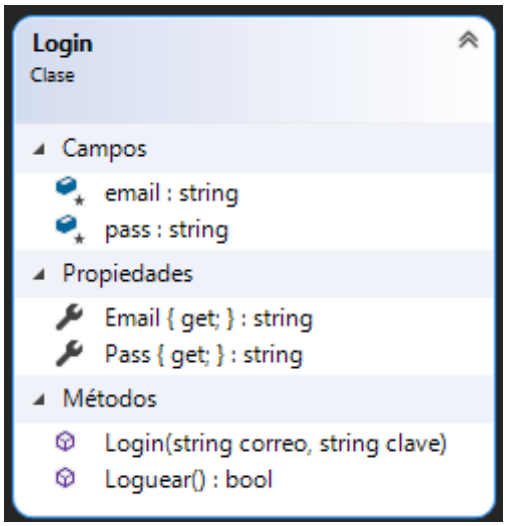
FINAL – LABORATORIO 2 – 2022

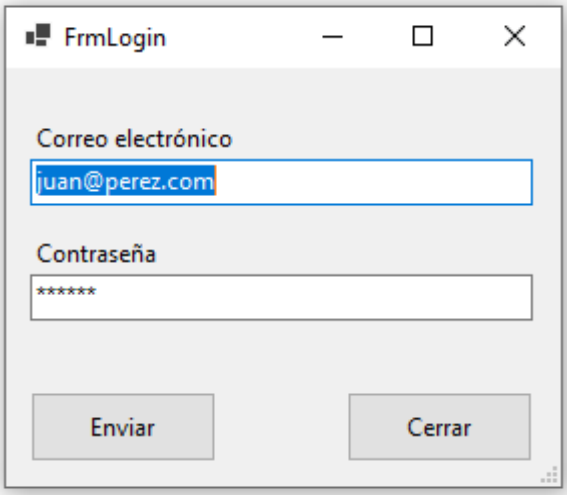
IMPORTANTE:

- **Dos (2) errores en el mismo tema anulan su puntaje.**
- La correcta documentación y reglas de estilo de la cátedra serán evaluadas.
- Colocar sus datos personales en el nombre de la carpeta principal y en la solución: ***Apellido.Nombre***. Ej: Pérez.Juan. No sé corregirán proyectos que no sea identificable su autor.
- Los exámenes que no compilen, no aprueban.
- **Reutilizar** tanto código como sea posible.
- Tanto en métodos, atributos y propiedades, colocar nombre de la clase (en estáticos), **this** o **base**.
- Aplicar los principios de los 4 pilares de la POO.
- La entrega será en un archivo comprimido, el cual debe contar con Apellido y Nombre, al igual que la solución. Se entregará al finalizar, mediante un Google Form.
- La duración del final es de 120 minutos.
- Partir del formulario dado.

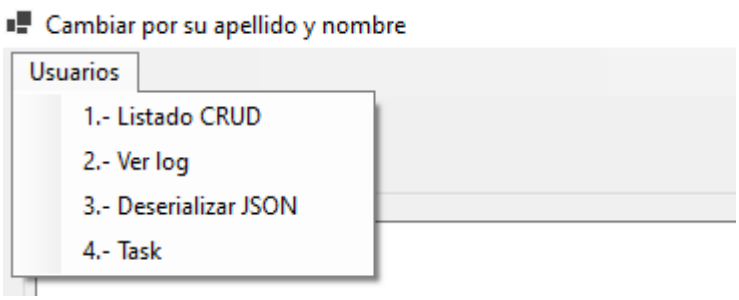
Todos los ítems que siguen serán evaluados, con puntaje, preste atención:

1. Cambiar el nombre de la carpeta de la Solución con sus datos personales: Apellido.Nombre.
2. Tener en cuenta el control de excepciones. NO debe haber excepciones no controladas.
3. Crear un proyecto de tipo Bibliotecas de Clase, con el nombre ***Entidades.Final***.
4. Crear en dicho proyecto la clase **Login.**
   Método de instancia público:
   Loguear(): Se conectará con la BD enviando correo y contraseña, si existe en la tabla usuarios, retornará true, caso contrario, false.
5. *FrmLogin* interactua con **Entidades.Final.Login**, si las credenciales son correctas se visualizará *FrmPrincipal*.

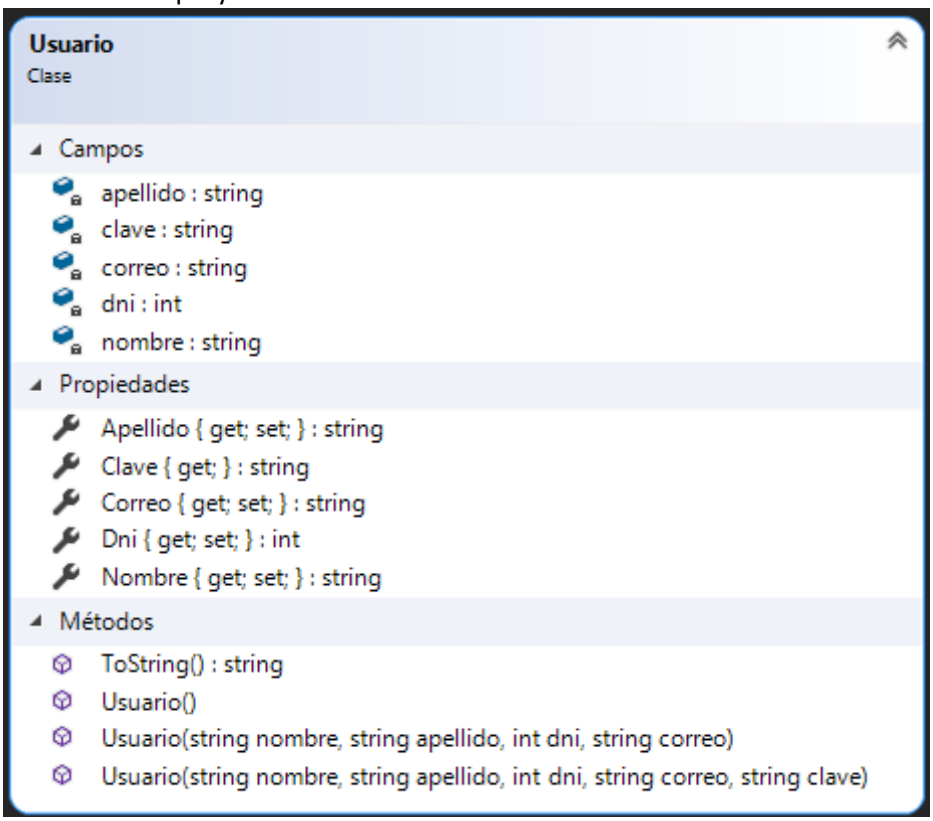




6. FrmPrincipal contiene un menú de opciones, como mustra la siguiente imagen.

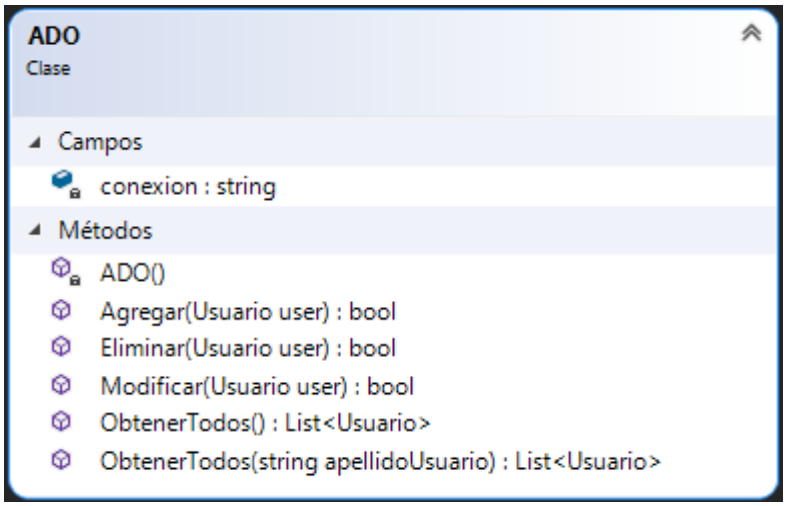


7. Agregar la clase **Usuario** al proyecto Entidades.Final:



Nota: En ningún caso, el método ToString debe de mostrar la clave del usuario.

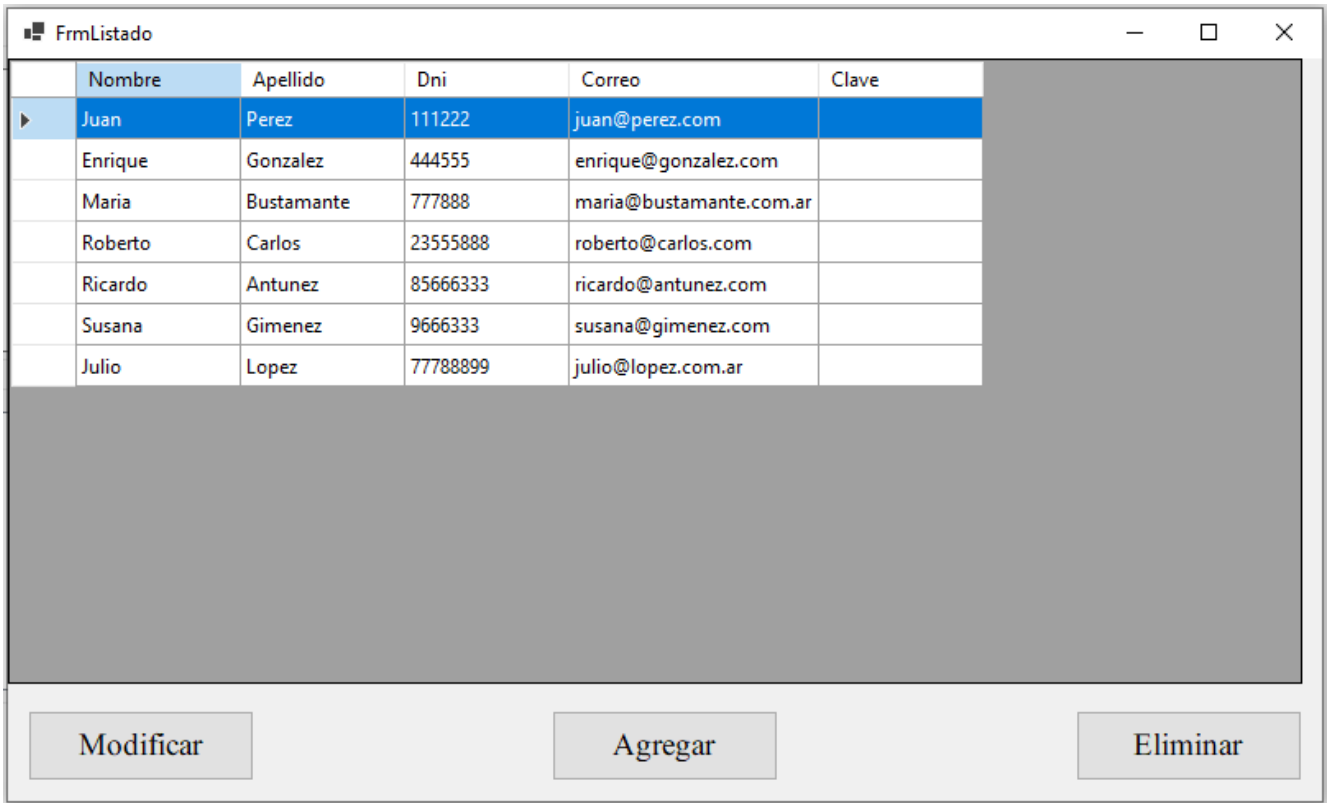
8. El CRUD sobre la base de datos se realizará mediante la clase ADO:



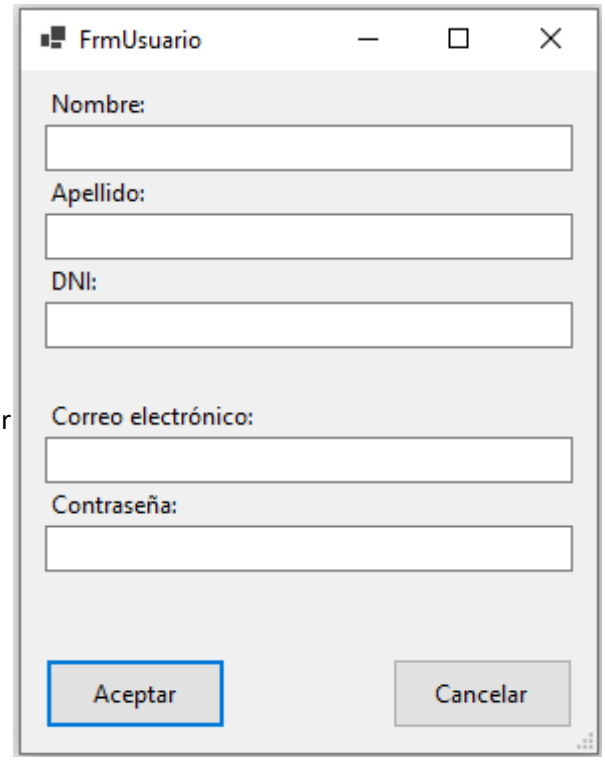
El método *ObtenerTodos* (y su sobrecarga) son métodos de clase y retornan una lista de Usuarios.
Los métodos *Agregar*, *Eliminar* y *Modificar* son métodos de instancia y retornan un booleano que indica si se pudo o no agregar, eliminar o modificar la base de datos.

Nota: tanto la modificación como la eliminación se harán por DNI.

9. **Menú -> Usuarios -> 1.- Listado CRUD:**
    a. Se accede a *FrmListado*.
    b. Contiene un ***DataGridView*** que está enlazado con una lista genérica de tipo Usuario (se carga desde la base de datos).
    c. Posee los botones: Agregar, Modificar y Eliminar.



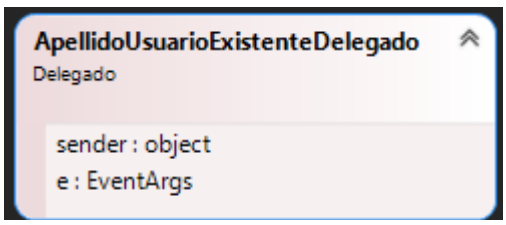
10. **Agregar:**
    a. Agrega un nuevo usuario a la base de datos utilizando *FrmUsuario*.
    b. FrmUsuario expone en una propiedad (pública y de sólo lectura) de tipo Usuario.
    c. Interactuar con la clase ADO.

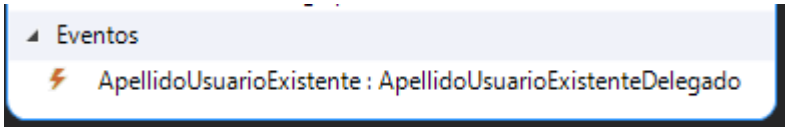


11. **Modificar:**
    a. Se muestra en *FrmUsuario* los valores del objeto seleccionado del DataGridView.
    b. El valor del DNI no se debe poder modificar (adecuar FrmUsuario).
    c. Interactuar con la clase ADO.
    d. Refrescar listado.

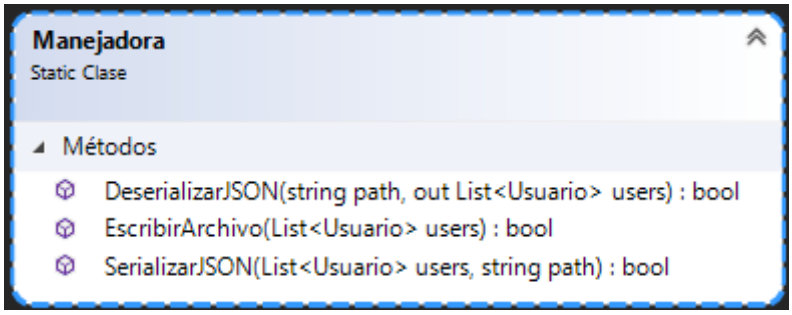
12. **Eliminar:**
    a. Se muestra en FrmUsuario los valores del objeto seleccionado del DataGridView.
    b. Interactuar con la clase ADO.
    c. Refrescar listado.

13. Agregar un evento en la clase ADO.
    a. El evento se llamará ***ApellidoUsuarioExistente***.
    b. Si el apellido del usuario a ser agregado (a la base de datos) ya existe, se disparará dicho evento.
    c. Utilizar método ObtenerTodos y pasar la lista de usuarios al manejador de eventos correspondiente.



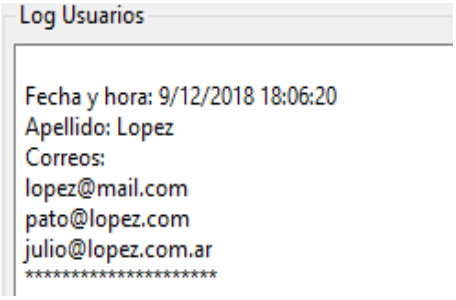


14. Adecuar el punto 10 para que al agregar un nuevo usuario:
    a. Si el apellido del usuario que se agregó ya existe en la base se disparará el evento ***ApellidoUsuarioExistente***.
    Asociar dinamicamente al manejador de eventos ***Manejador_apellidoExistenteLog*** y al manejador de eventos ***Manejador_apellidoExistenteJSON***.
    b. Una vez capturado el evento en ***Manejador_apellidoExistenteLog***, se pide que se guarde en un archivo de texto:
    La fecha (con hora, minutos y segundos) y en un nuevo renglón, el apellido (repetido) y TODOS los correos electrónicos para ese apellido.
    Se deben acumular los mensajes.
    El archivo se guardará con el nombre 'usuarios.log' en la carpeta 'Mis documentos' del cliente.
    El manejador de eventos (Manejador_apellidoExistente) invocará al método (de clase) ***EscribirArchivo***(List<Usuario>) (se alojará en la clase **Manejadora** en Entidades.Final), que retorna un booleano indicando si se pudo escribir o no.
    c. Una vez capturado el evento en ***Manejador_apellidoExistenteJSON***, serializará a JSON la lista de objetos de tipo Usuario que contiene a los usuarios cuyos apellidos coinciden con el del nuevo usuario agregado.
    El archivo se guardará en el escritorio del cliente con el nombre 'usuarios_repetidos.json'.
    El manejador de eventos (Manejador_apellidoExistenteJSON) invocará al método (de clase) ***SerializarJSON***(List<Usuario>, string) (se alojará en la clase **Manejadora** en Entidades.Final), que retorna un booleano indicando si se pudo escribir o no.



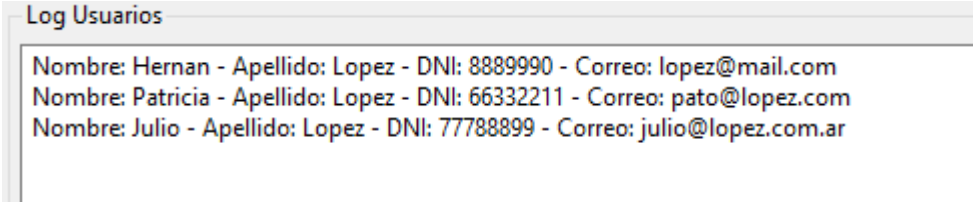
---

15. **Menú -> Usuarios -> 2.- Ver Log**
    a. Configurar el ***OpenFileDialog*** para poder seleccionar el log de usuarios.
    ///Título -> 'Abrir archivo de usuarios'
    ///Directorio por defecto -> Mis documentos
    ///Tipo de archivo (filtro) -> .log
    ///Extensión por defecto -> .log
    ///Nombre de archivo (defecto) -> usuarios



▶	Hernan	Lopez	8889990	lopez@mail.com	
	Patricia	Lopez	66332211	pato@lopez.com	
	Julio	Lopez	77788899	julio@lopez.com.ar	

16. **Menú -> Usuarios -> 3.- Deserializar JSON**
    a. Diseñar el método estático (Manejadora) ***DeserializarJSON***(string, out List<Usuario>): bool.
    b. Mostrar en txtUsuariosLog el contenido de la deserialización del archivo 'usuarios_repetidos.json'.



17. **Menú -> Usuarios -> 4.- Task**
    a. Iniciar un hilo secundario que permita visualizar en ***lstUsuarios*** los datos de todos los usuarios que están en la base de datos.
    El ciclo será continuo y se detendrá cuando se dispare el evento FormClosing de FrmPrincipal.
    El listado de usuarios se refrecará cada 1,5 segundos.
    b. Desarrollar un método que invoque al método *ADO.TraerTodos* y luego mostrar el contenido en el control especificado.
    c. La información mostrada debe iterar con el color de fondo (a negro) y el color de la fuente (a blanco) y lo intercambie (fondo a blanco y fuente a negro), agregando un retardo de 1.5 segundos por cada intercambio.

NOTA: propiedades BackColor (fondo) y ForeColor (fuente)
Colores:
System.Drawing.Color.Black (negro)
System.Drawing.Color.White (blanco)