Índice general

Capítulo 1

Introducción

1.1. Contexto y motivación

Se estima que el 60 % de los estudiantes de grado de la Universidad de Granada provienen de otras ciudades de España o del extranjero [1]. Muchos de ellos llegan por primera vez careciendo de información básica sobre qué establecimientos visitar y qué hacer durante su estancia. Son personas jóvenes con intenciones de crear vínculos y grupos con otros sobre sus aficiones y gustos compartidos.

Además de ser una ciudad universitaria, Granada es una ciudad histórica que atrae a muchos turistas que vienen por primera vez con la emoción de descubrir este rincón de España. Estos turistas enfrentan el obstáculo de tener que buscar entre todos los sitios disponibles hasta encontrar el más adecuado a sus preferencias.

Esta situación no es exclusiva de Granada; es un problema común en muchas ciudades del mundo, donde la falta de información sobre qué hacer o dónde ir se convierte en un desafío habitual.

La elección de este tema surge de mi propia experiencia al llegar a Granada por primera vez. En un entorno nuevo y desconocido, sin conocer a nadie y habiendo hecho amigos en la universidad, aún no sabía a qué lugares ir, cuál era el ambiente de esos sitios ni qué tipo de personas los frecuentaban.

Frente a este escenario, el propósito del proyecto es el desarrollo de una aplicación móvil que asista a las personas en la búsqueda de lugares o eventos

de ocio, ofreciendo una lista de establecimientos y eventos. De esta forma, se busca garantizar una experiencia de usuario óptima, permitiendo a los usuarios explorar descripciones detalladas de los establecimientos según el ambiente que ofrecen, y realizar búsquedas personalizadas según sus preferencias, sin necesidad de revisar individualmente cada página web de los establecimientos.

1.2. Objetivos

Dado este enfoque, se han analizado y planteado una serie de objetivos para la creación de una aplicación móvil que ofrezca una solución nueva y eficiente para la gestión y búsqueda de establecimientos orientados al ocio. El objetivo principal es proporcionar al usuario una herramienta que facilite este proceso.

A continuación, se presentarán los objetivos principales de la aplicación, los cuales serán descritos en detalle a lo largo del proyecto:

1. **Registro:** Que los usuarios y administradores de establecimientos puedan registrarse con sus credenciales personales y únicas.

2. **Facilitar la Búsqueda de Establecimientos:** Proporcionar un sistema que permita a los usuarios encontrar establecimientos de ocio según sus preferencias.

3. **Gestión de Eventos y Ofertas:** Permitir la creación, borrado y modificación de eventos y ofertas para que los establecimientos puedan anunciar información que le interese al usuario.

4. **Interacción Social:** Los usuarios pueden seguir a otros, crear actividades y organizar eventos grupales privados.

5. **Reseñas a Establecimientos:** Los usuarios pueden calificar la experiencia en un establecimiento dejando reseñas con una calificación y un mensaje

6. **Interfaz Intuitiva:** Desarrollar una interfaz de usuario que sea fácil de usar garantizando la experiencia de usuario óptima.

Con estos objetivos, salir un día por cuenta propia o con amigos en un entorno desconocido será mucho más sencillo. La aplicación permitirá organizar salidas, encontrar los mejores lugares de ocio y disfrutar de ofertas y

eventos sin la necesidad de navegar por múltiples páginas web. Además, la integración de funcionalidades sociales y de reseñas enriquecerá la experiencia del usuario, haciendo de cada salida una experiencia agradable y bien planificada.

Capítulo 2

Estado del Arte

2.1. Aplicaciones Similares

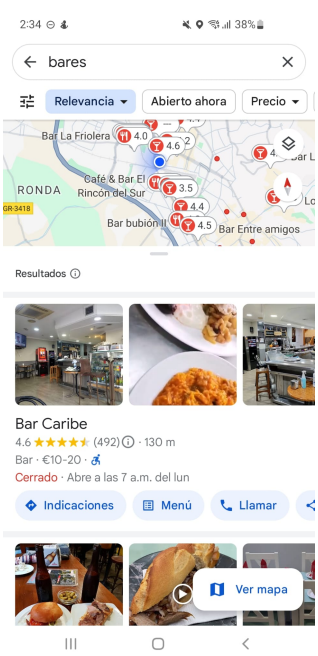
Existen múltiples aplicaciones diseñadas para facilitar la búsqueda de eventos y establecimientos sociales. Por ejemplo, aplicaciones como Yelp, Google y TripAdvisor permiten a los usuarios buscar restaurantes, bares, y eventos según las reseñas y valoraciones de otros usuarios.

1. **TripAdvisor:** Tripadvisor es la plataforma de orientación de viajes más grande del mundo. Ayuda a cientos de millones de personas cada mes en la planificación, reserva y realización de viajes. Los viajeros usan el sitio y la aplicación para descubrir alojamientos, actividades y restaurantes basados en las recomendaciones de otros viajeros.[2]

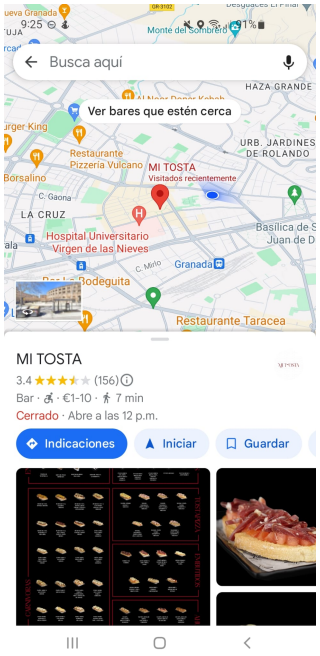
2. **Yelp:** Yelp conecta a las personas con excelentes negocios locales. Con información confiable sobre negocios locales, fotos y contenido de reseñas, Yelp proporciona una plataforma local integral para que los consumidores descubran, se conecten y realicen transacciones con negocios de todos los tamaños, facilitando la solicitud de presupuestos, la unión a listas de espera, la realización de reservas y la programación de citas o compras.[3]

3. **Google:** Google Maps permite a los turistas descubrir una amplia variedad de establecimientos de ocio, como restaurantes, bares, parques, museos y otros puntos de interés. La función de búsqueda integrada facilita encontrar lugares específicos o explorar categorías de interés en una zona determinada.[4]

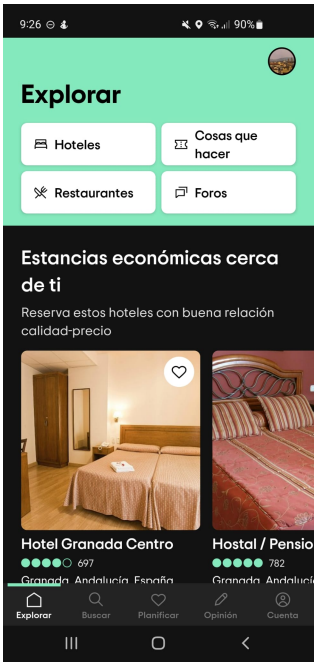
A continuación, se muestran imágenes de las aplicaciones mencionadas para su mejor visualización:



(a) Google Maps

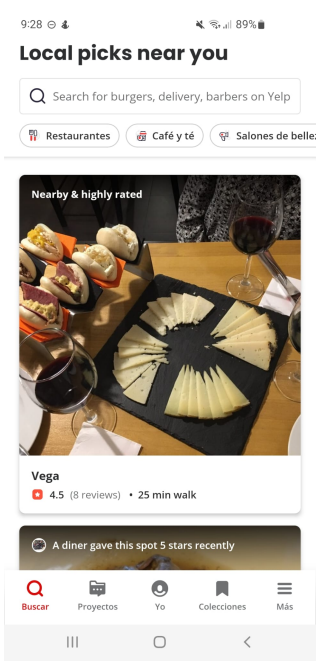


(b) Google Maps

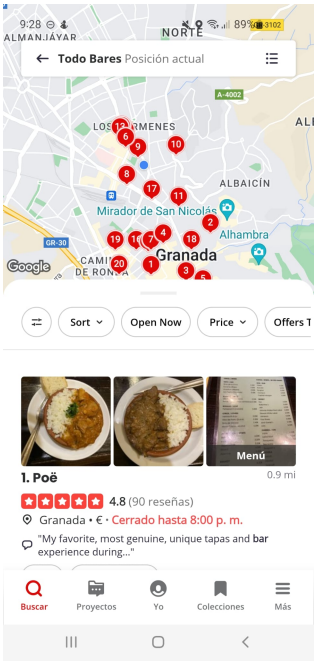


(c) TripAdvisor

(d) TripAdvisor



(e) Yelp



(f) Yelp

Figura 2.1: Comparación de aplicaciones orientadas al turismo

Estas aplicaciones aunque sean excelentes en muchos aspectos tienen ciertas limitaciones como pueden ser:

1. **Información Fragmentada:** Los usuarios deben de consultar distintas plataformas y seguir a diversos establecimientos en sus redes sociales para estar al tanto de eventos u ofertas, lo que resulta en una experiencia fragmentada. Esta dispersión no es sólo ineficiente, sino que puede llevar a que los usuarios pierdan el interés y abandonen la búsqueda con consecuencia que pierdan una oportunidad importante de encontrar ese lugar que buscan.

2. **Interacción Social Limitada:** Estas aplicaciones permiten leer reseñas y calificaciones en un establecimiento por parte de otros usuarios pero no permiten la creación de actividades entre amigos ni fomentan la organización y la interacción social.

3. **Promoción y Gestión para Establecimientos:** La gestión de los perfiles y la promoción de eventos y ofertas en estas plataformas no es una prioridad. Estas aplicaciones tienden a centrarse más en la experiencia del usuario, ofreciendo calificaciones y reseñas de los establecimientos, pero no promueven activamente los lugares con los eventos y ofertas que pueden tener.

4. **Enfoque Amplio y no Exclusivo al Ocio:** Estas aplicaciones abarcan un conjunto de funcionalidades no exclusivamente centradas en el ocio. Por ejemplo, TripAdvisor proporciona información sobre hoteles, atracciones turísticas y restaurantes, mientras que Google requiere búsquedas individuales de cada establecimiento hasta encontrar uno que se ajuste a las necesidades del usuario.

2.2. Soluciones HangOut

Para solucionar estas limitaciones se ha desarrollado HangOut, una aplicación móvil diseñada para asistir a las personas para encontrar un establecimiento o evento adecuado a sus gustos que integra y mejora las funcionalidades existentes con características innovadoras:

1. **Centralización de la Información:** HangOut centraliza la información de diversos establecimientos y eventos, permitiendo al usuario encontrar rápidamente opciones que se ajusten a sus preferencias sin necesidad de consultar múltiples fuentes.

2. **Interacción y Organización Social:** La aplicación permite a los usuarios seguir a otros, crear actividades y organizar salidas grupales de manera más eficiente, eliminando la necesidad de coordinarse a través de múltiples aplicaciones de mensajería.

3. **Herramientas para Administradores de Establecimiento:** HangOut proporciona a los administradores una plataforma para gestionar sus perfiles de establecimiento indicando el ambiente, eventos y ofertas, facilitando así la promoción y gestión de sus servicios de manera más eficaz.

4. **Enfoque Exclusivo:** La aplicación se distingue por su enfoque exclusivo en el ocio y la eficiencia en la organización de eventos. Combinando la funcionalidad de creaciones de actividades en grupos específicos con la búsqueda de establecimientos y la personalización según las preferencias del usuario, ofreciendo una solución más específica para el ocio.

	TripAdvisor	**Yelp**	**Google Maps**
Búsqueda por Filtros	Si	Si	Si
Centralizado	Si	Si	Si
Exclusivo	No	No	No
Organización Social	No	No	No
Promoción de Establecimientos	Si	No	No

Cuadro 2.1: Comparación de características de aplicaciones para el turismo

Capítulo 3

Planificación

3.1. Metodología de Desarrollo Ágil

Las metodologías ágiles de desarrollo de software son ampliamente utilizadas hoy en día debido a su alta flexibilidad y capacidad de adaptación. Estas metodologías permiten a los equipos de trabajo ser más productivos y eficientes, ya que tienen claridad sobre las tareas durante el desarrollo. En el proceso, el software se va adaptando a las nuevas necesidades y requerimientos que vayan surgiendo, facilitando la creación de aplicaciones funcionales. [5]

Este tipo de metodologías de desarrollo tienen ciertas características claves que hace que destaque respecto a las tradicionales:

1. **Flexibilidad y Agilidad:** Permiten la adaptación continua del software a medida que se identifican nuevas necesidades por parte de los usuarios.

2. **Desarrollo Incremental:** En cada ciclo de desarrollo se van agregando nuevas funcionalidades.

3. **Ciclos de Desarrollo Cortos:** Los ciclos de desarrollo, conocidos como iteraciones o sprints, son cortos y permiten la entrega continua de incremento funcionales del software, facilitando un feedback continuo.

3.1.1. Ejemplos de Metodologías

1. **Kanban:** Se basa en dividir las tareas en porciones más pequeñas y organizarlas en un tablero de trabajo con columnas de tareas pendientes, en curso y finalizadas. Este sistema crea un flujo de trabajo visual que ayuda a priorizar las tareas y mejorar el valor del producto.

2. **Scrum:** Es una metodología incremental que organiza los requisitos y tareas en ciclos cortos y fijos, denominados sprints. Las etapas incluyen la planificación del sprint, ejecución, reuniones diarias y revisión de los resultados. Cada iteración completa estas etapas, permitiendo entregar un producto funcional al final de cada sprint.

3. **Lean:** Enfocada en equipos pequeños y altamente capacitados, promueve la rápida ejecución de tareas y valora principalmente el compromiso y la capacidad de aprendizaje del equipo. Los ciclos cortos permiten adaptarse rápidamente a los cambios y mejorar el producto.

Para llevar a cabo mi proyecto, decidí utilizar una metodología de desarrollo ágil como es **Scrum**. Esta elección fue motivada por diferentes razones con el objetivo de maximizar la eficiencia y calidad del producto final. Los principales motivos fueron:

1. **Adaptación a Cambios Continuos:** Al desempeñar el rol de programador y usuario de la aplicación la capacidad de adaptación se volvió un papel muy importante, los requisitos podían cambiar a medida que avanzaba el desarrollo. Utilizando Scrum, con sus sprints cortos y revisiones frecuentes, me permitió ajustar el rumbo del proyecto ágilmente.

2. **Desarrollo Incremental:** Me permitió dividir el trabajo en partes manejables y poder completar incrementos funcionales del software de manera continua. Cada sprint incluía la implementación y prueba de nuevas funcionalidades.

3. **Mejora Continua:** Aunque trabajé de forma independiente, seguí prácticas de Scrum como revisiones de sprint para mantener la organización del proyecto. Adicionalmente, me reunía cada cierto tiempo con mi tutor para poder revisar mi avance.

3.2. Cronograma y Planificación

El proyecto comenzó a mitad de febrero de 2024, con la meta de ser completado a finales de junio del mismo año. Para visualizar la planificación del mismo, se presenta un Diagrama de Gantt. Este diagrama es una representación gráfica del cronograma del proyecto, donde las tareas se describen en el eje vertical y el tiempo empleado en cada una de ellas en el eje horizontal. El diagrama muestra las fechas de inicio y finalización del proyecto, facilitando el seguimiento y la gestión de las distintas etapas del desarrollo.

La aplicación se ha desarrollado utilizando tecnologías como Python y React Native, de las cuales no tenía experiencia previa. Por lo tanto, la planificación del proyecto se realizó teniendo en cuenta el tiempo necesario para aprender y familiarizarme con estos lenguajes. Este proceso incluyó la dedicación del tiempo adicional para la formación y la práctica de ambas tecnologías, asegurando así un desarrollo ágil y eficiente de la aplicación.

3.2.1. Diagrama de Gantt

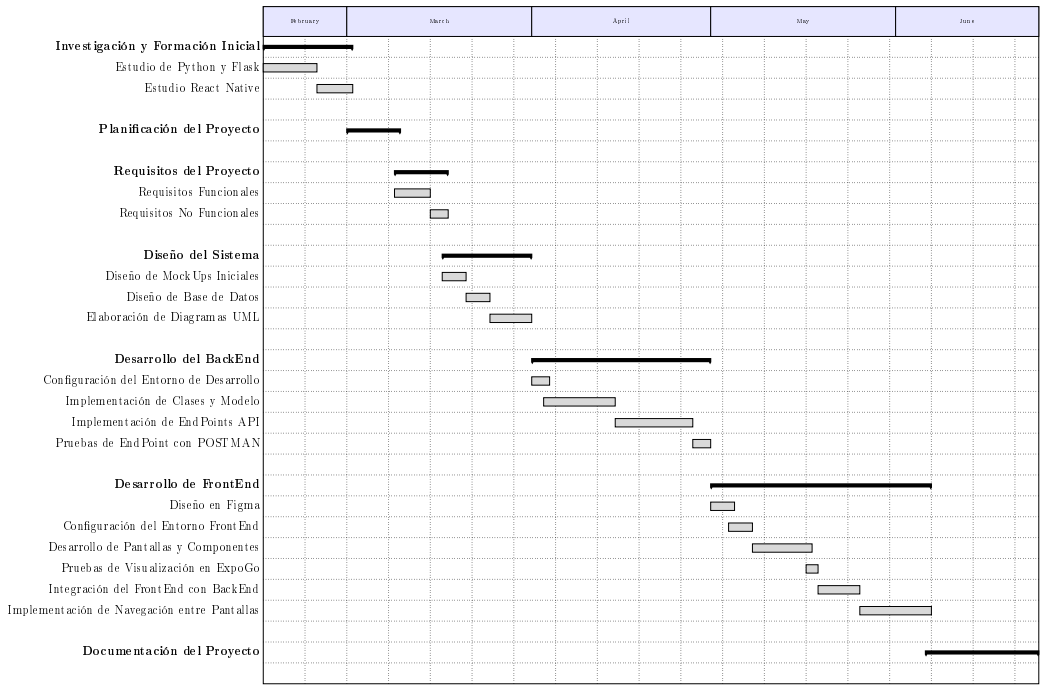


Figura 3.1: Diagrama de Gantt detallado con la planificación y fases de desarrollo

3.3. Recursos y Materiales

En este apartado se analizan los recursos y materiales utilizados durante el transcurso del proyecto.

3.3.1. Recursos Humanos

Para el desarrollo de la aplicación, el principal y único recurso humano he sido yo, ya que he desempeñado los roles de desarrollador y usuario. Además, es importante mencionar a mi tutor, quien me ha guiado a lo largo del proceso, proporcionando orientación y apoyo.

3.3.2. Materiales

Para el desarrollo del proyecto se ha utilizado un portátil personal y un segundo monitor, lo cual ha sido de gran ayuda a la hora de programar, permitiendo una mayor eficiencia y comodidad en la gestión de múltiples ventanas y herramientas. A parte de los materiales utilizados para el desarrollo también se ha utillzado un smartphone para la prueba del frontend utilizando la aplicación Expo Go:

1. **Portátil:** ASUS TUF F15 FX506HC - Portátil de 15.6"Full HD 144Hz (Intel Core i5-11400H, 16GB RAM, 512GB SSD, NVIDIA RTX 3050-4GB). Con valor actual de 699.99 € [6]

2. **Monitor:** BenQ GW2480 - Monitor IPS LED de 23.8 Pulgadas 1080p. Con valor actual de 99.99 € [7]

3. **Smartphone:** Samsung Galaxy S10e - Smartphone de 5.8”, Dual SIM, 128 GB. Con valor actual de 329.99 € [8]

3.4. Presupuesto

En este apartado se analiza y se describen los costes del proyecto utilizando información proporcionada por las bases de cotización para contingencias comunes del año 2024, según la Seguridad Social de España. El perfil será el de un ingeniero informático junior y teniendo en cuenta que las bases de cotización oscilan entre un mínimo de 1.847,40 € y un máximo de 4.720,50 €. Se asumirá el uso de la base mínima de cotización y la duración de 5 meses del proyecto. [9]

Con esta información el coste total del salario para un ingeniero informático junior durante la duración del proyecto sería de

$$\textbf{Salario Informatico Junior} = 1\,847{,}40\ €/\text{mes} \times 5\ \text{meses} = 9\,237 €$$

Si por su contraparte quisiéramos ver el coste total del salario para un ingeniero informático senior durante la duración del proyecto y utilizando el máximo de las bases de cotización mencionadas anteriormente, procederíamos de la siguiente manera:

$$\textbf{Salario Informatico Senior} = 4\,720{,}50\ €/\text{mes} \times 5\ \text{meses} = 23\,602{,}50 €$$

Estas estimaciones no toman costes adicionales más allá de los descritos anteriormente.

En lo que respecta a las licencias para este proyecto, todas las tecnologías utilizadas han sido de código abierto con el fin de minimizar gastos. Como consecuencia a esta elección el coste total en licencias ha sido nulo.

Para estimar el coste del hardware mencionado en el apartado de materiales, es importante considerar la estimación de vida útil de un dispositivo, como puede ser un portátil o un smartphone, que es de 5 años.

$$\textbf{Coste Materiales} = 699{,}99\ € + 329{,}99\ € + 99{,}99\ € = 1\,129{,}97\ €$$

$$\textbf{Coste Anual} = \frac{\text{Coste Material}}{5\ \text{años}} = 225{,}99\ €$$

$$\textbf{Coste Duración Proyecto} = \left(\frac{\text{Coste Anual}}{12\ \text{meses}}\right) \times 5\ \text{meses} = 94{,}16\ €$$

Sumando el coste del hardware al coste de los recursos humanos para un ingeniero informático junior, el coste total del proyecto durante estos 5 meses es de 9 331,16 €. Para un ingeniero informático senior, el coste total del proyecto sería de 23 696,66 €.

Capítulo 4

Análisis

A lo largo de esta sección se irán analizando los requisitos para el desarrollo del proyecto. La aplicación cubre la gestión de usuarios, administradores de establecimiento, establecimientos, actividades, eventos, ofertas y reseñas. Además, se describirán los requisitos no funcionales del sistema para garantizar una experiencia de usuario óptima y del correcto funcionamiento de la aplicación.

4.1. Requisitos Funcionales

Los requisitos funcionales son una parte esencial del desarrollo de sistemas, ya que capturan el comportamiento previo del sistema. Este comportamiento se puede expresar como servicios, tareas o funciones que el sistema debe realizar. Los requisitos funcionales se centran en lo que el sistema hará, describiendo el comportamiento del sistema en términos de las acciones específicas que debe llevar a cabo para cumplir con las necesidades y expectativas del usuario final. [10]

A continuación se describirán los requisitos funcionales, divididos en apartados según su gestión principal. Esto permitirá una comprensión clara y detallada de los procesos y funcionalidades que la aplicación debe incluir para cumplir con sus objetivos y proporcionar una experiencia completa a sus usuarios.

4.1.1. Gestión de Usuarios

Este apartado describe las funcionalidades relativas a los usuarios. Al hablar de "Usuario" se referirá tanto a los usuarios genéricos del sistema como a un administrador de establecimiento. A continuación, se detallarán las funcionalidades para poder satisfacer las necesidades básicas de los usuarios:

RF1 - Registro de Usuario	
Descripción	Permite a un nuevo usuario registrarse en el sistema.
Datos de entrada	Datos del usuario.
Datos de salida	Confirmación de registro exitoso.

Cuadro 4.1: RF1 - Registro de Usuario

RF2 - Inicio de Sesión	
Descripción	Permite a un usuario autenticarse e iniciar sesión en el sistema.
Datos de entrada	Credenciales del usuario.
Datos de salida	Pantalla de inicio de usuario dependiendo si es usuario genérico o administrador de establecimiento.

Cuadro 4.2: RF2 - Inicio de Sesión

RF3 - Consultar Usuario	
Descripción	Permite ver la información del usuario que ha iniciado sesión.
Datos de entrada	Identificador del usuario.
Datos de salida	Datos del usuario.

Cuadro 4.3: RF3 - Consultar Usuario

RF4 - Modificar Usuario	
Descripción	Permite actualizar los datos de un usuario en el sistema.
Datos de entrada	Identificador del usuario, datos actualizados del usuario.
Datos de salida	Confirmación de modificación exitosa.

Cuadro 4.4: RF4 - Modificar Usuario

RF5 - Baja de Usuario	
Descripción	Permite eliminar un usuario del sistema junto con sus datos asociados.
Datos de entrada	Identificador del usuario.
Datos de salida	Confirmación de baja exitosa.

Cuadro 4.5: RF5 - Baja de Usuario

RF6 - Seguir a Usuario	
Descripción	Permite a un usuario seguir a otro usuario en el sistema, añadiéndolo a su lista de seguidos.
Datos de entrada	Identificador del usuario a seguir.
Datos de salida	Confirmación de seguimiento de usuario, actualización de la lista de seguidos.

Cuadro 4.6: RF6 - Seguir a Usuario

RF7 - Dejar de Seguir a Usuario	
Descripción	Permite a un usuario dejar de seguir a otro usuario en el sistema, eliminándolo de su lista de seguidos.
Datos de entrada	Identificador del usuario a dejar de seguir.
Datos de salida	Confirmación de que se ha dejado de seguir a usuario, actualización de su lista de seguidos.

Cuadro 4.7: RF7 - Dejar de Seguir a Usuario

4.1.2. Gestión de Establecimientos

Este apartado describe las funcionalidades relativas a los establecimientos, es decir, qué se puede hacer con ellos. A continuación, se detallarán las principales características y acciones disponibles para la gestión de los establecimientos dentro de la aplicación:

RF8 - Crear Establecimiento	
Descripción	Permite a un administrador crear un nuevo establecimiento.
Datos de entrada	Datos del establecimiento.
Datos de salida	Confirmación de la creación del establecimiento, actualización de la lista de establecimientos del administrador.

Cuadro 4.8: RF8 - Crear Establecimiento

RF9 - Modificar Establecimiento	
Descripción	Permite a un administrador actualizar los datos de un establecimiento.
Datos de entrada	Identificador del establecimiento, datos actualizados del establecimiento.
Datos de salida	Confirmación de modificación exitosa.

Cuadro 4.9: RF9 - Modificar Establecimiento

RF10 - Consultar Establecimiento	
Descripción	Permite a un usuario o administrador consultar los datos de un establecimiento en específico.
Datos de entrada	Identificador del establecimiento.
Datos de salida	Datos del establecimiento.

Cuadro 4.10: RF10 - Consultar Establecimiento

RF11 - Eliminar Establecimiento	
Descripción	Permite a un administrador eliminar un establecimiento.
Datos de entrada	Identificador del establecimiento.
Datos de salida	Confirmación de eliminación exitosa, actualización de la lista del administrador.

Cuadro 4.11: RF11 - Eliminar Establecimiento

RF12 - Filtrar Establecimientos	
Descripción	Permite a un usuario poder filtrar los establecimientos según las preferencias indicadas.
Datos de entrada	Filtro aplicado.
Datos de salida	Establecimientos que cumplan con el filtro.

Cuadro 4.12: RF12 - Filtrar Establecimiento

4.1.3. Gestión de Eventos

Este apartado describe la gestión de los eventos asociados a un establecimiento específico. Los eventos son importantes porque permiten a los establecimientos promocionar sus servicios, ayudando a los usuarios a decidir cuál se ajusta según sus necesidades. A continuación, se detallarán las principales acciones disponibles para la gestión de eventos:

RF13 - Crear Evento	
Descripción	Permite a un administrador crear un nuevo evento asociado a un establecimiento.
Datos de entrada	Datos del evento
Datos de salida	Confirmación de creación del evento, actualización de la lista eventos del establecimiento en el cual se ha creado el evento

Cuadro 4.13: RF13 - Crear Evento

RF14 - Modificar Evento	
Descripción	Permite a un administrador de establecimiento modificar un nuevo evento existente.
Datos de entrada	Identificador del evento, datos actualizados del evento.
Datos de salida	Confirmación de modificación exitosa.

Cuadro 4.14: RF14 - Modificar Evento

RF15 - Consultar Evento	
Descripción	Permite a un usuario o administrador ver los datos de un evento en específico.
Datos de entrada	Identificador del evento.
Datos de salida	Datos del evento.

Cuadro 4.15: RF15 - Consultar Evento

RF16 - Eliminar Evento	
Descripción	Permite a un administrador eliminar un evento existente.
Datos de entrada	Identificador del evento.
Datos de salida	Confirmación de eliminación exitosa, actualización de la lista de eventos del establecimiento que contenía ese evento.

Cuadro 4.16: RF16 - Eliminar Evento

4.1.4. Gestión de Ofertas

Este apartado describe la gestión de las ofertas asociadas a un establecimiento específico. Las ofertas son importantes porque permiten a los establecimientos promocionar sus servicios, ayudando a los usuarios a decidir cuál les conviene más entre los disponibles. A continuación, se detallarán las principales acciones disponibles para la gestión de ofertas:

RF17 - Crear Oferta	
Descripción	Permiet a un administrador crear una nueva oferta asociada a un establecimiento.
Datos de entrada	Datos de la oferta.
Datos de salida	Confirmación de creación exitosa, actualizacion de la lista de ofertas del establecimiento en el cual se ha creado esa oferta.

Cuadro 4.17: RF17 - Crear Oferta

RF18 - Modificar Oferta	
Descripción	Permite a un administrador modificar una oferta existente.
Datos de entrada	Identificador de la oferta, datos actualizados de la oferta.
Datos de salida	Confirmación de modificación exitosa.

Cuadro 4.18: RF18 - Modificar Oferta

RF19 - Consultar Oferta	
Descripción	Permite a un usuario o administrador ver los datos de una oferta en específica.
Datos de entrada	Identificador de la oferta.
Datos de salida	Datos de la oferta.

Cuadro 4.19: RF19 - Consultar Oferta

RF20 - Eliminar Oferta	
Descripción	Permite a un administrador eliminar una oferta existente.
Datos de entrada	Identificador de la oferta.
Datos de salida	Confirmación de eliminación exitosa, actualización de la lista de ofertas del establecimiento que contenía esa oferta.

Cuadro 4.20: RF20 - Eliminar Oferta

4.1.5. Gestión de Actividades

Este apartado describe la gestión de las actividades creadas por los usuarios para su grupo social cercano, con el fin de organizar y planificar salidas. A continuación, se detallan las principales acciones disponibles para la gestión de actividades:

RF21 - Crear Actividad	
Descripción	Permite a un usuario crear una nueva actividad.
Datos de entrada	Datos de la actividad.
Datos de salida	Confirmación de creación de la actividad, actualización de la lista de actividades creadas del usuario.

Cuadro 4.21: RF21 - Crear Actividad

RF22 - Modificar Actividad	
Descripción	Permite a un usuario modificar una actividad creada.
Datos de entrada	Identificador de la actividad, datos actualizados de la actividad.
Datos de salida	Confirmación de modificación exitosa.

Cuadro 4.22: RF2 - Modificar Actividad

RF23 - Consultar Actividad	
Descripción	Permite a un usuario poder consultar los datos de una actividad en la cual este participe.
Datos de entrada	Identificador de la actividad.
Datos de salida	Datos de la actividad.

Cuadro 4.23: RF23 - Consultar Actividad

RF24 - Eliminar Actividad	
Descripción	Permite a un usuario eliminar una actividad creada.
Datos de entrada	Identificador de la actividad.
Datos de salida	Confirmación de eliminación exitosa, actualización de la lista de actividades creadas del usuario.

Cuadro 4.24: RF24 - Eliminar Actividad

RF25 - Invitar Usuario a Actividad	
Descripción	Permite a un usuario invitar a otros usuarios a participar en una actividad.
Datos de entrada	Identificador del usuario a invitar.
Datos de salida	Confirmación de adición de un nuevo usuario a la actividad, actualización de la lista de participantes de la actividad.

Cuadro 4.25: RF25 - Invitar Usuario a Actividad

4.1.6. Gestión de Reseñas

Este apartado describe la gestión de las reseñas a un establecimiento. Las reseñas son cruciales para mostrar a los usuarios los establecimientos según

las evaluaciones y comentarios de otros clientes. Además, son importantes para los administradores de establecimientos, ya que pueden utilizar el feedback para mejorar sus servicios. A continuación, se detallan las principales acciones para la gestión de las reseñas:

RF26 - Crear Reseña	
Descripción	Permite a un usuario dejar reseñas a un establecimiento.
Datos de entrada	dentificador del establecimiento, datos de la reseña.
Datos de salida	Confirmación de que la reseña ha sido publicada, actualización de la lista de reseñas del establecimiento, actualización de las listas de reseñas creadas del usuario.

Cuadro 4.26: RF26 - Crear Reseña

RF27 - Consultar Reseña	
Descripción	Permite a un usuario o administrador ver las reseñas de un establecimiento específico.
Datos de entrada	Identificador del establecimiento.
Datos de salida	Reseñas del establecimiento.

Cuadro 4.27: RF27 - Consultar Reseña

4.2. Requisitos No Funcionales

Los requisitos no funcionales son aquellos que especifican los criterios que pueden ser utilizados para juzgar el funcionamiento de un sistema, en lugar de describir los comportamientos específicos. A diferencia de los requisitos funcionales, que detallan lo que el sistema debe hacer, los requisitos no funcionales definen cómo debe ser el sistema en términos de calidad y restricciones operativas, asegurando que el sistema cumpla con los estándares de calidad esperados. [11]

RNF1 - Usabilidad	
Descripción	La aplicación será intuitiva y fácil de usar para los usuarios.
Criterios	Interfaz de usuario sencilla e intuitiva.

Cuadro 4.28: RNF1 - Usabilidad

RNF2 - Rendimiento	
Descripción	La aplicación deberá responder rápidamente a las consultas realizadas.
Criterios	Tiempos de respuestas rápidos para las peticiones realizadas.

Cuadro 4.29: RNF2 - Rendimiento

RNF3 - Seguridad	
Descripción	La aplicación protegerá la información sensible relacionada con los datos personales de los usuarios.
Criterios	Los datos sensibles de los usuarios solo podrán ser consultados por ellos mismos.

Cuadro 4.30: RNF3 - Seguridad

RNF4 - Compatibilidad	
Descripción	La aplicación funcionará correctamente independientemente del dispositivo móvil utilizado siempre y cuando sea un smartphone.
Criterios	El diseño del frontend en React Native permite poder desplegar el archivo APK para Android y el archivo para iOS.

Cuadro 4.31: RNF4 - Compatibilidad

RNF5 - Escalabilidad	
Descripción	La aplicación debe ser capaz de escalar y manejar el aumento de la carga manteniendo el nivel de rendimiento.
Criterios	El sistema permite el crecimiento de carga según aumente el número de usuarios gracias a su arquitectura.

Cuadro 4.32: RNF5 - Escalabilidad

RNF6 - Mantenibilidad	
Descripción	La aplicación debe poder proveer actualizaciones y mantenimiento regulable para su continuo desarrollo.
Criterios	La documentación se realizará de una forma clara permitiendo el entendimiento del código y podiendo realizar actualizaciones para el beneficio de la experiencia de los usuarios.

Cuadro 4.33: RNF6 - Mantenibilidad

4.3. Modelos de Caso de Uso

Los modelos de caso de uso son representaciones gráficas que describen cómo los usuarios interactúan con el sistema para lograr un objetivo específico. Estos modelos se utilizan para capturar y definir los requisitos funcionales del sistema., identificando las diferentes formas en que los usuarios pueden utilizar el sistema y las respuestas del sistema a estas interacciones. Los modelos de casos de uso ayudan a comunicar claramente las funcionalidades esperadas del sistema a todas las partes interesadas, asegurando que todos tengan una comprensión común de los requerimientos. [12]

4.3.1. Actores del Sistema

La interacción de la aplicación se basa en dos tipos de actores: usuarios genéricos y administradores de establecimiento. Es importante destacar que el acceso a la aplicación está restringida a usuarios registrados, ya que se busca asegurar que cada interacción dentro del sistema se realice de una manera personalizada.

1. **Usuario Genérico:** Los usuarios genéricos son los clientes de la aplicación, aquellos para los cuales está destinado su uso. Pueden consultar establecimientos según las preferencias indicadas, ver las ofertas y eventos disponibles, y crear actividades sociales. Estas funcionalidades permiten a los usuarios personalizar su experiencia y aprovechar al máximo los servicios ofrecidos por la aplicación.

2. **Administrador de Establecimiento:** Los administradores de establecimientos interactúan con la aplicación para la creación, administración y promoción de sus establecimientos. Esto incluye la creación de ofertas y eventos, así como la gestión general de la información del

establecimiento. Estas funcionalidades permiten a los administradores optimizar la visibilidad y el atractivo de sus establecimientos dentro de la aplicación.

4.3.2. Escenarios de Casos de Uso

CU1 - Registrar Usuario	
Actores	Usuario Genérico y Administrador de Establecimiento.
Precondiciones	El actor no debe estar registrado en el sistema.
Postcondiciones	El actor está registrado en el sistema.
Flujo principal	1. El actor accede a la pantalla de registro. 2. El actor ingresa sus datos. 3. El sistema valida sus datos. 4. El sistema registra al actor y confirma su registro.

Cuadro 4.34: CU-1 Registrar Usuario

CU2 - Autenticar Usuario	
Actores	Usuario Genérico y Administrador de Establecimiento
Precondiciones	El actor debe estar registrado en el sistema
Postcondiciones	El usuario está autenticado en el sistema
Flujo principal	1. El usuario accede a la pantalla de inicio de sesión. 2. El usuario ingresa sus credenciales 3. El sistema valida sus credenciales 4. El sistema permite el acceso al usuario y confirma la autenticación

Cuadro 4.35: CU2 - Autenticar Usuario

CU3 - Gestionar Usuario	
Actores	Usuario Genérico y Administrador de Establecimiento
Precondiciones	El actor debe haber iniciado sesión en el sistema
Postcondiciones	Los datos del usuario son gestionados
Flujo principal	1. El usuario accede a la pantalla de su perfil 2. El usuario puede elegir entre: *a*) Consultar sus datos (Consultar Datos de Usuario) *b*) Modificar sus datos (Modificar Usuario) *c*) Eliminar su cuenta (Baja de Usuario) 3. El sistema valida y confirma sus acciones

Cuadro 4.36: CU3 - Gestionar Usuario

CU4 - Consultar Establecimiento	
Actores	Usuario Genérico y Administrador de Establecimiento
Precondiciones	El actor debe haber iniciado sesión en el sistema
Postcondiciones	Se mostrarán los datos de el establecimiento consultado
Flujo principal	1. El actor accede a la pantalla principal del sistema 2. El actor navega hasta el establecimiento de interés 3. El actor selecciona el establecimiento 4. El sistema muestra los datos del establecimiento seleccionado

Cuadro 4.37: CU4 - Consultar Establecimiento

CU5 - Consultar Reseñas	
Actores	Usuario Genérico y Administrador de Establecimiento
Precondiciones	El actor debe haber iniciado sesión en el sistema
Postcondiciones	Se mostrarán los datos de las reseñas asociadas al establecimiento consultado
Flujo principal	1. El actor accede a la pantalla principal del sistema 2. El actor navega hasta el establecimiento de interés 3. El actor selecciona el establecimiento 4. El sistema muestra las reseñas asociadas a ese establecimiento

Cuadro 4.38: CU5 - Consultar Reseñas

CU6 -	
Actores	Usuario Genérico y Administrador de Establecimiento
Precondiciones	El actor debe haber iniciado sesión en el sistema
Postcondiciones	Se mostrarán los datos del evento asociado al establecimiento consultado
Flujo principal	1. El actor accede a la pantalla principal del sistema 2. El actor navega hasta el establecimiento de interés 3. El actor selecciona el establecimiento 4. El actor selecciona un evento dentro de ese establecimiento 5. El sistema muestra los datos del evento seleccionado.

Cuadro 4.39: CU6 - Consultar Eventos

CU7 - Consultar Ofertas	
Actores	Usuario Genérico y Administrador de Establecimiento
Precondiciones	El actor debe haber iniciado sesión en el sistema
Postcondiciones	
Flujo principal	1. El actor accede a la pantalla principal del sistema 2. El actor navega hasta el establecimiento de interés 3. El actor selecciona el establecimiento 4. El actor selecciona una oferta dentro de ese establecimiento 5. El sistema muestra los datos de la oferta seleccionada

Cuadro 4.40: CU7 - Consultar Ofertas

CU8 - Crear Establecimiento	
Actores	Administrador de Establecimiento
Precondiciones	El administrador de establecimiento debe haber iniciado sesión en el sistema
Postcondiciones	Confirmación de creación de establecimiento, se actualiza la lista de establecimientos del administrador.
Flujo principal	1. El administrador accede a su página principal. 2. El administrador selecciona la opción de Crear Establecimiento. 3. El administrador proporciona los datos del nuevo establecimiento. 4. El sistema valida los datos ingresados. 5. El sistema registra y confirma el nuevo establecimiento.

Cuadro 4.41: CU8 - Crear Establecimiento

CU9 - Gestionar Establecimiento	
Actores	Administrador de Establecimiento
Precondiciones	El administrador de establecimiento debe haber iniciado sesión en el sistema
Postcondiciones	Los datos del establecimiento han sido adecuadamente gestionados
Flujo principal	1. El administrador accede a su página principal. 2. El administrador selecciona un establecimiento entre los que este gestiona. 3. El administrador puede elegir entre: *a*) Modificar un establecimiento (Modificar Establecimiento) *b*) Eliminar un Establecimiento (Eliminar Establecimiento) 4. El sistema valida y confirma las acciones

Cuadro 4.42: CU9 - Gestionar Establecimiento

CU10 - Crear Evento	
Actores	Administrador de Establecimiento
Precondiciones	El administrador de establecimiento debe haber iniciado sesión en el sistema
Postcondiciones	Confirmación de creación de evento, se actualiza la lista de eventos del establecimiento.
Flujo principal	1. El administrador accede a su página principal 2. El administrador selecciona un establecimiento entre los que este gestiona. 3. El administrador selecciona la opción de Crear Evento simbolizada con el signo + al lado de la etiqueta Eventos. 4. El administrador proporciona los datos del nuevo evento. 5. El sistema valida los datos ingresados. 6. El sistema registra y confirma el nuevo evento

Cuadro 4.43: CU10 - Crear Evento

CU11 - Gestionar Evento	
Actores	Administrador de Establecimiento
Precondiciones	El administrador de establecimiento debe haber iniciado sesión en el sistema
Postcondiciones	Los datos del evento han sido adecuadamente gestionados
Flujo principal	1. El administrador accede a su página principal. 2. El administrador selecciona un establecimiento entre los que este gestiona. 3. El administrador selecciona un evento de los asociados a este establecimiento. 4. El administrador puede elegir entre: *a*) Modificar un evento (Modificar Evento) *b*) Eliminar un evento (Eliminar Evento) 5. El sistema valida y confirma las acciones

Cuadro 4.44: CU11 - Gestionar Evento

CU12 - Crear Oferta	
Actores	Administrador de Establecimiento
Precondiciones	El administrador de establecimiento debe haber iniciado sesión en el sistema
Postcondiciones	Confirmación de creación de oferta, se actualiza la lista de ofertas del establecimiento.
Flujo principal	1. El administrador accede a su página principal 2. El administrador selecciona un establecimiento entre los que este gestiona. 3. El administrador selecciona la opción de Crear Oferta simbolizada con el signo + al lado de la etiqueta Ofertas. 4. El administrador proporciona los datos de la nueva oferta. 5. El sistema valida los datos ingresados. 6. El sistema registra y confirma la nueva oferta

Cuadro 4.45: CU12 - Crear Oferta

CU13 - Gestionar Oferta	
Actores	Administrador de Establecimiento
Precondiciones	El administrador de establecimiento debe haber iniciado sesión en el sistema
Postcondiciones	Los datos de la oferta han sido adecuadamente gestionados
Flujo principal	1. El administrador accede a su página principal. 2. El administrador selecciona un establecimiento entre los que este gestiona. 3. El administrador selecciona una oferta de las asociadas a este establecimiento. 4. El administrador puede elegir entre: *a*) Modificar una oferta(Modificar Oferta) *b*) Eliminar una oferta(Eliminar Oferta) 5. El sistema valida y confirma las acciones

Cuadro 4.46: CU13 - Gestionar Oferta

CU14 - Crear Actividad	
Actores	Usuario Genérico
Precondiciones	El usuario debe haber iniciado sesión en el sistema
Postcondiciones	Confirmación de creación de actividad, se actualiza la lista de actividades creadas del usuario.
Flujo principal	1. El usuario accede a su página principal. 2. El usuario selecciona la opción de **Crear Actividad** simbolizada con el signo + en el footer de la aplicación. 3. El usuario proporciona los datos de la nueva actividad. 4. El sistema valida los datos ingresados. 5. El sistema confirma y registra la nueva actividad.

Cuadro 4.47: CU14 - Crear Actividad

CU15 - Gestionar Actividad	
Actores	Usuario Genérico
Precondiciones	El usuario debe haber iniciado sesión en el sistema
Postcondiciones	Los datos de la actividad han sido correctamente gestionado
Flujo principal	1. El usuario accede a su página principal. 2. El usuario navega hasta una actividad y la selecciona. 3. El usuario puede elegir entre: *a*) Modificar una actividad(Modificar Actividad) *b*) Eliminar una actividad(Eliminar Actividad) 4. El sistema valida y confirma las acciones

Cuadro 4.48: CU15 - Gestionar Actividad

CU16 - Crear Reseña	
Actores	Usuario Genérico
Precondiciones	El usuario debe haber iniciado sesión en el sistema
Postcondiciones	Confirmación de creación de reseña, se actualiza la lista de reseñas creadas por parte del usuario y se actualiza la lista reseñas del establecimiento
Flujo principal	1. El usuario accede a su página principal. 2. El usuario navega hasta un establecimiento y lo selecciona 3. El usuario selecciona la opción de Crear Reseña simbolizada con el signo + al final de la pantalla del establecimiento. 4. El usuario rellena la reseña asignándole una calificación al establecimiento. 5. El sistema valida los datos proporcionados para la creación de la reseña . 6. El sistema confirma y registra la nueva reseña

Cuadro 4.49: CU16 - Crear Reseña

CU17 - Filtrar Establecimientos	
Actores	Usuario Genérico
Precondiciones	El usuario debe haber iniciado sesión en el sistema
Postcondiciones	Se mostrarán los establecimientos que cumplan con el filtro seleciconado por el usuario
Flujo principal	1. El usuario accede a su página principal. 2. El usuario selecciona las preferencias. 3. El sistema valida las preferencias y muestra los establecimientos que cumplan con ese filtro.

Cuadro 4.50: CU17 - Filtrar Establecimientos

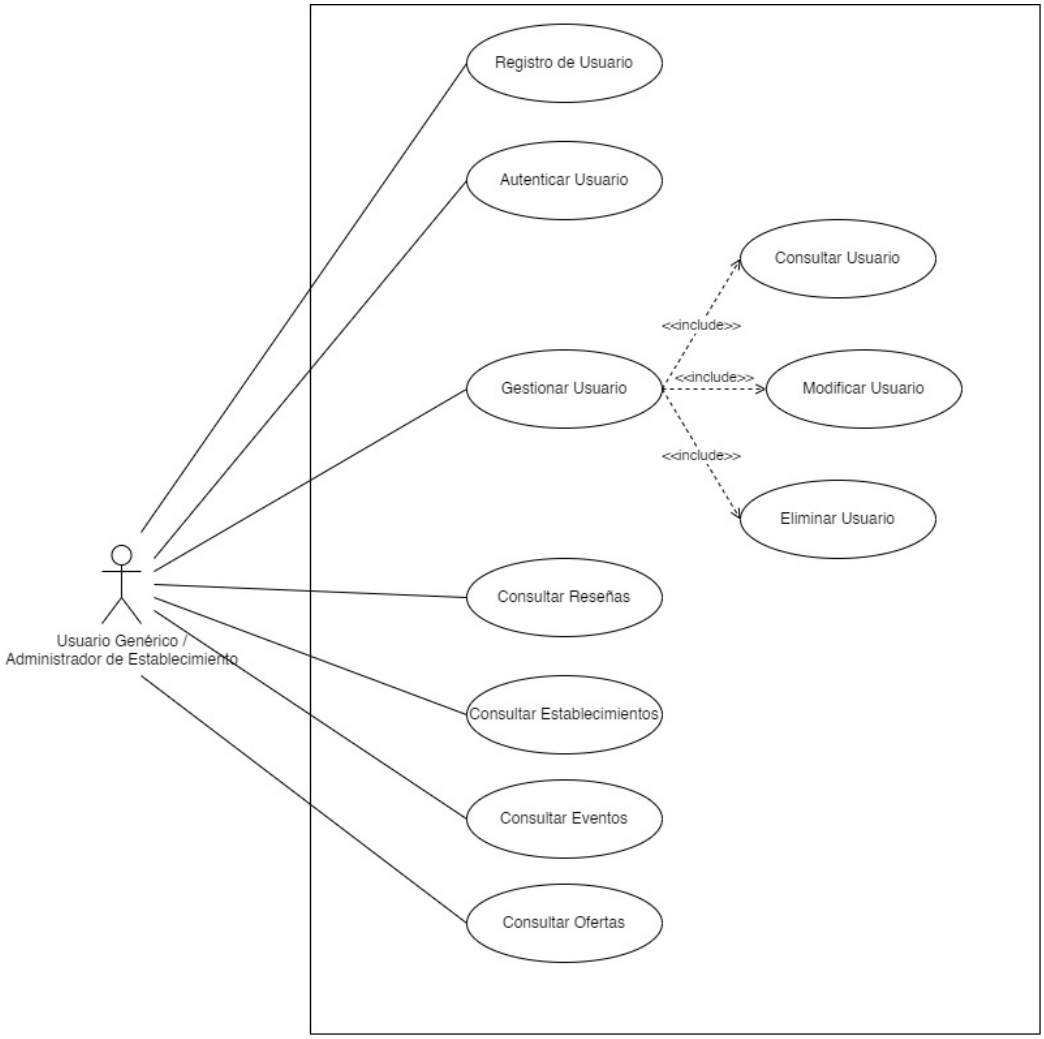


Figura 4.1: Caso de Uso con Actores Comunes

Figura 4.2: Caso de Uso con Administrador de Establecimiento

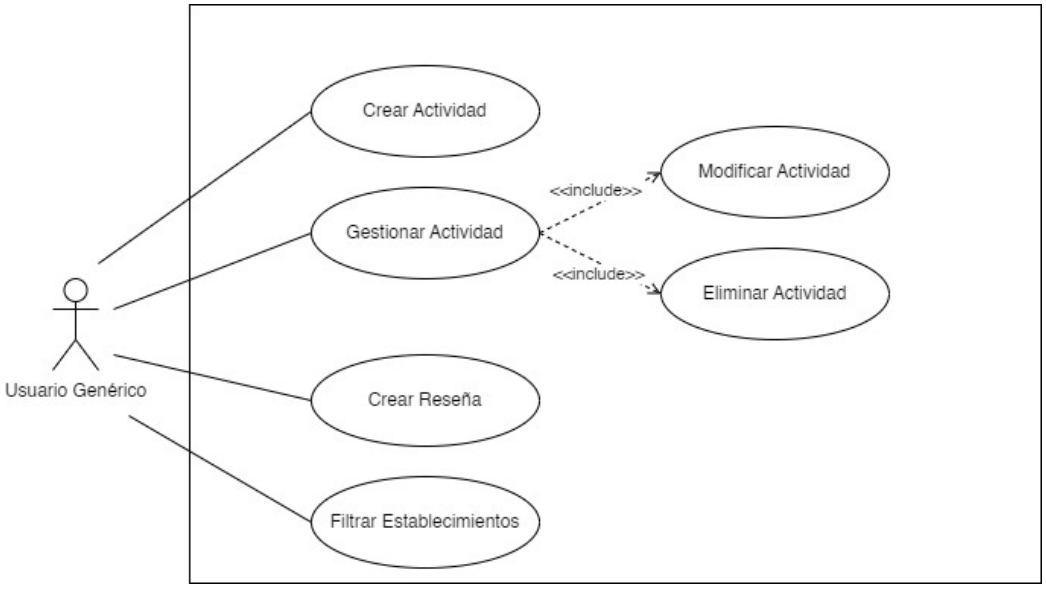


Figura 4.3: Caso de Uso con Usuario Genérico

Capítulo 5

Diseño

5.1. Arquitectura Software

La arquitectura software se refiere a la estructura o estructuras de un sistema software, que comprenden componentes de software, sus propiedades visibles externamente y las relaciones entre ellos. Es una disciplina esencial en la ingeniería de software que aborda la organización de un sistema como una composición de componentes, los protocolos de comunicación y sincronización, y la asignación de funciones a elementos de diseño. Una buena arquitectura de software garantiza que un sistema cumpla con requisitos clave como rendimiento, fiabilidad, portabilidad, escalabilidad e interoperabilidad, y sirve como un puente crítico entre los requisitos del sistema y el código implementado. [13]

Para el desarrollo de la aplicación móvil, se ha optado por una arquitectura cliente-servidor basada en una arquitectura orientada a servicios (SOA) debido a su capacidad para facilitar la comunicación entre el servidor y la aplicación móvil. La API, desarrollada en Python y Flask, maneja las solicitudes del cliente. Los usuarios realizan peticiones desde la aplicación móvil al servidor, y este responde con la información solicitada, Esta arquitectura permite una interacción eficiente entre el cliente y el servidor, asegurando una experiencia de usuario fluida.

5.1.1. Arquitectura Orientada a Servicios

La Arquitectura Orientada a Servicios (SOA) es un paradigma que organiza capacidades distribuidas en la red, haciendo visibles los servicios a los usuarios y permitiendo la interacción mediante intercambios de información para producir efectos reales. La llamada a servicios se basa en lenguajes y protocolos estándar, facilitando interoperabilidad y presentando las operaciones como atómicas desde la perspectiva del usuario. SOA permite soluciones flexibles, escalables y eficientes, liberando a los usuarios para concentrarse en la usabilidad sin preocuparse por los detalles técnicos internos. [14]

En el contexto de una arquitectura cliente-servidor, las llamadas a los servicios se realizan mediante llamadas HTTP, y la información se devuelve en formato JSON, que es el estándar para la intercomunicación en esta arquitectura. Este enfoque asegura una comunicación clara y estructurada entre el cliente y el servidor, permitiendo un manejo eficiente de los datos.

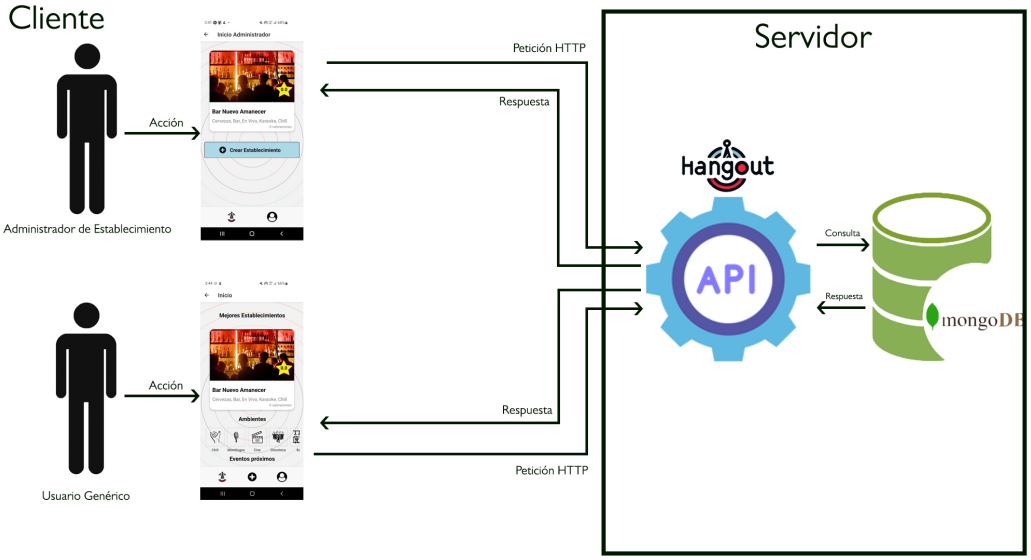


Figura 5.1: Arquitectura Orientada a Servicios basada en Cliente-Servidor

5.2. Tecnologías Utilizadas

Para el desarrollo de la aplicación móvil, se necesita tanto un backend como un frontend. El backend funciona como un servidor al que se hacen las peticiones, implementado mediante una API en Python con Flask y utilizando MongoDB como base de datos. El frontend incluye la interfaz de usuario,

desarrollada en React Native, desde donde los usuarios interactúan y realizan las llamadas a la API. En las secciones siguientes se detallarán estas implementaciones con mayor profundidad.

5.2.1. Desarrollo Backend

Python 3.12.3: Es un lenguaje de programación interpretado, orientado a objetos y de alto nivel con semántica dinámica. Sus estructuras de datos de alto nivel incorporadas, combinadas con la tipificación dinámica y el enlace dinámico, lo hacen muy atractivo para el desarrollo rápido de aplicaciones, así como para su uso como lenguaje de script o "pegamento"para conectar componentes existentes. La sintaxis simple y fácil de aprender de Python enfatiza la legibilidad y, por lo tanto, reduce el costo de mantenimiento del programa. Python soporta módulos y paquetes, lo que fomenta la modularidad del programa y la reutilización del código. [15]

La elección de utilizar Python para el desarrollo del backend fue una recomendación de mi tutor. Además, aprender uno de los lenguajes de programación más utilizados a nivel mundial fue una motivación adicional. Python es reconocido por su facilidad de aprendizaje y su sintaxis clara y directa, esto lo convierte en una opción ideal si ya se tiene experiencia en otros lenguajes. En mi caso, el conocimiento previo de Ruby, que posee algunas similitudes con Python, hizo que la transición y el aprendizaje resultara más sencillo.

Flask 3.0.2: Es un microframework para Python que facilita la creación de las API. Fue diseñado para ser sencillo y fácil de usar, permitiendo a los desarrolladores construir aplicaciones rápidamente con una mínima configuración. Flask es conocido por su flexibilidad y modularidad, lo que permite a los desarrolladores elegir las herramientas y bibliotecas que mejor se adapten a sus necesidades. [16]

El uso de Flask, además de ser una recomendación de mi tutor, se debió a la excelente documentación disponible sobre este framework. Las guías detallan desde la configuración inicial hasta los aspectos que pueden resultar más complejos, proporcionando indicaciones y ejemplos. Necesitaba un framework que además de tener una curva de aprendizaje sencilla, facilitara la conexión a bases de datos, y con la biblioteca `flask_pymongo` hizo que todo resultara más simple.

MongoDB 7.0.7: MongoDB es una base de datos de documentos que proporciona escalabilidad y flexibilidad, permitiendo almacenar datos en do-

cumentos similares a JSON. Esto facilita la variación y evolución de la estructura de datos con el tiempo. Ofrece consultas ad hoc, indexación avanzada y agregación en tiempo real. MongoDB está diseñado como una base de datos distribuida, asegurando alta disponibilidad y escalabilidad horizontal, siendo adecuado para aplicaciones modernas que requieren flexibilidad y rendimiento. [17]

El empleo de MongoDB se debe a las razones mencionadas anteriormente, ya que ofrece la biblioteca `flask_pymongo` que facilita la comunicación entre el framework y la base de datos. Además, como la descripción de MongoDB indica este permite el almacenamiento de datos en documentos similares a JSON, lo cual es lo ideal para las necesidades de la aplicación, ya que se buscaba una forma flexible y eficiente de gestionar la información.

5.2.2. Desarrollo Frontend

Node.js 20.12.2: Es un entorno de ejecución de JavaScript asíncrono y basado en eventos, diseñado para crear aplicaciones de red escalables. Permite manejar múltiples conexiones simultáneas sin bloquear el proceso, lo que lo hace eficiente y adecuado para el desarrollo de sistemas escalables. Node.js toma el modelo de eventos un poco más allá, presentando un bucle de eventos como una construcción de tiempo de ejecución en lugar de una biblioteca. [18]

El uso de Node.js es un requisito para utilizar React Native con Expo debido que este proporciona el entorno de ejecución necesario para las herramientas de desarrollo de JavaScript

Java Development Kit 17.0.10: Es un paquete de software necesario para el desarrollo de aplicaciones en Java. Incluye el JRE (Java Runtime Environment), que permite ejecutar programas Java. [19]

Este es otro requerimiento para el uso de React Native con Expo debido a que proporciona las herramientas necesarias para la compilación y ejecución del código para aplicaciones Android ya que estas se construyen con Java en su parte nativa.

React Native 0.74.1: React Native es un framework de desarrollo de aplicaciones móviles que permite crear aplicaciones nativas para Android y iOS y otras plataformas utilizando React y JavaScript. Desarrollado por Facebook, React Native combina lo mejor de React con las capacidades nativas, permitiendo a los desarrolladores escribir una vez y ejecutar en cualquier lu-

gar. Hace uso de componentes nativos como puede ser View, Text e Image que se traducen en bloques de construcción de la interfaz de usuario nativa de la plataforma, proporcionando un rendimiento y experiencia de usuario nativos. Esto permite crear el FrontEnd y desplegarlo tanto en la App Store de Apple como en la Play Store de Google. De allí su lema "Learn Once Write Anywhere" [20]

La utilización de React Native se basa principalmente en que consideré que sería la opción segura y sencilla para el desarrollo de una aplicación móvil, permitiendo escribir el código en un único formato y generar tanto el APK para Android como el archivo para iOS. La utilización de los componentes anteriormente mencionados facilita la lectura y mejora la experiencia de desarrollo. Además, React Native permite una alta reutilización de código gracias a la creación de componentes propios y reutilizables.

Un aspecto importante en mi decisión fue la disponibilidad de Expo, una herramienta que simplifica la configuración inicial, especialmente para aquellos que nunca han trabajado con React o React Native, permitiendo visualizar los cambios en tiempo real en una aplicación móvil.

5.2.3. Herramientas de Desarrollo

Visual Studio Code 1.90.0: Es un editor de código fuente ligero pero potente, desarrollado por Microsoft. Está disponible para Windows, macOS y Linux, y se utiliza para escribir, depurar y editar código. Ofrece características como resaltado de sintaxis, autocompletado de código, integración con Git, y un grupo de extensiones que permite personalizar y ampliar su funcionalidad para soportar una amplia gama de lenguajes de programación y flujos de trabajo de desarrollo. [21]

La principal razón de la elección de Visual Studio Code ha sido mi familiaridad con él, ya que lo he estado utilizando desde los inicios del grado. Sus extensiones facilitan en gran medida la programación. Además, como mencioné en la descripción, la funcionalidad de autocompletado lo convierte en una de las mejores opciones. También permite integrar terminales para ejecutar los comandos desde el mismo entorno, lo cual mejora la eficiencia a la hora de trabajar.

Android Studio 2023.3.1: Es el entorno de desarrollo integrado oficial para el desarrollo de aplicaciones Android. Ofrece herramientas avanzadas de edición de código, depuración, pruebas y rendimiento, y está diseñado para

acelerar el desarrollo y la calidad de las aplicaciones Android. [22]

Solo ha sido necesario utilizar Android Studio en el proyecto para ejecutar el emulador de Android y realizar pruebas en él, además de las pruebas que ya se realizaban en la aplicación Expo Go desde mi propio móvil.

Postman 10.24: Postman es una plataforma de colaboración para el desarrollo de APIs que simplifica cada etapa del ciclo de vida de las API, desde el diseño y la prueba hasta la implementación y el monitoreo. Permite a los desarrolladores enviar solicitudes HTTP, crear y ejecutar pruebas automatizadas, y documentar la API de manera más eficiente. [23]

Ha sido uno de los pilares fundamentales para el desarrollo de mi aplicación móvil especialmente en el backend ya que desde esta aplicación podía comprobar el correcto funcionamiento de los endpoints comprobando si al envíar una solicitud se devolvía una respuesta esperada. También ha sido crucial para el desarrollo del frontend, ya que desde esta herramienta he podido depurar errores y entender el porqué de las respuestas API no no devolvían lo esperado. Esto facilitó la identificación y corrección de problemas en las solicitudes y respuestas HTTP durante el desarrollo.

5.3. Modelo de Datos

Para la base de datos se ha utilizado MongoDB. Su estructura NoSQL basada en documentos JSON facilita la comunicación entre el cliente y el servidor, haciendo que el manejo de datos sea más flexible y eficiente. Dado que el diagrama de clases muestra todas las clases, cada una de estas se traduce en una colección, donde sus campos son los atributos de clase. Esto se verá con mayor claridad en el Diagrama de Clases, proporcionando una visión estructurada y comprensible de la organización de los datos.

5.4. Diseño Lógico

Los diagramas de clase UML modelan la estructura estática de un dominio de aplicación en términos de clases y relaciones entre ellas. Estos diagramas son esenciales en el diseño lógico software, ya que permiten visualizar y documentar entidades y sus interacciones, asegurando consistencia y cumplimiento de los requisitos. [24]

Para definir la elaboración del diseño lógico del sistema se ha optado por

crear el siguiente diagrama de clases

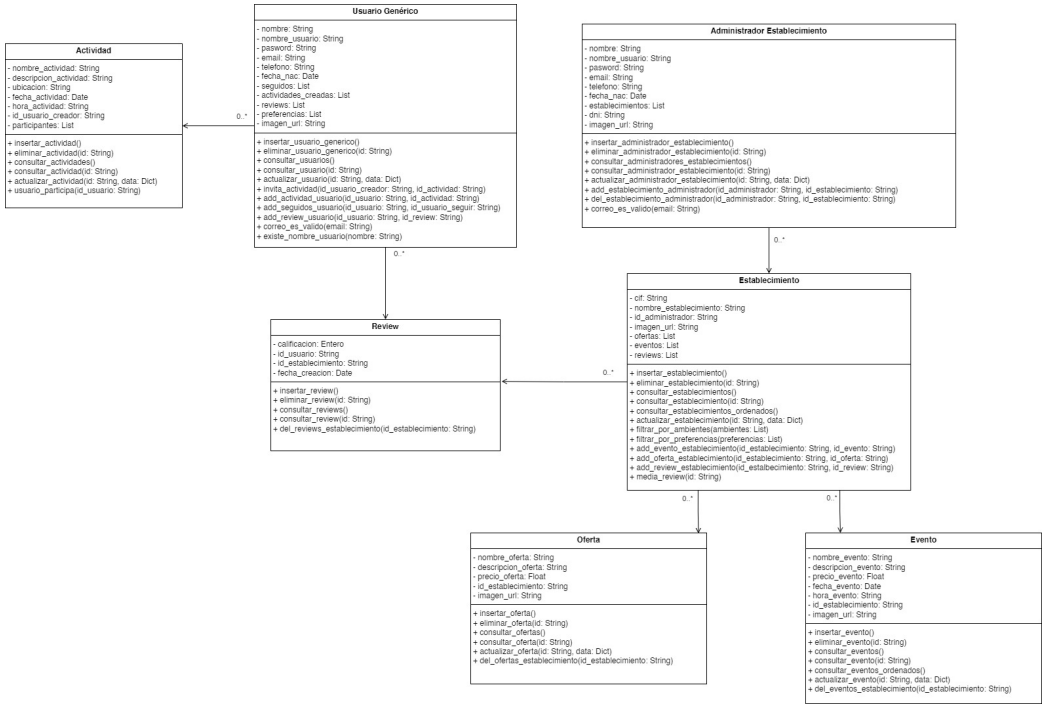


Figura 5.2: Diagrama de clases de la aplicación desarrollada

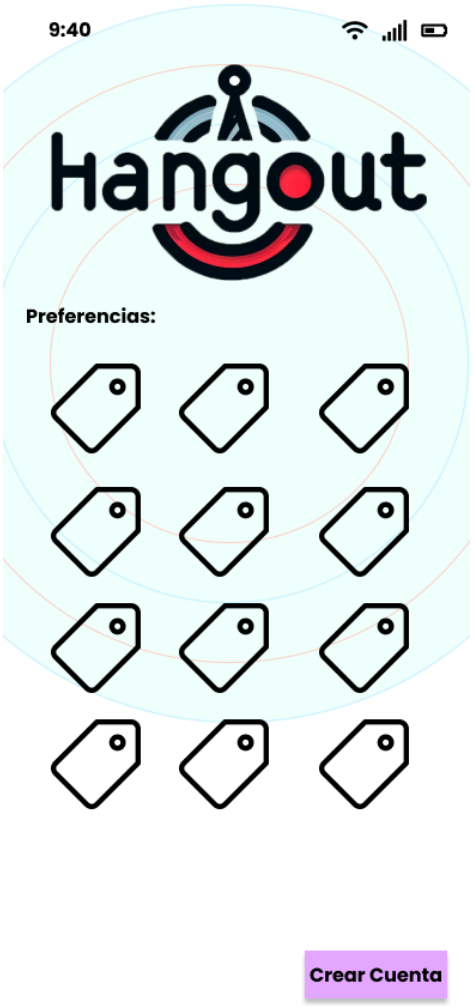
5.5. Diseño de MockUps para Frontend

El diseño de mockups para el frontend es una parte importante para el desarrollo de la interfaz de usuario de la aplicación. Los mockups son representaciones visuales de la interfaz que muestra cómo se verán las páginas y componentes de la aplicación antes de su implementación. Estos diseños ayudan a planificar la estructura y la navegación, asegurando una experiencia de usuario intuitiva y agradable.

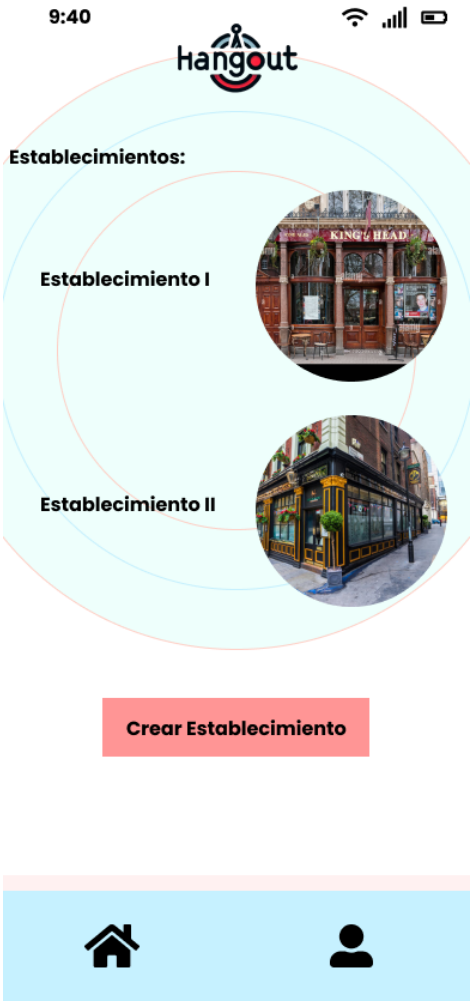
Para la elaboración de estos mockups me inspiré en un diseño de aplicación disponible en Behance. El diseño específico en el cual basé el diseño se titula “Eventify - Event and Party App Case Study” [25]. A continuación se presentan algunos mockups que utilicé para el diseño del frontend los cuales sirvieron de base para la creación de la interfaz de usuario.

(a) Inicio de aplicación

(b) Formularios de la aplicación

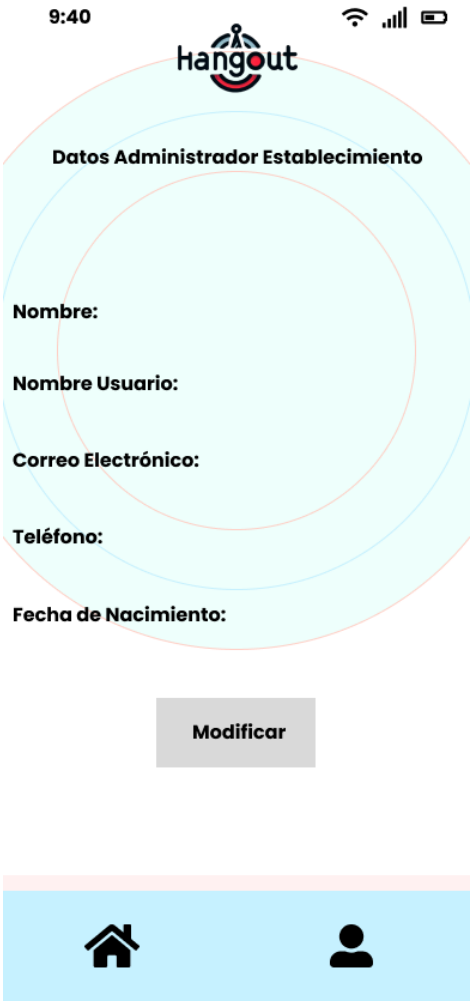


(a) Selección de preferencias



(b) Inicio de Administrador de Establecimiento

(a) Inicio de Usuario Genérico



(b) Muestra de Datos

Capítulo 6

Implementación

En esta sección se describen las etapas llevadas a cabo durante el proceso de la implementación del proyecto, los desafíos encontrados y las soluciones adoptadas. También se incluyen ejemplos de código y capturas de pantalla que ilustran el profreso y los resultados obtenidos.

6.1. Preparación del Entorno de Desarrollo

Antes de iniciar la implementación, se realizó la configuración del entorno de desarrollo. Esto incluyó la instalación de todas las dependencias necesarias y la configuración de herramientas de desarrollo mencionadas en el apartado previo, como Visual Studio Code y Postman.

6.2. Servidor

En este apartado, se describe la implementación en el lado del servidor de la aplicación. Se definirá cómo se realizó la estructura del código y su implementación, proporcionando algunos ejemplos para la creación de la API. A continuación, se definirán los pasos seguidos para lograr esta implementación.

1. **Definición de los Endpoints:** Se definieron los endpoints necesarios para gestionar las llamadas relacionadas con la administración de usuarios, establecimientos, eventos, ofertas y reseñas.

2. **Implementación del Modelo:** En esta etapa se implementaron las clases necesarias a partir del diagrama UML del capítulo anterior. Estas

clases se encargan de gestionar la comunicación con la base de datos y proporcionar una estructura clara y manejable para los datos utilizados en la aplicación.

3. **Implementación de Esquemas:** Se implementaron esquemas para la correcta validación de los datos enviados a la API. Para cada entidad se creó un archivo de esquema específico, lo cual organiza mejor el código y asegura que los datos recibidos cumplan con los requisitos esperados. Esta estructuración facilita la gestión y validación de datos, asegurando la integridad y la consistencia.

4. **Implementación de los Endpoints:** La implementación de los endpoints se realizó siguiendo una estructura RESTful. Para cada entidad se creó un archivo de servicio y un bluepring, lo cual evita la repetición de definiciones para cada endpoint. Esta estructura modular garantiza una comunicación eficiente y coherente. La organización del código mediante blueprints además facilita la futura escalabilidad de la API.

5. **Pruebas en Postman:** Se realizaron diferentes pruebas para comprobar el correcto funcionamiento de todos los endpoints utilizando la herramienta descrita anteriormente, Postman. Se verificaron las llamadas POST para inserciones a la base de datos, las llamadas GET para obtener información, las llamadas PUT para la modificación de datos y las llamadas DELETE para eliminarlos.

6.2.1. Servicios

El servidor atiende principalmente las solicitudes de creación, consulta, modificación y eliminación de entidades. Estas operaciones pueden ser llevadas a cabo por usuarios genéricos o administradores de establecimientos, dependiendo de las entidades que estén gestionando.

Autenticación y Roles de Usuario

Los usuarios podrán iniciar sesión con su cuenta, lo que generará un token de acceso que servirá en las cabeceras de las peticiones HTTP para identificar al usuario que realiza la solicitud. En algunas operaciones, este token será obligatorio para poder completarse.

Al iniciar sesión, al usuario también se le asignará un rol". Esto es necesario para mostrar las pantallas correspondientes a su rol cuando el usuario inicie sesión. Esto asegura que cada usuario vea y acceda únicamente a las

funcionalidades que le correspondan según su tipo de usuario.

Wilson Score

En el diagrama de clases del diseño lógico, cada establecimiento está asociado con reseñas que tienen una calificación entre 0 y 5. Esto da a lugar a una calificación media para cada establecimiento, calculada como la suma de todas las calificaciones divididas por el número total de reseñas del establecimiento. Aunque ordenar los establecimientos por calificación media puede ser útil, es importante considerar la fiabilidad de esta calificación. Por ejemplo, un establecimiento con sólo 2 reseñas y una media de 4.5 no es comparable con otro que tiene 250 reseñas y la misma calificación media. Para abordar este problema, se puede aplicar el método de Wilson Score.[26]

$$S_w = \frac{1}{1 + \frac{1}{n}z^2}\left[p + \frac{1}{2n} - z\sqrt{\frac{p(1-p)}{n} + \frac{z2}{4n2}}\right]$$

El método de Wilson Score ajusta las calificaciones en función del tamaño de la muestra, proporcionando una estimación más precisa de la calidad del establecimiento. Este método tiene en cuenta tanto la calificación media como la fiabilidad de esa calificación en función del número de reseñas. Es una herramienta útil para ordenar los establecimientos de manera más precisa, ya que pondera adecuadamente las calificaciones según la cantidad de datos disponibles.

Sea n la cantidad de reseñas de un establecimiento:

$$\text{Media del establecimiento} = \frac{\sum \text{calificaciones de reseñas}}{n}$$

Convertimos la media a una proporción en una escala de 0 a 1:

$$p = \frac{\text{Media del establecimiento}}{5}$$

El valor crítico z para un 95 % de confianza es aproximadamente 1.96.

Calculamos el denominador:

$$\text{Denominador} = 1 + \frac{z^2}{n}$$

Ajustamos la probabilidad en el centro:

$$\text{Probabilidad ajustada en el centro} = p + \frac{z^2}{2n}$$

La desviación estándar ajustada es:

$$\text{Desviación estándar ajustada} = \sqrt{\frac{p(1-p) + \frac{z^2}{4n}}{n}}$$

Finalmente, el límite inferior del Wilson Score se calcula como:

$$\text{Límite inferior} = \frac{\text{Probabilidad ajustada en el centro} - z \cdot \text{Desviación estándar ajustada}}{\text{Denominador}}$$

El Wilson Score se obtiene multiplicando el límite inferior por 5:

$$\text{Wilson Score} = \text{Límite inferior} \cdot 5$$

Algorithm 1 Cálculo del Wilson Score y ordenamiento de establecimientos

Require: establecimientos
Ensure: Lista de establecimientos ordenados por Wilson Score

```
function WILSON_SCORE(establecimiento)
    Obtener reseñas del establecimiento
    Calcular media de calificaciones
    Convertir media a proporción en escala de 0 a 1
    Calcular límite inferior del Wilson Score
    return Wilson Score
end function
function ORDENAR_ESTABLECIMIENTOS(establecimientos)
    puntajes ← []
    for cada establecimiento en establecimientos do
        puntaje ← wilson_score(establecimiento)
        Añadir (establecimiento, puntaje) a puntajes
    end for
    Ordenar puntajes por puntaje de forma descendente
    ordenados ← [puntaje[0] para puntaje en puntajes]
    return ordenados
end function
```

Filtrar Establecimientos

Existen dos tipos de filtrado de establecimiento en el backend:

1. **Filtro Personalizado:** Basado en las preferencias indicadas por el usuario en su cuenta. Este filtro utiliza una lógica OR, mostrando todos los establecimientos que cumplan con alguna de las preferencias del usuario.

2. **Filtro Usual:** Permite buscar las preferencias deseadas utilizando una lógica AND, mostrando únicamente los establecimientos que cumplan con todas las especificaciones indicadas por el usuario en su selector de ambientes en la página inicial.

6.3. Cliente

El cliente se refiere a la parte de la aplicación que reside y se ejecuta en los dispositivos de los usuarios. En la aplicación, la lógica principal se encuentra centralizada en el servidor, donde se almacenan y procesan los

datos esenciales. El cliente actúa como la interfaz que presenta estos datos al usuario final y le permite interactuar con ellos mediante llamadas a la API.

La implementación del cliente está diseñada para ofrecer una navegación intuitiva, interfaces atractivas y un rendimiento óptimo, garantizando una experiencia positiva.

Para la implementación del frontend ha sido necesario crear nuestros propios componentes, utilizarlos en pantallas, realizar la integración con llamadas al backend y, por último, implementar la navegación entre pantallas. Todo esto se describe a continuación:

6.3.1. Componentes Personalizados

Los componentes en React Native son unidades reutilizables y aisladas que definen la interfaz de usuario de una aplicación. Los componentes se pueden definir como funciones o clases de JavaScript y reciben entradas llamadas "props". Los componentes funcionales son simples funciones JavaScript que retornan elementos de React, mientras que los componentes de clase pueden contener más lógica y estado interno. Los componentes permiten componer interfaces complejas a partir de piezas más pequeñas, facilitando la reutilización y el mantenimiento del código. [27]

Para el desarrollo de la aplicación han sido creados los siguientes componentes:

1. **Establecimiento:** Cuadro con bordes que muestra el nombre, la imagen y el ambiente del establecimiento, así como su calificación media y el número de valoraciones.

2. **Evento:** Cuadro con bordes que muestra el nombre y la imagen del evento, además de su fecha.

3. **Oferta:** Cuadro con bordes que muestra el nombre de la oferta y su valor.

4. **Usuario:** Cuadro con bordes que muestra la imagen de perfil y el nombre de usuario.

5. **Review:** Cuadro que muestra la información de la reseña, incluyendo el nombre del creador y el establecimiento en el que se colocó.

6. **Actividad:** Cuadro que muestra información general de la actividad.

7. **Preferencia:** Muestra los ambientes existentes con una imagen y su texto descriptivo.

8. **Fondo:** Fondo común con círculos de varios colores para un diseño visualmente atractivo.

9. **Footer:** Footer para navegar poder navegar al perfil del usuario que ha iniciado sesión o a su pantalla de inicio.

Estos componentes fueron diseñados para crear una interfaz de usuario coherente y eficiente, permitiendo una navegación intuitiva y una experiencia de usuario satisfactoria. Algunos de estos componentes tienen la opción de ser pulsados para luego ver una pantalla con su información más detallada.

6.3.2. Pantallas

Se diseñaron pantallas tanto para los administradores de establecimientos como para los usuarios genéricos. Las pantallas, con su initerfaz atractiva, permiten a los usuarios interactuar con la aplicación de manera eficiente y sencilla. Esto se ha diseñado para proporcionar una experiencia de usuario fluida y efectiva, permitiendo tanto a los administradores como usuarios interactuar correctamente con la aplicación. Algunas pantallas creadas fueron:

1. **Registro:** La pantalla obtendrá los datos introducidos por el usuario en un formulario. Dentro de este formulario, habrá un campo específico para elegir si el usuario a crear es un administrador de establecimiento o un usuario genérico. Dependiendo de la selección realizada, el formulario solicitará DNI de ser un administrador de establecimiento o solicitará las preferencias de ser un usuario genérico.

2. **Inicio Usuario:** Será la pantalla inicial para aquellos que inicien sesión como usuario genérico. En esta página se pueden encontrar distintos componentes que mejoran la experiencia del usuario: establecimientos ordenados según las preferencias del usuario y utilizando el Wilson Score para asegurar una clasificación precisa basada en calificaciones y número de reseñas; una sección que muestra los próximos eventos a celebrarse, ordenados por fecha más próxima, permitiendo al usuario mantenerse informado sobre las actividades y eventos relevantes; y una visualización de las actividades próximas a realizarse en las cuales el usuario participe, facilitando la organización y seguimiento de

sus eventos sociales.

3. **Inicio Administrador:** Será la pantalla inicial para aquellos que inicien sesión como administrador. En esta página se le permite crear un nuevo establecimiento o gestionar aquellos que ya existan. Se utilizará el componente establecimiento para poder acceder a los datos específicos de un establecimiento.

4. **Datos Entidad:** Será una pantalla informativa que obtendrá los datos recibidos del backend y los mostrará por pantalla. El funcionamiento para mostrar los datos será similar en todas las pantallas de visualización de datos.

5. **Datos Establecimiento:** Existen dos pantallas para mostrar los datos del establecimiento. Desde la pantalla del **administrador**, se podrán ver todos los eventos y ofertas asociados al establecimiento y gestionarlos. En la pantalla del **usuario genérico**, se mostrará información detallada del establecimiento y habrá un botón que permitirá al usuario ver las ofertas o eventos, dependiendo de su interés. Esta diferenciación asegura que tanto los administradores como los usuarios genéricos tengan acceso a la información y funcionalidades relevantes de manera eficiente y organizada.

6.3.3. Integración con Backend

La integración del frontend con el backend sse realiza principalmente mediante la pulsación de botones en la interfaz de usuario. Al pulsar un botón, se invoca a una función dentro de la pantalla correspondiente. Esta función realiza una llamada HTTP al servidor utilizando la bibliotexa *axios*. Las principales llamadas realizadas son:

1. **Creación de Entidad:** Se hace una llamada a POST a la dirección de la API para crear el registro de la entidad, enviando los datos de la entidad proporcionados mediante el uso de formmularios.

2. **Obtención de Entidad:** Se hace una llamada GET a la dirección de la API para obtener el registro de la entidad. Se reciben los datos de la entidad y se muestran.

3. **Modificación de Entidad:** Se hace una llamada PUT a la dirección de la API para modificar el registro de la entidad, enviando los datos modificados de la entidad mediante el uso de formularios.

4. **Eliminación de Entidad:** Se hace una llamda DELETE a la dirección de la API para eliminar el registro de la entidad.

En el envío de cualquier petición HTTP, además se envía el token del usuario que ha inicado sesión. Este token se obtiene al iniciar sesión, cuando la API lo envía al cliente. Se incluye en las cabeceras de las solicitudes HTTP, permitiendo identificar y autenticar al usuario que realiza la petición.

6.3.4. Navegación

Para la navegación por las páaginas, como se ha mencionado anteriormente, hay dos tipos de usuarios principales. Para evitar la necesidad de crear dos pantallas de inicio de sesión diferentes, se ha implementado un único inicio de sesión que, al autenticar a un usuario, también identifica su rol. Dependiendo del rol, se mostrarán unas pantallas u otras para su navegación.

Es importante mencionar que todas las pantallas tienen un footer como se ha comentado en el apartado de componentes, este footer permite acceder al perfil del usuario o a la pantalla de inicio, según el tipo de usuario. En el caso del usuario genérico este footer también le permite acceder a la pantalla de creación de actividades.

Navegación Inicial

Primero, vemaos la navegación inicial desde que el usuario llega a la aplicación y se le permite registrarse o iniciar sesión.

Navegación del Usuario Genérico

La navegación del usuario genérico le permite ver establecimientos y organizar activiadaes sociales. Esta navegación está diseñada para proporcionar fácil acceso a las funciones que este necesite.

Navegación del Adminstrador de Establecimiento

Por último la navegación del administrador de establecimiento permite administrar el establecimiento que este elija y gestionar eventos u ofertas.

A continuación se muestran imágenes de las distintas navegaciones:

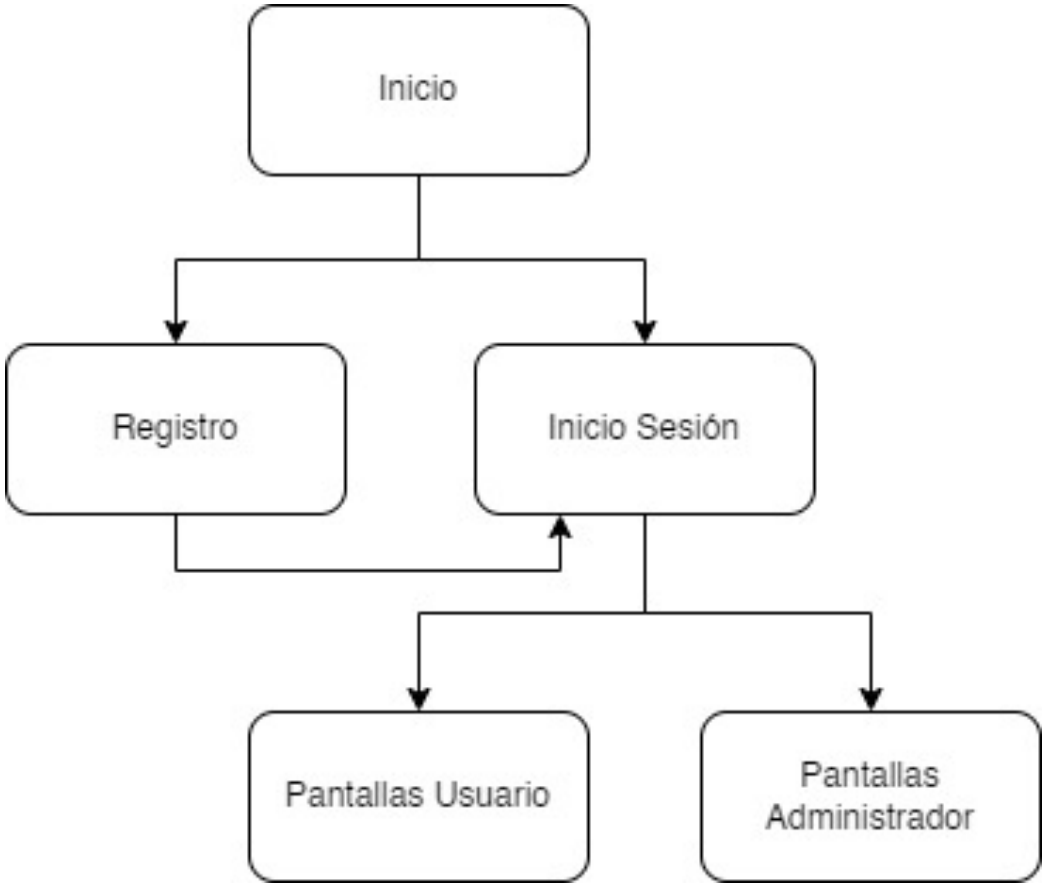


Figura 6.1: Navegación inicial de la aplicación

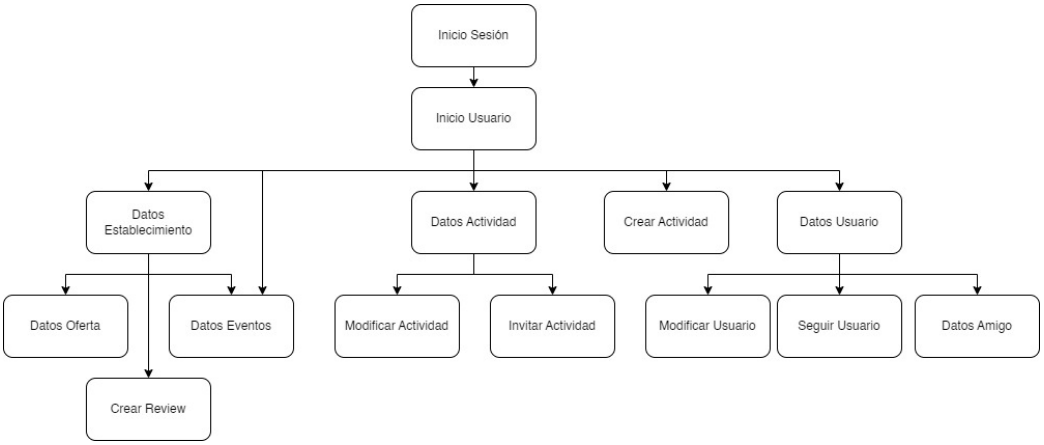


Figura 6.2: Navegación del usuario genérico en la aplicación

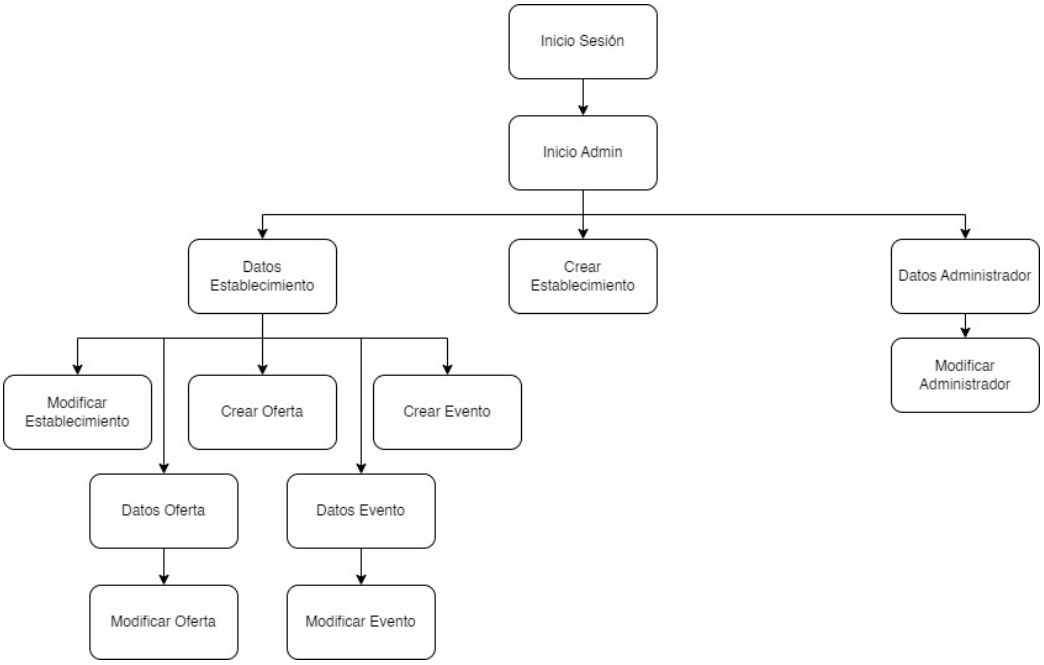


Figura 6.3: Navegación del administrador de establecimientos en la aplicación

Capítulo 7

Pruebas

Las pruebas en el desarrollo de una aplicación móvil aseguran funcionalidad y confiabilidad, detectando errores para prevenir fallos catastróficos. Validan los requisitos y el diseño desde etapas tempranas, mejorando la precisión y reduciendo costos. Al gestionar sistemáticamente el proceso de pruebas, estas mejoran la calidad del software, haciéndolas indispensables en las prácticas modernas del desarrollo de aplicaciones.[28]

Las pruebas realizadas en el proyecto han sido pruebas unitarias en el backend para verificar el correcto funcionamiento de las funciones principales del modelo. Cada entidad realiza las pruebas utilizando un *mock* para simular las peticiones y respuestas de las llamadas a la base de datos.

Para estas pruebas, se utilizó *pytest*, un marco de pruebas en Python que facilita la escritura de casos de pruebas simples y escalables. Pytest además tiene su propio mock gracias a la biblioteca *pytest-mock* que permite la simulación de interacciones con la base de datos. Pytest tiene las siguientes características:

1. **Detección automática de módulos y funciones de prueba**: Pytest encnuentra y ejecuta automaticamente las pruebas definidas en el proyecto sin ninguna configuración adicional.

2. **Informes detallados**: Elimina la necesidad de recordar nombres específicos de métodos de aserción, proporcionando informas claros y detallados sobre los fallos.

3. **Fixtures modulares**: Permiten gestionar recursos pequeños o parametrizados a largo plazo, facilitando la reutilización y modularidad en las pruebas.

Además de las pruebas unitarias realizadas, la aplicación se encuentra en fase alhpa donde el único usuario que ha interactuado con la aplicación he sido yo. He realizado pruebas desde el frontend para comprobar el correcto funcionamiento de las pantallas y su navegación, y pruebas en Postman para comprobar el envío de información en formato JSON a los diferentes endpoints de la API.

Capítulo 8

Despliegue

Bibliografía

[1] Universidad de Granada. Origen del estudiantado. Transparente. URL https://transparente.ugr.es/areas/estudiantado/origen-estudiantes.

[2] Tripadvisor. About tripadvisor. Tripadvisor. URL https://tripadvisor.mediaroom.com/us-about-us.

[3] Yelp. Fast facts. Yelp. URL https://www.yelp-press.com/company/fast-facts/default.aspx.

[4] Think with Google. How the travel research process plays out in time-to-make-a-plan moments. Think with Google. URL https://www.thinkwithgoogle.com/consumer-insights/consumer-trends/travel-research-process-make-a-plan-moments/.

[5] Santander Open Academy. Metodologías de desarrollo de software. Santander Open Academy. URL https://www.santanderopenacademy.com/es/blog/metodologias-desarrollo-software.html.

[6] Amazon. Asus tuf gaming f15 fx506hc-hn004, . URL https://www.amazon.es/ASUS-TUF-Gaming-F15-FX506HC-HN004/dp/B097T3XB81.

[7] Amazon. Samsung galaxy prism versión alemana, . URL https://www.amazon.es/Samsung-Galaxy-Prism-VersiÃşn-Alemana/dp/B07MTXWLRP.

[8] Amazon. Benq gw2480 pantalla inteligente superestrecho, . URL https://www.amazon.es/BenQ-GW2480-Pantalla-Inteligente-superestrecho/dp/B073NTCT4Q.

[9] Seguridad Social. Cotización / recaudación de trabajadores. Seguridad Social. URL https://www.seg-social.es/wps/portal/wss/

internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537#36538.

[10] R. Malan and D. Bredemeyer. Functional requirements and use cases. Bredemeyer Consulting, 2001.

[11] M. Glinz. On non-functional requirements. In *15th IEEE international requirements engineering conference (RE 2007)*, pages 21–26. IEEE, October 2007.

[12] IBM. Use case models. IBM Documentation. URL https://www.ibm.com/docs/es/dmrt/9.5?topic=approaches-use-case-models.

[13] D. Garlan. *Software architecture*. 2008.

[14] K. B. Laskey and K. Laskey. Service oriented architecture. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(1):101–105, 2009.

[15] Python Software Foundation. General python faq. Python.org, . URL https://www.python.org/doc/essays/blurb/.

[16] Python Basics. What is flask python? Python Basics. URL https://pythonbasics.org/what-is-flask-python/.

[17] MongoDB. ¿qué es mongodb? MongoDB. URL https://www.mongodb.com/es/company/what-is-mongodb.

[18] Node.js Foundation. About. Node.js, . URL https://nodejs.org/en/about.

[19] GeeksforGeeks. Jdk in java. GeeksforGeeks. URL https://www.geeksforgeeks.org/jdk-in-java/.

[20] React Native. Learn once, write anywhere. React Native. URL https://reactnative.dev.

[21] Microsoft. Visual studio code documentation. Visual Studio Code. URL https://code.visualstudio.com/docs.

[22] Android Developers. Introducción a android studio. Android Developers. URL https://developer.android.com/studio/intro?hl=es-419.

[23] Postman. What is postman? Postman. URL https://www.postman.com/product/what-is-postman/.

[24] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on uml class diagrams. *Artificial intelligence*, 168(1-2):70–118, 2005.

[25] Polina Rusenova. Eventify - event and party app case study. Behance. URL https://www.behance.net/gallery/197875467/Eventify-event-party-app-case-study?tracking_source=search_projects|mobile+design+ui+party+app&l=2.

[26] J. O. Talton III, K. Dusad, K. Koiliaris, and R. S. Kumar. How do people sort by ratings? In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–10, May 2019.

[27] React. Componentes y propiedades. React. URL https://es.legacy.reactjs.org/docs/components-and-props.html.

[28] H. Zhu, P. A. Hall, and J. H. May. Software unit test coverage and adequacy. *Acm computing surveys (csur)*, 29(4):366–427, 1997.