UNIVERSIDAD ABIERTA INTERAMERICANA

FACULTAD DE TECNOLOGÍA INFORMÁTICA

PROGRAMACIÓN Y ESTRUCTURA DE DATOS

Descripción del Proyecto

Desarrollar una aplicación de escritorio en .NET 8 que permita manipular y convertir archivos de texto en múltiples formatos mediante un menú interactivo por consola.

La información a registrar en dichos archivos será la siguiente:

Crear una clase alumno con los siguientes atributos:

Legajo, Apellido, Nombres, Número Documento, Email, Teléfono, almacenar todos como strings.

Formatos de Archivo Soportados

La aplicación trabajará con los siguientes formatos:

1. **TXT** - Archivos de texto plano
2. **CSV** - Valores separados por comas
3. **JSON** - Notación de objetos JavaScript
4. **XML** - Lenguaje de marcado extensible

Funcionalidades Requeridas

Menú Principal

=== GESTOR DE ARCHIVOS DE TEXTO ===

1. Crear nuevo archivo

2. Leer archivo existente

3. Modificar archivo existente

4. Eliminar archivo

5. Convertir entre formatos

6. Crear Reporte con Corte de control de un nivel

0. Salir

Detalle de Funcionalidades

1. Crear Nuevo Archivo

Descripción: Permite crear un nuevo archivo en el formato seleccionado y cargar registros de alumnos.

Flujo de Operación:

- Solicitar el nombre del archivo (sin extensión)
- Solicitar el formato de destino (TXT, CSV, JSON, XML)
- Solicitar la cantidad de alumnos a registrar
- Para cada alumno, solicitar:
    - Legajo
    - Apellido
    - Nombres
    - Número de Documento
    - Email
    - Teléfono
- Validar que los datos ingresados no estén vacíos
- Validar formato de email (opcional pero recomendado)
- Guardar el archivo con la extensión correspondiente
- Mostrar mensaje de confirmación con la ruta del archivo creado

Formato de Almacenamiento:

- **TXT**: Un alumno por línea, campos separados por pipe (|)

    001|García|Juan Carlos|12345678|juan.garcia@email.com|1122334455

    002|Pérez|María Laura|23456789|maria.perez@email.com|1133445566

- **CSV**: Encabezado + registros separados por comas

    Legajo,Apellido,Nombres,NumeroDocumento,Email,Telefono

    001,García,Juan Carlos,12345678,juan.garcia@email.com,1122334455

- **JSON**: Array de objetos Alumno

```
[
  {
    "Legajo": "001",
    "Apellido": "García",
    "Nombres": "Juan Carlos",
    "NumeroDocumento": "12345678",
    "Email": "juan.garcia@email.com",
    "Telefono": "1122334455"
  }
]
```

- **XML**: Estructura jerárquica

```
<Alumnos>
  <Alumno>
    <Legajo>001</Legajo>
    <Apellido>García</Apellido>
```

```
    <Nombres>Juan Carlos</Nombres>
    <NumeroDocumento>12345678</NumeroDocumento>
    <Email>juan.garcia@email.com</Email>
    <Telefono>1122334455</Telefono>
  </Alumno>
</Alumnos>
```

---

2. Leer Archivo Existente

Descripción: Permite visualizar el contenido de un archivo existente de forma formateada.

Flujo de Operación:

- Solicitar el nombre completo del archivo (con extensión)
- Verificar que el archivo exista
- Detectar automáticamente el formato según la extensión
- Leer y parsear el archivo según su formato
- Mostrar los datos en formato tabla por consola:
- ```
  ================================================================================|
  |Legajo    | Apellido     | Nombres          | Nro. Doc.    | Email                        | Teléfono       |
  ================================================================================|
  | 001       | García       | Juan Carlos      | 12345678   | juan.garcia@email.com    | 1122334455   |
  | 002       | Pérez        | María Laura      | 23456789   | maria.perez@email.com    | 1133445566   |
  ================================================================================T
  otal de alumnos: 2
  ```
- Manejar errores de lectura y mostrar mensajes apropiados
- Opción de pausar después de cada 20 registros (paginación)

---

3. Modificar Archivo Existente

Descripción: Permite editar registros existentes, agregar nuevos o eliminar alumnos del archivo.

Flujo de Operación:

- Solicitar el nombre del archivo a modificar
- Verificar que el archivo exista
- Cargar todos los registros en memoria
- Mostrar sub-menú de modificación:
- === OPCIONES DE MODIFICACIÓN ===
    1. Agregar nuevo alumno
    2. Modificar alumno existente (por legajo)
    3. Eliminar alumno (por legajo)
    4. Guardar y salir
    5. Cancelar sin guardar

  Opción 1 - Agregar Nuevo Alumno:
  - Solicitar todos los datos del nuevo alumno
  - Verificar que el legajo no exista previamente
  - Agregar al listado en memoria

  Opción 2 - Modificar Alumno:
  - Solicitar el legajo del alumno a modificar
  ```

- Mostrar los datos actuales
- Permitir editar campo por campo
- Opción de dejar el valor actual presionando Enter
- Actualizar el registro en memoria

Opción 3 - Eliminar Alumno:
- Solicitar el legajo del alumno a eliminar
- Mostrar los datos del alumno encontrado
- Solicitar confirmación
- Eliminar del listado en memoria

Opción 4 - Guardar:
- Crear backup del archivo original (renombrar con sufijo .bak)
- Guardar los cambios en el archivo original
- Mantener el formato original del archivo

Opción 5 - Cancelar:
- Descartar todos los cambios
- Volver al menú principal

---

4. Eliminar Archivo

Descripción: Permite eliminar físicamente un archivo del sistema.

Flujo de Operación:

- Solicitar el nombre completo del archivo (con extensión)
- Verificar que el archivo exista
- Mostrar información del archivo:
  - Nombre completo
  - Tamaño en KB
  - Fecha de creación
  - Fecha de última modificación
- Solicitar confirmación escribiendo "CONFIRMAR"
- Si se confirma:
  - Eliminar el archivo
  - Mostrar mensaje de éxito
- Si se cancela:
  - Volver al menú principal sin eliminar

---

5. Convertir Entre Formatos

Descripción: Permite convertir un archivo de alumnos de un formato a otro manteniendo la integridad de los datos.

Flujo de Operación:

- Solicitar el archivo de origen (con extensión)
- Verificar que el archivo exista
- Detectar formato de origen automáticamente

- Mostrar formatos de destino disponibles:
- Formato actual: CSV. Seleccione formato de destino: 1. TXT 2. JSON 3. XML
- Solicitar nombre del archivo de destino (sin extensión)
- Leer y parsear el archivo origen
- Convertir los datos al formato destino
- Guardar el nuevo archivo
- Mostrar resumen de la conversión:
- ✓ Conversión exitosa
  Archivo origen: alumnos.csv (3 registros) Archivo destino: alumnos.json (3 registros)

Conversiones Soportadas:

- TXT ↔ CSV ↔ JSON ↔ XML
- Mantener todos los datos sin pérdida de información
- Respetar la estructura de la clase Alumno

---

6. Crear Reporte con Corte de Control de un Nivel

Descripción: Genera un reporte agrupado por un campo específico (Apellido) con subtotales y formato profesional.

Flujo de Operación:

- Solicitar el archivo fuente (cualquier formato soportado)
- Verificar que el archivo exista y contenga datos
- Cargar todos los registros
- Ordenar los alumnos alfabéticamente por Apellido
- Agrupar por Apellido (corte de control)
- Generar el reporte con el siguiente formato:

```
================================================================================
                    REPORTE DE ALUMNOS POR APELLIDO
                    Fecha: 24/11/2025 15:30:45
================================================================================
APELLIDO: GARCÍA
--------------------------------------------------------------------------------
  Legajo: 001
  Nombres: Juan Carlos
  Documento: 12345678
  Email: juan.garcia@email.com
  Teléfono: 1122334455

  Legajo: 005
  Nombres: Ana María
  Documento: 34567890
  Email: ana.garcia@email.com
  Teléfono: 1144556677

  → Subtotal GARCÍA: 2 alumno(s)

APELLIDO: PÉREZ
--------------------------------------------------------------------------------
```

Legajo: 002
Nombres: María Laura
Documento: 23456789
Email: maria.perez@email.com
Teléfono: 1133445566

Legajo: 008
Nombres: Roberto Carlos
Documento: 45678901
Email: roberto.perez@email.com
Teléfono: 1155667788

→ Subtotal PÉREZ: 2 alumno(s)

```
================================================================================
                        RESUMEN GENERAL
================================================================================
Total de Apellidos diferentes: 2
Total de Alumnos registrados: 4
================================================================================
```

Funcionalidades del Reporte:

- Mostrar el reporte por pantalla
- Opción de guardar el reporte en archivo TXT
- Incluir fecha y hora de generación
- Contador de alumnos por apellido (subtotal)
- Totalizador general
- Formato claro y profesional con separadores visuales

Implementación Técnica:

- Usar LINQ GroupBy() para agrupar por Apellido
- Usar LINQ OrderBy() para ordenar
- Usar Count() para contar registros por grupo
- Formatear con String.PadRight() y String.PadLeft() para alineación

---

Criterios de Evaluación

Criterio	Descripción	Puntos
Funcionalidad Completa	Todas las opciones del menú funcionan correctamente según lo especificado	25%
Clase Alumno	Implementación correcta de la clase con todos sus atributos y métodos necesarios	10%

Criterio	Descripción	Puntos
Manejo de Archivos	Lectura, escritura y manipulación correcta de archivos en todos los formatos	15%
Conversión de Formatos	Conversión bidireccional sin pérdida de datos entre todos los formatos	15%
Reporte con Corte de Control	Implementación correcta del agrupamiento, subtotales y formato profesional	15%
Manejo de Excepciones	Try-catch apropiados, validaciones y mensajes de error informativos	10%
Calidad del Código	Código limpio, comentado, uso de métodos reutilizables, nomenclatura adecuada	5%
Uso de LINQ	Aplicación correcta de LINQ en búsquedas, filtros y agrupamientos	5%

Total: 100%

Requisitos Mínimos para Aprobar (60%)

- Funcionalidad 1, 2, 3 y 4 operativas
- Al menos 2 formatos de archivo funcionando correctamente
- Manejo básico de excepciones
- Clase Alumno implementada

Para Obtener Calificación Destacada (85%+)

- Todas las funcionalidades operativas y sin errores
- Validaciones robustas (email, legajos únicos, etc.)
- Código bien organizado y documentado
- Interfaz de usuario clara y amigable
- Reporte con formato profesional

📤 Formato de Entrega

1. Estructura de Carpetas

```
ApellidoNombre_GestorAlumnos/
│
├── src/
│   ├── Program.cs
│   ├── Menu.cs
│   ├── GestorArchivos.cs
│   ├── Conversor.cs
```

```
│   ├── GeneradorReportes.cs
│   └── Models/
│       └── Alumno.cs
│
├── docs/
│   ├── Manual_Usuario.pdf (o .docx)
│   └── Documentacion_Tecnica.pdf (opcional)
│
├── ejemplos/
│   ├── alumnos_ejemplo.txt
│   ├── alumnos_ejemplo.csv
│   ├── alumnos_ejemplo.json
│   └── alumnos_ejemplo.xml
│
├── README.md
└── ApellidoNombre_GestorAlumnos.sln
```

2. Archivo README.md

Debe contener:

- Nombre del alumno y legajo
- Descripción breve del proyecto
- Requisitos del sistema (.NET 8 SDK)
- Instrucciones de compilación:
- Instrucciones de uso básico
- Problemas conocidos (si existen)
- Extras implementados (si corresponde)

3. Manual de Usuario

Documento PDF o Word con:

- Portada (nombre del proyecto, alumno, fecha)
- Índice
- Introducción
- Capturas de pantalla de cada funcionalidad
- Ejemplos de uso paso a paso
- Casos de error comunes y sus soluciones

4. Video Demostración (3 - 5 minutos)

Debe mostrar:

- Ejecución del programa desde cero
- Creación de archivo con al menos 3 alumnos
- Lectura del archivo creado
- Modificación de un registro
- Conversión a otro formato
- Generación del reporte con corte de control
- Voz en off explicando cada paso (no obligatorio pero recomendado)

Plataformas sugeridas: Subirlos a alguna plataforma conocida, ejemplo YouTube (privado), Google Drive

5. Archivos de Ejemplo

Incluir al menos 3 archivos de prueba con datos reales de ejemplo:

- 1 archivo con mínimo 10 alumnos
- Apellidos repetidos para probar el corte de control
- Datos válidos y consistentes

6. Método de Envío

- **Formato**: Archivo ZIP o RAR
- **Nombre del archivo**: ApellidoNombre_GestorAlumnos.zip
- **Tamaño máximo**: 50 MB
- **Plataforma**: [Especificar: Campus Virtual, Email, Google Drive, etc.]
- **Fecha límite**: [A definir por el docente]

7. Checklist Pre-Entrega

Antes de entregar, verificar:

- [ ] El código compila sin errores ni warnings
- [ ] Todas las funcionalidades del menú funcionan
- [ ] Se incluyen archivos de ejemplo
- [ ] El README está completo
- [ ] El manual de usuario está en formato PDF
- [ ] El video está accesible y se reproduce correctamente
- [ ] El nombre del archivo ZIP cumple con el formato solicitado
- [ ] No se incluyen carpetas bin/ y obj/ en el ZIP
- [ ] Se probó el programa en una PC diferente (opcional pero recomendado)

Tiempo Estimado de Desarrollo

- **Duración sugerida**: 2-3 semanas
- **Horas estimadas**: 15-20 horas
- **Fecha de entrega**: 48 horas antes de la presentación a examen, enviar copia del trabajo, se pueden hacer modificaciones de último momento y se reportarán en el examen.

Recomendaciones Finales

1. Comenzar por implementar la clase Alumno y un formato simple (TXT o CSV)

2. Probar cada funcionalidad exhaustivamente antes de pasar a la siguiente
3. Implementar el manejo de excepciones desde el principio
4. Hacer commits frecuentes si se usa control de versiones (Git)
5. Consultar dudas tempranamente, no esperar al último momento
6. Probar con diferentes casos de uso y datos de entrada
7. Comentar el código en español de forma clara y concisa
