Memoria Práctica MAD: Photogram

Ramiro Serantes Garrido

Diego Fernández Torreira

Joaquín Paradelo Pérez

Índice

1. Arquitectura Global

En nuestra solución tenemos separada la parte visual (frontend, capa de vista Web) con el proyecto web, y la parte del modelo (backend) con el proyecto Model. Además, para realizar las pruebas del backend tenemos el proyecto Test.

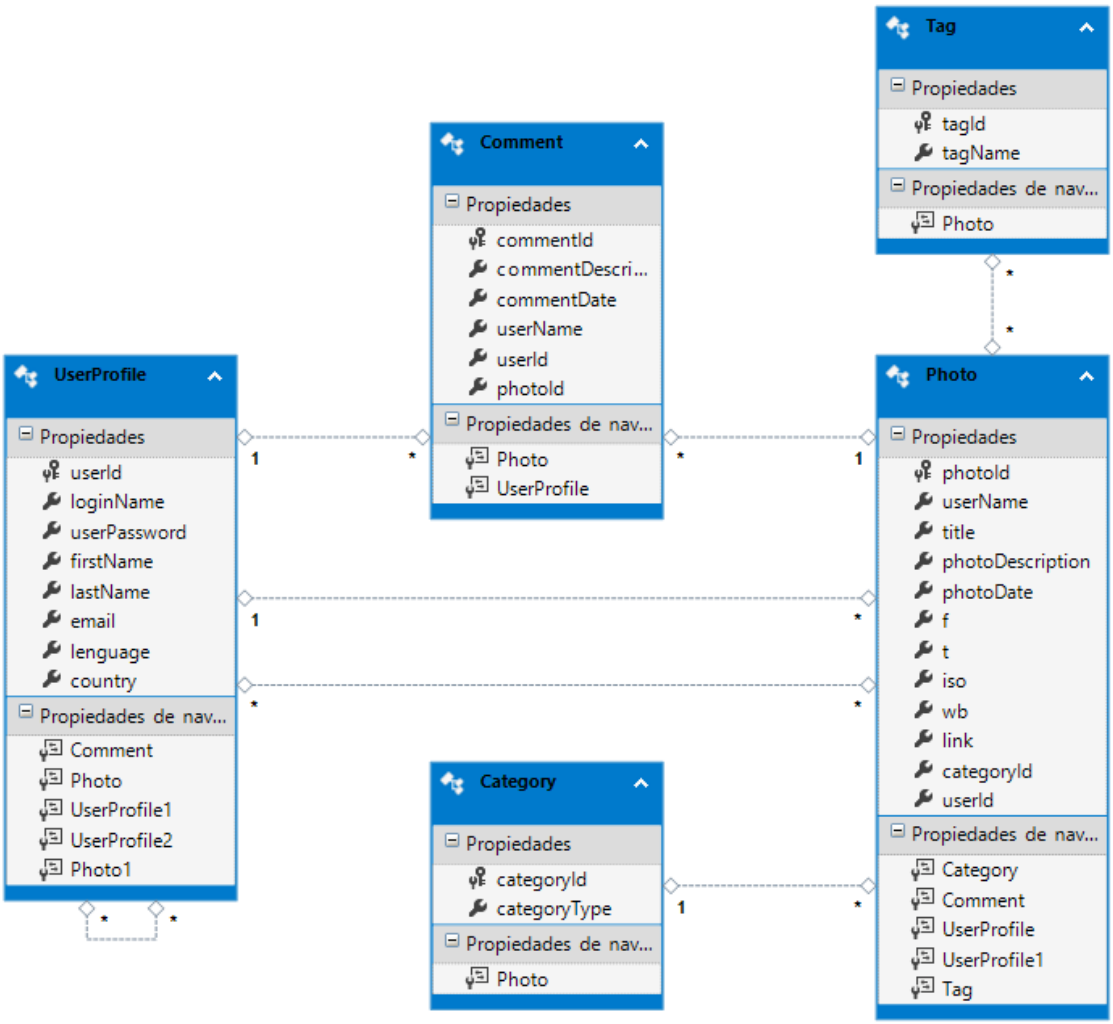
Dentro de nuestro proyecto Model tenemos la capa servicios, (carpeta *services* donde tenemos agrupados los casos de uso y todo lo relacionado con la lógica de negocio) dividido en tres servicios. Las consultas a base de datos las delegamos en los DAOs (carpeta *Daos*), haciendo uso de Entity Framework. También tenemos la carpeta *sql,* donde tenemos los archivos para crear la base de datos.

La capa Web (proyecto web) tiene la carpeta *Pages*, donde tenemos todas las páginas con las que el usuario va a interactuar. Aquí también hemos estructurado las carpetas de forma similar al modelo, separando Photo de User. Usaremos los formularios apoyándonos en la capa modelo para ejecutar los casos de uso.

En el proyecto *Test* usamos una base de datos distinta.

2. Modelo

2.1 Clases Persistentes



El EDMX ya nos muestra de forma visual las entidades y sus relaciones de navegación. Las clases son generadas automáticamente por Entity Framework a partir de este modelo conceptual.

2.2 Interfaces de los servicios ofrecidos por el modelo

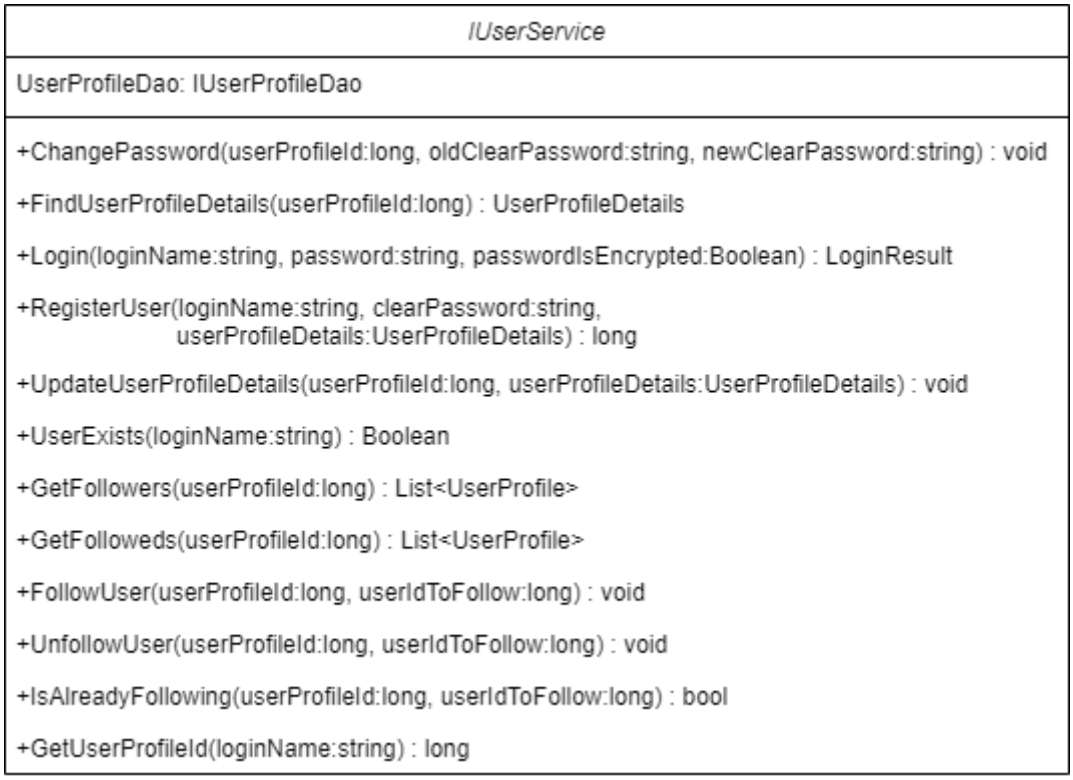
Decidimos agrupar los casos de uso que tuvieran relación lógica entre ellos. Así, desarrollamos tres servicios: CommentService, PhotoService y UserService.

UserService está centrado en todo lo relacionado con el usuario: crear usuario, inicio de sesión, etc.

PhotoService se centra en las fotos: subir foto, encontrar fotos, obtener likes en fotos, etc. Además, dentro tenemos una sección para los tags en las fotos.

CommentService se enfoca en todo lo relacionado con los comentarios en una foto: añadir, eliminar, etc. Decidimos separarlo de PhotoService por que puede ser reusado en otros servicios.

A continuación se muestra los diagramas de clases de cada interfaz de servicio.

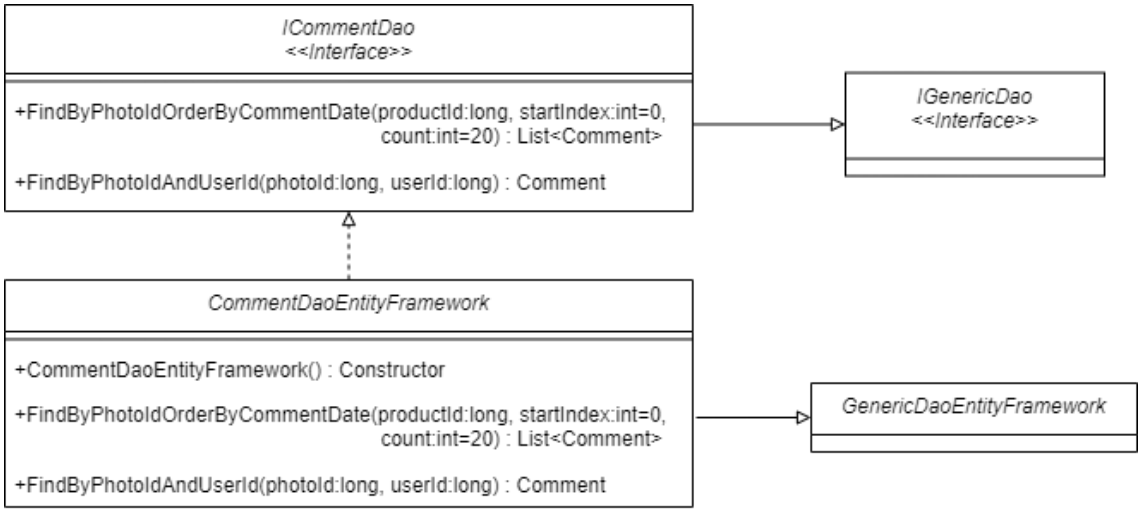



IPhotoService
UserProfileDao: IUserProfileDao
PhotoDao: IPhotoDao
CommentDao: ICommentDao
TagDao: ITagDao
CategoryDao: ICategoryDao
+UpdatePhotoDetails(photoId:long, title:string, photoDescription:string) : void
+FindAllPhotos(startIndex:int=0, count:int=20) : PhotoBlock
+FindAllPhotosByTag(tagId:long, startIndex:int=0, count:int=20) : PhotoBlock
+FindAllPhotosByUser(userId:long, startIndex:int=0, count:int=20) : PhotoBlock
+FindAllPhotosByCategoryAndKeyword(keyword:string, categoryId:long, startIndex:int=0, count:int=20) : PhotoBlock
+FindAllPhotosByKeyword(keyword:string, startIndex:int=0, count:int=20) : PhotoBlock
+FindPhoto(photoId:long) : Photo
+GenerateLike(userProfileId:long, photoId:long) : void
+DeleteLike(userProfileId:long, photoId:long) : void
+UploadPhoto(userName:string, title:string, description:string, f:long, t:long, iso:string, wb:long, categoryId:long, userId:long, newImage:Image) : long
+DeletePhoto(photoId:long, userId:long) : void
+AddTag(tagName:string, userId:long) : long
+FindTagByName(tagName:string) : Tag
+FindAllTags(startIndex:int=0, count:int=20) : TagBlock
+FindAllCategories() : List<Category>
+GetPhotoLikes(photoId:long) : int
+GetUsedTag(TagId:long) : int
+FindByPhotoTags(photoId:long) : List<Tag>
+DeleteTag(tagId:long, photoId:long) : void

2.3 Diseño de un DAO

Diagrama de clases de CommentDao.

IGenericDao y *GenericDaoEntityFramework* fueron implementados en ModelUtil.



2.4 Diseño de un servicio del modelo

Diagrama de clases de la implementación del servicio User

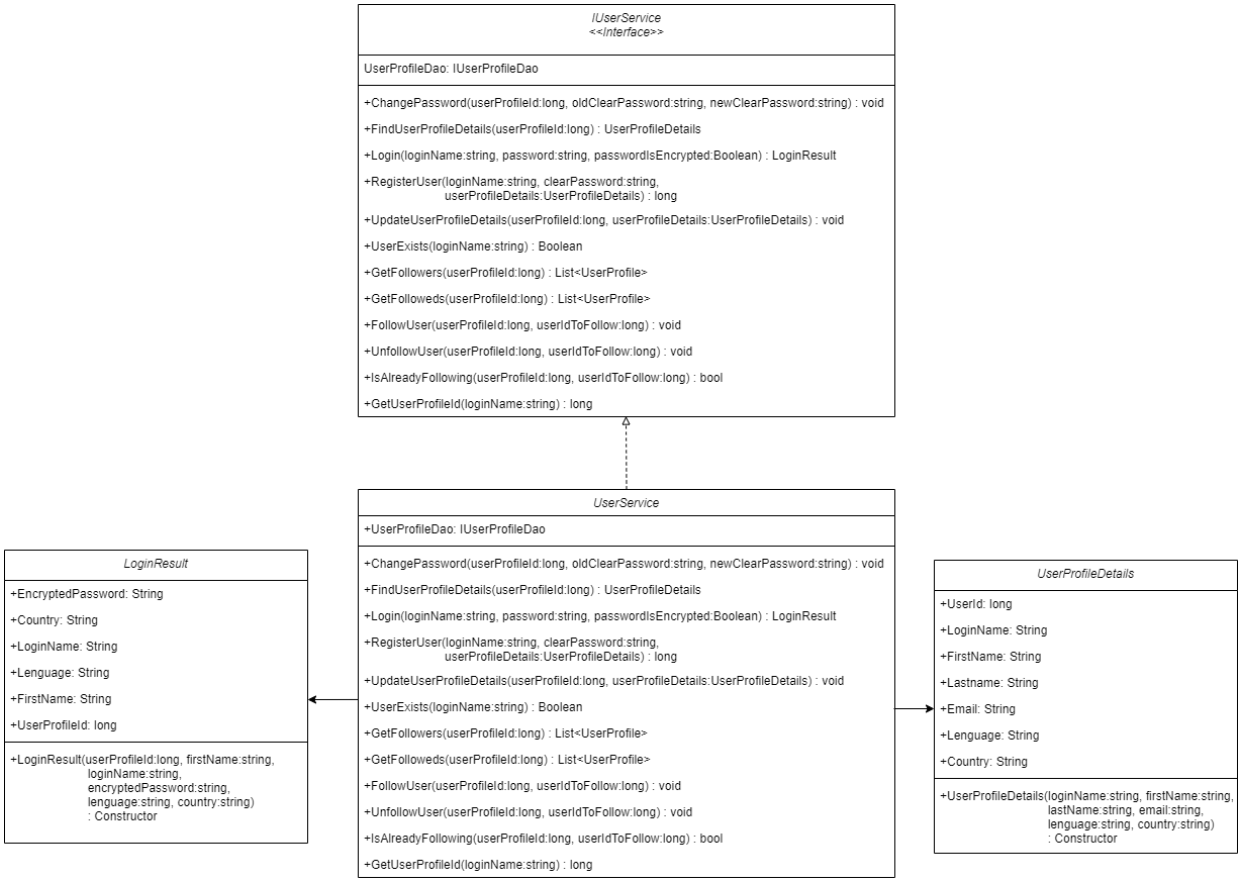
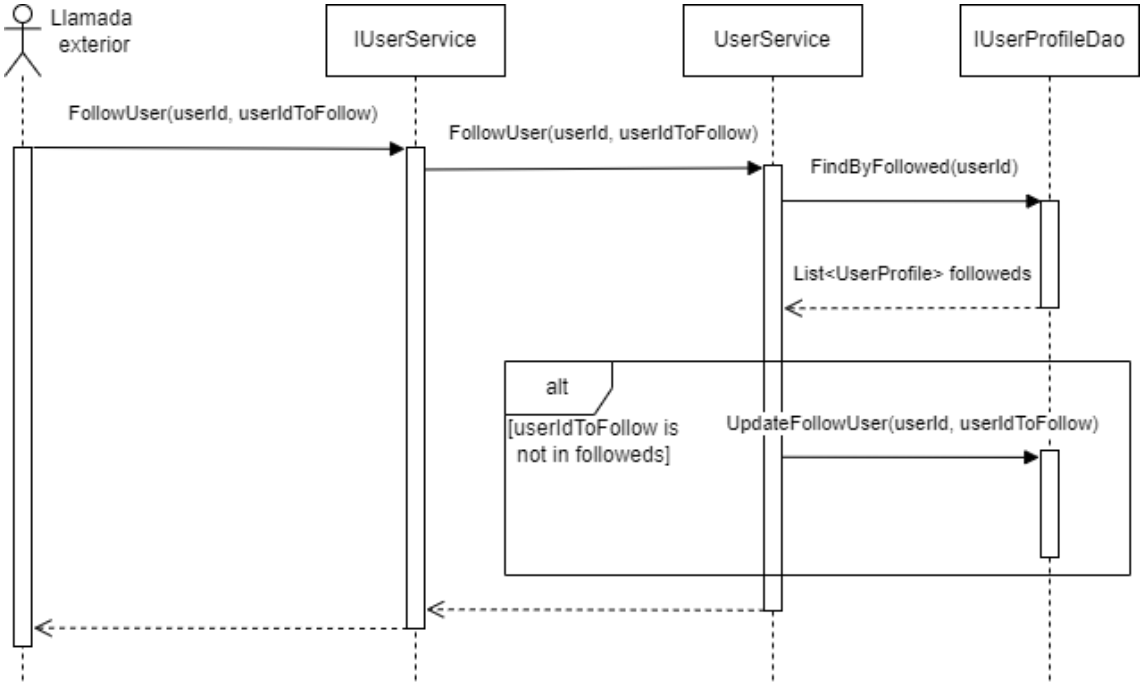
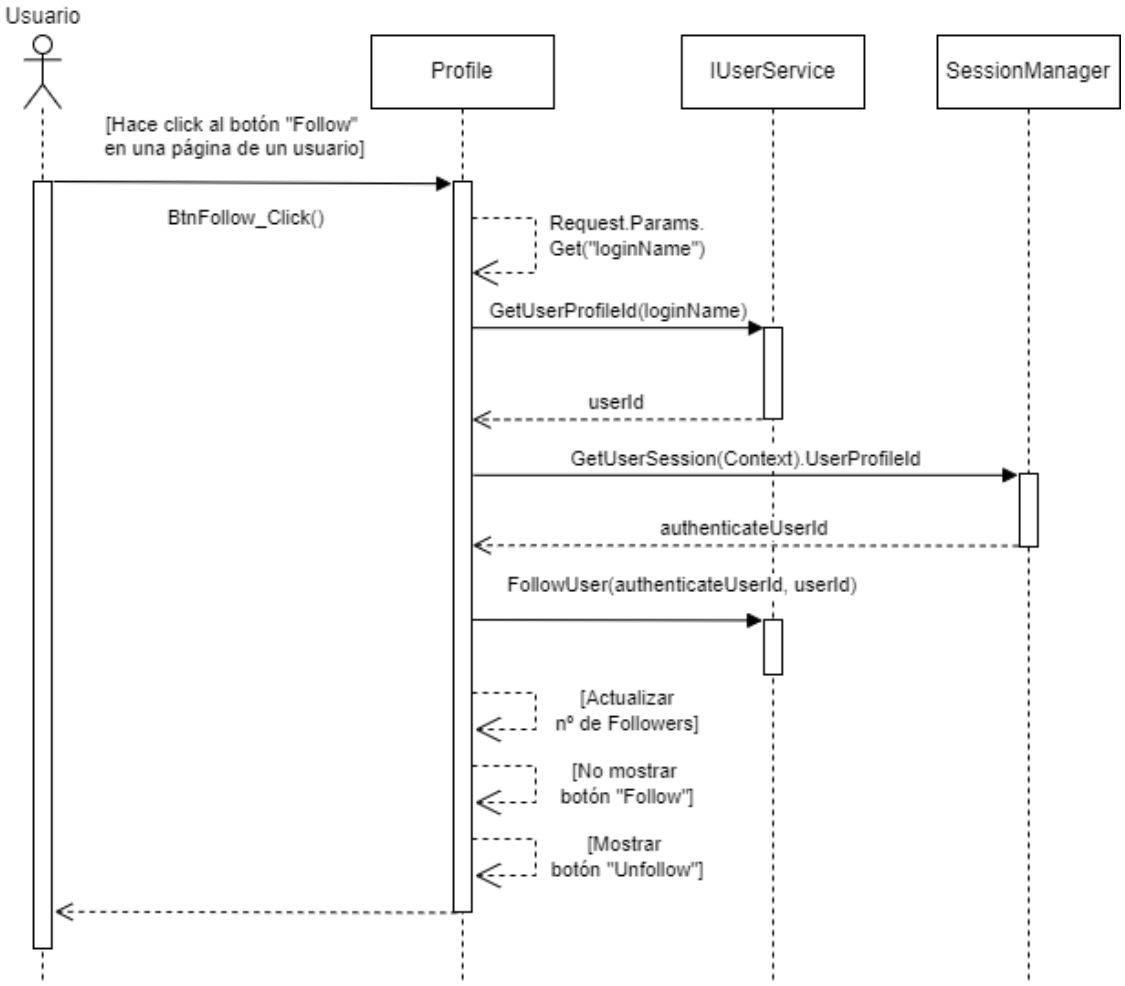


Diagrama de secuencia del Caso de Uso *FollowUser()*



3. Interfaz Gráfica

Como ya se mostró el caso de uso de FollowUser() de la capa modelo, el diagrama siguiente corresponde con la interacción del usuario cuando sigue a un usuario en la Web (le da click al botón de Follow). El usuario ya se sitúa en la página Profile.aspx de un usuario cualquiera (load_page() ya cargó) y está autenticado.



4. Funcionalidad adicional: Etiquetado de imágenes

El etiquetado de fotos requirió crear una nueva entidad en la base de datos *Tag*, que tiene una relación N:M con Photo. Creamos el DAO apropiado para la entidad y modificamos *PhotoService* para incluir funciones de creado y búsqueda de etiquetas.

En la capa web, incluímos en la plantilla base una nube de etiquetas, justo en el mismo footer de la página, que muestra todas las etiquetas creadas. Las más usadas aparecen de mayor tamaño. Al hacer click en cualquiera de ellas el usuario es redirigido a la lista de fotos que tienen esa etiqueta asociada.

Se puede añadir una etiqueta a la hora de subir una foto y posteriormente en los detalles de la foto, modificar las etiquetas una por una a gusto.

5. Compilación e instalación de la aplicación

Para compilar la solución se puede hacer desde ventana de comandos o desde el propio VisualStudio.

Para desplegar el proyecto para su uso público haremos los siguientes pasos:

1. Iremos al explorador de soluciones > botón derecho en el proyecto *Web* y Publicar.
2. Aquí podemos seleccionar el destino (una carpeta compartida que tendremos que subir a un servidor, por ejemplo *servidor web IIS*) o mismamente exportarlo a un servidor.
3. Tenemos que tener en cuenta que el proyecto Web y el proyecto Test dependen de Model. Esto se puede comprobar en propiedades de la solución > dependencias del proyecto.

6. Problemas conocidos

En la exploración de fotos se puede seleccionar la categoría en el desplegable de “categorías”. La id de categoría (categoryId) la consigue, pero no la tiene en cuenta en la búsqueda.