Marcos de Desarrollo

Memoria

Photogram

Miembros:

Álvaro Pérez Pedreira - alvaro.perez.pedreira@udc.es

Índice.

1- Arquitectura global.

Nuestra aplicación se compone de tres paquetes principales, los cuales son Model (el modelo), Test (el entorno de pruebas) y Web (la aplicación ASP.NET).

El paquete Model surge de la creación de un proyecto .NET, el paquete Test surge de un proyecto de test unitarios y el paquete Web surge de la creación de un proyecto Web vacío.

Los tres paquetes están conectados entre sí, para cumplir con el funcionamiento y propósito general de la aplicación.

En el paquete Model se ha implementado el EDM (Entity Data Model) representando las entidades de la base de datos (DB First) y se ha implementado toda la lógica de los servicios junto con sus respectivos DAOs.

En el paquete Test se ha implementado todas las pruebas unitarias de los casos de uso implementados en la práctica.

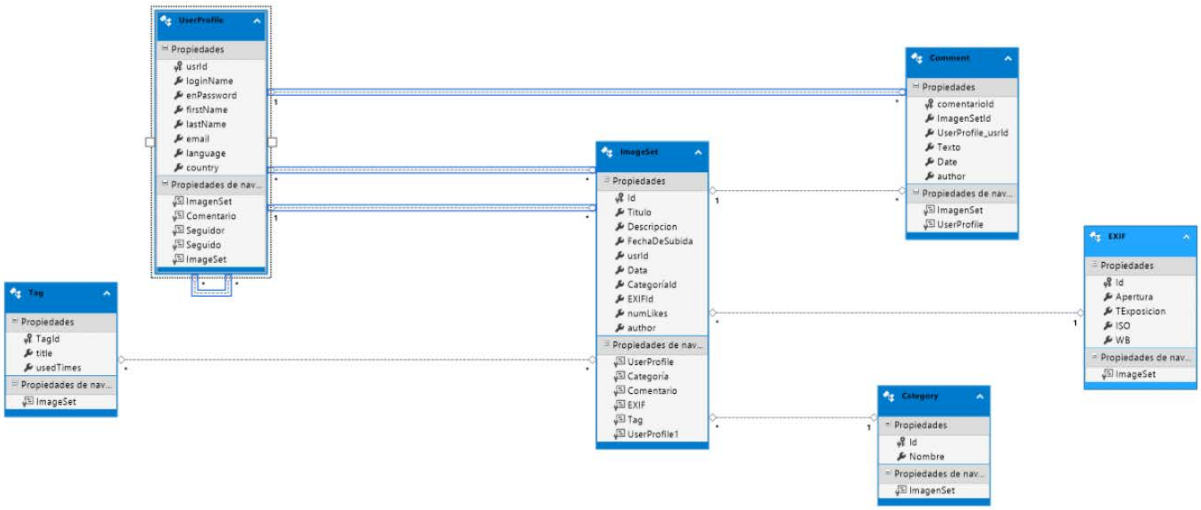
En el paquete Web se han implementado todas las páginas (Web Forms) que conforman el frontend de la aplicación. En este paquete también se han incluido todos los ficheros necesarios para que la aplicación cuente con un sistema de internacionalización, además también cuenta con un sistema de login con Google.

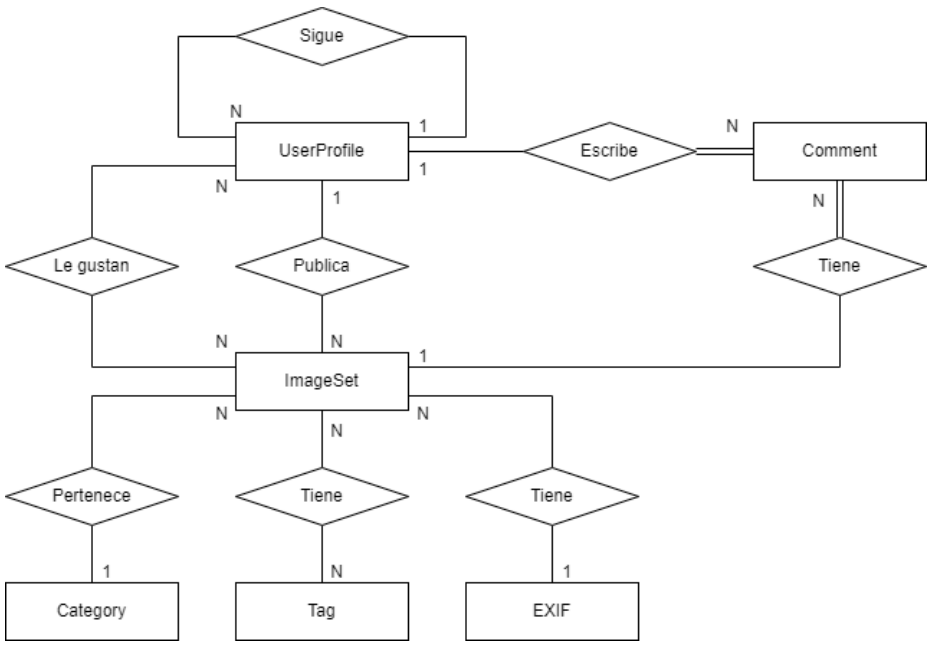
2- Modelo.

En el modelo, como se ha mencionado anteriormente, se ha implementado el EDM (Entity Data Model).

2.1- Clases persistentes.

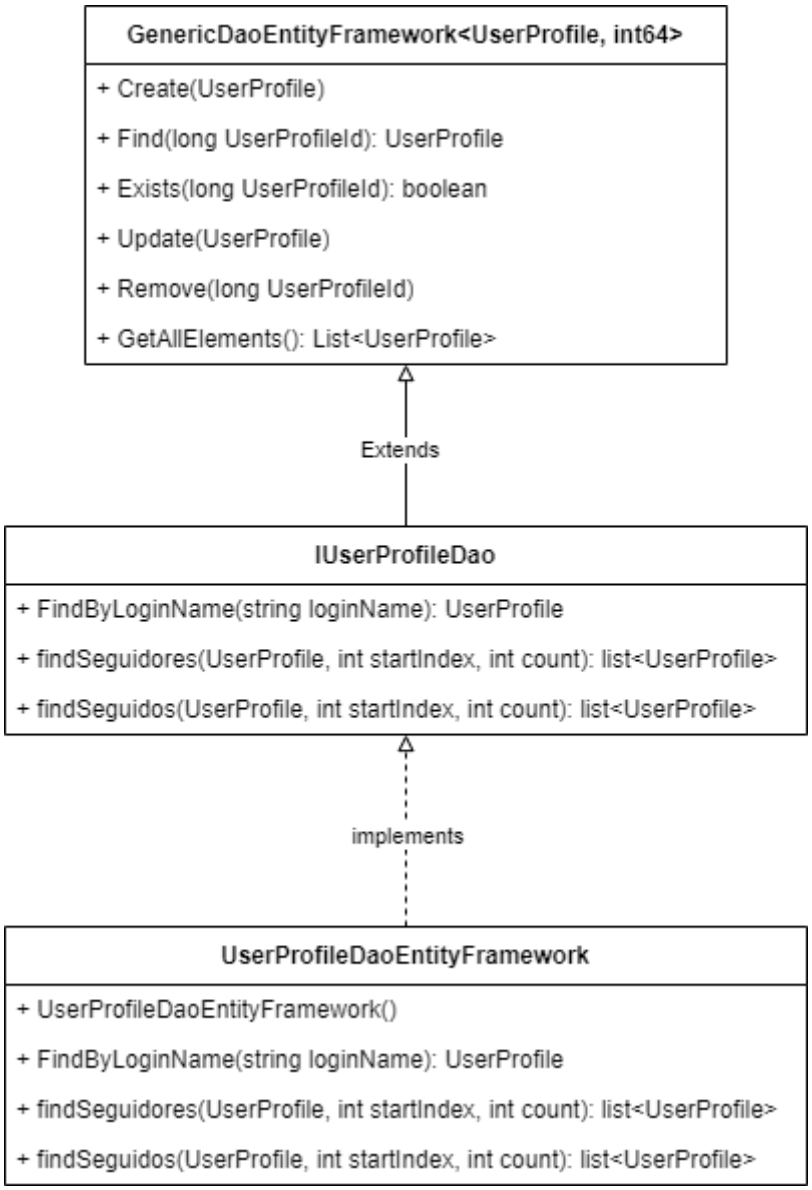
A continuación, se mostrará el .edmx junto con otro diagrama:





2.2- Diseño de un DAO.

Los DAOs se han implementado en base a un esquema general. Por ejemplo, para crear el DAO de usuarios hemos creado una carpeta dentro del directorio Model con el nombre de UserProfileDao, dentro de esta carpeta se ubican la interfaz del DAO y su Entity Framework (IUserProfileDao y IUserProfileDaoEntityFramework). El esquema que se muestra a continuación es igual para el resto de DAOs.



2.3- Interfaces de os servicios proporcionados por el modelo.

En nuestro proyecto, se han hecho dos agrupaciones para los casos de uso, ImageService y UserService. En UserService están todos los casos de uso que tienen que ver con el usuario, tales como autenticarse, registrarse, gestionar el perfil…

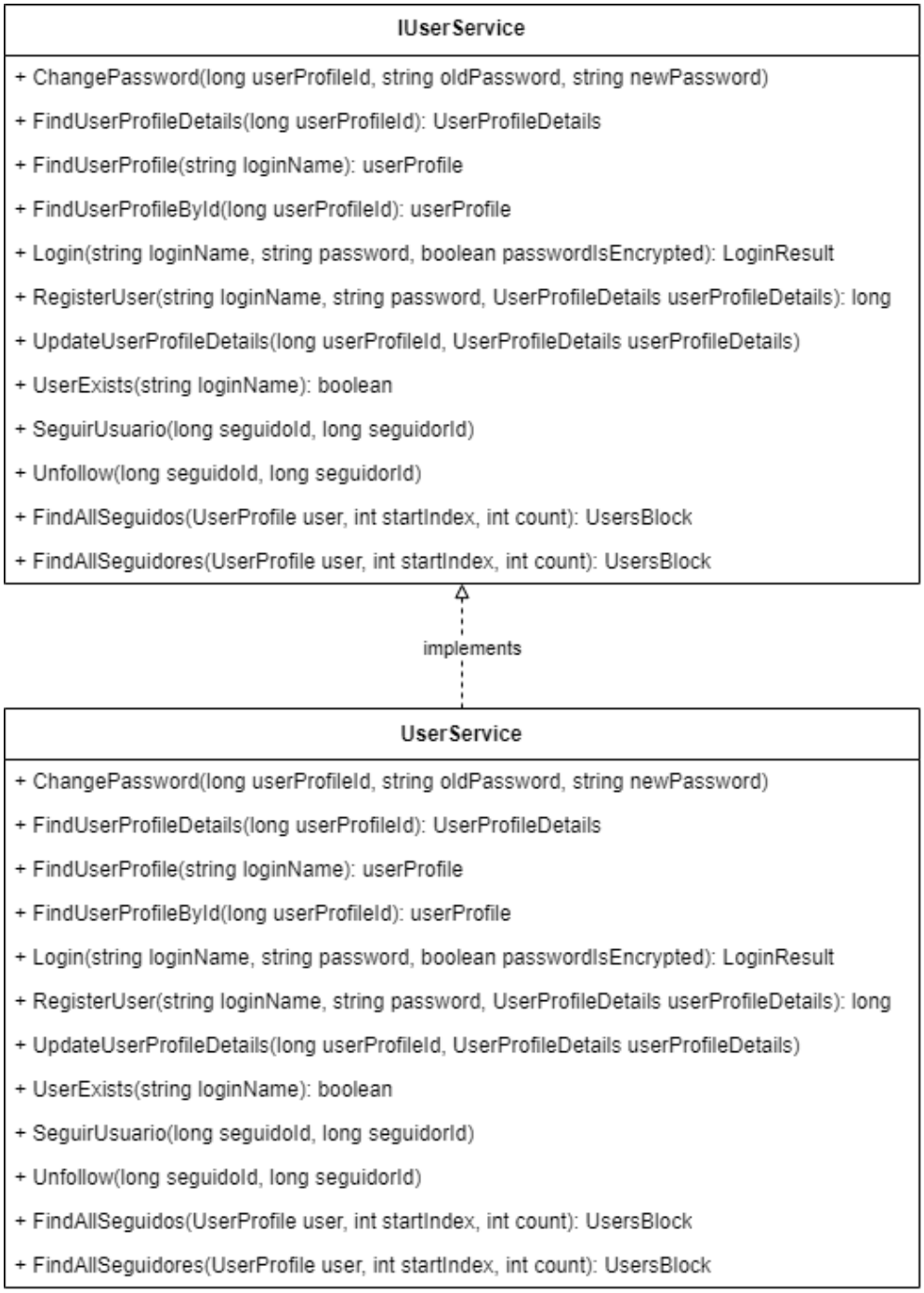
Por el otro lado tenemos ImageService, que agrupa todos los casos de uso de gestión de imágenes y elementos relacionados con ellas, como toda la gestión de tags, comentarios y likes.

2.4- Diseño de un servicio del modelo.

Para los servicios del modelo también se ha seguido el mismo patrón para todos. Se ha creado una carpeta con el formato xxxService en el directorio raíz del modelo y dentro de esta carpeta se ha creado una interfaz que contenga todos los casos de uso a implementar por el servicio y la clase que los implementa.

En caso de que se necesiten clases de Blocks o cualquier otra clase/funcionalidad que esté directamente relacionada con este servicio, también se incorporarán en esta carpeta.

Dentro de esta carpeta también se ha creado una carpeta con el nombre de Exception, en el que estarán todas las excepciones que se han encontrado en la ejecución del servicio.



2.5- Otros aspectos.

Como se ha mencionado anteriormente se han añadido en algunos servicios las clases necesarias para poder paginar (como, por ejemplo, las imágenes). Esto es porque si existe un número muy elevado de imágenes en nuestra aplicación, no queremos que un usuario las cargue todas de golpe cuando quiera visualizarlas en nuestra aplicación.

También se ha añadido una caché a nivel usuario para recuperar las imágenes de los perfiles de usuario.

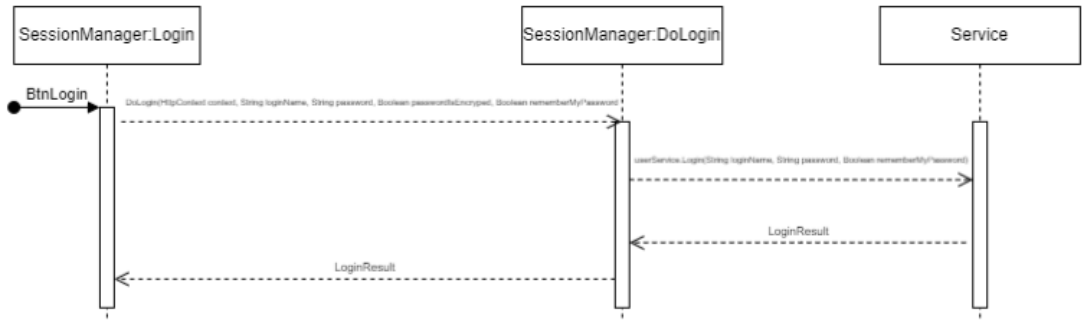
3- Test.

En el paquete del proyecto UnitTest se han realizado las pruebas y test de la aplicación. También se han añadido las clases necesarias para ejecutar algunos tests que prueban el correcto funcionamiento de los casos de usos implementados en el modelo.

Para realizar estos tests no hemos utilizado ninguno de los plugins mencionados en las diapositivas (intelliTest por ejemplo o la extensión Unit Test Generator).

4- Interfaz gráfica.

Se ha decidido mostrar el caso de uso de autenticarse en la aplicación. En la interfaz de usuario tenemos que pulsar en el botón de Login, este método ejecutará una función del SessionManager, que a su vez llama a otro método del archivo. Este último método hace una llamada a nuestro backend para realizar la funcionalidad.



En la primera línea pone DoLogin(HttpContext context, String loginName, String password, Boolean passwordIsEncryped, Boolean rememberMyPassword).

En la segunda línea pone userService.Login(String loginName, String password, Boolean rememberMyPassword).

5- Apartados optativos.

5.1- Etiquetado de imágenes.

Los usuarios podrán añadir tags a las imágenes en Photogram. Los tags se guardarán en la base de datos (siempre y cuando sean tags con distinto nombre y que tengan al menos una publicación asociada).

A la hora de añadir una imagen, el sistema comprueba si es un tag existente, y en caso de no serlo, creará uno nuevo.

El sistema además también llevará la cuenta de cuantas veces el tag esta siendo usado en las distintas publicaciones de la aplicación, también el usuario podrá añadir o eliminar tags a las imágenes que haya publicado en la aplicación.

5.2- Cacheado de búsquedas.

Se ha añadido una caché a nivel usuario para las búsquedas de las imágenes de los distintos perfiles de usuario, de modo que, si un usuario realiza la misma búsqueda más de una vez, no se forzará un acceso a la base de datos. Para implementar esto, se ha creado una carpeta Caché dentro del paquete Model y ahí es donde se ha implementado su lógica.

Entonces, una vez implementada la lógica, el funcionamiento sería que, a la hora de hacer una búsqueda de un perfil de usuario, el sistema comprobará si están las imágenes de ese usuario en caché (en este caso, el primer bloque de 3 imágenes de ese usuario), y si no están se accederá a base de datos.

Las imágenes en la cache se guardan en bloques de 3 (en un elemento CacheItem), que es lo que se muestra en la página donde se muestran los distintos perfiles de usuario, por lo que la caché también funciona cuando quieras pasar al siguiente bloque en el perfil del usuario que estes consultando (en caso de que este cacheado, sino accederá a la base de datos).

5.3- OAuth 2.0 y Bootstrap.

Se ha implementado un sistema de autenticación con Google usando Google API OAuth2.v2. Esto lo que significa es que un usuario podrá registrarse e iniciar sesión directamente con Google y acceder a todas las funcionalidades que tendría un usuario registrado normal.

A pesar de haber instalado Bootstrap y haberlo usado para la Card de las imágenes, se ha preferido optar por usar CSS nativo y a nuestro gusto para darle estilo al Frontend.

6- Compilación e instalación de la aplicación.

El primer paso para que la aplicación funcione sería ejecutar todos los Scripts SQL para generar la base de datos, crear sus tablas y añadir algunos campos, como categorías por defecto que necesitará un usuario para poder subir una imagen a la aplicación. Estos Scripts se encuentran en el paquete Model.

Una vez ejecutados los Scripts, tenemos que ejecutar el Script SQL del paquete de UnitTest, una vez se ha hecho esto se podrán ejecutar todas las pruebas del paquete UnitTest para comprobar que los casos de uso estén implementados correctamente. Para ello, primero se tendrá que compilar el proyecto y luego haciendo click derecho en el paquete de UnitTest, elegir la opción de ejecutar pruebas.

Para la instalación del proyecto Web simplemente hay que hacer click derecho y elegir la opción de “establecer como proyecto de inicio”, y en la parte superior en el botón de “play” aparecerá como ejecutable nuestro proyecto Web. Una vez pulsado ese botón se lanzará nuestro proyecto y seremos redirigidos a la página principal del mismo en nuestro navegador predeterminado.

7- Compilación e instalación de la aplicación.

Por algún motivo que no conseguí resolver, los links de “next” y “previous” de algunas páginas desaparecen si se tiene un zoom muy amplio, en 75% debería verse todo correctamente.