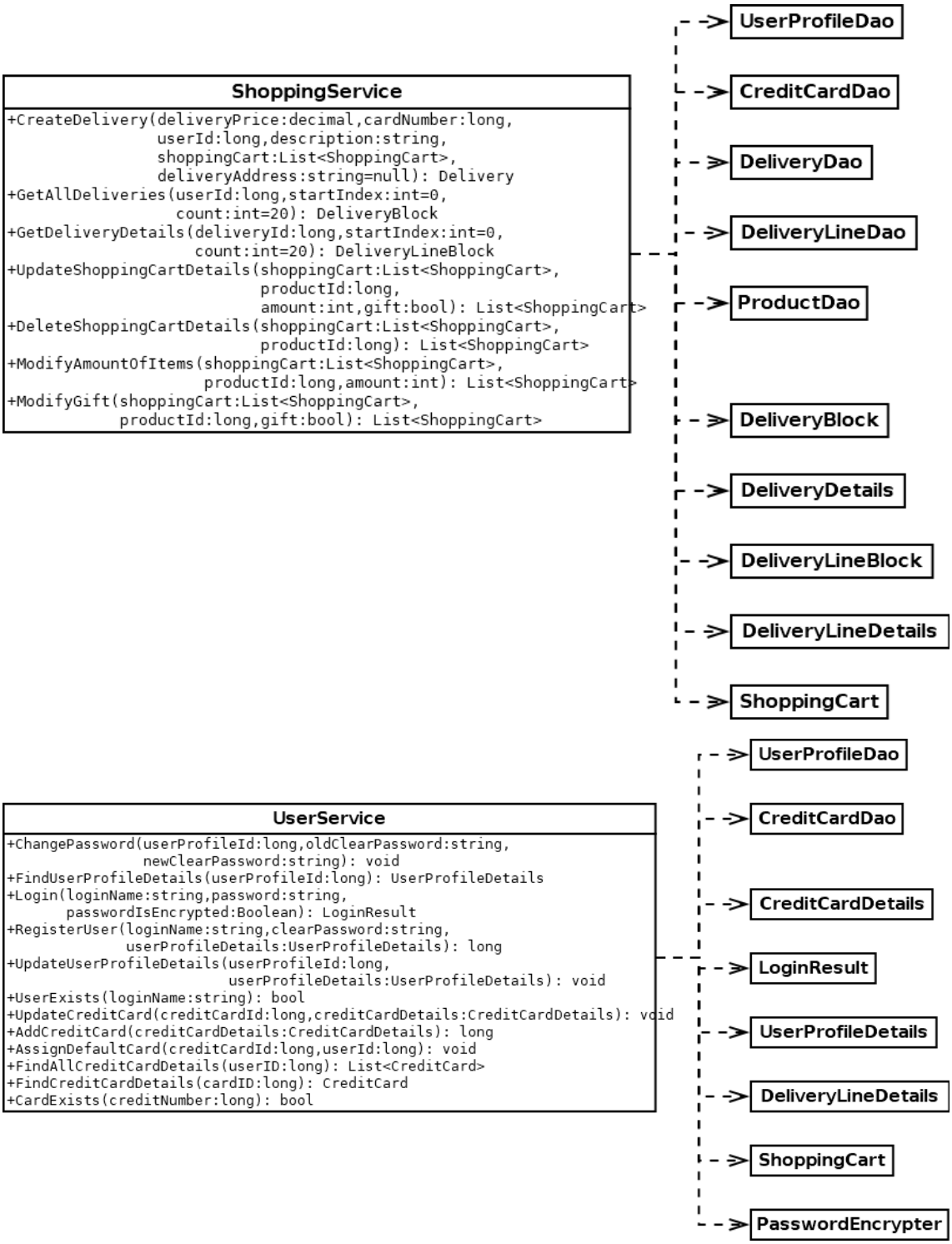
2.1. Clases persistentes



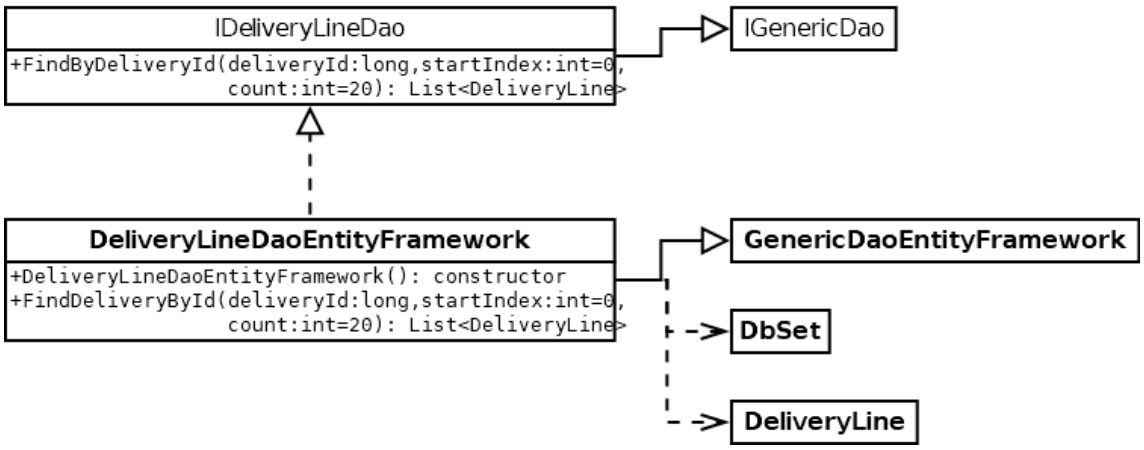
2.2. Interfaces de los servicios ofrecidos por el modelo

La agrupación de los casos de uso se desarrolló en función de la naturaleza de los datos a manejar. De esta forma, obtuvimos tres servicios: uno que maneja los datos de usuario, como nombre, dirección, tarjeta de crédito y demás; otro que gestiona la información de los productos, como el nombre, la cantidad o el precio; y otro que lleva el control de los datos de compras, como es el historial de compras, o los productos de cada pedido.

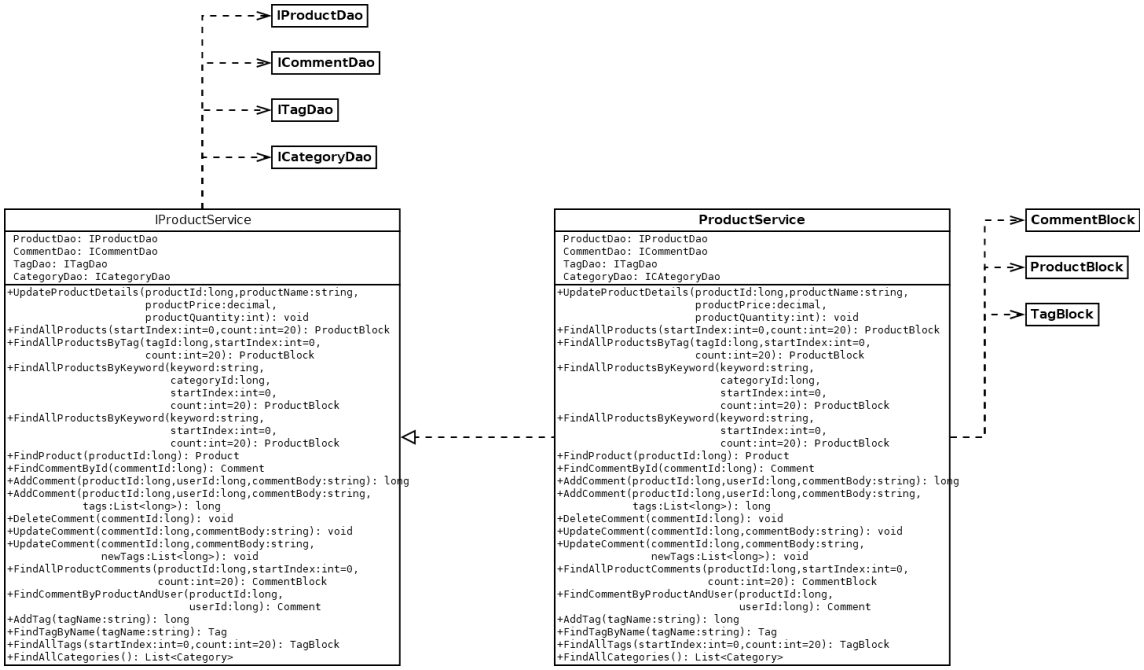


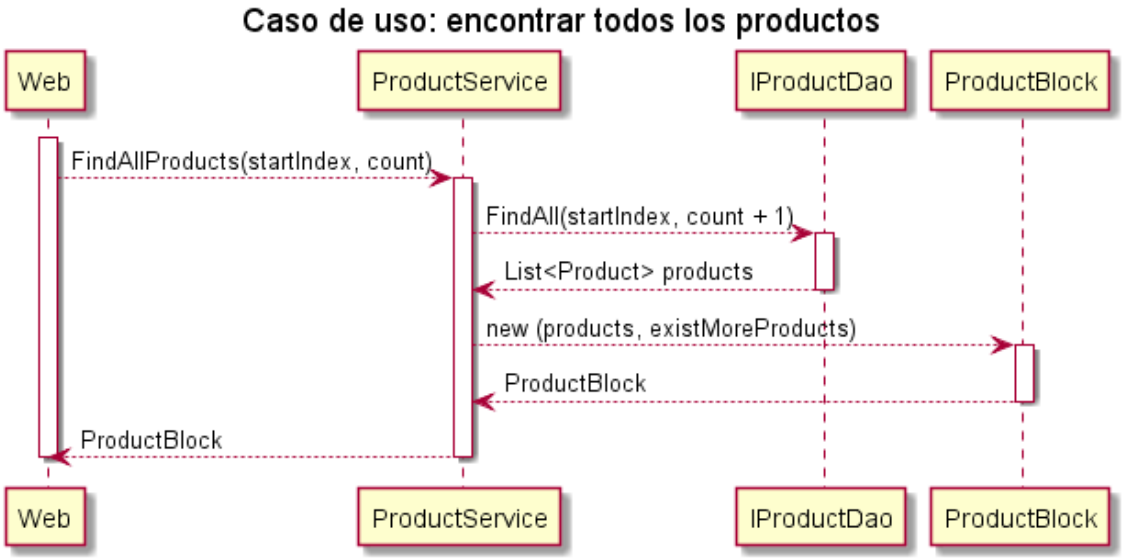

ShoppingService
+CreateDelivery(deliveryPrice:decimal,cardNumber:long,
userId:long,description:string,
shoppingCart:List<ShoppingCart>,
deliveryAddress:string=null): Delivery
+GetAllDeliveries(userId:long,startIndex:int=0,
count:int=20): DeliveryBlock
+GetDeliveryDetails(deliveryId:long,startIndex:int=0,
count:int=20): DeliveryLineBlock
+UpdateShoppingCartDetails(shoppingCart:List<ShoppingCart>,
productId:long,
amount:int,gift:bool): List<ShoppingCart>
+DeleteShoppingCartDetails(shoppingCart:List<ShoppingCart>,
productId:long): List<ShoppingCart>
+ModifyAmountOfItems(shoppingCart:List<ShoppingCart>,
productId:long,amount:int): List<ShoppingCart>
+ModifyGift(shoppingCart:List<ShoppingCart>,
productId:long,gift:bool): List<ShoppingCart>
UserProfileDao
CreditCardDao
DeliveryDao
DeliveryLineDao
ProductDao
DeliveryBlock
DeliveryDetails
DeliveryLineBlock
DeliveryLineDetails
ShoppingCart
UserService
+ChangePassword(userProfileId:long,oldClearPassword:string,
newClearPassword:string): void
+FindUserProfileDetails(userProfileId:long): UserProfileDetails
+Login(loginName:string,password:string,
passwordIsEncrypted:Boolean): LoginResult
+RegisterUser(loginName:string,clearPassword:string,
userProfileDetails:UserProfileDetails): long
+UpdateUserProfileDetails(userProfileId:long,
userProfileDetails:UserProfileDetails): void
+UserExists(loginName:string): bool
+UpdateCreditCard(creditCardId:long,creditCardDetails:CreditCardDetails): void
+AddCreditCard(creditCardDetails:CreditCardDetails): long
+AssignDefaultCard(creditCardId:long,userId:long): void
+FindAllCreditCardDetails(userID:long): List<CreditCard>
+FindCreditCardDetails(cardID:long): CreditCard
+CardExists(creditNumber:long): bool
UserProfileDao
CreditCardDao
CreditCardDetails
LoginResult
UserProfileDetails
DeliveryLineDetails
ShoppingCart
PasswordEncrypter

2.3. Diseño de un DAO

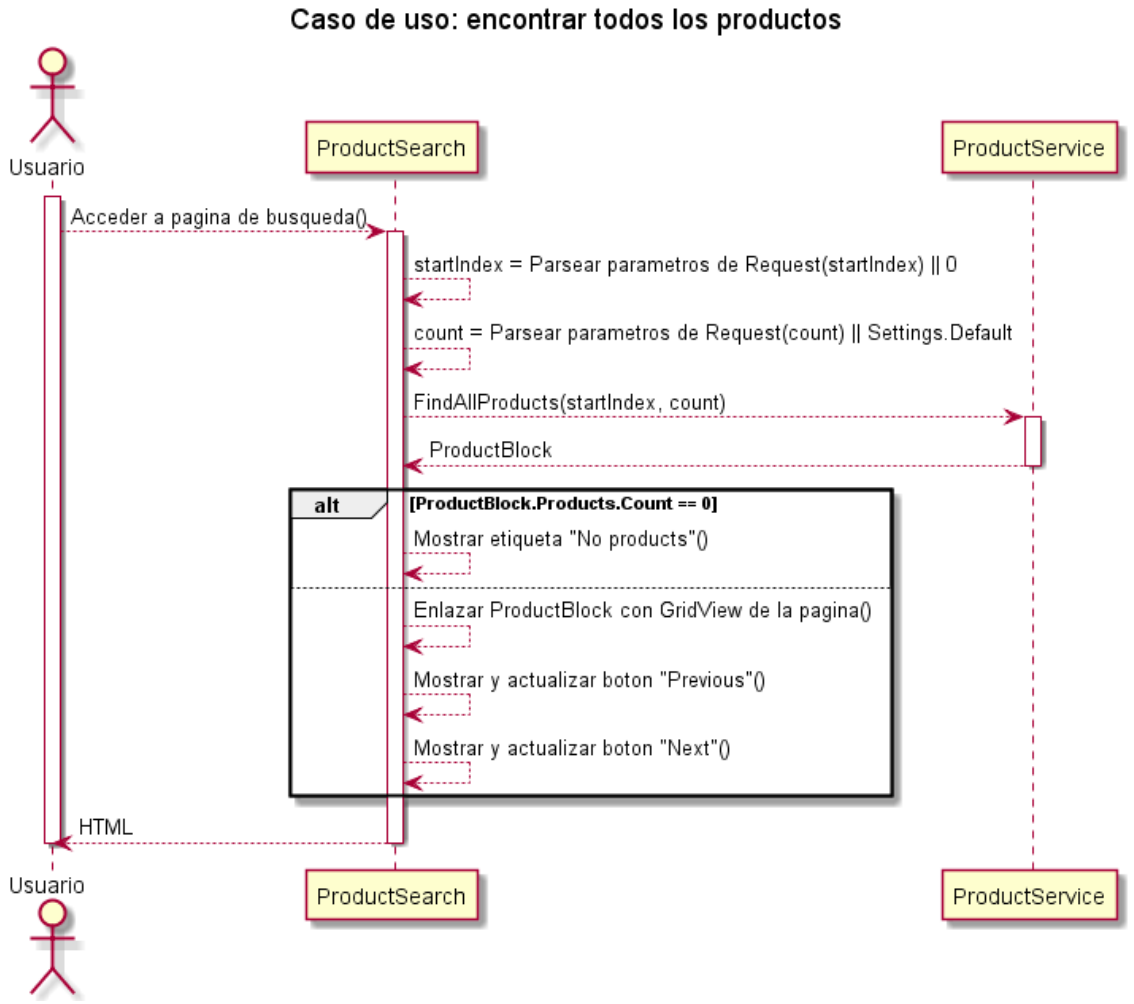


2.4. Diseño de un servicio del modelo





3. Interfaz Gráfica



4. Partes opcionales

4.1. Comentario de productos

Para permitir dejar comentarios en la página de cada producto, añadimos una entidad *Comment* a la base de datos, implementando su DAO correspondiente. Después, añadimos las funciones CRUD básicas y algunas más concretas a la clase servicio de productos.

En la capa Web, añadimos un botón en la página de detalles de producto para crear un comentario; y otro botón justo al lado para ver todos los comentarios. En la página de comentarios, aparecerá fijado el comentario hecho por el usuario logeado con un botón para modificarlo y otro para eliminarlo, si es que el usuario tiene un comentario sobre el producto. En cualquier caso, se mostrarán todos los comentarios referentes al producto, ordenados por fecha y paginados de ser necesario.

4.2. Etiquetado de comentarios

El etiquetado de comentarios requirió crear una nueva entidad en la base de datos, *Tag*, que se tiene una relación N a M con *Comment*. Creamos el DAO apropiado para la entidad, y modificamos el servicio de productos para incluir funciones de creado y búsqueda de etiquetas. También modificamos las funciones de actualización y creación de comentarios para contemplarlas.

En la capa Web, incluimos en la plantilla base una nube de etiquetas, justo debajo de la barra de navegación, que muestra todas las etiquetas creadas. Al hacer clic en cualquiera de ellas, el usuario es redirigido a la lista de productos que tienen un comentario asociado a esa etiqueta. Para añadir etiquetas, debe crearse un comentario. La vista de creación de comentarios se actualizó para admitir las etiquetas. Ahora pueden escribirse todas las etiquetas deseadas en el cuadro correspondiente, separadas por espacios. Las etiquetas no pueden contener espacios, y son convertidas a letras minúsculas antes de ser almacenadas en la base de datos. La página de modificación del comentario tiene exactamente la misma estructura.

5. Compilación e instalación de la aplicación

Para compilar la aplicación, debe hacerse clic derecho en la solución dentro de la ventana "Explorador de soluciones". Se hará clic en la opción "Propiedades" del menú contextual que se ha abierto, y se abrirá una nueva ventana. En el cuadro de la izquierda, debe hacerse clic en "Dependencias del proyecto". En el desplegable de la derecha se podrán seleccionar los proyectos incluidos dentro de la solución, y se podrán indicar sus dependencias en el cuadro justo debajo de este. Para el proyecto "Model", no debe haber ninguna dependencia seleccionada; y para "Test" y "Web", debe estar seleccionado sólo "Model". A continuación, se hará clic en el botón "Aceptar" en la parte inferior de la ventana.

Hecho esto, se debe hacer clic derecho en el proyecto "Web" dentro del explorador de soluciones, y seleccionar la opción "Publicar…". Se abrirá una pestaña en el editor con un panel de navegación a la izquierda. Aquí, se escogerá "Publicar". Si no hay un perfil creado previamente, habrá un botón "Iniciar". Al hacer clic se abrirá una ventana donde podrá seleccionarse el destino de la publicación: configurar una máquina virtual de Azure, un servidor FTP/IIS, compilar a una carpeta… En este último caso, se introducirá una carpeta de destino, y se hará clic en "Publicar". Cuando la terminal indique que ha terminado el proceso, podrá encontrarse el resultado de la compilación en la carpeta indicada.

Por último, para servir la aplicación, solo hay que copiar la carpeta recién generada en un servidor IIS o similar.