Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

Método de la Ingeniería

Fase 1: Identificación del problema

- **Problema**

Una nueva compañía muy importante de libros ha decidido abrir sus puertas en la sultana del Valle, esta compañía implementa un método muy eficiente e innovador para atender a sus clientes, la cual consiste en 4 fases:

- Fase 1: Esta fase consiste en la selección de los libros por medio de una gran pantalla donde el cliente puede ver los libros que desea.
- Fase 2: En esta fase el usuario busca sus libros de manera rápida y eficiente.
- Fase 3: En esta fase el usuario ingresa a la cola para esperar la caja que le toca para pagar los libros.
- Fase 4: En esta fase el usuario paga los libros.

La empresa le quiere mostrar a sus clientes el como funciona esta nueva atracción en la cuidad, para esto la compañía requiere un simulador del proceso de compra de libros.

- **Requerimientos Funcionales**

Nombre	***R1***
Resumen	Encontrar el precio total de todos los libros que compró el cliente
Entrada	Precios de los libros a comprar
Salida	Monto total de la compra

Nombre	***R2***
Resumen	Encontrar libro
Entrada	Identificador de Estantería ISBN del libro
Salida	Libro

Nombre	***R3***
Resumen	Encontrar Estantería
Entrada	Identificador de Estantería
Salida	Ninguna

Nombre	***R4***
Resumen	Añadir Cliente a la cola
Entrada	Cliente
Salida	Ninguna

Nombre	***R5***
Resumen	Desencolar cliente
Entrada	Ninguna
Salida	Cliente

Nombre	***R6***
Resumen	Añadir libro al carrito de compras
Entrada	Libro
Salida	Ninguna

Nombre	***R7***
Resumen	Desapilar libro al carrito de compras
Entrada	Ninguna
Salida	Libro

- **Requerimientos no funcionales**

Nombre	**RNF1**
Resumen	La complejidad temporal de las estructuras de datos utilizadas para la solución del problema debe ser 0(1)

Nombre	**RNF2**
Resumen	La interfaz gráfica que permita ingresar al usuario el número de casos de prueba

Nombre	**RNF3**
Resumen	La interfaz gráfica que permita ingresar al usuario el número de cajeros disponibles

Nombre	**RNF4**
Resumen	La interfaz gráfica que permita ingresar al usuario ingresar el número de estanterías con su identificador y el número de libros en dicha estantería

Nombre	**RNF5**
Resumen	La interfaz gráfica que permita ingresar al usuario ingresar el número de ISBN de los libros, seguido del precio de cada uno y su respectiva cantidad de ejemplares

Nombre	**RNF6**
Resumen	La interfaz gráfica que permita ingresar al usuario el número de clientes que ingresa a la libreria

Nombre	**RNF7**
Resumen	La interfaz gráfica que permita ingresar al usuario ingresar la cedula de cada cliente ingresad seguido de los códigos ISBN de cada libro que va a comprar

Fase 2: Recopilación de información

- **Definiciones**

  Librería: Una librería es un establecimiento comercial o tienda que se dedica a la venta de libros. Existen librerías de todo tipo, desde pequeños locales con pocos ejemplares hasta edificios enteros que ofrecen miles de publicaciones. Las librerías pueden pertenecer a una cadena con muchas sucursales o tener un único punto de venta.

  Hay librerías generalistas, que comercializan libros de diversas temáticas, mientras que otras se especializan en temas específicos. Es posible, en este sentido, encontrar librerías deportivas, librerías científicas, etc.

  Las librerías también pueden vender libros usados. Estas tiendas, por lo general, se dedican a comprar ejemplares a un cierto precio para revenderlo luego a uno mayor, obteniendo una ganancia con la diferencia.

  Las librerías virtuales, por su parte, se dedican a vender libros a través de Internet. Dichos libros pueden ser físicos (impresos) o digitales (para leer en la computadora o en dispositivos electrónicos).

  Libro: Un libro es una obra impresa, manuscrita o pintada en una serie de hojas de papel, pergamino, vitela u otro material, unidas por un lado (es decir, encuadernadas) y protegidas con tapas, también llamadas cubiertas. Un libro puede tratar sobre cualquier tema.

  Según la definición de la Unesco, un libro debe poseer 25 hojas mínimo (49 páginas), pues de 24 hojas sería un folleto y de una hasta cuatro páginas se consideran hojas sueltas (en una o dos hojas).

  También se llama "libro" a una obra de gran extensión publicada en varias unidades independientes, llamados "tomos" o "volúmenes". Otras veces se llama también "libro" a cada una de las partes de una obra, aunque físicamente se publiquen todas en un mismo volumen (ejemplo: Libros de la Biblia).

Hoy en día, no obstante, esta definición no queda circunscrita al mundo impreso o de los soportes físicos, dada la aparición y auge de los nuevos formatos documentales y especialmente de la World Wide Web. El libro digital o libro electrónico, conocido como *e-book*, está viendo incrementado su uso en el mundo del libro y en la práctica profesional bibliotecaria y documental. Además, el libro también puede encontrarse en formato audio, en cuyo caso se denomina audiolibro.

Fila: El vocablo francés *file* llegó a nuestro idioma como fila. Así se denomina a una serie ordenada de individuos o elementos que se disponen en línea.

Las filas suelen ser columnas de personas. Se trata de un método habitual de organización cuando tienen que establecerse turnos o cuando hay que lograr un orden para realizar algo. En los supermercados, por citar un caso, se forman filas frente a las cajas. Los clientes van realizando sus compras y, cuando llega el momento de pagar, se dirigen hacia el sector correspondiente: si otros compradores llegaron antes, se irá armando la fila espontáneamente.

Referencias

Gardey, J. P. (17 de 04 de 2017). *Definicion.De*. Obtenido de https://definicion.de/fila/

Porto, J. P. (03 de 07 de 2014). *Definiciones.De*. Obtenido de https://definicion.de/libreria/

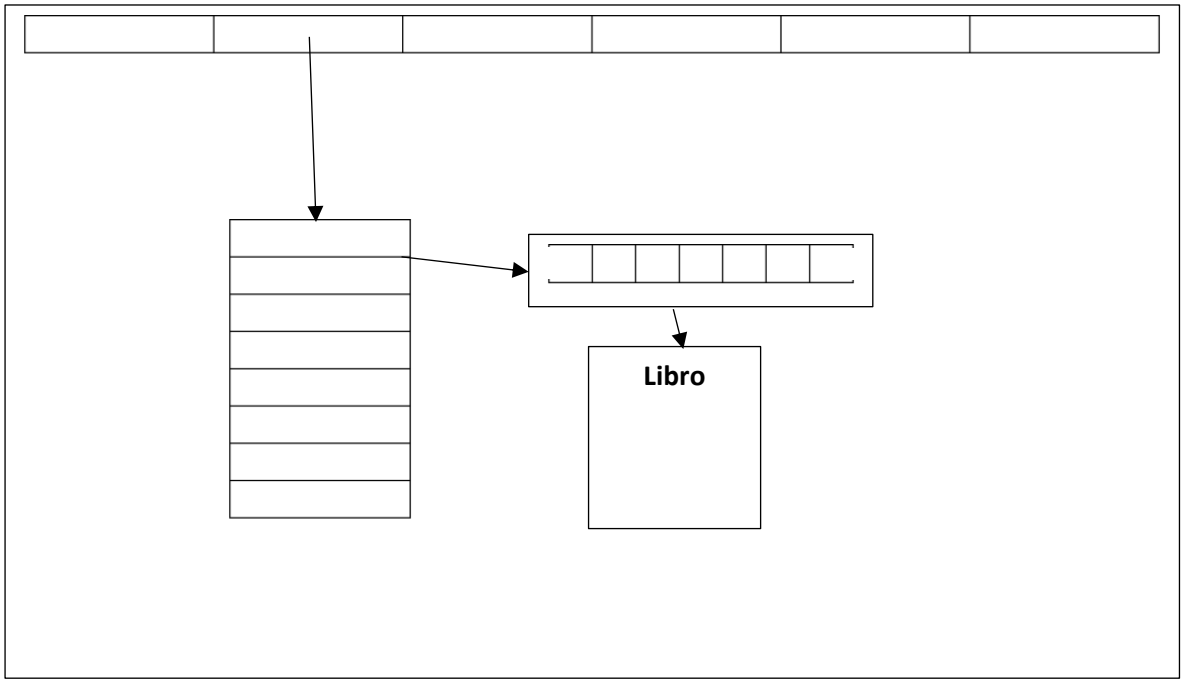
Villarruel, C. (19 de 01 de 2019). *Wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Libro

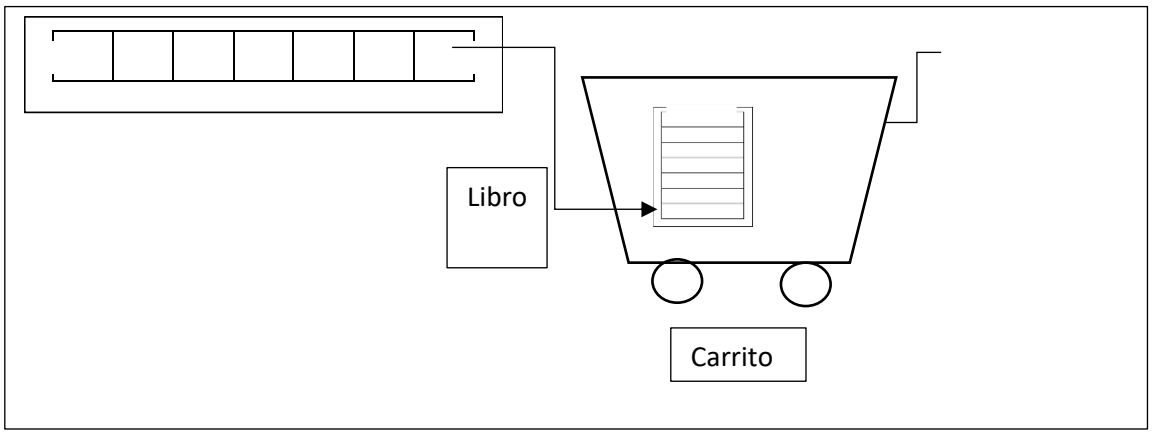
Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

Fase 3: Búsqueda de soluciones creativas

Alternativa 1:

En esta alternativa hemos optado por almacenar las estanterías en un arreglo de tamaño n en donde n representa el número de estanterías, en cada posición del arreglo se ubica una estantería representada como una tabla hash, cada slot de la tabla hash contiene una cola con la cantidad de libros agrupados de acuerdo a su número de ISBN. Por otra parte, cada libro que se extraiga de la cola va a ser agregado a una pila contenida en el carrito de compra.

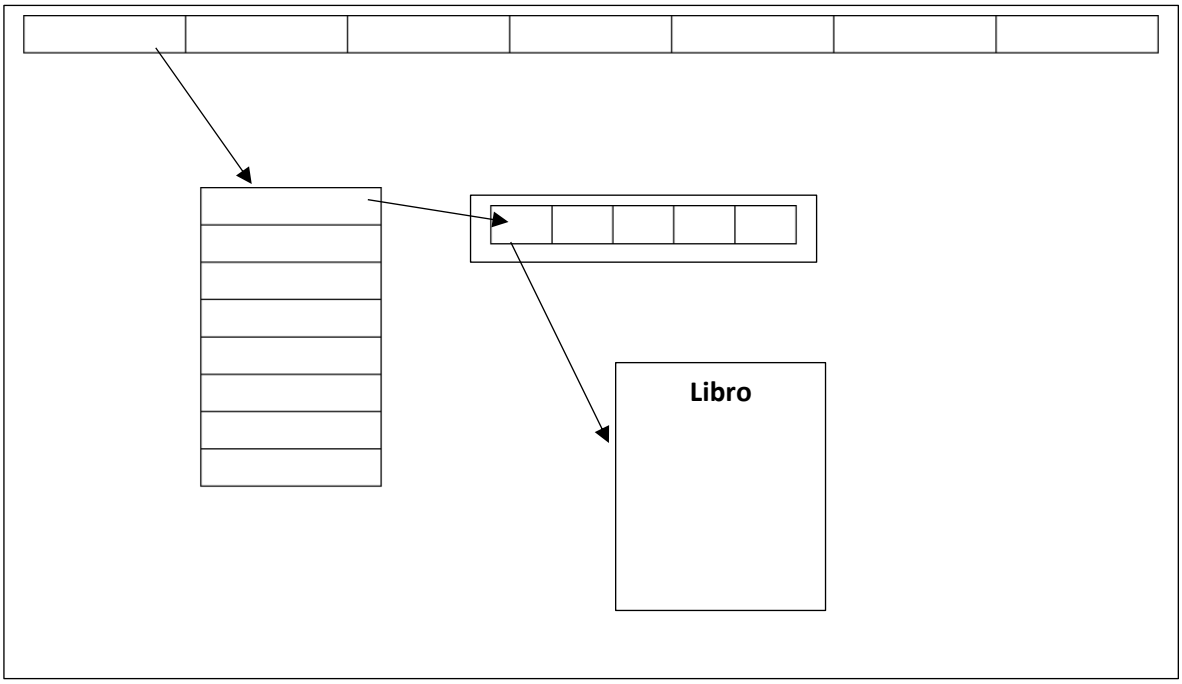


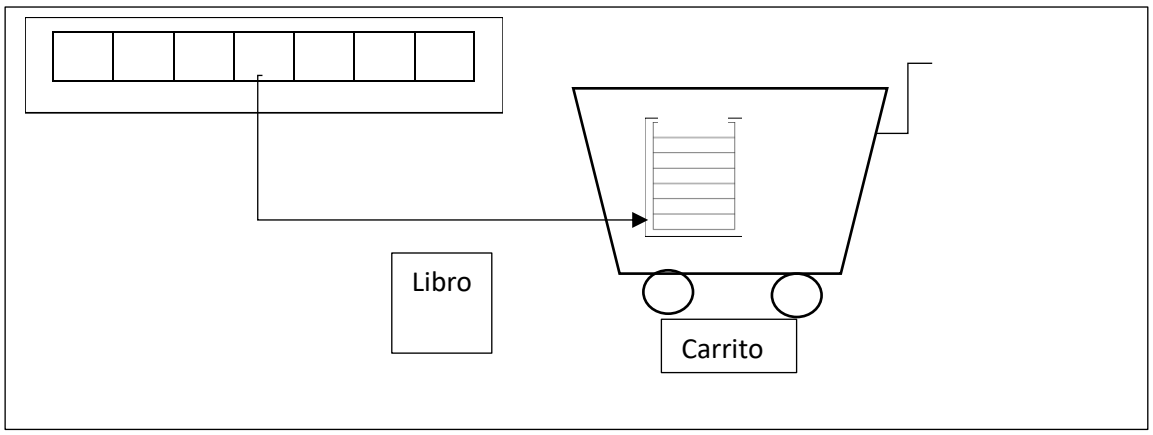


Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

- **Alternativa 2:**

En esta segunda alternativa se contempla la idea de almacenar todas las estanterías de libros en una tabla hash, dentro de cada posición se encuentra otra tabla hash que a su vez contiene una última tabla hash con el número de libros correspondientes al ISBN en cada slot. Por otra parte, para almacenar cada libro en el carrito de compra. Se utilizará una pila dentro del carrito.

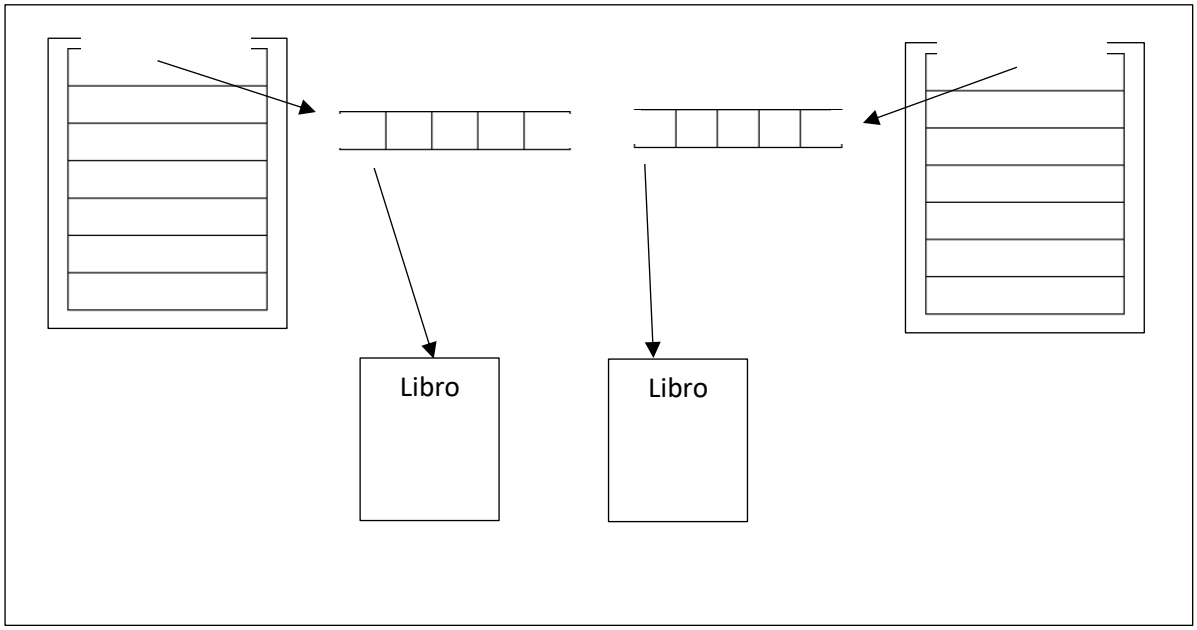


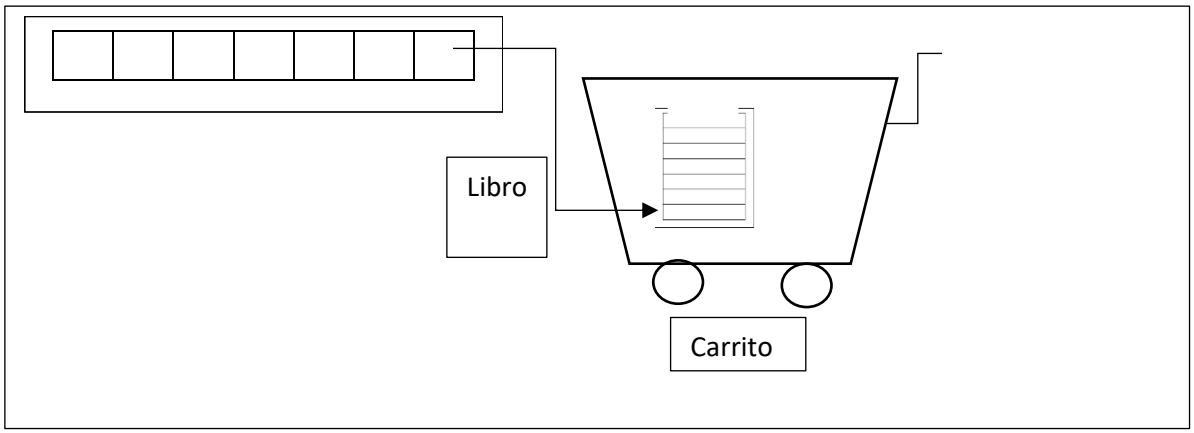


Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

- **Alternativa 3:**

En esta solución la idea es almacenar los libros en una cola, dicha cola está almacenada en cada slot de una pila que representa la estantería, por lo cual cada estantería representaría una pila diferente. Por otra parte, para guardar los libros se utilizará una pila contenida por el carrito de compra.

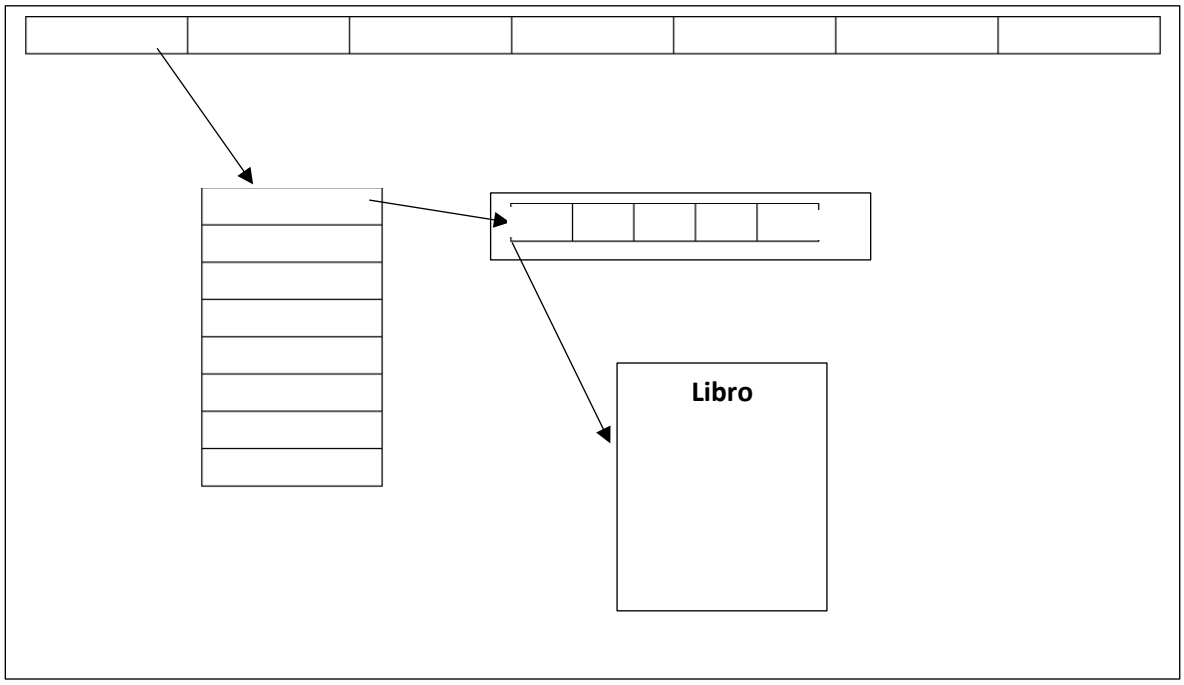


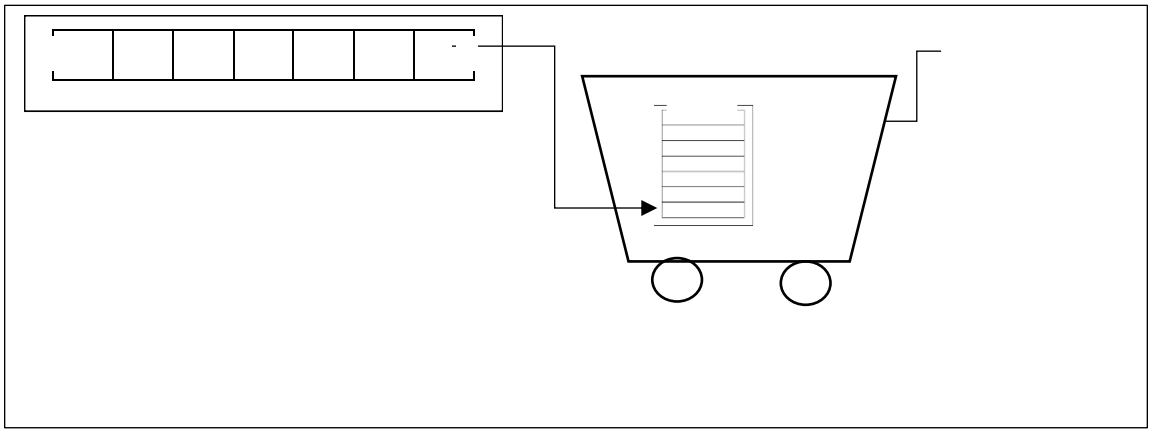


Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

- **Alternativa 4:**

Esta alternativa consiste en almacenar una estantería en cada slot de una tabla hash, cada estantería va a ser representada también por una tabla hash que a su vez va a contener una cola en su slot correspondiente, dicha cola va a representar los libros almacenados en ella. Para el almacenamiento de cada libro en el carrito será utilizada una pila

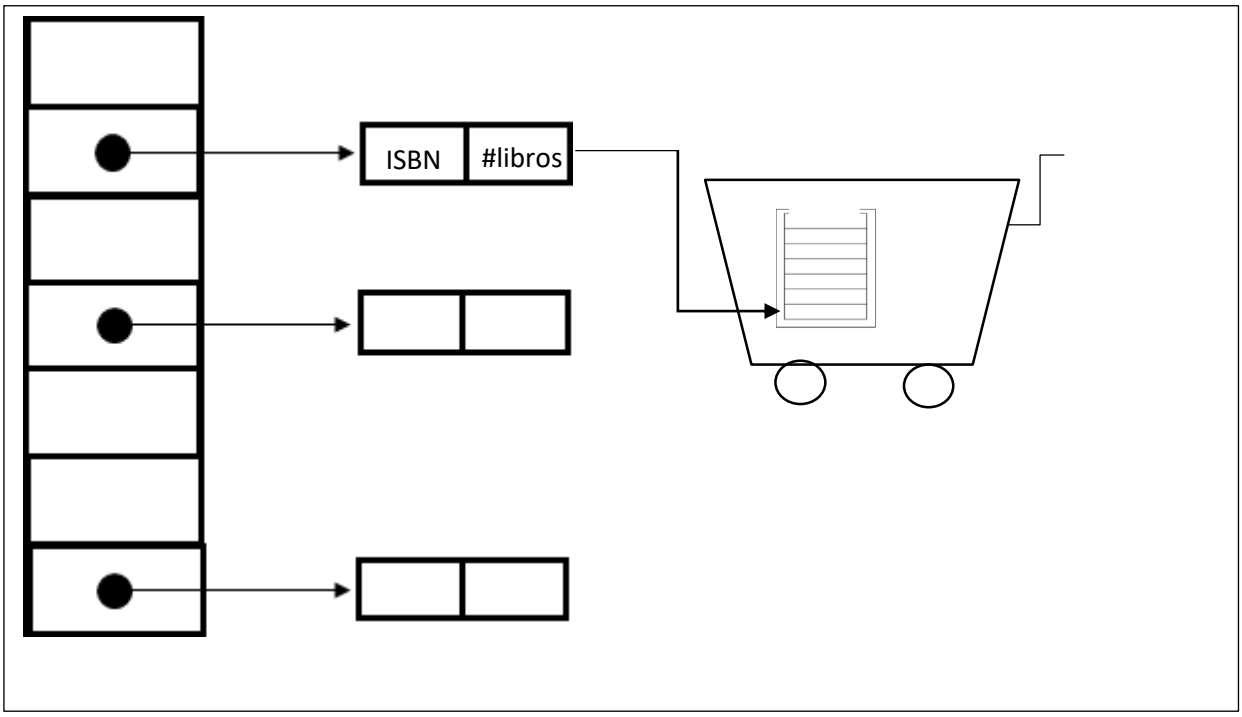




Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

Alternativa 5:

Esta alternativa consiste en almacenar todos los libros de la biblioteca en una única tabla hash, en dicha tabla la llave de cada slot será el ISBN de los libros y la cantidad de libros en el slot corresponderá al valor de dicha llave. Para el almacenamiento de los libros escogidos por el cliente será utilizada una pila.



Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

Fase 4: Transición de la formación de ideas a los diseños preliminares

- **Descarte de ideas no factibles**

Se descartaron las siguientes alternativas de la búsqueda de soluciones creativas debido a:

Alternativa 2	Esta alternativa se descarta debido a que el almacenamiento de libros en cada estante no corresponde a la dinámica de una tabla hash, ya que por la naturaleza del problema todos los libros contenidos en el hash son los mismos y solo releva su cantidad.
Alternativa 3	Esta alternativa es descartada porque al tomar cada estantería como una pila, no se podría acceder de manera independiente a cada slot de la pila ya que es necesario seguir la idea de primero en entrar – último en salir (FILO), por lo cual, si se desea acceder al estante ubicado en la base de la pila para tomar un libro, es necesario eliminar cada slot desde el top de la pila de manera descendente hasta llegar al slot que requiero. En ultimas esto es ineficiente e incorrecto.

Fase 5: Evaluación y selección de la mejor solución.

Actualmente se tienen 2 alternativas de solución las cuales resuelven el problema del prototipo que simula la forma en que se hace el proceso de la librería. Por lo tanto, a continuación, se definen una serie de criterios que ayudarán a escoger la mejor alternativa siguiendo como base el número de puntos que acumule cada una.

Criterios:

- **Criterio A: Complejidad temporal**
  Este criterio se basa en que tan eficiente en tiempo es la alternativa para solucionar el problema
  - Constante: 6 Puntos
  - Logarítmica: 5 Puntos
  - Lineal: 4 Puntos
  - Polinomial: 3 Puntos
  - Exponencial: 2 Puntos
  - Factorial: 1 Punto

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

- **Criterio B: Complejidad espacial**
  Este criterio se basa en que tan eficiente en espacio de memoria es la alternativa para solucionar el problema
  - Constante: 6 Puntos
  - Logarítmica: 5 Puntos
  - Lineal: 4 Puntos
  - Polinomial: 3 Puntos
  - Exponencial: 2 Puntos
  - Factorial: 1 Punto

- **Criterio C: Orden correcto de los libros**
  Este criterio se basa en el orden correcto en el que se muestran los libros una vez son empacados
  - Ordenados: 3
  - Medio ordenados: 2
  - Desordenados: 1

- **Criterio D: Valor correcto de la compra**
  Este criterio se basa en dar el valor correcto de la compra del cliente según el precio de los libros correspondientes que escogió.
  - Exacto: 3
  - Aproximado:2
  - Incorrecto: 1

- **Criterio E: Representatividad al método de compra**
  Este criterio se basa en que tan coherente o parecida es la alternativa al proceso usado por la biblioteca para la venta de libros.
  - Exacta: 3
  - Parecida: 2
  - Incoherente: 1

- **Criterio F: Informe completo de libros**
  Este criterio se basa en mostrar el código ISBN de todos libros escogidos por cada cliente.
  - Completo: 2
  - Incompleto: 1

- **Criterio G: Conveniencia para la implementación**
  Este criterio se basa en que tan conveniente resulta la alternativa para la implementación en java.
  - Altamente conveniente: 3
  - Medianamente conveniente: 2
  - No conveniente: 1

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

Evaluación:

Criterio / Alternativa	A	B	C	D	E	F	G	Total
1	4	4	3	3	3	2	2	22
4	6	4	3	3	1	2	2	21
5	6	4	3	3	1	1	1	19

Con base en los resultados obtenidos las alternativas 4 y 5 van a ser descarta y por lo tanto la alternativa 1 será la utilizada para darle solución al problema.

Paso 6: Preparación de informe y especificaciones.

- **Diseño de pruebas unitarias**

  - **Estructura de Datos: Stack**

Objetivo: Probar el correcto funcionamiento del método push(T objeto) para diferentes casos de prueba				
Clase: Stack		**Método:** push(T object)		
Caso #	**Descripción**	**Escenario**	**Valores de entrada**	**Resultado**
1 (Caso Estándar)	Se crea una pila de Enteros, la cual contiene los enteros {1, 2, 3, 4, 5}	stageOne()	Se entrega un entero, el cual su valor es 6	El algoritmo peek() debe de devolver el valor 6, demostrando que el entero se ingresó correctamente
2 (Caso Interesante)	Se crea una pila de String, la cual contiene las siguientes cadenas {“Hola”, “Soy”}	stageTwo()	Se entrega un String, cuya cadena es “Juan”	El algoritmo peek() debe de devolver la cadena “Juan”, demostrando que la cadena se ingresó correctamente
3 (Caso Limite)	Se crea una pila de Libros {l1.precio = 12000, l2.precio = 80000,	stageThree()	Se entrega un libro (l4) cuyo valor es de 100000	El algoritmo peek() debe de entregar un libro con precio de 100000 y el tamaño de la

	l3.precio = 50000}			pila debe de ser igual a 4

Objetivo: Probar el correcto funcionamiento del método pop() para diferentes casos de prueba				
Clase: Stack		**Método:** pop()		
Caso #	**Descripción**	**Escenario**	**Valores de entrada**	**Resultado**
1 (Caso Estándar)	Se crea una pila de Enteros, la cual contiene los enteros {1, 2, 3, 4, 5}	stageOne ()	Ninguna	El algoritmo pop() debe de devolver el valor de 5 y el algoritmo peek() debe de devolver el valor de 4
2 (Caso Interesante)	Se crea una pila de String, la cual contiene las siguientes cadenas {“Hola”, “Soy”}	stageTwo ()	Ninguna	El algoritmo pop() debe de devolver la cadena “Soy” y el algoritmo peek() debe de devolver la cadena “Hola”
3 (Caso Limite)	Se crea una pila de Libros {l1.precio = 12000, l2.precio = 80000, l3.precio = 50000}	stageThree()	Ninguna	El algoritmo pop debe de entregar a el libro l3 con precio de 50000 y el tamaño de la pila debe de disminuir a 2

Objetivo: Probar el correcto funcionamiento del método peek() para diferentes casos de prueba				
Clase: Stack		**Método:** peek()		
Caso #	**Descripción**	**Escenario**	**Valores de entrada**	**Resultado**
1 (Caso Estándar)	Se crea una pila de Enteros, la cual contiene los	stageOne ()	Ninguna	El algoritmo peek() debe de devolver el valor de 5

	enteros {1, 2, 3, 4, 5}			
2 (Caso Interesante)	Se crea una pila de String, la cual contiene las siguientes cadenas {“Hola”, “Soy”}	stageTwo ()	Ninguna	El algoritmo peek() debe de devolver la cadena “Soy”
3 (Caso Limite)	Se crea una pila de Libros {l1.precio = 12000, l2.precio = 80000, l3.precio = 50000}	stageThree()	Ninguna	El algoritmo peek() debe de retornar al libro l3 con precio de 50000 y el tamaño de la pila debe de ser 3

- **Estructura de Datos: Queue**

Objetivo: Probar el correcto funcionamiento del método enqueue() para diferentes casos de prueba				
Clase: Queue		**Método:** enqueue()		
Caso #	**Descripción**	**Escenario**	**Valores de entrada**	**Resultado**
1 (Caso Estándar)	Se crea una cola de Enteros, la cual contiene los enteros {1, 2, 3, 4, 5}	stageOne ()	Se entrega un entero, el cual su valor es 6	El algoritmo back() debe de devolver el valor de 6
2 (Caso Interesante)	Se crea una cola de String, la cual contiene las siguientes cadenas {“Hola”, “Soy”}	stageTwo ()	Se entrega un String cuya cadena es “Juan”	El algoritmo back() debe de devolver la cadena “Juan”
3 (Caso Limite)	Se crea una cola de Libros {l1.precio = 12000, l2.precio = 80000, l3.precio = 50000}	stageThree()	Se entrega un libro (l4) cuyo valor es de 100000	El algoritmo back() debe de devolver al libro l4 con precio de 100000 y el tamaño de la

				cola debe de ser 4

Objetivo: Probar el correcto funcionamiento del método dequeue() para diferentes casos de prueba				
Clase: Queue		**Método:** dequeue()		
Caso #	**Descripción**	**Escenario**	**Valores de entrada**	**Resultado**
1 (Caso Estándar)	Se crea una cola de Enteros, la cual contiene los enteros {1, 2, 3, 4, 5}	stageOne ()	Ninguna	El algoritmo dequeue() debe de devolver el valor de 1 y el algoritmo front() el valor de 2
2 (Caso Interesante)	Se crea una cola de String, la cual contiene las siguientes cadenas {“Hola”, “Soy”}	stageTwo ()	Ninguna	El algoritmo dequeue() debe de devolver la cadena “Hola” y el algoritmo front() debe devolver la cadena “Soy”
3 (Caso Limite)	Se crea una cola de Libros {l1.precio = 12000, l2.precio = 80000, l3.precio = 50000}	stageThree()	Ninguna	El algoritmo dequeue() debe de devolver l1 con precio de 12000 y el tamaño de la cola debe de ser 2

Objetivo: Probar el correcto funcionamiento del método front() para diferentes casos de prueba				
Clase: Queue		**Método:** front()		
Caso #	**Descripción**	**Escenario**	**Valores de entrada**	**Resultado**
1 (Caso Estándar)	Se crea una cola de Enteros, la cual contiene los	stageOne ()	Ninguna	El algoritmo front() debe de devolver el valor de 1

	enteros {1, 2, 3, 4, 5}			
2 (Caso Interesante)	Se crea una cola de String, la cual contiene las siguientes cadenas {“Hola”, “Soy”}	stageTwo ()	Ninguna	El algoritmo front() debe de devolver la cadena “Hola”
3 (Caso Limite)	Se crea una cola de Libros {l1.precio = 12000, l2.precio = 80000, l3.precio = 50000}	stageThree()	Ninguna	El algoritmo front() debe de devolver al libro l1() con precio de 12000 y el tamaño de la cola debe de ser igual a 3

- **Estructura de Datos: HashTable**

Objetivo: Probar el correcto funcionamiento del método insert(T, K) para diferentes casos de prueba				
Clase: HashTable		**Método:** insert(T, K)		
Caso #	**Descripción**	**Escenario**	**Valores de entrada**	**Resultado**
1 (Caso Estándar)	Se crea una HashTable de Enteros, la cual contiene los enteros {1, 2, 3, 4, 5} cuya llave es un String	stageOne ()	Se entrega un entero 6 cuya llave es el mismo número, pero convertido en String	El método search(K) debe de entregar al entero agregado buscado con la llave “6”
2 (Caso Interesante)	Se crea una HashTable de String, la cual contiene las siguientes cadenas {“Hola”, “Soy”} cuyas llaves son String	stageTwo ()	Se entrega una Cadena “Juan” cuya llave es la misma cadena	El método search(K) debe de retornar la cadena “Juan” al buscarlo con la llave “Juan”
3 (Caso Limite)	Se crea una Hash de libros	stageThree()	Se entrega un libro l4 cuyo	El algoritmo search(K) debe de devolver el

	que contiene los siguientes elementos {l1.precio = 12000, l2.precio = 80000, l3.precio = 50000} y cuya llave es el precio del libro		precio es de 100000	libro l4 al buscarlo con la llave 100000

Objetivo: Probar el correcto funcionamiento del método delete(K) para diferentes casos de prueba				
Clase: HashTable		**Método:** delete(K)		
Caso #	**Descripción**	**Escenario**	**Valores de entrada**	**Resultado**
1 (Caso Estándar)	Se crea una HashTable de Enteros, la cual contiene los enteros {1, 2, 3, 4, 5} cuya llave es un String	stageOne ()	Se entrega un String cuya cadena es “2”	El método delete(K) debe de retornar el valor entero de 2
2 (Caso Interesante)	Se crea una HashTable de String, la cual contiene las siguientes cadenas {“Hola”, “Soy”} cuyas llaves son String	stageTwo ()	Se entrega un String cuya cadena es “Soy”	El método delete(K) debe de retornar la cadena “Soy”
3 (Caso Limite)	Se crea una Hash de libros que contiene los siguientes elementos {l1.precio = 12000, l2.precio = 80000,	stageThree()	Se entrega un entero con el precio del libro que se quiere eliminar, la llave va a ser igual a 80000	El algoritmo delete(K) debe de entregar un libro con precio de 80000

	l3.precio = 50000} y cuya llave es el precio del libro			

Objetivo: Probar el correcto funcionamiento del método search(K) para diferentes casos de prueba				
Clase: HashTable		**Método:** search(K)		
Caso #	**Descripción**	**Escenario**	**Valores de entrada**	**Resultado**
1 (Caso Estándar)	Se crea una HashTable de Enteros, la cual contiene los enteros {1, 2, 3, 4, 5} cuya llave es un String	stageOne ()	Se entrega un String cuya cadena es “4”	El método search(K) debe de retornar el valor entero de 4
2 (Caso Interesante)	Se crea una HashTable de String, la cual contiene las siguientes cadenas {“Hola”, “Soy”} cuyas llaves son String	stageTwo ()	Se entrega un String cuya cadena es “Hola”	El método search(K) debe de retornar la cadena “Hola”
3 (Caso Limite)	Se crea una Hash de libros que contiene los siguientes elementos {l1.precio = 12000, l2.precio = 80000, l3.precio = 50000} y cuya llave es el precio del libro	stageThree()	Se entrega un entero con el precio del libro que se quiere eliminar, la llave va a ser igual a 12000	El algoritmo search(K) debe de entregar un libro con precio de 12000

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

- **Estructura de Datos: PriorityQueue (Min Heap)**

Objetivo: Probar el correcto funcionamiento del método insert(T) para diferentes casos de prueba				
Clase: PriorityQueue		**Método:** insert(T)		
Caso #	**Descripción**	**Escenario**	**Valores de entrada**	**Resultado**
1 (Caso Estándar)	Se crea una PriorityQueue de Enteros, la cual contiene los enteros {1, 2, 3, 4, 5}	stageOne ()	Se entrega un entero cuyo valor es 0 al método insert(T)	El método insert(T) debe poner al entero agregado en la cabeza y el método mínimum() debe en entregar el valor de 0
2 (Caso Interesante)	Se crea una PriorityQueue de String, la cual contiene las siguientes cadenas {“Dado”, “Elefante”, “Yuca”}	stageTwo ()	Se entrega una Cadena “Ada” al método insert(T)	El método insert(T) debe poner al String agregado en la cabeza y el método mínimum() debe en entregar al String “Ada”
3 (Caso Limite)	Se crea una PriorityQueue de Cliente, la cual contiene los siguientes Clientes {(cl1 = “Juan” , 2) , (cl2 = “Ana”, 5)}	stageThree()	Se entrega un cliente cl3 = “Carlos” , 1	El método insert(T) debe poner al cliente agregado en la cabeza y el método mínimum() debe en entregar al cliente Carlos

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

Objetivo: Probar el correcto funcionamiento del método extractMin() para diferentes casos de prueba				
Clase: PriorityQueue		**Método:** extractMin()		
Caso #	**Descripción**	**Escenario**	**Valores de entrada**	**Resultado**
1 (Caso Estándar)	Se crea una PriorityQueue de Enteros, la cual contiene los enteros {1, 2, 3, 4, 5}	stageOne ()	Ninguna	El método extractMin() debe de entregar al entero 1
2 (Caso Interesante)	Se crea una PriorityQueue de String, la cual contiene las siguientes cadenas {“Dado”, “Elefante”, “Yuca”}	stageTwo ()	Ninguna	El método extractMin() debe de entregar la cadena “Dado”
3 (Caso Limite)	Se crea una PriorityQueue de Cliente, la cual contiene los siguientes Clientes {(cl1 = “Juan” , 2) , (cl2 = “Ana”, 5)}	stageThree()	Ninguna	El método extractMin() debe de entregar al cliente cl1 con id

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

Objetivo: Probar el correcto funcionamiento del método minimum() para diferentes casos de prueba				
Clase: PriorityQueue		**Método:** minimum()		
Caso #	**Descripción**	**Escenario**	**Valores de entrada**	**Resultado**
1 (Caso Estándar)	Se crea una PriorityQueue de Enteros, la cual contiene los enteros {1, 2, 3, 4, 5}	stageOne ()	Ninguna	El método minimum() debe de entregar al entero 1
2 (Caso Interesante)	Se crea una PriorityQueue de String, la cual contiene las siguientes cadenas {“Dado”, “Elefante”, “Yuca”}	stageTwo ()	Ninguna	El método minimum() debe de entregar la cadena “Dado”
3 (Caso Limite)	Se crea una PriorityQueue de Cliente, la cual contiene los siguientes Clientes {(cl1 = “Juan” , 2) , (cl2 = “Ana”, 5)}	stageThree()	Ninguna	El método minimum(() debe de entregar al cl1 con su id

Diseños preliminares

Para los diseños preliminares hemos decidido hacer el algoritmo más importante de cada estructura de datos, esto nos servirá para tener una idea de cómo implementar estos algoritmos y aprovecharemos para sacar la complejidad espacial y temporal de cada algoritmo.

Método 1:

#	**push(T object)**	**C.E**	**C.T**
1	`Node<T, K> theNodo = new Node<T, K>(object)`	1	1
2	`if(root == null)`	0	1
3	`root = theNodo`	0	0
4	`last = theNodo`	0	0
5	`else`	0	0

6	`Node<T, K> aux = root`	1	1
7	`While(aux.getNext() != null)`	0	n
8	`aux = aux.getNext()`	0	n-1
9	`aux.setNext(theNodo)`	0	n-1
10	`last = theNodo`	0	n-1
11	`size++`	0	1

Método 2:

#	dequeue()	C.E	C.T
1	`if(root == null)`	0	1
2	`return null`	0	0
3	`else`	0	0
4	`Node<T, K> deleted = root`	1	1
5	`root = deleted.getNext()`	0	1
6	`size--`	0	1
7	`return deleted.getDate()`	0	1

Método 3:

#	search(K key)	C.E	C.T
1	`boolean found = false`	1	1
2	`T temporary = null`	1	1
3	`for(i = 0; i < arrayHash.length && !found; i++)`	0	n
4	`position = funcionHash(key, i)`	1	n-1
5	`if(arrayHash[position] != null)`	0	n-1
6	`if(arrayHash[position].getKey().equals(key))`	0	n-1
7	`temporary = (T) arrayHash[position].getDate()`	0	n-1
9	`found = true`	0	n-1
10	`return temporary`	0	1

Importante: Para sacar la complejidad del algoritmo search(K key) de la HashTable asumimos que la función Hash es muy mala y que estamos resolviendo nuestras colisiones por sondeo lineal.

Método 4:

#	extractMin()	C.E	C.T
1	`if (heap.getSize() <= 0)`	0	1
2	`return null`	0	0
3	`else`	0	0
4	`T minVal = heap.get(1)`	1	1
5	`heap.set(1, heap.get(heap.getSize()))`	0	1
6	`heap.remove(1)`	0	1
7	`return minVal`	0	1

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

Análisis de complejidad espacial

- ✓ **Método 1:**
  T(n) = 2
  T(n) = C
  Por medio del T(n) podemos concluir que la notación asintótica para la complejidad espacial del método 1 es: $O(1)$.

- ✓ **Método 2:**
  T(n) = 1
  T(n) = C
  Por medio del T(n) podemos concluir que la notación asintótica para la complejidad espacial del método 2 es: $O(1)$.

- ✓ **Método 3:**
  T(n) = 3
  T(n) = C
  Por medio del T(n) podemos concluir que la notación asintótica para la complejidad espacial del método 3 es: $O(1)$.

- ✓ **Método 4:**
  T(n) = 1
  T(n) = C
  Por medio del T(n) podemos concluir que la notación asintótica para la complejidad espacial del método 4 es: $O(1)$.

Análisis de complejidad temporal

- ✓ **Método 1:**
  T(n) = $4n + 1$
  T(n) = $An + B$
  Por medio del T(n) podemos concluir que la notación asintótica para la complejidad temporal del método 1 es: $O(n)$.

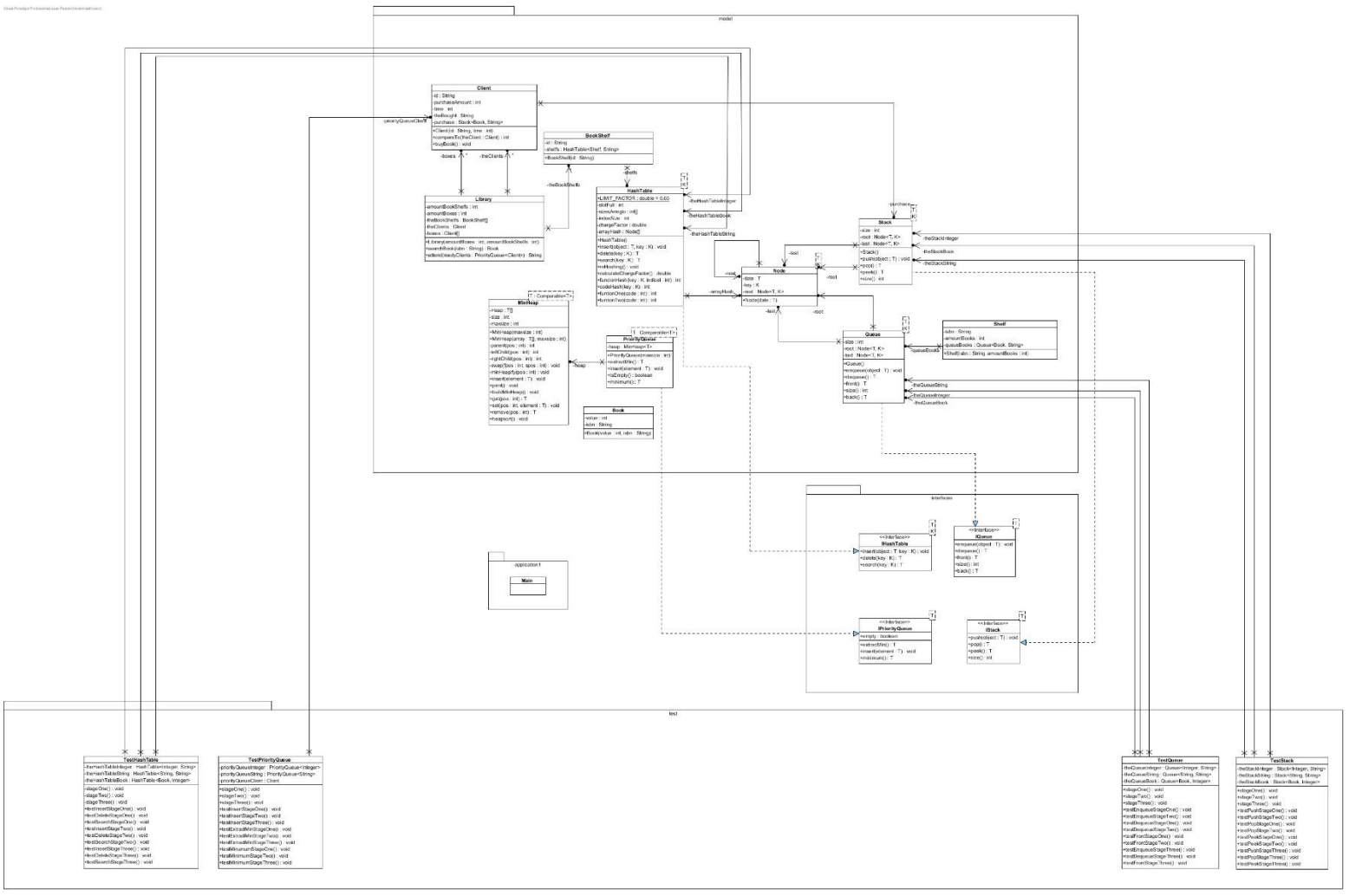
- ✓ **Método 2:**
  T(n) = 5
  T(n) = A
  Por medio del T(n) podemos concluir que la notación asintótica para la complejidad temporal del método 2 es: $O(1)$.

- ✓ **Método 3:**
  T(n) = $6n - 2$
  T(n) = $An - B$
  Por medio del T(n) podemos concluir que la notación asintótica para la complejidad temporal del método 3 es: $O(n)$.

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

- ✓ **Método 4:**
  T(n) = 5
  T(n) = C
  Por medio del T(n) podemos concluir que la notación asintótica para la complejidad espacial del método 4 es: $O(1)$.

Diagrama de clases



Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

Fase 7: Implementación

La implementación de la solución se encuentra en el siguiente repositorio de githud:

https://github.com/Juan-Puerta/Lab2-AED