Marco Fidel Vasquez
David Steven Montoya - A00362450
Camilo Escobar Arteaga - A00358632

REPORT OF ENGINEERING METHOD

PROBLEM IDENTIFICATION

Problem Necessities

- Generate registries about the inhabitants with the following data: An auto generated code, name, last name, gender, birthdate, height(it should be generated randomly within an interval which must be reasonable), nationality and a picture.
- The software should take names from this dataset for the registry. It should also take information from this dataset to generate the respective last name.
- The age distribution must be randomly generated based on the United States official distribution.
- The nationality must be generated in a way that keeps the percentages taken from this dataset.
- The pictures should be taken from https://thispersondoesnotexist.com/ and assigned to each client. Gender correspondence can be ignored.
- All the generated registries can be saved in some place where they can be accessed on further use.
- The database should be able to do CRUD operations (Create, Read, Update, Delete).

Definition of the problem

- The VIP Simulation team requires a software capable of efficiently processing CRUD (Create, Read, Update & Delete) operations in a database consisting of inhabitants of the American Continent. They require the use of the following datasets to generate the inhabitants, first we must use this dataset to generate the names and this other dataset for the last names. Additionally, it should use this United States age distribution to randomly generate a birthdate within those parameters, and this data of population by country to spread out the population consistently.

Functional Requirements:

FR1: Generate an instance of a person with the following attributes: a randomly generated code, a name, last name, gender, birthdate, height, nationality and a picture and save it to the database.

FR2: Use the various datasets to generate a random name, last name, gender distribution, age distribution (birthdate), height distribution, nationality and picture.

FR3: Have a text field where the user can input the amount of instances to be created, by default this text field should be set to the maximum.

FR4: Show a progress bar if the previous process takes more than 1 second to be completed and show the amount of time taken to register all of the given instances.

FR5: Search the database with the following criteria: Name, last name, full name, code. While the user is inputting the data, if it uses any of the first 3 criteria, the program should find the first 100 (parameterizable) closests names in the database. While the search is being done, to the right of the text field where the user does the search the number of close coincidences should be shown, also, when there are 20(parameterizable) or less coincidences, each name in the list should have a button to its right that loads the data to the interface, should it be clicked.

FR6: Update any camp except code from an element.

FR7: Delete an element from the database.

Non Functional Requirements:

NFR1: The structures to save the users into different ordering criteria will be an AVL Binary search tree.

NFR2: The database will be saved using Serializable.

Compilation of information

Definitions

Dataset: A collection of separate sets of information that is treated as a single unit by a computer.

Database: An organized collection of structured information, or data, typically stored electronically in a computer system.[1]

CRUD operations: CRUD refers to create, read, update and delete operations, which are defined down here.[2]

> CREATE procedures: Performs the INSERT statement to create a new record.
>
> READ procedures: Reads the table records based on the primary key noted within the input parameter.
>
> UPDATE procedures: Executes an UPDATE statement on the table based on the specified primary key for a record within the WHERE clause of the statement.

[1] https://www.oracle.com/database/what-is-database.html

[2] https://stackify.com/what-are-crud-operations/

DELETE procedures: Deletes a specified row in the WHERE clause.

TRIE: A trie is a data structure in the tree type that allows information recovery (from english reTRIEval).

POSSIBLE SOLUTIONS

Possible solutions to show results to the user while they digit the information:

Solution 1: Create a thread which will constantly search for people every second while the user digits the information.

Solution 2: Create a thread which will search for people as the user digits or deletes characters.

Solution 3: Create a method that uses javaFx textField's onKeyPressed that calls the search method every time the user presses a key.

Solution 4: Create a method that uses javaFx textField's onKeyTyped that calls the search method every time the user types a key.

Solution 5: Create a method that uses javaFx textField's onKeyReleased that calls the search method every time the user releases a key.

(autocompleteTextfield and contextMenu) (preguntarle a yusunguaira cual es mejor)(camilo le preguntó)

TRANSFORMING IDEA FORMULATION TO PRELIMINARY DESIGNS

Description solution 1: Create a thread which will constantly search for people every second while the user digits the information. It searches in the tree every second without regard to whether more characters are being written or not.

Description solution 2: Create a thread which will search for people as the user digits or deletes characters. It searches in the tree

Description solution 3: Create a method that uses JavaFx's textField's onKeyPressed, making it call the search method every time the user types a letter or a number.

Description solution 4: Create a method that uses JavaFx's textField's onKeyTyped, making it call the search method every time the user types a letter or a number.

Description solution 5: Create a method that uses JavaFx's textField's onKeyTyped, making it call the search method every time the user types a letter or a number and lets go of it.

EVALUATION AND SELECTION OF THE BEST SOLUTION

PREPARATION OF REPORTS AND SPECIFICATIONS

DESIGN IMPLEMENTATION