Estudiante:

Gerald Valverde Mc kenzie.

(2017112234)

Curso:

Algoritmos y Estructuras de Datos I.

(CE1103)

Profesor:

Luis Diego Noguera Mena.

Proyecto I: LinkedDB.

Fecha de entrega:

20/09/17

Grupo: 02

Tabla de contenido

CAPITULO 1:
INTRODUCCION.

1.1 Antecedentes del problema

Hoy en día, las bases de datos son uno de los “activos intangibles con mayor valor para la compañía**”** (Redacción KYOCERA, 2017), ya que estas poseen la información vital de cada uno de los procesos que se llevan a cabo en la empresa; como por ejemplo las ventas, clientes, rentabilidades, inventario, nombre e información de los empleados y proveedores. Haciendo así más fácil los procesos mejorando la eficacia y rapidez con que se realiza las tareas, además ofrece mayor seguridad a la hora de manejar información confidencial y mayor facilidad en cuanto a la repetibilidad de datos, ya que otras personas pueden acceder a ellas. Las bases de datos brindan una interfaz unificada, centralizada y organizada de datos, que facilitan, la modificación, inserción y eliminación de datos. Provocando así que los procesos sean más rápidos, y la visualización de información.

1.2 Planteamiento del problema

¿Cómo implementar una base de datos noSQL, por medio del lenguaje Java, creando una interfaz gráfica amigable con el usuario, y además que le permita realizar las operaciones solicitadas en las historias de usuarios?

1.3 Importancia

En las empresas el uso de las bases de datos es esencial ya que optimiza totalmente las operaciones, es de suma importancia ya que en cada una de las instancias se pueden crear tablas, con valores en los que cada valor está enlazado a otro valor único y así mismo se les relacionan a datos de otras tablas, permitiendo una mejor organización de los datos (Filella, 2017).

CAPITULO 2:
MARCO TEORICO.

2.1 Planificación y administración del proyecto

Para llevar con éxito el proyecto se realizó una etapa de planificación en la que se recopilaron varias técnicas para la división de trabajo y el entendimiento de problema planteado.

2.1.1 Historias de usuario.

A continuación, se observa las historias de usuario, las cuales son las especificaciones sobre las cosas que el usuario desea que el programa contenga. Los cuales poseen una estructura de tipo de usuario, feature que necesita y beneficio que recibe.

- Yo como usuario, necesito una base de Datos noSQL, que me permita definir, insertar, eliminar, actualizar y buscar documentos JSON, que me facilite la visualización y almacenaje de archivos JSON.
- Yo como gestor de la base de datos, deseo que se muestre un árbol de JSON Stores, para lograr visualizar la jerarquía y cada uno de los datos almacenados.
- Yo como usuario, necesito crear documentos dentro de cada Store con su nombre y una lista de atributos.
- Yo como usuario, necesito crear carpetas físicas, para poder agrupar los documentos.
- Yo como gestor del motor de bases de datos, necesito que cada atributo tenga características específicas para poder definir cada objeto.
- Yo como usuario de la Base de Datos, necesito un botón de commit, que me permita guardar los cambios realizados.
- Yo como gestor de la base de datos, necesito que cada vez que abra el programa cargue automáticamente los archivos guardados.
- Yo como usuario, deseo poder crear objetos que se relacionen entre si y reciban las características del atributo empleado, para crear una relación en los datos.
- Yo como gestor de la base de datos, necesito que la interfaz sea similar a la de Workbench, DataGrip o MySQL, para facilitar el uso y la navegación.
- Yo como usuario, necesito que cuando haga un nuevo documento, se cree un archivo formato JSON con el nombre del documento dentro de la carpeta correspondiente al store seleccionado.

2.1.2 Lista de features

Los features son cada una de las características que el usuario desea que se le implemente a su proyecto. Para identificar las características se debe contestar la siguiente pregunta*: "¿Qué es lo que el software va hacer?".* Usualmente en las historias de usuarios la oración que lleve un verbo de acción indica una característica. A demás, se dice que una buena característica es toda aquella que este centrada en la perspectiva del usuario (Patton, 2005).

Dentro de los features que el usuario solicitó a la entrega del proyecto se encuentran:

- Crear Stores.
- Crear carpetas físicas.
- Crear documentos.
- Asignar atributos.
- Crear objetos.
- Validar objetos.
- Guardar datos.
- Visualizar el motor de base de datos.
- Manipular Stores.
- Mostrar árbol de JSON Stores.
- Manipular documentos
- Manipular objetos.
- Cargar archivos.

2.1.3 Distribución de historias de usuario por criticalidad y secuencia de uso

Se debe agrupar cada uno de los features con sus respectivos detalles, por su criticalidad, tomando como parámetros que tan importante es realizar una característica. Para esto se acomodan verticalmente dependiendo de la frecuencia de uso de este, en donde la más alta son las que se usan siempre y conforme se baja las menos usadas. A demás se organizan de izquierda a derecha dependiendo de la secuencia en el que se va a emplear cada feature.

2.1.4 Minimal System Span

El Minimal System Span es un conjunto de features, que se agrupan en conjuntos lógicos, que dividen los procesos horizontalmente de principio a fin. Este conjunto de feature consisten en solo las características necesarias para crear la mínima versión de software funcional con los requerimientos más simples. Este sirve para demostrar al cliente una previa al avance del producto. En este proyecto como features necesarios para realizar la primera versión de software funcional se encuentran:

- Crear Stores.
- Crear documentos.
- Crear objetos.
- Validar objetos.
- Guardar archivos.

2.1.5 Asignación de user stories por miembro del equipo

Las user histories para llevar a cabo el proyecto se han divido en tres partes equitativas para cada uno de los integrantes del equipo.

Al **integrante A** del proyecto le corresponde:

Crear Stores:

- Crear una lista doblemente enlazada
- Se guarda el Store en la lista enlazada.

Crear Carpetas:

- Se crea una carpeta física en una ruta predefinida.

Cargar datos:

- Leer las carpetas físicas e insertarlas en la lista enlazada simple.

El **integrante B** del equipo debe:

Crear lista enlazada simple.

Objetos:

- Se guarda en una lista enlazada simple.
- Al crear validar que tenga los atributos correctos.
- Eliminar todos los objetos.

- Buscar objetos por atributos.
- Actualizar objetos.

Cargar datos:

- Crear método que lea archivos Json para luego insertarlos en listas.

El **integrante C del** proyecto debe realizar las histories relacionadas a:

Crear lista circular doblemente enlazada.

Crear Documentos:

- Crear un archivo JSON con el nombre del documento.
- Se guarda el store en una lista Circular doblemente enlazada.
- Crear una lista de atributos, con sus respectivas características: nombre, tipo, llave, Requerido o no, valor por defecto.

Manipular documentos:

- Realizar operaciones como: insertar, eliminar, actualizar y buscar.

Finalmente, se realiza en conjunto:

Visualizador del motor de DB:

- Desarrollar interfaz en JavaFX.
- Crear botón commit para guardar todos los cambios realizados. **(Integrante A)**

Mostrar Árbol:

- Mostrar los datos de la lista enlazada doblemente enlazada. **(Integrante A)**
- Mostrar los datos de la lista Doble circular. **(Integrante C)**
- Mostrar un listado con todos los objetos. **(Integrante B)**

2.1.6 Descomposición de cada user story en tareas

Las tareas son cada una de las acciones o puntos a realizar para poder llevar a cabo los features de cada una de las historias de usuario. Enseguida se desglosará de cada user hisory las tareas que se deben ejecutar.

1. **User history:** Yo como usuario, necesito una base de Datos noSQL, que me permita definir, insertar, eliminar, actualizar y buscar documentos JSON, que me facilite la visualización y almacenaje de archivos JSON.
   Tareas:
   - Crear documentos.
   - Insertar documentos.
   - Eliminar documentos.
   - Actualizar documentos.
   - Buscar documentos.

2. **User history:** Yo como gestor de la base de datos, deseo que se muestre un árbol de JSON Stores, para lograr visualizar la jerarquía y cada uno de los datos almacenados.
   Tareas:
   - Mostrar árbol en la interfaz gráfica.

3. **User history:** Yo como usuario, necesito crear documentos dentro de cada Store con su nombre y una lista de atributos.
   Tareas:
   - Crear Stores.
   - Asignarle un nombre al store.
   - Asignarle una lista de atributos al store

4. **User history:** Yo como usuario, necesito crear carpetas físicas, para poder agrupar los documentos.
   Tareas:
   - Crear carpetas físicas en una ruta predeterminada.
5. **User history:** Yo como gestor del motor de bases de datos, necesito que cada atributo tenga características específicas para poder definir cada objeto.
   Tareas:
   - Definir el nombre del atributo, el tipo, tipo de llave, si es requerido o no y su valor por defecto.

6. **User history:** Yo como usuario de la Base de Datos, necesito un botón de commit, que me permita guardar los cambios realizados.
   Tareas:
   - Crear un botón en la interfaz gráfica.
   - Crear método para guardar archivos en memoria física.

7. **User history:** Yo como gestor de la base de datos, necesito que cada vez que abra el programa cargue automáticamente los archivos guardados.
   Tareas:

- Crear método que recorra, y agrupe las carpetas y archivos json para luego cargarlos a las listas enlazadas.

8. **User history:** Yo como usuario, deseo poder crear objetos que se relacionen entre si y reciban las características del atributo empleado, para crear una relación en los datos.
   Tareas:
   - Relacionar cada objeto que sea de atributo foráneo.

9. **User history:** Yo como gestor de la base de datos, necesito que la interfaz sea similar a la de Workbench, DataGrip o MySQL, para facilitar el uso y la navegación.
   Tareas:
   - Implementar una base de datos que sea agradable para el usuario.

10. **User history:** Yo como usuario, necesito que cuando haga un nuevo documento, se cree un archivo formato JSON con el nombre del documento dentro de la carpeta correspondiente al store seleccionado.
    Tareas:
    - Crear documentos tipo json dentro de una carpeta específica.

2.2 Diseño

2.2.1 Diagrama de arquitectura de la solución

En este proyecto se utilizó el patrón de diseño MVC, el cual consiste en agrupar el proyecto en tres partes, guardar archivos. (Ver ilustración 2).

2.2.2 Diagrama de componentes

El diagrama de componentes muestra la relación de los objetos (Ver ilustración 3).

2.2.3 Diagrama de secuencia

El diagrama de secuencia sirve para ver el tiempo y la ejecución de algunos request del usuario al sistema(Ver ilustración 4).

2.2.4 Diagrama de clases

El diagrama de clase sirve para entender el problema y da una posible solución (Ver ilustración 5).

2.3 Implementación

2.3.1 Descripción de las bibliotecas utilizadas

Para llevar a cabo este proyecto se necesitó emplear varias librerías de Java, a continuación, se detallará el uso de estas y su funcionamiento.

2.3.1.1 LocalDateTime

Esta librería se encuentra en el paquete java.time.LocalDateTime la cual es una clase que facilita la manipulación de las fechas y horas sin importar la zona horaria, esta se encuentra basado en el sistema de calendario ISO. Posee métodos como el “from” que permite convertir la instancia de otro formato a LocalDateTime, y métodos “of” que permiten horas, minutos, días, semanas y meses (Silva, 2017) .En seguida se muestra cómo crear una instancia de esta clase:

```
LocalDateTime date = LocalDateTime.of(year, month,day,hour,min);
```

2.3.1.2 Simple Json 1.1

Esta librería se encuentra en el paquete org.json.simple permite la manipulación de datos tipo JSON; la clase JSONObject nos permite crear los archivos, la JSONParser la lectura de este tipo de archivos, y la JSONArray nos facilita la creación de arreglos de datos Json.

2.3.1.2 File

Se encuentra en java.io.File, esta nos permite la búsqueda y manipulación de los archivos del sistema. Se usa un directorio para definir donde se van a realizar las operaciones.

2.3.2 Descripción de las estructuras de datos desarrolladas

Para la elaboración del proyecto se emplearon listas enlazadas como estructura de datos.Las listas enlazadas son las ideales para este proyecto ya que se desconoce la cantidad de objetos, documentos y stores que el usuario va implementar en su base de datos. Específicamente se utilizaron listas enlazadas simple, doblemente enlazada y circular doblemente enlazada. Cada una de ellas, tiene sus respectivos métodos de inserción, eliminado, búsqueda, verificación y actualización de datos.

2.3.3 Descripción detallada de los algoritmos desarrollados

2.3.4 Problemas encontrados

Durante la elaboración del código se presentaron muchos problemas de incompatibilidad de tipo de datos, para arreglar estos problemas se empleó casteo a las clases, sin embargo, esto no era práctico, y no siempre funcionaba, por lo que se optó por la implementación de datos de tipo Generic.

Otro problema presentado fue errores a la hora de cargar los archivos, para esto se implementaron array que permitión saber cuáles llaves se encontraban en cada uno de los documentos JSON así se optimizó la búsqueda de datos.

CAPITULO 3:
CONCLUSION.

Al finalizar este proyecto se concluyó:

El uso de datos lineales dinámicos, son importantes en la manipulación de archivos, y facilitan la búsqueda de datos.

La implementación de patrones de diseño, facilita la manera en que las clases se relacionan y en cómo el usuario interactúa con el código.

Una buena planificación del proyecto, evita pérdidas de tiempo, y mejora el flujo en que se realizan las tareas.

CAPITULO 4:
ANEXOS.

4.1 Tabla de anexos

4.2 Diagrama de Planificación de proyecto

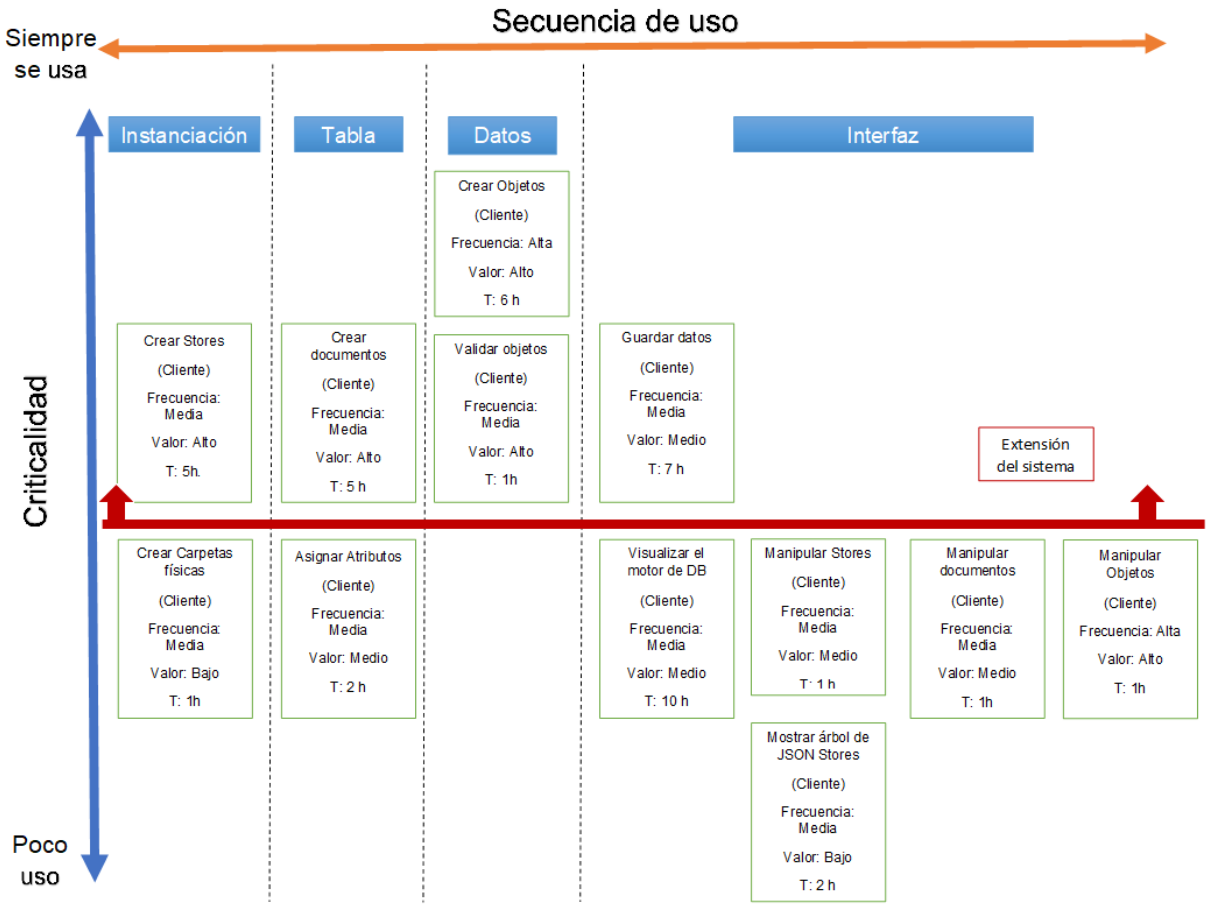

Ilustración 1 Diagrama Criticalidad vs Secuencia de uso

4.3 Diagrama de arquitectura

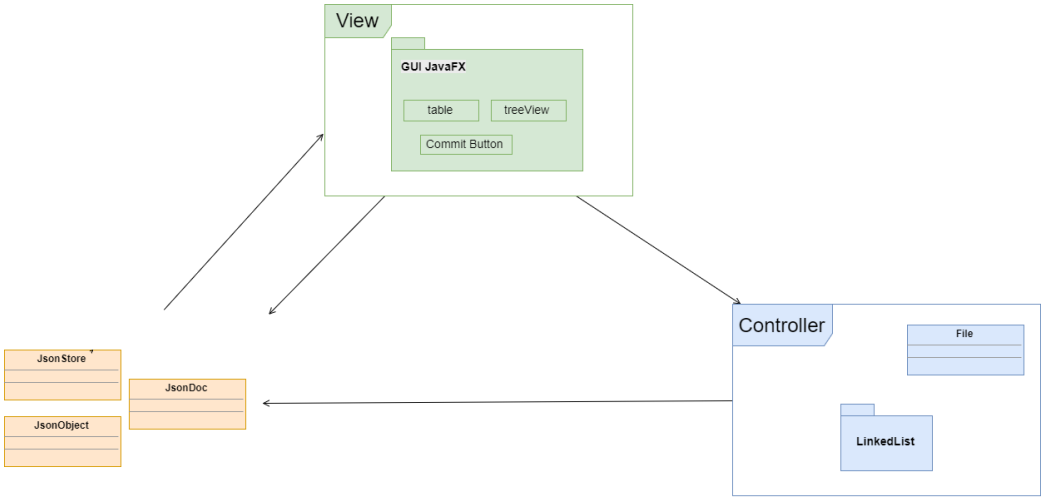

Ilustración 2 Diagrama MVC

4.4 Diagrama de componentes

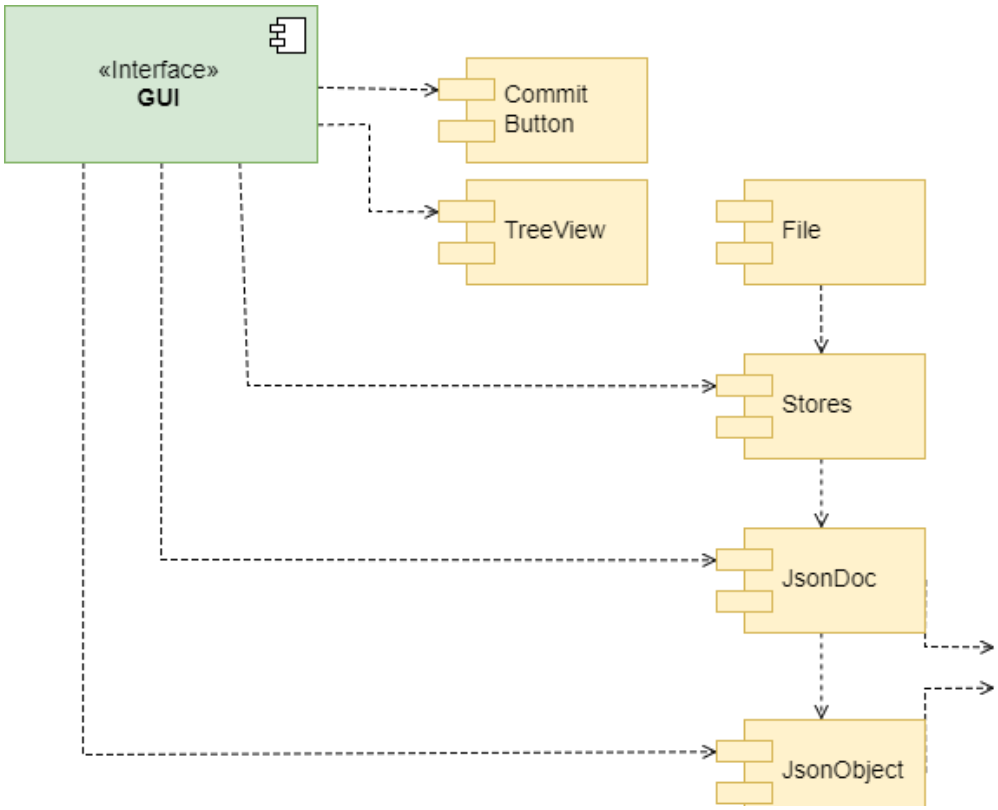

Ilustración 3 Diagrama de componentes

4.5 Diagrama de secuencia

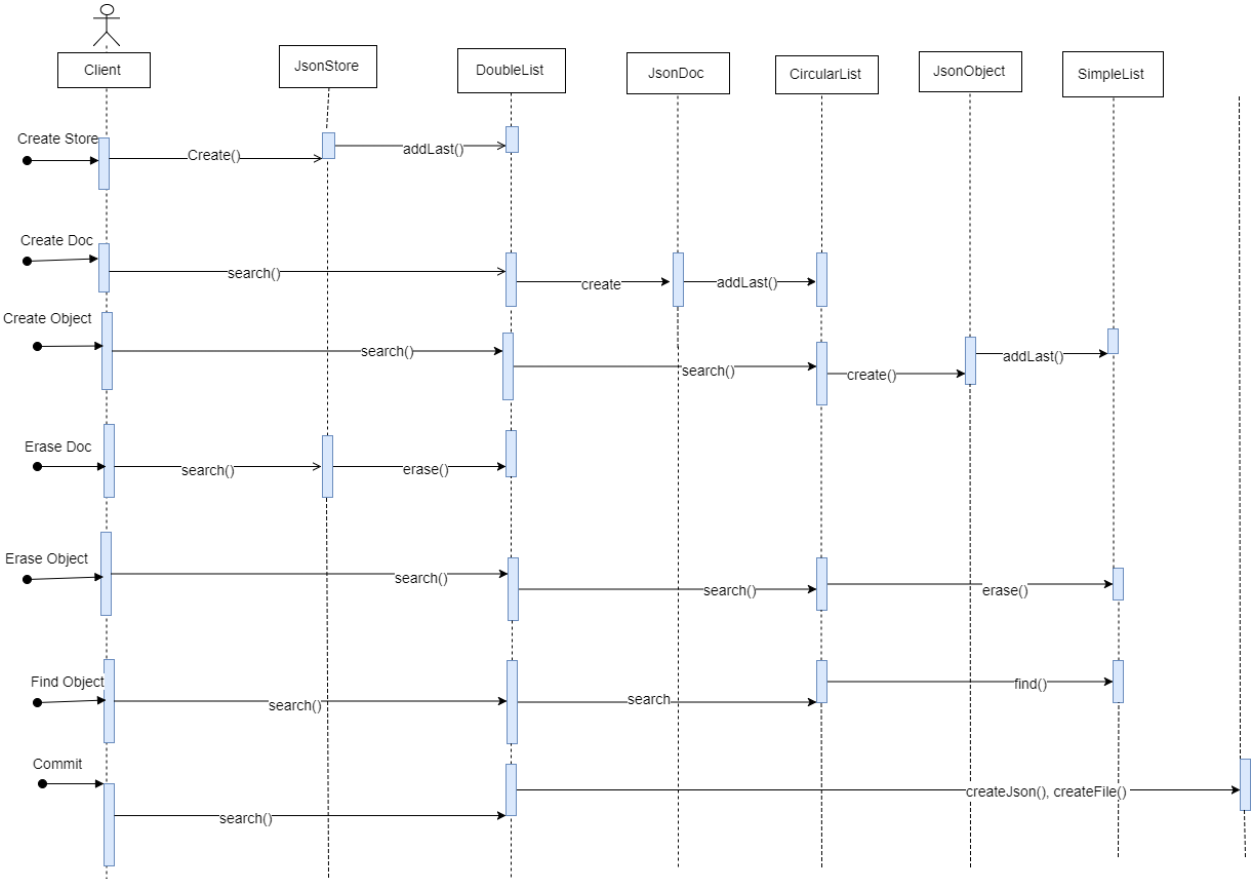

Ilustración 4 Diagrama de secuencia

4.6 Diagrama de clases

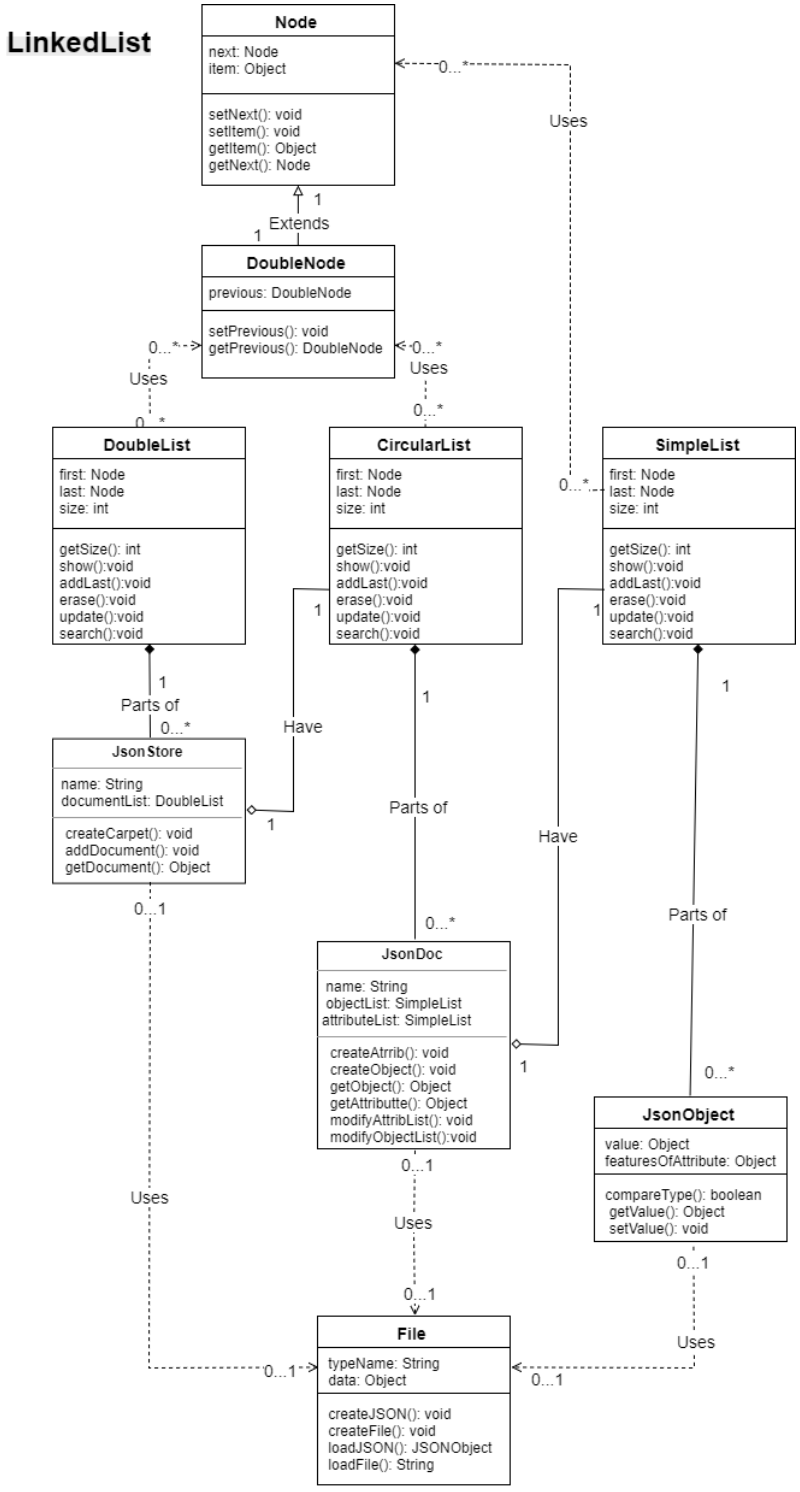

Ilustración 5 Diagrama de clases

CAPITULO 6:

BIBLIOGRAFIA.

5.Bibliografía

Filella, A. P. (11 de Mayo de 2017). *Linkedin*. Obtenido de https://es.linkedin.com/pulse/bases-de-datos-y-su-importancia-en-las-empresas-prieto-filella

Martínez, A. S. (14 de Octubre de 2015). *Data Centric.* Obtenido de http://www.datacentric.es/blog/bases-datos/importancia-bases-de-datos-2/

Microsoft. (02 de Setiembre de 2017). *Support Officce*. Obtenido de https://support.office.com/es-es/article/Conceptos-b%C3%A1sicos-sobre-bases-de-datos-a849ac16-07c7-4a31-9948-3c8c94a7c204

Patton, J. (Enero de 2005). *Sticky Minds.* Obtenido de www.stickyminds.com

Redacción KYOCERA. (05 de enero de 2017). *KYOCERA.* Obtenido de http://smarterworkspaces.kyocera.es/blog/importancia-gestion-de-base-de-datos/

Silva, D. (Diciembre de 2017). *Oracle*. Obtenido de http://www.oracle.com/technetwork/es/articles/java/paquete-java-time-2390472-esa.html