UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

INFORME PARADIGMAS DE PROGRAMACIÓN
PROGRAMACIÓN ORIENTADA A OBJETOS

Autor: Nicolás Venegas

Asignatura: Paradigmas de Programación

24 De Enero Del 2022

UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Tabla de Contenidos

1. Introducción

Para este informe se tiene como proyecto la elaboración de un editor de texto colaborativo que siga el estilo de Google Docs, esto mediante el lenguaje de programación Java guiándonos por el paradigma orientado a objetos,siguiendo la pauta de clases e instancias, herencias y relaciones entre objetos, posteriormente se visualizará el metodo de solucion del problema y las distintas clases necesarias para realizar la implementación.

2. Descripción del problema

Para este laboratorio se tiene la falta de un editor de texto colaborativo al estilo de Google Docs por lo que se solicitó la creación del mismo usando una tecnología en específico, que es este caso es java integrado a gradle, y siguiendo una cantidad mínima de requerimientos para que la herramienta sea funcional los cuales serán indicados más adelante en la seccion de analisis del problema.

3. Descripción breve del paradigma y los conceptos del mismo que se ven aplicados en este proyecto

En este desarrollo se ocupó él paradigmas orientado a objetos que como lo indica el nombre está basado en la percepción de elementos como objetos, teniendo estos atributos propios y comportamientos específicos.

En la construcción del programa se van a diferenciar 3 términos regularmente:

- Recursión: es el proceso donde una función se llama a sí misma con un objetivo, esta puede ser de cola, natural u arbórea dependiendo de cómo se implemente.

- clase: molde para poder crear objetos, estos definen sus atributos y métodos aplicables al objeto

- Objeto: es la instancia que se crea a la hora de ejecutar el programa, este puede definirse previamente en una clase y puede tener diversos comportamientos.

4. Análisis del problema respecto de los requisitos específicos que deben cubrir

Como se mencionó anteriormente se nos pide programar un editor de texto colaborativo del estilo Google Docs pero este tiene una lista de funcionalidades mínimas para que se considere como exitoso, siendo estas:

- Creación de la plataforma
- Registro de usuarios
- Logeo de usuarios
- Creación de documentos
- Compartir documentos entre usuario
- Agregar contenido a un documentos
- Restaurar Versiones de un documento
- Mostrar información del usuario junto a la plataforma creada

Como se puede ver en esta lista de requerimientos ya se puede ir deslumbrando lo que se necesitará para cada estructura, por ejemplo ya sabemos que una plataforma requerirá una lista con usuarios y que estos necesitan un nombre y contraseña para poder loguearse.

También es posible visualizar la necesidad de las siguientes clases que contengan distintos atributos (Ver diagrama inicial en anexo 1):

Clases	Atributos
Usuarios	Nombre Contraseña lista de documentos
Versión	Id Fecha Contenido
Acceso	Usuario Tipo Acceso
Documento	Nombre Autor Fecha de creación Lista de permisos Lista de usuarios compartidos lista de versiones Contenido
Plataforma	Nombre plataforma Fecha de creación usuario Activo

Menú	paradigmadocs run

Por último cada clase debe tener sus propios métodos, que al actuar en conjunto a la de las demás estructuras es posible realizar las tareas pedidas anteriormente (Ver diagrama post implementación en anexo 2).

5. Diseño de la solución

En esta sección de diseño de solución se especificará qué métodos fueron necesarios para las clases con el fin de crear las funcionalidades mínimas.

Antes de todo fue la preparación y programación de las clases dichas en el análisis de solución a partir de eso se creó un constructor para los objetos, y gracias a la ayuda de netbeans se creó un getter y un setter para cada atributo de todas las clases.

- **Registro de usuarios**
  - Para este método solo fue necesario recorrer los usuarios ya creados, y si el que se intenta agregar no existía, se agregaba a la lista de usuarios y retorna true, en caso contrario no se agrega y el método retorna false como indicación que algo falló.

- **Logeo de usuarios**
  - Para este método al igual que en register se recorren todos los usuarios y gracias al metodo canLogin, que indica si coincide el user y password de un usuarios con 2 strings entregados, podemos ver si un usuario puede logear, en caso de que pueda se agrega al usuario activo y el método retorna true.

- **Creación de documentos**

  - En este método se obtiene el usuario activo y posteriormente se inicializa un documento, finalmente este se agrega a la lista de documentos del usuario y retorna true, en caso de fallo retorna false para indicar de algún error.

- **Compartir documentos entre usuarios**
  - Este método fue apoyado principalmente por otros 3 métodos, el getDocumentById, que como el nombre indica devuelve un documento pasandole un id, canEdit que indica si un usuario tiene permiso de escritura en un documento y se Acceso que agrega un acceso a un documento si este no existe y lo modifica

si ya existe, luego en la implementación simplemente se extrajo el documento, se comprobó si el usuario activo tiene los permisos necesarios y en el caso que los tuviera se agrega el acceso con los permisos indicados

- **Agregar contenido a un documentos**
    - Para este método se ocupó un método de la clase documento que agrega contenido al documento y en este método se hicieron todas las comprobaciones necesarias para que el usuario activo pudiera escribir solamente si es el autor o tiene dicho permiso

- **Restaurar Versiones de un documento**
    - Esta fue muy simple ya que se reutilizaron la mayoría de funciones de la funcionalidad anterior, cambiando que en vez de agregar contenido al final del documento este se setea con el contenido guardado en la versión buscada por id (método creado en la clase Documento) y el contenido previo fue guardado en una nueva versión y añadido al documento.

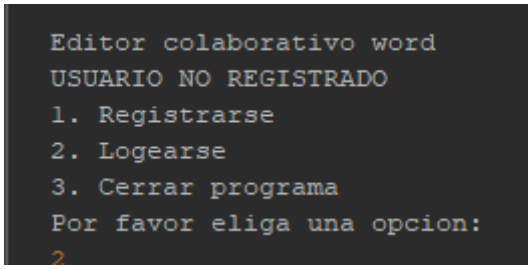
6. Aspectos de implementación

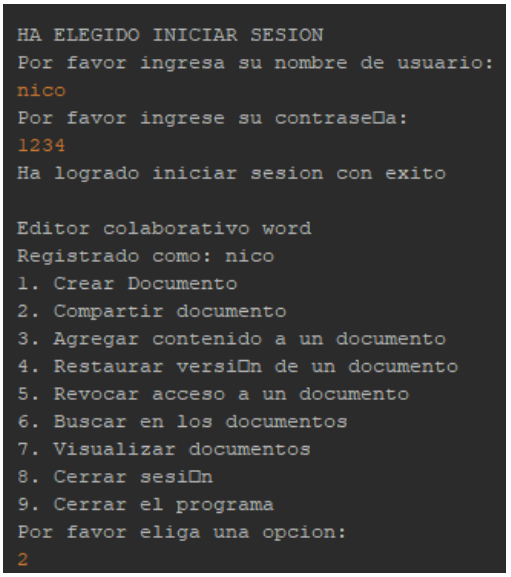
para este proyecto fue ocupado java en su versión 11 del JDK junto al editor e IDLE Apache Netbeans en su versión 12

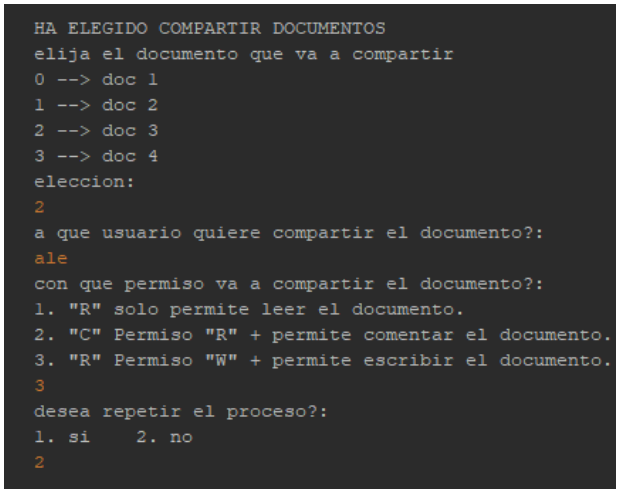
De igual manera no fue necesaria ninguna biblioteca adicional además de las que proporciona el mismo java, aunque si las contamos fueron ocupadas funcionalidades de la librería Java.Util que provee el mismo java.

7. Instrucciones de uso y resultados

No hay que comentar mucho en las instrucciones de uso a nivel de manual ya que el programa tiene una interfaz intuitiva para los usuarios, el programa inicia pidiendo un nombre para la plataforma, luego se entra al menú inicial sin estar logueado por lo que solo tendrán 3 opciones, registrarse, loguearse y cerrar el programa, luego de iniciar sesion el menú cambia dando las opciones completas al usuario.







(imagenes del menu en funcionamiento)

En cuanto a los resultados esperados de la funciones se puede decir que estas funcionan buena manera exceptuando los casos donde se introduzcan datos erróneos o inexistentes, arrojando un false(también esperado), esto ocurre debido a que este lenguaje y el laboratorio en general está orientado a usuarios que ya saben usar el lenguaje y tienen un plan de lo que quieren hacer por lo que a pesar de que es posible ocuparse de esos errores no nos corresponde hacerlo (en otras palabras no es nuestra responsabilidad).
En la sección de anexos es posible encontrar los ejemplos de todas las funciones siguiendo un camino entendible.

En el único anexo existente se puede encontrar algunos ejemplos del uso del programa.

8. Autoevaluación

Durante la creación del programa y desarrollo del laboratorio surgieron bastantes errores respecto a las funciones mínimas requeridas pero todos estos fueron solucionados dando de esta manera una versión estable y correcta del proyecto, por lo que la autoevaluación me concedo un aprobado ya que casi todos los requerimientos fueron cumplidos exceptuando por la muestra de información y a pesar de ser lograble siento que el primer motivo del fracaso en esta función es la falta de tiempo.

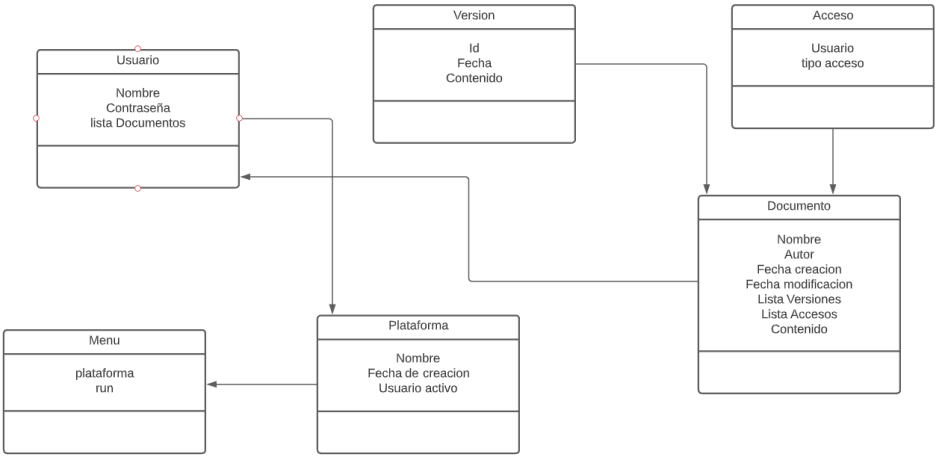
9. Conclusiones

Luego de realizar este proyecto es posible ver la programación con un nuevo punto de vista, a pesar de que hubo complicaciones al aplicar este punto de vista también es posible decir que ahora la versatilidad y proponer nuevas ideas o formas de abordar un problema han aumentado.

Si se compara con el proyecto de programación funcional y lógico debo decir que este es el que más me gustó ya que fue interesante ver todo como objetos y poder asignarles comportamientos y atributos, también siento que este tambien fue más sencillo pero no sabría decir si es por la experiencia de los 2 laboratorios anteriores o debido al paradigma como tal.

10. ANEXOS

1. Relación entre clases Mediante diagrama uml



2. Diagrama post implementación del proyecto