Método de Ingeniería

1. **Identificación del Problema:**

   En este punto se identifica el problema y las necesidades que se deben satisfacer a lo largo del proyecto:

   - Se debe permitir gestionar operaciones CRUD sobre una base de datos de personas.
   - Se deben crear un número de personas similares o equivalentes a la población total del continente americano.
   - La generación de los datos de una persona debe hacerse a partir de datasets proporcionados por el equipo de simulación.
   - La solución del problema debe tener persistencia en sus datos.
   - La solución del problema debe implementar pruebas unitarias automáticas de las estructuras de datos.
   - Para la solución se deben implementar arboles recursivos ABB y AVL.

   Interfaz Gráfica:

   - La solución del problema debe implementar en la interfaz formularios para cada una de las operaciones CRUD.
   - La interfaz de la solución debe tener un menú para cambiar los elementos de la ventana

2. **Recopilación de información:**

   Con el objetivo de tener conocimientos mas íntegros acerca de la problemática y aprender cómo implementar una solución, se hizo una búsqueda a fondo de las estructuras de datos genéricas que se deben usar.

   “Los árboles AVL están siempre equilibrados de tal modo que, para todos los nodos, la altura de la rama izquierda no difiere en más de una unidad de la altura de la rama derecha o viceversa. Gracias a esta forma de equilibrio (o balanceo), la complejidad de una búsqueda en uno de estos árboles se mantiene siempre en orden de complejidad O(log n).” … “El factor de equilibrio puede ser almacenado directamente en cada nodo o ser

computado a partir de las alturas de los subárboles.” … “Para conseguir esta propiedad de equilibrio, la inserción y el borrado de los nodos se ha de realizar de una forma especial. Si al realizar una operación de inserción o borrado se rompe la condición de equilibrio, hay que realizar una serie de rotaciones de los nodos.”

Fuentes:

- T.H Corner. Binary Search Trees. En Introduction to Algorithms. Capítulo 12, páginas 286-299. Tercera edición, 2009.
- https://en.wikipedia.org
- https://visualgo.net/en/bst
- https://www.geeksforgeeks.org/
- https://www.youtube.com/
- https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

3. **Búsqueda de soluciones creativas:**

Para este paso, se decidió hacer una lluvia de ideas para plantear todas las ideas posibles que pueden dar solución al problema. Los resultados que surgieron de la lluvia de ideas fueron:

a) Manejar los datos con ArrayList.
b) Implementar un árbol AVL.
c) Implementar un árbol binario no balanceable.
d) Implementación de un árbol roji-negro.

4. **Transición de la formulación de ideas a los diseños preliminares:**

En este paso vamos a evaluar las ideas que fueron planteadas en el anterior punto que podrían dar solución al problema planteado. Primero se descartarán las ideas que no son factibles. Descartaremos manejar los datos con un ArrayList debido a que este tipo de dato tiene un alto peso en memoria, por lo que manejar tantos datos como la población de toda América no se podría realizar.

La revisión de las otras alternativas nos conduce a los siguiente:

- Alternativa 1. Árbol AVL :
    - Tiene una complejidad alta a la hora de la búsqueda de datos dentro de la estructura y su implementación es costosa.
- Alternativa 2. Árbol Binario no Balanceable(ABB):
    - La búsqueda se complicaría, ya que esta alternativa se demoraría mucho tiempo en la búsqueda al no estar balanceado.
- Alternativa 3. Árbol Roji-Negro:
    - Esta alternativa supone una mayor complejidad a la hora de implementación.
    - Debido a sus condiciones de color, esta alternativa también supone una complejidad alta a la hora de buscar.

5. Evaluación y Selección:

Criterios:

Para la selección de la solución final tendremos en cuenta los siguientes criterios:

- Criterio A. Suplencia de Requerimientos:
    -[2] Cumple con todos los requerimientos.
    -[1] Incumple algún requerimiento.
- Criterio B. Complejidad general de la implementación.
    -[3] Complejidad Estándar.
    -[2] Complejidad Media Alta
    -[1] Complejidad Muy Alta.
- Criterio C. Complejidad en el método de búsqueda.
    -[3] Complejidad Estándar.
    -[2] Complejidad Media Alta
    -[1] Complejidad Muy Alta.
- Criterio D. Velocidad de búsqueda.
    -[3] Velocidad Alta
    -[2] Velocidad Media
    -[1] Velocidad Baja

	Criterio A	Criterio B	Criterio C	Criterio D
Alternativa 1. Árbol AVL.	Cumple con todos los requerimientos. 2	Complejidad Media Alta 2	Complejidad Media Alta 2	Velocidad Alta 3
Alternativa 2. Árbol Binario no Balanceable(ABB	Cumple con todos los requerimientos. 2	Complejidad Estándar 3	Complejidad Media Alta 2	Velocidad Baja 1
Alternativa 3. Árbol Roji-Negro	Cumple con todos los requerimientos. 2	Complejidad Muy Alta 1	Complejidad Muy Alta 3	Velocidad Media 2

Selección: De acuerdo con la evaluación anterior debemos tomar la alternativa 1 ya que obtuvo la mayor puntuación, y por lo tanto es la más viable.

6. **Preparación de Informes y Especificaciones:**

   Especificación del Problema:

   Problema: Gestión de operaciones CRUD sobre una base de datos.

   Entrada: *Para Crear y Actualizar:*

   - Nombre
   - Apellido
   - Edad
   - Fecha de Nacimiento
   - Sexo
   - Altura
   - Nacionalidad

   Salidas: Persona creada y añadida al árbol.

7. Implementación del diseño:

Implementación en java.

Lista de Tareas para implementar:

a. Generar los datos
b. Leer los datos
c. Actualizar los datos
d. Eliminar los datos

Especificación de Subrutinas:

a)

Nombre:	generateData
Descripción:	Método donde se crean el número de datos indicados por el usuario.
Entrada:	Número de datos a crear(amountTF).
Salida:	Duración total de la creación de los datos.

b)

Nombre:	keyTyped
Descripción:	Lee letra por letra lo que el usuario está escribiendo y en base al criterio de búsqueda elegida, busca a las personas.
Entrada:	El dato por el cual se está buscando a una persona(nombre, apellido, nombre completo o código)
Salida:	Devuelve la persona que encaje con las coincidencias

c)

Nombre:	remove
Descripción:	El método pide los datos que se quieran actualizar de la persona, el usuario los introduce y se actualizan automáticamente.
Entrada:	El dato que se quiere actualizar.
Salida:	Datos de la persona actualizados

d)

Nombre:	remove
Descripción:	El metodo eliminar una persona que está creada en el árbol
Entrada:	Dato por el cual se encontrará a la persona
Salida:	Persona eliminada del arbol