Universidad Rafael Landívar
Facultad de Ingeniería
Ingeniería en Sistemas
Estructura de datos I
Ing. Mejía Alvarado René Daniel

PROYECTO NO. 01
“LIBRETA DE CONTACTOS“

Monje Pérez, Javier Enrique
Carné No.1260524
Avila Mazariegos, Gerardo Javier
Carné No. 2002824

Guatemala 25 de abril de 2025

ANÁLISIS Y DISEÑO

Para cumplir con los requisitos y objetivos del programa, a continuación, se listan las principales entradas procesos y salidas del programa.

ENTRADAS

Crear Contacto

El programa creará el objeto contacto con los siguientes atributos.

- String Nombre: cadena de texto que guardará el nombre del contacto.
- String Apellido: cadena de texto que guardará el apellido del contacto.
- String Apodo: cadena de texto que guardará el apodo del contacto.
- String NumeroTelefono: cadena de texto que guardará el número de teléfono, se utilizará string, para facilitar el proceso de guardado de números, y facilitar restricciones.
- String email: cadena de texto que guardará el correo electrónico del usuario.
- String dirección: cadena de texto que almacene la dirección del usuario.
- LocalDate fechaNacimiento: almacenamiento de la fecha de nacimiento del contacto.
- Int ID: Variable tipo entero que almacena el id de los contactos. Generado automáticamente por el programa, según el orden en que se vayan agregando distintos contactos.

Otras Entradas

- Int opciones: Variable tipo entero que funcionará para navegar en los menús y opciones del programa.

PROCESOS

Crear árbol

En este apartado el usuario podrá elegir de qué manera querrá guardar sus contactos, para ello, habrá distintas formas de poder almacenarlos. Para cada atributo de los distintos contactos habrá un árbol AVL, AVL para facilitar la búsqueda en distintos procesos como agregar o eliminar contactos. Los árboles se compararán según el tipo de variable. Si el usuario ya escogió la opción de guardar sus contactos de alguna forma, no puede hacer un árbol nuevamente, es decir, si el usuario ya tiene un árbol que ordene sus contactos por nombre, no puede crear otro árbol de nombres, más sí podrá crear otro árbol que almacene sus contactos por algún otro atributo.

Si ya existen árboles y el usuario agrega uno, por medio de una travesía se obtendrán los contactos existentes almacenados previamente en otro árbol, y el programa irá guardándolos en el nuevo árbol creado según el criterio de ordenamiento.

Añadir contactos

Según los contactos agregados y árboles creados, los contactos se irán añadiendo a los árboles AVL, almacenándose.

Eliminar contactos
Se buscará el contacto que se desea eliminar, el usuario deberá colocar el ID del contacto, se buscará y encontrará en el contacto, y se eliminará en todos los árboles existentes.

Actualizar contacto
Se solicitará al usuario que agregue la información que desee cambiar del contacto, de esta forma se encontrará el contacto existente y se eliminará, se agregará el nuevo contacto con la información nueva utilizando la misma lógica de los procesos de añadir y eliminar contactos.

Búsqueda de contactos
Según el criterio de comparación que el usuario escoja, se hará una búsqueda en el árbol para encontrar coincidencias con el contacto que el usuario haya escogido.

Visualización de todos los contactos
Según el criterio de comparación que el usuario escoja, se hará una búsqueda Inorder de los contactos y se le mostrarán al usuario.

SALIDAS

- Mensajes de confirmación: se muestra en consola despues de realizar cada acción (agregar, eliminar y actualizar contactos)
- Resultado de búsquedas: se muestra en consola de la información del contacto encontrado por su respectivo id o cualquiera de los campos indexados (nombre, apellido, teléfono, entre otros)
- Listado de contactos: muestra todos los contactos que se almacenan en “contacts.csv”
- Mensajes de validación: muestra e informa al usuario sobre errores o restricciones.
- Información en forma CSV: Generación de archivos de índice “apellido-avl.txt” contenido de los archivos CSV

RESTRICCIONES

- ID único y generado automáticamente: Cada contacto debe tener un ID único el cual no puede ser ingresado manualmente por el usuario y antes de añadir algún otro ID se deben comprobar que no exista. Este ID debe ser generado automáticamente por el programa.
- Estructura de datos: Implementar arboles BST y AVL manualmente, sin uso de librerías o clases de terceros, estas estructuras se desarrollarán por el programador.
- Evitar duplicados: El programa debe de evitar que se ingresen dos campos iguales como “teléfono, correo electrónico” ya que esto indicaría que hay contactos duplicados.
- Validación: Cualquier excepción del programa debe manejarse con mensajes para que el programa no se detenga.

Diseño (PENDIENTE A CONFIRMACIÓN)

III. Preguntas

- ¿El usuario puede cambiar el id?
  No se puede actualizar este campo, pero si se pueden los demás.

- ¿Pide AVL y BST, pero como el AVL ya es BST, si solo se hace AVL estamos cumpliendo los dos requisitos?
  La persona debe de escoger que tipo de árbol puede escoger

- ¿El usuario debe crear sus árboles a gusto para ordenar, o estos ya deben estar generados al iniciar el programa?
  Si

- ¿Al abrir el programa el usuario solo debe tener un árbol que guarde sus contactos de la forma que él quiera, o debe tener varios?

- ¿Debemos hacer diagrama de flujo?

- ¿Índices?

- ¿Por nivel o se puede usar Inorder?

  Puede buscar por apellidos
  Los Id solo representan en los arboles BST-txt