UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

Departamento de Arquitectura de Computadores y Automática

TRABAJO DE FIN DE GRADO

AI See: aplicación para smart glasses de descripción del entorno para personas con baja visión

AI See: a smart glasses application for environment description for people with low vision

Notas finales de cada alumno en el trabajo:

Daniel Barroso Casado: 10, Laura de Cara Molina: 10, Álvaro Gómez Tejedor: 10, Hugo Herrero Desvoyes: 10

Dirigido por:
María Guijarro Mata-García,
Juan Bayón Fernández

Curso académico 2024-25

Convocatoria ordinaria

Agradecimientos

A nuestros tutores María Guijarro Mata-García y Juan Bayón Fernández, por habernos acompañado desde el principio en este proyecto. Gracias por estar siempre disponibles, orientarnos con paciencia y confiar en nosotros en cada paso. También agradecer a José Jiménez Jiménez, estudiante de doctorado, por su ayuda y apoyo en los laboratorios.

A nuestras familias, gracias por su cariño incondicional, su confianza en nosotros y por estar siempre ahí, tanto en los momentos de celebración como en los de dificultad. Sin su apoyo, esta etapa no habría sido la misma.

A nuestros compañeros durante estos años de carrera, gracias por compartir clases, trabajos, dudas y risas. En especial a aquellos con quienes hemos compartido más que asignaturas, ya que terminaron convirtiéndose en amigos y una parte tan importante de esta etapa.

Finalmente, a todas aquellas personas que, de una u otra forma, han formado parte de este proceso, gracias.

Resumen

AI See es una aplicación Android que desarrolla una funcionalidad de detección y descripción automática de imágenes orientada a personas con baja visión o ceguera, compatible con las gafas inteligentes Rockid Air. El objetivo principal del proyecto es fomentar la autonomía y seguridad de los usuarios mediante la conversión de contenido visual en descripciones auditivas comprensibles, utilizando modelos avanzados de inteligencia artificial. La aplicación permite la captura de imágenes tanto desde el móvil (galería o cámara), como desde la cámara de las gafas Rokid Air, las cuales son procesadas mediante el modelo Gemini, generando una descripción que se transmite por audio al usuario.

El desarrollo incluye también funcionalidades complementarias como gestión de usuarios, historial de descripciones, accesibilidad visual (modo alto contraste), soporte multilingüe y medidas de seguridad y privacidad. Además, se ha prestado atención a la experiencia del usuario mediante una interfaz accesible y adaptable a sus necesidades. El documento detalla el diseño, implementación, integración y validación de la solución propuesta, demostrando su utilidad como herramienta de asistencia visual portátil y accesible.

Palabras clave: descripción de imágenes, gafas inteligentes, Android, Gemini, inteligencia artificial, accesibilidad, asistencia visual, baja visión, ayuda auditiva.

Abstract

AI See is an Android application that implements an automatic image detection and description feature aimed at people with low vision or blindness, compatible with Rokid Air smart glasses. The main goal of the project is to promote user autonomy and safety by converting visual content into understandable auditory descriptions using advanced artificial intelligence models. The application allows image capture both from the mobile device (via gallery or camera) and from the Rokid Air glasses camera. The captured images are processed through the Gemini model, which generates descriptions that are delivered to the user via audio.

The development also includes complementary functionalities such as user management, description history, visual accessibility (high-contrast mode), multilingual support, and data privacy and security measures. Additionally, special attention has been given to user experience through an accessible and adaptable interface. This document details the design, implementation, integration, and validation of the proposed solution, demonstrating its usefulness as a portable and accessible visual assistance tool.

Keywords: image description, smart glasses, Android, Gemini, artificial intelligence, accessibility, visual assistance, low vision, auditory aid.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Motivación

El sentido de la vista es fundamental para la relación de las personas con su entorno exterior, ya sea el medio ambiente o las propias relaciones humanas, de forma que los problemas visuales afectan notablemente la vida de las personas que los padecen. Los problemas de visión pueden ser debidos a diversas causas: errores de refracción, cataratas, retinopatía diabética, glaucoma y degeneración macular por edad avanzada.

Según la Organización Mundial de la Salud (OMS o WHO), en el mundo hay aproximadamente 2.200 millones de personas con algún tipo de discapacidad visual de las que al menos 1.000 millones no están tratadas o podrían haberse evitado. La falta de visión produce una situación de deficiencia en la percepción de información que constituye un grave problema en la vida cotidiana de las personas afectadas. El costo anual estimado de la discapacidad visual es de unos US$ 411 000 millones en términos de productividad. [1]

Con este trabajo se pretende aportar al desarrollo de tecnologías de asistencia a personas con baja visión o ceguera para mejorar su calidad de vida, especialmente en cuanto a su autonomía y seguridad. Este es el objetivo de este trabajo fin de grado (TFG) que responde a una motivación de realizar un trabajo práctico que pueda beneficiar a personas con problemas más o menos graves de visión.

La descripción de imágenes puede ser una gran herramienta para las personas con baja visión, ya que les permite obtener información de su entorno. Esta funcionalidad puede ser de una gran utilidad en muchas situaciones cotidianas como: reconocer personas, identificar objetos, leer señales, tanto en una habitación de la casa como en el exterior, la calle o incluso en la naturaleza. Se aumenta así la independencia personal y reduce la necesidad de asistencia por parte de otras personas. Las descripciones precisas y detalladas mejoran la seguridad y capacidad de interactuar con el entorno, la inclusión social y la vida participativa de las personas con problemas visuales graves o moderados.

1.2. Objetivos y alcance del proyecto

El objetivo principal de este proyecto es desarrollar una funcionalidad de descripción de imágenes para personas con baja visión en una aplicación Android, diseñada para integrarse con Smart

Glasses. Este objetivo se desglosa en varios subobjetivos específicos:

1. Desarrollo de la funcionalidad de descripción de imágenes: implementar un sistema que capture imágenes del entorno y genere descripciones detalladas utilizando inteligencia artificial, que permita proporcionar información sobre el entorno, mejorando su autonomía y seguridad.
2. Integración con Smart Glasses: asegurar que la aplicación se comunique eficazmente con las smart glasses, permitiendo la captura de imágenes y la reproducción de descripciones a través de los altavoces integrados. Permitiendo ofrecer una solución portátil y accesible.
3. Accesibilidad de la interfaz: hacer un diseño accesible de la aplicación para personas con baja visión ofreciendo opciones de personalización, como modo alto contraste, para adaptarse a las necesidades específicas de cada usuario.
4. Gestión de usuarios y consultas: implementar una base de datos que permita gestionar usuarios, almacenar consultas y acceder a historiales de descripciones, facilitando el acceso a descripciones previas.
5. Implementación de medidas de seguridad de datos: desarrollar medidas de seguridad para proteger la información personal de los usuarios garantizando la privacidad y seguridad de la información almacenada en la aplicación.
6. Ofrecer la posibilidad de cambiar el idioma de la aplicación y de la descripción de las imágenes para llegar a un mayor número de usuarios.

El alcance del proyecto incluye el desarrollo completo de la aplicación, la integración con las Smart Glasses y la implementación de los objetivos mencionados. A continuación, se detallan los componentes específicos del alcance:

1. Desarrollo de la aplicación Android: la aplicación se desarrollará utilizando Java y Kotlin en Android Studio. Se implementarán las funcionalidades de captura de imágenes, generación de descripciones y reproducción de audio.
2. Integración con Smart Glasses: se utilizará el SDK proporcionado por el fabricante de las gafas para asegurar una comunicación fluida entre la aplicación y el hardware. Esta integración permitirá que las gafas capturen imágenes y reproduzcan las descripciones por los altavoces.
3. Implementación de la base de datos:se creará una base de datos para gestionar usuarios, consultas e historiales de descripciones. Esta base de datos permitirá a los usuarios acceder a descripciones previas y gestionar su información de manera eficiente.
4. Medidas de seguridad de datos: se implementarán medidas robustas de seguridad de datos, incluyendo la encriptación de datos sensibles y medidas contra posibles ataques.
5. Accesibilidad de la interfaz: garantizando compatibilidad con lectores de pantalla (como TalkBack de Android) y que todos los elementos interactivos tengan etiquetas accesibles (content descriptions) para facilitar su uso. Además de proporcionar la opción de alto contraste.
6. Implementación de cambio de idioma: se configurará el soporte multilenguaje mediante archivos de recursos en Android Studio.

El código fuente del proyecto se encuentra disponible en: `https://github.com/daniba02/TFG`

1.3. Metodología de trabajo

Para la organización y gestión del proyecto utilizamos Jira, una herramienta desarrollada por Atlassian que permite planificar, supervisar y gestionar proyectos de desarrollo de forma ágil y visual. Según IONOS, Jira es una plataforma especialmente útil en entornos colaborativos y de metodologías ágiles, facilitando la coordinación entre miembros del equipo y el seguimiento del progreso de las tareas.[2]

Nuestro trabajo se estructuró en periodos de tiempo definidos, al inicio de los cuales realizamos una reunión para decidir qué tareas íbamos a abordar durante ese periodo y repartir estas tareas entre los miembros del equipo. Dentro de Jira, configuramos un tablero de trabajo dividido en las siguientes columnas:

- Backlog: donde se recogen todas las tareas pendientes de abordar.
- En progreso: tareas en las que se está trabajando activamente.
- Bloqueado: tareas en las que no se puede continuar por algún motivo, ya sea técnico o de dependencia con otra tarea o persona.
- Revisar: tareas completadas, pendientes de ser revisadas por otro miembro del equipo.
- Listo: tareas finalizadas y validadas.

Este sistema visual permite tener una idea clara del estado del proyecto en todo momento y facilita la gestión de prioridades. De forma periódica, también se realizaban reuniones internas de seguimiento para comentar los avances, hablar de las posibles dificultades y colaborar ante bloqueos o dudas.

Este sistema visual nos permitió tener una idea clara del estado del proyecto en todo momento y facilitó la gestión de prioridades. Además, realizamos reuniones internas periódicas para comentar avances, resolver bloqueos y redefinir objetivos si era necesario.

En la Figura 1.1 se muestra el tablero de trabajo en uno de los sprints activos:

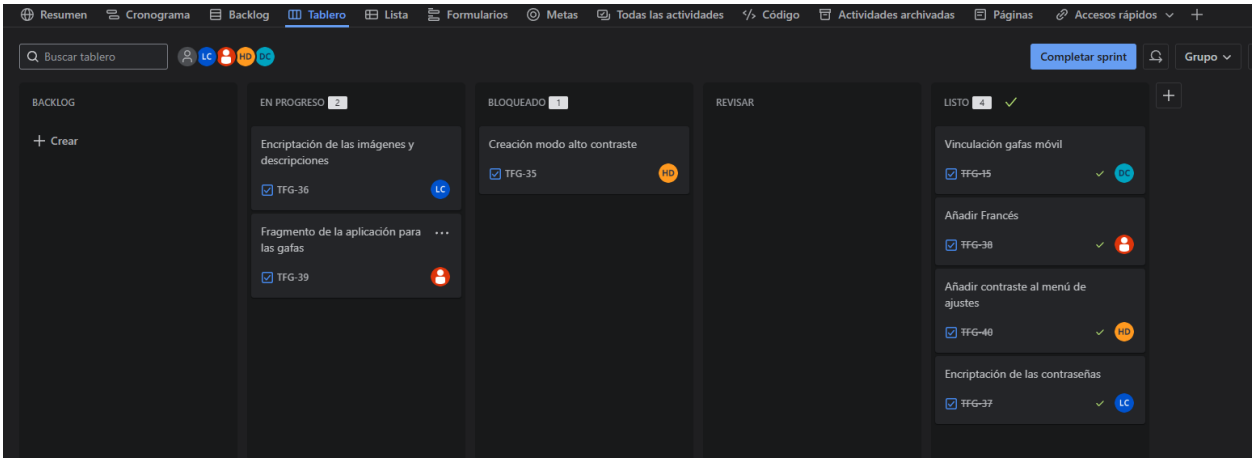

Figura 1.1: Tablero de Jira durante un sprint activo, con tareas distribuidas según su estado.

Resumen de los sprints realizados

Durante el desarrollo del proyecto se organizaron los sprints de la siguiente manera:

- **Sprint 1 - Inicio:** estudio de herramientas de desarrollo, diseño de interfaz, primeros scripts y conexión con APIs.
- **Sprint 2 - Base de la aplicación:** implementación del esqueleto de la app en Android Studio (pantallas base, integración inicial de Python).
- **Sprint 3 - Integración de funcionalidades:** migración del script de Python a Java, cambio de la IA a Gemini. Implementación de pantallas relacionadas con la base de datos y estudio de la accesibilidad de la aplicación.
- **Sprint 4 - Continuación integración funcionalidades:** generación de descripciones desde la cámara y galería. Creación del modo alto contraste y de la base de datos.
- **Sprint 5 - Gafas, Seguridad y Accesibilidad:** encriptación de datos, vinculación con las gafas inteligentes, mejoras de contraste e idioma.
- **Sprint 6 - UI y diseño final:** migración de las gafas a Kotlin, arreglos de accesibilidad, alertas y excepciones de error para el usaurio y pruebas de interfaz.
- **Sprint 7 - Pruebas y documentación:** arreglos finales, validaciones y pruebas, creación de diagramas técnicos, documentación de la aplicación y pruebas de rendimiento.

Estos sprints se etiquetaron dentro de Jira con sus tareas correspondientes como se menicona previamnete para poder seguir su evolución. No todas las tareas fueron terminadas dentro del tiempo que se había previsto del sprint debido a problemas técnicos y ajustes necesarios durante el desarrollo. En estos casos, las tareas pendientes se trasladaban al siguiente sprint.

Capítulo 2

Estado del arte

2.1. Tecnologías de asistencia para personas con discapacidad visual

La tecnología de asistencia para personas con discapacidad visual ha avanzado significativamente en los últimos años, ofreciendo soluciones innovadoras que mejoran su calidad de vida y autonomía. A continuación, se presentan algunas de las tecnologías más destacadas en este ámbito.

2.1.1. Dispositivos de reconocimiento táctil

Los dispositivos de reconocimiento táctil tuvieron una gran evolución desde las tradicionales pantallas braille, hasta los más recientes guantes hápticos. Por ejemplo, el proyecto BLITAB, ha desarrollado lo que ellos denominan "la primera tableta táctil para personas con discapacidad visual, capaz de convertir texto en braille en tiempo real". [3] El dispositivo está divido en dos partes. La zona superior está basada en una pantalla líquida que traduce en tiempo real al braille la información y contenidos (textos, imágenes y gráficos) que se encuentran en la parte inferior. [4]

En cuanto a los guantes hápticos, destaca el proyecto desarrollado por HaptX junto con la Universidad Old Dominion (ODU) y el Instituto de Tecnología de Georgia. Este proyecto busca crear una solución de "braille virtual"que permita a las personas con discapacidad visual explorar y navegar en entornos virtuales mediante la transmisión táctil de caracteres braille e imágenes digitales. Utilizando la tecnología de microfluidos de alta fidelidad patentada de HaptX y algoritmos de software de ODU, para mapear elementos léxicos y gráficos a una representación háptica precisa. [5]

2.1.2. Sistemas de navegación por audio

Los sistemas de navegación por audio, son herramientas diseñadas para asistir a personas con discapacidad visual en su orientación y movilidad, tanto en entornos exteriores como interiores mediante señales auditivas. Un ejemplo destacado es wayfindr, "una aplicación apoyada por la Unión Europea, para personas con discapacidad visual que les ayuda a moverse por las ciudades". [6]. El sistema se basa en la instalación de balizas Bluetooth (beacons) en lugares estratégicos, como estaciones de metro o centros comerciales. Estas balizas emiten señales que son detectadas por la

aplicación Wayfindr en el dispositivo móvil del usuario, proporcionando instrucciones auditivas paso a paso para guiarlo de manera segura a su destino. [7]

2.2. Smart Glasses en el mercado actual

Las gafas inteligentes o smart glasses han aparecido como una de las tecnologías con mayor futuro para la asistencia visual, combinando cámaras, procesamiento de información y sistemas de comunicación.

2.2.1. Evolución tecnológica

Desde el lanzamiento de Google Glass en 2013, la tecnología de gafas inteligentes ha evolucionado considerablemente. A partir de 2022, el mercado global de gafas inteligentes se valoró en aproximadamente $ 6.8 mil millones, con proyecciones que indican un crecimiento a alrededor de $ 31.5 mil millones para 2030, lo que refleja una tasa de crecimiento anual compuesta (CAGR) de más del 20 %. [8] Estos avances reflejan cómo la tecnología ha transformado las gafas inteligentes en herramientas esenciales para la inclusión y autonomía de las personas con discapacidad visual.

2.2.2. Principales dispositivos comerciales

Actualmente, varios dispositivos de gafas inteligentes están disponibles en el mercado, orientados tanto al consumidor general como a personas con discapacidades visuales. Entre los más relevantes se encuentra Envision Glasses, diseñadas específicamente para personas ciegas o con baja visión, que utilizan inteligencia artificial para leer textos, reconocer rostros y describir el entorno en tiempo real. [9]

Otro ejemplo son las Retiplus, las cuales contienen un sistema inteligente basado en tecnología de realidad aumentada orientado a personas con baja visión severa debido a patologías degenerativas de la retina como el Glaucoma, Retinosis Pigmentaria, Hemianopsias, Usher y otras afecciones de la visión causadas por Ictus y Daño Cerebral adquirido. Permite una rehabilitación visual funcional, mejorando la interacción con el entorno. [10]

También podemos encontrar modelos que no están diseñados explícitamente para personas con problemas de visión, como por ejemplo las Ray-Ban Meta Smart Glasses, desarrolladas en colaboración entre Meta y EssilorLuxottica, integran cámara, micrófonos y asistentes de voz. [11]

2.3. Aplicaciones móviles para asistencia visual

Las aplicaciones móviles han revolucionado la asistencia a personas con discapacidad visual, ofreciendo soluciones que van desde la navegación hasta el reconocimiento de objetos y lectura de textos. A continuación, se describen algunas de las más destacadas:

- **Be My Eyes** es una aplicación gratuita que conecta a personas ciegas o con baja visión con voluntarios videntes a través de videollamadas en vivo. Los usuarios pueden solicitar

asistencia para tareas cotidianas, como leer etiquetas o identificar objetos, recibiendo ayuda en tiempo real. [12]

- **LazarilloApp** proporciona información en tiempo real sobre la ubicación del usuario y su entorno. Utiliza GPS y síntesis de voz para guiar a las personas con discapacidad visual, ayudándolas a desplazarse de manera segura y autónoma. [13]
- **TapTapSee** es diseñada para identificar objetos mediante la cámara del dispositivo móvil. Los usuarios toman una fotografía y la aplicación describe el objeto en voz alta, siendo especialmente útil para reconocer productos, colores y otros elementos del entorno.[14]
- **Seeing AI**, desarrollada por Microsoft, utiliza inteligencia artificial para describir el mundo visual. Puede leer textos, identificar productos mediante códigos de barras, reconocer rostros y describir escenas, proporcionando una experiencia integral para usuarios con discapacidad visual. [15]

2.4. IA aplicada a la descripción de imágenes

La inteligencia artificial, particularmente las redes neuronales profundas, ha revolucionado la capacidad de describir imágenes automáticamente, un avance crucial para las personas con discapacidad visual. Los modelos de visión artificial actuales integran procesamiento de imágenes y lenguaje natural para generar descripciones comprensibles del entorno visual. A continuación se habla sobre algunos de los más importantes:

- From Your Eyes (FYE) es una empresa especializada en visión artificial que desarrolla soluciones basadas en la tecnología de Inteligencia Artificial que imitan la función visual del cerebro humano para reconocer, seguir y categorizar objetos y personas a través del procesamiento de imágenes de cámaras. "El modelo de IA de FYE permite procesar imágenes y vídeos en tiempo real, incluye un conjunto de más de 15 millones de datos, logra una precisión del 98,03 %" [16]
- Florence (Microsoft): es un modelo de visión artificial fundacional diseñado para abordar una amplia variedad de tareas de visión por computadora de manera generalizada. Ha sido entrenado con datos a escala web de imagen-texto, Florence puede adaptarse fácilmente a tareas como clasificación, detección de objetos, búsqueda, VQA, subtitulado de imágenes y reconocimiento de acciones en video. [17]
- Visual transformers: inicialmente se aplicó al campo de procesamiento de lenguaje natural, es un tipo de red neuronal profunda basada principalmente en el mecanismo de auto-atención. En el ámbito de la inteligencia artificial aplicada a imágenes, se han consolidado como una arquitectura clave para tareas como clasificación, segmentación y especialmente descripción automática de imágenes, donde se combinan con modelos de lenguaje para generar texto a partir de contenido visual. [18, 19]

2.5. Estudio de mercado

El mercado global de tecnologías de asistencia para personas con discapacidad visual está experimentando un importante crecimiento impulsado por un aumento en el envejecimiento de la

población y una creciente prevalencia de trastornos oculares. [20] Con el objetivo de identificar soluciones ya existentes para la asistencia visual mediante tecnologías inteligentes, se ha llevado a cabo un análisis del mercado actual. Este estudio pone el foco en dispositivos y aplicaciones dirigidos a personas con baja visión o ceguera, evaluando sus funcionalidades, precios y limitaciones.

2.5.1. OrCam MyEye 2

OrCam MyEye2 es un dispositivo innovador que se presenta como una de las opciones más avanzadas para la asistencia visual. Este dispositivo, que se coloca en la montura de unas gafas convencionales, actúa como una cámara inteligente que puede leer texto impreso, reconocer rostros, identificar objetos y billetes, y ofrecer descripciones detalladas de la escena circundante.

Características principales:

- Lectura de textos: el dispositivo puede leer textos de documentos, señales, pantallas y carteles. Funciona en condiciones normales de iluminación y requiere que el texto esté a una distancia de 30 a 40 cm de la cámara.
- Reconocimiento de rostros: OrCam es capaz de reconocer rostros previamente registrados y proporciona la información asociada a este, como el nombre de la persona.
- Identificación de productos y colores: permite identificar objetos si se ha realizado previamente un proceso de memorización.
- Autonomía y carga: la batería integrada tiene una duración de hasta dos horas de uso continuo y se carga en solo 40 minutos mediante un conector USB.
- Precio aproximado: $4.500 + IVA

[21, 22]

2.5.2. IrisVision

Ofrece una solución más asequible comparada con otros dispositivos avanzados. Este sistema se centra principalmente en la ampliación de imágenes, ayudando a las personas con baja visión a ver objetos y texto con un aumento de hasta 12 veces.

Características principales:

- Ampliación de imágenes: el sistema proporciona hasta 12x de aumento de la imagen, lo que permite a los usuarios ver detalles que de otro modo no sería posible.
- Modos personalizables: permite ajustar la imagen según las necesidades visuales del usuario, facilitando su uso tanto en interiores como en exteriores.
- Precio aproximado: $2.250

[23]

2.5.3. Microsoft Seeing AI

Seeing AI es una aplicación gratuita desarrollada por Microsoft que está diseñada para ayudar a personas con discapacidad visual mediante el uso de un smartphone.

Características principales:

- Lectura de textos: la función Short Text lee de manera instantánea cualquier texto detectado por la cámara.
- Identificación de productos: mediante la función Products, el sistema puede escanear códigos de barras y proporcionar información sobre el producto.
- Detección de personas: la app es capaz de identificar personas, indicando su edad, sexo y emociones.
- Escena y color: la función Scene describe lo que está ocurriendo en una escena, mientras que Color permite identificar colores.
- Solo compatible con iOS.
- Precio: gratuita

[15]

2.5.4. Retiplus

Retiplus es un sistema de rehabilitación visual funcional basado en Realidad Aumentada, diseñado específicamente para personas con baja visión severa. Este dispositivo está orientado a aquellos con patologías degenerativas de la retina, como glaucoma, retinosis pigmentaria o hemianopsias, y busca mejorar la calidad de vida de los usuarios mediante la optimización del campo visual.

Características principales:

- Realidad Aumentada: Utiliza la realidad aumentada para mejorar la visión de la persona, adaptándose a su patología específica.
- Rehabilitación visual: Se centra en la mejora funcional de la visión, ofreciendo ejercicios y ajustes personalizados según la enfermedad visual.
- Compatibilidad: Compatible con gafas inteligentes, lo que permite su uso en diversas situaciones.
- El precio del dispositivo es variable y depende de factores como el tratamiento específico y el entrenamiento necesario, ya que se adapta a las necesidades individuales de cada paciente.

[10]

El análisis del mercado actual muestra que existen múltiples opciones de dispositivos y aplicaciones que ayudan a personas con baja visión o ceguera, pero cada una de ellas presenta algunas limitaciones importantes.

Hay que analizar comparativamente las características de distintos dispositivos físicos. Por ejemplo OrCam MyEye 2 ofrece soluciones avanzadas pero a un precio elevado, lo que dificulta su acceso para una gran parte de la población. Por el contrario, IrisVision presenta una opción más asequible pero limitada a la ampliación de imágenes sin incluir funcionalidades de reconocimiento avanzado de objetos.

Las aplicaciones móviles, como Microsoft Seeing AI, son más económicas y ofrecen una amplia gama de funciones, pero requieren el uso constante de un smartphone y conexión a internet, lo que reduce su usabilidad en algunas circunstancias. Retiplus, mediante la Realidad Aumentada ofrece un enfoque novedoso, más allá la descripción de imágenes en tiempo real o la navegación asistida, su propósito principal es la rehabilitación visual personalizada adaptada a las necesidades concretas del usuario.

En general, el mercado está lleno de soluciones valiosas, pero la mayoría son caras o tienen limitaciones en su funcionalidad. Esto crea una oportunidad para el desarrollo de una solución híbrida que combine la portabilidad de aplicaciones móviles con la precisión y autonomía de dispositivos físicos, ofreciendo una descripción de imágenes en tiempo real integrada en las Smart Glasses Rockid air pro. Esta solución permite no solo mejorar la autonomía de las personas con baja visión, sino también facilitar el acceso a la tecnología para un público más amplio.

Capítulo 3

Análisis y Especificación

A la hora de desarrollar cualquier proyecto software, es muy importante definir de manera clara los requisitos que ha de cumplir el producto final, de esta manera podemos estructurar la idea del cliente para poder llevarlas a cabo durante el desarrollo del producto. Para saber qué funcionalidades tiene que realizar el sistema, se definen los requisitos funcionales, mientras que los requisitos no funcionales describen las características de calidad del sistema.

3.1. Análisis de requisitos funcionales

Aquí se detallan los requisitos funcionales que proporciona nuestro producto. Para ello hemos pensado en cuales pueden ser las necesidades de nuestros usuarios para así conocer qué funcionalidades van a poder cubrirlas eficazmente

- **RF-01 Registro de usuario**: un nuevo usuario debe ser capaz de registrarse en la aplicación, proporcionando su nombre y una contraseña segura.
- **RF-02 Inicio de sesión**: un usuario registrado debe ser capaz de iniciar sesión en la aplicación introduciendo su nombre de usuario y contraseña.
- **RF-03 Captura de imagen**: el usuario debe poder acceder a la cámara para capturar una imagen.
- **RF-04 Selección de imágenes desde la galería**: el usuario debe tener la opción de seleccionar imágenes almacenadas previamente en la galería del dispositivo Android.
- **RF-05 Recepción de descripciones por audio**: tras capturar o seleccionar una imagen, el usuario debe recibir la descripción de esta mediante salida de audio en el dispositivo.
- **RF-06 Conexión con Smart Glasses**: la aplicación debe permitir la conexión con las Smart Glasses Rockid air pro.
- **RF-07 Captura de fotografía mediante Smart Glasses**: el usuario debe ser capaz de capturar imágenes utilizando las gafas inteligentes y recibir la descripción por audio en las mismas.

- **RF-08 Acceso al historial de consultas**: los usuarios registrados deben poder consultar el historial de imágenes descritas anteriormente, accediendo tanto a las fotografías como a sus respectivas descripciones.
- **RF-09 Acceso al perfil del usuario**: los usuarios registrados deben poder acceder a su perfil a través del cual ver su información y poder modificar su contraseña o cerrar sesión.
- **RF-10 Cambio de contraste**: el usuario debe tener la opción de activar el modo de alto contraste dentro de la aplicación.
- **RF-11 Cambio de idioma**: el usuario debe poder seleccionar el idioma predeterminado para la navegación en la aplicación y para la reproducción de descripciones.

3.2. Análisis de requisitos no funcionales

En esta sección se detallan los requisitos no funcionales que tiene la aplicación. Aún siendo una aplicación con fines académicos, los requisitos no funcionales corresponden a los que debería tener una aplicación comercial del mismo ámbito.

- **Usabilidad**: la aplicación debe ser fácil de usar y navegar, permitiendo a cualquier usuario hacer uso de todas las funcionalidades sin necesidad de explicaciones explicitas.
- **Accesibilidad**: la aplicación debe ser accesible, en especial para personas con algún tipo de problema de visión.
- **Seguridad**: debe garantizar la seguridad y el correcto tratamiento de los datos del usuario que se recopilen para su correcto funcionamiento.
- **Rendimiento**: la aplicación debe ser responsiva y funcionar de forma fluida, sin tiempos de carga excesivos.
- **Mantenibilidad**: fácil de mantener y actualizar, permitiendo la incorporación de nuevas funcionalidades o la corrección de errores con facilidad.
- **Estabilidad**: debe estar libre de errores que puedan causar bloqueos o fallos en su funcionamiento.

3.3. Casos de uso

Esta sección describe los principales casos de uso que conforman la funcionalidad central de la aplicación. Estos se agrupan en dos grandes bloques: los casos relacionados con la descripción de imágenes, y los relacionados con la gestión de usuarios y el historial de descripciones, que hacen uso de la base de datos de la aplicación.

Casos de uso de descripción del entorno

La aplicación ofrece al usuario tres métodos principales para obtener una descripción auditiva de una imagen: mediante la cámara del dispositivo móvil, la selección de imágenes desde la galería,

y la captura a través de las gafas inteligentes Rokid Air. Una vez capturada o seleccionada la imagen, se genera una descripción que se reproduce en formato de audio.

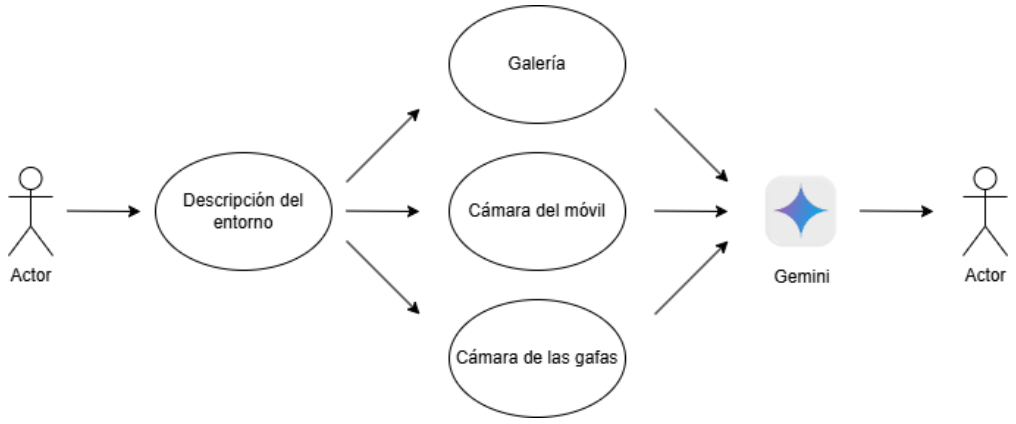

Figura 3.1: Diagrama de casos de uso de la aplicación – descripción del entorno.

Casos de uso de gestión de usuarios y consultas

Además de la funcionalidad principal de descripción de imágenes, la aplicación incorpora una serie de funciones relacionadas con la administración de usuarios y la gestión del historial de descripciones. Estas funciones incluyen el registro de nuevos usuarios, inicio de sesión, visualización y eliminación de consultas anteriores, modificación de perfil, y validación de credenciales, todo ello respaldado por una base de datos interna.

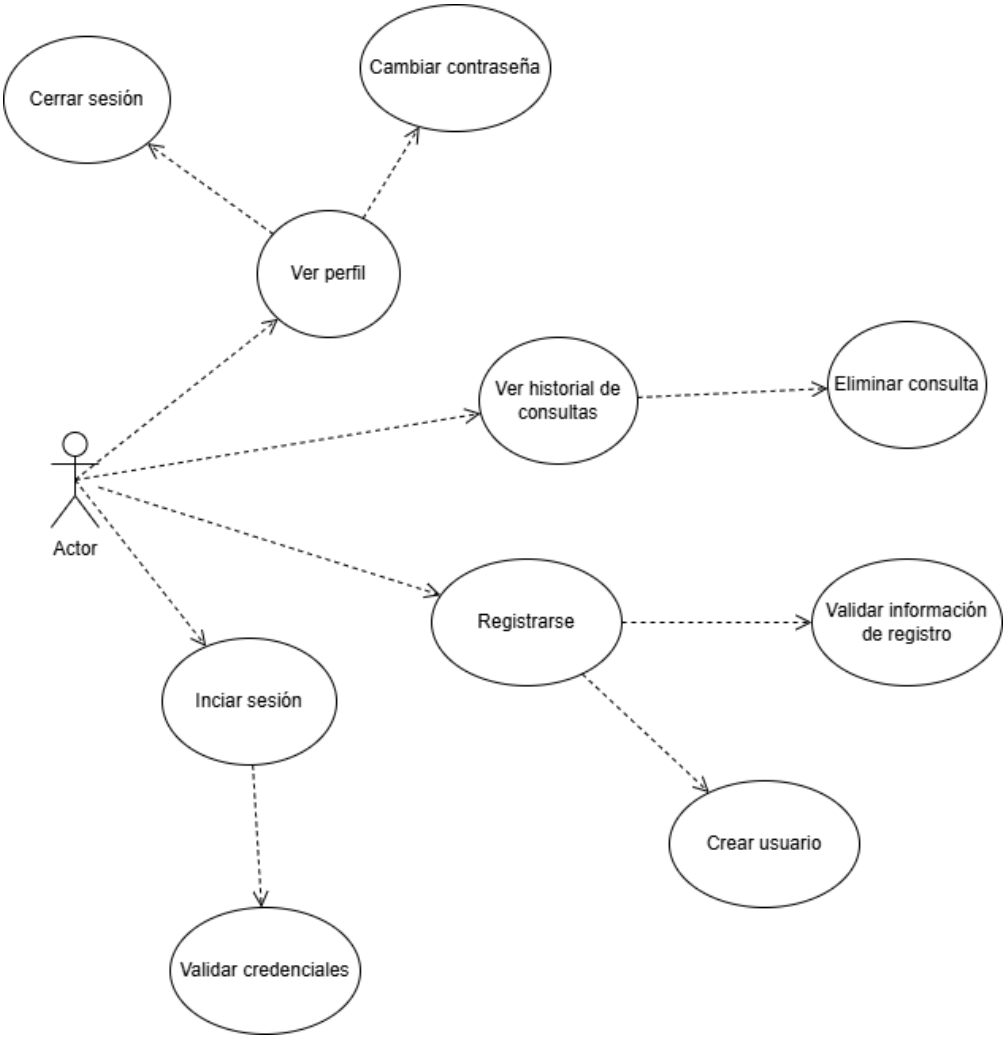

Figura 3.2: Diagrama de casos de uso de la aplicación – gestión de usuarios y consultas.

3.4. Diagramas de flujo

3.4.1. Descripción de imágenes mediante cámara del móvil

Este proceso corresponde al caso de uso de descripción del entorno mediante la cámara del dispositivo móvil. Al hacerlo, se abre la cámara del dispositivo y el usuario puede tomar una fotografía del entorno. Una vez realizada la captura, la imagen es enviada al sistema de análisis visual integrado con Gemini, que genera una descripción textual del contenido.

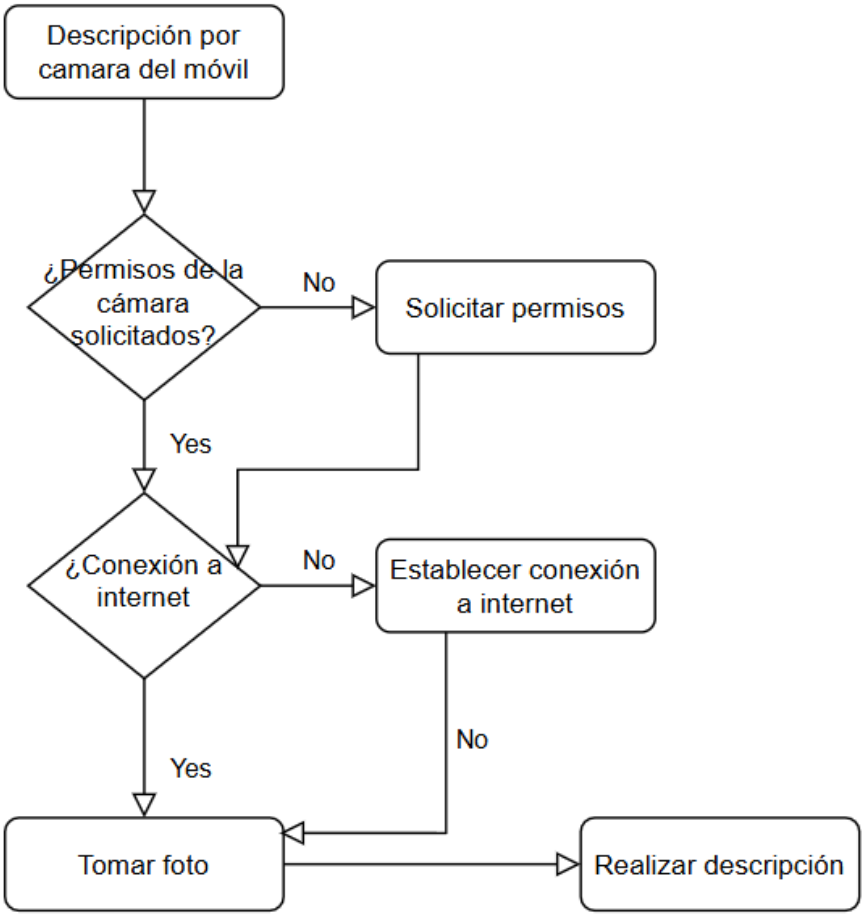

Figura 3.3: Diagrama del flujo de descripción de imágenes mediante la cámara del móvil.

Finalmente, esta descripción se reproduce mediante el sistema Text-To-Speech (TTS), permitiendo al usuario recibir la información auditivamente. Este proceso está completamente automatizado y no requiere que el usuario interactúe visualmente con la pantalla.

3.4.2. Descripción de imágenes desde la galería

Este proceso representa el flujo detallado correspondiente al caso de uso en el que el usuario selecciona una imagen previamente almacenada en el dispositivo. Accediendo a esta opción desde el menú principal donde se abre el explorador de imágenes (galería).

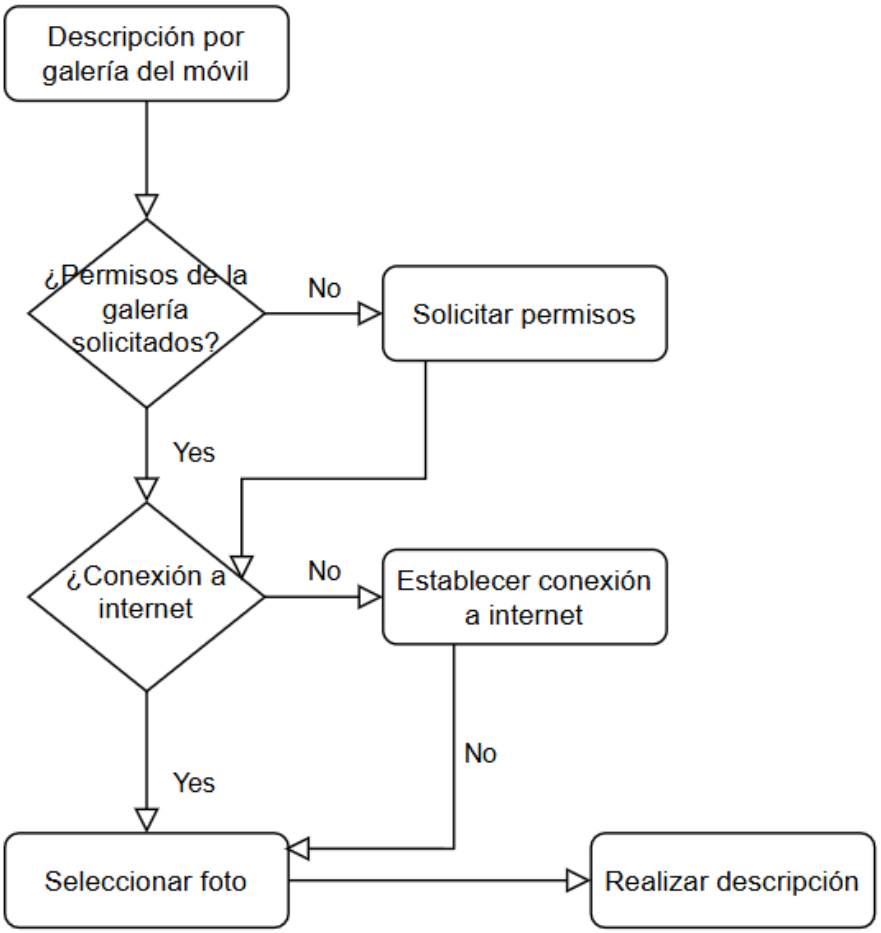

Figura 3.4: Diagrama del flujo de descripción de imágenes mediante la galería del móvil.

Una vez seleccionada una imagen, esta se procesa del mismo modo que en el caso anterior: se analiza utilizando Gemini y se genera una descripción que se reproduce por audio. Esta opción permite reutilizar imágenes previas, como fotografías guardadas, imágenes recibidas por mensajería o archivos almacenados por el usuario.

3.4.3. Descripción de imágenes mediante gafas Rokid

A continuación se detalla el flujo de proceso corresponde al caso de uso que permite la captura de imágenes mediante las gafas inteligentes Rokid Air, sin necesidad de sostener el teléfono móvil. Para que esta opción esté disponible, es necesario que las gafas estén conectadas al dispositivo Android mediante USB-C o inalámbricamente, según la configuración.

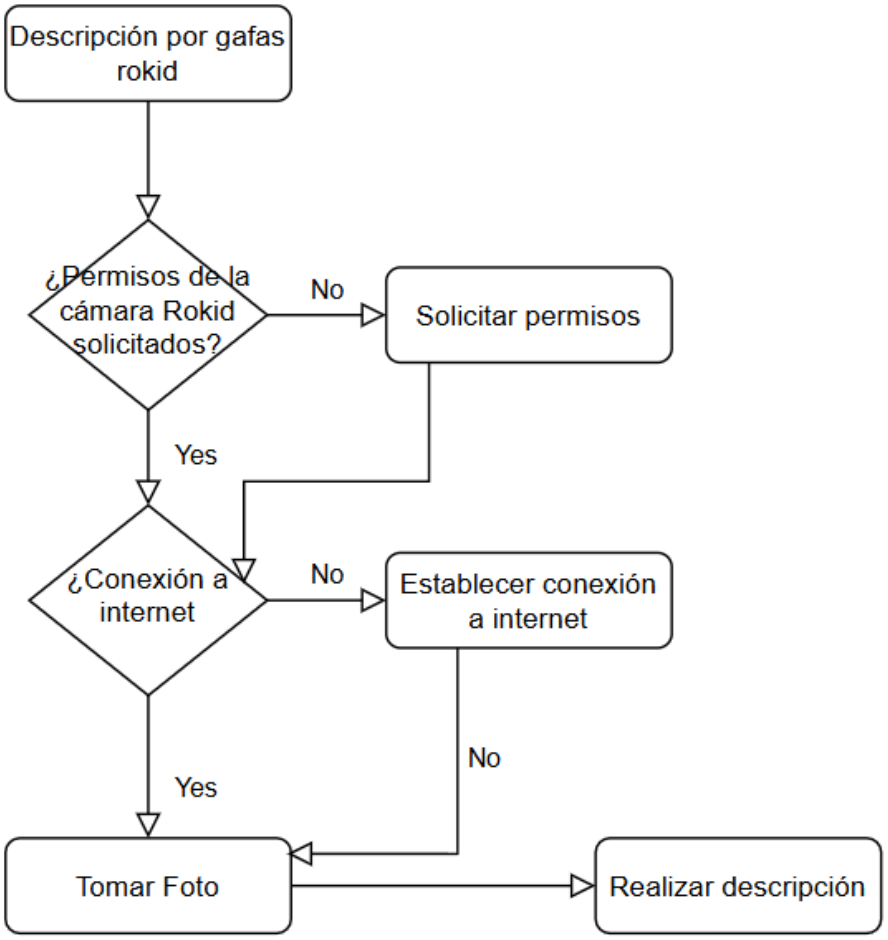

Figura 3.5: Diagrama del flujo de descripción de imágenes mediante la cámara de las gafas Rokid.

3.4.4. Inicio de sesión

Este proceso representa el flujo de autenticación de un usuario dentro del sistema. El usuario ingresa su nombre de usuario y contraseña, y el sistema valida las credenciales contra la base de datos. En caso de error, se presentan mensajes específicos, como usuario inexistente o contraseña incorrecta. Si las credenciales son correctas, se concede el acceso al sistema.

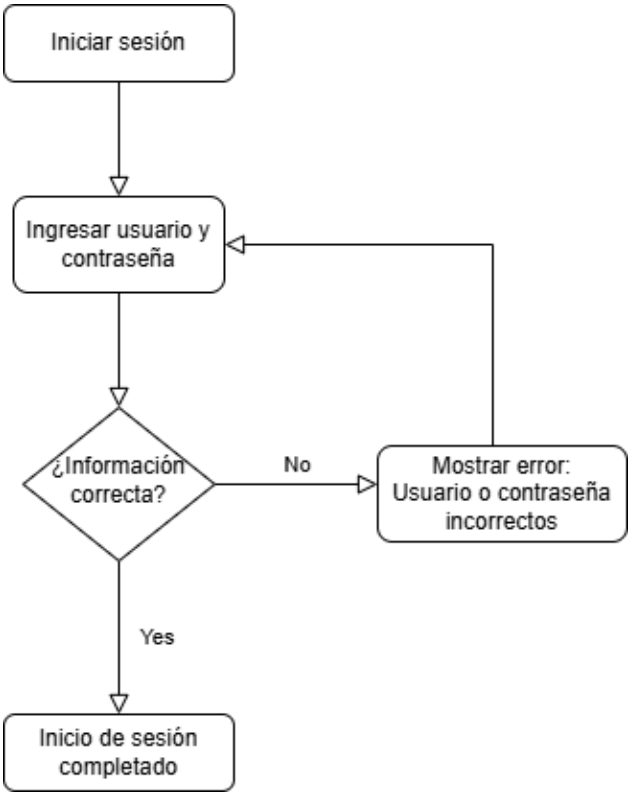

Figura 3.6: Diagrama del flujo de inicio de sesión.

3.4.5. Registro de usuario

Este diagrama describe el proceso completo de registro. El usuario introduce un nombre de usuario y una misma contraseña dos veces (para evitar posibles errores). El sistema verifica si el usuario ya existe y que no contenga caracteres inválidos, admitiéndose solo letras, números y guiones bajos. También se valida que las contraseñas coincidan y que tengan una longitud adecuada de mínimo 8 caracteres. Tras pasar todas las validaciones, se crea la cuenta y se redirige al inicio de sesión.

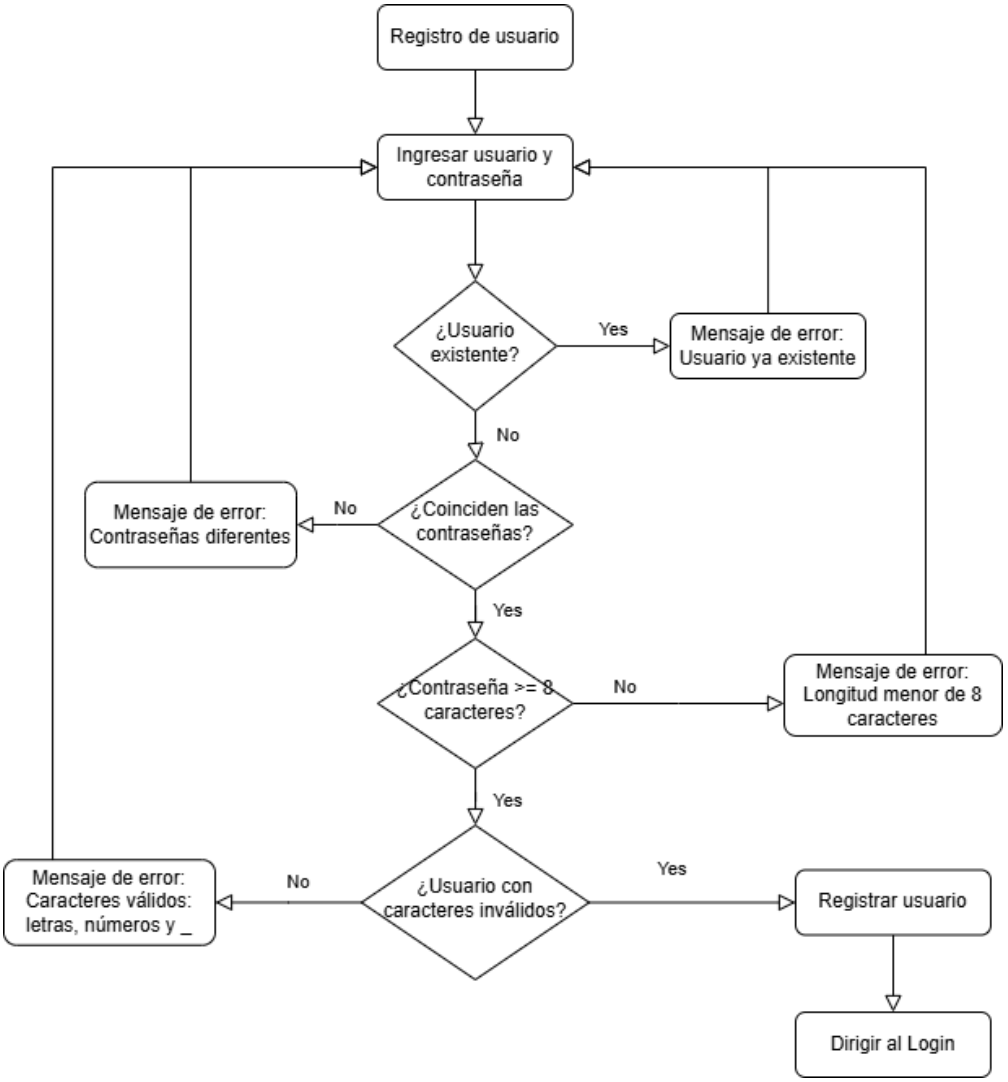

Figura 3.7: Diagrama del flujo de registro de usuario.

3.4.6. Gestión del perfil de usuario

El siguiente diagrama de flujo representa el comportamiento del sistema cuando el usuario intenta acceder a la sección de perfil. Se contempla tanto si el usuario había iniciado sesión previamente o no. Si el usuario no ha iniciado sesión, el fragmento del perfil contiene un mensaje informando que para poder acceder debe haber iniciado sesión y contiene un botón para autentificarse que redirige al login y otro para cancelar y volver a la página de incio (Home). Si decide iniciar sesión y este es exitoso, el sistema permite al usuario visualizar el perfil y acceder a sus funcionalidades.

Una vez en el perfil, el usuario puede realizar tres acciones principales: cerrar sesión, cambiar la contraseña o cancelar y volver a la pantalla anterior. En el caso del cierre de sesión, se solicita confirmación antes de redirigir al inicio. En cuanto al cambio de contraseña, el sistema solicita la contraseña actual y dos veces la nueva para validación. Se muestran mensajes específicos en caso de error, como cuando la contraseña anterior es incorrecta o las nuevas no coinciden y se valida que la nueva contraseña tenga una longitud mínima de ocho caracteres mostrando un mensaje de error en caso contrario. Si todo es correcto, se actualiza la contraseña con éxito.

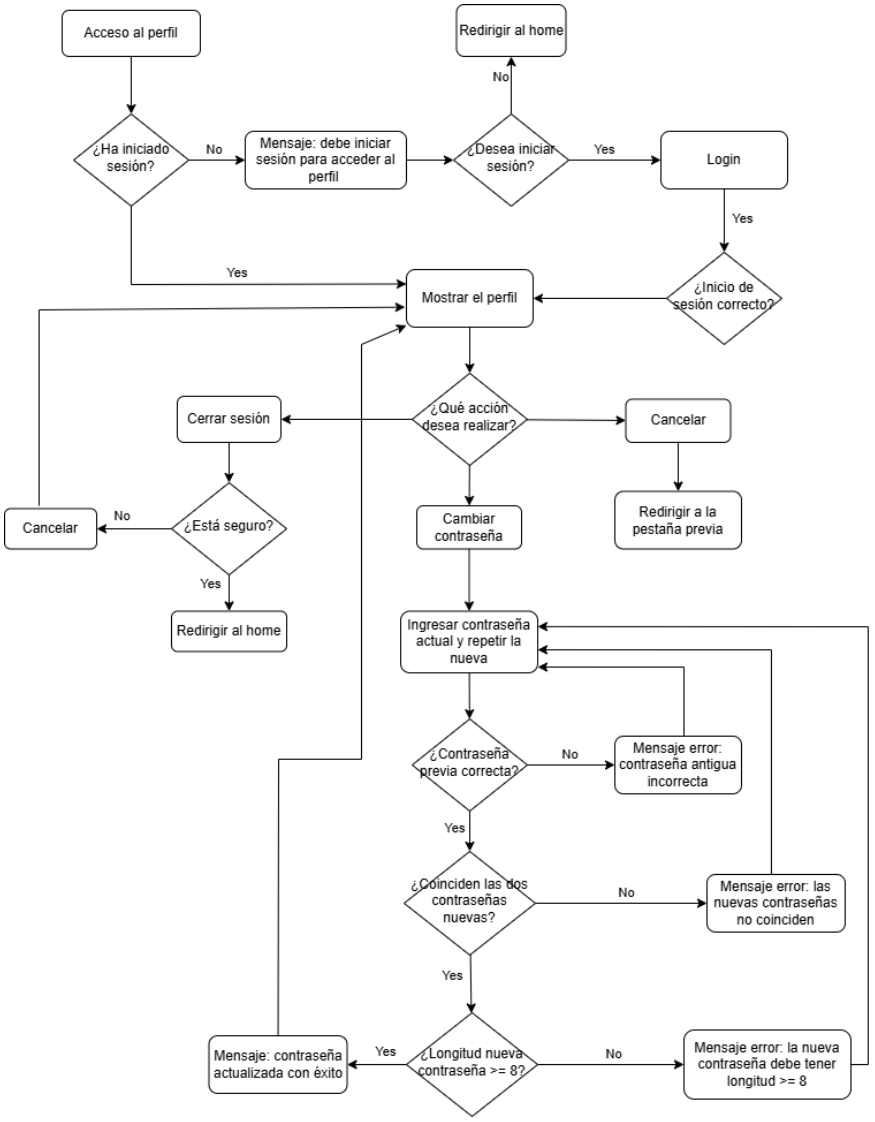

Figura 3.8: Diagrama del flujo del perfil de usuario.

Capítulo 4

Diseño

4.1. Diseño de la arquitectura del sistema

4.1.1. Visión General

La aplicación en su mayoría sigue una arquitectura basada en fragmentos, lo cual permite una navegación modular, ya que cada fragmento se ocupa de una única pantalla y una única funcionalidad, siendo el funcionamiento de las mismas independiente del resto.

Sin embargo, para gestionar la funcionalidad de las gafas inteligentes se ha optado por utilizar el patrón de arquitectura MVVM (Model-View-ViewModel), ya que, a diferencia del resto de fragmentos que son más sencillos, esta funcionalidad sí que es algo más exigente respecto a la gestión de la conexión de las gafas inteligentes y su cámara con la aplicación.

4.1.2. Componentes principales

Los principales módulos del sistema son:

- **Cámara del dispositivo**: se permite sacar una foto del entorno para recibir la descripción.
- **Galería**: se muestran las fotos de la galería del dispositivo, por si el usuario quiere recibir la descripción de alguna de ellas.
- **Gafas inteligentes**: gestiona la conexión con las gafas inteligentes y la captura de imágenes mediente la cámara que tiene integrada.
- **API Gemini**: se conecta con la API de Gemini para enviar las imágenes y recibir las descripciones de las mismas mediante inteligencia artificial.
- **Historial de consultas**: las consultas realizadas por el usuario se mostrarán aquí, por si el usuario quiere ver de nuevo alguna de las descripciones.
- **Accesibilidad**: se permite al usuario cambiar los colores de contraste entre dos opciones para que elija la opción con la que se sienta más cómodo.

4.1.3. Patrones de diseño

Se han usado dos patrones principales:

- **Gestión de fragmentos**: patrón común en aplicaciones Android que permite una navegación estructurada basada en fragmentos. Facilita la reutilización de componentes, la adaptación a diferentes tamaños de pantalla y una mejor organización del flujo de la aplicación.
- **Model-View-ViewModel**: se utiliza exclusivamente en la conexión de las gafas inteligentes con el dispositivo y la funcionalidad de las mismas en la aplicación.

4.1.4. Tecnologías principales

- **Plataforma**: Android
- **Lenguaje de programación**: se han usado tanto Kotlin como Java
- **Servicios utilizados**:
  - **Gemini**: modelo de IA utilizado para el análisis y descripción de imágenes. Se accede mediante API REST y se integran mecanismos de manejo de errores y control de latencia.
  - **SQLite**: sistema de gestión de base de datos local usado para almacenar descripciones previas e información de los usuarios.

4.1.5. Diagrama de componentes

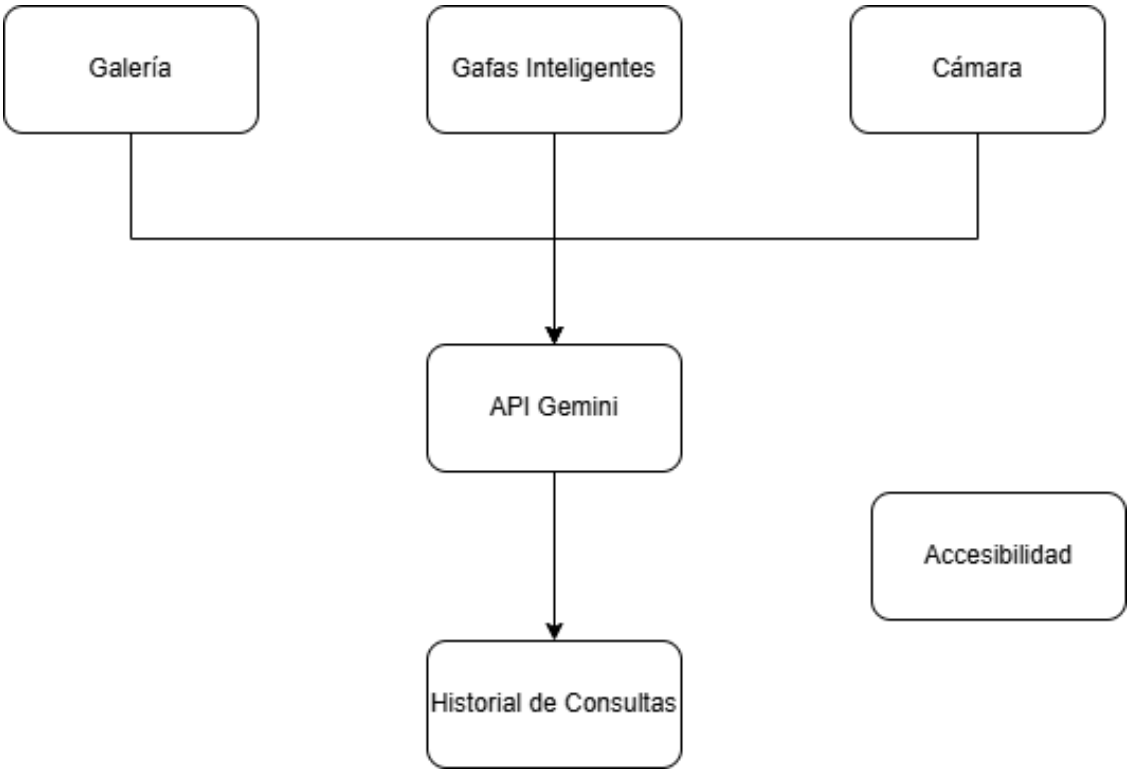

Figura 4.1: Diagrama de componentes del sistema

4.2. Diseño de la interfaz

En esta sección se explica el diseño de la interfaz gráfica de la aplicación, así como las decisiones tomadas para asegurar que sea accesible. La interfaz de la aplicación ha sido diseñada teniendo en cuenta las recomendaciones de diseño de Android y las pautas de accesibilidad de la WCAG (Web Content Accessibility Guidelines).

4.2.1. Accesibilidad

Compatibilidad con lector de pantalla

Una parte muy importante de la accesibilidad en aplicaciones móviles, es la compatibilidad con lectores de pantalla. En este caso, al ser una aplicación para dispositivos android, se ha asegurado la compatibilidad con TalkBack, el lector de pantalla nativo de Android. Para ello, se ha etiquetado correctamente todo el contenido de la aplicación, de manera que cada elemento tiene una descripción apropiada. Además, se ha asegurado que el flujo de navegación sea intuitivo, facilitando la interacción con la aplicación.

Tamaño de los elementos y fuentes

Tal como se recomienda en la guía de diseño de Android [24], todos los elementos tienen un tamaño mínimo de 48dp y un área de accionamiento de 56x56dp para asegurar que son fácilmente seleccionables.

Para el tamaño de las fuentes también se ha seguido la guía de diseño de Android, que recomienda usar píxeles escalables (sp) para asegurar que el texto se adapta a los diferentes ajustes de tamaño que pueda tener el usuario. En este caso, el tamaño de texto más pequeño que se ha utilizado ha sido 18sp para pequeños textos informativos, llegando a 48sp para los botones más importantes. El tamaño de texto recomendado por la guía de diseño de android es de 14sp, pero se ha optado por un tamaño mayor dado que la aplicación está destinada a personas con baja visión, que pueden tener dificultades para leer textos pequeños.

Elección de colores y contraste

La aplicación ha sido diseñada teniendo en cuenta principios de accesibilidad visual, especialmente en lo referente al contraste entre colores. Esta decisión es fundamental para usuarios con baja visión, degeneración macular, cataratas, retinopatía diabética o sensibilidad a la luz. Un contraste insuficiente puede dificultar la lectura, la navegación y el uso correcto de los elementos interactivos.

De acuerdo con las directrices de accesibilidad para el contenido web (WCAG 2.2), se recomienda que el texto y los elementos gráficos tengan un contraste mínimo de 4.5:1 respecto al fondo para garantizar una legibilidad adecuada en la mayoría de los casos [25]. Estas pautas han sido adoptadas también por plataformas móviles como Android e iOS como estándares de accesibilidad.

En la práctica, la aplicación ofrece tres temas visuales:

- **Tema claro**: Este tema se activa autmáticamente cuando el usuario está usando el modo

claro de su dispositivo. Tiene el fondo blanco, con texto e iconografía en color negro, ideal para entornos bien iluminados.

- **Tema oscuro**: Este tema se activa autmáticamente cuando el usuario está usando el modo oscuro de su dispositivo. Tiene el fondo negro, con texto blanco e iconos en azul oscuro, ideal para entornos con baja luminosidad, donde el blanco puede causar fatiga visual.
- **Tema de alto contraste**: Este tema se puede activar desde el menu de ajustes, y está pensado para personas con visibilidad reducida. Este tema tiene el fondo negro, con texto amarillo e iconos en negro con fondo cyan. Además, en las distintas pestañas de la aplicación se ha evitado usar rellenos coloridos, optando por dejar el fondo negro y delimitar la pestaña con un borde cyan.

4.2.2. Diseño de las pantallas

Pantalla de inicio

La pantalla de inicio es la pantalla principal de la aplicación, por tanto, se ha diseñado para ser sencilla, con el mínimo número de elementos posibles y así evitar una sobrecarga visual.

Como se puede ver en la Figura 4.2 el caso de haber iniciado sesión, en esta pantalla se muestra un historial con las consultas anteriores del usuario. Para cada una de ellas, se muestra la imagen que se consultó y, a su lado la descripción que se generó a través de Gemini. Para la funcionalidad de eliminar una consulta concreta se ha optado por una acción de deslizamiento hacia cualquiera de los lados que muestra en rojo, un color con un alto contraste tanto en fondo blanco como en fondo negro y un icono de papelera sobre este.

Si no se ha iniciado sesión, en la pantalla de inicio se le propone al usuario hacerlo como se muestra en la Figura 4.3.

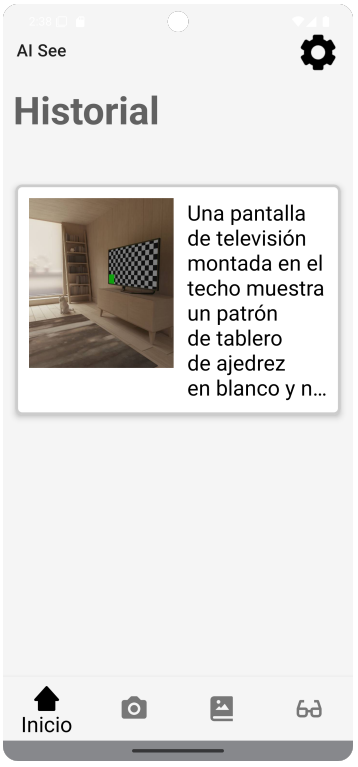

Figura 4.2: Pantalla de inicio si se ha iniciado sesión

Figura 4.3: Pantalla de inicio si no se ha iniciado sesión

Ajustes

La sección de ajustes es un menu sencillo en forma de lista en el que se muestran las opciones de cambio de idioma, administración del perfil y un botón para el cambio al modo alto contraste. Se puede ver en la Figura 4.4.

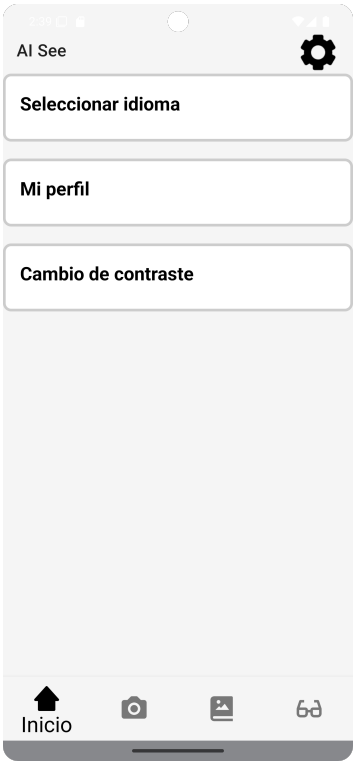

Figura 4.4: Pantalla de ajustes

Cámara

Como se observa en la Figura 4.5a pantalla de la cámara es estándar, muestra una previsualización de la foto que se va a tomar y tiene un botón circular para tomar la foto. Este diseño es muy común en cualquier aplicación que pueda tomar fotos y por tanto es un diseño familiar para cualquier usuario.

Galería

Esta pantalla muestra las fotos que hay en la galería del usuario, al igual que la cámara, el diseño de esta pantalla esta pensado para que resulte familiar al usuario. Figura 4.6.

Pantalla de monitorización de las gafas

Esta pantalla sirve de complemento al uso de las gafas y por lo tanto debe de mostrar la información de manera clara para poder procesarla en un vistazo. Con ese objetivo en mente, La pantalla de uso de las gafas sirve para monitorear el estado de sincronización con las gafas y mostrar la descripción de la foto que se tome con ellas.

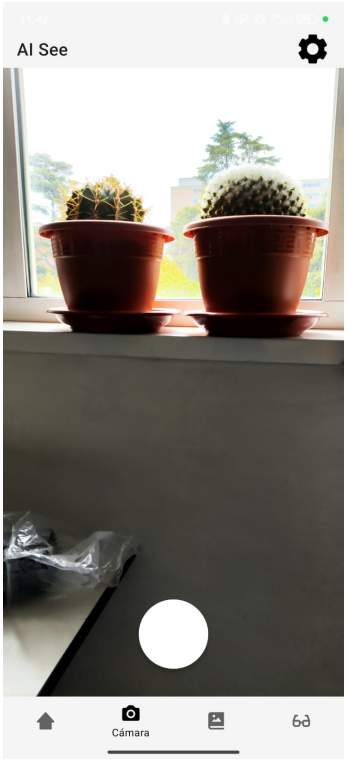

Figura 4.5: Pantalla de la cámara

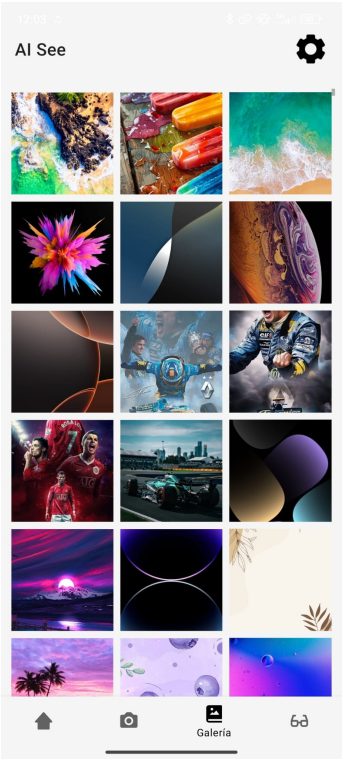

Figura 4.6: Pantalla de la galería

Barra de navegación

En la barra de navegación se encuentran los iconos de las distintas pantallas de la aplicación como se puede ver en la Figura 4.7, permitiendo navegar entre ellas al pulsarlos. Para identeficar en que pantalla se encuentra el usuario, el icono de la pantalla activa se muestra en el color correspondiente al tema activo, mientras que el resto de los iconos se muestran en gris. Además se muestra un texto debajo del icono activo que indica el nombre de la pantalla.

Este diseño puede ser poco accesible ya que no hay mucho contraste entre los iconos y el fondo, por lo que para el tema de alto contraste se ha optado por un diseño diferente, en el que los iconos inactivos se muestran en gris con fondo negro y el icono activo se muestra en negro pero con un fondo cyan, con un contraste mucho más alto. El diseño de la barra de navegación de alto contraste se muestra en la Figura 4.8.

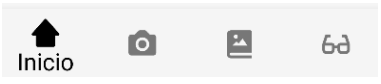

Figura 4.7: Barra de navegación en el modo claro

Figura 4.8: Barra de navegación en el modo alto contraste

4.3. Diseño de la base de datos

La base de datos diseñada para esta aplicación móvil tiene como objetivo gestionar la información de los usuarios, su historial de consultas y las descripciones de imágenes generadas por la aplicación.

4.3.1. Modelo entidad-relación

La base de datos está compuesta por tres entidades principales: `Usuario`, `Historial` y `Consulta`. A continuación, se describen sus relaciones:

- `Usuario`: contiene el nombre de usuario y el hash de la contraseña.
- `Historial`: relacionado uno a uno con el usuario.
- `Consulta`: está vinculada a un único historial. Contiene las descripciones y los datos binarios de la imagen cifrados.

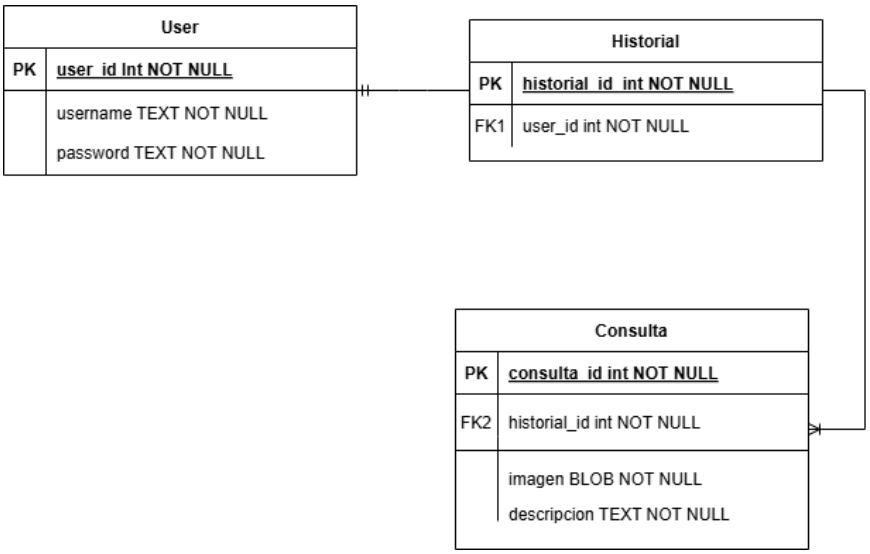

Figura 4.9: Diagrama Entidad-Relación del modelo de base de datos

4.3.2. Integridad referencial y relaciones

El diseño de la base de datos implementa claves foráneas para garantizar la integridad de las relaciones entre entidades. La tabla `historial` contiene una clave foránea que referencia a `usuario`, asegurando que cada historial esté vinculado a un usuario existente.

De forma similar, la tabla `consulta` incluye una clave foránea que apunta a la tabla `historial`, lo cual garantiza que cada descripción de imagen esté siempre asociada a un historial válido. Estas restricciones permiten mantener la coherencia del modelo, evitando la creación de registros huérfanos y facilitando potencialmente la eliminación en cascada de datos relacionados.

4.3.3. Normalización de la base de datos

Durante el diseño del modelo relacional se ha aplicado el proceso de normalización con el fin de minimizar redundancias, prevenir anomalías en las operaciones (inserción, actualización, eliminación) y garantizar la integridad de los datos.

A continuación, se detalla el cumplimiento de las formas normales principales:

- **Primera Forma Normal (1FN):** todos los atributos son atómicos, sin valores multivaluados ni estructuras repetitivas. Por ejemplo, los campos `username`, `descripcion` e `imagen` almacenan valores únicos y no divididos.
- **Segunda Forma Normal (2FN):** al no existir claves primarias compuestas, se cumple automáticamente. Además, todos los atributos no clave dependen completamente de la clave primaria de su tabla.
- **Tercera Forma Normal (3FN):** no hay dependencias transitivas entre atributos no clave. Todos los campos dependen únicamente de la clave primaria de su entidad correspondiente. Por ejemplo, en la entidad `Consulta`, los campos `descripcion` e `imagen` dependen exclusivamente de `consulta_id`.

En conclusión, el modelo propuesto cumple con la Tercera Forma Normal, lo que garantiza una estructura sólida, sin redundancias innecesarias, y preparada para futuras evoluciones

4.3.4. Consideraciones de escalabilidad

Aunque el sistema actual asocia a cada usuario un único historial, se ha optado por mantener dicha entidad de forma independiente para permitir su extensión en versiones futuras. Este diseño modular facilita implementar nuevas funcionalidades como múltiples historiales por usuario, clasificación de consultas por fecha o tipo, o incluso generación de estadísticas personalizadas.

Del mismo modo, el modelo está preparado para admitir nuevas entidades o relaciones sin necesidad de una reestructuración completa, permitiendo añadir por ejemplo:

- Etiquetas o categorías personalizadas para agrupar consultas.
- Campos adicionales en los historiales como fecha de creación, nombre del historial, tipo de entorno, etc.
- Mecanismos de compartición de historiales entre usuarios (en caso de supervisión médica o asistencial).

Este enfoque permite el crecimiento de la aplicación sin comprometer la solidez del modelo de datos.

4.3.5. Seguridad en el modelo de datos

Durante el diseño del modelo de datos se ha prestado especial atención a los aspectos relacionados con la seguridad y la protección de la información sensible de los usuarios. Estos principios se

han incorporado desde las fases iniciales del desarrollo, en línea con el enfoque de *privacidad desde el diseño* establecido en el artículo 25 del Reglamento General de Protección de Datos (RGPD).

Concretamente, se han adoptado las siguientes decisiones:

- **Minimización de datos:** el modelo evita almacenar información innecesaria o altamente identificativa, limitándose a campos esenciales como el nombre de usuario (único) y una contraseña cifrada.
- **Cifrado de información sensible:** tanto las contraseñas como las descripciones textuales e imágenes almacenadas en las consultas son cifradas antes de ser persistidas. Esto reduce el impacto de posibles accesos no autorizados a la base de datos.
- **Separación lógica de entidades:** la organización modular en torno a entidades como `Historial` o `Consulta` permite un mayor control sobre el acceso a los datos.
- **Integridad referencial:** el uso de claves foráneas garantiza que los datos relacionados mantengan una coherencia lógica, evitando registros huérfanos o inconsistentes que podrían dar lugar a errores o fugas de información.
- **Restricción de formato en campos críticos:** desde el diseño se ha planteado que campos como el nombre de usuario solo permitan caracteres seguros (letras, números y guiones bajos) y que las contraseñas cumplan con una longitud mínima, con el objetivo de prevenir entradas maliciosas o datos inválidos antes de ser almacenados.

Estas decisiones no solo fortalecen la seguridad técnica del sistema, sino que alinean el desarrollo con los principios fundamentales del RGPD, como el de minimización, limitación del tratamiento, e integridad y confidencialidad de los datos [26, 27].

4.4. Diseño del sistema de descripción de imágenes

La funcionalidad básica de la aplicación es mostrar una descripción de la imagen proporcionada. Para ello, se carga un modelo Gemini en el proyecto, para poder hacer las consultas para después mostrarlas y leerlas mediante la librería de Android TextToSpeech.

4.4.1. Justificación de la elección del modelo de Gemini

Durante la fase de diseño se evaluaron distintas alternativas para implementar la generación automática de descripciones de imágenes.

Se optó por utilizar el modelo Gemini de Google, debido a su compatibilidad con entornos Android, soporte más directo para el tratamiento de imágenes y disponibilidad de herramientas adaptadas al ecosistema de desarrollo móvil. Esta elección permitió simplificar la arquitectura del sistema.

A continuación, se muestra una comparativa entre las opciones analizadas:

Criterio	OpenAI / ChatGPT	Gemini (Google)
Lenguajes de integración	API REST genérica, no especializada en móviles	SDKs y APIs específicas para Android
Complejidad de implementación	Mayor (gestión de seguridad, comunicación servidor-cliente)	Menor (orientado a flujos móviles nativos)
Coste estimado	Variable según tráfico de datos y procesamiento	Integrado en servicios Google Cloud, modelo de precios estable
Soporte oficial en Android	Limitado	Extenso, documentación oficial disponible

Cuadro 4.1: Comparativa entre OpenAI/ChatGPT y Gemini para integración en Android

4.4.2. Arquitectura general de la funcionalidad

Esta funcionalidad sigue un flujo de funcionamiento de 3 etapas principales:

1. **Captura de la imagen**: la imagen es capturada por diferentes medios:
   - Por la cámara del dispositivo movil.
   - Por la cámara de las gafas inteligentes Rokid Air Pro.
   - Seleccionando una imagen de la galería.
2. **Procesamiento de la imagen**: teniendo la imagen, esta se envía al modelo de Gemini, capaz de analizar el contenido visual y generar a partir de ella y del prompt una descripción de la imagen. El modelo es accedido mediante una API, enviando la imagen y recibiendo la descripción de la misma.
3. **Conversión de la descripción a voz**: mediante la librería de Android TextToSpeech, se puede reproducir, ya sea desde los altavoces del dispositivo movil o de los de las gafas Rokid, la descripción proporcionada por el modelo.

4.4.3. Consideraciones a tener en cuenta

Al usar Gemini para el procesamiento de imágenes, surgen diferentes limitaciones en el sistema:

- **Latencia del sistema**: el envío de las imágenes al modelo para su procesamiento introduce un tiempo de espera que hay que asumir.
- **Dependencia a la conectividad**: el envío de las imágenes también requiere de estar conectado a la red, por lo que, en situaciones donde no se tenga acceso, el sistema no podrá dar una respuesta a la imagen enviada. Aunque al hacer la foto, esta se guarda en la galería por lo que puede ser accesible en otro momento.

4.5. Diseño de la integración con Smart Glasses

La integración de gafas inteligentes en nuestra aplicación representa un elemento diferenciador clave dentro del sistema, permitiendo una experiencia más fluida y natural para personas con visibilidad reducida. En concreto, se ha optado por utilizar las gafas Rokid Air, que ofrecen conectividad con dispositivos Android y una cámara integrada que permite la captura de imágenes desde la perspectiva del usuario.

4.5.1. Justificación de la elección de Rokid Air

Las Rokid Air han sido seleccionadas por su compatibilidad con smartphones Android, su ligereza, y especialmente por su capacidad de transmitir imagen en tiempo real a través de su cámara. Esto permite capturar de manera más intuitiva el entorno del usuario sin que este deba manipular el móvil, aumentando así la accesibilidad.

4.5.2. Modo de integración

Para integrar las gafas con la aplicación Android se ha seguido un enfoque basado en el reconocimiento del dispositivo como una fuente de entrada multimedia. Al conectarse mediante USB-C o mediante Miracast, la cámara de las Rokid Air es detectada como una cámara externa.

Desde el punto de vista del desarrollo, se han tenido en cuenta las siguientes consideraciones:

- **Detección de cámara externa:** se ha implementado una lógica que permite identificar si hay una cámara secundaria disponible en el sistema y ofrecer al usuario la opción de utilizarla desde el menú de captura.
- **Compatibilidad y permisos:** para asegurar el correcto funcionamiento, se ha adaptado la gestión de permisos (`Camera`, `USB`, `READ_EXTERNAL_STORAGE`) y se han realizado pruebas en dispositivos con distintas versiones de Android para garantizar compatibilidad.
- **Interfaz adaptativa:** al usar las Rokid Air, el usuario no necesita mirar el móvil, por lo que el botón de captura de imágenes integrado en la cámara permite la captura de fotos sin tener que mirar a la pantalla del móvil nada más que para acceder a la aplicación. Además, el altavoz de las gafas facilita la descripción de la imagen mediante audio.

4.6. Consideraciones éticas y legales

El uso de tecnologías de inteligencia artificial para el análisis de imágenes implica ciertos riesgos y responsabilidades desde el punto de vista legal y ético. En este proyecto, se utiliza el modelo multimodal *Gemini* desarrollado por Google para generar descripciones automáticas de imágenes proporcionadas por los usuarios.

4.6.1. Tratamiento de datos personales

Las imágenes analizadas por el sistema pueden contener información personal identificable, tales como rostros, objetos del entorno doméstico, documentos visibles, entre otros. Según el Reglamento General de Protección de Datos (RGPD), este tipo de contenido se consideran datos personales, y su tratamiento debe realizarse bajo estrictas garantías de seguridad, confidencialidad y transparencia.

4.6.2. Uso de Gemini y envío de datos a terceros

Gemini es una tecnología propiedad de Google que funciona en la nube. Esto implica que las imágenes deben ser transmitidas a servidores externos para ser procesadas. De acuerdo con la política de privacidad de los servicios de Google Cloud AI, los datos enviados pueden ser utilizados para:

- Procesar y responder a la solicitud del usuario.
- Mejorar los modelos mediante técnicas de aprendizaje federado o supervisado, salvo que se desactive explícitamente dicha opción.
- Almacenarse temporalmente en los servidores para depuración y mejora del servicio.

Por tanto, el uso de Gemini en un entorno real requeriría establecer un contrato de procesamiento de datos con Google, asegurar el consentimiento informado de los usuarios finales, e implementar medidas como la anonimización de imágenes o el uso de instancias locales cuando sea posible [28, 29].

Capítulo 5

Implementación

5.1. Tecnologías utilizadas

Para el desarrollo de la aplicación hemos usado diferentes tecnologías, lenguajes de programación y recursos externos.

5.1.1. Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones Android. Basado en IntelliJ IDEA, ofrece herramientas avanzadas que mejoran la productividad, como emuladores para probar aplicaciones en diversos dispositivos y versiones de Android, depuración en tiempo real y un editor visual de interfaces que permite diseñar layouts XML de manera intuitiva [30]. Además, facilita la instalación directa de aplicaciones en dispositivos físicos, lo cual es esencial para probar funcionalidades específicas, como la conexión con las gafas Rokid Air, que no pueden emularse virtualmente.

5.1.2. Lenguajes utilizados

En el desarrollo de la aplicación hemos usado 2 lenguajes:

- **Java**: es un lenguaje de programación orientado a objetos muy utilizado en el desarrollo software y el utilizado por Android en sus inicios. Es un lenguaje que conocemos bien por lo que decidimos empezar el desarrollo de la aplicación con Java, de manera que la primera versión que hicimos está implementada exclusivamente en este lenguaje, esto es la versión de la aplicación sin la conexión con las gafas.
- **Kotlin**: es un lenguaje más moderno que Java, y se está promoviendo como el favorito para Android desde Google. Es un lenguaje que se parece a Java, pero ofrece una sintaxis mas concisa, segura y expresiva, lo que mejora la legibilidad del código y la comprensión del mismo. En este proyecto se ha usado ya que, a la hora de integrar las gafas, los ejemplos que teníamos estaban exclusivamente en kotlin, por lo que, tras intentar hacer la implentación en Java y tener varios problemas, decidimos partir del proyecto de ejemplo que teníamos, por lo que empezamos a usarlo.

5.1.3. El modelo Gemini

Para la descripción de imágenes hemos usado el modelo de Gemini. Este modelo permite varios tipos de entrada, como texto e imágenes, que es justo lo que necesitamos en nuestra aplicación. Gemini es capaz de reconocer objetos en imagenes así como los detalles de estas, así como generar una descripción coherente y acorde a la imagen que queremos describir, lo cual es la principal finalidad. También es capaz de trabajar en distintos idiomas lo cual nos viene muy bien para añadir varios idomas a las opciones de la aplicación.

5.2. Organización del código y estructura del proyecto

5.2.1. Estructura general de carpetas

El código fuente del proyecto está organizado siguiendo una estructura modular basada en la funcionalidad. Esto permite separar las responsabilidades y facilita tanto el mantenimiento como la escalabilidad. La raíz del proyecto se encuentra dentro del paquete `com.rokid.rkglassdemokotlin`, que contiene los siguientes submódulos principales:

- **main**: clase principal `MainActivity` y su lógica de navegación y permisos.
- **hardware**, **camera**: módulos con estructura MVVM para interactuar con las gafas inteligentes y la cámara de estas.
- **presentacion**: contiene todos los fragmentos y actividades visuales, organizadas por funcionalidades.
- **negocio**: engloba tanto la capa de acceso a datos (DAOs, modelos y clase `AppDataBase`) como la lógica de negocio principal, conteniendo la clase `GeminiAPIResponse`, que se encarga de la conexión con la API de Gemini.
- **utils**: incluye clases auxiliares como cifrado, conversión de imágenes o gestión de estado.
- **res/**: recursos gráficos, visuales y textuales usados en la interfaz.

5.2.2. Separación de responsabilidades

La arquitectura del proyecto sigue el principio de responsabilidad única, el cual establece que cada módulo o clase debe tener una única razón para cambiar [31].

- **View (XML + Fragment/Activity)**: define y muestra la interfaz de usuario.
- **ViewModel**: actúa como intermediario entre la vista y los modelos.
- **Model**: contiene la lógica de negocio o los datos, incluyendo acceso a base de datos (DAOs).

5.2.3. Integración con recursos XML

Cada `Fragment` y `Activity` tiene asociado un layout en `res/layout`, vinculado mediante `ViewBinding`. Esto permite una conexión segura y eficiente entre el código y la interfaz visual.

Además, se usan archivos XML en `res/menu`, `res/values` y `res/drawable` para gestionar menús, cadenas de texto, estilos e iconografía, respectivamente.

5.2.4. Gestión de estado y configuración

La configuración del estado de sesión del usuario, así como sus preferencias personales (idioma, contraste, historial activo), se gestionan mediante el uso de **SharedPreferences**, un mecanismo de almacenamiento local clave-valor proporcionado por Android.

Este sistema permite conservar información de manera persistente incluso tras cerrar la aplicación, siempre y cuando el usuario no haya cerrado sesión manualmente. Es decir, si el usuario simplemente abandona la app, su estado autenticado y sus preferencias se mantendrán al volver a abrirla.

Los principales datos almacenados son:

- Usuario autenticado (nombre de usuario).
- Estado de sesión.
- Idioma seleccionado.
- Preferencia de contraste.
- ID del historial de consultas asociado al usuario.

Sin embargo, cuando el usuario pulsa explícitamente la opción de **cerrar sesión**, los valores almacenados en `SharedPreferences` se eliminan mediante el método `clear()`, asegurando que la siguiente vez que se inicie la app, se le requiera autenticarse nuevamente.

5.2.5. Relación entre capas

La arquitectura del sistema sigue el principio de arquitectura limpia, en la que las diferentes capas se comunican de forma unidireccional, evitando dependencias innecesarias y favoreciendo la modularidad.

El flujo de interacción entre las capas del proyecto puede describirse del siguiente modo:

1. La `Activity` o `Fragment`, ubicados principalmente en el paquete `presentacion`, capturan la acción del usuario (por ejemplo, pulsar el botón de captura de imagen en `CameraFragment`).
2. Esta acción es delegada al `ViewModel` correspondiente (en este caso sería `CameraViewModel`), el cual contiene la lógica de control y determina qué datos deben ser solicitados o procesados. El `ViewModel` no accede directamente a la interfaz ni a la base de datos, asegurando una separación clara.

3. Posteriormente, `ViewModel` se comunica con el `Model`, que incluye clases de dominio, acceso a datos mediante DAOs o lógica de negocio. Es el encargado del procesamiento de imágenes, gestionando la comunicación con la API de Gemini.
4. Una vez obtenidos y procesados los datos (por ejemplo, una descripción generada por Gemini o un resultado de base de datos), el `ViewModel` los expone mediante `LiveData` o callbacks a la `View`, que se actualiza automáticamente para reflejar el nuevo estado.

5.3. Implementación del sistema de descripción de imágenes

El sistema de descripción de imágenes constituye el núcleo funcional de la aplicación, permitiendo transformar una imagen tomada por el usuario en una descripción textual comprensible, que posteriormente es reproducida por audio. Para ello, se integró un modelo multimodal basado en la inteligencia artificial de Google: Gemini, perteneciente a la familia de modelos generativos con capacidades de comprensión de imágenes y texto.

El proceso de implementación se dividió en varias fases, que incluyeron una primera prueba con Python y una posterior migración a un entorno nativo Android, utilizando Java y Kotlin.

5.3.1. Enfoque inicial con Python y ChatGPT

En la etapa inicial del proyecto, se evaluó la posibilidad de utilizar GPT-4 con capacidades visuales a través de la API de OpenAI. Se desarrolló un script en Python que permitía codificar imágenes y enviarlas al modelo para su análisis y generación de descripciones.

Las pruebas iniciales mostraron que el modelo era capaz de identificar objetos, interpretar escenas y generar textos descriptivos coherentes. Sin embargo, esta solución presentaba limitaciones técnicas para su integración directa con Android:

- La ejecución de scripts Python en Android requería herramientas externas como Chaquopy, que ofrecían compatibilidad limitada.
- Las bibliotecas necesarias para acceder a la API de OpenAI no estaban completamente soportadas en entornos móviles.
- El enfoque obligaba a mantener una arquitectura más compleja, con posibilidad de latencia o errores en tiempo real.

También se intentó implementar la solución utilizando el modelo GPT-4 desde Java, pero debido a la falta de compatibilidad en ese momento con el nuevo modelo de GPT-4, no pudo ser posible la implementación.

Por estos motivos, se optó por migrar a una solución más integrada y nativa, eligiendo el modelo Gemini de Google como alternativa.

5.3.2. Migración a Gemini en Android

La implementación definitiva del sistema de descripción de imágenes se realizó utilizando Gemini 1.5 Flash, accedido a través del SDK de Google AI. Este modelo permite procesar entradas

multimodales (texto + imagen) y generar descripciones precisas, manteniendo compatibilidad con el entorno Android y permitiendo una integración sencilla en proyectos desarrollados en Java o Kotlin.

Para llevar a cabo esta integración, se siguieron las directrices proporcionadas en la documentación oficial de la API de Gemini para Java [32], la cual resultó clave para comprender el flujo de inicialización del modelo y la gestión asíncrona de las peticiones.

Inicialización del modelo Para inicializar el modelo, se crea una instancia de `GenerativeModel` indicando el nombre del modelo (`gemini-1.5-flash`) y una API key válida. Posteriormente, se convierte en una instancia asíncrona utilizando `GenerativeModelFutures` para facilitar la ejecución no bloqueante de las solicitudes.

Preparación de la imagen y el contenido Una vez capturada o seleccionada una imagen (desde la cámara, galería o gafas Rokid), esta se convierte en un objeto `Bitmap`. Se genera un contenido multimodal con un *prompt* definido por el equipo y la imagen en formato `Bitmap`.

Generación de la descripción (asincronía y sincronización) La generación del contenido se realiza de forma asíncrona usando `ListenableFuture` y `FutureCallback`, gestionando el resultado mediante un `CountDownLatch` para esperar la respuesta del modelo antes de continuar.

Este enfoque permite integrar Gemini sin bloquear el hilo principal de la interfaz y manteniendo la compatibilidad con el ciclo de vida de Android.

Conversión desde URI (soporte desde galería) Además, se implementó un método auxiliar para convertir una `Uri` (proveniente de la galería del dispositivo) en un objeto `Bitmap`, usando el `ContentResolver` del sistema.

Resultados obtenidos

El sistema de descripción con Gemini ha ofrecido resultados estables y consistentes. El modelo es capaz de:

- Identificar múltiples objetos dentro de una imagen.
- Interpretar contextos complejos (por ejemplo, ünа calle vacía de noche.º ünа mesa con varios platos").
- Generar descripciones coherentes y adaptadas al idioma, lo que permitirá en el futuro incorporar soporte multilingüe.
- Responder con rapidez, especialmente usando Gemini Flash, optimizado para peticiones ligeras.

Este módulo se ha integrado correctamente en los tres flujos de captura de imágenes de la app (cámara móvil, galería y gafas), siendo el punto central del sistema de accesibilidad basado en IA.

5.3.3. Selección del Prompt

A la hora de interectuar con un modelo de inteligencia artificial, ya sea Gemini, ChatGPT, etc., lo más importante es el prompt. Elegir un prompt claro ayuda a obtener una mejor respuesta del modelo en la mayoría de los casos, y en el caso de esta aplicación que se basa en generar descripciones de diferentes imágenes para gente con discapacidad visual, hay que tener en cuenta distintos factores. No conviene obtener una descripción general de la imagen, ya que esta puede dar información sobre elementos que no son importantes y omitir información acerca de los más importantes. Por ello, se tiene que saber qué elementos se espera recibir en la descripción de la imagen, debido a la naturaleza de la aplicación, estos elementos deben dar una información útil al usuario.

También hay que tener en cuenta que los modelos de lenguaje como Gemini no son deterministas por defecto, por lo que con una misma entrada se obtienen salidas diferentes. Por ello es importante generar un buen prompt para intentar controlar esto en la medida de lo posible. Estos modelos de lenguaje funcionan con una serie de parámetros que sirven para generar la respuesta, por ejemplo la temperatura, que controla la aleatoriedad de la siguiente palabra según se va generando dicha respuesta. La API de Gemini con la que se ha desarrollado esta parte permite cambiar estos parámetros para ajustarlos a distintas necesidades, sin embargo, al intentar reducir la aleatoriedad de la respuesta reduciendo el valor de la temperatura, para intentar conseguir unas respuestas más consistentes entre sí, no se ha visto un resultado significativo.

Para la selección del prompt, se decidió probar varios y compararlos entre sí. Para ello se seleccionaron varias imágenes con distintos elementos simulando como escaleras, paradas de autobús, etc., simulando imágenes reales que pudieran llegarle al prompt. Y seguidamente se usaron las mismas imágenes en los distintos prompts para comparar sus descripciones y el prompt que de unos resultados más satisfactorios.

El primer prompt que se usó fue: Describe resumidamente la siguiente imagen en español, principalmente para probar cada una de las distintas funcionalidades de la aplicación. Las descripciones que se generaban a partir de él, aunque correctas eran muy generales y omitían información importante, por ejemplo, si aparecía una señal indicando dónde están los baños en un centro comercial, mencionaban la señal pero no daban la información que la señal proporcionaba. Tampoco daban la posición relativa de cada uno de los objetos de interés, por lo que no satisfacía correctamente las expectativas para la aplicación.

Para el siguiente prompt se especificó más cómo debía ser la descripción de cara a que sea de utilidad a los usuarios de la aplicación. El prompt era: Describe esta imagen para una persona con discapacidad visual. Incluye los objetos presentes y su posición espacial, como escaleras, pasos de peatones o paradas de autobús. Si hay carteles o señales, transcribe su contenido. Si la imagen muestra documentos (como una carta de restaurante), lee todo el contenido en su contexto, incluyendo los nombres de los platos, sus ingredientes y precios. Este prompt daba más detalles sobre los elementos importantes como las escaleras y leía las señales pero sin dar su información. También daba demasiada inforamción irrelevante y al leer documentos daba un resumen del documento, lo que no siempre es deseable, por ejemplo si se trata de una carta de restaurante, que añadía caracteres como asteriscos, lo que al leerse la descripción en alto se perdía mucha claridad. Otro de los problemas del prompt era que por como estaba escrito, el modelo se centraba en encontrar los elementos importantes mencionados como escaleras o pasos de peatones, lo que hacía que en caso de no haber dichos elementos en la imagen, la descripción aclaraba que no se habían encontrado, lo cual es información irrelevante para el usuario. Por lo que sobre este prompt se fueron realizando

pequeñas modificaciones para llegar a uno que generase descripciones consistentes con la información relevante, evitando que mencionase elementos que no aparecían, diese información sobre las señales que aparecían en la imagen, y al leer documentos no generase caracteres de formato como los asteriscos, evitando así el ruido en la descripción haciendola más clara.

Se llegó así al prompt final: Describe esta imagen con precisión y utilidad para una persona con discapacidad visual. Menciona únicamente los elementos relevantes que aparecen, como señales, carteles o documentos. Indica claramente el contenido textual y la dirección que señalan las flechas, sin leer los símbolos o signos gráficos como flechas, asteriscos u otros marcadores visuales. Describe la ubicación y relación espacial de objetos importantes como escaleras, entradas o pasos de peatones. Si hay menús u otros textos, léelos completos y en contexto, incluyendo platos, ingredientes y precios, pero omite símbolos decorativos o de formato. No menciones lo que no está presente ni incluyas introducciones; da solo la descripción.

Este prompt genera descripciones consistentes de las imagenes, haciendo que la descripción para una misma imagen sea lo más parecida posible dando así los datos importantes en la gran mayoría de consultas, en algunos caso se menciona información de poco interés o errónea como en alguna señal de tráfico, pero no es lo común. Este prompt también funciona perfectamente para describir imágenes que no tienen ninguno de los elementos mencionados como importantes anteriormente, ofreciendo una descripción del contexto de la foto con los elementos que se encuentran y su posición relativa.

A continuación se muestra las descripciones generadas utilizando cada uno de los tres prompts mencionados (Figura 5.2, Figura 5.3, Figura 5.4) utilizando en cada caso la misma imágen (Figura 5.1)

Figura 5.1: Imagen utilizada en los prompts

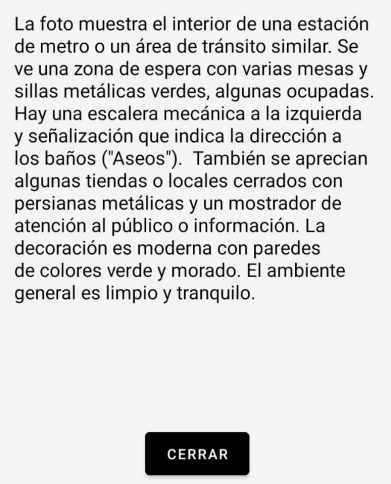

Figura 5.2: Descripción generada con prompt 1

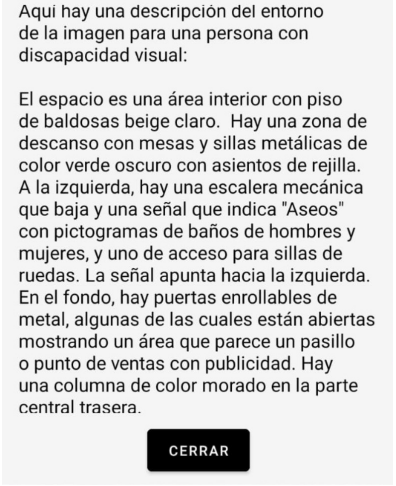

Figura 5.3: Descripción generada con prompt 2

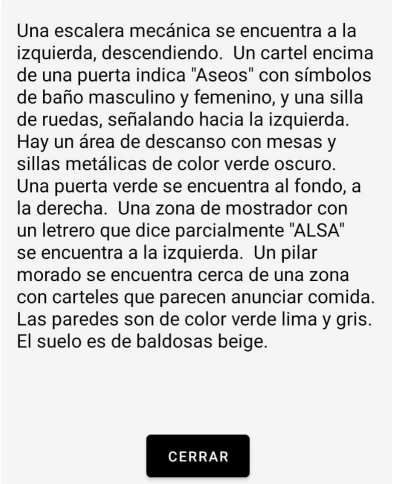

Figura 5.4: Descripción generada con prompt 3

5.4. Implementación de la base de datos y gestión de usuarios

5.4.1. Decisiones tecnológicas y problemas encontrados

Para realizar la implementación de la base de datos, inicialmente se consideró el uso de bases de datos externas como MySQL y soluciones locales más modernas como Room, una librería ORM oficial de Android para trabajar con SQLite de forma más estructurada. Sin embargo, durante el desarrollo se presentaron distintos problemas técnicos:

- Fallos en la sincronización del emulador con bases de datos externas en red local (MySQL).
- Dificultades en la configuración del entorno con Room: errores de compatibilidad entre las versiones de Room, Kotlin y el Gradle Plugin de Android.
- Problemas recurrentes al compilar debido a dependencias no resueltas de `androidx.room.*`, especialmente con conflictos entre `kapt` y `annotationProcessor`.

Dado el enfoque práctico del TFG y el tiempo limitado, se optó por implementar el sistema directamente en Java con SQLite nativo, utilizando la clase `SQLiteOpenHelper`. Esta decisión permitió un mayor control sobre la creación, apertura y consulta de la base de datos, sin depender de bibliotecas externas.

5.4.2. Modelo basado en DAOs

La arquitectura de la base de datos se ha implementado siguiendo el patrón de acceso a datos mediante DAOs (Data Access Object). Cada DAO actúa como una interfaz entre la aplicación y la base de datos, encapsulando las operaciones de acceso y manipulación de datos. Esto permite una mayor separación de responsabilidades, facilita la mantenibilidad y mejora la testabilidad del código. En este proyecto se han implementado los siguientes DAOs:

- `UserDao`: gestiona el registro, autenticación y actualización de credenciales de usuario.
- `HistorialDao`: se encarga de la creación y recuperación del historial asociado a cada usuario.
- `ConsultaDao`: administra la inserción y recuperación de descripciones de imágenes vinculadas a historiales.

De esta forma, se obtiene una arquitectura limpia y modular, separando la lógica de acceso a datos del resto de la aplicación. Esta práctica está alineada con las recomendaciones de Android Architecture Components [33].

5.4.3. Diagrama de clases del módulo de persistencia

La siguiente figura muestra la organización de las clases que forman parte de la capa de persistencia del sistema. Se incluyen las clases modelo, los objetos de acceso a datos (DAOs), y la clase `AppDataBase`, que centraliza la creación y el acceso a la base de datos.

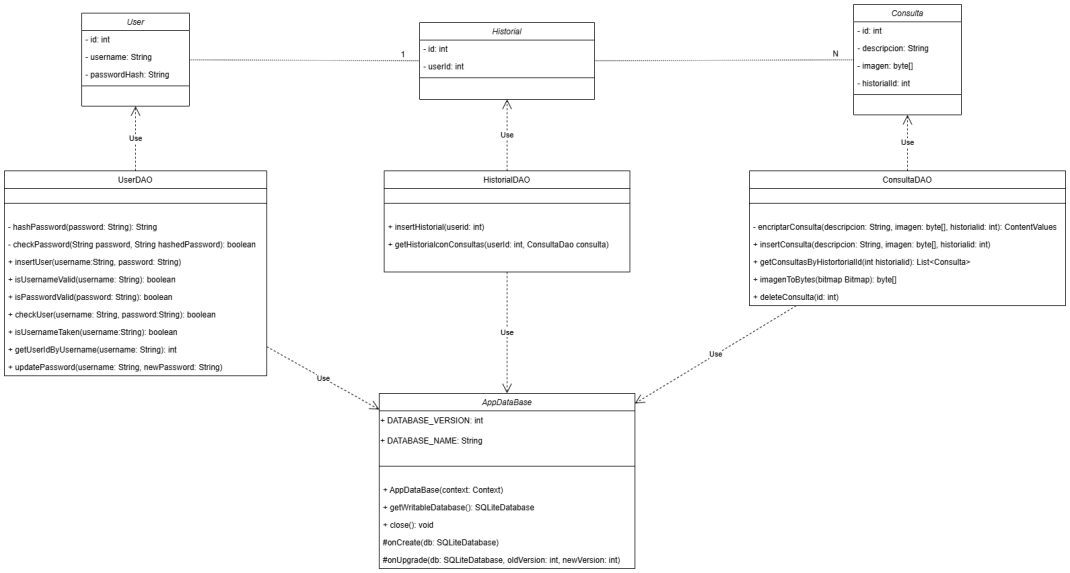

Figura 5.5: Diagrama de clases de la implementación de persistencia

5.4.4. Integración con la capa de presentación

La base de datos implementada mediante DAOs y clases modelo se comunica con la interfaz de usuario a través de distintas actividades y fragmentos que se encuentran organizados dentro del paquete `presentación/`.

Algunas de las vistas más relevantes que interactúan con los DAOs son:

- **LoginActivity** y **RegisterActivity**: utilizan `UserDao` para autenticar usuarios y registrar nuevas cuentas, incluyendo la creación automática de un historial asociado.
- **EditProfileFragment**: permite modificar la contraseña del usuario mediante una validación con `UserDao`.

- **ProfileFragment**: accede a los datos del usuario almacenados en `SharedPreferences`, actualizados tras la interacción con la base de datos.
- **ConsultaAdapter**: se encarga de mostrar en la interfaz las consultas almacenadas en la base de datos, obtenidas mediante `ConsultaDao`.
- **HomeFragment**: muestra el historial de consultas del usuario autenticado, permitiendo además la eliminación de consultas individuales mediante gestos de deslizamiento (swipe).

De este modo, se establece una clara división de responsabilidades entre la capa de presentación y la capa de persistencia, favoreciendo un diseño más modular, mantenible y escalable.

5.4.5. Medidas de seguridad en la capa de persistencia

Prevención de inyecciones SQL

Uno de los ataques más comunes en aplicaciones que interactúan con bases de datos es la **inyección SQL**, que consiste en insertar código malicioso a través de campos de entrada para manipular consultas y acceder a datos sin autorización [34]. Por ejemplo, introducir `' OR '1'='1` podría permitir eludir mecanismos de autenticación.

Para prevenir esta vulnerabilidad se han aplicado dos medidas principales:

- **Consultas preparadas:** todas las operaciones se realizan mediante *prepared statements*, que separan los datos de la lógica SQL, impidiendo que entradas maliciosas sean interpretadas como comandos.
- **Validación de entradas:** se comprueba el formato del nombre de usuario (solo caracteres alfanuméricos y guiones bajos) y se aplican restricciones de longitud y contenido a las contraseñas para evitar secuencias sospechosas.

Protección de contraseñas

Se ha empleado **BCrypt** para almacenar contraseñas, un algoritmo específicamente diseñado para este fin. BCrypt incorpora una sal aleatoria y un factor de coste configurable, lo que ralentiza los intentos de fuerza bruta y evita ataques basados en tablas precalculadas (rainbow tables). Durante la autenticación, la contraseña introducida se compara con el hash almacenado mediante `BCrypt.checkpw()`.

Cifrado de imágenes y descripciones

Los datos sensibles generados por la aplicación, como imágenes o descripciones que podrían contener información personal, se cifran utilizando el algoritmo **AES** en modo `CBC` con relleno `PKCS5Padding`. Este modo utiliza un `IV` aleatorio, diferente para cada operación, que evita que dos contenidos similares generen el mismo texto cifrado. El IV se almacena junto al contenido cifrado, lo que permite su posterior descifrado con seguridad.

Selección de algoritmos criptográficos

En la siguiente tabla se resumen los algoritmos evaluados y el motivo de su elección o descarte:

Algoritmo	**Tipo**	**Comentario**
MD5 / SHA-1	Hash	Obsoletos, inseguros frente a colisiones y ataques precalculados.
SHA-256	Hash	Seguro, pero no ideal para contraseñas por su velocidad de cómputo.
BCrypt	Hash lento	Ideal para contraseñas; incluye sal y es resistente a ataques.
AES-ECB	Cifrado simétrico	Inseguro: revela patrones en datos cifrados, especialmente imágenes.
AES-CBC	Cifrado simétrico	Modo robusto con IV aleatorio; adecuado para contenidos binarios.

Cuadro 5.1: Resumen de algoritmos criptográficos evaluados

Estas medidas están alineadas con las recomendaciones de seguridad de la OWASP Foundation [34, 35] y evitan el uso de algoritmos considerados inseguros por organismos como el NIST [36].

5.5. Implementación de la interfaz

En cuanto a la implementación de las decisiones de diseño respectivas a la interfaz, se ha seguido un enfoque modular, dividiendo la interfaz en componentes reutilizables y escalables.

A continuación se detallan los distintos módulos que se han utilizado en la implementación de la interfaz:

- **Layout**: En este módulo se encuentran todos los archivos xml que determinan el aspecto de cada una de las pantallas. En estos archivos se configuran los elementos como los botones o fotos, que tiene cada pantalla así como su organización dentro de la misma.
- **Drawable**: En este módulo se encuentran todos los elementos que se utilizan en las pantallas de la aplicación como botones, imágenes o cuadros de texto. Cabe destacar en particular el elemento que configura el color de fondo de los botones de la barra de navegación, que comprueba el tema que está utilizando el usuario y permite que en el tema de alto contraste cambie su color.
- **Menu**: En este módulo se encuentran los archivos que configuran los distintos menus de la aplicación, de momento sólo existe uno, la barra de navegación, pero está organizado de esta manera como medida de mantenibilidad.
- **Values**: En este módulo es donde se encuentran los archivos que establecen los valores de colores, dimensiones, strings, estilos y temas que se usan en el resto de archivos de la interfaz. Pueden existir varios archivos para cada una de estas áreas, para los colores y temas hay un archivo para el modo claro y otro para el modo oscuro, y para los strings hay uno por cada idioma que soporta la aplicación. En general este módulo es donde se establece el aspecto general de la aplicación.

5.6. Integración con Smart Glasses

La implementación de la integración con las gafas inteligentes Rokid Air se llevó a cabo utilizando el SDK oficial proporcionado por Rokid, disponible a través de su sitio web para desarrolladores. Este SDK permite un acceso más directo a los recursos del dispositivo, como la cámara, la gestión de entrada/salida, y la transmisión de datos entre las gafas y el terminal Android.

5.6.1. Uso del SDK de Rokid

El SDK de Rokid [37] nos ha permitido acceder a funcionalidades específicas del dispositivo, facilitando la comunicación entre la aplicación Android y las gafas. Entre las características que se han aprovechado del SDK, se encuentran:

- Acceso a la cámara integrada de las gafas como fuente de entrada para la captura de imágenes.
- Gestión de eventos de conexión y desconexión del dispositivo mediante USB-C.
- Interfaz de usuario básica proyectada en las gafas, útil para pruebas de depuración visual durante el desarrollo.
- Compatibilidad con gestos o comandos simples, aunque en nuestra implementación priorizamos la interacción por voz desde el móvil.

La documentación del SDK fue clave para entender cómo inicializar correctamente los servicios de la cámara y cómo tratar la señal de vídeo como un `SurfaceTexture` en Android para poder visualizar, capturar y procesar las imágenes.

5.6.2. Desarrollo con Kotlin

Dado que el SDK está optimizado para integrarse con aplicaciones modernas en Android, se optó por utilizar Kotlin para esta parte de la implementación. Kotlin permitió una mejor interoperabilidad con las APIs del SDK y facilitó el manejo de flujos asincrónicos y callbacks que requería el control de la cámara.

Se implementó un módulo específico en Kotlin encargado de:

- Detectar si las gafas están conectadas.
- Seleccionar la cámara correspondiente.
- Capturar una imagen bajo demanda.
- Enviar la imagen al módulo de descripción visual.

5.6.3. Flujo de captura e integración con Gemini

Una vez capturada la imagen desde las gafas, esta se transfiere directamente al módulo que se comunica con la API de Gemini. Esta etapa está desacoplada, de modo que la fuente de la imagen (móvil o gafas) no afecta al procesamiento posterior. El flujo general es el siguiente:

1. Detección y conexión de las Rokid Air.
2. Captura de imagen desde la cámara integrada en las gafas.
3. Procesamiento de la imagen por Gemini para obtener la descripción.
4. Reproducción del resultado por TTS para su entrega al usuario.

5.6.4. Pruebas y depuración

Durante el desarrollo, se realizaron diversas pruebas utilizando tanto emuladores como dispositivos físicos. El uso de gafas inteligentes implicó un entorno más complejo de pruebas, ya que fue necesario:

- Validar la compatibilidad de controladores en distintos modelos Android.
- Ajustar tiempos de respuesta y optimización del rendimiento.
- Depurar a ciegas, ya que el usuario final no ve directamente la pantalla del móvil.

En base a estos desafíos, se implementaron logs auditivos (por TTS) y vibraciones para confirmar acciones, mejorando así la experiencia de uso sin necesidad de interfaz visual.

5.7. Desafíos técnicos y soluciones adoptadas

Durante el desarrollo del proyecto se presentaron diversos desafíos técnicos relacionados principalmente con la integración de la inteligencia artificial (Gemini) y las gafas inteligentes Rokid Air dentro de un entorno Android. A continuación, se detallan los principales problemas encontrados y las soluciones adoptadas.

5.7.1. Integración inicial de modelos de IA en Android

El primer enfoque para implementar la funcionalidad de descripción de imágenes se centró en el uso de GPT-4 con capacidades visuales (GPT-4 Vision). Para ello, se desarrolló inicialmente un script en Python con el objetivo de ejecutarlo directamente desde la aplicación Android. La herramienta utilizada para esta integración fue Chaquopy, un plugin que permite ejecutar código Python dentro de Android Studio.

Sin embargo, este enfoque presentó diversos problemas:

- **Incompatibilidad de bibliotecas:** Las librerías necesarias para utilizar GPT-4 Vision no eran totalmente compatibles con Chaquopy, provocando errores durante la instalación y ejecución.
- **Limitaciones del entorno Android:** Ejecutar código Python en un entorno móvil presentó restricciones de rendimiento, gestión de dependencias y compatibilidad general.

- **Alternativa mediante API propia:** Se planteó la opción de desplegar un servidor que ejecutase el script Python y ofrecerlo como un endpoint al que la app pudiera hacer peticiones. No obstante, esta solución requería mantener un servidor activo de forma constante, lo cual no era viable dentro de los objetivos del TFG ni del alcance del proyecto.

Finalmente, se optó por abandonar esta vía y buscar una solución más robusta y nativa dentro del entorno Android.

5.7.2. Migración a Gemini y problemas de compatibilidad

La siguiente opción fue utilizar Gemini, el modelo de lenguaje multimodal desarrollado por Google, que sí ofrecía integración directa mediante SDKs compatibles con Android y bibliotecas en Java/Kotlin.

Inicialmente, el uso de Gemini se integró sin grandes dificultades, pero comenzaron a surgir problemas al intentar combinarlo con las funciones del SDK de las gafas Rokid. En concreto:

- **Poca documentación:** La documentación oficial del SDK de Rokid era escasa, lo que obligó al equipo a recurrir a foros, repositorios no oficiales y comunidades de desarrolladores para encontrar soluciones funcionales.
- **Incompatibilidad con Android SDK 34:** El proyecto había comenzado utilizando la versión más reciente del SDK de Android (34), pero muchas funciones del SDK de las gafas no funcionaban correctamente bajo esta versión. Tras varias pruebas, se identificó que el SDK de Rokid estaba diseñado y probado sobre la versión SDK 32.

5.7.3. Adaptación del proyecto a versiones compatibles

El siguiente paso fue intentar adaptar la aplicación al código de ejemplo oficial proporcionado por Rokid. Se migró el proyecto para usar el SDK 32, pero esto generó un nuevo conflicto: las bibliotecas de Gemini no eran compatibles con SDK 32.

Tras múltiples pruebas, se encontró un punto intermedio utilizando la versión SDK 33 de Android, que ofrecía compatibilidad tanto con las funciones de Gemini como con parte del SDK de las gafas. No obstante, surgió otro obstáculo técnico relevante:

- **Problemas con la vista de cámara:** El Surface necesario para renderizar la vista de la cámara en las gafas no funcionaba correctamente al integrarlo con nuestra lógica personalizada. La imagen no se mostraba correctamente, y tras debuggear múltiples veces no se logró encontrar un problema concreto.

5.7.4. Solución final adoptada

La solución definitiva fue tomar como base la clase de cámara incluida en el proyecto de ejemplo de Rokid, que sí gestionaba correctamente el Surface y los flujos de captura desde las gafas. Se modificó esa clase para adaptarla a nuestras necesidades, en especial:

- Añadir la funcionalidad de capturar una imagen bajo demanda.
- Enviar dicha imagen al sistema de Gemini.
- Reproducir la descripción mediante el módulo de TTS.

Este enfoque permitió finalmente una integración funcional entre las gafas Rokid, el modelo Gemini y el sistema Android en una versión estable (SDK 33).

Capítulo 6

Pruebas y Validación

6.1. Introducción

El proceso de pruebas de la aplicación se ha centrado principalmente en la verificación funcional de los distintos módulos desarrollados, con el objetivo de garantizar que cada funcionalidad se comporta según lo esperado en diferentes entornos. Las pruebas han sido realizadas de forma manual por los propios integrantes del equipo a lo largo del desarrollo del proyecto, y no se han llevado a cabo pruebas automatizadas ni test con usuarios externos debido a la naturaleza académica y a las limitaciones de tiempo y recursos del TFG.

A pesar de no contar con un protocolo formal de pruebas, se ha validado el funcionamiento de las principales características de la aplicación en escenarios reales, especialmente en lo referente a la integración con las gafas inteligentes Rokid Air, cuya disponibilidad estaba restringida al entorno del laboratorio.

6.2. Entorno de pruebas

Las pruebas se han realizado en dos contextos distintos:

- **Desde casa:** se validaron las funcionalidades que no dependían del uso de las gafas Rokid. Estas pruebas incluían la captura de imágenes con la cámara del dispositivo móvil, la selección de imágenes desde la galería, el envío a Gemini, la reproducción de la descripción mediante TTS y que el flujo de navegación dentro de la aplicación respondiera como se esperaba en cada caso.

  Además, se verificó el correcto funcionamiento de la base de datos, incluyendo que las validaciones se realizaran adecuadamente al crear una cuenta o iniciar sesión (manejo de errores y restricciones), que el historial de consultas funcionara correctamente y el acceso y funcionalidades del perfil.

- **En el laboratorio:** se realizaron las pruebas relacionadas con la integración de las gafas Rokid Air, ya que solo se disponía de acceso a este hardware en el laboratorio. Allí se probó la conexión de las gafas, la inicialización de su cámara, la captura de imágenes y la correcta integración con el flujo de descripción de la aplicación y la base de datos.

Herramientas y dispositivos utilizados:

- Android Studio (versión Electric Eel y posterior)
- Dispositivo Android (Samsung Galaxy A52, Android 13)
- Gafas Rokid Air conectadas vía USB-C
- Gemini API para la descripción de imágenes
- Base de datos SQLite local

6.3. Estrategia de validación

La estrategia de validación consistió en comprobar que cada funcionalidad individual funcionaba de forma aislada, y luego verificar que se integraba correctamente dentro del flujo global de la aplicación. Para ello se realizaron pruebas manuales tras cada avance en el desarrollo.

Entre las tareas realizadas durante la validación se incluyen:

- Verificación de que cada fuente de entrada (cámara del móvil, galería, gafas) permitía obtener correctamente una imagen.
- Comprobación del envío de imágenes al sistema Gemini y recepción de la respuesta.
- Confirmación de que el texto recibido se reproducía mediante TTS.
- Evaluación de la estabilidad y rendimiento de la aplicación durante el uso.
- Manejo de errores: comprobar qué ocurre si no hay conexión, si no se concede un permiso, si las gafas no están conectadas, etc.
- Validación de la autenticación: se comprobó que el sistema permite iniciar sesión solo con credenciales válidas, y que los errores se gestionan correctamente.
- Registro de usuarios: se verificó que las validaciones del formulario impiden el alta con datos incompletos o incorrectos, y que se notifica adecuadamente al usuario.
- Gestión del historial: se comprobó que las consultas realizadas se almacenan correctamente, pueden visualizarse en la pantalla correspondiente y eliminarse.
- Perfil de usuario: se validaron funcionalidades como el cierre de sesión, el cambio de contraseña con validaciones y la navegación entre vistas protegidas por autenticación.

6.4. Casos de prueba

A continuación, se detallan algunos de los casos de prueba más representativos que se llevaron a cabo, tanto a nivel de funcionalidad individual como de integración general de la aplicación.

Descripción mediante cámara del móvil

Este caso consistió en verificar el flujo completo desde la captura de una imagen hasta la generación de una descripción accesible por voz. Se accedió a la cámara desde la interfaz principal, se tomó una fotografía y, automáticamente, esta fue enviada al sistema Gemini. La respuesta generada por el modelo fue correctamente recibida, mostrada en pantalla y reproducida mediante el sistema de texto a voz (TTS). Se confirmó que no hubo errores en el proceso y que la experiencia fue fluida y coherente.

Descripción a partir de una imagen de la galería

Se seleccionó una imagen existente desde el explorador del sistema, con el fin de comprobar que esta vía de entrada funcionaba igual que la cámara. La imagen se procesó correctamente, la respuesta textual fue clara y también se reprodujo por TTS sin problemas.

Integración con gafas Rokid

Para probar la compatibilidad con las gafas Rokid, se conectaron estas al móvil mediante USB-C. La aplicación detectó la conexión e inició una actividad independiente para capturar la imagen desde la cámara de las gafas. Tras capturar la foto, el flujo de procesamiento y reproducción fue equivalente al del móvil. Se prestó especial atención a posibles errores durante el cambio de actividad o la gestión del dispositivo externo, y se verificó que todo funcionara con normalidad.

Gestión de errores por falta de conexión

Se realizó una prueba desconectando el dispositivo de la red para simular una situación sin acceso a Internet. Al intentar enviar una imagen para su análisis, la aplicación detectó la falta de conexión y mostró un mensaje de error informativo. No se produjo ningún cierre inesperado ni comportamiento anómalo, y al restablecer la conexión se pudo continuar con normalidad.

Registro de un nuevo usuario

En esta prueba se intentó registrar usuarios con combinaciones inválidas: campos vacíos, contraseñas demasiado cortas, contraseñas no coincidentes o nombres de usuario ya existentes. En todos los casos, la aplicación reaccionó correctamente mostrando mensajes de error claros, evitando que el usuario pudiera continuar hasta introducir datos válidos. Finalmente, se completó el registro con éxito y se comprobó que el usuario quedaba registrado en la base de datos.

Inicio de sesión

Se verificaron diferentes escenarios de autenticación. Con credenciales incorrectas, se mostraron errores apropiados sin permitir el acceso. En el caso de que fuesen ambas correctas, el inicio de sesión fue exitoso y redirigió al usuario a la pantalla principal. Se confirmó además que la sesión persistía al cerrar y volver a abrir la aplicación.

Historial de consultas

Se realizaron distintas pruebas para verificar el correcto almacenamiento de las consultas (imagen y descripción) tanto si fueron seleccionadas desde la galería como si fueron tomadas tanto de la cámara del móvil como desde las gafas. Se verificó que se almacenase en el historial asociado al usuario el cual había iniciado sesión. Se verificó que las entradas quedaban correctamente registradas en la base de datos y eran accesibles desde la interfaz de incio. Además, se pudo consultar el historial completo y eliminar entradas individuales mostrándose previamente un aviso de confirmación para evitar posibles errores.

También se comprobó que, en el caso de no haber iniciado sesión o de no existir consultas previas, la aplicación mostraba mensajes adecuados informando al usuario de la situación, en lugar de dejar la vista vacía o generar errores.

Gestión del perfil de usuario

Se accedió al perfil con una sesión iniciada para comprobar que esta sección estaba protegida. Dentro del perfil (una vez iniciada la sesión), se probó la funcionalidad de cambio de contraseña, introduciendo tanto combinaciones válidas como errores comunes (contraseña actual incorrecta, nueva demasiado corta, campos no coincidentes). Los mensajes de validación funcionaron correctamente. Por último, se usó la opción de cerrar sesión, comprobando que la aplicación regresaba al inicio de forma limpia y sin errores cerrando la sesión de manera correcta.

6.5. Resultados y observaciones

Las pruebas realizadas a lo largo del desarrollo permitieron detectar errores de forma temprana, ajustar funcionalidades y afinar detalles que influyen directamente en la experiencia de usuario. A nivel general, se observó que la integración con el sistema Gemini fue estable y consistente, sin interrupciones ni comportamientos inesperados. El sistema de texto a voz (TTS) funcionó correctamente en todos los escenarios probados, proporcionando una interacción auditiva fluida y confiable.

El manejo de imágenes desde la cámara del móvil y la galería del sistema fue sencillo y sin incidencias destacables. En cambio, la integración con las gafas Rokid supuso el mayor desafío técnico, especialmente en lo relacionado con la gestión del `SurfaceView`, por lo que se acabó optando por adaptar el código de ejemplo oficial proporcionado por el fabricante, permitiendo resolver los problemas iniciales y garantizar la compatibilidad. El uso de las gafas aportó, además, una gran mejora en la experiencia de uso, ya que permitía capturar imágenes sin necesidad de interactuar físicamente con el móvil.

Durante las sesiones de prueba también se detectaron ciertas inconsistencias relacionadas con la actualización de la interfaz. Por ejemplo, tras realizar una consulta, esta se almacenaba correctamente en la base de datos pero no se mostraba de inmediato en la vista de historial. De forma similar, al eliminar una entrada, esta desaparecía del almacenamiento pero seguía visible hasta cambiar de pestaña o reiniciar la vista. Esto también se observó al realizar las pruebas relacionadas con el perfil de los usuario, en las que se observó que, al iniciar sesión desde la pantalla de perfil, el sistema redirigía correctamente, pero la vista permanecía como si no se hubiese autenticado. Todos estos comportamientos fueron detectados gracias a las pruebas manuales realizas y posteriormente

analizados y solucionados mediante una mejor gestión del ciclo de vida de las actividades y la implementación de recargas forzadas del contenido tras operaciones clave.

Finalmente, aunque las pruebas se llevaron a cabo de forma manual y sin participación directa de usuarios finales con discapacidad visual, el proceso permitió validar todas las funcionalidades previstas y asegurar un funcionamiento estable del sistema en las condiciones contempladas. Estas observaciones fueron fundamentales para detectar errores sutiles, mejorar la lógica interna de ciertas operaciones y refinar el diseño de la experiencia de uso.

6.6. Limitaciones durante las pruebas

Durante el proceso de pruebas se encontraron diversas limitaciones que condicionaron su alcance:

- **Acceso restringido a las gafas Rokid:** solo se podía probar esta funcionalidad en el laboratorio, lo cual limitaba el número de pruebas y su frecuencia.
- **Falta de pruebas con usuarios reales:** no se pudo validar la aplicación con personas con discapacidad visual, por lo que no se tiene un feedback directo de parte del público objetivo.
- **Ausencia de pruebas automatizadas:** todo el proceso de validación se basó en pruebas manuales, lo cual implica una mayor carga de trabajo y menor cobertura de casos extremos o errores poco frecuentes.
- **Dificultades para integrar la cámara de las gafas dentro de la MainActivity:** la cámara de las gafas solo se pudo utilizar correctamente en una actividad separada, lo que limita la fluidez de la experiencia de usuario.

Capítulo 7

Resultados

7.1. Resultados de la descripción de imágenes

Para evaluar las descripciones se ha decidido crear una serie de métricas objetivas, para así poder tener un resultado objetivo. Las métricas más importantes que se han elegido han sido el detectar elementos importantes, como lo son las escaleras, las señales o carteles de información, pasos de peatones o paradas de autobús, así como el situarlas correctamente en la imagen y dar la información correcta, en el caso de que sean carteles. También se valora que reconozca otros elementos que se han considerado menos importantes, como establecimientos, máquinas expendedoras o tornos. Estas métricas se han elegido de esta manera ya que la descripción a veces puede ser muy general y perder detalles importantes, lo que le acaba restando utilidad a la aplicación teniendo en cuenta cuál es su idea de uso, por lo que se prefiere dar prioridad a estos elementos por pensar que su descripción le puede ser más útil al usuario frente a los elementos secundaraios.

Por ello se han seleccionado una serie de 17 imágenes en distintos entornos como estaciones de tren, de metro o en la calle, y 2 imágenes de cartas de restaurantes para ver cómo de bien se comporta dando este tipo de información. En estas imágenes hay un total de 59 elementos prioritarios de los cuales reconoce correctamente 51 elementos. Cabe resaltar la capacidad de leer carteles de información, que aún siendo difíciles de leer por la calidad de la foto es capaz de dar una una información bastante cercana a la realidad, aunque evidentemente tiene mejores resultados si el cartel es legible fácilmente. Los elementos prioritarios también se localizan correctamente en la imágen, y en la mayoría de casos se nombran los elementos secundarios que más resaltan en la imágen. En las imágenes de las cartas de restaurantes, se lee correctamente toda la información que se da en la carta, incluyendo los ingredientes y los precios de cada uno de los platos, aunque en algunos casos, aparecen carácteres de formato como asteríscos en la descripción, por lo que se hace menos comprensible al reproducir la descripción mediante el TextToSpeech.

Los principales errores en las descripciones son al haber alguno de los elementos prioritarios en segundo plano, por lo que se considera que la descripción tiene un buen rendimiento.

7.2. Funcionamiento general de la aplicación

7.2.1. Historial de consultas

En la Figura 7.1 se muestra el resultado de acceder al menú de inicio, el cual contiene el historial de las consultas que ha realizado el usuario cuando ha iniciado sesión previamente. En este caso, todavía no se han realizado búsquedas, por lo que se informa al usuario de ello.

En la Figura 7.2 se observa el mensaje mostrado cuando se accede al historial sin haber iniciado sesión.

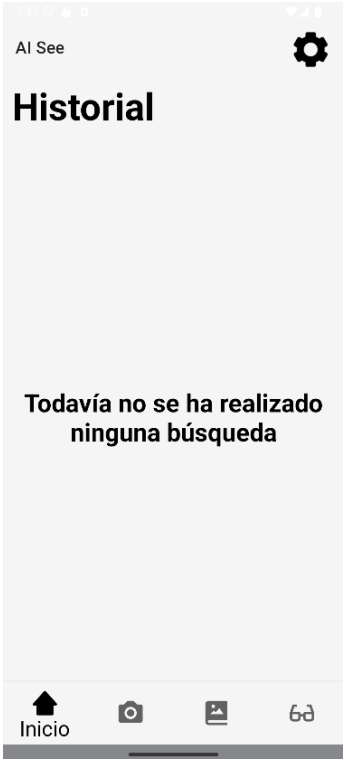

Figura 7.1: Historial vacío con usuario registrado

Figura 7.2: Acceso al historial sin sesión iniciada

A continuación, en la en la Figura 7.3 se muestra la pantalla de inicio con las últimas consultas realizadas por el usuario registrado.

Cuando un usuario pulsa sobre una consulta ya guardada del historial, esta se amplia y se vuelve a reproducir la descripción por los altavoces. Figura 7.4

Figura 7.3: Historial con descripciones generadas por el sistema

Figura 7.4: Consulta del historial ampliada

Como se muestra en la Figura 7.5, un usuario puede eliminar una consulta guardada de su historial deslizando esta hacia la derecha o izquierda y posteriormente sale una alerta para confirmar que realmente desea eliminar la consulta en caso de error. Figura 7.6

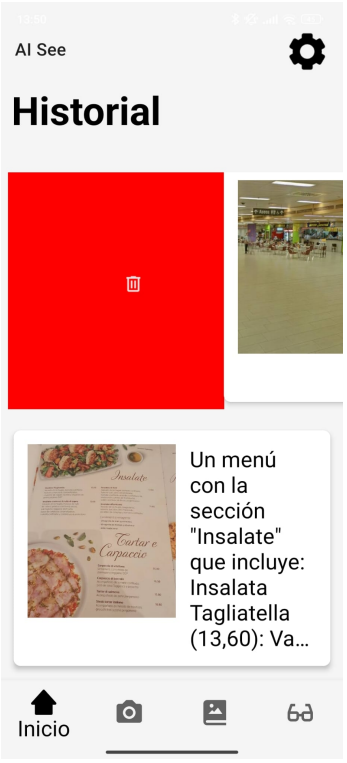

Figura 7.5: Eliminar consulta del historial

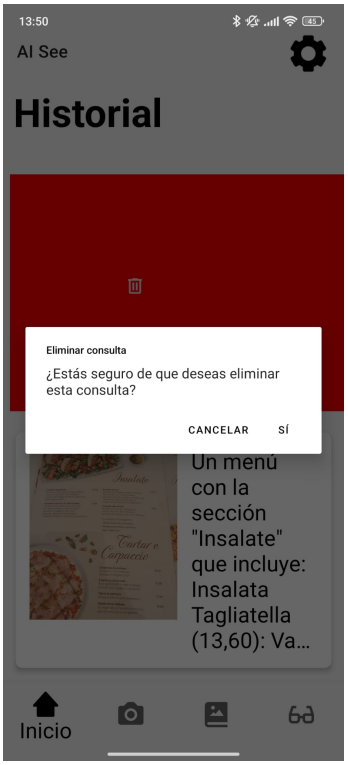

Figura 7.6: Confirmación para eliminar una consulta

7.2.2. Registro e Inicio de sesión

En la Figura 7.8 y 7.7 se muestran las pantallas de la funcionalidad de gestión de usuarios tanto del incio de sesión como la creación de la cuenta de un nuevo usuario.

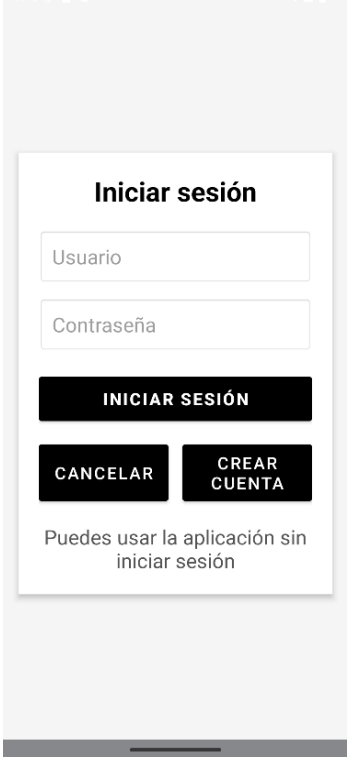

Figura 7.7: Inicio de sesión

Figura 7.8: Registro de un nuevo usuario

La aplicación proporciona mensajes de error claros cuando se introducen datos inválidos tanto en el inicio de sesión como en el registro. A continuación se muestran ejemplos visuales de estos mensajes.

En la Figura 7.9 se presentan distintos escenarios de error durante el proceso de registro: utilización de un nombre de usuario ya existente, las contraseñas no coinciden, la contraseña tiene menos de ocho caracteres y nombre de usuario contiene caracteres no válidos.

Por otro lado, la Figura 7.10 muestra el mensaje de error que aparece cuando el usuario intenta iniciar sesión con credenciales (nombre y/o contraseña) incorrectas.

Todos estos errores son detectados y notificados inmediatamente al usuario con mensajes específicos para que el usuario sepa exactamente cual es el problema para poder solucionarlo.

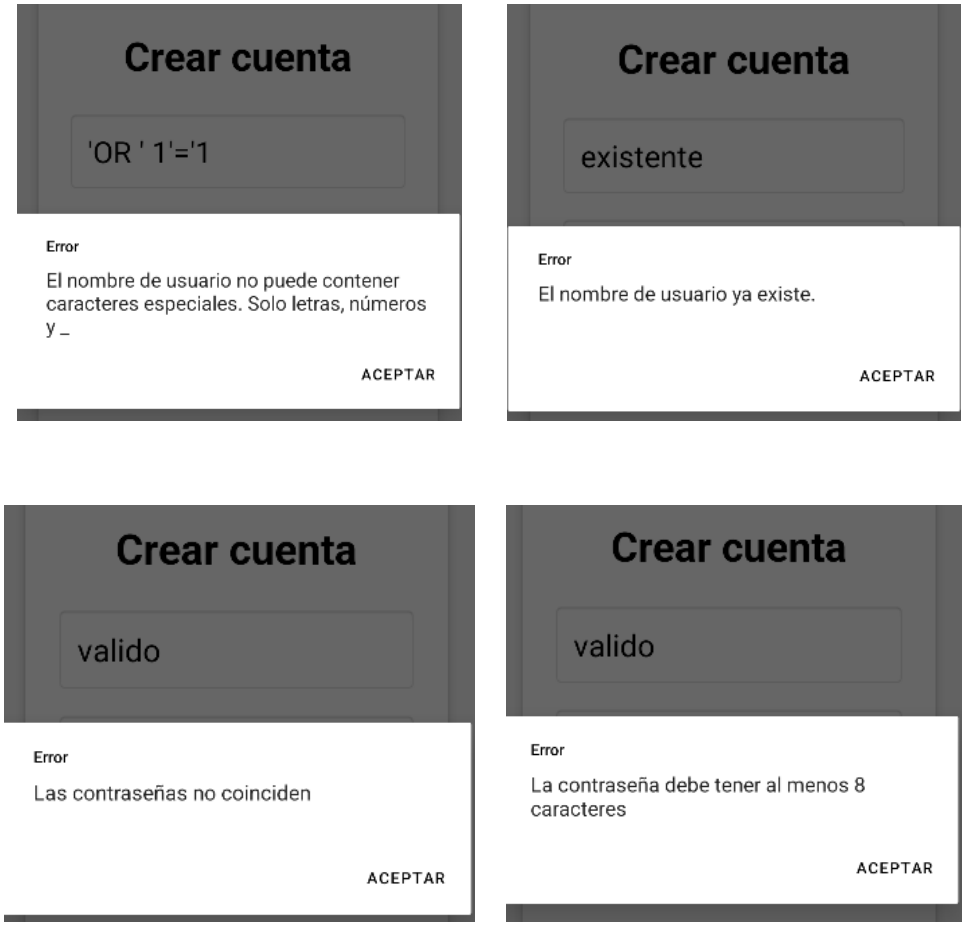

Figura 7.9: Mensajes durante el proceso decrear una cuenta nueva.

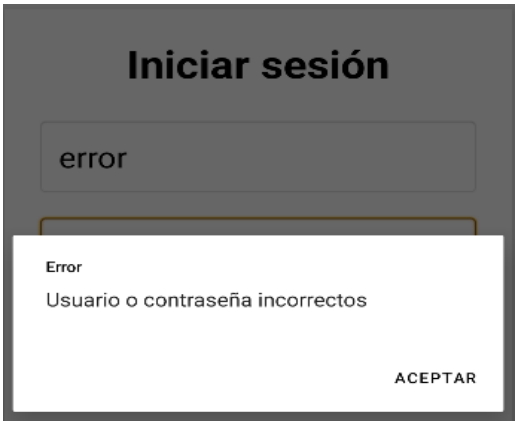

Figura 7.10: Mensaje durante el proceso de iniciar sesión.

7.2.3. Perfil del usuario

En esta sección se muestran las diferentes vistas disponibles dentro del apartado de perfil de usuario, según el estado de autenticación.

La Figura 7.11 presenta la pantalla de perfil cuando un usuario ha iniciado sesión. Se muestra su nombre de usuario y se habilitan las opciones de cambiar la contraseña o cerrar sesión. La Figura 7.12 corresponde a un usuario que no ha iniciado sesión y accede a su perfil, donde se le indica que debe iniciar sesión para acceder.

Por último, la Figura 7.13 muestra la pantalla de cambio de contraseña, donde se solicita la contraseña actual y la nueva (con confirmación) antes de aplicar los cambios.

Figura 7.11: Perfil de un usuario logueado

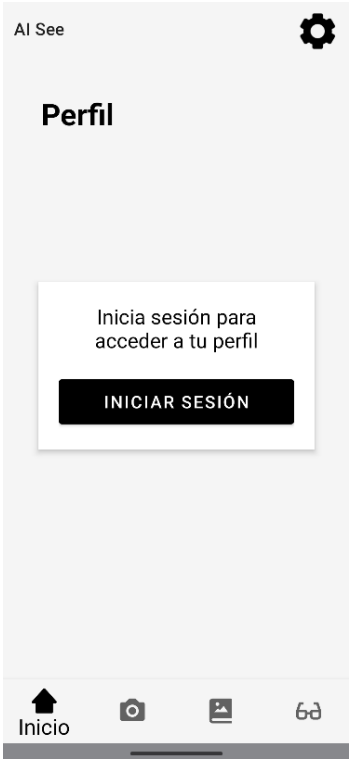

Figura 7.12: Perfil de un usuario sin loguear

Figura 7.13: Cambiar contraseña

Durante el proceso de cambio de contraseña, la aplicación valida diferentes condiciones. La Figura 7.14 muestra mensajes de error que aparecen cuando la nueva contraseña no cumple con la longitud mínima, las contraseñas no coinciden o la contraseña actual, la cual desea cambiar, es incorrecta. Además, se incluye un diálogo de confirmación para evitar la pérdida accidental de cambios si el usuario decide cancelar la operación.

Figura 7.14: Mensajes durante el proceso de cambio de contraseña.

Finalmente, cuando el usuario pulsa la opción de cerrar sesión desde su perfil, se muestra una alerta de confirmación (Figura 7.15) que previene cierres accidentales

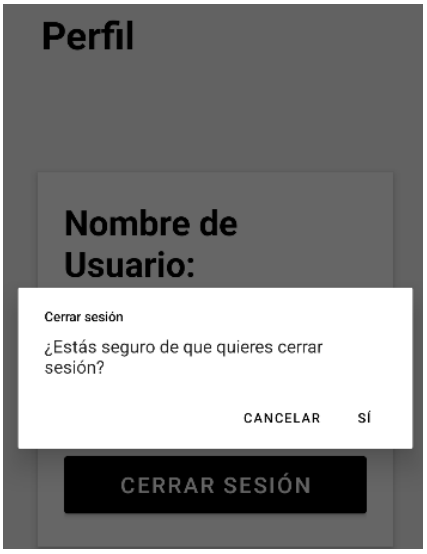

Figura 7.15: Confirmación al cerrar sesión desde el perfil del usuario.

7.2.4. Rendimiento de la aplicación

Mediante la herramienta profiler de Android Studio se ha hecho una medición del rendimiento de la aplicación haciendo un uso normal de la misma, para así poder ver los recursos que consume durante su ejecución.

En cuanto al uso de la CPU, generalmente es muy poco demandante teniendo en torno a un 6 % de uso durante la mayoría del tiempo llegando a picos de entorno un 23 % en el momento de tomar fotos, bajando rápidamente una vez tomada la fotografía. Por lo tanto, no es una aplicación demandante para la CPU del dispositivo y no sobrecarga el sistema.

En cuanto al consumo de la batería del dispositivo, también vemos una baja demanda, excepto al usar la cámara, que puede llegar a requerir un 1.5 amperios mientras se está usando, lo cual, es un consumo bastante alto, aunque normal ya que usar la cámara exige notablemente al dispositivo.

7.2.5. Tiempo de respuesta

El tiempo de respuesta del modelo depende principalmente de la conexión a internet de la que se disponga en el momento de realizar la consulta. Con una conexión estándar el tiempo de respuesta está en torno a los 4 segundos pudiendo bajar o subir dependiendo de la conexión. En el caso de que no se tenga conexión salta un mensaje de error indicando que no se ha podido realizar la consulta.

7.3. Captura y selección de imágenes

La aplicación permite al usuario capturar una imagen directamente desde la cámara del dispositivo o seleccionar una ya existente desde la galería. Estas funcionalidades están accesibles mediante la barra de navegación inferior. Figura 7.16 y Figura 7.17

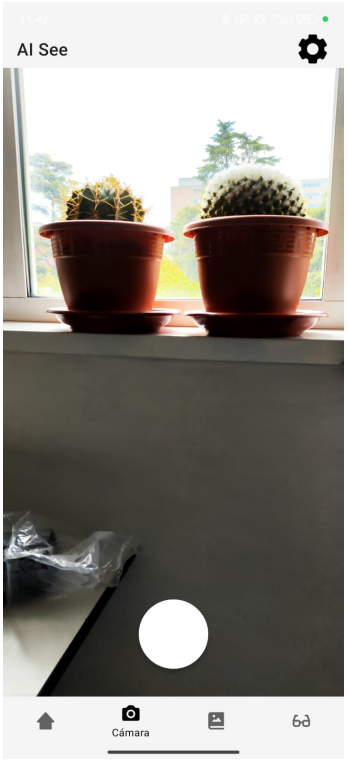

Figura 7.16: Pantalla de cámara para capturar imágenes desde la aplicación.

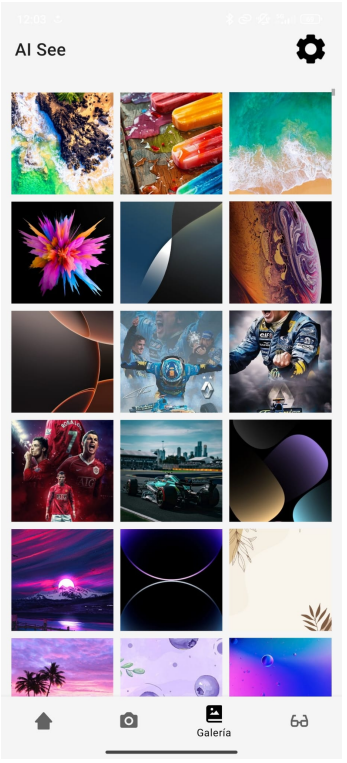

Figura 7.17: Pantalla de galería para seleccionar una imagen del dispositivo.

7.4. Cambio de idioma

La aplicación permite seleccionar el idioma de la interfaz desde el menú de configuración, como se muestra en la Figura 7.18. Actualmente se encuentran disponibles tres idiomas: español, inglés y francés. El cambio de idioma afecta únicamente los textos visibles en la interfaz, sin modificar la estructura o funcionalidad de las pantallas es por esto que solo se muestran algunas de las pantallas por idioma ya que en contenido no cambia a la interfaz en español ya mostrada.

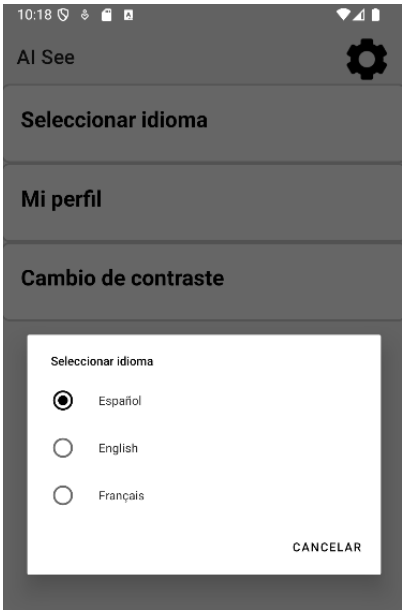

Figura 7.18: Selección de idioma desde la configuración de la aplicación.

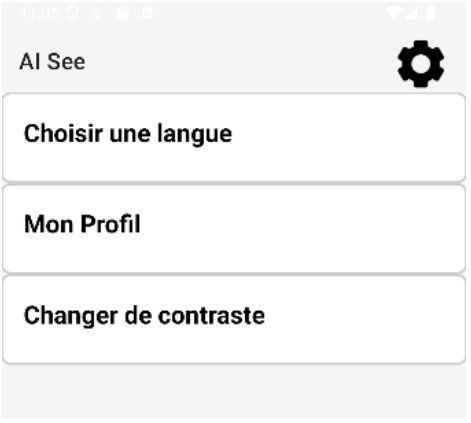

Figura 7.19: Menú de ajustes en francés

Las Figuras 7.21 y 7.20 contienen ejemplos del historial de consultas cuando ya se han realizado consultas en inglés y francés. En la Figura 7.22 se muestra una consulta del historial en inglés ampliada.

Figura 7.20: Historial con consultas en francés

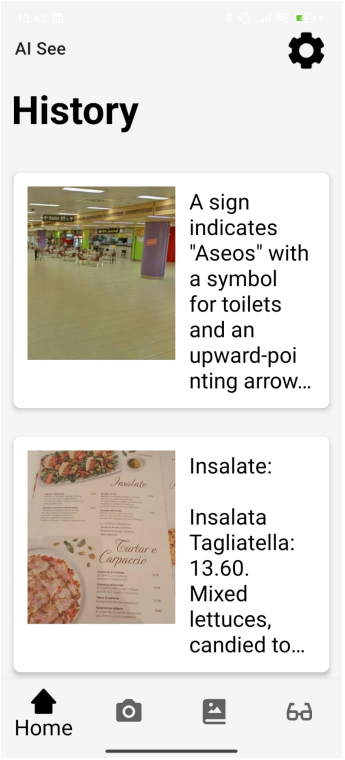

Figura 7.21: Historial con consultas en inglés

Figura 7.22: Consulta del historial en inglés ampliada

La Figura 7.23 muestra la pantalla de inicio de sesión en francés. Del mismo modo, en la Figura 7.24 se visualiza la interfaz para registrar un nuevo usuario en inglés. Finalmente, la Figura 7.25 presenta la página de perfil también en inglés.

Figura 7.23: Inicio de sesión en francés

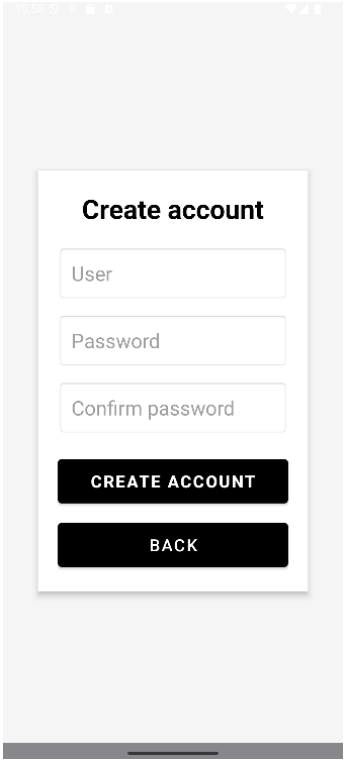

Figura 7.24: Registro de un nuevo usuario en inglés

Figura 7.25: Página de perfil en inglés

De la misma forma, todos los mensajes de error que muestra el usuario están traducidos para cada idioma como se muestra en los siguientes ejemplos: Figura 7.26, Figura 7.27, Figura 7.28

Figura 7.26: Error usuario o contraseña erroneo en francés

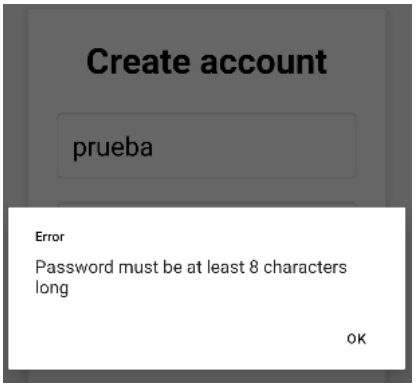

Figura 7.27: Error contraseña con longitud insuficiente en inglés

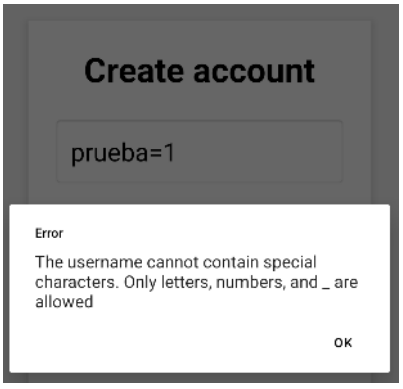

Figura 7.28: Error nombre de usuario con caracteres inválidos en inglés

7.5. Conexión con las gafas

La figura 7.29 muestra la pantalla al acceder a la pantalla de gafas, sin que las mismas estén conectadas.

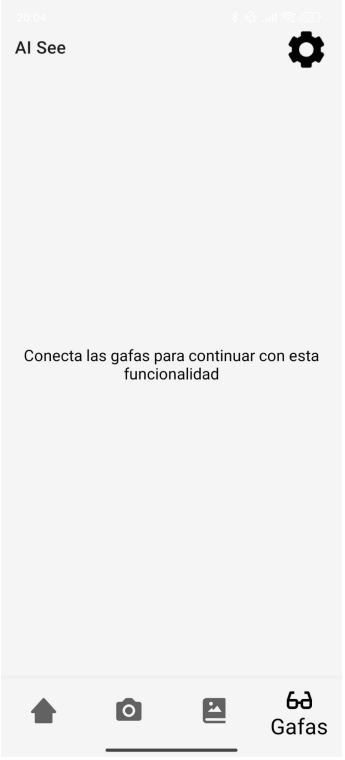

Figura 7.29: Pantalla gafas sin conexión

La figura 7.30 muestra la pantalla de las gafas al acceder a la pantalla de gafas cuando estas están conectadas. Presionando el botón, la cámara empieza a funcionar y el botón pasa a ser para dejar de usar la cámara, así como se muestra en la figura 7.31.

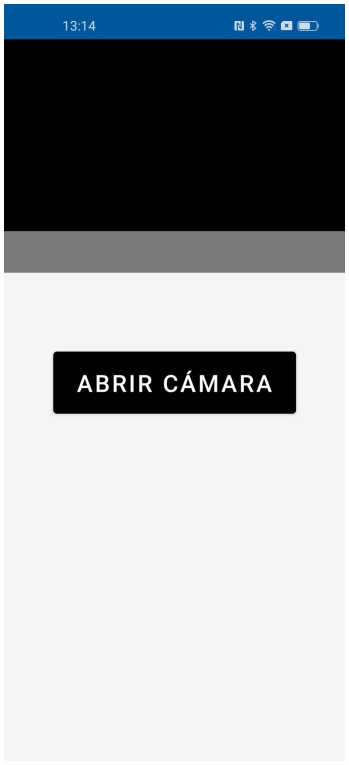

Figura 7.30: Pantalla para empezar a usar la cámara de las gafas

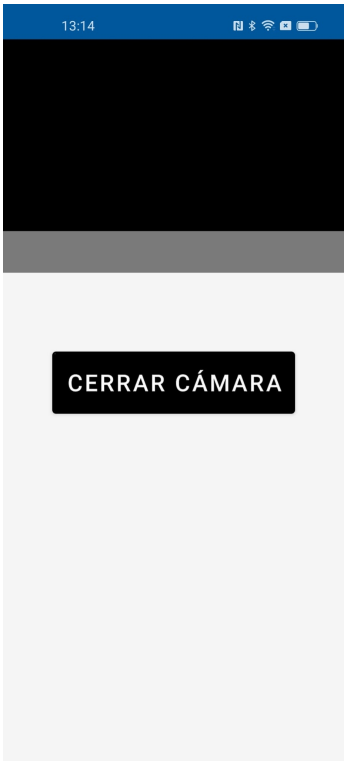

Figura 7.31: Pantalla para dejar de usar la cámara de las gafas

7.6. Modo alto contraste

Todas las pantallas mostradas en los apartados anteriores tienen su correspondiente en el modo alto contraste.

La aplicación permite cambiar a un modo de alto contraste en el menú de configuración, como se muestra en la Figura 7.32. Este modo cambia todas las pantallas de la aplicación para que sean más legibles por personas con baja visión, incluido el propio menú de configuración, como se muestra en la Figura 7.33

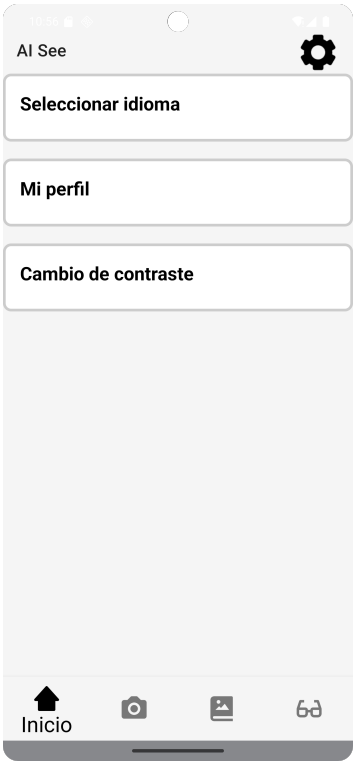

Figura 7.32: Menu de configuracion antes de cambiar al modo alto contraste.

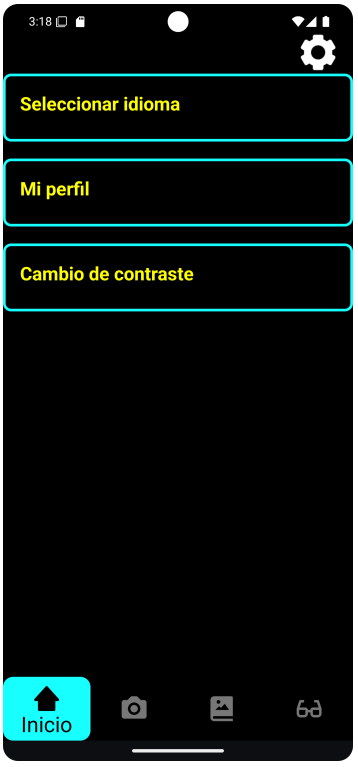

Figura 7.33: Menú de configuración en el modo alto contraste.

Las Figuras 7.34 y 7.35 muestran como cambia el menú de inicio de la aplicación cuando esta está en el modo alto contraste, con todas las funcionalidades anteriormente descritas.

Figura 7.34: Menú de inicio sin sesión iniciada en el modo alto contraste.

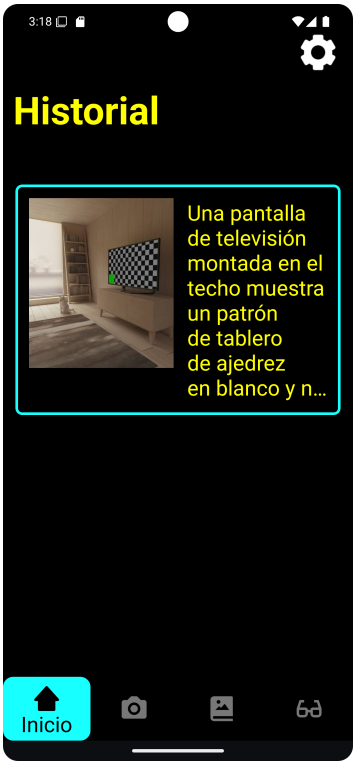

Figura 7.35: Menú de inicio con sesión iniciada en el modo alto contraste.

Si hacemos click en una consulta se muestra en grande como vemos en la Figura 7.36.

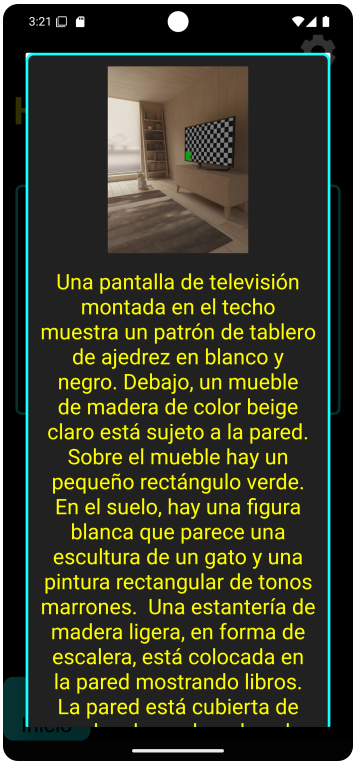

Figura 7.36: Pantalla de una consulta ampliada en modo alto contraste

Como vemos en las Figuras 7.37, 7.38 y 7.39, el modo alto contraste también cambia las pantallas de creación de una cuenta, inicio de sesión y cambio de contraseña, respectivamente.

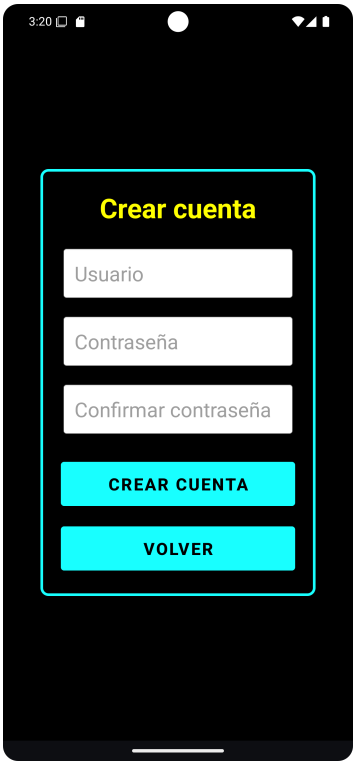

Figura 7.37: Pantalla de una consulta ampliada en modo alto contraste

Figura 7.38: Pantalla de una consulta ampliada en modo alto contraste

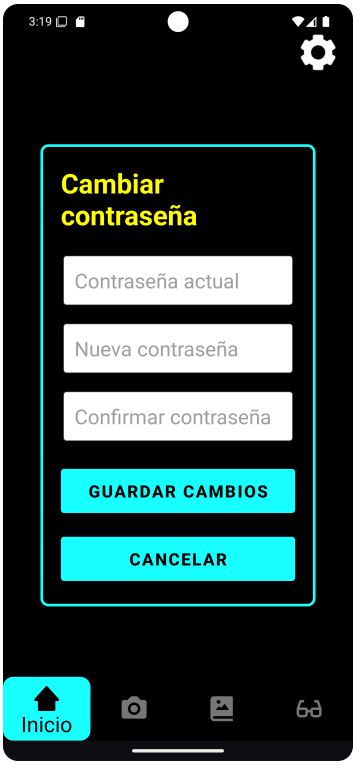

Figura 7.39: Pantalla de una consulta ampliada en modo alto contraste

Capítulo 8

Conclusiones y Trabajo Futuro

8.1. Conclusiones

El desarrollo de esta aplicación ha supuesto una gran experiencia tanto a nivel técnico como personal, permitiendo aplicar conocimientos adquiridos durante el grado en un proyecto con un claro enfoque social y accesible. La aplicación diseñada tiene como objetivo principal ayudar a personas con discapacidad visual parcial o total a comprender su entorno mediante descripciones generadas por inteligencia artificial y reproducidas por audio.

A lo largo del proyecto se ha trabajado con tecnologías actuales, como los modelos multimodales de lenguaje (Gemini) y la integración con dispositivos externos como las gafas inteligentes Rokid Air, lo cual ha añadido complejidad, pero también un gran valor al sistema. La posibilidad de capturar imágenes desde diferentes fuentes (cámara del móvil, galería y gafas) proporciona versatilidad y adapta la experiencia a distintas situaciones y necesidades del usuario.

El diseño de la aplicación ha seguido principios de accesibilidad, simplicidad e inclusión, ofreciendo una interfaz adaptada para que el usuario, dependiendo de sus capacidades visuales, pueda utilizarla sin barreras.

Este proyecto ha demostrado que es posible combinar tecnología puntera, accesibilidad y usabilidad en una única solución móvil. Además, se ha puesto de manifiesto la importancia de diseñar pensando en la diversidad de los usuarios, entendiendo que la tecnología debe ser una herramienta que acerque, no que excluya.

Las decisiones tomadas a lo largo del desarrollo, tanto en lo técnico como en lo funcional, han permitido sentar unas bases sólidas para futuras mejoras y ampliaciones, que se tratarán en los próximos apartados.

8.2. Objetivos cumplidos

Al inicio del proyecto se definieron una serie de objetivos tanto funcionales como técnicos, con el fin de desarrollar una aplicación útil, accesible y tecnológicamente viable para ayudar a personas con visibilidad reducida a identificar su entorno mediante descripciones auditivas. A continuación, se detallan los principales objetivos planteados:

- **Implementar un sistema de descripción automática de imágenes.**
  El sistema utiliza el modelo Gemini, capaz de interpretar imágenes y generar descripciones detalladas que son reproducidas mediante audio.
- **Ofrecer múltiples formas de entrada de imágenes.**
  El usuario puede capturar imágenes desde la cámara del móvil, seleccionar fotos de la galería o utilizar las gafas Rokid Air como fuente de entrada visual.
- **Integrar la aplicación con gafas inteligentes.**
  Se ha logrado establecer una integración funcional con las gafas Rokid Air, utilizando su cámara como input, adaptando el flujo de captura, y solucionando los problemas encontrados asociadas al SDK.
- **Implementar una base de datos para la gestión de usuarios e historiales.**
  Se ha desarrollado una base de datos local que permite registrar usuarios, almacenar consultas realizadas y acceder al historial de descripciones, lo que facilita una experiencia personalizada y un seguimiento útil para el usuario.
- **Desarrollar una aplicación Android accesible para personas con visibilidad reducida.**
  Se ha diseñado una interfaz adaptativa, con buen contraste, navegación simple y soporte auditivo mediante TTS, cumpliendo con los principios de accesibilidad.
- **Aplicar medidas de seguridad para la protección de datos.**
  La aplicación incorpora cifrado en el almacenamiento de datos sensibles, y se han aplicado buenas prácticas de seguridad en la gestión de información, garantizando la privacidad y protección de los usuarios.
- **Incluir un sistema de audio para entregar las descripciones generadas.**
  Todas las descripciones son leídas al usuario utilizando el sistema de Text-To-Speech (TTS) del dispositivo, facilitando una experiencia completamente auditiva.
- **Desarrollar una arquitectura modular y escalable.**
  El sistema se ha construido de forma desacoplada, permitiendo en el futuro añadir nuevas funciones o modelos sin tener que reestructurar la aplicación por completo.
- **Ofrecer soporte multilingüe.**
  Se han añadido dos idiomas adicionales al español (inglés y francés) que permiten cambiar tanto el idioma de la interfaz como el de las descripciones generadas, aumentando la accesibilidad a una audiencia más amplia.

Estos objetivos cumplidos reflejan el compromiso del equipo con la funcionalidad, la accesibilidad y la innovación tecnológica. El trabajo ha conseguido materializar una solución real y útil que responde a una necesidad concreta, cumpliendo satisfactoriamente con los fines propuestos al inicio del TFG.

8.3. Limitaciones encontradas

A pesar de haber alcanzado la mayoría de los objetivos planteados, durante el desarrollo del proyecto se presentaron diversas limitaciones técnicas y prácticas que condicionaron algunas deci-

siones e influyeron en el flujo de trabajo. A continuación, se detallan las principales limitaciones encontradas:

Problemas de compatibilidad entre bibliotecas y versiones del SDK de Android.

La integración simultánea de las bibliotecas de Gemini y el SDK de las gafas Rokid presentó varios conflictos, especialmente en lo relativo a la compatibilidad con diferentes versiones del SDK de Android Studio. En concreto, las gafas funcionaban mejor en versiones más antiguas (SDK 32), mientras que Gemini requería versiones más recientes. Finalmente, se encontró un punto intermedio funcional en SDK 33, aunque con ciertas restricciones en cuanto a estabilidad y mantenimiento futuro.

Falta de documentación del SDK de Rokid.

Uno de los mayores retos fue trabajar con el SDK proporcionado por Rokid, cuya documentación era escasa y poco detallada. Esto dificultó la implementación de funciones clave como la gestión de la cámara integrada en las gafas, forzando al equipo a buscar soluciones en foros, ejemplos de código y recursos externos.

Problemas con la visualización de la cámara de las gafas.

La implementación del Surface necesario para mostrar la imagen capturada desde las gafas no funcionaba correctamente con nuestra estructura inicial. Se optó finalmente por reutilizar el código de ejemplo oficial de Rokid, adaptándolo a las necesidades de la aplicación, lo que supuso una pérdida de flexibilidad en el diseño de la lógica propia.

Limitaciones en el uso de modelos de lenguaje avanzados desde Android.

En la fase inicial del proyecto se intentó integrar GPT-4 con capacidades visuales utilizando scripts en Python ejecutados desde Android mediante herramientas como Chaquopy. Esta solución resultó inviable debido a incompatibilidades de bibliotecas y limitaciones de rendimiento. También se valoró el uso de una API externa propia, pero se descartó por requerir un servidor activo y permanente.

Dependencia de conexión a Internet.

Tanto Gemini como el sistema TTS requieren conexión a Internet para funcionar correctamente. Esto implica que la aplicación no puede operar de forma completamente offline, lo cual puede limitar su uso en determinadas situaciones o entornos.

Dificultades con la implementación de la base de datos.

La integración de la base de datos para almacenar usuarios e historiales presentó varios retos técnicos relacionados con la configuración del entorno de desarrollo. Aunque en un primer intento se

intentó utilizar MySQL o la librería Room se tuvieron distintos problemas de compatibilidad entre las librerías y el plugin de Gradle y diversos problemas de dependencias. Esto llevó a implementar una gestión de datos de forma manual utilizando almacenamiento local, lo que limita la escalabilidad y mantenibilidad futura de la aplicación.

Conclusión limitaciones

Estos problemas obligaron a replantear la estrategia de persistencia y buscar configuraciones más estables, lo que consumió tiempo de desarrollo y limitó temporalmente el avance en otras funcionalidades.

Aunque estas limitaciones no impidieron la consecución de los objetivos principales, sí condicionaron ciertas decisiones de diseño y desarrollo. En algunos casos, se recurrió a soluciones alternativas, y en otros, se pospusieron mejoras para futuras versiones. Estas experiencias han aportado un aprendizaje valioso sobre la integración de hardware y software en proyectos reales y multidisciplinares.

8.4. Líneas de trabajo futuro y posibles mejoras

A partir de la implementación desarrollada, se han identificado diversas líneas de trabajo que permitirían mejorar y ampliar la funcionalidad de la aplicación. A continuación, se describen las principales propuestas de mejora:

- **Mejora de la integración con Smart Glasses:** durante el desarrollo se evidenció que la documentación del SDK de Rokid es limitada, por lo que se considera importante consolidar una integración más robusta, documentada y compatible con futuras versiones del dispositivo. Al igual que el estudio de compatibilidad e implementación con nuevos modelos de gafas que puedan surgir en el mercado.
- **Funcionalidad offline:** dado que el modelo Gemini y el sistema TTS requieren conexión a Internet, se plantea desarrollar una versión básica que funcione en modo local, mediante almacenamiento en caché o integración de modelos preentrenados.
- **Incorporación de análisis de vídeo:** extender la funcionalidad actual mediante el procesamiento de secuencias de vídeo, en lugar de imágenes estáticas, con el objetivo de ofrecer una descripción continua del entorno. Esta mejora permitiría detectar cambios dinámicos en la escena, identificar obstáculos en movimiento y aumentar la precisión en la interpretación de contextos complejos.
- **Ampliación del historial de consultas:** se propone añadir a las consultas metadatos como la fecha y hora, el dispositivo utilizado (smartphone o gafas inteligentes) y la posibilidad de clasificar las descripciones mediante etiquetas o marcadores en distintos historiales. Esto facilitaría la recuperación y organización de información pasada.
- **Optimización de la base de datos:** actualmente, la gestión de datos se realiza mediante almacenamiento local con SQLite. Se plantea migrar a soluciones más modernas como Room (Android Jetpack), o una base de datos remota para permitir sincronización en múltiples dispositivos.

- **Ampliación del soporte multilingüe:** actualmente se ofrece soporte en español, inglés y francés. Se propone añadir nuevos idiomas para ampliar el alcance de la aplicación y adaptarse a usuarios de distintas regiones.
- **Mayor personalización de la interfaz:** se recomienda ofrecer ajustes adicionales de accesibilidad, como modificación del tamaño de texto, temas con diferentes combinaciones de contraste, y modos de visualización adaptados a diversas patologías visuales.
- **Disponibilidad de distintos prompts:** dado que las necesidades de cada usuario son muy distintas, se considera añadir diferentes prompts para distintos escenarios o preferencias, así como por ejemplo, dar tres versiones de prompts, una con una descripción más detallada, otra con una descripción más resumida y una última con la finalidad de leer documentos, de esta manera, la aplicación se puede ajustar más a las preferencias y necesidades de cada usuario. También hay que tener en cuenta que cuánto más sencillo sea el prompt se pueden esperar mejores resultados del modelo, ya que este se puede especializar en ciertos elementos y ofrecer una descripción más útil para cada momento.

Estas propuestas servirían como base para una evolución progresiva del sistema, con el objetivo de hacerlo más útil, accesible y escalable para una mayor diversidad de usuarios.

Capítulo 9

Contribuciones personales

En esta sección se especifica la función que ha tenido cada uno de los integrantes durante el desarrollo del proyecto para poder completar el trabajo de manera satisfactoria teniendo en cuenta los objetivos previos que se habían planteado a la hora de idear la aplicación. Esto ha sido posible gracias al compromiso, conocimiento y habilidades de cada uno de los miembros del equipo, que al ser cada uno de una titulación diferente, han podido apoyarse mutuamente y aprender de las áreas de conocimiento que se han desarrollado con más profundidad en los diferentes grados. Así aunque ciertos miembros hayan tomado ciertas responsabilidades, todos los miembros han podido aportar algo diferente que ha ayudado a acelerar el proceso de desarrollo, o bien, resolver ciertos problemas que tenían parte del desarrollo paralizado.

Álvaro Gómez Tejedor

Álvaro Gómez aporta experiencia en programación en distintos lenguajes y entornos, teniendo una gran capacidad de adaptación a nuevos entornos y paradigmas de programación. Su gran fortaleza es su caracter perseverante, que le permite hacer frente a diferentes problemas, así como la capacidad de abordarlos con calma y tranquilidad logrando aportar diferentes puntos de vista para abordarlos.

En colaboración con otros miembros del grupo dió una primera idea de cómo podía verse la aplicación final, dando ideas de los diferentes diseños de las vistas que se pueden encontrar en la aplicación, así como definir algunos de los objetivos que debía cumplir la versión final de la aplicación.

La primera tarea de Álvaro fue desarrollar la vista de la galería, en la que se muestran las fotos que el usuario tiene en su dispositivo, para que así pueda elegir una de ellas para recibir la descripción de la imagen seleccionada. Al no tener conocimientos previos de android, tuvo que estudiar el entorno para pensar en las distintas maneras que había para desarrollar dicha funcionalidad. Para ello primero tuvo que aprender como se genstionaba la concesión de permisos que requieren las aplicaciones para acceder al almacenamiento interno del dispositivo. Teniendo ya el permiso, cargó las imágenes de tal modo que aparezcan siempre las imágenes que se han guardado en la galería más tarde, ya que se entiende que en caso de querer recibir una descripción de una imagen que está en la galería del dispositivo, la imagen seleccionada va a ser reciente, por lo que era la decisión de diseño más acertada. Y como la idea era poder seleccionar las imágenes para recibir su descripción, no podían ser únicamente imágenes que se mostraran, el usuario debía poder interaccionar con ellas, para ello se implementó un adapter para gestionar el click sobre una imagen y mandar esta al modelo, para recibir la respuesta. Como el tipo de variable de las imágenes que se muestran en

esta vista no es igual al tipo de una imagen tomada con la cámara, la función que manejaba la petición de las consultas al modelo no se ajustaba a este caso, por lo que se desarrolló una segunda función a partir de la primera que gestionase este caso.

La siguiente tarea que realizó Álvaro fue implementar la vista de ajustes para añadir seguidamente la opción de cambiar el idioma de la aplicación. Para lo cuál se investigó como se podía desarrollar esta funcionalidad de manera que fuera escalable para poder traducir la aplicación a más idiomas en un futuro. Se creó una lista de los idiomas para que el usuario eligiera su preferido, y al seleccionar uno se reinicia la aplicación seleccionando el nuevo idioma elegido. De esta manera, guardando todos los textos traducidos en su respectiva versión de la carpeta values, una por cada idioma, hace que se traduzcan todos, siendo muy fácil añadir más idiomas.

También desarrolló la lectura en voz alta de las descripciones de las imágenes, para ello se usó la clase de Android TextToSpeehc que permite convertir en voz el texto que se le mande leer. Para ello en primer lugar se declaró la instancia en la actividad principal para acceder a la instancia en los fragmentos de la cámara y el álbum, que es donde se usa, para una vez generada la descripción esta se lea en voz alta. Como esta clase lee en el idioma en el que esté la app, con la previa implementación de la selección del idioma ya estaba gestionado los diferentes idiomas en los que tenía que leer el TextToSpeech. Posteriormente se implementó también la lectura de la descripción generada por las gafas, que al estar en otra actividad diferente y no tener acceso a la instancia de la actividad principal, se creó una nueva instancia para poder gestionar dicha funcionalidad. La implementación en este caso es un poco diferente, ya que la descripción se genera en el viewModel pero se tiene que leer en la actividad, por lo que se implementó mediante un callback, de tal manera que cuando se genera la descripción en el viewModel, se invoca la función en la actividad de la cámara, que es la que termina leyendo la descripción de la fotografía tomada con la cámara de las gafas inteligentes.

Habiendo terminado sus tareas y viendo que la implementación de la conexión de las gafas inteligentes con la aplicación estaba paralizada debido a diversos errores se unió a intentar buscar la solución para poder proseguir con el desarrollo de la aplicación. Esta fue la parte más complicada a la hora de desarrollar la aplicación, tanto por la diversidad de errores que había cómo por la limitación de los recursos de la que se disponía, ya que era necesario trabajar en el laboratorio para poder usar el equipo neceario. Tras intentar implementar esta funcionalidad en la versión de la app que teníamos y tener diversos errores para los que no se encontraban solución, se decidió cambiar el proyecto a una versión mucho más parecida a la demo que daba el fabricante para probar las gafas. Aún así no se solucionaron los problemas, ya que encontrar una compatibilidad de versiones que permitiera el uso de Gemini para las descripciones como el sdk para poder hacer la conexión de las gafas no fue sencillo. Teniendo ya un entorno donde pudieran desarrollarse todas las funcionalidades de la aplicación, se empezó a buscar el error concreto que impedía la conexión, lo que logró solucionar con éxito Daniel. Aunque volvieron a salir problemas con la conexión de la cámara de las gafas, ya que la aplicación se cerraba al intentar usarla, o la pantalla de las gafas mostraba una pantalla en blanco en vez de la cámara en sí. Al final el error terminó siendo la petición de un permiso que estaba anticuado para la versión de android que estabamos usando para desarrollar la aplicación, por lo que una vez gestionado bien ese permiso se pudo seguir con el desarrollo de la aplicación para generar la descripción de la fotografía tomada por la cámara de las gafas, como pulir otros detalles de la aplicación que habían sido delegados a un segundo nivel de prioridad.

Una de las tareas más importantes fue la selección de un prompt que cumpliera con las espectativas, ya que no todas las descripciones de una imagen son igual de útiles, por ello había

que pensar en cuales eran los principales puntos a tener en cuenta para valorar una descripción, se seleccionaron una serie de métricas objetivas y se probaron en los distintos prompts, haciendo pequeños cambios para lograr unas descripciones correctas, consistentes y útiles. Estos prompts se probaron con diferentes imágenes y varias veces, ya que cada respuesta es variable, por lo que había que garantizar que todas las descripciones fueran correctas, más allá de que si se piden varias descripciones para la misma imágen, todas serán distintas entre sí. Una vez encontrado un prompt que satisfaciera correctamente las espectativas, se tradujo al resto de idiomas que soporta la aplicación para que en todos los idiomas se consiguieran unas respuestas parecidas, más allá de que los modelos de lenguaje pueden mostrar mejor o peor funcionamiento dependiendo del idioma en el que se use.

Una vez la aplicación desarrollada se pasó a realizar las pruebas para evaluar el funcionamiento de la misma. Álvaro realizó las pruebas de rendimiento y latencia, esto sin contar que el funcionamiento general de la aplicación se ha ido probando por todos los miembros durante el desarrollo de la aplicación, para ir comprobando que todas las funcionalidades evolucionaran correctamente. Para las pruebas de latencia, se midió el tiempo que tardaba en responder Gemini a cada una de las consultas, esto se hizo mientras se probaban cada uno de los prompts para aprovechar el tiempo de una manera más eficiente. Para el rendimiento de la aplicación se usó la herramienta de Android Studio profiler, que permite monotorizar el rendimiento de la aplicación en tiempo real, mientras ejecutaba la aplcación en el dispositivo movil. Tras correr varias simulaciones de uso, comprobó el rendimiento medio comparando todas entre sí, para tener un visión más fiable tanto del consumo de la CPU como del consumo de energía que la aplicación tiene sobre el dispositivo durante un uso normal de la misma.

Laura de Cara Molina

Desde el inicio del proyecto, Laura ha desempeñado un papel importante tanto en el desarrollo técnico como en la organización del trabajo en equipo. Su actitud resolutiva y colaborativa ha resultado clave para coordinar las tareas del grupo y mantener una buena comunicación. A nivel individual, ha desarrollado distintas áreas de la aplicación, desde la base de datos y seguridad, hasta el diseño de vistas funcionales y navegación, así como en la evaluación y selección de tecnologías apropiadas.

En las primeras fases del proyecto, Laura trabajó junto con Álvaro Gómez en la elaboración de los primeros bocetos de diseño, con el objetivo de definir una estructura clara tanto para la interfaz como para el flujo general de la aplicación. Esta etapa conceptual sirvió para establecer una visión compartida sobre cómo debía organizarse y comportarse la aplicación, definiendo unos objetivos iniciales.

Cabe destacar que al inicio del desarrollo Laura no tenía experiencia previa en Android Studio, por lo que aprendió de manera autodidacta para poder familiarizarse con el entorno y adquirir progresivamente una mayor fluidez en el desarrollo móvil.

Una vez asentadas las bases, una de las primeras tareas realizadas fue la elaboración del estudio del estado del arte, que sirvió de punto de partida para entender el contexto tecnológico del proyecto. En él se analizaron soluciones ya existentes orientadas a personas con discapacidad visual, incluyendo tanto aplicaciones móviles como dispositivos de smart glasses disponibles en el mercado, como OrCam MyEye, Seeing AI o Retiplus. Además, se exploró el papel de la inteligencia artificial en la generación de descripciones automáticas. Este análisis permitió no solo identificar funcionalidades útiles y limitaciones presentes en las soluciones actuales, como la falta de historial reutilizable o interfaces poco accesibles, sino también establecer criterios de accesibilidad y definir

los primeros requisitos de la aplicación.

Posteriormente, Laura participó en las primeras fases de diseño de la arquitectura de la aplicación, encargándose de la estructuración inicial de los fragmentos y del sistema de navegación de la barra inferior. Para ello, definió un esquema basado en una actividad principal que actuaba como controlador del flujo entre los distintos fragmentos. Cada uno de estos, representa una sección funcional independiente de la aplicación (historial, cámara, galería, conexión con gafas), y su navegación se gestiona mediante FragmentManager y transacciones controladas por la barra de navegación. Inicialmente, este flujo se implementó de forma estática, como prototipo navegable sin funcionalidad de fondo, lo que permitió validar la estructura general y preparar su posterior integración con los módulos de lógica y persistencia.

A continuación, se centró en el diseño de la base de datos. Esta se realizó con un enfoque estructurado y conforme a los principios establecidos por el Reglamento General de Protección de Datos (RGPD). Se evitó la recopilación innecesaria de datos personales, aplicando el principio de minimización, y se definieron estructuras con integridad referencial clara que facilitaran la trazabilidad y coherencia de los datos. La base de datos fue diseñada en su totalidad por Laura, incluyendo la creación del modelo entidad-relación, la normalización hasta la Tercera Forma Normal (3FN), y la especificación de relaciones entre entidades mediante claves primarias y foráneas. En un primer intento, se exploró la conexión con una base de datos externa en MySQL, pero debido a problemas de integración y compatibilidad en entorno Android, se optó finalmente por una solución local utilizando SQLite con Java, lo que además permitió un mayor control sobre el tratamiento de los datos sin depender de servidores externos.

Laura también diseñó e implementó todas las vistas asociadas al ciclo de vida del usuario, incluyendo el registro, el inicio de sesión, la visualización y edición del perfil, el cambio de contraseña y el cierre de sesión. Cada operación (como insertar nuevos usuarios, recuperar información por credenciales o actualizar datos del perfil) fue gestionada mediante métodos específicos definidos en las interfaces DAO, los cuales proporcionaban una interfaz segura y desacoplada para realizar operaciones sobre la base de datos, permitiendo así una separación clara entre la lógica de negocio y la capa de persistencia.

Desde esta fase inicial se tuvieron en cuenta principios de seguridad en el diseño de la base de datos y en la lógica de la aplicación. Posteriormente, Laura llevó a cabo un análisis detallado de diferentes algoritmos de encriptación, valorando aspectos clave como la compatibilidad, eficiencia computacional y robustez frente a ataques. Como resultado, se aplicó una capa de cifrado basada en algoritmos de hash (SHA-256) para proteger las contraseñas, asegurando que nunca se almacenaran en texto plano. Este mecanismo se aplicaba tanto en el alta de usuario como en las operaciones de autenticación, mediante la comparación del hash generado.

Asimismo, se cifraron campos sensibles relacionados con las imágenes y descripciones generadas, garantizando la confidencialidad de la información incluso ante accesos no autorizados. Para evitar vulnerabilidades comunes, se implementaron validaciones exhaustivas en los formularios y en el backend, como filtros frente a inyecciones SQL, control de entradas maliciosas o restricciones en accesos concurrentes. Estas medidas permitieron construir una arquitectura fiable y alineada con las buenas prácticas de desarrollo seguro.

Para mejorar la experiencia de usuario, se diseñó un sistema de retroalimentación visual que mostraba mensajes claros y específicos en cada caso: errores de autenticación, campos vacíos, contraseñas no seguras o confirmaciones tras realizar cambios correctamente. Estos mensajes están implementados principalmente mediante cuadros de diálogo (AlertDialog), adaptados al contexto

y al tipo de acción. Laura fue responsable de redactar e implementar estos mensajes de feedback que guían al usuario en caso de errores, validaciones fallidas o acciones exitosas. De este modo, se proporcionó una experiencia coherente, accesible y autónoma, permitiendo al usuario entender en todo momento qué estaba ocurriendo y cómo debía actuar.

Una de las tareas más relevantes fue el desarrollo completo de la funcionalidad del historial de consultas, que permite a los usuarios acceder a las descripciones generadas previamente por el sistema. Laura diseñó e implementó el flujo completo, desde el almacenamiento de cada consulta en la base de datos tras su generación (realizada por otras funcionalidades), pasando por su encriptación, hasta su recuperación y visualización estructurada en la interfaz. Para ello, desarrolló un RecyclerView personalizado mediante un adapter. Este componente incorporó funcionalidades como la eliminación de consultas mediante gestos *swipe*, la confirmación mediante diálogo y la posibilidad de ampliar consultas y reproducir de nuevo la descripción asociada mediante síntesis de voz, lo que aporta un importante valor en términos de accesibilidad para personas con discapacidad visual.

Asimismo, elaboró los diagramas de flujo correspondientes a todos los procesos vinculados a la base de datos (registro, inicio de sesión, edición de perfil, cambio de contraseña, etc.), así como los casos de uso funcionales que describen las principales interacciones del usuario con el sistema.

Por último, se revisaron las implicaciones legales del uso del modelo Gemini como servicio externo, evaluando los riesgos asociados a la transmisión de datos a terceros. Este análisis permitió definir un marco de uso ajustado a las exigencias del Reglamento General de Protección de Datos (RGPD) y a los principios éticos del proyecto, asegurando que toda la arquitectura técnica estuviera alineada con los requisitos legales y de privacidad establecidos.

Daniel Barroso Casado

Desde el inicio del proyecto, Daniel participó activamente en la generación de ideas iniciales en colaboración con el resto del equipo. Junto a sus compañeros, exploró posibles enfoques tanto visuales como funcionales para la aplicación, aportando propuestas relacionadas con la estructura general, el diseño de la interfaz y las posibles funcionalidades de accesibilidad orientadas al usuario final.

Posteriormente, se encargó de construir la primera estructura del proyecto en Android Studio, iniciando el desarrollo de los principales fragmentos y pantallas, e implementando los botones y secciones esenciales que definirían la navegación de la aplicación. Con el objetivo de afianzar sus conocimientos, profundizó en el entorno Android revisando documentación y apuntes, ya que aunque contaba con una base previa, todavía no tenía experiencia completa con todas las particularidades del entorno.

Una vez asentada la estructura del proyecto, Daniel abordó el desarrollo del primer sistema de descripción de imágenes, utilizando para ello un script en Python que se conectaba con la API de OpenAI y permitía enviar imágenes para su análisis por parte de GPT-4. Para integrar este script en la aplicación Android, empleó la herramienta Chaquopy, que permite ejecutar código Python en proyectos Android. Sin embargo, durante esta fase surgieron importantes problemas de compatibilidad entre las bibliotecas necesarias para GPT-4 y el entorno de ejecución de Android, lo que obligó al equipo a replantear el enfoque.

Como alternativa, Daniel intentó realizar la integración directamente en Java, desarrollando un segundo script con la intención de acceder al modelo de GPT desde el propio entorno Android. No obstante, en ese momento no existía una versión del modelo compatible con Java para el análisis

de imágenes, lo cual provocó un nuevo cambio de estrategia.

Finalmente, tras evaluar varias opciones, propuso el uso del modelo Gemini de Google, considerando su mejor integración con el ecosistema Android y su compatibilidad con Kotlin y Java. Daniel se encargó de la implementación completa del sistema de descripción con Gemini, logrando integrar la generación de descripciones de imágenes tanto desde la cámara del dispositivo móvil como desde la galería del sistema. Una vez verificado el correcto funcionamiento de estas funcionalidades, se inició la fase más compleja del proyecto: la integración de las gafas inteligentes Rokid Air.

Esta etapa presentó numerosos desafíos. Por un lado, la limitada disponibilidad física de las gafas, que solo podían utilizarse en el laboratorio de la Facultad de Física y durante franjas horarias concretas, dificultó el avance. Por otro lado, surgieron problemas técnicos relacionados con la compatibilidad entre el SDK de las gafas, las bibliotecas de Gemini y las versiones de Android Studio. Daniel asumió la responsabilidad de liderar esta parte del desarrollo, enfrentándose durante meses a errores recurrentes en la inicialización y funcionamiento de la cámara de las gafas.

Daniel inició una fase intensiva de investigación y pruebas para integrar correctamente las gafas Rokid Air dentro del entorno de desarrollo en Android SDK 34, que era la versión base del proyecto desde el comienzo. Sin embargo, al comenzar esta integración se encontró con un problema crítico relacionado con la autogeneración de clases necesarias para el funcionamiento del SDK de las gafas. A diferencia del proyecto de ejemplo oficial proporcionado por Rokid, donde estas clases se generaban correctamente a partir del proceso de compilación, en el entorno del equipo estas clases no eran generadas de forma automática, impidiendo así el uso de funcionalidades clave como el control de la cámara integrada.

Para intentar solucionar este inconveniente, Daniel llevó a cabo varias modificaciones tanto en los archivos Gradle como en el archivo AndroidManifest.xml, con el objetivo de replicar el comportamiento del proyecto de ejemplo y forzar la autogeneración de las clases ausentes. A pesar de múltiples intentos y diferentes configuraciones, este enfoque no tuvo éxito y las clases necesarias seguían sin generarse, lo que bloqueaba completamente el acceso al hardware de las gafas.

Ante esta situación, se tomó la decisión de migrar el proyecto hacia la estructura del ejemplo oficial de Rokid, el cual sí incluía la configuración mínima funcional del SDK. Este proyecto base, sin embargo, estaba desarrollado sobre el SDK 32 de Android, lo que dio lugar a una nueva dificultad: múltiples bibliotecas utilizadas por el equipo, especialmente la del modelo Gemini, no eran compatibles con dicha versión del SDK. Esta incompatibilidad se convirtió en un nuevo obstáculo, ya que imposibilitaba unificar en un mismo entorno tanto la lógica de IA como la integración del hardware externo.

Durante esta fase, Daniel trabajó intensamente en colaboración con su compañero Álvaro para identificar una versión intermedia del SDK que ofreciera compatibilidad con ambos componentes. Tras semanas de pruebas, ajustes de dependencias y reconfiguración del entorno, lograron estabilizar el proyecto en la versión SDK 33 de Android, un punto de equilibrio que permitía compilar correctamente las bibliotecas de Gemini y, al mismo tiempo, ejecutar con éxito el SDK de las gafas Rokid. Esta solución marcó un hito dentro del proyecto, ya que permitió consolidar en una única base de código las funcionalidades de descripción inteligente de imágenes y el uso de las gafas como herramienta de captura.

Durante esta fase también se enfrentó a un problema crítico: la cámara de las gafas mostraba una vista en blanco sin que existiera un error claro en la lógica del código. Tras varios días de depuración, y gracias a una colaboración con su compañero Álvaro, identificaron que el fallo estaba en los permisos de la cámara: se estaban solicitando permisos obsoletos no válidos para las versiones

modernas de Android. Una vez corregido este punto, Daniel pudo continuar con éxito la integración.

Completada esta fase, implementó una nueva vista personalizada para la cámara de las gafas, desarrollando además la lógica necesaria para capturar imágenes mediante el botón principal físico de las Rokid Air y enlazar el resultado con el sistema de descripción mediante Gemini.

Además de sus tareas de desarrollo, también participó en la optimización del prompt principal para las descripciones generadas por la IA y colaboró en la traducción de textos de la aplicación a varios idiomas, ampliando así su accesibilidad.

En conjunto, el trabajo de Daniel ha sido esencial en las fases más técnicas y complejas del proyecto, especialmente en todo lo relacionado con la inteligencia artificial, la estructura del proyecto y la integración de hardware externo. Su capacidad de adaptación ante problemas inesperados y su perseverancia en la depuración han sido claves para lograr una aplicación funcional, estable y útil.

Hugo Herrero Desvoyes

Desde el inicio del proyecto, Hugo tuvo un papel fundamental en la concepción y planificación de la aplicación. Participó activamente en las primeras reuniones de ideación, aportando una gran cantidad de ideas relevantes sobre las funcionalidades que debía tener la aplicación para ser útil, intuitiva y competitiva. Su capacidad para visualizar el producto final desde una perspectiva tanto técnica como de usuario fue clave para definir las bases del desarrollo. Durante esta fase, se discutieron aspectos como el flujo de navegación, las características mínimas viables y la experiencia de usuario.

Una de las primeras tareas técnicas de las que se hizo cargo fue la implementación de la vista de la cámara, una de las funcionalidades centrales de la aplicación, ya que permite al usuario tomar fotografías directamente desde la interfaz. Para ello, Hugo se encargó de investigar y familiarizarse con el paquete CameraX de Android, una herramienta moderna que facilita el uso de la cámara del dispositivo dentro de las aplicaciones móviles. Este proceso requirió no solo conocimientos técnicos sobre el uso de librerías de terceros, sino también una comprensión profunda de cómo gestionar correctamente los permisos del sistema, algo esencial en el ecosistema Android. La correcta solicitud y manejo de permisos para acceder tanto a la cámara como al almacenamiento interno del dispositivo fue un reto técnico importante, ya que implicaba garantizar la seguridad del usuario y evitar errores de ejecución que podrían comprometer la funcionalidad básica de la aplicación.

La siguiente tarea que realizó, y en la que estuvo involucrado la mayor parte del tiempo, fue el diseño visual de la aplicación. Su enfoque se basó en lograr una interfaz limpia, sencilla y coherente, capaz de ofrecer una experiencia fluida al usuario. Se propuso un diseño minimalista, que pudiera adaptarse a todas las pantallas sin sacrificar estética ni funcionalidad. Este trabajo se realizó de forma iterativa, comenzando por la pantalla de inicio, que sirvió como plantilla para el resto. Hugo preparó diferentes propuestas de diseño que fueron presentadas al equipo, y en colaboración con los demás integrantes, se discutieron y ajustaron en función de los comentarios recibidos. Tras alcanzar un consenso sobre el estilo general, se procedió a su aplicación sistemática en toda la aplicación.

Este proceso implicó una reestructuración significativa de los archivos XML originales. En las primeras versiones, las vistas estaban diseñadas de forma individual, sin un sistema centralizado de estilos y temas, lo que complicaba cualquier modificación global de la apariencia. Hugo propuso y llevó a cabo una refactorización del código visual, estructurando los archivos de estilos de manera que permitieran aplicar cambios generales con facilidad, como ajustes de colores, fuentes, márgenes y bordes. Esta mejora no solo facilitó el mantenimiento del código, sino que también permitió una mayor consistencia visual entre pantallas.

Por otro lado, y de forma paralela, Hugo se encargó de asegurar la accesibilidad de la aplicación, tanto a nivel visual como funcional. Esto incluyó la creación de dos modos visuales adicionales: uno oscuro y uno de alto contraste, con su necesaria elección de colores y estilos. Uno de los desafíos más notables en esta fase fue precisamente lograr que el estilo de alto contraste no rompiera con la identidad visual de la aplicación. Además, algunos elementos como ciertos iconos o botones, perdían visibilidad al cambiar entre temas, por lo que fue necesario adaptar tanto los recursos gráficos como los estilos para asegurar que se mostraran correctamente en todos los modos. Una vez definidos los colores y estilos finales, y tras recibir comentarios del equipo y de los tutores del proyecto, Hugo implementó un botón específico en la pantalla de ajustes, el cual permite al usuario cambiar al modo de alto contraste en cualquier momento, independientemente de si su dispositivo está en modo claro u oscuro. Este botón fue integrado en la pantalla de ajustes, junto a la opción de cambio de idioma, agrupando en un solo lugar las opciones de personalización de la aplicación.

Cabe destacar que la implementación del diseño de alto contraste coincidió en el tiempo con la incorporación de nuevas funcionalidades por parte del equipo. Esto significó que cada nueva pantalla que se añadía al proyecto debía ser inmediatamente adaptada a los diferentes temas visuales. Se realizó una revisión minuciosa de cada vista para asegurar que el diseño fuera coherente en todos los modos de visualización, lo que añadió un grado extra de complejidad y coordinación al desarrollo.

Finalmente, gracias a su dominio del francés, Hugo ayudó a Alvaro con la traducción de los textos de la aplicación al francés, asegurándose de que fueran no solamente correctos, sino también apropiados para el contexto de uso, una aplicación móvil en el que suelen predominar tecnicismos y anglicismos.

En resumen, la contribución de Hugo al proyecto ha sido esencial en varias áreas clave, desde la concepción inicial hasta el diseño visual y la accesibilidad, garantizando que la aplicación no solo sea funcional, sino también atractiva y fácil de usar para todos los usuarios.

Bibliografía

[1] O. M. de la Salud, "Salud visual y discapacidad," 2023, dispoible en: https://www.who.int/es/news-room/fact-sheets/detail/blindness-and-visual-impairment, Consultado en febrero de 2025.

[2] IONOS, "¿qué es jira?" 2023, dispoible en: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-jira/, Consultado en septiembre de 2024.

[3] Jorge Gil, "Blitab, la primera tableta táctil para personas con discapacidad visual," 2018, disponible en: https://graffica.info/blitab-la-primera-tableta-tactil-para-personas-con-discapacidad-visual/, Consultado octubre de 2024.

[4] Fundación Atresmedia, "Blitab, la primera tablet diseñada para personas ciegas," 2018, disponible en: https://fundacion.atresmedia.com/proyectos-anteriores/normalizacion-discapacidad/noticias/blitab-primera-tablet-disenada-personas-ciegas_201809075b9268980cf2a248884568fb.html, Consultado en octubre de 2024.

[5] Real o Virtual, "Braille virtual con los guantes haptx para personas con discapacidad visual," 2023, disponible en: https://www.realovirtual.com/noticias/14238/braille-virtual-guantes-haptx-personas-discapacidad-visual, Consultado en octubre de 2024.

[6] Mutua Madrileña, "Wayfindr, innovación para mejorar tu vida," 2023, disponible en: https://www.mutua.es/blog/estilo-vida/wayfindr-discapacidad-visual_post/, Consultado en noviembre de 2024.

[7] Asociación Doce, "Wayfindr open standard, un proyecto para que las personas con discapacidad visual puedan moverse de forma autónoma," 2019, disponible en: https://asociaciondoce.com/2019/03/22/wayfindr-open-standard-un-proyecto-para-que-las-personas-con-discapacidad-visual-puedan-moverse-de-forma-autonoma/, Consultado en octubre de 2024.

[8] Verified Market Reports, "Smart glasses market," 2023, disponible en: https://www.verifiedmarketreports.com/es/product/smart-glasses-market/, Consultado en febrero de 2025.

[9] Let's Envision, "Let's envision glasses," 2025, disponible en: https://www.letsenvision.com/glasses/es, Consultado en enero de 2025.

[10] Retiplus, "Tecnología de realidad aumentada para personas con baja visión," 2023, disponible en: https://retiplus.com/, Consultado en noviembre de 2024.

[11] Meta, "Ray-ban meta smart glasses," 2023, disponible en: https://www.meta.com/es/ai-glasses/, Consultado en noviembre de 2024.

[12] Be My Eyes. (2025) Veamos el mundo juntos. Disponible en: https://www.bemyeyes.com/language/spanish, Consultado en octubre de 2024.

[13] Sunrise Medical. (2025) Apps móviles para personas con discapacidad. Disponible en: https://www.sunrisemedical.es/blog/apps-moviles-para-personas-con-discapacidad, Consultado en noviembre de 2024.

[14] CloudSight. (2025) Taptapsee. Disponible en: https://es.wikipedia.org/wiki/CloudSight, Consultado en noviembre de 2024.

[15] Fundación ONCE. (2020) Seeing ai: potente app de reconocimiento de entornos para ios. Disponible en: https://cti.once.es/noticias/seeing-ai-potente-app-de-reconocimiento-de-entornos-para-ios Consultado en octubre de 2024.

[16] iSocial, "From your eyes: herramientas tecnológicas para personas con discapacidad visual," 2023, disponible en: https://isocial.cat/es/from-your-eyes-herramientas-tecnologicas-personas-discapacidad-visual/, Consultado en diciembre de 2024.

[17] Y. Chen and otros, "Florence: A new foundation model for vision-language tasks," 2021, disponible en: https://arxiv.org/abs/2111.11432, Consultado en abril de 2025.

[18] V. autores, "Transformers: State-of-the-art models," 2020, disponible en: https://arxiv.org/abs/2012.12556, Consultado en abril de 2025.

[19] ——, "Vision transformers for image recognition," 2022, disponible en: https://arxiv.org/abs/2205.14100, Consultado en abril de 2025.

[20] WiseGuy Reports. (2023) Assistive technologies for visual impairment market. Disponible en: https://www.wiseguyreports.com/es/reports/assistive-technologies-for-visual-impairment-market, Consultado en diciembre de 2024.

[21] Orientatech. (2019) Orcam myeye 2.0. Disponible en: https://orientatech.es/orcam-myeye-20/, Consultado en noviembre de 2024.

[22] OrCam Technologies. (2023) Orcam myeye. Disponible en: https://www.orcam.com/es-es/orcam-myeye, Consultado en noviembre de 2024.

[23] IrisVision. (2023) Soluciones de visión mejorada. Disponible en: https://irisvision.com/, Consultado en octubre de 2024.

[24] Android Developers. (2025) Accessibility guidelines. Disponible en: https://developer.android.com/design/ui/mobile/guides/foundations/accessibility, Consultado en diciembre de 2024.

[25] W3C. (2024) Web content accessibility guidelines (wcag) 2.2. Disponible en: https://www.w3.org/TR/WCAG22/, Consultado en marzo de 2025.

[26] PanelFit Project. (2023) Minimización de datos – directrices de privacidad y gobernanza. Disponible en: https://guidelines.panelfit.eu/es/ia-3/marco-general/privacidad-y-gobernanza-de-datos/disposiciones-del-rgpd/minimizacion-de-datos/, Consultado en diciembre de 2024.

[27] EsPúblico Gestiona. (2024) El obligado cumplimiento del principio de minimización de datos. Disponible en: https://www.espublicogestiona.com/es/blog/el-obligado-cumplimiento-del-principio-de-minimizacion-de-datos/, Consultado en marzo de 2025.

[28] Google AI, “Gemini and your data – how gemini handles user content,” 2025, disponible en: https://support.google.com/gemini/answer/13594961, Consultado en abril de 2025.

[29] European Commission, “Regulation (eu) 2016/679 (general data protection regulation),” 2016, disponible en: https://eur-lex.europa.eu/eli/reg/2016/679/oj, Consultado en marzo de 2025.

[30] Android Developers, “Introducción a android studio,” 2025, disponible en: https://developer.android.com/studio/intro?utm_source&hl=es-419, Consultado en septiembre de 2024.

[31] free Code Camp, “Los principios solid de programación orientada a objetos,” 2022, dispoible en: https://www.freecodecamp.org/espanol/news/los-principios-solid-explicados-en-espanol, Consultado en febrero de 2025.

[32] Google AI, “Documentación de la api gemini para java,” 2025, disponible en: https://ai.google.dev/gemini-api/docs?hl=es-419#java, Consultado en diciembre de 2024.

[33] Android Developers. (2025) Guide to app architecture: Room and repository. Disponible en: https://developer.android.com/topic/architecture, Consultado en noviembre de 2024.

[34] OWASP Foundation. (2025) Sql injection prevention cheat sheet. Disponible en: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html, Consultado en enero de 2025.

[35] ——. (2025) Cryptographic storage cheat sheet. Disponible en: https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html, Consultado en febrero de 2025.

[36] NIST, “Advanced encryption standard (aes),” 2001, disponible en: https://csrc.nist.gov/publications/detail/fips/197/final, Consultado en febrero de 2025.

[37] Rokid, “Documentación para desarrolladores en yuque,” 2025, disponible en: https://rokid.yuque.com/ouziyq/tvgpgk/cgzdpz, Consultado en marzo de 2025.