|  |
| --- |
| cifp Juan de colonia |
| Memorias de Biblionet |
| Desarrollo de interfaces: Proyecto 2 Eva |
|  |
| **Alexis López Briongos Dam2t** |
| **05/02/2024** |

|  |
| --- |
|  |

*Índice*

[Introducción al proyecto 2](#_Toc158056549)

[Construcción del proyecto 3](#_Toc158056550)

[1. Boceto 3](#_Toc158056551)

[2. Estructuración del proyecto: 6](#_Toc158056552)

[3. Patrón MVVM 9](#_Toc158056553)

[V\_Usuarios 9](#_Toc158056554)

[VM\_Usuario 10](#_Toc158056555)

[M\_Usuario 11](#_Toc158056556)

[M\_OperacionesUsuario 11](#_Toc158056557)

[4. Conclusiones 12](#_Toc158056558)

# Introducción al proyecto

1. **Título del Proyecto**: Biblionet
2. **Descripción del Proyecto**:
   * Esta aplicación tiene como objetivo principal facilitar a los usuarios la obtención de libros a través de préstamos. Los usuarios estándar pueden realizar solicitudes de préstamo, mientras que los usuarios administradores tienen la capacidad de gestionar aspectos clave como usuarios, libros, préstamos, y las incidencias reportadas por los usuarios.
3. **Objetivos: El proyecto tiene como objetivos:**

* Facilitar a los usuarios estándar la adquisición de libros a través de un sistema eficiente de préstamos.
* Permitir a los usuarios administradores la gestión efectiva de usuarios, libros, préstamos, y el manejo de incidencias.

1. **Público Objetivo**:
   * La aplicación está diseñada para atender a dos grupos de usuarios:
   * Usuarios Estándar: Personas que desean acceder a libros a través de préstamos.
   * Usuarios Administradores: Personal encargado de administrar y supervisar el sistema, gestionando usuarios, libros, préstamos, y abordando incidencias.
2. **Justificación**:
   * La creación de esta aplicación responde a la necesidad de proporcionar una plataforma eficiente y accesible para la obtención de libros mediante préstamos. La inclusión de funcionalidades administrativas facilita la gestión integral del sistema, permitiendo un control efectivo de usuarios, libros y préstamos. La implementación de esta solución contribuirá a mejorar la experiencia de los usuarios al mismo tiempo que optimiza la administración y operación del sistema.

# Construcción del proyecto

1. Boceto : este es la primera actualización del proyecto en el que he elaborado un boceto básico (está sujeto a cambios) en el que muestro las siguientes ventanas:



* Ventana de Inicio de sesión



* Ventana de creación de usuario



* Interfaz principal de Usuarios estándar.



* Interfaz usuario estándar donde podemos visualizar los libros para poder pedir prestados.

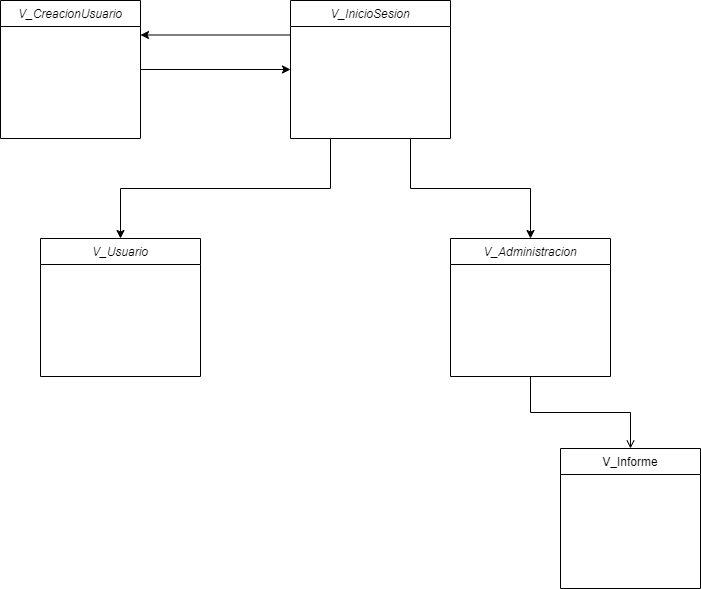


* Ventana de usuarios de tipo “Administrador” donde podemos visualizar las principales funcionalidades con este rol.

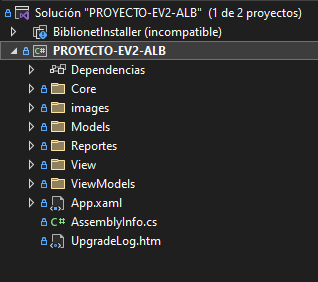


* Ventana de Administración de Libros en la cual podemos realizar operaciones tales como agregar, modificar y eliminar libros.

# Estructuración del proyecto:

****

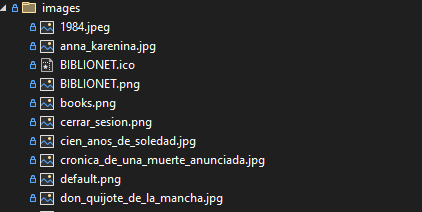
*Estas son las ventanas en las cuales operaremos durante todo el flujo de ejecución de la aplicación.*



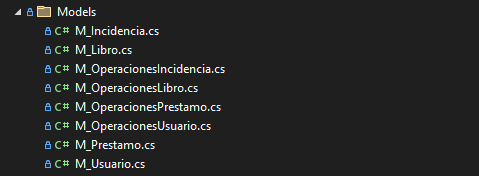
* Este es el cuerpo de la aplicación, en el cual contienen los siguientes componentes:

****

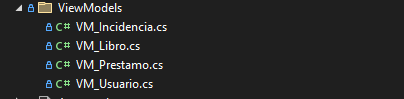
* **Core:** esta carpeta contiene todos los recursos de utilidad tales como:
  + **BBDD\_Biblioteca.sql:** este script SQL sirve para generar la base de datos con su estructura y datos.
  + **Comandos.cs:** clase en la cual contiene todos los comandos que luego utilizaremos en las vistas de los botones y atajos de los mismos.
  + **Conexión.cs:** clase (SINGLETON) en la cual contiene todos los métodos y atributos para establecer con el conector MySQL a la base de datos.
  + **Utils.cs:** clase en la cual contiene métodos tales como conversión de imágenes BitMapImage a un array de Bytes[] y viceversa para el almacenado de imágenes en la base de datos.

****

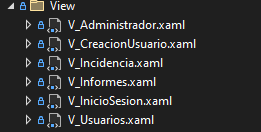
* **Images:** esta carpeta contiene todas las imágenes de las portadas de los libros, iconos, etc.

****

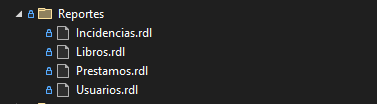
* **Models:** esta carpeta contiene todas las clases en las que guardaremos los datos de las tablas en objetos y todas las operaciones CRUD de dichos objetos.

****

* **ViewModels:** esta carpeta contiene todas las clases en las que se ven involucradas las lógicas de negocio y la comunicación entre las vistas y los modelos.

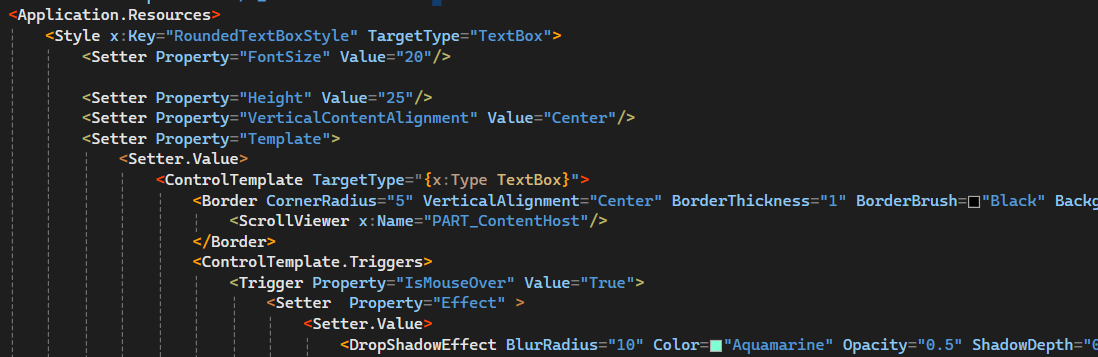
****

* **View:** esta carpeta engloba todas las clases relacionadas con la interfaz de usuario, diseñadas para la interacción visual de las ventanas.



* **Reportes:** en esta carpeta, se almacenan todos los documentos con extensión .rdl, los cuales despliegan informes detallados generados a partir de las diversas funcionalidades de la aplicación.



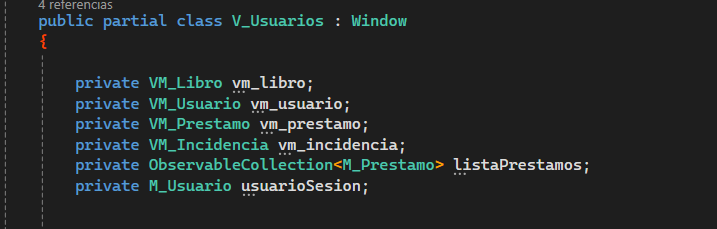


* **App.xaml:** En esta clase, se han centralizado todos los estilos que impactan en los controles de la aplicación, abarcando elementos como botones, cuadros de texto, y más.

# Patrón MVVM

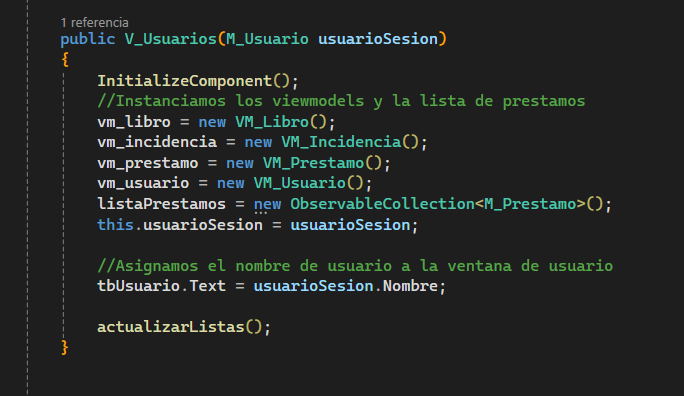
* Este proyecto se ha estructurado empleando el patrón MVVM (Modelo-Vista-ViewModel) con el objetivo de lograr una abstracción eficiente tanto de la información como de la interacción del usuario con la aplicación.
* A continuación mostraré un ejemplo de interacción entre las diferentes capas de la interfaz de usuario (estándar).

# V\_Usuarios

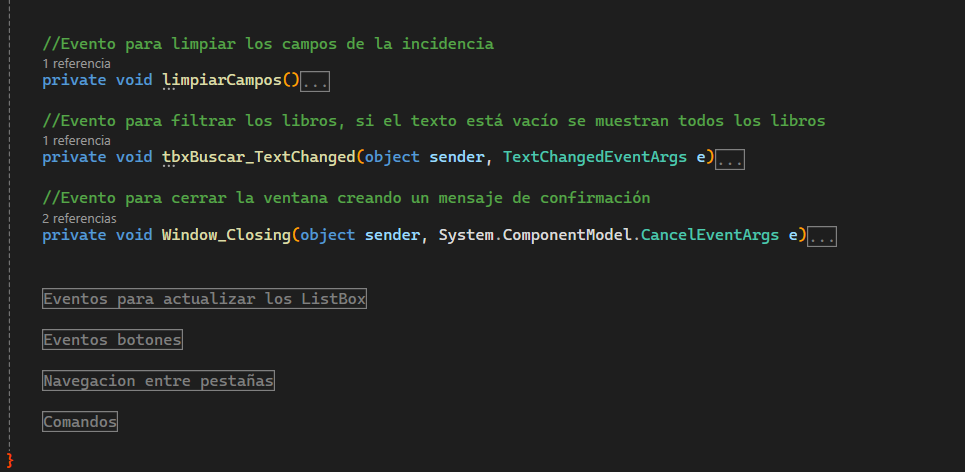


*Atributos de la clase V\_Usuarios*

* Esta es la clase V\_Usuarios la cual hace la función de vista para que el usuario pueda comunicarse con la aplicación.



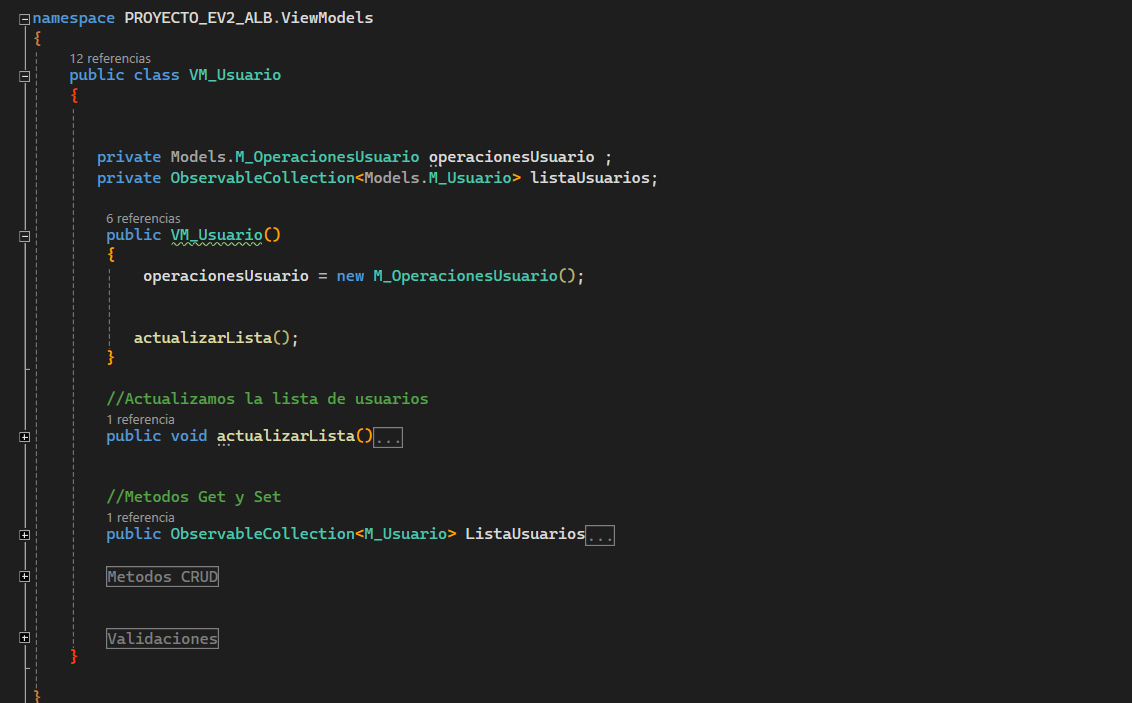
* En el constructor de la clase le pasaremos un objeto de tipo M\_Usuario (modelo del usuario) e inicializamos todos los ViewModels relacionado con las funcionalidades que tiene el usuario estándar en la aplicación.



*Métodos de la clase V\_Usuarios*

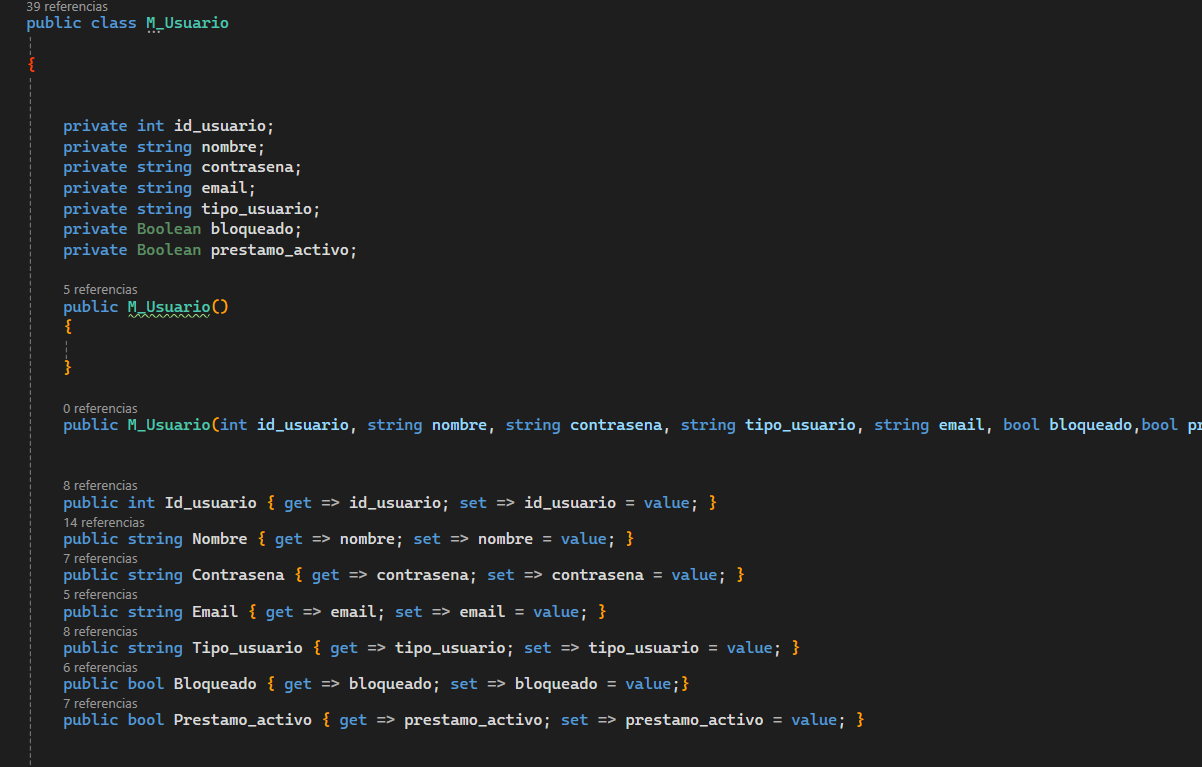
* También esta clase tendrá todos los métodos necesarios para poder intercambiar y visualizar información con el ViewModel afectado.

# VM\_Usuario



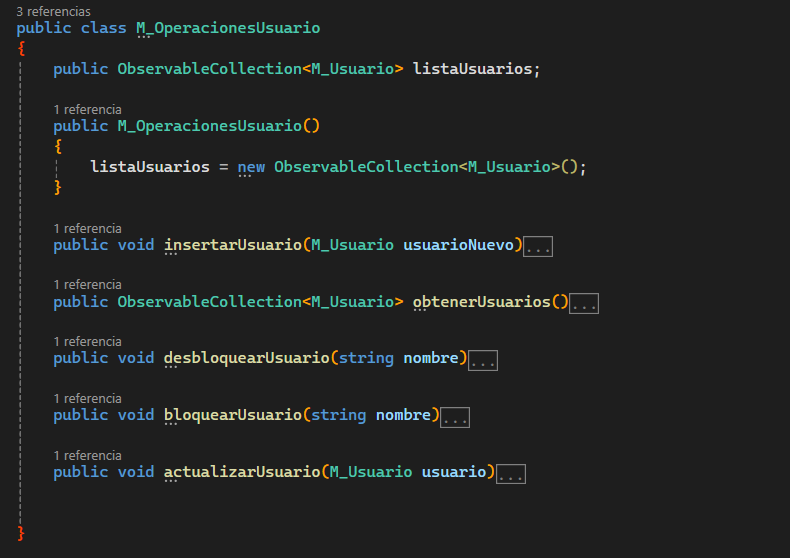
* En la clase VM\_Usuario será el intermediario, el medio de conexión que manejara la lógica de negocio entre el modelo(datos) y las entradas del usuario.
* Obtendrá métodos de validación para las entradas del usuario y los métodos CRUD para obtener información o actualizarla.

# M\_Usuario



* En esta clase, hemos implementado un objeto Usuario siguiendo el paradigma de programación orientada a objetos (POO). Este objeto POCO (Plain Old CLR Object) se utiliza para recuperar información de las tablas y almacenarla de manera estructurada.

# M\_OperacionesUsuario



* En esta clase, se han centralizado todas las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) que establecen la comunicación con la base de datos real.

# Conclusiones

¿Con qué dificultades nos hemos encontrado desde el proyecto inicial?

* + La comunicación entre las capas y la actualización en tiempo real de los datos.
  + Diseño de las interfaces.

¿Hemos tenido aplicar algún cambio de la idea inicial? ¿Por qué?

* + Sí, añadido de funcionalidad en el apartado de administración y reestructuración de los objetos préstamos para que pudiera albergar en él otros objetos.
  + Mejora visual de la interfaz para aprovechar más el espacio y sea más atractiva al ojo humano.

¿Hemos cumplido los objetivos especificados inicialmente?

* + Sí, incluso he añadido más funcionalidades de las planeadas en un principio.

¿Hay alguna propuesta de mejora que se podría plantear para el futuro?

* + Más mejoras en la interfaz visual.
  + Realización de informes con alguna herramienta más sencilla y efectiva.