Universidad Rafael Landívar

Facultad de Ingeniería

Ingeniería en Informática y sistemas

Manejo e implementación de archivos

Catedrático: Vivian Damaris Del Socorro Campos Gonzáles

***PROYECTO 2.***

Alison Gatica - 1136122

Camilo Solís - 231184

Guatemala, 6 de noviembre del 2024

**Introducción**

El proyecto en C# tiene como objetivo gestionar un sistema de mensajería y contactos, centrándose en la implementación y manejo de archivos secuenciales e indexados. En esta segunda fase, se amplían las funcionalidades desarrolladas previamente, agregando módulos.

La gestión de contactos permite agregar, buscar, modificar y eliminar contactos, almacenando los datos en el archivo secuencial, donde cada registro contiene el usuario, contacto, fecha de transacción, usuario que realizó la transacción y el estado del contacto. La clave primaria de este archivo es la combinación usuario-contacto, también los usuarios pueden gestionar listas de difusión como crearlas, buscarlas, modificarlas y eliminarlas asociadas a un usuario específico. Estas listas se almacenan en el archivo C:/MEIA/lista.txt, el cual incluye el nombre de la lista, el usuario que la creó, una descripción, el número de usuarios, la fecha de creación y el estado de la lista.

La gestión de Usuarios en Listas de Difusión permite asociar usuarios a listas de difusión, almacenando los datos en C:/MEIA/lista\_usuario.txt. La funcionalidad permite el ingreso, búsqueda, modificación y eliminación de usuarios dentro de listas de difusión. Este archivo es secuencial indexado y cada entrada tiene la estructura de registro, posición real, nombre\_lista, usuario, usuario\_asociado, fecha\_creación y estatus.

Cada operación de agregar o eliminar usuarios en listas de difusión debe actualizar el contador de usuarios en el archivo de listas. Además, el proyecto debe manejar adecuadamente errores y excepciones, mostrar botones específicos para cada acción, y cumplir con un análisis detallado de entradas, salidas, procesos y restricciones.

**Interfaz**

**Login Admin info**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente Interfaz de usuario gráfica

Descripción generada automáticamente

**Admin Edit admin**

Interfaz de usuario gráfica

Descripción generada automáticamente Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

**Edit user Search user**

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente Captura de pantalla de un celular

Descripción generada automáticamente

**User info Sign up**

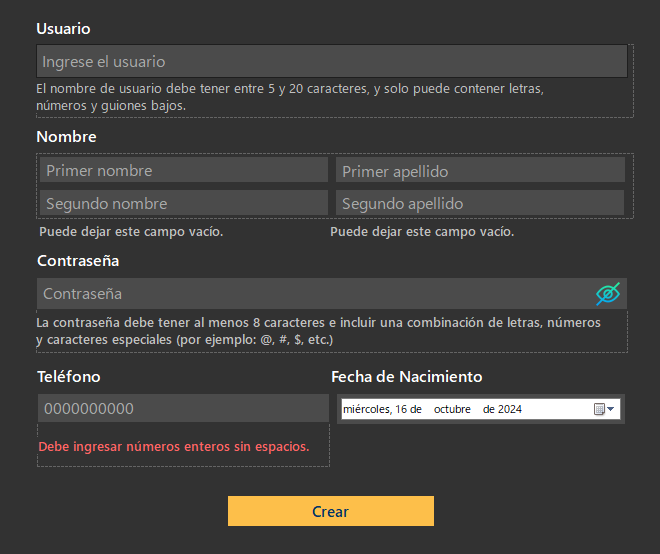
**Interfaz de usuario gráfica

Descripción generada automáticamente Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente**

**User AddUser**

**Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente **

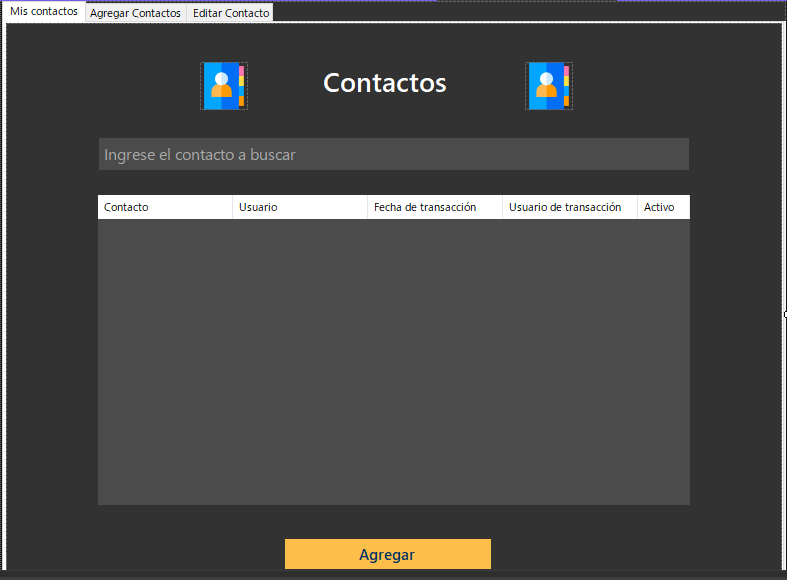
**Lista Crear lista**

**Captura de pantalla de un celular

Descripción generada automáticamente Captura de pantalla de un celular

Descripción generada automáticamente**

**Modificar lista Contactos**

** **

**Agregar contactos Modificar contactos**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**Diagrama**

Diagrama

Descripción generada automáticamente

**Bibliotecas**

**Using MessageNest.Forms;y using MessageNest:**

Estas bibliotecas parecen ser específicas para tu proyecto, posiblemente personalizadas o de un paquete externo llamado MessageNest. Sin el contexto de MessageNest, podría deducirse que proporciona funcionalidades relacionadas con mensajería o comunicaciones y que contiene controles personalizados para formularios de Windows.

**using System:**

Es la biblioteca principal del sistema .NET, proporcionando tipos y funcionalidades básicas, como tipos de datos primitivos, gestión de excepciones, consola, entre otros.

**using System.CodeDom:**

Pertenece al espacio de nombres de System y permite generar y compilar código en tiempo de ejecución. Es útil para crear aplicaciones que generan código fuente dinámicamente, compilan o manejan clases de código de manera programática.

**using System.Collections.Generic:**

Proporciona clases de colección genérica, como List<T>, Dictionary<TKey, TValue>, entre otras. Es esencial para el manejo de colecciones de datos de manera eficiente y segura en cuanto a tipos.

**using System.ComponentModel:**

Proporciona clases que permiten interactuar con los componentes en tiempo de diseño y en tiempo de ejecución, como la implementación de eventos y propiedades enlazables. Es útil en aplicaciones que requieren funcionalidades de notificación de cambios, como formularios de Windows.

**using System.Data:**

Contiene clases para manejar datos y conectarse a bases de datos. Permite manipular tablas, filas y columnas en un contexto desconectado, como el uso de DataTable, DataSet, y DataView.

**using System.Drawing:**

Ofrece funcionalidades para manipular gráficos, imágenes y colores. Esta biblioteca es útil para crear y manipular elementos visuales, como dibujar formas, imágenes y texto en controles de Windows.

**using System.Linq:**

LINQ (Language Integrated Query) permite realizar consultas sobre colecciones y otros tipos de datos. Facilita la filtración, selección y proyección de datos de una manera muy concisa y legible.

**using System.Runtime.InteropServices:**

Proporciona una serie de clases que permiten la interoperabilidad con código no administrado, es decir, llamadas a funciones de bibliotecas externas o sistemas operativos, como las API de Windows.

**using System.Text:**

Incluye clases para trabajar con textos, especialmente para manipular codificaciones de caracteres, como UTF-8 o ASCII, y el uso de StringBuilderpara operaciones de concatenación de cadenas de manera eficiente.

**using System.Threading.Tasks:**

Permite trabajar con tareas asincrónicas y operaciones paralelas, facilitando el manejo de la concurrencia en aplicaciones, lo cual es útil para mejorar el rendimiento en procesos que pueden ejecutarse en paralelo.

**using System.Windows.Forms:**

Proporciona clases para crear aplicaciones de escritorio basadas en formularios de Windows, como botones, cuadros de texto y otros controles gráficos.

**using System.IO:**

Contiene clases para trabajar con archivos y flujos de datos, permitiendo la lectura y escritura de archivos, la creación de directorios y el manejo de archivos y directorios.

**Conclusiones**

1. La implementación de archivos secuenciales e indexados refuerza la importancia de elegir estructuras de almacenamiento adecuadas según los requisitos de acceso y gestión de datos. Este proyecto demuestra como los archivos secuenciales facilitan la integridad y eficiencia en operaciones de almacenamiento de datos que dependen de claves únicas, mientras que el uso de índices en archivos permite un acceso más rápido y organizado para la gestión de listas de difusión.
2. La necesidad de cumplir con funcionalidades como agregar, buscar, modificar y eliminar contactos o usuarios en listas impulsan a una buena programación. Esto facilita la mantenibilidad y escalabilidad del código ya que cada funcionalidad esta bien separada y estructurada en métodos y clases específicas.
3. Un punto importante en el desarrollo de aplicaciones de gestión es la capacidad de prevenir y manejar errores que pueden surgir, como conflictos de claves, operaciones sobre registros inexistentes, y gestión de datos inconsistentes. Este proyecto fomenta una práctica efectiva en la captura y resolución de excepciones, garantizando una experiencia de usuario robusta y confiable.
4. La implementación de restricciones en las clases primarias y en la actualización de registros, como el incremento automático del número de usuarios en las listas de difusión subraya la importancia de mantener la integridad de los datos en todo momento. Esto es crucial para sistemas de mensajería y contactos, donde la precisión de la información y su actualización es tiempo real son esenciales.

**Recomendaciones**

1. Para mejorar la eficiencia, hay que considerar implementar estructuras de búsqueda como árboles o índices hash para archivos de gran tamaño. Esto permitiría la búsqueda y accesos más rápidos, especialmente en el archivo de listas de difusión donde puede haber múltiples registros asociados a un usuario.
2. Implementar un sistema de autenticación y autorización mas robusto para gestionar el acceso a las funciones de creación, modificación y eliminación de registros. Considerando cifrar datos sensibles como contraseñas y otras información personal para proteger la información de los usuarios.
3. Dado que el sistema puede crecer con el tiempo, así que habría que desarrollar un sistema de archivado o compresión para manejar registros antiguos y reducir el tamaño de archivos principales. Esto ayudaría a mejorar el rendimiento
4. Para garantizar la calidad y consistencia del proyecto, se pueden desarrollar una serie de pruebas unitarias y de integración automatizadas que cubran cada funcionalidad principal, como la gestión de contactos y lista de difusión. Esto facilitaría la detección de errores y asegura que los módulos funcionen de manera confiable.

**Manual de usuario**

<https://www.canva.com/design/DAE_C9w_3uQ/UHH_qbadI1TWQE0i93VZcg/edit?utm_content=DAE_C9w_3uQ&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton>

**Referencias**

[**https://learn.microsoft.com/es-es/dotnet/standard/class-library-overview**](https://learn.microsoft.com/es-es/dotnet/standard/class-library-overview)

[**https://learn.microsoft.com/es-es/dotnet/standard/class-libraries**](https://learn.microsoft.com/es-es/dotnet/standard/class-libraries)

[**https://ironsoftware.com/es/news/industry-news/csharp-library/**](https://ironsoftware.com/es/news/industry-news/csharp-library/)