

Clara Bujeda Muñoz

Tutora: Rosana Marín Berraondo

Centro: Salesianos Nuestra Señora del Pilar

Grado superior en aplicaciones multiplataforma

NinEIMGBOARD

Borrador Memoria TFG

**RESUMEN**

Un Image Board o tablón de imágenes llamado en español, es simplemente una aplicación web en la cual cualquier usuario puede publicar imágenes llamadas posts, asi como buscar diferentes imágenes como medio de inspiración.

El primer tablón de imágenes llamado 2channel surgió en 1999, el cual sigue activo en la actualidad nombrado como 5ch. los últimos tableros de imágenes fueron creados sobre 2013 los cuales usualmente usan tecnologías como php que actualmente son inseguras y desfasadas, en este TFG se desea realizar un sistema de tablón de imágenes con el uso de tecnologías actuales como por ejemplo SpringBoot y Angular para mejorar el rendimiento y seguridad de la aplicación web. Además de lo anterior se pretende mejorar el aspecto visual usando frameworks debido a que usualmente los tableros de imágenes se constituyen con css básico.

**INDICE**

Índice General

[1. Introducción](#Introducción) 6

[1.1.Contexto](#Contexto)6

[1.2.Objetivos](#Objetivos)7

[1.3.Motivación](#Motivación) 7

[1.4.Tecnologías usadas](#Tecnologías_usadas)8

[1.5.Organización de la memoria](#Organización_de_la_memoria)9

[2. Análisis del sistema](#Análisis_del_sistema) 10

[2.1.Sistema Inicial](#Sistema_Inicial)10

[2.2.Requisitos mínimos](#Requisitos_mínimos)11

[2.3. Identificación de los actores](#Identificación_de_los_actores)12

3. [Arquitectura del sistema](#Arquitectura_del_sistema)15

[3.1.Arquitectura Global](#Arquitectura_Global)15

[3.2.Diagrama estructural](#Diagrama_estructural)16

[3.2.1.General](#General)16

[3.2.2.Backend](#Backend)17

[3.2.3.Frontend](#Frontend)18

[3.2.4.BBDD](#BBDD)19

4. [Diseño e implementación del sistema](#Diseño_e_implementación_del_sistema) 22

[4.1.Datos y Estructura](#Datos_y_Estructura)22

[4.2.Modelo lógico](#Modelo_lógico)22

[4.3.Interfaces de usuario](#Interfaces_de_usuario)23

5. [Pruebas de funcionamiento](#Pruebas_de_funcionamiento)33

[5.1.Explicación](#Explicación)33

[5.2.Tabla de verificación](#Tabla_de_verificación)34

6. [Conclusión](#Conclusión)34

[6.1.Resultados Obtenidos](#Resultados_Obtenidos)35

[6.2.Análisis de trabajo](#Análisis_de_trabajo)36

[6.3.Mejoras Posibles](#Mejoras_Posibles)38

[6.4.Mercado](#Mercado)39

[6.5.Tiempo estimado](#Tiempo_estimado)41

[6.6.Valoración Personal](#Valoración_Personal)42

[7. Fuentes de Información](#Fuentes_de_Información)43

[7.1.Bibliografía](#Bibliografía)43

[7.2.Recursos](#Recursos)43

[7.3.Documentación](#Documentación)43

[8. Anexos](#Anexos)44

[8.1. Índice de terminología](#Indice_de_terminologia)44

[8.1. Ventana](#Ventanas_extras)46

[8.2. Resumen de](#Ventanas_extras) Código47

Índice de Figuras

[Figura 2.1.1 Diagrama inicial usando Thymeleaf como front y SpringBoot](#Figura_2_1_1_Diagrama_inicial_usando_Thy) 10

[Figura 2.1.2 Diagrama ideático usado para el proyecto final](#Figura_2_1_2_Diagrama_ideático_usado_par) 10

[Figura 3.1.1 Arquitectura global del sistema](#Figura_3_1_1_Arquitectura_global_del_sis) 15

[Figura 3.1.2 – Diagrama general de los servidores](#Figura_3_1_2__Diagrama_general_de_los_s) 16

[Figura 3.2.1 Diagrama de movimiento de datos entre capas](#Figura_3_2_1_Diagrama_de_movimiento_de_d) 17

[Figura 3.2.2 Diagrama de obtención de datos y muestreo](#Figura_3_2_2_Diagrama_de_obtención_de_da) 18

[Figura 3.2.3 Diagrama de la base de datos MySql](#Figura_3_2_3_Diagrama_de_la_base_de_dato) 20

[Figura 3.2.4 Diagrama de la base de datos SQLite](#Figura_4_3_4_Ejemplo_de_la_ventana_tags) 21

[Figura 4.2.1 Diagrama de flujo general de la aplicación](#Figura_4_2_1_Diagrama_de_flujo_general_d) 22

[Figura 4.3.1 Ejemplo estructural de la pantalla home](#Figura_4_3_1_Ejemplo_estructural_de_la_p) 23

[Figura 4.3.2 Ejemplo de la pestaña posts](#Figura_4_3_2_Ejemplo_de_la_pestaña_posts) 24

[Figura 4.3.3 Ejemplo de los detalles de un post](#Figura_4_3_3_Ejemplo_de_los_detalles_de_) 25

[Figura 4.3.4 Ejemplo de la ventana tags](#Figura_4_3_4_Ejemplo_de_la_ventana_tags) 26

[Figura 4.3.5 Login en la aplicación](#Figura_4_3_5_Login_en_la_aplicación) 27

[Figura 4.3.6 Registro en el sistema](#Figura_4_3_6_Registro_en_el_sistema) 28

[Figura 4.3.7 Página 404](#Figura_4_3_7_Página_404) 29

[Figura 4.3.8 Ventana Upload](#Figura_4_3_8_Ventana_Upload) 30

[Figura 4.3.9 Panel de usuario](#Figura_4_3_9_Panel_de_usuario)31

[Figura 4.3.10 Panel del sistema administrador](#Figura_4_3_9_Panel_del_sistema_administr) 32

[Figura 4.3.11 Panel de usuarios Idea general](#Figura_4_3_10_Panel_de_usuarios_Idea_gen) 32

Índice de Tablas

[Tabla 1.2.1 - Tabla de las tecnologías usadas](#Tabla_1_2_1___Tabla_de_las_tecnologías_u) 8

[Tabla 2.2.1 – Resumen de requisitos mínimos](#Tabla_2_2_1_Resumen_de_requisitos_míni) 11

[Tabla 5.2.1 – Pruebas Unitarias realizadas](#Tabla_5_2_1_Pruebas_Unitarias_realizad) 34

[Tabla 5.2.2 – Pruebas Visuales](#Tabla_5_2_2_Pruebas_Visuales) 34

[Tabla 6.3.1 – Mejoras posibles](#Tabla_6_3_1_Mejoras_posibles) 38

[Tabla 6.5.1 – Tiempo estimado](#Tabla_6_5_1_Tiempo_estimado) 41

[Tabla A.1 – Índice de terminología](#Tabla_A_1_Índice_de_terminología) 45

**Capítulo 1**

**Introducción**

Este documento tiene como objetivo brindar una explicación completa y detallada de la conformidad del proyecto NIB, así como de su estructura y los motivos que llevaron a su desarrollo. Para ello, se llevará a cabo un análisis exhaustivo del proyecto, que permitirá comprender de manera clara y concisa su alcance y objetivos.

Además, se abordarán aspectos específicos relacionados con la implementación y el desempeño del proyecto, incluyendo el diseño de la arquitectura, la metodología utilizada, la gestión de recursos y el control de calidad. También se describirá el proceso de evaluación de la conformidad del proyecto, así como las herramientas y tecnologías utilizadas para este fin.

En definitiva, este documento será una guía completa y detallada para comprender en profundidad la conformidad del proyecto NIB, y para valorar su relevancia y contribución en el ámbito en el que se desarrolla.

**1.1.** **Contexto**

Este proyecto se desarrolla de forma unipersonal en el centro escolar Salesianos Nuestra Señora del Pilar

**1.2.** **Objetivos**

Los objetivos en esencia de esta TFG es la experimentación de crear una aplicación web de tablón de imágenes usando tecnologías actuales para mejorar tanto el aspecto visual como el aspecto técnico, a continuación, se citan los objetivos generales que se espera conseguir con dicho proyecto.

1. Aprendizaje y funcionamiento de una web de tableros de imágenes asi como podría ser https://safebooru.org/ .
2. Mejorar diseño visual y estructura inspiradas además de modernizarlas.
3. Diseñar la aplicación en un sistema MVC para la facilidad de mantenimiento y actualización.
4. Estudiar un sistema de almacenamiento de datos MIME mediante guardado en BLOB en ficheros bd .
5. Estudiar un sistema para la separación del sistema de almacenamiento de binarios y datos textuales.

**1.3.** **Motivación**

Usualmente navegando por tableros de imágenes siempre me pregunte cuál era su funcionamiento, como comprimían los datos, como los enviaban o como almacenaban todo, en ese momento intente crear mi propio tablero de imágenes que surgió como un prototipo el cual solamente se podían subir imágenes y las guardaba en un sistema de ficheros, pero esto es simplemente ineficiente asi como afán de querer rehacer dicho proyecto me surgió la idea de volverlo a realizar en ese TFG.

**1.4.** **Tecnologías usadas**

En este proyecto se ha querido optar por el uso de nuevas y diferentes tecnologías

|  |  |  |
| --- | --- | --- |
| Tipo | Tecnología | Definición |
| Server Backend | SpringBoot | Server el cual gestionará la api que permitirá la salida y la entrada de datos. Además de esto tramitará a el servidor de datos y a los diferentes servidores binarios.  También tendrá la función del núcleo del programa. |
| Server Frontend | Angular | Server el cual gestionará el frontend de la web. |
| Server Datos | MySql | Server el cual almacenará y proveerá los datos de los usuarios además de información relacionadas con los posts y tags. |
| Servers Binarios | SQLite | Servers los cuales almacenarán los binarios de las imágenes de los usuarios para liberar tensión en el servidor MySql. |
| API S-A | API RESTful (JSON) | Método de comunicación entre el servidor SpringBoot y el Servidor Angular |
| Diseño | BootStrap | Framework con el cual se gestionará el diseño de la web (También se usará css puro para el funcionamiento correcto de la web) |
| Lanzadores | C# Windows Forms | Se uso C# Para la programación de un lanzador de la aplicación en un entorno visual o GUI para una mayor facilidad para la ejecución. |
| .BAT / CMD | Se escribió un script para el lanzamiento del programa en modo consola para un entorno de Windows. |
| .Sh / Bash | Se escribió un script para el lanzamiento del programa en modo consola para un entorno Linux. |

Tabla 1.2.1 - Tabla de las tecnologías usadas

**1.5.** **Organización de la memoria**

El primer capítulo de la memoria se ha dedicado a proporcionar una breve introducción y presentación del trabajo realizado. En el segundo capítulo se ha llevado a cabo un análisis exhaustivo que ha sido útil para establecer los objetivos y funcionalidades de la aplicación web. El tercer capítulo incluye una descripción detallada de la arquitectura general del proyecto, así como la arquitectura específica de la nueva aplicación web desarrollada. En el cuarto capítulo se detallan los pasos llevados a cabo en la segunda fase del proyecto, incluyendo la definición de los procesos involucrados, el modelo lógico de los datos y el diseño de la interfaz de usuario. Posteriormente en el quinto capítulo presenta todos los resultados obtenidos, proporciona una lista de posibles trabajos futuros y realiza una valoración personal sobre el trabajo realizado. En el capítulo sexto se proporciona una conclusión sobre los resultados finales de la aplicación y por último en el séptimo capítulo se muestra los diferentes recursos y fuentes de información usadas.

**Capítulo 2**

**Análisis del sistema**

**2.1.** **Sistema Inicial**

En un inicio el proyecto se ideo inicialmente con un servidor SpringBoot el cual almacenaba los datos en un Servidor MySql y los archivos binarios en una estructura de archivos local tal y como se muestra en la figura posterior.

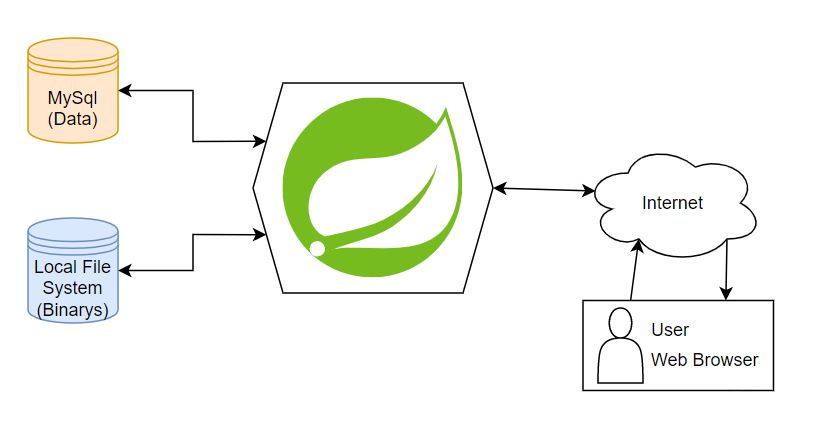
****

Figura 2.1.1 Diagrama inicial usando Thymeleaf como front y SpringBoot

Posteriormente al desear un mejor rendimiento tanto para el usuario como para la parte de datos y servidor se prefirió usar SpringBoot como servidor Backend y Angular para el servidor del front debido a que este tiene gran flexibilidad para la carga veloz de vistas webs, además de lo anterior se reemplazó el sistema de ficheros local con múltiples archivos SQLite debido a la facilidad de organización y mejora de rendimiento en cuanto a la búsqueda de los binarios tal y como se muestra en la siguiente figura.

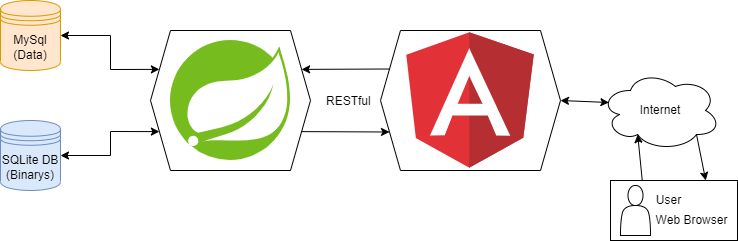


Figura 2.1.2 Diagrama ideático usado para el proyecto final

**2.2.** **Requisitos mínimos**

En este apartado se especificarán los requisitos mínimos necesarios para garantizar el correcto funcionamiento de la aplicación web en su servidor o computadora. Estos requisitos incluyen la versión mínima de los diferentes programas y otros requerimientos.

**Versión de software requerida:**

* Java: Se requiere la versión 17 de Java para ejecutar la aplicación web sin problemas. Asegúrese de tener instalada esta versión en su sistema.
* Node.js: La versión mínima de Node.js necesaria es la v18.12.0. Este entorno de tiempo de ejecución es esencial para el funcionamiento adecuado de la aplicación web. Verifique que tenga instalada esta versión en su servidor o computadora.
* npm package manager: Para gestionar las dependencias de la aplicación web, se necesita el administrador de paquetes npm en su versión v8.19 o superior. Asegúrese de contar con esta versión instalada en su sistema.
* MySQL: Se requiere la versión 8 de MySQL como sistema de gestión de bases de datos para la aplicación web. Verifique que tenga instalada esta versión y que esté correctamente configurada en su servidor o computadora.

**Sistema operativo compatible:**

* Windows 10 (+): Si su servidor o computadora utiliza el sistema operativo Windows, se recomienda que tenga instalada la versión 10 o superior para asegurar la compatibilidad y el rendimiento óptimo de la aplicación web.
* Linux: Si prefiere utilizar Linux como sistema operativo, asegúrese de tener una distribución compatible y actualizada para garantizar el correcto funcionamiento de la aplicación web aunque actualmente no se puede asegurar el correcto funcionamiento.

|  |  |
| --- | --- |
|  | Versión |
| Java | 17 |
| Node js | v18.12.0 |
| Mysql | 8 |
| npm package manager | v8.19 |
| SO | Windows 10 (+) / Linux |

Tabla 2.2.1 – Resumen de requisitos mínimos

**2.3.** **Identificación de los actores**

En este apartado se expresa los diferentes tipos de usuarios y las acciones que pueden y no pueden hacer además de las páginas que contienen dichas funciones.

**Usuario No registrado**

Un usuario en la aplicación podrá registrarse o no, en las siguientes tablas se muestra la identificación de los actores dependiendo de dichas causas.

**¿Qué puede hacer?**

Un Usuario al conectarse a la aplicación web vera una pantalla principal en la cual podrá dirigirse a posts, logearse (login), registrarse (register), ver los tags o realizar una búsqueda.

Búsqueda

* En la pantalla principal al efectuar una búsqueda se le redirigirá al usuario a la ventana de posts mostrándole la búsqueda efectuada.

Posts:

* Al entrar en el apartado de posts el usuario podrá ver el escenario dividido en dos secciones en el primero en el cual podrá realizar una búsqueda y el segundo en el cual podrá visualizar dicha búsqueda, en caso de que el usuario no hubiera realizado una búsqueda, se le mostraran por default los posts más nuevos.
* En caso de clicar en un post se le redirigirá al usuario a una ventana de detalles del post en la cual podrá ver tanto la imagen como sus tags y su información específica.

Login

* En caso de entrar al apartado de login se le solicitará al usuario su nombre de usuario además de su contraseña, una vez hecho esto le redirigirá de forma automática a la pantalla principal de un usuario registrado.

Register

* Al acceder en al apartado de register se solicitará al actor su nombre de usuario además de su email y la contraseña, lo anterior de forma obligatoria, de forma opcional se le solicitará su nombre y sus apellidos.

Tags

En dicha ventana se le mostrará al usuario los tags disponibles para ejercer la búsqueda.

**Usuario Registrado**

**¿Qué puede hacer?**

Un Usuario registrado será capaz de ver una pantalla principal en la cual podrá dirigirse a posts, Panel de usuario, Upload o la vista de los tags además de ser capaz de realizar una búsqueda.

Búsqueda

* En la pantalla principal al efectuar una búsqueda se le redirigirá al usuario a la ventana de posts mostrándole la búsqueda efectuada.

Posts:

* Al entrar en el apartado de posts el usuario podrá ver el escenario dividido en dos secciones en el primero en el cual podrá realizar una búsqueda y el segundo en el cual podrá visualizar dicha búsqueda, en caso de que el usuario no hubiera realizado una búsqueda, se le mostraran por default los posts más nuevos.
* En caso de clicar en un post se le redirigirá al usuario a una ventana de detalles del post en la cual podrá ver tanto la imagen como sus tags y su información específica además de poder comentar dicho post.

Panel de Usuario

* En dicha ventana el usuario deberá de ser capaz de modificar su configuración de usuario asi como personalizar sus datos.

Upload

* La función de "Upload" es una herramienta esencial en cualquier aplicación web que permita a los usuarios compartir contenido con la comunidad. La ventana de "Upload" es un espacio donde los usuarios registrados pueden crear y subir un nuevo post.
* Esta función es especialmente útil para aquellos que quieren compartir sus ideas, experiencias o conocimientos con el resto del mundo. Con la posibilidad de agregar etiquetas, títulos y descripciones, el usuario puede personalizar su post con tags para que sea más fácilmente encontrado por otros usuarios interesados en el mismo tema.

Tags

* En la ventana de tags, el usuario puede encontrar una lista de palabras clave que se utilizan para categorizar y etiquetar los recursos en la plataforma. Estos tags están diseñados para ayudar al usuario a encontrar posts que mejor se adapten a sus.

**Administrador**

Un administrador es aquel usuario el cual tendrá acceso a las funciones de control total de la aplicación.

**¿Qué puede hacer?**

Un administrador debe de ser capaz de efectuar lo mismo que un usuario además de tener la capacidad de eliminar usuarios, eliminar posts y modificar la configuración asimismo de tener la capacidad de retocar los atributos del mismo.

Panel de administrador

* En dicho panel el Administrador será capaz de modificar los datos del sitio web asi como su configuración.
* Además de lo anterior en dicho panel será capaz de banear usuarios, una vez baneado un usuario constará como un usuario anónimo y no modificable.

**Capítulo 3**

**Arquitectura del sistema**

En este capítulo se describe la arquitectura del sistema mencionado.

**3.1.** **Arquitectura Global**

El programa es capaz de correr sus partes tanto en el mismo servidor como en diferentes servidores tal y como se indica en la figura posterior.

La arquitectura en su esencia consta en 3 partes: BackEnd y Datos MIME, FrontEnd y por último datos de la aplicación. Todo esto esta maquetado con la estructura Modelo vista controlador (MVC) lo cual consta en que los datos, lo que controla los datos y lo que ve el usuario este estructurado de forma separada, con lo que conseguimos un mayor control en la estructuración del programa, mayor facilidad de actualización, arreglo de bugs y errores además de mayor seguridad en el sistema.

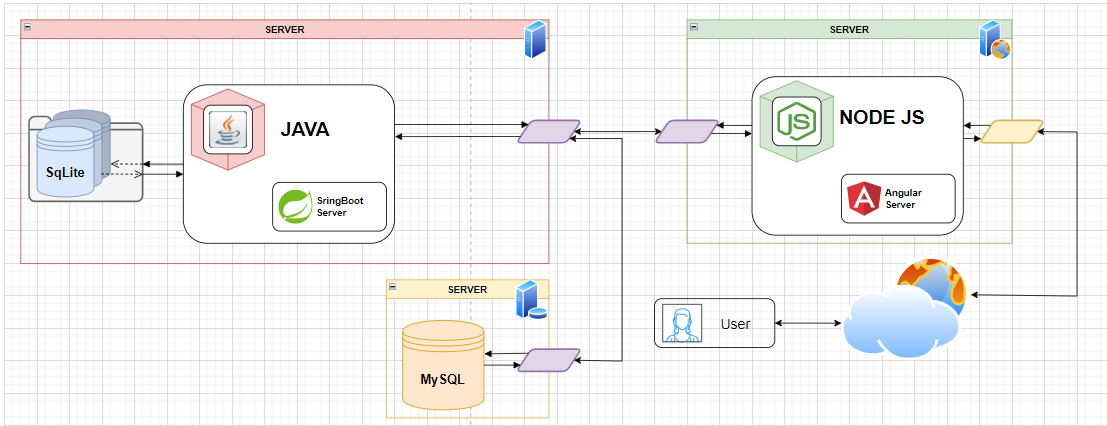


Figura 3.1.1 Arquitectura global del sistema

**3.2.** **Diagrama estructural**

En esta sección se presenta una representación gráfica de la estructura general del proyecto. El diagrama proporciona una visión detallada de las relaciones entre los diferentes componentes y módulos del sistema, así como su jerarquía y dependencias.

**3.2.1.** **General**

En esta sección se muestra la estructura general de la aplicación web la cual está compuesta por diferentes partes.

* Datos: Dicha parte almacena los diferentes datos la cual consta de dos bases de datos, la primera MySql la cual almacenara datos textuales y la segunda SQLite la cual almacenara principalmente datos de tipo BLOB.
* BackEnd: Esta sección de la aplicación se encarga del procesamiento de los datos y de proveerlos al frontend en caso de ser necesario, esto anterior esta desarrollado en SpringBoot debido al ser capaz de soportar más carga.
* FrontEnd: Esta área de la aplicación muestra los datos al usuario que han sido proveídos por el backend.

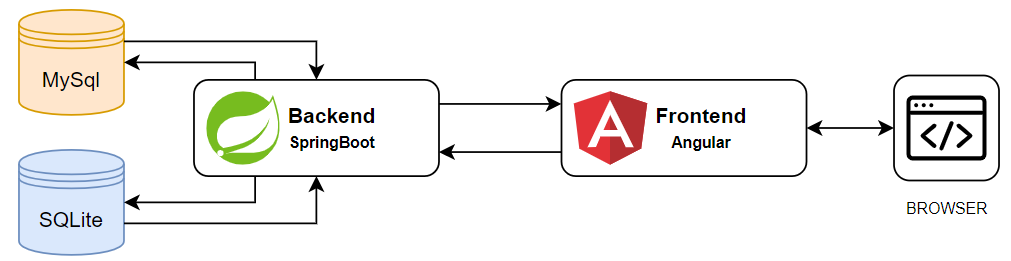
****

Figura 3.1.2 – Diagrama general de los servidores

**3.2.2.** **Backend**

El BackEnd de este proyecto ha sido desarrollado en SpringBoot.

Se ha elegido esta tecnología por varias razones, entre las que destaca su capacidad para facilitar la creación de aplicaciones robustas, escalables y seguras.

La estructura del backend ha sido subdividida en diferentes secciones. Los controladores los cuales manejan el uso de los datos proveyendo una api, los servicios e implementaciones los cuales manejan los datos que usan los controladores y por último la dao la cual usa JPA (CRUD) para comunicarse con las diferentes bases de datos usadas.

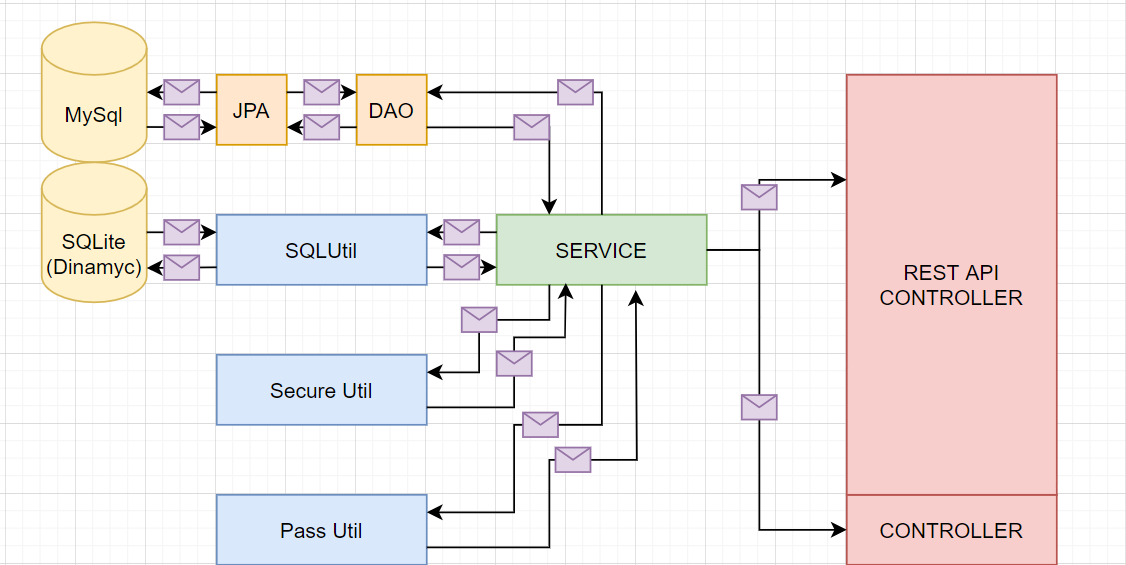
****

Figura 3.2.1 Diagrama de movimiento de datos entre capas

**3.2.3.** **Frontend**

El FrontEnd de este proyecto se ha realizado en Angular debido a sus numerosas ventajas y funcionalidades avanzadas que lo convierten en una herramienta ideal para el desarrollo de interfaces de usuario modernas y complejas.

Para la realización de dicha sección se ha usado los siguientes elementos propios del framework. Componentes, los cuales almacenan paginas ya sea para usarlas como layouts o como una ventana de la web, y servicios los cuales obtienen los datos, los procesan y los usan los componentes, además de dichos elementos propios, existen otros como archivos de configuración y diferentes scripts Ts.

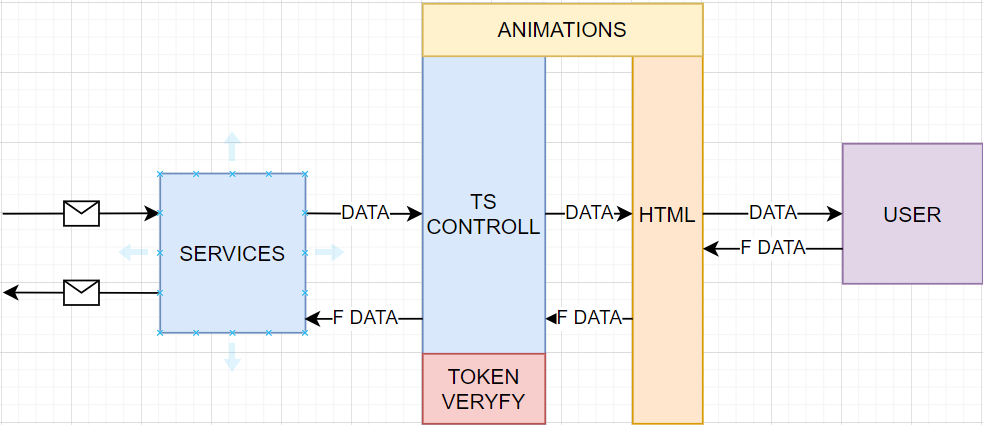
****

Figura 3.2.2 Diagrama de obtención de datos y muestreo.

**3.2.4.** **BBDD**

**MySql**

En la base de datos existen las siguientes tablas: usuario, post, tags, roles, comments, insult list. Cada una de estas tablas desempeña un papel fundamental en el funcionamiento y la organización de la base de datos.

La tabla "usuario" almacena la información de los usuarios registrados en el sistema. Aquí se guardan los datos como el nombre, dirección de correo electrónico, contraseña y otra información relevante que identifica a cada usuario de manera única.

La tabla "post" se utiliza para almacenar las publicaciones realizadas por los usuarios. Cada entrada en esta tabla contiene detalles sobre el contenido del post, como el título, el cuerpo del texto, la fecha de publicación y cualquier otro dato asociado.

La tabla "tags" permite categorizar los posts en función de palabras clave o etiquetas. Los tags proporcionan una forma eficiente de organizar y buscar contenido relacionado en la base de datos. Cada tag tiene una relación con uno o varios posts, lo que facilita la clasificación y el filtrado de información.

La tabla "roles" se utiliza para asignar diferentes niveles de permisos y privilegios a los usuarios. Por ejemplo, un usuario puede tener el rol de administrador, lo que le otorga acceso a funciones y acciones adicionales en el sistema. Los roles ayudan a gestionar la seguridad y la autorización en la base de datos.

La tabla "comments" se encarga de almacenar los comentarios realizados por los usuarios en los posts. Aquí se guardan los datos relacionados con cada comentario, como el autor, el contenido del comentario, la fecha y cualquier otro atributo relevante.

Por último, la tabla "insult list" se utiliza para almacenar una lista de insultos o palabras ofensivas que deben ser filtradas o moderadas en los comentarios o posts. Esta lista ayuda a mantener un ambiente respetuoso y seguro en la plataforma, evitando el contenido inapropiado.

En resumen, estas tablas en la base de datos sirven actualmente para organizar y gestionar información sobre los usuarios, publicaciones, etiquetas, roles, comentarios y controlar el contenido ofensivo, proporcionando una estructura sólida y eficiente para el funcionamiento del sistema.

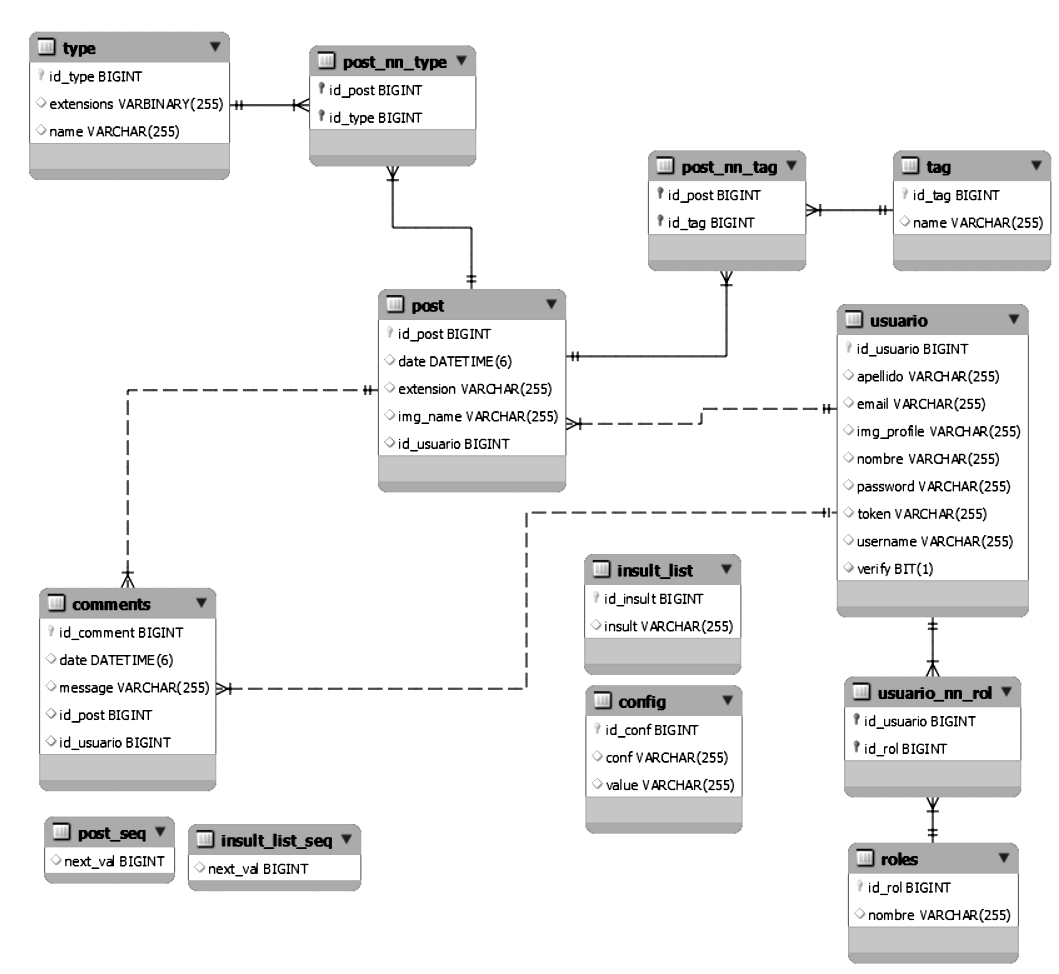
****

Figura 3.2.3 Diagrama de la base de datos MySql

**SQLite**

Para lograr una eficiente gestión de datos, se ha implementado la creación de una base de datos de LiteSql para cada usuario en nuestro sistema. Estas bases de datos individuales almacenan información clave que nos permite organizar y acceder de manera efectiva a los datos relacionados con cada usuario.

En cada una de estas bases de datos, se guardan tres elementos fundamentales: el identificador externo del post (iddb\_binary), el binario asociado y el identificador del sistema. Estos datos nos permiten realizar un seguimiento preciso de la información relacionada con cada usuario y asegurarnos de que los datos estén correctamente enlazados y disponibles para su consulta.

Cada base de datos individual se nombra siguiendo una convención específica. El nombre de archivo se compone del "idw" del usuario, que es un identificador único asignado a cada usuario en nuestro sistema, seguido de la extensión ".db". Esta estructura de nomenclatura nos ayuda a mantener un orden claro y organizado, facilitando la identificación de las bases de datos correspondientes a cada usuario.

Con esta implementación, podemos asegurar una gestión eficiente de la información, manteniendo la integridad de los datos y proporcionando un acceso rápido y preciso a los mismos. Cada usuario tiene su propio espacio individualizado dentro de la base de datos, lo que nos permite ofrecer un servicio personalizado y adaptado a las necesidades de cada usuario en nuestro sistema.

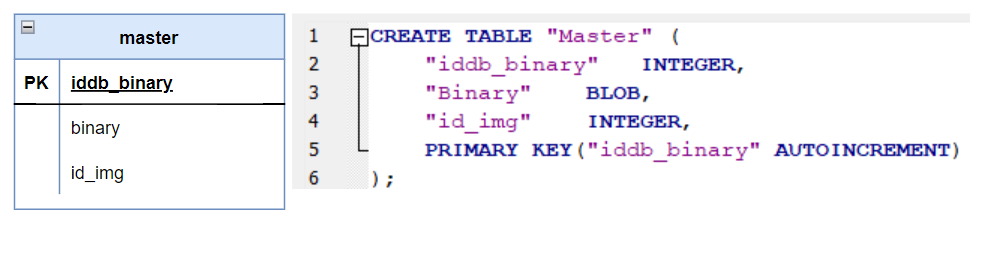


Figura 3.2.4 Diagrama de la base de datos SQLite

**Capítulo 4**

**Diseño e implementación del sistema**

**4.1.** **Datos y Estructura**

La aplicación cuenta con varias tablas, cada una con un propósito específico y un conjunto de datos asociados. En estas tablas se almacenan diferentes tipos de información que son necesarios para el correcto funcionamiento de la aplicación.

**4.2.** **Modelo lógico**

La aplicación web en general se compone en diferentes partes, debido a esto el ejecutable iniciara primero un configurador el cual establecerá las preferencias del sistema, y a posteriori iniciara los servidores en diferentes Threads esto en caso de que el inicio sea conjunto.

El Backend una vez iniciado entrara en modo de escucha para recibir diferentes peticiones y asi a posteriori responderlas.

El Frontend al iniciar esperara hasta que un usuario se conecte y a continuación solicitara los datos al backend y mostrara una página con los datos solicitados.

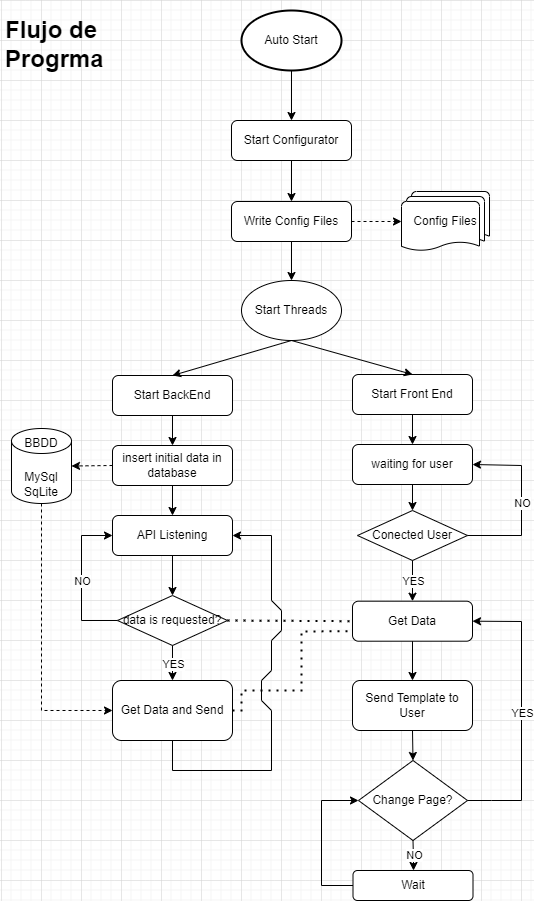


Figura 4.2.1 Diagrama de flujo general de la aplicación.

**4.3.** **Interfaces de usuario**

**Interfaz Publica**

La interfaz de usuario publica consiste en las siguientes ventanas: Home, Posts, Tags y los diferentes formularios de inicio de sesión y registro.

**Home**

En la Pantalla **Home** podremos efectuar una búsqueda o navegar por las diferentes pantallas permitidas al usuario como se han indicado anteriormente, A continuación, en la siguiente figura se muestra dicha ventana.

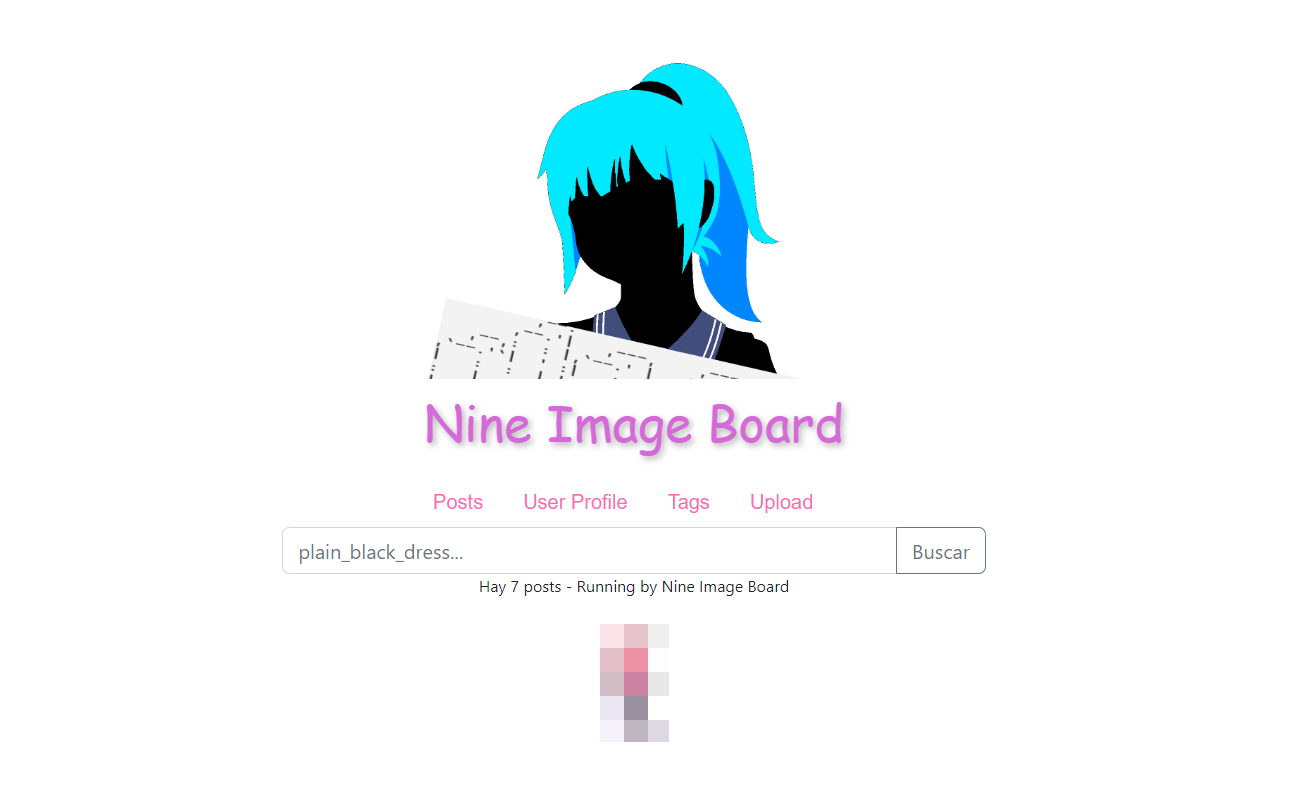


Figura 4.3.1 Ejemplo estructural de la pantalla home.

**Posts**

Desde la anterior ventana el usuario (no registrado) podrá acceder a Posts o Realizar una búsqueda en la Search bar tal y como se muestra en la anterior figura, una vez accedido a algunas de las siguientes opciones se mostrara al usuario la ventana de posts que la cual en caso de haber accedido sin realizar una búsqueda pertinente se mostrara al usuario la ventana posts con los últimos agregados, por el contrario de haber realizado una búsqueda se mostrarán dichos posts según la búsqueda tal y como se puede observar en la siguiente figura.

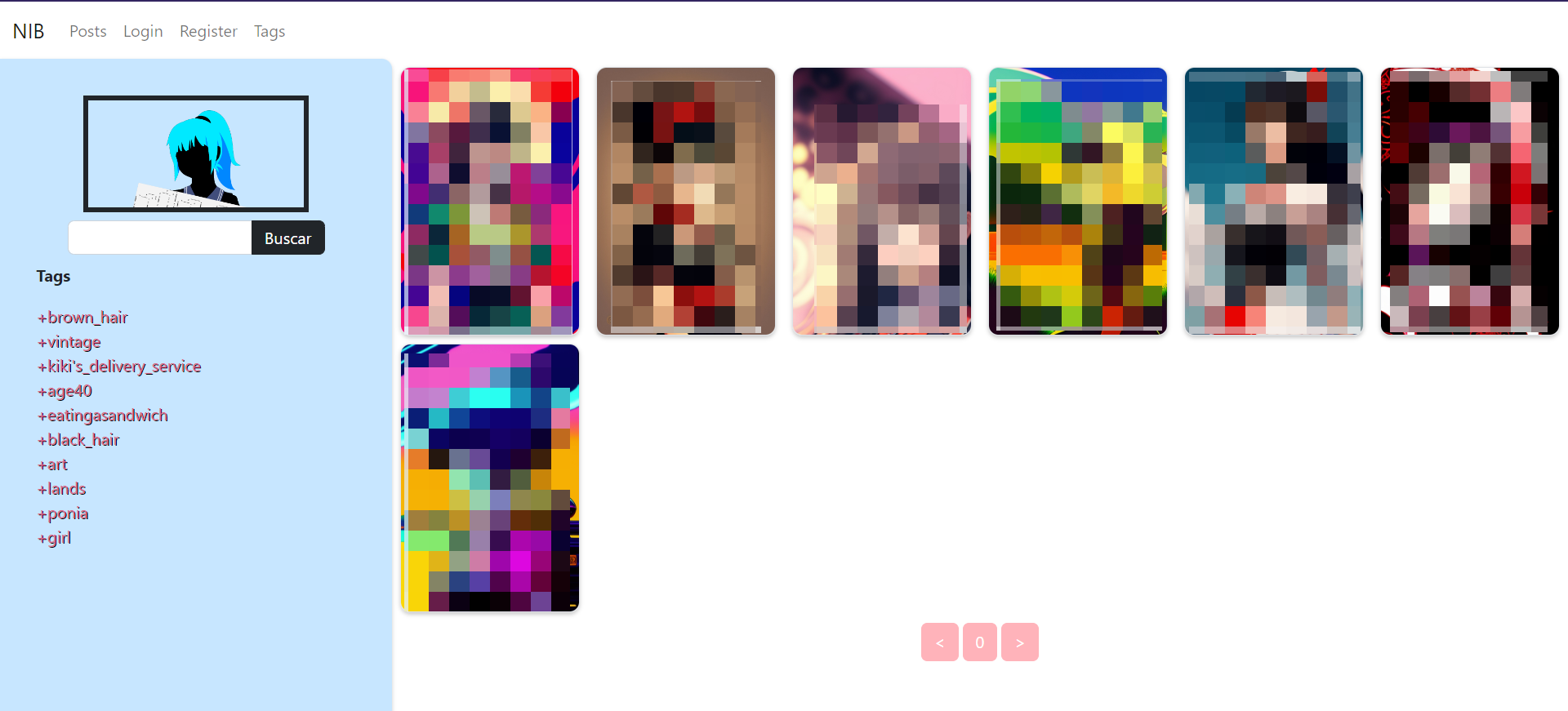
****

Figura 4.3.2 Ejemplo de la pestaña posts

**Detalles Posts**

En la ventana "Detalles de posts", el usuario será capaz de ver información más detallada sobre un post en particular. Esta ventana suele incluir información adicional, como el título del post, la fecha de publicación, el autor, las etiquetas asociadas, el contenido del post y los comentarios relacionados.

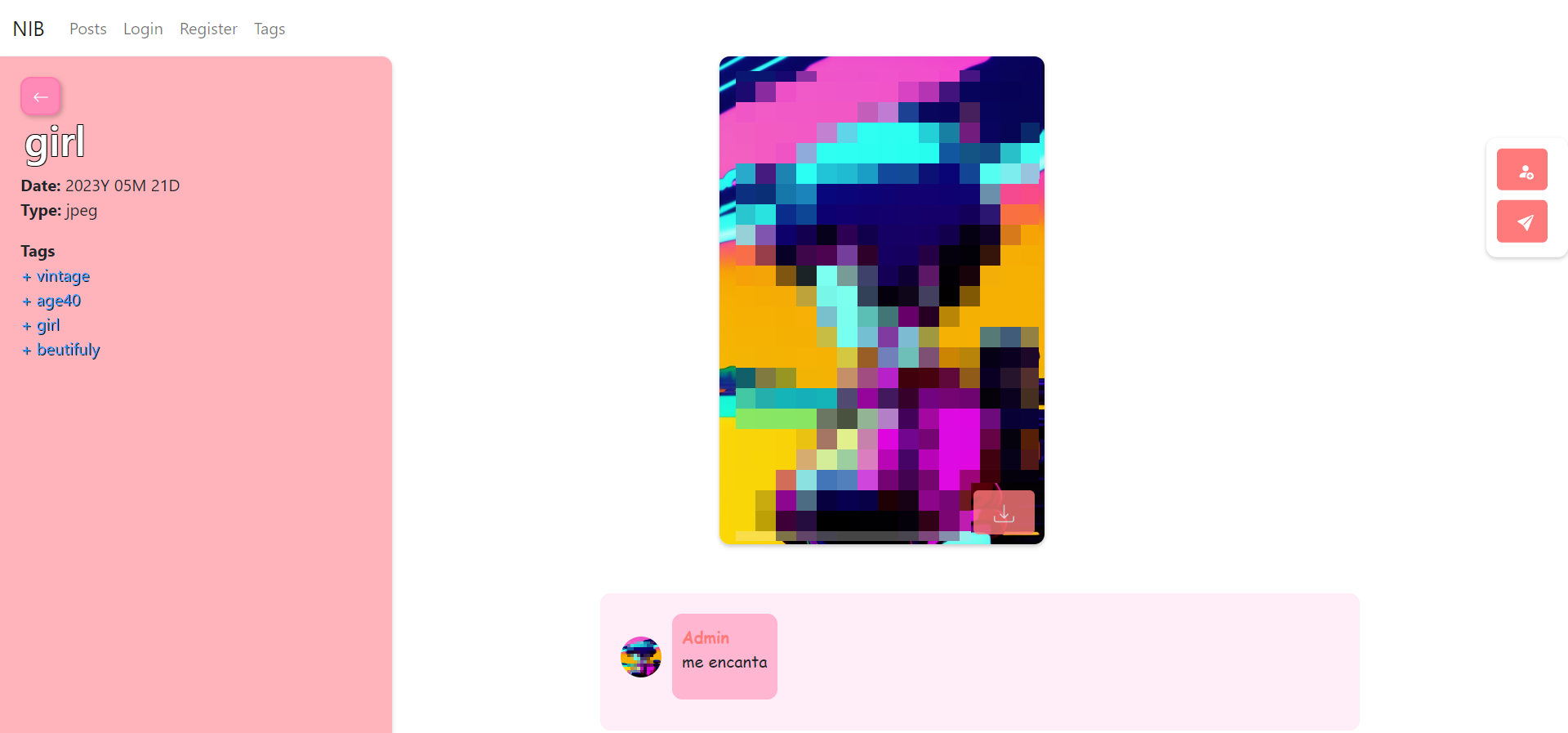
****

Figura 4.3.3 Ejemplo de los detalles de un post

**Tags**

En la pestaña "Tags" del sistema, el usuario será capaz de visualizar las diferentes etiquetas o "tags" asociadas a los posts o publicaciones de la plataforma. Estas etiquetas son palabras o frases que se utilizan para clasificar y agrupar el contenido de la plataforma en función de su temática o contenido.

En la pestaña "Tags", el usuario puede explorar las diferentes etiquetas disponibles y seleccionar aquellas que le interesen para visualizar los posts relacionados.

****

Figura 4.3.4 Ejemplo de la ventana tags

**Login**

En la ventana “**Login**” del sistema, el usuario es capaz de ingresar a su cuenta previamente registrada en la plataforma. Para poder acceder, se le solicita al usuario que ingrese su dirección de correo electrónico y su contraseña, los cuales son validados por el sistema antes de permitir el acceso a la cuenta.

Una vez que el usuario ha ingresado a su cuenta, tiene acceso a todas las funcionalidades y herramientas disponibles en la plataforma.

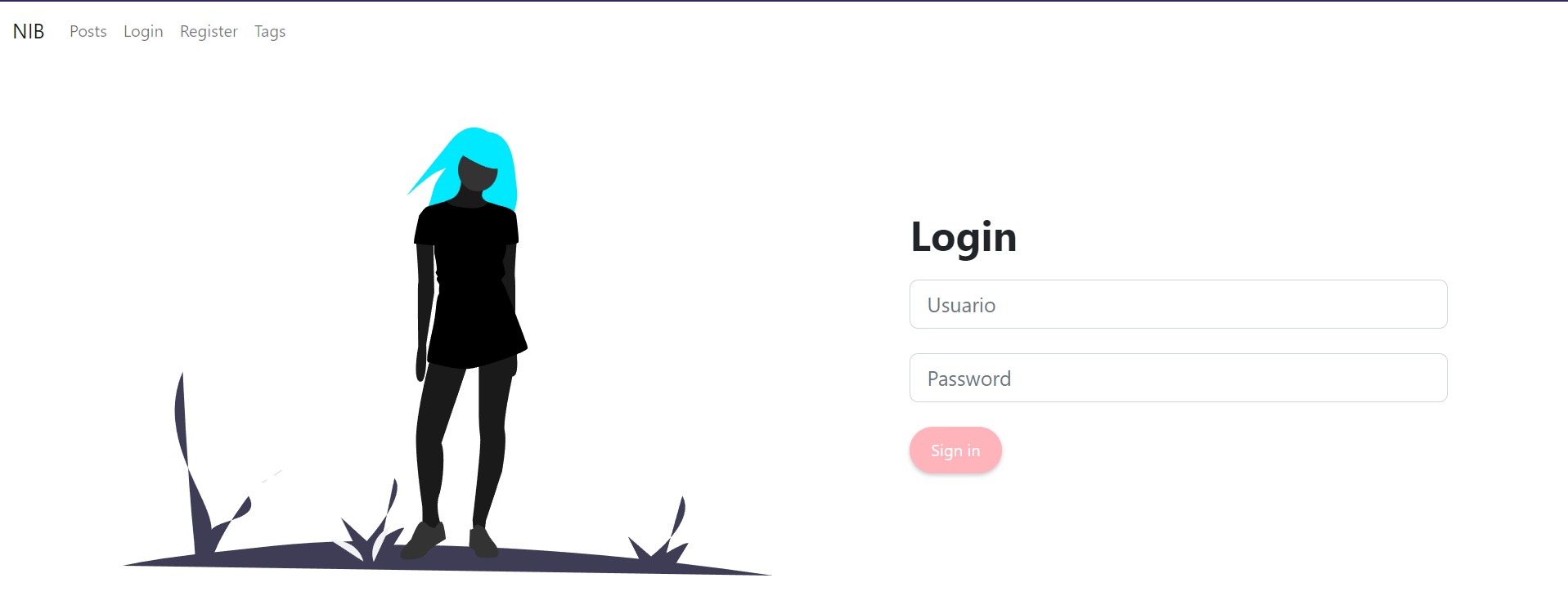
****

Figura 4.3.5 Login en la aplicación

**Register**

En la ventana “**register**” del sistema, el usuario es capaz de realizar el proceso de registro para crear una cuenta en la plataforma. En esta ventana, se le solicita al usuario que ingrese cierta información personal, como su nombre completo (opcional), dirección de correo electrónico, y una contraseña segura.

Una vez que el usuario ha proporcionado esta información, el sistema valida que los datos ingresados cumplan con los requisitos establecidos y, en caso de ser correctos, procede a registrar la cuenta del usuario. Es importante destacar que el proceso de registro se realiza de manera segura, utilizando medidas de seguridad y encriptación para proteger la información personal del usuario.

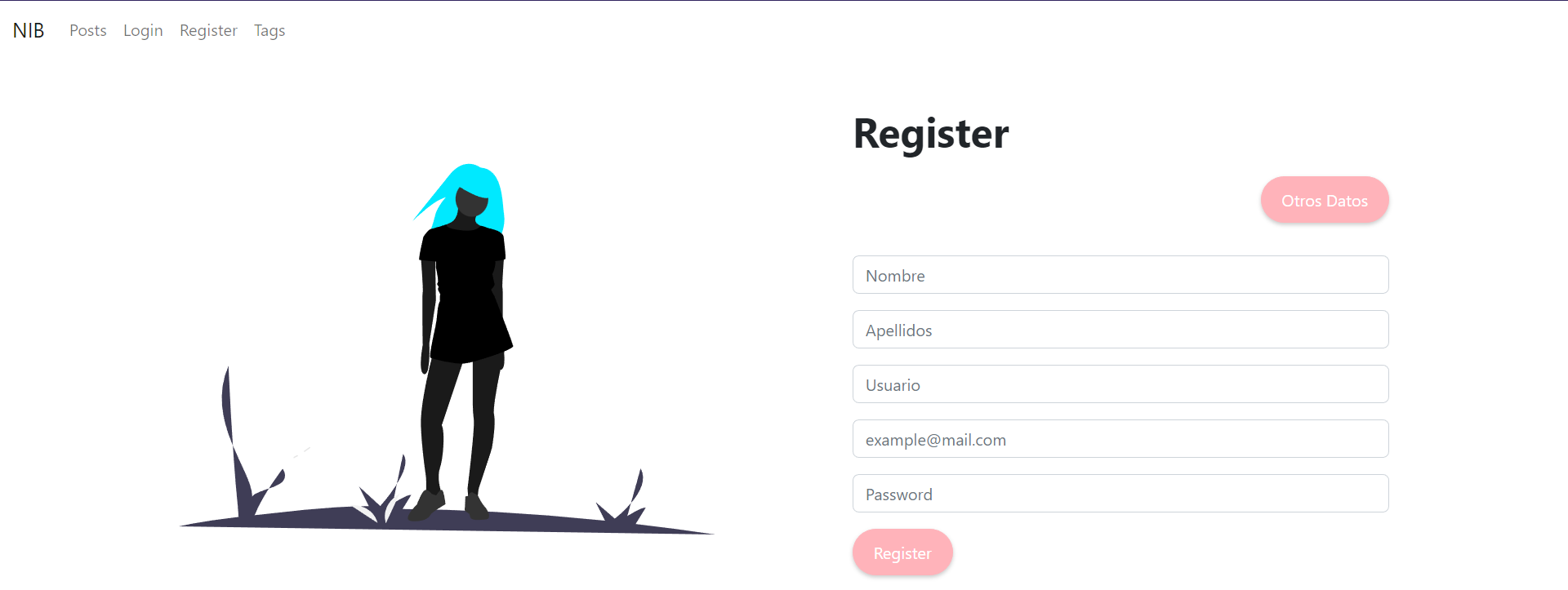


Figura 4.3.6 Registro en el sistema

**Error 404**

La pestaña "Error 404" del sistema es una página de error que se muestra cuando el usuario intenta acceder a una página que no existe en la plataforma. En esta sección, el usuario será capaz de visualizar un mensaje de error indicando que la página solicitada no se encuentra disponible.

Es una herramienta muy útil para mejorar la experiencia del usuario en la plataforma, ya que permite informar al usuario de manera clara y concisa que la página solicitada no existe, y evita confusiones o frustraciones por la falta de información.

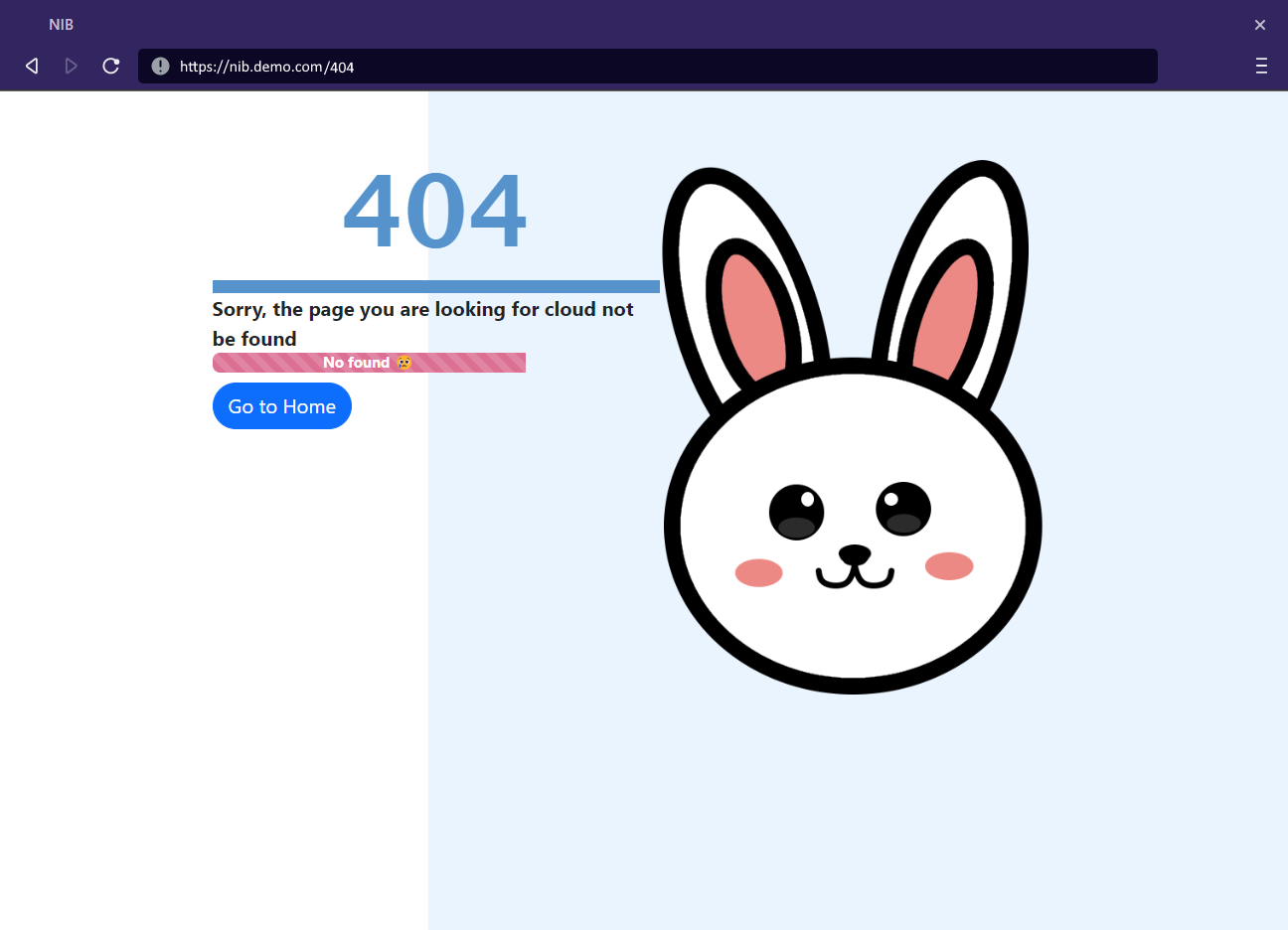
****

Figura 4.3.7 Página 404

A parte de lo anterior existen algunas páginas ocultas tal y como se describe en anexos apartado 2.2

**Interfaz Usuario Registrado**

La interfaz de un usuario es ligeramente diferente debido a que se muestra ligeramente diferente a la interfaz publica, ya que no contiene las opciones de login y registro y en vez de ello contiene las opciones de Upload y panel de usuario además de que en la pantalla de los detalles de posts el usuario es capaz de comentar dichas publicaciones.

**Upload**

En la ventana "Upload" el usuario tendrá la capacidad de subir posts con diferentes tags asociados, los cuales permitirán categorizar y organizar los posts de manera efectiva. Para esto, se incluirá un formulario donde el usuario podrá ingresar el título del post, su contenido y una lista de tags separados por comas.

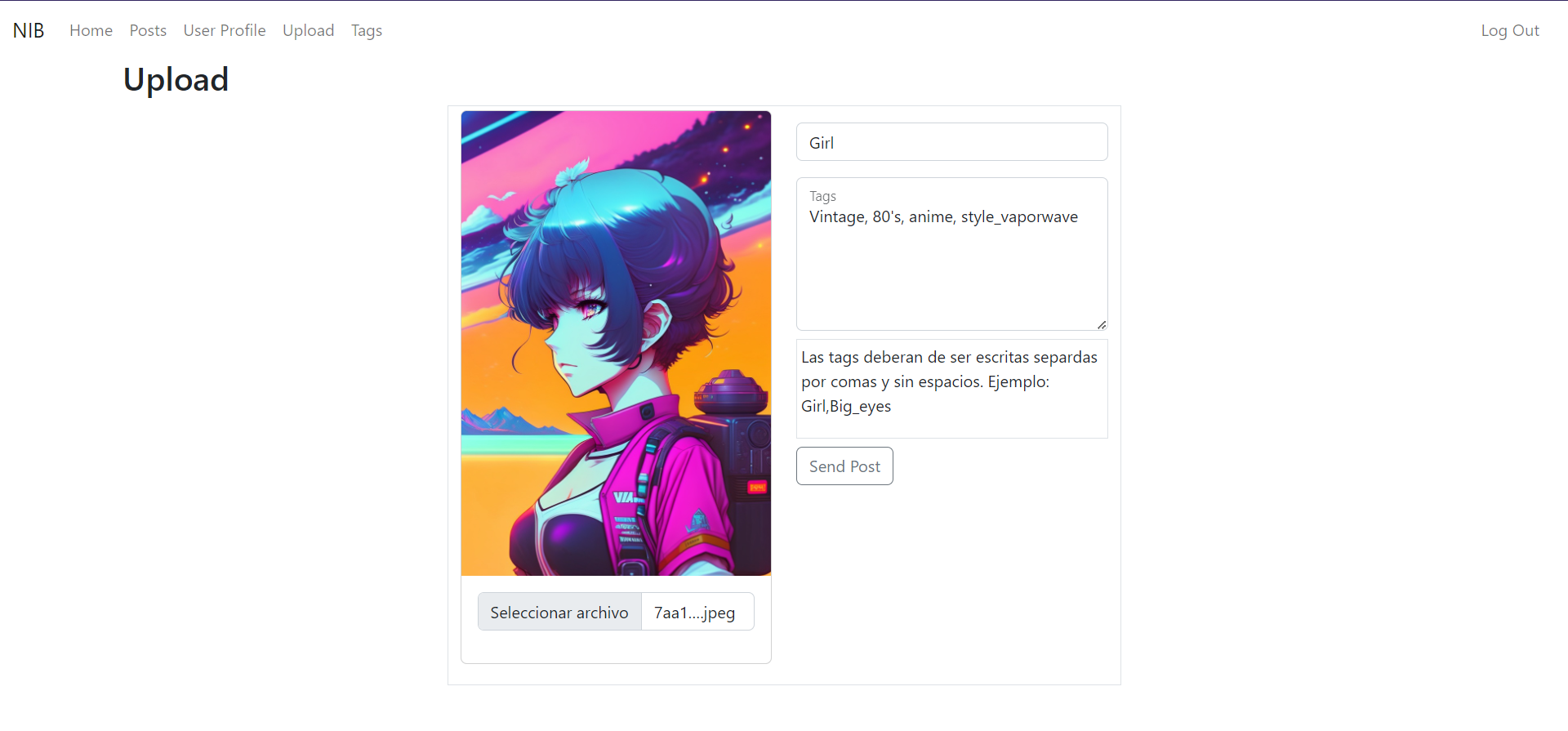
****

Figura 4.3.8 Ventana Upload

**Panel de usuario**

En la ventana "Panel de usuario" el usuario tendrá acceso a diferentes opciones y herramientas para administrar su cuenta. Una de las opciones principales será la posibilidad de editar y actualizar su perfil, incluyendo información personal como su nombre, correo electrónico, imagen de perfil y contraseña.

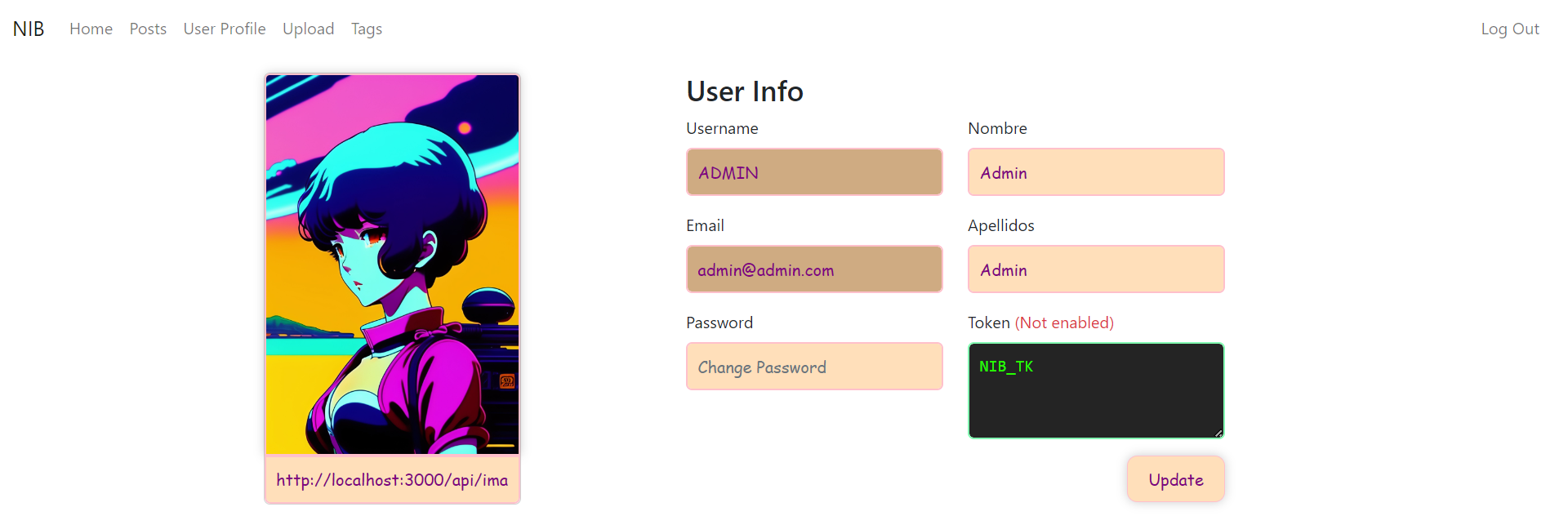
****

Figura 4.3.9 Panel de usuario

**Administrador**

La interfaz de un usuario administrador es igual que la de un usuario registrado salvo que tiene un panel en el cual puede modificar los ajustes de la página web asi como Panel de usuarios y funciones extras en algunas de las ventanas.

**Panel del Sistema**

En el panel del sistema, el administrador tendrá acceso a una serie de configuraciones avanzadas que le permitirán personalizar y ajustar diversos aspectos de la plataforma según las necesidades de la organización.

Entre las configuraciones disponibles, se incluyen opciones para modificar el nombre de la web asi como configuraciones internas del sistema.

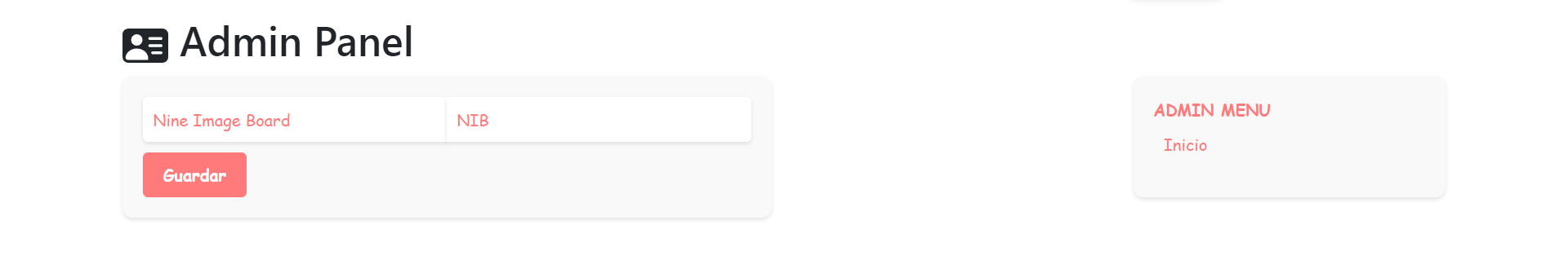
****

Figura 4.3.10 Panel del sistema administrador

**Panel de usuarios**

En la ventana "Panel de usuarios", un administrador tendrá acceso a opciones adicionales de administración que no estarán disponibles para los usuarios regulares. Entre ellas, se incluirá la posibilidad de banear y eliminar a usuarios de la plataforma.

Cuando un usuario es baneado por un administrador, su cuenta se desactivará temporalmente y se le impedirá acceder a la plataforma.

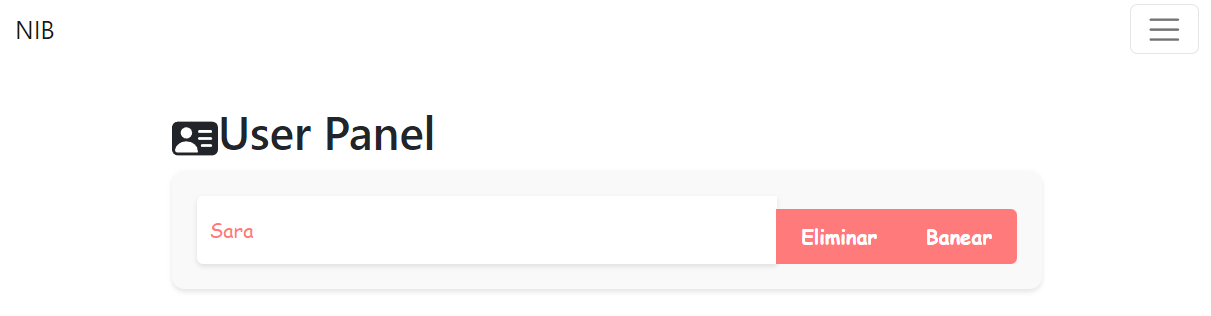
****

Figura 4.3.11 Panel de usuarios Idea general (no aplicada)

**Capítulo 5**

**Pruebas de funcionamiento**

**5.1.** **Explicación**

Para garantizar el correcto funcionamiento de la aplicación, se llevarán a cabo dos tipos de pruebas: pruebas unitarias y pruebas visuales.

Las pruebas unitarias se realizarán utilizando ng Test y JUnit. Estas pruebas se enfocarán en evaluar el funcionamiento individual de cada componente de la aplicación. Mediante ng Test, se verificará que cada unidad de código, como funciones o métodos, produce los resultados esperados y maneja correctamente los casos límite y las situaciones de error. Por su parte, JUnit permitirá realizar pruebas específicas en el código Java, asegurando la calidad de las funcionalidades implementadas.

Además de las pruebas unitarias, se llevarán a cabo pruebas visuales durante el desarrollo de la aplicación. Estas pruebas se centran en evaluar la interfaz de usuario y la experiencia del usuario en general. Se comprobará que los elementos visuales se muestran correctamente, que la navegación es intuitiva y que las interacciones responden de manera adecuada.

Al combinar las pruebas unitarias y las pruebas visuales, se busca obtener un sistema robusto y confiable. Las pruebas unitarias garantizan la calidad del código subyacente, mientras que las pruebas visuales aseguran una experiencia de usuario fluida y satisfactoria. Ambos tipos de pruebas son fundamentales para detectar y corregir posibles errores antes de que la aplicación sea implementada y puesta en manos de los usuarios finales.

**5.2.** **Tabla de verificación**

►Pruebas Unitarias

|  |  |
| --- | --- |
| Test de Apis | ✔️ |
| Test de filtrado correcto | ✔️ |
| Test de Token | ✔️ |
| Test de Transmisión de binarios | ✔️ |
| Test de almacenamiento BLOB | ✔️ |
| Test de autenticación | ✔️ |
| Test de encriptación de datos | ✔️ |
| Test de rendimiento | ✔️ |
| Test de manejo de errores | ✔️ |
| Test de escalabilidad | ✔️ |

Tabla 5.2.1 – Pruebas Unitarias realizadas

►Pruebas Visuales

|  |  |
| --- | --- |
| Función |  |
| Api count de posts | ✔️ |
| Api Tags | ✔️ |
| Api Posts | ✔️ |
| Api auth token | ✔️ |
| Api User | ✔️ |
| Pagina Home | ✔️ |
| Pagina User Profile | ✔️ |
| Pagina Admin Panel | ✔️ |
| Página 404 | ✔️ |
| Página de créditos (No finalizada) | ✖️ |
| Página Tags | ✔️ |
| Pagina Posts | ✔️ |
| Pagina Upload | ✔️ |

Tabla 5.2.2 – Pruebas Visuales

**Capítulo 6**

**Conclusión**

En la sección final de esta documentación, se presentarán las conclusiones que se han obtenido durante el desarrollo del Trabajo de Fin de Grado, así como las posibles tareas que se pueden llevar a cabo en el futuro para mejorar el sistema. Además, se proporcionará información sobre la cantidad de tiempo que se ha dedicado al proyecto y se realizará una evaluación personal del trabajo realizado.

**6.1.****Resultados Obtenidos**

Hasta la fecha, los resultados obtenidos en este proyecto han sido realmente prometedores. A pesar de enfrentarnos a limitaciones de tiempo y a la complejidad inherente del proyecto, hemos logrado alcanzar avances significativos en múltiples áreas.

Sin embargo, es importante destacar que debido a las restricciones de tiempo y a la complejidad que implica este proyecto, no hemos podido completar algunas secciones específicas. Estas secciones pendientes representan un desafío adicional que requerirá de una atención cuidadosa en el futuro.

Lo más destacado del proyecto en general es el sistema de binarios, que ha sido implementado de manera eficiente y optimizada para garantizar un rendimiento óptimo. Este sistema de binarios permite el procesamiento rápido y la manipulación eficiente de datos, lo cual resulta fundamental para el correcto funcionamiento de la aplicación.

Además, se ha puesto un gran énfasis en el cuidado del estilo en CSS. Se ha seguido una metodología de desarrollo limpio y modular, utilizando selectores bien estructurados y prácticas de nomenclatura coherentes. Esto no solo facilita la legibilidad y mantenibilidad del código, sino que también mejora la escalabilidad del proyecto, permitiendo la adición de nuevas características y la realización de cambios sin dificultades.

Otro aspecto destacado es el uso interno del back-end. Se ha desarrollado una arquitectura sólida y escalable, que permite manejar eficientemente las solicitudes del cliente y procesar los datos necesarios para brindar una experiencia fluida. Se han utilizado tecnologías y frameworks modernos, lo que ha permitido la implementación de funcionalidades avanzadas y la mejora del rendimiento en general.

En resumen, el proyecto se destaca por su sistema de binarios eficiente, el cuidado del estilo en CSS y el uso interno del back-end. Estas características han sido diseñadas y desarrolladas con el objetivo de lograr un proyecto de alta calidad, que ofrezca un rendimiento óptimo, una experiencia de usuario agradable y una base sólida para futuras expansiones y mejoras.

**6.2.A****nálisis de trabajo**

Tecnologías utilizadas:

* Spring Boot: Se utiliza como servidor Backend. Spring Boot es un framework de desarrollo de aplicaciones Java que proporciona un entorno simplificado para crear aplicaciones robustas y escalables.
* Angular: Se utiliza como servidor Frontend. Angular es un framework de desarrollo de aplicaciones web desarrollado en TypeScript y permite crear interfaces de usuario interactivas y dinámicas.
* MySQL: Se utiliza como base de datos para almacenar la información del programa. MySQL es un sistema de gestión de bases de datos relacional que ofrece un alto rendimiento y una amplia gama de funcionalidades.
* SQLite: Se utiliza como base de datos para almacenar los archivos binarios. SQLite es una base de datos ligera y de alto rendimiento que se integra fácilmente con aplicaciones y ofrece una buena organización y rendimiento en la búsqueda de los binarios.
* Bootstrap: Se utiliza como framework de CSS para el diseño y la presentación de la interfaz de usuario. Bootstrap facilita la creación de interfaces responsivas y atractivas.

Ventajas de las tecnologías seleccionadas:

* Spring Boot: Ofrece una configuración y puesta en marcha sencillas, así como una integración fluida con otras bibliotecas y frameworks de Spring. Proporciona un soporte sólido para el desarrollo de servicios web y la implementación de API RESTful.
* Angular: Permite crear una interfaz de usuario moderna y receptiva. Su estructura modular y la capacidad de carga veloz de vistas web son beneficiosos para este proyecto de tablón de imágenes.
* MySQL: Es un sistema de gestión de bases de datos ampliamente utilizado y confiable. Ofrece un alto rendimiento y una buena escalabilidad, lo que es importante para un programa que maneja una gran cantidad de datos.
* SQLite: Es una base de datos ligera y de fácil configuración, lo que facilita su integración en el proyecto. Proporciona un buen rendimiento en términos de búsqueda y organización de archivos binarios.
* Bootstrap: Facilita el diseño y la presentación de la interfaz de usuario. Proporciona un conjunto de estilos y componentes predefinidos que agilizan el desarrollo y aseguran una apariencia coherente en diferentes dispositivos y navegadores.

Funcionalidades principales del proyecto:

* Registro de usuarios: Permite a los usuarios crear una cuenta en el sistema para acceder a las funcionalidades del tablón de imágenes.
* Visualización de publicaciones: Los usuarios pueden ver las publicaciones existentes en el tablón de imágenes.
* Subida de publicaciones: Los usuarios pueden cargar imágenes y agregar información relevante para crear nuevas publicaciones en el tablón.
* Comentarios en publicaciones: Los usuarios pueden comentar las publicaciones existentes, lo que fomenta la interacción y la participación de la comunidad.
* Etiquetas en publicaciones: Los usuarios pueden agregar etiquetas a las publicaciones para clasificarlas y facilitar su búsqueda y organización.

**6.3.****Mejoras Posibles**

|  |  |
| --- | --- |
| Posibilidades | Inconvenientes |
| Una gran inflexión en el proyecto seria desarrollar su par de frontend en Android y iOS de tal forma que también hubiera soporte para dichas plataformas. | Sería posible de más tiempo de desarrollo. |
| Sería conveniente montar la aplicación en Docker para mayor facilidad al montarla. | Debido a la gran complejidad del programa sería necesario hacer un estudio sobre las variables conjuntas, esto sería posible realizarlo más adelante. |
| Lanzador para Linux | Debido a la falta de tiempo no fue posible hacer el lanzador para servidores Linux |
| Panel de Usuarios y administración | En posteriores versiones se tiene planeado finalizar esta parte |
| Mejora del sistema y regeneración de tokens | Debido a la complejidad y la falta de tiempo no ha sido posible. |

Tabla 6.3.1 – Mejoras posibles

**6.4.****Mercado**

**Comercialización:**

En cuanto al mercado, la comercialización es un aspecto importante a considerar. Aunque el objetivo principal del proyecto no sea la venta ni la comercialización en sí misma, es relevante evaluar su rentabilidad en caso de ser llevado al mercado. Una estrategia que se podría implementar para lograr esto es mediante el uso de anuncios (ADS) en diferentes páginas.

La inclusión de anuncios publicitarios dentro del proyecto permitiría generar ingresos y, por ende, hacerlo rentable. Estos anuncios podrían ser cuidadosamente seleccionados y colocados estratégicamente en las distintas secciones del proyecto, de manera que no afecten negativamente la experiencia del usuario, pero sí brinden una fuente adicional de ingresos.

Es importante destacar que la elección de las páginas donde se mostrarían los anuncios debe ser realizada con un enfoque adecuado, seleccionando sitios web relevantes y afines al tema del proyecto. Esto permitiría llegar a un público objetivo más amplio y aumentar las posibilidades de generar interés y clics en los anuncios.

Además, es recomendable utilizar herramientas de análisis y seguimiento para evaluar el rendimiento de los anuncios y realizar ajustes en caso necesario. Esto permitirá maximizar los resultados y optimizar la rentabilidad del proyecto en caso de ser comercializado.

Es fundamental tener en cuenta que la implementación de anuncios publicitarios debe ser cuidadosa y equilibrada, evitando saturar al usuario con excesiva publicidad. El objetivo principal del proyecto debe seguir siendo brindar un valor significativo a los usuarios, y los anuncios deben ser considerados como una forma complementaria de monetización sin comprometer la calidad y la usabilidad del proyecto en sí.

En resumen, aunque la venta y comercialización no sean el objetivo principal del proyecto, la inclusión de anuncios publicitarios puede ser una estrategia viable para hacerlo rentable en caso de ser comercializado. Mediante la selección adecuada de páginas y el seguimiento del rendimiento de los anuncios, se puede optimizar la rentabilidad sin comprometer la calidad del proyecto ni la experiencia del usuario.

**Público Objetivo:**

Además de las comunidades artísticas o fan arts, el mercado objetivo de esta aplicación se extiende a otros grupos relacionados con el arte y la creatividad. Esto incluye a los diseñadores gráficos, ilustradores, fotógrafos y cualquier persona apasionada por la expresión visual.

Estos grupos suelen encontrar en los tableros de imágenes una forma efectiva de compartir sus creaciones, recibir retroalimentación constructiva y establecer conexiones con otros artistas y entusiastas del arte. La aplicación proporciona un espacio virtual donde pueden exhibir y promocionar su trabajo de manera accesible y atractiva.

Además, el público objetivo de esta aplicación puede abarcar a aquellos que buscan inspiración y entretenimiento visual. Los amantes del arte en general, los aficionados a la moda, los entusiastas del diseño de interiores y las personas interesadas en descubrir nuevas tendencias y estilos también pueden beneficiarse de esta plataforma.

En resumen, el mercado objetivo de esta aplicación abarca a las comunidades artísticas y fan arts, así como a diseñadores, ilustradores, fotógrafos y entusiastas del arte en general. También atrae a personas en busca de inspiración y entretenimiento visual, así como a aquellos interesados en descubrir nuevas tendencias y formas de expresión artística en diversos medios visuales. La aplicación proporciona un espacio interactivo donde todos estos grupos pueden conectarse, compartir y disfrutar del arte en todas sus formas.

**6.5.****Tiempo estimado**

|  |  |
| --- | --- |
| Sección | Tiempo estimado |
| Memoria | 65h |
| Presentación | 12h |
| Lanzador Grafico | 8h |
| Lanzador de consola | 30m |
| Sistema de auto configuración | 2,11h |
| FrontEnd | 32h |
| BackEnd | 43h |

(El tiempo se ha calculado de forma aproximada a base de cronometraje del tiempo)

Tabla 6.5.1 – Tiempo estimado

**6.6.****Valoración Personal**

En el desarrollo de este proyecto he ido observando diversas dificultades asi como formas de lograr el objetivo de una forma diferente, ya que el objetivo de este proyecto además de la creación de dicha web con diversas tecnologías más actuales, era el reto de aprender dichas tecnologías y usarlas en conjunto para una gran satisfacción.

En conjunto el proyecto en si es complejo ya que usamos lenguajes como Java, C#, .bat, Js, Ts, Sql de los cuales algunos se han tenido que aprender desde cero. A parte del aprendizaje de la conexión de diferentes frameworks como SpringBoot o Angular y la creación de ventanas de usuario con Forms.

Además de lo anterior, uno de los desafíos más complejos fue la transmisión de datos binarios entre el backend y el frontend de la aplicación. Para lograrlo, fue necesario implementar un proceso en el que los binarios se convirtieran en formato base64 al ser cargados, luego se enviaran al backend y se formatearan como un tipo de dato BLOB (Binary Large Object), el cual se almacenaba en una base de datos local dinámica por usuario.

Esta tarea requería una cuidadosa manipulación de los datos, ya que la conversión de binarios a base64 y viceversa implicaba transformaciones específicas. En el lado del backend, se debían realizar las operaciones inversas para recuperar los datos binarios originales y garantizar la integridad de la información.

Además de la complejidad técnica de la transmisión y almacenamiento de los binarios, también se enfrentaban desafíos adicionales en la obtención de los datos. El proceso de obtención de datos debía tener en cuenta la estructura de la base de datos local dinámica, que almacenaba información de forma personalizada para cada usuario. Esto requería consultas y filtrado adecuados para obtener los datos relevantes de manera eficiente.

En resumen, la transmisión de binarios entre el backend y el frontend, junto con la manipulación y almacenamiento adecuados de dichos binarios, representó una de las tareas más complicadas en el desarrollo de la aplicación. Requirió una cuidadosa planificación y ejecución para garantizar que los datos se transmitieran correctamente y estuvieran disponibles para su recuperación posterior.

**Capítulo 7**

**Fuentes de Información**

**7.1.****Bibliografía**

Anandmeg. (2023, 8 mayo). *Creación de una aplicación de Windows Forms con C# - Visual Studio (Windows)*. Microsoft Learn. https://learn.microsoft.com/es-es/visualstudio/ide/create-csharp-winform-visual-studio

BillWagner. (2023, 15 febrero). *Un paseo por C#: información general*. Microsoft Learn. https://learn.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/

CBujeda. (s. f.). *GitHub - CBujeda/Nin-ImageBoard: Gestor web de imágenes publico.* GitHub. https://github.com/CBujeda/Nin-ImageBoard

colaboradores de Wikipedia. (2022). Archivo batch. *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/wiki/Archivo\_batch

colaboradores de Wikipedia. (2023). Microsoft .NET. *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/wiki/Microsoft\_.NET

**7.2.****Recursos**

*Counter General Booru*. (s.f.). Obtenido de https://safebooru.org/counter/{0-9}.gif

*Bootstrap Icons*. (s. f.). https://icons.getbootstrap.com/

*MySQL :: MySQL Community Server*. (s. f.). https://dev.mysql.com/downloads/mysql/

*Node.js*. (s. f.). Node.js. <https://nodejs.org/es>

*DB Browser for SQLite*. (s. f.). https://sqlitebrowser.org/

**7.3.****Documentación**

Contributors, M. O. J. T. A. B. (s. f.). *Get started with Bootstrap*. https://getbootstrap.com/docs/5.3/getting-started/introduction/

*CSS | MDN*. (2023, 13 marzo). https://developer.mozilla.org/es/docs/Web/CSS

*Spring Boot*. (s. f.). Spring Boot. https://spring.io/projects/spring-boot

*SQLite Documentation*. (s. f.). https://www.sqlite.org/docs.html

Wikipedia contributors. (2023). .properties. *Wikipedia*. https://en.wikipedia.org/wiki/.properties

**Anexos**

**1 -** **Índice de terminología**

|  |  |
| --- | --- |
| Palabra | Definición |
| Post | Campo con contenido multimedia de tipo imagen (binario) la cual está compuesta por su meta data, nombre, fecha de subida además de sus diferentes tags identificatorios |
| Binario | Archivo compuesto por un conjunto de Bytes pueden ser cualquier tipo de archivo de no texto |
| Tag | Etiqueta que identifica un post. Ej.:( Una imagen que aparece una chica se le añadirá la Tag “girl” ) |
| Backend | Servidor que actúa no de forma frontal al usuario |
| Frontend | Servidor que actúa de forma frontal al usuario |
| S-A | SpringBoot ↔ Angular |
| Api | Servicio web que provee datos |
| Ads | Anuncios |
| Java | Java es un lenguaje de programación orientado a objetos que se utiliza para desarrollar aplicaciones y programas de software. |
| C# | C# (pronunciado "C sharp") es un lenguaje de programación orientado a objetos creado por Microsoft. |
| Js | JS es una abreviatura comúnmente utilizada para referirse a JavaScript, un lenguaje de programación interpretado utilizado principalmente para agregar interactividad y dinamismo a las páginas web. |
| Ts | TS es una abreviatura comúnmente utilizada para referirse a TypeScript, un lenguaje de programación de código abierto desarrollado por Microsoft. TypeScript es un superset de JavaScript que agrega características adicionales. |
| .bat | .bat es una extensión de archivo utilizada para archivos de procesamiento por lotes en sistemas operativos de Windows. Un archivo .bat contiene un conjunto de comandos que se ejecutan secuencialmente cuando se inicia el archivo. |
| Forms | Forms es un sistema de creación de aplicaciones graficas desarrollado por Microsoft, se puede programar tanto en C# o en vb |
| SpringBoot | Spring Boot es un marco de desarrollo de aplicaciones de código abierto para Java que simplifica el proceso de configuración y creación de aplicaciones basadas en Spring Framework. |
| Thymeleaf | Thymeleaf es un motor de plantillas de código abierto para Java que se utiliza para crear y renderizar vistas en aplicaciones web. |
| Angular | Angular es un framework de desarrollo de aplicaciones web de código abierto y basado en JavaScript. Fue desarrollado por Google y se utiliza para crear aplicaciones web de una sola página y aplicaciones móviles híbridas. |
| NodeJs | Node.js es un entorno de tiempo de ejecución de código abierto basado en JavaScript que permite a los desarrolladores ejecutar código de JavaScript fuera del navegador web. |
| MIME | MIME (Multipurpose Internet Mail Extensions) es un estándar de Internet que define el formato de los mensajes de correo electrónico y otros contenidos transmitidos a través de la red. MIME se utiliza para describir el tipo de contenido de un archivo o mensaje, y para especificar cómo se deben codificar y decodificar los datos para su transmisión. Resumidamente serian archivos multimedia |
| BLOB | Los BLOB (Binary Large Objects) son elementos fundamentales en las bases de datos que permiten el almacenamiento de datos de gran tamaño que cambian de forma dinámica. |
| MVC | MVC es un patrón de arquitectura de software que se utiliza para separar la lógica de presentación de una aplicación web de la lógica de negocio y los datos subyacentes. La abreviatura MVC significa Modelo-Vista-Controlador. |
| JPA | JPA son las siglas de Java Persistence API, una especificación de Java que define un conjunto de interfaces y clases que se utilizan para persistir objetos Java en una base de datos relacional. |
| CRUD | CRUD es un acrónimo que se refiere a las operaciones básicas que se pueden realizar en una base de datos o sistema de almacenamiento de datos. CRUD significa Create (crear), Read (leer), Update (actualizar) y Delete (eliminar). |
| JUnit | JUnit es un framework de pruebas unitarias de código abierto para el lenguaje de programación Java. |

Tabla A.1 – Índice de terminología

**2 -** **Ventanas**

**2.2 - Paginas ocultas**

* Créditos, dicha ventana muestra la información sobre la creación del programa web a la cual solo se puede acceder desde en la página 404.

**3 – Resumen de código**

**Backend**

NibShell – Encargada de imprimir por pantalla todo lo que tenga que ver con la información de la consola

1. public class NIBShell {

2.

3. public NIBShell() {

4. super();

5. // TODO Auto-generated constructor stub

6. }

7.

8. public void print() {

9.

10. }

11. /\*

12. \* Pre:

13. \* Post: Metodo el cual imprime por consola infromacion.

14. \*/

15. public void printInfo(String str) {

16. String YELLOW\_BOLD = "\033[1;33m"; // YELLOW

17. String PURPLE\_BOLD = "\033[1;35m"; // PURPLE

18. String RESET = "\033[0m"; // RESET

19. String date = java.time.Clock.systemUTC().instant().toString();

20. System.out.println(date.substring(0, date.length()-1)+" "+YELLOW\_BOLD+"[NIB] "+PURPLE\_BOLD+" --- "+RESET + str);

21. }

22. /\*

23. \* Pre:

24. \* Post: Metodo el cual imprime por consola datos SQL

25. \*/

26. public void printInfoSql(String str) {

27. String YELLOW\_BOLD = "\033[1;33m"; // YELLOW

28. String CYAN = "\u001B[36m";

29. String PURPLE\_BOLD = "\033[1;35m"; // PURPLE

30. String RESET = "\033[0m"; // RESET

31. String date = java.time.Clock.systemUTC().instant().toString();

32. System.out.println(date.substring(0, date.length()-1)+" "+YELLOW\_BOLD+"[NIB] "+CYAN+"[SQL] "+PURPLE\_BOLD+" --- "+RESET + str);

33. }

34. }

35.

Password Encryptor – Clase la cual se encarga de gestionar contraseñas y demas

1. package com.nib.app.utils;

2. import java.security.MessageDigest;

3. import java.security.NoSuchAlgorithmException;

4. import java.security.SecureRandom;

5. import java.util.Base64;

6. public class PasswordEncryptor {

7. // Varoables de configuracion del encriptador

8. private final int SALT\_LENGTH = 16;

9. private final int ITERATIONS = 100000;

10. private final int KEY\_LENGTH = 256;

11.

12. public PasswordEncryptor() {

13. super();

14. }

15. /\*

16. \* Metodo el cual encripta una contraseña

17. \*/

18. public String encrypt(String password) {

19. try {

20. SecureRandom random = new SecureRandom();

21. byte[] salt = new byte[SALT\_LENGTH];

22. random.nextBytes(salt);

23.

24. MessageDigest md = MessageDigest.getInstance("SHA-256");

25. md.update(salt);

26. byte[] hashedPassword = md.digest(password.getBytes());

27.

28. for (int i = 0; i < ITERATIONS - 1; i++) {

29. md.reset();

30. hashedPassword = md.digest(hashedPassword);

31. }

32.

33. byte[] saltedPassword = new byte[SALT\_LENGTH + hashedPassword.length];

34. System.arraycopy(salt, 0, saltedPassword, 0, SALT\_LENGTH);

35. System.arraycopy(hashedPassword, 0, saltedPassword, SALT\_LENGTH, hashedPassword.length);

36. return Base64.getEncoder().encodeToString(saltedPassword);

37. } catch (NoSuchAlgorithmException e) {

38. throw new RuntimeException("Error al encriptar la contraseña: " + e.getMessage());

39. }

40. }

41.

42. /\*

43. \* Metodo el cual verifica si las contraseñas coinciden

44. \*/

45. public boolean checkPassword(String password, String encryptedPassword) {

46. byte[] saltedPassword = Base64.getDecoder().decode(encryptedPassword);

47. byte[] salt = new byte[SALT\_LENGTH];

48. System.arraycopy(saltedPassword, 0, salt, 0, SALT\_LENGTH);

49.

50. byte[] hashedPassword = new byte[saltedPassword.length - SALT\_LENGTH];

51. System.arraycopy(saltedPassword, SALT\_LENGTH, hashedPassword, 0, hashedPassword.length);

52.

53. try {

54. MessageDigest md = MessageDigest.getInstance("SHA-256");

55. md.update(salt);

56. byte[] testHashedPassword = md.digest(password.getBytes());

57.

58. for (int i = 0; i < ITERATIONS - 1; i++) {

59. md.reset();

60. testHashedPassword = md.digest(testHashedPassword);

61. }

62. return MessageDigest.isEqual(hashedPassword, testHashedPassword);

63.

64. } catch (NoSuchAlgorithmException e) {

65. throw new RuntimeException("Error al comprobar la contraseña: " + e.getMessage());

66. }

67. }

68. }

69.

TokenGen – Clase la cual genera un token de auth para el usuario

1. package com.nib.app.utils;

2.

3. import java.security.SecureRandom;

4.

5. public class TokenGen {

6.

7. /\*

8. \* Pre:

9. \* Post: Caracteres token

10. \*/

11. private final String UPPERCASE = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

12. private final String LOWERCASE = "abcdefghijklmnopqrstuvwxyz";

13. private final String NUMBERS = "0123456789";

14. private final String SYMBOLS = "!@$&\*\_=+-";

15.

16. private final String ALL\_CHARACTERS = UPPERCASE + LOWERCASE + NUMBERS + SYMBOLS;

17. private final SecureRandom random = new SecureRandom();

18.

19. public TokenGen() {

20. super();

21. // TODO Auto-generated constructor stub

22. }

23.

24. /\*

25. \* Pre:

26. \* Post: Metodo el cual genera un nuevo token

27. \*/

28. public String generate(int length) {

29. StringBuilder sb = new StringBuilder(length);

30.

31. // Add at least one character from each group

32. sb.append(getRandomChar(UPPERCASE));

33. sb.append(getRandomChar(LOWERCASE));

34. sb.append(getRandomChar(NUMBERS));

35. sb.append(getRandomChar(SYMBOLS));

36.

37. // Add remaining characters randomly

38. for (int i = 4; i < length; i++) {

39. sb.append(getRandomChar(ALL\_CHARACTERS));

40. }

41.

42. // Shuffle the characters to increase randomness

43. char[] chars = sb.toString().toCharArray();

44. for (int i = chars.length - 1; i > 0; i--) {

45. int j = random.nextInt(i + 1);

46. char temp = chars[i];

47. chars[i] = chars[j];

48. chars[j] = temp;

49. }

50.

51. return new String(chars);

52. }

53.

54. /\*

55. \* Pre:

56. \* Post: Metodo el cual obtiene un caracter aleatorio

57. \*/

58. private char getRandomChar(String characters) {

59. int index = random.nextInt(characters.length());

60. return characters.charAt(index);

61. }

62.

63. }

64.

SLUtils – Clase de generación de LiteSQL

1. package com.nib.app.utils.sqLite;

2.

3. import java.io.File;

4. import java.sql.Connection;

5. import java.sql.DriverManager;

6. import java.sql.PreparedStatement;

7. import java.sql.ResultSet;

8. import java.sql.SQLException;

9.

10. import com.nib.app.utils.NIBShell;

11.

12. public class SLUtils {

13.

14. private Connection connection;

15. private String id\_bbdd;

16.

17. private final String mainTable = "Master";

18.

19. /\*

20. \* Pre:

21. \* Post: Metodo el cual crea una conexion y crea una

22. \* bbdd local en caso de no existir

23. \*/

24. public Connection getConnection(String id\_bbdd) {

25. Connection connection = null;

26. this.id\_bbdd = id\_bbdd;

27. try {

28. Class.forName("org.sqlite.JDBC");

29. File dbFile = new File("staticBBDD/"+id\_bbdd+".db");

30. if (!dbFile.getParentFile().exists()) {

31. dbFile.getParentFile().mkdirs();

32. }

33. connection = DriverManager.getConnection("jdbc:sqlite:" + dbFile.getAbsolutePath());

34. //createTables(connection);

35. } catch (ClassNotFoundException | SQLException e) {

36. e.printStackTrace();

37. }

38. this.connection = connection;

39. return connection;

40. }

41.

42. /\*

43. \* Pre:

44. \* Post: Metodo el cual se encarga de generar las tablas de la base de datos local

45. \*/

46. public void createTables() throws SQLException {

47. this.printInfoSql("Creando Tablas de NIB BBDD 'Id = " + this.id\_bbdd +"'" );

48. PreparedStatement statement = this.connection.prepareStatement(

49. " CREATE TABLE IF NOT EXISTS "+mainTable+" ( "

50. + " iddb\_binary INTEGER PRIMARY KEY AUTOINCREMENT, "

51. + " Binary BLOB, "

52. + " id\_img INTEGER );"

53. );

54. statement.executeUpdate();

55. }

56.

57. /\*

58. \* Pre:

59. \* Post: Metodo el cual se encarga de al almacenar los binarios en la tabla

60. \*/

61. public void savePost(byte[] imageBytes,Long ext) throws SQLException {

62. PreparedStatement pstmt = this.connection.prepareStatement(

63. "INSERT INTO "+mainTable+" (Binary,id\_img) VALUES (?,?)"

64. );

65. pstmt.setBytes(1, imageBytes);

66. pstmt.setFloat(2, ext);

67. pstmt.executeUpdate();

68. pstmt.close();

69. this.connection.close();

70. }

71.

72. /\*

73. \* Pre:

74. \* Post: Metodo el cual obtiene un post (binarios) de la bbdd mediante el id alamcenado

75. \*/

76. public byte[] getPostByIdPost(Long id) {

77. try {

78. PreparedStatement pstmt =

79. this.connection.prepareStatement(

80. "SELECT Binary FROM "+mainTable+" WHERE id\_img = ?"

81. );

82. pstmt.setLong(1, id);

83. ResultSet rs = pstmt.executeQuery();

84. byte[] imageBytes = rs.getBytes("Binary");

85. rs.close();

86. pstmt.close();

87. this.closeConexion();

88. return imageBytes;

89. } catch (SQLException e) {

90. e.printStackTrace();

91. return null;

92. }

93. }

94.

95. /\*

96. \* Pre:

97. \* Post: Metodo con el cual cerramos una conexion

98. \*/

99. public void closeConexion() {

100. try {

101. this.connection.close();

102. } catch (SQLException e) {

103. // TODO Auto-generated catch block

104. e.printStackTrace();

105. }

106. }

107.

108. /\*

109. \* Pre:

110. \* Post: Metodo con el cual imprimimos por consola informacion

111. \* del sistema sqlLite

112. \*/

113. public void printInfoSql(String str) {

114. new NIBShell().printInfoSql(str);

115. }

116. }

117.

SCong – Objeto de transmisión de configuración

1. package com.nib.app.objects;

2.

3. /\*

4. \* Objeto de configuracion

5. \*/

6. public class SConf {

7.

8. private String token;

9. private String appname;

10. private String appshortname;

11.

12. public SConf() {

13. super();

14. // TODO Auto-generated constructor stub

15. }

16. public String getToken() {

17. return token;

18. }

19. public void setToken(String token) {

20. this.token = token;

21. }

22. public String getAppname() {

23. return appname;

24. }

25. public void setAppname(String appname) {

26. this.appname = appname;

27. }

28. public String getAppshortname() {

29. return appshortname;

30. }

31. public void setAppshortname(String appshortname) {

32. this.appshortname = appshortname;

33. }

34.

35.

36. }

PO – Generic Private Objects

1. package com.nib.app.objects;

2.

3. /\*

4. \* Generic Private Objects

5. \*/

6. public class PO {

7.

8. private String token;

9. private String data;

10. public PO() {

11. super();

12. // TODO Auto-generated constructor stub

13. }

14. public PO(String token, String data) {

15. super();

16. this.token = token;

17. this.data = data;

18. }

19. public String getToken() {

20. return token;

21. }

22. public void setToken(String token) {

23. this.token = token;

24. }

25. public String getData() {

26. return data;

27. }

28. public void setData(String data) {

29. this.data = data;

30. }

31. @Override

32. public String toString() {

33. return "PO [token=" + token + ", data=" + data + "]";

34. }

35.

36.

37. }

38.

Image – Objeto que almacena la Imagen

1. package com.nib.app.objects;

2.

3. /\*

4. \* Objeto el cual almacena una imagen en base64

5. \*/

6. public class Image {

7.

8. private String title;

9. private String tags;

10. private String base64Img;

11. private String token;

12.

13. public Image() {

14. super();

15. // TODO Auto-generated constructor stub

16. }

17.

18. public Image(String title, String tags, String base64Img, String token) {

19. super();

20. this.title = title;

21. this.tags = tags;

22. this.base64Img = base64Img;

23. this.token = token;

24. }

25.

26. public String getTitle() {

27. return title;

28. }

29.

30. public void setTitle(String title) {

31. this.title = title;

32. }

33.

34. public String getTags() {

35. return tags;

36. }

37.

38. public void setTags(String tags) {

39. this.tags = tags;

40. }

41.

42. public String getBase64Img() {

43. return base64Img;

44. }

45.

46. public void setBase64Img(String base64Img) {

47. this.base64Img = base64Img;

48. }

49.

50. public String getToken() {

51. return token;

52. }

53.

54. public void setToken(String token) {

55. this.token = token;

56. }

57.

58. @Override

59. public String toString() {

60. return "Image [title=" + title + ", tags=" + tags + ", base64Img=" + base64Img + ", token=" + token + "]";

61. }

62.

63.

64.

65. }

66.

FComment – Objeto de transmisión de comentario

1. package com.nib.app.objects;

2.

3. import java.util.Date;

4.

5. import com.nib.app.model.entity.Post;

6. import com.nib.app.model.entity.user.Usuario;

7.

8. import jakarta.persistence.Column;

9. import jakarta.persistence.JoinColumn;

10. import jakarta.persistence.ManyToOne;

11.

12. /\*

13. \* Objeto el cual almacena un comentario de front

14. \*/

15. public class FComment {

16.

17. private Long id\_comment;

18. private String token\_usuario;

19. private Long id\_post;

20. private String nameUsuario;

21. private String imgUsuario;

22.

23. private String message;

24.

25. private Date date; // Creacion del comentario

26.

27. public FComment() {

28. super();

29. // TODO Auto-generated constructor stub

30. }

31.

32. public FComment(Long id\_comment, String token\_usuario, Long id\_post, String nameUsuario, String imgUsuario,

33. String message, Date date) {

34. super();

35. this.id\_comment = id\_comment;

36. this.token\_usuario = token\_usuario;

37. this.id\_post = id\_post;

38. this.nameUsuario = nameUsuario;

39. this.imgUsuario = imgUsuario;

40. this.message = message;

41. this.date = date;

42. }

43.

44. public Long getId\_comment() {

45. return id\_comment;

46. }

47.

48. public void setId\_comment(Long id\_comment) {

49. this.id\_comment = id\_comment;

50. }

51.

52. public String getToken\_usuario() {

53. return token\_usuario;

54. }

55.

56. public void setToken\_usuario(String token\_usuario) {

57. this.token\_usuario = token\_usuario;

58. }

59.

60. public Long getId\_post() {

61. return id\_post;

62. }

63.

64. public void setId\_post(Long id\_post) {

65. this.id\_post = id\_post;

66. }

67.

68. public String getNameUsuario() {

69. return nameUsuario;

70. }

71.

72. public void setNameUsuario(String nameUsuario) {

73. this.nameUsuario = nameUsuario;

74. }

75.

76. public String getImgUsuario() {

77. return imgUsuario;

78. }

79.

80. public void setImgUsuario(String imgUsuario) {

81. this.imgUsuario = imgUsuario;

82. }

83.

84. public String getMessage() {

85. return message;

86. }

87.

88. public void setMessage(String message) {

89. this.message = message;

90. }

91.

92. public Date getDate() {

93. return date;

94. }

95.

96. public void setDate(Date date) {

97. this.date = date;

98. }

99.

100. @Override

101. public String toString() {

102. return "FComment [id\_comment=" + id\_comment + ", token\_usuario=" + token\_usuario + ", id\_post=" + id\_post

103. + ", nameUsuario=" + nameUsuario + ", imgUsuario=" + imgUsuario + ", message=" + message + ", date="

104. + date + "]";

105. }

106.

107.

108.

109.

110. }

BinaryFile – Objeto de transmisión de datos binarios como objetos tipo MIME o BLOB

1. package com.nib.app.objects;

2.

3. /\*

4. \* Objeto el cual almacena un dato binario

5. \* como imagenes, videos, etc (Formatos MIME)

6. \*/

7. public class BinaryFile {

8.

9. private byte[] data;

10. private String format;

11. public BinaryFile() {

12. super();

13. // TODO Auto-generated constructor stub

14. }

15. public BinaryFile(byte[] data, String format) {

16. super();

17. this.data = data;

18. this.format = format;

19. }

20. public byte[] getData() {

21. return data;

22. }

23. public void setData(byte[] data) {

24. this.data = data;

25. }

26. public String getFormat() {

27. return format;

28. }

29. public void setFormat(String format) {

30. this.format = format;

31. }

32.

33. }

34.

Objeto de usuario

1. package com.nib.app.model.entity.user;

2.

3. import java.io.Serializable;

4. import java.util.HashSet;

5. import java.util.Set;

6.

7. import com.fasterxml.jackson.annotation.JsonIgnore;

8. import com.fasterxml.jackson.annotation.JsonManagedReference;

9. import com.nib.app.model.entity.Post;

10.

11. import jakarta.persistence.CascadeType;

12. import jakarta.persistence.Column;

13. import jakarta.persistence.Entity;

14. import jakarta.persistence.FetchType;

15. import jakarta.persistence.GeneratedValue;

16. import jakarta.persistence.GenerationType;

17. import jakarta.persistence.Id;

18. import jakarta.persistence.JoinColumn;

19. import jakarta.persistence.JoinTable;

20. import jakarta.persistence.ManyToMany;

21. import jakarta.persistence.OneToMany;

22. import jakarta.persistence.Table;

23.

24. /\*

25. \* Entidad de usuario

26. \*/

27. @Entity

28. @Table(name = "usuario")

29. public class Usuario implements Serializable {

30.

31. private static final long serialVersionUID = -3256103979765596155L;

32. @Id

33. @GeneratedValue(strategy = GenerationType.IDENTITY)

34. @Column(name="id\_usuario")

35. private Long id\_usuario;

36.

37. private String username;

38. private String nombre;

39. private String apellido;

40. private String email;

41. private String password;

42. private boolean verify = true;

43. private String imgProfile;

44. private String token;

45.

46.

47. //@JsonManagedReference

48. @JsonIgnore

49. @OneToMany(mappedBy = "usuario",orphanRemoval = true, cascade = CascadeType.ALL)

50. private Set<Post> postList;

51.

52. //@JsonManagedReference

53.

54. @ManyToMany(cascade = CascadeType.ALL,fetch = FetchType.EAGER) //fetch = FetchType.EAGER, ,mappedBy = "userlist"

55. @JoinTable(

56. name="usuario\_nn\_rol",

57. joinColumns = @JoinColumn(name="id\_usuario",unique = false),

58. inverseJoinColumns = @JoinColumn(name="id\_rol",unique = false)

59. )

60. @JsonIgnore

61. private Set<Rol> roleslist;

62.

63.

64. public Usuario() {

65. super();

66. roleslist = new HashSet<>();

67.

68. // TODO Auto-generated constructor stub

69. }

70.

71.

72. public Usuario(Long id\_usuario, String username, String nombre, String apellido, String email, String password,

73. boolean verify, String imgProfile, Set<Post> postList, Set<Rol> roleslist) {

74. super();

75. this.id\_usuario = id\_usuario;

76. this.username = username;

77. this.nombre = nombre;

78. this.apellido = apellido;

79. this.email = email;

80. this.password = password;

81. this.verify = verify;

82. this.imgProfile = imgProfile;

83. this.postList = postList;

84. this.roleslist = roleslist;

85. }

86.

87.

88. public Long getId\_usuario() {

89. return id\_usuario;

90. }

91.

92.

93. public void setId\_usuario(Long id\_usuario) {

94. this.id\_usuario = id\_usuario;

95. }

96.

97.

98. public String getUsername() {

99. return username;

100. }

101.

102.

103. public void setUsername(String username) {

104. this.username = username;

105. }

106.

107.

108. public String getNombre() {

109. return nombre;

110. }

111.

112.

113. public void setNombre(String nombre) {

114. this.nombre = nombre;

115. }

116.

117.

118. public String getApellido() {

119. return apellido;

120. }

121.

122.

123. public void setApellido(String apellido) {

124. this.apellido = apellido;

125. }

126.

127.

128. public String getEmail() {

129. return email;

130. }

131.

132.

133. public void setEmail(String email) {

134. this.email = email;

135. }

136.

137.

138. public String getPassword() {

139. return password;

140. }

141.

142.

143. public void setPassword(String password) {

144. this.password = password;

145. }

146.

147.

148. public boolean isVerify() {

149. return verify;

150. }

151.

152.

153. public void setVerify(boolean verify) {

154. this.verify = verify;

155. }

156.

157.

158. public String getImgProfile() {

159. return imgProfile;

160. }

161.

162.

163. public void setImgProfile(String imgProfile) {

164. this.imgProfile = imgProfile;

165. }

166.

167.

168. public Set<Post> getPostList() {

169. return postList;

170. }

171.

172.

173. public void setPostList(Set<Post> postList) {

174. this.postList = postList;

175. }

176.

177.

178. public Set<Rol> getRoleslist() {

179. return roleslist;

180. }

181.

182.

183. public void setRoleslist(Set<Rol> roleslist) {

184. this.roleslist = roleslist;

185. }

186.

187.

188. public static long getSerialversionuid() {

189. return serialVersionUID;

190. }

191.

192.

193. @Override

194. public String toString() {

195. return "Usuario [id\_usuario:" + id\_usuario + ", username:" + username + ", nombre:" + nombre + ", apellido:"

196. + apellido + ", email:" + email + ", password:" + password + ", verify:" + verify + ", imgProfile:"

197. + imgProfile + ", postList:[NO VIEW], roleslist: [NO VIEW] ]";

198. }

199.

200. public String getToken() {

201. return token;

202. }

203.

204. public void setToken(String token) {

205. this.token = token;

206. }

207.

208. }

209.

Objeto de Rol  
  
 1. package com.nib.app.model.entity.user;

2.

3. import java.io.Serializable;

4. import java.util.HashSet;

5. import java.util.List;

6. import java.util.Set;

7.

8. import com.fasterxml.jackson.annotation.JsonBackReference;

9.

10. import jakarta.persistence.CascadeType;

11. import jakarta.persistence.Column;

12. import jakarta.persistence.Entity;

13. import jakarta.persistence.FetchType;

14. import jakarta.persistence.GeneratedValue;

15. import jakarta.persistence.GenerationType;

16. import jakarta.persistence.Id;

17. import jakarta.persistence.ManyToMany;

18. import jakarta.persistence.OneToMany;

19. import jakarta.persistence.Table;

20.

21. /\*

22. \* Entidad de roles

23. \*/

24. @Entity

25. @Table(name = "roles")

26. public class Rol implements Serializable {

27.

28. private static final long serialVersionUID = 3552707046208988658L;

29.

30. @Id

31. @GeneratedValue(strategy = GenerationType.IDENTITY)

32. @Column(name="id\_rol")

33. private Long id\_rol;

34. private String nombre;

35.

36. //@JsonBackReference

37. @ManyToMany(cascade = CascadeType.ALL,fetch = FetchType.LAZY,mappedBy = "roleslist")//,fetch = FetchType.LAZY,mappedBy = "rol"

38. private Set<Usuario> userlist;

39.

40. public Rol() {

41. super();

42. userlist = new HashSet<>();

43. // TODO Auto-generated constructor stub

44. }

45.

46. public Rol(Long id\_rol, String nombre, Set<Usuario> userlist) {

47. super();

48. this.id\_rol = id\_rol;

49. this.nombre = nombre;

50. this.userlist = userlist;

51. }

52.

53. public Long getId\_rol() {

54. return id\_rol;

55. }

56.

57. public void setId\_rol(Long id\_rol) {

58. this.id\_rol = id\_rol;

59. }

60.

61. public String getNombre() {

62. return nombre;

63. }

64.

65. public void setNombre(String nombre) {

66. this.nombre = nombre;

67. }

68.

69. public Set<Usuario> getUserlist() {

70. return userlist;

71. }

72.

73. public void setUserlist(Set<Usuario> userlist) {

74. this.userlist = userlist;

75. }

76.

77. public static long getSerialversionuid() {

78. return serialVersionUID;

79. }

80.

81. @Override

82. public String toString() {

83. return "Rol [id\_rol=" + id\_rol + ", nombre=" + nombre + ", userlist=" + userlist + "]";

84. }

85.

86.

87. }

88.

Objeto de Comentarios

1. package com.nib.app.model.entity;

2.

3. import java.util.Date;

4.

5. import com.nib.app.model.entity.user.Usuario;

6.

7. import jakarta.persistence.Column;

8. import jakarta.persistence.Entity;

9. import jakarta.persistence.GeneratedValue;

10. import jakarta.persistence.GenerationType;

11. import jakarta.persistence.Id;

12. import jakarta.persistence.JoinColumn;

13. import jakarta.persistence.ManyToOne;

14. import jakarta.persistence.Table;

15.

16. /\*

17. \* Entidad de comentarios

18. \*/

19. @Entity

20. @Table(name = "comments")

21. public class Comment {

22.

23. @Id

24. @GeneratedValue(strategy = GenerationType.IDENTITY)

25. @Column(name="id\_comment")

26. private Long id\_comment;

27.

28. @ManyToOne

29. @JoinColumn(name = "id\_usuario") // creamos la columna de tipo objeto con joinColumn

30. private Usuario usuario;

31.

32. @ManyToOne

33. @JoinColumn(name = "id\_post") // creamos la columna de tipo objeto con joinColumn

34. private Post post;

35.

36. private String message;

37.

38. private Date date;

39.

40. public Comment() {

41. super();

42. // TODO Auto-generated constructor stub

43. }

44.

45. public Comment(Long id\_comment, Usuario usuario, Post post, String message, Date date) {

46. super();

47. this.id\_comment = id\_comment;

48. this.usuario = usuario;

49. this.post = post;

50. this.message = message;

51. this.date = date;

52. }

53.

54. public Long getId\_comment() {

55. return id\_comment;

56. }

57.

58. public void setId\_comment(Long id\_comment) {

59. this.id\_comment = id\_comment;

60. }

61.

62. public Usuario getUsuario() {

63. return usuario;

64. }

65.

66. public void setUsuario(Usuario usuario) {

67. this.usuario = usuario;

68. }

69.

70. public Post getPost() {

71. return post;

72. }

73.

74. public void setPost(Post post) {

75. this.post = post;

76. }

77.

78. public String getMessage() {

79. return message;

80. }

81.

82. public void setMessage(String message) {

83. this.message = message;

84. }

85.

86. public Date getDate() {

87. return date;

88. }

89.

90. public void setDate(Date date) {

91. this.date = date;

92. }

93. }

94.

Objeto de Configuración

1. package com.nib.app.model.entity;

2.

3. import java.io.Serializable;

4.

5. import jakarta.persistence.Column;

6. import jakarta.persistence.Entity;

7. import jakarta.persistence.GeneratedValue;

8. import jakarta.persistence.GenerationType;

9. import jakarta.persistence.Id;

10. import jakarta.persistence.Table;

11.

12. /\*

13. \* Entidad de configuracion

14. \*/

15. @Entity

16. @Table(name = "config")

17. public class Config implements Serializable{

18.

19. private static final long serialVersionUID = -247852851119605382L;

20.

21. @Id

22. @GeneratedValue(strategy = GenerationType.IDENTITY)

23. @Column(name="id\_conf")

24. private Long id\_conf;

25.

26. @Column(name="conf",unique = true)

27. private String conf;

28. private String value;

29. public Config() {

30. super();

31. // TODO Auto-generated constructor stub

32. }

33.

34.

35. public Config(String conf, String value) {

36. super();

37. this.conf = conf;

38. this.value = value;

39. }

40. public Config(Long id\_conf, String conf, String value) {

41. super();

42. this.id\_conf = id\_conf;

43. this.conf = conf;

44. this.value = value;

45. }

46. public Long getId\_conf() {

47. return id\_conf;

48. }

49. public void setId\_conf(Long id\_conf) {

50. this.id\_conf = id\_conf;

51. }

52. public String getConf() {

53. return conf;

54. }

55. public void setConf(String conf) {

56. this.conf = conf;

57. }

58. public String getValue() {

59. return value;

60. }

61. public void setValue(String value) {

62. this.value = value;

63. }

64.

65.

66. @Override

67. public String toString() {

68. return "Config [id\_conf=" + id\_conf + ", conf=" + conf + ", value=" + value + "]";

69. }

70.

71.

72. }

73.

Objeto de listas de insultos, No implementado

1. package com.nib.app.model.entity;

2.

3. import java.io.Serializable;

4.

5. import jakarta.persistence.Column;

6. import jakarta.persistence.Entity;

7. import jakarta.persistence.GeneratedValue;

8. import jakarta.persistence.GenerationType;

9. import jakarta.persistence.Id;

10. import jakarta.persistence.SequenceGenerator;

11. import jakarta.persistence.Table;

12.

13. /\*

14. \* Entidad de insultList

15. \* // Implementacion en futuras versiones

16. \*/

17. @Entity

18. @Table(name = "InsultList")

19. public class InsultList implements Serializable{

20.

21. private static final long serialVersionUID = -6598750758332736257L;

22.

23. @Id

24. @GeneratedValue(strategy = GenerationType.SEQUENCE)

25. @SequenceGenerator(name="postsecuence",sequenceName="DB\_SEQUENCIA", allocationSize=1)

26. @Column(name="id\_insult")

27. private Long id\_insult;

28.

29. @Column(name="insult")

30. private String insult;

31. }

32.

Objeto de Posts

1. package com.nib.app.model.entity;

2.

3. import java.io.Serializable;

4. import java.util.Date;

5. import java.util.Set;

6.

7. import com.fasterxml.jackson.annotation.JsonBackReference;

8. import com.fasterxml.jackson.annotation.JsonIgnore;

9. import com.nib.app.model.entity.user.Usuario;

10.

11. import jakarta.persistence.Column;

12. import jakarta.persistence.Entity;

13. import jakarta.persistence.GeneratedValue;

14. import jakarta.persistence.GenerationType;

15. import jakarta.persistence.Id;

16. import jakarta.persistence.JoinColumn;

17. import jakarta.persistence.JoinTable;

18. import jakarta.persistence.ManyToMany;

19. import jakarta.persistence.ManyToOne;

20. import jakarta.persistence.SequenceGenerator;

21. import jakarta.persistence.Table;

22.

23. /\*

24. \* Entidad de post

25. \*/

26. @Entity

27. @Table(name = "post")

28. public class Post implements Serializable{

29.

30. private static final long serialVersionUID = -2462737385258353400L;

31.

32. @Id

33. @GeneratedValue(strategy = GenerationType.SEQUENCE)

34. @SequenceGenerator(name="postsecuence",sequenceName="DB\_SEQUENCIA", allocationSize=1)

35. @Column(name="id\_post")

36. private Long id\_post;

37. private String img\_name;

38. private String extension;

39. private Date date;

40.

41. //@JsonBackReference

42. @JsonIgnore

43. @ManyToOne

44. @JoinColumn(name = "id\_usuario") // creamos la columna de tipo objeto con joinColumn

45. private Usuario usuario;

46.

47. @JsonIgnore

48. @ManyToMany

49. @JoinTable(

50. name = "post\_NN\_type",

51. joinColumns = @JoinColumn(name = "id\_post"),

52. inverseJoinColumns = @JoinColumn(name = "id\_type"))

53.

54. //@JoinColumn(name = "id\_type") // creamos la columna de tipo objeto con joinColumn

55. private Set<Type> typelist;

56.

57. @JsonIgnore

58. @ManyToMany

59. @JoinTable(

60. name = "post\_NN\_tag",

61. joinColumns = @JoinColumn(name = "id\_post"),

62. inverseJoinColumns = @JoinColumn(name = "id\_tag"))

63.

64. //@JoinColumn(name = "tags")

65. private Set<Tag> taglist;

66.

67. public Post() {

68. super();

69. // TODO Auto-generated constructor stub

70. }

71.

72. public Post(Long id\_post, String img\_name, String extension, Date date, Usuario usuario, Set<Type> typelist,

73. Set<Tag> taglist) {

74. super();

75. this.id\_post = id\_post;

76. this.img\_name = img\_name;

77. this.extension = extension;

78. this.date = date;

79. this.usuario = usuario;

80. this.typelist = typelist;

81. this.taglist = taglist;

82. }

83.

84. public Long getId\_post() {

85. return id\_post;

86. }

87.

88. public void setId\_post(Long id\_post) {

89. this.id\_post = id\_post;

90. }

91.

92. public String getImg\_name() {

93. return img\_name;

94. }

95.

96. public void setImg\_name(String img\_name) {

97. this.img\_name = img\_name;

98. }

99.

100. public String getExtension() {

101. return extension;

102. }

103.

104. public void setExtension(String extension) {

105. this.extension = extension;

106. }

107.

108. public Date getDate() {

109. return date;

110. }

111.

112. public void setDate(Date date) {

113. this.date = date;

114. }

115.

116. public Usuario getUsuario() {

117. return usuario;

118. }

119.

120. public void setUsuario(Usuario usuario) {

121. this.usuario = usuario;

122. }

123.

124. public Set<Type> getTypelist() {

125. return typelist;

126. }

127.

128. public void setTypelist(Set<Type> typelist) {

129. this.typelist = typelist;

130. }

131.

132. public Set<Tag> getTaglist() {

133. return taglist;

134. }

135.

136. public void setTaglist(Set<Tag> taglist) {

137. this.taglist = taglist;

138. }

139.

140. public static long getSerialversionuid() {

141. return serialVersionUID;

142. }

143.

144.

145. }

146.

Objeto Tags

1. package com.nib.app.model.entity;

2.

3. import java.io.Serializable;

4. import java.util.Set;

5.

6. import jakarta.persistence.Column;

7. import jakarta.persistence.Entity;

8. import jakarta.persistence.GeneratedValue;

9. import jakarta.persistence.GenerationType;

10. import jakarta.persistence.Id;

11. import jakarta.persistence.ManyToMany;

12. import jakarta.persistence.Table;

13.

14. /\*

15. \* Entidad de Tag

16. \*/

17. @Entity

18. @Table(name = "tag")

19. public class Tag implements Serializable {

20.

21. private static final long serialVersionUID = -8511788052263461642L;

22.

23. @Id

24. @GeneratedValue(strategy = GenerationType.IDENTITY)

25. @Column(name="id\_tag")

26. private Long id\_tag;

27.

28. private String name;

29.

30. @ManyToMany(mappedBy = "taglist")

31. // @JoinColumn(name = "posts")

32. private Set<Post> postList;

33.

34. public Tag() {

35. super();

36. // TODO Auto-generated constructor stub

37. }

38.

39. public Tag(Long id\_tag, String name, Set<Post> postList) {

40. super();

41. this.id\_tag = id\_tag;

42. this.name = name;

43. this.postList = postList;

44. }

45.

46. public Long getId\_tag() {

47. return id\_tag;

48. }

49.

50. public void setId\_tag(Long id\_tag) {

51. this.id\_tag = id\_tag;

52. }

53.

54. public String getName() {

55. return name;

56. }

57.

58. public void setName(String name) {

59. this.name = name;

60. }

61.

62. public Set<Post> getPostList() {

63. return postList;

64. }

65.

66. public void setPostList(Set<Post> postList) {

67. this.postList = postList;

68. }

69.

70. public static long getSerialversionuid() {

71. return serialVersionUID;

72. }

73.

74.

75.

76. }

77.

Objeto Type – No implementado y deprecado

1. package com.nib.app.model.entity;

2.

3. import java.io.Serializable;

4. import java.util.Set;

5.

6. import jakarta.persistence.Column;

7. import jakarta.persistence.Entity;

8. import jakarta.persistence.GeneratedValue;

9. import jakarta.persistence.GenerationType;

10. import jakarta.persistence.Id;

11. import jakarta.persistence.ManyToMany;

12. import jakarta.persistence.Table;

13.

14. /\*

15. \* Entidad de tipo de dato

16. \* // Entidad deprecada a eliminacion en futuras versiones

17. \*/

18. @Entity

19. @Table(name = "type")

20. public class Type implements Serializable {

21.

22. private static final long serialVersionUID = 8124193360669468165L;

23.

24. @Id

25. @GeneratedValue(strategy = GenerationType.IDENTITY)

26. @Column(name="id\_type")

27. private Long id\_type;

28.

29. private String name;

30. private String[] extensions;

31.

32.

33. @ManyToMany(mappedBy = "typelist")

34. // @JoinColumn(name = "posts")

35. private Set<Post> postList;

36.

37.

38. public Type() {

39. super();

40. // TODO Auto-generated constructor stub

41. }

42.

43.

44. public Type(Long id\_type, String name, String[] extensions, Set<Post> postList) {

45. super();

46. this.id\_type = id\_type;

47. this.name = name;

48. this.extensions = extensions;

49. this.postList = postList;

50. }

51.

52.

53. public Long getId\_type() {

54. return id\_type;

55. }

56.

57.

58. public void setId\_type(Long id\_type) {

59. this.id\_type = id\_type;

60. }

61.

62.

63. public String getName() {

64. return name;

65. }

66.

67.

68. public void setName(String name) {

69. this.name = name;

70. }

71.

72.

73. public String[] getExtensions() {

74. return extensions;

75. }

76.

77.

78. public void setExtensions(String[] extensions) {

79. this.extensions = extensions;

80. }

81.

82.

83. public Set<Post> getPostList() {

84. return postList;

85. }

86.

87.

88. public void setPostList(Set<Post> postList) {

89. this.postList = postList;

90. }

91.

92.

93. public static long getSerialversionuid() {

94. return serialVersionUID;

95. }

96.

97.

98.

99. }

100.

Servicio de Comentarios

1. package com.nib.app.model.impl;

2.

3. import java.util.ArrayList;

4. import java.util.Date;

5. import java.util.List;

6.

7. import org.springframework.beans.factory.annotation.Autowired;

8. import org.springframework.stereotype.Service;

9.

10. import com.nib.app.model.dao.usuario.CommentDAO;

11. import com.nib.app.model.entity.Comment;

12. import com.nib.app.model.entity.Post;

13. import com.nib.app.model.entity.user.Usuario;

14. import com.nib.app.model.service.CommentService;

15. import com.nib.app.model.service.PostService;

16. import com.nib.app.model.service.UsuarioService;

17. import com.nib.app.objects.FComment;

18.

19. /\*

20. \* Implementacion del servicio de comentarios

21. \*/

22. @Service

23. public class CommentServiceImpl implements CommentService {

24.

25. @Autowired

26. private CommentDAO commentDAO;

27.

28. @Autowired

29. private PostService postService;

30.

31. @Autowired

32. private UsuarioService usuarioService;

33.

34. /\*

35. \* Pre:

36. \* Post: Metodo para almacenar un comentario

37. \*/

38. @Override

39. public boolean saveComment(FComment comment) {

40. Comment c = new Comment();

41. c.setMessage(comment.getMessage());

42. c.setDate(new Date());

43. Usuario u = usuarioService.findUsernameByToken(comment.getToken\_usuario());

44. Post p = postService.getPostById(comment.getId\_post());

45. c.setPost(p);

46. c.setUsuario(u);

47. if(u == null || p == null) {

48. return false;

49. }

50. try {

51. commentDAO.save(c);

52. }catch (Exception e) {

53. System.out.println("[ERROR] - Error al guardar el comentario.");

54. return false;

55. }

56.

57. return true;

58. }

59.

60. /\*

61. \* Pre:

62. \* Post: Metodo el cual devuelve una lista de comentarios por su post

63. \*/

64. @Override

65. public List<FComment> getCommentsByPostId(Long postId) {

66. List<Comment> c = commentDAO.findCommentsByIdPost(postId);

67. List<FComment> fc = new ArrayList<FComment>();

68.

69. for (Comment comment : c) {

70. fc.add(new FComment(

71. comment.getId\_comment(),

72. "",

73. comment.getPost().getId\_post(),

74. comment.getUsuario().getNombre(),

75. comment.getUsuario().getImgProfile(),

76. comment.getMessage(),

77. comment.getDate()));

78. }

79. return fc;

80. }

81. }

82.

Servicio de configuración

1. package com.nib.app.model.impl;

2.

3. import java.util.List;

4.

5. import org.springframework.beans.factory.annotation.Autowired;

6. import org.springframework.stereotype.Service;

7.

8. import com.nib.app.model.dao.usuario.ConfigDAO;

9. import com.nib.app.model.entity.Config;

10. import com.nib.app.model.service.ConfService;

11.

12. /\*

13. \* Implementacion del servicio de configuracion

14. \*/

15. @Service

16. public class ConfServiceImpl implements ConfService {

17.

18. @Autowired

19. private ConfigDAO confdao;

20.

21. /\*

22. \* Pre:

23. \* Post: Metodo para obtener las configuraciones disponibles totales

24. \*/

25. @Override

26. public Long getCont() {

27. return confdao.count();

28. }

29.

30. /\*

31. \* Pre:

32. \* Post: Metodo para añadir una configuration

33. \*/

34. @Override

35. public void addConf(Config conf) {

36. confdao.save(conf);

37. }

38.

39. /\*

40. \* Pre:

41. \* Post: Metodo para obtener la configuracion por su tipo

42. \*/

43. @Override

44. public Config getConfbyConf(String conf) {

45. List<Config> c = confdao.findbyConfig(conf);

46. Config confe = null;

47. if(c.size() > 0) {

48. confe = c.get(0);

49. }

50. return confe;

51. }

52.

53. }

54.

Servicio de Posts

1. package com.nib.app.model.impl;

2.

3. import java.sql.SQLException;

4. import java.util.Base64;

5. import java.util.Collections;

6. import java.util.Date;

7. import java.util.HashSet;

8. import java.util.List;

9. import java.util.Optional;

10.

11. import org.springframework.beans.factory.annotation.Autowired;

12. import org.springframework.data.domain.Page;

13. import org.springframework.data.domain.PageImpl;

14. import org.springframework.data.domain.Pageable;

15. import org.springframework.stereotype.Service;

16.

17. import com.nib.app.model.dao.usuario.PostDAO;

18. import com.nib.app.model.entity.Post;

19. import com.nib.app.model.entity.Tag;

20. import com.nib.app.model.entity.user.Usuario;

21. import com.nib.app.model.service.PostService;

22. import com.nib.app.model.service.TagService;

23. import com.nib.app.model.service.UsuarioService;

24. import com.nib.app.objects.BinaryFile;

25. import com.nib.app.objects.Image;

26. import com.nib.app.utils.sqLite.SLUtils;

27.

28. /\*

29. \* Implementacion de Post Service

30. \*/

31. @Service

32. public class PostServiceImpl implements PostService {

33.

34. @Autowired

35. private PostDAO post\_dao;

36.

37. @Autowired

38. private UsuarioService usuario\_service;

39.

40. @Autowired

41. private TagService tag\_service;

42.

43. /\*

44. \* Pre:

45. \* Post: Metodo para obtener la cantidad de post existentes

46. \*/

47. @Override

48. public long getPostCount() {

49. return post\_dao.count();

50. }

51.

52. /\*

53. \* Pre:

54. \* Post: Metodo el cual obtene las paginas de los posts

55. \*/

56. @Override

57. public Page<Post> getPaginasPosts(Pageable pageable) {

58. return post\_dao.findAll(pageable);

59. }

60.

61. /\*

62. \* Pre:

63. \* Post: Metodo el cual almacena un Post el las bases de datos

64. \*/

65. @Override

66. public boolean savePost(Image image) {

67. String[] base64\_str = image.getBase64Img().split("base64,");

68. if(base64\_str.length == 2) {

69. //System.out.println(image.getToken());

70. try {

71. byte[] imageBytes = Base64.getDecoder().decode(base64\_str[1]);

72. Usuario u = usuario\_service.findUsernameByToken(image.getToken());

73. if(u != null) {

74. Post p = new Post();

75. p.setUsuario(u);

76. p.setExtension(base64\_str[0].replaceAll("data", "").replaceAll(":", "").replaceAll(";", ""));

77. p.setImg\_name(image.getTitle());

78. p.setDate(new Date());

79. p.setTaglist(new HashSet<Tag>());

80. Post ps = savePostDatBBDD(p);

81. if(ps != null) {

82. String[] tags = image.getTags().split(",");

83. for(String tag : tags) {

84. tag = tag.replaceAll(" ", "").replaceAll(";", "").toLowerCase();

85. Tag tg = tag\_service.findByName(tag);

86. if(tg != null) {

87. ps.getTaglist().add(tg);

88. }else {

89. if(tag != "") { // Evitamos crear un tag sin nada

90. Tag ntag = new Tag();

91. ntag.setName(tag);

92. Tag stag = tag\_service.saveTag(ntag);

93. if(stag != null) {//entra en stag != null

94. ps.getTaglist().add(stag);

95. }

96. }

97. }

98. }

99. Post ps2 = savePostDatBBDD(ps);

100. // codigo almacenar en la localBBDD

101. //System.out.println(ps2.getId\_post());

102. if(ps2 != null) {

103. savePostLocalBBDD(imageBytes,ps2.getId\_post(),u.getUsername());

104. }

105. return true;

106. }

107. }else {

108. System.err.println("[ERROR] [si831] - user null");

109. }

110. }catch(Exception e) {

111. System.err.println("[ERROR] [si001] - Error al decodificar la base de una imagen \n" + e.toString());

112. }

113. //System.out.println("Base ok");

114.

115. }

116. return false;

117. }

118.

119. /\*

120. \* Pre:

121. \* Post: Metodo para almacenar los binarios en la bbdd local

122. \*/

123. private boolean savePostLocalBBDD(byte[] imageBytes,Long id\_img,String username) {

124. SLUtils s = new SLUtils();

125. s.getConnection(username);

126. try {

127. s.savePost(imageBytes, id\_img);

128. return true;

129. } catch (SQLException e) {

130. e.printStackTrace();

131. return false;

132. }

133.

134. }

135.

136. /\*

137. \* Pre:

138. \* Post: Metodo para almacenar los datos de los posts en la base de datos

139. \*/

140. @Override

141. public Post savePostDatBBDD(Post post) {

142. try {

143. return post\_dao.save(post);

144. }catch(Exception e) {

145. return null;

146. }

147.

148. }

149.

150. /\*

151. \* Pre:

152. \* Post: Metodo para Obtener los binarios por el id del post

153. \*/

154. @Override

155. public BinaryFile getImageById\_Post(Long id) {

156. Optional<Post> p = post\_dao.findById(id);

157. if(!p.isEmpty()) {

158. BinaryFile f = new BinaryFile();

159.

160. Post post = p.get();

161. f.setFormat(post.getExtension());

162.

163. SLUtils u = new SLUtils();

164. u.getConnection(post.getUsuario().getUsername());

165. byte[] b = u.getPostByIdPost(id);

166. f.setData(b);

167.

168. return f;

169. }else {

170. return null;

171. }

172. }

173.

174. /\*

175. \* Pre:

176. \* Post: Metodo para obtener los posts filtrados por los tags

177. \*/

178. @Override

179. public Page<Post> getPaginasToPostsByTag(Pageable pageable, String tag) {

180. List<Post> posts = post\_dao.findPostsByTag(tag);

181. int start = (int) pageable.getOffset();

182. if (start >= posts.size()) { // Evitamos bug de datos inexistentes

183. return new PageImpl<>(Collections.emptyList(), pageable, 0);

184. }

185. int end = Math.min((start + pageable.getPageSize()), posts.size());

186. Page<Post> pages = new PageImpl<>(posts.subList(start, end), pageable, posts.size());

187. return pages;

188. }

189.

190. /\*

191. \* Pre:

192. \* Post: Metodo para Obtener la info de un post post su id

193. \*/

194. @Override

195. public Post getPostById(Long id) {

196. Optional<Post> p = post\_dao.findById(id);

197. if(p.isPresent()) {

198. return p.get();

199. }else {

200. return null;

201. }

202. }

203. }

204.

Servicio de Roles

1. package com.nib.app.model.impl;

2.

3. import java.util.ArrayList;

4. import java.util.List;

5.

6. import org.springframework.beans.factory.annotation.Autowired;

7. import org.springframework.stereotype.Service;

8.

9. import com.nib.app.model.dao.usuario.RolDAO;

10. import com.nib.app.model.entity.user.Rol;

11. import com.nib.app.model.service.RolService;

12.

13. /\*

14. \* Implementacion del servicio de roles

15. \*/

16. @Service

17. public class RolServiceImpl implements RolService {

18.

19. @Autowired

20. private RolDAO roldao;

21.

22. /\*

23. \* Pre:

24. \* Post: Metodo con el cual almacenamos un rol en la BBDD

25. \*/

26. @Override

27. public Rol saveRol(Rol rol) {

28. Rol localRol = findByName(rol);

29.

30. if(localRol != null) {

31. System.err.println("[ERROR] [si003] - El Rol a insertar ya existe");

32. }else {

33. try {

34. localRol = roldao.save(rol);

35. }catch(Exception e) {

36. System.out.println("[ERROR] [si004] - Error al insertar un rol");

37. }

38. }

39. return localRol;

40. }

41.

42. /\*

43. \* Pre:

44. \* Post: Metodo el cual devuelve una lista de roles

45. \*/

46. @Override

47. public List<Rol> saveRolList(List<Rol> roles) {

48. List<Rol> localRoles = new ArrayList<Rol>();

49. System.out.println("Hola");

50. System.out.println(roles.size());

51. for(int i = 0; i < roles.size();i++) {

52. localRoles.add(i, this.saveRol(roles.get(i)));

53. }

54. return localRoles;

55. }

56.

57. /\*

58. \* Pre:

59. \* Post: Metodo el cual busca un rol por su nombre

60. \* con un objeto de tipo Rol

61. \*/

62. @Override

63. public Rol findByName(Rol rol) {

64. Rol r = findByName(rol.getNombre());

65. if(r == null) {

66. return null;

67. }else {

68. return r;

69. }

70. }

71.

72. /\*

73. \* Pre:

74. \* Post: Metodo el cual busca un rol por su nombre

75. \*/

76. @Override

77. public Rol findByName(String name) {

78. Rol r = roldao.findByNombre(name);

79. if(r == null) {

80. return null;

81. }else {

82. return r;

83. }

84. }

85.

86. /\*

87. \* Pre:

88. \* Post: Metodo el cual verifica si existe un rol

89. \*/

90. @Override

91. public boolean isExistsRolByName(Rol rol) {

92. if(findByName(rol) != null) {

93. return true;

94. }else {

95. return false;

96. }

97. }

98.

99. /\*

100. \* Pre:

101. \* Post: Metodo el cual devuelve el rol default

102. \*/

103. @Override

104. public Rol getDefaultRol() {

105. Rol r = findByName("USER");

106. if(r == null) {

107. return null;

108. }else {

109. return r;

110. }

111. }

112.

113. /\*

114. \* Pre:

115. \* Post: Metodo el cual verifica si un usuario es administrador

116. \*/

117. @Override

118. public boolean isthisUserAdminByToken(String token) {

119. boolean isAdmin = false;

120. List<Rol> r = roldao.findRolesByUserToken(token);

121. for(int i = 0; i < r.size(); i++) {

122. if(r.get(i).getNombre().equalsIgnoreCase("ADMIN")) {

123. isAdmin = true;

124. }

125. }

126. return isAdmin;

127. }

128.

129.

130.

131. }

132.

Servicio de Tags

1. package com.nib.app.model.impl;

2.

3. import java.util.List;

4. import java.util.Optional;

5.

6. import org.springframework.beans.factory.annotation.Autowired;

7. import org.springframework.data.domain.Page;

8. import org.springframework.data.domain.Pageable;

9. import org.springframework.stereotype.Service;

10.

11. import com.nib.app.model.dao.usuario.TagDAO;

12. import com.nib.app.model.entity.Tag;

13. import com.nib.app.model.service.TagService;

14.

15. /\*

16. \* Implementacion de servicio de tags

17. \*/

18.

19. @Service

20. public class TagServiceImpl implements TagService {

21.

22. @Autowired

23. private TagDAO tagdao;

24.

25. /\*

26. \* Pre:

27. \* Post: Metodo con el cual obtenemos tags aleatorias

28. \*/

29. @Override

30. public List<Tag> getRandomTags(int limit) {

31. return tagdao.getRandomerTag(limit);

32. }

33.

34. /\*

35. \* Pre:

36. \* Post: Metodo con el cual obtenemos un page de tags

37. \*/

38. @Override

39. public Page<Tag> getPaginasTags(Pageable pageable) {

40. return tagdao.findAll(pageable);

41.

42. }

43.

44. /\*

45. \* Pre:

46. \* Post: Metodo con el cual buscamos una tag por su nombre

47. \*/

48. @Override

49. public Tag findByName(String name) {

50. Optional<Tag> tag = tagdao.findByName(name);

51. if(!tag.isEmpty()) {

52. return tag.get();

53. }else {

54. return null;

55. }

56.

57. }

58.

59. /\*

60. \* Pre:

61. \* Post: Metodo con el cual almacenamos una tag

62. \*/

63. @Override

64. public Tag saveTag(Tag tag) {

65. try {

66. return tagdao.save(tag);

67. } catch (Exception e) {

68. System.err.println("[ERROR] [si001] - Error al guardar un tag \n");

69. return null;

70. }

71. }

72.

73. /\*

74. \* Pre:

75. \* Post: Metodo con el cual obenemos las tags de un post

76. \*/

77. @Override

78. public List<Tag> getTagsByIdPost(Long id) {

79. List<Tag> tags = tagdao.getTagsByPostId(id);

80. return tags;

81. }

82.

83. }

84.

Servicio de Usuario

1. package com.nib.app.model.impl;

2.

3. import java.sql.SQLException;

4. import java.util.Optional;

5.

6. import org.springframework.beans.factory.annotation.Autowired;

7. import org.springframework.stereotype.Service;

8.

9. import com.nib.app.model.dao.usuario.UsuarioDAO;

10. import com.nib.app.model.entity.user.Usuario;

11. import com.nib.app.model.service.UsuarioService;

12. import com.nib.app.utils.PasswordEncryptor;

13. import com.nib.app.utils.TokenGen;

14. import com.nib.app.utils.sqLite.SLUtils;

15.

16. /\*

17. \* Implementacion del servicio de usuario

18. \*/

19.

20. @Service

21. public class UsuarioServiceImpl implements UsuarioService {

22.

23. /\*

24. \* Datos de configuracion

25. \*/

26. private final String TOKEN\_INIT = "NIB\_TK\_";

27. private final int TOKEN\_MAX = 100;

28.

29. @Autowired

30. private UsuarioDAO usuarioDAO;

31.

32. /\*

33. \* Metodo para registrar un usuario en la BBDD

34. \*/

35. @Override

36. public boolean saveUsuario(Usuario usuario){

37. Usuario localUser = findByUsername(usuario.getUsername());

38. if(localUser != null) {

39. localUser.setPassword(null);

40. localUser.setToken(null);

41. System.err.println("[ERROR] [si002] - El usuario a insertar ya existe");

42. return false;

43. }else{

44. TokenGen t = new TokenGen();

45. PasswordEncryptor encryptor = new PasswordEncryptor();

46. //System.out.println(usuario.toString());

47. usuario.setToken(this.TOKEN\_INIT+usuario.getUsername()+"\_"+t.generate(TOKEN\_MAX));

48. try {

49. createBinaryData(usuario.getUsername());

50. } catch (SQLException e1) {

51. // TODO Auto-generated catch block

52. e1.printStackTrace();

53. }

54. usuario.setPassword(encryptor.encrypt(usuario.getPassword()));

55. usuario.setVerify(true); // A Modificar en versiones posteriores

56. //System.out.println(usuario.toString());

57. try {

58. usuarioDAO.save(usuario);

59. }catch(Exception e) {

60. System.err.println("[ERROR] [si001] - Error al añadir un usuario \n" + e.toString());

61. }

62. }

63. return true;

64. }

65.

66. /\*

67. \* Metodo para crear la base de datos y sus tablas por usuario

68. \*/

69. private void createBinaryData(String id) throws SQLException {

70. SLUtils u = new SLUtils();

71. u.getConnection(id);

72. u.createTables();

73. }

74.

75. /\*

76. \* Pre:

77. \* Post: Metodo para buscar un usuario por su

78. \* nombre de usuario.

79. \*/

80. @Override

81. public Usuario findByUsername(String usuario) {

82. return usuarioDAO.findByUsername(usuario);

83. }

84.

85. /\*

86. \* Pre:

87. \* Post: Metodo para buscar un usuario por su nombreç

88. \* de usuario pero con un obrjeto de tipo usuario

89. \*/

90. @Override

91. public Usuario findByUsername(Usuario usuario) {

92. return findByUsername(usuario.getUsername());

93. }

94.

95. /\*

96. \* Pre:

97. \* Post: Metodo con el cualverificamos si existe un usuario

98. \* filtrado por su nombre de usuario.

99. \*/

100. @Override

101. public boolean isExistsUsernameByUsername(Usuario usuario) {

102. if(findByUsername(usuario) != null) {

103. return true;

104. }else {

105. return false;

106. }

107. }

108.

109. /\*

110. \* Pre:

111. \* Post: Metodo con el cual eliminamos un usuario de la bbdd

112. \*/

113. @Override

114. public boolean deleteUser(Usuario usuario) {

115. try {

116. deleteUserById(usuario.getId\_usuario());

117. return true;

118. }catch(Exception e) {

119. return false;

120. }

121.

122. }

123.

124. /\*

125. \* Pre:

126. \* Post: Metodo con el cual eliminamos un usuario

127. \* por su id de usuario

128. \*/

129. @Override

130. public boolean deleteUserById(Long id) {

131. try {

132. usuarioDAO.deleteById(id);

133. return true;

134. }catch(Exception e) {

135. return false;

136. }

137.

138. }

139.

140.

141. /\*

142. \* Pre:

143. \* Post: Metodo con el cual logueamos un usuario

144. \*/

145. @Override

146. public String login(Usuario usuario) {

147. Usuario localUser = findByUsername(usuario.getUsername());

148. String token = "ERROR";

149. if(localUser != null) {

150. PasswordEncryptor encrypt = new PasswordEncryptor();

151. if(encrypt.checkPassword(usuario.getPassword(), localUser.getPassword())) {

152. token = localUser.getToken();

153. return token;

154. }

155. System.err.println("[AVISO] [av032] - Usuario credenciales erroneas \n");

156. }else {

157. System.err.println("[AVISO] [av032] - Usuario Inexistente \n");

158. }

159. return token;

160. }

161.

162. /\*

163. \* Pre:

164. \* Post: Metodo con el cual verificamos un token del usuario

165. \*/

166. @Override

167. public boolean verifyToken(String token) {

168. Optional<Usuario> us = usuarioDAO.findByToken(token);

169. if(!us.isEmpty()) {

170. return true;

171. }else {

172. return false;

173. }

174. }

175.

176. /\*

177. \* Pre:

178. \* Post: Metodo con el cual buscamos un usuario por su token

179. \*/

180. @Override

181. public Usuario findUsernameByToken(String token) {

182. Optional<Usuario> us = usuarioDAO.findByToken(token);

183. if(!us.isEmpty()) {

184. return us.get();

185. }else {

186. return null;

187. }

188. }

189.

190. /\*

191. \* Pre:

192. \* Post: Metodo con el cual actualizamos a un usuario

193. \*/

194. @Override

195. public boolean updateUser(Usuario usuario) {

196. Usuario localUser = findByUsername(usuario.getUsername());

197. if(localUser != null) {

198. try {

199. usuarioDAO.save(usuario);

200. return true;

201. }catch(Exception e) {

202. System.err.println("[ERROR] [si001] - Error al añadir un usuario \n" + e.toString());

203. return false;

204. }

205. }else{

206. return false;

207. }

208. }

209. }

210.

Controlador de autentificación

1. package com.nib.app.controllers.api.auth;

2.

3. import org.springframework.beans.factory.annotation.Autowired;

4. import org.springframework.http.HttpStatus;

5. import org.springframework.http.ResponseEntity;

6. import org.springframework.web.bind.annotation.CrossOrigin;

7. import org.springframework.web.bind.annotation.PostMapping;

8. import org.springframework.web.bind.annotation.RequestBody;

9. import org.springframework.web.bind.annotation.RequestMapping;

10. import org.springframework.web.bind.annotation.RequestMethod;

11. import org.springframework.web.bind.annotation.RestController;

12.

13. import com.nib.app.model.entity.user.Usuario;

14. import com.nib.app.model.impl.UsuarioServiceImpl;

15. import com.nib.app.model.service.RolService;

16. import com.nib.app.objects.PO;

17.

18. @RestController

19. @RequestMapping("/api/auth")

20. @CrossOrigin(origins = {"\*"},methods= {RequestMethod.GET,RequestMethod.POST})

21. public class AuthController {

22.

23. @Autowired

24. private UsuarioServiceImpl usuarioServiceImpl;

25.

26. @Autowired

27. private RolService rolService;

28.

29. /\*

30. \* Pre:

31. \* Post: Metodo para autenetificar y registrar un usuario

32. \*/

33. @PostMapping("/register")

34. public ResponseEntity<String> authRegister(@RequestBody Usuario usuario) {

35. try {

36. usuario.setId\_usuario(null);

37. usuario.setToken(null);

38. usuario.setVerify(true);

39. usuario.setImgProfile("image.png");

40. boolean u = usuarioServiceImpl.saveUsuario(usuario);

41. if(u == false) {

42. //System.out.println("Hola");

43. return new ResponseEntity<>(formatJson("ERROR"), HttpStatus.OK);

44.

45. }

46. //System.out.println(usuario.toString());

47.

48. return new ResponseEntity<>(formatJson("OK"), HttpStatus.OK);

49. } catch (Exception e) {

50. return new ResponseEntity<>(formatJson("ERROR"), HttpStatus.INTERNAL\_SERVER\_ERROR);

51. }

52. }

53.

54. /\*

55. \* Pre:

56. \* Post: metodo para loguear a un usuario

57. \*/

58. @PostMapping("/login")

59. public ResponseEntity<String> authLogin(@RequestBody Usuario usuario) {

60. try {

61. String token = usuarioServiceImpl.login(usuario);

62. return new ResponseEntity<>(formatJson(token), HttpStatus.OK);

63. } catch (Exception e) {

64. return new ResponseEntity<>(formatJson("ERR://401/UNATHORIZED"), HttpStatus.INTERNAL\_SERVER\_ERROR);

65. }

66. }

67.

68. /\*

69. \* Pre:

70. \* Post: Metodo para verificar el token de un usuario

71. \*/

72. @PostMapping("/verifyToken")

73. public ResponseEntity<String> veifyToken(@RequestBody PO token){

74. try {

75. //System.out.println(token.getToken());

76. boolean result = usuarioServiceImpl.verifyToken(token.getToken());

77. return new ResponseEntity<String>(formatJson(result), HttpStatus.OK);

78. } catch (Exception e) {

79. return new ResponseEntity<>(formatJson(false), HttpStatus.INTERNAL\_SERVER\_ERROR);

80. }

81. }

82.

83. public String formatJson(boolean value) {return formatJson(value+"");}

84. public String formatJson(String value) {

85. return "{\"value\":\""+value+"\"}";

86. }

87.

88. /\*

89. \* Pre:

90. \* Post: Metodo para verificar a un administrador

91. \*/

92. @PostMapping("/verifyAdmin")

93. public ResponseEntity<String> veifyAdmin(@RequestBody PO token){

94. try {

95. boolean result = rolService.isthisUserAdminByToken(token.getToken());

96. //System.out.println(result);

97. return new ResponseEntity<String>(formatJson(result), HttpStatus.OK);

98. } catch (Exception e) {

99. return new ResponseEntity<>(formatJson(false), HttpStatus.INTERNAL\_SERVER\_ERROR);

100. }

101. }

102.

103. /\*

104. \* Nota:

105. \* En futuras verisiones se añadira el reseteo de token.

106. \*/

107.

108. }

109.

Controlador de posts – con métodos de auth integrados y privados

1. package com.nib.app.controllers.api.auth;

2.

3. import java.util.List;

4. import java.util.regex.Pattern;

5.

6. import org.springframework.beans.factory.annotation.Autowired;

7. import org.springframework.http.HttpStatus;

8. import org.springframework.http.ResponseEntity;

9. import org.springframework.web.bind.annotation.CrossOrigin;

10. import org.springframework.web.bind.annotation.GetMapping;

11. import org.springframework.web.bind.annotation.PostMapping;

12. import org.springframework.web.bind.annotation.RequestBody;

13. import org.springframework.web.bind.annotation.RequestMapping;

14. import org.springframework.web.bind.annotation.RequestMethod;

15. import org.springframework.web.bind.annotation.RequestParam;

16. import org.springframework.web.bind.annotation.RestController;

17.

18. import com.nib.app.model.entity.Post;

19. import com.nib.app.model.entity.Tag;

20. import com.nib.app.model.service.PostService;

21. import com.nib.app.model.service.TagService;

22. import com.nib.app.objects.Image;

23.

24. @RestController

25. @RequestMapping("/api/private/post")

26. @CrossOrigin(origins = {"\*"},methods= {RequestMethod.GET,RequestMethod.POST})

27. public class PostController {

28.

29. @Autowired

30. private PostService postService;

31.

32. @Autowired

33. private TagService tagService;

34.

35. /\*

36. \* Pre:

37. \* Post: Metodo para crear un nuevo post

38. \*/

39. @PostMapping("/postnew")

40. public ResponseEntity<?> create(

41. @RequestBody Image file

42. ){

43. try {

44. boolean result = postService.savePost(file);

45. return new ResponseEntity<Boolean>(result,HttpStatus.CREATED);

46. } catch (Exception e) {

47. // TODO Auto-generated catch block

48. e.printStackTrace();

49. return new ResponseEntity<Boolean>(false,HttpStatus.CREATED);

50. }

51. }

52.

53. /\*

54. \* Pre:

55. \* Post: Metodo para obtener la informacion de un post

56. \*/

57. @GetMapping("/getPostInfo")

58. public ResponseEntity<?> getPostInfo(@RequestParam("id\_post") String id){

59. try {

60. if (Pattern.matches("-?\\d+", id)) {

61. Long identificador = Long.parseLong(id);

62. Post p = postService.getPostById(identificador);

63. return new ResponseEntity<Post>(p,HttpStatus.CREATED);

64. } else {

65. System.out.println("La variable no es un número.");

66.

67. return new ResponseEntity<Boolean>(false,HttpStatus.CREATED);

68. }

69.

70. } catch (Exception e) {

71. // TODO Auto-generated catch block

72. e.printStackTrace();

73. return new ResponseEntity<Boolean>(false,HttpStatus.CREATED);

74. }

75. }

76.

77. /\*

78. \* Pre:

79. \* Post: Metodo para obtener las tags de un post

80. \*/

81. @GetMapping("/getTagsByPost")

82. public ResponseEntity<?> getTagsByPost(@RequestParam("id\_post") String id){

83. try {

84. if (Pattern.matches("-?\\d+", id)) {

85. Long identificador = Long.parseLong(id);

86. List<Tag> t = tagService.getTagsByIdPost(identificador);

87. return new ResponseEntity<List<Tag>>(t,HttpStatus.CREATED);

88. } else {

89. System.out.println("La variable no es un número.");

90.

91. return new ResponseEntity<Boolean>(false,HttpStatus.CREATED);

92. }

93.

94. } catch (Exception e) {

95. // TODO Auto-generated catch block

96. e.printStackTrace();

97. return new ResponseEntity<Boolean>(false,HttpStatus.CREATED);

98. }

99. }

100. }

101.

Controlador de devolución de imágenes

1. package com.nib.app.controllers.api.provPrivate;

2.

3. import java.io.IOException;

4. import java.nio.file.Files;

5. import java.nio.file.Paths;

6.

7. import org.springframework.beans.factory.annotation.Autowired;

8. import org.springframework.http.HttpHeaders;

9. import org.springframework.http.HttpStatus;

10. import org.springframework.http.MediaType;

11. import org.springframework.http.ResponseEntity;

12. import org.springframework.web.bind.annotation.CrossOrigin;

13. import org.springframework.web.bind.annotation.GetMapping;

14. import org.springframework.web.bind.annotation.PathVariable;

15. import org.springframework.web.bind.annotation.RequestMapping;

16. import org.springframework.web.bind.annotation.RequestMethod;

17. import org.springframework.web.bind.annotation.RestController;

18.

19. import com.nib.app.model.service.PostService;

20. import com.nib.app.objects.BinaryFile;

21.

22. @RestController

23. @RequestMapping("/api/image")

24. @CrossOrigin(origins = {"\*"},methods= {RequestMethod.GET,RequestMethod.POST})

25. public class ImageController {

26.

27. @Autowired

28. private PostService postService;

29.

30. /\*

31. \* Pre:

32. \* Post: Metodo para obtener una imagen de la base de datos

33. \*/

34. @GetMapping("/get/{id\_image}")

35. public ResponseEntity<?> getImage(

36. @PathVariable("id\_image") Long id\_image

37. ) {

38. try {

39. BinaryFile f = postService.getImageById\_Post(id\_image);

40. // Carga la imagen desde un archivo o una base de datos

41. byte[] imagenBytes = f.getData();

42.

43. if(imagenBytes == null) {

44. return new ResponseEntity<>(false, HttpStatus.BAD\_REQUEST);

45. }

46.

47. // Crea una respuesta HTTP con la imagen y los encabezados adecuados

48. HttpHeaders headers = new HttpHeaders();

49. if(f.getFormat().contains("png")) { // Verifiacion de formato

50. headers.setContentType(MediaType.IMAGE\_PNG);

51. }else if(f.getFormat().contains("jpg")) {

52. headers.setContentType(MediaType.IMAGE\_JPEG);

53. }else if(f.getFormat().contains("gif")) {

54. headers.setContentType(MediaType.IMAGE\_GIF);

55. }else {

56. headers.setContentType(MediaType.IMAGE\_PNG);

57. }

58.

59. headers.setContentLength(imagenBytes.length);

60.

61. return new ResponseEntity<byte[]>(imagenBytes, headers, HttpStatus.OK);

62.

63. } catch (IllegalArgumentException e) {

64. return new ResponseEntity<>(false, HttpStatus.BAD\_REQUEST);

65. }

66. }

67. /\*

68. \*

69. // Metodo deprecado

70. @PostMapping("/save")

71. public ResponseEntity<?> saveImage(@RequestBody PO data) {

72. try {

73. byte[] imageBytes = Base64.getDecoder().decode(data.getData());

74. // Aquí se puede guardar la imagen en el servidor o en una base de datos

75.

76.

77. return new ResponseEntity<>("Imagen guardada correctamente", HttpStatus.OK);

78. } catch (IllegalArgumentException e) {

79. return new ResponseEntity<>("La imagen no está en formato Base64", HttpStatus.BAD\_REQUEST);

80. }

81. }

82. \*/

83. }

84.

Controlador de configuración sistemática

1. package com.nib.app.controllers.api.provPrivate;

2.

3. import org.springframework.beans.factory.annotation.Autowired;

4. import org.springframework.http.HttpStatus;

5. import org.springframework.http.ResponseEntity;

6. import org.springframework.web.bind.annotation.CrossOrigin;

7. import org.springframework.web.bind.annotation.PostMapping;

8. import org.springframework.web.bind.annotation.RequestBody;

9. import org.springframework.web.bind.annotation.RequestMapping;

10. import org.springframework.web.bind.annotation.RequestMethod;

11. import org.springframework.web.bind.annotation.RestController;

12.

13. import com.nib.app.model.entity.Config;

14. import com.nib.app.model.service.ConfService;

15. import com.nib.app.model.service.RolService;

16. import com.nib.app.objects.SConf;

17.

18. @RestController

19. @RequestMapping("/api/private/conf")

20. @CrossOrigin(origins = {"\*"},methods= {RequestMethod.GET,RequestMethod.POST})

21. public class PrivateConfController {

22.

23. @Autowired

24. private RolService rolService;

25.

26. @Autowired

27. private ConfService configService;

28.

29. /\*

30. \* Pre:

31. \* Post: Metodo para cambiar la configuración del sistema

32. \*/

33. @PostMapping("/change")

34. public ResponseEntity<?> changeConf(

35. @RequestBody SConf sconf

36. ){

37. try {

38. String token = sconf.getToken();

39. boolean result = rolService.isthisUserAdminByToken(token);

40. if(result) {

41. Config appNameLocal = configService.getConfbyConf("app\_name");

42. Config appShortNameLocal = configService.getConfbyConf("app\_short\_name");

43. appNameLocal.setValue(sconf.getAppname());

44. appShortNameLocal.setValue(sconf.getAppshortname());

45. configService.addConf(appNameLocal);

46. configService.addConf(appShortNameLocal);

47. return new ResponseEntity<Boolean>(true,HttpStatus.CREATED);

48. }

49. return new ResponseEntity<Boolean>(false,HttpStatus.CREATED);

50. } catch (Exception e) {

51. e.printStackTrace();

52. return new ResponseEntity<Boolean>(false,HttpStatus.CREATED);

53. }

54. }

55. }

56.

Controlador de comentarios

1. package com.nib.app.controllers.api.provPublic;

2.

3. import java.util.List;

4.

5. import org.springframework.beans.factory.annotation.Autowired;

6. import org.springframework.http.HttpStatus;

7. import org.springframework.http.ResponseEntity;

8. import org.springframework.web.bind.annotation.CrossOrigin;

9. import org.springframework.web.bind.annotation.PostMapping;

10. import org.springframework.web.bind.annotation.RequestBody;

11. import org.springframework.web.bind.annotation.RequestMapping;

12. import org.springframework.web.bind.annotation.RequestMethod;

13. import org.springframework.web.bind.annotation.RestController;

14.

15. import com.nib.app.model.entity.Post;

16. import com.nib.app.model.service.CommentService;

17. import com.nib.app.objects.FComment;

18.

19. @RestController

20. @RequestMapping("/api/public/comments")

21. @CrossOrigin(origins = {"\*"},methods= {RequestMethod.GET,RequestMethod.POST})

22. public class CommentsController {

23.

24. @Autowired

25. private CommentService commentService;

26.

27. /\*

28. \* Pre:

29. \* Post: Controlador para almacenar un nuevo comentario

30. \*/

31. @PostMapping("/save")

32. public ResponseEntity<?> create(

33. @RequestBody FComment fcomment

34. ){

35. try {

36. boolean result = commentService.saveComment(fcomment);

37. return new ResponseEntity<Boolean>(result,HttpStatus.CREATED);

38. } catch (Exception e) {

39. // TODO Auto-generated catch block

40. e.printStackTrace();

41. return new ResponseEntity<Boolean>(false,HttpStatus.CREATED);

42. }

43. }

44.

45. /\*

46. \* Pre:

47. \* Post: Metodo para obtener una lista de comnetarios de un post

48. \*/

49. @PostMapping("/get")

50. public ResponseEntity<?> getComments(

51. @RequestBody Post p

52. ){

53. try {

54. //System.out.println(p.getId\_post());

55. List<FComment> comments = commentService.getCommentsByPostId(p.getId\_post());

56. //System.out.println("XD" );

57. /\*

58. for (FComment comment : comments) {

59. System.out.println(comment.toString());

60. }\*/

61. //boolean result = commentService.saveComment(fcomment);

62. return new ResponseEntity<List<FComment>>(comments,HttpStatus.CREATED);

63. } catch (Exception e) {

64. e.printStackTrace();

65. return new ResponseEntity<Boolean>(false,HttpStatus.CREATED);

66. }

67. }

68. }

69.

Controlador de acceso a configuración publica

1. package com.nib.app.controllers.api.provPublic;

2.

3. import org.springframework.beans.factory.annotation.Autowired;

4. import org.springframework.http.HttpStatus;

5. import org.springframework.http.ResponseEntity;

6. import org.springframework.web.bind.annotation.CrossOrigin;

7. import org.springframework.web.bind.annotation.GetMapping;

8. import org.springframework.web.bind.annotation.RequestMapping;

9. import org.springframework.web.bind.annotation.RequestMethod;

10. import org.springframework.web.bind.annotation.RequestParam;

11. import org.springframework.web.bind.annotation.RestController;

12.

13. import com.nib.app.model.entity.Config;

14. import com.nib.app.model.service.ConfService;

15.

16. @RestController

17. @RequestMapping("/api/public/conf")

18. @CrossOrigin(origins = {"\*"},methods= {RequestMethod.GET,RequestMethod.POST})

19. public class PublicConf {

20.

21. @Autowired

22. private ConfService confserv;

23.

24. /\*

25. \* Pre:

26. \* Post: Metodo el cual devuelve el nombre de la aplicación

27. \* // Metodo deperecado a eliminar en futuras versiones

28. \*/

29. @GetMapping("/name")

30. public ResponseEntity<String> getName() {

31. try {

32. String name = "Nine Image Board";

33. Config conf = confserv.getConfbyConf("app\_name");

34. if(conf != null) {

35. name = conf.getValue();

36. }

37. return new ResponseEntity<>("\""+name+"\"", HttpStatus.OK);

38. } catch (Exception e) {

39. return new ResponseEntity<>("", HttpStatus.INTERNAL\_SERVER\_ERROR);

40. }

41. }

42.

43. /\*

44. \* Pre:

45. \* Post: Metodo el cual obtiene una configuracion de la aplicacion

46. \*/

47. @GetMapping("/get")

48. public ResponseEntity<String> getConf(@RequestParam(defaultValue = "app\_name") String type) {

49. try {

50. String conf\_String = "ERR://GET:CONF";

51. Config conf = confserv.getConfbyConf(type);

52. if(conf != null) {

53. conf\_String = conf.getValue();

54. }

55. return new ResponseEntity<>("\""+conf\_String+"\"", HttpStatus.OK);

56. } catch (Exception e) {

57. return new ResponseEntity<>("", HttpStatus.INTERNAL\_SERVER\_ERROR);

58. }

59. }

60.

61. }

62.

Controlador de acceso de posts y publico

1. package com.nib.app.controllers.api.provPublic;

2.

3. import org.springframework.beans.factory.annotation.Autowired;

4. import org.springframework.data.domain.Page;

5. import org.springframework.data.domain.PageRequest;

6. import org.springframework.data.domain.Sort;

7. import org.springframework.http.HttpStatus;

8. import org.springframework.http.ResponseEntity;

9. import org.springframework.web.bind.annotation.CrossOrigin;

10. import org.springframework.web.bind.annotation.GetMapping;

11. import org.springframework.web.bind.annotation.PathVariable;

12. import org.springframework.web.bind.annotation.RequestMapping;

13. import org.springframework.web.bind.annotation.RequestMethod;

14. import org.springframework.web.bind.annotation.RequestParam;

15. import org.springframework.web.bind.annotation.RestController;

16.

17. import com.nib.app.config.NibConfig;

18. import com.nib.app.model.entity.Post;

19. import com.nib.app.model.entity.user.Usuario;

20. import com.nib.app.model.service.PostService;

21.

22. @RestController

23. @RequestMapping("/api/public/posts")

24. @CrossOrigin(origins = {"\*"},methods= {RequestMethod.GET,RequestMethod.POST})

25. public class PublicPosts {

26.

27. @Autowired

28. private PostService post\_serv;

29.

30. /\*

31. \* Pre:

32. \* Post: metodo el cual devuelve el numero de posts

33. \*/

34. @GetMapping("/count")

35. public ResponseEntity<Long> getCount() {

36. try {

37. return new ResponseEntity<>(post\_serv.getPostCount(), HttpStatus.OK);

38. } catch (Exception e) {

39. return new ResponseEntity<>(0L, HttpStatus.INTERNAL\_SERVER\_ERROR);

40. }

41. }

42.

43. /\*

44. \* Pre:

45. \* post: Metodo el cual devuelve los posts paginados

46. \*/

47. @GetMapping("/postsLimited")

48. public ResponseEntity<Page<Post>> getPosts(

49. @RequestParam(defaultValue = "0") int page,

50. @RequestParam(defaultValue = "10") int size,

51. @RequestParam(defaultValue = "date") String order,

52. @RequestParam(defaultValue = "false") boolean asc) {

53. Page<Post> posts = null;

54. try {

55.

56. if(asc) {

57. posts = post\_serv.getPaginasPosts(

58. PageRequest.of(page, size, Sort.by(order)));

59. }else {

60. posts = post\_serv.getPaginasPosts(

61. PageRequest.of(page, size, Sort.by(order).descending()));

62. }

63. return new ResponseEntity<>(posts, HttpStatus.OK);

64.

65. } catch (Exception e) {

66. return new ResponseEntity<>(posts, HttpStatus.INTERNAL\_SERVER\_ERROR);

67. }

68. }

69.

70. /\*

71. \* Pre:

72. \* Post: Metodo el cual devuelve los posts filtrados por tags

73. \*/

74. @GetMapping("/postsfiterbytag")

75. public ResponseEntity<Page<Post>> getPostsByTag(

76. @RequestParam(defaultValue = "0") int page,

77. @RequestParam(defaultValue = "10") int size,

78. @RequestParam(defaultValue = "date") String order,

79. @RequestParam(defaultValue = "false") boolean asc,

80. @RequestParam(defaultValue = "") String tag) {

81. Page<Post> posts = null;

82. try {

83. if(asc) {

84. posts = post\_serv.getPaginasToPostsByTag(

85. PageRequest.of(page, size, Sort.by(order)),tag);

86. }else {

87. posts = post\_serv.getPaginasToPostsByTag(

88. PageRequest.of(page, size, Sort.by(order).descending()),tag);

89. }

90. return new ResponseEntity<>(posts, HttpStatus.OK);

91.

92. } catch (Exception e) {

93. return new ResponseEntity<>(posts, HttpStatus.INTERNAL\_SERVER\_ERROR);

94. }

95. }

96. }

97.

Tags Publicas

1. package com.nib.app.controllers.api.provPublic;

2.

3. import java.util.ArrayList;

4. import java.util.List;

5.

6. import org.springframework.beans.factory.annotation.Autowired;

7. import org.springframework.data.domain.Page;

8. import org.springframework.data.domain.PageRequest;

9. import org.springframework.data.domain.Sort;

10. import org.springframework.http.HttpStatus;

11. import org.springframework.http.ResponseEntity;

12. import org.springframework.web.bind.annotation.CrossOrigin;

13. import org.springframework.web.bind.annotation.GetMapping;

14. import org.springframework.web.bind.annotation.RequestMapping;

15. import org.springframework.web.bind.annotation.RequestMethod;

16. import org.springframework.web.bind.annotation.RequestParam;

17. import org.springframework.web.bind.annotation.RestController;

18.

19. import com.nib.app.model.entity.Post;

20. import com.nib.app.model.entity.Tag;

21. import com.nib.app.model.service.TagService;

22.

23. @RestController

24. @RequestMapping("/api/public/tags")

25. @CrossOrigin(origins = {"\*"},methods= {RequestMethod.GET,RequestMethod.POST})

26. public class PublicTags {

27.

28. @Autowired

29. private TagService tagserv;

30.

31. /\*

32. \* Pre:

33. \* Post: Metodo el cual devuelve 10 tags random

34. \*/

35. @GetMapping("/random10")

36. public ResponseEntity<List<Tag>> getUser() {

37. try {

38. List<Tag> a = tagserv.getRandomTags(10);

39. return new ResponseEntity<>(a, HttpStatus.OK);

40. } catch (Exception e) {

41. return new ResponseEntity<>(new ArrayList<Tag>(), HttpStatus.INTERNAL\_SERVER\_ERROR);

42. }

43. }

44.

45. /\*

46. \* Pre:

47. \* Post: Metodo el cual devuelve una lista paginada de tags

48. \*/

49. @GetMapping("/tagsLimited") // Modifyed name of the metodth

50. public ResponseEntity<Page<Tag>> getTags(

51. @RequestParam(defaultValue = "0") int page,

52. @RequestParam(defaultValue = "10") int size,

53. @RequestParam(defaultValue = "name") String order,

54. @RequestParam(defaultValue = "false") boolean asc) {

55. Page<Tag> posts = null;

56. try {

57. System.out.println(asc);

58. if(asc) {

59. posts = tagserv.getPaginasTags(

60. PageRequest.of(page, size, Sort.by(order)));

61. }else {

62. posts = tagserv.getPaginasTags(

63. PageRequest.of(page, size, Sort.by(order).descending()));

64. }

65. posts.getTotalPages();

66. return new ResponseEntity<>(posts, HttpStatus.OK);

67. } catch (Exception e) {

68. return new ResponseEntity<>(posts, HttpStatus.INTERNAL\_SERVER\_ERROR);

69. }

70. }

71. }

72.

Controlador de usuario

1. package com.nib.app.controllers.api.usuario;

2.

3. import org.springframework.beans.factory.annotation.Autowired;

4. import org.springframework.http.HttpStatus;

5. import org.springframework.http.ResponseEntity;

6. import org.springframework.web.bind.annotation.CrossOrigin;

7. import org.springframework.web.bind.annotation.DeleteMapping;

8. import org.springframework.web.bind.annotation.GetMapping;

9. import org.springframework.web.bind.annotation.PathVariable;

10. import org.springframework.web.bind.annotation.PostMapping;

11. import org.springframework.web.bind.annotation.RequestBody;

12. import org.springframework.web.bind.annotation.RequestMapping;

13. import org.springframework.web.bind.annotation.RequestMethod;

14. import org.springframework.web.bind.annotation.RestController;

15.

16. import com.nib.app.model.entity.user.Rol;

17. import com.nib.app.model.entity.user.Usuario;

18. import com.nib.app.model.service.RolService;

19. import com.nib.app.model.service.UsuarioService;

20. import com.nib.app.objects.PO;

21. import com.nib.app.utils.PasswordEncryptor;

22.

23. @RestController

24. @RequestMapping("/api/users")

25. @CrossOrigin(origins = {"\*"},methods= {RequestMethod.GET,RequestMethod.POST})

26. public class UsuarioController {

27.

28. @Autowired

29. private UsuarioService usuarioService;

30.

31. @Autowired

32. private RolService rolService;

33.

34. /\*

35. \* Pre:

36. \* Post: Metodo el cual devuelve la informacion de un usuario

37. \* siempre y cuando sea este mismo usuario

38. \*/

39. @PostMapping("/get")

40. public Usuario getUser(@RequestBody PO token) {

41. Usuario tmp = usuarioService.findUsernameByToken(token.getToken());

42. tmp.setPassword(null);

43. tmp.setToken(null);

44.

45. return tmp;

46. }

47.

48. /\*

49. \* Pre:

50. \* Post: Metodo el cual actualiza los datos de un usuario

51. \* Nota: Se debe mejorar el sistema de almacenamiento de

52. \* contraseña en fututas veriones.

53. \*/

54. @PostMapping("/update")

55. public ResponseEntity<String> updateUser(@RequestBody Usuario usuario) {

56. boolean correct = false;

57. try {

58. String nombre = usuario.getNombre();

59. String apellido = usuario.getApellido();

60. String newPassword = usuario.getPassword();

61. //System.out.println(newPassword);

62. String image = usuario.getImgProfile();

63. String token = usuario.getToken();

64. Usuario lu = usuarioService.findUsernameByToken(token);

65.

66. if(lu != null) {

67. //System.out.println("Ha entrado :3");

68. if(newPassword != null && !newPassword.equalsIgnoreCase(" ")) {

69. PasswordEncryptor encryptor = new PasswordEncryptor();

70. lu.setPassword(encryptor.encrypt(newPassword));

71. //System.out.println("Ha entrado :3");

72. }

73. lu.setImgProfile(image);

74. lu.setApellido(apellido);

75. lu.setNombre(nombre);

76. correct = usuarioService.updateUser(lu);

77. }

78. return new ResponseEntity<>(formatJson(correct), HttpStatus.OK);

79. } catch (Exception e) {

80. return new ResponseEntity<>(formatJson(correct), HttpStatus.INTERNAL\_SERVER\_ERROR);

81. }

82. }

83.

84. public String formatJson(boolean value) {return formatJson(value+"");}

85. public String formatJson(String value) {

86. return "{\"value\":\""+value+"\"}";

87. }

88. /\*

89.

90. // Deprecated

91. @DeleteMapping("/delete\_{userID}")

92. public boolean deleteUser(@PathVariable("userID") String userID) {

93. long id = -1L;

94. try {

95. id = Long.parseLong(userID);

96. } catch(Exception e) {

97.

98. }

99. boolean tmp = false;

100. if(id != -1L) {

101. tmp = usuarioService.deleteUserById(id);

102. }

103. return tmp;

104. }

105. \*/

106.

107. // DEPRECATED

108. /\*

109. @RequestMapping(value = "/userAdd", method = RequestMethod.POST)

110. public Usuario saveNewUser(@RequestBody Usuario usuario){//@RequestBody Usuario usuario) {

111.

112. Usuario tmp = null;

113. //System.out.println("HOALAAA");

114. //System.out.println(usuario);

115. if(usuario != null) {

116.

117. System.out.println(usuario.getNombre());

118. Rol r = rolService.getDefaultRol();

119. usuario.getRoleslist().add(r);

120. tmp = usuarioService.saveUsuario(usuario);

121. }

122. return tmp;

123. }\*/

124.

125. }

126.

Launcher de Nib Backend

1. package com.nib.app;

2.

3. import java.util.ArrayList;

4. import java.util.List;

5. import java.util.Set;

6.

7. import org.springframework.beans.factory.annotation.Autowired;

8. import org.springframework.boot.CommandLineRunner;

9. import org.springframework.boot.SpringApplication;

10. import org.springframework.boot.autoconfigure.SpringBootApplication;

11.

12. import com.nib.app.model.entity.Config;

13. import com.nib.app.model.entity.user.Rol;

14. import com.nib.app.model.entity.user.Usuario;

15. import com.nib.app.model.service.ConfService;

16. import com.nib.app.model.service.RolService;

17. import com.nib.app.model.service.UsuarioService;

18. import com.nib.app.utils.NIBShell;

19.

20. import jakarta.transaction.Transactional;

21.

22. /\*

23. \* NIB BACKEND APPLICATION

24. \* --------------------------------

25. \* Created by Clara Bujeda Muñoz

26. \* Date: 2023

27. \* email: clarabujedamunoz@gmail.com

28. \*/

29. @SpringBootApplication

30. public class BackendNibApplication implements CommandLineRunner {

31.

32. @Autowired

33. private UsuarioService usuarioService;

34.

35. @Autowired

36. private RolService rolService;

37.

38. @Autowired

39. private ConfService confserv;

40.

41.

42. public static void main(String[] args) {

43. SpringApplication.run(BackendNibApplication.class, args);

44. }

45.

46. /\*

47. \* Pre:

48. \* Post: Metodo de preconfiguración

49. \*/

50. @Transactional

51. @Override

52. public void run(String... args) throws Exception {

53. printInfo("Verifying configuration...");

54. if(confserv.getCont() == 0) {

55. printInfo("Updating configuration");

56. confserv.addConf(new Config("app\_name","Nine Image Board"));

57. confserv.addConf(new Config("app\_short\_name","NIB"));

58. }

59. String username = "ADMIN";

60. Usuario usuario = new Usuario();

61. usuario.setUsername("ADMIN");

62. printInfo("Verifying Users");

63. if(!usuarioService.isExistsUsernameByUsername(usuario)) {

64. printInfo("Creating user ADMIN...");

65. Rol r = new Rol();

66. r.setId\_rol(1L);

67. r.setNombre("ADMIN");

68. addRol(r);

69. Rol r2 = new Rol();

70. r2.setId\_rol(2L);

71. r2.setNombre("USER");

72. addRol(r2);

73. usuario.setNombre("Admin");

74. usuario.setApellido("Admin");

75. usuario.setPassword("ADMIN");

76. usuario.setEmail("admin@admin.com");

77. usuario.setImgProfile("image.png");

78. usuario.getRoleslist().add(rolService.findByName(r));

79. usuarioService.saveUsuario(usuario);

80. }

81. }

82.

83. /\*

84. \* Pre:

85. \* Post: Metodo con el cual almacenamos un nuevo rol en la BBDD

86. \*/

87. private void addRol(Rol rol) {

88. if(!rolService.isExistsRolByName(rol)) {

89. rolService.saveRol(rol);

90. }

91. }

92.

93. /\*

94. \* Pre:

95. \* Post: Metodo con el cual imprimimos por consola.

96. \*/

97. public void printInfo(String str) {

98. new NIBShell().printInfo(str);

99. /\*

100. String YELLOW\_BOLD = "\033[1;33m"; // YELLOW

101. String PURPLE\_BOLD = "\033[1;35m"; // PURPLE

102. String RESET = "\033[0m"; // RESET

103. String date = java.time.Clock.systemUTC().instant().toString();

104. System.out.println(date.substring(0, date.length()-1)+" "+YELLOW\_BOLD+"[NIB] "+PURPLE\_BOLD+" --- "+RESET + str);

105. \*/

106. }

107.

108. }

109.

**Configurator**

Main

1. package app;

2.

3. import java.io.File;

4. import java.io.FileInputStream;

5. import java.io.FileNotFoundException;

6. import java.io.IOException;

7. import java.io.InputStream;

8. import java.util.Formatter;

9. import java.util.Properties;

10. import java.util.Scanner;

11.

12. public class Main {

13.

14. private static String \_fCyan="[36m";

15. private static String \_fBMag="[95m";

16. public static void main(String[] args) {

17.

18. Properties prop = new Properties();

19.

20. File f = new File("");

21. String path = f.getAbsolutePath();

22. String[] fold = path.split("\\\\");

23. String path\_raiz = "";

24. for (int i = 0; i < fold.length - 2; i++) {

25. path\_raiz = path\_raiz + "\\" + fold[i];

26. }

27. System.out.println("Generando proyecto.. \n In >"+path\_raiz);

28. File backprops = new File(path\_raiz + "\\PROYECT NIB\\Compiled\\back.properties");

29. File frontStart = new File(path\_raiz + "\\Start Front.bat");

30. File backStart = new File(path\_raiz + "\\Start Backend.bat");

31. File props = new File(path\_raiz + "\\config.properties");

32. if (props.exists()) {

33. try {

34. InputStream is = new FileInputStream(props);

35. try {

36. prop.load(is);

37. String NIB\_B\_Port = prop.getProperty("NIB\_B\_Port");

38. String NIB\_F\_Port = prop.getProperty("NIB\_F\_Port");

39. String NIB\_B\_ip = prop.getProperty("NIB\_B\_ip");

40. String MySql\_ip = prop.getProperty("MySql\_ip");

41. String MySql\_port = prop.getProperty("MySql\_Port");

42. String MySql\_bbdd = prop.getProperty("MySql\_bbdd");

43. String MySql\_user = prop.getProperty("MySql\_user");

44. String MySql\_pass = prop.getProperty("MySql\_passs");

45. String NIB\_B\_Version = prop.getProperty("NIB\_B\_Version");

46. String NIB\_DEV\_OPTION = prop.getProperty("DEV");

47.

48. String ngPath = prop.getProperty("ngPath");

49. String javaPath = prop.getProperty("javaPath");

50. // Default Values

51. if(ngPath == null) {ngPath = "ng";}

52. if(javaPath == null) {javaPath = "java";}

53. if(NIB\_B\_Port == null) {NIB\_B\_Port = "3000";}

54. if(NIB\_F\_Port == null) {NIB\_F\_Port = "8080";}

55. if(NIB\_B\_ip == null) {NIB\_B\_ip = "localhost";}

56. if(MySql\_ip == null) {MySql\_ip = "localhost";}

57. if(MySql\_port == null) {MySql\_port = "3306";}

58. if(MySql\_bbdd == null) {MySql\_bbdd = "ninbooru";}

59. if(MySql\_user == null) {MySql\_user = "root";}

60. if(MySql\_pass == null) {MySql\_pass = "root";}

61. if(NIB\_B\_Version == null) {NIB\_B\_Version = "0.1.1-SNAPSHOT";}

62. boolean dev = false;

63. if(NIB\_DEV\_OPTION == null) {

64. dev = false;

65. }else if(NIB\_DEV\_OPTION.equalsIgnoreCase("true")) {

66. dev = true;

67. }

68.

69.

70. System.out.println("[GEN] back.properties");

71. String genPropBack = ""

72. + "server.port="+NIB\_B\_Port+"\n"

73. + "spring.datasource.url=jdbc:mysql://"+MySql\_ip+":"+MySql\_port+"/"+MySql\_bbdd+"\n"

74. + "spring.datasource.username="+MySql\_user+"\n"

75. + "spring.datasource.password="+MySql\_pass+"\n"

76. + "spring.jpa.properties.hibenate.dialect=org.hibenate.dialect.MySQL8Dialect"+"\n"

77. + "spring.jpa.generate-ddl=true"+"\n"

78. + "spring.jpa.hibernate.ddl-auto=update"+"\n";

79.

80. if(dev == true) {

81. genPropBack = genPropBack + "\n"

82. + "spring.jpa.show-sql=true "+"\n"

83. + "spring.jpa.properties.hibenate.format\_sql=true "+"\n";

84. }

85. System.out.println("Configuración aplicada back: " + genPropBack);

86. // Back props

87. Formatter fo = new Formatter(backprops);

88. fo.format(genPropBack);

89. fo.flush();

90. fo.close();

91. System.out.println("[GEN] Start Front.bat");

92. String genStartFront =

93. "cd ./PROYECT NIB \n"

94. + "cd ./FrontEnd\_NIB \n"

95. + "title NINE IMAGE BOARD - FrontEnd \n"

96. + "chcp 1252 \n"

97. + "@ECHO OFF \n"

98. + "cls \n"

99. + "echo. \n"

100. + "echo. \n"

101. + "echo ,--. \n"

102. + "echo ,--.'^| ,---, ,---,. \r\n"

103. + "echo ,--,: : ^|,`--.' ^| ,' .' \\\r\n"

104. + "echo ,`--.'`^| ' :^| : :,---.' .' ^|\r\n"

105. + "echo ^| : : ^| ^|: ^| '^| ^| ^|: ^|\r\n"

106. + "echo : ^| \\ ^| :^| : ^|: : : /\r\n"

107. + "echo ^| : ' '; ^|' ' ;: ^| ; \r\n"

108. + "echo ' ' ;. ;^| ^| ^|^| : \\\r\n"

109. + "echo ^| ^| ^| \\ ^|' : ;^| ^| . ^|\r\n"

110. + "echo ' : ^| ; .'^| ^| '' : '; ^|\r\n"

111. + "echo ^| ^| '`--' ' : ^|^| ^| ^| ; \r\n"

112. + "echo ' : ^| ; ^|.' ^| : / \r\n"

113. + "echo ; ^|.' '---' ^| ^| ,' (v1.0.1)\r\n"

114. + "echo '---' `----' \r\n"

115. + "echo Power by Angular\r\n"

116. + "echo.\r\n"

117. + ""+ngPath+" serve --port "+NIB\_F\_Port+"\r\n"

118. + "pause > nul\r\n"

119. + "exit";

120. Formatter foStart = new Formatter(frontStart);

121. foStart.format(genStartFront);

122. foStart.flush();

123. foStart.close();

124. System.out.println("[GEN] Start Backend.bat");

125. String genStartBack =

126. "cd ./PROYECT NIB \r\n"

127. + "cd ./Compiled \r\n"

128. + "title NINE IMAGE BOARD - BackendEnd \r\n"

129. + "\r\n"

130. + "@ECHO OFF\r\n"

131. + "cls\r\n"

132. + "echo."+\_fCyan+"\r\n"

133. + "echo. \r\n"

134. + "echo ,--. \r\n"

135. + "echo ,--.'^| ,---, ,---,. \r\n"

136. + "echo ,--,: : ^|,`--.' ^| ,' .' \\\r\n"

137. + "echo ,`--.'`^| ' :^| : :,---.' .' ^|\r\n"

138. + "echo ^| : : ^| ^|: ^| '^| ^| ^|: ^|\r\n"

139. + "echo : ^| \\ ^| :^| : ^|: : : /\r\n"

140. + "echo ^| : ' '; ^|' ' ;: ^| ; \r\n"

141. + "echo ' ' ;. ;^| ^| ^|^| : \\\r\n"

142. + "echo ^| ^| ^| \\ ^|' : ;^| ^| . ^|\r\n"

143. + "echo ' : ^| ; .'^| ^| '' : '; ^|\r\n"

144. + "echo ^| ^| '`--' ' : ^|^| ^| ^| ; \r\n"

145. + "echo ' : ^| ; ^|.' ^| : / \r\n"

146. + "echo ; ^|.' '---' ^| ^| ,' (v1.0.1)\r\n"

147. + "echo '---' `----' \r\n"

148. + "echo Power by SpringBoot\r\n"+ "echo "+\_fBMag+" \n"

149. + "\r\n"

150. + "\""+javaPath+"\" -jar Backend\_NIB-"+NIB\_B\_Version + ".jar --spring.config.location=back.properties"+"\n"

151. + "pause > nul \n"

152. + "exit";

153.

154. Formatter fbStart = new Formatter(backStart);

155. fbStart.format(genStartBack);

156. fbStart.flush();

157. fbStart.close();

158.

159.

160. } catch (IOException e) {

161. e.printStackTrace();

162. }

163. } catch (FileNotFoundException e) {

164. e.printStackTrace();

165. }

166. } else {

167. System.out.println("No existe archivo de propiedades");

168. }

169. }

170. }

**Launcher Shell**

Auto Start

1. title starter

2. Set \_fGreen=[32m

3. Set \_fYellow=[33m

4. Set \_fBRed=[91m

5. @echo off

6. cls

7. echo %\_fGreen%

8. echo.

9. echo ,--.

10. echo ,--.'^| ,---, ,---,.

11. echo ,--,: : ^|,`--.' ^| ,' .' \

12. echo ,`--.'`^| ' :^| : :,---.' .' ^|

13. echo ^| : : ^| ^|: ^| '^| ^| ^|: ^|

14. echo : ^| \ ^| :^| : ^|: : : /

15. echo ^| : ' '; ^|' ' ;: ^| ;

16. echo ' ' ;. ;^| ^| ^|^| : \

17. echo ^| ^| ^| \ ^|' : ;^| ^| . ^|

18. echo ' : ^| ; .'^| ^| '' : '; ^|

19. echo ^| ^| '`--' ' : ^|^| ^| ^| ;

20. echo ' : ^| ; ^|.' ^| : /

21. echo ; ^|.' '---' ^| ^| ,' %\_fYellow% (v1.0.1) %\_fGreen%

22. echo '---' `----'

23. echo %\_fBRed%APP INIT

24. echo %\_fYellow%

25. echo Generando Resources....

26. cd ./PROYECT NIB/Compiled

27. java -jar NIB\_CONFIGURATOR.jar

28. cd ../..

29. echo Iniciando servidores....

30. start "" "./Start Backend.bat"

31. start "" "./Start Front.bat"

32. pause

33. exit

34.

Start Backend

1. cd ./PROYECT NIB

2. cd ./Compiled

3. title NINE IMAGE BOARD - BackendEnd

4.

5. @ECHO OFF

6. cls

7. echo.[36m

8. echo.

9. echo ,--.

10. echo ,--.'^| ,---, ,---,.

11. echo ,--,: : ^|,`--.' ^| ,' .' \

12. echo ,`--.'`^| ' :^| : :,---.' .' ^|

13. echo ^| : : ^| ^|: ^| '^| ^| ^|: ^|

14. echo : ^| \ ^| :^| : ^|: : : /

15. echo ^| : ' '; ^|' ' ;: ^| ;

16. echo ' ' ;. ;^| ^| ^|^| : \

17. echo ^| ^| ^| \ ^|' : ;^| ^| . ^|

18. echo ' : ^| ; .'^| ^| '' : '; ^|

19. echo ^| ^| '`--' ' : ^|^| ^| ^| ;

20. echo ' : ^| ; ^|.' ^| : /

21. echo ; ^|.' '---' ^| ^| ,' (v1.0.1)

22. echo '---' `----'

23. echo Power by SpringBoot

24. echo [95m

25.

26. "C:\Program Files\Java\jdk-17\bin\java.exe" -jar Backend\_NIB-0.1.1-SNAPSHOT.jar --spring.config.location=back.properties

27. pause > nul

28. exit

29.

Start Front

1. cd ./PROYECT NIB

2. cd ./FrontEnd\_NIB

3. title NINE IMAGE BOARD - FrontEnd

4. chcp 1252

5. @ECHO OFF

6. cls

7. echo.

8. echo.

9. echo ,--.

10. echo ,--.'^| ,---, ,---,.

11. echo ,--,: : ^|,`--.' ^| ,' .' \

12. echo ,`--.'`^| ' :^| : :,---.' .' ^|

13. echo ^| : : ^| ^|: ^| '^| ^| ^|: ^|

14. echo : ^| \ ^| :^| : ^|: : : /

15. echo ^| : ' '; ^|' ' ;: ^| ;

16. echo ' ' ;. ;^| ^| ^|^| : \

17. echo ^| ^| ^| \ ^|' : ;^| ^| . ^|

18. echo ' : ^| ; .'^| ^| '' : '; ^|

19. echo ^| ^| '`--' ' : ^|^| ^| ^| ;

20. echo ' : ^| ; ^|.' ^| : /

21. echo ; ^|.' '---' ^| ^| ,' (v1.0.1)

22. echo '---' `----'

23. echo Power by Angular

24. echo.

25. ng serve --port 8080

26. pause > nul

27. exit

28.

Configurator

1. title starter

2. Set \_fGreen=[32m

3. Set \_fYellow=[33m

4. Set \_fBRed=[91m

5. @echo off

6. cls

7. echo %\_fGreen%

8. echo.

9. echo ,--.

10. echo ,--.'^| ,---, ,---,.

11. echo ,--,: : ^|,`--.' ^| ,' .' \

12. echo ,`--.'`^| ' :^| : :,---.' .' ^|

13. echo ^| : : ^| ^|: ^| '^| ^| ^|: ^|

14. echo : ^| \ ^| :^| : ^|: : : /

15. echo ^| : ' '; ^|' ' ;: ^| ;

16. echo ' ' ;. ;^| ^| ^|^| : \

17. echo ^| ^| ^| \ ^|' : ;^| ^| . ^|

18. echo ' : ^| ; .'^| ^| '' : '; ^|

19. echo ^| ^| '`--' ' : ^|^| ^| ^| ;

20. echo ' : ^| ; ^|.' ^| : /

21. echo ; ^|.' '---' ^| ^| ,' %\_fYellow% (v1.0.1) %\_fGreen%

22. echo '---' `----'

23. echo %\_fBRed%APP INIT

24. echo %\_fYellow%

25. echo Generando Resources....

26. cd ./PROYECT NIB/Compiled

27. java -jar NIB\_CONFIGURATOR.jar

28.

29. exit

30.

**Lanzador Grafico**

Código

1. using System;

2. using System.Collections.Generic;

3. using System.ComponentModel;

4. using System.Data;

5. using System.Diagnostics;

6. using System.Drawing;

7. using System.Linq;

8. using System.Text;

9. using System.Threading;

10. using System.Threading.Tasks;

11. using System.Windows.Forms;

12. using System.Net;

13.

14. namespace APPSTART

15. {

16. public partial class Form1 : Form

17. {

18. Thread t0;

19.

20. //private BackgroundWorker backgroundWorker;

21.

22. public Form1()

23. {

24. InitializeComponent();

25. Config\_gen.Enabled = false;

26. StartBack.Enabled = false;

27. front\_server.Enabled = false;

28. front\_server.Text = "STOP";

29. StartBack.Text = "STOP";

30. StartBack.BackColor = Color.Red;

31. front\_server.BackColor = Color.Red;

32. Config\_gen.BackColor = Color.Orange;

33. start\_all.Text = "START";

34. addShell("Aplicación Iniciada");

35. }

36.

37. //-----------------------START APP

38.

39. private void executebat(String path) {

40. string batFilePath = @"" + path + "";

41. Process process = new Process();

42. process.StartInfo.FileName = "cmd.exe";

43. process.StartInfo.Arguments = "/c " + batFilePath;

44. process.StartInfo.CreateNoWindow = true;

45. process.StartInfo.UseShellExecute = false;

46. process.Start();

47. process.WaitForExit();

48. int exitCode = process.ExitCode;

49. Console.WriteLine("Exit Code: " + exitCode);

50. addShell(" [ Exit Code ] " + exitCode);

51. }

52.

53. private void start\_all\_Click(object sender, EventArgs e)

54. {

55. addShell("Scripts Iniciados");

56. this.t0 = new Thread(() => this.executebat("\"./Start Auto.bat\""));

57. this.t0.Start();

58. }

59. private void StartBack\_Click(object sender, EventArgs e)

60. {

61.

62.

63. }

64.

65.

66.

67. private void front\_server\_Click(object sender, EventArgs e)

68. {

69. }

70.

71. private void Config\_gen\_Click(object sender, EventArgs e)

72. {

73.

74. }

75.

76. // ------------------------ Integrated shell

77. private void addShell(String txt) { addShell("INFO", txt); }

78. private void addShell(String type, String txt) {

79. shell.Text = "『" + getTime() + "』" + " [" + type + "] ► " + txt + " \n" + shell.Text;

80. }

81. public String getTime() {

82. return DateTime.Now.ToString("HH:mm:ss tt");

83. }

84.

85.

86. private void panel2\_Paint(object sender, PaintEventArgs e)

87. {

88.

89. }

90.

91. private void goConfig\_Click(object sender, EventArgs e)

92. {

93. CopnfigProperties c = new CopnfigProperties();

94. c.ShowDialog();

95. }

96.

97. //-----------------------INSTALL SECCTION -----------------------

98.

99. private void auto\_install\_Click(object sender, EventArgs e)

100. {

101. if (!testNode()) {

102. addShell("", "Descargando version de node");

103.

104. }

105. }

106. private void test\_node\_Click(object sender, EventArgs e)

107. {

108. testNode();

109. }

110.

111. private String nodeVersion = "v18.12.0";

112. private String node64 = "https://nodejs.org/dist/v18.12.0/node-v18.12.0-x64.msi";

113. private String node86 = "https://nodejs.org/dist/v18.12.0/node-v18.12.0-x86.msi";

114. private void download(String urlDownlad, String name) {

115.

116. {

117. //string url = "https://www.example.com/archivo.zip"; // URL del archivo a descargar

118. string destino = @".down/"; // Ruta de destino para guardar el archivo descargado

119.

120. WebClient webClient = new WebClient();

121.

122. try

123. {

124. webClient.DownloadFile(urlDownlad, destino);

125. Console.WriteLine("Descarga completada.");

126. }

127. catch (Exception ex)

128. {

129. Console.WriteLine("Error al descargar el archivo: " + ex.Message);

130. }

131. }

132. }

133.

134. private Boolean test64\_86(){

135.

136. return true;

137. }

138.

139. private Boolean testNode() {

140. // Ejecutar el comando "node --version" en el símbolo del sistema

141. string output = RunCommand("node --version");

142. if (!string.IsNullOrEmpty(output))

143. {

144. addShell("", "Node.js está instalado.");

145. addShell("", "Versión: " + output);

146. if (output.Equals(this.nodeVersion)) {

147. return true;

148. }

149. else

150. {

151. return false;

152. }

153.

154. }

155. else

156. {

157. addShell("", "Node.js no está instalado.");

158. return false;

159. }

160. }

161.

162.

163. static string RunCommand(string command)

164. {

165. Process process = new Process();

166. process.StartInfo.FileName = "cmd.exe";

167. process.StartInfo.Arguments = "/c " + command;

168. process.StartInfo.RedirectStandardOutput = true;

169. process.StartInfo.UseShellExecute = false;

170. process.StartInfo.CreateNoWindow = true;

171.

172. process.Start();

173. string output = process.StandardOutput.ReadToEnd().Trim();

174. process.WaitForExit();

175.

176. return output;

177. }

178.

179.

180. }

181. }

Formulario

1. namespace APPSTART

2. {

3. partial class Form1

4. {

5. /// <summary>

6. /// Required designer variable.

7. /// </summary>

8. private System.ComponentModel.IContainer components = null;

9.

10. /// <summary>

11. /// Clean up any resources being used.

12. /// </summary>

13. /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

14. protected override void Dispose(bool disposing)

15. {

16. if (disposing && (components != null))

17. {

18. components.Dispose();

19. }

20. base.Dispose(disposing);

21. }

22.

23. #region Windows Form Designer generated code

24.

25. /// <summary>

26. /// Required method for Designer support - do not modify

27. /// the contents of this method with the code editor.

28. /// </summary>

29. private void InitializeComponent()

30. {

31. System.ComponentModel.ComponentResourceManager resources = new System.ComponentModel.ComponentResourceManager(typeof(Form1));

32. this.label1 = new System.Windows.Forms.Label();

33. this.label2 = new System.Windows.Forms.Label();

34. this.label3 = new System.Windows.Forms.Label();

35. this.StartBack = new System.Windows.Forms.Button();

36. this.front\_server = new System.Windows.Forms.Button();

37. this.Config\_gen = new System.Windows.Forms.Button();

38. this.start\_all = new System.Windows.Forms.Button();

39. this.label4 = new System.Windows.Forms.Label();

40. this.panel1 = new System.Windows.Forms.Panel();

41. this.label5 = new System.Windows.Forms.Label();

42. this.panel2 = new System.Windows.Forms.Panel();

43. this.shell = new System.Windows.Forms.Label();

44. this.goConfig = new System.Windows.Forms.Button();

45. this.auto\_install = new System.Windows.Forms.Button();

46. this.label6 = new System.Windows.Forms.Label();

47. this.panel3 = new System.Windows.Forms.Panel();

48. this.test\_node = new System.Windows.Forms.Button();

49. this.panel4 = new System.Windows.Forms.Panel();

50. this.panel1.SuspendLayout();

51. this.panel2.SuspendLayout();

52. this.panel3.SuspendLayout();

53. this.panel4.SuspendLayout();

54. this.SuspendLayout();

55. //

56. // label1

57. //

58. resources.ApplyResources(this.label1, "label1");

59. this.label1.Name = "label1";

60. //

61. // label2

62. //

63. resources.ApplyResources(this.label2, "label2");

64. this.label2.Name = "label2";

65. //

66. // label3

67. //

68. resources.ApplyResources(this.label3, "label3");

69. this.label3.Name = "label3";

70. //

71. // StartBack

72. //

73. resources.ApplyResources(this.StartBack, "StartBack");

74. this.StartBack.Name = "StartBack";

75. this.StartBack.UseVisualStyleBackColor = true;

76. this.StartBack.Click += new System.EventHandler(this.StartBack\_Click);

77. //

78. // front\_server

79. //

80. resources.ApplyResources(this.front\_server, "front\_server");

81. this.front\_server.Name = "front\_server";

82. this.front\_server.UseVisualStyleBackColor = true;

83. this.front\_server.Click += new System.EventHandler(this.front\_server\_Click);

84. //

85. // Config\_gen

86. //

87. resources.ApplyResources(this.Config\_gen, "Config\_gen");

88. this.Config\_gen.Name = "Config\_gen";

89. this.Config\_gen.UseVisualStyleBackColor = true;

90. this.Config\_gen.Click += new System.EventHandler(this.Config\_gen\_Click);

91. //

92. // start\_all

93. //

94. resources.ApplyResources(this.start\_all, "start\_all");

95. this.start\_all.Name = "start\_all";

96. this.start\_all.UseVisualStyleBackColor = true;

97. this.start\_all.Click += new System.EventHandler(this.start\_all\_Click);

98. //

99. // label4

100. //

101. resources.ApplyResources(this.label4, "label4");

102. this.label4.Name = "label4";

103. //

104. // panel1

105. //

106. resources.ApplyResources(this.panel1, "panel1");

107. this.panel1.BackColor = System.Drawing.Color.Transparent;

108. this.panel1.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;

109. this.panel1.Controls.Add(this.label5);

110. this.panel1.Name = "panel1";

111. //

112. // label5

113. //

114. resources.ApplyResources(this.label5, "label5");

115. this.label5.Name = "label5";

116. //

117. // panel2

118. //

119. resources.ApplyResources(this.panel2, "panel2");

120. this.panel2.BackColor = System.Drawing.Color.Black;

121. this.panel2.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;

122. this.panel2.Controls.Add(this.shell);

123. this.panel2.Name = "panel2";

124. this.panel2.Paint += new System.Windows.Forms.PaintEventHandler(this.panel2\_Paint);

125. //

126. // shell

127. //

128. resources.ApplyResources(this.shell, "shell");

129. this.shell.ForeColor = System.Drawing.Color.Yellow;

130. this.shell.Name = "shell";

131. //

132. // goConfig

133. //

134. resources.ApplyResources(this.goConfig, "goConfig");

135. this.goConfig.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(255)))), ((int)(((byte)(192)))), ((int)(((byte)(128)))));

136. this.goConfig.Name = "goConfig";

137. this.goConfig.UseVisualStyleBackColor = false;

138. this.goConfig.Click += new System.EventHandler(this.goConfig\_Click);

139. //

140. // auto\_install

141. //

142. resources.ApplyResources(this.auto\_install, "auto\_install");

143. this.auto\_install.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(192)))), ((int)(((byte)(255)))), ((int)(((byte)(255)))));

144. this.auto\_install.Name = "auto\_install";

145. this.auto\_install.UseVisualStyleBackColor = false;

146. this.auto\_install.Click += new System.EventHandler(this.auto\_install\_Click);

147. //

148. // label6

149. //

150. resources.ApplyResources(this.label6, "label6");

151. this.label6.Name = "label6";

152. //

153. // panel3

154. //

155. resources.ApplyResources(this.panel3, "panel3");

156. this.panel3.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;

157. this.panel3.Controls.Add(this.test\_node);

158. this.panel3.Controls.Add(this.label6);

159. this.panel3.Controls.Add(this.auto\_install);

160. this.panel3.Name = "panel3";

161. //

162. // test\_node

163. //

164. resources.ApplyResources(this.test\_node, "test\_node");

165. this.test\_node.BackColor = System.Drawing.Color.LemonChiffon;

166. this.test\_node.Name = "test\_node";

167. this.test\_node.UseVisualStyleBackColor = false;

168. this.test\_node.Click += new System.EventHandler(this.test\_node\_Click);

169. //

170. // panel4

171. //

172. resources.ApplyResources(this.panel4, "panel4");

173. this.panel4.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;

174. this.panel4.Controls.Add(this.label4);

175. this.panel4.Controls.Add(this.start\_all);

176. this.panel4.Name = "panel4";

177. //

178. // Form1

179. //

180. resources.ApplyResources(this, "$this");

181. this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

182. this.Controls.Add(this.goConfig);

183. this.Controls.Add(this.panel2);

184. this.Controls.Add(this.Config\_gen);

185. this.Controls.Add(this.front\_server);

186. this.Controls.Add(this.StartBack);

187. this.Controls.Add(this.label3);

188. this.Controls.Add(this.label2);

189. this.Controls.Add(this.label1);

190. this.Controls.Add(this.panel1);

191. this.Controls.Add(this.panel4);

192. this.Controls.Add(this.panel3);

193. this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;

194. this.MaximizeBox = false;

195. this.Name = "Form1";

196. this.panel1.ResumeLayout(false);

197. this.panel1.PerformLayout();

198. this.panel2.ResumeLayout(false);

199. this.panel2.PerformLayout();

200. this.panel3.ResumeLayout(false);

201. this.panel3.PerformLayout();

202. this.panel4.ResumeLayout(false);

203. this.panel4.PerformLayout();

204. this.ResumeLayout(false);

205. this.PerformLayout();

206.

207. }

208.

209. #endregion

210.

211. private System.Windows.Forms.Label label1;

212. private System.Windows.Forms.Label label2;

213. private System.Windows.Forms.Label label3;

214. private System.Windows.Forms.Button StartBack;

215. private System.Windows.Forms.Button front\_server;

216. private System.Windows.Forms.Button Config\_gen;

217. private System.Windows.Forms.Button start\_all;

218. private System.Windows.Forms.Label label4;

219. private System.Windows.Forms.Panel panel1;

220. private System.Windows.Forms.Label label5;

221. private System.Windows.Forms.Panel panel2;

222. private System.Windows.Forms.Label shell;

223. private System.Windows.Forms.Button goConfig;

224. private System.Windows.Forms.Button auto\_install;

225. private System.Windows.Forms.Label label6;

226. private System.Windows.Forms.Panel panel3;

227. private System.Windows.Forms.Panel panel4;

228. private System.Windows.Forms.Button test\_node;

229. }

230. }

231.

**FrontEnd**

Modulo app

1. import { NgModule } from '@angular/core';

2. import { BrowserModule } from '@angular/platform-browser';

3.

4. import { AppRoutingModule } from './app-routing.module';

5. import { AppComponent } from './app.component';

6.

7. import { TranslateLoader, TranslateModule } from '@ngx-translate/core';

8. import { TranslateHttpLoader } from '@ngx-translate/http-loader';

9. import {HttpClient, HttpClientModule} from '@angular/common/http';

10.

11. import { NavbarComponent } from './components/navbar/navbar.component';

12. import { SignupComponent } from './pages/signup/signup.component';

13. import { LoginComponent } from './pages/login/login.component';

14. import { RouterModule, Routes } from '@angular/router';

15. import { PublicHomeComponent } from './pages/public-home/public-home.component';

16. import { TermsAndConditionsComponent } from './pages/terms-and-conditions/terms-and-conditions.component';

17. import { TagsComponent } from './pages/tags/tags.component';

18. import { formatCurrency } from '@angular/common';

19. import { SwitchLangComponent } from './components/switch-lang/switch-lang.component';

20. import { PostsComponent } from './pages/posts/posts.component';

21. import { NabarPostsComponent } from './components/nabar-posts/nabar-posts.component';

22. import { PostcontainerComponent } from './pages/postcontainer/postcontainer.component';

23. import { NoFoundComponent } from './pages/system/no-found/no-found.component';

24. import { CreatebyComponent } from './pages/system/createby/createby.component';

25. import { NavbarStaticPageComponent } from './components/navbar-static-page/navbar-static-page.component';

26. import { FormsModule } from '@angular/forms';

27. import { UserProfileComponent } from './private/pages/user-profile/user-profile.component';

28. import { UnautorizedComponent } from './pages/unautorized/unautorized.component';

29. import { PrivateHomeComponent } from './private/pages/private-home/private-home.component';

30. import { PrivateNavComponent } from './private/components/private-nav/private-nav.component';

31. import { PrivatePostsComponent } from './private/pages/private-posts/private-posts.component';

32. import { UploadComponent } from './private/pages/upload/upload.component';

33. import { ContainTagsComponent } from './pages/contain-tags/contain-tags.component';

34. import { PrivateTagsComponent } from './private/pages/private-tags/private-tags.component';

35. import { InfoPostsComponent } from './pages/info-posts/info-posts.component';

36. import { AdminPanelComponent } from './private/pages/administration/admin-panel/admin-panel.component';

37.

38. const priv = "private/"

39.

40. const appRoutes:Routes = [

41.   {path:"",component:PublicHomeComponent},

42.   {path:"home",component:PublicHomeComponent},

43.   {path:"posts",component:PostcontainerComponent},

44.   {path:"posts/:id",component:PostcontainerComponent},

45.

46.   {path:"login",component:LoginComponent},

47.   {path:"signup",component:SignupComponent},

48.   {path:"terms-and-conditions",component:TermsAndConditionsComponent},

49.   {path:"tags",component:ContainTagsComponent},

50.

51.   // public and private components

52.   { path: 'info/post/:id', component: InfoPostsComponent },

53.

54.  //private

55.  {path:priv+"userprofile",component:UserProfileComponent},

56.  {path:priv+"home",component:PrivateHomeComponent},

57.  {path:priv+"posts",component:PrivatePostsComponent },

58.  {path:priv+"posts/:id",component:PrivatePostsComponent },

59.  {path:priv+"upload",component:UploadComponent },

60.  {path:priv+"tags",component:PrivateTagsComponent },

61.

62. //SPECIAL

63.  {path:"creadoradenibsuperkawaiipage",component:CreatebyComponent},

64.  {path:"unautorized",component:UnautorizedComponent},

65.  {path:"\*\*",pathMatch:"full",component:NoFoundComponent},

66.

67. ]

68.

69. @NgModule({

70.   declarations: [

71.     AppComponent,

72.     NavbarComponent,

73.     SignupComponent,

74.     LoginComponent,

75.     PublicHomeComponent,

76.     TermsAndConditionsComponent,

77.     TagsComponent,

78.     SwitchLangComponent,

79.     PostsComponent,

80.     NabarPostsComponent,

81.     PostcontainerComponent,

82.     NoFoundComponent,

83.     CreatebyComponent,

84.     NavbarStaticPageComponent,

85.     UserProfileComponent,

86.     UnautorizedComponent,

87.     PrivateHomeComponent,

88.     PrivateNavComponent,

89.     PrivatePostsComponent,

90.     UploadComponent,

91.     ContainTagsComponent,

92.     PrivateTagsComponent,

93.     InfoPostsComponent,

94.     AdminPanelComponent,

95.   ],

96.   imports: [

97.     BrowserModule,

98.     AppRoutingModule,

99.     HttpClientModule,

100.     FormsModule,

101.     TranslateModule.forRoot({

102.       loader: {

103.         provide: TranslateLoader,

104.         useFactory: httpTranslateLoader,

105.         deps: [HttpClient]

106.       }

107.     }),

108.     RouterModule.forRoot(appRoutes)

109.   ],

110.   providers: [],

111.   bootstrap: [AppComponent]

112. })

113. export class AppModule { }

114.

115. export function httpTranslateLoader(http: HttpClient){

116. return new TranslateHttpLoader(http);

117. }

118.

119.

Componente app y Sistema de multi idioma

1. import { Component } from '@angular/core';

2.

3. import { TranslateService } from '@ngx-translate/core';

4. @Component({

5.   selector: 'app-root',

6.   templateUrl: './app.component.html',

7.   styleUrls: ['./app.component.css']

8. })

9. export class AppComponent {

10.   title = 'FrontEnd\_NIB';

11.   constructor(

12.     public translate: TranslateService

13.   ) {

14.     const lang = translate.getBrowserLang();

15.     translate.addLangs(["es","en","ja"]);

16.     if((lang !== 'es') && (lang !== 'en')){

17.       translate.setDefaultLang('en');

18.     }else if(lang === 'es'){

19.       translate.setDefaultLang('es');

20.     }else if(lang === 'en'){

21.       translate.setDefaultLang('en');

22.     }else if(lang === 'ja'){

23.       translate.setDefaultLang('ja');

24.     }

25.

26.   }

27.

28. }

29.

30.

Servicio de acceso a las tags

1. import { HttpClient } from '@angular/common/http';

2. import { Injectable } from '@angular/core';

3. import { backserv } from 'config';

4. import { Observable } from 'rxjs';

5.

6. @Injectable({

7.   providedIn: 'root'

8. })

9. export class TagsService {

10.

11.   constructor(private httpClient: HttpClient) { } //: Observable<Any>

12.

13.   public tagsPages(page:number,size:number,order:string, asc: boolean){

14.       let apiPostsPageURL = "http://"+backserv.ipnibbackserver+":"+backserv.portnibbackserver

15.                     +"/api/public/tags/tagsLimited?";

16.       return this.httpClient.get<any>(apiPostsPageURL + `size=${size}&page=${page}&order=${order}&asc=${asc}`)

17.   }

18.

19.   public tagsRandom(){

20.     let apiPostsPageURL = "http://"+backserv.ipnibbackserver+":"+backserv.portnibbackserver

21.                   +"/api/public/tags/random10";

22.     return this.httpClient.get<any>(apiPostsPageURL)

23. }

24. }

25.

26.

Servicio de sesiones y tokens

1. import { Injectable } from '@angular/core';

2. import { HttpClient } from '@angular/common/http';

3. import { backserv } from 'config';

4. @Injectable({

5.   providedIn: 'root'

6. })

7. export class SessionTokenService {

8.

9.   constructor(private httpClient: HttpClient) { }

10.

11.   private key: string = "nib\_token";

12.

13.   //----------------TOKEN STORAGE ----------------

14.

15.   public writeToken(token: string){

16.     //sessionStorage.setItem(this.key, token);

17.     localStorage.setItem(this.key, token);

18.   }

19.   public readToken(){

20.     //const miDato = sessionStorage.getItem(this.key) || null;

21.     const miDato = localStorage.getItem(this.key) || null;

22.     return miDato;

23.   }

24.

25.   // SOLICITING TOKEN TO BACKEND ----------------

26.

27.   public makeLogin(json: any){

28.     let apiPostsCont = "http://" + backserv.ipnibbackserver + ":" + backserv.portnibbackserver + "/api/auth/login";

29.     return this.httpClient.post<any>(apiPostsCont,json)

30.   }

31.

32.   public makeRegister(json: any){

33.     let apiPostsCont = "http://" + backserv.ipnibbackserver + ":" + backserv.portnibbackserver + "/api/auth/register";

34.     return this.httpClient.post<any>(apiPostsCont,json)

35.   }

36.

37.   // Verify token to backend     ----------------

38.

39.   public verifyToken(json: any){

40.     let apiPostsCont = "http://" + backserv.ipnibbackserver + ":" + backserv.portnibbackserver + "/api/auth/verifyToken";

41.     return this.httpClient.post<any>(apiPostsCont,json)

42.   }

43.

44.   public infoUserToken(json: any){

45.     let apiPostsCont = "http://" + backserv.ipnibbackserver + ":" + backserv.portnibbackserver + "/api/users/get";

46.     return this.httpClient.post<any>(apiPostsCont,json)

47.   }

48.

49.   public userUpdateByToken(json: any){

50.     let apiPostsCont = "http://" + backserv.ipnibbackserver + ":" + backserv.portnibbackserver + "/api/users/update";

51.     return this.httpClient.post<any>(apiPostsCont,json)

52.   }

53.

54.   public verifyAdmin(json: any){

55.     let apiPostsCont = "http://" + backserv.ipnibbackserver + ":" + backserv.portnibbackserver + "/api/auth/verifyAdmin";

56.     return this.httpClient.post<any>(apiPostsCont,json)

57.   }

58. }

59.

60.

Servicio de Posts

import { HttpClient,HttpHeaders } from '@angular/common/http';

2. import { Injectable } from '@angular/core';

3. import { backserv } from 'config';

4. import { Observable } from 'rxjs';

5.

6. @Injectable({

7.   providedIn: 'root'

8. })

9. export class PostsService {

10.

11.   constructor(private httpClient: HttpClient) { } //: Observable<Any>

12.

13.   public postsPages(page:number,size:number,order:string, asc: boolean){

14.     let apiPostsPageURL = "http://"+backserv.ipnibbackserver+":"+backserv.portnibbackserver

15.                   +"/api/public/posts/postsLimited?";

16.     return this.httpClient.get<any>(apiPostsPageURL + `size=${size}&page=${page}&order=${order}&asc=${asc}`)

17.   }

18.

19.   public postsPagesByTag(page:number,size:number,order:string, asc: boolean,tag:string){

20.     let apiPostsPageURL = "http://"+backserv.ipnibbackserver+":"+backserv.portnibbackserver

21.                   +"/api/public/posts/postsfiterbytag?";

22.     return this.httpClient.get<any>(apiPostsPageURL + `size=${size}&page=${page}&order=${order}&asc=${asc}&tag=${tag}`)

23.   }

24.

25.   public getContPosts(){

26.     let apiPostsCont = "http://" + backserv.ipnibbackserver + ":" + backserv.portnibbackserver + "/api/public/posts/count";

27.     return this.httpClient.get<any>(apiPostsCont)

28.   }

29.

30.   public sendNewPost(title: string, tags: string, file: String, token: string){

31.     console.log("Send")

32.     const json = { title: title, tags: tags,base64Img:file, token: token};

33.     let apiPostsSave = "http://" + backserv.ipnibbackserver + ":" + backserv.portnibbackserver + "/api/private/post/postnew";

34.     console.log(apiPostsSave)

35.     return this.httpClient.post<any>(apiPostsSave, json);//, { headers: headers }

36.   }

37.

38.   public getInfoPost(id: String){

39.     let apiPostsCont =

40.     "http://" + backserv.ipnibbackserver + ":" + backserv.portnibbackserver + "/api/private/post/getPostInfo"

41.     + `?id\_post=${id}`;

42.     return this.httpClient.get<any>(apiPostsCont)

43.   }

44.

45.   public getTagsByPostID(id: String){

46.     let apiPostsCont =

47.     "http://" + backserv.ipnibbackserver + ":" + backserv.portnibbackserver + "/api/private/post/getTagsByPost"

48.     + `?id\_post=${id}`;

49.     return this.httpClient.get<any>(apiPostsCont)

50.   }

51.

52.   public sendNewCommentforPost(token\_usuario :String, id\_post:number, message:String){

53.     const json = { token\_usuario:token\_usuario,id\_post:id\_post,message:message};

54.     let apiPostsSave = "http://" + backserv.ipnibbackserver + ":" + backserv.portnibbackserver + "/api/public/comments/save";

55.     return this.httpClient.post<any>(apiPostsSave, json);//, { headers: headers }

56.   }

57.

58.   public getCommentsforPost(id\_post:number){

59.     const json = {id\_post:id\_post};

60.     let apiPostsSave = "http://" + backserv.ipnibbackserver + ":" + backserv.portnibbackserver + "/api/public/comments/get";

61.     return this.httpClient.post<any>(apiPostsSave, json);//, { headers: headers }

62.   }

63.

64. }

Servicio de Configuración

1. import { Injectable } from '@angular/core';

2. import { backserv } from 'config';

3. import { HttpClient } from '@angular/common/http';

4.

5. @Injectable({

6.   providedIn: 'root'

7. })

8. export class ConfAPPService {

9.

10.   constructor(private httpClient: HttpClient) { }

11.

12.   public getNameAPP(type: string){

13.     let apiNameApp = "http://"+backserv.ipnibbackserver+":"+backserv.portnibbackserver

14.                     +"/api/public/conf/get?"+`type=${type}`;

15.     return this.httpClient.get<any>(apiNameApp);

16.   }

17.

18.   public updateConf(json: any){

19.     let apiPostsCont = "http://" + backserv.ipnibbackserver + ":" + backserv.portnibbackserver + "/api/private/conf/change";

20.     return this.httpClient.post<any>(apiPostsCont,json)

21.   }

22. }

23.

24.

Las clases graficas no se incluyen debido a la extensión del proyecto.

Página del proyecto  
https://github.com/CBujeda/TFG-PROYECT-Nine\_Image\_Board

Agradecimientos

En este punto de mi camino académico, me gustaría expresar mi profundo agradecimiento a todas las personas que hicieron posible la realización de este Trabajo de Fin de Grado.

Quiero mostrar mi gratitud a todos los profesores y personal docente del área de Aplicaciones Multiplataforma por brindarme una educación de calidad y fomentar mi crecimiento académico.

Sus enseñanzas y dedicación han dejado una huella perdurable en mi formación.

Principalmente me gustaría agradecer a Rosana Marín Berraondo y a Álvaro Juan Ciriaco debido a su paciencia y sus enseñanzas a lo largo de estos dos años.

Lo físico se puede romper o perder, pero lo que aprendemos siempre es

nuestro y nos ayuda a crear cosas increíbles.

Clara Bujeda Muñoz 2023