Jhon Stiven Arboleda, Juan Sebastián Barrera, Alejandro García

PHASE 1:

1. **Identification of the problem**

   We want to develop a software prototype that allows you to manage CRUD operations in a database on the American continent.

   Needed:
   - ➢ The simulation of creating a close number of records of the population of the American continent for 2020 (around one billion people).

   Problem: Efficient database storage, each record must be editable and removable.

2. Research

Functional requirements:

1. Add a new user to the database. The following data must be taken into account: code, name, surname, sex, date of birth, height, nationality and photograph.
2. Update a person's record in the database. All user information must be editable, except for the code that is automatically generated.
3. Delete a person's record from the database.
4. Search for a person in the database by any criteria name, surname, full name (name + surname) and code.
5. Generate a database with user information with an amount close to the Latin American population

PHASE 2:

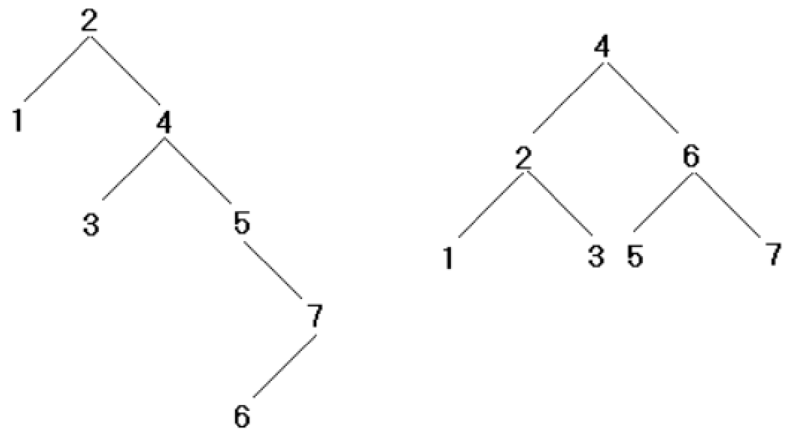
CRUD: In programming, **c**reate, **r**ead, **u**pdate and **d**elete (with the acronym CRUD) are the four basic functions of database persistence. Alternative terms are sometimes used when defining the four basic CRUD functions, such as "retrieve" instead of "read", "modify" instead of "update" or "destroy" instead of "delete". CRUD is also sometimes used to describe user interface conventions that make it easy to view, search, and modify information; often used in form programming (forms) and reports (reports)

BST: A **B**inary **S**earch **T**ree satisfies the condition that the left subtree of any node (if it is not empty) contains smaller values than the one that contains that node, and the right subtree (if it is not empty) contains larger values.

For these definitions, it is considered that there is an established order relationship between the elements of the nodes. Whether a certain relationship is defined or not depends on each programming language. From this it follows that there can be different binary search trees for the same set of elements.

The figure shows 2 BSTs built on the same set of integers:

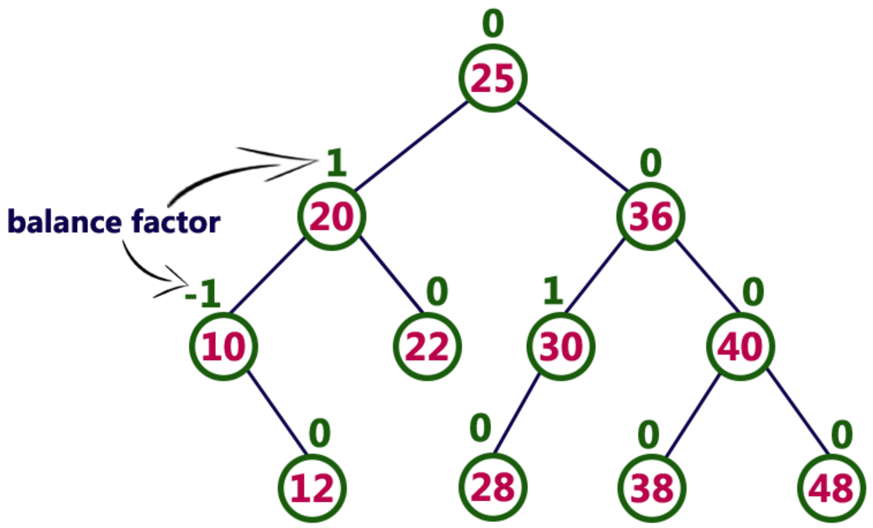
BST example:



AVL: The AVL tree takes its name from the initials of the surnames of its inventors, Georgii Adelson-Velskii and Yevgeniy Landis. They made it known in the publication of an article in 1962, "An algorithm for the organization of information" ("An algorithm for the organization of information").

AVL trees are always balanced in such a way that for all nodes, the height of the left branch does not differ by more than one unit from the height of the right branch or vice versa. Thanks to this form of balance (or balancing), the complexity of a search in one of these trees is always kept in order of complexity O (log n). The balance factor can be stored directly at each node or computed from the heights of the subtrees.

AVL example:



RedBlack Tree: A red-black tree is an abstract type of data. Specifically, it is a balanced binary search tree

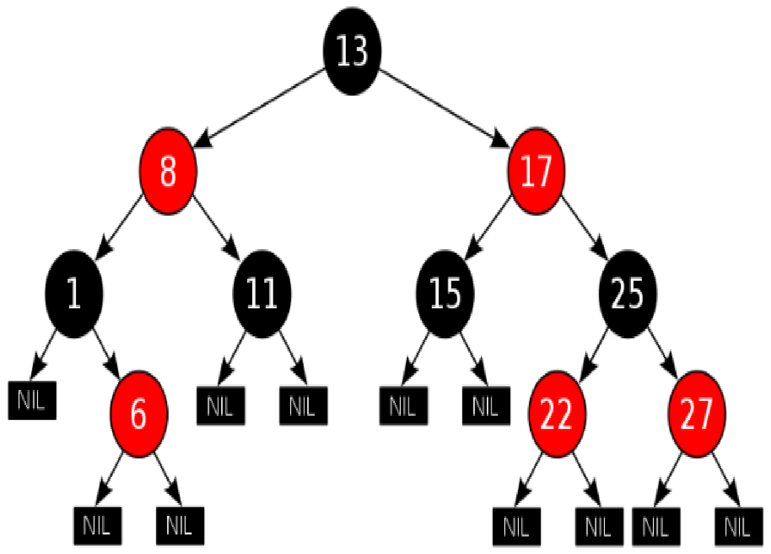
A red-black tree is a binary search tree in which each node has a color attribute whose value is red or black. From now on, a node is said to be red or black referring to that attribute.

In addition to the requirements imposed on conventional binary search trees, the following rules must be satisfied to have a valid red-black tree:

- ✓ Every node is either red or black.
- ✓ The root is black.
- ✓ All leaves (NULL) are black.
- ✓ Every red node must have two black child nodes.

- ✓ Each path from a given node to its descendant leaves contains the same number of black nodes.

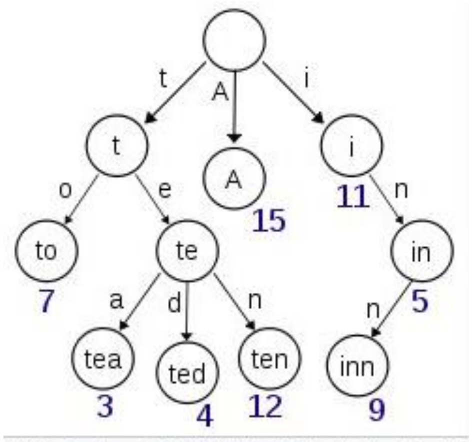
RedBlack Tree example:



Trie: A "trie" is a tree-like data structure that allows the retrieval of information (hence its name from reTRIEval). The information stored in a trie is a set of keys, where a key is a sequence of symbols belonging to an alphabet. The keys are stored in the leaves of the tree and the internal nodes are gateways to guide the search. The tree is structured in such a way that each letter of the key is placed in a node so that the children of a node represent the different possibilities of different symbols that can continue to the symbol represented by the parent node.

Search in the Trie: It begins at the root of the tree. If the symbol, we are looking for is A then the search continues in the subtree associated with the symbol A hanging from the root. It continues in the same way until reaching the leaf node. Then the string associated with the leaf node is compared and if it matches the search string then the search has ended in success, if not then the element is not found in the tree.

Trie example:



A trie for keys "A","to", "tea", "ted", "ten", "i", "in", and "inn".

REFERENCES:

BST definition:

https://es.wikipedia.org/wiki/Árbol_binario_de_búsqueda

Tree AVL definition.

https://sites.google.com/a/espe.edu.ec/programacion-ii/home/arboles/arboles-avl

Trie definition.

https://es.wikipedia.org/wiki/Trie

Tree RedBlack definition

https://es.wikipedia.org/wiki/Árbol_rojo-negro

CRUD definition

https://es.wikipedia.org/wiki/CRUD

PHASE 3:

Brainstorming:

Alternative 1: Generate the database from a page for random data generation. The data (users) are grouped in the "Trie" data structure.

Alternative 2: Data collection from existing surveys and databases. The data (users) are grouped in the arrayList data structure.

Alternative 3: Generate the required amount of data from different pages. The data (users) are grouped in the AVL data structure.

Alternative 4: Use an existing database. The data (users) are grouped in the BST data structure.

Alternative 5: The user must enter the required data. The data (users) are grouped in the AVL data structure.

Alternative 6:

PHASE 4:

IDEAS DISCARDED

Alternative 2: Taking surveys is too expensive and time consuming. The arrayList is not a good data structure for the amount of data to store.

Alternative 5: It is too time consuming to enter the amount of data required.

Alternative 6:

SHORTLISTED IDEAS:

Alternative 1: Generate the database from a page for random data generation. The data (users) are grouped in the "Trie" data structure. The data can be generated randomly from this page: http://www.generatedata.com/?lang=es. Four "trie" trees are created to perform each of the requested searches (Name, surname, full name <Name + "" + Surname> code)

Alternative 3: Generate the required amount of data from different pages. The data (users) are grouped in the AVL data structure. The data are generated from different criteria such as the following:

- ✓ https://www.indexmundi.com/es/estados_unidos/distribucion_por_edad.html for age.
- ✓ https://www.kaggle.com/tanuprabhu/population-by-country-2020 for the distribution of the number of people by country
- ✓ https://data.world/alexandra/baby-names for the names
- ✓ https://data.world/uscensusbureau/frequently-occurring-surnames-from-the-census-2010 for surnames

For each search to be carried out (by name, surname, full name or code) an AVL tree is implemented

Alternative 4: Use an existing database. The data (users) are grouped in the BST data structure. Search for databases on the web that are quite extensive, which meet all the requirements to complete the database. To effectively search for each criterion, 4 different ABBs will be implemented.

PHASE 5:

Criterion A: Efficiency in the implemented data structure

[3 points]: Very time efficient.
[2 points]: Little efficient in time
[1 points]: Nothing efficient in time

Criterion B: Structure storage capacity

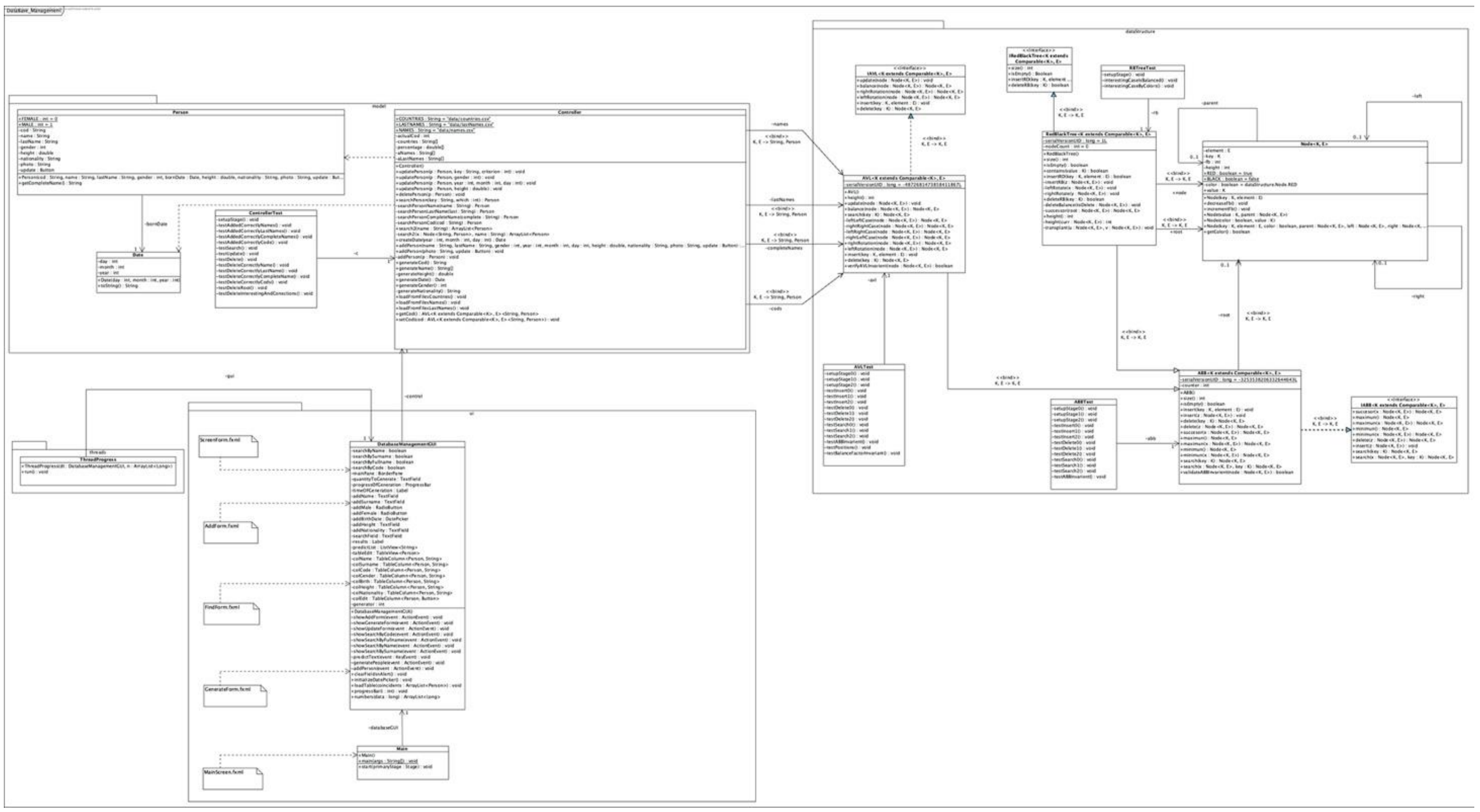
- **[3 points]** Good capacity. No problem storing large amounts of data
- **[2 points]** Normal capacity. It can cause some problems by having too much data.
- **[1 points]** Not good for storing inaccurate amounts of data

Criterion C: Access to get the data to be generated

- **[3 points**] It is easily accessible and the amount necessary to use
- **[2 points]** It is easily accessible but the amount to be obtained is not enough
- **[1 points]** Not easy to get and not enough

SOLUTION	**Criterion A**	**Criterion B**	**Criterion C**	**Total points**
Alternative 1	2	3	2	7
Alternative 3	3	3	3	9
Alternative 4	2	3	1	6

Best alternative: 3


Database_Management
model
Person
Date
ControllerTest
Controller
-bornDate
-names
-lastNames
-completeNames
-cods
-gui
-control
threads
ThreadProgress
ui
DatabaseManagementGUI
ScreenForm.fxml
AddForm.fxml
FindForm.fxml
GenerateForm.fxml
MainScreen.fxml
Main
-databaseGUI
dataStructure
«interface» IAVL<K extends Comparable<K>, E>
AVL<K extends Comparable<K>, E>
AVLTest
«interface» IRedBlackTree<K extends Comparable<K>, E>
RBTreeTest
RedBlackTree<K extends Comparable<K>, E>
Node<K, E>
ABB<K extends Comparable<K>, E>
ABBTest
«interface» IABB<K extends Comparable<K>, E>
«bind» K, E -> String, Person
«bind» K, E -> K, E
-parent
-left
-right
-root
-rb
-abb
-avl
0..1