Mariana De la Vega Martínez

Proyecto Final: minifacebook

El proyecto final desarrollado consiste en una aplicación de escritorio que simula la mínima funcionalidad de Facebook. A continuación se listan las especificaciones de dicho desarrollo:

- ❏ UI creada con Windows Forms en Visual Studio
- ❏ Lenguaje de Desarrollo: C#
- ❏ Base de Datos: SQL (Microsoft SQL Server)

1. Instalación y configuración del sistema

Para instalar la aplicación de escritorio en cualquier computadora se tiene que:

1. Clonar el repositorio de Github que se encuentra en esta liga: https://github.com/0206627/ProyectoPatrones.git

2. Después de descargarlo, abrir el archivo de solución de Visual Studio.
    a. En los archivos existe una carpeta llamada *images*. Ahí se encuentran todas las imágenes que se utilizan como base en el proyecto.
    b. Para que estas imágenes puedan ser utilizadas se necesita modificar el atributo *imageRoute* en la clase estática *Configuration* (en el archivo *Configuration.cs* dentro del proyecto de Windows Forms) para que haga referencia a la carpeta *images* mencionada en el inciso anterior.
    c. Para que las referencias funcionen, debido a conflictos entre proyectos de tipo net y core, se tienen que realizar varios pasos:
        i. Compilar los proyectos *Core* y *DAL* de la solución.
        ii. En Visual Studio, en el proyecto *MiniFacebookVisual* dar click derecho → Agregar → Referencia. Cuando aparezca la nueva ventana, dar click en el botón “Examinar”:
            1. Ir a la carpeta donde se guardó el repositorio, ir a la carpeta del proyecto *Core* → bin → Debug → net472 → *Core.dll*
            2. Volver a dar click en el botón de “Examinar” e ir a la carpeta donde se guardó el repositorio, ir a la carpeta del proyecto *DAL*→ bin → Debug → net472 → *DAL.dll*
        iii. Click en “Aceptar” para agregar las referencias.

3. Para la base de datos se necesita utilizar una de SQL Server,
    a. Para la creación de la base de datos se necesita el archivo que se encuentra en la carpeta *dataBaseScripts* del repositorio. Con dicho archivo se crea la base de datos en SQL Server.
    b. Para conectar la base de datos en VisualStudio, en el archivo de *Configuration.cs*, cambiar el atributo *connectionString* con la nueva cadena de conexión.

2. Arquitectura de Software

Para el desarrollo de la aplicación fue necesaria la creación de un API que se encarga de la conexión con la base de datos y lleva a cabo todas las consultas a la misma. En este desarrollo se utilizó una arquitectura de capas. Se utilizaron 3 capas en el proyecto del API que se conectan con una capa en la UI.

a) Core: donde se generaron las clases de los objetos principales necesarios para guardar la información de la red social.
b) Data Access Layer: aquí se implementaron todos los métodos donde se llevan a cabo los querys a base de datos.
c) Servicios: toma los métodos de la capa anterior para que sean consumidos por el controlador del API.
d) UI: la aplicación de Windows Forms consume los métodos del API conectándose al controlador.

Se eligió esta arquitectura porque proporciona al proyecto con una modularidad que facilita la detección y corrección de errores. De igual manera la arquitectura de capas permite que las capas conozcan y manipulen solamente lo necesario.

3. Patrones de Diseño

a) Creacionales

- **Singleton**: este patrón se utilizó para la conexión a base de datos. Cada llamada a base de datos requiere de conectarse a la misma y para ello se utiliza el paquete de C# de Sql. Para evitar crear distintas instancias de dicha conexión cada que se lleve a cabo un query, se decidió implementarlo como Singleton.

- **Builder**: para la creación del UI relacionado con el muro (feed), los posts en los distintos perfiles, el botón de like, la sección para agregar comentarios y la vista de todos los comentarios se utilizó este patrón. Se decidió implementarlo porque la creación de dichos elementos es muy parecida. Se crearon dos *Builders*, uno para los Feeds y otro para los elementos extras. De esta manera es más fácil implementar los atributos de cada elemento (sólo en el Builder) y hacer las modificaciones en los métodos necesarios.

b) Estructurales

- **Proxy**: este patrón fue el más obvio porque permitió consumir los métodos del API. De esta forma no se tiene que llamar explícitamente las rutas del API en el código principal, si no un simple método del patrón.

- **Decorator**: para éste fue un poco difícil encontrar un uso en el proyecto. Se decidió utilizarlo para el proceso de *tags* en los posts. Se tiene un post como elemento principal y se le agregan *decoradores* que son los tags. Así se hace un anidamiento de los mismos para incluirlos en base de datos.

c) Conductuales

- **Template Method**: todos los métodos que involucran a la base de datos se tienen que conectar, procesar alguna información y luego desconectar. Por ello se decidió implementar este patrón en esa sección. Sin embargo, los métodos reciben parámetros distintos y regresan también diferentes tipos de datos. Así que sólo se implementó el patrón en los métodos que tuvieran esas similitudes y resultaron ser los que checan, crean y eliminan amistades y solicitudes de amistad.

- **Chain of Responsibility**: después de investigar este patrón, me di cuenta que un proceso en mi aplicación que se divide en etapas es el Log In. Así que decidí implementar el patrón en esta parte. El patrón delega el chequeo del correo y contraseña a dos handlers: uno checa el correo y el otro la contraseña.

4. Base de Datos

Utilizando Microsoft SQL Server se creó una base de datos llamada *minifacebook*. En ésta se crearon 6 tablas:

I.	users	III.	friends	V.	likes
II.	post	IV.	comments	VI.	tags

Combinando los datos de estas 6 tablas se logró obtener toda la información necesaria para la aplicación.