Tarea integradora 3

FIBA DATA MANAGEMENT

Giovanni Mosquera Diazgranados | A00365672
Juan Pablo Sanin | A00296776
Juan Camilo Zorrilla Calvache | A00365972
Juan Sebastián Rodriguez | A00365843

"Un aplicativo moderno para manejar datos de forma masiva con estructuras de datos modernas las cuales facilitan su almacenamiento y accesiblidad posterior para realizar consultas y demás, interfáz gráfica implementada"

Requerimientos Funcionales

- RF1: El programa debe cargar los datos sobre jugadores junto a sus estadisticas de forma masiva dentro de la aplicación con archivos de extensión .CSV.
- RF2: El programa debe realizar consultas de forma rápida según la estructura de datos utilizada para la categorización de la información seleccionada, los criterios pueden ser rubros estadísticos o sencillamente el nombre del jugador.
- RF3: La información del programa deberá estar guardada en memoria secundaria y asegurar un rápido acceso a los datos ya que la cantidad es masiva.
- RF4: El programa debe realizar consultas según el rubro estadístico seleccionado por el usuario y asegurar una rápida respuesta, estas consultas pueden ser con índices o no.
- RF5: El programa debe poder filtrar la información con índices superior e inferior.
- RF6: El programa debe tener 4 árboles AVL y 1 árbol ABB como estructuras de datos que almacenan los rubros estadísticos.
- RF7: El programa debe utilizar una base de datos con mínimo 200 mil datos.
- RF8: El programa debe mostrar el tiempo que tarda una búsqueda.

BOCETO DISEÑO DE LA INTERFÁZ

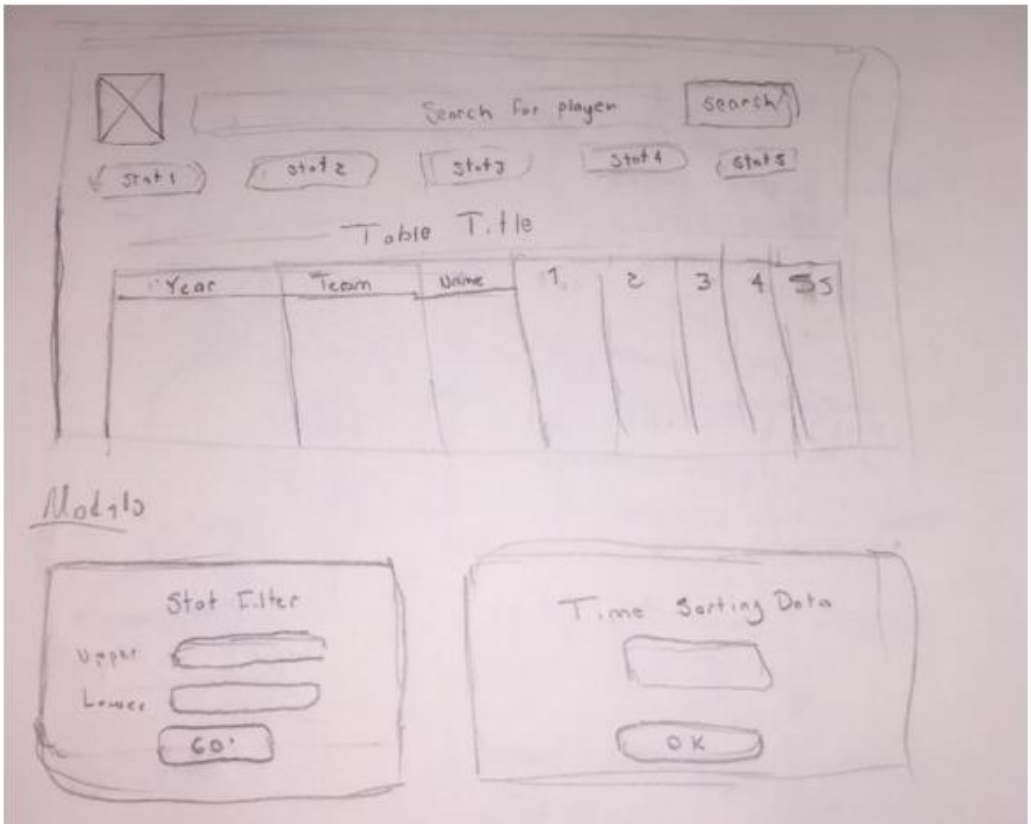
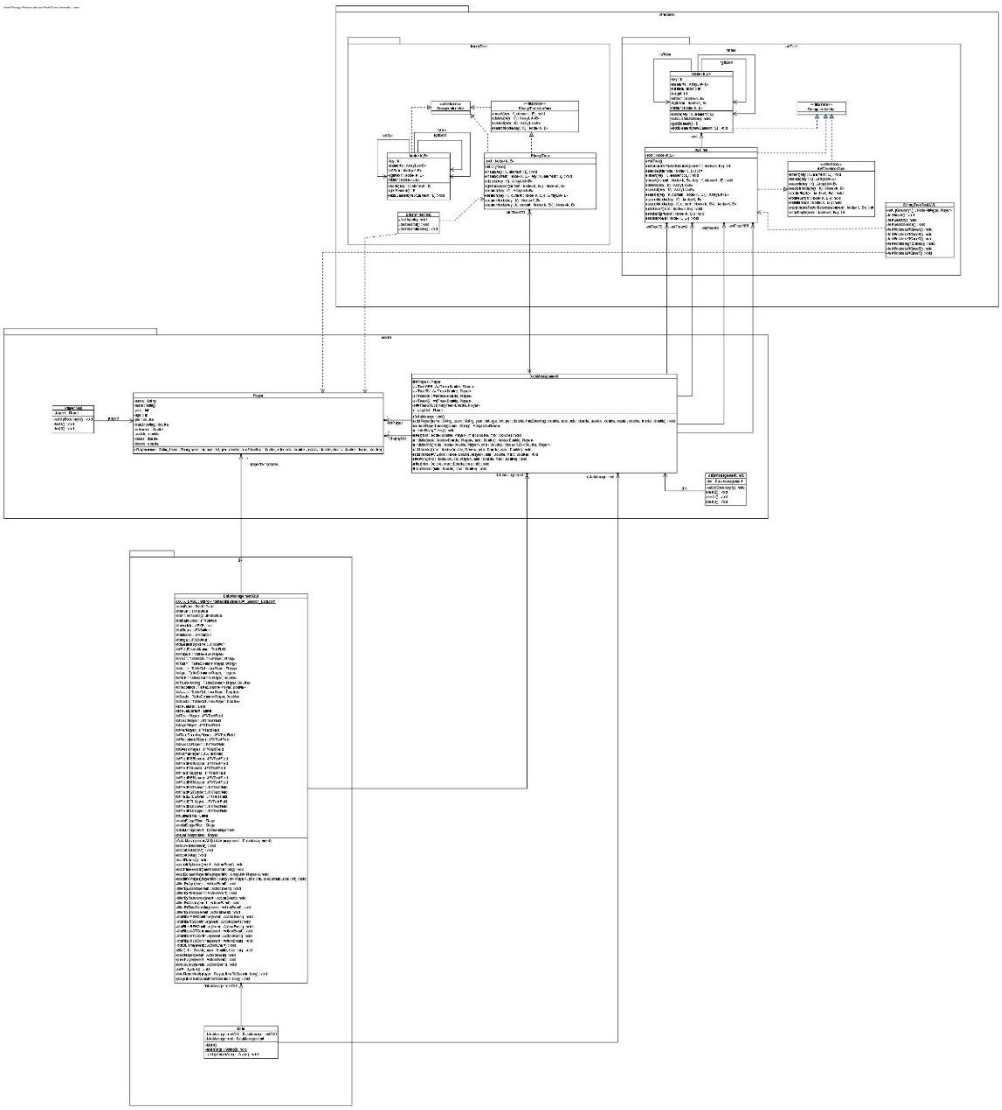


Diagrama de Clases



Diseño de casos de prueba

Nombre	Clase	Escenario
setUpEscenary1	Player	"michael", "CHI", 1990, 40, 2, 1, 20, 39, 0.4, 20
setUpEscenary1	DataManagement	dm = new DataManagement() "michael", "CHI", 1990, 40, 2, 1, 20, 39, 0.4, 20 “cobe", "CHI", 1985, 40, 2, 1, 20, 39, 0.4, 20 “lebron", "CHI", 1995, 40, 2, 1, 20, 39, 0.4, 20) “lebron", "CHI", 1993, 40, 2, 1, 20, 39, 0.4, 20) “lebron", "CHI", 1997, 40, 2, 1, 20, 39, 0.4, 20) “Pepe", "CHI", 1997, 40, 2, 1, 20, 39, 0.4, 20);
setupScenary1	BinaryTreeTestAVL	("Sanin", "HOU", 2000, 20, 4, 5, 20, 30, 2, 30); ("Giovanni", "CHI", 2002, 18, 2, 2.1, 3, 5, 1, 10); ("Sebastian", "BOS", 2003, 17, 1, 1, 2, 3, 4, 6);

Clase	Metodo	Escenario	Valores de entrada	Resultado
BinaryTreeTest	testInsert()		"michael", "CHI", 1990, 40, 2, 1, 20, 39, 0.4, 20); “cobe", "CHI", 1985, 40, 2, 1, 20, 39, 0.4, 20); "lebron", "CHI", 1995, 40, 2, 1, 20, 39, 0.4, 20); “raul", "CHI", 1993, 40, 2, 1, 20, 39, 0.4, 20); "lebron", "CHI", 1997, 40, 2, 1, 20, 39, 0.4, 20); “james", "CHI", 1990, 40, 2, 1, 20, 39, 0.4, 20);	Los valores agregados al árbol de búsqueda binaria se agregan y se enlazan correctamente entre ellos. Y se referencian correctamente por medio de sus ramas.
BinaryTreeTest	testSearch()		"michael", "CHI", 1990, 40, 2, 1, 20, 39, 0.4, 20 “cobe", "CHI", 1985, 40, 2, 1, 20, 39, 0.4, 20 “lebron", "CHI", 1995, 40, 2, 1, 20, 39, 0.4, 20) “lebron", "CHI", 1993, 40, 2, 1, 20, 39, 0.4, 20) “lebron", "CHI", 1997, 40, 2, 1, 20, 39, 0.4, 20)	La búsqueda me retorna correctamente los elementos a según el parámetro de búsqueda.
BinaryTreeTest	testSearchNode()		"michael", "CHI", 1990, 40, 2, 1, 20, 39, 0.4, 20 “cobe", "CHI", 1985, 40, 2, 1, 20, 39, 0.4, 20 “lebron", "CHI", 1995, 40, 2, 1, 20, 39, 0.4, 20) “lebron", "CHI", 1993, 40, 2, 1, 20, 39, 0.4, 20) “lebron", "CHI", 1997, 40, 2, 1, 20, 39, 0.4, 20) “Pepe", "CHI", 1997, 40, 2, 1, 20, 39, 0.4, 20);	Me retorna correctamente el nodo a buscar de acuerdo con el parámetro de búsqueda.

Clase	Metodo	Escenario	Valores de entrada	Resultado
BinaryTreeTestAVL	testInsert ()	setUpEscenary1	("michael", "CHI", 1990, 40, 2, 1, 20, 39, 0.4, 20); ("cobe", "CHI", 1985, 40, 2, 1, 20, 39, 0.4, 20); ("lebron", "CHI", 1995, 40, 2, 1, 20, 39, 0.4, 20); ("raul", "CHI", 1993, 40, 2, 1, 20, 39, 0.4, 20); ("lebron", "CHI", 1997, 40, 2, 1, 20, 39, 0.4, 20); ("james", "CHI", 1990, 40, 2, 1, 20, 39, 0.4, 20); ("Giovanni", "CHI", 2000, 40, 2, 1, 20, 39, 0.4, 20);	Los valores agregados al árbol de búsqueda binaria se agregan y se enlazan correctamente entre ellos. Y se referencian correctamente por medio de sus ramas balanceadas.
BinaryTreeTestAVL	testSearch()		"michael", "CHI", 1990, 40, 2, 1, 20, 39, 0.4, 20 “cobe", "CHI", 1985, 40, 2, 1, 20, 39, 0.4, 20 “lebron", "CHI", 1995, 40, 2, 1, 20, 39, 0.4, 20) “lebron", "CHI", 1993, 40, 2, 1, 20, 39, 0.4, 20) “lebron", "CHI", 1997, 40, 2, 1, 20, 39, 0.4, 20)	Elemento a buscar de forma correcta de acuerdo al parámetro de búsqueda
BinaryTreeTestAVL	testSearchNode()		"michael", "CHI", 1990, 40, 2, 1, 20, 39, 0.4, 20 “cobe", "CHI", 1985, 40, 2, 1, 20, 39, 0.4, 20	Me retorna correctamente el nodo a buscar de acuerdo con

			“lebron", "CHI", 1995, 40, 2, 1, 20, 39, 0.4, 20) “lebron", "CHI", 1993, 40, 2, 1, 20, 39, 0.4, 20) “lebron", "CHI", 1997, 40, 2, 1, 20, 39, 0.4, 20) “Pepe", "CHI", 1997, 40, 2, 1, 20, 39, 0.4, 20);	el parámetro de búsqueda.
BinaryTreeTestAVL	testRotateLeftCaseA()	setUpEscenary1		Ajusta el balanceo del árbol de acuerdo al caso de desbalance que se encuentre
BinaryTreeTestAVL	testRotateLeftCaseB()	setUpEscenary1		Ajusta el balanceo del árbol de acuerdo al caso de desbalance que se encuentre
BinaryTreeTestAVL	testRotateLeftCaseC()	setUpEscenary1		Ajusta el balanceo del árbol de acuerdo al caso de desbalance que se encuentre
BinaryTreeTestAVL	testRotateLeftCaseD()	setUpEscenary1		Ajusta el balanceo del árbol de acuerdo al caso de desbalance que se encuentre
BinaryTreeTestAVL	testRotateLeftCaseE()	setUpEscenary1		Ajusta el balanceo del árbol de acuerdo al caso de desbalance que se encuentre

BinaryTreeTestAVL	testRotateLeftCaseF()	setUpEscenary1		Ajusta el balanceo del árbol de acuerdo al caso de desbalance que se encuentre

Clase	Metodo	Escenario	Valores de entrada	Resultado
DataManagement	getListPlayer ()	setUpEscenary1	"michael", "CHI", 1990, 40, 2, 1, 20, 39, 0.4, 20); “cobe", "CHI", 1985, 40, 2, 1, 20, 39, 0.4, 20); "lebron", "CHI", 1995, 40, 2, 1, 20, 39, 0.4, 20); “raul", "CHI", 1993, 40, 2, 1, 20, 39, 0.4, 20); "lebron", "CHI", 1997, 40, 2, 1, 20, 39, 0.4, 20); “james", "CHI", 1990, 40, 2, 1, 20, 39, 0.4, 20);	Los valores agregados correctamente a la lista de jugadores
DataManagement	searchPlayerLinearly()	setUpEscenary1	“lebron”	Correctamente todos los jugadores con el mismo nombre y los adiciona a un arreglo de muestra
DataManagement	linealSearch()	setUpEscenary1	10-30	Rango de jugaores con un numero de bloques dentro del rango seleccionado
DataManagement	createBinaryTrees()	setUpEscenary1		Al crear los arboles binaries se denomina una raiz para cada uno de los arboles

DataManagement	Filter()	setUpEscenary1	Filtrado de minimo 1 y maximo 2	Filtra dependiendo del atributo y organiza dependiendo de el rango