UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

Departamento de Ingeniería en Computación e Informática

INFORME DE PROYECTO

Detección de paneles para reconocimiento de recorrido de transporte público para usuarios no vidente.

Autor(es): Cristian Fritis

Angelina Orozco

Benjamín Poblete

Asignatura: Proyecto II

Profesor(es): Diego Aracena Pizarro

ARICA, 07 de enero 2021

Fecha	Versión	Descripción	Autor(es)
13/10/2020	1.0	Versión preliminar del formato	Cristian Fritis Angelina Orozco Benjamín Poblete
30/10/2020	1.1	Se corrige y se completa lo que estaba en la versión 1.0	Cristian Fritis Angelina Orozco Benjamín Poblete
31/10/2020	1.2	Se agrega la planificación de riesgos	Cristian Fritis Angelina Orozco Benjamín Poblete
03/11/2020	1.3	Se agrega la introducción, escenarios problemas y solución Se agrega el punto 3.2	Cristian Fritis Angelina Orozco Benjamín Poblete
19/12/2020	1.4	Se agrega la especificación de requerimientos Se agrega los requerimientos no funcionales Se agrega la descripción de arquitectura Se agrega los casos de uso	Cristian Fritis Angelina Orozco Benjamín Poblete
20/12/2020	1.5	Se agregan las descripciones de los casos de uso Se agregan los diagramas de secuencia	Cristian Fritis Angelina Orozco Benjamín Poblete
22/12/2020	1.6	Se agrega el modelo de clases Se actualiza la carta Gantt	Cristian Fritis Angelina Orozco Benjamín Poblete
05/01/2021	1.7	Se agregar el Plan de integración Se agregar el Modelo de implementación Se agregar las Pruebas de la aplicación	Cristian Fritis Angelina Orozco Benjamín Poblete

06/01/2021	1.8	Se actualiza la carta Gantt Se agrega las conclusiones	Cristian Fritis Angelina Orozco Benjamín Poblete

Índice

1. Panorama general

1.1 Resumen del Proyecto

- **Introducción:**

En el mundo existen aproximadamente 285 millones de personas con discapacidad visual. Según un estudio del 2004, en Chile existen 634906 personas con problemas visuales, por lo que, a lo largo del tiempo, la gente ciega, impedidos visuales o analfabetas ha tenido problemas para utilizar el transporte público con facilidad, esto es porque no pueden visualizar correctamente los tableros de las micros o los colectivos que hay en la ciudad y, por consiguiente, no pueden distinguir que transporte utilizar, es por esto que necesitan de asistencia de una persona para poder utilizar estos medios o les costará demasiado movilizarse (en la Figura 1 se muestra el problema encontrado).

Entonces se ha considerado crear una aplicación que permita guiar a la gente con problemas a la vista o problemas para distinguir los transportes, de modo que los ayude a llegar a los paraderos establecidos en la ciudad utilizando el audio del celular, de igual manera, deberá poder reconocer los tableros de las micros que se acerquen, esto se logrará utilizando la cámara del celular para escanear estos objetos, a su vez, se utilizará el micrófono del celular para emitir una alerta cuando se acerque el transporte solicitado por el usuario (en la Figura 2 se muestra la solución propuesta para el problema).

- **Escenarios problema y solución:**

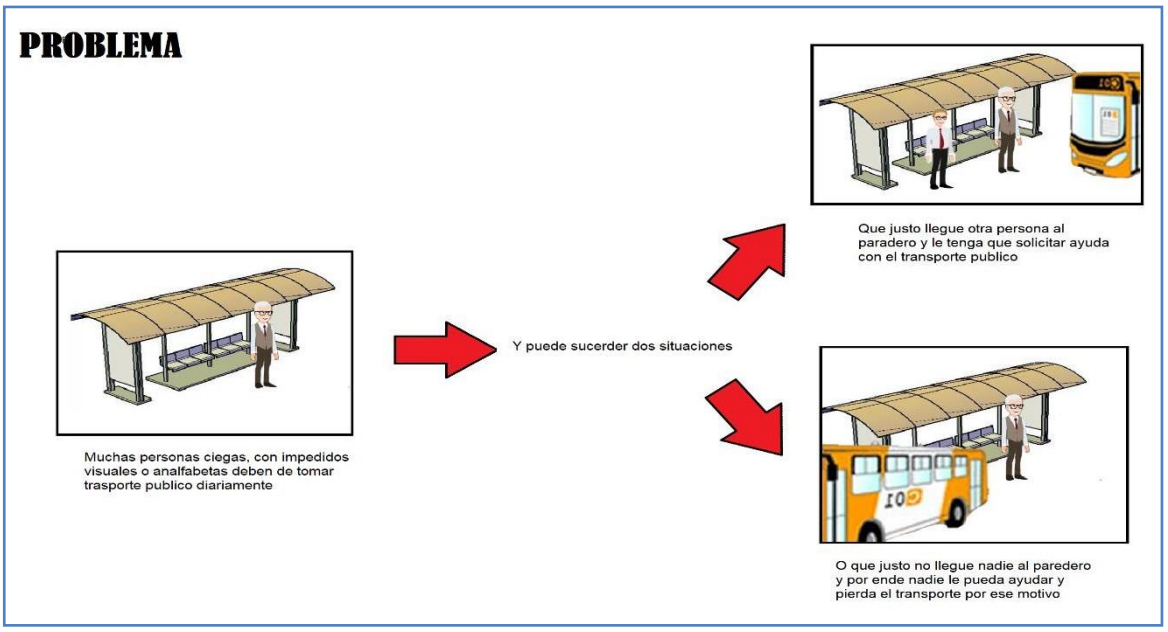


Figura 1: Escenario Problema

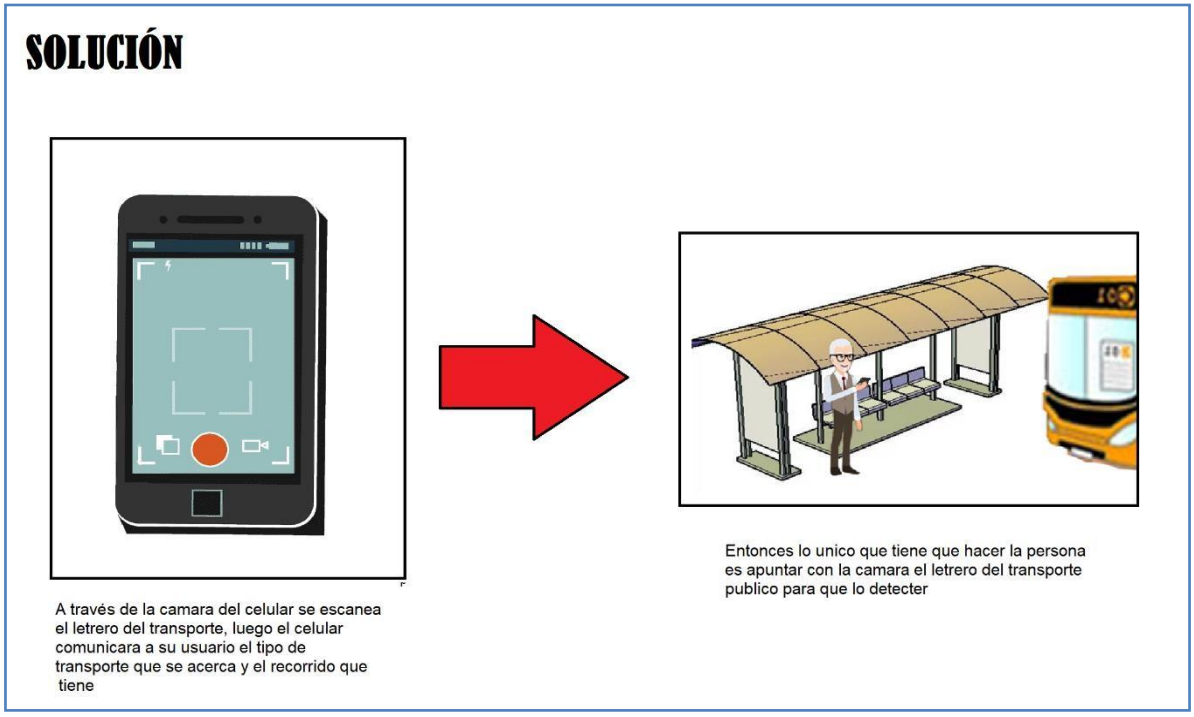


Figura 2: Escenario solución

- **Propósito:**

El propósito del proyecto consiste en crear una aplicación que permita guiar a la gente con problemas a la vista o problemas para visualizar los transportes, de modo que ayude a una persona en un paradero establecido en la ciudad mediante su celular, de modo que, deberá poder reconocer los tableros de las micros que se acerquen, esto se logrará utilizando la cámara del celular para escanear estos objetos, a su vez, se utilizará el micrófono del celular para emitir una alerta cuando se acerque el transporte solicitado por el usuario.

- **Alcance:**

Se ha considerado crear una aplicación que permita guiar a la gente con problemas a la vista o problemas para distinguir los transportes. El software contará con módulos para ubicar la posición de su usuario y guiarlo a su destino, además, de poder escanear texto y traducirlo a sonido. Así mismo, será desarrollado usando el lenguaje Python

- **Objetivo General:**

Mejorar la autonomía de las personas con discapacidad visual o problemas para distinguir de manera independiente el transporte público que necesitan en un paradero.

- **Objetivos específicos:**

  - Recopilar información sobre aplicaciones para gente ciega.
  - Desarrollar una interfaz para la aplicación en Smartphone.
  - Desarrollar la aplicación de la solución seleccionada.
  - Realizar pruebas de funcionamiento.
  - Integrar y entregar el producto final

- **Suposiciones y restricciones:**

  a) Suposiciones:

  - Se asume que las personas con problemas visuales o de analfabetismo que utilicen la aplicación tienen un celular con características apropiadas, es decir, con cámara frontal y sensores como el GPS.

  - Se asume que las personas solo tienen problemas de carácter visuales o analfabetismo.

  b) Restricciones:

  - La persona debe estar en un paradero
  - El Smartphone debe apuntar directamente al tablero del transporte publico
  - La aplicación funcionará en un Smartphone que tenga una cámara
  - La aplicación debe tener un traductor de texto para que se produzca el sonido
  - El proyecto debe ser realizado en el plazo impuesto por el profesor.
  - La aplicación debe poder ser usada por una persona con problema visual o analfabeta.

- **Entregables del proyecto:**

1. Bitácoras semanales
2. Informe de avance.
3. Presentación de avance.
4. Informe final.
5. Presentación final.
6. Manual de usuario.
7. Wiki del proyecto
8. Producto final

2. Organización del Proyecto

2.1 Personal y entidades

Jefe de proyecto, Diseñador, Programador, Redactor de informes

2.2 Roles y responsabilidades

Jefe de proyecto: Es la persona que coordina, organiza y representa al equipo de trabajo. El responsable es: Benjamín Poblete

Diseñador: Personal encargado de diseñar la interacción apk-usuario. El responsable es: Angelina Orozco

Programador: Personal encargado de realizar la programación. El responsable es: Cristian Fritis

Redactor de informes: Se encargan de la documentación del proyecto, de esta forma, realizan las bitácoras del proyecto, formulan los informes requeridos y se encargan de entregarlos en los plazos establecidos. El responsable es: Benjamín Poblete

2.3 Mecanismo de comunicaciones

Para poder tener una buena comunicación, se creó un grupo de Discord, el cual nos permitirá estar en contacto cada vez que haya una idea o para resolver los problemas que se nos presenten.

Para mayor comodidad al momento de realizar informes o presentaciones, se hará uso de la plataforma google drive, para compartir archivos y modificarlos de manera simultánea.

3. Planificación de los procesos de gestión

3.1 Planificación inicial del proyecto

- **Planificación de estimaciones:**

Tiempo estimado para el proyecto: 3 meses

Recurso	Valor	Cantidad
Notebooks	$800.000	3
Smartphone	$400.000	3
Software de desarrollo Python	De libre acceso	3
Microsoft Office	$40.000	3
Sueldo total de cada integrante	$1.000.000	3

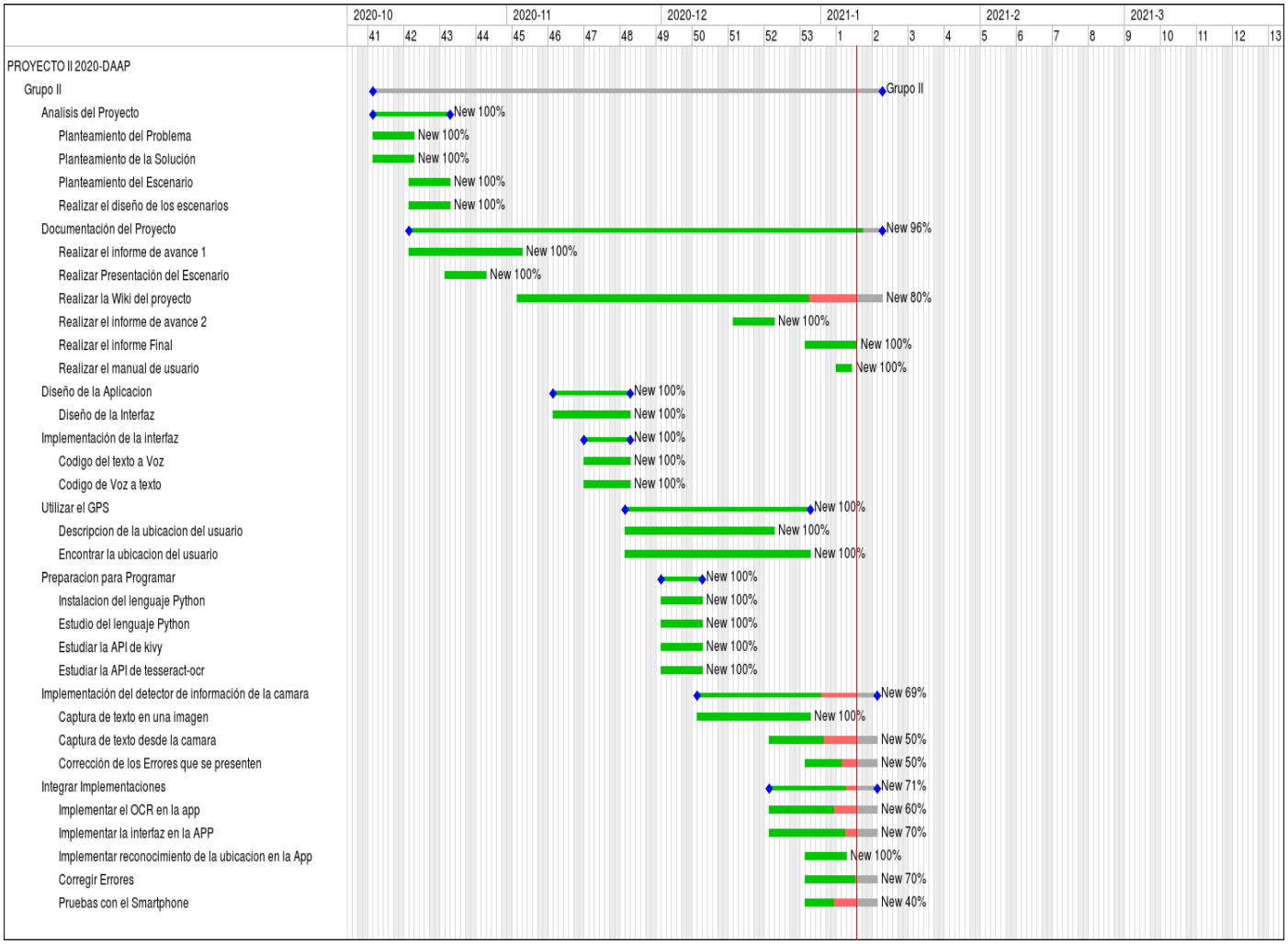
Costo total del proyecto	$6.720.000

- **Planificación de Recursos humanos:**

Diseñador:3
Programador: 3
Documentador: 3
Jefe de Proyecto:1

3.2 Lista de actividades

- **Actividades de trabajos**



- Planteamiento del problema
  Descripción: Se analizan los diferentes problemas que tienen las personas no videntes o analfabetas y se decide cual vamos a solucionar
  Responsable: Benjamín Poblete
  Producto: Problema definido

- Planteamiento de la solución
  Descripción: Se analizan las diferentes soluciones que podemos implementar para resolver el problema elegido y se decide la solución que vamos a realizar
  Responsable: Cristian Fritis
  Producto: Solución definida

- Planteamiento del escenario
  Descripción: Se plantea los posibles escenarios que puede tener el problema y la solución que escogimos
  Responsable: Angelina Orozco
  Producto: Escenarios del problema y solución definidos

- Realizar el diseño de los escenarios
  Descripción: Se diseña los escenarios elegidos para el problema y la solución de este
  Responsable: Angelina Orozco
  Producto: Diseños de los escenarios

- Realizar el informe de avance 1
  Descripción: Se realiza el informe de avance del proyecto que incluye la formulación de cómo se llevará a cabo el proyecto.
  Responsable: Angelina Orozco
  Producto: Informe de avance 1

- Realizar presentación del escenario
  Descripción: Se realiza la presentación de los escenarios en la que se expondrá los diseños de los escenarios elegidos con una breve argumentación
  Responsable: Benjamín Poblete
  Producto: Presentación del escenario

- Realizar la Wiki del Proyecto:
  Descripción: Un repertorio de información, donde todos los integrantes pueden agregar información esencial sobre el funcionamiento de la aplicación e imágenes de los avances que llevamos.
  Responsable: Benjamín Poblete.
  Producto: Wiki del Proyecto.

- Realizar el informe de avance 2
  Descripción: Se realiza el informe de avance 2 del proyecto 2
  Responsable: Benjamín Poblete
  Producto: Informe de avance 2

- Realizar el informe final
  Descripción: Se realiza el informe final del proyecto 2
  Responsable: Angelina Orozco
  Producto: Informe Final

- Realizar el manual de usuario
  Descripción: Se realiza el manual de usuario para proyecto 2
  Responsable: Benjamín Poblete
  Producto: Manual usuario

- Diseño de la interfaz
  Descripción: Se diseña la interfaz tratando que sea la más accesible para el usuario
  Responsable: Angelina Orozco
  Producto: Diseño de la interfaz

- Código del texto a voz
  Descripción: Se realiza el código para poder pasar de un texto a voz, para que así la aplicación de pueda comunicar con el usuario
  Responsable: Angelina Orozco
  Producto: Código del texto a voz terminado

- Código de voz a texto
  Descripción: Se realiza el código para poder pasar de voz a texto, para que así el usuario le pueda comunicar con la aplicación y que esta haga lo pedido
  Responsable: Angelina Orozco
  Producto: Código de voz a texto terminado

- Descripción de la ubicación del usuario
  Descripción: Se implementa el algoritmo para describir la ubicación del usuario a partir de la latitud y longitud.
  Responsable: Benjamín Poblete
  Producto: Descripción de la ubicación del usuario

- Encontrar la ubicación del usuario
  Descripción: Se implementa la detección de la latitud y longitud de la ubicación del usuario utilizando el GPS del celular
  Responsable: Benjamín Poblete
  Producto: Ubicación del usuario

- Instalación del Lenguaje Python:
  Descripción: Se va a dedicar un tiempo para la instalación del lenguaje de programación Python, ayudando también a los integrantes que no puedan realizar la instalación de esta.
  Responsable: Cristian Fritis
  Producto: En ambiente de programación Python.

- Estudio del Lenguaje Python:
  Descripción: Se va a hacer un repaso de la API de Python para empezar a programar la aplicación.
  Responsable: Cristian Fritis
  Producto: Entendimiento básico de las funciones a la hora de programar.

- Estudiar la Api de Kivy:
  Descripción: se va a realizar una investigación de la Api de kivy donde se desarrollará la aplicación para el Móvil.
  Responsable: Benjamín Poblete
  Producto: Preparación a la hora de desarrollar la parte móvil de la aplicación.

- Estudia la Api de Tesseract-OCR:
  Descripción: se va a hacer una investigación de la Api de Tesseract donde se hará la codificación para la captura de texto por medio de una imagen/cámara, buscando las funciones importantes que serán necesarias para este proyecto.
  Responsable: Cristián Fritis
  Producto: Preparación para realizar el reconocimiento de caracteres a la hora de programar.

- Captura de texto en una imagen:
  Descripción: se implementará la codificación de la captura de texto por medio de una imagen estática sacada de internet o por cámara.
  Responsable: Cristián Fritis
  Producto: Capacidad de que capture el texto que esté presente en una imagen.

- Captura de texto desde la cámara
  Descripción: Se crea e implementa el algoritmo de detección de texto por imagen.
  Responsable: Cristian Fritis
  Producto: Algoritmo de detección de texto en imagen

- Corrección de los Errores que se presenten
  Descripción: Se corrigen los errores que se generen en el algoritmo de detección de texto por imagen.
  Responsable: Cristian Fritis
  Producto: Algoritmo de detección de texto en imagen mejorado

- Implementar el OCR en la app
  Descripción: Se crea e implementa el algoritmo de traducción de texto en imagen.
  Responsable: Cristian Fritis
  Producto: Algoritmo de traducción de texto en imagen.

- Implementar la interfaz en la App
  Descripción: Se crea e implementa el algoritmo de la interfaz
  Responsable: Cristian Fritis
  Producto: Algoritmo de la interfaz

- Implementar el reconocimiento de la ubicación en la App
  Descripción: Se crea e implementa un algoritmo que permita mostrar la ubicación actual de usuario
  Responsable: Cristian Fritis
  Producto: Algoritmo de geolocalización

- Corregir Errores
  Descripción: Se corrigen los errores que se generen en el algoritmo de geolocalización y mapa.
  Responsable: Benjamín Poblete
  Producto: Algoritmo de geolocalización y mapa mejorado.

- Pruebas con el Smartphone
  Descripción: Se realizan pruebas con la aplicación terminada para detectar errores y corregirlos.
  Responsable: Angelina Orozco
  Producto: Aplicación final.

- **Asignación de tiempo**

Planificación del proyecto: 2-3 semanas.
Ejecución del proyecto: 6 a 7 semanas.
Cierre de proyecto: 1 semana.

3.3 Planificación de la gestión de riesgos

RIESGOS	PROBABILIDAD DE OCURRENCIA	NIVEL DE IMPACTO	ACCIÓN REMEDIAL
Un integrante del grupo se encuentre indisponible.	40%	1	Suplir su falta con otro integrante del grupo.
Que algún sensor que vamos a ocupar de los Smartphone se dañen	35%	2	Ocupar otro Smartphone que esté disponible.
Que el Smartphone deje de funcionar o se dañe.	35%	2	Ocupar otro Smartphone que esté disponible.
Los programas se pierden debido a un error en el dispositivo de almacenamiento.	50%	1	Crear respaldos en distintos pc de los integrantes del grupo.

Niveles de impacto:

- 1: Catastrófico
- 2: Crítico
- 3: Marginal
- 4: Despreciable

4. Análisis de la Arquitectura

4.1 Especificación de requerimientos

Requerimiento Funcional	**Descripción**
1.La aplicación debe comunicarse con su usuario emitiendo sonido.	Debido a que la aplicación que se está diseñando va dirigida para usuarios con problemas a la vista, esta debe dar a entender sus solicitudes al usuario mediante mensajes de audio.
2. La aplicación debe de utilizar la cámara del celular para captar imágenes.	La aplicación debe utilizar la cámara del celular para escanear los tableros de las micros y/o colectivos.
3. La aplicación debe poder comunicar la ubicación del usuario.	La aplicación debe utilizar el GPS del celular y emitir mediante un mensaje de audio al usuario su ubicación.
4. La aplicación debe entender lo que el usuario le comunica.	La aplicación debe entender los comandos de voz que el usuario comunica.

4.2 Lista de requerimientos no funcionales

Requerimiento no Funcional	**Descripción**
1. Límite del uso de la aplicación	La aplicación solo podrá usarse con conexión a internet debido a la necesidad de ingresar a un servicio web para obtener la dirección del usuario.
2. Límite de zona	La aplicación solo podrá usarse en la ciudad de Arica, Chile debido a que el reconocimiento de imagen esta optimizado para los tableros de los transportes de la ciudad.

4.3 Descripción de la arquitectura

Figura 3: Arquitectura

En la figura 3 se puede observar la arquitectura del proyecto con sus respetivos pasos ordenados numéricamente, los cuales consisten en:

1. Interfaz de aplicación con la que el usuario debe interactuar, esta mostrará en la pantalla principal un botón que el usuario tiene que apretar para que la aplicación emita los audios correspondientes.

2. Smartphone que utiliza una aplicación con la cual el cliente, o usuario, puede conocer su ubicación o puede escanear los tableros de las micros o colectivos para diferenciar a cada uno de estos.

3. El usuario interactúa con la aplicación mediante comandos de voz, es así como esta puede determinar cómo actuar.

4. Utiliza la cámara para analizar los tableros de este tipo y los traduce a texto, para luego ser emitidos como audio.

5. Utiliza el sensor GPS para analizar la posición del usuario y así dar a conocer la ubicación de este.

4.4 Diseño de la interface de Usuario

Al momento de iniciar la aplicación aparecerá un mensaje en la pantalla, a la vez que el celular comunica que se debe presionar la pantalla (Figura 4).

Figura 4: Pantalla de inicio

Una vez se presione la pantalla, el celular dará dos opciones para que el usuario elija (Figura 5), estas opciones deben ser seleccionadas comunicándole al celular a través de la voz del usuario (Figura 6 e Figura 7).

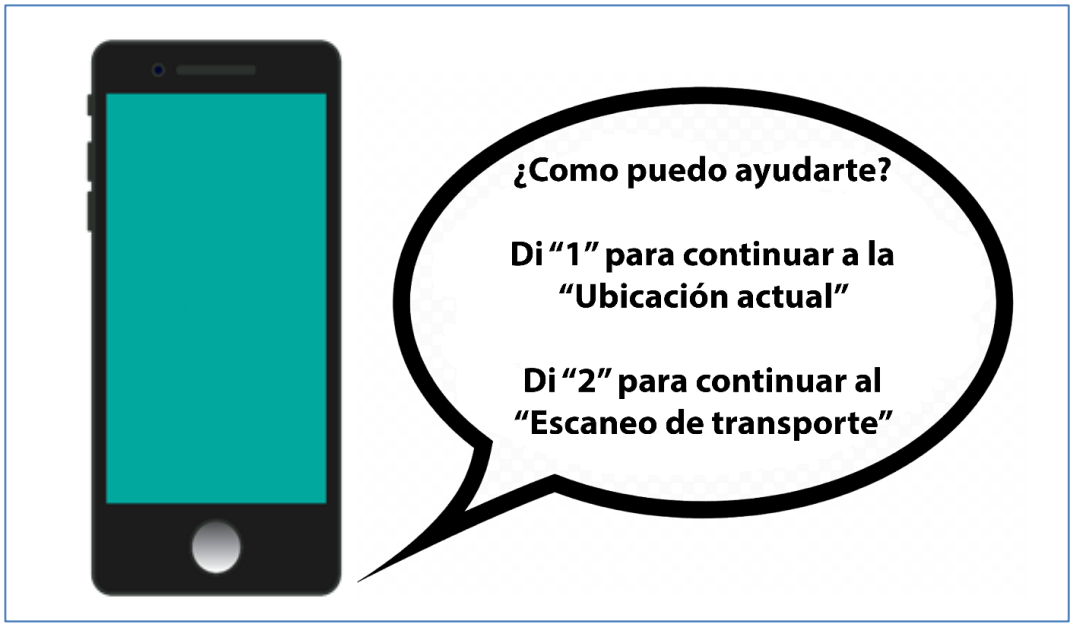


Figura 5: Opciones que da la App

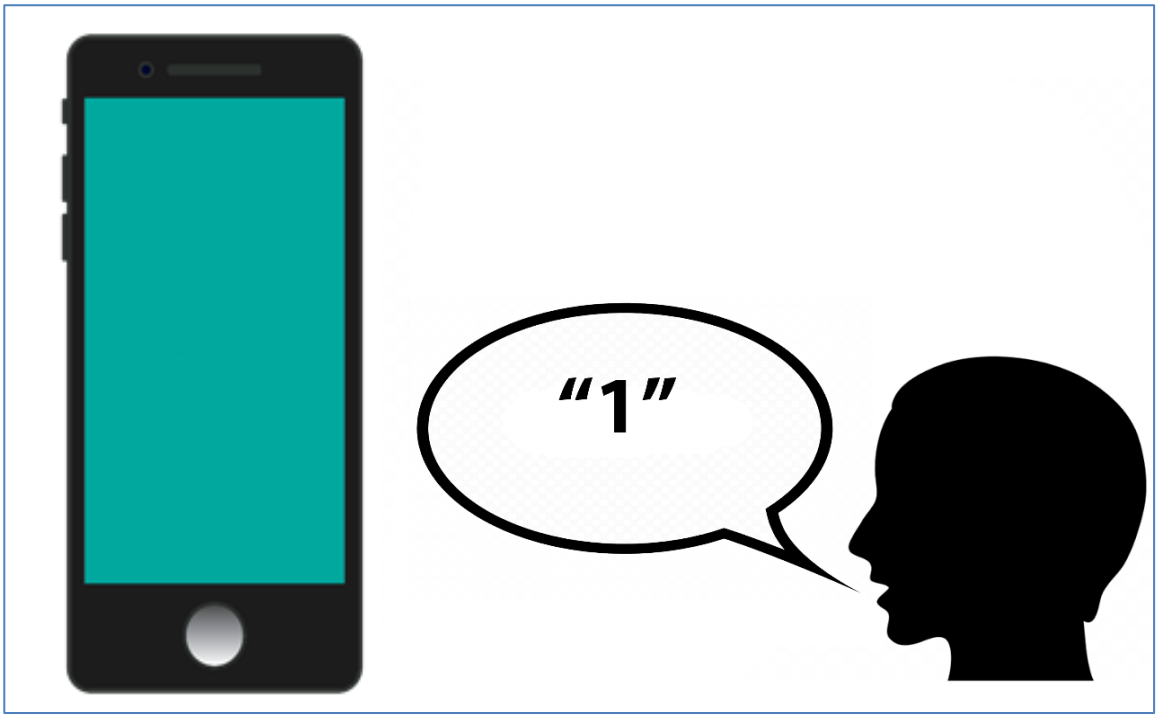


Figura 6: El usuario dice 1 para elegir la opción 1 de la App

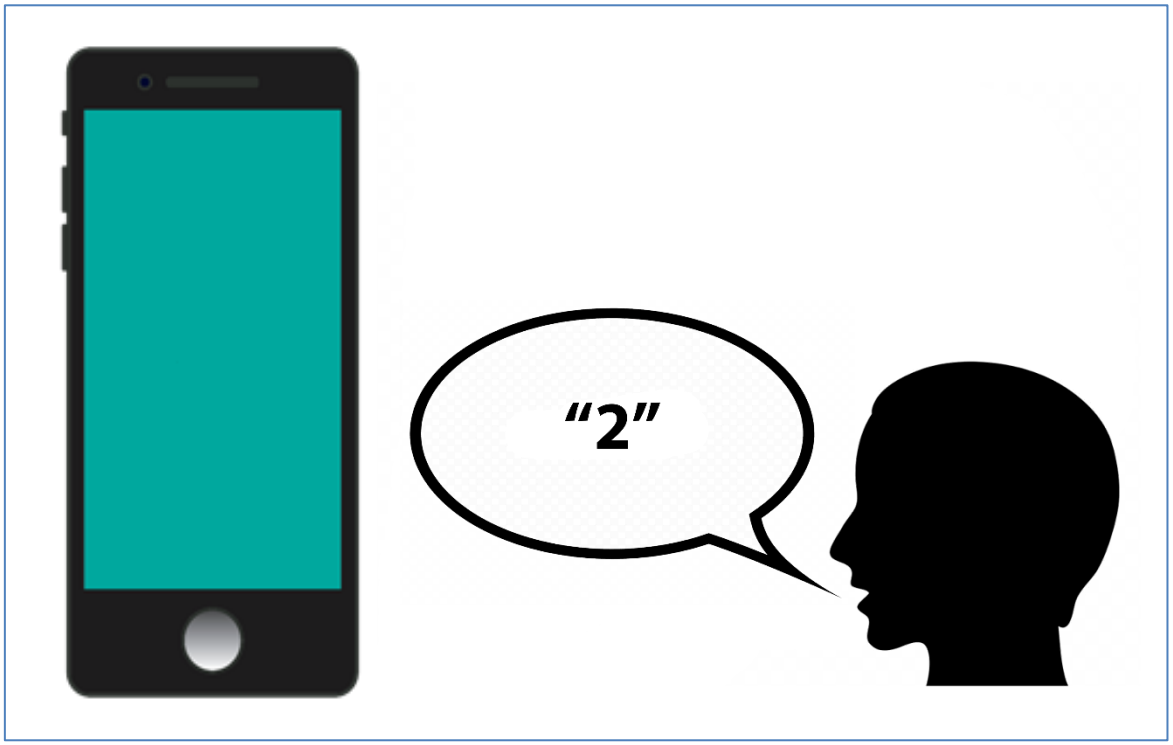


Figura 7: El usuario dice 2 para elegir la opción 2 de la App

En caso de que se elija la opción 1, el celular le dirá al usuario su ubicación actual (Figura 8).

Figura 8: La App comunica la ubicación actual del usuario

Por otro lado, si se elige la opción 2, el celular preguntará que micro o colectivo se busca (Figura 9). y, luego, empezará a utilizar la cámara y procederá a escanear los letreros de las micros o colectivos (Figura 10).

Figura 9: La App solicita el transporte que necesita el usuario

Figura 10: La App utiliza la cámara para poder escanear los letreros de las micros.

4.5 Modelo caso de uso

En la figura 11 se puede observar el caso de uso del proyecto, en la cual se puede constatar como el usuario puede interactuar con la aplicación y viceversa.

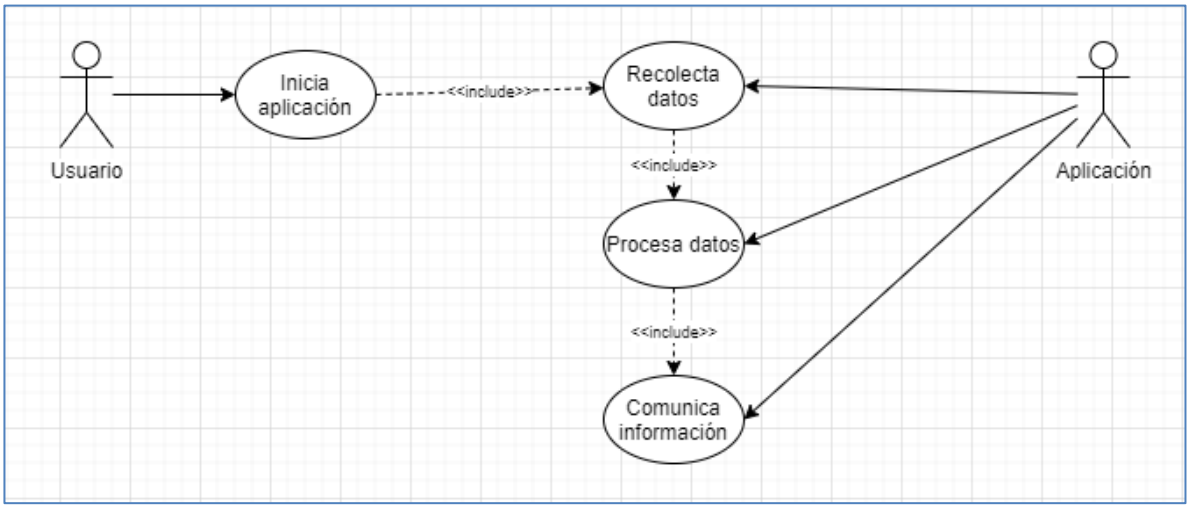


Figura 11: Caso de uso del proyecto

4.6 Diagramas de caso de uso de sistema

<table>
<tr><td colspan="2">Inicia aplicación</td></tr>
<tr><td colspan="2">Fecha: 22/12/2020</td></tr>
<tr><td colspan="2">Descripción: Permite al usuario iniciar la aplicación.</td></tr>
<tr><td colspan="2">Actores: Usuario</td></tr>
<tr><td colspan="2">Precondiciones: La aplicación debe tener acceso a internet debido a la necesidad de ingresar a un servicio web para obtener la dirección del usuario. También tiene que acceso al GPS y a la cámara</td></tr>
<tr><td colspan="2">Flujo normal:</td></tr>
<tr><td>Usuario:

1.- Inicia la aplicación.

3.- Selecciona una opción mediante su voz.</td><td>Sistema:

2.- Emite un mensaje de voz dando la opción de saber la ubicación o la de empezar a escanear con la cámara.

4.- Incluye el caso de uso “Procesar datos”.</td></tr>
<tr><td colspan="2">Postcondiciones: La aplicación sabe los datos que necesita obtener.</td></tr>
</table>

<table>
<tr><td colspan="2">Procesar datos</td></tr>
<tr><td colspan="2">Fecha: 22/12/2020</td></tr>
<tr><td colspan="2">Descripción: Permite a la aplicación obtener y procesar datos.</td></tr>
<tr><td colspan="2">Actores: Aplicación</td></tr>
<tr><td colspan="2">Precondiciones: La aplicación debe tener acceso a internet debido a la necesidad de ingresar a un servicio web para obtener la dirección del usuario. También tiene que acceso al GPS y a la cámara</td></tr>
<tr><td colspan="2">Flujo normal:</td></tr>
<tr><td>Aplicación:

1.- Solicita datos a la cámara o GPS.

4.- Utiliza los datos para traducirlos a texto.</td><td>Sistema:

2.- Captura los datos solicitados

3.- Entrega datos solicitados.</td></tr>
<tr><td colspan="2">Postcondiciones: La aplicación obtendrá información del celular.</td></tr>
</table>

<table>
<tr><td colspan="2">Compartir información</td></tr>
<tr><td colspan="2">Fecha: 22/12/2020</td></tr>
<tr><td colspan="2">Descripción: La aplicación compartirá la información con el usuario.</td></tr>
<tr><td colspan="2">Actores: Aplicación</td></tr>
<tr><td colspan="2">Precondiciones: La aplicación debe tener acceso a internet debido a la necesidad de ingresar a un servicio web para obtener la dirección del usuario. También tiene que acceso al GPS y a la cámara</td></tr>
<tr><td colspan="2">Flujo normal:</td></tr>
<tr><td>Aplicación:

1.- Genera mensajes de audio a partir de la información obtenida.</td><td>Sistema:

2.- Emite los mensajes de audio.</td></tr>
<tr><td colspan="2">Postcondiciones: El usuario escuchara la información obtenida por la aplicación.</td></tr>
</table>

5. Planificación del Diseño

5.1 Diagramas de secuencia

Las figuras 12, 13 y 14 son los diagramas de secuencia correspondientes a los casos de usos de sistemas descrito en el punto 4.6

- **Diagrama de secuencia "Iniciar Aplicación"**

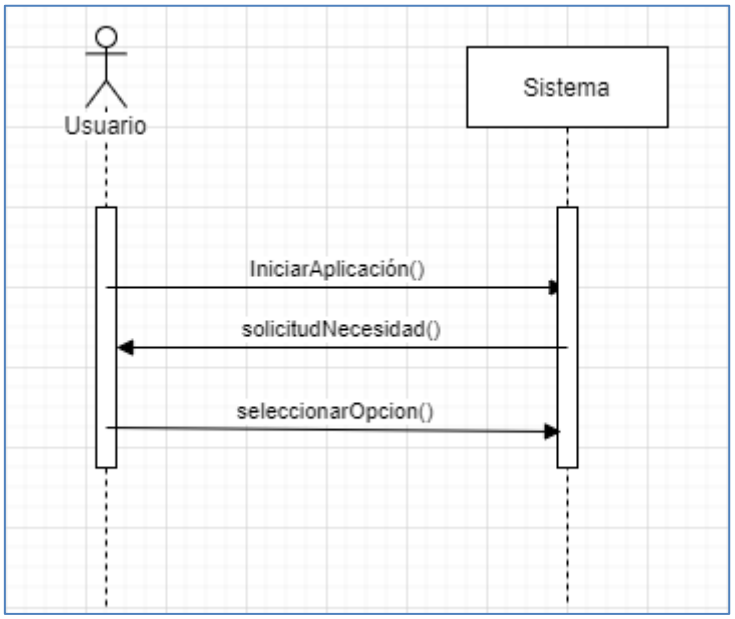


Figura 12: Diagrama de secuencia del caso de uso "Iniciar Aplicación"

- **Diagrama de secuencia “Procesar datos”**

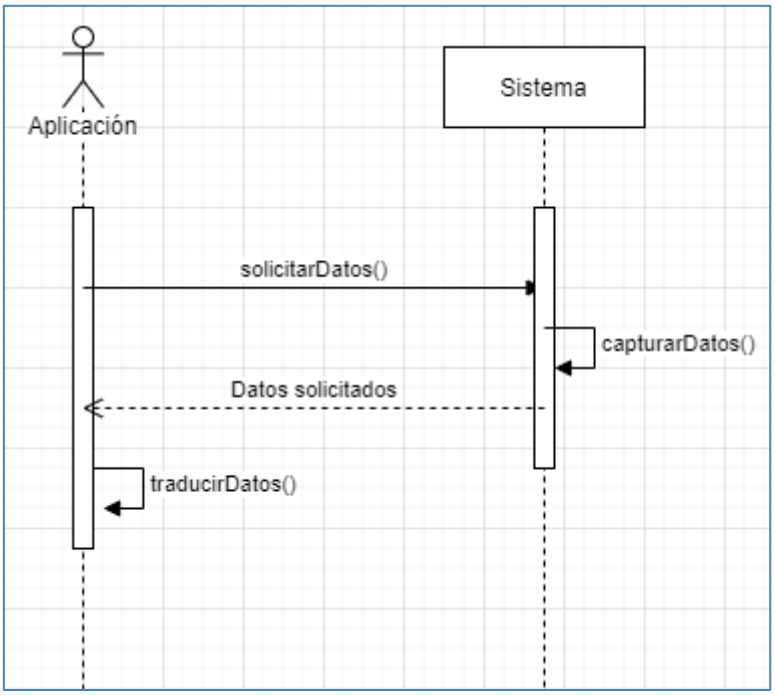


Figura 13: Diagrama de secuencia del caso de uso "Procesar datos"

- **Diagrama de secuencia “Compartir información”**

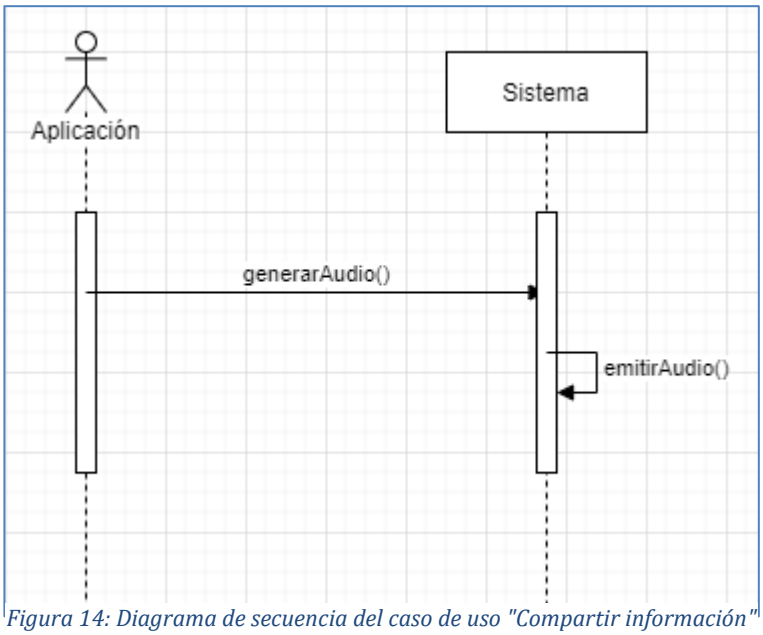


Figura 14: Diagrama de secuencia del caso de uso "Compartir información"

5.2 Modelo de clases

En la figura 15 se puede observar el modelo de clases con sus respectivas clases, atributos y funciones o metodos.

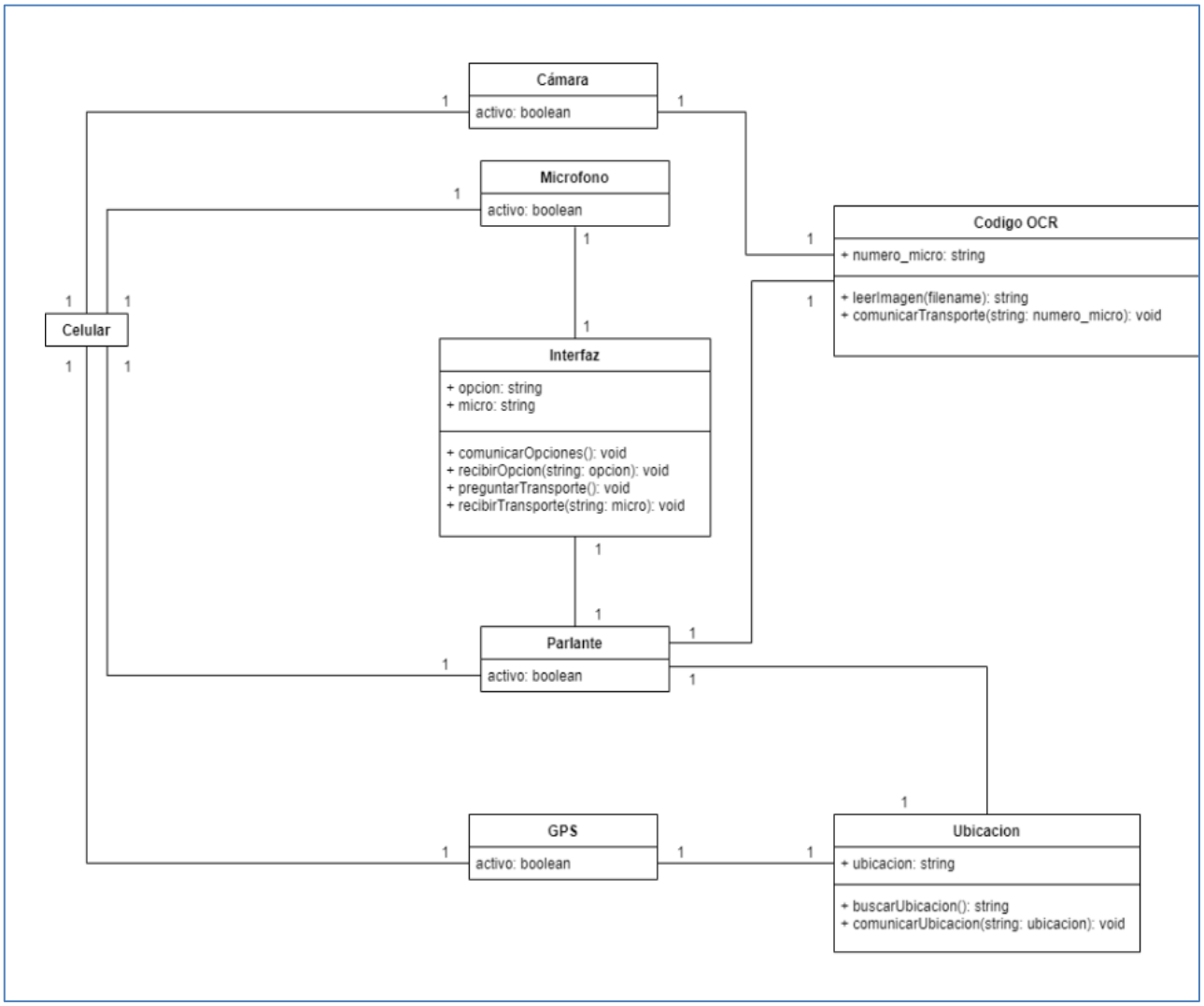


Figura 15: Modelos de clases

6. Implementación

6.1 Plan de integración

El proyecto consiste en la identificación de los tableros de las micros, para lograr este objetivo, se ha dividido el trabajo de la aplicación en varios puntos:

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

Para lograr esto, se ha utilizado las librerías Kivy, gtts, y speech_recognition, la primera permite crear la parte grafica de la aplicación, esta es una librería para python de código abierto usada para el desarrollo de aplicaciones, por otro lado, la librería gtts y función SoundLoader (proveniente de la librería kivy) son utilizadas para la comunicación de la aplicación con el usuario, pues gtts genera un archivo mp3 de audio a partir de texto y, luego, SoundLoader se encarga de reproducir estos archivos, esto ha servido como parte de la interfaz de la aplicación y para comunicar información pertinente para el cliente. Finalmente, la librería speechrecognition permite que el usuario comunique información a la aplicación, permitiendo a esta última que el audio generado por el usuario sea traducido a texto en la aplicación.

- **Identificación de la posición del usuario**

Este apartado se ha logrado gracias a las librerías plyer y geopy, la primera ha servido para obtener la latitud y longitud del usuario utilizando el sensor GPS del dispositivo y, la segunda, para obtener la información exacta de donde se encuentra el cliente, pues utiliza varios servicios web de geo codificación.

- **Identificación del tablero de la micro**

Para cumplir con este objetivo se han utilizado las librerías cv2, kivy, re y pyteseract. La librería cv2 permite la manipulación de las imágenes y videocaptura, este se ha utilizado para aumentar la precisión del escaneo de imágenes, la librería kivy se ha utilizado para usar la cámara del dispositivo, la librería re sirve para la utilización de expresiones irregulares, en este caso se utilizó para reconocer patrones, luego, la librería pyteseract sirve para el reconocimiento óptico de los frames de los caracteres.

6.2 Modelo de implementación

Como ya se ha mencionado con anterioridad, se han utilizado los modulos Kivy, Gtts, SoundLoader, speech_recognition, plyer, geopy, cv2, re y pyteseract, a continuación, se presentarán ejemplos de cómo se implementaron.

- **Kivy:**

```
from kivy.uix.button import Button

from kivy.app import App

from functools import partial

from gtts import gTTS

import speech_recognition as sr

from playsound import playsound

import time

class KivyButton(App):

    def audio1(self):···

    def eleccionOpcion(self):···

    def disable(self, instance, *args):···

    def update(self, instance, *args):···

    def build(self):
        mybtn = Button(text="Presione la pantalla para iniciar")

        mybtn.bind(on_press=partial(self.disable, mybtn))

        mybtn.bind(on_press=partial(self.update, mybtn))

        texto = "Presione la pantalla para iniciar"
        lenguaje = 'es'

        audio = gTTS(text = texto, lang = lenguaje, slow = False)
        audio.save('audio0.mp3')
        NOMBRE_ARCHIVO = "audio0.mp3"
        playsound(NOMBRE_ARCHIVO)
```

Figura 16: Parte del código donde se crea el boton

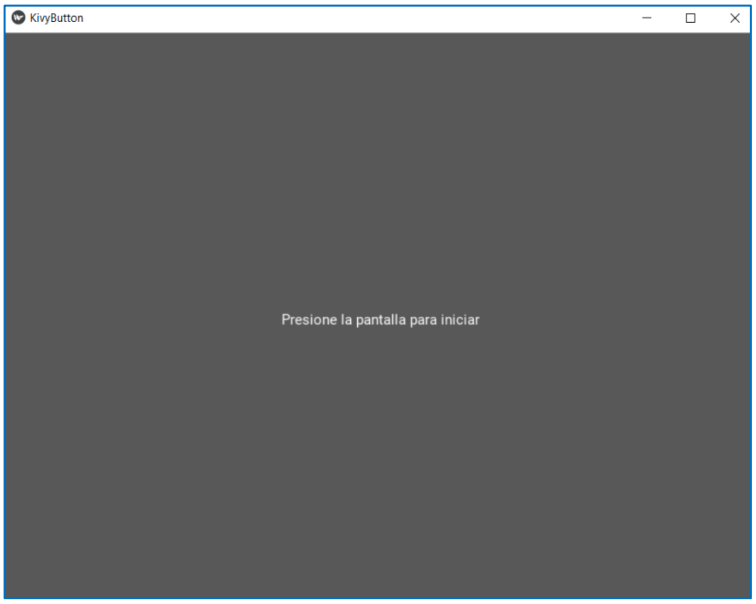


Figura 17: Pantalla donde se puede observar el botón creado

Como se puede apreciar en las figuras 16 y 17, kivy permite crear la parte grafica de la aplicación y agregar distintos elementos para generar una interfaz, en este caso, se crea un botón para iniciar con la aplicación.

- **Gtts, SoundLoader y speech_recognition:**

```
def update(self, instance, *args):


    texto = "¿Como puedo ayudarte? Di Opcion 1 Para continuar a la 'Busquera de tu trasnporte' , Di Opcion 2 para continuar al 'escaneo de transporte'"
    lenguaje = 'es'
    audio = gTTS(text = texto, lang = lenguaje, slow = False)
    audio.save('audio.wav')
    NOMBRE_ARCHIVO = 'audio.wav'
    sound = SoundLoader.load(NOMBRE_ARCHIVO)
    sound.play()
    #playsound(NOMBRE_ARCHIVO)

    time.sleep(10)
    r = sr.Recognizer()

    with sr.Microphone() as source:
        print('Speak Anything : ')
        audio = r.listen(source)

        try:
            text = r.recognize_google(audio)
            print('You said: {}'.format(text))
```

Figura 18: Parte del código donde se utiliza la función gTTs

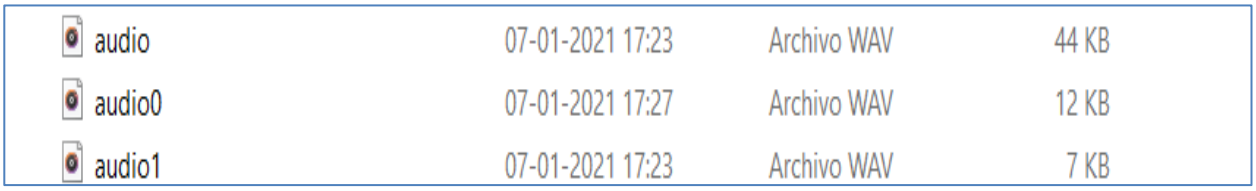


Figura 19: Creación de los audios

```
def update(self, instance, *args):


    texto = "¿Como puedo ayudarte? Di Opcion 1 Para continuar a la 'Busquera de tu trasnporte' , Di Opcion 2 para continuar al 'escaneo de transporte'"
    lenguaje = 'es'
    audio = gTTS(text = texto, lang = lenguaje, slow = False)
    audio.save('audio.wav')
    NOMBRE_ARCHIVO = 'audio.wav'
    sound = SoundLoader.load(NOMBRE_ARCHIVO)
    sound.play()
    #playsound(NOMBRE_ARCHIVO)

    time.sleep(10)
    r = sr.Recognizer()

    with sr.Microphone() as source:
        print('Speak Anything : ')
        audio = r.listen(source)

        try:
            text = r.recognize_google(audio)
            print('You said: {}'.format(text))

            if (text.lower() == "uno"):

                texto1 = "Su ubicacion es:"
                lenguaje1 = 'es'

                audio1 = gTTS(text = texto1, lang = lenguaje1, slow = False)
                audio1.save('audio1.wav')
                NOMBRE_ARCHIVO1 = 'audio1.wav'
```

Figura 20: Parte del código donde se utiliza la librería speech_recognition

Como se puede apreciar en las figuras 18, 19 y 20, la librería gtts permite generar unos archivos de audio en formato wav mediante la función gTTS, en el ejemplo, estos tienen como objetivo preguntar al usuario que quiere hacer, luego, estos son utilizados por la función SoundLoader, la cual proviene de la librería Kivy, de tal modo que la aplicación emite el audio. Finalmente, cuando el usuario de una respuesta, se utilizará la librería speech_recognition como en la figura 20, de modo que la respuesta del usuario se guarde como un texto.

- **Plyer y Geopy:**

```
obtain_location(self, **kwargs):
    ctx = ssl.create_default_context(cafile=certifi.where())
    geopy.geocoders.options.default_ssl_context = ctx
    geolocator = Nominatim(user_agent="luckyxdxc@gmail.com")
    location=geolocator.reverse(Point(kwargs['lat'],kwargs['lon']))
    location_cut=str(location.address)
    position=int(location_cut.find(', Arica'))
    location_cut=location_cut[0:position]
    popupWindow = Popup(title = 'L0calizacion', content=Label(text=location_cut), size_hint= (None,None), size=(400,400))
    popupWindow.open()
```

Figura 21: Parte del código donde se ocupa la librería Plyer y Geopy

Como se puede apreciar en la figura 21, la librería Plyer permite utilizar el GPS de los dispositivos (en este caso, un smartphone), esto implica que se puede obtener la latitud y longitud cercana o exacta del usuario, esta información es utilizada con la librería Geopy para realizar una geocodificación, esto quiere decir que se obtendrá la dirección asociada a la latitud y longitud obtenida.

- **cv2, Kivy, re y Pyteseract:**

```
def update(self, dt):
    ret, frame = self.capture.read() # capturar frame por frame
    if ret:
        # convertir a textura
        buf1 = cv2.flip(frame, 0)
        buf = buf1.tostring()
        image_texture = Texture.create(
            size=(frame.shape[1], frame.shape[0]), colorfmt='bgr')
        image_texture.blit_buffer(buf, colorfmt='bgr', bufferfmt='ubyte')
        #muestra la imagen en camara
        self.texture = image_texture
        #timestr = time.strftime("%Y%m%d_%H%M%S")
        read_and_write(frame)
```

Figura 22: Parte del código donde se actualiza el cuadro de la imagen

```python
def ocr_detector(img):
    d = pytesseract.image_to_data(img, output_type=Output.DICT)
    keys = list(d.keys())
    number_pattern = r'[\d+\$]+$'
    n_boxes = len(d['text'])
    for i in range(n_boxes):
        if int(d['conf'][i]) > 60:
            if bool(re.match(number_pattern, d['text'][i])):
                (x, y, w, h) = (d['left'][i], d['top'][i], d['width'][i], d['height'][i])
                img = cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
    return img
```

Figura 24: Parte del código donde se ocupa la función ocr_detector es el reconocedor óptico de caracteres

```python
def read_img_invert(filename):
    img = cv2.imread(filename)
    img = cv2.resize(img, (620,480) )
    #Escala de Grises, Desenfoque Gausiano, umbral de UTSU
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (3,3), 0)
    thresh = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]
    #se transforma la imagen para quitar el ruido que produzca esta y se invierte
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3,3))
    opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations=1)
    invert = 255 - opening
    #print(pytesseract.image_to_string(img, config='--psm 6',lang='spa'))
    return invert
```

Figura 23: Parte del código donde se ocupa la función read_img_invert que convierte la imagen que está grabando

Como se puede apreciar en las figuras 22, 23 y 24, la función update realiza la actualización del cuadro de imagen en el instante que se encuentra disponible, en el caso que se encuentre disponible, lo primero que hace es voltear la imagen con cv2 y se convierte en string para la lectura como textura del kivy(smartphone), luego esa textura muestra el instante captado en la cámara para mostrarla como imagen en el teléfono. La función ocr_detector es el reconocedor óptico de caracteres, toma la imagen enviada y la convierte en datos, de la cual se crea una variable para que solo reconozca números (number_pattern), y si tiene n_boxes que es el número de texto que tiene la imagen. La función for revisa el rango de todos los textos y se hace una condición con la variable creada anteriormente (que solo sean números), si esa condición se cumple, en la imagen final solo aparecerá un rectángulo donde están los números. La función read_img_invert convierte la imagen que está grabando y la manipula a un tamaño más óptimo para su lectura, se modifica la escala de grises, su desenfoque gaussiano y el umbral de UTSU, para luego quitar el ruido que produzca la imagen y se invierten sus colores. Todo usando la libreria de CV2.

6.3 Pruebas de la aplicación

En un principio, se fue probando la aplicación por partes, esto quiere decir que se dividió la aplicación en partes (como se mencionó en el capítulo 6), durante estas pruebas ocurrió lo siguiente:

- **Problemas implementando la librería plyer:**

  El funcionamiento de la librería plyer es vital, puesto que permite obtener la latitud y longitud del usuario utilizando el sensor GPS del Smartphone, sin embargo, debido a problemas con la documentación de esta, no se lograba entender cómo implementarla junto a la aplicación, por lo que se tuvieron que hacer varias pruebas para lograr que funcionara.

- **Problemas implementando librería de texto a voz:**

  En un principio, se utilizaba la librería Pytsx3 para realizar la traducción de texto a audio en la aplicación, sin embargo, al momento de compilar la aplicación para utilizarla en el Smartphone, ocurrían problemas con el reconocimiento de esta, por lo que se tuvo que reemplazar por la librería gTTS y la función SoundLoader (que se puede importar de la librería Kivy) para realizar la labor de esta.

- **Problema con el reconocimiento de fuentes de los tableros de las micros:**

  Al momento de intentar reconocer los tableros de las micros, la aplicación tenía problemas, pues necesitaba información para poder reconocer el tipo de fuente con el que estos estaban hechos, es por esto que se necesitan varias fotos de los distintos tableros de las micros para que la aplicación pueda comprender el patrón de estos.

7. Conclusiones

Luego de todo lo expuesto a lo largo del informe, se puede concluir como equipo los siguientes puntos:

En primer lugar, hay que destacar que el objetivo del proyecto es crear una aplicación que permita escanear los tableros de las micros de la ciudad para indicar la línea de recorrido a la que corresponde. La aplicación a diseñar está dirigida hacia un gran rango de personas, que van desde personas totalmente ciegas o con ceguera parcial, con problemas de vista de distintos tipos y gente que sea analfabeta. Para lograr este objetivo se planificó y se implementó la creación de una aplicación con una interfaz que funcione con mensajes de audio y comandos de voz, además de un mecanismo de identificación de los tableros de las micros, implementado gracias a una herramienta denominada OpenCV. Finalmente, se le ha integrado una funcionalidad a la aplicación para indicar la posición del usuario, esto a través del uso del GPS del celular y de un servicio web que provee la información de la localización de este.

Por parte del proyecto, se concluye que la creación de aplicaciones utilizando la librería Kivy es mucho más fácil estando en sistemas operativos como macOS o Linux, pues es en estos donde la herramienta Buildozer, que es con la cual se empaquetan las aplicaciones y con la cual Kivy está orientada a trabajar, puede funcionar correctamente. Sin embargo, lo anteriormente mencionado puede ser realizado sin Buildozer, pero de forma más dificultosa y requeriría mayor tiempo, por lo que una mejor opción es ocupar una máquina virtual para utilizar la herramienta.

Entre otros aspectos, cabe destacar que, debido al objetivo del proyecto, resultó interesante darle un enfoque distinto al desarrollo de la aplicación, pues la interfaz no podía ser hecha con texto o imágenes, y la información que se quisiera compartir con el usuario no podía ser entregada de manera convencional, por lo que se debía pensar con cuidado los detalles que se quisieran incorporar en la aplicación, como, por ejemplo, en un momento del desarrollo casi se incorpora un mapa en la aplicación, lo cual resulto en una obvia idea descartada, ya que una persona con problemas a la vista no iba a poder apreciar este detalle y solo resultaría en una pérdida de tiempo. Otro ejemplo que se puede compartir, es el aspecto que involucra recibir una respuesta del usuario, pues se concluyó que lo mejor sería recibir respuestas cortas, esto puede ser apreciado cuando el usuario debe decir “1” o “2” para que la aplicación sepa si debe dar la ubicación del usuario o activar la cámara para empezar a escanear los tableros.

En segundo lugar, cabe mencionar que, la falta de experiencia para crear una aplicación crea bastantes desventajas, en especial cuando se deben cumplir con fechas de entrega, puesto que se debe de estudiar o, en otros casos, ir probando y fallando hasta tener éxito para que algo funcione como se espera. Sin embargo, a pesar de lo mencionado anteriormente, hay casos en los que no se encuentra la solución esperada, por lo que hay que buscar alternativas que reemplacen los módulos que se estaban utilizando en un principio.

Por otro lado, gracias a la realización de este proyecto, se han adquirido conocimientos nuevos, al igual que experiencia, la cual ha permitido al equipo crecer, ya sea por el hecho de adquirir experiencia trabajando con aplicaciones, creando algoritmos con Python, en la formulación de proyectos y trabajando en equipo.

8. Referencias

Básica:

- Sepulveda, M., & Silva, C. (s.f.). *Informe de situación actual y estado del arte*.https://cetram.org/wp/wp-content/uploads/2014/01/Proyecto-de-Accesibilidad- Para-Personas-con-Discapacidad-Visual.pdf

- *Kivy: Cross-platform Python Framework for NUI Development*. (s.f.). Obtenido de https://kivy.org/#home

- Zelic, F., & Sable, A. (s.f.). *[Tutorial] OCR in Python with Tesseract, OpenCV and Pytesseract*. Obtenido de https://nanonets.com/blog/ocr-with-tesseract/#ocrwithpytesseractandopencv

Complementaria:

- Apuntes del profesor