

SOCIALSHARE



Trabajo fin de ciclo de
Guillermo Martínez de la Riva

Índice

Introducción (1)	2
Objetivos del TFC (2)	2
Objetivos de la aplicación (2.1)	2
Fases del proyecto (3)	3
Requisitos (3.1)	3
Requisitos funcionales (3.1.1)	3
Requisitos no funcionales (3.1.2)	4
Análisis (3.2)	5
Diseño (3.3)	12
Arquitectura del sistema (3.3.1)	12
Capa de presentación (3.3.2)	12
Capa de datos (3.3.3)	13
Interfaz gráfica (3.3.4)	14
Diseño de la app (3.3.4.1)	14
Diseño previo a la app (3.3.4.2)	14
importación de iconos (3.3.4.3)	16
Aplicar diseño en la app (3.3.4.4)	16
Implementación y configuración (3.4)	18
Tecnologías empleadas (3.4.1)	18
Metodología de trabajo (3.4.2)	19
Planificación (3.4.3)	19
Desarrollo de la aplicación (3.4.4)	20
Sprint 1	20
Sprint 2	22
Sprint 3	24
Sprint 4	26
Sprint 5	28
Pruebas (3.5)	29
Ampliación y posibles mejoras (4)	30
Conclusión (5)	31

Bibliografía (6)**31**

Introducción (1)

La creación de este proyecto ha sido fruto del uso de las diferentes redes sociales que hay en el mercado.

Existen muchas redes sociales no tan conocidas, las cuales solo están desarrolladas en android, por ello, quería desarrollar una aplicación híbrida, que puede ser una gran solución para desarrolladores pequeños con menos recursos. De esta forma optimizamos el tiempo y quitamos el gasto que conlleva tener que comprarse un Mac para poder programar para ios.

De todas formas más adelante comentaré algunas desventajas que me he ido encontrando con el desarrollo.

Por otro lado, quería hacer un diseño un poco diferente al resto, sin romper la estructura básica que tiene una red social.

Por último, quería aprender una nueva tecnología y gracias a la idea de que fuera una aplicación híbrida he tenido la necesidad de usar en este caso Ionic.

Objetivos del TFC (2)

Con la realización de este trabajo, se pretende desarrollar una aplicación móvil que ofrezca una funcionalidad similar a Instagram y Twitter.

Gracias a las tecnologías empleadas que hablaré posteriormente, podré desarrollar esta aplicación con un mismo lenguaje tanto para IOS como para Android.

Otro de los objetivos de este trabajo, es percibir qué ventajas y desventajas tiene desarrollar aplicaciones híbridas, frente a las aplicaciones nativas.

Un punto muy importante de este proyecto es poder realizar esta aplicación con los conocimientos básicos adquiridos durante el curso y aplicarlo a lenguajes y tecnologías diferentes, para ver cómo me desarrollaría en un entorno desconocido y con la necesidad de investigar para llevar a cabo el trabajo.

Objetivos de la aplicación (2.1)

En este apartado se definen los requisitos que debe cumplir la aplicación:

- La interfaz de la aplicación, debe adaptarse en dispositivos móviles y en tabletas.
- La aplicación debe dar soporte a dispositivos móviles con Android e IOS.
- Gestionar el almacenamiento de información y archivos multimedia tanto en la nube como en el propio dispositivo.
- La aplicación debe cargar la información del usuario con y sus publicaciones.
- La aplicación permite compartir fotos, videos y comentarios.

- EL usuario en la aplicación puede modificar su usuario y correo.
- El usuario puede ver las publicaciones que se han guardado.
- El usuario puede dar “me gusta” a las publicaciones

Fases del proyecto (3)

Requisitos (3.1)

Requisitos funcionales (3.1.1)

Nombre: Iniciar sesión.

Descripción: La aplicación permitirá autenticarse al usuario, a través de unas credenciales. En caso de ser incorrectas, la aplicación ha de ser capaz de diferenciar entre los dos casos.

Nombre: Crear usuario.

Descripción: La aplicación permitirá registrar al usuario, a través de los datos requeridos. En caso de ser incorrectos, la aplicación ha de ser capaz de diferenciar entre los dos casos.

Nombre: Ver publicaciones de los usuarios.

Descripción: La aplicación debe mostrar en una ventana las publicaciones de los diferentes usuarios, la cual se deberá actualizar según añadan más publicaciones.

Nombre: Añadir / Eliminar like.

Descripción: El usuario podrá dar y quitar like de una publicación desde cualquier ventana de la app donde haya publicaciones.

Nombre: Añadir / Eliminar publicaciones guardadas.

Descripción: El usuario podrá añadir y eliminar de guardados de una publicación desde cualquier ventana de la app donde haya publicaciones.

Nombre: Mostrar los usuarios que han dado like.

Descripción: El usuario ver quien ha dado like a una publicación desde cualquier ventana de la app donde haya publicaciones.

Nombre: Mostrar perfil de un usuario.

Descripción: El usuario podrá acceder al perfil de otro usuario desde el nombre o foto de perfil que se muestra en las publicaciones.

Nombre: Creación de publicaciones.

Descripción: La aplicación permitirá al usuario añadir varias imágenes y comentarios para la creación de la publicación. El usuario, tendrá acceso a la galería o a la cámara del dispositivo

Nombre: Editar perfil.

Descripción: El usuario podrá modificar sus atributos (Foto de perfil, nombre, email, descripción).

Nombre: Perfil personal.

Descripción: El usuario tendrá una ventana desde la que podrá ver sus publicaciones, las publicaciones guardadas, el historial de “me gustas” y la posibilidad de editar perfil.

Nombre: Token.

Descripción: La aplicación creará un token con las credenciales del usuario para mayor seguridad. Este token tendrá caducidad, de esta manera el usuario tendrá que volver a autenticarse cuando el tiempo expire.

Requisitos no funcionales (3.1.2)

Nombre: Diseño.

Descripción: La aplicación tendrá un diseño limpio y con colores cálidos. Tendrá una armonía entre colores para evitar cansar la vista del usuario.

Nombre: Gestión de errores.

Descripción: La aplicación ha de tratar los errores para evitar que se cierre inesperadamente. Además, deberá avisar de los errores con un lenguaje adecuado para los usuarios.

Nombre: Evitar pérdidas de información.

Descripción: La aplicación debe mantener la información del usuario de forma íntegra durante el envío de información al servidor, o el cambio de orientación del dispositivo, o en los cambios de actividades dentro de la app y aplicaciones externas.

Nombre: Portabilidad

Descripción: La aplicación deberá funcionar correctamente en diferentes versiones de dispositivos Android e IOS y adaptarse a los diferentes tamaños de pantalla.

Nombre: Interfaces

Descripción: Las interfaces de usuario deben ser intuitivas y de fácil aprendizaje para los usuarios.

Nombre: Carga de usuarios

Descripción: El sistema permitirá la carga de varios usuarios concurrentes.

Nombre: Ram.

Descripción: La aplicación no deberá consumir más de 500 Mb de ram

Nombre: Disco.

Descripción: La aplicación no podrá ocupar más de 700MB en disco.

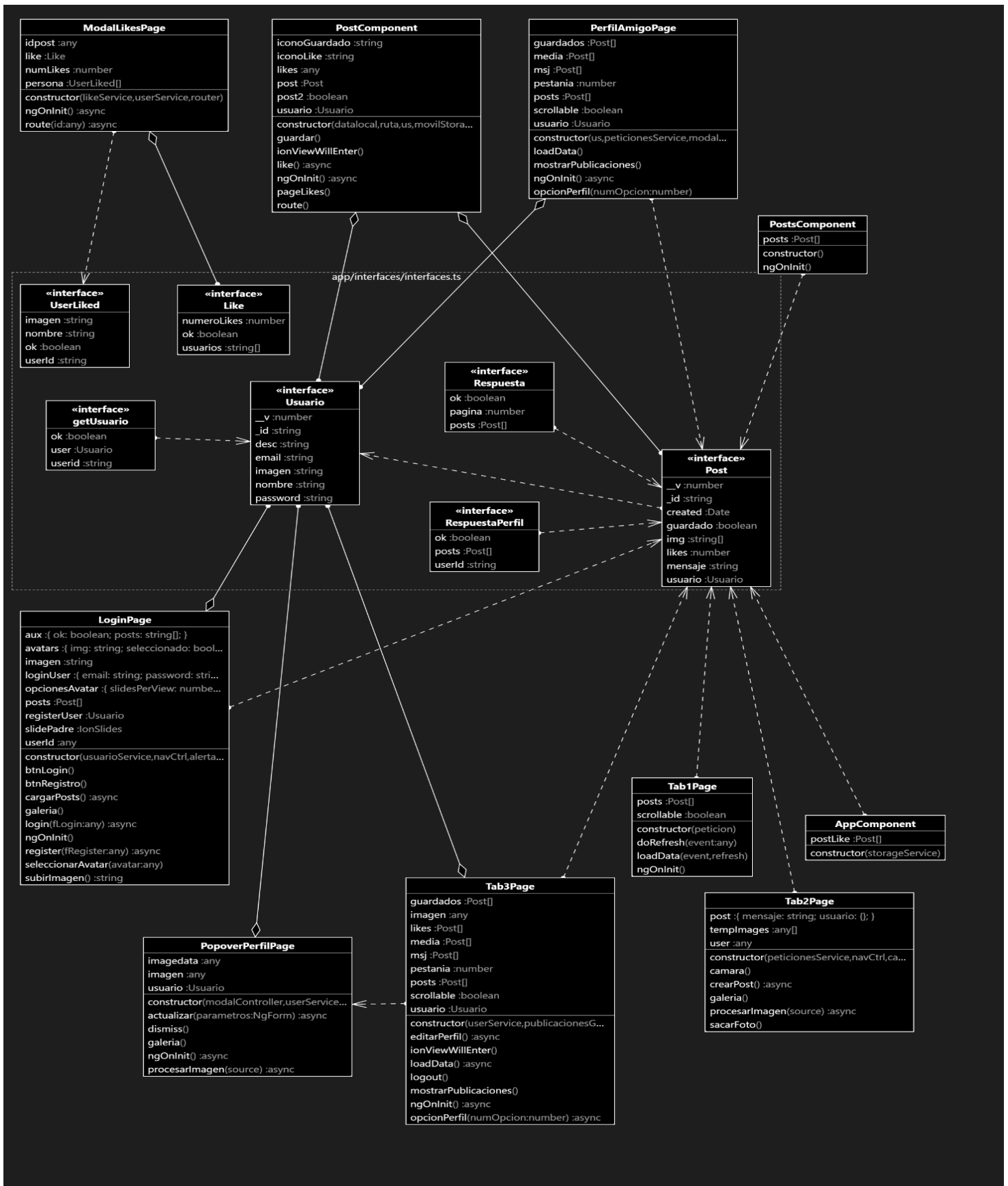
Nombre: Protección de datos.

Descripción: Deberá respetar la ley de protección de datos.

Análisis (3.2)

En primer lugar se muestra el modelo de negocio en forma de diagrama de clases, en él se muestran las diferentes entidades que forman parte de la aplicación con sus propiedades y métodos, salvo los servicios que conectan con la base de datos.

Diagrama de clases



Por otro lado, se muestran varios diagramas de casos de uso implementando los requisitos funcionales vistos anteriormente.

Diagrama de casos de uso general

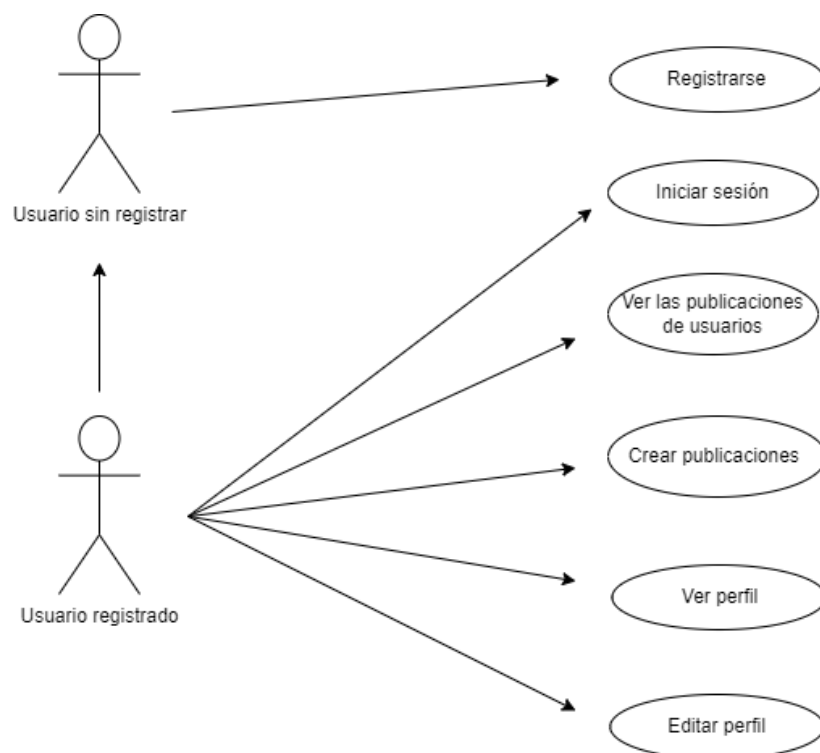


Diagrama de casos de uso de ver publicaciones de usuario

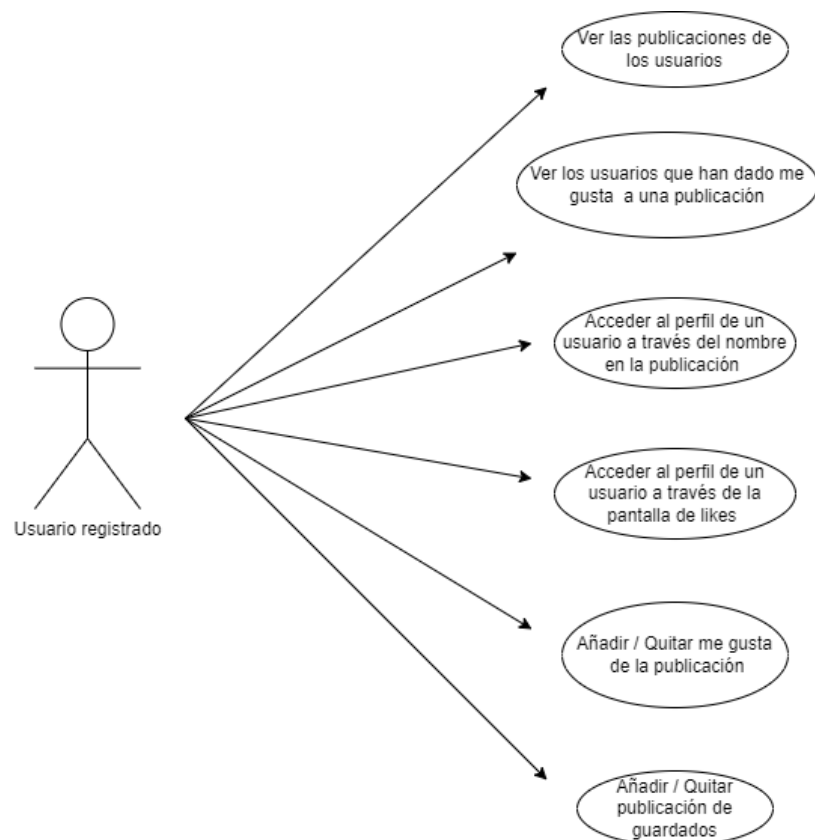


Diagrama de casos de uso de crear publicaciones

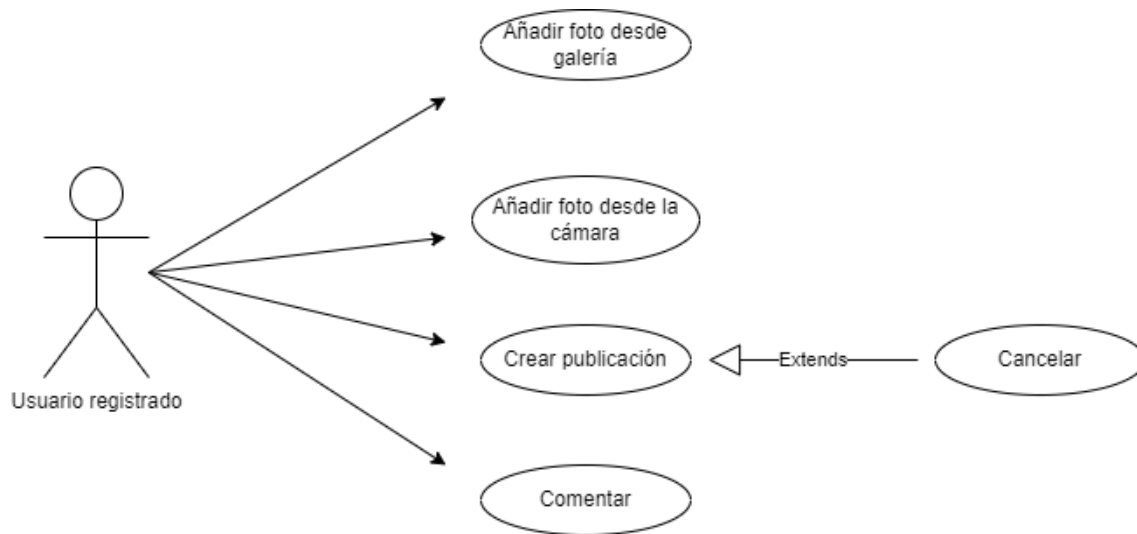


Diagrama de casos de uso de ver perfil

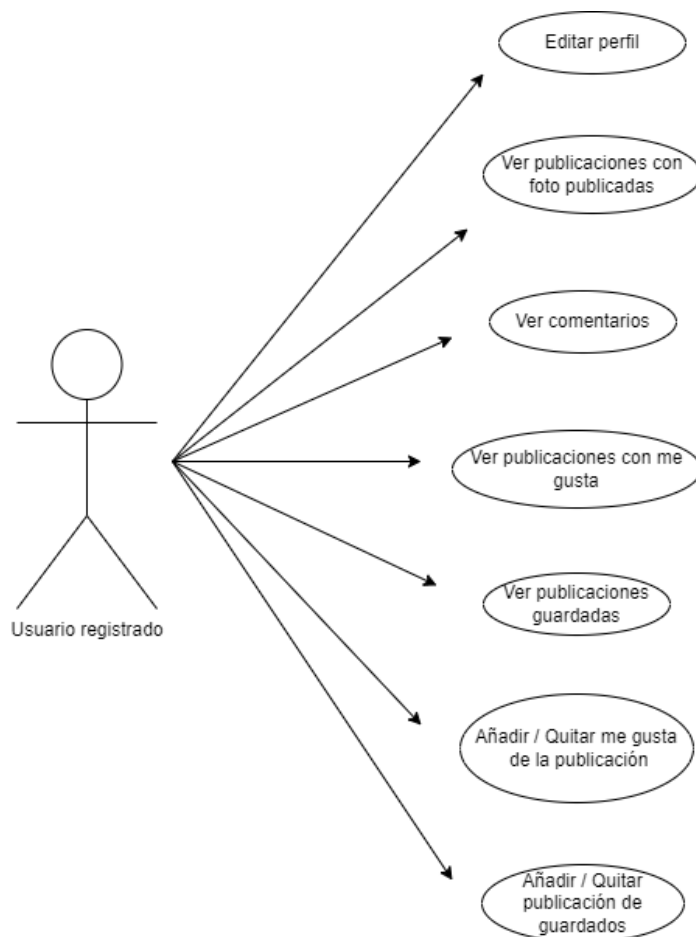
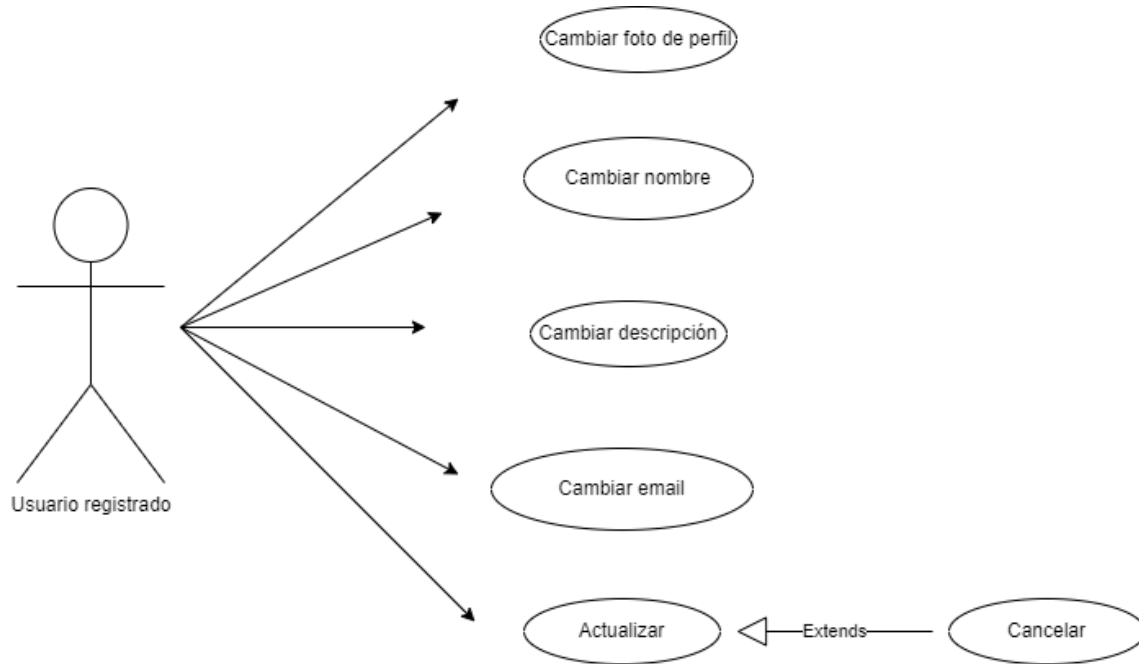


Diagrama de casos de uso de editar perfil



Además, he hecho el diagrama de flujo con todas las actividades que tiene la app y sus posibles decisiones.



Diseño (3.3)

Hacer el diseño de una aplicación es costoso, por eso he decidido dividirlo en 4 partes:

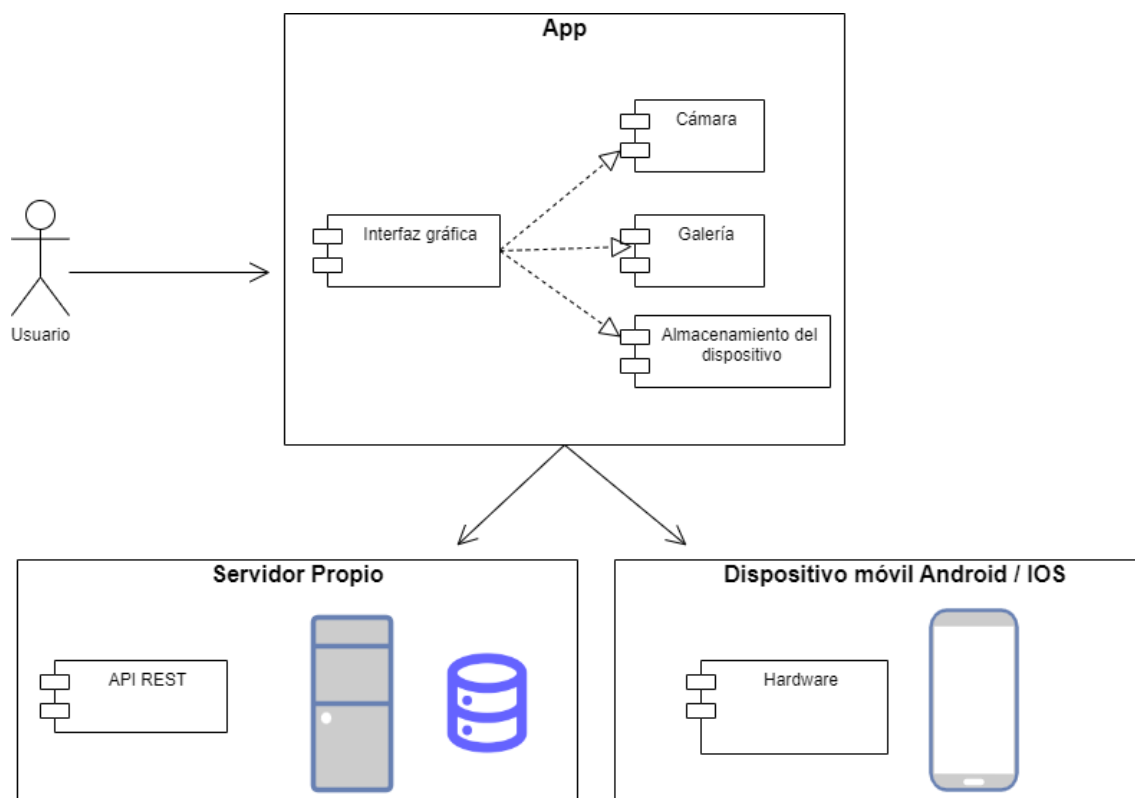
Arquitectura del sistema (3.3.1)

La aplicación de este proyecto, para poder funcionar tendrá que ser capaz de interactuar con diferentes componentes externos a la aplicación. Algunos formarán parte del sistema mientras que otros serán de terceras partes.

El principal componente externo con el que interactuará la aplicación será un servidor propio, que se encargará de recoger y procesar toda la información que el usuario introduzca a la app a través de la API.

Algunas características hardware usadas son el sistema de almacenamiento, la cámara, la galería y el acceso a Internet.

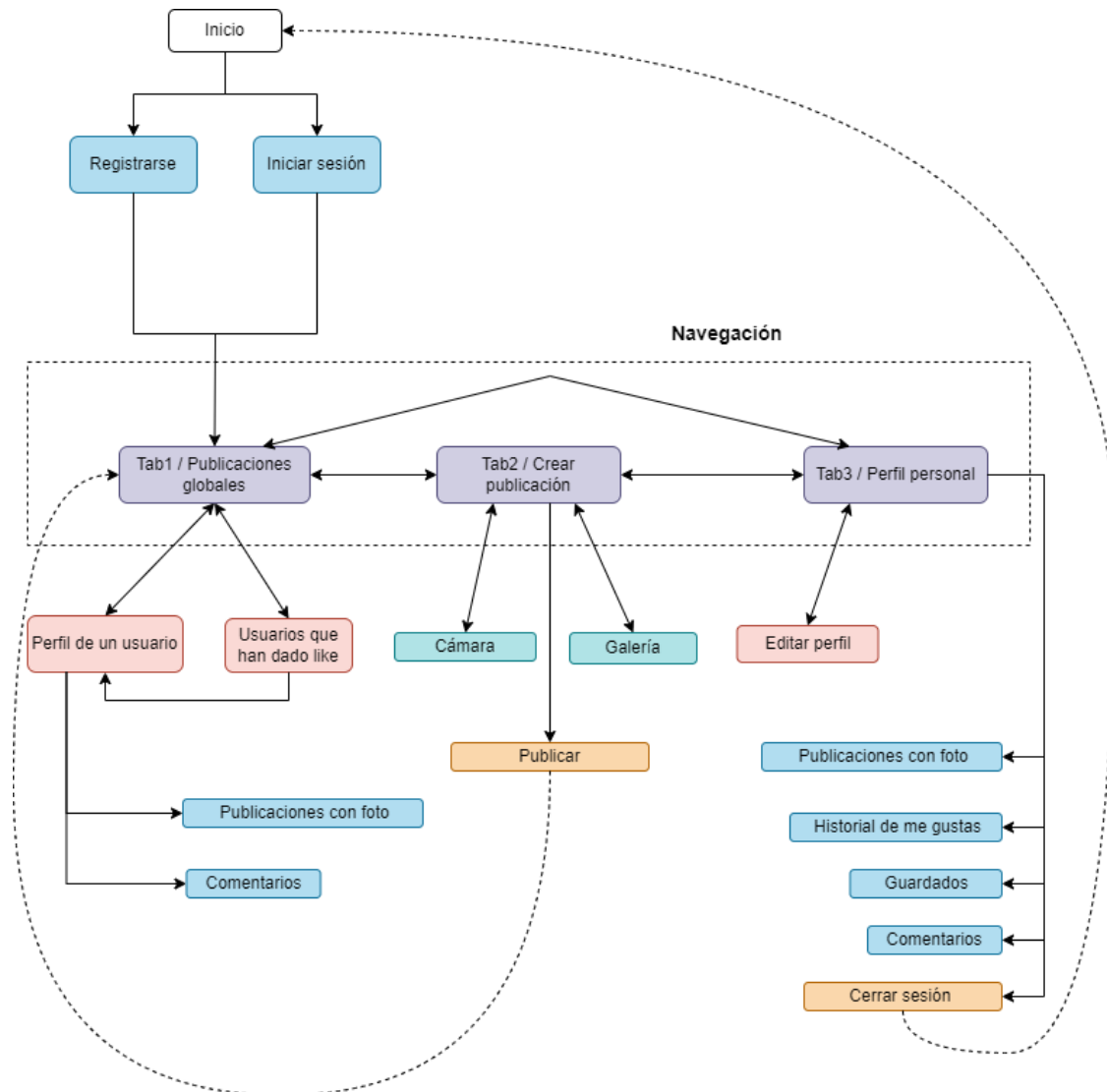
Diagrama de componentes



Capa de presentación (3.3.2)

También he realizado un diagrama de navegabilidad para identificar mejor las pantallas que tiene la aplicación y su comportamiento.

Diagrama de navegabilidad



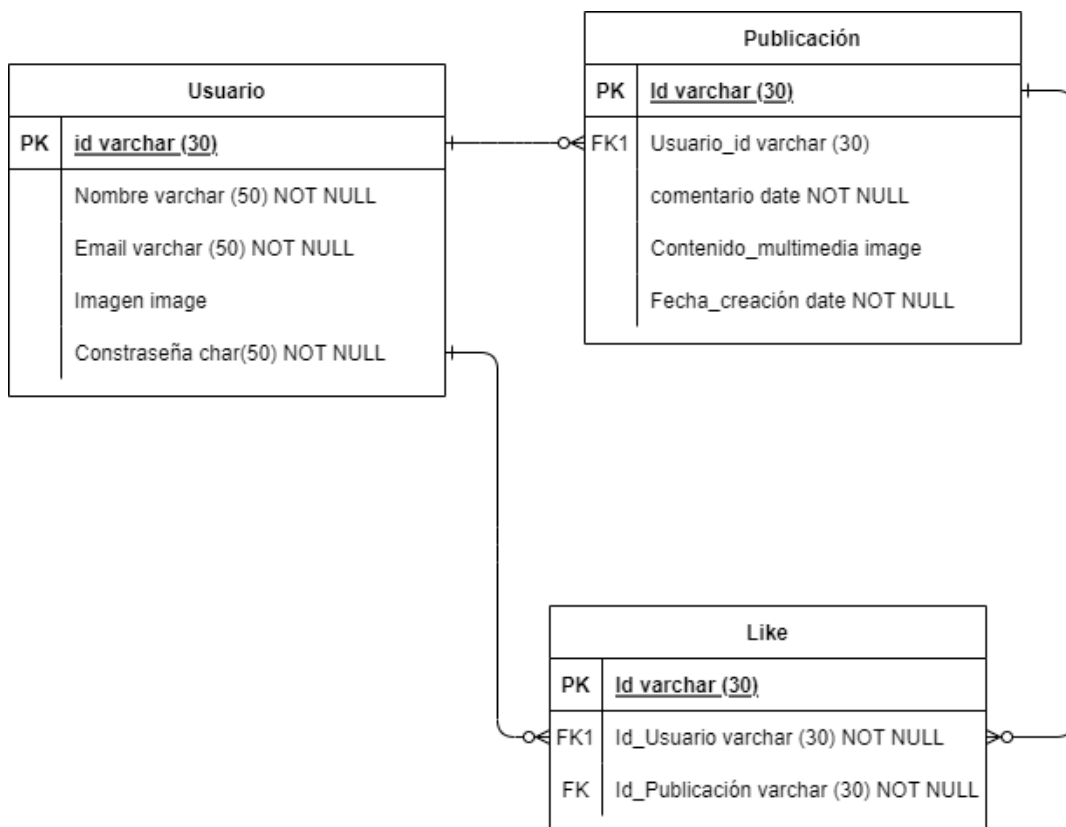
Leyenda

	Ventanas		Ventanas flotantes
	Menú de navegación (Tabs)		Acciones con redirección
	Ventana de una app externa		Redirección

Capa de datos (3.3.3)

He decidido utilizar un sistema gestor de base de datos nosql, por lo que el diagrama de entidad relación es una adaptación a como sería con una base de datos relacional.

Una ventaja de esta app es la poca complejidad que tiene la base de datos porque se basa en 3 tablas.



Interfaz gráfica (3.3.4)

Diseño de la app (3.3.4.1)

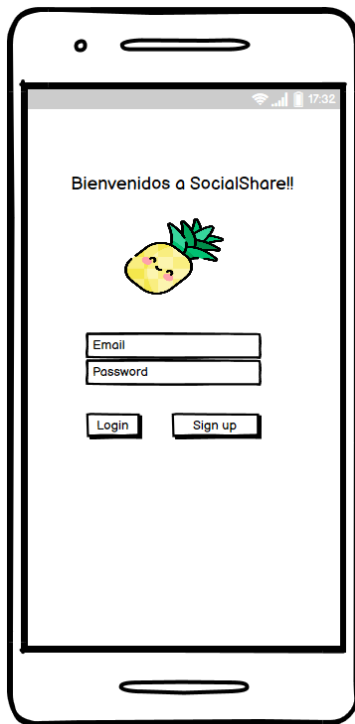
El diseño de la app es otra parte muy importante. Una app con muchas funcionalidades pero un diseño pobre, puede pasar más desapercibida que una app con pocas funcionalidades pero muy vistosa.

La idea inicial de la app era hacer una red social con aspecto minimalista y que inspire sensación de calma. Al final me he decantado por unos colores relacionados con el verano y con tonos pastel.


Diseño previo a la app (3.3.4.2)

El diseño previo está realizado con balsamiq.

Iniciar sesión



Bienvenidos a SocialShare!!

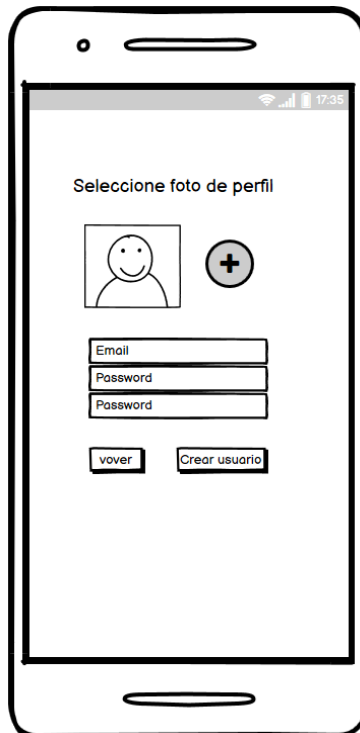


Email

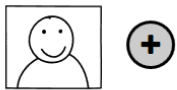
Password

Login Sign up

Registrarse



Seleccione foto de perfil



Email

Password

Password

over Crear usuario

Perfil personal



Heading

Guillermo

...Descripción...

Editar perfil

mis imágenes publicaciones que me gustan

mis comentarios publicaciones guardadas

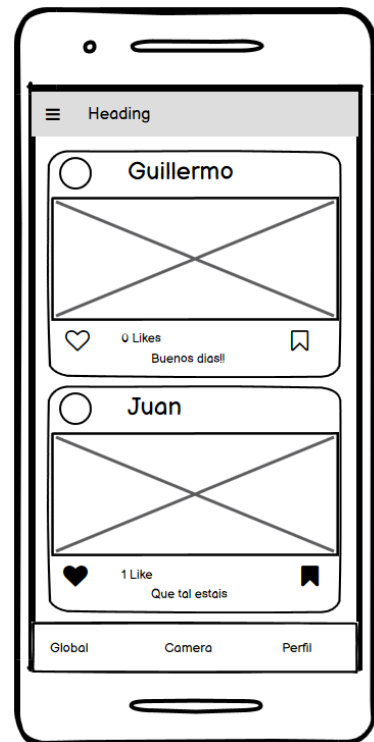
Mis publicaciones

Guillermo

1 Like

Global Camera Perfil

Publicaciones globales



Heading

Guillermo

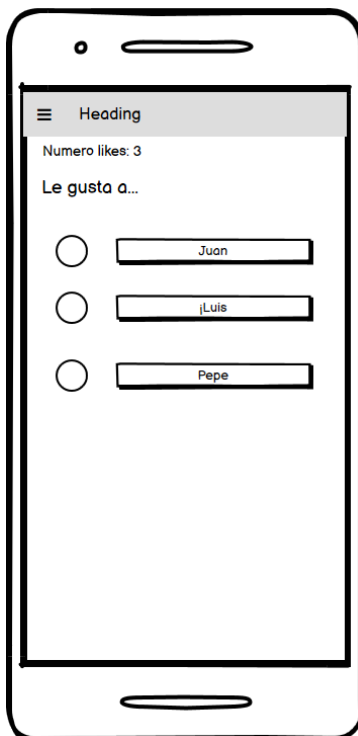
0 Likes Buenos días!!

Juan

1 Like Que tal estais

Global Camera Perfil

Ventana de likes



Heading

Numero likes: 3

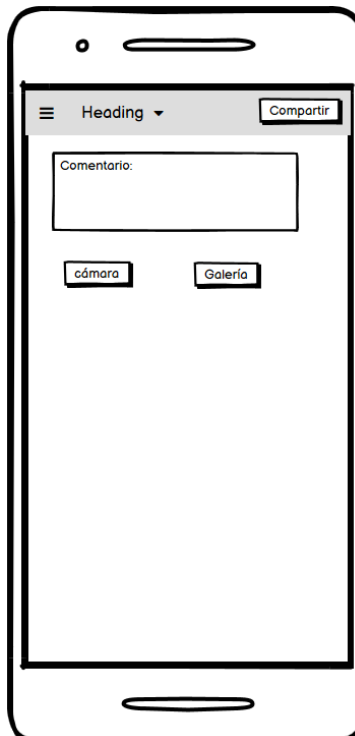
Le gusta a...

☐ Juan

☐ ¡Luis

☐ Pepe

Crear publicación



Heading

Compartir

Comentario:

cámara Galería

Editar perfil



Heading

Editar perfil

Email

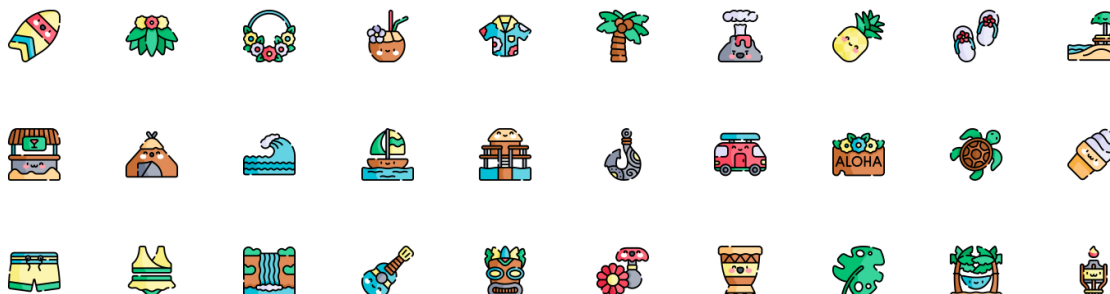
Nombre

Descripción

Actualizar

importación de iconos (3.3.4.3)


En la búsqueda de iconos al final, me decanté por iconos tropicales, ya que transmiten la sensación de calma y verano. Además, he combinado los colores de los iconos, con los de la app.



Aplicar diseño en la app (3.3.4.4)

Iniciar sesión

Bienvenido a SocialShare !!




Email socialsharetest@gmail.com

Password

[LOGIN](#) [SIGN UP](#)

Registrarse

Seleccione foto de perfil:



Email socialsharetest@gmail.com

Nombre guiller


Password

[VOLVER](#) [CREAR USUARIO](#)

Publicaciones globales

SocialShare

guiller nuevo@nuevo.com



1 likes

holaaa

guiller nuevo@nuevo.com

2 likes

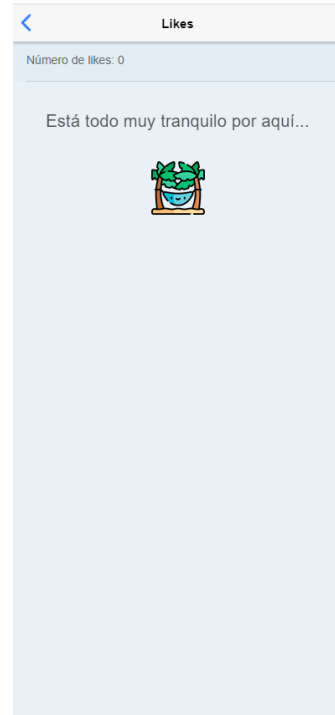
que tal estais hoy? 😊

guiller nuevo@nuevo.com

0 likes

[Home](#) [Camera](#) [Profile](#)

Ventana likes



Editar perfil



Implementación y configuración (3.4)

Una vez presentado el diseño para la aplicación de este proyecto, el siguiente paso es implementar la aplicación a partir de este diseño.

Tecnologías empleadas (3.4.1)

Para desarrollar este proyecto, he usado diferentes tecnologías y herramientas.

Framework:

He decidido escoger Ionic con Angular por sus facilidades en desarrollar una app móvil muy vistosa y con posibilidad de añadirle todas las funcionalidades necesarias a través de sus propias herramientas o plugins.

Este framework permite desarrollar una aplicación móvil a través de lenguajes usados para web.

Además, en la página web de Ionic aparecen los ejemplos del uso de las herramientas de una forma muy visual y así facilita el tiempo de realizar pruebas en la aplicación.

Entorno de trabajo:

Esta aplicación ha sido desarrollada con Visual Studio Code.

Es un entorno gratuito el cual funciona por extensiones y gracias a ello me ha facilitado bastante el desarrollo de la aplicación.

Junto a este entorno estaremos usando GIT con Github para el control de versiones, lo que nos agilizará el trabajo frente a posibles errores o pérdida del propio trabajo.

Lenguajes usados:

El servidor, la API y el back de la app está desarrollado en Typescript.

La interfaz gráfica de la app está desarrollada con HTML, CSS, componentes de Ionic y Angular.

Base de datos:

Está creada en MongoDB con Javascript.

Otros recursos utilizados:

Balsamiq para el diseño de la interfaz.

Draw.io para la creación de los diagramas.

Classdiagram-ts para la creación del diagrama de clases.

Appicon para la creación de los iconos de la app.

Ngrok para el despliegue de la app.

Metodología de trabajo (3.4.2)

Para este trabajo, vamos a seguir una metodología ágil, con la cual dividiré las horas de trabajo en 5 sprints de dos semanas cada uno. En cada uno de estos sprints se realizarán varias tareas. En el primer sprint, como es habitual llevará más tiempo la estructuración del proyecto y la instalación de todas las herramientas necesarias. Finalmente, en el último se dedicará más tiempo en realización de pruebas y solución de posibles bugs.

Planificación (3.4.3)

En este apartado se pretende pautar cómo será la evolución de este trabajo.

A continuación, se proponen unas tareas divididas en 5 sprints.

La idea principal es intentar hacer todas las tareas planeadas, en caso de terminar todas las tareas con antelación se podrán coger tareas del siguiente sprint, en caso contrario, las tareas planeadas que no haya dado tiempo de realizar se añadirán a los sprints siguientes.

Análisis de los sprints a realizar (resumen)

- Sprint 1:
Instalación y preparación del entorno e inicio del desarrollo del servidor.
- Sprint 2:
Terminar el desarrollo de las principales llamadas de la api e instalación de plugins necesarios.
- Sprint 3:
Ampliación de la base de datos y servicios de la api.
Inicio del desarrollo de la app.
- Sprint 4:
Terminar el desarrollo de la aplicación y revisión de bugs.
- Sprint 5:
Desplegar la aplicación y la base de datos a un entorno real.
Revisión de posibles bugs.

Desarrollo de la aplicación (3.4.4)

En esta fase, describiré de una forma más concreta el desarrollo efectuado en los sprints.

Sprint 1

Tiempo estimado: 8h.

Tiempo dedicado: 14h.

Tareas a realizar:

- **Instalación de todos los requisitos para poder desarrollar la app.**
- **Creación de la base de datos en MongoDB.**
- **Comienzo del desarrollo del backend y de los servicios que implementaremos.**
 - Servicio de inicio de sesión.
 - Servicio de registro de usuario.
 - Servicio de actualización de datos del usuario.
 - Configuraciones básicas.

Desarrollo del sprint

Instalación de todos los requisitos para poder desarrollar la app

Para empezar a desarrollar la aplicación, en primer lugar, tuve que definir las herramientas que voy a necesitar.

El entorno de desarrollo que usaré es Visual Studio Code (vsc), por las posibilidades que tiene gracias a las extensiones.

Aparte de las extensiones que tiene vsc, he tenido que instalar varios clientes:

- Ionic.
- Angular.
- Node JS.
- NPM.

Por otro lado, también instalé MongoDB para la base de datos local y para llevar un control de versiones, he usado GIT con Github.

Creación de la base de datos en MongoDB

Para crear la base de datos, se puede crear desde MongoDB Compass de forma visual, pero lo haré directamente desde el código, ya que si no existe la base de datos o las tablas se crearán automáticamente al compilar.

La creación de la base de datos a través del código, está hecha en typescript con ayuda de dos módulos de node:

- Express.
- Mongoose.

Por otro lado, en el index, he añadido las rutas de los archivos que crearán las tablas de la base de datos.

```
//rutas app
server.app.use('/user',userRoutes);
server.app.use('/posts',postRoutes);
server.app.use('/likes',likesRoutes);
```

Muy importante, para conectar el servidor con la base de datos he puesto la conexión que previamente creé al instalar mongodb usando el paquete mongoose.

Comienzo del desarrollo del backend y de los servicios que implementaremos

Antes de ponerme con la creación de los usuarios, crearé la interfaz de los usuarios para definir qué atributos tendrán:

- Nombre nombre de usuario obligatorio.
- Imagen imagen de perfil no obligatorio.
- Email correo del usuario obligatorio.
- Password contraseña del usuario obligatorio. A través de un método comprobaré si la contraseña coincide ya que, con bcrypt se encriptó la contraseña.

La interfaz está creada con typescript y el paquete mongoose.

Servicio de registro de usuario

Este servicio añade al usuario a la base de datos y genera el token que se usará más adelante.

Servicio de actualización de datos del usuario

Este servicio actualiza la información del usuario.

Para ello, primero verifica el token y si es válido actualiza los datos y actualiza el token.

Configuraciones básicas

En las configuraciones básicas entran todas las clases necesarias que he ido creando que hacen tareas pequeñas, imágenes que he necesitado, etc.

- file-system, es la clase que guarda las imágenes de la base de datos.
- assets, guarda la imagen que se pondrá en caso de que el usuario no tenga foto de perfil.
- middleware autenticación, comprueba que el token es correcto o no.

Conclusión

En este sprint al final me ha costado más de lo planeado, ya que he tenido que aprender cómo funciona el paquete mongoose, que es uno de los principales paquetes para configurar el servidor. Además, tuve que aprender como crear las llamadas de get y post con Typescript.

Sprint 2

Tiempo estimado: 4h.

Tiempo dedicado: 3h.

Tareas a realizar:

- **Desarrollar base de datos y API:**
 - o Servicio de creación de publicaciones.
 - o Servicio de obtención de publicaciones con paginación.
 - o Servicio de obtención del token.
 - o Servicio obtención de usuario por token.
 - o Generar contraseñas seguras
 - o Generar nombres únicos para las imágenes.
 - o Instalación de plugins.

Desarrollo del sprint

Servicio de creación de publicaciones

Para la creación de una publicación dividiré el servicio en dos.

Por un lado tendré la importación de archivos multimedia y por el otro, tendré el mensaje que se añade a la publicación.

```
//crear post
> postRoutes.post('/',[verificarToken],(req:any,res:Response)=>{ ...
});

//servicio para subir imagenes y videos
> postRoutes.post('/upload',[verificarToken],async (req:any,res:Response)=>{ ...
});
```

Servicio de obtención de publicaciones con paginación

Crearé un servicio de obtención de publicaciones .

Necesito que devuelva las publicaciones paginadas, para que no colapse la base de datos con llamadas que tengan un tiempo de espera demasiado largo.

Servicio de obtención del token

Para crear el servicio de obtención del token, tendré que importar el plugin *jsonwebtoken*. Con este plugin he creado el token y le he dado un tiempo de caducidad.

Para comprobar que el token sigue siendo válido, el plugin tiene su propia función, la cual usa la semilla que previamente establecí (la semilla es mi firma en el token, así sabré que está creado por mi).

Servicio obtención de usuario por token

Para obtener el usuario a través del token, el primer paso es comprobar si el token sigue siendo válido. Una vez hecho esto, en caso afirmativo, se decodificará el token y devolverá el usuario.

Generar contraseñas seguras

Para generar una contraseña segura, he decidido usar el plugin *bcrypt*.

Sirve para encriptar la contraseña. De esta forma, subiré a la base de datos la contraseña encriptada para mayor seguridad.

Para comprobar en el inicio de sesión si la contraseña es correcta, encriptaré la contraseña que el usuario introduzca y se compararán.

Generar nombres únicos para las imágenes

Para generar nombres únicos decidí usar el plugin *uniqid*.

La forma de uso es muy sencilla, solo tendré que llamar al método *uniqid()* y creará el id.

Una vez tenga el id único, tendré que asignarle el id a la imagen sin perder la extensión que tiene (png, jpg, jpeg).

Conclusión

En este sprint ha sido más fácil desarrollar las llamadas, ya que tenía una base del primer sprint. Lo único nuevo ha sido el paquete *uniqid* y *jsonwebtoken*, que son bastante intuitivos de usar.

Sprint 3

Tiempo estimado: 5h.

Tiempo dedicado: 8h.

Tareas a realizar:

- **Desarrollar base de datos y API:**

- Creación de la tabla de likes.
- Añadir nuevos servicios.
 - Dar like.
 - Dar dislike.
 - Obtención de los likes por usuario.
 - Obtención de número de likes por post.
- **Desarrollar app:**
 - Creación de actividades.
 - Conectar la API.
 - Añadir las funcionalidades entre las actividades.

Desarrollo del sprint

Creación de la tabla de likes

Para la creación de la tabla de likes, tendré que añadir dos archivos:

- Crear el modelo con sus campos (id usuario, id publicación y número de likes).
- Crear archivo con todos los servicios.

Por último tendré que añadir en el index la ruta de likes. De esta forma, al compilar se añadirá automáticamente la tabla a la base de datos.

Dar like / Dar dislike

El uso de estos servicios es muy sencillo, ya que funcionan como un contador tanto para sumar como para restar.

Obtención de los likes por usuario

Con este servicio puedes devolver todos los ids de las publicaciones que haya dado like un usuario. Se complementa con la obtención de publicación a través del id de la publicación (hablaré de él, en el próximo spring).

Obtención de número de likes por post

Este servicio, sirve para tener el recuento de los likes que tiene cada post.

Creación de actividades

En este apartado, se creará la estructura de la app.

La estructura está dividida en 2:

- Login/Registro.
 - Login: Es la actividad que se carga al iniciar la app. En caso de ya haber iniciado sesión entrará directamente a los tabs.
 - Registro: Es donde el usuario se registra por primera vez. Una vez te hayas registrado, te redirige directamente a los tabs.
- Tabs.
 - Tab1 → Global: En esta actividad se cargan las publicaciones de la gente.
 - Perfil amigo: Desde las publicaciones podemos entrar en el perfil de la gente.
 - Ventana likes: Desde los likes de las publicaciones podemos ver quien ha dado like y quienes son.
 - Tab2 → Publicar: En esta actividad podremos añadir comentarios a la publicación, podremos escoger entre sacar una foto o elegir foto o video desde la galería.
 - Tab3 → Perfil: en esta actividad podremos ver el contenido de nuestro perfil. Tenemos la posibilidad de ver nuestras publicaciones, nuestros comentarios, las publicaciones guardadas y las publicaciones que nos han gustado.
 - Editar perfil: Desde el perfil, tenemos un botón de editar, que nos lleva a otra actividad, donde podremos cambiar la foto, el nombre, el email y la descripción del perfil.

Conectar la API

Para conectar la app con la API, he tenido que crear la conexión en los enviroment.

Una vez hecho esto, he creado las diferentes llamadas a los servicios para que se comuniquen con la API.

Añadir las funcionalidades entre las actividades

En esta tarea, se enlazan todas las actividades mencionadas en la creación de actividades.

En la primera actividad (Login/registro) cuando un usuario se registre o inicie sesión, tendrá que redirigir a la actividad con las publicaciones globales y guardar su información para las diversas interacciones en la app.

Dentro de las interacciones, es muy importante la información del usuario que está usando la app, como la información de los otros usuarios. Esta información se necesita para que la API nos deje disfrutar del contenido.

Conclusión

Este sprint ha estado más interesante al ver parte de la interfaz gráfica, ya que aparentemente el final del desarrollo estaba más cerca.

Sprint 4

Tiempo estimado: 12h.

Tiempo dedicado: 8h 30m.

Tareas a realizar:

- **Modificación de la API:**
 - Añadir nuevos servicios.
 - Obtención de atributos a través del id del usuario.
 - Obtención de todas las publicaciones de un usuario.
 - Obtención de publicación a través del id de la publicación.
- **Modificación de la app:**
 - Guardar en el storage las publicaciones guardadas.
 - Resolución de bugs.
 - Comunicar las conexiones creadas.
- **Diseño de la app**

Desarrollo del sprint.

Modificación de la API:

Durante el desarrollo de la app me he dado cuenta, que necesitaba otras llamadas específicas para el funcionamiento correcto de la app y sus características.

Modificación de la app:

Obtención de atributos a través del id del usuario

Se obtienen los atributos del usuario con su id. Esto nos sirve cuando queramos ir al perfil de un usuario a través de una publicación.

Obtención de todas las publicaciones de un usuario

Este servicio, obtiene todas las publicaciones del usuario, para poder cargarlas en nuestro perfil.

Obtención de publicación a través del id de la publicación

Este servicio, se complementa con el de obtención de likes por usuario.

Juntando estos dos servicios, podré devolver las publicaciones que ha dado like un usuario.

Guardar en el storage las publicaciones guardadas

Esta tarea consiste en guardar los enlaces a las publicaciones. De esta forma cuando entremos en la app, en el apartado de publicaciones guardadas, aparecerán y se actualizará la información de la publicación (los likes, la foto de perfil que se muestra, el nombre de usuario.).

Comunicar las conexiones creadas

Consiste en comunicar las llamadas de la API creadas en la app, con las clases que necesitan usarlas.

Resolución de bugs

Esta tarea es muy importante, en ella he tenido que resolver todos los fallos que pueden ocasionar las comunicaciones con la API, errores de refresco de datos, errores de rutas no definidas, etc. Retomaré esta tarea en el próximo spring para una revisión más exhaustiva.

Diseño de la app

En esta tarea, se implementa el diseño que he mostrado en el punto 3.3.4.4.

El diseño está realizado con componentes de Ionic, Angular, HTML y CSS.

Conclusión

Me ha gustado bastante este sprint, ya que el diseño y las funcionalidades estaban terminadas y ya aparentaba un aspecto profesional.

Sprint 5

Tiempo estimado: 8h.

Tiempo dedicado: 20h

Tareas a realizar:

- **Despliegue de la base de datos y api en un hosting.**
- **Despliegue de la app en un dispositivo físico.**
- **Resolución de bugs.**

Desarrollo del sprint

Despliegue de la base de datos y api en un hosting

Tras una investigación e implementación de posibles hostings en los cuales alojar la base de datos y la api, he comparado las ventajas y desventajas que tenían.

Comparando las posibilidades que me ofrecían Firebase, Heroku y MongoDB Atlas, he decidido usar mi propio ordenador como hosting y de esta forma no tener límite de llamadas a la api, ni límite de almacenamiento.

Para usar mi ordenador de hosting he usado ngrok, es un servicio que nos permite crear nuestro servidor local en un subdominio para poder visualizarlo fuera de la LAN, a través de internet.

Una vez desplegado, para un consumo menor de recursos he decidido usar una raspberry Pi. Con esta solución puedo tener un servidor desde casa con un consumo como el de la carga de un móvil y las limitaciones de almacenamiento solo serán las propias del hardware.

Despliegue de la app en un dispositivo físico

Una vez esté desplegado el hosting, simplemente hay que compilar la app usando Ionic capacitor build android o ionic capacitor build IOS.

Tras compilar paso la app al móvil.

Resolución de bugs

Al desplegar la app en un móvil o emulador, la subida de archivos multimedia a las publicaciones y a la foto de perfil, no funcionaban.

He tenido que crear un nuevo método en la parte de la base de datos para que guarde las imágenes y las devuelva a través de un pipe que se añade en la app.

Conclusión

Este es print ha sido el más difícil de todos. Al intentar desplegar la base de datos y la API no conseguí que terminase de funcionar todo correctamente. Subir la base de datos a un hosting era fácil, pero los conflictos eran con la API. Finalmente conseguí que funcionara correctamente y encontré otro problema con las imágenes que posteriormente solucioné.

Pruebas (3.5)

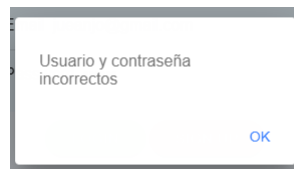
Una vez implementado todo, el siguiente paso es asegurarse de que se cumplen todos los requisitos definidos. Para ello se han definido una serie de pruebas que comprobarán que todo funcione correctamente.

Por otro lado, he añadido una serie de validaciones para gestionar posibles errores o incidencias.

A través de las acciones posibles que se pueden realizar en la app definidas en los diagramas de navegabilidad y de flujo, he ido comprobando las acciones que pueden causar algún problema y estos son los resultados:

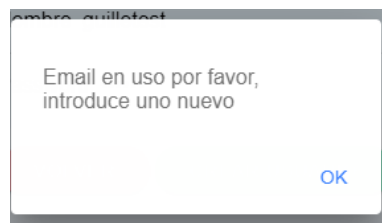
Login

Cuando un usuario introduce un email o contraseña inválidos, la app les mostrará un mensaje.

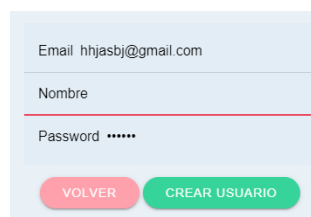


Registro

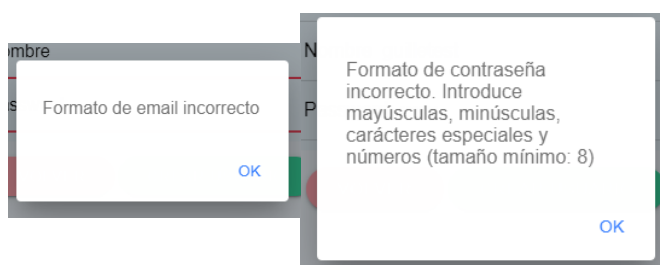
Cuando el usuario intenta registrarse con un email ya existente en la base de datos, la app mostrará un mensaje.



Cuando el usuario intenta crear un usuario sin nombre, la app pondrá un marco en rojo.



Cuando un usuario intenta crear un usuario con un email inválido y una contraseña que no cumple las restricciones.



Editar perfil

Cuando el usuario actualiza la información del perfil, en caso de no actualizarse avisará también.

Se actualizó correctamente

Crear publicación

Cuando el usuario sube una publicación, en caso de error avisará también.

Imagen subida correctamente

Ampliación y posibles mejoras (4)

- Creación de historias.
- Notificación cuando interactúan con tus publicaciones.
- Chat con los usuarios.
- Implementar SendGrid para notificaciones por correo.
- Añadir comentarios a las publicaciones de otras personas.
- Usuario administrador

Creación de historias

Esta idea estaba planteada para añadir una opción desde la creación de publicaciones.

La idea era hacerlo a través de unos slides que permite ionic.

Notificación cuando interactúan con tus publicaciones

Esta idea a mi parecer es necesaria para una mejor experiencia de usuario.

Chat con los usuarios

Esta idea, estaba pensada su realización a través de un complemento que tiene ionic con firebase para enviar información en tiempo real.

Implementar SendGrid para notificaciones por correo

Esta idea se usaría para notificar al usuario cuando se registra y para cambios importantes en la app.

Añadir comentarios a las publicaciones de otras personas

Esta idea, también es necesaria para una mejor experiencia de usuario, ya que así se podrán expresar mejor ante las publicaciones.

Usuario administrador

Esta idea es la más importante, de esta manera habrá un mejor control sobre los usuarios. En estos momentos en caso de querer controlar lo que pasa en la aplicación se puede hacer desde la parte del servidor.

Conclusión (5)

Con la realización de este proyecto, he podido trabajar con distintas tecnologías informáticas que hasta el momento eran desconocidas para mí. Por un lado, he tenido una primera experiencia en programación móvil de forma híbrida, ya que hasta el

momento solamente había realizado programas para un solo sistema operativo. Por otro lado, he podido aprender cómo desarrollar la base de datos y la api desde 0 con Typescript y diferentes paquetes, también he aprendido cómo tratar imágenes y videos a la hora de guardarlos y al mostrarlos.

Por otro lado, me he dado cuenta que esta aplicación híbrida necesitaría un último paso para que funcione en IOS. Se puede compilar la aplicación para IOS desde Windows pero para gestionar los permisos, te pide un último paso, que se hace desde un Mac.

Este proyecto me ha presentado momentos con dificultades, sobre todo con el despliegue y el tratamiento de los archivos multimedia, aun así, tras usar esta tecnología y ver qué resultados he tenido, no será la última aplicación que desarrolle con Ionic por su comodidad.

Bibliografía (6)

Stack overflow

<https://es.stackoverflow.com/>

Draw.io

<https://app.diagrams.net/>

Appicon

<https://appicon.co/>

Classdiagram-ts

<https://marketplace.visualstudio.com/items?itemName=AlexShen.classdiagram-ts>

Ionicframework

<https://ionicframework.com/>

Flaticon

<https://www.flaticon.es/>

Firebase

<https://firebase.google.com/>

Heroku

<https://id.heroku.com/login>

MongoDB

<https://www.mongodb.com/>

Desarrollo del backend

<https://youtu.be/zRo2tvQpus8>

<https://www.youtube.com/watch?v=yC18hkVZ3BM&list=PLCKuOXG0bPi3nKe-CHNQ5jwJ5V4SR77yd>

Enviar imágenes

<https://dev.to/oscar/sending-file-in-a-post-request-22ep>

Despliegue

<https://ngrok.com/>