**MINI INSTAGRAM**

**FUNCIONALIDAD**

* LogIn
* Registro
* Buscar Personas
* Seguir/Dejar Seguir Usuarios
* Visualizar Perfil de Usuario
* Desplegar seguidos/res
* Cargar Feed
* Agregar/Borrar comentarios
* Like Posts
* Guardar Posts
* Borrar Posts
* Visualizar Posts
* Publicar Posts
* Editar Perfil
* Agregar/Borrar Stories

**PATRONES DE DISEÑO**

* Factory Method - Constructor (Paneles)
* Single ton - Constructor (Proxy)
* Ncapas - Estructura (MiniInstagramAPI)
* Proxy - Estructura (Consumir Api)
* Template Method – Comportamiento (GetUser, GetPosts, LogIn, Registro)
* Strategy – Comportamiento (Ordenar Posts fecha Ascendente)

**JUSTIFICACION**

Decidí elaborar mi proyecto en C# en Visual Studio, utilizando Windows Forms, porque son una herramienta sencilla y fácil de utilizar. Además, la documentación oficial de Microsoft es muy buena.

**Patrones Constructores**

**Factory Method** – Este patrón fue utilizado ya que en mi solución necesitaba varios tipos de paneles, estos deberían aparecer dinámicamente y además deberían aparecer varias instancias de ellos. En este caso emplee el patrón que funcionase con una interfaz que se encargara de crear los paneles, o regresar el panel principal creado. De esta forma tenia varios tipos de constructores, que implementaran mi interfaz IObjeto del cual saldría la creación del tipo de panel necesario.

**Single ton** – Utilicé este patrón para crear una instancia persistente de mi Proxy, es decir, aproveché la simplicidad de este para restringir la creación de nuevas instancias. Esto para evitar múltiples conexiones al api.

**Patrones de Estructura**

**Ncapas** – Para esta implementación, utilicé mi solución de mi api, la divide en varios proyectos que funcionasen como capaz. Todo esto para mantener un control de los datos, que las capaz inferiores tuviesen mis modelos de construcción y acceso datos. Siendo así la capa de Servicios y el Controlador las mas importantes ya que son las que llevan la conexión con otras aplicaciones, y de ellas depende la comunicación del api.

**Proxy** – Utilice el patrón para consumir mi api, de esta manera mi interfaz fue más fácil de implementar ya que debía ser idéntica a mi interfaz de servicios del api. Mi conexión fue mas sencilla y mis peticiones fueron más fáciles de implementar al contar con una estructura encargada de hacerlas.

**Patrones de Comportamiento**

**Template Method** - Este patrón fue implementado aprovechando el comportamiento de mis clases de acceso a datos, es decir, varias de ellas necesitaban emplear las mismas funciones para lograr su funcionalidad. Así aproveché su función y logré aplicar el patrón en cuatro de mis clases GetUser, GetPosts, LogIn, Registro.

**Strategy** – En este caso use el patrón para crear una estrategia de ordenamiento, es decir mi interfaz implementa un método Ordenar el cual se encargará de recibir la lista de Posts y ordenarla. Para mis estrategias concretas solo utilicé un orden Ascendente. Con ello mi lista será ordenada por eventos más recientes.

**INSTALACION**

Modificar los archivos appsettings.json de ambos proyectos.

**REPOSITORIOS**

<https://github.com/Feresaul/MiniInstagram.git>

<https://github.com/Feresaul/MiniInstagramApi.git>