

# TRABAJO FIN DE GRADO



**UCAM**

UNIVERSIDAD CATÓLICA  
DE MURCIA

## ESCUELA POLITÉCNICA SUPERIOR

*Grado en Ingeniería Informática*

---

PublishWhere: aplicación web de publicación en  
múltiples redes sociales

*Autor/a:*

*D. Ronald Ernesto Tejada Ríos*

*Director/a:*

Dra. Dña María Magdalena Cantabella Sabater

*Murcia, mayo de 2024*

Ronald Ernesto Tejada Ríos

# TRABAJO FIN DE GRADO



**UCAM**  
UNIVERSIDAD CATÓLICA  
DE MURCIA

## ESCUELA POLITÉCNICA SUPERIOR

*Grado en Ingeniería Informática*

---

PublishWhere: herramienta de publicación en múltiples  
redes sociales

*Autor/a:*

*D. Ronald Ernesto Tejada Ríos*

*Director/a:*

Dra. Dña. Magdalena Cantabella Sabater

*Murcia, mayo de 2024*

<https://PublishWhere.com>







## ÍNDICE

<b>RESUMEN .....</b>	<b>13</b>
<b>1. INTRODUCCIÓN .....</b>	<b>17</b>
1.1. Motivación .....	17
1.2. Definición .....	17
1.3. Objetivos propuestos (generales y específicos).....	18
<b>2. ESTADO DEL ARTE .....</b>	<b>19</b>
2.1. Conceptos relevantes del dominio de aplicación. ....	19
2.2. Relación con proyectos con la misma funcionalidad.....	21
2.2.1. Planes gratuitos de proyectos similares.....	22
2.2.2. Planes de pago de proyectos similares .....	22
2.2.3. Costo anual de referencia del mejor proyecto similar .....	23
2.3. Estudio de viabilidad .....	24
2.3.1. Alcance del proyecto.....	24
2.3.2. Estudio de la situación actual .....	26
2.3.3. Estudio y valoración de las alternativas de solución .....	28
2.3.4. Selección de la solución .....	28
<b>3. METODOLOGÍAS USADAS.....</b>	<b>29</b>
3.1. Metodologías tradicionales .....	29
3.2. Metodologías ágiles .....	29
3.3. Selección de metodología .....	29
3.4. Metodologías ágiles .....	30
3.5. Scrum.....	31
3.5.1. Roles Scrum .....	31
3.5.2. Historias de usuario .....	32
<b>4. TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS EN EL PROYECTO</b>	<b>33</b>
4.1. Control de versiones y gestión ágil .....	33

4.2.	Full Stack Framework .....	33
4.2.1.	Base Datos .....	34
4.2.2.	Backend.....	34
4.2.3.	Frontend .....	35
4.3.	Herramientas de desarrollo .....	36
4.4.	Infraestructura y servicios .....	37
4.5.	Hardware .....	38
<b>5.</b>	<b>ESTIMACIÓN DE RECURSOS Y PLANIFICACIÓN .....</b>	<b>39</b>
5.1.	Sprint 0.....	39
5.1.1.	Equipo Scrum .....	39
5.1.2.	Product Backlog.....	39
5.1.3.	Estimación temporal del proyecto .....	42
5.1.4.	Estimación económica del proyecto.....	44
<b>6.</b>	<b>Desarrollo del proyecto .....</b>	<b>47</b>
6.1.	Sprint 0.....	47
6.1.1.	Planificación del sprint .....	47
6.1.2.	Desarrollo del sprint .....	47
6.1.3.	Revisión del sprint.....	48
6.1.4.	Retrospectiva del sprint .....	49
6.2.	Sprint 1.....	49
6.2.1.	Planificación del sprint 1 .....	49
6.2.2.	Desarrollo del sprint 1 .....	50
6.2.3.	Revisión del sprint 1.....	52
6.2.4.	Retrospectiva del sprint 1 .....	53
6.3.	Sprint 2.....	54
6.3.1.	Planificación del sprint 2 .....	54
6.3.2.	Desarrollo del sprint 2 .....	54



6.3.3.	Revisión del sprint 2.....	56
6.3.4.	Retrospectiva del sprint 2 .....	57
6.4.	Sprint 3.....	57
6.4.1.	Planificación del sprint 3 .....	57
6.4.2.	Desarrollo del sprint 3 .....	58
6.4.3.	Revisión del sprint 3.....	60
6.4.4.	Retrospectiva del sprint 3 .....	61
6.5.	Sprint 4.....	61
6.5.1.	Planificación del sprint 4 .....	61
6.5.2.	Desarrollo del sprint 4 .....	62
6.5.3.	Revisión del sprint 4.....	64
6.5.4.	Retrospectiva del sprint 4 .....	65
6.6.	Sprint 5.....	65
6.6.1.	Planificación del sprint 5 .....	65
6.6.2.	Desarrollo del sprint 5 .....	66
6.6.3.	Revisión del sprint 5.....	66
6.6.4.	Retrospectiva del sprint 5 .....	67
<b>7.</b>	<b>CONFIGURACIÓN Y DESPLIEGUE DE LA SOLUCIÓN.....</b>	<b>69</b>
7.1.	Dominio y VPS.....	69
7.2.	Configuración de seguridad básica.....	69
7.3.	Instalar software en el VPS.....	70
7.4.	Configurar repositorio.....	70
7.5.	Monitorización y rendimiento .....	71
7.6.	Continuos Integration / Continuos Deployment (CI/CD).....	72
<b>8.</b>	<b>CONCLUSIONES .....</b>	<b>75</b>
8.1.	Objetivos alcanzados .....	75
8.2.	Conclusiones del trabajo y personales.....	76

8.3. Vías futuras.....	77
<b>9. Bibliografía .....</b>	<b>78</b>

## ÍNDICE DE ELEMENTOS GRÁFICOS

### **TABLA**

Tabla 1 Proveedores de redes sociales de proyecto similares.....	21
Tabla 2 Cantidad de usuarios, marcas y redes sociales del plan gratuito de proyectos similares.....	22
Tabla 3 Costos de proyectos similares de sus planes de pago.....	23
Tabla 4 Comparación metodologías ágiles vs tradicionales.....	30
Tabla 5 Comparativa React, Angular y Vuejs (Navarro, 2023).....	35
Tabla 6 Roles Scrum.....	39
Tabla 7 Product Backlog Original .....	40
Tabla 8 Equivalencias de tareas y puntos de historia .....	42
Tabla 9 Puntos de Historia aplicando MoSCoW.....	43
Tabla 10 Planificación temporal MoSCoW .....	43
Tabla 11 Planificación MoSCoW optimista.....	44
Tabla 12 Planificación MoSCoW pesimista .....	44
Tabla 13 Costos indirectos del proyecto .....	45
Tabla 14 Costos variables recurrentes anuales del proyecto .....	46
Tabla 15 Tareas del sprint 0.....	47
Tabla 16 Tareas del sprint 0 - revisión .....	48
Tabla 17 Tareas del sprint 1 .....	49
Tabla 18 Tareas del sprint 1 - revisión .....	53
Tabla 19 Tareas del sprint 2.....	54
Tabla 20 Tareas del sprint 2 - revisión .....	56
Tabla 21 Tareas del sprint 3.....	57
Tabla 22 Tareas del sprint 3 - revisión .....	60
Tabla 23 Tareas del sprint 4.....	61
Tabla 24 Tareas del sprint 4 - revisión .....	64
Tabla 25 Tareas del sprint 5.....	66
Tabla 26 Tareas del sprint 5 - revisión .....	66

## **ILUSTRACIONES**

Ilustración 1 Tendencia costo anual Later con marcas extras .....	24
Ilustración 2 Esquema SSL/TLS usando Cloudfare (Cloudfare, s.f.) .....	26
Ilustración 3 Ventajas y desventajas del uso directo del proveedor de una red social .....	27
Ilustración 4 Ventajas y desventajas del uso de un proyecto similar .....	27
Ilustración 5 Benchmark E/S - Node.js vs .NET .....	35
Ilustración 6 Costos de Hostinger.....	45
Ilustración 7 Costo anual del bucket usando AWS S3 .....	46
Ilustración 8 Diagrama de microservicios .....	47
Ilustración 9 Tarea de jira .....	50
Ilustración 10 Diagrama entidad relación .....	51
Ilustración 11 Entidades Next-Auth .....	51
Ilustración 12 Agregar miembros al equipo .....	55
Ilustración 13 Oauth Flow .....	56
Ilustración 14 Verificación de App - Business Manager – META.....	58
Ilustración 15 Requisitos para el uso de permisos de una aplicación de META .....	59
Ilustración 16 Graph API Explorer - META Developers.....	59
Ilustración 17 Biblioteca digital por marca .....	63
Ilustración 18 Calendario de publicaciones - sin funcionalidad .....	64
Ilustración 19 Configuración del DNS y el dominio.....	69
Ilustración 20 PM2 Monitoring .....	71
Ilustración 21 UptimeRobot .....	72

## RESUMEN

Las redes sociales están en la vida cotidiana de las personas. Obligando a empresas y personas a usarlas para aprovechar y mejorar las oportunidades de negocio o conexiones. En este trabajo se tiene como objetivo desarrollar una aplicación web para publicar en múltiples redes sociales. Esta aplicación permite un número ilimitado de usuarios, de equipos y de marcas. Facilitando la publicación y planeación de contenido por medio de una biblioteca digital y calendario de publicaciones. Durante el desarrollo de este proyecto se ha implementado la metodología de Scrum para llevar el seguimiento del proyecto en conjunto de las partes interesadas, cliente y equipo de desarrollo. Se han realizado 5 sprints y como resultado se ha creado una aplicación web para escritorio que sido publicada en el internet para el acceso de usuarios finales.

**Palabras claves:** redes sociales, desarrollo web, microservicios, integración continua, servicios en la nube.



## **ABSTRACT**

Social networks are in people's daily lives. Forcing companies and individuals to use them to take advantage and improve business opportunities or connections. In this work, we aim to develop a web application to publish on multiple social networks. This application allows an unlimited number of users, teams, and brands. Facilitating the publication and planning of content through a digital library and calendar of publications. During the development of this project, the Scrum methodology was implemented to keep track of the project together with the stakeholders, product owner, and development team. Five sprints have been carried out and as a result, a desktop web application has been created and published on the Internet for end-user access.

**Keywords:** social networks, web development, microservices, continuous integration, cloud services





## 1. INTRODUCCIÓN

### 1.1. Motivación

En la actualidad las redes sociales están en auge y tendencia. Creando la necesidad de tener presencia en múltiples redes sociales por medio del contenido que se publica en cada una de ellas, siendo esto una tarea relativamente tediosa si hay que hacerlo una vez por cada red social. Lo que conlleva a su vez tener muchas pestañas del navegador abiertas o aplicaciones instaladas, adaptarse a cada interfaz, el tiempo de subir los archivos y todos los demás procesos que podrían implicarse entre medio.

En la actualidad existen varias soluciones informáticas web que solventan este problema y podrían facilitar la vida a los creadores de contenido. Sin embargo, en su mayoría estas son de pago o bien tienen limitaciones en cuanto a la cantidad de cuentas de redes sociales que una persona puede registrar, la cantidad de miembros de un equipo o bien limitaciones en cuanto a la cantidad de publicaciones mensuales que se pueden hacer.

### 1.2. Definición

Este trabajo de fin de grado se ha desarrollado una aplicación web capaz de tener registradas a tu disposición cuentas de redes sociales ilimitadas. La aplicación permite gestionar estas redes sociales a través de marcas y equipos de trabajo sin límite de usuarios. Obteniendo a largo plazo y escalabilidad costes menores que tener que usar una de las plataformas ya existentes que cobran por cantidad de redes sociales o usuarios registrados. Cumpliendo además las funcionalidades específicas extras requeridas por el cliente. Siendo el cliente objetivo final una agencia digital.

Como cliente y producto owner se ha tenido la colaboración de Marcela Peraz. Licenciada en Marketing y Dirección Comercial graduada de la UCAM e influencer reconocida con millones de seguidores en redes sociales por su talento en el maquillaje artístico.

La funcionalidad del proyecto radica en publicar contenido en múltiples redes sociales de manera sencilla y genérica integrado en una sola aplicación

web. Como funcionalidades extras del proyecto se tiene una biblioteca de contenido para visualizar los archivos multimedia de las marcas registradas, así como hacer seguimiento de los archivos si han sido usados en publicaciones, programados o que aún no ha sido usados. Además, la aplicación ofrece calendarios de contenido donde se encuentran las publicaciones que se han hecho en la plataforma. Estas publicaciones son administrables entre los miembros de una marca, pero solo de lectura para terceros sin necesidad de estar registrados, sino a través de enlaces públicos con tiempo de expiración.

Finalmente, la aplicación final ha sido hospedada en un servidor privado al que se puede acceder por medio del siguiente dominio <https://publishwhere.com>.

### **1.3. Objetivos propuestos (generales y específicos)**

El primer nivel representa objetivos generales (OG) y el segundo nivel los objetivos específicos (OE) asociados a su respectivo objetivo general:

- OG1- Desarrollar una aplicación web para publicar contenido en múltiples redes sociales.
  - OE1.1 Administrar marcas con una cantidad ilimitada de marcas dentro de la aplicación.
  - OE1.2 Disponer de una biblioteca compartida por marca de archivos multimedia.
  - OE1.3 Permitir el registro ilimitado de usuario dentro de la aplicación
- OG2- Hospedar la aplicación en internet para ser accedida por los usuarios finales.
  - OE2.1 Implementar una arquitectura de microservicios para mejor escalabilidad.
  - OE2.3 Implementar medidas de seguridad al servidor
  - OE2.3 Automatizar los despliegues al servidor

## 2. ESTADO DEL ARTE

### 2.1. Conceptos relevantes del dominio de aplicación.

En el software para la gestión de redes sociales hay conceptos que se mantienen constantes, aunque en ocasiones llamados por diversos nombres. A medida de unificarlos y mantener términos constantes a lo largo del trabajo, se han estandarizado los términos de la siguiente manera:

- **Proveedor de red social:** se entiende como el nombre de una red social. Por ejemplo, YouTube, Twitter (X), Facebook, Instagram, etc.
- **Usuario:** es la persona final que usará la aplicación creada. Un usuario es capaz de crear marcas. Además, dentro de las marcas que pertenece es capaz de archivos a la biblioteca digital, agregar redes sociales, así como crear o programar publicaciones.
- **Marca:** es un conjunto de redes sociales agrupadas para administrarse en conjunto. Esta posee un usuario administrador de la marca y un equipo de usuarios. Una marca funciona como una agrupación de cuentas de redes sociales. Cada marca posee una biblioteca digital propia de archivos multimedia compartidos para el uso interno de las redes sociales dentro de la marca.
- **Administrador:** se entiende como administrador a la persona que crea la marca dentro de la aplicación. Es un usuario encargado de administrar los miembros del equipo de su marca. Únicamente esta persona tiene permisos para borrar su marca o cambiar de administrador.
- **Equipo:** es un conjunto de usuarios asociados a una marca. Estos usuarios se encargan de agregar o eliminar redes sociales, subir archivos a la biblioteca digital.
- **Biblioteca digital:** es el nombre otorgado al espacio virtual propio que cada marca posee donde se encuentran los archivos multimedia (imágenes y videos). Estos archivos se usarán posteriormente para crear publicaciones o programarlas y para compartir archivos entre miembros del equipo.

- **Red social:** entiéndase como una página de Facebook, cuenta de Instagram Business o Profesional, una cuenta de TikTok, un canal de YouTube o una cuenta de Twitter (X).
- **Publicación:** se refiere al contenido que va a ser subido a una red social. Este puede ser como un archivo multimedia individual o múltiples, acompañado opcionalmente de texto. Existen casos en los que se podría subir una mezcla de texto, fotos y videos, dependiendo de los tipos de publicaciones disponibles.

Estos serían los conceptos generales para un entendimiento global de la aplicación. Sin embargo, existen también conceptos más técnicos que son relevantes a lo largo del desarrollo de la aplicación. Estos son los siguientes:

- **Oauth:** es un protocolo de autorización que permite iniciar sesión en aplicaciones propias a través de autenticarte en un servicio de un tercero sin necesidad de compartir las credenciales reales del usuario, como la contraseña, en la aplicación propia. Como respuesta del protocolo se recibe un token que contiene información sobre los privilegios recibidos del servicio externo. Este token posee un tiempo límite de duración, tras terminarse el usuario tiene que volver a iniciar sesión y obtener un nuevo token (Cloudflare, s.f.).
- **Microservicios:** se refiere a software organizado por módulos para el desarrollo de aplicaciones complejas. Los microservicios son una arquitectura flexible, escalable y ágil. Estas al ejecutarse operan como instancias por microservicio (módulo), pueden estar en un solo servidor o en múltiples. A medida que se requiere más recursos pueden crearse nuevas instancias del módulo para escalar el software según la demanda o bien dejarse definidas desde el inicio una cantidad fija de instancias. Siendo una solución beneficiosa para proyectos donde ciertos módulos requieran mayor cantidad de procesamiento o recursos (Intel, s.f.).
- **VPS:** por sus siglas en el inglés VPS significa “Virtual Private Server”. Un VPS posee recursos dedicados de memoria, núcleos de CPU, espacio de almacenamiento, IP pública o compartida. Los VPS son una excelente

solución por su nivel de personalización y control pues permiten a los usuarios instalar el software que necesiten.

- **Dominio:** es un identificador único para un sitio web. Se asocia por medio de un DNS (Sistema de Nombres de Dominio) a un IP para facilitar su búsqueda y acceso en del internet.
- **Bucket:** término asociado al servicio de “AWS S3” (AWS, s.f.). Se entenderá como bucket a lo largo del proyecto al contenedor escalable proporcionado como servicio de almacenamiento por una entidad tercera. En el bucket se almacenan los archivos multimedia subidos por los usuarios por medio de la aplicación web creada.

## 2.2. Relación con proyectos con la misma funcionalidad

Para este apartado se van a analizar 3 cuentas aplicaciones similares, donde se analizará sus capas gratuitas y sus planes de pago. Tomando un enfoque especial a los planes gratuitos donde nos interesa ver costos por miembros extras del equipo, red social extra o bien marca extra.

Es importante aclarar que los proyectos que se mencionan a continuación ofrecen otras funcionalidades extras que no se abordan en este trabajo. Por lo cual, su alto costo es justificado por sus otras funcionalidades. Sin embargo, para el alcance de este proyecto se ha decidido centrarse en la cantidad de cuentas de redes sociales, cantidad de marcas y la cantidad de usuarios para trabajar en equipo.

Cada proyecto implementa distintas redes sociales internamente. A continuación, se presentan en la tabla 1 los proyectos a analizar y sus proveedores de redes sociales implementados:

*Tabla 1 Proveedores de redes sociales de proyecto similares*

Proveedor	Later	Hootsuite	Planoly	Agorapulse
Instagram	x	x	x	x
Facebook	x	x	x	x
Twitter (X)	x	x	x	x

Pinterest	x	x	x	x
TikTok	x	x	x	x
LinkedIn	x	x	x	x
YouTube	x	x	x	x
Google My Business	-	-	-	x

### 2.2.1. Planes gratuitos de proyectos similares

En primera instancia se analizan los planes gratuitos de la competencia.

Tabla 2 Cantidad de usuarios, marcas y redes sociales del plan gratuito de proyectos similares

Proyecto	Cant. Marcas	Cant. Redes Sociales	Cant. Usuarios
Later	-	-	-
Hootsuite	-	-	-
Planoly	-	-	-
Agorapulse	-	3 (I)	1

Para este análisis de las tablas 2 y 3 se entiende que (I) representa a redes sociales independientes. Es decir, es posible tener dos o más cuentas de redes sociales del mismo proveedor. Por ejemplo, dos cuentas de Instagram y una de Facebook o bien tres cuentas de Facebook. Por otra parte, (U) representa que son redes sociales únicas por proveedor. Por ejemplo, no es posible tener dos cuentas de Instagram dentro de la misma marca.

### 2.2.2. Planes de pago de proyectos similares

Los distintos proyectos ofrecen varios planes a medida. Para la tabla 3 se considera el plan que:

- Acepte agregar marcas extras
- Acepte agregar redes sociales extras
- Acepte agregar usuarios extras
- En caso de que no cumpla los requisitos anteriores, se elegirá el plan que más se acerque a la cantidad de usuario, marcas y redes sociales de los planes de los otros proyectos que sí cumplen estos requisitos.

Tabla 3 Costos de proyectos similares de sus planes de pago

Proyecto	Costo mensual	Costo anual	Cant. Marcas	Cant. Redes Sociales	Cant. Usuarios	Precio marca extra	Precio red social extra	Precio usuario extra
Later	\$ 45,00	\$ 360,00	3	7 (U)	3	\$ 10,00	-	\$ 3,33
Hootsuite	\$ 266,60	\$ 3.199,14	-	20 (I)	3	-	Negociable	Negociable
Planoly	\$ 43,00	\$ 438,00	2	7 (U)	6	-	-	-
Agorapulse	\$ 69,00	\$ 588,00	-	10 (I)	1	-	\$ 15,00	\$ 69,00

Para unificar la moneda de la tabla 3 los montos económicos se encuentran en dólares. Para la conversión de euros a dólares se ha usado la tasa de cambio de 1 USD = 0.9340 EUR (Xe, 2024).

En la tabla 3, los precios extras por marca, red social o usuario son precios mensuales.

### 2.2.3. Costo anual de referencia del mejor proyecto similar

En cuanto a costo-solución, la mejor alternativa que se observa en la tabla 3 que cumple los requerimientos buscados es Later. Como referencia, se calculará el costo anual asumiendo 10 usuarios y 20 marcas.

*costo anual = costo anual base*

*+ (marcas extras \* precio marca extra + usuarios extras \* precio usuario extra) \* 12*

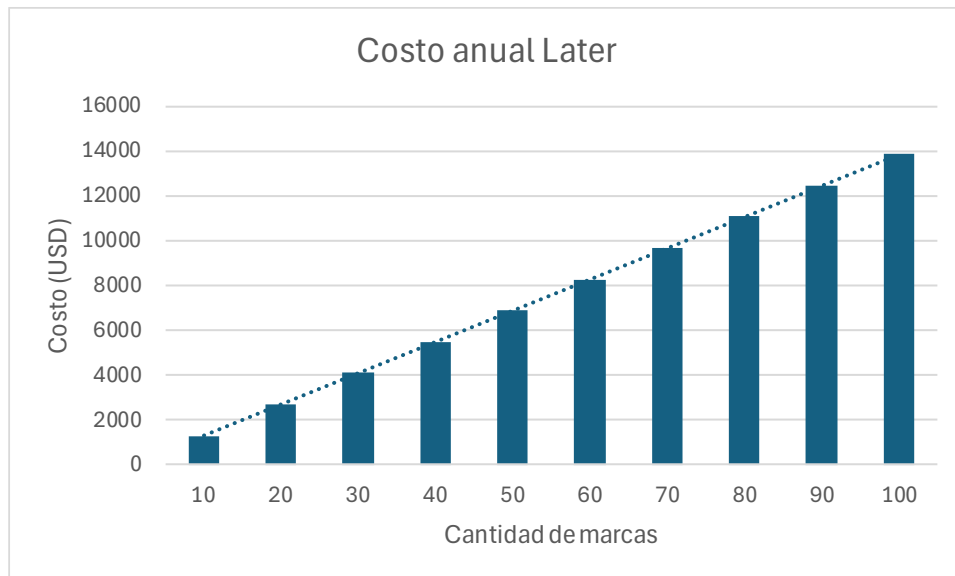
*costo anual = 360 + ((20 - 3) \* 10 + (10 - 3) \* 3.33) \* 12*

*costo anual = 2679.72 USD = 2502.86 EUR*

Si fuese 15 usuarios y 30 marcas o 20 usuarios y 40 marcas, el costo anual asciende a 4079,52 USD (3810.27 EUR) y 5479.32 USD (5117.68 EUR).

Si mantenemos una tendencia de que por cada cantidad de usuarios hay el doble de marcas. Se puede observar la tendencia de costos anuales de Later como se muestra a continuación en la ilustración 1. Concluyendo que el costo

anual puede llegar a ser considerablemente alto a medida que la cantidad de marcas y miembros del equipo aumenta.



*Ilustración 1 Tendencia costo anual Later con marcas extras*

## 2.3. Estudio de viabilidad

### 2.3.1. Alcance del proyecto

El proyecto alcance del proyecto cuenta con un tipo de cliente específico y cinco etapas principales que abarcan en su mayoría la magnitud total de la aplicación web construida. La aplicación pretende ser usada en computadoras y no en teléfonos móviles.

El cliente que se busca como usuario final es una agencia digital. Puesto que se busca tener cantidad de marcas y usuario ilimitados, favoreciendo a una agencia que tiene una gran rotación de marcas administradas debido a su gran volumen de cliente. Así como sus empleados que serían los usuarios finales quienes serían los encargados de gestionar dichas marcas.

En cuanto a las etapas:

La primera es la autenticación de usuarios. Tiene que ser una forma simple de iniciar sesión. Capaz de registrar una gran cantidad de usuarios sin problemas. Teniendo en cuenta la seguridad de las credenciales de los usuarios.



La segunda es la administración de marcas. En esta fase se implementa la creación y eliminación de marcas. Cualquier usuario de la aplicación puede crear una marca. La gestión de un equipo de trabajo será responsabilidad del administrador de la marca (el creador de la marca). Los miembros de la marca pueden posible agregar una cantidad ilimitada de cuentas de redes sociales.

Es importante aclarar que en algunos casos no siempre será posible agregar una cuenta de redes sociales. Esto es debido a los requerimientos actuales de las API Developer de los proveedores de redes sociales. Estos proveedores poseen ciertos métodos de autenticación y verificación de su uso que en algunos casos requieren tener un negocio verificado con la documentación legal en regla para su uso en proyectos de producción. Por lo cual, únicamente podrán agregar cuentas de redes sociales los usuarios que han sido agregados manualmente como usuarios tester en la configuración de la aplicación dentro del API Developer del proveedor.

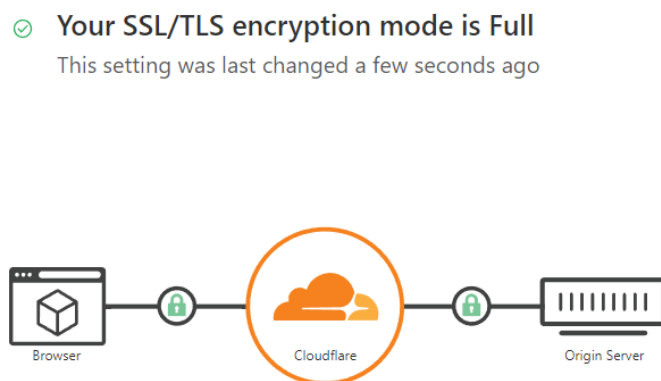
La tercera etapa es la creación de una biblioteca digital compartida donde en cada marca se tendrán archivos multimedia de video e imágenes para su posterior uso (opcional) en publicaciones. Estos archivos serán visualizados por los miembros de la marca y podrán etiquetarlos para facilitar su organización y búsqueda. Además de marcarlos como favoritos, eliminarlos. Estos archivos estarán almacenados en un bucket para mejorar su escalabilidad y gestión de espacio.

La cuarta etapa es la publicación en redes sociales. Estas publicaciones pueden ser en tiempo real o programadas en una fecha futura. Los usuarios pueden dentro de las marcas publicar en las redes sociales ya registradas de antemano. Principalmente dentro de lo posible se busca publicar únicamente videos y fotos de manera individual en primera instancia. Es decir, o una foto o un video en una o en múltiples redes sociales; cuando la red social acepte dicho tipo de archivo.

Para el alcance de este TFG se ha decidido ignorar la publicación masiva o altos volúmenes de publicaciones diarias. Existiendo la posibilidad de fallar dichas publicaciones al superar el límite permitido de la capa gratuita proporcionado por las API Developer. La razón es que no existe presupuesto

para el pago de estas mismas o bien hay un límite para el uso del API cuando no se posee una aplicación o negocio verificado. Así que los ejemplos de publicaciones en tiempo real o programados se mantendrán en cantidades bajas. Siendo consecuente la abstinencia de pruebas de estrés. Sin embargo, a pesar de ello se va a considerar una arquitectura escalable capaz de soportar grandes cargas de trabajo para su compatibilidad a futuro cuando se haga uso de las API de pago que ofrezcan mayor capacidad.

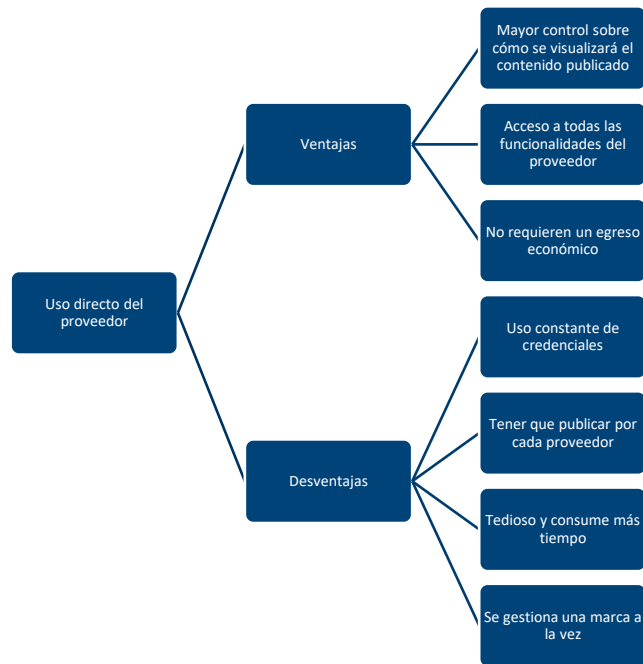
La quinta y última fase será la publicación de la aplicación en internet. Para esto se hará uso de un VPS donde se gestionarán los microservicios que componen la aplicación. Es esta etapa será necesario la posesión de un dominio para el sitio web, su configuración DNS y seguridad por medio de un certificado SSL/TLS como se muestra en la ilustración 2. Entre otras configuraciones de seguridad.



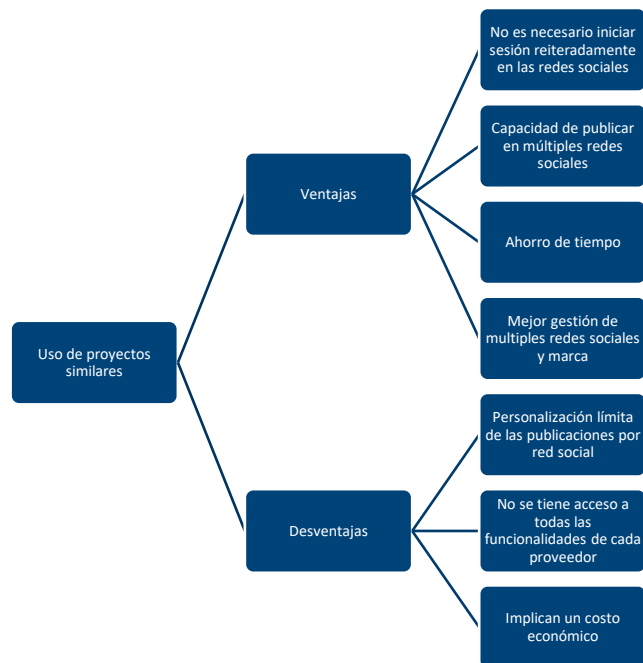
*Ilustración 2 Esquema SSL/TLS usando Cloudfare (Cloudfare, s.f.)*

### 2.3.2. Estudio de la situación actual

La situación actual se encuentra en dos posibles escenarios o la combinación de estos. La primera es la publicación de contenido directamente a través de los sitios web de los proveedores de redes sociales (ilustración 3). La segunda opción es el uso de algún proyecto similar como los que se mencionan en el apartado 2.2.



*Ilustración 3 Ventajas y desventajas del uso directo del proveedor de una red social*



*Ilustración 4 Ventajas y desventajas del uso de un proyecto similar*

Observando las ilustraciones 3 y 4 se puede observar que ambos tienen sus ventajas y desventajas. Siendo el uso directo de los proveedores una excelente opción para manejar una sola marca o una pequeña cantidad de redes sociales. Mientras que el uso de proyectos similares beneficioso cuando esta cantidad empieza a crecer.

### *2.3.3. Estudio y valoración de las alternativas de solución*

Existen 3 posibles soluciones para la situación planteada:

Primera solución: el uso directo de las páginas de cada proveedor de red social. Es solución no conlleva costos económicos asociados directos. Aunque sí los lleva indirectos en cuanto al tiempo empleado en cada red social, el cambio de cuentas, el manejo de credenciales y la falta de colaboración entre equipos. Caso favorable para trabajadores o creadores de contenido independientes.

Segunda solución: uso de proyectos similares. Esta solución conlleva costos económicos y es la más rápida de implementar si se tiene el dinero. Permite el trabajo en equipo y con muchas marcas. Sin embargo, los costos pueden escalar en sobremedida a medida que la cantidad de marcas administradas o miembros del equipo aumentan. Una solución ideal para un equipo mediano y que quiere actuar rápido.

Tercera opción: una aplicación a la medida. Esta solución requiere tiempo para realizarse y costos tanto de creación como de mantenimiento (dominio, bucket, servidores, etc.). Solución orientada a grandes equipos de trabajo que administran muchas marcas. Esta solución depende del costo beneficio. Siendo una mejor, igual o peor solución económicamente que la solución segunda en función de la cantidad de marcas y usuarios.

### *2.3.4. Selección de la solución*

Tomando en cuenta que el cliente objetivo es una agencia digital, el alto volumen de marcas y usuarios finales como se plantea en el alcance del proyecto. La mejor solución es hacer un software a la medida. Para ello se planea crear un MVP durante este proyecto el cual posteriormente seguirá en desarrollo al terminar la carrera.

### 3. METODOLOGÍAS USADAS

Para elegir la metodología a usar primero es necesario tener un conocimiento de las metodologías existentes y ver cual se adapta mejor al proyecto. Las metodologías se clasifican en dos grandes categorías: metodologías ágiles y tradicionales. En base al conocimiento de estos y sus tipos de metodologías por categoría se procederá a decidir la metodología usada (Ailin Orjuela Duarte, Las Metodologías de Desarrollo Ágil como una Oportunidad, 2008).

#### 3.1. Metodologías tradicionales

Las metodologías tradicionales son secuenciales con etapas bien definidas. Priorizan estas etapas bien definidas y son rígidas en cuanto a su implementación. Son metodologías que suelen usarse cuando hay requerimientos bien definidos que no suelen cambiar con el tiempo. Estas metodologías son como se mencionaba antes son rigurosas y su gran documentación que genera cada una de sus actividades (Ailin Orjuela Duarte, 2008)

#### 3.2. Metodologías ágiles

Las metodologías ágiles están pensadas para proyectos de iteraciones rápidas que requieren la colaboración constante del cliente para sacar el máximo valor posible. Se utilizan para soluciones a la medida donde se tiene en cuenta la flexibilidad o cambio de requisitos. Garantizando a pesar de ello un software de calidad (Ailin Orjuela Duarte, Las Metodologías de Desarrollo Ágil como una Oportunidad, 2008).

#### 3.3. Selección de metodología

Para decidir se ha decidido usar la tabla 4 de comparaciones de ambas metodologías proporcionada por el artículo científico de la Revista Avances en Informática de la Universidad Nacional de Colombia (Ailin Orjuela Duarte, 2008).

Tabla 4 Comparación metodologías ágiles vs tradicionales

Consideraciones	Metodologías Ágiles	Metodologías Tradicionales
<b>Fundamentos</b>	Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
<b>Adaptabilidad</b>	Preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
<b>Reglas de trabajo</b>	Reglas de trabajo impuestas internamente (por el equipo).	Reglas de trabajo impuestas externamente.
<b>Control del proceso</b>	Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
<b>Contratos</b>	Flexibilidad en los contratos debido a la respuesta a cambios.	Existe un contrato prefijado.
<b>Interacción con el cliente</b>	El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones en determinadas etapas del proceso.
<b>Tamaño del grupo</b>	Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos trabajando en diferentes tareas.
<b>Artefactos</b>	Pocos artefactos.	Más artefactos.
<b>Roles</b>	Pocos roles.	Más roles.
<b>Enfoque en arquitectura</b>	Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Como se muestra en la tabla 4, las metodologías ágiles son más convenientes para este proyecto por su flexibilidad a los cambios, pequeño equipo de trabajo, proceso menos controlado y que el cliente es parte del equipo.

### 3.4. Metodologías ágiles

Entre estas tenemos como las 3 más conocidas a Scrum, Kanban y Extreme Programming:

**Scrum**, posee roles bien definidos como scrum master, producto owner y developers. Se caracteriza por trabajar en sprints en los cuales cada sprint se pretende proveer valor al cliente por medio de avances del producto. Se centra en revisiones regulares y cambios. Además de una constante comunicación con el cliente y abierto a cambios y nuevas tareas.

**Kanban**, se tiene una lista de tareas de las cuales se va seleccionando las que se realizan por el equipo de trabajo. Las tareas son elegidas con el cliente y el objetivo es pasarlas todas del estado “Por hacer” a “Hecho”. Completando el proyecto al terminar las tareas. Dichas tareas pueden cambiar y son definidas en conjunto con el cliente. La visualización del trabajo siempre está presente y es fácil intuir lo que falta para terminar el proyecto o cuanto se ha avanzado (Kanban, 2022).

**Extreme Programming**, se caracteriza por rápidas entregas y despliegues. Suele implementarse el pair programming para trabajar en parejas y solucionar problemas y motivarse entre sí. Enfatiza la refactorización de código y las pruebas unitarias.

### 3.5. Scrum

Se ha seleccionado Scrum como la metodología a utilizar en este proyecto puesto que la intención es en cada sprint obtener un producto funcional con nuevas funcionalidades. Los objetivos del proyecto están definidos a nivel global, Sin embargo, el proyecto está abierto a cambios. Además, se cuenta con la colaboración de Marcela Peraz como cliente la cual estará durante el proceso de desarrollo del proyecto.

#### 3.5.1. Roles Scrum

Entre los roles se encuentran 3 roles prioritarios bien definidos:

- **Product Owner (cliente):** es el responsable de maximizar el valor del producto backlog y de cada sprint. Esta dentro del proyecto para ayudar a definir las tareas para el próximo sprint en cuanto a lo que consideré que generé más valor al producto. Es el encargado de decidir si acepta, rechaza o acepta con cambios los resultados de cada sprint.
- **Scrum master:** es un facilitador del equipo el cual se encarga de la planificación de cada sprint, su revisión y de su retrospectiva. Ayuda al equipo en lo que está a su alcance y es el encargado de convertir los requerimientos del cliente tras cada reunión en historias de usuario para el equipo de trabajo.

- Development team: es el equipo de desarrollo el cual cuenta con distintos roles como programadores, diseñadores, testers, devops, etc. Es un equipo multidisciplinario de poco integrantes los cuales se complementan entre sí para desarrollar el producto requerido por el cliente.

### 3.5.2. *Historias de usuario*

Son las tareas que se abordan en cada sprint. Estas vienen dadas por su descripción y tareas. Se puntúan con “puntos de historia”, el cual sirve para definir la dificultad de cada tarea. Además, también tienen un nivel de prioridad el cual es asignado según el valor que el cliente considere que esa historia posee.

La prioridad suele ser asignada por el scrum master. Por otro lado, los puntos de historia suelen ser asignados por alguien experimentado del equipo o bien por todos los miembros a través del planning poker.

Planning poker es cuando todos los miembros puntúan cada historia según ellos estimen necesario. Finalmente se discuten las opiniones entre quien asignó menos y más puntos y se llega a un censo final entre el equipo para definir la cantidad de puntos de historia de la tarea.



## 4. TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS EN EL PROYECTO

Este apartado surge del sprint 0 tras la primera reunión con el cliente. En este apartado se abarca las tecnologías usadas y las decisiones tomadas para elegir las en caso de requerir un análisis. Se abordan herramientas tanto de software como de hardware e infraestructura.

### 4.1. Control de versiones y gestión ágil

**Git** es una herramienta de control de versiones open source liviana y de alto rendimiento.

**Github** es una aplicación web de Microsoft compatible con git para el control de versiones de repositorios. Esta aplicación se usa para conectar el repositorio local de git a la nube y sincronizarlo remotamente el repositorio de la nube en Github.

**Github Desktop** es una aplicación proporcionada por Github que facilita el uso de Git-Github. Útil para cambiar eficientemente de ramas, hacer commits, revertir versiones, crear pull request o mover el stash entre ramas.

**Jira** es un software de seguimiento de proyectos e incidencias (Atlassian, s.f.). Este software es útil por sus integraciones con Github puesto que permite la creación de nuevas ramas de manera fácil en función de las tareas e historias de usuario del producto backlog. Asimismo, llevar el seguimiento del estado de estas tareas y su organización por sprints.

### 4.2. Full Stack Framework

Entre ellas podemos encontrar una gran variedad como XAMP, MERN, MEAN y muchas otras. Siendo todas muy útiles para el desarrollo según la necesidad o el entorno. Este proyecto no se tiene ninguna limitación de tecnologías en específico, por lo cual primero se tomará en cuenta consideraciones generales que permitan que el proyecto se desarrolle de la mejor manera. Eligiendo primero la base de datos, luego el backend y posteriormente el frontend. Para cada elección se tomará en cuenta las elecciones anteriores.

#### 4.2.1. *Base Datos*

SQL vs NOSQL, según los requisitos abordados del producto backlog en el apartado 5 es preferible usar una base de datos NOSQL. Esto es debido a la naturaleza del proyecto que requiere esquemas complejos de las entidades de la base de datos. Resolviendo este problema con una base de datos NOSQL la consultas, evitando tener un exceso de tablas y sus respectivos JOINS. Además, se requieren atributos que podrían ser nulo, de longitud muy corta o bien muy larga. Siendo esto un inconveniente para las bases de datos SQL por su almacenamiento en bloques. Por último, las bases de datos NOSQL son bases de datos de escalabilidad horizontal y se almacenan por clave-valor, ofreciendo un buen rendimiento de búsqueda (TestGorilla, 2024).

**MongoDB**, base de datos NOSQL de alto rendimiento administrada en la nube por MongoDB Atlas. Esta es la base de datos usada por los stacks MERN y MEAN, siendo el motivo principal por el cual ha sido escogida por su gran apoyo de la comunidad, documentación y estabilidad en el mercado.

#### 4.2.2. *Backend*

En el punto anterior se decidió la base de datos, ahora es importante elegir un framework backend para utilizar MongoDB en el proyecto. Idealmente se quiere un framework o entorno que sea rápido en E/S puesto que el proyecto requiere un gran nivel de entrada y salida de ficheros tanto para la biblioteca digital como para subir los ficheros a los proveedores de redes sociales. Además, se requiere que el framework posea tipado para mejorar la experiencia de desarrollo y evitar errores a priori.

Se han considerado los entornos de desarrollo de **.NET** y **Node.js** para el backend. Como previamente se mencionaba los procesos de E/S son prioridad, así que se han ejecutado las siguientes métricas propias de un backend en ambas tecnologías como prueba de concepto. Para esta prueba se ha hecho un endpoint sencillo que guarda 2 archivos (una imagen de 3MB y un video de 100MB) y se ejecuta 10 veces (ilustración 5).

Seleccionar archivo	photo3M.jpg
Seleccionar archivo	video100M.mp4
10	Express <span>▼</span> <span>Submit</span>

- Express ACTIVO.
- .NET ACTIVO.
- dotnet - Iniciando proceso a las 3:42:41 con 10 iteraciones.
- dotnet - Terminado a las 3:42:47 y se ha tardado 6.26 segundos
- dotnet - Éxitos: 10 de 10
- express - Iniciando proceso a las 3:42:51 con 10 iteraciones.
- express - Terminado a las 3:42:53 y se ha tardado 1.869 segundos
- express - Éxitos: 10 de 10

*Ilustración 5 Benchmark E/S - Node.js vs .NET*

Obteniendo resultados significativamente mejores con **Node.js** para el manejo de E/S, más de 3 veces más rápido (ilustración 5). Siendo este elegido para el backend usando **TypeScript** para mejor control del tipo de datos.

#### 4.2.3. Frontend

Previamente se ha elegido usar MongoDB, Nodejs y TypeScript. Tomando en cuenta dichas tecnologías y lenguajes el frontend podría estar hecho en React, Angular o Vuejs.

*Tabla 5 Comparativa React, Angular y Vuejs (Navarro, 2023)*

Consideraciones	React	Angular	Vuejs
Curva aprendizaje	Media	Alta	Baja
Rendimiento	Alto	Alto	Alto
Robustez	Altamente adaptable, pero puede requerir más configuraciones y decisiones por parte del desarrollador	Arquitectura robusta y definida	Altamente adaptable, no necesariamente el mejor para aplicaciones grandes y complejas
Data Binding	One way	Two way	Two way
Componentes	Sí	Sí	Sí

Según los datos de la tabla 5, se ha decidido descartar Vuejs por la consideración de robustez ya que la aplicación a crear se considera de escala mediana-grande y conlleva múltiples estados simultáneamente.

Angular parece ser el indicado debido a su arquitectura. Sin embargo, su curva de aprendizaje y Data Binding podrían ser perjudiciales para el desarrollo debido al atraso en etapas tempranas del proyecto y posibles brechas de seguridad debido al two way data binding (tabla 5).

React parece también ser una buena opción por tener menor curva de aprendizaje y usar one way data binding (permitiendo al desarrollador el control total de los estados de la aplicación). Sin embargo, en robustez sigue siendo superado por Angular (tabla 5).

**Next.js** es un framework escalable de desarrollo fullstack que se puede usar tanto con TypeScript como JavaScript. Utiliza componentes de React, es decir ofrece one way data binding. Estos componentes se pueden ejecutar en el cliente o en el servidor. Dichos componentes del servidor implementan SSR (Server Side Rendering) como Angular. El SSR ofrece páginas compatibilidad con SEO y gran escalabilidad y velocidad de respuesta por el cacheo de las páginas (Next.js, s.f.).

Next.js es un framework robusto, reduciendo la cantidad de decisiones de librerías extras a usar. Además, Next.js implementa “Server Actions” de manera estable desde su versión 14. Estas acciones son funciones que se ejecuta en el servidor (Next.js, s.f.). Ofreciendo una excelente experiencia de desarrollo puesto que al usar TypeScript si se modifica un server action, el compilador detecta automáticamente los cambios en la interfaz usada en el frontend. Esto ofrece el beneficio de reducir significativamente las incongruencias entre el backend y el frontend al ser compilados independientemente.

Por lo tanto, se ha decidido usar Next.js como framework de desarrollo porque suple las carencias tanto de React como de Angular y a la vez cumple los requisitos previamente definidos por la base de datos y el backend.

#### **4.3. Herramientas de desarrollo**

**Visual Studio Code** es un editor de código open source de Microsoft el cual ofrece soporte para múltiples lenguajes de programación. También soporta

múltiples integraciones de extensiones que serán útiles para el desarrollo. También

**MobaXTerm** es una terminal mejorada para Windows la cual ofrece una excelente experiencia de uso para protocolos de SSH, FTP, FTPS, herramientas de red y mucho más (Mobatek, s.f.). Esta será usada para configurar el VPS.

#### 4.4. Infraestructura y servicios

**AWS S3** es un servicio de AWS (Amazon Web Services) para el almacenamiento de ficheros de manera escalable.

**MongoDB Atlas** es un servicio de la nube que ofrece bases de datos de MongoDB para simplificar el proceso de desarrollo.

**Hostinger** es una empresa que provee servicio de la nube como VPS, dominios, certificados SSL o hosting para páginas web.

**Cloudfare** es una compañía que se ofrece servicios de DNS, hosting, SSL, protección de ataques DDoS (Distributed Denial of Service), CDN (Content Delivery Network), entre otros.

**NVM (Node Version Manager)** es una herramienta para manejar múltiples versiones de Node.js.

**Nginx** es servidor web de código abierto que se puede utilizar como proxy inverso o caché HTTP (nginx, s.f.). Este será usado en el servidor en conjunto con la configuración del DNS para redirigir el tráfico a la aplicación en el puerto correspondiente.

**PM2** es un gestor de procesos de producción para aplicaciones Node.js con un balanceador de carga incorporado. Permite mantener las aplicaciones siempre activas, reiniciarlas en caso de fallo y monitorizar sus recursos y logs (Keymetrics Inc, s.f.).

**Github Actions** es un servicio de CI/CD que Github para configurar en cada repositorio.

#### 4.5. Hardware

Para el desarrollo del proyecto, se ha utilizado un ordenador portátil **ASUS TUF GAMING A15** con las siguientes características:

- Procesador AMD Ryzen 7 6800H with Radeon Graphics
- 16GB de RAM DDR5.
- Tarjeta gráfica NVIDIA GeForce RTX 3070 Laptop GPU y AMD Radeon (TM).
- 1TB de almacenamiento SSD NVMe M.2
- Sistema Operativo Windows 11

## 5. ESTIMACIÓN DE RECURSOS Y PLANIFICACIÓN

En este apartado se pretende explicar cómo se ha hecho la planificación y estimación de recursos del proyecto usando Scrum. Se mencionan los roles del proyecto, las historias de usuario y tareas a desarrollar junto a su priorización en el proyecto, una planificación temporal del proyecto y su estimación económica.

### 5.1. Sprint 0

#### 5.1.1. Equipo Scrum

A continuación, en la tabla 6, se muestra la asignación de los roles del proyecto y sus respectivos encargados. Las funciones de cada rol se describen en el apartado 3.

Tabla 6 Roles Scrum

Rol	Encargado
Product Owner	Marcela Peraz
Scrum Master	Ronald Ernesto Tejada Ríos
Developer	Ronald Ernesto Tejada Ríos

#### 5.1.2. Product Backlog

Tras la primera reunión con el Product Owner, se han definido las siguientes historias de usuario y sus respectivas tareas (tabla 7). Para ello se ha utilizado la técnica de priorización MoSCoW que consiste en clasificar las tareas según su nivel de importancia (Kuhn, 2009) como se muestra a continuación:

- **Must:** es una tarea principal del proyecto y debe tenerla.
- **Should:** es una tarea que brinda valor e idealmente debe de tenerla, aunque podría posponerse.
- **Could:** es una tarea que podría tenerse, no es prioridad y si no
- **Won't:** este requisito no es necesario actualmente, pero podría agregarse en el futuro.

Tabla 7 Product Backlog Original

ID	Historia de usuario (épicas)	ID tarea	Tarea	Prioridad	Puntos historia (PH)
H1	Análisis inicial	H1.1	Diseño de la solución	Must	3
		H1.2	Selección de tecnologías	Must	5
H2	Configuración de entorno	H2.1	Creación de la estructura repositorio	Must	1
		H2.2	Configuración de la base de datos	Must	2
		H2.3	Diagrama entidad-relación	Should	1
		H2.4	Librería de UI	Should	1
H3	Autenticación de usuarios	H3.1	Selección de método de autenticación	Should	2
		H3.2	Iniciar sesión	Must	2
		H3.3	Cerrar sesión	Must	1
		H3.4	Verificar sesión en las páginas dentro de la aplicación	Must	1
H4	Administración de marcas	H4.1	Crear una marca	Must	3
		H4.2	Visualizar las marcas	Must	2
		H4.3	Seleccionar una marca global	Must	1
		H4.4	Modificar el nombre de una marca	Could	2
		H4.5	Agregar miembros a la marca (admin)	Must	3
		H4.6	Ver miembros del equipo	Must	2
		H4.7	Eliminar miembros a la marca (admin)	Should	1
H5	Administración de redes sociales de una marca	H5.1	Agregar cuenta de YouTube	Must	7
		H5.2	Agregar cuenta de Facebook	Must	7
		H5.3	Agregar cuenta de TikTok	Must	7
		H5.4	Agregar cuenta de Instagram	Should	7
		H5.5	Agregar cuenta de Twitter (X)	Should	7
		H5.6	Eliminar cuenta de red social (admin)	Could	2
		H5.7	Actualizar tokens de acceso automáticamente	Must	7
H6	Biblioteca digital compartida	H6.1	Configurar bucket	Must	5
		H6.2	Subir archivos de manera segura	Should	5
		H6.3	Ver imágenes subidas	Must	1
		H6.4	Ver videos subidos	Must	2
		H6.5	Ordenar por tipo de archivo	Must	1
		H6.6	Buscar archivos por nombre	Should	2
		H6.7	Visualizar quién subió los archivos	Should	1
		H6.8	Clasificar los archivos por etiquetas	Could	5
		H6.9	Marcar archivos como favoritos	Could	3
		H6.10	Eliminar archivos	Should	2
		H6.11	Visualizar los archivos que no han sido usado en publicaciones	Must	2
		H6.12	Visualizar los archivos que ya han sido usado en publicaciones	Must	2



		H6.13	Visualizar los archivos que han sido programados para publicar	Should	2
		H6.14	Descargar archivos subidos	Must	1
H7	Publicación en redes sociales	H7.1	Publicar un video en Facebook	Must	7
		H7.2	Publicar un video en TikTok	Must	7
		H7.3	Publicar un video en Instagram	Should	7
		H7.4	Publicar un video en YouTube	Must	7
		H7.5	Publicar un video en Twitter (X)	Should	7
		H7.6	Publicar una imagen en Facebook	Should	7
		H7.7	Publicar una imagen en Instagram	Should	7
		H7.8	Publicar una imagen en Twitter (X)	Should	7
		H7.9	Publicar una imagen en múltiples redes sociales	Must	7
		H7.10	Publicar un video en múltiples redes sociales	Must	7
		H7.11	Publicar una imagen o video en una fecha futura	Must	7
		H7.12	Visualizar las publicaciones por marca en forma de calendario	Must	7
		H7.13	Calendario público no editable para enviar a clientes	Should	7
		H7.14	Publicar múltiples fotos en Facebook	Could	7
		H7.15	Publicar múltiples fotos en Instagram	Could	7
		H7.16	Publicar múltiples fotos en Twitter (X)	Could	7
		H7.17	Publicar múltiples videos en Facebook	Could	7
		H7.18	Publicar múltiples videos en Instagram	Could	7
		H7.19	Publicar múltiples videos en Twitter	Could	7
H8	Despliegue	H8.1	Comprar dominio	Must	2
		H8.2	Comprar VPS	Must	2
		H8.3	Configurar seguridad básica	Should	3
		H8.4	Instalar software en el VPS	Must	2
		H8.5	Configurar repositorio	Must	2
		H8.6	Crear los archivos .env para producción	Must	1
		H8.7	Configurar sitios web y DNS	Must	3
		H8.8	Iniciar las aplicaciones utilizando un administrador de procesos	Must	2
		H8.9	Configurar monitorización y rendimiento	Should	3
		H8.10	Configurar CI/CD	Should	5

Obteniendo 269 puntos de historia de todas las tareas a realizar del producto backlog.

La ponderación usada para calificar los puntos de historia se puede ver en la tabla 8. Se ha decidido que cada tarea realizada toma en cuenta las acciones tanto del backend como el frontend y demás acciones relacionadas para cumplir dicha tarea.

*Tabla 8 Equivalencias de tareas y puntos de historia*

Puntos historia	Tarea de referencia
1	GET-DELETE de una entidad
2	POST-PUT de una entidad GET-DELETE de entidades relacionadas
3	POST-PUT entidades relacionadas
5	Integración con software de terceros (GET-DELETE)
7	Integración con software de terceros (POST-PUT)

### *5.1.3. Estimación temporal del proyecto*

#### *5.1.3.1 Estimación temporal global*

Se ha decidido no usar diagramas de Gantt en este proyecto puesto que al usarlos en una metodología ágil estos tendrían que actualizarse casi constantemente. Siendo una carga innecesaria que no aporte mucho valor (Bara, 2020). Más bien se ha usado el concepto de “épicas” siendo éstas las historias de usuario descritas en el producto backlog que han sido descompuestas en tareas (Atlassian, s.f.).

Para realizar una estimación temporal se ha decidido utilizar un cálculo basado únicamente en los puntos de historia de usuario. Tomando como referencia personal que cada PH es equivalente 2 o 3 horas de trabajo. Decidiendo usarse las 2.5 horas por PH. Además, se considera que la carga de trabajo semanal será de 25 horas semanales y que cada sprint tendrá una duración de 2 semanas. Es decir, cada sprint se planea abordar en promedio 20 PH.

$$\text{Tiempo (horas)} = \text{PH totales} * \text{horas por punto de historia}$$

$$\text{Tiempo (horas)} = 269 \text{ puntos} * 2.5 \text{ horas/punto}$$

$$\text{Tiempo (horas)} = 672.5 \text{ horas}$$

Siendo estas un total 672.5 horas, equivalente a aproximadamente 26.9 semanas, es decir 13-14 sprints. Sin embargo, cumplir todas las expectativas del cliente es inviable temporalmente puesto que el proyecto se inició el jueves 1 de febrero de 2024, fecha de la reunión inicial con el cliente, hasta el 20 de mayo del mismo año (fecha de entrega del TFG) hay 16 semanas. Por lo tanto, se hará una estimación temporal basada en MoSCoW para obtener el mayor valor posible en el tiempo proporcionado.

#### 5.1.3.2 Estimación temporal aplicando MoSCoW

Para este apartado se tendrá en cuenta la cantidad de PH por cada tipo de prioridad (tabla 9). Observando de esta manera, la cantidad de PH de las tareas Must es menos de la mitad de los PH totales.

Tabla 9 Puntos de Historia aplicando MoSCoW

Prioridad	Puntos historia
Must	131
Should	84
Could	54
Won't	0
Total	269

En el apartado 5.1.3.1 se encontró la limitante temporal de 16 semanas para realizar el proyecto. En contraparte, al observar la tabla 10 se puede inferir que es posible obtener un MVP (Minimum Viable Product) al realizar únicamente las tareas Must y algunas tareas Should.

Tabla 10 Planificación temporal MoSCoW

Prioridad	Horas	Semanas	Meses
Must	327,5	13,1	3,1
Must+Should	537,5	21,5	5,1
Must+Should+Could	672,5	26,9	6,4

#### 5.1.3.3 Estimación temporal optimista y pesimista aplicando MoSCoW

Tras validar la viabilidad temporal del proyecto es importante en cuenta que los PH asignados a las tareas son estimaciones aproximadas en base a una tabla de ponderaciones (tabla 11) y a la experiencia propia. Por lo cual, no están libres de errores. A continuación, se hará una segunda y tercera planificación

MoSCoW tomando en cuenta un tiempo de realización de PH optimista (2 horas/PH) y uno pesimista (3 horas/PH).

- Planificación MoSCoW optimista

Tabla 11 Planificación MoSCoW optimista

Prioridad	Horas	Semanas	Meses
Must	262	10,5	2,5
Must+Should	430	17,2	4,1
Must+Should+Could	538	21,5	5,1

- Planificación MoSCoW pesimista

Tabla 12 Planificación MoSCoW pesimista

Prioridad	Horas	Semanas	Meses
Must	393	15,7	3,7
Must+Should	645	25,8	6,1
Must+Should+Could	807	32,3	7,7

Logrando en el peor de los casos, pesimista (tabla 12), conseguir las tareas de tipo Must para el MVP del Product Owner. Dejando las demás tareas para posteriores sprints como vías futuras. Mientras que, en el mejor caso, optimistas (tabla 11), se conseguiría aproximadamente todas las tareas de Must y Should

#### 5.1.4. Estimación económica del proyecto

Para este cálculo se tomará en cuenta el salario anual bruto de 21.100€ para un programador Full Stack Junior (Jobtec, 2024), los gastos indirectos del proyecto y los costos asociados al pago de servicios necesarios para poner en marcha la aplicación final.

- Costos fijos:

El primer costo es el relacionado al salario del programador. Como se ha mencionado en el apartado de estimación temporal global, se disponen de 16 semanas para realizar el proyecto. Sin embargo, por problemas personales de una lesión del hombro y complicaciones de diabetes el proyecto ha tenido 3

semanas de descanso total. Por lo tanto, el salario devengado se calculará en función a las 13 semanas.

El cual se calcula a continuación:

$$\text{Salario} = \frac{13 \text{ semanas}}{52 \text{ semanas/año}} * \left( \frac{25 \frac{\text{horas}}{\text{semana}}}{40 \frac{\text{horas}}{\text{semana}}} \right) * 21.100 \text{ €/año}$$

$$\text{Salario} = 5,275 \text{ €}$$

Posteriormente, los demás costos fijos implicados se muestran en la tabla 13.

Tabla 13 Costos indirectos del proyecto

Costos Fijos	Calculo	Costo
Agua, luz, internet	100 €/mes * 4 meses	400,00 €
Depreciación equipo hardware	1450€ * 4 meses / 24 meses vida útil	241,67 €
Curso formación para el cliente	10 horas	250,00 €
Soporte anual (opcional)	Dudas y cambios pequeños	1.000,00 €
Salario	4 meses devengados	5.275,00 €
Total		7.166,67 €

- Costos variables
  - Costo anual de servicios contratados:

Como se puede ver en la ilustración 6, el primer gasto es asociado al VPS y el segundo al Dominio (<https://publishwhere.com>).

ID de la factura	Creado en	Pagado el	Precio total	
H_10040992	2024-05-05	2024-05-05	191,37 €	Reembolso ▼
H_9537096	2024-04-19	2024-04-19	12,30 €	▼

Ilustración 6 Costos de Hostinger

El tercer servicio recurrente sería el almacenamiento de archivos. Para ello se ha tomado en cuenta la calculadora de precios de AWS para obtener el siguiente costo mensual en base a una aproximación de los recursos mensuales

que se espera operar en la aplicación (ilustración 7). Obteniendo un costo anual estimado de 111.52 USD = 104.16 EUR.

Export date: 9/2/2024

Language: English

Estimate URL: <https://calculator.aws/#/estimate?id=3a1eee672c761bda14682be997a59741148add57>

Estimate summary

Upfront cost	Monthly cost	Total 12 months cost
0.28 USD	9.27 USD	111.52 USD
		Includes upfront cost

Detailed Estimate

Name	Group	Region	Upfront cost	Monthly cost
Amazon Simple Storage Service (S3)	No group applied	EU (Spain)	0.28 USD	9.27 USD

Status: -

Description: PublishWhere TFG

Config summary: S3 Standard storage (390 GB per month), S3 Standard Average Object Size (100 MB), PUT, COPY, POST, LIST requests to S3 Standard (4000), GET, SELECT, and all other requests from S3 Standard (12000), Data returned by S3 Select (390 GB per month) DT Inbound: Internet (390 GB per month), DT Outbound: Not selected (1 TB per month)

Ilustración 7 Costo anual del bucket usando AWS S3

Desglosándose los gastos anuales recurrentes por servicios como se muestran en la tabla 14. Estos gastos el cliente tiene que pagarlos anualmente para poder ejecutar el proyecto en producción y accederlo. Dichos costos pueden variar en cuanto al consumo requerido del cliente, estos costos solo sirven de referencia.

Tabla 14 Costos variables recurrentes anuales del proyecto

Servicios	Calculo	Costo anual
Dominio	12,30€ primer año	12,30 €
VPS	191,37 € por 2 años	95,69 €
Bucket S3	AWS Calculator	104,16 €
Total		212,15 €

## 6. DESARROLLO DEL PROYECTO

### 6.1. Sprint 0

#### 6.1.1. Planificación del sprint

Duración: 2 semanas

Este sprint se da luego de la reunión inicial con el cliente. Las tareas del sprint se muestran en la tabla 15. Los PH totales son 8.

Tabla 15 Tareas del sprint 0

ID tarea	Tarea	Prioridad	Puntos historia (PH)
H1.1	Diseño de la solución	Must	3
H1.2	Selección de tecnologías	Must	5

#### 6.1.2. Desarrollo del sprint

**H1.1** se ha decidido aplicar una solución monorepo de múltiples proyectos. Distribuyendo así la responsabilidad del proyecto en microservicios (ilustración 8).

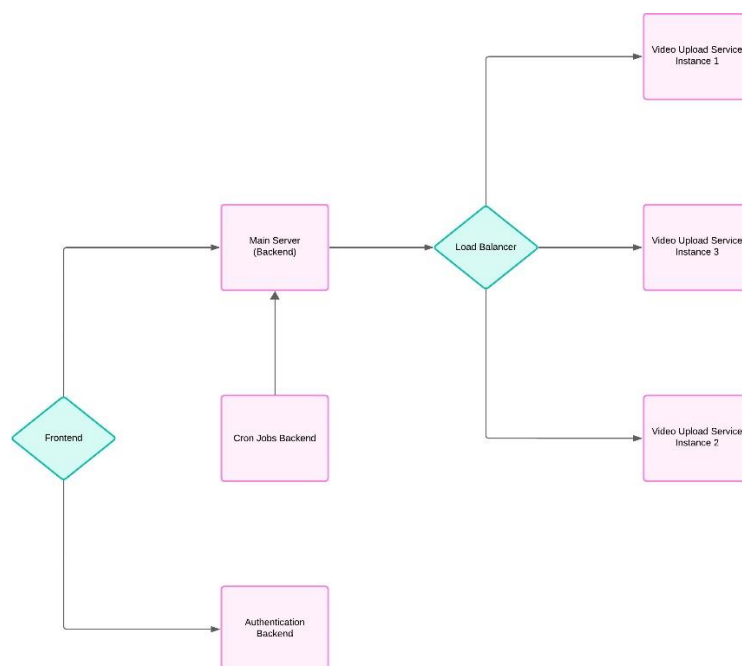


Ilustración 8 Diagrama de microservicios

- Video Upload Service es el encargado de publicar videos en redes sociales. Obtiene los datos respectivos de la base de datos y los ficheros del bucket para comenzar el proceso de publicación. Este proceso puede requerir muchos recursos en función del tamaño de los ficheros o la cantidad de publicaciones simultaneas.
- Load Balancer es el encargado de distribuir las publicaciones entre las 3 instancias de Video Upload Service.
- Main Server es el backend principal del proyecto. Aquí se gestiona lógica de autenticación de usuario, peticiones al servidor por el frontend y sus respectivas respuestas, administrar la lógica de seguridad de las sesiones y llamar al Video Upload Service.
- Cron Job Backend es el encargado de refrescar los tokens de seguridad de Oauth obtenidos de las redes sociales, así como de verificar si hay publicaciones programadas pendientes y hacer la solicitud al backend principal para publicar videos.
- Autenticación Backend es el encargado de la lógica de Oauth y sesiones con proveedores de redes sociales. La intención es separar la lógica de la sesión de usuario y la lógica de las sesiones con tokens bearer de los proveedores que son información delicada y confidencial.
- Frontend es la aplicación web final que el usuario tiene en su navegador que se comunica únicamente con el Main Server.

**H1.2** ha sido desarrollado a lo largo del apartado 4.

### 6.1.3. *Revisión del sprint*

Para este apartado todas las tareas fueron completadas con éxito (tabla 16).

Tabla 16 Tareas del sprint 0 - revisión

ID tarea	Tarea	Prioridad	Puntos historia (PH)	Puntos reales	Estado
H1.1	Diseño de la solución	Must	3	2	Completado



H1.2	Selección de tecnologías	Must	5	7	Completado
------	--------------------------	------	---	---	------------

#### 6.1.4. Retrospectiva del sprint

La elección de las tecnologías tomó más tiempo del previsto. El equipo de desarrollo no tiene experiencia con el framework seleccionado así que el tiempo libre de este sprint se ha dedicado a aprender la nueva tecnología.

### 6.2. Sprint 1

En este sprint no hubo reunión con el cliente puesto que se especificó que había que hacer un análisis inicial del proyecto y luego el siguiente sprint (sprint 1) se iniciaría con el desarrollo.

#### 6.2.1. Planificación del sprint 1

Duración: 2 semanas

Para este sprint se tienen 17 PH como se muestra en la tabla 17.

Tabla 17 Tareas del sprint 1

ID tarea	Tarea	Prioridad	Puntos historia (PH)
H2.1	Creación de la estructura repositorio	Must	1
H2.2	Configuración de la base de datos	Must	2
H2.3	Diagrama entidad-relación	Should	1
H2.4	Librería de UI	Should	1
H3.1	Selección de método de autenticación	Should	2
H3.2	Iniciar sesión	Must	2
H3.3	Cerrar sesión	Must	1
H3.4	Verificar sesión en las páginas dentro de la aplicación	Must	1
H4.1	Crear una marca	Must	3
H4.2	Visualizar las marcas	Must	2
H4.3	Seleccionar una marca global	Must	1

### 6.2.2. Desarrollo del sprint 1

H2.1 para esta tarea se ha creado una organización de Github y un repositorio el cual ha sido enlazado a Jira para su seguimiento de control de versiones (ilustración 9). En el repositorio se creó 4 proyectos uno para Nextjs (Main Server - Frontend) y los demás con Express para Cron Jobs Service, autenticación service y el Video Upload Service mencionados en H1.1.

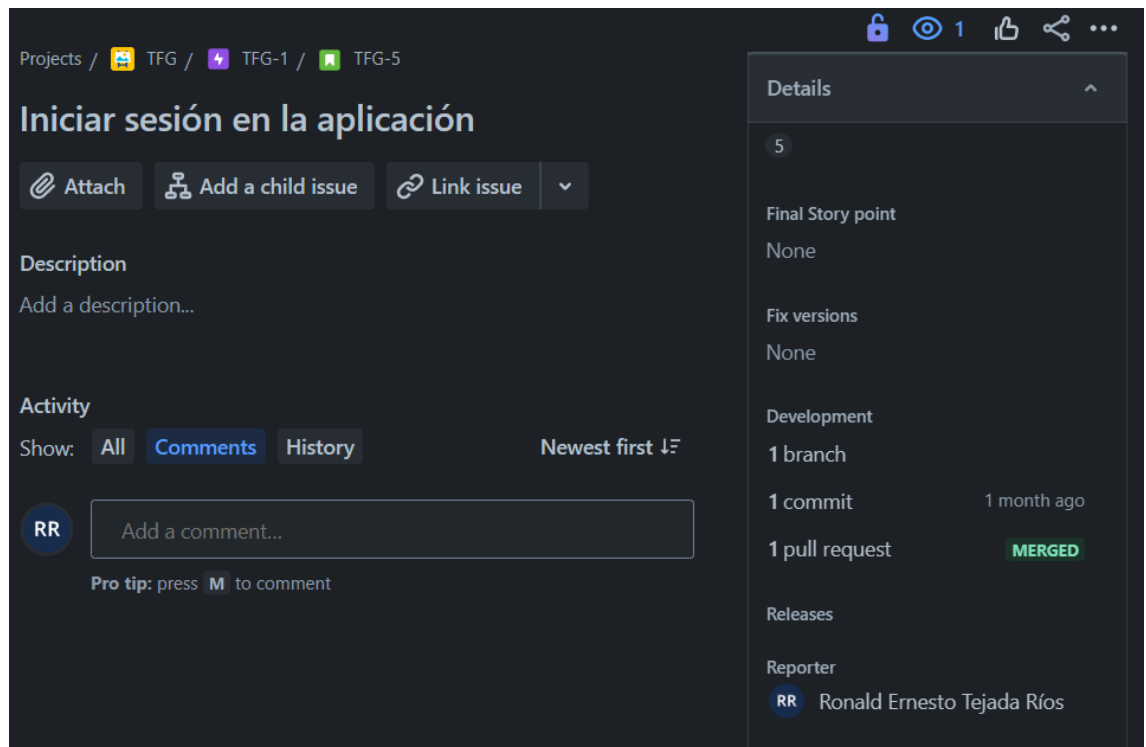


Ilustración 9 Tarea de jira

H2.2 se creó la base de datos de MongoDB en MongoDB Atlas, se ha configurado y testeado la conexión a la base de datos en el backend principal.

H2.3 el diagrama de entidad relación se muestra en la ilustración 10. Por el momento este únicamente pretende tener la relación de las entidades para un análisis general sin contemplar sus atributos.

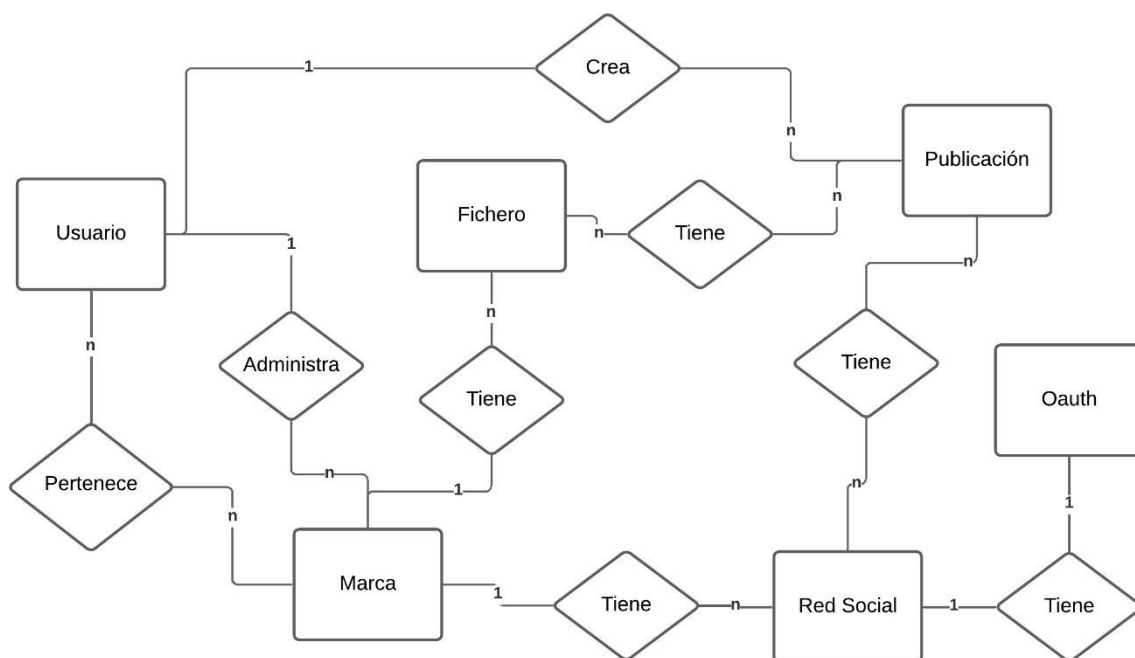
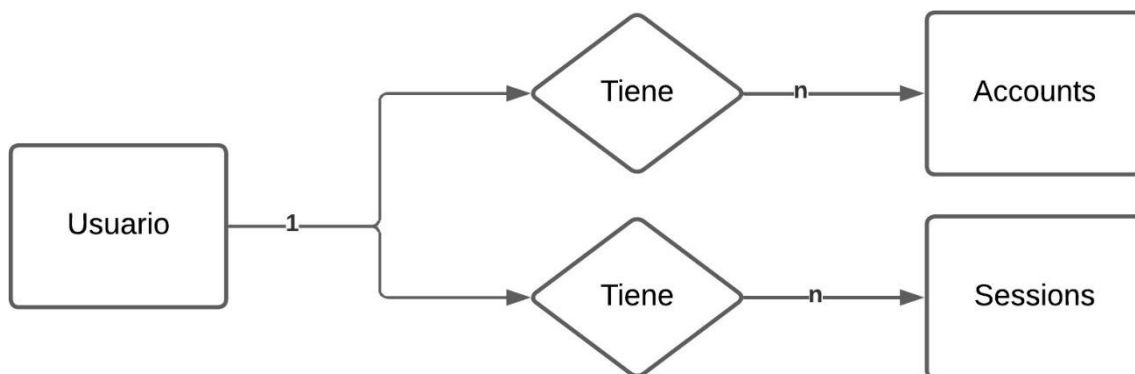


Ilustración 10 Diagrama entidad relación

H2.4 para esto se ha seleccionado la librería de Shadcn/ui por su excelente documentación, implementación con Next.js y su facilidad de editar todos los componentes de manera global puesto que crea instancias de los componentes de la librería en los componentes reutilizables del proyecto.

H3.1 para esta tarea se han considerado 3 soluciones posibles para autenticar usuarios en la aplicación. Usar contraseña y correo en conjunto a un JWT o usar Next-Auth. La primera opción se descartó porque la idea principal es iniciar sesión de manera rápida y sencilla. En consecuencia, Next-Auth fue la librería utilizada porque implementa fácilmente en cuanto a código y experiencia de usuario el inicio de sesión por medio de un proveedor OAuth.



*Ilustración 11 Entidades Next-Auth*

Tal y como se observa en la ilustración 11, Next-Auth proporciona dos nuevas entidades a la base de datos. Sessions donde se tienen las múltiples sesiones activas del usuario (sustituyendo el JWT y funcionando como mecanismo de seguridad) y Accounts que es donde se guarda la cuenta de Google usada para iniciar sesión dentro de la aplicación.

H3.2 al implementar Next-Auth fue necesario configurar el client\_ID y client\_secret por medio de la consola de Google Developers al crear una aplicación y crear las credenciales de Oauth mencionadas. Teniendo esto configurado en el archivo de .env y el driver del proveedor de la base de datos (mongoose). Se pudo configurar exitosamente el botón de iniciar sesión.

H3.3 el proceso de cerrar sesión fue extremadamente fácil puesto que es solo llamar una función proporcionada por Next-Auth. Lo importante a considerar acá es borrar los posibles datos guardados en el local storage/cookies para evitar inconvenientes en la próxima sesión.

H3.4 se implementó en las vistas pertinentes que si no existía una sesión válida proporcionada por Next-Auth, el usuario debía de ser redirigido a la página de home.

H4.1 para crear una marca es necesario que un usuario tenga una sesión válida. Este al crear una marca solo necesita ingresar el nombre de la marca a crear y esta es creada. El usuario administrador de la marca es el usuario que la creo y directamente este se guarda como miembro del equipo de la marca.

H4.2 y H4.3 para resolverlos se ha decidido mostrar las marcas en el header y al seleccionar una marca está se guarda en el local storage por medio de un hook de react.

### *6.2.3. Revisión del sprint 1*

Todas las tareas del sprint 1 fueron completadas con éxito. Obteniendo un total de 2.5 PH más de lo que se había planificado (tabla 18).

Tabla 18 Tareas del sprint 1 - revisión

ID tarea	Tarea	Prioridad	Puntos historia (PH)	Puntos reales	Estado
H2.1	Creación de la estructura repositorio	Must	1	3	Completado
H2.2	Configuración de la base de datos	Must	2	1	Completado
H2.3	Diagrama entidad-relación	Should	1	2	Completado
H2.4	Librería de UI	Should	1	1	Completado
H3.1	Selección de método de autenticación	Should	2	3	Completado
H3.2	Iniciar sesión	Must	2	3	Completado
H3.3	Cerrar sesión	Must	1	0,5	Completado
H3.4	Verificar sesión en las páginas dentro de la aplicación	Must	1	1	Completado
H4.1	Crear una marca	Must	3	2	Completado
H4.2	Visualizar las marcas	Must	2	2	Completado
H4.3	Seleccionar una marca global	Must	1	1	Completado

Al finalizar el sprint hubo una corta reunión con el producto owner para mostrarle el avance. Este quedó satisfecho, aunque a la vez esperaba ver un avance mayor. Se le explicó las decisiones tomadas y el desarrollo del sprint 0, quedando satisfecho ya que las decisiones previas influirían positivamente en el proyecto. Finalmente, se comentó lo que se esperaba para el siguiente sprint.

#### 6.2.4. Retrospectiva del sprint 1

El sprint se desarrolló sin problemas. El único inconveniente fue la curva de aprendizaje con el framework y el uso de librerías de terceros. Sin embargo, esto estaba dentro de las previsiones.

### 6.3. Sprint 2

#### 6.3.1. Planificación del sprint 2

Duración: 2 semanas

Para este sprint se han planificado 15 PH (tabla 19). Siendo menos debido al período de exámenes parciales. Se espera dedicar más PH para el próximo sprint.

Tabla 19 Tareas del sprint 2

ID tarea	Tarea	Prioridad	Puntos historia (PH)
H4.4	Modificar el nombre de una marca	Could	2
H4.5	Agregar miembros a la marca (admin)	Must	3
H4.6	Ver miembros del equipo	Must	2
H4.7	Eliminar miembros a la marca (admin)	Should	1
H5.1	Agregar cuenta de YouTube	Must	7

#### 6.3.2. Desarrollo del sprint 2

H4.4 esta tarea implica seleccionar una marca y modificar su nombre. Para ello se creó una nueva vista que pretende ser para la edición de esta tarea y las demás tareas planificadas del sprint.

Dicha tarea requirió crea un nuevo visualizador de tareas, un contexto global para manejar el estado y compartirlo entre los distintos componentes. Finalmente, al implementar todo esto, fue posible cambiar el nombre de una marca y actualizar las vistas en tiempo real tanto como los cambios en la base de datos.

H4.5 funcionalidad solo para el admin de la aplicación. Para agregar un miembro se creó una alerta con todos los usuarios de la aplicación dividiéndolos entre los que no pertenecen y los que ya pertenecen al equipo. El usuario tiene que seleccionar los miembros del equipo y guardar los cambios (ilustración 12).

**Agregar miembros al equipo: Temp** ×

Selecciona a las personas que quieras agregar

Buscar a:

---

Miembros disponibles: María Isabel Cornejo Rojas - micornejo@alu.ucam.edu

---

Miembros seleccionados: • Ronald Tejada (Ris) - rris402@gmail.com

---

Asegurate de guardar tus cambios **Guardar**

*Ilustración 12 Agregar miembros al equipo*

H4.6 los miembros agregados se muestran en la vista mencionada en función de la marca seleccionada.

H4.7 para eliminar es necesario validar que el usuario sea el admin de la marca. Para este alcance, se ha decidido que el admin no cambia nunca y no puede eliminarse a sí mismo.

H5.1 para agregar una red social de YouTube es necesario configurar el YouTube Data API V3 desde la consola de desarrollo de Google. Para volver a crear nuevamente credenciales de Oauth (client ID y client secret). En la ilustración 13 se muestra el flujo de Oauth que se tiene que llevar a cabo para obtener los tokens de acceso del usuario. En este caso OauthServer sería Google YouTube Data API V3 Oauth, Backend Server el servicio de autenticación mencionado en H1.1 y Frontend App sería el cliente de Next.js.

Posteriormente al obtener el token es necesario validar si la red social ya existe en la marca. Hacemos una petición al YouTube DATA API V3 para obtener los datos de la red social. Si no existe se crea y si existe simplemente se actualizan los datos de la Red Social y de Oauth (entidad que se muestra en la ilustración 10). Es importante hacer una separación de las entidades de red social y de Oauth por seguridad de los tokens de accesos que Oauth almacena.

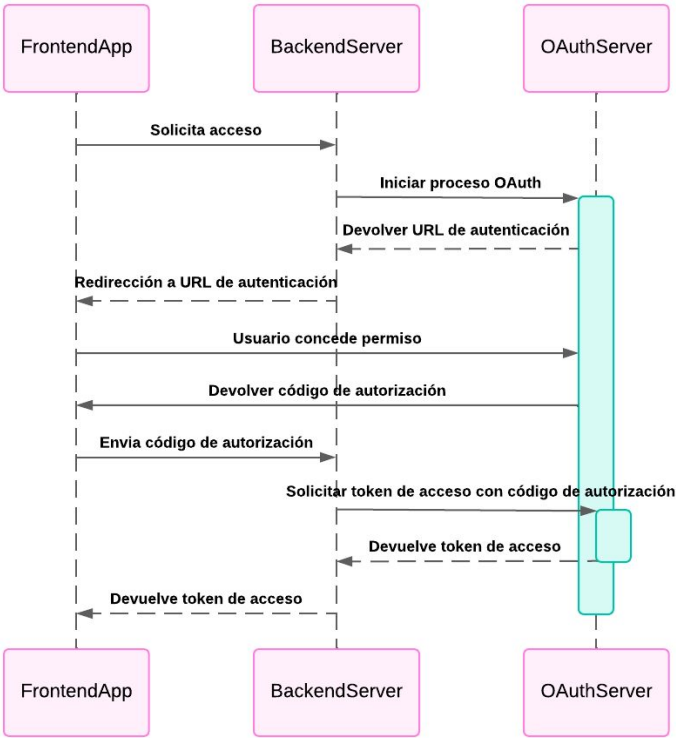


Ilustración 13 Oauth Flow

Es importante aclarar que la aplicación funciona actualmente en localhost y con los usuarios de prueba agregados manualmente. Para poder agregar usuarios externos hay que pasar por un proceso de verificación y tener un dominio valido para los callbacks de la aplicación.

6.3.3. *Revisión del sprint 2*

Todas las tareas fueron completadas con éxito. Finalmente se hicieron 24 PH, muchos más de los que originalmente fueron planificados (tabla 20).

Tabla 20 Tareas del sprint 2 - revisión

ID tarea	Tarea	Prioridad	Puntos historia (PH)	Puntos reales	Estado
H4.4	Modificar el nombre de una marca	Could	2	3	Completado
H4.5	Agregar miembros a la marca (admin)	Must	3	5	Completado
H4.6	Ver miembros del equipo	Must	2	2	Completado
H4.7	Eliminar miembros a la marca (admin)	Should	1	2	Completado
H5.1	Agregar cuenta de YouTube	Must	7	15	Completado



Para este sprint el cliente hizo una corrección y es que no le queda claro cuando alguien es admin o no al visualizar las marcas. Se ha propuesto hacer un icono que identifique que la persona que ha iniciado sesión es quien es el administrador de la marca. Además, solicitó una mejora de la interfaz que muestra las redes sociales asociadas a una marca. Recomendando el uso de los colores de cada red social.

#### 6.3.4. Retrospectiva del sprint 2

Este sprint se dedicó muchas horas de investigación y aprendizaje. Lo más difícil, fue el manejo de MongoDB en Next.js puesto que este framework implementa un entorno serverless. Es decir, cada petición se considera un evento independiente, produciendo que en cada petición originalmente no haya ningún esquema de MongoDB mapeado aún con la intención de usar los mínimos recursos posibles. En consecuencia, la arquitectura del código del proyecto Next.js para administrar las peticiones de la base de datos tuvo que ser reorientada para su correcto funcionamiento. Siendo este motivo que los PH reales de la tarea H5.1 fuesen tan altos.

Inicialmente se planteó una menor cantidad de horas invertidas este sprint. De modo que los 9 PH extras equivalentes a 22.5 horas extras.

### 6.4. Sprint 3

#### 6.4.1. Planificación del sprint 3

Duración 2 semanas.

Para este sprint se tienen 31 PH como se muestra en la tabla 21. El objetivo de este sprint es aprovechar el tiempo de las vacaciones y aumentar la cantidad de historias.

Tabla 21 Tareas del sprint 3

ID tarea	Tarea	Prioridad	Puntos historia (PH)
H5.2	Agregar cuenta de Facebook	Must	7
H5.3	Agregar cuenta de TikTok	Must	7
H5.6	Eliminar cuenta de red social (admin)	Could	2

H5.7	Actualizar tokens de acceso automáticamente	Must	7
H6.1	Configurar bucket	Must	5
HE.1	Agregar icono diferenciador al admin de una marca	Should	1
HE.2	Mejora visualización de las redes sociales de una marca	Should	2

#### 6.4.2. Desarrollo del sprint 3

H5.2 para agregar una cuenta de Facebook es necesario tener una cuenta de Facebook Business Manager, crear una cuenta de Meta Developer y crear la aplicación. Este proceso implementa un protocolo OAuth como YouTube. Sin embargo, el proceso de Facebook es más complicado puesto que para poder usar sus servicios es necesario verificar el Business Manager primero y luego la aplicación (ilustración 14).

##### Verifications

###### Business verification

This is required to get access to data from users (for some apps this is called advanced access). Only people with full control of a Business Account can complete this process. [Learn more about business verification](#).



Ronald Ris Business  
ID: 214453013078894  
● Incomplete

Remove

Complete verification

###### Access verification

Verify that your business is a Tech Provider. This is an additional step that is required to get access to the Meta business assets and information of other businesses. We review submissions and follow up within 5 days. [Learn about access verification](#)

⚠ To start access verification, you need to complete business verification.  
[Start business verification](#)

Start verification

Ilustración 14 Verificación de App - Business Manager – META

Es posible proceder con la aplicación sin verificar, pero únicamente con los scopes (permisos) que no requieren acceso avanzado. Sin embargo, si los scopes requieren acceso avanzado como se muestra en la ilustración 15 es necesario verificar el Business Manager por medio de una empresa real con sus datos de contribuyente de impuestos, registro del negocio y otros documentos.

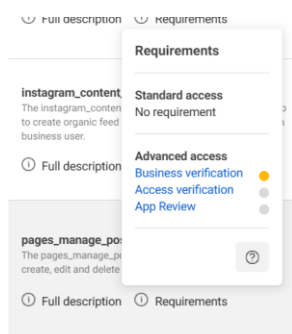


Ilustración 15 Requisitos para el uso de permisos de una aplicación de META

Por lo tanto, obtener los tokens de usuarios de manera dinámica fue imposibilitado. La única forma de conseguirlo fue por medio de Graph API Explorer de META como se muestra en la ilustración 16. Sin embargo, esta es una solución manual, por lo cual no cumple con lo que se busca en el proyecto.

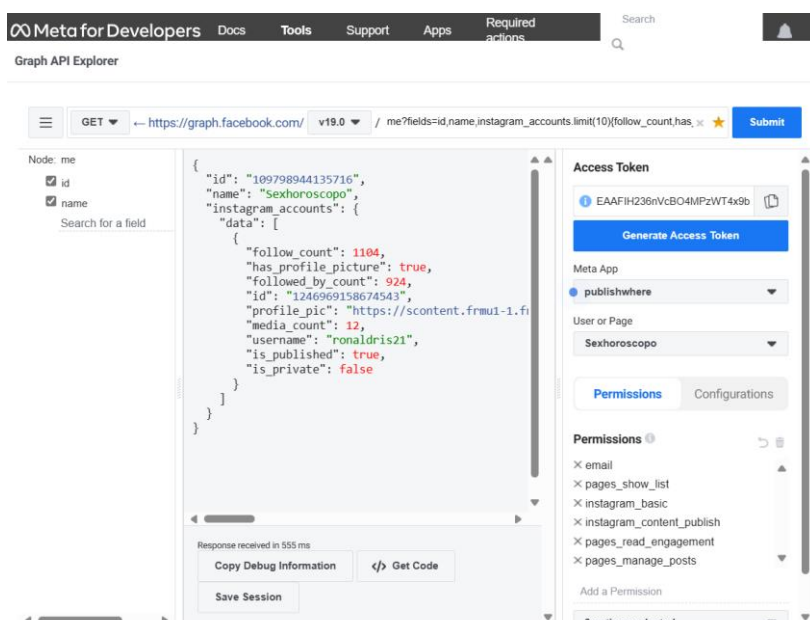


Ilustración 16 Graph API Explorer - META Developers

H5.3 para agregar una cuenta es necesario crear una aplicación en la consola de TikTok Developer. El proceso es relativamente sencillo. Sin embargo, TikTok requiere una aplicación con términos de servicio y políticas de seguridad de un dominio válido, así como la validación de los URL del dominio donde se encuentra la app como el URL donde se encuentra el bucket. Por lo cual, se ha desistido temporalmente con esta tarea porque hay que hacer las tareas de despliegue del apartado 7 para poder continuar con esta tarea.

H5.6 esta tarea elimina la red social de la marca y a la vez su información Oauth. Únicamente el admin de la marca puede eliminar la red social.

H5.7 actualmente solo hay cuentas de YouTube así que el servicio de renovación de tokens únicamente implementa esta red social. Es un cron de Express que se ejecuta cada 20 minutos para revisar si hay algún token que va a expirar en menos de 50 minutos. Se ha decidido hacerlo así puesto que en caso de que el backend llegase a caer y el contador se reiniciase, siempre tendría una oportunidad extra para refrescar el token Oauth.

H6.1 se ha elegido AWS S3 como bucket. Para configurar un bucket es necesario crear necesario elegir donde se ubicarán los datos. El bucket actual se ubica en la región Europa (París) eu-west-3. Y tiene políticas de acceso públicas y de escritura por medio de claves de acceso privada.

HE.1 para esto se ha agregado un icono que se renderiza al lado izquierda del nombre de una marca como identificador que el usuario logueado es el admin de esa marca.

HE.2 se han agregado tarjetas con el icono del proveedor de la red social y abajo el nombre de la cuenta de Red Social registrada en la marca.

#### 6.4.3. Revisión del sprint 3

En este sprint hubo múltiples tareas que han sido bloqueadas por motivos externos o bien porque requieren de otras tareas para poder ejecutarlas. Como se muestra en la tabla 22, en este sprint solo se lograron 14 PH.

Tabla 22 Tareas del sprint 3 - revisión

ID tarea	Tarea	Prioridad	Puntos historia (PH)	Puntos reales	Estado
x	Agregar cuenta de Facebook	Must	7		Bloqueada
H5.3	Agregar cuenta de TikTok	Must	7		Bloqueada
H5.6	Eliminar cuenta de red social (admin)	Could	2	3	Completado
H5.7	Actualizar tokens de acceso automáticamente	Must	7	3	Parcial
H6.1	Configurar bucket	Must	5	5	Completado
HE.1	Agregar icono diferenciador al admin de una marca	Should	1	1	Completado

HE.2	Mejora visualización de las redes sociales de una marca	Should	2	2	Completado
------	---	--------	---	---	------------

La duración del sprint fue 3 semanas en lugar de las 2 semanas planificadas. Esto fue debido a una lesión del hombro izquierdo que me impidió trabajar por varios días.

Esta semana no hubo reunión con el producto owner por falta de disponibilidad de su parte. Se seleccionarán las historias de usuario para el siguiente sprint por parte del scrum máster sin intervención del cliente.

#### 6.4.4. Retrospectiva del sprint 3

Se encontraron nuevamente problemas, esta vez por terceros o bien tareas que han sido bloqueadas. Si el problema no se soluciona de manera rápida solo se me ocurren dos soluciones: extender el proyecto o bien omitir dichas redes sociales.

Por el momento, se estima que aún es posible obtener un MVP para la entrega del TFG el 20 de mayo de 2024. Sin embargo, es necesario aumentar la velocidad de PH por sprint o bien no encontrar nuevos problemas en los próximos sprints.

### 6.5. Sprint 4

#### 6.5.1. Planificación del sprint 4

Duración 2 semanas.

Para este sprint se han planificado 28 puntos de historia (tabla 23). La intención es compensar los inconvenientes anteriores por medio de un mayor esfuerzo del equipo de desarrollo.

Tabla 23 Tareas del sprint 4

ID tarea	Tarea	Prioridad	Puntos historia (PH)
H6.2	Subir archivos de manera segura	Should	5
H6.3	Ver imágenes subidas	Must	1

H6.4	Ver videos subidos	Must	2
H6.5	Ordenar por tipo de archivo	Must	1
H6.6	Buscar archivos por nombre	Should	2
H6.7	Visualizar quién subió los archivos	Should	1
H6.10	Eliminar archivos	Should	2
H6.11	Visualizar los archivos que no han sido usado en publicaciones	Must	2
H6.12	Visualizar los archivos que ya han sido usado en publicaciones	Must	2
H6.13	Visualizar los archivos que han sido programados para publicar	Should	2
H6.14	Descargar archivos subidos	Must	1
H7.12	Visualizar las publicaciones por marca en forma de calendario	Must	7

#### 6.5.2. *Desarrollo del sprint 4*

H6.2 para subir archivos de manera segura se ha configurado el siguiente flujo. Desde el frontend por medio de un formulario para subir el archivo, mandando a llamar un server action que retorna un URL firmado en base al SHA256, el tipo del fichero y su tamaño. Lidiando así con subidas de ficheros no autorizados fuera del frontend por el formulario proporcionado. Este URL se retorna al cliente y se sube el archivo directamente al bucket de AWS S3.

H6.3 se leen los ficheros de la base de datos y se hacen las peticiones al bucket para mostrarlos en la biblioteca digital.

H6.4 similar al H6.3, pero esta vez con videos. La diferencia es que según el tipo de datos ahora renderiza un video que el usuario puede reproducir.

H6.5 se filtran por medio de un selector de Shacdn. Es posible filtrar por: Todos, videos e imágenes.

H6.6 se ha creado una barra de buscar que venía integrada con la librería de react-table para filtrar los ficheros.

H6.7 Se modificaron los componentes creados en H6.3 y H6.4 agregando a través de popular los datos del fichero con los datos del creador a través de una consulta de MongoDB.

H6.10 se usa un atributo de la entidad Fichero llamado `shouldDelete` para gestionar si un archivo debe eliminarse o no. Actualmente la aplicación no elimina los ficheros, solo los envía temporalmente a la papelera de reciclaje con la intención de posteriormente borrarlos por medio del Cron Job Service para eliminar el fichero si ha pasado 15 días en la papelera de reciclaje.

Para las tareas H6.11, H6.12 y H6.13 se usa la siguiente vista (ilustración 17). Para crear las vistas se ha tratado de dividir en componentes de la manera más óptima posible, donde según el tipo de ficheros que se quiere visualizar es la consulta que se hace a la base de datos.

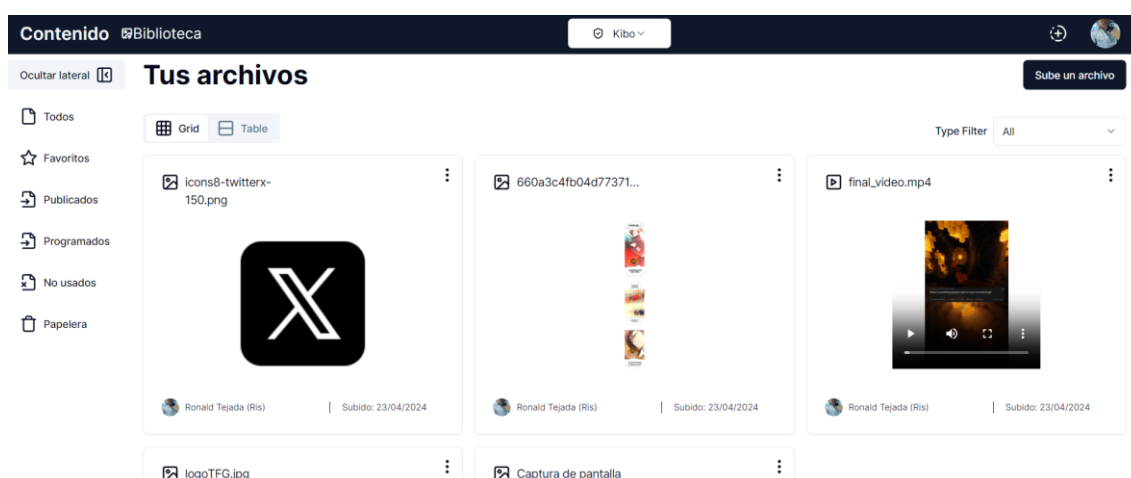


Ilustración 17 Biblioteca digital por marca

H6.14 se ha creado un botón en las acciones de la tarjeta de cada fichero para descargar. El archivo se descarga automáticamente al darle clic en descargar.

H7.12 para este apartado se ha seguido un tutorial para un clone de Google Calendar hecho en React con Js (Codes, 2021). Se tomo en cuenta la interfaz y se convirtieron los scripts de Js a Ts para poder utilizar el tipado de datos y las interfaces de las Publicaciones. Siendo así compatible con los esquemas de la base de datos de MongoDB. El resultado de la vista sin las publicaciones agregadas se puede ver en la ilustración 18.

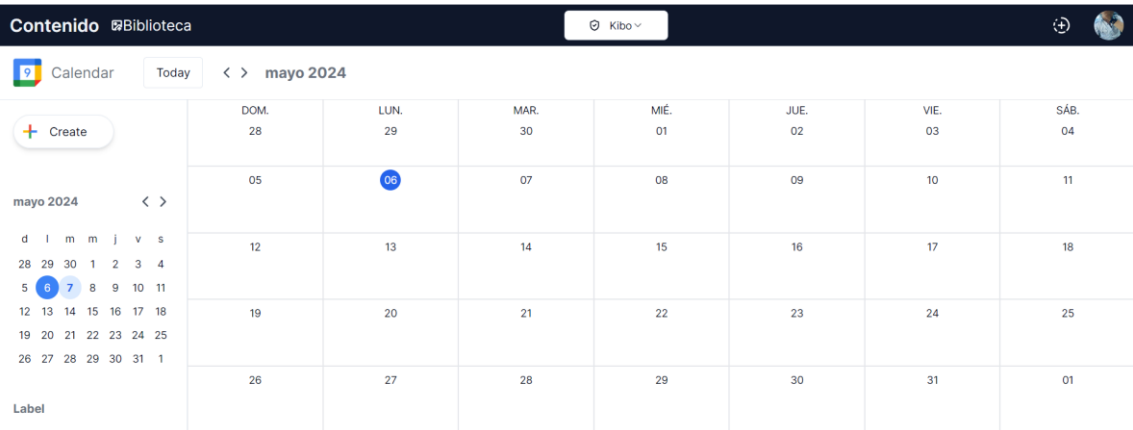


Ilustración 18 Calendario de publicaciones - sin funcionalidad

6.5.3. Revisión del sprint 4

Los puntos de PH reales del sprint 4 fueron 26 (tabla 24). Todas las tareas fueron completadas con éxito a excepción de una que será retomada posteriormente cuando se creen publicaciones para poder mostrarlas.

Tabla 24 Tareas del sprint 4 - revisión

ID tarea	Tarea	Prioridad	Puntos historia (PH)	Puntos reales	Estado
H6.2	Subir archivos de manera segura	Should	5	5	Completado
H6.3	Ver imágenes subidas	Must	1	1	Completado
H6.4	Ver videos subidos	Must	2	3	Completado
H6.5	Ordenar por tipo de archivo	Must	1	1	Completado
H6.6	Buscar archivos por nombre	Should	2	3	Completado
H6.7	Visualizar quién subió los archivos	Should	1	1	Completado
H6.10	Eliminar archivos	Should	2	2	Completado
H6.11	Visualizar los archivos que no han sido usado en publicaciones	Must	2	3	Completado
H6.12	Visualizar los archivos que ya han sido usado en publicaciones	Must	2	3	Completado
H6.13	Visualizar los archivos que han sido programados para publicar	Should	2	3	Completado
H6.14	Descargar archivos subidos	Must	1	1	Completado
H7.12	Visualizar las publicaciones por marca en forma de calendario	Must	7		Parcial



Al terminar el sprint se tuvo una reunión corta con el producto owner donde se le mostraron los avances actuales. Ella se mostró satisfecha por los cambios y los avances visuales de la biblioteca digital y la idea de usar el mismo formato de Google Calendar ya que es una interfaz conocida.

Por otra parte, se le comento los problemas encontrados en el sprint 3 con Facebook e Instagram para agregar cuentas. La necesidad de tener una cuenta verificada de Business Manager con una empresa real. Su respuesta fue que intentará resolverlo por mí mismo de alguna forma y ver si puedo obtener la verificación.

Finalmente, se le hizo entender que el siguiente sprint es necesario hacer la parte de Despliegue para ya que hay tareas bloqueadas que requieren este paso. El cliente estuvo de acuerdo.

#### *6.5.4. Retrospectiva del sprint 4*

Este sprint fue bastante fructífero. La curva de aprendizaje del framework ha sido manejada con éxito y se ha conseguido fluidez con Typescript.

Quedan 2 sprints hasta la entrega del proyecto, se estima que la velocidad de PH por sprint aumentará y eso será conveniente para conseguir las historias faltantes. Además, el último sprint se tendrá mayor disponibilidad de tiempo para dedicar al proyecto.

### **6.6. Sprint 5**

#### *6.6.1. Planificación del sprint 5*

Duración 2 semanas.

Para este sprint se han planificado 25 PH (tabla 25). El objetivo principal de este sprint es configurar el entorno en la nube para que la aplicación sea accedida por usuarios finales.

Tabla 25 Tareas del sprint 5

ID tarea	Tarea	Prioridad	Puntos historia (PH)
H8.1	Comprar dominio	Must	2
H8.2	Comprar VPS	Must	2
H8.3	Configurar seguridad básica	Should	3
H8.4	Instalar software en el VPS	Must	2
H8.5	Configurar repositorio	Must	2
H8.6	Crear los archivos .env para producción	Must	1
H8.7	Configurar sitios web y DNS	Must	3
H8.8	Iniciar las aplicaciones utilizando un administrador de procesos	Must	2
H8.9	Configurar monitorización y rendimiento	Should	3
H8.10	Configurar CI/CD	Should	5

#### 6.6.2. Desarrollo del sprint 5

El desarrollo completo de este apartado se desarrolla en el apartado 7.

#### 6.6.3. Revisión del sprint 5

Para terminar este sprint se consiguieron 29 PH y todas las historias (tabla 26). Además, la duración de este sprint ha excedido las 2 semanas. El motivo fue la elaboración de la memoria presente para la entrega al tutor.

Tabla 26 Tareas del sprint 5 - revisión

ID tarea	Tarea	Prioridad	Puntos historia (PH)	Puntos reales	Estado
H8.1	Comprar dominio	Must	2	1	Completado
H8.2	Comprar VPS	Must	2	2	Completado
H8.3	Configurar seguridad básica	Should	3	2	Completado
H8.4	Instalar software en el VPS	Must	2	2	Completado
H8.5	Configurar repositorio	Must	2	3	Completado
H8.6	Crear los archivos .env para producción	Must	1	1	Completado
H8.7	Configurar sitios web y DNS	Must	3	3	Completado
H8.8	Iniciar las aplicaciones utilizando un administrador de procesos	Must	2	3	Completado
H8.9	Configurar monitorización y rendimiento	Should	3	2	Completado

H8.10	Configurar CI/CD	Should	5	10	Completado
-------	------------------	--------	---	----	------------

En la reunión con el cliente este estuvo satisfecho que la aplicación ya es accesible desde su portátil. Se volvió a tocar el tema sobre Meta – Business Manager y se intentó hacerlo con su cuenta personal. Sin embargo, tampoco estaba aprobada como un negocio legal. Así que se concluyó en omitir tanto Facebook como Instagram en el alcance del proyecto.

El cliente queda expectante para la última reunión el jueves 16 de mayo de 2024 para la versión final del software y poder ver que se publica en redes sociales.

#### 6.6.4. Retrospectiva del sprint 5

Al configurar la aplicación en el VPS algunas funcionalidades dejaron de funcionar correctamente. Parte del tiempo de este sprint se dedicaron a solventar estos problemas. Algunas configuraciones decidieron eliminarse y tendrán que volver a hacerse. Hay una funcionalidad que está limitando actualmente el CI/CD al momento de hacer el build, se tomará cartas en el asunto para modificar la configuración de webpack del proyecto.

En general, se obtuvo un buen resultado este sprint. El próximo se espera dedicar el doble de horas por semana para cumplir el objetivo general 1 en el último sprint pues es el único de los objetivos aún pendiente.

Ronald Ernesto Tejada Ríos

## 7. CONFIGURACIÓN Y DESPLIEGUE DE LA SOLUCIÓN

### 7.1. Dominio y VPS

Para este paso se ha usado Hostinger para comprar el dominio <https://PublishWhere.com> y un VPS donde se desplegarán las aplicaciones.

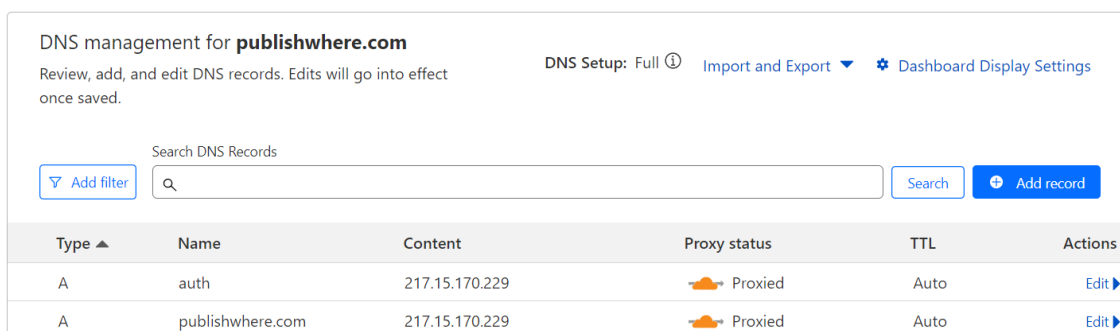
Las especificaciones del VPS (KVM 2) son:

- 2 núcleos de vCPU
- 8 GB RAM
- 100 GB de espacio en disco NVMe
- 8 TB de ancho de banda

### 7.2. Configuración de seguridad básica

Para esto se ha utilizado Cloudflare para configurar SSL/TLS entre el frontend, Cloudflare y el servidor como se observa en la ilustración 2. Asimismo, se ha configurado que convierta todas las peticiones de HTTP a HTTPS.

Además, Cloudflare ofrece mecanismos de defensa ante ataques DDOS y DOS, como CDN mejorando la velocidad de carga del sitio web y se ha configurado el DNS como se muestra en la ilustración 19:



DNS management for **publishwhere.com**

Review, add, and edit DNS records. Edits will go into effect once saved.

DNS Setup: Full ⓘ Import and Export ▾ ⚙ Dashboard Display Settings

Search DNS Records

▼ Add filter  Search + Add record

Type ▲	Name	Content	Proxy status	TTL	Actions
A	auth	217.15.170.229	☁ Proxied	Auto	Edit ▶
A	publishwhere.com	217.15.170.229	☁ Proxied	Auto	Edit ▶

Ilustración 19 Configuración del DNS y el dominio

### 7.3. Instalar software en el VPS

A través de MobaXterm por medio de una conexión SSH se han instalado varias herramientas en el VPS para ejecutar las aplicaciones. Entre ellas se encuentran nginx, git, nvm y pm2.

Se ha configurado y activado los archivos de nginx para los sitios web públicos <https://PublishWhere.com> y <https://Auth.PublishWhere.com>. Es importante reiniciar el proceso de nginx con systemctl para que los cambios surjan efecto.

Git para descargar el repositorio.

NVM para configurar node.js versión 20.

PM2 para administrar las aplicaciones de node.js en producción.

### 7.4. Configurar repositorio

Usando git se descarga el repositorio y por medio de FTP se suben los archivos .env de producción. Es importante que los client ID y client secret de las aplicaciones Oauth de los proveedores de redes sociales tiene que cambiarse por las claves de producciones con sus respectivos callback al dominio. Además, es importante cambiar las IP de los microservicios por IP en función del dominio o subdominio correspondiente de publishwhere.com.

Para configurar la ejecución constante del proyecto es importante configurar el script de ecosystem.config.js dentro de cada aplicación de Express y Next.js. Estos scripts se utilizarán para posteriormente ejecutar pm2 e iniciar los procesos. En estos scripts se configura el tiempo de reinicio de una aplicación si falla, sus logs de información y errores y la cantidad de instancias que pm2 va a ejecutar.

Es importante guardar la configuración de los servicios ejecutándose con pm2 save. Así cada vez que se reinicie la computadora, los mismos servicios se inician de manera automática.

## 7.5. Monitorización y rendimiento

Una parte importante es ofrecer el servicio 24 horas los 7 días de la semana. Para esto se han configurado las herramientas web de UptimeRobot y PM2 Monitoring.

PM2 Monitoring se encarga de mostrar en tiempo real los recursos que consume las aplicaciones que se ejecutan en el servidor. Así como ver los logs en tiempo real, siendo extremadamente útil para los primeros despliegues en el servidor. Sin embargo, una de sus funciones más útiles es la capacidad de poder reiniciar servicios en caso de alguna emergencia o error inesperado (ilustración 20).

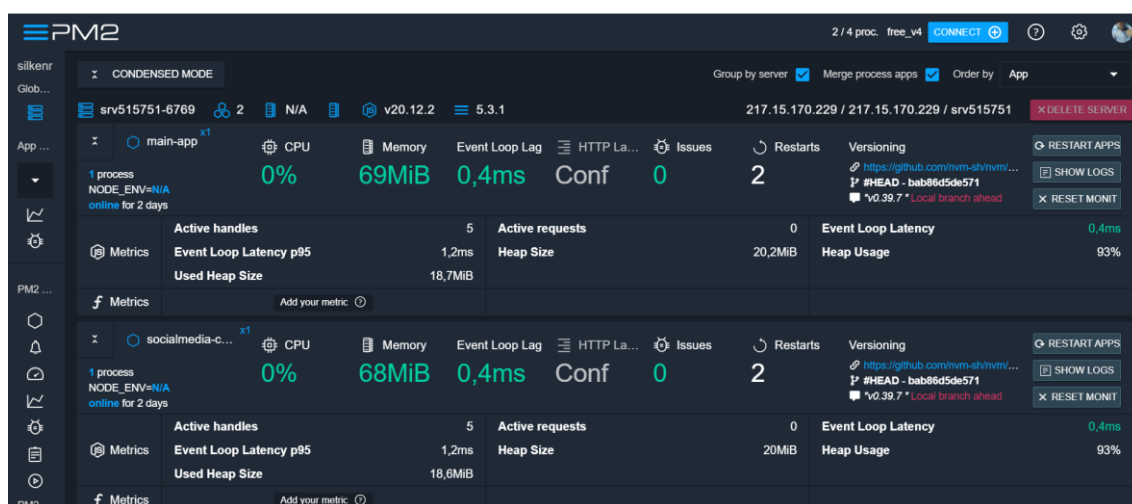


Ilustración 20 PM2 Monitoring

UptimeRobot es una aplicación web con la cual se lleva el seguimiento de un sitio web y si está activo. Hace peticiones cada 5 minutos para verificar que todo se está ejecutando correctamente (ilustración 21). En caso de que el sistema este caído, UptimeRobot envía un correo electrónico avisando que el servidor no ha respondido correctamente.

Al mezclar tanto UptimeRobot y PM2 Monitoring, juntos proveen una excelente combinación. Si el VPS no es capaz de reiniciar el servicio, UptimeRobot notifica por un correo y desde cualquier navegador es posible reiniciar las aplicaciones ejecutándose desde PM2 Monitoring.

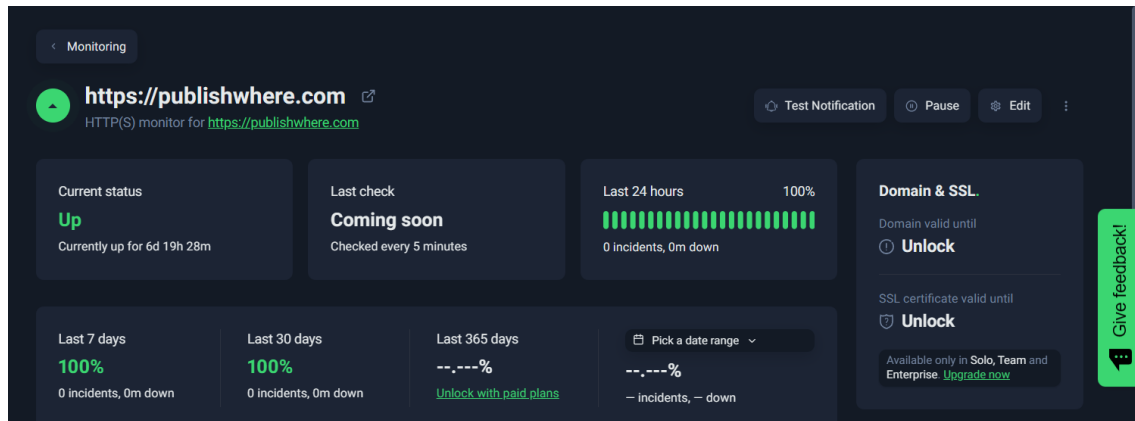


Ilustración 21 UptimeRobot

## 7.6. Continuos Integration / Continuos Deployment (CI/CD)

Se ha configurado un workflow de Github Actions para configurar ci/cd en el VPS. El flujo es el siguiente:

- Al recibir cualquier commit o pull request (closed) en la rama main, este ejecuta el flujo.
- El flujo se ejecuta en una maquina Ubuntu con Node.js 20 en la rama principal.
- Instala todas las dependencias e intenta hacer el build de la aplicación.
  - Primero verifica el tipado de Typescript para garantizar la seguridad y congruencia de las interfaces en toda la aplicación.
  - Ejecuta la herramienta de Eslint para encontrar potenciales errores o advertencias (Eslint, 2024). Dependiendo de la configuración puede ignorar las advertencias y errores en específicos. Sin embargo, por defecto al encontrar un error este detiene todo el workflow de Github Actions
  - Next.js implementa webpack. Webpack se encarga de empaquetar el código, reorganizarlo y optimizarlo. Durante este proceso se comprueba si hay errores o problemas de módulos (Webpack, s.f.). En caso de encontrarse con ellos, el build falla y se detiene el workflow de Github Actions.



- Si el build es correcto. Ahora el workflow procede a conectarse por SSH al servidor por medio de la IP, el usuario y una clave RSA privada.
- Al estar en el servidor acá se ejecutan 3 comandos:
  - `cd /home/ubuntu/ris/PublishWhere`
  - `git pull origin main`
  - `/root/.nvm/versions/node/v20.12.2/bin/pm2 restart all`

Dando por terminado el flujo de integración y despliegue continuo.

Ronald Ernesto Tejada Ríos

## 8. CONCLUSIONES

### 8.1. Objetivos alcanzados

En el apartado 1 se han definido los objetivos generales y específicos de este proyecto. En resumen, se ha obtenido uno de los dos objetivos generales y se han obtenido seis de los seis objetivos específicos.

- OG1: Este objetivo aún no ha sido cumplido. Falta el último sprint de la aplicación en el que se cubrirá este objetivo general.
  - OE1.1: La aplicación efectivamente permite el manejo de una cantidad ilimitada de marcas dentro de la plataforma.
  - OE1.2: La aplicación posee una biblioteca digital por marca totalmente funcional y escalable para archivos multimedia de tipo video e imagen.
  - OE1.3: La aplicación no se ha implementa un inicio de sesión con Google que no tiene restricciones de usuarios. Cualquier persona con una cuenta de Google puede ingresar, por ende, no hay límite de usuarios.
- OG2- La aplicación ha sido hospedada en un dominio web el cual es accesible por cualquier persona que tenga el enlace.
  - OE2.1: Se ha implementado una arquitectura de multiservicios por medio de un repositorio monorepo y su implementación por medio de pm2.
  - OE2.3: Se han implementado medidas de seguridad del servidor por medio de Cloudfare y certificados SSL/TLS.
  - OE2.3: Se ha implementado un flujo de trabajo de Github Action que implementa automatiza los despliegues en el servidor desde la rama main del repositorio de Github.

## 8.2. Conclusiones del trabajo y personales

Este trabajo me ha dejado conclusiones tanto profesionales como personales. Las cuáles serán útiles para la toma de decisiones en proyectos futuros.

### **Profesionales:**

- Al momento de trabajar con software de terceros es útil hacer pruebas tempranas para anticipar problemas futuros o si es necesario buscar otra alternativa.
- El manejo de versiones de software por ramas es extremadamente útil para el desarrollo. Permite concentrarse en la tarea a hacer y no en avanzar un poco de todo a la vez. Permitiendo a la vez obtener el mayor valor posible en el menor tiempo debido a la definición de la tarea que se pretende resolver.
- La planificación positiva es contraproducente. Es mejor hacer pequeñas iteraciones y más sprints que pocos sprints y grandes periodos de tiempo.

### **Personales:**

- Cosas como los términos y condiciones son extremadamente importantes. Por culpa de ellos varias veces mis aplicaciones de API Developer fueron denegadas por no cumplir las normas establecidas. La parte legal es más importante de lo que consideraba.
- Al inicio de un proyecto es fácil encontrarte ante la parálisis de la tarea al observar tantas tareas. Trabajar con un cliente real fue una excelente experiencia porque sentía la presión y las expectativas de una persona sobre mi trabajo. Originalmente estaba abrumado por la gran cantidad de tareas, sin embargo, buscar obtener un MVP puede ser tu mejor opción en lugar de ir haciendo todas las tareas en orden “cronológico”.

### 8.3. Vías futuras

En este apartado se mencionan los posibles rumbos que este proyecto puede tomar tras la entrega final del TFG.

- Uso de la aplicación por invitación. Así una persona es agregada directamente al equipo de trabajo de una marca.
- Crear una LLC para poder usar las API Developer de Meta a través de la verificación del Business Manager y poder acceder a los servicios de publicar video en Instagram y Facebook.
- Agregar más redes sociales.
- Implementar distintos tipos de publicaciones como carruseles o historias de 24 horas.
- Implementar funciones de chat y notificaciones.
- Automatizar la publicación de contenido. Es decir, dejar las publicaciones preparadas de antemano, pero que el sistema decida cuando publicarlas por medio de datos estadísticos de la cuenta o por inteligencia artificial.

## 9. BIBLIOGRAFÍA

Ailin Orjuela Duarte, M. M. (2008). Las Metodologías de Desarrollo Ágil como una Oportunidad. *Revista Avances en Sistemas e Informática*, 161.

Ailin Orjuela Duarte, M. M. (2008). Las Metodologías de Desarrollo Ágil como una Oportunidad. *Revista Avances en Sistemas e Informática*, 159-160.

Ailin Orjuela Duarte, M. M. (2008). Las Metodologías de Desarrollo Ágil como una Oportunidad. *Revista Avances en Sistemas e Informática*, 159.

Atlassian. (s.f.). *Historias, epics e iniciativas*. Recuperado el 5 de mayo de 2024, de Agile: <https://www.atlassian.com/es/agile/project-management/epics-stories-themes>

Atlassian. (s.f.). *Jira*. Obtenido de <https://www.atlassian.com/es/software/jira>

AWS. (s.f.). *Descripción general de los buckets - Amazon Simple Storage Service*. Recuperado el 30 de abril de 2024, de [https://docs.aws.amazon.com/es\\_es/AmazonS3/latest/userguide/UsingBucket.html](https://docs.aws.amazon.com/es_es/AmazonS3/latest/userguide/UsingBucket.html)

Bara, M. (11 de mayo de 2020). *Diagramas Gantt en Agile... ¿una buena combinación?* Obtenido de OBS Business School: <https://www.obsbusiness.school/blog/diagramas-gantt-en-agile-una-buena-combinacion>

Cloudflare. (s.f.). *SSL/TSL*. Obtenido de <https://dash.cloudflare.com/7cdc6e00ed153ee7dddd56b1b1d4bf42/publicshwhere.com/ssl-tls>

Cloudflare. (s.f.). *¿Qué es OAuth? | SAML vs. OAuth | Cloudflare*. Recuperado el 23 de mayo de 2024, de <https://www.cloudflare.com/es-es/learning/access-management/what-is-oauth/>

Codes, E. (21 de septiembre de 2021). *Google Calendar Clone with React - React Hooks ,React Context and Tailwind*. Obtenido de Youtube: <https://www.youtube.com/watch?v=KUKyTRYGrnU>

Eslint. (3 de mayo de 2024). *Find and fix problems in your JavaScript code - ESLint - Pluggable JavaScript Linter*. Obtenido de <https://eslint.org/>

Intel. (s.f.). *Microservicios y arquitectura de microservicios*. Recuperado el 30 de abril de 2024, de <https://www.intel.la/content/www/xl/es/cloud-computing/microservices.html>

Jobtec. (2024). *Sueldo del Full Stack Developer en España*. Recuperado el 05 de mayo de 2024, de Salarios: <https://www.jobted.es/salario/full-stack-developer>

Kanban. (2022). *The Official Guide to the Kanban Method*. Seattle: Mauvius Group Inc.

Keymetrics Inc. (s.f.). *PM2 - Home*. Obtenido de <https://pm2.io/>

Kuhn, J. (2009). Decrypting the MoSCoW Analysis. *itSM Solutions*, 1-2.

Mobatek. (s.f.). *MobaXterm*. Obtenido de <https://mobaxterm.mobatek.net/>

Navarro, D. (2023). *Comparativa entre Vue.js, React y Angular: ¿Cuál es el mejor framework de JavaScript en 2023?* Obtenido de programee: <https://programee.com/programacion/comparativa-vue-react-angular/#:~:text=Curva%20de%20Aprendizaje%20Vue.js%20destaca%20por%20su%20simplicidad,puede%20resultar%20m%C3%A1s%20desafiante%20para%20los%20nuevos%20usuarios>.

Next.js. (s.f.). *Next.js by Vercel - The React Framework*. Obtenido de <https://nextjs.org/>

Next.js. (s.f.). *Server Actions and Mutations*. Obtenido de <https://nextjs.org/docs/app/building-your-application/data-fetching/server-actions-and-mutations>

nginx. (s.f.). *nginx*. Obtenido de <https://nginx.org/en/>

TestGorilla. (3 de mayo de 2024). *SQL vs. NoSQL: comparación completa de características, diferencias y más*. Obtenido de <https://www.testgorilla.com/es/blog/sql-vs-nosql/>

Webpack. (s.f.). *Webpack*. Obtenido de <https://webpack.js.org/>

Xe. (30 de abril de 2024). *1 USD to EUR - US Dollars to Euros Exchange Rate*.

Recuperado el 30 de abril de 2024, de  
<https://www.xe.com/currencyconverter/convert/?Amount=1&From=USD&To=EUR>