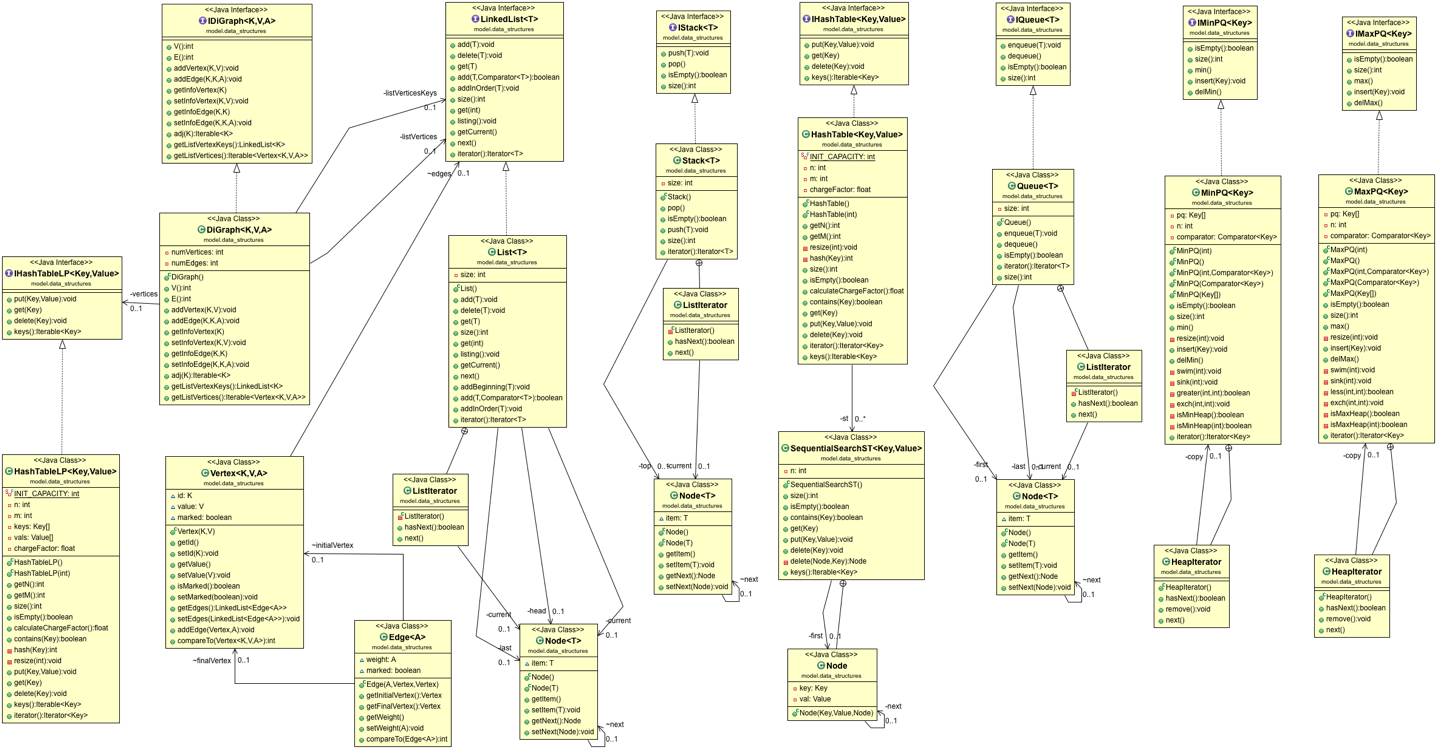
Juan Esteban Méndez 201531707

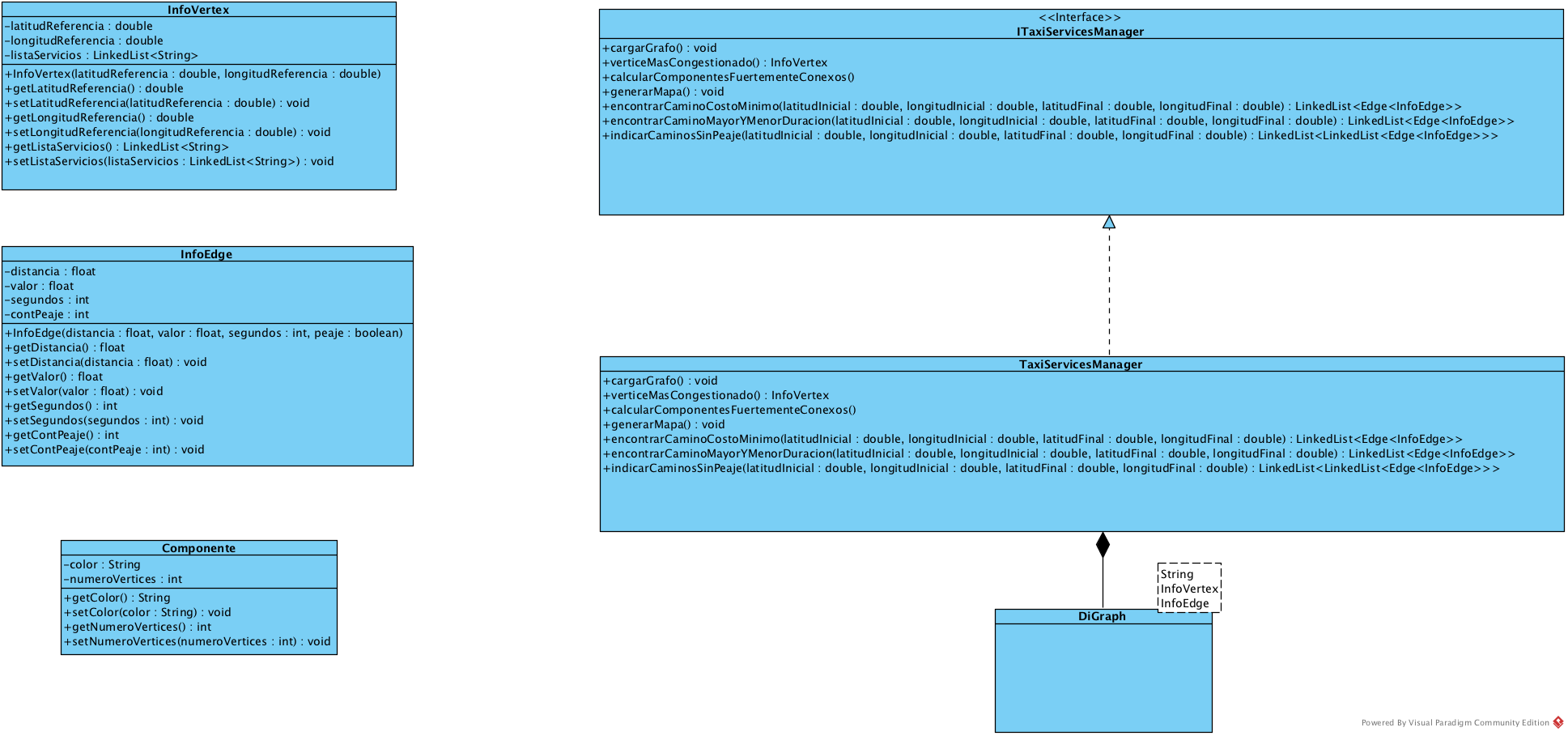
Santiago Rojas

**Documento de Diseño Proyecto 3**

1. Modelo UML de las estructuras de datos:



1. Modelo UML del mundo:



1. Estimación de complejidades de requerimientos:
2. Mostrar en la consola de texto la información del vértice más congestionado en Chicago (aquel que contiene la mayor cantidad de servicios que salen y llegan a él): su (latitud, longitud), total servicios que salieron y total de servicios que llegaron.

* Para el primer requerimiento, se ha decidido usar una lista ordenada con los vértices del grafo, de esta manera se espera hallar el vértice más congestionado con mayor facilidad. Se necesitará un recorrido total de la lista para así poder determinar cual es el vértice con mayor congestión en el grafo, por ende, la complejidad es de ~N siendo N el numero de elementos en la lista.

Complejidad: O(N)

1. Calcule los componentes fuertemente conexos presentes en el grafo. Asígnele un color a los vértices que componen un componente. Retorne una lista en donde en cada nodo, se tiene la información de un componente conexo (color, número de vértices que lo componen). Muestre en la consola de texto el total de componentes identificadas y la información de la lista de componentes conexo.

* Para el segundo requerimiento se usará el algoritmo Kosojuru-Shair, puesto que se necesita calcular los componentes fuertemente conexos y mostrar su información. Siendo V el numero de vértices en el grafo y E el número de arcos, se estima que la complejidad del algoritmo será ~V+E

Complejidad: O(V+E)

1. A partir del grafo cargado al inicio y de los componentes conectados encontrados en el punto 2, genere un mapa coloreado de la red vial de Chicago utilizando Google Maps

* Para el tercer requerimiento, se decidió usar una lista ordenada de vértices. Cada vértice contará a su vez con una lista de servicios. De esta manera se tendrá la información de cada vértice y el número de servicios que tiene para así calcular el radio del círculo a colorear en google maps. Se necesitará un recorrido total de la lista de vértices para determinar el porcentaje con respecto al total de servicios procesados. Siendo N el numero de elementos en la lista, se estima que el requerimiento tendrá una complejidad de ~N.

Complejidad: O(N)

1. Encontrar el camino de costo mínimo (menor distancia) para un servicio que inicia en un punto (latitud, longitud) escogido aleatoriamente de la información cargada del archivo de calles (StreetLines.csv) y finaliza en un punto (latitud, longitud) escogido también de manera aleatoria del archivo de calles. Inicie el camino en el vértice del grafo más cercano a su punto de inicio y finalícela en el vértice más cercano del grafo a su punto de destino. Muestre en la consola de texto el camino a seguir, informando sus vértices, el tiempo estimado, la distancia estimada y el valor estimado a pagar.

* Para el cuarto requerimiento se pide el camino de costo mínimo de dos ubicaciones que entraran por parámetro, por lo cual se usará el algoritmo Dijkstra. El algoritmo de Dijkstra se utiliza para encontrar el camino de costo mínimo entre dos vértices en un grafo. Su complejidad es de ~, siendo E el numero de arcos y V el número de vértices en el grafo.

Complejidad: O (|E| log |V|)

1. Dado un servicio que inicia en un punto (latitud, longitud) escogido aleatoriamente de la información cargada del archivo de calles (StreetLines.csv) y finaliza en un punto (latitud, longitud) escogido también de manera aleatoria del archivo de calles. Aproxime los puntos de inicio y fin a los vértices más cercanos en el grafo y encuentre los caminos de mayor y menor duración entre dichos puntos. Muestre en la consola de texto los dos caminos, y para cada uno de ellos: sus vértices, el tiempo estimado, la distancia estimada y el valor estimado a pagar.

* Para el quinto requerimiento, se necesitan los caminos conformados por arcos más y menos costosos entre 2 vértices que entraran por parámetro, por lo cual se usará el algoritmo Dijkstra. Se espera hacer una pequeña modificación en dicho algoritmo para que también pueda retornar el camino con los arcos más costosos, siendo en este caso la duración del servicio guardado en el arco el criterio de comparación para determinar cual arco es menor o mayor. La complejidad del requerimiento será la misma complejidad que tiene el algoritmo descubierto por Edsger Dijkstra.

Complejidad: O (|E| log |V|)

1. Dado un servicio que inicia en un punto (latitud, longitud) escogido aleatoriamente de la información cargada del archivo de calles (StreetLines.csv) y finaliza en un punto (latitud, longitud) escogido también de manera aleatoria del archivo de calles. Indique si existen caminos entre ambos puntos, en los que no deba pagar peaje. Si dichos caminos existen, retorne una lista con todos ellos, ordenadas de menor a mayor por tiempo y de mayor a menor por costo.

* Para el sexto y último requerimiento, se utilizará el algoritmo DFS (DepthFirst Search) para primero encontrar si los dos puntos se encuentran en la misma componente, lo que significará que existe por lo menos algún camino entre los dos vértices. Este algoritmo tiene una complejidad de ~V+E, siendo V el número de vértices y E el número de arcos en el grafo. Si sí existe por lo menos un camino, se procederá a utilizar un algoritmo que permita encontrar todos los caminos posibles que no contengan peajes.

Complejidad: O(V+E)