UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

Departamento de Ingeniería en Computación e Informática

Invidentes conociendo su entorno
“BlueBlind”

Autores:

- **Nicolas Barraza**
- **Diego Honores**
- **Juan Rojas**

Empresa: Vision Localizational

Profesor: Diego Aracena Pizarro

Asignatura: Proyecto 2

Arica, Chile 21 de diciembre de 2021

Historial de cambios

Dueño del documento: Empresa “Vision Localizational”

Fecha	Versión	Descripción	Autor(es)
05/10/2021	1.0	Versión preliminar del formato	Nicolas Barraza Diego Honores Juan Rojas
19/10/2021	1.1	Definición inicial de las actividades de trabajo, Avance de la carta Gantt y Asignación del tiempo	Nicolas Barraza Diego Honores Juan Rojas
22/10/2021	1.2	Se avanzó en la definición de riesgos y se hizo arreglos al formato del documento	Diego Honores Juan Rojas
23/10/2021	1.3	Definición de roles, costos, introducción y conclusión.	Nicolas Barraza Diego Honores Juan Rojas
24/10/2021	1.4	Alcance y asignación de roles a actividades de trabajo	Nicolas Barraza
25/10/2021	1.5	Término del avance 1	Nicolas Barraza
26/10/2021	1.6	Correcciones menores	Juan Rojas
02/11/2021	2.0	Inicio desarrollo avance 2	Juan Rojas
14/11/2021	2.1	Descripciones de casos de uso, requisitos funcionales y no funcionales. Actualización del diagrama de diseño.	Juan Rojas
16/11/2021	2.2	Diagramas de secuencia. Descripción inicial de de arquitectura de sistema. Descripción inicial de interfaz de usuario.	Juan Rojas
23/11/2021	2.3	Mejoras en la arquitectura del sistema y la interfaz de usuario. Mejoras en el escenario del problema, objetivos y riesgos.	Diego Honores Juan Rojas
26/11/2021	2.4	Desarrollo de modelo de clases.	Juan Rojas
27/11/2021	2.5	Descripción de la planificación	Juan Rojas

		de la documentación.	
29/11/2021	2.6	Conclusión actualizada.	Juan Rojas
09/12/2021	3.0	Inicio desarrollo informe final	Diego Honores Juan Rojas
10/12/2021	3.1	Planificación de integración Conclusión actualizada.	Diego Honores Juan Rojas
19/12/2021	3.2	Correcciones del informe	Juan Rojas
20/12/2021	4.0	Informe final	Diego Honores Juan Rojas

Índice de contenidos

Índice de ilustraciones

Índice de tablas

1. Panorama general

1.1. Resumen del proyecto:

1.1.1. Introducción:

Con frecuencia, las personas que padezcan de alguna discapacidad visual suelen necesitar asistencia de objetos, animales lazarillos u otras personas que no tengan esa misma condición, para poder realizar todas aquellas actividades que componen su rutina diaria, ya sea ir de un lugar a otro, leer algún texto, realizar compras, entre muchas otras cosas. Con el pasar del tiempo, sin embargo, se han desarrollado diferentes tecnologías de asistencia de visión por computadora que buscan ayudar a facilitar la realización de dichas actividades sin necesidad de requerir ayuda externa. Entre los dilemas a los que se enfrentan personas con discapacidad visual está el de cómo poder trasladarse dentro de un espacio cerrado si no se conoce la distribución del mismo. Es aquí donde surge este proyecto creado por la empresa “Viral Localizational”, que se enfocara en guiar a las personas mediante el uso de sensores.

1.1.2. Escenario del problema:

En el día a día, las personas que presentan discapacidad visual se enfrentan con distintas dificultades al llegar a lugares nuevos, ya sea un restaurante, un supermercado, una casa, entre otros. Entre las dudas que le surgen están:

- ¿Qué dimensión tendrá este lugar?
- ¿Dónde se encontrarán los baños?
- ¿Cómo estará distribuido este lugar?

Normalmente las respuestas o soluciones es que estas personas reciben son poco exactas y confusas, tales como:

- Indicaciones con cantidad de pasos y la dirección.(esto es poco exacto ya que no todas las personas hacen los pasos de una misma longitud , por lo que para algunas personas son más pasos y para otras menos).
- La persona los lleva al lugar.

Con estas respuestas la persona no puede saber cómo llegar a estos sectores cuando está parado desde otro punto.

Ilustración 1. Ejemplo día a día de personas con discapacidad visual.

1.1.3. Escenario de la solución:

Es por ello que como solución se presenta BlueBlind, una aplicación para dispositivos móviles que indica mediante audios, la ubicación de distintos lugares identificados por códigos QR. De esta forma, se dan indicaciones al usuario no vidente para poder guiarse dentro de un espacio determinado.

Ilustración 2. Esquema del escenario de la solución propuesta (1).

Las Indicaciones que se dan son:

- Dimensión del lugar.
- Pisos que posee.
- Divisiones y sectores existentes y sus respectivas ubicaciones.
- De tener más de un piso, ubicación de las escaleras.
- Ubicación del o los baños.

Luego de las indicaciones anteriores, se accede al apartado de dirigir al usuario.

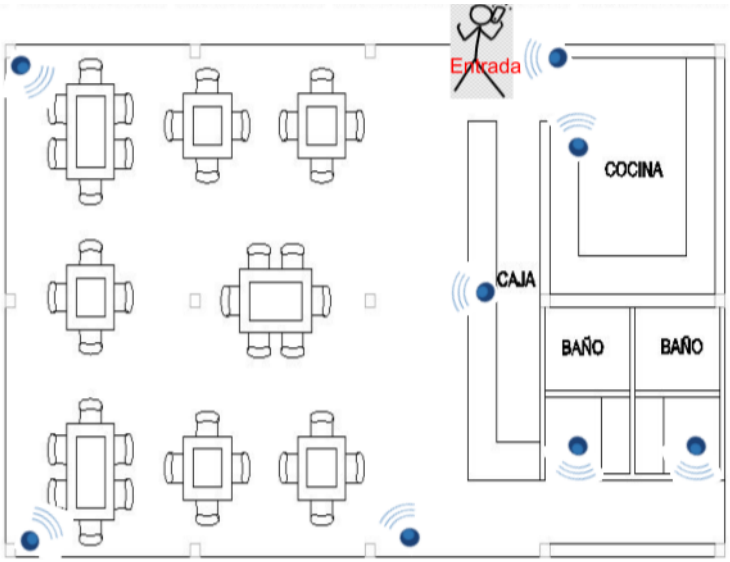

Ilustración 3. Esquema del escenario de la solución propuesta (2).

- Se utilizan códigos QR identificadores ubicados en las distintas secciones del lugar para que cada uno identifique una ubicación.
- El usuario accede al escáner y con la cámara de su dispositivo móvil escanea el código QR identificador.
- Mediante audio se le indica al usuario la ruta más rápida para llegar al punto destino dentro del lugar.

1.1.4. Propósito:

Este proyecto permitirá la creación de una aplicación para dispositivos móviles que ayude a las personas con discapacidad visual a conocer su entorno en espacios reducidos además de guiarlas dentro de estos.

1.1.5. Alcance:

La aplicación de software cuenta con un apartado de guía dentro de un lugar cerrado mediante un lector de códigos QR identificadores, cuya información será dada mediante comandos de audio al usuario. La aplicación será usada en un smartphone con sistema operativo Android.

1.1.6. Objetivo general:

Desarrollar una aplicación móvil enfocada en guiar a personas con discapacidad visual en el desplazamiento dentro de un lugar cerrado.

1.1.7. Objetivos específicos:

- Realizar estudios acerca del escenario de aplicación.

- Estudiar y aplicar la tecnología de creación y lectura de códigos QR para los puntos de control de guía al no vidente.
- Diseñar y desarrollar la aplicación móvil.
- Realizar pruebas del funcionamiento de la aplicación en el escenario determinado.

1.1.8. Suposiciones

Se busca que este proyecto mejore la vida diaria de las personas que poseen dificultades visuales, otorgándoles una mayor autonomía.

1.1.9. Restricciones

- El equipo será de un máximo 3 personas.
- El proyecto se realizará en un periodo de un semestre académico.

1.1.10. Entregables

- Bitácoras semanales.
- Informe de avance.
- Presentación de avance.
- Informe final.
- Presentación final.
- Manual de usuario.
- Wiki del proyecto.
- Producto final.

2. Organización del proyecto

2.1. Personal y entidades internas

- Jefe de proyecto.
- Programador.
- Diseñador gráfico.
- Redactor de documentos.

2.2. Roles y responsabilidades

Tabla 1: Roles y responsabilidades.

Rol	Descripción del rol	Responsable
Jefe de proyecto	Es quien se encarga de organizar y coordinar con el equipo de trabajo, siendo quien toma las decisiones más importantes con respecto al proyecto, tomando en cuenta la opinión del resto del equipo.	Nicolas Barraza
Programador	Es quien realiza el código de la aplicación, además de encargarse de los errores que puedan ocurrir en la misma.	Nicolas Barraza
Diseñador gráfico	Se encarga de realizar los diseños de la aplicación software a desarrollar.	Diego Honores
Redactor de documentos	Responsable de la documentación del proyecto (bitácoras, informe, carta Gantt, etc.)	Juan Rojas

2.3. Mecanismos de comunicación

Entre los distintos mecanismos de comunicación ha utilizar por los miembros del equipo de trabajo, se han destacado los siguientes:

1. **Whatsapp**: Aplicación de mensajería instantánea para teléfonos inteligentes, en la que se envían y reciben mensajes mediante Internet, así como otros archivos. El equipo de trabajo ha creado un grupo de Whatsapp en el cual se utiliza principalmente para determinar los horarios en los que se realizarán las reuniones de trabajo, así como también para comunicar cualquier inconveniente para asistir a las mismas que pueda presentar uno o más miembros del equipo.

Ilustración 4. Icono de WhatsApp.

2. **Discord**: Servicio de mensajería instantánea freeware de chat de voz VoIP, video y chat por texto. Se cuenta con un servidor de Discord en el que se realizan las reuniones de trabajo del equipo, siendo también el medio por donde se planifican las siguientes reuniones, las distintas tareas a realizar, donde se comparte la información relevante para el proyecto y para comunicarse.

Ilustración 5. Icono de Discord.

3. **Zoom**: Medio de telecomunicaciones con el cual se realizan reuniones de trabajo, planificaciones de tareas, se comparte información relevante con el proyecto, entre otras cosas.

Ilustración 6. Icono de Zoom.

4. **Google Drive**: Servicio de alojamiento de archivos en el cuál se tiene creada una carpeta en donde almacenar los archivos de informes, bitácoras, presentaciones, diagramas, entre otros que estén relacionados con el proyecto. En Google Drive es donde el equipo de trabajo se dedicará a desarrollar los informes y presentaciones del proyecto.

Ilustración 7. Icono de Google Drive.

5. **Google Meet**: Servicio de videotelefonía desarrollado por Google. Utilizado en ocasiones especiales cuando se necesite utilizar la opción de compartir pantalla para mostrar algo al resto del equipo, y Discord o Zoom no se encuentren disponibles.

Ilustración 8. Icono de Google Meet.

6. **Redmine**: Herramienta web de gestión de proyectos en la cuál se subirán todos los entregables referentes al proyectos. El equipo de trabajo podrá a su vez editar la carta Gantt que proporciona la aplicación, detallar las actividades y planificarlas en un calendario, administrar la wiki del proyecto, entre otros.

Ilustración 9. Icono de Redmine.

3. Planificación de los procesos de gestión

3.1. Planificación inicial del proyecto

3.1.1. Planificación de estimaciones

Tabla 2: Planificación de estimaciones.

Recurso	Coste individual	Cantidad total	Coste total
Notebooks	$600.000	3	$1.800.000
Teléfono móvil	$150.000	3	$450.000
Android Studio	Gratuito	3	Gratuito
Visual Studio Code	Gratuito	3	Gratuito
Repositorio de documentos Google Drive	Gratuito	1	Gratuito
Documentos de Google (Google Drive)	Gratuito	1	Gratuito
Sueldo de integrantes del equipo	($850.000 por mes) En 3 meses y medio = $2.975.000	3	$8.925.000
Costo total del proyecto	$11.175.000		

3.1.2. Planificación de recursos humanos

Cada uno de los roles preestablecidos será asignado a cierta cantidad distinta de integrantes del equipo por rol.

- **Jefe de proyecto**: Como mínimo se deberá tener 1 jefe de proyecto. Pero todos los integrantes del equipo deberán tener responsabilidades similares con el proyecto.

- **Programador**: 2 integrantes como mínimo deberán encargarse de la programación del software. Todo el equipo debe asegurarse de revisar los códigos del software.
- **Diseñador gráfico**: 1 integrantes se encargará de realizar el diseño de la aplicación software, consultado a los demás integrantes del equipo si es que el diseño es el ideal.
- **Redactor de documentos**: 1 a 2 personas se encargaran de la documentación de bitácoras y carta Gantt. El resto de entregables (el informe de proyecto por ejemplo) deberán ser realizados por todo el equipo completo.

3.2. Lista de actividades

3.2.1. Actividades de trabajo

- **Definición preliminar del proyecto**
  - **Descripción**: Se define de forma preliminar el proyecto. Esto consiste en definir el nombre del mismo, del equipo/empresa encargado de desarrollarlo, y la problemática a abordar
  - **Responsable(s)**: Todo el equipo de trabajo.

- **Definir los escenarios del problema y de la solución**
  - **Descripción**: Preparar el escenario que presente la problemática y una posible solución para resolverla
  - **Responsable(s)**: Todo el equipo de trabajo.

- **Presentar los escenarios del problema y de la solución**
  - **Descripción**: Realizar una presentación que describa los escenarios de la problemática y de la solución propuesta anteriormente definidos.
  - **Responsable(s)**: Todo el equipo de trabajo.

- **Desarrollar el informe del proyecto**
  - **Descripción**: Realizar durante todo el periodo de desarrollo del proyecto un informe acerca del mismo
  - **Responsable(s)**: Todos los miembros del equipo.

- **Realizar la planificación del proyecto**
  - **Descripción**: Desarrollar el primer avance del informe del proyecto, el cual consiste en la planificación de este último.
  - **Responsable(s)**: Todos los miembros del equipo.

- **Diseñar la interfaz de la aplicación software**
  - **Descripción**: Se realiza el diseño de la interfaz de la aplicación software.

- **Responsable(s)**:Diego Honores.

- **Programación**
  - **Descripción**: Programar la aplicación software a desarrollar.
  - **Responsable(s)**: Nicolas Barraza.

- **Estudiar Android Studio**
  - **Descripción**: Aprender a utilizar Android Studio y estudiar el lenguaje de programación Java.
  - **Responsable(s)**: Nicolas Barraza.

- **Realizar pruebas de funcionamiento**
  - **Descripción**: Realizar prueba del funcionamiento de la aplicación software.
  - **Responsable(s)**: Todos los miembros del equipo.

- **Presentar avance 1**
  - **Descripción**: Realizar la presentación del primer avance del informe del proyecto
  - **Responsable(s)**: Todos los miembros del equipo.

- **Manual de usuario**
  - **Descripción**:Realización de manual de usuario, donde se explica cómo se usa la app.
  - **Responsable(s)**: Juan Rojas.

- **Presentar avance 2**
  - **Descripción**: Realizar la presentación del segundo avance del informe del proyecto.
  - **Responsable(s)**: Todos los miembros del equipo.

- **Cierre del proyecto**
  - **Descripción**: Se realizan las últimas actividades referentes al proyecto, terminando el informe e implementando los últimos cambios al software.
  - **Responsable(s)**: Todos los miembros del equipo.

- **Presentar el proyecto**
  - **Descripción**: Se presenta el proyecto de software finalizado.
  - **Responsable(s)**: Todos los miembros del equipo.

3.2.2. Asignación de tiempo

- Planificación del proyecto: De 4 a 5 semanas
- Desarrollo y Ejecución del proyecto: De 6 a 7 semanas
- Cierre del proyecto: De 1 a 2 semanas

3.2.3. Carta Gantt

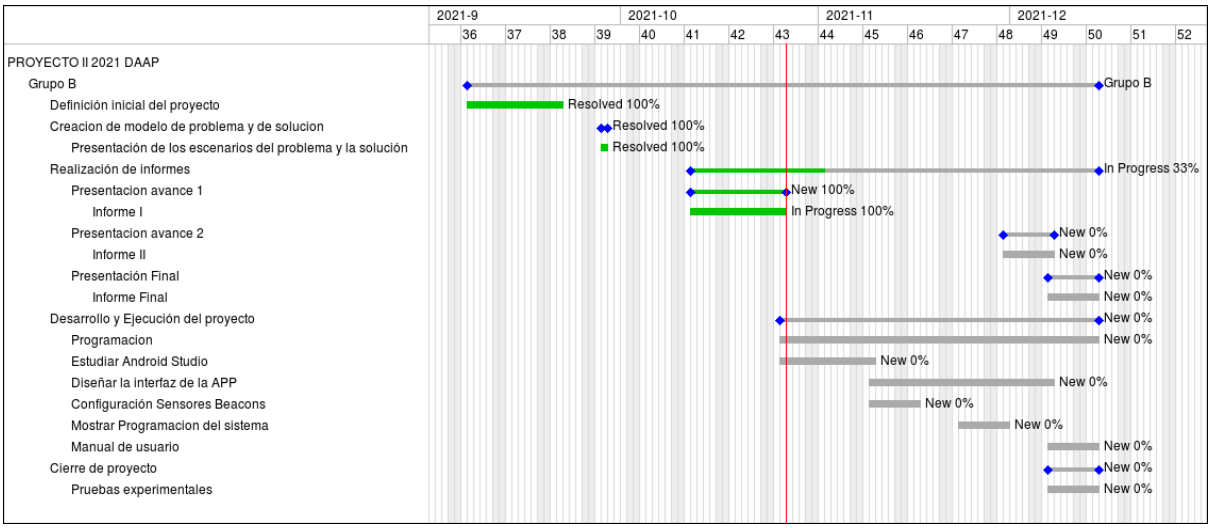

Ilustración 10. Carta Gantt original.

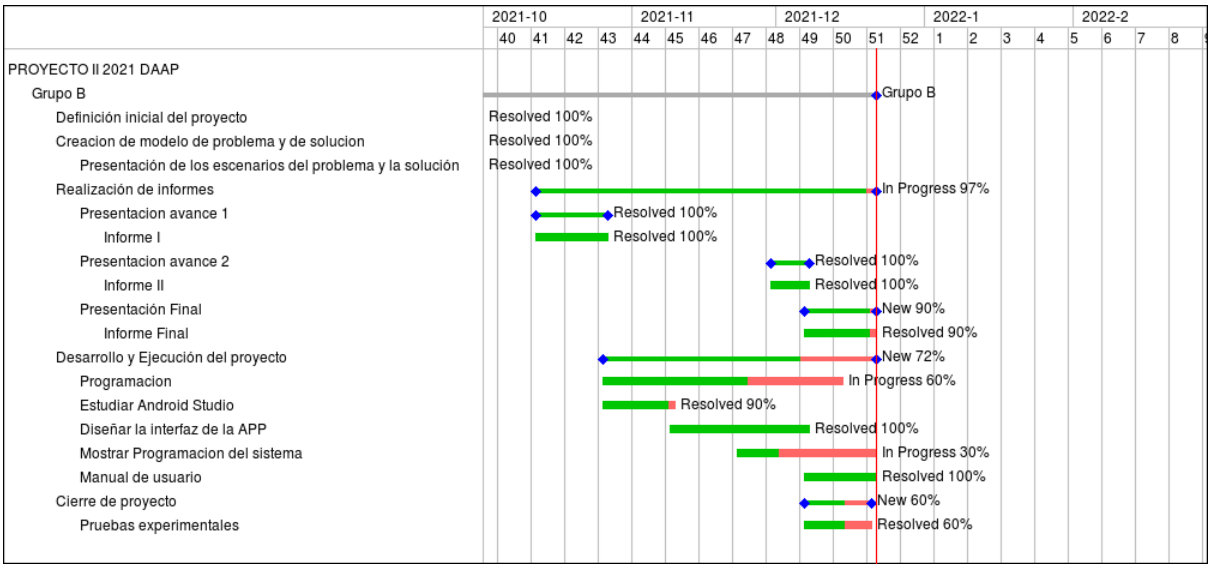

Ilustración 11. Carta Gantt actualizada.

3.3. Planificación de la gestión de riesgos

Tabla 3: Planificación de la gestión de riesgos.

Riesgos	Probabilidad de ocurrencia	Nivel de impacto	Acción remedial
Los teléfonos móviles no son compatibles con el software desarrollado	40%	2	Utilizar herramientas de desarrollo de software compatibles con el sistema operativo de los teléfonos móviles.
Se dañan uno de los notebooks	10%	2	Trabajar en otros notebooks. Asegurarse de que tengan los programas software necesarios para trabajar.
Se pierden los códigos de programación del Software.	20%	1	Asegurarse de hacer copias y respaldarlas para futuros usos.
Uno o más de los integrantes puede enfermarse o no estar disponible para las reuniones de trabajo	15%	2	Redistribuir las tareas de esos integrantes al resto del equipo.
Uno de los códigos de la aplicación, causen errores o problemas en el celular	20%	2	Revisar el código que cause los errores y buscar una posible solución.
Uno de los integrantes no cumple con sus responsabilidades de trabajo	10%	1	Se le realizará advertencias a dicho integrante junto con la posibilidad de expulsión del equipo en caso de que dicho comportamiento se vuelva a repetir.
Se presentan problemas de conexión a los servicios de trabajo en línea (Google Drive, Discord, Whatsapp, etc.)	20%	3	Realizar avances de informes y guardarlos en medios que no requieran de conexión a internet (Microsoft Word por ejemplo). En caso de los medios de comunicación, utilizar llamadas telefónicas o mensajes de texto para comunicarse con el resto del equipo.

4. Planificacion de procesos tecnicos

4.1. Modelo de proceso

4.1.1. Diagrama de casos de uso

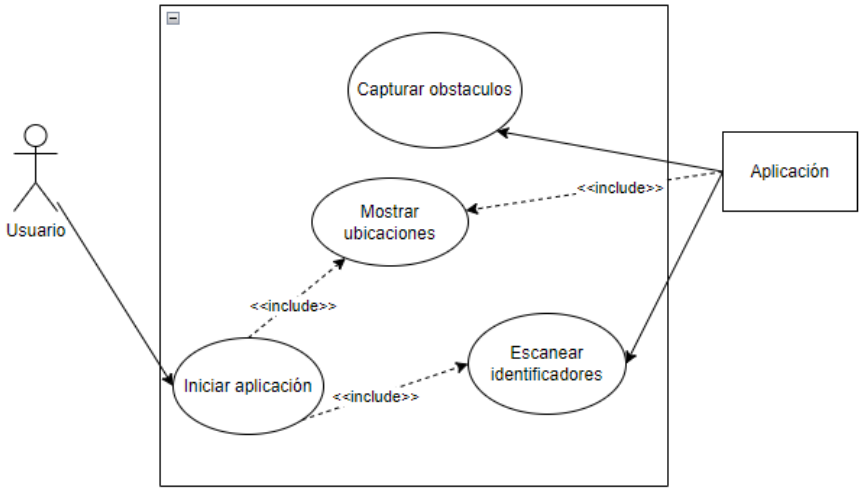

Ilustración 12. Diagrama de casos de uso.

4.1.2. Descripciones de casos de uso de sistema

Tabla 4: C.U.S. "Iniciar aplicación".

Nombre:	Iniciar aplicación
Actores:	Usuario
Descripción:	Permite al usuario iniciar la aplicación
Precondiciones: La aplicación debe estar instalada en el dispositivo móvil del usuario, el cual debe tener acceso al micrófono y cámara.	
Flujo normal	
ACTOR	SISTEMA
1.- El usuario usa un comando de voz para iniciar la aplicación. 3.- El usuario usa un comando de voz para usar el escáner de la aplicación.	2.- La aplicación se inicia 4.- Incluye el caso de uso "**Escanear identificadores**"

Flujo alternativo	
3.1.- El usuario usa un comando de voz para acceder a la lista de ubicaciones.	3.2.- Incluye el caso de uso "**Mostrar ubicaciones**"
3.1.1.- El usuario usa un comando de voz para acceder a la lista de ubicaciones.	3.1.2.- Se cierra la aplicación.
Postcondiciones: La aplicación se encuentra iniciada y lista para realizar sus funcionalidades.	

Tabla 5: C.U.S. "Escanear identificadores".

Nombre:	Escanear identificadores
Actores:	Usuario/Aplicación
Descripción:	Permite que la aplicación escanee los identificadores de las ubicaciones.
Precondiciones: La aplicación debe estar iniciada y con acceso a la cámara y al micrófono del dispositivo móvil.	
Flujo normal	
ACTOR 1.- El usuario acerca la cámara del dispositivo móvil al código QR identificador.	SISTEMA 2.- La aplicación escanea el identificador. 3.- La aplicación calcula la distancia entre el usuario y el identificador 4.- La aplicación menciona por comando de voz la ubicación asociada y su distancia actual.
Flujo alternativo	
	4.1.- El sistema manda un mensaje de error por voz indicando que no se puede obtener la distancia.
Postcondiciones: Se señala la ubicación del destino y su distancia.	

Tabla 6: C.U.S. "Mostrar ubicaciones".

Nombre:	Mostrar ubicaciones
Actores:	Aplicación
Descripción:	Permite que la aplicación muestre la lista de las ubicaciones disponibles
Precondiciones: La aplicación debe estar iniciada y con acceso a la cámara y al micrófono del dispositivo móvil.	
Flujo normal	
ACTOR	SISTEMA 1.- La aplicación muestra una lista de las ubicaciones disponibles. 2.- La aplicación indica por comando de voz cada una de las ubicaciones.
Flujo alternativo	
	2.1.- La aplicación manda un mensaje de error indicando que no hay ubicaciones disponibles.
Postcondiciones: Ninguna.	

Tabla 7: C.U.S. "Capturar obstáculos".

Nombre:	Capturar obstáculos
Actores:	Aplicación
Descripción:	Permite a la aplicación capturar los obstáculos que se presenten en el camino entre el usuario y su destino.
Precondiciones: La aplicación debe estar iniciada y con acceso a la cámara y al micrófono del dispositivo móvil.	
Flujo normal	
ACTOR 1.- El usuario apunta la cámara al frente suyo.	SISTEMA 2.- La aplicación empieza a capturar imágenes de los obstáculos en frente del

	camino. 3.- La aplicación indica mediante voz los obstáculos que se encuentren de frente.
Flujo alternativo	
Postcondiciones: Ninguna.	

4.1.3. Diagramas de secuencia

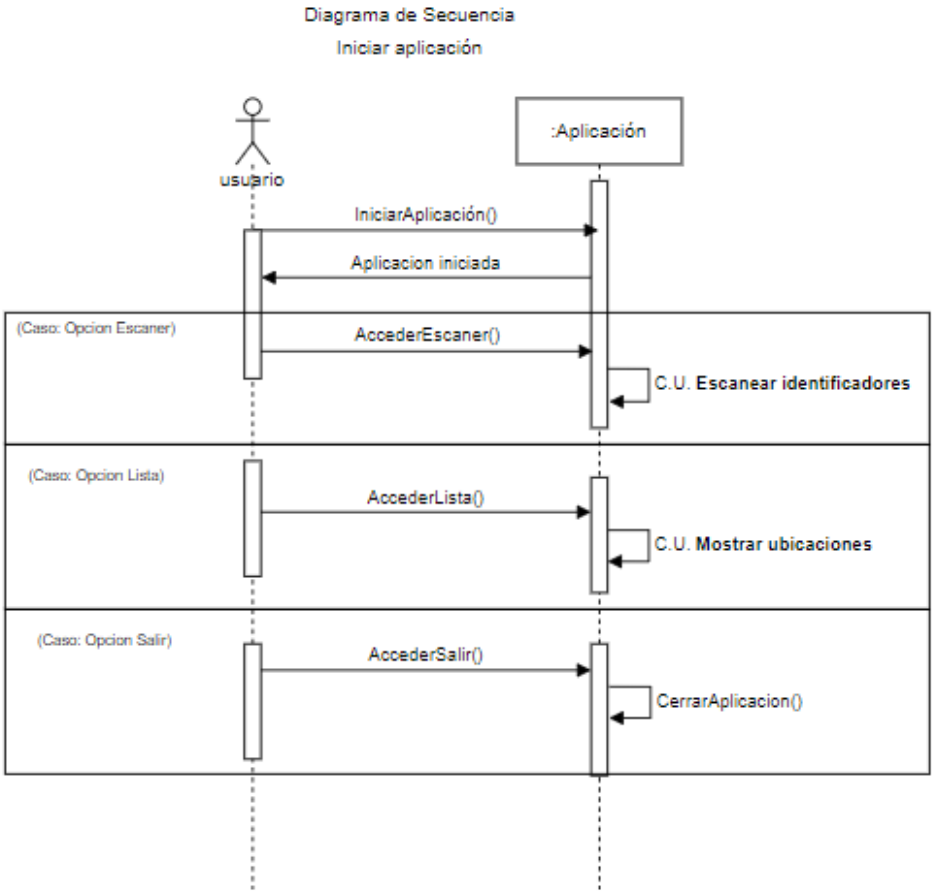

Ilustración 13. Diagrama de secuencia "Iniciar aplicación".

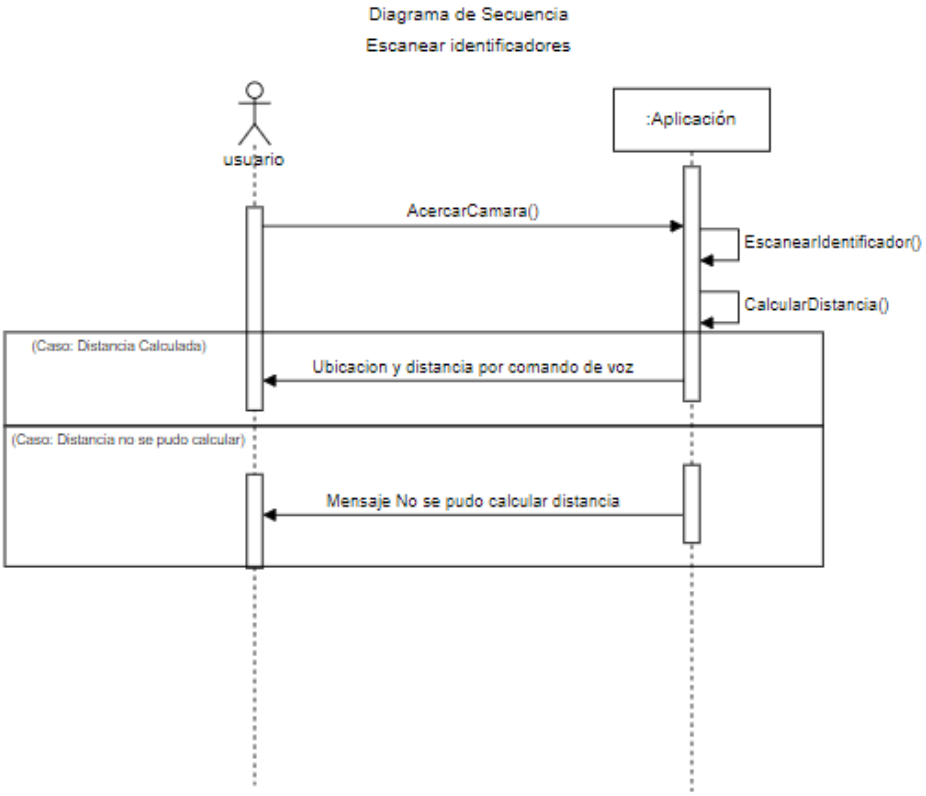

Ilustración 14. Diagrama de secuencia "Escanear identificadores".

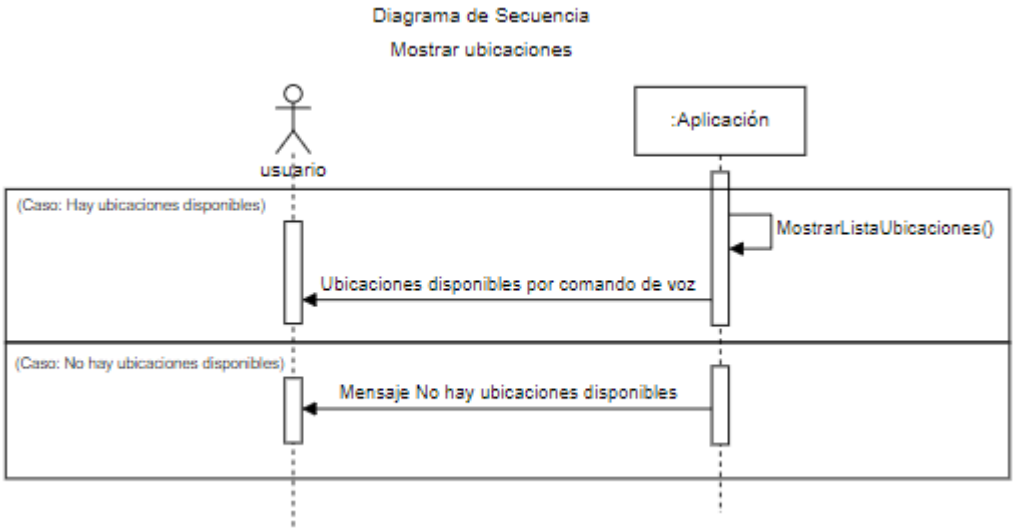

Ilustración 15. Diagrama de secuencia "Mostrar ubicaciones".

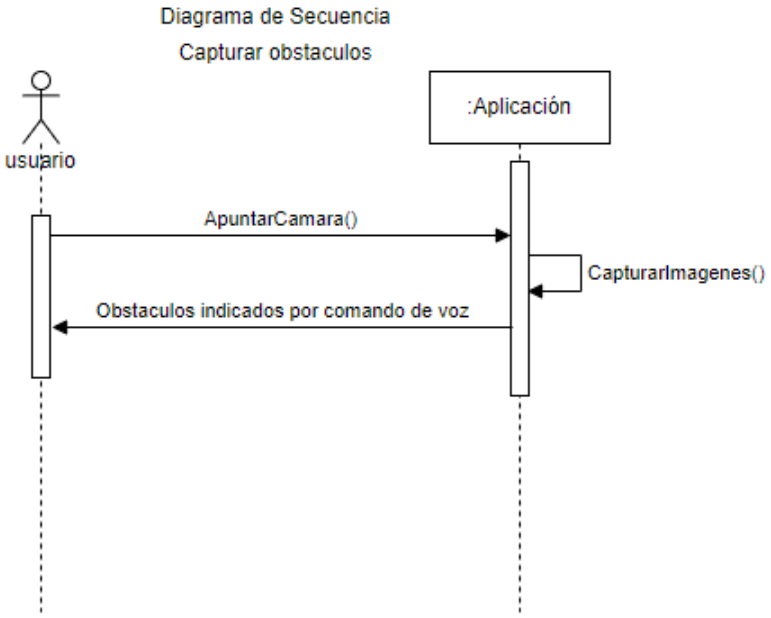

Ilustración 16. Diagrama de secuencia "Capturar obstáculos".

4.1.4. Modelo de clases

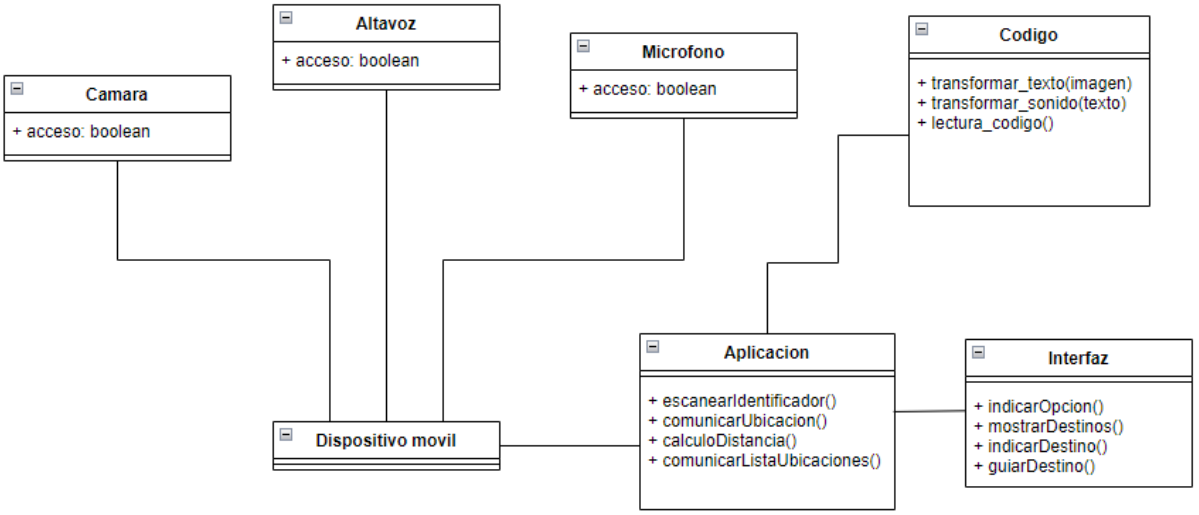

Ilustración 17. Diagrama de modelo de clases.

4.1.5. Descripción de arquitectura

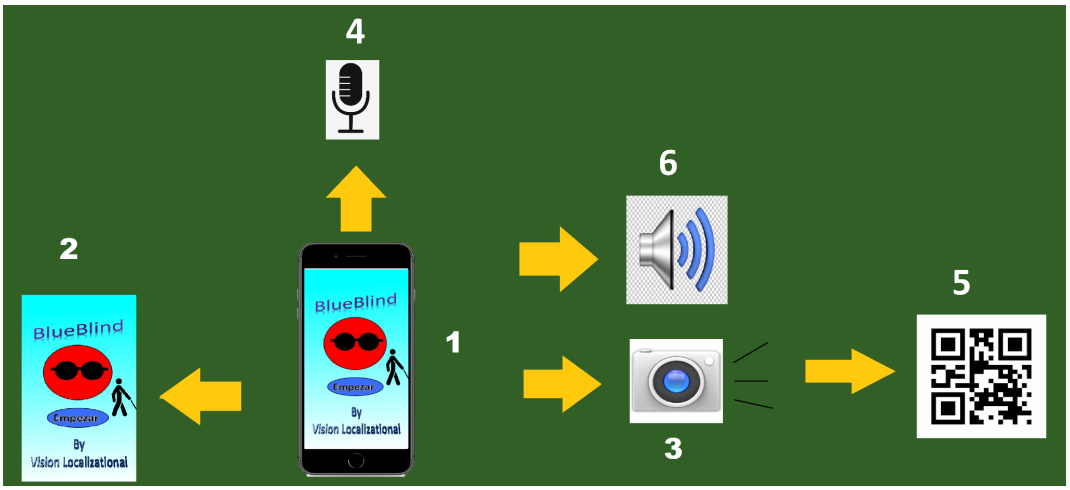

Ilustración 18. Diagrama de arquitectura de sistema.

1. El dispositivo móvil que contiene la aplicación instalada que dará indicaciones al usuario por medio de comando de voz.

2. La interfaz del software con la que interactúa el usuario a través de comandos de voz.

3. La cámara del dispositivo por la cual podrá analizar los códigos QR identificadores de las ubicaciones.

4. El micrófono o dispositivo de entrada de audio del dispositivo móvil. El usuario lo utiliza para indicar a la ubicación a la cual desea ir.

5. El código QR identificador de las ubicaciones es escaneado por la cámara de la aplicación, la cual utiliza comandos de voz para indicar al usuario la distancia a la que se encuentra.

6. El dispositivo de salida de audio del dispositivo móvil, por el cual la aplicación indicará al usuario la distancia de la ubicación de los sensores u obstáculos.

4.1.6. Diseño de la interfaz de usuario

Ilustración 19. Pantalla de carga de BlueBlind.

En esta ilustración se aprecia la interfaz de la pantalla de carga de la aplicación. Una vez pasados 2 segundos, se iniciará la aplicación.

Ilustración 20. Menú principal de BlueBlind.

En esta ilustración se muestra el menú principal de la aplicación. Mediante el pulsado del botón de micrófono, el usuario puede acceder a una de las tres opciones que se muestran en pantalla mediante comandos de voz que la aplicación registrará. En caso de que se ingrese un comando incorrecto, la aplicación mandará un mensaje de error.

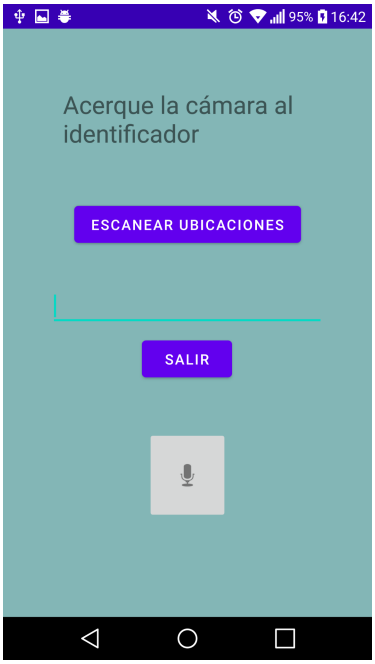

Ilustración 21. Ventana de escáner de BlueBlind.

Aquí se aprecia en la ventana del escáner de códigos QR. Al igual que en el menú principal, el usuario pulsa el botón de micrófono para enviar comandos, que la aplicación registrará, para acceder al modo escáner o para volver al menú principal. la lista de los sensores presentes en la ubicación actual. En caso de que se ingrese un comando incorrecto, la aplicación mandará un mensaje de error.

4.1.7. Especificación de requerimientos funcionales

Tabla 8: Especificación de requerimientos funcionales.

Requerimiento funcional	**Descripción**
La aplicación debe poder reconocer la voz del usuario.	La aplicación debe ser capaz de poder acceder al micrófono del dispositivo móvil para que el usuario pueda realizar indicaciones por medio de voz.
La aplicación debe de poder reproducir audios.	La aplicación debe de poder reproducir audios que indiquen los lugares detectados así como las indicaciones para llegar a cada una de ellas.
La aplicación debe de poder acceder a la cámara.	La aplicación debe de poder acceder a la cámara del dispositivo móvil para

	poder capturar los obstáculos del lugar, así como los identificadores de código QR.
La aplicación debe poder leer códigos QR.	La aplicación debe ser capaz de poder leer códigos QR que identifican las ubicaciones.

4.1.8. Especificación de requerimientos no funcionales

Tabla 9: Especificación de requerimientos no funcionales.

Requerimiento no funcional	**Descripción**
La aplicación solo puede ser instalada en Android.	La aplicación solo puede y debe ser instalada en dispositivos móviles con sistema operativo Android.
La aplicación debe ser hecha en Android Studio.	La aplicación debe ser desarrollada en el entorno de desarrollo Android Studio.
La aplicación solo funciona en interiores.	Dada la naturaleza de la aplicación, está solo puede usarse dentro de una ubicación, sea un edificio, un restaurante, una tienda, entre otros.
La aplicación está escrita en código Java.	La aplicación al ser creada en Android Studio, está escrita en lenguaje de programación Java, que es uno de los lenguajes soportados por el entorno de desarrollo.

4.2. Herramientas y técnicas

Herramientas: Android Studio (lenguaje de programación Java), Google Drive, Draw.io, Google Docs, Microsoft Office.

5. Planificación de procesos de soporte

5.1. Planificación de la documentación

- **Manual de usuario**: Es un documento que contiene las diferentes instrucciones de instalación y uso de la aplicación para que el usuario pueda utilizarla de la manera más sencilla y correcta posible.

- **Bitacoras**: Un conjunto de documentos que describen las principales tareas realizadas en una reunión de trabajo de equipo, así como las tareas planeadas para la siguiente reunión, sugerencias a desarrollar y los temas posibles a tratar.

- **Carta Gantt**: Es una línea de tiempo de la planificación y desarrollo de futuras tareas relacionadas con el avance del proyecto, cada una de las cuales sean futuras, pasadas o en progreso, está asociada a un plazo de tiempo aproximado determinado.

- **Wiki del proyecto**: Es un blog presente en la página de Redmine que detalla de forma general la realización del proyecto, sus aspectos que lo componen y su propósito.

- **Informe final de proyecto**: Es el presente documento que describe todo el apartado de planificación del proyecto, así como el análisis, diseño e implementación de la aplicación software desarrollada.

6. Implementación

6.1. Plan de integración

La aplicación software propuesta permite indicar mediante comandos de voz la distancia en donde se encuentra cierta ubicación demarcada por un código QR que la aplicación leerá, lo que permite que el usuario pueda desplazarse dentro de un entorno. Para realizar la integración consta de un número de partes, cada una de las cuales está compuesta por diversos modelos o APIs implementadas para la definición o funcionamiento correcto de cada parte de la aplicación:

- **La comunicación entre usuario y aplicación, y visceversa**:
  Dado que la aplicación está destinada a usuarios que padezcan de alguna discapacidad visual, es fundamental establecer una comunicación mediante comandos de voz sea desde el usuario a la aplicación, como en el caso contrario. Para ello se hace uso de las APIs "RecognizerIntent" o "Recognizer Speech" para que la aplicación recibe los comandos de voz del usuario y los transforme en cadenas de caracteres que serán utilizadas para realizar ciertas acciones, y "Text to Speech" para que la aplicación transforme cadenas de caracteres resultantes en comandos de voz que le dirá al usuario, ya sea para guiarlo al leer los identificadores, o para indicarle que se ha producido algún error.

- **Escaneo de identificadores de ubicaciones**:
  Debido que la idea original de la aplicación en la que se hacía uso de sensores beacons se ha descartado se debe de hacer uso de módulos de lectura de códigos QR que sirvan de identificadores para las ubicaciones por las que guiar al usuario. Junto con ello, se hace uso de APIs o módulos para calcular la distancia entre el código identificador y el usuario. De esta forma, se puede guiar al usuario hasta la ubicación identificada por medio de la distancia. Para poder leer códigos QR, se hace uso de la biblioteca de código abierto Zxing, utilizado también para códigos de barra. En cambio, para poder calcular la distancia entre usuario e identificador, se hace uso de…

6.2. Modelos de implementación

- **Lectura de códigos QR con Zxing**:
  La biblioteca Zxing provee de funcionalidades con las que se puedan escanear códigos QR. Dado que las ubicaciones son identificadas por estos códigos, se usa esta biblioteca para así poder guiar al usuario.

- **Transformación de texto a comandos de voz con TextToSpeech**:
  TextToSpeech es una API que transforma una cadena de caracteres a voz. Se utiliza principalmente para que la aplicación de el nombre y la distancia de la ubicación

identificada por un código QR escaneado. También se usa para mandar mensajes de error al usuario.

- **Cálculo de la distancia del código QR identificador con**:

- **Transformación de comandos de voz a texto con RecognizerIntent**:
  Reconizer Intent o Recognizer Speech es una API da la posibilidad al usuario de poder comunicarse con la aplicación a través de comandos de voz. La aplicación recibe un comando de voz y lo transforma a una cadena de caracteres con la que según sea su valor, se podrán realizar diferentes acciones. Es utilizada para que el usuario pueda acceder a las distintas funciones de la aplicación.

6.3. Módulos implementados

- **Lectura de código QR**:
  Para poder hacer uso de las funcionalidades de la biblioteca Zxing, primero debemos exportar la biblioteca en sí añadiendola en el archivo gradle module.

```
implementation 'com.journeyapps:zxing-android-embedded:4.1.0'
```

Ilustración 22. Implementación de Zxing en gradle module.

En la siguiente figura se puede apreciar que se implementa un nuevo objeto IntentIntegrator al cual se le dan algunas propiedades al lector en si.Entre esas se destaca que es lo que puede leer con setDesiredBarcodeFormats, una leyenda que aparezca en el lector con setPrompt, la cámara utilizada con setCameraId (0 es el valor de la cámara trasera), setBeepEnabled para que se dé una alerta al escanear el codigo, setBarcodeImageEnable para que pueda leer imágenes correctamente, y por último initiateScan para iniciar el escaneo.

```
if (strSpeech2Text.equalsIgnoreCase( anotherString: "escanear ubicaciones")){
    IntentIntegrator integrador = new IntentIntegrator( activity: Escanner.this);

    integrador.setDesiredBarcodeFormats(IntentIntegrator.ALL_CODE_TYPES);
    integrador.setPrompt("Lector- BlueBlind");
    integrador.setCameraId(0);
    integrador.setBeepEnabled(true);
    integrador.setBarcodeImageEnabled(true);
    integrador.initiateScan();
}
```

Ilustración 23. Implementación del lector IntentIntegrator.

Mediante este código, se utiliza el resultado obtenido por el IntentIntegrator (el lector QR) para guardar el contenido del código QR en una variable String s.

```
IntentResult result = IntentIntegrator.parseActivityResult(requestCode, resultCode, data);

if(result != null)
{
    if(result.getContents() == null){
        Toast.makeText( context: this, text: "Lectura no valida", Toast.LENGTH_LONG).show();

    }else {

        Toast.makeText( context: this,result.getContents(), Toast.LENGTH_LONG).show();
        txtResultado.setText(result.getContents());
        String s = txtResultado.getText().toString();
        int speech = textToSpeech.speak(s, TextToSpeech.QUEUE_FLUSH,  params: null);
    }
}
else{

    super.onActivityResult(requestCode, resultCode, data);
}
```

Ilustración 24. Implementación del guardado de contenidos del código QR.

- **Transformación de texto a comandos de voz**:
  Para poder hacer uso de TextToSpeech para la transformación de textos a comandos de voz, primero se debe importar la librería de TextToSpeech en la actividad o clase en donde se utilizará, como se muestra en la siguiente ilustración:

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

Ilustración 25. Implementación de la librería TextToSpeech.

Luego, se deben crear las variables textToSpeech de tipo TextToSpeech con la que se hará uso de las funciones de texto a comando de voz, y una variable Locale spanish con la que permite el uso del idioma español para TextToSpeech, pues no es uno de los idiomas que viene por defecto en la librería.

```
TextToSpeech textToSpeech;
Locale spanish = new Locale( language: "es",  country: "ES");
```

Ilustración 26. Creación de variables TextToSpeechtextToSpeech y Locale spanish.

En la implementación de la siguiente imagen, se establece el idioma a utilizar (en este caso el español gracias a la variable spanish) en textToSpeech, esto mediante setLanguage.

```
textToSpeech = new TextToSpeech(getApplicationContext()
        , new TextToSpeech.OnInitListener() {

    @Override
    public void onInit(int i){
        if (i == TextToSpeech.SUCCESS){
            int lang = textToSpeech.setLanguage(spanish);
        }
    }
});
```

Ilustración 27. Establecimiento del idioma español para textToSpeech.

Utilizando parte de la implementación mostrada en la ilustración 24, se hace uso del método textToSpeech.speak, con la variable s como uno de sus argumentos, para que la aplicación pueda transmitir por comando de voz el contenido del código QR.

```
IntentResult result = IntentIntegrator.parseActivityResult(requestCode, resultCode, data);

if(result != null)
{
    if(result.getContents() == null){
        Toast.makeText( context: this, text: "Lectura no valida", Toast.LENGTH_LONG).show();

    }else {

        Toast.makeText( context: this,result.getContents(), Toast.LENGTH_LONG).show();
        txtResultado.setText(result.getContents());
        String s = txtResultado.getText().toString();
        int speech = textToSpeech.speak(s, TextToSpeech.QUEUE_FLUSH,  params: null);
    }
}
else{

    super.onActivityResult(requestCode, resultCode, data);
}
```

Ilustración 28. Usando textToSpeech para que la aplicación mencione el contenido de s por comando de voz.

De forma similar, se utiliza textToSpeech para que la aplicación envíe mensajes de error por comando de voz en caso de que el usuario use comandos de voz erróneos. En dichos casos, como se ve en la siguiente imagen, la aplicación comunica el mensaje “El comando ingresado es incorrecto”.

```
else{
    String error = "El comando ingresado es incorrecto";
    textToSpeech.speak(error, TextToSpeech.QUEUE_FLUSH,  params: null);
}
```

Ilustración 29. Uso de textToSpeech para comunicar mensajes de error.

- **Cálculo de la distancia del código QR identificador**:

- **Transformación de comandos de voz a texto**:
  Para poder captar comandos de voz del usuario y transformarlos en strings que la aplicación utilizará para realizar ciertas acciones dependiendo de su valor, primero debemos importar la librería de RecognizerIntent como se muestra en la siguiente figura.

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

Ilustración 30. Implementación de la librería RecognizerIntent.

Luego, se crea una clase RECOGNIZE_SPEECH_ACTIVITY con valor 1 para poder hacer uso de los servicios de voz de Google.

```
private static final int RECOGNIZE_SPEECH_ACTIVITY = 1;
```

Ilustración 31. Definición de clase RECOGNIZE_SPEECH_ACTIVITY.

Se crea el método Hablar, el cual permite hacer uso del servicio de voz de Google. Además, gracias a RecognizerIntent.EXTRA_LANGUAGE_MODEL, se puede definir el idioma que detectara la aplicación; a su vez, se hace uso de un try-catch para verificar si el dispositivo móvil soporta o no el servicio de reconocimiento de voz.

```
public void Hablar(View v) {
    Intent intentActionRecognizeSpeech = new Intent(
            RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    // Configura el Lenguaje (Español-México)
    intentActionRecognizeSpeech.putExtra(
            RecognizerIntent.EXTRA_LANGUAGE_MODEL, value: "es-MX");
    try {
        startActivityForResult(intentActionRecognizeSpeech,
                RECOGNIZE_SPEECH_ACTIVITY);
    } catch (ActivityNotFoundException a) {
        Toast.makeText(getApplicationContext(),
                text: "Tú dispositivo no soporta el reconocimiento por voz",
                Toast.LENGTH_SHORT).show();
    }
}
```

Ilustración 32. Implementación del método Hablar.

En las siguientes dos imágenes se aprecia la implementación del método onActivityResult tanto en la clase de la ventana Escáner, como en la ventana principal. En onActivityResult, se realiza una transformación de los resultados obtenidos por el método Hablar, en strings. Se puede ver que se guardan los valores de speech mediante la función get(0), en un string strSpeech2Text que será comparado ciertos comandos específicos mediante la función equalsIgnoreCase, de

esta forma no habrá distinción entre mayúsculas o minúsculas. Si el valor de strSpeech2Text es similar a uno de los valores, la aplicación accede a una de las funciones o ventanas descritas dentro del if correspondiente. Por ejemplo, en la ilustración 33, si el usuario dice el comando “escanear ubicaciones”, se accedera al lector de códigos QR generado por IntentIntegrator.

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    switch (requestCode) {
        case RECOGNIZE_SPEECH_ACTIVITY:
            if (resultCode == RESULT_OK && null != data) {
                ArrayList<String> speech = data
                        .getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
                String strSpeech2Text = speech.get(0);
                grabar.setText(strSpeech2Text);

                if (strSpeech2Text.equalsIgnoreCase( anotherString: "escanear ubicaciones")){
                    IntentIntegrator integrador = new IntentIntegrator( activity: Escanner.this);

                    integrador.setDesiredBarcodeFormats(IntentIntegrator.ALL_CODE_TYPES);
                    integrador.setPrompt("Lector- BlueBlind");
                    integrador.setCameraId(0);
                    integrador.setBeepEnabled(true);
                    integrador.setBarcodeImageEnabled(true);
                    integrador.initiateScan();
                }
                else if(strSpeech2Text.equalsIgnoreCase( anotherString: "salir")){
                    Intent intent = new Intent( packageContext: Escanner.this,Home.class);
                    startActivity(intent);
                    finish();
                }
```

Ilustración 33. Método onActivityResult en la clase del Escáner.

Por otro lado, en la ilustración 34, si por ejemplo el usuario usa el comando salir, la aplicación se cerrará. Por último, si la aplicación capta un comando de voz erróneo o no aceptado por el usuario, ocurrirá lo mostrado en la ilustración 29.

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    switch (requestCode) {
        case RECOGNIZE_SPEECH_ACTIVITY:
            if (resultCode == RESULT_OK && null != data) {
                ArrayList<String> speech = data
                        .getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
                String strSpeech2Text = speech.get(0);
                grabar.setText(strSpeech2Text);

                if (strSpeech2Text.equalsIgnoreCase( anotherString: "escáner")){
                    Intent intent = new Intent( packageContext: Home.this,Escanner.class);
                    startActivity(intent);
                    finish();
                }
                else if(strSpeech2Text.equalsIgnoreCase( anotherString: "lista")){

                }
                else if(strSpeech2Text.equalsIgnoreCase( anotherString: "salir")){
                    finish();
                }
```

Ilustración 34. Método onActivityResult en la clase del menú principal.

6.3. Pruebas realizadas de la aplicación

- **Prueba nº 1**:

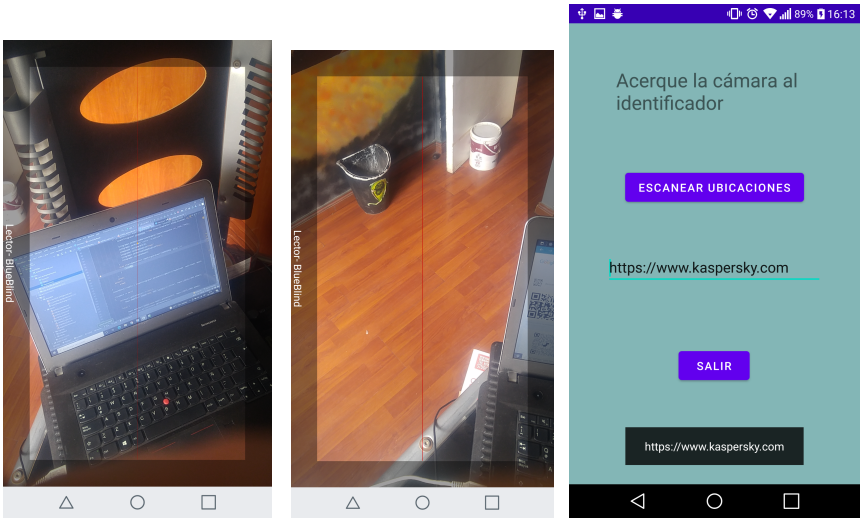

Ilustración 35, 36 y 37. Prueba 1. Escaner QR (35). Escaner apunto de leer un código QR (36). Interfaz antigua de Escáner recibiendo código QR como texto y narrándolo como comando de voz (37).

En la primera prueba se probó el uso del lector QR, así como del comando TextToSpeech. Al acceder al escáner, se acercó el lector a un código QR de prueba en internet. El marco de lectura del escáner es bastante grande así que leyó el

código apenas se acercó un poco. Al leerlo, se guardó el contenido del código en un String que se visualizó en un textView en la ventana de escaner. Al mismo tiempo, la aplicación utilizo TextToSpeech para leer en voz alta el contenido del QR en el textView.

- **Prueba nº 2**:

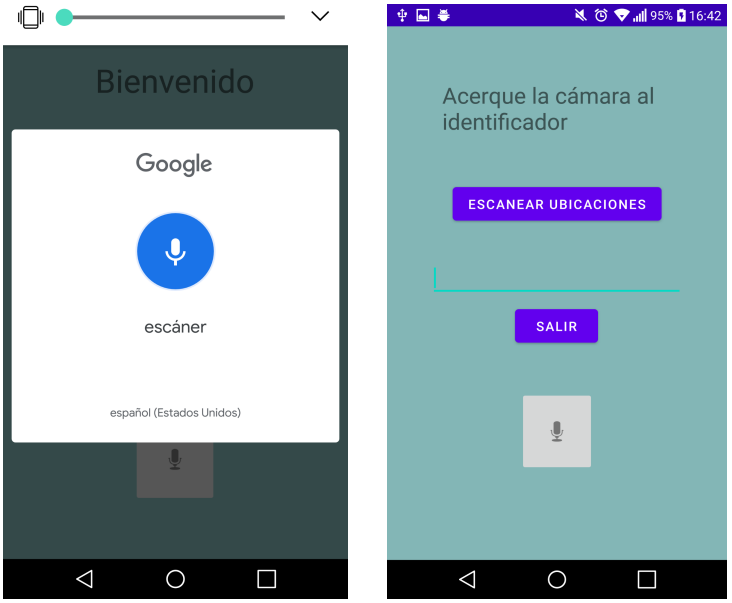

Ilustración 38 y 39. Prueba 2. La aplicación haciendo uso de un servicio de Google para captar los comandos de voz del usuario (38). La aplicación accediendo al menú de escáner desde el menú principal después de captar un comando de voz del usuario (39).

- Para la segunda prueba se probó el funcionamiento de RecognizerIntent a la hora de recibir comandos de voz de parte del usuario. Una vez iniciada la aplicación, el usuario pulsa en el botón de micrófono para acceder al modo de comandos de voz de Google. Si la aplicación capta algunas palabras clave, esta accedera a algunas de las funciones que presenta. Al captar el comando “escáner”, la aplicación accede a la ventana de Escaner. Captando el comando “salir”, se cierra la aplicación. Captando un comando erróneo, la aplicación utilizará TextToSpeech para mandar por comando de voz el siguiente mensaje de error: “El comando ingresado es incorrecto”.

7. Problemas encontrados

Entre los diversos problemas encontrados durante el desarrollo de la aplicación, se pueden mencionar:

1. **Fallas en la instalación de OpenCV**: Para poder obtener las distancias entre el dispositivo móvil del usuario y el código QR, se pensó en un principio en ocupar funciones proporcionadas por la biblioteca libre de visión computacional OpenCV para ello. Sin embargo, debido a problemas posiblemente relacionados con la versión utilizada de Android Studio, no se pudo instalar correctamente OpenCV.

2. **Fallos en el cálculo de la distancia**: Si bien se optó por el uso de funciones de SensorManager para poder utilizar funciones que permitan realizar el cálculo de la distancia, también se generó un problema referente al mismo que consiste en que se necesita de una forma de calcular el ángulo de la cámara para obtener la distancia total. Al no estar ese cálculo del ángulo, la aplicación se cierra de forma abrupta.

3. **Cierres inesperados de la aplicación al captar comandos de voz erróneos**: En algunas ocasiones, al probar la funcionalidad de captura de comandos de voz del usuario en vez de que la aplicación mande un mensaje de error por comando de voz, la aplicación se cierra inesperadamente.

8. Soluciones propuestas

En este apartado se presentan soluciones relacionadas a algunos de los problemas presentados en el apartado anterior.

1. **Utilizar otra API o método diferente para capturar la distancia**: Dados los problemas al instalar OpenCV, es recomendable buscar otra API o métodos que permitan hacer el cálculo de la distancia entre el usuario y el código QR identificador.

2. **Usar una versión actualizada de la aplicación**: El que la aplicación se cierre inesperadamente al capturar comandos de voz erróneos, puede deberse a que se posee una versión desactualizada o corrupta de la aplicación. Lo que se recomienda es descargar una versión actualizada y estable de la aplicación.

9. Conclusiones

Durante todo un periodo de trabajo que abarcó un aproximado de cuatro meses, se realizó la planificación y desarrollo de un proyecto de trabajo titulado “Invidentes conociendo su entorno”, que consiste en la fabricación de una aplicación software para dispositivos móviles que permite guiar a usuarios con discapacidades visuales a trasladarse dentro de un espacio interior, como podría ser un edificio o un restaurante, entre otros.

Una vez planificada se empezó el desarrollo de la aplicación, apodada como “BlueBlind”, pasando por la especificación de sus funcionalidades, su diseño y requerimientos, hasta llegar a su implementación y la realización y registro de pruebas de su funcionamiento.

En el transcurso del proyecto, se documentó el avance realizado del mismo en diversas formas, ya sean bitácoras que describen los realizado en las distintas reuniones de equipo de trabajo, con una Carta Gantt que sirva como una línea de tiempo de la organización de las tareas a realizar, o por medio del presente informe que detalle todo lo mencionado con anterioridad y que tenga relación con el proyecto en sí.

Llegado a esta etapa de finalización, se debe de destacar que a la hora de planificar el proyecto y realizar el apartado de análisis y diseño de la aplicación, se pudo realizar sin mayores inconvenientes, llegando a tener un proyecto consistente. Pero una vez llegado el proceso de implementación se vio que la idea de solución original era algo ambiciosa. Esto debido a que se presentaron dificultades en cuanto a la búsqueda de bibliotecas y módulos que se pudieran implementar en lenguaje de programación Java, utilizado en el entorno de desarrollo utilizado, y que a su vez sirvieran para el funcionamiento de la aplicación. Por ende, se tuvieron que realizar cambios en la planificación, análisis y diseño para que ahora no se utilizarán sensores beacons, Bluetooth ni bases de datos, sino que ahora la aplicación se enfocaría en el escaneo y cálculo de distancia de códigos QR identificadores de las ubicaciones.

Esto en general provocó algunas complicaciones en el desarrollo de la aplicación en sí, como lo es un retraso en el mismo, así como que no se pudieron implementar algunas de las funcionalidades planificadas. Pero a pesar de todo, se llegó a una versión estable y funcional que presenta la funcionalidad principal descrita en la idea actualizada de la aplicación.

En un futuro cercano, se busca poder seguir trabajando en la aplicación, buscando implementar las funcionalidades que no llegaron a la versión actual, así como también posiblemente agregar algunas otras extras. Junto con ello, la empresa cuenta con el objetivo de poder desarrollar otras aplicaciones con las que se pueda hacer uso de aquellas bibliotecas, módulos y elementos que fueron descartados durante el cambio en la planificación del proyecto actual, así como cualquier otra nueva tecnología que pueda resultar interesante o importante de implementar.

10. Referencias

[1] D. Aracena Pizarro y J. Cordova Guarachi. “Proyecto II Piloto 2do Semestre 2021”. Departamento de Ingeniería Civil en Computacion e Informatica, Universidad de Tarapacá, sede Arica, Chile, 2021.

[2] Wikipedia. [Online]. “Discord”. Disponible en: https://es.wikipedia.org/wiki/Discord

[3] Wikipedia. [Online]. “Redmine”. Disponible en: https://es.wikipedia.org/wiki/Redmine

[4] Wikipedia. [Online]. “Google Drive”. Disponible en: https://es.wikipedia.org/wiki/Google_Drive

[5] Wikipedia. [Online]. “WhatsApp”. Disponible en: https://es.wikipedia.org/wiki/WhatsApp

[6] Androfast. [Online]. “Cómo hacer una aplicación reconocimiento de voz en android studio”. Disponible en: https://www.androfast.com/2016/07/como-hacer-una-aplicacion.html

[7] Youtube. [Online]. Códigos de Programación - MR. “Lector de códigos de barras y QR - Android Studio”. Disponible en: https://www.youtube.com/watch?v=elZP3JFFdVk&ab_channel=C%C3%B3digosdeProgramaci%C3%B3n-MR

[8] Android Developers. [Online]. “Download Android Studio and SDK tools” Download Android Studio and SDK tools | Android Developers

[9] Youtube. [Online]. Tecno Flash. “Reconocimiento de voz en android studio”. Disponible en: https://www.youtube.com/watch?v=G362fF5G-NY&ab_channel=TecnoFlash

[10] Youtube. [Online]. Android Coding. “How to Convert Text to Speech in Android Studio | TextToSpeech | Android Coding”. Disponible en: https://www.youtube.com/watch?v=QFEiolJcldc&ab_channel=AndroidCoding

[11] Youtube. [Online]. Anconde. “Configure openCV in android studio arctic fox version 2020.3.1 |”. Disponible en: https://www.youtube.com/watch?v=psoeNfFAKL8&ab_channel=Ancode

[12] BiblioGuías [Online]. “Estilo IEEE - Citas y elaboración de bibliografía: el plagio y el uso ético de la información: Estilo IEEE”. Disponible en: https://biblioguias.uam.es/citar/estilo_ieee

[13] Altronics. [Online]. “i-Beacon BCN01 BLE”. Disponible en: https://altronics.cl/bcn01-ble-ibeacon

[14] PyImageSearch. [Online] “Find distance from camera to object/marker using Python and OpenCV”. Disponible en: https://www.pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/

[15] Teltonika Telematics. [Online]. “SOLUCIONES DE RASTREO EN ESPACIOS INTERIORES”. Disponible en: https://teltonika-gps.com/es/industries/use-cases/indoor-tracking-solutions/

[16] Philips iluminación. [Online]. “Posicionamiento de interiores – Las 4 tecnologías punteras”. Disponible en: https://www.lighting.philips.es/soporte/contacto/tendencias-en-iluminacion/para-los-profesionales/indoor-positioning-posicionamiento-interiores

[17] Google. [Online]. “Mapas de interiores – Información – Google Maps”. Disponible en: https://www.google.com/maps/about/partners/indoormaps/

[18] Xataka. [Online]. “Esta alternativa al GPS funciona en interiores y es capaz de detectar la altura para saber en qué planta nos situamos”. Disponible en: https://www.xataka.com/servicios/esta-alternativa-al-gps-funciona-interiores-capaz-detectar-altura-para-saber-que-planta-nos-situamos

[19] Situm. [Online]. “Situm Posicionamiento en Interiores para guiado y localización”. Disponible en: https://situm.com/es/

[20] D. Aracena Pizarro y J. Cordova Guarachi. “Formato informe 2021”. Departamento de Ingeniería Civil en Computacion e Informatica, Universidad de Tarapacá, sede Arica, Chile, 2021.

[20] D. Aracena Pizarro y J. Cordova Guarachi. “Ejemplos de Escenario de aplicación”. Departamento de Ingeniería Civil en Computacion e Informatica, Universidad de Tarapacá, sede Arica, Chile, 2021.