UNIVERSIDAD ICESI	**Tarea Integradora #2** **Facultad de Ingeniería** **Algoritmos y Estructuras De Datos** **Grupo: 01** **Semestre 2021 - B**

FIBA Data Base

Integrantes:

Juan David Ballesteros Valencia - A00306456

Camilo González Velasco - A00370263

Samuel Guerrero Viveros – A00365567

DOCENTE: Johnatan Garzón Montesdeoca

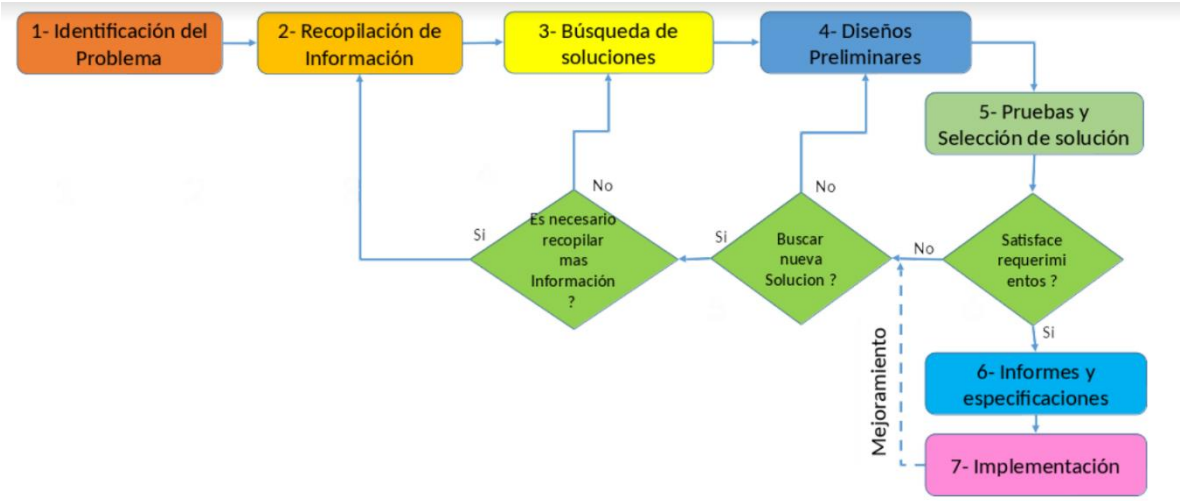
Contenido

MÉTODO DE LA INGENIERÍA

Contexto: El basketball al ser uno de los deportes más populares, importantes e influyentes alrededor del globo es propenso a evolucionar. Con el trascurrir de los años y a medida que el profesionalismo ha ido avanzando se requiere llevar a cabo un seguimiento a detalle con el fin de mejorar el juego, sus dinámicas, sus estrategias o, en general, observar hacia donde se dirige el deporte en la actualidad. Logrando que sus millones de aficionados disfruten del mejor espectáculo posible. Lo anterior se obtiene con un gran análisis de datos "estadísticas" recolectar datos, ordenarlos y clasificarlos, una gran tarea teniendo en cuenta la cantidad de jugadores que ejercen como profesionales este deporte.

Desarrollo de la solución:

Para resolver la situación anterior se eligió el Método de la Ingeniería para desarrollar la solución siguiendo un enfoque sistemático y acorde con la situación problemática planteada. Con base en la descripción del Método de la Ingeniería del libro "Introduction to Engineering" de Paul Wright, se definió el siguiente diagrama de flujo, cuyos pasos seguiremos en el desarrollo de la solución.



I. IDENTIFICACIÓN DEL PROBLEMA

Definición del problema

La Federación Internacional de Baloncesto, mejor conocida como FIBA desea consolidar un software que permita almacenar y acceder de forma rápida a la información de cada uno de los jugadores de este deporte a nivel mundial.

Necesidades y Síntomas

La FIBA ha solicitado la implementación de una herramienta para el manejo de grandes cantidades de información que permita ingresar datos, eliminar o modificar.

- La solución debe permitir agregar información de manera masiva a través de archivos .csv.
- La solución debe permitir agregar información de manera manual a través de una interfaz gráfica.
- La solución debe brindar una forma de búsquedas rápidas y eficientes mediante uno o varios criterios de búsqueda (en este caso parámetros estadísticos)
- La solución debe almacenar y procesar la información en memoria secundaria.

II. RECOPILACIÓN DE INFORMACIÓN

Hay algunas definiciones que son necesarias abordar para poder entender en su totalidad el problema, algunos de ellos son conceptos técnicos en informática y otros más relacionados con al lenguaje del deporte.

CRUD: (Create, read, update,delete) Representa las operaciones básicas que debe tener una implementación de una base de datos, los cuales son: crear, leer, actualizar y eliminar información de la misma.

Complejidad algorítmica: Concepto matemático que describe la eficiencia en recursos informáticos a la hora de realizar una secuencia de operaciones.

Data set: Conjunto o colección de datos especifica, recopilada de varias fuentes orientada a características o aspectos específicos.

Árbol binario de búsqueda (ABB o BTS): Es una estructura de datos en informática vista como un árbol ordenado, en el que cada nodo tiene 0, 1 ó 2 hijos (hijo izquierdo e hijo derecho)

Árbol binario de búsqueda auto balanceada (AVL): En informática, es una estructura para gestionar datos la cual nos ofrece una complejidad garantizada (log n) en sus principales operaciones: búsqueda, inserción y eliminación.

Árbol rojo y negro (R&N): Un árbol rojo-negro es un tipo abstracto de datos. Concretamente, es un árbol binario de búsqueda equilibrada, una estructura de datos utilizada en informatica y ciencias de la computación. [1]La estructura original fue creada por Rudolf Bayer en 1972, quien le dio el nombre de "árbol-B binarios simétricos", pero tomó su nombre moderno en un trabajo de Leo J. Guibas y Rober Sedgewick realizado en 1978.

Punto: En el baloncesto, un *punto* se utiliza para realizar un seguimiento de la puntuación en un partido. Los puntos pueden ser acumulados por hacer tiros de campo (dos o tres puntos) o tiros libres (un punto). Si un jugador anota un tiro de campo dentro de la línea de tres puntos, el jugador anota dos puntos. Si el jugador anota un tiro de campo más allá de la línea de tres puntos, el jugador marca tres puntos. El equipo que ha registrado el mayor número de puntos al final de un partido es declarado ganador de ese partido.

Asistencia: Una asistencia es un pase a un jugador que se encuentra en una posición de ventaja o que le ayuda a conseguir una canasta.

Rebote: Un *rebote* en baloncesto es el acto de conseguir la posesión del balón después de un lanzamiento de campo o de un tiro libre fallado. Son uno de los objetivos fundamentales de los jugadores altos de un equipo, aunque cualquier jugador en pista puede intentar conseguirlos.

[1] Tomado de: Tomado de wikipedia

Robo: Los robos se acreditan al jugador defensivo que causa primero la perdida de balón, incluso si no terminan con la posesión de la pelota viva. Para ganar un robo, el jugador defensivo debe ser el iniciador de la acción que causo la perdida de balón, no solo el benefactor. Cada vez que un robo se registra por un jugador defensivo, un jugador ofensivo debe ser acreditado con un balón perdido.

III. BÚSQUEDA DE SOLUCIONES CREATIVAS

Para desarrollar las alternativas expuestas a continuación se realizó la técnica de lluvia de ideas: Generación espontánea de ideas diseñadas para resolver un problema. Sin embargo, cabe destacar que la solución tiene que ser una aplicación (software) que cumpla con las indicaciones requeridas por la FIBA. Por ende, las alternativas se dividirán por los criterios que exige la solución.

Almacenar información de manera masiva

a. Cargar datos a través de archivos .csv.

b. Añadir jugadores a través de interfaz

c. Elaborar un módulo de software que use información a partir de un conjunto de datos de diseño propio.

Búsquedas eficientes con varios criterios

a. Búsqueda de datos de jugadores por parámetro.

b. Búsqueda binaria en tablas de hash.

c. Implementación de estructuras como grafos, que brindan un recorrido parcial dada una secuencia de símbolos.

d. Búsqueda binaria a través de árboles binarios de búsqueda.

e. Búsqueda a través de árboles binarios de búsqueda auto-balanceados (AVL) , (R&N)

Eliminación de datos:

a. Solicitar al usuario el dato a eliminar y luego removerlo de la base de datos.

b. Eliminar datos por parámetros.

c. Eliminar datos por interfaz.

Es propicio mencionar que los ejes centrales con los cuales la solución se creo son la manera en que se carga los datos y la estructuras para búscalos. A raíz de estos dos factores la solución se adapta.

IV. TRANSICIÓN DE LAS IDEAS A LOS DISEÑOS PRELIMINARES

En esta fase vamos a descartar las peores alternativas que no brindan una solución adecuada a las necesidades del problema.

Almacenar información de manera masiva

Alternativa C: Elaborar un módulo de software que use información a partir de un conjunto de datos de diseño propio. Esta idea se descarta porque es muy ineficiente ya que, tomaría mucho tiempo recolectar datos de esta manera.

Búsquedas eficientes con varios criterios

Alternativa A: Búsqueda de datos de jugadores por parámetro. Esta búsqueda es una opción poco eficiente, dado que buscar los jugadores por parámetros sería una búsqueda línea la cual tardaría demasiado (dependiendo del dispositivo o computador) aunque de igual manera dejando de lados factores técnicos este tipo de búsqueda no es eficaz.

Alternativa B: Búsqueda binaria en tablas de hash. Aunque las **tablas hash** brinden una baja complejidad temporal en su búsqueda en varios casos, eso no sucede cuando se trabaja con mucha información, por lo que no es eficiente para el problema planteado.

Alternativa C: Implementación de estructuras como grafos, que brindan un recorrido parcial dada una secuencia de símbolos. La estructura de los grafos no se ha tratado en el semestre y debido a esto no se puede realizar la implementación de estos. No sin disponer tiempo suficiente para hacer una investigación sobre el tema, tiempo que no se posee.

Eliminación de datos:

Por ahora no se descarta ninguna opción disponible en este apartado.

V. EVALUACIÓN Y SELECCIÓN DE LA MEJOR SOLUCIÓN

Para evaluar las soluciones planteadas anteriormente, se plantearán diversos criterios para cada problema propuesto hallando así la mejor solución en cada contexto.

Criterios:

Almacenar información de manera masiva

Criterio A. Estabilidad de eficiencia para cualquier caso ▪

- [1] Baja
- [2] Media
- [3] Alta

Criterio B. Eficiencia.

- [1] Media
- [2] Alta

Búsquedas eficientes con varios criterios

Criterio A. Capacidad para mantener la misma complejidad

- [1] Incapaz
- [2] Capaz

Eliminación de datos

Criterio A. Eliminación simple de los datos.

[1] Baja
[2] Media
[3] Alta

Almacenar información de manera masiva:

	Criterio A	Criterio B	Total
Alternativa A	3	2	5
Alternativa B	2	1	3

Se escoge la alternativa a, la cual es cargar los datos leyendo archivos .csv, aunque es preciso aclarar que la alternativa B no es mala opción e incluso será utilizada para la solución del problema, pero solo es utilizada para agregar una cantidad muy pequeña datos que el usuario desee.

Búsquedas eficientes con varios criterios:

	Criterio A	**Total**
Alternativa D	**1**	**1**
Alternativa E	**2**	**2**

Se escoge la alternativa E la cual corresponde a los arboles auto-balanceados AVL o Rojo y negro. Sin embargo, para fines prácticos de este proyecto se utilizarán arboles auto-balanceados y no balanceados, evidenciando la diferencia en los tiempos de búsqueda de los datos.

Eliminación de datos:

	Criterio A	**Total**
Alternativa A	**2**	**2**
Alternativa B	**1**	**1**
Alternativa C	**3**	**3**

Se escoge la alternativa C la cual permite eliminar todos los datos mediante un solo botón en la GUI.

En definitiva, el programa empleara como estructuras de datos arboles auto-balanciados como lo son el árbol AVL y R&N y arboles binarios de búsqueda con el fin de comparar y evidenciar la diferencia en los tiempos de búsqueda. Por otro lado, el programa cargara los datos leyendo archivos de formato .csv y importara los datos cuando el usuario los requiera, guardando los cambios que se realicen en el archivo y de esta manera manteniendo una persistencia por lectura de archivo y no por serialización dado que esta opción haría al programa muy lento, el simple caso de cargar los datos siempre que se abra el programa y guardar en cada momento los cambios quita eficiencia. Cabe mencionar que al leer los datos por .csv se mantiene todo en memoria secundaria.

VI. PREPARACIÓN DE INFORMES Y ESPECIFICACIONES

El TAD del árbol binario de búsqueda se encuentra en la carpeta docs del proyecto: https://github.com/camilogonzalez7424/FIBA-data base/blob/master/docs/TADBSTREE.pdf

El TAD del árbol binario de búsqueda auto-balanceado (AVL) se encuentra en el siguiente en la carpeta docs del proyecto: https://github.com/camilogonzalez7424/FIBA-data-base/blob/master/docs/TADAVL.pdf

El TAD del árbol binario de búsqueda auto-balanceado (R&N) se encuentra en la carpeta docs del proyecto: https://github.com/camilogonzalez7424/FIBA-data-base/blob/master/docs/TADRBT.pdf

VII. IMPLEMENTACIÓN DEL DISEÑO

La implementación del diseño fue realizada en el lenguaje de programación Java, junto con Scene Builder el cual se usó para construir la interfaz gráfica GUI. El proyecto se encuentra almacenada en el siguiente repositorio: https://github.com/camilogonzalez7424/FIBA-data-base

Cabe destacar que se probó y se implementó la solución final con el modelo MVC (Modelo vista controlador) dado que se iba a trabajar en diferentes pantallas y de esta manera se podía independizar ciertas áreas del código obteniendo escalabilidad.

Bibliografía:

colaboradores de Wikipedia. (s. f.). *Robo (baloncesto - Wikipedia, la enciclopedia libre*. Robo Baloncesto. Recuperado 22 de octubre de 2021, de https://es.wikipedia.org/wiki/Robo_(baloncesto

colaboradores de Wikipedia. (2021a, julio 1). *Asistencia (baloncesto)*. Wikipedia, la enciclopedia libre. Recuperado 22 de octubre de 2021, de https://es.wikipedia.org/wiki/Asistencia_(baloncesto)

colaboradores de Wikipedia. (2021b, septiembre 29). *Punto (baloncesto)*. Wikipedia, la enciclopedia libre. Recuperado 20 de octubre de 2021, de https://es.wikipedia.org/wiki/Punto_(baloncesto)#:%7E:text=En%20el%20baloncesto%2C%20un%20punto,el%20jugador%20anota%20dos%20puntos.

colaboradores de Wikipedia. (2021c, octubre 28). *Rebote (baloncesto)*. Wikipedia, la enciclopedia libre. Recuperado 22 de octubre de 2021, de https://es.wikipedia.org/wiki/Rebote_(baloncesto)#:%7E:text=Un%20rebote%20en%20baloncesto%20es,en%20pista%20puede%20intentar%20conseguirlos.

Imagen tomada de: https://c8.alamy.com/compes/pjxnan/silueta-de-un-jugador-de-baloncesto-puntos-lineas-triangulos-efectos-de-color-y-el-fondo-en-capas-separadas-el-color-se-puede-cambiar-en-un-solo-clic-vecto-pjxnan.jpg

Requerimientos

Req.1. Guardar datos de mayor relevancia de cada uno de los profesionales del baloncesto en el planeta.

Req.1.1 Guardar el nombre de los jugadores.

Req.1.2 Guardar la edad de los jugadores.

Req.1.3 Guardar el equipo de baloncesto al que pertenecen de los jugadores.

Req.1.4 Guardar el registro de puntos de los jugadores.

Req.1.5 Guardar el registro de rebotes de los jugadores.

Req.1.6 Guardar el registro de asistencias de los jugadores.

Req.1.7 Guardar el registro de partidos jugados de los jugadores.

Req.1.8 Guardar el registro de robos de los jugadores.

Req.2. Permitir ingresar datos.

Req.2.1. Ingresar información de manera masiva (con archivos .csv por ejemplo).

Req.2.2. Ingresar información a través de una interfaz.

Req.3. Permitir efectuar diferentes consultas que permitan realizar análisis sobre estos datos en la base.

Req.4. Eliminar registros.

Req.5. Modificar registros.

Req.6. Permite realizar consultas de jugadores utilizando como criterios de búsqueda las categorías estadísticas incluidas (por ejemplo, encontrar aquellos jugadores que han anotado 10 puntos por partido, o aquellos con más de 20 rebotes por partido).

Req.6.1. Filtrar por nombre.

Req.6.2. Filtrar por edad.

Req.6.3. Filtrar por equipo.

Req.6.4. Filtrar por alguna de 5 estadísticas (e.g. puntos por partido, rebotes por partido, asistencias por partido, robos por partido, bloqueos por partido).

Req.7. Almacenar millones de datos.

Req.8. Brindar a rapidez para el acceso de los datos para cuatro parámetros.

Req.9. Guardar la información en memoria secundaria.

Req.10. Mostar el tiempo que tarda buscar la información del jugador solicitado.

DISEÑO DE PRUEBAS UNITARIAS

Diseño de escenarios

Name	Class	Stage
setUpScenery1	App	
setUpScenery1	ABB	
setUpScenery1	AVLTreeNode	
setUpScenery1	AVLTree	
setUpScenery2	AVLTree	
setUpScenery1	RedAndBlackTree	

Clase	Metodo	Escenario	Valores de entrada	Resultado
AVLTreeTest	testAdd()	setUpScenery1	k=10, v=20 k=11, v=20 k=12, v=20 k=13, v=20	Los valores agregados al arbol árbol de búsqueda binario auto-balanceable se agreagan y enlazan correctamente entre ellos. Se referencian correcatmente por medio de sus ramas balanceadas.

AVLTreeTest	testRemove()	setUpScenery1	k=15, v=7	Elemento a eliminar de forma correcta de acuerdo al parametro de busqueda
AVLTreeTest	testRemove()	setUpScenery2	k=10, v=null k=5, v=null k=6, v=null k=3, v=null k=4, v=null k=7, v=null k=15, v=null k=12, v=null k=17, v=null k=16, v=null	Elemento a eliminar de forma correcta de acuerdo al parametro de busqueda
AVLTreeTest	testSearch()	setUpScenery1	k=15, v=7	Elemento a buscar de forma correcta de acuerdo al parametro de busqueda
AVLTreeTest	testSearch()	setUpScenery2	k=10, v=null k=5, v=null k=6, v=null k=3, v=null k=4, v=null k=7, v=null k=15, v=null k=12, v=null k=17, v=null k=16, v=null	Elemento a buscar de forma correcta de acuerdo al parametro de busqueda
AVLTreeTest	testIsEmpty()	setUpScenery1	k=2, v=404	Metodo que retorna si esta vacio o no.
AVLTreeTest	testKeyExists()	setUpScenery1	k=2, v=404 k=3, v=202	Metodo que retorna si esta vacio o no.
AVLTreeTest	testRotateLeftCaseA()	setUpScenery2	-	Ajusta el balanceo del arbol de acuerdo al caso de desbalance que se encuentre

AVLTreeTest	testRotateLeftCaseB()	setUpScenery3	-	Ajusta el balanceo del arbol de acuerdo al caso de desbalance que se encuentre
AVLTreeTest	testRotateLeftCaseC()	setUpScenery4	-	Ajusta el balanceo del arbol de acuerdo al caso de desbalance que se encuentre
AVLTreeTest	testRotateLeftCaseD()	setUpScenery5	-	Ajusta el balanceo del arbol de acuerdo al caso de desbalance que se encuentre
AVLTreeTest	testRotateLeftCaseE()	setUpScenery6	-	Ajusta el balanceo del arbol de acuerdo al caso de desbalance que se encuentre
AVLTreeTest	testRotateLeftCaseF()	setUpScenery7	-	Ajusta el balanceo del arbol de acuerdo al caso de desbalance que se encuentre
ABBTest	testInsert()	setUpScenery1	k=4, v=5 k=5, v=4 k=6, v=8	Los valores agregados al arbol árbol de búsqueda binario se agreagan y enlazan correctamente entre ellos. Se referencian correcatmente por medio de sus ramas.
ABBTest	testSearch()	setUpScenery1, setUpScenery2	k=1, v=305 k=2, v=35 k=3, v=23	Elemento a buscar de forma correcta de acuerdo al parametro de busqueda
ABBTest	testSearchNode()	setUpScenery1	k=4, v=5 k=5, v=4 k=6, v=8	Metodo que retorna correctamente el nodo buscado de acuerdo con el parametro de busqueda.

RedAndBlackTreeTest	testInsert()	setup1	k=10, v=1 k=12, v=2 k=15, v=3 k=5, v=4 k=6, v=5	Los valores agregados al arbol árbol de búsqueda binario se agreagan y enlazan correctamente entre ellos. Se referencian correcatmente por medio de sus ramas.
RedAndBlackTreeTest	testSearch()	setup1	k=12, v=2 k=5, v=7 k=6, v=9 k=77, v=4	Elemento a buscar de forma correcta de acuerdo al parametro de busqueda
RedAndBlackTreeTest	testRemove()	setup1	k=12, v=2 k=5, v=7	Elemento a eliminar de forma correcta de acuerdo al parametro de busqueda
AppTest	testApp()	setUpScenery1		Prueba correctamente la funcionalidad de la app
AppTest	testAddPlayer()	setUpScenery1	("El bicho,19,Tapitas,3,4,5,6,7",app.getPlayers().toString().replaceAll("\\[|\\]", ""))	Agregar un jugador de manera correcta
AppTest	testClean()	setUpScenery1	("El bicho,19,Tapitas,3,4,5,6,7",app.getPlayers().toString().replaceAll("\\[|\\]", ""))	Clean

DIAGRAMA DE CLASES