Método de la Ingeniería

Integrantes:

1. Valentina Arias Parra
2. José Luis Restrepo Obando
3. Alejandro Saldarriaga

Fase 1: Identificación del problema.

- Contexto problemático (Síntomas y necesidades)

  Desarrollo de un prototipo de software que permita gestionar eficientemente las operaciones de crear, leer, actualizar y eliminar, en una base de datos de personas del continente americano. Teniendo en cuenta que el continente tiene aproximadamente un poco más de mil millones de personas hasta el 2020. Además, que cada persona debe tener código, nombre, apellido, sexo, fecha de nacimiento, estatura, nacionalidad y fotografía. Una vez se hayan generado los datos, deben guardarse en la base de datos del programa, por lo tanto, debe haber persistencia.

  En cuanto a la operación de crear personas todo debe ser diligenciado menos el código que se genera automáticamente. Para actualizar todos los campos deben ser editables menos el del código, y además debe incluir la opción de eliminar. Para la operación buscar se debe hacer por 4 criterios y el programa debe permitir que a media que se digiten los caracteres de búsqueda, aparezca una lista emergente, que empiece con los caracteres digitados hasta el momento.

- Definición del problema

  Gestionar eficientemente las operaciones CRUD (Create, Read, Update y Delete) sobre una base de datos de personas del continente americano que son alrededor de mil millones de personas.

- Requerimientos funcionales

  - **RF01.** Generar un registro de la cantidad de personas que desee el usuario con un máximo de mil millones de personas.

  - **RF02.** Generar el código de forma aleatoria para cada persona.

  - **RF03.** Generar la fecha de nacimiento de forma aleatoria para cada persona.

  - **RF04.** Generar de forma aleatoria la estatura de cada persona.

  - **RF05.** Indicar cuanto tiempo se demora la generación de personas, y mostrar el proceso en una barra de progreso.

  - **RF06.** Los datos del programa deben ser persistentes.

  - **RF07.** El programa debe permitir agregar personas.

  - **RF08.** Buscar a una persona por el nombre.

  - **RF09.** Buscar una persona por el Apellido.

  - **RF10.** Buscar una persona por el nombre completo (nombre + apellido)

  - **RF11.** Buscar una persona por el código.

  - **RF12.** Actualizar los campos de una persona existente, permitiendo editar toda la información menos el código.

  - **RF13.** Eliminar personas.

- **RF14.** El programa debe sugerir opción para autocompletar al momento de buscar.

Fase 2: Conceptos importantes para modelar el problema.

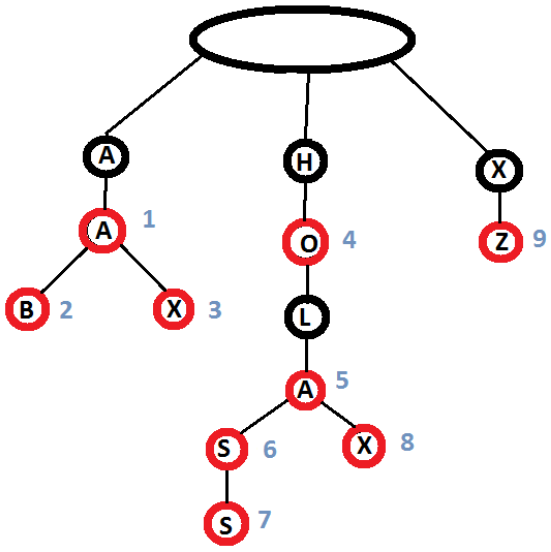
- Marco teórico

Estructura de datos:

Son una forma de almacenar y ordenar información estructuradamente con el fin de realizar sobre estas distintas operaciones de manera eficiente. Existen distintos tipos de estructuras de datos que son útiles de acuerdo con el contexto del problema. Por ejemplo, un AVL Tree, un Binary Search Tree (BST) o un Trie.

- **AVL Tree:** Un árbol AVL es un tipo especial de árbol binario ideado por los matemáticos rusos Adelson-Velskii y Landis. Fue el primer árbol de búsqueda binario auto-balanceable que se ideó.
- **Binary Search Tree (BST):** Un árbol binario de búsqueda (BTS) es un árbol binario con la propiedad de que todos los elementos almacenados en el subárbol izquierdo de cualquier nodo x son menores que el elemento almacenado en x ,y todos los elementos almacenados en el subárbol derecho de x son mayores que el elemento almacenado en x.
- **Trie:** es una estructura de datos que se utiliza en strings. Se puede utilizar para realizar búsquedas de manera óptima, con una complejidad casi lineal (dependiendo de la implementación y las restricciones del problema). La idea del Trie es armar un árbol, donde los nodos son las letras de las palabras. Las letras que están en rojo indican los nodos donde termina una palabra. Es decir, si recorremos el árbol desde la raíz hasta ese nodo, formaremos una de las palabras iniciales. Si lo recorremos en Preorden, podemos ordenar alfabéticamente a las palabras que forman el Trie. Los números escritos expresan el orden alfabético de las palabras.

hola
aab
holas
holass
xz
aa
aax
holax
ho



- Árbol rojo-negro: Un árbol rojo-negro es un tipo abstracto de datos. Concretamente, es un árbol binario de búsqueda equilibrado, una estructura de datos utilizada en informática y ciencias de la computación. La estructura original fue creada por Rudolf Bayer en 1972, que le dio el nombre de “árboles-B binarios simétricos”, pero tomó su nombre moderno en un trabajo de Leo J. Guibas y Robert Sedgewick realizado en 1978. Es complejo, pero tiene un buen peor caso de tiempo de ejecución para sus operaciones y es eficiente en la práctica. Puede buscar, insertar y borrar en un tiempo O(log n), donde n es el número de elementos del árbol.
  Un árbol rojo-negro es un árbol binario de búsqueda en el que cada nodo tiene un atributo de color cuyo valor es rojo o negro. En adelante, se dice que un nodo es rojo o negro haciendo referencia a dicho atributo.

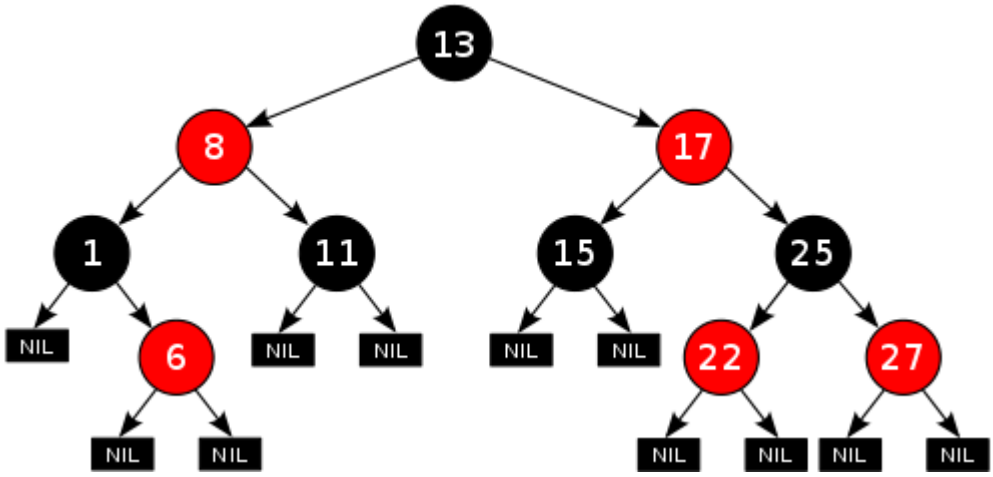
  Además de los requisitos impuestos a los árboles binarios de búsqueda convencionales, se deben satisfacer las siguientes reglas para tener un árbol rojo-negro válido:

  1. Todo nodo es o bien rojo o bien negro.
  2. La raíz es negra.
  3. Todas las hojas (NULL) son negras.

4. Todo nodo rojo debe tener dos nodos hijos negros.

Cada camino desde un nodo dado a sus hojas descendientes contiene el mismo número de nodos negros.

Estas reglas producen una regla crucial para los árboles rojo-negro: el camino más largo desde la raíz hasta una hoja no es más largo que dos veces el camino más corto desde la raíz a una hoja. El resultado es que dicho árbol está aproximadamente equilibrado.



Base de datos:

Es una herramienta que recopila, organiza y relaciona datos de una organización para poder acceder a ellos de forma eficiente y segura. Las bases de datos son ampliamente usadas en la industria para almacenar todo tipo de información, desde datos de usuarios, hasta pedidos, cuentas, entre otros. Entre los tipos de bases de datos, se encuentran la base de datos orientadas a objetos, la cual almacena los datos en forma de objetos y clases.

CRUD:

Está estrechamente vinculado a la gestión de datos digitales. CRUD hace referencia a un acrónimo en el que se reúnen las primeras letras de las cuatro operaciones fundamentales de aplicaciones persistentes en sistemas de bases de datos:

- **Create**: Crear registros
- **Read bzw. Retrieve:** Leer registros

- **Update:** Actualizar registros
- **Delete bzw. Destroy:** Borrar registrosg

Fuentes:

- Sites.google.com. 2020. *Arboles AVL - Estructuras De Datos En Java*. [online] Available at: https://sites.google.com/a/espe.edu.ec/programacion-ii/home/arboles/arboles-avl
- Decsai.ugr.es. 2020. *ARBOLES BINARIOS DE BUSQUEDA*. [online] Available at: http://decsai.ugr.es/~jfv/ed1/c++/cdrom4/paginaWeb/abb.htm
- OpenWebinars.net. 2020. *¿Qué Son Las Estructuras De Datos Y Por Qué Son Útiles?*. [online] Available at: https://openwebinars.net/blog/que-son-las-estructuras-de-datos-y-por-que-son-tan-utiles/
- TIC Portal. 2020. *Base De Datos : ¿Qué Tipos Hay Y Cómo Funciona Conectada A Un Software?*. [online] Available at: https://www.ticportal.es/glosario-tic/base-datos-database
- Es.wikipedia.org. 2020. *CRUD*. [online] Available at: https://es.wikipedia.org/wiki/CRUD
- Árbol rojo-negro. Es.wikipedia.org. (2020). Retrieved 7 November 2020, from https://es.wikipedia.org/wiki/%C3%81rbol_rojo-negro.

Fase 3: Búsqueda de soluciones creativas

1. **Idea 1:** Diseñar e implementar un software que utilice como estructura de datos un **ArrayList** que almacene los usuarios generados aleatoriamente y permita realizar las operaciones CRUD sobre la base de datos, siendo persistente por medio de serializable.

2. **Idea 2:** Diseñar e implementar un software que utilice una **LinkedList** y sea capaz de generar registros en base a lo requerido y tenga persistencia que aplique las operaciones CRUD.

3. Idea 3: Diseñar e implementar un software que utilice como estructura de datos un **Array** de tamaño definido que almacene los usuarios generados aleatoriamente y permita realizar las operaciones CRUD sobre la base de datos, para después realizar serialización mediante archivos de texto

4. Idea 4: Diseñar e implementar un software que por cada usuario generado lo haga persistente en un **archivo de texto** de tal forma que cuando se cree un usuario inmediatamente se escriba en el archivo y cuando se vayan a realizar las operaciones CRUD ingrese al archivo busque el usuario y modifique su campo.

5. Idea 5: Diseñar e implementar un software que utilice como estructuras de datos **árboles** y poder realizar las operaciones CRUD en la base de datos. al mismo tiempo generar búsquedas de usuarios y cuando los resultados sean menores o iguales a 20 desplegar un menú de sugerencia de posibles usuarios buscados.

6. Idea 6: Diseñar e implementar un software que utilice como estructura de datos un **Trie** que almacene los usuarios generados aleatoriamente, sea persistente mediante archivos de texto y aplique las operaciones CRUD sobre la base de datos.

Fase 4: Diseños preliminares.

Ideas descartadas: En primer lugar, tenemos que descartar las alternativas 1, 3, 4 y dado que estas son poco eficientes y no cumplen con las funcionalidades necesarias para el funcionamiento completo del programa. Por tanto, se hace imposible el uso de alguna de estas alternativas. Además, las otras tres opciones se pueden considerar más eficientes.

Ideas posibles:

- Idea 2: LinkedList

  En esta alternativa estaríamos usando una LinkedList que es una estructura de datos lineal. El tiempo requerido para realizar las operaciones puede llegar a ser muy extenso. Entonces difícilmente puede llegar a cumplir con todos los requerimientos planteados anteriormente.

- Idea 5: Arboles

  A esta alternativa la podríamos enforcar en arboles AVL, BST, que son estructuras de datos eficientes y facilitan la realización de las diferentes operaciones en un menor tiempo.

- Idea 6: Trie

  Usando la estructura de datos Trie. Se puede utilizar para realizar búsquedas de manera óptima, con una complejidad casi lineal, es una muy bueno opción para realizar operaciones CRUD pero no tenemos mucho conocimiento acerca de la implementación.

Fase 5: Evaluación y selección de soluciones.

Rúbrica de evaluación

#	Estructura	Precisión de la solución		Eficiencia		Completitud		Facilidad de implementación		TOTAL
		25%		25%		25%		25%		100%
2	**LinkedList**	Esta alternativa entrega una precision aproximada	**4**	Puede almacenar grandes volumenes de información pero el accder a esta no es eficiente	**3**	No cumple con todos los requerimientos del programa	**3,5**	Su implementación es sencilla lo que facilita la completa implementación del software	**4,5**	**3,75**
5	**Arboles (AVL, BST)**	Esta alternativa entrega una precision exacta	**5**	Puede almacenar grandes volumenes de información y se puede accder a ellos de manera eficiente debido a los algoritmos de balanceo. Com: O(mlogn)	**4,5**	Permite cumplir con todos los requeirmiento sy funcionalidades del programa	**5**	Cumple con todas la soluciones y a la vez se facilita la implementación del software	**5**	**4,875**
6	**Trie**	Esta alternativa entrega una precision exacta	**5**	Almacena la información de tal manera que puede cumplir con el requerimiento de autocompletado precisamente. Com: O(m)	**5**	Cumple con los requerimientos y funcionalidades del programa	**5**	Cumple con las soluciones sin embargo, su implementación requiere de mas esfuerzo y analisis que las otras estructuras	**4**	**4,75**