TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

"*AI Publicity* - Aplicación Híbrida desarrollada en *Flutter* para identificar logotipos en una imagen"

Curso 2018/2019

Alumno/a:

Ayad Mercer Laaouissi Jones

Director/es:

Francisco Gabriel Guil Reyes

Agradecimientos

En primer lugar, me gustaría comenzar este trabajo haciendo mención especial a mi familia, por su gran apoyo. Gracias a ellos he aprendido que con gran esfuerzo se consigue gran recompensa.

Me gustaría también mostrar mi agradecimiento a los profesores de Ingeniería Informática de la Universidad de Almería y, en especial, a mi tutor Francisco Guil, por sus consejos y por ser un gran profesional en su labor académica.

Contenido

1. Introducción 13
   1.1. Motivación 13
   1.2. Objetivos 14
   1.3. Planificación temporal 16
   1.4. Tecnologías utilizadas 17
      1.4.1. Tecnologías 17
      1.4.2. Herramientas 17
   1.5. Estructura del documento 18
2. Estado del arte 19
   2.1. Instagram 19
   2.2. YouTube 20
   2.3. Gmail 21
   2.4. Tripadvisor 23
3. Tecnologías y herramientas 25
   3.1. Herramientas 25
      3.1.1. Android Studio 25
      3.1.2. *Flutter* 26
      3.1.3. Postman 27
      3.1.4. *Visual Studio Code* 28
      3.1.5. *Marvel* 29
      3.1.6. *MySQL Workbench* 31
      3.1.7. *phpMyAdmin* 32
      3.1.8. Docker 33
      3.1.9. React Native 34
      3.1.10. Ionic 35
      3.1.11. Apache Cordova 36
      3.1.12. Cloud Vision 36
   3.2. Tecnologías 37
      3.2.1. *Dart* 37
      3.2.2. MySQL 37
      3.2.3. Node.js 38
      3.2.4. MySQL + Node.js 38
      3.2.5. JSON 40
      3.2.6. JavaScript 40

3.3. Herramienta optada para el desarrollo de aplicación híbrida 41
3.3.1. Ionic vs Flutter 41
3.3.2. *React Native* vs *Flutter* 42
3.3.3. Apache Cordova vs Flutter 43
3.3.4. Crecimiento de Flutter 44
4. AI Publicity 45
4.1. Especificación preliminar 45
4.2. Análisis del sistema 46
4.2.1. Usuarios del sistema 46
4.2.2. Actores del sistema 47
4.2.3. Sistemas externos 48
4.2.4. Funciones del sistema 48
4.3. Casos de uso 62
4.4. Diagrama de clases 64
4.5. Diagrama de la base de datos 66
4.6. Diagramas de secuencias 67
4.6.1. Añadir comentario a publicación 67
4.6.2. Cambiar imagen de perfil 67
4.6.3. Añadir publicación 68
4.6.4. Modificar información de usuario 69
4.6.5. Cambiar contraseña de usuario 69
4.6.6. Cargar contenido de publicación 70
4.6.7. Dar me gusta a una publicación 71
4.6.8. Cargar comentarios de una publicación 71
4.6.9. Cargar listado de ganancias 72
4.6.10. Cargar listado de notificaciones de la empresa 72
4.6.11. Cargar listado de publicaciones de publicidad de la empresa 73
4.6.12. Iniciar sesión 73
4.6.13. Cambiar notificación a visualizado 74
4.6.14. Registro 74
4.6.15. Cargar seguidores 75
4.6.16. Seguir usuario 75
4.6.17. Cargar seguidos 76
4.6.18. Valorar publicación 76
4.7. Aspectos destacados del sistema y la implementación 77
4.7.1. Aspectos de la API *Rest* 77

4.7.2. Aspectos de la base datos 79
5. Resultados 81
5.1. Registro 81
5.2. Iniciar sesión 83
5.3. Página principal y menú lateral 84
5.4. Buscar usuario 85
5.5. Visualizar perfil propio 86
5.6. Cambiar imagen de perfil 87
5.7. Perfil usuario y seguir usuario 88
5.8. Ajustes 90
5.8.1. Cambiar información de usuario 91
5.8.2. Cambiar contraseña de usuario 92
5.8.3. Cambiar cuenta de usuario normal a empresa 95
5.9. Valorar y dar me gusta a una publicación 96
5.10. Añadir publicación 97
5.11. Menú lateral cuenta de empresa 102
5.13. Ganancias y pagos 104
6. Conclusiones y trabajo futuro 109
6.1. Conclusiones 109
6.2. Trabajos futuros 109
Bibliografía 111

Índice de figuras

Figura 1: API Rest con Node.js y MySQL 14
Figura 2: Cronograma 16
Figura 3: Duración en horas para la resolución de este TFG. 17
Figura 4: Imagen corporativa de la empresa Instagram 19
Figura 5: Imagen corporativa de la empresa YouTube 20
Figura 6: Captura de pantalla aplicación YouTube 21
Figura 7: Imagen corporativa de la empresa Google 21
Figura 8: Captura de pantalla de la aplicación Gmail 22
Figura 9: Imagen corporativa de la empresa Tripadvisor 23
Figura 10: Captura de pantalla de negocio en Tripadvisor 24
Figura 11: Imagen corporativa Android Studio 25
Figura 12: Captura de pantalla de Android Studio con el framework Flutter 26
Figura 13: Imagen corporativa Flutter 26
Figura 14: Imagen corporativa Postman 27
Figura 15: Captura de pantalla de la herramienta Postman 27
Figura 16: Imagen corporativa Visual Studio Code 28
Figura 17: Captura de pantalla de la herramienta Visual Studio Code 29
Figura 18: Imagen corporativa Marvel 29
Figura 19: Captura de pantalla de la herramienta Marvel 30
Figura 20: Captura de pantalla de la ventana registro de la aplicación Marvel 30
Figura 21: Imagen corporativa MySQL Workbench 31
Figura 22: Captura de pantalla de la herramienta MySQL Workbench 31
Figura 23: Imagen corporativa phpMyAdmin 32
Figura 24: Captura de pantalla de la herramienta phpMyAdmin 33
Figura 25: Imagen corporativa Docker 33
Figura 26: Captura de pantalla ejecutando el servidor con Docker 34
Figura 27: Imagen corporativa React Native 34
Figura 28: Imagen corporativa Ionic 35
Figura 29: Imagen corporativa Apache Cordova 36
Figura 30: Imagen corporativa Dart 37
Figura 31: Imagen corporativa MySQL 37
Figura 32: Imagen corporativa Node.js 38
Figura 33: Imagen corporativa MySQL + Node.js 38
Figura 34: Captura de pantalla para la conexión de Node.js a la base de datos MySQL 39
Figura 35: Imagen corporativa JSON 40

Figura 36: Imagen corporativa JavaScript 40
Figura 37: Imagen corporativa Flutter e Ionic..... 41
Figura 38: Imagen corporativa Flutter y React Native..... 42
Figura 39: Imagen corporativa Flutter y Apache Cordova..... 43
Figura 40: Crecimiento de Flutter..... 44
Figura 41: Conexión a la base de datos mydb..... 77
Figura 42: Rutas e imagenes de la API Rest 78
Figura 43: Nombre de las imágenes..... 78
Figura 44: Línea de comandos de Windows desplegando MySQL y PhpMyAdmin 79
Figura 45: Fichero docker-compose.yml..... 79
Figura 46: PhpMyAdmin 80
Figura 47: PhpMyAdmin tablas de la base de datos 80
Figura 48: Ventana de registro..... 81
Figura 49: Post a la API Rest del registro de usuario..... 82
Figura 50: Información del usuario en phpMyAdmin (Comprobación de contraseña cifrada) .. 82
Figura 51: Ventana de inicio de sesión 83
Figura 52 y 53: Menú lateral y página principal 84
Figura 54 y 55: Buscar usuario y buscando por nombre de usuario 85
Figura 56 y 57: Perfil propio pestaña publicación y perfil propio pestaña imágenes 86
Figura 58 y 59: Seleccionar imagen e imagen de perfil cambiado..... 87
Figura 60: Paquete image picker..... 87
Figura 61: Programación para seleccionar una imagen..... 88
Figura 62 y 63: Perfil usuario sin seguidores y perfil usuario con seguidor 88
Figura 64 y 65: Listado de seguidores y listado de seguidos 89
Figura 66: Ventana de ajustes 90
Figura 67: Cambiar datos personales de usuario..... 91
Figura 68: Cambiar contraseña de usuario..... 92
Figura 69 y 70: Alerta rellenar todos los campos y alerta mínimo de caracteres..... 93
Figura 71 y 72: Alerta de contraseña incorrecta y alerta de contraseña no coincide 94
Figura 73: Contraseña cambiada correctamente 94
Figura 74: Ventana cambiar a cuenta de empresa..... 95
Figura 75 y 76: Publicación sin valoración y publicación con valoración..... 96
Figura 77 y 78: Ventana añadir publicación y ventana añadir publicación con datos 97
Figura 79: Paquete multi_image_picker..... 99
Figura 80: Paquete geolocator..... 100
Figura 81: Opción eliminar imágenes en la ventana añadir publicación..... 100
Figura 82: Publicación subida correctamente 101

Figura 83: Menú lateral cuenta de empresa..... 102
Figura 84 y 85: Notificaciones y visualización de notificación 103
Figura 86 y 87: Ganancias usuario y pagos de empresa..... 104
Figura 88: Cálculo del dinero por la publicidad..... 104
Figura 89 y 90: Ajustes vista empresa y Añadir ranking empresa 105
Figura 91: Actualización de ranking 107
Figura 92: Calcular ranking de usuario 107

1. Introducción

El presente Trabajo Fin de Grado pretende satisfacer la necesidad de las empresas para anunciar sus productos de una forma excepcional, siendo el usuario el que realice la publicidad de sus productos de una forma innovadora. De esta manera, el producto publicitado se transmitirá entre los usuarios que forman una red social con una mayor rapidez y confianza.

Cuando un usuario sube una foto en su perfil, de forma automática su contenido será analizado con el objetivo de reconocer algún logotipo en la imagen. Si se localiza el logo de una empresa registrada, la aplicación le enviará una notificación a la empresa avisando de este hecho. Se va a permitir la posibilidad de analizar más de un logotipo en una imagen. Como ejemplo, si un usuario sube una foto bebiendo una Coca Cola y llevando una camiseta Adidas, la notificación les llegará a ambas empresas, informándoles de este hecho publicitario (se asume que la empresa y el usuario estén registrados en *AI Publicity*).

La notificación a cada empresa registrada llevará asociada la fecha, identificador de usuario y el grado de importancia del usuario en ese instante de tiempo determinado (*UserRank*), el cual se computará teniendo en cuenta el número de seguidores y sus actividades (esencia de una red social).

Se deberá tener en cuenta, el índice *UserRank* (UR), de manera que cuanto mayor sea el valor, mayor será la recompensa obtenida por cada acto publicitario. Siguiendo con la filosofía de *PageRank* (el sistema de ranking diseñado por *Google*), el rango de UR será de 0 a 10. Para un usuario determinado, su valor UR fluctuará en el tiempo y, por lo tanto, se deberá tener en cuenta el instante en el que se produce el hecho publicitario. Así pues, si un usuario sube la misma foto en distintas fechas, la recompensa podrá será distinta, siempre dependiendo de su índice UR.

Las empresas serán las que decidan el importe base por cada vez que alguien les hace publicidad, siendo modulado según su valor UR. El importe base será la cantidad que se pagaría a un usuario con UR = 10 (valor máximo).

Esta aplicación ha sido pensada y desarrollada como una aplicación móvil híbrida, tanto para *iOS* como para *Android*. Almacenamos los datos del usuario en una base de datos *MySQL* [19], y para el acceso a dicha base de datos se ha desarrollado una *API Rest* [12] implementada con *Node.js* [18].

1.1. Motivación

Uno de los motivos por lo que surge esta idea es, básicamente, para ayudar a las empresas a darse a conocer con mayor facilidad permitiendo un sistema automático y fiable de publicidad y su

consecuente gratificación a los usuarios, que son los que realmente pueden conseguir que las empresas sean conocidas. No obstante, existen muchas aplicaciones que contienen anuncios de empresas que desean conseguir la fama (mediante el uso de los conocidos ads – abreviatura del inglés *advertising*). Ha sido comprobado que estos anuncios son ignorados y pueden llegar a ser molestos para los usuarios [10]. Si imaginamos que un amigo o conocido está utilizando *AI Publicity* y colabora publicitariamente con el negocio, ¿nos atrae la empresa por igual en las dos aplicaciones?

Con esta propuesta, el coste será menor para las empresas en publicidad y llegará a los usuarios con mayor facilidad.

1.2. Objetivos

El presente Trabajo Fin de Grado tiene como objetivo la implementación de una red social para aplicación móvil, tanto para *iOS* como para *Android*, que será desarrollada en el *framework Flutter* [4], que permitirá a las empresas llevar un seguimiento de las personas que hacen publicidad de sus productos. De este modo, las empresas pueden ponerse en contacto con la persona que patrocina sus productos y gratificar mediante recompensa.

Se ha utilizado el modelo cliente servidor [17] para el desarrollo de la aplicación. Por un lado, en la parte cliente (*front-end*) [16], se encuentra la aplicación móvil desarrollada en *Flutter* [4], la cual emitirá peticiones *http* a la *API Rest* [12], situada en el lado servidor. Por otro lado, en la parte servidor (*back-end*) [16] se implementa una API utilizando los servicios *Rest* [12]. Dicho servidor se encarga de dar la respuesta demandada por el cliente.

Figura 1: API Rest con Node.js y MySQL

Para la elaboración de la aplicación, se han cumplido los siguientes objetivos:

- Creación de una aplicación híbrida cliente (*front-end*) [16] utilizando el *framework Flutter*. Tenemos las siguientes funcionalidades de la aplicación:
    1. Registro.
    2. Iniciar Sesión.
    3. Añadir publicaciones.
    4. Visualizar publicaciones.
    5. Análisis de logotipos en una imagen.
    6. Comentar publicación.
    7. Valorar publicación.
    8. Dar me gusta a una publicación.
    9. Buscar usuarios.
    10. Seguir usuarios.
    11. Visualizar ganancias (Usuario).
    12. Visualizar pagos (Empresa).
- Creación de una aplicación para el servidor (*back-end*) con el fin de realizar la comunicación entre la app cliente y el servidor.
    1. Gestión de usuarios.
    2. Gestión de empresas.
    3. Gestión de monetización.
    4. Gestión de comentarios.
    5. Gestión de publicaciones.
    6. Gestión de seguidores.
    7. Gestión de seguidos.
    8. Gestión de valoraciones en una publicación.
    9. Gestión de me gusta en una publicación.
    10. Gestión de notificaciones.
    11. Gestión de logotipos encontrados en una imagen.

1.3. Planificación temporal

El siguiente cronograma indica el periodo de duración de este Trabajo Fin de Grado.

Etapas del proyecto																
Semanas	1ª	2ª	3ª	4ª	1ª	2ª	3ª	4ª	1ª	2ª	3ª	4ª	1ª	2ª	3ª	4ª
Cronograma TFG																
1. Creación de los prototipos.																
2. Instalación y configuración de software																
3. Aprendizaje de *Dart*.																
4. Aprendizaje de *Flutter*																
5. Desarrollo de aplicación híbrida con *Flutter*.																
6. Desarrollo de aplicación servidor.																
7. Fases de pruebas.																
8. Elaboración de la memoria del TFG.																

Figura 2: Cronograma

El desarrollo de este proyecto ha sido planificado en ocho fases principales:

1. La primera fase consiste en la creación de los prototipos. De este modo, podemos descubrir errores y posibles mejores. El prototipado es una fase importante, puesto que los beneficios son realmente insuperables. Los prototipos de las interfaces de usuario han sido desarrollados con *Marvel* [1].
2. La segunda fase consiste en la preparación del entorno de desarrollo. Una vez seleccionada las plataformas de interés para el desarrollo de la aplicación (aplicación híbrida para *Android* e *iOS*), se comenzará a instalar y configurar todo el software necesario para el desarrollo de la aplicación.
3. La tercera fase consiste en el aprendizaje del lenguaje *Dart* [3].
4. La cuarta fase consiste en el aprendizaje del *framework Flutter* [4].
5. La quinta fase consiste en el desarrollo de la aplicación en el lado cliente (*front-end*) [16]. En dicha fase creamos la interfaz de usuario con el *framework Flut*ter.
6. La sexta fase consiste en el desarrollo de la aplicación en el lado servidor (*back-end*) [16], la cual crearemos las tablas de la base de datos en *MySQL* [19] y la *API R*est [12] en *Node.js* [18].

7. En esta fase se realiza la comprobación del correcto funcionamiento de la aplicación en busca de posibles errores.
8. La última fase es la elaboración de la presente memoria.

18	horas/semana
16	Semanas
300	horas desarrollo CTFG Y TFG
18	horas preparación y realización de la defensa

Figura 3: Duración en horas para la resolución de este TFG.

1.4. Tecnologías utilizadas

Se han utilizado las siguientes tecnologías y herramientas para el desarrollo de este Trabajo Fin de Grado:

1.4.1. Tecnologías

- *Dart* [3]: Es un lenguaje de programación de código abierto y, además, es el lenguaje de programación utilizado en el *framework Flutter [4]* desarrollado por *Google*.
- *MySQL* [19]: Es un sistema de gestión de base de datos relacional y está considerada como una de las bases de datos más popular.
- *Node*.js [18]: Es un entorno de ejecución multiplataforma escrito en *JavaScript* para desarrollar aplicaciones web del lado del servidor.
- *JSON* [20]: Es un formato de texto ligero para el intercambio de datos, es sencillo de leer e interpretar para humanos, mientras que, para las máquinas es simple interpretarlo y generarlo.
- *Javascript* [24]: JavaScript es un lenguaje de programación interpretado, ECMAScript. Se define como un lenguaje de programación orientado a objetos de alto nivel.

1.4.2. Herramientas

- *Android Studio* [6]: Es el entorno de desarrollo oficial para la plataforma *Android*. Instalando el SDK de *Flutter* podemos utilizar *Android Studio* para el desarrollo de nuestra aplicación con el *framework Flutter*.
- *Flutter* [4]: *Flutter* es el kit de herramientas de UI de *Google* para el desarrollo de aplicaciones, compiladas nativamente para móvil, web y escritorio desde una única base de código.
- *Postman* [7]: Es una herramienta para comprobar el funcionamiento de nuestras API *Rest* permitiendo peticiones de una manera sencilla y rápida.

- *Visual Studio Code* [8]: Es un editor de código desarrollado por *Microsoft*. Es compatible con distintos lenguajes de programación.
- Dispositivos móviles: Herramienta necesaria para la comprobación de nuestra aplicación. Se ha utilizado emuladores de dispositivos móviles de *Android S*tudio.
- *Marvel* [9]: Aplicación web para la creación de prototipos.
- *MySQL W*orkbench [22]: Es una herramienta visual para el diseño de base de datos en *MySQL*.
- *Docker* [21]: Es un proyecto de código abierto que despliega aplicaciones dentro contenedores software.
- *React Native* [25] es un *framework* que permite desarrollar aplicaciones móviles híbridas para iOS y para Android.
- Ionic [26] es un framework para el desarrollo de aplicaciones móviles híbridas que utiliza HTML5, CSS y Apache Cordova como base.

1.5. Estructura del documento

La estructura del presente Trabajo Fin de Grado se detalla a continuación:

- Capítulo 1: Introducción. En este capítulo se introduce el Trabajo Fin de Grado describiendo los objetivos, motivación, planificación temporal, así como las tecnologías y herramientas utilizadas para el desarrollo de este.
- Capítulo 2: Estado del arte. En este capítulo se describen brevemente aquellas aplicaciones las cuales tienen cierta similitud y han sido de gran ayuda para el desarrollo de *AI Publicity*.
- Capítulo 3: Tecnologías y herramientas. Se enumera con detalle todas las herramientas y tecnologías utilizadas para el desarrollo de *AI Publicity*.
- Capítulo 4: *AI Publicty*. En dicho capítulo se explica de forma detallada cada uno de los procesos seguidos en cada fase para conseguir el desarrollo de la aplicación.
- Capítulo 5: Resultados. En este capítulo se mostrarán las distintas funcionalidades de la aplicación con la explicación de su funcionamiento.
- Capítulo 6: Conclusiones y trabajo futuro. En este capítulo se citará lo aprendido al realizar este proyecto, así como las posibles mejores y ampliaciones de este proyecto.
- Bibliografía: Se enumeran las referencias y fuentes consultadas para la elaboración de este Trabajo Fin de Grado.

2. Estado del arte

En este capítulo vamos a ver algunas de las aplicaciones relacionadas con *AI Publicty*, así como algunas aplicaciones que han sido de gran ayuda para escoger las mejores funcionalidades posibles.

2.1. Instagram

Figura 4: Imagen corporativa de la empresa Instagram

Instagram es una palabra formada por dos palabras: (1) *-insta*, que hace referencia a su momento instantáneo; y (2) *-gram*, refiriéndose a telegrama, que antiguamente era una manera de compartir mensajes. Fue creada en 2010 y es la primera red social en incorporar filtros fotográficos tan avanzados para la subida de imágenes. Es una red social que solo es utilizada por dispositivos móviles. Actualmente, *Instagram* fue comprado por *Facebook*, ya que ha tenido mucho éxito y ha tenido un gran crecimiento de usuarios.

Instagram es una red social que trata de subir imágenes y compartir imágenes o videos dándoles un retoque profesional y compartir dichos medios con tus seguidores, o incluso compartirlas con otras redes como, por ejemplo, *Facebook*, *Pinterest*, *WhatsApp*, etc. Esta red social es muy conocida y muchos usuarios la utilizan para acercarse a sus marcas favoritas o a otras personas de interés.

Al ser una aplicación famosa y muy bien valorada por los usuarios, decidí seguir el estilo de diseño de esta aplicación porque muchas marcas han tenido éxito por la gran oportunidad que

ofrece la herramienta para poder comunicarse con muchos usuarios. El diseño de la aplicación me parece una buena guía para conseguir buenos resultados en *AI Publicity*.

Los componentes que han servido de guía son: Publicaciones, Me gusta y Comentarios.

2.2. YouTube

Figura 5: Imagen corporativa de la empresa YouTube

YouTube es una plataforma digital muy popular dedicada a compartir videos. Además, también se puede emitir videos en directo, o grabar para compartir dichos videos posteriormente. *YouTube* se puede considerar una red social audiovisual, puesto que permite que otros usuarios se suscriban a tu canal, o que comenten el contenido, con el fin de obtener seguidores y suscriptores.

YouTube fue creado en 2005 con la idea inicial de poder compartir videos. Esta plataforma cuenta con muchas ventajas, entre ellas:

- Mostrar los trabajos audiovisuales creados.
- Celebrar un congreso online.
- Incrementar la visibilidad online de negocios.
- Suscribirte al canal de una persona o empresa.
- Comentar o dar "*like*" al contenido de otros usuarios.
- Emplear videos en la plataforma web.
- Emitir en directo.
- Monetizar un proyecto digital.
- Elegir la privacidad de un contenido.

Al inspeccionar y trabajar anteriormente con dicha plataforma, he podido comprobar que es muy útil y de gran ayuda. En esta aplicación encontré un componente interesante para incluir en *AI Publicity*: la barra de pestaña inferior.

Figura 6: Captura de pantalla aplicación YouTube

2.3. Gmail

Figura 7: Imagen corporativa de la empresa Google

Gmail es un servicio gratuito de correo electrónico patrocinado por Google. Tiene una capacidad de almacenaje de mensajes de 15 GB compartida con *Google Drive* y *Google Fotos*. Tiene integrado un servicio de chat y cuenta con muchas herramientas útiles para antivirus y de filtrado spam, asegurando la cuenta.

Gmail cuenta con múltiples servicios, como Google Mapas o el que nos permite crear documentos en línea para compartir con otros usuarios.

Como se ha mencionado anteriormente, *Gmail* dispone de *Drive*, es decir, el disco duro virtual de *Google*. Se trata de una herramienta que dispone de mucha capacidad de almacenamiento de todo lo que se necesite almacenar.

Hay que añadir que *Gmail* no solo sirve para aplicaciones o servicios de *Google*, sino que se puede utilizar como una dirección de contacto para registrarse en redes sociales y en diferentes aplicaciones.

La aplicación *Gmail* es una aplicación muy sencilla e intuitiva. Después de manejar los servicios y su contenido, se ha podido observar que será muy favorable utilizar el menú lateral de esta aplicación muy intuitivo para cambiar entre las ventanas de la aplicación. Por ello, opté por añadir este menú a la aplicación: por su facilidad en el manejo de cambio de ventanas y por su claridad.

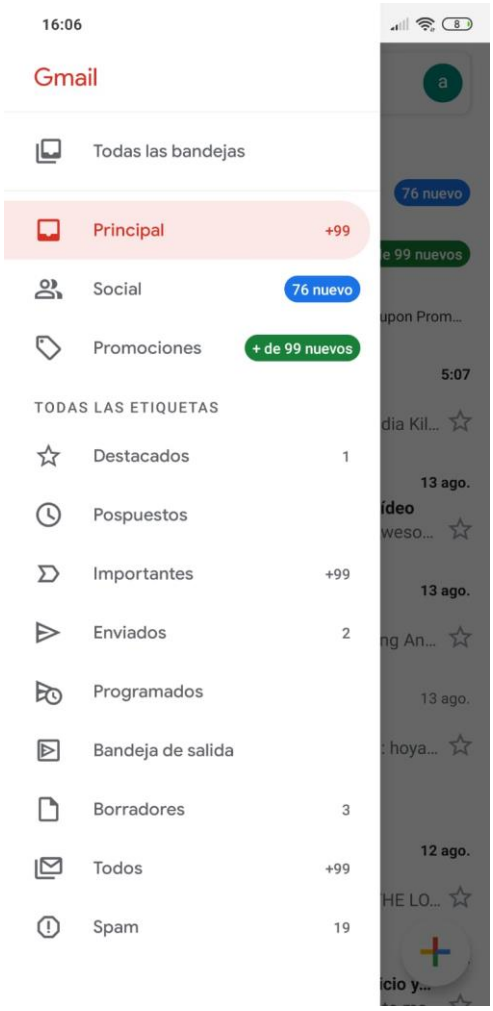


Figura 8: Captura de pantalla de la aplicación Gmail

2.4. Tripadvisor

Figura 9: Imagen corporativa de la empresa Tripadvisor

Tripadvisor es una plataforma turística que se creó en el año 1999 por la escasa información que se ofrecía de cada lugar. Comenzó siendo un lugar para consultar las opiniones de los viajeros y, actualmente, se ha convertido en un comprador de la industria hostelera. *Tripavisor* depende de una serie de algoritmos que se encargan de la posición de un alojamiento o de un restaurante. Estos algoritmos se basan en mencionar la calidad, la cantidad y la antigüedad de las reseñas.

Hay que añadir que, *Tripadvisor* creó la opción de que los consumidores pudiesen realizar reservas dentro de la plataforma, con el objetivo de mejorar la experiencia del usuario. Por ello, muchas empresas se anuncian a través de esta aplicación para obtener mayor éxito y obtener una buena posición dentro de ella.

Tripadvisor se centra en restaurantes, hoteles, vuelos, eventos y alquileres vacacionales. Esta aplicación permite visualizar los distintos negocios o lugares con las valoraciones de los usuarios. La valoración se mide de 1 a 5, mostrando la media de la valoración total. Esta funcionalidad ha sido añadida en *AI Publicity* para que los usuarios puedan votar y ver las votaciones de las publicaciones. De esta manera, los usuarios si desconocen el lugar o producto pueden saber la opinión de otros usuarios.

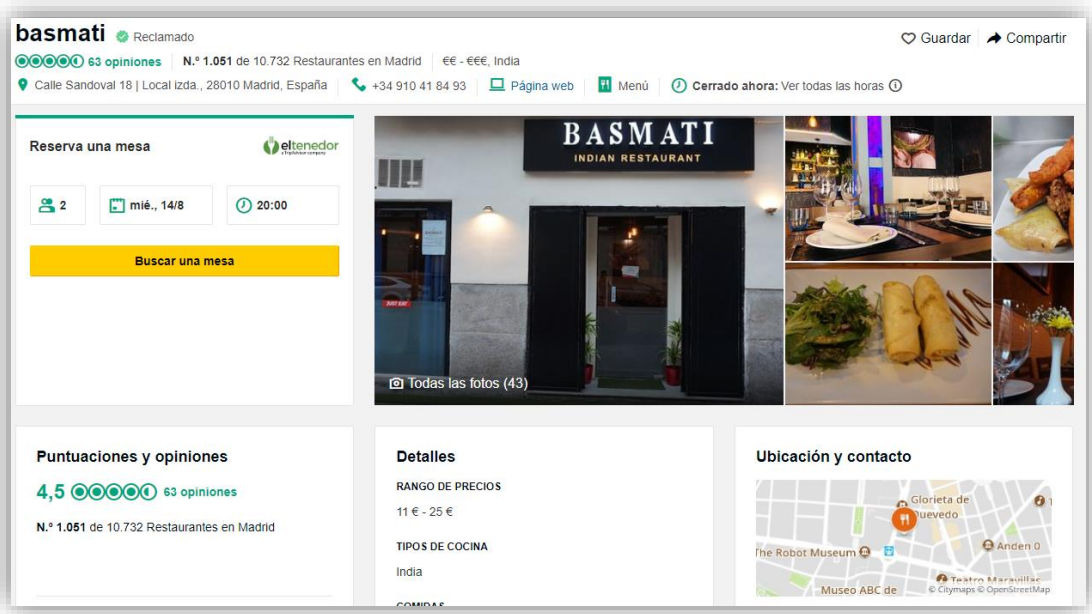


Figura 10: Captura de pantalla de negocio en Tripadvisor

3. Tecnologías y herramientas

En este capítulo se explica todas las tecnologías y herramientas que se han utilizado para el desarrollo de *AI Publicity*.

3.1. Herramientas

3.1.1. Android Studio

Figura 11: Imagen corporativa Android Studio

Android Studio es el entorno de desarrollo oficial para la plataforma *Android* fomentado por *Google*. Está escrito en Java y es multiplataforma. Está disponible para *Windows*, *Mac* y *Linux*. Dispone de varias características, entre ellas, un dispositivo virtual de *Android* que es utilizado para ejecutar y probar aplicaciones, renderizado en tiempo real, herramientas para detectar problemas de rendimiento, usabilidad, compatibilidad, etc.

Android Studio ha sido utilizado para el desarrollo de la aplicación móvil. Se ha instalado el SDK de *Flutter* [4] en *Android Studio* para poder desarrollar la aplicación con este framework. Una vez instalado, se puede crear un nuevo proyecto con *Flutter* [4] y empezar a trabajar en nuevos proyectos.

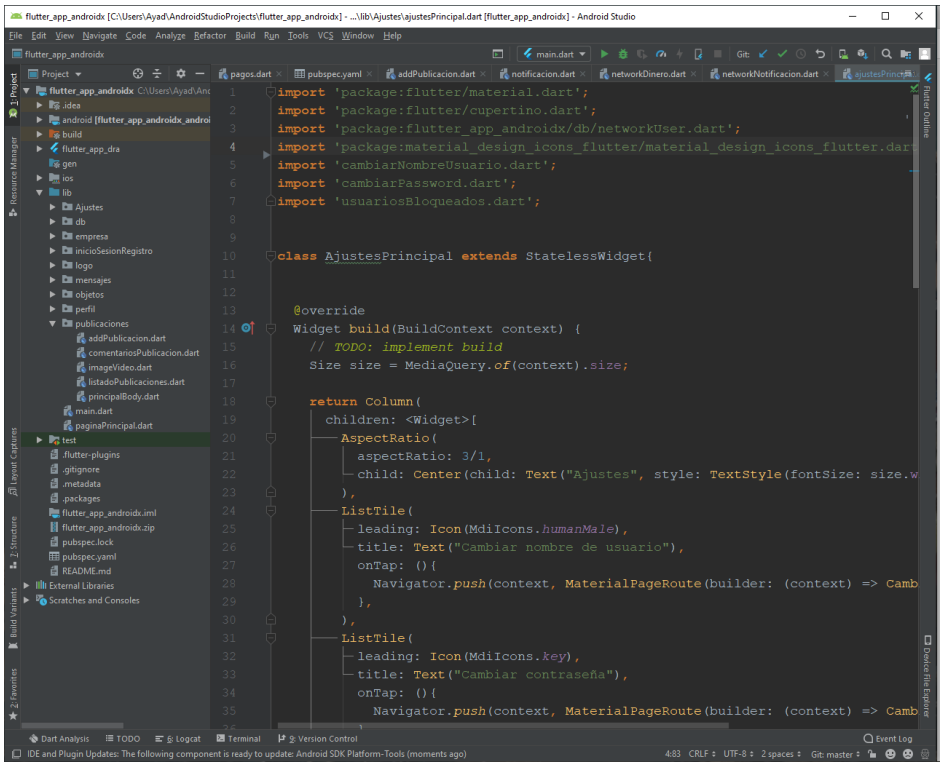


Figura 12: Captura de pantalla de Android Studio con el framework Flutter

Esta herramienta ha sido utilizada para la parte cliente (*front-end*) [16].

3.1.2. *Flutter*

Figura 13: Imagen corporativa Flutter

Flutter [4] es el kit de herramientas de UI de *Google* para el desarrollo de aplicaciones compiladas nativamente para móvil, web y escritorio desde una única base de código. Es un SDK de código abierto escrito en el lenguaje de programación *Dart*, desarrollado por *Google*. Para diseñar una aplicación en *Flutter* [4] es importante conocer sus *widgets*. Los *widgets* son, básicamente, los componentes de diseño que se utilizarán para crear la interfaz de usuario en este framework. En *Flutter* [4] se puede utilizar diseño tanto de *Android* como de *iOS*.

Para empezar a utilizar esta herramienta se ha instalado en *Android Studio* el plugin de *Flutter* [4] y *Dart* [3].

3.1.3. Postman

Figura 14: Imagen corporativa Postman

Esta herramienta tiene la funcionalidad de realizar pruebas a una API *Rest* y verificar el correcto funcionamiento de la API. Permite realizar distintas peticiones a una API *Rest* como GET, POST, DELETE, PUT, etc.

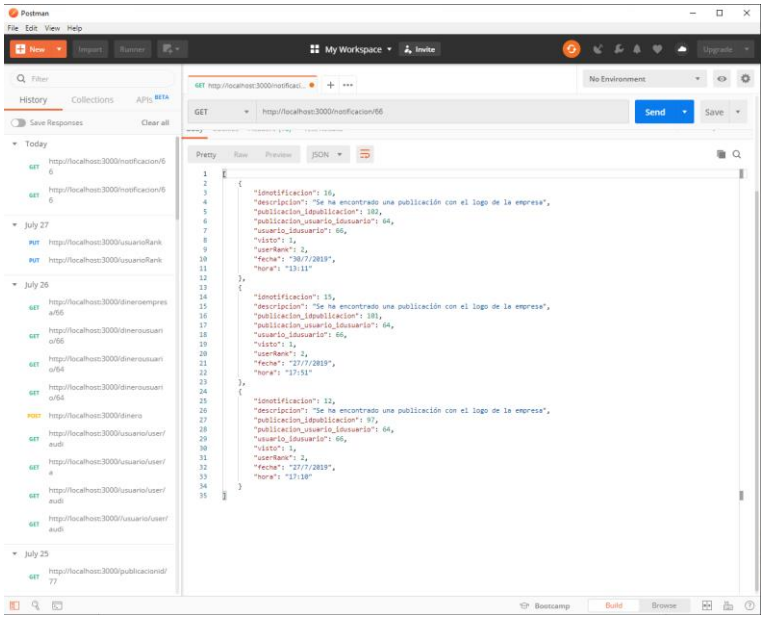


Figura 15: Captura de pantalla de la herramienta Postman

Esta herramienta ha sido de gran utilidad para probar el correcto funcionamiento de la API Rest, en el desarrollo de la parte servidor.

3.1.4. Visual Studio Code

Figura 16: Imagen corporativa Visual Studio Code

Visual Studio Code es un editor de código desarrollado por *Microsoft* y es compatible con distintos lenguajes de programación. Este editor de texto contiene varias herramientas potentes de desarrollo. Además, es gratuito y de código abierto.

Todas las características de *Visual Studio Code* no están expuestas en los menús o interfaz. Se puede acceder a otras características a través de la paleta de comandos o a través de archivos JSON.

Este editor se ha utilizado en el proyecto para el lado servidor (back-end) [16]. Ha encajado perfectamente en esta parte debido a su compatibilidad con distintos lenguajes de programación. Con un solo editor de texto se ha programado en *JavaScript* y *Docker* [21].

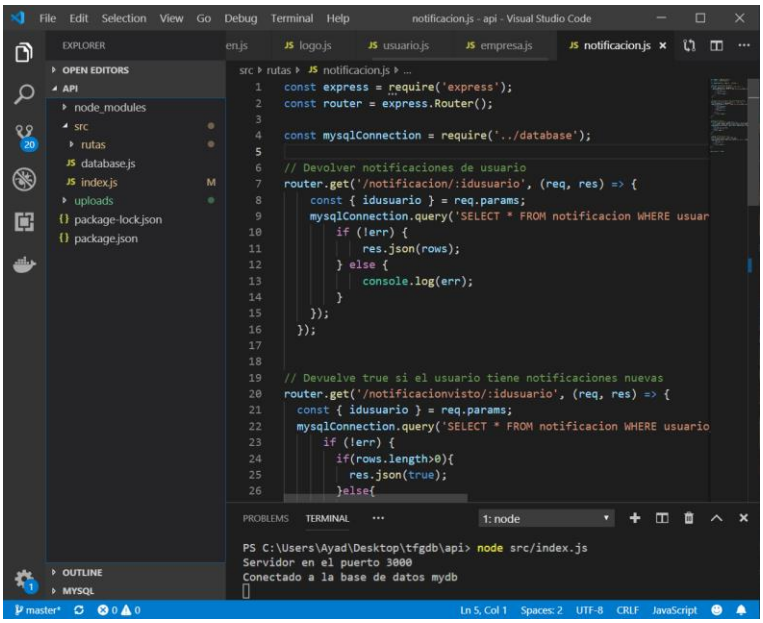


Figura 17: Captura de pantalla de la herramienta Visual Studio Code

3.1.5. *Marvel*

Figura 18: Imagen corporativa Marvel

Marvel es una aplicación web para la creación de prototipos. En el presente Trabajo Fin de grado ha sido muy útil esta herramienta, ya que se ha utilizado para la creación del diseño de la interfaz de la aplicación. De esta manera, al tener el diseño finalizado se he agilizado bastante el trabajo.

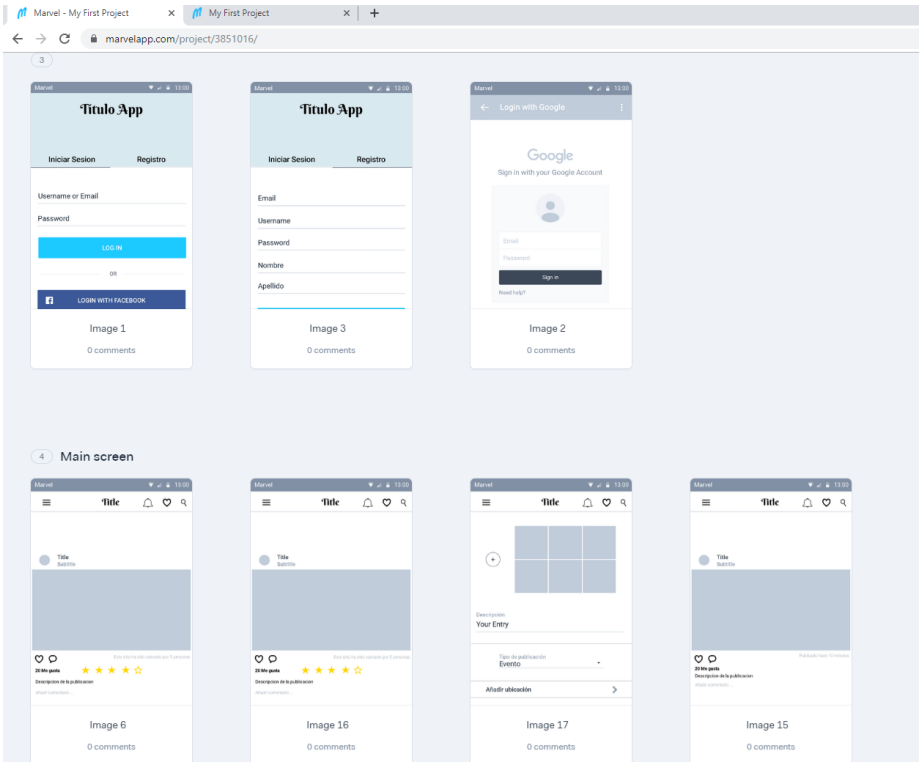


Figura 19: Captura de pantalla de la herramienta Marvel

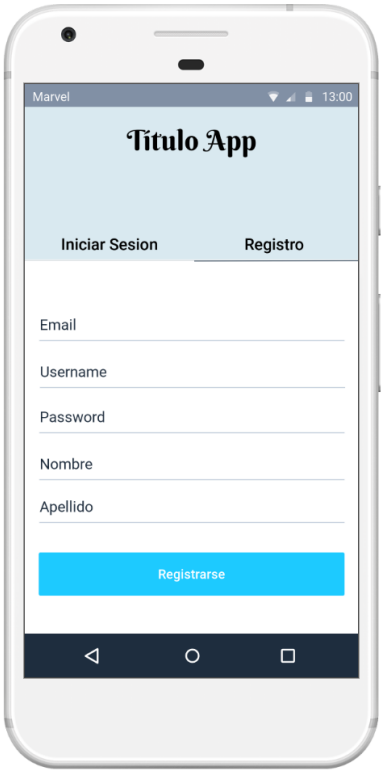


Figura 20: Captura de pantalla de la ventana registro de la aplicación Marvel

3.1.6. MySQL Workbench

Figura 21: Imagen corporativa MySQL Workbench

MySQL Workbench es una herramienta visual utilizada para diseñar la base de datos en *MySQL*. Con esta herramienta se ha conseguido un buen resultado a la hora de crear la base de datos, debido a su facilidad en la visualización de las tablas y relaciones entre ellas.

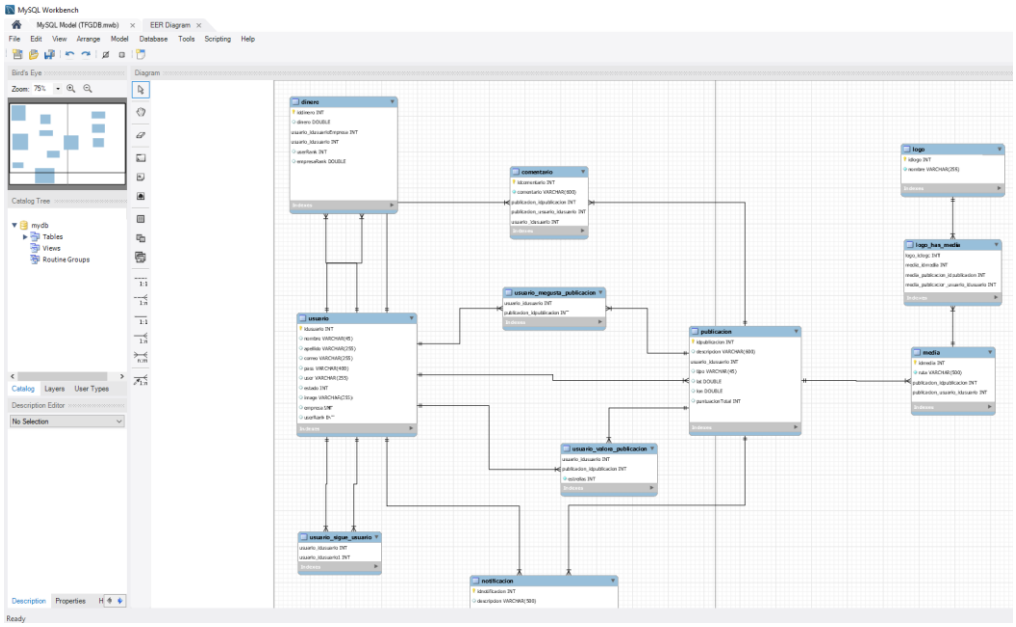


Figura 22: Captura de pantalla de la herramienta MySQL Workbench

En la Figura 22 se puede visualizar todas las tablas que componen la base de datos y sus relaciones.

3.1.7. *phpMyAdmin*

Figura 23: Imagen corporativa phpMyAdmin

PhpMyAdmin es una herramienta escrita en PHP con el objetivo de manejar la administración de MySQL [19] a través de páginas web. Con esta herramienta se puede crear tablas, añadir contenido a las tablas, borrar contenido de las tablas, editar, etc.

Se encuentra vigente desde 1998, siendo muy bien evaluado por sus usuarios. Esta herramienta puede correr en servidores web y soporte de PHP y MySQL.

La herramienta incluye las siguientes especificaciones:

- Interfaz web para la gestión gráfica.
- Exporta datos a varios formatos.
- Permite crear consultas complejas.
- Búsqueda global.
- Permite administrar datos de múltiples servidores.
- Puede importar datos desde archivos CSV y SQL.
- Maneja base de datos MySQL, MariaDB y Drizzle.

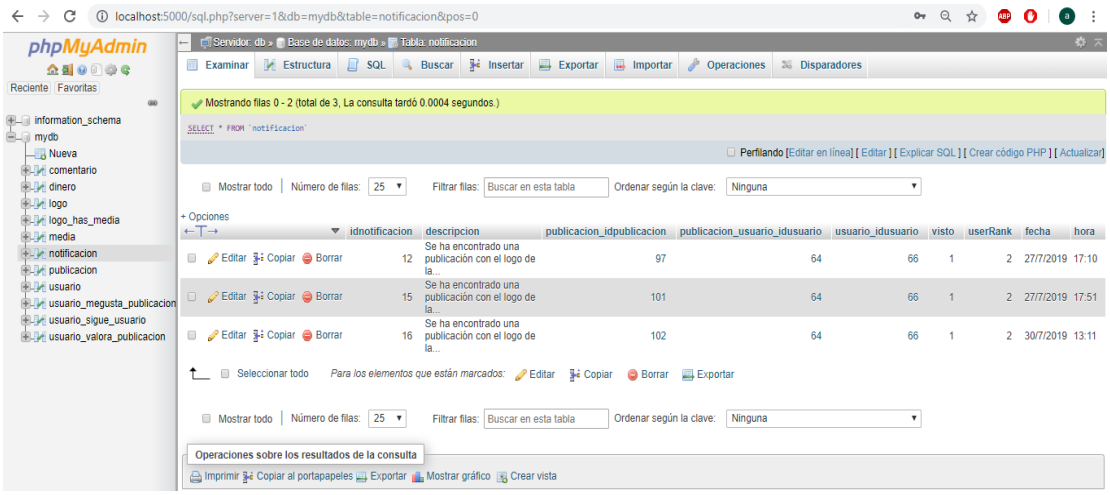


Figura 24: Captura de pantalla de la herramienta phpMyAdmin

PhpMyAdmin ha sido una herramienta muy útil en dicho proyecto, ya que, gracias a su funcionalidad, se ha podido gestionar (visualizar, eliminar, añadir o modificar) con facilidad los datos de la base de datos, entre otras cosas.

3.1.8. Docker

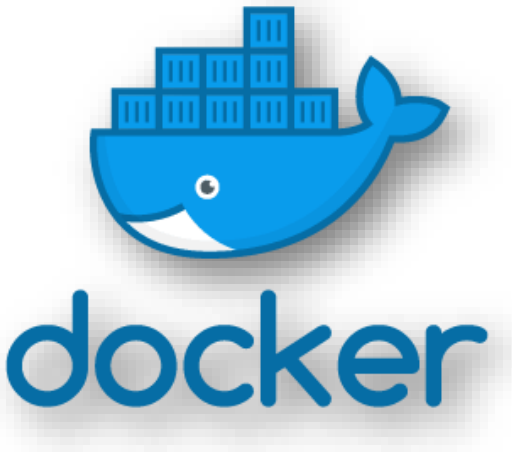


Figura 25: Imagen corporativa Docker

Docker es una herramienta de código abierto que despliega aplicaciones dentro de contenedores software. Dicha herramienta evita la sobrecarga de iniciar y mantener máquinas virtuales, puesto que utiliza características de aislamiento de recursos del *kernel Linux* para permitir que contenedores independientes se ejecuten dentro de una sola instancia de *Linux*.

Con *Docker* se puede empaquetar una aplicación en un contenedor y este podrá ser ejecutado en cualquier servidor *Linux*. Esto permite la portabilidad de ejecución de la aplicación.

La gran diferencia de *Docker* respecto a máquinas virtuales, es que *Docker* no requiere incluir un sistema operativo independiente.

En el presente Trabajo Fin de Grado vamos a ejecutar el lado servidor (*back-end*) [16] en *Docker*. De esta manera, podemos ejecutar el servidor con una línea de comando.

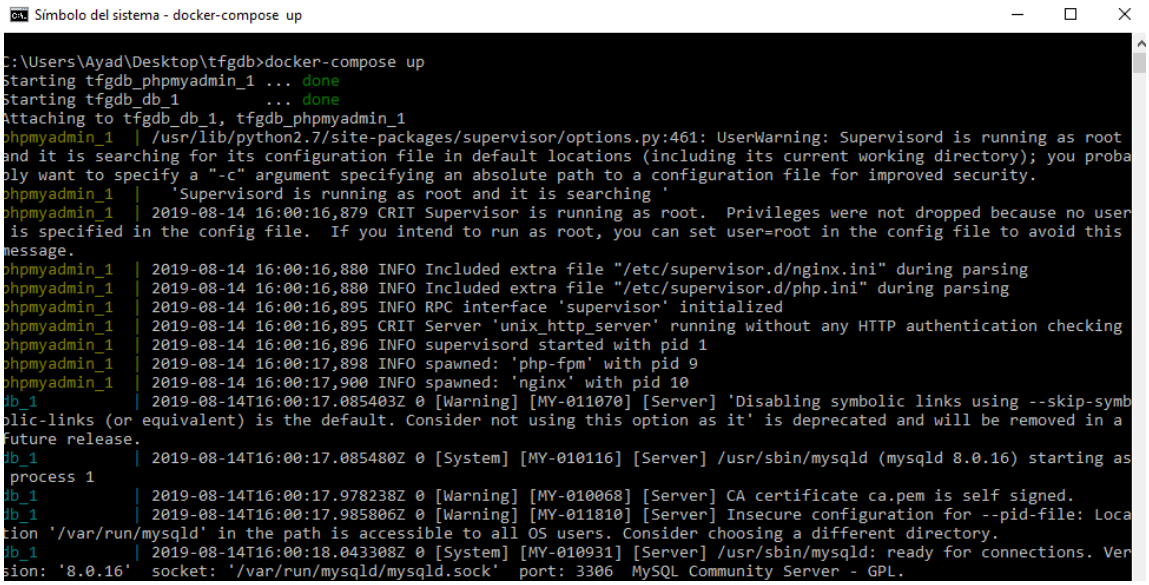


Figura 26: Captura de pantalla ejecutando el servidor con Docker

En el desarrollo de este proyecto vamos a ejecutar *phpMyAdmin*, *MySQL* y API *Rest* con *Docker*.

3.1.9. React Native

Figura 27: Imagen corporativa React Native

React Native [25] es un *framework* que permite desarrollar aplicaciones móviles híbridas utilizando JavaScript [24]. Con *React N*ative [25] tenemos acceso directo a todas las APIs y vistas nativas que ofrece el sistema operativo nativo. De esta manera, la experiencia y rendimiento es el mismo de una aplicación nativa.

Las aplicaciones más populares que usa el *framework React Native* [25] son:

- Facebook
- Instagram
- Skype
- Pinterest

React Native [25] no compila código JavaScript en el correspondiente código nativo directamente, ya que Java y Objective C/Swift son lenguajes fuertemente tipados mientras que JavaScript no lo es. En lugar de eso, un conjunto de componentes en *React* tiene su correspondiente equivalente en vista y componentes nativos.

3.1.10. Ionic

Figura 28: Imagen corporativa Ionic

Ionic [26] es un *framework* para el desarrollo de aplicaciones móviles híbridas que utiliza HTML5, CSS y *Apache Cordova* como base. También se puede usar *AngularJS* para gestionar las aplicaciones.

Las características de este *framework* son las siguientes:

- Utiliza JavaScript para el desarrollo de proyectos.
- Hace uso de HTML5 y CSS para ofrecer la mejor experiencia al usuario.
- Soporta el diseño adaptable de pantalla del dispositivo. De esta manera, soporta la gran mayoría de los dispositivos del mercado.
- Ofrece una gran cantidad de componentes listas para utilizar sin necesidad de programar.
- Dispone de una herramienta de línea de comandos con la que se puede crear, construir, probar y lanzar las aplicaciones.

3.1.11. Apache Cordova

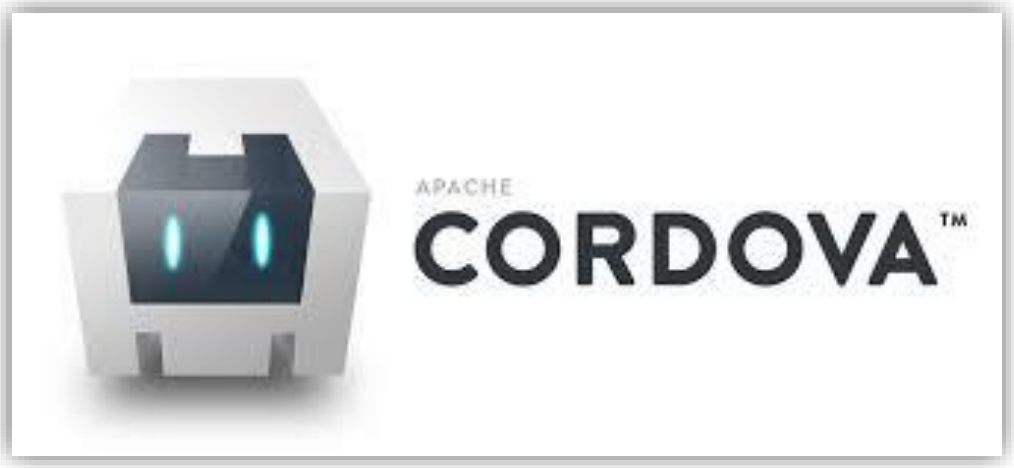


Figura 29: Imagen corporativa Apache Cordova

Apache Cordova es un conjunto de APIs que permite desarrollar aplicaciones móviles híbridas con acceso a funcionalidades nativas del dispositivo desde JavaScript [24]. Se puede desarrollar aplicaciones sin la necesidad de programar en el lenguaje nativo del sistema operativo. Este *framework* utiliza HTML5, CSS Y JavaScript para el desarrollo de aplicaciones. Las aplicaciones que son desarrolladas en este *framework* son empaquetados como aplicaciones nativas.

3.1.12. Cloud Vision

Cloud Vision [2] ofrece modelos de aprendizaje previamente preparados a través de la *API Rest*. Permite crear aplicaciones con distintas características de detección de imágenes, como la detección de rostros, etiquetado de imágenes, logotipos, puntos de referencia, etc.

Esta herramienta nos da la opción de compilar la app en dispositivos móviles con el *kit de AA* para *Firebase*, que proporciona SDK nativos de iOS y Android con la opción de usar las herramientas de reconocimiento de *Cloud Vision* y las API de Vision de AA en el dispositivo.

El *kit de AA* para *Firebase* se encuentra en su versión Beta, y algunas de las características que nos ofrece *Cloud Vision* [2] no está disponible en el *Kit de AA* como, por ejemplo, el reconocimiento de logotipos.

En la aplicación móvil se utilizará la API de *Cloud visión* [2], ya que no se podrá utilizar el *Kit de AA* porque no tenemos acceso a la herramienta de reconocimiento de logotipos por encontrarse en la versión beta.

3.2. Tecnologías

3.2.1. *Dart*

Figura 30: Imagen corporativa Dart

Dart es un lenguaje de programación de código abierto y utilizado en el framework *Flutter* [4], desarrollado por *Google*. Fue revelado en la conferencia *GOTO* en Dinamarca el día 10 de octubre de 2011. La primera versión estable apareció en noviembre del 2013, tras más de 2 años de su presentación.

El SDK de *Dart* es el kit de desarrollo de software de *Dart*. Incluye todas sus librerías y tiene herramientas muy útiles como el compilador de *Dart-to-JavaScript* que te permite compilar código *Dart* en *JavaScript* y la máquina virtual de *Dart*.

3.2.2. MySQL

Figura 31: Imagen corporativa MySQL

MySQL es un sistema de gestión de base de datos relacional y está considerada como uno de los sistemas de gestión más popular. Es utilizado en muchas aplicaciones grandes y populares como *Facebook*, *Twitter*, *YouTube*, etc.

MySQL es un sistema relacional de gestión de base de datos. Guarda sus datos en tablas en vez de guardar todos los datos en un archivo. Las tablas están conectadas mediante relaciones y con esto se consigue la posibilidad de combinar datos entre diferentes tablas.

Opté por elegir *MySQL* por su forma de gestionar relaciones entre las tablas, ya que el número de relacones de mi base de datos es significativo. Si hubiera elegido una base de datos *NoSQL* contendría mucha información duplicada en la base de datos.

3.2.3. Node.js

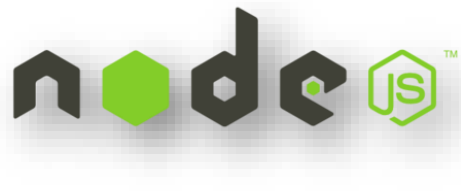


Figura 32: Imagen corporativa Node.js

Node.js [18] es un entorno en tiempo de ejecución multiplataforma y de código abierto, utilizado para el desarrollo de aplicaciones del lado del servidor basado en el lenguaje de programación *ECMAScript*.

Node.js [18] fue creado en 2009 con la intención de ser útil en la creación de programas de red altamente escalables, por ejemplo, en servidores web.

Node.js [18] tiene una arquitectura dirigida a eventos que permite entradas y salidas asíncronas. Estas elecciones tienen como objetivo la optimización del rendimiento y la escalabilidad en aplicaciones con muchas operaciones de entrada/salida y, además, de aplicación en tiempo real.

En este Trabajo Fin de Grado he optado por utilizar *Node.js* por su eficiencia y gran popularidad en las grandes compañías, como es el caso de *Netflix*, *PayPal*, *Uber*, *Ebay*, etc.

3.2.4. MySQL + Node.js

Figura 33: Imagen corporativa MySQL + Node.js

REST [23] es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON. Es una

alternativa en auge a otros protocolos estándar de intercambio de datos como SOAP (*Simple Object Access Protocol*), que disponen de una gran capacidad, pero también mucha complejidad. A veces es preferible una solución más sencilla de manipulación de datos como REST.

Características de REST:

- Protocolo cliente/servidor sin estado: cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla.
- Los objetos en REST siempre se manipulan a partir de la URI.
- Las operaciones más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son cuatro: POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar).

Ventajas de REST:

- Separa por completo la interfaz de usuario y el almacenamiento de datos.
- La API Rest siempre es independiente del tipo de plataformas o lenguajes.

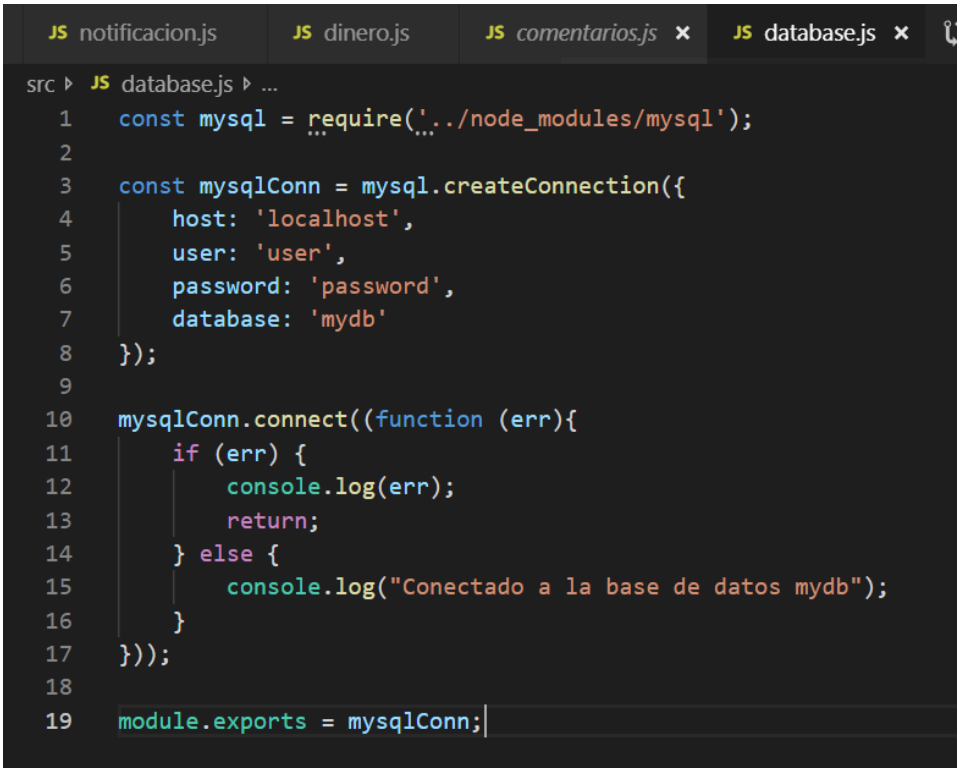


Figura 34: Captura de pantalla para la conexión de Node.js a la base de datos MySQL

3.2.5. JSON

Figura 35: Imagen corporativa JSON

JSON [20] es un formato de texto ligero utilizado para el intercambio de datos. Es sencillo de leer e interpretar para humanos, mientras que para las máquinas es simple interpretarlo y generarlo.

Las ventajas de JSON sobre XML son las siguientes:

- La sintaxis de JSON es más simple y fácil de entender que la sintaxis de XML.
- Al compartir de datos es más eficiente el uso de JSON, debido a que los datos se guardan en vectores, mientras que en XML se guardan en árboles.

En el proyecto se utilizará JSON para el intercambio de datos entre el cliente y servidor.

3.2.6. JavaScript

Figura 36: Imagen corporativa JavaScript

JavaScript es un lenguaje de programación interpretado *ECMAScript*. Se define como un lenguaje de programación orientado a objetos, débilmente tipado y dinámico. Se utiliza tanto en el lado cliente (*front-end*) [16] como en el lado servidor (*back-end*) [16].

Características de *JavaScript*:

- Es un lenguaje de alto nivel, ya que es más parecido al lenguaje natural humano y menos al lenguaje de las máquinas.

- El tipo está asociado al valor, no a la variable. Por ejemplo, un variable "z" en un momento dado puede ser un tipo de dato de tipo entero y, más adelante, puede ser una variable de tipo cadena.
- Utiliza prototipos en vez de clases para utilizar herencia.

En nuestro caso se utilizará para el lado servidor (*back-end*) [16] debido a que nuestra API Rest *Node.js* utiliza este lenguaje de programación.

JavaScript es una marca registrada de *Oracle Corporation*.

3.3. Herramienta optada para el desarrollo de aplicación híbrida

En el desarrollo del Trabajo Fin de Grado se ha optado por elegir el framework *Flutter* por su gran rendimiento en de desarrollo de aplicaciones móviles.

En mi opinión, me parece buena idea desarrollar aplicaciones haciendo uso de un mismo lenguaje, como es el caso de *Flutter* que hace uso de su lenguaje *Dart*. En las demás herramientas para el desarrollo de aplicaciones híbridas se debe programar en tres lenguajes distintos, como son: HTML, CSS y JavaScript. Esta es una de las razones principales por las que se ha optado desarrollar la aplicación móvil en este *framework*, además de su novedad e interés en profundizar en su lógica de desarrollo

Las diferencias con las distintas herramientas analizadas para el desarrollo de aplicaciones híbridas son las siguientes:

3.3.1. Ionic vs Flutter

Figura 37: Imagen corporativa Flutter e Ionic

Con el *framework* de *Flutter* y de *Ionic* [23] se pueden crear aplicaciones de alto rendimiento que funciona en todos los dispositivos.

Las diferencias son las siguientes:

- Al programar con *Ionic* se utilizan las herramientas y lenguajes de la web; con *Flutter* se utiliza un lenguaje de programación llamado *Dart* [3].
- *Flutter* ofrece muy buen rendimiento en los dispositivos móviles, pero es una mala elección para implementaciones basadas en web debido a las limitaciones de su arquitectura.

Con dicha comparación de estas dos herramientas para el desarrollo de aplicaciones móviles híbridas se ha podido comprobar que el rendimiento es mejor en el caso de *Flutter*.

3.3.2. *React Native* vs *Flutter*

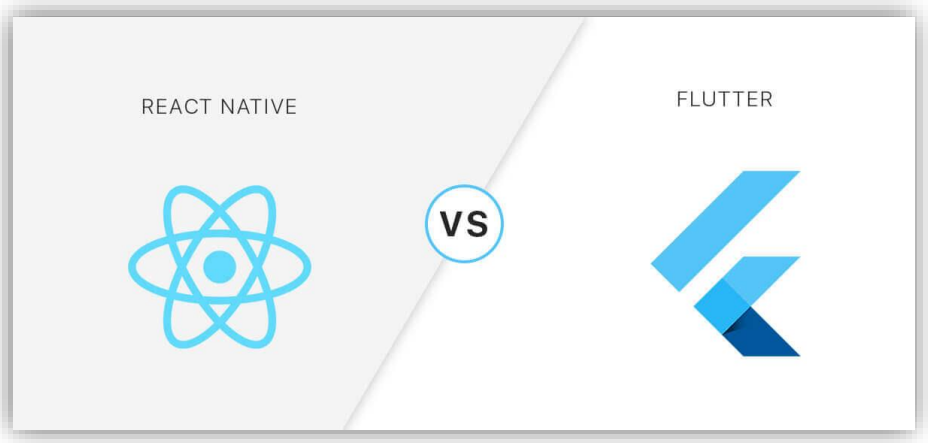


Figura 38: Imagen corporativa Flutter y React Native

Flutter y *React N*ative [24] son los *frameworks* más populares para el desarrollo de aplicaciones híbridas.

Las diferencias son las siguientes:

- *React Native* se basa en componentes nativos, mientras que *Flutter* funciona con un conjunto de *widgets*. Los widgets son los mejores para obtener un diseño personalizado y con soporte nativo.
- Al desarrollar una aplicación, el tiempo de desarrollo es mayor en *Flutter*. Los componentes listos para utilizar en *React Native* consiguen un desarrollo más simple y rápido.
- En *React Native*, en ocasiones, los desarrolladores se enfrentan a problemas cuando ejecutan la arquitectura de aplicación híbrida, pero en *Flutter*, al reutilizar el código, se facilita el uso a los desarrolladores.

- Con *React Native* se han desarrollado proyectos de gran tamaño y muy populares como *Facebook*, *Instagram*, *Skype*, etc. Sin embargo, *Flutter*, al ser una herramienta nueva, no encontramos aplicaciones tan reconocidas como es en el caso de *React Native*.

Comparando estos dos *frameworks* hemos podido comprobar que son apropiadas para el desarrollo de aplicaciones móviles híbridas.

En mi opinión, estas herramientas están muy igualadas para el desarrollo de aplicaciones móviles, pero he optado por *Flutter* ya que considero muy interesante el planteamiento de desarrollo de aplicaciones usando widgtes; además de la gran ventaja de hacer uso de un único lenguaje de programación.

3.3.3. Apache Cordova vs Flutter

Figura 39: Imagen corporativa Flutter y Apache Cordova

Las diferencias son las siguientes:

- En los dos casos creamos una única base de código y funcionará en todos los dispositivos.
- En *Apache Cordova* se deberá programar en HTML, CSS y JavaScript [17], y en *Flutter* se deberá programar en *Dart* [3].
- En el caso de *Apache Cordova* tenemos una desventaja a la hora de procesar porque no suele ser nativo.
- En *Flutter* nos encontramos con aplicaciones compiladas de forma nativa.

Comparando estos dos *frameworks* hemos podido observar que el rendimiento de *Flutter* es mayor respecto a *Apache Cordova*, ya que en *Flutter* las aplicaciones se compilan de forma nativa.

3.3.4. Crecimiento de Flutter

En la Figura 39 nos encontramos con el interés de *Flutter* en *StackOverFlow* comparándose con las distintas herramientas de desarrollo de aplicaciones híbridas. *Flutter* ha conseguido en poco tiempo una popularidad muy grande en el desarrollo de aplicaciones como se muestra en la gráfica.

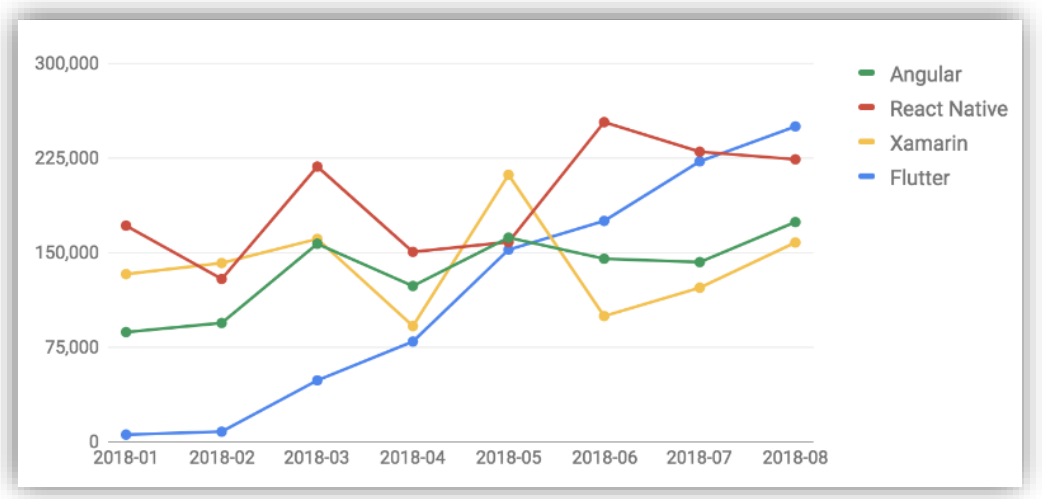


Figura 40: Crecimiento de Flutter

En *Flutter* podemos conseguir una tasa de refresco de 120 fotogramas por segundo y en las otras herramientas podemos alcanzar un máximo de 60 fotogramas por segundo. Con la herramienta *Flutter* podemos apreciar un rendimiento mayor respecto a las otras herramientas.

4. AI Publicity

4.1. Especificación preliminar

Se precisa de un sistema para plataforma móvil hibrida que cumpla con las siguientes características:

- El sistema estará compuesto por dos componentes principales: una parte cliente y una parte servidor. La parte cliente será desarrollada con el *framework Fl*utter [4] y la parte servidor será desarrollada como una API *Rest* utilizando *Node.js y MySQL*.
- El sistema permitirá a los usuarios darse de alta en el sistema. Para darse de alta necesitará añadir información del usuario y necesitará verificar el correo electrónico.
- El sistema permitirá la autentificación de usuarios ya existentes mediante su correo electrónico y clave de usuario.
- EL sistema ofrecerá la posibilidad de recuperar la contraseña, enviando una nueva contraseña al correo electrónico.
- Existen dos tipos de usuarios: Usuario normal y Usuario empresa.
- Cualquier usuario podrá subir una publicación.
- Al subir una imagen en una publicación, el sistema analizará la imagen en busca de algún logotipo. Al encontrar algún logo se le mandará una notificación a la empresa.
- Los usuarios podrán interactuar con las publicaciones, indicando “Me gusta” o valorar una publicación de 1 a 5 estrellas o escribir un comentario al respecto. Los comentarios no admiten respuestas.
- Las interacciones “Me gusta” dispondrán de un contador.
- Las interacciones de valoración de una publicación dispondrán de un contador y de la valoración media de las valoraciones de los usuarios.
- Un usuario podrá ser seguido por cualquier usuario y el usuario podrá seguir a cualquier usuario del sistema.
- En el perfil del usuario se podrá visualizar los usuarios que sigue y a los que le siguen.
- El usuario tendrá a su disposición un buscador para realizar búsquedas de usuarios del sistema.
- El usuario dispondrá de un listado de ganancias (usuario normal) o pagos (usuario empresa).

4.2. Análisis del sistema

4.2.1. Usuarios del sistema

U-001	**Usuario No Registrado**
Descripción	Al iniciar la aplicación el usuario no se encuentra registrado en la aplicación.
Comentarios	Este usuario debe registrarse o iniciar sesión para acceder al contenido de la aplicación.

U-002	**Usuario Registrado**
Descripción	Tiene a disposición diferentes funcionalidades, y es quien interactúa en la aplicación. Este usuario puede ser de dos tipos: usuario normal o usuario empresa.
Comentarios	Usuario normal: hace publicidad de los diferentes productos o sitios de una empresa. Usuario empresa: empresa con una cuenta para patrocinar sus productos y recibir publicidad de los usuarios de sus productos o sitios.

4.2.2. Actores del sistema

A-001	**Usuario No Registrado**
Descripción	Este actor solo dispone de tres funcionalidades: Iniciar sesión, registrarse o recuperar contraseña.

A-002	**Usuario Registrado**
Descripción	Tiene a disposición diferentes funcionalidades, y es quien interactúa en la aplicación. Este usuario puede ser de dos tipos: usuario normal o usuario empresa.

A-003	**Correo Electrónico**
Descripción	Sistema externo que sirve para poder realizar el registro del usuario, así como también la recuperación de contraseñas

4.2.3. Sistemas externos

SE-001	**Correo Electrónico**
Descripción	Sistema que gestiona y almacena correos electrónicos de un usuario, necesario para la confirmación del registro correcto de un usuario.
Comentarios	El usuario recibirá un email para la confirmación de su correo electrónico y para la recuperación de contraseña.

4.2.4. Funciones del sistema

FS-001	**Buscar usuario**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá buscar a cualquier empresa o usuario en la aplicación.
Comentarios	El usuario podrá visualizar al usuario y a todo su contenido.

FS-002	**Visualizar notificaciones**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá visualizar las notificaciones recibidas.
Comentarios	El usuario puede ver la información de esa notificación.

FS-003	**Buscar usuario**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá buscar a cualquier empresa o usuario en la aplicación.
Comentarios	El usuario podrá ver al usuario y todo su contenido.

FS-004	**Visualizar perfil propio**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá ver su perfil propio con todas sus publicaciones.
Comentarios	El usuario podrá cambiar su imagen de perfil en esta venta.

FS-005	**Visualizar página principal**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá ver la ventana principal de la aplicación con distintos listados de publicaciones.
Comentarios	El usuario podrá filtrar por los distintos listados de publicaciones.

FS-006	**Visualizar ajustes**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá visualizar la ventana de ajustes para realizar cambios en su cuenta.
Comentarios	El usuario podrá cambiar el nombre, contraseña, ver usuarios bloqueados, desactivar cuenta y cambiar a cuenta de empresa.

FS-007	**Visualizar publicación**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá visualizar el contenido y toda la información de una publicación.
Comentarios	El usuario podrá añadir valoraciones a la publicación y dar me gusta.

FS-008	**Dar o quitar me gusta**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá dar like a una publicación.
Comentarios	Al presionar en el icono del corazón añadirá o quitara el like de la publicación.

FS-009	**Añadir ubicación**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá añadir la ubicación de la publicación.
Comentarios	Ninguno

FS-010	**Valorar publicación**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá dar una valoración de una publicación.
Comentarios	La valoración se mide de 1 a 5 estrellas.

FS-011	**Añadir comentario**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá añadir un comentario a la publicación.
Comentarios	Ninguno.

FS-012	**Visualizar comentarios**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá ver los comentarios de una publicación.
Comentarios	Ninguno.

FS-013	**Añadir publicación**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá añadir una publicación nueva.
Comentarios	El usuario podrá añadir en la publicación imágenes, ubicación y una descripción.

FS-014	**Visualizar perfil**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá ver el perfil de otro usuario o empresa.
Comentarios	Ninguno.

FS-015	**Cambiar imagen perfil propio**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá cambiar la imagen de perfil.
Comentarios	Ninguno.

FS-016	**Seguir usuario**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá seguir a otro usuario o empresa.
Comentarios	Ninguno.

FS-017	**Dejar de seguir usuario**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá dejar de seguir otro usuario o empresa.
Comentarios	Ninguno.

FS-018	**Visualizar imágenes de usuario**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá ver todas las imágenes del usuario en su perfil.
Comentarios	Ninguno.

FS-019	**Cambiar nombre de usuario**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá cambiar su nombre de usuario, su nombre y apellidos.
Comentarios	Ninguno.

FS-020	**Cambiar contraseña**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá cambiar su contraseña.
Comentarios	Ninguno.

FS-021	**Cambiar a cuenta empresa**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá cambiar su cuenta a cuenta de empresa.
Comentarios	Ninguno.

FS-022	**Desactivar cuenta**
Actores	U-002 Usuario Registrado
Descripción	El usuario podrá desactivar su cuenta temporalmente en la aplicación.
Comentarios	Ninguno.

FS-023	**Iniciar sesión**
Actores	U-001 Usuario No Registrado
Descripción	El usuario podrá iniciar sesión en la aplicación.
Comentarios	Para poder acceder necesitara su email y su clave.

FS-024	**Registrarse**
Actores	U-001 Usuario No Registrado
Descripción	El usuario podrá registrarse en la aplicación.
Comentarios	Deberá rellenar la información solicitada y verificar su correo electrónico.

FS-025	**Recuperar contraseña**
Actores	U-001 Usuario No Registrado
Descripción	El usuario podrá recuperar la contraseña de la aplicación.
Comentarios	Se le enviará una nueva contraseña al correo electrónico.

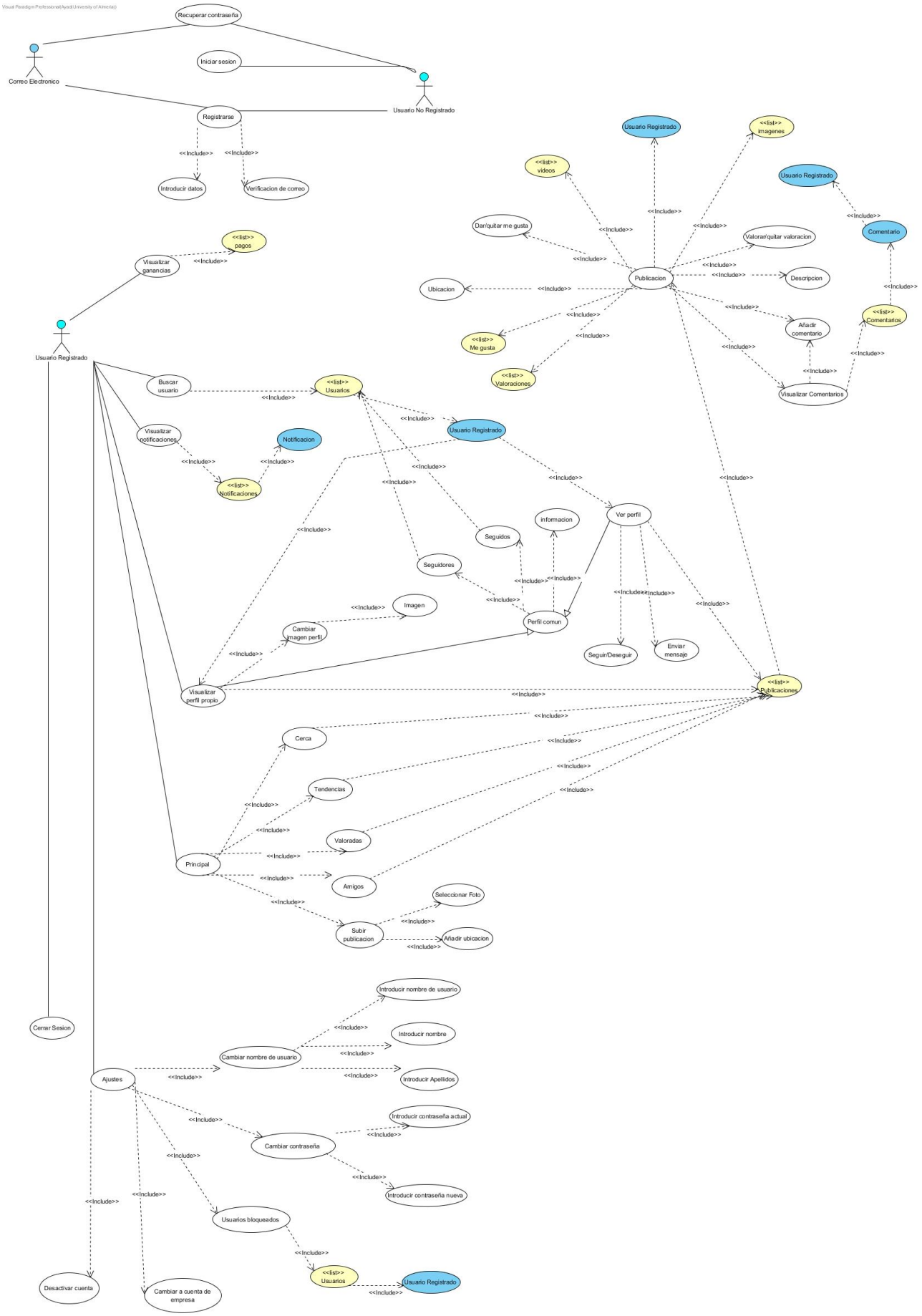
FS-026	**Visualizar seguidores**
Actores	U-002 Usuario Registrado
Descripción	Un usuario puede visualizar sus seguidores en forma de lista.
Comentarios	Ninguno

FS-027	**Visualizar seguidos**
Actores	U-002 Usuario Registrado
Descripción	Un usuario puede visualizar a los usuarios que sigue en forma de lista.
Comentarios	Ninguno.

4.3. Casos de uso

En el diagrama de casos de uso se han utilizado distintos colores para distinguir cada elemento que interviene en la aplicación:

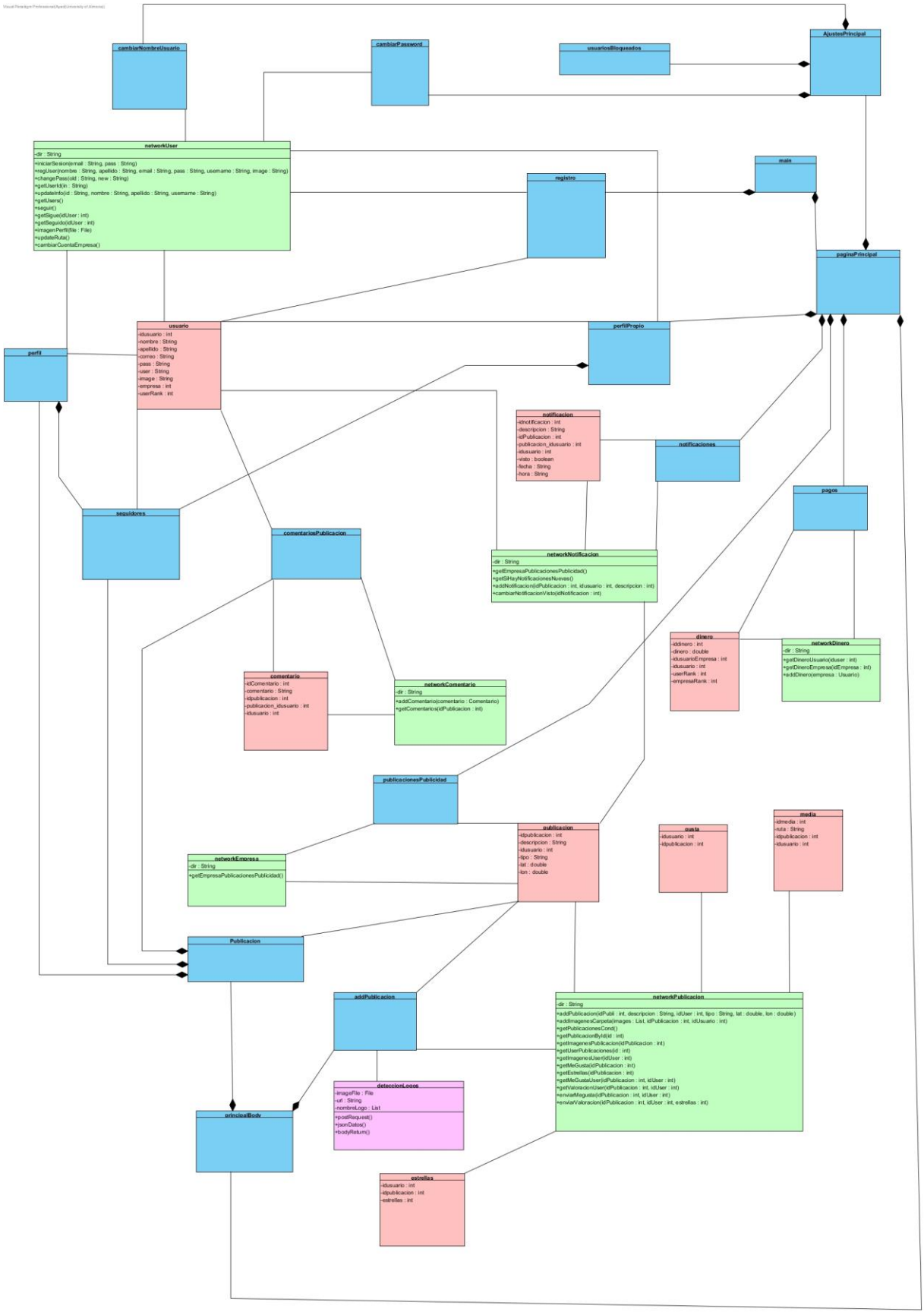
- Azul claro: actores que intervienen en la aplicación.
- Amarillo: listado de objetos de la aplicación.
- Blanco: funcionalidades del sistema.
- Azul oscuro: Objetos del sistema.


Recuperar contraseña
Correo Electronico
Iniciar sesion
Usuario No Registrado
Registrarse
<<Include>>
Introducir datos
Verificacion de correo
Usuario Registrado
<<list>> imagenes
<<list>> videos
Usuario Registrado
Comentario
Dar/quitar me gusta
Valorar/quitar valoracion
Publicacion
Descripcion
Ubicacion
<<list>> pagos
Visualizar ganancias
Añadir comentario
<<list>> Comentarios
<<list>> Me gusta
<<list>> Valoraciones
Visualizar Comentarios
Usuario Registrado
Buscar usuario
<<list>> Usuarios
Visualizar notificaciones
Notificacion
<<list>> Notificaciones
Usuario Registrado
Ver perfil
informacion
Seguidos
Seguidores
Imagen
Cambiar imagen perfil
Perfil comun
Seguir/Deseguir
Enviar mensaje
Visualizar perfil propio
<<list>> Publicaciones
Cerca
Tendencias
Valoradas
Principal
Amigos
Seleccionar Foto
Subir publicacion
Añadir ubicacion
Cerrar Sesion
Introducir nombre de usuario
Introducir nombre
Cambiar nombre de usuario
Introducir Apellidos
Ajustes
Introducir contraseña actual
Cambiar contraseña
Introducir contraseña nueva
Usuarios bloqueados
<<list>> Usuarios
Usuario Registrado
Desactivar cuenta
Cambiar a cuenta de empresa

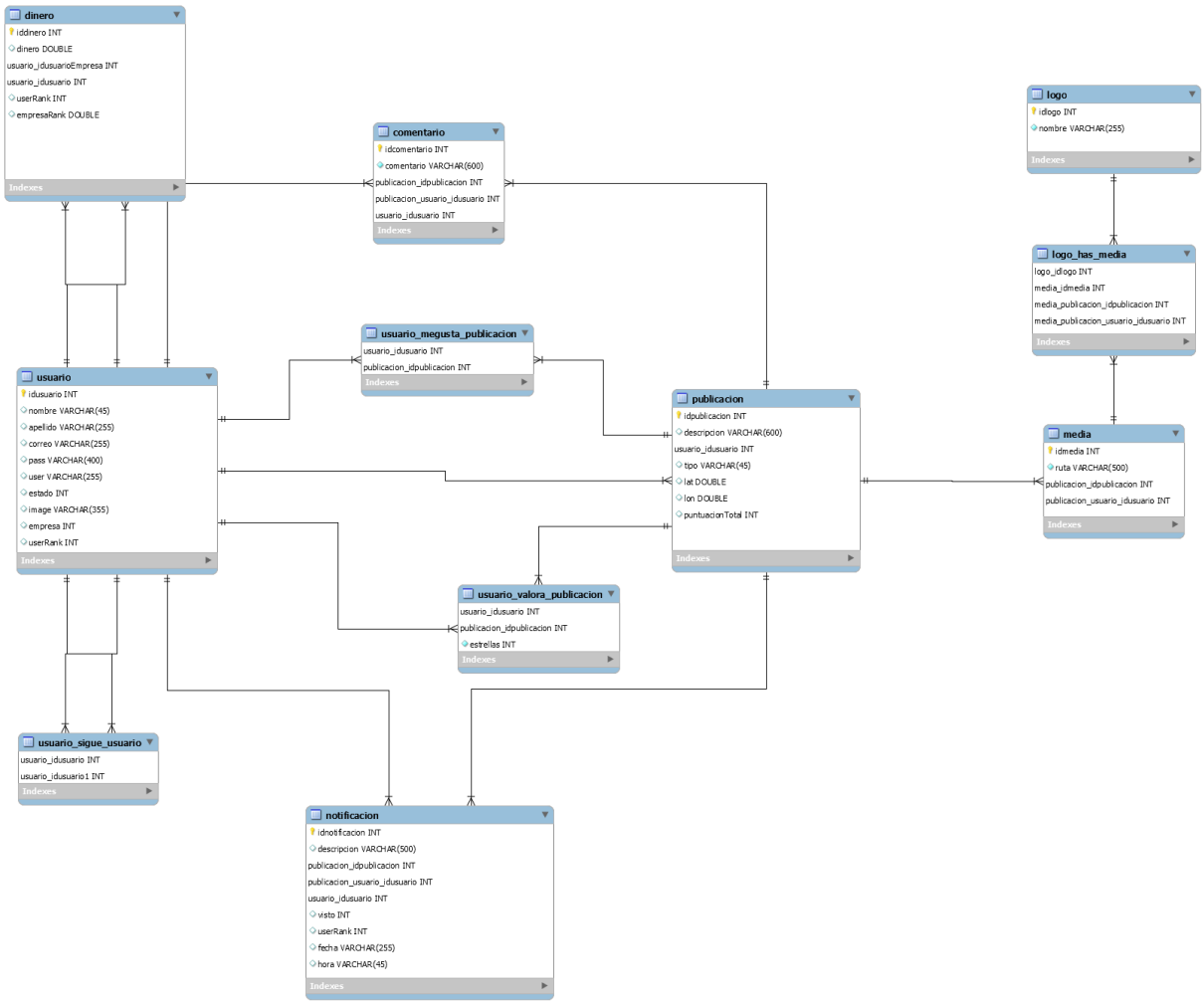
4.4. Diagrama de clases

En el diagrama de clases se han utilizado diferentes colores para distinguir cada elemento que interviene en la aplicación:

- Azul: ventanas de la aplicación.
- Verde: comunicación con la API *Rest*.
- Salmón: objetos de la aplicación.


cambiarNombreUsuario
cambiarPassword
usuariosBloqueados
AjustesPrincipal
networkUser
-dir : String
+iniciarSesion(email : String, pass : String)
+regUser(nombre : String, apellido : String, email : String, pass : String, username : String, image : String)
+changePass(old : String, new : String)
+getUserId(in : String)
+updateInfo(id : String, nombre : String, apellido : String, username : String)
+getUsers()
+seguir()
+getSigue(idUser : int)
+getSeguido(idUser : int)
+imagenPerfil(file : File)
+updateRuta()
+cambiarCuentaEmpresa()
registro
main
paginaPrincipal
usuario
-idusuario : int
-nombre : String
-apellido : String
-correo : String
-pass : String
-user : String
-image : String
-empresa : int
-userRank : int
perfil
perfilPropio
notificacion
-idnotificacion : int
-descripcion : String
-idPublicacion : int
-publicacion_idusuario : int
-idusuario : int
-visto : boolean
-fecha : String
-hora : String
notificaciones
pagos
seguidores
comentariosPublicacion
networkNotificacion
-dir : String
+getEmpresaPublicacionesPublicidad()
+getSiHayNotificacionesNuevas()
+addNotificacion(idPublicacion : int, idusuario : int, descripcion : int)
+cambiarNotificacionVisto(idNotificacion : int)
dinero
-iddinero : int
-dinero : double
-idusuarioEmpresa : int
-idusuario : int
-userRank : int
-empresaRank : int
networkDinero
-dir : String
+getDineroUsuario(idUser : int)
+getDineroEmpresa(idEmpresa : int)
+addDinero(empresa : Usuario)
comentario
-idComentario : int
-comentario : String
-idpublicacion : int
-publicacion_idusuario : int
-idusuario : int
networkComentario
-dir : String
+addComentario(comentario : Comentario)
+getComentarios(idPublicacion : int)
publicacionesPublicidad
publicacion
-idpublicacion : int
-descripcion : String
-idusuario : int
-tipo : String
-lat : double
-lon : double
networkEmpresa
-dir : String
+getEmpresaPublicacionesPublicidad()
cuenta
-idusuario : int
-idpublicacion : int
media
-idmedia : int
-ruta : String
-idpublicacion : int
-idusuario : int
Publicacion
addPublicacion
networkPublicacion
-dir : String
+addPublicacion(idPubli : int, descripcion : String, idUser : int, tipo : String, lat : double, lon : double)
+addImagenesCarpeta(images : List, idPublicacion : int, idUsuario : int)
+getPublicacionesCond()
+getPublicacionById(id : int)
+getImagenesPublicacion(idPublicacion : int)
+getUserPublicaciones(id : int)
+getImagenesUser(idUser : int)
+getMeGusta(idPublicacion : int)
+getEstrellas(idPublicacion : int)
+getMeGustaUser(idPublicacion : int, idUser : int)
+getValoracionUser(idPublicacion : int, idUser : int)
+enviarMegusta(idPublicacion : int, idUser : int)
+enviarValoracion(idPublicacion : int, idUser : int, estrellas : int)
deteccionLogos
-imageFile : File
-url : String
-nombreLogo : List
+postRequest()
+jsonDatos()
+bodyReturn()
principalBody
estrellas
-idusuario : int
-idpublicacion : int
-estrellas : int

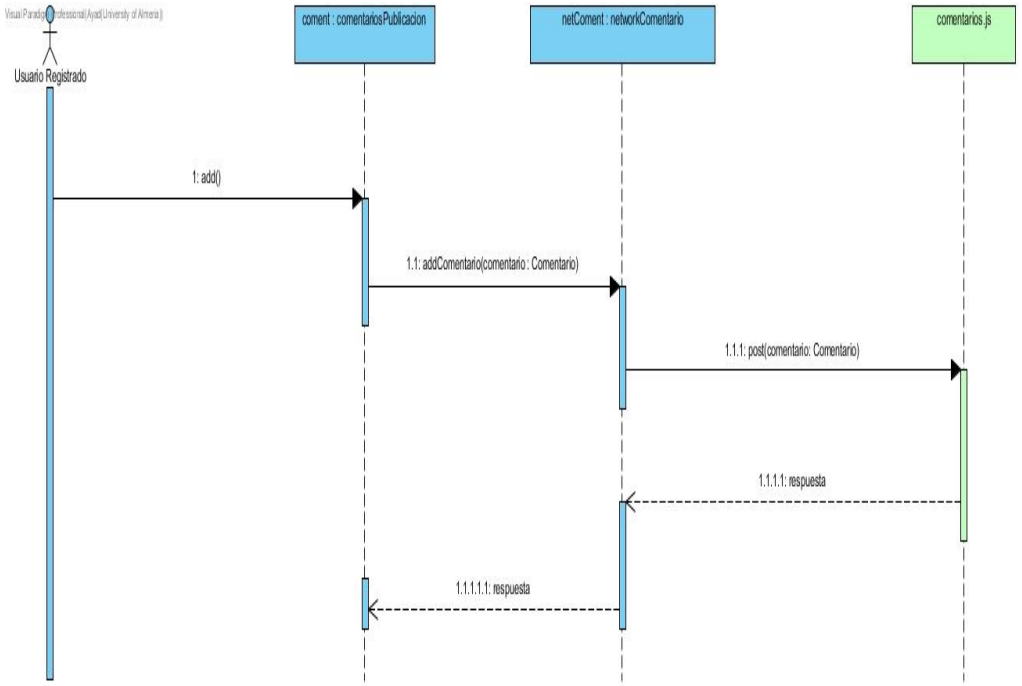
4.5. Diagrama de la base de datos



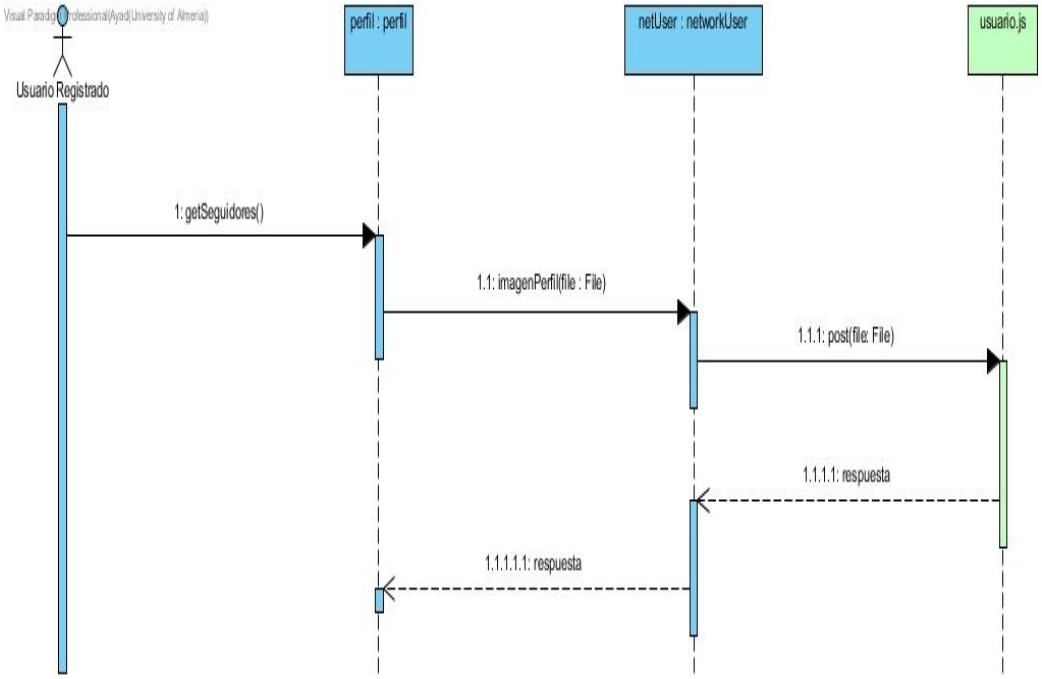
4.6. Diagramas de secuencias

En el diagrama de secuencias se ha utilizado dos colores para distinguir entre la aplicación híbrida (Color azul) y la API *Rest* (Color verde). En estos diagramas podemos observar las llamadas a la API *Rest* para guardar, cargar o modificar datos del sistema.

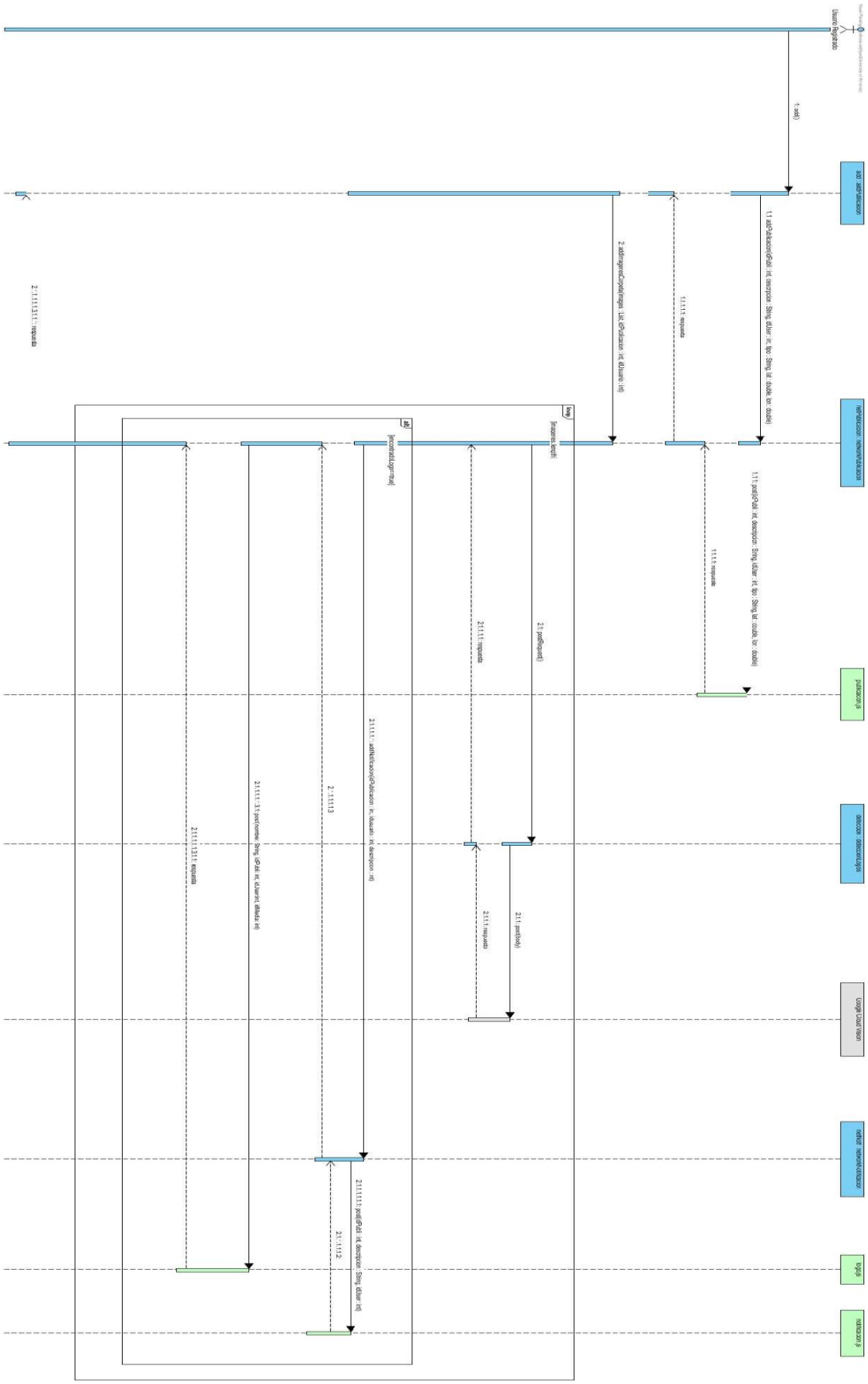
4.6.1. Añadir comentario a publicación



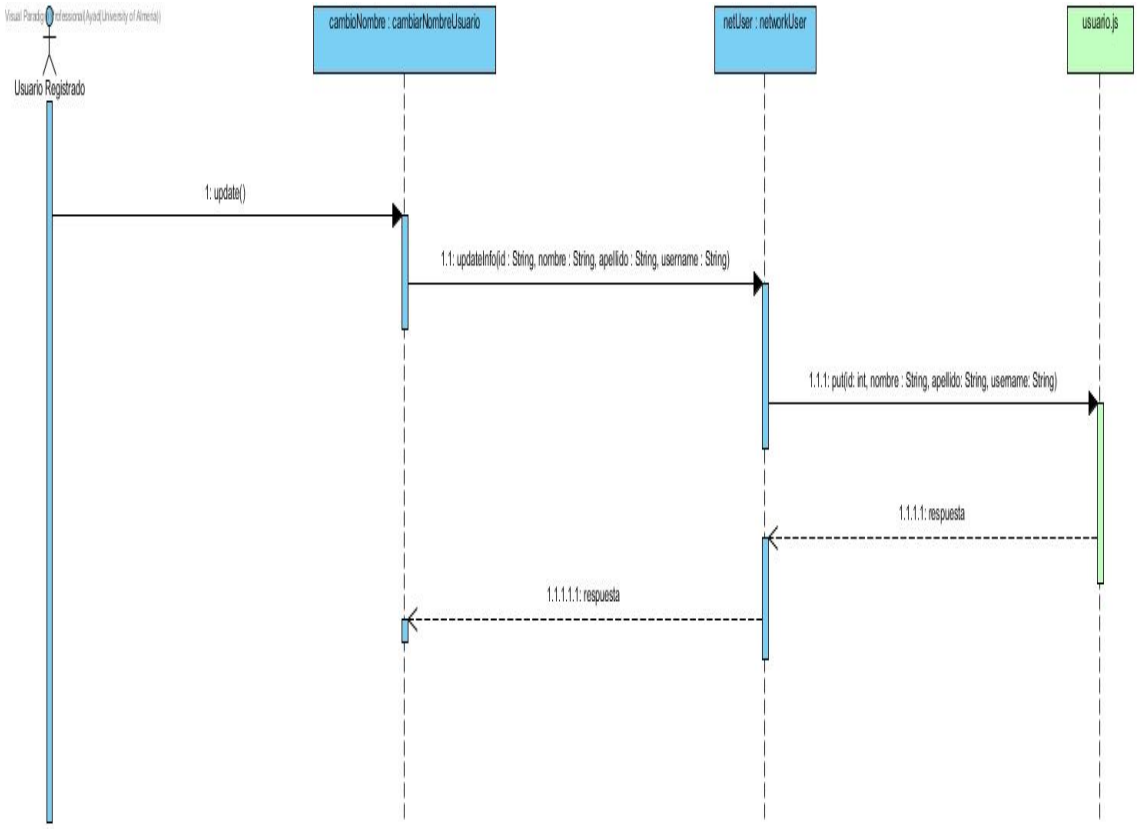
4.6.2. Cambiar imagen de perfil



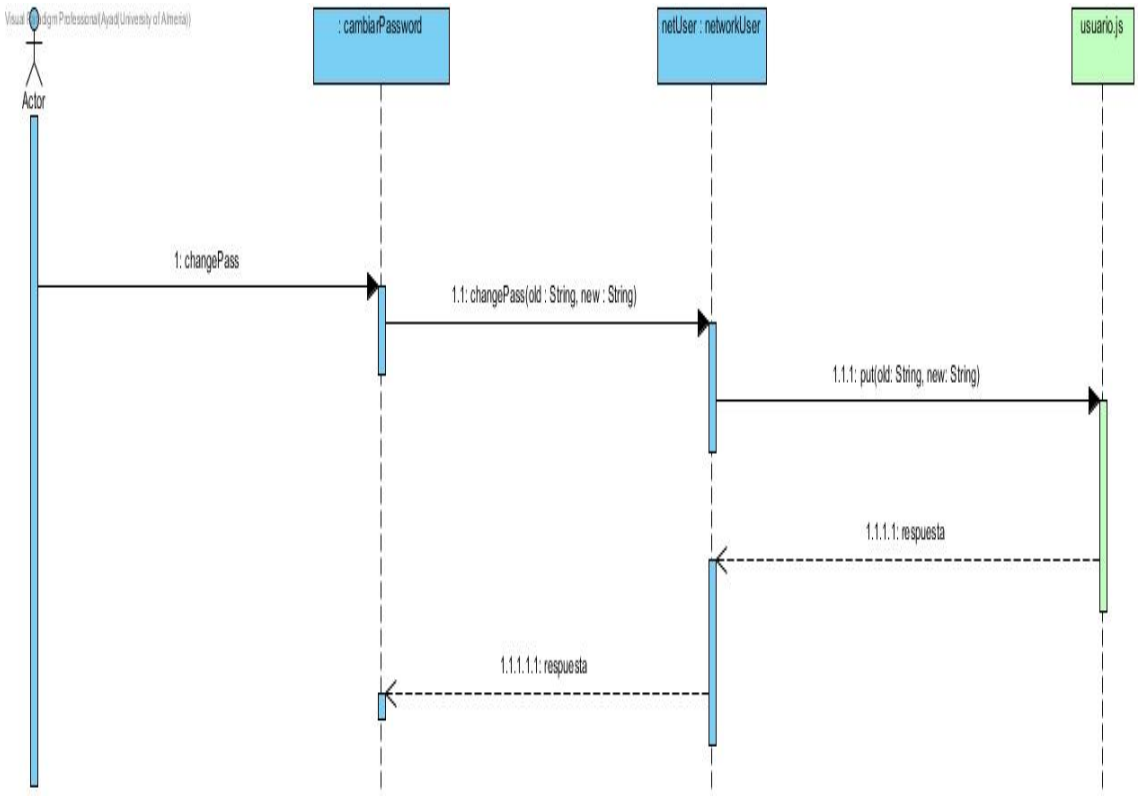
4.6.3. Añadir publicación



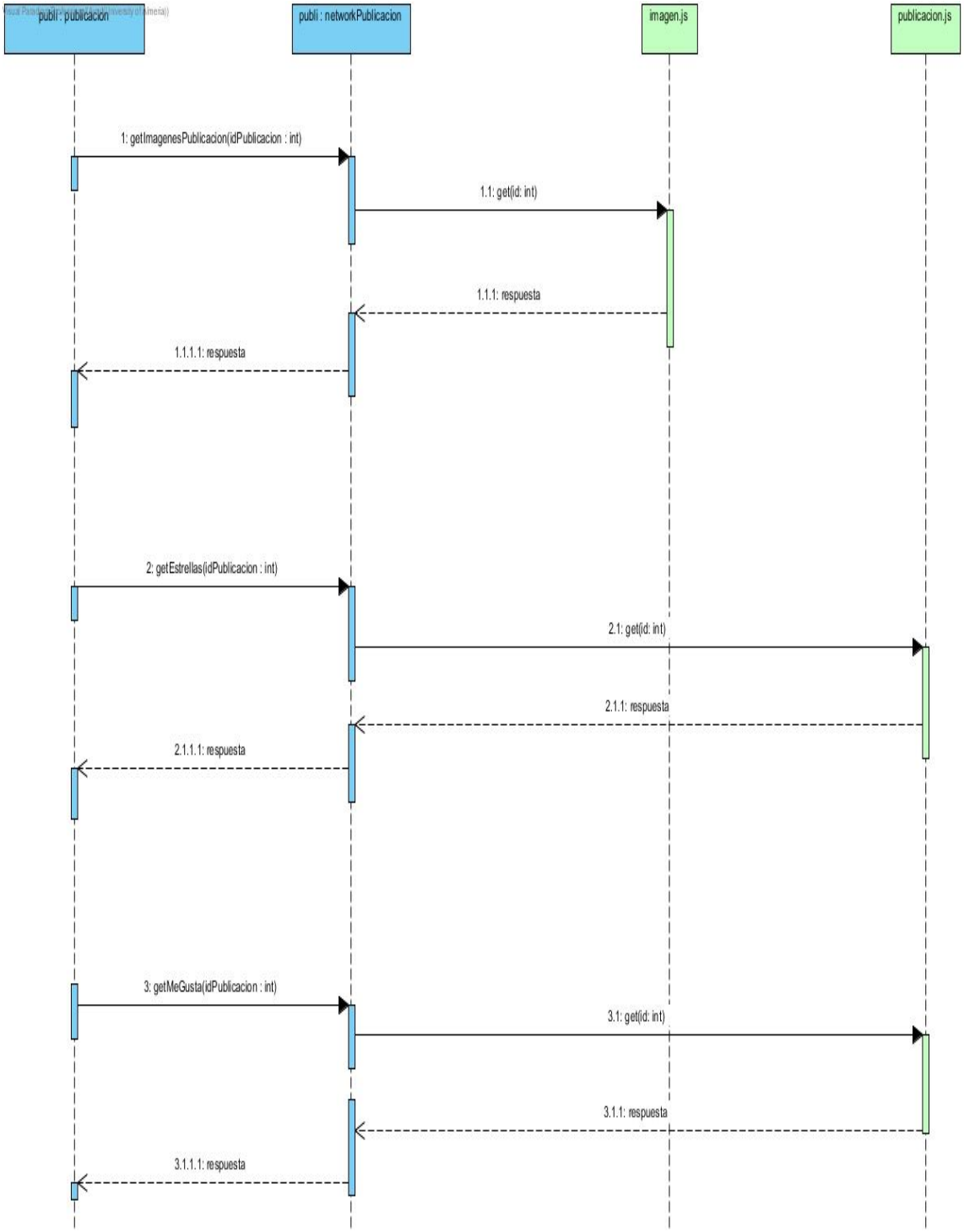
4.6.4. Modificar información de usuario



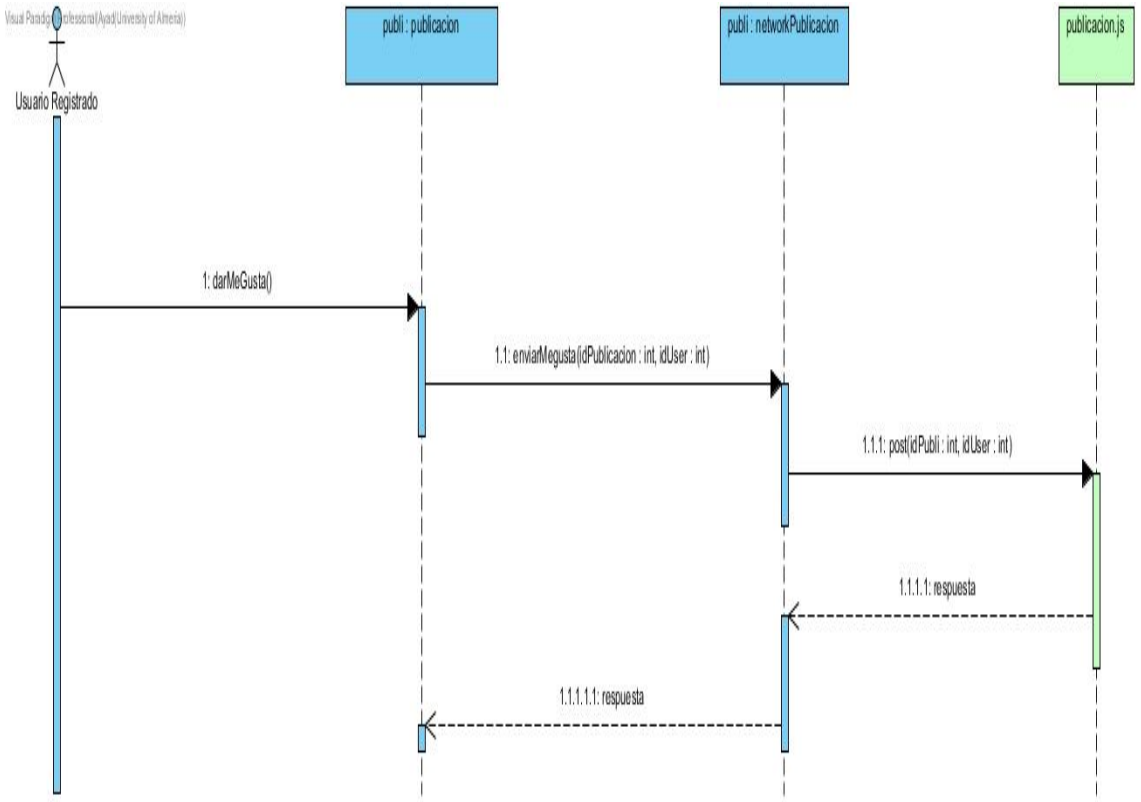
4.6.5. Cambiar contraseña de usuario



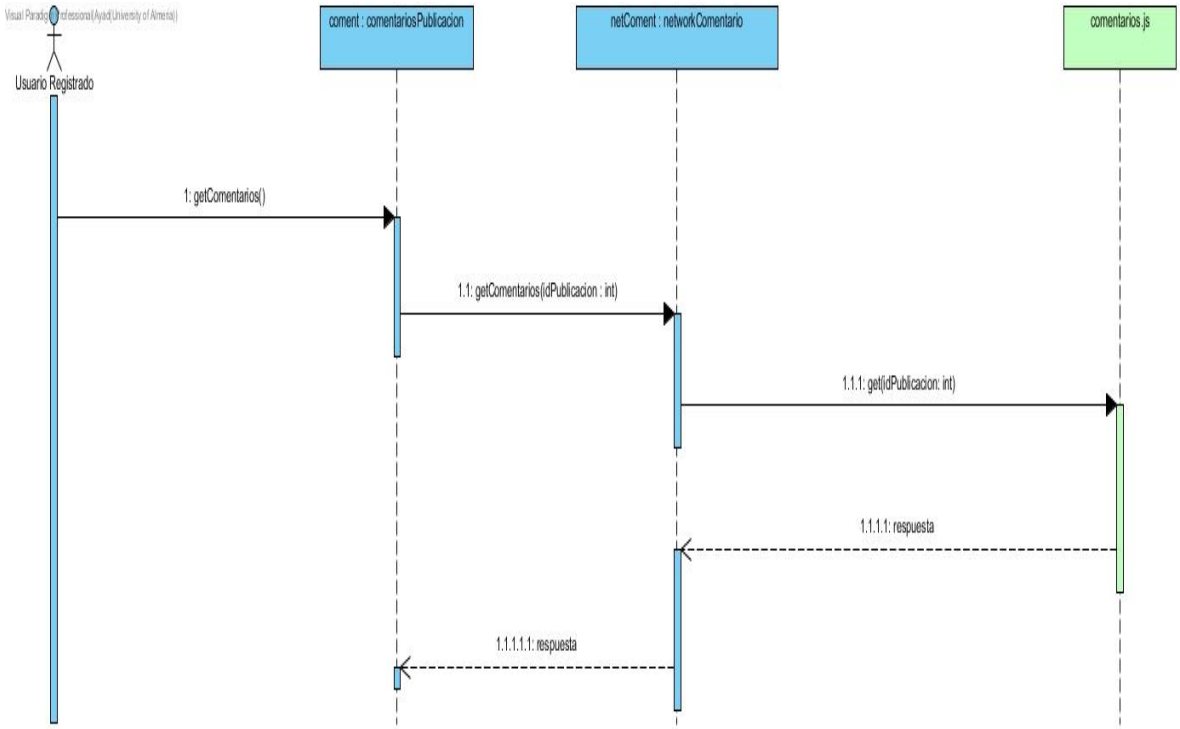
4.6.6. Cargar contenido de publicación



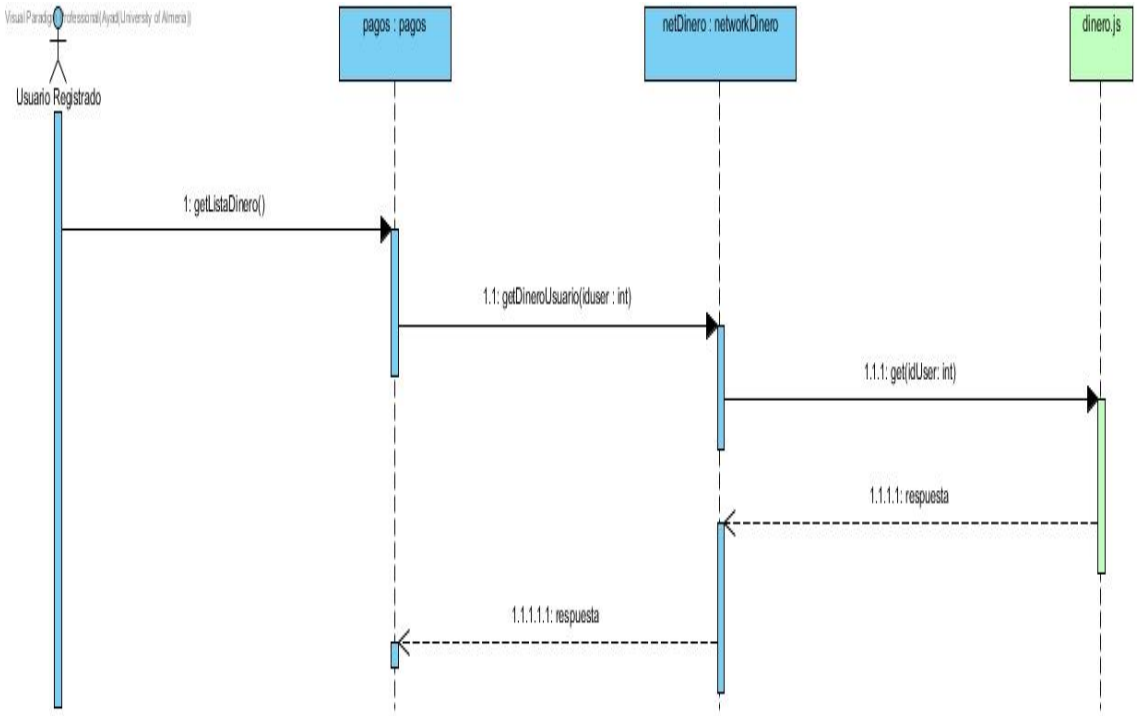
4.6.7. Dar me gusta a una publicación



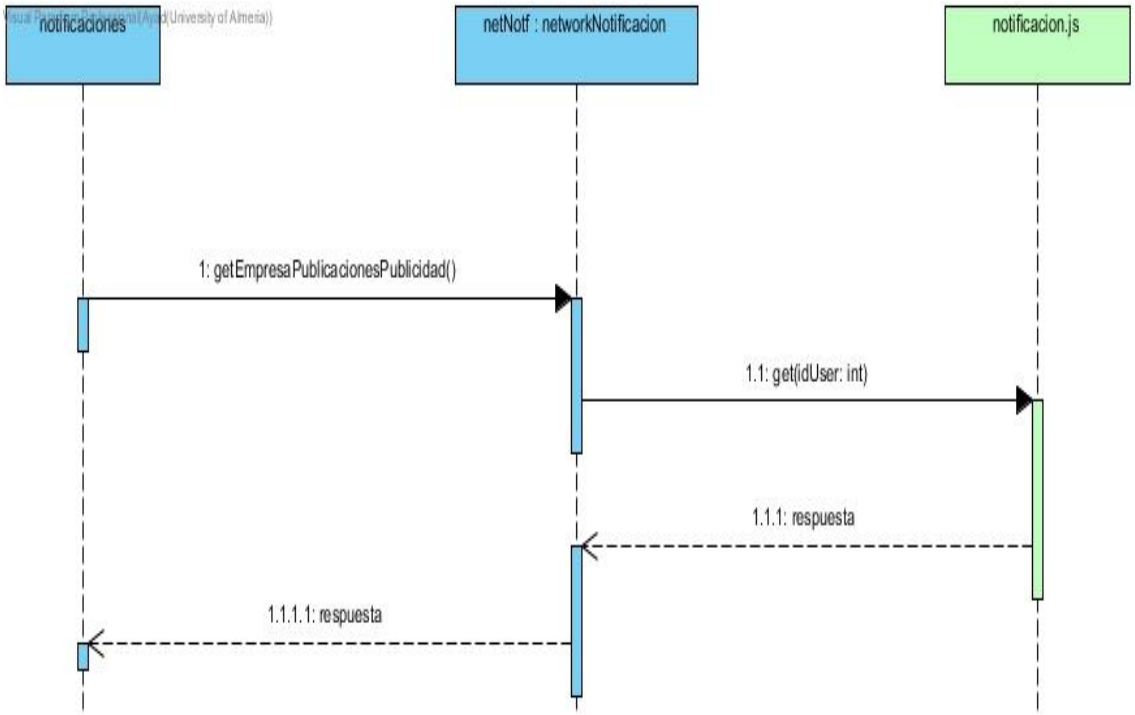
4.6.8. Cargar comentarios de una publicación



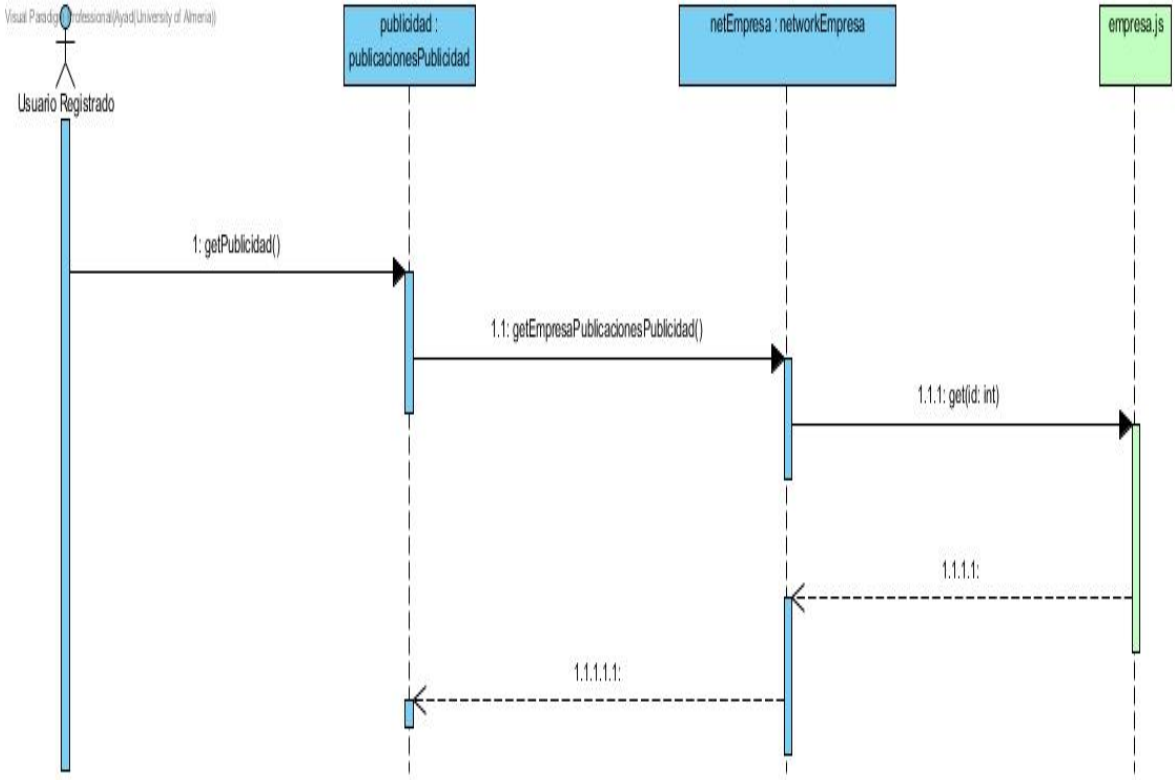
4.6.9. Cargar listado de ganancias



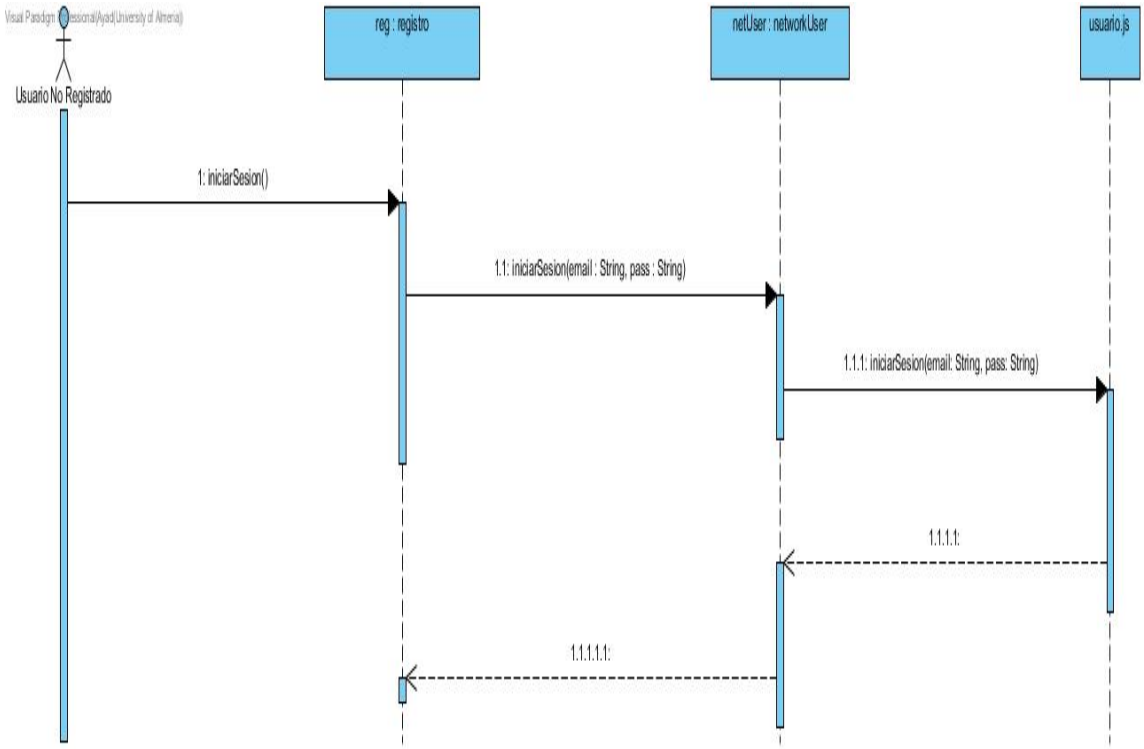
4.6.10. Cargar listado de notificaciones de la empresa



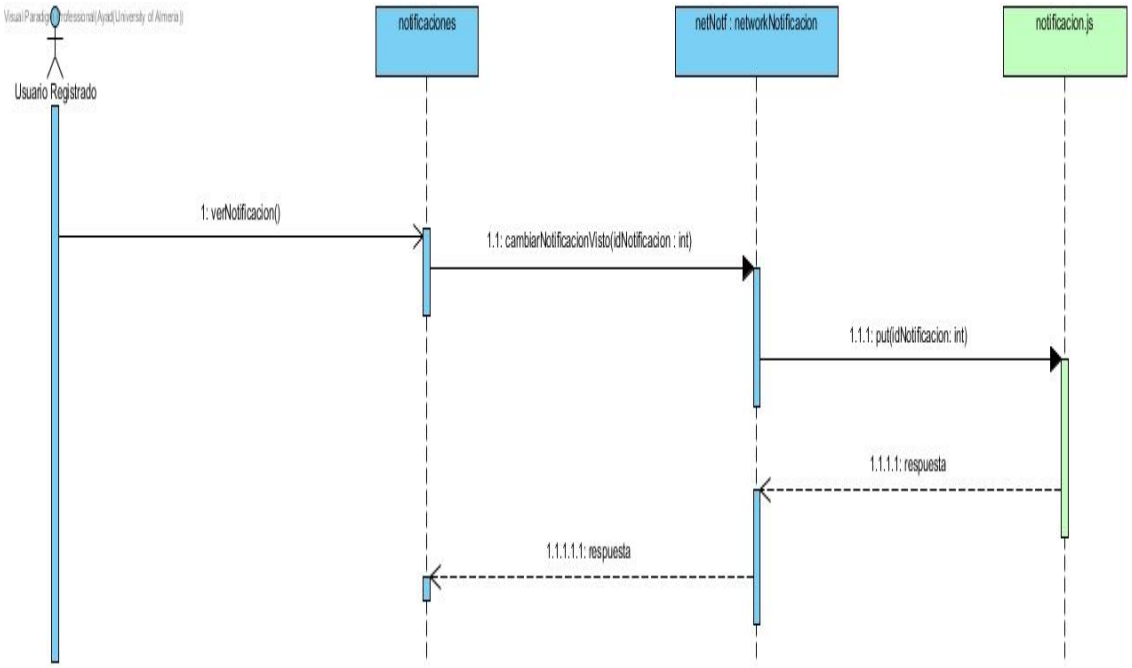
4.6.11. Cargar listado de publicaciones de publicidad de la empresa



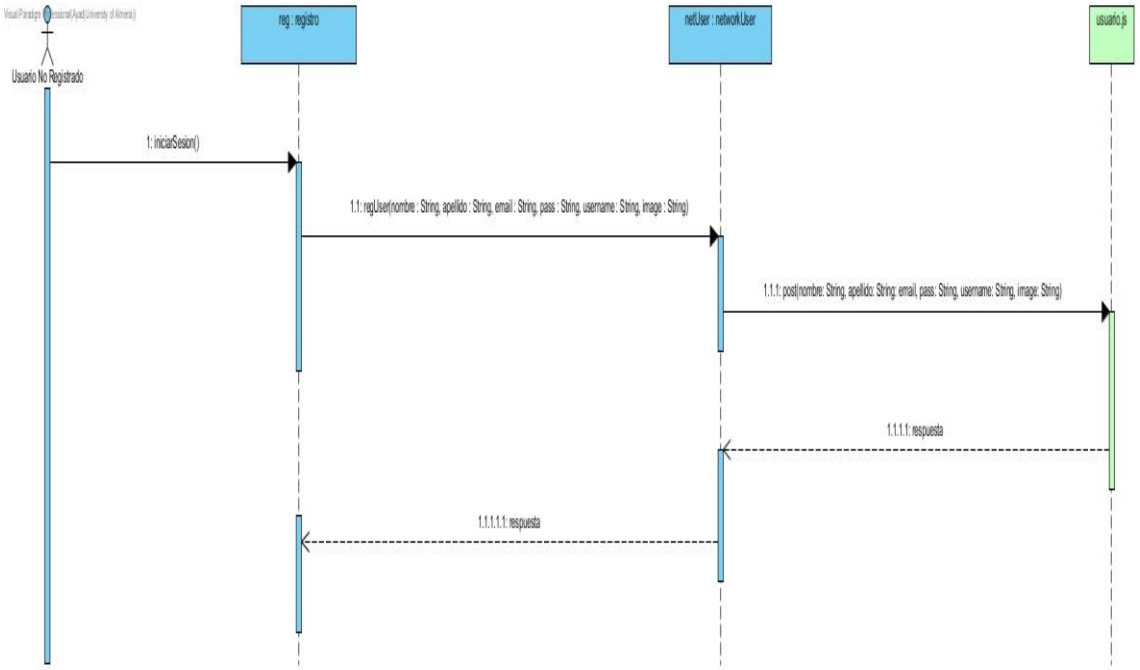
4.6.12. Iniciar sesión



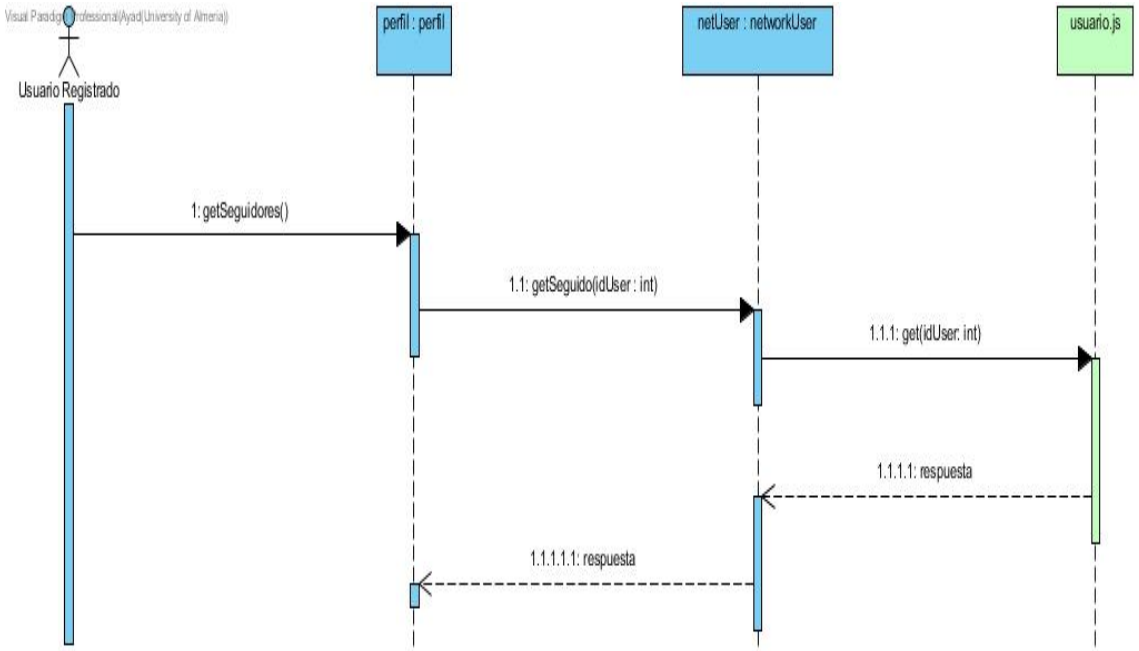
4.6.13. Cambiar notificación a visualizado



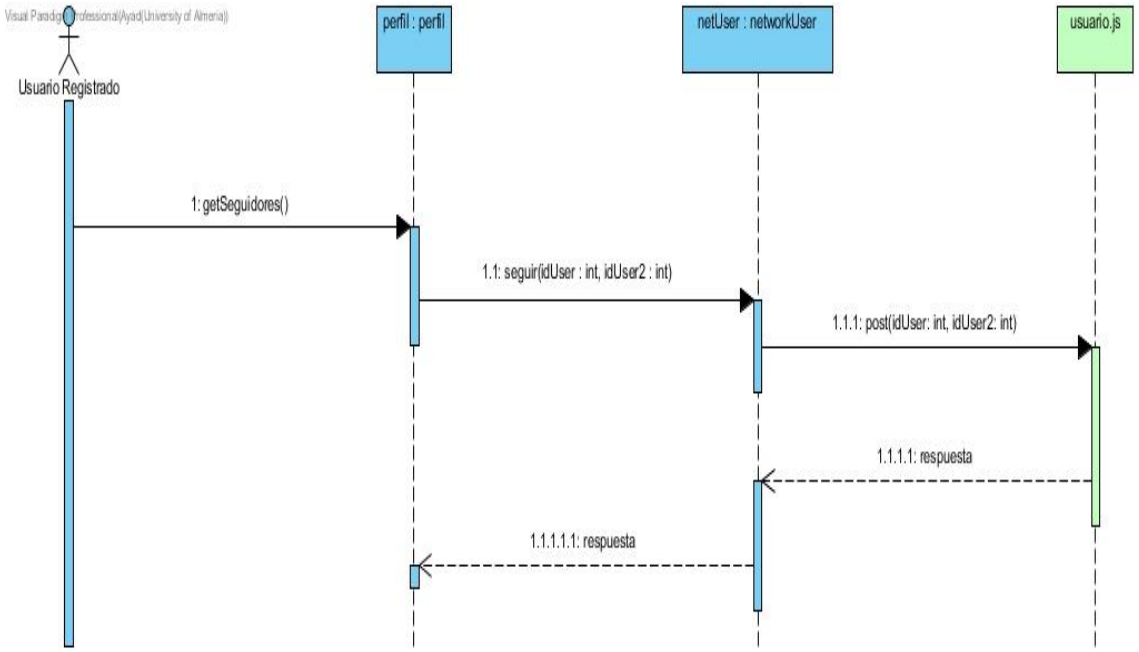
4.6.14. Registro



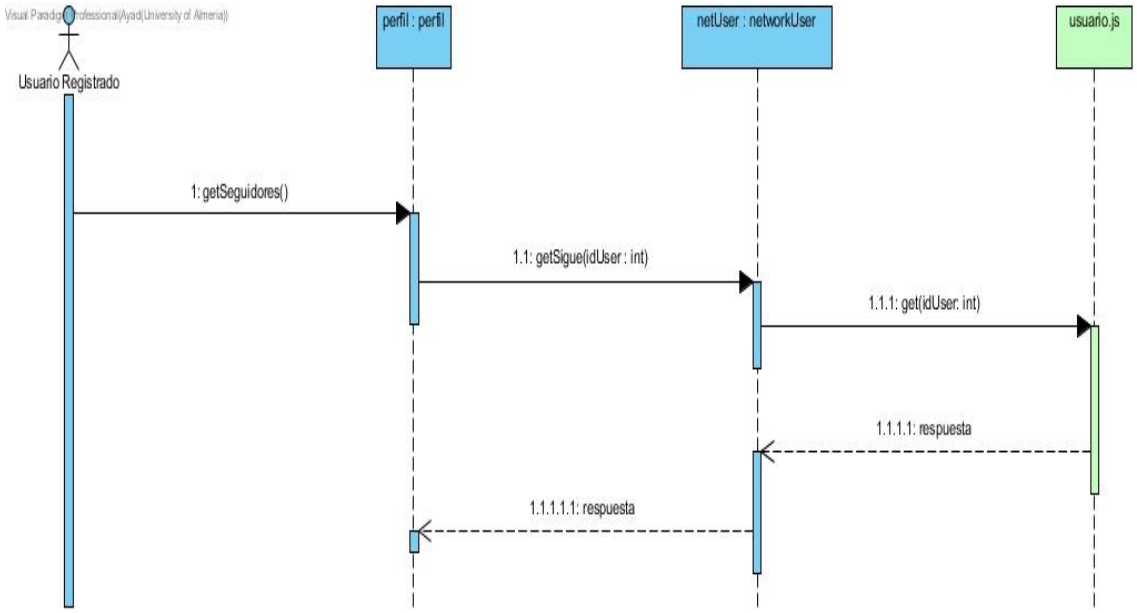
4.6.15. Cargar seguidores



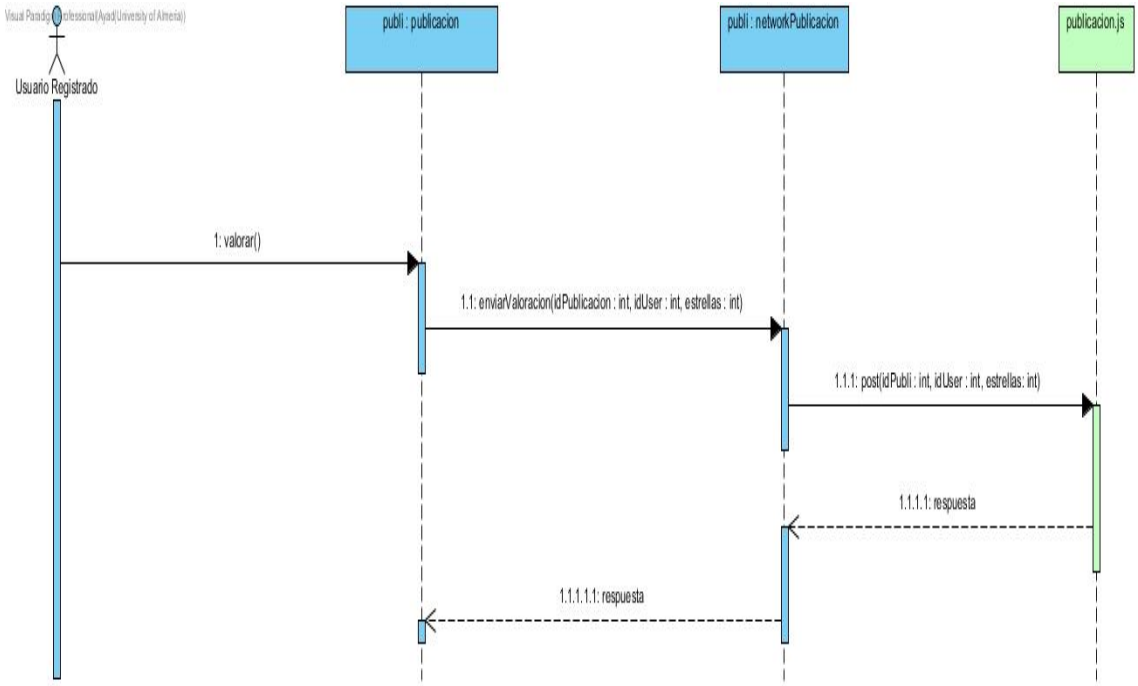
4.6.16. Seguir usuario



4.6.17. Cargar seguidos



4.6.18. Valorar publicación



4.7. Aspectos destacados del sistema y la implementación

4.7.1. Aspectos de la API *Rest*

Las peticiones a la API *Rest* en *Node.js* [18] se realizará a la dirección http://10.0.2.2 en el puerto 3000. El intercambio de información se realizará mediante el formato JSON [20].

La conexión a la API será en local del PC y se conectará a la base de datos de MySQL [19] con el nombre mydb. Para acceder a esta base de datos se deberá acceder con el usuario y contraseña de la base de datos.

```
1   const mysql = require('../node_modules/mysql');
2
3   const mysqlConn = mysql.createConnection({
4       host: 'localhost',
5       user: 'user',
6       password: 'password',
7       database: 'mydb'
8   });
9
10  mysqlConn.connect((function (err){
11      if (err) {
12          console.log(err);
13          return;
14      } else {
15          console.log("Conectado a la base de datos mydb");
16      }
17  }));
18
19  module.exports = mysqlConn;
```

Figura 41: Conexión a la base de datos mydb

La API *Rest* se encuentra dividida en siete rutas principales:

- Usuario.js
- Dinero.js
- Empresa.js
- Imagen.js
- Logo.js
- Notificacion.js
- Publicacion.js

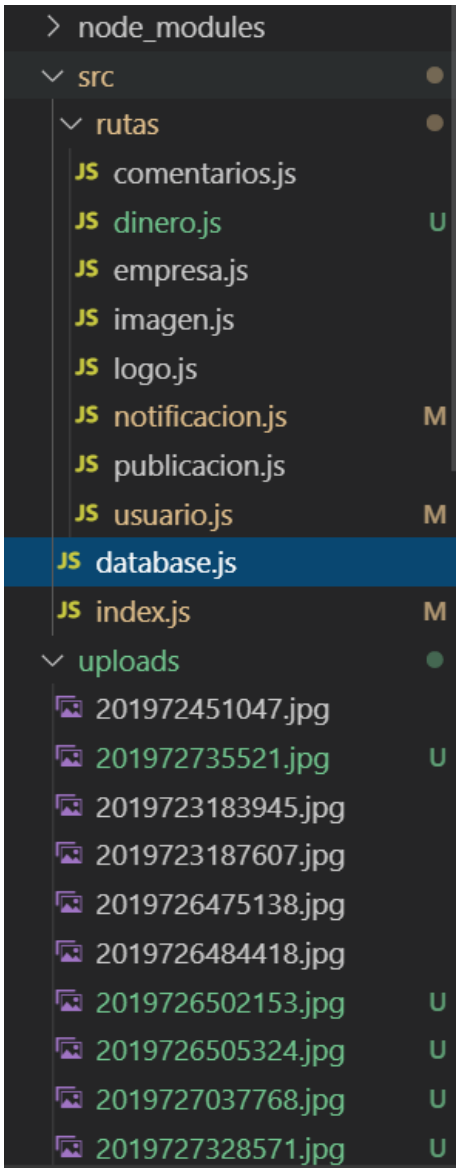


Figura 42: Rutas e imagenes de la API Rest

En la Figura 42 podemos observar las siete rutas principales de nuestra API *Rest* en *Node.js* [18]. Las imágenes de las publicaciones y usuarios se guardan en el directorio *uploads*, y en la tabla de la base de datos se guarda la ruta de la imagen que accede a esa carpeta con la ruta del nombre de la imagen.

```
now.year.toString()+now.month.toString()+now.day.toString()+now.minute.toString()+now.second.toString()+now.millisecond.toString()+".jpg";
```

Figura 43: Nombre de las imágenes

A la hora de insertar imágenes en la carpeta de *uploads* se le asignará un nombre único para evitar nombres repetidos, y la manera de conseguir esto será añadiendo el nombre con el año + mes + día + minuto + segundo + milisegundo.

4.7.2. Aspectos de la base datos

Para levantar la base de datos en *MySQL* [19] y *PhpMyAdmin* se utilizará *Docker* [21].

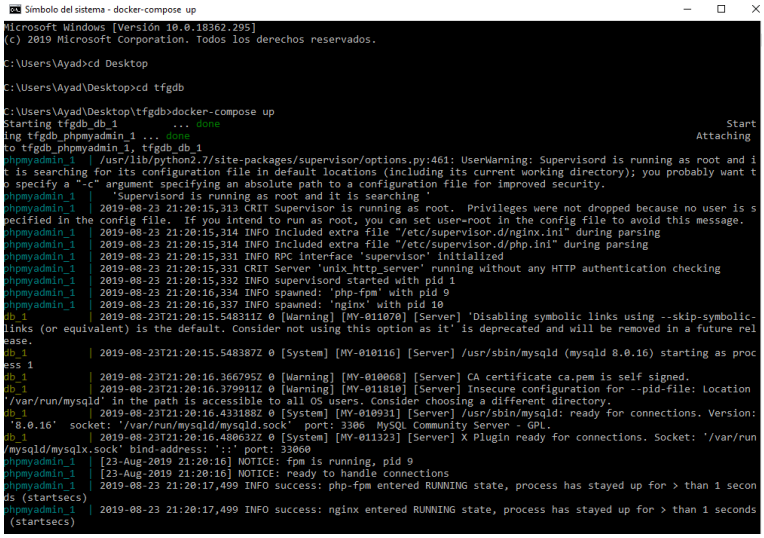


Figura 44: Línea de comandos de Windows desplegando MySQL y PhpMyAdmin

En la Figura 44 podemos observar que con una sola línea de comando se puede desplegar *MySQL* [19] y *PhpMyAdmin*.

La configuración necesaria está en un fichero llamado *docker-compose.yml*, en el que se debe especificar las imágenes que serán utilizados por nuestros contenedores, puertos expuestos, volúmenes, etc.

docker-compose.yml

```yaml
version: '3'
services:
  db:
    image: mysql:latest
    ports:
    - "3306:3306"
    command: --character-set-server=utf8mb4 --collation-server=utf8mb4_bin
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_USER: user
      MYSQL_PASSWORD: password
      MYSQL_DATABASE: mydb

  phpmyadmin:
    image: phpmyadmin/phpmyadmin:latest
    ports:
    - "5000:80"
```

Figura 45: Fichero docker-compose.yml

En la Figura 45 se puede observar que hemos definido dos servicios (*db* y *phpmyadmin*). En este fichero se le está indicando que conecte *MySQL* [20] en el puerto 3306 y *phpMyAdmin* en el puerto 5000.

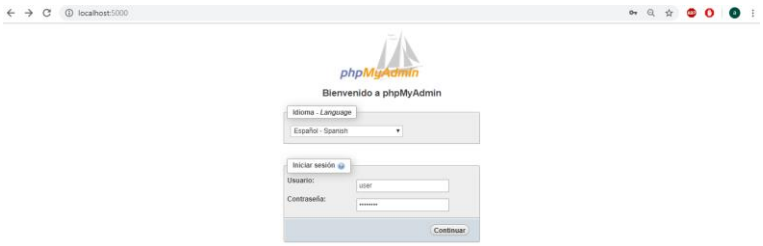


Figura 46: PhpMyAdmin

Al ingresar en el navegador con el puerto 5000 nos encontramos con la pantalla que se muestra en la Figura 46. Al introducir el nombre de usuario y la contraseña accedemos a toda la información de la base de datos, pudiendo administrar con facilidad toda la información de la base datos.

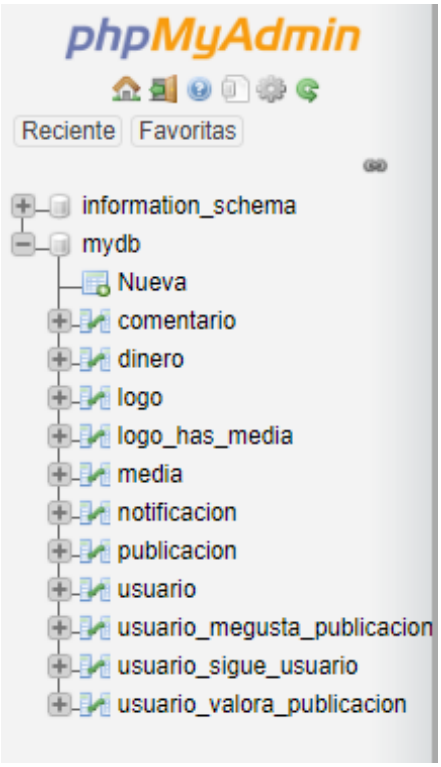


Figura 47: PhpMyAdmin tablas de la base de datos

En la Figura 47 podemos observar todas las tablas que contendrá nuestra base de datos. Está herramienta nos facilitará toda la información de la base de datos e incluso nos permitirá gestionar toda esa información.

5. Resultados

En este capítulo, se mostrará el funcionamiento de la aplicación y se mostrarán detalles importantes de la implementación.

5.1. Registro

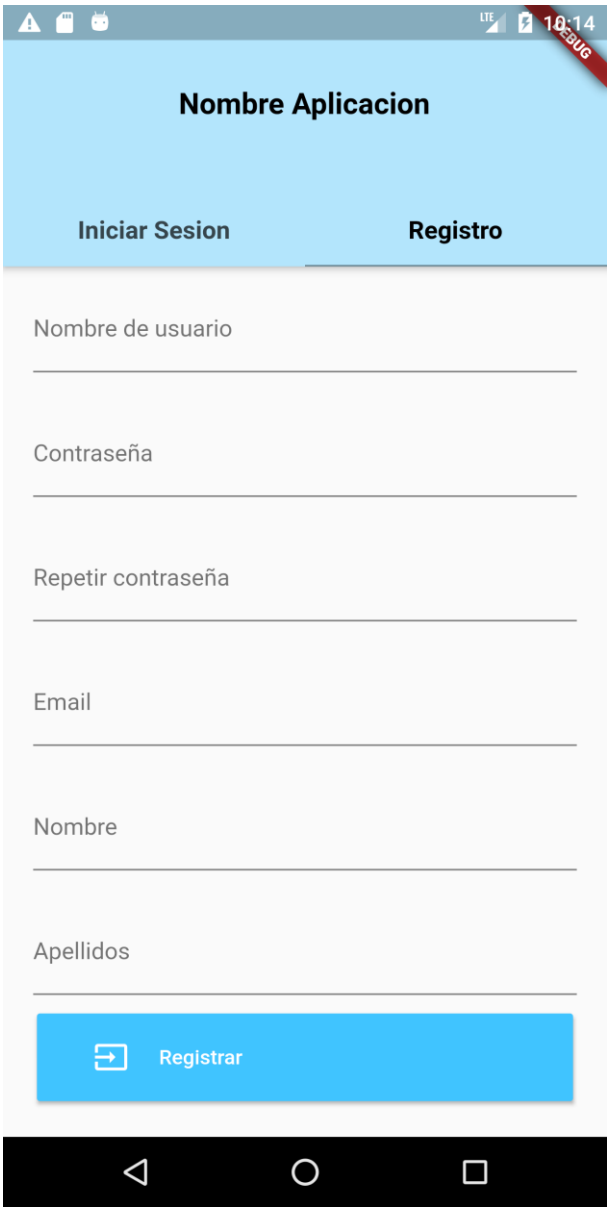


Figura 48: Ventana de registro

Al iniciar la aplicación nos encontraremos con la ventana que se muestra en la Figura 48. En la parte superior de la aplicación nos encontramos con una barra de pestañas donde podemos elegir la opción deseada (iniciar sesión o registrarse). En este caso se utilizará la pestaña de registro.

Para completar el registro debemos añadir todos los datos indicados a los campos de texto.

Una vez completado el registro, se enviará los datos a la API, pero para conseguir seguridad en la clave se encriptará con el paquete *dbcrypt* antes de su envío.

```
Future<String> regUser(String nombre, String apellido, String email, String pass, String username, String image) async {
  String password = DBCrypt().hashpw(pass, DBCrypt().gensalt());
  Usuario userReg = Usuario(null, nombre, apellido, email, password, username, image, 0, 1);

  var response = await _dio.post(this._dir, data: userReg.toJson(),);
  return response.data;
}
```

Figura 49: Post a la API Rest del registro de usuario

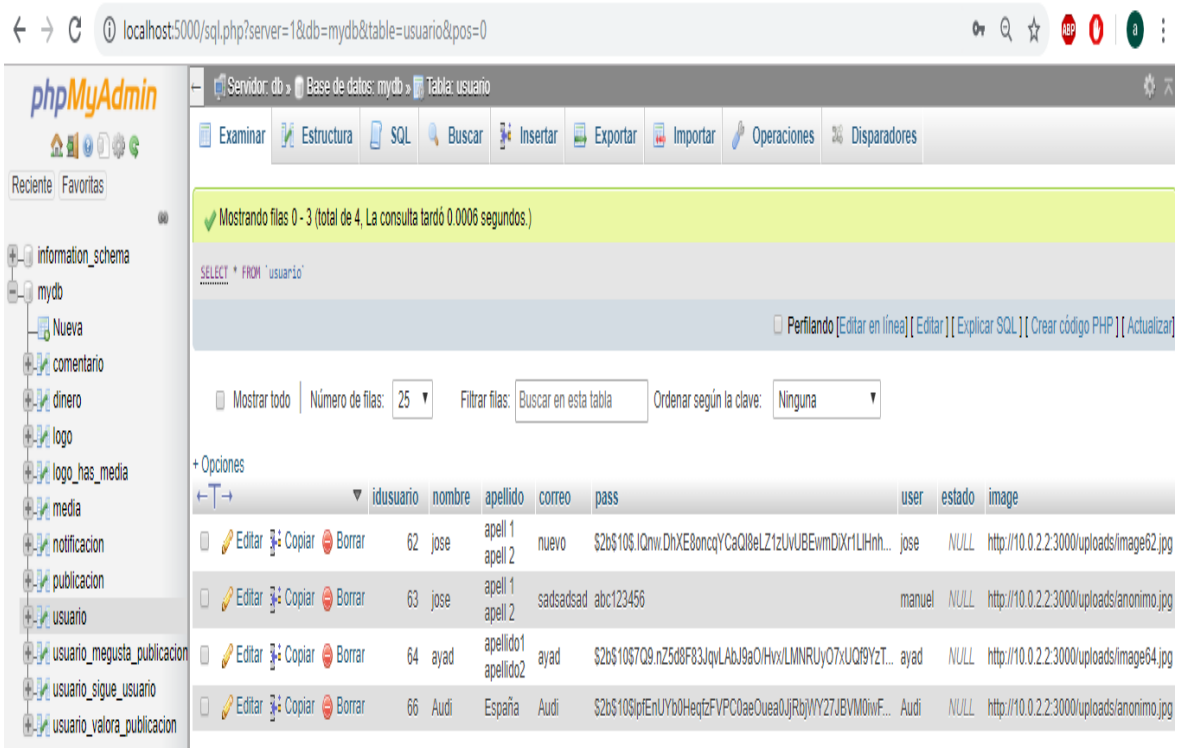


Figura 50: Información del usuario en phpMyAdmin (Comprobación de contraseña cifrada)

Como se puede observar en la Figura 49, la contraseña se envía cifrada, cuestión que podemos comprobar en la Figura 50.

5.2. Iniciar sesión

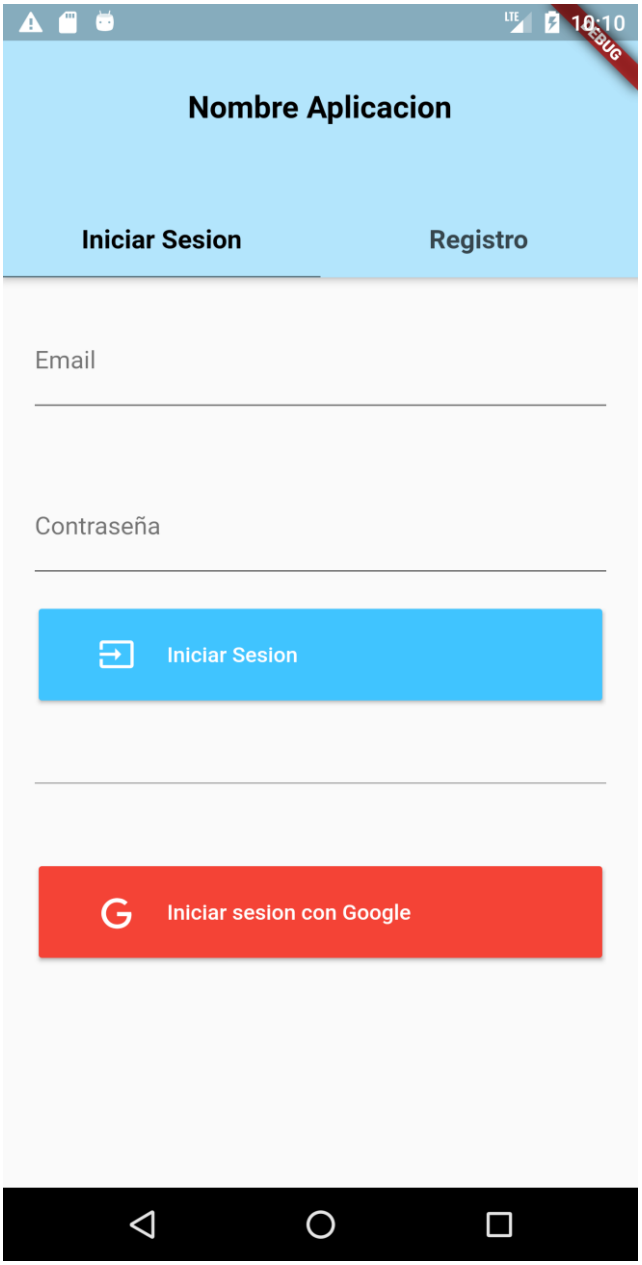


Figura 51: Ventana de inicio de sesión

Para iniciar sesión en la aplicación se deberá añadir en el campo de texto el correo electrónico y la contraseña.

5.3. Página principal y menú lateral

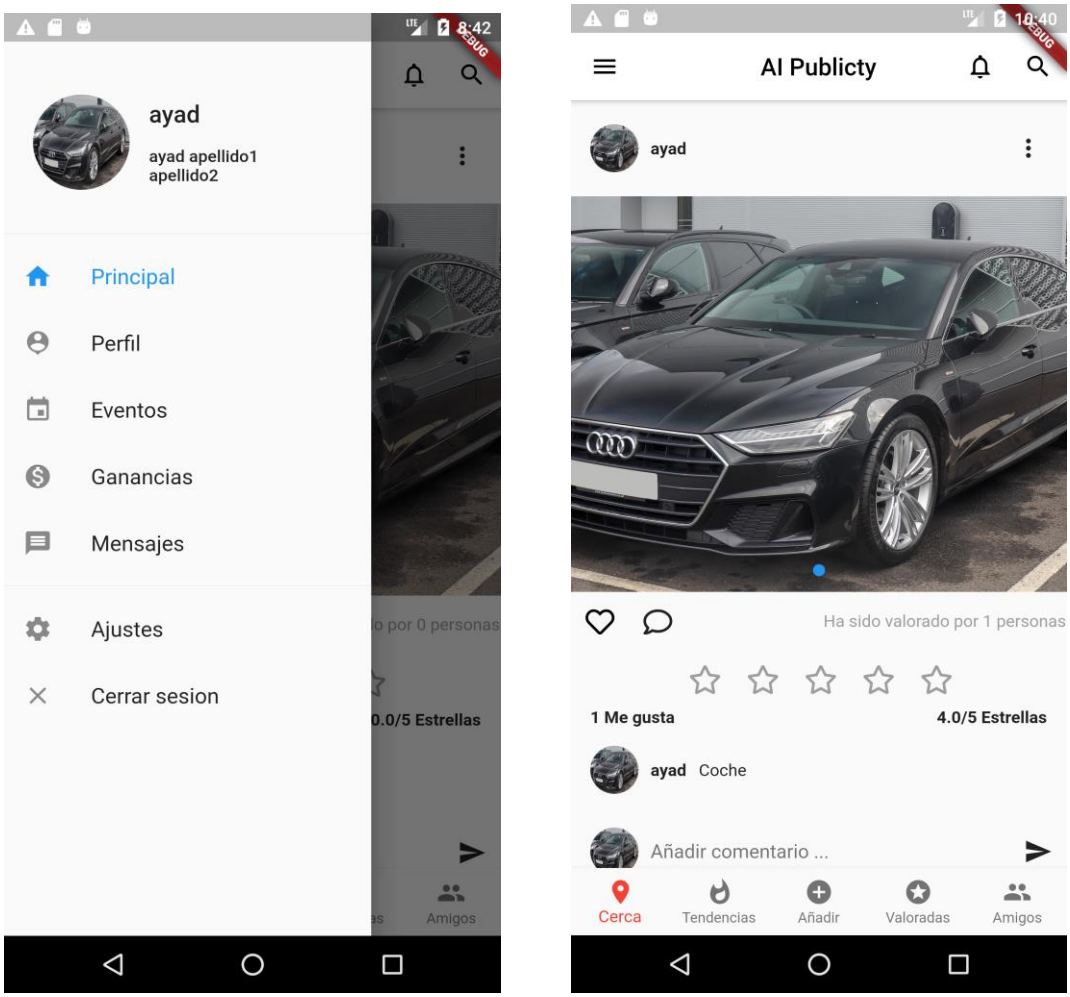


Figura 52 y 53: Menú lateral y página principal

En la Figura 52 se muestra el menú lateral de la aplicación, que utilizaremos para poder cambiar de ventana con gran facilidad en la aplicación.

En la Figura 53 se muestra la página principal de la aplicación. En esta página principal tenemos acceso a una barra inferior con cinco pestañas donde podemos intercambiar entre sus distintos contenidos: Cerca (publicaciones cercanas), Tendencias (publicaciones nuevas), Añadir (añadir una publicación nueva), Valoradas (publicaciones mejor valoradas) y Amigos (publicaciones de amigos).

5.4. Buscar usuario

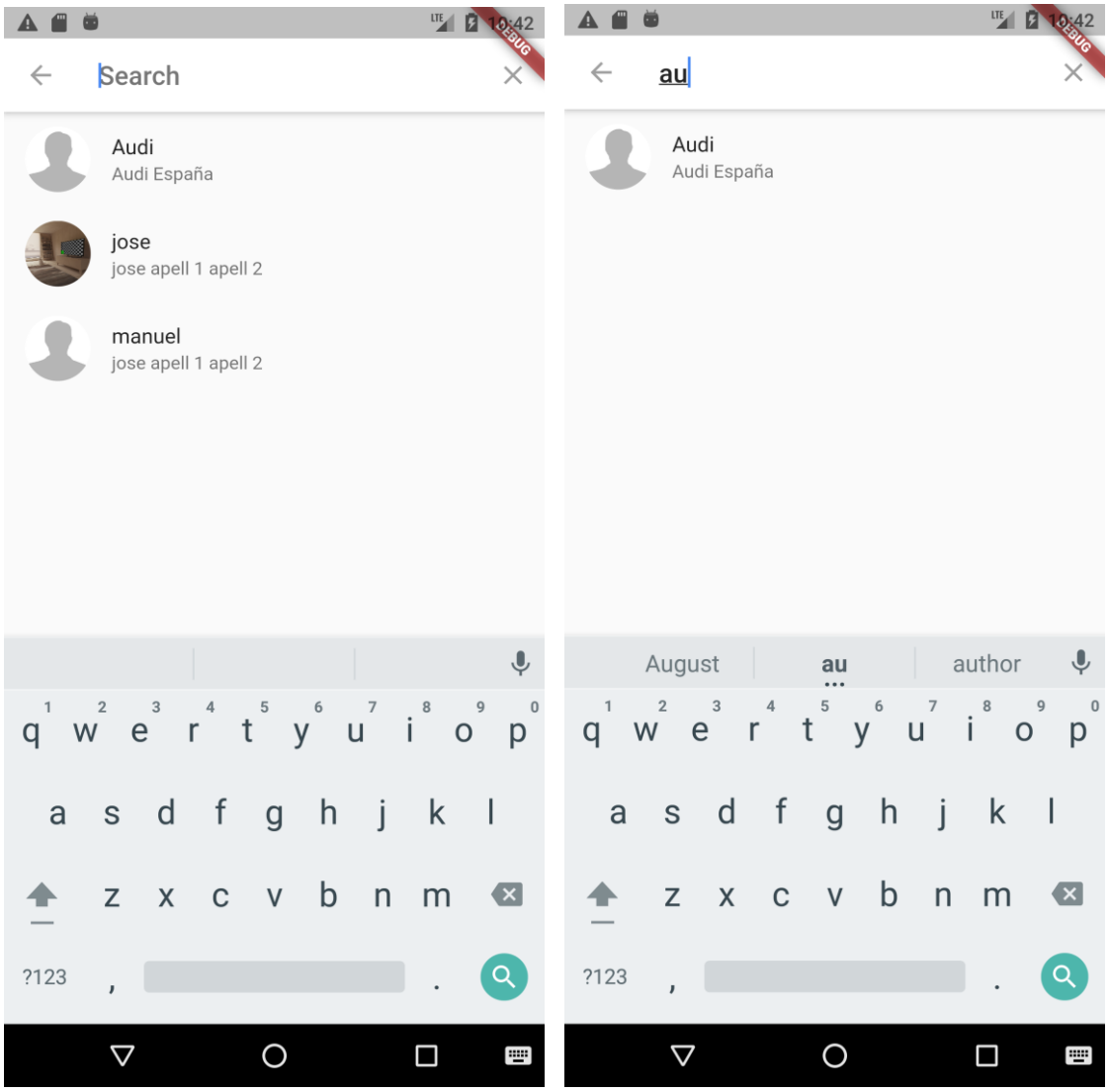
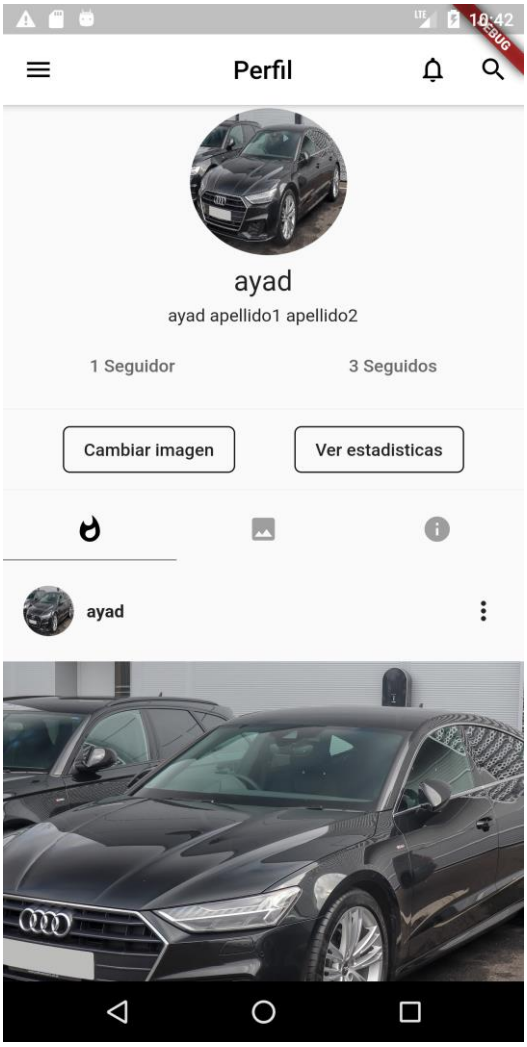


Figura 54 y 55: Buscar usuario y buscando por nombre de usuario

En la Figura 53 podemos encontrar en la esquina superior derecha un icono para la búsqueda de usuario. Al presionar en el icono nos encontramos con la ventana de la Figura 54, búsqueda de usuarios. En la Figura 55 podemos observar que cuando insertamos el nombre de usuario filtramos su búsqueda de usuarios encontrando el usuario deseado.

5.5. Visualizar perfil propio



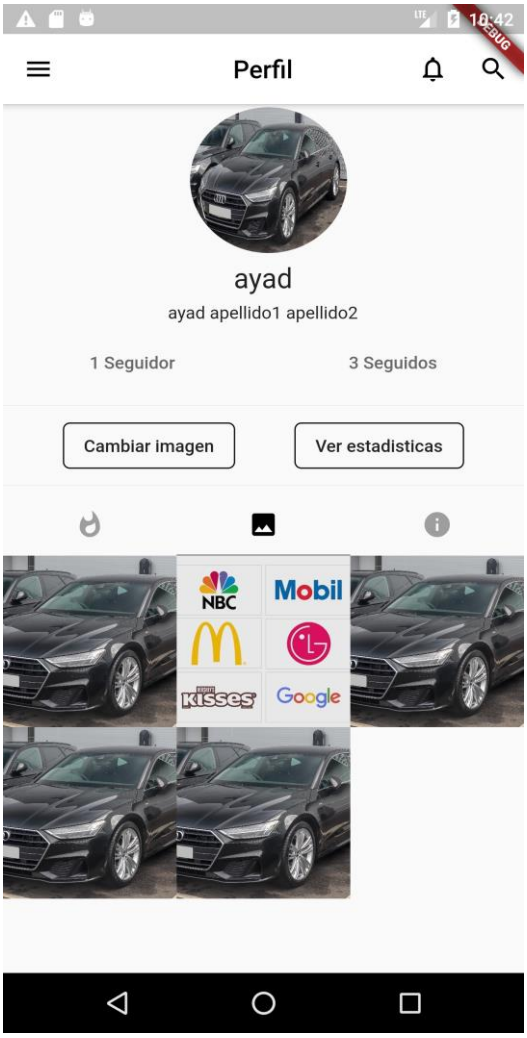


Figura 56 y 57: Perfil propio pestaña publicación y perfil propio pestaña imágenes

En las Figuras 56 y 57 podemos visualizar el perfil propio del usuario. En la Figura 56 nos encontramos en la pestaña de las publicaciones del usuario. En esta pestaña se encuentran todas las publicaciones que ha subido el usuario. En la Figura 57, como podemos observar, nos encontramos en la pestaña de imágenes, en la que se muestran todas las imágenes publicadas por el usuario.

5.6. Cambiar imagen de perfil

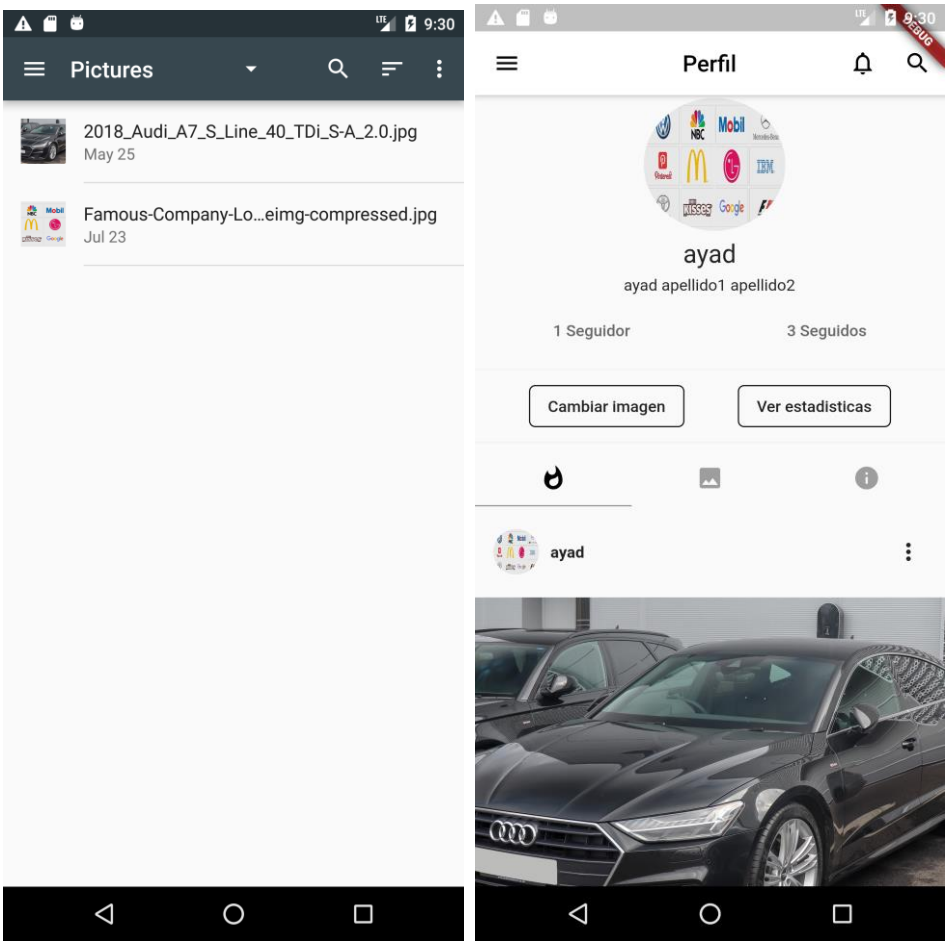


Figura 58 y 59: Seleccionar imagen e imagen de perfil cambiado

Al presionar el botón cambiar imagen en el perfil del usuario se nos abrirá la ventana para seleccionar una imagen (ver Figura 58). Al seleccionar la imagen deseada podemos comprobar que se ha actualizado correctamente (Figura 59).

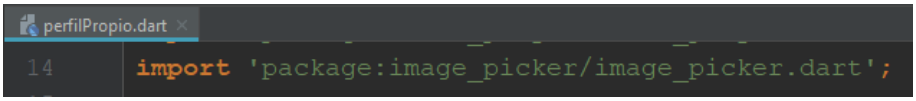


Figura 60: Paquete image picker

Tal y como se observa en la Figura 60, se ha utilizado el paquete *image_picker* para selección de una imagen del dispositivo.

```
Future getImage() async {
  var image = await ImagePicker.pickImage(source: ImageSource.gallery);
```

Figura 61: Programación para seleccionar una imagen

En la Figura 61 se muestra cómo se almacena en la variable imagen guardará la imagen seleccionada (Figura 58).

5.7. Perfil usuario y seguir usuario

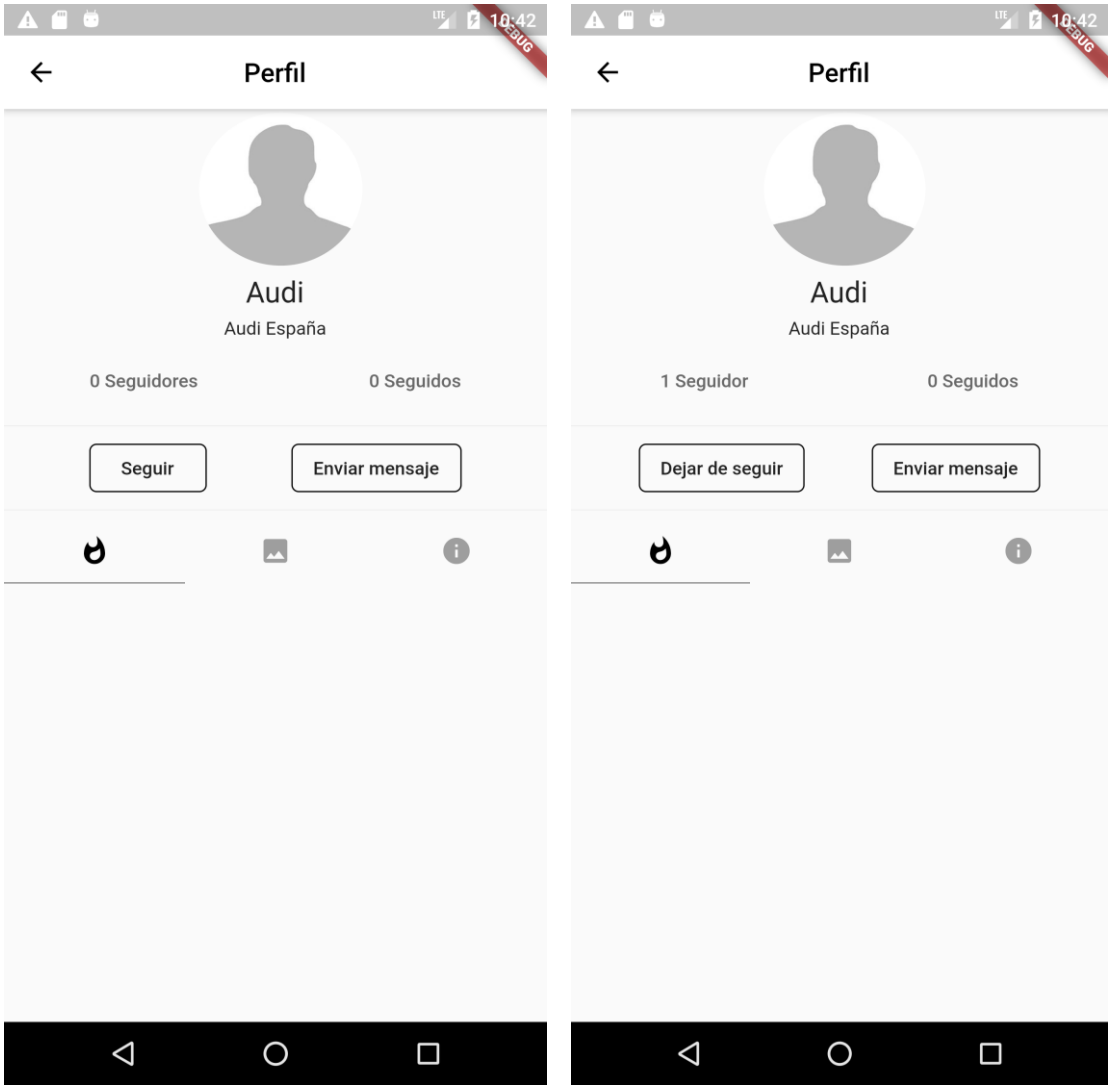


Figura 62 y 63: Perfil usuario sin seguidores y perfil usuario con seguidor

En las Figuras 62 y 63 nos encontramos con la vista de perfil de usuario del sistema; ahí es donde existe la opción de poder seguir al usuario. En la Figura 62 nos encontramos con la captura de pantalla del usuario sin ser seguido; en la Figura 63 se muestra la indicación de que seguimos al usuario.

Al seleccionar el botón de seguidores (o seguidos) se abrirá una nueva ventana con el listado de usuarios que le siguen (o que sigue). De esta manera, podemos visualizar la información de seguidores (o de seguidos) tanto del usuario propietario como de cualquier usuario del sistema.

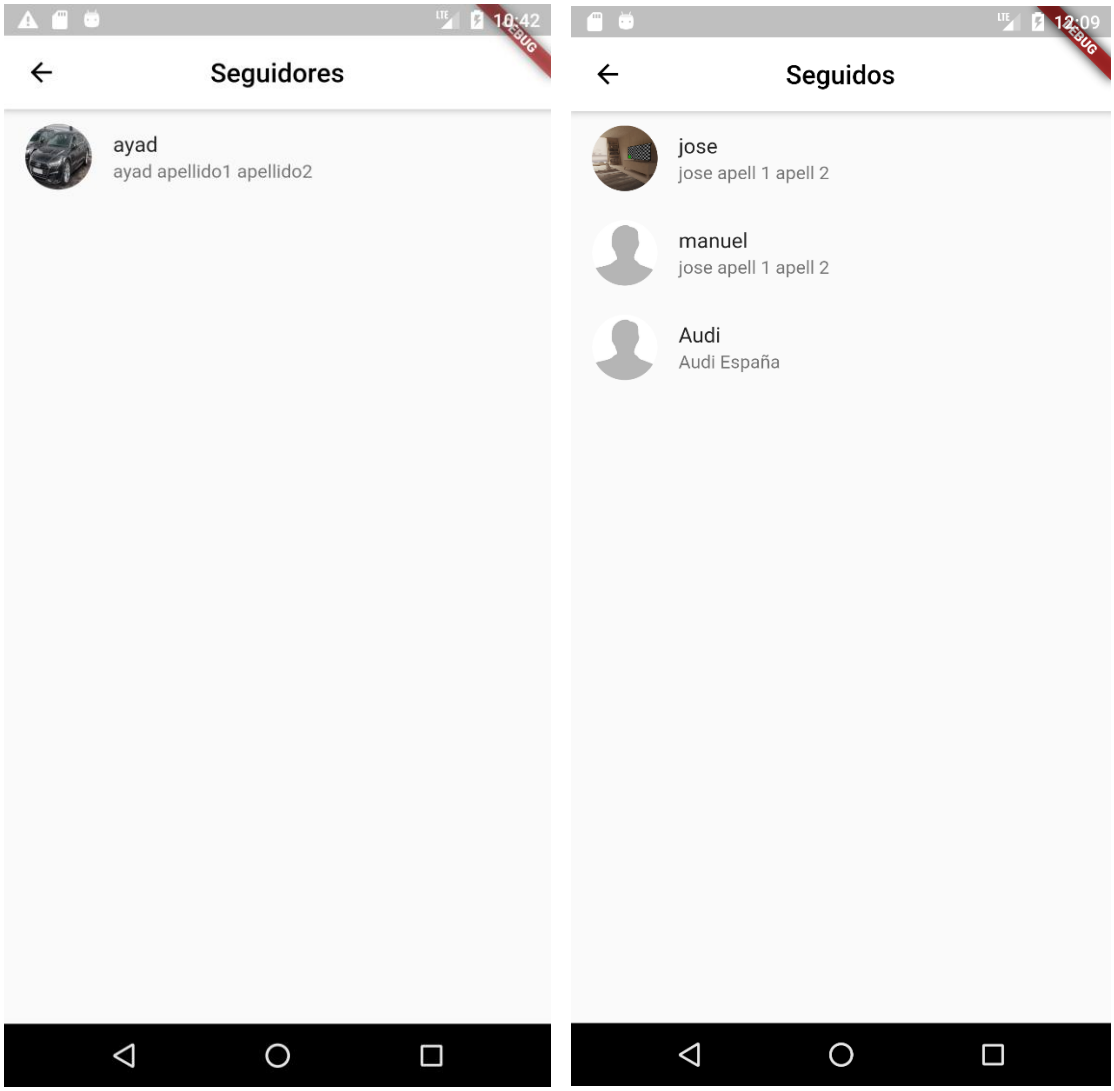


Figura 64 y 65: Listado de seguidores y listado de seguidos

En la Figura 64 se visualiza la lista de usuarios que siguen a la empresa Audi

Al ingresar en el perfil propio del usuario Ayad, seleccionamos el listado de seguidos, abriéndose una ventana con todos los usuarios seguidos. En la Figura 65 se encuentra el listado de seguidos de Ayad; podemos comprobar que se ha seguido correctamente al usuario empresa Audi.

5.8. Ajustes

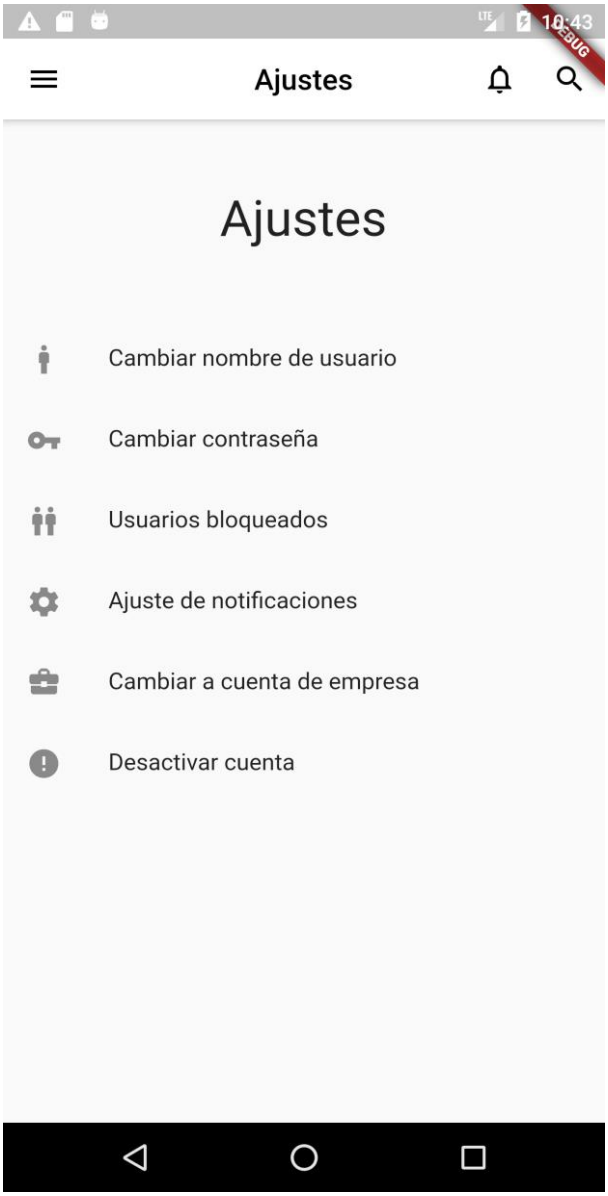


Figura 66: Ventana de ajustes

En esta ventana podemos realizar cambios a la cuenta del usuario. Contiene seis secciones con diferentes usos:

- Cambiar nombre de usuario: en esta sección se abrirá una nueva ventana con la opción de cambiar el nombre de usuario, el nombre y los apellidos.
- Cambiar contraseña: en esta sección podemos cambiar la contraseña del usuario. Al presionar en esta sección se abrirá una ventana donde debemos ingresar la contraseña antigua e ingresar la contraseña nueva del usuario.
- Usuarios bloqueados: en esta sección se abrirá el listado de usuarios que han sido bloqueados por el usuario.

- Ajuste de notificaciones: en esta ventana vamos a tener la posibilidad de configurar las notificaciones del sistema.
- Cambiar a cuenta de empresa: podemos cambiar la cuenta a una cuenta de empresa. Debemos confirmar la acción de cambiar la cuenta al ser presionada.

A continuación, se muestra sus respectivas capturas de pantalla de las distintas secciones de los ajustes del sistema.

5.8.1. Cambiar información de usuario

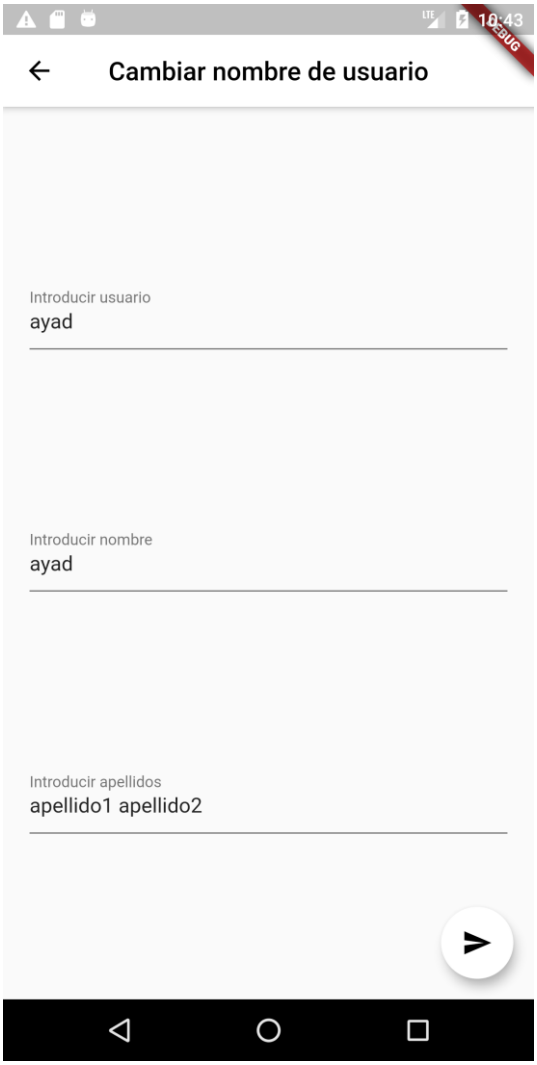


Figura 67: Cambiar datos personales de usuario

Al seleccionar la acción *Cambiar nombre de usuario* (Figura 66) se abrirá la ventana de la Figura 67. Como podemos observar, la ventana nos completa los datos del usuario en los campos de texto, ya que le puede interesar al usuario tener estos campos completados y sólo realizar cambios al campo que desea cambiar.

5.8.2. Cambiar contraseña de usuario

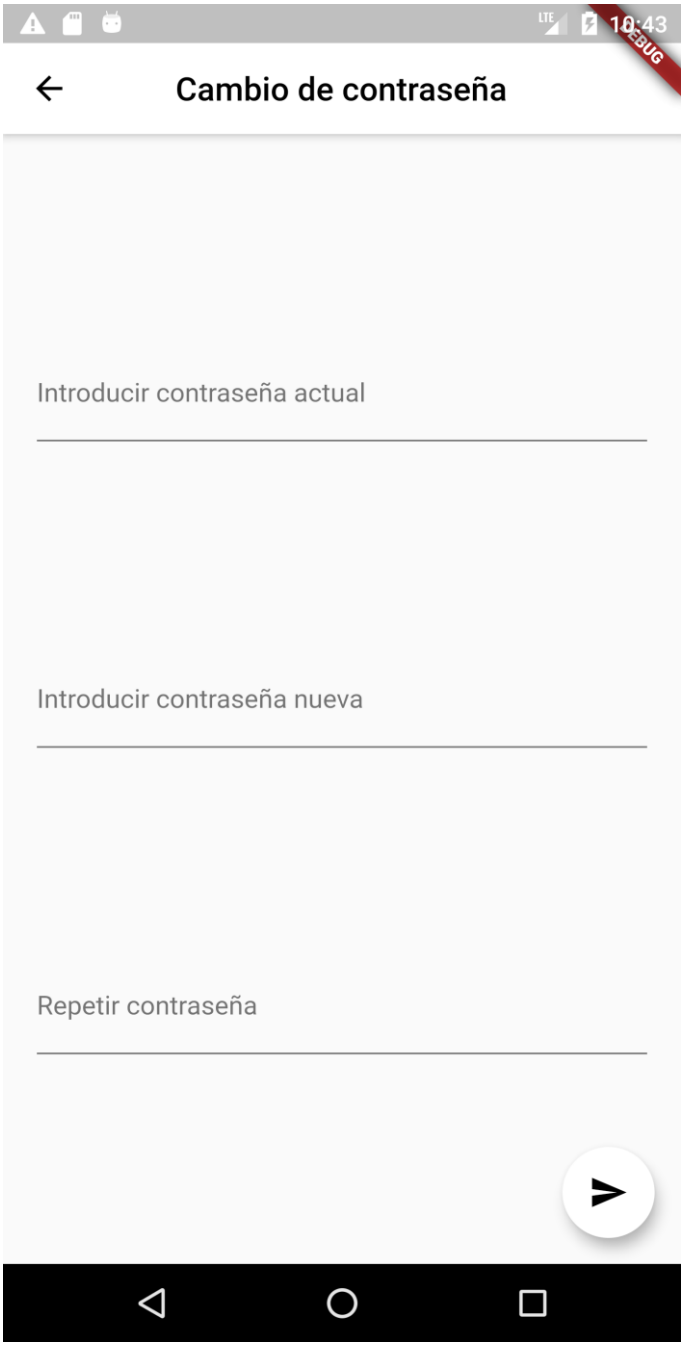


Figura 68: Cambiar contraseña de usuario

Al seleccionar la acción *Cambiar de contraseña* se abrirá la ventana de la Figura 68. A continuación, se mostrarán las distintas alertas al intentar cambiar la contraseña del usuario.

- El primer paso es comprobar que todos los campos de texto estén rellenos para cambiar la contraseña, de lo contrario nos mostrará el mensaje que aparece en la Figura 69.

- El segundo paso es comprobar que la contraseña antigua coincida con la contraseña del usuario. En caso contrario se mostrará la alerta de la Figura 71. Con esta comprobación el sistema realizará el cambio de contraseña.
- El tercer paso es comprobar que la contraseña contiene al menos un mínimo de 8 caracteres. De lo contrario mostrará la alerta de la Figura 70.
- El último paso es comprobar que los dos últimos campos de texto sean iguales; de lo contrario se mostrará el mensaje de la Figura 72. El sistema comprobará estos dos últimos campos para conocer con seguridad la nueva contraseña del usuario, ya que el usuario se puede equivocar al escribir su contraseña nueva.

Cuando el sistema consigue cambiar la contraseña del usuario avisará con el mensaje de la Figura 73.

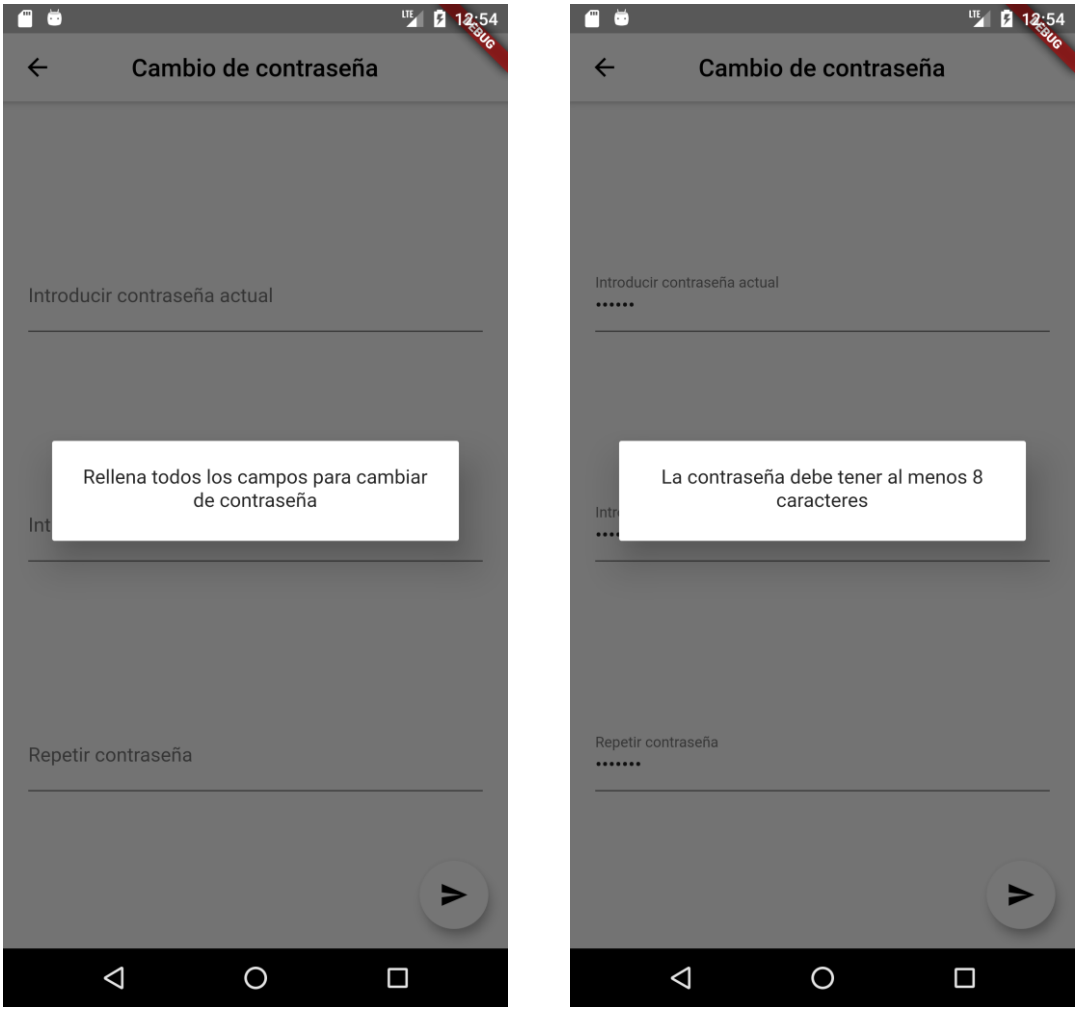


Figura 69 y 70: Alerta rellenar todos los campos y alerta mínimo de caracteres

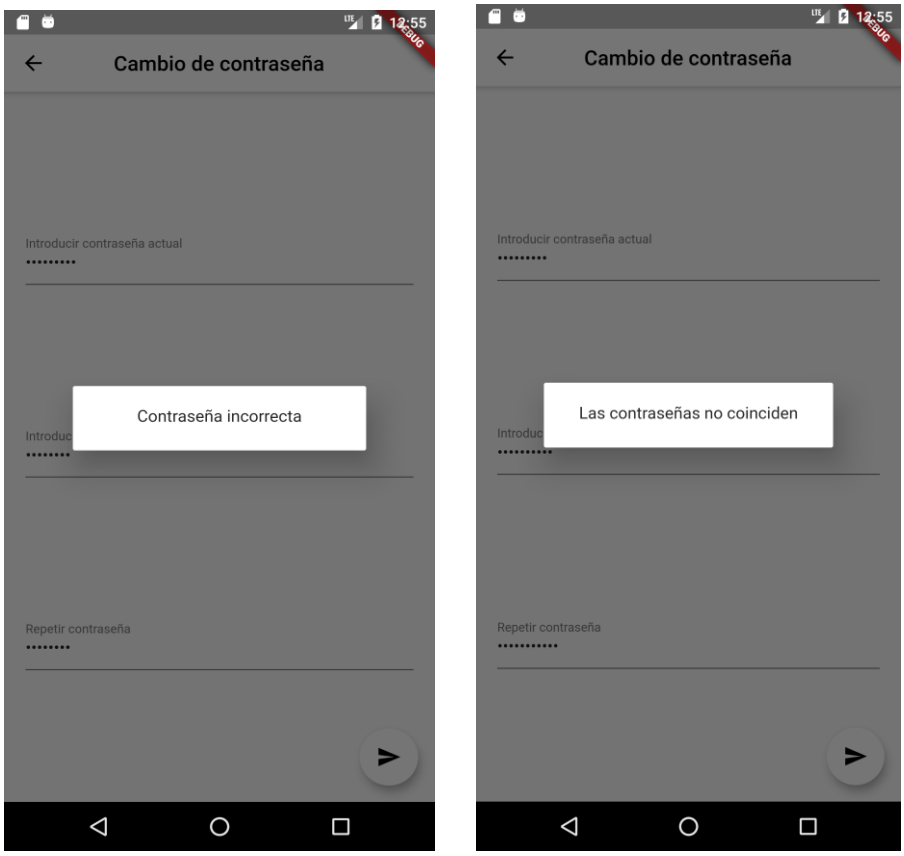


Figura 71 y 72: Alerta de contraseña incorrecta y alerta de contraseña no coincide

Figura 73: Contraseña cambiada correctamente

5.8.3. Cambiar cuenta de usuario normal a empresa

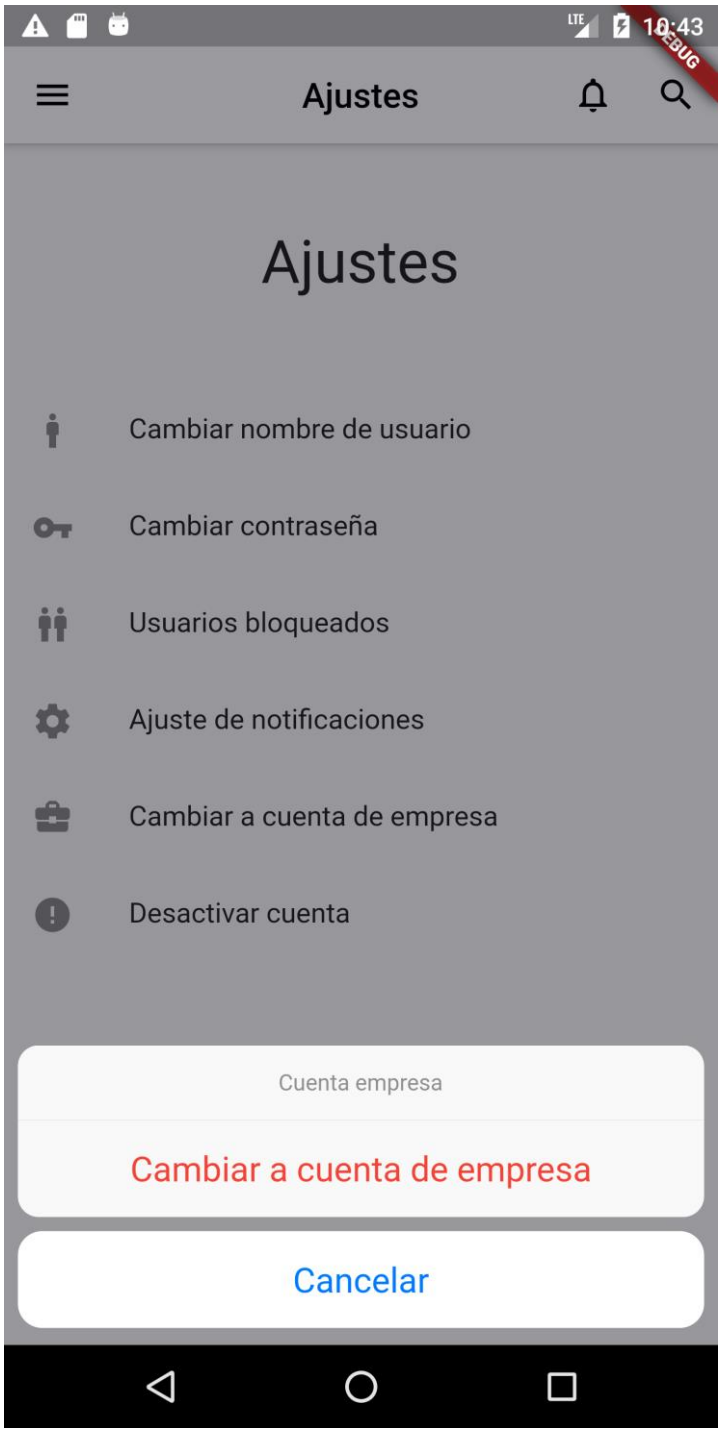


Figura 74: Ventana cambiar a cuenta de empresa

En la ventana de ajustes de la aplicación nos encontraremos con la acción de cambiar la cuenta de usuario normal a una cuenta de empresa. Al presionar la acción de cambiar la cuenta de empresa nos solicitará una comprobación de esta acción. Una vez seleccionado, la cuenta será cambiada a cuenta de empresa donde se obtendrá diferentes funcionalidades dentro del sistema.

5.9. Valorar y dar me gusta a una publicación

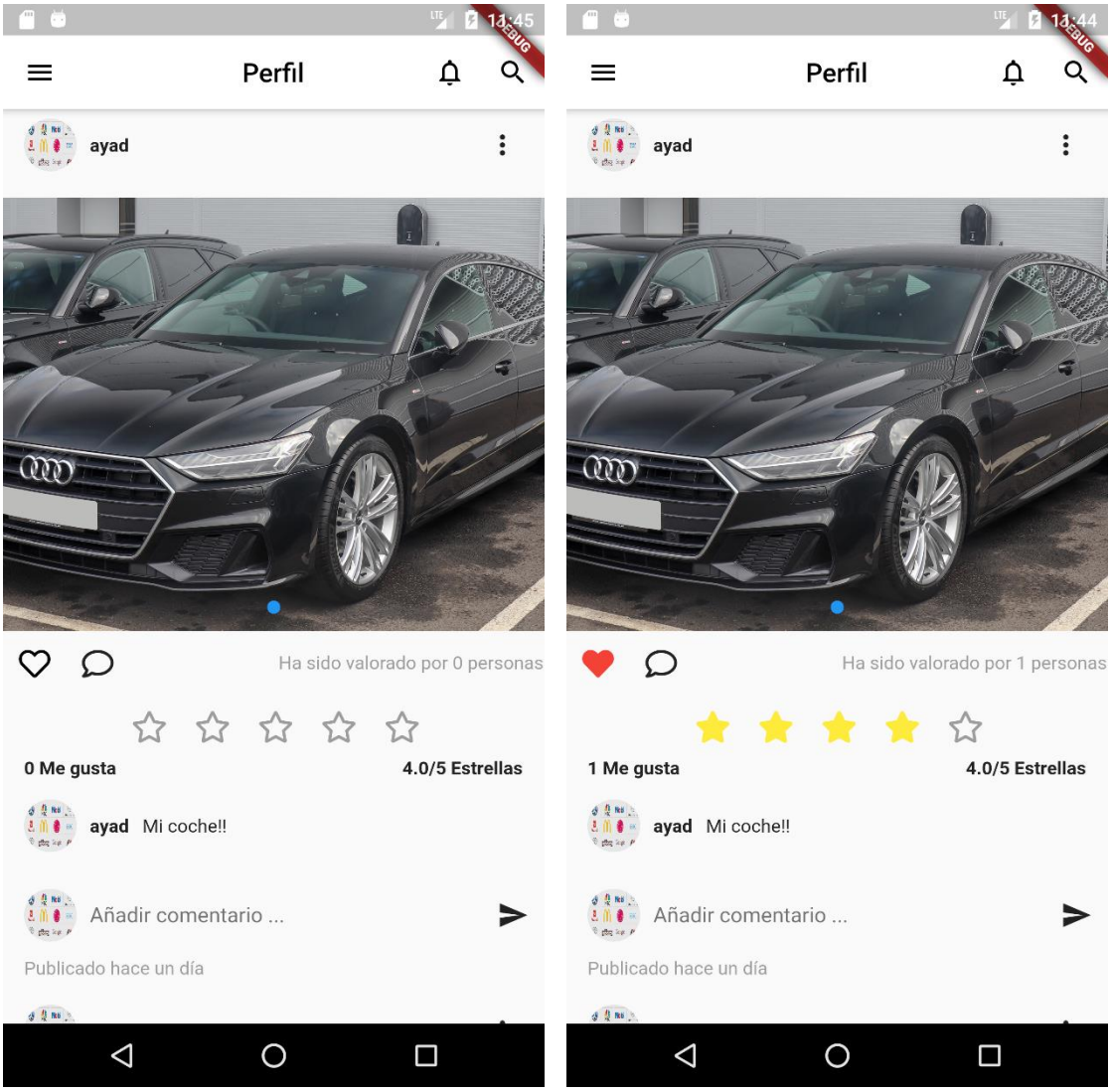


Figura 75 y 76: Publicación sin valoración y publicación con valoración

Todos los usuarios del sistema podrán valorar y dar a me gusta a una publicación. Contiene las siguientes características:

- Las valoraciones se miden de 1 a 5 y se muestra la media total de los usuarios que han valorado la publicación.
- El usuario podrá quitar la valoración presionando la estrella final marcada. Por ejemplo, en el caso de la figura 76 el usuario deberá presionar la cuarta estrella para quitar su puntuación.
- El usuario podrá dar o quitar me gusta pulsando el icono del corazón.
- La publicación mostrará el número de me gusta y personas que han valorado la publicación como se muestra en las Figura 75 y 76.

5.10. Añadir publicación

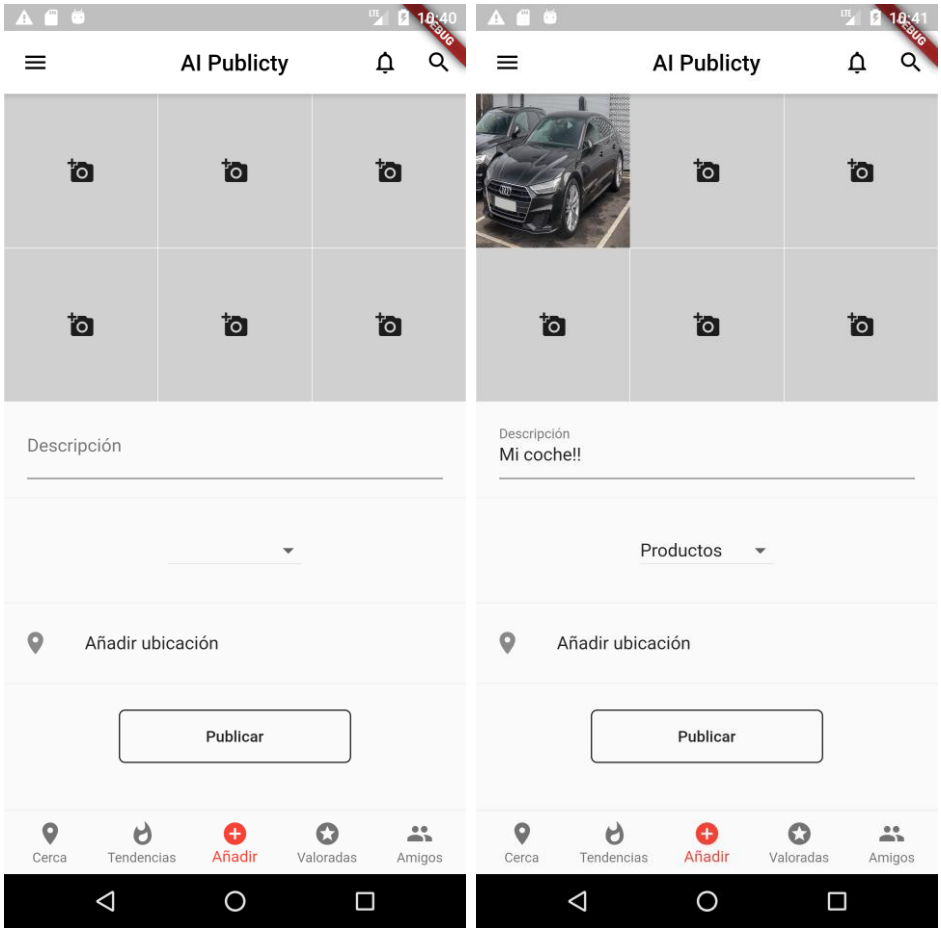


Figura 77 y 78: Ventana añadir publicación y ventana añadir publicación con datos

En la página principal de la aplicación se encuentra una barra de pestañas inferior en el cual tendrá la opción de subir una nueva publicación. Al entrar en esta sección se encontrará la posibilidad de añadir varias imágenes, una descripción, tipo de publicación y ubicación (ver Figuras 76 y 77).

Al añadir una publicación el sistema buscará en la imagen algún logotipo; en caso afirmativo, se informará a la empresa de la publicidad. Es posible encontrar más de un logotipo en una imagen. Con esta publicidad el usuario recibirá una recompensa cuyo valor dependerá de su ranking en la aplicación.

Para el análisis de logotipos se utiliza *Cloud Vision* [2] que utiliza una base de datos con miles de logotipos para la identificación de las marcas. Se deberá realizar una petición POST proporcionando el cuerpo de la solicitud a la dirección:

https://vision.googleapis.com/v1/images:annotate?key=YOUR_API_KEY

En la dirección se deberá incorporar nuestra clave para poder utilizar esta API.

En el cuerpo de la solicitud debemos añadir la imagen que se desea analizar en base 64 en la variable *content*. El cuerpo de la solicitud debe ser como el siguiente ejemplo:

```
{
 "requests": [
  {
   "image": {
    "content": "/9j/7QBEUGhvdG9zaG9...base64-encoded-image-content...fXNWzvDEeYxxxzj/Coa6Bax//Z"
   },
   "features": [
    {
     "type": "LOGO_DETECTION"
    }
   ]
  }
 ]
}
```

Al realizar la petición POST con la imagen que deseamos analizar, recibiremos una respuesta en formato JSON con el siguiente cuerpo:

```
{
 "responses": [
  {
   "logoAnnotations": [
    {
     "mid": "/m/045c7b",
     "description": "Google",
     "score": 0.32291126,
     "boundingPoly": {
      "vertices": [
       {
        "x": 63,
        "y": 18
       },
```

```
          {
           "x": 123,
           "y": 18
          },
          {
           "x": 123,
           "y": 38
          },
          {
           "x": 63,
           "y": 38
          }
        ]
       }
      }
    ]
   }
 ]
}
```


En la variable *Description* recibimos el nombre del logotipo encontrado.

A continuación, se citará los paquetes utilizados en el desarrollo de esta ventana:

- En la selección de imágenes se ha utilizado el paquete *multi_image_picker*, tal y como se muestra en la Figura 79. Con este paquete tenemos la posibilidad de seleccionar varias imágenes. El usuario deberá aceptar los permisos necesarios para poder seleccionar las imágenes de su teléfono.
- La aplicación utilizará el paquete *geolocator* para encontrar la ubicación del usuario. El usuario deberá aceptar los permisos necesarios para usar la ubicación de su dispositivo.

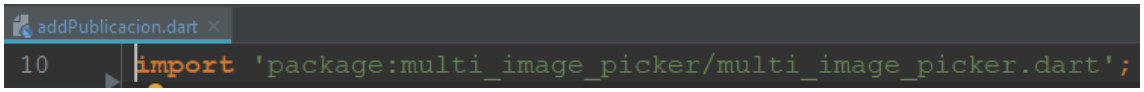


Figura 79: Paquete multi_image_picker

```
addPublicacion.dart
10  import 'package:geolocator/geolocator.dart';
```

Figura 80: Paquete geolocator

Al presionar una imagen en la ventana añadir publicación, el sistema le permitirá eliminar esa imagen o eliminar todas las imágenes (ver Figura 81).

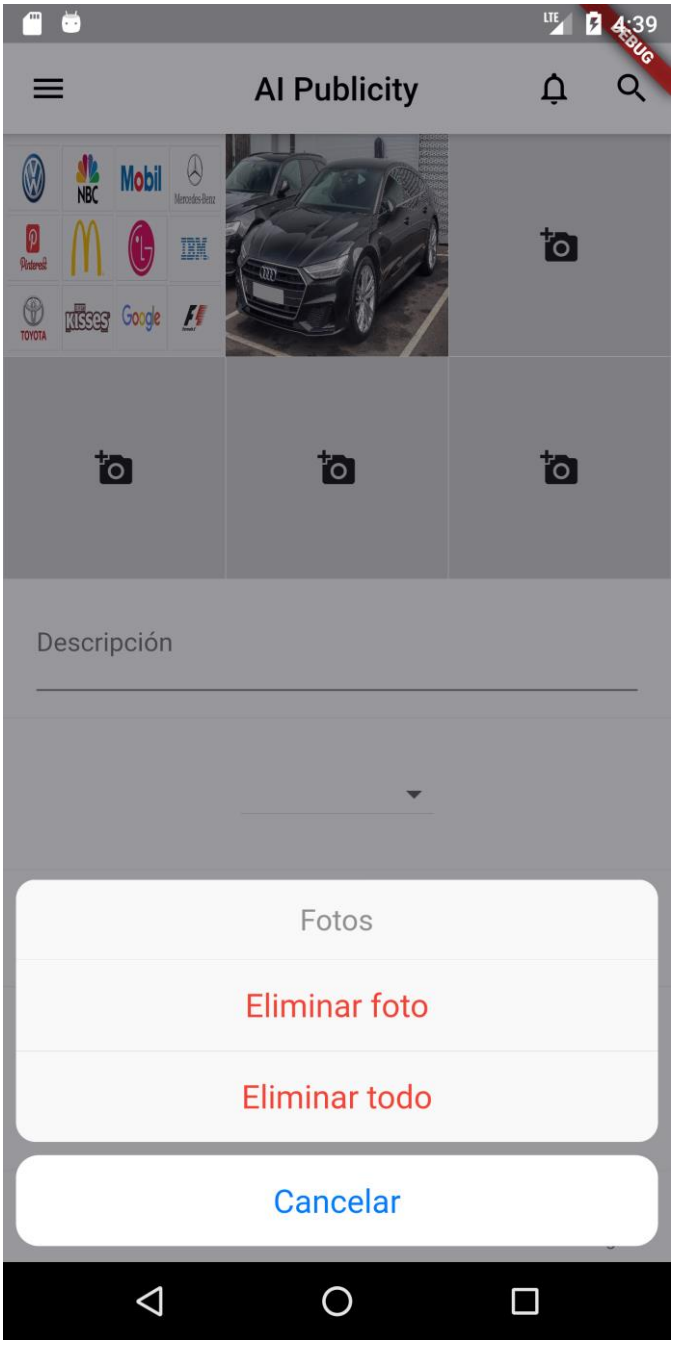


Figura 81: Opción eliminar imágenes en la ventana añadir publicación

La aplicación avisará con un mensaje cuando la publicación se haya subido correctamente (Figura 82).

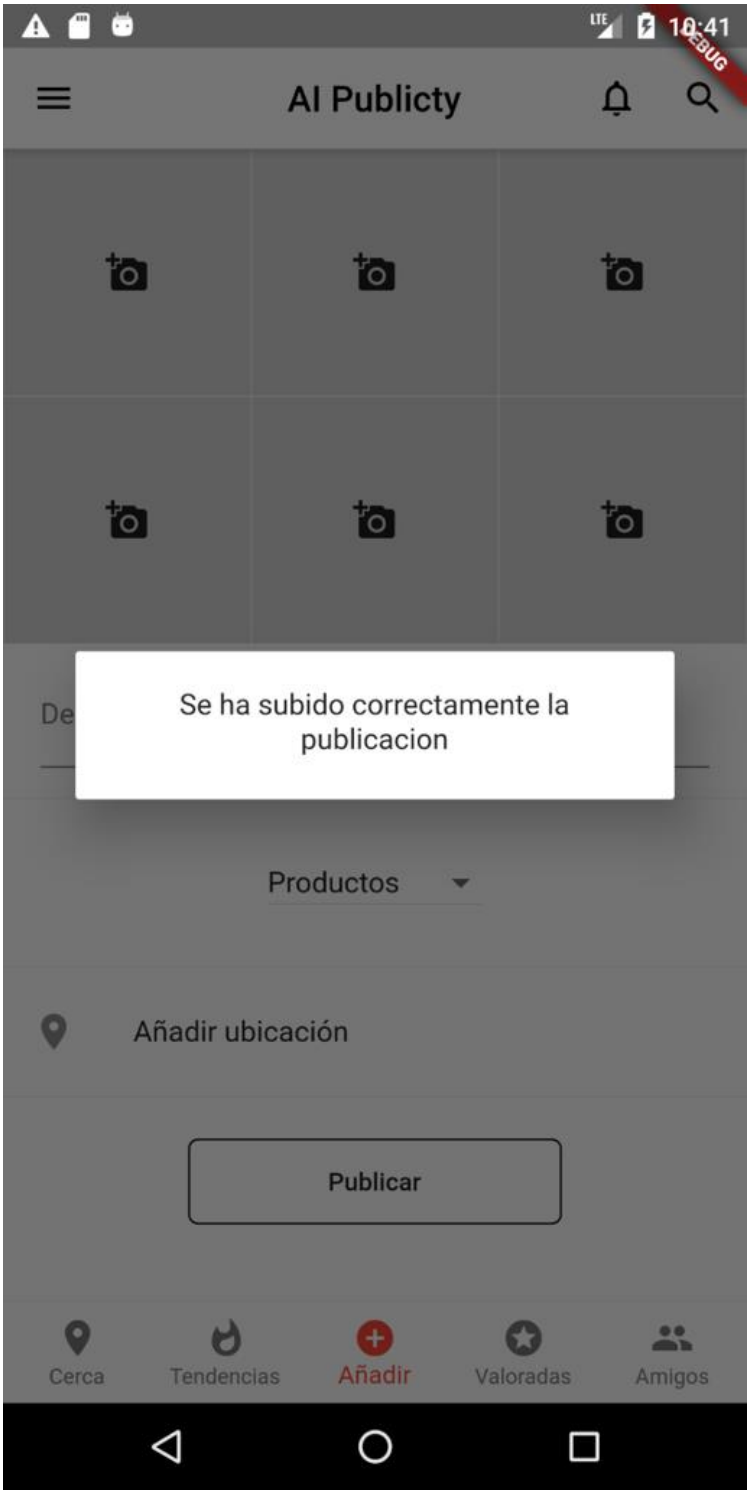


Figura 82: Publicación subida correctamente

5.11. Menú lateral cuenta de empresa

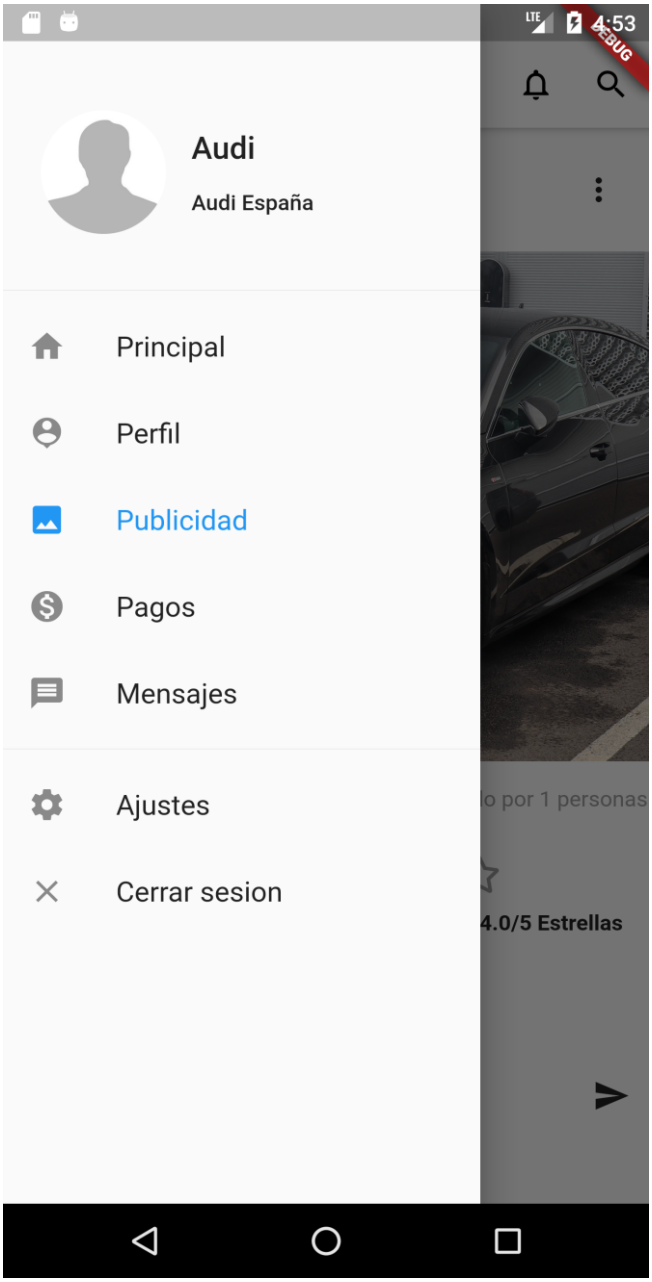


Figura 83: Menú lateral cuenta de empresa

En el menú lateral de la cuenta usuario normal, respecto del menú lateral de la cuenta de empresa, solo cambia la ventana publicidad, la cual mostrará todas las publicaciones de la publicidad de los usuarios y pagos que se deberá realizar a los usuarios por la publicidad.

5.12. Notificaciones de publicidad

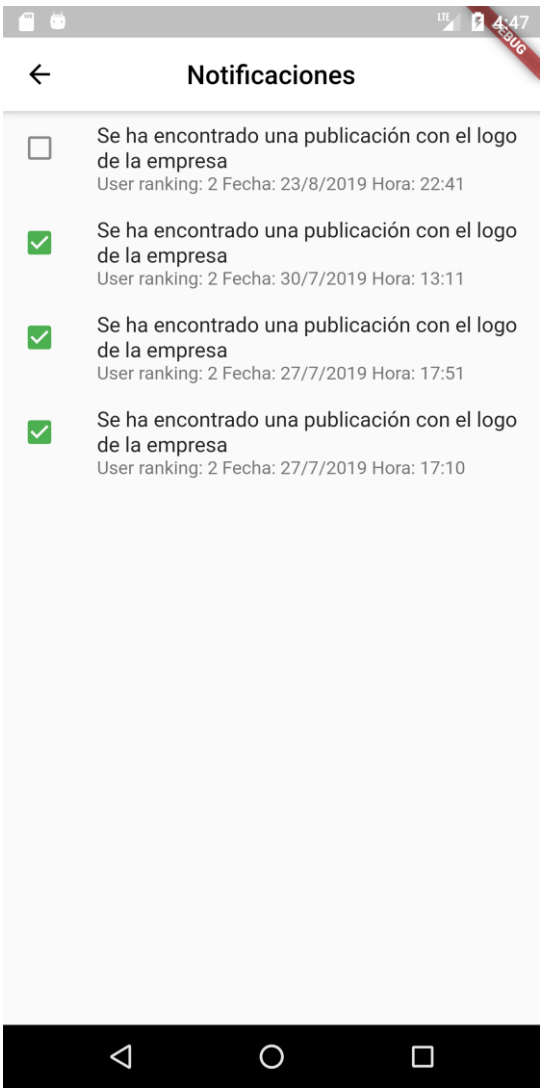


Figura 84 y 85: Notificaciones y visualización de notificación

Al presionar sobre la campana de notificaciones se abrirá la ventana de notificaciones que se muestra en la Figura 84. En la ventana de notificaciones presionamos la notificación que se desea visualizar, abriéndose la ventana con la publicación de la publicidad (ver Figura 85). Cuando se visualiza una notificación nueva aparecerá el icono en visto.

5.13. Ganancias y pagos

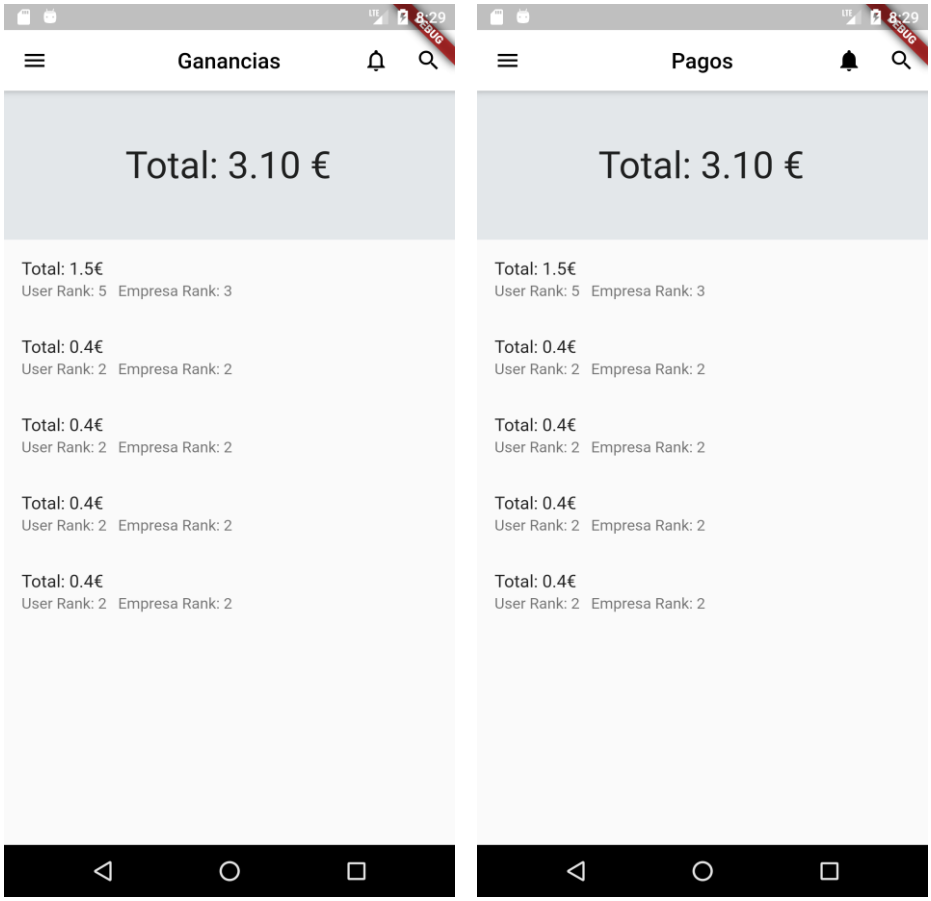


Figura 86 y 87: Ganancias usuario y pagos de empresa

A las ventanas de las Figura 86 y 87 se accede desde el menú lateral de la aplicación (ver Figuras 52 y 83).

Cuanto mayor sea el ranking del usuario y el ranking de la empresa, mayor será el beneficio obtenido por el usuario. El índice de *UserRank* y *EmpresaRank* se valora de 0 a 10. El cómputo del importe total deberá tener en cuenta estos valores justo en la fecha que ha sido notificado, ya que varían con el tiempo.

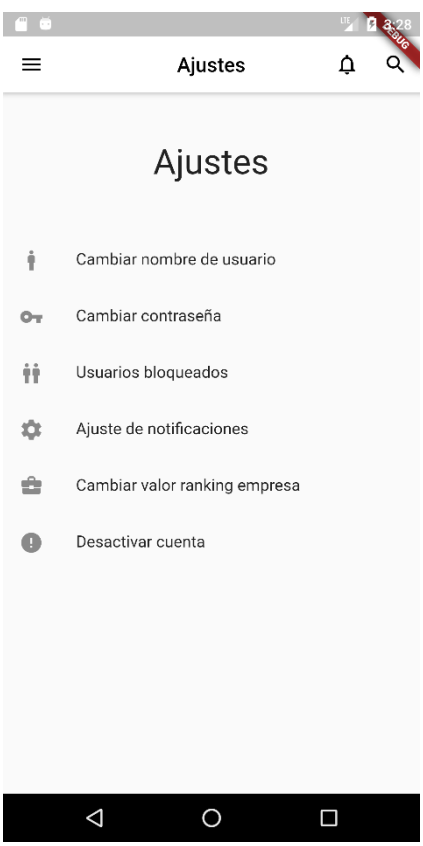
```
double dinero = empresa.userRank * (User.user.userRank / 10);
```

Figura 88: Cálculo del dinero por la publicidad

En la Figura 88 podemos observar cómo se realiza el cálculo del importe por la publicidad dependiendo del ranking del usuario y de la empresa.

El ranking de la empresa será decisión de la empresa indicando el importe máximo que desea pagar por la publicidad. Por ejemplo, si la empresa decide pagar un importe máximo de 3 euros

por la publicidad, se realizará el cálculo con el porcentaje del *UserRank*. Si este usuario consigue un ranking de 5 será el 50 por ciento del máximo de la empresa, y en este caso el resultado será 3 x 0,5 =1.5€.



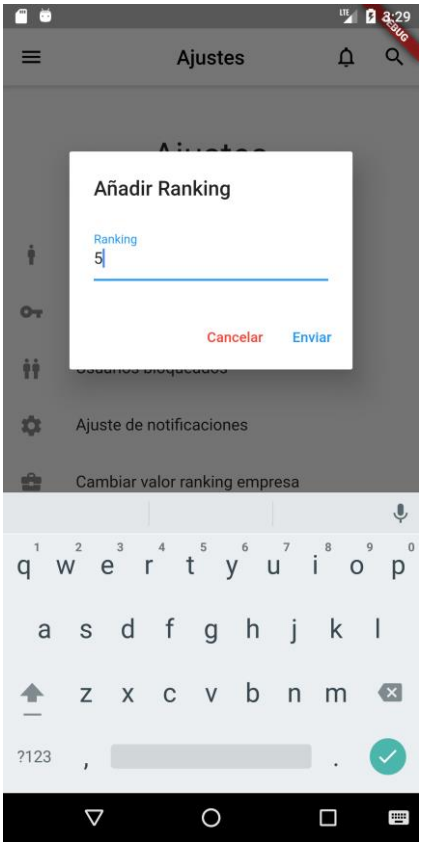


Figura 89 y 90: Ajustes vista empresa y Añadir ranking empresa

En la ventana de la Figura 89 podemos realizar cambios a la cuenta de la empresa. El cambio que destaca, respecto al usuario normal, es la acción *Cambiar valor ranking empresa*. Al presionar esta acción la empresa podrá cambiar su *ranking* como se muestra en la Figura 90.

El ranking de usuario se calcula mediante el número de "me gusta" total, el número de valoraciones total y la valoración media total.

El primer paso es conseguir el resultado total de "me gusta" del usuario, y para conseguir este resultado se realiza la consulta siguiente desde la API a la base de datos:

```
router.get('/ranknumgusta/:id', (req, res) => {
  const { id } = req.params;
  mysqlConnection.query('SELECT COUNT(*)
      FROM usuario_megusta_publicacion WHERE publicacion_idpublicacion
      IN (SELECT idpublicacion FROM publicacion WHERE usuario_idusuario
      = ?)', [id], (err, rows, fields) => {
      if (!err) {
        res.json(rows[0]);
```

```
      } else {
          console.log(err);
      }
  });
});
```

El segundo paso es conseguir el número total de valoraciones de un usuario. El total de valoraciones se consigue con la siguiente consulta desde la *API* a la base de datos:

```
router.get('/ranknumvalor/:id', (req, res) => {
  const { id } = req.params;
  mysqlConnection.query('SELECT COUNT(*) FROM
      usuario_valora_publicacion WHERE publicacion_idpublicacion
      IN (SELECT idpublicacion FROM publicacion WHERE
      usuario_idusuario = ?)', [id], (err, rows, fields) => {
      if (!err) {
        res.json(rows[0]);
      } else {
          console.log(err);
      }
  });
});
```

El tercer paso es conseguir la valoración media total del usuario. Esta media se consigue con la siguiente consulta a la base de datos:

```
router.get('/rankmediavalor/:id', (req, res) => {
  const { id } = req.params;
  mysqlConnection.query('SELECT AVG(estrellas)
      FROM usuario_valora_publicacion WHERE publicacion_idpublicacion
      IN (SELECT idpublicacion FROM publicacion WHERE
      usuario_idusuario = ?)', [id], (err, rows, fields) => {
      if (!err) {
        res.json(rows[0]);
      } else {
          console.log(err);
      }
  });
});
```

Al tener estos tres resultados podemos proceder a realizar el *ranking* del usuario. La manera de conseguir el ranking en el sistema es la siguiente:

El cálculo se realiza con el usuario que ha obtenido el mejor resultado en la aplicación. Por ejemplo, si el usuario con el mayor número de “me gusta” tiene un total de 50, realizamos el cálculo con su total para obtener nuestro porcentaje de ranking respecto al mejor resultado en esta

parte. Al tener un total de 25 “me gusta” el resultado es: 25/50 = 0.5 = 50%. El cálculo se realiza de la misma manera en los tres casos.

En el caso de encontrar algún dato mayor que el guardado en alguna de las partes se realizará una actualización de los datos como se muestra en la Figura 91.

```
if(gusta>rank.numgusta || valoraciones>rank.numvaloraciones || mediaValoraciones>rank.valormedio){
  await _dio.put(this._dirRank, data: {
    "numgusta": gusta>rank.numgusta ? gusta : rank.numgusta,
    "numvaloraciones": valoraciones>rank.numvaloraciones ? valoraciones : rank.numvaloraciones,
    "valormedio": mediaValoraciones>rank.valormedio ? mediaValoraciones : rank.valormedio
  });
}
```

Figura 91: Actualización de ranking

Una vez obtenido el porcentaje de las tres partes, se deberá sumar estas tres partes para conseguir el resultado total en las tres partes, y este resultado se dividirá entre tres para conseguir la media total en las tres partes. El último paso es multiplicar la media total de las tres partes por diez para conseguir un valor de 0 a 10.

En la Figura 92 se puede observar la fórmula para conseguir el resultado del *UserRank*.

```
double rankUser = (gusta / rank.numgusta) + (valoraciones / rank.numvaloraciones) + (mediaValoraciones / rank.valormedio);
rankUser = (rankUser/3)*10;
```

Figura 92: Calcular ranking de usuario

6. Conclusiones y trabajo futuro

6.1. Conclusiones

El desarrollo de este proyecto ha supuesto un gran reto, ya que se ha hecho uso de herramientas nuevas y nuevos lenguajes de programación avanzados; el desarrollo del proyecto ha sido complejo. Como dato que nos puede servir para cuantificar la complejidad de la aplicación, se destaca el hecho de tener una base de datos compuesta por 11 tablas. Pero este gran reto se ha resuelto de forma satisfactoria, cumpliendo con éxito todos los objetivos iniciales marcados en el anteproyecto.

Este proyecto ha permitido seguir el ciclo de un proyecto software pasando por sus distintas fases, tales como análisis, diseño, codificación y pruebas. El resultado conseguido es un producto software híbrido para aplicaciones móviles con una API Rest.

En el presente Trabajo Fin de Grado se ha creado una aplicación híbrida para dispositivos móviles con el *framework* de *Flutter* [4], que permite a las empresas llevar un seguimiento de las personas que hacen publicidad de sus productos con fines comerciales. Toda la información del sistema se guarda en una base de datos MySQL [19] y, para realizar peticiones al servidor, se ha utilizado una API Rest con *Node*.js [18].

En mi opinión me ha parecido un proyecto muy interesante en el que, al utilizar la arquitectura cliente servidor [17], he adquirido conocimientos tanto en la parte *front-end* [16] como en la parte *back-end* [16].

Como conclusión, considero que todo lo aprendido en este proyecto me ha servido de gran ayuda para comprender la lógica de desarrollo de aplicaciones compleja, sirviendo como base para mi futuro profesional.

6.2. Trabajos futuros

He tomado los siguientes puntos a mejorar o implementar en un futuro:

- Desarrollar una aplicación web, ya que únicamente se ha desarrollado el proyecto para aplicaciones móviles.
- Permitir utilizar la app sin la necesidad de estar registrado en el sistema.
- Recibir publicidad recomendada por el historial del usuario en la aplicación.
- Incluir filtros en la aplicación para las imágenes.
- Analizar videos de una publicación en busca de logotipos.
- Automatizar los ingresos y pagos directamente a la cuenta bancaria por la publicidad.

Bibliografía

- [1] Documentación oficial para la instalación y configuración de Flutter: https://flutter.dev/docs/get-started/install
- [2] Documentación oficial API de Google Cloud Vision: https://cloud.google.com/vision/docs/request
- [3] Documentación oficial para el aprendizaje de Dart: https://www.dartlang.org/tutorials
- [4] Documentación oficial para el aprendizaje de Flutter: https://flutter.dev/docs/reference/tutorials
- [5] Documentación oficial de Firebase Cloud Firestore: https://firebase.google.com/docs/firestore/?hl=es-419
- [6] Documentación oficial Android Studio: https://developer.android.com/training/basics/firstapp?hl=es-419
- [7] Documentación oficial Postman: https://learning.getpostman.com/docs/postman/api_documentation/intro_to_api_documentation/
- [8] Documentación oficial Visual Studio Code: https://code.visualstudio.com/docs/introvideos/basics
- [9] Prototipo Marvel: https://marvelapp.com/
- [10] Estudio Nielsen sobre las formas de publicidad más odiadas en web: https://www.nielsen.com/es/es/insights/news/2013/la-publicidad-con-humor-la-quemas-cala.html
- [11] Paquete para realizar peticiones http: https://pub.dev/packages/http
- [12] API REST: https://www.idento.es/blog/desarrollo-web/que-es-una-api-rest/
- [13] Documentación oficial framework React Native: https://facebook.github.io/reactnative/
- [14] Documentación oficial framework Ionic: https://ionicframework.com/
- [15] Documentación oficial framework PhoneGap: https://phonegap.com/
- [16] Front-end y Back-end: https://www.campusmvp.es/recursos/post/Desarrollador-web-Front-end-back-end-y-full-stack-Quien-es-quien.aspx
- [17] Artículo cliente servidor: https://blog.infranetworking.com/modelo-cliente-servidor/
- [18] Documentación oficial Node.js: https://nodejs.org/es/
- [19] Documentación oficial MySQL: https://www.mysql.com/
- [20] Documentación oficial JSON: https://www.json.org/json-es.html

- [21] Documentación oficial Docker: https://www.docker.com/
- [22] Documentación oficial MySQL Workbench: https://www.mysql.com/products/workbench/
- [23] Articulo REST: https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos
- [24] Documentación oficial JavaScript: https://www.javascript.com/
- [25] Documentación oficial framework React Native: https://facebook.github.io/reactnative/
- [26] Documentación oficial framework Ionic: https://ionicframework.com/
- [27] Documentación oficial framework PhoneGap: https://phonegap.com/
- [28] Artículo de Flutter e Ionic: https://www.syntonize.com/flutter-o-ionic/
- [29] Artículo de Flutter y React Native: https://www.openinnova.es/flutter-vs-react-native-espanol-cual-es-mejor/

UNIVERSIDAD DE ALMERÍA

Resumen/Abstract

El presente Trabajo Fin de Grado pretende satisfacer la necesidad de las empresas para anunciar sus productos de una forma excepcional. Se trata de una aplicación, llamada *AI Publicity*, en la que usuario es el que realiza la publicidad de productos de empresas forma innovadora en un entorno centrado en una red social. Entre sus características principales, se destaca el hecho de que cuando un usuario añade una publicación, el sistema se encargará de analizar las imágenes detectando la existencia de logotipos; el usuario, al hacer publicidad de productos, recibirá una recompensa económica por parte de la empresa.

AI Publicity se ha desarrollado como una aplicación móvil híbrida, válida tanto para *iOS* como para *Android*, haciendo uso del *framework Flutter*, una API *Rest* en *Node.js* y una base de datos MySQL.

This Final Degree Project aims to satisfy the need of companies to advertise their products in an exceptional way. It is an application, called AI Publicity, in which the user is the one who advertises company products in an innovative way in an environment centered on a social network. Among its main features, stand out the fact when the user adds a publication, the system will analyze the images looking for the existence of logos; The user, when advertising products, will receive an economic reward from the company.

AI Publicity has been developed as a hybrid mobile application, valid for both iOS and Android, using the Flutter framework, a Rest API in Node.js and a MySQL database.