Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

Método de la ingeniería

Fase 1: Identificación del problema.

Contexto problemático:

En economía, un mercado financiero es un espacio (físico o virtual o ambos) en el que se realizan los intercambios de instrumentos financieros y se definen sus precios. En general, cualquier mercado de materias primas podría ser considerado como un mercado financiero si el propósito del comprador no es el consumo inmediato del producto, sino el retraso del consumo en el tiempo. Los mercados financieros están afectados por las fuerzas de oferta y demanda. Los mercados colocan a todos los vendedores en el mismo lugar, haciendo así más fácil encontrar posibles compradores. A la economía que confía ante todo en la interacción entre compradores y vendedores para destinar los recursos se le llama economía de mercado, en contraste con la economía planificada.

- **Problema**

Actualmente las bosa de valores de Colombia (BVC) no permite tranzar con acciones internacionales ni trabajar con el mercado de divisas o derivados. Debido a esto la BVC quiere aprovechar estas coyunturas y quiere consolidar una aplicación, que permita manejar los datos de algunos mercados de divisas y de acciones internacionales.
La empresa desea que la aplicación tenga los siguientes requerimientos funcionales y no funcionales.

- **Requerimientos Funcionales**

Requerimiento funcional 1: El programa debe de permitir consultar el precio más alto de una acción o un mercado de divisas en un rango de tiempo.
Entradas: Rango de tiempo
Salidas: Precio más alto de la acción o de la divisa

Requerimiento funcional 2: El programa debe de permitir consultar el precio más bajo de una acción o un mercado de divisas en un rango de tiempo.
Entradas: Rango de tiempo
Salidas: Precio más bajo de la acción o de la divisa

Requerimiento funcional 3: El programa debe de permitir consultar el periodo de tiempo donde una acción / mercado de divisas tuvo su mayor crecimiento.
Entradas: Acción / Mercado de divisas
Salidas: Periodo de tiempo con mayor crecimiento

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

Requerimiento funcional 4: El programa debe de mostrar cuáles acciones / Mercado de divisas superan un valor en un rango de tiempo.
Entradas: Rango de tiempo, Valor a superar
Salidas: Acciones o Mercado de divisas que superan ese valor en un determinado periodo de tiempo

Requerimiento funcional 5: El programa debe de permitir consultar cuales son las 3 acciones / Mercados que presentaron mayor crecimiento en un rango de tiempo.
Entradas: Rango de tiempo
Salidas: Acciones o Mercado de divisas con mayor crecimiento en un rango de tiempo

Requerimiento funcional 6: El programa debe de permitir ingresar datos por medio de archivos de texto
Entradas: Archivo de texto
Salidas: Datos Ingresados

Requerimiento funcional 7: El programa debe de permitir ingresar datos de manera individual
Entradas: Acción / Divisas / Bitcoin
Salidas: Dato Ingresado

- **Requerimientos no funcionales**

Requerimiento no funcional #1: Para el mercado de divisas su complejidad de las operaciones básicas debe ser O(Log n)

Requerimiento no funcional #2: Para el mercado de acciones su complejidad de las operaciones básicas debe ser O(Log n)

Requerimiento no funcional #3: Para el mercado de BitCoins su complejidad de las operaciones básicas debe ser O(n

Requerimiento no funcional #4: Mostrar una gráfica del estado de los precios de una acción con un color

Requerimiento no funcional #5: Mostrar una gráfica del estado de los precios de un mercado de divisas con un color

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

Fase 2: Recopilación de la información.

Referencias

Arias, A. S. (2019). Economipedia. Obtenido de Economipedia: https://economipedia.com/definiciones/ley-de-oferta-y-demanda.html

Arias, J. S. (11 de 04 de 2019). Economipedia. Obtenido de https://economipedia.com/definiciones/mercado-de-divisas-forex.html

Criptonoticias. (11 de 04 de 2019). Criptonoticias. Obtenido de Criptonoticias: https://www.criptonoticias.com/informacion/que-es-bitcoin/

cupintranet. (12 de 04 de 2019). cupintranet. Obtenido de cupintranet: https://cupintranet.virtual.uniandes.edu.co/sitio/index.php/cursos/estructuras-de-datos/cupi2collections/estructuras-de-datos/arbol-rojo-negro

fxstreet. (12 de 04 de 2019). fxstreet. Obtenido de fxstreet: https://www.fxstreet.es/rates-charts/gbpcad

Gurin, S. (2004). tdlp. Obtenido de tdlp: http://es.tldp.org/Tutoriales/doc-programacion-arboles-avl/avl-trees.pdf

iForex. (11 de 04 de 2019). iForex. Obtenido de iForex: https://www.iforex.mx/%C3%ADndices/dow-jones

Limited, I. G. (04 de 04 de 2019). Ig. Obtenido de Ig: https://www.ig.com/es/invertir-en-criptomonedas/que-son-las-criptomonedas#information-banner-dismiss

Miguel A, G. (17 de 09 de 2012). applesfera. Obtenido de applesfera: https://www.applesfera.com/respuestas/alguien-que-me-explique-por-favor-por-que-aapl-es-apple

Perez, A. B. (24 de 10 de 2018). Enciclopedia financiera. Obtenido de https://www.enciclopediafinanciera.com/mercados-financieros/mercados-de-acciones.htm

tecnicasdetrading. (12 de 04 de 2019). tecnicasdetrading. Obtenido de tecnicasdetrading: https://www.tecnicasdetrading.com/2015/12/par-de-divisas-usdjpy.html

tecnincasdetrading. (12 de 04 de 2019). tecnincasdetrading. Obtenido de tecnincasdetrading: https://www.tecnicasdetrading.com/2015/11/par-de-divisas-eurusd.html

Tradingview. (12 de 04 de 2019). Tradingview. Obtenido de Tradingview: https://es.tradingview.com/symbols/XAUUSD/

Wikipedia. (03 de 04 de 2019). Wikipedia. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/S%26P_500

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

Mercado de acciones:
El Capital Social de una empresa representa el capital originalmente pagado o invertido en el negocio por sus fundadores. Sirve como garantía para los acreedores de una empresa, ya que no pueden ser retirados en detrimento de los acreedores.
El capital social de una empresa se divide en acciones, el total de los que deben declararse en el momento de la creación de empresas. Según la cantidad total de dinero invertido en el negocio, cada acción tiene un cierto valor nominal, comúnmente conocido como el valor nominal de la acción. Las acciones representan una fracción de la propiedad en una empresa. (Perez, 2018)

Mercado de divisas:
El mercado de divisas o mercado cambiario es un mercado que se caracteriza por el libre cambio de divisas, es decir, su objetivo principal es el de facilitar el comercio internacional y la inversión. También se conoce como FOREX (Foreign Exchange, que se traduce como intercambio de monedas extranjeras).
En ese espacio físico o virtual se fija el precio de cada moneda denominado tipo de cambio. Dicha cotización depende exclusivamente de la oferta y demanda de los participantes.
Cabe precisar que en el mercado cambiario no se negocia solo efectivo. Por el contrario, también se comercializan depósitos registrados en instituciones financieras o documentos que otorguen el derecho a cobrar una cantidad de dinero. (Arias, 2019)

Criptomoneda:
Las criptomonedas son monedas virtuales. Pueden ser intercambiadas y operadas como cualquier otra divisa tradicional, pero están fuera del control de los gobiernos e instituciones financieras.
Existe un gran número de criptodivisas disponibles, todas con sus propias características y aplicaciones. Las que tienen mayor capitalización de mercado son -al menos por ahora- una minoría, que incluye bitcoin, bitcoin cash, ether, litecoin, ripple y dash. (Limited, 2019)

Bitcoin:
Lo primero que se debe saber es que Bitcoin es tanto una moneda como un sistema digital. Como moneda puede servir para todo lo que cualquier moneda sirve, pero en lugar de tener un ente gubernamental —como un banco central— que lo emita y lo respalde, se basa por completo en el sistema digital que fue ideado por su creador, Satoshi Nakamoto. Como consecuencia, una de las características más resaltantes de Bitcoin es que no le pertenece a ningún país o gobierno; y dado que su creador es anónimo y decidió que su invento fuera de licencia libre, tampoco le pertenece a ningún individuo o compañía privada. Quienes mantienen en funcionamiento su plataforma son los propios usuarios. (Criptonoticias, 2019)

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

La tasa de US30:
El US 30 es un instrumento financiero popular que se basa en el desempeño del índice Dow Jones Industrial Average Future. Puede cambiarlo en cualquier dirección, hacia arriba o hacia abajo, utilizando apalancamiento de hasta 200:1. iFOREX ofrece la oportunidad de invertir en el US 30, que se basa en el rendimiento del índice futuro del Dow Jones, en forma de CFD.
El Dow Jones Índice Industrial es uno de los índices bursátiles estadounidenses más famosos y seguidos de cerca. Fue fundado en 1896 por Charles Dow, editor del Wall Street Journal y co-fundador de US 30 & Company. Contiene 30 de las empresas más grandes cotizadas en bolsa con sede en los Estados Unidos de América. El índice está basado en un promedio ponderado de precios, y para compensar los efectos de divisiones de acciones y otros ajustes, actualmente es un promedio escalonado. (iForex, 2019)

US SPX 500:
El índice Standard & Poor's 500 (Standard & Poor's 500 Index) también conocido como S&P 500 es uno de los índices bursátiles más importantes de Estados Unidos. Al S&P 500 se lo considera el índice más representativo de la situación real del mercado.
El índice se basa en la capitalización bursátil de 500 grandes empresas que poseen acciones que cotizan en las bolsas NYSE o NASDAQ, el índice captura aproximadamente el 80% de toda la capitalización de mercado en Estados Unidos. Los componentes del índice S&P 500 y su ponderación son determinados por S&P Dow Jones Indices. Se diferencia de otros índices de mercados financieros de Estados Unidos, tales como el Dow Jones Industrial Average o el índice Nasdaq Composite, en la diversidad de los rubros que lo conforman y en su metodología de ponderación. (Wikipedia, 2019).

AAPL:
símbolo que se usa en la bolsa, en el NASDAQ, para Apple Computer, Inc. (Miguel A, 2012)

MSFT:
símbolo que se usa en la bolsa, en el NASDAQ, para Microsoft Computer, Inc.

WTI:
West Texas Intermediate (WTI) es una corriente de crudo producido en Texas y el sur de Oklahoma que sirve como referencia para fijar el precio de otras corrientes de crudo.

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

XAUUSD:
Es el código para el Oro en la tabla periódica de elementos, y el precio arriba indicado corresponde a la cotización del oro en dólares norteamericanos, que es el criterio común para medir su valor alrededor del mundo. (Tradingview, 2019)

ERUSUD:
Se define (EUR/USD) como la abreviatura para el par de divisas o cruce del euro y el dólar estadounidense, las monedas de la Zona Euro (EUR) y Estados Unidos (USD). Este par de divisas muestra cual es la cantidad de dólares estadounidenses (la divisa de cotización) que se requiere para comprar un euro (la divisa base).
El valor o cotización del par EUR/USD es mostrado en términos de 1 euro por una cantidad determinada de dólares estadounidenses, es decir el valor de 1 EUR en USD. En otras palabras, el valor del EUR/USD indica cual es la cotización del euro en términos del dólar estadounidense. Por ejemplo, si el par se negocia a 1,1500 esto significa que se necesita 1,1500 dólares estadounidenses para comprar 1 euro.
Negociar con el par de divisas EUR/USD también se conoce como operar con el "Euro". (tecnincasdetrading, 2019)

GBPCAD:
GBP/CAD es la abreviatura del par de la libra esterlina y el dólar canadiense. Muestra cuánto vale el GBP (moneda base) medido frente al CAD (moneda contraria). Por ejemplo, GBP/CAD = 1.6685 indica que una libra esterlina puede comprar 1.6685 dólares canadienses. (fxstreet, 2019).

USDJPY:
el par de divisas USD/JPY (dólar estadounidense/yen japonés) es uno de los más negociados en los mercados internacionales de divisas. Los inversores y especuladores interesados en operar con este par deben analizar con cuidado la economía japonesa y la evolución de las cotizaciones del USD/JPY con el fin de determinar si este es un instrumento financiero que se adapta a sus estrategias y necesidades de inversión. Como veremos a continuación, el USD/JPY tiene características que lo convierten en un mercado interesante para muchos traders, incluyendo especuladores, sin embargo en ocasiones puede resultar demasiado riesgoso para los traders que tienen una alta aversión al riesgo ya que presenta periodos de elevada volatilidad, aunque no tanto como otros pares relacionados con el yen como el EUR/JPY o el GBP/JPY. (tecnicasdetrading, 2019).

ARBOL ROJO-NEGRO:
Un árbol rojo negro es un árbol binario donde cada nodo tiene también un atributo de color, cuyo valor puede ser o rojo o negro. Las hojas de un árbol rojo negro son irrelevantes y no tienen datos.

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

Para que un árbol binario ordenado sea considerado como un árbol rojo negro debe cumplir con las siguientes propiedades:

La raíz del árbol es negra.
Los hijos de un nodo rojo son negros.
Las hojas del árbol son negras.
Todas las ramas del árbol (caminos desde la raíz hasta una hoja) tienen el mismo número de nodos negros.

Este tipo de estructuras sirven para el almacenamiento de elementos entre los que exista una relación de orden. La complejidad de la búsqueda en los árboles rojo negro es O(log2 n).
En el caso concreto de implementación en Cupi2Collections esta estructura fue desarrollada para soportar objetos de cualquier tipo, siempre que la clase a la que pertenecen implemente la interface Comparable necesaria para la inserción de los elementos de forma ordenada. (cupintranet, 2019)

Árbol AVL:
Un árbol AVL es un árbol binario de búsqueda que cumple con la condición de que la diferencia entre las alturas de los subárboles de cada uno de sus nodos es, como mucho 1.
La denominación de árbol AVL viene dada por los creadores de tal estructura (Adelson-Velskii y Landis).
Recordamos que un árbol binario de búsqueda es un árbol binario en el cual cada nodo cumple con que todos los nodos de su subárbol izquierdo son menores que la raíz y todos los nodos del subárbol derecho son mayores que la raíz.
Recordamos también que el tiempo de las operaciones sobre un árbol binario de búsqueda son O(log n) promedio, pero el peor caso es O(n), donde n es el número de elementos.
La propiedad de equilibrio que debe cumplir un árbol para ser AVL asegura que la profundidad del árbol sea O(log(n)), por lo que las operaciones sobre estas estructuras no deberán recorrer mucho para hallar el elemento deseado. Como se verá, el tiempo de ejecución de las operaciónes sobre estos árboles es, a lo sumo O(log(n)) en el peor caso, donde n es la cantidad de elementos del árbol.
Sin embargo, y como era de esperarse, esta misma propiedad de equilibrio de los árboles AVL implica una dificultad a la hora de insertar o eliminar elementos: estas operaciones pueden no conservar dicha propiedad. (Gurin, 2004)

Ley de oferta y demanda:
La ley de la oferta y la demanda es el principio básico sobre el que se basa una economía de mercado. Este principio refleja la relación que existe entre la demanda de un producto y la cantidad ofrecida de ese producto teniendo en cuenta el precio al que se vende el producto.
Así, según el precio que haya en el mercado de un bien, los oferentes están dispuestos a fabricar un número determinado de ese bien. Al igual que los

demandantes están dispuestos a comprar un número determinado de ese bien, dependiendo del precio. El punto donde existe un equilibrio porque los demandantes están dispuestos a comprar las mismas unidades que los oferentes quieren fabricar, por el mismo precio, se llama equilibrio de mercado o punto de equilibrio.

Según esta teoría, la ley de la demanda establece que, manteniéndose todo lo demás constante (ceteris paribus), la cantidad demandada de un bien disminuye cuando el precio de ese bien aumenta. Por el otro lado, la ley de la oferta indica que, manteniéndose todo lo demás constante (ceteris paribus), la cantidad ofrecida de un bien aumenta cuando lo hace su precio.

Así, la curva de la oferta y la curva de la demanda muestran como varía la cantidad ofrecida o demandada, respectivamente, según varía el precio de ese bien. (Arias A. S., 2019)-

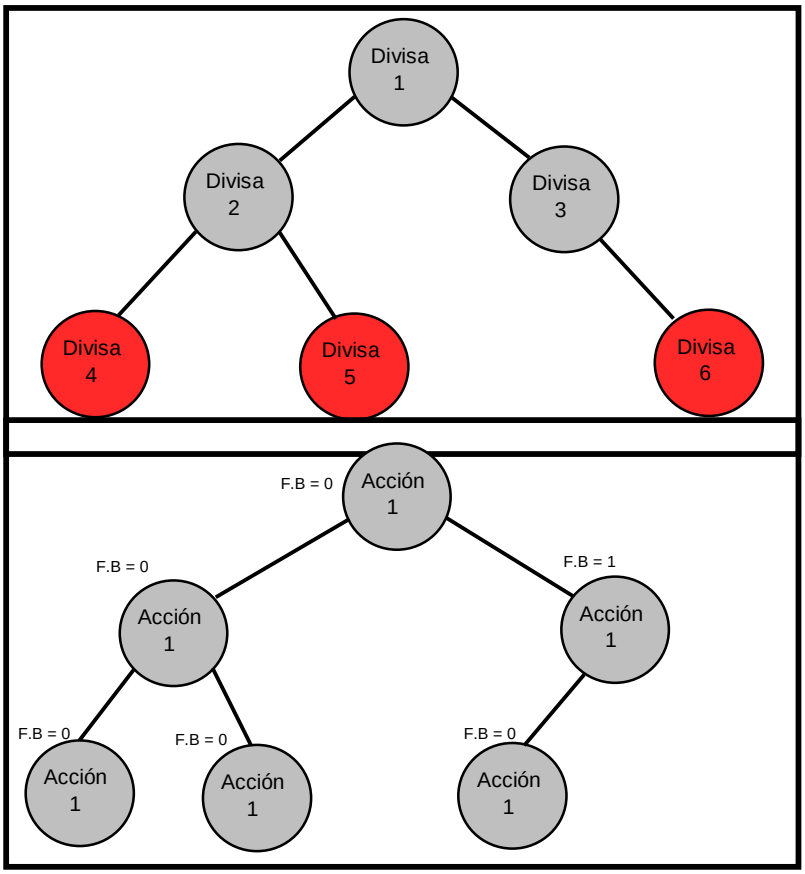
Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

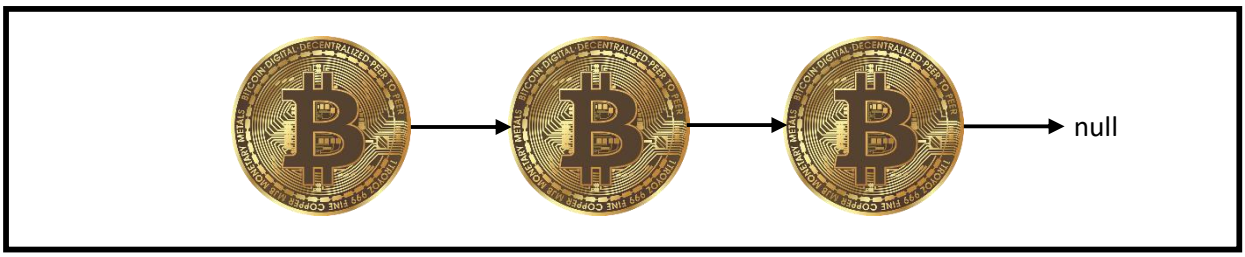
Fase 3: Búsqueda de soluciones creativas.

Para la solución de este problema necesitamos enfocarnos en las estructuras de datos que se adecuan más para almacenar las acciones, divisas y bitcoins, esto nos dará que el programa sea eficiente y preciso a la hora de dar la solución.

- **Alternativa 1**
  Esta alternativa se basa en almacenar las acciones en un árbol AVL, las divisas en un árbol Rojo-Negro y por último los bitcoins en una lista enlazada.

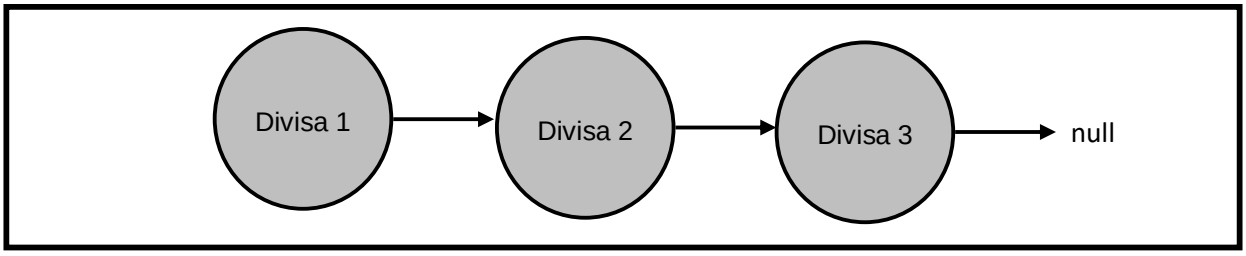
A

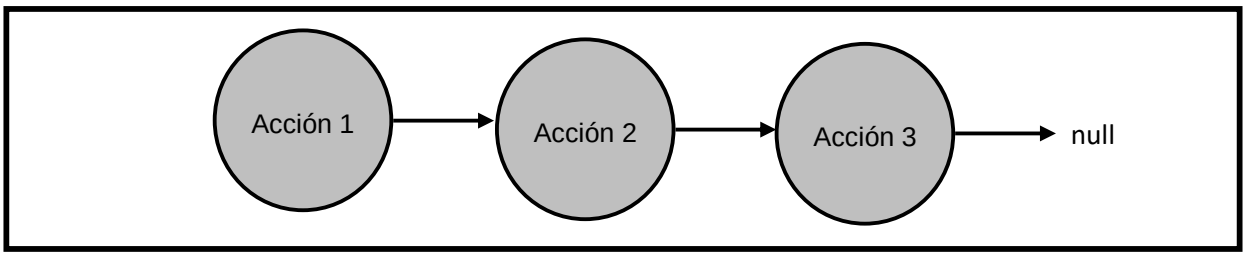


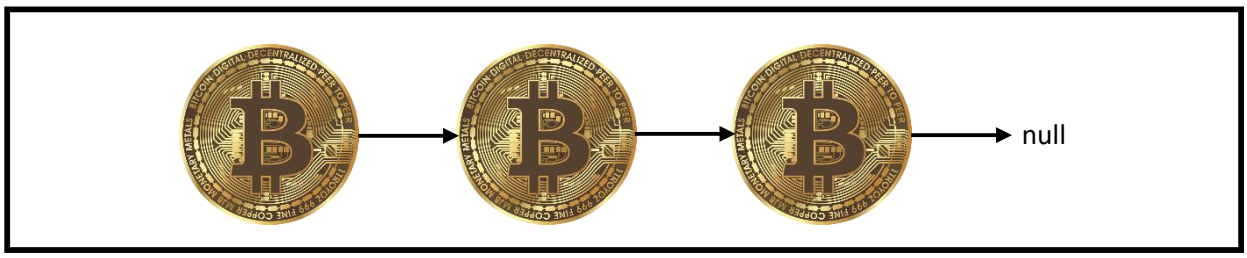


Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

- **Alternativa 2**
  En esta alternativa utilizaremos listas enlazadas para conectar las acciones, divisas y bitcoins.





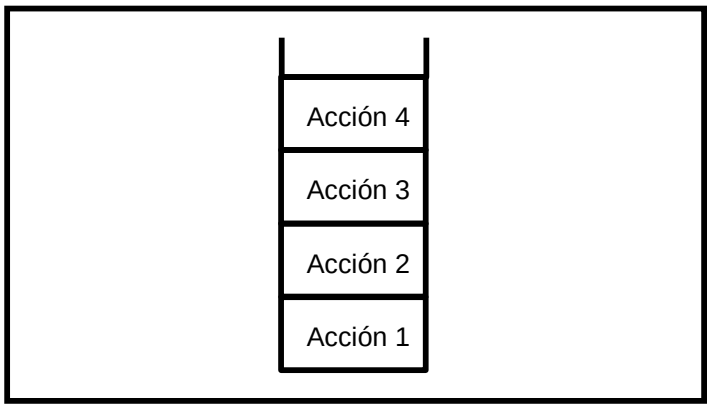


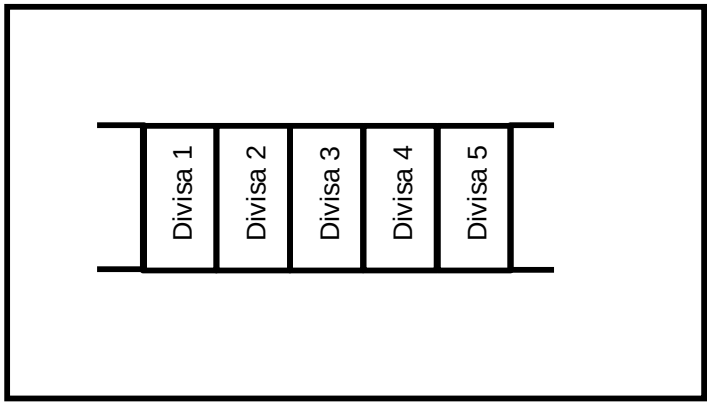
Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

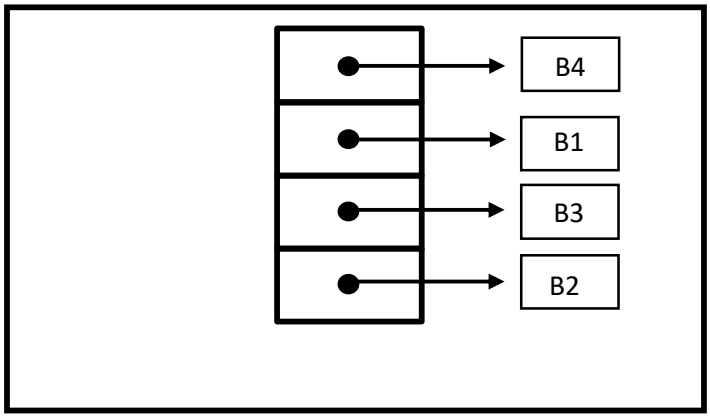
- **Alternativa 3**

  En esta alternativa se optará por almacenar las acciones en una pila, las divisas en una cola y los bitcoins en TablaHash.

  .



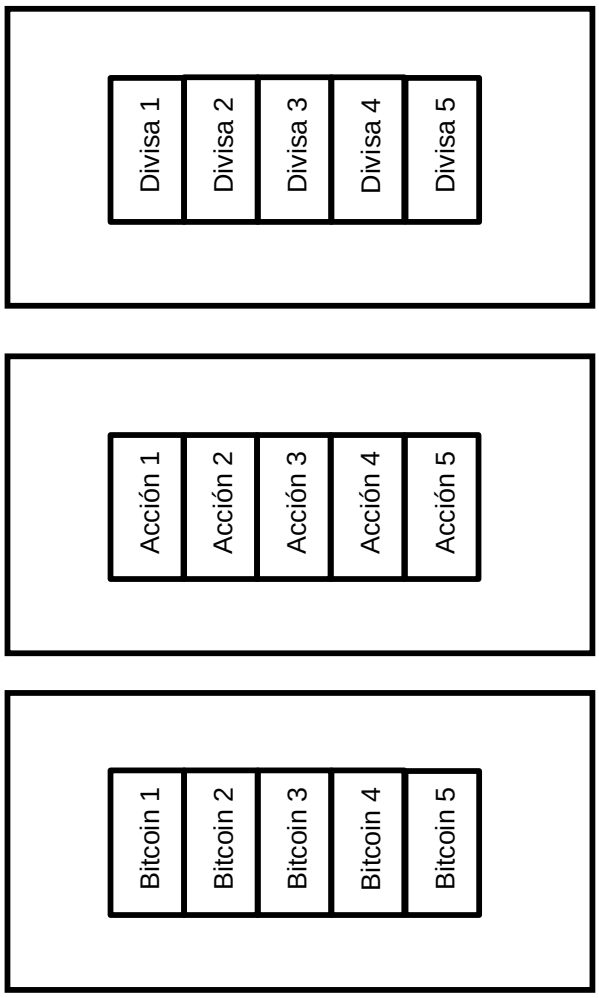




Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

- **Alternativa 4**

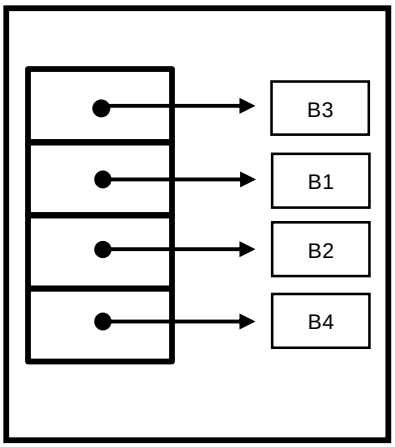
  Esta alternativa se basa en poner las acciones, divisas y bitcoins en arreglos para así poder tener acceso a ellos de manera lineal.

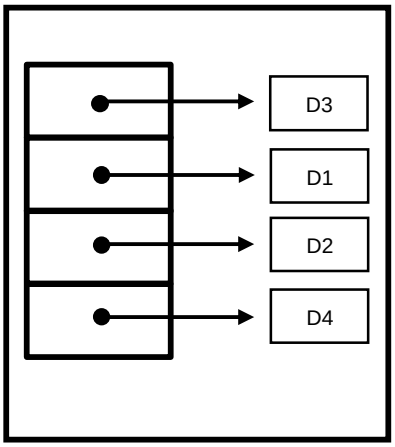


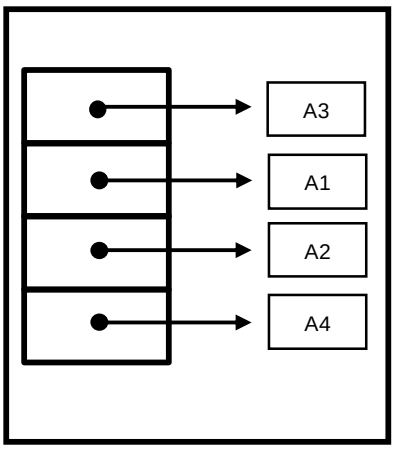
Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

- **Alternativa 5**

  Esta alternativa se basa en poner las acciones, divisas y bitcoins en tres tablas hash y así poder acceder cualquier acción, divisa o bitcoin en tiempo constante.



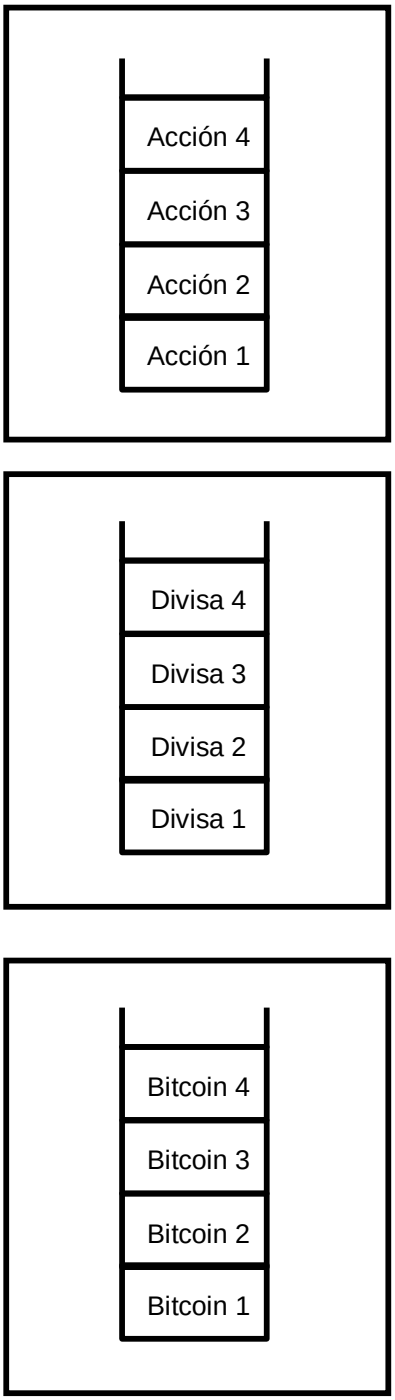




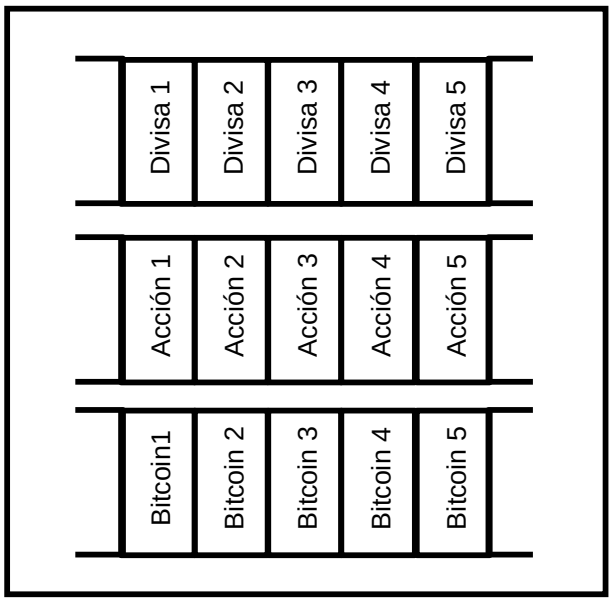
Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

- **Alternativa 6**

  En esta alternativa pondremos todas las acciones, divisas y bitcoins en pilas así podremos obtener los últimos elementos que se agregaron en tiempo constante.



- **Alternativa 7**
  En esta alternativa pondremos todas las acciones, divisas y bitcoins en colas así podremos obtener los primeros y últimos elementos que se agregaron en tiempo constante.



Fase 4: Transición de la formulación de ideas a los diseños preliminares.

- **Descarte de ideas no factibles**

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

Alternativa 3	Se descartó esta alternativa debido a que almacenar las acciones en una pila sólo nos permitiría obtener la última acción y para obtener las otras tendríamos que eliminar todas hasta que obtengamos la que queremos. Almacenar las divisas en una cola no tendría sentido porque sólo podemos conocer la primera y la última divisa y para conocer las demás tendríamos que eliminar divisas. Almacenar los bitcoins en una TablaHash se podría hacer, pero no va con el contexto del programa que el cliente pidió.
Alternativa 4	Esta alternativa se descarta porque almacenar todas las divisas, acciones y bitcoins en un arreglo es una solución que para la gran cantidad de datos que vamos a almacenar resultaría muy ineficiente.

Alternativa 5	Esta alternativa se descarta ya que almacenar todos las divisas, acciones y bitcoins en HashTables es eficiente con respecto al tiempo, no tiene nada que ver con el contexto del problema.
Alternativa 6	Esta alternativa se descarta debido a que almacenar todos los datos en pilas, no resultaría nada eficiente en complejidad espacial y temporal.
Alternativa 7	Esta alternativa se descarta debido a que almacenar todos los datos en colas, no resultaría nada eficiente en complejidad espacial y temporal, además, no podríamos consultar los datos sin tender que eliminar otros asiendo que perdamos información.

Fase 5: Evaluación y selección de la mejor solución.

Actualmente tenemos dos alternativas de solución, las cuales resuelven nuestro problema, como sólo necesitamos una alternativa, definiremos una serie de criterios que nos ayudaran a escoger la mejor solución para nuestro problema.

- **Criterios**

  - **Criterio A: Representación fiel al contexto del problema.**
    Este criterio se basa en que tan representativa es la alternativa al contexto del problema.
    - Fiel: 3 puntos
    - Regular: 2 puntos
    - Mala: 1 punto

  - **Criterio B: Complejidad Espacial en Acciones y Divisas**
    Este criterio se basa en que tan eficiente en espacio de memoria es la alternativa para las acciones y divisas para la solución del 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 C: Complejidad Temporal en Acciones y Divisas

Este criterio se basa en que tan eficiente es en tiempo la alternativa para las acciones y divisas para la solución del problema.

- Constante: 6 puntos
- Logarítmica: 5 puntos
- Lineal: 4 puntos
- Polinomial: 3 puntos
- Exponencial: 2 puntos
- Factorial: 1 punto

Criterio D: Complejidad Temporal en Bitcoins

Este criterio se basa en que tan eficiente es en tiempo la alternativa para los bitcoins.

- Constante: 6 puntos
- Logarítmica: 5 puntos
- Lineal: 4 puntos
- Polinomial: 3 puntos
- Exponencial: 2 puntos
- Factorial: 1 punto

Criterio E: Complejidad Espacial en Bitcoins

Este criterio se basa en que tan eficiente en espacio de memoria es la alternativa para los bitcoins.

- Constante: 6 puntos
- Logarítmica: 5 puntos
- Lineal: 4 puntos
- Polinomial: 3 puntos
- Exponencial: 2 puntos
- Factorial: 1 punto

- **Evaluación:**

	Criterio A	Criterio B	Criterio C	Criterio D	Criterio E	Total
Alternativa 1	3	4	5	4	4	20
Alternativa 2	2	4	4	4	4	18

Con base en los resultados obtenidos la alternativa 2 va a ser descartada y por tanto la alternativa 1 será la utilizada para dar solución al problema.

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

Fase 6: Preparación de informe y especificaciones

Diseño de pruebas unitarias

- **Estructura de Datos: AvlTree**

Objetivo: Probar el correcto funcionamiento del método insert(T objeto) para diferentes casos de prueba				
Clase: AvlTree		**Método:** Instert(T object)		
Caso #	**Descripción**	**Escenario**	**Valores de entrada**	**Resultado**
1 (Caso Estándar)	Se crea un árbol AVL de Enteros, el cual contiene los enteros {10, 5, 15, 3, 8, 12, 17}	stageOne()	Se insertan dos valores enteros los cuales son: 1 ,-1	El algoritmo insert() debe de devolver el valor true, demostrando que los entero si están en el árbol AVL.
2 (Caso Interesante)	Se crea un árbol AVL de Strings, el cual contiene las siguientes cadenas {“m”, “f”,”r”,”c”,”e”,”o”,”z”}	stageTwo()	Se insertan dos strings, los cuales son: “a”,”b”	El algoritmo insert() debe de devolver el valor true, demostrando que las cadenas si están en el árbol AVL.
3 (Caso Limite)	Se crea una un árbol AVL de doubles, el cual tiene los siguientes valores de tipo double {10.0, 5.0, 15.0, 3.0, 8.0, 12.0, 17.0 }	stageThree()	Se insertan dos strings, los cuales son: 1.0,-1.0	El algoritmo insert() debe de devolver el valor true, demostrando que los doubles si están en el árbol AVL.

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

Objetivo: Probar el correcto funcionamiento del método delete(T objeto) para diferentes casos de prueba				
Clase: AvlTree		**Método:** delete(T object)		
Caso #	**Descripción**	**Escenario**	**Valores de entrada**	**Resultado**
1 (Caso Estándar)	Se crea un árbol AVL de Enteros, el cual contiene los enteros {10, 5, 15, 3, 8, 12, 17}	stageOne()	Se entregan los dos enteros a borrar con valores 10 y 12	El algoritmo delete() debe de devolver el entero con valor 12 correspondiente a la raiz, y posterior mente devuelve el entero con valor 15 correspondiente a la nueva raiz . Por ultimo retorna el tamaño del árbol AVL que debe ser 5
2 (Caso Interesante)	Se crea un árbol AVL de Strings, el cual contiene las siguientes cadenas {“m”, “f”,”r”,”c”,”e”,”o”,”z”}	stageTwo()	Se entregan los dos Strings a borrar con valores “m” y “o”	El algoritmo delete() debe de devolver el String con valor “o” correspondiente a la raiz, y posterior mente devuelve el String con valor “r” correspondiente a la nueva raiz. Por ultimo retorna el tamaño del árbol AVL que debe ser 5
3	Se crea un árbol AVL de doubles, el	stageThree()	Se entregan	El algoritmo delete() debe

(Caso Limite)	cual contiene los doubles {10.0, 5.0, 15.0, 3.0, 8.0, 12.0, 17.0}		los dos enteros a borrar con valores 10.0 y 12.0	de devolver el entero con valor 12.0 correspondiente a la raiz, y posterior mente devuelve el entero con valor 15.0 correspondiente a la nueva raiz . Por ultimo retorna el tamaño del árbol AVL que debe ser 5

Objetivo: Probar el correcto funcionamiento del método search(T objeto) para diferentes casos de prueba				
Clase: AvlTree			**Método:** search(T object)	
Caso #	**Descripción**	**Escenario**	**Valores de entrada**	**Resultado**
1 (Caso Estándar)	Se crea un árbol AVL de Enteros, el cual contiene los enteros {10, 5, 15, 3, 8, 12, 17}	stageOne()	Se entrega un entero a buscar con valor 333	El algoritmo search() debe de devolver true si el elemento con valor 333 está en el árbol y false en caso contrario
2 (Caso Interesante)	Se crea un árbol AVL de Strings, el cual contiene las siguientes cadenas {“m”, “f”,”r”,”c”,”e”,”o”,”z”}	stageTwo()	Se entrega un String a buscar con valor “a”	El algoritmo search() debe de devolver true si el elemento con valor “a” está en el árbol y false en caso contrario
3 (Caso Limite)	Se crea un árbol AVL de doubles, el cual contiene los doubles {10.0, 5.0,	stageThree()	Se entrega un entero a buscar	El algoritmo search() debe de devolver true si el elemento con valor 333.0

	15.0, 3.0, 8.0, 12.0, 17.0}		con valor 333.0	está en el árbol y false en caso contrario

- **Estructura de Datos: RedBlackTree**

Objetivo: Probar el correcto funcionamiento del método insert(T objeto) para diferentes casos de prueba				
Clase: RedBlackTree		**Método:** Instert(T object)		
Caso #	**Descripción**	**Escenario**	**Valores de entrada**	**Resultado**
1 (Caso Estándar)	Se crea un árbol roji-negro de Enteros, el cual contiene los enteros {10, 5, 15, 3, 8, 12, 17}	stageOne()	Se insertan dos valores enteros los cuales son: 20 y 22	El algoritmo insert() debe de devolver el valor true, demostrando que los enteros si están en el árbol roji-negro.
2 (Caso Interesante)	Se crea un árbol roji-negro de Strings, el cual contiene las siguientes cadenas {“m”, “f”,”r”,”c”,”e”,”o”,”z”}	stageTwo()	Se insertan dos strings, los cuales son: “a”,”q”	El algoritmo insert() debe de devolver el valor true, demostrando que las cadenas si están en el árbol roji-negro.
3 (Caso Limite)	Se crea una un árbol roji-negro de doubles, el cual tiene los siguientes valores de tipo double {10.0, 5.0, 15.0, 3.0, 8.0, 12.0, 17.0 }	stageThree()	Se insertan dos strings, los cuales son: 20.0, 22.0	El algoritmo insert() debe de devolver el valor true, demostrando que los doubles si están en el árbol roji-negro.

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

Objetivo: Probar el correcto funcionamiento del método delete(T objeto) para diferentes casos de prueba				
Clase: RedBlackTree		**Método:** delete(T object)		
Caso #	**Descripción**	**Escenario**	**Valores de entrada**	**Resultado**
1 (Caso Estándar)	Se crea un árbol roji-negro de enteros, el cual contiene los enteros {10, 5, 15, 3, 8, 12, 17}	stageOne()	Se entregan los dos enteros a borrar con valores 10 y 8	El algoritmo delete() debe de devolver el entero con valor 8 correspondiente a la raiz, y posterior mente devuelve el entero con valor 5 correspondiente a la nueva raiz.
2 (Caso Interesante)	Se crea un árbol roji-negro de Strings, el cual contiene las siguientes cadenas {“m”, “f”,”r”,”c”,”e”,”o”,”z”}	stageTwo()	Se entregan los dos Strings a borrar con valores “m” y “f”	El algoritmo delete() debe de devolver el String con valor “f” correspondiente a la raiz, y posterior mente devuelve el String con valor “e” correspondiente a la nueva raiz.
3 (Caso Limite)	Se crea un árbol roji-negro de doubles, el cual contiene los doubles {10.0, 5.0, 15.0, 3.0, 8.0, 12.0, 17.0}	stageThree()	Se entregan los dos enteros a borrar con valores 10.0 y 8.0	El algoritmo delete() debe de devolver el entero con valor 8.0 correspondiente a la raiz, y posterior mente devuelve el

				entero con valor 5.0 correspondiente a la nueva raiz .

Objetivo: Probar el correcto funcionamiento del método search(T objeto) para diferentes casos de prueba				
Clase: RedBlackTree		**Método:** search(T object)		
Caso #	**Descripción**	**Escenario**	**Valores de entrada**	**Resultado**
1 (Caso Estándar)	Se crea un árbol roji-negro de Enteros, el cual contiene los enteros {10, 5, 15, 3, 8, 12, 17}	stageOne()	Se entrega un entero a buscar con valor 333	El algoritmo search() debe de devolver true si el elemento con valor 333 está en el árbol roji-negro
2 (Caso Interesante)	Se crea un árbol roji-negro de Strings, el cual contiene las siguientes cadenas {“m”, “f”,”r”,”c”,”e”,”o”,”z”}	stageTwo()	Se entrega un String a buscar con valor “a”	El algoritmo search() debe de devolver true si el elemento con valor “a” está en el árbol roji-negro
3 (Caso Limite)	Se crea un árbol roji-negro de doubles, el cual contiene los doubles {10.0, 5.0, 15.0, 3.0, 8.0, 12.0, 17.0}	stageThree()	Se entrega un entero a buscar con valor 333.0	El algoritmo search() debe de devolver true si el elemento con valor 333.0 está en el árbol roji-negro

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

- **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.

Árbol AVL:

Método 1:

#	search(NodeAVL<T> node, T value)	C.E	C.T
1	`if (node == null)`	0	1
2	`return false`	0	1
3	`int cmp = value.compareTo(node.value)`	1	1
4	`if (cmp < 0)`	0	1
5	`return search(node.left, value)`	0	N
6	`if (cmp > 0)`	0	1
7	`return search(node.right, value)`	0	N
8	`return true`	0	1

Método 2:

#	insert(NodeAVL<T> node, T value)	C.E	C.T
1	`if (node == null)`	0	1
2	`return new NodeAVL<T>(value)`	0	1
3	`int cmp = value.compareTo(node.value)`	1	1
4	`if (cmp < 0)`	0	1
5	`node.left = insert(node.left, value)`	1	N
6	`else`		0
7	`node.right = insert(node.right, value)`	1	N
8	`update(node)`	0	1
9	`return balance(node)`	0	1

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

Método 3:

#	insert(T value)	C.E	C.T
1	`if (value == null)`	0	1
2	`return false`	0	1
3	`if (!search(root, value))`	0	1
4	`root = insert(root, value)`	1	N
5	`nodeCount++`	1	1
6	`return true`	0	1
7	`return false`	0	1

Árbol rojo-negro:

Método 4:

#	delete(T date)	C.E	C.T
1	`if( root == null )`	0	1
2	`throw new ElementDontExist( "El Ã¡rbol se encuentra vacio" )`	0	1
3	`if( root.getInfoNode().compareTo( date ) == 0 & root.sheetRightSon() & root.sheetLeftSon()) {`	0	1
4	`root = null`	1	1
5	`else`	0	
6	`NodeRB<T> r2 = root.getNode( date ).delete( )`	1	N
7	`root = r2 != null && r2.getFather( ) == null ? r2 : root`	1	1

Método 5:

#	deleteCaseOne(Root r)	C.E	C.T
1	`if( father != null )`	0	1
2	`this.deleteCaseTwo( r )`	0	N
3	`else`		
4	`r.answer = null`	1	1

Método 6:

#	deleteCaseThree(Root r)	C.E	C.T
1	`NodeRB<T> hermano = getBrother( )`	1	1
2	`if( father.color == BLACK && hermano.color == BLACK && hermano.blackSons( ) )`	0	1
3	`hermano.setColor( RED )`	0	1
4	`father.deleteCaseOne( r )`	0	1
5	`else`		
6	`deleteCaseFour( r )`	0	N

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

- **Análisis de complejidad espacial**

✓ **Método 1:**
T(n) = 1
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) = 3
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) = 2
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) = 3
T(n) = C
Por medio del T(n) podemos concluir que la notación asintótica para la complejidad espacial del método 4 es: $O(1)$.

✓ **Método 5:**
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: $O(1)$.

✓ **Método 6:**
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: $O(1)$.

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

- **Análisis de complejidad temporal**

✓ **Método 1:**
T(n) = $2n + 6$
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) = $2n + 6$
T(n) = $An + B$
Por medio del T(n) podemos concluir que la notación asintótica para la complejidad temporal del método 2 es: $O(n)$.

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

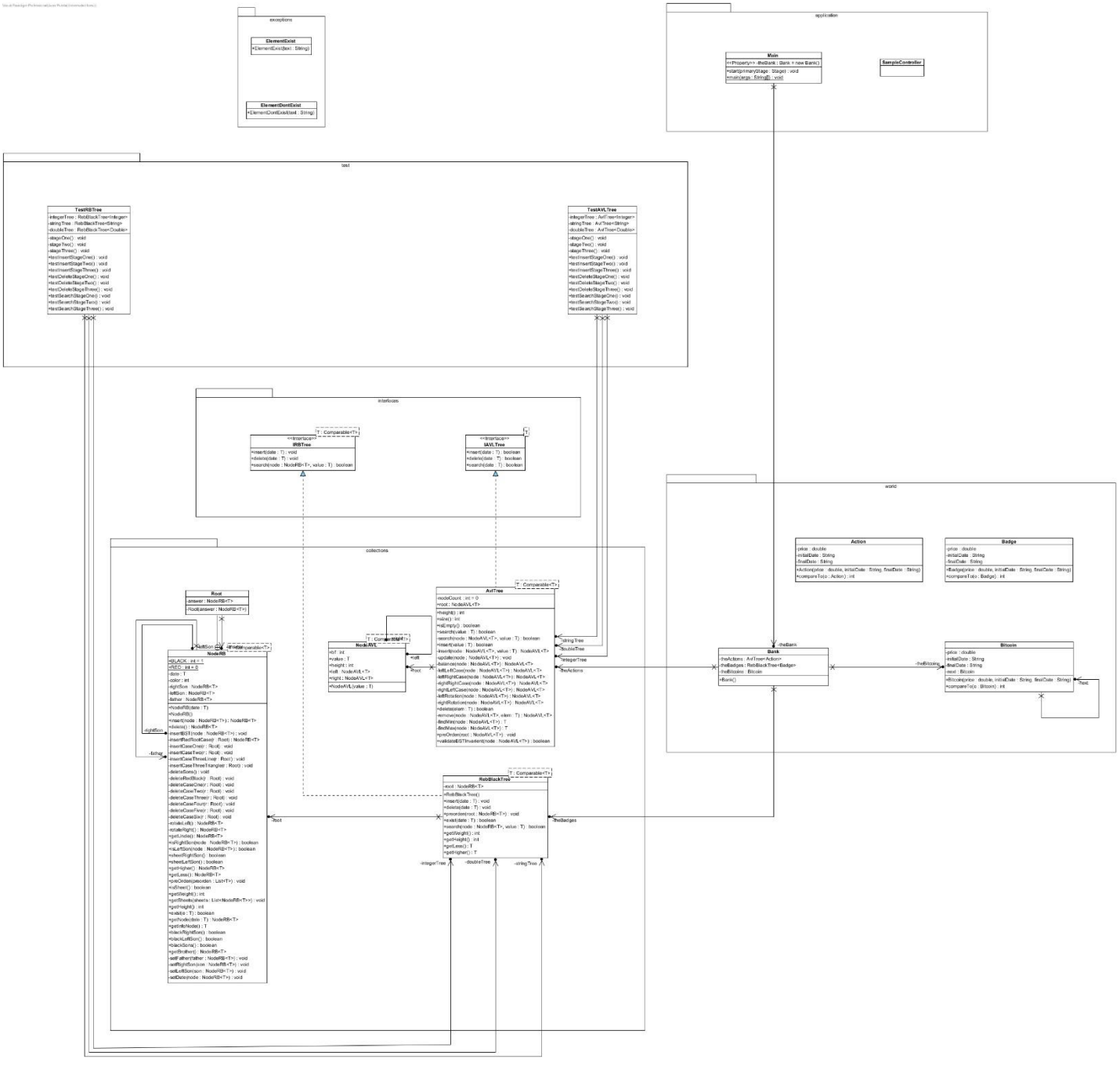
✓ **Método 4:**
T(n) = $n + 5$
T(n) = $n + B$
Por medio del T(n) podemos concluir que la notación asintótica para la complejidad espacial del método 4 es: $O(n)$.

✓ **Método 5:**
T(n) = $n + 2$
T(n) = $n + B$
Por medio del T(n) podemos concluir que la notación asintótica para la complejidad espacial del método 4 es: $O(n)$.

✓ **Método 6:**
T(n) = $n + 4$
T(n) = $n + B$
Por medio del T(n) podemos concluir que la notación asintótica para la complejidad espacial del método 4 es: $O(n)$.

Juan Sebastián Puerta
Fabian David Portilla
Juan Camilo Castillo

- **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/Lab3-AED