Laura Maya

201423854

Carlos Peñaloza

201531973

Al menos tres requerimientos funcionales para cada parte:

Parte A

Nombre	R1. Lectura archivos
Resumen	Se utiliza la librería JSON para leer los archivos y procesar la información
Entradas	
Ruta de los archivos	
Resultados	
Se ha guardado la información de los archivos en las estructuras de datos	

Nombre	R2. Consultar los local authorities
Resumen	Teniendo ordenado el heap de los local authorities de mayor a menor prioridad, lo recorre y va imprimiendo la información de cada uno.
Entradas	
Ninguna	
Resultados	
Se imprimió en consola los local authorities con su respectiva información	

Nombre	R3. Consultar el número de accidentes en un local authority en un año.
Resumen	Dado un año y un local authority, se imprime en consola la cantidad de accidentes que se reportaron en ese año y ese local authority.
Entradas	
El año y el local authority del cual se quiere conocer la información	
Resultados	
Se imprimió en consola la cantidad de accidentes ocurridos en el año y local authority que entraron por parámetro	

Laura Maya

201423854

Carlos Peñaloza

201531973

Nombre	R4. Consultar el flujo de carros y de vehículos en general en un año y un local authority
Resumen	Busca el flujo de carros y flujo de todos los vehículos en un local authority en algún año
Entradas	
Nombre del local authority y año del que se quiere conocer la información	
Resultados	
Se imprime en consola el flujo de carros y de todos los vehiculos del año y el local authority solicitado	

Parte B

Nombre	R1. Lectura archivos
Resumen	Se utiliza la librería JSON para leer los archivos y procesar la información
Entradas	
Ruta de los archivos	
Resultados	
Se ha guardado la información de los archivos en las estructuras de datos	

Nombre	R2. Primera consulta
Resumen	Imprime la información de las autoridades locales de mayor a menor
Entradas	
Resultados	
El usuario ve en la consola la información de las autoridades locales	

Nombre	R3. Segunda consulta

Laura Maya
201423854

Carlos Peñaloza
201531973

Resumen	Busca el número de accidentes en una autoridad local en algún año
Entradas	
Código de la autoridad local y año en el cual se desea realizar la consulta	
Resultados	
Se imprime por consola el número de accidentes ocurridos	

Nombre	R4. Tercera consulta
Resumen	Busca el número de accidentes, flujo de carros y flujo de todos los vehículos en una autoridad local en algún año
Entradas	
Nombre de la autoridad local y año en el cual se desea realizar la consulta	
Resultados	
Se imprime por consola la información solicitada	

Parte C

Nombre	R1. Comparar heap con lista y con arreglo
Resumen	Se compara el tiempo de ejecución del heap implementado con lista y con arreglo cambiando la cantidad de elementos
Entradas	
Resultados	
Se conoce el tiempo de ejecución de ambos heaps	

Nombre	R2. Se comparan las dos formas de hacer la segunda consulta.
Resumen	Se ejecuta 100 veces los puntos 2A y 2B y se obtiene el tiempo promedio de cada uno para compararlos.

Laura Maya
201423854

Carlos Peñaloza
201531973

Entradas
Resultados
Se conoce el tiempo promedio de cada método.

Nombre	R3. Comparar el número de colisiones cambiando el tamaño inicial de las tablas de hash
Resumen	Se inicializan las tablas de hash con diferentes valores y se registran el número de colisiones que ocurren en cada caso
Entradas	
Resultados	
Se conoce el número de colisiones y el tiempo de búsqueda de cada intento	

Nombre	R4. Comparar tiempos tercera consulta
Resumen	Se comparan los tiempos de ejecución de la tercera consulta en la parte A y la parte B
Entradas	
Resultados	
Se conocen y comparan los tiempos de ejecución de la tercera consulta por ambos métodos.	

Estimación de Complejidad temporal de requerimientos funcionales a partir del uso de Estructuras de datos:

La lectura de datos, tiene una complejidad temporal de O(n^2), puesto que los archivos de flujo y accidentes deben ser leídos en su totalidad para guardar cada dato, lo cual implica hacer dos recorridos, con el fin de guardar los datos en la primera columna con su respectivo valor en la

segunda. Durante este proceso además se leen los archivos de pesos, pero estos no tienen un impacto significativo en la complejidad temporal puesto que no se realizan recorridos en ellos.
La primera consulta tiene una complejidad de O(n) puesto que es lo que le toma al heap realizar el ordenamiento de los datos ya que cuando están organizados (lo cual toma log(n) pero ocurrió durante la lectura), solo se debe iterar sobre la lista imprimiendo cada uno de los datos.
La segunda consulta tiene una complejidad temporal constante puesto que los datos ya se encuentran en la tabla de Hash por lo cual solo es necesario realizar get dos veces, con lo cual por medio de la función de Hash se obtiene el valor buscado.
La tercera consulta tiene una complejidad temporal de O(log(n)) puesto que se buscan valores en una tabla de Hash, lo cual toma tiempo constante, y en árboles rojo negro, lo cual toma log(n) al ser esta la altura máxima, y por lo tanto, la cantidad de elementos que se tendría que recorrer en el peor caso.

Explicación/Justificación de las estructuras de datos:

Tabla de Hash: La utilización de tablas de Hash es útil para el proyecto ya que estas permiten buscar e insertar datos con una complejidad temporal de O(1), sin importar el tamaño de la estructura, siempre y cuando se utilice una función de Hash adecuada.

Cola de prioridad: Las colas de prioridad son importantes ya que estas permiten ordenar la lista según se desee (mayor a menor o viceversa) mientras se están leyendo y agregando los datos, lo cual reduce la complejidad temporal requerida en el proyecto anterior al tener que leer y ordenar por aparte antes de poder procesar los datos.

Árbol rojo negro: Las operaciones de insertar, remover y obtener elementos tienen complejidad de O(log(n)). Es una estructura que se balancea automáticamente por lo cual se garantiza que la complejidad temporal no cambiará. Es particularmente útil cuando se deben insertar o eliminar datos frecuentemente, y además ofrece todas las ventajas de un árbol 23 con la facilidad de uso de un árbol de búsqueda binario.

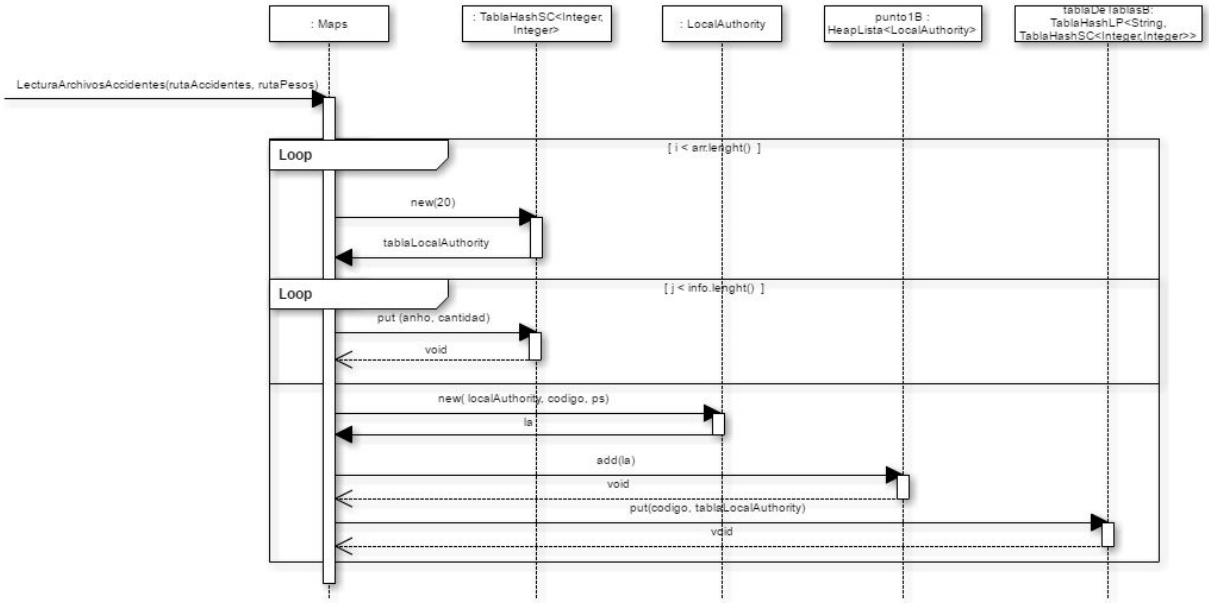
Laura Maya

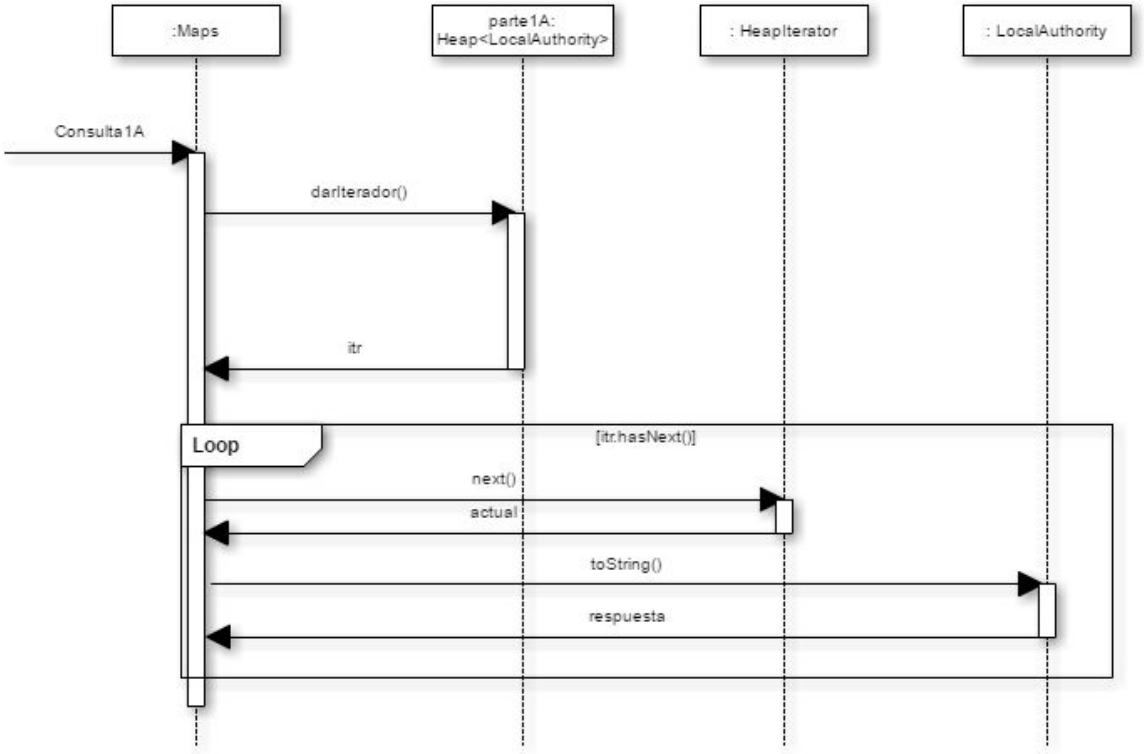
201423854

Carlos Peñaloza

201531973

Diagramas de Secuencia



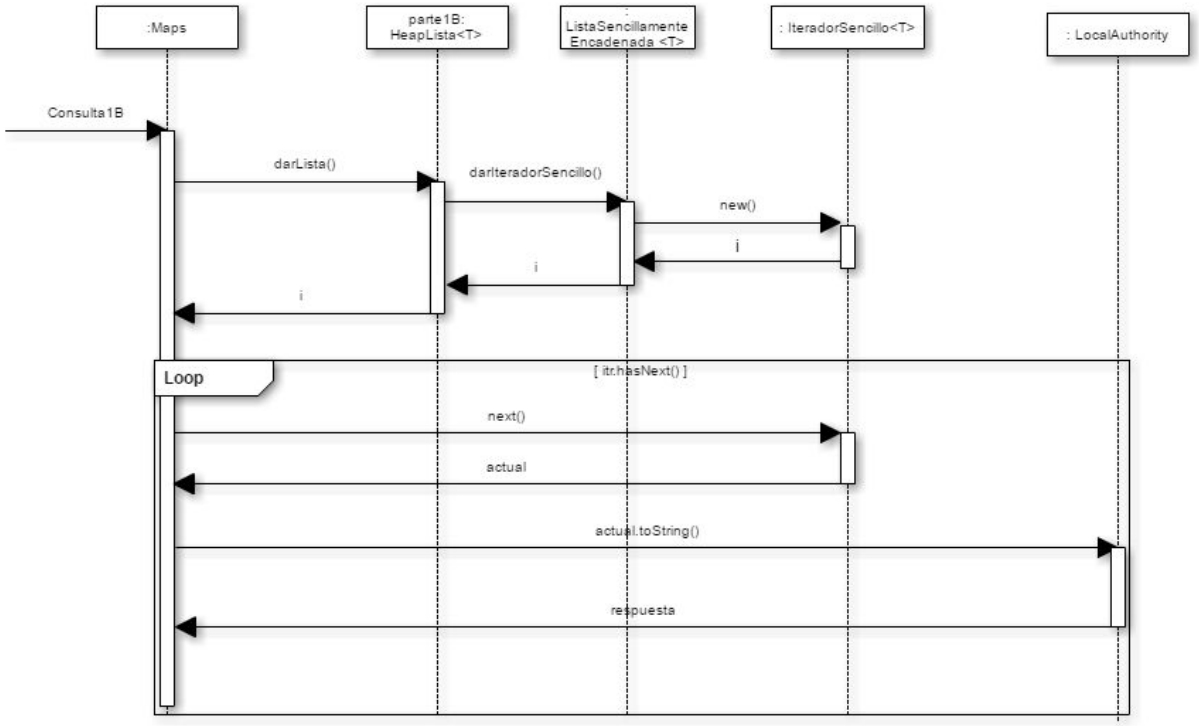


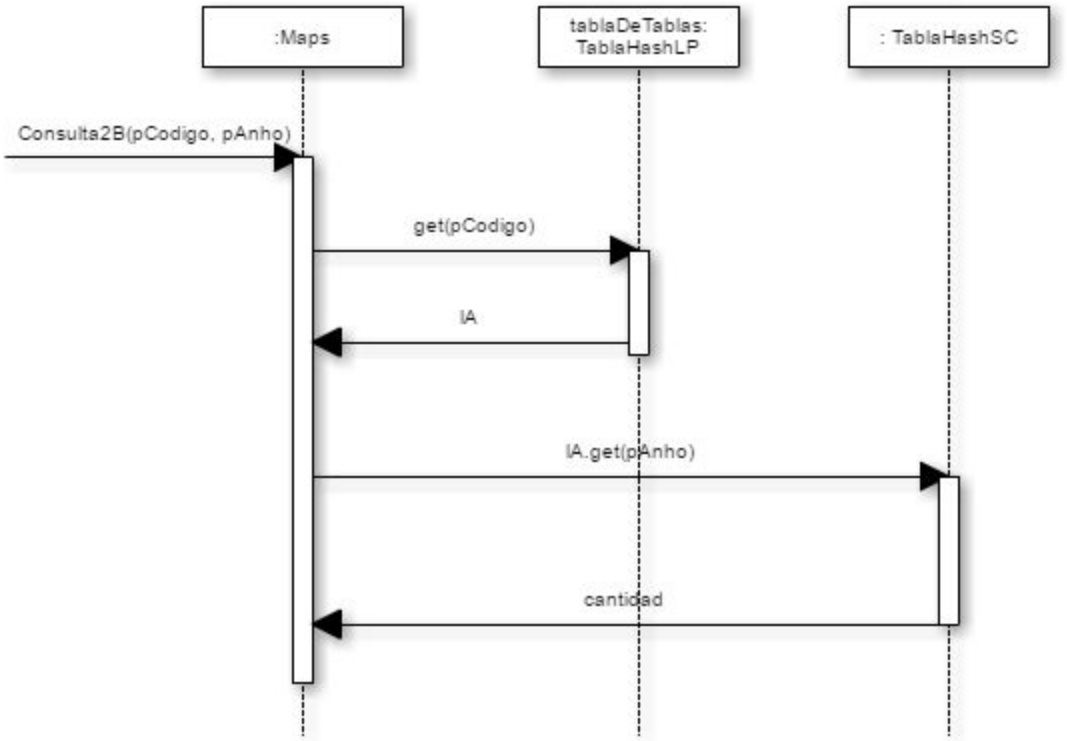
Laura Maya

201423854

Carlos Peñaloza

201531973



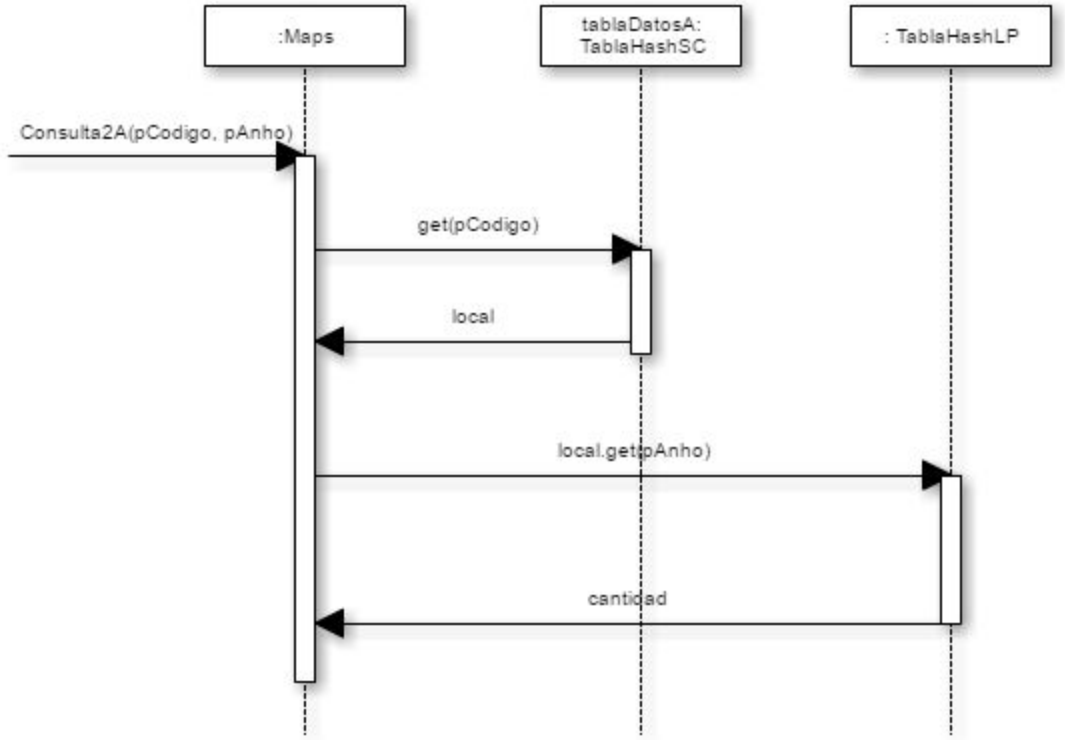


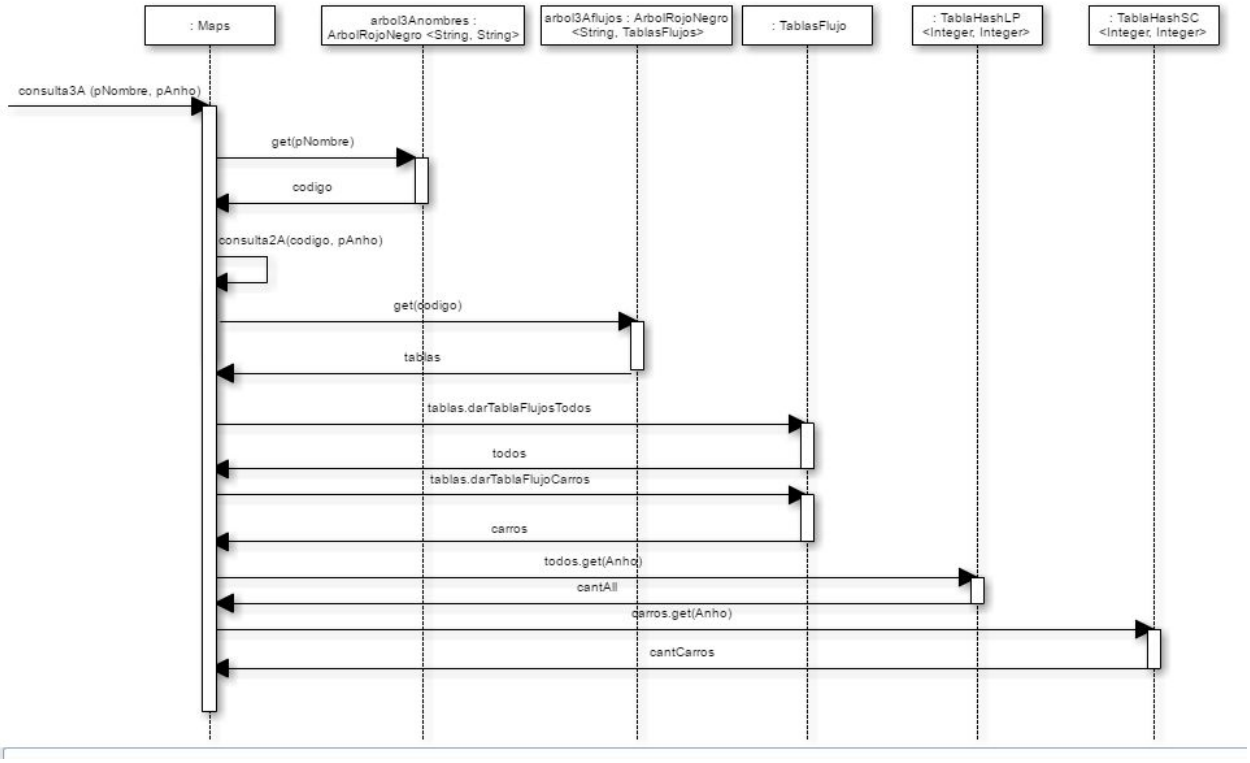
Laura Maya

201423854

Carlos Peñaloza

201531973



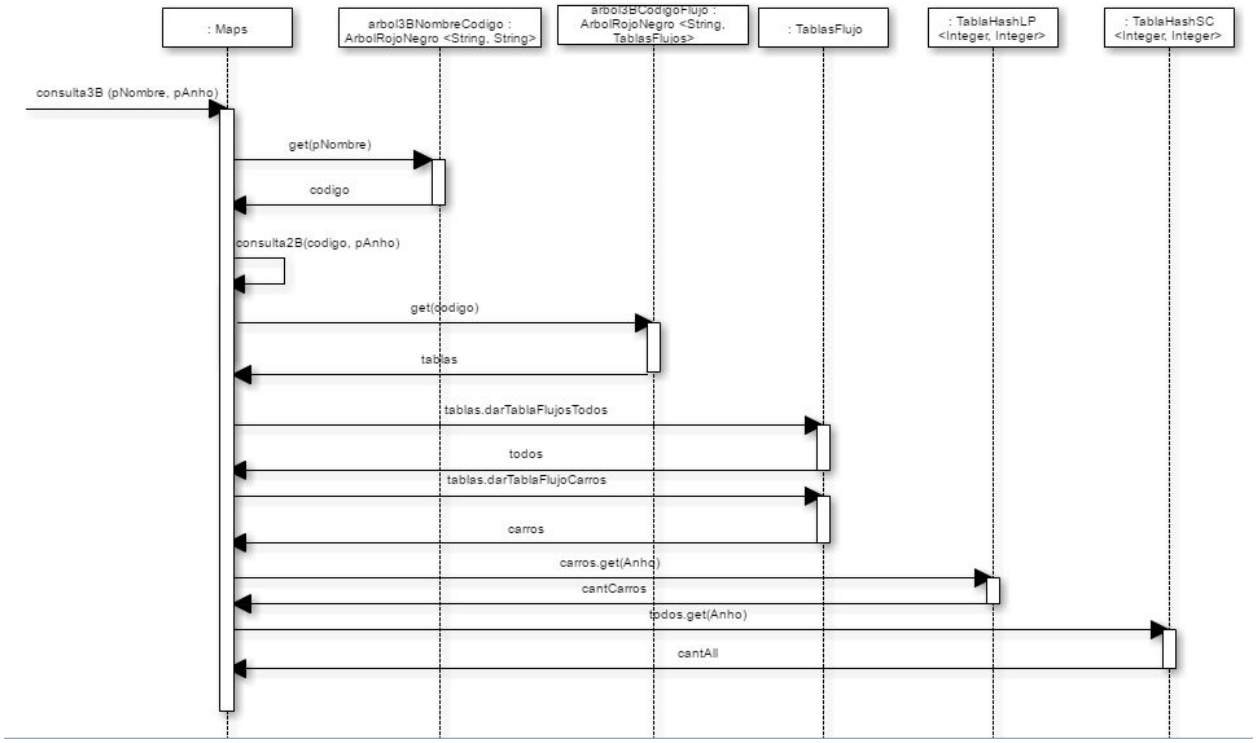


Laura Maya

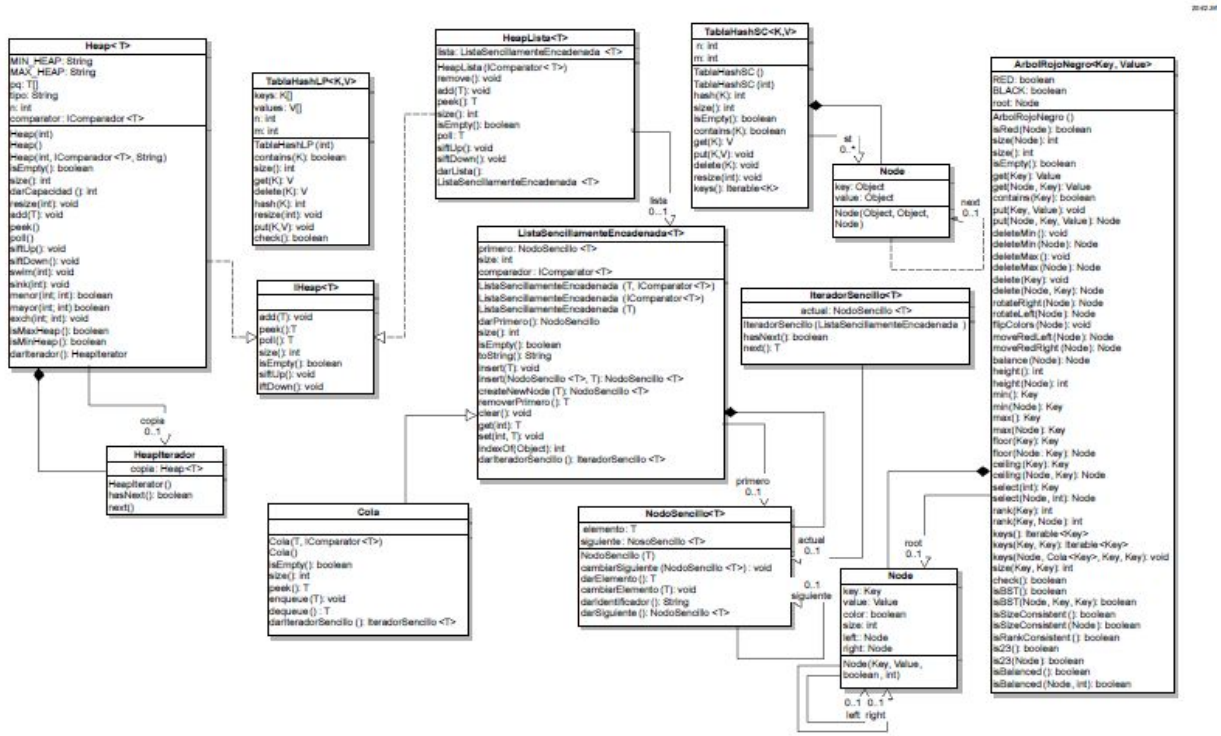
201423854

Carlos Peñaloza

201531973



Diagramas UML



Laura Maya

201423854

Carlos Peñaloza

201531973