UNet

Juan Andrés Bueno Hortua, Juan David Díaz García y Juan Diego Tovar Cárdenas
No. de Equipo Trabajo: 7

I. INTRODUCCIÓN

A continuación se realizarán las especificaciones del problema de aplicación a resolver como proyecto de clase. Para esto se realizará una descripción funcional del proyecto, se presentará el segundo avance realizado, se hará un breve análisis de los resultados obtenidos y una comparación con los resultados de la primera entrega.

II. DESCRIPCIÓN DEL PROBLEMA A RESOLVER

Actualmente el acceso a cierta información académica no es óptima y puede mejorar. El SIA no siempre permite ver el contenido de la asignatura, los conocimientos necesarios para verla o sus prerrequisitos. Esto lleva a que el estudiante tenga dificultades en su inscripción de materias ya que no tiene la información necesaria para tener seguridad sobre las materias que va a ver o desea ver debido a que el sistema no se la proporciona de manera adecuada. De manera análoga, en muchos casos, dada la falta de información de prerrequisitos, el estudiante no sabe qué materias puede ver y pierde la oportunidad de inscribir asignaturas interesantes porque desconocía la posibilidad de inscribirla.

Nuestra propuesta tiene como objetivo dar al estudiante un entorno amigable donde pueda reportar y visualizar su avance en su plan de estudios, así mismo como las materias que puede o no inscribir, además de sus respectivos planes, prerrequisitos y desbloqueos.

III. USUARIOS DEL PRODUCTO DE SOFTWARE

Los usuarios de la aplicación serán estudiantes activos del Departamento Ingeniería Mecánica y Mecatrónica de la Facultad de Ingeniería de la Universidad Nacional de Colombia que deseen organizar y visualizar sus avances de carrera. Podrán hacer uso diario de la aplicación y acceder tanto a sus datos personales como a las de las diferentes asignaturas.

IV. REQUERIMIENTOS FUNCIONALES DEL SOFTWARE

A. *Mostrar la malla curricular de la carrera de manera gráfica*
   1) *Descripción:* se muestra en pantalla la malla curricular y permite navegar por la misma.
   2) *Acciones iniciadoras y comportamiento esperado:* seleccionar el plan de estudio que desea consultar. El programa mostrará las materias correspondientes de dicho plan de forma esquemática por semestres.
   3) *Requerimientos:* acceder a las materias almacenadas de ese plan y su información básica como el nombre y mostrarlas en pantalla.

B. *Mostrar información general de la malla*
   1) *Descripción:* se muestran en colores las materias aprobadas, la nota de las mismas, también o materias disponibles para inscribir.
   2) *Acciones iniciadoras y comportamiento esperado:* el usuario puede elegir entre dos botones dependiendo si quiere observar las materias aprobadas o disponibles para inscribir. Una vez oprimido, la malla colorea las materias que cumplan estas condiciones.
   3) *Requerimientos:* consultar las estructuras de datos que guardan las materias que cumplan dicha condición.

C. *Mostrar y editar información particular de una materia del plan*
   1) *Descripción:* se muestra la tipología de la asignatura, nota de la misma, prerrequisitos, y que asignaturas desbloquea después de ser aprobada. Se considera también incluir una breve descripción de la materia
   2) *Acciones iniciadoras y comportamiento esperado:* una vez en el plan deseado, se selecciona la materia que se desea consultar y esta mostrará toda su información asociada.
   3) *Requerimientos:*
      i. Busqueda parcial en el conjunto de todas las materias y acceso a su información.
      ii. Edición y actualización de las notas.
      iii. Solo admite valores entre 0.0 y 5.0, de lo contrario, impide el registro de la información y avisa al usuario.

D. *Inserción, edición o eliminación de asignatura de libre elección u optativa*
   1) *Descripción:* dado que hay un gran catálogo elegibles en el componente de libre elección, se propone que el estudiante inserte manualmente las materias de libre elección o seleccione las materias optativas cursadas especificando mínimamente el nombre y el número de créditos.
   2) *Acciones iniciadoras y comportamiento esperado:* se seleccionan los créditos de libre elección y se insertan una o varias asignaturas. Se solicitará el nombre y el número de créditos. En caso de superar los 8 se mostrará un cuadro de texto preguntando si se tiene seguridad del valor (sólo serán entradas válidas los números naturales). En el caso de las optativas se puede seleccionar de una lista predefinida. Así mismo se permite la eliminación de las mismas. Se irá añadiendo gráficamente a la malla.
   3) *Requerimientos:*
      i. Búsqueda parcial de las materias cursadas, obtención de la nota de las mismas.
      ii. En caso de ingresar datos no admitidos, se detendrá el registro y se notificará al usuario del error.
      iii. Modificación de la interfaz

E. Calcular P.A.P.A.

1) *Descripción:* se calcula y muestra el P.A.P.A. del estudiante.
2) *Acciones iniciadoras y comportamiento esperado:* al editar alguna nota se recalcula el PAPA. Para consultarlo, se accede mediante un botón.
3) *Requerimientos:* busqueda parcial de las materias cursadas, obtención de la nota de las mismas y cálculo matemático.

F. Mostrar avance de carrera:

1) *Descripción:* dependiendo del número de créditos aprobados respecto a los créditos obligatorios, en cada una de las componentes del plan, 3 gráficos ilustrarán el avance en cada componente y a su vez se mostraran la cantidad de crédito aprobados con los respectivos promedios (PA y PAPA).
2) *Acciones iniciadoras y comportamiento esperado:* una vez se seleccione el plan de estudio deseada, esta opción será desplegada gráficamente para su consulta debajo de la malla.
3) *Requerimientos:* consultar listas que guardan las materias aprobadas y hacer los cálculos requeridos para mostrarlos en la interfaz.

V. AVANCE EN LA IMPLEMENTACIÓN DE LA INTERFAZ DE USUARIO

Mockups primera entrega:

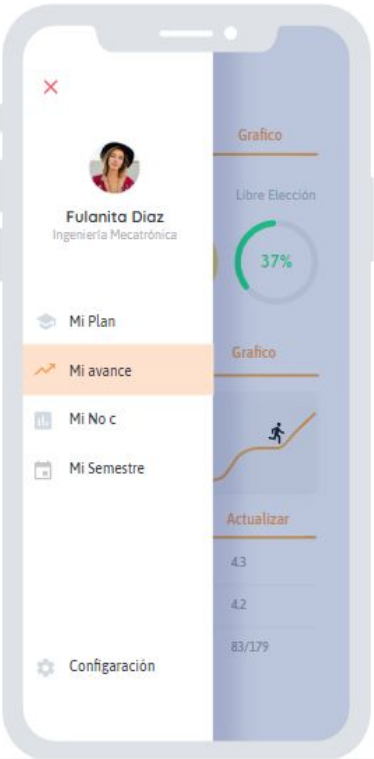

Fig. 1 Mockup opciones de la aplicación

Figura 2: Mockup información disponible en la sección mi plan

Fig. 3 Mockup información disponible en la sección mi avance

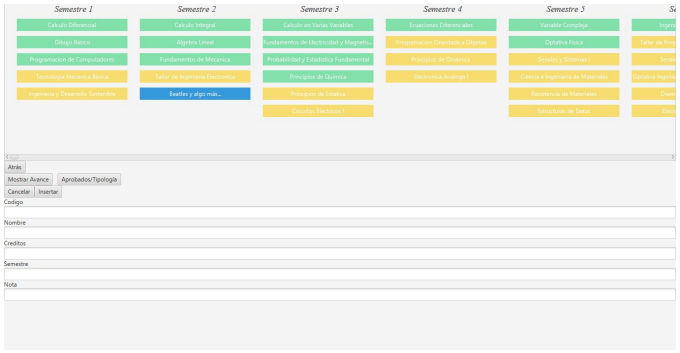

Fig. 4 Interfaz gráfica mostrando las materias según tipología

Fig. 5 Interfaz gráfica mostrando las materias según el estado de aprobado, reprobado, desbloqueadas y bloqueadas.

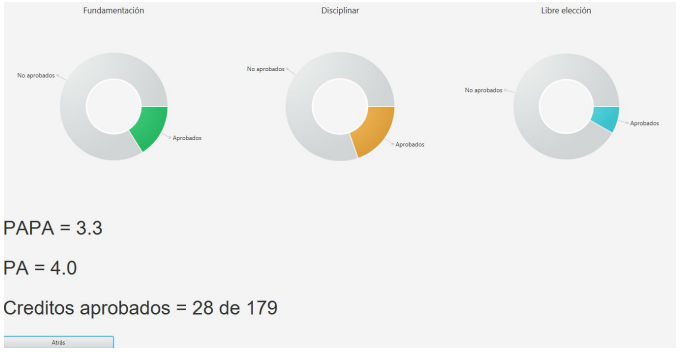

Fig. 6 Pantalla que muestra el avance en cada componente del plan de manera porcentual

VI. ENTORNOS DE DESARROLLO Y DE OPERACIÓN

El software se desarrollará en el IDE de NetBeans haciendo uso de JavaFX para el desarrollo de la interfaz gráfica y el programa podrá ser ejecutado desde cualquier dispositivo con Java.

VII. DESCRIPCIÓN GENERAL DEL PROTOTIPO FINAL IMPLEMENTADO

En la primera entrega, se hizo entrega de un primer prototipo que cumplía los requerimientos A y D de forma muy rudimentaria a nivel gráfico como de las estructuras utilizadas.

En el segundo avance, se hizo el cambio de dichas estructuras por unas más eficientes como árboles y pilas sin afectar la funcionalidad. No obstante, algunas implementaciones, a consideración del equipo, no fueron alteradas ya que coordinaban adecuadamente con las nuevas estructuras en términos de eficiencia. Lo anterior nos permitió desarrollar el resto de requerimientos inicialmente planteados, así logramos incorporar las funcionalidades B, C, E y F por el lado lógico, además de mejorar las anteriormente incorporadas.
En esta entrega introdujimos a las tablas Hash como estructura de datos principal en la aplicación de tal suerte que se logró una complejidad constante para todas las funcionalidades que fuera posible, así mismo desarrollamos una nueva funcionalidad que se tenía considerada, esta es la visualización de las materias disponibles para ver.

En conclusión, en esta entrega se mejoraron y optimizaron las funcionalidades que ya habían sido implementadas gracias al uso de las tablas Hash permitiendo así un prototipo más ligero y eficiente.

VIII. NUEVAS ESTRUCTURAS DE DATOS IMPLEMENTADAS.

A. *Tablas Hash*

Se implementaron para la funcionalidad de búsqueda y en general para el almacenamiento de las materias. Cada materia es un objeto definido con diferentes atributos tales como: Nombre, código, créditos, tipología, prerrequisito y nota. Estos objetos son Hasheados utilizando una función que toma como parámetro el código de la materia y retorna un entero que indica la posición en la que se almacenará el objeto. La función de Hash tiene forma:

$$h(x) = ((ax + b) \bmod c) \bmod m$$

para nuestro caso particular tomamos:

$$h(x) = ((23\,x + 53) \bmod 12990007) \bmod m$$

Donde m corresponde al tamaño de la tabla.

Así mismo para el tratamiento de las colisiones empleamos direccionamiento abierto con sondeo cuadrático.

B. *Grafos*

Para la verificación de prerrequisitos se usó un grafo no dirigido donde cada vértice es una materia y cada arista es una relación de prerrequisito, cada materia guarda las materias que tiene como prerrequisitos y las materias de las cuales es prerrequisito.

IX. PRUEBAS DEL PROTOTIPO

En la siguiente tabla se presentan los tiempos de ejecución de las tres principales funcionalidades del prototipo usando listas árboles y tablas hash que corresponden con la estructura principal implementada en cada entrega:

TABLA I
Tiempos de ejecución en milisegundos de las principales funcionalidades para distintos volúmenes de datos

N datos	Tiempos de ejecución por operaciones [ms]					
	Cargar materias			Consultar materias		
	Listas	Árboles	Hash	Listas	Árboles	Hash
500	287	52	1	3	0	0.0083
1k	160	58	1	4	0	0.0023
5k	407	75	2	5	0	0.0081
10k	469	80	2	5	0	0.0031
50k	11725	98	1	14	0	0.0037
100k	78457	104	2	24	1	0.0035
500k	1232902	126	66	OME	2	0.0021
1M	3400255	138	160	OME	2	0.0030
Big O	O(n^2)	O(nlog(n))	O(n)	O(n)	O(log(n))	O(1)

OME = OutOfMemoryError

N datos	Tiempos de ejecución por operaciones [ms]		
	Ingresar electiva		
	Listas	Árboles	Hash
500	0	0	0.006
1k	1	1	0.004
5k	1	1	0.002
10k	3	1	0.001
50k	13	1	0.001
100k	17	8	0.001
500k	OME	9	0.001
1M	OME	9	0.0011
Big O	O(n)	O(log(n))	O(1)

OME = OutOfMemoryError

A continuación se presentan las pruebas realizadas sobre las nuevas estructuras de datos que no tienen contraparte en la primera entrega:

TABLA II
Tiempos de ejecución en milisegundos de las nuevas funcionalidades para distintos volúmenes de datos

N datos	Tiempos de ejecución por operaciones [ms]		
	Cargar materias	Eliminar materia raíz	Ingresar electiva
	Max Heap	Max Heap	Max Heap
500	8	0	0
1k	19	0	0
5k	28	0	0
10k	715	0	0
50k	10426	0	1
100k	21126	0	1
500k	360299	0	2
1M	1130053	0	2
Big O	O(nlog(n))	O(log(n))	O(log(n))

X. ANÁLISIS COMPARATIVO Y RESULTADOS DE PRUEBAS

El uso de las tablas hash como estructura de datos principal en la aplicación trajo consigo un aumento rotundo en la eficiencia de todas las operaciones basadas en búsqueda o acceso a la materias.

Inicialmente con el uso de estructuras de datos lineales la operación de búsqueda tenía una complejidad lineal, posteriormente con el uso de árboles AVL logramos una complejidad logarítmica, ahora gracias a las tablas Hash logramos un acceso constante a las asignaturas. Por lo que se puede concluir que respecto a la complejidad y así mismo respecto al tiempo necesario se logra una gran mejora.

Sin embargo desde el punto de vista de la memoria o espacio requerido, en las dos primeras dos avances con el empleo de los árboles y listas solo se requeria el espacio para guardar las materias, es decir, si teníamos una set de n materias , requerimos un árbol o lista de n nodos, sin embargo con las tablas en necesario emplear 2n espacios dado que buscamos que siempre el factor de carga de la tabla sea menor a 0.5 a fin de evitar las colisiones. Si bien en ambos casos el espacio requerido es de orden lineal, consideramos pertinente tener en cuenta esta consideración.

Por último las funcionalidades referentes a la interfaz gráfica tales como el dibujo de las materias, siempre tendrá una complejidad constante dado que en necesario acceder a cada materia para obtener la información que definirá la manera en que será dibujada(dependiendo la selección del usuario podrá ver tipologías, estado de aprobado, o incluso las materias disponibles para ver).

Así en conclusión con el empleo de las tablas hash se permite la edición, inserción y acceso a las materias y su información teniendo una complejidad constante, sin embargo esto se traduce en un costo mayor de espacio necesario.

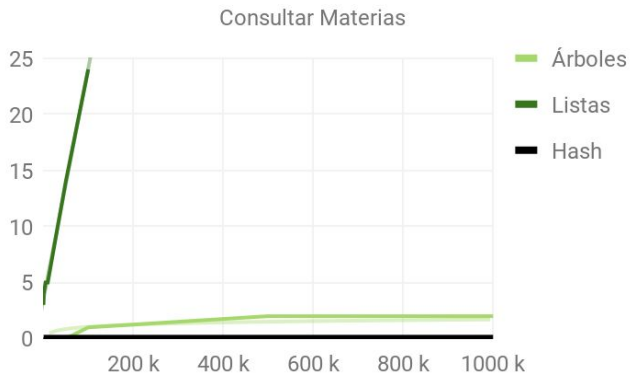

Fig. 7. Comparación de tiempo de ejecución (en ms) entre árboles, listas y hash para la funcionalidad consultar materias

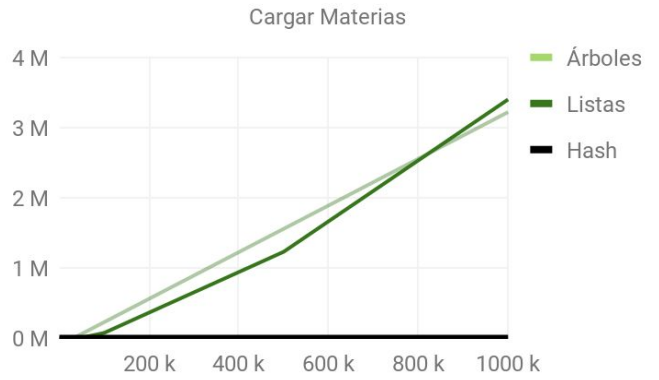

Fig. 8. Comparación de tiempo de ejecución (en ms) entre entre árboles, listas y hash para la funcionalidad cargar materias

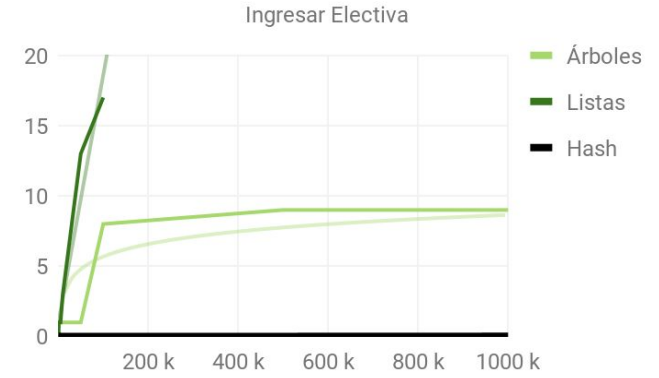

Fig. 9. Comparación de tiempo de ejecución (en ms) entre entre árboles, listas y hash para la funcionalidad ingresar electiva

Estos fueron los resultados de las funcionalidades probadas en los Max Heaps para diferentes volúmenes de datos:

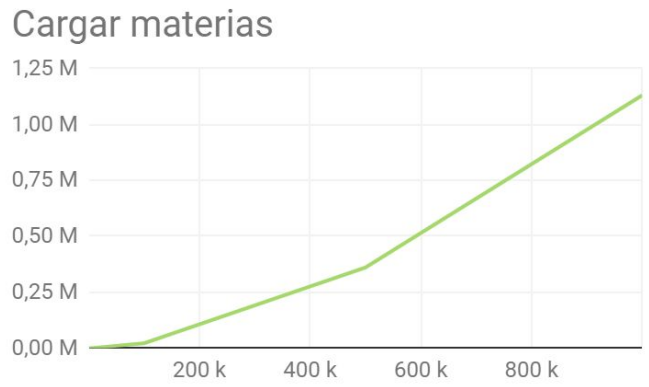

Fig. 10 Tiempo de ejecución (en ms) según la cantidad de datos para la funcionalidad cargar materias en el Heap

Fig. 11. Tiempo de ejecución (en ms) según la cantidad de datos para la funcionalidad eliminar la materia de mayor prioridad en el Heap

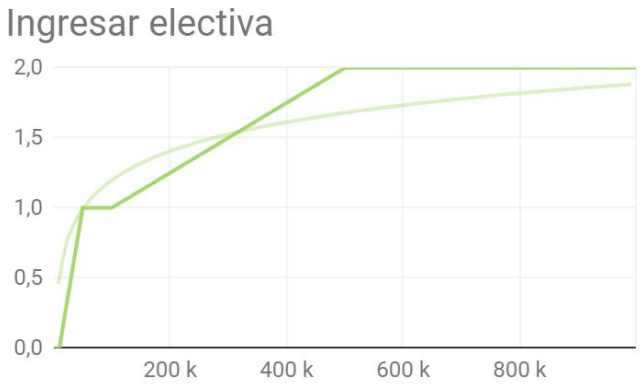

Fig. 12 Tiempo de ejecución (en ms) según la cantidad de datos para la funcionalidad ingresar una electiva en el Heap

Estos fueron los resultados de las funcionalidades probadas en los Árboles AVL para diferentes volúmenes de datos:

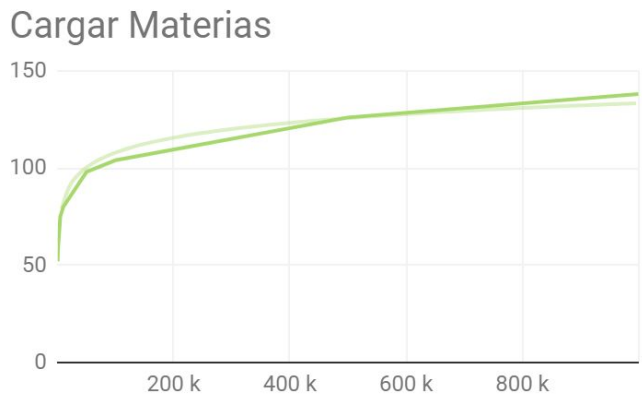

Fig. 13 Tiempo de ejecución (en ms) según la cantidad de datos para la funcionalidad cargar materias usando Árboles AVL

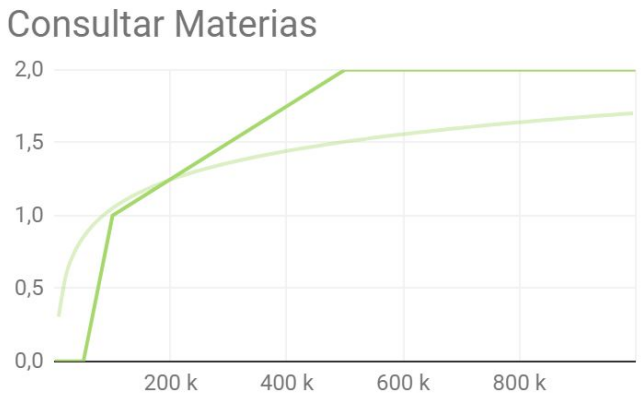

Fig. 14. Tiempo de ejecución (en ms) según la cantidad de datos para la funcionalidad consultar materias usando árboles AVL

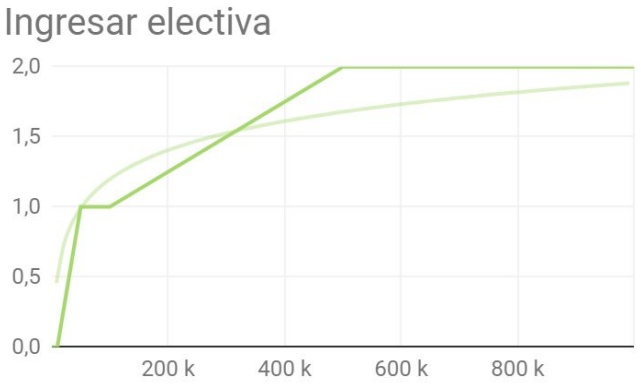

Fig. 15. Tiempo de ejecución (en ms) según la cantidad de datos para la funcionalidad ingresar una electiva usando árboles AVL

Estos fueron los resultados de las funcionalidades probadas en las tablas hash para diferentes volúmenes de datos:

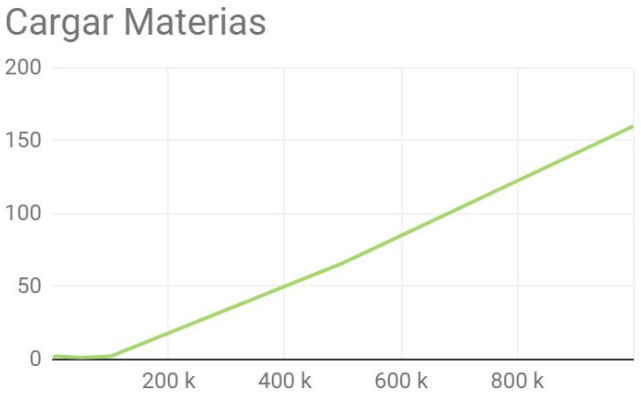

Fig. 16 .Tiempo de ejecución (en ms) según la cantidad de datos para la funcionalidad cargar materias usando tablas hash

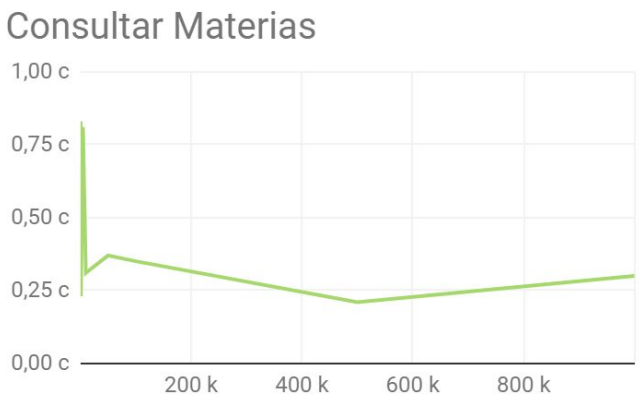

Fig. 17. Tiempo de ejecución (en ms) según la cantidad de datos para la funcionalidad consultar materias usando tablas hash

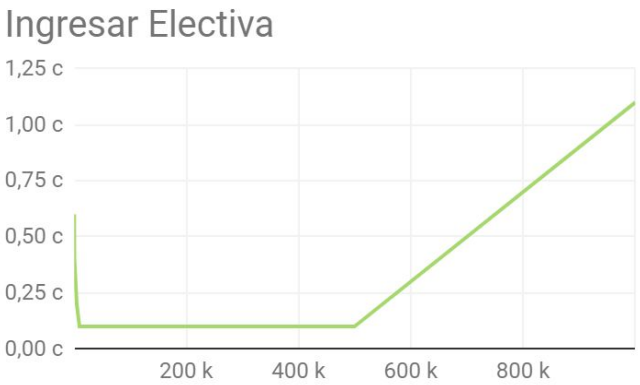

Fig. 18. Tiempo de ejecución (en ms) según la cantidad de datos para la funcionalidad ingresar electiva usando tablas hash

XI. DIFICULTADES Y LECCIONES APRENDIDAS

A continuación se resumen las dificultades de la segunda entrega del proyecto de clase y qué lección se aprendió por cada una de ellas:

TABLA III
Dificultades y lecciones aprendidas durante la tercera entrega del proyecto de clase

Dificultad	**Lección aprendida**
Dificultades con la aplicación de Tablas Hash: Rehash.	Se interiorizaron los conceptos de las tablas hash y la implementación de sus métodos
Error de casteo de objetos al llamarlos desde las tablas Hash	Trabajar con métodos Get y Set para mantener el encapsulamiento
Diferencias creativas al momento de decidir cómo implementar el hash (Función de hash y tamaño)	Saber lidiar con las diferentes ideas que cada integrante tiene priorizando el respeto y la escucha