Visionova

Soluciones Tecnológicas para la Inclusión de Personas con Discapacidad Visual en el Entorno Urbano

---

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

UNIVERSITAT DE BARCELONA

Autor: Ivan Mansilla Flores
Tutor: Ricardo Marques

Resumen

Las personas con discapacidad visual enfrentan numerosos desafíos en su vida diaria, desde leer un menú en un restaurante hasta desplazarse por la ciudad evitando rutas peligrosas. En España y en el mundo, millones de personas ciegas o con baja visión luchan por mantener su independencia y calidad de vida, a menudo enfrentando barreras significativas en su acceso a la tecnología y la información.
A pesar de los avances tecnológicos, las soluciones existentes no siempre logran cubrir todas las necesidades de este colectivo. Aunque hay aplicaciones móviles diseñadas para ayudar a personas ciegas, como aquellas que identifican objetos o proporcionan asistencia en la navegación, estas herramientas no ofrecen una solución integral y muchas veces son limitadas en su funcionalidad y accesibilidad.

El presente trabajo de fin de grado (TFG) se centra en la creación y desarrollo de "Visionova", una aplicación móvil destinada a mejorar significativamente la calidad de vida de las personas con discapacidad visual. Visionova pretende integrar varias funcionalidades de asistencia en una sola plataforma, utilizando tecnologías avanzadas como el reconocimiento de voz, la geolocalización y el análisis de imágenes.

El objetivo principal de Visionova es proporcionar una herramienta multifuncional que facilite tareas diarias como la identificación de objetos, la lectura de textos impresos, y la navegación en interiores y exteriores de manera segura y autónoma. La aplicación está diseñada para ser intuitiva y accesible, ofreciendo una interfaz sencilla que permite a los usuarios acceder a todas las funciones con facilidad.
En este TFG se detallan los diferentes componentes y tecnologías utilizados en el desarrollo de Visionova, incluyendo el uso de React Native para el desarrollo multiplataforma, la integración con servicios de Google Cloud para el reconocimiento de imágenes y la implementación de un sistema de navegación basado en geolocalización. También se presentan pruebas y evaluaciones de la usabilidad de la aplicación, destacando sus ventajas y áreas de mejora.

El trabajo concluye con un análisis de los resultados obtenidos en las pruebas de usuario, demostrando que Visionova puede mejorar significativamente la independencia y la calidad de vida de las personas con discapacidad visual. Se identifican futuras líneas de desarrollo para seguir optimizando la aplicación y ampliando sus funcionalidades, con el fin de proporcionar una herramienta aún más completa y eficaz para este colectivo.

Resum

Les persones amb discapacitat visual enfronten nombrosos desafiaments en la seva vida diària, des de llegir un menú en un restaurant fins a desplaçar-se per la ciutat evitant rutes perilloses. A Espanya i en el món, milions de persones cegues o amb baixa visió lluiten per mantenir la seva independència i qualitat de vida, sovint enfrontant barreres significatives en el seu accés a la tecnologia i la informació. Malgrat els avanços tecnològics, les solucions existents no sempre aconsegueixen cobrir totes les necessitats d'aquest col·lectiu. Encara que hi ha aplicacions mòbils dissenyades per a ajudar a persones cegues, com aquelles que identifiquen objectes o proporcionen assistència en la navegació, aquestes eines no ofereixen una solució integral i moltes vegades són limitades en la seva funcionalitat i accessibilitat.

El present treball de fi de grau (TFG) se centra en la creació i desenvolupament de "Visionova", una aplicació mòbil destinada a millorar significativament la qualitat de vida de les persones amb discapacitat visual. Visionova pretén integrar diverses funcionalitats d'assistència en una sola plataforma, utilitzant tecnologies avançades com el reconeixement de veu, la geolocalització i l'anàlisi d'imatges.

L'objectiu principal de Visionova és proporcionar una eina multifuncional que faciliti tasques diàries com la identificació d'objectes, la lectura de textos impresos, i la navegació en interiors i exteriors de manera segura i autònoma. L'aplicació està dissenyada per a ser intuïtiva i accessible, oferint una interfície senzilla que permet als usuaris accedir a totes les funcions amb facilitat. En aquest TFG es detallen els diferents components i tecnologies utilitzats en el desenvolupament de Visionova, incloent-hi l'ús de React Native per al desenvolupament multiplataforma, la integració amb serveis de Google Cloud per al reconeixement d'imatges i la implementació d'un sistema de navegació basat en geolocalització. També es presenten proves i avaluacions de la usabilitat de l'aplicació, destacant els seus avantatges i àrees de millora.

El treball conclou amb una anàlisi dels resultats obtinguts en les proves d'usuari, demostrant que Visionova pot millorar significativament la independència i la qualitat de vida de les persones amb discapacitat visual. S'identifiquen futures línies de desenvolupament per a continuar optimitzant l'aplicació i ampliant les seves funcionalitats, amb la finalitat de proporcionar una eina encara més completa i eficaç per a aquest col·lectiu.

Abstract

People with visual impairment face numerous challenges in their daily lives, from reading a menu in a restaurant to getting around the city while avoiding dangerous routes. In Spain and around the world, millions of people who are blind or have low vision struggle to maintain their independence and quality of life, often facing significant barriers in their access to technology and information.
Despite technological advances, existing solutions do not always meet all the needs of this group. Although there are mobile applications designed to help blind people, such as those that identify objects or provide navigation assistance, these tools do not offer a comprehensive solution and are often limited in their functionality and accessibility.

This thesis focuses on the creation and development of "Visionova", a mobile application designed to significantly improve the quality of life of visually impaired people. Visionova aims to integrate several assistive functionalities in a single platform, using advanced technologies such as voice recognition, geolocation and image analysis.

Visionova's main objective is to provide a multifunctional tool that facilitates everyday tasks such as identifying objects, reading printed text, and navigating indoors and outdoors safely and autonomously. The application is designed to be intuitive and accessible, offering a simple interface that allows users to access all functions with ease.
This TFG details the different components and technologies used in the development of Visionova, including the use of React Native for cross-platform development, the integration with Google Cloud services for image recognition and the implementation of a geolocation-based navigation system. Tests and evaluations of the usability of the application are also presented, highlighting its advantages and areas for improvement.

The paper concludes with an analysis of the results obtained in the user tests, demonstrating that Visionova can significantly improve the independence and quality of life of visually impaired people. Future lines of development are identified to further optimize the application and expand its functionalities, in order to provide an even more complete and effective tool for this group.

Índice general

Índice de Figuras

Figura

1. Introducción

1.1. Motivación

Desde una edad temprana, he estado rodeado de familiares y amigos cercanos con discapacidades visuales, lo que me ha permitido comprender de primera mano los desafíos a los que se enfrentan diariamente. Esta comprensión se ha visto reforzada a través de mi labor como voluntario en la ONCE (Organización Nacional de Ciegos Españoles) y como árbitro de goalball, un deporte diseñado específicamente para personas con discapacidad visual. Estas experiencias me han brindado una perspectiva única sobre las necesidades y las dificultades que enfrentan las personas con discapacidad visual y me han motivado profundamente a buscar soluciones tecnológicas que puedan mejorar su calidad de vida.

A lo largo de los años, he observado cómo la falta de aplicaciones accesibles puede limitar significativamente la independencia de las personas con discapacidad visual. Sin embargo, también he sido testigo del impacto positivo que pueden tener las herramientas tecnológicas adecuadamente diseñadas. La posibilidad de utilizar una aplicación que facilite la movilidad y el acceso a la información puede transformar la vida de una persona ciega o con baja visión, brindándole una mayor autonomía y seguridad.

Mi motivación para desarrollar esta aplicación surge de la combinación de estas experiencias personales y profesionales. He visto cómo aplicaciones bien diseñadas pueden hacer una diferencia significativa, y estoy convencido de que, mediante el uso de tecnologías modernas como React Native, podemos crear soluciones que sean tanto eficaces como accesibles. Este Trabajo de Fin de Grado (TFG) representa una oportunidad para aplicar mis conocimientos en ingeniería informática a un proyecto con un impacto social tangible, y estoy decidido a aprovechar al máximo esta oportunidad para contribuir de manera positiva a la comunidad de personas con discapacidad visual.

1.2. Objetivos y enfoque

El desarrollo de esta aplicación tiene varios objetivos específicos que guiarán el proceso de creación y asegurará que la aplicación sea útil y accesible para las personas con discapacidad visual. Los objetivos principales son los siguientes:

1. **Planificación y Organización**: Establecer un plan de trabajo detallado que incluya objetivos específicos y plazos para cada etapa del desarrollo del proyecto. Esto garantizará que el desarrollo de la aplicación se realice de manera eficiente y dentro del plazo establecido.

2. **Investigación de Tecnologías Relevantes**: Explorar y seleccionar las tecnologías más adecuadas para el desarrollo de la aplicación, como React Native, Firebase y APIs de geolocalización y transporte público. La investigación exhaustiva de estas

tecnologías permitirá integrar funcionalidades clave en la aplicación.

3. **Integración de API's Necesarias**: Identificar y evaluar las API's disponibles para integrar funcionalidades como la obtención de datos de transporte público y la geolocalización. Esto asegurará que la aplicación proporcione información precisa y actualizada a los usuarios.

4. **Tecnología de Voz para Discapacitados Visuales**: Explorar soluciones de tecnología de voz que puedan mejorar la accesibilidad de la aplicación. Esto incluirá la investigación de tecnologías de conversión de texto a voz que permitan a los usuarios escuchar el contenido de las imágenes capturadas.

5. **Desarrollo de la Base de la Aplicación**: Definir los requisitos funcionales y no funcionales de la aplicación, realizar un análisis DAFO (Debilidades, Amenazas, Fortalezas y Oportunidades) y definir los casos de uso. Esto proporcionará una base sólida para el desarrollo de la aplicación.

6. **Implementación de Funcionalidades Clave**: Desarrollar e implementar funcionalidades específicas como la conversión de imágenes a texto y de texto a voz, la grabación y compartición de rutas, y la creación de perfiles de usuario. Estas funcionalidades serán fundamentales para mejorar la experiencia de usuario y la accesibilidad de la aplicación.

7. **Documentación y Revisión Final**: Elaborar documentación detallada que describa el proceso de desarrollo de la aplicación, incluyendo diagramas de flujo y manuales de usuario. Realizar una revisión exhaustiva de la aplicación para identificar posibles errores o mejoras, y realizar los ajustes necesarios antes de la entrega final.

1.3. Planificación previa

El desarrollo de este proyecto se planificó que se llevaría a cabo en varias fases bien definidas, cada una con tareas específicas y plazos claros. La siguiente tabla detalla el plan de trabajo, incluyendo las tareas, los responsables, las fechas de inicio y fin, y el estado de cada tarea supuestamente, dejando un margen de un mes hasta la entrega, para así evitar posibles retrasos.

TAREAS	EMPEZAR	FIN	DÍAS
Reunión inicial con tutor para definir TFG	2024-01-26	2024-01-26	1
Definición de objetivos y desarrollo del TFG	2024-01-27	2024-02-01	6
Elaboración de la estructura de la aplicación (Requisitos..	2024-02-02	2024-02-11	10
Investigación sobre tecnologías para discapacidad visual	2024-02-12	2024-02-12	1
Investigación y aprendizaje de React Native	2024-02-13	2024-02-14	2
Aprendizaje de Expo o herramientas similares	2024-02-15	2024-02-15	1
Desarrollo base de la aplicación	2024-02-16	2024-02-19	4
Conexión a la base de datos Firebase	2024-02-20	2024-02-23	4
Integración de API de cámara y herramientas conversión	2024-02-24	2024-02-27	4
Desarrollo de convertidor imagen, texto y voz	2024-02-28	2024-03-13	15
Integración de API de navegación	2024-03-14	2024-03-17	4
Desarrollo de grabación y guia de rutas	2024-03-18	2024-04-01	15
Desarrollo de todo el diseño e interfaz	2024-04-02	2024-04-06	5
Desarrollo de la mini red social	2024-04-07	2024-04-14	8
Documentación del proceso y resultados	2024-04-15	2024-04-28	14
Revisión final y ajustes	2024-05-01	2024-05-12	12
Entrega del TFG	2024-05-11	2024-05-11	1

***Figura 1**: Calendario de planificación previa*

De esta forma visual y organizativa ha sido muy útil durante el curso del TFG, ya que he podido ir modificando el primer calendario inicial si era necesario dependiendo la dificultad de las tareas durante su desarrollo o donde debía centrarme si tenía puntos atrasados.
Este plan de trabajo detallado proporciona una hoja de ruta clara para el desarrollo de la aplicación, asegurando que cada etapa del proyecto se complete de manera oportuna y eficiente. La estructura del plan permite una evaluación continua del progreso y facilita los ajustes necesarios para garantizar que el proyecto cumpla con sus objetivos.

Con este enfoque sistemático y bien organizado, confío en que este proyecto no solo será una valiosa contribución a mi desarrollo académico y profesional, sino que también tendrá un impacto positivo en la vida de las personas con discapacidad visual.

2. Antecedentes

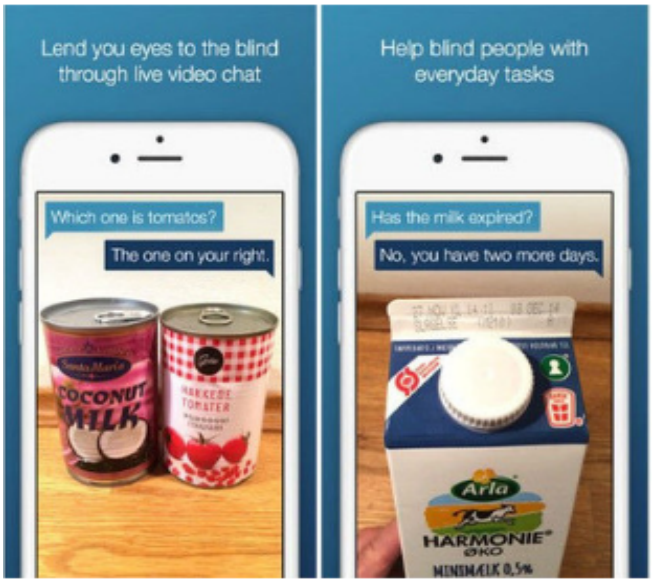
En este apartado se tratará el tema sobre cuánta tecnología en cuanto aplicaciones móviles hay relacionadas con la ayuda en el día a día para los discapacitados visuales. También se investigará las diferentes tecnologías para llevar a cabo el objetivo de dar vida a Vizinova, explorando las mejores soluciones y herramientas.
Con esto en mente, se quiere investigar los puntos fuertes para tratar de realizar un estado del arte, para así seguir con las ventajas que podemos encontrar en el mercado y sobre todo los puntos débiles que tienen las aplicaciones para mejorarlo. Es decir, se quiere lograr el objetivo de hacer algo diferente, una aplicación que pueda englobar diferentes funcionalidades, algo que no podamos encontrar en el mercado y sobre todo mejorar lo ya existente. Para lograrlo, se plantearon los siguientes objetivos: Investigar aplicaciones específicas relacionadas a nuestro objetivo del TFG y más generales, puntos fuertes y débiles de dichas aplicaciones, extraer conclusiones que puedan ser útiles.

2.1. Análisis de aplicaciones móviles relacionadas

La tecnología móvil ha demostrado ser una herramienta poderosa para mejorar la calidad de vida de las personas con discapacidad visual, ofreciendo soluciones innovadoras que facilitan su integración en la sociedad y su independencia en actividades cotidianas. A continuación, se presenta un análisis de algunas aplicaciones móviles relevantes en este campo y cómo podrían inspirar el desarrollo de mi propia aplicación, Visionova, ya sea viendo los puntos fuertes como los puntos débiles de dichas apps [1] [2].

BeMyEyes

Esta aplicación [3] se destaca por su enfoque en la asistencia en tiempo real a través de videollamadas. Permite a las personas ciegas o con discapacidad visual recibir ayuda de voluntarios para realizar tareas cotidianas, como leer etiquetas o identificar objetos. Su punto fuerte radica en la comunidad de voluntarios dispuestos a ayudar, lo que proporciona una solución inmediata a las necesidades de los usuarios. Para Visionova, este modelo comunitario podría inspirar la implementación de una función de asistencia en vivo, donde los usuarios puedan recibir ayuda de otros usuarios en tiempo real durante su navegación urbana.



► Fundada por Hans Jørgen Wiberg.

► Año de fundación: 2015.

► Descargas aproximadas: Más de 4 millones.

► Disponibilidad: Disponible en iOS y Android

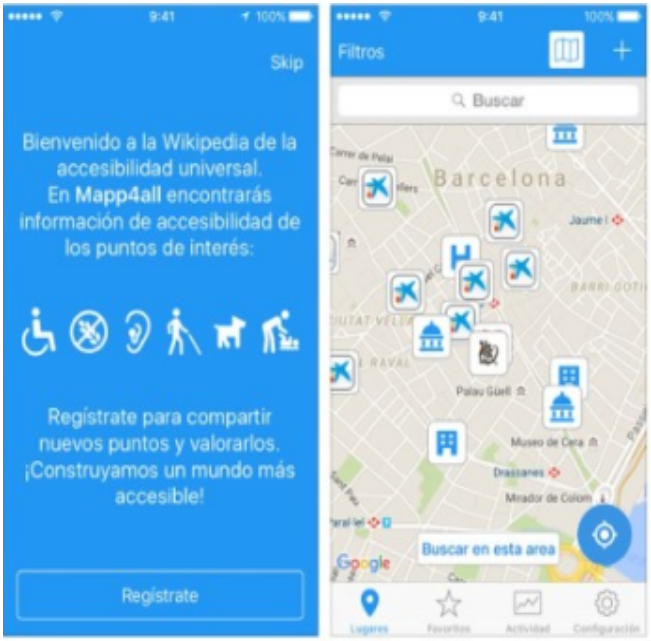
***Figura 2**: Ejemplo de aplicación BeMyEyes*

Punto fuerte: Su enfoque en la asistencia en tiempo real a través de videollamadas permite una solución inmediata a las necesidades de los usuarios, lo cual es valioso para situaciones urgentes o complejas.

Punto débil: La dependencia de voluntarios para proporcionar ayuda puede resultar en tiempos de espera variables y una experiencia inconsistente, lo que podría frustrar a los usuarios que necesitan ayuda rápida y confiable. Para Visionova, este modelo comunitario podría inspirar la implementación de una función de asistencia en vivo, pero también debe considerarse cómo garantizar la disponibilidad constante de ayuda.

Mapp4all

Una aplicación [4] centrada en proporcionar información sobre la accesibilidad en entornos urbanos. Ofrece datos sobre la ubicación de rampas, ascensores y otros elementos que facilitan la movilidad de personas con discapacidad. Su fortaleza reside en su enfoque en la accesibilidad física del entorno. Para Visionova, esta funcionalidad podría inspirar la inclusión de datos sobre accesibilidad en las rutas sugeridas, permitiendo a los usuarios planificar sus viajes de manera más informada y segura.



► Fundada por la organización Map4all.

► Año de fundación: No hay información precisa disponible, pero ha estado activa durante varios años.

► Descargas aproximadas: No hay datos exactos disponibles, pero tiene una base de usuarios significativa.

► Disponibilidad: Disponible en iOS y Android.

***Figura 3**: Ejemplo aplicación Map4all*

Punto fuerte: Su enfoque en proporcionar información sobre la accesibilidad física del entorno urbano es valioso para ayudar a las personas con discapacidad visual a planificar sus viajes de manera más segura e informada.

Punto débil: La aplicación puede carecer de datos actualizados y/o completos sobre la accesibilidad en ciertas áreas, lo que podría limitar su utilidad para los usuarios en ciertos lugares. Para Visionova, la inclusión de datos sobre accesibilidad en las rutas sugeridas podría ser inspirada por esta función, pero se debe abordar la actualización y precisión de estos datos

TapTapSee

Esta aplicación [5] utiliza la tecnología de reconocimiento de imágenes para proporcionar descripciones auditivas de objetos capturados en fotos. Su capacidad para identificar objetos en tiempo real es impresionante y ofrece una solución innovadora para personas con discapacidad visual. Para Visionova, esta funcionalidad podría inspirar la integración de tecnología de reconocimiento de imágenes para ayudar a los usuarios a identificar puntos de referencia y obstáculos durante su navegación.

► Fundada por CamFind Inc.

► Año de fundación: 2012.

► Descargas aproximadas: Más de 1 millón.

► Disponibilidad: Disponible en iOS y Android.

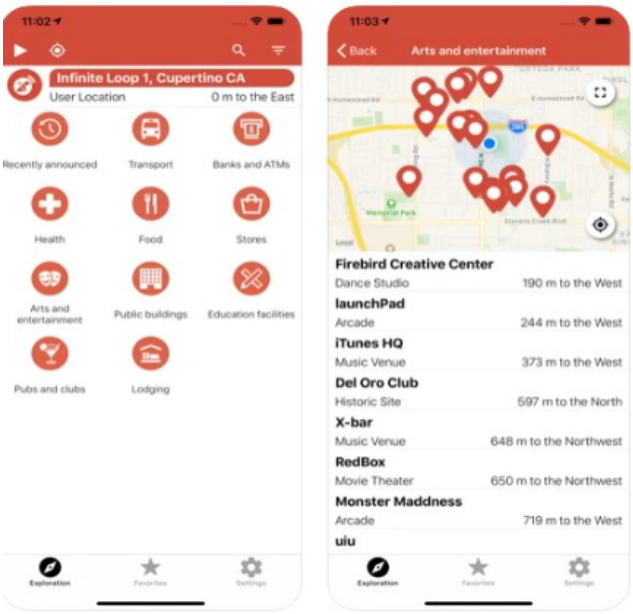
***Figura 4**: Ejemplo aplicación TapTapSee*

Punto fuerte: Su capacidad para identificar objetos en tiempo real utilizando tecnología de reconocimiento de imágenes ofrece una solución innovadora para ayudar a las personas con discapacidad visual a identificar puntos de referencia y obstáculos durante su navegación.

Punto débil: La precisión de la identificación de objetos puede ser variable y algunas veces inexacta, lo que podría afectar la confiabilidad de la aplicación para los usuarios. Para Visionova, la integración de tecnología de reconocimiento de imágenes podría ser inspirada por esta función, pero se debe garantizar la precisión y confiabilidad del reconocimiento.

Lazarillo GPS

Lazarillo GPS [6] ofrece instrucciones de navegación por voz y proporciona información sobre lugares de interés cercanos y obstáculos en el camino. Su enfoque en la navegación precisa y la retroalimentación auditiva en tiempo real lo convierte en una herramienta valiosa para personas con discapacidad visual. Para Visionova, esta funcionalidad podría inspirar la inclusión de instrucciones de navegación por voz y alertas sobre obstáculos en las rutas sugeridas.



► Fundada por René Espinoza y Ricardo Soto.

► Año de fundación: 2015.

► Descargas aproximadas: Más de 100,000.

► Disponibilidad: Disponible en iOS y Android.

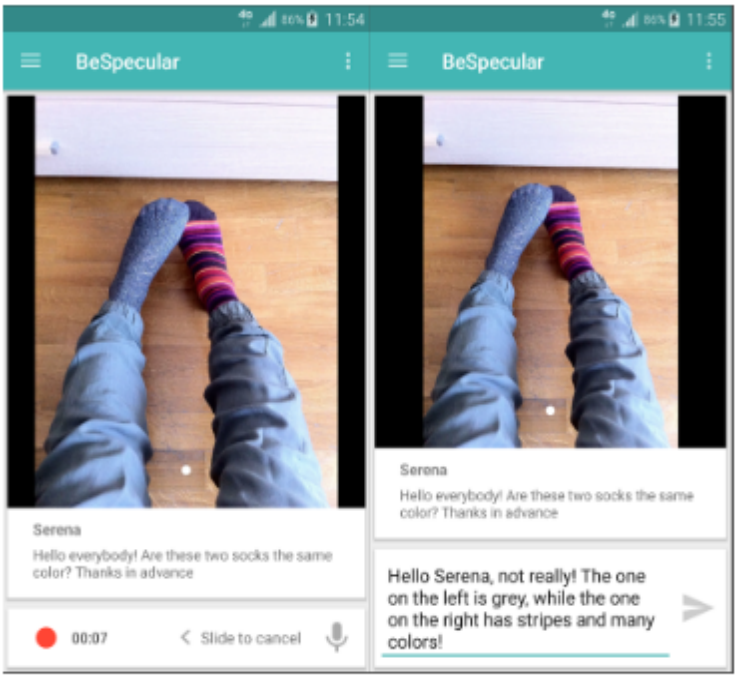
***Figura 5**: Ejemplo aplicación LazarilloGPS*

Punto fuerte: Ofrece instrucciones de navegación por voz y proporciona información sobre lugares de interés cercanos y obstáculos en el camino, lo que es valioso para ayudar a las personas con discapacidad visual a navegar de manera segura y eficiente.

Punto débil: La aplicación puede tener limitaciones en la disponibilidad de datos precisos sobre lugares de interés y obstáculos en ciertas áreas, lo que podría afectar su utilidad para los usuarios en ciertos lugares. Para Visionova, la inclusión de instrucciones de navegación por voz y alertas sobre obstáculos en las rutas sugeridas podría ser inspirada por esta función, pero debe abordarse la precisión y actualización de estos datos.

BeSpecular

BeSpecular [7] es una aplicación móvil que conecta a personas ciegas o con discapacidad visual con voluntarios que pueden ayudar respondiendo preguntas sobre imágenes enviadas. Los usuarios pueden tomar una foto de lo que necesitan ver y enviarla a la comunidad de voluntarios. Estos voluntarios luego proporcionan descripciones verbales o respuestas a las preguntas de los usuarios basadas en la imagen. Esta aplicación fomenta la interacción entre personas con discapacidad visual y personas sin discapacidad, permitiendo un intercambio de información útil y experiencias.



► Fundada por Leandro Franco.

► Año de fundación: 2015.

► Descargas aproximadas: Más de 100,000.

► Disponibilidad: Disponible en iOS y Android.

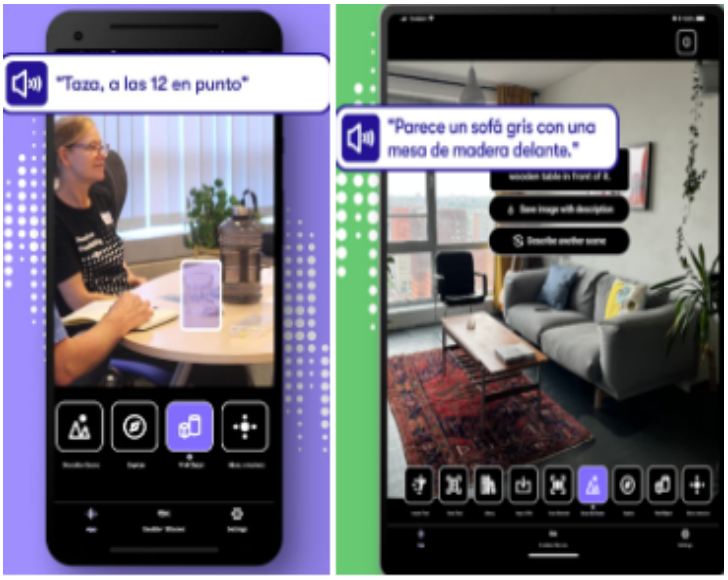
***Figura 6**: Ejemplo aplicación BeSpecular*

Punto fuerte: La capacidad de BeSpecular para conectar a personas ciegas o con discapacidad visual con voluntarios dispuestos a proporcionar descripciones visuales en tiempo real es su mayor fortaleza. Esto brinda a los usuarios la posibilidad de obtener ayuda inmediata y personalizada para una amplia variedad de tareas y situaciones cotidianas, lo que puede aumentar su independencia y autonomía.

Punto débil: Sin embargo, el punto débil de BeSpecular radica en su dependencia de la disponibilidad y la participación de voluntarios. Esto puede llevar a tiempos de respuesta variables y a una experiencia inconsistente para los usuarios. Además, la calidad y precisión de las descripciones visuales pueden variar según la habilidad y la disponibilidad de los voluntarios, lo que podría afectar la utilidad y la satisfacción del usuario.

Envisión

Es una aplicación móvil [8] que utiliza tecnología de inteligencia artificial para reconocer y describir objetos y escenas en tiempo real a través de la cámara de un dispositivo móvil. Esta aplicación ha ganado popularidad desde su lanzamiento y ha sido ampliamente descargada por usuarios en todo el mundo.



► Fundadores: Karthik Kannan y Karthik Mahadevan.

► Año de fundación: Envision fue lanzada en 2018.

► Descargas aproximadas: Más de 100,000.

► Disponibilidad: La aplicación Envision está disponible tanto para dispositivos iOS como Android.

***Figura 7**: Ejemplo aplicación Envisión*

Punto fuerte: Envision destaca por proporcionar retroalimentación auditiva en tiempo real sobre el entorno circundante mediante el reconocimiento y la descripción de objetos, texto y escenas. Esto es fundamental para mejorar la autonomía y calidad de vida de las personas con discapacidad visual, y podría inspirar características similares en Visionova para mejorar la experiencia del usuario.

Punto débil: Su dependencia de una conexión a Internet estable para el procesamiento en la nube podría limitar su funcionalidad en entornos con conectividad limitada. Además, la precisión del reconocimiento puede variar según la calidad de la imagen y la complejidad del entorno, lo que afectaría la confiabilidad y la satisfacción del usuario. Abordar estos desafíos sería crucial para Visionova.

En definitiva, las conclusiones que podemos extraer revelan un panorama diverso de aplicaciones móviles diseñadas para mejorar la vida cotidiana de las personas con discapacidad visual. Desde asistencia en tiempo real hasta reconocimiento de objetos y navegación por voz, estas aplicaciones ofrecen soluciones innovadoras pero también enfrentan desafíos significativos.

Be My Eyes destaca por su enfoque en la asistencia en tiempo real a través de videollamadas, mientras que Mapp4all se centra en proporcionar información sobre la accesibilidad física del entorno urbano. Por otro lado, TapTapSee utiliza tecnología de reconocimiento de imágenes para identificar objetos en tiempo real, y Lazarillo GPS ofrece instrucciones de navegación por voz y alertas sobre obstáculos.

Sin embargo, ninguna de estas aplicaciones aborda completamente la visión de Visionova: una aplicación móvil diseñada para ayudar a las personas con discapacidad visual a navegar por el mundo de manera segura y autónoma. Visionova se inspira en las

fortalezas de estas aplicaciones, desde la posibilidad de poder llegar a cualquier punto y hasta la asistencia y su navegación por voz, pero también aportar cosas novedosas como el leer textos, ya sean cartas importantes o en momentos más cotidianos como poder leer una carta de un restaurante, que sin otra persona esto no sería posible, incluyendo el poder llegar a un punto con la ruta más fácil posible. Podríamos decir que se busca superar sus limitaciones.
Por ejemplo, la función de reconocimiento de objetos o textos en tiempo real de Envision y la navegación por voz de Lazarillo GPS ofrecen puntos de partida valiosos para el desarrollo de Visionova. Sin embargo, Visionova aspira a ser más que una herramienta de asistencia; busca crear una comunidad inclusiva y colaborativa a través de funciones de redes sociales donde los usuarios pueden compartir rutas, experiencias y apoyo mutuo.

En resumen, el estado del arte destaca las fortalezas y debilidades de las aplicaciones existentes, brindando valiosas lecciones para el desarrollo de Visionova como una solución integral y centrada en el usuario para las necesidades de las personas con discapacidad visual y mejorar los puntos débiles y/o limitaciones que se pueden encontrar.

2.2. Lenguaje y herramientas

2.2.1 React Native

React Native es un marco de desarrollo de aplicaciones móviles de código abierto creado por Facebook. Se basa en React, un popular marco de desarrollo web, y permite a los desarrolladores crear aplicaciones móviles multiplataforma utilizando JavaScript y React.

Características principales de React Native

- Multiplataforma: Una de las principales ventajas [9] de React Native es su capacidad para desarrollar aplicaciones móviles tanto para iOS como para Android con un solo código base. Esto permite ahorrar tiempo y recursos al desarrollar y mantener aplicaciones para ambas plataformas.

***Figura 8**: Lenguaje React Native para los sistemas operativos de iOS y Android*

- Componentes reutilizables: React Native utiliza un enfoque basado en componentes, lo que significa que los componentes de la interfaz de usuario pueden ser reutilizados en diferentes partes de la aplicación, lo que facilita la creación de interfaces coherentes y escalables.
- Rendimiento nativo: Aunque React Native utiliza JavaScript para la lógica de la aplicación, compila componentes nativos que se ejecutan directamente en el dispositivo. Esto proporciona un rendimiento similar al de las aplicaciones nativas, lo que garantiza una experiencia de usuario fluida y receptiva.
- Soporte de terceros: React Native cuenta con una amplia comunidad de desarrolladores y una gran cantidad de bibliotecas y herramientas de código abierto disponibles para facilitar el desarrollo de aplicaciones. Esto incluye bibliotecas para navegación, gestión de estado, animaciones y mucho más.

Ventajas de utilizar React Native [10]:

- Desarrollo rápido: Gracias a su enfoque de desarrollo basado en componentes y su capacidad para compartir código entre plataformas, React Native permite un desarrollo rápido y eficiente de aplicaciones móviles [9].
- Actualizaciones en tiempo real: React Native facilita la implementación de actualizaciones en tiempo real, lo que permite a los desarrolladores realizar cambios y correcciones de forma rápida y sencilla sin necesidad de esperar a que se aprueben las actualizaciones de la tienda de aplicaciones.
- Gran comunidad y soporte: La comunidad de desarrolladores de React Native es muy activa y ofrece un amplio soporte a través de foros, grupos de discusión y recursos educativos. Esto facilita el aprendizaje y la resolución de problemas durante el desarrollo de aplicaciones.

Limitaciones y desafíos de React Native:

- Rendimiento en aplicaciones complejas: Aunque React Native ofrece un rendimiento nativo en la mayoría de los casos, puede experimentar problemas de

rendimiento en aplicaciones complejas [11] que requieren una gran cantidad de operaciones intensivas en CPU o gráficos.
- Dependencia de componentes nativos: En algunos casos, puede ser necesario escribir componentes nativos personalizados para acceder a funcionalidades específicas de la plataforma que no están disponibles directamente en React Native. Esto puede añadir complejidad al desarrollo y requerir conocimientos adicionales de desarrollo nativo.

Podríamos decir que React Native es una excelente opción para el desarrollo de aplicaciones móviles multiplataforma debido a su rapidez de desarrollo, rendimiento nativo y gran comunidad de soporte. Con sus características únicas y su amplio ecosistema de herramientas y bibliotecas, React Native ofrece una solución robusta y eficiente para crear aplicaciones móviles modernas y escalables.

2.2.2 Alternativas de desarrollo de aplicaciones móviles

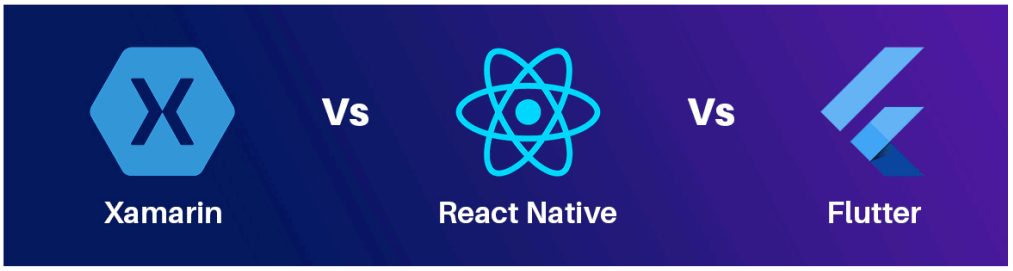
Cuando se trata de desarrollar aplicaciones móviles, hay varias alternativas disponibles [12] [13] además de React Native. Por lo tanto, a continuación vamos a explorar y entrar en detalle de algunas de estas alternativas, como Flutter, Xamarin [14] que posiblemente sean las más conocidas y otras tecnologías emergentes]:

Flutter

- Flutter [15] es un framework de desarrollo de aplicaciones móviles de código abierto creado por Google. Utiliza el lenguaje de programación Dart y ofrece un conjunto completo de widgets personalizables para crear interfaces de usuario atractivas y de alto rendimiento. Flutter permite a los desarrolladores compilar una única base de código para iOS y Android, lo que facilita el desarrollo de aplicaciones multiplataforma. Su enfoque en el rendimiento, la productividad y la belleza visual lo convierte en una opción atractiva para muchos desarrolladores, y por estas razones la convierte en una de las más conocidas [16] [17].

Xamarin

- Xamarin [18] es una plataforma de desarrollo de aplicaciones móviles propiedad de Microsoft. Utiliza el lenguaje de programación C# y permite a los desarrolladores crear aplicaciones nativas para iOS, Android y Windows desde una única base de código. Xamarin ofrece una amplia gama de herramientas y bibliotecas para simplificar el desarrollo de aplicaciones móviles, así como una integración estrecha con las plataformas y servicios de Microsoft. Su capacidad para compartir código entre las plataformas [20] y su sólido soporte empresarial lo hacen popular entre los desarrolladores corporativos [19].



Year introduced	2011	2015	2018
Backed by	Microsoft	Facebook	Google
Presentation language	XAML and/or xamarin.forms	Proprietary but looks like JSX	Dart
Procedural Language	C#	JavaScript	Dart
Still need to know some truly native development	Very high	High	Low
Recent popularity trend	Slightly decreasing	Increasing	Increasing
Cost	Teams of > 5 must buy a license to Visual Studio	Free	Free

***Figura 9**: Comparativa entre Xamarin, React Native y Flutter. Los 3 lenguajes más importantes en aplicaciones móviles.*

Aquí [Figura 9] presentamos una comparativa entre los tres lenguajes con más potencial para el desarrollo de aplicaciones móviles: React Native, Flutter y Xamarin [20]. Cada uno de ellos tiene sus propios puntos fuertes y consideraciones clave.

React Native se destaca por su equilibrio entre facilidad de aprendizaje y potencia. Aunque puede ser un poco más difícil de aprender que Flutter, ofrece una curva de aprendizaje suave para aquellos familiarizados con JavaScript, un lenguaje de programación ampliamente utilizado y similar a otros aprendidos en el curso de grado. Además, la popularidad de JavaScript está en constante aumento, lo que proporciona una amplia comunidad de desarrolladores, recursos y bibliotecas de código abierto para respaldar el desarrollo en React Native.

Flutter, por otro lado, se destaca por su enfoque en la creación de interfaces de usuario altamente personalizables y estéticamente atractivas. Utiliza el lenguaje de programación Dart, que, aunque puede ser menos conocido que JavaScript, ofrece una sintaxis clara y concisa que facilita el desarrollo. La principal ventaja de Flutter es su capacidad para compilar código en aplicaciones nativas altamente optimizadas para iOS y Android, lo que garantiza un rendimiento y una experiencia de usuario consistentes en todas las plataformas.

Xamarin, desarrollado por Microsoft, es otra opción popular para el desarrollo de aplicaciones móviles. Utiliza el lenguaje de programación C#, que es ampliamente utilizado en el desarrollo de software empresarial y sistemas Windows. Xamarin permite a los desarrolladores compartir una sola base de código para crear aplicaciones nativas para iOS, Android y Windows, lo que ofrece una gran flexibilidad y eficiencia en el desarrollo.

Es decir, mientras que React Native ofrece un equilibrio entre facilidad de aprendizaje y potencia, Flutter destaca en la creación de interfaces de usuario altamente personalizables y Xamarin brinda flexibilidad para desarrolladores familiarizados con C#. Finalmente he optado por React Native debido a su equilibrio entre facilidad de aprendizaje y potencia. Además, el hecho de tener algunas nociones en React, un lenguaje similar, facilita el proceso de aprendizaje y desarrollo. Considerando sus características, como su creciente popularidad y la amplia comunidad de desarrolladores, se percibe como la mejor opción para el proyecto. Además, se ve como un desafío interesante y muy enriquecedor para el desarrollo personal, lo que motiva a abordar este proyecto con entusiasmo y tratarlo como un reto.

Otras Tecnologías Emergentes

Además de Flutter y Xamarin, existen otras tecnologías emergentes que los desarrolladores pueden considerar para el desarrollo de aplicaciones móviles:

- NativeScript: Un framework de desarrollo de aplicaciones móviles de código abierto que utiliza JavaScript y TypeScript para crear aplicaciones nativas para iOS y Android [21].
- Ionic [22]: Un framework de desarrollo de aplicaciones móviles que utiliza tecnologías web estándar como HTML, CSS y JavaScript para crear aplicaciones multiplataforma con un aspecto y un rendimiento nativos.
- SwiftUI: Un framework de interfaz de usuario declarativa para el desarrollo de aplicaciones iOS nativas utilizando Swift [23], el lenguaje de programación de Apple.
- Kotlin Multiplatform: Una tecnología [24] que permite a los desarrolladores compartir código entre las plataformas iOS y Android utilizando Kotlin, un lenguaje de programación desarrollado por JetBrains y respaldado por Google. Posiblemente será la siguiente candidata por su evolución constante.

Estas son solo algunas de las alternativas disponibles para el desarrollo de aplicaciones móviles. Cada una tiene sus propias ventajas y desventajas, y la elección de la tecnología adecuada dependerá de los requisitos del proyecto, las habilidades del equipo de desarrollo y las preferencias del desarrollador.

2.2.3 Expo

Expo es una plataforma de código abierto que simplifica el proceso de desarrollo de aplicaciones móviles utilizando React Native. Ofrece un conjunto de herramientas y servicios que permiten a los desarrolladores crear, desarrollar y distribuir aplicaciones móviles de forma rápida y eficiente [25].

Entre sus características podemos destacar las más importante, y así entender el porqué de la elección de Expo para desarrollar Visionova:

- **Desarrollo Rápido:** Expo proporciona un entorno de desarrollo rápido que elimina la necesidad de configurar y mantener complejas herramientas de desarrollo. Los desarrolladores pueden comenzar a trabajar en su aplicación de inmediato sin necesidad de instalar o configurar software adicional.
- **Componentes y APIs Integradas:** Expo incluye una amplia gama de componentes y APIs integradas que permiten a los desarrolladores acceder a funcionalidades nativas del dispositivo [26], como la cámara, el GPS y los sensores de movimiento, con solo unas pocas líneas de código.
- **Herramientas de Depuración y Pruebas:** Expo ofrece herramientas de depuración y pruebas que facilitan la identificación y corrección de errores en el código de la aplicación. Los desarrolladores pueden realizar pruebas en dispositivos reales o emuladores para garantizar un rendimiento óptimo en diferentes entornos [27].
- **Servicios de Implementación y Distribución:** Expo simplifica el proceso de implementación y distribución de aplicaciones móviles, permitiendo a los desarrolladores generar archivos de aplicación nativos (APK para Android, IPA para iOS) con un solo comando. Además, Expo [28] ofrece servicios de alojamiento para la distribución de aplicaciones en las tiendas de aplicaciones de Google Play y App Store.

Alternativas a Expo

Hemos hablado bastante sobre lo genial que és Expo, ya que es una opción popular para el desarrollo de aplicaciones con React Native, pero existen otras alternativas que los desarrolladores pueden considerar:

- **React Native CLI:** Utilizar React Native CLI (Command Line Interface) [28] proporciona a los desarrolladores un mayor control sobre el proceso de

desarrollo, permitiendo una configuración más personalizada y acceso directo al código nativo subyacente. Sin embargo, esto puede requerir más tiempo y esfuerzo para configurar y mantener.

- **Código Puro:** Algunos desarrolladores optan por desarrollar aplicaciones móviles utilizando código nativo [29] [30] (Swift para iOS, Kotlin para Android) en lugar de utilizar frameworks como React Native. Si bien esto puede proporcionar un rendimiento óptimo y un control completo sobre la aplicación, también puede ser más complejo y requerir habilidades de programación específicas para cada plataforma.

2.2.4 Combinación React Native & Expo

Está ha sido mi elección para desarrollar nuestra app, ya que React Native junto con Expo es una combinación poderosa para el desarrollo de aplicaciones móviles. React Native es un marco de desarrollo de aplicaciones móviles que utiliza JavaScript y permite a los desarrolladores crear aplicaciones nativas para iOS y Android utilizando una base de código compartida [31]. Por otro lado, Expo es una plataforma y conjunto de herramientas que simplifica el desarrollo, la compilación y el despliegue de aplicaciones React Native al proporcionar una serie de características y servicios útiles como ya hemos comentado más en detalle previamente.

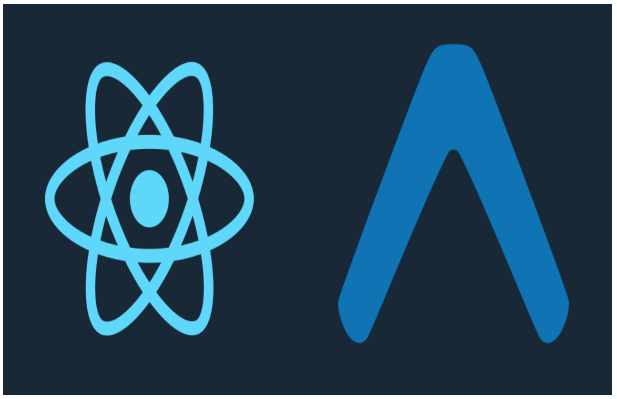


***Figura 10**: Logos de React Native y Expo.*

Una de las principales ventajas de utilizar React Native con Expo es su enfoque en la productividad y la facilidad de uso. Expo elimina la necesidad de configurar y gestionar entornos de desarrollo complejos, lo que permite a los desarrolladores como es mi caso en centrarnos en la escritura de código y en la creación de experiencias de usuario excepcionales [32]. Además, Expo proporciona una amplia gama de API y componentes predefinidos que simplifican tareas comunes como la gestión de notificaciones push, el acceso a la cámara del dispositivo y la autenticación de usuarios [33].

Otra ventaja importante de utilizar React Native con Expo es su capacidad para acelerar el ciclo de desarrollo y pruebas. Expo ofrece herramientas integradas para la vista previa en tiempo real de aplicaciones en dispositivos físicos y emuladores, lo que facilita la iteración rápida y la depuración de problemas [34]. Además, Expo simplifica el proceso de distribución de aplicaciones al proporcionar servicios de construcción y publicación que permiten a los desarrolladores generar paquetes de aplicaciones listos para su distribución en las tiendas de aplicaciones de iOS y Android.

3. Diseño previo

3.1 Análisis DAFO

Un análisis DAFO es un método importante de análisis empresarial, ya que permite un examen exhaustivo de las características internas y externas que afectan su crecimiento y éxito. Este enfoque proporciona una visión integral al identificar los aspectos positivos y negativos del proyecto, proporcionando así una base sólida para la toma de decisiones estratégicas.
La siguiente figura {número} muestra el análisis DAFO del proyecto Visionova, esta investigación se organiza en diferentes categorías relacionadas con diferentes aspectos del proyecto, las cuales se dividen en categorías específicas. A través de este estudio, se examinan en detalle las fortalezas, debilidades, oportunidades y amenazas que enfrenta Visionova en su camino hacia la implementación y el éxito.
Al comprender mejor estas categorías y sus implicaciones, puede desarrollar una estrategia eficaz para maximizar las fortalezas, abordar las debilidades, explotar las oportunidades y mitigar las amenazas. Esta investigación proporcionará una valiosa orientación para un mayor desarrollo y mejora de Visionova, asegurando relevancia, impacto y sostenibilidad a largo plazo.

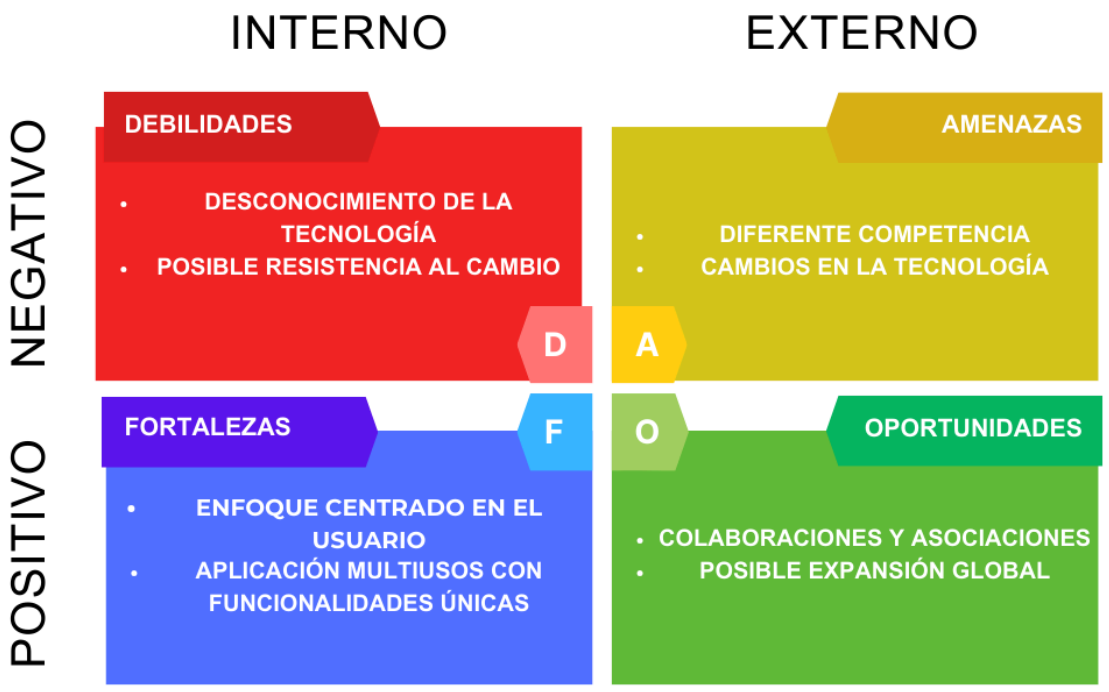


***Figura 11**: Representación de análisis DAFO. Se destaca los puntos de cada apartado, mostrando si són internos, externos, negativos o positivos.*

Para entrar más en detalle de este análisis vamos a comentarlo más en profundidad, explicando cada punto de la Figura {número}, dónde primero comentaremos los puntos negativos de forma interna y externa, y posteriormente los puntos positivos y de la misma forma, internos y externos.

Debilidades

- Desconocimiento de la tecnología:
  Una posible y potencial debilidad de Visionova es la falta de habilidades técnicas entre algunos usuarios, especialmente aquellos que no saben cómo utilizar aplicaciones móviles o dispositivos inteligentes. Esto puede obstaculizar la aceptación de la aplicación y requerir esfuerzos adicionales de educación y capacitación para garantizar que los usuarios comprendan cómo usarla.
- Posible resistencia al cambio:
  Algunas personas con baja visión pueden estar acostumbradas a los métodos de navegación tradicionales y es posible que no puedan utilizar las nuevas tecnologías. La resistencia al cambio puede surgir del miedo a lo desconocido o de la comodidad con las opciones existentes.

Amenazas

- Diferente competencia:
  La presencia de diferentes competidores en el mercado de aplicaciones para personas con discapacidad visual representa una amenaza para Visionova. La competencia puede dificultar la diferenciación de la aplicación y requerir estrategias efectivas de marketing y posicionamiento en el mercado para destacar entre la multitud. Aunque al ser aplicaciones para usuarios más específicos la competencia no es tan grande como en otros campos.
- Cambios en la tecnología:
  Los rápidos avances tecnológicos pueden representar un riesgo para Visionova si la aplicación no se mantiene al día con las últimas tendencias y desarrollos. Los cambios en la tecnología pueden hacer que las funciones existentes queden obsoletas o sean menos efectivas, y pueden afectar el rendimiento y la competitividad de la aplicación en el mercado.

Fortalezas

- Enfoque centrado en el usuario:
  Una de las fortalezas clave de Visionova es su enfoque centrado en el usuario, especialmente diseñado para satisfacer las necesidades y desafíos de las personas con discapacidad visual. Esto se refleja en la cuidadosa atención a la accesibilidad, la facilidad de uso y los complementos que ayudan a los usuarios reales.
- Aplicación multiusos con funcionalidades únicas:
  Nuestra app se destaca como una aplicación multipropósito con funcionalidades únicas que van más allá de las soluciones tradicionales. Agregar funciones como una mini red social para compartir técnicas y experiencias agrega valor adicional a la aplicación y la hace más interesante y útil para los usuarios.

Oportunidades

- Colaboraciones y asociaciones:

Existe la oportunidad de establecer colaboraciones y asociaciones estratégicas con organizaciones que apoyan a las personas con discapacidad visual, instituciones académicas, empresas de tecnología y otras partes interesadas. Esta colaboración puede ayudar a mejorar y promover la aplicación, ampliar su alcance y aumentar su impacto positivo en la comunidad.

- Posible expansión global:
  Hay una gran oportunidad de expandirse globalmente, llegando a personas con discapacidad visual en diferentes países y culturas. Adaptar la aplicación para satisfacer las necesidades específicas de diferentes comunidades y trabajar estrechamente con las comunidades puede abrir nuevas oportunidades de mercado y aumentar la cantidad de usuarios de Visionova incluso hablando a nivel internacional.

Aprovecharemos las debilidades y posibles amenazas para ayudar a fortalecer nuestros mejores puntos y utilizar las oportunidades para así conseguir nuestro objetivo de crear una aplicación inclusiva y fácil de usar, empoderando a todas las personas con discapacidad visual para que puedan utilizarla de manera efectiva y sin obstáculos, siempre en primera linea de la tecnología y intentando diferenciarnos del resto de mercado, creando una aplicación con diferentes utilidades, desde funciones específicas para las personas ciegas como funciones que pueden ser útiles para cualquier usuario.

3.2 Obtención de requisitos

Aquí se enumeran los diferentes requisitos funcionales y no funcionales que se han obtenido para este proyecto, además de los respectivos casos de uso.

3.2.1 Requisitos funcionales

- RF-USER-01: El usuario podrá registrarse en la aplicación proporcionando información personal básica y creando una contraseña segura.

- RF-USER-02: El usuario podrá mediante un sistema de inicio de sesión acceder a sus cuentas utilizando sus credenciales previamente registradas.

- RF-SYSTEM-01: El sistema verificará la validez de los datos ingresados durante el registro y el inicio de sesión, incluyendo la longitud y complejidad de la contraseña.

- RF-USER-03: El usuario podrá cerrar sesión en cualquier momento.

- RF-USER-04: El usuario podrá escanear imágenes para recibir retroalimentación auditiva y visual sobre su contenido.

- RF-GUEST-01: El invitado podrá escanear imágenes para recibir retroalimentación auditiva y visual sobre su contenido.

- RF-SYSTEM-02: El sistema dará la información a través de voz de la imagen escaneada.

- RF-SYSTEM-03: El sistema dará la información a través de un texto de la imagen escaneada.

- RF-USER-05: Los usuarios podrán grabar sus propias rutas.

- RF-USER-06: Los usuarios podrán guardar sus propias rutas dentro de la aplicación para futuras referencias y navegación.

- RF-USER-07: Los usuarios podrán ver rutas de diferentes usuarios.

- RF-GUEST-02: Los invitados podrán ver rutas de diferentes usuarios y usarlas.

- RF-USER-08: El usuario tendrá un perfil personal donde podrán acceder y gestionar sus rutas guardadas.

- RF-USER-09: El usuario podrá ver sus seguidores y a quién sigue.

- RF-USER-10: Los usuarios podrán seguir a otros usuarios dentro de la aplicación

- RF-USER-11: Opcionalmente, se permitirá a los usuarios añadir rutas a una lista de favoritos para un acceso rápido y fácil en el futuro.

3.2.2 Requisitos no funcionales

3.2.2.1 Requisitos de aspecto

- RNF-INT-01: La interfaz de usuario se diseñará siguiendo la paleta de colores especificada para garantizar una experiencia visual agradable y consistente.

- RNF-INT-02: La aplicación será diseñada de forma intuitiva y fácil de usar, con una navegación clara y accesible para usuarios de todos los niveles de habilidad.

3.2.2.2 Requisitos de funcionamiento

- RNF-SC-01: Se implementarán medidas de seguridad robustas para proteger la privacidad y los datos de los usuarios, cumpliendo con los estándares de seguridad de la industria.

3.2.2.3 Requisitos operacionales

- RNF-EF-01: La aplicación será compatible con dispositivos móviles con sistemas operativos iOS y Android.

- RNF-ENT-01: La aplicación integrará tecnologías de reconocimiento de imágenes y voz para ofrecer retroalimentación auditiva y visual sobre el contenido de las imágenes escaneadas.

- RNF-AR-01: La aplicación estará vinculada a la API de Google Maps Platform para acceder a datos de mapas y servicios de geolocalización.

- RNF-AR-02: Se utilizarán APIs relacionadas con el reconocimiento de imágenes y voz para proporcionar funcionalidades clave dentro de la aplicación.

- RNF-AR-10: La aplicación incluirá un video explicativo dentro de la aplicación para guiar a los usuarios sobre cómo utilizar sus funciones principales, utilizando tanto instrucciones de voz como imágenes.

3.3 Casos de uso

En esta sección, exploraremos los diferentes escenarios de interacción que un usuario puede tener con la aplicación, representados a través de casos de uso. Estos casos de uso describen las diversas acciones que un usuario puede llevar a cabo dentro de la aplicación y están vinculados a los requisitos correspondientes. En el diagrama de casos de uso, que se muestra en la Figura 14, se visualizan de manera gráfica todas estas interacciones.

Los casos de uso, identificados por códigos específicos, proporcionan una visión detallada de cómo los usuarios interactúan con la aplicación en diferentes situaciones. Además, cada caso de uso está asociado con los requisitos funcionales correspondientes, lo que asegura que todas las funcionalidades de la aplicación estén respaldadas por un caso de uso específico [35].

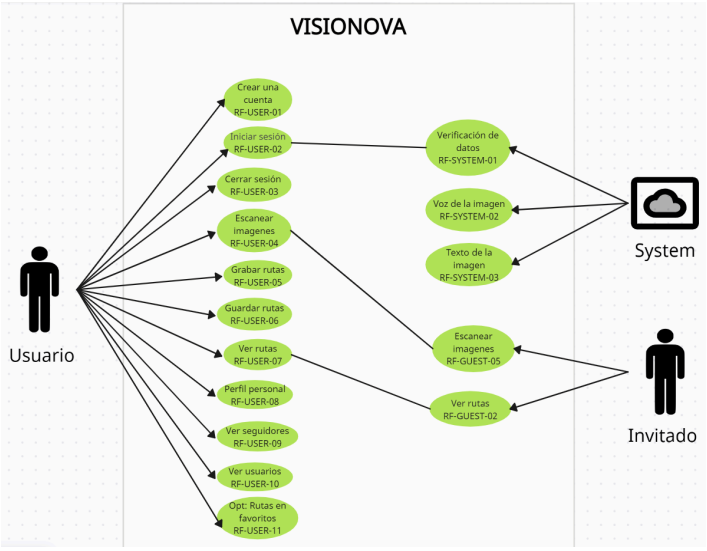


***Figura 12**: Diagrama de los casos de uso.. Se ve representado todas las historias de usuario, tanto si son para usuarios, invitados o del propio sistema, y si tienen alguna relación entre ellas*

Para mejor visualización, también se podrá encontrar en el Anexo.

CU-USER-01	Registro en la aplicación
Descripción	El usuario crea una nueva cuenta en la aplicación
Precondición	El usuario no está registrado en la aplicación
Postcondición	El usuario está registrado y puede iniciar sesión en la aplicación
Requisitos relacionados	RF-USER-01, RF-SYSTEM-01, RNF-SC-01

Escenario principal
1. El usuario abre la aplicación Visionova. 2. Se le presenta al usuario la opción de registro. 3. El usuario completa el formulario de registro, proporcionando información personal y creando una contraseña segura. 4. La aplicación verifica la validez de los datos ingresados y registra al usuario. 5. Se muestra un mensaje de confirmación de registro. 6. El usuario puede iniciar sesión con las credenciales recién creadas.
Escenario alternativo
4..1. La contraseña no cumple con los requisitos mínimos de seguridad. 4..2. La aplicación muestra un mensaje de error y solicita al usuario que elija una contraseña más segura. 4..3. El usuario proporciona una nueva contraseña y se registra exitosamente.

***Figura A.1**: RF-USER-01. Caso de uso de registro en la aplicación*

CU-USER-02	Inicio de sesión en la aplicación
Descripción	El usuario inicia sesión en la aplicación utilizando sus credenciales previamente registradas
Precondición	El usuario ha creado una cuenta en la aplicación
Postcondición	El usuario está dentro de la aplicación con una sesión iniciada
Requisitos relacionados	RF-USER-02, RF-SYSTEM-01, RNF-SC-01

Escenario principal
1. El usuario abre la aplicación Visionova. 2. La aplicación le indica al usuario que debe iniciar sesión con Google. 3. Se le muestran al usuario las diferentes cuentas de Google con las que puede iniciar sesión. 4. El usuario selecciona una cuenta. 5. El usuario ha iniciado sesión. 6. El usuario es redirigido a la pantalla principal de la aplicación.
Escenario alternativo
5.1. No se ha conseguido iniciar sesión. 5.2. El usuario es redirigido a la página de inicio.

***Figura A.2**: RF-USER-02. Caso de uso de inicio de sesión en la aplicación*

CU-USER-04	Escanear imagenes
Descripción	Escaneo de imágenes para recibir retroalimentación auditiva y visual
Precondición	El usuario está dentro de la aplicación.
Postcondición	El usuario recibe retroalimentación auditiva y visual sobre el contenido de la imagen escaneada.
Requisitos relacionados	RF-USER-04, RF-SYSTEM-02, RF-SYSTEM-03, RNF-ENT-01

Escenario principal
1. El usuario selecciona la opción de escanear imágenes en la pantalla principal de la aplicación Visionova. 2. La cámara del dispositivo se activa y el usuario apunta hacia la imagen que desea escanear. 3. La aplicación captura la imagen y la procesa utilizando tecnologías de reconocimiento de imágenes y voz. 4. La aplicación proporciona retroalimentación auditiva sobre el contenido de la imagen escaneada. 5. La aplicación muestra un texto descriptivo del contenido de la imagen en la pantalla.

Escenario alternativo
4.1. La aplicación no puede reconocer el contenido de la imagen. 4.2. La aplicación informa al usuario de que no se pudo identificar el contenido de la imagen y sugiere intentarlo nuevamente.

***Figura A.4**: RF-USER-04. Caso de uso de la funcionalidad de escanear imágenes.*

Se muestra la figura del A.4 en lugar de la figura A.3 para mostrar un caso de uso diferente a los primeros, donde se puede apreciar que ahora es una de las principales funcionalidades de Visionova.

Los siguientes casos de usos incluido el A.3 aparecen en el anexo.

3.4 Paleta de colores e iconos

La elección de colores en una aplicación es crucial porque afecta directamente la usabilidad, accesibilidad, y la experiencia del usuario. Los colores no solo definen la estética de la aplicación, sino que también influyen en la legibilidad, la percepción de la información, y la facilidad de uso, especialmente para personas con visibilidad reducida, que en caso de Visionova, es algo muy importante a tener en cuenta, ya que la aplicación está enfocada en que cualquier tipo de usuario pueda utilizarla [36].

Hay varias razones para entender la importancia de este apartado entre varias, las más importantes són:

Legibilidad: Colores adecuados aseguran que el texto y otros elementos de la interfaz sean fácilmente legibles. Contrastes fuertes entre el fondo y el texto son esenciales para que el contenido sea accesible y más en la aplicación que estoy desarrollando.

- **Accesibilidad**: Aproximadamente 8% de los hombres y 0.5% de las mujeres tienen algún tipo de daltonismo. Usar combinaciones de colores accesibles ayuda a garantizar que todos los usuarios puedan utilizar la aplicación efectivamente como he comentado anteriormente.
- **Consistencia**: Un esquema de colores bien pensado ayuda a mantener una interfaz de usuario consistente. Esto mejora la experiencia del usuario, haciéndola más intuitiva y agradable.
- **Enfoque y Prioridad:** Los colores pueden guiar a los usuarios hacia las áreas más importantes de la pantalla. Los colores más brillantes y llamativos pueden destacar botones o acciones importantes, mientras que los colores más neutros pueden ser usados para los elementos de fondo, y por lo tanto se ha seguido una guía donde todos los botones tienen el mismo color, exceptuando si son para iniciar sesión por ejemplo que tiene un color distinto, así dándole la prioridad que se merece.
- **Estado Emocional:** Los colores pueden afectar el estado emocional de los usuarios. Colores cálidos como el rojo o naranja pueden ser energizantes y motivadores, mientras que colores fríos como el azul y el verde pueden

ser calmantes y relajantes. Por eso para Visionova se ha escogido este último tipo de colores.

Siguiendo lo comentado, elegir colores fríos y fuertes como azul y lila puede ser una excelente opción, ya que estos colores suelen ser relajantes y menos fatigosos para la vista. Además, los colores fuertes pueden asegurar que los elementos importantes sean fácilmente identificables. Aquí presento un esquema de colores basado en los principios que he estado siguiendo:

Azul (Fuerte):

- Azul Marino (#000080) para los encabezados y botones principales.
- Azul Claro (#87CEEB) para los fondos y áreas de texto secundario.

Lila (Fuerte):

- Lila Oscuro (#8A2BE2) para destacar elementos interactivos.
- Lavanda (#E6E6FA) para fondos o áreas con menor importancia.

Contrastes:

- Texto Blanco (#FFFFFF) sobre fondos oscuros.
- Texto Negro (#000000) sobre fondos claros.

Colores de Soporte:

- Verde Turquesa (#40E0D0) para mensajes de éxito o confirmaciones.
- Naranja (#FFA500) para advertencias o elementos de atención.

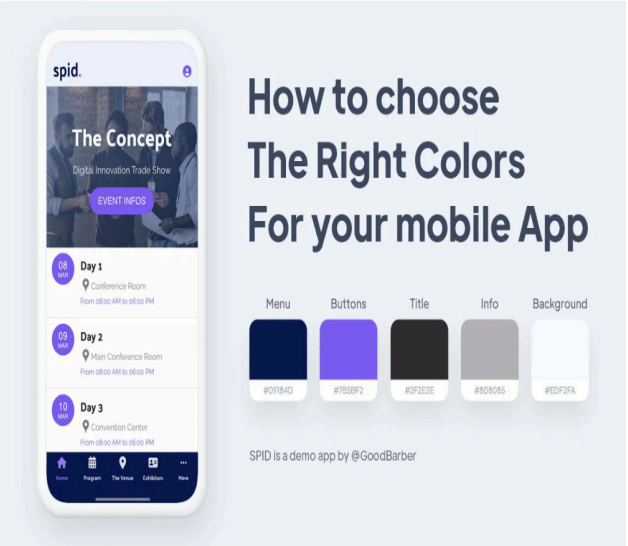


***Figura 13**:: Guía de colores empleados.*

Si nos enfocamos a personas con visibilidad reducida, ya que es algo que a diferencia de otras aplicaciones, yo si debo diseñar una aplicación accesible para personas con visibilidad reducida, por lo tanto hay que seguir ciertas pautas que van en relación a las comentadas anteriormente. Tener un contraste alto, asegurándome de que haya un contraste suficiente entre el texto y el fondo. La **WCAG (Web Content Accessibility Guidelines)** recomienda una relación de contraste de al menos 4.5:1 para texto normal y 3:1 para texto grande [36.5]..

Tambien debemos evitar combinaciones de colores que son difíciles de distinguir para las personas con daltonismo, como rojo/verde o azul/amarillo. Y de la mano también he usado colores neutrales de fondo más claro con texto oscuro, para así mejorar la legibilidad. Por último y no menos importante, también es importante utilizar indicadores no visuales, logrando así que a parte de los colores, he usado patrones, texturas o iconos para transmitir información importante, el cual es el siguiente punto que trataremos más adelante.

Otro punto es la coherencia, es decir, es esencial que los colores seleccionados sigan un tema coherente para que la aplicación tenga una apariencia uniforme y profesional. La coherencia de los colores facilita la navegación y hace que la aplicación sea visualmente atractiva y funcional.
Alguna de las cosas que he utilizado para mantener la consistencia temática es tener una paleta limitada, ya que limitando mi paleta a 3-5 colores principales evitamos sobrecargar visualmente a los usuarios. Así como tener una aplicación consistente, utilizando los colores de manera consistente en toda la aplicación para elementos similares (por ejemplo, todos los botones de acción en azul marino y en cambio los de Iniciar Sesion, o Log Out de color lila). Y por último y asegurarme de la elección he realizado pruebas de accesibilidad de colores utilizando herramientas como Contrast Checker para asegurarte de que los colores sean accesibles para todos los usuarios.

Para seguir la coherencia hemos seguido temas como:<

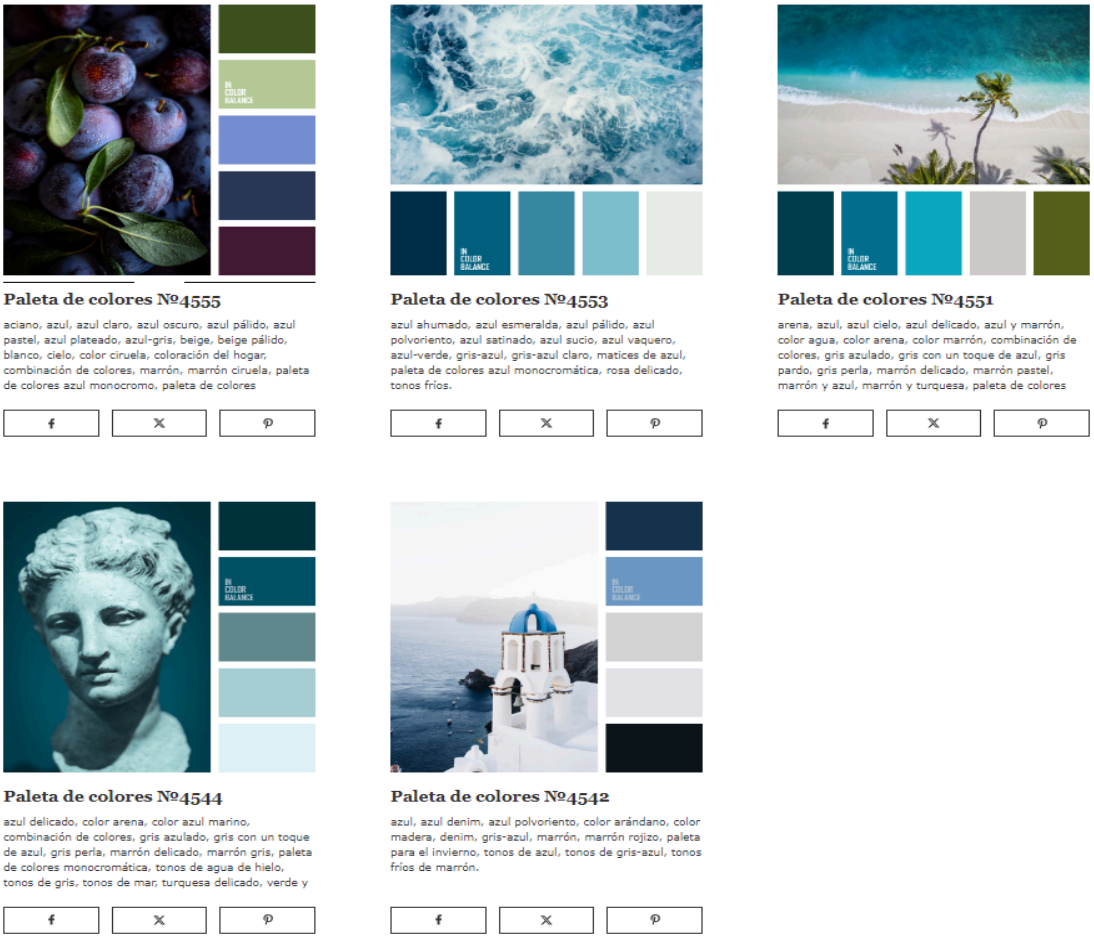


***Figura 14**: Posibilidad de diferentes temas de colores. De esta forma se ha seguido siempre una coherencia entre los colores de la aplicación*

Mezclando varias coherencias pero siempre siguiendo un patrón, teniendo en cuenta los puntos anteriormente destacados y el caso de la visibilidad reducida y la accesibilidad en Visionova [37]. Ya que la elección de colores no solo mejora la estética de la aplicación sino que también juega un papel crítico en la accesibilidad y usabilidad,

especialmente para los usuarios comentados con visibilidad reducida. Los colores fríos y fuertes que he elegido, aplicados de manera consistente y con un buen contraste, hacen que mi aplicación sea atractiva, funcional y accesible.

Iconos

En el desarrollo de aplicaciones móviles, los iconos juegan un papel crucial en la interfaz de usuario (UI). No solo mejoran la estética de la aplicación, sino que también facilitan la navegación y la comprensión de las funcionalidades, especialmente para usuarios con discapacidad visual. En Visionova, se ha optado por utilizar una amplia gama de iconos de la biblioteca Material Community Icons. Esta elección no solo se ha basado en la variedad y calidad de los iconos disponibles, sino también en la facilidad de integración con React Native y Expo.
Entre diferentes opciones se optó finalmente por Material Community Icons, la cual es una biblioteca de iconos que se basa en los principios de diseño de Material Design, desarrollados por Google. Esta biblioteca proporciona más de 5,000 iconos de alta calidad que cubren una amplia variedad de categorías y casos de uso. Al utilizar esta biblioteca en Visionova, hemos podido garantizar una experiencia de usuario coherente y atractiva.
En cuanto la integración de Material Community Icons en una aplicación React Native como es Visionova es relativamente sencilla, gracias a la excelente documentación y soporte de la comunidad. Los iconos se pueden importar y utilizar directamente en los componentes de React Native, lo que facilita su implementación en diversas partes de la aplicación.
La integración se realiza mediante la librería *react-native-vector-icons,* que es compatible con Expo, permitiendo un desarrollo más rápido y eficiente.En Visionova, los iconos de Material Community Icons se utilizan en diversas pantallas y funcionalidades para mejorar la accesibilidad y la usabilidad de la aplicación.

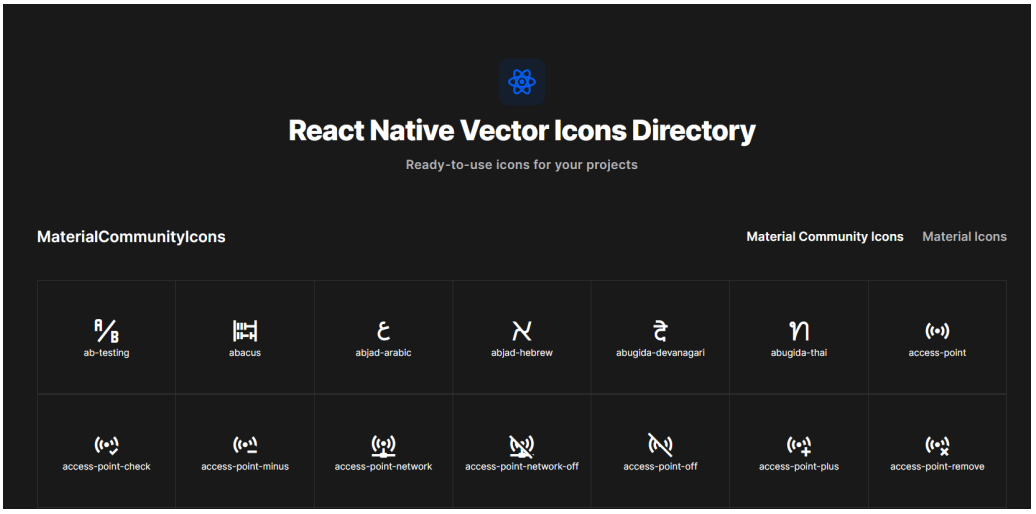


***Figura 15**: Web de MaterialCommunityIcons. Aquí se muestran los diferentes iconos que se pueden ver en Visionova*

4. Metodología y Desarrollo

4.1. ¿Qué es una API?

En el desarrollo de una aplicación móvil, especialmente una como la que estoy desarrollando que tiene un propósito tan significativo como Visionova, la selección de las API's adecuadas es algo crucial para el éxito del proyecto y de los objetivos que tenía en mente. Visionova, diseñada enfocada para personas con discapacidad visual a pesar de que cualquier usuario la puede utilizar, presenta una serie de funcionalidades complejas y específicas, que requieren un manejo adecuado de datos, multimedia y geolocalización. Implementar API's desde cero en este contexto ha sido una experiencia muy importante para mi, pero sobre todo un desafío técnico considerable, ya que nunca antes había implementado API's desde 0, eligiendo las más adecuadas, y configurando todo. Antes de adentrarnos en estas más específicas que he utilizado, es importante comprender el panorama general de las API's disponibles y cómo cada una puede aportar valor a una aplicación móvil de estas características [38].

Las API's (Application Programming Interfaces) son herramientas fundamentales en el desarrollo de aplicaciones modernas. Permiten a los desarrolladores acceder a las funcionalidades de servicios externos sin necesidad de entender su implementación interna. Esto no solo ahorra tiempo, sino que también permite la integración de tecnologías avanzadas en aplicaciones relativamente sencillas. En el contexto de Visionova, varias API's podían haber sido consideradas, por lo tanto primero vamos a explorar las diferentes opciones que podrían haber existido, ya sean por su importancia dentro del mercado o por sus soluciones.

Microsoft Azure Cognitive Services: Una suite robusta que ofrece servicios de visión por computadora, reconocimiento de texto y síntesis de voz. Similar a Google Cloud, proporciona API's para OCR (reconocimiento óptico de caracteres), traducción de texto a voz y análisis de imágenes [39].

Amazon Web Services (AWS) AI Services: Tal vez sea la más famosa entre todas. AWS ofrece una amplia gama de servicios de inteligencia artificial que incluyen Amazon Rekognition para análisis de imágenes, Amazon Polly para conversión de texto a voz, y Amazon Location Service para servicios de geolocalización [40].

IBM Watson: Otra alternativa poderosa que proporciona servicios de análisis de imágenes, conversión de texto a voz y análisis de datos, con un enfoque fuerte en el aprendizaje automático y la inteligencia artificial.

Mapbox: Una alternativa a Google Maps, que proporciona servicios de mapeo y geolocalización con una alta personalización y control sobre la visualización de los mapas.

OpenCV: Una librería de visión por computadora de código abierto, útil para tareas de procesamiento de imágenes y video, aunque requiere una mayor cantidad de trabajo en la implementación comparado con servicios de nube como Google Cloud Vision [41].

Ahora vamos a centrarnos en las soluciones que he tomado para Visionova, entrando en detalle de cómo funcionan para entenderlas a fondo. Tras considerar las diversas opciones disponibles, la elección de las API's para Visionova se basó en varios factores, incluyendo la facilidad de integración, la documentación y soporte disponible, el costo, y la capacidad de satisfacer los requerimientos específicos de la aplicación. A continuación, se describen las API's seleccionadas y por qué fueron elegidas sobre otras opciones.

4.1.1. Diferentes API's en Visionova

Google cloud vision

La API de Google Cloud Vision es un servicio de visión por computadora que permite a los desarrolladores integrar capacidades de reconocimiento de imágenes en sus aplicaciones. Utiliza tecnologías avanzadas de machine learning para analizar el contenido visual y extraer información útil, como texto, objetos, rostros, etc., de imágenes y videos [42].

Algunas de las funciones principales de la API de Google Cloud Vision incluyen:

- **Reconocimiento óptico de caracteres (OCR)**: La API puede identificar y extraer texto de imágenes, incluso en varios idiomas y en diferentes estilos de escritura.
- **Detección de objetos**: Puede identificar objetos y detectar sus ubicaciones en una imagen, así como proporcionar información sobre la confianza de la detección.
- **Detección de rostros**: Puede detectar rostros humanos en imágenes, así como identificar características faciales como ojos, nariz, boca, etc.

- **Detección de contenido inapropiado:** Puede analizar imágenes en busca de contenido inapropiado o sensible, como violencia, desnudez, drogas, etc.
- **Reconocimiento de etiquetas**: Puede asignar etiquetas descriptivas a imágenes para ayudar a categorizar y organizar el contenido visual.

***Figura 16**: Logo de la API Google Vision.*

Y si especificamos más, en cómo he utilizado la API de Google Cloud Vision en mi proyecto, he implementado la función de reconocimiento de texto (OCR) para extraer texto de imágenes capturadas con la cámara de la aplicación. Esto me ha permitido analizar el texto contenido en las imágenes y utilizarlo para diferentes propósitos, como la traducción, el análisis de datos o la automatización de tareas, que en mi caso me centraré en el reconocimiento del texto para posteriormente trasladarlo a voz.
En la implementación, he utilizado la API de Google Cloud Vision junto con la biblioteca Axios para realizar solicitudes HTTP al servicio de visión de Google Cloud. Al capturar

una imagen con la cámara de la aplicación, debo convertir la imagen en formato base64, ya que es el formato en la que se espera que esté y posteriormente la envió a la API de Google Cloud Visión para que realice el reconocimiento de texto. Una vez que recibes la respuesta de la API, procesas los resultados y utilizo la información como me sea necesario.

Text-to-speech

Para la funcionalidad de conversión de texto a voz, esta API proporciona voces naturales y de alta calidad, lo cual es crucial para una aplicación destinada a personas con discapacidad visual. La capacidad de personalizar las voces y su compatibilidad con múltiples idiomas y acentos la hace superior a otras opciones como Amazon Polly o IBM Watson en este contexto [43].

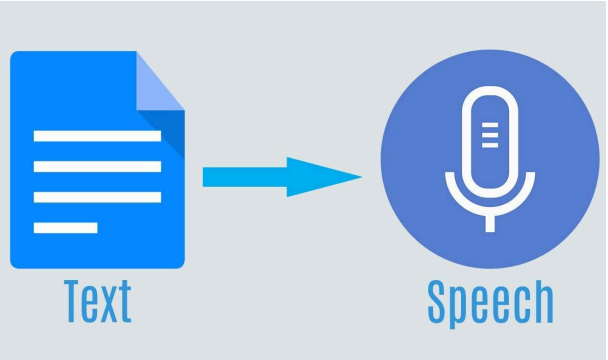


***Figura 17**: Representación visual de la API Text-To-Speech.*

React native maps y expo location API

La API de expo-location se utiliza generalmente para acceder a la ubicación del dispositivo. Proporciona funciones para solicitar permisos de ubicación, obtener la ubicación actual y suscribirse a actualizaciones de ubicación [44].
Estas API's las seleccioné para las funcionalidades de mapeo y geolocalización. React Native Maps proporciona una integración sencilla y eficaz con Google Maps, permitiendo un control detallado sobre los componentes del mapa. La Expo Location API facilita la obtención de la ubicación en tiempo real del usuario, algo esencial para la grabación y seguimiento de rutas.

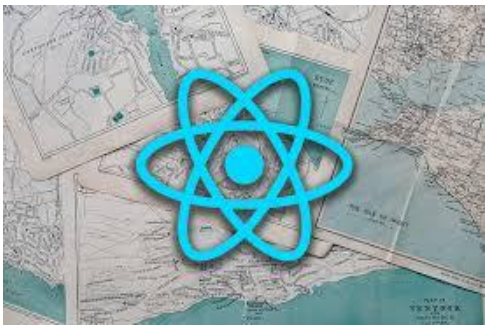


***Figura 18**: Logo de React Native Maps.*

Se podría decir que estas son una de las más importantes, ya que son las que se han utilizado para desarrollar y lograr la funcionalidad de las rutas.

Expo file system API

La API de expo-file-system se utiliza para realizar operaciones de escritura y lectura de archivos en el sistema de archivos del dispositivo. Se utiliza para guardar las coordenadas de la ruta en formato JSON y convertirlas a formato KML para su visualización en Google Maps.
Necesaria para manejar el almacenamiento de archivos localmente en el dispositivo, esta API, por lo tanto como he comentado, permite guardar y recuperar datos de manera eficiente, asegurando que las rutas grabadas y otros datos importantes estén disponibles incluso sin conexión a internet, un dato muy importante, ya que hace que nuestra aplicación de un salto de calidad, permitiendo a los usuarios tener la posibilidad de guardar rutas a pesar de no tener conexión.

React native view shot

Esta API permite capturar vistas de la aplicación como imágenes, lo cual es útil para guardar instantáneas de mapas o cualquier otra vista que el usuario necesite conservar. Centrándonos en Visionova, ha permitido que a la hora de guardar una ruta, también se guarde una instantánea del mapa en ese preciso momento para posteriormente se muestre junto a la ruta.

Firebase

Más adelante hablaremos en profundidad de Firebase, ya que es nuestra base de datos en Visionova, pero no solo se queda en eso, sino que es utilizado para la autenticación de usuarios, almacenamiento de datos y sincronización en tiempo real, Firebase proporciona una solución completa y escalable para la gestión de bases de datos y usuarios [45].

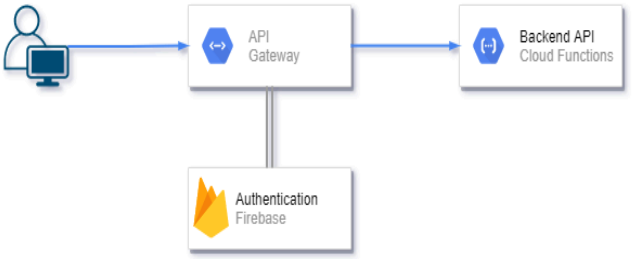


***Figura 19**: Firebase. Proceso de conectividad con el uso de las API's necesarias*

Su integración con React Native es excelente, y la capacidad de Firebase para manejar datos en tiempo real y autenticar usuarios de manera segura es superior a otras soluciones como AWS Amplify en este contexto.

Expo media library, expo audio y axios

Para terminar tenemos 3 que ayudan a facilitar el desarrollo y la configuración de las API's y las funcionalidades que tiene Visionova.

Expo Media Library: Permite el acceso y la manipulación de archivos multimedia almacenados en el dispositivo del usuario, crucial para las funcionalidades relacionadas con la cámara y el almacenamiento de imágenes capturadas [46].

Axios: Una biblioteca de cliente HTTP que facilita las solicitudes a los servidores. Su simplicidad y eficiencia en el manejo de solicitudes HTTP la hacen preferible a alternativas como Fetch API, especialmente en una aplicación con múltiples integraciones de API.

Expo Audio: Esta API facilita la reproducción de audio en la aplicación, esencial para las funcionalidades de conversión de texto a voz y otras necesidades de audio.

En definitiva podemos decir que la implementación de estas API's en Visionova ha sido un proceso meticuloso y deliberado, ya que como he comentado, nunca antes había desarrollado e implementado API's desde cero, configurando una por una.
Cada API fue seleccionada no solo por su capacidad para cumplir con los requisitos funcionales de la aplicación, sino también por su facilidad de integración y soporte robusto. La combinación de Google Cloud Vision y Text-to-Speech, React Native Maps, Expo Location API y otras herramientas específicas de Expo y Firebase, asegura que Visionova no solo es funcionalmente rica, sino también eficiente y fácil de mantener.
Esta experiencia ha hecho que crezca tanto mi comprensión como en habilidades en el uso de API's, y me ha permitido demostrar el poder y la flexibilidad que estas herramientas pueden ofrecer en el desarrollo de aplicaciones móviles complejas.

4.1.2. Comparativa de API's: ¿Por qué las seleccionadas para Visionova son las mejores?

Vamos a explorar porque las API's escogidas son las más idóneas para Visionova, teniendo en cuenta nuestros objetivos, el lenguaje que utilizamos y la disponibilidad para implementarlas. Para ello se hará una comparativa entre las escogidas contra su posible alternativa y veremos cuál tiene más ventajas.

Google Cloud Vision API vs. Microsoft Azure Cognitive Services y Amazon Rekognition

Empezamos con la comparativa entre las 3 más grandes, posiblemente la principal elección y de las más importantes, ya que es uno de los centros de Visionova.
Google Cloud Vision API fue seleccionada principalmente por su alta precisión y la robustez de su reconocimiento óptico de caracteres (OCR).
Aunque Microsoft Azure Cognitive Services y Amazon Rekognition también ofrecen capacidades de OCR, Google Cloud Vision ha demostrado consistentemente una mayor precisión en el reconocimiento de texto en diversos idiomas y condiciones de iluminación. Además, la integración con React Native y la documentación detallada de Google proporcionan una ventaja significativa en términos de facilidad de uso y soporte técnico [47].

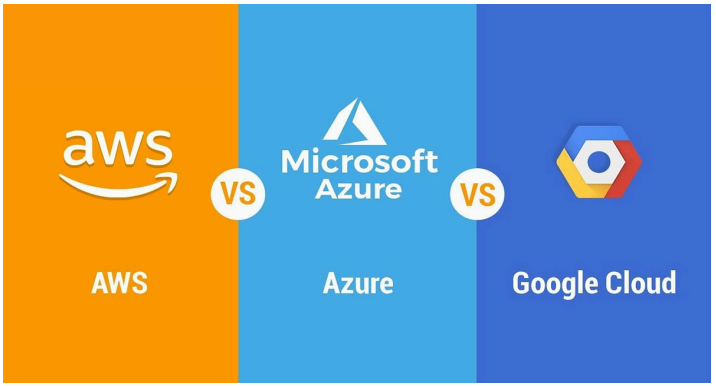


***Figura 20**: Comparativa visual entre AWS, Azure y Google Cloud.*

Google Text-to-Speech API vs. Amazon Polly e IBM Watson Text-to-Speech

La API de Google Text-to-Speech se distingue por la naturalidad y calidad de sus voces, ofreciendo una experiencia auditiva superior que es crucial para usuarios con discapacidad visual. Si bien Amazon Polly y IBM Watson también son opciones sólidas, la personalización de voces y la fluidez natural que proporciona Google Text-to-Speech son superiores, facilitando una mejor comprensión y una experiencia más agradable para los usuarios finales [48].

React Native Maps y Expo Location API vs. Mapbox y AWS Location Service

React Native Maps, en combinación con Expo Location API, proporciona una integración directa y eficaz con Google Maps, que es ampliamente conocido por su precisión y actualización constante de datos de mapas. Mapbox, aunque altamente personalizable, puede requerir más configuración y ajustes para alcanzar el mismo nivel de detalle y funcionalidad proporcionado por Google Maps. AWS Location Service, aunque robusto, no ofrece la misma facilidad de integración y uso con React Native, lo que puede aumentar la complejidad del desarrollo [49].

Expo File System API vs. Open Source Alternatives

Expo File System API ofrece una solución sencilla y eficaz para el manejo de archivos en dispositivos móviles, con una integración directa en el entorno de Expo. Alternativas como React Native FS o Native File System pueden requerir configuraciones adicionales y no ofrecen el mismo nivel de soporte y documentación que Expo File System API, lo que puede ralentizar el desarrollo y aumentar la carga de trabajo del desarrollador [50].

Firebase vs. AWS Amplify y Backendless

Firebase es una solución integral que combina autenticación de usuarios, almacenamiento de datos en tiempo real y sincronización, todo dentro de un ecosistema bien soportado y documentado. AWS Amplify es una alternativa poderosa pero puede ser más compleja de implementar y configurar para aplicaciones móviles. Backendless,

aunque menos conocido, ofrece servicios similares, pero su comunidad y soporte no son tan extensos como los de Firebase, lo que puede ser un inconveniente para desarrolladores que buscan una solución confiable y ampliamente respaldada [51].

Vision AI by Clarifai

También hay que comentar una alternativa interesante y no tan conocida, ya que a diferencia de las comentadas, tiene varios servicios que podrían sustituir a más de uno., estamos hablando de **Vision AI** de **Clarifai**.
Clarifai ofrece capacidades avanzadas de análisis de imágenes y videos, que incluyen no solo OCR, sino también reconocimiento de objetos, análisis de escenas y predicción de contenidos. Aunque no tan popular como Google Cloud Visión o Microsoft Azure Cognitive Services, Clarifai se destaca por su enfoque en inteligencia artificial aplicada y su capacidad de personalización a través de modelos de IA entrenables [52]

Aunque hay alternativas muy potentes como Microsoft Azure Cognitive Services, Amazon Rekognition, Mapbox, y AWS Amplify ofrecen funcionalidades similares, las API's seleccionadas (Google Cloud Vision, Google Text-to-Speech, React Native Maps, Expo Location API, Expo File System API, Firebase, etc.) proporcionan una mejor integración, facilidad de uso, y soporte técnico, lo cual es crucial para el éxito de Visionova.

Además, la exploración de alternativas menos conocidas como Clarifai Visión AI puede abrir nuevas posibilidades para futuras expansiones y personalizaciones de la aplicación.

4.2. Organización y metodología (github, trello)

Es bien sabido que la organización y metodología son pilares esenciales en el desarrollo de software, especialmente cuando se busca crear una aplicación tan innovadora y útil como Visionova. En este proyecto, se ha adoptado la **metodología Agile**, que se caracteriza por su enfoque en la flexibilidad, la colaboración y la entrega continua de valor. Esta metodología ha sido crucial para gestionar el proyecto de manera eficiente, permitiendo una rápida adaptación a los cambios y asegurando un enfoque constante en la satisfacción del usuario final. Para la gestión de tareas y la planificación, se ha utilizado **Trello,** una herramienta que facilita la implementación de técnicas de Scrum, como el uso de **Scrum Poker** para estimaciones. También se ha utilizado como método de organización a nivel de código y proyecto, **Github**, una plataforma online de desarrollo de software que se usa para almacenar, supervisar y trabajar con proyectos de software. Es ampliamente conocida por casi todos los desarrolladores, ya que facilita mucho la organización y el control sobre Visionova. Ahora entraremos más en detalle en cada una de ellas.

4.2.1. Metodología Agile

Agile actualmente es una metodología que promueve la entrega incremental del producto, la colaboración entre equipos multifuncionales y la capacidad de responder rápidamente a los cambios. Por todos estos motivos, actualmente es una de las

metodologías que más se están implementando en el mundo laboral, y que más proyección tiene. En lugar de planificar todo el proyecto de una sola vez, Agile divide el trabajo en iteraciones más pequeñas y manejables, conocidas como sprints. En Visionova, la implementación de Agile comenzó con la creación de un calendario de objetivos, que sirvió como hoja de ruta para el desarrollo y la documentación. Esta planificación inicial ayudó a definir claramente los hitos y a establecer una dirección clara para el proyecto, estimando unas fechas de más o menos cuando podría llevar a cabo cada objetivo.

Para gestionar las tareas, se crearon tickets en Trello tanto para la memoria como para el desarrollo de la aplicación, ya que ambos tenían su columna de "In progress" y "Done" y compartían el "Backlog". Cada ticket representaba una unidad de trabajo específica, con descripciones detalladas y criterios de aceptación claros.
La estimación del tiempo y la dificultad de cada tarea se realizó mediante la técnica de Scrum Poker, asignando puntos de 1, 2, 3, 5 y 8, donde 1 representaba una tarea simple y 8 una tarea compleja. Esta técnica permitió establecer expectativas realistas y priorizar las tareas de manera efectiva, ya que gracias a los story points, podía planificarme de cuantos días estimaba tardar en realizar la tarea.

El seguimiento y la adaptación fueron constantes a lo largo del desarrollo del proyecto. Se realizaron revisiones periódicas de los tickets para ajustar las estimaciones y prioridades según fuera necesario. Esta flexibilidad me permitió responder rápidamente a los desafíos y cambios, manteniendo el proyecto en curso. Además, se buscó retroalimentación constante para mejorar continuamente el producto, lo que es una característica fundamental de la metodología Agile y que ayudó ampliamente al desarrollo de Visionova.

4.2.2. Trello

Trello es una herramienta de gestión de proyectos basada en tableros que permite a los equipos organizar tareas, colaborar y seguir el progreso de manera visual y eficiente. En Visionova, Trello fue esencial para implementar y mantener la metodología Agile.

Tableros de Proyecto:

- **División de Tableros**: Se crearon tableros separados para la memoria y el desarrollo de la aplicación, cada uno con columnas específicas para las diferentes etapas del trabajo: To Do, In Progress, y Done. Esto facilitó la organización y el seguimiento de las tareas en ambos aspectos del proyecto.

Tickets y Listas:

- **Detallado de Tareas**: Cada ticket en Trello contenía descripciones generales, criterios de aceptación el cual más o menos al solo ser yo, no hizo falta apuntarlos en cada tarea y puntos asignados mediante Scrum Poker. Esta claridad me ayudó a comprender lo que esperaba de cada tarea.

- **Etiquetas y Prioridades**: Se utilizaron etiquetas de colores y prioridades para categorizar y priorizar las tareas, asegurando que las tareas más críticas se completaran primero y que cualquier dependencia entre tareas se gestionara adecuadamente. Así a nivel visual podía ayudarme a comprobar cómo avanzaba y que faltaba.

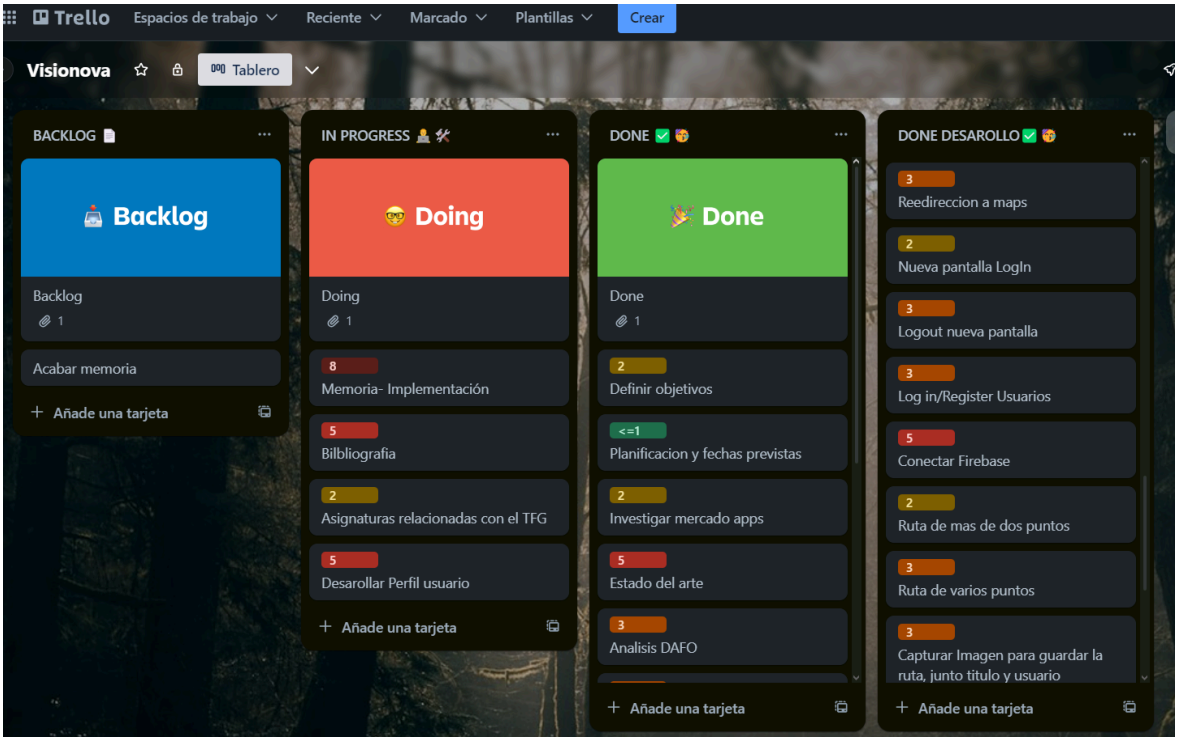


Figura 21: Trello de Visionova.. Se muestra el tablero de Trello con los tickets y los story points dependiendo la dificultad de las tareas

Colaboración y Comunicación:

- **Comentarios y Actualizaciones**: Trello me permite agregar comentarios y actualizaciones directamente en los tickets, facilitando la comunicación sobre el progreso, los obstáculos y las soluciones. Esta funcionalidad fue vital para mantener un flujo de trabajo transparente y colaborativo.

- **Checklists**: Para tareas más complejas, se crearon checklists dentro de los tickets para subdividirlas en pasos más manejables, asegurando un seguimiento detallado del progreso.

Por lo tanto al juntar el apoyo de Trello junto a la metodología Agile se obtuvo unos beneficios más que suficientes siendo una estrategia efectiva para gestionar el desarrollo de Visionova. Este enfoque ha permitido una organización meticulosa del trabajo, una estimación realista del tiempo y esfuerzo necesario, y una capacidad constante para adaptarme a los cambios y mejorar el producto.

4.2.3. Github

Posiblemente otro de los pilares de todo Visionova, ya que sin Github, Visionova no se hubiese podido llevar a cabo A lo largo del desarrollo de Visionova, GitHub ha sido una herramienta esencial para organizar el código, gestionar versiones y garantizar la integridad del proyecto.
Definiéndolo un poco para entender qué es exactamente, podemos decir que GitHub es una plataforma de desarrollo colaborativo que se basa en el sistema de control de versiones Git. Gracias a su robustez y facilidad de uso, ha permitido una gestión eficiente del desarrollo, facilitando la colaboración y asegurando que cada cambio se integre de manera segura y controlada.

Configuración inicial y ramas principales

El primer paso en la gestión del proyecto con GitHub fue la creación del repositorio inicial. Este repositorio comenzó sin ningún contenido, proporcionando una base limpia sobre la cual construir. Para organizar el trabajo, se crearon dos ramas principales: ***production*** y *develop*. La rama ***production*** se utiliza para las versiones finales y estables del software, garantizando que todo lo que se integra en esta rama funcione de manera óptima. Por otro lado, la rama ***develop*** sirve como el entorno de trabajo activo, donde se integran y prueban nuevas funcionalidades antes de promoverlas a ***production***.

Ramas de nuevos desarrollos y correcciones

Para mantener un flujo de trabajo ordenado y minimizar los riesgos de errores, se adoptó una estrategia de ramas específica para nuevas funcionalidades y correcciones de errores.
Cada nueva funcionalidad se desarrollaba en una rama de características, siguiendo la convención de nombres ***feature/..-..*** Esto permitía trabajar de manera aislada, sin afectar el código existente en ***develop*** o ***production***. Por ejemplo, una nueva función de reconocimiento de texto se desarrollaba en ***feature/text-recognition.***
De manera similar, cualquier corrección de errores se manejaba en ramas de corrección, siguiendo la convención ***fix/..-..*** Este enfoque aseguraba que los problemas se resolvieran de manera aislada antes de integrarlos en la rama principal. Por ejemplo, un error en la función de geolocalización se corregía en ***fix/geolocation-bug***. Por lo tanto he ido dividiendo el proyecto en varios lugares a la vez:

- **Rama Production**: Esta rama se reserva para las versiones finales y estables del software. Todo el código en ***production*** está probado y garantizado para funcionar perfectamente. Podemos decir que es donde está la aplicación Visionova en su máximo esplendor.
- **Rama Develop**: Utilizada para el trabajo en curso, ***develop*** actúa como un entorno de integración continua donde se prueban nuevas funcionalidades y correcciones antes de ser promovidas a ***production***
- **Ramas de desarrollo:** Como he comentado anteriormente, aquí tenemos dos tipos de ramas, las fixes para arreglar bugs, pero sobre todo las feature/…-... donde se ha ido desarrollando todas las funcionalidades sin miedo a romper

nada. Algo esencial en mi caso ya que había muchas cosas que jamás antes había hecho.

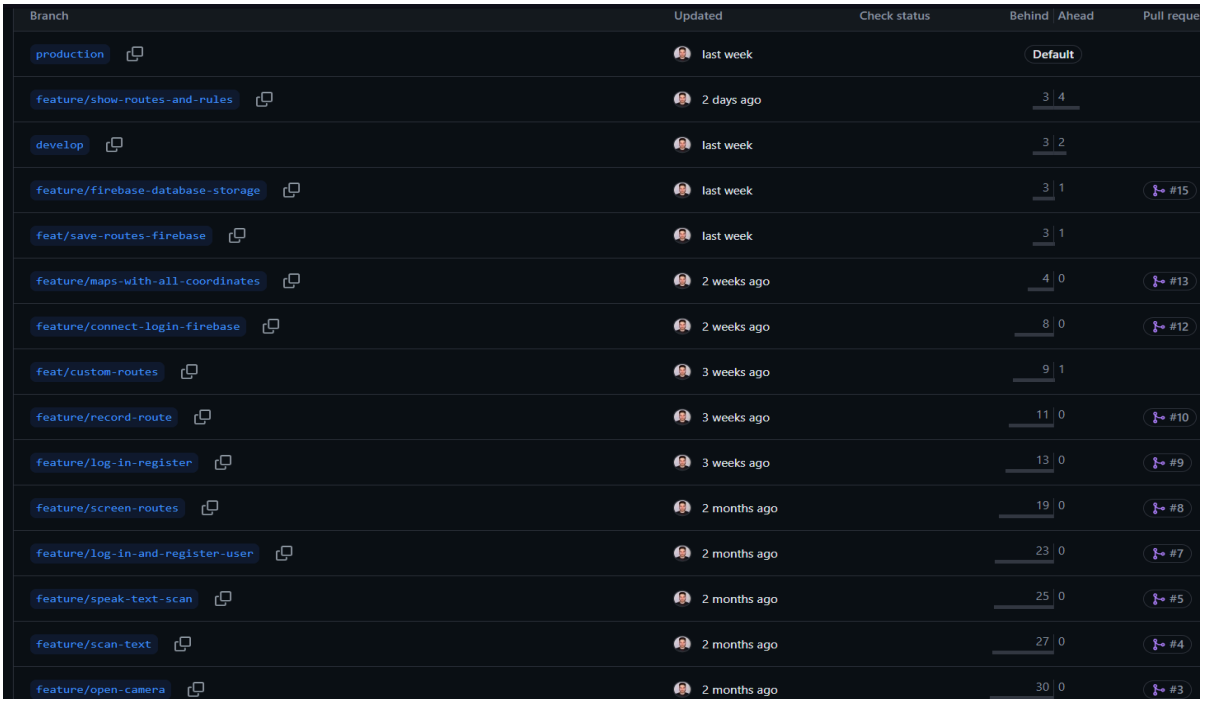


***Figura 22**: Github de Visionova. Visualización de todas las ramas empleadas para el desarrollo de la app*

Proceso de Integración y Merge Requests

El proceso de integración de nuevas funcionalidades y correcciones seguía una serie de pasos bien definidos para asegurar la calidad del código y la estabilidad del proyecto. Primero, se desarrollaba la funcionalidad o se corregía el error en la rama correspondiente, probando los cambios de manera aislada. Una vez completada y probada la funcionalidad, se creaba una merge request hacia la rama *develop*. En esta solicitud, se detallaban los cambios realizados y, en muchos casos, se incluían imágenes o descripciones detalladas para proporcionar contexto adicional sobre los cambios.

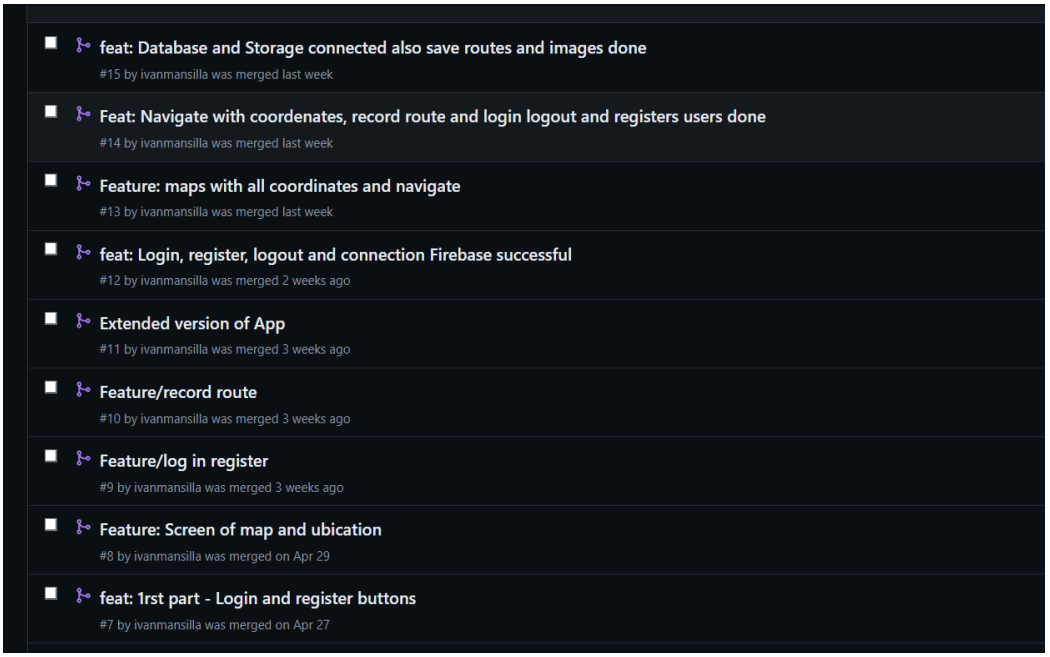


***Figura 23**: Pull request de github. Todas las merge request empleadas, siguiendo la misma metodología*

Antes de aceptar la merge request, se revisaban los cambios y se resolvían posibles conflictos. Aunque los conflictos fueron mínimos gracias a la buena organización que he ido comentando, la revisión era una parte esencial para asegurar que el código nuevo no introdujera errores.
Tras la revisión y resolución de conflictos, los cambios se integraban en la rama ***develop***, permitiendo realizar pruebas adicionales en un entorno más amplio. Una vez que varios cambios se habían integrado y probado en ***develop***, y se tenía una versión estable y funcional, se creaba una merge request hacia ***production***. Este paso final aseguraba que solo las versiones probadas y estables llegaran a la rama de producción, garantizando la calidad del software lanzado.

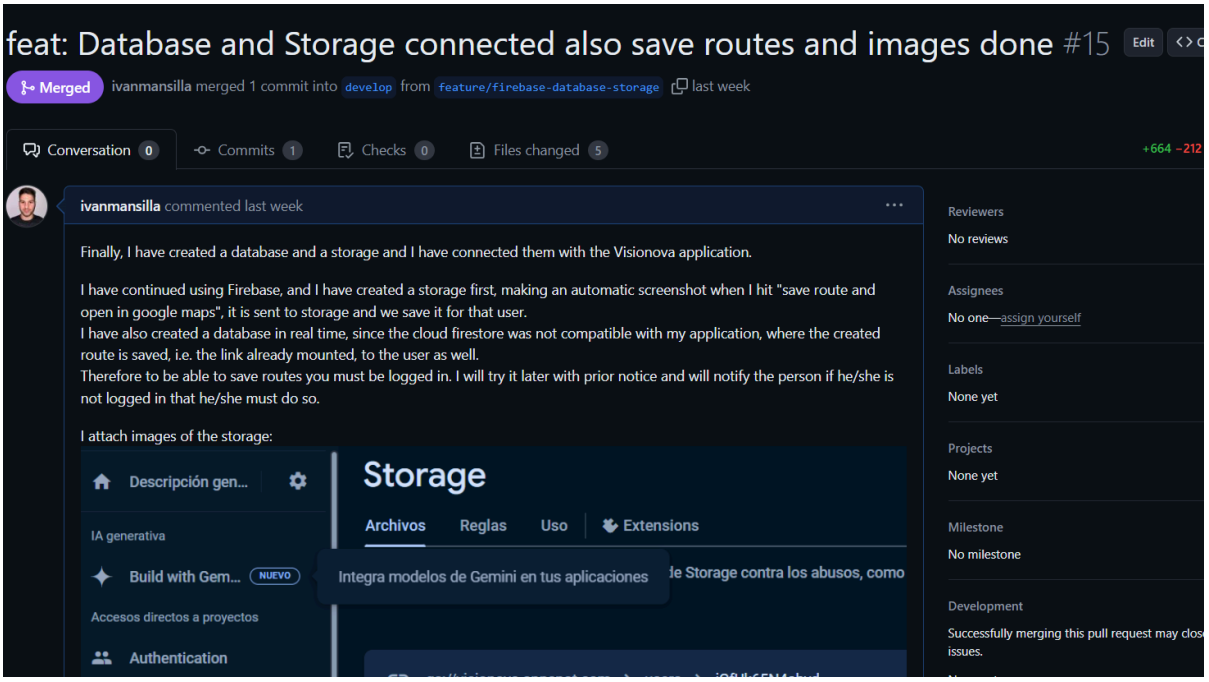


***Figura 24**: Ejemplo de merge request.. Definiendo los cambios realizados, opcionalmente con imágenes adjuntas*

Uso de Comandos Git y Resolución de Conflictos

El uso de comandos de Git en Bash fue fundamental para gestionar el proyecto de manera eficiente. Comandos como ***git add, git commit, git stash, git branch, git checkout, git push y git merge*** se utilizaron regularmente para controlar los cambios y mantener el flujo de trabajo. La resolución de conflictos, aunque rara, se manejó con cuidado para asegurar que el código final se mantuviera limpio y funcional. La experiencia adquirida a lo largo de la carrera en el uso de Git y GitHub resultó invaluable, permitiendo una gestión del proyecto sin sobresaltos y con un alto grado de profesionalismo.

Beneficios del Uso de GitHub

El uso de GitHub aportó numerosos beneficios al desarrollo de Visionova:

- **Control de Versiones**: Permite un seguimiento detallado de todos los cambios realizados en el código, facilitando la reversión de errores y la recuperación de versiones anteriores.
- **Colaboración**: En un futuro podría facilitar la colaboración entre desarrolladores, permitiendo trabajar en diferentes partes del proyecto de manera simultánea y coordinada en el caso de que en lugar de ser yo solo, haya más desarrolladores en Visionova.
- **Integración Continua**: La estructura de ramas y el uso de merge requests aseguran una integración continua y controlada, mejorando la calidad del código y la estabilidad del proyecto.

- **Transparencia**: Proporciona una visión clara y transparente del estado del proyecto, facilitando la gestión y la toma de decisiones informadas.

Github en comparación a otras alternativas

Existen varias otras herramientas que ofrecen funcionalidades similares, como GitLab, Bitbucket y SourceForge. A continuación, presentaré una comparación detallada de GitHub con estas alternativas, destacando las razones por las cuales GitHub fue seleccionado como la mejor opción para el desarrollo de Visionova.

GitHub vs. GitLab

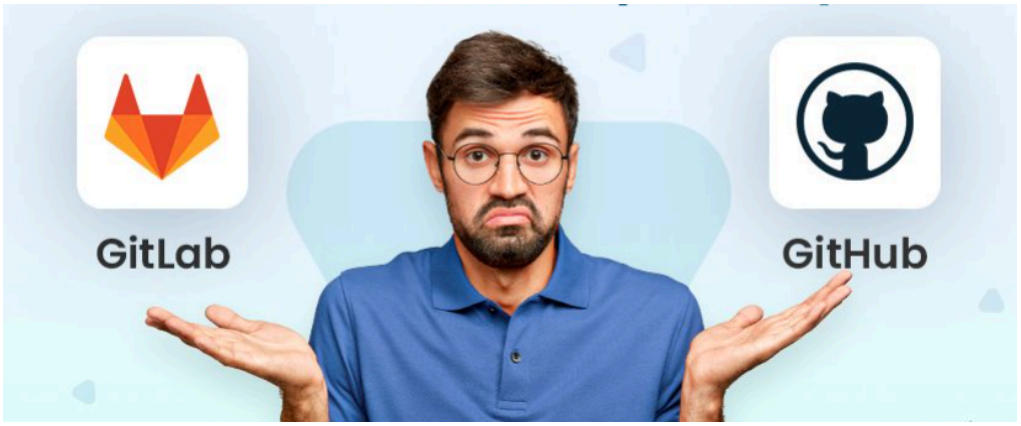


***Figura 25**: Duda entre que es mejor GitHub o GitLab.*

- **Popularidad y Comunidad**: GitHub tiene una comunidad más grande y activa que GitLab. Esto significa más recursos, más contribuciones de código abierto y una mayor probabilidad de encontrar soluciones a problemas comunes. La popularidad de GitHub también facilita la colaboración con otros desarrolladores, ya que muchos ya estamos familiarizados con la plataforma, en mi caso algo esencial, ya que durante la carrera github es algo que he visto en gran parte de asignaturas relacionadas con programación / proyectos.
- **Integraciones y Herramientas**: GitHub ofrece una amplia gama de integraciones con otras herramientas de desarrollo y servicios de terceros. Aunque GitLab también proporciona integraciones, GitHub tiene una ventaja en términos de variedad y facilidad de uso. Esto incluye integraciones con servicios de CI/CD, herramientas de gestión de proyectos, y plataformas de despliegue en la nube.
- **Interfaz de Usuario**: La interfaz de usuario de GitHub es intuitiva y fácil de usar, lo que facilita la navegación y la gestión de repositorios incluso para los nuevos usuarios. Aunque GitLab ha mejorado su interfaz con el tiempo, lo cual es algo que tuve en cuenta a la hora de elegir Gitlab como posible candidato, muchos desarrolladores encontramos que GitHub ofrece una experiencia de usuario más pulida y accesible.

GitHub vs. Bitbucket

- **Soporte para Git**: Mientras que Bitbucket soporta tanto Git como Mercurial, GitHub se centra exclusivamente en Git, lo que le permite ofrecer una experiencia más optimizada y específica para este sistema de control de versiones, donde ya tenía experiencia previa gracias tanto a mi experiencia profesional como durante el grado..
- **Capacidades de Colaboración**: GitHub se destaca por sus características de colaboración, como los pul y mergel requests, que facilitan las revisiones de código y la colaboración en proyectos, algo como he explicado previamente he utilizado para Visionova. Bitbucket también ofrece estas características, pero GitHub ha perfeccionado estas herramientas a lo largo de los años, haciendo que el proceso de colaboración sea más fluido y eficiente.
- **Comunidades y Repositorios Públicos**: GitHub es el hogar de una vasta cantidad de repositorios públicos de proyectos de código abierto, lo que lo convierte en un recurso invaluable para los desarrolladores que buscan colaborar en proyectos comunitarios. Bitbucket, aunque también soporta repositorios públicos, es más comúnmente utilizado para proyectos privados y corporativos.

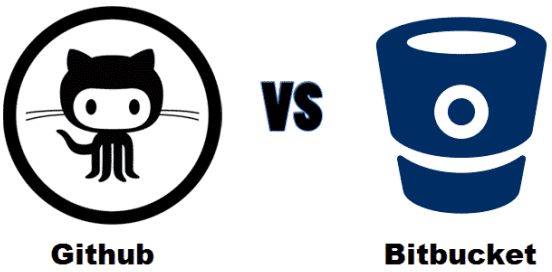


***Figura 26**: GitHub vs Bitbucket.*

Una vez vista la comparativa,hemos visto que existen varias plataformas competentes para la gestión de repositorios, es cierto que cualquiera de ellas podría ser una candidata a usarse en Visionova, pero ya no solo por mi experiencia previa, sino que GitHub se destaca por su facilidad de uso, fuerte comunidad, amplias integraciones y herramientas de colaboración eficaces. Estas características hacen que GitHub sea la mejor opción para el desarrollo de Visionova, asegurando un entorno de desarrollo robusto y eficiente.

4.3 Estructura del proyecto

El proyecto Visionova se organiza de manera estructurada y modular para garantizar un desarrollo eficiente, mantenible y escalable. A continuación, voy a detallar en profundidad la estructura del código, explicando la función de cada archivo y carpeta, y cómo estos componentes se integran para formar una aplicación cohesiva y funcional.

4.3.1. Visionova por dentro

Componente Principal: App.jsx

El archivo *App.jsx* actúa como el punto de entrada principal de la aplicación Visionova. Este archivo es crucial porque configura toda la navegación de la aplicación. Utilizando

bibliotecas como React Navigation, *App.jsx* define las rutas y las pantallas a las que los usuarios pueden acceder. Esta configuración no solo facilita la navegación del usuario, sino que también organiza el flujo de la aplicación de manera lógica y eficiente. En el contexto de Visionova, donde la accesibilidad y la usabilidad son primordiales, una navegación bien estructurada asegura que los usuarios con discapacidad visual puedan moverse fácilmente entre diferentes funcionalidades de la aplicación, como la cámara, el mapa, y las rutas guardadas.

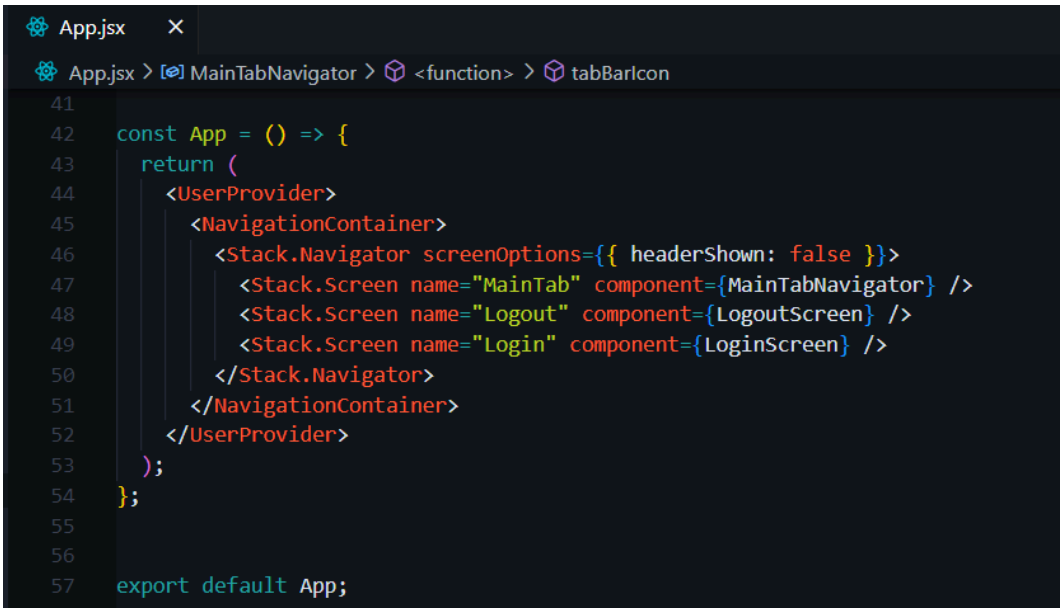


***Figura 27**: Trozo de código del componente principal App.jsx. Aquí es donde se controlan las rutas de la aplicación, se muestra el uso de Stack para lograrlo*

Carpeta Components

La carpeta components es fundamental para la organización del código de Visionova. Aquí se encuentran todos los componentes reutilizables, organizados en subcarpetas para mantener una estructura limpia y coherente.

Screens: Esta subcarpeta contiene los componentes que representan las diferentes pantallas de Visionova. Cada pantalla se implementa en su propio archivo, lo que facilita la gestión y el desarrollo modular. Por ejemplo, las pantallas de inicio de sesión (Login), de inicio (Home), de cámara (Camera), y de grabación de rutas (Record) se encuentran en esta carpeta. Esta organización permite que cada pantalla se desarrolle y mantenga de manera independiente, lo que es crucial para una aplicación con múltiples funcionalidades como Visionova.

Styles: En Visionova, se ha optado por utilizar *styled-components* para gestionar los estilos de manera modular y mantenible. *styled-components* permite escribir estilos en JavaScript, lo que proporciona varias ventajas sobre el uso de CSS tradicional.

- Encapsulamiento de Estilos: Con *styled-components*, los estilos se mantienen localizados a los componentes, evitando conflictos de estilos globales y

asegurando que cada componente mantenga su diseño previsto.

- Tematización: Facilita la implementación de temas, permitiendo cambiar el aspecto de la aplicación de manera dinámica y coherente. Esto es especialmente útil para aplicaciones que requieren ajustes visuales significativos basados en la preferencia del usuario o accesibilidad.

- Mantenibilidad: Al mantener los estilos junto con los componentes, se simplifica la actualización y el mantenimiento de los mismos, ya que los desarrolladores pueden ver y modificar los estilos en el mismo archivo donde se define la lógica del componente.

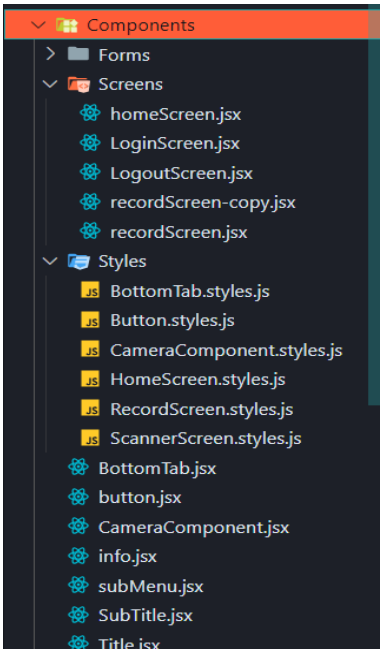


***Figura 28**: Estructura de Visionova.*

Componentes Reutilizables: Además de las pantallas principales, la carpeta incluye componentes reutilizables como botones, títulos e información. Estos componentes genéricos se diseñan para ser utilizados en múltiples partes de la aplicación, promoviendo la consistencia en el diseño y la funcionalidad. La creación de una biblioteca de componentes reutilizables asegura que elementos comunes mantengan un aspecto y comportamiento uniforme en toda la aplicación.

Funcionalidades Específicas: Algunos componentes encapsulan lógica específica para hacer el código más legible y mantenible. Por ejemplo, BottomTab maneja la lógica y la presentación de la barra de navegación inferior, un elemento crucial para la navegación accesible en Visionova.

Carpeta Hooks

Los hooks personalizados se encuentran en una subcarpeta dedicada. Los hooks son funciones especiales en React que permiten usar el estado y otras características de React sin escribir una clase. En Visionova, he utilizado hooks personalizados como useAuth y useContext para encapsular la lógica específica de la autenticación y el manejo del contexto global de la aplicación, respectivamente.

- useAuth: Este hook maneja la autenticación del usuario. Controla el estado de autenticación y escucha los cambios en el estado del usuario, proporcionando una interfaz sencilla para autenticar, cerrar sesión y recuperar información del usuario.
- useContext: Este hook facilita el acceso y la manipulación del contexto global de la aplicación, permitiendo compartir estados y funciones entre componentes sin necesidad de pasar props manualmente.

Figura 29: *Ejemplo del código del uso de hooks. En este caso del useContext.jsx*

4.3.2. Archivos principales de Visionova y beneficios de la estructura modular

Además de las carpetas mencionadas, las cuales son importantes, hay archivos clave en el nivel raíz del proyecto que juegan roles fundamentales, a diferencia de los anteriores comentados, estos tratan de hacer que el proyecto se mantenga, indicando rutas o configuraciones:

- **index.js**: Este archivo es el punto de entrada principal para la aplicación y se encarga de registrar el componente raíz *(App.jsx)* y configurar cualquier entorno específico de la plataforma. Es crucial para asegurar que la aplicación se inicie correctamente y se ejecute en el entorno deseado.
- **firebase.js**: Configuración e inicialización de Firebase. Firebase es esencial en Visionova para manejar la autenticación de usuarios, la base de datos en tiempo real, y el almacenamiento. Este archivo centraliza la configuración de Firebase, facilitando su gestión y actualización.

En Visionova he seguido una estructura modular, la cual ofrece varios beneficios importantes que podemos destacar como ahora la **Mantenibilidad**, es decir, separo el código en componentes específicos y reutilizables facilitando así la localización y corrección de errores,, igual que la adición de nuevas funcionalidades sin afectar otras partes del código, que junto a Github es algo esencial. Lo siguiente que he seguido es la **Escalabilidad**, que gracias a la estructura modular me permite escalar la aplicación fácilmente añadiendo nuevos componentes y funcionalidades sin introducir complejidad innecesaria. Seguimos con la **Reutilización**, donde como he comentado, los componentes reutilizables y los hooks personalizados me permiten utilizar la misma lógica y elementos visuales en múltiples partes de la aplicación, promoviendo la coherencia y reduciendo la duplicación de código.
Por último, si pensamos a largo plazo, como he ido intentando durante el desarrollo de Visionova, encontramos la **Colaboración** entre desarrolladores, ya que gracias a como todo está organizado y estructurado, permite que cada uno en un futuro pueda trabajar en componentes específicos de manera independiente, minimizando conflictos y mejorando la eficiencia del desarrollo.

La estructura del proyecto Visionova se ha diseñado para maximizar la eficiencia, mantenibilidad y escalabilidad del desarrollo. La clara división de componentes, la utilización de hooks personalizados y la adopción de *styled-components* aseguran que el proyecto sea fácil de entender, modificar y escalar, permitiendo una experiencia de usuario coherente y accesible.

4.4 Desarrollo e implementación

El desarrollo e implementación del proyecto Visionova ha sido una experiencia enriquecedora y desafiante, que me ha permitido adquirir nuevos conocimientos y fortalecer los ya existentes. A lo largo de este TFG, he navegado por un gran mar de tecnologías y herramientas, algunas completamente nuevas para mí, y otras en las que ya tenía cierta experiencia. Este proceso ha sido fundamental no solo para el desarrollo de Visionova, sino también para mi crecimiento profesional como ingeniero informático.

4.4.1. Progresión y vista general

Antes de comenzar con Visionova, contaba con una base sólida en programación y desarrollo de software, gracias a mis estudios universitarios y a mi experiencia profesional. Sin embargo, Visionova me planteó una serie de retos que me impulsaron a aprender y adaptarme rápidamente. Uno de los principales desafíos fue el uso de React Native y Expo, tecnologías con las que no tenía experiencia previa. Estas herramientas son esenciales para el desarrollo de aplicaciones móviles multiplataforma, y aprender a utilizarlas fue una curva de aprendizaje empinada.

Al principio, incluso tareas básicas como la instalación de React Native y Expo [53] fueron complicadas. La configuración inicial, la instalación de dependencias y la resolución de problemas de compatibilidad requerían una considerable cantidad de tiempo y esfuerzo durante todo el proyecto, pero poco a poco, a medida que me familiarizaba con el entorno de desarrollo y comprendía mejor su funcionamiento, empecé a sentirme más cómodo y eficiente.

Desarrollar Visionova desde cero ha sido mi primer gran proyecto individual de esta magnitud. Esta experiencia me ha obligado a adoptar y utilizar diversas herramientas de organización y gestión de proyectos, como Trello para el seguimiento de tareas y GitHub para el control de versiones. Implementar una metodología ágil, planificar cada fase del proyecto y seguir un calendario de objetivos han sido prácticas cruciales para mantener el rumbo y la eficiencia del desarrollo. A medida que avanzaba el desarrollo de Visionova implicó el incremento del uso de numerosas librerías y paquetes, cada uno contribuyendo de manera específica al funcionamiento de la aplicación. La gestión de estas dependencias se realizó principalmente mediante npm (Node Package Manager) y npx, herramientas que facilitan la instalación y gestión de paquetes en un proyecto JavaScript. Expo también desempeñó un papel central, proporcionando un conjunto de herramientas y servicios que simplifican el desarrollo de aplicaciones con React Native.

Una de las dificultades recurrentes fue la compatibilidad de las librerías con Expo. Algunas librerías no eran compatibles con Expo de manera directa, lo que requería buscar alternativas o soluciones temporales. **Axios**: Utilizada para realizar solicitudes HTTP, crucial para interactuar con APIs como Google Cloud Visión y Firebase [54] [55]. Para mostrar la magnitud tan grande de librerías, API's, herramientas que he utilizado para desarrollar este proyecto podemos prestar atención al *package.json* donde podemos encontrar todas las dependencias instaladas entre otras cosas.

***Figura 30**: Representación del package.json. Se muestran todas las librerías y de la magnitud el proyecto*

***Figura 31**: Dependencias utilizadas.*

Podemos detallan algunas de las tantas librerías clave y su papel en Visionova:

- **Google Cloud Vision API**: Fundamental para la funcionalidad de reconocimiento de texto a partir de imágenes capturadas por la cámara de la aplicación.

- **Expo Location API**: Utilizada para obtener la ubicación del usuario en tiempo real, necesaria para funcionalidades de seguimiento y grabación de rutas.
- **Expo File System API**: Permite el manejo de archivos dentro de la aplicación, necesario para almacenar y gestionar las imágenes capturadas y las rutas guardadas.
- **React Native Maps**: Utilizada para mostrar mapas interactivos en la aplicación, permitiendo a los usuarios visualizar sus rutas y ubicaciones.
- **Firebase**: Utilizado para la autenticación de usuarios y el almacenamiento de datos en la nube, proporcionando una backend robusta y escalable.

El manejo de dependencias y paquetes fue una parte crítica del desarrollo de Visionova. La instalación de las librerías necesarias y la resolución de conflictos de versiones han requerido una atención constante y algo muy importante a destacar.

4.4.2. Funcionalidades clave de Visionova

El desarrollo de Visionova se centra en ofrecer funcionalidades esenciales para mejorar la accesibilidad de personas con discapacidad visual. Dos de las funcionalidades más destacadas son el componente de la cámara, que permite el reconocimiento de texto a partir de imágenes, y la pantalla de grabación de rutas, que actualiza la ubicación del usuario en tiempo real y permite guardar y visualizar rutas. Visionova presenta estos dos componentes fundamentales que constituyen la columna vertebral de su funcionalidad, desarrollados con React Native y Expo, están diseñados para ofrecer una experiencia intuitiva y accesible para los usuarios con discapacidad visual. Vamos a analizar en detalle ambas funcionalidades, destacando sus características, desafíos técnicos y aspectos relevantes del desarrollo [56].

Componente de cámara

El componente de la cámara en Visionova permite a los usuarios capturar imágenes y obtener reconocimiento de texto (OCR) a partir de estas imágenes.. Este componente presenta las siguientes funcionalidades clave:

- **Captura de Imágenes**: Utilizando la API de Expo Camera, los usuarios pueden tomar fotografías fácilmente con su dispositivo móvil [57].
- **Reconocimiento de Texto**: La integración con la API de Google Cloud Vision permite el reconocimiento de texto en imágenes capturadas, ofreciendo a los usuarios una herramienta para leer texto impreso [58].

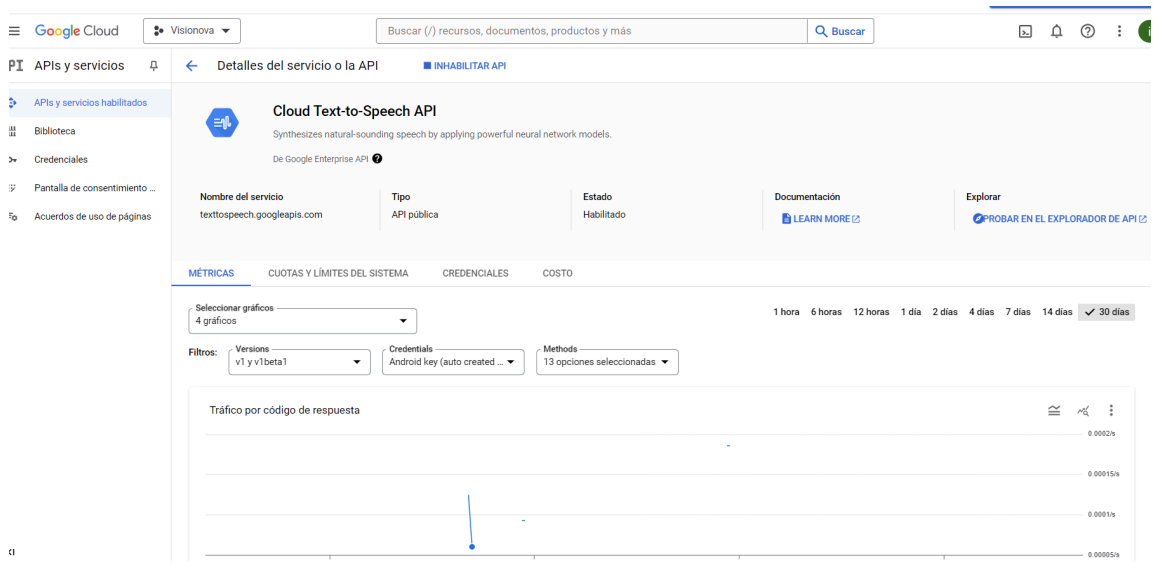


***Figura 32**: API de Text-To-Speech desde Google Cloud. Para entenderlo mejor, se muestra como se ve la API en la consola de Google Cloud*

Desafíos Técnicos:

- **Integración de la API de Google Cloud Vision**: Conectar y utilizar la API de Google Cloud Vision requirió una gestión precisa de las solicitudes HTTP y el procesamiento de las respuestas JSON devueltas por la API. La conversión de imágenes a formato base64 y la implementación de la lógica para el reconocimiento de texto fueron aspectos críticos de esta integración.

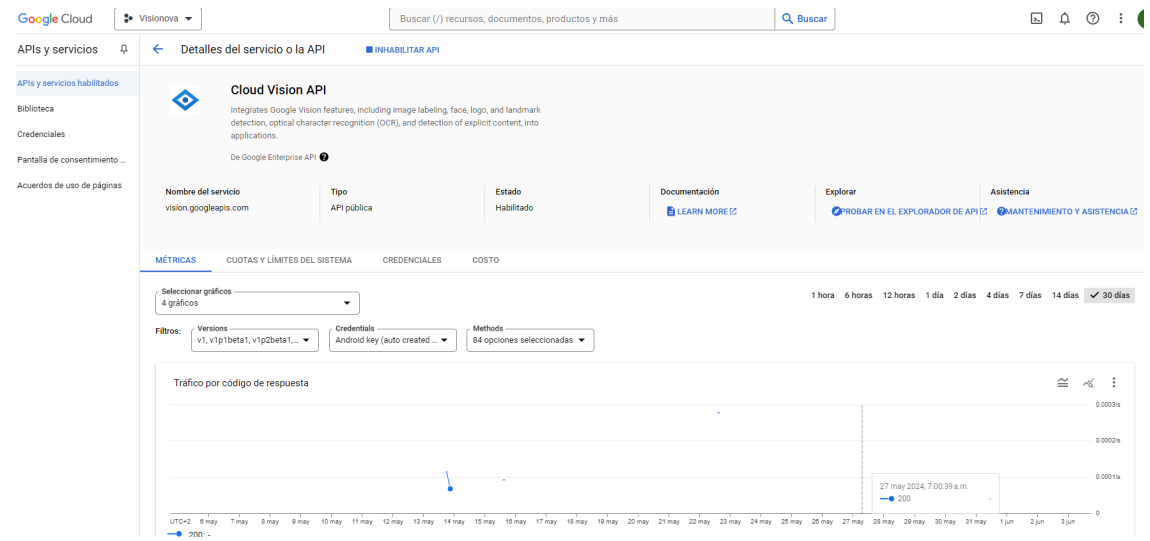


***Figura 33**: API de detectar texto desde Google Cloud. Para entenderlo mejor, se muestra como se ve la API en la consola de Google Cloud*

- **Manejo de la Reproducción de Audio**: La integración con la API de Google Text-to-Speech introdujo desafíos adicionales en la reproducción de audio. Se

implementaron referencias (refs) para controlar el estado de reproducción de audio y garantizar una reproducción y cancelación precisas del texto reconocido.

Pantalla de Grabación de Rutas

La pantalla de grabación de rutas en Visionova permite a los usuarios registrar y guardar sus recorridos mientras se desplazan. Esta funcionalidad incluye las siguientes características destacadas [59] [60]:

- **Seguimiento de ubicación en tiempo real**: Utilizando la API de Expo Location, los usuarios pueden rastrear su ubicación en tiempo real y visualizar su ruta en un mapa interactivo.
- **Almacenamiento de rutas**: Las rutas grabadas se almacenan en Firebase Realtime Database, lo que permite a los usuarios acceder y gestionar sus recorridos desde cualquier dispositivo [61].

Desafíos Técnicos:

- **Optimización del consumo de batería**: El seguimiento continuo de la ubicación en tiempo real puede ser intensivo en términos de consumo de batería. Se implementaron estrategias para optimizar el consumo de energía, como la configuración de intervalos de actualización de ubicación y la gestión eficiente de los recursos del dispositivo [62].

- **Integración con firebase**: La sincronización de datos con Firebase Realtime Database implicó la gestión de la autenticación de usuarios y el almacenamiento de datos en la nube. Se utilizaron bibliotecas y SDK proporcionados por Firebase para garantizar una integración segura y robusta con la plataforma Firebase.

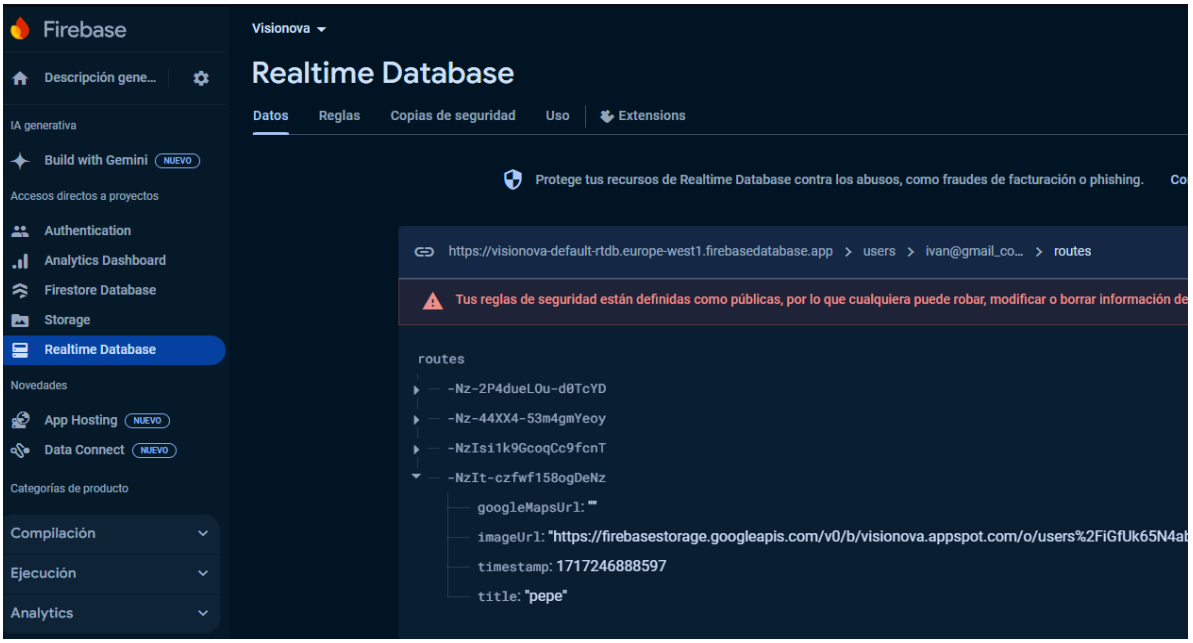


***Figura 34**: Visualización de la consola de Firebase. Concretamente se muestra la base de datos Realtime Database*

Ambos componentes clave de Visionova representan un esfuerzo considerable en el desarrollo de aplicaciones móviles accesibles y funcionales. Desde la integración de API hasta la gestión de la ubicación en tiempo real, cada aspecto del desarrollo se abordó con atención al detalle y un enfoque centrado en el usuario [63] [64].

4.4.3 Principios SOLID implementados

La aplicación Visionova se ha desarrollado siguiendo los principios SOLID, los cuales son fundamentales para crear un software mantenible, extensible y robusto. Estos principios, que se centran en el diseño orientado a objetos, aseguran que el código sea fácil de entender y de modificar. A continuación, se explica cómo se han aplicado estos principios en el desarrollo de Visionova [65]:

1. **Single Responsibility Principle (SRP)**: Cada componente en Visionova tiene una responsabilidad única. Por ejemplo, el componente *CameraComponent* se encarga exclusivamente de la funcionalidad relacionada con la cámara, como capturar imágenes y reconocer texto, mientras que el *RecordScreen* maneja exclusivamente la grabación y almacenamiento de rutas. Esto asegura que cada componente tenga una única razón para cambiar, lo que facilita el mantenimiento y la comprensión del código.

2. **Open/Closed Principle (OCP)**: Los componentes están diseñados para ser abiertos a la extensión pero cerrados a la modificación. Por ejemplo, si se necesita añadir una nueva funcionalidad de reconocimiento de texto, se puede extender el componente de cámara sin modificar su código existente, posiblemente mediante la implementación de un nuevo servicio que se integre con la API de Google Cloud Vision.

3. **Liskov Substitution Principle (LSP)**: Este principio se asegura de que los objetos de una clase derivada puedan sustituir objetos de una clase base sin afectar el comportamiento del programa. En Visionova, esto se observa en la forma en que los componentes de cámara y grabación de rutas utilizan diversas APIs y servicios. Cualquier cambio o sustitución de estos servicios (por ejemplo, cambiando de Google Cloud Vision a otro servicio de OCR) se puede hacer sin afectar al resto del código que depende de estos componentes.

4. **Interface Segregation Principle (ISP)**: En lugar de utilizar interfaces grandes y generales, se utilizan interfaces específicas y pequeñas. Por ejemplo, en la integración con Firebase, las interfaces para autenticación, almacenamiento y base de datos están claramente segregadas, permitiendo que cada módulo interactúe solo con los métodos que necesita.

5. **Dependency Inversion Principle (DIP)**: Visionova depende de abstracciones en lugar de implementaciones concretas. Esto se ve en la forma en que se inyectan dependencias como la cámara, el reconocimiento de texto y los servicios de Firebase, permitiendo que estos componentes sean fácilmente reemplazables o modificables sin alterar la lógica de la aplicación.

Metro para visualizar

Para asegurar una construcción eficiente y evitar problemas comunes en el desarrollo con React Native, Visionova utiliza una configuración optimizada de Metro, el bundler de JavaScript utilizado por React Native. La configuración de Metro se ajusta para manejar adecuadamente los assets, como las imágenes capturadas y los mapas, garantizando que se carguen y procesen eficientemente.

Figura 35: Pequeño código que muestra cómo se visualiza a través de Metro.

4.5 Base de datos y uso de Firebase

4.5.1 Autenticación

Inicialmente, la autenticación en Visionova presentó varios desafíos. Aunque se consideró el uso de Auth0, finalmente se optó por Firebase Authentication debido a su integración más sencilla y a la amplia documentación disponible. Firebase Authentication permite gestionar usuarios de manera segura y eficaz, ofreciendo soporte para múltiples proveedores de inicio de sesión (correo electrónico, Google, Facebook, etc.) [66] [67]..

La integración con Firebase Authentication fue complicada debido a la necesidad de asegurar una sincronización adecuada entre la aplicación y el backend, especialmente cuando se trata de manejar estados de sesión y autenticación en tiempo real. A pesar de estos desafíos, la elección de Firebase se justificó por su robustez y su capacidad para manejar fácilmente la autenticación de usuarios en aplicaciones móviles.

4.5.2 Base de datos - Firebase

Firebase Realtime Database fue elegido para almacenar y sincronizar los datos de las rutas grabadas. Aunque inicialmente se consideró el uso de Cloud Firestore, se encontró que Realtime Database era más compatible con Expo y ofrecía ventajas específicas para las necesidades de Visionova [68].

Ventajas de Firebase Realtime Database:

- **Sincronización en Tiempo Real**: Permite que los datos se sincronicen en tiempo real entre los dispositivos de los usuarios y la base de datos, lo que es crucial para aplicaciones que requieren datos actualizados constantemente.
- **Fácil de Integrar**: La API de Realtime Database es sencilla de usar y se integra bien con la arquitectura de una aplicación React Native.

- **Escalabilidad**: Aunque Realtime Database es más adecuado para proyectos más pequeños o con menos datos que Cloud Firestore, sigue ofreciendo escalabilidad suficiente para el alcance de Visionova.

Almacenamiento de Datos y Rutas e Imágenes

Las rutas grabadas se almacenan en la Realtime Database bajo la estructura de usuarios y rutas. Cada ruta incluye el URL de Google Maps, la URL de la imagen capturada, el timestamp, y el título de la ruta. La estructura de la base de datos es la siguiente:

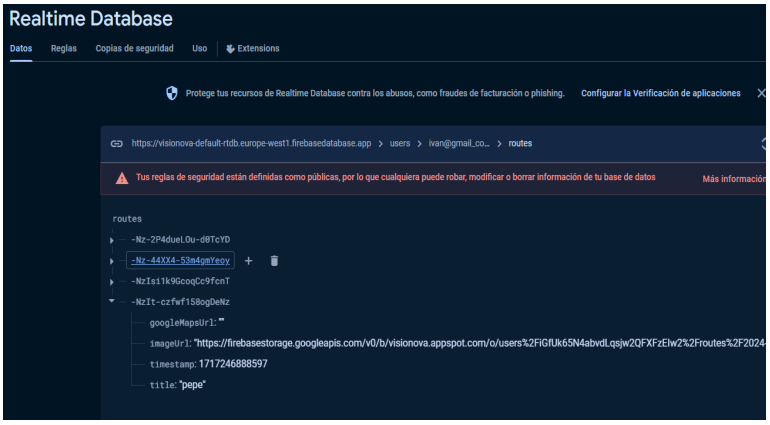


***Figura 36**: Dentro de la base de datos. Vista de como es el guardado de las rutas.*

[La podemos encontrar en mayor tamaño en el Anexo]

Cada ruta incluye información esencial, lo que permite a los usuarios revisar sus rutas pasadas y acceder a las imágenes y mapas asociados.

Para almacenar las imágenes capturadas durante la grabación de rutas, se utiliza Firebase Storage. Las imágenes se guardan en la nube y se referencian mediante URLs en la Realtime Database, lo que permite un acceso rápido y eficiente a los archivos sin ocupar espacio en el dispositivo del usuario

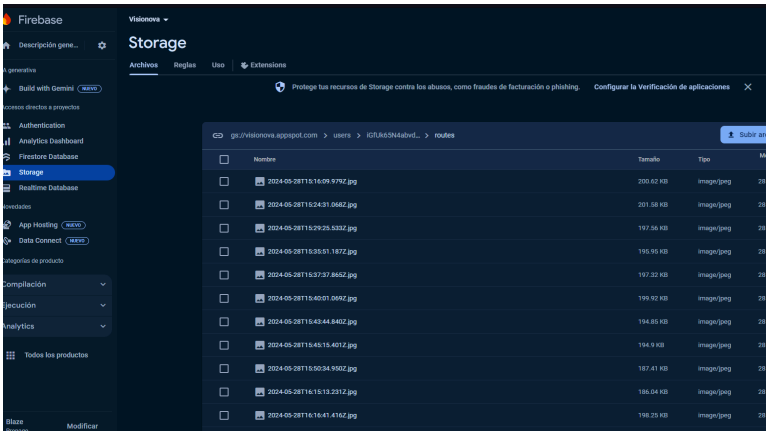


***Figura 37**: Firebase Storage. Almacenaje de las imágenes.*

[La podemos encontrar en mayor tamaño en el Anexo]

Como conclusión [69] en todo el apartado de implementación y desarrollo podemos decir que el desarrollo de Visionova ha sido un proceso complejo pero gratificante, implicando la aplicación de principios sólidos de diseño de software y la integración de múltiples servicios en la nube para crear una aplicación robusta y funcional [70]. El uso de Firebase para la autenticación y el almacenamiento de datos ha proporcionado una base sólida, aunque no sin sus desafíos, y la implementación cuidadosa de los principios SOLID ha asegurado que el código sea mantenible y extensible a largo plazo.

5. Resultados

El objetivo principal de Visionova era crear una aplicación móvil accesible e intuitiva que proporcionara herramientas útiles para personas con discapacidad visual. A lo largo de su desarrollo, se buscaba garantizar una experiencia de usuario óptima, teniendo en cuenta la facilidad de uso y la eficacia de las funcionalidades implementadas.
Ahora se mostrará cuales han sido los resultados de la aplicación, mostrando los puntos fuertes y los mejorables, entrando en detalle en cada pantalla para entender cómo se ha compuesto y que es Visionova.

5.1 Pantalla principal, inicio de sesión y perfil

Para comenzar el recorrido por Visionova, nos encontramos con la pantalla principal, conocida como "Home". Esta pantalla representa el corazón de la aplicación, donde podemos llegar a todas las funcionalidades clave diseñadas para ofrecer una experiencia excepcional a los usuarios, especialmente aquellos con discapacidad visual. Desde el diseño hasta la disposición de los elementos, cada detalle se ha concebido con el objetivo de brindar accesibilidad y facilidad de uso..

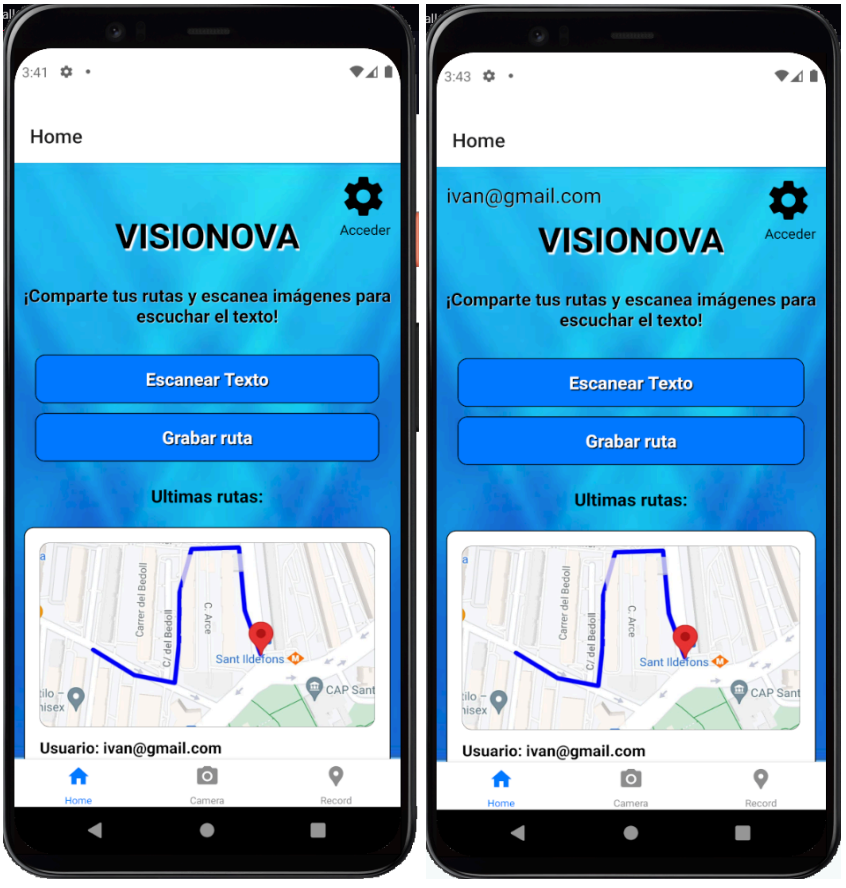


***Figura B.1**: Pantalla homeScreen. sin iniciar sesión.*
***Figura B.2**: Pantalla homeScreen.habiendo iniciado sesión.*

Al abrir la aplicación, los usuarios son recibidos por un diseño claro y conciso que garantiza una navegación intuitiva desde el primer momento. En esta pantalla, el fondo

de pantalla ha sido seleccionado con previo análisis para garantizar una visibilidad óptima, considerando las necesidades específicas de los usuarios con discapacidad visual. Además, se han incorporado palabras clave que pueden ser dictadas por el dispositivo móvil, facilitando la comprensión de la pantalla para aquellos que dependen de la voz del dispositivo.
La disposición de los elementos en la pantalla principal también se ha diseñado con la usabilidad en mente. En la parte inferior de la pantalla, los usuarios encuentran un conjunto de botones de navegación que les permiten acceder fácilmente a las funcionalidades principales de la aplicación. Este bottomTab consiste en tres botones que dirigen a los usuarios a la pantalla de inicio, la cámara y la pantalla de grabación de rutas. Esta organización intuitiva facilita la navegación entre las diferentes partes de la aplicación, mejorando significativamente la experiencia del usuario.

Además de la navegación, la pantalla principal también presenta elementos importantes como el nombre de la aplicación y un ícono de ajustes en la esquina superior derecha. Si el usuario está logueado, su nombre aparecerá en la esquina superior izquierda, proporcionando una experiencia personalizada y familiar desde el principio [Figura B.2].

Uno de los aspectos más destacados de la pantalla principal son las "Últimas Rutas". Esta sección muestra las rutas más recientes subidas por los usuarios en forma de tarjetas. Cada tarjeta incluye una imagen representativa, el nombre del usuario que la ha subido y, opcionalmente, un título descriptivo. Esta disposición intuitiva permite a los usuarios explorar rápidamente las últimas rutas compartidas por la comunidad de Visionova.

Figura B.3: *Como se muestran las rutas en la homeScreen.*

Después de explorar la pantalla principal de Visionova, el siguiente paso natural es iniciar sesión para acceder a todas las funcionalidades de la aplicación. Esto se logra fácilmente a través de la opción de ajustes representada por una rueda que se encuentra en la esquina superior derecha de la pantalla principal.
Al hacer clic en la rueda de ajustes, los usuarios son recibidos con un menú desplegable que ofrece dos opciones distintas dependiendo de su estado de inicio de sesión. Si el usuario aún no ha iniciado sesión, verá un botón de "Login" que le permite acceder al proceso de inicio de sesión. Por otro lado, si el usuario ya ha iniciado sesión, verá las opciones de "Logout" y "Mi perfil"..

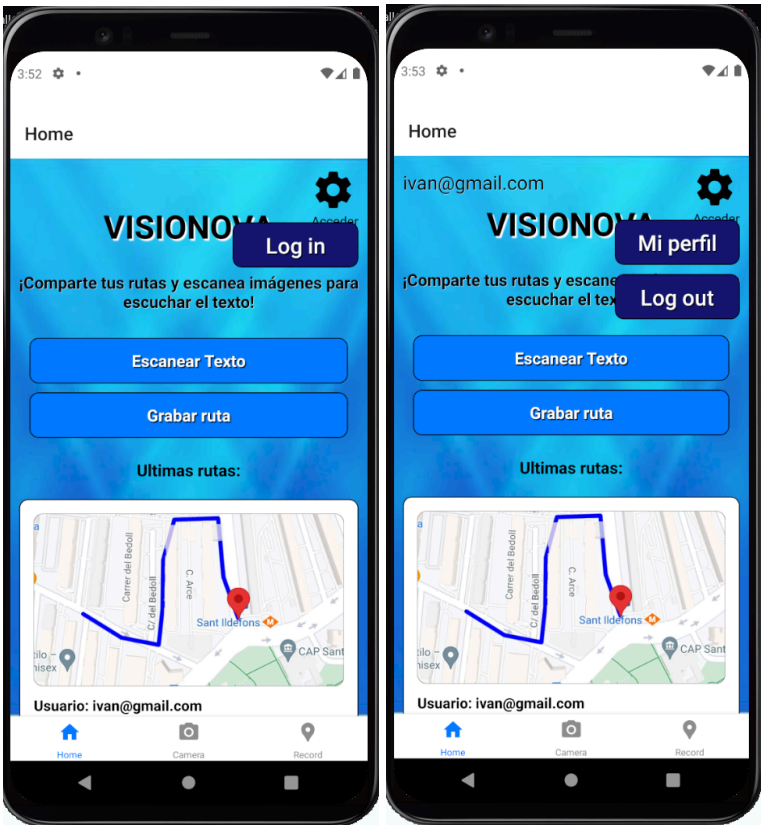


***Figura B.4**: Pantalla homeScreen con ajustes desplegados. Se muestran botones de Log in, Log out y Mi perfil*

En el caso de iniciar sesión, Visionova ofrece un diseño simple pero efectivo que permite al usuario ingresar su correo electrónico y contraseña. Al clicar en "Login", si las credenciales son válidas, el usuario será redirigido automáticamente a la pantalla de inicio. Sin embargo, si las credenciales son incorrectas o el usuario intenta iniciar sesión sin estar registrado, aparecerá un mensaje de error indicando la situación.

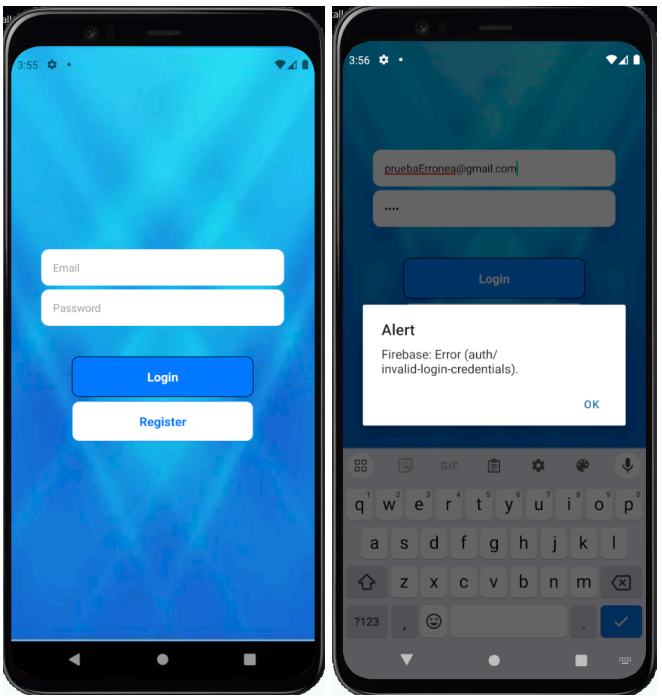


***Figura B.5**: Pantalla de Log in,, incluido ejemplo de error al entrar.*

Una vez que el usuario ha iniciado sesión correctamente, se abrirá la pantalla de inicio, brindándole acceso a todas las funcionalidades de la aplicación. Por otro lado, si el usuario opta por cerrar sesión, se le presentará una pantalla simple con un mensaje de confirmación. Aquí, el usuario puede elegir entre confirmar el cierre de sesión o cancelar la acción y volver a la pantalla de inicio.

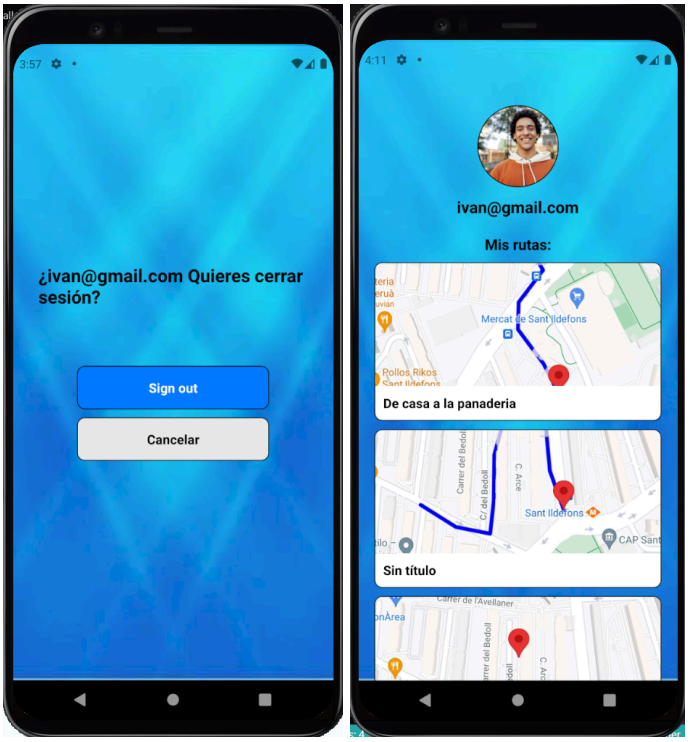


***Figura B.6**: Pantalla de Log out.*
***Figura B.7**: Pantalla de Mi perfil, con rutas incluidas.*

En todos los casos, desde el proceso de inicio de sesión hasta el cierre de sesión, Visionova ha seguido un estilo coherente y fácil de entender en toda la aplicación. Esto garantiza una experiencia de usuario consistente y cómoda, donde los usuarios pueden navegar con confianza y seguridad en cada paso del camino. Ahora, después de explorar el proceso de inicio de sesión, continuaremos con la exploración de la funcionalidad "Mi perfil".

En la pantalla de "Mi perfil" de Visionova, los usuarios encontrarán una interfaz intuitiva y bien diseñada que les permite gestionar su información personal y acceder fácilmente a sus rutas guardadas. Esta pantalla está cuidadosamente diseñada para brindar una experiencia de usuario coherente y amigable, manteniendo la misma estética visual y funcionalidad que caracteriza a toda la aplicación [Figura B.7].

En la parte superior de la pantalla, los usuarios verán una sección dedicada a su imagen de perfil. Si el usuario ha cargado una imagen previamente, esta se mostrará aquí. De lo contrario, aparecerá un ícono de imagen predeterminado con un texto que indica "Añadir imagen", lo que invita al usuario a cargar una foto para personalizar aún más su perfil.
Justo debajo de la imagen de perfil, se muestra el nombre del usuario, lo que proporciona una identificación clara y rápida. Esta simple inclusión ayuda a los usuarios a sentirse bienvenidos y conectados con su cuenta personal.
Sin embargo, la parte más destacada de la pantalla de "Mi perfil" es la sección de "Mis rutas". Aquí es donde los usuarios pueden acceder y gestionar todas las rutas que han guardado en la aplicación. Al igual que en la pantalla de inicio, las rutas se presentan en forma de tarjetas, lo que facilita la visualización y comprensión de la información.

Cada tarjeta de ruta incluye una imagen representativa de la ruta y, si está disponible, un título descriptivo de la ruta. Esta presentación clara y ordenada permite a los usuarios identificar rápidamente las rutas que desean acceder o modificar.

5.2 Funcionalidad de la pantalla cámara

Una vez hemos visto lo que compone Visionova, ahora vamos a entrar más en detalle en lo que le da el significado y le hace única, es decir las funcionalidades.

Primero de todo tenemos la funcionalidad de la cámara en Visionova es una característica central que permite a los usuarios realizar tareas importantes de forma rápida y eficiente. Lo más notable es que esta pantalla está diseñada para ser accesible para cualquier usuario, incluso aquellos que no están registrados en la aplicación, lo que amplía su utilidad y alcance.

Cuando un usuario accede a la pantalla de la cámara, se encuentra directamente con la interfaz de la cámara del dispositivo, lo que elimina cualquier paso adicional o complejidad innecesaria. Esto simplifica enormemente la experiencia del usuario, ya que no necesita navegar a través de múltiples menús o configuraciones para acceder a la funcionalidad de la cámara. Se le pide el permiso para poder utilizar la cámara, ya que es algo esencial para la seguridad de los usuarios.

***Figura B.8**: Pantalla de cámara, pidiendo permisos al usuario.*
Figura B.9: Pantalla de cámara, realizando fotografía.
***Figura B.10**: Pantalla de cámara con foto realizada.*

En la parte inferior de la pantalla, se coloca un botón claramente identificable para capturar la imagen. Este botón sigue el diseño estándar de los dispositivos móviles, lo que facilita su reconocimiento y uso para cualquier usuario familiarizado con la interacción de la cámara en sus teléfonos [Figura B.9]. Después de capturar la imagen, se muestra en la pantalla de inmediato, ofreciendo al usuario dos opciones claras: cerrar la imagen o guardarla. Esta decisión de diseño se basa en la premisa de proporcionar al usuario un control completo sobre sus acciones y facilitar la navegación intuitiva [Figura B.10]

Una vez que se ha realizado la imagen, Visionova desencadena automáticamente el proceso de reconocimiento de texto. Este proceso es fundamental para la funcionalidad de la aplicación, ya que permite que el texto contenido en la imagen sea accesible para usuarios con discapacidades visuales. Es importante destacar que el sistema de reconocimiento de texto de Visionova está diseñado para ser altamente efectivo, incluso en condiciones subóptimas, como texto borroso o distante, algo lógico que ocurra con una alta probabilidad teniendo en cuenta a los usuarios en los que me he enfocado.

```
LOG  Texto reconocido: vida fuera como una telaraña, un laberinto de a
nativas que se extienden desde un único punto cen-
tral. Tendríamos todo tipo de opciones y ninguna
sería irreversible, porque siempre habría otra ruta de
regreso. En cambio, sólo hay una recta, sin más alter-
nativa, un impulso implacable y una sola dirección.
Jonathan ha muerto. Charles ha muerto. Pero no
os han dejado en absoluto.
Cada vez que hago un crucigrama, pienso en Char-
s. Me pregunto qué diría, si sabría descifrar la última
ista, si sabría la solución que a mí se me escapa. Cada
z que veo a un hombre con las uñas de los pies un
co demasiado largas, pienso en Charles. Pienso en
spies feos y en su insistencia en ir con sandalias por
casa en verano. Cada vez que veo un nudo de cor-
```

***Figura B.11**: Texto reconocido. Para mostrar que funciona, se muestra el texto de la imagen.*

5.3 Funcionalidad de la pantalla mapa

Ahora sigamos con la siguiente funcionalidad, la más grande y posiblemente innovadora que ofrece la app: Esta funcionalidad de grabación de rutas en Visionova se destaca por su enfoque directo y su diseño intuitivo, que prioriza la experiencia del usuario y su facilidad de uso. Comentar que esta funcionalidad es única y exclusiva para los usuarios registrados, por lo tanto si se quiere acceder sin haber registrado o iniciado sesión, se puede utilizar igualmente el mapa, ya que no se quiere prohibir todo totalmente, pero aparece un mensaje aconsejando que inicien sesion para poder grabar rutas, así logramos atraer a más usuarios y hacer una comunidad más grande y fuerte.

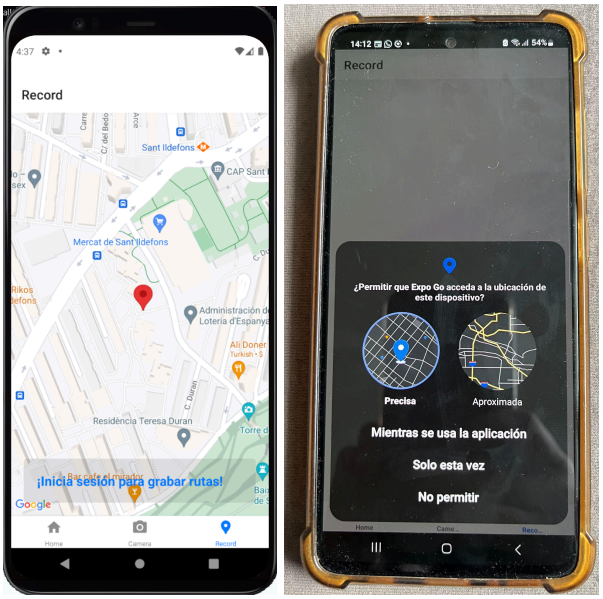


***Figura B.12**: Pantalla de mapa sin haber iniciado sesión.*
***Figura B.13**: Pantalla de mapa pidiendo permisos al entrar.*

Al haber iniciado sesión y al abrir esta función, el usuario se encuentra con un mapa interactivo que solicita permiso para acceder a su ubicación, un paso fundamental para la funcionalidad de la aplicación [Figura B.13].

Una vez que se concede el permiso, el mapa muestra la ubicación del usuario con un marcador, lo que proporciona una referencia visual clara desde el inicio. En la parte superior de la pantalla, se encuentra un campo de entrada donde el usuario puede ingresar un título para la ruta que está a punto de grabar. Esta característica permite al usuario describir la ruta o agregar información relevante antes de iniciar la grabación, lo que mejora la experiencia de usuario personalizada.
En la parte inferior de la pantalla, se presentan una serie de botones diseñados para facilitar la interacción del usuario con la aplicación. Estos botones incluyen la opción de Grabar ruta/Detener ruta, que se adapta según el estado actual de la grabación. La presencia de un botón Limpiar ofrece la posibilidad de borrar el mapa si el usuario desea grabar una nueva ruta sin confusiones, ya que durante la grabación de la ruta, para facilitar el seguimiento, y que el usuario en todo momento pueda observar si la ruta es la correcta, aparece una linia de color azul siguiendo al usuario.

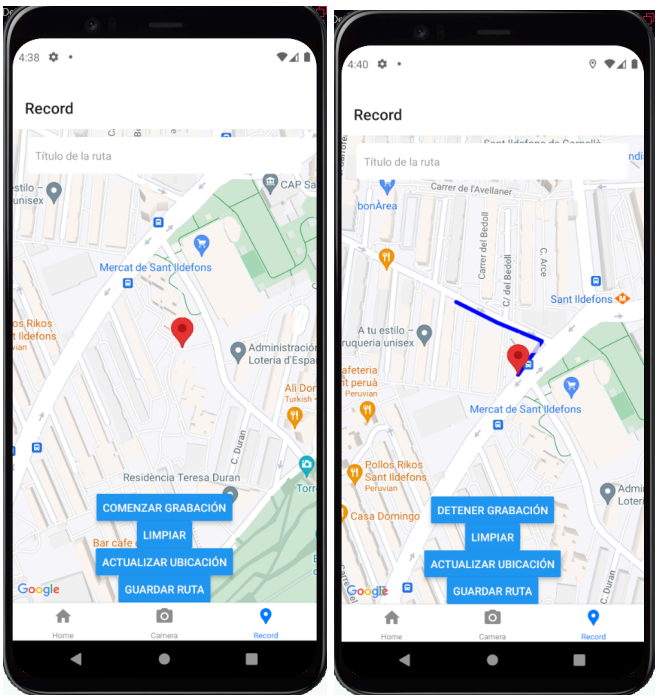


***Figura B.14**: Pantalla de mapa, simulando ruta. Se simula la ruta, siguiendo con una línea azul, y con los botones y posibilidad de título.*

Además, se incluye un botón de Actualizar ubicación, que permite al usuario centrar el mapa en su ubicación actual en cualquier momento durante la grabación. Esta función es especialmente útil para los usuarios que desean mantenerse en el camino deseado o necesitan orientación adicional durante el recorrido.

El botón final, Guardar ruta, es una característica crítica que permite al usuario guardar la ruta una vez que se ha completado. Después de guardar la ruta, aparece un mensaje de confirmación para informar al usuario que la acción se ha completado con éxito, lo que refuerza la retroalimentación positiva y la sensación de logro.

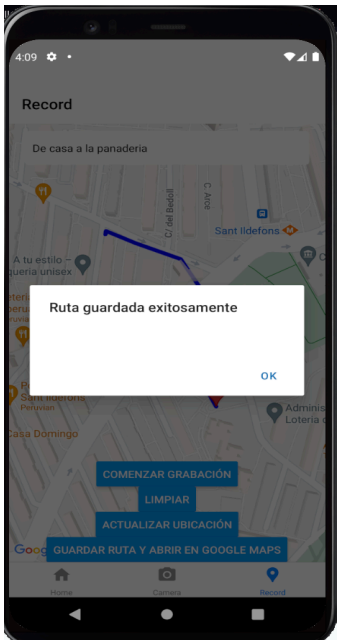


***Figura B.15**: Guardando la ruta correctamente.*
Al guardar una ruta, se muestra un mensaje para confirmárselo al usuario

5.4 Conclusiones de los resultados

Ahora finalmente tras el trabajo realizado se puede decir que Visionova es una aplicación innovadora que ha logrado alcanzar con éxito los objetivos planteados, ofreciendo una experiencia única y accesible para sus usuarios. Desde su concepción, la aplicación se ha centrado en brindar herramientas poderosas para personas con discapacidad visual, priorizando la accesibilidad y la usabilidad en cada paso del proceso.

Uno de los principales objetivos de Visionova era proporcionar una plataforma integral que permitiera a las personas con discapacidad visual acceder a información de manera fácil y rápida. Este objetivo se ha cumplido mediante el desarrollo de funcionalidades como la lectura de texto mediante la cámara y la grabación de rutas mediante GPS, que utilizan tecnologías avanzadas para ofrecer una experiencia inclusiva y efectiva.
La aplicación se distingue por su diseño intuitivo y su enfoque centrado en el usuario, lo que garantiza que todas las funcionalidades sean accesibles y fáciles de usar para cualquier usuario, independientemente de su nivel de habilidad tecnológica. Además, Visionova se ha destacado por su capacidad para adaptarse a las necesidades cambiantes de sus usuarios, incorporando retroalimentación y mejoras constantes para garantizar una experiencia óptima.
Otro aspecto destacado de Visionova es su enfoque en la innovación y la tecnología de vanguardia. Desde la integración de la inteligencia artificial para la lectura de texto hasta el uso del GPS para la grabación de rutas, la aplicación aprovecha las últimas tecnologías para ofrecer soluciones efectivas y eficientes para sus usuarios.

En definitiva, estoy encantado de ver que Visionova ha logrado alcanzar todos los objetivos que me propuse desde el principio. La aplicación ha demostrado ser una herramienta innovadora y efectiva para personas con discapacidad visual, ofreciendo una experiencia accesible y útil en cada paso del camino.

A pesar de algunos desafíos, como la parte de seguirse entre usuarios que aún no ha sido implementada, estoy satisfecho con los resultados obtenidos. Sé que esta funcionalidad es importante para la interacción entre los usuarios, y será una prioridad en futuras actualizaciones de la aplicación. Sin embargo, estoy orgulloso del progreso que hemos logrado hasta ahora. Visionova ha superado todas mis expectativas, brindando soluciones efectivas para la lectura de texto y la grabación de rutas, y destacando por su diseño intuitivo y su enfoque centrado en el usuario.

Mirando hacia el futuro, estoy emocionado por las oportunidades que esperan a la aplicación. Seguiré trabajando para mejorar y expandir Visionova, incorporando nuevas funcionalidades y mejoras basadas en la retroalimentación de los usuarios y las últimas innovaciones tecnológicas.

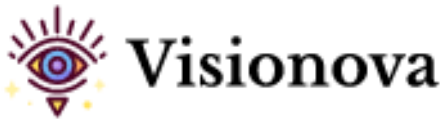


***Figura B.16**: Posible futuro logo de Visionova*

6. Relación con el grado

La realización del Trabajo de Fin de Grado (TFG), en este caso la aplicación Visionova, ha sido una experiencia profundamente relacionada con los conocimientos y habilidades adquiridos a lo largo del Grado en Ingeniería Informática.
Sin la formación tan amplia proporcionada por el grado, el desarrollo de Visionova desde cero no habría sido posible. La carrera me ha dotado no sólo de conocimientos técnicos específicos, sino también de una mentalidad analítica y una capacidad para aprender nuevas tecnologías rápidamente, resolver problemas complejos y gestionar proyectos de software completos. Entrando más en detalle, de cómo diferentes asignaturas entre muchas otras me han ayudado con áreas específicas, resaltar que aunque comente algunas, definitivamente todas han aportado en mi su grano de arena.

6.7.1 Projecte Integrat de Software (PIS)

Esta asignatura ha sido esencial en el desarrollo de Visionova. En PIS, tuve la oportunidad de trabajar en la creación de una aplicación en Java con Android Studio junto a varios compañeros. Esta experiencia fue invaluable, ya que me enseñó no sólo los aspectos técnicos del desarrollo de aplicaciones móviles, sino también la importancia del trabajo en equipo y la colaboración efectiva. Aprendí a dividir el trabajo, a coordinarme con otros desarrolladores, que aunque en Visionova solo yo era el único desarrollados, trabajé como si fuese más de uno, para facilitar en el futuro el trabajo, y por tema de organización, y por lo tanto ayudó a integrar las diferentes partes del proyecto en una aplicación cohesiva. Además, el uso de emuladores de Android en PIS fue directamente aplicable al desarrollo y prueba de Visionova, facilitando el proceso de depuración y optimización de la aplicación en diversas plataformas y dispositivos.

6.7.2 Enginyeria del Software

La asignatura de Enginyeria del Software me proporcionó una base sólida en organización y planificación de proyectos, así como en metodologías ágiles como Scrum. Estas habilidades fueron fundamentales para estructurar el desarrollo de Visionova. Utilicé herramientas de gestión de proyectos como Trello y Kanban para seguir el progreso del proyecto, asignar tareas y gestionar el tiempo de manera eficiente. Además, la práctica de estimar el esfuerzo requerido para cada tarea utilizando Story Points me ayudó a planificar y priorizar el trabajo de manera efectiva, asegurando que se cumplieran los plazos y objetivos del proyecto. La experiencia de conectar una aplicación con un backend fue directamente transferible al trabajo con Visionova, donde tuve que integrar múltiples APIs y servicios backend.

6.7.3 Factors Humans (FFHH)

En Factors Humans, aprendí la importancia de la accesibilidad y el diseño visual en el desarrollo de aplicaciones. Estos conocimientos fueron cruciales para hacer que Visionova sea accesible y atractiva para los usuarios. La experiencia con el lenguaje VUE, similar a React, facilitó la transición a React Native, permitiéndome aplicar principios de diseño y accesibilidad aprendidos en FFHH a la interfaz de usuario de Visionova. Esta asignatura me ayudó a entender cómo diseñar aplicaciones que sean no solo

funcionales, sino también intuitivas y agradables de usar, mejorando así la experiencia del usuario.

6.7.4 Disseny de Software (DS)

Los principios SOLID y las buenas prácticas de diseño de software enseñados en Disseny de Software fueron fundamentales para asegurar que Visionova se desarrollara de manera estructurada y mantenible. Aplicar estos principios me permitió crear un código limpio y modular, facilitando futuras modificaciones y mejoras en la aplicación. La adherencia a estos principios también contribuyó a la estabilidad y eficiencia de la aplicación, asegurando que cada componente cumpliera una función específica y se integrara perfectamente con el resto del sistema.

6.7.5 Sistemas Operatius (SO)

La asignatura de Sistemas Operatius me proporcionó habilidades prácticas en el uso de Git y el manejo de comandos, herramientas esenciales para el control de versiones y la colaboración en el desarrollo de software. Utilizar Git me permitió llevar un seguimiento detallado de los cambios en el código, colaborar con otros desarrolladores de manera efectiva y gestionar diferentes versiones del proyecto de manera eficiente. Esta experiencia fue crucial para mantener el control y la organización del código fuente de Visionova.

6.7.6 Diferentes asginaturas

Software Distribuit (SD), Gráfics, Visió Artificial (VA), Factors Humans (FFHH), Projecte Integrat de Software (PIS), Inteligencia Artificial (IA), entre muchas otras hicieron posible que la experiencia adquirida en estas asignaturas con GitHub y el trabajo en equipo fue esencial para gestionar los objetivos del proyecto y cumplirlos de manera eficiente. El trabajo colaborativo en GitHub, junto con la gestión de objetivos y tareas en herramientas como Trello, me preparó para abordar el proyecto de Visionova de manera organizada y efectiva. Estas habilidades fueron cruciales para coordinar el desarrollo, integrar diferentes partes del sistema y asegurar que se cumplieran los plazos y objetivos del proyecto.

6.7.7 Tecnologies Multimedia (TM)

La asignatura de Tecnologies Multimedia fue particularmente útil para la integración de funciones multimedia en Visionova. Los conocimientos adquiridos sobre la conversión de imágenes a base64 y la manipulación de datos multimedia me permitieron implementar características avanzadas en la aplicación, como la captura y almacenamiento de imágenes y la conversión de rutas a formatos visuales atractivos. Estas habilidades fueron importantes para mejorar la funcionalidad y la usabilidad de Visionova.

6.7.8 Programación en General

Las asignaturas de programación, como Programación, Estructura de Dades y Algorismica Avanzada, Inteligencia Artificial… Proporcionaron la base necesaria para desarrollar las habilidades de programación requeridas para implementar la lógica de Visionova. Estas asignaturas me enseñaron a escribir código eficiente, a estructurar datos de manera efectiva y a resolver problemas algorítmicos complejos. La formación en estas áreas fue crucial para el desarrollo de la lógica central de Visionova y para la implementación de sus funcionalidades clave.

Es decir, una vez se ha contemplado de forma general todos los casos específicamente de como me ha ayudado la carrera en este proyecto, puedo decir felizmente que, el Grado en Ingeniería Informática ha sido fundamental para el desarrollo de Visionova. Cada asignatura ha aportado conocimientos y habilidades específicas que han sido directamente aplicables al proyecto. Desde la organización y planificación del desarrollo, hasta la implementación técnica y la resolución de problemas, la formación recibida en el grado ha sido esencial para llevar a cabo este TFG de manera exitosa y más llevadera.

7. Conclusiones, discusión y trabajo futuro

Desarrollar una aplicación dirigida a usuarios con visibilidad reducida ha presentado numerosos desafíos únicos que requieren una atención minuciosa a detalles que normalmente no se considerarían en el desarrollo de aplicaciones convencionales. A través de este trabajo, he podido establecer las bases y generar una herramienta útil que puede significativamente mejorar la vida diaria de las personas con discapacidad visual. Visionova, aunque aún no está completamente terminada, ha demostrado ser una solución viable y efectiva para abordar muchas de las necesidades de este colectivo.

El desarrollo de Visionova ha implicado una extensa revisión de todo durante el desarrollo y un análisis detallado de las tecnologías disponibles, así como una planificación meticulosa para garantizar que cada componente de la aplicación sea accesible y funcional para usuarios con discapacidad visual. Este proceso ha sido esencial para entender las limitaciones y oportunidades de las tecnologías utilizadas, permitiendo crear una herramienta que no solo cumple con las expectativas iniciales, sino que también destaca en un mercado con pocas soluciones integrales.

Los experimentos realizados para evaluar la usabilidad y funcionalidad de Visionova han confirmado que, aunque hay áreas que requieren más desarrollo y optimización, la aplicación ya proporciona una asistencia significativa en tareas diarias como la identificación de objetos y la navegación. Estos resultados subrayan la importancia de continuar puliendo y ampliando las capacidades de Visionova, asegurando que pueda ofrecer un soporte aún más robusto y versátil para sus usuarios tanto con discapacidad visual como los que no la padecen..

A pesar de no haber alcanzado el 100% de las funcionalidades inicialmente previstas, estoy satisfecho con el progreso logrado. La intención original incluía características avanzadas como el seguimiento preciso del usuario en tiempo real, que no se pudieron implementar completamente debido a limitaciones técnicas y de tiempo. Sin embargo, el hecho de que Visionova esté casi completa y operativa es un logro significativo, considerando la complejidad y las especificaciones únicas del proyecto y no solo por eso sino porque el estudio de mercado y el estado del arte realizados al inicio del proyecto no identificaron ninguna aplicación existente con las características y funcionalidades integradas que Visionova ofrece. Esto hace que Visionova no solo sea una herramienta innovadora, sino también una solución única en el mercado, lo que refuerza su potencial impacto positivo en la vida de las personas con discapacidad visual. La documentación y el estudio de los sistemas utilizados han sido cruciales para el desarrollo de esta aplicación. Sin la consulta exhaustiva de recursos y la adquisición de nuevos conocimientos en diversas tecnologías, no habría sido posible alcanzar el nivel de desarrollo actual. Este proceso no solo ha sido fundamental para la creación de Visionova, sino que también ha enriquecido mi formación y habilidades en el desarrollo de software accesible y en la gestión de proyectos tecnológicos complejos.

Mirando hacia el futuro, estoy comprometido a seguir desarrollando Visionova, con la firme convicción de que puede ser una herramienta esencial para las personas con discapacidad visual. Además, he estructurado el proyecto de manera que otros

desarrolladores puedan colaborar y continuar mejorando la aplicación. Esto incluye una organización meticulosa del repositorio y la implementación de principios de desarrollo de software que faciliten la comprensión y expansión del código.

Entre los próximos pasos, planeo trabajar en la integración de nuevas funcionalidades que no pudieron ser completadas en esta primera fase, como el seguimiento preciso del usuario y la mejora de los algoritmos de reconocimiento de objetos. También tengo la intención de realizar pruebas de usuario más extensas para pulir aún más la usabilidad de la aplicación y asegurarme de que satisface las necesidades de sus usuarios de la mejor manera posible.

Bibliografía

[1] Feliz Vita : Aplicaciones para ciegos. Disponible en: https://felizvita.com/aplicaciones-para-ciegos/ [Consulta: 28 de enero de 2024]

[2] Infotecnovision : Recopilación de aplicaciones móviles que facilitan la vida a personas ciegas y con baja visión. Disponible en: https://infotecnovision.com/recopilacion-de-aplicaciones-moviles-que-facilitan-la-vida-a-personas-ciegas-y-con-baja-vision/ [Consulta: 28 de enero de 2024]

[3] Be My Eyes : Be My Eyes en español. Disponible en: https://www.bemyeyes.com/language/spanish [Consulta: 30 de enero de 2024]

[4] Mapp4all : Mapp4all Features. Disponible en: https://mapp4all.app/#features [Consulta: 4 de febrero de 2024]

[5] TapTapSee : TapTapSee. Disponible en: https://taptapseeapp.com/ [Consulta: 5 de febrero de 2024]

[6] LazarilloGPS : Lazarillo App. Disponible en: https://lazarillo.app/es/app/ [Consulta: 6 de febrero de 2024]

[7] BeSpecular: BeSpecular App. Disponible en: https://www.bespecular.com [Consulta: 6 de febrero de 2024]

[8] Envision: Envision. Disponible en: https://www.letsenvision.com/app [Consulta: 7 de febrero de 2024]

[9] Doonamis : React Native: Qué es, ventajas y desventajas. Disponible en: https://www.doonamis.com/react-native-que-es-ventajas-desventajas/ [Consulta: 13 de febrero de 2024]

[10] Back4App : Ventajas y desventajas de React Native reveladas. Disponible en: https://blog.back4app.com/es/react-native-ventajas-y-desventajas-reveladas/ [Consulta: 14 de febrero de 2024]

[11] KeepCoding : Ventajas e inconvenientes de React Native. Disponible en: https://keepcoding.io/blog/ventajas-e-inconvenientes-de-react-native/ [Consulta: 14 de febrero de 2024]

[12] Back4App : Los 10 principales lenguajes de desarrollo de aplicaciones. Disponible en: https://blog.back4app.com/es/los-10-principales-lenguajes-de-desarrollo-de-aplicaciones/ [Consulta: 16 de febrero de 2024]

[13] Immune Institute : Lenguajes de programación para móvil. Disponible en: https://immune.institute/blog/lenguajes-de-programacion-para-movil/ [Consulta: 17 de febrero de 2024]

[14] ED Team : 6 lenguajes para desarrollo móvil. Disponible en: https://ed.team/blog/6-lenguajes-para-desarrollo-movil [Consulta: 18 de febrero de 2024]

[15] SlashMobility : Pros y contras de Flutter. Disponible en: https://slashmobility.com/blog/2019/06/pros-y-contras-de-flutter/ [Consulta: 20 de febrero de 2024]

[16] Redwerk : Ventajas y desventajas del Flutter. Disponible en: https://redwerk.es/blog/ventajas-y-desventajas-del-flutter/ [Consulta: 20 de febrero de 2024]

[17] Androides: Flutter vs Xamarin. Disponible en: https://www.3androides.com/actualidad/236-flutter-vs-xamarin [Consulta: 22 de febrero de 2024]

[18] Inmediatum : Ventajas y desventajas de apps desarrolladas en Xamarin. Disponible en: https://inmediatum.com/blog/ingenieria/ventajas-y-desventajas-de-apps-desarrolladas-en-xamarin/ [Consulta: 24 de febrero de 2024]

[19] Back4App : Xamarin vs React Native. Disponible en: https://blog.back4app.com/es/xamarin-vs-react-native/ [Consulta: 24 de febrero de 2024]

[20] Imagina Formación : Flutter, React Native o Xamarin: ¿Cuál es el mejor framework para desarrollar una aplicación móvil?. Disponible en: https://imaginaformacion.com/tutoriales/flutter-react-native-o-xamarin-cual-es-el-mejor-framework-para-desarrollar-una-aplicacion-movil [Consulta: 18 de febrero de 2024]

[21] Wikipedia: NativeScript. Disponible en: https://es.wikipedia.org/wiki/NativeScript [Consulta: 25 de febrero de 2024]

[22] Profile : ¿Qué es Ionic?. Disponible en: https://profile.es/blog/que-es-ionic/ [Consulta: 26 de febrero de 2024]

[23] AvantgardeIT: SwiftUI: Un cambio de paradigma. Disponible en: https://avantgardeit.es/swiftui-un-cambio-de-paradigma/ [Consulta: 26 de febrero de 2024]

[24] Android Developers: Kotlin Multiplatform Overview. Disponible en: https://developer.android.com/kotlin/overview?hl=es-419 [Consulta: 26 de febrero de 2024]

[25] Luis Lizama (2023): Introducción a Expo: Construyendo aplicaciones móviles fácilmente con JavaScript y React Native. Disponible en: https://luislizama.com/2023/05/05/Introduccion-a-Expo-construyendo-aplicaciones-moviles-facilmente-con-JavaScript-y-React-Native/ [Consulta: 28 de febrero]

[26] Tuts+ : Desarrollo más fácil con Expo en React Native. Disponible en: https://code.tutsplus.com/es/easier-react-native-development-with-expo--cms-30546t [Consulta: 11 de marzo de 2024]

[27] FaztWeb: React Native Expo. Disponible en: https://faztweb.com/contenido/react-native-expo [Consulta: 14 de marzo de 2024]

[28] InnoCV : Duelo de desarrolladores: React Native vs Desarrollo nativo. Disponible en: https://www.innocv.com/web/noticias/duelo-de-desarrolladores-react-native-vs-desarrollo-nativo [Consulta: 2 de marzo de 2024]

[29] Softworth Solutions : Expo vs React Native CLI. Disponible en: https://medium.com/@softworthsolutionspvtltd/expo-vs-react-native-cli-7e47c7630039 [Consulta: 4 de marzo de 2024]

[30] Reintech : Cómo usar React Native con la CLI de React Native. Disponible en: https://reintech.io/blog/how-to-use-react-native-with-the-react-native-cli [Consulta: 6 de marzo de 2024]

[31] CampusMVP : React Native y Expo: ¿Qué son y cómo se relacionan?. Disponible en: https://www.campusmvp.es/recursos/post/react-native-y-expo-que-son-y-como-se-relacionan.aspx [Consulta: 28 de febrero de 2024]

[32] OnMind: React Native & Expo. Disponible en: https://onmind.co/doc/code/es/ReactNative&Expo.md [Consulta: 29 de febrero de 2024]

[33] NeoCodevs : Desarrollo móvil con Expo y React Native. Disponible en: https://www.linkedin.com/pulse/desarrollo-móvil-con-expo-react-native-neocodevs/?originalSubdomain=es [Consulta: 7 de marzo de 2024]

[34] CampusMVP : React Native y Expo: ¿Qué son y cómo se relacionan?. Disponible en: https://www.campusmvp.es/recursos/post/react-native-y-expo-que-son-y-como-se-relacionan.aspx [Consulta: 10 de marzo de 2024]

[35] Visual Paradigm : Free Use Case Diagram Tool. Disponible en: https://online.visual-paradigm.com/es/diagrams/solutions/free-use-case-diagram-tool/ [Consulta: 8 de febrero de 2024]

[35] GoodBarber : Cómo elegir los colores adecuados para tu aplicación móvil. Disponible en: https://es.goodbarber.com/blog/como-elegir-los-colores-adecuados-para-tu-aplicacion-movil-a814/ [Consulta: 17 de marzo de 2024]

[36.5] Paletas de Colores : Paletas de colores. Disponible en: https://paletasdecolores.com/category/siniy/ [Consulta: 18 de marzo de 2024]

[37] HTML Color Codes : HTML Color Codes. Disponible en: https://htmlcolorcodes.com/es/ [Consulta: 23 de marzo de 2024]

[38] Xataka: ¿Qué es una API?. Disponible en: https://www.xataka.com/basics/api-que-sirve [Consulta 1 de Abril de 2024]

[39] Ia Cognitive Services: Introducción a los servicios cognitivos de Azure. Disponible en: https://www2.deloitte.com/es/es/pages/technology/articles/introduccion-a-los-servicios-cognitivos-de-azure.html [Consulta 2 de Abril de 2024]

[40] Amazon Rekognition: ¿Qué es Amazon Rekognition?. Disponible en: https://docs.aws.amazon.com/es_es/rekognition/latest/dg/what-is.html [Consulta 2 de Abril de 2024]

[41] Cursin: Ibm Watson, OpenCV.. Disponible en: https://cursin.net/aprende-a-utilizar-python-watson-ai-y-opencv-en-el-curso-gratuito-de-ibm/ [Consulta 4 de Abril de 2024]

[42] Google Cloud Vision : Google CloudVision Authentication. Disponible en: https://cloud.google.com/vision/docs/authentication?hl=es-419 [Consulta: 5 de abril de 2024]

[43] Google Cloud Text-to-Speech : Google Cloud Text-to-Speech API. Disponible en: https://cloud.google.com/text-to-speech?hl=es_419 [Consulta: 14 de abril de 2024]

[44] Aníbal Ramírez (2020): "Integrar mapas en React Native usando React Native Maps 2020". Medium, 26 de abril. Disponible en: https://medium.com/@anlramirezs11/integrar-mapas-en-react-native-usando-react-native-maps-2020-9ed9c81ce476 [Consulta: 26 de abril de 2024]

[45] Firebase : Firebase Setup for Android. Disponible en: https://firebase.google.com/docs/android/setup?hl=es [Consulta: 19 de mayo de 2024]

[46] Expo Documentation: Todas las API's de Expo. Disponible en: https://docs.expo.dev/versions/latest/sdk/media-library/ [Consulta: 19 de mayo de 2024]

[47] Comparativa: Amazon Rekognition, Azure Vision, Google Cloud Vision. Disponible en: https://www.linkedin.com/pulse/comparative-example-amazon-rekognition-azure-vision-google-vasilescu/ [Consulta: 30 de Abril de 2024]

[48] Comparativa detector texto: Text-to-Speech alternatives. Disponible en: https://unrealspeech.com/compare/amazon-polly-text-to-speech-vs-google-text-to-speech#:~:text=Based%20on%20the%20Mean%20Opinion,and%20non-fiction%20voice%20quality. [Consulta: 30 de Abril 2024]

[49] Comparativa: React Native Maps y Expo Location API vs. Mapbox y AWS Location Service. Disponible en: https://stackoverflow.com/questions/75937899/what-is-the-best-way-to-display-a-map-in-rn [Consulta: 5 de mayo de 2024]

[50] Comparativa: Expo File System API vs. Open Source Alternatives. Disponible en: https://www.echowaves.com/post/simple-alternative-to-react-native-async-storage [Consulta: 6 de mayo de 2024]

[51] Comparativa: Firebase vs. AWS Amplify y Backendless. Disponible en: https://blog.back4app.com/es/comparacion-entre-aws-amplify-y-google-firebase/ [Consulta: 6 de mayo de 2024]

[52] Alternativa: Vision AI by Clarifai. Disponible en: https://www.f6s.com/company/clarifai-computervisionai#about [Consulta: 7 de mayo de 2024]

[53] Gallardo Ramos, D. : Cómo comenzar un proyecto en React Native. Disponible en: https://gallardoramos.com/comenzar-proyecto-react-native/ [Consulta: 20 de marzo de 2024]

[54] MrBrent981 (2019): Configuring Android Emulator in Expo, [Vídeo online]. Disponible en: https://www.youtube.com/watch?v=wb6xKsdXfOw [Consulta: 22 de marzo de 2024]

[55] FaztWeb : React Native Expo, [Vídeo online]. Disponible en: https://www.youtube.com/watch?v=ql4J6SpLXZA [Consulta: 21 de mayo de 2024]

[56] Cristian León (2020): "React Native Stack Navigation with Tab Navigation". Medium, 23 de abril. Disponible en: https://cristhianleonli.medium.com/react-native-stack-navigation-with-tab-navigation-31e5fb9927a4 [Consulta: 23 de abril de 2024]

[57] Academind : React Native Camera API, [Vídeo online]. Disponible en: https://www.youtube.com/watch?v=4VtX3SjeMdY [Consulta: 28 de marzo de 2024]

[58] SitePoint : Started with Google Cloud's Text-to-Speech API. Disponible en: https://www.sitepoint.com/started-with-google-clouds-text-to-speech-api/ [Consulta: 13 de abril de 2024]

[59] GitHub : "Map Integration in React Native". Disponible en: https://github.com/betomoedano/react-native-camera/blob/main/App.js [Consulta: 12 de abril de 2024]

[60] Google Developers : Google Maps Documentation. Disponible en: https://developers.google.com/maps/documentation/javascript/react-map?hl=es-419 [Consulta: 27 de abril de 2024]

[61] Pallavi Khedle : "Google Map Integration with React Native Expo: Mobile and Web". Medium, 11 de mayo. Disponible en: https://medium.com/@pallavi8khedle/google-map-integration-with-react-native-expo-mobile-and-web-348a920c84c0 [Consulta: 11 de mayo de 2024]

[62] Expo Documentation : Expo SDK Upgrade Guide. Disponible en:

https://docs.expo.dev/workflow/upgrading-expo-sdk-walkthrough/ [Consulta: 6 de mayo de 2024]

[63] Sarmila Sivaraja (2020): "Integrating Real-Time Map with User Location in a React Native App". Medium, 9 de mayo. Disponible en: https://medium.com/@Sarmilasivaraja/integrating-real-time-map-with-user-location-in-a-react-native-app-d0bef63ba3b2 [Consulta: 9 de mayo de 2024]

[64] Burcu Ozdmr : "How to Draw Directions Route on a Map in React Native". Medium, 13 de mayo. Disponible en: https://medium.com/@burcuozdmr/how-to-draw-directions-route-on-a-map-in-react-native-a52f46706cf5 [Consulta: 13 de mayo de 2024]

[65] Principios SOLID: 5 claves. Disponible en: https://thepower.education/blog/principios-solid [Consulta: 13 de mayo de 2024]

[66] Auth0 : Auth0 Integration Guide. Disponible en: https://manage.auth0.com/welcome/ [Consulta: 18 de mayo de 2024]

[67] Code Academy : Auth0 Setup Guide, [Vídeo online]. Disponible en: https://www.youtube.com/watch?v=W6KUjYnGbkk [Consulta: 20 de mayo de 2024]

[68] Code Academy : Auth0 Authentication Implementation, [Vídeo online]. Disponible en: https://www.youtube.com/watch?v=wr3VmbZdVA4 [Consulta: 20 de mayo de 2024]

[69] Gallardo Ramos, D. : Cómo comenzar un proyecto en React Native. Disponible en: https://gallardoramos.com/comenzar-proyecto-react-native/ [Consulta: 10 de febrero de 2024]

[70] React Native Voice : Voice Project on GitHub. Disponible en: https://github.com/react-native-voice/voice [Consulta: 8 de febrero de 2024]

Anexo

CU-USER-03	Cerrar sesión de la aplicación
Descripción	El usuario cierre de sesión en la aplicación
Precondición	El usuario ha iniciado sesión en la aplicación
Postcondición	El usuario ha cerrado sesión y está fuera de la aplicación
Requisitos relacionados	RF-USER-03

Escenario principal
1. El usuario accede a la configuración de su perfil en la aplicación Visionova. 2. Se presenta al usuario la opción de cerrar sesión. 3. El usuario confirma su decisión de cerrar sesión. 4. La aplicación cierra la sesión del usuario y lo redirige a la pantalla de inicio de sesión.
Escenario alternativo
N/A

***Figura A.3**: RF-USER-013 Caso de uso de cerrar sesión de la aplicación.*

CU-USER-05	Grabar rutas
Descripción	Grabación de rutas personalizadas
Precondición	El usuario está dentro de la aplicación.
Postcondición	El usuario está dentro de la aplicación. Postcondición: Se ha grabado una nueva ruta en la aplicación.
Requisitos relacionados	RF-USER-05

Escenario principal
1. El usuario accede a la función de grabación de rutas en la pantalla principal de la aplicación Visionova. 2. La aplicación activa el GPS del dispositivo y comienza a registrar la ubicación del usuario.

3. El usuario realiza su recorrido habitual, marcando los puntos de interés durante el trayecto. 4. El usuario finaliza la grabación de la ruta. 5. La aplicación guarda la ruta grabada en la base de datos del usuario para su posterior acceso.
Escenario alternativo
4.1. El usuario decide cancelar la grabación de la ruta. 4.2. La aplicación detiene la grabación y descarta los datos registrados.

***Figura A.5**: RF-USER-05. Caso de uso de grabar rutas*

CU-USER-06	Guardar rutas
Descripción	Guardar rutas en la aplicación para poder visualizarlas después
Precondición	El usuario ha grabado una ruta en la aplicación
Postcondición	La ruta grabada se guarda correctamente en la aplicación para futuras referencias y navegación
Requisitos relacionados	RF-USER-06

Escenario principal
1. Después de grabar una ruta, el usuario decide dar a la opción de guardar ruta en la pantalla de finalización de grabación. 2. La aplicación solicita al usuario que ingrese un nombre descriptivo para la ruta. 3. El usuario proporciona un nombre y confirma la acción de guardar la ruta. 4. La aplicación almacena la ruta con el nombre proporcionado en la base de datos del usuario. 5. Se muestra un mensaje de confirmación de que la ruta se ha guardado exitosamente.
Escenario alternativo
3.1. El usuario decide cancelar la acción de guardar la ruta. 3.2. La aplicación cancela el proceso de guardado y vuelve a la pantalla de finalización de grabación de ruta.

***Figura A.6:** RF-USER-06. Caso de uso de guardar rutas*

CU-USER-07	Ver rutas
Descripción	Visualización de rutas de diferentes usuarios
Precondición	El usuario está dentro de la aplicación y ha iniciado sesión
Postcondición	El usuario puede ver las rutas grabadas por otros usuarios
Requisitos relacionados	RF-USER-07, RF-GUEST-02

Escenario principal
1. El usuario accede a la sección de rutas de la aplicación. 2. La aplicación muestra una lista de las rutas disponibles, incluidas las del usuario y las de otros usuarios. 3. El usuario selecciona una ruta específica de la lista para ver más detalles. 4. La aplicación muestra información detallada sobre la ruta seleccionada, incluida la ubicación, la distancia, etc. 5. El usuario puede explorar y visualizar la ruta en el mapa dentro de la aplicación.
Escenario alternativo
3.1. No hay rutas disponibles de otros usuarios. 3.2. La aplicación muestra un mensaje indicando que no hay rutas disponibles de otros usuarios en ese momento.

Figura A.7: *RF-USER-07. Caso de uso de ver rutas*

CU-USER-08	Perfil personal
Descripción	Gestión de rutas guardadas en el perfil del usuario
Precondición	El usuario ha iniciado sesión en la aplicación y tiene rutas guardadas en su perfil
Postcondición	El usuario puede acceder y gestionar sus rutas guardadas dentro de la sección de perfil de la aplicación
Requisitos relacionados	RF-USER-08

Escenario principal

1. El usuario accede a la sección de perfil dentro de la aplicación. 2. En la sección de perfil, el usuario selecciona la opción de "Mis rutas". 3. La aplicación muestra una lista de las rutas guardadas por el usuario, junto con información adicional como la distancia, la duración, etc. 4. El usuario puede seleccionar una ruta específica de la lista para ver más detalles o realizar acciones adicionales. 5. La aplicación proporciona opciones para editar o eliminar la ruta seleccionada. 6. El usuario realiza la acción deseada (editar o eliminar) sobre la ruta seleccionada.
Escenario alternativo
5.1. El usuario decide cancelar la acción sobre la ruta. 5.2. La aplicación cancela la acción y regresa a la lista de rutas guardadas del usuario.

Figura A.8: *RF-USER-08. Caso de uso de registro en la aplicación*

CU-USER-09	Ver seguidores
Descripción	Visualización de seguidores y usuarios seguidos en el perfil del usuario
Precondición	El usuario ha iniciado sesión en la aplicación
Postcondición	El usuario puede ver una lista de sus seguidores y a quién sigue dentro de la sección de perfil de la aplicación.
Requisitos relacionados	RF-USER-09

Escenario principal
1. El usuario accede a la sección de perfil dentro de la aplicación. 2. En la sección de perfil, el usuario selecciona la opción de "Seguidores" o "Siguiendo". 3. La aplicación muestra una lista de los usuarios que siguen al usuario o a quienes el usuario sigue, respectivamente. 4. El usuario puede explorar la lista de seguidores o usuarios seguidos y ver sus perfiles individuales si lo desea. 5. El usuario puede volver atrás para regresar a su perfil. 6. El usuario repite los pasos anteriores para ver a quién sigue o sus seguidores dentro de la aplicación.

Escenario alternativo
3.1. El usuario no tiene seguidores o usuarios seguidos. 3.2. La aplicación muestra un mensaje indicando que el usuario no tiene seguidores o usuarios seguidos en ese momento.

***Figura A.9**: RF-USER-09. Caso de uso de ver seguidores*

CU-USER-10	Seguir usuarios
Descripción	Seguir a otros usuarios
Precondición	El usuario ha iniciado sesión
Postcondición	El usuario sigue a otro usuario y puede ver sus publicaciones en su feed
Requisitos relacionados	RF-USER-10

Escenario principal
1. El usuario navega por la aplicación y encuentra el perfil de otro usuario que desea seguir. 2. En el perfil del otro usuario, el usuario selecciona la opción de "Seguir". 3. La aplicación registra la acción del usuario y actualiza la lista de usuarios seguidos en el perfil del usuario. 4. El usuario recibe una confirmación de que ha seguido con éxito al otro usuario. 5. El usuario puede ver las publicaciones y actividades del usuario seguido en su feed de la aplicación.
Escenario alternativo
1.1. El usuario ya sigue al perfil del otro usuario. 1.2. La aplicación le muestra que ya sigue al usuario. 1.3. El usuario puede volver hacia atrás.

***Figura A.10**: RF-USER-10. Caso de uso de seguir usuarios*

CU-USER-11	Ruta en favoritos
Descripción	Añadir rutas a una lista de favoritos
Precondición	El usuario ha iniciado sesión en la aplicación y tiene una ruta disponible para añadir a favoritos

Postcondición	La ruta seleccionada se añade con éxito a la lista de favoritos del usuario
Requisitos relacionados	RF-USER-11

Escenario principal
1. El usuario navega por la aplicación y accede a su lista de rutas guardadas en su perfil. 2. En la lista de rutas guardadas, el usuario selecciona la opción de "Añadir a favoritos" junto a la ruta que desea marcar como favorita. 3. La aplicación registra la acción del usuario y añade la ruta seleccionada a la lista de favoritos del usuario. 4. El usuario recibe una confirmación de que la ruta ha sido añadida con éxito a sus favoritos.
Escenario alternativo
3.a.1. La ruta seleccionada ya está marcada como favorita. 3.a.2. La aplicación muestra un mensaje indicando que la ruta ya está en la lista de favoritos del usuario. 3.a.3. El usuario confirma que desea mantener la ruta en la lista de favoritos. 3.a.4. La aplicación regresa al usuario a su lista de rutas guardadas.

***Figura A.11**: RF-USER-11. Caso de uso de ruta en favoritos*

***Figura 12**: Diagrama de los casos de uso.. Se ve representado todas las historias de usuario, tanto si son para usuarios, invitados o del propio sistema, y si tienen alguna relación entre ellas*

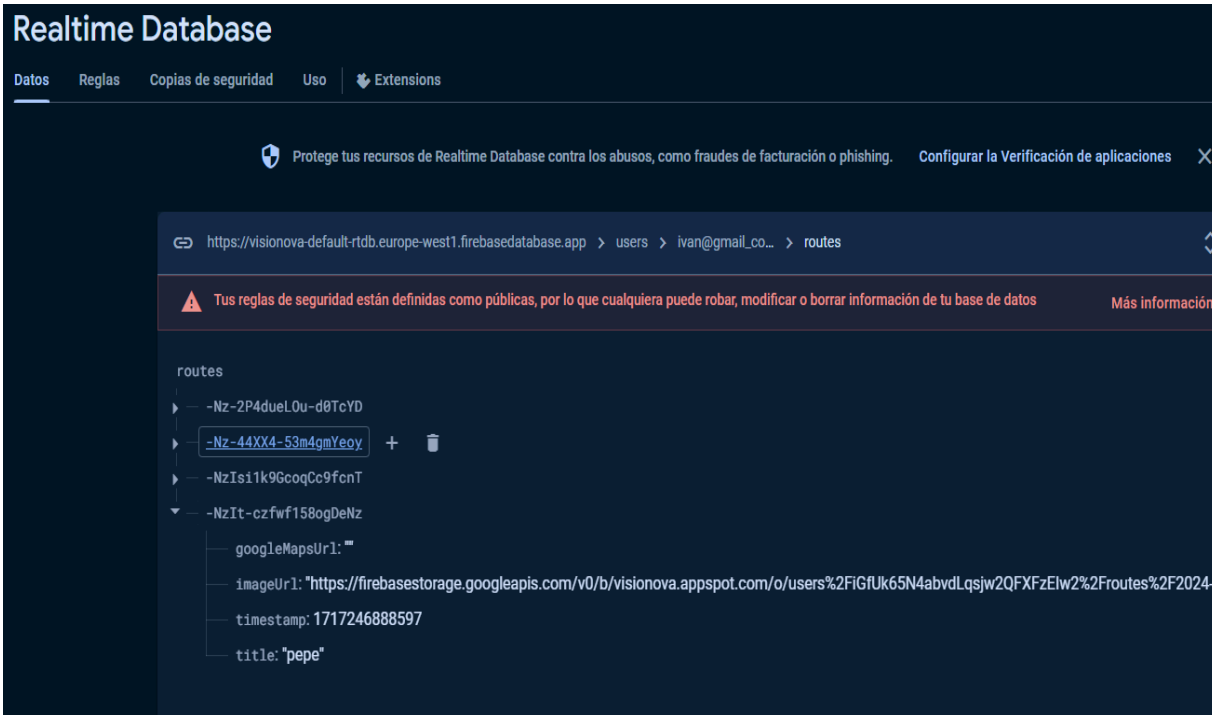


***Figura 36**: Dentro de la base de datos. Vista de como es el guardado de las rutas.*

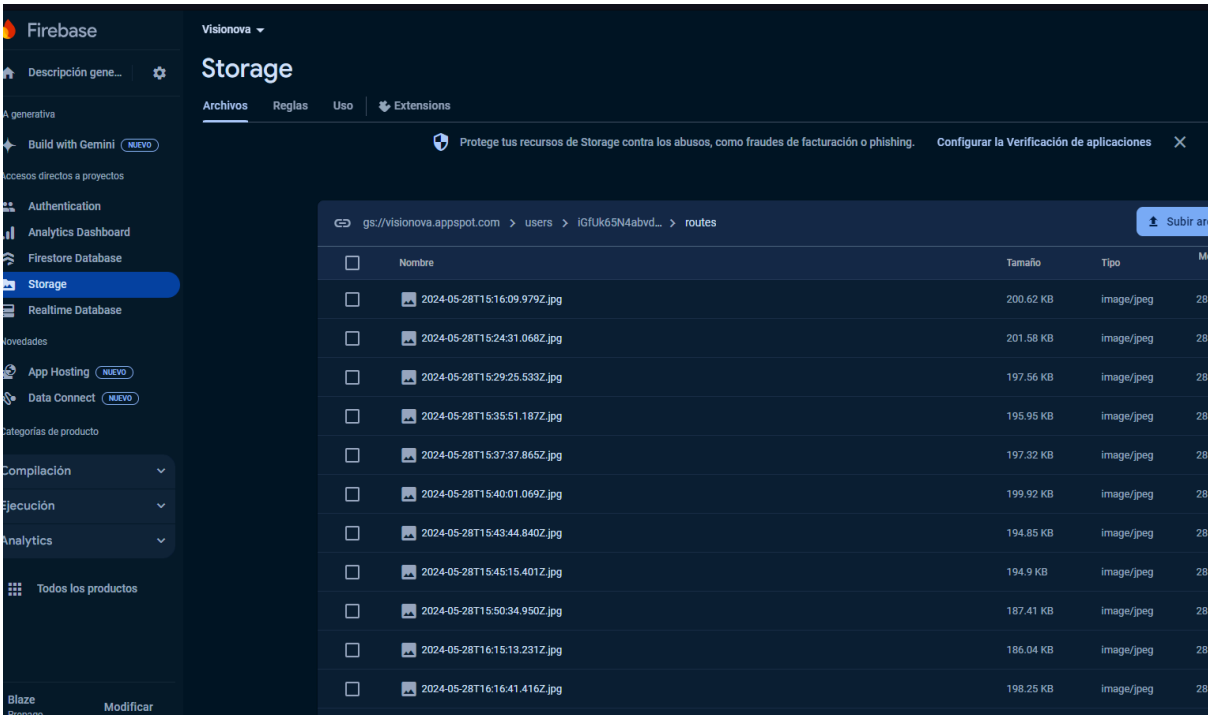


***Figura 37**: Firebase Storage. Almacenaje de las imágenes.*