# Integrantes

|  |  |  |
| --- | --- | --- |
| Integrante 1 | Juan Sebastián Villamil | 201715419 |
| Integrante 2 | Daniel Bernal | 201519654 |

# Documentación de los requerimientos

## Requerimiento 1A

|  |  |
| --- | --- |
| Atributo | Descripción |
| Datos de entrada | 1. *P1: Latitud-Longitud del punto 1* 2. *P2: Latitud-Longitud del punto 2* |
| Datos de salida | *La impresión se realiza por consola* |
| Estimación de complejidad | *Se es requerido un mínimum-path entre dos nodos del grafo, por lo que para su búsqueda óptima se usará Dijkstra, dado que la consulta de la información sobre los arcos se puede estimar constante la complejidad será siendo el peor caso por ende* |

## Requerimiento 2A

|  |  |
| --- | --- |
| Atributo | Descripción |
| Datos de entrada | *M: Cantidad de puntos a considerar* |
| Datos de salida | *La impresión se realiza por consola* |
| Estimación de complejidad | *Ya que es requerida la conexión para los puntos requeridos minimizando globalmente el costo de conexión, será requerido un árbol que conecte los puntos. Por lo cual se usará un MST para la interconexión, por lo que será implementado el algoritmo de prim con cola de prioridad sumado a la búsqueda de los M más graves:* |

## Requerimiento 1B

|  |  |
| --- | --- |
| Atributo | Descripción |
| Datos de entrada | 1. *P1: Latitud-Longitud del punto 1* 2. *P2: Latitud-Longitud del punto 2* |
| Datos de salida | *La impresión se realiza por consola* |
| Estimación de complejidad | *Se es requerido un mínimum-path entre dos nodos del grafo, por lo que para su búsqueda óptima se usará Dijkstra, dado que la consulta de la información sobre los arcos se puede estimar constante la complejidad será siendo el peor caso por ende «El total de comparendos se estimará en la carga»* |

## Requerimiento 2B

|  |  |
| --- | --- |
| Atributo | Descripción |
| Datos de entrada | *M: Cantidad de puntos a considerar* |
| Datos de salida | *La impresión se realiza por consola* |
| Estimación de complejidad | *Ya que es requerida la conexión para los puntos requeridos minimizando globalmente el costo de conexión, será requerido un árbol que conecte los puntos. Por lo cual se usará un MST para la interconexión, por lo que será implementado el algoritmo de prim sumando la búsqueda de M vértices con mayor número de comparendos:* |

## Requerimiento 1C

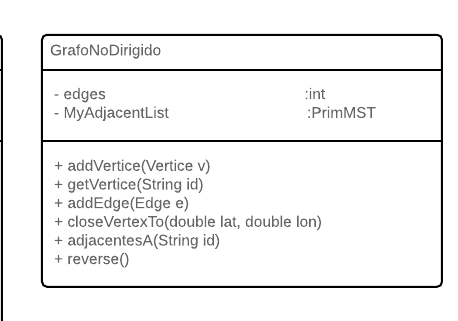
|  |  |
| --- | --- |
| Atributo | Descripción |
| Datos de entrada | *M: Cantidad de puntos a considerar* |
| Datos de salida | *La impresión se realiza por consola* |
| Estimación de complejidad | *Debido a que se hace unta transformación de la distancia física de la malla vial, las rutas mínimas será equivalentes a la menor cantidad de saltos. Para esto comparando Dijkstra y BFS en complejidad temporal hay una equivalencia, sin embargo, por facilidad de implementación, se usará un BFS resultando en:* |

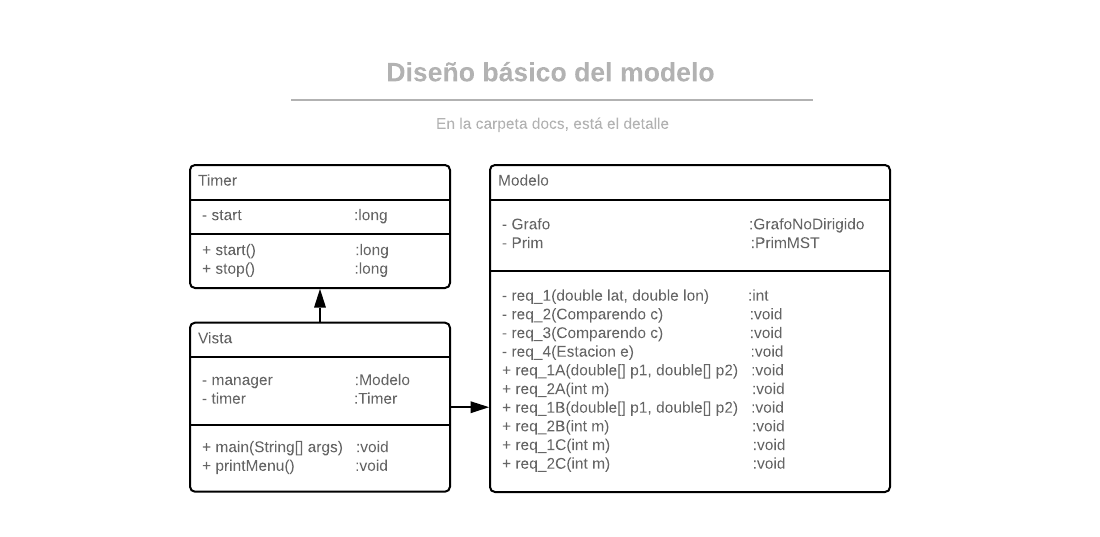
## Requerimiento 2C

|  |  |
| --- | --- |
| Atributo | Descripción |
| Datos de entrada | *No aplica* |
| Datos de salida | *La impresión se realiza por consola* |
| Estimación de complejidad | *Primero se reduce el grafo a uno que cumpla las condiciones prescritas, este proceso analiza todos los arcos y nodos de la red, por ende sumado a la creación de los componentes conexos luego de reducir el grafo. Para esta reducción se decidió utilizar el algoritmo de kosaraju dado por ende una complejidad conjunta:* |

# Estructuras de datos

## Grafo

 Estructura genérica usada en todos los requerimientos al ser la base de la información con la que estos son respondidos

La implementación de este está basada en el uso de un SeparateChainingHashST para una búsqueda optimizada en los nodos del árbol, logrando así una consulta más rápida para cada uno de estos. Esta estructura (SeparateChainingHashST) a su vez usa Queues para algunos de sus métodos. Además dentro de cada nodo del grafo, se usa un RedBlackBST para así optimizar la búsqueda de los comparendos asignados a dicho nodo. Quedamos entonces con el siguiente diagrama.

Y finalmente, siguiendo la arquitectura vista controlador