**Método de la ingeniería- Proyecto Final**

1. **Identificación del Problema**
2. La empresa requiere conocer la ruta más económica para la entrega de sus pedidos.
3. El programa debe permitir agregar o eliminar ubicaciones y calcular la ruta más corta y económica para realizar las entregas.
4. El programa debe mostrar cuales lugares ya fueron visitados (o realizado la entrega) para no volver a visitarlo o realizar la entrega.
5. El programa debe informar sobre el consumo de gasolina realizado en la ruta de entrega.
6. El programa debe tener información sobre la gasolina para poder realizar cálculos sobre los gastos de las entregas.

**Definición del problema**

Una empresa de entregas anónima de la ciudad de Cali requiere un programa economizar el consumo de combustible de sus camiones, mediante una funcionalidad que permita hallar la ruta rápida y económica entregar los pedidos. Además de eso, se requiere también que se pueda revisar desde la oficina central que pedidos se han entregado viendo en que parte de la ruta va el camión. Lo anterior con el fin de tomar datos para analizar la eficiencia de los conductores y sus ayudantes, y de esa manera tomar decisiones futuras sobre la cantidad de entregas que se deberían realizar diariamente. La empresa ha sido muy clara y especifica; quiere que el programa sea modelado por uno o varios grafos, los que se necesiten. El programa debe permitir la visualización de los camiones y sus rutas, y por consiguiente el mapa de la ciudad de Cali. Adicionalmente, se debe tener información sobre la gasolina (precio y galones usados por ruta) para calcular cuánto dinero se está invirtiendo, aproximadamente, en combustible por ruta.

Requerimientos Funcionales:

Requerimientos No Funcionales:

1. Recopilación de la información
2. Búsqueda soluciones creativas
3. Transición de las ideas a los diseños preliminares.
4. Evaluación y selección de la mejor solución
5. Preparación de informes y especificaciones

1. Tad Grafo

|  |
| --- |
| TAD Graph |
| Representation: |
| Invariat:  No direccionado |
| Operations:  Grafo Graph  AddVert Vertex  AddArist Vertex, Vertex, Integer  getVert Vertex getVerts  DeleVert Vertex  DeleArist Arist  BFS Vertex  DFS Vertex |
|  |

|  |
| --- |
| AddVert(Vertex) |
| Adds a Vertex to the graph. Pre: void Post: V = V + 1 |

|  |
| --- |
| AddArit(Vertex,Vertex,Integer) |
| Adds a weighted edge from one vertex to another. Pre: void Post: Graph.weight = Graph.weight + e.weight |

|  |
| --- |
| getVert (Vertex) |
| Gives the information of a Vertex. Pre: Vertex  Graph ᴧ Vertex ≠ nil Post: Vertex |

|  |
| --- |
| getVerts () |
| Gives the list of the vertexes of the graph. Pre: Vertex(i)  Graph ᴧ Vertex(i) ≠ nil Post: Vertex {V1,V2...Vn} |

|  |
| --- |
| DeleVert(Vertex) |
| Deletes a Vertex of the graph. Pre:  Vertex  Graph ᴧ  Vertex ≠ nil Post: V  = V - 1 |

|  |
| --- |
| DeleArist (Edge) |
| Deletes an edge of the graph. Pre: Edge  Graph ᴧ Edge ≠ nil  Post: Graph.weight |

= Graph.weight - e.weight

|  |
| --- |
| BFS(Vertex) |
| Explores all the neighbor nodes of a vertex selected as root node. Pre: Vertex  Graph ᴧ Vertex ≠ nil Post:  void |

|  |
| --- |
| DFS(Vertex) |
| Explores the possible routes as far as possible of a vertex selected as root node. Pre: Vertex  Graph ᴧ Vertex ≠ nil Post:  void |

-Diseño de pruebas unitarias

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P.1 | Objetivo de la prueba= comprobar si se agrega un vértice | | | |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Grafh | AddVert(Vertex) | Escenario 1 |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P.2 | Objetivo de la prueba= comprobar si se agrega un arista | | |  |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Grafh | AddArit(Vertex,Vertex,Integer) | Escenario 1 |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P.3 | Objetivo de la prueba= comprobar si retorna el vertice | | |  |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Grafh | getVert (Vertex) | Escenario 1 |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P.4 | Objetivo de la prueba= comprobar si retorna los vertices | | | |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Grafh | getVerts () | Escenario 1 |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P.5 | Objetivo de la prueba= comprobar si elimina un vertice | | |  |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Grafh | DeleVert(Vertex) | Escenario 1 |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P.6 | Objetivo de la prueba= comprobar si elimina una arista | | |  |
| Clase | Método | Escenario | Valores de entrada | Resultado |
| Grafh | DeleArist (Edge) | Escenario 1 |  |  |

1. Implementación del diseño

Enlace al repositorio de GitHub: <https://github.com/Sergiiok/FinalProjectAED>