

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
SANTIAGO - CHILE



**“APLICACIÓN AIMES: ASISTENTE INTELIGENTE PARA
USUARIOS CON DISCAPACIDAD VISUAL EN EL METRO DE
SANTIAGO”**

ANTONIA JESÚS FIGUEROA ARREDONDO

**MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL
TELEMÁTICO**

PROFESOR GUÍA:

NICOLÁS TORRES RUDLOFF

PROFESOR CORREFERENTE:

MARIE GONZÁLEZ INOSTROZA

AGOSTO - 2026



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Aplicación AIMES: Asistente inteligente para usuarios con discapacidad visual en el metro de Santiago

Nombre del candidato(a): Antonia Jesús Figueroa Arredondo

Carrera / Grado: Ingeniería Civil Telemática

Campus: San Joaquín Departamento: Departamento de Electrónica

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Nicolás Torres Rudloff, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO** contiene información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por (marcar una opción):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 19-10-2026

Firma: 

Estudiante o Candidato(a):

Fecha: 19-10-2026

Firma: 

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Agradecimientos

Quisiera agradecer a todas las personas que me acompañaron durante el desarrollo de esta memoria y a lo largo de mi formación universitaria. Este trabajo representa el cierre de una etapa marcada por esfuerzo, aprendizaje y perseverancia, y no habría sido posible sin el apoyo recibido en distintos momentos del proceso.

Agradezco especialmente a mi familia y a mis personas más cercanas, por su paciencia, compañía y contención durante los periodos de mayor exigencia. Sus palabras de ánimo y su presencia fueron fundamentales para continuar avanzando incluso en los momentos más difíciles.

A Pablo, mi mejor amigo, por acompañarme de forma constante durante este proceso, por escucharme, contenerme y recordarme que era capaz de llegar hasta el final incluso cuando yo misma lo dudaba. Su amistad fue un apoyo fundamental en esta etapa.

También agradezco a quienes contribuyeron desde el ámbito académico y técnico al desarrollo de este proyecto. Las orientaciones recibidas en etapas iniciales, junto con las conversaciones, revisiones y comentarios realizados durante el proceso, permitieron dar forma a AIMES y fortalecer su enfoque como solución móvil de asistencia.

Finalmente, agradezco a las personas con discapacidad visual cuyas experiencias y necesidades inspiraron el propósito de esta memoria. Este trabajo busca aportar, desde la tecnología, a una movilidad más autónoma, accesible y digna en entornos urbanos complejos como el Metro de Santiago.

Con gratitud, cierro esta etapa reconociendo no solo el resultado obtenido, sino también el crecimiento personal y profesional que significó llegar hasta aquí.

Para ella.

Porque, aun cuando el trabajo consumía la mayor parte de sus días, nunca dejó de creer que este momento llegaría. Finalmente logró cumplir un sueño que por mucho tiempo esperó su oportunidad, demostrando que la perseverancia siempre encuentra su recompensa. Espero que culmine esta etapa con orgullo, sabiendo que cada esfuerzo, cada sacrificio y cada momento que se dedicó a sí misma valieron la pena. Que este logro le recuerde siempre que es capaz de alcanzar todo aquello por lo que decide luchar.

Resumen

En Chile, la discapacidad visual constituye una limitación significativa para el desplazamiento autónomo, especialmente en entornos de transporte público de alta complejidad como el Metro de Santiago. Las estaciones combinan ruido ambiental, alta densidad de flujo peatonal, señalética variable y obstáculos dinámicos, factores que incrementan la carga cognitiva requerida para orientarse y elevan el riesgo de desorientación o accidentes. Aunque existen apoyos como señalización podotáctil o asistencia de terceros, su disponibilidad y efectividad no siempre son consistentes, y muchas soluciones tecnológicas dependen de infraestructura externa o condiciones controladas. En este escenario, el uso del teléfono móvil como plataforma de asistencia presenta una alternativa escalable, al aprovechar sensores y capacidad de cómputo en un dispositivo ya disponible para gran parte de la población.

A partir de este problema, en esta memoria se desarrolla AIMES, un sistema de asistencia orientado a personas con discapacidad visual que utiliza visión por computador y retroalimentación auditiva para entregar instrucciones de orientación a partir de la cámara del teléfono, sin requerir beacons ni equipamiento adicional en las estaciones. El trabajo incluye el diseño de una arquitectura modular que integra captura de video en tiempo real, detección de elementos del entorno y traducción de esas detecciones a mensajes claros para el usuario. El aporte hacia el proyecto se centra en la migración e implementación del sistema como aplicación móvil nativa, incorporando un pipeline en dispositivo que gestiona permisos y ciclo de vida de la cámara, ejecuta inferencia local, y produce salidas mediante texto-a-voz, además de componentes de control por comandos de voz y visualización de depuración para pruebas. Finalmente, se reportan pruebas funcionales y de desempeño orientadas a validar la ejecución en tiempo real en el teléfono, junto con limitaciones observadas y líneas de mejora para ampliar robustez y alcance del sistema en escenarios reales.

Palabras Clave: Discapacidad visual, visión por computador, Android, asistencia auditiva, Metro de Santiago, detección de objetos.

Abstract

In Chile, visual impairment is a significant limitation to independent travel, especially in highly complex public transportation environments such as the Santiago Metro. Stations combine environmental noise, high pedestrian traffic density, variable signage, and dynamic obstacles, factors that increase the cognitive load required for orientation and raise the risk of disorientation or accidents. Although there are aids such as tactile paving and third-party assistance, their availability and effectiveness are not always consistent, and many technological solutions depend on external infrastructure or controlled conditions. In this scenario, the use of mobile phones as an assistance platform presents a scalable alternative, taking advantage of sensors and computing power in a device that is already available to a large part of the population.

Based on this problem, this thesis develops AIMES, an assistance system aimed at people with visual impairments that uses computer vision and auditory feedback to provide orientation instructions from the phone's camera, without requiring beacons or additional equipment at stations. The work includes the design of a modular architecture that integrates real-time video capture, detection of elements in the environment, and translation of those detections into clear messages for the user. The main contribution of this thesis focuses on the migration and implementation of the system as a native mobile application, incorporating an on-device pipeline that manages camera permissions and lifecycle, performs local inference, and produces text-to-speech outputs, as well as voice command control components and debugging visualization for testing. Finally, functional and performance tests are reported to validate real-time execution on the phone, along with observed limitations and areas for improvement to expand the robustness and scope of the system in real-world scenarios.

Keywords: Visual impairment, computer vision, Android, auditory feedback, Santiago Metro, object detection.

Glosario

accesibilidad conjunto de condiciones que permiten que una persona pueda utilizar un entorno, servicio o sistema de manera autónoma, segura y comprensible, independiente de sus capacidades físicas, sensoriales o cognitivas.

aplicación nativa aplicación desarrollada directamente para un sistema operativo específico, como Android, utilizando sus lenguajes, librerías y APIs propias.

asistencia auditiva mecanismo de apoyo basado en la entrega de información mediante audio, utilizado en AIMES para comunicar instrucciones, confirmaciones y resultados de orientación.

asistencia visual tecnología o sistema orientado a apoyar a personas con discapacidad visual mediante captura, procesamiento e interpretación de información del entorno.

bounding box caja delimitadora generada por un modelo de detección para indicar la ubicación aproximada de un objeto dentro de una imagen.

dataset conjunto de datos utilizado para entrenar, validar o probar modelos de visión por computador. En AIMES corresponde principalmente a imágenes etiquetadas del Metro de Santiago.

detección de objetos tarea de visión por computador que permite identificar y localizar elementos dentro de una imagen o video mediante etiquetas y cajas delimitadoras.

edge computing enfoque de procesamiento en el cual los datos se analizan directamente en el dispositivo local, reduciendo dependencia de servidores externos o

conectividad.

frame imagen individual obtenida desde una secuencia de video, utilizada como entrada para el procesamiento visual del sistema.

inferencia proceso mediante el cual un modelo entrenado procesa una entrada, como una imagen o frame, para generar predicciones o detecciones.

orientación inmediata enfoque de asistencia utilizado en AIMES que entrega información breve sobre qué elemento se detecta, dónde se ubica y qué tan cerca se encuentra, sin implementar navegación completa puerta a puerta.

pipeline visual flujo de procesamiento encargado de capturar imágenes, preprocesarlas, ejecutar inferencia y postprocesar detecciones para obtener información útil del entorno.

piso podotáctil superficie con relieve utilizada como guía física para personas con discapacidad visual en espacios públicos, estaciones y recorridos peatonales.

postprocesamiento etapa posterior a la inferencia en la que se filtran, ordenan o ajustan las salidas del modelo para obtener detecciones finales utilizables por la aplicación.

preprocesamiento etapa previa a la inferencia donde una imagen se transforma, redimensiona o normaliza para cumplir con el formato de entrada requerido por el modelo.

retroalimentación auditiva respuesta entregada mediante audio para comunicar al usuario información sobre el estado del sistema, objetos detectados, posición relativa o instrucciones de orientación.

segmentación técnica de visión por computador que permite identificar regiones específicas de una imagen, generando máscaras asociadas a objetos o clases detectadas.

señalética conjunto de señales visuales utilizadas para orientar, informar o advertir dentro de un espacio físico, como estaciones del Metro de Santiago.

umbral de confianza valor mínimo utilizado para aceptar o descartar una predicción generada por un modelo de detección.

Acrónimos

AIMES Asistencia Inteligente en Metro de Santiago.

API Application Programming Interface.

APK Android Package Kit.

ASR Automatic Speech Recognition.

CPU Central Processing Unit.

CV Computer Vision.

DCNN Deep Convolutional Neural Network.

DETR DETection TRansformer.

DNN Deep Neural Network.

ENDIDE Encuesta Nacional de Discapacidad y Dependencia.

FPS Frames Per Second.

GPU Graphics Processing Unit.

GUI Graphical User Interface.

IDE Integrated Development Environment.

INE Instituto Nacional de Estadísticas.

IoT Internet of Things.

IoU Intersection over Union.

mAP mean Average Precision.

ML Machine Learning.

MVP Minimum Viable Product.

NMS Non-Maximum Suppression.

OCR Optical Character Recognition.

OMS Organización Mundial de la Salud.

PC Personal Computer.

PR Precision-Recall.

RAM Random Access Memory.

SAM Segment Anything Model.

SSD Single Shot MultiBox Detector.

STT Speech-to-Text.

TFLite TensorFlow Lite.

TTS Text-to-Speech.

UI User Interface.

UX User Experience.

ViT Vision Transformer.

WCAG Web Content Accessibility Guidelines.

YOLO You Only Look Once.

Tabla de Contenido

Glosario	v
Acrónimos	viii
1 Introducción	1
1.1 Situación Actual	1
1.2 Desafíos en Chile	3
1.3 Problemas de Accesibilidad en el Transporte Público	4
1.4 Definición del Enfoque de Solución	6
2 Estado del Arte	7
2.1 Contextual	7
2.2 Tecnológico	8
2.2.1 Tecnologías de Software	9
2.2.2 Trabajos relacionados y desarrollo de aplicaciones	15
3 Contexto de la Solución	21
3.1 Requisitos Funcionales	21
3.1.1 Captura de imágenes en tiempo real	22
3.1.2 Modelo de detección de señales y texto del Metro de Santiago	22
3.1.3 Reconocimiento de Voz	22
3.1.4 Orientación y retroalimentación mediante salida de voz	23
3.2 Requisitos no Funcionales	23
3.2.1 Compatibilidad	23
3.2.2 Portabilidad	24
3.2.3 Confiabilidad	24
3.2.4 Usabilidad	24
3.3 Requisitos de Interfaces	24
3.3.1 Cámara del dispositivo móvil	25

TABLA DE CONTENIDO

3.3.2	Aplicación móvil	25
3.3.3	Auriculares o salidas de audio	25
3.4	Requisitos del Entorno de Implementación y Uso	25
3.4.1	Condiciones acústicas	26
3.4.2	Condiciones visuales	26
3.4.3	Procesamiento rápido y baja latencia	26
3.4.4	Operación sin conexión	26
3.5	Requisitos de Desarrollo	27
3.6	Entorno de Soporte y Desarrollo	28
3.7	Perfiles de Usuario	29
4	Diseño de la Solución	34
4.1	Marco general	35
4.1.1	Visión general de la solución	36
4.1.2	Diagrama de Contexto	37
4.1.3	Diagrama de Arquitectura	38
4.1.4	Enumeración de Módulos	38
4.1.5	Descripción funcional de módulos	39
4.1.6	Matriz de Requisitos Funcionales y Módulos	50
4.1.7	Revisión de Arquitectura de Diseño	51
4.2	Evolución del prototipo y decisión de migración	52
4.2.1	Prototipo en computador: validación del pipeline visual con Python + OpenCV	55
4.2.2	Prototipo inicial con Kivy/Buildozer (Vosk + OpenCV + RH- Voice)	57
4.2.3	Migración a Android nativo con Jetpack Compose	59
4.3	Delimitación del alcance implementado	60
4.4	Diseño de interacción del sistema	61
4.4.1	Principios de diseño	62

TABLA DE CONTENIDO

4.4.2	Flujo general de interacción	63
4.4.3	Modelo conceptual de navegación e intención	64
4.4.4	Modelo real de estados de la aplicación implementada	66
4.5	Síntesis del diseño y transición a la implementación	67
5	Implementación de la Interfaz Nativa de AIMES en Android	69
5.1	Entorno tecnológico y estructura del proyecto	69
5.1.1	Dependencias principales	71
5.1.2	Estructura del proyecto	72
5.2	Arquitectura general de la aplicación	72
5.3	Pipeline visual: cámara, preprocesamiento e inferencia	76
5.4	Interpretación de resultados y orientación	80
5.5	Reconocimiento de texto y validación de destino	84
5.6	Interacción por voz y retroalimentación auditiva	85
5.6.1	Interacción por Voz	85
5.6.2	Retroalimentación Auditiva	86
5.7	Interfaz móvil	87
5.8	Manejo de permisos y ciclo de vida	96
5.8.1	Manejo de Permisos y recursos	96
5.8.2	Ciclo de vida de la Aplicación	97
5.9	Síntesis de la implementación	98
6	Resultados	100
6.1	Contexto General del Prototipo	100
6.1.1	Confección de Dataset	100
6.1.2	Entrenamiento de Modelos de Computer Vision	106
6.1.3	Desarrollo de Aplicación Móvil	107
6.2	Resultados	108
6.2.1	Dataset	108
6.2.2	Modelos de Computer Vision	110

ÍNDICE DE FIGURAS

6.2.3	Rendimiento de Modelos Obtenidos	114
6.2.4	Aplicación Móvil	115
6.3	Síntesis y discusión de resultados	118
7	Conclusiones	120
7.1	Conclusiones generales	120
7.2	Cumplimiento de objetivos	121
7.3	Limitaciones	122
7.4	Trabajo futuro	124
7.5	Cierre final	125

Referencias

Anexos

.1	Pauta de entrevista en terreno a Don Iván	
----	---	--

Índice de figuras

1.1	Entornos de conflicto inicial. Al momento de orientar una persona, se detectan obstáculos como mobiliario, otros usuarios de Metro y señalización no accesible.	3
4.1	Diagrama de contexto del sistema AIMES.	37
4.2	Diagrama de arquitectura del sistema AIMES.	38
4.3	Flujo general de interacción de AIMES basado en activación, análisis y retroalimentación auditiva.	63
4.4	Modelo conceptual: activación del asistente y entrada al modo de guía/asistencia.	64
4.5	Modelo conceptual: confirmación de intención y transición a ejecución.	65
4.6	Modelo conceptual: respuesta ante ausencia de elementos detectables o fallos de contexto, incluyendo mensajes de error.	65
4.7	Diagrama de estados de la aplicación AIMES implementada.	66

ÍNDICE DE FIGURAS

5.1	Arquitectura de código de la aplicación AIMES implementada en Android nativo.	73
5.2	Flujo de información entre los componentes internos de la aplicación AIMES.	75
5.3	Pipeline visual implementado en AIMES: captura, preprocesamiento, inferencia y postprocesamiento de detecciones.	76
5.4	Ejemplo ilustrativo de las etapas de preprocesamiento y postprocesamiento aplicadas en el pipeline visual de AIMES.	77
5.5	Flujo de interpretación espacial de detecciones para generación de orientación en AIMES.	80
5.6	Diseño conceptual de la interfaz móvil AIMES utilizado como referencia inicial.	88
5.7	Paleta de colores de la interfaz gráfica AIMES	89
5.8	Pantalla principal implementada en la aplicación móvil AIMES.	90
5.9	Pantalla de cámara y pruebas técnicas implementada en AIMES.	91
5.10	Funcionalidad de captura de imágenes en estado activo con visualización de métricas.	92
5.11	Pantalla de instrucciones implementada en AIMES.	93
5.12	Pantalla de configuración implementada en AIMES.	94
5.13	Opciones implementadas para el diseño del botón de alerta.	95
5.14	Pantalla asociada al botón de alerta implementada en AIMES.	95
6.1	Primer set de labels	105
6.2	Segundo set de labels	105
6.3	Tercer set de labels	106
6.4	Cuarto set de labels	106
6.5	Distribución de instancias etiquetadas para las clases presentes en el dataset de señales-detección.	108
6.6	Distribución de instancias etiquetadas para las clases presentes en el dataset de señales-segmentación.	109

ÍNDICE DE TABLAS

6.7	Comparación de modelos: parámetros, arquitecturas y resultados de mAP obtenidos.	110
6.8	Comparación de Confianza en los 3 modelos.	111
6.9	Comparación de Matriz de confusión en los 3 modelos.	111
6.10	Comparación de Precision-Recall en los 3 modelos.	111
6.11	Parámetros, arquitecturas y resultados de mAP obtenidos de YOLOv11n-Seg.	112
6.12	Curva de F1-score por <i>confidence threshold</i> del modelo YOLOv11n-Seg	112
6.13	Curva Precision–Recall (<i>PR</i>) del modelo YOLOv11n-Seg.	113
6.14	Matriz de confusión del modelo YOLOv11n-Seg.	113
6.15	Primer prototipo de la interfaz móvil desarrollado en Kivy, descartado en fase posterior.	116

Índice de tablas

1	Comparación de aplicaciones de asistencia visual	8
2	Tendencias tecnológicas en asistencia visual y relación con AIMES . . .	19
3	Novedades tecnológicas recientes aplicables a sistemas como AIMES .	20
4	Resumen de módulos del sistema AIMES.	39
5	Matriz de trazabilidad entre requisitos y módulos del sistema.	51
6	Limitaciones críticas observadas en el prototipo Kivy/Buildozer y su impacto.	59
7	Capacidades consideradas en el alcance implementado de AIMES. . . .	61
8	Stack tecnológico utilizado en la implementación móvil de AIMES. . .	70
9	Evolución del dataset y configuraciones de aumento.	102
10	Configuración de preprocesamiento y aumentos por versión del dataset.	103
11	Comparación de rendimiento y consumo de recursos de los modelos evaluados.	114

1. Introducción

1.1. Situación Actual

La discapacidad visual constituye una de las condiciones con mayor impacto en la autonomía personal y la participación en la vida urbana, particularmente en contextos donde el acceso a la información y la orientación dependen mayoritariamente de estímulos visuales. En ciudades densas y con sistemas de transporte complejos, como el Área Metropolitana de Santiago, esta condición se traduce en barreras persistentes para el desplazamiento independiente, el acceso a servicios y la integración social. En este contexto, resulta fundamental comprender la magnitud y características de la población con discapacidad visual, tanto a nivel nacional como internacional, como base para el diseño de soluciones tecnológicas orientadas a la accesibilidad y la inclusión.

En Chile, de acuerdo a la Encuesta Nacional de Discapacidad y Dependencia (EN-DIDE, 2022), organizada por el Observatorio Social del Ministerio de Desarrollo Social y Familia[1], se presentan los siguientes resultados respecto a personas que presentan alguna discapacidad visual:

- Se estima que aproximadamente 4,6 millones de adultos presentan algún grado de pérdida de visión; de ellos, cerca de 153 mil viven con ceguera total.
- La Región Metropolitana concentra el 22,7 % de los adultos con pérdida de visión y el 24,9 % de los adultos con ceguera en Chile.
- Cerca de 12 mil niños y adolescentes presentan casos de ceguera, mientras que 535 mil tienen ceguera total.

Esta información se complementa con reportes publicados por la Organización Mundial de la Salud (OMS, 2023)[2], que señalan:

- En el mundo hay al menos 2.200 millones de personas con deterioro de la visión

CAPÍTULO 1 : INTRODUCCIÓN

cercana o lejana. Como mínimo en mil millones de estos casos, la discapacidad visual podría haberse evitado o minimizado con el tratamiento adecuado.

- Las principales causas a nivel mundial de la discapacidad visual y la ceguera son los errores de refracción y las cataratas.
- Aunque la pérdida de visión puede afectar a personas de todas las edades, la mayoría de las personas con discapacidad visual y ceguera superan los 50 años.

El *World Report on Vision*, publicado por la OMS el año 2019[3], identifica desafíos estructurales persistentes en la atención y rehabilitación de la discapacidad visual a nivel global. Entre los principales se encuentran la cobertura insuficiente de servicios de salud visual, especialmente en etapas tempranas de detección, la fragmentación de la atención ocular dentro de los sistemas de salud, y la limitada disponibilidad de soluciones de rehabilitación funcional para personas con pérdida visual irreversible. Asimismo, el informe destaca brechas significativas en accesibilidad, tanto geográficas como socioeconómicas, que afectan de manera desproporcionada a personas mayores, mujeres y poblaciones vulnerables. A estos factores se suma el crecimiento sostenido de la población con discapacidad visual asociado al envejecimiento demográfico y a enfermedades crónicas, lo que tensiona aún más los sistemas de salud existentes y evidencia la necesidad de enfoques complementarios que integren tecnología, asistencia en tiempo real y modelos centrados en la persona.

Se adaptó este contexto global para identificar desafíos en distintos ambientes de Chile, como la infraestructura del transporte público, específicamente, Metro de Santiago. En este escenario, se hace necesario potenciar nuevas iniciativas que permitan mejorar la calidad de vida y la autonomía de las personas con discapacidad visual, especialmente en su movilidad y acceso a servicios esenciales.

1.2. Desafíos en Chile

El Metro de Santiago constituye uno de los sistemas de transporte público más relevantes del país, caracterizado por su alta concurrencia y dinamismo. Sin embargo, el entorno de cada estación puede variar significativamente, lo que dificulta una experiencia uniforme y segura para personas con discapacidad visual. La figura 1.1 ejemplifica esta variabilidad mediante dos fotografías de andenes del Metro de Santiago, en las que se identifican cuatro factores recurrentes de conflicto: la presencia de obstáculos físicos y mobiliario en el entorno, la ausencia de sistemas de ayuda en tiempo real (auditivos o visuales adaptados), el uso de señalización tradicional no accesible para personas ciegas, y la consecuente dependencia de terceros para orientarse o desplazarse con seguridad.

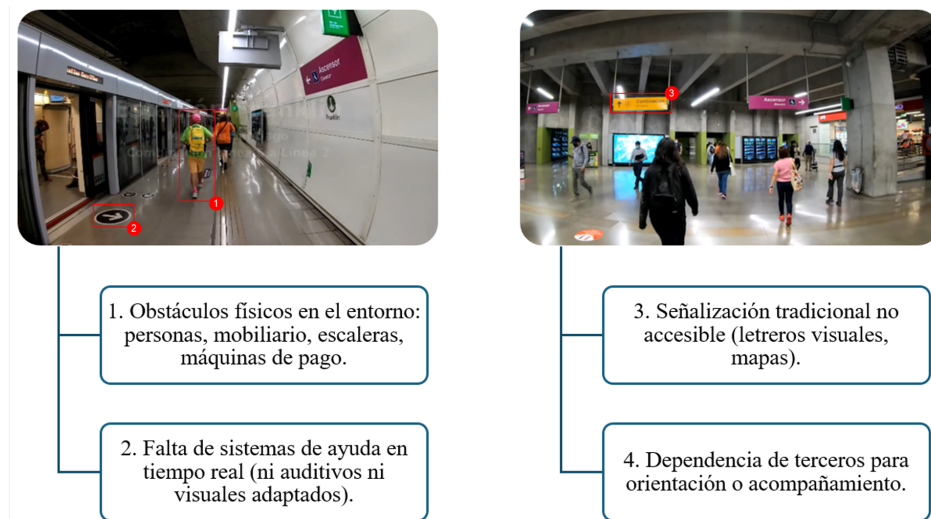


Figura 1.1: Entornos de conflicto inicial. Al momento de orientar una persona, se detectan obstáculos como mobiliario, otros usuarios de Metro y señalización no accesible.

Como problema central está la falta de autonomía y seguridad para personas con discapacidad visual en entornos de transporte urbano.

Factores de conflicto

- **Limitaciones de señalética tradicional:** La señalética visual no es interpreta-

da por personas ciegas o con baja visión, lo que les dificulta identificar salidas, destinos y obstáculos.

- **Entorno dinámico e impredecible:** Las estaciones tienen flujo de personas y cambios constantes (construcciones, bloqueos, nuevas rutas) que dificultan la planificación anticipada.
- **Soluciones en infraestructura limitadas:** Sistemas como pisos podotáctiles y señaléticas escritas en braille que indican caminos únicos hacia ascensores preferenciales y que son utilizados indebidamente. [4]
- **Soluciones tecnológicas generales:** Aplicaciones de asistencia existentes como Be My Eyes[5] y Seeing AI[6] no están integradas al entorno ni ofrecen respuesta contextualizada y en tiempo real. Por otro lado, la aplicación de Lazarillo[7] [8] sí cumple con estas características de integración, pero se limita a ser un mapa de navegación en estaciones puntuales.
- **Aumento del riesgo:** La imposibilidad de identificar obstáculos o rutas puede incrementar el riesgo percibido y la dependencia de terceros, lo que limita la autonomía.

1.3. Problemas de Accesibilidad en el Transporte Público

Las personas con discapacidad visual se enfrentan de manera cotidiana a múltiples barreras dentro del sistema de transporte público, las cuales no se limitan únicamente a la infraestructura física. La ausencia de información accesible en tiempo real, la señalización predominantemente visual y la alta variabilidad del entorno —como cambios de andén, interrupciones del servicio o flujos intensos de personas— generan escenarios de desorientación, inseguridad y sobrecarga cognitiva. Estas condiciones dificultan la movilidad autónoma y obligan, en muchos casos, a depender de terceros para realizar acciones básicas como identificar estaciones, orientarse dentro de recintos o confirmar decisiones de desplazamiento.

CAPÍTULO 1 : INTRODUCCIÓN

Si bien la ayuda de otras personas puede ser valiosa, esta no siempre está disponible ni se entrega de forma adecuada. Existen situaciones en las que las personas con discapacidad visual no logran solicitar asistencia, así como otras en las que la ayuda ofrecida resulta tardía, invasiva o poco clara, afectando la experiencia de viaje y la percepción de seguridad personal. Estas dinámicas han sido documentadas tanto por organizaciones como por testimonios directos de personas con discapacidad visual, quienes señalan que la falta de apoyo oportuno y respetuoso constituye una de las principales dificultades en el uso del transporte público.[9][10]

Como consecuencia de estas barreras persistentes, la frecuencia de uso del transporte público por parte de personas con discapacidad visual tiende a ser menor en comparación con la población general. Esta menor utilización no responde a una falta de interés, sino a la acumulación de experiencias negativas, riesgos percibidos y pérdida de autonomía. Si bien existen políticas públicas e iniciativas orientadas a la inclusión social y laboral de personas ciegas, estas se ven limitadas cuando las condiciones de movilidad urbana no garantizan un desplazamiento autónomo y seguro. En el largo plazo, esta situación contribuye a reforzar procesos de exclusión del espacio urbano, limitando el acceso efectivo a oportunidades laborales, educativas y sociales, y profundizando desigualdades estructurales en la participación plena de la vida urbana.[11]

1.4. Definición del Enfoque de Solución

La integración de tecnologías emergentes en la infraestructura de transporte puede ofrecer soluciones reales y efectivas para las personas con discapacidad visual. Es fundamental adoptar un enfoque accesible desde el diseño de los sistemas, priorizando la autonomía y dignidad de los usuarios.

Además, se hace indispensable establecer alianzas estratégicas con empresas tecnológicas y fundaciones especializadas en inclusión, con el objetivo de desarrollar herramientas que respondan directamente a las necesidades de este grupo de personas.

Elementos relevantes:

- **Usuario:** El usuario principal es la persona con discapacidad visual que necesita desplazarse de manera independiente por estaciones de metro. El sistema busca facilitar su movilidad brindando orientación en tiempo real mediante retroalimentación auditiva, identificación de señales y detección de obstáculos. Los usuarios secundarios incluyen familiares, cuidadores, personal del Metro y organizaciones de apoyo, que podrían configurar, supervisar o asistir en la implementación del sistema.
- **Producto:** El producto es un sistema de asistencia inteligente para personas con discapacidad visual que combina procesamiento digital de imágenes en tiempo real, detección de señales y obstáculos, retroalimentación auditiva mediante texto a voz (TTS) y recepción de comandos por voz (STT).
- **Mercado:** El mercado objetivo abarca soluciones de tecnología asistiva y accesibilidad urbana, orientadas a personas con discapacidad visual. Se considera la posible colaboración con actores clave como Metro de Santiago (por su interés en accesibilidad e inclusión), organizaciones gubernamentales y ONGs dedicadas a la inclusión de personas con discapacidad. Es importante destacar que el desarrollo de este proyecto es sin fines de lucro; por esta razón, se habla de una colaboración con actores clave y no de posibles clientes.

2. Estado del Arte

Esta sección considera dos aspectos importantes durante el desarrollo de AIMES: Contextual y Tecnológico. Ambos tienen el enfoque de establecer las bases con las que se desarrollan las funcionalidades, diseño de UX y PD, diseño de calidad de vida y de integración de tecnologías en el sistema.

2.1. Contextual

Si bien el desarrollo de este proyecto representa una oportunidad significativa para promover la autonomía de personas con discapacidad visual, también implica un desafío importante: la baja familiarización de este grupo con tecnologías móviles y digitales. Según ENDIDE 2022, solo el 1,1 % de las personas adultas ciegas y el 0,1 % de quienes tienen pérdida de visión utilizan herramientas tecnológicas de accesibilidad como lectores de pantalla o software especializado. Esta baja adopción tecnológica evidencia una brecha digital profunda, que podría afectar directamente la usabilidad y efectividad de soluciones tecnológicas si no se diseñan considerando sus habilidades, experiencias previas y contexto cotidiano.[12]

Por ello, este proyecto se plantea como una investigación exploratoria, que no solo busca implementar una solución, sino también comprender los hábitos, barreras y posibilidades reales de adopción tecnológica en esta población. No se pretende imponer una herramienta, sino co-diseñar una experiencia de asistencia accesible, gradual y ajustada a la realidad de los usuarios, con el fin de generar impacto social realista y sostenible.

El entorno objetivo para implementar la solución AIMES es el Metro de Santiago, por su arquitectura reconocible a gran escala. Sin embargo, como se aprecia en las imágenes de la Figura 1.1, la señalética, distribución y elementos de cada estación varían considerablemente entre líneas, lo que dificulta construir un dataset representativo de toda la red en el tiempo disponible para este trabajo. Por esta razón, se debe limitar el alcance de la infraestructura a alimentar en el dataset para el sistema, considerando las

líneas más recientes, como la línea 6 o la línea 3 para acotar el desarrollo y aprendizaje de AIMES, ya que son líneas que generalmente siguen el mismo patrón de espacios claros donde destacan los colores de las señales frente a los sobrios de las paredes, suelos, techos y ornamentas.

2.2. Tecnológico

Consideraciones deben realizarse respecto a qué hay presente en el mercado y cómo AIMES se acerca a estos productos. En la Tabla 1 se muestra una comparativa de distintas soluciones tecnológicas respecto a AIMES, considerando distintos atributos como el tipo de entrada y salida, si funcionan en tiempo real, qué ofrecen y si son de uso local o en línea.

Aplicación	Entrada	Salida	Tiempo real	Navegación	Offline
Seeing AI [6]	Imagen	Voz	No (foto)	No	No
Be My Eyes [5]	Video en vivo	Voluntario remoto	Sí	No	No
Aira [13]	Video en vivo	Agente humano	Sí (humano)	Sí (humano)	No
AIMES	Imagen + Video	Voz	Sí	Sí (Metro)	Sí

Tabla 1: Comparación de aplicaciones de asistencia visual

Por otro lado, respecto a las tendencias y novedades tecnológicas consideradas, se hizo revisión a distintos artículos que implementan innovaciones propias como arquitecturas conocidas que cumplan con la detección de objetos, complementado con retroalimentación auditiva.

Respecto a las tendencias, se aprecia una alta integración de tecnologías de Ultralytics YOLO y MobileNet [14], con resultados diversos dependiendo de la tarea asignada y los recursos disponibles.

Para contrastar con las tendencias, se realiza una revisión a distintas tecnologías emergentes haciendo un traceback desde el eslabón más antiguo hasta su innovación más reciente y de interés, así como su implementación en aplicaciones orientadas a

usuarios con discapacidad visual.

Esta revisión se separa en dos categorías:

2.2.1. Tecnologías de Software

Segment Anything Model (SAM): El modelo Segment Anything (SAM), propuesto por Meta AI[15][16], introdujo un paradigma generalista para la segmentación automática de objetos, permitiendo generar máscaras de alta calidad a partir de entradas simples como puntos, cajas o máscaras previas. A través de su arquitectura basada en Vision Transformers (ViT-H), SAM demostró una capacidad sin precedentes para segmentar objetos sin requerir entrenamiento específico por clase. A partir de la versión 8.0.89, el framework de Ultralytics incorporó soporte para SAM, permitiendo combinarlo fácilmente con tareas de detección en el ecosistema YOLOv8. Esta integración ofrece nuevas posibilidades para sistemas como AIMES, especialmente en tareas de anotación automatizada o análisis visual más detallado.

Avance a Faster SAM / MobileSAM: MobileSAM es una implementación eficiente del modelo original, basada en el trabajo *Faster Segment Anything* de Zhang et al. (2023)[17][18], que utiliza un codificador Tiny-ViT más liviano y una técnica de destilación desacoplada para preservar precisión con menor carga computacional. Esta versión permite realizar segmentación interactiva en tiempo real sobre dispositivos edge, alineándose con los requerimientos de AIMES para operar localmente sin depender de servidores remotos ni conectividad constante.

Lanzamiento de Ultralytics YOLOv1 (2015)[19]: La familia de modelos YOLO (You Only Look Once) marcó un hito en la detección de objetos en tiempo real desde su primera versión en 2015. Su enfoque unificado y de extremo a extremo permitió realizar detección con alta velocidad y precisión en una sola pasada por la red. Esta arquitectura ha servido como base para múltiples aplicaciones móviles y edge, lo que la

convierte en un referente fundamental para soluciones como AIMES, donde se requiere detección eficiente en tiempo real.

Lanzamientos de YOLOv11 y YOLOv12 (2024–2025)[20]: Las versiones más recientes de YOLOv11 y YOLOv12 han incorporado mecanismos de atención más refinados, fusión de características jerárquicas y mejoras en el entrenamiento semi-supervisado. Estas versiones apuntan a una mayor robustez frente a condiciones adversas y menor dependencia de grandes conjuntos de datos anotados, fortaleciendo su aplicabilidad en soluciones en tiempo real sobre plataformas embebidas.

DETR (2020): El modelo *DEtection TRansformer* (DETR)[21] introdujo por primera vez una arquitectura basada completamente en Transformers para la tarea de detección de objetos. A diferencia de los enfoques tradicionales basados en anchors, DETR reformula el problema como una tarea de predicción directa de objetos, lo que reduce la necesidad de postprocesamiento. Aunque su uso está más enfocado a dispositivos con alta capacidad de cómputo, su innovación ha inspirado múltiples variantes eficientes relevantes para sistemas como AIMES.

RepViT (2024): El artículo *RepViT: Revisiting Mobile CNN from ViT Perspective*[22] propone una nueva familia de arquitecturas de redes convolucionales eficientes para dispositivos móviles, inspiradas en el diseño de Transformers (ViT). Su principal contribución radica en combinar la eficiencia estructural de CNNs con principios de diseño de ViT, logrando superar a modelos como MobileViT y EfficientFormer tanto en precisión como en latencia. Es relevante para AIMES al presentar un enfoque moderno y liviano para tareas de visión por computador en tiempo real, especialmente útil en dispositivos edge.

MobileNet, CNN para smartphones (2017): MobileNet[23] introduce un diseño eficiente basado en convoluciones separables en profundidad que reduce significativa-

mente la complejidad computacional y el tamaño del modelo sin comprometer demasiado la precisión. Esta arquitectura está optimizada para correr en dispositivos con recursos limitados como smartphones, facilitando aplicaciones en tiempo real. Gracias a su bajo consumo de memoria y capacidad para mantener un rendimiento competitivo, MobileNet ha sido ampliamente adoptado en sistemas embebidos y móviles, sentando las bases para el desarrollo de redes ligeras aplicadas en sistemas de asistencia visual y otras aplicaciones móviles.

MobileViT, integración de transformer (2021): MobileViT[24] es una arquitectura híbrida que combina la eficiencia de las convoluciones de MobileNet con la capacidad de modelado global de los Transformers. A diferencia de modelos puramente convolucionales, MobileViT introduce bloques de atención que permiten capturar dependencias espaciales de largo alcance sin incrementar significativamente la complejidad computacional. Su diseño modular y compacto lo hace especialmente adecuado para tareas de visión por computador en tiempo real en dispositivos móviles. Esta arquitectura ofrece una buena relación entre precisión y eficiencia, lo que la convierte en una opción viable para reemplazar o complementar backbones convencionales en sistemas embebidos como AIMES, donde se requiere procesar simultáneamente detección, segmentación y OCR con bajo consumo de recursos.

Arquitectura DCNN Two Branches (2025): El enfoque propuesto por los autores en *Efficient Real-Time Pathfinding for Visually Impaired Individuals*[25] trata sobre una arquitectura de red neuronal convolucional profunda con dos ramas paralelas para segmentación semántica en tiempo real, diseñada específicamente para asistir a personas con discapacidad visual en la navegación. Una rama se enfoca en capturar detalles espaciales mediante canales anchos y capas superficiales (“demarcation path”), mientras que la otra extrae características semánticas profundas (“path with rich features”). Esta configuración permite balancear precisión y eficiencia computacional, alcanzando un 72,6 % de mIoU y una velocidad de 120 fotogramas por segundo (FPS) en GPU

CAPÍTULO 2 : ESTADO DEL ARTE

RTX 4090, favoreciendo la implementación en dispositivos portátiles con restricción de recursos. El modelo mejora la comprensión del entorno al identificar caminos y obstáculos, incrementando la autonomía y seguridad de los usuarios.

VOSK (2022): VOSK[26] es una librería de reconocimiento de voz de código abierto que permite realizar transcripción de voz a texto completamente offline. Está optimizada para dispositivos con recursos limitados, como teléfonos móviles, y soporta múltiples idiomas, incluyendo español. VOSK fue seleccionada para AIMES por su bajo consumo de recursos, velocidad de respuesta aceptable y capacidad de funcionar sin conexión, lo cual es clave en entornos como estaciones de Metro donde la conectividad puede ser variable.

RHVoice (2022): RHVoice[27] es un motor de síntesis de voz (Text-to-Speech) desarrollado bajo licencia libre, con soporte para varios idiomas y voces naturales, incluyendo opciones en español. Su rendimiento estable en entornos móviles y su posibilidad de operación completamente offline lo hacen especialmente apto para sistemas accesibles como AIMES, que requieren respuestas auditivas claras, empáticas y sin depender de la nube.

Kivy (2011): Kivy[28] es un framework de desarrollo multiplataforma en Python orientado a la creación de aplicaciones con interfaces gráficas y soporte táctil. Durante las primeras etapas de prototipado de AIMES, Kivy fue utilizado en conjunto con Buildozer para generar aplicaciones Android empaquetadas en formato APK. No obstante, su uso fue finalmente descartado debido a limitaciones en la integración con APIs nativas de Android, la falta de soporte avanzado para funcionalidades de accesibilidad del sistema operativo, el acceso restringido a servicios de audio y sensores en tiempo real, y el rendimiento subóptimo al trabajar con visión artificial y módulos de voz. Esto motivó la migración hacia desarrollo nativo con Android Studio, lo cual permitió mayor estabilidad, control de permisos, acceso directo a componentes del sistema y cumplimiento

estricto de estándares de accesibilidad móvil.

Android Studio (2013): Android Studio[29] es el entorno de desarrollo integrado (IDE) oficial para la creación de aplicaciones Android, basado en IntelliJ IDEA y mantenido por Google. Ofrece herramientas avanzadas para diseño de interfaces, depuración en tiempo real, emulación de dispositivos, integración con Gradle y optimización de rendimiento. En AIMES, Android Studio se utiliza como entorno central para empaquetar, compilar y desplegar la aplicación móvil, garantizando compatibilidad con versiones recientes de Android y acceso a APIs nativas necesarias para sensores, audio en tiempo real y permisos del sistema.

Kotlin (2011): Kotlin[30] es un lenguaje de programación moderno utilizado ampliamente en el desarrollo Android, caracterizado por su interoperabilidad con Java, sintaxis concisa y soporte para patrones de programación orientados a aplicaciones móviles. En AIMES, Kotlin se utiliza como lenguaje principal para estructurar los módulos de interfaz, procesamiento visual, lógica de asistencia e interacción por voz.

Jetpack Compose (2021): Jetpack Compose[31] corresponde al framework declarativo de interfaces de usuario para Android. Su enfoque permite construir pantallas a partir del estado de la aplicación, facilitando la creación de interfaces dinámicas y modulares. Para AIMES, esta tecnología resulta relevante porque permite implementar una interfaz simple, adaptable y coherente con una experiencia principalmente auditiva, además de facilitar la creación de pantallas de prueba para observar métricas, detecciones y estados internos del sistema.

CameraX (2019): CameraX[32] es una librería de Android orientada a simplificar el acceso a la cámara del dispositivo, manteniendo compatibilidad con el ciclo de vida de la aplicación. Esta tecnología permite configurar una vista previa de cámara y, al mismo tiempo, analizar frames en tiempo real mediante componentes como Preview

e **ImageAnalysis**. Su uso es especialmente relevante para AIMES, ya que el sistema requiere capturar el entorno de manera continua, estable y con baja latencia para alimentar el modelo de visión por computador.

TensorFlow Lite (2017): TensorFlow Lite[33] es una versión ligera de TensorFlow orientada a la ejecución de modelos de aprendizaje automático en dispositivos móviles y embebidos. Su objetivo es permitir inferencia local con menor consumo de recursos, reduciendo la dependencia de servidores externos o conectividad permanente. En el contexto de AIMES, TensorFlow Lite permite ejecutar el modelo de detección directamente en el teléfono, manteniendo el enfoque de procesamiento local necesario para una asistencia en tiempo real.

APIs nativas de voz de Android: Android dispone de componentes nativos para reconocimiento y síntesis de voz, entre ellos **SpeechRecognizer**[34] y **TextToSpeech**[35]. Estas APIs permiten capturar comandos hablados del usuario y generar respuestas auditivas desde la propia aplicación. Para AIMES, su integración resulta fundamental, ya que la interacción principal se basa en comandos de voz y retroalimentación auditiva, evitando depender de una interfaz visual tradicional.

WCAG (2018): La *Web Content Accessibility Guidelines*[36] (WCAG) es un conjunto de directrices que promueve la perceptibilidad, operabilidad, comprensibilidad y robustez de los contenidos digitales. Este enfoque busca asegurar una experiencia inclusiva para personas con discapacidad visual, especialmente aquellas con baja visión o daltonismo, al garantizar que los sitios web y aplicaciones sean accesibles a través de diversas tecnologías de asistencia, como lectores de pantalla, así como mediante el uso de contrastes adecuados, navegación mediante teclado y otros métodos de interacción inclusivos.

La revisión de arquitecturas y modelos recientes evidencia una clara tendencia hacia redes neuronales cada vez más eficientes y especializadas para dispositivos con recur-

limitados, como smartphones o dispositivos embebidos, sin sacrificar la precisión y velocidad requeridas en aplicaciones en tiempo real. Tecnologías como YOLO y sus evoluciones, junto con propuestas recientes como RepViT, MobileSAM y arquitecturas especializadas de segmentación, ofrecen bases sólidas para el desarrollo de sistemas que combinan detección y segmentación con retroalimentación auditiva. En particular, la implementación de YOLOv12 se presenta como una opción idónea para la detección rápida y precisa, mientras que modelos ligeros basados en RepViT o versiones optimizadas de arquitecturas dual-branch para segmentación permiten complementar la comprensión del entorno con un bajo costo computacional. El presente trabajo apunta a integrar estas innovaciones para construir una solución optimizada que permita una asistencia visual en tiempo real, accesible, robusta y práctica para personas con discapacidad visual, maximizando tanto la autonomía como la seguridad en entornos urbanos y complejos como el Metro de Santiago.

Por otra parte, las tecnologías enfocadas en el diseño de aplicaciones móviles son clave para la integración del modelo de CV en un software compactado y compatible con estándares de accesibilidad para personas con discapacidad visual. La correcta implementación de la interfaz, siguiendo los lineamientos de la WCAG, permitirá que las personas con discapacidad visual interactúen de manera efectiva y autónoma con la aplicación, garantizando una experiencia fluida y sin barreras, mejorando su accesibilidad a la información y servicios proporcionados por la aplicación móvil.

2.2.2. Trabajos relacionados y desarrollo de aplicaciones

Interpretación espacial y retroalimentación auditiva (2018) Diversos trabajos han abordado la asistencia a personas con discapacidad visual media. En el paper *Visually Impaired Users can Locate and Grasp Objects Under the Guidance of Computer Vision and Non-Visual Feedback*[37] proponen un sistema experimental de localización de objetos que utiliza una cámara monocular y algoritmos de visión para identificar la posición del objeto dentro del campo visual. Debido a que la cámara no entrega infor-

mación de profundidad tridimensional, el sistema se concentra en la posición bidimensional del objeto y la transforma en códigos espaciales discretos, tales como izquierda, centro, derecha, arriba o abajo. A partir de esta codificación, se genera retroalimentación auditiva o vibrotáctil para guiar al usuario hacia el objeto de interés.

De forma complementaria, los autores de *Visual Assistance for Blind Using Image Processing*[38] presentan un sistema de asistencia visual basado en procesamiento de imágenes, comandos de voz y salida auditiva. En dicho trabajo, la cámara captura video, este se divide en frames, se aplican algoritmos de detección de objetos o señalética, y posteriormente se entrega al usuario una indicación auditiva sobre la ubicación del objeto, utilizando instrucciones como izquierda, derecha o centro. Esta arquitectura valida el uso de un ciclo compuesto por comando de voz, captura visual, detección, estimación de posición y retroalimentación auditiva.

Conformación de datasets para asistencia visual (2019) La conformación de un dataset representativo constituye una etapa crítica en sistemas de asistencia visual basados en visión por computador. En *Object Detection and Narrator for Visually Impaired People*[39] proponen un sistema de detección y narración de objetos para personas con discapacidad visual, utilizando modelos de aprendizaje automático para identificar elementos del entorno y comunicar los resultados mediante audio. En dicho trabajo, los autores destacan que la disponibilidad de grandes volúmenes de datos ha sido un factor clave para el avance de los modelos de detección de objetos, y utilizan modelos entrenados sobre conjuntos de datos generales para reconocer objetos cotidianos.

Evaluación de modelos de Deep Learning en smartphones (2021) El artículo *Smart at What Cost? Characterizing Mobile Deep Learning Inference Performance*[40] presenta un estudio holístico sobre el uso y desempeño de modelos de redes neuronales profundas (DNNs) en aplicaciones móviles del mundo real. A través del análisis de más de 16 mil aplicaciones populares en la Google Play Store, los autores caracterizan las arquitecturas, optimizaciones y cargas computacionales asociadas a los modelos DNN

utilizados en dispositivos móviles de distintas gamas y generaciones. Para ello, desarrollaron *gaugeNN*, una herramienta que automatiza la extracción, medición y evaluación del rendimiento de estos modelos. El estudio revela una creciente penetración de DNNs en el ecosistema móvil, así como importantes desafíos en términos de latencia, consumo energético y compatibilidad con hardware heterogéneo. Además, se destacan las brechas entre las técnicas avanzadas propuestas en la literatura y su adopción práctica, enfatizando la necesidad de herramientas de despliegue optimizadas para un entorno tan dinámico y limitado en recursos como el móvil.

Implementación de YOLO para ejecución remota (2023-2024) Los autores de *Implementing Visual Assistant Using YOLO and SSD for Visually-Impaired Persons*[41] presentan una arquitectura de aplicación en la que los modelos de detección de objetos, específicamente YOLO y SSD, se ejecutan de manera remota en un servidor, mientras que el dispositivo del usuario (por ejemplo, un smartphone) actúa como cliente ligero. Esta solución permite superar las limitaciones de procesamiento local en dispositivos móviles al enviar las imágenes capturadas hacia el servidor, realizar allí la inferencia, y retornar los resultados al dispositivo, donde se entrega retroalimentación auditiva al usuario. Aunque este enfoque impone dependencias de conectividad y latencia de red, demuestra ser una opción viable para desplegar modelos complejos de visión por computador en tiempo real para personas con discapacidad visual, sin requerir hardware de alto rendimiento en el dispositivo cliente. Sin embargo, las limitaciones están asociadas a la latencia adquirida por la carga y subida de datos, como el riesgo de falta de conectividad por la movilidad del dispositivo.

Por otro lado, los autores de *Mobile App for Enhancing Accessibility Among the Visually Impaired*[42] proponen una aplicación móvil orientada a personas con discapacidad visual que integra detección de objetos en tiempo real, extracción de texto mediante OCR, detección de divisa y retroalimentación auditiva. El trabajo identifica limitaciones recurrentes en aplicaciones de asistencia visual, tales como problemas de precisión, alta latencia y dificultades de uso, aspectos que también son relevantes para

CAPÍTULO 2 : ESTADO DEL ARTE

el diseño de AIMES. Su propuesta utiliza modelos de visión por computador, OCR y síntesis de voz para entregar información del entorno al usuario, reforzando la tendencia hacia soluciones móviles multimodales basadas en cámara y audio.

En conjunto, los trabajos revisados evidencian una evolución desde sistemas de asistencia basados principalmente en la detección de objetos y la generación de instrucciones espaciales hacia arquitecturas multimodales capaces de integrar visión por computador, reconocimiento de texto, comandos de voz y síntesis de voz. Asimismo, se observa una presencia recurrente de algoritmos de detección orientados al procesamiento en tiempo real, particularmente arquitecturas de la familia YOLO, debido a su capacidad para realizar inferencia sobre secuencias de video con baja latencia. Estos antecedentes respaldan el desarrollo de AIMES como una solución multimodal que integre detección y comprensión visual, procesamiento local e interacción mediante voz, priorizando la baja latencia, la autonomía operativa y la reducción de la dependencia de conectividad en entornos urbanos complejos.

En las Tablas 2 y 3 se exponen resúmenes de las tecnologías destacables y de interés para este proyecto. La columna *Implementación en AIMES* de la Tabla 2 clasifica cada tecnología como **obligatoria**, cuando corresponde a un componente que se utilizará sí o sí para el funcionamiento del sistema, o como **deseable**, cuando se trata de una funcionalidad que podría enriquecer la aplicación pero no es indispensable para su versión actual.

Tendencia	Descripción	Implementación en AIMES
Modelos ligeros de CV (MobileNet, RepViT)	Redes convolucionales optimizadas para dispositivos móviles con baja latencia y alta precisión.	Obligatorio
OCR en tiempo real	Reconocimiento de texto en ambientes dinámicos como estaciones o paneles de trenes.	Obligatorio
Fusión multimodal	Combinación de visión, texto y voz para una experiencia más robusta.	Obligatorio
Interacción por voz personalizada	Asistentes con prompts adaptados al contexto y preferencias del usuario.	Obligatorio
Edge Computing	Procesamiento directamente en el dispositivo para evitar depender de internet.	Obligatorio
Mapeo semántico del entorno	Reconocimiento contextual de objetos y señales para entender el entorno.	Deseable
Framework multiplataforma (Kivy) [28]	Desarrollo de aplicaciones móviles con GUI en Python, adaptable a dispositivos Android.	Deseable
Framework Android Studio	Desarrollo de aplicaciones móviles con GUI en Kotlin. APK inmediata desde su compilador.	Obligatorio
Kotlin + Jetpack Compose	Desarrollo nativo de interfaz móvil declarativa, modular y basada en estado.	Obligatorio
CameraX	Captura de cámara y análisis de frames en tiempo real integrado al ciclo de vida Android.	Obligatorio
TensorFlow Lite	Ejecución local de modelos de visión artificial en dispositivos móviles.	Obligatorio
APIs nativas STT/TTS Android	Reconocimiento de comandos de voz y generación de retroalimentación auditiva.	Obligatorio

Tabla 2: Tendencias tecnológicas en asistencia visual y relación con AIMES

Innovación	Descripción
MobileViT[24] (2021)	Híbrido entre MobileNet y ViT, combina eficiencia estructural de CNNs con el modelado global de Transformers.
RepViT[22] (CVPR 2024)	CNNs móviles inspiradas en ViT que superan a MobileViT y EfficientFormer en precisión y latencia para edge computing.
MobileSAM[17] (2023)	Versión ligera de SAM basada en Tiny-ViT, permite segmentación eficiente en dispositivos móviles.
YOLOv12[20][43] (2025)	Versión más reciente y robusta de la familia YOLO, con mecanismos de atención jerárquicos y optimización para ejecución en edge.
Arquitectura DCNN Two-Branches[25] (2025)	Red especializada con dos rutas paralelas para segmentación en tiempo real, pensada para asistencia a personas con discapacidad visual.
VOSK[26] (2022)	Motor de reconocimiento de voz offline, eficiente en móviles, con soporte multilingüe.
RHVoice[27] (2022)	Motor de texto a voz de código abierto con voces naturales y operación completamente offline.

***Tabla 3:** Novedades tecnológicas recientes aplicables a sistemas como AI-MES*

3. Contexto de la Solución

La presente sección describe el contexto técnico y operativo de la solución propuesta por AIMES, estableciendo los requisitos que guían su diseño, implementación y uso en un entorno real. Se definen:

- **Requisitos Funcionales.** Cuales especifican las capacidades esenciales del sistema para percibir el entorno, interactuar con el usuario y entregar asistencia en tiempo real.
- **Requisitos No Funcionales.** Orientados a garantizar atributos de calidad como usabilidad, confiabilidad y portabilidad.
- **Requisitos de Interfaces.** Describen los medios físicos y digitales a través de los cuales el usuario interactúa con la solución.
- **Requisitos del Entorno de Implementación y Uso.** Considerando las condiciones propias del transporte público metropolitano, tales como ruido ambiental, variabilidad del entorno y conectividad limitada.
- **Requisitos de Desarrollo.** Cuales delimitan las herramientas de hardware y software necesarias para construir, probar y desplegar la solución de manera consistente con los objetivos de accesibilidad, rendimiento y escalabilidad del proyecto.

3.1. Requisitos Funcionales

Los requisitos funcionales definen las capacidades esenciales que debe cumplir AIMES para asistir de manera efectiva a personas con discapacidad visual durante su desplazamiento en el transporte público. Estos requisitos describen las funciones mínimas necesarias para percibir el entorno, interpretar información relevante y comunicarla al usuario de forma accesible y oportuna, permitiendo una interacción activa entre el sistema y la persona usuaria.

3.1.1. Captura de imágenes en tiempo real

El sistema debe permitir y hacer uso de una cámara, ya sea integrada en un teléfono móvil o una cámara externa compatible, además de manipular la captura de imágenes del entorno de forma continua y en tiempo real.

Se espera que la captura sea estable y frecuente, permitiendo el análisis dinámico del entorno, considerando variaciones de iluminación, movimiento del usuario y cambios en la escena propios del desplazamiento dentro de las estaciones del Metro de Santiago.

3.1.2. Modelo de detección de señales y texto del Metro de Santiago

AIMES debe integrar modelos de *Computer Vision* especializados en la detección de señales visuales relevantes del entorno del Metro de Santiago, tales como letreros direccionales, numeración de andenes, nombres de estaciones, señalética de seguridad y pantallas informativas.

Adicionalmente, el sistema deberá incorporar técnicas de reconocimiento óptico de caracteres (OCR) para interpretar el texto presente en dichas señales direccionales, las pantallas del estado de salud del metro y simbología para información de rápido acceso.

Con ambas características, se permite transformar la información visual estructurada y no estructurada en datos semánticos utilizables por el sistema.

3.1.3. Reconocimiento de Voz

El sistema deberá ser capaz de recibir instrucciones verbales por parte del usuario mediante el reconocimiento de voz, permitiendo una interacción natural y manos libres. Estas instrucciones podrán incluir solicitudes de orientación, confirmaciones, repetición de información o consultas contextuales.

El sistema relacionará dichas instrucciones con la información que haya sido recolectada a través de la captura de imágenes. Así, en conjunto, le ofrece al usuario

CAPÍTULO 3 : CONTEXTO DE LA SOLUCIÓN

distintas respuestas, desde la orientación para transitar dentro de la estación hasta la ruta que debe seguir dentro de la red del metro.

3.1.4. Orientación y retroalimentación mediante salida de voz

AIMES deberá proporcionar información al usuario a través de audio en tiempo real, utilizando síntesis de voz clara y comprensible. Esta información incluirá:

- Indicaciones de orientación espacial
- Descripción de elementos relevantes del entorno
- Alertas de seguridad
- Confirmaciones de acciones

La comunicación deberá ser oportuna y contextual, evitando la sobrecarga cognitiva del usuario y priorizando la información crítica durante el desplazamiento.

3.2. Requisitos no Funcionales

Los requisitos no funcionales establecen atributos de calidad que condicionan el desempeño global de AIMES, más allá de las funciones específicas que el sistema ofrece. Estos requisitos son especialmente relevantes en un contexto de uso crítico como el transporte público, donde aspectos como la confiabilidad, usabilidad y compatibilidad técnica impactan directamente en la seguridad, confianza y experiencia de la persona usuaria.

3.2.1. Compatibilidad

El sistema debe asegurar la correcta interoperabilidad entre sus distintos componentes, incluyendo módulos de captura de imágenes, procesamiento visual, reconocimiento de voz, síntesis de audio y la aplicación móvil. Esta compatibilidad debe permitir un

CAPÍTULO 3 : CONTEXTO DE LA SOLUCIÓN

flujo continuo de información entre los módulos, evitando pérdidas de datos o inconsistencias que puedan afectar la entrega de asistencia al usuario.

3.2.2. Portabilidad

AIMES debe ser una solución portable y accesible, implementada principalmente como una aplicación móvil desarrollada para el sistema operativo Android. Esta portabilidad permite que el sistema sea utilizado en dispositivos comúnmente disponibles, sin requerir hardware especializado adicional, facilitando su adopción por parte de los usuarios finales.

3.2.3. Confiabilidad

El sistema debe operar de manera continua y estable durante toda la experiencia de desplazamiento del usuario dentro del Metro de Santiago. Esto implica mantener un funcionamiento consistente de los módulos de percepción, interpretación y comunicación, minimizando interrupciones o fallos que puedan generar desorientación, pérdida de información o situaciones de riesgo.

3.2.4. Usabilidad

AIMES debe presentar una alta usabilidad, considerando que sus usuarios poseen discapacidad visual. La interacción debe realizarse principalmente mediante comandos de voz claros y concisos, junto con una retroalimentación auditiva comprensible y estructurada. El diseño de la interacción debe minimizar la carga cognitiva del usuario y facilitar un aprendizaje rápido del sistema.

3.3. Requisitos de Interfaces

Los requisitos de interfaces describen los medios físicos y digitales a través de los cuales el usuario interactúa con AIMES. Dado que el sistema está orientado a personas

CAPÍTULO 3 : CONTEXTO DE LA SOLUCIÓN

con discapacidad visual, estas interfaces deben priorizar la accesibilidad, reducir la dependencia de estímulos visuales y permitir una interacción natural basada en audio y comandos de voz.

3.3.1. Cámara del dispositivo móvil

El sistema debe hacer uso de la cámara del dispositivo móvil para capturar imágenes del entorno del usuario. Esta interfaz constituye el principal medio de percepción visual del sistema, permitiendo la adquisición de información necesaria para el procesamiento mediante modelos de visión artificial y reconocimiento de texto.

3.3.2. Aplicación móvil

AIMES debe contar con una aplicación móvil que centralice la interacción con el usuario. Esta aplicación permitirá la configuración básica del sistema, la recepción de comandos de voz, la ejecución de los módulos de procesamiento y la entrega de asistencia auditiva en tiempo real. La interfaz debe estar diseñada para operar sin requerir interacción visual directa.

3.3.3. Auriculares o salidas de audio

El sistema debe utilizar auriculares o las salidas de audio del dispositivo móvil para emitir instrucciones de navegación, descripciones del entorno y alertas relevantes. Esta interfaz es clave para garantizar una comunicación clara y privada, especialmente en entornos ruidosos como estaciones y vagones del Metro de Santiago.

3.4. Requisitos del Entorno de Implementación y Uso

AIMES está concebido para operar en un entorno urbano complejo y altamente dinámico, como el Metro de Santiago. En consecuencia, resulta necesario considerar

CAPÍTULO 3 : CONTEXTO DE LA SOLUCIÓN

explícitamente las condiciones del entorno de implementación y uso, las cuales influyen directamente en el desempeño del sistema y en la experiencia del usuario durante situaciones reales de desplazamiento.

3.4.1. Condiciones acústicas

El sistema debe considerar entornos con altos niveles de ruido ambiental. Por ello, las funciones de salida de audio deben priorizar mensajes breves, estructurados y oportunos, permitiendo que la información crítica sea percibida claramente sin sobrecargar al usuario.

3.4.2. Condiciones visuales

AIMES debe ser robusto frente a variaciones en las condiciones visuales del entorno, tales como cambios de iluminación, presencia de sombras, reflejos en superficies y oclusiones parciales. Esta robustez es fundamental para mantener un desempeño consistente de los modelos de visión artificial en escenarios reales.

3.4.3. Procesamiento rápido y baja latencia

El sistema debe procesar la información capturada y entregar la asistencia al usuario con latencias reducidas. La rapidez en el procesamiento es esencial para que las indicaciones proporcionadas sean relevantes y útiles en el momento en que el usuario debe tomar decisiones durante su desplazamiento.

3.4.4. Operación sin conexión

AIMES debe ser capaz de operar de manera local, sin depender exclusivamente de una conexión a internet. Esta característica permite asegurar la continuidad del servicio en estaciones o tramos donde la conectividad es limitada o inexistente, garantizando una asistencia constante al usuario.

3.5. Requisitos de Desarrollo

Los requisitos de desarrollo delimitan los recursos técnicos necesarios para diseñar, implementar y validar la solución propuesta. Estos requisitos reflejan las decisiones adoptadas durante el desarrollo del prototipo de AIMES, considerando herramientas accesibles, entornos ampliamente soportados y tecnologías compatibles con el despliegue en dispositivos móviles Android.

Hardware de Desarrollo

1. Computador: Para el diseño, codificación y entrenamiento de modelos de visión artificial y reconocimiento de voz a texto.
2. Cámara: Para simular la entrada de video en pruebas locales del sistema.
3. Auriculares o parlantes externos: Para probar la retroalimentación auditiva (text-to-speech).

Estos elementos permiten simular condiciones de uso reales durante las etapas de desarrollo y prueba, facilitando la evaluación temprana de la captura de información visual y la entrega de retroalimentación auditiva antes de su despliegue en dispositivos móviles.

Software de Desarrollo

1. Entrenamiento de modelo de detección de señales y texto: OpenCV y Roboflow.
2. Python: Consideración del framework Kivy [28] y Ultralytics [44] para el desarrollo inicial de la interfaz móvil y modelos de CV en lenguaje Python.
3. Android Studio: Utilización de Android Studio para el diseño, compilación, depuración y generación del ejecutable de la aplicación en dispositivos Android.
4. Kotlin y Jetpack Compose: Utilización de Kotlin como lenguaje principal de desarrollo nativo y Jetpack Compose para la construcción de la interfaz móvil, navegación entre pantallas y gestión de estados visuales.

5. CameraX: Librería utilizada para acceder a la cámara del dispositivo móvil, mostrar la vista previa y analizar frames en tiempo real dentro de la aplicación Android.
6. TensorFlow Lite: Motor utilizado para ejecutar localmente el modelo de detección en formato `.tflite` dentro del dispositivo móvil.
7. Para la asistencia por voz: En el prototipo inicial se consideraron librerías compatibles con Kivy, potencialmente Plyer [45]; sin embargo, en la implementación final se utilizaron APIs nativas de Android, específicamente `SpeechRecognizer` para reconocimiento de voz y `TextToSpeech` para retroalimentación auditiva.
8. Sistema Operativo: Windows, para desarrollar la solución. Android, como plataforma objetivo para la migración e implementación final de la solución móvil.

La selección de estas herramientas responde a la necesidad de integrar modelos de visión artificial, procesamiento de audio y desarrollo móvil en un entorno coherente. Durante el desarrollo se evaluaron distintas alternativas, optándose finalmente por aquellas que ofrecen mayor control, estabilidad y compatibilidad con las capacidades nativas de Android, lo que favorece la escalabilidad y mantenibilidad del sistema.

3.6. Entorno de Soporte y Desarrollo

El entorno de soporte y desarrollo de AIMES combina hardware accesible y software de código abierto con herramientas de despliegue nativo en dispositivos móviles. El diseño, entrenamiento de modelos y pruebas iniciales se realizan en un computador de escritorio, mientras que los emuladores de dispositivos Android permiten validar las interacciones y latencias en condiciones similares al uso real.

En cuanto a la construcción de los dataset utilizados para el entrenamiento de los modelos de visión por computador, se emplea **Roboflow** como plataforma de etiquetado, versionado y gestión del dataset. Esta herramienta permite anotar las imágenes

capturadas en el Metro de Santiago, aplicar transformaciones de preprocesamiento y aumento de datos, y exportar los conjuntos resultantes en formatos compatibles con los modelos YOLO utilizados por el sistema. La construcción de estos datasets corresponde a un trabajo colectivo dentro del proyecto AIMES, por lo que esta memoria no tiene como objetivo detallar exhaustivamente dicho proceso, sino utilizarlo como recurso técnico sobre el cual se reportan los resultados obtenidos en el Capítulo 6.

En la etapa de prototipado temprano, se evaluó el uso de Kivy y su empaquetado mediante Buildozer para la generación de APKs. Sin embargo, esta alternativa fue descartada debido a limitaciones en el acceso a APIs nativas de Android, la falta de soporte robusto para accesibilidad a nivel de sistema y la necesidad de integrar módulos de audio, visión y permisos con baja latencia.

En consecuencia, la arquitectura móvil de AIMES pasó a desarrollarse de forma nativa en Android Studio, lo que permite un mayor control sobre la interfaz de usuario, uso de componentes accesibles del sistema operativo, integración directa con motores STT/TTS, sensores y librerías de visión. Este ecosistema asegura una transición fluida desde prototipos funcionales hacia una aplicación móvil estable, escalable y alineada con las buenas prácticas de desarrollo Android.

Este enfoque de desarrollo permite que las decisiones técnicas adoptadas respondan directamente a las condiciones reales de uso del sistema en el Metro de Santiago, asegurando coherencia entre el entorno de desarrollo, las capacidades de la plataforma móvil y las necesidades del usuario final.

3.7. Perfiles de Usuario

Esta sección presenta los distintos perfiles de usuario identificados para el desarrollo de AIMES, considerando las características, necesidades y contextos de las personas con discapacidad visual que podrían utilizar la aplicación. Los rangos etarios fueron determinados en base a la información obtenida del **Estudio de Personas con Discapacidad Visual en Chile según ENDIDE 2022** [12]. Estos antecedentes indican que

CAPÍTULO 3 : CONTEXTO DE LA SOLUCIÓN

la población con discapacidad visual se concentra principalmente en adultos y adultos mayores, aunque también se consideraron jóvenes adultos, quienes muestran mayor familiaridad con el uso de dispositivos móviles.

Joven con Discapacidad Visual

- **Socioeconómico y cultural:** Nivel socioeconómico diverso, pero prevalecen medio o medio bajo; generalmente con acceso a educación superior o media completa. [46]
- **Ocupacional:** Estudiante universitario o trabajador en formación.
- **Etario:** 18–29 años.
- **Características físicas, fisiológicas, psicológicas:** Se presume una buena salud física. Independiente en su vida diaria. Tiene autonomía en toma de decisiones. No posee afecciones mentales que afecten su independencia.
- **Otros:** Busca independencia total en el desplazamiento. Se infiere que utiliza smartphone de forma habitual y podría estar familiarizado con aplicaciones de apoyo similares.

Adulto con Discapacidad Visual

- **Socioeconómico y cultural:** Nivel socioeconómico diverso. Puede tener escolaridad media o técnica.[46]
- **Ocupacional:** Trabajador formal/informal o cesante.
- **Etario:** 30–59 años.
- **Características físicas, fisiológicas, psicológicas:** Capacidad física variable y sin afecciones mentales que puedan perjudicar su autonomía. Puede presentar condiciones asociadas como glaucoma.

CAPÍTULO 3 : CONTEXTO DE LA SOLUCIÓN

- **Otros:** Uso diario u ocasional de ayudas técnicas. Se presume que al menos está familiarizado con las nuevas tecnologías.

Persona Mayor con Discapacidad Visual

- **Socioeconómico y cultural:** Nivel socioeconómico bajo a medio. Poco familiarizado con la tecnología. Riesgo de exclusión digital.[46]
- **Ocupacional:** Jubilado o dependiente de apoyo familiar o estatal.
- **Etario:** 60 años o más.
- **Características físicas, fisiológicas, psicológicas:** Puede tener limitaciones físicas asociadas a la edad. Bajo manejo de tecnología. Posible comorbilidad (existencia de dos o más enfermedades).
- **Otros:** Requiere sistema simplificado. Mayor necesidad de soporte o asistencia con la tecnología.

Cuidador o Familiar

- **Socioeconómico y cultural:** Nivel socioeconómico diverso. Generalmente de entorno directo del usuario final. Compromiso afectivo.[46]
- **Ocupacional:** Familiar, acompañante o cuidador contratado.
- **Etario:** 20–65 años.
- **Características físicas, fisiológicas, psicológicas:** Buen estado físico y mental. Capacidad para asistir y supervisar.
- **Otros:** Rol secundario pero clave en configuración y soporte del sistema.

Asistente o Trabajador del Metro

- **Socioeconómico y cultural:** Empleado del sistema de transporte público. Capacitado en atención a usuarios.[46]

- **Ocupacional:** Personal de atención al pasajero.
- **Etario:** 25–60 años.
- **Características físicas, fisiológicas, psicológicas:** Buen estado físico. Formación en atención al público. Capacidad para resolver situaciones de emergencia.
- **Otros:** Puede actuar como punto de apoyo ocasional. No opera directamente el sistema.

Se han identificado dos categorías de usuarios: principales, que incluyen jóvenes, adultos y adultos mayores con discapacidad visual; y secundarios, conformados por cuidadores, familiares y asistentes del metro. Sobre la base de esta clasificación, se ha determinado que el enfoque principal será el adulto con discapacidad visual, dado que la prevalencia de esta condición es significativamente mayor en dicho grupo etario.[1], [12]

Revisión de Requerimientos Los requerimientos funcionales fueron revisados para asegurar que cubran las características mínimas necesarias para el MVP, permitiendo pruebas en terreno y la validación inicial del sistema con usuarios reales. Se confirmó que estos requisitos permiten generar una versión demostrable y operativa de AIMES, sobre la cual será posible escalar módulos adicionales de accesibilidad, navegación y personalización en fases posteriores.

En cuanto a los requerimientos no funcionales y de interfaces, la revisión verificó que estos aportan estructura y consistencia al proyecto, al orientar su implementación hacia estándares de accesibilidad, tiempos de respuesta definidos, operación offline y compatibilidad con dispositivos Android de gama media. Esta evaluación permitió asegurar coherencia entre las distintas categorías de requisitos y su alineación con los objetivos de inclusión, rendimiento y usabilidad del proyecto, considerando explícitamente las restricciones del entorno urbano y del transporte público.

Respecto a los requisitos de desarrollo, se re-evaluaron ciertas librerías y frameworks inicialmente considerados en el prototipo, como Kivy y sus módulos de audio,

CAPÍTULO 3 : CONTEXTO DE LA SOLUCIÓN

los cuales presentaron incompatibilidades con APIs nativas de Android y limitaciones en la integración con servicios de accesibilidad. Como resultado, se decidió migrar hacia un desarrollo nativo en Android Studio, lo que habilita el uso directo de motores de STT/TTS del sistema, acceso a sensores, permisos avanzados y componentes de interfaz accesibles. Esta transición asegura mayor estabilidad técnica, soporte a largo plazo y cumplimiento de las guías de accesibilidad oficiales de Android y WCAG.

4. Diseño de la Solución

En este capítulo se describe en detalle el proceso de desarrollo que permitió materializar AIMES como una aplicación móvil funcional y operable en condiciones reales. El trabajo se enfocó en transformar un prototipo conceptual en un sistema ejecutable en Android, capaz de capturar video en tiempo real, ejecutar inferencia local sobre los cuadros obtenidos y entregar orientación mediante retroalimentación auditiva clara y oportuna para el usuario.

El desarrollo se organizó de manera incremental, siguiendo un ciclo iterativo de diseño, implementación y validación en el dispositivo, lo que permitió refinar tanto la arquitectura del sistema como las decisiones tecnológicas adoptadas. Se priorizaron decisiones de diseño orientadas a la robustez frente a fallos, la modularidad de los componentes y la observabilidad del comportamiento interno del sistema durante las pruebas. También se prestó especial atención al registro de métricas de desempeño (FPS, confianza promedio y herramientas de depuración visual), que facilitarían la detección y corrección de errores en el terreno.

Asimismo, se incorporaron de manera sistemática las lecciones obtenidas de enfoques previos de prototipado que no resultaron adecuados para la operación y desarrollo de la aplicación en un contexto real. Estos prototipos iniciales, si bien útiles para explorar la viabilidad de los modelos de visión por computadora, presentaron limitaciones importantes en términos de estabilidad, experiencia del usuario y facilidad de depuración. A partir de estos aprendizajes, se redefinió la arquitectura hacia un diseño más desacoplado, con interfaces bien definidas entre los módulos de captura, procesamiento e interacción con el usuario, lo que facilitó tanto la evolución del sistema como su mantenimiento.

En las siguientes secciones se detallan las principales decisiones de arquitectura, las tecnologías utilizadas, el flujo de datos extremo a extremo y los mecanismos de prueba y validación adoptados para asegurar que AIMES cumpliera con los requisitos de desempeño y usabilidad planteados.

4.1. Marco general

El propósito de este trabajo fue desarrollar y consolidar AIMES como una solución móvil ejecutable y utilizable en condiciones cercanas al mundo real, orientada a asistir a personas con discapacidad visual durante su desplazamiento en un entorno particularmente exigente como el Metro de Santiago. Este contexto combina alta afluencia de personas, ruido ambiental, variaciones de iluminación, señalización heterogénea y obstáculos dinámicos, factores que dificultan la orientación y aumentan la incertidumbre al desplazarse de manera autónoma [1][4][10]. Frente a estas condiciones, el desafío no consiste únicamente en detectar elementos del entorno, sino en transformar esa percepción en información práctica, oportuna y comprensible para el usuario.

Para abordar este objetivo, el desarrollo se orientó a construir un flujo funcional extremo a extremo en un teléfono móvil: capturar video en tiempo real desde la cámara, procesar la escena mediante visión por computador para identificar elementos relevantes, interpretar los resultados en términos semánticos (por ejemplo, ubicación relativa y cercanía) y entregar retroalimentación auditiva mediante mensajes breves. Esta definición del flujo no solo facilita la integración técnica, sino que también permite validar progresivamente el comportamiento del sistema a través de pruebas iterativas, incorporando herramientas de depuración y parámetros ajustables para adaptar el rendimiento a distintos escenarios.

El alcance de esta versión se centra en **orientación inmediata**, entendida como la capacidad de responder a preguntas esenciales para la toma de decisiones durante la marcha: *qué se encuentra en el entorno, dónde se ubica respecto al usuario y cuán lejos está*. Este enfoque se diferencia deliberadamente de una navegación completa puerta-a-puerta. La navegación integral requiere componentes adicionales como mapas detallados del entorno, modelos de conectividad espacial, mecanismos de localización robustos y, idealmente, validación extensa con usuarios finales en múltiples estaciones y condiciones operacionales. En consecuencia, AIMES se plantea como una base funcional y escalable, donde la orientación en tiempo real constituye el núcleo que habilita

futuras extensiones hacia una navegación más avanzada.

Bajo este marco, el desarrollo de AIMES se apoya en tres pilares fundamentales:

- **Necesidad social:** contribuir a mejorar la movilidad, independencia y participación de personas con discapacidad visual en sistemas de transporte masivo como el Metro de Santiago. La asistencia propuesta busca reducir barreras en tareas cotidianas, disminuyendo la dependencia de terceros y aumentando la percepción de seguridad durante el desplazamiento.
- **Base tecnológica habilitante:** aprovechar avances recientes en visión artificial, reconocimiento de voz y capacidad de cómputo en dispositivos móviles para ejecutar la percepción y la interpretación en el propio teléfono. Esto permite una solución potencialmente escalable, que no depende de infraestructura externa (por ejemplo, beacons o cámaras instaladas) y que puede iterar mediante actualización de modelos y lógica sin modificar el entorno físico.
- **Diseño centrado en el usuario:** aplicar un enfoque de diseño centrado en necesidades reales, privilegiando interacción accesible y comprensible en movilidad. En la práctica, esto implica priorizar manos libres mediante voz, mensajes cortos y accionables, confirmación auditiva de acciones, tolerancia a ruido ambiental y mecanismos que eviten saturación por repetición de instrucciones.

Este marco establece el punto de partida conceptual y práctico para definir objetivos, alcance y decisiones de diseño a lo largo de la memoria. En particular, justifica por qué el desarrollo se concentra en una solución móvil nativa con un pipeline en tiempo real y por qué se privilegia la traducción de resultados visuales a instrucciones auditivas simples, consistentes y calibrables en terreno.

4.1.1. Visión general de la solución

Antes de describir la evolución del prototipo y la decisión de migración tecnológica, se presenta una vista de alto nivel del sistema mediante un diagrama de contexto y un

CAPÍTULO 4 : DISEÑO DE LA SOLUCIÓN

diagrama de arquitectura. Esto permite fijar los actores, entradas y salidas, junto con los módulos principales y su relación con los requisitos funcionales. Con esta base, las secciones posteriores pueden enfocarse en el proceso de prototipado y en los problemas concretos que motivaron la transición hacia una implementación nativa.

4.1.2. Diagrama de Contexto

La Figura 4.1 presenta el diagrama de contexto de AIMES, que resume la interacción entre el usuario, el dispositivo móvil y el sistema de asistencia. Se distinguen las entradas principales: (i) la **entrada visual** proveniente de la cámara, que habilita la detección e interpretación del entorno, y (ii) la **entrada del usuario** mediante comandos de voz, utilizada para controlar el asistente y definir objetivos. La salida principal corresponde a la **asistencia por audio**, emitida como instrucciones comprensibles a través del parlante o auriculares.

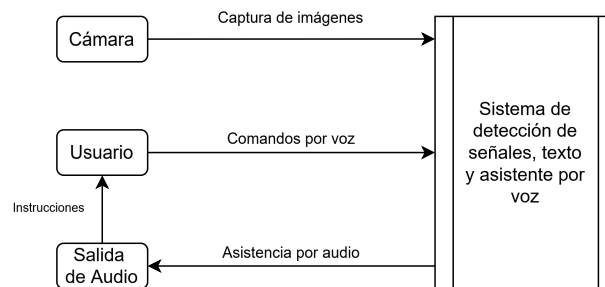


Figura 4.1: Diagrama de contexto del sistema AIMES.

Este diagrama es relevante para el desarrollo de la solución porque establece el supuesto central del proyecto: **la asistencia debe operar usando únicamente el teléfono**, sin depender de infraestructura adicional. A partir de esta interacción básica se derivan decisiones posteriores, como el énfasis en procesamiento en tiempo real en el dispositivo y el diseño de interacción manos libres.

CAPÍTULO 4 : DISEÑO DE LA SOLUCIÓN

4.1.3. Diagrama de Arquitectura

La Figura 4.2 detalla los componentes funcionales del sistema. La entrada visual se procesa mediante un bloque de **captura, preprocesamiento y detección**, que identifica elementos relevantes del entorno. Cuando la escena contiene señalética u otros elementos textuales, se aplica **OCR** para extraer información que puede apoyar decisiones de orientación (por ejemplo, validar si el camino corresponde a la estación destino). En paralelo, la entrada de voz del usuario se interpreta mediante **STT**, habilitando funciones como iniciar/detener guía o definir un objetivo de búsqueda.

A partir de la información visual y del contexto generado por la interacción, el sistema produce una respuesta auditiva a través de **TTS**. Para favorecer continuidad, el sistema mantiene un historial mínimo (por ejemplo, la última asistencia emitida) de modo que las indicaciones no sean contradictorias ni excesivamente repetitivas entre frames consecutivos.

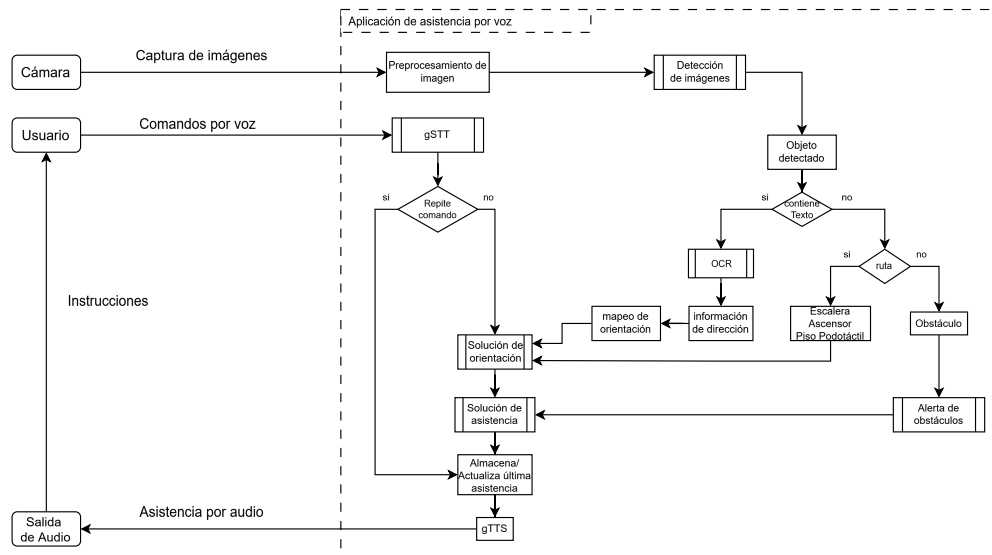


Figura 4.2: Diagrama de arquitectura del sistema AIMES.

4.1.4. Enumeración de Módulos

Para mantener separación de responsabilidades y permitir evolución incremental del sistema, AIMES se organiza en cuatro módulos principales, resumidos en la Tabla

CAPÍTULO 4 : DISEÑO DE LA SOLUCIÓN

4. Esta división es útil tanto para planificar el desarrollo como para trazar qué componentes habilitan cada requisito funcional.

Módulo	Propósito	Sección
Investigación del Usuario y Diseño de Producto (UX_PD)	Comprender contexto de uso, necesidades y barreras de personas con discapacidad visual para diseñar una interacción accesible y funcional.	4.1.5
Captura, Corrección y Preprocesamiento Visual (Ex_Im)	Obtener video en tiempo real desde el dispositivo móvil y procesarlo para su uso por los módulos de detección.	4.1.5
Detección y Dirección con modelo de <i>computer vision</i> (Det_OI + Dir_OI)	Detectar señales/elementos relevantes y generar información útil para orientar al usuario hacia su objetivo.	4.1.5
Interacción por Voz (Rec_Prompt + TaV_Dir)	Permitir comandos por voz y entregar retroalimentación auditiva mediante síntesis de voz.	4.1.5

Tabla 4: Resumen de módulos del sistema AIMES.

4.1.5. Descripción funcional de módulos

A continuación se describen los módulos del sistema AIMES y su rol dentro de la arquitectura propuesta. Para cada módulo se presenta su propósito y alcance, junto con dependencias, supuestos y restricciones. Además, se incluye el *estado de implementación* correspondiente a esta memoria, de modo que sea explícito qué componentes fueron desarrollados y cuáles quedan como trabajo futuro.

Módulo 1: Investigación del Usuario y Diseño de Producto (UX_PD)

Propósito: Comprender el contexto de uso, necesidades y barreras de personas con

CAPÍTULO 4 : DISEÑO DE LA SOLUCIÓN

discapacidad visual para diseñar una experiencia de interacción accesible, funcional y segura. Este módulo establece los criterios de diseño que guían los módulos técnicos, definiendo qué información debe comunicar el sistema, cómo debe hacerlo en un entorno ruidoso y qué restricciones de interacción deben respetarse para no aumentar la carga cognitiva del usuario.

Descripción y desarrollo: El desarrollo de UX_PD se estructuró en dos niveles complementarios. En primer lugar, se realizó una investigación de base (*investigación dura*) orientada a recopilar información relevante para fundamentar decisiones del sistema: tecnologías de accesibilidad disponibles, patrones de interacción auditiva, soluciones existentes y consideraciones de seguridad y privacidad al operar con cámara y audio en movilidad. Esta etapa alimenta el Estado del Arte y permite justificar decisiones técnicas posteriores (por ejemplo, priorización de retroalimentación auditiva y operación sin infraestructura externa).

En segundo lugar, durante el avance de los módulos técnicos se ejecutó una investigación de apoyo (*investigación ligera*), orientada a resolver necesidades emergentes del desarrollo: selección de estrategias de interacción por voz, criterios de diseño de mensajes, y revisión de prácticas de accesibilidad en Android. Este enfoque evitó expandir el alcance teórico sin impacto práctico, privilegiando evidencia útil para implementar y probar el sistema.

Alcance: Este módulo abarca desde la investigación inicial hasta la definición de flujos de interacción y criterios de accesibilidad. Incluye entrevistas y observaciones en terreno, diseño de interacción con retroalimentación auditiva, y criterios para compatibilidad con lectores de pantalla. También contempla el levantamiento de riesgos asociados al uso de smartphone en movilidad y medidas para mitigarlos (por ejemplo, evitar interacción innecesaria con información sensible y privilegiar ejecución local cuando sea posible).

Dependencias: Es un módulo inicial que no depende de componentes técnicos para comenzar, pero sus resultados condicionan el funcionamiento de los módulos posterior-

CAPÍTULO 4 : DISEÑO DE LA SOLUCIÓN

res. En particular, define prioridades de clases a detectar (por relevancia funcional), estructura de mensajes auditivos, y criterios de control manos libres.

Supuestos:

- Acceso a personas con discapacidad visual para entrevistas y retroalimentación.
- Posibilidad de realizar observaciones y pruebas en estaciones reales o escenarios simulados.
- Disponibilidad de literatura y guías de accesibilidad (Android/lectores de pantalla).

Restricciones:

- Tiempo limitado para iteraciones completas de diseño con validación en terreno.
- Disponibilidad acotada de participantes para pruebas durante el desarrollo.
- Condiciones logísticas y permisos para realizar levantamiento en estaciones.

Estructura:

- *Entradas:* entrevistas, observaciones, registros de terreno, revisión bibliográfica y tecnológica.
- *Ejecución:* análisis de contexto, definición de flujos, criterios de interacción, diseño de mensajes y evaluación iterativa.
- *Salidas:* perfil de usuario, requisitos de accesibilidad, flujos de interacción y criterios para módulos técnicos.

Hallazgos clave en visitas y entrevistas: Durante la visita a la Biblioteca Central para Ciegos se observaron prácticas de interacción altamente auditivas. En computadores, los usuarios utilizan lectores de pantalla (por ejemplo, NVDA[47]) y navegación

CAPÍTULO 4 : DISEÑO DE LA SOLUCIÓN

mediante teclado, descartando el mouse. En teléfonos, se utiliza TalkBack[48] (Android) o VoiceOver[49] (iPhone), confirmando que el smartphone es un dispositivo central en actividades cotidianas. Este hallazgo apoyó la decisión de construir AIMES como una solución móvil, considerando desde el diseño la compatibilidad con funciones de accesibilidad integradas.

En el contexto del Metro, se destacó la importancia del piso podotáctil como guía física esencial, por lo que referenciarlo mediante detección y mensajes auditivos no es redundante, sino funcional. Además, se observó la necesidad de confirmación clara y oportuna de acciones (por ejemplo, cuando el sistema cambia de estado o interpreta un comando), ya que el usuario requiere confianza en la intención del sistema para actuar con seguridad.

Se hizo una entrevista en terreno a Don Iván, una persona con discapacidad visual que se desplaza de forma independiente por la red de Metro. Según la encuesta del Anexo.1, se tuvo la siguiente retroalimentación:

Don Iván señaló que utiliza principalmente su bastón para orientarse y desplazarse por el Metro, pero enfrenta dificultades al interactuar con pantallas táctiles, especialmente en las máquinas de recarga Bip!, debido a la falta de referencias sobre dónde tocar. Ante esto, consideró que una aplicación capaz de asistir en la ubicación del dedo o leer la información de la pantalla podría facilitar su autonomía. También destacó la utilidad de disponer de información sobre servicios cercanos, como cajeros automáticos y tiendas, cuya ubicación no siempre puede identificar con claridad.

Asimismo, mencionó dificultades para determinar su posición en el andén al bajar del tren, lo que afecta su orientación posterior. Consideró relevante que un sistema pudiera identificar las salidas específicas de una estación y proporcionar indicaciones según el destino. Finalmente, propuso el uso de una cámara externa acompañada de un altavoz como una alternativa para entregar información del entorno de manera más segura y eficiente.

Tras aquella entrevista, se identificó necesidades concretas: dificultad para interac-

CAPÍTULO 4 : DISEÑO DE LA SOLUCIÓN

tuar con pantallas táctiles públicas (máquinas de recarga Bip), interés por ubicar servicios dentro de estaciones (cajeros, tiendas), incertidumbre al bajar del tren respecto a la ubicación dentro del andén, y valor en conocer salidas específicas para seleccionar la dirección correcta. También surgió la idea de explorar hardware complementario como: cámara externa (tipo GoPRO), y/o audífonos manos libres, como alternativa más segura al uso del teléfono personal; en esta memoria se considera como línea futura, manteniendo el teléfono como plataforma base por escalabilidad y adopción.

Requisitos específicos derivados del contexto: A partir de los hallazgos se definieron requisitos orientadores para el diseño:

1. Compatibilidad con TalkBack/VoiceOver y no interferencia con lectores de pantalla.
2. Retroalimentación auditiva breve, clara y comprensible en entornos ruidosos.
3. Detección y referencia explícita a piso podotáctil como guía.
4. Orientación hacia puntos de interés y servicios cercanos dentro de estaciones.
5. Confirmación auditiva del estado del sistema y de comandos interpretados.
6. Consideración de consumo energético y estabilidad del sistema en uso prolongado.

Estado de implementación en esta memoria: En el desarrollo individual de esta memoria, el aporte de UX_PD se refleja principalmente en criterios trasladados a la aplicación móvil:

- Priorización de operación manos libres mediante comandos por voz y salida por TTS.
- Diseño de mensajes cortos y directos (evitando saturación y repetición).
- Inclusión de piso podotáctil y objetos de interés como clases relevantes del detector.

CAPÍTULO 4 : DISEÑO DE LA SOLUCIÓN

- Incorporación de OCR como mecanismo auxiliar para validar dirección/estación cuando la señalética es visible.
- Consideración de estabilidad y control del ciclo de vida como requisito práctico para uso en terreno.

Módulo 2: Captura, Corrección y Preprocesamiento Visual (Ex_Im)

Propósito: Obtener video del entorno en tiempo real desde el dispositivo móvil y transformarlo en una representación adecuada para los módulos de detección y orientación. Este módulo es crítico porque condiciona directamente la estabilidad del sistema: si la captura es inestable, si los frames llegan con retraso o si el formato no es consistente, la detección pierde utilidad y la retroalimentación auditiva se vuelve tardía o errática.

Descripción y desarrollo: El desarrollo de Ex_Im se enfocó en asegurar una cadena confiable de captura y preparación de frames en un contexto móvil. Para ello, se definieron criterios técnicos mínimos sobre: resolución de entrada, orientación de imagen, latencia de captura, estabilidad del stream y costo computacional del preprocesamiento. Dado que AIMES funciona como asistencia en tiempo real, la prioridad no fue maximizar calidad visual, sino garantizar **continuidad y consistencia** del pipeline bajo condiciones variables.

En términos funcionales, este módulo incluye: (i) inicialización y control del stream de cámara, (ii) extracción de frames para análisis, (iii) conversión entre formatos de imagen requeridos por el modelo (por ejemplo, desde el formato de cámara hacia `Bitmap`/buffer), (iv) ajustes básicos previos a inferencia (reescalado, normalización y rotación), y (v) mecanismos de control para evitar acumulación de frames cuando el procesamiento es más lento que la captura.

Adicionalmente, en la definición del proyecto se consideró la posibilidad de incorporar una etapa de corrección de imagen para mitigar interferencias físicas en la cámara (manchas, rayas, stickers). Esta línea se mantiene como una extensión futura, ya que requiere datasets específicos y modelos de reconstrucción visual, además de un costo

CAPÍTULO 4 : DISEÑO DE LA SOLUCIÓN

computacional relevante para ejecución en dispositivo.

Alcance: Incluye la captura de video en tiempo real desde el móvil, conversión y preprocesamiento de frames para inferencia, y criterios de estabilidad/latencia para asegurar que la detección sea oportuna. La corrección avanzada de imagen se considera fuera del alcance implementado en esta memoria (línea de trabajo futuro) salvo etapas básicas de normalización o ajustes necesarios para el modelo.

Dependencias: Depende de los criterios de interacción definidos en UX_PD (por ejemplo, necesidad de baja latencia y mensajes oportunos) y de los requerimientos del modelo del Módulo 3 (tamaño de entrada, formato, normalización).

Supuestos:

- El dispositivo cuenta con cámara con resolución y estabilidad suficientes para captura continua.
- La iluminación y movimiento del entorno pueden variar, pero el pipeline debe mantener consistencia.

Restricciones:

- Recursos limitados en smartphone: CPU/GPU, batería y memoria.
- Latencia acotada: el sistema debe evitar retrasos acumulativos entre captura y respuesta.

Estructura:

- *Entradas:* stream de cámara del dispositivo.
- *Ejecución:* captura, conversión de formato, rotación/reescalado, preparación del frame para inferencia.
- *Salidas:* frames consistentes y listos para detección/orientación.

Estado de implementación en esta memoria

En la versión desarrollada para esta memoria:

- Se implementó captura en tiempo real y preprocesamiento básico: conversión, rotación, redimensionamiento y preparación para inferencia.
- Se incorporó conversión y ajuste de orientación (rotación) para asegurar coherencia de entrada al modelo.
- Se priorizó estabilidad del pipeline y reducción de backlog de frames para mantener respuesta oportuna.
- La corrección avanzada de imagen (autoencoders/GANs) queda planteada como trabajo futuro.

Módulo 3: Detección de Objetos y Dirección (Det_OI + Dir_OI)

Propósito: Detectar elementos visuales relevantes del entorno en tiempo real y traducirlos en información espacial útil para orientar al usuario. Este módulo transforma la imagen en conocimiento accionable: identifica clases de interés (por ejemplo, piso podotáctil y objetos relevantes) y genera una representación simplificada que pueda comunicarse por audio (posición relativa y cercanía).

Descripción y desarrollo: El desarrollo de Det_OI + Dir_OI se organizó en dos componentes: detección y generación de dirección/orientación. En la fase de detección, se utilizó un modelo entrenado para clases relevantes del entorno del Metro, optimizado para ejecutarse en un teléfono mediante un formato ligero (por ejemplo, TFLite). La ejecución en dispositivo implica equilibrar precisión y costo computacional, priorizando estabilidad y tiempos de respuesta consistentes.

Tras la inferencia, se aplica postprocesamiento para filtrar detecciones poco confiables y reducir duplicidad (por ejemplo, mediante umbrales de confianza y supresión de cajas solapadas). Con la lista final de detecciones, la fase de dirección espacial construye una interpretación de alto nivel: se estima la **posición relativa** del objeto dentro del

CAPÍTULO 4 : DISEÑO DE LA SOLUCIÓN

campo visual (izquierda/centro/derecha) y una **distancia relativa** (cerca/medio/lejos) basada en heurísticas geométricas, calibrables por dispositivo. Esta elección privilegia interpretabilidad y baja latencia, siendo adecuada para un MVP de orientación inmediata.

Además, este módulo integra OCR como apoyo cuando la escena contiene señalética: el texto detectado puede utilizarse para validar si el camino corresponde a la estación destino o a la dirección esperada, reduciendo ambigüedad en decisiones de orientación.

Alcance: Incluye detección en tiempo real en el dispositivo, postprocesamiento, visualización de depuración mediante overlays y generación de orientación simple basada en posición y distancia relativa. Incluye OCR como mecanismo auxiliar cuando hay texto relevante en escena.

Dependencias: Depende del Módulo 2 para recibir frames consistentes y con el formato correcto, y del Módulo 1 para priorizar clases relevantes y definir qué información es útil comunicar al usuario.

Supuestos:

- El modelo de detección fue entrenado o ajustado para condiciones cercanas al Metro (variabilidad de iluminación y oclusiones).
- Las clases seleccionadas representan elementos de utilidad real para la orientación inmediata.

Restricciones:

- Compromiso precisión vs velocidad: la inferencia debe ser rápida para mantener utilidad.
- Variabilidad visual alta: movimiento de personas, oclusiones y fondos complejos.

Estructura:

CAPÍTULO 4 : DISEÑO DE LA SOLUCIÓN

- *Entradas:* frames preprocesados desde Ex_Im.
- *Ejecución:* inferencia del modelo, filtrado/postprocesamiento, estimación de posición y distancia, OCR (si aplica).
- *Salidas:* detecciones priorizadas y representación espacial (posición/distancia relativa) para ser comunicada al usuario.

Estado de implementación en esta memoria

En la versión desarrollada para esta memoria:

- Se integró inferencia en dispositivo con modelo TFLite y etiquetas del sistema.
- Se incorporó postprocesamiento (umbral y reducción de duplicados) y visualización de depuración mediante overlay.
- Se implementó orientación inmediata basada en posición relativa y distancia estimada por heurística calibrable.
- Se incorporó OCR para apoyar validación de dirección/estación cuando la señalética es visible.

Módulo 4: Interacción por Voz (Rec_Prompts + TaV_Dir)

Propósito: Permitir que el usuario controle el sistema sin depender de interacción visual, habilitando comandos por voz y retroalimentación auditiva. Este módulo convierte AIMES en un asistente usable en movilidad: interpreta intenciones del usuario (STT) y comunica instrucciones claras mediante síntesis de voz (TTS).

Descripción y desarrollo: El desarrollo de Rec_Prompts + TaV_Dir se enfocó en la integración móvil nativa para asegurar compatibilidad, estabilidad y control fino del ciclo de audio. La interacción se organiza en dos flujos: (i) **recepción de comandos** del usuario mediante STT para iniciar/detener la guía o definir objetivos, y (ii) **entrega de asistencia** mediante TTS, generando instrucciones breves y accionables a partir del contexto visual del Módulo 3.

CAPÍTULO 4 : DISEÑO DE LA SOLUCIÓN

Una dificultad central en sistemas que combinan STT y TTS es evitar interferencia entre ambos: si el reconocedor permanece activo durante la síntesis, existe riesgo de auto-escucha (el sistema interpreta su propia voz) y gatilla acciones no intencionales. Por lo tanto, se incorporan reglas de control para pausar/reanudar escucha según el estado del asistente, junto con mecanismos de reducción de repetición: umbrales mínimos de confianza, temporizadores (*cooldown*) y emisión condicionada a cambios relevantes (por ejemplo, cambio de lado o cambio de cercanía). Esta estrategia busca evitar saturación auditiva, manteniendo instrucciones útiles y coherentes.

El diseño de mensajes se alinea con lo observado en UX_PD: frases cortas, vocabulario controlado y confirmación de acciones (por ejemplo, al fijar objetivo o activar guía), considerando que el entorno del Metro presenta ruido y distracciones.

Alcance: Incluye reconocimiento de voz para comandos esenciales del sistema y síntesis de voz para retroalimentación e instrucciones de orientación. No incluye conversación abierta; se restringe a un set de comandos y respuestas diseñadas para control y guía.

Dependencias: Depende del Módulo 3 para disponer del contexto visual (detecciones + orientación), y del Módulo 1 para definir estilo de mensajes, confirmaciones y criterios de accesibilidad.

Supuestos:

- El usuario utilizará salida de audio del dispositivo y, idealmente, auriculares para mayor claridad.
- La aplicación puede operar con ruido de fondo, aunque la tasa de reconocimiento puede variar.

Restricciones:

- Ruido ambiental y reverberación en estaciones afectan desempeño del STT.
- Latencia de respuesta debe ser baja para mantener utilidad en movimiento.

Estructura:

- *Entradas:* comandos de voz del usuario; contexto visual/orientación del sistema.
- *Ejecución:* STT e interpretación de intención; generación de frases; control de repetición; TTS.
- *Salidas:* confirmaciones e instrucciones auditivas claras y contextualizadas.

Estado de implementación en esta memoria

En la versión desarrollada para esta memoria:

- Se implementó TTS nativo de Android para emitir instrucciones y confirmaciones.
- Se integró reconocimiento de voz nativo (STT) para comandos de control y definición de objetivo.
- Se incorporaron reglas para evitar saturación y repetición excesiva, y para coordinar STT/TTS.

4.1.6. Matriz de Requisitos Funcionales y Módulos

La Tabla 5 muestra la trazabilidad entre requisitos funcionales y módulos. Esta matriz permite verificar coherencia del diseño y, al mismo tiempo, identificar rápidamente qué módulos deben estar operativos para habilitar cada capacidad del sistema. Más adelante, esta trazabilidad es útil para justificar decisiones de priorización durante el prototipado y la migración tecnológica.

CAPÍTULO 4 : DISEÑO DE LA SOLUCIÓN

Requisito	Módulo 1	Módulo 2	Módulo 3	Módulo 4
Captura de imágenes en tiempo real		X		
Modelo de detección de señales y texto del Metro			X	
Reconocimiento de texto a voz (comandos del usuario)				X
Orientación a través de voz para el usuario	X		X	X
Uso de cámara del móvil para detección		X		
Aplicación móvil para interacción	X			X
Auriculares o salida de audio para instrucciones				X
Hardware de desarrollo (PC, cámara, auriculares)	X	X	X	X
Software de desarrollo (OpenCV, Android Studio, etc.)	X	X	X	X

Tabla 5: Matriz de trazabilidad entre requisitos y módulos del sistema.

4.1.7. Revisión de Arquitectura de Diseño

A partir del diagrama de arquitectura (Figura 4.2), se revisó la coherencia entre los bloques definidos, los módulos y los requisitos del sistema. En esta revisión se identificaron componentes de alta complejidad, que requerían definiciones más precisas para evitar ambigüedades de implementación y riesgos de integración.

Como resultado, se constató que la división por módulos representa adecuadamente la intención del proyecto y permite organizar el avance por hitos. Sin embargo, también quedó en evidencia la necesidad de priorizar algunos bloques que, aunque deseables

desde el punto de vista funcional o de experiencia de usuario, no aportaban directamente al MVP dentro del tiempo disponible, o bien introducían dependencias tecnológicas adicionales difíciles de resolver en la primera iteración.

Esta revisión justificó una estrategia incremental, donde primero se valida el flujo extremo a extremo (captura–detección–interpretación–voz) y luego se incorporan mejoras. De hecho, se decidió concentrar los esfuerzos en asegurar la robustez del procesamiento principal de la señal y la estabilidad de las interfaces entre módulos, dejando para fases posteriores funcionalidades avanzadas, como optimizaciones de rendimiento, soporte para configuraciones más complejas y mecanismos de personalización.

Con esta base definida, la sección siguiente describe cómo los prototipos iniciales permitieron validar parte del flujo y, al mismo tiempo, evidenciaron problemas prácticos que llevaron a migrar hacia una implementación nativa. Estos prototipos funcionaron además como herramientas de exploración rápida, ayudando a precisar los requisitos reales de cada bloque, depurar supuestos de diseño y ajustar el alcance del MVP en función de las restricciones técnicas y de calendario identificadas durante la evaluación de la arquitectura.

4.2. Evolución del prototipo y decisión de migración

El desarrollo de AIMES se apoyó en una estrategia incremental basada en prototipos, con el objetivo de reducir la incertidumbre técnica y validar tempranamente el flujo de interacción antes de consolidar una aplicación móvil en Android. En concreto, el proceso consideró dos etapas previas de prototipado: (i) un prototipo ejecutado en computador, utilizado como entorno controlado para validar el flujo general del sistema y las piezas de visión por computador; y (ii) un prototipo orientado a dispositivo móvil mediante un enfoque multiplataforma, utilizado para evaluar la factibilidad de integrar cámara, audio e interacción por voz en un formato cercano al uso final. Las lecciones obtenidas en estas etapas condujeron finalmente a la decisión de implementar AIMES como una aplicación nativa en Android, priorizando la robustez y el desempeño en

tiempo real.

En la primera etapa, se construyó un prototipo en computadora que funcionó como base para consolidar el flujo lógico de la solución. Esta decisión se justificó porque el desarrollo y la evaluación de modelos de visión (entrenamiento, pruebas iniciales y ajustes de detección), se realizan naturalmente en un entorno de escritorio, donde existe mayor control sobre los recursos de cómputo, herramientas de depuración y disponibilidad de librerías. En este prototipo, se definió un pipeline simple que replica el comportamiento esperado de la aplicación: adquisición de imagen o video, ejecución del modelo de detección, procesamiento de resultados y generación de una salida interpretada. Este enfoque permitió verificar de manera rápida la coherencia del flujo, identificar necesidades de preprocesamiento y postprocesamiento, y establecer criterios para traducir detecciones a instrucciones comprensibles antes de enfrentar las restricciones de un teléfono.

Posteriormente, se desarrolló un segundo prototipo orientado al uso móvil, construido con herramientas de desarrollo rápido y compatibilidad multiplataforma. El objetivo de esta etapa fue trasladar el flujo validado en computador hacia un escenario más cercano al real, integrando funcionalidades críticas del dispositivo, como cámara en tiempo real y componentes de voz (reconocimiento y síntesis). En este prototipo se priorizó la rapidez de implementación, buscando obtener una versión operativa en poco tiempo y con un esfuerzo de ingeniería acotado, con el fin de validar la interacción y continuidad del pipeline en movilidad. Este prototipo se centró en:

- Comprobar la viabilidad de los flujos de interacción propuestos.
- Validar el modelo conceptual de AIMES con usuarios y partes interesadas.
- Identificar tempranamente cuellos de botella en la experiencia de uso.

Esta etapa proporcionó una retroalimentación valiosa sobre la usabilidad y sobre los requisitos funcionales que no habían sido plenamente capturados en el diseño inicial.

CAPÍTULO 4 : DISEÑO DE LA SOLUCIÓN

Sin embargo, también puso de manifiesto las limitaciones propias del enfoque multiplataforma, en particular relacionadas con el rendimiento, la latencia en la actualización de la interfaz y el acceso eficiente a los recursos del sistema que eran necesarios y mandatorios para el uso de la aplicación.

En consecuencia, el prototipo dejó de evaluarse sólo como una prueba de concepto y pasó a ser un insumo para tomar decisiones de arquitectura. En ese punto, las restricciones observadas se tradujeron en criterios concretos de implementación, lo que llevó a definir la migración a una solución nativa como el camino más adecuado para sostener el desempeño y la estabilidad requeridos.

A partir de estos resultados, se tomó la decisión de migrar a una implementación nativa. Los motivos principales fueron:

- **Robustez:** necesidad de un control más fino sobre gestión de memoria, manejo de errores y estabilidad en escenarios de uso prolongado.
- **Desempeño en tiempo real:** requerimiento de tiempos de respuesta consistentes y baja latencia para procesamiento y visualización, difícil de garantizar con la capa de abstracción del prototipo multiplataforma.
- **Integración con el sistema:** acceso directo a APIs nativas y optimizaciones específicas de la plataforma que no eran plenamente accesibles desde la solución inicial.

La migración se planteó de forma incremental para reducir riesgo y mantener continuidad con los flujos ya validados. En una primera etapa, se aislaron los módulos críticos identificados durante el prototipado (por ejemplo, los relacionados con captura y procesamiento en tiempo real), reimplementándolos de manera nativa y comparando su comportamiento con el prototipo original. Luego, se sustituyeron progresivamente los demás componentes hasta obtener una versión completamente nativa que preservara la lógica e interacción validadas, pero con mejores garantías de rendimiento y estabilidad.

Finalmente, y para contextualizar esta decisión dentro del proceso global de desarrollo, en la sección siguiente se describen las dos etapas de prototipado realizadas (prototipo en computador y prototipo multiplataforma), junto con los aprendizajes que guiaron la transición hacia Android nativo.

4.2.1. Prototipo en computador: validación del pipeline visual con Python + OpenCV

El objetivo de esta etapa era definir un procedimiento experimental que permitiera evaluar distintas configuraciones de la arquitectura YOLOv11 en su versión nano (YOLOv11n), el cual es el más ligero para dispositivos móviles, y seleccionar aquella que ofreciera el mejor equilibrio entre precisión de detección y rendimiento computacional para su implementación en dispositivos móviles.

Se consideraron las siguientes configuraciones de modelo:

- **YOLOv11n Tradicional**, correspondiente a la arquitectura base de Ultralytics.
- **YOLOv11n + P2**, donde se incorpora una capa extra para priorización de detección de objetos pequeños/lejanos en el frame a procesar.
- **YOLOv11n + RepViT**, reemplaza el backbone tradicional con uno que utiliza Transformers, orientado a usos con recursos limitados.

Entrenamiento del modelo: El conjunto de datos fue dividido en subconjuntos de entrenamiento, validación y prueba mediante la plataforma Roboflow. Posteriormente, cada configuración fue entrenada utilizando los mismos parámetros experimentales, permitiendo comparar objetivamente el desempeño obtenido por cada arquitectura.

Las imágenes que conforman el dataset fueron capturadas mediante dos dispositivos móviles: un Samsung Galaxy A33 y un Redmi Note 8. El levantamiento se realizó mediante visitas en terreno cubriendo la totalidad de las estaciones de la Línea 6 del Metro de Santiago, y de forma acotada en cuatro estaciones de la Línea 3 (Ñuble, Ferrocarril, Los Libertadores y Plaza Egaña), en distintos horarios del día con el fin

CAPÍTULO 4 : DISEÑO DE LA SOLUCIÓN

de capturar variabilidad de condiciones de iluminación y afluencia de personas. Las capturas incluyen tanto escenas con presencia de obstáculos y elementos del entorno (mobiliario, flujo peatonal) como imágenes claras y sin interferencias de cada uno de los objetos de interés considerados por el sistema de detección.

Durante el entrenamiento, el modelo ajusta iterativamente sus pesos mediante retropropagación (backpropagation), minimizando la función de pérdida (loss) asociada a la localización y clasificación de los objetos presentes en las imágenes. Este proceso continúa hasta completar el número de cien épocas, generando así el modelo de detección.

Criterios de Selección de modelo: Con el objetivo de seleccionar la configuración más adecuada para su integración en la aplicación Android, se definieron dos criterios principales de evaluación:

1. **Desempeño de detección**, el cual define la capacidad del modelo en detectar los elementos de interés en el entorno.
2. **Rendimiento Computacional**, el cual define el desempeño en detección y el costo computacional asociado a cada configuración.

Bajo ambos criterios mencionados, se tienen los siguientes parámetros que determinan la factibilidad del modelo para su uso con la aplicación móvil:

- **mAP@0.5 (mean Average Precision):** Promedio de la precisión obtenida para todas las clases, considerando una detección correcta cuando la intersección sobre la unión.
- **Precisión (Precision):** Proporción de detecciones realizadas por el modelo que corresponden efectivamente a objetos presentes en la imagen.
- **Recall (Exhaustividad):** Proporción de objetos reales que fueron correctamente detectados por el modelo.

- **F1 Score:** Corresponde a la media armónica entre la precisión y el recall.
- **Número de parámetros:** Representa el tamaño de la arquitectura del modelo.
- **Frames por segundo (FPS):** Cantidad de imágenes que el modelo es capaz de procesar por segundo durante la inferencia.
- **Tiempo de inferencia (latencia):** Tiempo promedio requerido para procesar una imagen y generar las detecciones.
- **Consumo de recursos:** Considera el peso de la aplicación y capacidad de procesamiento durante la ejecución del modelo.

La selección del modelo se realizó considerando el balance entre ambos criterios, buscando obtener un modelo que mantenga un desempeño adecuado mientras satisface los requisitos de procesamiento en tiempo real de la aplicación móvil.

4.2.2. Prototipo inicial con Kivy/Buildozer (Vosk + OpenCV + RHVoice)

La primera aproximación móvil de AIMES se construyó con un enfoque multiplataforma utilizando Kivy/Buildozer, integrando reconocimiento de voz offline con Vosk, síntesis de voz con RHVoice y procesamiento visual con OpenCV. La motivación fue avanzar rápidamente hacia una versión funcional en teléfono, reutilizando componentes ya desarrollados en Python y validando tempranamente la hipótesis de uso: capturar escena, detectar elementos relevantes y comunicar orientación mediante audio, sin depender de infraestructura externa.

En un entorno de escritorio, el flujo base resultaba demostrable: era posible ejecutar procesamiento visual y observar resultados de detección sobre videos o imágenes, además de ensayar la lógica general de interacción. Sin embargo, al intentar trasladar ese mismo prototipo al teléfono, el principal obstáculo fue **la imposibilidad de ejecutar la aplicación de forma efectiva en el dispositivo**. En términos prácticos, la aplicación compilaba, generaba el APK e incluso podía instalarse, pero no abría en el teléfono

(salía inmediatamente o fallaba al iniciar). Este problema se transformó en un bloqueador temprano, ya que impedía validar en terreno la integración real de cámara, audio y rendimiento.

Este comportamiento evidenció una brecha importante entre lo que era posible validar en el computador y lo que ocurría en el teléfono. Desde el PC se podía observar parte del comportamiento del pipeline (por ejemplo: procesamiento y salidas intermedias), pero en el dispositivo no existía una visualización equivalente ni una trazabilidad simple para entender qué componente fallaba durante el inicio. En otras palabras, el prototipo permitía explorar la idea a nivel conceptual, pero la falta de ejecución estable en el teléfono impidió confirmar aspectos críticos para el caso de uso real: inicialización de cámara, permisos, manejo de audio y desempeño continuo.

Problema 1: falla de ejecución en dispositivo (compila/instala, pero no inicia). El problema más determinante en esta etapa fue que el prototipo no alcanzó un estado “operable” en Android. Aunque la herramienta de empaquetado generaba el APK y el proceso de instalación se completaba, la app no lograba iniciar correctamente. En términos de desarrollo, esto implicó que gran parte del esfuerzo se destinara a resolver problemas de integración y empaquetado en lugar de avanzar en funcionalidad. Además, al fallar en la etapa más básica (arranque), no era posible iterar con pruebas reales, calibrar parámetros ni validar interacción.

Problema 2: diferencias PC vs teléfono (validación parcial y no transferible). Aunque ciertos componentes eran verificables en computador (por ejemplo: detecciones sobre videos del Metro o comportamiento de módulos aislados), la transición al teléfono introdujo restricciones que el prototipo no logró absorber. La diferencia no fue solamente de rendimiento; fue de **capacidad de ejecución** y de **observabilidad**. En PC, el entorno de desarrollo permitía inspeccionar fácilmente salidas intermedias, imprimir trazas y reproducir el pipeline. En el teléfono, en cambio, la falta de ejecución estable y de herramientas de diagnóstico accesibles en ese stack multiplataforma

CAPÍTULO 4 : DISEÑO DE LA SOLUCIÓN

dificultó identificar causas raíz, generando un ciclo de iteración lento y poco confiable. En la Tabla 6 se resumen las limitaciones e impactos de la primera implementación.

Área	Impacto observado
Ejecución en dispositivo	Compila e instala, pero no inicia en el teléfono (bloqueador principal).
Brecha PC vs teléfono	Validación posible en PC, pero no transferible a uso real por falta de ejecución/observabilidad en móvil.
Permisos/compatibilidad	Flujos de permisos y variabilidad de dispositivos incrementan el costo de depuración y validación.

***Tabla 6:** Limitaciones críticas observadas en el prototipo Kivy/Buildozer y su impacto.*

En síntesis, esta etapa fue útil para explorar flujos y validar parcialmente la hipótesis del asistente, pero evidenció un problema determinante: el prototipo no alcanzó un nivel de ejecutabilidad y control suficiente en el teléfono. La combinación de (i) fallos de arranque pese a compilar e instalar, (ii) validación no transferible desde PC al entorno móvil, y (iii) fragilidad en cámara/audio, condujo a concluir que la solución requería una implementación nativa para garantizar control del ciclo de vida, trazabilidad, rendimiento y estabilidad en tiempo real.

4.2.3. Migración a Android nativo con Jetpack Compose

Dado que el objetivo final requiere una solución utilizable en terreno, la migración a una implementación nativa se volvió una condición necesaria. Esta decisión permitió controlar directamente:

- la captura y análisis de cámara en tiempo real,
- la ejecución de inferencia local (sin depender de conectividad),

- la interacción por voz (STT) y retroalimentación por voz (TTS) con reglas de estabilidad,
- el manejo de permisos, ciclo de vida y trazabilidad de ejecución.

Además, al implementar la interfaz con Jetpack Compose, fue posible construir pantallas de prueba y depuración dentro de la misma app, acelerando la iteración durante pruebas.

4.3. Delimitación del alcance implementado

A partir de la arquitectura propuesta y de las restricciones detectadas durante el prototipado, la implementación descrita en esta memoria se acota a un conjunto de capacidades centradas en la asistencia inmediata. El propósito no fue desarrollar una navegación completa en la red de Metro, sino construir una base móvil funcional capaz de percibir el entorno, interpretar los elementos relevantes y comunicar información útil al usuario mediante audio. A continuación, en la Tabla 7 se pueden observar las distintas capacidades desarrolladas dentro de la aplicación AIMES.

CAPÍTULO 4 : DISEÑO DE LA SOLUCIÓN

Capacidad	Alcance desarrollado
Detección de objetos de interés	Identificación de elementos relevantes del entorno y generación de información sobre su posición relativa respecto de la cámara.
Detección de piso podotáctil	Reconocimiento de líneas y superficies podotáctiles como elementos prioritarios para apoyar la orientación del usuario.
Estimación de distancia	Clasificación aproximada de la cercanía del objeto mediante categorías interpretables, tales como cerca, medio y lejos.
Reconocimiento de texto	Aplicación de OCR sobre señalética para contrastar el texto reconocido con la estación o destino definido por el usuario.
Interacción por voz	Recepción de comandos y entrega de instrucciones auditivas mediante mecanismos nativos de Android.

Tabla 7: Capacidades consideradas en el alcance implementado de AIMES.

4.4. Diseño de interacción del sistema

En AIMES, el diseño de interacción cumple un rol estructural: traduce capacidades técnicas (captura visual, detección, OCR, STT/TTS) en una experiencia utilizable por personas con discapacidad visual en un entorno altamente demandante como el Metro. A diferencia de aplicaciones tradicionales donde la interfaz visual es el canal principal, en este sistema la interacción se fundamenta en un modelo **auditivo y por voz**, donde el usuario debe poder comprender el estado del sistema, controlar acciones y recibir orientación sin requerir atención visual sostenida.

Desde un punto de vista de ingeniería de interacción, el problema central no es únicamente “qué detecta” el sistema, sino **cómo comunica** esa información y **cómo permite al usuario controlar el asistente** en condiciones reales: ruido ambiental, interrupciones, movimiento continuo y variabilidad del entorno. Por ello, el diseño se organiza en principios y flujos que acotan la complejidad y reducen carga cognitiva.

4.4.1. Principios de diseño

El diseño de interacción se guía por cuatro principios fundamentales:

- **Accesibilidad como canal principal:** la interfaz debe ser operable mediante voz y audio, de modo que las funciones esenciales puedan ejecutarse sin depender de visión ni de interacción táctil compleja.
- **Mensajes accionables y de baja carga cognitiva:** la salida del sistema debe priorizar información inmediatamente útil para la toma de decisiones (por ejemplo, presencia de elemento, ubicación relativa y cercanía), evitando descripciones extensas que compitan con la atención del usuario durante el desplazamiento.
- **Retroalimentación y control del estado:** el sistema debe comunicar cambios de estado relevantes (por ejemplo, inicio/detención de asistencia, objetivo establecido, error de reconocimiento) y mantener coherencia temporal, reduciendo instrucciones redundantes o contradictorias entre instantes consecutivos.
- **Adaptabilidad:** dado que los usuarios presentan distintos niveles de discapacidad visual y distintos estilos de interacción, el diseño contempla la posibilidad de ajustar parámetros del asistente (p.ej., frecuencia de mensajes o estilo de confirmación) sin alterar el flujo general.

Estos principios establecen restricciones intencionales al diseño: se privilegia claridad por sobre exhaustividad, y control por sobre automatización completa. Esto es

coherente con el alcance definido para AIMES como sistema de **orientación inmediata** y no como navegación integral puerta-a-puerta.

4.4.2. Flujo general de interacción

El modelo de interacción se organiza como un ciclo recurrente que comienza con la activación del asistente y continúa con captura/análisis y entrega de instrucciones, manteniendo la posibilidad de interrupción o cambio de objetivo por parte del usuario. La Figura 4.3 resume este flujo general.

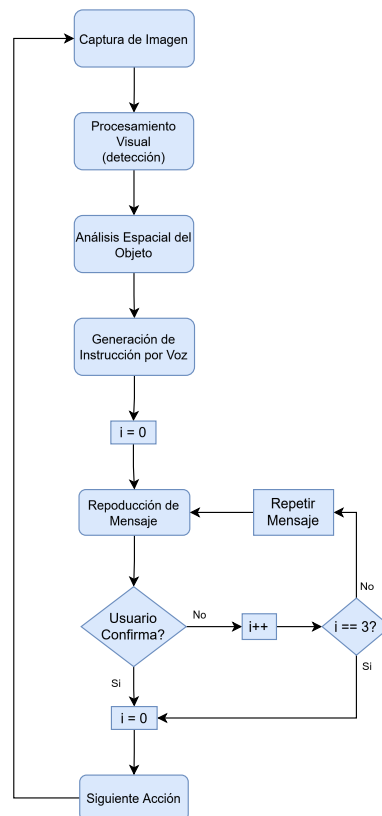


Figura 4.3: Flujo general de interacción de AIMES basado en activación, análisis y retroalimentación auditiva.

Este ciclo es relevante porque explicita que AIMES opera de manera **interactiva y orientada al contexto**: el sistema produce instrucciones a partir de la escena visual, mientras que el usuario puede ajustar o redefinir objetivos mediante voz. En consecuen-

cia, la interfaz debe funcionar como un mecanismo de coordinación entre el estado del asistente y la intención del usuario.

4.4.3. Modelo conceptual de navegación e intención

Además del flujo general, se definió un modelo conceptual secuencial que organiza la interacción en etapas: inicio, confirmación de intención y ejecución de acción. Este enfoque reduce la complejidad del control por voz, ya que limita la cantidad de estados posibles y permite confirmar acciones antes de ejecutarlas, aumentando la confianza del usuario. Las Figuras 4.4–4.5 y 4.6 representan este modelo.

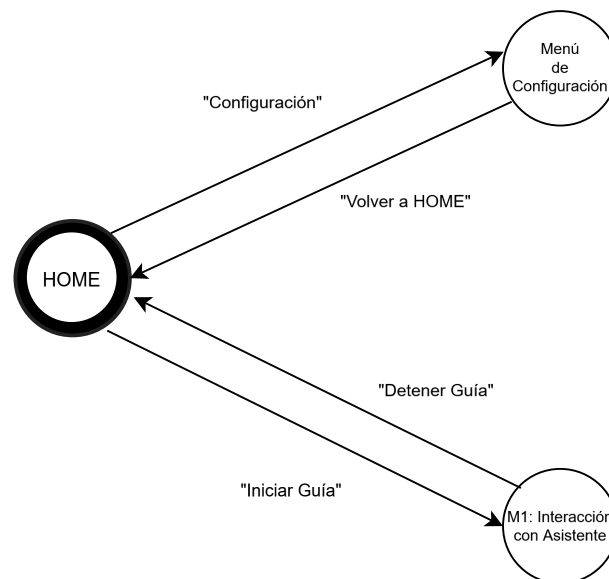


Figura 4.4: Modelo conceptual: activación del asistente y entrada al modo de guía/asistencia.

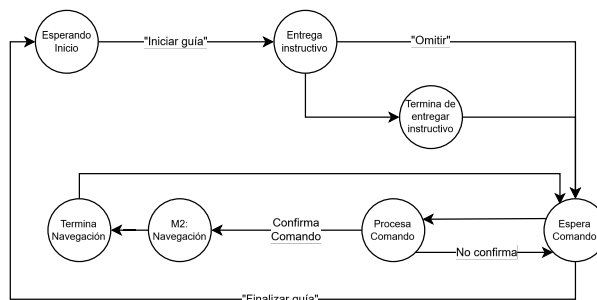


Figura 4.5: Modelo conceptual: confirmación de intención y transición a ejecución.

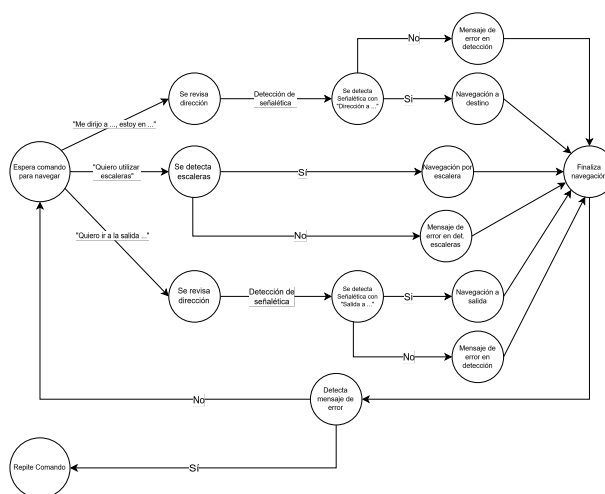


Figura 4.6: Modelo conceptual: respuesta ante ausencia de elementos detectables o fallos de contexto, incluyendo mensajes de error.

Este modelo conceptual también delimita el alcance de forma explícita: la versión descrita en esta memoria se centra en orientar mediante detección y comunicación auditiva (qué/dónde/cuán lejos), y deja como extensión futura la navegación completa que requiere localización, mapas internos y validación extensa con usuarios finales.

En conjunto, estos principios y diagramas definen la interfaz de AIMES como un **modelo de interacción por estados**, donde el asistente alterna entre escucha de intención, análisis del entorno y entrega de retroalimentación auditiva. Esta conceptualización es la base para el siguiente capítulo, donde se describe la implementación técnica: cómo estos estados se materializan en módulos software, cómo se integra el procesa-

CAPÍTULO 4 : DISEÑO DE LA SOLUCIÓN

miento visual en tiempo real, y cómo se coordina el ciclo STT/TTS para entregar una experiencia consistente.

4.4.4. Modelo real de estados de la aplicación implementada

Con el fin de sintetizar el comportamiento funcional de la aplicación desarrollada, se definió un diagrama de estados que representa el ciclo principal de operación de AIMES en su implementación móvil nativa, que, a diferencia de los diagramas conceptuales presentados previamente, se centra exclusivamente en las capacidades efectivamente alcanzadas en la aplicación: gestión de permisos, activación del asistente, captura y análisis visual en tiempo real, interpretación de objetos de interés y texto, generación de orientación auditiva y control por comandos de voz.

Como se observa en la Figura 4.7, la aplicación se organiza en un flujo cíclico donde el usuario puede activar funciones por voz, iniciar el análisis del entorno mediante cámara y recibir asistencia auditiva contextual. Este modelo también contempla eventos de excepción, como la ausencia de permisos, la falta de detecciones relevantes o la detención explícita de la guía, permitiendo representar de forma compacta el comportamiento real del sistema antes de entrar en los detalles de implementación técnica.

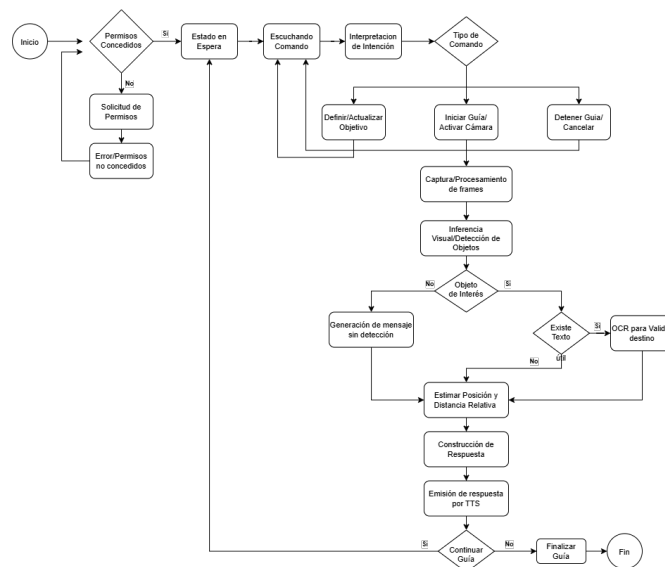


Figura 4.7: Diagrama de estados de la aplicación AIMES implementada.

El diagrama muestra que la aplicación parte de un estado de inicialización y verificación de permisos, una condición necesaria para acceder a la cámara y al audio. Una vez habilitada, la aplicación entra en un estado de espera, desde el cual puede escuchar comandos del usuario y activar la guía. Si se inicia el modo cámara, el sistema entra en un ciclo de captura, preprocesamiento e inferencia visual, por lo que, cuando se detectan elementos relevantes, estos se interpretan en términos de posición y distancia relativa; si la escena contiene texto útil, se activa OCR para apoyar la validación del destino. Finalmente, el sistema construye y emite una respuesta auditiva mediante TTS, retornando luego al estado de escucha para continuar con la asistencia o finalizar la interacción.

4.5. Síntesis del diseño y transición a la implementación

El diseño presentado en este capítulo permitió dar forma clara a la arquitectura conceptual de AIMES, definir con precisión sus módulos principales y fijar qué funciones tendría la versión móvil. A partir del análisis de requisitos, la revisión de la arquitectura y las lecciones aprendidas durante las etapas de prototipado, se llegó a una solución pensada para brindar asistencia inmediata, enfocada en detectar elementos relevantes, estimar su posición y distancia relativa, reconocer texto y ofrecer retroalimentación auditiva.

Además, la evaluación del prototipo multiplataforma mostró que el enfoque basado en Kivy y Buildozer no garantizaba un funcionamiento estable en el teléfono, aunque la aplicación pudiera compilarse e instalarse. Esta limitación, sumada a la necesidad de acceder directamente a la cámara, el audio, los permisos y el ciclo de vida del sistema operativo, llevó a optar por una implementación nativa en Android.

Con estas decisiones, se estableció una arquitectura modular formada por bloques de captura y preprocesamiento visual, detección e interpretación espacial, reconocimiento de texto e interacción por voz. Esta organización sirve como base para el capítulo siguiente, donde se describe la implementación técnica de AIMES en Android Stu-

CAPÍTULO 4 : DISEÑO DE LA SOLUCIÓN

dio con Kotlin y Jetpack Compose, explicando en detalle cómo se integran CameraX, TensorFlow Lite, SpeechRecognizer y TextToSpeech, así como los mecanismos de gestión de estado, permisos y ejecución en tiempo real.

5. Implementación de la Interfaz Nativa de AIMES en Android

Como punto de partida de este capítulo, se presenta el entorno tecnológico utilizado para implementar el prototipo final de AIMES. A diferencia de los capítulos anteriores, donde estas tecnologías fueron descritas desde una perspectiva conceptual y de selección técnica, aquí se consideran únicamente desde su rol dentro de la aplicación Android desarrollada.

La implementación final se construyó de forma nativa en Android, como respuesta a las limitaciones observadas en el prototipo multiplataforma con Kivy/Buildozer. Esta decisión permitió acceder directamente a recursos del sistema operativo, como cámara, micrófono, síntesis de voz, permisos y ciclo de vida de la aplicación, además de mejorar la trazabilidad de errores durante las pruebas en dispositivo.

5.1. Entorno tecnológico y estructura del proyecto

El stack tecnológico utilizado en la implementación final se resume en la Tabla 8. Cada herramienta cumple un rol específico dentro del flujo de AIMES, desde la construcción de la interfaz hasta la captura de imágenes, ejecución del modelo, interpretación de resultados y generación de retroalimentación auditiva.

CAPÍTULO 5 : IMPLEMENTACIÓN DE LA INTERFAZ NATIVA DE AIMES EN ANDROID

Tecnología	Uso dentro de AIMES
Android Studio	Entorno utilizado para compilar, depurar y desplegar la aplicación nativa en dispositivos Android.
Kotlin	Lenguaje principal para implementar la lógica de la aplicación, módulos visuales, orientación e interacción por voz.
Jetpack Compose	Construcción de la interfaz móvil, pantallas auxiliares, navegación y vista técnica de prueba.
CameraX	Captura de cámara, vista previa y análisis de frames en tiempo real.
TensorFlow Lite	Ejecución local del modelo de detección en formato <code>.tflite</code> .
SpeechRecognizer	Reconocimiento de comandos de voz del usuario mediante APIs nativas de Android.
TextToSpeech	Generación de instrucciones y confirmaciones auditivas.

Tabla 8: Stack tecnológico utilizado en la implementación móvil de AIMES.

En conjunto, estas tecnologías permitieron implementar un flujo móvil completo compuesto por captura de cámara, procesamiento visual, inferencia local, interpretación de resultados y retroalimentación auditiva. La elección de un entorno nativo no sólo resolvió problemas de despliegue, sino que además entregó mayor control sobre el comportamiento de la aplicación en tiempo real.

5.1.1. Dependencias principales

La implementación de AIMES requirió integrar dependencias asociadas a tres áreas principales: interfaz móvil, captura/procesamiento visual e interacción por voz. Dependencias que fueron definidas en los módulos del diseño, y que permiten materializar y conectar la lógica de la aplicación con los recursos físicos del dispositivo.

Para el desarrollo completo de la aplicación se emplearon diversos componentes ya mencionados. Como base, se utilizó principalmente Jetpack Compose para toda la interfaz visual, aprovechando distintos elementos de esta tecnología, entre ellos componentes para el diseño de pantallas, la gestión de estados visuales y la navegación entre vistas. Gracias a ello fue posible construir una interfaz funcional para el usuario y una pantalla de prueba o depuración, en la que se pueden visualizar la cámara, las detecciones y las métricas internas del sistema.

Posteriormente se añadieron las dependencias de CameraX, responsables de manejar la vista previa de la cámara y el análisis de los fotogramas. Estas bibliotecas fueron esenciales para el desarrollo de la aplicación, ya que permiten separar el flujo de visualización del flujo de procesamiento, algo clave en una app que necesita mostrar el entorno y, al mismo tiempo, ejecutar inferencias sobre los cuadros capturados.

Para la etapa de prueba de detección y entrenamiento se integró TensorFlow Lite, con el fin de ejecutar el modelo de detección directamente en el dispositivo. Esta dependencia posibilita cargar el modelo almacenado en los recursos de la aplicación, preparar los tensores de entrada y procesar la salida del modelo para obtener detecciones interpretables, en conjunto con un archivo de etiquetas asociado que sirve para mapear las salidas numéricas a clases relevantes del entorno.

Por último, la interacción por voz se implementó principalmente mediante APIs nativas de Android, lo que hizo innecesario depender de motores externos como Vosk o RHVoice en la versión final. El reconocimiento de comandos se llevó a cabo utilizando `SpeechRecognizer`, mientras que la retroalimentación auditiva se generó mediante `TextToSpeech`.

5.1.2. Estructura del proyecto

El proyecto Android se organizó siguiendo una separación funcional de responsabilidades, con el objetivo de facilitar la mantención, depuración e integración incremental de los módulos. Esta organización permite distinguir entre la interfaz de usuario, la lógica de asistencia, el procesamiento visual y la interpretación espacial de las detecciones.

A nivel general, la aplicación se estructura en torno a un punto de entrada principal, componentes de interfaz en Compose, módulos de visión y orientación, y recursos locales necesarios para la inferencia. La Figura 4.7 muestra una vista del funcionamiento del desarrollo de la aplicación, el cual en la siguiente sección se detalla a nivel de código. Para alcanzar el objetivo de funcionamiento y organización planteado, se inicia describiendo su propósito, función y las interacciones entre estos elementos; posteriormente se presenta el pipeline visual, la inferencia local, la orientación y la interacción por medio de audio, y finalmente se aborda la gestión de permisos y el ciclo de vida.

5.2. Arquitectura general de la aplicación

Esta sección presenta la arquitectura interna de la aplicación AIMES implementada en Android nativo. El objetivo es describir cómo los módulos funcionales definidos se traducen en componentes concretos de software, identificando los principales archivos de código, sus responsabilidades y la forma en que fluye la información entre ellos.

A nivel general, la aplicación se organiza bajo una estructura modular que separa la interfaz de usuario, la lógica de asistencia, el procesamiento visual, la interpretación espacial y la interacción por voz. Esta separación permite que cada componente cumpla una responsabilidad específica dentro del flujo de asistencia, facilitando la depuración, la integración incremental y la mantención del sistema.

En la Figura 5.1 se presenta la arquitectura de código de la aplicación AIMES, donde se identifican los componentes principales implementados en el proyecto Android.

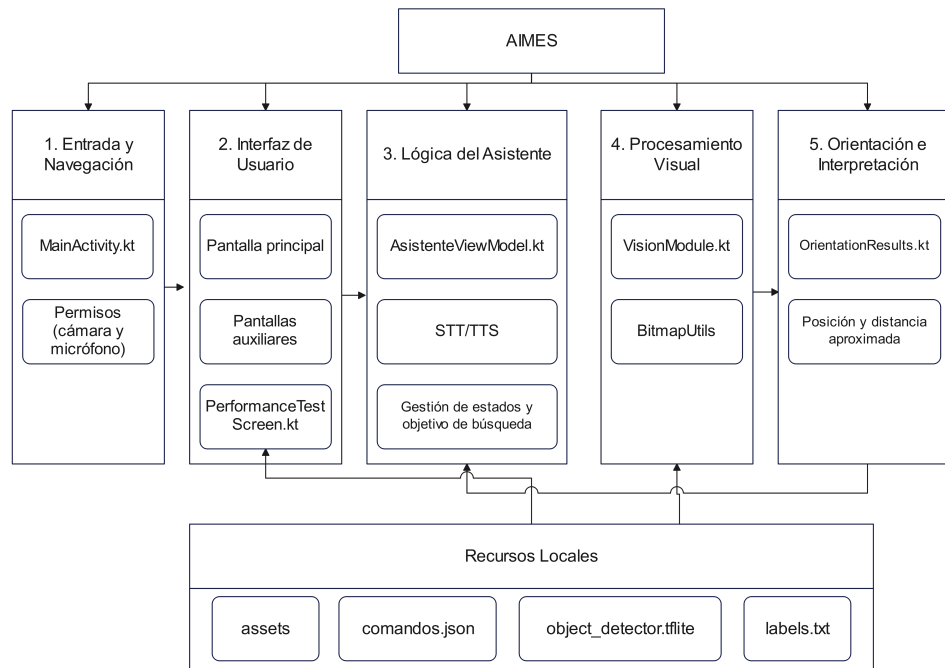


Figura 5.1: Arquitectura de código de la aplicación AIMES implementada en Android nativo.

Dentro de esta arquitectura se identifican los siguientes archivos y recursos principales:

- **MainActivity.kt:** corresponde al punto de entrada de la aplicación. En este archivo se inicializa la interfaz principal, se gestionan permisos de cámara y micrófono, y se define la navegación entre pantallas mediante Jetpack Compose. Además, coordina el acceso hacia la pantalla de prueba de cámara y orientación.
- **AsistenteViewModel.kt:** concentra la lógica principal del asistente. Este componente administra el estado de la interfaz, procesa comandos de voz, controla la activación de la guía, gestiona el objetivo de búsqueda y coordina las respuestas auditivas mediante TextToSpeech. También incorpora reglas para evitar la repetición excesiva de instrucciones.
- **VisionModule.kt:** implementa el módulo visual principal de la aplicación. Su responsabilidad es iniciar la cámara, recibir frames en tiempo real, ejecutar la

inferencia local del modelo de visión y retornar las detecciones obtenidas junto con métricas de desempeño, como FPS y confianza promedio.

- **BitmapUtils:** contiene utilidades para transformar los frames capturados por CameraX desde el formato ImageProxy hacia Bitmap. Esta conversión es necesaria para preparar las imágenes antes de enviarlas al modelo de detección.
- **OrientationResults.kt:** define las estructuras y lógica asociadas a la orientación espacial. A partir de las detecciones visuales, este componente genera resultados interpretables para el usuario, incluyendo posición relativa, distancia aproximada y selección del objeto más relevante.
- **PerformanceTestScreen.kt:** corresponde a la pantalla de prueba e integración técnica. En ella se visualiza la cámara en tiempo real, se muestran las cajas de detección, se despliegan métricas de rendimiento y se habilitan controles para probar la guía por voz y ajustar parámetros de orientación.
- **VisionAnalyzer.kt:** corresponde a un componente de análisis visual utilizado como apoyo durante el desarrollo. Su presencia permite mantener separada la lógica de análisis de imágenes y sirvió como referencia para la validación del pipeline visual.
- **Recursos locales:** la aplicación utiliza archivos almacenados en la carpeta assets, entre ellos el modelo object_detector.tflite, el archivo labels.txt con las clases detectables y el archivo comandos.json con comandos base para la interacción por voz.

Adicionalmente, la aplicación final considera la integración de un componente OCR como parte del flujo global de AIMES, orientado a reconocer texto presente en señalética del Metro y apoyar la validación de estación o destino. Sin embargo, en esta memoria se aborda únicamente desde la perspectiva de su integración funcional dentro de la arquitectura general, sin profundizar en su implementación interna.

CAPÍTULO 5 : IMPLEMENTACIÓN DE LA INTERFAZ NATIVA DE AIMES EN ANDROID

Una vez definidos los componentes principales, resulta importante describir cómo se mueve la información dentro de la aplicación, ya que este flujo determina la capacidad del sistema para entregar una respuesta oportuna al usuario. En términos generales, el proceso comienza cuando el usuario interactúa con la aplicación mediante la interfaz o un comando de voz. Luego, el sistema activa la cámara, captura frames del entorno, ejecuta el modelo de visión, interpreta los resultados y genera una respuesta auditiva.

La Figura 5.2 resume el flujo de información entre los componentes principales de la aplicación.

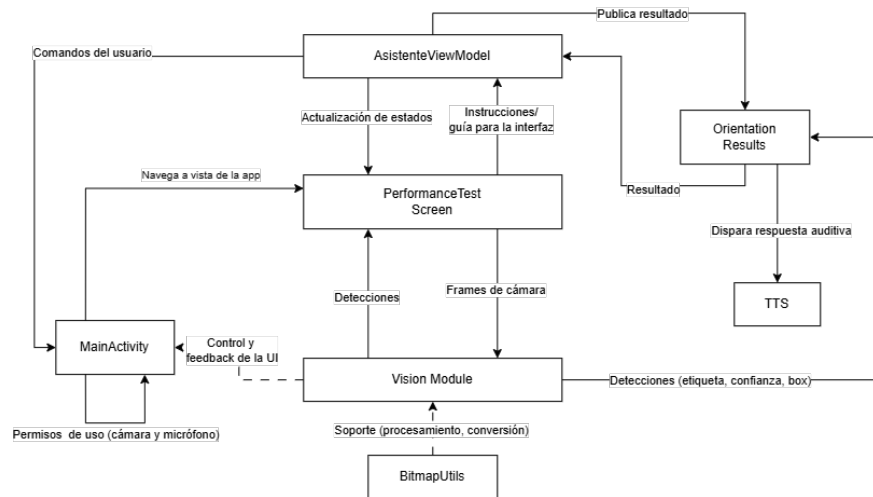


Figura 5.2: Flujo de información entre los componentes internos de la aplicación AIMES.

En este flujo, MainActivity actúa como punto de entrada y coordinador inicial de navegación y permisos. Cuando el usuario solicita iniciar la cámara o activar la guía, la aplicación navega hacia PerformanceTestScreen, donde se integra el flujo visual. Desde esta pantalla se invoca VisionModule, que recibe los frames de cámara, los preprocesa con apoyo de BitmapUtils y ejecuta la inferencia local del modelo TensorFlow Lite. Como resultado, se obtiene una lista de detecciones con etiqueta, confianza y posición de caja.

Posteriormente, dichas detecciones son enviadas al motor de orientación definido en OrientationResults.kt. Este componente transforma las salidas del modelo en

información semántica más útil para el usuario, como la ubicación relativa del elemento detectado y su distancia aproximada. Finalmente, cuando la guía por voz se encuentra activa, `AsistenteViewModel` recibe el resultado priorizado y genera una respuesta auditiva mediante `TextToSpeech`.

De esta manera, la arquitectura implementada permite pasar desde una entrada visual capturada por la cámara hasta una instrucción auditiva comprensible para el usuario.

5.3. Pipeline visual: cámara, preprocesamiento e inferencia

El pipeline visual constituye uno de los componentes centrales de la aplicación AIMES, ya que permite transformar la entrada capturada por la cámara del teléfono en detecciones interpretables por el sistema. En la implementación final, este flujo se desarrolla principalmente en el archivo `VisionModule.kt`, el cual integra la captura de video en tiempo real, el preprocesamiento de imágenes, la ejecución del modelo de visión y el postprocesamiento de resultados.

La Figura 5.3 resume el flujo implementado, desde la captura del frame hasta la obtención de detecciones finales.

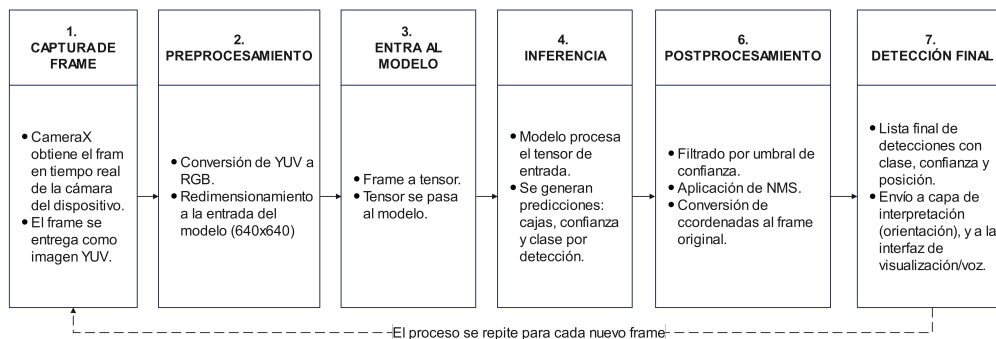


Figura 5.3: Pipeline visual implementado en AIMES: captura, preprocesamiento, inferencia y postprocesamiento de detecciones.

CAPÍTULO 5 : IMPLEMENTACIÓN DE LA INTERFAZ NATIVA DE AIMES EN ANDROID

Con el fin de complementar la explicación del pipeline visual, la Figura 5.4 presenta un ejemplo ilustrativo de las etapas de preprocesamiento y postprocesamiento. Esta figura no corresponde a una salida experimental real del modelo implementado, sino a una representación visual de apoyo para explicar el tipo de transformación que ocurre dentro del flujo: desde un frame original, pasando por su preparación para inferencia, hasta la obtención de detecciones finales luego del filtrado y la supresión no máxima.



Figura 5.4: Ejemplo ilustrativo de las etapas de preprocesamiento y postprocesamiento aplicadas en el pipeline visual de AIMES.

En este ejemplo, el preprocesamiento representa las operaciones necesarias para adaptar la imagen al formato esperado por el modelo, incluyendo corrección de orientación y redimensionamiento. Por su parte, el postprocesamiento representa la etapa en que las salidas del modelo son filtradas por confianza, asociadas a etiquetas y cajas delimitadoras, y depuradas mediante supresión no máxima para obtener una lista final de detecciones.

A partir de este flujo, la primera etapa implementada corresponde a la captura de cámara, la cual se implementó utilizando CameraX, librería nativa de Android que permite administrar el acceso a la cámara respetando el ciclo de vida de la aplicación. Dentro de este flujo se configuran dos componentes principales: Preview, utilizado para mostrar la vista previa de la cámara en pantalla, e ImageAnalysis, encargado de

CAPÍTULO 5 : IMPLEMENTACIÓN DE LA INTERFAZ NATIVA DE AIMES EN ANDROID

entregar frames al sistema para su procesamiento. Adicionalmente, para evitar acumulación de imágenes y reducir la latencia, el analizador fue configurado con la estrategia `STRATEGY_KEEP_ONLY_LATEST`, lo que permite procesar únicamente el frame más reciente disponible.

Cada frame recibido por el analizador se obtiene inicialmente como un objeto `ImageProxy`. Dado que el modelo de visión requiere una entrada en formato de imagen procesable, se implementó la utilidad `BitmapUtils`, encargada de convertir el frame capturado a `Bitmap`. Durante esta conversión se considera el valor de rotación entregado por `CameraX` mediante `imageInfo.rotationDegrees`, el cual indica la orientación con la que fue capturada la imagen respecto al dispositivo. A partir de este valor, se aplica una transformación de rotación sobre el `Bitmap`, de modo que la imagen utilizada por el modelo mantenga la orientación correcta antes del redimensionamiento e inferencia.

Posteriormente, la imagen se redimensiona al tamaño de entrada esperado por el modelo, el cual es de: 640x640 píxeles.

La inferencia se ejecuta localmente mediante `TensorFlow Lite`, cargando el modelo `object_detector.tflite` desde los recursos locales de la aplicación. Asimismo, se utiliza el archivo `labels.txt` para asociar las salidas numéricas del modelo con las clases entrenadas, tales como ascensor, entrada, escaleras (normales o mecánicas), estación, piso podotáctil y señalética del Metro. Al ejecutarse en el propio dispositivo, el sistema mantiene el enfoque de operación local definido para AIMES, evitando depender de conectividad externa durante el uso.

Una vez obtenida la salida del modelo, el sistema realiza una etapa de postprocesamiento orientada a transformar los valores crudos de inferencia en detecciones utilizables por la aplicación. En este proceso se descartan aquellas predicciones cuya confianza es inferior al umbral definido en el código, correspondiente a 0,70. Luego, a cada detección válida se le asigna una etiqueta a partir del archivo `labels.txt` y se construye su caja delimitadora dentro del espacio de entrada del modelo. Para re-

CAPÍTULO 5 : IMPLEMENTACIÓN DE LA INTERFAZ NATIVA DE AIMES EN ANDROID

ducir detecciones repetidas sobre un mismo elemento, se aplica supresión no máxima (*Non-Maximum Suppression*, NMS), utilizando un umbral de intersección sobre unión (*IoU*) de 0,45. Finalmente, las detecciones resultantes se ordenan de mayor a menor confianza y se limita la salida a un máximo de 50 resultados, obteniendo así una lista compuesta por etiqueta, confianza y coordenadas de caja.

Junto con las detecciones, el pipeline visual calcula métricas internas de funcionamiento que permiten observar el comportamiento del sistema durante las pruebas en dispositivo. En particular, se estima la tasa de cuadros procesados por segundo (FPS) y la confianza promedio de las detecciones generadas. El cálculo matemático se define así:

El cálculo de FPS se realiza contando la cantidad de frames procesados dentro de una ventana temporal y normalizando dicho valor a un segundo. Matemáticamente, esta métrica se expresa como:

$$FPS = \frac{N_{frames} \cdot 1000}{t_{actual} - t_{anterior}}$$

donde N_{frames} corresponde al número de frames procesados desde la última actualización, y $t_{actual} - t_{anterior}$ representa el tiempo transcurrido en milisegundos. En la implementación, esta métrica se actualiza cuando ha transcurrido al menos un segundo desde la medición anterior.

Por otra parte, la confianza promedio se calcula a partir de los puntajes de confianza de las detecciones obtenidas en un frame determinado. Si existen n detecciones válidas, cada una con confianza s_i , la métrica se define como:

$$Confianza_{promedio} = \frac{1}{n} \sum_{i=1}^n s_i \cdot 100$$

En caso de que no existan detecciones válidas, la confianza promedio se define como 0. Estas métricas son enviadas a la interfaz de prueba, donde permiten visualizar el desempeño del pipeline en tiempo real y apoyar la depuración de la aplicación durante

las pruebas con cámara activa.

5.4. Interpretación de resultados y orientación

Las detecciones generadas por el modelo de visión no son comunicadas directamente al usuario, ya que una etiqueta y una caja de detección no constituyen por sí solas una instrucción accesible. Por esta razón, AIMES incorpora una etapa de interpretación espacial encargada de transformar los resultados visuales en información semántica breve y comprensible. Esta lógica se implementa principalmente en `OrientationResults.kt`, donde se define el motor de orientación utilizado por la aplicación.

La Figura 5.5 presenta el flujo de interpretación aplicado sobre las detecciones generadas por el modelo.

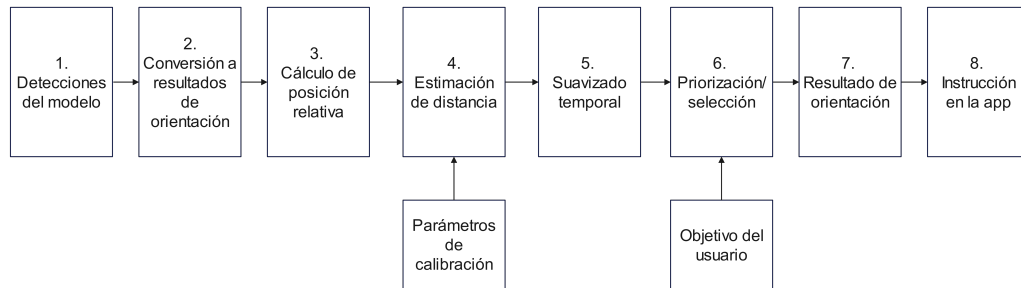


Figura 5.5: Flujo de interpretación espacial de detecciones para generación de orientación en AIMES.

El objetivo de esta etapa es responder a tres preguntas fundamentales para el usuario: qué elemento fue detectado, dónde se encuentra respecto a la cámara y qué tan cerca está. Para esto, cada detección se transforma en un resultado de orientación que contiene la etiqueta del objeto, la confianza del modelo, la posición relativa, la distancia estimada y la caja asociada.

La posición relativa se determina a partir del centro horizontal de la caja de detección. Para simplificar la comunicación auditiva, el frame se divide en zonas laterales

CAPÍTULO 5 : IMPLEMENTACIÓN DE LA INTERFAZ NATIVA DE AIMES EN ANDROID

y centrales, permitiendo clasificar cada elemento como ubicado a la izquierda, centro-izquierda, centro, centro-derecha o derecha. Posteriormente, estas categorías se traducen a expresiones más naturales para el usuario, como: “a la izquierda”, “al frente” o “a la derecha”.

Es importante señalar que esta estimación espacial se realiza sobre la imagen ya normalizada por el pipeline visual. Es decir, antes de calcular la posición relativa de una detección, el frame capturado por la cámara ya fue convertido y rotado según la orientación reportada por CameraX, por lo que si esta corrección no se aplicara, las cajas de detección podrían quedar desalineadas respecto a la escena observada por el usuario, provocando errores en la interpretación espacial, por ejemplo, indicar un objeto a la izquierda cuando visualmente corresponde a otra zona del encuadre.

La distancia se estima de forma relativa mediante una heurística geométrica basada en la ubicación vertical de la caja dentro del frame. En términos generales, un objeto cuya caja aparece más cercana a la parte inferior de la imagen se interpreta como más próximo al usuario y para controlar esta estimación, se utiliza un parámetro calibrable `povK`, junto con umbrales de cercanía y distancia media. Con esto, el sistema clasifica los elementos detectados en categorías simples como cerca, a media distancia o lejos.

Dado que las detecciones pueden variar entre frames consecutivos, se incorpora un mecanismo de suavizado mediante promedio móvil exponencial. El cual permite reducir cambios bruscos en la estimación de distancia y entregar instrucciones más estables, evitando que el sistema alterne rápidamente entre categorías cuando el objeto se mantiene en una posición similar.

Otro aspecto importante es la elección del objeto prioritario indicado por el usuario. En la implementación, este se establece cuando el usuario define un objetivo concreto; a continuación, la etiqueta objetivo asociada al comando, representada internamente como `targetLabel`, se transforma a minúsculas y se eliminan los espacios sobrantes antes de comparar con las etiquetas de las detecciones generadas por el modelo. Después, el motor de orientación filtra la lista de resultados para conservar únicamente

CAPÍTULO 5 : IMPLEMENTACIÓN DE LA INTERFAZ NATIVA DE AIMES EN ANDROID

aquellas detecciones cuya etiqueta coincida de forma exacta con el objetivo especificado. En caso de no encontrarse ninguna coincidencia, el sistema no selecciona objetos alternativos, evitando así proporcionar instrucciones sobre elementos que el usuario no ha solicitado.

Cuando no existe un objetivo definido, el sistema evalúa todas las detecciones disponibles y selecciona aquella con mayor prioridad mediante una función de puntaje. Esta función utiliza la distancia como criterio principal y la confianza de la detección como un ajuste secundario. Para evitar que la confianza domine sobre la cercanía del objeto, esta se pondera mediante un factor pequeño, de modo que su efecto sea utilizado principalmente como criterio de desempate entre detecciones de distancia similar. Cuando existe una distancia estimada en metros, el puntaje utilizado puede expresarse como:

$$score = d_{metros} - (confianza \cdot 0,01)$$

donde d_{metros} corresponde a la distancia estimada del objeto, $confianza$ representa el puntaje entregado por el modelo de detección, y α corresponde a un factor de ponderación definido en la implementación. En este caso, se utiliza $\alpha = 0,01$, por lo que la cercanía del objeto mantiene mayor peso en la selección final.

En caso de no contar con una estimación en metros, se utiliza la categoría de distancia como criterio base, asignando menor valor a los objetos más cercanos:

$$score = bucket - \alpha \cdot confianza$$

donde $bucket = 0$ corresponde a objetos cercanos, $bucket = 1$ a objetos a media distancia y $bucket = 2$ a objetos lejanos. Al seleccionarse el resultado con menor puntaje, la estrategia prioriza detecciones cercanas y utiliza la confianza únicamente como ajuste secundario. De esta forma, no se busca comparar directamente magnitudes físicas con probabilidades, sino construir una regla práctica de priorización para decidir qué detección comunicar al usuario.

CAPÍTULO 5 : IMPLEMENTACIÓN DE LA INTERFAZ NATIVA DE AIMES EN ANDROID

Finalmente, el resultado seleccionado se transforma en una frase breve mediante una función de formateo para voz, es decir: una detección puede convertirse en una instrucción del tipo: “piso podotáctil al frente, cerca” o “escalera mecánica a la derecha, a media distancia”. De esta manera, AIMES traduce la salida del modelo de visión en una indicación auditiva orientada a la toma de decisiones durante el desplazamiento.

La interpretación de resultados cumple, por tanto, un rol intermedio entre la visión por computador y la interacción por voz, es decir, mientras el pipeline visual identifica elementos del entorno, el motor de orientación los convierte en información accionable. Esta separación permite que el sistema mantenga una arquitectura modular y facilita futuras mejoras, como nuevos criterios de priorización, calibración por dispositivo o incorporación de información contextual adicional.

5.5. Reconocimiento de texto y validación de destino

Como complemento al flujo principal de detección y orientación, la aplicación integra un componente de reconocimiento óptico de caracteres (OCR) orientado a extraer texto desde señalética visible del Metro. En esta memoria, dicho componente se aborda desde la perspectiva de su integración funcional dentro de la aplicación móvil, sin profundizar en su implementación interna.

El proceso se apoya en las detecciones generadas por `VisionModule`, donde cada elemento detectado se representa mediante una caja delimitadora o *bounding box*, la cual permite ubicar el objeto dentro de la imagen. Cuando la etiqueta asociada a una detección corresponde a señalética, el sistema puede utilizar dicha región de interés para aplicar OCR sobre el área detectada, evitando procesar el frame completo.

El texto obtenido mediante OCR se utiliza posteriormente como información de contexto para apoyar la orientación del usuario. En particular, este componente permite complementar la lógica de validación de destino cuando la señalética observada contiene información relacionada con estaciones o direcciones de viaje.

Para validar el destino del usuario, el sistema utiliza la información entregada inicialmente mediante comandos de voz. En este proceso, las estaciones se almacenan en dos variables principales: `estacion_actual` y `estacion_destino`. La primera corresponde a la estación identificada como ubicación actual del usuario, mientras que la segunda corresponde a la estación que el usuario indicó como destino al iniciar la guía.

La lógica de validación se basa en una lista ordenada de estaciones. Por ejemplo, si el usuario se encuentra en Ñuñoa y su destino corresponde a Los Leones, el índice del destino es superior al índice de la estación actual, y por lo tanto, el sistema determina que el desplazamiento debe realizarse en dirección a Los Leones. En cambio, si el usuario se encuentra en Ñuñoa y desea llegar a Cerrillos, el índice del destino es inferior al de la estación actual, por lo que la dirección indicada será Cerrillos.

A partir de esta lógica interna, el sistema puede entregar mensajes simples al usua-

rio, tales como:

- **Estación no encontrada:** cuando alguna de las estaciones entregadas no se encuentra en la lista considerada por el sistema.
- **Ya estás en la estación de destino:** cuando `estacion_actual` coincide con `estacion_destino`.
- **Debes ir en dirección a:** cuando la estación de destino es distinta de la estación actual y el sistema puede determinar la dirección correspondiente.

De esta forma, el OCR funciona como una capa complementaria de validación contextual, apoyando la orientación cuando existe señalética visible, pero sin reemplazar el flujo principal basado en detección, interpretación espacial y retroalimentación auditiva.

5.6. Interacción por voz y retroalimentación auditiva

La interacción por voz se centra principalmente en el archivo `AsistenteViewModel`, mientras que el archivo `OrientationResults` procesa la información espacial obtenida a partir de las detecciones visuales para generar la retroalimentación dirigida al usuario.

5.6.1. Interacción por Voz

El archivo `AsistenteViewModel` utiliza la API de `SpeechRecognizer` de Android para implementar la función de reconocimiento automático del habla (STT). Cuando el usuario activa el micrófono desde la interfaz principal, `MainActivity` verifica previamente que la aplicación posea el permiso para utilizar el micrófono y posteriormente ejecuta el método `startListening` del `AsistenteViewModel`.

Una vez iniciado el reconocimiento, `SpeechRecognizer` transforma la entrada de voz en texto. El sistema considera tanto resultados parciales como resultados finales,

permitiendo actualizar el texto reconocido mientras el usuario habla. Cuando se obtiene el resultado final, el sistema selecciona la hipótesis que presenta el mayor nivel de confianza entre las alternativas entregadas por el reconocedor y posteriormente la envía al método `procesarComando`.

El método `procesarComando` se encarga de interpretar el texto reconocido y determinar la intención del usuario. Para ello, normaliza el texto y lo compara con un conjunto de comandos definidos en el archivo `comandos.json`. Entre las acciones disponibles se encuentran:

- búsqueda de objetos, como ascensores o salida.
- indicación de la estación actual.
- selección de una estación de destino.

Por ejemplo, una instrucción como *“buscar ascensor”* permite identificar la intención *buscar_objeto* y establecer *ascensor* como el objeto que debe ser localizado. De forma similar, expresiones como *“estoy en Franklin”* permiten establecer la estación actual, mientras que una instrucción como *“me dirijo a Ñuñoa”* permite establecer la estación de destino.

5.6.2. Retroalimentación Auditiva

La información espacial generada en `OrientationResults.kt` incluye elementos como la clase del objeto detectado, su nivel de confianza, su posición relativa y su distancia aproximada. Esta información se almacena en estructuras de tipo `OrientationResult`, permitiendo transformar las detecciones visuales en información comprensible para el usuario.

Posteriormente, una función de formateo para voz transforma estos resultados en una descripción textual destinada a la síntesis auditiva. La posición se convierte en expresiones como *“a la izquierda”*, *“al frente”* o *“a la derecha”*, mientras que la

distancia se expresa mediante categorías como “*cerca*”, “*a media distancia*” o “*lejos*”. Por ejemplo, una detección puede generar una instrucción como “*escalera mecánica a la derecha, cerca*” o “*piso podotáctil al frente, a media distancia*”.

Finalmente, las instrucciones son convertidas en mensajes auditivos mediante `TextToSpeech`, componente que permite entregar retroalimentación al usuario sin depender de una interfaz visual, manteniendo la coherencia con el enfoque de accesibilidad definido para AIMES.

Con el objetivo de evitar una reproducción excesiva de instrucciones, el sistema incorpora mecanismos de control de repetición; la función `speakOnce()` evita reproducir nuevamente un mensaje idéntico al anterior. Además, durante el procesamiento mediante OCR se establece un intervalo mínimo entre ejecuciones consecutivas, mientras que para las instrucciones de navegación se almacena el último tipo de dirección comunicado, evitando repetir una misma indicación mientras esta no cambie.

En conjunto, la interacción por voz y la retroalimentación auditiva permiten cerrar el ciclo de asistencia de AIMES: el usuario entrega comandos hablados, el sistema interpreta la intención, procesa el entorno visual y comunica una respuesta breve y accionable mediante audio.

5.7. Interfaz móvil

La interfaz móvil de AIMES fue implementada en Android nativo utilizando Jetpack Compose y su desarrollo toma como punto de partida el diseño conceptual definido previamente a nivel de proyecto, el cual priorizaba una experiencia accesible mediante interacción auditiva y por voz. Sin embargo, al llevar dicho diseño a una aplicación funcional, fue necesario adaptar algunos elementos para responder a las restricciones reales de implementación, validación técnica y operación en dispositivo móvil.

La interfaz implementada toma como base los lineamientos definidos durante el diseño conceptual del proyecto, los cuales eran: criterios de accesibilidad, uso de boto-

CAPÍTULO 5 : IMPLEMENTACIÓN DE LA INTERFAZ NATIVA DE AIMES EN ANDROID

nes simples y visibles, navegación mediante comandos de voz, y pantallas asociadas a funciones como ayuda, configuración, alerta y acceso al modo de asistencia.

Durante la implementación, algunas decisiones visuales y funcionales fueron ajustadas para adaptarse al comportamiento real de la aplicación en dispositivo. No obstante, los elementos conceptuales principales fueron conservados: interacción simple, apoyo auditivo, acceso directo a funciones clave y una vista técnica que permitiera validar el funcionamiento del sistema.

La Figura 5.6 presenta una referencia del diseño conceptual considerado durante el desarrollo que sirvió como guía inicial para definir la organización general de la aplicación.



Figura 5.6: *Diseño conceptual de la interfaz móvil AIMES utilizado como referencia inicial.*

A partir de esta referencia, la aplicación implementada se estructuró en una pantalla principal y pantallas auxiliares, donde la pantalla principal permite iniciar la interacción con el asistente, acceder al micrófono y entrar al modo cámara. Esta vista actúa como punto de entrada para el usuario y se vincula con la lógica de permisos, navegación y

CAPÍTULO 5 : IMPLEMENTACIÓN DE LA INTERFAZ NATIVA DE AIMES EN ANDROID

activación de funcionalidades. Por otro lado, las pantallas auxiliares funcionan como ayuda, configuración de la aplicación y alerta.

Adicionalmente, se implementó una paleta de colores especial, que se fundamentó en la necesidad de asegurar contrastes superiores a 4.5:1, recomendados por WCAG[36], y en evitar combinaciones problemáticas para personas con daltonismo.

A continuación, se muestra en la Figura 5.7 la paleta de colores utilizada junto a sus respectivos códigos.



Figura 5.7: Paleta de colores de la interfaz gráfica AIMES

CAPÍTULO 5 : IMPLEMENTACIÓN DE LA INTERFAZ NATIVA DE AIMES EN ANDROID

Tomando todo lo anterior, en la Figura 5.8 se muestra la pantalla principal implementada en la aplicación Android.

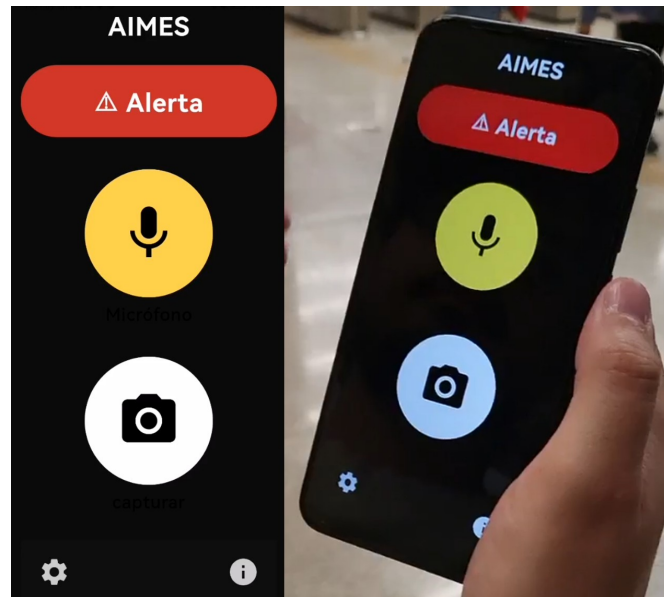


Figura 5.8: Pantalla principal implementada en la aplicación móvil AIMES.

El elemento más relevante de la interfaz implementada corresponde a la pantalla de prueba y validación técnica, desarrollada en `PerformanceTestScreen.kt`, que permite tener una vista integrada de la cámara en tiempo real, el overlay de detecciones, las métricas del pipeline visual y controles para activar la guía por voz. A diferencia de las pantallas orientadas al usuario final, esta vista cumple principalmente un rol de depuración, ya que permite observar directamente el comportamiento interno del sistema durante las pruebas en dispositivo.

El comportamiento interno se refiere a que dentro de esta pantalla se muestran las distintas métricas utilizadas para evaluar si la aplicación mantiene un comportamiento adecuado durante la ejecución. Estas métricas son: los FPS y la confianza promedio.

CAPÍTULO 5 : IMPLEMENTACIÓN DE LA INTERFAZ NATIVA DE AIMES EN ANDROID

La Figura 5.9 presenta la pantalla de cámara y prueba implementada en la aplicación.

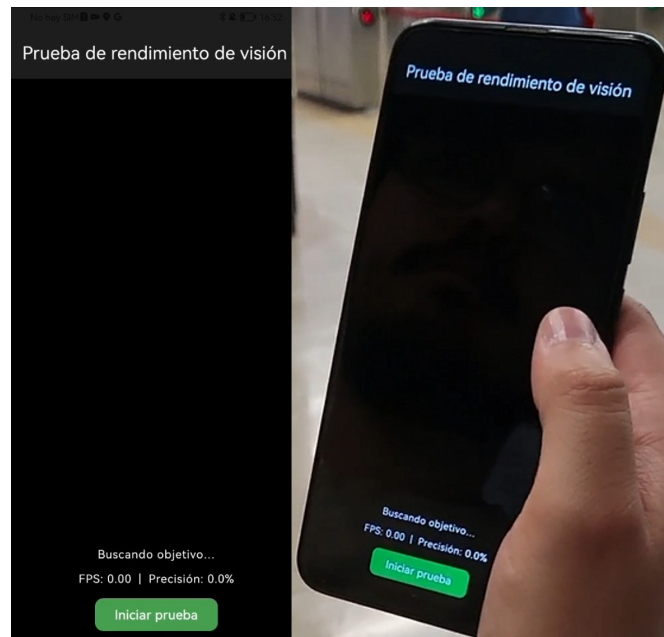


Figura 5.9: *Pantalla de cámara y pruebas técnicas implementada en AIMES.*

Desde el punto de vista funcional, esta interfaz permite cerrar el ciclo de integración de AIMES: el usuario activa la cámara, el sistema procesa la escena, obtiene detecciones, interpreta la posición del objeto y puede emitir una respuesta auditiva. En paralelo, la vista técnica permite validar que el pipeline visual funcione correctamente, observar las clases detectadas y verificar el comportamiento de las métricas durante la ejecución.

CAPÍTULO 5 : IMPLEMENTACIÓN DE LA INTERFAZ NATIVA DE AIMES EN ANDROID

La Figura 5.10 muestra un ejemplo de la aplicación en funcionamiento, con detecciones visibles sobre la imagen de cámara.

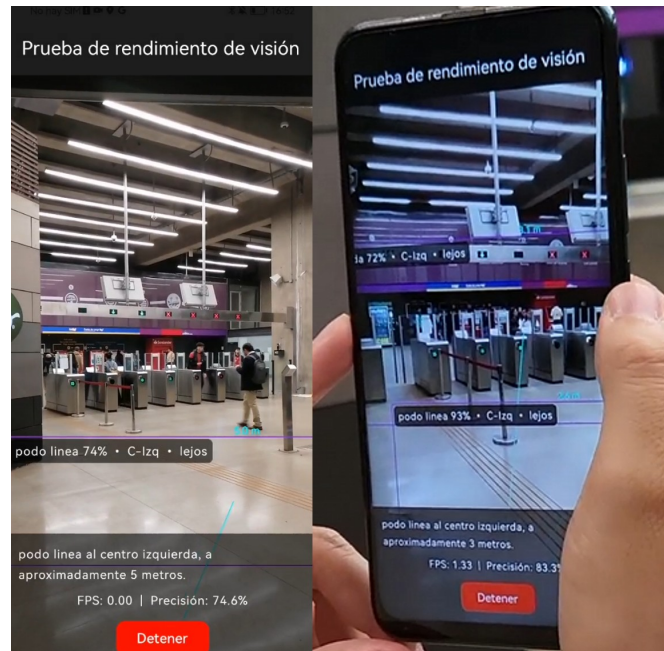


Figura 5.10: Funcionalidad de captura de imágenes en estado activo con visualización de métricas.

Además de la vista de cámara y métricas, la aplicación incorpora pantallas auxiliares que complementan la experiencia de uso y permiten proyectar funcionalidades de soporte para versiones futuras. Entre ellas se encuentra la pantalla de instrucciones mostrada en la Figura 5.11, cuyo propósito es orientar al usuario sobre el funcionamiento básico de AIMES, indicando el uso de los botones principales, su ubicación y las acciones disponibles dentro de la aplicación.

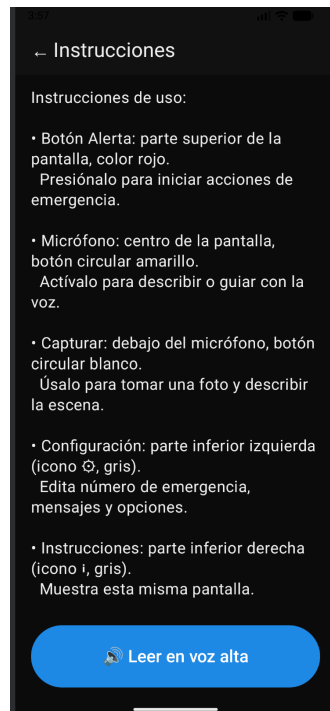


Figura 5.11: Pantalla de instrucciones implementada en AIMES.

También se implementó una pantalla de configuración mostrada en la Figura 5.12, orientada principalmente a una persona de apoyo o cuidador, permitiendo definir parámetros asociados al uso de la aplicación como: opciones de accesibilidad, tamaño de texto, uso de interacción por voz y retroalimentación mediante vibración. Estas dos últimas opciones quedaron contempladas a nivel de interfaz, pero su integración completa se mantiene como trabajo futuro. Asimismo, la pantalla permite configurar un contacto de emergencia y un mensaje predefinido, los cuales pueden ser utilizados posteriormente desde la funcionalidad de alerta.

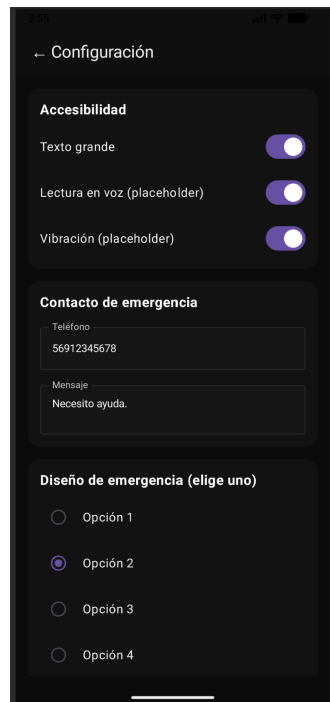


Figura 5.12: Pantalla de configuración implementada en AIMES.

Dentro de esta misma lógica, se propusieron cuatro alternativas conceptuales para el botón de alerta como se puede ver en la Figura 5.13. La inclusión de estas variantes tuvo un propósito exploratorio, permitiendo disponer de distintas opciones de interacción para evaluar posteriormente cuál resulta más clara, cómoda y accesible para el usuario. Finalmente, la pantalla asociada al botón de alerta mostrada en la Figura 5.14, despliega accesos rápidos para contactar a la persona registrada como contacto de emergencia o enviar un mensaje predefinido configurado previamente.

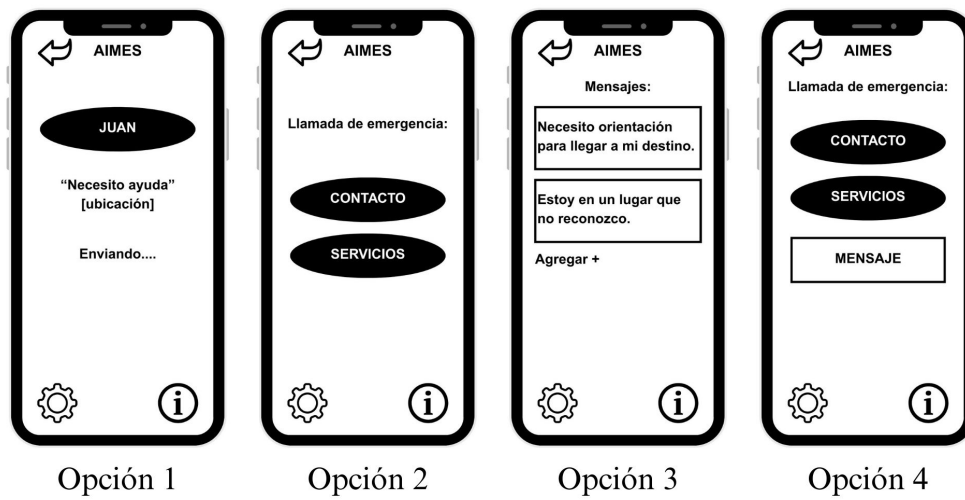


Figura 5.13: Opciones implementadas para el diseño del botón de alerta.

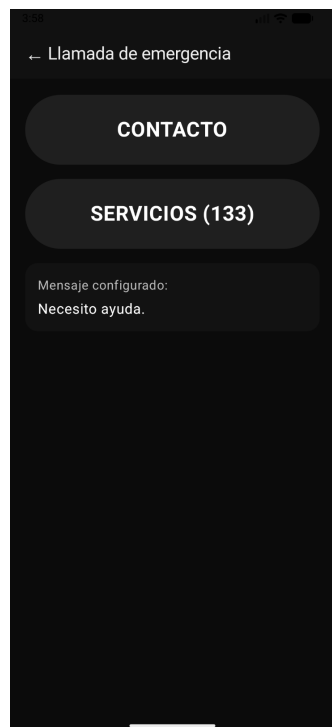


Figura 5.14: Pantalla asociada al botón de alerta implementada en AIMES.

En conjunto, estas pantallas auxiliares complementan el flujo principal de AIMES al ofrecer soporte, configuración y una base para acciones de emergencia. Aunque el

núcleo funcional de esta memoria se concentra en la captura de cámara, inferencia visual, orientación e interacción auditiva, estas vistas permiten estructurar la aplicación como una solución móvil más completa y preparada para futuras iteraciones centradas en accesibilidad, personalización y seguridad del usuario.

La interfaz móvil implementada cumple un rol doble dentro del prototipo. En primer lugar, permite al usuario acceder de manera simple a las funciones principales de AIMES, como: la activación del asistente, el uso del micrófono y el ingreso al modo cámara. En segundo lugar, funciona como una herramienta de apoyo al desarrollo, ya que permite observar en tiempo real el comportamiento de los módulos internos mediante la visualización de la cámara, las detecciones, las métricas de rendimiento y los controles de prueba asociados a la orientación y la guía por voz.

Esta decisión permitió validar en el dispositivo móvil el flujo principal de AIMES: cámara, inferencia, orientación y retroalimentación auditiva. Así, la interfaz implementada funciona tanto como capa de interacción como herramienta técnica para futuras mejoras centradas en el usuario.

5.8. Manejo de permisos y ciclo de vida

5.8.1. Manejo de Permisos y recursos

El uso de los recursos del dispositivo se controla a través del sistema de permisos de Android, como el permiso `RECORD_AUDIO`, el cual es necesario para acceder al micrófono. En `MainActivity`, antes de iniciar el reconocimiento de voz, se verifica si dicho permiso ha sido concedido mediante `ContextCompat.checkSelfPermission`.

Si el permiso no se encuentra disponible, la aplicación solicita autorización al usuario mediante `RequestPermission`. El reconocimiento de voz solamente se inicia después de que el permiso haya sido concedido.

El acceso a la cámara sigue el mismo principio de seguridad mencionado. La utilización de `CameraX` permite gestionar el acceso a la cámara mediante componentes

compatibles con el ciclo de vida de la actividad, evitando mantener recursos de captura activos cuando la pantalla correspondiente deja de estar disponible.

Además de la gestión de permisos, la aplicación incorpora mecanismos para controlar el uso de recursos durante la ejecución. Por ejemplo, el procesamiento visual utiliza imágenes de dimensiones controladas y, cuando se requiere reconocimiento de texto, el OCR se aplica sobre regiones acotadas de la imagen asociadas a señalética detectada, evitando analizar el frame completo. Asimismo, la variable `ocrBusy` impide la ejecución simultánea de procesos OCR.

En la interacción auditiva, la función `speakOnce()` evita la reproducción repetitiva de una misma instrucción, mientras que el procesamiento de navegación incorpora un intervalo mínimo entre ejecuciones consecutivas. Por otro lado, la variable `continuousListening` permite controlar la ejecución continua del reconocimiento de voz, evitando mantener activa la escucha en situaciones donde no corresponde.

5.8.2. Ciclo de vida de la Aplicación

La arquitectura utiliza componentes de Android que permiten asociar las funcionalidades principales al ciclo de vida de la aplicación. En particular, `AsistenteViewModel` extiende `AndroidViewModel`, lo que permite mantener el estado asociado al asistente de manera independiente de la reconstrucción de la interfaz de usuario.

La instancia del *ViewModel* se obtiene mediante `by viewModels()` en `MainActivity`, por lo que, de esta forma, el estado relacionado con el reconocimiento de voz, las respuestas del asistente, el objetivo de búsqueda y la guía activa no depende directamente de la reconstrucción de los componentes visuales de la interfaz.

Al finalizar el ciclo de vida del *ViewModel*, se ejecuta `onCleared()`, donde se liberan explícitamente los recursos utilizados por el reconocimiento y la síntesis de voz. Aquí, se detiene y libera el motor `TextToSpeech`, en el que se destruye la instancia de `SpeechRecognizer` y se eliminan las tareas pendientes del `Handler`. Esto evita mantener procesos activos cuando el componente ya no es necesario y contribuye a un

comportamiento más estable de la aplicación.

5.9. Síntesis de la implementación

A lo largo de este capítulo se describió la implementación técnica de AIMES como una aplicación móvil nativa para Android, en la que, a partir de la arquitectura conceptual previamente definida, los módulos del sistema se concretaron en componentes específicos de software, estructurados en torno a la interfaz de usuario, la lógica del asistente, el procesamiento visual, la interpretación espacial, la interacción por voz y la gestión de los recursos del dispositivo.

La implementación permitió consolidar un flujo funcional extremo a extremo. Primero, la aplicación gestiona permisos y navegación desde `MainActivity`, permitiendo acceder a las funciones principales del sistema. Luego, mediante `VisionModule`, se activa la cámara del dispositivo, se procesan frames en tiempo real y se ejecuta inferencia local con el modelo de detección. Las salidas visuales obtenidas son posteriormente transformadas por el motor de orientación en información semántica, como posición relativa, distancia aproximada y selección del objeto prioritario. Finalmente, esta información puede ser comunicada al usuario mediante retroalimentación auditiva utilizando las APIs nativas de voz de Android.

Además del flujo visual y auditivo principal, la aplicación integra elementos complementarios relevantes para el funcionamiento global de AIMES, como el reconocimiento de texto para apoyar la validación de destino, pantallas auxiliares de ayuda, configuración y alerta, y una vista técnica de prueba que permite observar detecciones, métricas y comportamiento interno del sistema. Esta última fue muy importante durante el desarrollo, ya que permitió validar directamente en el dispositivo móvil la integración entre cámara, inferencia, orientación y respuesta auditiva.

En términos de aporte técnico, la implementación nativa permitió superar las limitaciones observadas en el prototipo multiplataforma anterior, entregando mayor control sobre permisos, ciclo de vida, recursos de cámara, audio y ejecución en tiempo real.

CAPÍTULO 5 : IMPLEMENTACIÓN DE LA INTERFAZ NATIVA DE AIMES EN ANDROID

Asimismo, la separación entre componentes favoreció la depuración del sistema y dejó una base más mantenible para futuras iteraciones.

En síntesis, el resultado de este capítulo es una aplicación Android funcional que integra los componentes principales requeridos para la orientación inmediata propuesta por AIMES. Esta implementación constituye la base sobre la cual se analizan, en el capítulo siguiente, los resultados obtenidos a partir del dataset, los modelos de visión y la ejecución del prototipo móvil en escenarios de prueba.

6. Resultados

6.1. Contexto General del Prototipo

El prototipo de AIMES, considerando los módulos descritos en el Capítulo 5, se estructura a partir de tres ejes principales de desarrollo, los cuales permiten abordar de manera integral el diseño, implementación y validación del sistema:

- **Confección del *Dataset*:** orientada a la recopilación y preparación de datos visuales representativos del entorno del Metro de Santiago.
- **Entrenamiento de Modelos de *Computer Vision*:** enfocado en modelos de detección y segmentación de objetos y señalética relevante.
- **Desarrollo de la Aplicación Móvil:** encargada de la integración de los modelos, la interacción con el usuario y la entrega de asistencia en tiempo real.

En particular, se consideran modelos de detección basados en la arquitectura YOLOv11n, con el objetivo de realizar un aporte exploratorio mediante el análisis de variaciones y ajustes del modelo que permitan optimizar su desempeño en el contexto específico del proyecto.

Por otro lado, la integración del modelo de segmentación YOLOv11n-segmentation se justifica por el uso de las máscaras generadas por el modelo, las cuales permiten refinar el algoritmo de orientación. Estas máscaras facilitan una interpretación más precisa de las detecciones realizadas en los entornos de prueba, contribuyendo a una guía más efectiva del usuario durante su desplazamiento.

6.1.1. Confección de Dataset

Mediante la toma y compilación de imágenes de la infraestructura de Metro de Santiago (cuyo proceso de adquisición se detalla en la Sección 4.2.1) se construyen datasets que son almacenados en la plataforma de Roboflow.

CAPÍTULO 6 : RESULTADOS

La construcción del dataset se desarrolló de manera iterativa mediante procesos de incorporación, etiquetado, revisión y ajuste de las imágenes utilizadas para el entrenamiento del modelo.

El análisis presentado en este trabajo comienza con la versión v16, correspondiente a la primera versión desarrollada bajo los requerimientos definidos para el proyecto. A partir de esta versión se estableció una configuración de referencia para el procesamiento y aumento de datos, sobre la cual se realizaron modificaciones sucesivas con el objetivo de evaluar su efecto en el entrenamiento del modelo hasta llegar a su versión v23.

La Tabla 9 muestra la distribución de imágenes para el entrenamiento, validación, testing y las modificaciones principales que se hicieron sobre el dataset.

CAPÍTULO 6 : RESULTADOS

Versión	Train	Validación	Test	Total	Principal modificación
v16	1.275 (90 %)	100 (7 %)	40 (3 %)	1.415	Configuración inicial de aumentos
v17	1.275 (90 %)	100 (7 %)	40 (3 %)	1.415	Aumento de <i>Cutout</i> : 9 regiones de 5 %
v18	1.275 (90 %)	100 (7 %)	40 (3 %)	1.415	Aumento de <i>Cutout</i> : 10 regiones de 6 %
v19	2.685 (90 %)	189 (6 %)	103 (4 %)	2.977	Gran expansión del dataset
v20	2.685 (90 %)	189 (6 %)	103 (4 %)	2.977	Modificación del recorte y contraste
v21	1.034 (83 %)	132 (11 %)	73 (6 %)	1.239	Depuración del dataset + nueva configuración
v22	1.034 (83 %)	132 (11 %)	73 (6 %)	1.239	Simplificación de aumentos
v23	2.237 (92 %)	132 (5 %)	73 (3 %)	2.442	Dataset depurado y ampliado para configuración final

Tabla 9: Evolución del dataset y configuraciones de aumento.

La Tabla 10 muestra los parámetros de preprocesamiento y aumentación de datos para robustez del dataset.

CAPÍTULO 6 : RESULTADOS

Versión	Preprocesamiento	Rotación	Shear	Otros aumentos
v16	Auto-Orient, recorte 25–75 %, Resize 640×640, ecualización adaptativa	$\pm 15^\circ$	$\pm 10^\circ$ H/V	Cutout: 7 cajas de 15 %
v17	Igual a v16	$\pm 15^\circ$	$\pm 10^\circ$ H/V	Cutout: 9 cajas de 5 %
v18	Igual a v16	$\pm 15^\circ$	$\pm 10^\circ$ H/V	Cutout: 10 cajas de 6 %
v19	Auto-Orient, recorte 25–75 %, Resize 640×640, ecualización adaptativa	$\pm 15^\circ$	$\pm 10^\circ$ H/V	Sin aumentos adicionales
v20	Recorte 12–87 % H / 25–85 % V, Resize 640×640, <i>Contrast Stretching</i>	$\pm 6^\circ$	$\pm 5^\circ$ H / $\pm 10^\circ$ V	—
v21	Resize 640×640	$\pm 5^\circ$	—	Cutout: 8 cajas de 15 %; Blur hasta 2,5 px
v22	Auto-Orient, Resize 640×640	$\pm 5^\circ$	—	Blur hasta 2,5 px
v23	Auto-Orient, Resize 640×640	$\pm 6^\circ$	—	Blur hasta 2,5 px

Tabla 10: Configuración de preprocesamiento y aumentos por versión del dataset.

CAPÍTULO 6 : RESULTADOS

La versión v16 se estableció como configuración de referencia, incorporando operaciones de orientación automática, recorte, redimensionamiento a 640×640 , ecualización adaptativa y transformaciones geométricas. Posteriormente, las versiones v17 y v18 permitieron evaluar distintas intensidades de la técnica *Cutout*.

En la versión v19 se realizó una expansión significativa del conjunto de datos, aumentando el número total de imágenes desde 1.415 hasta 2.977. Esta modificación permitió evaluar el efecto de incrementar la diversidad de muestras disponibles para el entrenamiento.

Posteriormente, las versiones v20, v21 y v22 estuvieron orientadas a reducir progresivamente la intensidad y complejidad de las transformaciones aplicadas. Esta decisión se fundamentó en el principio de que las técnicas de aumento deben introducir variabilidad sin generar ejemplos excesivamente alejados de la distribución de datos esperada durante la operación real del sistema.

Finalmente, la versión v23 incorporó una configuración más conservadora, basada en Auto-Orient, redimensionamiento a 640×640 , rotación de $\pm 6^\circ$ y aplicación de desenfoque de hasta 2,5 píxeles. Esta configuración busca mantener un equilibrio entre la introducción de variabilidad durante el entrenamiento y la conservación de las características visuales relevantes para la detección.

Tras el análisis de cada una de las configuraciones establecidas en las versiones, se considera la v23 como la configuración candidata final, esto al proporcionar el mejor compromiso entre variabilidad sintética, preservación de información visual y desempeño de detección.

Por otro lado, se realizó el etiquetado de las imágenes para luego ser utilizadas en el entrenamiento del modelo YOLO implementado. A continuación, se indican las 13 clases etiquetadas, representadas en las Figuras 6.1-6.2-6.3 y 6.4.

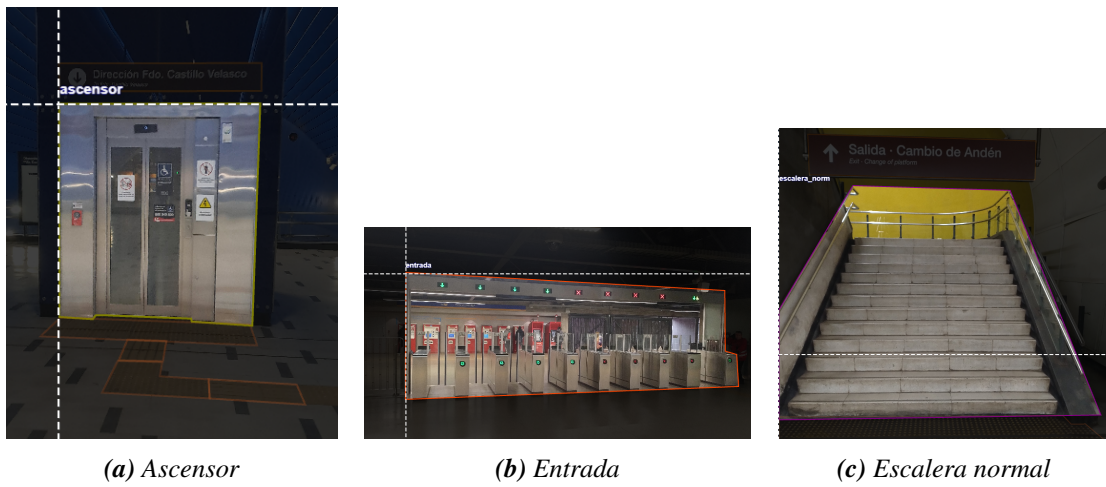


Figura 6.1: Primer set de labels

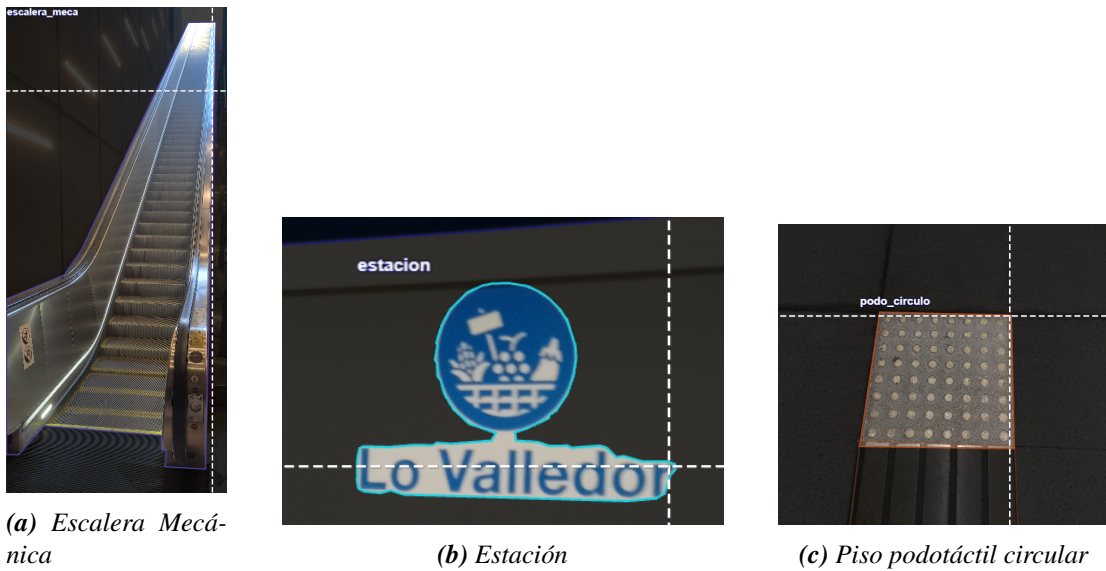


Figura 6.2: Segundo set de labels

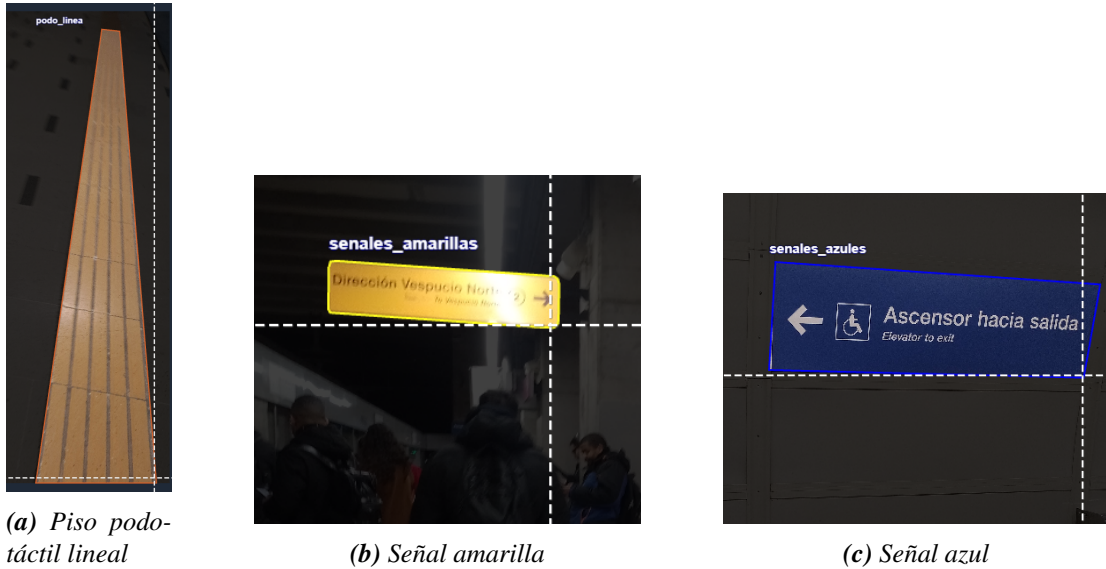


Figura 6.3: Tercer set de labels



Figura 6.4: Cuarto set de labels

6.1.2. Entrenamiento de Modelos de Computer Vision

Una vez completada la etapa de etiquetado, las dos variantes del dataset son utilizadas para el entrenamiento de modelos de visión computacional bajo YOLOv11n, en sus versiones orientadas a *detección* (YOLOv11n) y *segmentación de instancias* (YOLOv11n-seg).

El entrenamiento se realiza en el entorno de Google Colab, seleccionando la GPU dedicada *GPU T4*, y siguiendo una metodología iterativa que incluye:

1. Entrenamiento, validación y prueba con las divisiones respectivas del dataset.
2. Ajuste de hiperparámetros:
 - Épocas: 100
 - Tamaño de imagen: 640
 - Tamaño de batch: 32
 - Inicialización: pesos preentrenados en COCO, seguidos de fine-tuning sobre el dataset propio del Metro de Santiago.
3. Uso de métricas estándar como **mAP@50**, **mAP@50-95**, **F1-score** y *latencia* en dispositivos móviles, proporcionadas por la misma librería de Ultralytics al entrenar modelos.

6.1.3. Desarrollo de Aplicación Móvil

El desarrollo de la aplicación móvil tuvo como objetivo trasladar el flujo conceptual de AIMES hacia una versión ejecutable en dispositivos Android. Esta etapa buscó integrar en una misma plataforma la interfaz de usuario, la activación de cámara, el procesamiento visual, la interpretación de detecciones y la retroalimentación auditiva, manteniendo coherencia con los modelos conceptuales definidos previamente.

Inicialmente se desarrolló un prototipo con Kivy/Buildozer, orientado a reutilizar componentes en Python y validar de forma temprana la estructura de interacción. Si bien esta versión permitió probar parte del flujo en un entorno de escritorio Windows, al ser empaquetada como APK presentó fallas críticas de ejecución en Android, impidiendo su despliegue en el dispositivo objetivo.

Debido a estas limitaciones, el desarrollo móvil fue retomado en Android Studio utilizando Kotlin y Jetpack Compose, lo que permitió acceder directamente a recursos nativos del sistema, como cámara, micrófono, permisos, ciclo de vida, reconocimiento

CAPÍTULO 6 : RESULTADOS

de voz y síntesis auditiva, además de mejorar la depuración y estabilidad de la aplicación.

De esta manera, la versión en Android se consolidó como el entorno principal de integración de AIMES (siguiendo los lineamientos de accesibilidad de Android y WCAG 2.1), conectando los módulos de captura, inferencia, interpretación espacial e interacción por voz.

6.2. Resultados

6.2.1. Dataset

La tarea de captura de imágenes a lo largo del desarrollo del proyecto, concluye en el dataset presente en [50], construido por el equipo a cargo de AIMES almacenado en la plataforma Roboflow en formato de modelos de detección. la Figura 6.5, corresponde a la distribución de instancias etiquetadas por clase del dataset:

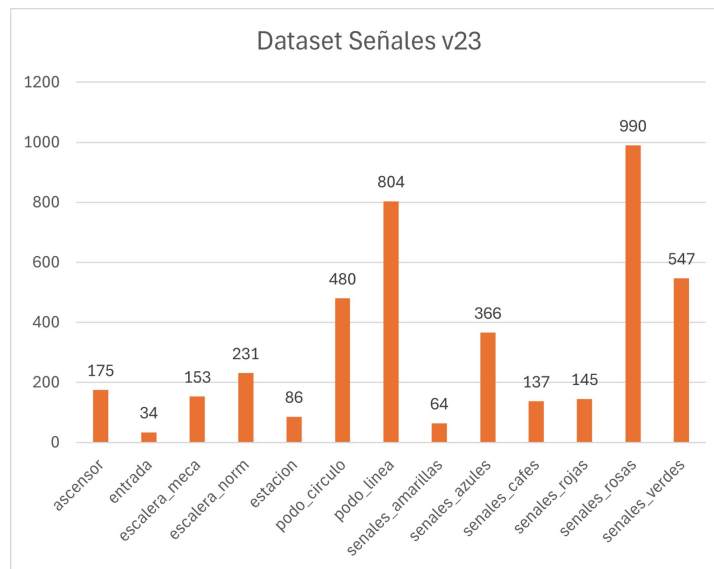


Figura 6.5: Distribución de instancias etiquetadas para las clases presentes en el dataset de señales-detección.

CAPÍTULO 6 : RESULTADOS

Por otro lado, el dataset de la figura 6.6 corresponde a una distribución ajustada de las imágenes, pero en formato para modelos de segmentación que se puede encontrar en [51]. A continuación, su distribución de instancias etiquetadas por clase:

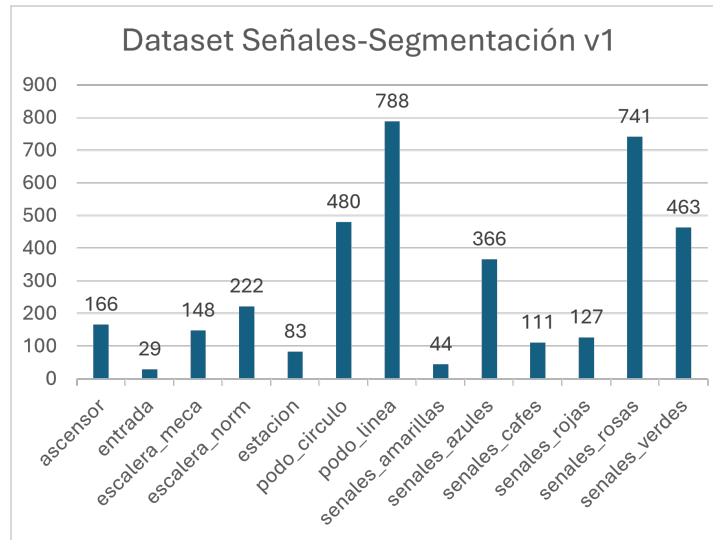


Figura 6.6: Distribución de instancias etiquetadas para las clases presentes en el dataset de señales-segmentación.

De la figura 6.5, se aprecia que es la versión 23 del dataset que ha sido construido desde 2023 para el propósito de AIMES. A día de la confección de este avance de hito, la distribución de las etiquetas indica un claro desequilibrio que beneficia las clases como 'señales rosadas', predominantes en la línea 6 del Metro de Santiago. Por otro lado, para 6.6, la versión 1 fue específicamente creada para la integración de un modelo de segmentación, por lo que es un repositorio independiente y reciente en Roboflow.

6.2.2. Modelos de Computer Vision

El desequilibrio antes mencionado, puede afectar en los entrenamientos de los modelos de *computer vision* que sean utilizados por el equipo, los cuales consisten actualmente de:

Modelos de Detección

- YOLOv11n
- YOLOv11n + capa P2.
- YOLOv11n + capas basadas en Transformer (RepViT).

Los modelos anteriores tienen diferencias en la cantidad de parámetros utilizados como en qué tipo de capas son aplicadas para cada paso en el entrenamiento. La Figura 6.7 indica cantidades y el resumen de precisión en base al mAP.

Característica	YOLOv11n + p2	YOLOv11n + RepViT	YOLOv11n Tradicional
Backbone	C2f-ELAN	C2f-ELAN	C2f-ELAN
Configuración	Agrega escala P2 y bloques C3k2 para objetos pequeños.	Backbone tipo RepViT con bloques C3k2 y SPPF.	Por defecto
Precisión mAP@0.5 All Classes	0.56	0.631	0.698
Parámetros estimados	~2.74 millones	> 5 millones	~2.7 millones
Épocas	100	100	100

Figura 6.7: Comparación de modelos: parámetros, arquitecturas y resultados de mAP obtenidos.

Utilizando la librería de Ultralytics, los modelos son entrenados con el dataset y se obtienen estadísticas desde el *trainer*, mostradas en las Figuras 6.8–6.9 y 6.10:

CAPÍTULO 6 : RESULTADOS

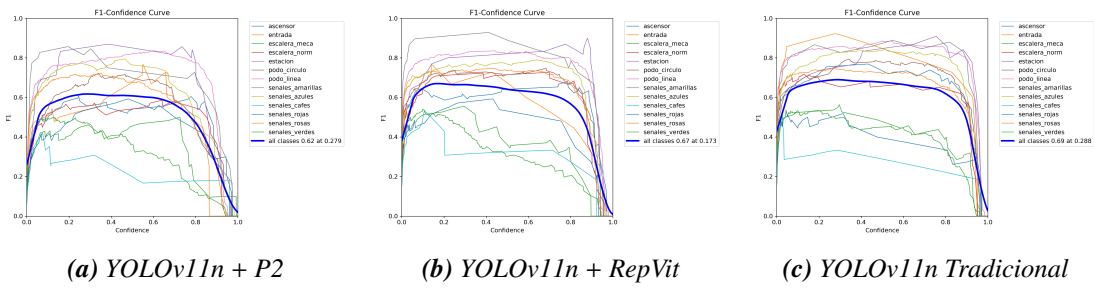


Figura 6.8: Comparación de Confianza en los 3 modelos.

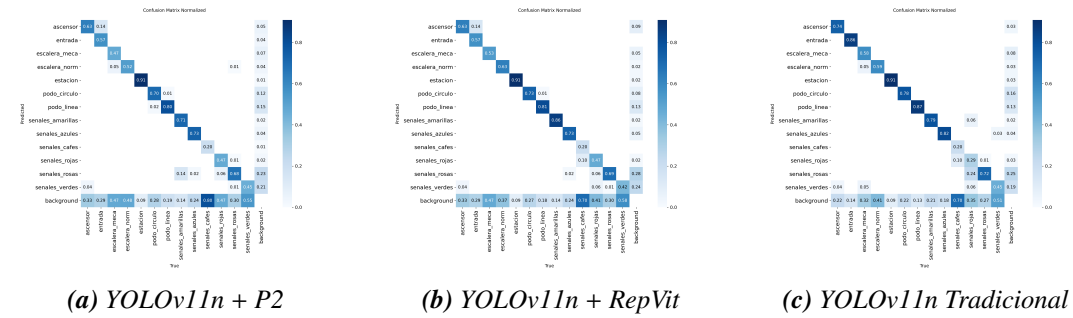


Figura 6.9: Comparación de Matriz de confusión en los 3 modelos.

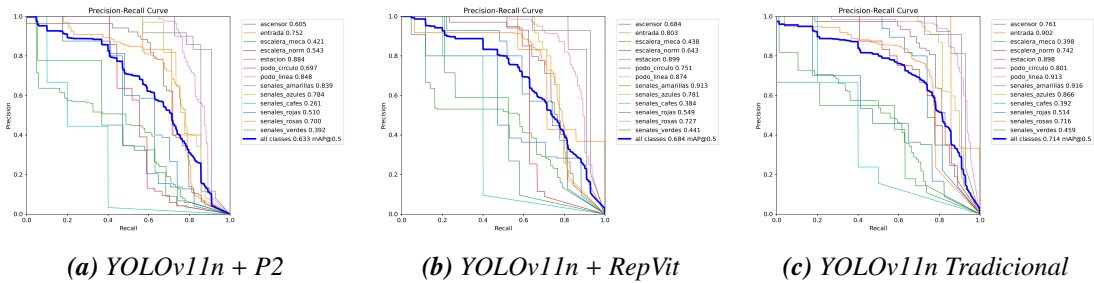


Figura 6.10: Comparación de Precision-Recall en los 3 modelos.

Los resultados de las métricas F1 (6.8), Matriz de Confusión (6.9) y Precision-Recall (6.10) evidencian el desequilibrio de clases del dataset. Las clases con menor representación, como señales cafés o verdes, obtienen menores valores de F1 y mayor confusión. En contraste, las clases más numerosas muestran un mejor rendimiento en precisión y recall, lo que confirma que la distribución de instancias es un factor crítico para el desempeño general de los modelos.

Modelo de Segmentación

- YOLOv11n-Seg (versión nano con cabeza de segmentación)

En la Figura 6.11 se aprecia que fue entrenado con características similares a los modelos de detección.

Característica	YOLOv11n-Segmentación
Backbone	C2f-ELAN
Configuración	Por defecto
Precisión mAP@0.5 All Classes	0.715
Parámetros calculados	~2.900.000
Image Size	512
Épocas	100

Figura 6.11: Parámetros, arquitecturas y resultados de mAP obtenidos de YOLO11n-Seg.

Los parámetros de medición obtenidos corresponden a las figuras 6.12, 6.13 y 6.14.

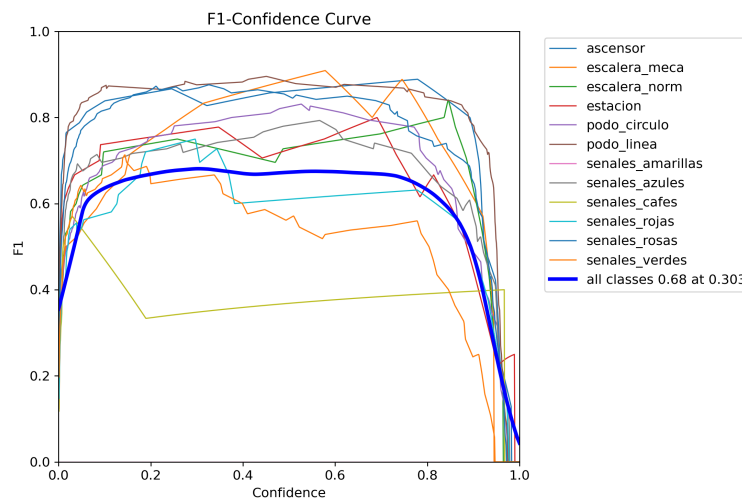


Figura 6.12: Curva de F1-score por confidence threshold del modelo YOLOv11n-Seg

CAPÍTULO 6 : RESULTADOS

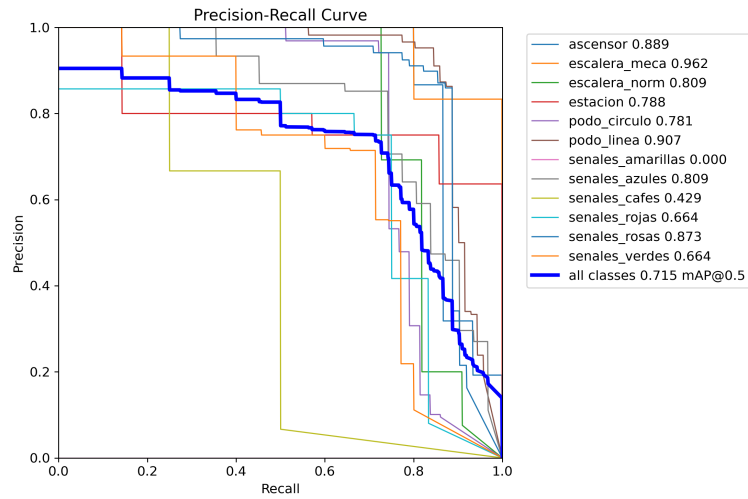


Figura 6.13: Curva Precision–Recall (PR) del modelo YOLOv11n-Seg.

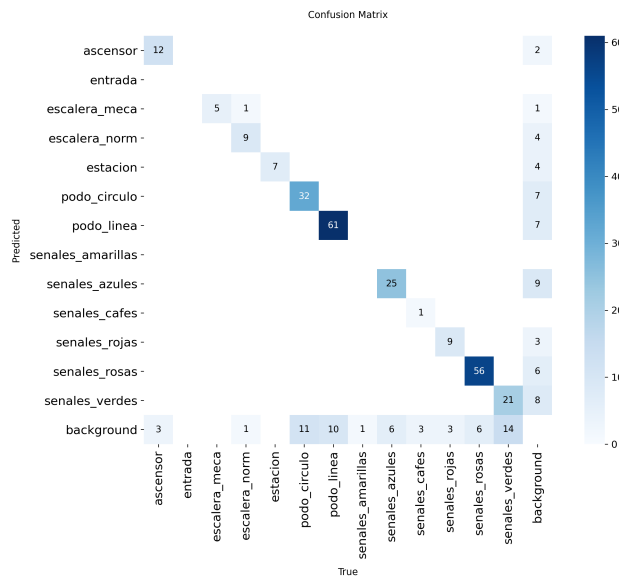


Figura 6.14: Matriz de confusión del modelo YOLOv11n-Seg.

De ambos tipos de modelos, se aprecia la robustez en las clases más presentes de los datasets. Sin embargo, se decide avanzar con el modelo de segmentación para sacar provecho de los resultados *máscaras* que se obtengan de él.

6.2.3. Rendimiento de Modelos Obtenidos

La diferencia entre YOLOv11n-Trad y YOLOv11n-P2 es relativamente reducida, mientras que YOLOv11n-RepViT presenta una disminución considerable en la capacidad de procesamiento. Particularmente, el tiempo de inferencia de este último supera el segundo por imagen, lo que representa una limitación importante para un sistema orientado a procesamiento en tiempo real.

En cuanto al consumo de recursos, se obtienen los siguientes resultados que se muestran en la Tabla 11:

Modelo	FPS	Inferencia	Uso de recursos
YOLOv11n-Trad	14.60	68.48 ms	10.14 MB 34.48 % CPU
YOLOv11n-P2	12.46	80.26 ms	16.85 MB 32.08 % CPU
YOLOv11n-RepViT	0.99	1008.10 ms	194.43 MB 45.84 % CPU

Tabla 11: Comparación de rendimiento y consumo de recursos de los modelos evaluados.

A partir de los resultados obtenidos, se observa que YOLOv11n-Trad presenta el comportamiento más equilibrado entre las métricas evaluadas. Si bien YOLOv11n-P2 presenta un menor consumo promedio de CPU, esta ventaja es acompañada por una disminución en el rendimiento temporal y un incremento en el tamaño del modelo. En contraste, YOLOv11n-RepViT presenta un costo computacional considerablemente superior, junto con una reducción significativa en la tasa de procesamiento.

En consecuencia, se selecciona YOLOv11n-Trad como modelo para la implementa-

ción del sistema, debido a que alcanza el mayor número de FPS (14.60), el menor tiempo de inferencia (68.48 ms) y el menor tamaño de modelo (10.14 MB). Aunque YOLOv11n-P2 presenta un menor uso promedio de CPU, esta diferencia no resulta suficiente para compensar su menor velocidad de procesamiento y mayor tamaño. Por su parte, YOLOv11n-RepViT no resulta adecuado para el escenario de aplicación debido a su baja tasa de procesamiento y elevado consumo de memoria.

6.2.4. Aplicación Móvil

En relación con la aplicación móvil, los resultados obtenidos permiten evidenciar el cumplimiento del flujo de interacción definido previamente en los modelos conceptuales del sistema, según lo descrito en la Figura 4.6, donde se contempla la respuesta del sistema ante detecciones válidas, ausencia de elementos relevantes o fallos de contexto. De la misma manera, la interfaz final conserva la lógica general propuesta en el diseño conceptual presentado en la Figura 5.6, aunque con ajustes propios de la migración hacia una implementación nativa en Android.

En una primera etapa se desarrolló un prototipo utilizando el framework Kivy, mostrado en la Figura 6.15. Este prototipo permitió explorar la estructura inicial de la interfaz y validar parcialmente el flujo de interacción en un entorno de escritorio Windows. Sin embargo, al ser empaquetado como APK para dispositivos Android, presentó fallas críticas de ejecución que impidieron su despliegue en el entorno real previsto. Debido a estas limitaciones, principalmente asociadas a compatibilidad, rendimiento, trazabilidad de errores y acceso a APIs nativas de Android, se decidió descartar Kivy como solución definitiva.

A partir de esta experiencia, el desarrollo de la interfaz móvil fue retomado utilizando Android Studio con Kotlin, lo que permitió integrar funciones nativas del sistema operativo, soporte estable para cámara, manejo de permisos, síntesis de voz y una interfaz visual acorde con los requisitos definidos en el Capítulo 3. Como resultado, la aplicación logró ejecutarse directamente en un dispositivo Android, como se puede ver

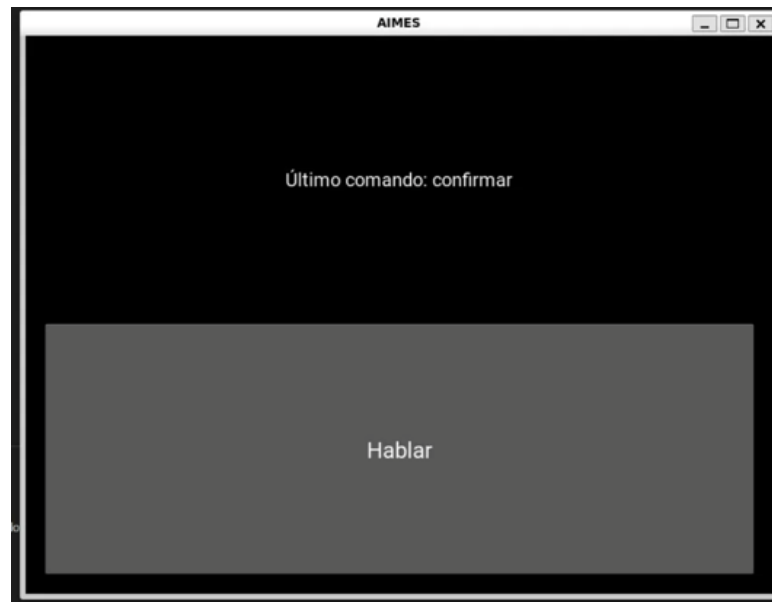


Figura 6.15: Primer prototipo de la interfaz móvil desarrollado en Kivy, descartado en fase posterior.

en las Figuras 5.8 y 5.9, superando así la principal limitación observada en el prototipo desarrollado con Kivy/Buildozer.

Desde el punto de vista funcional, el resultado más relevante fue la consolidación de un flujo operativo en dispositivo móvil, descrito en la Figura 5.5, donde la aplicación permite iniciar desde una pantalla principal, acceder al modo de cámara, procesar frames capturados en tiempo real, visualizar detecciones y generar información de orientación a partir de ellas. Este comportamiento se relaciona directamente con el modelo conceptual planteado, ya que el sistema logra pasar desde la activación de la asistencia hasta la interpretación de elementos detectados y la generación de una respuesta para el usuario.

La vista técnica de prueba, descrita previamente en el Capítulo 5, permitió verificar la integración entre los módulos principales de la aplicación. En esta pantalla como se presenta en la Figura 5.10, se visualiza la cámara activa, las detecciones generadas por el modelo y métricas internas utilizadas durante la depuración, tales como FPS, precisión y confianza promedio. Esto permitió comprobar que el modelo podía ejecu-

CAPÍTULO 6 : RESULTADOS

tarse localmente en el teléfono y que sus salidas podían ser utilizadas por la capa de interpretación espacial para generar orientación.

Además, la interfaz móvil permitió validar que los componentes definidos en el diseño no quedaron únicamente como elementos conceptuales, sino que fueron trasladados a una aplicación funcional. Particularmente, se logró integrar navegación entre pantallas, activación del flujo visual, visualización de resultados, control de guía y retroalimentación auditiva. De esta manera, la aplicación cumple el flujo principal propuesto para AIMES: capturar el entorno, procesarlo localmente, obtener detecciones, interpretarlas especialmente y entregar una salida comprensible para apoyar la orientación del usuario. Este flujo se puede ver descrito a nivel de implementación de código de la aplicación en la Figura 5.2.

A partir de lo anterior, los resultados de la aplicación móvil muestran que la migración a Android nativo permitió obtener una versión ejecutable y funcional de AIMES. Si bien la validación realizada corresponde a una prueba técnica del prototipo y no a una evaluación formal con usuarios finales, la implementación permitió comprobar la factibilidad del sistema en la plataforma objetivo y verificar el cumplimiento del flujo de interacción definido en los modelos conceptuales del proyecto.

Para complementar lo descrito, se dispone de material adicional con registros de prueba y capturas del flujo de interacción de la aplicación.

El material complementario asociado a esta etapa incluye:

- registros en video de pruebas realizadas en terreno;
- capturas adicionales del flujo de interacción de la aplicación;
- evidencia visual del funcionamiento de la cámara, detecciones y métricas de ejecución.

Este material se encuentra disponible en un repositorio en línea y puede ser solicitado a los autores para su revisión.

6.3. Síntesis y discusión de resultados

Los resultados obtenidos permiten evaluar AIMES desde tres dimensiones complementarias: la disponibilidad de información visual representativa, el desempeño de los modelos de visión por computador y la factibilidad de integrar dichos componentes en una aplicación móvil nativa.

En primer lugar, la construcción de los conjuntos de datos permitió disponer de material específico del Metro de Santiago, incluyendo señalética, accesos, escaleras y pisos podotáctiles. No obstante, el análisis de la distribución de instancias evidenció un desbalance entre clases, con una mayor representación de elementos propios de la Línea 6 y una presencia reducida de otras categorías. Esta situación se refleja en los resultados de entrenamiento, donde las clases con mayor cantidad de ejemplos presentan un comportamiento más estable, mientras que las categorías minoritarias muestran mayores niveles de confusión y menor consistencia.

En segundo lugar, la comparación de modelos permitió analizar distintas alternativas de detección y segmentación. Aunque los resultados demuestran que es posible reconocer clases relevantes del entorno, la precisión obtenida no debe interpretarse como garantía de desempeño uniforme en todas las estaciones o condiciones de uso. Las variaciones de iluminación, oclusión, distancia y movimiento continúan representando desafíos, especialmente para clases con menor representación en los conjuntos de entrenamiento.

En tercer lugar, la aplicación desarrollada en Android permitió comprobar la factibilidad de integrar captura de video, inferencia local, visualización de detecciones e interacción auditiva en un dispositivo móvil. A diferencia del prototipo multiplataforma, la versión nativa logró ejecutarse en el teléfono y proporcionó acceso directo a los recursos de cámara, audio y ciclo de vida de Android. Esto constituye el principal resultado del trabajo individual presentado en esta memoria, ya que transforma los componentes previamente evaluados de forma aislada en un flujo operativo dentro de la plataforma objetivo.

CAPÍTULO 6 : RESULTADOS

Sin embargo, los resultados corresponden a una validación técnica y funcional del prototipo, no a una validación definitiva de seguridad o usabilidad. Aún se requiere realizar pruebas sistemáticas con personas con discapacidad visual, medir la latencia extremo a extremo, evaluar el consumo energético y estudiar el comportamiento del sistema en un conjunto más amplio de estaciones y dispositivos. Asimismo, la estimación de distancia utilizada entrega una referencia aproximada y debe ser calibrada antes de considerarse una medida física precisa.

En conjunto, los resultados demuestran que la arquitectura propuesta es técnicamente viable como base para una herramienta de orientación inmediata. La integración nativa permitió superar el bloqueo de despliegue del prototipo anterior y establecer una plataforma desde la cual es posible continuar refinando los modelos, la interpretación espacial y la experiencia auditiva. Estos hallazgos permiten dar paso al capítulo siguiente, donde se presentan las conclusiones del trabajo, el cumplimiento de los objetivos y las líneas de desarrollo futuro.

7. Conclusiones

7.1. Conclusiones generales

El desarrollo de AIMES permitió avanzar en la construcción de una solución móvil orientada a apoyar la autonomía de personas con discapacidad visual en un entorno complejo como el Metro de Santiago. A partir del problema identificado, esta memoria abordó el diseño, evolución e implementación de una aplicación capaz de capturar información visual desde la cámara del teléfono, procesarla mediante visión por computador y transformarla en retroalimentación auditiva comprensible para el usuario.

Una de las principales conclusiones del trabajo es que el teléfono móvil constituye una plataforma viable para prototipar soluciones de asistencia accesible sin requerir infraestructura adicional en las estaciones. Esta decisión resulta coherente con el enfoque de AIMES, ya que permite aprovechar recursos disponibles en el propio dispositivo, como cámara, micrófono, capacidad de procesamiento y salida de audio. En este sentido, la solución implementada demuestra que es posible integrar en una aplicación Android un flujo funcional compuesto por captura de cámara, inferencia local, interpretación espacial, comandos de voz y síntesis de instrucciones auditivas.

Asimismo, el proceso de desarrollo evidenció la importancia de avanzar mediante prototipos incrementales. Las primeras aproximaciones permitieron validar ideas generales del sistema, pero también mostraron limitaciones importantes al intentar trasladar un enfoque multiplataforma al teléfono. En particular, el prototipo basado en Kivy/Buildozer permitió explorar el flujo conceptual, pero no entregó la estabilidad necesaria para ejecutar la aplicación de forma confiable en Android. Esta experiencia justificó la migración hacia una implementación nativa, decisión que permitió obtener mayor control sobre permisos, ciclo de vida, cámara, audio y herramientas de depuración.

Desde el punto de vista funcional, AIMES logró consolidar una base técnica para orientación inmediata. La aplicación no busca resolver una navegación completa puerta

a puerta dentro del Metro, sino entregar información útil y oportuna sobre elementos detectados en el entorno, su ubicación relativa y su cercanía aproximada. Esta delimitación permitió concentrar el desarrollo en un flujo alcanzable y validable dentro del tiempo disponible, manteniendo abierta la posibilidad de extender el sistema hacia funciones más avanzadas en el futuro.

En consecuencia, la memoria cumple con el propósito de transformar una propuesta conceptual en una aplicación móvil funcional, organizada en módulos concretos y ejecutable sobre Android. Es importante mencionar que el resultado obtenido no corresponde a una solución final lista para uso masivo, sino a una base técnica funcional sobre la cual es posible seguir iterando, evaluando y refinando AIMES en escenarios reales.

7.2. Cumplimiento de objetivos

Respecto al objetivo de construir una solución de asistencia basada en visión por computador y retroalimentación auditiva, se logró implementar una aplicación móvil nativa que integra los componentes principales requeridos para el funcionamiento de AIMES. La aplicación permite activar la cámara del dispositivo, procesar frames en tiempo real, ejecutar inferencia local mediante un modelo de detección, interpretar los resultados obtenidos y entregar información al usuario mediante voz.

En relación con la captura y procesamiento visual, se implementó un pipeline basado en CameraX y TensorFlow Lite, permitiendo ejecutar el modelo directamente en el dispositivo móvil. Esta implementación mantiene el enfoque de operación local definido para AIMES y evita depender de conectividad externa durante la inferencia. Además, se incorporaron métricas internas como FPS, precisión y confianza promedio, utilizadas para observar el comportamiento del sistema durante las pruebas.

En cuanto a la interpretación de resultados, se desarrolló una capa de orientación encargada de traducir las detecciones visuales en información semántica, que permite estimar posición relativa, distancia aproximada y objeto prioritario, generando una

CAPÍTULO 7 : CONCLUSIONES

representación más útil para el usuario que la salida cruda del modelo. De esta manera, las detecciones dejan de ser únicamente cajas y etiquetas, y pasan a convertirse en instrucciones breves, comprensibles y accionables.

Respecto a la interacción por voz, se integraron APIs nativas de Android para reconocimiento de comandos y síntesis de voz. Esto permitió habilitar una interacción coherente con las necesidades de personas con discapacidad visual, privilegiando el uso de comandos hablados y retroalimentación auditiva. Además, se incorporaron mecanismos para reducir la repetición excesiva de mensajes, evitando saturar al usuario con instrucciones redundantes.

También se incorporó el reconocimiento de texto como componente complementario dentro del flujo global de AIMES. Su función es apoyar la validación de destino o dirección cuando se detecta señalética con información textual relevante. Si bien este componente forma parte del sistema final, en esta memoria se considera principalmente desde su integración funcional con la aplicación móvil.

Por último, se implementaron pantallas principales, pantallas auxiliares y una vista técnica de prueba. Esta última fue clave para validar el comportamiento interno del sistema, ya que permitió visualizar la cámara, las detecciones, las métricas y los controles de prueba directamente en el teléfono. Con ello, la aplicación no solo funcionó como interfaz de usuario, sino también como herramienta de depuración para verificar el flujo cámara–inferencia–orientación–voz.

7.3. Limitaciones

A pesar de los avances obtenidos, la solución desarrollada presenta limitaciones que deben considerarse al interpretar los resultados. En primer lugar, la validación realizada corresponde principalmente a una validación técnica y funcional del prototipo, por lo que aún no puede considerarse una validación definitiva de usabilidad, seguridad o desempeño en condiciones reales con usuarios finales.

CAPÍTULO 7 : CONCLUSIONES

En segundo lugar, la orientación generada por el sistema se basa en estimaciones relativas de posición y distancia, que permiten entregar información útil para una primera versión de orientación inmediata; sin embargo, no constituye una medición física precisa ni reemplaza un sistema completo de localización o navegación interior. Por esta razón, las indicaciones deben interpretarse como apoyo contextual y no como instrucciones absolutas de desplazamiento.

Otra limitación relevante se relaciona con el desempeño de los modelos de visión, ya que, si bien los resultados permitieron seleccionar una alternativa adecuada para integración móvil, el comportamiento del sistema puede verse afectado por iluminación, oclusiones, distancia al objeto, movimiento del usuario y variabilidad entre estaciones. Además, el desbalance de clases del dataset influye en el desempeño de detección, especialmente en categorías con menor representación.

En términos de rendimiento, los resultados obtenidos muestran que el modelo seleccionado presenta un balance favorable entre velocidad, tiempo de inferencia y consumo de recursos. Sin embargo, estos valores corresponden a pruebas controladas y deben validarse en una mayor variedad de dispositivos y escenarios. En una aplicación orientada a asistencia en movilidad, la estabilidad temporal del sistema es tan importante como la precisión del modelo, por lo que futuras pruebas deben considerar también latencia extremo a extremo, consumo energético y comportamiento durante uso prolongado.

También se debe considerar que algunas funcionalidades complementarias, como configuración avanzada, retroalimentación mediante vibración, alertas automatizadas o navegación completa por estación, quedaron como base de interfaz o proyección futura. Estas funciones permiten ampliar el alcance de AIMES, pero no constituyen aún el núcleo validado de la implementación descrita.

Del mismo modo, el sistema requiere pruebas más extensas con personas con discapacidad visual, lo cual es fundamental para evaluar si las instrucciones auditivas son realmente comprensibles, oportunas y seguras durante el desplazamiento. Sin esta validación, no es posible afirmar que la solución sea completamente adecuada para uso

cotidiano, aunque sí se puede concluir que existe una base técnica funcional para continuar su desarrollo.

7.4. Trabajo futuro

Como trabajo futuro, se considera prioritario validar AIMES con usuarios finales, tanto en entornos controlados como en escenarios representativos del Metro de Santiago. Estas pruebas permitirían evaluar la comprensión de las instrucciones auditivas, la carga cognitiva, la confianza en el sistema y la utilidad real de la asistencia durante el desplazamiento.

También se requiere fortalecer el modelo de visión mediante la ampliación y balance del dataset, incorporando más estaciones, condiciones de iluminación, ángulos de captura, niveles de congestión y variaciones de señalética. Esto permitiría mejorar la robustez del sistema frente a escenarios reales y reducir errores en clases con menor representación.

Otra línea de mejora corresponde a la estimación de distancia y orientación, ya que la versión actual utiliza una aproximación relativa basada en la posición de las detecciones dentro del frame, suficiente para una primera versión funcional, pero que podría complementarse con calibración por dispositivo, sensores adicionales o técnicas de seguimiento temporal más estables.

Finalmente, AIMES podría evolucionar desde una herramienta de orientación inmediata hacia un sistema de navegación asistida más completo. Para ello sería necesario integrar mapas internos, localización dentro de estaciones, rutas alternativas e identificación de salidas específicas, considerando siempre nuevas pruebas de seguridad y usabilidad con personas con discapacidad visual.

7.5. Cierre final

En conclusión, AIMES representa una base funcional para explorar el uso de visión por computador e interacción auditiva como apoyo a la movilidad de personas con discapacidad visual en el Metro de Santiago. La memoria permitió demostrar que una solución móvil nativa puede integrar captura visual, inferencia local, orientación espacial y retroalimentación por voz en un mismo dispositivo, sin depender de infraestructura externa.

Aunque el sistema aún requiere validaciones más extensas y mejoras antes de considerarse una solución final para uso cotidiano, el trabajo realizado establece una plataforma técnica sobre la cual continuar iterando. La migración a Android nativo, la implementación del pipeline visual y la integración de voz permitieron transformar AIMES desde una propuesta conceptual hacia un prototipo móvil funcional, abriendo camino para futuras versiones más robustas, accesibles y centradas en las necesidades reales de sus usuarios.

Referencias

- [1] Ministerio de Desarrollo Social y Familia. *Resultados Encuesta Nacional de Discapacidad y Dependencia 2022: Población adulta*. [https://observatorio.ministeriodesarrollosocial.gob.cl/storage/docs/endide/230128_Resultados_ENDIDE_2022\(Poblacion-adulta\).pdf](https://observatorio.ministeriodesarrollosocial.gob.cl/storage/docs/endide/230128_Resultados_ENDIDE_2022(Poblacion-adulta).pdf). Gobierno de Chile. 2022.
- [2] World Health Organization: WHO. *Ceguera y discapacidad visual*. es. Ago. de 2023. URL: <https://www.who.int/es/news-room/fact-sheets/detail/blindness-and-visual-impairment>.
- [3] World Health Organization: WHO. *World report on vision*. 2019. URL: <https://www.who.int/docs/default-source/documents/publications/world-vision-report-accessible.pdf>.
- [4] Francia Quezada. «Reportaje revela que ambulantes han dañado cerca del 30 % de los ascensores del Metro de Santiago». es-ES. En: *BioBioChile - La Red De Prensa Más Grande De Chile* (ene. de 2023). URL: <https://www.biobiochile.cl/noticias/nacional/region-metropolitana/2023/06/05/reportaje-revela-que-ambulantes-han-danado-cerca-del-30-de-los-ascensores-del-metro-de-santiago.shtml>.
- [5] Erum Salam. «I tried Be My Eyes, the popular app that pairs blind people with helpers». En: *The Guardian* (jul. de 2019). URL: <https://www.theguardian.com/lifeandstyle/2019/jul/12/be-my-eyes-app-blind-people-helpers>.
- [6] Perkins School for the Blind. *Microsoft seeing AI and low Vision Review – Perkins School for the Blind*. en-US. Oct. de 2022. URL: <https://www.perkins.org/resource/microsoft-seeing-ai-and-low-vision-review/>.
- [7] Perkins School for the Blind. *Lazarillo: a free accessible GPS app for the blind and visually impaired – Perkins School for the blind*. en-US. Jul. de 2024. URL:

<https://www.perkins.org/resource/lazarillo-free-accessible-gps-app-blind-and-visually-impaired/>.

- [8] Metro de Santiago. Nov. de 2023. URL: <https://www.metro.cl/noticias/lazarillo-la-aplicacion-que-ayuda-a-las-personas-ciegas-se-expande-en-la-red-de-metro-y-alcanza-las-13-estaciones>.
- [9] Augusto Valenzuela. *¿Cómo ayudar sin meter la pata?* Nov. de 2018. URL: <https://www.uc.cl/noticias/como-ayudar-a-alguien-con-discapacidad-sin-meter-la-pata/>.
- [10] Clinic. *“Cuando alguien pasa de largo, da entre pena y rabia”: Cómo ayudar correctamente a una persona ciega en la calle y que no resulte incómodo.* es-CL. Jun. de 2024. URL: <https://www.theclinic.cl/2024/06/26/como-ayudar-de-la-forma-correcta-a-una-persona-ciega-en-la-calle/>.
- [11] Agenda País. «Inclusión laboral: los rubros en los que las personas ciegas han logrado encontrar trabajo y mantenerse con éxito». En: *El Mostrador* (mar. de 2023). URL: <https://www.elmostrador.cl/agenda-pais/2020/10/22/inclusion-laboral-los-rubros-en-los-que-las-personas-ciegas-han-logrado-encontrar-trabajo-y-mantenerse-con-exito/>.
- [12] Fundación Luz y Autonomía e inclusión para personas ceguera y bajo visión. *Discapacidad visual en Chile - Fundación Luz | Autonomía e inclusión para personas ceguera y bajo visión.* es-CL. <https://fundacionluz.cl/area/estudio-sobre-personas-con-discapacidad-visual-en-chile-fundacion-luz-endide-2022/>. Oct. de 2025. URL: <https://fundacionluz.cl/area/discapacidad-visual-en-chile/>.
- [13] Perkins School for the Blind. *Using Aira with low vision – Perkins School for the Blind.* en-US. Jul. de 2023. URL: <https://www.perkins.org/resource/aira-low-vision-review/>.

- [14] Ghatan Fauzi Nugraha, Aradea, Rianto y Rina Mardiati. «Object Detection for the Visually Impaired: A Systematic Literature Review». En: *2024 10th International Conference on Wireless and Telematics (ICWT)*. 2024, págs. 1-16. DOI: 10.1109/ICWT62080.2024.10674727.
- [15] Ultralytics. *SAM (Segment Anything Model)*. URL: <https://docs.ultralytics.com/models/sam/>.
- [16] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar y Ross Girshick. «Segment Anything». En: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. de 2023, págs. 4015-4026.
- [17] Chaoning Zhang, Dongshen Han, Yu Qiao, Jung Uk Kim, Sung-Ho Bae, Seungkyu Lee y Choong Seon Hong. *Faster Segment Anything: Towards Lightweight SAM for Mobile Applications*. 2023. arXiv: 2306.14289 [cs.CV]. URL: <https://arxiv.org/abs/2306.14289>.
- [18] Ultralytics. *MobileSAM (Mobile Segment Anything Model)*. URL: <https://docs.ultralytics.com/es/models/mobile-sam/>.
- [19] Joseph Redmon, Santosh Divvala, Ross Girshick y Ali Farhadi. *You only look once: Unified, Real-Time Object Detection*. Ene. de 2015. URL: <https://arxiv.org/abs/1506.02640>.
- [20] So Development. *Comparing YOLOv11 and YOLOv12: A Deep Dive into the Next-Generation Object Detection Models*. en-US. Feb. de 2025. URL: <https://so-development.org/comparing-yolov11-and-yolov12-a-deep-dive-into-the-next-generation-object-detection-models/>.
- [21] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov y Sergey Zagoruyko. *End-to-End Object Detection with Transformers*. 2020. arXiv: 2005.12872 [cs.CV]. URL: <https://arxiv.org/abs/2005.12872>.

- [22] Ao Wang, Hui Chen, Zijia Lin, Jungong Han y Guiguang Ding. «Rep ViT: Re-visiting Mobile CNN From ViT Perspective». En: *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024, págs. 15909-15920. doi: 10.1109/CVPR52733.2024.01506.
- [23] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto y Hartwig Adam. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV]. URL: <https://arxiv.org/abs/1704.04861>.
- [24] Sachin Mehta y Mohammad Rastegari. *MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer*. 2022. arXiv: 2110.02178 [cs.CV]. URL: <https://arxiv.org/abs/2110.02178>.
- [25] Tadeh Ghahremanians y Hossein Mahvash Mohammadi. «Efficient Real-Time Pathfinding for Visually Impaired Individuals». En: *IEEE Access* 13 (2025), págs. 71323-71334. doi: 10.1109/ACCESS.2025.3562247.
- [26] Alpha Cephei Org. *Vosk Webpage*. 2022. URL: <https://alphacephei.com/vosk/>.
- [27] Olga Yakovleva. *RHVoice github*. 2022. URL: <https://github.com/RHVoice/RHVoice>.
- [28] Kivy Authors. *Kivy Documentation*. 2011. URL: <https://kivy.org/doc/stable/>.
- [29] *Android Studio App Tools - Android Developers*. URL: <https://developer.android.com/studio>.
- [30] Developers. *Android's Kotlin-first approach*. <https://developer.android.com/kotlin/first/>. 2026.
- [31] Developers. *Get started with Jetpack Compose*. <https://developer.android.com/develop/ui/compose/documentation/>. 2026.

- [32] Developers. *CameraX overview*. <https://developer.android.com/media/camera/camerax/>. 2026.
- [33] GeeksforGeeks. *Introduction to TensorFlow Lite*. <https://www.geeksforgeeks.org/deep-learning/introduction-to-tensorflow-lite/>. 2025.
- [34] Developers. *SpeechRecognizer*. <https://developer.android.com/reference/android/speech/SpeechRecognizer/>. 2026.
- [35] Developers. *TextToSpeech*. <https://developer.android.com/reference/android/speech/tts/TextToSpeech/>. 2026.
- [36] World Wide Web Consortium. *Web Content Accessibility Guidelines (WCAG) 2.1*. <https://www.w3.org/TR/WCAG21/>. 2018.
- [37] Nii Mante y James D. Weiland. «Visually Impaired Users can Locate and Grasp Objects Under the Guidance of Computer Vision and Non-Visual Feedback». En: *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2018, págs. 1-4. DOI: 10.1109/EMBC.2018.8512918.
- [38] B Deepthi Jain, Shwetha M Thakur y K V Suresh. «Visual Assistance for Blind Using Image Processing». En: *2018 International Conference on Communication and Signal Processing (ICCSP)*. 2018, págs. 0499-0503. DOI: 10.1109/ICCSP.2018.8524251.
- [39] Jawaaid Nasreen, Warsi Arif, Asad Ali Shaikh, Yahya Muhammad y Monaisha Abdullah. «Object Detection and Narrator for Visually Impaired People». En: *2019 IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*. 2019, págs. 1-4. DOI: 10.1109/ICETAS48360.2019.9117405.
- [40] Mario Almeida, Stefanos Laskaridis, Abhinav Mehrotra, Lukasz Dudziak, Ilias Leontiadis y Nicholas D. Lane. *Smart at what cost? Characterising Mobile Deep*

Neural Networks in the wild. 2021. arXiv: 2109.13963 [cs.LG]. URL: <https://arxiv.org/abs/2109.13963>.

- [41] Ratnesh Litoriya, Kailash Chandra Bandhu, Ishika Rajawat, Hany Jagwani y Chirayu Yadav. «Implementing Visual Assistant using YOLO and SSD for Visually-Impaired Persons». En: *Journal of Automation, Mobile Robotics and Intelligent Systems* 17.4 (feb. de 2024), págs. 79-87. doi: 10.14313/JAMRIS/4-2023/33. URL: <https://www.jamris.org/index.php/JAMRIS/article/view/844>.
- [42] Vinish Pujari, Karthik Madnal y Divya Premchandran. «Mobile App for Enhancing Accessibility Among the Visually Impaired». En: *2024 2nd DMIHER International Conference on Artificial Intelligence in Healthcare, Education and Industry (IDICAIEI)*. 2024, págs. 1-6. doi: 10.1109/IDICAIEI61867.2024.10842749.
- [43] Rahima Khanam y Muhammad Hussain. *A Review of YOLOv12: Attention-Based Enhancements vs. Previous Versions*. 2025. arXiv: 2504.11995 [cs.CV]. URL: <https://arxiv.org/abs/2504.11995>.
- [44] URL: <https://github.com/ultralytics>.
- [45] Kivy. *GitHub - kivy/plyer: Plyer is a platform-independent Python wrapper for platform-dependent APIs*. URL: <https://github.com/kivy/plyer>.
- [46] Instituto Nacional de Estadísticas (INE). *Censo 2024 – Primeros resultados*. <https://www.ine.gob.cl>. INE. 2024.
- [47] Braille Chile. *NVDA: Un Lector de Pantalla Gratuito y Accesible*. <https://www.braillechile.cl/post/nvda-un-lector-de-pantalla-gratuito-y-accesible/>. 2024.
- [48] Ver Sin Límites Accesibilidad. *TalkBack, qué es, cómo funciona y para que sirve*. <https://versinlimitesaccesibilidad.com/talkback-que-es-como-funciona-y-para-que-sirve/>. 2021.

- [49] mac-OS.org. *¿Qué es VoiceOver?* <https://es.mac-os.org/?p=16514/>. 2023.
- [50] AIMES. *Senales Computer Vision Model*. 2025. URL: <https://universe.roboflow.com/aimes/senales-cnijl>.
- [51] AIMES. *Senales-segmentation Computer Vision Dataset*. 2025. URL: <https://universe.roboflow.com/aimes/senales-segmentation-zvzpo>.

Anexos

.1. Pauta de entrevista en terreno a Don Iván

La siguiente pauta corresponde al conjunto de preguntas utilizadas como guía durante la entrevista en terreno.

1. ¿Cómo se moviliza habitualmente dentro del Metro y qué herramientas utiliza para orientarse?

Me movilizó de forma independiente dentro del Metro y utilizo principalmente mi bastón para orientarme por las estaciones.

2. ¿Qué dificultades encuentra al interactuar con elementos o dispositivos presentes dentro de las estaciones?

Una de las principales dificultades que tengo es interactuar con las pantallas, especialmente con las máquinas para recarga. Muchas veces no tengo idea de qué parte de la pantalla estoy tocando, por lo que se me dificulta realizar algunas cosas.

3. ¿De qué manera una aplicación o sistema tecnológico podría ayudarlo frente a estas dificultades?

Creo que una aplicación o sistema podría ayudarme indicando dónde debo tocar en la pantalla o leyendo la información que aparece en ella, así la puedo usar sin pedirle ayuda a nadie.

4. Además de las máquinas de recarga, ¿qué otro tipo de información considera útil conocer dentro de una estación?

Me sería útil conocer qué servicios tengo cerca dentro de la estación, por ejemplo, cajeros automáticos o tiendas como Oxxo o Pronto. A veces necesito acceder a estos lugares, pero no tengo referencias claras de dónde están.

5. ¿Existen dificultades para determinar su ubicación dentro del andén después de bajar del tren?

Si muchas veces, cuando bajo del tren, no sé realmente si estoy al principio, al medio o al final del andén. Eso dificulta la orientación y saber hacia dónde debo continuar.

6. ¿Qué información considera que podría ayudarlo a orientarse mejor al momento de salir de una estación?

Me ayudaría que la aplicación pudiera indicarme cuáles son las salidas de la estación y a qué dirección corresponde cada una. Así me da más libertad de elegir a qué salida tengo que tomar o hacia donde me debo dirigir.

7. Pensando en una solución tecnológica para apoyar el desplazamiento, ¿qué tipo de dispositivo considera que podría ser útil?

Creo que podría ser útil una cámara externa, acompañada de un altavoz. Ya que al usar el teléfono constantemente igual corro el riesgo a que me puedan robar, pero es más cómodo igual.

8. ¿Hay alguna otra situación dentro del Metro que considere importante abordar mediante una solución tecnológica de asistencia?

Sí, sería útil que la aplicación avisara cuando hay cambios en la estación, como ascensores malos, escaleras cerradas o mucha gente en el camino.