|Trabajo práctico n° 4 |

Leandro Muñoz, División A

El proyecto que desarrollé pretende brindar un servicio de gestión de clientes para gimnasios.

Sus funcionalidades son: Loguearse con un usuario libre para mostrar un mensaje de bienvenida al iniciar la app, pero con una contraseña predefinida (usuario: **nombre de la persona que ingresa**, contraseña: **gimnasio22**). Se podría agregar un formulario de registro para que el usuario pueda darle uso, pero no me pareció necesario hacerlo en este trabajo ya que no se pedía en las consignas. Si bien el Login no se pidió, me pareció que quedaba estético y decidí añadirlo.

Dicho proyecto permite agregar clientes (**edad mínima para inscribirse al gimnasio 16 años, sólo se pueden poner números en el DNI y letras en los campos nombre y apellido**), buscar clientes ya sea para modificarlos o eliminarlos utilizando el DNI. Si el DNI es inexistente, el botón buscar no hará nada.

Los botones eliminar y modificar desglosan el mismo formulario donde se pueden realizar cualquiera de las dos funciones. Lo realicé de esta manera debido a que no encontré un ícono que represente la función de modificar y eliminar al mismo tiempo. Así que decidí poner los dos íconos y linkearles el mismo diseño del formulario y sus funcionalidades.

El botón cerrar sesión da por finalizado el programa. Lo ideal sería que vuelva a abrir el formulario de Login, pero no lo logré hacer funcionar de manera efectiva y decidí posponerlo para hacerlo correctamente más adelante.

También cuenta con los cuatro botones para serializar y deserializar JSON y para serializar y deserializar XML.

Implementación de los temas en el proyecto:

- **Excepciones:** A mi criterio, cualquier posible excepción estaba controlada de que no ocurra en el ingreso de datos mismo. Pero para implementar el tema creé una excepción llamada **ExcepcionDniInvalido** que tiene como objetivo no permitir un ingreso de datos distinto al tipo numérico. Está implementada en la clase Cliente Encontrado, **línea 74.**
- **Pruebas unitarias:** Implementadas en el archivo llamado PruebasUnitarias dónde se prueban los métodos AgregarElemento y BuscarElemento.
- **Tipos genéricos:** Creé una clase llamada Sistema donde implemento dicho tema haciendo genéricos los métodos AgregarElemento y BuscarElemento. También creé una interfaz genérica ISerializadora.
- **Interfaces:** Aplicado dicho concepto en la interfaz genérica ISerializadora, que contiene a Serializar y Deserializar.

- **Archivos:** El programa posee las clases SerializadoraJSON y SerializadoraXML dónde en su desarrollo se implementa el concepto.

- **Base de datos:** Clase ClientesDB posee los métodos que interactúan con la **BBDD**. En la carpeta del archivo está la Query que crea dicha **BBDD**.
- **Delegados y expresiones lambda / Hilos :** Fue implementado para poder lanzar nuevos hilos. En **Gestor_Profes** se puede ver dicha implementación de ambos temas.
- **Eventos:** Implementado en **Gestor_Profes** para informar cuando un cliente se añade exitosamente.

 Cuestiones a corregir de la revisión de Felipe Bustos: 

- **Unit test.cs mal nombrado. :** **Corregido ,** ahora se llama TestsUnitarios.cs

- Al dejar un campo vacío en el Agregar, muestra la advertencia pero lo agrega igual, no debería.: **Corregido**.

- El eliminar o modificar estaría bueno que esten separados, porque los botones de arriba terminan llevando al mismo lugar una vez encontrado el cliente.: **Corregido**. Ahora hay un solo botón que lleva a la misma sección. El botón es **Buscar Cliente** y lleva a la sección donde se puede eliminar o modificar el cliente .

- En la busqueda del cliente no esta bueno que si no existe un cliente con ese dni no haga nada, deberias mostrar un mensaje de que no existe o algo para mostrarle a la persona que esta usando el programa que no existe porque da la sensación de que no funciona.: **Corregido**. Ahora muestra un mensaje si el Dni no está en la base de datos del programa.

- Y si hay algun cambio, ya sea que se agregó un cliente o se eliminó se recarge la lista automaticamente y no tener que darle al boton de importar de la base de datos, es confuso y poco util.: **Corregido**. Ahora automáticamente importa la base de datos apenas inicia la aplicación y si hay cambios refresca la lista.