# Proyecto I LinkedDB

Bertha Brenes

2017101604

Algoritmos y Estructuras de Datos 1

Ingenieria en Computadores
Instituto Tecnologico de Costa Rica
Cartago, Costa Rica
2017/08/06

# Índice

**1. Introducción** **2**

**2. Descripción del problema** **3**

**3. Planificación del proyecto** **3**

**4. Diseño Proyecto** **5**
4.1. Arquitectura de solucion . . . . . 5
4.2. Diagrama de componentes . . . . . 6
4.3. Diagrama de secuencias . . . . . 7
4.4. Diagrama de clases . . . . . 8

**5. Implementación** **11**
5.1. Descripcion de librerias . . . . . 11
5.2. Estructura de Datos . . . . . 12
5.3. Algoritmos Desarrollados . . . . . 12
5.4. Problemas Encontrados . . . . . 12

**6. Conclusión** **14**

**7. Bibliografía** **15**

**8. Anexos** **16**

# 1. Introducción

Una base de datos es una biblioteca donde yo puedo almacenar información y luego acceder a ella. Existe dos tipos de bases de datos:

- Segun su contenido
- Segun su variabilidad de datos

Para este trabajo vamos a ver segun su variabilidad de datos que se divide en:

- Estaticos: Se usa normalmente para almacenar datos historicos que no se van a modificar y se utilizaran en un futuro para alguna investigación
- Dinamicos: Son aquellos datos que se van a cambiar con el tiempo.Y por eso usamos este tipo, pues necesitamos operar sobre esos datos.

Ademas de eso, las bases de datos se dividen en el tipo de modelo de datos.

- Modelos jerarquicas
- Modelos en red
- Modelos transaccionales
- Modelos relacionales
- Modelos orientadas a objetos
- Modelos documentales
- Modelos avanzados
- Modelos declarativos

Para este proyecto debemos usar un tipo de modelo no relacional.
Una base de datos no relacional (NoSQL) nace a raiz de la necesidad de sistemas y principalmente de kas redes sociales a utilizar una base de datos que permitiera agregar diferentes tipos de datos que no necesariamente tuvieran relacion entre si, que su volumen de datos fuera alto pero no perdiera su rapidez con lo cual tenemos tres caracteristicas principales que hacen un modelo NoSLQ ideal para cierto tipo de problemas. Los sistemas NoSQL mas usados en este momento son:

- MongoDB
- Cassandra
- Redis
- CouchDB

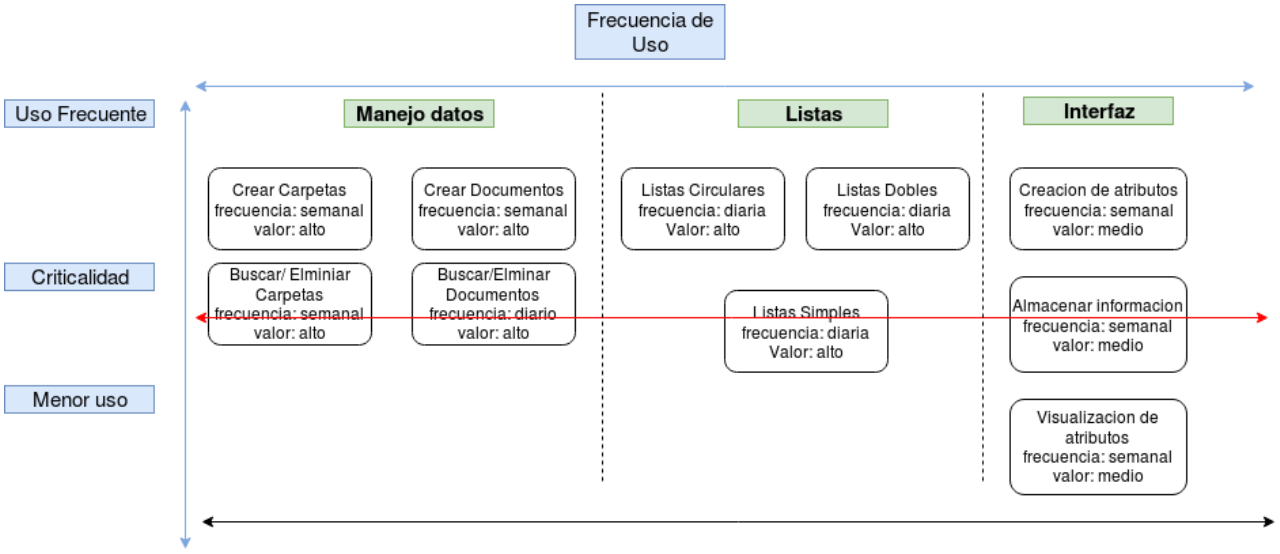
## 2. Descripción del problema

Para este proyecto se nos solicito desarrollar un motor de bases de datos, utilizando documentos json, que yo pueda buscar, almacenar, eliminar y demas. Para esta interfaz se debe tener un lugar donde almacenar informacion es decir carpetas, para cada carpeta se puede crear documentos .json que serviran para almacenar a su vez objetos json. Estos objetos pueden ser atributos de los documentos json, estos atributos tienen que ser guardados por el tipo de atributo que el usuario le asigne, ademas el usuario puede escoger si quiere que los atributos fueran llaves foraneas o primaria.Una llave es una referencia a otro u atributo json con el cual yo pueda enlazarlo.

## 3. Planificación del proyecto

Features identificados:

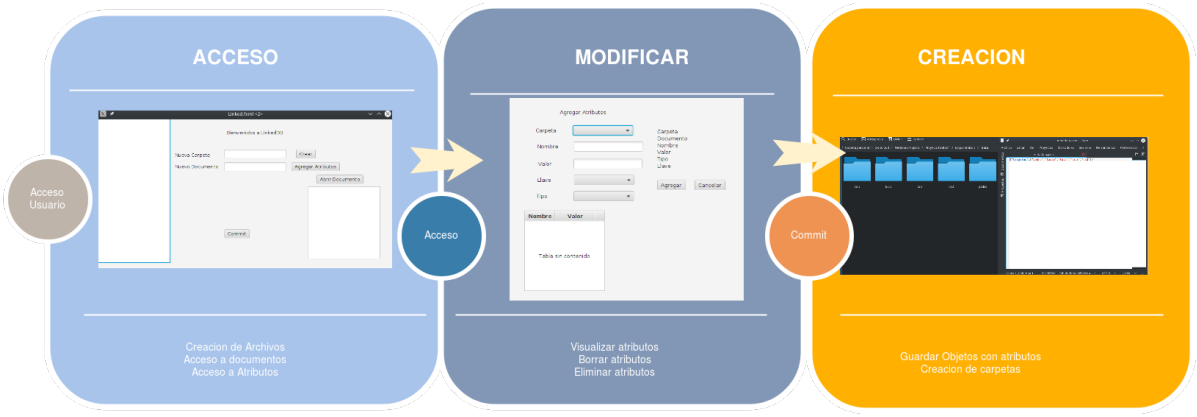
- Listas
- Manejos de Documentos y carpetas
- Manejo de Objetos y Arrays en JSON
- Manejo de JSON para metadata
- Interfaz principal
- Interfaz secundaria
- TreeView



| Fecha | Actividad | Tiempo Invertido |
|---|---|---|
| 7/08/17 | Iniciacion de investigacion y proyecto | 5 |
| 8/08/17 | Pruebas funcionamiento json | 4 |
| 11/08/17 | Creacion Metadata | 5 |
| 27/08/17 | Listas enlazadas | 6 |
| 3/09/17 | listas simples,json y atributos | |
| 7 4/09/17 | listas circulares | 6 |
| 7/09/17 | documentacion y javaDoc | 5 |
| 10/09/17 | Listas doblemente enlazada | 7 |
| 11/09/17 | investigacion interfaz | 7 |
| 14/09/17 | metadatas de objetos | 9 |
| 15/09/17 | Arreglo e investigacion objetos | 24 |
| 16/09/17 | Interfaz y diseño | 9 |
| 17/09/17 | Interfaz y metodos | 10 |

# 4. Diseño Proyecto

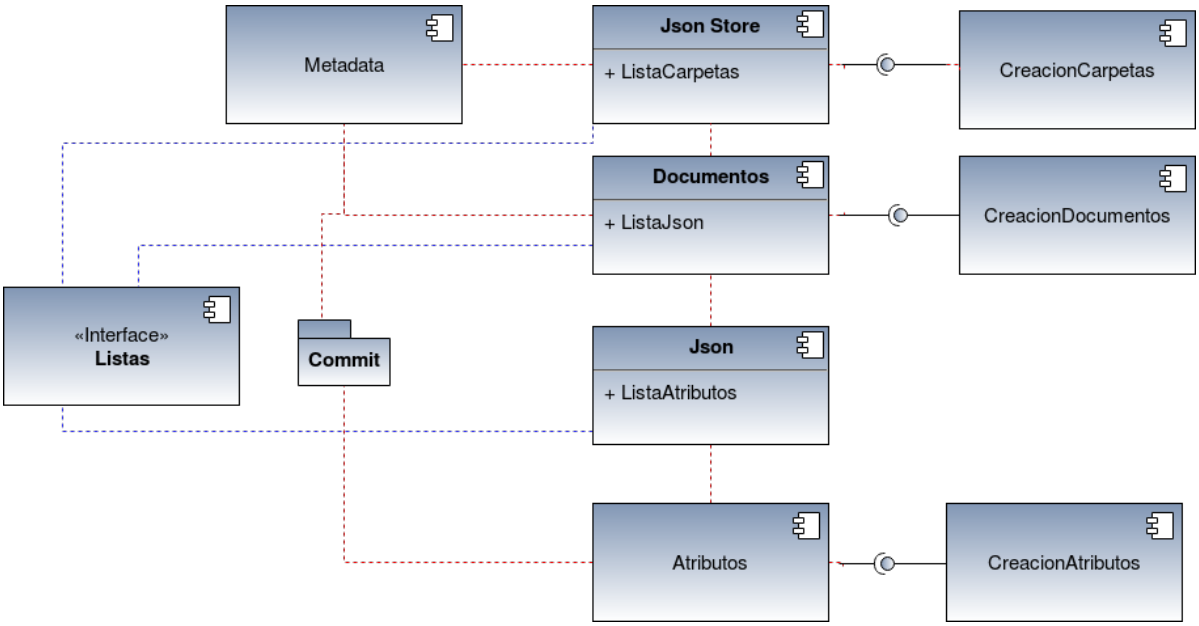
## 4.1. Arquitectura de solucion




Para el diagrama de arquitectura de solución segun la investigación realizada era escoger las 3 cosas que constituyeran la solución al problema. En este caso de una base de datos lo dividi de forma que:

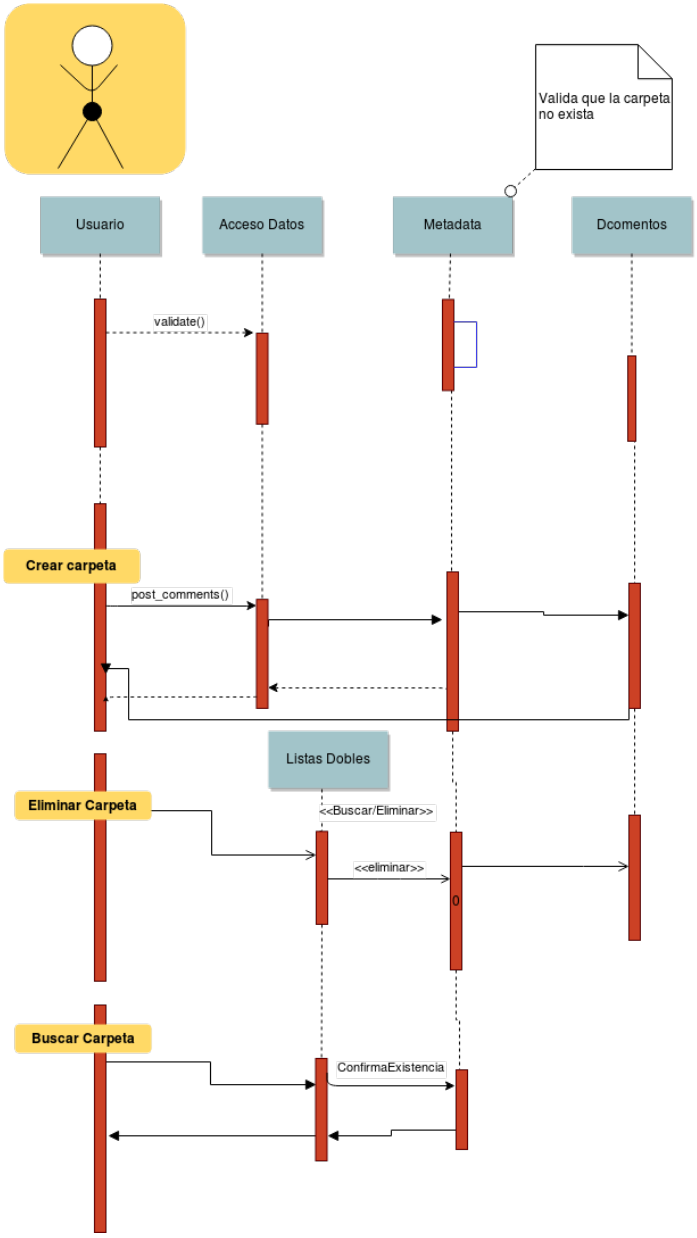
- Primera linea: la division del problema
  - Acceso a los datos
  - Modificar los datos
  - Guardar los objetos
- Central linea: Interfaz con usuario
- Ultima linea: Acciones
  - Acceso a datos: Aqui las principales acciones es la visualizacion y crear nuevas carpetas, documentos y demas.
  - Modificar datos: Aca se van a crear, modificar y visualizar los objetos de cada documento. ademas de acceder a la tabla de atributos por archivo
  - Guardar los objetos: Todas las carpetas y documentos seran creados al momentos solo los atributos se visualizaran en el momento de que se guardan.

## 4.2. Diagrama de componentes




Los componentes principales aparecen en la descripcion de manera que estos contenian las listas de forma empaquetada pero a su vez pudiera funcionar, los primeros enlaces son las creadores de botones y demas que proveen interfaz y los componentes usan esas interfaces

## 4.3. Diagrama de secuencias

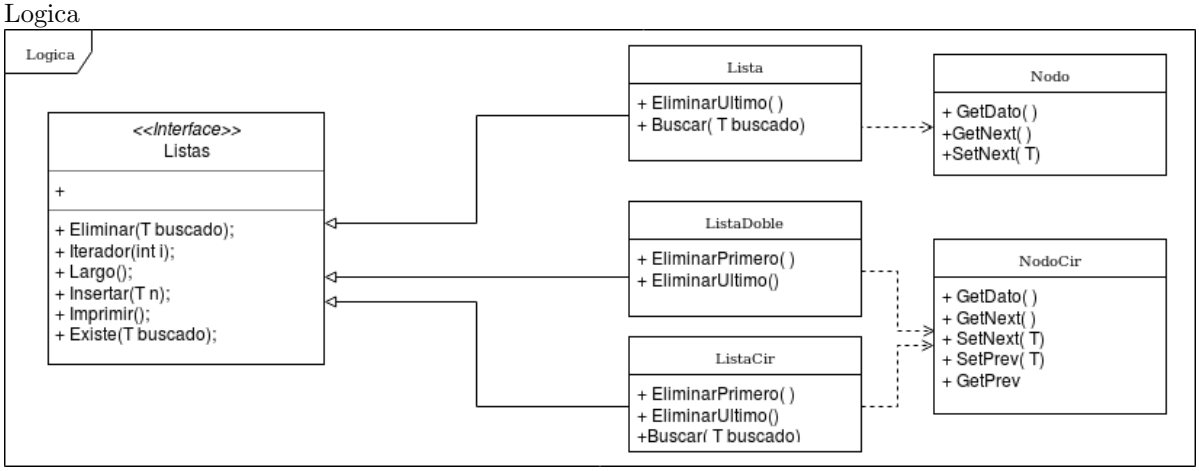



Para el diagrama de secuencias se utilizo las compilaciones principales y funcionales en el programa, la creación de archivos nornales, donde el programa verifica que el archivos que se vaya a crear no exista para no repetirlo y ademas agrega el nodo a lista. Para las busqueda y eliminacion de archivos, se trabaja todo desde las listas y se confirma en la metadata y documentos

## 4.4. Diagrama de clases

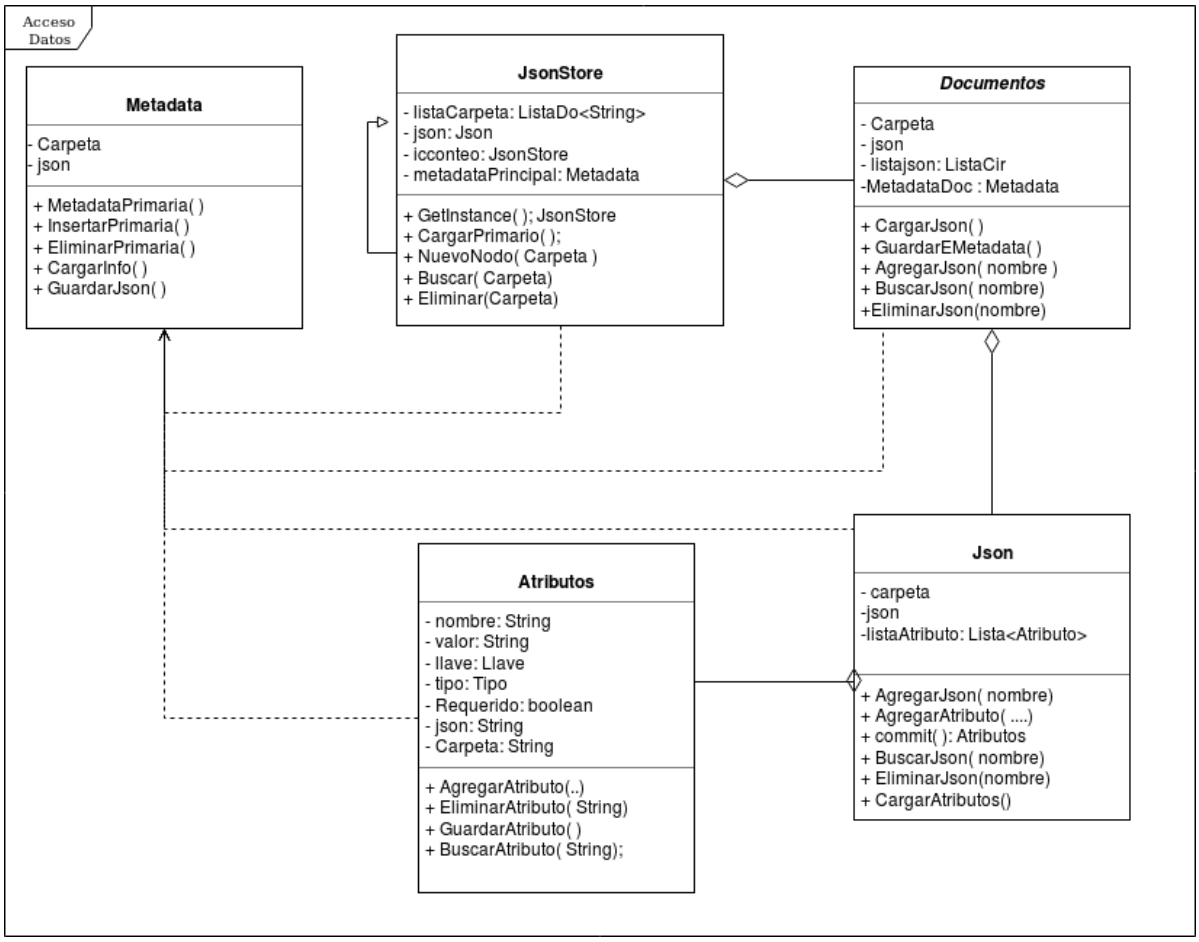
Para esta parte lo decidi dividir en 3 grupos, que se puede visualizar aca

- Logica



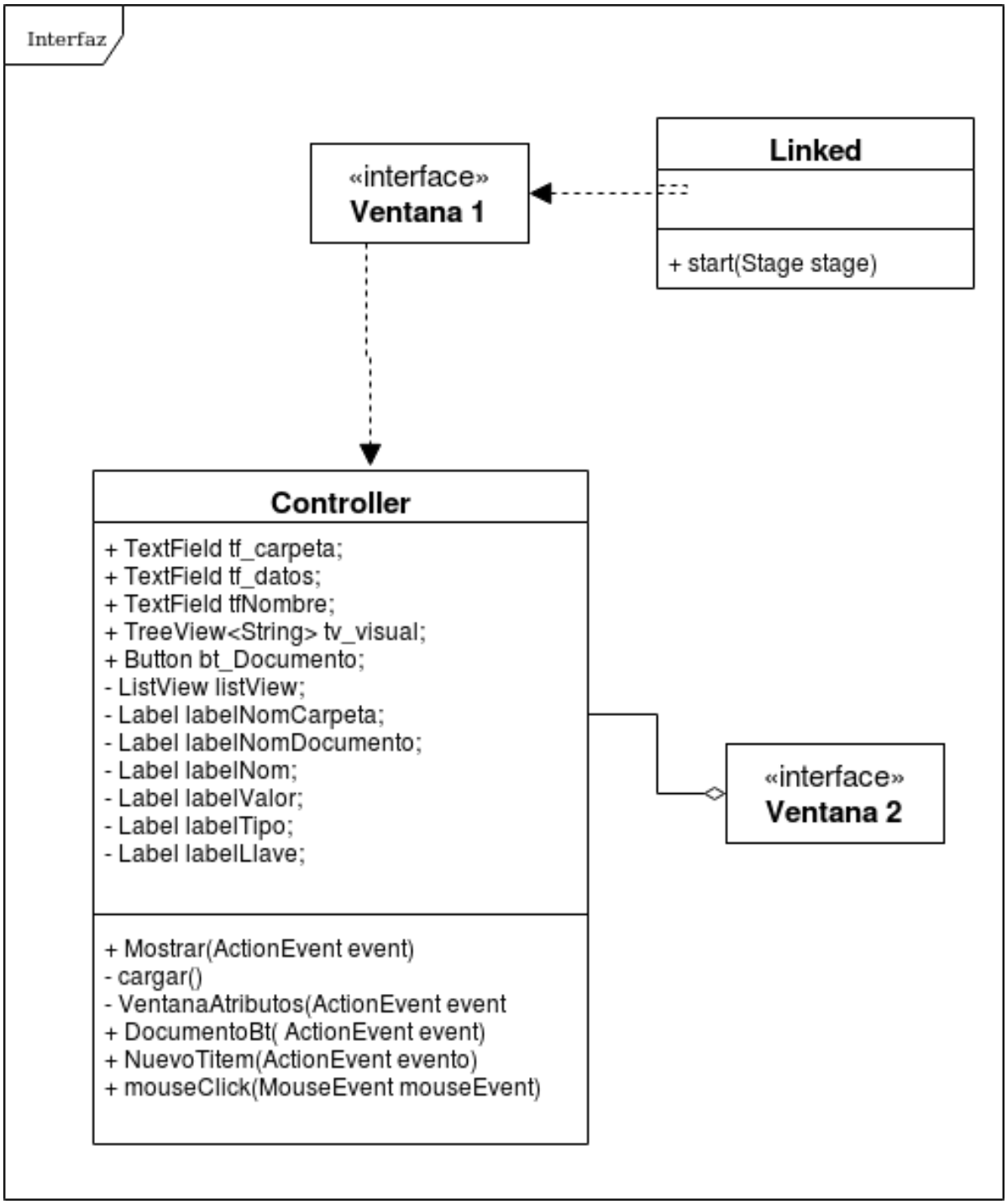

Para este sistema se dejo como logica el maenjo de las listas

- Acceso a datos




Para el accesos todo el manejo de las carpetas y sus creciones

- Interfaz




para la interfaz como se trabaja con scene builder no quise trabajar nin-

guna clase externa a las previstas y trabajar eso si sobre lo trabajado.

# 5. Implementación

## 5.1. Descripcion de librerias

Las librerias utilizadas en este proyecto fueron

- java util
  Se utilizaron:
  - Iterator : Se utilizo para iterar en los elementos de los documentos json y asi cargar las listas al iniciar los programas
  - logging.Level
  - logging.Logger : Este y el logging. Level se utilizo para imprimir errores.
- java io
  Se utilizaron:
  - File : Para lograr crear o abrir archivos se utilizo esta libreria
  - FileReader : Se utilizo para la lectura de archivos
  - IOException : Se utilizo en los try/catch por si el codigo generaba un error al tratar de ejecur el try
  - FileWriter : Se utilizo para la lectura de archivos
- json simple
  Se utilizaron:
  - JSONArray : La libreria que mas se utilizo para lograr crear arrays tipo json en las metadata y documentos
  - JSONObject: Esta libreria tambien se uso mucho pues permitia crear objetos json.
  - Parser: Esta libreria permitia parsear la información en los json y obtener los datos ademas tenia su propio exception por si el código generaba errores al parsear
- javafx
  Se utilizaron:
  - Application: Permite cargar el programa e inicializar el archivo .fxml
  - FXML: Esta es la pagina encargada de contener solamente los elementos que se van a visualizar. y ademas permite ser abiertos en otra aplicacion aparta para su diseño

- scene: Es una clase que contiene más clases para los graficos que se van a mostrar, ademas de que proporciona clases para poder programar los elementos de las FXML
- Stage: Permite darle crear y darle atributos a la ventana propiamente como setear el titulo y los anchos.
- Event: Se utiliza cuando tengo algun boton o label que necesito generarle accion al ser presionados.

## 5.2. Estructura de Datos

La estructura de datos desarrolladas en este proyecto fueron las listas. Se desarrollaron:

- Listas Doblemente enlazadas:
  Estas se utilizaron para almacenar los jsonStore es decir las carpetas creadas, cada vez que creaba un nuevo nodo, se creaba una nueva nueva carpeta lo cual inicializaba una lista circular de documentos de forma que quedaran enlazadas, a la hora de borrar las carpetas tambien se borraban los elementos en la listas.

- Listas Circulares:
  Estas se utilizaron para almacenar la informacion de los documentos existentes es decir archivos json, que contendrian los objetos. Al igual que todas la lista circular al crear un nuevo documento se inicializar la lista de los objetos que va a contener ese documento

- Listas simples:
  Si lo vemos jerarquicamente las listas simples son las raices de nuestra estructura de listas pues despues de ella no hay otras estructuras, las listas simples son inicializadas por varios nodos pero ellas no inicializan a ningun nodo. Las listas simples almacenan los objetos que van a contener los documentos, estos objetos son atributos.

## 5.3. Algoritmos Desarrollados

Se tiene una clase que almacena toda la informacion que se va creando en la metadata, la unica informacion que no se almacena son los atributos hasta que se le de commit al programa. En cuando a las carpeta se tiene una metadata que guarda las carpetas que se van creando en la metadata y al mismo tiempo las almacena en una lista doble, tambien se crean las metadatas secundarias que son metadatas que se crean por store y almacenan los documentos que se vayan creando y a su vez los objetos que el usuario guarda.

## 5.4. Problemas Encontrados

Los problemas encontrados resueltos con exito fueron a la hora de realizar las listas, pues el tema de eliminar un dato especifica de la lista se me compli-

co bastante y lo que hice fue reutilizar codigo de metodos que ya tenia como eliminar primero o ultimo, lo que reduzo la cantidad de codigo y soluciono el problema.
Otro problema fue trabajar la metadata y que esta se actualizara si yo borraba un elemento, esto lo hice enlazando directamente las listas a la metadata.
Otro problema que tuve fue que si yo intentaba escribir datos en los documentos propiamente estos borraban todo lo que habia en los documentos.
Problemas encontrados sin exitos: Sinceramente todo lo que fue respecto a la interfaz fue una interminable sesion de encontrar un bug, solucionarlo y encontrar otro y asi sucesivamente.
El primer problema fue con intentar actualizar la informacion y que el treeview mostrara las carpetas existentes y las nuevas, esto no lo logre hacer.
Otro problema y mas grande es que no logre acceder a la informacion de los textfiel aun me dice que estos estan apuntando a nullpointer que sigo sin solucionar pues si le cambio su nombre de id, el programa deja de funcionar.
He tenido muchos problemas con respecto a si le agrego un nombre id mal al componente, el programa por ende tampoco carga la pantalla incial.
Esto certeramente no hubiera pasado si hubiera trabajado la interfaz y la logica al mismo tiempo y no como lo hice que primero trabaje toda la logica y de ultimo trabaje la interfaz

# 6. Conclusión

En la cotidianidad de los programadores tiene que enfrentarse al uso de las bases de datos para lograr administrar cientos de miles de datos y trabajar con ellas, esta labor no puede recaer por entero y desde 0 en el programador por lo que vimos estas bases datos que nos ayuda a trabajar los datos y procesarlos. Los sistemas que vimos de bases de datos, maneja la informacion en memoria pero esta puede ser desechadas hasta que yo le indique que escriba en memoria. A raiz de la investigación realizada para este proyecto termino de comprender los conceptos del funcionamiento de una base de datos, sus tipos y su utilidad; asi como los modelos caracteristicos. Ademas lograr entender los conceptos de abstraccion y el sistema OOP que tiene Java, pues viene siendo bastante distinto entender el paradigma de programacion orientada a Objetos en Python que en Java que es un paradigma especializado par a OOP.
Queda mucho por aprender con respecto a la interfaz pues como se expresa anteriormente el error más grande fue trabajar la logica y la interfaz por separado.

# 7. Bibliografía

## Referencias


[1] Visual Studio, *Diagramas de secuencia UML* https://msdn.microsoft.com/es-es/library/dd409377.aspx .2015

[2] Aprender a programar, *Paquete java.util* https://www.aprenderaprogramar.com/

[3] Oracle Documentation, *All classes* https://docs.oracle.com/javase/7/docs/api/allclasses-noframe.html

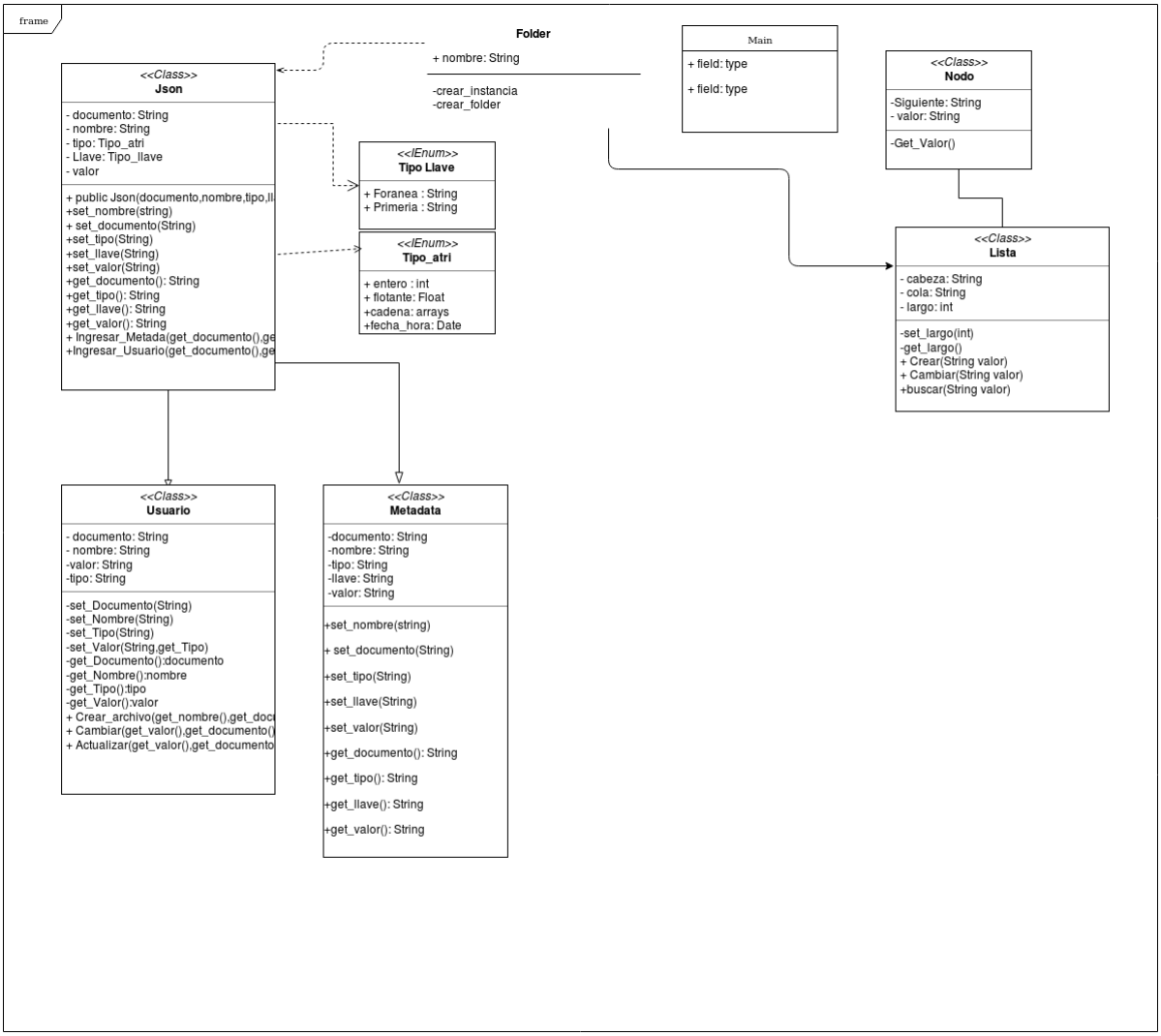
[4] Software Development Tutorials, *JavaFX with Scene Builder tutorials* https://youtube.com

[5] Slideshare , *Diagrama de componentes* Cruz Victor https://es.slideshare.net/uitron/diagrama-de-componentes-7551535

[6] Silicon , *Bases de Datos No Relacionales (NoSQL)* José María Baquero 13 diciembre,2016 http://www.silicon.es/

[7] Cloud Computing, *MongoDB* Andrés Purriños 28 Mayo, 2014 http://www.teknlife.com/reportaje/mongodb-son-las-bases-de-datos-relacionales-el-futuro/

# 8. Anexos




mi primer diagrama de clases para desarrollar este proyecto