Universidad Rafael Landívar

Facultad de Ingeniería.

Ingeniería en informática y sistemas.

Estructura de Datos – Sección: 01.

Catedrático: MEJÍA ALVARADO RENÉ DANIEL

PROYECTO 1

(LIBRETA DE CONTACTOS)

Julio Sebastián Hernández Batres – 1105824

Guatemala, 25 de abril de 2025

I. Introducción

En el informe documenta el desarrollo de una aplicación de consola en Java denominada Contactos. Esta aplicación fue diseñada como parte del curso de Estructura de Datos I y permite a los usuarios gestionar eficientemente una lista de contactos personales. La aplicación facilita la organización, búsqueda, edición y eliminación de contactos, utilizando estructuras de datos como Árboles Binarios de Búsqueda (BST) y Árboles AVL para optimizar el acceso y manejo de la información.

Primero, empecé creando la clase Contacto, que es donde se guarda toda la información de una persona: nombre, apellido, apodo, teléfono, correo, dirección y fecha de nacimiento. También se le asigna un ID único generado automáticamente por el sistema. A esta clase le apliqué validaciones para que no se puedan ingresar datos con símbolos raros, letras donde van números o fechas de nacimiento que sean mayores a la fecha actual.

Después hice la clase GestorCSV, que se encarga de guardar y cargar los contactos desde un archivo CSV. Esto hace que, aunque apaguemos el programa, los contactos queden guardados. También desde ahí se puede importar o exportar contactos y reconstruir los índices que se usan para las búsquedas. Además, puse mensajes que le avisan al usuario cuando una acción fue realizada correctamente.

Luego, implementé dos estructuras de datos: el Árbol Binario de Búsqueda (BST) y el Árbol AVL. Estas estructuras sirven para organizar los contactos por campos como nombre, apellido o teléfono y así buscar más rápido. Se usan nodos personalizados (NodoBST y NodoAVL) y cada árbol tiene su clase propia con métodos para insertar, buscar y recorrer los datos por niveles.

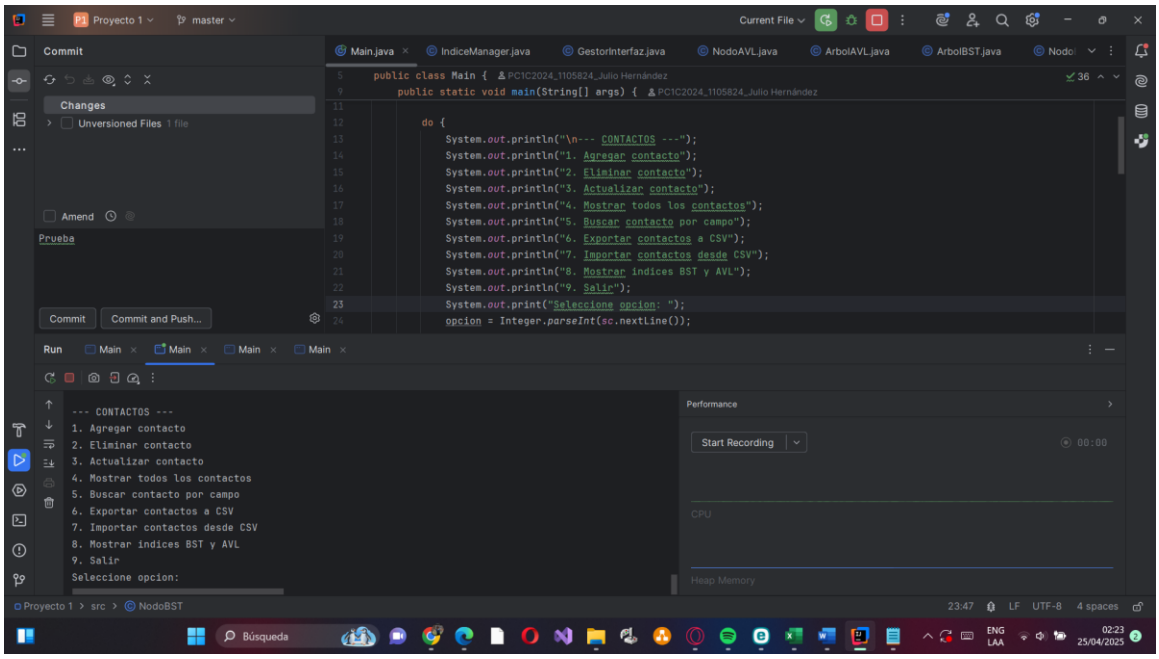
También programé la clase IndiceManager, que se encarga de crear los índices para los árboles. Por ejemplo, si el usuario quiere crear un índice por apellido usando AVL, el sistema genera un archivo llamado apellido-avl.txt con los IDs ordenados por ese campo. Esto sirve para reconstruir el árbol en otra ocasión sin perder el orden.

La clase GestorInterfaz es donde está todo el menú principal. Desde ahí se puede agregar, eliminar, actualizar y buscar contactos por cualquier campo. También se puede importar, exportar, mostrar todos los contactos o crear índices. Este menú está hecho para ser claro, y si el usuario comete un error, el sistema le avisa y le pide volver a intentarlo correctamente.

Por último, está la clase Main, que simplemente ejecuta todo el programa desde consola. Toda la estructura del proyecto está ordenada y dividida por archivos para que sea fácil de entender. El sistema es robusto, modular y cumple con todos los requisitos del proyecto. Puedo demostrar cómo funciona paso a paso porque conozco bien el código y cada parte que implementé.

II. Funcionalidades Principales

A continuación, se describen las funcionalidades clave implementadas en la aplicación:



• Agregar nuevos contactos con campos validados rigurosamente.

• Eliminar contactos existentes en cualquier momento, mediante su ID.

• Actualizar la información de un contacto, eligiendo qué campos editar.

• Buscar contactos usando distintos criterios: nombre, apellido, apodo, o número de teléfono.

• Mostrar todos los contactos guardados en la libreta.

• Exportar los contactos a un archivo CSV para respaldo externo.

• Importar contactos desde archivos CSV ya existentes.

• Crear y visualizar índices en estructuras BST o AVL según el campo seleccionado.

III. Validaciones Implementadas

Se implementaron múltiples validaciones para asegurar la integridad de los datos ingresados por el usuario:

• Nombre y apellido: deben contener únicamente letras. No se aceptan números ni símbolos especiales.

• Apodo: puede contener letras y números, pero no símbolos especiales.

• Número de teléfono: debe tener exactamente 8 dígitos numéricos. No se permiten letras ni símbolos.

• Fecha de nacimiento: debe ingresarse en formato DD-MM-AAAA. No puede ser posterior a la fecha actual.

IV. Estructuras de Datos

Para indexar eficientemente los contactos, se implementaron dos tipos de estructuras de datos avanzadas:
• Árbol Binario de Búsqueda (BST): se utiliza para realizar búsquedas más rápidas en comparación con listas simples.
• Árbol AVL: una versión balanceada del BST que garantiza tiempos óptimos de inserción, eliminación y búsqueda.

Los índices generados se almacenan en archivos de texto separados según la estructura utilizada y el campo indexado (por ejemplo, 'apellido-avl.txt' contiene el recorrido por niveles de un árbol AVL basado en los apellidos).

V. Importación y Exportación de Datos

La aplicación permite guardar los contactos en un archivo CSV denominado 'contacts.csv', y también importar nuevos contactos desde archivos CSV externos. Esto permite mantener respaldos o cargar grandes volúmenes de datos de forma sencilla. Al completar cada operación, se muestra un mensaje confirmando su éxito.

VI. Conclusión

La aplicación Contactos cumple con todos los objetivos establecidos. A través del uso de estructuras de datos eficientes, validaciones estrictas, modularidad en el código y una interfaz clara para el usuario, se logró construir una herramienta robusta, útil y educativa. Este proyecto fortalece los conocimientos en programación orientada a objetos, manipulación de archivos, y el diseño de estructuras de datos en Java.