UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

Departamento de Ingeniería en Computación e Informática

Plan del proyecto

Asistente de audio que orienta el desplazamiento entre calles para gente no vidente

Autor(es): Olver Arce

Victor Castro

Ismael Rojas

Asignatura: Proyecto II

Nombre del grupo: SpartAPP

Profesor(es): Diego Alberto Aracena Pizarro

ARICA, 16-12-2021

1. HISTORIAL DE CAMBIOS

Fecha	Versión	Descripción	Autor(es)
05/10/2021	1.0	Versión preliminar del proyecto	Olver Arce Victor Castro Ismael Rojas
10/10/2021	1.1	Avance sobre gestión de riesgos, tareas repartidas.	Olver Arce Victor Castro Ismael Rojas
12/10/2021	1.2	Orden del informe	Ismael Rojas
16/10/2021	1.3	Avance de introducción, planificación y organización del proyecto	Olver Arce Victor Castro Ismael Rojas
19/10/2021	1.4	Avance de suposiciones, restricciones, mecanismos de comunicación y planificación de gestión	Olver Arce Ismael Rojas
23/10/2021	1.5	Corrección de introducción, conclusión y formato de referencia	Olver Arce Ismael Rojas
21/11/2021	1.6	Orden y nuevo contenido añadido para el segundo informe	Ismael Rojas
23/11/2021	1.7	Corrección del informe y avance del análisis	Olver Arce Ismael Rojas
27/11/2021	1.8	Avance sobre el modelo del diseño y	Olver Arce Victor Castro Ismael Rojas

		secuencia	
29/11/2021	**1.9**	**Avance de la implementación y trabajo futuro**	**Olver Arce Victor Castro Ismael Rojas**
06/12/2021	**2.0**	**Corrección del modelo de casos de uso**	**Victor Castro**
07/12/2021	**2.1**	**Corrección del informe 2**	**Olver Arce Victor Castro Ismael Rojas**
16/12/2021	**2.2**	**Revisión final del Informe**	**Olver Arce Victor Castro Ismael Rojas**

2. TABLA DE CONTENIDO

3. INTRODUCCIÓN

La discapacidad es una condición desafortunada de una persona, adquirida durante su gestación, un evento desafortunado, nacimiento o infancia, que son aquellos que tienen deficiencias en el funcionamiento físico, mental o sensorial.**[1]** Entre una de estas discapacidades está la discapacidad visual, donde conlleva problemas en la calidad de la visión de un individuo, estas pueden producir afecciones de distintos grados, que van desde la deficiencia visual hasta la ceguera absoluta.**[2]**

Esta discapacidad lamentablemente produce dificultades para participar en actividades propias en la vida cotidiana, tales como: apreciar los lugares del mundo, disfrutar de entretenimiento visual, o conocer con detalle el mundo y sus cambios. Para enfrentar su discapacidad visual, deben adentrarse a descubrir y construir el mundo por medio de otras sensaciones mucho más parciales, como olores, sabores, sonidos, tacto o ayuda de individuos que le puedan guiar a su ubicación deseada.**[3]**

Es por ello que el grupo “SpartAPP” se encargará de proponer un proyecto que ayude a las personas invidentes, esta será una aplicación para el teléfono que guíe al invidente mediante audio por GPS a lugares de una ciudad. El invidente dirá la dirección y el teléfono móvil transmitirá los kilómetros y trayectorias que faltan, además de decir su ubicación actual, guardar rutas de las ciudades o alertar señales.

Se debe tener claro la planificación inicial, los propósitos, alcances, objetivos, la suposición y las restricciones del proyecto para que no se lleve un lío en el transcurso del trabajo. También se debe considerar la organización de roles donde están definidos los requerimientos y quienes de los participantes del grupo van a hacerlas, estos roles transmitirán su tema mediante mecanismos de comunicación como también las estimaciones de recursos, costos y los procesos que se concebirá en el transcurso de este proyecto y las reglas de gestión de riesgos.**[4]** Para finalizar están los análisis del proyecto, la implementación final y las pruebas del proyecto.

4. PANORAMA GENERAL

Según la OMS, a nivel mundial, por lo menos 2200 millones de personas tienen deficiencia visual o ceguera.**[5]**

La movilidad es uno de los principales retos a los que se enfrentan en su día a día, especialmente para desplazarse en entornos que les resultan poco familiares o desconocidos, lugares donde les invade la inseguridad.

Las personas invidentes tienen una mayor dificultad en poder desplazarse autónomamente, puede ocasionar problemas tales como:

- Realizar una ruta no conocida
- No reconocer direcciones de calles cercanas
- No detectar ciertos obstáculos
- Dificultad en encontrar la salida de emergencia.

4.1. RESUMEN DEL PROYECTO

- Propósito

  El propósito del proyecto es desarrollar una aplicación de teléfono móvil que ayude a una persona no vidente, con esto el invidente puede desplazarse de manera autosuficiente con más seguridad.

- Alcance

  El alcance del proyecto es desde obtener datos del punto en la persona que se encuentre, hasta obtener datos para guiar a la persona a un punto dentro del mapa. Para lograr esto se utilizará el sensor GPS del teléfono móvil para conseguir información de su posicionamiento actual, una base de datos basada en un servidor de aplicación de mapas que ofrece información de rutas por satélite, el micrófono y audio móvil para transmitir la comunicación del invidente con el celular.

- Objetivos

  1. Objetivo general

     Proponer una gestión de guía de rutas de un servidor de mapas implementado en base de datos con audio y GPS en la que el usuario no vidente pueda ser asistido, para que así, el invidente pueda ir hacia los lugares que él desea.

  2. Objetivos específicos
     - Gestionar Rutas para poder usarla más tarde.
     - Narrar la dirección de las calles y orientación de la ruta actual.
     - Alertar de obstáculos, semáforos y pasos peatonales, paraderos.
     - Diseñar una interfaz adecuada que sea precisa y no confusa para el usuario invidente.
     - Realizar pruebas de la aplicación y análisis de resultados para un desempeño adecuado.

- Suposiciones
  - La persona invidente debe tener un teléfono móvil con datos móviles.

- Se asume que la persona solo tiene problemas de visión, y no otros problemas como la discapacidad auditiva, física, etc.
- Se asume que el teléfono móvil de invidente esté funcionando, y tenga la capacidad correcta de usar el GPS, micrófono, redes móviles, capacidad para descargarlo y ejecutar la aplicación según la versión del Android.
- Se asume que el lugar que necesita ir el usuario estará contenido dentro de la base de datos del servidor de mapas, o que este lugar no esté bloqueado. O sea, el usuario dispone de diferentes alternativas (lugares) como para poder seleccionar una ruta o destino. Además de que los mapas deben estar relativamente actualizados.
- Se asume que la tecnología a usar es suficientemente segura como para que no haya errores de rutas y direccionar al usuario a un lugar al azar, y desorientarlo.

- Restricciones
  - El sistema deberá consumir pocos recursos a fin de no acabar la batería del celular en poco tiempo o en medio de la ruta.
  - El sistema que se debe diseñar e implementar, es un sistema de visión para invidentes (ciego o incapacitado visual) no invasivo. **[6]**
  - El sistema a realizar deberá contar con un celular Android y la herramienta de GPS de modo de obtener la ubicación de su localización actual.
  - El sistema deberá servir como asistente para traducir lo que sucede en su navegación, y a la vez necesitará responder a medida que el usuario realice una consulta.
  - El proyecto tiene que ser terminado antes del término semestre académico.

5. ORGANIZACIÓN DEL PROYECTO

El tipo de personal que se debe tener para el proyecto son los siguientes:

- Programador
- Diseñador gráfico
- Jefe de proyecto

5.1. PERSONAL Y ENTIDADES INTERNAS

A continuación, se mostrarán las responsabilidades de cada integrante

Nombre	**Responsabilidad**
Olver Arce	Programador.
Ismael Rojas	Jefe de proyecto.
Victor Castro	Diseñador gráfico.

5.2. ROLES Y RESPONSABILIDADES

- Programador

  Aporta código escalable al proyecto, depurando y codificando según los requerimientos de la App.
- Diseñador gráfico

  Genera diseño preliminar (mockups), a partir de las ideas generadas por el equipo de trabajo, basándose primero en la funcionalidad y luego el estilo.
- Jefe de proyecto

  Revisa y supervisa el progreso de los otros roles, así como los informes, bitácoras y las correcciones de estas, para luego fijar las nuevas tareas que se deberán realizar, en torno al progreso del proyecto.

5.3. MECANISMOS DE COMUNICACIÓN

Para el desarrollo del proyecto, se utilizarán los diferentes mecanismos de comunicación con su respectiva modalidad de uso:

Mecanismo	**Modo de uso**
Correo electrónico (Gmail)	Aplicación de mensajería, es importante para poder acceder a otros servicios en los que es necesario tener una cuenta de correo electrónico, además de enviar archivos correos a utilizar: ● victor.castro.gonzalez@alumnos.uta.cl ● ismael.rojas.flores@alumnos.uta.cl ● olverarce01@gmail.com
WhatsAPP	Aplicación de mensajería, que se utilizará para enviar y recibir mensajes rápidos, así como imágenes, audios, notas de voz, y documentos varios, etc.
Discord	Aplicación de chat, en la que se trabajará en un canal de comunicación, para poder reunirse, coordinar, hablar del trabajo y enviar ideas, enlaces a documentos o investigaciones relacionadas. además de usar la opción que dispone de compartir pantalla
GitHub	Repositorio Online que permite gestionar proyectos y controlar versiones de código, además utilizada para almacenar trabajos y guardar las fechas de los cambios. Enlace repositorio del proyecto: ● https://github.com/olverarce01/AppProyect-II.git

6. PLANIFICACIÓN DE LOS PROCESOS DE GESTIÓN

Este punto se compone de tres secciones: la sección inicial del proyecto en que se especificarán los recursos tanto humanos como hardware-software con la que se trabajará, además de sus costos. La otra sección será la lista de actividades en la que se detallarán las diferentes actividades que se deberán realizar para llevar a cabo el proyecto por el tiempo. Finalmente, la sección planificación de riesgos en la que se especificarán los riesgos dividiéndolos en niveles como: de débil a catastrófico e indicando la posible solución a cada uno de ellos.

6.1. PLANIFICACIÓN INICIAL DEL PROYECTO

- Planificación de estimaciones

  A continuación, se describirán los diferentes recursos hardware-software requeridos para el desarrollo del proyecto:

Recurso	**Producto**
Hardware	Computador Costos energéticos GPS
Software	GitHub Android Studio API

  Cada elemento será estimado por medio de sitios de comercio electrónico, empresas productoras y distribuidoras, etc. Finalmente, directamente de los sitios de las empresas que disponen servicios de programas.

Elemento	**Cantidad**	**Costo**
Computador portatil	3 unidades	$500.000
Teléfono móvil Android con GPS	3 unidades	$200.000
Costo CGE	3 boletas	$4.600
GitHub	3 cuentas	$0
Google docs	3 cuentas	$0 (software libre)

Android Studio	3 unidades	$0 (software libre)
API Base de dato “Firebase”	3 unidades	$0 (software libre)
Costo total hardware:		$2.113.800

- Planificación de recursos humanos

Para el desarrollo del proyecto, será necesario de identificar los diferentes roles para aprovechar mejor el tiempo y el factor precio hora: **[7] [8] [9]**

Rol	Número de personas	Precio Hora
Programador	1	$7.385
Diseñador	1	$6.154
Jefe de Proyecto	1	$8.615
Total:		$22.154
Promedio:		$7.384

A continuación, se realizará el costo mensual por cada integrante, usando como valor-hora, el promedio de los precios respecto a cada rol:

	Horas	Valor
Hora de trabajo	1 hora	$7.384
Tiempo total de trabajo (Mes)	42 horas	
Costo total por integrante mensual	$310.128	

Luego de analizar los costos de cada integrante, se calcula los costos estimados por el equipo para el proyecto:

Equipo: 3 Integrantes	Costo
Proyección de Sueldo	$10 000

Valor de equipo	$930.384
Costo de Equipo	$940.384
Total, por tiempo del proyecto (3 meses)	$2.821.152

Finalmente, se describen los costos que conlleva el proyecto tanto en recursos hardware software y recursos humanos, considerando la Holgura:

Recurso	Costo
Hardware y software (más Holgura)	$2.113.800 + $70.000
Recursos humanos	$2.821.152
Total	$5.004.952

6.2. LISTA DE ACTIVIDADES

- Actividades de trabajo

  La siguiente tabla trata sobre los tiempos asignados a través del semestre, para poder llevar a cabo las actividades para el proyecto, tanto como el tiempo de duración estimado, como de los responsables de la actividad.

Actividad	Tiempo a dedicar	Responsable(s)
1.- Fase preliminar de la idea propuesta.	1 semana	- Todos los integrantes del proyecto.
2.- Desarrollo de la idea y las posibles herramientas a utilizar en ella.	1 semana	- Todos los integrantes del proyecto.
3.- Finalización de la presentación del escenario del problema, y el informe de esta.	1 semana	- Todos los integrantes del proyecto.
4.- Aprendizaje Android Studio con Git.	2 semanas	- Olver Arce - Victor Castro
5.- Modelos gráficos UML, Entidad Relación, Mockups.	1 semana	- Olver Arce - Ismael Rojas

6.- API Google maps y prueba funcionamiento.	1 semana	- Olver Arce - Ismael Rojas
7.- Filtro de datos en un determinado radio de distancia.	1 semana	- Olver Arce - Victor Castro
8.- Agregar asistente de voz de datos.	1 semana	- Olver Arce - Victor Castro
9.- Encontrar un sitio.	1 semana	- Todos los integrantes del proyecto.
10.- Dirigir al usuario.	2 semanas	- Todos los integrantes del proyecto.
11.- Guardar sitios preferidos y frecuentes del usuario.	1 semana	- Ismael Rojas - Victor Castro
12.- Pruebas de Usabilidad.	1 semana	- Todos los integrantes del proyecto.
13.- Documentación y formas de uso.	1 semana	- Olver Arce
14.- Corrección del informe final.	1 semana	- Ismael Rojas

- Asignación de tiempo

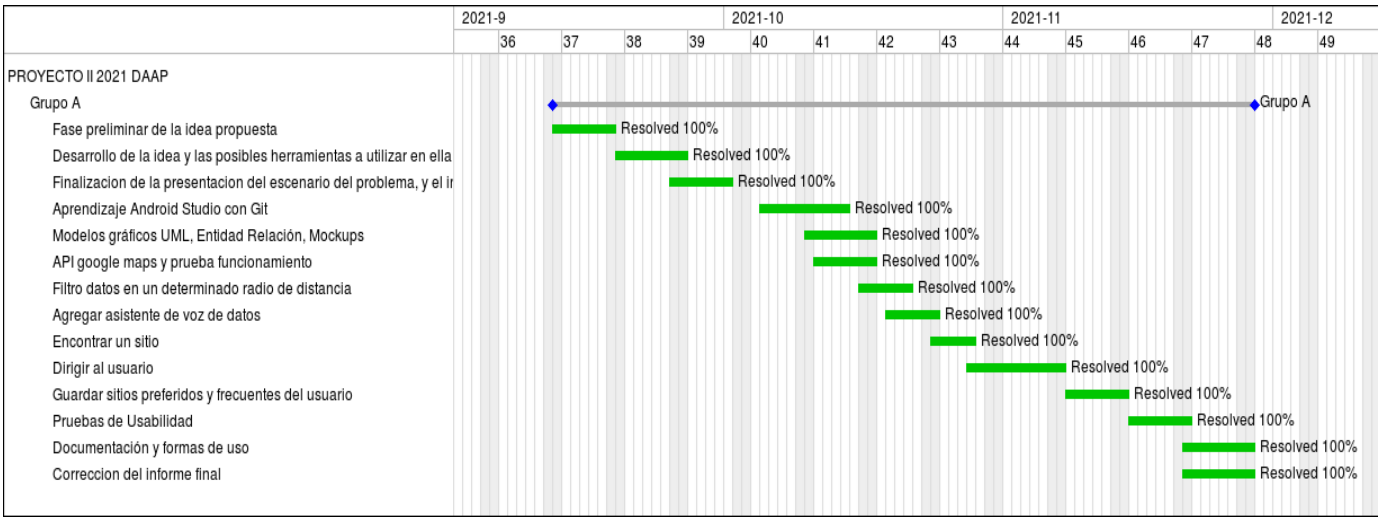


Figura 1: Representación de los Tiempos, Carta Gantt

6.3. PLANIFICACIÓN DE RIESGOS

Para tomar en cuenta los factores que pudiesen interrumpir el buen desarrollo del proyecto, a continuación, se describe la categoría de los riesgos, y se enlistan:

Categoría de Riesgos	
1	Catastrófico
2	Crítico
3	Marginal
4	Despreciable

Riesgo	Probabilidad de ocurrencia	Nivel de impacto	Acción remedial
Falta en el tiempo de entrega asignado	15%	1	Comunicar al profesor sobre la situación y explicarle los motivos de la falta cometida.
Salida, enfermedad o accidente de un integrante del equipo	30%	2	Repartir las tareas propuestas entre los integrantes presentes en la sesión de trabajo.
Pérdida de conexión del mapa (caso servidor de mapa online)	60%	2	Se debe tener como respaldo un mapa descargado a la anterior actualización del servidor, para que sea reemplazado con el mapa actual, hasta que llegue la conexión.
El uso de Google Maps no produjo los resultados esperados en el proyecto.	30%	3	Se requiere buscar otro servidor de aplicaciones de mapas que encaje con el progreso del proyecto.
Coste de tecnología.	45%	2	El dinero será recolectado a base de cooperación grupal. En caso de que se requiera una mayor cantidad, se debe buscar otra

			alternativa de tecnología.
Falta de entendimiento en los programas usados.	80%	2	El integrante tendrá que investigar y estudiar sobre el tema. Si aún no logra entender, deberá hacer consultas al grupo o al profesor.
Mala organización en las juntas de trabajo.	50%	2	El grupo deberá organizar juntas con más preocupación y una fecha en que todo el grupo pueda asistir si o si, en caso de que un integrante aún no puede asistir y sin su debido justificado se le penalizará.
Aplicación no soporta a las otras versiones.	20%	1	La aplicación debe cambiar o actualizar a una versión en el cual logre soportar una mayor cantidad de versiones de celulares.

7. ANÁLISIS

Esta sección detalla el análisis a fondo del proyecto, establecido mediante diagramas donde el sistema hace o requiere de casos para que se produzca la interacción que se desea obtener, transmitido mediante un modelo, la descripción de sus casos de uso, la interfaz y la especificación de requerimientos funcionales y no funcionales.

7.1. MODELO DE CASOS DE USO

En la figura delante se puede observar el caso de uso del proyecto, se puede ver cómo el usuario invidente interactúa mediante el asistente de audio y el sistema lo progresa.

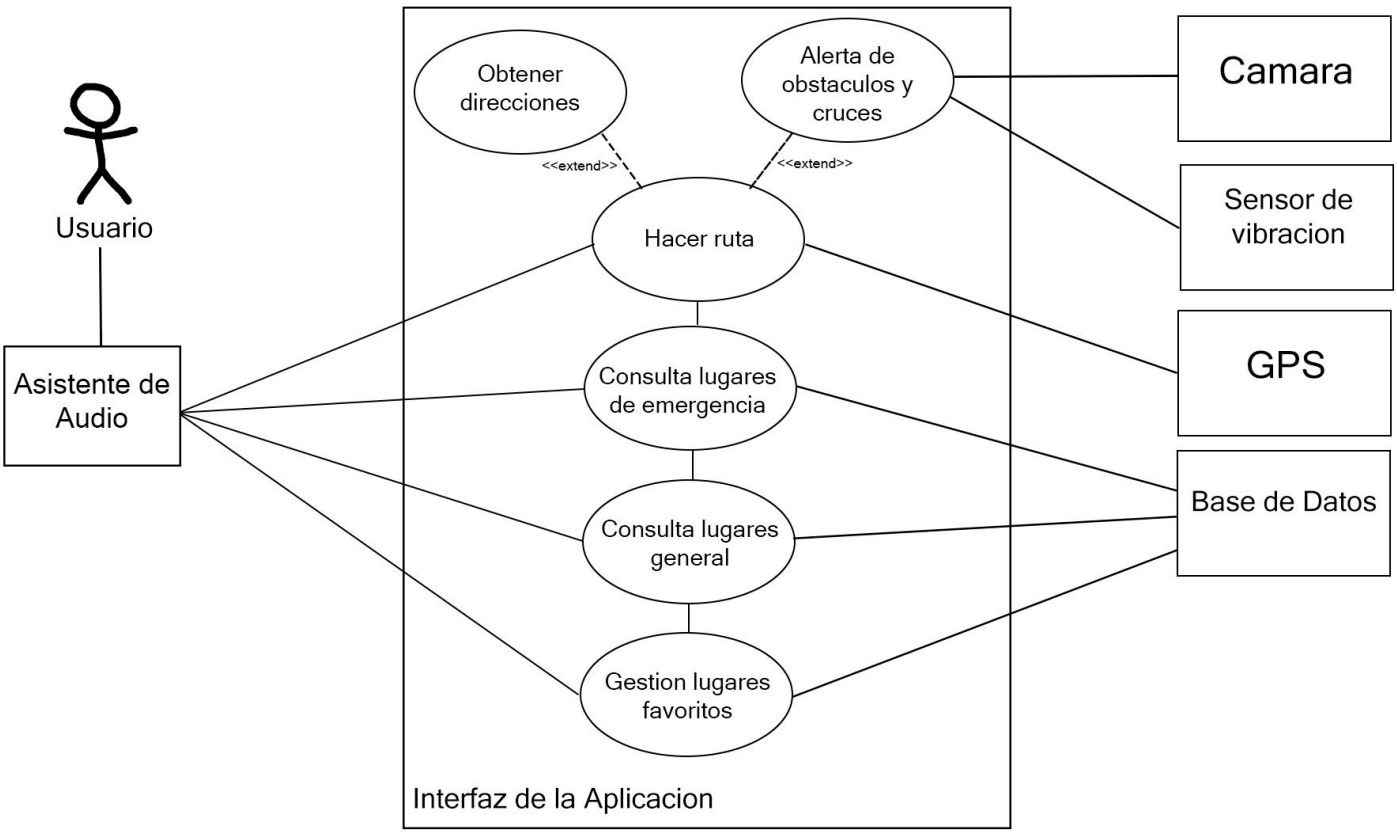


Figura 2: Diagrama casos de uso del proyecto

A continuación, se presentan la definición de los casos de uso del sistema:

<table>
<tr><td colspan="3">Nombre CUS: Consulta Lugares De Emergencia.</td></tr>
<tr><td colspan="3">Autor/Fecha: Olver Arce/13-11.</td></tr>
<tr><td colspan="3">Descripción: Permite obtener información de algunos lugares de emergencia.</td></tr>
<tr><td colspan="3">Actor: Usuario, Asistente de audio, Base de datos.</td></tr>
<tr><td colspan="3">Precondición:
- Lista de lugares de emergencia.</td></tr>
<tr><td>Flujo principal: Usuario, Asistente de audio.

1. El usuario realiza una consulta de lugares de emergencia, hablando por el asistente de audio: (“consulta de emergencia(nombre, tipo emergencia, radio de búsqueda)”).

4. El usuario selecciona un lugar de la lista, hablando por el asistente de audio: (posición, nombre) respectivo.

6. Inicia ruta, hablando por el asistente de audio: (“Iniciar ruta a {nombre del lugar}”). <<extiende>> Hacer ruta.</td><td>Flujo principal: Base de Datos.

2. Se cargan los lugares de emergencia y entrega al sistema el nombre, posición, dirección, detalles y tipo de los lugares con coincidencias de lo que dijo el usuario:</td><td>Flujo principal: Sistema.

3. Relata la lista de los lugares de emergencia con coincidencias que se obtuvieron en la Base de datos: (posición, nombre).

5. Relata detalles del lugar seleccionado (nombre, tipo, dirección, detalles, distancia al lugar).</td></tr>
<tr><td colspan="3">Postcondiciones:
- Se obtiene una lista de lugares de emergencia con coincidencias.</td></tr>
<tr><td colspan="3">Valor medible: Tiempo de búsqueda de un lugar de emergencia.</td></tr>
</table>

Nombre CUS: Consulta Lugares De General		
Autor/Fecha: Olver Arce/13-11.		
Descripción: Consulta los lugares dichos por el usuario, el sistema le dará los detalles del lugar y si quiere guardar a favoritos.		
Actor: Usuario, Asistente de audio, Bases de datos.		
Precondición: - Lista de lugares. - Lista de lugares favoritos.		
Flujo Principal: Usuario. 1. Realiza una consulta de lugares, hablando por el asistente de audio: **(“consulta (tipo lugar, radio de búsqueda)”)**. 4. Selecciona un lugar de la lista, hablando por el asistente de audio: **(posición, nombre)** respectivo. 6. Inicia ruta, hablando por el asistente de audio:**(“Iniciar ruta a {nombre del lugar}”)**. <<extiende>> Hacer ruta.	Flujo Principal: Bases de datos. 2. Se cargan los lugares de general y entrega al sistema el **nombre**, **posición**, **dirección**, **detalles** y **tipo** de los lugares con coincidencias de lo que dijo el usuario:	Flujo Principal: Sistema. 3. Relata la lista de los lugares con coincidencias **(posición, nombre)**. 5. Relata detalles del lugar seleccionado **(nombre, tipo, dirección, detalles)**.
Flujo alternativo: Usuario. 1.1. Si la consulta pasa a ser un dato en específico, habla por el asistente de audio: **(“consulta(nombre o**	Flujo alternativo: Sistema. 1.2. Se cargan los lugares de	Flujo Alternativo: Sistema.

dirección)"). 1.5. Inicia ruta, hablando por el asistente de audio: **("Iniciar ruta a {nombre del lugar}")**. <<extiende>> Hacer ruta. 4.1. El usuario decide guardar o modificar el lugar que visitó, en su lista de favoritos. <<extiende>> Gestión Lugares favoritos.	general y entrega al sistema el **nombre**, **posición**, **dirección**, **detalles** y **tipo** de los lugares con coincidencias de lo que dijo el usuario:	1.3. Realiza la búsqueda del lugar **(nombre o dirección)**. 1.4. Relata detalles del lugar seleccionado **(tipo, detalles)**.
Postcondiciones: - Se obtiene una lista de lugares con coincidencias.		
Valor medible: Tiempo de búsqueda de un lugar		

Nombre CUS: Hacer Ruta.		
Autor/Fecha: Olver Arce/13-11.		
Descripción: Permite la asistencia de direcciones para desplazarse a un lugar señalado.		
Actor: Usuario, Asistente de audio, GPS.		
Precondición: - Lugar (nombre, dirección).		
Flujo Principal: Usuario.	Flujo Principal: Sistema.	Flujo Principal: GPS.

3. Confirma la ruta hablando con el asistente de audio: **(“confirmó iniciar la ruta”)**.	1. Relata detalles del lugar seleccionado. 2. Solicita confirmación para iniciar la ruta. 4. Inicia la ruta y relata indicaciones para ir a la ruta <<extiende>> Obtener direcciones. 5. Alerta de Obstáculos, Cruces y pasos peatonales. <<extiende>> Alerta de obstáculos y cruces.	
6. Solicita dirección actual, hablando con el asistente de audio: **(“¿cuál es mi dirección?”)**	8. Relata **dirección actual**.	7. El sensor GPS del teléfono entrega **nombre, sentido, distancia** y **dirección** de la dirección actual al sistema.
9. Solicita distancia al destino, hablando con el asistente de audio: **(“¿qué distancia falta?”)**.	10. El sistema hace una diferencia entre la distancia de base de datos anteriormente recibida con la distancia del sensor gps, y relata la **distancia al destino**. 11. Relata **(“se ha llegado al destino”)**.	
Postcondiciones: - Finaliza la ruta.		
Valor medible: Tiempo de desplazamiento a un lugar.		

<table>
<tr><td colspan="3">Nombre CUS: Obtener direcciones.</td></tr>
<tr><td colspan="3">Autor/Fecha: Olver Arce/13-11.</td></tr>
<tr><td colspan="3">Descripción: Permite la asistencia con direcciones para desplazarse a un lugar.</td></tr>
<tr><td colspan="3">Actor: Usuario, Asistente de audio, GPS.</td></tr>
<tr><td colspan="3">Precondición:
- Lugar (nombre, dirección).</td></tr>
<tr><td>Flujo Principal: Usuario.</td><td>Flujo Principal: Sistema.

1. Relata (sentido, dirección), del usuario al final, sugiriendo una corrección en el desplazamiento (girar ciertos grados y avanzar).</td><td>Flujo Principal: GPS.</td></tr>
<tr><td colspan="3">Postcondiciones:
- El usuario recibe indicaciones para continuar la ruta.</td></tr>
<tr><td colspan="3">Valor medible: Tiempo de desplazamiento a un lugar.</td></tr>
</table>

<table>
<tr><td colspan="3">Nombre CUS: Alerta de obstáculos y cruces.</td></tr>
<tr><td colspan="3">Autor/Fecha: Olver Arce/13-11.</td></tr>
<tr><td colspan="3">Descripción: Permite la alerta frente a objetos y señaléticas que contiene la ruta.</td></tr>
<tr><td colspan="3">Actor: Usuario, Asistente de audio, GPS.</td></tr>
<tr><td colspan="3">Precondición:
- Lugar (nombre, dirección).
- Cámara del móvil.</td></tr>
<tr><td>Flujo Principal: Usuario.
1. El usuario enfoca la cámara hacia la ruta.</td><td>Flujo Principal: Sistema.

2. Identifica y relata los obstáculos y señales en frente de la ruta (“hay un obstáculo, cuidado”).

3. Identifica y relata los cruces de autos y cruces peatonales (“hay cruce,</td><td>Flujo Principal: GPS.</td></tr>
</table>

	cuidado").	
Postcondiciones: - El usuario recibe alertas para que lo tenga en cuenta.		
Valor medible: Precaución al desplazarse.		

Nombre CUS: Gestión Lugares favoritos		
Autor/Fecha: Olver Arce/13-11.		
Descripción: Permite la gestión de lugares favoritos.		
Actor: Usuario, Asistente de audio, Bases de datos.		
Precondición: - Lista de lugares favoritos.		
Flujo Principal: Usuario. 1. Solicita guardar un lugar, hablando con el asistente de audio: **(" Guardar lugar (nombre, tipo, dirección, detalles) a favoritos")**.	Flujo Principal: Bases de datos. 2. Guarda el lugar en la base de datos: **nombre**, **tipo**, **dirección**, **detalles** a lugares favoritos de la base de datos.	Flujo Principal: Sistema. 3. Relata al usuario que el lugar se guardó exitosamente en la lista de lugares favoritos.
Flujo Alternativo: Usuario. 1.1. Solicita buscar un lugar, hablando con el asistente de audio: **("consulta (nombre, tipo, dirección)")**. 1.4. Selecciona un lugar de la lista, hablando con el asistente de audio:	Flujo Alternativo: Bases de datos. 1.2. Se cargan los lugares de general o favoritos y entrega al sistema el **nombre**, **posición**, **dirección**, **detalles** y **tipo** de los lugares con coincidencias de lo que dijo el usuario:	Flujo Alternativo: Sistema. 1.3. Relata una lista de los lugares coincidentes al lugar **(posición, nombre)**.

(posición, nombre) respectivo.		1.5. Relata detalles del lugar seleccionado **(nombre, tipo, dirección, detalles)**.
Flujo Alternativo: Usuario. 1.1.1. Solicita eliminar un lugar, hablando con el asistente de audio: **(“elimina lugar (nombre, tipo, dirección)”)**. 1.1.4. Selecciona un lugar de la lista hablando con el asistente de audio: **(posición, nombre)** respectivo. 1.1.5. Confirma eliminación del lugar, hablando **(“confirmo eliminación”)**.	Flujo Alternativo: Bases de datos. 1.1.2. Se cargan los lugares de favoritos y entrega al sistema el **nombre**, **posición**, **dirección**, **detalles** y **tipo** de los lugares con coincidencias de lo que dijo el usuario: 1.1.6. Elimina el lugar de lugares favoritos.	Flujo Alternativo: Sistema. 1.1.3. Relata una lista de los lugares coincidentes al lugar **(posición, nombre)**. .1.1.4. Relata detalles del lugar seleccionado **(nombre, tipo, dirección, detalles)**. 1.1.6. Relata al usuario que el lugar se eliminó exitosamente en la lista de lugares favoritos.
Postcondiciones: - Se gestiona la lista de lugares favoritos.		
Valor medible: Tiempo de búsqueda de un lugar		

7.2. DESCRIPCIÓN DE ARQUITECTURA

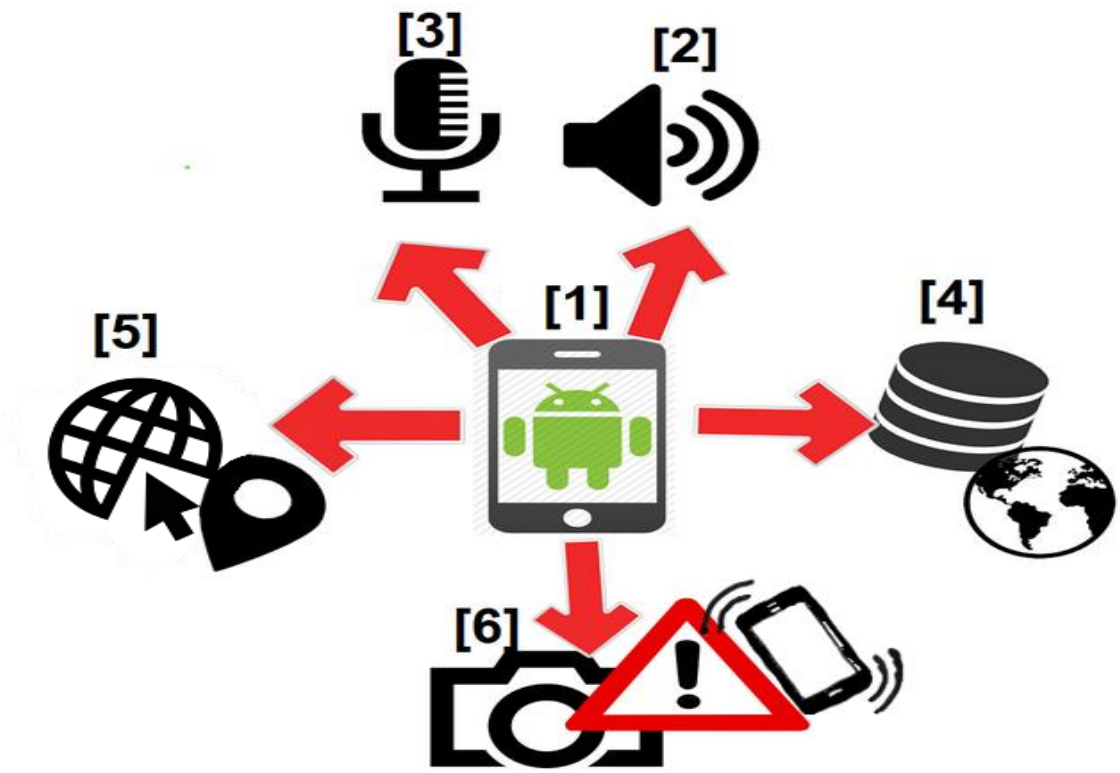


Figura 3: Descripción de arquitectura

En la figura anterior se puede ver la descripción de la arquitectura del proyecto, en la cual señala:

1. Dispositivo móvil Android funcional en la cual señala la interfaz de la aplicación, estará compuesta por botones y campos de texto traducidos a voz.
2. Permisos del sensor del audio del teléfono móvil, señala el audio hecho por la aplicación para que este guíe al invidente.
3. Permisos del micrófono para el dispositivo móvil, para que el invidente diga por voz las instrucciones a la aplicación.
4. Procesamiento de la orden dada por el usuario mediante el uso de la base de datos, basada en el servidor de mapas de Google maps. Se buscan datos coincidentes sobre la dirección dada del usuario.
5. Se produce un Reconocimiento de habla reiteradamente por el usuario a la aplicación y la aplicación al usuario, para esta función se necesitan permisos para tener acceso al internet, y el sensor GPS del teléfono, el usuario entrega instrucciones y la aplicación busca y entrega la información que el invidente desea además de conocer su ubicación actual.

6. Se le brinda un reconocimiento de objetos por cámara por si encuentra un lugar peligroso para el usuario, si lo encuentra el teléfono empezará a vibrar.

7.3. DOCUMENTO DE DISEÑO DE INTERFACE USUARIO

Esta aplicación es destinada para los usuarios no invidentes, por lo tanto, la interfaz debe ser diseñada de una manera sencilla, entendible y que no utilice tanta interacción al usuario.

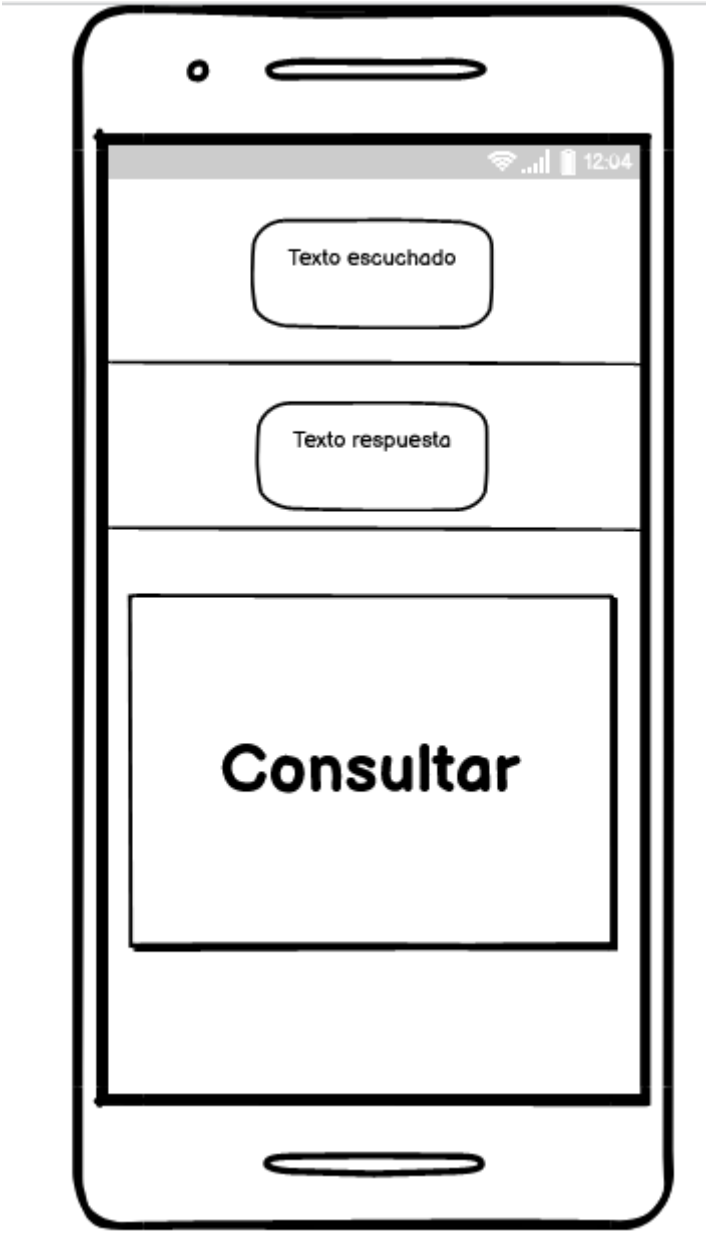


Figura 4: Interfaz principal

Tras iniciar la aplicación, se mostrará la interfaz principal que se ve en la imagen anterior. Tiene un botón, y dos campos de texto. El botón se llama “Consultar” es en donde el invidente puede consultar mediante el audio (entrada de audio) a la aplicación sobre la ruta de lugar que este quiera ir, el invidente crea una “instrucción” para la aplicación.

El primer campo de texto se llama “Texto escuchado” donde el audio producido por el botón “Consultar” pasa a texto que se ve en ese campo, dicho campo es recibido por la aplicación donde deduce lo que quiere decir el usuario, tras terminar de procesarlo lo transmite en otro campo de texto, llamado “Texto respuesta”, este campo de texto transmite una salida de audio para que el usuario pueda oír la respuesta de la aplicación.

El usuario, tras confirmar la ruta que la aplicación señaló, pasa al método “Hacer Ruta”, donde se agrega una nueva función para hacer el llamado de la cámara, lo que hace es iniciar un reconocimiento de objetos mediante la cámara para detectar señales de emergencia frente a objetos y señaléticas que están alrededor. Al decir “Activar cámara”, la interfaz se verá como indica la siguiente figura:

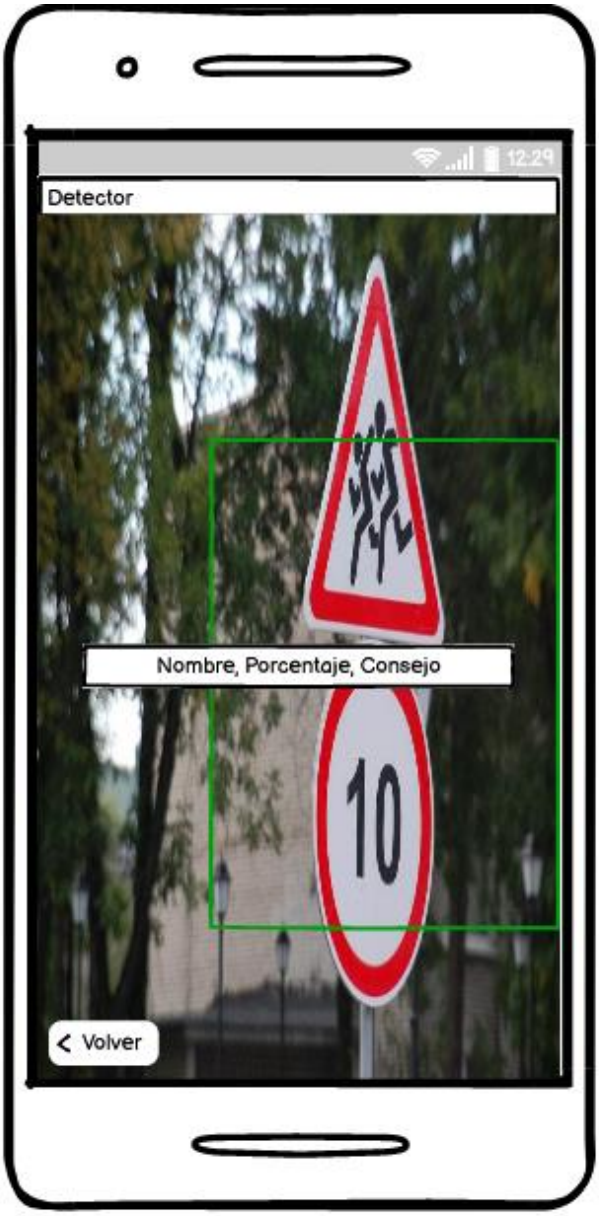


Figura 5: Simulación del reconocimiento de objetos por cámara

Se pone automáticamente la cámara, y empezará el método de reconocimiento de objetos, veremos que al señalar un peligro o una señalética saldrá un campo de texto diciendo el nombre del objeto, el porcentaje del reconocimiento del objeto peligroso y el consejo que es la frase de la aplicación a voz donde señala al usuario que tenga cuidado tras ser reconocido con éxito como un objeto peligroso.

7.4. ESPECIFICACIÓN DE REQUERIMIENTOS

Requerimientos funcionales	Descripción
La aplicación debe utilizar la entrada de audio.	Se usa para que la aplicación reciba la instrucción de la persona no invidente, de donde este quiera ir.
La aplicación debe utilizar la salida de audio.	Se usa para que la aplicación guíe al invidente hacia el destino que este ha pedido.
La aplicación debe utilizar la cámara y la vibración del teléfono móvil.	Se usa para alertar a los usuarios sobre los obstáculos y cruces que pueden poner en peligro al invidente.
La aplicación debe tener acceso al internet.	Se utiliza ya que una API text to speech usa el funcionamiento para hacer que grabe el audio, pero se necesita acceso al internet para que lo haga correctamente.
La aplicación debe utilizar el sensor GPS del teléfono móvil.	Se usa para conocer la ubicación actual del usuario.

7.5. LISTA DE REQUERIMIENTOS NO FUNCIONALES

Requerimientos no funcionales	Descripción
La aplicación debe obtener la ruta del lugar.	Es de vital importancia que la aplicación tenga la ruta para que pueda ser transmitida al invidente, diciendo los detalles y sus coordenadas.
La aplicación debe producir el audio de texto claramente.	Para que el invidente entienda debe producir un audio de tono agradable y precisa.
La aplicación debe ser instalada en Android.	La aplicación debe estar instalada en la versión Android 5 o superior.

8. DISEÑO

El diseño proporciona una organización estratégica de lo que se debe lograr en el proyecto tales como planificar características clave, la estructura y los principales objetivos del proyecto. En esta sección se muestran los diagramas UML anclados del proyecto.

8.1. MODELO DE DISEÑO

Diagrama de clases, con las clases que van a interactuar en el sistema:

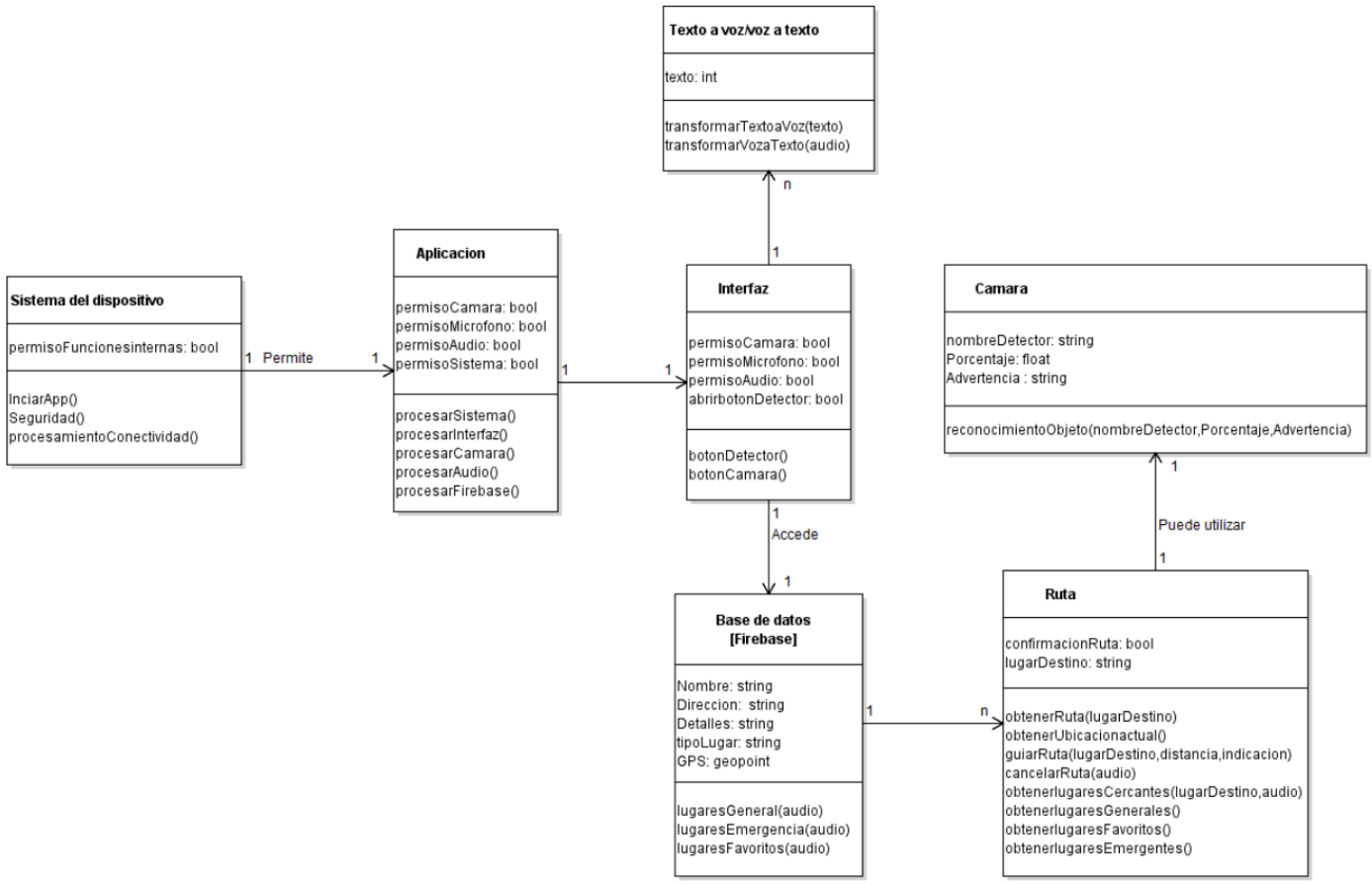


Figura 6: Modelo de clases

8.2. MODELO DE INTERACCIÓN

Diagrama de secuencia cuando el usuario está en movimiento:

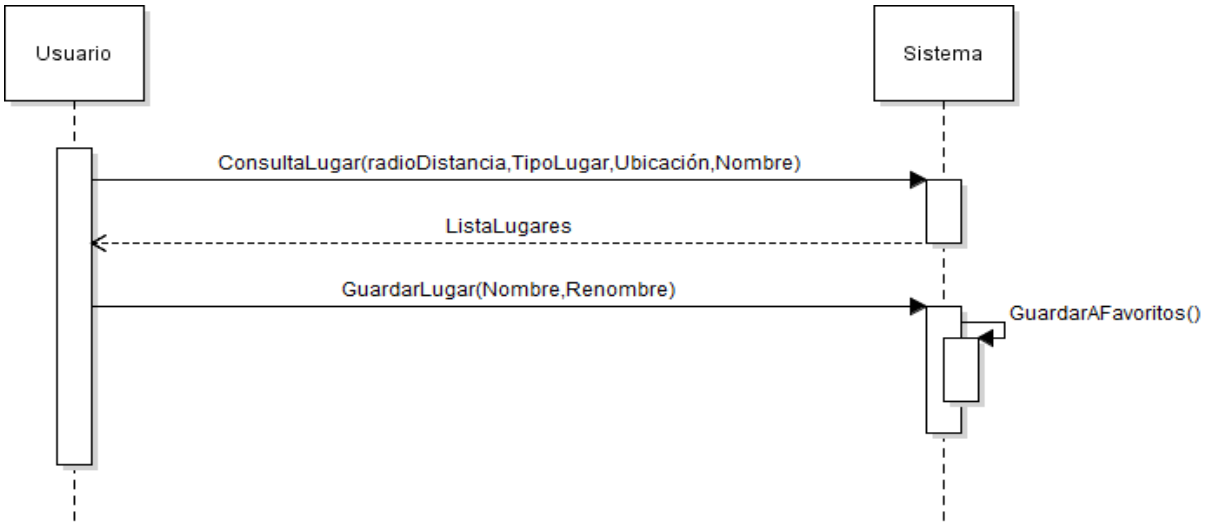


Figura 7: Diagrama de secuencia cuando el usuario está en movimiento

Diagrama de secuencia cuando el usuario consulta un lugar:

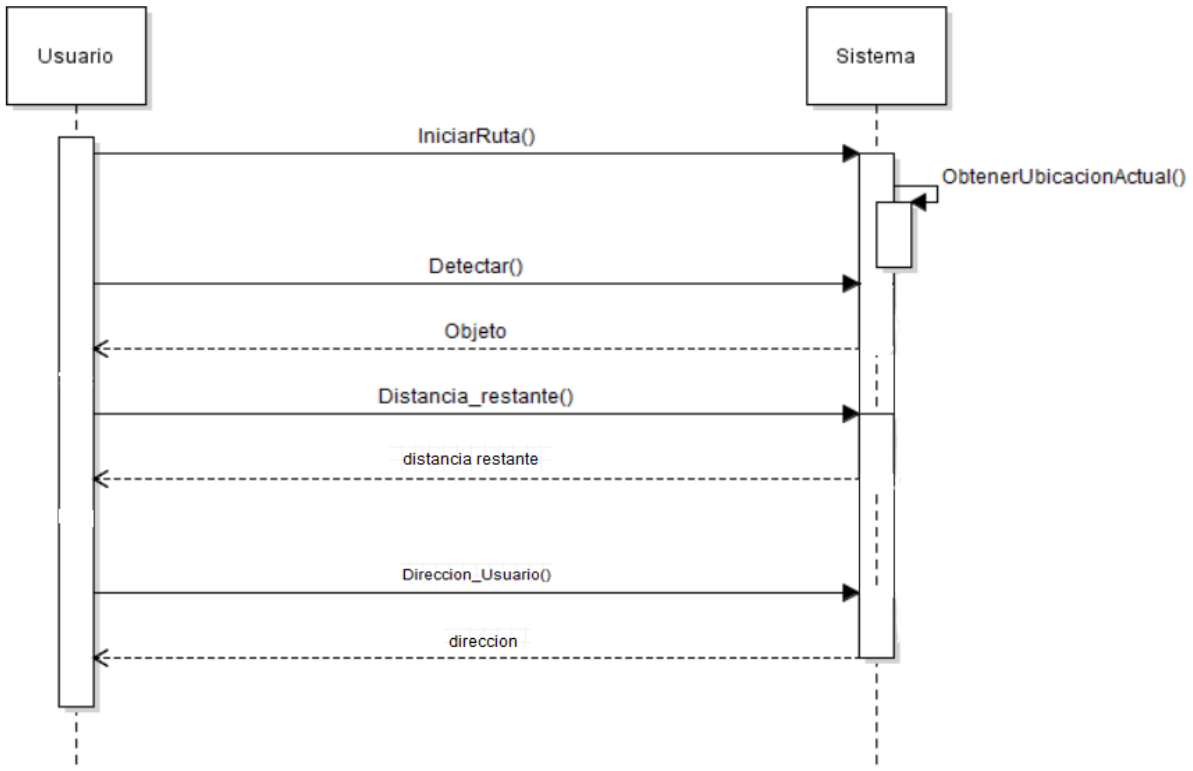


Figura 8: Diagrama de secuencia cuando el usuario consulta un lugar

Diagrama de secuencia cuando el usuario utiliza el reconocimiento de objetos por cámara:

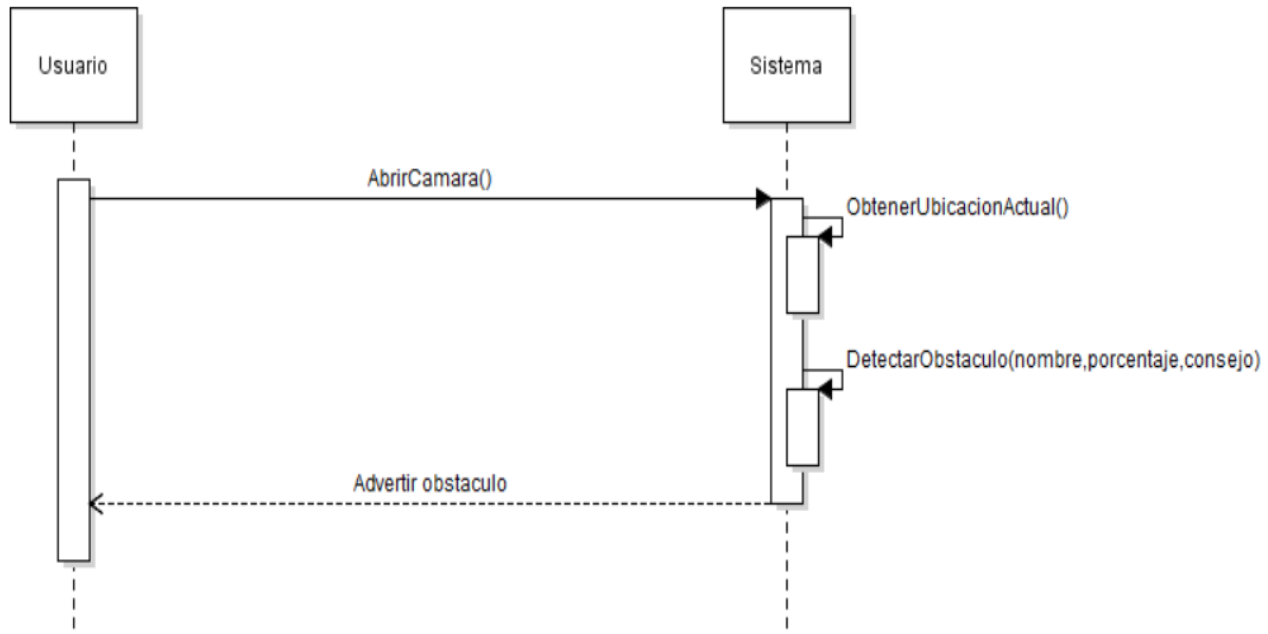


Figura 9: Diagrama de secuencia cuando el usuario utiliza el reconocimiento de objetos por cámara

9. IMPLEMENTACIÓN

La implementación es la idea puesta en ejecución del proyecto, realizando una especificación técnica y códigos de programación que se ponen en práctica al proyecto. Son dadas mediante una especificación o un estándar.

9.1. PLAN DE INTEGRACIÓN

La aplicación cuenta con tres apartados: La interfaz gráfica que permite la comunicación entre la aplicación y el usuario y el reconocimiento de obstáculos.

- **Comunicación entre la aplicación y el usuario**

  La forma en que estos se integra es que el usuario mediante su uso en la interfaz gráfica entrega la entrada de audio del lugar que este quiere, esto se ha podido integrar con la API “Text to speech” y “Recognizer speech” especializado para Android Studio, esas API nos dan la función de repartir audio de voz entre el usuario y el sistema de forma correcta. El audio entregado es transformado en texto y con eso busca similitudes de lugar con la base de datos donde están los lugares de los mapas, la base de datos fue integrada gracias a la plataforma “Firebase” ya que nos da una opción de crear una base de datos de forma gratuita, con lo cual entrega la salida de texto donde se transforma en audio, y con esto puede guiar al invidente a su ruta destinada.

- **Reconocimiento de obstáculos**

  El reconocimiento de objetos es utilizada para alertar por cámara cruces y objetos peligrosos, y es integrada gracias a los componentes de firebase y “cameraX” que proporciona más directa la rotación de las imágenes. En la última actualización se usó la biblioteca “Tensorflow” para ayudar al entrenamiento de imágenes.

9.3. MODELO DE IMPLEMENTACIÓN

En esta aplicación se está utilizando la IDE de Android Studio donde se utiliza el lenguaje Java y Kotlin. Dicho esto, se están usando estas API:

- **Text to speech**

  Esta API convierte una voz en texto, basada en la tecnología IA de Google.

  Nos servirá, ya que utilizar la voz es de vital importancia en este proyecto. El invidente al dar la consulta, esa voz es transformada en texto para que la aplicación trate de guiar al usuario hacia su destino.

- **Localization**

  La localización API es el proceso de renderizar el contenido de su aplicación a otros idiomas y personalizar su aplicación para cada mercado objetivo al que desea apoyar.

  Serviría como opcional por si el invidente habla en otro idioma que no sea español, además de tener otras opciones para el ámbito de programación.

- **Recognizer speech**

  Esta API permite la comunicación hablada entre seres humanos y computadoras.

  Con esto se transmitirá el texto a voz lo que la aplicación quiere mandar al usuario, y así habilita una comunicación hablada entre seres humanos y computadoras, invidente entrega la consulta a voz y la aplicación entrega la dirección de la ruta también a voz.

- **Firebase**

  Es una plataforma de desarrollo de aplicaciones, en este caso, nos proporciona una base de datos API.

  Se utilizará la firebase para las bases de datos, este guardará las direcciones generales como también los lugares de emergencia, estarán los detalles, el tipo de lugar, la latitud y longitud de esa ruta.

- **CameraX**

  Es una biblioteca que proporciona funciones para la app de cámaras, incluyen mejoras de rotación hacia las imágenes, complementos para agregar efectos de cámara y soluciones de compatibilidad.

  Se utiliza para el reconocimiento de objetos y detecta mejor los obstáculos de la cámara.

9.4. MÓDULOS IMPLEMENTADOS

Módulos:

- **Reconocimiento de voz:** En este módulo, se intenta detectar la voz del usuario, cabe destacar que se puede señalar el idioma de la voz a reconocer. Para esto se debe importar RecognizerIntent como en la figura 10.

```
import android.speech.RecognizerIntent;
```

Figura 10: librería RecognizerIntent

En la figura 11, se puede ver que se define una intención de reconocimiento de voz, se le agregan datos extra como el idioma y se inicia una escucha hacia la función “ActivityForResult”.

```
public void hablar(View v){
    Intent hablar = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    hablar.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, "es-MX");
    startActivityForResult(hablar, RECONOCEDOR_VOZ);
}
```

Figura 11: función hablar

En la figura 12, se puede apreciar que dado Cierto código de resultado (tipo de respuesta), código de solicitud (estado solicitud), se logró obtener la data y su posterior paso a cadena de caracteres.

```
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(resultCode == RESULT_OK && requestCode == RECONOCEDOR_VOZ){
        ArrayList<String> reconocido = data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
        String escuchado = reconocido.get(0);
        prepararRespuesta(escuchado);
    }
}
```

Figura 12: función onActivityResult

- **Lectura de una cadena de caracteres:** En este módulo se intenta pasar un texto a voz, para ello se deberá importar TextToSpeech como en la figura 13.

```
import android.speech.tts.TextToSpeech;
```

Figura 13: librería TextToSpeech

En la figura 14, se puede ver la inicialización del lector de texto, definiendo el idioma con que pronunciar texto.

```
mTTS=new TextToSpeech(this, new TextToSpeech.OnInitListener() {
    @Override
    public void onInit(int status) {
        if(status==TextToSpeech.SUCCESS){
            int result=mTTS.setLanguage(Locale.getDefault());
            if(result==TextToSpeech.LANG_MISSING_DATA||result==TextToSpeech.LANG_NOT_SUPPORTED){
                Log.e("TTS","Language not supported");
            }
        }
        else{
            Log.e("TTS","Initialization failed");
        }
    }
});
```

Figura 14: inicialización instancia de TextToSpeech

En la figura 15, se puede se define una función que lee un texto, definiendo "QUEUE_ADD" con lo que, si se realizan múltiples solicitudes de leer, estas podrán guardarse en una cola y ejecutarse de una a la vez.

```
private void speak(String text){
    mTTS.speak(text,TextToSpeech.QUEUE_ADD,null);
}
```

Figura 15: función speak

- **Obtener dirección y ubicación GPS actual:** En este módulo se obtiene la dirección y ubicación GPS del usuario, para ello se necesita importar Location.

```
import com.google.android.gms.location
```

Figura 16: librería Location

En la figura 17, se aprecia la llamada reiterada de parte de "LocationCallback", para que a medida que pase el tiempo, se pueda obtener la localización y dirección más actual del usuario.

```
LocationCallback locationCallback= new LocationCallback() {
    @Override
    public void onLocationResult(LocationResult locationResult) {
        if(locationResult ==null){
            return;
        }
        for(Location location: locationResult.getLocations()) {
            try {
                Geocoder geocoder=new Geocoder(MainActivity.this, Locale.getDefault());
                List<Address> addresses = geocoder.getFromLocation(location.getLatitude(),location.getLongitude(),1);
                myCurrentLocation= new GeoPoint(location.getLatitude(),location.getLongitude());
                myAddress=addresses.get(0).getAddressLine(0);
            } catch (IOException e) {
                e.printStackTrace();
            }

        }
    }
};
```

Figura 17: inicialización instancia LocationCallback

- **Consulta base de datos de lugares**: En este módulo se realizan consultas a la base de datos de la Firestore Database, por lo que para ello se debe implementar Firestore como en la figura 18.

```
import com.google.firebase.firestore
```

Figura 18: librería firestore

En la figura 19, se define una referencia a un documento, con el cual se realizará una consulta, si esta tiene éxito “OnSuccessListener” se devolverá un documento de tipo “DocumentSnapshot” con todos los datos relacionados al documento.

```
DocumentReference docRef=db.collection("lugaresEmergencia").document(id);
docRef.get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
    @Override
    public void onSuccess(@NonNull DocumentSnapshot documentSnapshot) {
        speak(documentSnapshot.getId());
    }
});
```

Figura 19: inicialización instancia DocumentReference

En la figura 20, se puede apreciar otro tipo de consulta en la que, se devolverá una colección de documentos, con lo que dependiendo si “task.isSuccessful()", se podrá acceder a los diferentes documentos e ir realizando operaciones entre ellos.

```
db.collection( contexto)
        .whereEqualTo("tipoLugar", lugar)
        .get()
        .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                if(task.isSuccessful()){
                    int distanciaMenor=999999999;
                    String idMenor = "N.A";
                    for(QueryDocumentSnapshot document: task.getResult()){
                        Map<String, Object> data = document.getData();
                        GeoPoint point= (GeoPoint) data.get("gps");
                        int dis=calculateDistanceByHaversineFormula(myCurrentLocation.getLongitude(),myCurrentLocation.getLatitude(),point.getLongitude(),point.getLatitude());
                        if(dis< distanciaMenor){
                            distanciaMenor=dis;
                            idMenor=document.getId();
                        }
                    }
                    speak("El lugar mas cercano es: "+idMenor+", con una distancia de: "+String.valueOf(distanciaMenor)+" Metros");
                    respuesta.setText("El lugar mas cercano es: "+idMenor+", con una distancia de: "+String.valueOf(distanciaMenor)+" Metros");
                }else{
                    Log.d("xd","Error getting documents:", task.getException());
                }
            }
        });
```

Figura 20: consulta a una colección de documentos

- **Análisis de imágenes de la cámara y detección de objetos:** En este módulo se realiza la detección de objetos por medio de la cámara, para ello se debe implementar “mlkit.vision.objects” y “ImageAnalyser” como en la figura 21 y figura 22.

```
import com.google.mlkit.vision.objects
```

Figura 21: librería mlkit.vision.objects

```
import androidx.camera.core.ImageAnalysis;
```

Figura 22: librería ImageAnalyser

En la figura 23, se puede ver la incorporación de una escucha al proveedor de la cámara, con lo que a medida que ocurran eventos, se ejecutará la función “bindImageAnalysis” la cual analizará lo provisto por la cámara.

```
cameraProviderFuture.addListener(()->{
    try {
        ProcessCameraProvider cameraProvider=cameraProviderFuture.get();
        //bindPreview(cameraProvider);
        bindImageAnalysis(cameraProvider);
    } catch (ExecutionException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

},ContextCompat.getMainExecutor(this));
```

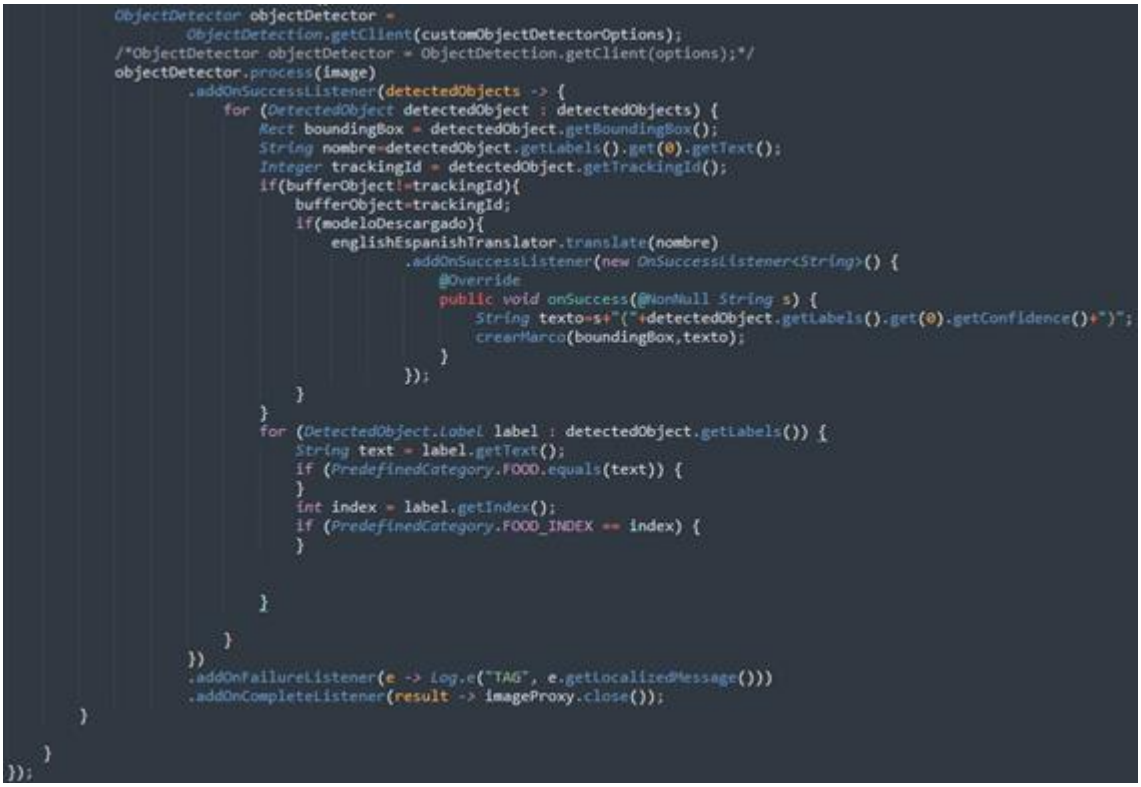
Figura 23: agregación de una escucha

En la figura 24, se define una función en la que en su primera parte, se definen configuraciones, sobre detección de objetos mediante cámara o imágenes, el modelo de reconocimiento de objetos a utilizar, (en este caso “mnasnet_1.3_224_1_metadata_1.tflite”), y ajustes respecto al nivel de confianza de la decisión respecto a reconocer un objeto, máximo de etiquetas o posibles objetos compatibles con el objeto a reconocer.

```
private void bindImageAnalysis(@NonNull ProcessCameraProvider cameraProvider) {
    cameraProvider.unbindAll();
    Preview preview=new Preview.Builder().build();
    CameraSelector cameraSelector = new CameraSelector.Builder()
            .requireLensFacing(CameraSelector.LENS_FACING_BACK)
            .build();
    ImageAnalysis imageAnalysis=
            new ImageAnalysis.Builder()
                    .setTargetResolution(new Size(previewView.getWidth(),previewView.getHeight()))
                    .setBackpressureStrategy(ImageAnalysis.STRATEGY_KEEP_ONLY_LATEST)
                    .build();
    imageAnalysis.setAnalyzer(ContextCompat.getMainExecutor(this), new ImageAnalysis.Analyzer() {
        @Override
        public void analyze(@NonNull ImageProxy imageProxy) {
            int rotationDegrees=imageProxy.getImageInfo().getRotationDegrees();
            @SuppressLint("UnsafeOptInUsageError") Image mediaImage=imageProxy.getImage();
            if(mediaImage!=null){
                InputImage image= InputImage.fromMediaImage(mediaImage,rotationDegrees);
                LocalModel localModel=
                        new LocalModel.Builder()
                                .setAssetFilePath("mnasnet_1.3_224_1_metadata_1.tflite")
                                .build();
                CustomObjectDetectorOptions customObjectDetectorOptions =
                        new CustomObjectDetectorOptions.Builder(localModel)
                                .setDetectorMode(CustomObjectDetectorOptions.STREAM_MODE)
                                .enableClassification()
                                .setClassificationConfidenceThreshold(0.5f)
                                .setMaxPerObjectLabelCount(1)
                                .build();
```

Figura 24: primera parte de función bindImageAnalyser

En la figura 25, se define la segunda parte de la función bindImageAnalyser, en la cual se manda a procesar la imagen (objectDetector.proccess(image)), con lo que, si se tiene éxito, se obtiene una variable de tipo “detectObjects”, con todas las etiquetas de un posible objeto detectado, adicionalmente, de la variable, se puede obtener, el rectángulo que enmarca el objeto detectado, el nombre del objeto detectado, el Id asignado al objeto detectado. Finalmente se pueden realizar otras operaciones.



```
        ObjectDetector objectDetector =
                ObjectDetection.getClient(customObjectDetectorOptions);
        /*ObjectDetector objectDetector = ObjectDetection.getClient(options);*/
        objectDetector.process(image)
                .addOnSuccessListener(detectedObjects -> {
                    for (DetectedObject detectedObject : detectedObjects) {
                        Rect boundingBox = detectedObject.getBoundingBox();
                        String nombre=detectedObject.getLabels().get(0).getText();
                        Integer trackingId = detectedObject.getTrackingId();
                        if(bufferObject!=trackingId){
                            bufferObject=trackingId;
                            if(modeloDescargado){
                                englishEspanishTranslator.translate(nombre)
                                        .addOnSuccessListener(new OnSuccessListener<String>() {
                                            @Override
                                            public void onSuccess(@NonNull String s) {
                                                String texto=s+"("+detectedObject.getLabels().get(0).getConfidence()+")";
                                                crearMarco(boundingBox,texto);
                                            }
                                        });
                            }
                        }
                        for (DetectedObject.Label label : detectedObject.getLabels()) {
                            String text = label.getText();
                            if (PredefinedCategory.FOOD.equals(text)) {
                            }
                            int index = label.getIndex();
                            if (PredefinedCategory.FOOD_INDEX == index) {
                            }


                        }
                    }
                })
                .addOnFailureListener(e -> Log.e("TAG", e.getLocalizedMessage()))
                .addOnCompleteListener(result -> imageProxy.close());
    }

}
});
```

Figura 25: segunda parte de función bindImageAnalyser

En la figura 26, se logra definir una función para lograr mostrar un marco y texto asociado al objeto detectado en cámara, con lo cual este marco se añade a un “frameLayout”.

```
private void crearMarco(Rect boundingBox, String texto) {
    if(frameLayout.getChildCount()>1){frameLayout.removeViewAt(1);}
    marco= new Draw(this,boundingBox,texto);
    frameLayout.addView(marco);
}
```

Figura 26: función crearMarco

En la figura 27, se logra definir una clase (en el lenguaje de programación Kotlin) la cual va a servir para poder dibujar el marco de un objeto detectado, para ello se necesita del contexto o Activity a la cual dibujar, un rectángulo de referencia para el marco y texto a dibujar.

```
class Draw(context: Context?, var rect:Rect, var text:String):View(context) {
    lateinit var boundaryPaint: Paint
    lateinit var textPaint: Paint
    init {
        init()
    }
    private fun init(){
        boundaryPaint=Paint()
        boundaryPaint.color= Color.GREEN
        boundaryPaint.strokeWidth=10f
        boundaryPaint.style=Paint.Style.STROKE

        textPaint=Paint()
        textPaint.color=Color.WHITE
        textPaint.strokeWidth=50f
        textPaint.style=Paint.Style.FILL
        textPaint.textSize=60f
    }

    override fun onDraw(canvas: Canvas?) {
        super.onDraw(canvas)
        canvas?.drawRect(rect.left.toFloat(),rect.top.toFloat(),rect.right.toFloat(),rect.bottom.toFloat(),boundaryPaint)
        canvas?.drawText(text,rect.centerX().toFloat(),rect.centerY().toFloat(),textPaint)

    }
}
```

Figura 27: clase Draw

Traducción de texto:

En este módulo se realiza la traducción de un texto del Idioma Inglés al Español, producto de que la información de un objeto detectado se retornaba en el idioma Inglés, Para ello se debe implementar “FirebaseTranslator” como en la figura 28.

```
import com.google.firebase.ml.naturallanguage.translate.FirebaseTranslator;
```

Figura 28: librería FirebaseTranslator

En la figura 29, se definen opciones sobre el idioma de origen, y el idioma a traducir, adicionalmente se definen condiciones de uso de internet y estado de la descarga del modelo que se utilizará para traducir, adicionalmente se define un estado “modeloDescargado” para verificar antes de cualquier traducción si el modelo efectivamente se descargó.

```
FirebaseTranslatorOptions optionsTranslate=
        new FirebaseTranslatorOptions.Builder()
        .setSourceLanguage(FirebaseTranslateLanguage.EN)
        .setTargetLanguage(FirebaseTranslateLanguage.ES)
        .build();
englishEspanishTranslator=
        FirebaseNaturalLanguage.getInstance().getTranslator(optionsTranslate);
FirebaseModelDownloadConditions conditions = new FirebaseModelDownloadConditions.Builder()
        .requireWifi()
        .build();
englishEspanishTranslator.downloadModelIfNeeded(conditions)
        .addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(@NonNull Void unused) {
                modeloDescargado=true;
            }
        })
        .addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                //model no descargo
            }
        });
```

Figura 29: definición de opciones y condiciones de traducción

En la figura 30, se crea una escucha, con lo cual si tiene éxito, la oración contenida en nombre, se logrará traducir, y realizar posteriormente otras operaciones adicionales.

```
if(modeloDescargado){
    englishEspanishTranslator.translate(nombre)
            .addOnSuccessListener(new OnSuccessListener<String>() {
                @Override
                public void onSuccess(@NonNull String s) {
                    String texto=s+"("+detectedObject.getLabels().get(0).getConfidence()+")";
                    crearMarco(boundingBox,texto);
                }
            });
}
```

Figura 30: añadiendo una escucha para traducir un nombre

Entrenamiento de Imágenes:

Saltamos al google “Colab”, permite al usuario escribir y ejecutar código arbitrario de Python en un navegador. Se utilizó esta opción para hacer el entrenamiento de imágenes. En la figura 31 se puede ver la instalación del creador de modelos.

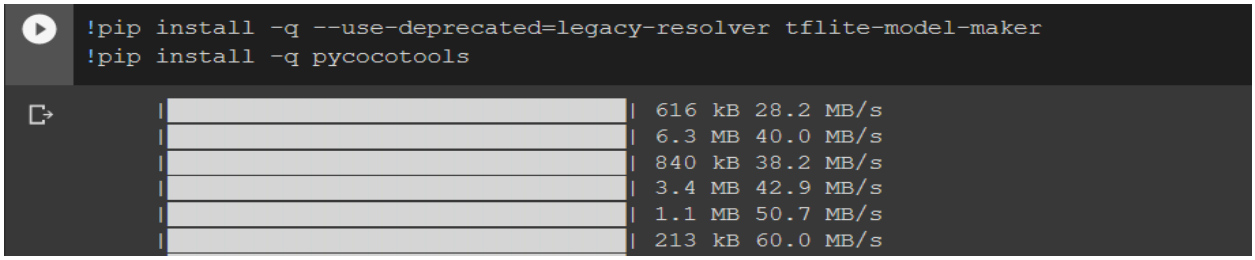


Figura 31: instalación del modelo, Colab

En la figura 32, se importan los paquetes necesarios para el entrenamiento de modelo, para tener más funciones en cuanto a formato de exportación, con etiquetas o sin, optimización, etc. Se instala el API de “Kaggle” para importar el “DataSet” con todas las imágenes necesarias.

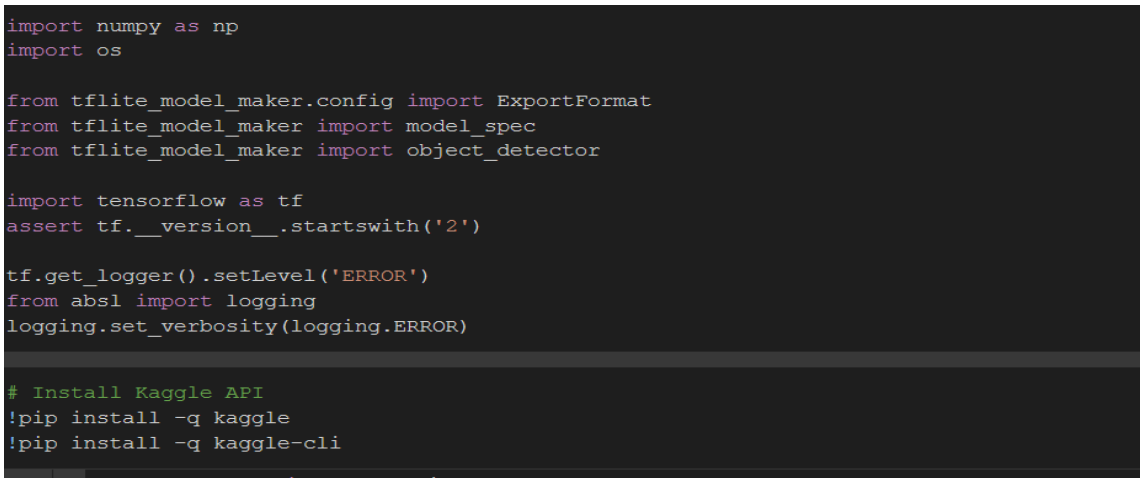


Figura 32: importación del modelo, Colab

En la figura 33, se puede observar la integración de las llaves de acceso a la cuenta Kaggle, y la descarga del DataSet, también se cargan las imágenes de entrenamiento, en que se desea entrenar los pasos de cebra.

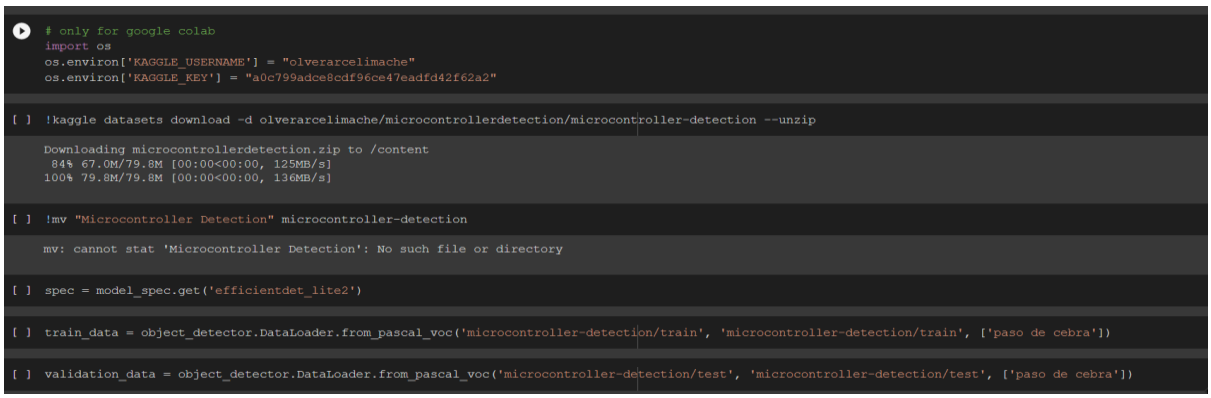


Figura 33: Integración del modelo, colab

En la figura 34, está el procesamiento de entrenamiento de imágenes, se realizó por 50 epoch (demoro 6 horas y media para completar), entre más “epoch” obtiene más precisión.

Figura 34: Entrenamiento de imágenes del modelo, colab

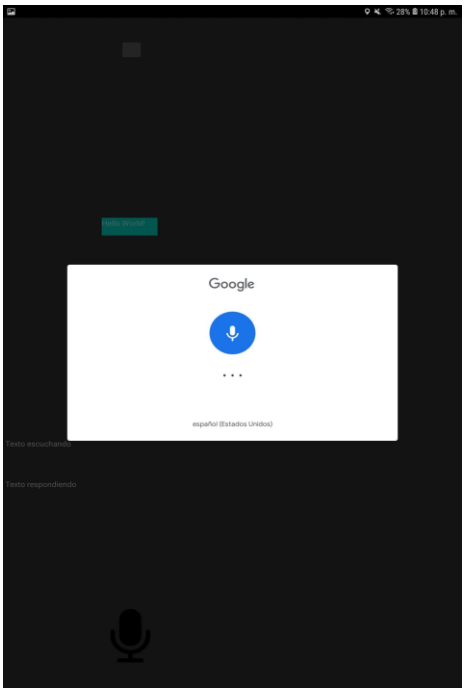
Después de eso, se implementa código para exportar el modelo **[10].** En esta implementación se puede incluir opciones de exportar etiquetas.

```
[ ]  model.export(export_dir='.', export_format=[ExportFormat.SAVED_MODEL, ExportFormat.LABEL])
```

Figura 35: exportación del modelo, colab

9.5. REPORTE DE REVISIÓN

- **Primera prueba**



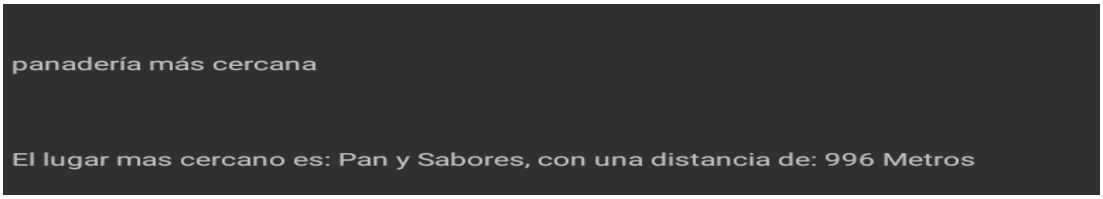
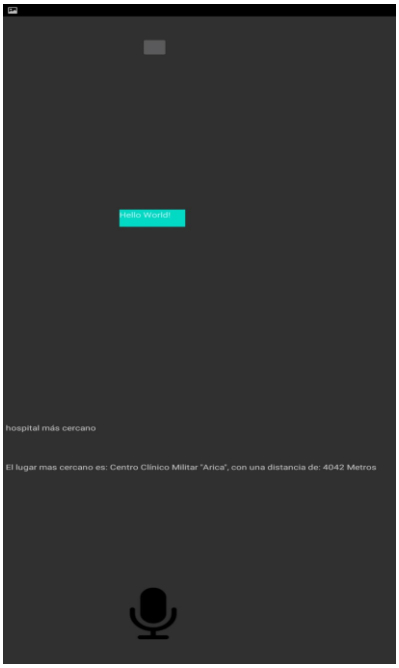


Figura 36, 37 y 38: Primera prueba. 36: Interfaz beta, 37: Prueba funcionamiento Text to Speech, 38: Texto respuesta y Texto escuchado en una panadería

En esta prueba se experimentó en un celular android, se puso a prueba la API text to speech y la base de datos. Esta prueba tuvo como resultado satisfactorio ya que el audio a voz funcionaba en el teléfono móvil, y la aplicación entregó los datos del lugar cercano. En este caso es una panadería.



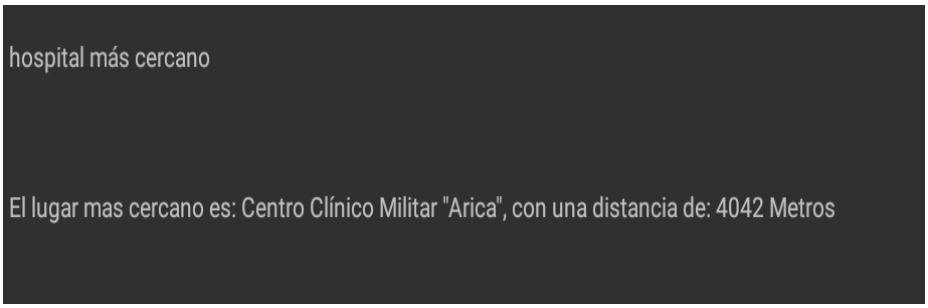
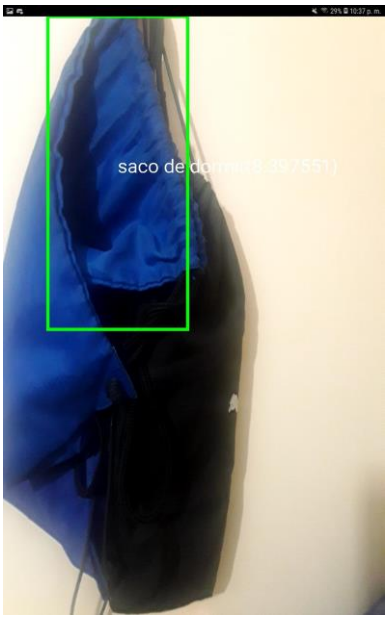


Figura 39, 40: Primera prueba parte 2. 39: Interfaz beta parte 2, 40: Texto respuesta y Texto escuchado en un hospital

También se experimentó los lugares emergentes, como el hospital. Esta prueba resultó ser satisfactoria.

- **Segunda prueba**



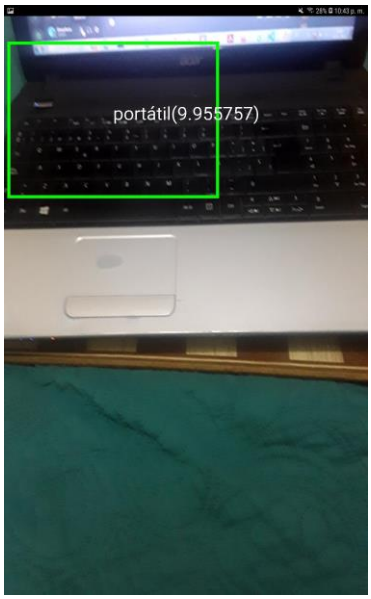


Figura 41, 42: Segunda prueba. 41: Prueba reconocimiento de objeto en un saco, 42: Prueba reconocimiento de objeto en un computador portátil

Se experimentó mediante una cámara del teléfono celular Android el proceso de reconocimiento de objetos. Podemos ver que, si funciona, detectó un saco de dormir, y en la otra figura detecta un portátil. Cabe destacar que el porcentaje que sale en la descripción del objeto es el porcentaje de reconocimiento del objeto.

- **Tercera prueba**

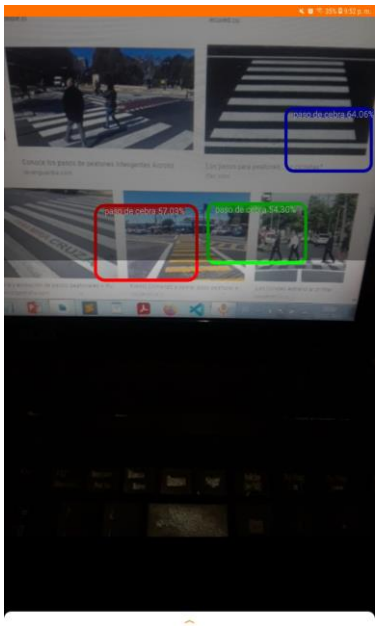


Figura 43: Reconocimiento de objeto actualizado

Se actualizó el reconocimiento de objetos, entrenando con Tensorflow y DataSet, ahora detecta los pasos de cebras y con más precisión.

- **Cuarta prueba**

Figura 44: Interfaz actualizada, preguntando una consulta

Se actualizó la interfaz, se cambió el fondo de pantalla, los campos de texto, y el botón de consultar con el fin de que sea más legible para el usuario.

10. PROBLEMAS ENCONTRADOS

- **Problema 1: API de google maps**
  La API de google maps nos pide tarjeta de crédito para que se pueda utilizar, no permitía prepago, todas las tarjetas probadas por allí fueron rechazadas, se intentó probar la tarjeta de banco estado y aun así tiraba error.
- **Problema 2: API de mapbox**
  La API de mapbox aunque esta es gratuita, en la versión de Android studio la información de como instalarlo es antigua, por lo tanto al implementarlo en la versión actual no detecta los componentes de mapbox.

11. SOLUCIONES PROPUESTAS

- **Solución del problema 1: Cambiar de API**
  Debido a que nos pide tarjeta en Google maps, es mejor buscar otra API, ya que además de rechazar varias tarjetas, nos pide facturación al terminar de "gastar 300$" que dan al verificar la tarjeta, produce desconfianza.
- **Solución de problema 2: Integrar bases de datos mediante firebase**
  Con la base de datos nos permitirá guardar los lugares, su latitud y longitud, donde estas coordenadas son sacadas de Google maps pero guardadas en los datos de firebase.

12. CONCLUSIONES

Para progresar en un proyecto, se necesita analizar varios temas, tales como definir objetivos claros para sentar las bases de la aplicación, definir los alcances de esta aplicación como sus limitaciones con el usuario, identificar los recursos digitales y humanos disponibles y sus costos, identificar las tareas y asignar roles para llevar a cabo estos trabajos, establecer medios de comunicación para mantener el contacto entre los roles, fijar una planificación en la que se realizarán las tareas y así cumplir con los objetivos, finalmente identificar los posibles riesgos que pueden suceder en el transcurso del proyecto y plantear una solución:

- Los objetivos que se tratan son de gran ayuda para tener en claro los fundamentos del proyecto y cómo se va a llevar a cabo el trabajo para lograr esos objetivos.
- Las identificaciones de los roles proporcionan dividir las tareas y así especializar a nuestro equipo según sus habilidades, con lo que se realizará el proyecto e ir identificando las tareas correspondientes a cada uno.
- Los mecanismos de comunicación entregan las formas en cómo utilizar diferentes medios y distribuir la información que se va generando en torno al proyecto.
- Las diferentes planeaciones facilitan la detección de recursos a utilizar, actividades que son las tareas en las que se usarán esos recursos y los riesgos, que son los riesgos de las diferentes actividades con sus recursos.

13. TRABAJO FUTURO

Como es bien sabido, la tecnología innova con cada año que pasa, y pese a que hoy en día las herramientas utilizadas para este proyecto no son tan avanzadas o precisas. Más adelante, cabe la posibilidad de que se retome este concepto y se reutilice para futuros desarrolladores, ya sea por la limitación tecnológica, como por exceder el alcance de este proyecto.

Como continuación del proyecto, existen varios puntos de los cuales se puede seguir desarrollando o implementando, que, debido al límite de tiempo, se tuvo que dejar pendiente, o simplemente inconclusa.

- **Mostrar el mapa con la ubicación del usuario en tiempo real**
  Si bien el concepto visual no es algo que vaya de la mano junto con una persona invidente, si puede ir de parte de un acompañante que no lo fuese, por lo tanto, en ese sentido el concepto visual facilita la comprensión de la aplicación para usuarios que estén acompañando y/o ayudando a otro que sea invidente. Al final la integración de una API de Geolocalización no se utilizó, debido al nulo concepto de ello y por los límites de tiempo.

- **Expandir el idioma**
  Al final la lengua utilizada para el proyecto es netamente español. Por lo tanto, refinar la parte del idioma y desarrollar una opción de esta, sería un buen avance en la aplicación, en términos de alcance del usuario.

- **Lector de texto**
  Si bien, se implementó una función parecida a lo que es el lector de texto, pero no se vio con más detalles, debido al bajo nivel de relevancia que se le dio. Pero para un usuario invidente es muy importante leer lo que se encuentra alrededor, ya sea para guiarse en su trayecto, como por curiosidad. Por lo tanto, la implementación de esta función a futuro, sería de gran ayuda para una persona invidente.

14. REFERENCIAS

1. Guía didáctica para la inclusión en educación inicial y básica(2010), Consejo nacional de fomento educativo. Disponible: https://www.gob.mx/cms/uploads/attachment/file/106810/discapacidad-visual.pdf
2. Guía de apoyo técnico-pedagógico(2007, Dic), Mineduc. Disponible: https://especial.mineduc.cl/wp-content/uploads/sites/31/2016/08/GuiaVisual.pdf
3. Aspectos sobre las actividades acuáticas para personas con discapacidad visual(2009), José Luís Vaquero https://www.munideporte.com/imagenes/documentacion/ficheros/20090309140108Discapacidad_visual-JoseL_Vaquero.pdf
4. Planificación plan de proyecto, Diego Aracena Pizarro. Disponible: http://pomerape.uta.cl/redmine/attachments/download/1781/Propuesta%20FORMATO%20PLAN%20DE%20PROYECTO%202.docx
5. La OMS presenta el primer Informe mundial sobre la visión(2019, Oct), OMS. Disponible: https://www.who.int/es/news/item/08-10-2019-who-launches-first-world-report-on-vision
6. Sistema asistencial para invidentes mediante visión computacional y/o Procesamiento de imágenes, Diego Aracena Pizarro. Disponible:http://pomerape.uta.cl/redmine/attachments/download/1773/Proyecto%20II%20Piloto%202do%20Semestre%202021v1.0.pdf
7. Salario medio desarrollador Android 2021 (2021),talent.com. Disponible: https://cl.talent.com/salary?job=desarrollador+android#:~:text=El%20salario%20promedio%20de%20Desarrollador%20Android%20en%20Chile%20es%20de,anuales%20o%20%247.385%20por%20hora
8. Salario medio para Diseñador Ux 2021 (2021),talent.com. Disponible: https://cl.talent.com/salary?job=dise%C3%B1ador+ux#:~:text=El%20salario%20dise%C3%B1ador%20ux%20promedio,a%C3%B1o%20o%20%246.154%20por%20hora
9. Salario medio para Project Manager 2021 (2021),talet.com. Disponible: https://cl.talent.com/salary?job=project+manager#:~:text=El%20salario%20promedio%20de%20Project%20Manager%20en%20Chile%20es%20de,anuales%20o%20%248.615%20por%20hora
10. Entrenamiento Tensorflow model por Google colab. Disponible: https://www.youtube.com/watch?v=sarZ_FZfDxs