Instituto Tecnológico de Costa Rica

Algoritmos y Estructuras de Datos I

Proyecto #1

LinkedDB

Prof. Antonio Gonzales Torres

Estudiante: Eduardo Quiroga Alfaro

2017079534

Fecha de Entrega: 19 de Setiembre del 2017

Índice

Introducción

LinkedDB es un motor de bases de datos que almacena archivos “.JSON” y entre sus funciones tiene crear, eliminar y actualizar archivos existentes. Esta aplicación está programada en Java para un acceso más universal, y que el cliente pueda utilizarlo en cualquier sistema que se proponga.

La aplicación tiene una interfaz gráfica donde se muestran los datos de los archivos que se selecionan, y los detalles de estos. Dentro de los archivos “.JSON” hay objetos JSON, que almacenan los datos de manera similar a los archivos “.txt”.

El almacenamiento de datos se divide en tres diferentes estructuras, las cuales a la vez son estructuras de listas diferentes. La aplicación crea Stores, dentro de estos existen documentos. Dentro de los documentos se almacenan objetos JSON que tienen diferentes atributos asignables. Considerando esto, a su vez se debe considerar que los Stores son parte de listas dobles enlazadas, los documentos son parte de una lista circular y los objetos JSON son parte de una lista simple enlazada. Así que, los Stores, y los documentos son nodos y listas a la vez, mientras que los objetos son simplemente nodos de los documentos.

El paradigma de programación aplicado es la programación orientada a objetos, que presenta muchísimas facilidades, como la modularidad, la herencia de clases, o el polimorfismo, entre muchas otras. Todo esto nos permite tener un desarrollo del proyecto que no se podría alcanzar de la misma manera con otro paradigma de programación.

1. Descripción del problema

LinkedDB, al tener las estructuras de almacenamiento de datos que se plantean, presenta problemas simples, pero recurrentes. Entre estos, el primero que se considera a la hora del desarrollo es cómo se van a cargar los datos a la memoria dinámica cuando se inicie la aplicación luego de ya haberla utilizado una o varias veces. Otro problema es cómo convertir archivos y carpetas en estructuras de listas, y como señalar detalles o atributos como los punteros y el "root" de las listas.

Durante el desarrollo se presentaron problemas con el manejo de los archivos JSON y con la manipulación de estos a nivel de código. Muchas veces se encontraban conflictos a la hora de interpretar los archivos y tratar de manipular los tipos de datos para el manejo de estos. La biblioteca SimpleJSON fue la implementada para esto, y a pesar de tener pocas opciones para desarrollo, comparada con otras bibliotecas, es muy fácil de implementar, y simple de manejar.

Finalmente, para el desarrollo de la interfaz gráfica se utilizó la aplicación Scene Builder, y esta presentó un problema con el controlador al principió del desarrollo, por lo que no detectaba los metodos ni los atributos necesarios para configurar los diferentes detalles de la interfaz gráfica.

2. Planificación y administración del proyecto

Features e historias de Usuario

Entre los diferentes features que tiene el proyecto, se incluyen:

- Crear Stores, documentos, y objetos JSON
- Eliminar Stores, documentos, y objetos JSON
- Modificar Stores, documentos, y objetos JSON
- Mostrar Stores, documentos, y objetos JSON mediante una interfaz grafica
- Ligar los diferentes objetos mediante llaves foraneas o primarias

Historias de usuario:

- Como usuario de la aplicación, sería util que la interfaz gráfica tuviera un diseño simple para facilitar el uso de la misma
- Como usuario de la aplicación, sería util que esta tenga un apartado, o una descripción de las funciones que puedo utilizar.
- Como usuario de la aplicación, me gustaría que el almacenamiento de los archivos sea fácil de accesar
- Como developer/encargado de mantenimiento de la aplicación, sería util que los archivos de configuración sean fáciles de accesar
- Como developer/encargado de mantenimiento de la aplicación, sería util que el código de la aplicación tuviera un orden simple para que sea fácil interpretarlo
- Como developer/encargado de mantenimiento de la aplicación, sería util que el código estuviera comentado de manera util para que sea fácil interpretarlo

Distribución de historias de usuario por criticalidad

Se presentan las siguientes historias de usuario como punto de partida para el desarrollo de la aplicación:

- Como usuario de la aplicación, sería util que la interfaz gráfica tuviera un diseño simple para facilitar el uso de la misma
- Como usuario de la aplicación, sería util que esta tenga un apartado, o una descripción de las funciones que puedo utilizar.
- Como usuario de la aplicación, me gustaría que el almacenamiento de los archivos sea fácil de accesar
- Como developer/encargado de mantenimiento de la aplicación, sería util que los archivos de configuración sean fáciles de accesar
- Como developer/encargado de mantenimiento de la aplicación, sería util que el código de la aplicación tuviera un orden simple para que sea fácil interpretarlo
- Como developer/encargado de mantenimiento de la aplicación, sería util que el código estuviera comentado de manera util para que sea fácil interpretarlo

Al ordenarlas por criticalidad o importancia, se ubican de la siguiente manera, donde el número uno es la más crítica, y el cuatro es la menos crítica:

1. Como developer/encargado de mantenimiento de la aplicación, sería util que el código de la aplicación tuviera un orden simple para que sea fácil interpretarlo
2. Como developer/encargado de mantenimiento de la aplicación, sería util que el código estuviera comentado de manera util para que sea fácil interpretarlo
3. Como usuario de la aplicación, sería util que la interfaz gráfica tuviera un diseño simple para facilitar el uso de la misma
4. Como usuario de la aplicación, sería util que esta tenga un apartado, o una descripción de las funciones que puedo utilizar.
5. Como usuario de la aplicación, me gustaría que el almacenamiento de los archivos sea fácil de accesar
6. Como developer/encargado de mantenimiento de la aplicación, sería util que los archivos de configuración sean fáciles de accesar

Minimal System Span

La aplicación se probó en un sistema con las siguientes especificaciones:

- Intel Celeron CPU 925 @ 2.30GHz
- Memoria RAM 4GB
- Sistema operativo Linux Mint 18.2 Sonya 64 bits
- Java Developer Kit 8
- Biblioteca SimpleJSON

Se recomienda utilizar un sistema con estas especificaciones, o cercanas a las mismas. El sistema operativo puede variar mientras se utilice la version de Java equivalente para ejecutar el programa.

Agrupación y decomposición de historias de usuario

En el apartado de “Historias de usuario” podemos ver que se pueden dividir en “Usuario” y “Developer/Mantenimiento”. Esto porque los usuarios de diferente nivel, tienen diferentes necesidades. Mientras que un usuario común necesita cosas como simplicidad y facilidad en la interface grafica, un encargado de mantenimiento necesita que el código sea legible, que se entienda con facilidad, y que sea versatil para poder darle mantenimiento continuo sin tener alguna complicación. Por esto agrupamos las historias de usuario de la siguiente manera:

Usuario	Mantenimiento
• Como usuario de la aplicación, sería util que la interfaz gráfica tuviera un diseño simple para facilitar el uso de la misma • Como usuario de la aplicación, sería util que esta tenga un apartado, o una descripción de las funciones que puedo utilizar. • Como usuario de la aplicación, me gustaría que el almacenamiento de los archivos sea fácil de accesar	• Como developer/encargado de mantenimiento de la aplicación, sería util que el código de la aplicación tuviera un orden simple para que sea fácil interpretarlo • Como developer/encargado de mantenimiento de la aplicación, sería util que el código estuviera comentado de manera util para que sea fácil interpretarlo • Como developer/encargado de mantenimiento de la aplicación, sería util que los archivos de configuración sean fáciles de accesar

El desarrollo incremental de estas se ve por su orden de importancia, se trabajan simultaneamente una de cada columna, por ejemplo la primera de la izquierda junto con la primera de la derecha y así sucesivamente.

Debido a que el equipo es de una sola persona, todas las historias de usuario le quedan como tarea al responsable del proyecto, Eduardo Quiroga Alfaro.

Las tareas por realizar para las historias de usuario, y las features consideradas se pueden dividir en dos fases. La primera siendo todo el backbone del proyecto, todas las funciones que se debe cumplir para que el proyecto funcione correctamente sin necesidad de la interfaz gráfica. Y la segunda es la

interfaz gráfica, donde implementamos todas las partes del backbone para que tenfan un acceso “User friendly” y sean utilizadas con facilidad. Por esto se manejó la siguiente lista de tareas:

Fase 1	Fase 2
• Crear los nodos JSONStore • Crear un manager para los JSONStores • Implementar la estructura de Lista Doble a los JSONStores • Implementar una forma de cargar los Stores ya existentes a la memoria dinámica • Crear un manager de Documentos • Implementar la estructura de Lista Circular a los Documentos • Crear un manager para los Objetos JSON que se almacenan en los documentos • Para la finalización, se implementó una estructura de Facade que accesa a todos los manager anteriores de una forma más simple.	• Implementar a la fase 1 una interfaz gráfica que tome en cuenta las historias de usuario y tenga un diseño simple y fácil de utilizar, y a la vez sea funcional y elegante.

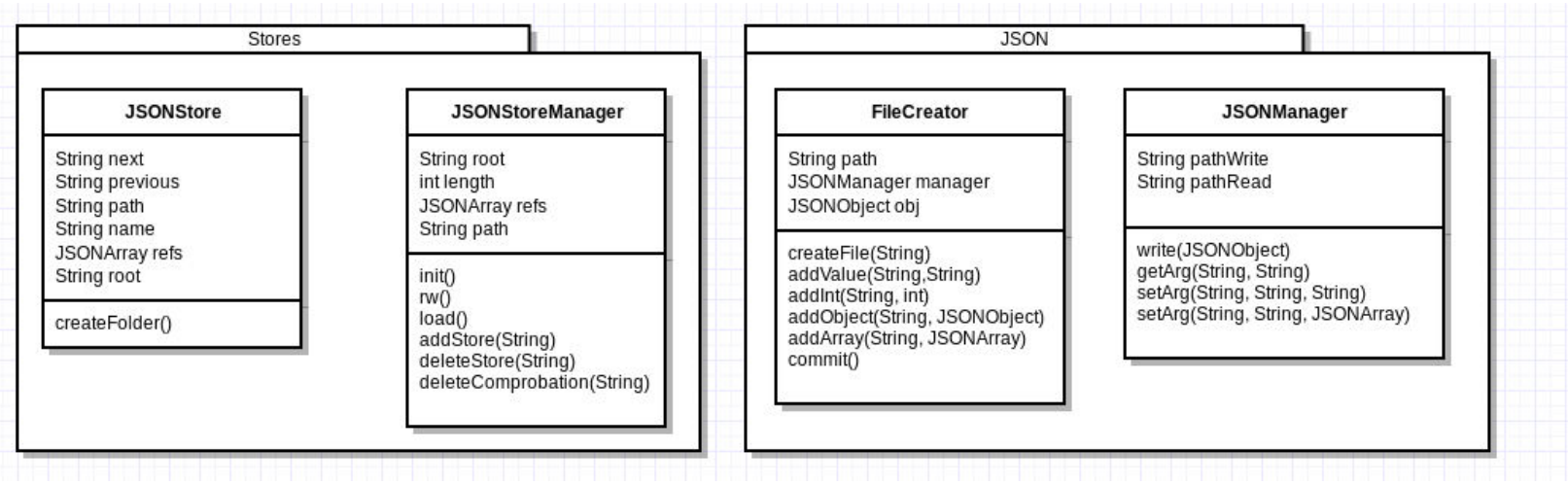
Bitácora

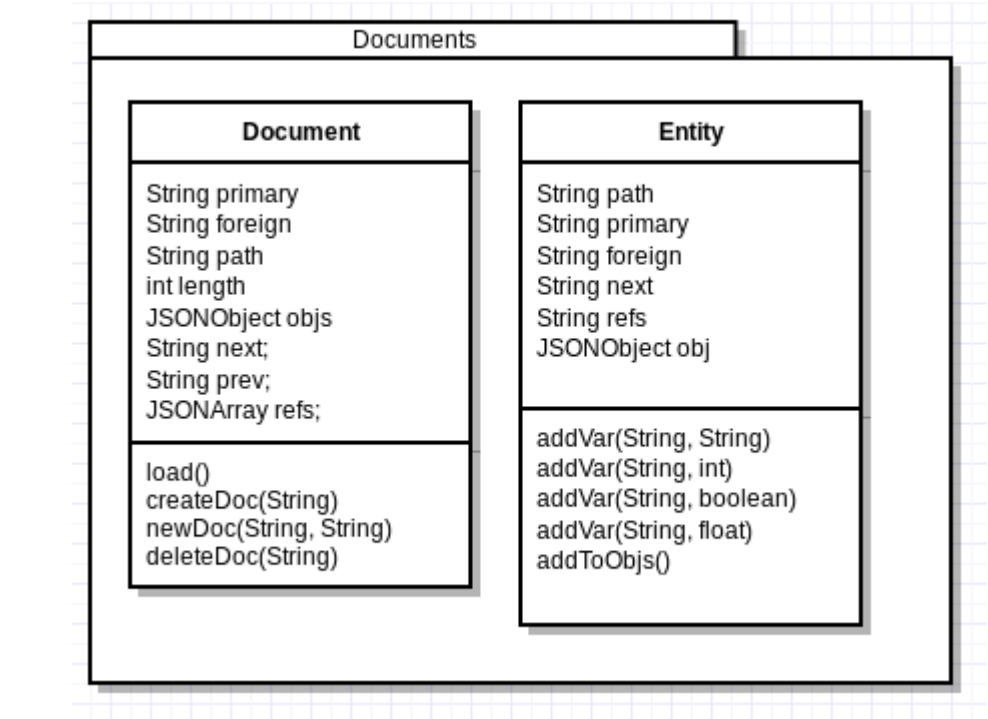
- 10/8/17 Se inicia con el proyecto luego de haber estudiado las estructuras y las herramientas que se van a utilizar.
- 11/8/17 Se escribe el codigo de los JSON Stores y se inicia con el proceso de guardado de datos. 2 horas dedicadas.
- 15/8/17 Se finaliza el codigo de los JSON Stores, se inicia con el manager de los JSON Stores para la creacion y almacenamiennto de estos. 2 horas dedicadas. Se encontró un problema con la estructura de los JSON Stores, y el manejo de los punteros "next" y "previous"
- 19/8/17 Se resuelven los errores encontrados anteriormente y se deja incompleto el manager de los JSON Stores. 1 hora dedicada.
- 24/8/17 Se crea un repositorio de GitHub para el codigo y se continua trabajando en el codigo. 1 hora dedicada
- 26/8/17 Se finaliza con el almacenamiento de los Stores y la implementación de un archivo ".config" dentro de cada una donde se almacenan datos claves para cargar estos a la memoria dinamica y trabajar con ellos.
- Se encuentra un problema con la creacion de los Stores, a la hora de crear todos juntos se ligan correctamente con los punteros, pero si se crea uno por uno, no es capaz de cargar los punteros para agregar nuevos Stores al final de la lista. Tiempo dedicado: 4 horas.
- 29/8/17 Se resuelven los errores encontrados anteriormente. Los Stores se terminan de implementar como una lista doble enlazada. Se plantea la información necesaria para pasar a la siguiente instrucción de crear documentos dentro de los Stores. Tiempo dedicado: 10 horas.
- 6/9/17 Los documentos se terminan y se inicia con el manejo y almacenamiento de estos. Se encuentra el mismo error que en los Stores, y además no se almacenan los nombres de estos en un arreglo que contiene todos los nombres de los documentos contenidos dentro de un Store. Se dedican 3 horas este dia.
- 16/9/17 Se corrijen los errores de los documentos y se implementan los objetos JSON posibles de añadir en cada uno de los documentos. Se dedican 5 horas este día al trabajo.
- 18/9/17 Se finaliza con la lógica y se implementa una interfaz de usuario para poder manejar facilmente los archivos JSON y sus respectivos Stores. Se dedican 3 horas de trabajo.

Aparte de esta bitácora, hay un progreso paralelo sin documentar en la parte de la interfaz gráfica, ya que durante la implementación del código, se investiga sobre el funcionamiento del Scene Builder y como es aplicable para este proyecto.

3. Diseño del proyecto

Diagramas de clases





4. Implementación

Bibliotecas utilizadas

Para el manejo de los archivos JSON se utilizó la biblioteca SimpleJSON. Esta ya viene incluida en la página de Github asociada al proyecto. La bilbioteca a pesar de carecer de flexibilidad para ciertas situaciones, se utiliza de una manera fácil y práctica. Algunas de las clases utilizadas de esta biblioteca fueron JSONObject, JSONArray, y JSONParser.

Estructuras de datos utilizadas

Las estructuras de datos que se utilizaron fueron las listas enlazadas, de estas se utilizaron la lista sencilla, o lista simple enlazada, la lista doble enlazada, y la lista doble circular.

Las estructuras de listas consisten en una estructura compuesta por nodos. La lista tiene dos atributos básicos que se repiten en las tres ya mencionadas, el "root" y el "length". Estos vienen siendo el primer nodo, y la cantidad de nodos que hay, respectivamente. Por otro lado, los nodos tienen tres dos atributos básicos, los punteros y los datos. En el caso de las listas sencillas, los nodos van a tener un solo puntero que va a señalar al siguiente nodo que venga en la lista, mientras que los de las listas dobles y dobles ciculares, tienen dos nodos para apuntar al siguiente y al anterior. En caso de las listas circulares, el puntero "next" del último nodo va a apuntar al primer elemento, mientras que el puntero "previous" del primer elemento va a apuntar al último nodo. Si este no es el caso de que es una lista circular, estos van a apuntar a *null*.

Algoritmos desarrollados

Los algoritmos desarrollados en este proyecto consistían en leer y guardar archivos JSON. No existe mucha complejidad en estos pero sí pueden ser un poco conflictivos a la hora de trabajar con ellos a nivel de programación.

La primera fase de esta parte que fue desarrollada es un Manager para poder crear y eliminar Stores, este es llamado JSONStoreManager. Los metodos son simples, se maneja una ruta donde se guardan los archivos, y en esta se genera un archivo ".config", en caso de que exista simplemente lo carga. Dentro de este archivo existen referencias a todos los Stores, y atributos especificos para que estos sean parte de la lista doble enlazada que se especifica.

La segunda fase fueron los documentos. Estos son archivos JSON que se almacenan dentro de los Stores, y se manejan con una clase llamada Document, encargada de generar, eliminar y actualizar los documentos que se almacenan en los Stores. Cabe destacar, que el JSONStoreManager también genera otros archivos ".config" dentro de los Stores para poder hacer referencia y cargar cada uno de los documentos a la memoria dinamica.

Para finalizar, dentro de los documentos existen más objetos JSON, llamados entidades, y estás son las que se encargan de almacenar los datos dentro de los documentos. Estos pueden tener cualquier cantidad de atributos además de los ya especificados. Las entidades se manejan mediante una clase llamada Entity.

Para finalizar, se implementó una estructura llamada UserInterface, para simplificar el acceso a las estructuras de datos que se manejan, y tener una interacción más limpia, de manera que no se complique la programación e implementación directa en futuras ocasiones.

Todas las clases mencionadas vienen empaquetadas en el archivo correspondiente.

Problemas Encontrados

A la hora de trabajar con esta clase de archivos, los principales problemas encontrados se resuelven todos de la misma forma, porque se causan por la misma razón. Al guardar datos en el disco duro, debemos encontrar la forma de que la ruta que nos brindan para la lectura y escritura sea correcta, porque de lo contrario no se van a efectuar los cambios propuestos. Además de esto, los demás problemas que se presentaron tenían que ver con los tipos de archivos, por ejemplo que no se puede convertir un String en un JSONObject por medio de la forma convencional, entre otros.

La solución se puede implementar completa a pesar de los problemas causados por la interfaz gráfica, pero si se considera solamente el Backbone del proyecto, todo funciona a la perfección.

Recomendaciones:

• Utilizar Eclipse para programar • Utilizar la bilbioteca SimpleJSON para manjera los JSON • Conocer y saber manejar archivos JSON • Tener un conocimiento básico de Java • Implementar las soluciones de manera modular para encontrar los errores paso por paso • Crear un programa previo para planificar las fases del proyecto	• Hacer todos los diagramas planeados antes de comenzar con el código • Llevar una bitácora al día y documentar todos los avances • Utilizar GitHub para el versionamiento de código • Avanzar de forma paralela con la interfaz grafica y la lógica del backbone

Conclusión

Este proyecto fue una experiencia que deja una enseñanza bastante buena sobre estructuras básicas de datos como las listas, además de cómo manejar Java y hacer aplicaciones en el mismo.

Java es un lenguaje bastante completo y aprovechable ya que pertenece al paradigma de orientación a objetos. En mi caso el que más utilicé fue el polimorfismo para poder utilizar un mismo metodo más de una vez, y simplemente cambiar los datos de entrada.

Uno de los mayores retos encontrados fue buscar la manera de cargar los archivos ya existentes en la memoria dinamica, y poder accesar a ellos a pesar de que no estuvieran cargados.

Poniendo de lado el código, siento que fue un proyecto bastante pesado, porque todos los temas vistos fueron temas nuevos, y esto junto que es un proyecto individual, genera un gran peso de trabajo para el programador, ya que no se puede dividir las tareas de una forma eficaz entre varios trabajadores.

Para finalizar, me gustaría agregar que fue un proyecto del que se aprendió bastante, a pesar de no haber logrado completar al 100% los objetivos, es un proyecto que fue bastante aprovechable.

Bibliografía

- Deitel y Deitel, (2012). Cómo programar en Java. 9th ed. México: Pearson Educación.
- What Is JavaFX? | JavaFX 2 Tutorials and Documentation. (2013, April). Retrieved from http://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm

Anexos

Página de GitHub: https://github.com/ed-quiroga-2103/Proyecto1-Datos-I