Debateo

Autor: Alexandru Stan
Tutor: Maria Teresa Alhama Chapresto
Ciclo: Desarrollo de Aplicaciones Web
Centro: IES Pío Baroja
Fecha: 2023

INDICE

INTRODUCCIÓN

Se trata de una red social muy similar a “Reddit” en versión hispanohablante.

Básicamente lo que se plantea es una aplicación que no se base tanto en las personas como pueden ser Instagram, Facebook... etc, sino que esté dividida en distintas comunidades que representen conceptos, hobbies... etc, dónde diversas personas con intereses y gustos similares puedan reunirse a debatir, charlar, aportar conocimiento, hacer y responder preguntas... etc.

A diferencia de las aplicaciones similares que existen actualmente en el mercado, se intenta desarrollar una mucho más llamativa visualmente e intuitiva.

Por último, la aplicación utiliza una base de datos MySQL.

Una aplicación similar a Debateo podría ser Quora.

PLANIFICACIÓN

- 17/04/2023
  - Página principal de la aplicación, con un formulario de inicio de sesión y otro de registro.
- 08/05/2023
  - Diseño responsive de la página principal.
  - Estructura de la parte backend realizada. (Servicios, Modelo, Controladores..)
  - Inicio de sesión funcional (sin validación de datos)
  - Registro funcional. (sin validación de datos)
- 22/05/2023
  - Front-end de la página de feed y página de perfil de usuario.
- 29/05/2023
  - Scroll infinito en la página feed
- 5/06/2023
  - Scroll infinito en todas las páginas
  - Arreglos menores a nivel de estilos

    - Se añade la tabla “Seen” para rastrear las publicaciones ya vistas por un usuario
    - Crear comunidad
    - Suscribirse
    - Crear publicacion
- 11/06/2023
    - Borrar comunidad

REVISIÓN DE TECNOLOGÍAS

- Visual Studio Code

Es el editor de código que utilizo en todos los proyectos de frontend.

Dispone de una interfaz simple, personalizable e intuitiva, además de ser altamente configurable a través de las miles de extensiones creadas por la comunidad

- Spring Tools Suite 4 for Eclipse

Un IDE de Java basado en eclipse y configurado de tal manera que sea idóneo para proyectos en Spring.

Entre sus ventajas podemos destacar que permite la creación rapida de proyectos Spring Boot a través del Initializr que tiene integrado, con lo cual no tendremos que ir a páginas externas para generar dichos proyectos.

También tiene corrección de codigo y sugerencias personalizadas para Spring.

- HTML, CSS y Javascript

Son las tecnologías que dominan el desarrollo web frontend, nada que destacar en este apartado.

- React

Framework de Javascript para la creación de interfaces basadas en componentes reutilizables.

Lo he elegido por las siguientes razones:

Proporciona una manera mucho más cómoda y rápida de trabajar con el DOM de HTML, lo cual se puede hacer muy tedioso en Javascript vanilla.

Proporciona mejor rendimiento que Javascript vanilla a través de una técnica llamada "Virtual DOM" que consiste en una copia del DOM original de HTML sobre el que realizaremos todos los cambios que queramos. Los cambios en el DOM Virtual serán comparados con el DOM original y solo se actualizarán aquellas secciones que necesiten una actualizacion.

React está hecho para crear proyectos modulares debido a sus componentes, lo cual ayuda a la detección de errores y mantenimiento del código.

Los hooks de React hacen muy fáciles tareas que de cualquier otra forma serían mucho más tediosas, algunos de los hooks que más utilizo son useEffect, useState, y useNavigate

- Redux

Es una libería de React muy útil a la hora de trabajar con el estado de manera global.

Los estados son más que nada variables que al sufrir un cambio de valor provocan el re-renderizado del componente en el que se encuentran.

React no permite transferir props entre componentes hermanos, por lo tanto si necesitamos utilizar un mismo estado en 2

componentes que se encuentran al mismo nivel, Redux es la solucion.

- Material-UI

Se trata de una libreria de componentes de React ya estilizados, no obstante que ya estén estilizados no significa que no puedan ser personalizados a través de temas, props, o simplemente css nativo.

Tiene ventajas y desventajas, aun así, es una libreria que ahorra mucho tiempo a la hora de estilizar las aplicaciones web sin perder en apartado artistico.

- Jquery

Lo utilizo para ocasiones puntuales en las que me ahorra tiempo, como por ejemplo en las animaciones, seleccionar elementos..etc

- Java

Se trata de uno de los lenguajes más utilizados a nivel mundial en el backend, las principales características que me han hecho elegirlo por encima de otros lenguajes de programación como por ejemplo PHP serían:

Es un lenguaje con un tipado fuerte, lo cual hace que el código sea más seguro y legible.

Tiene una sintaxis más moderna y un ecosistema muy amplio dado que es un lenguaje multiplataforma.

- Spring

Un framework muy amplio compuesto de varios módulos, de los cuales utilizo Spring Web y Spring Data JPA

Tiene una comunidad muy activa pese a ser un framework relativamente antiguo.

Proporciona una manera muy facil de crear proyectos con todas las dependencias necesarias a través de Spring Boot.

- HeidiSQL y MySQL

En este caso como gestor de bases de datos utilizo Heidi por simple comodidad.

ESTRUCTURA

A grandes rasgos el proyecto se divide en 2 carpetas:

- app, que contiene todo el frontend

Dentro de app vamos a encontrar el tipico archivo package.json que contiene todas las dependencias necesarias para que el proyecto funcione.

Tambien encontramos el archivo index.html que contiene un único div llamado "root" en el que se van a renderizar todos los componentes de la aplicación y en definitiva va a contener absolutamente todo.

Asimismo también tenemos un archivo llamado index.jsx que se encargará de renderizar el componente App, del cual hablaremos más adelante.

Dentro de la carpeta src vamos a encontrar el código fuente de la aplicación separado en varias carpetas.

- assets

- js
- react
- redux-store

Dentro de assets vamos a encontrar todos los archivos relacionados con los estilos de la página, como pueden ser archivos .css e imágenes.

La carpeta js contiene algunas funciones que he considerado demasiado largas como para escribirlas directamente en los componentes en cuestión, como por ejemplo la funcion de inicio de sesión y crear cuenta.

Dentro de la carpeta React vamos a encontrar una cantidad importante de subcarpetas, lo que se intenta es tener una estructura lo más modular posible para ayudar a la detección de bugs, creacion de nuevas funcionalidades en el futuro y la reutilización de componentes, que es el punto fuerte de React.

Dicho lo anterior, nos encontramos con 2 carpetas y un fichero

- componentes
- paginas
- App.jsx

Empezando por lo más cercano al usuario, tenemos el componente App, que contendrá un enrutador que renderizará un componente u otro según la ruta en la que nos encontremos.

Se trata de una manera de simular una navegación entre distintas páginas, aunque tratandose de una SPA realmente no exista una "navegación" como tal.

El componente App enviará la página en cuestión a index.jsx y este lo renderizará en el index.html

Dentro de páginas tenemos los componentes que conforman una página independiente como tal, como pueden ser la página principal donde tenemos el inicio de sesion y el formulario para la creación de nuevas cuentas.

Por otro lado tenemos la carpeta componentes, que contiene las piezas que conformarán las páginas en sí

Dentro de componentes vamos a encontrar una carpeta llamada "reusable" que como su propio nombre indica contiene todos los componentes que se utilizan en más de una página, como pueden ser la cabecera que contiene una barra de búsqueda y un menú de navegación, como también un componente al que se le pasa la ruta de una imagen y la renderiza por pantalla.

Aparte de la carpeta reusable tenemos una carpeta por cada página, que contendrán los componentes que conforman a la misma

Todas las carpetas anteriormente mencionadas tienen una estructura similar, un archivo por cada sección de la página (body,header y footer)

En caso de que alguna de las secciones estén formadas por varios componentes más pequeños, habrá una carpeta que los contendrá, como la carpeta body que contiene los componentes que conforman la página body.jsx, por ejemplo.

- demo, el proyecto maven que contiene todas las clases java

Para poder ver los archivos de backend de una manera más cómoda, se recomienda importar el proyecto en un IDE (A poder ser Spring Tools Suite for Eclipse, todo lo referente a la importación del proyecto se explica aqui).

En la raíz del proyecto encontramos el archivo pom.xml que contendrá todas las dependencias maven necesarias para que el proyecto funcione correctamente.

En la carpeta src/main/java vamos a encontrar el paquete es.debateo, que contendrá la clase DemoApplication.java con el método main junto a 4 paquetes:

- Controladores
- DTO
- Modelos
- Servicios
- Repositorios

FUNCIONAMIENTO BACKEND

• Backend

El backend sigue el patrón de diseño "modelo-vista-controlador", por lo tanto, el código Java está dividido en los 4 paquetes mencionados en el apartado anterior.

1. Controladores

El paquete de los controladores contiene las clases que van a estar esucuchando las peticiones de los clientes, es decir, los endpoints.

En pos de la modularización y la legibilidad del código existe un controlador por cada entidad de la aplicación (usuarios,comunidades... etc).

Todos los controladores son de tipo REST, ya que al estar trabajando con React no voy a devolver una vista como tal (en html por ejemplo) sino que simplemente devuelvo información que será recibida y tratada por el frontend.

Los controladores estan mapeados según el tipo de peticion HTTP que se está realizando, en total utilizo los 4 métodos más importantes (GET,PUT,POST,DELETE)

Todos los controladores devuelven un objeto de tipo ResponseEntity, el cual acepta como parámetros un objeto genérico y un código HTTP.

En la mayoria de los controladores se manejan los códigos de estado HTTP más tipicos, como pueden ser

- 200 OK
- 404 NOT FOUND
- 400 BAD REQUEST

Cuando los controladores reciben una petición van a llamar a uno de los métodos de la clase servicio correspondiente, a la cual accede a través de una inyección de dependencias gracias al contenedor de beans de Spring.

2. DTO

El paquete DTO, por sus siglas Data Transfer Object, contiene clases que tienen la unica finalidad de envolver datos y transportarlos.

Al estar trabajando con un lenguaje de programación con un tipado fuerte como Java, las objetos que se manejan tienen que ser una instancia de una clase (al contrario que en javascript, donde se pueden crear objetos sin necesidad de que tengan una clase asociada).

Normalmente utilizo DTO's cuando es necesario manejar datos de tablas distintas en un mismo contenedor, como por ejemplo la clase ServiceResponse.

ServiceResponse es una clase que actúa como intermediario entre los controladores y los servicios.

Una vez los servicios terminen de ejecutar toda la lógica, devolveran todos los datos necesarios al controlador mediante ServiceResponse.

3. Modelo

Como mencioné anteriormente, estoy utilizando el ORM Hibernate que se encuentra integrado dentro del módulo de Spring Data JPA, por lo tanto, necesito clases que sean representaciones de la base de datos.

Es un paquete compuesto de clases que representan tablas, con sus respectivos constructores, getters y setters.

Por otro lado dentro de este paquete encontramos otro llamado ComplexID que contiene wrappers para las clases que tienen varias claves primarias. (más adelante se explica la necesidad)

4. Repositorios

Se trata de interfaces que extienden de la interfaz JpaRepository, la cual exige que se le pasen como tipos genericos una entidad y un identificador.

Aquí entra en acción el paquete ComplexID, la clase JpaRepository admite por defecto un último campo ID, por lo tanto se necesita un objeto que los envuelva a todos en caso de que haya más de uno.

Hay una interfaz por cada entidad y estas serán inyectadas en las clases Servicio, proporcionando acceso a los métodos más típicos para realizar acciones sobre la base de datos, como pueden ser findAll(), findById(), existsById(), deleteById()... etc.

Además, tambien se permite definir consultas personalizadas, tanto en SQL nativo como en JPQL(Java Persistence Query Language), este último siendo un lenguaje basado en SQL adaptado a Java.

5. Servicios

Contienen toda la lógica de la aplicación, y sirven como intermediarios entre los repositorios, a los cuales les van a solicitar métodos y funcionalidades, y los controladores, a los cuales les van a devolver información.

FUNCIONAMIENTO FRONTEND

- Principal

Una vez desplegada la aplicación, al ejecutarla nos encontramos con una página con un carusel y 2 formularios.

También hay un enlace de color naranja que al ser clickeado alternará entre el formulario de Login y Registro

Al registrar y enviar el formulario de registro nos puede aparecer 1 de 2 mensajes mostrandonos si la operación ha tenido exito o no.

- Feed

Una vez estemos logueados, se guardará un objeto en sessionStorage con los datos del usuario (exceptuando la contraseña por motivos de seguridad)

Cuando el componente del feed se renderice, hará una peticion al servidor y traerá una página de 5 publicaciones que sean de comunidades a las que estemos siguiendo.

Al mostrar los componentes por pantalla, se le pondra un Observer al 3 elemento de cada página, dicho observer tendrá un callback

que se ejecutará cada vez que dicho elemento aparezca en pantalla, lo cual provocará que se realice una segunda petición al servidor que traerá otras 5 publicaciones.

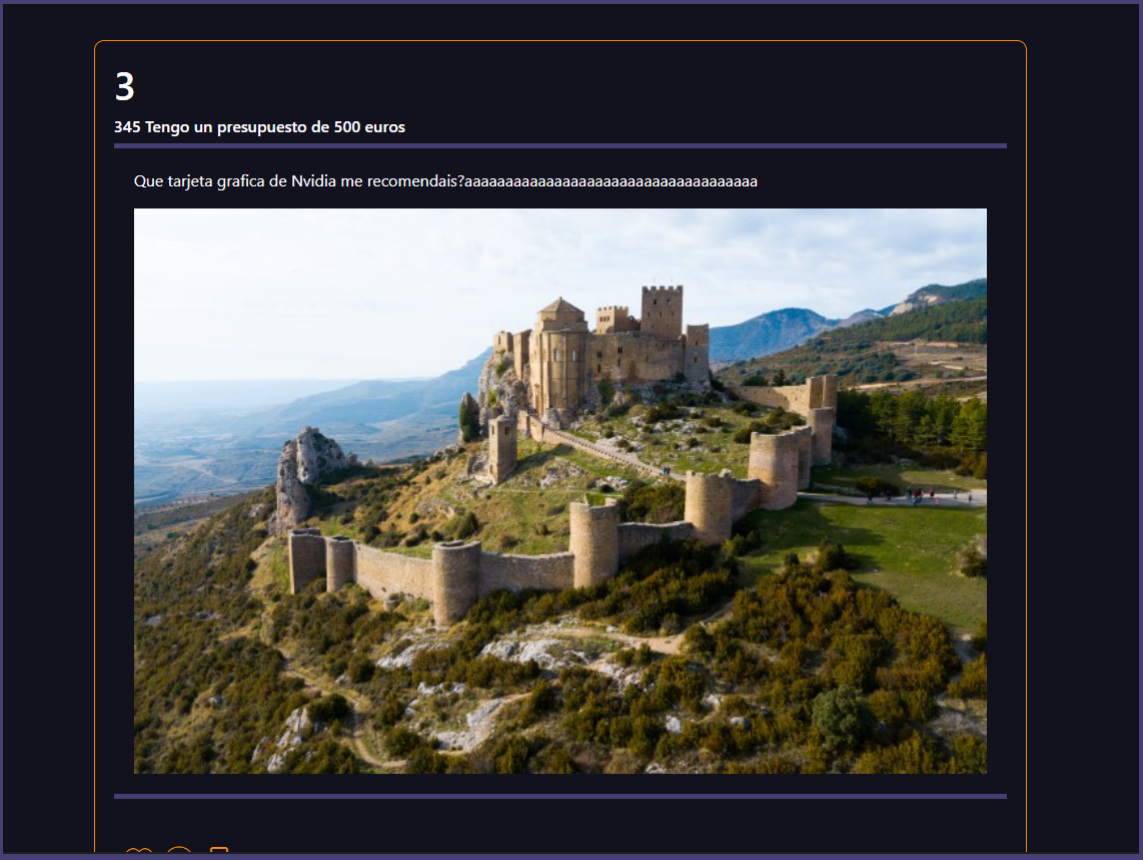
Cada vez que el servidor envía una página de publicaciones al cliente en paralelo hará una llamada a la base de datos y dejara constancia de que el usuario X ha visto las publicaciones A,B,C,D y E.

Lo que se intenta conseguir con esta operación es evitar que, si el usuario reinicia la página, le salgan publicaciones que ya ha visto.

Las publicaciones constan de una cabecera un body y un footer.

En la cabecera encontraremos el titulo de la publicacion y el id de la misma por motivos de testing.

El body tiene un tamaño máximo definido para evitar que una publicación ocupe demasiado espacio en pantalla, por ello viene integrado con un scroll.



Las imágenes estan guardadas en la base de datos y tambien tienen un tamaño máximo fijo.

En la misma página tambien podemos encontrar una cabecera con una barra de búsqueda y un menú.

La barra de busqueda hara una petición al servidor buscando comunidades que empiecen por la cadena introducida por el usuario (evento onchange)

Por razones de rendimiento, declaro la función de búsqueda dentro de un setTimeout de 300 milisegundos, de esta forma si la persona que hace la búsqueda escribe medianamente rápido, la petición no se hará hasta que acabe de escribir.

Dentro de perfil encontraremos 2 secciones.

- Una sección en la que se detallan los datos del usuario conectado actualmente
- Una sección en la que se muestran las publicaciones que ha publicado dicho usuario.

- Community

Dentro de community tendremos una cabecera con la información de la comunidad que estamos investigando y todas las publicaciones asociadas.

Tambien disponemos de un boton cuyo proposito es crear nuevas publicaciones.

Dentro de esta misma página, si el que la esta consultando es el creador de la comunidad, en cada publicación aparecerá un boton que le permitirá borrar la misma.

CONCLUSIONES

- Imagenes

Decidí guardar las imagenes en la propia base de datos como objetos binarios, lo cual puede ser una buena idea en aplicaciones que no dependan tanto de las imagenes.

Los archivos BLOB ocupan mucho espacio y a la hora de enviarlos por HTTP generan JSONs demasiado largos.

Los BLOB no pueden ser enviados a través de JSON, por lo tanto hay que codificarlos a base64.

Una vez el cliente recibe la imagen, la tiene que decodificar, convertir el array de bytes resultante en un BLOB y extraer una URL del mismo.

Todos esos procesos necesitan muchos recursos y no es viable para aplicaciones grandes.

Aun sabiendo todo esto decidí tirar hacia delante y hacerlo de esta forma, no tanto por las ventajas que pueda tener (ya que no tiene ninguna) sino por simplemente aprender más.

En proyectos futuros probablemente me decante por guardar las imagenes directamente en el sistema de archivos del servidor.

- Exceso de tecnologías

Como ya mencioné en el apartado de revisión de tecnologías, la gran mayoria del código de mi proyecto esta escrito en lenguajes / frameworks que no se han visto en clase. (React y Spring), lo cual conllevó mucho tiempo que tuve que invertir en aprendizaje, dejandome bastante apurado a la hora de programar como tal.

Este exceso de tecnologías ha provocado que muchas de las funcionalidades que tenia en mente no acabasen implementadas a la fecha de entrega del proyecto.

Aun así considero que recurrir a tecnologías nuevas ha sido un gran acierto a largo plazo ya que se trata de lenguajes y frameworks muy utilizados y valorados en la industria.

Por otro lado también he perdido mucho tiempo experimentando con errores que eran fruto de mi inexperienencia con dichas tecnologías.

ESPECIFICACIONES

En este aplicación los roles de usuario giran en torno a qué pueden hacer en cada comunidad.

A grandes rasgos nos podemos encontrar 3 tipos:

- CREADOR:

Es el creador de una comunidad en particular, dentro de la misma tiene la posibilidad de eliminar cualquier publicación y asignar MODERADORES en la comunidad en cuestión.

El único usuario que puede dar permisos a otros es el CREADOR.

- MODERADOR

Puede eliminar publicaciones en una comunidad en particular, pero no puede acceder a la pestaña de administración.

- MIEMBRO

Es un usuario estandár, puede borrar sus propias publicaciones pero no las de los demás.

Todos los usuarios tienen el mismo punto de acceso y navegan por las mismas páginas.

Excepto CREADOR, que tiene acceso a una página adicional llamada Administración.

En cada comunidad habrá un icono para indicar que tipo de permiso tiene el usuario logueado.

 Una corona indica que eres el creador de la comunidad

 Este icono indica que eres moderador.

Si no se muestra ningún icono, eres MIEMBRO.

FUTURO

Hay muchas cosas por añadir / mejorar

- Un sistema de mensajeria entre usuarios
- Un sistema de seguridad para la API con JWT o similar
- Sustituir CSS por Tailwind, intentaré conseguir que los estilos estén encapsulados en un solo componente.

ANEXOS

- Base de datos

La base de datos está compuesta por 8 tablas:

categories(id_category,category_name)

comments(comment_id,post_id,username,comment_date,

comment_text)

communities(community_id,community_name,community_description,community_image,community_background_image,community_members,community_creator,sensitive_content,categoria)

likes(username,post_id)

posts(publication_id,user,community,likes,publication_title,publication_body,publication_image)

seen(username,publication_id)

subscriptions(username,community_id,subscription_date,subscription_level)

users(username,password,name,birth_date,mail)

replies(replyId,username,commentId,replyDate,replyText)

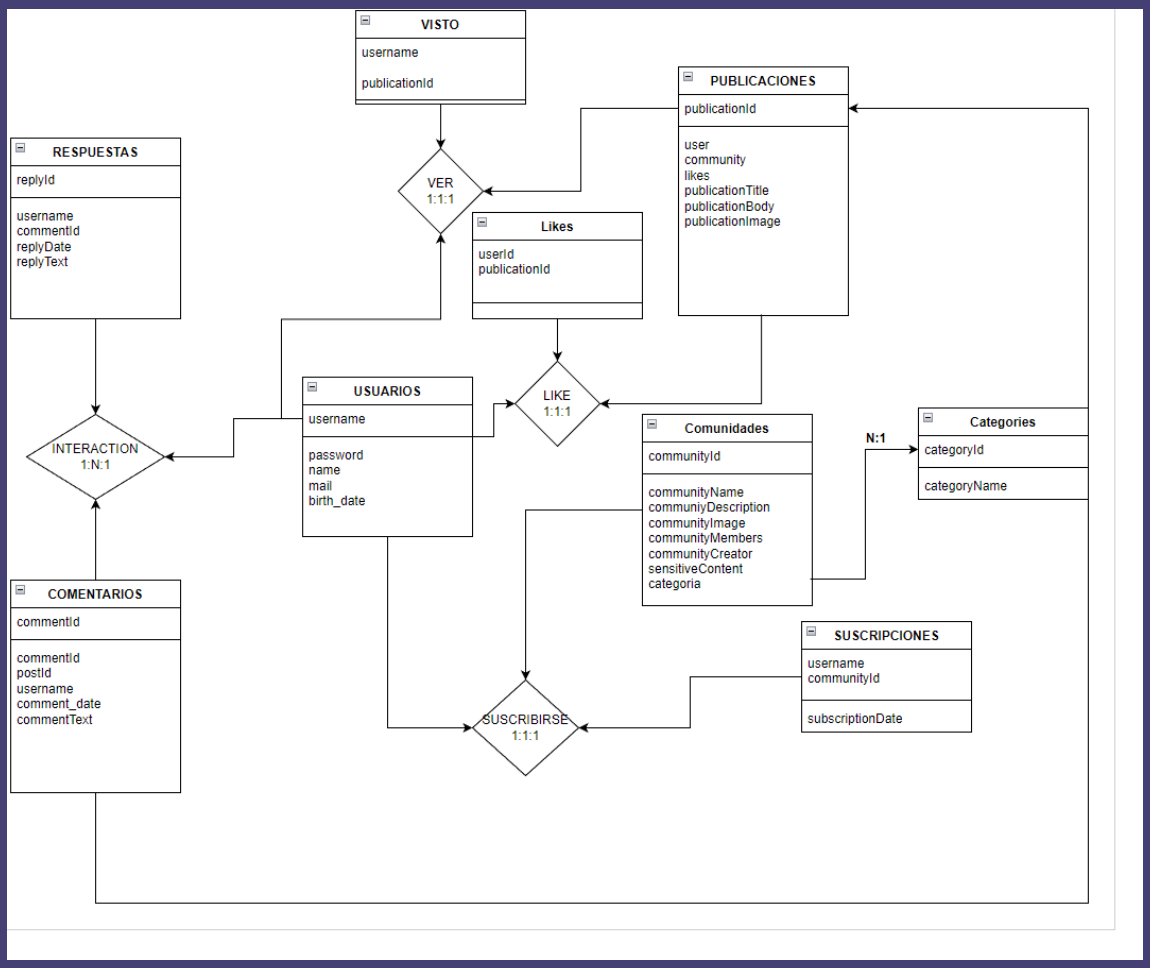
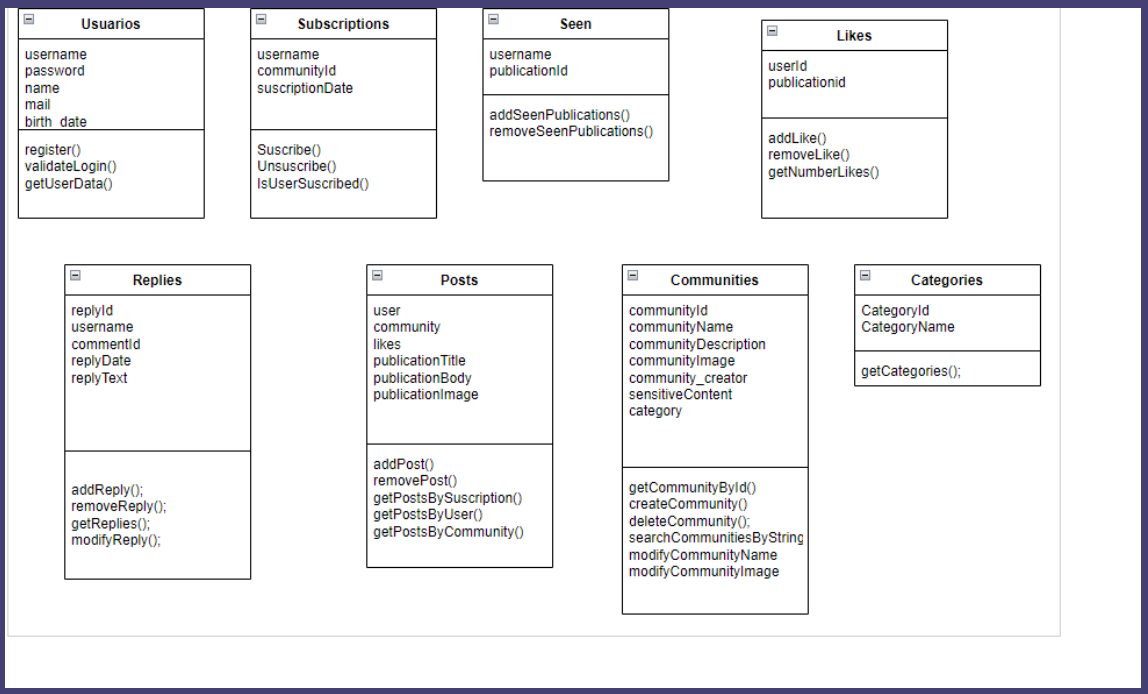


DIAGRAMA DE CLASES


Usuarios
username
password
name
mail
birth_date
register()
validateLogin()
getUserData()
Subscriptions
username
communityId
suscriptionDate
Suscribe()
Unsuscribe()
IsUserSuscribed()
Seen
username
publicationId
addSeenPublications()
removeSeenPublications()
Likes
userId
publicationid
addLike()
removeLike()
getNumberLikes()
Replies
replyId
username
commentId
replyDate
replyText
addReply();
removeReply();
getReplies();
modifyReply();
Posts
user
community
likes
publicationTitle
publicationBody
publicationImage
addPost()
removePost()
getPostsBySuscription()
getPostsByUser()
getPostsByCommunity()
Communities
communityId
communityName
communityDescription
communityImage
community_creator
sensitiveContent
category
getCommunityById()
createCommunity()
deleteCommunity();
searchCommunitiesByString
modifyCommunityName
modifyCommunityImage
Categories
CategoryId
CategoryName
getCategories();

BIBLIOGRAFÍA

https://www.pildorasinformaticas.es

https://openwebinars.net/accounts/login/

https://stackoverflow.com

https://midu.dev

https://www.w3schools.com