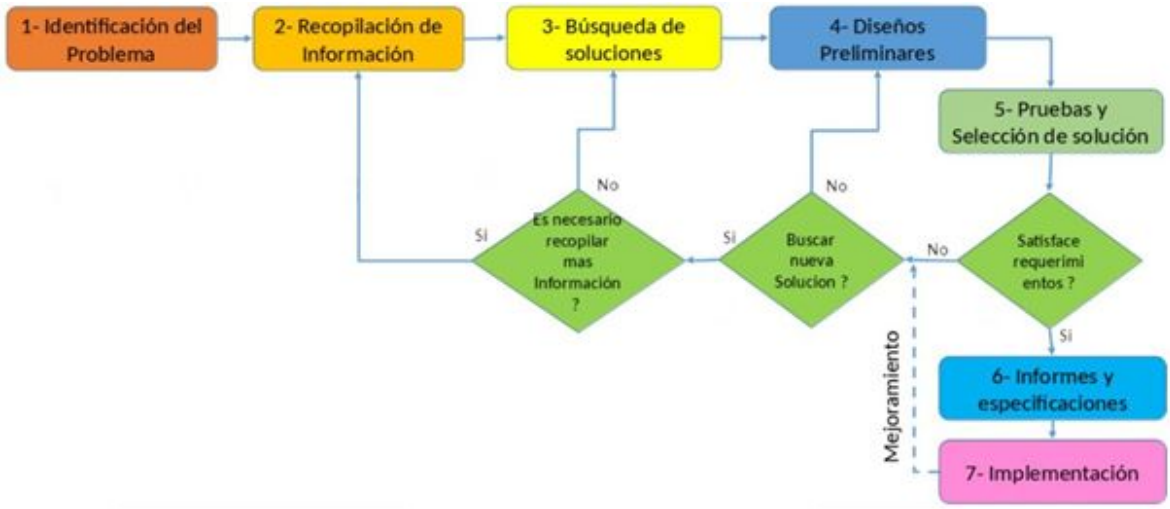
Desarrollo de la Solución

Para resolver la situación anterior se eligió el Método de la Ingeniería para desarrollar la solución siguiendo un enfoque sistemático y acorde con la situación problemática planteada.

Con base en la descripción del Método de la Ingeniería del libro “Introduction to Engineering” de Paul Wright, se definió el siguiente diagrama de flujo, cuyos pasos seguiremos en el desarrollo de la solución



Fase 1. Identificación del problema

Contexto problemático (identificando causas y síntomas):

Como sabemos el transporte en las ciudades principales de Colombia son un poco desastrosas, por eso se quiere hacer un programa que nos ayude a usar uno de los transportes más tecnológicos de nuestro país, el metro. Nuestra aplicación tendrá un mapa con las estaciones del Metro de Medellín y el usuario podrá escoger la estación de salida y la de llegada para luego saber la ruta más adecuada para llegar a su destino en el tiempo más corto.

Identificación y definición concreta y sin ambigüedad el problema: Definición del Problema

Se nos ha encargado crear una App que integre un sistema capaz de recomendar a los usuarios del Metro de Medellín la mejor y más eficiente ruta posible para que los usuarios lleguen en el menor tiempo posible a sus destinos.

Identificación de necesidades y síntomas

Consideraciones:

1. El problema solo se enfocará en encontrar las mejores rutas para el Usuario y dar con ello estimaciones de la duración del viaje, los tiempos de espera para subir al Metro no son tenidos en cuenta.

2. Esta App es una aproximado de las rutas del Metro, mas no es la versión final del mismo por lo que las rutas no tendrá las mismas proporciones de distancias, se tomarán otros datos para la aproximación y análisis de la misma.

Fase 2. Recopilación de la información necesaria

Como para este problema se ven involucrados algunos conceptos que no todos tienen conocimiento, entonces se hizo una búsqueda de los conceptos más importante de manera superficial para poder entender la solución que se va a llevar a cabo.

Grafo:

En matemáticas y ciencias de la computación, un grafo (del griego grafos: dibujo, imagen) es un conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto.1 Son objeto de estudio de la teoría de grafos.

Grafo Ponderado:

Un grafo ponderado, pesado o con costos es un grafo donde cada arista tiene asociado un valor o etiqueta, para representar el costo, peso, longitud, etc.

Representaciones de Grafos:

- **Lista de Adyacencia:**

  El grafo está representado por un arreglo de listas de adyacencia. Para un vértice i, la lista de adyacencia está formada por todos los vértices adyacentes a i. Puede construirse en tiempo lineal, y las inserciones pueden hacerse al principio de cada lista, con lo que se asegura tiempo constante

- **Matriz de Adyacencia:**

  Una matriz de adyacencia es una matriz M de dimensión n*n, en donde n es el número de vértices que almacena valores booleanos, donde M[i,j] es verdadero (o contiene un peso) si y sólo si existe un arco que vaya del vértice i al vértice j. La inicialización llevaría un tiempo del O ((V^2)).

- **Matriz de Incidencia:**

  El grafo está representado por una matriz de A (aristas) por V (vértices), donde [arista, vértice] contiene la información de la arista (conectado o no conectado).

- **Búsqueda en anchura (BFS)**

La búsqueda en anchura es otro procedimiento para visitar sistemáticamente todos los vértices de un grafo. Es adecuado especialmente para resolver problemas de optimización, en los que se deba elegir la mejor solución entre varias posibles.

- **Búsqueda en profundidad (DFS)**

  Una Búsqueda en profundidad es un algoritmo que permite recorrer todos los nodos de un grafo o árbol (teoría de grafos) de manera ordenada, pero no uniforme. Su funcionamiento consiste en ir expandiendo todos y cada uno de los nodos que va localizando, de forma recurrente, en un camino concreto. Cuando ya no quedan más nodos que visitar en dicho camino, regresa (Backtracking), de modo que repite el mismo proceso con cada uno de los hermanos del nodo ya procesado.

- **Algoritmo de Dijkstra:**

  El algoritmo de Dijkstra, también llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto, dado un vértice origen, hacia el resto de los vértices en un grafo que tiene pesos en cada arista.

- **Algoritmo de Floyd-Warshall:**

  El problema que intenta resolver este algoritmo es el de encontrar el camino más corto entre todos los pares de nodos o vértices de un grafo. Esto es semejante a construir una tabla con todas las distancias mínimas entre pares de ciudades de un mapa, indicando además la ruta a seguir para ir de la primera ciudad a la segunda.

- **Algoritmo de Kruskal:**

  El algoritmo de Kruskal es un algoritmo de la teoría de grafos para encontrar un árbol recubridor mínimo en un grafo conexo y ponderado. Es decir, busca un subconjunto de aristas que, formando un árbol, incluyen todos los vértices y donde el valor de la suma de todas las aristas del árbol es el mínimo.

- **Algoritmo de Prim:**

  El algoritmo de Prim es un algoritmo perteneciente a la teoría de los grafos para encontrar un árbol recubridor mínimo en un grafo conexo, no dirigido y cuyas aristas están etiquetadas.

- **Árbol Mínimo de Expansión:**

  Dado un grafo conexo y no dirigido, un árbol recubridor mínimo de ese grafo es un subgrafo que tiene que ser un árbol y contener todos los vértices del grafo inicial.

<u>Fase 3. Búsqueda de soluciones creativas</u>

Para dar solución al requerimiento “Generar la ruta más corta”, se llega a que los algoritmos más eficientes y útiles para cumplir ésta funcionalidad son BFS, DFS, Dijkstra y Floyd-Warshall. Estos nos ayudarán a obtener de un grafo la mejor representación para visualizar el mejor camino entre un punto cualquiera A y un punto cualquiera B.

Además, con lo anterior se puede dar un tiempo aproximado de cuánto tiempo tarda la ruta que tomará el usuario. También se mostrará la mejor ruta al usuario y le da una idea de cuánto tiempo le tomara esa ruta.

Fase 4. Transición de la formulación de ideas a los diseños preliminares

Las estructuras que mejor puede representar la solución de nuestro problema son los grafos, ya que los vértices serían las estaciones y las aristas son las conexiones entre ellas. Se hará uso de lo algoritmos de Dijkstra y Floyd-Warshall, ya que dan la ruta más corta de uno a más vértices y la ruta más corta entre cada par de vértices, respectivamente.

Adicionalmente, se implementará la interfaz con JavaFX para ofrecerle al usuario mayor satisfacción a la vista y mejor comodidad, ya que este UI tiene la especialidad de tener herramientas para facilitar el uso de la aplicación.

Fase 5. Evaluación y selección de la mejor solución

- Criterios:
- **A: Estabilidad**
    - **Estable: 2**
    - **Inestable: 1**
- **B: Búsqueda del camino más corto en un grafo**
    - **Si: 2**
    - **No: 1**
- **C: Implementación**
    - **Fácil: 2**
    - **Difícil: 1**
- **D: Complejidad**
    - **Eficiente: 2**
    - **Ineficiente: 1**

Algoritmos\Criterios	**A**	**B**	**C**	**D**	**Total**
Floyd-Warshall	**2**	**2**	**2**	**2**	**8**
Dijkstra	**2**	**2**	**2**	**2**	**8**
BFS	**2**	**1**	**2**	**2**	**7**
DFS	**2**	**1**	**2**	**2**	**7**
Prim	**2**	**1**	**2**	**2**	**7**
Kruskal	**2**	**1**	**2**	**2**	**7**

Fase 6. Preparación de informes y especificaciones

Con lo que podemos concluimos con este respectivo a este estudio y su respectivo análisis, buscando los mejores algoritmos y estructuras para ser implementados en el programa especificado y así satisface los requerimientos antes planificados, se han seleccionado el algoritmo Dijkstra y Floyd-Warshall tanto para saber la menor distancia de estación a estación como saber que ruta debo tomar para llegar de uno a otra extrema de la forma más eficiente.