Cecilia Llamazares López

i.es San Andrés del Rabanedo, Desarrollo de aplicaciones web

Descripción breve

[Dibujar su lector con un resumen de la participación. Normalmente es un breve resumen del documento.   
Cuando esté listo para agregar contenido, haga clic aquí y empiece a escribir.]

FillMein

Red social de diferentes temas

Tabla de contenido

[**1.** **Estudio del problema y análisis del sistema** 2](#_Toc135686942)

[**1.1** **Introducción** 2](#_Toc135686943)

[**1.2** **Finalidad** 2](#_Toc135686944)

[**1.3** **Requisitos** 2](#_Toc135686945)

[**2.** **Recursos** 2](#_Toc135686946)

[**2.1** **Recursos hardware** 2](#_Toc135686947)

[**2.1.1** **Recurso para el desarrollo: Lenovo IdeaPad 330** 2](#_Toc135686948)

[**2.2** **Recursos software** 3](#_Toc135686949)

[**2.2.1** **Visual Studio Code** 3](#_Toc135686950)

[**2.2.2** **Postman** 3](#_Toc135686951)

[**3.** **Planificación** 4](#_Toc135686952)

[**3.1** **Planificación temporal** 4](#_Toc135686953)

[**3.2** **Planificación económica** 4](#_Toc135686954)

[**4.** **Desarrollo y pruebas** 4](#_Toc135686955)

[**4.1** **Diseño** 4](#_Toc135686956)

[**4.2** **Implementación** 7](#_Toc135686957)

[**4.2.1** **API** 7](#_Toc135686958)

[**4.2.2** **Inicio de sesión y registro** 9](#_Toc135686959)

[**4.2.3** **Perfil** 10](#_Toc135686960)

[**4.2.4** **Página de inicio** 12](#_Toc135686961)

[**5.** **Conclusiones finales** 14](#_Toc135686962)

[**5.1** **Grado de cumplimiento de los requisitos fijados** 14](#_Toc135686963)

[**5.2** **Propuestas de mejora o ampliaciones futuras** 14](#_Toc135686964)

[**6.** **Guías** 14](#_Toc135686965)

[**6.1** **Guía de uso** 14](#_Toc135686966)

[**6.2** **Guía de instalación** 16](#_Toc135686967)

[**7.** **Referencias bibliográficas** 16](#_Toc135686968)

# **Estudio del problema y análisis del sistema**

## **Introducción**

Fill Me In es una red social en la que los usuarios podrán compartir todo tipo de momentos e información sobre un tema en específico.

## **Finalidad**

Este proyecto pretende ofrecer un aspecto informativo en cuanto a los diferentes usuarios de la aplicación y del mundo actual en caso de que una cuenta esté especializada en compartir información de este tipo.

## **Requisitos**

Los requisitos de la aplicación son los siguientes:

* **Cuentas profesionales**: Cuentas que se dedican única y exclusivamente a publicar contenido relacionado con el tema que se eligió a la hora de su creación. Ej: Agricultura.
* **Muro**: Cada cuenta tiene una zona llamada “muro”, donde la gente podrá poner varios comentarios de esta, puede ser a modo de sugerencia.
* **Valoraciones**: Las cuentas profesionales tienen un sistema de valoración en el que los usuarios podrán decidir si es una buena cuenta que cumple con su función o no, dependiendo de sus contenidos.

Las entradas de todos los usuarios también se podrán valorar, especificando mediante un botón si al usuario le ha gustado la publicación.

* **Suscripción a temas**: Los usuarios pueden suscribirse a diversos temas de interés y podrán buscar entradas acordes a estos temas elegidos, ya sea por novedades o por mejores valorados.
* **Filtros de búsqueda**: Los usuarios pueden buscar entradas por cuentas normales, cuentas profesionales, cuentas a las que siguen o por temas. También pueden buscar entradas de un tema en específico mejores valoradas, así como las últimas entradas de estos. También pueden buscar entradas de usuarios en concreto.

# **Recursos**

## **Recursos hardware**

### **Recurso para el desarrollo: Lenovo IdeaPad 330**

Este portátil es un portátil desarrollado por Lenovo. Es el portátil ideal para llevar a cabo tareas del día a día y no es difícil de utilizar.

Es el portátil que actualmente poseo y utilizo para tareas de programador. Ya vino de por sí con bastante espacio, 1 T, y es bastante barato ya que su precio no llegó a los 600 €.

Va bastante bien y rápido y su diseño me es agradable, aparte considero a Lenovo como una empresa de buena calidad.

## **Recursos software**

### **Visual Studio Code**

El software que utilizo principalmente para el proyecto es Visual Studio Code. Es un editor de código fuente que soporta de muchos lenguajes de programación, como Java, JavaScript, PHP, Python, etc.

**Ventajas**:

* Es un editor gratuito de código abierto.
* Soporta muchos lenguajes de programación.
* Es compatible con varias plataformas (Windows, macOS y Linux)
* Tiene muchas opciones de personalización (Extensiones y apariencia)

**Desventajas**:

* Interfaz de usuario confusa para nuevos usuarios.
* Bajo rendimiento en proyectos grandes.
* Requiere conocimientos técnicos para la personalización.

### **Postman**

Utilizo esta aplicación para mandar peticiones al servidor de api que estoy utilizando y ver las respuestas que me envía.

Me resulta una aplicación muy sencilla de utilizar para ver todas aquellas respuestas que recibas.

Estas respuestas pueden variar de formato, si el servicio lo permite. Como el que yo estoy utilizando es parte HTML y parte JSON, lo utilizo mayormente para ver esa parte en JSON y poder leerlo más tarde. (Ver en la **Figura 4**)

Captura de pantalla de computadora

Descripción generada automáticamente

Figura 2.

La **Figura 2** es un ejemplo del enlace que estoy utilizando, y cómo se visualiza la respuesta en la **Figura 5** de manera organizada, así como las diferentes solicitudes en archivos de la **Figura 3**.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 4.

Figura 3.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente con confianza media

Figura 5.

# **Planificación**

## **Planificación temporal**

## **Planificación económica**

# **Desarrollo y pruebas**

## **Diseño**

La base de datos de la página web consta de tres tablas: Usuarios, muro y entradas. Se va a centrar principalmente en la de usuarios, ya que de esta se puede sacar todo lo demás.

No he creado claves foráneas puesto que lo vi más sencillo haciendo simplemente referencia al nombre del usuario en las otras dos tablas.

En la **Figura 7** se pueden ver los diferentes campos de cada una de las tablas.

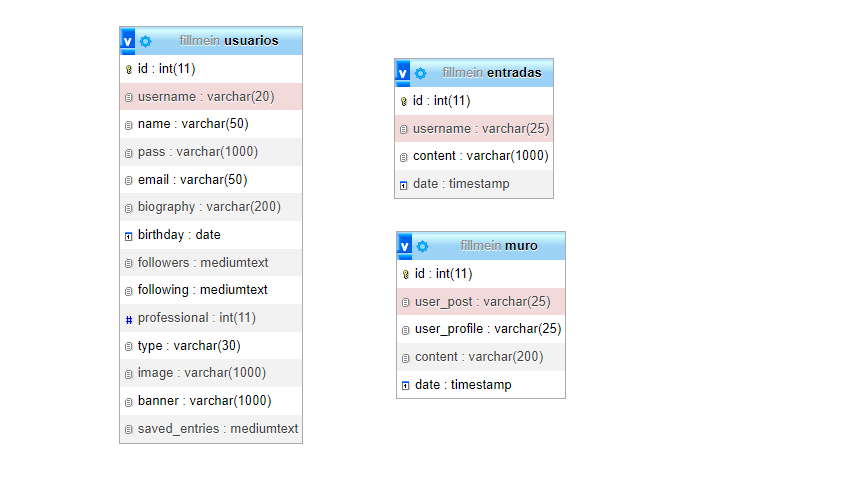


Figura 6.

La paleta de colores de la página web se basa en tonos morados, grises, blancos y amarillo. Predomina el color morado en todas las secciones de esta, por lo que será el color representativo de la aplicación.



Logo de la aplicación
Figura 7.

En cuanto al icono de la aplicación, será algo muy sencillo, lo suficiente como para poder representar a la página web de manera adecuada con su tema de colores.

Es un cuadrado con la letra F en morado situada en el centro.

Figura 8.

En cuanto a los mockups, en las siguientes imágenes se puede ver el diseño principal de algunas de las pantallas:

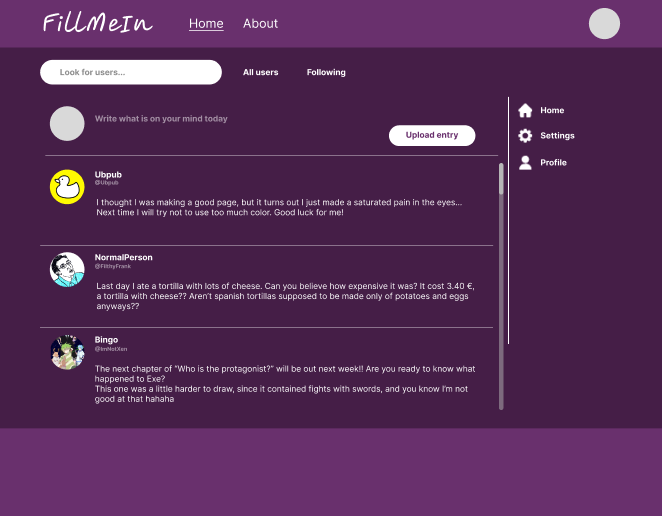
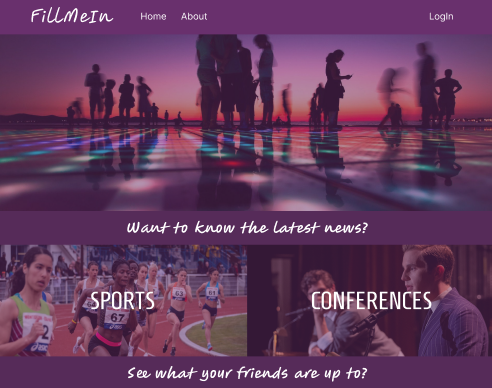


Figura 9. Figura 10.

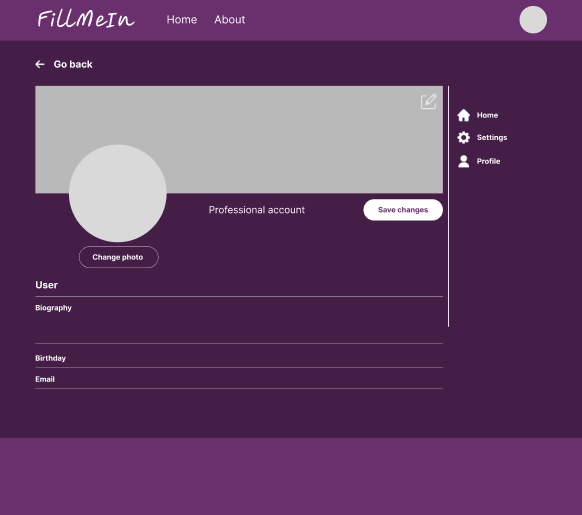
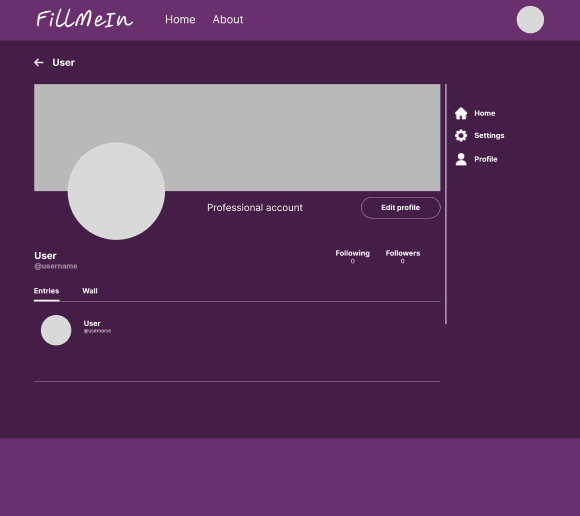


Figura 11. Figura 12.

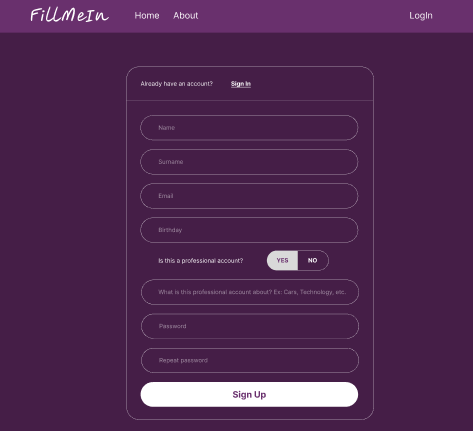
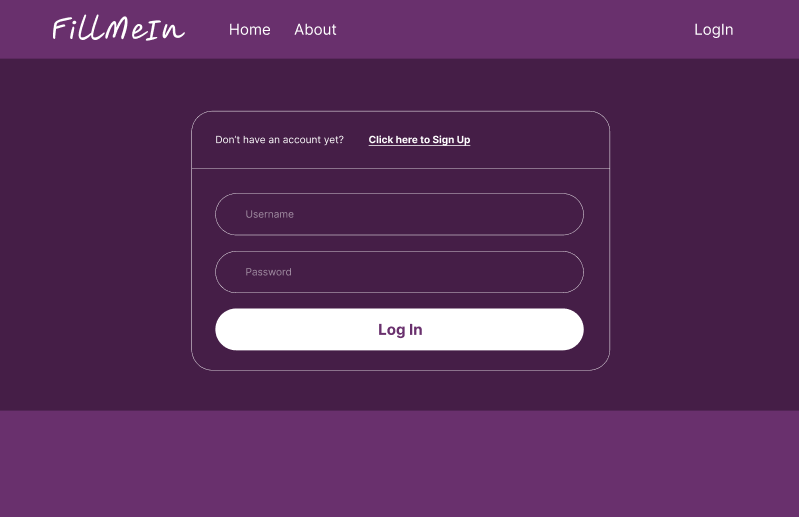


Figura 13. Figura 14.

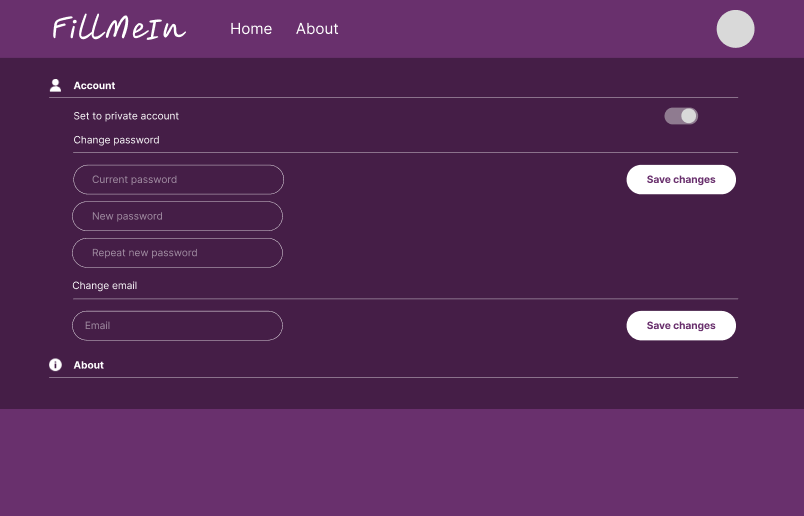


Figura 15.

## **Implementación**

### **API**

Creación de diferentes Apis en la parte de servidor para poder pasar los datos de la base de datos a la parte del cliente.

Las Apis son las siguientes:

* **Conexión**: Donde se inicia la conexión a la base de datos. Esta clase es la que se utilizará en las demás Apis para hacer consultas.
* **Usuarios**: Compuesto por dos comprobaciones: GET y POST. En la petición GET comprueba si se le ha pasado el nombre de usuario, la contraseña o el id en la URL. También genera un JWT para mantener iniciada la sesión del usuario. Esto se consigue con una “*key*” y un “*Payload*”:

****

Figura 16.

**Texto

Descripción generada automáticamente**

Figura 17.

En la **Figura 17** se genera una variable “*payload*” en la que agrega el nombre de usuario como valor ‘iss’. Seguido, crea un JWT con ayuda del JWT::encode, el cual cifra el contenido, en este caso el “*payload*”, “*key*” y el tipo de cifrado HS256.

Seguido, se añade este JWT al usuario obtenido en un nuevo campo llamado ‘*webToken*’.

En cuanto al POST, recibe unos parámetros del cuerpo de la petición de la parte JavaScript, comprueba que en la base de datos no exista un registro con el mismo nombre de usuario e inserta los datos.

* **Filtro de usuarios**: En el método GET busca a los usuarios cuyos nombres de usuario o tipo de cuenta profesional coincidan o tengan los caracteres que se han pasado en la URL.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 18.

En la **Figura 18** hace una comprobación, si ha recibido otro parámetro que se llame “*professional*” comprueba si su valor es “personal” o “profesional”. En cada caso añade a la consulta en la parte WHERE una comprobación para la columna “*professional*”. El 0 significa falso y el 1 verdadero. Esto se explicará más adelante.

El POST de este API se utiliza para editar los seguidores, las personas a las que sigue el usuario conectado y sus entradas guardadas. Llegan unos valores en el cuerpo y, mediante esos valores, realiza las siguientes comprobaciones:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 19.

* **Editar usuario**: Consta de dos comprobaciones: POST y DELETE. La parte POST actualiza el usuario con los datos modificados enviados en el cuerpo de la petición.

La parte DELETE elimina al usuario, recogiendo el id y la contraseña enviados en el cuerpo.

* **Entradas**: Las entradas están compuestas también por comprobaciones GET y POST como todas las demás. En el GET obtiene las entradas con el id o nombre de usuario que se ha pasado. En la sentencia lo ordena por “*date*”, que es la fecha de la publicación de las entradas, de manera descendente, para que las entradas más recientes aparezcan en la parte superior.

En el POST crea una variable donde almacena la fecha local actual.



Figura 20.

Inserta los valores obtenidos, en este caso el nombre de usuario, el contenido y la fecha generada, en la tabla de entradas.

* **Muro**: La parte GET de este API obtiene los registros del muro mediante el id, el usuario que publicó la opinión o al usuario al que se la han publicado, y lo ordena por fecha de publicación también descendiente. La parte POST hace lo mismo que en el API de las entradas, pero para la tabla del **muro**, en el que se inserta el nombre de usuario del usuario que la publicó, el nombre de usuario en el que la opinión ha sido publicada (hablo de su muro del perfil), el contenido de la opinión y la fecha generada anteriormente.

### **Inicio de sesión y registro**

A la hora de implementar el inicio de sesión y el registro hago uso de los Apis creados anteriormente.

En la parte de **registro**, una vez pulsado el botón de registrarse (hecho con un event listener), se comprobarán los valores uno a uno, ya sea mediante patrones o si están vacíos. Si hay algún campo que esté mal se generarán varias alertas indicándolo.

Imagen que contiene Texto

Descripción generada automáticamente

Figura 21.

En la **Figura 21** se pueden ver los diferentes patrones que he utilizado para la fecha de cumpleaños y el correo electrónico. Se comprueba si el valor recogido es válido mediante ese patrón con el siguiente código:



Figura 23.

Una vez comprobados los campos y que todos sean válidos, se generará un JSON con la siguiente estructura:

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 24.

Seguido se hará un **fetch** o una petición al API de usuarios para insertar al usuario, pasando este JSON generado en el cuerpo de la petición.

Texto

Descripción generada automáticamente

Figura 25.

Importante que se indique en el código de error 409 que el nombre de usuario ya está escogido, por lo que el usuario tendrá que elegir otro nombre.

En la parte del **login** comprueba las credenciales del usuario, tanto el nombre de usuario como la contraseña. Si todo va bien almacena el “*webToken*” generado en la parte del servidor, el nombre del usuario, el id y su enlace de imagen si tiene una en el almacenamiento local del navegador de la siguiente manera:

Texto

Descripción generada automáticamente

Figura 26.

En caso de que algo vaya mal, se le notificará al usuario que hay algo que ha fallado y que compruebe sus credenciales.

Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 27.

### **Perfil**

Aquí explicaré el tema de la vista del perfil y la edición de este en el lado del cliente.

**PRIMERO LA VISTA DE PERFIL NORMAL**

El perfil tiene tres secciones diferentes de entradas: Las entradas publicadas, el mural en el que los usuarios pueden publicar cosas sobre el dueño del perfil y las entradas guardadas.

Voy controlando la vista de cada una de estas secciones con un eventListener click y ocultando los demás contenedores dependiendo en qué opción pulse.

Para obtener los datos del usuario y mostrarlos en el perfil primero compruebo que exista un “*webToken*” con el que trabajar. Comprueba mediante el almacenamiento local si tiene foto de perfil o no, en caso de que tenga recupera el enlace mediante el almacenamiento local, ya que lo guardé con anterioridad.

En cuanto a la otra información, es necesario hacer una petición al servidor para obtener los datos.

Texto

Descripción generada automáticamente

Figura 28.

Con el fetch de la **Figura 28**, pasando el nombre de usuario en el enlace, se obtienen todos los datos necesarios. Ahora podremos pintarlos en la página seleccionándolos y cambiando su valor como en la siguiente imagen:

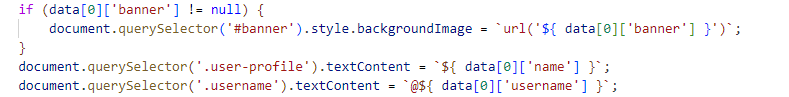


Figura 29.

Para **obtener las entradas** de este usuario y mostrarlas en su sección de entradas, hace falta otra petición, pero esta vez al API de las entradas, pasando en la URL el nombre del usuario.

Por cada entrada obtenida va creando un elemento en el HTML con la información necesaria obtenida.

En cuanto a la **visión del mural**, se hace de la misma manera que con las entradas. Haciendo un fetch al API del mural, pasando su nombre de usuario en la URL. Por cada opinión obtenida se crea una opinión en la parte HTML haciendo un fetch al usuario que la publicó, que se obtiene del nombre de usuario (*user\_post*) de la opinión. Ahora se pueden rellenar la opinión con los datos obtenidos.

El usuario puede subir una opinión dentro de su mismo perfil, gracias a una llamada al API del muro, pasando la opinión a modo de JSON en el cuerpo de la petición.

Para **ver las entradas guardadas** del usuario primero hace una petición al API de usuarios pasando el nombre de usuario en la URL. Dentro de esta, si todo ha ido bien, se comprueba que tiene entradas guardadas y hace un fetch a estas pasando sus id, para después hacer otra petición para obtener toda la información del usuario que ha publicado la entrada y poder pintarla de la misma manera que antes en la página.

**SEGUNDO LA EDICIÓN DE PERFIL**

Utiliza la misma comprobación con los mismos patrones que en el registro, pero esta vez los campos no se pueden guardar si no se ha introducido la contraseña dos veces.

A la hora de guardar la biografía, como puede haber caracteres especiales como (‘, & ó `), llamo a una función que filtra esos símbolos por código HTML. En la **Figura 30** se puede ver la función utilizada.

Texto

Descripción generada automáticamente

Figura 30.

Tengo otra función que hace lo mismo, pero al revés, por si acaso sea necesaria. Cuando se pulsa el botón de guardar campos y está todo correcto, hará un **fetch** al API de editar, donde pasará un JSON con los campos editados. Si sale algo mal mostrará un error en pantalla. El JSON tiene la estructura que se muestra en la **Figura 31**.

Texto

Descripción generada automáticamente

Figura 31.

### **Página de inicio**

En la página de inicio se muestran todas las entradas y se hace uso de filtros para buscar usuarios o entradas, dependiendo de si la persona sigue al usuario o no.

**INICIO Y OBTENCIÓN DE ENTRADAS**

Obtengo las entradas de la misma manera que en el perfil del usuario y las añado al HTML, con la única diferencia de que, al pulsar en la imagen, nombre o nombre de usuario del usuario en concreto, me lleva a su página de perfil. Esto se consigue dentro del fetch del usuario de la entrada, donde se hará lo de la **Figura 32** para cada elemento mencionado anteriormente.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Figura 32.

La primera condición comprueba si es el usuario que ha iniciado sesión, pata llevarle a su página de perfil con la opción de editar. La otra condición los lleva a otro **js** donde tendrán la opción de seguir el perfil.

En cada entrada hay un icono de guardado que cambia de clase al ser pulsado.

**FILTROS EN EL CAMPO DE TEXTO**

Cuando el campo de texto del menú superior es modificado, irán apareciendo los usuarios cuyo nombre coincida o contenga el valor obtenido de este. Si el campo de texto está vacío, obtiene todas las entradas, pero si tiene algo escrito, hará lo mencionado anteriormente.

Para obtener a los usuarios con el valor del campo, se hace un **fetch** al API de filtrado de usuarios, donde se creará un contenedor para cada usuario, con su foto, su nombre y su nombre de usuario. Si el usuario es una cuenta profesional, también aparecerá el tipo de cuenta que és.

En cuanto a los filtros de mostrar todos los usuarios o sólo a los que sigue, se selecciona el que tenga la clase ***selected*** y, en caso de que sea sólo el de los seguidos, se hace un **fetch** al API de usuarios pasando el nombre de usuario del usuario que ha iniciado sesión. Obtenidos los datos se hace una petición a las entradas y por cada entrada obtenida otra petición a los usuarios, con el nombre del usuario de la entrada en la URL. Aquí va comprobando si el id del usuario de la entrada aparece en la lista de usuarios a los que sigue el usuario con la sesión iniciada. En caso de que aparezca la entrada se muestra. Esto se puede ver en la **Figura 33**.



Figura 33.

**Data\_user** es el usuario con la sesión iniciada, y **data\_user­\_found** es el usuario que ha subido la entrada.

**SUBIDA DE ENTRADAS Y GUARDADO DE ESTAS**

Pulsado el botón de Subir entrada, genera un JSON de la entrada y con un **fetch** a las entradas la sube.

Para guardar una entrada, una vez pulsado el icono de guardar, realizo una petición al API de usuarios con el nombre de usuario. En la **Figura 34** añado las entradas guardadas del usuario a un array, y en la **Figura 35** compruebo si la entrada va a ser guardada o no, para añadirla al array o para eliminarla.

**Texto

Descripción generada automáticamente**

Figura 34.

Texto

Descripción generada automáticamente con confianza media

Figura 35.

Finalmente hace un **fetch** al API de filtros de usuario y añade la entrada pasada como JSON.

### **Perfil de otros usuarios.**

Lo único que resaltar aquí es la parte de seguir al usuario, todo lo demás se hace exactamente igual que en el perfil del usuario con la sesión iniciada.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 36.

En la **Figura 36**, lo que hace es ir cambiar la clase del botón cuando se pulsa. Dependiendo de su clase, lo elimina de seguidos o lo añade. En cuanto los parámetros de la función **getUserInfo**, se pasa el nombre del usuario, una variable booleana que sirve para determinar si se añade el usuario o no, que se explicará más adelante, el segundo es otro booleano para determinar si se le sigue actualmente o no, y el tercero otro booleano para determinar si se elimina o no.

Si el usuario que ha iniciado sesión no le sigue, hago lo mismo que hice con las entradas guardadas. Dentro del **fetch** al usuario creo un array para los seguidores de este usuario y compruebo con el último parámetro pasado en la función si se quiere dejar de seguir o no. Con un **fetch** lo paso al servidor y lo actualiza en la base de datos.

En caso de que el tercer parámetro fuese false, que significa que no le sigue, llamaría a otra función que haría lo mismo pasando la variable ***add***, pero esta vez para los seguidos, que solo se ejecuta si se quiere añadir. Si esa variable es falsa lo único que hace es cambiar la clase al botón de seguir para que aparezca como que ya le sigue, en caso de que lo encuentre como en la **Figura 37** aparece.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 37.

# **Conclusiones finales**

## **Grado de cumplimiento de los requisitos fijados**

## **Propuestas de mejora o ampliaciones futuras**

# **Guías**

## **Guía de uso**

Nada más entrar en la página, el usuario podrá acceder desde el menú superior al inicio en la pestaña “**Home**”, a información de la página, en la pestaña “**About**” e iniciar sesión en “**LogIn**” en la parte derecha del menú superior.



Figura 40.

En la pantalla de Inicio de sesión hay un enlace en la parte superior que lleva a la pantalla de registro. En esta pantalla hay un campo especial, que aparece si se ha seleccionado la opción de **YES** en el campo de “Cuenta profesional”. Estas cuentas se dedicas especialmente a subir contenido relacionado con el tipo de cuenta que han escrito en este campo.

Interfaz de usuario gráfica, Gráfico

Descripción generada automáticamente

Figura 41.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Figura 42.

Una vez creado el perfil o iniciado sesión, se podrá acceder a la ventana del perfil pulsando sobre la imagen gris o imagen de perfil del menú superior en la parte derecha. También se puede acceder desde la página de inicio, en el menú lateral derecho, en el apartado de “Perfil”. Así mismo, podrá cerrar la sesión pulsando en el icono de al lado.

Imagen borrosa de una persona

Descripción generada automáticamente con confianza baja

Figura 43.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 44.

El usuario podrá editar su perfil pulsando en el botón de “Editar perfil” que aparece en la ventana de este. En esta pestaña, el usuario podrá modificar el nombre, la biografía, la fecha de nacimiento y el email, escribiendo su contraseña dos veces para poder guardar los cambios.

Texto

Descripción generada automáticamente

Figura 45.

En la página de inicio, si el usuario ha iniciado sesión, podrá crear entradas con el campo de texto que aparece por encima de todas las entradas. Una vez escrito lo que se quiere publicar, sólo deberá pulsar el botón de “Subir entrada” para publicarla.

Imagen que contiene Forma

Descripción generada automáticamente

Figura 46.

Si se quiere ver el perfil de un usuario, sólo se deberá pulsar en su nombre o en su foto, y le llevará a la página de su perfil, donde podrán seguirle y mirar las entradas que ha publicado.

Diagrama

Descripción generada automáticamente con confianza media

Figura 47.

Naranjas y manzanas

Descripción generada automáticamente con confianza media

Figura 48.

## **Guía de instalación**

# **Referencias bibliográficas**