**POSIBLES MODIFICACIONES:**

* ~~Agrear en el form de config una opción donde permita ingresar un path de un archivo, o la carga de un archivo, para desp aprovechar en usar excepciones o algún otro tema.~~

**Excepciones:**

● ~~Todo tipo de excepción debe ser controlada. No debería haber ningún caso en el que~~

~~el programa se rompa abruptamente.~~

● Por otro lado se deberá crear mínimo una excepción propia. La misma debe

responder a una lógica determinada, según el proyecto que estén desarrollando (Por

ejemplo que encapsule algún método que desarrolle alguna lógica, o que resuelva

algo particular de su proyecto).

**Archivos y serialización:**

~~● Deberán guardar en un archivo txt, un log de errores, documentando fecha y hora~~

~~del incidente, descripción, clase y método en donde ocurrió.~~

~~● El programa deberá contar con una opción de configuraciones que serán leídas~~

~~desde un archivo Json. Cualquier cambio en estas configuraciones deben ser~~

~~persistidas en el mismo archivo. Por ejemplo: imágenes, colores, fuentes, sonidos,~~

~~etc.~~

~~● Una opción del programa permitirá guardar en xml algún dato específico generado~~

~~por su programa: por ejemplo el stock de productos~~.

**Generics:**

● Aplicar el concepto de Generics en donde se considere necesario.

**Interfaces:**

● Implementar un mínimo de tres interfaces. La primera (y la típica) es interfaces y

serialización. El resto deberá adaptarse a las necesidades de su proyecto.

🡨 IMensajeError / ILoginUsuario

**Test Unitario:**

● ~~Realizar un mínimo de 3 métodos de test unitario. Los métodos que se prueban~~

~~deben tener cierto grado de complejidad (por ejemplo no aplicaría si quiero probar~~

~~un método que calcula un promedio).~~

**Base de datos:**

~~● Aplicar consultas dml para manipular los datos de su programa.Los métodos~~

~~correspondientes al motor, tienen que estar debidamente separados en clases.~~

~~● No deberá haber más datos hardcodeados en el sistema.~~

~~● El programa deberá permitir generar nuevos operarios y supervisores por medio de~~

~~un registro que a su vez serán guardados en una base de datos.~~

~~● El supervisor también deberá poder ser capaz de realizar consultas de tipo~~

~~CRUD(Create - Read - Update - Delete) en su panel de control.~~

**Delegados:**

~~● Aplicar delegados a distintos métodos de ordenamiento utilizando expresiones~~

~~lambda.~~  🡨 aplicado en clase Administracion.

~~● Crear un delegado de tipo propio.~~  🡨 En FrmLogin cargaDeUsuariosDataGrid

Y en FrmConfiguracion en método IniciarCarga

● Aplicar tipos de delegados.

● Envío de datos entre formularios mediante delegados.

**Eventos:**

● ~~Crear un mínimo de 2 eventos propios que deben ser disparados y capturados en~~

~~distintas situaciones.~~ 🡨 EventoLogError y EventoLoginUsuario en administración.

~~● Manejar eventos predeterminados (ej el evento click de un botón) de manera~~

~~dinámica (Event Handler)~~ 🡨 FrmConfiguracion

● Aplicar un mínimo de dos tareas asincrónicas. Un ejemplo podría ser permitir la

cancelación de la carga de un archivo muy pesado. Otro ejemplo: realizar la carga

de un archivo mediante un hilo. 🡨 CargarArchivo() en FrmConfiguracion