Universidad de Santiago de Chile

Paradigmas de la Programación (13310-0-A-1)

Informe 3 Paradigma Orientado a Objetos

Docente: Roberto González Ibáñez

Estudiante: Nícolas Farfán Cheneaux

Enero 2022

Contenido

Introducción

El siguiente paradigma de esta asignatura corresponde al orientado a objetos, el siguiente informe describirá el problema a modelar y solucionar para crear una plataforma funcional con todas las características que poseen, del tipo Editar, Compartir

Descripción del problema

Las plataformas de documentos colaborativas tienen múltiples usos y aplicaciones actualmente, sus operaciones principales abarcan desde lo simple, *crear* y *escribir* un documento propio a lo más complejo editar un documento compartido junto con otros usuario, es posible añadir texto, eliminarlo y cada vez que se realice un cambio queda registrado en el historial de versiones como una nueva versión, y el cambio realizado a la última versión o activa se convierte en la nueva versión activa, además sobre los documentos existen permisos (escritura, lectura, comentarios y compartir), para más información ver diagrama(Anexo 1), además sobre los documentos es posible buscar texto en versiones a las cuales se tenga acceso, es decir un usuario anteriormente compartió su documento con este, además se permite revocar los accesos a los usuarios que anteriormente se compartió.

Descripción y uso del paradigma

Los objetos pertenecientes a una clase son una abstracción de la realidad, la cual nos permite estar más cercanos a los objetos u entidades con los que interactuamos en la cotidianidad, pudiendo ser descritos mediante atributos o características y métodos o comportamientos, el modelamiento de estas entidades nos permite interactuar directamente desde el código, existen técnicas herencia, polimorfismo, acoplamiento y encapsulamiento, no es lo mismo que la programación estructurada, ya que las funciones están separadas, en cambio en la programación orientada a objetos, los objetos se relacionan entre sí, comunicándose mediante mensajes.

Análisis del Problema

La presente plataforma de documentos presentada en los requerimientos funciona con 2 elementos particulares: Los **usuarios** y los **documentos,** en este sentido podemos desglosar el problema en 2 entidades fundamentales, donde la primera de estas tiene el control sobre el segundo, pudiendo crear, editar, otorgar acceso, sobre este.

Por otra parte, los documentos tienen su propia estructura y un identificador clave, cada documento es único y puede ser usado al mismo tiempo por otros usuarios (En términos de edición y creación de versiones).

Datos de entrada:

El usuario requiere una autenticación previa para ingresar y usar las características de la plataforma, tiene un conjunto de operaciones que puede hacer simplemente estando registrado en esta, caso contrario no puede acceder a ninguna herramienta.

Por otra parte, no es necesario que el usuario tenga un documento para empezar a editar o crear, puede ser colaborador de otro documento previamente creado por otro usuario.
Los permisos que un usuario puede llegar a tener son: ***Escritura, Comentarios, Lectura***

Los documentos tienen una estructura particular la cual se rige por lo requerido:

- Id
- Autor
- Título
- Usuarios con acceso
- Historial de versiones

Tienen los permisos de write, comment ,read , los permisos son jerárquicos, por lo tanto se estructura de la siguiente manera:

Símbolos: A: autor, W: escritura, C: comentarios, R: lectura, (-): Cualquier Usuario Registrado

- Logearse **(-)**
- Registrarse **(-)**
- Deslogearse **(-)**
- Crear Documento **(-)**
- Añadir Texto **(W)**
- Rollback **(A)**
- Revocar Accesos **(A)**
- Buscar en Documentos **(-)**
- Visualize **(-)**
- Eliminar Texto **(-)**
- Buscar y reemplazar **(W)**
- Aplicar estilos **(-)**
- Insertar comentarios **(C)**

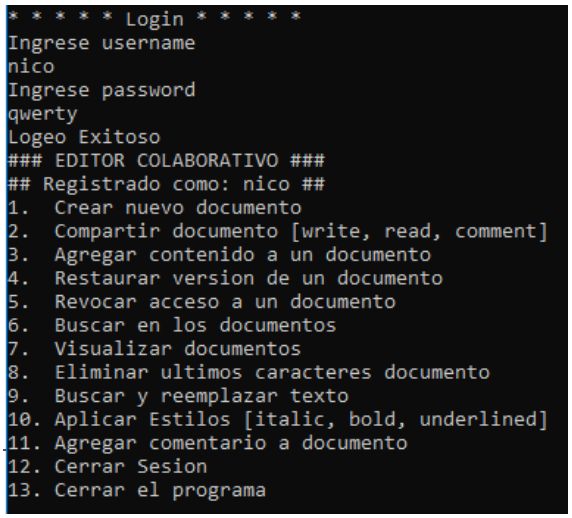
El historial de una versión posee el id, contenido, fecha, lista de comentarios

Datos de salida:

En términos de esta plataforma, se desplegará un menú interactivo inicialmente de la siguiente forma:

```
### EDITOR COLABORATIVO ###
Escoja la opcion que desea realizar:
1. Registrarse
2. Logearse
3. Visualizar Plataforma
4. Salir
Ingrese la opcion:
```

Si el usuario no está registrado, debe antes registrarse e iniciar sesión, para luego al logearse mostrar el menú de usuario logeado:



Donde se muestran todas las operaciones que pueden ser realizadas, descritas anteriormente.

Diseño de la solución

El presente proyecto está basado la programación orientada a objetos, por lo tanto, se modelan las clases:

1) Usuario: Representación de un usuario miembro de la plataforma el cual debe estar previamente registrado, posee un username, password y una fecha de registro, lista de documentos con acceso y lista de documentos creados.
2) Documento: Representación de un documento de la plataforma, posee el dueño de este, contenido, título, fecha de creación, historial y permisos.
3) Acceso: Representación de un acceso a un usuario, compuesto por el usuario del acceso y su tipo de permiso [write, read, comment]
4) Comentario: Representación de un comentario a un documento, compuesto por la fecha del comentario, autor del comentario, contenido del documento a comentar, y el comentario.
5) Versión: Representación de una versión de un documento, compuesto por el id de versión, contenido de esta, fecha de creación de la versión y comentarios de la versión.
6) Editor: Plataforma de edición, la cual crea instancias y métodos de todas las clases anteriores.

Para una mejora en la estructuración se dividió el proyecto en 3 paques distintos:

ParadigmaDocs_Model(Modelo): Contiene todas las clases del proyecto.

ParadigmaDocs_View(Vista): Contiene a la clase Menú donde se implementa el menú por consola.

ParadigmaDocs_Controller(Controlador): Contiene todos los métodos que se realizan con todas las clases y comunica el Menú con el Modelo.

Operaciones de Registro / Logeo

Se utilizan los métodos isLogeado() para determinar si un usuario está logeado en la paltafomra y setLogeado() para activar la sesión de un usuario.

De estos métodos se puede desprender las funciones: **Login, Logout**

Operaciones sobre los documentos

Una vez la autenticación del usuario esté correcta, ahora este usuario está habilitado para realizar cualquier operación, siempre y cuando tenga los permisos adecuados, sin embargo, existen operaciones las cuales no es necesario tener un permiso por ejemplo: **Create/editorToString,**

Creación de documentos:

Create: *(Ver Anexo 2)*

1) El usuario miembro de paradigmadocs esta autorizado de crear los documentos que el desee, identificado por un **id** único y un autor único, más adelante se añadirán colaboradores.

Operaciones de acceso a los documentos:

Share: *(Ver Anexo 3)*

1) El usuario ingresa una lista de permisos y una lista de usuarios registrados a los que se le quiere otorgar, tal permiso.
2) El resultado es añadir a tales usuarios a la lista de accesos, los cuales de ahora en adelante podrán realizar las operaciones que sus permisos le permitan hacer.

Revoke All Accesses: *(Ver Anexo 4)*

1) Contrario a Share, esta operación permite al dueño del documento eliminar todos los accesos que en un principio había otorgado mediante Share.
2) El resultado es eliminar todos los permisos de los documentos que el usuario prefiera.

Operaciones de edición documentos

Estas operaciones pueden ser realizadas por usuarios que tienen permiso de edición sobre los documentos, permiso “W” y los autores, estas son:

Add *(Ver Anexo 5)*

1) Recurre a la versión activa del documento y añade al final de la cadena de texto el nuevo texto a añadir.
2) Crea una nueva versión con el nuevo texto añadido.

Rollback *(ver Anexo 6)*

1) Recurre al historial del documento mediante su id, busca una nueva versión, mediante su Id y cambia la versión activa por alguna escogida por el usuario.
2) El resultado es una nueva versión activa actualizada y la anterior versión se añade al historial.

Delete *(ver Anexo 7)*

1) Recurre a la versión activa del documento y elimina los últimos “n” caracteres ingresados por el usuario.
2) El resultado es una nueva versión creada por el cambio anterior de eliminar los “n” caracteres, la cual pasa a ser la versión activa.

EditorToString *(ver Anexo 8)*

1) Obtiene toda la información de paradigmadocs para convertirlo en String, predicados independientes ejectuan la transformación de cada TDA, (Documento, Versión, Acceso, Usuario)
2) Como el resultado anterior son listas de Strings, se transforma a string mediante predicados de concatenación de String.

Operaciones de búsqueda

Search *(ver Anexo 9)*

1) El usuario ingresa que carácter o palabra desea buscar en todos los documentos que tenga acceso (Sea dueño o se le haya sido compartido).
2) El resultado es todos los documentos en donde se haya encontrado una coincidencia.

Search and Replace *(ver Anexo 10)*

1) El usuario recurre a la versión activa de un documento (propio o con permiso de edición sobre este).
2) Ingresa el carácter o palabra buscada y si se encuentran coincidencias es reemplazado por otro carácter o palabra ingresado por el usuario
3) El resultado es una nueva versión activa producto del reemplazo de una palabra/carácter.

Operaciones de búsqueda

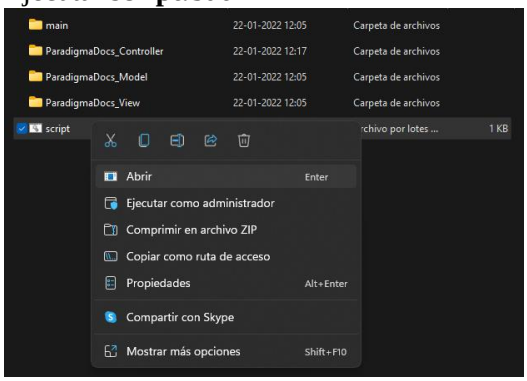
Comment *(ver Anexo 10)*

1) El usuario selecciona un documento y su versión activa y un texto perteneciente a esta, para insertar un comentario.
2) El comentario permanece anclado a tal versión/documento mediante un objeto del tipo comentario que almacena el autor/contenido cometario/ sección del texto que se comentó.

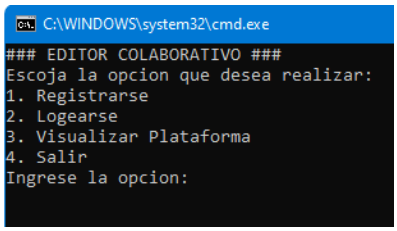
Consideraciones de Implementación

Instrucciones de Uso

1) Clonar el repositorio y dirige a la carpeta **src**
2) Ejecutar **script.bat**



3) Se abrirá una consola de la siguiente forma, el programa se está ejecutando correctamente



Bibliotecas empleadas

Para este proyecto no se consideró el uso de ninguna biblioteca en particular, solo nativas de java, sin ninguna importación.

Compilador o Interprete usado

Lenguaje de programación Java, compilado en OpenJDK versión 11

Ambiente de desarrollo

Se utilizó el IDE, IntelliJ IDEA 2021.3.1

Razones de la elección

Es requerimiento trabajar en este lenguaje y en su respectivo compilador y versión.

Resultados Esperados

Se espera que el proyecto sea útil y cumpla con los requerimientos de una plataforma de documentos.

Posibles Errores

Errores con la compilación y versión de JDK.

Resultados y Autoevaluación

Finalmente probando el proyecto con los ejemplos provistos, obtenemos una plataforma completa con todas las funciones, funcionando correctamente.

Evaluación

Requerimiento	Descripción	Resultado
Clases	Los nuevos tipos de datos creados cumplen su función y forman parte del resultado final.	
Menú	Menú interactivo para el uso de la plataforma	
Register	Registrar a cualquier nuevo usuario que quiera utilizar ParadigmaDocs.	
Login	Permite iniciar sesión a cualquier usuario previamente registrado y queda habilitado para realizar cualquier operación.	
Logout	Permite cerrar sesión a su cuenta	
Create	Permite al usuario crear documentos.	
Share	Permite al usuario compartir documentos con otros usuarios de la plataforma.	
Add	Permite al usuario añadir texto a los documentos, los cuales es dueño o posee permiso de edición.	

Rollback	Permite al usuario restaurar una versión anterior no activa del documento y convertirla ahora en una nueva versión activa.	
Search	Permite buscar texto en un documento	
Visualize	Permite visualizar ParadigmaDocs de una forma clara, entendible para el usuario.	
Delete	Permite al usuario eliminar contenido de la versión activa de los documentos los cuales dueño o posee permiso de edición.	
Search And Replace	Permite al propietario de un documento eliminar todos los accesos anteriormente otorgados	
Apply Styles	Permite al usuario buscar texto en cualquier documento y añadirle estilos del tipo #\i #\b #\u	
Comment	Permite buscar texto en la versión activa del documento, y añadir un comentario	

Diagrama de Análisis

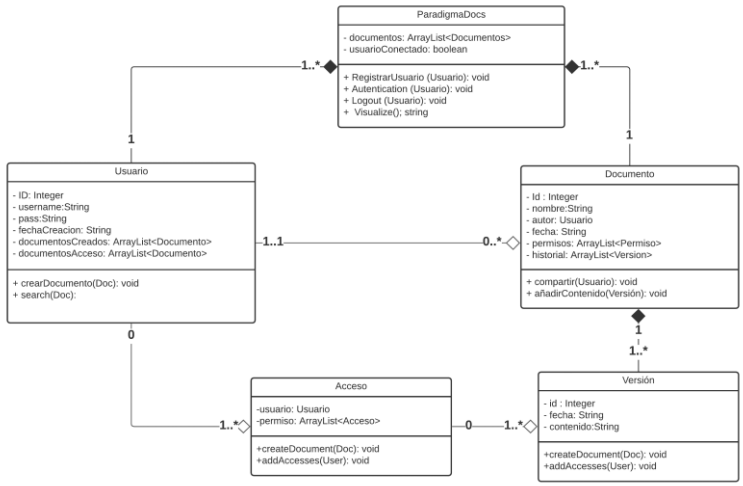
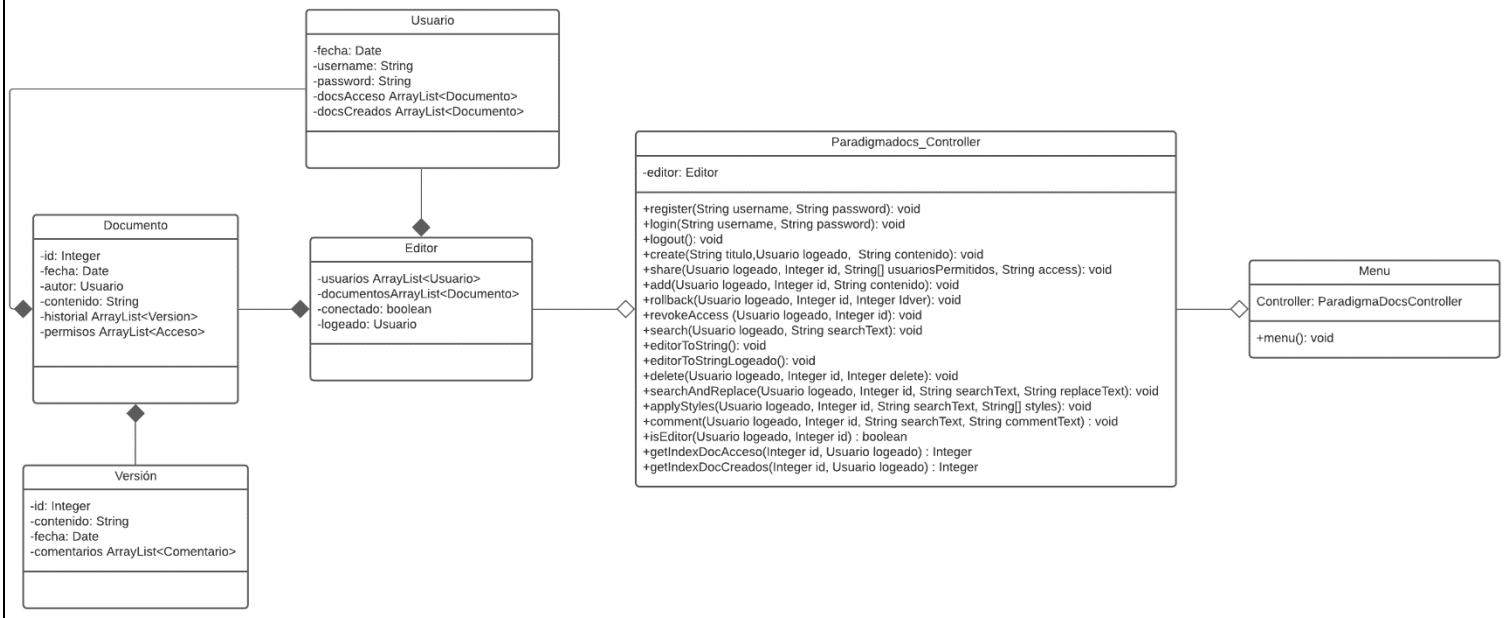


Diagrama de Diseño



Conclusión

Alcances

La programación orientada a objetos tiene como objetivo mejorar en el largo plazo la mantenibilidad del proyecto, estructurándolo de forma en que se quieran cambiar u modificar funcionalidades, los objetos por otra parte al ser abstracciones de la realidad, son útiles para el modelamiento de entidades perfectamente representables en código.

Limitaciones

El modelamiento en este paradigma se debe tomar por etapas y paso por paso para tener una solución mantible en el tiempo, empleando recursos como los diagramas de clases y herramientas de modelamiento y conceptos como la cohesión y acoplamiento, en todas las clases y paquetes.

Contraste Paradigma Anterior (Funcional y Lógico)

En los pasados proyectos, como tal no había que modelar un sistema que en el futuro logre incorporar nuevas funcionalidades, y objetos, por lo que anteriormente solo se trabajaba con conceptos básicos de la programación funcional como las funciones y en Prolog declarando lo que es o no es.

En el presente paradigma al utilizar la programación imperativa, se debe recurrir a declarar más cosas y decirle al programa como lo tiene que hacer, por lo tanto, el código se torna más extenso.

Referencias

UML Class Diagram Tutorial. (2022). Retrieved January 23, 2022, from Visual-paradigm.com website: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/

Great Learning Team. (2021, January 2). OOPs concepts in Java | What is OOPs in Java? Retrieved January 23, 2022, from GreatLearning Blog: Free Resources what Matters to shape your Career! website: https://www.mygreatlearning.com/blog/oops-concepts-in-java/#:~:text=OOps%2C%20concepts%20in%20java%20is,real%2Dworld%20entities%20in%20programs.

Roberto González. (2021). Programación funcional, Paradigmas de programación. Sitio web: https://uvirtual.usach.cl/moodle/

Anexo

Ejemplos:

Create:

```
 * * * * * Create * * * * *
Ingrese titulo del nuevo documento
NICODOC
Ingrese contenido del nuevo documento
HOLAEJEMPLO
Documento creado con exito
```

```
        * * * * * NICODOC id(10) * * * * *
HOLAEJEMPLO
Numero de versiones [1]
-> Permisos
```

Share:

```
* * * * * Share * * * * *
Ingrese el ID del documento al que quiere conceder permisos
10
Ingrese los usuarios separados por espacio
kim chuck
Ingrese permiso del tipo:
[w]: escritura
[r]: lectura
[c]: comentarios
r
Permiso [r] concedido a kim
Permiso [r] concedido a chuck
```

```
        * * * * * NICODOC id(10) * * * * *
HOLAEJEMPLO
Numero de versiones [1]
-> Permisos
        kim -> leer
        chuck -> leer
```

Add:

```
* * * * * Add * * * * *
Ingrese el ID
10
Ingrese contenido
ANIADIENDO EJEMPLO
```

```
        * * * * * NICODOC id(10) * * * * *
HOLAEJEMPLOANIADIENDO EJEMPLO
Numero de versiones [2]
-> Permisos
        kim -> leer
        chuck -> leer
```

```
* * * * * Add * * * * *
Ingrese el ID
10
Ingrese contenido
ANIADIENDO EJEMPLO OTRA VEZ
Se agrego ANIADIENDO EJEMPLO OTRA VEZ en documento id(10)
```

```
        * * * * * NICODOC id(10) * * * * *
HOLAEJEMPLOANIADIENDO EJEMPLOANIADIENDO EJEMPLO OTRA VEZ
Numero de versiones [3]
```

Rollback:

```
        * * * * * NICODOC id(10) * * * * *
HOLAEJEMPLO
Numero de versiones [3]
-> Permisos
        kim -> leer
        chuck -> leer
```

```
* * * * * Rollback * * * * *
Ingrese ID documento
10
Ingrese id version
0
HOLAEJEMPLO
```

RevokeAccess:

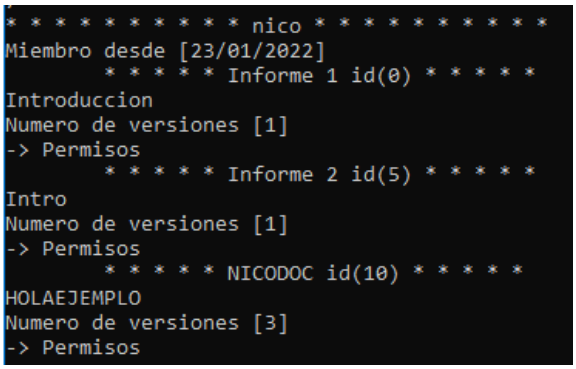
```
* * * * * Revoke Access * * * * *
Ingrese ID documento
10
Acccesos revocados del documento (10) Satisfactorio
```

```
        * * * * * NICODOC id(10) * * * * *
HOLAEJEMPLO
Numero de versiones [3]
-> Permisos
```

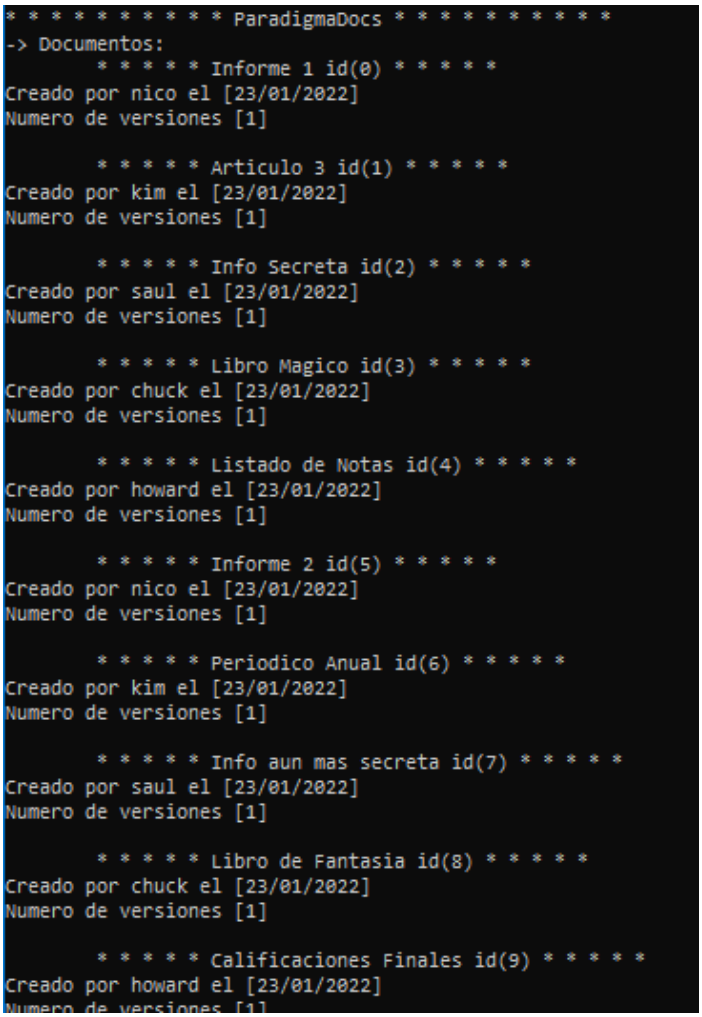
Search:

```
* * * * * Search * * * * *
Ingrese texto a buscar
HOLA
id(10) NICODOC
id(10) NICODOC
id(10) NICODOC
```

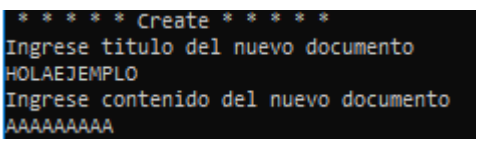
Visualize (logeado):



Visualize (No logeado):



Delete:



```
       * * * * * HOLAEJEMPLO id(10) * * * * *
AAAAAAAAA
Numero de versiones [1]
-> Permisos
```

```
* * * * * Delete * * * * *
Ingrese el ID
10
Ingrese el numero de caracteres a eliminar
5
Se eliminaron 5 caracteres, resultando: AAAA en documento id(10)
```

```
       * * * * * HOLAEJEMPLO id(10) * * * * *
AAAA
Numero de versiones [2]
```

Search And Replace:

```
* * * * * Search and Replace * * * * *
Ingrese ID documento
10
Ingrese el texto buscado
AAAA
Ingrese el texto a reemplazar
OOOOO
Operacion Exitosa
```

```
       * * * * * HOLAEJEMPLO id(10) * * * * *
OOOOO
Numero de versiones [3]
```

ApplyStyles:

```
* * * * * Apply Styles * * * * *
Ingrese ID documento
10
Ingrese el texto buscado
OO
Ingrese los estilos del tipo [#\i, #\u #\b] separados por espacio [_]
#\i #\u
Operacion Exitosa
```

```
       * * * * * HOLAEJEMPLO id(10) * * * * *
 [#\i#\u]OO[#\i#\u]  [#\i#\u]OO[#\i#\u] O
Numero de versiones [4]
-> Permisos
```

Comment:

```
* * * * * Comment * * * * *
Ingrese ID documento
10
Ingrese el texto a comentar
O
Ingrese el comentario
HOLACOMENTARIO
Operacion Exitosa
```

```
       * * * * * HOLAEJEMPLO id(10) * * * * *
 [#\i#\u]OO[#\i#\u]  [#\i#\u]OO[#\i#\u] O
Numero de versiones [5]
-> Permisos
 * * Comentario * *
       [nico] [23/01/2022]
       * O * -> [HOLACOMENTARIO]
```