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

Paradigmas de Programación

Laboratorio N°3

Alumno Victor Hurtado Meza
Sección: B-2.
Profesor: Víctor Flores Sánchez.

Santiago-Chile
2-2021

Tabla de Contenidos

CAPÍTULO 1. INTRODUCCIÓN

El siguiente informe tiene como objetivo mostrar al lector todos los detalles relacionados al Laboratorio 3 de Paradigmas de Programación, los cuales a grandes rasgos pueden ser resumidos como descripción del problema, descripción del paradigma utilizado, el cómo se resolvió el problema y las diversas estructuras, TDAs o clases que se utilizaron para la implementación de la solución, además de la forma de utilizar este programa a través de Javac y la explicación de las diversas funciones implementadas.

1.1.1 Descripción del problema

Como alumnos de la asignatura de Paradigmas de programación, para el presente semestre 2/2021, se nos pide desarrollar un editor de texto colaborativo similar a Google Docs, entre otros. Para lograrlo, se debe implementar características que usa una plataforma, como por ejemplo: Crear una cuenta, iniciar sesión, crear un documento, etc. Además, cada usuario tiene características como nombre, contraseña, fecha de creación, etc. Para ello, se les solicitó a los alumnos que el programa debiese estar escrito en el lenguaje de Java y esto conlleva a seguir el patrón MVC (model, visual, control), es decir, se distribuirá el código en distintas secciones, lo cual permite poder clasificar la información, la lógica del sistema y la interfaz que se le presenta al usuario.

1.1.2 Descripción del paradigma utilizado

En esta ocasión, se solicitó que se utilice el paradigma POO (Programacion Orientada a Objetos) para crear la solución al problema. POO tiene características bastante representativas, como la abstracción donde buscamos los detalles fundamentales que puede tener un objeto, el polimorfismo y la herencia, donde un objeto adquiere las propiedades y métodos de otro objeto. En POO encontramos clases, que son modelos que definen atributos y métodos comunes, a todos los objetos que sean de dicha clase.

CAPÍTULO 2. SOLUCIÓN DEL PROBLEMA

2.1.1 Análisis del problema respecto a sus requisitos específicos.

Para el desarrollo de la solución se solicitan los siguientes requisitos funcionales específicos:

- Implementar abstracciones apropiadas para el problema mediante el usoclases y estructuras que modelen de forma correcta el problema entregado.
- La creación de un Menú interactivo por terminal, el cual debe incluir un menú por terminal/consola que permita la interacción del usuario con la solución.
- Método authentication: Función que permite que registrar usuarios enuna plataforma, iniciar sesión en una plataforma o desconectarse de una plataforma.
- Método create: Función que permite a un usuario con sesión iniciada en la plataforma crear un nuevo documento.
- Método Share: permite compartir un documento con otros usuarios especificando el tipo de acceso a éste (lectura, escritura, comentarios).
- Método add: Funcionalidad que permite añadir texto al final de la versión actual/activa del documento.
- Método rollback: permite restaurar una versión anterior de un documento.
- Método revokeAccess: permite a un usuario revocar todos los permisos concedidos a otros usuarios en un documento.
- Método Search: permite al usuario buscar documentos (propios o que le hayan sido compartidos) que contengan un texto específico.
- Método visualize: permite a obtener una representación visual y comprensible por el usuario del editor colaborativo. Esta funcionalidad está dividida en dos partes: Transformar el editor colaborativo a un string (EditorToString) e imprimir el string recién generado (PrintEditor).

- Además de los métodos antes mencionados, se da la libertad de escoger otros métodos que se pueden implementar en el programa. En este caso no se implementaron métodos no obligatorios.

2.1.1 Diseño de la solución.

El diseño de esta solución consiste en crear distintas clases que representen de manera apropiada el problema, con eso en mente, se implementaron las siguientes clases que se presentarán a continuación.

Clase Usuario:

Corresponde a la presentación de un usuario en una Plataforma. La clase User tiene como atributos una id, única para cada usuario, un nombre de usuario como string,una contraseña para autenticar al usuario, una lista de Documentos que corresponden a las publicaciones hechas por el usuario. Además de tener como métodos agregarPubliUser() y addUserPost().

Clase Publicaciones:

Clase correspondiente a lo que sería un documento dentro de la plataforma. La clase Publicaciones tiene como atributos una id única, un String que corresponde al título del documento, un String con el contenido del documento, una fecha de creación del documento, un Usuario que vendría a ser el autor, 3 listas de Strings con los permitidos del documento y una lista de Versiones correspondiente al historial del documento. Además, tiene como métodos addEscritura() para agregar los permitidos a escritura, addLectura() para agregar los permitidos a lectura, addComentario() para agregar los permitidos a comentarios, y addPostVersion() para agregar la publicación anterior a la lista de versiones (historial).

Clase PL:

Clase principal del programa, en donde interactúan y se agrupan las demás clases. Tiene como atributos una lista de tipo Usuario con los usuarios registrados en ala plataforma, una lista de tipo Publicaciones con los documentos realizados en la plataforma, un atributo tipo Usuario con toda la información del usuario online y una variable tipo boolean que marca si hay un usuario online o no.

Clase Versiones:

Clase correspondiente a lo que sería el historial dentro de la plataforma. La claseVersiones tiene como atributos una id única, un String que corresponde al título del documento, un String con el contenido del documento, una fecha de creación del documento, un Usuario que vendría a ser el autor y las 3 listas de Strings con los permitidos del documento.

```
### Bienvenido a Google Docs ###
Escoja la opción que desea realizar:
1. LOGEARSE
2. REGISTRARSE
3. VISUALIZAR PLATAFORMA
4. SALIR
Eliga una opcion
3
### Bienvenido a Google Docs ###
### Actualmente contamos con 0 usuarios registrados ###
### Aun no existen documentos en la plataforma ###

### Bienvenido a Google Docs ###
Escoja la opción que desea realizar:
1. LOGEARSE
2. REGISTRARSE
3. VISUALIZAR PLATAFORMA
4. SALIR
Eliga una opcion
2
Register

Ingrese su nombre de usuario:
victor
Ingrese su contraseña:
pass
Registrado con exito

### Bienvenido a Google Docs ###
Escoja la opción que desea realizar:
1. LOGEARSE
2. REGISTRARSE
3. VISUALIZAR PLATAFORMA
4. SALIR
Eliga una opcion
4
Saliendo
```

Figura 1: menú inicial del programa, mostrando algunas opciones y la plataforma en cuestion.

Con dicho menú creado, tenemos distintas alternativas para manejarnos dentro de la plataforma, principalmente se sugiere poder iniciar sesión como “usuario1” con contraseña “pass”, luego de eso se mostrará un menú secundario con las demás opciones disponibles para el usuario:

```
Login

Ingrese su nombre de usuario:
user1
Ingrese su contraseña:
pass
### Bienvenido a GoogleDocs###
## Registrado como: user1 ##
Escoja la opción que desea realizar:
1. Crear nuevo documento
2. Compartir documento
3. Agregar contenido a un documento
4. Restaurar versión de un documento
5. Revocar acceso a un documento
6. Buscar en los documentos
7. Visualizar usuario
8. Cerrar sesión
9. Salir del programa
Introduzca su opción:
```

Figura 2: Segundo menú disponible para un usuario con sesión iniciada.

Para llevar a cabo la implementación de la solución se hizo uso de diagramas UML para poder conceptualizar de mejor manera la forma en la que se relacionan las clases:

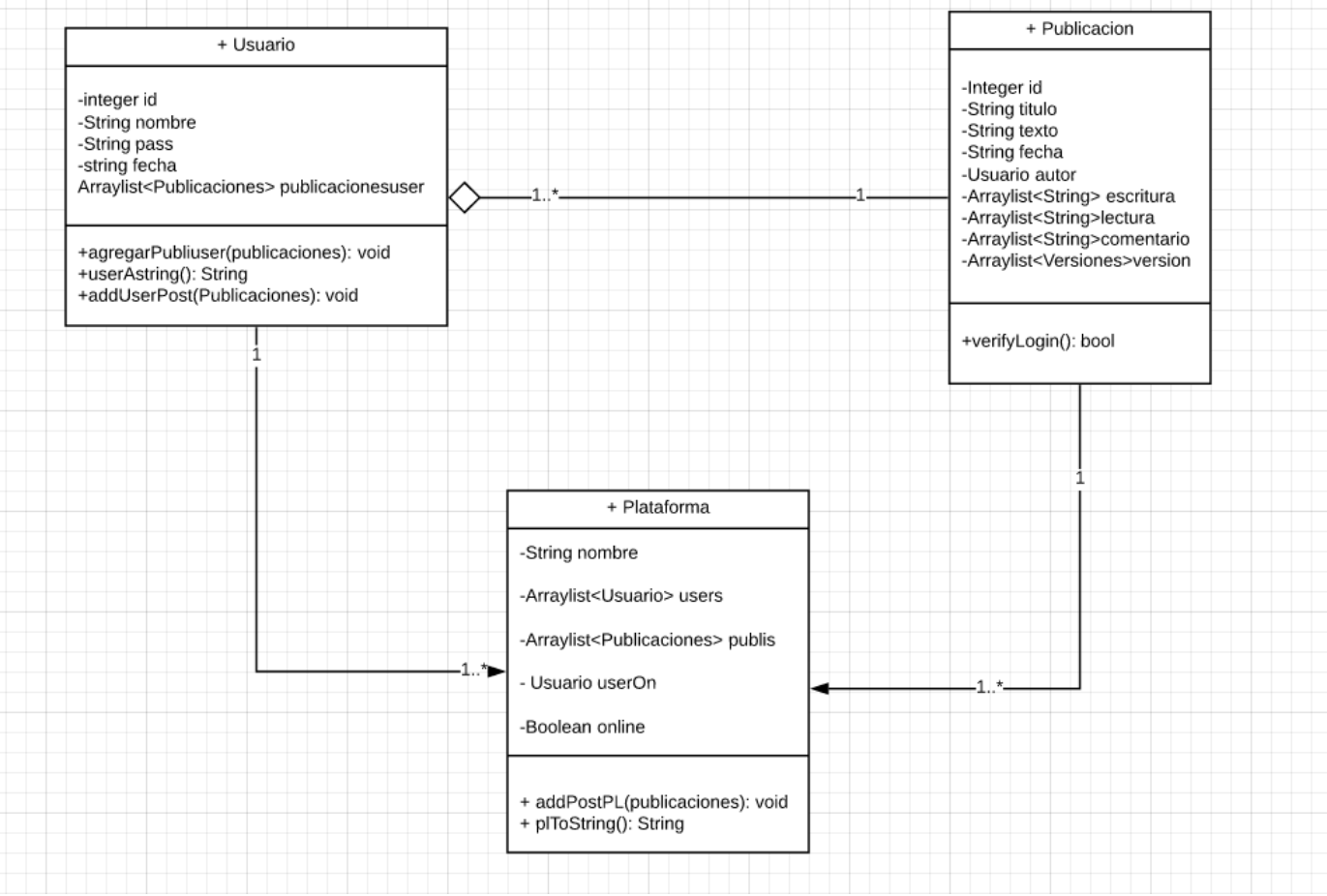

Figura 3: Diagrama de clases al comienzo del proyecto

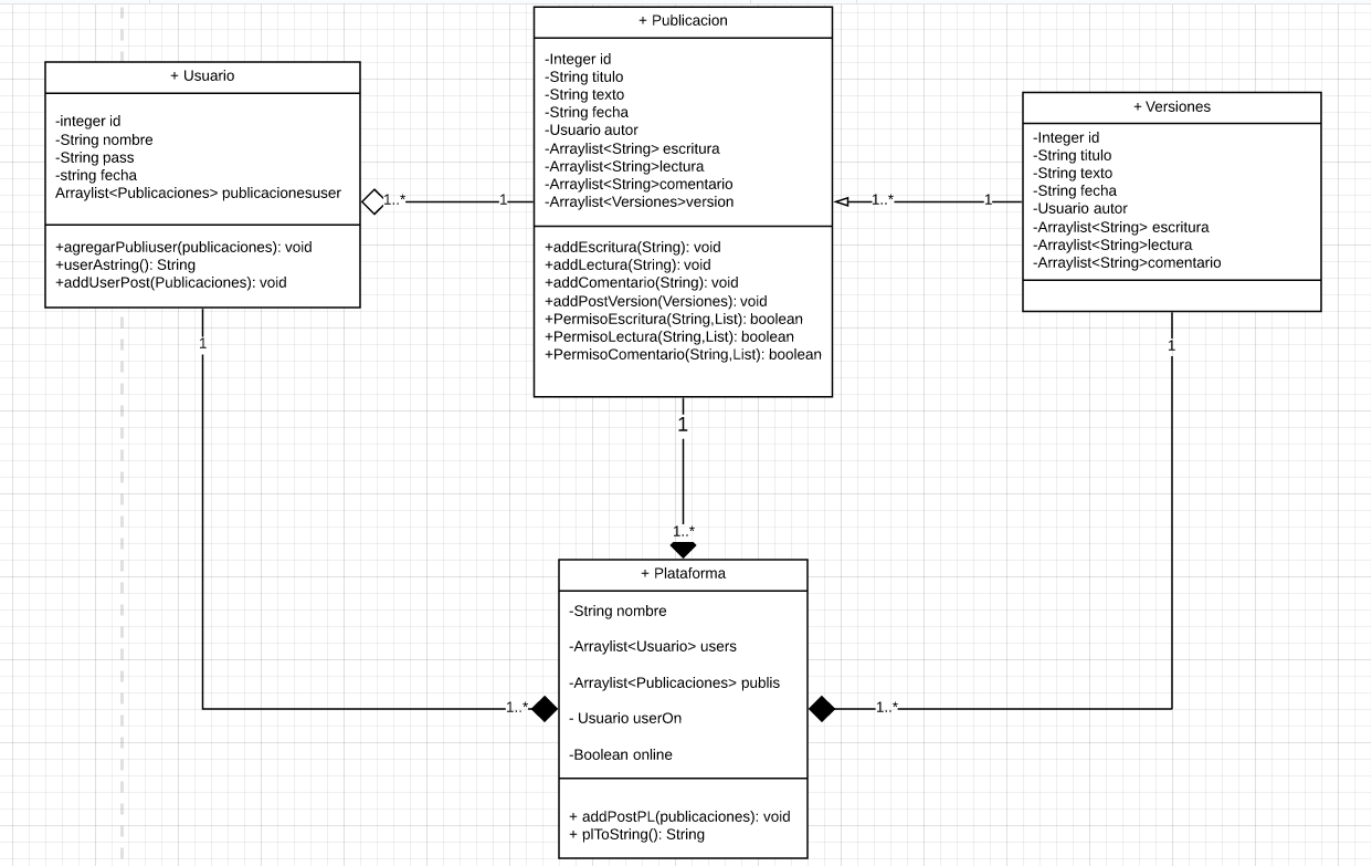

Figura 4: Diagrama UML de la solución final

Como se puede ver, en este caso, a la solución final se le agrego una nueva clase, la clase Versiones, la cual busca modelar de mejor manera el historial que se le dan a las publicaciones, además abre las posibilidades de volver a una versión anterior de un documento.

2.1.1 Aspectos de implementación.

Este programa fue escrito en el lenguaje de Java y el IDE utilizado fue IntelliJ IDEA, en este laboratorio se hizo uso de librerías tales como java.util.Date, java.util.ArrayList, java.text.SimpleDateFormat, entre otros, las cuales son librerías estándares de java.

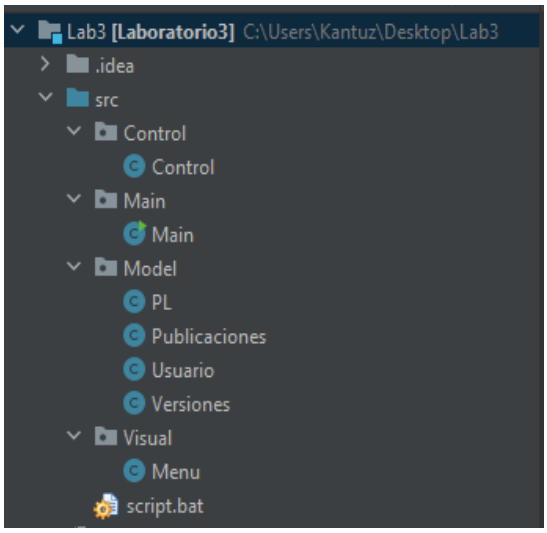

Figura 5: Organización del programa

El programa este compuesto por 4 carpetas, las cuales corresponden a las carpetas siguiendo el formato MVC.

En la carpeta Model tenemos 4 archivos los cuales corresponden a las clases que se pueden ver en la figura 5, en esta carpeta encontramos los archivos "PL.java", "Publicaciones.java", "Usuario.java" y "Versiones.java".

En la carpeta Visual podemos encontrar el archivo "Menu.java" el cual está a cargo de todo lo visual que podemos ver en la consola y controlar las acciones que el usuario es capaz de realizar.

En la carpeta Control podemos encontrar el archivo "Control.java", clase que permiteque el usuario realice todas las acciones que requiera por parte del menú, en esta clase podemos encontrar las funciones que se han requerido

En la carpeta Main podemos encontrar el archivo Main.java, el cual crea una plataforma, crea un controlador, crea un menú y hace el llamado final al menú.

2.1.1 Instrucciones de uso.

En primer lugar, verificar que se tenga instalado Java 11, y en la carpeta donde se encuentre el src abrir una consola en esa dirección. Con ello, podemos ejecutar el “script.bat” el cual nos compilará todos los archivos y nos llevará directamente al menú, una vez dentro del programa es cosa de seguir las instrucciones que se nos muestre por pantalla para poder interactuar con el programa.

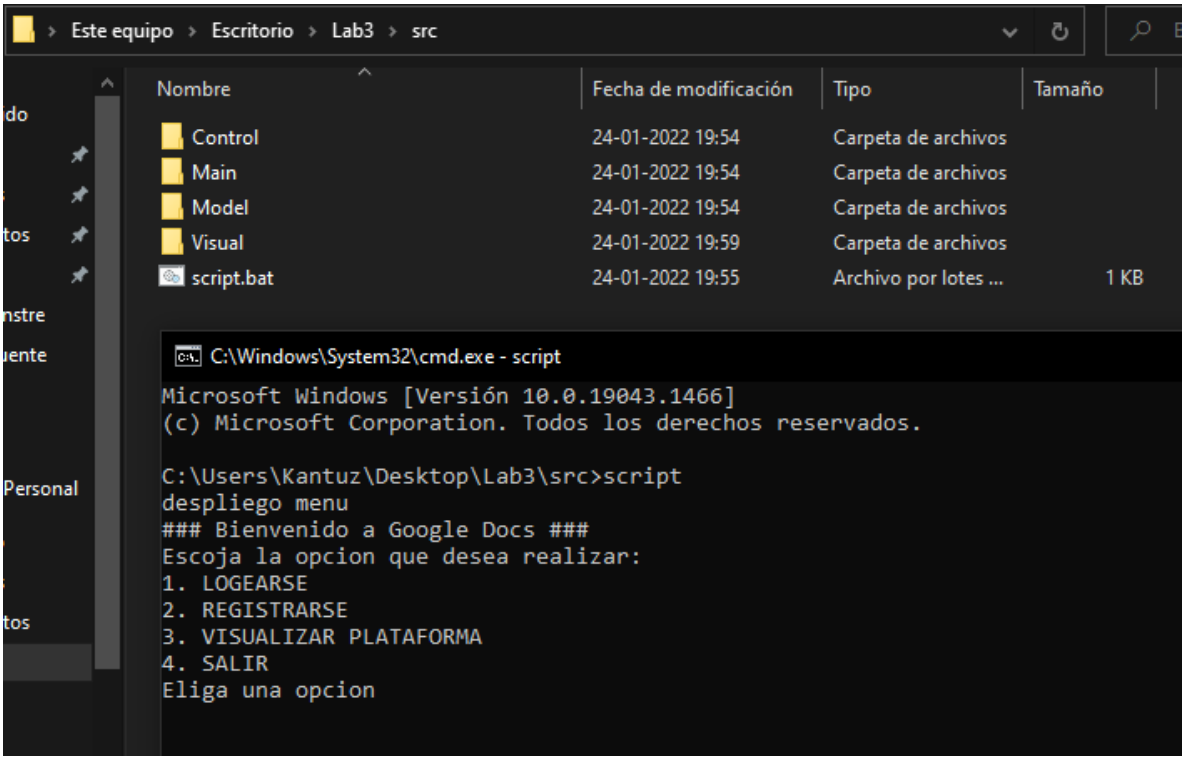

Figura 6: ejecución del programa

El laboratorio cuenta con los métodos presentados en la figura 6 y los siguientes al momento de loguearte:

1. Crear nuevo documento.
2. Compartir documento.
3. Agregar contenido a un documento.
4. Restaurar versión de un documento.
5. Revocar acceso a un documento.
6. Buscar en los documentos.
7. Visualizar documentos.
8. Cerrar sesión.
9. Cerrar el programa.

2.2Resultados y autoevaluación.

En esta sección se hará una revisión de los resultados, para ello se verificarán los requisitosno funcionales y funcionales solicitados

Requerimientos no funcionales:

Lenguaje Java obligatorio	Logrado: el lenguaje usado en el programa es Java.
Versión: utiliza OpenJDK en su versión 11	Logrado: la versión del JDK es la 11.05
Interacciones con el programa: Todas las interacciones con el programa deben ser mediante consola/terminal.	Logrado: El usuario solamente puede interactuar por la red social mediante el uso del menú por consola.
Uso del paradigma	Logrado: Se implemento de manera correcta el paradigma orientado a objetos
Organización del código	Logrado: El programa se encuentra bien organizado siguiendo el patrón MVC.
Diagrama de análisis	Logrado: Se crea un diagrama de análisis previo a la implementación de la solución.
Diagrama de diseño	Logrado: Se presenta un diagrama final con la verdadera implementación de la solución donde se muestra las clases finales.
Historial: al menos 10 commits en un mínimo de 2 semanas, con un mínimo de 10 commits.	Se cumple el mínimo de 2 semanas: primer commit Jan 10, 2022 segundo commit Jan 24, 2022
Prerequisitos de funciones:	Logrado, cada función tiene su prerrequisito implementado como se solicitó.

Requerimientos Funcionales:
Todas las funciones cumplen con los requerimientos funcionales exigidos por la coordinación, por lo que este punto también fue respetado en su gran totalidad.
(1 en todos los métodos “obligatorios”)

2.2 Conclusión.

Concluyendo, finalmente, se puede decir que Java es un lenguaje de programación bastante poderoso en comparación a los lenguajes usados anteriormente, se siente bastante cómodo para escribir y además se puede mantener un orden entre los distintos tipos de clases.

En este caso, se logró cumplir con el rango de commits aunque no son del todo correctos, por temas de carga académica me fui obligado a dividir los tiempos y eso pasa la cuenta,pero se está bastante satisfecho con el trabajo logrado, hubo un buen aprendizaje del paradigma y se pudo crear una implementación al problema acorde a lo que se requería. Aspectos a mejorar, dividir bien los tiempos para realizar los laboratorios a tiempo para poder mantener una implementación de commits acorde a lo solicitado.

Anexo

```
### Bienvenido a Google Docs ###
### Actualmente contamos con 2 usuarios registrados ###
### Mira lo ultimo que se ha publicado ###
Autor: victor ID publicacion: 1 fecha:24/01/2022
titulo:titulo1
cuerpo:texto1
Permiso escritura:[]
Permiso lectura:[]
Permiso comentario:[]
Cantidad de Versiones: 0

Autor: juan ID publicacion: 2 fecha:24/01/2022
titulo:titulo2
cuerpo:texto2
Permiso escritura:[]
Permiso lectura:[]
Permiso comentario:[]
Cantidad de Versiones: 0
```

Anexo 1: Printear lo que se encuentra en la red social