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** 3](#_Toc135219084)

[**1.1** **Introducción** 3](#_Toc135219085)

[**1.2** **Finalidad** 3](#_Toc135219086)

[**1.3** **Requisitos** 3](#_Toc135219087)

[**2.** **Recursos** 3](#_Toc135219088)

[**2.1** **Recursos hardware** 3](#_Toc135219089)

[**2.1.1** **Recurso para el desarrollo: Lenovo IdeaPad 330** 3](#_Toc135219090)

[**2.2** **Recursos software** 4](#_Toc135219091)

[**2.2.1** **Visual Studio Code** 4](#_Toc135219092)

[**2.2.2** **Postman** 4](#_Toc135219093)

[**3.** **Planificación** 5](#_Toc135219094)

[**3.1** **Planificación temporal** 5](#_Toc135219095)

[**3.2** **Planificación económica** 5](#_Toc135219096)

[**4.** **Desarrollo y pruebas** 5](#_Toc135219097)

[**4.1** **Fase 1: Documentación** 5](#_Toc135219098)

[**4.2** **Fase 2: Realización de esquemas, diseños y mockup** 5](#_Toc135219099)

[**4.3** **Fase 3: Landing Page, conexión, usuario, inicio de sesión y registro** 5](#_Toc135219100)

[**4.3.1** **Landing Page** 5](#_Toc135219101)

[**4.3.2** **Conexión** 5](#_Toc135219102)

[**4.3.3 Usuario** 6](#_Toc135219103)

[**4.3.4** **Inicio de sesión** 8](#_Toc135219104)

[**4.3.5** **Registro** 10](#_Toc135219105)

[**4.4** **Fase 4: Entradas, página de inicio, perfil, editar php y edición de perfil** 13](#_Toc135219106)

[**4.4.1** **Entradas** 13](#_Toc135219107)

[**4.4.2** **Página de inicio** 13](#_Toc135219108)

[**4.4.3** **Perfil** 15](#_Toc135219109)

[**4.4.4** **Editar PHP** 17](#_Toc135219110)

[**4.4.5** **Edición de perfil** 18](#_Toc135219111)

[**4.5** **Fase 5: Página de perfiles y filtros** 19](#_Toc135219112)

[**4.5.1** **Página de perfiles** 19](#_Toc135219113)

[**4.5.2** **Filtros** 19](#_Toc135219114)

[**5** **Conclusiones finales** 19](#_Toc135219115)

[**5.4** **Grado de cumplimiento de los requisitos fijados** 19](#_Toc135219116)

[**5.5** **Propuestas de mejora o ampliaciones futuras** 19](#_Toc135219117)

[**6** **Guías** 19](#_Toc135219118)

[**6.4** **Guía de uso** 19](#_Toc135219119)

[**6.5** **Guía de instalación** 19](#_Toc135219120)

[**7** **Referencias bibliográficas** 19](#_Toc135219121)

# **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**

## **Fase 1: Documentación**

## **Fase 2: Realización de esquemas, diseños y mockup**

## **Fase 3: Landing Page, conexión, usuario, inicio de sesión y registro**

### **Landing Page**

### **Conexión**

Creamos una conexión en un fichero PHP para poder utilizarla en las peticiones tanto de usuarios como de entradas.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 6.

Se declaran las variables necesarias para acceder a la base de datos como el nombre de esta, el usuario y la contraseña. En el constructor se inicializa la conexión.

### **4.3.3 Usuario**

Para poder utilizar la página de inicio de sesión y de registro es necesario crear un fichero .php donde procesar las peticiones. En el caso del proyecto, el archivo se llama usuario.php.

El tema de los usuarios, concretamente en el inicio de sesión, utiliza un JWT, un “*Web Token*” necesario para comprobar que el que realiza las peticiones es el usuario que ha iniciado sesión y no otro.

Imagen que contiene Logotipo

Descripción generada automáticamente

Figura 7.

La variable **$key** que se ve en la **Figura 7** es la variable que se va a utilizar a la hora de crear un JWT para el usuario.

En la **Figura 8** de abajo se encuentra la comprobación para la parte **POST** de un usuario. Esta parte se utiliza principalmente en el inicio de sesión. Primero genera una consulta donde selecciona todos los usuarios. A continuación, comprueba si se le han pasado diferentes campos en la **url** para añadirlos a la consulta. En la parte de “*hashPass*” cifra la contraseña recibida para añadirla a la consulta, ya que se almacenan cifradas en la base de datos.

Finalmente ejecuta esta consulta y la almacena en la variable **$result.**

Texto

Descripción generada automáticamente con confianza media

Figura 8.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 9.

La continuación del código en la **Figura 9** comprueba si se ha obtenido algún registro que coincidiese con los valores obtenidos de la **url**, en caso de no encontrar ninguno devuelve el código de error **404 Not Found**. Si encuentra algo obtiene los usuarios devueltos en por la consulta, y con la variable **$payload** genera un “*Token*” con el nombre del usuario.

A continuación, cifra dentro de **$jwt** los valores generados anteriormente en **$payload** y **$key**, para almacenarlo dentro del usuario obtenido en un nuevo campo llamado “***Web Token***”.

En cuanto a la parte **POST** de los usuarios, la cual se utiliza a la hora de registrarse, comienza de la misma manera, solo que en este caso comprueba si se ha recibido un **json** en la cabecera de la petición. Obtiene el usuario del **json** recibido y cifra la contraseña con ayuda de **hash**.

A continuación, crea una consulta para obtener todos los usuarios con el mismo nombre de usuario, ya que en la base de datos el nombre de usuario no se debe repetir.

Texto

Descripción generada automáticamente

Figura 10.

Como se ve en la **Figura 11**, si no se ha obtenido ningún usuario con el mismo nombre de usuario procede a hacer la consulta de inserción con los datos obtenidos del **json**.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 11.

### **Inicio de sesión**

Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 12.

La pantalla de inicio de sesión consiste en un recuadro con dos campos a rellenar: el nombre de usuario y la contraseña. Una vez pulsado en el botón de “*Log In*” se comprueba si los campos están vacíos mediante una función en JavaScript.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 13.

Al método de la **Figura 13** se le pasa los valores de **nombre de usuario** y **contraseña**. Crea una variable para devolver al final del método, esta variable determinará si es válido intento de inicio de sesión o no. Lo único que se comprueba es que los campos no estén vacíos, y, en caso de que lo estén, se le cambiará el estilo a un borde de color rojo para remarcarlos.

A la hora de comprobar si las credenciales son correctas, la parte del cliente realiza una petición al servidor con un **fetch**.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura 14.

Primero comprueba si lo que nos ha devuelto la comprobación de campos es válido o no. Genera una **url** con el nombre de usuario y la contraseña para que sean comprobados en el servidor.

Texto

Descripción generada automáticamente

Figura 15.

En la **Figura 15** se puede ver cómo comprueba si ha devuelto algún dato, que solo se ejecuta cuando la petición devuelve un **200 OK**. Almacena los datos obtenidos en el **Almacenamiento Local** del navegador y redirige al usuario a la ventana inicial.

### **Registro**

Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 16.

Es un formulario un cuanto extenso al cual se accede desde la parte superior del inicio de sesión, un enlace que lleva al usuario a la pestaña de registro.

La parte de selección de la cuenta profesional consiste en dos botones. Al seleccionar **YES** aparece el campo de tipo de cuenta. Cuando selecciona **NO**, ese campo desaparece. Se puede ver el código en la **Figura 17**.

En cuanto a los campos de la contraseña, los iconos de la derecha permiten mostrar u ocultar la contraseña con cada click en este mediante el código de la **Figura 18**. Lo único que hace es cambiar el tipo de input según el icono que se haya seleccionado.

Texto

Descripción generada automáticamente

Figura 17.

Texto

Descripción generada automáticamente con confianza baja

Figura 18.

Para comprobar los campos utiliza el mismo método que en la parte de inicio de sesión, a excepción de que esta vez hay muchos más campos. Para los campos de correo electrónico y fecha de nacimiento se hace uso de patrones, y en caso de que no coincidan con estos, una alerta aparecerá debajo del campo que está mal. Se puede ver en la **Figura 19** y **Figura 20**.

Texto

Descripción generada automáticamente con confianza media

Figura 19.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Figura 20.

Una vez pulsado el botón de registro se llama a un método que genera un JSON con los valores rellenados del formulario. Este JSON se pasa en el **fetch** en la parte del **body**.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura 21.

Texto

Descripción generada automáticamente

Figura 22.

En la **Figura 22** se ve que hace uso del ***JSON.strigify(user)*** para pasarlo como una cadena. En la parte del servidor Coge esa cadena y la pasa a **JSON**.

Si todo va bien y devuelve un **200** el usuario será redirigido a la pantalla de inicio con la cuenta creada correctamente.

## **Fase 4: Entradas, página de inicio, perfil, editar php y edición de perfil**

### **Entradas**

Con un fichero PHP parecido al de los usuarios se obtienen e insertan las entradas.

En la **Figura 23** se puede ver que tiene la misma estructura que el **GET** de los usuarios. En este caso ordena las entradas mediante id para que aparezcan de más recientes a más antiguas en la página de inicio.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 23.

En la petición **POST** se realiza las mismas comprobaciones que en los usuarios. Finalmente se inserta con la sentencia SQL creada con los valores obtenidos.

Texto

Descripción generada automáticamente con confianza media

Figura 24.

### **Página de inicio**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteFigura 25.

Esta página está formada por un menú superior, con un campo de búsqueda y dos filtros. En la parte inferior hay dos secciones: Una para las entradas y otro menú lateral.

En la parte de las entradas, si el usuario no ha iniciado sesión aparecerá un mensaje que indique que debe iniciar sesión para poder publicar entradas. Como se ve en la **Figura 24**, cuando el usuario tiene la sesión iniciada sí se le da permiso para escribir entradas.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Figura 26.

En la parte de JavaScript, primero selecciona el contenedor donde aparecen las entradas de los usuarios para tenerlo como variable global en todo el fichero .js.

Primero crea un evento para el botón de “**Publicar entrada**” y a continuación hace un **fetch** a todas las entradas de la base de datos.

En la **Figura 27**, dentro del evento del botón se comprueba si el campo del ***TextArea*** está vacío o no. (El texto a publicar en la entrada). Con esto genera una entrada como **JSON** y la crea con un fetch pasándola en su cuerpo. Se puede ver el fetch en la **Figura 28**.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 27.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura 28.

### **Perfil**

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Figura 29.

El perfil está compuesto por un “*banner*”, un botón de **Editar perfil**, información de la cuenta como nombre, nombre de usuario, seguidos y seguidores, y entradas y muro.

Tiene la misma estructura que la página de inicio, con la parte derecha y el menú de la izquierda.

Para mostrar la información del usuario actual en pantalla primero se comprueba que el usuario tenga la sesión iniciada gracias a su “***Web Token***” en el almacenamiento local.

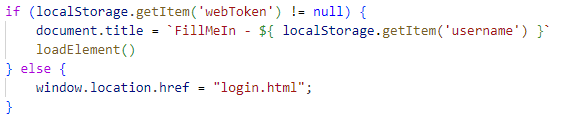


Figura 30.

Con la información del Almacenamiento Local cambia el título de la pestaña para poner el nombre del usuario y carga los demás elementos.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 31.

En la **Figura 32** se comprueba si el usuario tiene foto de perfil para poder ponerla en la página. Esta imagen se almacena también en el almacenamiento local, ya que va a ser uno de los campos que más se utilicen.

A continuación, obtiene los valores del usuario con el usuario sacado del Almacenamiento Local. Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 32.

Este fetch pasa en la **url** el nombre del usuario conectado y devuelve un **json** con los datos de este. Si la respuesta devuelve un **200 OK** procede a rellenar los datos de la página.

Texto

Descripción generada automáticamente

Figura 33.

Seguimos con la parte de carga de elementos, en la que se añade un evento **onClick** al botón de **Editar Perfil**. Este lleva al usuario a la página de edición de perfil.

En la parte de abajo se puede ver dos secciones: Entradas y muro.

En la parte de entradas aparecerán todas las entradas que el usuario ha publicado, y en la parte del Muro, se verán todos los comentarios que el usuario ha recibido en su perfil.

Se obtienen estas entradas con el mismo **fetch** en la parte de la página principal, a diferencia de que ahora le pasamos el nombre del usuario del perfil.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 34.

Con cada **item** obtenido hace un **fetch** al usuario pasando el contenido de estas entradas y crea el elemento HTML para pintarlo en la página, con la misma estructura que en la página principal.

Texto

Descripción generada automáticamente

Figura 35.

### **Editar PHP**

Esta es la parte API que hace posible la actualización del perfil en su edición.

Texto

Descripción generada automáticamente

Figura 36.

Después de todas sus comprobaciones como en los casos anteriores, en la parte **POST** se cifra la contraseña y comprueba si el nombre del usuario y la contraseña coinciden. En caso de que sí exista, genera una sentencia **UPDATE** que actualiza todos los campos del formulario con los valores que se han pasado.

En cuanto a la parte **DELETE**, como se puede ver en la **Figura 37**, se obtienen los valores como si fuese un **GET**, se cifra la contraseña pero se genera una consulta de DELETE con el id del usuario y la contraseña.

Texto

Descripción generada automáticamente

Figura 37.

### **Edición de perfil**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 38.

Es un formulario para cambiar los campos. Estos valores se obtienen del mismo modo que se obtuvieron en la parte del perfil.

Para guardar estos cambios es necesario escribir la contraseña actual dos veces, en caso de no hacerlo, los cambios no se guardarán, o más bien, no dejará hacerlo. Los marcará en rojo aquellos campos que estén mal, a veces incluso con un mensaje con el por qué ese campo está incorrecto.

Patrón de fondo

Descripción generada automáticamente con confianza media

Figura 39.

## **Fase 5: Página de perfiles y filtros**

### **Página de perfiles**

### **Filtros**

# **Conclusiones finales**

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

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

# **Guías**

## **Guía de uso**

## **Guía de instalación**

# **Referencias bibliográficas**