

CFGS DESARROLLO DE APLICACIONES WEB
Proyecto Final de Ciclo

SOCIAL BOOK



Autor: Julio César Novelli

Tutor: Javier Atoche Ortega

Fecha de entrega:

Convocatoria: 1S 2020/2021

Enlace hosting: <https://rinconesdemallorca.com/>

INDICE DE CONTENIDO

1.-INTRODUCCIÓN	7
1.1.-MOTIVACIÓN	7
1.2.-ABSTRACT	7
1.3.-OBJETIVOS PROPUESTOS.....	8
2.-METODOLOGÍA UTILIZADA.....	9
3.-TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS EN EL PROYECTO.....	9
IDE.....	9
Control de Versiones	9
Plataforma de desarrollo colaborativo	10
Base de Datos.....	10
Intercambio de datos.....	10
Maquetación de la Web	11
Lenguajes de Programación	11
Framework.....	11
Iconos.....	12
Diseño de esquemas y diagramas	12
Diseño de bocetos de la interfaz	12
4.-ESTIMACIÓN DE RECURSOS Y PLANIFICACIÓN	12
Análisis	12
Diseño	12
Codificación y pruebas.....	13
5.-DESARROLLO DEL PROYECTO	13
5.1.-ANÁLISIS	13
Requisitos funcionales.....	13
Requisitos no funcionales.....	14
Esquema Entidad-Relación.....	15
Modelo Relacional	17
5.2.-DISEÑO.....	18
Diseño de la interfaz de usuario	19
Creación de la Base de Datos.....	22
Composer	22
Instalación de Laravel	23
Host Virtual	24
Configuración de la Base de Datos en Laravel	26

Modelos.....	27
Configuración de las relaciones entre modelos.....	28
Login y registro de usuario.....	29
Controladores.....	30
UserController.php.....	30
PostController.php.....	33
LikeController.php.....	33
CommentController.php.....	35
FollowerController.php.....	35
HomeController.php.....	35
Middleware.....	35
Vistas.....	35
Rutas.....	37
Helpers.....	38
Scripts.....	43
Maquetado del proyecto.....	44
6.-DESPLIEGUE Y PRUEBAS.....	44
6.1.-PLAN DE PRUEBA.....	44
7.-CONCLUSIONES.....	49
7.1.-OBJETIVOS ALCANZADOS.....	49
7.2.-CONCLUSIONES DEL TRABAJO.....	49
7.3.-VÍAS FUTURAS.....	50
8.-GLOSARIO.....	51
9.-BIBLIOGRAFÍA.....	52
Libros.....	52
Páginas web.....	52
10.-ANEXOS.....	54
10.1.-MANUAL DE INSTALACIÓN.....	54
Instalación en entorno local.....	54
Despliegue en un hosting.....	68
10.2.-MANUAL DE USUARIO.....	75
Registro de usuario.....	75
Acceso a la plataforma.....	76
Pantalla de bienvenida.....	77
Menú del usuario.....	78
Modificar datos del usuario.....	78

Crear una publicación	79
Pantalla del detalle de la publicación	80
Modificación de publicación.....	82
Eliminar publicación.....	82
Comentar publicación	82
Like.....	82
Dislike	82
Buscar usuarios	83
Seguir una cuenta.....	84
Dejar de seguir una cuenta	84

INDICE DE FIGURAS

Figura 1: Diagrama de Gantt. Fuente: Elaboración propia.	13
Figura 2: Diagrama de Gantt. Fuente: Elaboración propia.	13
Figura 3: Esquema Entidad-Relación. Fuente: Elaboración propia.	15
Figura 4: Diagrama del Modelo Relacional. Fuente: Elaboración propia.	17
Figura 5: Diagrama de clases. Fuente: Elaboración propia.	18
Figura 6: Diagrama de Casos de Uso. Fuente: Elaboración propia.....	18
Figura 7: Vista de Login. Fuente: Elaboración propia.	19
Figura 8: Vista de Registro de usuario. Fuente: Elaboración propia.	19
Figura 9: Vista del perfil de usuario. Fuente: Elaboración propia.....	20
Figura 10: Vista de listado de publicaciones. Fuente: Elaboración propia.....	20
Figura 11: Vista de creación de publicación. Fuente: Elaboración propia.....	20
Figura 12: Vista de modificación de los datos del usuario. Fuente: Elaboración propia.	20
Figura 13: Vista de detalle de una publicación. Fuente: Elaboración propia.....	21
Figura 14: Vista de buscador de usuarios. Fuente: Elaboración propia.	21
Figura 16: Estructura de archivos generado por Laravel. Fuente: Elaboración propia.	23
Figura 17: Modificación del archivo hosts. Fuente: Elaboración propia.	24
Figura 18: Modificación del archivo httpd-vhosts.conf. Fuente: Elaboración propia.	25
Figura 19: Vista principal de Laravel. Fuente: Elaboración propia.	26
Figura 20: Configuración de la Base de Datos. Fuente: Elaboración propia.	27
Figura 21: Configuración de relaciones entre modelos. Fuente: Elaboración propia.....	28
Figura 22: Validación y creación de un registro de usuario. Fuente: Elaboración propia.	29
Figura 23: Función updtate. Fuente: Elaboración propia.	31
Figura 24: Configuración del Storage. Fuente: Elaboración propia.....	32

Figura 25: Detalle del método que registra un like. Fuente: Elaboración propia.	34
Figura 26: Archivo mymainpage.blade.php. Fuente: Elaboración propia.	36
Figura 27: Rutas del proyecto. Fuente: Elaboración propia.	37
Figura 28 Helper que consigue tiempo de publicación y registro de usuario. Fuente: Elaboración propia.	39
Figura 29: Provider. Fuente: Elaboración propia.	40
Figura 30: Arreglo providers. Fuente: Elaboración propia.	41
Figura 31: Arreglo alias. Fuente: Elaboración propia.	42
Figura 32: Script sistema de likes. Fuente: Elaboración propia.	43
Figura 33: Descarga de XAMPP. Fuente: Elaboración propia.	54
Figura 34: Instalación de XAMPP. Fuente: Elaboración propia.	55
Figura 35: Instalación de XAMPP. Setup. Fuente: Elaboración propia.	55
Figura 36: Instalación de XAMPP. Selección de componentes. Fuente: Elaboración propia.	56
Figura 37: Instalación de XAMPP. Ruta de instalación. Fuente: Elaboración propia.	56
Figura 38: Instalación de XAMPP. Últimos pasos de la configuración. Fuente: Elaboración propia.	57
Figura 39: Instalación de XAMPP. Instalación preparada. Fuente: Elaboración propia.	57
Figura 40: Instalación de XAMPP. Inicio de instalación. Fuente: Elaboración propia.	58
Figura 41: Instalación de XAMPP. Finalización de la instalación. Fuente: Elaboración propia.	58
Figura 42: Instalación de XAMPP. Panel de control. Fuente: Elaboración propia.	59
Figura 43: Puesta en marcha de los servicios. Fuente: Elaboración propia.	59
Figura 44: Servicios en marcha. Fuente: Elaboración propia.	60
Figura 45: Despliegue del proyecto en local. Fuente: Elaboración propia.	61
Figura 46: Estructura de archivos del proyecto. Fuente: Elaboración propia.	62
Figura 47: Acceso al gestor de base de datos. Fuente: Elaboración propia.	63
Figura 48: Importación de la base de datos. Fuente: Elaboración propia.	64
Figura 49: Base de datos importada. Fuente: Elaboración propia.	64
Figura 50: Archivo httpd-vhosts.conf. Fuente: Elaboración propia.	65
Figura 51: Archivo httpd-vhosts.conf modificado. Fuente: Elaboración propia.	66
Figura 52: Archivo hosts modificado. Fuente: Elaboración propia.	67
Figura 53: Acceso al proyecto. Fuente: Elaboración propia.	67
Figura 54: Preparación de archivos para subir al servidor. Fuente: Elaboración propia.	68
Figura 55: Administrador de archivos de cPanel. Fuente: Elaboración propia.	69
Figura 56: Carpeta public_html. Fuente: Elaboración propia.	69
Figura 57: Carpeta laravel. Fuente: Elaboración propia.	70
Figura 58: Base de Datos. Fuente: Elaboración propia.	70
Figura 59: Modificación index.php. Fuente: Elaboración propia.	71

Figura 60: Modificación app.php. Fuente: Elaboración propia.	72
Figura 61: Modificación database.php. Fuente: Elaboración propia.....	72
Figura 62: Modificación AppServiceProvider.php. Fuente: Elaboración propia.....	73
Figura 63: Modificación .env. Fuente: Elaboración propia.....	73
Figura 64: Pantalla de inicio de la web. Fuente: Elaboración propia.....	74
Figura 65: Pantalla de registro. Fuente: Elaboración propia.....	75
Figura 66: Pantalla de inicio. Login. Fuente: Elaboración propia.....	76
Figura 67: Pantalla de bienvenida. Fuente: Elaboración propia.....	77
Figura 68: Menú de usuario. Fuente: Elaboración propia.	78
Figura 69: Modificación de datos del usuario. Fuente: Elaboración propia.....	78
Figura 70: Creación de una publicación. Fuente: Elaboración propia.....	79
Figura 71: Pantalla del detalle de la publicación. Fuente: Elaboración propia.....	80
Figura 72: Cabecera del detalle de la publicación. Fuente: Elaboración propia.....	81
Figura 73: Cuerpo del detalle de la publicación. Fuente: Elaboración propia.....	81
Figura 74: Pantalla de búsqueda de usuarios. Fuente: Elaboración propia.....	83

INDICE DE TABLAS

Tabla 1: Pruebas basadas en pruebas de caja negra. Fuente: Elaboración propia.....	48
--	----

1.-INTRODUCCIÓN

1.1.-MOTIVACIÓN

Bien conocido es que el amor por los libros puede llegar a cuotas incalculables por parte de todo aquel que sea un lector asiduo. Muchos de ellos han comenzado desde muy pequeños, cogiendo el hábito de la lectura, influenciados por los padres o alguna otra persona cercana. Aprendiendo a apreciar y cuidar cada libro que por sus manos pase. Es gracias a la lectura que cada lector es capaz de descubrir mundos increíbles, vivir aventuras fuera de su alcance, viajar a lugares que no creería posible, conocer personajes que permanecerán a su lado para toda la vida, disfrutando y adquiriendo el conocimiento y las experiencias que en cada uno de ellos están escritas. Siendo tal la pasión de un lector por los libros, siempre estará a la búsqueda de nuevas lecturas, nuevos autores o temáticas, tratando de encontrar a otros lectores con sus mismos gustos.

Ayudar a los lectores a encontrar respuestas a dichas búsquedas y posibilitar la interacción con otros lectores es la principal motivación por la que se dispone la creación de una aplicación web dedicada a dicho propósito. Por medio de esta plataforma, el lector será capaz de descubrir cuentas de otros lectores con gustos afines, realizando un seguimiento de las mismas y pudiendo guardar sus publicaciones favoritas para futuras lecturas. Al mismo tiempo, el usuario tendrá la posibilidad de comentar cada publicación, ampliando sus conocimientos como los del resto de usuarios.

1.2.-ABSTRACT

Quick access to information is one of the main objectives to be achieved today. A person who is looking for answers to their concerns is not willing to spend a lot of time in such a search. It is for this reason that they want to solve their searches in a single place and, if it is possible, find an answer that aligns with their tastes. For a regular reader, discovering new readings can be a really hard work, with a lot of research and time invested, because the offer is as wide as it is varied. Thanks to the application created for this project, an user will be able to find other readers with similar tastes and discover the books they have read. In this way, the user will quickly find their next book to read, with a review or description of it and a discussion thread about its content. At the same time, the user will be able to create their favorite book lists, track accounts with similar tastes and publish their own readings, in order to share its with the rest of the platform community. This is how this application will allow the users to create their personal libraries and turn it into a social space where they can interact with other readers, promoting cultural enrichment and the discovery of new titles, authors and topics to read.

1.3.-OBJETIVOS PROPUESTOS

Los objetivos de este proyecto son la realización de una aplicación web que buscará crear una biblioteca personal y convertirla en una plataforma social, donde un usuario pueda compartir sus gustos literarios, descubriendo nuevos libros y/o autores mediante la interacción con otros usuarios registrados en la misma. El usuario podrá hacer publicaciones sobre sus libros favoritos, ver las de otros usuarios, indicar que le gustan, seguir a otras cuentas e interactuar con las mismas por medio de una caja de comentarios en cada una de las publicaciones que se realicen. La plataforma será privada, lo cual implicará que el usuario se registre y deba realizar un login para ingresar a la misma. Para el correcto registro en la aplicación, el usuario deberá introducir sus nombres y apellidos, un nombre de usuario, una dirección de email, su imagen de perfil y una contraseña. Una vez registrado, el ingreso a la plataforma se realizará utilizando la dirección de email y la contraseña previamente registrada. Las publicaciones realizadas por los usuarios recogerán el título del libro, una imagen de su portada, el autor del libro, la editorial, el número de páginas y una breve descripción o review del mismo. Cada una de las publicaciones que realice un usuario se listarán en su página de perfil y podrán ser modificadas o eliminadas por éste en el momento que lo crea oportuno. Un usuario podrá indicar que le gusta la publicación de otro usuario por medio de un icono de un corazón. Esta acción hará que dicha publicación se guarde como favorita y se listará en la página de favoritos del usuario, pudiendo eliminar tal acción en el momento que lo desee. También se contará con la opción de seguir una o varias cuentas registradas en la web. De esta manera, el usuario podrá ver listadas las publicaciones de las cuentas que esté siguiendo en la página principal de la plataforma. En caso de querer descubrir cuentas nuevas, el usuario contará con la opción de buscar usuarios concretos o ver el listado completo de publicaciones realizadas en la plataforma. La búsqueda de usuarios se realizará en la página destinada a dicha acción. En ésta se listarán todos los usuarios registrados y contará con un cajón de búsqueda, para filtrar la lista por el nombre introducido en el mismo. Por otro lado, el listado de todas las publicaciones de la plataforma se encontrará en la página global de la web. Aquí se podrá ver en orden de publicación, tanto las de las cuentas seguidas como la de los demás usuarios registrados. Por último, los usuarios podrán comentar cada publicación que deseen por medio de una caja de comentarios. Todos los datos que recoja la aplicación serán almacenados en una base de datos, implementada y gestionada con MySQL.

2.-METODOLOGÍA UTILIZADA

Existen varias metodologías para llevar a cabo un proyecto. Cada una de ellas tiene sus ventajas y desventajas. De todas ellas, la que más se adapta al proyecto es el **modelo en cascada con retroalimentación**. Este cuenta con cinco fases bien definidas. Se comienza con el **análisis**, seguido del **diseño**, la **codificación**, sus **pruebas** y finaliza con el **mantenimiento**. Una de las principales ventajas con las que cuenta este modelo es el poder retomar la etapa anterior del proyecto en caso de que en una fase se encuentren errores o que hayan cambiado algunos de los requisitos del mismo. A su vez, facilita la comprensión y la planificación del trabajo, aumentando la calidad del producto final. El modelo también es recomendado cuando los requisitos del proyecto son estables y no se necesita realizar versiones intermedias del mismo. Estas cualidades son otro reflejo del proyecto actual, por lo cual se considera su elección como la más adecuada. A su vez, se hará uso de la programación orientada a objetos sobre un **MVC**, *modelo vista controlador*.

3.-TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS EN EL PROYECTO

IDE

Para el desarrollo del proyecto se han tenido en cuenta varios de los entornos integrados de desarrollo disponibles en el mercado. Finalmente, se ha elegido **Visual Studio Code**. Este IDE (*Integrated Development Environment*) es de código abierto, gratuito y está desarrollado por Microsoft. Cuenta con muchas herramientas que facilitan el trabajo del programador y es compatible con una gran cantidad de lenguajes de programación. A su vez, cuenta con soporte para el control de versiones, depuración, corrección de sintaxis, autocompletado y refactorización de código. Todas estas cualidades hacen del mismo una herramienta completa para la realización del proyecto actual.

Control de Versiones

Un control de versiones es una herramienta que gestiona de manera automática los cambios que se realizan sobre un proyecto. Esta herramienta ayuda a llevar un correcto control sobre dichos cambios, permitiendo deshacer los mismos o retomar el proyecto a estados más tempranos y continuar trabajando a partir del mismo. Es una gran herramienta para el trabajo en equipo, puesto que da la posibilidad de trabajar sobre distintas ramas de desarrollo para, una vez finalizadas, unir las en la rama principal del proyecto. Todo esto de forma segura. Se encuentran varias opciones en el mercado de controles de versiones, siendo algunas de ellas de pago y otras

gratuitas. Para este proyecto se ha elegido **GIT**, puesto que se integra muy bien con el IDE de Visual Studio Code agilizando enormemente su uso

Plataforma de desarrollo colaborativo

GitHub es una plataforma de desarrollo colaborativo que almacena proyectos por medio del control de versiones GIT. Gracias a esta herramienta se pueden administrar las distintas etapas de un proyecto, creando distintas ramas de trabajo e, incluso, codificando desde el mismo. Sumamente útil para el trabajo en equipo o individual. Es una plataforma perfecta para trabajar de la mano de GIT, manteniendo el proyecto en un repositorio accesible en la red. Se pueden encontrar otras alternativas a esta plataforma, aunque GitHub es una de las más extendidas y documentadas del mercado. Aunque parte de sus utilidades son de pago, GitHub permite trabajar con un proyecto como el actual sin la necesidad de dichas utilidades, presentándose como la gran opción en este caso.

Base de Datos

Para este proyecto se empleará una base de datos relacional utilizando el lenguaje **SQL** (*Structured Query Language*), fundamental en los sistemas de gestión de bases de datos relacionales. La principal característica del lenguaje SQL es que es declarativo en lugar de imperativo.

Se encuentran varias herramientas para gestionar una base de datos, entre las cuales está **MySQL**. Esta es un gestor de base de datos relacional ampliamente utilizada en aplicaciones web. MySQL es una base de datos de rápida lectura, aunque no trabaja muy bien en entornos donde se necesiten altas concurrencias de modificación de datos. Esta cualidad la convierte en una herramienta ideal para el proyecto actual. A su vez, para facilitar el trabajo, se utiliza una interfaz gráfica de usuario para la gestión de la base de datos. Para este proyecto se ha seleccionado **PhpMyAdmin**, siendo gratuita y de fácil acceso desde un navegador web.

Intercambio de datos

Para parte de algunas funcionalidades de este proyecto ha sido necesaria la utilización del intercambio de datos, haciendo uso del formato **JSON** (*JavaScript Object Notation*). Este, como se da a entender anteriormente, es un formato de intercambio de datos.

Maquetación de la Web

Para la maquetación de la página web se ha elegido el lenguaje de marcado **HTML** (*HyperText Markup Language*), siendo uno de los lenguajes más importantes en la definición del contenido de una web y creando su estructura básica. A su vez, se utilizará **CSS** (*Cascading Style Sheets*) para dar estilo a la presentación de la interfaz web. CSS es un lenguaje de presentación denominado como hojas de estilo en cascada y es imprescindible a la hora de darle un aspecto más elegante a la estructura generada con HTML. Por último, para agilizar parte de la maquetación y los estilos se implantará **Bootstrap**, un framework que reúne una serie de herramientas y plantillas predefinidas basadas en HTML y CSS.

Lenguajes de Programación

Entre las varias opciones con las que se cuenta a la hora de elegir un lenguaje de programación para el entorno servidor, la elección para este proyecto ha sido **PHP** (*Hypertext Preprocessor*). Principalmente, por ser uno de los lenguajes más extendidos en dicho ámbito y por contar con una amplia documentación sobre el mismo. Otra de las razones por la cual se elige este lenguaje en el proyecto actual es por la utilización del framework **Laravel**, sobre el cual se entrará en detalle más adelante.

Por otro lado, para parte de las funcionalidades de cara al cliente en este proyecto se ha utilizado el lenguaje de programación **JavaScript**, haciendo uso de su biblioteca **jQuery** para conseguir darle mayor dinamismo a la web. También se ha hecho uso de la comunicación asíncrona con el servidor mediante **AJAX** (*Asynchronous JavaScript And XML*), evitando el refresco de la página, mejorando la experiencia del usuario.

Framework

Teniendo en cuenta el tiempo disponible y las distintas necesidades del proyecto se ha decidido utilizar un entorno de trabajo que nos brinde una base firme para la organización y el desarrollo del mismo. En este caso se llevará a cabo con **Laravel**.

Si bien es posible hacer uso del **Modelo Vista Controlador** tradicional con este framework, Laravel utiliza un sistema de rutas definidas por el programador, las que hacen más limpio y legible el código resultante. Una de las principales características por las que se eligió este framework como base de desarrollo de este proyecto es la inclusión de su sistema de mapeo de datos relacionales, **Eloquent**. Este **ORM** (*Object-Relational Mapping*) facilita enormemente la creación de modelos, reduciendo en buena cuota la cantidad de código a escribir. Al igual que con el sistema de rutas, su uso es opcional. Otra característica a destacar es su sistema de procesamiento de plantillas, destinado al desarrollo de las vistas, llamado **Blade**. Este sistema

ayuda a tener un código más legible y resumido en las vistas a la vez que es procesado más rápidamente. Contando con amplia documentación y la automatización de gran parte de los procesos de desarrollo de código, Laravel es la elección acertada para el realizar el proyecto actual.

Iconos

Para mejorar el aspecto de la web se ha hecho uso de una serie de iconos desarrollados con CSS, incluidos en la biblioteca de la plataforma **Font Awesome**.

Diseño de esquemas y diagramas

Para la realización de estos trabajos se ha hecho uso de la aplicación web **Draw.io**. Esta herramienta es gratuita y cuenta con una amplia biblioteca de plantillas predefinidas.

Diseño de bocetos de la interfaz

Para la realización de la maquetación orientativa de la interfaz de usuario se utilizó la aplicación **Paint**.

4.-ESTIMACIÓN DE RECURSOS Y PLANIFICACIÓN

La estimación inicial del tiempo global que ocupará el desarrollo del proyecto se ha calculado con base en las horas totales dadas por el centro educativo a la realización del módulo de Proyecto de Desarrollo de Aplicaciones Web. Éste es de 99 horas totales. Las tareas sobre las que se estimarán los tiempos de trabajo son las de análisis, diseño, codificación y pruebas. Para conseguir unos valores más exactos se considera oportuno redondear este tiempo en 100 horas y englobar el tiempo de las pruebas junto al de la codificación, puesto que se realizarán en conjunto. De este modo, se puede estimar que inicialmente el trabajo sobre el proyecto tomará los siguientes tiempos:

Análisis

Se dedicará un 25% del tiempo total, resultando en unas 25 horas.

Diseño

Se dedicará un 25% del tiempo total estimativo, siendo unas 25 horas.

Codificación y pruebas

Estimado en el 50% restante del total, resultando unas 50 horas.

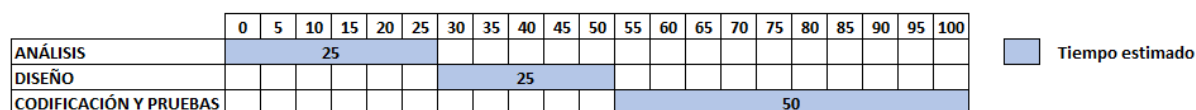


Figura 1: Diagrama de Gantt. Fuente: Elaboración propia.

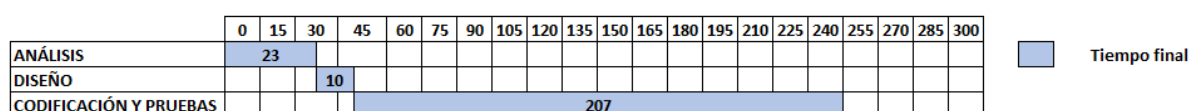


Figura 2: Diagrama de Gantt. Fuente: Elaboración propia.

Como queda patente en los diagramas de Gantt, entre la estimación de tiempo inicial y el tiempo real que ha requerido el proyecto, la diferencia es clara. Sobre todo, en el tiempo estimado de codificación y pruebas. Pese a que se trabajó con herramientas destinadas al ahorro de código y a la simplificación del mismo, el tiempo ocupado en este ha superado ampliamente el estimado. Se tiene en consideración que, el hecho de trabajar con tecnologías nuevas y el tiempo que implica estudiarlas y ponerlas en práctica, han sido los factores decisivos a la hora de marcar esta diferencia de tiempos.

5.-DESARROLLO DEL PROYECTO

5.1.-ANÁLISIS

Luego del análisis sobre el proyecto se desglosan los siguientes requisitos funcionales y no funcionales del mismo:

Requisitos funcionales

- Un usuario podrá registrar una cuenta en la plataforma por medio de sus nombres, apellidos, nombre de usuario, email, password y una imagen de perfil.
- Un usuario podrá acceder a su cuenta utilizando su email y contraseña.
- Un usuario podrá modificar sus datos personales.
- Un usuario podrá crear publicaciones con un título, imagen de portada, autor, editorial, cantidad de páginas y una breve descripción.

- Un usuario podrá indicar que le gusta su publicación o la de otro usuario, siendo añadida a sus favoritos.
- Un usuario podrá indicar que le deja de gustar su publicación o la de otro usuario, dejando de estar añadida a sus favoritos.
- Un usuario podrá ver que cuenta ha indicado que le gusta su publicación o la de otro usuario.
- Un usuario podrá comentar en su publicación o en la de otro usuario.
- Un usuario podrá modificar su publicación.
- Un usuario podrá eliminar su publicación.
- Un usuario podrá buscar las cuentas de otros usuarios.
- Un usuario podrá seguir a otro usuario, quedando listadas sus publicaciones en la vista principal del usuario.
- Un usuario podrá dejar de seguir a otro usuario, dejando de quedar listadas sus publicaciones en la vista principal del usuario.
- Un usuario podrá ver el listado de publicaciones de todas las cuentas registradas en la plataforma mediante la vista global de la misma.
- Un usuario podrá visitar el perfil de otro usuario, pudiendo ver el listado de todas sus publicaciones.
- Un usuario podrá ver que cuentas siguen o cuales está siguiendo otro usuario.
- Un usuario podrá salir de la plataforma mediante un logout.

Requisitos no funcionales

- La aplicación funcionará en cualquier navegador web.
- Se utilizarán los lenguajes HTML, CSS y Bootstrap para la maquetación de la web.
- Se utilizará MySQL para la gestión de la base de datos.
- Se utilizarán los lenguajes JavaScript y jQuery para parte de las funcionalidades de la web.
- Se utilizará Ajax para la comunicación en segundo plano con el servidor para parte de las funcionalidades de la web.
- Se utilizará PHP con Laravel para la programación en el entorno servidor.
- La interfaz de usuario será implementada a través de ventanas en el navegador web, debiendo ser de fácil utilización e intuitiva.

Esquema Entidad-Relación

Teniendo en cuenta el análisis del proyecto y sus necesidades, se elabora el siguiente Esquema Entidad-Relación:

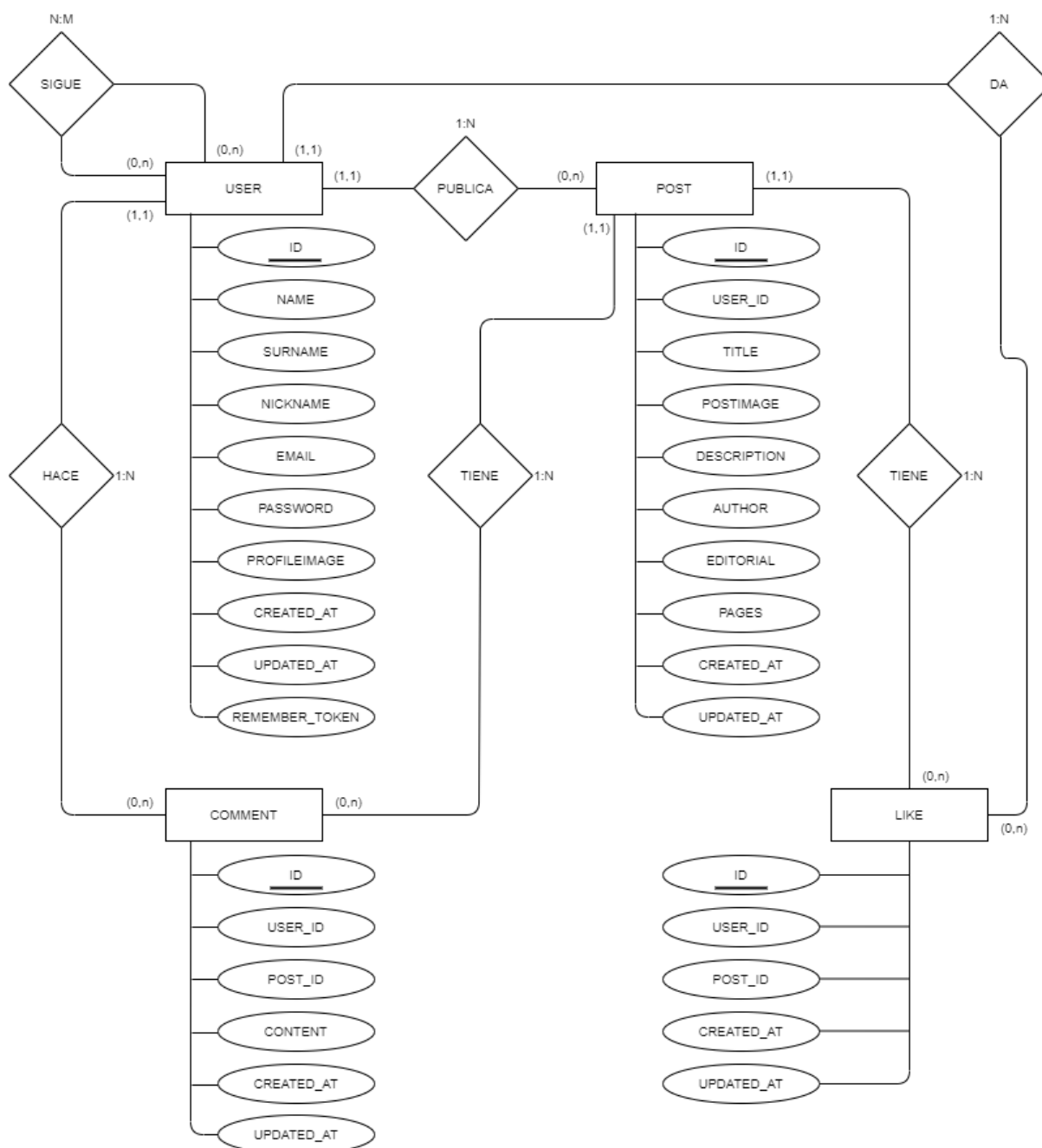


Figura 3: Esquema Entidad-Relación. Fuente: Elaboración propia.

El diseño del Esquema Entidad-Relación de la Figura 3 consta con la siguiente información:

Entidad “**USER**”. Usuario que utilizará la plataforma. Almacenará sus nombres, apellidos, nombre de usuario, email, contraseña y una imagen de perfil. A su vez, registra la fecha de creación y de modificación del registro.

Entidad “**POST**”. Es la entidad que recoge el registro de una publicación. Esta almacenará el título, imagen de portada, autor, editorial, cantidad de páginas y una descripción. También guardará la referencia del usuario que realiza la publicación. Al igual que con la entidad “USER”, almacenará la fecha de creación y modificación del registro.

Entidad “**COMMENT**”. Recogerá la información del registro de un comentario realizado por un usuario. Almacenará el contenido del comentario, como la referencia del usuario que lo realiza y la de la publicación sobre la cual lo indica. También almacenará la fecha de creación y modificación.

Entidad “**LIKE**”. Recogerá el registro de un like, almacenando la fecha de creación y modificación del mismo, como la referencia del usuario que lo realiza y la de la publicación sobre la cual lo indica.

Como se puede observar en el Esquema Entidad-Relación, la entidad “USER” cuenta con una relación reflexiva de muchos a muchos. De esta relación derivará una tabla nueva en la base de datos, la cual se llamará “**FOLLOWERS**”. En esta se almacenará la referencia del usuario que indica que seguirá una cuenta y la del usuario de la cuenta seguida.

Modelo Relacional

Finalmente, podemos elaborar un **modelo relacional** de la base de datos. Este modelo será el siguiente:

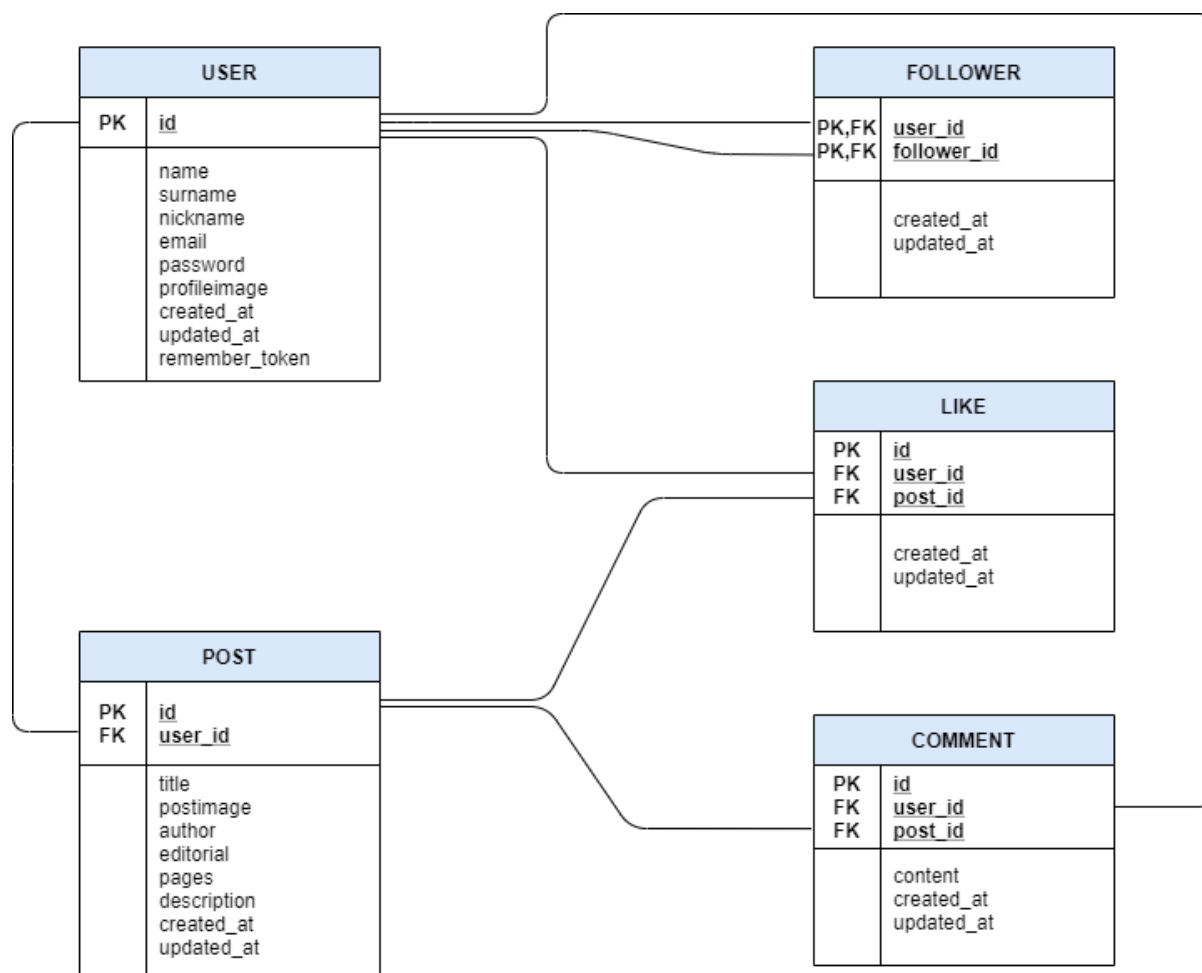


Figura 4: Diagrama del Modelo Relacional. Fuente: Elaboración propia.

Cabe apuntar la importancia que tiene el anterior análisis para una correcta realización de la estructura de la base de datos. Tanto para diferenciar las distintas entidades que la conforman como sus atributos. Fundamentalmente sus relaciones, siendo el factor decisivo para el correcto funcionamiento del proyecto.

5.2.-DISEÑO

Se realiza un diagrama de clases para una mejor comprensión del sistema.

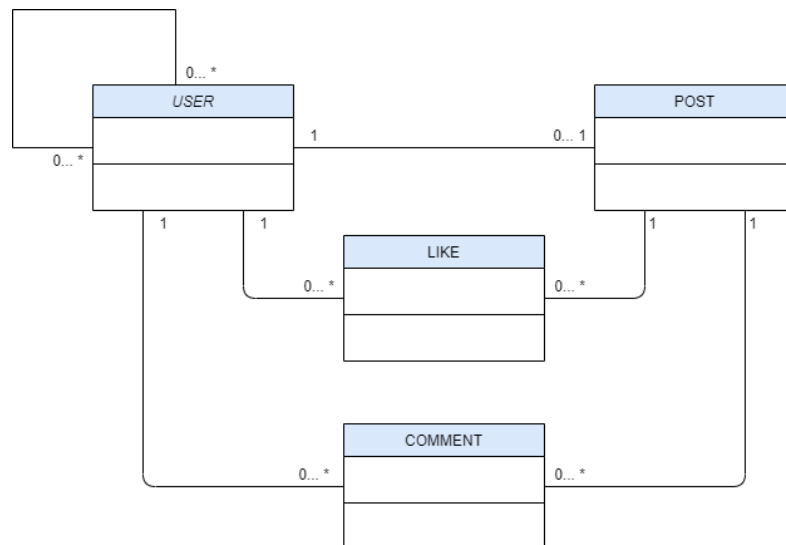


Figura 5: Diagrama de clases. Fuente: Elaboración propia.

Seguido, se realiza un diagrama de casos de uso para facilitar la comprensión del sistema y las futuras pruebas del mismo. Dicho diagrama es el siguiente:

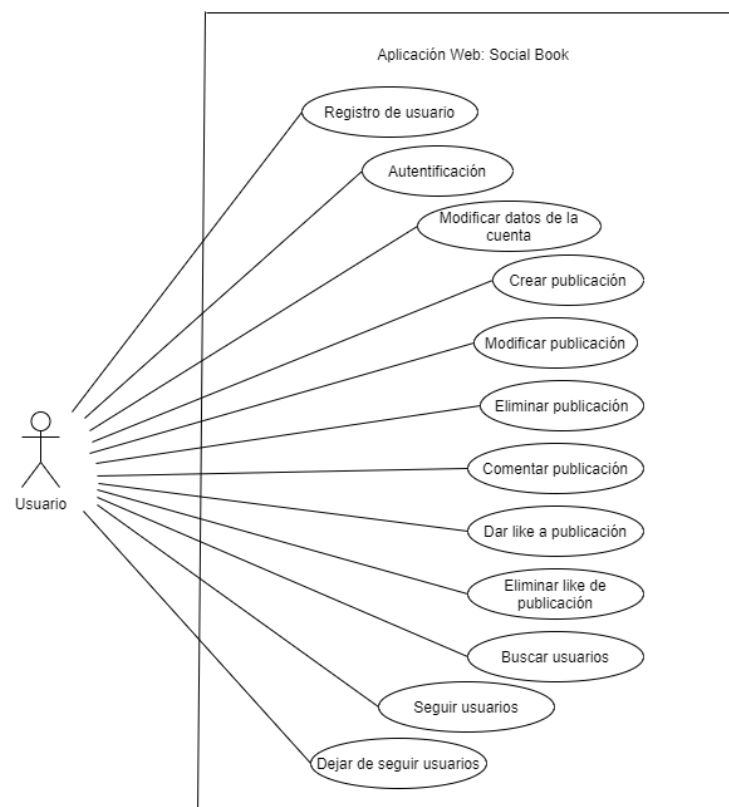


Figura 6: Diagrama de Casos de Uso. Fuente: Elaboración propia.

Teniendo clara la estructura y las funcionalidades de la aplicación a desarrollar se comienza con el diseño de la interfaz y la codificación.

Diseño de la interfaz de usuario

Por medio de la aplicación Paint se realizó un diseño estimativo sobre cómo debería de ser la interfaz del usuario. En las siguientes figuras se pueden observar las principales vistas de la aplicación. Siendo que, por ejemplo, el formulario de creación de una publicación como el de modificación de la misma son iguales, se decidió no incluirlo en los bocetos.

Cabe aclarar que las siguientes imágenes son orientativas. El maquetado final, como se podrá observar más adelante, es algo distinto.

The wireframe shows a header bar with 'Social Book' on the left and 'Login Registro' on the right. Below the header is a section titled 'Login'. Inside this section, there are two input fields: 'E-Mail' and 'Password'. Below these fields is a blue button labeled 'Login'.

Figura 7: Vista de Login. Fuente: Elaboración propia.

The wireframe shows a header bar with 'Social Book' on the left and 'Login Registro' on the right. Below the header is a section titled 'Registro'. Inside this section, there are several input fields: 'Nombre', 'Apellido', 'Nickname', 'E-Mail', 'Avatar' (with a 'Seleccionar archivo' button), 'Password', and 'Confirmar Password'. At the bottom of the section is a blue button labeled 'Registro'.

Figura 8: Vista de Registro de usuario. Fuente: Elaboración propia.



Figura 9: Vista del perfil de usuario. Fuente: Elaboración propia.

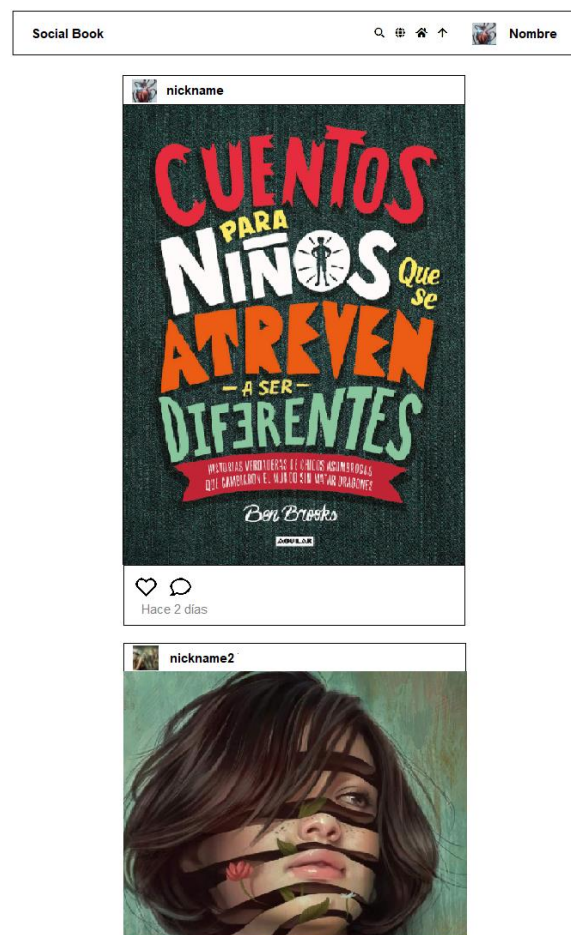


Figura 10: Vista de listado de publicaciones. Fuente: Elaboración propia.

Social Book

Crear Publicación

Título

Imagen

Autor

Editorial

Número de páginas

Descripción

Figura 11: Vista de creación de publicación. Fuente: Elaboración propia.

Social Book

Configuración

Nombre

Apellido

Nickname

E-Mail

Avatar

Figura 12: Vista de modificación de los datos del usuario. Fuente: Elaboración propia.

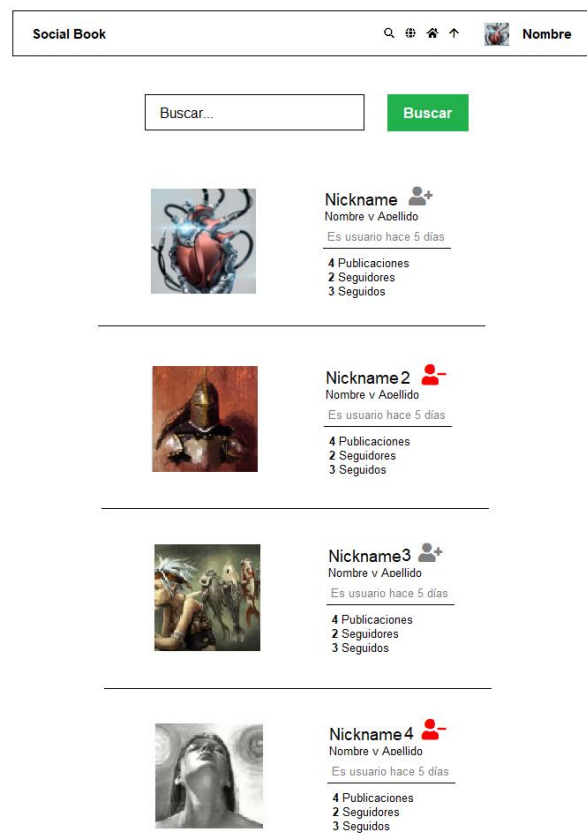


Figura 13: Vista de detalle de una publicación.
Fuente: Elaboración propia.

Figura 14: Vista de buscador de usuarios. Fuente:
Elaboración propia.

Creación de la Base de Datos

Una vez realizado el Esquema Entidad-Relación se pudo comenzar con la creación de la base de datos. Haciendo uso de la interfaz gráfica de usuario de PhpMyAdmin, se crea una nueva base de datos y se le da nombre. Seguido, se crean las distintas tablas que conformarán dicha base de datos. Esto se realizó con el uso de código SQL. En la siguiente figura se puede ver un ejemplo de la creación de una de las tablas:



Figura 15: Creación de la tabla USERS. Fuente: Elaboración propia.

Finalizada la creación de la base de datos con todas sus tablas, llegó el turno de preparar el entorno de trabajo. Para la realización del proyecto actual se escogió utilizar el lenguaje PHP junto al framework Laravel, entre otros.

Composer

Lo primero que se ha realizado es la instalación de **Composer**. Este es un componente que gestiona las dependencias de PHP, mediante el cual podemos instalar todo tipo de librerías que utilizará nuestro proyecto.

La instalación de Composer se realiza mediante la descarga del mismo desde su web:

<https://getcomposer.org/Composer-Setup.exe>

Una vez descargado solo queda ejecutar el archivo y seguir los pasos indicados en la instalación.

Instalación de Laravel

Por medio de Composer, se realiza la instalación de Laravel. Se lleva a cabo accediendo al directorio donde se desea realizar el proyecto y, desde allí y por medio de la terminal del sistema, se ejecuta el siguiente código:

```
composer create-project laravel/laravel SocialBook "5.6.*"
```

Una vez finalizada la instalación, en el directorio de trabajo queda compuesta la estructura de trabajo que ofrece Laravel. Dicha estructura de archivos es la que se puede observar en la siguiente figura.

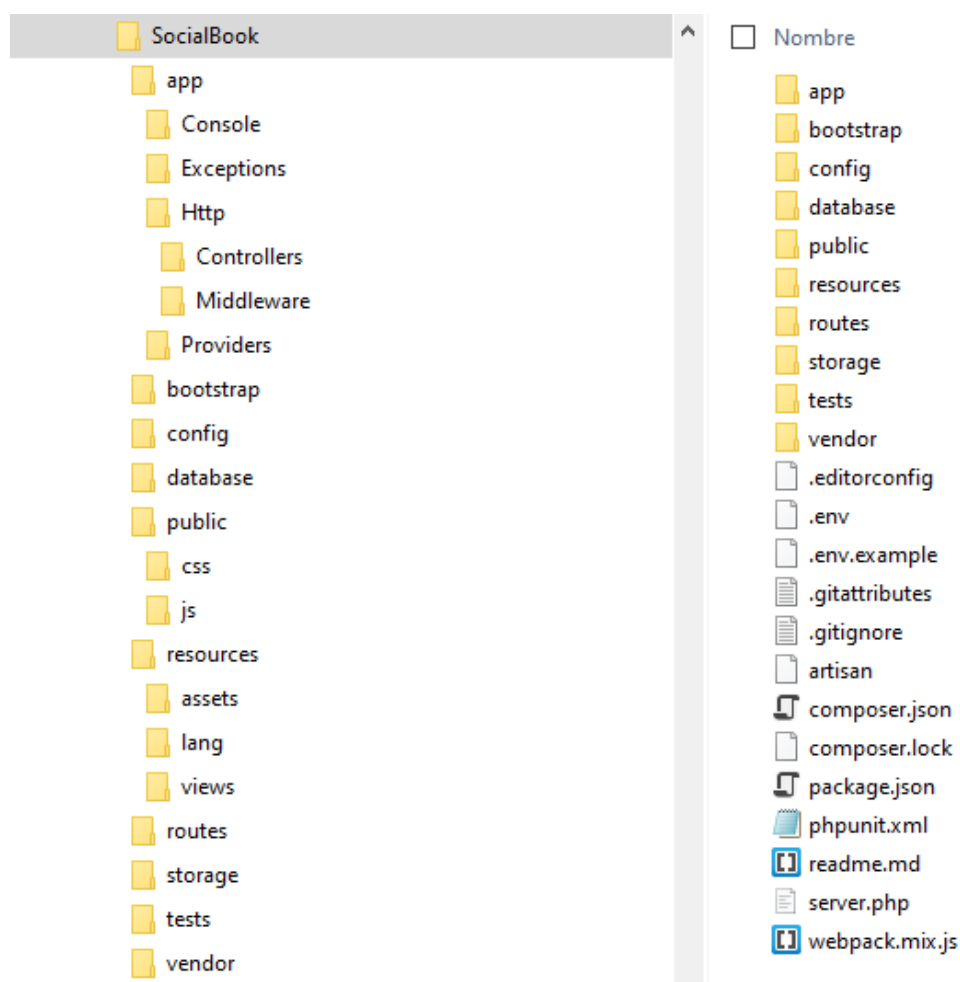


Figura 16: Estructura de archivos generado por Laravel. Fuente: Elaboración propia.

Una de las grandes ventajas de utilizar con este Framework es el que ya está estructurado para trabajar con el Modelo Vista Controlador, haciendo más claro el código y mucho mejor organizado.

También trae integrado por defecto la librería jQuery y Bootstrap y muchas funcionalidades necesarias ya codificadas. Como puede ser el sistema de autenticación de usuarios. Otra gran cualidad con la que cuenta es su ORM, Eloquent. Gracias a éste, se agiliza enormemente el trabajo con la base de datos, aunque es muy importante su correcta configuración.

Host Virtual

Para trabajar y realizar pruebas en un ámbito local fue necesario crear un host virtual. Para esto se debió crear un host virtual en Windows editando el archivo *hosts* y luego, configurar Apache para que identifique el host virtual. Esto se logró modificando el archivo *httpd-vhosts.conf*.

```
1  # Copyright (c) 1993-2009 Microsoft Corp.
2  #
3  # This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
4  #
5  # This file contains the mappings of IP addresses to host names. Each
6  # entry should be kept on an individual line. The IP address should
7  # be placed in the first column followed by the corresponding host name.
8  # The IP address and the host name should be separated by at least one
9  # space.
10 #
11 # Additionally, comments (such as these) may be inserted on individual
12 # lines or following the machine name denoted by a '#' symbol.
13 #
14 # For example:
15 #
16 #       102.54.94.97       rhino.acme.com           # source server
17 #       38.25.63.10       x.acme.com               # x client host
18
19 # localhost name resolution is handled within DNS itself.
20 #   127.0.0.1       localhost
21 #   ::1             localhost
22
23 127.0.0.1       SocialBook.com
24
```

Figura 17: Modificación del archivo *hosts*. Fuente: Elaboración propia.


```
1 # Virtual Hosts
2 #
3 # Required modules: mod_log_config
4
5 # If you want to maintain multiple domains/hostnames on your
6 # machine you can setup VirtualHost containers for them. Most configurations
7 # use only name-based virtual hosts so the server doesn't need to worry about
8 # IP addresses. This is indicated by the asterisks in the directives below.
9 #
10 # Please see the documentation at
11 # <URL:http://httpd.apache.org/docs/2.4/vhosts/>
12 # for further details before you try to setup virtual hosts.
13 #
14 # You may use the command line option '-S' to verify your virtual host
15 # configuration.
16
17 #
18 # Use name-based virtual hosting.
19 #
20 ##NameVirtualHost *:80
21 #
22 # VirtualHost example:
23 # Almost any Apache directive may go into a VirtualHost container.
24 # The first VirtualHost section is used for all requests that do not
25 # match a ##ServerName or ##ServerAlias in any <VirtualHost> block.
26 #
27 ##<VirtualHost *:80>
28     ##ServerAdmin webmaster@dummy-host.example.com
29     ##DocumentRoot "C:/xampp/htdocs/dummy-host.example.com"
30     ##ServerName dummy-host.example.com
31     ##ServerAlias www.dummy-host.example.com
32     ##ErrorLog "logs/dummy-host.example.com-error.log"
33     ##CustomLog "logs/dummy-host.example.com-access.log" common
34 ##</VirtualHost>
35
36 ##<VirtualHost *:80>
37     ##ServerAdmin webmaster@dummy-host2.example.com
38     ##DocumentRoot "C:/xampp/htdocs/dummy-host2.example.com"
39     ##ServerName dummy-host2.example.com
40     ##ErrorLog "logs/dummy-host2.example.com-error.log"
41     ##CustomLog "logs/dummy-host2.example.com-access.log" common
42 ##</VirtualHost>
43
44
45 <VirtualHost *:80>
46     DocumentRoot "C:/xampp/htdocs/"
47     ServerName localhost
48 </VirtualHost>
49
50 <VirtualHost *:80>
51     DocumentRoot "C:/xampp/htdocs/SocialBook/public"
52     ServerName SocialBook.com
53 </VirtualHost>
54
55
```

Figura 18: Modificación del archivo `httpd-vhosts.conf`. Fuente: Elaboración propia.

Una vez realizadas las modificaciones, se reinicia el servidor y se prueba el correcto funcionamiento. En un navegador web, se ingresa a la URL que se definió devolviendo la vista principal de Laravel.

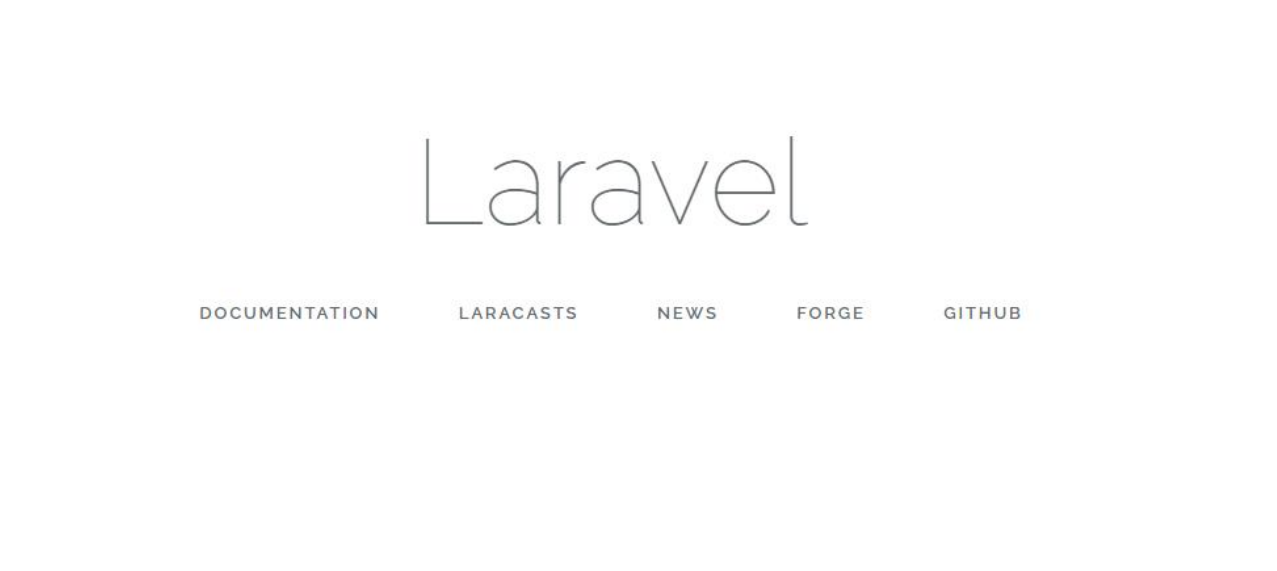
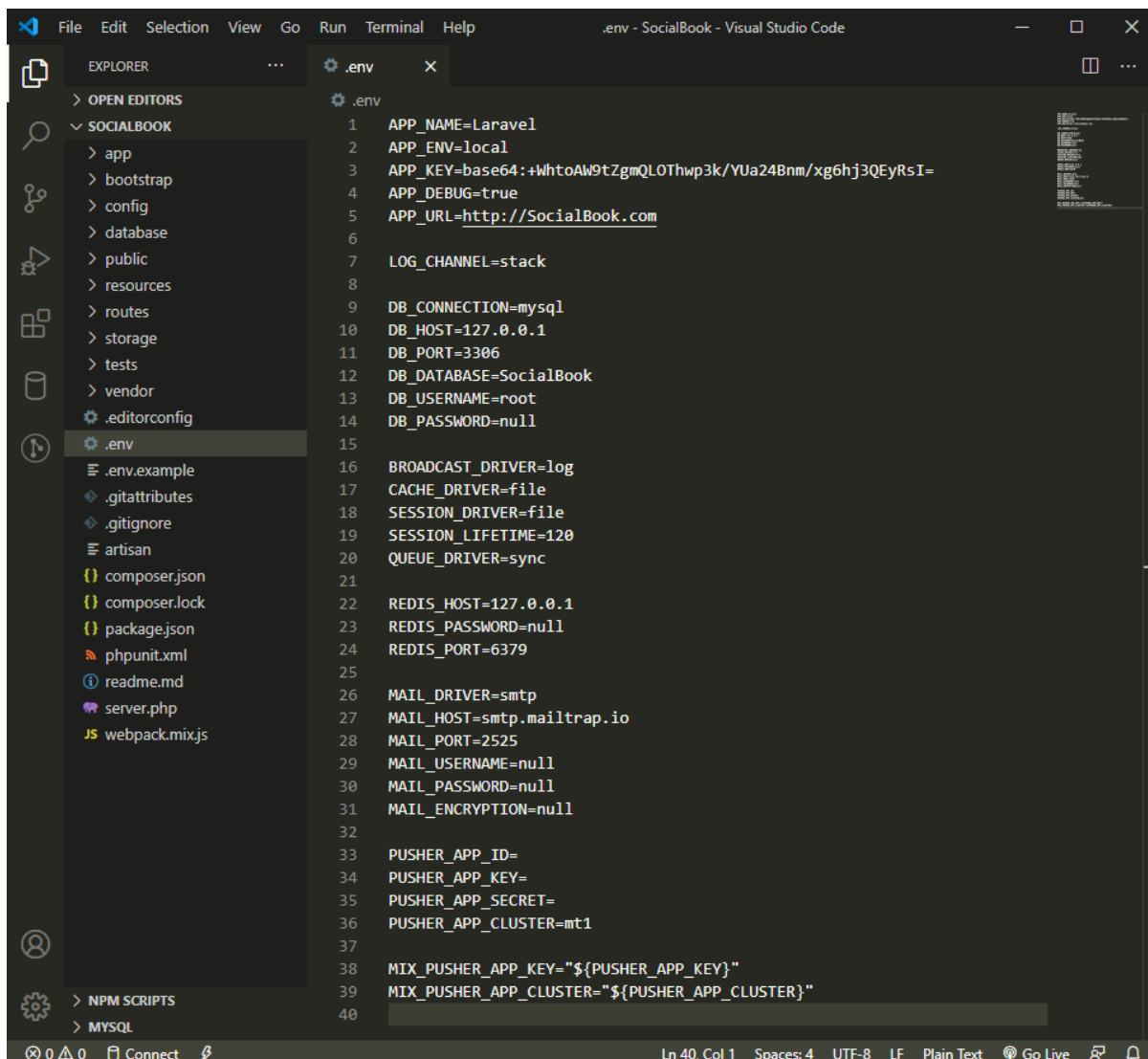


Figura 19: Vista principal de Laravel. Fuente: Elaboración propia.

Configuración de la Base de Datos en Laravel

Ya con el proyecto en marcha y el correcto funcionamiento del host virtual, se continuó con la configuración de la base de datos en Laravel. Para esto es necesario modificar el archivo creado por el framework llamado **.env**, el cual se encuentra en la raíz del directorio.

En este archivo se indica la URL del proyecto en APP_URL. En el bloque de la base de datos, se modifican los parámetros destinados a definir el nombre de la misma, el nombre del usuario y la contraseña. Estos son DB_DATABASE, DB_USERNAME y DB_PASSWORD. Estos sencillos pasos son los necesarios para lograr conectar la base de datos con nuestro proyecto en Laravel en un entorno local. Una vez finalizado el proyecto, cuando se haya de desplegar en un hosting, se deberán modificar el resto de parámetros de acuerdo a las necesidades del momento.



```
.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:1WhtoAw9tZgmQLOThwp3k/YUa24Bnm/xg6hj3QEYRsI=
4 APP_DEBUG=true
5 APP_URL=http://SocialBook.com
6
7 LOG_CHANNEL=stack
8
9 DB_CONNECTION=mysql
10 DB_HOST=127.0.0.1
11 DB_PORT=3306
12 DB_DATABASE=SocialBook
13 DB_USERNAME=root
14 DB_PASSWORD=null
15
16 BROADCAST_DRIVER=log
17 CACHE_DRIVER=file
18 SESSION_DRIVER=file
19 SESSION_LIFETIME=120
20 QUEUE_DRIVER=sync
21
22 REDIS_HOST=127.0.0.1
23 REDIS_PASSWORD=null
24 REDIS_PORT=6379
25
26 MAIL_DRIVER=smtp
27 MAIL_HOST=smtp.mailtrap.io
28 MAIL_PORT=2525
29 MAIL_USERNAME=null
30 MAIL_PASSWORD=null
31 MAIL_ENCRYPTION=null
32
33 PUSHER_APP_ID=
34 PUSHER_APP_KEY=
35 PUSHER_APP_SECRET=
36 PUSHER_APP_CLUSTER=mt1
37
38 MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
39 MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
40
```

Figura 20: Configuración de la Base de Datos. Fuente: Elaboración propia.

Ya con la base de datos configurada y conectada, se continúa con la creación de los modelos.

Modelos

Uno de los componentes más importantes del Modelo Vista Controlador son los modelos. El modelo o entidad dentro de Laravel representará a una tabla de la base de datos. Estos son clases que al utilizarlas instancian un objeto que representa un registro de la base de datos. Al instalar Laravel, este nos crea a priori el modelo User. Es aquí donde ya viene configurada la autenticación de usuario. Es de esta forma que solo se ha de crear solo el resto de modelos del proyecto. Para esta tarea se utiliza una herramienta de Laravel llamada **Artisan**.

Artisan es una interfaz de línea de comandos que tiene Laravel, mediante el cual creamos los modelos. La línea de código para este trabajo es:

```
php artisan make:model <nombre del modelo>
```

De esta forma es creamos los modelos de este proyecto. Los modelos se crean dentro de la carpeta app. Estos modelos son clases que extienden de la clase Model de Laravel y se guardan como un archivo con extensión php. Este proyecto cuenta con cuatro modelos. *User.php*, *post.php*, *comment.php* y *like.php*

Configuración de las relaciones entre modelos

Gracias al ORM Eloquent, se puede trabajar de manera muy sencilla con la base de datos del proyecto. Para esto es necesario indicar en los modelos el tipo de relación que tendrán entre ellos. De esta manera, primero se indica en cada modelo sobre que tabla se trabajará y luego se definen las relaciones. En la siguiente figura se puede observar un ejemplo de esto sobre el modelo *Comment*.

```
app > comment.php > ...

1  <?php
2
3  namespace App;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class comment extends Model
8  {
9      //Indica con que tabla trabaja el modelo
10     protected $table = "comments";
11
12     //Relación de muchos a uno
13     public function user(){
14         return $this->belongsTo("App\User", "user_id");
15     }
16
17     //Relación de muchos a uno
18     public function posts(){
19         return $this->belongsTo("App\Post", "post_id");
20     }
21
22     //Relación de uno a muchos
23     public function likes(){
24         return $this->hasMany("App\Like");
25     }
26 }
27
```

Figura 21: Configuración de relaciones entre modelos. Fuente: Elaboración propia.

Login y registro de usuario

En Laravel ya contamos con una serie de clases y métodos que agilizan enormemente el desarrollo del login y el registro de usuarios. Es por esto que, con Laravel, generar el login y el registro de usuarios prácticamente se resume a ejecutar un comando desde la línea de comandos artisan. Una vez generado el código, sólo se modifican los puntos específicos que necesita el proyecto. La línea de comando en cuestión es la siguiente:

```
php artisan make:auth
```

De esta forma se generan varios controladores, entre los que están los de login y registro, sus métodos, vistas y rutas necesarias para el correcto funcionamiento del mismo.

Los controladores se crean en la carpeta *app/Http/Controllers*, mientras que las vistas se guardan en la carpeta *resources/views*. A su vez, las rutas estarán guardadas en la carpeta *routes*. Cabe destacar el proceso de registro de un usuario. Para éste, Laravel crea una vista (*register.blade.php*), la cual se modificó para las necesidades del proyecto. Luego, en el controlador *RegisterController.php*, se modificó el validador de datos y se codificó una función para la creación del registro. En la siguiente figura se pueden observar dichas codificaciones.

```
53 //Valida los datos recibidos por el formulario
54 protected function validator(array $data)
55 {
56     return Validator::make($data, [
57         'name' => 'required|string|max:255',
58         'surname' => 'required|string|max:255',
59         'nickname' => 'required|string|max:255',
60         'email' => 'required|string|email|max:255|unique:users',
61         'password' => 'required|string|min:6|confirmed',
62         'profileimage' => 'required|image',
63     ]);
64 }
65
66 //Realiza la inserción de un registro de usuario
67 protected function create(array $data)
68 {
69     //Recoge el nombre del archivo de la imagen
70     $profileimage = $data['profileimage'];
71
72     //Extrae la extensión del archivo y concatena con el nickname del usuario para darle un nombre único a la imagen
73     $ext = $profileimage->getClientOriginalExtension();
74     $imageName = $data['nickname'];
75     $fileName = $imageName . '.' . $ext;
76
77     //Guarda la imagen en la carpeta storage/app/users
78     Storage::disk('users')->put($fileName, File::get($profileimage));
79
80     return User::create([
81         'role' => 'user',
82         'name' => $data['name'],
83         'surname' => $data['surname'],
84         'nickname' => $data['nickname'],
85         'email' => $data['email'],
86         'password' => Hash::make($data['password']),
87         'profileimage' => $fileName,
88     ]);
89 }
```

Figura 22: Validación y creación de un registro de usuario. Fuente: Elaboración propia.

Controladores

Seguido al trabajo anterior se procede a crear el resto de controladores. En ellos se trabajará toda la lógica del proyecto y enviarán información a las vistas del mismo. Una vez más se hace uso de los comandos artisan. Para la creación de cada controlador se utiliza el siguiente comando, donde el nombre del controlar se complementa con la palabra Controller:

```
php artisan make:controller <nombre del controlador>Controller
```

Un ejemplo en este proyecto es el controlador Post. En este caso la línea de comando queda de la siguiente manera:

```
php artisan make:controller PostController
```

Estos controladores son clases que extienden de la clase Controller de Laravel y se guardan como archivos con extensión php. Este proyecto cuenta con cinco archivos de creación propia más los que genera automáticamente el framework. Ellos son *UserController.php*, *PostController.php*, *CommentController.php*, *LikeController.php* y *FollowerController.php*.

UserController.php

Este es el controlador que se encarga de toda la lógica correspondiente al manejo de usuarios. Cuenta con varios métodos, entre los que podemos destacar el de *update*, la cual recoge los datos ingresados en la vista de modificación de datos del usuario, realiza su validación y guarda el nuevo registro. A su vez, recoge la imagen de perfil subida por el usuario y la guarda cambiándole el nombre a uno predefinido. Es decir, la función extrae la extensión del archivo la concatena con el nombre de usuario autenticado, dando por resultado un nombre de archivo nuevo. Este nombre de archivo se guardará en la base de datos y la imagen irá a la carpeta storage de Laravel con el mismo nombre recogido en la base de datos. En la siguiente figura se puede observar el código de la función en este proyecto.

```
app > Http > Controllers > UserController.php > PHP Intelephense > UserController
24 //Realiza modificación de un registro de Usuario en la Base de Datos
25 public function update(Request $request){
26
27     //Recoge el usuario identificado
28     $user = \Auth::user();
29     $id = $user->id;
30
31     //Validación de los datos recibidos
32     $validate = $this->validate($request, [
33         'name' => 'required|string|max:255',
34         'surname' => 'required|string|max:255',
35         'nickname' => 'required|string|max:255|unique:users,nickname,'.$id,
36         'email' => 'required|string|email|max:255|unique:users,email,'.$id,
37         'profileimage' => 'image'
38     ]);
39
40     //Recoge los datos enviados por el formulario
41     $name = $request->input('name');
42     $surname = $request->input('surname');
43     $nickname = $request->input('nickname');
44     $email = $request->input('email');
45     $profileimage = $request->file('profileimage');
46
47     //Asigna nuevos valores al objeto del usuario
48     $user->name = $name;
49     $user->surname = $surname;
50     $user->nickname = $nickname;
51     $user->email = $email;
52
53     //Subir imagen de perfil
54     if($profileimage){
55         //Extrae la extensión del archivo y concatena con el nickname del usuario para darle un nombre único a la imagen
56         $ext = $profileimage->getClientOriginalExtension();
57         $imageName = $user->nickname;
58         $fileName = $imageName . '.' . $ext;
59
60         //Guarda la imagen en la carpeta storage/app/users
61         Storage::disk('users')->put($fileName, File::get($profileimage));
62
63         //Setea el nombre de la imagen de perfil con el que se guarda en el disco
64         $user->profileimage = $fileName;
65     }
66
67     //Ejecuta la consulta y modifica los datos en la Base de Datos
68     $user->update();
69
70     //Realiza una redirección con un mensaje de actualización correctamente realizada
71     return redirect()->route('user.config')->with(['message' => 'Usuario actualizado correctamente']);
72 }
```

Figura 23: Función update. Fuente: Elaboración propia.

Es importante aclarar que, para el correcto funcionamiento del Storage de Laravel, es necesaria su configuración. Esta se realiza en el archivo *filesystems.php*, dentro de la carpeta *config*. Es en este archivo donde se encuentra un array que recoge distintos arrays de destino para las imágenes u otros archivos. Para este proyecto se han creado dos discos. Uno para las imágenes de usuarios y otro para las de las publicaciones. La siguiente figura muestra estos ejemplos.

```
44     'disks' => [  
45         'local' => [  
46             'driver' => 'local',  
47             'root' => storage_path('app'),  
48         ],  
49         'public' => [  
50             'driver' => 'local',  
51             'root' => storage_path('app/public'),  
52             'url' => env('APP_URL').'/storage',  
53             'visibility' => 'public',  
54         ],  
55         //Destino de las imágenes de Usuario  
56         'users' => [  
57             'driver' => 'local',  
58             'root' => storage_path('app/users'),  
59             'url' => env('APP_URL').'/storage',  
60             'visibility' => 'public',  
61         ],  
62         //Destino de las imágenes de Publicaciones  
63         'posts' => [  
64             'driver' => 'local',  
65             'root' => storage_path('app/posts'),  
66             'url' => env('APP_URL').'/storage',  
67             'visibility' => 'public',  
68         ],  
69         's3' => [  
70             'driver' => 's3',  
71             'key' => env('AWS_ACCESS_KEY_ID'),  
72             'secret' => env('AWS_SECRET_ACCESS_KEY'),  
73             'region' => env('AWS_DEFAULT_REGION'),  
74             'bucket' => env('AWS_BUCKET'),  
75             'url' => env('AWS_URL'),  
76         ],  
77     ],  
78     81  
82  
83     ],  
84
```

Figura 24: Configuración del Storage. Fuente: Elaboración propia.

Los demás métodos que conforman el controlador de User son los siguientes:

config(). Se encarga de cargar la vista con el formulario de modificación de los datos personales de un usuario.

getImage(). Recupera la imagen de perfil de usuario del storage.

profile(). Carga la vista del perfil de usuario.

showUsers(). Recoge y muestra en una pantalla todos los usuarios buscados en la plataforma.

showFollowers(). Recoge los seguidores de un usuario.

showFollowing(). Recoge las cuentas seguidas por un usuario.

PostController.php

Este es el controlador que se encarga de toda la lógica correspondiente al manejo de las publicaciones realizadas por los usuarios. Es muy parecido al anterior, con alguna diferencia, como el agregado de un método para eliminar una publicación.

Los siguientes son los métodos que conforman el controlador:

index(). Lista y muestra las publicaciones realizadas por las cuentas seguidas por el usuario identificado.

create(). Carga la vista con el formulario para crear una nueva publicación.

save(). Recoge y valida los datos introducidos en el formulario. Finalmente, guarda el registro de en la base de datos.

getImage(). Recupera la imagen de portada de la publicación del storage.

postdetail(). Carga la vista del detalle de una publicación.

delete(). Elimina una publicación.

edit(). Recoge los datos de una publicación a modificar y renderiza la vista con el formulario para ello.

update(). Recoge y valida los datos introducidos en el formulario. Finalmente, guarda el registro de en la base de datos.

LikeController.php

La lógica encargada del trabajo con todo lo relacionado a los likes dados por los usuarios sobre una publicación se realiza en este controlador. Para esto, cuenta con los siguientes métodos:

like(). Registra un like en la base de datos.

dislike(). Elimina el registro de un like de la base de datos.

userLikes(). Recoge todos los registros de likes realizados por el usuario y los envía a la vista que listará todas las publicaciones favoritas del mismo.

whoLikes(). Recoge todos los usuarios que han indicado que les gusta una publicación.

De entre los anteriores métodos, los más interesantes a desarrollar en detalle son los de *like()* y *dislike()*. Siendo muy parecidos ambos, nos enfocaremos en el primero de ellos. En él se recogen los datos del usuario identificado, para saber quién estará indicando que le gusta una publicación. Seguido, se realiza la consulta a la base de datos, comprobando que no existe un registro previo de like de este usuario sobre la publicación. De no existir, realiza el registro en la base de datos y coge el total de likes con los que cuenta la publicación. Luego, devuelve estos valores en formato JSON. En este proyecto, el registro de likes o dislikes se realiza en segundo plano, utilizando una petición AJAX, lo cual se detallará más adelante. El uso del formato JSON es, justamente, para facilitar el intercambio de datos que ocupan estas funciones. En la siguiente figura se recoge el método anteriormente descrito.

```
16 //Registra un like en la Base de Datos
17 public function like($post_id){
18
19     //Recoge el objeto de usuario
20     $user = \Auth::user();
21
22     //Comprueba si existe un registro de like en la Base de Datos
23     $likeExist = Like::WHERE('user_id', $user->id)
24                 ->WHERE('post_id', $post_id)
25                 ->count();
26
27     //Si no existe un registro de like previo guarda el nuevo registro
28     if($likeExist == 0){
29
30         //Crea el objeto de like
31         $like = new Like();
32
33         //Setea los valores del objeto
34         $like->user_id = $user->id;
35         $like->post_id = (int)$post_id;
36
37         //Guarda un registro de like en la Base de Datos
38         $like->save();
39
40         //Recoge la cantidad de likes de la publicación
41         $likeCount = Like::WHERE('post_id', $post_id)->count();
42
43         //Devuelve un json con los datos del registro
44         return response()->json([
45             'like' => $like,
46             'count' => $likeCount,
47             'message' => 'Has realizado un like'
48         ]);
49     }
50 }
```

Figura 25: Detalle del método que registra un like. Fuente: Elaboración propia.

CommentController.php

Sencillo controlador encargado de la lógica sobre el trabajo con todo lo que concierne a los comentarios sobre una publicación. Cuenta sólo con dos métodos.

save(). Encargado de realizar el registro del comentario en la base de datos.

delete(). Elimina un registro de comentario de la base de datos.

FollowerController.php

Cuenta con los métodos necesarios para seguir o dejar de seguir una cuenta.

follow(). Registra un seguidor en la base de datos.

unfollow(). Elimina un registro de seguidor de la base de datos.

HomeController.php

Este controlador lo crea Laravel. En éste se codificó el siguiente método:

Index(). Recoge y envía a la vista todas las publicaciones registradas en la web.

Middleware

Otro aspecto importante a destacar es que todos los controladores incluyen un método que controla que el acceso a los métodos del controlador sea solo permitido a un usuario autenticado. Esto lo consigue haciendo uso de un **middleware** de autenticación.

Un middleware trabaja como filtro de peticiones HTTP y, en el caso de la autenticación, Laravel lo incluye por defecto.

Vistas

Estas son las que mostrarán al cliente la información deseada y sobre las que el mismo interactuará con el sistema. En Laravel, las vistas se guardan como archivos Blade y se organizan en la carpeta *Views*.

De estas vistas se han de diferenciar las creadas por defecto por el framework y las codificadas para el proyecto. Entre las primeras se pueden destacar las de *login.blade.php* y *register.blade.php*, ambas en la carpeta *auth*, dentro de *views*. Esta última se ha modificado para las necesidades del proyecto, añadiendo campos al formulario. A su vez, se encuentra el archivo *app.blade.php*, el cual es la vista principal sobre la que se cargarán las demás. Es en ésta donde

se codifica la cabecera del proyecto, se indican los metadatos, se importan los estilos, las librerías, los scripts, etc. También se codifica el menú principal del proyecto. Este archivo se encuentra en la carpeta *views*, dentro de *layouts*.

Las vistas que no han sido creadas por Laravel, se han organizado por secciones, donde cada una corresponde a un modelo en específico. Es así que tenemos las vistas correspondientes a *user*, *post* y *likes*, cada cual en su respectiva carpeta con mismo nombre. Luego, para evitar repetir código, las vistas recurrentes se han guardado en la carpeta *includes*. Estas serán incluidas en las que sean necesarias con el comando *@include*.

Un ejemplo de cómo se trabaja con las vistas en Laravel es la siguiente figura. Esta corresponde al archivo *mymainpage.blade.php*. Este es la vista a la que el usuario accederá una vez logueado o registrado. Es en esta vista donde se listarán las publicaciones realizadas por las cuentas que el usuario siga. En caso que el usuario aún no esté siguiendo ninguna cuenta, se mostrará varias opciones de acciones a realizar.

```
resources > views > mymainpage.blade.php > ...
1  @extends('layouts.app')
2  @section('content')
3  <div class="container">
4      <div class="row justify-content-center">
5          <div class="col-md-8">
6              <div class="card-header">
7                  <i class="far fa-smile"></i> Mis cuentas favoritas
8              </div>
9              <br/>
10             <!-- Muestra un mensaje y accesos a búsquedas en caso de no tener aún contenido que visualizar -->
11             @if(count(Auth::User()->followers) == 0)
12                 <div class="card-header">
13                     <i class="fas fa-exclamation-circle"></i> Aún no sigues ninguna cuenta
14                 </div>
15                 <div class="card-body">
16                     <!-- Redirecciona a la página de búsqueda de usuarios -->
17                     <div class="nav-item">
18                         <a class="nav-link" href="{{ route('user.showUsers') }}" title="Buscar">
19                             <i class="fas fa-search"></i> Descubre nuevas cuentas
20                         </a>
21                     </div>
22                     <!-- Redirecciona a la página general -->
23                     <div class="nav-item">
24                         <a class="nav-link" href="{{ route('home') }}" title="Global">
25                             <i class="fas fa-globe"></i> Descubre nuevas publicaciones
26                         </a>
27                     </div>
28                     <!-- Redirecciona a la página de creación de una publicación -->
29                     <div class="nav-item">
30                         <a class="nav-link" href="{{ route('post.create') }}" title="Subir">
31                             <i class="fas fa-arrow-up"></i> Crea una publicación
32                         </a>
33                     </div>
34                 </div>
35             @endif
36             <!--Muestra la Publicaciones de los usuarios seguidos por el usuario identificado-->
37             @foreach ($posts as $post)
38                 @foreach($users as $user)
39                     @if($post->user_id == $user->id)
40                         @include('includes.post', ['post' => $post])
41                     @endif
42                 @endforeach
43             @endforeach
44         </div>
45     </div>
46 </div>
47 @endsection
```

Figura 26: Archivo *mymainpage.blade.php*. Fuente: Elaboración propia.

Rutas

El sistema de rutas en Laravel agiliza la configuración de las mismas, dándonos la posibilidad de configurarlas, apuntando a los métodos necesarios de cada controlador, pasándole parámetros y dándole nombres. De esta forma conseguimos crear URLs limpias y amigables. Para configurar las rutas, Laravel cuenta con un directorio en la raíz llamado *routes*. Dentro se encuentra el archivo *web.php*, mediante el cual se encarga de la configuración de dichas rutas.

En la siguiente figura se puede observar la configuración de las rutas de este proyecto.

```
13
14 Route::get('/', function () {
15     return view('welcome');
16 });
17
18 Auth::routes();
19
20 //Rutas página principal
21 Route::get('/index', 'HomeController@index')->name('home');
22 Route::get('/', 'PostController@index')->name('mymainpage');
23
24 //Rutas de Usuarios
25 Route::get('/config', 'UserController@config')->name('user.config');
26 Route::post('/user/update', 'UserController@update')->name('user.update');
27 Route::get('/user/avatar/{fileName}', 'UserController@getImage')->name('user.avatar');
28 Route::get('/profile/{id}', 'UserController@profile')->name('profile');
29 Route::get('/user/showUsers/{search?}', 'UserController@showUsers')->name('user.showUsers');
30 Route::get('/user/showFollowers/{id}', 'UserController@showFollowers')->name('user.showFollowers');
31 Route::get('/user/showFollowing/{id}', 'UserController@showFollowing')->name('user.showFollowing');
32
33 //Rutas de Seguidores
34 Route::get('/follow/{user_id}', 'FollowerController@follow')->name('follow.save');
35 Route::get('/unfollow/{user_id}', 'FollowerController@unfollow')->name('unfollow.delete');
36
37 //Rutas de publicaciones
38 Route::get('/createPost', 'PostController@create')->name('post.create');
39 Route::post('/post/save', 'PostController@save')->name('post.save');
40 Route::get('/post/image/{fileName}', 'PostController@getImage')->name('post.image');
41 Route::get('/post/delete/{id}', 'PostController@delete')->name('post.delete');
42 Route::get('/post/edit/{id}', 'PostController@edit')->name('post.edit');
43 Route::get('/post/{id}', 'PostController@postdetail')->name('post.postdetail');
44 Route::post('/post/update', 'PostController@update')->name('post.update');
45
46 //Rutas de Comentarios
47 Route::post('/comment/save', 'CommentController@save')->name('comment.save');
48 Route::get('/comment/delete/{id}', 'CommentController@delete')->name('comment.delete');
49
50 //Rutas de Likes
51 Route::get('/like/{post_id}', 'LikeController@like')->name('like.save');
52 Route::get('/dislike/{post_id}', 'LikeController@dislike')->name('like.delete');
53 Route::get('/userLikes', 'LikeController@userLikes')->name('userLikes');
54 Route::get('/whoLikes/{post_id}', 'LikeController@whoLikes')->name('whoLikes');
```

Figura 27: Rutas del proyecto. Fuente: Elaboración propia.

Como se puede observar, las rutas se construyen indicando el método de envío de datos (GET o POST), seguido del formato que tendrá la URL y al método que apuntará. Finalmente, se le da un nombre.

Helpers

En el proyecto se puede ver cuánto tiempo ha transcurrido desde la realización de una publicación, como el mismo desde que un usuario creó su cuenta. Para conseguir este resultado se ha codificado un helper. En Laravel, los helpers son funciones globales de PHP previamente definidas por el framework. Estas pueden ser utilizadas desde cualquier parte del proyecto y, Laravel, cuenta con gran cantidad de ellas. A su vez y como en el caso de este proyecto, se pueden crear helpers propios con funcionalidades específicas.

Dentro del directorio *app* se encuentra la carpeta *helpers*. Es dentro de ella donde se almacenan los helpers. En el caso de este proyecto, el helper no es más que una clase con un método que se encarga de la lógica necesaria para el cálculo del valor de tiempo que se mostrará en el detalle de la publicación como en el del perfil del usuario. Es importante aclarar que el método ha de ser estático.

En la siguiente figura se puede observar el helper codificado.

```
app > helpers > timeFormat.php > ...
1  <?php
2  //Espacio de nombre
3  namespace App\Helpers;
4
5  use Illuminate\Support\Facades\DB;
6
7  //Calcula y formatea el tiempo transcurrido de una publicación o un registro de usuario.
8  class TimeFormat {
9
10     public static function Since($fecha) {
11         //Comprueba si hay una fecha registrada.
12         if ($fecha == null) {
13             return "No hay fecha registrada";
14         }
15         //Recoge el valor de la fecha sobre la que hará el cálculo.
16         $fechaInicio = $fecha;
17         //Da el formato al valor del tiempo e indica un valor inicial.
18         $inicioDesde = $fechaInicio->diff(new \DateTime(date("Y-m-d") . " " . date("H:i:s")));
19         //Realiza los cálculos
20         if ($inicioDesde->y == 0) {
21             if ($inicioDesde->m == 0) {
22                 if ($inicioDesde->d == 0) {
23                     if ($inicioDesde->h == 0) {
24                         if ($inicioDesde->i == 0) {
25                             if ($inicioDesde->s == 0) {
26                                 $resultado = $inicioDesde->s . ' segundos';
27                             } else {
28                                 if ($inicioDesde->s == 1) {
29                                     $resultado = $inicioDesde->s . ' segundo';
30                                 } else {
31                                     $resultado = $inicioDesde->s . ' segundos';
32                                 }
33                             }
34                         } else {
35                             if ($inicioDesde->i == 1) {
36                                 $resultado = $inicioDesde->i . ' minuto';
37                             } else {
38                                 $resultado = $inicioDesde->i . ' minutos';
39                             }
40                         }
41                     } else {
42                         if ($inicioDesde->h == 1) {
43                             $resultado = $inicioDesde->h . ' hora';
44                         } else {
45                             $resultado = $inicioDesde->h . ' horas';
46                         }
47                     }
48                 } else {
49                     if ($inicioDesde->d == 1) {
50                         $resultado = $inicioDesde->d . ' día';
51                     } else {
52                         $resultado = $inicioDesde->d . ' días';
53                     }
54                 }
55             } else {
56                 if ($inicioDesde->m == 1) {
57                     $resultado = $inicioDesde->m . ' mes';
58                 } else {
59                     $resultado = $inicioDesde->m . ' meses';
60                 }
61             }
62         } else {
63             if ($inicioDesde->y == 1) {
64                 $resultado = $inicioDesde->y . ' año';
65             } else {
66                 $resultado = $inicioDesde->y . ' años';
67             }
68         }
69         //Devuelve el resultado
70         return "Hace " . $resultado;
71     }
72 }
```

Figura 28 Helper que consigue tiempo de publicación y registro de usuario. Fuente: Elaboración propia.

Realizada la codificación, se crea un **provider** de este helper. Un provider es el proveedor de servicio del framework en el que se registra el helper codificado, siendo cargado con el resto de dependencias por parte de Laravel. Para crear el provider se utilizó el siguiente comando en artisan:

```
php artisan make:provider timeFormatServiceProvider
```

Esto crea un archivo dentro del directorio *Providers*. El archivo cuenta con un método *register*. Es en éste donde se indica la ruta donde se encuentra el helper codificado.

```
app > Providers > timeFormatServiceProvider.php > ...
1  <?php
2
3  namespace App\Providers;
4
5  use Illuminate\Support\ServiceProvider;
6
7  class timeFormatServiceProvider extends ServiceProvider
8  {
9      /**
10       * Bootstrap services.
11       *
12       * @return void
13       */
14     public function boot()
15     {
16         //
17     }
18
19     /**
20      * Register services.
21      *
22      * @return void
23      */
24     public function register()
25     {
26         require_once app_path() . '/Helpers/timeFormat.php';
27     }
28 }
29 }
```

Figura 29: Provider. Fuente: Elaboración propia.

Finalmente, se configura la carga del provider en Laravel. Dicha configuración se realiza en el archivo *app.php* disponible en la carpeta *config*. Se agrega el provider en el arreglo *providers* y se define el alias del mismo en el arreglo de *alias*es.

```
122     'providers' => [  
123  
124         /*  
125          * Laravel Framework Service Providers...  
126          */  
127         Illuminate\Auth\AuthServiceProvider::class,  
128         Illuminate\Broadcasting\BroadcastServiceProvider::class,  
129         Illuminate\Bus\BusServiceProvider::class,  
130         Illuminate\Cache\CacheServiceProvider::class,  
131         Illuminate\Foundation\Providers\ConsoleSupportServiceProvider::class,  
132         Illuminate\Cookie\CookieServiceProvider::class,  
133         Illuminate\Database\DatabaseServiceProvider::class,  
134         Illuminate\Encryption\EncryptionServiceProvider::class,  
135         Illuminate\Filesystem\FilesystemServiceProvider::class,  
136         Illuminate\Foundation\Providers\FoundationServiceProvider::class,  
137         Illuminate\Hashing\HashServiceProvider::class,  
138         Illuminate\Mail\MailServiceProvider::class,  
139         Illuminate\Notifications\NotificationServiceProvider::class,  
140         Illuminate\Pagination\PaginationServiceProvider::class,  
141         Illuminate\Pipeline\PipelineServiceProvider::class,  
142         Illuminate\Queue\QueueServiceProvider::class,  
143         Illuminate\Redis\RedisServiceProvider::class,  
144         Illuminate\Auth\Passwords>PasswordResetServiceProvider::class,  
145         Illuminate\Session\SessionServiceProvider::class,  
146         Illuminate\Translation\TranslationServiceProvider::class,  
147         Illuminate\Validation\ValidationServiceProvider::class,  
148         Illuminate\View\ViewServiceProvider::class,  
149  
150         /*  
151          * Package Service Providers...  
152          */  
153  
154         /*  
155          * Application Service Providers...  
156          */  
157  
158         App\Providers\AppServiceProvider::class,  
159         App\Providers\AuthServiceProvider::class,  
160         // App\Providers\BroadcastServiceProvider::class,  
161         App\Providers\EventServiceProvider::class,  
162         App\Providers\RouteServiceProvider::class,  
163         App\Providers\timeFormatServiceProvider::class  
164     ],  
165
```

Figura 30: Arreglo providers. Fuente: Elaboración propia.

```
178 'aliases' => [  
179  
180     'App' => Illuminate\Support\Facades\App::class,  
181     'Artisan' => Illuminate\Support\Facades\Artisan::class,  
182     'Auth' => Illuminate\Support\Facades\Auth::class,  
183     'Blade' => Illuminate\Support\Facades\Blade::class,  
184     'Broadcast' => Illuminate\Support\Facades\Broadcast::class,  
185     'Bus' => Illuminate\Support\Facades\Bus::class,  
186     'Cache' => Illuminate\Support\Facades\Cache::class,  
187     'Config' => Illuminate\Support\Facades\Config::class,  
188     'Cookie' => Illuminate\Support\Facades\Cookie::class,  
189     'Crypt' => Illuminate\Support\Facades\Crypt::class,  
190     'DB' => Illuminate\Support\Facades\DB::class,  
191     'Eloquent' => Illuminate\Database\Eloquent\Model::class,  
192     'Event' => Illuminate\Support\Facades\Event::class,  
193     'File' => Illuminate\Support\Facades\File::class,  
194     'Gate' => Illuminate\Support\Facades\Gate::class,  
195     'Hash' => Illuminate\Support\Facades\Hash::class,  
196     'Lang' => Illuminate\Support\Facades\Lang::class,  
197     'Log' => Illuminate\Support\Facades\Log::class,  
198     'Mail' => Illuminate\Support\Facades\Mail::class,  
199     'Notification' => Illuminate\Support\Facades\Notification::class,  
200     'Password' => Illuminate\Support\Facades>Password::class,  
201     'Queue' => Illuminate\Support\Facades\Queue::class,  
202     'Redirect' => Illuminate\Support\Facades\Redirect::class,  
203     'Redis' => Illuminate\Support\Facades\Redis::class,  
204     'Request' => Illuminate\Support\Facades\Request::class,  
205     'Response' => Illuminate\Support\Facades\Response::class,  
206     'Route' => Illuminate\Support\Facades\Route::class,  
207     'Schema' => Illuminate\Support\Facades\Schema::class,  
208     'Session' => Illuminate\Support\Facades\Session::class,  
209     'Storage' => Illuminate\Support\Facades\Storage::class,  
210     'URL' => Illuminate\Support\Facades\URL::class,  
211     'Validator' => Illuminate\Support\Facades\Validator::class,  
212     'View' => Illuminate\Support\Facades\View::class,  
213     'TimeFormat' => App\Helpers\TimeFormat::class  
214  
215 ],
```

Figura 31: Arreglo aliases. Fuente: Elaboración propia.

De esta forma queda codificado, generado y configurado el helper.

Scripts

El proyecto cuenta con algunas funcionalidades que son realizadas sin la necesidad de recargar la página, como lo son el sistema de likes y de seguir cuentas. Para ellas se ha hecho uso de scripts que realizan estas funciones y trabajan con la base de datos en segundo plano utilizando AJAX.

Para el sistema de likes, se codificó un script que cambia el color del icono y guarda o elimina el registro de like en la base de datos de manera automática. También, en la pantalla de detalle de la publicación, actualiza el conteo de likes de la misma de manera instantánea. Para esto, el script identifica si ya hay un registro de like sobre la publicación, leyendo la clase del mismo. Si lo hay, cambia la clase a *dislike*, cambia el icono y elimina el registro de la base de datos por medio de AJAX. A su vez, descuenta en uno el valor total de likes de la publicación, imprimiendo el nuevo conteo obtenido de la respuesta que devuelve el método *dislike* del controlador. En caso de no existir un like previo, el script cambia la clase del icono a *like*, cambia el icono al color rojo, y realiza el registro del like en la base de datos, al mismo tiempo que actualiza el conteo total de likes y lo imprime en el detalle de la misma. En la siguiente figura se puede apreciar dicho script.

```
2 //Define la url del proyecto
3 var url = 'http://proyectofinalaravel.com/';
4
5 window.addEventListener("load", function(){
6
7     //indica que, al pasar el cursor sobre los iconos, se muestre un puntero.
8     $('.icons-like .like').css('cursor', 'pointer');
9     $('.icons-like .dislike').css('cursor', 'pointer');
10
11     //Llama a la función.
12     likes()
13
14     //Realiza el registro de un like y dislike mediante una petición Ajax
15     function likes(){
16
17         $('.icons-like').off().on('click', function(){
18             //Comprueba que el icono tenga la clase like
19             var likeExist = $(this).children().hasClass('like')
20             //Cambia el nombre de la clase y elimina el registro de like de la base de datos.
21             if(likeExist){
22                 $(this).children().removeClass('fas like').addClass('far dislike')
23                 console.log('dislike')
24                 $.ajax({
25                     url: url + 'dislike/' + $(this).data('id'),
26                     type: 'GET',
27                     success: function(response){
28                         //Actualiza el conteo de likes.
29                         $('.countLikes').text(response.count + ' Me gustas');
30                         console.log(response.count);
31                     }
32                 });
33             }
34             //Cambia el nombre de la clase y realiza el registro de like en la base de datos.
35             else{
36                 $(this).children().removeClass('far dislike').addClass('fas like')
37                 console.log('like')
38                 $.ajax({
39                     url: url + 'like/' + $(this).data('id'),
40                     type: 'GET',
41                     success: function(response){
42                         //Actualiza el conteo de likes.
43                         $('.countLikes').text(response.count + ' Me gustas');
44                         console.log(response.count);
45                     }
46                 });
47             }
48             //Llama a la función.
49             likes()
50         });
51     }
52 }
```

Figura 32: Script sistema de likes. Fuente: Elaboración propia.

El script que trabaja el sistema de seguimiento es idéntico al anterior. La única diferencia está en el nombre de las clases de los iconos y que éste no necesita actualizar ningún conteo.

Cabe destacar que el archivo con los scripts se aloja en el directorio *public*, dentro de la carpeta *js*, con nombre ***script.js***.

Maquetado del proyecto

Completa la codificación del proyecto, se lleva a cabo la maquetación. Si bien gran parte de la misma se ha ido realizando a la par de la codificación, otra parte se realizó una vez completada la misma. Para el maquetado se hizo uso de Bootstrap, aprovechando la sencillez en la implementación de su sistema de tarjetas. Estas son contenedores estructurados en cabecera de contenido, cuerpo de contenido y pie del mismo. De esta forma se estructuro prácticamente todo el contenido del proyecto. Luego, para los estilos del proyecto se utilizó CSS, creando un archivo llamado ***style.css*** con toda su codificación. Este archivo se aloja en la carpeta *public*, dentro del directorio *css*. Con este se ha trabajado, mayormente, los tamaños de fuentes, colores de las mismas, tamaño de los iconos, de las imágenes de perfil y algunos comportamientos de, por ejemplo, los links, los campos de textos, etc.

6.-DESPLIEGUE Y PRUEBAS

6.1.-PLAN DE PRUEBA

Las pruebas realizadas han sido de acuerdo a cada caso de uso. En la siguiente tabla quedan representadas.

Nº	ESPECIFICACIÓN DE PRUEBAS
1	<p>Objetivo probado: Registro de usuario.</p> <p>Requisitos probados: Permitir el registro de usuario y el autenticado del mismo.</p> <p>Pruebas a realizar: Se realiza el registro de un usuario, ingresando su nombre, apellido, nickname, email, imagen de perfil, password y la confirmación del password.</p>
2	<p>Objetivo probado: Registro de usuario.</p> <p>Requisitos probados: No permitir registro si falta alguno o todos los datos requeridos.</p> <p>Pruebas a realizar: Se intenta registro sin la introducción de uno/varios/ningún dato. Registro no permitido.</p>

3	<p>Objetivo probado: Registro de usuario.</p> <p>Requisitos probados: No permitir registro si la imagen de perfil no es un archivo de imagen.</p> <p>Pruebas a realizar: Se intenta registro introduciendo un tipo de archivo no permitido. Registro no permitido.</p>
4	<p>Objetivo probado: Registro de usuario.</p> <p>Requisitos probados: No permitir registro si nickname y/o email son datos duplicados.</p> <p>Pruebas a realizar: Se intenta registro introduciendo un nickname y/o email que ya tienen un registro previo. Registro no permitido.</p>
5	<p>Objetivo probado: Registro de usuario.</p> <p>Requisitos probados: No permitir registro si el formato del email no es correcto.</p> <p>Pruebas a realizar: Se intenta registro introduciendo un email con un formato erróneo. Registro no permitido.</p>
6	<p>Objetivo probado: Registro de usuario.</p> <p>Requisitos probados: No permitir registro si el password no coincide con la confirmación del mismo.</p> <p>Pruebas a realizar: Se intenta registro introduciendo una confirmación de password incorrecta. Registro no permitido.</p>
7	<p>Objetivo probado: Registro de usuario.</p> <p>Requisitos probados: No permitir registro si el password es inferior a 6 caracteres.</p> <p>Pruebas a realizar: Se intenta registro introduciendo password inferior a 6 caracteres. Registro no permitido.</p>
8	<p>Objetivo probado: Autenticado del usuario.</p> <p>Requisitos probados: Login del usuario registrado.</p> <p>Pruebas a realizar: Se realiza el autenticado del usuario por medio del email y el password.</p>
9	<p>Objetivo probado: Autenticado del usuario.</p> <p>Requisitos probados: No permitir el login si falta email y/o password.</p> <p>Pruebas a realizar: Se intenta el login sin la introducción de uno o ambos datos. Autenticación no permitida.</p>

10	<p>Objetivo probado: Modificación de datos del usuario.</p> <p>Requisitos probados: Modificación de los datos del usuario.</p> <p>Pruebas a realizar: Se realiza la modificación de los datos del usuario.</p>
11	<p>Objetivo probado: Modificación de datos del usuario.</p> <p>Requisitos probados: No permitir la modificación de los datos del usuario si el nickname y/o email ya tienen un registro previo.</p> <p>Pruebas a realizar: Se intenta modificar los datos del usuario introduciendo un nickname y/o email que ya tienen un registro previo. Modificación no permitida.</p>
12	<p>Objetivo probado: Modificación de datos del usuario.</p> <p>Requisitos probados: No permitir la modificación de los datos del usuario si el formato del email no es correcto.</p> <p>Pruebas a realizar: Se intenta modificar los datos del usuario introduciendo un email con un formato erróneo. Modificación no permitida.</p>
13	<p>Objetivo probado: Modificación de datos del usuario.</p> <p>Requisitos probados: No permitir la modificación de los datos del usuario si la imagen de perfil no es un archivo de imagen.</p> <p>Pruebas a realizar: Se intenta modificar los datos del usuario introduciendo un tipo de archivo no permitido. Modificación no permitida.</p>
13	<p>Objetivo probado: Crear publicación.</p> <p>Requisitos probados: Creación de una publicación introduciendo título, imagen, autor, editorial, páginas y descripción.</p> <p>Pruebas a realizar: Se realiza la creación de una publicación.</p>
14	<p>Objetivo probado: Crear publicación.</p> <p>Requisitos probados: No permitir la creación de una publicación si falta introducir uno/varios/todos los datos.</p> <p>Pruebas a realizar: Se intenta crear una publicación sin la introducción de uno/varios/ningún dato. Creación de la publicación no permitida.</p>
15	<p>Objetivo probado: Crear publicación.</p> <p>Requisitos probados: No permitir la creación de una publicación si la imagen de la misma no es un archivo de imagen.</p> <p>Pruebas a realizar: Se intenta crear una publicación sin la introducción un tipo de archivo no permitido. Creación de la publicación no permitida.</p>

16	<p>Objetivo probado: Modificación de una publicación.</p> <p>Requisitos probados: Modificación de una publicación introduciendo título, imagen, autor, editorial, páginas y descripción.</p> <p>Pruebas a realizar: Se realiza la modificación de una publicación.</p>
17	<p>Objetivo probado: Modificación de una publicación.</p> <p>Requisitos probados: No permitir la modificación de una publicación si la imagen de la misma no es un archivo de imagen.</p> <p>Pruebas a realizar: Se intenta modificar una publicación sin la introducción un tipo de archivo no permitido. Creación de la publicación no permitida.</p>
18	<p>Objetivo probado: Eliminar publicación.</p> <p>Requisitos probados: Eliminación de una publicación.</p> <p>Pruebas a realizar: Se realiza la eliminación de una publicación.</p>
19	<p>Objetivo probado: Realizar comentario en una publicación.</p> <p>Requisitos probados: Realizar un comentario en una publicación.</p> <p>Pruebas a realizar: Se realiza un comentario en una publicación. Comentario registrado.</p>
20	<p>Objetivo probado: Dar like en una publicación.</p> <p>Requisitos probados: Registrar un like en una publicación.</p> <p>Pruebas a realizar: Se cliquea en el icono de like a una publicación. Like registrado y cambia de color del icono.</p>
21	<p>Objetivo probado: Dar like en una publicación.</p> <p>Requisitos probados: Si se registrar un like en una publicación, en la página de su detalle se actualizará el conteo total de los mismos.</p> <p>Pruebas a realizar: Se cliquea en el icono de like a una publicación. Conteo actualizado.</p>
22	<p>Objetivo probado: Eliminar like de una publicación.</p> <p>Requisitos probados: Eliminar un registro de like en una publicación.</p> <p>Pruebas a realizar: Se cliquea en el icono de dislike en una publicación. Like eliminado y cambia de color del icono.</p>
23	<p>Objetivo probado: Eliminar like de una publicación.</p> <p>Requisitos probados: Si se elimina el registro de un like en una publicación, en la página de su detalle se actualizará el conteo total de los mismos.</p> <p>Pruebas a realizar: Se cliquea en el icono de dislike en una publicación. Conteo actualizado.</p>

24	<p>Objetivo probado: Buscar usuarios.</p> <p>Requisitos probados: Búsqueda de usuarios introduciendo un carácter, una cadena de caracteres o nada.</p> <p>Pruebas a realizar: Se introduce un carácter/ una cadena de caracteres/ ninguno y se listan las coincidencias.</p>
25	<p>Objetivo probado: Seguir usuarios.</p> <p>Requisitos probados: Guardar el registro de seguimiento de un usuario.</p> <p>Pruebas a realizar: Se cliquea en el icono de seguir un usuario. Registro guardado.</p>
26	<p>Objetivo probado: Seguir usuarios.</p> <p>Requisitos probados: Si se realiza el registro de seguimiento de un usuario se actualiza el icono y su color.</p> <p>Pruebas a realizar: Se cliquea en el icono de seguir un usuario. Registro guardado y cambio de icono y color.</p>
27	<p>Objetivo probado: Dejar de seguir usuarios.</p> <p>Requisitos probados: Eliminar el registro de seguimiento de un usuario.</p> <p>Pruebas a realizar: Se cliquea en el icono de dejar de seguir un usuario. Registro eliminado.</p>
28	<p>Objetivo probado: Dejar de seguir usuarios.</p> <p>Requisitos probados: Si se elimina el registro de seguimiento de un usuario se actualiza el icono y su color.</p> <p>Pruebas a realizar: Se cliquea en el icono de dejar de seguir un usuario. Registro eliminado y cambio de icono y color.</p>

Tabla 1: Pruebas basadas en pruebas de caja negra. Fuente: Elaboración propia.

7.-CONCLUSIONES

7.1.-OBJETIVOS ALCANZADOS

Recapitulando sobre los objetivos propuestos al inicio del proyecto y comparándolos con los alcanzados al final del mismo se puede deducir que han sido cumplidos en su totalidad. Se ha conseguido construir una aplicación web completa, donde un usuario es capaz de crear una cuenta, ingresar a la plataforma con los datos del registro, crear publicaciones, modificarlas o eliminarlas, visualizar las de otros usuarios de la plataforma, indicar que le gusta una publicación, realizar comentarios en ellas, buscar usuarios, seguir o dejar de seguir otras cuentas, etc.

7.2.-CONCLUSIONES DEL TRABAJO

Una vez finalizado el proyecto y asentadas las distintas fases del mismo he podido recoger algunas conclusiones sobre el trabajo realizado. Es probable que lo más complicado del proyecto haya sido conseguir una idea realizable con los conocimientos obtenidos del ciclo. Sobre todo, lograr que la misma sea atractiva e interesante, tanto para mí, como realizador del trabajo, como para quien tenga la responsabilidad de examinarlo. Otro punto sobre el cual he debido pensar bastante, sopesando las ventajas y desventajas que podría acarrear, ha sido la elección de las tecnologías a utilizar. Entre ellas me planteé algunas que no habían sido estudiadas en el ciclo, por el afán de aprenderlas poniéndolas en práctica con la realización de este proyecto. Seguido, con una idea base y la elección de las tecnologías a utilizar, surgió el siguiente reto. Este corresponde al análisis del proyecto. Terminado el trabajo puedo decir que el hacer un correcto análisis del mismo ha sido una pieza fundamental para su realización. El conseguir una estructuración sólida sobre la cual se construiría el proyecto, como puede ser la base de datos, fue una de las fases que más preocupación me generó, puesto que sabía que de ello dependería el lograr los objetivos propuestos con éxito. Ya con la base definida, solo quedo comenzar a codificar.

La codificación del proyecto fue otro reto, sobre todo por la utilización de tecnologías que no conocía previamente. Primero debí familiarizarme con el uso del framework Laravel. Pese a las grandes ventajas que me proporciono su utilización, me vi obligado a invertir mucho tiempo en su estudio y en entender como trabajar con él. El haber estado leyendo, viendo videos, experimentando y haciendo pruebas con este framework ha resultado en un aumento significativo en conocimientos y experiencia sobre una herramienta nueva y muy interesante. Pese a que al comenzar el proyecto me encontré con la incertidumbre de si sería capaz de realizarlo con una herramienta que hasta ese momento desconocía, una vez finalizado me siento muy contento por haberme decidido por su uso. De igual manera resulto mi experiencia con la utilización de Bootstrap. Aunque en este caso, el uso que hice de esta herramienta ha sido bastante básico.

Siguiendo con lo que respecta a la codificación, otra dificultad con la que me encontré fue en el desarrollo del sistema de likes y se seguir usuarios. Para esto me había propuesto hacerlo sin la necesidad de refrescar la pantalla, con un trabajo en segundo plano. Es de esta manera que trabaje con un script desde donde se realizaba la comunicación con la base de datos con AJAX. Puesto que, si bien es algo que se estudia en el ciclo, no tenía la suficiente experiencia en el uso del mismo. Finalmente, luego de varias lecturas y pruebas, logre conseguir el objetivo deseado. En lo personal, la realización de este proyecto ha supuesto la adquisición de conocimientos nuevos y una mejor comprensión de otros previamente estudiados, ganando experiencia y confianza para futuros trabajos.

7.3.-VÍAS FUTURAS

Esta aplicación web es totalmente escalable, por lo cual se pueden ampliar enormemente sus funcionalidades. Algunas de ellas podrían ser las de agregar la opción de identificar una publicación con una categoría. A su vez, configurar un sistema de roles. De esta manera tendríamos usuarios administradores encargados de, por ejemplo, definir nuevas categorías de libros. Teniendo la opción de definir una publicación dentro de una categoría, se podría implementar un sistema de búsquedas de libros. De esta forma, un usuario podría buscar un libro realizando un filtro sobre el título, autor, editorial y categoría. Estos filtros también podrían utilizarse en las distintas páginas de la plataforma, permitiendo listar aquellas publicaciones que se ajusten al gusto del usuario. Otra ampliación del proyecto podría ser la de dar la posibilidad de enviar y recibir mensajes privados entre usuarios, aumentando enormemente la interacción entre ellos. Si bien el proyecto es básicamente una red social dedicada a los libros, con una ampliación de la base de datos y otras modificaciones de codificación se podría fácilmente convertir en una plataforma social de intercambio de libros. Así, quien realice la publicación sobre un libro podría indicar si lo quiere regalar. De esta manera se abriría un mundo de nuevas funcionalidades y un camino nuevo por recorrer.

8.-GLOSARIO

- **AJAX:** Asynchronous JavaScript and XML. Es un grupo de técnicas que permiten el funcionamiento de manera asíncrona de una web, realizando el procesamiento de toda solicitud de datos al servidor en segundo plano.
- **Artisan:** Interfaz de líneas de comandos.
- **Blade:** Sistema de plantillas de Laravel.
- **Bootstrap:** Librería de código abierto utilizada para el desarrollo de aplicaciones web.
- **Composer:** Gestionador de paquetes de PHP.
- **CSS:** Cascading Style Sheets. Lenguaje que define el estilo de diseño de un documento estructurado por un lenguaje de marcado.
- **Draw.io:** Plataforma web para la creación de distintos tipos de diagramas.
- **Eloquent:** ORM de Laravel.
- **Font Awesome:** Plataforma web que provee gran variedad de tipografías e iconos.
- **Framework:** Marco de trabajo estructurado para la agilización de procesos y el asegurado de buenas prácticas junto a la consistencia del código.
- **GIT:** Software de control de versiones.
- **GitHub:** Aplicación web de desarrollo colaborativo.
- **HTML:** HyperText Markup Language. Lenguaje de marcado utilizado en el desarrollo web.
- **IDE:** Entorno de desarrollo integrado. Software para el desarrollo de aplicaciones.
- **JavaScript:** Lenguaje de programación del lado del cliente.
- **jQuery:** Biblioteca de JavaScript.
- **JSON:** JavaScript Object Notation. Formato de texto utilizado para el intercambio de datos.
- **Laravel:** Framework de PHP.
- **MVC:** Model View Controller. Sistema de desarrollo de software que organiza los datos, la interfaz y la lógica de control en componentes separados.
- **MySQL:** Sistema gestor de base de datos relacional.
- **ORM:** Object Relational Mapping. Modelo que mapea la estructura de una base de datos relacional.
- **PHP:** Lenguaje de programación del lado del servidor.
- **PhpMyAdmin:** Interfaz gráfica para la administración del sistema gestor de base de datos MySQL.
- **SQL:** Structured Query Language. Lenguaje para la administración y recuperación de datos.
- **Visual Studio Code:** Editor de código desarrollado por Microsoft.

9.-BIBLIOGRAFÍA

Libros

- Entornos de Desarrollo. Editorial Garceta.
- Lenguajes de Marcas. Editorial Garceta.
- Base de Datos. Editorial Garceta.
- Desarrollo Web en entorno servidor. Editorial Garceta.
- Desarrollo Web en entorno cliente con JavaScript. Editorial Garceta.
- Diseño de Interfaces Web. Editorial Garceta.

Páginas web

- <https://fontawesome.com/>
- <https://developer.mozilla.org/en-US/>
- <https://getcomposer.org/>
- <https://codea.app/php/generar-json-desde-una-consulta-mysql>
- <https://davidwalsh.name/detect-ajax>
- <https://jorgesanchez.net/manuales/gbd/disenio-logico-relacional.html>
- https://ikastaroak.birt.eus/edu/argitalpen/backupa/20200331/1920k/es/DAMDAW/BD/BD03/es_DAMDAW_BD03_Contenidos/website_24_relaciones_reflexivas.html
- <https://josejuansanchez.org/bd/unidad-03-teoria/index.html#relaciones-con-cardinalidad-nn>
- <https://platzi.com/blog/middlewares-laravel/>
- <https://code.tutsplus.com/es/tutorials/how-to-create-a-laravel-helper--cms-28537>
- <https://laracasts.com/discuss/channels/eloquent/eloquent-sync-associate?page=1>
- https://www.youtube.com/watch?v=efocZNA5T_A&list=PLhCiuvlix-rT96yLQTTTr3-eTZl6sbhVte
- <https://www.youtube.com/watch?v=4Mdk7voadzQ>
- https://www.youtube.com/playlist?list=PLU8oAlHdN5Bk-qkvjER90g2c_jVmpAHBh
- <https://www.youtube.com/watch?v=LWfYTAEp8&t=324s>
- <https://soporte.dongee.com/es/articles/3299207-desplegar-laravel-en-cpanel>
- <https://www.nigmacode.com/laravel/Subir-proyecto-laravel-a-hosting>
- <https://www.pildorasinformaticas.es/course/php-mysql/>
- <https://www.pildorasinformaticas.es/course/php-mysql/php-mysql-modulo-2/>
- <https://www.pildorasinformaticas.es/course/javascript-desde-0/curriculum/>
- <https://www.pildorasinformaticas.es/course/html-5/>
- <https://www.pildorasinformaticas.es/course/css-avanzado-desde-0/>

-
- <https://www.w3schools.com/>
 - <https://es.stackoverflow.com/>
 - <https://victorroblesweb.es/>
 - <https://laravel.com/>
 - <https://www.php.net/manual/es/index.php>
 - <https://api.jquery.com/>
 - <https://github.com/>

10.-ANEXOS

10.1.-MANUAL DE INSTALACIÓN

Pese a que el proyecto se realizó al completo en local, una vez probado y finalizado se ha instalado en un servidor. No obstante, se describe el proceso de instalación del proyecto en el entorno local y el procedimiento seguido para el despliegue en el servidor.

Instalación en entorno local

Para poder utilizar el proyecto en un entorno local se debe contar con un servidor web, un intérprete para el lenguaje PHP y un sistema de gestión de base de datos. Si bien es posible instalar cada componente por separado, la opción más sencilla es la de utilizar el paquete de software **XAMPP**. Este software engloba las tecnologías anteriormente mencionadas en un único entorno, haciendo más ágil su gestión.

Instalación de XAMPP

Se debe descargar el software desde su página oficial:

<https://www.apachefriends.org/es/download.html>

Apache Friends

Descargar Complementos Alojamiento Comunidad Acerca de Buscar. Buscar ES

Descargar

XAMPP es una distribución de Apache fácil de instalar que contiene MariaDB, PHP y Perl. Simplemente descarga y ejecuta el instalador. ¡Es así de fácil!

XAMPP para Windows 7.2.34, 7.3.23 & 7.4.11

Versión	¿Qué está incluido?	Suma de comprobación	Descargar (64 bit)	Tamaño
7.2.34 / PHP 7.2.34	¿Qué está incluido?	md5 sha1	Descargar (64 bit)	153 Mb
7.3.23 / PHP 7.3.23	¿Qué está incluido?	md5 sha1	Descargar (64 bit)	153 Mb
7.4.11 / PHP 7.4.11	¿Qué está incluido?	md5 sha1	Descargar (64 bit)	154 Mb

Requisitos Complementos Más Descargas »

Windows XP or 2003 are not supported. You can download a compatible version of XAMPP for these platforms here.

XAMPP para Linux 7.2.34, 7.3.23 & 7.4.11

Documentación/FAQs

No hay un manual para XAMPP. Escribimos la documentación en forma de preguntas frecuentes (FAQs). ¿Tienes una pregunta que no está respondida? Prueba los Foros o Stack Overflow.

- Linux Preguntas frecuentes
- Windows Preguntas frecuentes
- OS X Preguntas frecuentes
- OS X XAMPP-VM Preguntas frecuentes

Complementos

Bitnami proporciona herramientas gratuitas para instalar Drupal, Joomla!, WordPress y muchas otras aplicaciones populares en XAMPP. Visita Bitnami XAMPP o accede a la

Figura 33: Descarga de XAMPP. Fuente: Elaboración propia.

Se encontrarán varias opciones de descarga, tanto para Windows como para Linux o Mac. En el caso de Windows, se ha de seleccionar la última versión.

Una vez descargado, se ejecuta al archivo de instalación. Windows preguntará si deseamos realizar cambios en el equipo, a lo que se indica que sí.

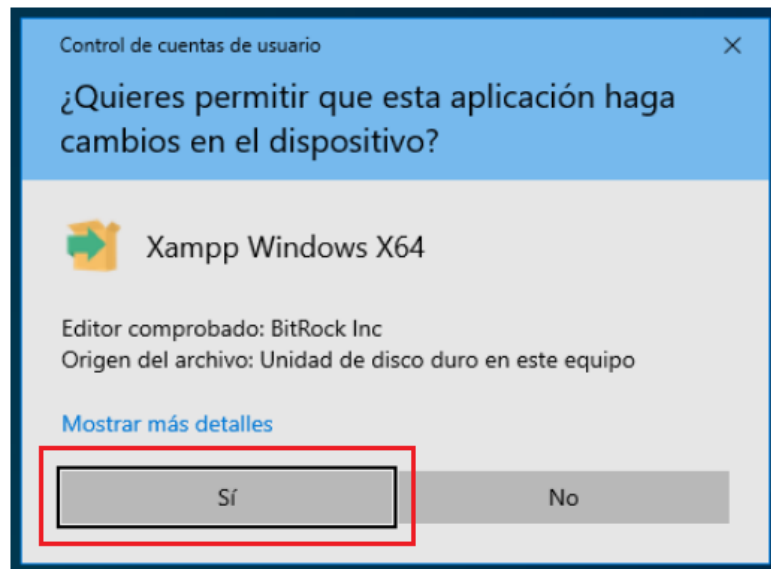


Figura 34: Instalación de XAMPP. Fuente: Elaboración propia.

Seguido, comienza la instalación de XAMPP. El instalador lanzará una primera pantalla indicando que comenzará la instalación. Se debe presionar el botón de “Next” para continuar con ésta.

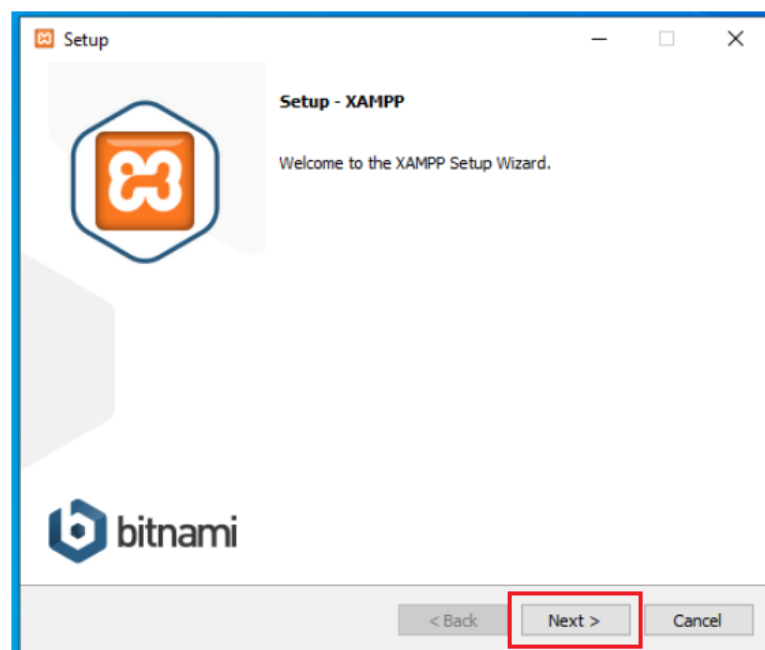


Figura 35: Instalación de XAMPP. Setup. Fuente: Elaboración propia.

Luego, seleccionar los componentes que se quieren instalar. Se han de seleccionar los de Apache, MySQL PHP y phpMyAdmin. Se continúa la instalación presionando en el botón “Next”.

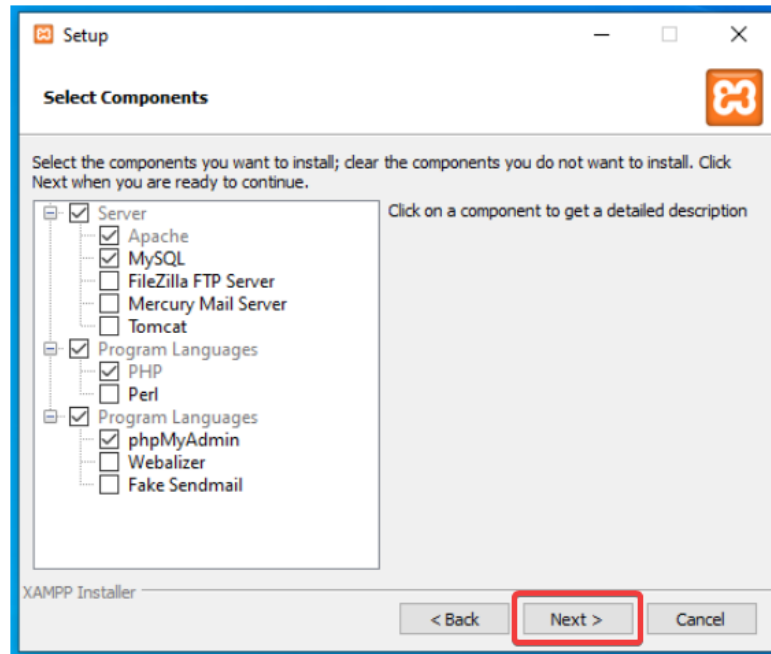


Figura 36: Instalación de XAMPP. Selección de componentes. Fuente: Elaboración propia.

Seguido, se debe indicar la ruta donde se instalará el paquete. En este caso se deja la que indica el instalador por defecto. Se continúa presionando en el botón “Next”.

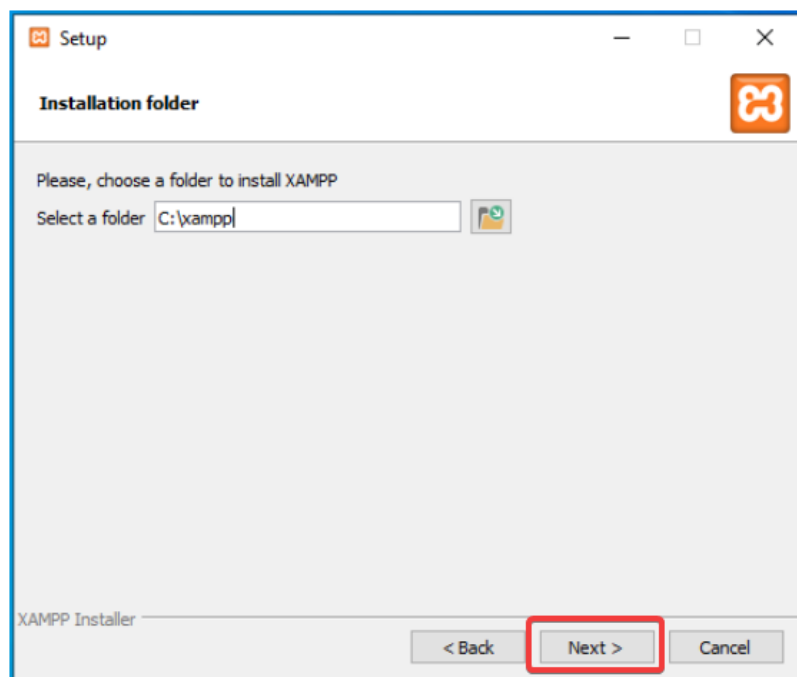


Figura 37: Instalación de XAMPP. Ruta de instalación. Fuente: Elaboración propia.

La instalación continua con una pantalla donde se pregunta si se quiere aprender más sobre XAMPP, en la cual se ha de indicar que no, desmarcando la casilla correspondiente. Luego se debe presionar en el botón de “Next” para seguir con el proceso.

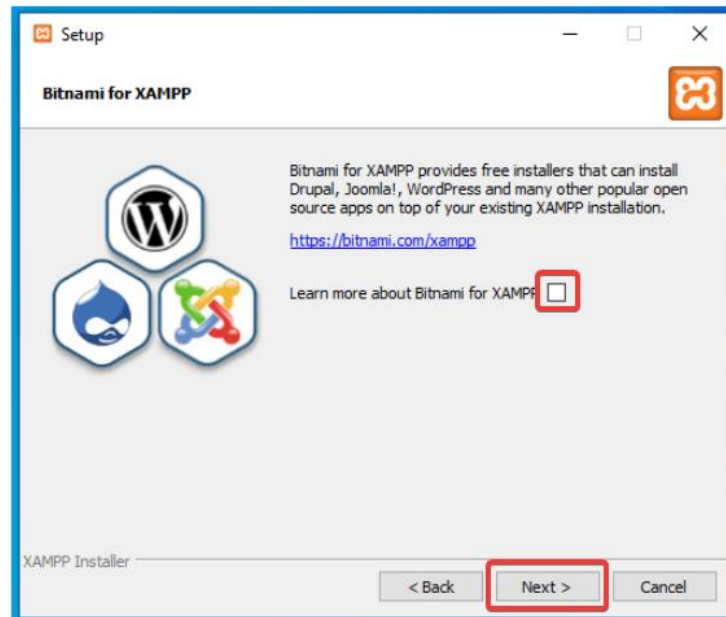


Figura 38: Instalación de XAMPP. Últimos pasos de la configuración. Fuente: Elaboración propia.

Finalmente, se muestra una última ventana indicando que la configuración está terminada y el software está listo para ser instalado. Se debe presionar el botón “Next” para comenzar con el proceso.

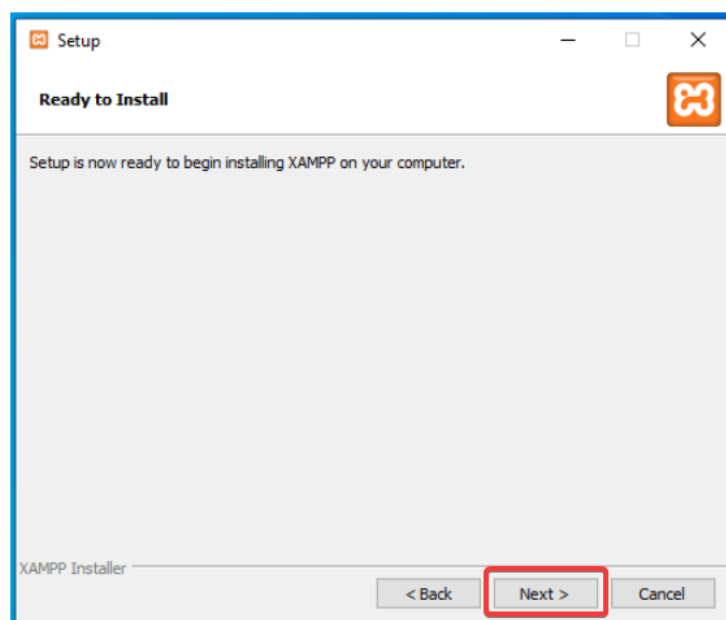


Figura 39: Instalación de XAMPP. Instalación preparada. Fuente: Elaboración propia.

Seguido, comienza la instalación.

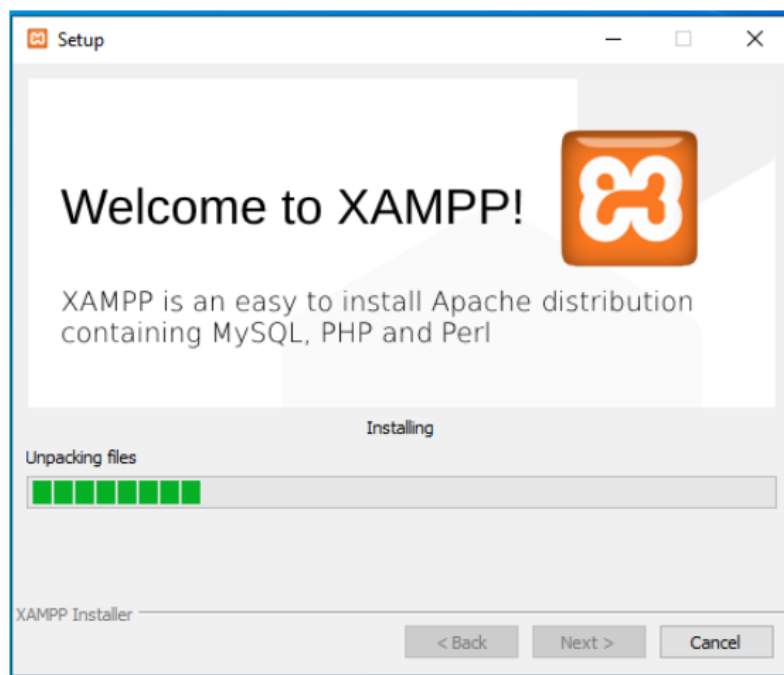


Figura 40: Instalación de XAMPP. Inicio de instalación. Fuente: Elaboración propia.

Finalizada la instalación se podrá lanzar el programa marcando la casilla correspondiente en la ventana del instalador y presionando el botón “Finish”.

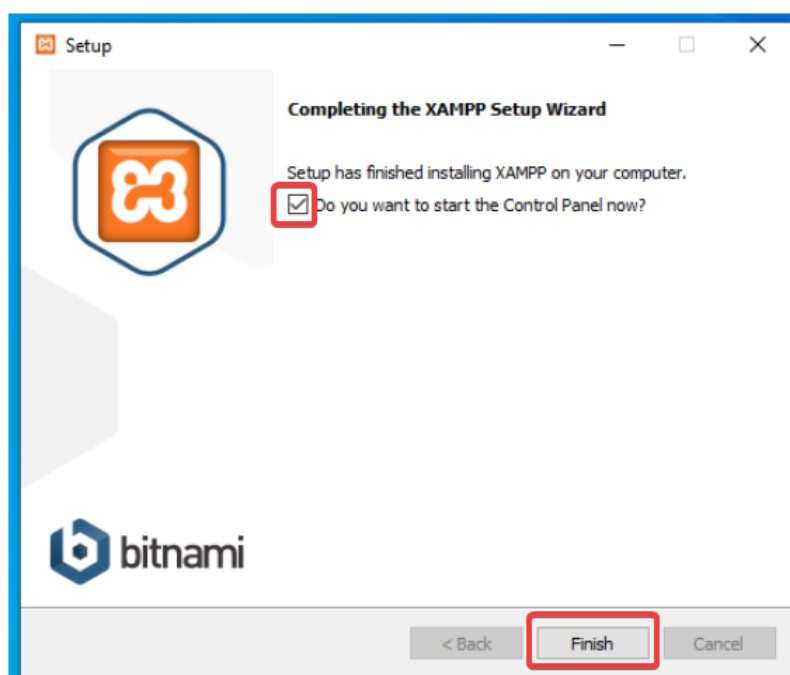


Figura 41: Instalación de XAMPP. Finalización de la instalación. Fuente: Elaboración propia.

Una vez instalado, se lanza el panel de control de XAMPP mostrando la siguiente ventana.

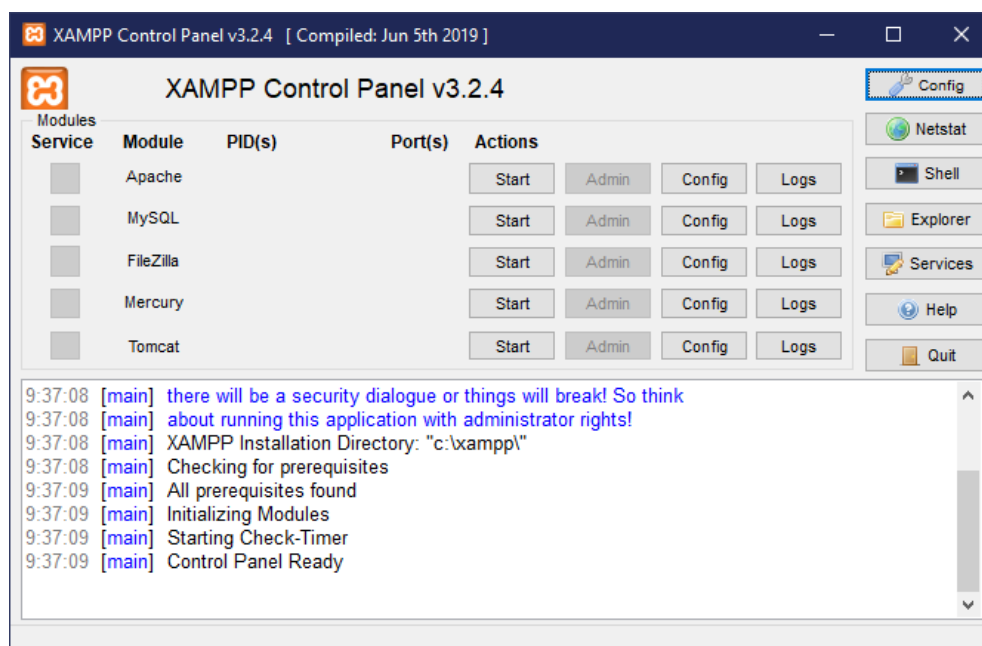


Figura 42: Instalación de XAMPP. Panel de control. Fuente: Elaboración propia.

Para poner en funcionamiento los servicios necesarios para la ejecución del proyecto hay que presionar en el botón de “Start” correspondientes a Apache y MySQL.

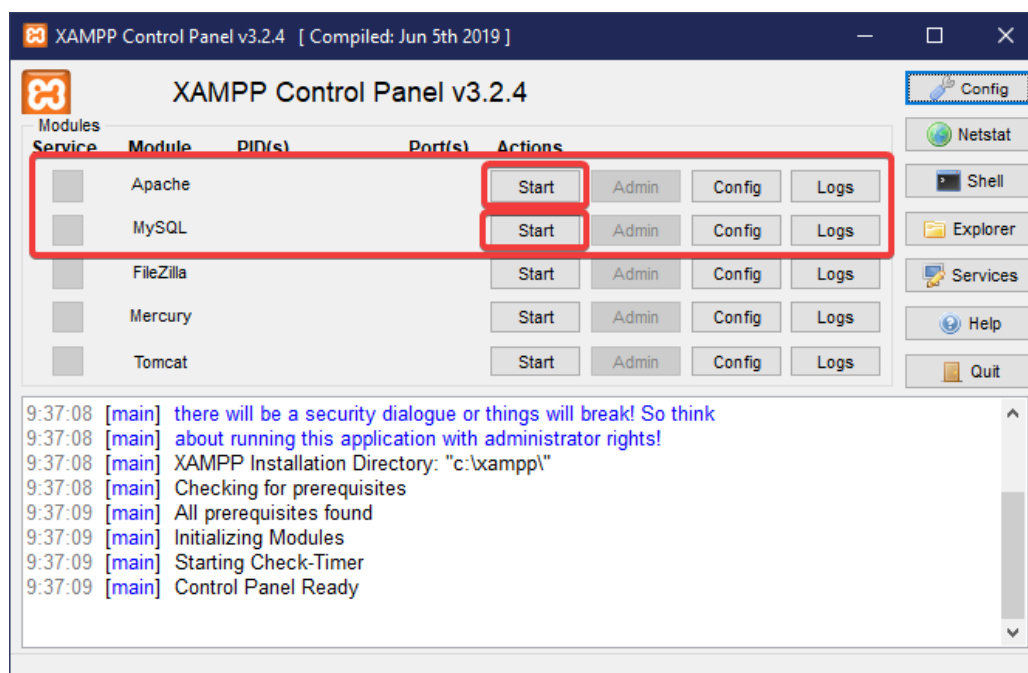


Figura 43: Puesta en marcha de los servicios. Fuente: Elaboración propia.

Una vez iniciados, se mostrarán por pantalla los servicios en funcionamiento y los puertos que utilizan. Para detenerlos sólo habrá que pulsar los botones de “Stop”.

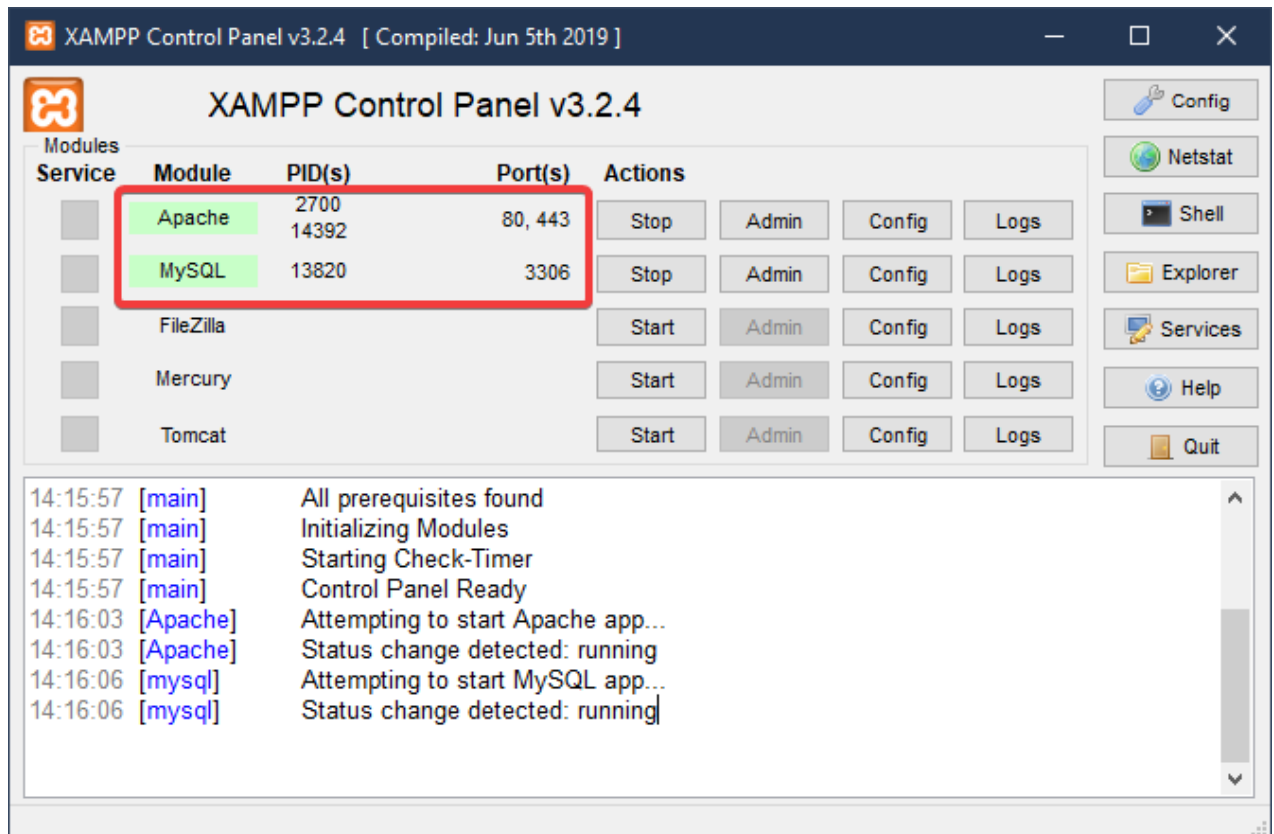


Figura 44: Servicios en marcha. Fuente: Elaboración propia.

Despliegue del paquete del proyecto

Una vez instalado el paquete con todos los servicios necesarios para el correcto funcionamiento del proyecto, se deben extraer los archivos de “SocialBook.zip” en el directorio que XAMPP destina para esto. Este directorio es “htdocs”. Se encuentra dentro del directorio “xampp”.

`C:\xampp\htdocs`

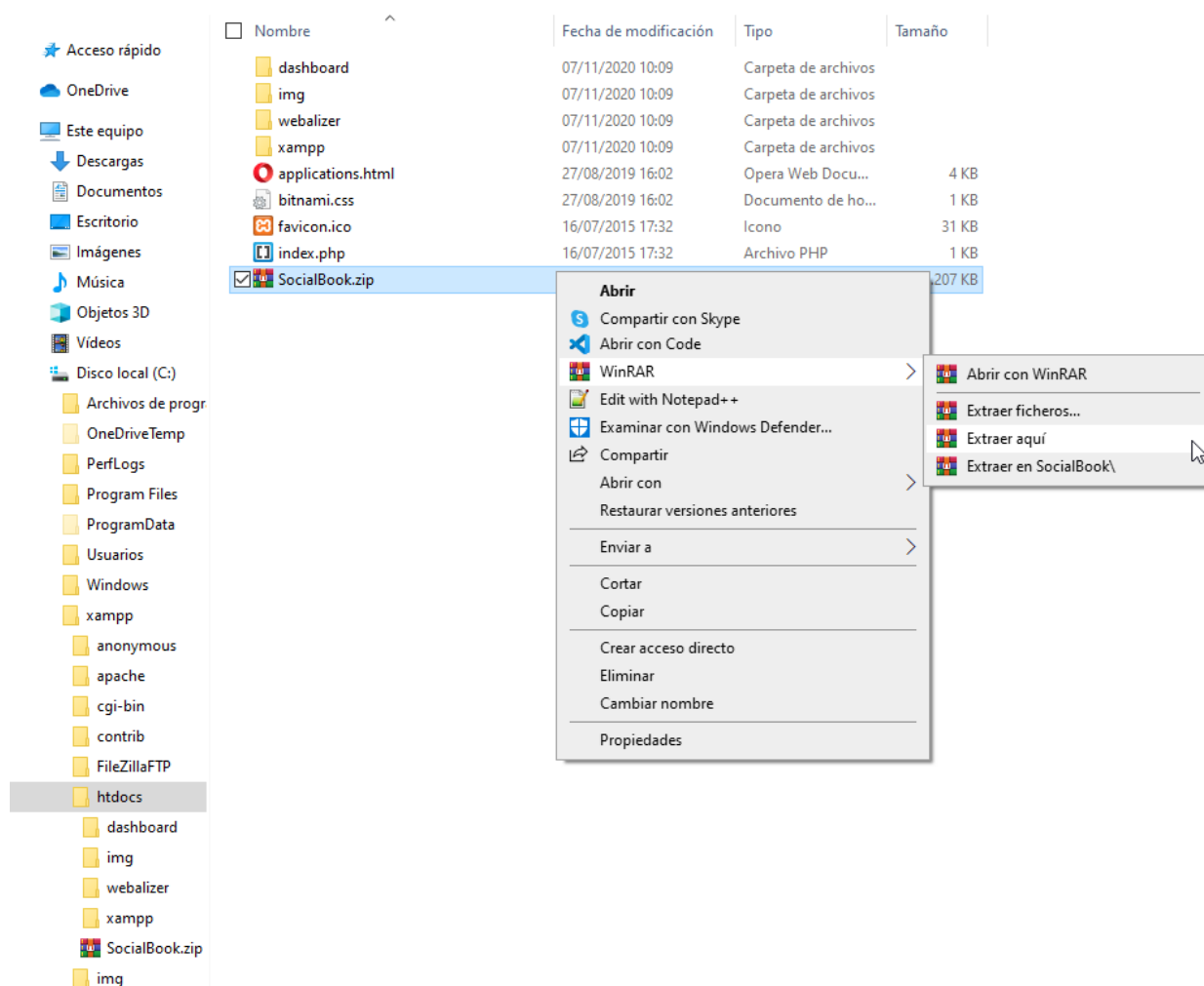


Figura 45: Despliegue del proyecto en local. Fuente: Elaboración propia.

De esta manera, queda totalmente desplegado el proyecto en el entorno local, mostrando la siguiente estructura de carpetas:

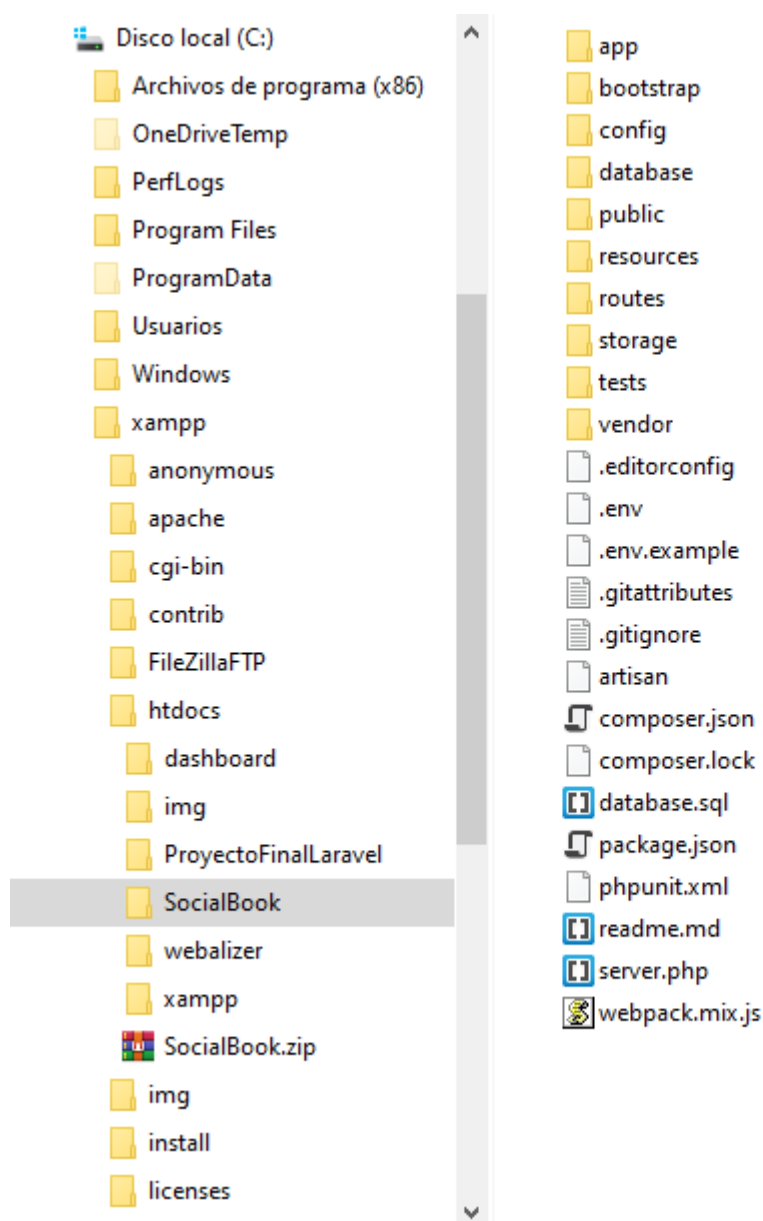


Figura 46: Estructura de archivos del proyecto. Fuente: Elaboración propia.

Base de datos

A continuación, se debe crear la base de datos. Para esto, dentro de la raíz del proyecto, se proporcionan el archivo *database.sql* con los códigos que se han de introducir en el gestor de base de datos. Estos códigos son los de la construcción de la base de datos e introducción de algunos datos para la prueba del proyecto.

La ruta donde se encuentra el archivo es la siguiente:

`C:\xampp\htdocs\SocialBook`

Para acceder a la interface del gestor de base de datos se ha de lanzar el panel de control de XAMPP y presionar el botón de “Admin” del servicio MySQL.

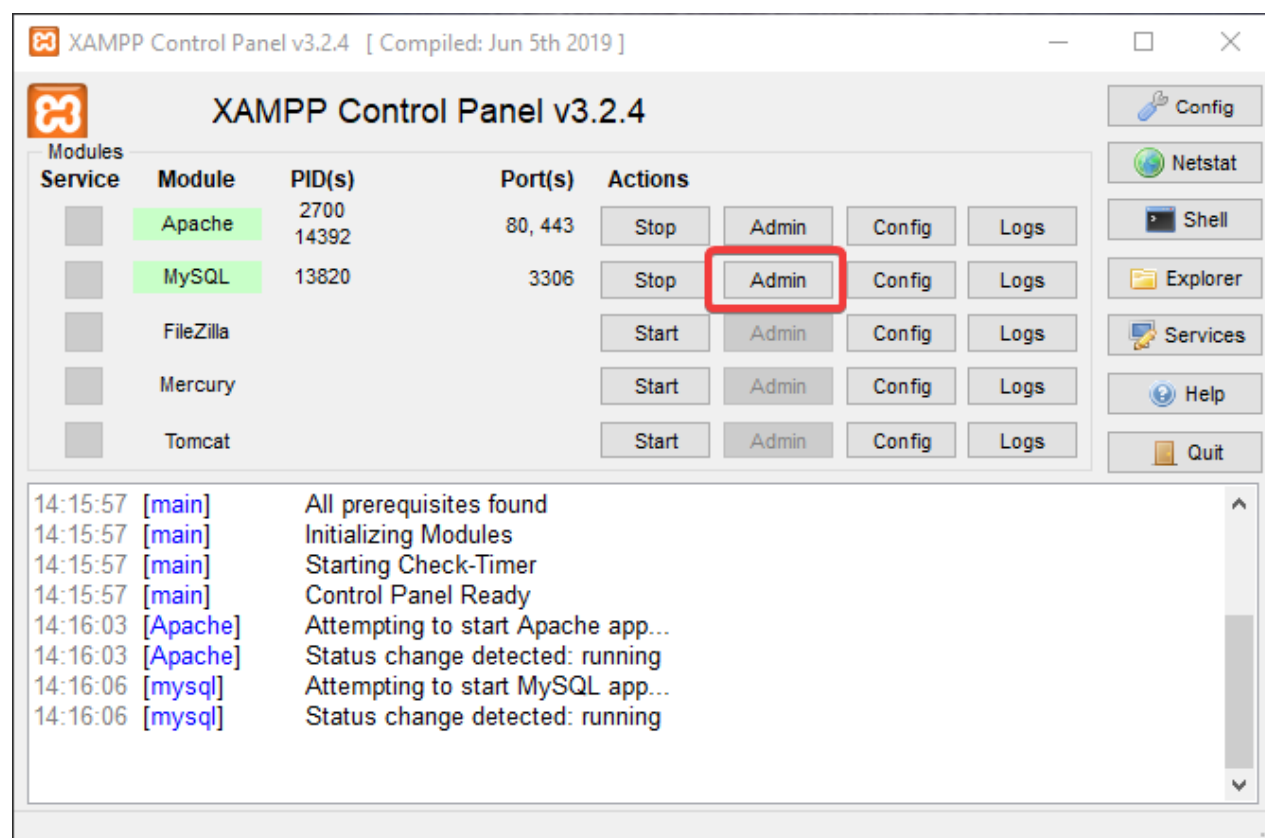


Figura 47: Acceso al gestor de base de datos. Fuente: Elaboración propia.

Esto redirige a la interfaz del gestor de base de datos en una ventana del navegador. En la pestaña “Importar”, se selecciona el archivo *database.sql* anteriormente mencionado y se presiona el botón de continuar.

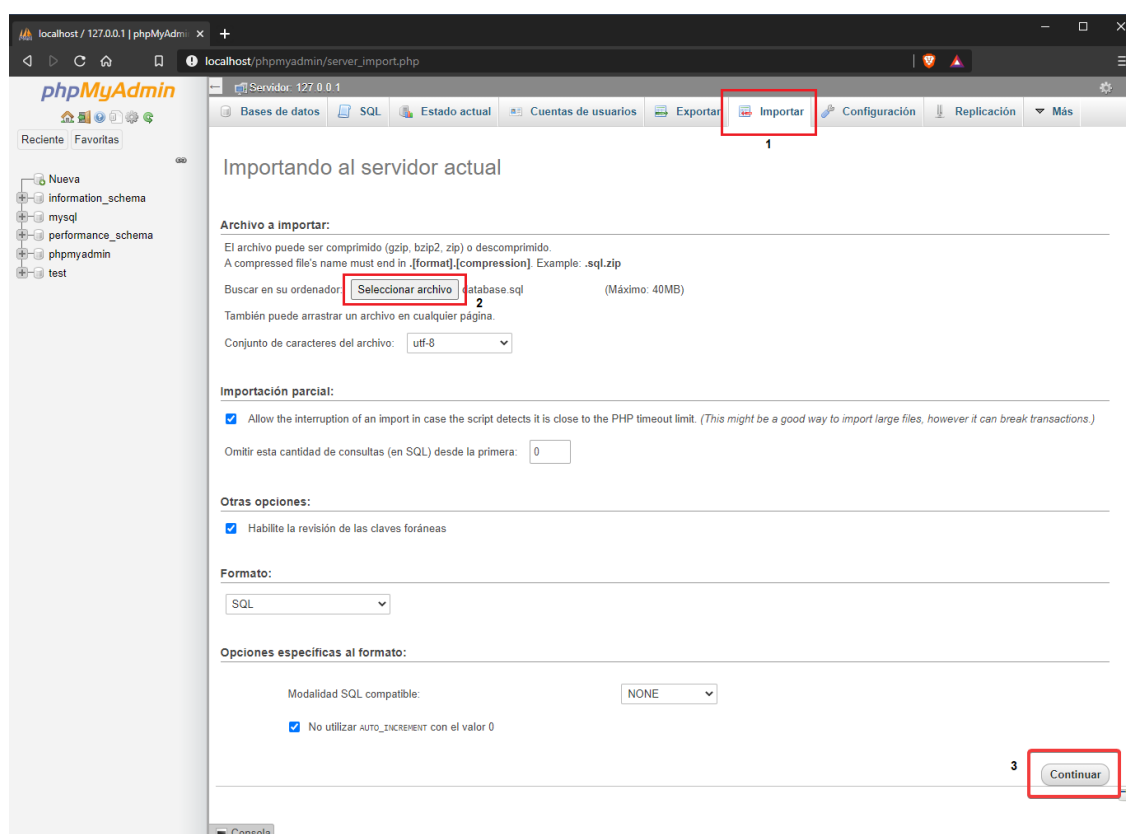


Figura 48: Importación de la base de datos. Fuente: Elaboración propia.

Una vez finalizada la importación, se mostrarán los mensajes de éxito en la tarea y, presionando en el icono de “Recargar”, se refrescará el panel de navegación, mostrando la base de datos importada.

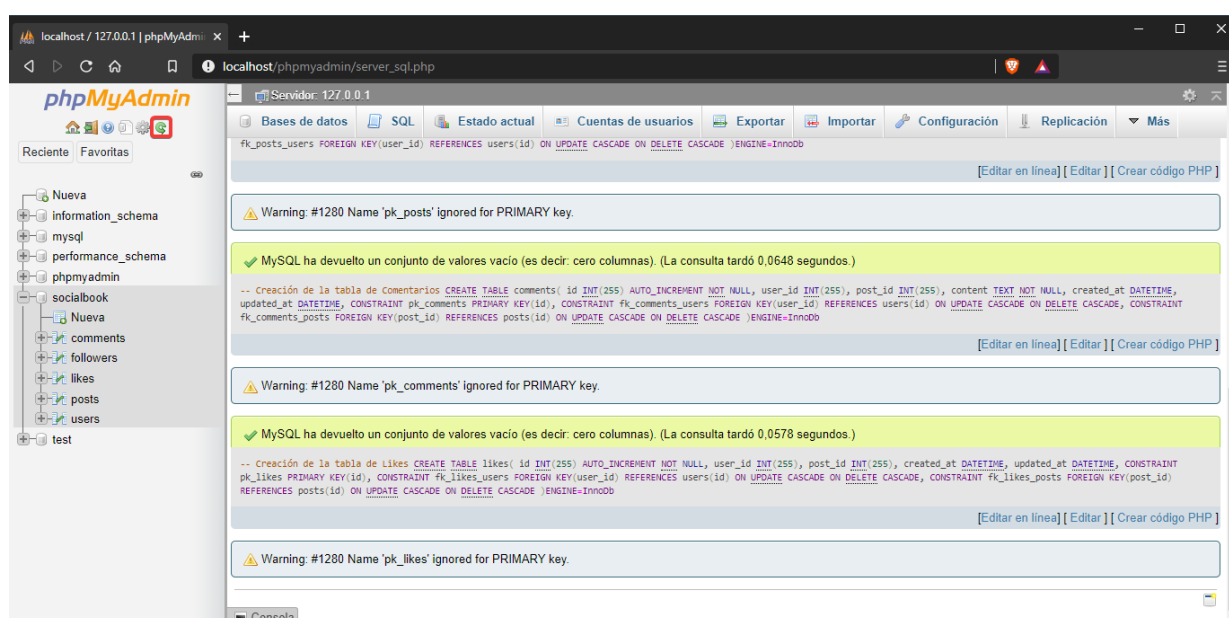


Figura 49: Base de datos importada. Fuente: Elaboración propia.

Host Virtual

Finalmente, para poder acceder al proyecto de manera local utilizando URL's limpias se ha de configurar un host virtual. Este paso ya se ha explicado como se realiza en el apartado 5.- *Desarrollo del proyecto, sección 5.2.-Diseño, Host Virtual*. Igualmente, para no perder la continuidad de la lectura, se vuelve a desarrollar de manera más práctica.

Lo primero que se ha de hacer es modificar el archivo *httpd-vhosts.conf* de Apache. La ruta de acceso a este archivo es la siguiente:

C:\xampp\apache\conf\extra\httpd-vhosts.conf

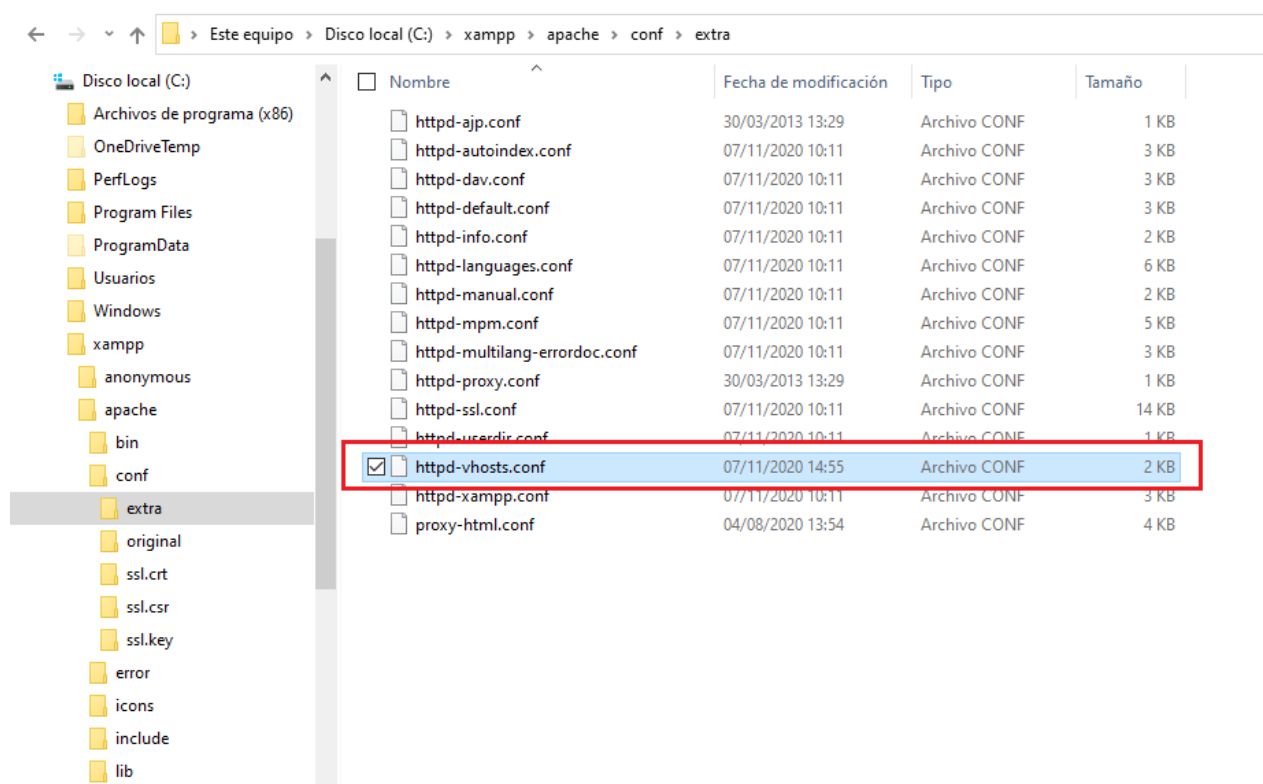
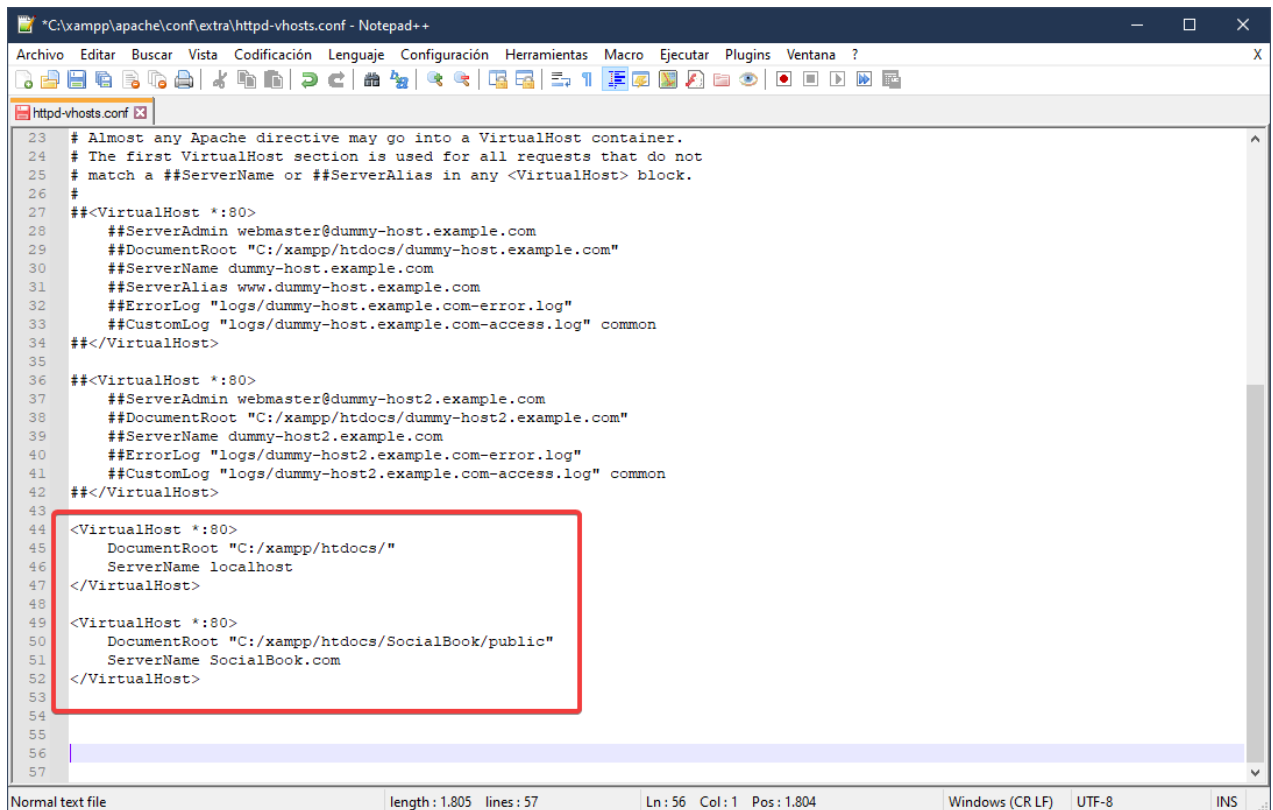


Figura 50: Archivo httpd-vhosts.conf. Fuente: Elaboración propia.

Se abre el archivo con algún editor de texto y, al final del mismo, se agrega el siguiente bloque de código:

```
<VirtualHost *:80>
    DocumentRoot "C:/xampp/htdocs/"
    ServerName localhost
</VirtualHost>
```

```
<VirtualHost *:80>
    DocumentRoot "C:/xampp/htdocs/SocialBook/public"
    ServerName SocialBook.com
</VirtualHost>
```



```
23 # Almost any Apache directive may go into a VirtualHost container.
24 # The first VirtualHost section is used for all requests that do not
25 # match a ##ServerName or ##ServerAlias in any <VirtualHost> block.
26 #
27 ##<VirtualHost *:80>
28 ##ServerAdmin webmaster@dummy-host.example.com
29 ##DocumentRoot "C:/xampp/htdocs/dummy-host.example.com"
30 ##ServerName dummy-host.example.com
31 ##ServerAlias www.dummy-host.example.com
32 ##ErrorLog "logs/dummy-host.example.com-error.log"
33 ##CustomLog "logs/dummy-host.example.com-access.log" common
34 ##</VirtualHost>
35
36 ##<VirtualHost *:80>
37 ##ServerAdmin webmaster@dummy-host2.example.com
38 ##DocumentRoot "C:/xampp/htdocs/dummy-host2.example.com"
39 ##ServerName dummy-host2.example.com
40 ##ErrorLog "logs/dummy-host2.example.com-error.log"
41 ##CustomLog "logs/dummy-host2.example.com-access.log" common
42 ##</VirtualHost>
43
44 <VirtualHost *:80>
45     DocumentRoot "C:/xampp/htdocs/"
46     ServerName localhost
47 </VirtualHost>
48
49 <VirtualHost *:80>
50     DocumentRoot "C:/xampp/htdocs/SocialBook/public"
51     ServerName SocialBook.com
52 </VirtualHost>
53
54
55
56
57
```

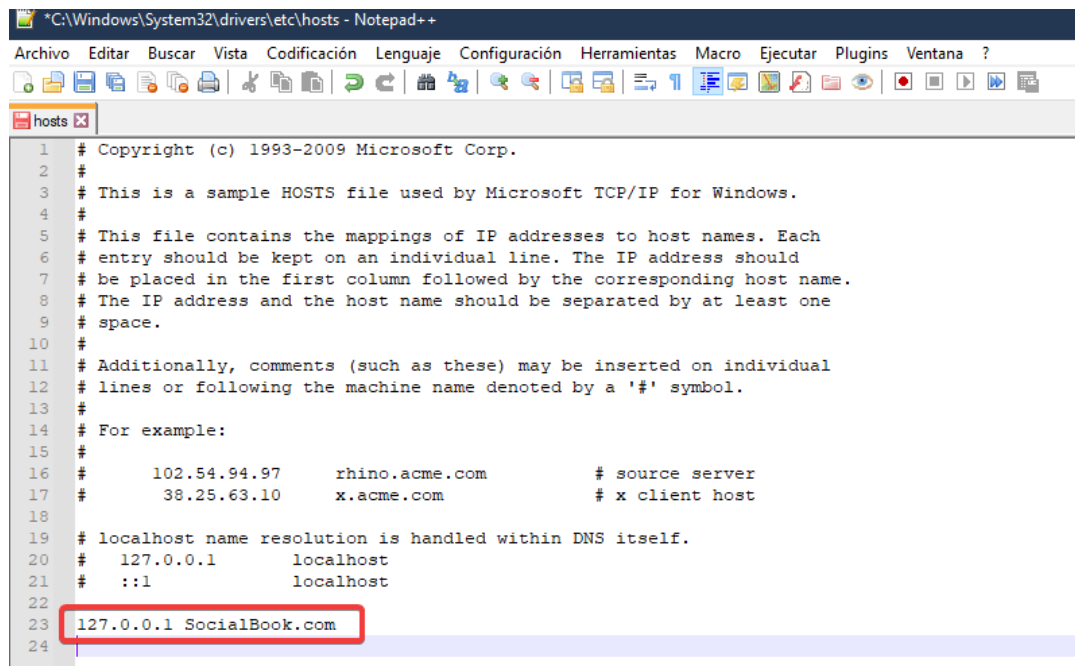
Figura 51: Archivo httpd-vhosts.conf modificado. Fuente: Elaboración propia.

Seguido a esto, se ha de realizar una modificación el archivo *hosts* de Windows. En este archivo se relaciona el nombre del host a la dirección IP local. La ruta de acceso a este archivo es la siguiente:

C:\Windows\System32\drivers\etc\hosts

Se abre el archivo con cualquier editor de texto y se agrega el siguiente bloque de código:

127.0.0.1 SocialBook.com



```

1 # Copyright (c) 1993-2009 Microsoft Corp.
2 #
3 # This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
4 #
5 # This file contains the mappings of IP addresses to host names. Each
6 # entry should be kept on an individual line. The IP address should
7 # be placed in the first column followed by the corresponding host name.
8 # The IP address and the host name should be separated by at least one
9 # space.
10 #
11 # Additionally, comments (such as these) may be inserted on individual
12 # lines or following the machine name denoted by a '#' symbol.
13 #
14 # For example:
15 #
16 #         102.54.94.97       rhino.acme.com          # source server
17 #         38.25.63.10       x.acme.com              # x client host
18 #
19 # localhost name resolution is handled within DNS itself.
20 # 127.0.0.1       localhost
21 # ::1            localhost
22 #
23 127.0.0.1 SocialBook.com
24

```

Figura 52: Archivo hosts modificado. Fuente: Elaboración propia.

Finalizada la configuración del host virtual, se reinician los servicios de Apache y MySQL en XAMPP, quedando el entorno preparado para el acceso al proyecto.

Acceso al proyecto

El acceso al proyecto se realiza desde cualquier navegador web disponible, introduciendo la siguiente dirección web:

Socialbook.com

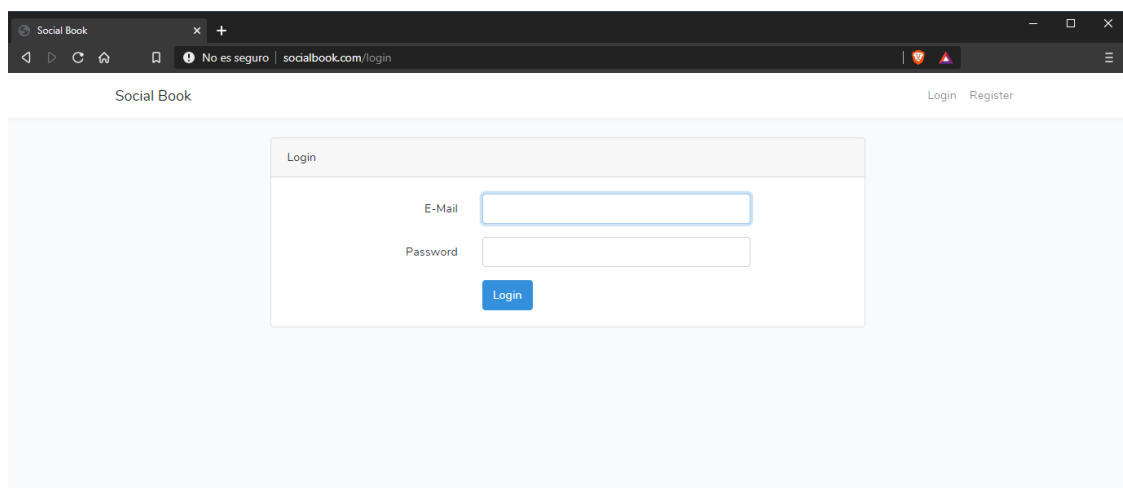


Figura 53: Acceso al proyecto. Fuente: Elaboración propia.

Despliegue en un hosting

Para este trabajo se ha utilizado un servidor previamente contratado, por lo cual el nombre del dominio no tiene nada que ver con el del proyecto en sí, pero cumple con todo lo necesario de cara una corrección y comprobación de todas las funcionalidades del mismo.

Preparación de los paquetes a instalar

Lo primero que se realizó fue la separación de los paquetes a subir en el servidor. Este se administra desde el panel de control **cPanel**.

El primer paquete es el que corresponde a la carpeta *public* del proyecto, mientras que el segundo engloba el resto de contenido.

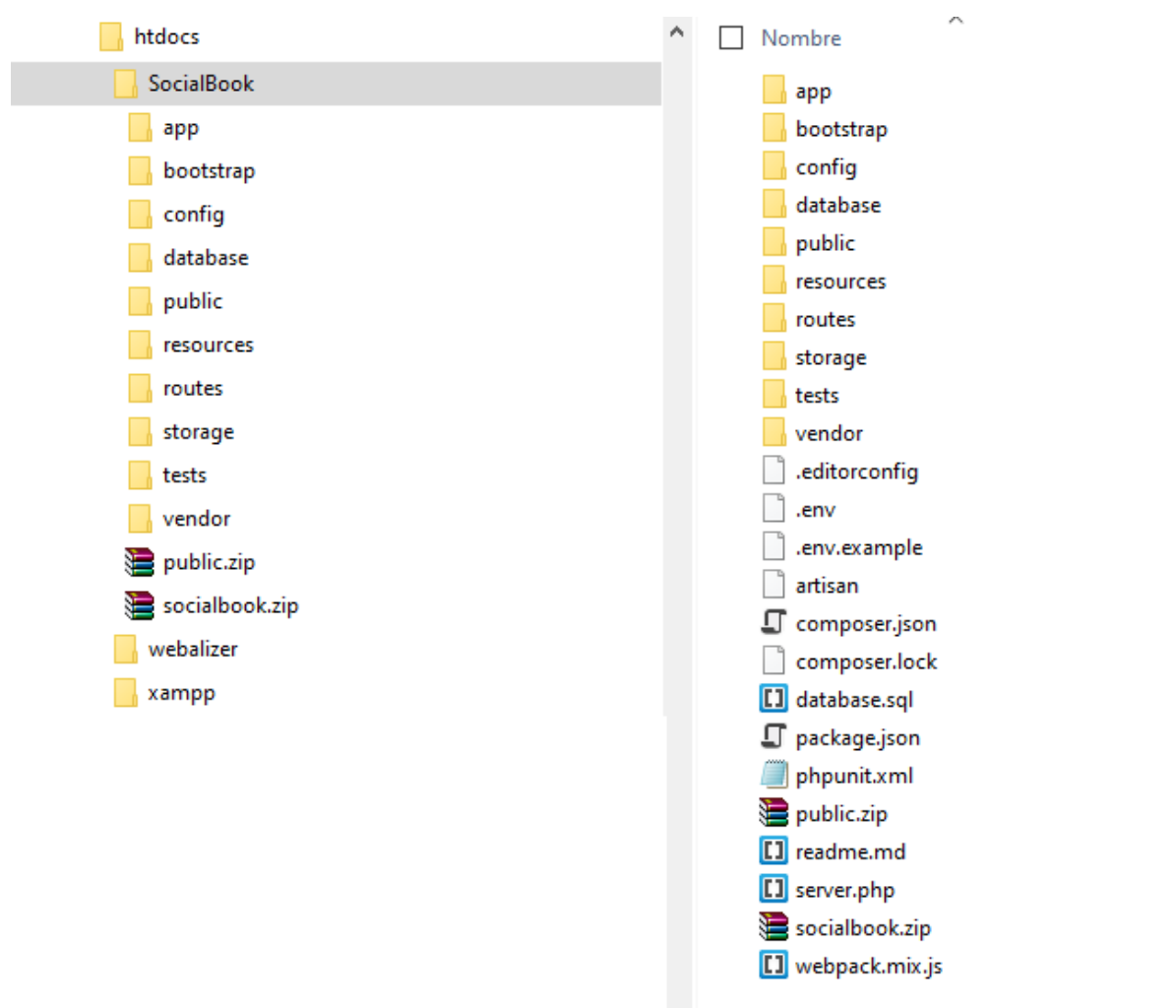


Figura 54: Preparación de archivos para subir al servidor. Fuente: Elaboración propia.

Los archivos de la carpeta *public* se comprimen en un .zip con nombre **public.zip**. Los demás archivos se comprimen en un archivo de nombre **socialbook.zip**.

Seguido, accedemos al gestor de archivos de cPanel, desde donde tenemos acceso a la raíz del disco del servidor contratado.

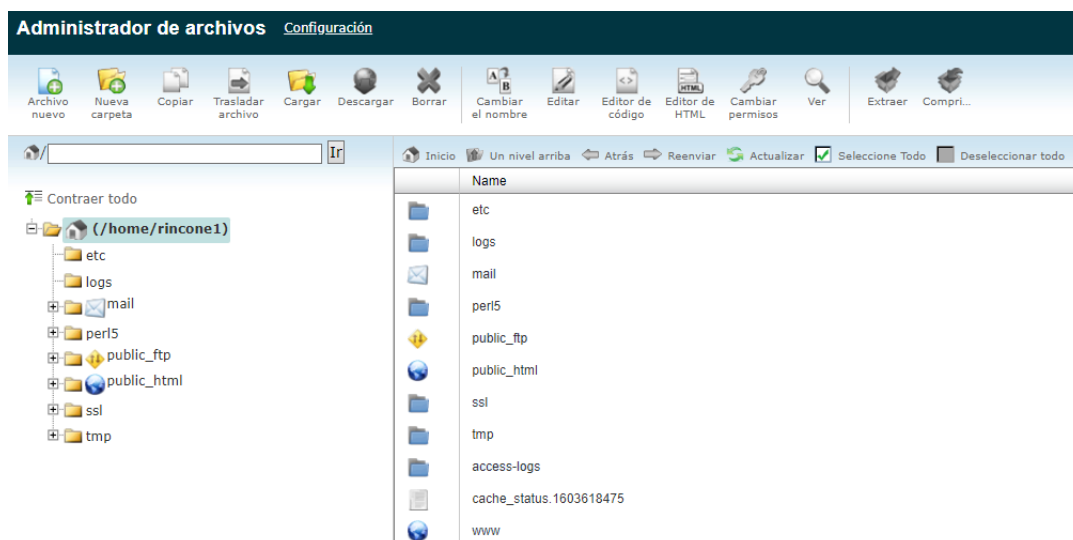


Figura 55: Administrador de archivos de cPanel. Fuente: Elaboración propia.

Subida del proyecto al servidor

Ya con los dos paquetes de archivos creados, se carga dentro de la carpeta *public_html* del servidor el archivo *public.zip* y se extraen los archivos que contiene. Es en esta carpeta donde estarán todos los archivos que se ejecutarán al ingresar el nombre del dominio en un navegador.

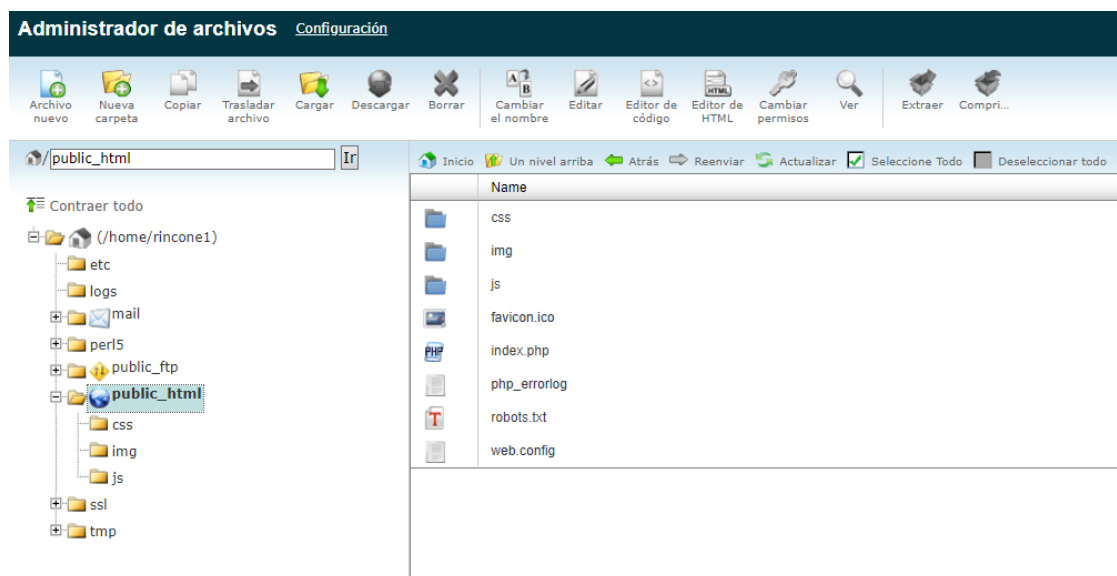


Figura 56: Carpeta *public_html*. Fuente: Elaboración propia.

Luego, se crea una carpeta nueva en la raíz del servidor, donde se alojarán el resto de archivos del proyecto. En el caso de este proyecto, llevará el nombre *laravel*. Es en esta donde estarán todos los archivos de la estructura generada al crear el proyecto con laravel y los codificados para sus funcionalidades.

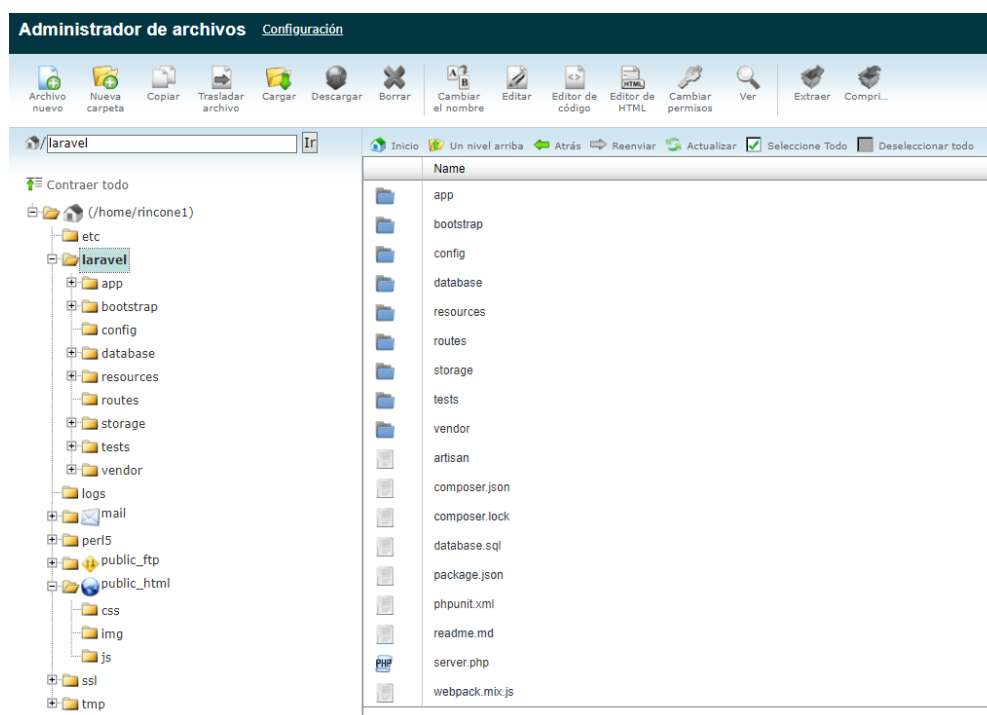


Figura 57: Carpeta laravel. Fuente: Elaboración propia.

Creación de la Base de Datos

El hosting contratado ofrece la administración de una base de datos. Es así que, al igual que cuando se trabajó en local, se crea una base de datos a la que apuntará el proyecto instalado en el servidor. Este trabajo no difiere mucho del anteriormente realizado, solo se diferencia con la creación de un usuario sobre el cual habrá que dar permisos para su manipulación.

Desde el panel de control cPanel, se accede al asistente de base de datos. En este se le asigna un nombre a la base de datos, nombre al usuario, una contraseña y se le asignan los permisos de manipulación de la misma. Estos datos son de importancia, puesto que se utilizarán en la configuración de conexión en laravel.

BASE DE DATOS	TAMAÑO	USUARIOS CON PRIVILEGIO	ACCIONES
rincone1_socialbook	0.64 MB	rincone1_root	Cambiar el nombre Borrar

Figura 58: Base de Datos. Fuente: Elaboración propia.

Configuración del servidor en Laravel

Ya con el proyecto subido al servidor y con la base de datos creada, solo queda realizar unas pocas modificaciones en el código del proyecto. Estas se realizan en los siguientes archivos:

index.php

Este se encuentra en la carpeta *public_html*. Se modifican las rutas encargadas de cargar bootstrap y el autoloader de clases, indicando que han de apuntar a la carpeta que se ha creado con el proyecto, llamada *laravel*.

```
13 | -----
14 | Register The Auto Loader
15 | -----
16 |
17 | Composer provides a convenient, automatically generated class loader for
18 | our application. We just need to utilize it! We'll simply require it
19 | into the script here so that we don't have to worry about manual
20 | loading any of our classes later on. It feels great to relax.
21 |
22 | */
23 |
24 | require __DIR__.'../../laravel/vendor/autoload.php';
25 |
26 | /*
27 | -----
28 | Turn On The Lights
29 | -----
30 |
31 | We need to illuminate PHP development, so let us turn on the lights.
32 | This bootstraps the framework and gets it ready for use, then it
33 | will load up this application so that we can run it and send
34 | the responses back to the browser and delight our users.
35 |
36 | */
37 |
38 | $app = require_once __DIR__.'../../laravel/bootstrap/app.php';
39 |
40 | /*
41 | -----
42 | Run The Application
43 | -----
44 |
45 | Once we have the application, we can handle the incoming request
46 | through the kernel, and send the associated response back to
47 | the client's browser allowing them to enjoy the creative
48 | and wonderful application we have prepared for them.
49 |
50 | */
```

Figura 59: Modificación *index.php*. Fuente: Elaboración propia.

app.php

Este se encuentra en la carpeta *laravel/config*. Se modifica el “ambiente” en el cual el proyecto se correrá, cambiando de estar en local a en producción, y la URL del mismo.

```
/*
|-----
| Application Environment
|-----
|
| This value determines the "environment" your application is currently
| running in. This may determine how you prefer to configure various
| services your application utilizes. Set this in your ".env" file.
|
*/

'env' => env('APP_ENV', 'production'),

/*
|-----
| Application URL
|-----
|
| This URL is used by the console to properly generate URLs when using
| the Artisan command line tool. You should set this to the root of
| your application so that it is used when running Artisan tasks.
|
*/

'url' => env('APP_URL', 'https://rinconesdemallorca.com'),
```

Figura 60: Modificación app.php. Fuente: Elaboración propia.

database.php

Este se encuentra en la carpeta *laravel/config*. Se modifican las variables de conexión a la base de datos.

```
'mysql' => [
    'driver' => 'mysql',
    'host' => env('DB_HOST', '127.0.0.1'),
    'port' => env('DB_PORT', '3306'),
    'database' => env('DB_DATABASE', 'rincone1_socialbook'),
    'username' => env('DB_USERNAME', 'rincone1_root'),
    'password' => env('DB_PASSWORD', 'gfdh0H0Jhlg558*/gf'),
    'unix_socket' => env('DB_SOCKET', ''),
    'charset' => 'utf8mb4',
    'collation' => 'utf8mb4_unicode_ci',
    'prefix' => '',
    'strict' => true,
    'engine' => null,
],
```

Figura 61: Modificación database.php. Fuente: Elaboración propia.

AppServiceProvider.php

Este se encuentra en la carpeta `laravel/app/providers`. Se modifica la función `register`, configurando la nueva ruta pública del proyecto.

```
/**
 * Register any application services.
 *
 * @return void
 */
public function register()
{
    $this->app->bind('path.public',function(){
        return'/home/rincones1/public_html';
    });
}
```

Figura 62: Modificación `AppServiceProvider.php`. Fuente: Elaboración propia.

.env

Este se encuentra en la carpeta `laravel`. Se modifican las variables de conexión a la base de datos.

```
APP_NAME=Laravel
APP_ENV=production
APP_KEY=base64:yUuSswSyWPVJCxdqu0Gskru8bvHsx87XKmDGDuDv80=
APP_DEBUG=true
APP_URL=https://rinconesdemallorca.com/

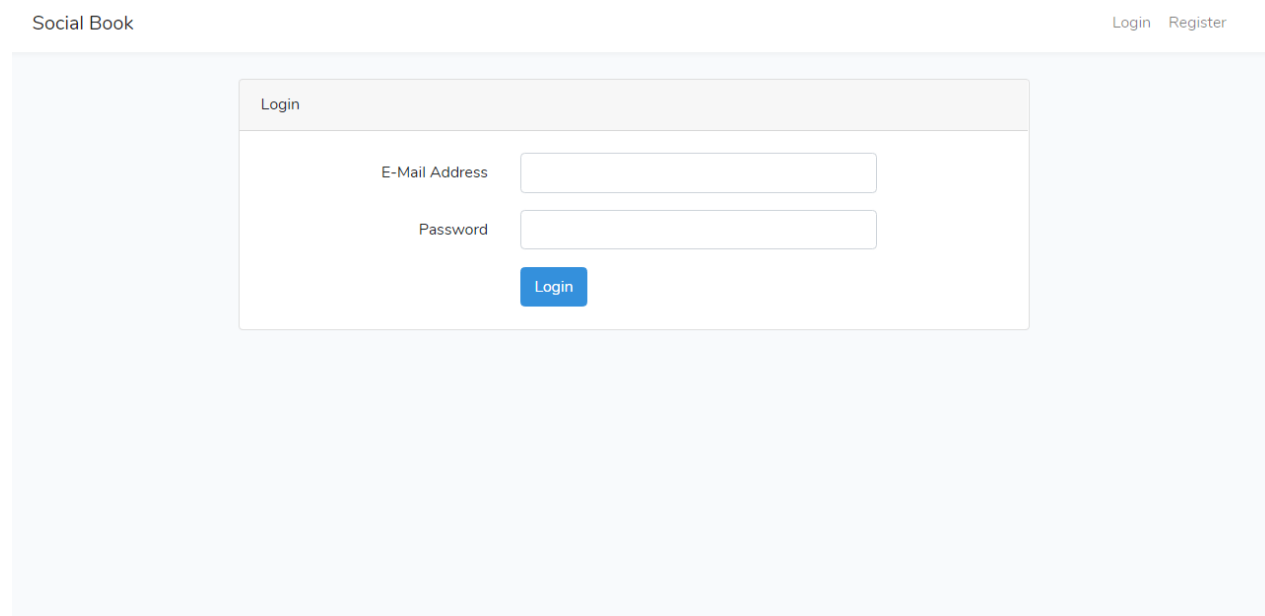
LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=rincones1_socialbook
DB_USERNAME=rincones1_root
DB_PASSWORD=gfdh0HOJhlg558*/gf
```

Figura 63: Modificación `.env`. Fuente: Elaboración propia.

Accediendo a la aplicación

Con todo configurado, tan solo quede acceder a la aplicación. Desde cualquier navegador web e introduciendo la URL de la aplicación, se accede a la pantalla de inicio de la misma.



Social Book Login Register

Login

E-Mail Address

Password

Figura 64: Pantalla de inicio de la web. Fuente: Elaboración propia.

10.2.-MANUAL DE USUARIO

Registro de usuario

Se realiza accediendo a la pantalla de registro, donde un usuario ha de introducir los datos indicados en la siguiente figura.

Social Book Login Registro

Register

Nombre	<input type="text" value="Julio César"/>	1
Apellido	<input type="text" value="Novelli"/>	2
Nickname	<input type="text" value="JulioNovelli"/>	3
E-Mail	<input type="text" value="julionovelli@hotmail.com"/>	4
Avatar	<input type="button" value="Seleccionar archivo"/> N...	5
Password	<input type="password" value="....."/>	6
Confirmar Password	<input type="password" value="....."/>	7

8

Figura 65: Pantalla de registro. Fuente: Elaboración propia.

1. Nombre. Introducir el nombre del usuario.
2. Apellido. Introducir el apellido del usuario.
3. Nickname. Introducir el nombre de usuario.
4. E-Mail. Introducir el email del usuario.
5. Avatar. Seleccionar una imagen de perfil.
6. Password. Introducir la contraseña.
7. Confirmar Password. Introducir nuevamente la contraseña.
8. Botón de registro. Realiza el registro de la cuenta.

Acceso a la plataforma

Se realiza por medio de un login. Esta pantalla será la primera en mostrarse al acceder al dominio de la web. El usuario ha de introducir el email y la contraseña con la cual se haya registrado en la plataforma.

The screenshot shows the login interface of the 'Social Book' platform. At the top left, the text 'Social Book' is labeled with a '1'. At the top right, there are two buttons: 'Login' (labeled with a '2') and 'Registro' (labeled with a '3'). In the center, there is a 'Login' form box. Inside this box, the 'E-Mail' field contains the text 'julionovelli@hotmail.com' and is labeled with a '4'. Below it, the 'Password' field is filled with dots and is labeled with a '5'. At the bottom of the form box is a blue 'Login' button, labeled with a '6'.

Figura 66: Pantalla de inicio. Login. Fuente: Elaboración propia.

1. Nombre de la web. Redirige a la pantalla de inicio.
2. Login. Redirige a la pantalla de login.
3. Registro. Redirige a la pantalla de registro de usuario.
4. E-Mail. Email con el cual el usuario se ha registrado.
5. Password. Se introduce la contraseña con la que se ha registrado el usuario.
6. Botón de Login. Realiza el logueo del usuario.

Pantalla de bienvenida

Es la pantalla a la que se accede luego de registrar una cuenta. Esta cuenta con opciones de acciones rápidas.



Figura 67: Pantalla de bienvenida. Fuente: Elaboración propia.

1. Buscar. Redirige a la pantalla de búsqueda de cuentas de usuarios ya registrados.
2. Global. Redirige a la pantalla donde se listan todas las publicaciones realizadas en la web.
3. Inicio. Redirige a la pantalla donde se listan todas las publicaciones realizadas por las cuentas seguidas por el usuario identificado.
4. Subir. Redirige a la pantalla donde se realiza una publicación nueva.
5. Avatar. Imagen de perfil del usuario identificado. Redirige a la pantalla de perfil del usuario identificado.
6. Menú de usuario. Expande un menú con acciones específicas del usuario identificado.
7. Descubrir cuentas. Redirige a la pantalla de búsqueda de cuentas de usuarios ya registrados.
8. Descubrir publicaciones. Redirige a la pantalla donde se listan todas las publicaciones realizadas en la web.
9. Crear publicación. Redirige a la pantalla donde se realiza una publicación nueva.

Menú del usuario

Menú expandible con acciones específicas del usuario identificado.

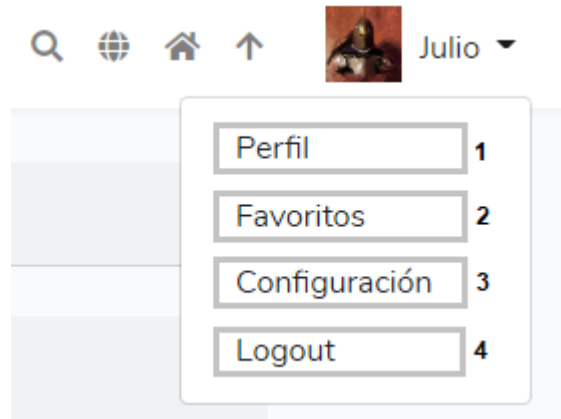


Figura 68: Menú de usuario. Fuente: Elaboración propia.

1. Perfil. Redirige a la pantalla de perfil del usuario identificado.
2. Favoritos. Redirige a la pantalla donde se listan todas las publicaciones a las cuales el usuario identificado les ha dado un like.
3. Configuración. Redirige a la pantalla donde se modifican los datos del usuario identificado.
4. Logout. Cierra la sesión del usuario identificado, redirigiendo a la pantalla de Login.

Modificar datos del usuario

Si, una vez registrado un usuario desea cambiar algunos de sus datos, esta es la pantalla donde hacerlo. Se accede por medio del menú de usuario, pinchando en la opción de configuración.

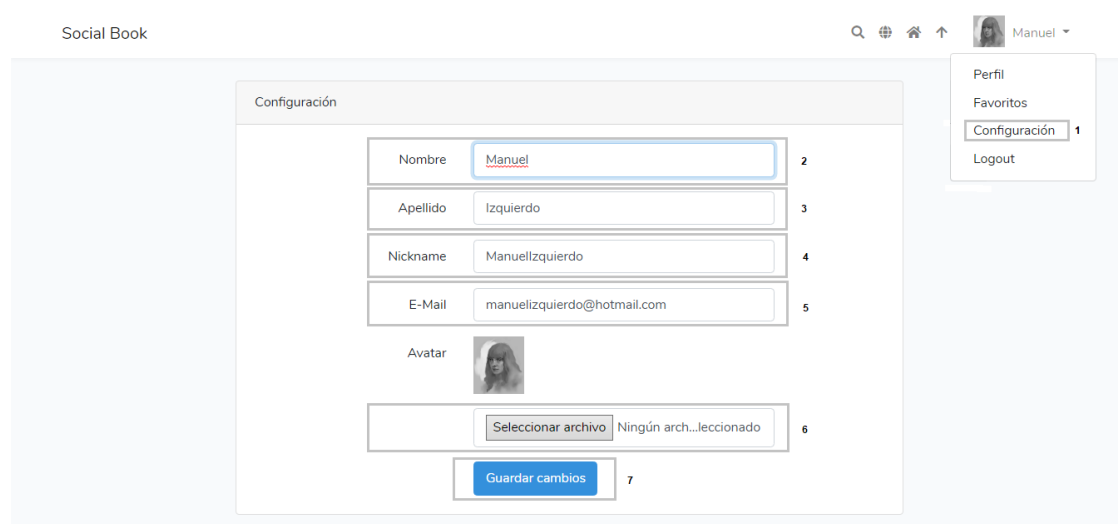


Figura 69: Modificación de datos del usuario. Fuente: Elaboración propia.

1. Configuración. Acceso al formulario de modificación de datos del usuario.
2. Nombre del usuario.
3. Apellido del usuario.
4. Nickname. Nombre de usuario con el que se lo identificará en la web.
5. E-Mail del usuario.
6. Avatar. Imagen de perfil del usuario.
7. Botón para guardar los cambios.

Crear una publicación

Se realiza accediendo a la pantalla de “Crear Publicación”. Esta pantalla es un formulario donde el usuario ha de introducir el título, la portada, autor, editorial, número de páginas y descripción del libro a publicar.

Figura 70: Creación de una publicación. Fuente: Elaboración propia.

1. Acceso al formulario de publicación.
2. Título del libro.
3. Imagen. Portada del libro.
4. Autor del libro.
5. Editorial del libro.
6. Número de páginas del libro.
7. Breve descripción sobre el tema del libro.
8. Botón para subir la publicación.

Pantalla del detalle de la publicación

Es en esta pantalla donde se visualizará al completo toda la información sobre la publicación.

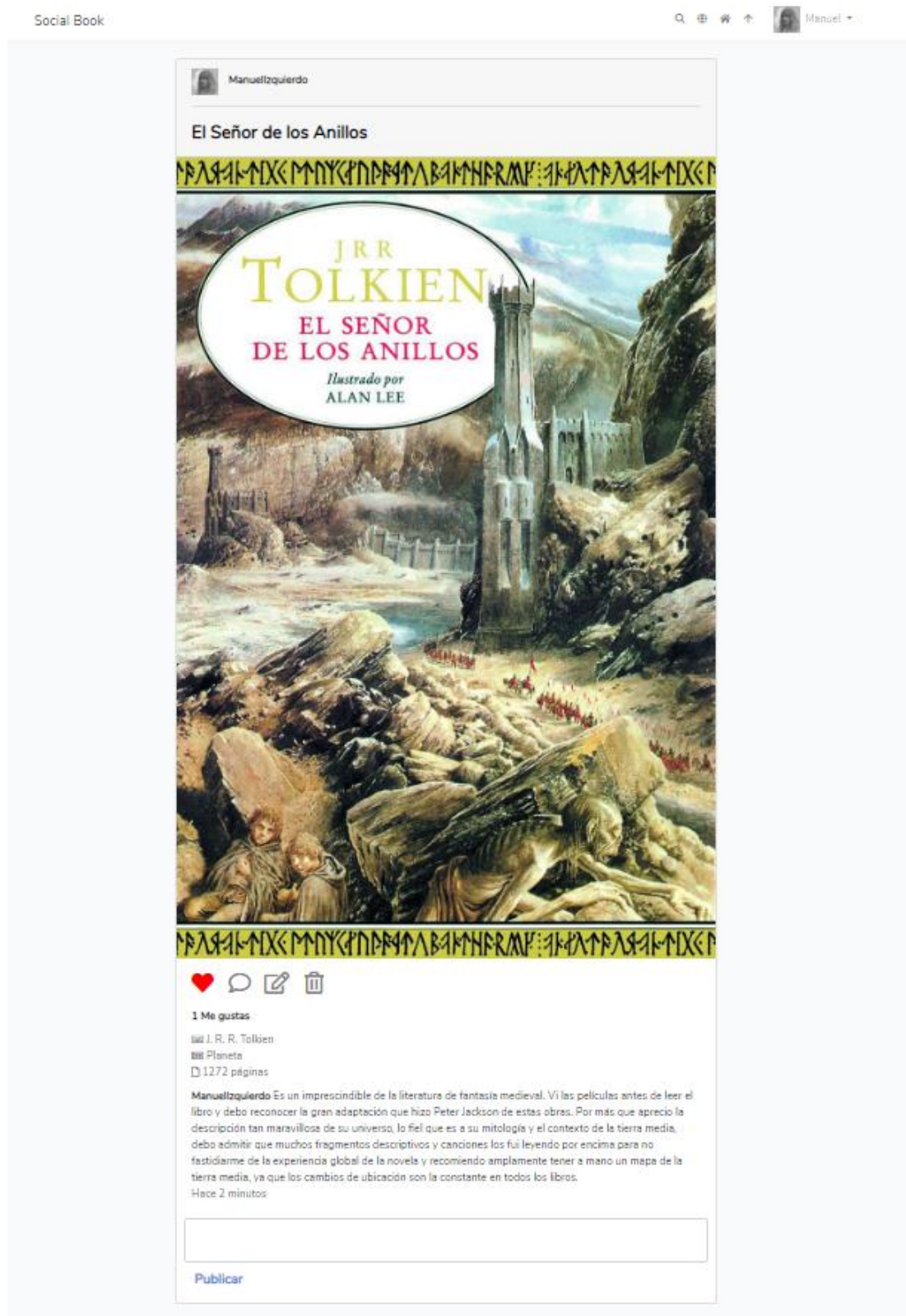


Figura 71: Pantalla del detalle de la publicación. Fuente: Elaboración propia.

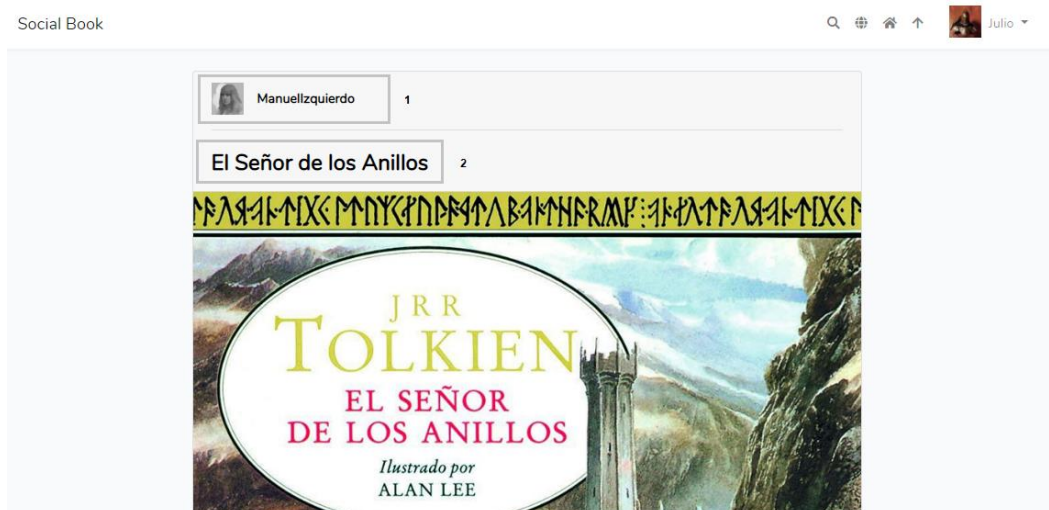


Figura 72: Cabecera del detalle de la publicación. Fuente: Elaboración propia.

1. Creador de la publicación. Pinchando sobre el mismo se accede a la pantalla de perfil del usuario.
2. Título del libro.

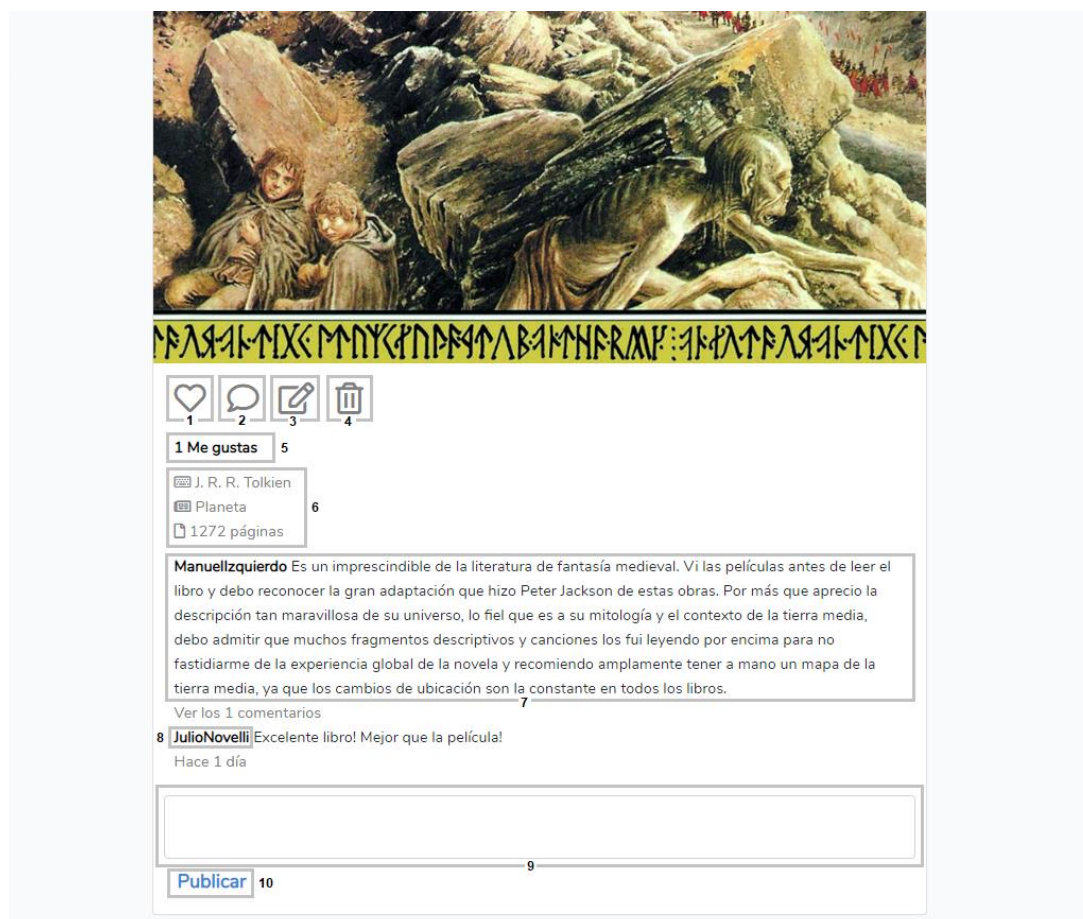


Figura 73: Cuerpo del detalle de la publicación. Fuente: Elaboración propia.

1. Me Gusta. Registra un like y guarda la publicación en favoritos.
2. Comentar. Realizar un comentario en la publicación.
3. Editar. Accede al formulario de modificación de la publicación.
4. Borrar. Elimina la publicación.
5. Muestra la cantidad de likes que tiene la publicación. Pinchando se accede a la pantalla que lista los usuarios que han dado me gusta a la publicación.
6. Datos del libro.
7. Breve descripción sobre el tema del libro.
8. Nickname del usuario que realizó el comentario. Pinchando se accede a su perfil.
9. Caja de comentarios.
10. Botón para confirmar el comentario.

Modificación de publicación

La modificación de una publicación solo es posible para el autor de la misma. Si, una vez realizada una publicación, el usuario desea modificar la misma, esta es la pantalla donde hacerlo. Se accede por medio del icono “*Editar*” al pie de la publicación. Esta pantalla es idéntica a la de creación de publicación. Ver Figura 70.

Eliminar publicación

La eliminación de una publicación solo es posible para el autor de la misma. El usuario ha de pinchar sobre el icono de “*Borrar*”. Ver Figura 73.

Comentar publicación

Opción accesible desde la pantalla de detalle de la publicación o pinchando en el icono de “*Comentar*”. Ver Figura 73.

Like

Opción disponible al pie de cada publicación por medio del icono del corazón. Al pinchar una vez, el corazón cambia al color rojo, indicando que se ha registrado el like y que la publicación pasa a estar listada en los favoritos del usuario.

Dislike

Para eliminar un like, el usuario ha de pinchar sobre el icono del corazón de color rojo. Una vez hecho esto, el corazón cambiará de color y eliminará de la lista de favoritos la publicación.

Buscar usuarios

En la pantalla de búsqueda de cuentas accesible con el icono de la lupa, se encuentra un cajón de búsqueda, donde se realiza un filtro sobre las cuentas listadas en la pantalla.

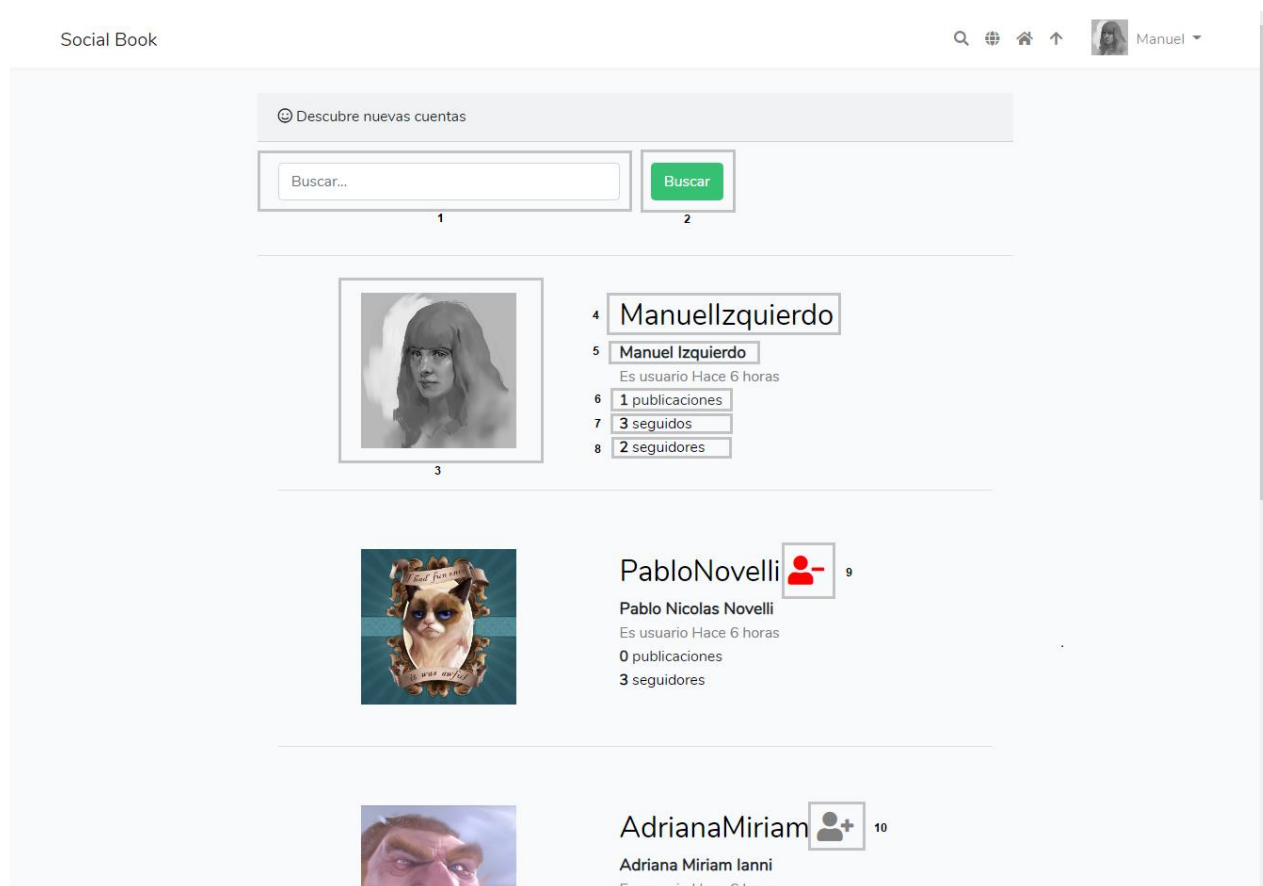


Figura 74: Pantalla de búsqueda de usuarios. Fuente: Elaboración propia.

1. Cajón de búsqueda.
2. Botón para realizar la búsqueda.
3. Imagen de perfil. Pinchando se accede al perfil del usuario.
4. Nickname. Nombre de usuario. Pinchando se accede al perfil del usuario.
5. Nombre y apellido del usuario. Pinchando se accede al perfil del usuario.
6. Cantidad de publicaciones realizadas por el usuario.
7. Número de cuentas seguidas por el usuario. Pinchando se accede a la lista de las mismas.
8. Número de cuentas que siguen al usuario. Pinchando se accede a la lista de las mismas.
9. Icono para dejar de seguir una cuenta.
10. Icono para seguir una cuenta.

Seguir una cuenta

Opción accesible desde el perfil de un usuario o el listado de cuentas registradas en la plataforma.

Se registra el seguimiento de una cuenta pinchando en el icono de “*Seguir*”. Ver Figura 74.

Dejar de seguir una cuenta

Opción accesible desde el perfil de un usuario o el listado de cuentas registradas en la plataforma.

Se registra el seguimiento de una cuenta pinchando en el icono de “*Dejar de Seguir*”. Ver Figura 74.