La Pluma Digital

Marcos Arjona Comino

Proyecto 2ºDAM A

Índice:

1. Introducción

1.1. Resumen del proyecto

La Pluma Digital es una red social pensada especialmente para lectores y escritores. La idea surgió tras hablar con varias personas aficionadas a la lectura, y darme cuenta de que, aunque hay redes sociales para casi todo, no existe una donde los amantes de los libros puedan reunirse para hablar tranquilamente sobre lo que leen, los autores que siguen o simplemente para descubrir nuevas lecturas.

Muchas veces, quienes leemos sentimos que nos falta un espacio donde poder comentar lo que nos ha parecido un libro, debatir sobre una historia, o incluso encontrar a otras personas con gustos parecidos. Y lo mismo pasa con quienes escriben: a menudo no tienen una plataforma donde compartir lo que crean ni recibir feedback real de lectores interesados.

De ahí nace La Pluma Digital: una red social donde poder conocer gente que también disfrute de la lectura, comentar libros, seguir a tus autores o autoras favoritas, e incluso ver qué libros están siendo los más comentados por la comunidad. Además, si alguien quiere compartir un libro propio, también puede hacerlo.

En definitiva, es un lugar pensado para conectar a personas que tienen algo en común: su amor por los libros. Un espacio donde la lectura no es solo un pasatiempo, sino una excusa para conocer gente, compartir ideas y descubrir nuevas historias.

1.2. Explicación de la aplicación:

Una de las funciones principales es la posibilidad de seguir a otros usuarios. Si dos personas se siguen mutuamente, pueden ver el correo electrónico del otro, por si en algún momento quieren contactar directamente. Esto puede ser útil tanto para hacer nuevas amistades como para posibles colaboraciones literarias o clubs de lectura.

Además, cada usuario puede guardar una lista de escritores favoritos, y acceder fácilmente a los libros que han escrito. Esta función permite seguir la trayectoria de autores que te gustan o descubrir nuevas publicaciones suyas sin tener que buscarlas cada vez.

Por supuesto, los libros son el eje central de la aplicación. Cada uno tendrá una ficha donde cualquier usuario podrá dejar comentarios, ya sea una opinión, una recomendación o una crítica constructiva. Así, otras personas podrán leer distintas valoraciones antes de decidir si quieren leer ese libro o no. También es una forma muy interesante de generar conversación entre lectores, más allá del simple “me gustó o no me gustó”.

Otra sección destacada es la de rankings, donde se muestran los libros y autores más populares de la plataforma. Estos rankings se generan automáticamente en función del número de comentarios que han recibido, lo cual refleja qué obras están generando más movimiento dentro de la comunidad. Es una forma rápida de saber qué se está leyendo, qué está dando que hablar y quizás descubrir algo que no conocías.

Finalmente, los usuarios también podrán utilizar un buscador de libros, con el que podrán consultar si un título existe en la base de datos, ver sus detalles, leer lo que otras personas opinan e incluso encontrar libros similares. Es una herramienta muy útil tanto para quienes van con una idea clara como para quienes simplemente quieren explorar nuevas lecturas.

En resumen, La Pluma Digital es mucho más que una red social: es un punto de encuentro para personas que aman leer y escribir. Un espacio donde se puede opinar libremente sobre libros, descubrir autores interesantes y, sobre todo, conectar con gente con la que compartir esa misma pasión por la literatura.

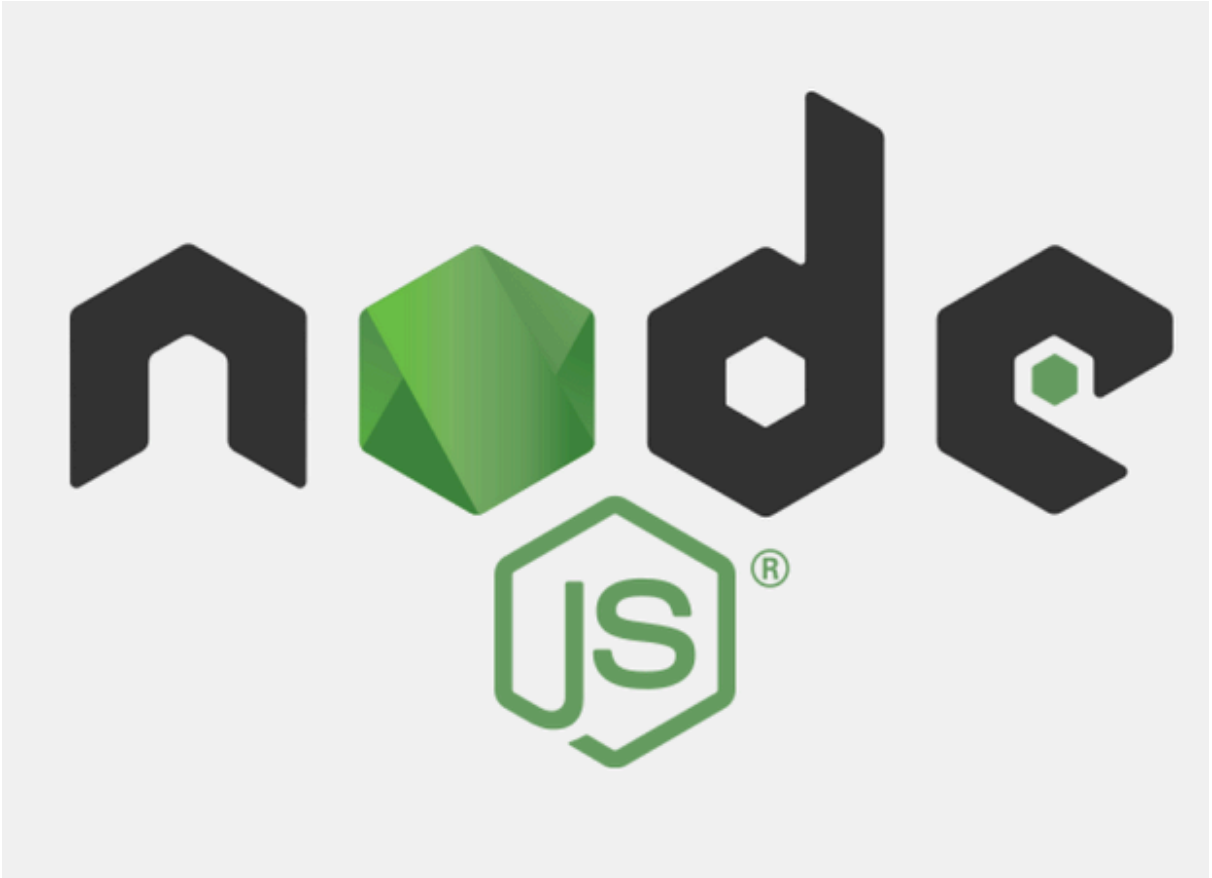
1.3. Tecnologías usadas

Para el desarrollo de La Pluma Digital he utilizado un conjunto de tecnologías que me permiten mantener el proyecto bien estructurado, funcional y fácil de mantener. He apostado por herramientas modernas, pero también prácticas, que me han permitido enfocarme tanto en la lógica como en la experiencia del usuario.

Backend:

El servidor está construido con Node.js, que es una tecnología muy versátil basada en JavaScript. He elegido Node porque, además de ser muy popular, permite escribir código del lado del servidor con un lenguaje que ya conozco bien. Esto agiliza mucho el desarrollo y facilita la integración con otras partes del proyecto.

Para manejar las rutas, peticiones y middleware, utilizo Express.js, un framework muy ligero y flexible que se adapta muy bien a este tipo de aplicaciones.



Base de datos:

La información de usuarios, libros, autores, comentarios, etc., se guarda en una base de datos relacional MySQL. Es una herramienta muy estable y potente, ideal para manejar grandes cantidades de datos estructurados y relaciones entre ellos (como un autor y sus libros, o un usuario y sus publicaciones).

El proyecto está contenedorizado con Docker. Esto significa que todo el entorno de la aplicación (el servidor, la base de datos, las dependencias...). Se puede empaquetar en contenedores que funcionan igual en cualquier máquina. Usar Docker me ha ayudado a que el proyecto sea mucho más fácil de desplegar y probar en diferentes entornos sin preocuparme por configuraciones locales.

Con Docker Compose, puedo levantar tanto el backend como la base de datos con un solo comando, lo que acelera el desarrollo y también permite que cualquier otra persona pueda ejecutar el proyecto sin complicaciones técnicas.

Frontend:

En la parte visual he utilizado Pug, un motor de plantillas que permite generar HTML de forma dinámica y ordenada. Esto hace que las vistas sean más fáciles de mantener y modificar. Además, gracias a los layouts reutilizables, puedo mantener coherencia en el diseño sin repetir código innecesariamente.

Autenticación y Seguridad:

Para asegurar las rutas privadas y gestionar el inicio de sesión, utilizo JWT (JSON Web Tokens), que permite mantener sesiones de forma segura y ligera. Las contraseñas se cifran con bcrypt, para que nunca se almacenen en texto plano.

Gestión de fechas:

Para trabajar con fechas y horas, algo muy común en publicaciones y rankings, utilizo Moment.js, que simplifica mucho el formato y cálculo de fechas.

2. Requisitos

2.1. Requisitos funcionales

La aplicación está diseñada para ofrecer una experiencia completa y cómoda a cualquier persona interesada en la lectura o la escritura. A continuación, se detallan las principales funciones que ofrece el sistema:

Autenticación y Autorización

Para acceder a ciertas funcionalidades, el sistema incluye un sistema de usuarios con distintas opciones:

- Permite el registro de nuevos usuarios, de forma sencilla.
- Ofrece inicio y cierre de sesión para mantener una sesión segura.
- Utiliza tokens JWT para verificar la identidad del usuario y proteger las rutas privadas.
- Muestra contenido adaptado según el rol del usuario, ya sea usuario estándar, moderador o administrador. Cada uno tiene permisos distintos, lo que ayuda a organizar y gestionar mejor la plataforma.

Gestión de Libros y Autores

Una de las bases de esta red social es su sistema para gestionar libros y autores, tanto para consulta como para publicación. En esta parte:

- Se puede ver el listado completo de libros disponibles.
- También se muestra el listado de autores registrados.
- Los usuarios con cuenta pueden subir libros nuevos a la plataforma.
- Es posible acceder a una vista detallada tanto de cada libro como de cada autor, con información clave y publicaciones asociadas.

Ranking

El sistema genera distintos rankings para destacar los contenidos más activos dentro de la comunidad:

- Crea rankings de libros en función del número de veces que han sido mencionados en publicaciones.
- También genera rankings de autores con el mismo criterio.
- Los rankings se muestran en formato tabla, incluyendo el nombre, una imagen representativa y su posición actual.

Perfil de Usuario

Cada usuario registrado tiene acceso a su perfil personal, desde donde puede:

- Consultar su posición en el ranking de publicaciones.
- Ver todas las publicaciones que ha hecho.
- Acceder a su descripción personal, la cual puede modificar en cualquier momento.

Publicaciones

Las publicaciones son el centro de la actividad dentro de La Pluma Digital. En este sentido:

- Los usuarios pueden escribir y compartir contenido relacionado con libros o autores.
- Cada publicación está vinculada a un libro o autor específico a través de sus respectivos identificadores, lo que permite una organización clara y coherente del contenido.

2.2. Requisitos no funcionales

Además de las funcionalidades principales, la aplicación también cumple con una serie de requisitos técnicos y de diseño que garantizan que el sistema funcione de forma segura, eficiente y agradable para el usuario.

Seguridad

La protección de los datos y el control de accesos son prioritarios. Por eso, se han implementado varias medidas:

- Los tokens JWT están firmados con una clave secreta robusta para evitar manipulaciones
- Las contraseñas de los usuarios se almacenan de forma segura, utilizando técnicas de cifrado como `bcrypt`
- Las cookies se configuran con opciones como `httpOnly` y `secure` en entornos de producción, para evitar accesos no autorizados desde el navegador.

Rendimiento

El sistema está optimizado para responder rápido y escalar si es necesario:

- Las consultas más pesadas, como las del ranking, se gestionan con funciones avanzadas de SQL, como las ventanas de clasificación (`RANK()`), que permiten obtener resultados eficientes sin sobrecargar la base de datos.

- Las rutas protegidas realizan una validación del token JWT antes de acceder a la base de datos, ahorrando recursos cuando el usuario no

está autenticado.

Usabilidad

La experiencia del usuario se ha tenido en cuenta desde el inicio:

- La interfaz es clara, con una navegación sencilla entre secciones como libros, autores, rankings y publicaciones
- El encabezado de la página cambia de forma dinámica en función de si el usuario ha iniciado sesión o no, facilitando el acceso a las acciones más relevantes en cada momento (como iniciar sesión, cerrar sesión o acceder al perfil

Mantenibilidad

El código está estructurado para facilitar su mantenimiento y evolución futura:

- Se sigue una arquitectura clara y modular, basada en controladores, rutas, middlewares y vistas.

- Las vistas están construidas con plantillas reutilizables mediante Pug, lo que permite mantener una coherencia visual y reducir la duplicación de código.

- Cada función del sistema está separada por responsabilidad, haciendo más fácil detectar errores, aplicar mejoras o escalar el proyecto.

Compatibilidad

El sistema ha sido probado y funciona correctamente en los principales navegadores modernos, como Google Chrome, Mozilla Firefox y Microsoft Edge.

Contenedorización con Docker

Para facilitar la instalación, despliegue y portabilidad del proyecto, se ha utilizado Docker. Esta herramienta permite encapsular toda la aplicación, junto con sus dependencias y configuraciones, en contenedores estables y reproducibles. Algunas ventajas que aporta Docker al proyecto son:

- **Facilidad de despliegue**: se puede levantar el entorno completo con un simple `docker-compose up`, sin necesidad de instalar dependencias manualmente.

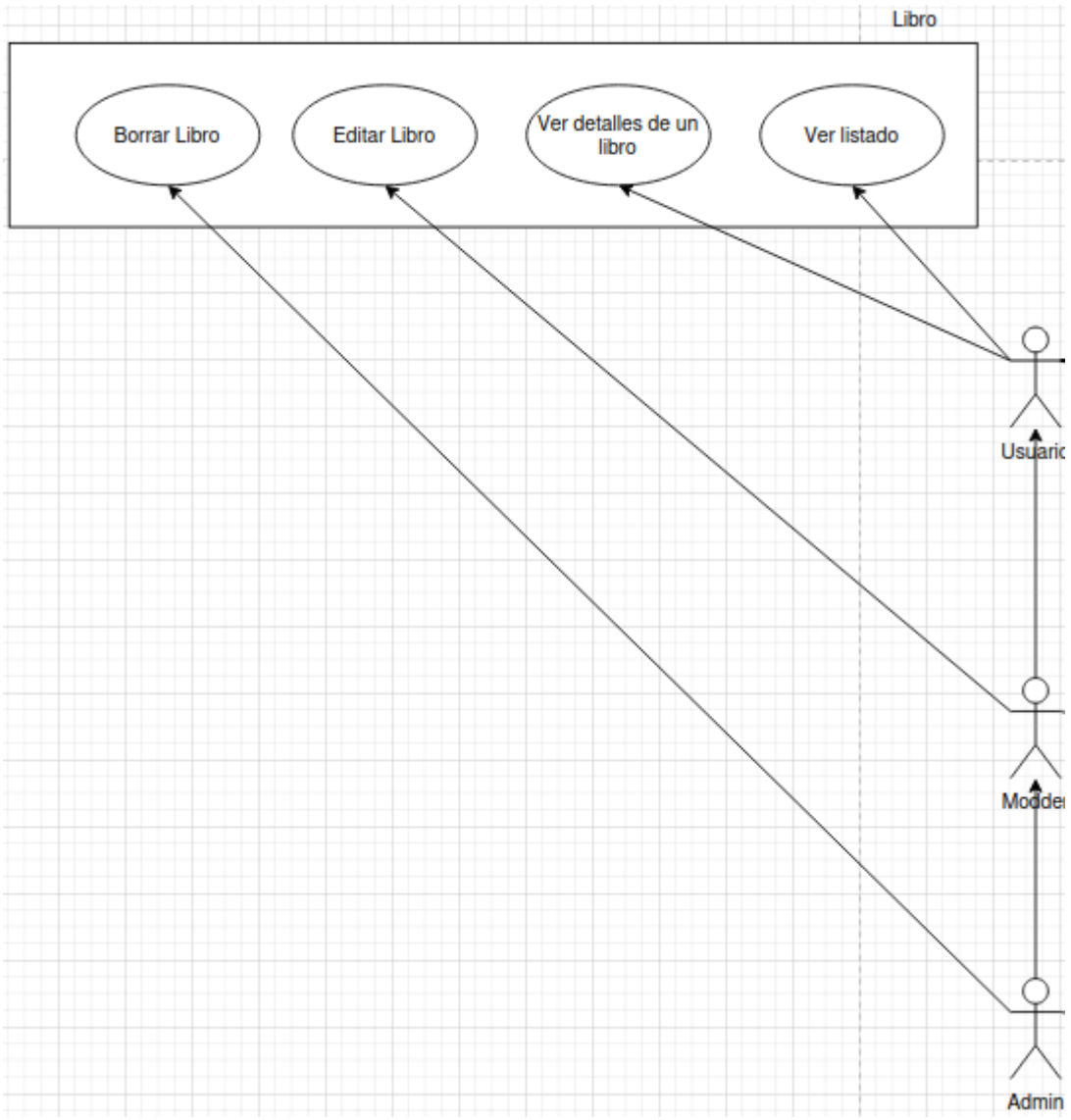
- **Entorno unificado**: tanto en desarrollo como en producción, el sistema se comporta igual, lo que reduce errores por diferencias de configuración.

- **Aislamiento**: cada servicio (por ejemplo, backend, base de datos) corre en su propio contenedor, lo que mejora la seguridad y la organización del entorno.

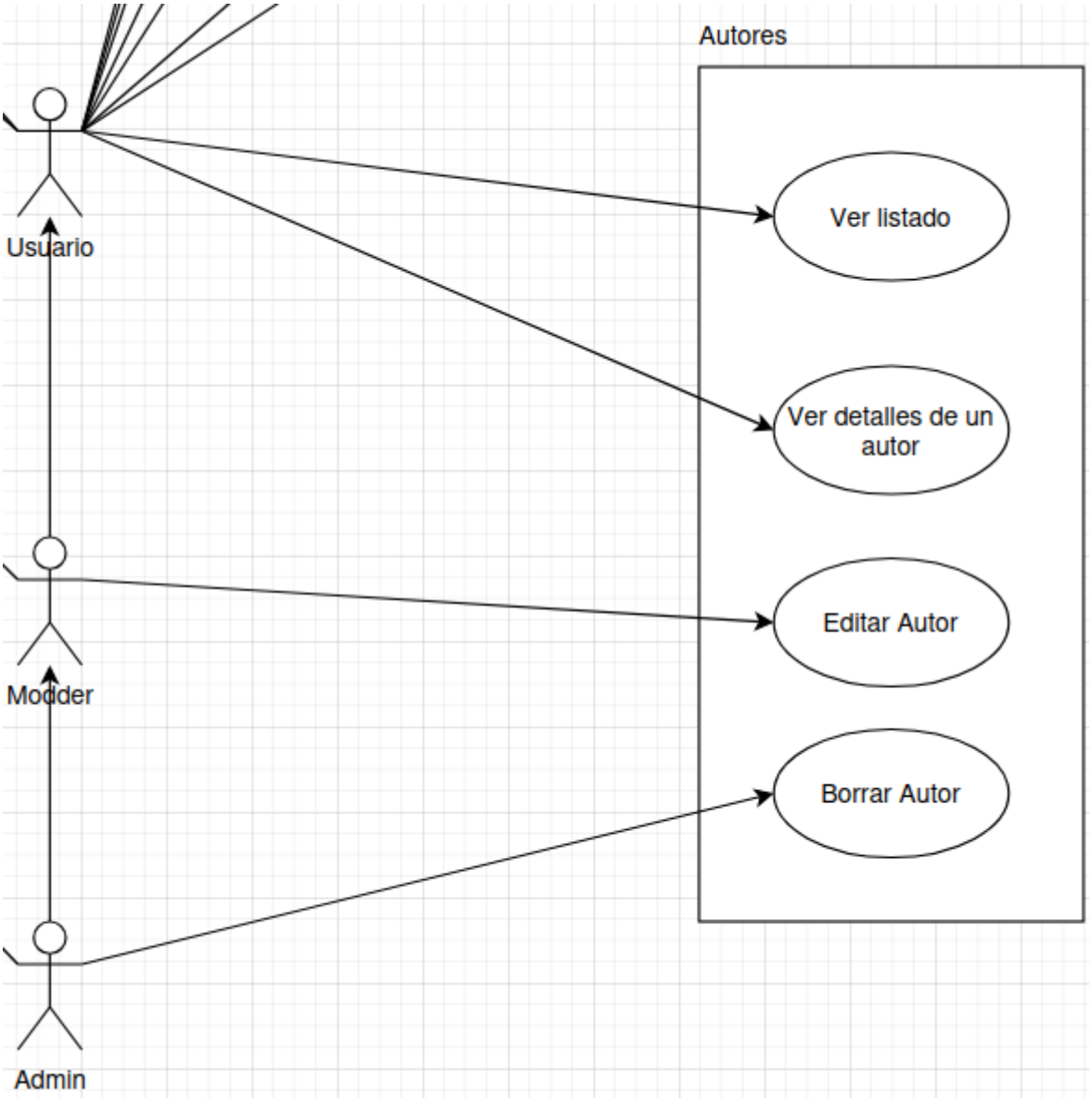
3. Diagramas y Casos

3.1. Casos de uso:

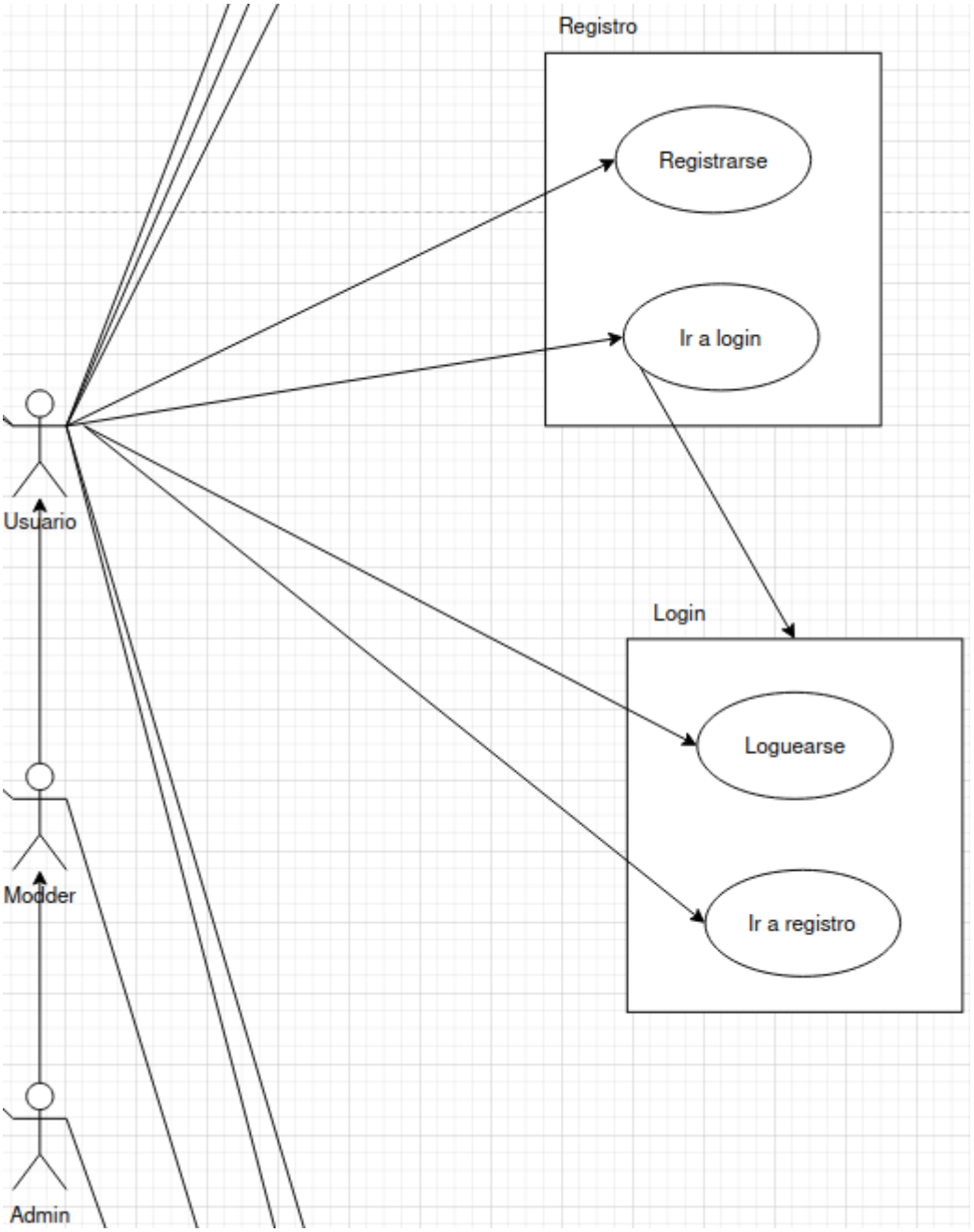
- Libro caso de uso:



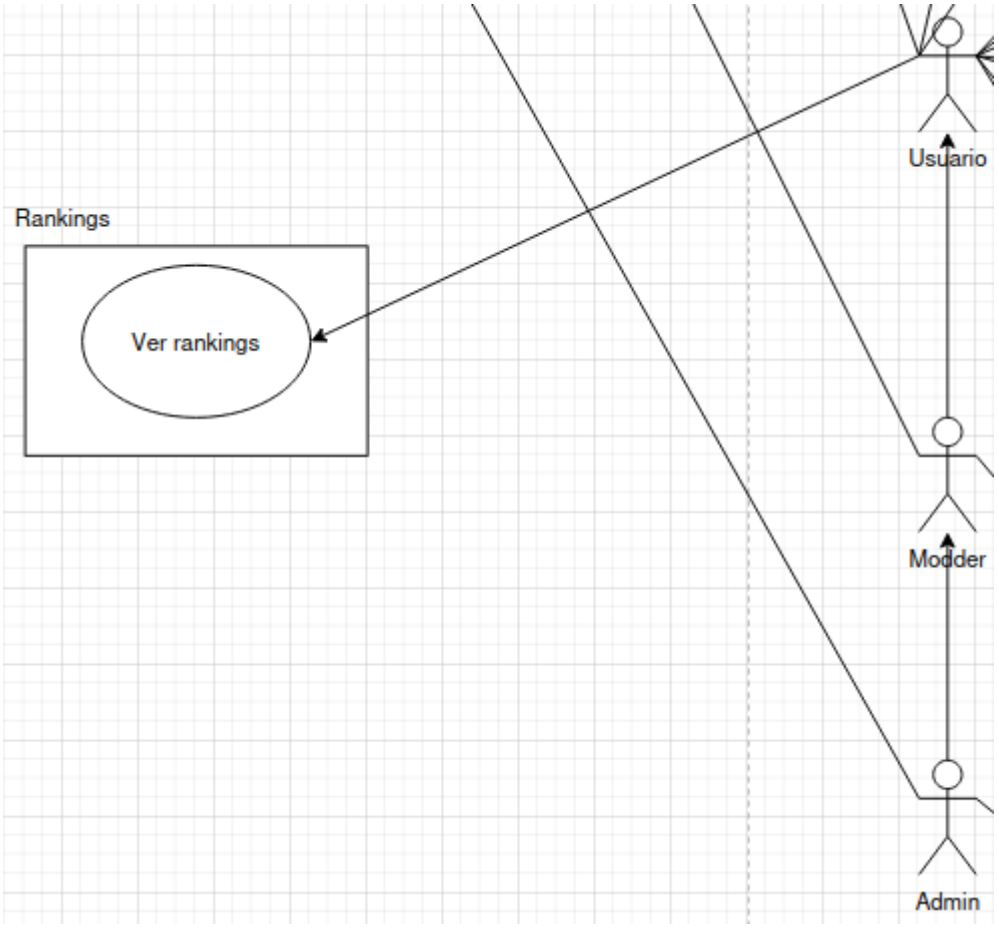
- Autores caso de uso:



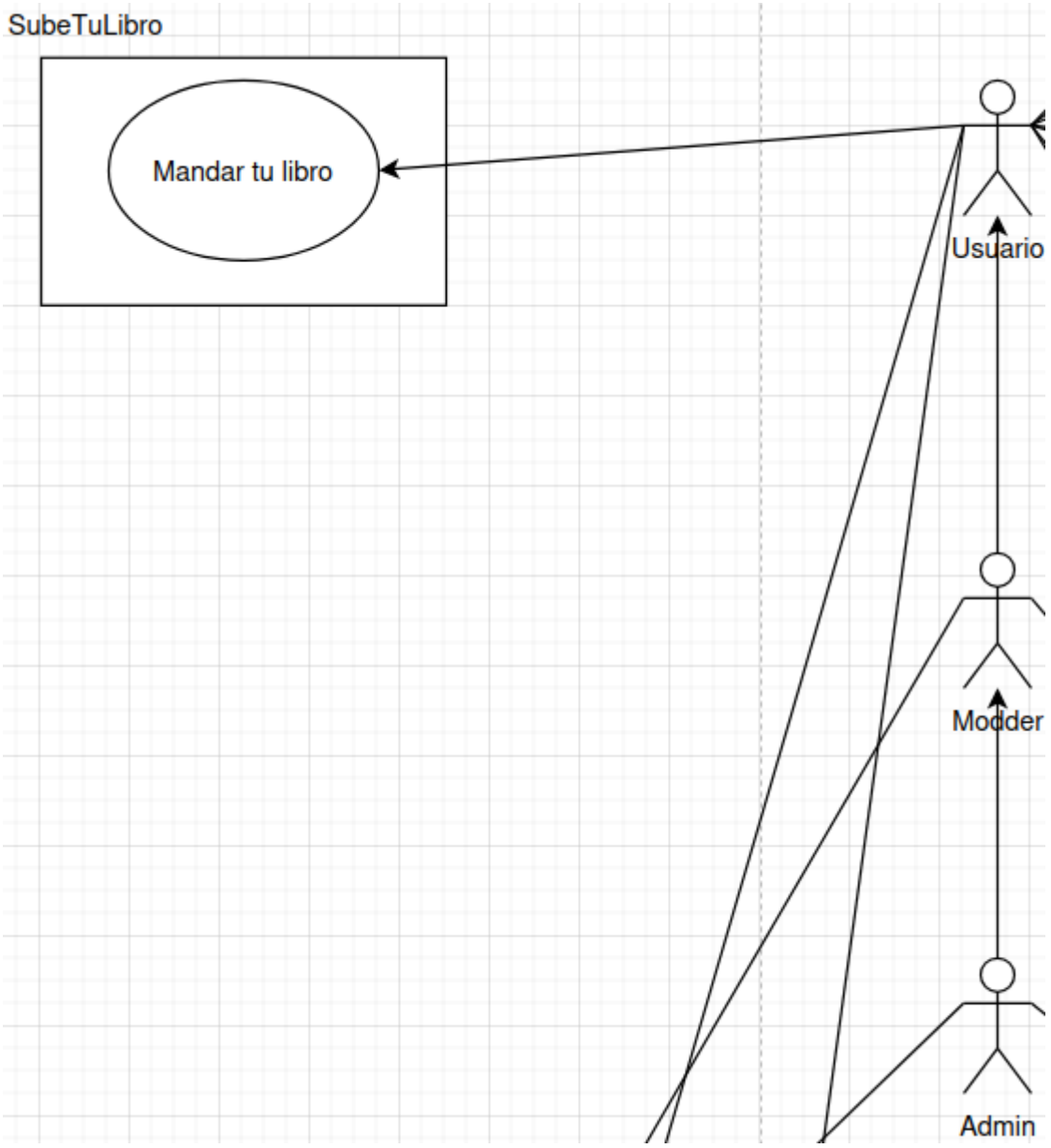
- Login y registro:



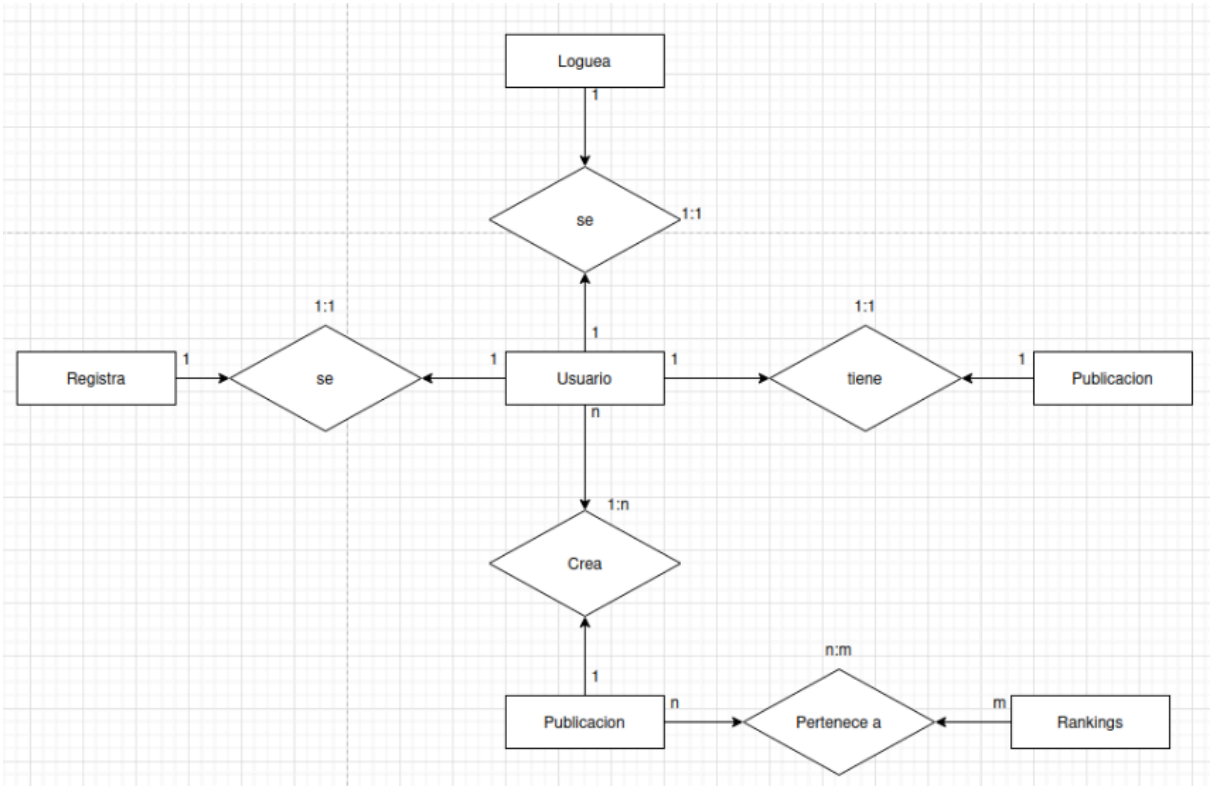
- Ranking:



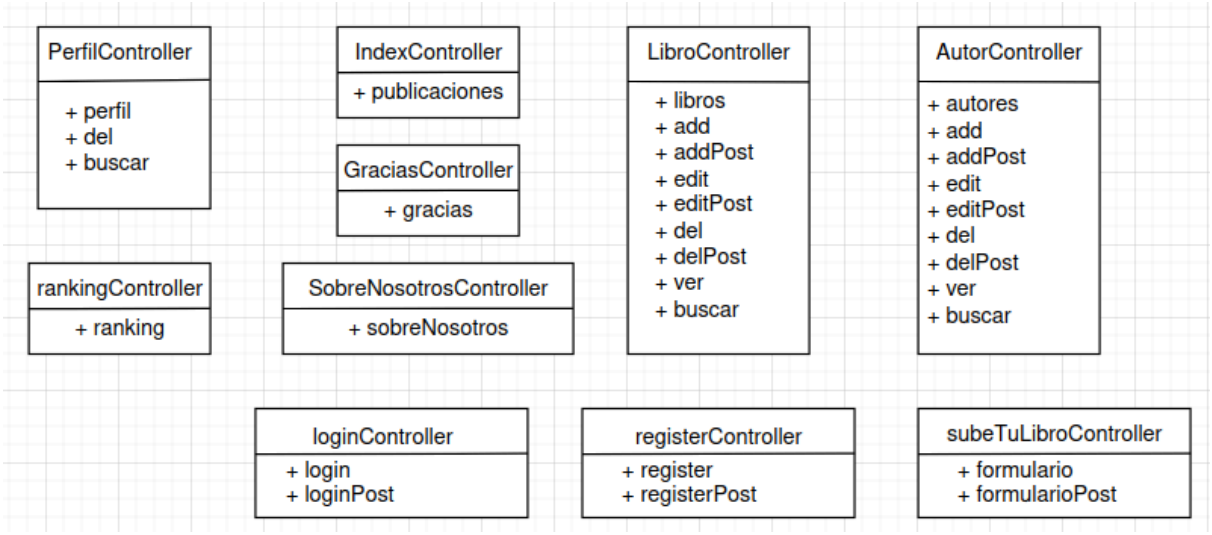
- Sube tu libro:



3.2. Diagrama Entidad-Relación

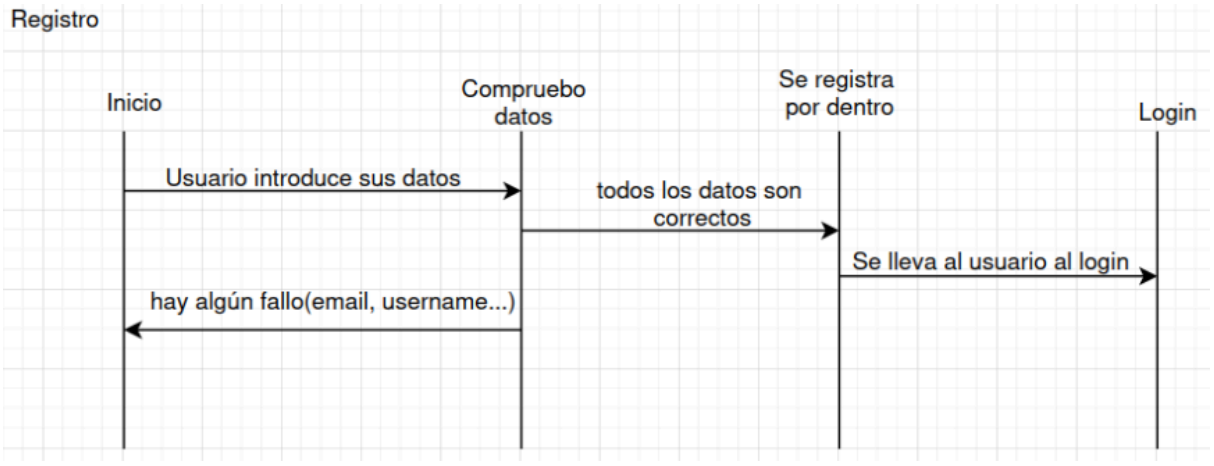


3.3. Diagramas de clases

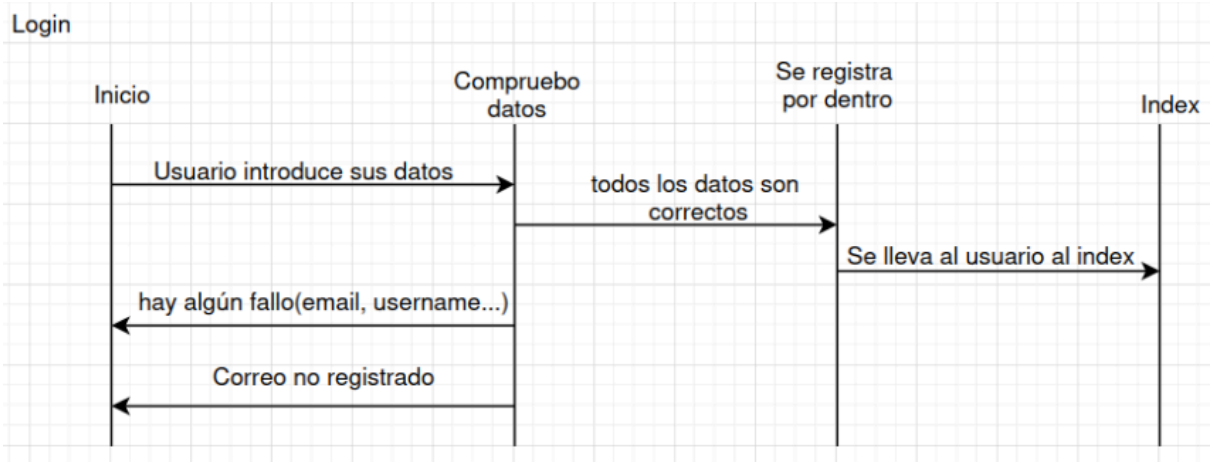


3.4. Diagrama de secuencia

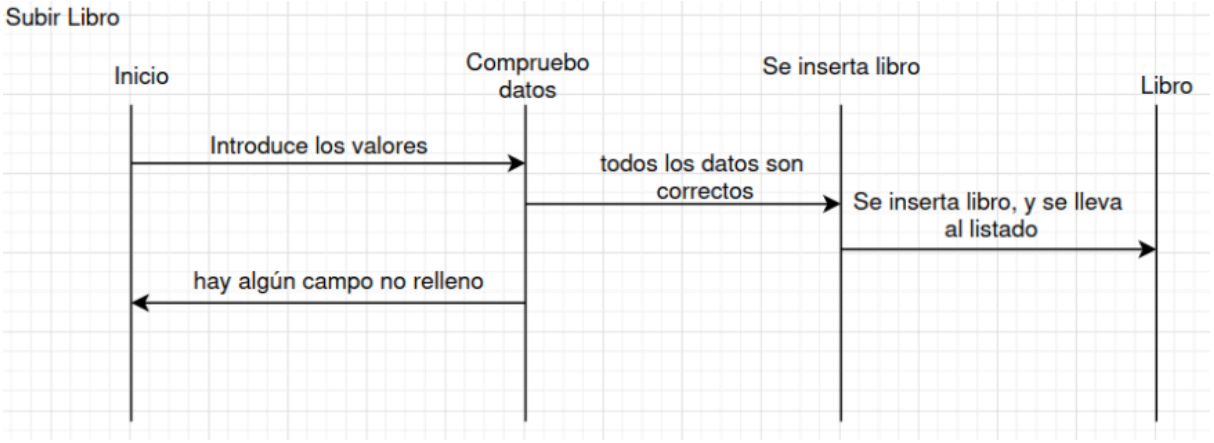
- Registro secuencia



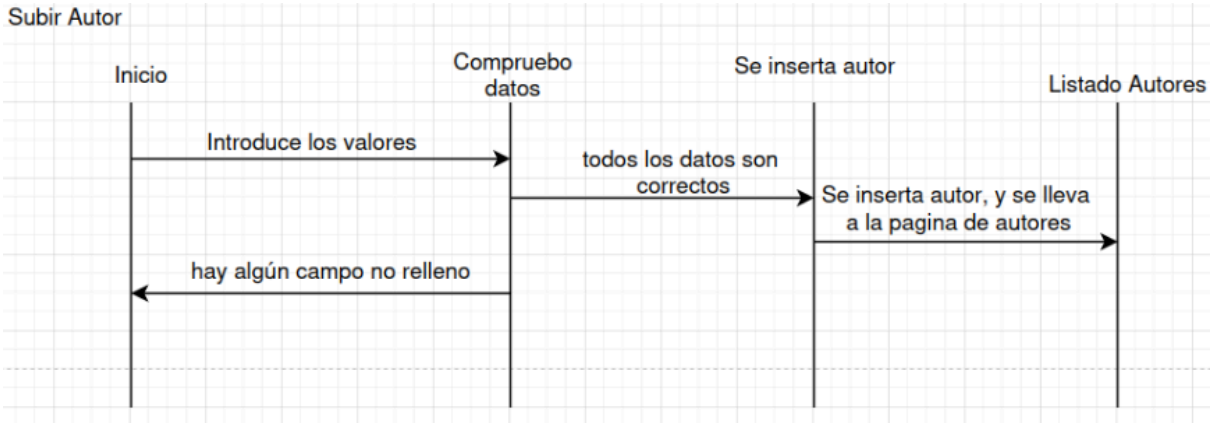
- Login secuencia



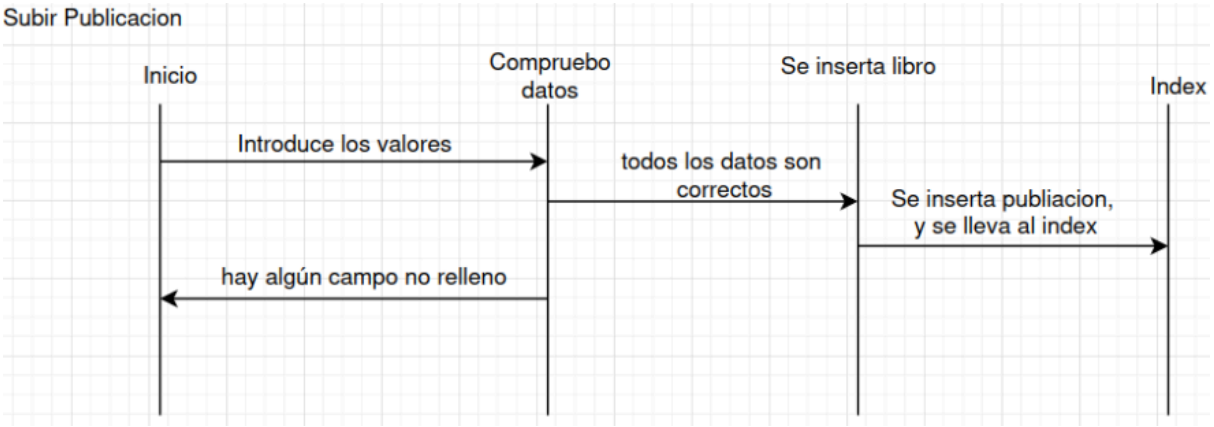
- Libros Añadir secuencia



- Autores añadir secuencia



- Publicaciones



4. Patrón de vista.

Para la vista he usado el patrón modelo vista controlador (MVC), la estructura de carpetas es:

├── app.js

├── db.js

├── package.json

├── package-lock.json

├── .gitignore

├── controllers

│ ├── autorController.js

```
│   ├── graciasController.js
│   ├── indexController.js
│   ├── librosController.js
│   ├── loginController.js
│   ├── perfilController.js
│   ├── rankingController.js
│   ├── registerController.js
│   ├── sobreNosotrosController.js
│   ├── subeTuLibroController.js
│   └── laplumadigital
├── middleware
│   └── verificarJWT.js
├── node_modules
├── public
│   ├── css
│   │   └── styles.css
│   └── images
│       ├── antonio.jpg
│       ├── autoresCasoUso.png
│       ├── autorSecuencia.png
│       ├── banner.jpg
│       ├── Clases.png
│       ├── EntidadRelacion.png
│       ├── fondo.png
│       ├── librosCasoUso.png
```

```
│       ├── libroSecuencia.png
│       ├── loginsecuencia.png
│       ├── loginyregistroCasoUso.png
│       ├── logo.png
│       ├── publicacionesCasoUso.png
│       ├── publicacionSecuencia.png
│       ├── rankingCasoUso.png
│       ├── registroSecuencia.png
│       └── subeTuLibroCasoUso.png
├── router
├── views
│   ├── autores
│   ├── gracias
│   ├── libros
│   ├── login
│   ├── miperfil
│   ├── ranking
│   ├── register
│   ├── sobreNosotros
│   ├── subeTuLibro
│   ├── templates
│   └── index.pug
```

5. Código:

Ejemplo de un pug sencillo:

```
extends ../templates/layout

block head

block content

 div.centro

   div.tamano

     h1 Registrate con nosotros

     form(action="/register", method="post")

       label(for="UserName") Username:

       input(type="text", name="UserName")

       br

       br

       label(for="Nombre") Nombre:

       input(type="text", name="Nombre")

       br

       br

       label(for="Descripcion") Descripcion:

       input(type="text", name="Descripcion")

       br

       br

       label(for="Email") Email:

       input(type="text", name="Email")

       br

       br
```

```
        label(for="Contraseña") Contraseña:
        input(type="password", name="Contraseña")
        br
        br
        label(for="Foto") Foto (URL):
        input(type="text", name="Foto")
        br
        br
        input.enlace-registrar(type="submit", value="Registrate!!")
        br
        br
        a.enlace-enviar(href='/login') ¿Ya tienes cuenta? Logueate aquí!!
```

Ejemplo sencillo de código del JWT:

```
const jwt = require('jsonwebtoken');

module.exports = (req, res, next) => {

    const rutasPublicas = ['/', '/register', '/login'];

    const token = req.cookies.token;

    res.locals.tipo = null

    res.locals.usuarioAutenticado = false;

    if (rutasPublicas.includes(req.path)) {

        if (token) {

            jwt.verify(token, process.env.JWT_SECRET, (err, decoded) => {

                if (!err) {

                    req.user = decoded;
```

```
                    res.locals.usuarioAutenticado = true;
                    res.locals.tipo = decoded.tipo
                }
            });
        }
        return next();
    }
    if (!token) {
        return res.render('login/login');
    }
    jwt.verify(token, process.env.JWT_SECRET, (err, decoded) => {
        if (err) {
            return res.render('login/login');
        }
        req.user = decoded;
        res.locals.usuarioAutenticado = true;
        res.locals.tipo = decoded.tipo
        next();
    });
};
```

Este archivo es el middleware encargado de proteger las rutas privadas de la aplicación. Su función principal es verificar si el usuario está autenticado a través de un token JWT almacenado en las cookies.

En primer lugar, se definen unas rutas públicas (`/`, `/register` y `/login`) a las que se puede acceder sin necesidad de estar autenticado. Si un usuario intenta

acceder a una de estas rutas y tiene un token, el sistema lo verifica y, si es válido, guarda en `res.locals` información útil como el tipo de usuario y si está autenticado.

Esta información se puede utilizar luego en las vistas para mostrar contenido personalizado según el tipo de usuario. En el caso de que se trate de una ruta privada, el middleware comprueba si hay un token presente; si no lo hay o si no es válido, redirige automáticamente al usuario a la página de login.

En resumen, este archivo sirve para proteger las páginas privadas, asegurar que los usuarios estén correctamente identificados antes de acceder a ciertos contenidos, y facilitar el control de acceso según los roles definidos en la aplicación.

Ejemplo de controller:

```
const db = require('../db.js');

const moment = require('moment');



exports.ranking = (req, res) => {

   db.query(

        'SELECT RANK() OVER (ORDER BY COUNT(p.idLibro) DESC) as ranking, COUNT(*) as contador, l.Titulo, l.Foto, p.idLibro as idLibro FROM Publicaciones p JOIN Libros l ON p.idLibro = l.idLibro GROUP BY p.idLibro, l.Titulo, l.Foto ORDER BY COUNT(p.idLibro) DESC'

   , (err, rank)=> {

       if (err) {

           return res.send('Fallo a la hora del ranking' + err)

       }

       if (rank.length === 0) {

           return res.send('Fallo en la respuesta')
```

```
        }

        db.query(

                'SELECT RANK() OVER (ORDER BY COUNT(p.idAutor) DESC) as ranking,
COUNT(*) as contador, a.Nombre, a.Foto, p.idAutor as idAutor FROM Publicaciones
p JOIN Autores a ON a.idAutor = p.idAutor GROUP BY p.idAutor, a.Nombre, a.Foto
ORDER BY COUNT(p.idAutor) DESC',

            (err, rankA) => {

                if(err){

                    return res.send('Fallo a la hora del ranking autores' + err)

                }

                if(rankA.length === 0) {

                    return res.send('Fallo en la respuesta')

                }

                res.render('ranking/ranking', {rankings: rank, rankA: rankA})

            }

        )

    })

}
```

Este código se encarga de generar y mostrar los rankings tanto de libros como de autores dentro de la plataforma. Utiliza consultas SQL para contar cuántas publicaciones están asociadas a cada libro y autor, y con esa información establece un orden jerárquico utilizando la función RANK() de SQL.

En el caso de los libros, se cuentan cuántas veces han sido mencionados en las publicaciones, y se muestran junto a su título, foto y posición en el ranking. De manera similar, para los autores se realiza la misma operación, mostrando su nombre, imagen y puesto correspondiente. Ambos rankings se agrupan por el identificador del libro o del autor, lo que permite tener datos precisos y sin duplicados.

Una vez obtenida la información, los resultados se envían a la vista correspondiente para ser mostrados al usuario de forma clara y ordenada. Este controlador permite a los usuarios conocer de forma rápida qué libros y autores son los más comentados en la red social, fomentando así la interacción y el descubrimiento de nuevos contenidos populares.

6. Conclusiones

Este proyecto me ha tomado un buen montón de horas, sobre todo porque no tenía mucho conocimiento de la tecnología que usé.

Me tocó aprender cosas nuevas sobre la marcha, enfrentar algunos problemas que no esperaba y buscar la forma de solucionarlos. Pero la verdad es que ha sido muy gratificante. Ver cómo esa idea que tenía en la cabeza poco a poco fue tomando forma y al final se volvió realidad me ha dejado muy contento.

Además, siento que aprendí un montón durante todo el proceso, y eso hace que todo el esfuerzo haya valido la pena. Al final, no solo terminé con un proyecto listo, sino que también gané experiencia y confianza para lo que venga después.