Alejandro Fonseca Forero
Diego Alejandro Garcia Cardona

Functional Requirements:

Req 1: Generate the records of a certain number of people with the following data: Code (Autogenerated), name, last name, gender, birthdate, height, nationality and photograph. The maximum number of records that can be generated will be 1 million people. In addition, if the generation takes more than 1 second, a progress bar is displayed on the screen. The generation time will be displayed at the end of the process.
Req 2: Add a person's record with all of its data: Name, last name, gender, birthdate, height and nationality.
Req 3: Update a person's record, displaying the actual data that can be changed. All data can be changed, except for code. The record is loaded after a search.
Req 4: Delete a person's record. The record is loaded after a search.
Req 5: Search a person's record, using one of the four criteria:

- Search by name.
- Search by surname.
- Search by full name. (name + surname).
- Search by code.

As the user inputs characters in the text field for the search criteria by name, surname and full name, a list should appear with a maximum of up to 100 people who match the characters entered so far, whom they can choose to edit, delete, or just see the record's data.

Engineering method:

PHASE 1: Problem identification

Symptoms and needs identification:

- The VIP Simulation team requires software that allows the efficient management of CRUD operations on a records' database.
- A software that uses CRUD operations efficiently does not exist.
- The software must guarantee certainty in its records and access to the database, correctly filtering the information as ordered by the user.
- The program must be efficient enough so that users can carry out their actions correctly, without any unintended consequence.

Problem definition

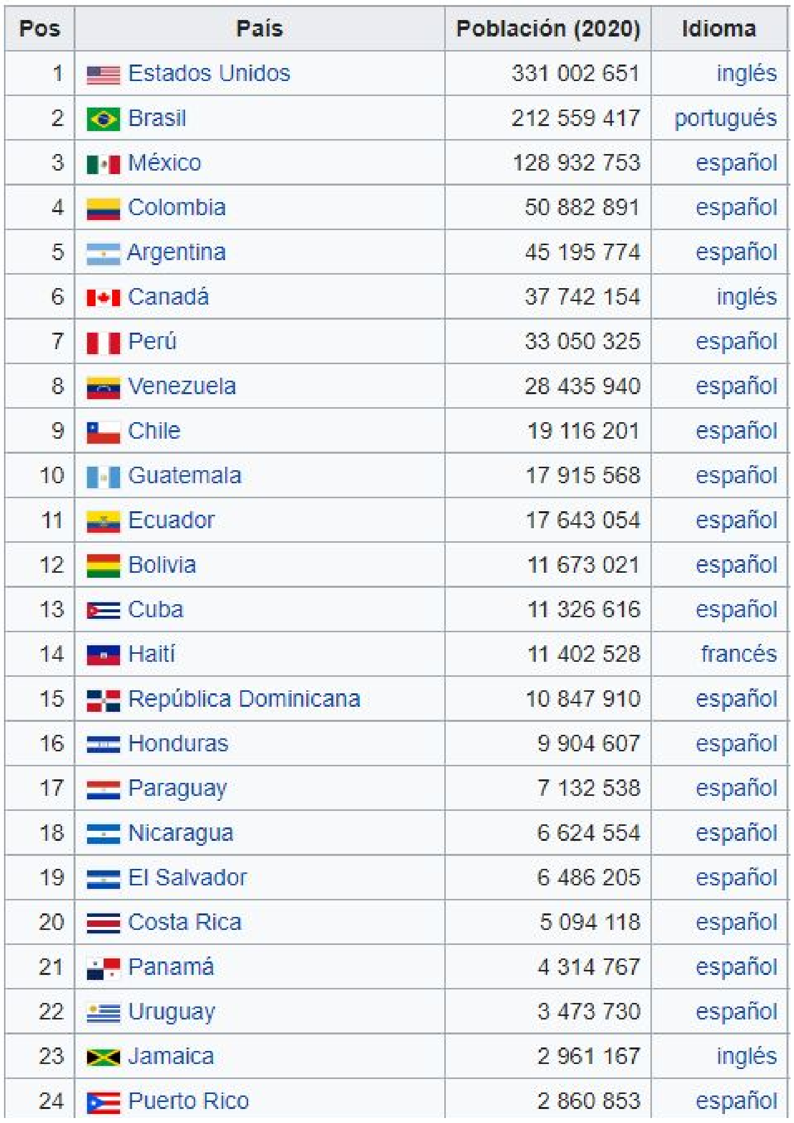
The VIP Simulation team requires a program that allows the efficient management of CRUD operations on a database of people records belonging to the American continent.

Alejandro Fonseca Forero
Diego Alejandro Garcia Cardona

PHASE 2: Necessary information compilation

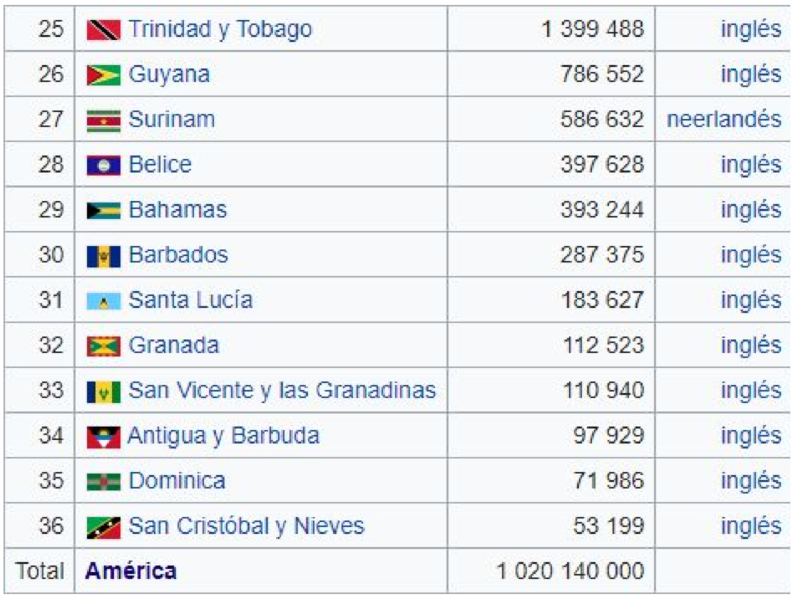
For a better understanding of the problem, the search of demographic information from the American continent is a must.

American continent:
The following table shows the American countries' name, their population and official language:



Pos	País	Población (2020)	Idioma
1	Estados Unidos	331 002 651	inglés
2	Brasil	212 559 417	portugués
3	México	128 932 753	español
4	Colombia	50 882 891	español
5	Argentina	45 195 774	español
6	Canadá	37 742 154	inglés
7	Perú	33 050 325	español
8	Venezuela	28 435 940	español
9	Chile	19 116 201	español
10	Guatemala	17 915 568	español
11	Ecuador	17 643 054	español
12	Bolivia	11 673 021	español
13	Cuba	11 326 616	español
14	Haití	11 402 528	francés
15	República Dominicana	10 847 910	español
16	Honduras	9 904 607	español
17	Paraguay	7 132 538	español
18	Nicaragua	6 624 554	español
19	El Salvador	6 486 205	español
20	Costa Rica	5 094 118	español
21	Panamá	4 314 767	español
22	Uruguay	3 473 730	español
23	Jamaica	2 961 167	inglés
24	Puerto Rico	2 860 853	español

Alejandro Fonseca Forero
Diego Alejandro Garcia Cardona



25	Trinidad y Tobago	1 399 488	inglés
26	Guyana	786 552	inglés
27	Surinam	586 632	neerlandés
28	Belice	397 628	inglés
29	Bahamas	393 244	inglés
30	Barbados	287 375	inglés
31	Santa Lucía	183 627	inglés
32	Granada	112 523	inglés
33	San Vicente y las Granadinas	110 940	inglés
34	Antigua y Barbuda	97 929	inglés
35	Dominica	71 986	inglés
36	San Cristóbal y Nieves	53 199	inglés
Total	**América**	1 020 140 000	

American age distribution, based on EE.UU data:

0-14 years: 18,62% (males 31.329.121 /females 29.984.705)
15-24 years: 13,12% (males 22.119.340 /females 21.082.599)
25-54 years: 39,29% (males 64.858.646 /females 64.496.889)
55-64 years: 12,94% (males 20.578.432 /females 22.040.267)
65 years or more: 16,03% (males 23.489.515 /females 29.276.951) (2018 est.)

First name & last name distribution:
Full names(first name & last name) were based on two data sets: One with the most common first names in english-speaking countries, and the other with the most common last names in english-speaking countries

CRUD Operations:

The **CRUD** concept is closely linked to digital data management. **CRUD** refers to an acronym in which the first letters of the four fundamental operations of persistent applications in database systems meet:

-Create (Create records)
-Read bzw. Retrieve (Read Records)
-Update (Update records)
-Delete bzw. Destroy (Delete Records)

Simply put, **CRUD** summarizes the functions required by a user to create and manage data. Several data management processes are based on **CRUD**, in which these operations are specifically adapted to the requirements of the system and the user, either for database management or for the use of applications. For experts, operations are the typical and indispensable access tools to check, for example, database problems, while for users, CRUD means creating an account (create yourself) and use it (read), update it(update) or delete (delete) at any time.

Trie:

Introduced in 1959 independently by Rene de la Briandais and Edward Fredkin, a trie is a tree-like data structure that allows information retrieval (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. Therefore, searching in a trie is done in a similar way to searching a dictionary.

Source:

-https://www.indexmundi.com/es/estados_unidos/distribucion_por_edad.html
-https://es.wikipedia.org/wiki/Anexo:Pa%C3%ADses_de_Am%C3%A9rica_por_poblaci%C3%B3n
-https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/crud-las-principales-operaciones-de-bases-de-datos/
-https://data.world/alexandra/baby-names
-https://data.world/uscensusbureau/frequently-occurring-surnames-from-the-census-2010
https://es.wikipedia.org/wiki/Trie

PHASE 3: SEARCH FOR CREATIVE SOLUTIONS

For the proposal of ideas, certain factors were defined, which could be replaced to generate new ideas. These are:

- Data structure
- Graphical user interface
- Search recommendation engine and progress bar.

Alternative 1:

For the storage of people, an array will be implemented, on which the **CRUD** operations will be executed. JavaFX will be used for interface development. Threads will be implemented for the search recommendation engine and the log generation progress bar.

Alternative 2:

For the storage of people, an arrayList will be implemented, on which the **CRUD** operations will be executed. JavaFX will be used for interface development. Threads will be implemented for the search recommendation engine and the log generation progress bar.

Alternative 3:
For the storage of people, a Stack will be implemented, on which the **CRUD** operations will be executed. JavaFX will be used for interface development. Threads will be implemented for the search recommendation engine and the log generation progress bar.

Alternative 4:

For the storage of people, a LinkedList will be implemented, on which the **CRUD** operations will be executed. JavaFX will be used for interface development. Threads will be implemented for the search recommendation engine and the log generation progress bar.

Alternative 5:

For the storage of people, a Hash Table will be implemented, on which the **CRUD** operations will be executed. JavaFX will be used for interface development. Threads will be implemented for the search recommendation engine and the log generation progress bar.

Alternative 6:

For the storage of people, an AVL Tree (**TRIE**) will be implemented, on which the **CRUD** operations will be executed. JavaFX will be used for interface development. Threads will be implemented for the search recommendation engine and the log generation progress bar.

PHASE 4: TRANSITION FROM FORMULATION OF IDEAS TO PRELIMINARY DESIGNS

After deliberation, we decided to discard the following ideas:

Alternative 1:
- This alternative is not viable because an array is a fixed-size data structure, therefore, it can limit us when registering people in our database.

Alternativa 2:

- This alternative is not viable because the arrayList as a data structure is inefficient for the amount of data we intend to handle, greatly increasing the times of operations.

Alternative 3:

- This alternative is not viable due to the nature of the Stack (FIFO) which makes an efficient search and elimination implementation impossible.

PHASE 5: EVALUATION AND SELECTION OF THE BEST SOLUTION

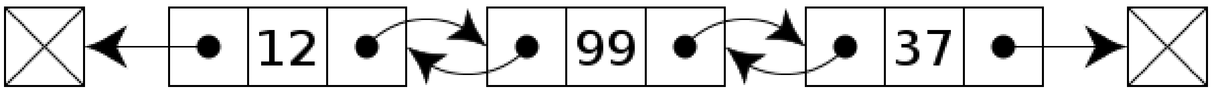
Before we can choose the appropriate alternative, we first analyze various information about each of the resulting alternatives:

Alternative 4:

A **linked list** is a linear collection of data elements whose order is not given by their physical placement in memory. Instead, each element points to the next. It is a data structure consisting of a collection of nodes which together represent a sequence.

The principal benefit of a linked list over a conventional array is that the list elements can be easily inserted or removed without reallocation or reorganization of the entire structure because the data items need not be stored contiguously in memory or on disk, while restructuring an array at run-time is a much more expensive operation.

It can be single, double and circular, providing a variety of implementations for different problems.



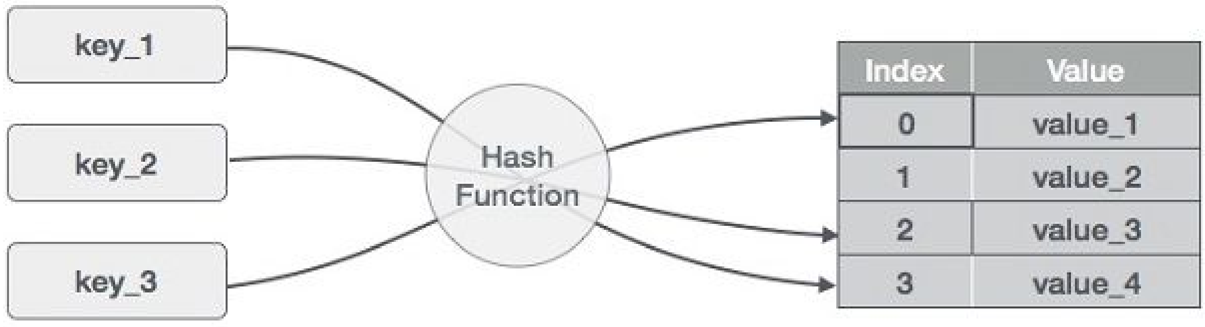
Double linked list

Alternative 5:
Hash Table is a data structure which stores data in an associative manner. In a hash table, data is stored in an array format, where each data value has its own unique index value. Access of data becomes very fast if we know the index of the desired data.

Thus, it becomes a data structure in which insertion and search operations are very fast irrespective of the size of the data. Hash Table uses an array as a storage medium and uses a hash technique to generate an index where an element is to be inserted or is to be located from.

Alejandro Fonseca Forero
Diego Alejandro Garcia Cardona

Hashing
Hashing is a technique to convert a range of key values into a range of indexes of an array. We're going to use a modulo operator to get a range of key values. Consider an example of a hash table of size 20, and the following items are to be stored. Items are in the (key,value) format.



Alternative 6:

AVL Tree.
A Binary Search Tree (BST) is a tree in which all the nodes follow the below-mentioned properties −

The value of the key of the left sub-tree is less than the value of its parent (root) node's key.

The value of the key of the right sub-tree is greater than or equal to the value of its parent (root) node's key.

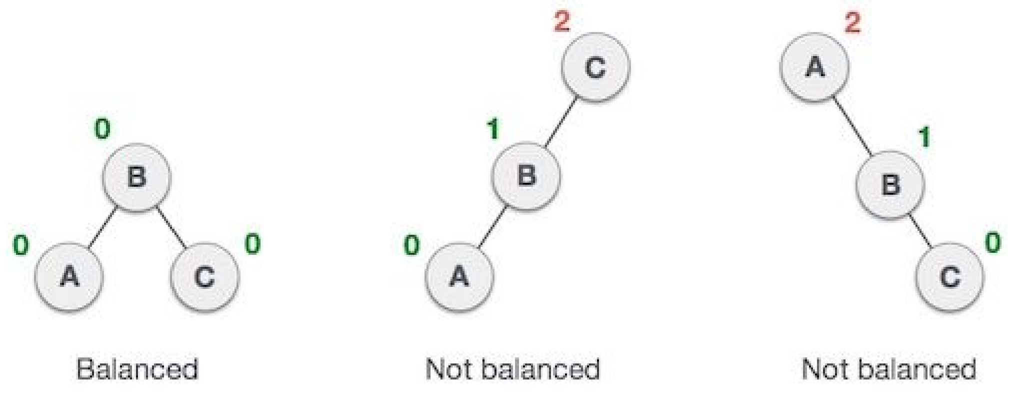
Thus, BST divides all its sub-trees into two segments; the left sub-tree and the right sub-tree and can be defined as:

left_subtree (keys) < node (key) ≤ right_subtree (keys)

Named after their inventor Adelson, Velski & Landis, AVL trees are height balancing binary search tree. AVL tree checks the height of the left and the right sub-trees and assures that the difference is not more than 1. This difference is called the Balance Factor.

Here we see that the first tree is balanced and the next two trees are not balanced

Alejandro Fonseca Forero
Diego Alejandro Garcia Cardona



Bibliography:

- https://en.wikipedia.org/wiki/Linked_list#Singly_linked_list
- https://www.tutorialspoint.com/data_structures_algorithms/hash_data_structure.htm
- https://www.tutorialspoint.com/data_structures_algorithms/avl_tree_algorithm.htm

To choose the appropriate alternative for the development of our solution, we define a rubric that allows us to rate the following aspects from 1 to 5

- Temporal complexity in insertion and search methods.
- Structure behavior when handling large databases.
- Flexibility and comfort in its implementation.

Alejandro Fonseca Forero
Diego Alejandro Garcia Cardona

Rúbrica:

Alternative	**Time complexity in insertion and search**	**Large capacity handling**	**Ease of implementation**	**Total score**
Alternativa 4(Linked List)	O(n) 2	Too slow 1	Easy 5	8
Alternativa 5(Hash Table)	O(1) average O(n) worst case 4	Fast (Direct access). 4	Colisions. 2	10
Alternativa 6(AVL Tree)	O(log n) 5	Fast(Balanced data, keeping easy handling) 5	Inherited class of ABB. 5	15

The winning alternative has been 6, which implements an AVL tree. This is the alternative adopted for the development of our solution.