Tarea Integradora III

ICESI ROUTES

Daniela Olarte Borja - A00368359

Juan Jacobo Garcia Aristizabal - A00368502

Santiago Arévalo Valencia - A00368494

Alexander Sánchez Sánchez- A00368238

Programa de Ingeniería de Sistemas

Facultad de Ingeniería

Periodo II de 2021

Universidad Icesi

Cali, Colombia

INDEX

STATEMENT

Ser “el nuevo” en un lugar es algo que todas las personas enfrentan al entrar a alguna comunidad, grupo o trabajo. Muchas de estas veces las personas tienden a sentir miedo, frustración o incluso angustia al no saber a qué entorno cada quien se enfrentará. Entrar a la universidad es uno de los casos, pues nadie sabe a quién conocerá, qué ambiente verá y cómo se adaptará a este nuevo lugar.

Juan es un joven de 17 años que recién acaba de iniciar su carrera en la Universidad ICESI. Lamentablemente, por situaciones de covid-19 no pudo asistir a Matileo (actividad universitaria que tiene como objetivo integrar a los estudiantes, hacer que estos recuerden conceptos académicos básicos y puedan conocer un poco del campus), por lo que, en su primer día de clases llegó 20 minutos tarde a su única clase del día y el profesor lo marcó como ‘Ausente’.

El joven que estudia Ingeniería de Sistemas, se sintió frustrado al llegar tarde a su primera clase y decidió contactarse con su directora de programa para que lo ayudara a encontrar una solución. Después de una larga charla, la directora identificó que el problema era que Juan no conocía bien la universidad y por eso había llegado tarde a su clase, por lo que le pidió a un estudiante de octavo semestre que le mostrara la universidad.

Al siguiente día Juan tuvo el mismo problema de llegada tarde, lo que hacía que tuviera ahora 2 faltas en su clase de Introducción a las TIC. Este acudió esta vez donde su directora de carrera y pudieron identificar que el problema no era que Juan no conocía la universidad sino que no sabía tomar la ruta más eficiente para así llegar a tiempo a su clase.

Su directora de carrera le pidió a 3 estudiantes de Ingeniería de Sistemas y 1 de Ingeniería Telemática que hicieran un programa para ayudar a los nuevos estudiantes. Por lo que les pidió que el programa tuviera un mapa donde los estudiantes pudieran seleccionar su ubicación y el lugar a donde quisieran llegar, mostrando así una lista ordenada de opciones de rutas. De ese modo, la mejor ruta sería la primera de la lista, es decir, la ruta con menor tiempo, haciendo que la lista se vea de forma ascendente en cuanto al tiempo. Además, la directora quiere que la aplicación muestre qué tan lejos (en metros) está el estudiante de su destino y un tiempo estimado (en segundos) de cuánto se tardaría un estudiante de un lugar a otro caminando a ritmo constante.

FUNCTIONAL REQUIREMENTS

R1. Select the start location.
R2. Select the final location.
R3. Show all routes to destination.
R4. Show sorted routes, placing the lowest distance on top of the list.
R5. Show distance from the start location to the final location of the best route in meters.
R5. Shows estimated time from the start location to the final location of the best route, assuming a constant pace in seconds.

ENGINEERING METHOD

1.Identification of the problem:

ICESI routes objective is to help new students when they first get to know their university and the routes to take in order to get to their desired place in the shortest time. In this program, the student should select the place where they are currently located and also select the place where they want to head. In a matter of seconds the program should provide the student with all the possible routes to get to their destination. The route list is sorted, making the first option of the list the best route to take. The list also shows how far the student's destination is from where they are and an estimation of the time that it should take to get there.

2.The collection of the necessary information:

Graph: A graph is a non-empty set of objects called vertices (or nodes) and a selection of pairs of vertices, called edges (edge in English) that can be oriented or not. Typically, a graph is represented through a series of points (the vertices) connected by lines (the edges). The Formally, a graph is defined as {\ displaystyle G = (V, A)} {\ displaystyle G = (V, A)}, where V is a set whose elements are the vertices of the graph and A is a set whose elements are the edges , which are pairs (ordered if the graph is directed) of elements in V.

Source: http://www.unipamplona.edu.co/unipamplona/portalIG/home_23/recursos/general/11072012/grafo3.pdf

Adjacency List: In this data structure the idea is to associate to each vertex i of the graph a list containing all those vertices j that are adjacent to it. In this way, it will only reserve memory for arcs adjacent to i and not for all possible arcs that could have i as origin. The graph, therefore, is represented by means of a vector of n components (if | V | = n) where each component is going to be an adjacency list corresponding to each of the vertices of the graph. Each element of the list consists of a field indicating the adjacent vertex.

Source: https://es.slideshare.net/FrankDoria/lista-de-adyacencia

Adjacency Matrix: It is a Boolean matrix that represents the connections between pairs of vertices. The matrix of adjacency of a graph is symmetric. If a vertex is isolated then the corresponding row(column) is composed only of zeros. If the graph is simple then the adjacency matrix contains only zeros and ones (binary matrix) and the diagonal is made up of only zeros.

Source: https://www.utm.mx/~mgarcia/ED4(Grafos).pdf

Depth-first Traversal (DFS): Is a technique that is used to traverse a tree or a graph. DFS technique starts with a root node and then traverses the adjacent nodes of the root node by going deeper into the graph. In the DFS technique, the nodes are traversed depth-wise until there are no more children to explore. Once we reach the leaf node (no more child nodes), the DFS backtracks and starts with other nodes and carries out traversal in a similar manner. DFS technique uses a stack data structure to store the nodes that are being traversed.

Breadth-first Traversal (BFS): Is a technique that uses a queue to store the nodes of the graph. As against the DFS technique, in BFS we traverse the graph breadth-wise. This means we traverse the graph level wise. When we explore all the vertices or nodes at one level we proceed to the next level.

Source: https://www.softwaretestinghelp.com/java-graph-tutorial/

Dijkstra's algorithm: Also called minimum paths algorithm, it is an algorithm for determining the shortest path given a vertex origin to the rest of vertices in a graph with weights on each edge. Its name refers to Edsger Dijkstra, who first described it in 1959. Some characteristics of the algorithm it is:
• A greedy algorithm.
• Work in stages, and take the best solution at each stage without considering future consequences.
• The optimum found in one stage can be modified later if a better solution emerges.

Source: https://www.ecured.cu/Algoritmo_de_Dijkstra

3.Search of creative solutions:

In order to come up with creative solutions to the problem at hand, we decided to brainstorm, with each team member contributing one or two quick solutions that, at the same time, met the requirements established in the project.

Alternatives:

1. The first idea about the implementation of the graph structure in a local problem was to implement a map of the ICESI University, in which, specifically the first semester students or those who are just getting to know the university campus, could have a guide through this application. This would represent the main buildings, auditoriums and campus spaces, also showing the paths that connect them.

2. Then we came up with an idea that we considered more practical for the user, in which he/she could interact with the application and choose his origin and destination. In this solution, the student must select the place where he is currently located and also select the place where he wants to go. In a matter of seconds the program should provide the student

with all possible routes to reach his destination. The list of routes is sorted, making the first option on the list the best route to take. The list also shows how far away the learner's destination is and an estimate of how long it should take to get there.

3. Another proposed solution was to implement a program that, depending on the student's location, would show the nearest places or spaces to which he/she could go with their respective distance and time. The student could choose from among the options of these places to advance in the route.

4. Another idea that arose was to implement a guide for a first semester student or one who was just getting to know the university campus, which shows him the classrooms where he has his classes for each of the subjects, along with the distance and the time it takes him to get to each of these.

5. Another proposed solution was to implement a program that would ask the student how much time they have to arrive at their destination. According to it, the program would show a sorted list placing the best route to the desired destination.

6. Our final idea was to implement a program that would show the user's nearest destination and have the user rate the program's accuracy so that other students could discuss with each other which is the best route to take.

4.Moving from ideas to preliminary designs:

Criterias:

- Solution precision; The solution is viable and meets the effective development of all the requirements of the problem.

1. Imprecise
2. Almost Accurate
3. Accurate

- Effectiveness: Regardless of the complexity, the solution complies with a perfect effectiveness lacking any logical error

1. No Effective
2. Effective Medium
3. Effective

- Usability: The solution can be used because it meets all the proposed objectives

1. Not usable at all
2. Complex
3. Usable

- Accessible: The solution presents easy access to the data generated

1. Not accessible
2. Moderately accessible
3. Accessible

	Solution Precision	Effectiveness	Usability	Accessible
Alternative 1	2	3	2	2
Alternative 2	3	3	3	3
Alternative 3	2	1	1	3

Alternative 1: 9
Alternative 2: 12
Alternative 3: 7

According to the previous evaluation, **Alternative 2** should be selected, since it obtained the highest score according to defined criteria.

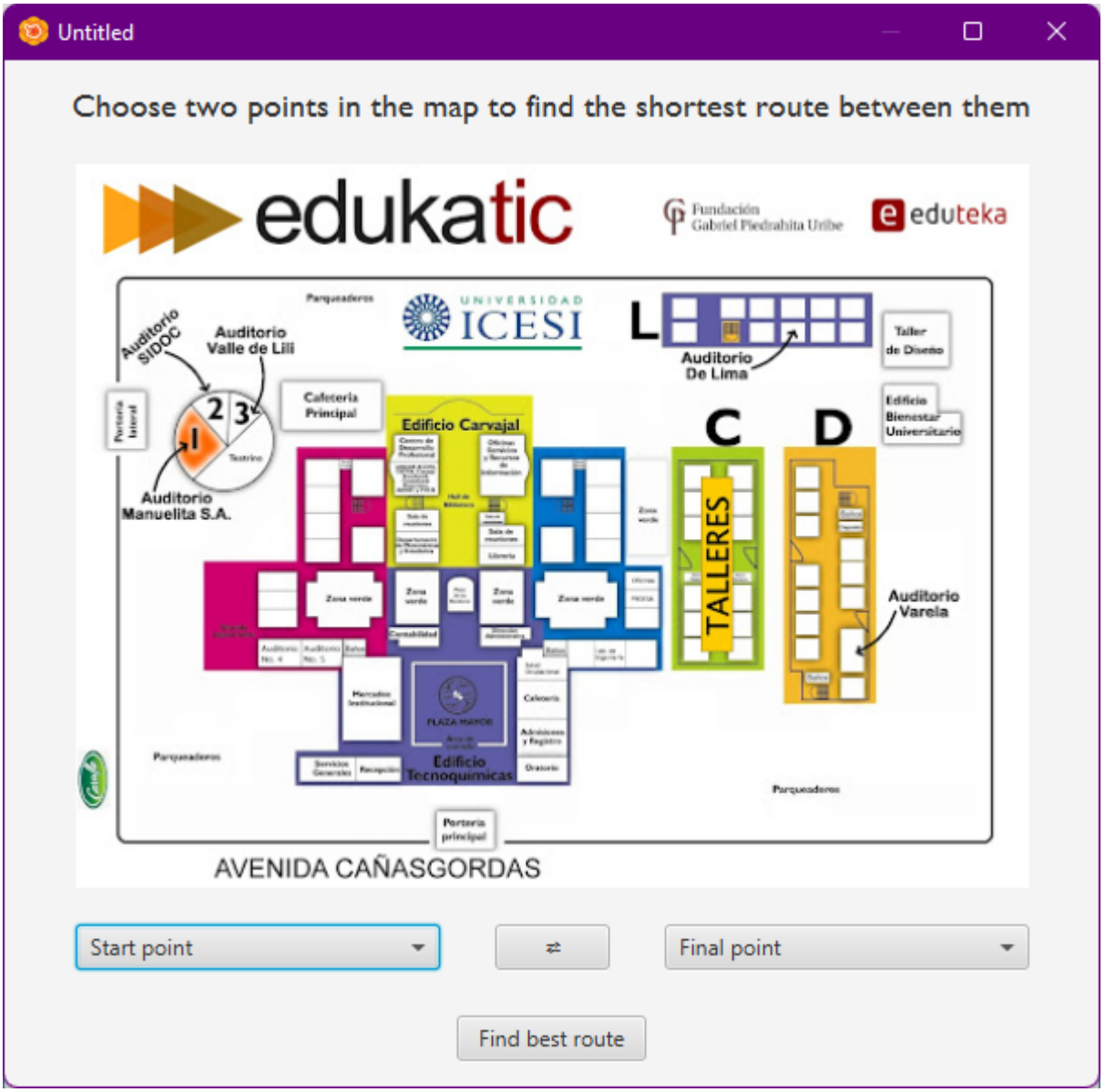
5.The evaluation and selection of the preferred solution:

We chose alternative 2, as it presents the easiest way to collect information by having only the main buildings, auditoriums and spaces. In addition, it is a more interactive idea with the user by allowing him to choose from where he is and where he wants to go. Additionally, this solution employs algorithms for the search of paths with minimum length, which, in the context of the student with the need to move within the campus would mean less travel time, which sometimes could be very useful.

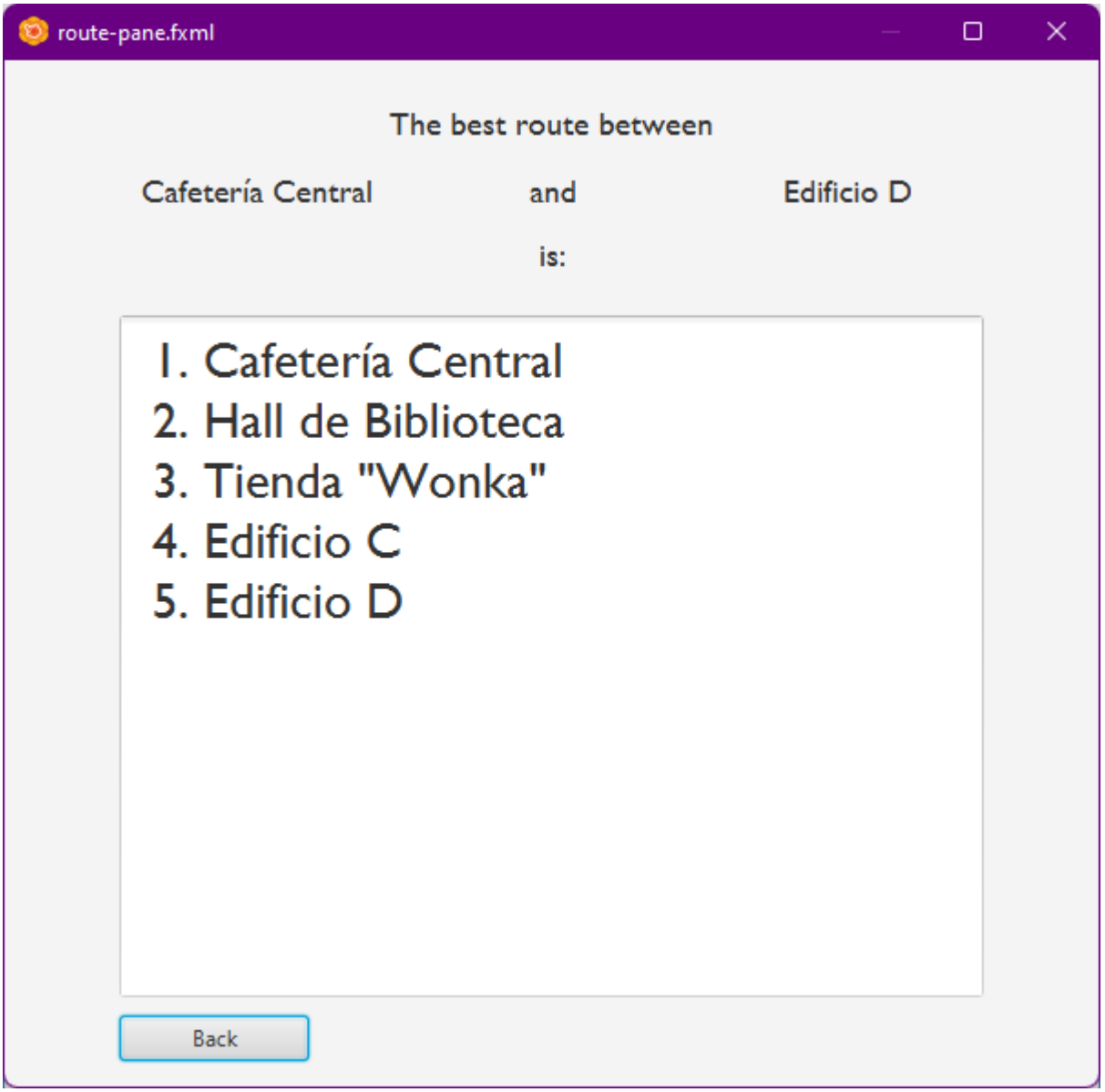
We discarded alternative 1 and 3 since they do not contain a graph solution. They do provide a good solution for the nearest route to take even if it is not a sorted list, however, it wouldn't be as efficient as having a graph which contains an adjacency list.

6.Design implementation:

- **First Screen:** where the user can choose two points, according to the shown map, which will be found the best route between them. Additionally, these two points can be exchangeable to improve the user experience.



- **Second Screen:** where the user can visualize the best route calculated by the program, quickly. Plus, from this screen, the user will be able to go to the previous screen and search another route.


route-pane.fxml
The best route between
Cafetería Central
and
Edificio D
is:
1. Cafetería Central
2. Hall de Biblioteca
3. Tienda "Wonka"
4. Edificio C
5. Edificio D
Back