

## MEMORIA ESCRITA DEL PROYECTO

CFGS Desarrollo de Aplicaciones Web

Autor:

Tutor: Raúl Sacristán



# Hobby Link

2S2021

## ÍNDICE DE CONTENIDOS

<b>1. INTRODUCCIÓN</b>	<b>2</b>
1.1 MOTIVACIÓN	3
1.2 ABSTRACT	3
1.3 OBJETIVOS PROPUESTOS	4
1.3.1 Objetivos específicos	4
<b>2. METODOLOGÍA UTILIZADA</b>	<b>4</b>
2.1 FASES DE DISEÑO DEL PROYECTO	5
<b>3. TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS</b>	<b>6</b>
3.1 SISTEMA OPERATIVO	6
3.2 LENGUAJE DE PROGRAMACIÓN	7
3.2.1 Librerías, framework y herramientas de Node.js	7
3.3 IDE	8
3.4 BASE DE DATOS	8
3.5 CONTROL DE VERSIONES	8
3.6 MAQUETACIÓN WEB	9
3.7 SERVICIOS EN LA NUBE	9
3.8 DISEÑO DEL TABLERO KANBAN	9
3.9 DISEÑO DE DIAGRAMAS Y BORRADORES	9
<b>4. ESTIMACIÓN DE RECURSOS Y PLANIFICACIÓN</b>	<b>10</b>
<b>5. DESARROLLO DEL PROYECTO</b>	<b>11</b>
5.1 ANÁLISIS	21
5.1.1 Requisitos funcionales	11
5.1.2 Requisitos no funcionales	12
5.2 DISEÑO	14
5.2.1 Diagramas y casos de uso	14
5.2.2 Diseño de la interfaz	19
5.3 IMPLEMENTACIÓN	21
<b>6. DESPLIEGUE Y PRUEBAS</b>	<b>24</b>
<b>7. CONCLUSIONES</b>	<b>26</b>
7.1 OBJETIVOS ALCANZADOS	26
7.2 CONCLUSIONES DEL TRABAJO	27
7.3 VÍAS FUTURAS	28
<b>8. GLOSARIO</b>	<b>31</b>
<b>9. BIBLIOGRAFÍA</b>	<b>31</b>

## ÍNDICE DE FIGURAS

Ilustración 1: Tablero Kanban. Fuente: Elaboración propia.	5
Ilustración 2: Diagrama Gantt estimación inicial. Fuente: Elaboración propia.	10
Ilustración 3: Diagrama Gantt estimación final. Fuente: Elaboración propia.	11
Ilustración 4: Diagrama Entidad-Relación. Fuente: Elaboración Propia.	13
Ilustración 5: Diagrama casos de uso (usuario no registrado). Fuente: Elaboración propia.	15
Ilustración 6: Diagrama casos de uso (usuario registrado). Fuente: Elaboración propia.	15
Ilustración 7: Esbozo menú principal. Fuente: Elaboración propia.	19
Ilustración 9: Barra superior alternativa. Fuente: Elaboración propia.	20
Ilustración 8: Formulario crear publicación. Fuente: Elaboración propia.	20
Ilustración 10: Formulario Iniciar sesión. Fuente: Elaboración propia.	20
Ilustración 11: Visor publicación con comentarios. Fuente: Elaboración propia.	21
Ilustración 12: Carpeta modelos. Fuente: Elaboración propia.	21
Ilustración 13: codificación del esquema y modelo. Fuente: Elaboración propia.	22
Ilustración 14: Carpeta controladores. Fuente: Elaboración propia.	22
Ilustración 15: Lógica del controlador comentarios.js. Fuente: Elaboración propia.	23
Ilustración 16: Carpeta Vistas. Fuente: Elaboración propia.	23

## 1. INTRODUCCIÓN

Hobby Link se trata de una plataforma web y red social que permite a sus usuarios crear y compartir publicaciones con el resto de usuarios de la plataforma, con la posibilidad de adjuntar imágenes, enlaces y ficheros en estas. Las publicaciones serán accesibles para todos los usuarios de la plataforma; podrán ser valoradas por el resto de usuarios, según criterio propio, mediante un medidor de satisfacción conformado por cinco estrellas, y un mensaje de texto con un límite de caracteres permitido. La principal función de la plataforma es servir de un espacio en el que puedan interactuar sus usuarios mediante publicaciones, sea cual fuera su contenido.

## 1.1 MOTIVACIÓN

En la actualidad, la influencia de las tecnologías e internet es enorme, la sociedad se encuentra más unida que nunca; multitudes de personas navegan la red diariamente con el propósito de formar nuevas conexiones sociales, conservar las existentes o simplemente compartir sus vivencias e inquietudes con otros individuos. El ser humano es social por naturaleza, y es por ello que, junto a la expansión de Internet, surgen las comúnmente denominadas redes sociales en línea, plataformas que pretenden facilitar la construcción de círculos sociales y la interacción entre sus integrantes, a través de la red.

Hobby Link está totalmente inspirada en las redes sociales que imperan en nuestro día a día. Pretende heredar las bases que forman parte de famosas redes sociales, como por ejemplo Twitter o Instagram, e implementar algunas de sus funciones (a una menor escala), con tal de simular parcialmente el entorno y la experiencia de usuario de la que gozan sus predecesoras.

A pesar de ser una demo con funcionalidades básicas comunes a una red social, considero que es ideal como proyecto final del ciclo para demostrar que he adquirido los conocimientos impartidos durante el grado superior, ya que por básicas que sean las funcionalidades, estas requieren forzosamente del desarrollo de una aplicación cliente-servidor, con la implementación de una base de datos, y un despliegue para que sea accesible por cualquier usuario a través de internet.

## 1.2 ABSTRACT

Hobby Link is a social media site, where users can create and publish posts with or without images, links and files attached to them. To use the web application, users are first required to register by providing a username and password (both must be nonexistent in the database). Users can access each post and leave a review made of plain text plus a star based rating indicator. Published posts are shared between all users from the platform, and each one has his own show page in which the reviews are stored. To delete or edit a previously created post or review, the user must be logged in the platform, but the delete and edit buttons will only be visible to the original author.

The main objective of Hobby Link is to be an extremely simplified version of today's most popular social media websites, such as Twitter and Instagram. It was made just for educational purposes, so I could put together everything I've learned about web development during the last two years. It uses the following technologies: Node.js in conjunction with Express.js as server side language, MongoDB (Mongoose) as NoSQL database, Passport.js for authentication, EJS as templating engine, and Bootstrap for styling and its grid system.

### 1.3 OBJETIVOS PROPUESTOS

El objetivo es realizar una aplicación web que sirva como red social en la que los usuarios puedan registrarse, crear publicaciones (texto junto a imágenes, archivos o enlaces), editar y eliminar publicaciones previamente creadas y compartir sus publicaciones con otros usuarios de la red para que éstas sean comentadas.

#### 1.3.1 Objetivos específicos

- Consolidar los conocimientos adquiridos durante el ciclo formativo.
- Ampliar mis conocimientos de **Javascript** mediante el uso de diversas librerías.
- Estudiar **EJS** como engine para el renderizado de plantillas desde el servidor.
- Crear una aplicación **CRUD** tanto para las publicaciones como para las valoraciones.
- Estudiar **NoSQL MongoDB** para la implementación de una base de datos.
- Estudiar **Mongoose** cómo ODM para MongoDB .
- Estudiar **Node.js** para el backend.
- Estudiar **Express.js** cómo framework de Node.js para facilitar el enrutamiento.
- Estudiar **Passport.js** para implementar la autenticación de usuarios.
- Crear una página web intuitiva y totalmente responsive gracias al uso de **Bootstrap**.
- Desplegar la aplicación web en la nube gracias a **MongoDB Atlas** y **Heroku**.
- Mantener un buen control de versiones gracias a **Git**.
- Usar e interiorizar la metodología de trabajo **GitFlow**.
- Aplicar buenas pautas de programación y estructuración de directorios.

## 2. METODOLOGÍA UTILIZADA

Para la realización del proyecto se ha utilizado la metodología ágil de trabajo **Kanban**.

Esta metodología de trabajo fue creada y puesta en práctica por primera vez en Toyota, una empresa de origen nipón dedicada al sector automovilístico. El término proviene del japonés

y viene a significar “tablero”. Tal y como su nombre indica, la metodología Kanban hace uso de un tablero en el que se dibujan 3 columnas (generalmente “TO DO”, “IN PROGRESS” y “DONE”) que reflejan los estados del flujo de trabajo: tareas por hacer, tareas en progreso y tareas acabadas. En cada una de estas columnas se irán colocando tarjetas visuales por cada tarea o elemento, según el estado en el que se encuentren.

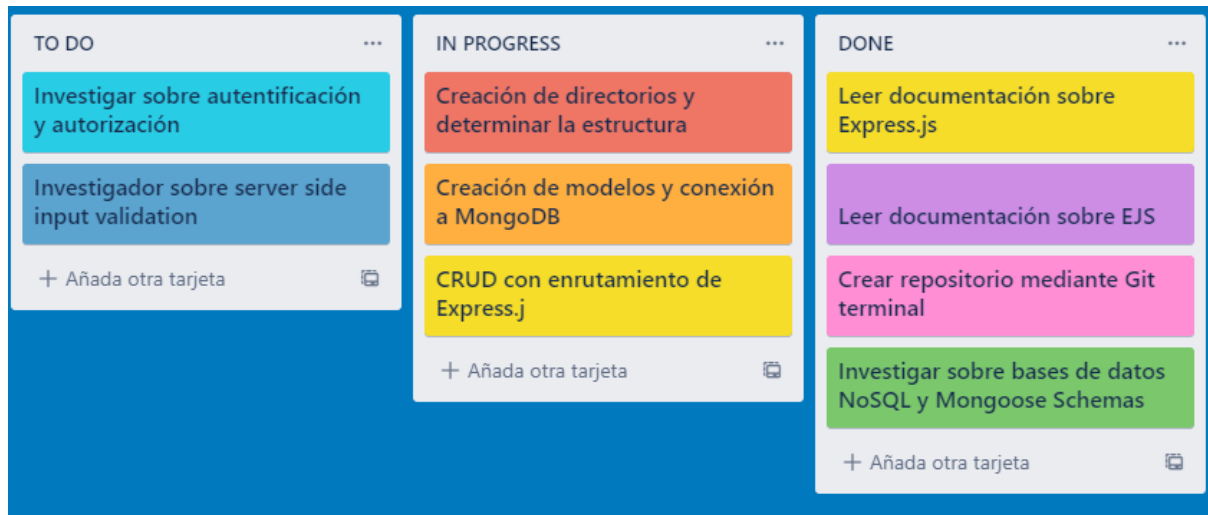


Ilustración 1: Tablero Kanban. Fuente: Elaboración propia.

Se ha escogido esta metodología ya que ofrece una representación visual del estado del proyecto y el flujo de trabajo, lo que facilita el seguimiento. Es fácil de implementar y no requiere de un gran tiempo de planificación, ideal debido al poco tiempo disponible para la realización del proyecto. Por último, otro punto a destacar es la flexibilidad, ya que permite definir tus propias tareas y establecer su prioridad respecto a otras, gracias al uso de columnas.

## 2.1 FASES DE DISEÑO DEL PROYECTO

Las fases para el desarrollo del proyecto han sido las siguientes:

- **Inicialización:** en todo ciclo de desarrollo debe haber una inicialización. Aquí se da con una idea general en la que orientar el desarrollo, teniendo en cuenta los requisitos mínimos, plazos de entrega, y otros aspectos limitantes, para garantizar un desarrollo óptimo.
- **Planificación:** se pasa a definir todas las funcionalidades con las que contará el proyecto e identificar la tecnología idónea para garantizar su futura implementación, después de estudiar e interiorizar dicha tecnología. Una vez definidas las

funcionalidades y tecnologías, nos apoyamos en el método Kanban para dividir cada recurso en pequeñas tareas representadas por tarjetas.

- **Realización de tareas:** en esta fase se van completando las tareas colocadas en el tablero Kanban. Se da prioridad a las tareas que tratan sobre el análisis y documentación de tecnologías a poner en práctica, después siguen las tareas que ponen en uso dichas tecnologías y se pasa a la codificación de pequeñas funcionalidades.
- **Control:** fase previa a la finalización del proyecto. Se revisa cada una de las tarjetas creadas desde el inicio del proyecto para asegurar que se han cumplido todos los objetivos definidos en ellas y seguidamente se pone a prueba cada funcionalidad implementada, mediante varias herramientas de testeo, para garantizar el correcto funcionamiento de la aplicación. Si durante esta etapa resulta que quedan funcionalidades o objetivos por cumplir, se detecta el mal funcionamiento de algún aspecto de la aplicación o el resultado no es sencillamente el esperado, se procede a la creación de tarjetas Kanban adicionales para hacer un seguimiento del proceso de depuración.
- **Despliegue:** una vez completadas todas las tareas (detectado por un vistazo fugaz al tablero Kanban) comienza la fase de despliegue. En esta fase el objetivo es lanzar nuestra aplicación a la red para que así sea accesible para cualquier usuario. Requiere de pequeños retoques en nuestro código para definir las respectivas variables de entorno que requieren los servicios de hosting.

### 3. TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS

#### 3.1 SISTEMA OPERATIVO

Se ha decidido desarrollar la aplicación web en el sistema operativo **Windows 10**, ya que he sido usuario asiduo desde hace bastante tiempo y estoy familiarizado con su interfaz gráfica. No obstante, reconozco que otra muy buena opción hubiera sido **Ubuntu Linux**, un sistema operativo open-source, por lo que no tiene costes de licencia, con un uso estándar a través del terminal y líneas de comandos, que al dominarlo puede mejorar notablemente la productividad en un entorno de desarrollo si lo comparamos con la productividad obtenida haciendo uso de una interfaz gráfica.

## 3.2 LENGUAJE DE PROGRAMACIÓN

El lenguaje de programación elegido para el desarrollo de la aplicación web ha sido única y exclusivamente **Javascript**, ya que es el lenguaje de programación para las aplicaciones web por excelencia. Se utiliza desde hace más de 20 años, por lo que cuenta con un gran soporte, y es útil tanto para el desarrollo frontend como para el desarrollo backend. Como es lógico, para poder ejecutar Javascript en el lado del servidor, se ha utilizado Node.js.

**Node.js** es un entorno que permite ejecutar Javascript desde nuestra propia máquina, sin necesidad de un navegador. Es especialmente potente en cuanto a velocidad de procesamiento. Goza de un gran ecosistema gracias a [npm](#), el gestor de paquetes por defecto de Node.js, que además sirve como un marketplace de herramientas y librerías para Javascript. Por último, otro de los puntos a favor, es que es capaz de manejar peticiones asíncronas, esto quiere decir que si por alguna razón topa con una operación que vaya a demorarse en el tiempo (I/O), Node.js la delega a un segundo plano y puede continuar recibiendo peticiones mientras la operación asíncrona se completa.

### 3.2.1 Librerías, framework y herramientas de Node.js

A continuación se indican todos los recursos descargados desde **npm** que han sido utilizados para la realización del proyecto:

- [joi](#): librería que permite definir esquemas para validar tipos de datos en Javascript.
- [passport](#): herramienta que incluye varias funciones con el objetivo de facilitar la autenticación de peticiones.
- [express](#): famoso web framework de Node.js que facilita la definición de rutas y el manejo de métodos HTTP.
- [ejs](#): sistema de plantillas que sirve como motor de renderizado en express. Permite generar contenido HTML con simple Javascript.
- [dotenv](#): sencilla extensión que permite hacer uso de variables de entorno durante el desarrollo.
- [helmet](#): herramienta que proporciona una capa extra de seguridad gracias a la inclusión de HTTP headers para mitigar las vulnerabilidades más comunes de las aplicaciones web.
- [method-override](#): herramienta que permite usar los métodos HTTP PUT y DELETE en lugares donde el cliente no los soporte.
- [sanitize-html](#): herramienta que proporciona una capa extra de seguridad frente ataques XSS.



- [multer](#): herramienta que hace posible implementar la función de subir archivos.

### 3.3 IDE

Para la codificación del proyecto se ha utilizado el editor de texto **Visual Studio Code**, un IDE open-source, desarrollado por Microsoft que no requiere costes de licencia. Las razones por las que se ha escogido VS code son las siguientes: es rápido, soporta una gran cantidad de lenguajes, incluye funcionalidades muy útiles para cualquier desarrollador (indentación automática, selección múltiple, paréntesis y llaves resaltadas, etc.), cuenta con su propia marketplace de add-ons y tiene una gran comunidad detrás.

### 3.4 BASE DE DATOS

Para almacenar los datos que requieran de permanencia, se ha decidido utilizar una base de datos no relacional denominada **MongoDB**. Mongo se define a sí misma como “*La base de datos líder para aplicaciones modernas*”. Al contrario que las bases de datos relacionales tradicionales, MongoDB no se guía por el modelo de filas y columnas, sino que almacena datos en documentos tipo JSON, un formato que bebe directamente y recuerda mucho a los objetos literales de Javascript, por lo que resulta muy intuitivo para cualquier desarrollador web experimentado.

Además, junto a MongoDB se hará uso del ODM **Mongoose**, soportado por Node.js. La principal ventaja de Mongoose es la abstracción respecto a Mongo *vanilla*. Esta herramienta facilita la definición de esquemas, validación de colecciones y creación de modelos de datos.

### 3.5 CONTROL DE VERSIONES

Se ha utilizado **Git** para hacer un control de versiones del proyecto. Llamamos control de versiones a las herramientas que se encargan de gestionar automáticamente los cambios realizados sobre un repositorio. Son especialmente útiles para coordinar proyectos en los que trabajan dos o más personas. Permiten deshacer cambios realizados o directamente trabajar sobre una versión temprana del proyecto. También ofrecen la posibilidad de crear ramas alternativas dentro un repositorio, de manera que si se realizan cambios, o implementan ciertas funcionalidades, la rama principal no se vea alterada.

### 3.6 MAQUETACIÓN WEB

Para definir el contenido y crear el esqueleto básico de nuestras plantillas se ha utilizado el lenguaje de marcado **HTML**. Para estilizar la presentación de dicho contenido se ha usado el lenguaje de hoja de estilos **CSS**. Además, gracias a la librería **Bootstrap** y su sistema de rejillas, se ha conseguido diseñar una web totalmente responsive.

### 3.7 SERVICIOS EN LA NUBE

Para desplegar la web, se van a requerir los servicios de almacenamiento en la nube ofrecidos por las siguientes plataformas:

- [Heroku](#): servicio cloud con soporte para aplicaciones desarrolladas en Node.js.
- [MongoDB Atlas](#): servicio cloud para bases de datos.
- [Cloudinary](#): servicio cloud para almacenar las imágenes subidas desde nuestra aplicación. Cuenta con una API muy potente que permite transformar las imágenes almacenadas mediante parámetros URL.

### 3.8 DISEÑO DEL TABLERO KANBAN

**Trello** es una herramienta muy utilizada en el mundo del desarrollo a la hora de manejar proyectos y grupos de trabajo. Permite organizar tareas mediante un sistema de tableros, de una manera muy elegante e intuitiva, por lo que agiliza el seguimiento del flujo de trabajo. Se ha utilizado para visualizar y aplicar la metodología ágil Kanban.

### 3.9 DISEÑO DE DIAGRAMAS Y BORRADORES

- **Microsoft Excel** para crear los diagramas de Gantt.
- **Moon Modeler** para construir el ERD.
- **Paint** para realizar los bosquejos de nuestra interfaz de usuario.

#### 4. ESTIMACIÓN DE RECURSOS Y PLANIFICACIÓN

Se ha hecho uso de un diagrama de Gantt para la planificación del proyecto, ya que es una herramienta ideal para exponer de forma visual el tiempo estimado o dedicado a ciertas actividades pertenecientes a un proyecto, durante un periodo largo de tiempo.

La estimación de recursos y planificación inicial se realizó teniendo como base el número total de horas asignado al módulo “Proyecto de Desarrollo de Aplicaciones Web”; es decir, 100 horas totales (redondeado de 99). Los tiempos se estimaron sobre las siguientes actividades:

- **Análisis:** desarrollar en profundidad la idea general del proyecto y definición de requisitos funcionales y no funcionales.
- **Diseño:** diseño de la interfaz de usuario.
- **Memoria y documentación:** desde la búsqueda de documentación, pasando por el aprendizaje y ampliación de conocimientos, hasta la redacción y finalización de la memoria.
- **Codificación:** desde la creación del repositorio, pasando por el desarrollo de cada funcionalidad, refactorización, corrección de bugs, hasta el despliegue de la web app.
- **Presentación** desde la creación y construcción del Prezi, hasta la grabación para la defensa del proyecto.

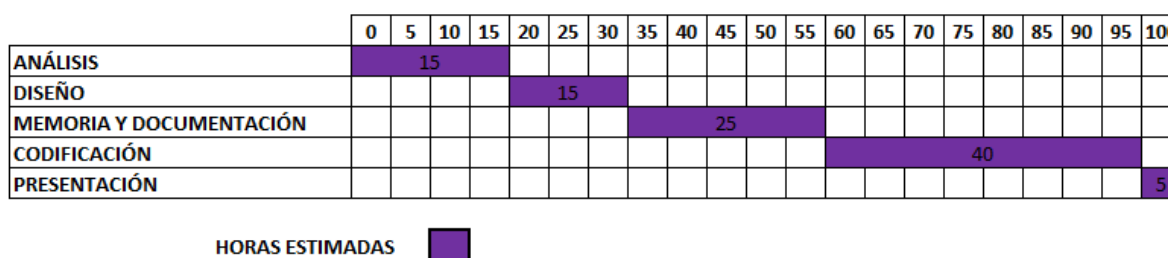


Ilustración 2: Diagrama Gantt estimación inicial. Fuente: Elaboración propia.

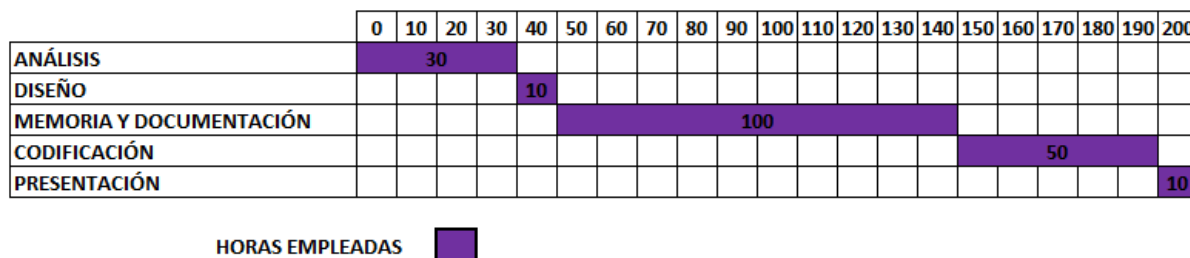


Ilustración 3: Diagrama Gantt estimación final. Fuente: Elaboración propia.

Como se puede apreciar, para finalizar el proyecto fueron necesarias un total de doscientas horas, el doble del tiempo previsto. Principalmente, el desajuste se encuentra en “MEMORIA Y DOCUMENTACIÓN”, ya que a la hora de hacer la estimación inicial no tuve en cuenta el tiempo que me iba a tomar dominar cada una de las tecnologías necesarias. También queda patente un tiempo mayor del estimado en “ANÁLISIS”, debido a que tuve dificultades a la hora de definir los requisitos, por una falta de conocimientos técnicos.

## 5. DESARROLLO DEL PROYECTO

### 5.1 ANÁLISIS

A continuación se muestran los requisitos funcionales y no funcionales:

#### 5.1.1 Requisitos funcionales

- El usuario puede registrarse en la plataforma introduciendo un nombre de usuario y una contraseña.
- El usuario puede acceder a las funciones de la plataforma identificándose con su nombre de usuario y contraseña.
- El usuario puede visualizar y acceder a las publicaciones creadas.
- El usuario identificado puede crear publicaciones.
- El usuario identificado puede añadir un comentario a su propia publicación o publicaciones creadas por otros usuarios.
- Las publicaciones estarán compuestas de un título, autor, portada y breve descripción.
- Los comentarios estarán compuestos de una nota, autor, y breve descripción.
- El usuario puede borrar su publicación.
- El usuario puede modificar su publicación.

- El usuario puede borrar su comentario.
- El usuario puede cerrar la sesión.

### 5.1.2 Requisitos no funcionales

- Se podrá acceder a la plataforma desde cualquier navegador.
- La aplicación web será responsive y con un diseño intuitivo.
- Se utilizará Bootstrap para la maquetación y estilos.
- Se utilizará MongoDB junto a Mongoose para la gestión de bases de datos.
- Se utilizará Node.js y Express como entorno servidor para desarrollar la parte lógica del proyecto.
- Se aplicará Javascript junto a EJS para el engine de plantillas renderizadas desde el servidor.
- Librería Helmet para reforzar la seguridad de la plataforma frente ataques maliciosos.
- Se aplicará el patrón de diseño MVC para desarrollar la aplicación web y estructurar los directorios.
- Se utilizará Git para el control de versiones y gestión del proyecto.
- Se utilizará Kanban para planificar y gestionar el flujo de trabajo.
- Código limpio.
- Código autoexplicativo.

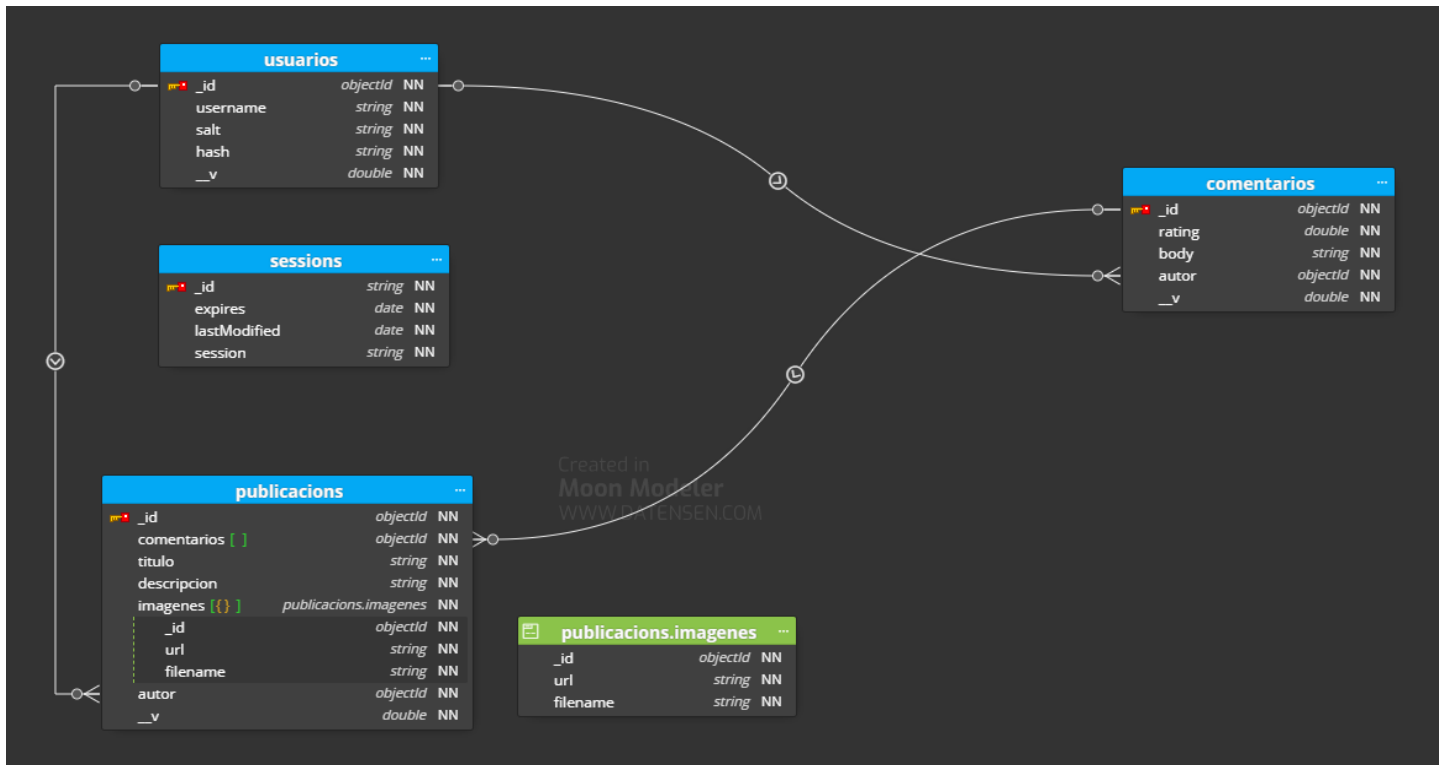
Para la elaboración del proyecto fue esencial construir un diagrama de Entidad-Relación.

Los diagramas de entidad relación son una herramienta muy utilizada a la hora de realizar un proyecto ya que nos ofrecen una representación visual de nuestro sistema de base de datos y de cómo se relacionan las entidades existentes dentro de la misma.

Cada cuadrado del diagrama representa una entidad, mientras que las líneas representan las relaciones existentes entre cada entidad.

Es importante mencionar que el sistema de gestión de bases de datos utilizado en este proyecto, MongoDB, se define como una base de datos no relacional, ya que almacena la información de una forma distinta a las tradicionales bases de datos relaciones.

Aun así, la propia información sigue relacionada entre sí. Es por eso que se ha visto oportuno realizar el ERD que se muestra a continuación:



En el diagrama se aprecian tres entidades:

- **usuarios**: se almacenan las credenciales de los usuarios registrados.
  - **\_id**: identificador único para cada usuario.
  - **username**: nombre con el que se ha registrado el usuario.
  - **salt**: medida de seguridad para añadir aleatoriedad a la contraseña cifrada del usuario en caso de que haya más de un usuario con la misma contraseña.
  - **hash**: contraseña cifrada del usuario.
- **publicaciones**: publicaciones creadas por usuarios registrados.
  - **\_id**: identificador al que se accedera para visualizar la publicación desde la plantilla, acceder a ella para editarla o borrarla.
  - **comentarios**: referencia a comentarios en el que se poblarán los comentarios.
  - **título**: introducir por el usuario desde el formulario.

- **descripción:** introducir por el usuario desde el formulario.
  - **imágenes:** las imágenes se subirán desde el equipo local del usuario hasta nuestro servicio de almacenamiento en la nube.
    - **\_id:** identificador de la imagen.
    - **url:** enlace a la imagen almacenada en cloudinary.
    - **filename:** nombre del archivo subido por el usuario.
  - **autor:** usuario que ha creado la publicación.
- **comentarios**
    - **\_id:** identificador único de cada comentario.
    - **rating:** nota introducida por el usuario desde un elemento radio.
    - **body:** texto del comentario.
    - **autor:** nombre del usuario registrado que ha creado el comentario.
- **sessions:** cookie que se almacena de manera local en el equipo de cada usuario. Gracias a la cookie, la sesión del usuario persistirá hasta que el usuario decida cerrar sesión, o hasta la fecha indicada dentro de la propiedad **expires**.

## 5.2 DISEÑO

Esta sección estará dividida en dos partes. En la primera pasaremos a mostrar los diagramas de casos de uso y los distintos casos de usos que se han tenido en cuenta a la hora de realizar el diseño de la aplicación web, y en la segunda mostraremos bosquejos de la interfaz de usuario realizados durante las fases iniciales del proyecto.

### 5.2.1 Diagramas y casos de uso

El diagrama de casos de uso es una herramienta que se emplea para agilizar la comprensión del funcionamiento de una aplicación web, como interactúa el usuario con las distintas funcionalidades de nuestra plataforma, y qué limitaciones existen según el estado del usuario. Para realizar los diagramas de casos de uso se han tenido en cuenta dos estados para un usuario: registrado y no registrado.

- Diagrama de casos de uso para el usuario no registrado:

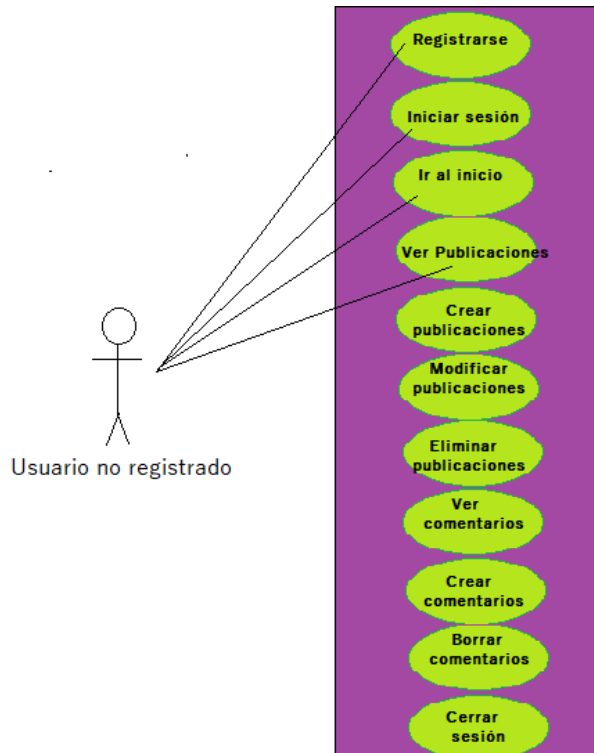
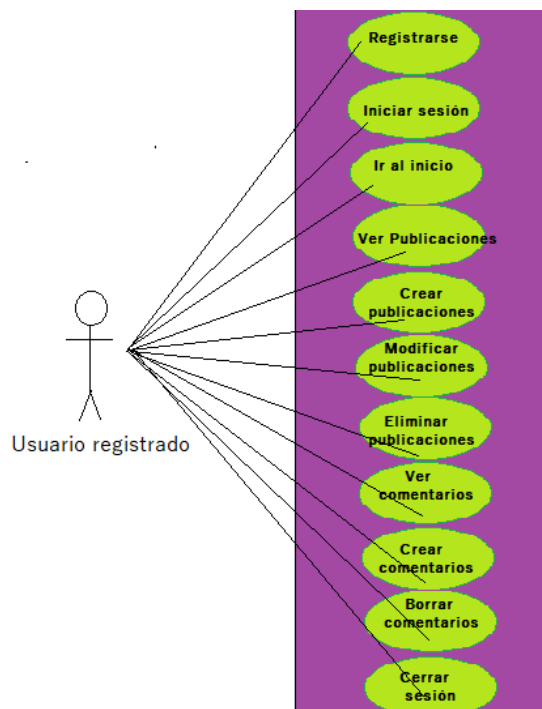


Ilustración 5: Diagrama casos de uso (usuario no registrado). Fuente: Elaboración propia.

- Diagrama de casos de uso para el usuario registrado:





## Casos de uso

<b>Caso de uso 1</b>	Ver publicaciones
<b>Precondición</b>	<ol style="list-style-type: none"> <li>1. El usuario ha accedido a la plataforma.</li> <li>2. El usuario ha hecho click sobre el botón “Ver publicaciones”.</li> </ol>
<b>Postcondición</b>	El sistema redirige al usuario a la vista donde se imprimen todas las publicaciones almacenadas en la base datos.

<b>Caso de uso 2</b>	Ver comentarios de una publicación
<b>Casos de uso relacionados</b>	Caso de uso 1
<b>Precondición</b>	<ol style="list-style-type: none"> <li>1. El usuario se encuentra en la página donde se muestran todas las publicaciones.</li> <li>2. El usuario hace click sobre el botón “Ver comentarios” de una publicación concreta.</li> </ol>
<b>Postcondición</b>	El sistema redirige al usuario a la vista donde se imprime tanto la publicación como los comentarios almacenados en la base de datos.

<b>Caso de uso 3</b>	Registro
<b>Precondición</b>	<ol style="list-style-type: none"> <li>1. El usuario no ha iniciado sesión</li> <li>2. El usuario ha hecho click sobre el botón “Registrarse”</li> <li>3. El usuario introduce nombre de usuario y contraseña.</li> <li>4. El usuario presiona el botón “registrarse”</li> </ol>
<b>Postcondición</b>	Los datos introducidos por el usuario se almacenan en nuestra bases de datos dentro de la colección usuarios. La contraseña se almacena cifrada. Nuestro servidor le devuelve al usuario una cookie con su sesión e Id para identificar más adelante y que su sesión persista.
<b>Excepciones</b>	El nombre de usuario introducido por el usuario ya existe en nuestra base de datos.

<b>Caso de uso 4</b>	Iniciar sesión
<b>Casos de usos</b>	Caso de uso 3

<b>relacionados</b>	
<b>Precondición</b>	<ol style="list-style-type: none"> <li>1. El usuario hace click sobre el botón “Iniciar sesión”</li> <li>2. El usuario introduce nombre de usuario y contraseña</li> <li>3. El usuario presiona el botón “Log-In”</li> </ol>
<b>Postcondición</b>	El usuario se ha registrado previamente
<b>Excepciones</b>	Los datos introducidos por el usuario no son correctos/no concuerdan con los almacenados en la base de datos.

<b>Caso de uso 5</b>	Crear una publicación
<b>Casos de uso relacionados</b>	Caso de uso 4
<b>Precondición</b>	<ol style="list-style-type: none"> <li>1. El usuario ha hecho click sobre el botón “Crear publicación”.</li> <li>2. El usuario tiene la sesión iniciada.</li> <li>3. El usuario escribe un título</li> <li>4. El usuario escribe una descripción.</li> <li>5. El usuario sube una imagen.</li> <li>6. El usuario presiona el botón “Añadir publicación”</li> </ol>
<b>Postcondición</b>	La lógica de nuestro servidor recupera los datos mandados por el usuario, los introduce en nuestra base de datos y redirige al usuario a la plantilla donde se imprime la publicación que ha creado.
<b>Excepciones</b>	El usuario no está registrado

<b>Caso de uso 6</b>	Editar una publicación
<b>Casos de uso relacionados</b>	Caso de uso 2, Caso de uso 5
<b>Precondición</b>	<ol style="list-style-type: none"> <li>1. El usuario hace click sobre el botón “Editar”</li> <li>2. El usuario rellena los datos nuevos.</li> <li>3. El usuario presiona el botón “Actualizar publicación”</li> </ol>
<b>Postcondición</b>	La lógica de nuestro servidor recupera los datos mandados por el usuario, busca la publicación en la base de datos gracias al identificador que se encuentra en los parámetros URL, actualiza los datos y redirige al usuario a la plantilla donde se imprime la publicación.
<b>Excepciones</b>	El usuario no es el autor de la publicación

<b>Caso de uso 7</b>	Borrar una publicación
----------------------	------------------------

<b>Casos de uso relacionados</b>	Caso de uso 2, Caso de uso 5
<b>Precondición</b>	1. El usuario ha presionado el botón “Eliminar”.
<b>Postcondición</b>	La lógica de nuestro servidor recupera el ID de la publicación, se busca en la base datos y se elimina de la colección publicaciones.
<b>Excepciones</b>	El usuario no es el autor de la publicación

<b>Caso de uso 8</b>	Crear un comentario
<b>Casos de uso relacionados</b>	Caso de uso 2, Caso de uso 5
<b>Precondición</b>	1. El usuario introduce un texto en el formulario de comentario. 2. El usuario presiona el botón “enviar”.
<b>Postcondición</b>	La lógica de nuestro servidor recupera los datos mandados por el usuario, los introduce en nuestra base de datos y redirige al usuario a la plantilla donde se imprime la publicación que ha creado.
<b>Excepciones</b>	El usuario no está registrado

<b>Caso de uso 9</b>	Borrar un comentario
<b>Casos de uso relacionados</b>	Caso de uso 8
<b>Precondición</b>	1. El usuario presiona el botón “eliminar”
<b>Postcondición</b>	Se elimina ese comentario concreto almacenado en nuestra base de datos dentro de la colección comentarios.
<b>Excepciones</b>	El usuario no es el autor del comentario

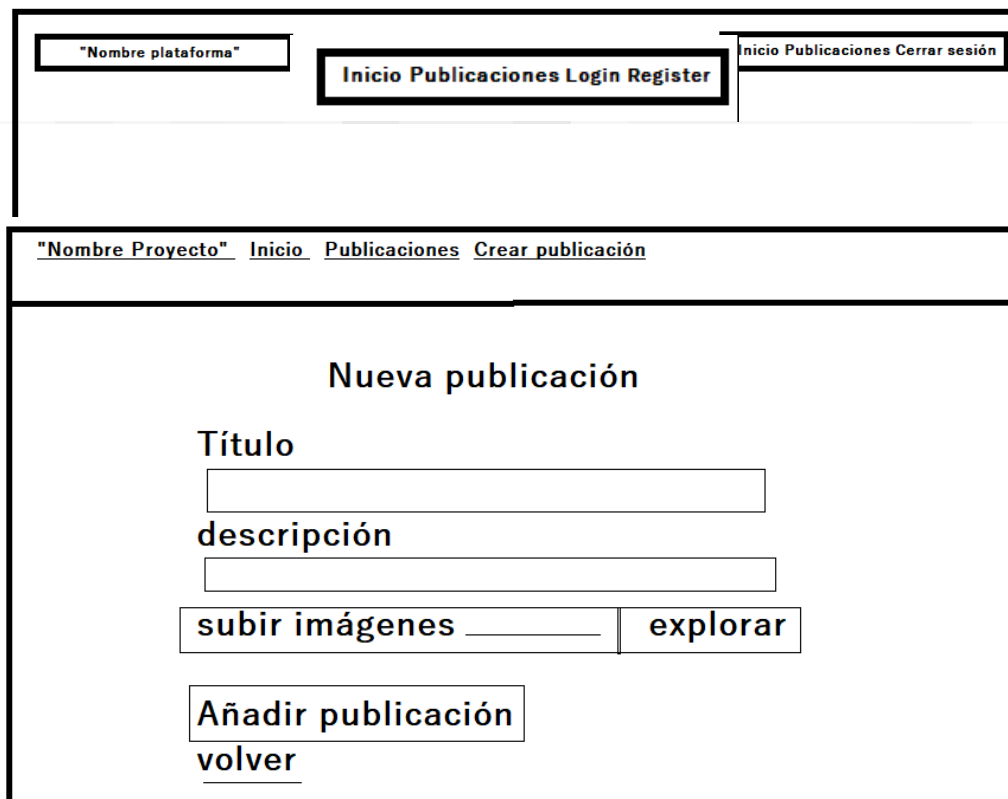
<b>Caso de uso 10</b>	Cerrar sesión
<b>Casos de uso relacionados</b>	Caso de uso 4
<b>Precondición</b>	1. El usuario presiona el botón “Cerrar sesión”.
<b>Postcondición</b>	La lógica de nuestro servidor limpia la cookie que mantiene la sesión del usuario.

<b>Caso de uso 11</b>	Volver al inicio
<b>Precondición</b>	1. El usuario se encuentra en cualquier apartado que no sea el inicio 2. El usuario presiona el botón "Inicio"
<b>Postcondición</b>	El botón redirige al usuario a nuestra página de inicio.

## 5.2.2 Diseño de la interfaz

Antes de comenzar la codificación del proyecto, se consideró necesario realizar un esquema visual de la interfaz de nuestra plataforma y cada una de sus secciones. Los borradores se crearon digitalmente mediante la herramienta de edición Paint. Son orientativos, el resultado final puede variar según cómo se desarrolle la implementación de los requisitos funcionales.

A continuación se mostrarán todas las vistas de nuestra página (se han omitido algunos formularios para evitar la repetitividad):



"Nombre plataforma"

Inicio Publicaciones Login Register

Inicio Publicaciones Cerrar sesión

"Nombre Proyecto" Inicio Publicaciones Crear publicación

Nueva publicación

Título

descripción

subir imágenes

explorar

Añadir publicación

[volver](#)

19

"Nombre Proyecto"
Inicio
Publicaciones
Crear publicación

IMAGEN

Título  
descripción

nombre del autor

autor  
☆☆☆☆☆  
comentario

autor  
☆☆☆☆☆  
comentario

autor  
☆☆☆☆☆  
comentario

autor  
☆☆☆☆☆  
comentario

"Nombre Proyecto"
Inicio
Publicaciones
Crear publicación

"IMAGEN"

Login

Usuario

Contraseña

LOGIN

### 5.3 IMPLEMENTACIÓN

Por último, en esta sección veremos en profundidad todas las partes que componen nuestro equipo, como hemos estructurado tanto los directorios como los propios ficheros, y el papel ha cobrado el patrón de diseño de MVC.

Siguiendo el patrón MVC, la estructura de nuestro proyecto se ha dividido en tres capas: vistas, modelos y controladores, de modo que el código destinado al almacenamiento nunca

se encuentra junto al código destinado a imprimir los datos y HTML. El controlador es la parte lógica de nuestro proyecto que conecta el modelo con las vistas y maneja las rutas definidas gracias a express.

La capa de modelos es donde definimos los esquemas y creamos las colecciones de nuestra base de datos. Dentro de la carpeta modelos se crean 3 ficheros para cada colección que va a componer nuestro proyecto: comentario, publicación y usuario.

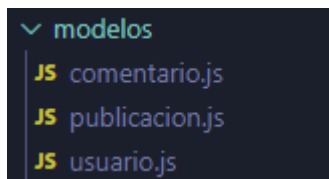


Ilustración 12: Carpeta modelos. Fuente: Elaboración propia.

Como se puede apreciar en la siguiente imagen, primero definimos el esquema de nuestro modelo. El esquema recuerda al formato de tipo JSON. En este definimos las propiedades que va a tener cada objeto creado a partir de nuestro esquema, e indicamos el tipo de dato que debe ser. Luego exportamos el modelo creado para usarlo más adelante en las otras capas, aislando así totalmente nuestro código dedicado a la creación y definición del modelo

```
const comentarioSchema = new Schema({
  body: String,
  rating: Number,
  autor: {
    type: Schema.Types.ObjectId,
    ref: 'Usuario'
  }
});

module.exports = mongoose.model("Comentario", comentarioSchema);
```

Ilustración 13: codificación del esquema y modelo. Fuente: Elaboración propia.

La capa de controladores es la encargada de responder a las peticiones que se lanzan desde nuestra aplicación por el cliente. Los controladores sirven de enlace para la capa de modelos y la capa de vistas.

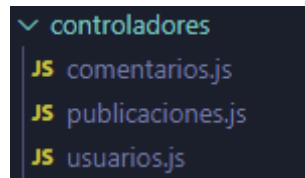


Ilustración 14: Carpeta controladores. Fuente: Elaboración propia.

En la siguiente imagen podemos ver cómo se importan los modelos “Comentarios” y “Publicacion” para ser usados por el controlador “comentarios.js”. En el controlador definimos dos funciones “createComentario” y “deleteComentario”. Ambas realizan un proceso similar, por lo que únicamente se comentará la primera:

- se guardan los parámetros que se encuentran dentro la URL en una variable.
- se crea una variable que invoca al modelo “Comentario” y a la vez crea un nuevo comentario.
- se asignan las propiedades de “comentario” mediante los parámetros de la URL
- se invoca el método “save()” sobre el comentario para guardarlo en la base de datos bajo la colección comentarios.
- redirige al usuario.

```
const Publicacion = require('../modelos/publicacion');
const Comentario = require('../modelos/comentario');

module.exports.createComentario = async (req, res) => {
  const publicacion = await Publicacion.findById(req.params.id);
  const comentario = new Comentario(req.body.comentario);
  comentario.autor = req.user._id;
  publicacion.comentarios.push(comentario);
  await comentario.save();
  await publicacion.save();
  req.flash('exito', 'Nuevo comentario creado!');
  res.redirect(`/publicaciones/${publicacion._id}`);
}

module.exports.deleteComentario = async (req, res) => {
  const { id, comentarioId } = req.params;
  await Publicacion.findByIdAndUpdate(id, { $pull: { comentarios: comentarioId } });
  await Comentario.findByIdAndDelete(comentarioId);
  req.flash('exito', 'Comentario eliminado');
  res.redirect(`/publicaciones/${id}`);
}
```

Ilustración 15: Lógica del controlador comentarios.js. Fuente: Elaboración propia.

Por último, la capa de vistas se dedica a la visualización de nuestra aplicación web. Nuestros ficheros de vistas son de formato “.ejs”, ya que se ha utilizado la librería EJS para utilizar javascript juntamente a HTML y renderizar nuestras páginas directamente desde el servidor.



Ilustración 16: Carpeta Vistas. Fuente: Elaboración propia.

## 6. DESPLIEGUE Y PRUEBAS

Se recogen a continuación todas las pruebas realizadas durante la realización del proyecto para comprobar el correcto funcionamiento de la plataforma de acuerdo a los casos de uso.

### PRUEBA 1: Registro del usuario

**Objetivo probado:** Registro

**Requisitos probados:** Se confirma que el usuario, una vez se encuentra en la ventana de registro, al rellenar el formulario y presionar el botón “Registrarse”, sus datos quedan guardados en la base de datos.

**Pruebas realizadas:** Se intenta enviar el registro introduciendo un formato de email incorrecto, o sin rellenar ninguna casilla, a lo que nuestro sistema detecta satisfactoriamente la incidencia y manda un mensaje de error al usuario.

### PRUEBA 2: Login del usuario

**Objetivo probado:** Inicio de sesión

**Requisitos probados:** Se confirma que el usuario, una vez se encuentra en la ventana



de Inicio de sesión, y rellena el formulario con sus credenciales, puede acceder a la aplicación si éstas son correctas.

**Pruebas realizadas:** Se intenta iniciar sesión introduciendo un formato de email incorrecto, sin rellenar ninguna casilla, o con credenciales que no concuerden a las guardadas en nuestra base de datos, a lo que nuestro sistema detecta satisfactoriamente la incidencia y manda un mensaje de error al usuario.

### PRUEBA 3: Visualizar publicaciones creadas

**Objetivo probado:** Correcta visualización de todas las publicaciones guardadas en la base de datos

**Requisitos probados:** Se confirma que cualquier usuario, tanto registrados como no registrados, pueden dirigirse a la vista donde se renderizan todas las publicaciones, presionando el botón “Ver publicaciones”

**Pruebas realizadas:** Presionamos el botón “Ver publicaciones” localizado en la barra superior de nuestra página. El botón nos redirige correctamente a “/publicaciones”, donde se renderizan todas las publicaciones guardadas en nuestra base de datos.

### PRUEBA 4: Cerrar sesión

**Objetivo probado:** Correcto funcionamiento del botón “Cerrar Sesión” para usuarios con la sesión iniciada.

**Requisitos probados:** Se confirma que el usuario con sesión iniciada, es capaz de visualizar el botón “Cerrar Sesión”. Al presionar dicho botón, la sesión queda eliminada y el usuario no tendrá acceso a las funciones de la plataforma.

**Pruebas realizadas:** Verificamos que el botón aparece únicamente cuando hay una sesión iniciada en el navegador, y desaparece cuando no hay ninguna sesión iniciada.

### PRUEBA 5: Crear publicación

**Objetivo probado:** Correcto funcionamiento del botón “Crear publicación” y del formulario.

**Requisitos probados:** Se confirma que únicamente los usuarios registrados pueden hacer uso del formulario para crear publicaciones. En el formulario se deben rellenar obligatoriamente todos los campos (título, descripción e imagen). La nueva publicación debe guardarse en la base de datos al enviar el formulario.

**Pruebas realizadas:** Iniciamos sesión, presionamos el botón “Crear publicación” depositado en la barra superior de nuestra página, que nos redirige a “/publicaciones/new”, dónde se encuentra el formulario para crear una nueva publicación. Procedemos a intentar enviar el formulario sin rellenar ningún campo, para verificar que nuestra aplicación realiza las validaciones pertinentes y no nos permite mandar el

formulario. Una vez rellenos los campos, mandamos el formulario, este se guarda en la base de datos y la aplicación nos redirige a la publicación recién creada.

#### PRUEBA 6: Editar publicación

**Objetivo probado:** Correcto funcionamiento del botón “Editar publicación” y del formulario.

**Requisitos probados:** Se confirma que únicamente los usuarios registrados y autores de dicha publicación pueden hacer uso del formulario para editar publicaciones. En el formulario aparecen todos los campos (título, descripción e imagen) y el autor puede cambiarlos a su antojo. La publicación en la base de datos debe actualizarse con los nuevos datos al enviar el formulario.

**Pruebas realizadas:** Iniciamos sesión, presionamos el botón “Editar” localizado en la parte inferior de nuestra publicación, que nos redirige a “/publicaciones/ID/edit”, dónde se encuentra el formulario para editar nuestra publicación. Procedemos a intentar enviar el formulario sin rellenar ningún campo, para verificar que nuestra aplicación realiza las validaciones pertinentes y no nos permite mandar el formulario. Una vez rellenos los campos, mandamos el formulario, este se guarda en la base de datos y la aplicación nos redirige a la publicación con los nuevos datos actualizados.

#### PRUEBA 7: Eliminar publicación

**Objetivo probado:** Correcto funcionamiento del botón “Eliminar” en publicaciones.

**Requisitos probados:** Se confirma que únicamente los usuarios registrados y autores de dicha publicación pueden visualizar y hacer uso del botón. Al pulsar el botón la publicación se borrará completamente de la base de datos.

**Pruebas realizadas:** Iniciamos sesión, presionamos el botón “Eliminar” localizado en la parte inferior de nuestra publicación.

#### PRUEBA 8: Comentar publicación

**Objetivo probado:** Correcto funcionamiento del botón “Enviar comentario”.

**Requisitos probados:** Se confirma que únicamente los usuarios registrados son capaces de visualizar el botón de “Enviar comentarios” al entrar en una publicación. Al rellenar el campo y presionar el botón, el comentario se guardará en la base de datos y se mostrará en pantalla.

**Pruebas a realizar:** Iniciamos sesión, presionamos el botón “Enviar” localizado en el lateral derecho de nuestra pantalla, dentro de una publicación concreta. Procedemos a intentar enviar el comentario sin texto, para verificar que nuestra aplicación realiza las validaciones pertinentes y no nos permite mandar el comentario.

## PRUEBA 9: Eliminar comentario

**Objetivo probado:** Correcto funcionamiento del botón “Eliminar” en comentarios.

**Requisitos probados:** Se confirma que únicamente los usuarios registrados y autores de dicho comentario pueden visualizar y hacer uso del botón. Al pulsar el botón el comentario se borrará completamente de la base de datos.

**Pruebas realizadas:** Iniciamos sesión, presionamos el botón “Eliminar” localizado en la parte inferior de nuestro comentario.

## 7. CONCLUSIONES

### 7.1 OBJETIVOS ALCANZADOS

Al repasar los objetivos fijados al inicio del proyecto, podemos apreciar que se han cumplido la mayoría de ellos. Se ha desarrollado y desplegado de forma satisfactoria una aplicación web que permite y exige la autenticación del usuario para llevar a cabo ciertas acciones (crear publicaciones vinculadas al mismo, eliminar o editar publicaciones propias, añadir comentarios a publicaciones pertenecientes a otros usuarios), recoge datos introducidos por el usuario y los almacena en una base de datos persistente, ofrece al usuario la posibilidad de subir una imagen junto a su publicación, contiene un sistema de valoración asociado a cada comentario para calificar a su respectiva publicación y presenta la información de forma ordenada e intuitiva.

Sin embargo, no se ha podido proceder al despliegue de la página web debido al tiempo limitado para realizar el proyecto.

### 7.2 CONCLUSIONES DEL TRABAJO

Realizar el proyecto ha supuesto un gran desafío, ya que hasta entonces no había desarrollado una aplicación web de esta escala y dudaba que con mis conocimientos pudiera obtener un resultado digno. Dar con una idea realista sobre la que realizar el proyecto también supuso un inconveniente debido a que debí debatir entre escoger un tema ambicioso con el que me sintiera satisfecho, con el riesgo añadido de ser incapaz de finalizarlo, o por lo contrario, algo más humilde que simplemente demostrara que he asentado los conocimientos impartidos en el curso. Finalmente, me decanté por un punto intermedio: una temática básica, pero desarrollarla mediante el uso de tecnologías desconocidas, con el objetivo de expandir mi formación como desarrollador web.

Al comienzo de la fase de codificación tuve muchas dificultades ya que nunca había trabajado con Node.js, ni tenía experiencia en bases de datos no relacionales. Como todo buen programador, recopilé videos, documentación, cursos y repositorios que pusieran las tecnologías en práctica. Fueron unos semanas de aprendizaje intensivas, pero muy fructíferas, ya que aprendí a trabajar con operaciones asíncronas en el entorno de Node en combinación con Express Router, estudié el patrón de diseño MVC, investigué sobre Moongoose y pude definir la estructura básica de la base de datos.

En conclusión, me encuentro muy satisfecho con el resultado del proyecto. La realización ha resultado en una experiencia muy positiva ya que me ha permitido descubrir, aprender y poner en práctica tecnologías de las que carecía de experiencia. Además, he podido ejercitar cualidades que todo programador debería tener, como la solvencia a la hora de enfrentarse a un problema, la capacidad de autoaprendizaje, o el control y gestión de versiones.

### 7.3 VÍAS FUTURAS

A continuación se muestran las posibles funcionalidades que podrían ser implementadas en el futuro:

- Historial de anteriores publicaciones visitadas por el usuario.
- Ampliar modelo de usuario para que contenga imagen de perfil y biografía.
- Implementar perfiles individuales de usuario que muestren imagen de perfil y biografía. Accesibles para otros usuarios de la red.
- Añadir etiquetas para agrupar y filtrar publicaciones.
- Buscador en el que introduzcas una etiqueta y la página muestre todas las publicaciones con esta asignada.
- Posibilidad de agregar como amigos a otros usuarios de la red.
- Implementar la posibilidad de mandar o recibir mensajes privados entre usuarios que hayan establecido una amistad en la plataforma.
- Implementar autenticación con cuentas de Google, Twitter y Facebook mediante Passport.js.
- Desplegar la página web, mediante Heroku y MongoDB Atlas, para hacerla accesible por cualquier usuario de la red.

## 8. GLOSARIO

A continuación se muestra una lista de algunos términos utilizados en la memoria junto a su definición:

- **Framework:** entorno de abstracción de un lenguaje determinado que provee nuevas formas de realizar un proyecto, estableciendo e imponiendo nuevas normas.
- **IDE:** editor de texto que se utiliza para el desarrollo de aplicaciones informáticas. Proporciona servicios y herramientas que facilitan notablemente la tarea de un programador.
- **MVC:** Modelo-Vista-Controlador es un patrón de diseño que consiste en abstraer distintas partes de nuestro código y dividir las en tres capas distintas, cada capa para una funcionalidad concreta.
- **Google:** multinacional estadounidense dedicada al sector tecnológico que ofrece varios servicios y productos.
- **Facebook:** plataforma que principalmente se utiliza para conocer gente, compartir contenido y mantenerte conectado con tus amistades a través de Internet.
- **Twitter:** plataforma que principalmente se utiliza para crear contenido y compartirlo con otras personas mediante mensajes de texto con un límite de caracteres establecido.
- **Instagram:** plataforma que principalmente se utiliza para crear contenido y compartir imágenes con otros usuarios.
- **Demo:** término usado principalmente en el mundo de la informática para referirse a un prototipo.
- **Engine:** motor de renderizado utilizado para procesar la información requerida y mostrarla en pantalla al usuario.
- **Servidor:** parte lógica de una aplicación encargada de conectar al cliente con los datos y ofrecer determinados servicios.
- **CRUD:** término para referirse a las funciones básicas en bases de datos: “Crear, Leer, Actualizar y Borrar”
- **Backend:** término utilizado en el sector tecnológico para referirse a las tecnologías del lado del servidor.
- **Router (express):** utilidad que ofrece el framework Express para facilitar el manejo de las rutas de nuestra aplicación.
- **Responsive:** es un término que hace referencia a la adaptabilidad de una aplicación o tecnología a cualquier dispositivo.

- **Directorio:** es la dirección en la que se almacenan nuestros ficheros y carpetas.
- **Testing:** término que hace referencia a las pruebas técnicas que se realizan sobre una aplicación para detectar errores.
- **Hosting:** servicio que provee almacenamiento en internet para cualquier tipo de aplicación o fichero.
- **Open-Source:** “código abierto”. Es una práctica de desarrollo que pretende dejar el código a la disponibilidad de cualquier persona y de manera gratuita.
- **Terminal:** consola virtual en la que podemos interactuar con el equipo mediante el uso de comandos introducidos con el teclado.
- **Marketplace:** plataformas digitales que hacen de intermediario entre compradores y vendederos. Existen distintos tipos de marketplaces según el tipo de oferta: productos, servicios o trabajos.
- **I/O:** abreviatura de *input/output*, entrada/salida en castellano, hace referencia principalmente a las peticiones HTTP, y a las operaciones de lectura y escritura a una base de datos.
- **Librería:** en el mundo de la programación, el término librería hace referencia a una librería de instrucciones o programas. Una librería es un conjunto de funciones y clases creada con el objetivo de facilitar la programación.
- **HTTP:** abreviatura de *Hypertext Transfer Protocol*, protocolo de transferencia de hipertextos en castellano, es un método que permite la transferencia de datos a través de hipertexto (contenidos de una página web) desde un servidor, mediante la red, hasta un cliente.
- **HTML:** abreviatura de *HyperText Markup Language*, lenguaje de marcado de hipertexto en castellano, permite construir páginas web gracias al uso de secciones, párrafos, encabezados, enlaces, etc. Es esencial para definir la estructura básica.
- **XSS:** abreviatura de *Cross-site scripting*. Es un ataque cibernético que explota alguna vulnerabilidad de una página web para así introducir scripts dañinos con el objetivo de modificar dicha página web o acceder a información sensible.
- **CSS:** abreviatura de *Cascading Style Sheets*, Hoja de estilos en cascada en castellano, es un lenguaje de marcas utilizado durante el desarrollo de páginas web con el objetivo de mejorar la presentación gráfica y diseño de un archivo HTML.
- **Add-on:** complemento relacionado con una aplicación o programa que se añade con el objetivo de obtener una funcionalidad específica.
- **JSON:** abreviatura de *JavaScript Object Notation*, es un formato de texto utilizado para el intercambio de datos. Su estructura recuerda a los objetos literales provenientes del lenguaje de programación Javascript.

- **ODM:** abreviatura de *Object Data Model*, es una técnica para hacer consultas o realizar operaciones CRUD a una base de datos, principalmente bases de datos no relacionales, haciendo uso de un paradigma orientado a objetos literales.
- **API:** abreviatura de *Application Programming Interface*, interfaz que permite y facilita que aplicaciones distintas puedan comunicarse.
- **URL:** abreviatura de *Uniform Resource Locator*, Localizador de Recursos Uniforme en castellano, es una dirección que lleva a un recurso concreto localizado en la Web.
- **Cloud:** forma abreviada de *cloud computing*, nube en castellano, hace referencia a servidores remotos que ofrecen servicios como almacenamiento o streaming.
- **Repositorio:** término utilizado para referirnos a una localización (servidor, directorio, etc.) en el que están alojados nuestros proyectos de desarrollo.
- **HTTP headers:** parámetros enviados en una petición o respuesta HTTP al cliente o al servidor que contienen información de la transacción.
- **Indentación:** desplazar un bloque de código hacia la derecha insertando tabulaciones, con el objetivo de separarlo del margen izquierdo y distinguirlo mejor de los bloques de código adyacentes.
- **ERD:** abreviatura de Entity Relationship Diagram, Diagrama de Entidad-Relación en castellano, herramienta que nos ofrece una representación visual de nuestro sistema de base de datos y de cómo se relacionan las entidades existentes dentro de la misma.
- **Cookie:** archivo almacenado en el navegador que se envía entre un cliente y servidor. Se usan para identificar al usuario en un sitio web específico y guardar información sobre su comportamiento.
- **Log-In:** proceso que permite al usuario registrarse, identificarse e ingresar a la aplicación mediante unas credenciales provistas por el usuario.

## 9. BIBLIOGRAFÍA

Sobre recursos de programación web:

- [MDN Web Docs](#)
- [W3Schools](#)
- [SoftAuthor](#)

Sobre metodologías de trabajo y Kanban:

- <https://okhosting.com/blog/metodologias-del-desarrollo-de-software/>
- <https://www2.deloitte.com/es/es/pages/technology/articles/que-es-metodologia-kanban.html>

- <https://www.apd.es/metodologia-kanban/>
- <https://blog.becas-santander.com/es/metodologias-desarrollo-software.html>
- <https://www.fhios.es/metodologia-kanban-pros-y-contras/>
- <https://www.viewnext.com/el-ciclo-de-vida-de-las-metodologias-agiles-de-desarrollo/>

Sobre bases de datos:

- [Relational databases vs NoSQL document databases](#)

Sobre cursos Udemy:

- [Intro to Git - Ian Schoonover](#)

Sobre comunidades y resolución de dudas:

- [Stack Overflow](#)
- [/r/javascript/](#)
- [/r/webdev/](#)

Sobre diseño y maquetación web:

- [CSS-TRICKS](#)

