Análisis, diseño, desarrollo, implementación e integración de nuevos módulos al proyecto plataforma de anuncios para la comunidad universitaria UIS (UISAds).

Elkin Darío Fernández Celis

Trabajo de Grado para Optar al título de Ingeniero de Sistemas

Director

Carlos Adolfo Beltrán Castro

MSc en Ingeniería de Sistemas e Informática

Codirector

Leonel Parra Pinilla

MSc en Ingeniería de Sistemas e Informática

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de Ingeniería de Sistemas e Informática

Bucaramanga

2022

Agradecimientos

Primeramente, a Dios por permitirme pasar por esta etapa en la vida y poder cumplir a cabalidad con ella.

A mis padres Eli Fernández Jaimes y Geinny Celis Suarez, a mi hermana Kelly Tatiana Fernández Celis por su comprensión y apoyo en todo el proceso de formación académica profesional y personal.

A los profesores Carlos Adolfo Beltrán y Leonel Parra Pinilla por su confianza e instrucción en la realización de este proyecto de grado.

A mis compañeros de Ingeniería de sistemas y del grupo Calumet por el apoyo brindado en el proceso académico.

Tabla de Contenido

Pág.

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Resumen

Título: Análisis, diseño, desarrollo, implementación e integración de nuevos módulos al proyecto plataforma de anuncios para la comunidad universitaria UIS (UISAds).[*]

Autor: Elkin Darío Fernández Celis[*]

Palabras Clave: Anuncios, Aplicativo, Comunidad, Módulos, Plataforma, Servicios

Descripción: El aplicativo móvil UISAds es un prototipo de plataforma de anuncios con el fin de dar a los miembros de la comunidad universitaria un lugar en el espacio virtual, donde puedan publicar anuncios y avisos en los cuales ofrezcan tutorías, asesorías, traducciones, arriendos, ventas, entre otros anuncios ofrecidos para la misma comunidad UIS, con el fin de ayudar a establecer un contacto entre las partes interesadas en un servicio específico y lograr que se relacionen entre ellos. Los requerimientos actuales de los usuarios al interactuar con la plataforma llevaron a la necesidad de diseñar una segunda versión del prototipo del aplicativo móvil UISAds que incluyera dentro de estas, nuevas funcionalidades; tal como el acceso a la plataforma con medios más simples y rápidos de autenticación, empleando sistemas de inicio de sesión a través de Facebook y Google para lograr esto; así como también la posibilidad de ingreso como invitado sin tener la necesidad de crear una cuenta de usuario, para poder solo visualizar los anuncios publicados, pero con la limitante de no poder interactuar con ellos; la opción de reportar los anuncios que puedan ser considerados como inapropiados, donde si el número de reportes es representativo respecto al número de interacciones que ha tenido el anuncio este sería eliminado de forma automática; mejorar la experiencia del usuario permitiéndole seleccionar temas de su interés, facultando al usuario el ingreso de palabras claves al sistema, para con base en estas pueda recibir notificaciones de los anuncios más relevantes para él.

Actualmente, existen los módulos de autenticación, anuncios, usuarios, pero no existe un módulo para el manejo de las notificaciones, por lo cual se requiere la creación de este, es requerido también la actualización de los módulos anteriormente mencionados para dar solución a las nuevas necesidades previstas de los usuarios en la plataforma.

---

[*] Trabajo de Grado

[**] Facultad de Ingenierías Físico-mecánicas. Escuela de Ingeniería de Sistemas e Informática. Programa académico. Director: Carlos Adolfo Beltrán Castro. MSc en Ingeniería de Sistemas e Informática. Codirector: Leonel Parra Pinilla. MSc en Ingeniería de Sistemas e Informática

Abstract

Title: Analysis, design, development, implementation, and integration of new modules to the announcement platform project for the UIS university community (UISAds).[*]

Author(s): Elkin Darío Fernández Celis [**]

Key Words: Announcements, Application, Community, Modules, Platform, Services.

Description: The UISAds mobile application is a prototype of an advertisement platform with the purpose of giving members of the university community a place in the virtual space, where they can post ads and announcements in which they offer tutoring, consulting, translations, leases, sales, among other ads offered for the same UIS community, in order to help establish contact between parties interested in a specific service and make them relate to each other. The current requirements of users when interacting with the platform led to the need to design a second version of the prototype of the UISAds mobile application that would include new functionalities, such as access to the platform with simpler and faster means of authentication, using login systems through Facebook and Google to achieve this, as well as the possibility of entering as a guest without having to create a user account, in order to only view the ads published, but with the limitation of not being able to interact with them; the option to report ads that may be considered inappropriate, where if the number of reports is representative with respect to the number of interactions that the ad has had, it would be automatically eliminated; improve the user experience by allowing the user to select topics of interest, allowing the user to enter keywords into the system, so that based on these keywords he/she can receive notifications of the most relevant ads for him/her.

Currently, there are modules for authentication, announcements, users, but there is no module for the management of notifications, so the creation of this module is required, it is also required to update the modules to provide a solution to the new needs of users in the platform.

[*] Degree Work

[**] Physical- Mechanical Engineering Faculty. Systems and Informatics Engineering School. Project director: Msc Carlos Adolfo Beltrán Castro. Project co-director: Msc Leonel Parra Pinilla.

Introducción

El aplicativo móvil UISAds es una plataforma que permite la conexión e integración de los miembros de la comunidad universitaria que se encuentran interesados en ofrecer o recibir un servicio. En la actualidad en la búsqueda de mejorar la forma en que ofrecen sus servicios han surgido diferentes necesidades entre estos miembros, tales como ofrecer servicios de tutorías, asesorías, productos de emprendimiento, ventas de artículos, préstamos, e incluso arriendos.

Anteriormente, antes del uso del aplicativo UISAds, la creación y publicación de anuncios de los servicios ofrecidos era realizada en grupos de redes sociales en los que la comunidad universitaria interactúa y donde suele compartir información y contenidos de diversas indoles, tales como “Información UIS 2.0” o “comunidad uis”, estos grupos aunque fueron de gran ayuda para que las personas dieran a conocer sus iniciativas poseen la desventaja de ser caóticos, con un distribución poco amigable y de difícil búsqueda para un usuario que desee dar con un anuncio en específico, y sumándole a esto la necesidad creciente de una variedad de servicios por gran parte de la comunidad UIS, era necesario introducir una solución que agrupará estos servicios por categorías específicas y se enfocará en anunciar esto a los más interesados, mejorando su objetividad. Para alcanzar este objetivo se debe mantener y mejorar las diferentes funciones que ofrece la plataforma UISAds a las personas que interactúan en ella, además tener en cuenta las nuevas funcionalidades a crear, en pro de satisfacer las necesidades no atendidas que crecen en los usuarios de la comunidad universitaria, los cuales, en su gran mayoría, pertenecen al estudiantado de la universidad.

La plataforma actual de anuncios para la comunidad universitaria (UISAds), fue una idea pensada y proyectada como solución por parte de estudiantes para estudiantes y demás miembros

que pertenecen a la Universidad Industrial de Santander, pero ante las constantes necesidades percibidas para los usuarios del aplicativo, y en búsqueda de cumplir el objetivo por el cual fue creada y diseñada la plataforma de anuncios, como un espacio virtual de conexión entre usuarios que ofrecen y exponen sus productos y servicios a sus interesados, se vio en la obligación de actualizar la plataforma con los nuevos requerimientos detectados para los usuarios que utilizan este espacio que esta para la interacción de las personas de la comunidad.

En el presente documento se muestran los diferentes pasos que se llevaron a cabo desde el inicio del proyecto, pasando por las diferentes etapas de su diseño, desarrollo y construcción, la toma del prototipo base del proyecto inicial de la plataforma UISAds, la socialización y el análisis de los requerimientos presentados con base en la necesidad del cambio continuo y en pro de generar un buen diseño de las nuevas funcionalidades que complementaran a la plataforma, la implementación de la metodología en cascada buscando que en cada etapa de desarrollo dentro de esta metodología ser lo más analíticos y precisos posible para alcanzar en la finalización del proyecto un buen producto de software.

Además, se hará uso de herramientas especializadas para el desarrollo de software que permitan simplificar y construir los nuevos módulos del aplicativo de forma intuitiva y amigable con el usuario, cumpliendo con los estándares iniciales que se plantearon en el prototipo inicial. Las herramientas usadas en este caso para el aplicativo en la parte del desarrollo móvil o frontend de la aplicación serán: Flutter, un framework de Google enfocado en el desarrollo de aplicaciones móviles, ya que cuenta con ventajas muy importantes en el desarrollo móvil por permitir compilar de forma nativa estas aplicaciones, NodeJS y Express, un framework de JavaScript por parte del lado del servidor, usado en la parte backend del proyecto, que son herramientas usadas para crear aplicaciones escalables y de fácil desarrollo.

En conclusión, con la nueva versión de la plataforma UISAds se espera que se den facilidades a los usuarios que publican un servicio, de buscar que su publicación tenga el mayor alcance posible en la comunidad universitaria que hace uso de la plataforma, facilitando su ingreso a esta, además permitiendo que los interesados se enteren de las nuevas actualizaciones de sus intereses y puedan estar al tanto de todo lo relacionado a sus anuncios favoritos.

1. Planteamiento y Justificación

El aplicativo móvil UISAds consiste en una plataforma de anuncios enfocada en la comunidad universitaria UIS, es un prototipo de plataforma móvil desarrollado por Jorge Andrés Triana Mojica y Jhon Andrés Parra Rodríguez que dota a los miembros de la comunidad de un lugar donde puedan publicar anuncios y avisos tales como tutorías, asesorías, traducciones, préstamos, e incluso arriendos, ofrecidos por miembros dentro de la misma comunidad UIS permitiendo que haya un contacto entre las partes interesadas de un servicio en específico.

En la plataforma desarrollada se evidenciaron las nuevas necesidades de los usuarios del sistema al interactuar con el aplicativo. Entre estos requerimientos esenciales está la falta de opciones de autenticación dentro de la plataforma que conlleva a una experiencia de ingreso menos fluida, por lo cual se desea añadir nuevas formas de ingreso mediante el uso de plataformas de terceros como lo son Google y Facebook. En la sección de visualización de anuncios no hay manera de que se pueda reportar una publicación que pueda ser considerada como inapropiada por parte de los usuarios, actualmente está la opción de colocar un “me gusta” o un “no me gusta” dependiendo de lo mostrado en el anuncio, pero esto puede llegar a ser insuficiente para el control de contenido no deseado.

Además, se vio en la necesidad de construir un módulo de notificaciones con el fin de informar a los usuarios acerca de nuevos anuncios que podrían ser de su interés en la plataforma. Adicionalmente con el fin de lograr una mayor atención de los usuarios potenciales de la comunidad universitaria en la plataforma, con base en los principios de diseño de interfaces móviles (Think With Google, 2016) se prevé la necesidad de una aplicación con un acceso más rápido, intuitivo y cómodo, ya que obligar a los usuarios a registrarse para poder visualizar los

anuncios puede resultar tedioso y aumenta las posibilidades de que el usuario no continúe en la búsqueda y abandone la aplicación.

Se pretende con estas mejoras hacer una plataforma más interactiva y enfocada en los requerimientos a futuro que puedan tener los usuarios, además de permitir que el aplicativo sea más útil y posea un mayor tiempo de vida dificultando su obsolescencia, esto enfocado en dar mantenimiento a lo ya elaborado, desarrollando nuevos servicios y anticipando necesidades futuras en la plataforma.

2. Objetivos

2.1 Objetivo General

Desarrollar nuevos módulos a la plataforma de anuncios para la comunidad universitaria UIS (UISAds) en búsqueda de hacerla más amigable con el usuario, brindando mayor capacidad de acciones y mejor gestión a sus anuncios de interés.

2.2 Objetivos Específicos

- En el módulo de perfil, posibilitar al usuario añadir sus temas de interés mediante palabras clave con el fin de notificarle anuncios más relevantes.
- Crear un módulo de notificaciones que, con base en los intereses seleccionados por el usuario, informe a este acerca de nuevas publicaciones.
- En el módulo de perfil de usuarios, implementar las siguientes funcionalidades:
    - Permitir el almacenamiento de un historial de anuncios vistos recientemente, no mayor a 100 anuncios, con fecha menor a un mes de haber sido visualizados.
    - Habilitar al usuario la opción de marcar y almacenar en su cuenta anuncios como favoritos, en el caso de que este desee visualizarlos fácilmente en un futuro.
- En el módulo de autenticación e ingreso a la plataforma implementar las siguientes funcionalidades:
    - Posibilitar un ingreso a la plataforma sin tener la necesidad de estar registrado en ella, esto incorporando la funcionalidad de ingreso como invitado, en la que el usuario pueda solo visualizar anuncios sin la capacidad de generar nuevos anuncios o contactar con el anunciante del mismo.

    - Mejorar la fluidez y seguridad de acceso a la plataforma integrando al ingreso del aplicativo, autenticación por grandes plataformas como lo son Google o Facebook, buscando con esto mejorar la experiencia del usuario.

- En el módulo de anuncios, permitir el reporte de un anuncio o clasificado como inapropiado, en busca de dar más herramientas al usuario para el control de contenido que no cumple con lo que promete o busca timar al usuario.
- Validar e integrar las nuevas funcionalidades desarrolladas en los módulos existentes, verificando que se cumplan los objetivos y requerimientos planteados, además de comprobar el correcto funcionamiento de la aplicación con lo nuevo implementado.

3. Marco de Referencia

3.1 Marco Conceptual

Para una mejor comprensión del proyecto se presentan los términos más relevantes y necesarios para el autor.

3.1.1 Anuncio Clasificado

Se puede definir en pocas palabras a un clasificado como: "Anuncio por líneas o palabras en la prensa periódica". (Real Academia Española, s.f., definición 3). Estos anuncios pueden estar publicados ya sea en un medio online o en un medio físico, siendo en la actualidad los más visualizados y con mayor alcance los anuncios del medio online. Esto en gran medida por la revolución digital que ha llevado incluso a empezar a afectar a los medios físicos y llevarlos a adaptarse o desaparecer. Como lo ocurrido con uno de los periódicos de clasificados muy popular por los años 2000, que debido al alcance y reducción de sus ingresos por medios impresos disminuyeron considerablemente, optaron por ofrecer su servicio de forma online. (Cierra el periódico Segundamano después de 30 años de clasificados, 2008). Los anuncios clasificados pueden ser considerados como espacios "alquilados" en los cuales se busca ofrecer a un usuario interesado o comprador un servicio o un bien, que puede ser comprado, intercambiado o vendido por la persona que publica el anuncio o vendedor. Por lo general, se agrupan en categorías, en busca de ser más fácilmente encontrados, por ejemplo, Anuncios de inmuebles, agrupan ya sea su venta, su alquiler o su permuta por otro bien. Su estructura básica consta de un título, descripción del clasificado o anuncio, número de contacto del anunciante y en algunos casos alguna imagen referente al anuncio, que permita tener una vista más detallada de lo que se ofrece.

3.1.2 Marketplace

Esta palabra es la unión de dos terminados usados en el lenguaje inglés, como lo son market cuyo significado es mercado y place, que quiere decir lugar. En resumen, este término se puede definir como un lugar donde comprar, un "escaparate virtual" en el cual los clientes tienen acceso a diferentes productos ofertados por marcas o empresas. En este ambiente virtual podemos decir que "varios comerciantes colocan sus productos a la venta en un sólo canal. Por esta razón, se puede decir que es también una versión online de un centro comercial: varias tiendas juntas para atraer a un mayor público." (Campos, 2017)

3.1.3 Economía Colaborativa

"Con economía colaborativa, se hace referencia a los nuevos sistemas de producción y consumo de bienes y servicios surgidos a principios de este siglo gracias a las posibilidades ofrecidas por los avances de la tecnología de la información para intercambiar y compartir dichos bienes y/o servicios" (Alfonso, 2016).

Tenemos además que este negocio está en constante evolución, ya que diariamente surgen nuevos espacios donde se hace uso de este concepto. Tenemos diferentes tipos de economía colaborativa que es determinada por la relación que hay entre las partes que la conforman de estas tenemos 4 Tipos:

- **Consumo Colaborativo:** Realizado mediante plataformas digitales, en las cuales los usuarios intercambian bienes y servicios. Hay una gran variedad de productos o servicios a los que acceden ya sea a cambio de dinero o por medio de un trueque.

- **Conocimiento Abierto:** son todas las actividades en las que se difunden conocimientos de manera altruista y sin barreras legales o administrativas. Se puede presentar de manera física o en plataformas a las que acuda el usuario. (OnTwice, 2019)
- **Producción colaborativa:** se trata de lugares físicos o digitales en los que se difunden y producen servicios de cualquier índole. Lo podemos diferenciar con los otros tipos en que la creación del contenido se produce también de manera colaborativa, no se limitan a compartirlo. (OnTwice, 2019)
- **Finanzas colaborativas:** pueden considerarse como sistemas de crédito tales como préstamos, ahorros, microcréditos, donaciones y financiación colectiva o crowdfunding. Por medio del Crowdfunding se puede aportar a diversas causas económicamente ya sea proyectos musicales, culturales, artísticos, entre otros.

3.1.4 Aplicativo Móvil

Un aplicativo o app móvil es el tipo de aplicación que se enfoca en la ejecución en un dispositivo móvil, ya sea un smartphone o una tablet. Este tipo de aplicación suelen ser pequeñas unidades de software con unas funciones limitadas, con un objetivo como tal de proporcionar una solución específica a un problema o requisito de un usuario, además de proporcionar una experiencia de calidad. Se diferencian de las aplicaciones de escritorio, en que las apps móviles se alejan de los sistemas de software integrado (aplicaciones ya incluidas con el hardware), puede proporcionar una funcionalidad que esté aislada y limitada, como puede ser una calculadora, un juego o un navegador web móvil.

Según el proceso de compilación de una aplicación, se permite clasificar en diferentes tipos tales como:

- **Aplicaciones Nativas:** Son aplicaciones diseñadas para un sistema operativo particular, son propios de una plataforma específica, con grandes ventajas en el rendimiento y en la Experiencia de usuario, debido a que se aprovecha la interfaz de usuario del dispositivo nativo y se accede a una gama de opciones proporcionada por la API que permite acelerar el desarrollo y construcción de esta, además de ampliar los límites para el uso de la app. El inconveniente con estas aplicaciones es el costo de construcción debido a que se enfoca a un sólo sistema operativo móvil, se ejecutará sólo en esa plataforma, por ejemplo, una aplicación para Android no se puede ejecutar en iOS, implicando esto que se deben tener dos equipos de desarrollo si queremos impactar o diseñar algo para las dos plataformas.
- **Aplicaciones Híbridas:** Son aplicaciones desarrolladas para ser usadas en varias plataformas con una misma base de código, pueden acceder a diferentes funcionalidades de los dispositivos tales como cámara, contactos, GPS, entre otras. Tienen un coste mucho menor en desarrollo que el de una app nativa, pero presentan el problema de que puede no estar optimizadas para cada sistema operativo lo que puede disminuir el rendimiento de estas. Se puede diferenciar también dentro de las aplicaciones híbridas el tipo de tecnología que usan ya sea las tecnologías que traducen a código nativo tales como React Native, que aunque se tarda más en escribir código se presenta una mejora en el rendimiento de estas acercándose a un rendimiento nativo, o las que ejecutan una aplicación como una aplicación web dentro de un contenedor o web frame que se ejecuta dentro de una app nativa, que suelen tener un poco menos rendimiento y sus elementos visuales presentan diferentes formas a los de una aplicación nativa en general.

3.1.5 Modelo Vista Controlador

Es un patrón de diseño que permite el desarrollo de aplicaciones interactivas, las cuales se dividen en tres elementos (modelo, controlador y vista) como se expresa en la figura 1. La vista proporciona una interfaz gráfica al usuario que permite ver la información proporcionada por el modelo, el controlador permite la interacción entre el modelo y la vista, está "gestiona" los datos proporcionados por el modelo y los envía a la vista y viceversa, y el modelo es un esquema en el cual se guardan los datos del sistema.

Figura 1 *Ejemplo Patrón de Diseño Modelo Vista Controlador*

Nota: El gráfico muestra el patrón MVC en los diferentes componentes de la aplicación. Adaptado de EasyAppCode por J. Herrera, 2020, EasyAppCode (https://www.easyappcode.com/upload/post-792545902.jpg). Obra de dominio público.

3.1.6 API REST

El concepto de REST o transferencia de estado representacional por sus siglas en inglés, nació en el año 2000 como parte de la tesis doctoral ‘Architectural Styles and the Design of Network-based Software Architectures’ escrita por Roy Fielding, cambiando por completo la ingeniería del software luego de esta publicación (BBVA, 2016). REST es un protocolo que define un conjunto de directrices sobre cómo debe comportarse la arquitectura de un sistema en la web, este hace un énfasis en la estabilidad de sus componentes y sus interacciones, así como también busca una uniformidad en las interfaces, mejorar la velocidad de navegación usando para esto el almacenamiento cache en un arquitectura en capas, la encapsulación de sistemas heredados, el fortalecimiento de la seguridad y la implementación independiente de componentes (Fielding, 2000).

Una API REST o API RESTful consiste en una API que obedece estos parámetros y restricciones impuestos por REST, esto buscando que la API web sea confiable y con un protocolo sin estado, suelen estar basadas en métodos HTTP con los cuales pueden ya sea acceder a recursos a través de parámetros codificados en la URL o a la transmisión de datos con el uso de JSON o XML (Fielding, 2014), siendo sus operaciones más importantes la de creación POST, la de lectura GET, la de edición PUT y finalmente la de eliminación DELETE.

3.1.7 JSON Web Token (JWT)

Usado comúnmente para el intercambio de la identidad y privilegios de un usuario entre un proveedor de identidad y un proveedor de servicios en un sistema de terceros, los JWT codifican

notificaciones para que se transmitan como un objeto JSON que es usado ya sea como carga útil de una estructura JSON Web Signature (JWS) o como texto sin formato de una estructura JSON Web Encryption (JWE) lo que le permite estar firmado digitalmente o protegido por integridad con un código de autenticación de mensajes y/o encriptado (Jones; Bradley; Sakimura, 2015).

Su estructura principal está compuesta por tres partes; el encabezado, también llamado header; la carga útil, también llamada payload y la firma de autenticación, también llamada signature. El encabezado es un elemento JSON que describe el tipo de token del documento y el método de cifrado usado en él, en el ejemplo de la figura 1 podríamos ver que este token está firmado con HMAC-SHA256. La carga útil contiene un conjunto de “claims” que describen la información que queremos transmitir y que el servidor puede usar para manejar correctamente la autenticación, estos "claims” pueden contener información sobre el emisor del token, el asunto del token, la hora de creación del token y el tiempo con el cual este dejará de ser valido, en el ejemplo de la figura 1 podemos ver 4 ejemplos de campos personalizados para el token, los cuales serán cifrados en base 64 y conformaran el cuerpo del token. La firma es generada concadenando el encabezado y la carga útil ya cifrada, separándolas por puntos, para luego ser procesada por un algoritmo se cifrado fuerte, HMAC-SHA256 en este caso, como se aprecia en la figura 2.

Figura 2. *Ejemplo de la estructura de un token JWT*

Nota: El gráfico muestra la estructura que compone a un JWT. Adaptado de Toptal por T. Tkalec, 2022, Toptal (toptal-blog-image-1426676395222.jpeg (700×499)). Obra de dominio público.

3.1.8 Framework

Especialmente útil en el desarrollo ágil de sofware, el concepto de framework, o también conocido en español como marco de trabajo (La Red Martínez; Acosta; Mata; Bachmann; Vallejos, 2012), es ampliamente usado desde hace tiempo, siendo este una aplicación semicompleta, un conjunto de abstracciones en la que el software, proporcionando una funcionalidad genérica y una estructura común reutilizable para las aplicaciones, la cual puede cambiarse selectivamente mediante un código adicional escrito por el usuario, los desarrolladores pueden integrar el marco de trabajo en su propia aplicación y lo amplían de tal manera que cumple con sus requisitos específicos, proporcionando así un software específico de la aplicación. Los marcos se diferencian de los juegos de herramientas en que proporcionan una estructura coherente en lugar de un simple conjunto de clases auxiliares (Johnson; Foote, 1988).

3.2 Marco Tecnológico

Se presentan las siguientes herramientas tecnológicas que se usaron durante el desarrollo del proyecto.

3.2.1 Flutter

Es un kit de herramientas creado por Google en 2016, permite la creación de aplicaciones multiplataforma basadas en el mismo código base, Flutter posee un sistema de widgets que facilitan la elaboración de las aplicaciones, además de poseer un rendimiento nativo, las aplicaciones en Flutter están escritas en Dart, es un lenguaje de programación desarrollado por Google. Flutter es un proyecto open-source (Flutter, s.f)

3.2.2 NodeJS

Lanzado en 2009, Node Js es un entorno de ejecución que permite la creación de aplicaciones escritas en JavaScript del lado del servidor. Usa un modelo orientado a eventos asincrónicos enfocado en la entrada/salida de datos. Al ser un proyecto open-source posee una gran comunidad que al cabo de los años han desarrollado dependencias sólidas las cuales pueden ser usadas de manera gratuita mediante el sistema de gestión de paquetes npm. (NodeJS, s.f)

3.2.3 Express

Es el framework más popular para la creación de aplicaciones del lado del servidor de NodeJS. Basado en el módulo http de Nodejs. Los componentes que conforman el framework se conocen como middlewares y son la columna principal del mismo. Permite una flexibilidad a la hora de personalizarlos y esto concede a los desarrolladores poder configurar sus aplicaciones web de una manera más escalable permitiendo su trazabilidad a lo largo del tiempo. (Mozilla, 2020)

3.2.4 Github

Github es una plataforma de alojamiento de código abierto para equipos que trabajan en proyectos de software. Posee un sinnúmero de funcionalidades para facilitar la comunicación entre usuarios lo que permite una mayor productividad en los equipos de trabajo. Con lo anterior, Github permite centralizar un proyecto de software y mantener un seguimiento en cada una de las fases del mismo. (Xataka, 2019)

3.2.5 MongoDB

MongoDB es una base de datos no relacional y de código abierto enfocada a documentos, organiza la información de una manera muy flexible mediante estructuras llamadas BSON (representación binaria de JSON), no posee un esquema para los documentos lo que da la libertad de agregar datos dependiendo de las necesidades de cada registro. Además, posee un sistema de referenciación lo que permite realizar consulta entre diferentes documentos como se haría en una base de datos relacional mediante joins.(Genbeta, 2014)

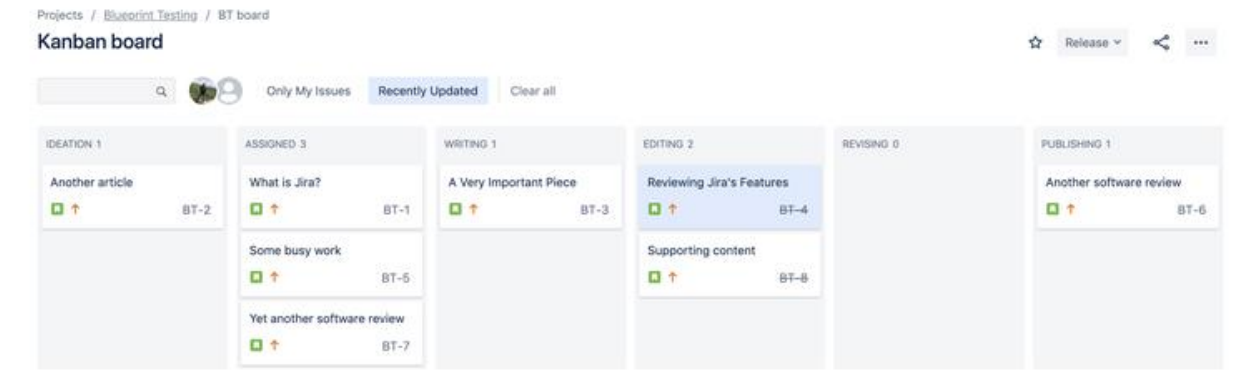
3.2.6 Jira

En los últimos años cada vez más empresas buscan mejoras en su productividad con la ayuda de herramientas tecnológicas, y apuntando a este objetivo existe Jira, esta utilidad se basa en la búsqueda de una eficiente gestión de procesos, en la gestión de flujos de trabajo particularmente útiles en la gestión de errores e incidencias, la resolución de problemas y la gestión de proyectos operativos. Entre sus principales funcionalidades se encuentra el uso de una tabla de

tareas conocida como Kanban, por su nombre en Japones, en la cual se puede hacer una gestión de tareas según su estado de progreso, de inicio a fin, por ejemplo, moviéndose del estado “Por hacer” a el estado “En progreso” para finalmente llegar al estado “Terminado”, en la figura 3 podemos ver un ejemplo de un Kanban en Jira.

Con el paso del tiempo Jira ha incorporado más utilidades a su plataforma, añadiendo capas de complejidad muy útiles para el funcionamiento de equipos Scrum, conteniendo plantillas y herramientas adecuadas para equipos agiles permitiendo seguir la capacidad y la velocidad del equipo.

Figura 3. *Ejemplo de un Kanban en Jira*



*Nota: El gráfico muestra la estructura usada para el Kanban mostrado en Jira. Adaptado de The Ascent por N. Morphus, 2022, The Ascent (*What_is_Jira_-_1_-_Kanban_board_vAsVohX.width-750.png (750×319) (foolcdn.com)*). CC-BY-NC-ND.*

3.2.7 Figma

A la hora de iniciar un proyecto la creación de un prototipo es clave para la comunicación, discusión y refinamiento de ideas entre los diseñadores y programadores del mismo, además ayuda en la temprana detección de errores, inconsistencias o redundancias que podría poseer la propuesta, Figma se presenta como una web que busca brindar ayuda a todo esto, permitiendo la generación de prototipos a través de un editor de gráficos vectoriales escalables donde los equipos de desarrollo puedan trabajar y aportar en conjunto y en tiempo real para la creación del diseño y el prototipo del proyecto.

Figma ha logrado posicionarse como la plataforma líder y predilecta a la hora de crear un diseño y un prototipo, siendo para algunos considerada como Github pero para diseñadores de interfaces (Lee, 2019). Esto ya que posee gran variedad de plantillas y además cuenta con una estructura intuitiva y rápida de usar, logrando que los usuarios puedan generar diseños con rapidez gracias a tener un uso completamente visual, evitando que los miembros del equipo deban ingresar código o programar para la creación del prototipo.

3.3 Estado del Arte

En el entorno actual, encontramos diversos ejemplos de aplicaciones que presentan sus anuncios a las personas interesadas de la mejor forma posible, categorizadas para permitir a los usuarios encontrar las publicaciones de su interés y realizar un pronto contacto con el anunciante. Estas herramientas fueron un punto de inspiración para el desarrollo de las nuevas funcionalidades en UISAds, debido a que muchas de estas han resultado exitosamente implementadas y las han

convertido en apps aún más populares, llevando a traer más personas a usarlas. Algunas de estas aplicaciones son:

3.3.1 OLX

Podemos considerar OLX como una plataforma de anuncios clasificados que facilita la visualización de estos y sirve para la venta, compra o intercambio de productos o servicios. Al igual que Facebook MarketPlace, permite a los anunciantes y personas interesadas comunicarse directamente, a través de la plataforma, sin necesidad de cancelar o pagar un producto en el momento. Olx no interviene en la venta tal como si lo hace mercado libre o ebay, en sí permite que tanto la persona interesada como el anunciante acuerden el medio de pago y comercian entre ellos. Limitándose a únicamente ser el intermediario de contacto, su forma de trabajo emula el editorial de clasificados de un sitio web de noticias. La simplicidad de la interacción entre comprador y vendedor la hace la plataforma perfecta para los usuarios debido a una de las grandes ventajas presentes basada en la geolocalización de los productos acorde a la persona interesada, es decir, la cercanía de comprador y vendedor es el punto esencial del éxito de este sistema.

Es tan simple la manera de vender, que solamente basta con realizar el respectivo registro, crear un anuncio, definir la categoría específica del anuncio y publicarlo, permaneciendo publicado un cierto periodo de tiempo y además de que existe la posibilidad de destacar el anuncio pagando una cierta cantidad para aumentar aún más la posibilidad de captar a un comprador. Eso sí, es importante saber que en este caso OLX al ser un intermediario no posee responsabilidad alguna en algún problema ocurrido entre un comprador y un vendedor.

Que los pagos se manejan directamente entre las partes interesadas concede el beneficio de no tener que preocuparse de manejar una gran infraestructura para captar pagos en los diferentes mercados, tal como Mercado Libre lo hace con Mercado Pago. Teniendo en cuenta que esto va

acorde a la filosofía “simple y para todos”, mientras en países europeos las medidas son muy estrictas con respecto a las transacciones, en otros lugares más específicamente como en Suramérica hay menos “burocracia” y es donde OLX más ha alzado su influencia.

Las principales opciones y funcionalidades presentes en la plataforma para compradores y vendedores son:

- Vista de Anuncios más recientes basados en la localización del usuario a comprar.
- Filtro por categorías más específicas dependiendo de categorías más generales, tales como Empleos, Servicios, Vehículos, Juguetes, entre otros. Ya sea por ejemplo un vehículo, particularmente una moto con sus diversas opciones como modelo, marca, año, rango de precio, entre otras.
- Filtro de anuncios de categorías más la ubicación de los anuncios en plataforma.
- Posibilidad de publicar un anuncio destacado para darle un alcance más grande a los posibles compradores que visiten el sitio.
- Elegir qué datos de contacto suministrar para un interesado en un clasificado.
- Compra de paquetes de anuncios para destacar, ideal para vendedores interesados en vender en más de una categoría.

3.3.2 Facebook Marketplace

Viene siendo una alternativa reciente expuesta por Facebook (ahora Meta) en su plataforma para el intercambio, compra y venta de bienes usados o nuevos, permitiendo el contacto rápido e interactivo entre los interesados, permitiendo además que un vendedor aparezca en la plataforma con un perfil de Vendedor conectado a su perfil principal. Está disponible en su versión web y móvil, siendo muy versátil y abierto.

Para el poco tiempo que lleva tiene buenas estadísticas, según los informes a corte de Mayo de 2018, las ventas rondaron los 40 mil millones de dólares estadounidenses, por lo que sus ganancias se han incrementado en más de la mitad. (CNET, 2018). En ese momento, alrededor de dos séptimas partes de las personas en el mundo usaban la plataforma de Facebook todos los días. Ahora, hay alrededor de 90 millones de páginas comerciales en Facebook y 140 millones de empresas utilizan este servicio; el 85% de las empresas estadounidenses empezaron a utilizarlo en 2016 y, finalmente, alrededor de 7 millones de personas utilizan este servicio para anunciarse allí. (Businessofapps, 2021).

Facebook MarketPlace ha ofrecido funcionalidades muy parecidas a las presentes en Mercado Libre, pero también nuevas características, en resumen, de estas se tienen:

- En el ámbito de los usuarios que quieren comprar, disponemos de una gran cantidad de categorías para mejorar la búsqueda de productos.
- Le permite limitar su búsqueda a través de filtros específicos.
- Permite la descripción de productos a vender o intercambiar.
- También te permite establecer límites mínimos y máximos para el precio del producto que estás buscando.
- Para los inmuebles hay filtros más específicos, tales como: número de habitaciones, número de baños; costo de búsqueda, área de búsqueda, etc.
- Facilita la conexión entre compradores y vendedores porque permite enviar mensajes directos entre ellos para que ellos mismos se encarguen de establecer las condiciones de venta o intercambio.
- Una característica recientemente agregada es la posibilidad de negocio entre comprador y vendedor esto gracias a el contacto directo a través de la misma plataforma y al rango de

precio dado por el vendedor al comprador, algo que no se maneja en ML o en alguna otra plataforma conocida.

3.3.3 Vendiste.com

"Anunciar un artículo que ya no use y que lo compre su vecino" (El Tiempo, 2017). La frase constituía la primera impresión dada a un usuario que le hace recordar las tradicionales ventas de garaje, así es la aplicación Vendiste, en la que se puede escoger el rango de distancia de los anuncios ya sea a 1KM, 5KM, 10KM y hasta 50KM.

Para lograr este objetivo, la aplicación utiliza la geolocalización para mostrar a los usuarios los productos a la venta en su industria. De esta forma, al poner en contacto a personas muy cercanas entre sí, el proceso de compraventa se vuelve más rápido y sencillo. Este concepto es similar al de Tinder, Badoo, Bumble y otras aplicaciones. Una vez que "emparejas" un producto que atrae tu atención con una distancia cercana, puedes comenzar charlando con el vendedor, acordar un punto de encuentro y completar la transacción. Este método tiene como objetivo reducir el fraude que puede ocurrir cuando las transacciones de divisas esperadas se realizan a través de la red (Echeverry, 2017).

"Nuestro enfoque es móvil, las ofertas de Mercado Libre o Amazon han dejado un segmento sin cubrir, que es el de lo local. Esto ya se hacía con los grupos de Facebook, en los que se dan a conocer productos que están a la venta y eso funciona muy bien porque no cuesta nada", explica el ingeniero Julián Builes, miembro del equipo de Vendiste, que desde su salida al mercado en junio de 2017 ya tenía 140.000 descargas (Echeverry, 2017).

Por tanto, quien quiera vender productos que ya no están en uso, solo necesita subir información básica como el nombre del producto, descripción, precio y una o más fotos a la aplicación. A partir de ahí, el interesado deberá enviar un mensaje al propietario para concertar una reunión e intercambiar productos a cambio de dinero.

De igual forma, Builes garantiza que la aplicación está diseñada con varios filtros de protección de datos para que no se compartan correos electrónicos, direcciones residenciales o números de teléfono, sino que se utilizarán con sistemas de mensajería internos.

Las funcionalidades que ofrece el aplicativo móvil y que resultan en un atractivo para los usuarios de la plataforma Vendiste son:

- Autenticación por medio de Facebook, lo que permite generar confianza entre los usuarios de la plataforma.
- Sistema de Georreferenciación, que permite visualizar los productos más cercanos al lugar de la persona que realizaba la búsqueda.
- Realización del intercambio o venta de un producto o servicio directamente entre las partes implicadas, sin intermediarios.
- No cobro al usuario que anuncia un producto, pero a cambio la aplicación mostrará anuncios publicitarios.
- La posibilidad de realizar una oferta de parte del comprador al vendedor ya sea para acordar un precio más bajo.

3.3.4 Mercado Libre

Es una empresa de comercio electrónico que proporciona a los usuarios las posibilidades de vender, comprar, anunciar y hasta pagar diferentes bienes o servicios ya sea nuevos

proporcionados por vendedores oficiales o terceros particulares, esta plataforma conecta a fabricantes, minoristas, mayoristas y hasta compañías grandes que ofrecen sus productos a través de Mercado Libre (ML). Se ha extendido en diversos países en Latinoamérica y Centroamérica llegando incluso a Estados Unidos. ML se ha convertido en una fuente de generación de ingresos para más de 52000 personas que ofrecen sus productos a través de ella, según las estadísticas en 2009 más de tres millones de personas y empresas vendieron por lo menos un artículo a través de este medio (Minuto Uno, 2010). En 2016, ML tenía un estimado de 174 millones de usuarios en el continente y había vendido 181 millones de productos en dicho año (La República, 2017).

Pero qué es lo que hace a Mercado Libre tan popular, se puede decir que sus diversas características presentes en su plataforma tales como:

- Historial de productos vistos recientemente con fecha menor a un mes.
- Sección de búsqueda que permite encontrar de manera más rápida un producto con características especificadas en los filtros.
- En la Sección de ofertas y demandas cuenta con una herramienta de PQR y ayuda muy útil en caso de algún problema con cualquier producto.
- Admite el uso de diferentes medios de pago, entre los que está PSE, Efecty, Visa, Mastercard, entre otros. Además de tener presente la opción de diferir el pago hasta en 48 cuotas (Esto es para tarjetas de crédito)
- El carrito de compras permite la visualización de los productos escogidos, también su coste y el monto total de la compra.
- Sección de favoritos muy útil para el seguimiento de productos en lista de compra futura.
- Posee un sistema de Recomendados que se nutre de las últimas búsquedas y te ofrece mejores alternativas a lo que buscas o productos relacionados.

Para el 2020 el aplicativo móvil de Mercado Libre contaba con más de 100 millones de descargas y aproximadamente 9 millones y medio de comentarios los cuales la mantuvieron con un promedio de 4.8 estrellas de calificación (Marketplacepulse, 2021).

4. Metodología

En la creación y desarrollo de los nuevos módulos de la aplicación UISAds se eligió la metodología de desarrollo en cascada adaptada para el desarrollo del primer prototipo del proyecto, esto debido a la necesidad de ir a la par con los nuevos requerimientos surgidos en la primera parte del análisis y diseño del sistema, dividiendo la metodología en 7 etapas para la construcción de la versión 2 del software del aplicativo.

4.1 Determinación de requerimientos

Basados en los nuevos requerimientos presentados en el proyecto inicial, en esta etapa se procederá a organizar los requerimientos funcionales y los no funcionales, teniendo en cuenta en los no funcionales que es importante que el rendimiento de la aplicación no se pierda por la adición de las nuevas funcionalidades, que el impacto no sea negativo y contribuya al mejoramiento de la plataforma en seguridad e integridad de los datos almacenados, con base en las encuestas realizadas en la parte final del proyecto versión 1 por los primeros desarrolladores de la aplicación donde se detallaron las necesidades presentadas por los usuarios de la comunidad universitaria, se evaluaron las posibles opciones y las utilidades que serán implementadas en la plataforma.

4.2 Diseño del sistema

En esta etapa se manejará el diseño del nuevo software con base en el análisis y la construcción de los modelos en UML, diagramas de flujo y diagramas de secuencia en búsqueda de una mejor comprensión del funcionamiento del aplicativo con las nuevas funcionalidades, también se establecen las primeras vistas del prototipo, el diseño de las interfaces gráficas de la aplicación móvil y la navegación entre estas, integrándolas al diseño inicial del aplicativo para la mejora en la usabilidad para el usuario.

Se construyen el prototipo, enfocado en cumplir los objetivos del proyecto, puliendo los pequeños detalles que lograrán que la aplicación mejore su diseño inicial y sea más sencilla de usar por el usuario.

4.3 Desarrollo del software

Durante esta etapa se realizará la programación y codificación fundamentados en la anterior etapa, se basa en la arquitectura de software montada en la etapa de diseño con base en los requerimientos tomados y refinados, se trabaja en la programación móvil o Frontend de la aplicación y su respectivo backend, buscando conectar con los artefactos ya desarrollados en la plataforma y se continué con el correcto funcionamiento de esta, para esto lo nuevo desarrollado debe acoplarse bien en lo ya construido.

4.4 Limpieza y pruebas

En esta etapa de limpieza se verifica él funcionamiento y revisión del código implementado para la construcción del aplicativo, se busca calidad por medio de las pruebas pertinentes unitarias, se busca y revisa cualquier error que se deberá solventar para llegar a la implementación del proyecto, se realizara la documentación de estas pruebas aplicando un control de calidad al código construido, llevando que a futuro cuando se necesite desarrollar nuevas funcionalidades o realizar el mantenimiento pertinente a la plataforma, si se encuentra con un error el desarrollador pueda revisar los estados anteriores, las pruebas realizadas y pueda corregir el error o construir la nueva mejora.

4.5 Implementación y verificación del programa

En esta fase se enfocará en lo ya realizado, ya que se ha finalizado el desarrollo y la codificación, se generan las pruebas de integración correspondientes para buscar posibles defectos en la información consultada, errores en la carga de datos, errores en construcción de widgets, adaptación de la pantalla o errores de enrutamiento a los puntos finales en el back, esto tendrá el fin de establecer que el código dentro de la plataforma es más limpio y está libre de inconsistencias que podrían afectar su operatividad futura y la necesidad de mantenimiento antes de lo esperado. También se evaluarán las capacidades de la plataforma, comparándolas con los requerimientos iniciales presentados en la construcción del proyecto y con los actuales presentados para esta mejora en la versión con el objetivo de identificar debilidades o redundancias presenta, y solucionarlas para hacer más eficiente el proceso de integración con las aplicaciones existentes.

4.6 Documentación del aplicativo

Esta esta fase se busca realizar una documentación extensiva al código favoreciendo la mantenibilidad al futuro de los diferentes artefactos del aplicativo, se presenta el proceso de construcción como un paso a paso buscando hacer entendible a una persona interesada en conocer el funcionamiento general de la plataforma, con la debida autorización de los desarrolladores, comprender la lógica del negocio planteada, analizar el código usado en la solución, para darle un entendimiento general y para que en un futuro en un mantenimiento se pueda realizar de la mejor forma, rápido y efectivo, debido a los cambios tecnológicos actuales se intenta construir software que perduré y pueda ser extensible a futuro.

4.7 Ajustes finales.

Finalmente se hace una verificación general del aplicativo, donde se prueba el aplicativo con algunos usuarios beta o de prueba para analizar si lo construido se adecua a lo planteado inicialmente y en busca de tomar sus opiniones y observaciones para en caso de ser mínimas puedan ser aplicadas al proyecto o añadidas a un requerimiento a futuro con el fin de cimentar las bases para un nuevo desarrollo y mejora de lo construido, debido a que el software debe estar en un proceso de renovación y actualización para evitar la obsolescencia antes del tiempo esperado.

5. Desarrollo del proyecto

5.1 Determinación de requerimientos.

Con la finalidad de llegar al cumplimiento de los objetivos planteados inicialmente en el proyecto, se definieron un listado de requerimientos funcionales y no funcionales que deben ser completados a cabalidad en la plataforma, los requerimientos funcionales fueron construidos y analizados con base en, lo que se expuso en el plan de proyecto como las necesidades planteadas por los usuarios potenciales del aplicativo en la comunidad universitaria, presentados y justificados en este documento, por lo tanto en el siguiente apartado se expresan y describen los requerimientos refinados. Los requerimientos no funcionales se tomaron en cuenta a lo planteado en la primera versión de este aplicativo, debido a que las nuevas integraciones y módulos realizados para este no deben afectar o minimizar el impacto a lo que se planteó inicialmente para el proyecto.

5.1.1 Requerimientos funcionales.

En este apartado se muestran los requerimientos funcionales del proyecto, en una tabla adaptada a la metodología con la identificación del requerimiento, nombre, la prioridad para el aplicativo y la descripción de su funcionalidad.

Tabla 1.*Requerimiento funcional: Registrar usuario por medios alternativos*

Requerimiento funcional	
Requerimiento	RF01
Nombre	Registro de usuarios por medios alternativos
Prioridad	Alta

Descripción	Los usuarios deberán poder registrarse en aplicativo empleando un sistema de registro single sign on, mediante el uso de una cuenta de una aplicación externa, como lo puede ser el registro mediante cuenta de google o el registro mediante cuenta de Facebook, usando un estándar como OAuth herramientas para la autenticación implementadas por estas dos plataformas.

Tabla 2. *Requerimiento funcional: Inicio de sesión por medios alternativos*

Requerimiento funcional	
Requerimiento	RF02
Nombre	Inicio de sesión por medios alternativos
Prioridad	Alta
Descripción	Los usuarios ya registrados en el aplicativo mediante el sistema single sign on, deberán poder autenticarse e iniciar sesión accediendo a su cuenta externa elegida al momento de registrar su usuario.

Tabla 3. *Requerimiento funcional: Acceso mediante invitado*

Requerimiento funcional	
Requerimiento	RF03
Nombre	Acceso mediante invitado
Prioridad	Alta

Descripción	Las personas deberán poder acceder al contenido del aplicativo sin la necesidad de crear una cuenta de usuario, esto a través de una opción de acceso como invitado en la pantalla de inicio de sesión.

Tabla 4. *Requerimiento funcional: Limitar cuenta invitado*

Requerimiento funcional	
Requerimiento	RF04
Nombre	Limitar cuenta invitado
Prioridad	Alta
Descripción	Las personas que accedan al aplicativo mediante la opción de acceso de invitado solo deberán poder visualizar los diferentes anuncios publicados, y no deberán poder crear nuevos anuncios, valorar, reportar anuncios. ni tampoco tendrán un perfil de usuario asociado a ellos.

Tabla 5. *Requerimiento funcional: Guardar anuncios como favoritos*

Requerimiento funcional	
Requerimiento	RF05
Nombre	Guardar anuncios como favoritos
Prioridad	Alta
Descripción	Los usuarios al visualizar un anuncio deberán tener a su disposición la opción de marcar un anuncio como favorito, al hacer esto se almacenará un

	acceso al anuncio en una sección especial dentro que aplicativo que contenga todos los anuncios marcados como favoritos.

Tabla 6. *Requerimiento funcional: Limitar cantidad de anuncios favoritos*

Requerimiento funcional	
Requerimiento	RF06
Nombre	Limitar cantidad de anuncios favoritos
Prioridad	Media
Descripción	La cantidad máxima de anuncios que un usuario podrá marcar como favoritos al mismo tiempo deberá ser de 30 anuncios.

Tabla 7. *Requerimiento funcional: Incorporar historial de visualización*

Requerimiento funcional	
Requerimiento	RF07
Nombre	Incorporar historial de visualización
Prioridad	Alta
Descripción	El usuario deberá poder acceder a una sección dentro del aplicativo donde pueda visualizar de forma ordenada los anuncios que él ha visualizado con anterioridad.

Tabla 8. *Requerimiento funcional: Desmarcar anuncios de favoritos*

Requerimiento funcional	
Requerimiento	RF08
Nombre	Desmarcar anuncios de favoritos
Prioridad	Media
Descripción	El usuario deberá poder desmarcar anuncios que haya marcado previamente como favoritos, y al excederse este límite al llegar a no tener anuncios por desmarcar el usuario deberá ser informado de ello.

Tabla 9. *Requerimiento funcional: Limitar cantidad de anuncios en historial*

Requerimiento funcional	
Requerimiento	RF09
Nombre	Limitar cantidad de anuncios en historial
Prioridad	Media
Descripción	La cantidad máxima de anuncios que se registrará en historial de un usuario deberá ser de no mayor a 100 anuncios y al excederse este limite el anuncio más antiguo será excluido del historial para dar lugar al más reciente.

Tabla 10. *Requerimiento funcional: Definir temas de interés*

Requerimiento funcional	
Requerimiento	RF10
Nombre	Definir temas de interés
Prioridad	Alta

Descripción	El usuario deberá poder definir, en una sección especial del aplicativo, cuáles son sus temas de interés, esto mediante un sistema de palabras clave, localizada en una sección del aplicativo.

Tabla 11. *Requerimiento funcional: Limitar cantidad de palabra clave tema de interés*

Requerimiento funcional	
Requerimiento	RF11
Nombre	Limitar cantidad de palabra clave tema de interés
Prioridad	Media
Descripción	La cantidad máxima de palabras clave relacionadas a sus temas de interés que podrá tener un usuario será de 5 palabras clave.

Tabla 12. *Requerimiento funcional: Eliminar palabra clave tema de interés*

Requerimiento funcional	
Requerimiento	RF12
Nombre	Eliminar palabra clave tema de interés
Prioridad	Media
Descripción	El usuario deberá poder eliminar sus palabras clave, previamente delimitadas, del listado de términos de su interés.

Tabla 13. *Requerimiento funcional: Recibir notificaciones según temas de interés*

Requerimiento funcional	
Requerimiento	RF13
Nombre	Recibir notificaciones según temas de interés
Prioridad	Alta
Descripción	El usuario deberá recibir notificaciones en el aplicativo al iniciar sesión sobre nuevos anuncios publicados que correspondan a sus temas de interés previamente señalados.

Tabla 14. *Requerimiento funcional: Activar o desactivar notificaciones*

Requerimiento funcional	
Requerimiento	RF14
Nombre	Activar o desactivar notificaciones
Prioridad	Media
Descripción	El usuario deberá poder decidir si desea activar o desactivar las notificaciones que recibirá del sistema en base a sus intereses, si se tiene desactivada la opción, al iniciar sesión no se mostraran los últimos anuncios con base a sus intereses.

Tabla 15. *Requerimiento funcional: Marcar anuncio como inapropiado*

Requerimiento funcional	
Requerimiento	RF15

Nombre	Marcar anuncio como inapropiado
Prioridad	Alta
Descripción	El usuario deberá poder reportar un anuncio como inapropiado si este lo considera necesario.

Tabla 16. *Requerimiento funcional: Confirmación de acciones reporte de anuncio*

Requerimiento funcional	
Requerimiento	RF16
Nombre	Confirmación de acciones reporte de anuncio
Prioridad	Media
Descripción	El usuario deberá poder recibir una confirmación de la acción a realizar con el fin de retroalimentarse en las acciones que realicé en el aplicativo

Tabla 17. *Requerimiento funcional: Modal de confirmación de permisos limitados invitado*

Requerimiento funcional	
Requerimiento	RF17
Nombre	Modal de confirmación de permisos limitados invitado
Prioridad	Media

Descripción	El usuario al ingresar como invitado deberá poder visualizar un modal cada vez que desee navegar por la aplicación, que le indique que el usuario tiene permisos limitados y permitirle redirigirse al registro o inicio de sesión.

5.1.2 Requerimientos no funcionales.

En este apartado se describen los aspectos no establecidos en el comportamiento funcional del sistema, pero de todas formas el usuario debe detallar de estos los que sean posibles visualizar.

Tabla 18. *Requerimiento no funcional: Escalabilidad*

Requerimiento no funcional	
Requerimiento	RNF01
Nombre	Escalabilidad
Prioridad	Alta
Descripción	Los cambios implementados en el aplicativo deben contar con la capacidad de ser ampliados o mejorados según sea el caso.

Tabla 19. *Requerimiento no funcional: Amigabilidad con el usuario*

Requerimiento no funcional	
Requerimiento	RNF02
Nombre	Usabilidad
Prioridad	Alta

Descripción	Las nuevas utilidades a implementar en el aplicativo deben ser de fácil entendimiento para el usuario, de tal forma que a un nuevo usuario le sea fácil dar uso a estas utilidades con una capacitación básica previa.

Tabla 20. *Requerimiento no funcional: Integridad de datos*

Requerimiento no funcional	
Requerimiento	RNF03
Nombre	Integridad de datos
Prioridad	Alta
Descripción	El ingreso de información en el interior del aplicativo no debe romper su integridad de datos, evitando registros duplicados, demasiado cortos o extensos, y velando por que la información requerida sea completa en su ingreso.

Tabla 21. *Requerimiento no funcional: Definición de cantidades máximas de registros por usuario*

Requerimiento no funcional	
Requerimiento	RNF04
Nombre	Definición de cantidades máximas de registros por usuario
Prioridad	Alta
Descripción	Los datos asociados al usuario, según sea el caso, deben tener definidos diferentes topes que delimiten la máxima cantidad de registros almacenados

	para el dato, como longitud máxima de su historial o la cantidad máxima de temas de interés que pueda tener.

Tabla 22. *Requerimiento no funcional: Acceso seguro*

Requerimiento no funcional	
Requerimiento	RNF05
Nombre	Acceso seguro
Prioridad	Alta
Descripción	El acceso del usuario a la plataforma en todos sus diferentes métodos debe ser seguro y privado, protegiendo la información de ingreso ante posibles amenazas.

Tabla 23. *Requerimiento no funcional: Compatibilidad con Android*

Requerimiento no funcional	
Requerimiento	RNF06
Nombre	Compatibilidad con Android
Prioridad	Alta
Descripción	El aplicativo móvil debe poder ser compatible y ejecutado en dispositivos Android con versión superior a Android 5.0 Lollipop.

5.2 Análisis y diseño del software

El análisis y diseño de software para el proyecto tenemos en cuenta las diferentes tareas para planificar el desarrollo del software, es importante que se tome toda la información disponible dada en los requerimientos y se pueda establecer con base en estos, la lógica del sistema, expresarla en diagramas y estructuras comunes usadas en la construcción de un sistema de software, tales como el uso de los diagramas en UML tales como los diagramas de casos de uso, diagramas de secuencia y diagramas de flujo que se emplearan para dar una cara más funcional al proyecto, incluyendo también el análisis a las interfaces gráficas iniciales desarrolladas en base a los diagramas construidos y enfocadas en establecer un entendimiento más visual de las funcionalidades al desarrollador.

5.2.1 Casos de uso

Por medio de los casos de uso, mostramos el proceso que se lleva a cabo por medio de la interacción de los usuarios con el sistema objetivo, el comportamiento de estos y las acciones que ellos pueden llevar a cabo, en las figuras 4,5 y 6, se presentan los casos de uso más relevantes y que dan valor al aplicativo móvil.

Figura 4. *Casos de uso del aplicativo, autenticación de usuarios*

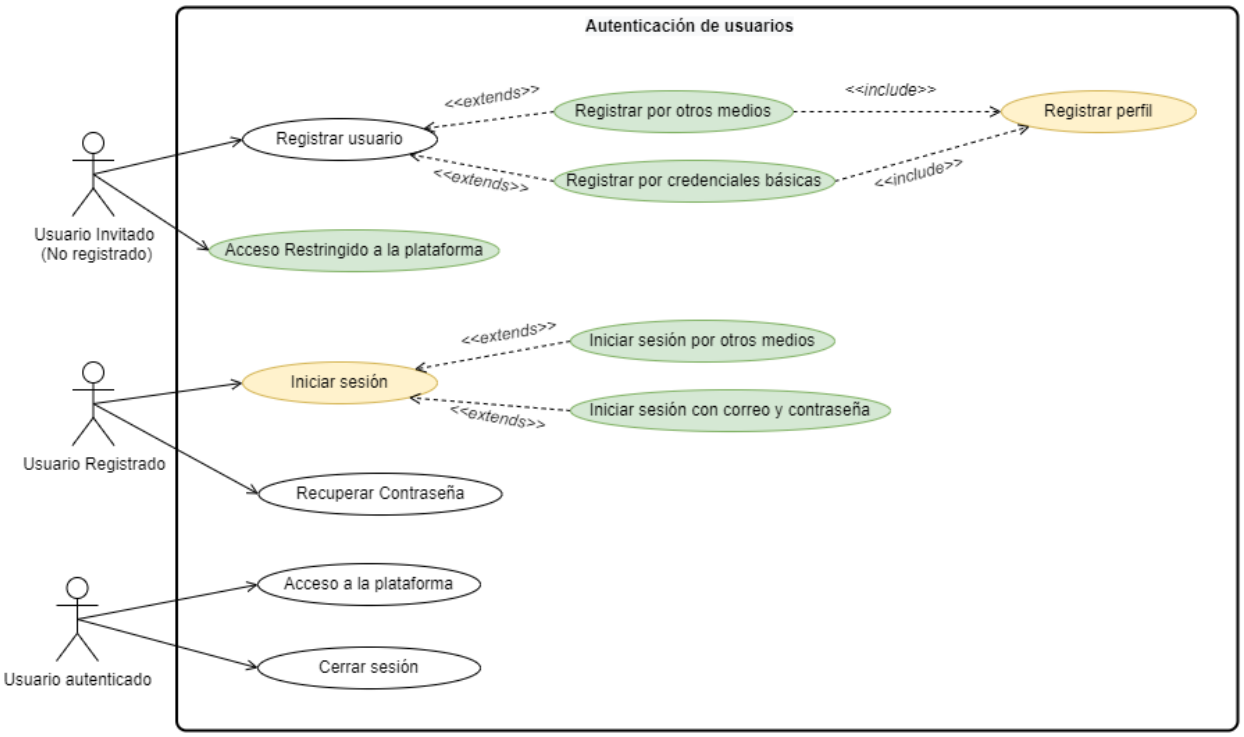


Figura 5. *Casos de uso del aplicativo, gestión del perfil*

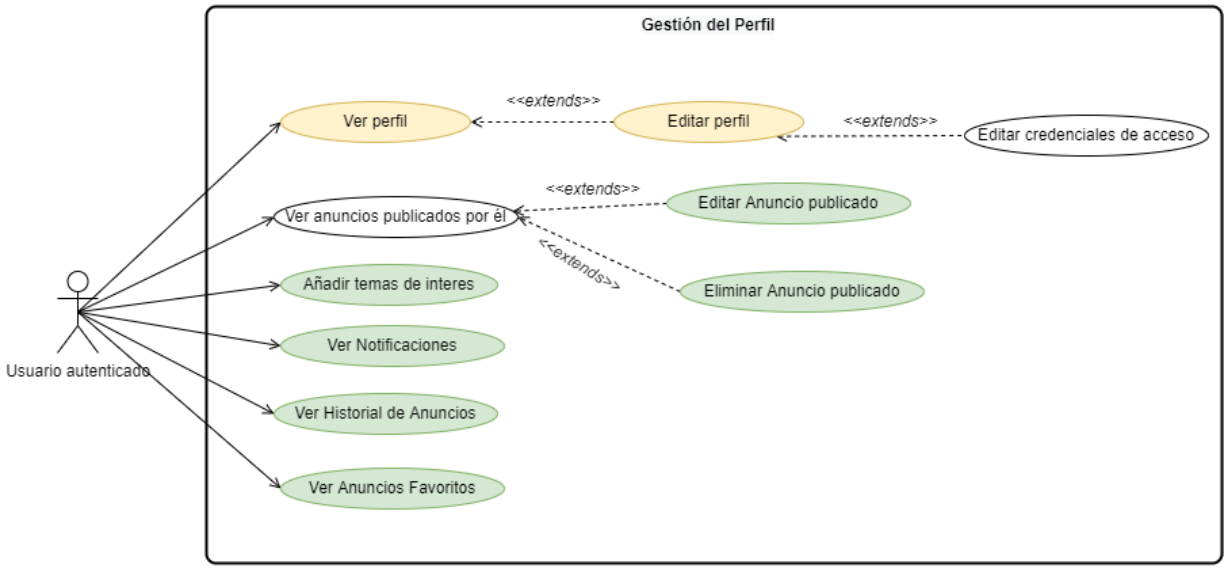
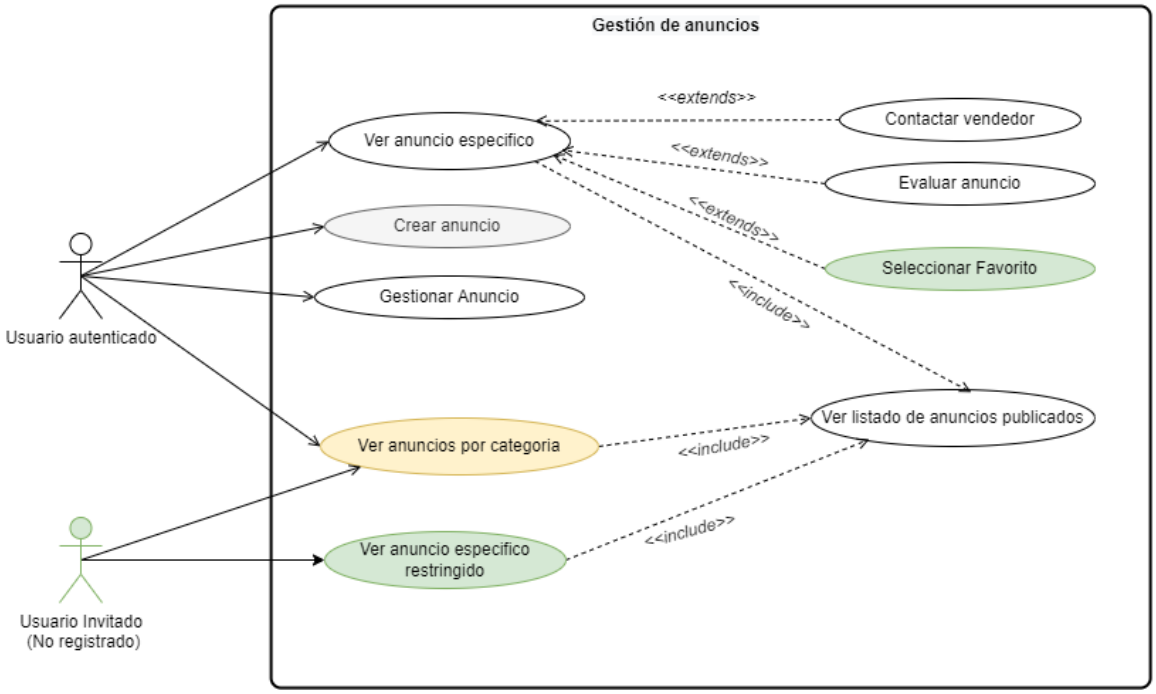


Figura 6. *Casos de uso del aplicativo, gestión de usuarios*



Los actores que conforman y pueden interactuar con el sistemas son 3, como se estableció en la primera versión del aplicativo, en las ilustraciones se muestra la modificación en los casos de uso que fueron construidos en la versión uno, ubicando con colores amarillo y verde para modificación y creación respectivamente de los elementos que conforman los casos de uso, donde el "Usuario no registrado" (invitado) ahora tiene la posibilidad de ingresar al aplicativo y visualizar los anuncios y hasta revisar un anuncio especifico pero de manera restringida, ya que como se muestra en el la Figura 6, el anuncio especifico restringido comprende solo su visualización, más no su calificación, agregación a favorito o el contacto con el vendedor directamente, el segundo actor en el caso de uso es el "Usuario Autenticado" el cual tiene todos los privilegios del invitado y además sin restricción para calificar, visualizar el anuncio, contactar al vendedor y seleccionar

un anuncio visualizado como favorito, además el usuario Autenticado puede iniciar sesión ya sea con cualquiera de sus cuentas de Google o Facebook como se especifica en la Figura 4 en el caso de uso “Iniciar sesión por otros medios” y puede en la gestión de su perfil agregar hasta 5 temas que le sea de interés para recibir o no notificación como se muestra en la Figura 5.

5.2.2 Tarjetas CRC

Las tarjetas CRC representan la colaboración entre clases, son un conjunto de tarjetas que permiten la ilustración de una clase, sus responsabilidades y sus colaboradores, en una labor de análisis inicial partiendo de las funcionalidades expresadas en los casos de uso, se procedió a hacer un listado de las clases que existen ya en el sistema actual del aplicativo y las nuevas que se agregarían en base a los nuevos requerimientos proporcionados, por lo tanto en las tarjetas CRC implementadas en la versión inicial del aplicativo se tomaron las tarjetas que serían alcanzadas por este nuevo desarrollo y se colocaron a continuación junto con las dos tarjetas CRC nuevas que surgieron como respuesta a lo planteado por los requerimientos actuales, estas son Vista y Reporte, las cuales tienen enmarcadas las responsabilidades y los colaboradores que le ayudaran a cumplir o formarán parte del cumplimiento de sus deberes.

Tabla 24. *Tarjeta CRC Usuario*

Clase: Usuario	
Responsabilidades	**Colaboradores**
• Ingresar a la aplicación por otros medios • Registrar en la aplicación por otros medios • Validar tipo de sesión	

Tabla 25. *Tarjeta CRC Perfil*

Clase: Perfil	
Responsabilidades	**Colaboradores**
• Mostrar un listado de intereses. • Agregar Interés a la lista de intereses. • Eliminar Interés de la lista de intereses. • Mostrar un listado de anuncios favoritos • Remover un anuncio favorito • Agregar un anuncio favorito • Mostrar una lista de notificaciones en base a intereses • Limpiar la lista de notificaciones	• Usuario • Anuncio • Adjunto • Voto

Tabla 26. *Tarjeta CRC Reporte*

Clase: Reporte	
Responsabilidades	**Colaboradores**
• Registrar un reporte de un anuncio en la aplicación. • Eliminar un reporte registrado a un anuncio.	• Anuncio • Perfil

Tabla 27. *Tarjeta CRC Vista*

Clase: Vista	
Responsabilidades	**Colaboradores**
• Registrar una visualización de un anuncio • Mostrar un listado de anuncios filtrado por el perfil que los visualizó. • Mostrar de manera descendente la lista de anuncios por perfil que visualizó.	• Anuncio • Perfil

5.2.3 Diagramas de secuencia

En la presente sección se muestran los diagramas de secuencia, ubicados en el Lenguaje Unificado de Modelado (UML), en los diagramas de interacción, estos tipos de diagramas y en específico el diagrama de secuencia, modelan como los diferentes elementos que componen el sistema se unen para su funcionamiento, los diagramas de secuencia indican al ingeniero de software como funcionan los requisitos del sistema a diseñar y sirven para la documentación de procesos presentes en las capas del sistema. A continuación, se presentan los diagramas de secuencia diseñados para las funcionalidades más importantes del aplicativo.

Figura 7. *Diagramas de secuencia del aplicativo, Agregar favorito*

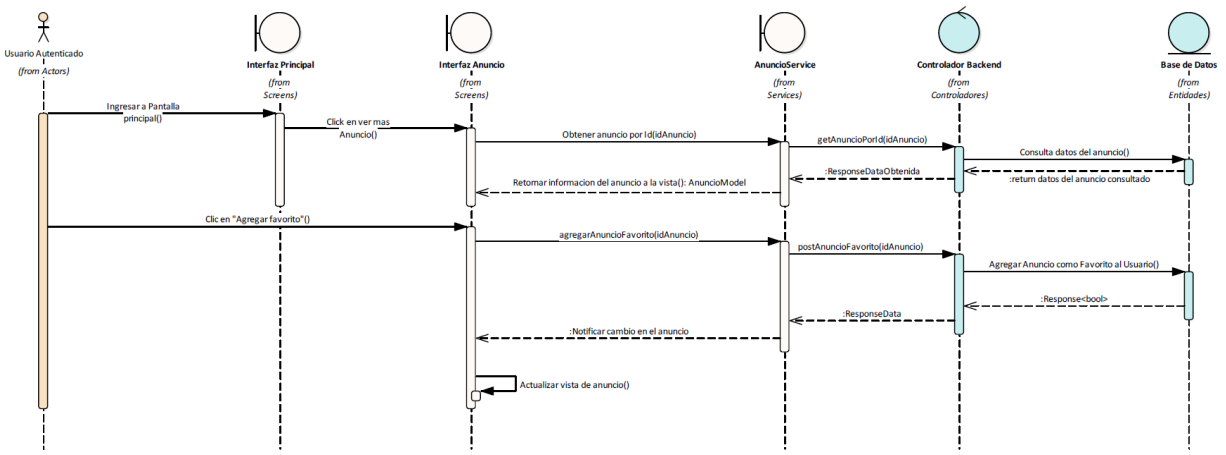


Figura 8. *Diagramas de secuencia del aplicativo, Reportar anuncio*

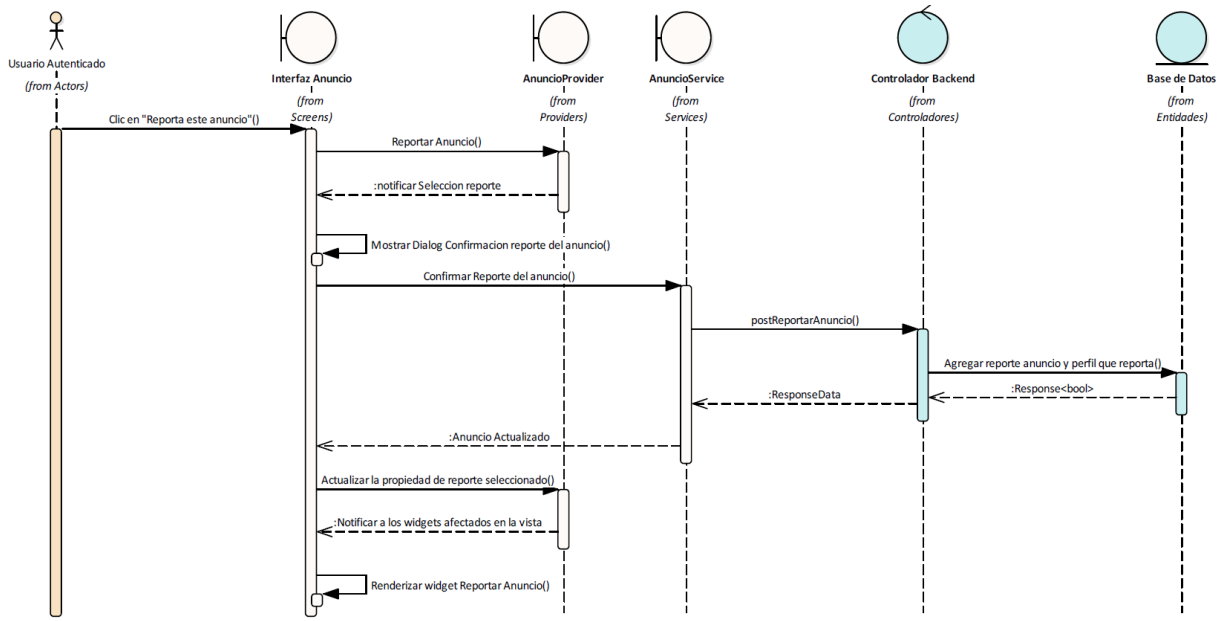
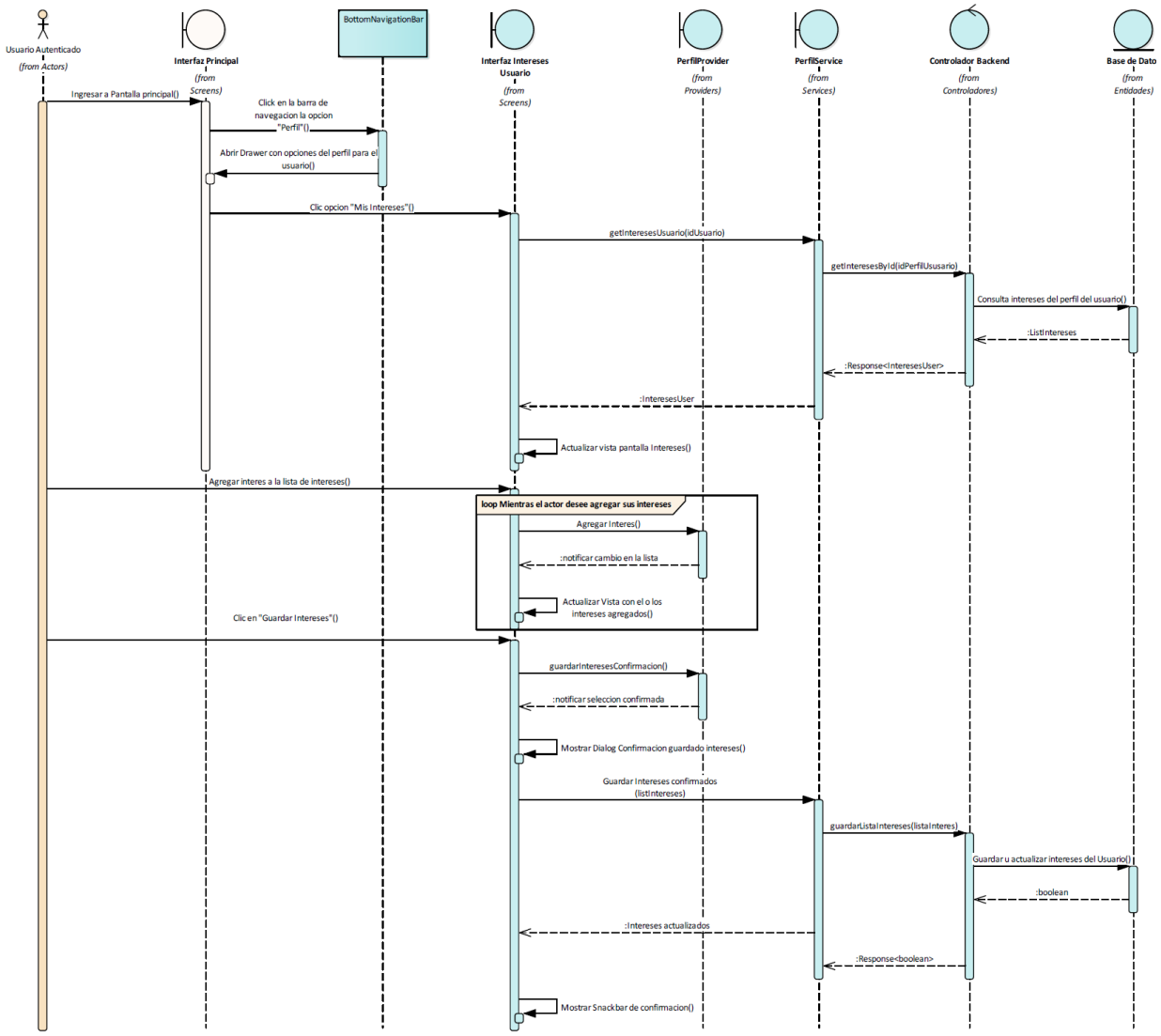


Figura 9. *Diagramas de secuencia del aplicativo, Agregar intereses*

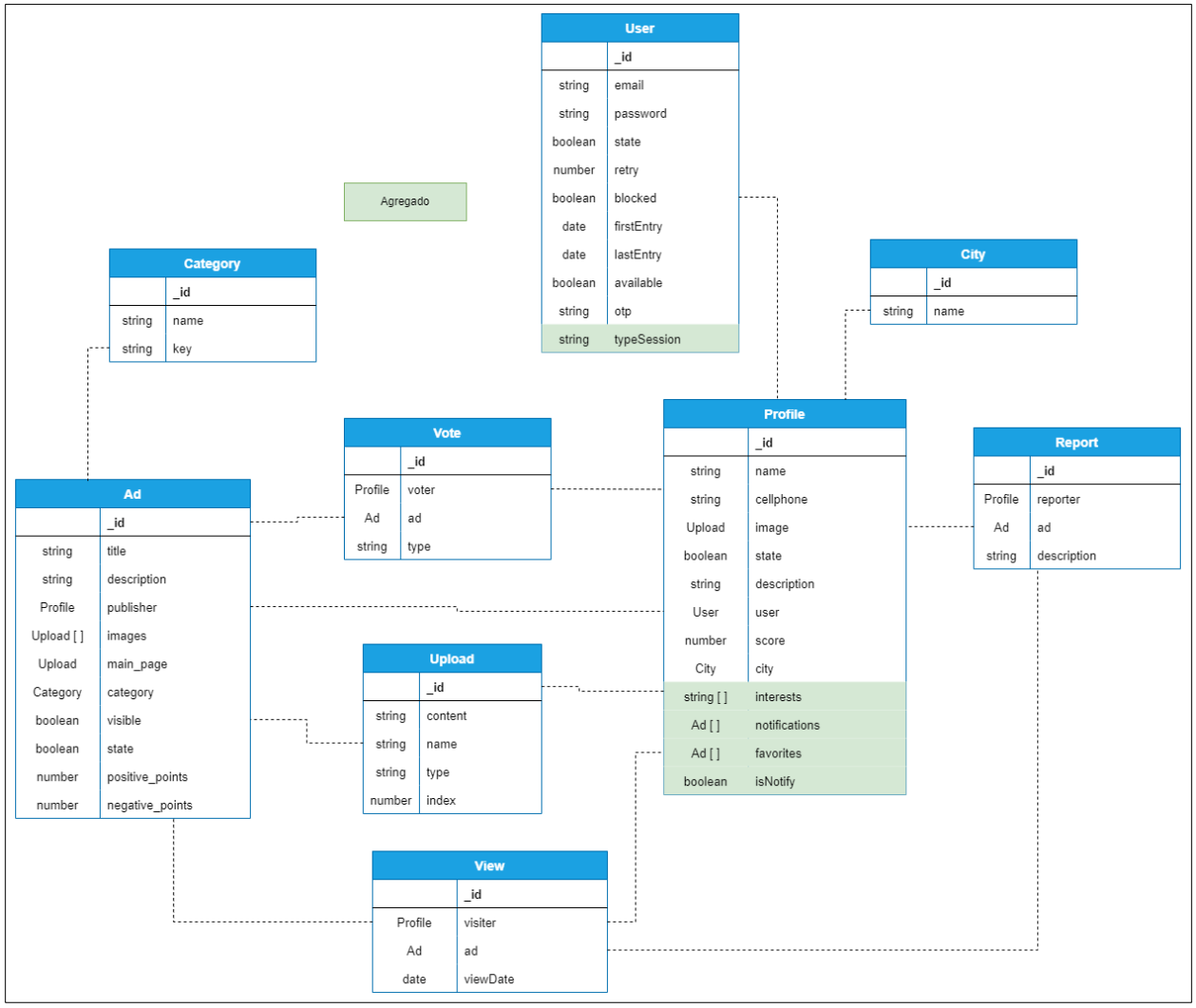


5.2.4 Modelo de base de datos

El siguiente modelo de base de datos presenta un total de 9 tablas en comparación con la versión 1 que su planteamiento inicial estableció 7 tablas, con las 2 tablas agregadas View y Report se cumplirá con los objetivos de almacenar la información requerida para que el funcionamiento del aplicativo.

5.2.4.1 Diagrama modelo de base de datos

Figura 10. *Diagrama de base de datos*



5.2.4.1 Descripción del modelo.

A continuación, se muestra una descripción de las tablas del modelo de base de datos utilizado en el sistema inicial y los campos agregados en las tablas diseñadas para este fin se muestran en color verde, se realiza la agregación de dos tablas nuevas como lo son View y Report necesarias para el cumplimiento de los objetivos.

Tabla User. Esta tabla almacena los datos de acceso del usuario, correo, contraseña, hora de acceso, establecidos desde la versión inicial del aplicativo, ahora con la modificación de que se agrega el tipo de sesión con la que se registró el usuario en el sistema.

Tabla Profile. Esta tabla almacena los datos propios del perfil de cada uno de los usuarios, siendo los datos importantes el nombre, referencia, contacto al whatsapp, descripción del perfil, entre otros, además se agrega como campos nuevos intereses, notificaciones, favoritos y un campo booleano que permitirá controlar si se desea o no ser notificado.

Tabla City. Esta tabla se mantiene sin modificar almacenando el listado de ciudades donde se habilita el aplicativo.

Tabla Ad. Esta tabla no se modifica mantiene la información de los datos relacionados a los anuncios, como título, descripción, puntos positivos y negativos, referencia del perfil que lo publicó.

Tabla Vote. Esta tabla no se modifica permite almacenar las referencias de los votos realizados y el tipo, si fue positivo o negativo donde se relaciona un perfil de usuario con un anuncio.

Tabla Upload. Esta tabla no se modifica con respecto a la versión inicial, donde se relaciona un listado de imágenes que contiene el anuncio, relacionado al perfil del que publica, en este upload se mantiene la imagen almacenada en formato de base64.

Tabla Category. Esta tabla no se modifica permite mantener la información de las categorías disponibles dentro de la aplicación y son las que permiten relacionarse a un anuncio para identificar a cuál pertenece.

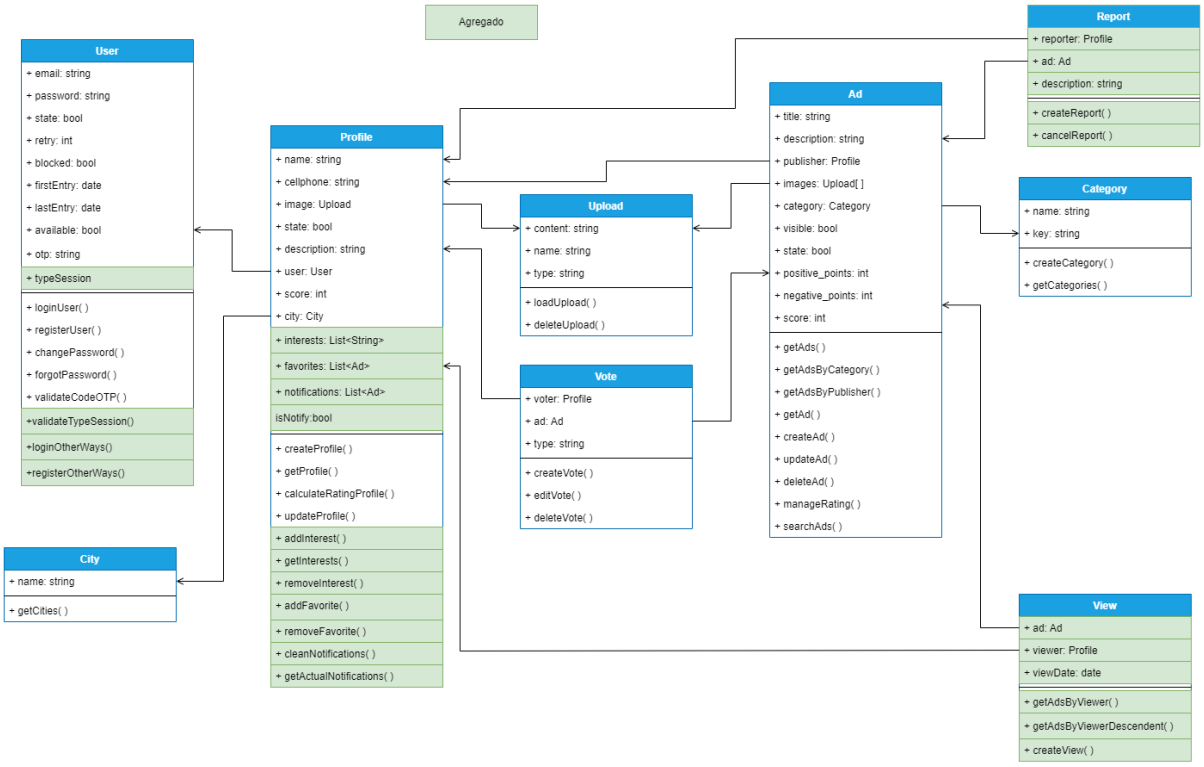
Tabla Report. Esta tabla es nueva en la aplicación y permite almacenar el id de la persona que reporta un anuncio y el anuncio que sufre el reporte, esto para mantener un control de la persona que realiza el reporte de una publicación no deseada.

Tabla View. Esta tabla es nueva en la aplicación y permite que se almacene el id de perfil que visualiza un anuncio especifico, y el id de este anuncio esta referencia se usa más adelante para definir el historial de anuncios asociado a un perfil de usuario.

5.2.5 Diagrama de clases

En el diagrama se presentan las clases nuevas agregadas como Report y View, sus respectivos métodos y atributos y con un color verde se enfoca lo que se modificó u agregó para el desarrollo de esta nueva versión del aplicativo.

Figura 11. *Diagrama de clases del aplicativo*



5.3 Diseño de la interfaz gráfica

El análisis y diseño de software para el proyecto tenemos en cuenta las diferentes tareas para planificar el desarrollo del software, es importante que se tome toda la información disponible dada en los requerimientos y se pueda establecer con base en estos, la lógica del sistema, expresarla en diagramas y estructuras comunes usadas en la construcción de un sistema de software, tales como el uso de los diagramas en UML tales como los diagramas de casos de uso, diagramas de secuencia y diagramas de flujo que se emplearan para dar una cara más funcional al proyecto, incluyendo también el análisis a las interfaces gráficas iniciales desarrolladas en base a los

diagramas construidos y enfocadas en establecer un entendimiento más visual de las funcionalidades al desarrollador.

5.3.1 Bases Generales de la interfaz

5.3.1.1 Nombre y logo del aplicativo

Continuando con las pautas establecidas en la primera versión del aplicativo, se decidió mantener el nombre y logo de la versión inicial, ya que se considera que este logra ilustrar la misión de la aplicación de una manera minimalista y clara.

Figura 12. *Logo y nombre del aplicativo*

5.3.1.2 Colores del aplicativo

Continuando con el objetivo de la primera versión del aplicativo se decidió mantener la paleta de colores empleada en la mayoría de los lugares del aplicativo, así mismo, para otorgar a él aplicativo un estilo renovado y más claro se decidió modificar los colores de los iconos de las categorías.

Tabla 28. *Colores principales del aplicativo*

Color	RGB	HEX
	R: 103 G: 185 B: 62	#67B93E
	R: 255 G: 185 B: 255	#FFFFFF
	R: 62 G: 185 B: 107	#3EB96B

5.3.1.3 Iconografía principal

Se describe toda la iconografía empleada en el aplicativo junto con la coloración que toma en el aplicativo y su descripción, respecto a la primera versión de este aplicativo el estilo visual y la paleta de colores de algunos iconos fue modificado, como se evidencia en la tabla 29.

Tabla 29. *Iconografía*

Icono	Color	Descripción
		Icono empleado en la categoría general y punto central del aplicativo, esta categoría representa el punto de inicia de la interfaz

		de anuncios del aplicativo, este icono está presente en el listado de categorías del aplicativo.
		Icono empleado en la categoría principal, todos los anuncios publicados en la plataforma pertenecen a esta categoría, este icono está presente en el listado de categorías del aplicativo.
		Icono empleado en la categoría destinada para las publicaciones que quieran dar a conocer contenido que incluya productos alimenticios o servicios de alimentación, este icono está presente en el listado de categorías del aplicativo.
		Icono empleado en la categoría creada para mantener las publicaciones relacionadas al mundo del deporte y la actividad física.
		Icono empleado en la categoría hecha para ser un espacio donde los usuarios puedan publicar/buscar ofertas de empleo, o dar a conocer programas de prácticas laborales o pasantías empresariales, este icono está presente en el listado de categorías del aplicativo.
		Icono empleado en la categoría creada para que los usuarios puedan publicar anuncios que estén destinados al ofrecimiento de servicios varios, tal y como lo puede ser el de reparación, traducciones, transporte, etc. Este icono está presente en el listado de categorías del aplicativo.

		Icono empleado en la categoría destinada para contener los anuncios que busquen dar a conocer artículos de ropa, calzado, moda y vestimenta en general, este icono está presente en el listado de categorías del aplicativo.
		Icono empleado en la categoría para incluir anuncios cuyo propósito sea promocionar piezas artísticas o de literatura de propiedad del usuario, este icono está presente en el listado de categorías del aplicativo.
		Icono empleado en la categoría dispuesta para la publicación de anuncios que busquen dar a conocer espacios en arriendo o artículos en alquiler, este icono está presente en el listado de categorías del aplicativo.
		Icono empleado en la categoría destinada para la publicación de artículos electrónicos, nuevos o de segunda mano, este icono está presente en el listado de categorías del aplicativo.
		Icono empleado en la categoría dispuesta para contener anuncios cuyo propósito sea el de ofrecer servicios de tutorías, clases y enseñanza en general, este icono está presente en el listado de categorías del aplicativo.
		Icono presente en la pantalla inicial del aplicativo, este icono busca representar la funcionalidad de registro del aplicativo.

	Ninguno	Icono de Google presente en la pantalla de inicio de sesión del aplicativo, este icono indica el acceso al inicio de sesión vía cuenta de Google.
		Icono de Facebook presente en la pantalla de inicio de sesión del aplicativo, este icono indica el acceso al inicio de sesión vía cuenta de Facebook.
		Icono presente en la pantalla de inicio de sesión del aplicativo, este icono busca apoyar visualmente la funcionalidad de inicio de sesión mediante el email registrado en el aplicativo.
		Icono presente en la pantalla de inicio de sesión del aplicativo, este icono busca indicar el acceso a la funcionalidad de inicio de sesión como invitado.
		Icono de inicio hallado en la barra de navegación inferior presente durante la navegación por el aplicativo, este icono busca apoyar la funcionalidad de volver a la vista principal de anuncios.
		Icono de búsqueda presente en la barra de navegación inferior y en la barra superior del aplicativo, este icono indica el acceso a la funcionalidad de búsqueda de anuncios del aplicativo.
		Icono que busca indicar el acceso a la funcionalidad de publicación de un nuevo anuncio, este icono está presente en la barra de navegación inferior del aplicativo.

		Icono que indica el acceso al perfil de usuario, este icono está presente en la barra de navegación inferior del aplicativo.
		Icono que busca indicar la interacción con la funcionalidad almacenamiento de favoritos del aplicativo, este icono está sente en la barra de navegación inferior y en el menú lateral no acceso al listado de anuncios guardados como favoritos, más también se encuentra presente en la visualización de ncios indicando la posibilidad de añadir un anuncio a favoritos.
		Icono presente en la vista de anuncio al agregar un anuncio como favoritos, este icono busca indicar que un anuncio ya fue agregado al listado de favoritos.
		Icono de WhatsApp presente en la vista anuncios del aplicativo, este icono busca indicar el acceso al contacto directo, vía WhatsApp, con el anunciante de la publicación.
		Icono “like” presente en la vista anuncios del aplicativo, este icono permite acceder la funcionalidad de valorar positivamente un anuncio, la coloración de este icono puede variar según sea el estado de esta acción, según si el usuario a aportado un voto positivo o no.

	Icono “dislike” presente en la vista anuncios del aplicativo, este icono permite acceder la funcionalidad de valorar negativamente un anuncio, la coloración de este icono puede variar según sea el estado de esta acción, según si el usuario a aportado un voto negativo o no.
	Icono de bloqueo presente en la vista de anuncios del aplicativo, este icono busca apoyar visualmente la funcionalidad del aplicativo de reporte de anuncios como inadecuados.
	Icono presente en el menú lateral del aplicativo, este icono busca indicar el acceso al listado de publicaciones propias realizadas por el usuario, esta opción brinda también el acceso al perfil de usuario.
	Icono “notificación” presente en la vista del menú lateral del aplicativo, este icono indica el acceso al apartado de notificaciones del aplicativo.
	Icono “listado” presente en la vista del menú lateral del aplicativo, este icono busca indicar el acceso al apartado y funcionalidad de listado de intereses presente en el aplicativo.
	Icono “reloj” presente en el menú lateral del aplicativo, este icono busca apoyar visualmente el acceso al apartado y registro del historial del aplicativo.

		Icono “comentario” presente en el menú lateral del aplicativo, este icono busca indicar el acceso a la funcionalidad que permite el envió de comentarios a los desarrolladores de la app.
		Icono “información” presente en el menú lateral del aplicativo, este icono busca apoyar visualmente el acceso al aparte de más información sobre aplicativo lugar donde se puede conocer información detallada sobre la aplicación.
		Icono de “salida” presente en el menú lateral del aplicativo, este icono busca apoyar visualmente la funcionalidad de cierre de sesión y de salida del aplicativo.
		Icono “correcto” que busca indicar el correcto funcionamiento de algunas funcionalidades del aplicativo, este icono puede ser encontrado en el mensaje de alerta luego de iniciar sesión de forma exitosa.
		Icono presente en algunos de los diferentes mensajes de alerta del aplicativo, este icono busca indicar que el mensaje contiene información importante para el usuario.

	Icono presente en algunos de los diferentes mensajes de alerta del aplicativo, este icono busca indicar que la alerta contiene información muy importante, debido a un posible error o que la acción que se está realizando no tiene marcha atrás.

5.3.2 Vistas del diseño Inicial

Continuando con el mismo esquema de diseño empleado en el desarrollo de la primera versión del aplicativo, se realizó un diseño inicial que sentará las bases y definiera como se cumplirán los requerimientos planteados en el proyecto a través de la interfaz gráfica, para este fin fueron elaborados las siguientes vistas, las cuales en el marco de este proyecto será referidas como diseño inicial.

Figura 13. *Vista inicios de sesión*

Buscando el cumplimiento de los requerimientos funcionales RF01, RF02 y RF03 se plantea la modificación de la presente vista, ver figura 13, incorporando a ella el registro y acceso mediante cuenta de Google, cuenta de Facebook y el acceso como invitado, junto con la opción de inicio de sesión tradicional implementada en la versión inicial del aplicativo.

Figura 14. *Vista invitado*

Si el usuario decide acceder como invitado al aplicativo se tendrá que encontrar con una vista similar a esta, desde la cual solo podrá visualizar los diferentes anuncios que contiene el aplicativo, pero no deberá poder publicar anuncios ni contactar con los anunciantes de los mismos, teniendo deshabilitada esta opción, dando cumplimiento al requerimiento funcional RF04.

Figura 15. *Vista menú invitado*

Como parte de las limitaciones que posee una cuenta de invitado, este al desplegar el menú del aplicativo solo deberá poder encontrarse con las opciones de información de la aplicación y la opción de cerrar sesión.

Figura 16. *Vista menú principal*

El usuario al iniciar sesión por cualquier método diferente del ingreso como invitado deberá poder encontrarse este menú lateral, desde el cual podrá acceder a las nuevas funcionalidades incorporadas en el aplicativo, tales como; la gestión de las notificaciones del aplicativo, dando cumplimiento a los requerimientos funcionales RF13 y RF14; el listado de anuncios marcados como favoritos, cumpliendo el requerimiento funcional RF05; la sección para definir sus temas de interés, buscando cumplir el requerimiento funcional RF10; el acceso a el historial de los anuncios previamente visualizados, apuntando al requerimiento funcional RF07; junto con las opciones ya presentes en la primera versión del aplicativo como el acceso al perfil de usuario, la sección de envió de comentarios, ver información sobre la aplicación y el botón de cierre de sesión.

Figura 17. *Vista anuncio*

El usuario al visualizar un anuncio se encontrará con una vista renovada respecto a la primera versión del aplicativo, ver Figura 17, ahora deberá poder visualizar una opción para marcar el anuncio como inadecuado, opción a la cual podrá recurrir según sea el caso, buscando esto el cumplimiento del requerimiento funcional RF15, además de esta nueva función, el usuario a través de un botón deberá poder marcar o desmarcar el anuncio como favorito, opción que al ser marcada anexará el anuncio a la lista de anuncios guardados, a la cual es posible acceder desde el menú

lateral, apuntando esta parte del diseño al cumplimiento de los requerimientos funcionales RF05 y RF08.

Figura 18. *Vista favoritos*

Desde el menú lateral el usuario deberá poder acceder a una vista con las características de la figura 18, los anuncios marcados previamente como favoritos deberán ser agregados a esta sección especial la cual solo podrá ser accedida por el usuario y deberá tener un límite máximo de anuncios guardados, buscando con esto el cumplimiento de los requerimientos funcionales RF05 y RF06, también los anuncios se listarán de acuerdo a su fecha de publicación.

Figura 19. *Vista historial*

Desde el menú lateral se deberá poder acceder esta sección, presente en la Figura 19, desde la cual serán listados cronológicamente los diferentes anuncios que ha visualizado el usuario durante el uso del aplicativo, teniendo esta sección una cantidad máxima de anuncios que pueden ser guardados, apuntando al cumplimiento de los requerimientos funcionales RF07 y RF09.

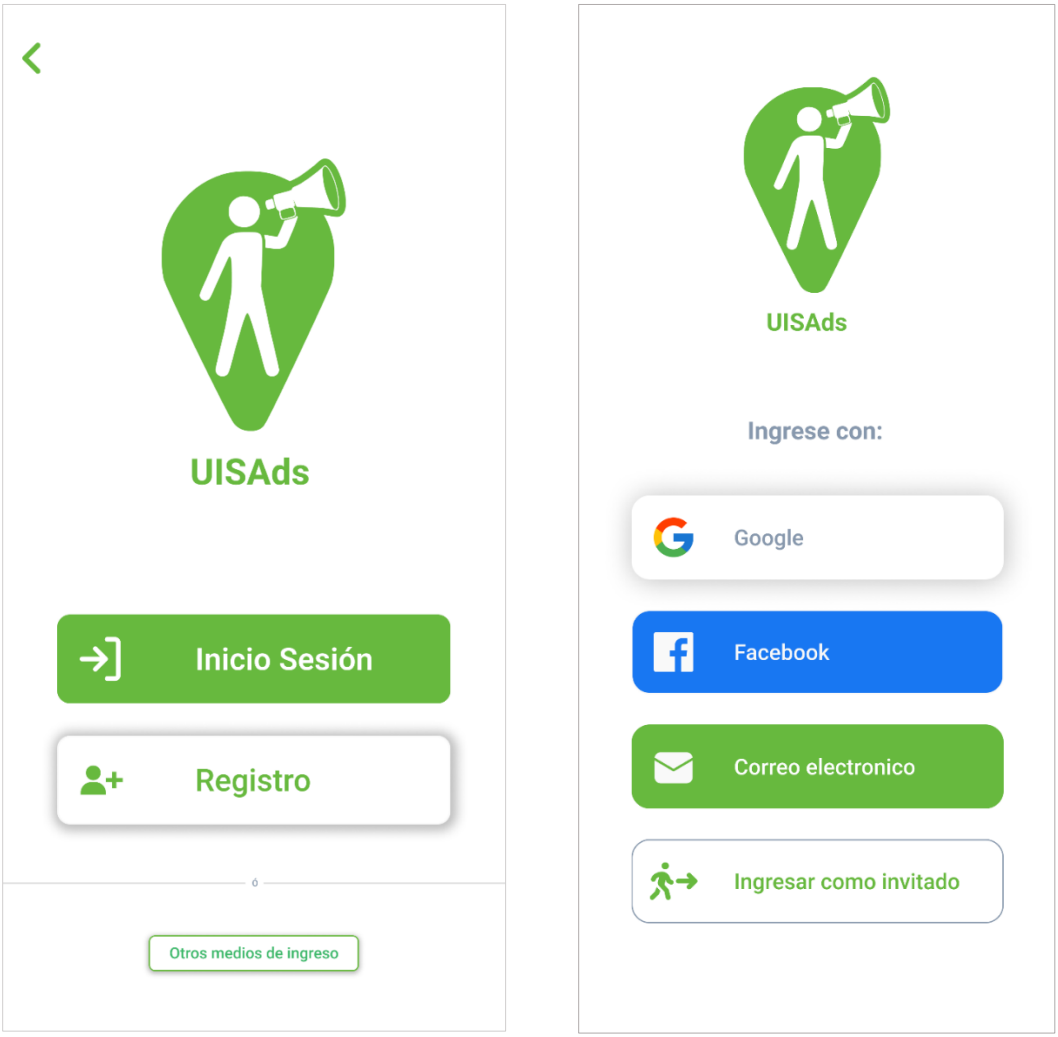
Figura 20. *Vista intereses*

Desde el menú lateral el usuario deberá poder acceder a la sección de intereses, presente en la Figura 20, desde la cual el usuario podrá ingresar términos de su interés, los cuales el sistema del aplicativo tendrá en cuenta al momento de enviarle notificaciones, buscando con esto cumplir los requerimientos funcionales RF10, RF11, RF12, RF13.

5.3.3 Vistas diseño final

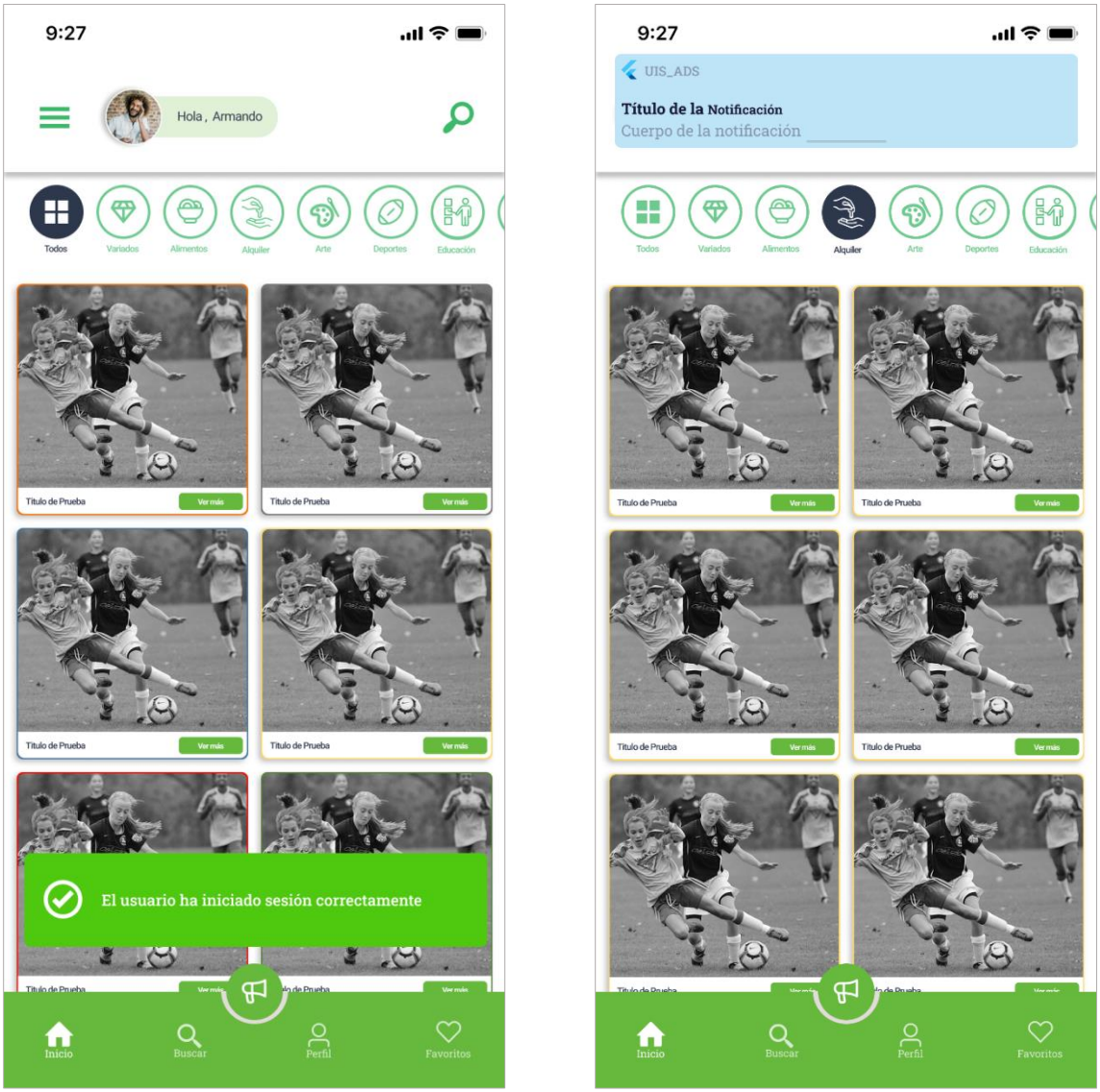
Continuando con la metodología propuesta en la primera versión del aplicativo (Triana & Parra, 2022), se elaboró un segundo diseño de interfaz de usuario, tomando las bases definidas al interior del diseño inicial y apuntando a ser más cercano a la interfaz final que vera el usuario en el aplicativo, para la creación de este segundo diseño fue empleada la herramienta de generación de prototipos web *figma.com.*

Figura 21. *Vista pantalla inicio e inicio de sesión versión final*



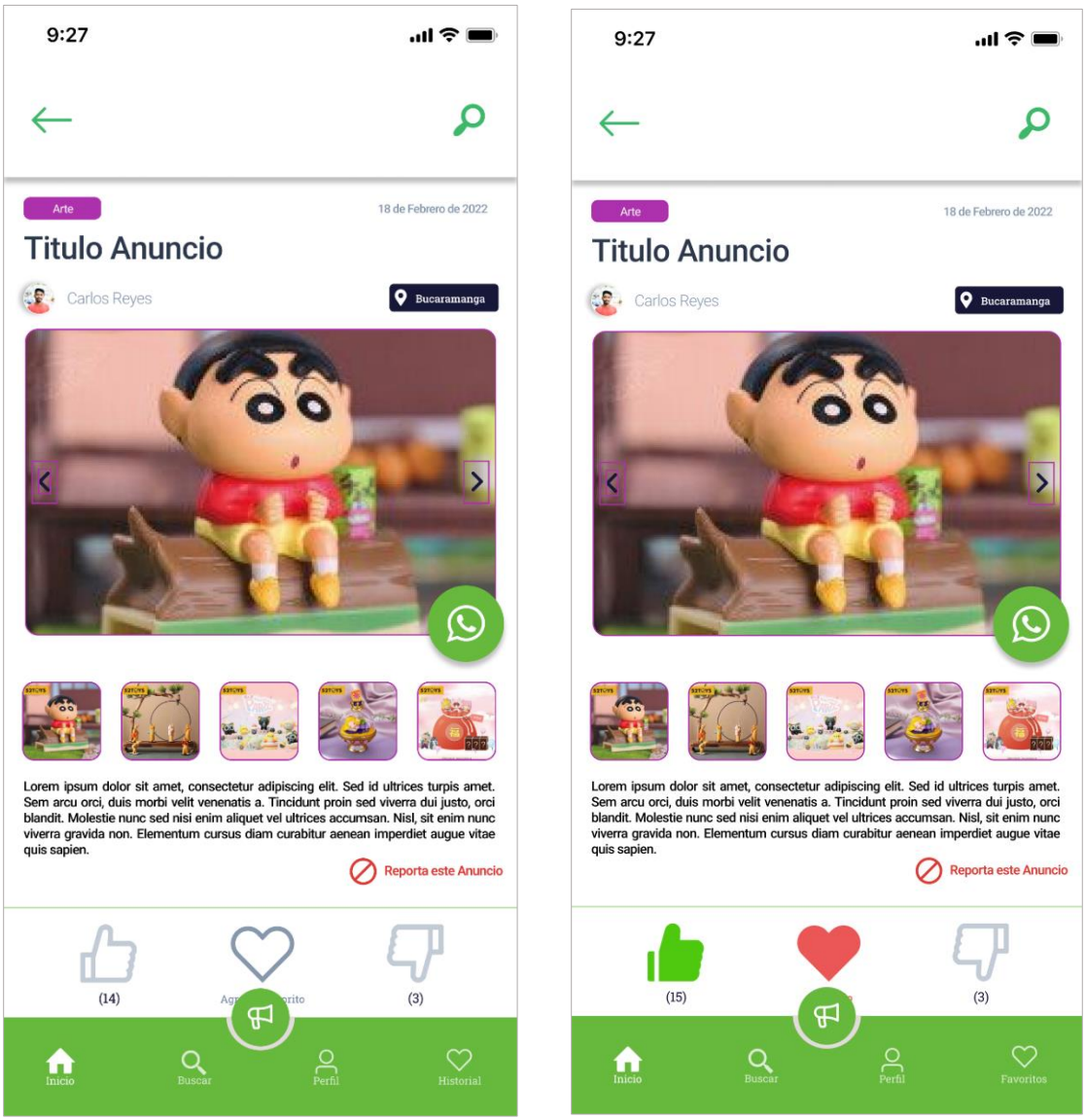
Apuntando a una mejora gráfica y buscando una mejor visualización estética de la vista se desarrollaron las vistas presentes en la Figura 21, respecto a la primera versión del aplicativo fue modificado el estilo de los botones de acceso junto con esto se incorporaron como formas alternas de inicio de sesión el acceso mediante cuenta de Google o Facebook, además de esto el acceso como invitado, buscando cumplir los requerimientos funcionales del aplicativo.

Figura 22. *Vista categoría versión final*



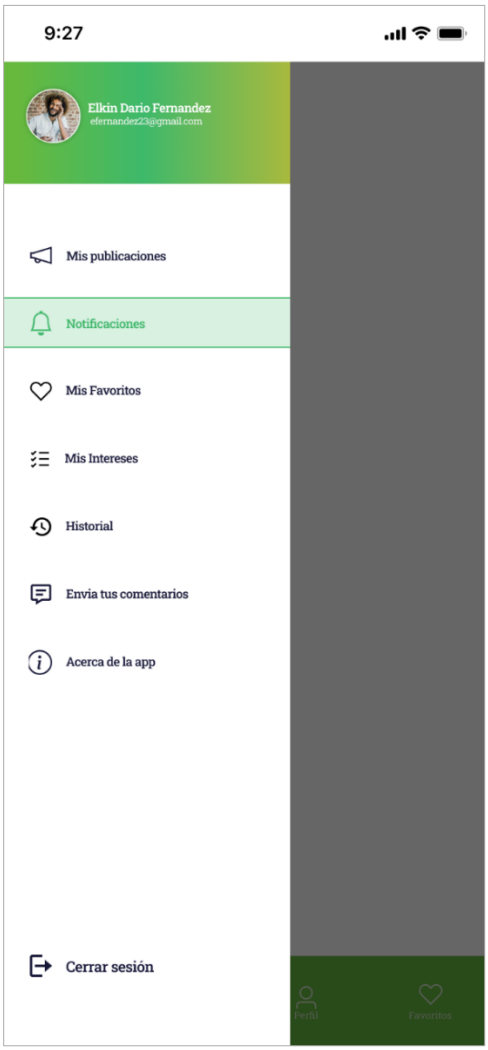
Al momento en el cual el usuario inicie sesión encontrará la vista de la categoría principal, ver Figura 22, siguiendo el funcionamiento de la primera versión del aplicativo, pero a diferencia de esta primera versión el usuario encontrará un mensaje de bienvenida notificándole el correcto inicio de sesión, también el usuario recibirá una notificación a su dispositivo informándole sobre los nuevos anuncios publicados según sus temas de interés previamente definidos, además de esto el usuario podrá ver varios cambios implementados en esta vista, como un rediseño de los logos empleados para las categorías junto con una mejora en la barra de navegación inferior, siendo incorporado en ella el botón de publicación de un nuevo anuncio, junto con la posibilidad de acceder a la sección donde quedan almacenados los anuncios que haya marcado como favoritos con anterioridad.

Figura 23. *Vista anuncio versión final*



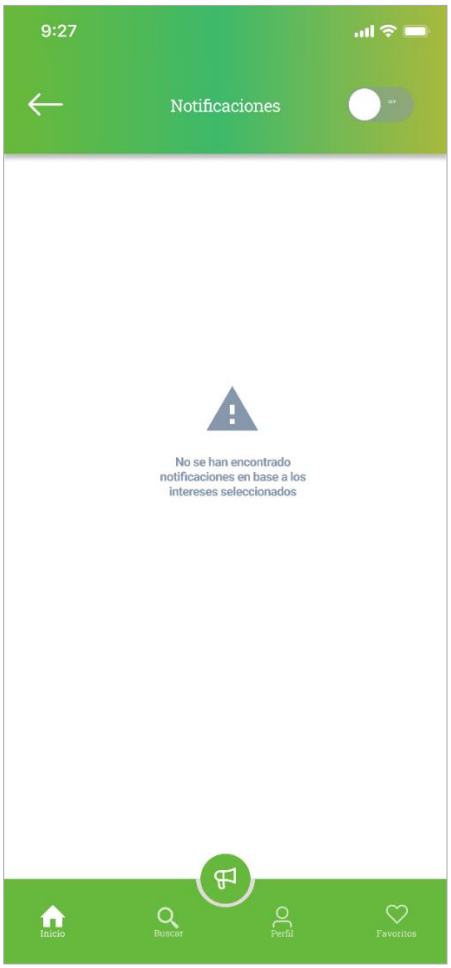
En la Figura 23 tenemos esta nueva versión de la vista anuncio, donde fueron agregadas nuevas funcionalidades apuntando a cumplir los requerimientos funcionales del proyecto, en torno a esta vista fue agregada una opción para el reporte del anuncio, en caso de que el usuario considere que este es inadecuado o incumple las normas comunitarias de la institución, además de esta nueva funcionalidad fue añadido un botón en la parte central, con el cual el usuario podrá añadir el anuncio a la sección de anuncios favoritos del aplicativo.

Figura 24. *Vista menú lateral versión final*



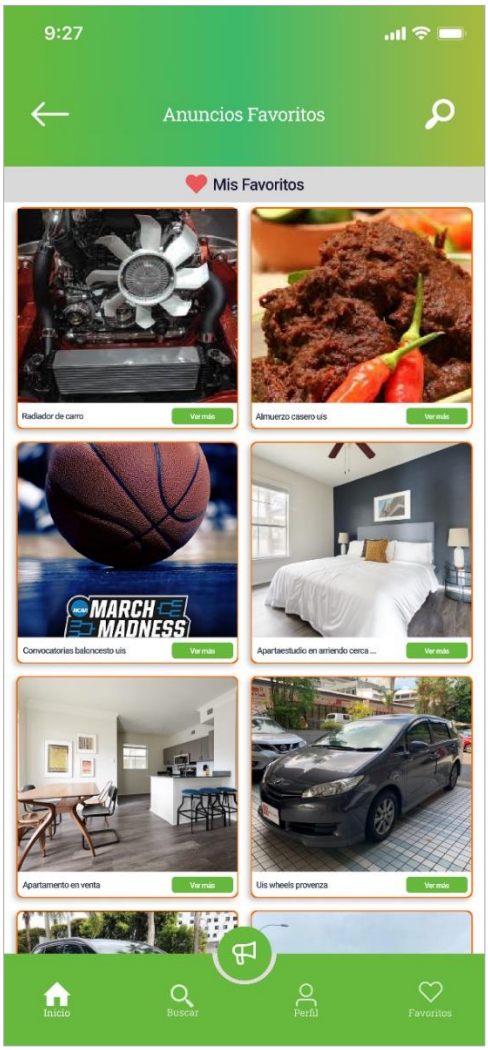
Como cambios en esta segunda versión del aplicativo fueron incorporados en el menú lateral, ver figura 24, nuevas opciones según los requerimientos del proyecto, como el acceso al apartado de gestión de notificaciones a través de la opción “notificaciones” en el menú, los accesos a las secciones de favoritos e historial y finalmente las opciones ya presentes en la primera versión del aplicativo como la opción de envío de comentarios a los desarrolladores, la opción para conocer más datos sobre la aplicación y la opción de cerrado de sesión.

Figura 25. *Vista apartado notificaciones*



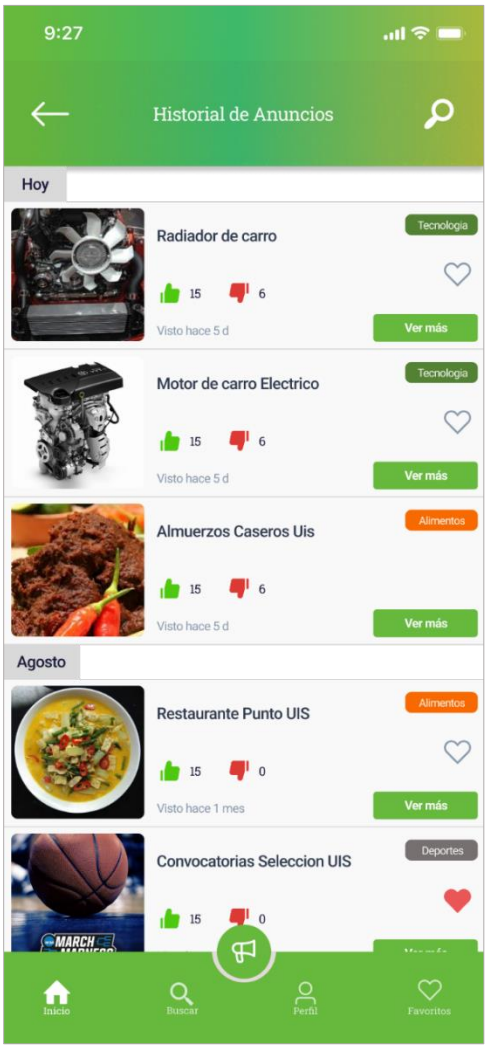
Si en el menú lateral el usuario opta por acceder a la opción “Notificaciones” accederá a la presente vista, ver figura 25, desde la cual podrá activar o desactivar el envió de notificaciones por parte del aplicativo a su dispositivo además de visualizar las ultimas notificaciones generadas según sus intereses.

Figura 26. *Vista favoritos versión final*



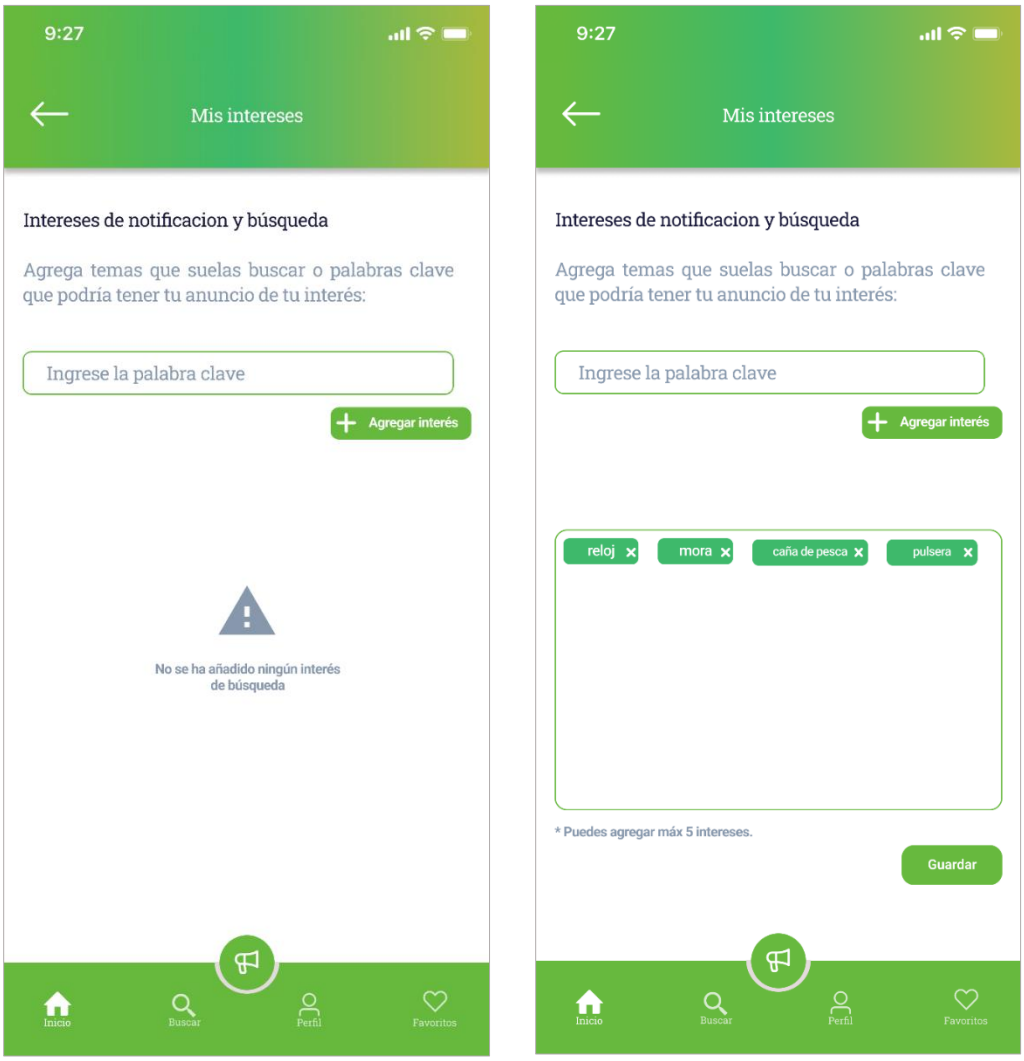
Si desde el menú lateral el usuario opta por ingresar a la opción “Mis Favoritos” ingresará a la presente pantalla, ver figura 26, en la cual podrá visualizar los diferentes anuncios a los cuales el haya marcados como favoritos durante su visualización, en caso de que un usuario desee eliminar un anuncio de su listado de favoritos podrá desmarcar esta opción desde la visualización del mismo.

Figura 27. *Vista historial versión final*



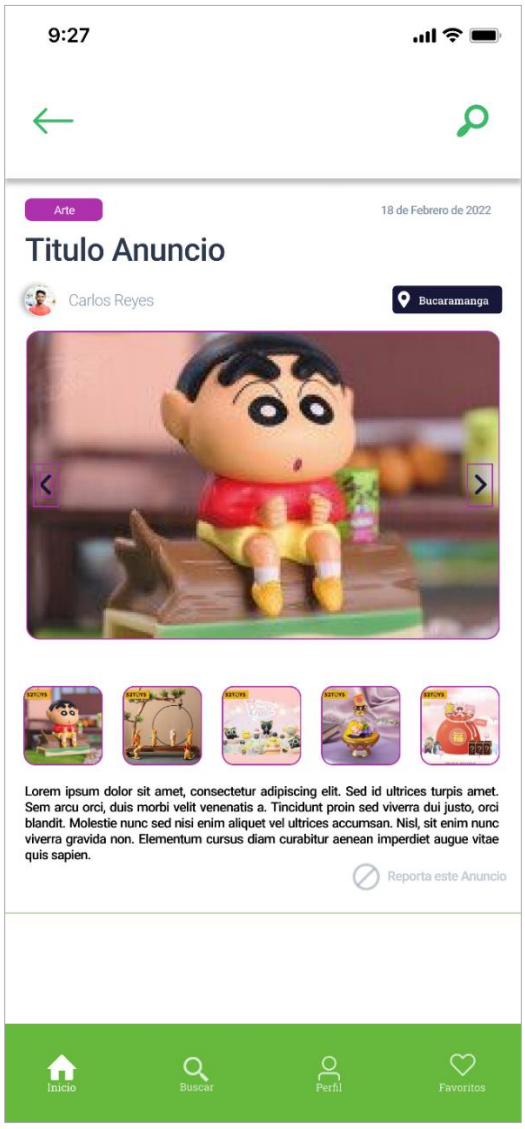
Si desde el menú lateral el usuario opta por ingresar a la opción “Historial” accederá a la presente vista, ver figura 27, donde podrá visualizar por orden cronológico los últimos 100 anuncios visualizados, desde esta vista podrá ver una versión resumida de la información de cada anuncio y desde la cual podrá ver la categoría a la cual pertenece y si el anuncio fue agregado a favoritos.

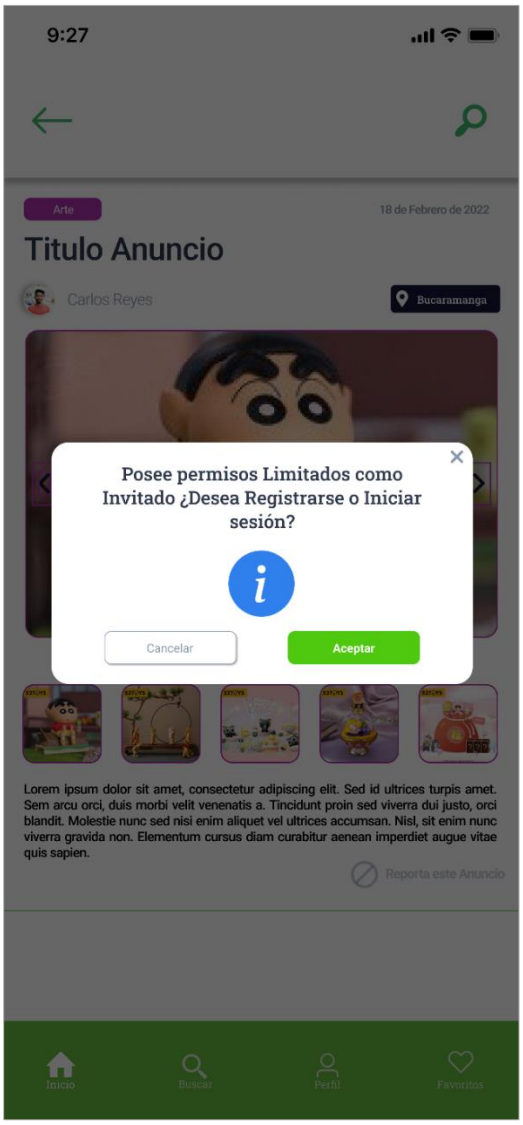
Figura 28. *Vista intereses versión final*



Si desde el menú lateral el usuario opta por ingresar a la opción “Mis Intereses” accederá a la presente vista, ver figura 28, sección en la cual el usuario podrá agregar hasta cinco términos clave diferentes, los cuales el aplicativo tendrá en cuenta a la hora de mostrarle notificaciones al usuario, según los últimos anuncios publicados que contengan los términos anteriormente descritos. Desde esta vista el usuario podrá añadir y remover los términos clave de su interés.

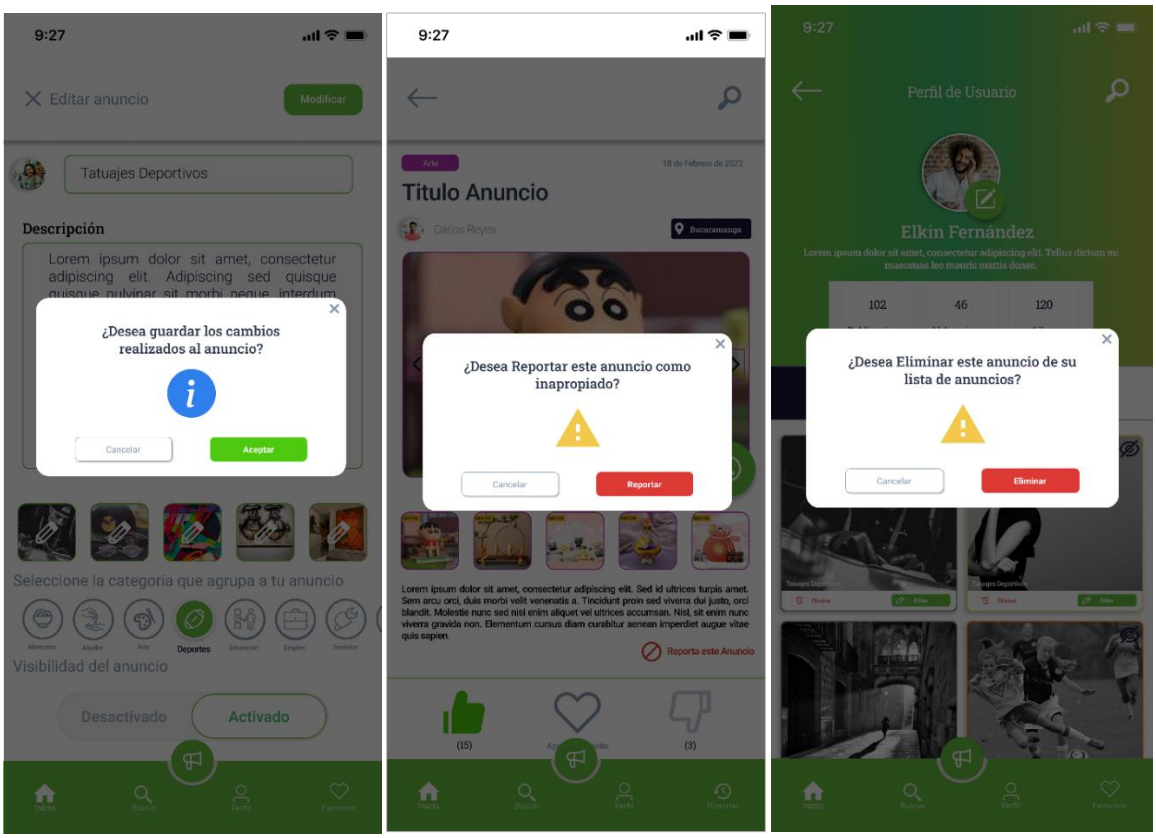
Figura 29. *Vista usuario invitado versión final*





En el caso donde el usuario al momento de iniciar sesión opte por acceder como invitado tendrá sus funciones limitadas, y al acceder a la visualización de un anuncio no tendrá todas las opciones normales disponibles, ver figura 29, en esta visualización como invitado el usuario no podrá valorar la publicación, y dado que no posee un perfil de usuario en esta modalidad tampoco podrá marcar el anuncio como favorito, además de esto tendrá inhabilitada la opción de reportar anuncios, solo podrá visualizar el contenido del anuncio.

Figura 30. *Vista mensajes de alerta versión final*



Con respecto a la primera versión del aplicativo, en esta segunda versión se han implementado mensajes de alerta y de confirmación a la hora de realizar diferentes acciones, como por ejemplo a la hora de realizar la edición de información de un anuncio, cuando se realiza el reporte de un anuncio como inadecuado o el despliegue de un mensaje de confirmación a la hora de eliminar un anuncio.

5.4 Desarrollo

El desarrollo del aplicativo móvil implico el manejo de diversos conceptos usados en la ingeniería de software, además de la etapa de análisis y diseño, en la etapa de construcción del

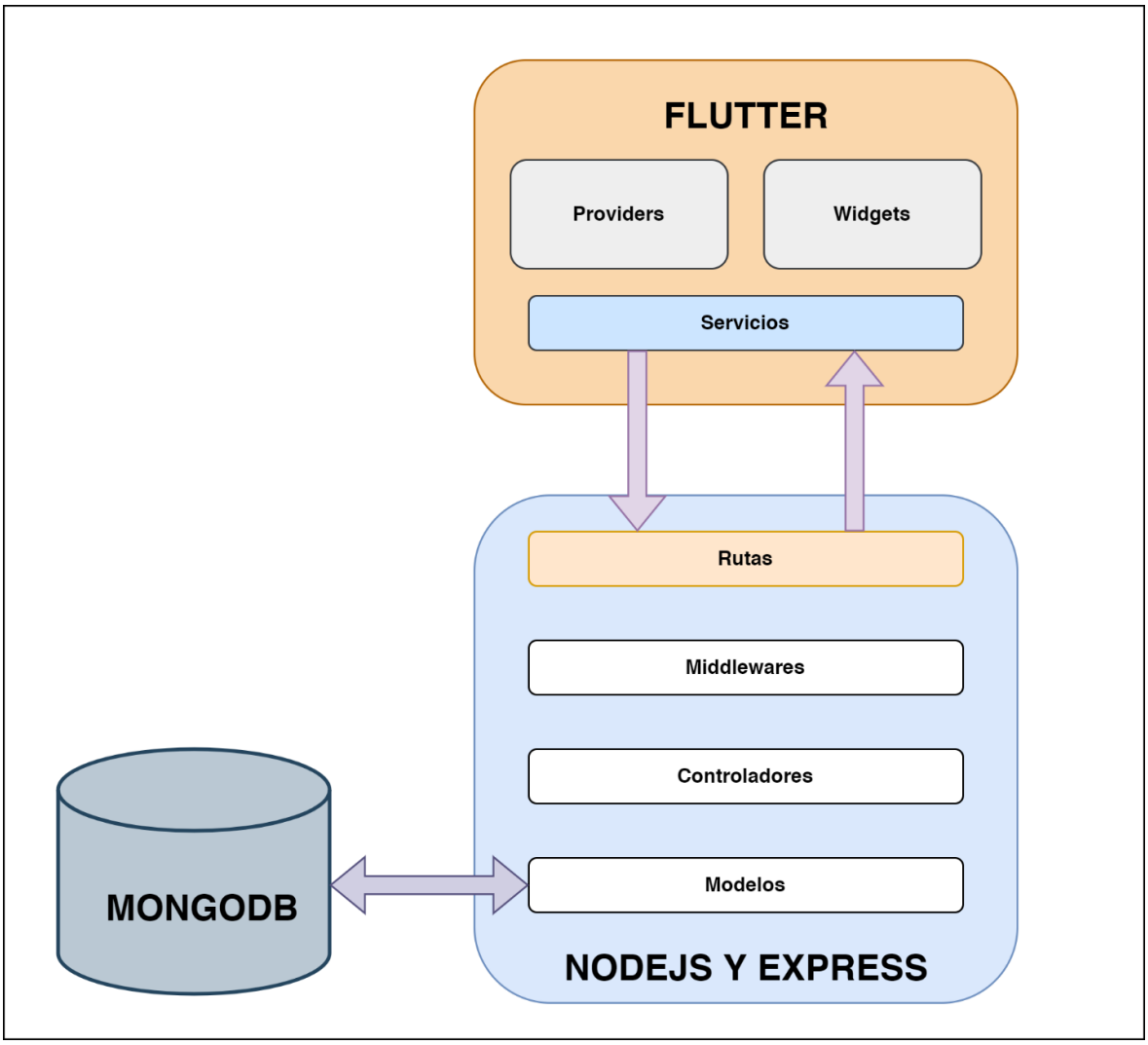
software se deben tener en cuenta elementos que ayudan a que la solución empleada para el cumplimiento de los objetivos, pueda ser llevada a cabo sin ningún contratiempo, se define también las herramientas proporcionadas por los lenguajes de programación, para la construcción de código que sea extensible, reutilizable y menos propenso a errores, pero para llegar a esta fase, es necesario una planeación inicial tal como la Arquitectura que tendrá el software, la comunicación entre las diferentes capas de la aplicación, la división de la presentación y la lógica del negocio, la trazabilidad en el proceso desarrollado y finalmente el proceso de pruebas y puesta a punto, no siendo más se presenta en las siguientes subsecciones el proceso llevado a través de esta planeación inicial en el desarrollo.

5.4.1 Arquitectura

La arquitectura definida en la cual se trabajara el prototipo, está compuesta de diversos componentes tanto en la parte Frontend de la aplicación que será desarrollada con el Framework Flutter de Google, en el cual su lenguaje base es Dart, este Framework de desarrollo está clasificado para el desarrollo de aplicaciones hibridas, ya que responde con una misma base de código a los requerimientos de las plataformas Android y IOS, siendo recientemente agregado el soporte para desarrollo para plataforma web, Escritorio Windows y Mac os, pero para este caso se enfoca en el desarrollo móvil para la plataforma Android, debido a que se cuentan con recursos limitados para el compilado de la aplicación en IOS y además el enfoque inicial de la aplicación es en la plataforma más usada en los dispositivos móviles en los usuarios de la comunidad universitaria, en el componente visual Flutter nos provee de Widgets que son los elementos principales de la aplicación, también para el manejo del estado de estos widgets y sus cambios en base a la información que recibe, se maneja el gestor de estado provider, que permite en base a la información recibida por los servicios notificar los cambios en la data al os widgets para que

procedan a su reconstrucción con la data actualizada. Para la parte del componente Backend el controlador y el modelo están definidos en el entorno de ejecución NodeJs, un entorno creado para la ejecución de código Javascript para el lado del servidor, donde se usa el framework Express que permite la construcción de aplicaciones back de manera rápida y sencilla, donde se puede estructurar la aplicación con cuatro capas las cuales son: rutas, controladores, middlewares y modelos. Las rutas son el intermediario entre el cliente y el servidor a través de las peticiones mediante el protocolo HTTP, los middlewares son las funciones empleadas para el manejo y manipulación de la data recibida, pueden ser usadas para la validación y verificación de la data recibida por parte del cliente, los controladores permiten la interacción con la BD para la agregación, eliminación, búsqueda y actualización de la información que en ella se contiene. Para el almacenamiento de la data se usa una base de datos NoSQL como lo es MongoDB, se usan modelos para hacer referencia a las entidades que se construyen o que componen la base de datos.

Figura 31. *Arquitectura del aplicativo*



Nota: El gráfico muestra la arquitectura de la aplicación. Adaptado de PROTOTIPO DE PLATAFORMA DE ANUNCIOS PARA LA COMUNIDAD UNIVERSITARIA UIS por J. Triana y J.Parra, 2022. Todos los derechos reservados. Reproducido con permiso de los autores.

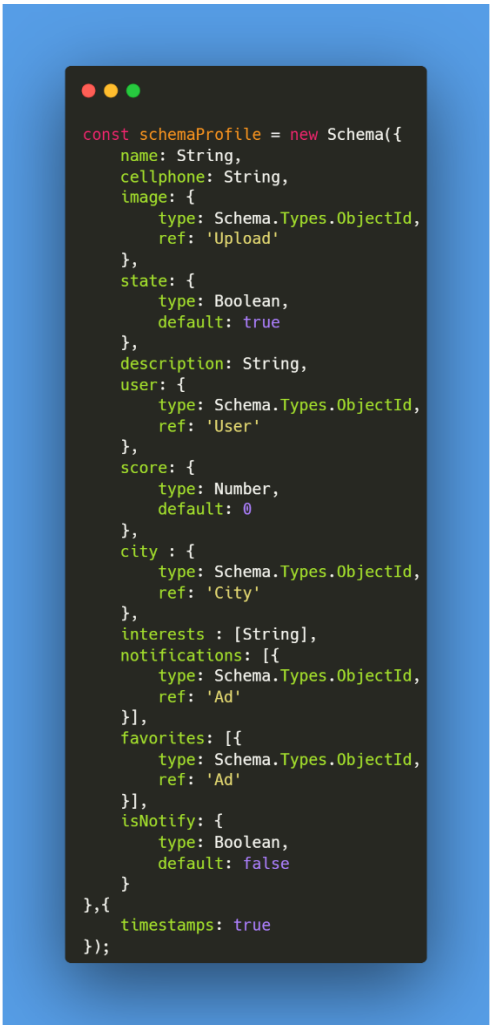
5.4.2 Backend

El Backend contiene la gran parte de lógica de la aplicación, este componente se ubica en el servidor, creado en Express y con el uso de diversas librerías para el manejo de la data y la información, se compone de los diversos elementos mostrados a continuación.

5.4.2.1 *Modelo*

Los modelos en NodeJs son usados para representar una entidad en la base de datos, en específico representan un documento o un registro único de la base de datos. Para destacar el caso de la colección Profile o Perfil, dentro de este se guardarán los documentos de tipo Profile, en el código se manejó el modelo Profile, con todos los campos proporcionados en el código, al crear un Profile, se crea una instancia de ese modelo, en resumen, los objetos creados permiten de una forma más sencilla, administrar la información que se recibirá del Front, manejándola en el back y finalmente almacenándola en la base de datos.

Figura 32. *Modelo de base de datos Profile codificado en el Back*



```
const schemaProfile = new Schema({
    name: String,
    cellphone: String,
    image: {
        type: Schema.Types.ObjectId,
        ref: 'Upload'
    },
    state: {
        type: Boolean,
        default: true
    },
    description: String,
    user: {
        type: Schema.Types.ObjectId,
        ref: 'User'
    },
    score: {
        type: Number,
        default: 0
    },
    city : {
        type: Schema.Types.ObjectId,
        ref: 'City'
    },
    interests : [String],
    notifications: [{
        type: Schema.Types.ObjectId,
        ref: 'Ad'
    }],
    favorites: [{
        type: Schema.Types.ObjectId,
        ref: 'Ad'
    }],
    isNotify: {
        type: Boolean,
        default: false
    }
},{
    timestamps: true
});
```

En la Figura 32 se representa el modelo usando la clase Schema proporcionada por la librería de Moongose, esta librería me permite conectarme a la base de datos de MongoDb, en la figura se muestra los nombres de las diferentes propiedades como name, cellphone, interests, entre otras mostradas. Estas propiedades representan las columnas de mi objeto, dentro de estas definiciones se describe el tipo de dato que contendrá, siendo algo posible el uso de referencias a otros objetos por medio del *ObjectId,* que referencia al id del registro del tipo en el caso de *notifications,* la referencia se realiza a los registros de tipo Ad, esto nos permite traer la información

asociada a este objeto. Finalmente se encapsula en la función model, que nos permite a la estructura creada proporcionarle un nombre y asociarlo al tipo *Schema* definido.

5.4.2.2 *Rutas*

Las rutas hacen referencia a los métodos HTTP como lo son GET, POST, PUT/PATCH, y DELETE, Express permite la definición de rutas directamente en el objeto app, es una forma sencilla del manejo de rutas, pero implica que en caso de un crecimiento de la app puede llegar a ser un problema la organización de rutas, para esto en express se puede usar el manejo del objeto Router como se expresa en la Figura 33, este objeto permite regular la estructura de organización de las rutas, la instancia funciona como una instancia singleton, donde le podemos ir añadiendo las rutas a la instancia, en la Figura 33 se expresa la ruta del reporte asociada al objeto Router.

Figura 33. *Código JavaScript para ruta Reporte en el Back*

```
const router = Router();

router.post('/',
    validateJWT,
    isProfileExists,
    check('ad','La publicación debe ser obligatoria').not().isEmpty(),
    check('ad').isMongoId(),
    validateFields,
    createReport
);
```

Para la ruta establecida, se define el método POST y la ruta que se asociara a este método como primer parámetro, de tal forma que cuando se apunte a este endpoint desde el front se pueda recibir la respuesta de parte del back, los parámetros siguientes son las funciones o middlewares

por las cuales pasará la información o el request enviado desde el front, que procederá a validar y a pasar la información al último parámetro colocado, que es el controlador que manejará la información en la base de datos.

5.4.2.3 Middlewares

Los middlewares son considerados como funciones, que permiten realizar algo antes de que una petición sea procesada, como por ejemplo la comprobación de que, si un usuario ha realizado la autenticación de la cuenta, o como en la Figura 34 que se comprueba si un perfil existe en BD antes de proceder a regresar información al usuario, estos tipos de funciones se usan para manejar los objetos request y response, objetos de solicitud y respuesta del servidor que usa express para la administración de los datos. Los middlewares son llamados también funciones intermedias, ya que permiten evaluar la request y en base a la lógica del negocio, pueden finalizar la operación o invocar la siguiente función que está en la pila de middlewares.

La Figura 34 muestra la estructura del código de un middleware, su funcionamiento base indica, que hay una consulta a la BD para revisar si un perfil existe y en caso positivo, retornara esta respuesta al usuario de que no se pudo realizar la petición ingresada desde la ruta y detenida en este middleware debido a que el usuario “Ya posee un perfil de usuario”, los parámetros recibidos por el middleware son en orden el request, response y next, los dos primeros son objetos usados por express ya detallados en el inicio y el tercer parámetro es la función *next()* que se utiliza para continuar a la siguiente función o middleware que espera una respuesta de lo actual para continuar.

Figura 34. *Código de función validadora o middleware*

```
// * Middleware para validar si el perfil existe en la base de datos
// * En esta middleware se valida por usuario
const isProfileExists = async ( req = request, res = response, next ) => {
    const profileExist = await Profile.findOne({ user: req.user.id });
    if( profileExist ) {
        return res.status(400).json({
            msg: 'Ya posee un perfil de usuario'
        });
    }
    next();
}
```

5.4.2.4 Controlador

Un controlador es una función que podría ser considerado un middleware, porque en parte lo es solamente, su cambio es en base a que no posee el parámetro next, ya que es la última función disponible en la pila del stack de middlewares, este controlador recibe la información “exitosa” luego de las respectivas primeras validaciones “intermedias” realizadas, su función será afectar la base de datos por medio de las operaciones CRUD a los registros presentes en esta, a través del modelo definido en el backend. El controlador permite devolver los datos al usuario luego de realizar las operaciones en los modelos, pero los datos devueltos van acompañados de las respuestas de servidor, siendo 200, para este caso una respuesta exitosa y 500, 400, 404 para indicar respuestas no exitosas, estos códigos de respuesta del servidor son unos de los más comúnmente usados para expresar la información a un usuario que realizó una petición y desea saber cuál es el estado de esta. Para el caso de la Figura 35 el controlador recibe el request y el response, extrae información del cuerpo del request y en base a esta información realiza la búsqueda, para este caso

si ya existe un reporte asociado a este perfil, en caso positivo devolverá su respectivo código de respuesta y el JSON con el mensaje de que el reporte ya se realizó, en caso contrario se continuará y se establecerá una nueva inserción de un reporte asociado a un perfil y un anuncio y se devuelve la información exitosa al usuario mediante el código de respuesta HTPP 200 y el JSON con la información de que la petición fue realizada exitosamente.

Figura 35. *Código de un controlador en el Back*

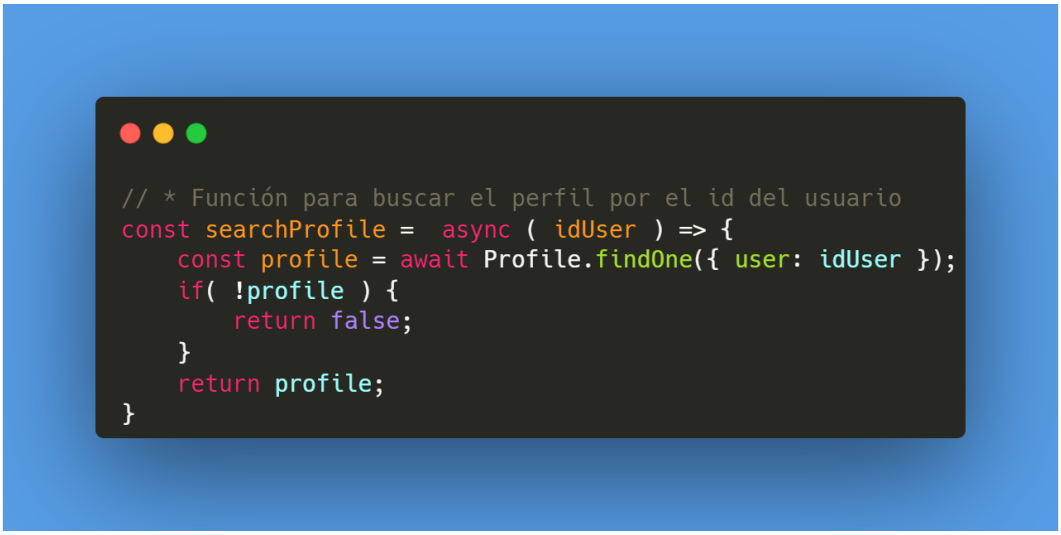
```
const createReport = async ( req = request, res = response ) => {
    try {
        const { ad } = req.body;
        const { user } = req;
        const profile = searchProfile( user._id );
        const reportExists = await Report.findOne({ reporter: profile._id, ad });
        if( reportExists ) {
            return res.status(400).json({ msg : 'Ya ha reportado este anuncio!' });
        }
        const reportNew = new Report({
            reporter: profile._id,
            ad
        });
        const reportSaved = await reportNew.save();
        if( !reportSaved ) {
            return res.status(400).json({ msg : 'No se pudo guardar el reporte' });
        }
        return res.status(200).json({
            msg: 'Se creo el reporte con exito'
        });
    } catch (error) {
        return res.status(404).json({ msg : `No se pueden visualizar los anuncios ${error}` });
    }
}
```

5.4.2.5 *Helper*

Los helpers o utils son funciones complementarias que permiten la organización y manejo de información en los controladores, como se muestra en la Figura 36 donde este helper de la Figura 17 es invocado y utilizado para consultar la información respectiva a la existencia de un

perfil, necesaria para el proceso que realiza el controlador en el proceso de crear un reporte a un anuncio, esta función de búsqueda como se le conoce a este helper en específico no solamente se usará en este controlador, sino que puede ser necesario su uso en otro lugar, así que se necesita tener centralizado para poder usarlo en cualquier instancia del servidor, esta funcionalidad se realiza y permite que no haya un acoplamiento y código repetitivo en el servidor.

Figura 36. *Código función helper*



```
// * Función para buscar el perfil por el id del usuario
const searchProfile =  async ( idUser ) => {
    const profile = await Profile.findOne({ user: idUser });
    if( !profile ) {
        return false;
    }
    return profile;
}
```

5.4.2.6 Listado peticiones HTTP (Endpoints)

Se crearon las rutas a las que el Front por medio de peticiones HTTP consulta y trae información a la vista, por lo tanto, estos servicios utilizados para el correcto funcionamiento del aplicativo, junto a su respectivo Método y descripción se presenta a continuación:

Tabla 30. *Rutas de autenticación con otros medios*

URI	Método	Descripción
/api/auth/google	POST	Autentica al usuario con Google.

/api/auth/facebook/token	POST	Autentica al usuario con Facebook.

Tabla 31. *Ruta para categoría*

URI	Método	Descripción
/api/category/:id	GET	Obtiene una categoría en base a un id.

Tabla 32. *Rutas para favorito*

URI	Método	Descripción
/api/profile/favorite-ad	POST	Guarda un anuncio como favorito.
/api/profile/favorite-ad/:id	GET	Obtiene la lista de favoritos asociados a un usuario.
/api/profile/favorite-ad/:id	DELETE	Elimina un registro asociado al id de un usuario en los favoritos.

Tabla 33. *Rutas para el historial de Anuncios*

URI	Método	Descripción
/api/historial	POST	Añade un anuncio a la lista del historial de un usuario.
/api/historial/:id	GET	Obtiene el historial de anuncios visualizados por un usuario.
/api/historial/:id	DELETE	Elimina todos los registros del historial asociados a un usuario

Tabla 34. *Rutas para los intereses*

URI	Método	Descripción
/api/interest	POST	Guarda una lista de intereses asociados a un usuario
/api/interest/:id	GET	Obtiene la lista de intereses de un usuario en especifico
/api/interest/:id	DELETE	Elimina el total de registros de intereses asociados a un usuario

Tabla 35. *Rutas para las notificaciones*

URI	Método	Descripción
/api/notification	POST	Permite activar o desactivar7 notificaciones
/api/notification/:id	GET	Obtiene el listado de anuncios en base a los intereses del usuario

Tabla 36. *Rutas para reportes*

URI	Método	Descripción
/api/report	POST	Reporta un anuncio especifico
/api/report/:params	DELETE	Elimina un reporte realizado a un anuncio especifico.

5.4.3 Frontend

El Frontend de la aplicación móvil comprende la parte visual que se presenta al usuario y la parte con la que este interactúa, de tal forma que se busca que para el usuario sea lo más intuitiva

y fácil de usar y que no conlleve mucho conocimiento del usuario aprender las acciones básicas dentro de esta, de tal manera que se desarrolló la aplicación en Flutter, que es un framework de Google diseñado para construir aplicaciones móviles de una forma fácil y sencilla enfocada en dar excelentes experiencias de usuario desde una misma base de código, basada en el lenguaje Dart, la estructura usada en el Frontend se divide en los siguientes componentes.

5.4.3.1 *Modelo*

Un modelo es una clase plana o PODOs (Plain Old Dart Object), que se usa para representar los objetos que mostrarán la información en la aplicación, fueron construidos en base a los modelos representados en el backend para la persistencia de los datos. En si se puede decir que "Los modelos empleados en el Frontend apuntan a los datos puros que se presentan en la pantalla del aplicativo, todos estos datos son ingresados en las clases o plantillas generadas para este fin" *(Triana & Parra, 2022, p.123).*

Figura 37. *Modelo para la data en el Frontend*

```dart
Profile profileFromMap(String str) => Profile.fromMap(json.decode(str));

String profileToMap(Profile data) => json.encode(data.toMap());
class Profile {
    Profile({
        required this.interests,
        required this.notifications,
        required this.favorites,
        required this.isNotify,
        required this.name,
        required this.cellphone,
        required this.email,
        required this.state,
        required this.score,
        required this.city,
        required this.image,
        required this.createdAt,
        required this.updatedAt,
        required this.description,
        required this.uid,
    });
    // El dynamic es para que acepte cualquier tipo de dato pero es un tipo Ad
    List<String> interests;
    List<dynamic> notifications;
    List<dynamic> favorites;
    bool isNotify;
    String name;
    String cellphone;
    String email;
    bool state;
    int score;
    String city;
    Upload image;
    String createdAt;
    String updatedAt;
    String description;
    String uid;
```

```dart
factory Profile.fromMap(Map<String, dynamic> json) => Profile(
        interests: List<String>.from(json["interests"].map((x) => x)),
        notifications: List<dynamic>.from(json["notifications"].map((x) => x)),
        favorites: List<dynamic>.from(json["favorites"].map((x) => x)),
        isNotify: json["isNotify"],
        name: json["name"] ?? '',
        cellphone: json["cellphone"] ?? '',
        email: json["email"] ?? '',
        state: json["state"] ?? true,
        score: json["score"] ?? 0,
        city: json["city"] ?? '',
        image: Upload.fromMap( json["image"] ?? {}),
        createdAt: json["createdAt"] ?? '',
        updatedAt: json["updatedAt"] ?? '',
        description: json["description"] ?? '',
        uid: json["uid"] ?? '',
    );

    Map<String, dynamic> toMap() => {
        "interests": List<String>.from(interests.map((x) => x)),
        "notifications": List<dynamic>.from(notifications.map((x) => x)),
        "favorites": List<dynamic>.from(favorites.map((x) => x)),
        "isNotify": isNotify,
        "name": name,
        "cellphone": cellphone,
        "email": email,
        "state": state.toString(),
        "score": score.toString(),
        "city": city,
        "image":  json.encode( image.toMap() ),
        "createdAt": createdAt,
        "updatedAt": updatedAt,
        "description": description,
        "uid": uid,
    };
}
```

En la Figura 37 se presenta el modelo que se usa para representar un perfil en la aplicación, este modelo fue actualizado en base a los requerimientos funcionales planteados, este modelo recibirá la información del backend luego de iniciar sesión, es la base de nuestra vista y con esto mostramos la información en ella. “Los PODOs, actúan como una forma sencilla de tomar un JSON, decodificando y mostrando los datos deserializados en la aplicación, lugar donde transformamos las claves y valores del JSON en propiedades de una clase” *(Triana & Parra, 2022, p.123).* Dentro de este modelo se posee el método Factory, el cual es un patrón de diseño creacional que permite crear objetos por medio de una interfaz en una superclase, para este caso se toma la información decodificada del JSON recibido por parte del backend y se transforma en PODOs del tipo presentado en la figura 37, las propiedades del JSON son mapeadas a mi clase modelo en DART y de tal forma podemos empezar a mostrar la información recibida al usuario, en la aplicación por medio de las pantallas y widgets que componen el ambiente visual del aplicativo.

5.4.3.2 *Servicio*

En el aplicativo los servicios se usan para contener diferentes funcionalidades, se expresan en clases, son la contraparte de los helpers en el backend, un servicio permite separar la lógica y sus detalles de la implementación para la aplicación, donde solo es llamado y ejecutada su tarea.

> Algunos ejemplos de servicios que son usados en la aplicación son la obtención de datos de un API, almacenar datos en el LocalStorage del dispositivo, gestionar la conectividad de la aplicación, manejar los temas de la aplicación, realizar cálculos complejos o en gran cantidad, trabajar con paquetes de terceros, entre otros *(Triana & Parra, 2022, p.124).*

Para el caso del servicio expresado en la Figura 38 permite realizar el proceso de conexión con el servicio de Google para la autenticación, en este servicio específico se da la solución al problema de ingreso y registro de otros medios, nos permite autenticarnos en la aplicación sin necesidad de usar una contraseña y por medio del estándar Auth0. En este servicio también se realiza el proceso de envió de las credenciales al backend para la respectiva comprobación de que el usuario autenticado por medio de Google, está registrado o necesario que se registre o en caso contrario inicie sesión en el aplicativo.

Figura 38. *Código de servicio en Frontend*

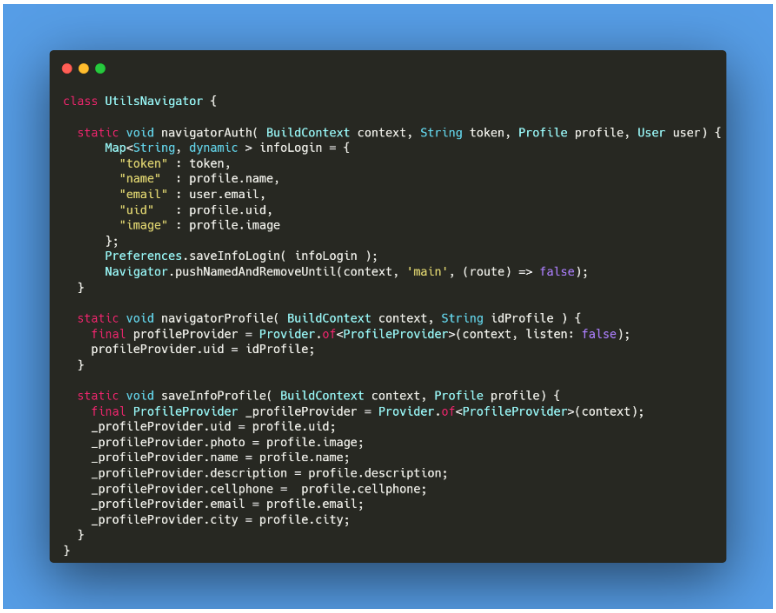
```dart
class GoogleSigninService {

  static GoogleSignIn _googleSignIn = GoogleSignIn(
    scopes: [
      'email'
    ],
  );

  static Future<GoogleSignInAccount?> signInWithGoogle() async {
    try {
      final GoogleSignInAccount? googleUser = await _googleSignIn.signIn();
      if (googleUser != null) {

        final GoogleSignInAuthentication googleAuth = await googleUser.authentication;
        if (googleAuth.accessToken != null && googleAuth.idToken != null) {
          return googleUser;
        } else {
          throw Exception('No se pudo obtener la autenticacion de Google');
        }
      } else {
        throw Exception('No se pudo obtener la autenticacion de Google');
      }
    } catch (e) {
      log('Error en el servicio de Google Signin: $e');
      return null;
    }
  }

  // Metodo para cerrar sesion con Google
  static Future<void> signOutGoogle() async {
    await _googleSignIn.signOut();
  }
  // Metodo para revisar si esta logeado con Google
  static Future<bool> isSignedInGoogle() async {
    return await _googleSignIn.isSignedIn();
  }

}
```

5.4.3.3 Utilidades

En el desarrollo de software, es ideal mantener buenas prácticas en programación, se desea que en el proceso de codificación el código sea limpio, reutilizable y escalable y pueda ser fácilmente cambiado, esto en busca de reducir el tiempo en las tareas de mantenimiento que se realiza al software con el propósito de aumentar su vida útil, muchas veces se tiende a encontrar mucho código redundante que puede ser reducido a una sola función genérica que realice esto en los diversos lugares que se repite, para esto surge el concepto de utilidades. Las utilidades son porciones de código que se pueden usar de manera repetitiva, lo ideal en el desarrollo es tener todo lo menos acoplado posible, con las utilidades podemos optimizar el código y mejorar la mantenibilidad de este, en el Frontend se implementaron diversas utilidades, tales como

navegación autorizada, contacto con el vendedor, utilidades de búsqueda entre otras, en la siguiente Figura se detalla el uso de la clase *UtilsNavigator* para realizar una navegación autorizada entre las pantallas y el almacenamiento del token actualizado, esto se repite en diversas pantallas y por lo tanto para no tener código redundante se centraliza en una clase para controlar las navegaciones.

Figura 39. *Utilidad usada en el Frontend*



```dart
class UtilsNavigator {

  static void navigatorAuth( BuildContext context, String token, Profile profile, User user) {
      Map<String, dynamic > infoLogin = {
        "token" : token,
        "name"  : profile.name,
        "email" : user.email,
        "uid"   : profile.uid,
        "image" : profile.image
      };
      Preferences.saveInfoLogin( infoLogin );
      Navigator.pushNamedAndRemoveUntil(context, 'main', (route) => false);
  }

  static void navigatorProfile( BuildContext context, String idProfile ) {
    final profileProvider = Provider.of<ProfileProvider>(context, listen: false);
    profileProvider.uid = idProfile;
  }

  static void saveInfoProfile( BuildContext context, Profile profile) {
    final ProfileProvider _profileProvider = Provider.of<ProfileProvider>(context);
    _profileProvider.uid = profile.uid;
    _profileProvider.photo = profile.image;
    _profileProvider.name = profile.name;
    _profileProvider.description = profile.description;
    _profileProvider.cellphone =  profile.cellphone;
    _profileProvider.email = profile.email;
    _profileProvider.city = profile.city;
  }
}
```

5.4.3.4 *Provider*

La gestión de estados en una aplicación en Flutter, es muy importante debido a que en Flutter todo se componen por elementos conocidos como widgets. Los widgets pueden ser de dos tipos Statelesswidget o Statefullwidget, los StateFullWidgets son widgets que cambian de estado si el usuario interactúa con él, los Statelesswidget son conocidos como widgets sin estado, no son dinámicos es decir no pueden cambiar su apariencia con base en los eventos que se realicen por las interacciones de los usuarios, un estado de un widget se almacena en un State, permitiendo una separación entre el widget y su apariencia con el estado de este, en los StateFullWidget el estado

se afecta con una función *setState(),* para el cambio de estados con StateFullWidget, puede ser basta con este *setState().*

Para la gestión de estados es ideal el uso del *setState()* en una aplicación con un alcance y tamaño pequeño, "pero cuando se tiene que trabajar con una aplicación que contenga múltiples pantallas, gestione diferentes servicios o cargue información que se necesite mostrar entre diferentes widgets, el proceso se vuelve muy complejo de gestionar, llegando a ser imposible su control y la conexión de información entre los diferentes widgets hijos y padres, por ello surge la necesidad de implementar un patrón que cumpla una adecuada gestión del estado de la aplicación*" (Triana & Parra, 2022, p.126).* Para el caso de la figura 40 tenemos la implementación de un provider que afectara la página de intereses, ya que este gestor de estados al recibir un widget puede afectar los StatelessWidgets, debido a que notifica según el lugar en el que se use a los widgets para que se redibujen y actualicen la información de cada uno, para que esto funcione se debe tener un widget Provider como padre de los widgets a comunicar.

Figura 40. *Código de provider para los intereses*

```dart
class InterestPageProvider with  ChangeNotifier {

  String _interes = '';
  final _formKey = LabeledGlobalKey<FormState>('interest_form');
  GlobalKey<FormState> get formKey => _formKey;
  List<String> _interests = [];

  String get interes => _interes;
  set interes(String value) {
    _interes = value;
    notifyListeners();
  }
  List<String> get interests => _interests;
  set interests( List<String> value ) {
    _interests = value;
    notifyListeners();
  }
  // Metodo para agregar un interes
  void addInterest( String value ) {
    _interests.add(value);
    notifyListeners();
  }
  // Metodo para eliminar un interes
  void removeInterest( String value ) {
    _interests.remove(value);
    notifyListeners();
  }
  // Metodo para validar el interes
  String? validateInterest( String? value ) {
    if (value!.isEmpty) {
      return 'Campo Requerido';
    }
    return '';
  }
  // Metodo para limpiar la lista de intereses
  void cleanInterests() {
    _interests.clear();
    notifyListeners();
  }
}
```

Hay que tener en cuenta que el gestor provider puede ser muy eficiente, pero a la vez se puede volver muy complejo, debido a la cantidad de providers a administrar, por eso la mayoría de las veces se ubican en un multiprovider de manera global en la aplicación, para tenerlos de forma más centralizada como se muestra en la figura 41, para el caso de que una propiedad de los InterestPageProvider este enlazada en un widget al hacer una actualización de esta "solamente se reconstruirá el widget que haya sido afectado, lo cual permite reducir la carga de trabajo que se realizada en la aplicación, posibilitando una ejecución más eficiente y sin retrasos." *(Triana & Parra, 2022, p.128).*

Figura 41. *Codificación multiprovider*

```
class AppState extends StatelessWidget {
  const AppState({Key? key}) : super(key: key);


  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider(create: (_) => BottomNavigationBarProvider()),
        ChangeNotifierProvider(create: (_) => ProfileProvider()),
        ChangeNotifierProvider(create: (_) => RegisterFormProvider())
      ],
      child: const App(),
    );
  }
}
```

Nota: El gráfico muestra la construcción de un multiprovider en código. Adaptado de PROTOTIPO DE PLATAFORMA DE ANUNCIOS PARA LA COMUNIDAD UNIVERSITARIA UIS (p.128) por J. Triana y J.Parra, 2022. Todos los derechos reservados. Reproducido con permiso de los autores.

5.4.3.5 *Vistas*

Para construir una vista en Flutter es necesario entender cuál es la forma básica de construcción de una vista. En Flutter todo se compone por widgets, una pantalla o screen forma parte de un árbol de widgets, hay diferentes tipos de estos, lo que permite que se puedan mostrar en totalidad en el dispositivo. “Los widgets describen como deberá mostrarse la vista según su configuración y estado actuales, (…), los cuales pueden ser usados para estructurar parte de la

aplicación, tales como Row widget, Column Widget, Container Widget, Stack Widget, Text Widget, entre otros" *(Triana & Parra, 2022, p.128).*

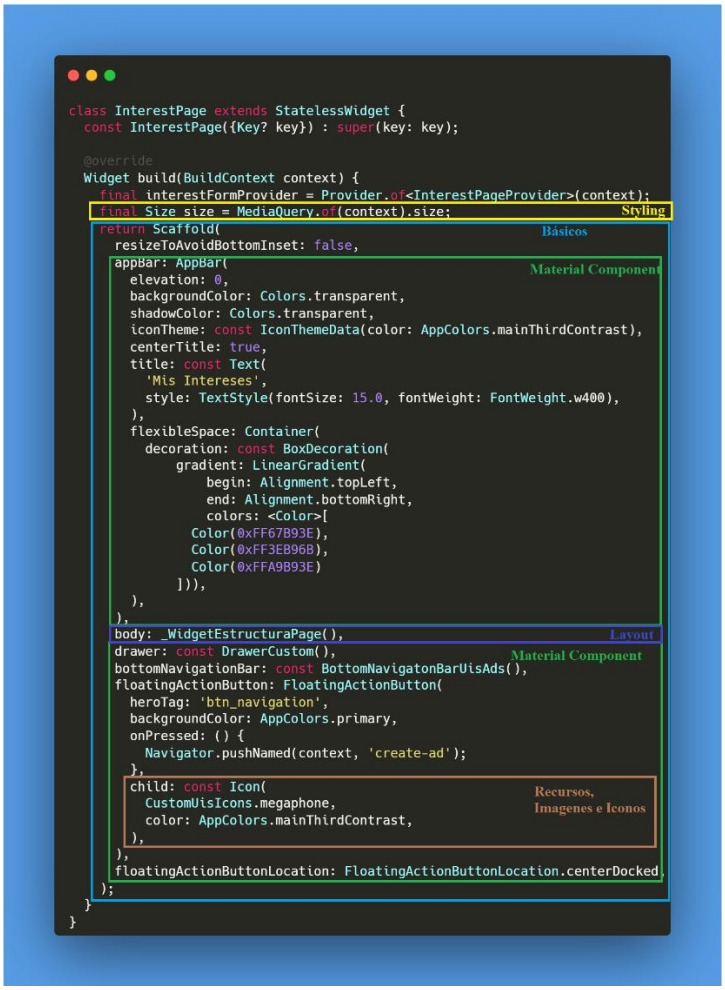
En la documentación oficial de Flutter, se exponen los diferentes widgets categorizados en 14 categorías, los presentes en el desarrollo del primer prototipo del aplicativo y el actual fueron:

- Accesibilidad
- Recursos, imágenes e iconos
- Async
- Básicos
- Inputs
- Modelos de interacción
- Layout o estructura de diseño
- Scrolling
- Styling
- Text
- Painting and efects
- Componentes de material

Es de tener en cuenta que en la construcción de una vista para el aplicativo en Flutter, interactuamos con los diferentes categorías de widgets presentes, la estructura montada es una combinación de widgets tales como widgets de layout que me permiten, ubicar widgets en la pantalla según se necesiten, dentro de estos widgets layouts podemos mostrar widgets de tipo Recursos, que nos permiten mostrar imágenes e iconos en el aplicativo, también widgets de texto

que permiten la visualización de un texto al usuario, en fin, la estructura puede ser muy variable, pero los más usados para estructurar en general son los Material Components que son widgets que permiten estilizar en base al diseño proporcionado por Google (Material Design) y los Layout que permiten como se mencionaba anteriormente armar las bases para la ubicación de más widgets en el árbol de widgets.

Figura 42. *Diseño de una vista general en código.*



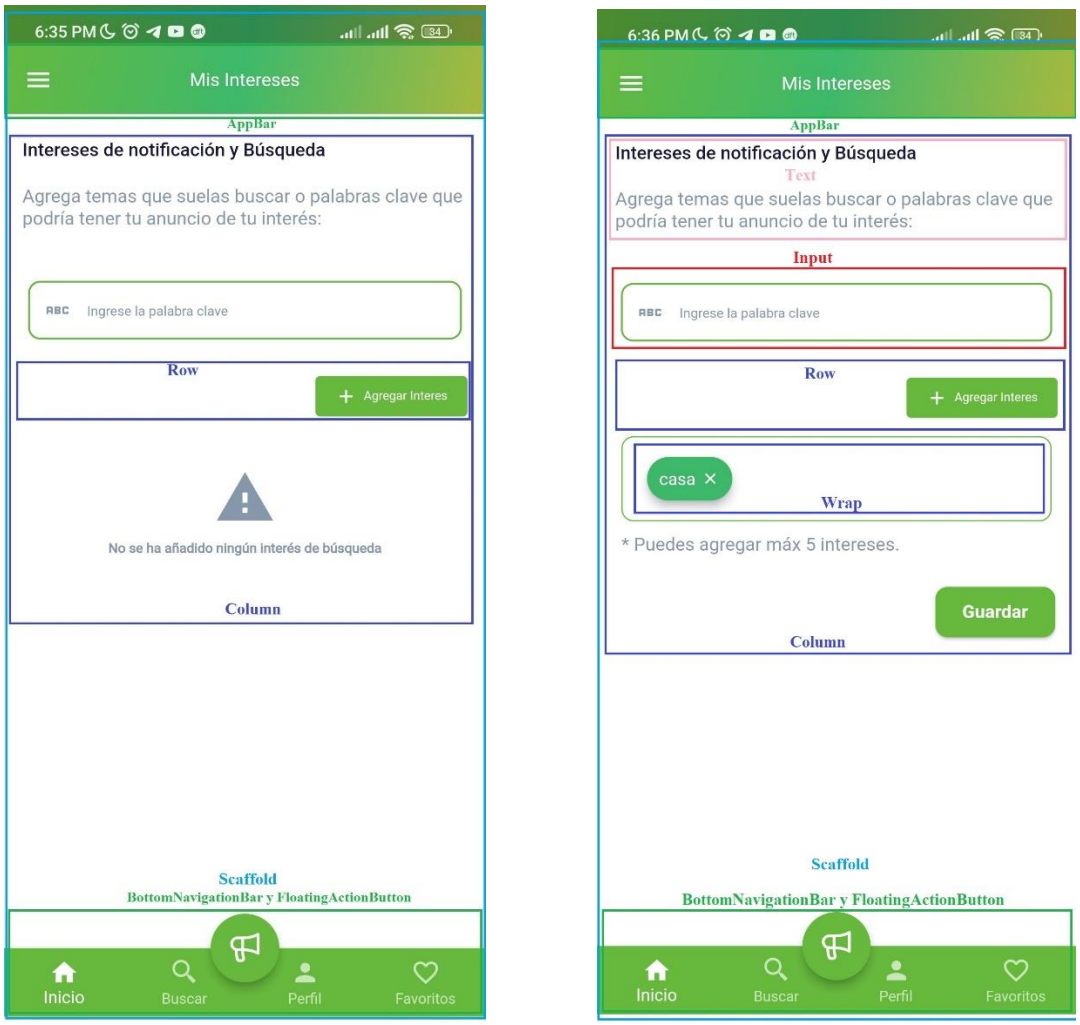
En la Figura 42 se muestra la construcción de la pantalla para los Intereses mostrados para el perfil de un usuario en el aplicativo de UISAds, la estructura montada inicialmente se realizó

con un widget perteneciente a los componentes de Material y a los Widgets básicos como lo es el Scaffold. Con el Scaffold se construye una estructura de diseño con los elementos básicos en una pantalla, tales como lo es el Appbar, el body, el BottomNavigationBar, un Drawer, entre otros widgets *(Triana & Parra, 2022).* La estructura básica para este caso creada en la Figura 42 tenemos además de componentes Básicos, componentes de Material y componentes de tipo Recursos, Imágenes e Iconos, además el body internamente constituye un widget Layout, este me permite tener una estructura básica o diferentes widgets internos ya sean layouts entre otros, en la Figura 43, se muestra la estructura del layout usada en el body para anidar más widgets, el primer nivel del layout es un Container, el cual es un Widget que funciona como un contenedor de otro widget, este es un tipo de Single-child layout, ya que solo puede contener un hijo a la vez, dentro de este tenemos un Widget de tipo Scrolling, que permite que los widgets anidados dentro de este puedan tener un tipo de scroll, para el caso del SingleChildScrollView es un tipo de widget puede agregar Scrolling a un solo Widget, en este caso su hijo es un tipo Input que nos permitirá controlar las entradas de texto por parte del usuario, al interior de este anidamos un multiple-layout widget como lo es el Column que contendrá múltiples Widgets, esta es la estructura montada para la pantalla de los Intereses.

Figura 43. *Diseño de una vista general en código.*

La parte visual de este código diseñado para la vista de intereses, en base a las condiciones expresadas en el código puede presentarse de dos formas distintas, la primera vista es con base en el caso de que no se posean intereses agregados por el usuario, por lo tanto se mostrará un widget vacío condicionado a la lista de intereses si está vacía como se expresa en la Figura 44 en la parte de código, para la parte resultado de la codificación, esta se expresa en la siguiente Figura a continuación:

Figura 44. *Ejemplo estructura de la vista de Intereses vacía y con Intereses*



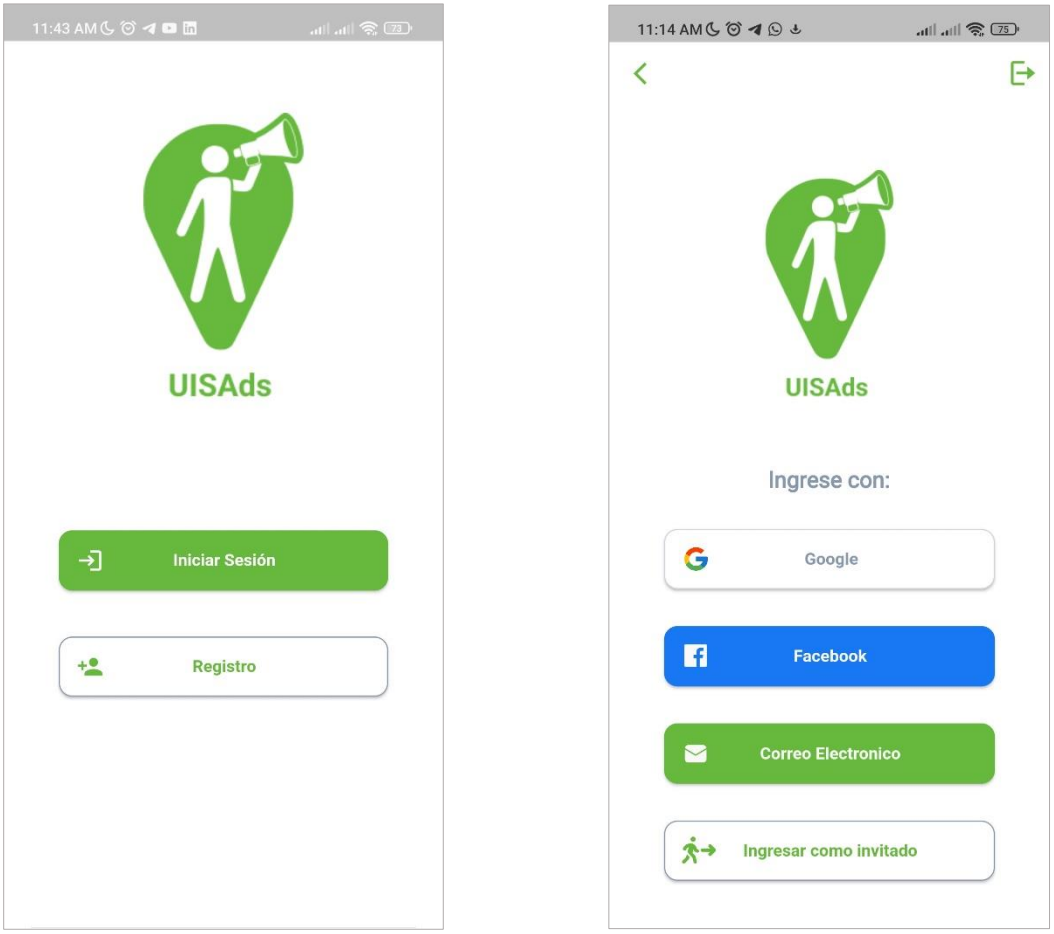
En la Figura 44 se ilustran las estructuras de widgets usadas para la construcción de esta pantalla, solo las generales en base a las 14 categorías definidas por el Framework, para este caso tenemos los widgets Row, Column y Wrap pertenecientes a los widgets tipo Layout, al igual que el Scaffold que entra entre básicos y Layout, también se presenta los Widgets pertenecientes a los tipo Material Components BottomNavigationBar y FloatingActionButton y finalmente se muestra el Widget perteneciente a la categoría Inputs con un delineado rojo. Estos componentes de manera

general son los elementos que forman parte del árbol de widgets en la aplicación y conforman la estructura de la página de InteresesPage, mostrada en código en las Figuras 43 y 44.

5.4.4 Despliegue prototipo

En esta sección se ubican las vistas principales añadidas al aplicativo en esta versión desarrollada, para el cumplimiento de las funcionalidades planteadas en el proyecto de grado.

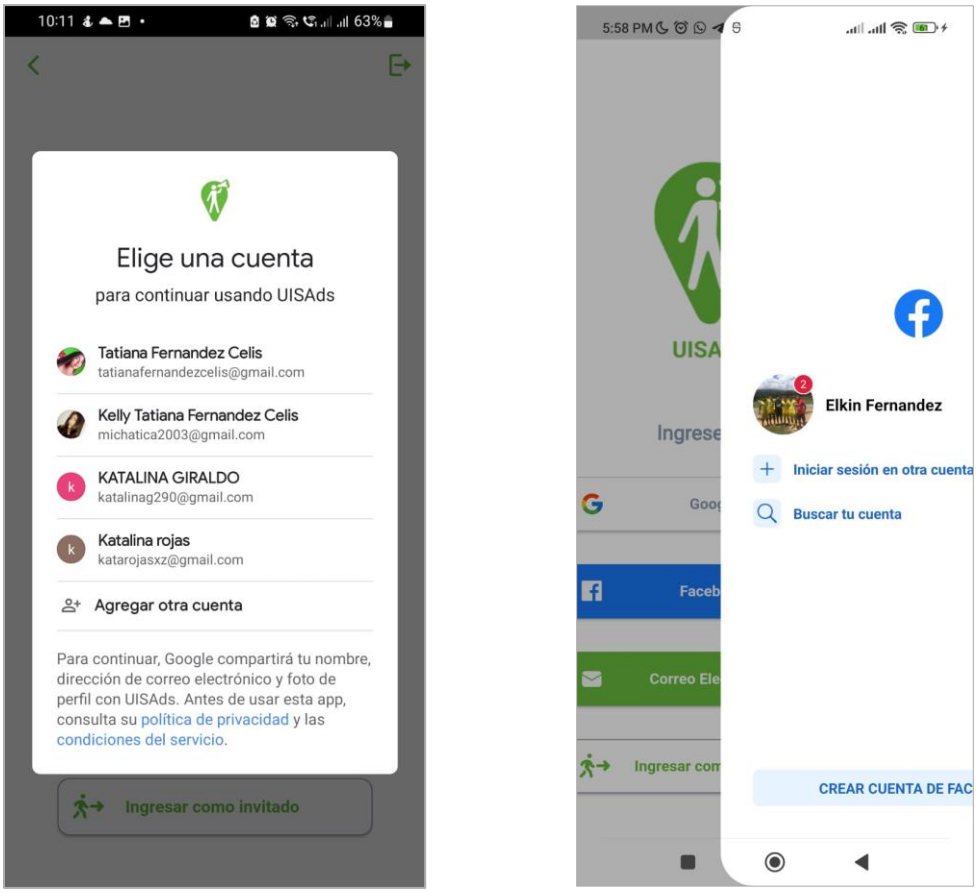
Figura 45. *Pantallas finales ingreso a la aplicación inicial e inicio otros medios*



En la figura 45 se muestran las pantallas finales en el aplicativo enfocadas en el inicio de sesión, autenticación y registro dentro del aplicativo, a través de estas pantallas el usuario puede ingresar al aplicativo de 4 maneras distintas, sea por medio de la autenticación con Google, Facebook, Correo electrónico registrado en la app, o el ingreso a la app como invitado, las personas

pueden usar los medios de autenticación de Google y Facebook integrado en la app para por medio de sus sesiones iniciadas en sus dispositivos en sus plataformas de redes sociales, puedan autenticarse en UISAds y acceder a los anuncios publicados en la plataforma.

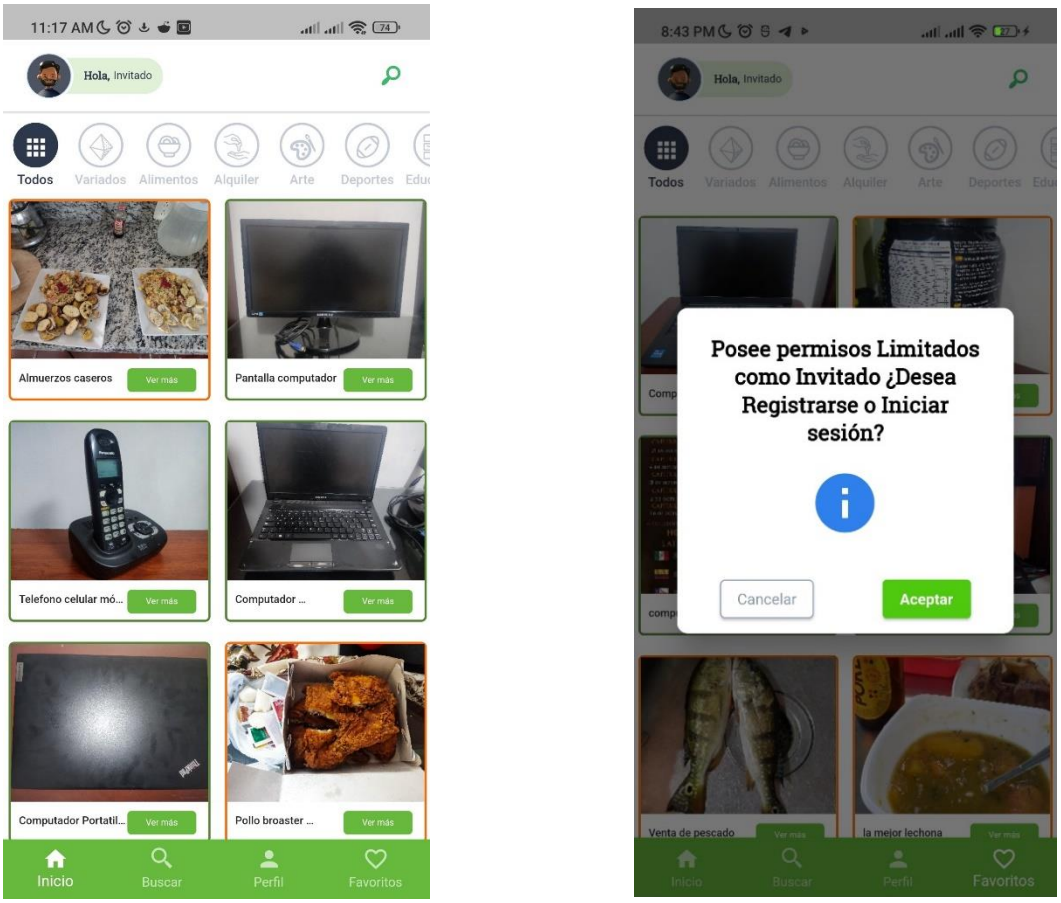
Figura 46. *Pantallas finales autenticación otros medios Google y Facebook*



En la figura 46 se muestran las pantallas finales al intentar autenticarse con Google y Facebook en el aplicativo, para el caso del ingreso por Google se despliega un Modal dispuesto por la librería de Google Auth en Flutter y permite si se tiene iniciada la sesión de una cuenta Google Play en el dispositivo seleccionar una de estas, en caso contrario permite añadir una cuenta e intentar autenticarse en la plataforma por medio de la cuenta agregada, para la autenticación con Facebook la librería Facebook Auth en Flutter permite que si el usuario no está conectado a la app

de Facebook redirigirse a un navegador para autenticarse con Facebook y proceder a autenticarse en UISAds.

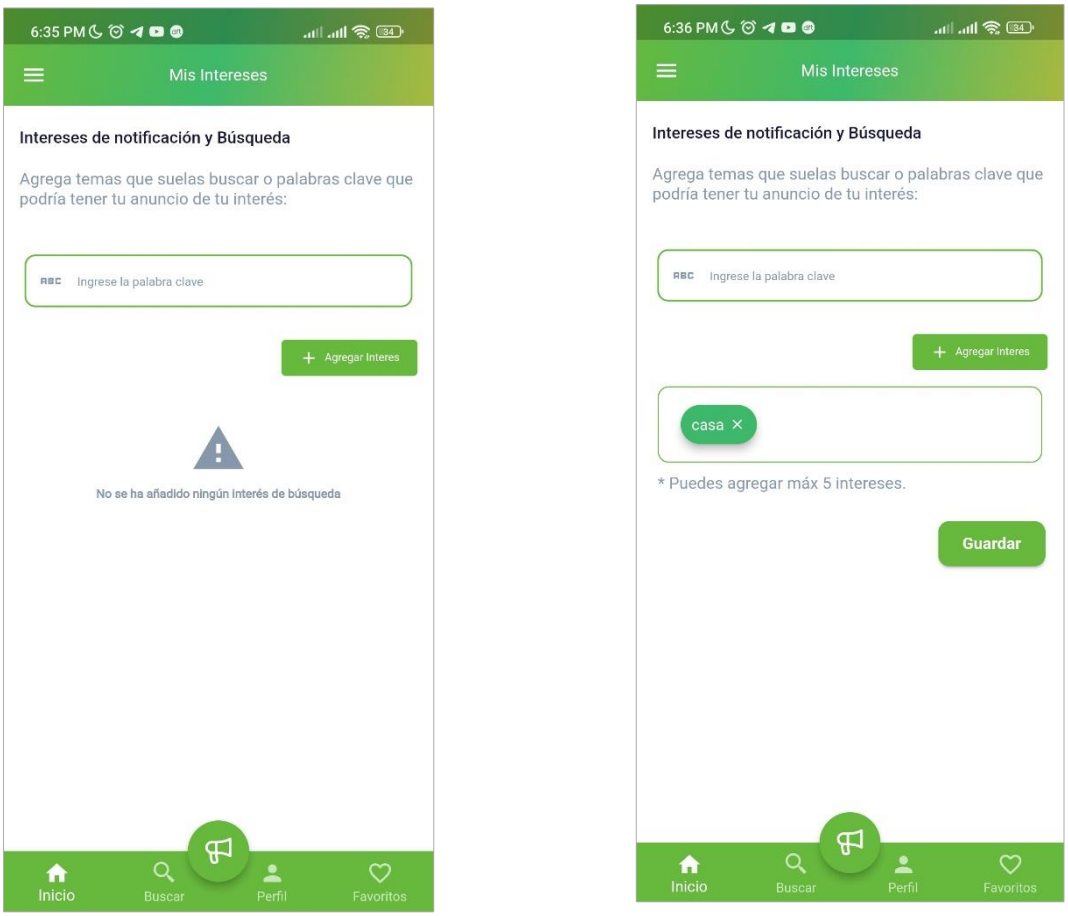
Figura 47. *Pantallas finales ingreso a la app como invitado y anuncio invitado*



En la Figura 47 se observan las pantallas finales al ingreso a la app como invitado la primera hace referencia a cuando un invitado ingresa a la app lo que puede ver de la app y a lo que no puede acceder en la app con privilegios limitados, recordados con un modal que le indica que si desea realizar más acciones además de visualizar anuncios se registre o ingrese a la aplicación si desea realizar acciones más específicas, tales como un reporte de anuncio o navegar a pantallas

tales como perfil del usuario o favoritos, ya que el invitado no tiene acceso a eso, solamente a visualizar la información presente en los anuncios a excepción de los datos para contactar al anunciante.

Figura 48. *Pantallas finales agregación de intereses de un usuario*



En la Figura 48 se compone la vista de intereses del usuario, se dividió en dos pantallas en las cuales se muestra los intereses que el usuario ha seleccionado, en caso de no haber seleccionado ninguno interés se muestra al usuario un widget con icono y texto que indica que no se ha agregado un interés de búsqueda, en caso de que se empiece a agregar intereses aparecerán en el cuadro de

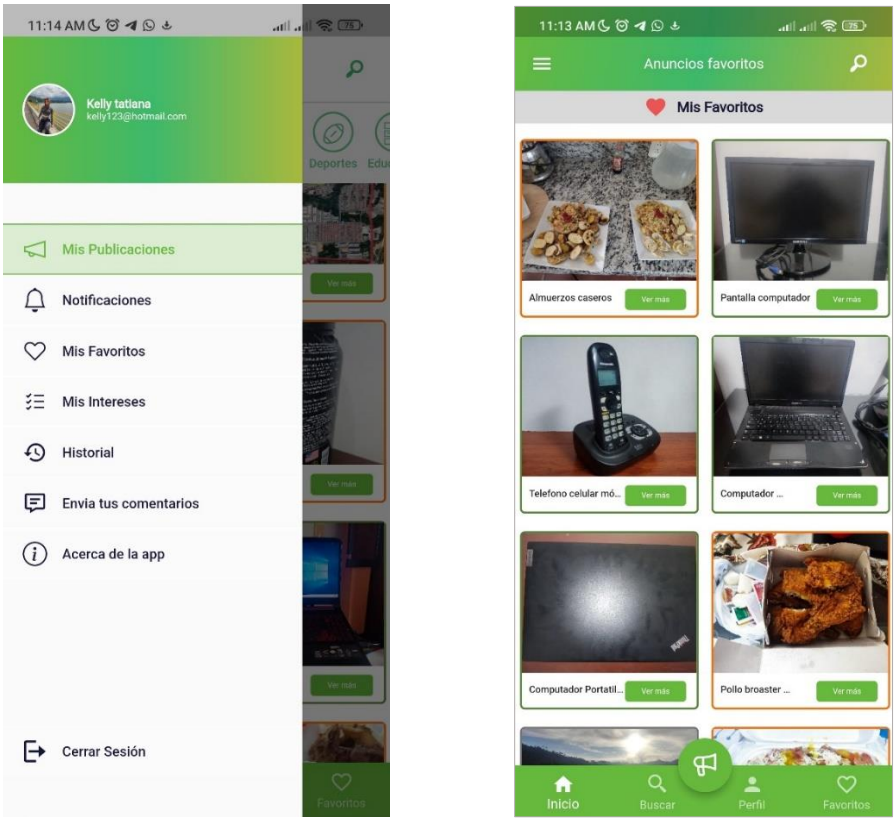
intereses, una lista de máximo cinco intereses en el recuadro, si la lista está llena o se desea eliminar un interés, se da clic en el icono “X” para eliminar un interés de la lista de intereses.

Figura 49. *Pantallas finales notificaciones en base a intereses*

En la Figura 49 se muestra la pantalla de notificaciones, esta pantalla de notificaciones contiene en su Appbar en la parte superior, un Switch para el control de si se quieren recibir notificaciones en base a los intereses seleccionados por el usuario, es de destacar que por defecto la opción esta desactivada, si el usuario desea revisar las notificaciones en base a los intereses que agregó, además de tener una lista de intereses guardada, debe activar este switch, así cada vez que

este inicie sesión verá en una notificación en pantalla la lista de anuncios reciente con base en los intereses que seleccionó, en caso de que no posea intereses seleccionados o no hayan anuncios recientes con base a sus intereses, entonces el usuario en su pantalla de notificaciones tendrá el mensaje expresado en la figura de “No hay notificaciones en base a sus intereses seleccionados”.

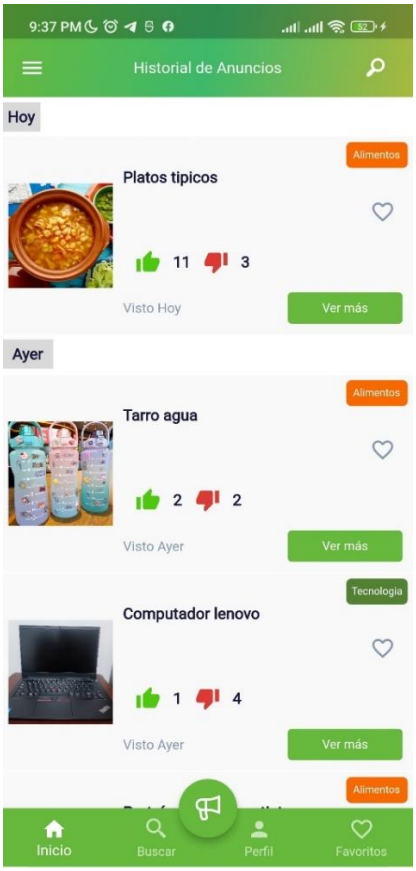
Figura 50. *Pantallas finales anuncios favoritos*



En la Figura 50 se muestra la pantalla de los anuncios favoritos que ha seleccionado un usuario, para navegar a esta pantalla de favoritos se presentan dos opciones, la primera es navegar directamente desde la parte inferior del BottomNavigationBar, en la opción Favoritos, esto llevará al usuario a la página de favoritos, donde puede ver la lista de favoritos que ha seleccionado recientemente. La segunda opción de navegación esta expresada en la parte superior del Appbar el icono de menú, al seleccionarlo se muestran las opciones a las que tiene acceso la persona

autenticada, en este caso, a notificaciones, perfil del usuario, intereses, historial, entre otras. Pero para este caso con seleccionar la opción de Mis Favoritos se redirigirá a la pantalla de favoritos seleccionados, donde se pueden visualizar los anuncios favoritos.

Figura 51. *Pantalla final historial de anuncios*



En la figura 51 se muestra la vista del historial de anuncios de la aplicación, medio por el cual el usuario de la aplicación puede revisar la lista de anuncios que ha consultado últimamente, con un límite de anuncios en el historial almacenados siendo de 100 definido en los requerimientos funcionales, se mostrarán en esta pantalla los anuncios más recientes con este límite de anuncios, es para destacar que la lista de anuncios mostrada en pantalla al usuario viene paginada, para mantener el rendimiento de la app y la construcción de los anuncios en la parte visual en base a lo

que el usuario proceda a navegar por la pantalla, la visualización también se hace por orden de los más recientes visualizados a los más antiguos, cumpliendo con el límite establecido.

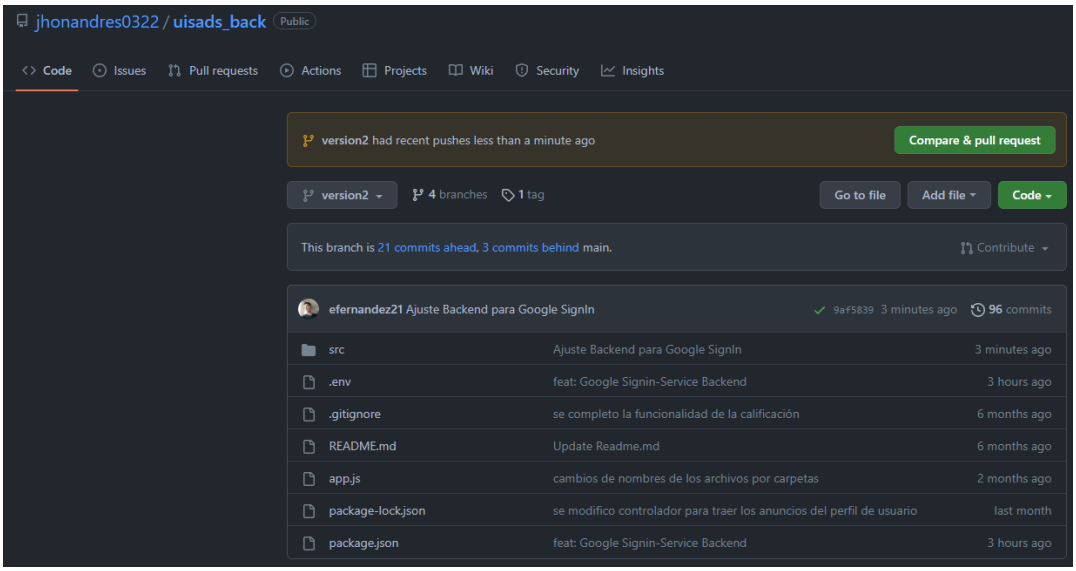
5.4.5 Trazabilidad

Se manejo la trazabilidad del código usando la plataforma Github, cuya base es Git para el control de versiones en el desarrollo de software, se usaron 2 repositorios que ya estaban previamente creados debido a que este proyecto es una continuación del prototipo 1 inicial pero enfocado en una nueva versión del aplicativo en base a los requerimientos planteados en este documento, los repositorios previamente mencionados están divididos en “uisads_back” que es el repositorio remoto en el cual se maneja el backend del aplicativo y “uisads_app” que es el repositorio remoto para el Frontend, en las Figuras 52 y 53 se muestran a continuación.

5.4.5.1 Componente Backend

El componente del back del aplicativo está ubicado en el repositorio “uisads_back” que se puede acceder mediante el siguiente link https://github.com/jhonandres0322/uisads_back/tree/version2

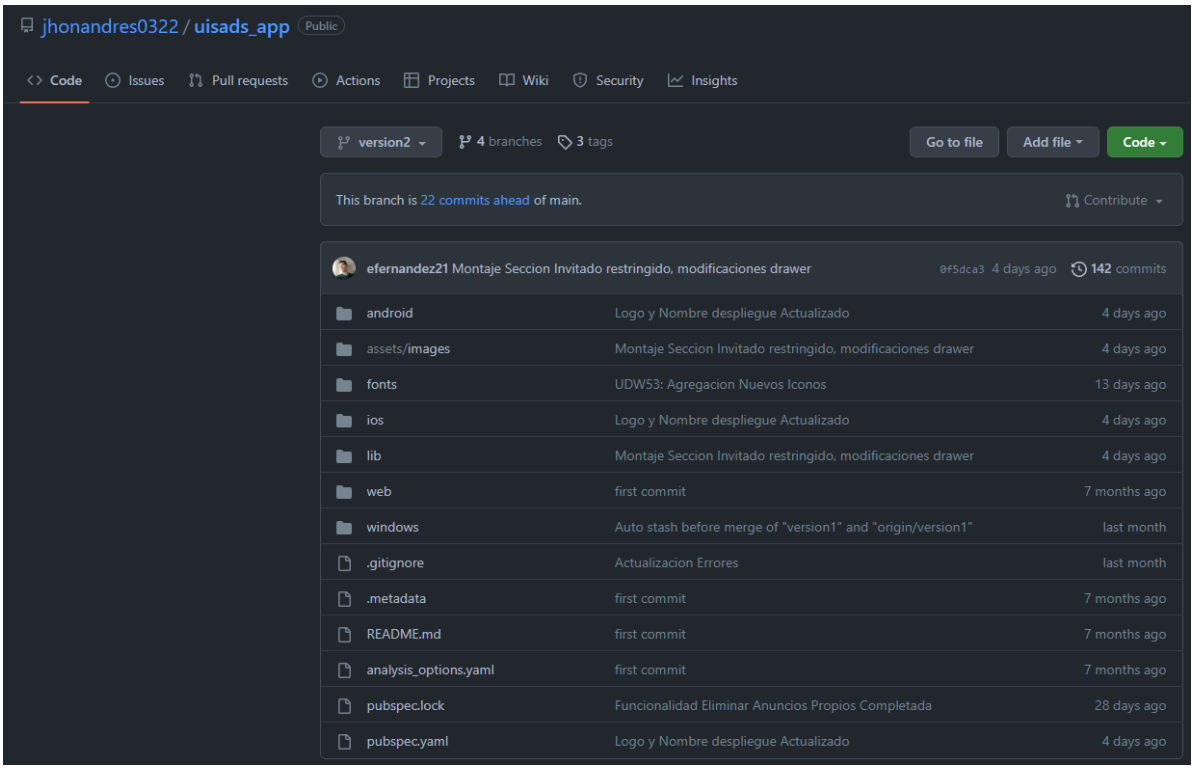
Figura 52. *Repositorio componente Github backend*



5.4.5.2 Componente Frontend

El componente del front del aplicativo está ubicado en el repositorio “uisads_app” que se puede acceder mediante el siguiente link https://github.com/jhonandres0322/uisads_app

Figura 53. *Repositorio componente Github frontend*


jhonandres0322 / uisads_app
Public
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
version2
4 branches
3 tags
Go to file
Add file
Code
This branch is 22 commits ahead of main.
Contribute
efernandez21 Montaje Seccion Invitado restringido, modificaciones drawer
0f5dca3 4 days ago
142 commits
android Logo y Nombre despliegue Actualizado 4 days ago
assets/images Montaje Seccion Invitado restringido, modificaciones drawer 4 days ago
fonts UDW53: Agregacion Nuevos Iconos 13 days ago
ios Logo y Nombre despliegue Actualizado 4 days ago
lib Montaje Seccion Invitado restringido, modificaciones drawer 4 days ago
web first commit 7 months ago
windows Auto stash before merge of "version1" and "origin/version1" last month
.gitignore Actualizacion Errores last month
.metadata first commit 7 months ago
README.md first commit 7 months ago
analysis_options.yaml first commit 7 months ago
pubspec.lock Funcionalidad Eliminar Anuncios Propios Completada 28 days ago
pubspec.yaml Logo y Nombre despliegue Actualizado 4 days ago

6. Pruebas

En busca de validar el correcto funcionamiento de los requerimientos planteados se diseñaron las siguientes pruebas al aplicativo.

6.1 Pruebas de Integración

Se tomó como punto de partida el requerimiento funcional RF02, debido a que el RF01 comparte similitudes con el funcionamiento de los requerimientos funcionales en la aplicación, por lo tanto, para no repetir resultados en el documento de resultados, se colocó el resultado aquí presente.

Tabla 37. *Prueba Inicio sesión con Google*

Iniciar sesión Google		
Email Registrado en Google	correoregistrado1@gmail.com	correonoregis@gmail.com
Respuesta	Mensaje: “Inicio de sesión correcto por medio de Google”	Mensaje: “No existe una cuenta con el correo correonoregis@gmail.com”
Resultado	Correcto	Correcto

Tabla 38. *Prueba Autenticación con Google y envió de Tokens al Back*

Autenticación Token generado Google			
Token autenticación	tokenautenticacionvalido	tokenautenticacionnovalido	-

Respuesta	Token Google valido	Token de google no valido	No se recibió token en la petición
Resultado	Correcto	Correcto	Correcto

Tabla 39. *Prueba inicio sesión Google con usuario bloqueado en BD*

Inicio de sesión	
Inicio de sesión	Inicio de sesión con correo registrado y usuario bloqueado por intentos en correo electrónico
Respuesta	Mensaje: "El usuario se encuentra bloqueado, Contacte al administrador"
Resultado	Correcto

Tabla 40. *Prueba Inicio sesión con Facebook*

Iniciar sesión Google		
Email Registrado en Facebook	correoregistrado2@gmail.com	correonoregis2@gmail.com
Respuesta	*Mensaje: "*Inicio de sesión correcto por medio de Facebook"	*Mensaje*: "No existe una cuenta con el correo correonoregis@gmail.com"

Resultado	Correcto	Correcto

Tabla 41. *Prueba Autenticación con Facebook y envió de Tokens al Back*

Autenticación Token generado Facebook					
Token autenticación	tokenautenticacionval	tokenautenticacionnoval	tokenautenticacionval	tokenautenticacion	-
UserId	useridvalido	useridvalido	useridnovalido	-	useridvalido
Respuesta	Token Facebook valido	Mensaje: "Token de Facebook no valido"	Mensaje: "Token de facebook no valido, por favor verifique"	Mensaje: "No se recibió el id del usuario en la petición"	Mensaje: "No se recibió token en la petición"
Resultado	Correcto	Correcto	Correcto	Correcto	Correcto

Tabla 42. *Prueba inicio sesión Facebook con usuario bloqueado en BD*

Inicio de sesión	
Inicio de sesión	Inicio de sesión con correo registrado y usuario

	bloqueado por intentos en correo electrónico
Respuesta	El usuario se encuentra bloqueado, Contacte al administrador
Resultado	Correcto

Tabla 43. *Prueba agregar anuncio como favorito*

Añadir anuncio favorito		
Botón "Favorito" **no seleccionado**	Seleccionado	-
Botón "Favorito" **seleccionado**	-	Seleccionado
Respuesta	*Anuncio favorito Agregado*	*Anuncio favorito deseleccionado*
Resultado	Correcto	Correcto

Tabla 44. *Prueba reportar anuncio no deseado*

Reportar anuncio no deseado		
Botón "Reportar anuncio" **no seleccionado**	Seleccionado	-
Botón "Reportar anuncio" **seleccionado**	-	Seleccionado
Respuesta	*Mensaje: "¿Desea reportar este anuncio como inapropiado?"*	*Mensaje: "¿Desea quitar el reporte al anuncio seleccionado?"*
Resultado	Correcto	Correcto

Tabla 45. *Prueba reportar anuncio no deseado confirmación modal*

Reportar anuncio no deseado confirmación		
Opción "Reportar"	Seleccionado	-
Opción "Cancelar"	-	Seleccionado
Respuesta	*Mensaje: "Se ha Reportado el anuncio exitosamente"*	*Anuncio no reportado*

Resultado	Correcto	Correcto

Tabla 46. *Prueba añadir tema de interés*

Agregar interés					
Interés	Interes1	-	Interes1	In	Interes6
Respuesta	*Interés agregado*	*Mensaje: "Interés vacío o repetido"*	*Mensaje: "Interés vacío o repetido"*	*Mensaje: "Interés menor a 3 caracteres"*	*Mensaje: "No se puede agregar más de 5 intereses"*
Resultado	Correcto	Correcto	Correcto	Correcto	Correcto

Tabla 47. *Prueba guardar temas de interés agregados*

Guardar temas de interés

Botón para **“**Guardar”	Clic en Guardar
Respuesta	*¿Desea guardar los intereses seleccionados?*
Resultado	Correcto

Tabla 48. *Prueba Guardar intereses confirmación modal*

Guardar intereses confirmación		
Opción “Guardar”	Seleccionado	-
Opción “Cancelar”	-	Seleccionado
Respuesta	*Mensaje: “Se han guardado los intereses seleccionados”*	*Intereses no guardados*
Resultado	Correcto	Correcto

Tabla 49. *Prueba Ingreso sesión como invitado*

Iniciar sesión como invitado	
Botón "I*ngresar como invitado"*	Clic en el botón
Respuesta	Ha ingresado como invitado
Resultado	Correcto

Tabla 50. *Prueba Buscar anuncio, ver perfil y navegar a favoritos en invitado*

Buscar anuncio, ver perfil y navegar a favoritos en ingreso invitado			
Opción *"Buscar"*	Seleccionado	-	-
Opción *"Ver perfil"*	-	Seleccionado	-
Opción *"Ver favoritos"*	-	-	Seleccionado
Respuesta	*Mensaje: "Posee permisos limitados como Invitado*	*Mensaje: "Posee permisos limitados como Invitado ¿Desea*	*Mensaje: "Posee permisos limitados como Invitado ¿Desea*

	¿Desea registrarse o iniciar sesión?"	*registrarse o iniciar sesión?"*	*registrarse o iniciar sesión?"*
Resultado	Correcto	Correcto	Correcto

Tabla 51. *Prueba acción restringida confirmación modal*

Buscar anuncio, ver perfil y navegar a favoritos confirmación modal		
Botón "Aceptar"	Seleccionado	-
Botón "Cancelar"	-	Seleccionado
Respuesta	*Navegación a pantalla de inicio aplicación*	*Mantenerse en invitado*
Resultado	Correcto	Correcto

Tabla 52. *Prueba consultar historial de anuncios*

Consultar historial de anuncios

Opción del menú Drawer	Clic en Menú y en Historial
Respuesta	*Pantalla con el listado de anuncios ordenados desde el más reciente al más antiguo.*
Resultado	Correcto

Tabla 53. *Prueba consultar favoritos*

Consultar anuncios favoritos	
Opción del menú Drawer	Clic en Menú y en Mis Favoritos
Respuesta	*Pantalla con el listado de anuncios f*avoritos *ordenados desde el más reciente al más antiguo.*
Resultado	Correcto

Tabla 54. *Prueba ver anuncios notificados*

Ver anuncios notificados nuevos en base a los intereses	
Opción del menú Drawer	Clic en Menú y en Historial
Respuesta	*Pantalla con el listado de anuncios recientes notificados ordenados desde el más reciente al más antiguo.*
Resultado	Correcto

Tabla 55. *Prueba desactivar notificaciones de intereses*

Desactivar notificaciones de intereses	
Switch "Desactivar Notificaciones"	Clic en el Switch
Respuesta	*Mensaje "Se han desactivado las notificaciones a este perfil"*

Resultado	Correcto

Tabla 56. *Prueba activar notificaciones de intereses*

Activar notificaciones de intereses	
Switch "Activar Notificaciones"	Clic en el Switch
Respuesta	*Mensaje "Se han Activado las notificaciones a este perfil"*
Resultado	Correcto

6.2 Verificación de requerimientos

6.2.1 Verificación de requerimientos funcionales

Tabla 57. *Registro de comprobación requerimientos funcionales.*

Requerimiento	Implementación	Estado
RF1	Funcionalidad implementada correctamente en la pantalla de Registro de usuario.	OK
RF2	Funcionalidad implementada correctamente en la pantalla Inicio de sesión o login.	OK
RF3	Funcionalidad implementada correctamente en las pantallas de Inicio sesión y principal invitado.	OK
RF4	Funcionalidad implementada correctamente en las pantallas de principal invitado y anuncio invitado.	OK
RF5	Funcionalidad implementada correctamente en la pantalla de anuncio, acción “Agregar Favorito”.	OK
RF6	Funcionalidad implementada correctamente en la pantalla de anuncio, acción “Agregar Favorito” y la pantalla de mis favoritos con la lista de anuncios.	OK
RF7	Funcionalidad implementada correctamente en la pantalla de historial de anuncios accesible mediante el menú del perfil del usuario.	OK
RF8	Funcionalidad implementada correctamente en las pantallas de mis favoritos, en la lista de anuncios y la pantalla de anuncio en la opción de favorito agregado.	OK
RF9	Funcionalidad implementada correctamente en el backend del aplicativo y visualizada en la pantalla del historial de anuncios, accesible mediante el menú de perfil de usuario, opción historial.	OK

RF10	Funcionalidad implementada correctamente en la pantalla de temas de interés, vista por medio del perfil de usuario, opción mis intereses.	OK
RF11	Funcionalidad implementada correctamente en la pantalla de temas de interés, controlada mediante validaciones del front y del back a recibir la información enviada.	OK
RF12	Funcionalidad implementada correctamente en la pantalla de temas de interés, validaciones por parte de front y back para la eliminación de los intereses.	OK
RF13	Funcionalidad implementada correctamente en la pantalla de notificaciones, con base en los intereses definidos por el usuario en la pantalla de intereses.	OK
RF14	Funcionalidad implementada correctamente en la pantalla de notificaciones, accesible mediante el menú del perfil del usuario, opción notificaciones.	OK
RF15	Funcionalidad implementada correctamente en la pantalla de anuncio, opción “Reportar anuncio”.	OK
RF16	Funcionalidad implementada correctamente en la pantalla de anuncio, opción “Reportar anuncio”, ejecuta una ventana emergente para la confirmación de la acción por parte del usuario.	OK
RF17	Funcionalidad implementada correctamente en las pantallas principal invitado y visualización anuncio restringido.	OK

6.2.2 Verificación de requerimientos no funcionales

Tabla 58. *Registro de comprobación requerimientos no funcionales.*

Requerimiento	Implementación	Estado
RNF1	Funcionalidad implementada al estar soportado el prototipo del aplicativo en un servidor que garantiza el funcionamiento continuo, mediante una cuenta estudiantil en Heroku, en caso de aumentar la cantidad de usuarios, por medio de Heroku se pueden aumentar los recursos tales como almacenamiento y la cantidad de RAM, debido a que este plantea un modelo Platform as a Service (*PaaS).*	OK
RNF2	Funcionalidad implementada, se actualizaron iconografía, enfoque de colores, y retroalimentación de acciones realizadas en la app por parte del usuario.	OK
RNF3	Funcionalidad implementada al corroborar que los tipos de datos ingresados y respuesta recibida es válida, mediante las pruebas de integración realizadas.	OK
RNF4	Funcionalidad implementada al corroborar mediante la lógica implementada tanto en back y en front para no permitir el almacenamiento mayor al permitido, definido con los requerimientos funcionales.	OK
RNF5	Funcionalidad implementada mediante la verificación mediante el protocolo Auth0 a través de las plataformas de Google y Facebook.	OK

RNF6	Funcionalidad implementada en el desarrollo del aplicativo con compatibilidad para dispositivos Android 5.0 y posteriores.	OK

6.3 Observaciones y restricciones

- El aplicativo está dirigido a personas cercanas o pertenecientes a la comunidad de la Universidad Industrial de Santander, por tal motivo el alcance del aplicativo está limitado a Bucaramanga y sus alrededores, connotando que su uso no está limitado a solo estudiantes UIS (*Triana & Parra, 2022, p.155).*
- El aplicativo fue diseñado inicialmente enfocado en los dispositivos móviles con sistema operativo Android versiones superiores a 5.0 (Lollipop), debido a que el apk compilado fue definido para esta versión y superiores.
- Los archivos de subida del aplicativo son solamente imágenes, el aplicativo internamente no permite la subida de archivos diferentes a fotos e imágenes.
- El aplicativo busca ser solo un mediador entre usuarios interesados en un bien y los anunciantes del mismo, por este motivo el aplicativo no permite pagos o compras dentro de la app (*Triana & Parra, 2022, p.155).*
- El tamaño total del APK generado es de 9,2 MB en su versión de producción, el dispositivo debe tener aproximadamente en espacio de almacenamiento unas 100 MB para el almacenamiento de la aplicación y los datos del usuario autenticado. Con un teléfono con 2GB de RAM es el requerimiento mínimo para una buena ejecución de la aplicación.

- La sesión de invitado tiene permisos restringidos, solo se puede ver la lista de todos los anuncios disponibles, visualizarlos, pero no se puede contactar al vendedor, ni realizar una calificación, ni reporte esto con el fin de llevar al usuario a que tenga una visión inicial de la app y proceda a registrarse.
- El registro e inicio de sesión de Facebook está restringido a usuarios de prueba agregados directamente en la consola de Meta Developers, esto debido al modo en el que se encuentra la aplicación en modo beta y por lo tanto la solicitud para cargar data de cualquier usuario está en revisión por parte de Facebook para que se habilite para que todos los usuarios puedan acceder a la app, por medio de esta red social sin restricciones.

7. Encuesta

En función de realizar las validaciones con los usuarios, se realizó la entrega del aplicativo a un público de prueba de 25 personas, las cuales usaron el aplicativo durante tres días y probaron en sí las funciones más importantes de este, luego de haber realizado las respectivas pruebas a la plataforma versión 2, se les entregó una encuesta validando en ella los objetivos principales del proyecto plasmados en la aplicación y también la búsqueda de puntos de mejora a esta para el futuro.

7.1 Preguntas

Las preguntas realizadas en la encuesta fueron las siguientes:

1. Como meta de este proyecto se tiene el de mejorar la experiencia de usuario, por lo cual, en comparación con la primera versión del aplicativo, ¿percibió usted una mejora en el acceso y navegación dentro de la aplicación?
2. ¿Comparando las vistas de la pantalla principal del aplicativo en sus 2 versiones, teniendo el antes y el después, considera que hubo una mejora en la experiencia del usuario al manejar las interfaces de esta vista?
3. ¿Comparando las vistas de visualización de anuncio del aplicativo en sus 2 versiones, teniendo el antes y el después, considera que hubo una mejora en la experiencia del usuario al manejar las interfaces de esta vista?
4. ¿En escala de 1 a 5 que tan cómodas y útiles considera las notificaciones personalizadas con base en los intereses que selecciona un usuario por el aplicativo?

Donde 1 es no considero las notificaciones cómodas ni útiles y 5 es considero las notificaciones muy cómodas y útiles.

5. ¿En una escala de 1 a 5 que tan útil considera la opción de inicio de sesión como invitado al aplicativo? Donde 1 es no considero para nada útil el acceso como invitado y 5 considero muy útil el acceso como invitado.
6. En una escala de 1 a 5 que tan beneficioso le parece la integración de dos plataformas como Google y Facebook para el ingreso y registro al aplicativo, donde 1 es no me parece para nada beneficioso, y 5 es me parece muy beneficioso.
7. ¿En una escala de 1 a 5 que tan necesaria sentía la incorporación de una funcionalidad de reporte de anuncios al interior del aplicativo? Donde 1 es no me parecía necesaria y 5 es me parecería muy necesaria.
8. En base al historial de anuncios disponible al navegar por el aplicativo ¿En una escala de 1 a 5 que tan necesario sentía la incorporación de esta funcionalidad al interior del aplicativo? Donde 1 es no me parecía necesaria y 5 es me parecería muy necesaria.
9. En base a la característica de añadir anuncios como favoritos, permitiéndole tener un mejor control de los anuncios ¿En una escala de 1 a 5 que tan necesaria sentía la incorporación de esta funcionalidad al interior del aplicativo? Donde 1 es no me parecía necesaria y 5 es me parecería muy necesaria.
10. ¿Consideraría útil la incorporación de una funcionalidad de filtrado de anuncios según la distancia que hay desde donde fue publicado el anuncio hasta el lugar de tu ubicación? (Si/No/Tal vez)

7.2 Resultados

Luego de presentada la encuesta a través de Google Forms a los usuarios de prueba del aplicativo a las 25 personas disponibles, todas estas pertenecientes a la comunidad UIS, en su mayoría estudiantes, los cuales luego de realizar las pruebas en los 3 días, respondieron cada pregunta llevando a recolectar los siguientes resultados.

Figura 54. *Resultados encuesta, pregunta 1*

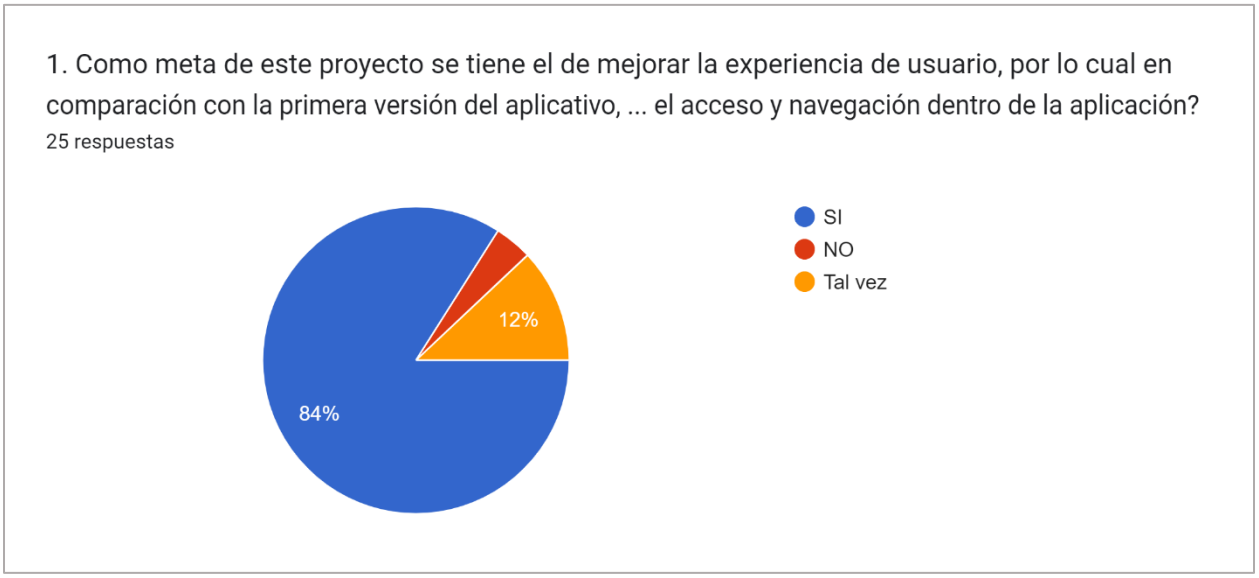


Figura 55. *Resultados encuesta, pregunta 2*

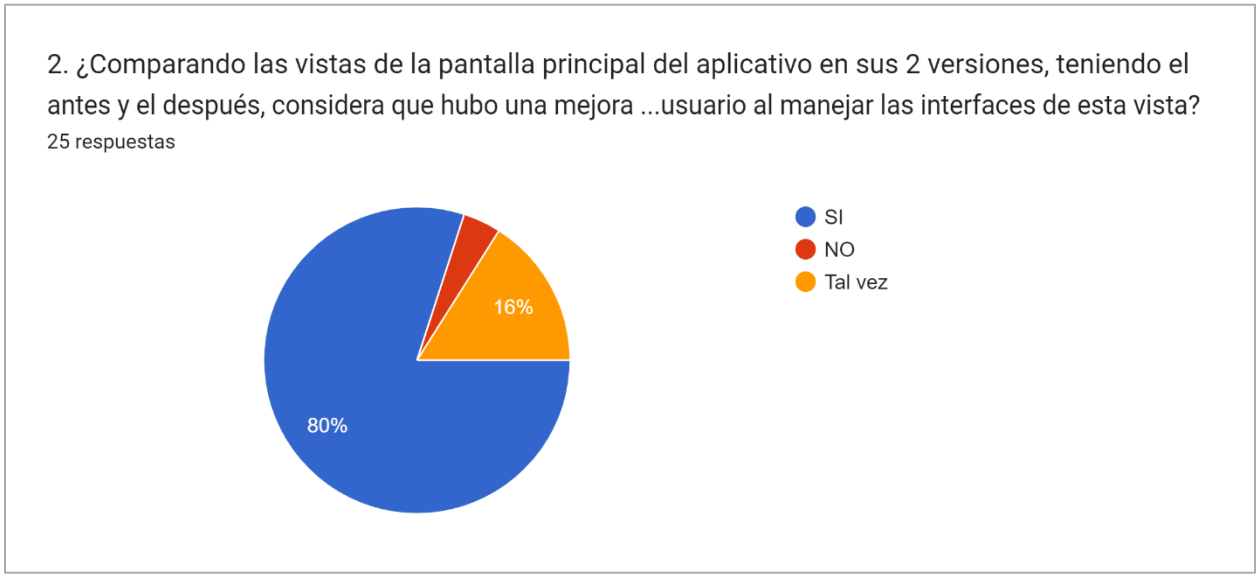


Figura 56. *Resultados encuesta, pregunta 3*

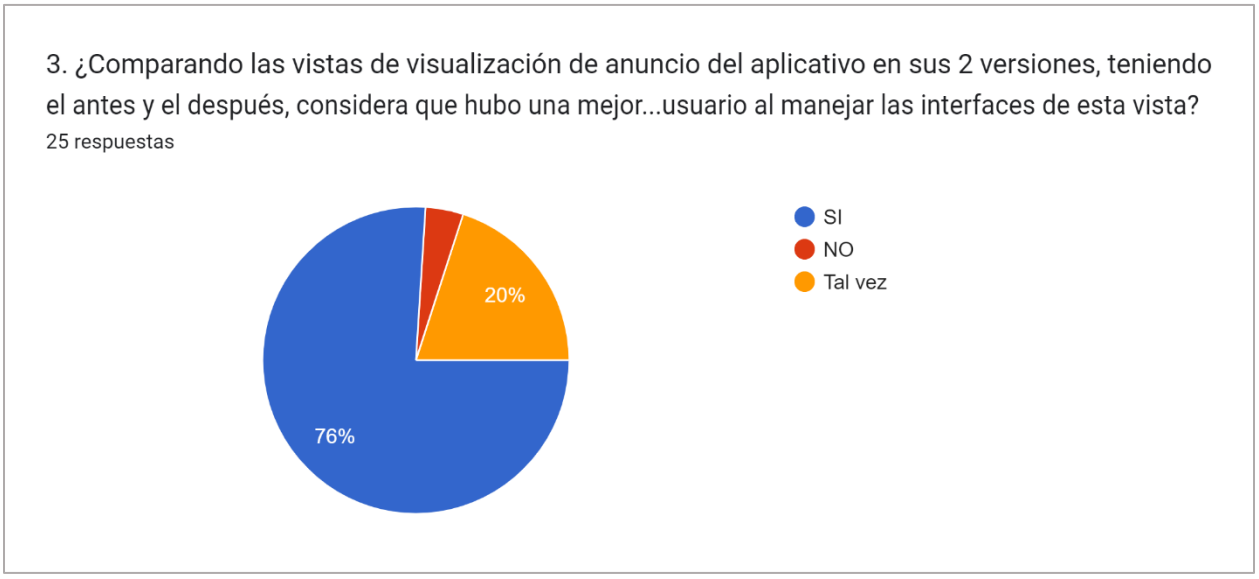


Figura 57. *Resultados encuesta, pregunta 4*

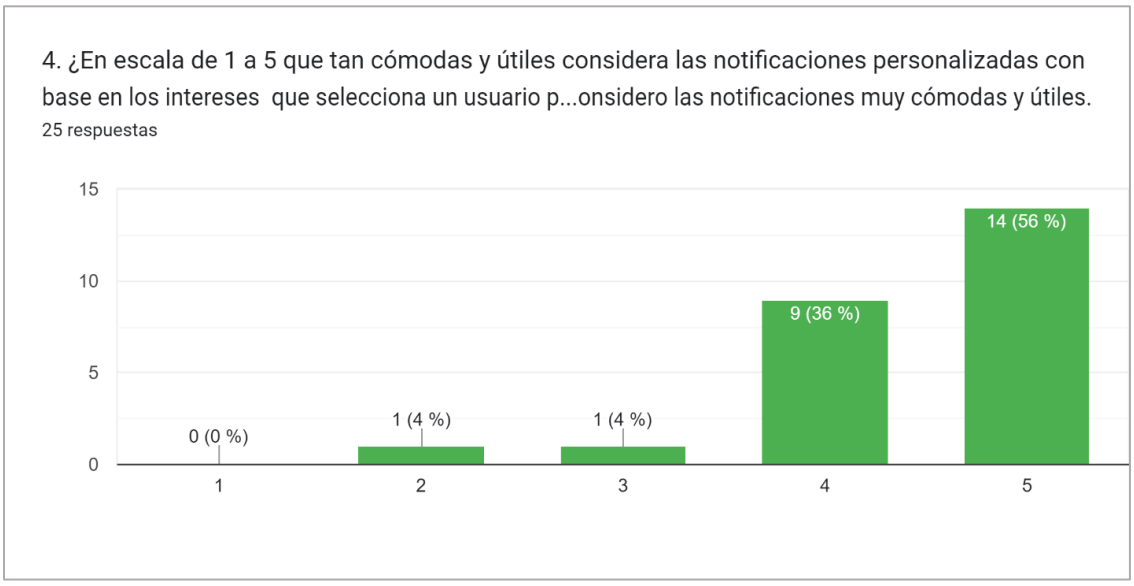


Figura 58. *Resultados encuesta, pregunta 5*

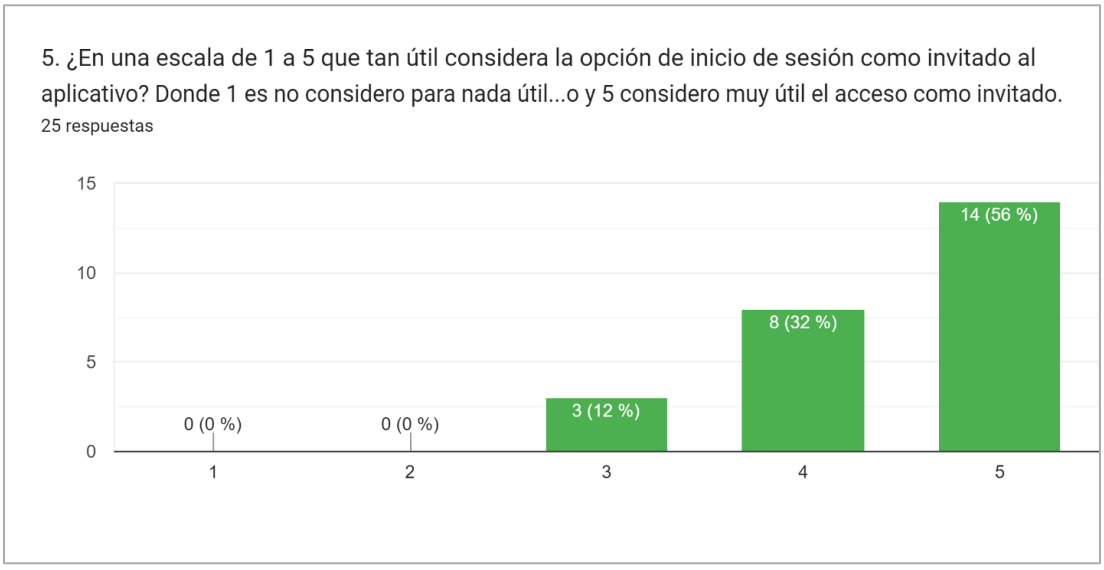


Figura 59. *Resultados encuesta, pregunta 6*

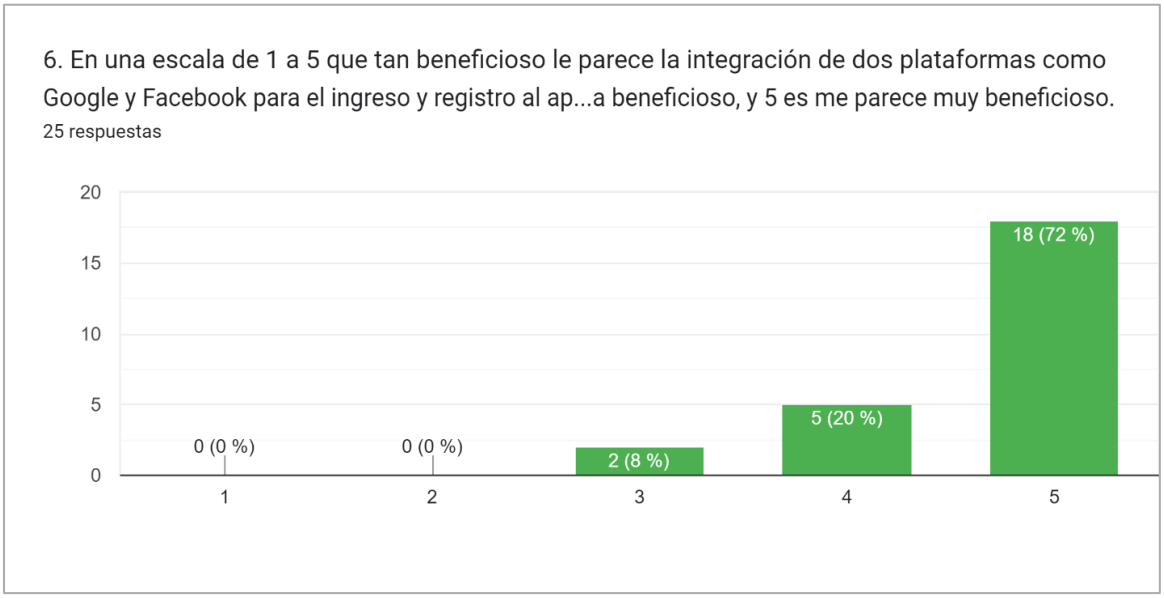


Figura 60. *Resultados encuesta, pregunta 7*

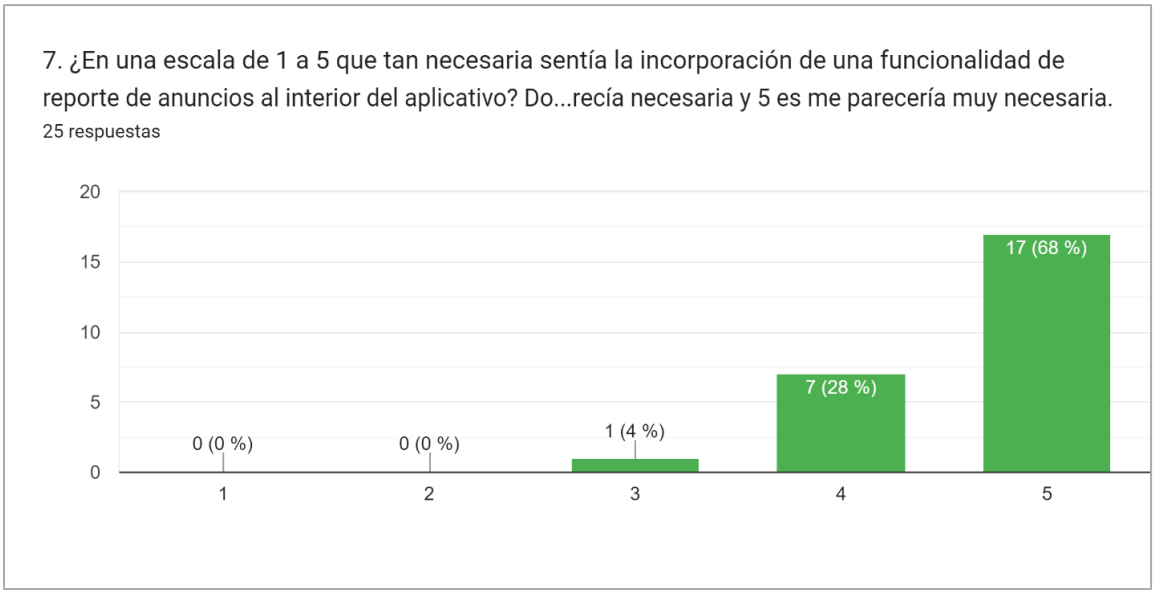


Figura 61. *Resultados encuesta, pregunta 8*

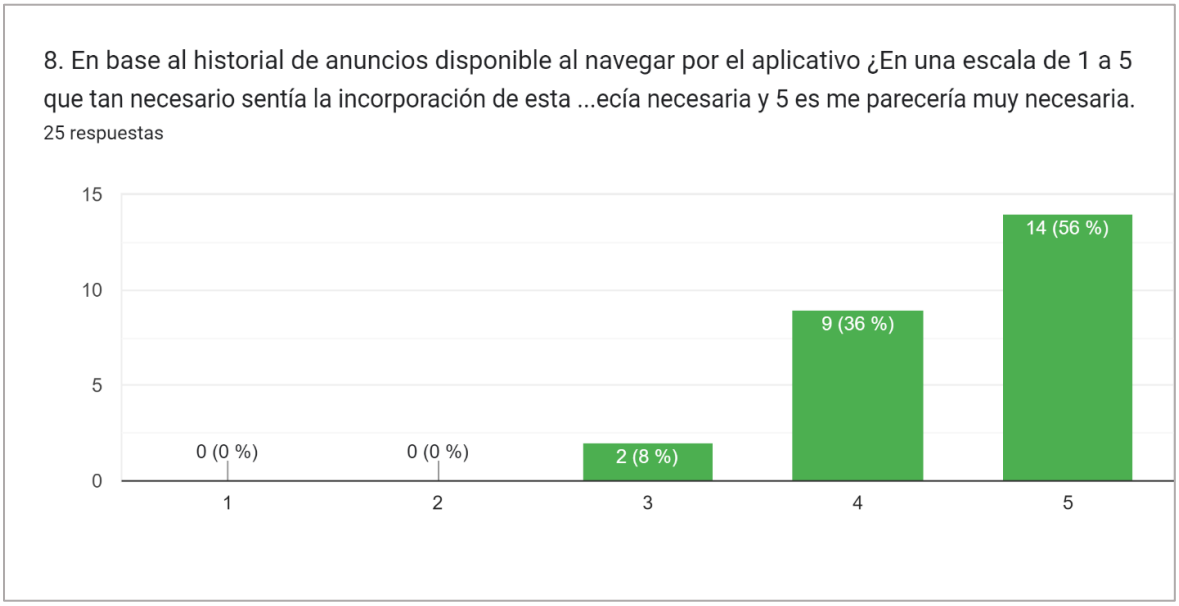


Figura 62. *Resultados encuesta, pregunta 9*

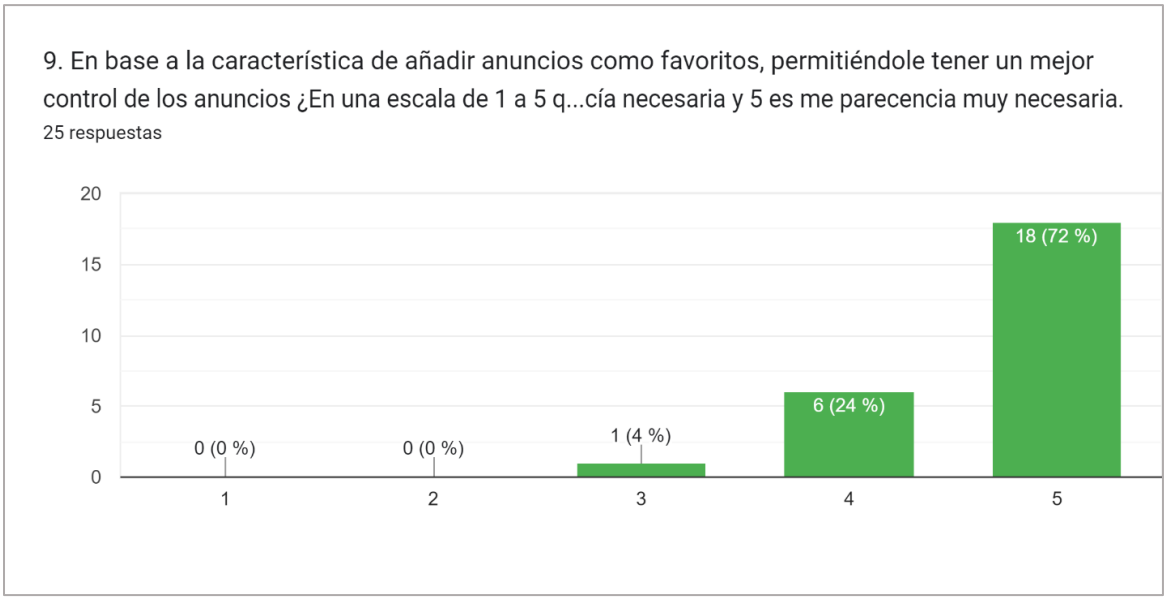


Figura 63. *Resultados encuesta, pregunta 10*

8. Recomendaciones

El software debe estar en continua mejora para evitar su obsolescencia antes de tiempo y pueda seguir siendo usado como una herramienta útil para el usuario que depende de él. El mantenimiento de un producto desarrollado permite mejorar lo inicialmente planeado, corrección de problemas o errores, mejorar el rendimiento, entre otras cosas. El mantenimiento forma parte del ciclo de vida del software, por lo tanto los desarrolladores deben estar constantemente buscando oportunidades de mejora para seguir teniendo un software competitivo que de un valor a los usuarios que lo ejecutan, por esto es importante para la plataforma UISAds, cada oportunidad de mejora que se presenta, este proyecto fue una oportunidad de mejora dada con base en las opiniones de los posibles beneficiarios de su construcción inicial, así que no siendo ajeno a esto, se plantea unas mejoras que para los siguientes prototipos, pueden atrapar más público potencial y permitir una mejor difusión y aceptación del aplicativo, entre estas cosas están las siguientes recomendaciones:

- Implementar un sistema de administración que permita un control de las categorías, anuncios reportados, usuarios mal intencionados, problemas externos con los que se puede encontrar el aplicativo en su fase de producción.
- Permitir además del reporte de anuncios que fue implementado en la aplicación, una forma de reportar al usuario vendedor, debido a que el reporte general está enfocado en el anuncio, sería ideal que al usuario que ofreció este anuncio se le pueda tener un control y realizar un correctivo en el caso anuncios inapropiados.

- Posibilidad de que la aplicación permita una comunicación interna entre el usuario interesado y el vendedor, un chat interno entre ellos, para no tener que ir directamente a WhatsApp.
- Permitir a un usuario que reseñe a otro usuario o realicé un comentario sobre un anuncio especifico sea positivo o negativo.
- En educación, que se pueda implementar un módulo enfocado en personas que ofrezcan tutorías y publicar los horarios de disponibilidad para dictar estas.
- Integrar Google Maps o un servicio de mapas para conocer la ubicación de los anuncios de tipo comida y alquileres, saber la ubicación especifica de los lugares donde se ofrecen este tipo de anuncios.

9. Conclusiones

Surgiendo desde las recomendaciones del proyecto inicial que dio origen a este aplicativo UISAds, se logró plasmar una finalidad principal en el proyecto, la mejora de la experiencia de usuario.

Durante el desarrollo del proyecto se logró utilizar tecnologías que actualmente gozan de un gran uso en el proceso de codificación del desarrollo móvil Frontend y el desarrollo Backend de una aplicación, tecnologías que fueron usadas en la primera versión del aplicativo y donde fue necesario su aprendizaje para poder cumplir las respectivas labores de mantenibilidad de manera óptima, este aprendizaje resultó desafiante al contener temas externos al pensum impartido en la formación académica, y respecto a los cuales se debía tener conceptos claros de su funcionamiento y como se maneja su control de estados, sus clases, su navegación, sus rutas, entre otras características propias de los frameworks para la construcción del aplicativo. Se implementó una metodología en cascada que sigue un proceso en base a lo aprendido en ingeniería de software, abarcando las fases de análisis, diseño, codificación y pruebas de la solución desarrollado según los requerimientos funcionales y no funcionales expuestos. Mediante el proceso de pruebas de integridad realizadas se logró la corrección de pequeños errores e inconsistencias, las cuales pudieron ser corregidas y ajustadas sin afectar negativamente el desempeño del aplicativo y preservando el correcto cumplimiento de los objetivos planteados.

Como puntos importantes durante el desarrollo de este proyecto se detectó el control de versiones y la trazabilidad empleada, puntos que pudieron ser llevados a cabo por medio de la plataforma Jira, con la cual se gestionó la creación de tareas y solución de problemas presentados en el desarrollo del aplicativo, lo que permitió una adecuada gestión de los tiempos de desarrollo

y pruebas realizadas a la plataforma, en Jira se describía una funcionalidad presentada y en base a lo concluido en el análisis y diseño se identificaban las tareas que se necesitaban para cumplir con los requerimientos escritos. Para llevar a cabo el control de versiones fue empleado el software GIT, siendo este un sistema de control de versiones que permite la eficiencia y un adecuado manejo en la gestión de código desarrollado para una solución software, teniendo como complemento la plataforma Github para el manejo de estos repositorios de código de forma remota, permitiendo la colaboración entre los diferentes desarrolladores que trabajan en un software, y en el marco de este proyecto para efectuar una extensión de un desarrollo previo, donde luego de ser habilitado el acceso a los repositorios de código se logró complementar el desarrollo de este aplicativo.

Se logró comprobar que las nuevas características incorporadas al aplicativo tuvieron una buena recepción por parte de los usuarios, donde en el caso particular de las notificaciones de anuncios con base a sus intereses, los usuarios son recibidos, al ingreso en la aplicación, con anuncios recientes de sus intereses, dando valor a lo que el usuario considera relevante, haciéndole sentir que sus intereses son relevantes para el aplicativo y este los tiene presentes desde el primer momento en que el inicia sesión en él. La funcionalidad de almacenamiento del historial de anuncios visualizados le brinda al usuario la posibilidad de poseer un control y un registro de su navegación por el aplicativo. La sección de anuncios favoritos y la funcionalidad de marcarlos como tal son la forma en la que el usuario puede sentir que no perderá el acceso a un anuncio de su interés, brindándole la posibilidad poder verlo otra vez en otra oportunidad. Se puso una especial atención a los comentarios de los usuarios encuestados en las pruebas realizadas en la primera versión del proyecto, comentarios que apuntaban a la necesidad de nuevos métodos de ingreso al aplicativo, como un ingreso con métodos de autentificación muy comunes como Facebook y Google que les permitieran un ingreso más sencillo, seguro, ágil y cómodo al aplicativo, y un inicio

de sesión como invitado, el cual dotará a nuevos potenciales usuarios activos, un primer y veloz vistazo al entorno y las posibilidades que este ofrece, motivándoles a registrarse en el aplicativo para continuar disfrutando de los recursos dispuestos en este.

En búsqueda de solucionar las necesidades más comunes de los usuarios y apuntando a optimizar el manejo de la información expuesta en los anuncios, se logró establecer un método con el cual un usuario correctamente registrado puede reportar un anuncio que considere inapropiado y que no debería estar en la plataforma, otorgándole una herramienta con la que puede contribuir a llevar un espacio virtual seguro, donde se pueda confiar en el contenido expuesto, dándole al público una vía para combatir el contenido que intente timar o engañar a los usuarios que lo visualizan.

Finalmente, se considera relevante resaltar la importancia de las herramientas Flutter y NodeJs en la construcción de este proyecto, herramientas que vieron su uso en las secciones Frontend y Backend del desarrollo de la aplicación, siendo al día de hoy instrumentos de amplio uso entre la comunidad de desarrollo móvil, cuya enseñanza en el ámbito académico debería ser pertinente y vital en la formación académica de nuevos profesionales en ingeniería de sistemas.

Referencias Bibliográficas

Al-Heeti, A. (2018, 1 mayo). Facebook Marketplace is used in 70 countries, by 800 million people monthly. CNET. Recuperado 21 de diciembre de 2021, de https://www.cnet.com/tech/tech-industry/facebook-marketplace-is-used-in-70-countries-by-800-million-people-monthly/.

Alfonso, R. (2016). Economía colaborativa: Un nuevo mercado para la economía social. CIRIEC-España, Revista de Economía Pública, Social y Cooperativa, 88,231-258. http://ciriec.es/wp-content/uploads/2016/07/COMUN-215-T10-Rosalia-Alfonso-Sanchez-ok.pdf

Arango, D. (2017, 31 marzo). La 'app' para vender artículos usados en su barrio desde su celular. Portafolio.co. Recuperado 21 de diciembre de 2021, de https://www.portafolio.co/negocios/emprendimiento/aplicacion-colombiana-para-hacer-ventas-en-su-barrio-desde-su-celular-504589.

BBVA, API Market. (2020). "API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos". BBVA API Market. Obtenido de https://www.bbvaapimarket.com/es/mundo-api/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos/.

Build and release an Android app. (s. f.). Flutter. Recuperado 16 de septiembre de 2022, de https://docs.flutter.dev/deployment/android.

Campos, R. (2020, 8 julio). Marketplace: ¿qué es y cuál es su importancia? VTEX. Recuperado 18 de diciembre de 2021, de https://vtex.com/latam/blog/operaciones-latam/marketplace-que-es-y-cual-es-su-importancia/.

Cierra el periódico Segundamano después de 30 años de clasificados. (2008, 7 noviembre). El Confidencial. Recuperado 15 de mayo de 2022, de

https://www.elconfidencial.com/archivo/2008/11/07/comunicacion_36_cierra_periodico_segundamano_despues_clasificados.html.

Echeverry, M. (2017, 12 julio). Vendiste: la aplicación colombiana que digitaliza las ventas de garaje. Xataka Colombia. Recuperado 21 de diciembre de 2021, de https://www.xataka.com.co/aplicaciones/vendiste-la-app-colombiana-que-digitalizo-las-ventas-de-garaje.

Editorial La República S.A.S. (2017, 4 marzo). El portal Mercado Libre vendió 181,2 millones de productos en 2016. Diario La República. Recuperado 20 de diciembre de 2021, de https://www.larepublica.co/empresas/el-portal-mercado-libre-vendio-1812-millones-de-productos-en-2016-2479791.

Fielding, Roy Thomas. (2014). "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, Section 4". IETF. Internet Engineering Task Force (IETF). Obtenido de https://tools.ietf.org/html/rfc7231#section-4.

Fielding, Roy Thomas (2000). "Chapter 5: Representational State Transfer (REST)". Architectural Styles and the Design of Network-based Software Architectures (Ph.D.). University of California, Irvine. Obtenido de https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

GenBeta. (2014, 3 febrero). MongoDB: qué es, cómo funciona y cuándo podemos usarlo (o no). Recuperado 13 de junio de 2022, de https://www.genbeta.com/desarrollo/mongodb-que-es-como-funciona-y-cuando-podemos-usarlo-o-no.

Google. (2016, Junio). Principles of Mobile App Design: Engage Users and Drive Conversions [Diapositivas]. Think With Google. https://think.storage.googleapis.com/docs/principles-of-mobile-app-design-engage-users-and-drive-conversions.pdf

Herrera, J. (2020, 2 septiembre). Patrón de diseño MVC. ¿Qué es y cómo puedo utilizarlo? Easy App CODE. Recuperado 19 de diciembre de 2021, de https://www.easyappcode.com/patron-de-diseno-mvc-que-es-y-como-puedo-utilizarlo.

Iqbal, M. (s. f.). Facebook Revenue and Usage Statistics. Business of Apps. Recuperado 21 de diciembre de 2021, de https://www.businessofapps.com/data/facebook-statistics/.

Johnson E. Ralph, Foote Brian. (1988) “Designing Reusable Classes”. "Journal of Object-Oriented Programming", Manning Publications, ISBN 1935182021.

Jones, Michael B.; Bradley, Bradley; Sakimura, Sakimura (2015). “JSON Web Token (JWT)”. ISSN 2070-1721. RFC 7519. Obtenido de https://datatracker.ietf.org/doc/html/rfc7519.

La Red Martínez, D. L., Acosta, J. C., Mata, L. E., Bachmann, N. G., & Vallejos, O. (2012). “Aprendizaje combinado, aprendizaje electrónico centrado en el alumno y nuevas tecnologías”. In VII Congreso de Tecnología en Educación y Educación en Tecnología.

Lee Dami, (2019). “Figma’s new community profiles let users view and remix design files”. The Verge. Obtenido de https://www.theverge.com/2019/10/22/20920929/figma-community-profiles-collaborative-design-beta.

Marketplace Pulse. (s. f.). Mercado Libre Statistics. Recuperado 20 de diciembre de 2021, de https://www.marketplacepulse.com/stats/mercadolibre

Minuto Uno. (2010, 19 agosto). MercadoLibre, en números. Recuperado 20 de diciembre de 2021, de https://web.archive.org/web/20170719225900/http://www.minutouno.com/notas/133942-mercadolibre-numeros

Morpus Nicolás. (2022). “What Is Jira: An Overview of a Unique Project Management Tool”. Fool.com. Obtenido de https://www.fool.com/the-ascent/small-business/project-management/articles/what-is-jira/.

Mozilla. (2020, 8 diciembre). Introducción a Express/Node - Aprende sobre desarrollo web | MDN. Mozilla MDN WebDocs. Recuperado 13 de junio de 2022, de https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction

OnTwice. (2021, 15 noviembre). ¿Qué es la economía colaborativa? Tu Futuro Próximo. Recuperado 19 de diciembre de 2021, de https://tufuturoproximo.santanderconsumer.es/economia-personal/consumo-responsable/que-es-la-economia-colaborativa/

OpenJS Foundation. (s. f.). NodeJS. Node.js. Recuperado 13 de junio de 2022, de https://nodejs.org/es/about/

Real Academia Española. (s. f.). Clasificado, clasificada. En Diccionario de la lengua española. Recuperado 15 de mayo de 2022, de https://dle.rae.es/clasificado?m=form

Triana, J., & Parra, J. (2022). PROTOTIPO DE PLATAFORMA DE ANUNCIOS PARA LA COMUNIDAD UNIVERSITARIA UIS [Tesis de pregrado]. Universidad Industrial de Santander.

Xataka. (2019, 30 octubre). Qué es Github y qué es lo que le ofrece a los desarrolladores. Recuperado 13 de junio de 2022, de https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores