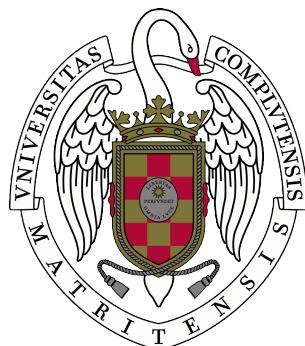

Aplicación móvil para facilitar la interpretación de imágenes a personas con discapacidad visual

**Mobile application to improve the interpretation
of images for people with visual disabilities**



**Trabajo de Fin de Grado
Curso 2023–2024**

Autores

Walid Bousnitra Bousnitra
Daniel Leo Cansado
Carlos Martín Recio
Sirma Sosa Fernández

Directores

Alberto Díaz Esteban
Raquel Hervás Ballesteros

Grado en Ingeniería Informática

Facultad de Informática

Universidad Complutense de Madrid

Aplicación móvil para facilitar la interpretación de imágenes a personas con discapacidad visual

Mobile application to improve the interpretation of images for people with visual disabilities

Trabajo de Fin de Grado en Ingeniería Informática

Autores

Walid Bousnitra Bousnitra
Daniel Leo Cansado
Carlos Martín Recio
Sirma Sosa Fernández

Directores

Alberto Díaz Esteban
Raquel Hervás Ballesteros

Convocatoria: Junio 2024

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

27 de Mayo de 2024

Agradecimientos

Agradecimientos a nuestros tutores, Alberto y Raquel, por confiar en nosotros para desarrollar el proyecto y por ayudarnos desde el primer día en todo lo posible. Agradecer también a Víctor Alberto y Margarita por dedicarnos su tiempo para mejorar nuestro trabajo ya que su valoración es muy importante para nosotros.

Resumen

En su día a día, las personas con discapacidad visual se encuentran con la dificultad de poder obtener información de una imagen en su dispositivo móvil. Hay múltiples aplicaciones destinadas a solucionar estos problemas de interpretación de imágenes. La gran mayoría de ellas se enfrentan a importantes obstáculos respecto a funcionamiento y eficacia, ya que no llegan a cubrir necesidades en su totalidad o se centran solamente en una parte del problema sin llegar a ser totalmente útiles para personas con este tipo de discapacidad. Por otro lado, hay problemas que no encuentran solución en el mercado de aplicaciones móviles como el reconocimiento facial de personas en aplicaciones que describan imágenes.

Haciendo uso de las tecnologías de descripción de imágenes existentes, se ofrece al usuario la siguiente experiencia de interpretación de una imagen cargada en su dispositivo móvil:

- Una primera descripción de la imagen en su totalidad, la cual permite a la persona que la escucha hacerse una idea general de lo que se va a encontrar una vez comience a explorar la imagen.
- Seguido de esta primera descripción, se le nombran al usuario los objetos o entidades identificados.
- Explorando la imagen mediante toques, el usuario puede ir ubicando cada uno de los objetos de la imagen. Además, se puede recibir una descripción más específica de cada uno de ellos, como por ejemplo género para personas o color para objetos.
- Una vez ubicados los objetos o las personas identificadas, el usuario podrá hacer una pulsación larga sobre estos para recibir una descripción detallada de los mismos.

En resumen, hemos alcanzado nuestro objetivo de proporcionar más información relevante a través de nuestra aplicación. Logrando agrupar funcionalidades que antes estaban dispersas en distintas aplicaciones. Para comprobar la efectividad de las soluciones desarrolladas realizamos evaluaciones con personas que presentaban diversas discapacidades visuales y pusimos en práctica todas las funcionalidades que desarrollamos. Estas personas pudieron extraer información significativa de las imágenes sin tener conocimiento previo sobre ellas y nuestra implementación les facilitó la interpretación de las imágenes. Además, nos indicaron aspectos que les gustaría que implementáramos en futuras versiones, como mejorar la accesibilidad para que puedan utilizar la aplicación sin ayuda externa y la capacidad de reconocer personas que aparecen con frecuencia en sus galerías de fotos.

Todo el contenido de este trabajo está publicado en el siguiente repositorio:
<https://github.com/NILGroup/TFG-2324-ImagenesCiegos>

Palabras clave

Discapacidad Visual, Interpretación de Imágenes, Aplicaciones Móviles, Accesibilidad, Experiencia del Usuario.

Abstract

Mobile application to improve the interpretation of images for people with visual disabilities

In their day-to-day, visually impaired people find it difficult to obtain information from an image on their mobile device. There are many applications designed to solve these image interpretation problems. The vast majority of them face significant obstacles in terms of functioning and effectiveness, as they do not cover all needs or focus only on a part of the problem without becoming fully useful for people with this type of disability. On the other hand, there are problems that cannot be solved in the mobile application market, such as facial recognition of people in applications that describe images.

Using existing image description technologies, the user is offered the following experience of interpreting an image loaded on their mobile device:

- First, a description of the image in its entirety, which allows the listener to get a general idea of what to find once you begin to explore the image.
- Following this first description, the user is named the identified objects or entities.
- By tapping the image, the user can locate each of the objects in the image. In addition, you can receive a more specific description of each of them, such as gender for people or color for objects.
- Once the objects or persons identified are located, the user will be able to long press on them for a detailed description.

To sum up, we have achieved our goal of providing more relevant information through our application. By bringing together functionalities that were previously scattered in different applications. To test the effectiveness of the developed solutions, we carried out evaluations with people with various visual impairments and put into practice all the functionalities we developed. These people were able to extract meaningful information from the images without any prior knowledge about them and our implementation made it easier for them to interpret the images. They also indicated aspects that they would like us to implement in future versions, such as improving accessibility so that they can use the application without external assistance and the ability to recognise people who appear frequently in their photo galleries.

All the content of this work is published in the following repository: <https://github.com/NILGroup/TFG-2324-ImagenesCiegos>

Keywords

Visual Impairment, Image Interpretation, Mobile Applications, Accessibility, User Experience

Índice

Resumen	vii
Abstract	ix
1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	2
1.3. Estructura del Documento	3
2. Introduction	5
2.1. Motivation	5
2.2. Objectives	6
2.3. Structure of the Document	6
3. Estado de la Cuestión	9
3.1. Tecnologías de descripción de imágenes	9
3.1.1. Imagga	9
3.1.2. COCO	11
3.1.3. OID	12
3.1.4. Google Cloud Vision	13
3.1.5. RoboFlow	15
3.1.6. Hugging Face	15
3.1.7. Clarifai	16
3.1.8. Microsoft Azure	17
3.1.9. Google Firebase	19
3.1.10. ColorThief	21
3.1.11. Palette	21
3.2. Trabajo de Fin de Grado Anterior	22
3.2.1. Arquitectura	22
3.2.2. Diseño centrado en el usuario	23
3.2.3. Conclusiones y trabajo futuro	24

4. Funcionamiento de la aplicación	25
4.1. Selección de imagen	25
4.2. Descripción General de la imagen	27
4.3. Navegación por la imagen	28
4.3.1. Descripción Cuantitativa	28
4.3.2. Objetos	28
5. Implementación del Sistema	33
5.1. Prototipo tecnológico	33
5.2. Arquitectura	34
5.3. Servidor	34
5.3.1. Cloud Functions	35
5.3.2. Modelos	37
5.4. Cliente	40
5.5. Modularidad	45
5.6. Gestión de coordenadas	46
5.7. Gestión de los colores	46
5.8. Limitaciones actuales de la aplicación	49
5.8.1. Talkback	49
5.8.2. Descripción del fondo de las imágenes	50
6. Evaluación	53
6.1. Participantes	53
6.2. Diseño experimental	53
6.2.1. Tareas	55
6.2.2. Materiales y Entorno de evaluación	55
6.3. Desarrollo de la sesión de evaluación	55
6.3.1. Primera imagen para Víctor Alberto (Figura 6.2)	56
6.3.2. Primera imagen para Margarita (Figura 6.4)	57
6.3.3. Segunda imagen para Víctor Alberto (Figura 6.5)	58
6.3.4. Segunda imagen para Margarita (Figura 6.6)	59
6.3.5. Tercera imagen para ambos usuarios (Figura 6.7)	61
6.4. Análisis de los datos obtenidos	62
6.4.1. SUS	64
7. Conclusiones y Trabajo Futuro	67
7.1. Conclusiones	67
7.2. Trabajo Futuro	69
8. Conclusions and Future Work	71
8.1. Conclusions	71
8.2. Future Work	73
9. Contribuciones Personales	75

9.1. Wалид Буснитра	75
9.2. Сирма Соса Фернандес	77
9.3. Карлос Мартин Рекио	80
9.4. Даниэль Лео Кансадо	81
Bibliografía	85

Índice de figuras

3.1.	Visualización de servicio de <i>Tagging</i>	10
3.2.	Visualización de servicio de Categorización	10
3.3.	Visualización de servicio de <i>Background Removal</i>	11
3.4.	Ejemplo de segmentación en <i>COCO</i>	12
3.5.	Ejemplo de cómo se muestra la información de <i>COCO</i> mediante un JSON	12
3.6.	Ejemplo de imagen anotada con OID	14
3.7.	Ejemplo de operación de aumento de datos con Roboflow	16
3.8.	Ejemplo de detección de objetos con Roboflow	17
3.9.	Ejemplo de detección de objetos en imágenes con el modelo <i>facebook/detrresnet-50-dc5</i> de <i>Hugging Face</i>	18
3.10.	Una de las descripciones proporcionadas con Azure para esta imagen es “ <i>Foto en blanco y negro de una ciudad</i> ”	19
3.11.	Ejemplos de funcionalidades Microsoft Azure	20
4.1.	Inicio de la aplicación	26
4.2.	Selección de imagen	26
4.3.	Opciones para seleccionar imagen	26
4.4.	Display de la imagen según su orientación	27
4.5.	Imagen para la que se genera la descripción “ <i>mujer sentada en un banco con un perro y un teléfono celular</i> ”	28
4.6.	Imagen con espacios en blanco	29
4.7.	Se han obtenido: dos personas y un sofá	29
4.8.	Objetos: Coche, Coche 1	30
4.9.	Información adicional persona	31
4.10.	Información adicional objeto	31
4.11.	una mujer inclinada sobre una mesa de billar con un taco en la mano	32
5.1.	Diagrama de las API	34
5.2.	Imagen para explicar los modelos	37
5.3.	<i>Layout</i> de la aplicación	40
5.4.	Diagrama UML	41
5.5.	Dibujo Bounding Box	43

5.6.	Representación de los valores necesarios para los cálculos de coordenadas	48
5.7.	Modelo de profundidad	50
5.8.	Transformación de profundidad	51
6.1.	Imágenes para los usuarios	56
6.2.	Primera imagen para Víctor Alberto	57
6.3.	Sesión de evaluación	57
6.4.	Primera imagen para Margarita	58
6.5.	Segunda imagen para Víctor Alberto	59
6.6.	Segunda imagen para Margarita	60
6.7.	Tercera imagen para ambos	61

Índice de tablas

5.1. Posibles Encuadres	47
6.1. Resultados encuesta SUS	64

Capítulo 1

Introducción

El primer capítulo empieza con una breve introducción sobre el tema en cuestión así como la motivación, los objetivos y la estructura del documento.

“Lo único peor a no tener vista es no tener visión”
— Hellen Keller

1.1. Motivación

La ONCE¹ considera discapacidad visual a la limitación total o parcial de la vista. Se mide a través de diversos parámetros, como la capacidad lectora de cerca y de lejos, el campo visual o la agudeza visual. Las personas con discapacidad visual presentan una clara desventaja en una sociedad altamente conectada y dependiente de la tecnologías. Estas tecnologías deben su buen funcionamiento a la interacción entre aplicación y persona mediante información visual que se recibe y con la que se interactúa.

El ser humano interacciona con su entorno gracias a los estímulos del mundo exterior, percibidos a través de los sentidos. Uno de los sentidos más importantes para el ser humano es el de la vista. Las imágenes son la forma de interpretación que mejor encaja con la inteligencia visual-espacial del ser humano. Sin embargo hay otras formas de percibir el entorno como a través del oído (ecolocalización o audiodescripción).

Las personas con discapacidad visual enfrentan considerables desafíos al llevar a cabo las tareas cotidianas. En los últimos años, han surgido adaptaciones y tecnologías diseñadas para fomentar la inclusividad en la interacción con dispositivos, como la accesibilidad que ofrece *iOS*. Esta innovación permite que las personas con discapacidad visual manejen un dispositivo móvil con la misma comodidad que aquellos que no enfrentan estas dificultades, obteniendo así una mejora significativa en su calidad de vida.

La existencia de aplicaciones móviles adaptadas ha facilitado que las personas invidentes utilicen sus teléfonos de manera sencilla, mejorando la lectura de correos electrónicos, la respuesta a mensajes y la comunicación con familiares. A pesar de

¹<https://www.once.es/dejanos-ayudarte/la-discapacidad-visual>

estos avances, es importante señalar que el mundo aún no está completamente adaptado para satisfacer las necesidades de las personas invidentes.

Es una realidad que la tecnología se ha convertido en una herramienta indispensable en la cotidianidad actual. Por consiguiente, su uso diario se debe tanto a la efectividad como a la facilidad que proporciona su uso y comprensión. Por ello, el desarrollo de una aplicación de interpretación de imágenes, ayudará a las personas con discapacidad visual a una mayor inclusión en la sociedad en la que se encuentra inmersa, puesto que les proporcionará información y comprensión de la realidad que les rodea, la cual debido a su discapacidad se podría ver mermada. Mediante una descripción detallada, precisa y rápida de imágenes, se proporcionaría mayor independencia a la hora de obtener esa información, que de otra manera, estas personas, se verían condicionadas a necesitar ayuda externa para comprenderla. Dicha aplicación se podría convertir en un recurso de gran utilidad, para aportar un plus a su autonomía personal.

1.2. Objetivos

Nuestro principal objetivo es el desarrollo de una aplicación móvil para que las personas con discapacidad visual puedan interpretar de manera precisa las imágenes. Para ello es necesario la realización de distintos objetivos.

A partir del trabajo previamente realizado en su Trabajo de Fin de Grado (Amor Sanz et al., 2023) por otros compañeros y habiendo analizado el mismo, contamos con una base sólida. Nos podemos hacer una idea de cuales son las principales características y requisitos a incluir en nuestra aplicación gracias a las entrevistas ya realizadas a personas con visibilidad reducida, sentando así las bases de nuestra aplicación. También llevaron a cabo entrevistas para la evaluación de la aplicación final, es por ello que contamos con un gran feedback acerca de qué podemos hacer para mejorar la aplicación.

Ya con la base de la aplicación definida, investigaremos distintos algoritmos de procesado de imágenes. Sopesaremos todas las opciones eligiendo las mejores alternativas que den pie a una aplicación de mayor calidad. Estos modelos tienen que encargarse de proporcionar una descripción de la imagen y por otro lado detectar objetos dentro de la misma. Gracias a estos modelos se podrá proporcionar una descripción general de la imagen y otra más concreta. La descripción general se basará en explicar a grandes rasgos qué se encuentra en la foto. Mientras que la descripción más concreta, será sobre los propios objetos que se encuentran en la imagen, dando información sobre los mismos.

Para implementar esta aplicación usaremos un sistema que se conecte a un servidor que incluirá los modelos IA para el procesado de imágenes. Este servidor devolverá la información del procesado al cliente. En el cliente tendremos la aplicación que recibirá la información de procesado de servidor. La aplicación debe dar la posibilidad de elegir qué archivo quiere elegir para su procesamiento. Con el uso de un sistema cliente-servidor haremos que la aplicación sea portable.

1.3. Estructura del Documento

Este proyecto sigue una estructura en la que cada uno de los capítulos contienen lo siguiente:

- El capítulo 1 y 2 contiene una introducción al tema en cuestión, así como la motivación y los objetivos del trabajo. Siendo el capítulo 1 en español y el 2 en inglés.
- El capítulo 3 contiene el estado de la cuestión que explica las tecnologías actuales. Principalmente se describen las funcionalidades de descripción de imágenes y detección de objetos.
- El capítulo 4 contiene el funcionamiento de la aplicación. En este apartado se muestran todas las funcionalidades disponibles para el usuario.
- El capítulo 5 contiene el desarrollo en detalle de la implementación del sistema. El núcleo de este capítulo está en la arquitectura que muestra toda la implementación realizada pero también se incluyen aspectos importantes que han ido surgiendo durante el desarrollo.
- El capítulo 6 contiene una evaluación de la aplicación. La evaluación se realizó con usuarios reales y se incluye su interacción con la aplicación y su opinión sobre la misma.
- El capítulo 7 y 8 contiene las conclusiones tras haber realizado el trabajo así como ideas para versiones futuras. Siendo el capítulo 7 en español y el 8 en inglés
- El capítulo 9 contiene las contribuciones personales de cada uno de los integrantes del equipo.

Finalmente, se incluye un apartado con la bibliografía que respalda las tecnologías externas que se han usado.

Capítulo 2

Introduction

The first chapter starts with a brief introduction to the subject at hand, as well as the motivation, objectives, and structure of the document.

“The only thing worse than being blind is having sight but no vision”
— Hellen Keller

2.1. Motivation

The ONCE¹ defines visual impairment as the total or partial limitation of vision. It is measured through various parameters, such as the ability to read up close and at a distance, the visual field, or visual acuity. People with visual impairment are at a clear disadvantage in a highly connected and technology-dependent society. These technologies successful because of the interaction between the application and the person through visual information that is received and interacted with.

Human beings interact with their environment thanks to stimuli from the outside world, perceived through the senses. One of the most important senses for humans is sight. Images are the form of interpretation that best fits the visual-spatial intelligence of humans. However, there are other ways to perceive the environment, such as through hearing (echolocation or audiodescription).

People with visual impairment face considerable challenges in carrying out daily tasks. In recent years, adaptations and technologies have emerged designed to promote inclusivity in the interaction with devices, such as the accessibility offered by *IOS*. This innovation allows people with visual impairment to handle a mobile device with the same ease as those who do not face these difficulties, thus achieving a significant improvement in their quality of life.

The existence of adapted mobile applications has made it easier for blind people to use their phones, improving the reading of emails, responding to messages, and communicating with family members. Despite these advances, it is important to note that the world is still not fully adapted to meet the needs of blind people.

¹<https://www.once.es/dejanos-ayudarte/la-discapacidad-visual>

It is a reality that technology has become an indispensable tool in today's daily life. Consequently, its daily use is due both to its effectiveness and to the ease it provides in its use and understanding. Therefore, the development of an image interpretation application will help people with visual impairment to achieve greater inclusion in the society in which they are immersed, as it will provide them with information and understanding of the reality that surrounds them, which, due to their disability, could be diminished. Through a detailed, precise, and rapid description of images, greater independence in obtaining this information would be provided, which otherwise, these people would be conditioned to need external help to understand. Such an application could become a very useful resource, adding to their personal autonomy.

2.2. Objectives

Our main objective is the development of a mobile application so that people with visual impairment can accurately interpret images. For this, it is necessary to achieve various goals.

Based on the work previously done in their Final Degree Project (Amor Sanz et al., 2023) by other colleagues and having analyzed it, we have a solid foundation. We can get an idea of the main features and requirements to include in our application thanks to the interviews already conducted with people with reduced visibility, thus laying the foundation for our application. They also conducted interviews for the evaluation of the final application, which is why we have great feedback on what we can do to improve the application.

With the application base already defined, we will investigate different image processing algorithms. Trying all the options and choosing the best alternatives that lead to a higher quality application. These models must provide a description of the image and, on the other hand, detect objects within it. Thanks to these models, a general description of the image and a more specific one can be provided. The general description will be based on explaining in broad strokes what is in the photo, while the more specific description will be about the objects in the image, providing information about them.

To implement this application, we will use a system that connects to a server that will include the AI models for image processing. This server will return the processing information to the client. On the client, we will have the application that will receive the processing information from the server. The application must give the option to choose which file it wants to process. Using a client-server system will make the application portable.

2.3. Structure of the Document

This project follows a structure in which each of the chapters contains the following:

- Chapters 1 and 2 contain an introduction to the subject, as well as the moti-

vation and objectives of the work, with Chapter 1 in Spanish and Chapter 2 in English.

- Chapter 3 contains the state of the art, explaining current technologies. Primarily, image description and object detection functionalities are described.
- Chapter 4 contains the operation of the application. This section shows all the functionalities available to the user.
- Chapter 5 contains the detailed development of the system implementation. The core of this chapter is the architecture that shows all the implementation done, but important aspects that have arisen during the development are also included.
- Chapter 6 contains an evaluation of the application. The evaluation was carried out with real users and includes their interaction with the application and their opinion about it.
- Chapters 7 and 8 contain the conclusions after the work was done and ideas for future versions, with Chapter 7 in Spanish and Chapter 8 in English.
- Chapter 9 contains the personal contributions of each team member.

Finally, a section with the bibliography is included, supporting the external technologies used.

1.

Capítulo 3

Estado de la Cuestión

En este capítulo se analizan diferentes tecnologías y herramientas que existen actualmente para la descripción de imágenes. Tanto modelos IA para la descripción de imágenes, como modelos IA para la detección de objetos. También se incluye el proyecto “*Desarrollo de una aplicación móvil para la generación de descripciones de imágenes para personas con discapacidad visual*” (Amor Sanz et al., 2023), que fue el precursor del trabajo actual.

3.1. Tecnologías de descripción de imágenes

A continuación, se explicarán algunos de los descriptores de imágenes más punteros que podemos encontrar actualmente. Se han utilizado para desarrollar nuestros objetivos, así como para realizar una investigación y poder llevarla a cabo con más exactitud.

3.1.1. Imagga

*Imagga*¹ es una plataforma en línea y de pago que ofrece multitud de servicios sobre el reconocimiento de imágenes para desarrolladores y empresas. Es muy completa, ya que proporciona diferentes APIs para poder utilizar en una misma imagen, ya sea sacar el fondo de una imagen o para mostrar las distintas categorías en las que se clasifica. A causa de que el servicio gratuito es demasiado limitado como para poder utilizarla, hemos optado por descartarla. La plataforma ofrece una serie de funciones entre las que se encuentran:

- Color: Analiza los colores de la imagen y los categoriza por porcentajes según la cantidad de píxeles que ocupan en la imagen.
- *Tagging*: Tal y como se puede ver en la Figura 3.1, detecta los diferentes objetos contenidos en una imagen y crea unas etiquetas que se le añaden a la imagen según la información obtenida de los objetos de la foto.

¹<https://imagga.com>

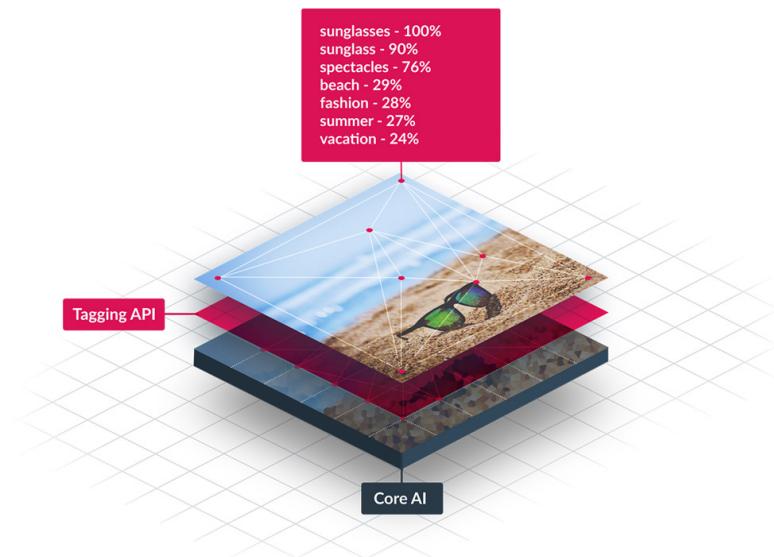


Figura 3.1: Visualización de servicio de *Tagging*

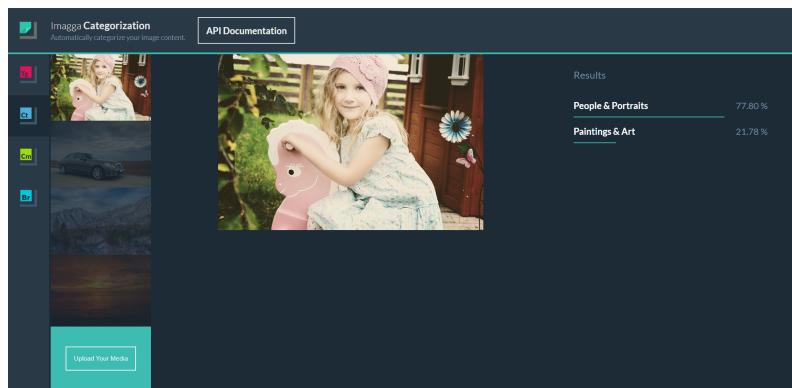


Figura 3.2: Visualización de servicio de Categorización

- **Categorización:** Elige entre una variedad de categorías para clasificar de forma general las imágenes. Se puede ver un ejemplo en la Figura 3.2.
- ***Facial Recognition:*** Detección de rasgos faciales. Aunque no se especifica de quiénes son los rasgos al aparecer más de una persona.
- ***Background Removal:*** Como se puede ver en la Figura 3.3, está diseñada para eliminar el fondo de una imagen mediante el análisis del objeto principal en la misma.

Cabe destacar que contiene una base de datos completa, que reconoce una amplia paleta de colores y además es capaz de realizar descripciones en 46 idiomas.

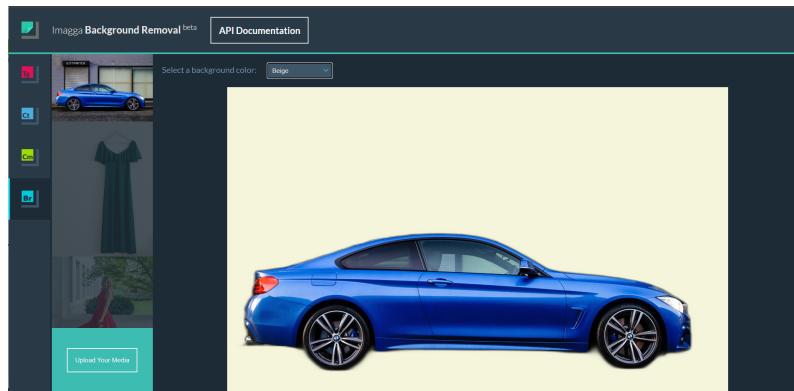


Figura 3.3: Visualización de servicio de *Background Removal*

3.1.2. COCO

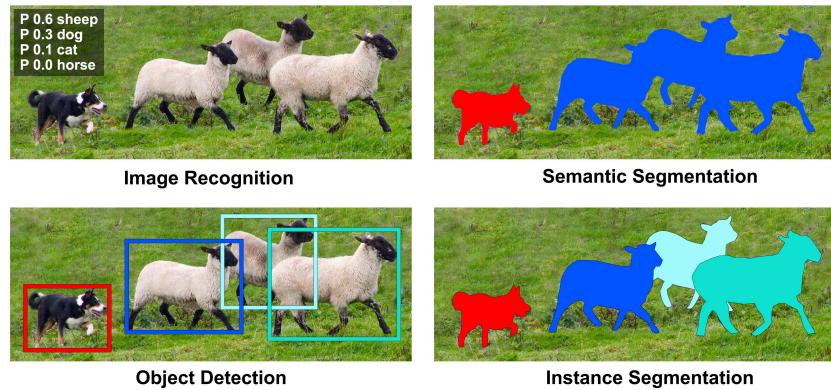
*COCO*²(*Common Objects in Context*) es un *dataset* enfocado en la detección de objetos y segmentación de imágenes, publicado por *Microsoft*. La interpretación de imágenes es un objetivo muy presente en la visión por computación. Esto implica el reconocimiento de objetos que aparecen en una imagen, localizar estos objetos, determinar los atributos de los mismos y las relaciones entre ellos. *COCO* contiene una cantidad enorme de fotos que representan objetos comunes en situaciones complejas. Asimismo, cuenta con 2,5 millones de objetos etiquetados con 80 categorías en cada una de las 328.000 imágenes para describirlas. Sus principales características son:

- Segmentación de objetos: Cada objeto individual que aparece en la imagen es identificado y diferenciado del resto mediante *bounding boxes*³, lo cual permite a los modelos aprender la ubicación y la clase de objeto en una imagen, como se puede ver en la Figura 3.4.
- Descriptor de Imágenes: Esta función se encarga de mostrar unas etiquetas en formato *JSON* con la información referida a lo que contiene la imagen.
- Segmentación semántica: Este tipo de segmentación, se diferencia de la segmentación de objetos en cómo clasifica cada píxel de una imagen en una clase predefinida, recogiendo lo más importante. Es capaz dada una imagen de una ciudad, de clasificar las carreteras, edificios... Esto también se puede ver en la Figura 3.4
- Las imágenes de *COCO* se capturaron en entornos naturales. Esto es importante, ya que muchos de los sistemas de reconocimiento fallan al ser usados en estos entornos.

COCO utiliza un formato *JSON*, el cual es el encargado de otorgar información sobre cada *dataset* y todas sus imágenes. En la Figura 3.5 se puede ver cómo queda reflejada la información.

²<https://cocodataset.org/>

³Cuadro delimitador que contiene el objeto detectado

Figura 3.4: Ejemplo de segmentación en *COCO*

```
{
    "info": info,
    "images": [image],
    "annotations": [annotation],
    "licenses": [license],
}

info{
    "year": int,
    "version": str,
    "description": str,
    "contributor": str,
    "url": str,
    "date_created": datetime,
}

image{
    "id": int,
    "width": int,
    "height": int,
    "file_name": str,
    "license": int,
    "flickr_url": str,
    "coco_url": str,
    "date_captured": datetime,
}

license{
    "id": int,
    "name": str,
    "url": str,
}
```

Figura 3.5: Ejemplo de cómo se muestra la información de *COCO* mediante un JSON

COCO tiene como objetivo proporcionar una gran base de datos de imágenes realistas y variadas, con el fin de poder entrenar modelos de detección de objetos, los cuales pueden ser usados con distintos fines. Pero también, se puede usar para la generación de descripciones de imágenes.

3.1.3. OID

*OID*⁴ (*Open Images Dataset*), al igual que *COCO* es un *dataset* de imágenes para el entrenamiento de modelos en la detección de objetos en imágenes. Alberga alrededor de 9 millones de imágenes anotadas con etiquetas, objetos marcados con *bounding boxes*, máscaras para la segmentación de objetos y relaciones visuales.

⁴https://storage.googleapis.com/openimages/web/factsfigures_v7.html

Algunas de sus características son:

- Contiene un total de 16 millones de *bounding boxes* para 600 clases de objetos sobre 1,9 millones de imágenes. Esto lo convierte en uno de los *dataset* más extensos hasta la fecha. Las *bounding boxes* han sido dibujadas de manera manual. Las imágenes son muy diversas y suelen contener situaciones complejas con diferentes objetos.
- Las anotaciones también proporcionan relaciones visuales, indicando un par de objetos en relaciones particulares (p.e. “mujer tocando la guitarra”), propiedades de objetos (p.e. “mesa de madera”) y acciones humanas (p.e. “una mujer saltando”). En total contiene 3,3 millones de anotaciones de 1.466 tripletes de relaciones distintas.
- Contiene 2,8 millones de objetos en 350 clases, para la segmentación de objetos. Las máscaras para la segmentación, marcan el contorno del objeto aportando un mayor nivel de detalle.
- Proporciona descripciones multimodales de imágenes, sincronizando texto, voz y el trazado del ratón sobre los objetos que serán descritos.
- Cuenta con un alto número de etiquetas *point-level*. Estas son muy útiles para el entrenamiento y evaluación de la segmentación semántica.
- El *dataset* está anotado con 61,4 millones de etiquetas a nivel de imagen que abarcan 20.638 clases.

La Figura 3.6, es un ejemplo de una imagen de *OID*. La anotación para esta foto proporcionada por *OID* es la siguiente: “En el primer plano de la foto hay tarros, bebidas, salchichas y otras piezas de comida situadas en la mesa. En el centro de la imagen hay mesas, sillas y otros objetos. En el fondo hay edificios, árboles, coches, camino y otros objetos.”

3.1.4. Google Cloud Vision

*Google Cloud Vision*⁵ es una herramienta que permite analizar un gran número de imágenes, extrayendo una gran multitud de datos, los cuales pueden ser sobre detección de objetos, detección de rostros, reconocimiento de logotipos, análisis de contenido inapropiado, detección de etiquetas y categorización de las mismas con el fin de poder facilitar su interpretación. A continuación, explicamos más en detalle cómo funciona cada funcionalidad:

- Detección de rostros: Esta función permite identificar una o varias caras humanas en una imagen al analizar propiedades y atributos clave, lo que facilita la determinación del estado emocional de la persona. Al detectar una cara, proporciona un *bounding box* que delimita su ubicación en la imagen. Es importante destacar que esta detección no implica el reconocimiento facial individual

⁵<https://cloud.google.com/vision/docs/features-list?hl=es-419>

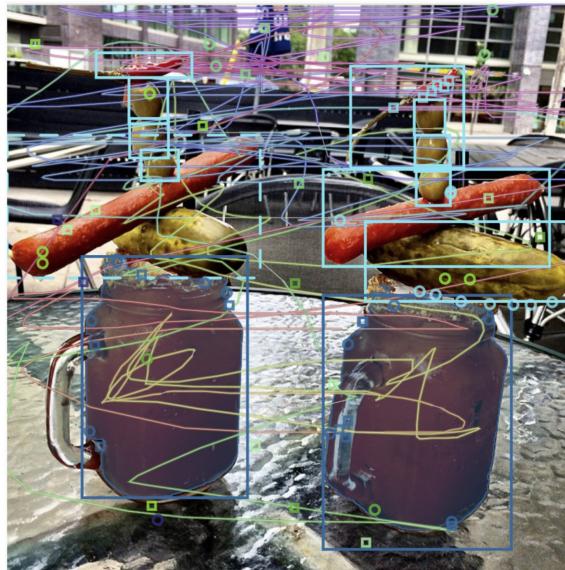


Figura 3.6: Ejemplo de imagen anotada con OID

de una persona específica. En su lugar, busca coincidencias entre la información biométrica del rostro detectado y los datos almacenados y etiquetados. Esta característica funciona de manera óptima cuando la cara está orientada frontalmente. Además, requiere una distancia mínima entre las pupilas de 32 píxeles, factor que resulta ser esencial en el proceso de detección.

- *Landmark detection*: Permite la detección de puntos de referencia. Detecta estructuras populares, como podrían ser monumentos, edificios emblemáticos o construcciones de la antigüedad. Además, tiene la capacidad de detectar las puntos de referencia en un archivo local (siempre que sea enviado como una *String* codificada en Base64), así como de archivos remotos que se encuentren en *Google Cloud Storage*⁶ (servicio de almacenamiento de nube) o en la web. La información que devuelve es la latitud y altitud del *landmark* detectado.
- Detección de etiquetas: Identifica y extrae información acerca de las entidades de una imagen. Asimismo, pueden identificar ubicaciones, acciones, animales, productos y demás. El usuario también puede crear etiquetas personalizadas de orientación, que le permitirá clasificar imágenes. Estas etiquetas sólo se muestran en inglés. La información devuelta está en formato *JSON* junto con el porcentaje de acierto que denota cuán preciso se fue al determinar dichas etiquetas.
- Detección de texto: Gracias a esta característica se puede detectar y extraer texto en imágenes mediante el uso del Reconocimiento Óptico de Caracteres. Es compatible con varios idiomas. Devuelve una cadena de texto al usuario, con sus coordenadas, palabras individuales y sus *bounding boxes*. Por otro lado, esta detección se puede llevar a cabo sobre documentos de texto. En el *JSON*

⁶<https://cloud.google.com/storage?hl=es-419>

de respuesta, se incluye los datos referentes a la página, la división, el bloque, párrafo y palabra.

- Detección de logos: Se encarga de detectar los logotipos populares que se encuentran en una imagen, local o remota.

3.1.5. RoboFlow

*RoboFlow*⁷ es una plataforma en línea que ofrece una amplia gama de herramientas para convertir imágenes en información útil, como descripciones de la imagen o la identificación de objetos presentes en ella. Todo ello en formato *JSON*. Además, destaca como una poderosa herramienta para la detección de objetos en imágenes. Sus numerosas funcionalidades, simplifican el procesamiento, organización y anotación de imágenes. Entre sus características principales se encuentran:

- Carga de datos: Cada usuario debe cargar sus propias imágenes o vídeos. También se pueden usar bases de datos como *COCO* con el fin de crear un modelo de visión por ordenador funcional de manera rápida.
- Anotación de datos: Una vez cargados los datos, se puede usar sus herramientas para etiquetar y marcar objetos de interés en los vídeos o imágenes.
- Formación de modelos: Una vez se han cargado los datos y se ha llevado a cabo la anotación, *RoboFlow* genera de manera automática los modelos de visión para la detección de objetos.
- Aumento de datos: Como se puede ver en la Figura 3.7 permite aumentar el *dataset*, con el fin de entrenar el modelo mediante la rotación de las imágenes, aplicación de filtros...
- Despliegue de modelos: Después de generar los modelos, esta herramienta facilita su despliegue de manera sencilla. Además, ofrece la posibilidad de utilizar estos modelos como puntos de conexión de API. El modelo generado es versátil y puede ser utilizado en diversas modalidades, tales como WebCam, tiempo real, *Hosted API* e *iOS* entre otras opciones.

En la Figura 3.8 se observa el resultado obtenido en la detección de objetos. Mediante *bounding boxes*, cada una con un color, identifica de manera precisa las diferentes monedas. Las monedas que son del mismo tipo aparecen con el mismo color.

3.1.6. Hugging Face

*Hugging Face*⁸ es un repositorio global que alberga una plataforma donde se desarrollan modelos de *deep learning*. Destaca principalmente por aquellos modelos dedicados a procesamiento de lenguaje natural, pero también incluye modelos de visión por computadora. Tanto usuarios, como empresas publican aquí sus modelos

⁷<https://roboflow.com/>

⁸<https://huggingface.co/docs/hub/index>

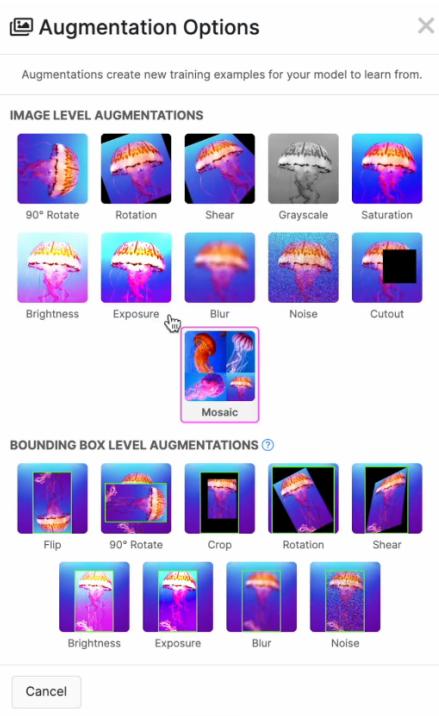


Figura 3.7: Ejemplo de operación de aumento de datos con Roboflow

de tecnología. Todos estos modelos se encuentran publicados en su página web, donde se puede filtrar por tipos de modelos. Hay una gran variedad de modelos a nuestra disposición: resúmenes de textos, clasificación de textos, generación de texto, traducciones... Cuenta con un gran número de *datasets*, superando los 5.000. A su vez, cada *dataset* cuenta con una amplia documentación que permite explorar datos directamente desde el navegador. En la Figura 3.9 se puede ver un ejemplo del uso del modelo *facebook/detrresnet-50* que permite la detección de objetos en imágenes. Cada animal está contenido en una *bounding box*. Además, indica el porcentaje de confianza sobre el mismo.

Para poder utilizar todos los modelos que incluye esta plataforma, *Hugging Face* ofrece una API a través de la cual se puede realizar llamadas a los distintos modelos, con el fin de poder usarlo de manera externa a la propia plataforma.

3.1.7. Clarifai

*Clarifai*⁹ es una destacada empresa en el campo de la inteligencia artificial. Se especializa en la visión artificial, procesamiento de lenguaje natural y reconocimiento de audio. La API *Predict* de *Clarifai* ofrece una potente herramienta que permite obtener un conjunto de categorías, como mostrar atributos en forma de etiquetas relacionados con la imagen. También permite sacar una descripción de lo que se puede ver de la propia imagen. Con un enfoque en el preprocesamiento de datos y el desarrollo de algoritmos, *Clarifai*, destaca en el procesamiento de información visual, extracción de etiquetas y análisis de imágenes. El largo recorrido que presenta

⁹<https://www.clarifai.com/>

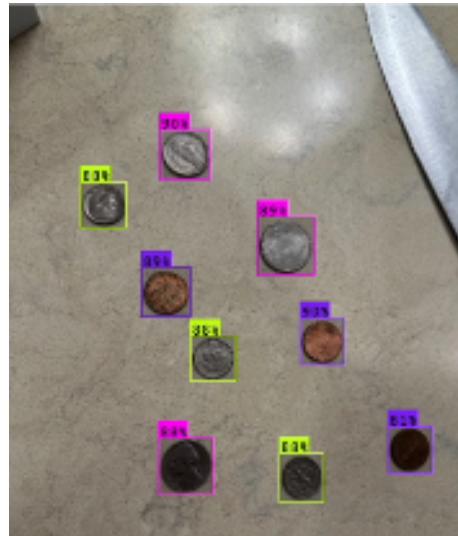


Figura 3.8: Ejemplo de detección de objetos con Roboflow

Clarifai en procesamiento de lenguaje natural y visión artificial, lo convierten en un recurso valioso para aplicaciones que requieren comprensión y análisis avanzado de contenido visual.

Clarifai proporciona un servicio en línea al que los desarrolladores o empresas pueden cargar sus imágenes. Utiliza su API para analizar las imágenes, y aprovecha herramientas de código abierto para el procesamiento de lenguaje natural en Python, con el objetivo de identificar las conexiones sintácticas entre palabras y obtener información valiosa.

Además, *Clarifai* cuenta con su propio *dataset* en el que puedes crear, explorar y modificar para entrenarlo según tus necesidades específicas, lo que brinda la flexibilidad de adaptar sus capacidades a tus propios propósitos. El motivo de que no utilicemos *Clarifai* proviene del limitado uso gratuito que proporciona, ya que las solicitudes que se pueden hacer son muy escasas para nuestros propósitos.

3.1.8. Microsoft Azure

*Microsoft Azure*¹⁰ es una plataforma que permite analizar imágenes y devuelve información detallada acerca de la imagen y los objetos que muestra. Cuenta con las siguientes funcionalidades:

- Descripción de imágenes: Como podemos observar en la Figura 3.10, permite la evaluación de los objetos detectados, generando una frase coherente que describe lo que se ha identificado en la imagen para el usuario. Puede devolver más de una respuesta en función del contenido de la imagen. Estas frases tienen asociadas una puntuación que varía en función de la confianza asociada a dicha descripción. Las descripciones se basan en un conjunto de miles de objetos reconocibles, que se usan para sugerir etiquetas para la imagen. Por ejemplo, si

¹⁰<https://learn.microsoft.com/es-es/training/modules/analyze-images-computer-vision/2-image-analysis-azure>

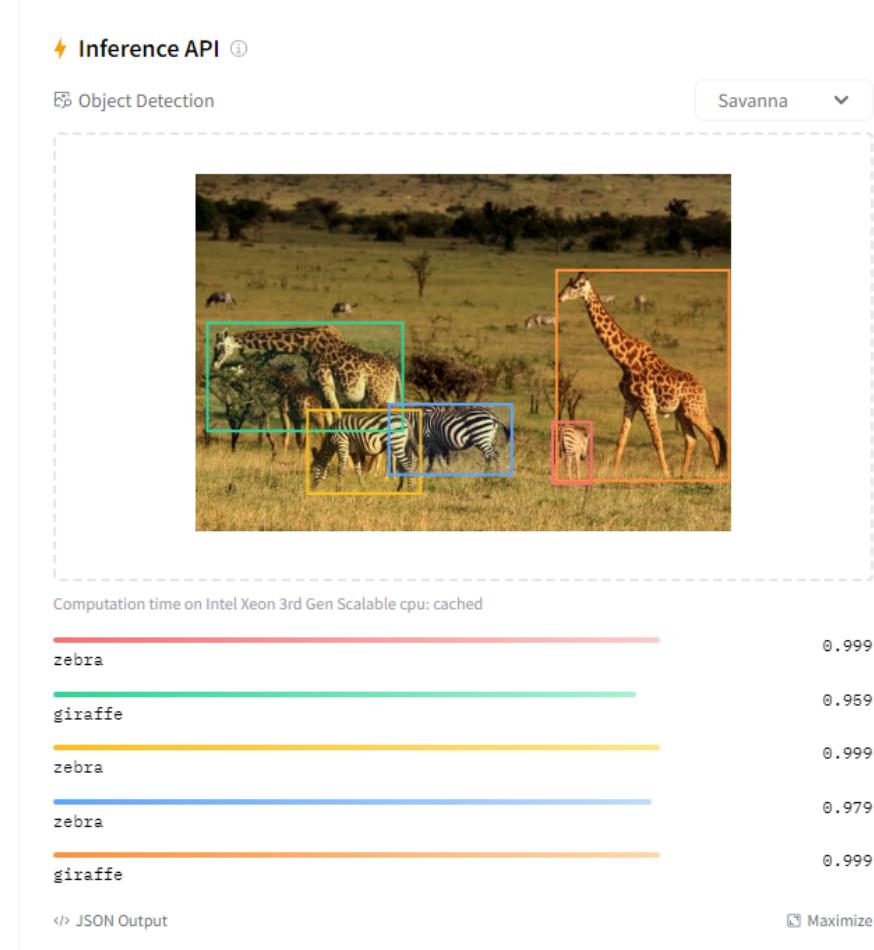


Figura 3.9: Ejemplo de detección de objetos en imágenes con el modelo `facebook/detrresnet-50-dc5` de *Hugging Face*

tuviéramos una imagen de dos niños jugando en un parque, nos saldrían como etiquetas: juego, diversión, parque, niño, felicidad. Estas etiquetas pueden ser útiles para indexar una imagen junto a un conjunto de términos clave que pueden ser usados para buscar imágenes con atributos específicos.

- Detección de objetos: Funciona similar al etiquetado. Como se muestra en la Figura 3.11 (a) puede identificar objetos comunes, pero en lugar de etiquetar o proporcionar etiquetas solo para objetos conocidos, es capaz de devolver lo que se conoce como coordenadas del *bounding box* que recibirá un conjunto de coordenadas que indicarán la ubicación del objeto.
- Detección de marcas comerciales: Esta funcionalidad es capaz de detectar si en la imagen aparece alguna marca reconocida globalmente. Las miles de marcas reconocidas de manera global se encuentran almacenadas en una base de datos. Además, también devuelve en *bounding box* el logotipo de la marca, así como una puntuación de confianza.
- Reconocimiento facial: La funcionalidad que proporciona la detección de caras es un subconjunto del servicio *Face Azure AI*, que incluye funcionalidades más



Figura 3.10: Una de las descripciones proporcionadas con Azure para esta imagen es “*Foto en blanco y negro de una ciudad*”

completas para el reconocimiento facial. Esta tecnología es capaz no sólo de reconocer rostros, si no también de determinar la edad. También tiene un *box* límite para la ubicación de las caras. En la Figura 3.11 (b) podemos ver esta funcionalidad.

- Categorización de imágenes: después de analizar una imagen es capaz de categorizar la misma, añadiendo un nombre a esta categoría. Por ejemplo, en la Figura 3.11(a) nos podría devolver como categoría urbano o ciudad. Esta categorización permite que se puedan detectar personajes famosos dentro de la foto, así como puntos de referencia populares. Actualmente, el conjunto de categorías posibles está limitado.
- Reconocimiento de texto: Es capaz de detectar caracteres en imágenes, tanto en formato impreso como escrito a mano.
- Identificar imágenes predefinidas, reconocer patrones de color al discernir tanto el fondo como los tonos generales de la imagen, y finalmente, supervisar contenido inapropiado.

3.1.9. Google Firebase

*Google Firebase*¹¹ es una plataforma de desarrollo de aplicaciones móviles y web. Proporciona una colección de servicios integrados y herramientas que ayudan a construir y mejorar las aplicaciones de manera rápida y eficiente. Los componentes principales de *Firebase* son los siguientes:

- Base de datos en tiempo real (*Realtime Database*): Es una base de datos NoSQL la cual permite el acceso seguro a la base de datos directamente desde el código del lado del cliente. Los datos persisten localmente e incluso sin conexión. Los eventos en tiempo real continúan activándose, cuando el dispositivo

¹¹<https://firebase.google.com/docs?hl=es>



Figura 3.11: Ejemplos de funcionalidades Microsoft Azure

recupera la conexión, *Realtime Database* sincroniza los cambios locales con las actualizaciones remotas que ocurrieron mientras el cliente estaba desconectado.

- Autenticación: Ofrece servicios *backend* para autenticar a los usuarios en su aplicación, admitiendo la autenticación a través de contraseñas, números de teléfono y proveedores de identidades federados populares como Google, Facebook y Twitter, entre otros.
- Almacenamiento (*Storage*): Almacena los archivos en un depósito Google Cloud Storage, lo que permite almacenar archivos de forma eficiente y hacerlos accesibles tanto a través de *Firebase* como de la infraestructura de Google Cloud. Esta integración proporciona una flexibilidad excepcional, ya que podemos cargar y descargar archivos desde nuestras aplicaciones móviles utilizando los SDK de *Firebase* para almacenamiento en la nube.

Además, esta combinación nos permite realizar operaciones de procesamiento del lado del servidor, como filtrado de imágenes o transcodificación de videos, aprovechando las API de Google Cloud Storage. Esto significa que podemos manipular archivos directamente desde nuestros clientes móviles, pero también realizar tareas más pesadas y complejas en el servidor, manteniendo así una experiencia fluida y eficiente para nuestros usuarios. Es una solución muy completa y poderosa para gestionar archivos en nuestras aplicaciones.

- *Hosting*: Servicio de alojamiento web que permite a los desarrolladores implementar y hospedar sus sitios web estáticos y aplicaciones web en una infraestructura segura y escalable
- *Cloud Functions*: Es un *framework* sin servidor que permite la ejecución automatizada de código *backend* en respuesta a eventos desencadenados por funciones de *Firebase* o solicitudes *HTTPS*. El código desarrollado en JavaScript

o *TypeScript* se almacena en la infraestructura de nube de *Google* y se ejecuta en un entorno administrado. Esta solución elimina la necesidad de administrar o escalar servidores propios, ya que *Firebase* se encarga de gestionar la infraestructura subyacente de forma transparente.

- *Analytics*: Un sistema de análisis que ofrece un análisis exhaustivo del desempeño y la conducta de las aplicaciones, abarcando datos relevantes sobre usuarios, eventos y conversiones.
- *Cloud Messaging*: servicio de mensajería en la nube que permite a los desarrolladores enviar mensajes a dispositivos móviles y web para mantener a los usuarios comprometidos y actualizarlos con información relevante.

3.1.10. ColorThief

*Color Thief*¹² es una biblioteca de JavaScript que permite a los desarrolladores extraer la paleta de colores de una imagen. Esta herramienta es útil para obtener el color dominante de una imagen o una paleta de colores completa. La biblioteca es ligera y simple, y puede funcionar tanto en Node.js como en el navegador sin dependencias externas. Para obtener el color dominante de una imagen, la API proporciona el método `getColor()`, que devuelve el color como una matriz de tres enteros que representan los valores de rojo, verde y azul (RGB).

Además, *Color Thief* también proporciona el método `getPalette()` para obtener una paleta de colores de la imagen. Este método agrupa colores similares y devuelve la paleta como una matriz que contiene colores, donde cada color es una matriz de tres enteros.

3.1.11. Palette

La biblioteca *Palette*¹³ es una herramienta proporcionada por *Jetpack* (conjunto de herramientas, bibliotecas y guías proporcionadas por *Android*) que se utiliza para extraer colores destacados de las imágenes en una aplicación. Estos colores extraídos pueden ser utilizados para mejorar el diseño visual de la aplicación, permitiendo la creación de temas personalizados y la aplicación de colores específicos a diferentes elementos visuales.

Al utilizar la biblioteca *Palette*, los desarrolladores pueden generar paletas de colores a partir de imágenes, lo que les permite acceder al color principal de una imagen y a otros perfiles de color. Esto es útil para adaptar dinámicamente el esquema de colores de la aplicación según el contenido visual que se está mostrando.

Además, la biblioteca de *Palette* proporciona métodos para personalizar las paletas generadas, como la especificación del número máximo de colores o el área de la imagen a utilizar para la extracción de colores.

¹²<https://lokeshdhakar.com/projects/color-thief/>

¹³<https://developer.android.com/reference/androidx/palette/graphics/package-summary>

3.2. Trabajo de Fin de Grado Anterior

En esta sección se hará una revisión del Trabajo de Fin de Grado “*Desarrollo de una aplicación móvil para la generación de descripciones de imágenes para personas con discapacidad visual*” (Amor Sanz et al., 2023). También se explicarán los puntos principales y necesarios para entender el proyecto como son el objetivo, el diseño y las conclusiones finales.

El objetivo principal era el desarrollo de una aplicación capaz de generar descripciones de imágenes. Como objetivos más específicos en primer lugar, se priorizaba el enfoque centrado en el usuario, lo cual implicaba la realización de un estudio exhaustivo con individuos con discapacidad visual con el fin de identificar y comprender a profundidad sus necesidades. Este estudio consistió en la realización de entrevistas detalladas, destinadas a indagar sobre el modo en que dichos usuarios interactuaban con las aplicaciones existentes, así como para explorar sus preocupaciones y desafíos específicos, particularmente en lo que respecta a la obtención de información relevante para ellos.

En segundo lugar, se llevó a cabo un análisis riguroso para adquirir un conocimiento profundo de las técnicas de inteligencia artificial pertinentes que permitirían la implementación de las funcionalidades deseadas. Este proceso de estudio y evaluación fue crucial para garantizar que la aplicación resultante cumpliera con los estándares de accesibilidad y usabilidad requeridos para satisfacer las necesidades de las personas con discapacidades visuales. Finalmente, hubo una evaluación con usuarios para identificar posibles mejoras.

3.2.1. Arquitectura

En el proyecto se usó una arquitectura cliente-servidor, cuya labor fue repartir las tareas para una buena gestión de recursos.

El cliente, en este caso una aplicación *Android*, selecciona la imagen y esta es enviada al servidor. El cliente recibe el *JSON* procesado por el servidor y se muestra al usuario en forma de audio. La interfaz de la aplicación móvil incluye dos botones, uno para seleccionar una imagen y otro para obtener la descripción de la misma. Estos se situaron en las esquinas inferiores ya que son las posiciones más sencillas de ubicar. El lector por voz usado es el servicio *TextToSpeech*, incluido en *Android Studio*. Este permite ajustar la voz, el idioma y la velocidad de lectura. La conexión con el servidor se realiza mediante peticiones *HTTP* de tipo *POST*.

El servidor recibe la imagen desde el cliente y se procesa internamente para generar el *JSON*. En este caso, se usó un servidor escrito en *Python* llamado *Flask*¹⁴. Este servidor actúa como intermediario entre el cliente y las distintas API, necesarias para las funcionalidades de la aplicación.

Para la funcionalidad de descripción de imagen se usó la plataforma *Hugging Face* (sección 3.1.6). Esta plataforma genera un texto descriptivo de una imagen asociada. En concreto, se eligió el modelo *SalesForce/blip-image-captioning-large*. Desde este modelo pre-entrenado se obtiene un *JSON* compuesto por una clave-valor, donde la

¹⁴<https://flask.palletsprojects.com/es/latest/>

clave es *generated_text* y la descripción.

Para la funcionalidad de detección de entidades, inicialmente se usó la plataforma *Roboflow* (sección 3.1.5), ya que esta proporciona mucha información sobre la implementación. Esta plataforma genera un *JSON* con una serie de datos sobre los objetos detectados (clase, coordenadas). Sin embargo, el modelo probado solo detectaba perros, gatos y patos. El segundo enfoque fue usando de nuevo *Hugging Face*, en concreto el modelo *facebook/detr-resnet-101*. Este modelo detecta todo tipo de objetos mostrando el nivel de precisión y las coordenadas del objeto. Finalmente, debido a que estos modelos producen texto en inglés, se usó el modelo *Helsinki-NLP/opusmt-en-es* como traductor del inglés al español.

3.2.2. Diseño centrado en el usuario

Como se ha mencionado anteriormente, el desarrollo del proyecto se realizó con un diseño centrado en el usuario. Esta filosofía de diseño tiene como objetivo conocer las necesidades concretas directamente desde los usuarios finales. Para corregir errores y añadir posibles mejoras se realizan evaluaciones en cada iteración del desarrollo hasta conseguir la versión final.

Hubo una entrevista con personas con discapacidad visual para conocer los requisitos iniciales. En esta entrevista se sacaron las siguientes conclusiones. La aplicación debe ser simple para que su uso no sea complejo y requiera ayuda externa. El sistema operativo mejor preparado en cuanto a accesibilidad es iOS¹⁵. El problema de las aplicaciones existentes son las descripciones poco detalladas. Incluir un lector de pantalla por voz para narrar las descripciones y para poder navegar por los botones de la aplicación.

Las mismas personas fueron entrevistadas para la evaluación final de la aplicación. Esta entrevista, tenía como fase previa el uso de la aplicación por parte de cada usuario, seguida de una serie de preguntas de usabilidad usando el cuestionario SUS¹⁶. A raíz de esta evaluación, se obtuvieron las siguientes conclusiones:

- Puntos positivos: Se cumplió el objetivo de conseguir una aplicación sencilla que no requiera de ayuda externa para su uso, ya que se puede ver tanto en los vídeos donde las personas con discapacidad la utilizan, como en las conclusiones que sacaron. Estas personas utilizaban la aplicación por ellos mismos sin requerir de nadie ni de otra aplicación secundaria. La navegación por la imagen es de gran utilidad. Las descripciones aportan suficiente información.
- Puntos negativos: El fondo de las imágenes es difícil de identificar y por tanto hay poca información del entorno. No se diferencia entre personas y especies de animales. El usuario no puede identificar bien los límites de la imagen. El tiempo de carga de las imágenes es muy elevado. Problemas de compatibilidad con las funciones de accesibilidad del sistema operativo.

¹⁵Sistema operativo móvil de código cerrado desarrollado por Apple Inc.

¹⁶System Usability Scale

3.2.3. Conclusiones y trabajo futuro

En este apartado están las conclusiones obtenidas al finalizar el proyecto, entre las cuales está que consiguieron que las personas con discapacidad visual pudieran tener una representación en el espacio de como estaban situados los distintos objetos de la imagen, además de tener una primera toma de contacto con la descripción auditiva que salía al pulsar uno de los botones. Por otro lado, se consiguió que fuera una aplicación fácil de usar para ellos. Este apartado es interesante, ya que ofrece ideas para posibles mejoras, Además de conocer el rumbo a seguir para futuros proyectos.

Algunos de estos aspectos a mejorar son los siguientes:

- Desarrollar la aplicación para *iOS*, ya que es el sistema que más utilizan personas con discapacidad visual.
- Mejorar las descripciones, incluyendo detalles como diferenciación de personas u objetos.
- Mejorar tiempo de respuesta del servidor.
- Ajustar tamaño de los *bounding boxes* para que sean más precisos con el tamaño real de los objetos de la imagen.
- Añadir nuevas funcionalidades, como poder describir el fondo de las imágenes y dar más contexto sobre estas.
- Añadir ajustes en la aplicación para cambiar idioma o cambiar la velocidad de reproducción de la descripción.
- Controlar los límites de la imagen a la hora de navegar por la imagen.

Capítulo 4

Funcionamiento de la aplicación

Este capítulo contiene una explicación en detalle y desde el punto de vista del usuario de las diferentes funcionalidades de las que se dispone, cómo activar estas funcionalidades y la información final que aportan al usuario.

Al iniciar la aplicación, se presupone que el usuario tiene activado el sistema de accesibilidad del dispositivo. El sistema de accesibilidad, *Talkback* en *Android* y *VoiceOver* en *iOS* es necesario, ya que gracias a este el usuario puede usar las opciones de selección de imagen navegando por su dispositivo en el entorno al que está acostumbrado. Este sistema consiste en la interacción entre la persona con discapacidad visual y su dispositivo. Al tenerlo activado, la primera acción que se produce es decir de forma auditiva la pantalla en la que se encuentra. Una vez el usuario pulsa sobre los botones que se encuentran en dicha pantalla, se seleccionará el botón y dirá cual es. Despues se necesita de un doble toque sobre el mismo para que entre en acción la funcionalidad del botón pulsado. Además este sistema también da información auditiva sobre notificaciones.

4.1. Selección de imagen

Una vez iniciada la aplicación el usuario puede recorrer los botones de la interfaz los cuales incluyen una descripción. Como se puede ver en la Figura 4.1 la aplicación cuenta con dos botones. El primero de ellos, situado en la esquina inferior izquierda de la pantalla, es el botón con la descripción “*Seleccionar imagen*”, que como su propio nombre indica permite al usuario cargar una imagen en pantalla para su posterior interpretación. Una vez pulsado el botón aparece un cuadro con las dos opciones para cargar la imagen (Figura 4.2). Finalmente y para acabar de cargar la imagen para cada una de las dos opciones el dispositivo cambia al entorno correspondiente y como se puede ver en la Figura 4.3 se accede a la aplicación de cámara o a la aplicación de galería propias del dispositivo. El segundo botón está ubicado en la esquina inferior derecha, tiene como descripción “*Obtener descripción*”, y permite al usuario reproducir la descripción general siempre que se quiera.

Un vez cargada la imagen, esta se muestra por pantalla. Las imágenes pueden tener dos tipos de orientación, vertical u horizontal. Este aspecto es importante ya que buscamos fijar la imagen con el mayor tamaño posible para facilitar la navegación

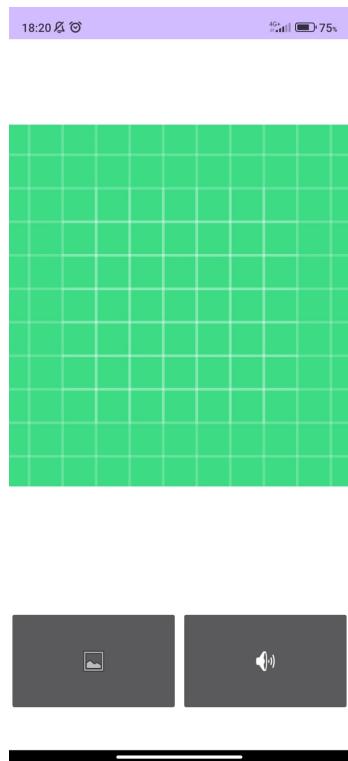


Figura 4.1: Inicio de la aplicación

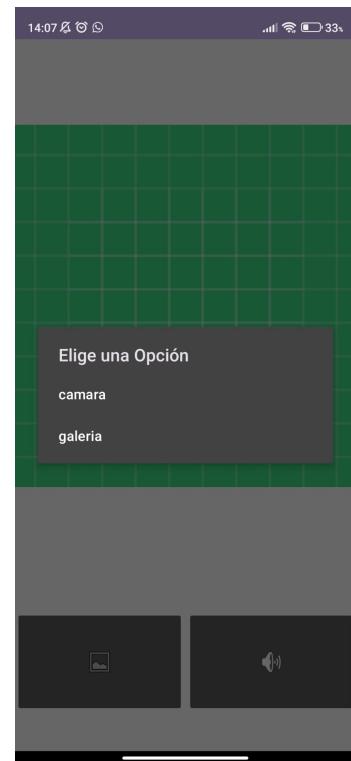


Figura 4.2: Selección de imagen

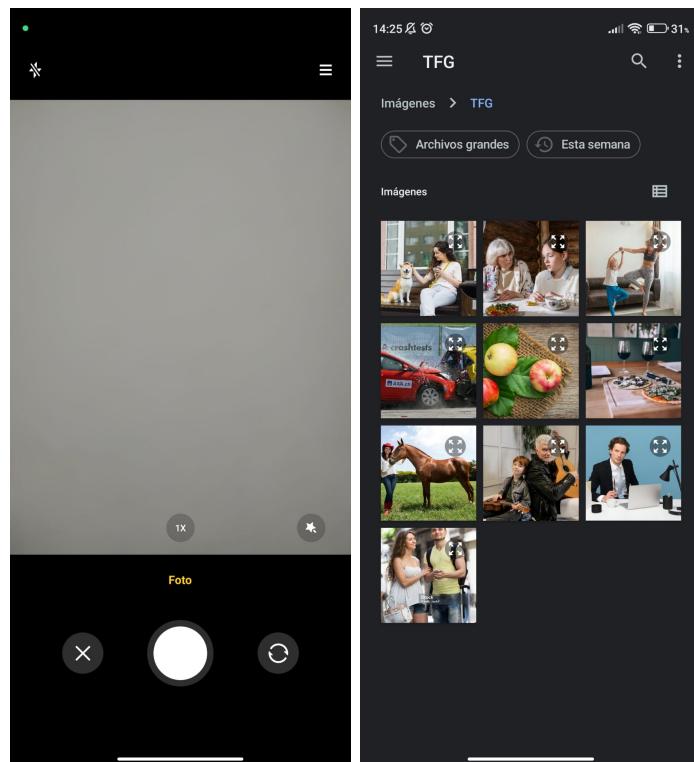


Figura 4.3: Opciones para seleccionar imagen

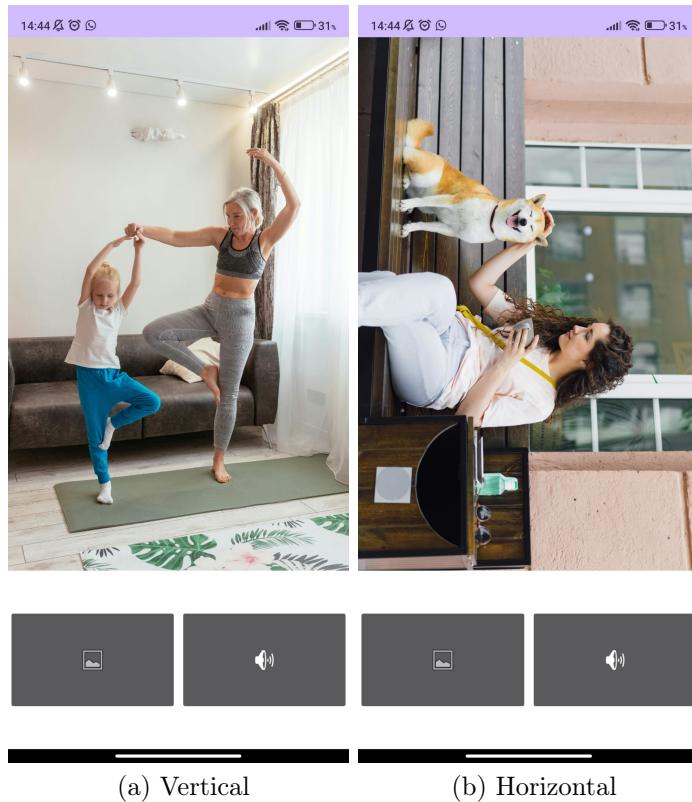


Figura 4.4: Display de la imagen según su orientación

y poder llegar a todos detalles de la imagen, como pueden ser los objetos pequeños. Como podemos ver en la Figura 4.4 dependiendo de la orientación, la imagen se muestra rotada o no. Debido a esta rotación se debe informar al usuario cuando corresponda, por tanto, con el lector por voz se reproduce la frase “*La imagen está en horizontal, gire el dispositivo hacia la izquierda.*”.

4.2. Descripción General de la imagen

Una vez cargada la imagen y habiendo informado al usuario de la rotación comienza la fase de descripción. Al inicio de esta fase se procesa la imagen. Esta acción tiene un retardo y por tanto se informa al usuario una vez más, reproduciendo la frase “*Obteniendo descripción*”. Una vez procesada la imagen se comienza con una descripción cualitativa y general de la imagen. En la Figura 4.5 podemos ver la imagen con su respectiva descripción que es reproducida al usuario.

A partir de este momento el usuario tiene disponible la descripción cualitativa para volver a escucharla en cualquier momento. Si desea escucharla, simplemente deberá pulsar el segundo botón de la interfaz, situado en la esquina inferior derecha de la pantalla.



Figura 4.5: Imagen para la que se genera la descripción "*mujer sentada en un banco con un perro y un teléfono celular*"

4.3. Navegación por la imagen

Finalizada la fase de descripción continuamos con la fase de navegación. En este fase el usuario podrá pulsar en la pantalla obteniendo una amplia variedad de información.

4.3.1. Descripción Cuantitativa

Bajo la premisa de que el usuario debe estar informado en todo momento de como se muestra la imagen, debemos analizar las posibilidades para evitar confusiones. Esto se debe a que las proporciones de la imagen pueden ser distintas a las del dispositivo, por lo que pueden quedar zonas en la pantalla sin imagen. En la Figura 4.5 podemos ver como al cargar en la aplicación la imagen, esta se muestra con espacios en blanco a ambos lados. En esta situación si el usuario pulsa en las zonas rayadas en rojo de la Figura 4.6, se reproduce la frase "*Estás fuera de la imagen*" que permitirá al usuario conocer los límites de la imagen.

Una vez el usuario pulse por primera vez dentro de los límites de la imagen obtiene una descripción cuantitativa. Esta descripción enumera todos los objetos detectados para evitar que el usuario deje posteriormente objetos sin pulsar. En la Figura 4.7 podemos ver una imagen con su respectiva descripción cuantitativa.

4.3.2. Objetos

En la fase de navegación se lleva a cabo sobre los objetos detectados pero esta vez de forma individual. Cada vez que el usuario pulse sobre la imagen se reproduce el nombre del objeto que se encuentra en ese punto. Estos objetos cuentan con un sufijo numérico, en caso que exista más de un objeto con el mismo nombre, para así facilitar su diferenciación, como se puede ver en la Figura 4.8.

Dado que los objetos pueden no ocupar la totalidad de la imagen, en muchos puntos de la imagen no se encuentra ningún objeto. En caso de que el usuario pulse

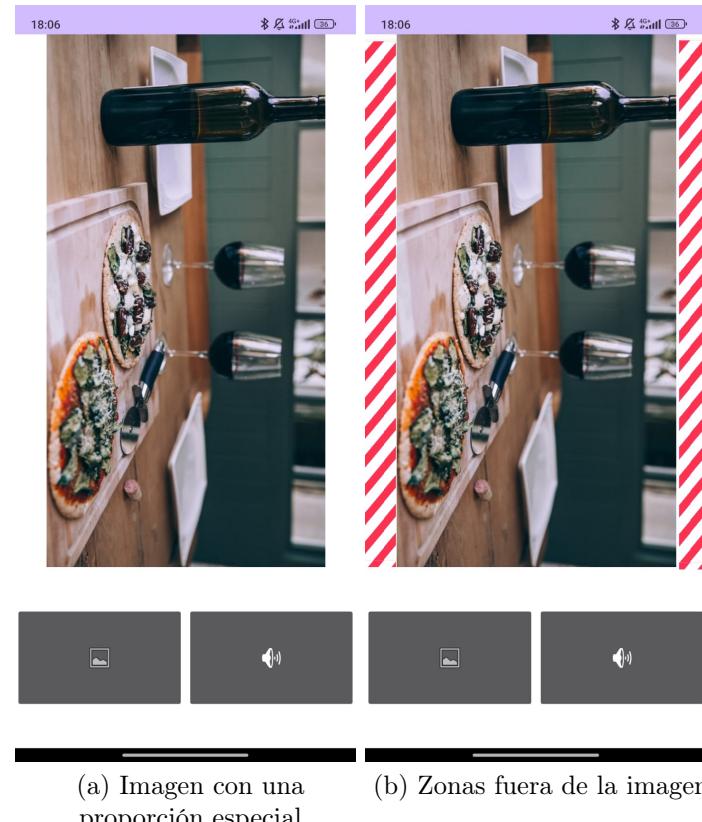


Figura 4.6: Imagen con espacios en blanco



Figura 4.7: Se han obtenido: dos personas y un sofá



Figura 4.8: Objetos: Coche, Coche 1

sobre un punto sin objeto, como puede ser el fondo de la imagen, se reproduce la frase “*No hay ningún objeto.*”.

En el otro lado del espectro, en un mismo punto de la imagen puede haber más de un objeto. Por ello, en el caso de pulsar en un punto en el que se encuentre más de un objeto se produce una lista con los distintos objetos con sus respectivas informaciones adicionales.

Los objetos tienen información adicional, la cual contiene las cualidades específicas del objeto más allá de su nombre. Para esta información los objetos se diferencian entre objetos persona y el resto de objetos.

- Persona:

Si el usuario pulsa sobre una persona en la imagen, se reproduce “*Persona*”. Adicionalmente para las personas se reproduce “*Pulsa de nuevo para obtener más información*” (este texto se reproducirá siempre que pulse sobre una persona hasta que se haya cargado la información adicional de la persona, pudiéndose dar el caso de que no se reproduzca este texto si carga rápidamente). Si el usuario pulsa de nuevo se obtiene la información adicional, que en este caso es el género y la edad de la persona. Para el género se diferencia entre hombre y mujer, sin embargo si después del procesamiento de la imagen el género no está definido se omite esta diferenciación. Para la edad establecemos tres niveles: joven, de mediana edad o de avanzada edad. Una vez más si esta característica no está clara se establece la edad como no definida. En la Figura 4.9 podemos ver algunos ejemplos de esta funcionalidad.

- Resto de objetos:

Si el usuario pulsa sobre un objeto que no es persona se reproduce el nombre del objeto. Para estos objetos la información adicional será el color.



Figura 4.9: Información adicional persona



Figura 4.10: Información adicional objeto

A continuación en la Figura 4.10 podemos ver algunos ejemplos de esta funcionalidad:

Otra de las funcionalidades dentro de la fase de navegación es la descripción detallada. Esta descripción, al igual que la descripción inicial, se trata de una descripción cualitativa, salvo que solo se aplica al objeto deseado. Para acceder a esta descripción se debe mantener pulsado sobre un objeto y se reproduce dicha descripción. En la Figura 4.11 podemos ver la descripción detallada de una de las personas en la imagen.

Finalmente si el usuario desea cargar otra imagen o finalizar el uso de la aplicación, hay que reactivar el sistema de accesibilidad para volver al uso normal de su dispositivo.



Figura 4.11: una mujer inclinada sobre una mesa de billar con un taco en la mano

Capítulo 5

Implementación del Sistema

En este capítulo vamos a hablar sobre la estructura del cliente y del servidor que usa la aplicación. Así como de distintos aspectos relevantes como la modularidad de la aplicación, la gestión de colores y coordenadas y por último de las limitaciones que hemos encontrado durante la implementación de las distintas características de nuestra aplicación.

5.1. Prototipo tecnológico

Con la intención de explorar las distintas tecnologías que se podían usar en el proyecto, comenzamos realizando un prototipo tecnológico para explorar e integrar las más adecuadas. Para este prototipo se estudiaron distintas posibilidades para iniciar el desarrollo de la aplicación.

- **Flask:** se desarrolló una aplicación básica, con el objetivo de hacer pruebas utilizando un servidor *Flask*. Esta opción quedó descartada tras intentar enviar una imagen desde nuestra aplicación al servidor sin éxito.
- **Google Firebase:** también se investigó la tecnología de *Google Firebase* y su compatibilidad con *Android Studio*. Se descubrió que esta opción ofrecía una usabilidad más directa y sencilla, además de integrarse fácilmente con otras herramientas de *Google* ya incorporadas en *Android Studio*.

Se tomó la decisión de desarrollar la aplicación usando *Google Firebase*. Sin embargo, la inteligencia artificial propia de *Firebase* no cumplía con nuestros objetivos de manera satisfactoria. Debido a esta limitación, se barajaron otras posibilidades, como implementar nuestra lógica utilizando *Cloud Functions*. Esto se hizo enviando la imagen a *Firebase Storage* y accediendo a ella desde *Cloud Functions*, pero usando esta estrategia tampoco obtuvimos los resultados deseados.

Finalmente, se solucionó este problema enviando las imágenes desde la aplicación a las *Cloud Functions* codificadas en Base64. Desde estas funciones, se hacen llamadas a diversas APIs de *Hugging Face*, y estas APIs son las que contienen los distintos modelos a través de los cuales obtenemos información sobre la imagen.

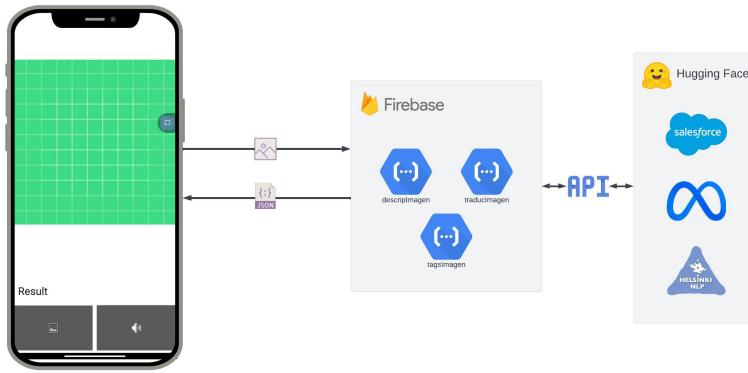


Figura 5.1: Diagrama de las API

5.2. Arquitectura

La aplicación está implementada siguiendo un modelo cliente servidor. En este entorno, el flujo de datos se inicia con la transmisión de la imagen desde el dispositivo móvil al servidor remoto.

Una vez que la imagen llega al servidor *Firebase*, se desencadena un conjunto de operaciones automatizadas que comprenden la interacción con diversas APIs. Estas APIs son invocadas por el servidor desde las *Cloud Functions* para procesar la la imagen recibida. Es fundamental destacar que este proceso de análisis no se realiza internamente en el servidor, sino que se delega a las APIs externas, que ejecutan las tareas de reconocimiento y comprensión de contenido visual. Esta externalización del procesamiento garantiza un rendimiento óptimo y una precisión en la interpretación de los elementos presentes en la imagen.

Una vez que las APIs han llevado a cabo el procesamiento de la imagen y han extraído la información relevante, los resultados se estructuran en formato *JSON*. Este archivo *JSON* contiene los datos procesados.

Finalmente, este archivo *JSON* es enviado de vuelta al dispositivo móvil del usuario, completando así el ciclo de interacción entre el cliente y el servidor. El dispositivo móvil recibe entonces los datos procesados y los presenta de manera sencilla y comprensible al usuario.

La Figura 5.1 proporciona una representación visual de este proceso, mostrando la arquitectura general del sistema, desde la transmisión inicial de la imagen hasta la recepción y presentación de los datos procesados en el dispositivo móvil.

5.3. Servidor

Firebase es una plataforma de desarrollo de aplicaciones respaldada por Google. Su principal función es facilitar la creación de aplicaciones web y móviles de alta calidad de manera rápida. *Firebase* ofrece una variedad de herramientas y servicios que simplifican las tareas de desarrollo y gestión, entre ellas, las *Cloud Functions*, que veremos en más detalle.

El principal motivo por el que nos decantamos para usar *Firebase* fue la facilidad

con la que se puede conectar con una aplicación móvil. Para realizar dicha conexión con nuestra aplicación, únicamente hay que añadir los SDKs necesarios a las dependencias de la aplicación y de esta forma tendremos acceso total a las funcionalidades de *Firebase* de una forma muy sencilla.

5.3.1. Cloud Functions

Las *Cloud Functions* son un *framework* que permite ejecutar automáticamente código en respuesta a eventos activados por funciones de *Firebase* y solicitudes HTTPS. En este servicio hemos creado funciones que reciben las imágenes codificadas en Base64 por el cliente y las envían a las APIs de *Hugging Face*. Una vez llamadas, devuelven la información requerida, como puede ser una descripción de la imagen o un listado de los objetos detectados, entre otros. Estas funciones están programadas en JavaScript y se encuentran en un archivo dentro de la carpeta del proyecto. Para desplegarlas y que *Firebase* tenga acceso a las mismas, almacenándolas en la nube, hay que usar la consola del sistema.

Sobre este *framework* hemos implementado la siguiente lógica. El cliente manda una imagen a las *Cloud Functions* codificada en Base64. Una vez tengamos la codificación en Base64 se lleva a cabo la decodificación de la misma para poder enviarlas a las APIs que contienen los modelos. La codificación en Base64 hace que el envío de la imagen desde el cliente al servidor sea sencilla y efectiva. Por otro lado, con el fin de que el cliente no gestione los posibles errores devueltos por las APIs, siempre que estas devuelva un error se vuelve a enviar la imagen a las APIs. En el siguiente código vemos la función de entrada entre el cliente y el servidor:

```
1 exports.tagsImagen = onCall((data, response) => {
2     console.log("Función de tags conectada");
3     const base64 = data.url;
4     const buffer = Buffer.from(base64, 'base64');
5
6     function recursive(buffer) {
7         return tagging(buffer).then((response) => {
8             const jsonResponse = JSON.stringify(response);
9             if (response.error) {
10                 return recursive(buffer);
11             } else {
12                 return jsonResponse;
13             }
14         });
15     }
16     return recursive(buffer);
17 });
```

Una vez decodificada la imagen, procedemos a realizar la consulta a las APIs del modelo correspondiente. La consulta se realiza a través del *Inference API*, servicio

gratuito y *serverless*. El código de la consulta a los modelos se incluye en los repositorios de los modelos. La consulta se realiza mediante una petición *HTTPS request* e incluye un token único de autenticación:

```

1  async function tagging(data) {
2      const response = await fetch(
3          "https://api-inference.huggingface.co/models/facebook/detr-resnet-50",
4          {
5              headers: { Authorization: "Bearer *****" },
6              method: "POST",
7              body: data,
8          }
9      );
10     const result = await response.json();
11     return result;
12 }
```

Como se explicará a continuación, también usamos modelos para la traducción de texto. Al igual que con las imágenes, el cliente envía en este caso una cadena *String* que contiene el texto deseado, y posteriormente se envía esta cadena a la API correspondiente para su traducción:

```

1  exports.traducDescrip = onCall((data, response) => {
2      console.log("Función de traducción conectada");
3      return query({ "inputs": data.texto }).then((response) => {
4          return JSON.stringify(response);
5      });
6  });
7
8  async function query(data) {
9      const response = await fetch(
10         "https://api-inference.huggingface.co/models/Helsinki-NLP/opus-mt-en-es",
11         {
12             headers: { Authorization: "Bearer *****" },
13             method: "POST",
14             body: JSON.stringify(data),
15         }
16     );
17     const result = await response.json();
18     return result;
19 }
```

Finalmente y después de obtener el *JSON* de respuesta del modelo este se transforma en un *String* para el flujo de salida de las Cloud Functions y se manda al cliente para que realice las operaciones pertinentes.



Figura 5.2: Imagen para explicar los modelos

5.3.2. Modelos

En cuanto a los modelos usados podemos diferenciar entre modelos para la obtención de información de las imágenes y modelos para el tratamiento del texto. Para el uso de todos estos modelos utilizamos la plataforma *Hugging Face* y para una mejor explicación usaremos como ejemplo de la Figura 5.2:

- Descripción de imágenes: Para llevar a cabo esta tarea elegimos el modelo *SalesForce/blip-image-captioning-large*¹(Li et al., 2022) ya que es uno de los más potentes de la actualidad, superando a otros modelos como *ALBEF* o *SimVLM*². Este modelo es capaz de generar descripciones detalladas de las imágenes basándose en su contenido visual. El modelo recibe como entrada una imagen, la procesa y genera una respuesta. Esta respuesta contiene la descripción en inglés y se representa mediante un *JSONArray*, aunque únicamente cuenta con una clave-valor. La salida de este modelo para la Figura 5.2 sería:

```

1   [
2     {"generated_text": "bride and groom kissing in a field
3       ↵ at sunset"}
  ]

```

- Detección de objetos: En cuanto a la detección de objetos usamos el modelo *facebook/detr-resnet-50-dc5*³(Carion et al., 2020). Al igual que el modelo anterior, este es uno de los que mejores resultados han obtenido⁴. Este modelo es capaz de detectar y localizar objetos en una imagen. Proporciona información

¹<https://huggingface.co/Salesforce/blip-image-captioning-large>

²<https://arxiv.org/pdf/2201.12086.pdf>

³<https://huggingface.co/facebook/detr-resnet-50-dc5>

⁴<https://arxiv.org/pdf/2005.12872.pdf>

detallada sobre los objetos presentes en la imagen. Al igual que el modelo anterior recibe una imagen y proporciona un *JSONArray*, con toda la lista de los objetos detectados, como respuesta. Cada objeto detectado viene representado como un *JSONObject* dentro del *JSONArray*. Este *JSON* contiene el tipo de objeto, la *bounding box* que contiene al objeto y la precisión. Para la Figura 5.2 obtendríamos la siguiente salida:

```

1   [
2     {
3       "score": "0.998",
4       "label": "person",
5       "box": {
6         "xmin": "375",
7         "xmax": "59",
8         "ymin": "535",
9         "ymax": "714"
10      }
11    },
12    {
13      "score": "0.982",
14      "label": "person",
15      "box": {
16        "xmin": "448",
17        "xmax": "98",
18        "ymin": "1037",
19        "ymax": "788"
20      }
21    }
22 ]

```

- Información acerca de la persona: Para poder obtener más información acerca de las personas que aparezcan en la foto hemos hecho uso de los siguientes modelos: *dima806/man_woman_face_image_detection*⁵(Iakubovskyi, 2023a) y *dima806/faces_age_detection*⁶(Iakubovskyi, 2023b). El primer modelo permite a partir de la imagen de una persona diferenciar entre *woman* y *man*, mientras que el segundo modelo nos da información acerca de su edad, diferenciando entre *old*, *middle* y *young*. Ambos modelos reciben una imagen de la persona y devuelven un *JSONArray* que contiene las diferentes opciones junto la fiabilidad de cada una, esta fiabilidad se tiene en cuenta para establecer un umbral de acierto. Estas serían la salidas para los diferentes modelos respectivamente.

```

1   [

```

⁵https://huggingface.co/dima806/man_woman_face_image_detection

⁶https://huggingface.co/dima806/faces_age_detection

```

2      {
3          "label": "woman",
4          "score": 0.9633112549781799
5      },
6      {
7          "label": "man",
8          "score": 0.036688774824142456
9      }
10 ]

```

```

1  [
2  {
3      "label": "OLD",
4      "score": 0.34316298365592957
5  },
6  {
7      "label": "YOUNG",
8      "score": 0.3313744366168976
9  },
10 {
11     "label": "MIDDLE",
12     "score": 0.3254626393318176
13 }
14 ]

```

- Traducción de texto: Dado que los modelos anteriores producen texto en inglés y nuestra aplicación está en español, hacemos uso del modelo *Helsinki-NLP/opusmt-en-es*⁷(Tiedemann, 2020) para traducir las salidas de los modelos anteriores, es decir el campo del *JSON* deseado, del inglés al español. Este modelo es un sistema de traducción automática neuronal que puede traducir texto de inglés a español manteniendo la semántica y el contexto del texto original. El modelo requiere un texto como entrada, es decir un objeto *String*, en nuestro caso a este modelo lo utilizamos para traducir, o bien el campo *generated_text* del modelo de descripción de imágenes, o para el campo *label* del modelo de detección de objetos. Al igual que los otros modelos este también devuelve un *JSONArray*, que cuenta con una clave valor que representa el texto traducido.

```

1  [
2      {"translation_text": "novia y novio bes\u00e1ndose en un
3          \u2192 campo al atardecer"}
4  ]

```

⁷<https://huggingface.co/Helsinki-NLP/opus-mt-en-es>



Figura 5.3: *Layout* de la aplicación

5.4. Cliente

El cliente ha sido desarrollado para el sistema operativo *Android*. Si bien es cierto que los usuarios coinciden en que la accesibilidad es mejor en dispositivos iOS, desarrollar para este sistema operativo supone unos costes que no son fácilmente abordables en un Trabajo de Fin de Grado.

El proyecto consta de una única pantalla (*MainActivity*), cuyo diseño se realizó teniendo en cuenta los datos extraídos en el Trabajo de Fin de Grado anterior (Amor Sanz et al., 2023). Como se puede ver en la Figura 5.3 la interfaz es muy simple, aspecto muy importante para los usuarios.

El lenguaje usado para programar la aplicación es Java y en la Figura 5.4 podemos ver el diagrama UML del proyecto, que representa la estructura estática de nuestra aplicación, mostrando las clases que componen el sistema, así como sus atributos, métodos y las relaciones entre ellas. Esto proporciona una vista general de la organización y funcionamiento del sistema, lo que facilita la comprensión de su diseño y arquitectura. A continuación haremos una enumeración de los distintos paquetes con sus respectivas clases que componen la aplicación además del *MainActivity* que es la clase principal, la cual se encarga de gestionar las distintas clases que componen el programa para el correcto funcionamiento de la aplicación. Esta estructuración se ha realizado para poder tener una mayor legibilidad y coherencia a la hora de programar. Los paquetes son: hilo, imagen, server.

- Hilo:

- Hilo extiende de Thread: Recibe la imagen codificada en Base64 y una instancia de la clase *FireFunctions*. Inicializa con estos valores las variables *Firebase* e imagen para que las clases que extienden Hilo puedan usarlas.

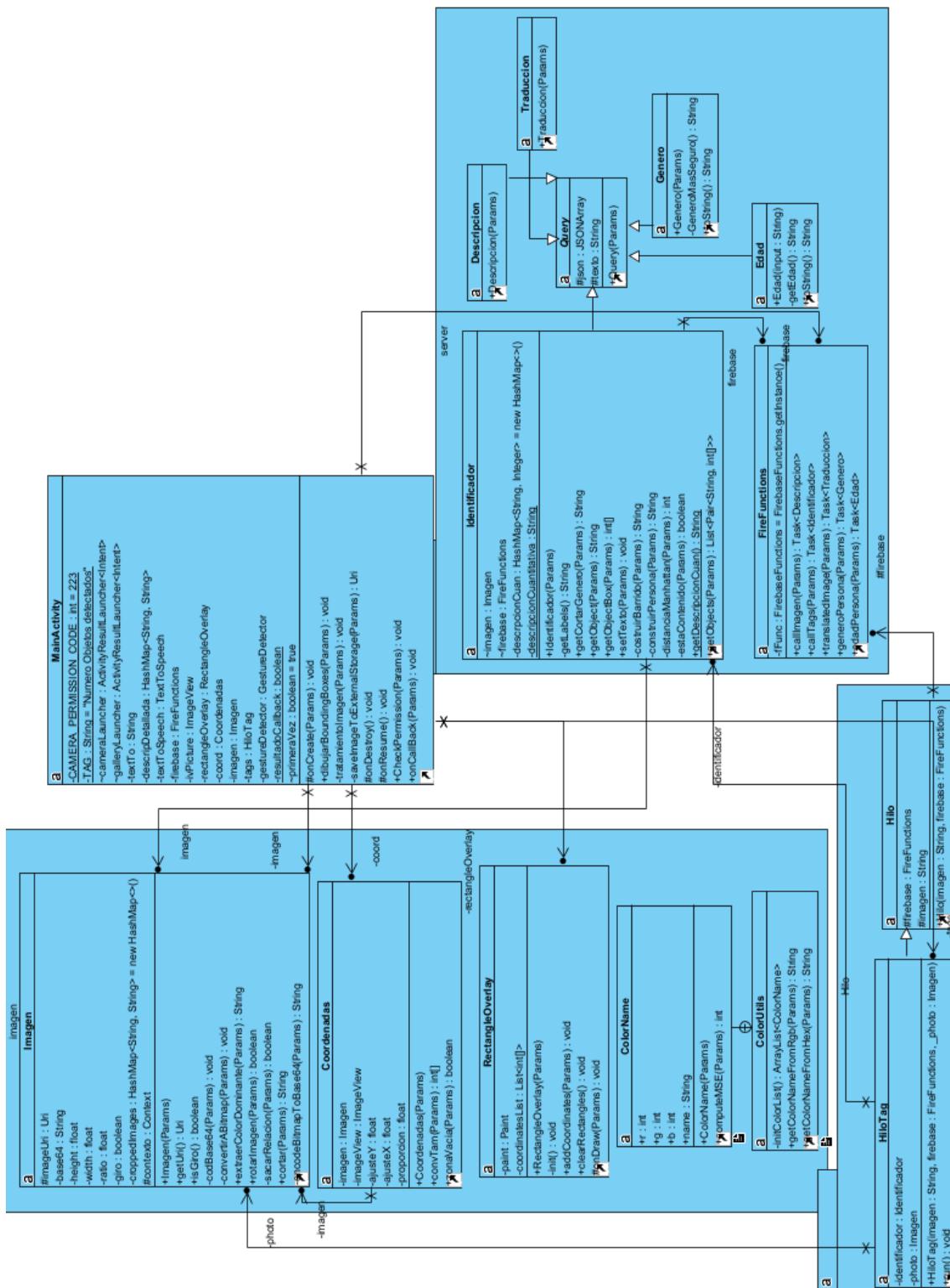


Figura 5.4: Diagrama UML

- HiloTag extiende Hilo: Recibe, al igual que la clase Hilo, la imagen en Base64 y una instancia de la clase *FireFunctions*, que pasa al super, es decir, a la clase Hilo, para su inicialización, así como una instancia de la clase Imagen. Por otro lado también se encarga de invocar el método *callTags* de la clase *FireFunctions* que se encarga de llamar a la *Cloud Function* encargada de detectar los distintos objetos de la imagen. También invoca al método *transaltedImage* de la clase *FireFunctions*, para la traducción de los objetos.

- Imagen:

- Coordenadas: se encarga de calcular y ajustar las coordenadas de una imagen dentro de un *ImageView*. Al ser inicializada con una instancia de *ImageView* y una instancia de *Imagen*, la clase calcula el ajuste necesario para mostrar la imagen correctamente dentro del *ImageView*, considerando la relación de aspecto de la imagen y la orientación de la misma. El método *convTam* convierte las coordenadas de la imagen original a las coordenadas del *ImageView*, teniendo en cuenta cualquier rotación que se haya aplicado a la imagen. Además, el método *zonaVacia* verifica si una posición dada dentro del *ImageView* corresponde a un área vacía sin contenido de la imagen. En resumen, la clase *Coordenadas* facilita el manejo de las coordenadas y el ajuste de imágenes dentro de un *ImageView*. Para más detalle sobre la clase *Coordenadas* ver sección 5.6 .
- Imagen: proporciona funcionalidades para manipular imágenes, incluyendo la conversión de una imagen a Base64, rotación de imágenes, y recorte de imágenes. Al ser inicializada con un contexto y una *URI* de imagen, la clase extrae la representación Base64 de la imagen y calcula su relación de aspecto. La función *rotarImagen* rota la imagen en 90 grados si es necesario, mientras que *sacarRelacion* determina si la imagen debe mostrarse en orientación vertical u horizontal.
- RectangleOverlay: extiende de *View*, define una vista personalizada que permite mostrar rectángulos superpuestos en una imagen o en otra vista. Al ser inicializada, configura un objeto *Paint* para dibujar los rectángulos con un borde de color rojo. En la Figura 5.5 se puede ver un ejemplo de uso. Destacar que esta clase ha sido programada con el fin de ayudarnos en el desarrollo de la aplicación, y no proporciona ninguna funcionalidad extra al usuario.
- Colores: la biblioteca *Palette* utiliza algoritmos de análisis de imágenes para identificar el color predominante en una imagen. Una vez obtenido el color resultante está en hexadecimal. La clase *ColorUtils.java* convierte el resultado a texto.

- Server:

- Query: proporciona una estructura base para procesar datos *JSON* en una aplicación *Android*. Al ser inicializada con una cadena de entrada, la clase convierte esta cadena en un objeto *JSONArray*, que luego se puede

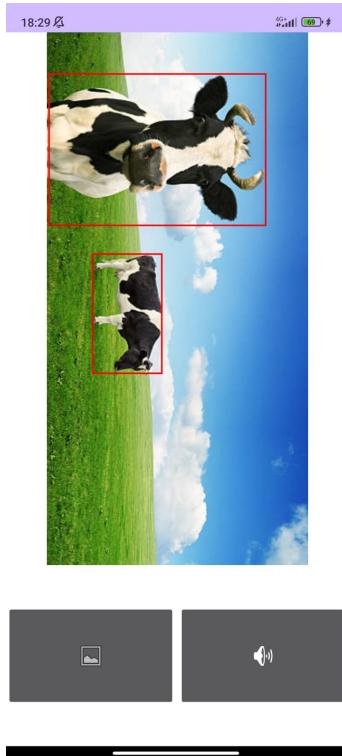


Figura 5.5: Dibujo Bounding Box

utilizar para acceder y manipular datos *JSON*. La clase contiene métodos para obtener el texto procesado y el objeto *JSONArray* subyacente. Esta abstracción facilita la implementación de clases especializadas que procesan datos *JSON* de manera específica para diferentes propósitos dentro de la aplicación. En resumen, la clase *Query* sirve como una base genérica para el procesamiento de datos *JSON* en una aplicación *Android*, permitiendo la creación de clases especializadas que extienden su funcionalidad según sea necesario.

- Descripción: extiende la clase *Query* y se utiliza para procesar una entrada de texto y extraer información específica de un objeto *JSON*. Al ser inicializada con una cadena de entrada, la clase utiliza la clase base *Query* para procesar el texto y extraer datos de un objeto *JSON*. Luego, obtiene el texto generado del primer elemento del objeto *JSON* y lo almacena en una variable llamada *texto*. Esta clase proporciona una forma conveniente de obtener descripciones generadas a partir de un objeto *JSON* en una aplicación.
- FireFunctions: administra las llamadas a las funciones de *Firebase* en una aplicación *Android*. Esta clase utiliza *Firebase Functions* para ejecutar operaciones en el servidor. Al recibir una *URI*, los métodos *callImagen* y *callTags* solicitan descripciones e identificadores asociados con una imagen respectivamente, mientras que *translatedImage* y *getColor* gestionan la traducción de texto y la obtención de colores mediante llamadas a las funciones correspondientes en el servidor. La clase procesa las respuestas

de las llamadas y devuelve instancias de clases específicas que manejan los datos obtenidos, proporcionando así una interfaz simplificada para interactuar con *Firebase Functions* en la aplicación *Android*.

- Identificador: extiende la funcionalidad de la clase base *Query* para obtener etiquetas asociadas con la imagen. Procesa datos *JSON* para identificar objetos dentro de una imagen. Mediante el método *getObject*, se identifican objetos contenidos en coordenadas específicas dentro de la imagen, mientras que *getObjectBox* devuelve las coordenadas del *bounding box* del objeto, siempre y cuando se haya pulsado sobre un objeto. El método *setTexto* asigna etiquetas únicas a los objetos detectados, en el caso de los sufijos sobre personas, se delega en funciones secundarias, que realizan la misma tarea, para así evitar duplicados, además es aquí donde se llama a las funciones encargadas de obtener información acerca del género y la edad de las personas. Utilizando métodos internos como *distanciaManhattan* y *estaContenido*, la clase evalúa si un punto está contenido dentro de la caja del objeto, lo que permite una identificación precisa de objetos dentro de la imagen. También, es aquí donde se construye la descripción cuantitativa usando el método *construirBarrido*. En resumen, la clase *Identificador* facilita la identificación y manipulación de objetos en una imagen a través de datos *JSON* en una aplicación *Android*.
- Traducción: extiende la clase *Query* y se utiliza para procesar datos *JSON* relacionados con traducciones en una aplicación *Android*. Al ser inicializada con una cadena de entrada, la clase utiliza la clase base *Query* para convertir la cadena en un objeto *JSONArray*. Luego, extrae el texto de traducción del primer elemento del objeto *JSON* y lo almacena en una variable llamada *texto*. Esta clase proporciona una forma de procesar y obtener texto traducido a partir de datos *JSON* en la aplicación.
- Género: extiende la funcionalidad de la clase *Query* para obtener el género de las distintas personas de la imagen. A través del método *getGenero* se selecciona el objeto *JSON* con mayor fiabilidad, es decir si es hombre o mujer y devuelve dicha cadena para poder construir la etiqueta del objeto.
- Edad: Esta clase es prácticamente idéntica a la clase *Genero*, salvo que en vez de evaluar entre hombre y mujer, se evalúa entre los tres posibles rangos de edad joven, mediana edad, avanzada edad, devolviendo, al igual que antes, una cadena que indica la edad y que se añade a la etiqueta del objeto.

MainActivity: Esta clase gestiona la interacción del usuario con la aplicación, incluida la captura y selección de imágenes, el procesamiento de datos *JSON* relacionados con las imágenes, la detección de objetos en imágenes, la traducción de texto, y la síntesis de voz. Al inicializar la actividad, se establecen los elementos de interfaz de usuario y se configuran los escuchadores de eventos para responder a las acciones del usuario, como la selección de imágenes de la cámara o la galería. La clase utiliza

varios hilos de ejecución para realizar operaciones asíncronas, como la detección de etiquetas de objetos en imágenes. Además, integra funcionalidades como dibujar re cuadros alrededor de objetos detectados en imágenes y proporcionar descripciones verbales de los objetos al usuario a través de síntesis de voz. En resumen, *MainActivity* actúa como el punto central de control para la funcionalidad principal de la aplicación, permitiendo la interacción del usuario con las características de procesamiento de imágenes y texto implementadas en la aplicación. Maneja el ciclo de vida de la actividad, asegurando la liberación adecuada de recursos y la gestión de permisos necesarios para el funcionamiento de la aplicación, como los permisos de la cámara.

5.5. Modularidad

La modularidad es un aspecto fundamental para el desarrollo de software, ya que permite obtener proyectos más escalables, flexibles y fáciles de entender para futuras colaboraciones. Añadir nuevas funciones a nuestra aplicación es una tarea sencilla gracias a la división y organización del código en módulos que realizan tareas independientes.

Para añadir nuevas funciones se deberán seguir los siguientes pasos:

1. Analizar si la funcionalidad requiere cambios en la interfaz de la aplicación. Si es necesario se debe modificar el archivo *activity_main.xml* y los correspondientes cambios en *MainActivity.java*.
2. Si se requiere utilizar tecnologías externas, la comunicación entre el cliente y las APIs se debe realizar desde las *Cloud Functions*. Para ello se pueden usar el formato de las funciones en */server/functions/index.js*. Es por este motivo que también nos decantamos por usar un servidor, ya que si queremos que nuestra aplicación use otros modelos solo será necesario modificar el archivo anteriormente mencionado. Desde la aplicación se debe modificar la clase *FireFunctions.java* añadiendo una función usando el formato de las funciones existentes como guía.
3. Las funciones en *FireFunctions.java* construyen objetos *Query.java* por ello para encapsular operaciones a realizar con la respuesta del servidor se debe extender la clase *Query.java* donde se inicializará el *JSON* con la respuesta del servidor.
4. Finalmente existen dos clases auxiliares, la clase *Coordenadas.java*, la cual se usará para redimensionar los datos obtenidos del servidor. La clase *RectangleOverlay.java* permite realizar trazos sobre la imagen para la fase de desarrollo.

Todas los aspectos de implementación tanto en el cliente como en el servidor en este apartado están explicadas en detalle en la sección 5.4

5.6. Gestión de coordenadas

Para el cálculo de coordenadas hay que tener en cuenta tres aspectos: la orientación de la imagen, la redimensión de la imagen y el encuadre en la pantalla. En el código, las coordenadas donde el usuario pulsa se toman en relación al *ImageView* de la Figura 5.3 y el origen está situado en la esquina superior izquierda.

En la Figura 5.6 podemos ver los distintos datos que serán necesarios para cubrir todos los aspectos mencionados previamente y gracias a la Tabla 5.1 podemos ver como se usarán estos datos para cubrir todas las posibilidades y poder gestionar de manera correcta las coordenadas. La orientación de la imagen es un atributo booleano de la clase *Imagen* que indica si la imagen se muestra rotada 90 grados a la derecha. La orientación es importante para las coordenadas ya que el eje Y de la imagen pasa a ser el eje X y viceversa.

Para la redimensión el cálculo más importante será la proporción o ratio tanto de la imagen como del *ImageView*. El ratio permitirá saber si la imagen sobre la que se ajusta la imagen *ImageView* y partir de ello calcular el factor de proporcionalidad necesario para reajustar la imagen. La proporción se calcula de la siguiente forma:

$$\text{Ratio} = \frac{\text{height}}{\text{width}}$$

Debido a que la imagen siempre se muestra al máximo tamaño posible el encuadre de la imagen provoca espacios en blanco. Estos espacios son *ajusteX* y *ajusteY* en la Figura 5.6 con los cuales podemos calcular el desplazamiento necesario en las coordenadas. La función *zonaVacia* nos permite saber si el usuario está pulsando fuera de la imagen.

```

1  public boolean zonaVacia(float x, float y){
2      return x<ajusteY || y<ajusteX || x>imageView.getWidth()-ajusteY
3          || y>imageView.getHeight()-ajusteX;
}
```

5.7. Gestión de los colores

Algo que puede resultar realmente útil a la hora de describir una imagen es detectar el color de las entidades u objetos que estamos situando en la misma. Esto aporta información valiosa, ya sea saber el color de una prenda de ropa, o simplemente el color de un coche que hay aparcado en una calle. Para la identificación del color de las entidades, se barajaron principalmente dos opciones, *Color Thief* y la biblioteca *Palette*.

Tras realizar pruebas de integración con ambas opciones se decidió usar la biblioteca *Palette* por los siguientes motivos:

1. Integración directa con la aplicación cliente: *Palette* es una biblioteca que se integra directamente en el código de la aplicación cliente. Esto significa

Interfaz	Proporción	Ajustes
	$\frac{ivHeight}{Height}$	$ajusteX = 0$ $ajusteY = \frac{ivWidth - Width * proporcion}{2}$
	$\frac{ivWidth}{Width}$	$ajusteX = \frac{ivHeight - Height * proporcion}{2}$ $ajusteY = 0$
	$\frac{ivHeight}{Width}$	$ajusteX = 0$ $ajusteY = \frac{ivHeight - Width * proporcion}{2}$
	$\frac{ivWidth}{Height}$	$ajusteX = \frac{ivWidth - Height * proporcion}{2}$ $ajusteY = 0$

Tabla 5.1: Posibles Encuadres

que puedes acceder a ella sin necesidad de comunicarte con servicios en la

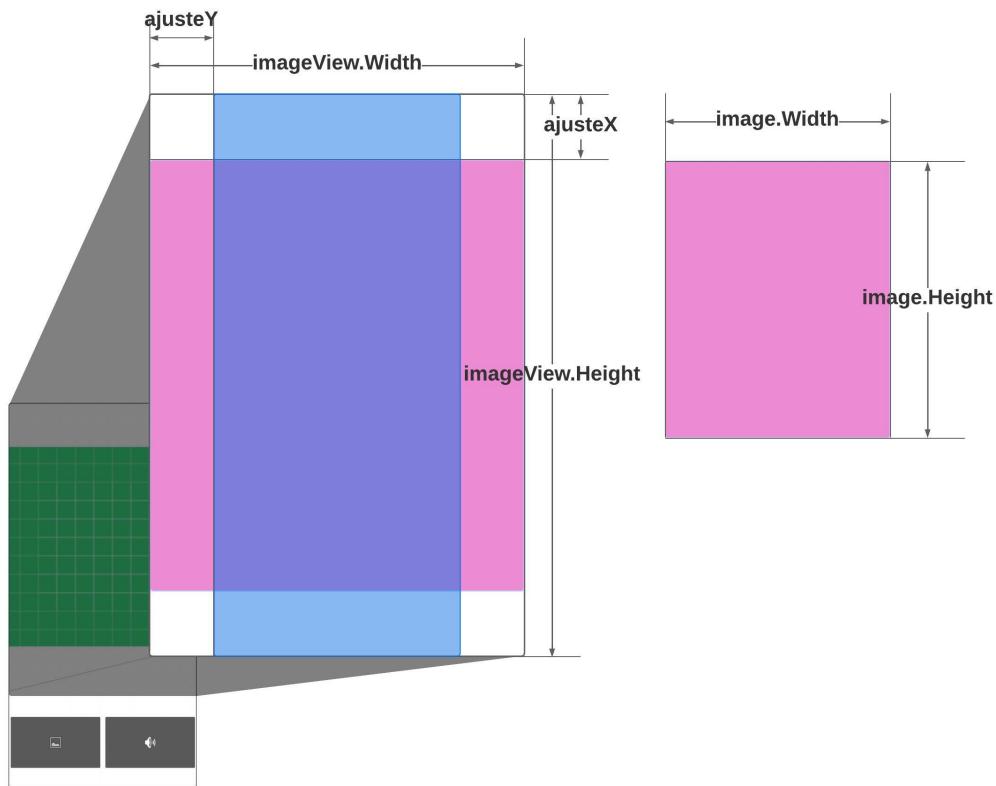


Figura 5.6: Representación de los valores necesarios para los cálculos de coordenadas

nube o funciones externas. Al utilizar *Palette* en el cliente, puedes procesar las imágenes localmente y obtener los colores dominantes de manera inmediata, sin depender de llamadas adicionales a *Cloud Functions*.

2. Personalización y control: Al trabajar con *Palette*, tienes un mayor control sobre cómo se extraen y utilizan los colores. Puedes ajustar los parámetros según las necesidades específicas de tu aplicación.
3. Optimización de la experiencia del usuario: Al procesar los colores en el cliente, puedes mejorar la velocidad de respuesta para los usuarios. No es necesario esperar a que se completen las llamadas a servicios en la nube. Esto contribuye a una experiencia más fluida y rápida para los usuarios de tu aplicación.

La biblioteca *Palette* devuelve los colores en hexadecimal. Es por ello que se ha utilizado una clase llamada *ColorUtils.java*⁸, la cual hemos modificado para hacerla más precisa. Esta clase “traduce” el resultado buscando el color más próximo a los colores definidos en la librería *color* de java. Una vez traducido, el usuario podrá conocer el color del objeto en cuestión, junto al *tag* del objeto al hacer un toque sobre el *bounding box*.

⁸<https://gist.github.com/XiaoxiaoLi/8031146#file-colorutils-java>

5.8. Limitaciones actuales de la aplicación

Durante el desarrollo surgen gran variedad de ideas de las cuales muchas no se pueden llegar a completar debido a ciertas limitaciones técnicas.

5.8.1. Talkback

El lector de pantalla *TalkBack* de *Google* para dispositivos *Android*, junto con su equivalente en *iOS*, *VoiceOver*, son herramientas de accesibilidad diseñadas para proporcionar retroalimentación auditiva sobre acciones, notificaciones y contenido en la pantalla, especialmente dirigidas a personas con discapacidades visuales.

La funcionalidad de *TalkBack*, ampliamente integrada en dispositivos móviles como teléfonos inteligentes y tabletas, así como en software de computadora, emplea síntesis de voz para verbalizar el contenido visual presente en la pantalla, incluyendo menús, iconos, botones y texto. Esta capacidad facilita la navegación del dispositivo, la interacción con aplicaciones y el acceso a la información de manera independiente para los usuarios con discapacidad visual.

De manera precisa, la operación de *TalkBack* consiste en que al pulsar una vez sobre un elemento en la pantalla, se anuncia de forma audible qué botón se ha activado, y si se vuelve a pulsar sobre el mismo elemento, se ejecuta la acción estándar asociada a dicho botón sin tener activada la función de *TalkBack*.

En el contexto de nuestra aplicación se proyectaba implementar una personalización en el funcionamiento de *TalkBack*. Inicialmente, al activarse *TalkBack*, se indicaría la presencia de dos botones, “cámara” y “galería”, los cuales podrían ser utilizados de manera convencional. Sin embargo, la modificación significativa se produciría una vez que la imagen estuviera cargada en el dispositivo. En este momento, se requeriría desactivar *TalkBack* para permitir que la persona con discapacidad visual pueda explorar y detectar los diferentes objetos presentes en la imagen, ya que el *TalkBack* solo permite realizar ciertas acciones, anulando otras, por ejemplo la acción de hacer un solo toque en la pantalla no se puede registrar si el *TalkBack* está activado. Por lo tanto es necesaria la desactivación de *TalkBack*, debido a que su funcionamiento básico no permite la exploración de imágenes, limitándose a identificar el área ocupada por la imagen, lo cual no satisface el objetivo principal de nuestra aplicación, que es describir imágenes de manera efectiva.

El problema principal surge cuando se intenta modificar esta función de accesibilidad, ya que está profundamente integrada en el sistema operativo *Android*. Cualquier intento de alteración podría desencadenar fallos de seguridad, dado que dicha función realiza cambios en el núcleo del sistema operativo, lo que podría ser aprovechado para realizar acciones maliciosas. Por tanto, resulta inviable implementar una modificación de *TalkBack* para permitir su desactivación al cargar una imagen, con el fin de facilitar la interacción de personas con discapacidad visual con la pantalla táctil.

5.8.2. Descripción del fondo de las imágenes

Uno de los aspectos o funcionalidades exploradas pero no implementadas ha sido la identificación y descripción del fondo de la imagen. Con el objetivo de mejorar la interpretación de la imagen podemos dividir cada imagen en dos partes. Por un lado tenemos el fondo de la imagen y por otro los objetos que aparecen sobre este. Los modelos de detección existentes actualmente no reconocen al fondo de la imagen como objeto y por ello se pierde información valiosa para la interpretación de esta.

De la misma forma que hay una relación entre el fondo y los objetos, también se puede definir una relación entre objetos. Esta relación se definiría como la posición relativa entre objetos para obtener información del tipo: delante, detrás, cerca, lejos, etc. Para abordar este problema una de las soluciones que se plantearon fue la profundidad de la imagen. La profundidad se refiere a la distancia entre un objeto o píxel de la imagen y el punto de vista de la cámara.

La profundidad la podemos conseguir a partir del modelo *LiheYoung/depth-anything-small-hf*⁹ (Yang et al., 2024). Como se puede ver en la Figura 5.7 con este modelo obtenemos un mapa de calor donde el negro representa el punto más lejano, el blanco el punto más cercano y toda una gama de grises que representan las distancias proporcionales. Una vez tenemos esta información junto con los *bounding boxes* que ya conocemos, podemos obtener las profundidades de los distintos objetos. Dado que cada tono de gris representa se puede representar mediante un número del 0 al 255, podemos establecer la profundidad de un objeto como la media de todos los píxeles del *bounding box*. Podemos ver la transformación en la Figura 5.8.

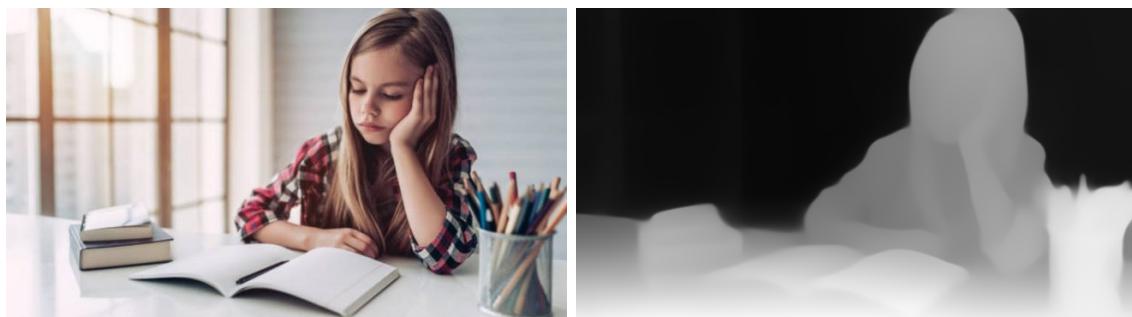
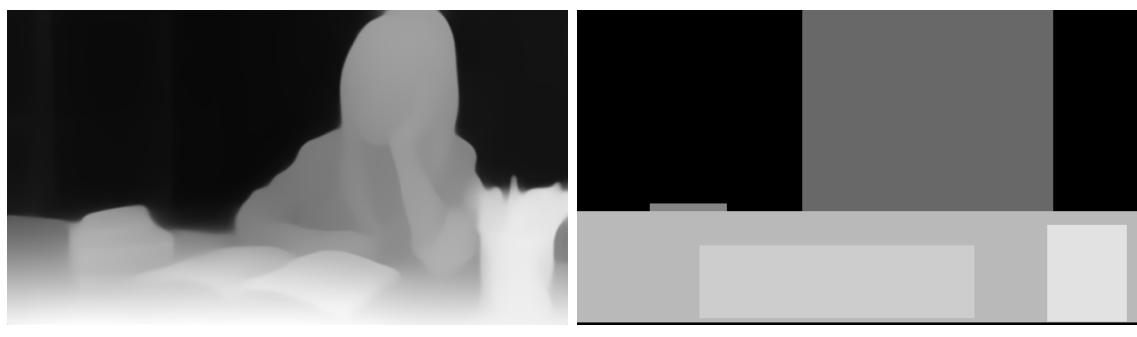


Figura 5.7: Modelo de profundidad

Esta funcionalidad no se implementó en la aplicación final debido a que para el modelo de profundidad *Hugging Face* no dispone de un *endpoint* que pudiera ser integrado rápidamente en la arquitectura de la aplicación.

⁹<https://huggingface.co/LiheYoung/depth-anything-small-hf>



(a) Mapa de calor

(b) Profundidad *bounding boxes*

Figura 5.8: Transformación de profundidad

Capítulo 6

Evaluación

Generalmente, los usuarios son necesarios en las últimas fases del desarrollo. La evaluación con el usuario permite comprobar que se han satisfecho las expectativas de los desarrolladores así como las necesidades del usuario. Con esta evaluación pretendemos evaluar la usabilidad, experiencia del usuario y comprobar que la funcionalidad cumple con su objetivo. Por otro lado permite identificar posibles errores y mejoras que habrían sido más difíciles de descubrir sin comprobar cómo los usuarios usan la aplicación.

6.1. Participantes

Los participantes de nuestra evaluación fueron dos personas con discapacidad visual. Estas personas son Víctor Alberto y Margarita, participantes en el análisis de requisitos y en la evaluación final del Trabajo de Fin de Grado anterior (Amor Sanz et al., 2023).

Para poner en contexto a los usuarios a continuación tenemos una pequeña descripción de cada uno:

- Víctor Alberto:

Investigador predoctoral en accesibilidad web y RSC por la UNED. Ciego de nacimiento con percepción parcial de luz. Utiliza las tecnologías de asistencia con frecuencia en su día a día.

- Margarita:

Ciega con un 10 % y un 4 % de visión en cada ojo. Utiliza todas las funcionalidades de su teléfono móvil mediante el sistema de accesibilidad.

6.2. Diseño experimental

A continuación, se muestra en detalle el procedimiento a seguir durante la evaluación, incluyendo tareas específicas, cómo interactuarán con la imagen a través de

la aplicación y cómo recopilaremos su valioso feedback para mejorar la experiencia del usuario.

Buenos días a todos. Hoy les vamos a presentar nuestra aplicación, diseñada para poder interpretar imágenes para personas con discapacidad visual.

Para ponerlos un poco en contexto, nuestra aplicación tiene su base en la aplicación que probaron el año pasado, la cual les sacaba una descripción de la imagen, además de poder navegar en ella para saber donde se ubican los distintos objetos de la imagen. Este año nosotros nos pusimos el objetivo de mejorar esa aplicación, optimizando y añadiendo nuevas funcionalidades para poder mostrar más información precisa y relevante para ustedes.

Con esta evaluación buscamos que esta aplicación les pueda ser útil a la hora de conocer mucha más información sobre las imágenes, además de cómo y dónde están ubicados los objetos relevantes de la misma. Una vez hayan probado esta versión pueden darnos una opinión sincera, así como decirnos posibles mejoras.

A continuación les vamos a explicar en qué va a consistir el experimento. Les pondremos 3 imágenes a cada uno de ustedes y podrán navegar por la pantalla para obtener más información además de recibirla de manera auditiva. Una vez hayan recabado toda la información posible, se les pedirá un feedback sobre la imagen para saber la veracidad de la información dada, y de si la que han podido recopilar por su cuenta es útil.

Su privacidad es importante para nosotros. Nos comprometemos a proteger sus datos personales y a utilizarlos de manera responsable. Por ello, queremos ser transparentes en cuanto a cómo obtenemos su consentimiento para la recolección y uso de sus datos, y cómo puede acceder a la información sobre nuestras prácticas de privacidad.

■ *Consentimiento:*

- *Solicitud de consentimiento: Al darnos su consentimiento de poder recopilar información. La información que guardaremos será la siguiente: nombre, datos de uso de la app y opinión sobre la misma.*
- *Opciones de consentimiento: Ustedes tienen control sobre los datos que comparte con nosotros. Pueden elegir qué datos desea proporcionar y para qué fines desea que los usemos.*
- *Retiro del consentimiento: Pueden retirar su consentimiento en cualquier momento, comunicándonos que no quieren que se recopile su información para este proyecto.*

■ *Nuestro compromiso con la privacidad:*

- *Uso responsable de datos: Solo utilizamos sus datos personales para los fines que le hemos comunicado y de acuerdo con la ley aplicable.*
- *No compartimos sus datos: No vendemos ni alquilamos sus datos personales a tercero*

6.2.1. Tareas

Una vez introducida la aplicación, explicaremos los pasos a realizar para cada usuario.

1. *Para preparar el dispositivo, nos metemos en la galería a través de la aplicación y seleccionamos una imagen para que se pueda utilizar esta en plenas condiciones.*
2. *Las distintas funcionalidades a realizar son:*
 - a) *Escuchar la información auditiva proporcionada.*
 - b) *Pulsar para ver cuantos objetos hay.*
 - c) *Navegar por la pantalla para recopilar información sobre la imagen.*
 - d) *Mantener pulsado sobre un objeto para recibir información adicional.*
3. *Por último les pedimos Feedback sobre la exploración de la imagen, así como una descripción final sobre la misma a partir de la información que hayáis obtenido por medio de las distintas funcionalidades.*

6.2.2. Materiales y Entorno de evaluación

La sesión se realizó de manera presencial en la Facultad de Informática de la Universidad Complutense de Madrid y fue grabada para permitir una visualización posterior y una mejor revisión. Los usuarios probaron el funcionamiento de nuestra aplicación, y para ello se cargaron tres imágenes a cada uno, las dos primeras imágenes eran diferentes para cada usuario y como última la misma para ambos usuarios, con el fin de poder comparar resultados. Todas las funcionalidades implementadas estaban a su disposición para que pudieran proporcionarnos una interpretación de la imagen. Las imágenes que seleccionamos son imágenes que mostraban bien todas las funcionalidades implementadas, de manera que cada usuario probó con una imagen en la que solo aparecían objetos y otras dos imágenes en las que aparecían personas y objetos. Las imágenes usadas durante esta sesión se pueden ver en la Figura 6.1.

Los usuarios utilizaron dos teléfonos *Android* con la última versión de nuestra aplicación instalada. Después de las descripciones y la navegación por la imagen, se realizaron preguntas para saber que habían podido interpretar o no, y si las funcionalidades habían resultado útiles. También se les pidió una descripción de la imagen a partir de la información que pudieron obtener.

Finalmente se realizó un cuestionario de usabilidad SUS.

6.3. Desarrollo de la sesión de evaluación

Durante la evaluación, mostramos a los usuarios varias imágenes con las que intentamos reflejar situaciones normales de la vida cotidiana. Queríamos que fueran escenas con las que la gente se pudiera identificar fácilmente. A continuación, se mostrará la información proporcionada por la aplicación obtenida por cada usuario

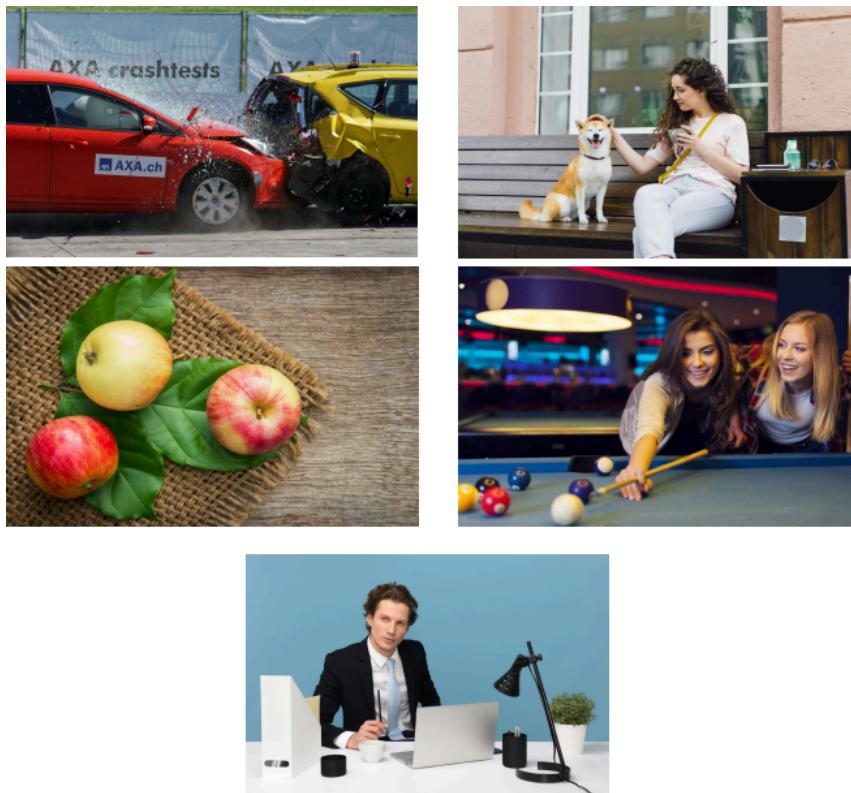


Figura 6.1: Imágenes para los usuarios

en cada una de las imágenes. Los objetos sobre los que no se haya pulsado, aparecen en la lista de objetos pero incluyendo “no encontrado” entre paréntesis. Las palabras que aparecen sin traducir se debieron a algún error en la traducción de las descripciones y aparecen tal y como se le mostraron al usuario.

6.3.1. Primera imagen para Víctor Alberto (Figura 6.2)

Para esta imagen, la información proporcionada por la aplicación, según hemos explicado anteriormente, fue la siguiente:

- Descripción: there are three apples on a burlock cloth with leaves.
- Descripción traducida: hay tres manzanas en un paño de *burlock* con hojas.
- Descripción cuantitativa: Se han detectado 3 manzanas en la imagen.
- Objetos detectados y sus colores:
 - Manzana rojo
 - Manzana 1 salmón
 - Manzana 2 rojo
 - Descripción detallada: Hay una manzana roja sentada sobre una hoja sobre una mesa.

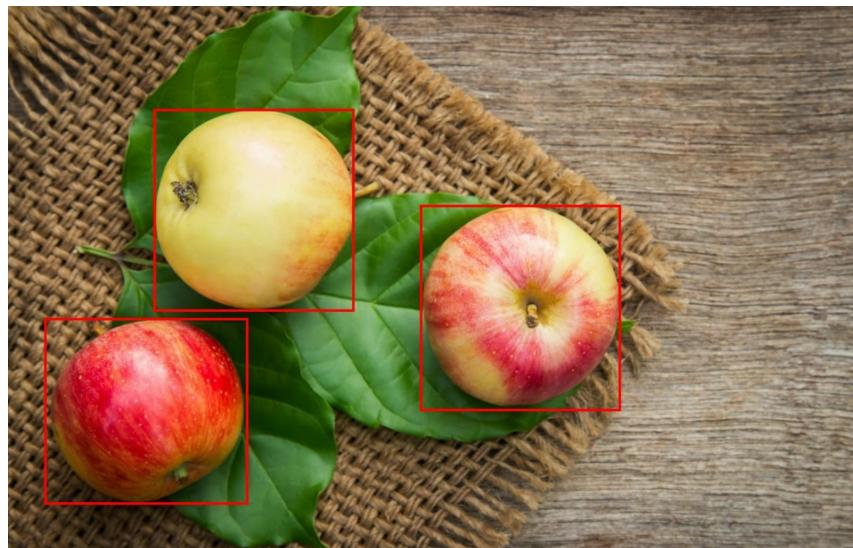


Figura 6.2: Primera imagen para Víctor Alberto

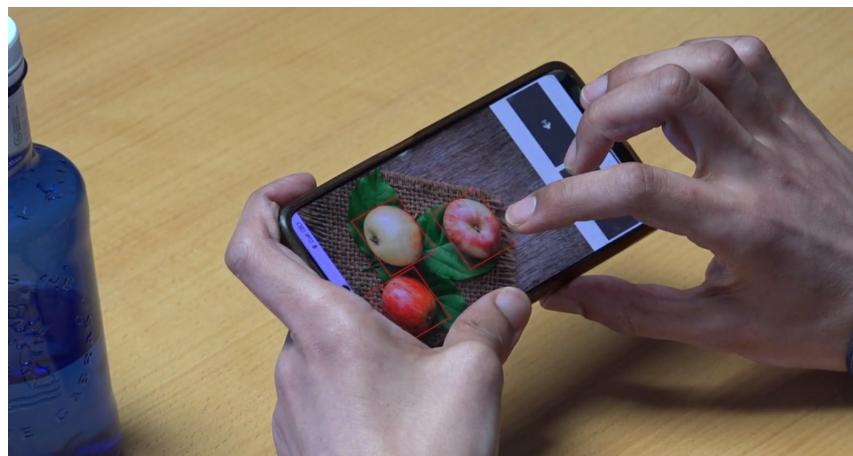


Figura 6.3: Sesión de evaluación

Para esta imagen la descripción proporcionada por Víctor Alberto fue: *3 manzanas rojas encima de la mesa, cada una encima de la otra*. Fue capaz de encontrar todos los objetos de manera rápida al navegar por la imagen (Figura 6.3), aunque el hecho de que hubiera manzana sin sufijo le resultó confuso, por lo que pensaba que no había encontrado todas. Gracias a los colores pudo saber que las manzanas no eran todas del mismo color.

6.3.2. Primera imagen para Margarita (Figura 6.4)

Para esta imagen, la información proporcionada por la aplicación, según hemos explicado anteriormente, fue la siguiente:

- Descripción: cars are in a crash with a sign on the side of the road.
- Descripción traducida: coches están en un accidente con una señal en el lado de la carretera.



Figura 6.4: Primera imagen para Margarita

- Descripción cuantitativa: Se han detectado 2 coches en la imagen.
- Objetos detectados y sus colores:
 - Coche rojo
 - Descripción detallada: Coche amarillo con un frente roto y un coche rojo con una capucha rota.
 - Coche 1 rojo
 - Descripción detallada: Hay un coche rojo que se estrella en el lado de la carretera.

Margarita proporcionó la siguiente descripción para la imagen: *Hay dos coches que se han dado un golpe*. A pesar de la aparente simpleza de la foto, estuvo navegando por ella un largo tiempo. Desde un primer momento pudo identificar bien ambos coches, no tuvo éxito al encontrar la señal que se dice en la descripción del sistema ya que no existía. Obtuvo información extra de cada coche gracias a la descripción detallada, sin embargo, le resultó confuso que esta dijera para coche1 un color distinto. No consiguió hacerse una idea de cómo era el accidente o de si había o no personas implicadas.

6.3.3. Segunda imagen para Víctor Alberto (Figura 6.5)

Para esta imagen, la información proporcionada por la aplicación, según hemos explicado anteriormente, fue la siguiente:

- Descripción: woman sitting on a bench with a dog and a cell phone.
- Descripción traducida: mujer sentada en un banco con un perro y un teléfono celular.

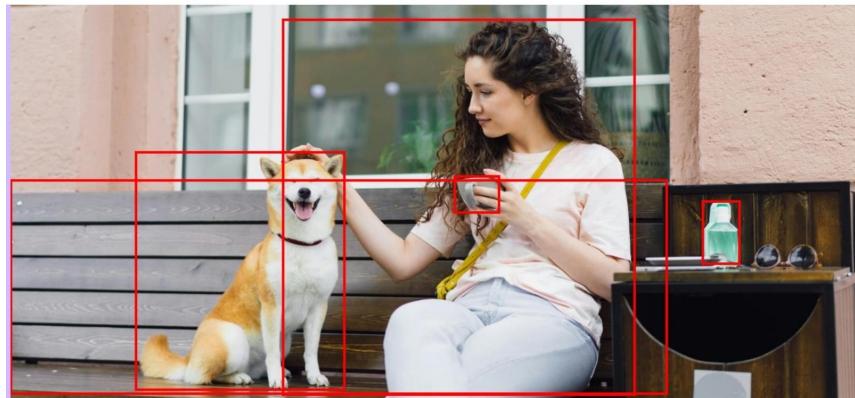


Figura 6.5: Segunda imagen para Víctor Alberto

- Descripción cuantitativa: Se han detectado un o una teléfono celular, un o una perro, un o una bench, un o una persona y un o una botella en la imagen.
- Objetos detectados y sus colores:
 - Perro rosa claro
 - Mujer joven (Persona)
 - Descripción detallada: mujer sentada en un banco con un perro y un teléfono celular
 - Bench marrón
 - Botella (No encontrado)
 - Teléfono celular (No encontrado)

La descripción que proporcionó Víctor Alberto para esta imagen fue: *Una mujer sentada a la derecha del banco y el perro debajo del banco*. Durante la navegación, debido a las proporciones, la imagen no ocupaba el ancho de la pantalla, sin embargo pudo saber en todo momento cuando estaba pulsando o no fuera de la imagen gracias al mensaje auditivo. Fue capaz de determinar que la mujer estaba sentada en el extremo derecho de la imagen, por medio de la navegación, gracias a que al haber superposición con el banco, al pulsar sobre la mujer, se nombran ambos objetos. Si bien la descripción detallada de la mujer no le aportó información extra, sí que gracias a otra funcionalidad pudo saber que se trataba de una mujer joven, aunque le hubiera gustado conocer más detalles como el pelo, estatura... Esto mismo ocurre con el perro, también le gustaría conocer más detalles y no sólo el color.

6.3.4. Segunda imagen para Margarita (Figura 6.6)

Para esta imagen, la información proporcionada por la aplicación, según hemos explicado anteriormente, fue la siguiente:



Figura 6.6: Segunda imagen para Margarita

- Descripción: two women playing pool in a bar with a pool table
- Descripción traducida: Dos mujeres jugando al billar en un bar con una mesa de billar
- Descripción cuantitativa: Se han detectado 2 personas, 6 bola deportivas y un o una bola deportiva en la imagen.
- Objetos detectados y sus colores:
 - Bola deportiva 4 rojo
 - Bola deportiva 1 caqui
 - Descripción detallada: Hay una bola de billar amarilla y un taco negro en una mesa.
 - Bola deportiva 2 rojo
 - Bola deportiva rosa
 - Bola deportiva (no encontrada)
 - Bola deportiva (no encontrada)
 - Bola deportiva (no encontrada)
 - Mujer joven 1 (persona 1)
 - Descripción detallada: Rubia mujer jugando piscina con amigos en un bar
 - Mujer joven (persona)
 - Descripción detallada: Dos mujeres jugando al billar en un bar con una mesa de billar

Margarita proporcionó la siguiente descripción para la imagen: *Hay dos mujeres, una de ellas es rubia, están en un bar, una de ellas tiene un taco y que están las bolas encima de la mesa, por lo tanto estarán en plena partida.* A priori, esta imagen al tener tantos objetos, se podría esperar que Marga navegaría un largo rato por ella. Sin embargo, no fue así. Gracias a la descripción detallada de una de las personas pudo obtener información sobre el color de pelo de esta. En la información que proporciona el sistema no se sugiere en ningún momento que la imagen haya sido tomada durante una partida. Sin embargo, Margarita al identificar las distintas bolas en la mesa, fue capaz de intuirlo. No le hizo falta ni localizar todas las bolas para poder describir la imagen. Además, también gracias a la descripción general sabían que se trataba de un bar.

6.3.5. Tercera imagen para ambos usuarios (Figura 6.7)

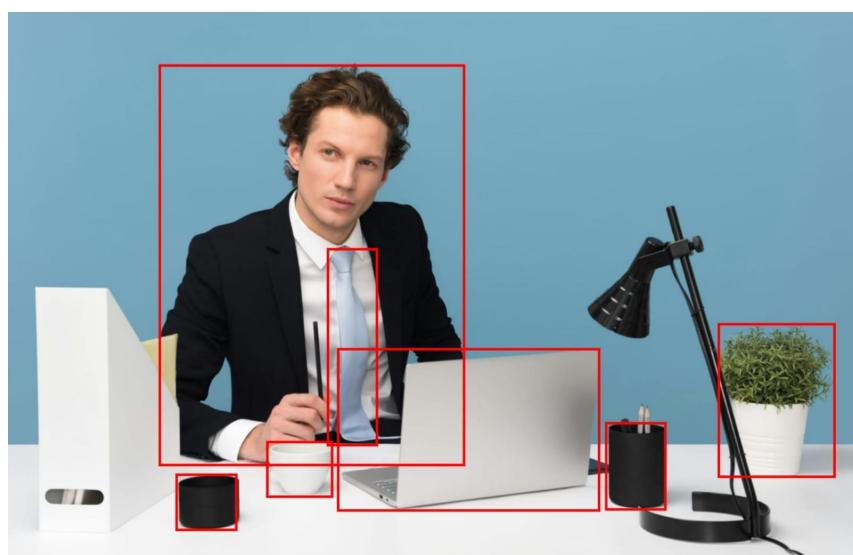


Figura 6.7: Tercera imagen para ambos

Para esta imagen, la información proporcionada por la aplicación, según hemos explicado anteriormente, fue la siguiente:

- Descripción: there is a man sitting at a desk with a laptop and a cup
- Descripción traducida: hay un hombre sentado en un escritorio con un portátil y una taza
- Descripción cuantitativa: Se han detectado un o una tie, 3 tazas, un o una persona, un o una planta ahuecada y un o una laptop.
- Objetos detectados y sus colores:
 - Hombre de mediana edad (Persona)
 - Hay un hombre en un traje, sentado en un escritorio con un portátil.
 - Laptop marrón

- Planta ahuecada marrón
 - Descripción detallada: Hay una lámpara negra que está en la mesa junto a una planta.
- Tie marrón
- Taza marrón
 - Descripción detallada: Hay una lámpara de escritorio negra y un porta plumas en una mesa blanca.
- Taza 1 (no encontrado)
- Taza 2 (no encontrado)

La descripción que proporcionó Víctor Alberto fue: *Un señor de mediana edad, sentado en el escritorio con el ordenador*. Durante la navegación, al pulsar en la persona, no había cargado aún la información acerca del género y la edad, por lo que saltó el mensaje auditivo: *pulse otra vez para obtener más información*. Esto le resultó útil, y cuando posteriormente pulsó otra vez, ya pudo obtener más información (género y edad), complementando así la descripción general proporcionada por la aplicación.

La descripción que proporcionó Margarita fue: *Mesa de oficina con una lámpara, una planta, y una taza para los bolígrafos. Un señor de mediana edad, en traje y con corbata, sentado detrás del escritorio con su portátil*. Durante la navegación, el uso de las distintas funcionalidades le brindó bastante información extra. En ninguna de las dos descripciones iniciales dadas por la aplicación, se menciona que haya alguna lámpara negra, sin embargo, esto sí se menciona en la descripción detallada de uno de los objetos. En estas descripciones tampoco se dice nada acerca de que en una de las tazas se guarden bolígrafos. Sin embargo, gracias a la descripción detallada de la propia taza se menciona que hay un portaplumas, gracias a esto pudo intuir que se trataba de una taza con bolígrafos. Gracias a la información extra que se da de la persona, pudo situar en un rango de edad al hombre. Aunque le costó encontrar algunos objetos y que la descripción inicial le confundía acerca de dónde estaba sentado el hombre se pudo hacer una idea precisa de la imagen.

6.4. Análisis de los datos obtenidos

Durante la evaluación, se fueron recopilando diversas opiniones e ideas que se les pudiesen ocurrir a los usuarios. A continuación dejamos un listado de las mismas:

- Cuando uno pulsa fuera de la imagen, en zonas cercanas al marco de la pantalla, la aplicación te avisa de que estás fuera de la imagen, lo cual les resulta bastante útil. Pero en ocasiones existe un tramo vacío entre la imagen y los botones. Durante la evaluación nos hemos dado cuenta de que al tocar en esa zona no se da ningún aviso.
- La aplicación informaba a los usuarios cuando la imagen estaba girada y que debían girar el dispositivo para una correcta exploración. Según los usuarios les era de gran ayuda porque suelen tener problemas con la orientación.

- Un dato muy relevante fue conocer que el proporcionarles una información cuantitativa para saber qué objetos había en la imagen de los cuales podían recibir más información detallada además de poder saber donde se encontraban dentro de la imagen les fue muy útil.
- La utilización de los sufijos fue de gran importancia a la hora de diferenciar entre objetos de la misma clase.
- Los objetos de forma individual proporcionaban colores. Fue un punto positivo comentado por los usuarios. Debido a que en su día a día tienen dificultades con saber que color es un objeto por lo que les gustó que la aplicación dijera de que color era cada objeto.
- Una cosa que les pareció muy importante fue el hecho de que cuando obtenían más información específica de una persona, podían saber el género y la edad. Un dato que repitieron varias veces a cerca de lo importante y útil que era esa información.
- Vieron como algo muy positivo que a la hora de explorar la imagen, cuando no había ningún objeto donde pulsaban dentro de la imagen les comunicara de forma auditiva que no había ningún objeto.
- Otro problema con el que se encontraron los usuarios es que en ocasiones la información tarda demasiado en cargar. Esto resultaba confuso para los usuarios ya que muchas veces desconocían si se estaba cargando o no la información. Había veces que la carga después de la acción realizada por el usuario tardaba varios segundos. Este problema se presenta sobretodo en la descripción detallada. Como solución a este problema los usuarios sugirieron que la aplicación avise al usuario cuando esta información se esté cargando.
- Fallos en los colores. En algunas ocasiones el color descrito por la aplicación no era correcto. Esto también provocaba, en algunos casos, que el color proporcionado por la descripción detallada fuera diferente al color calculado por la aplicación. Un ejemplo claro de este problema se dio en la imagen del accidente entre dos coches en el que al describir uno de los coches la descripción corta te decía un color, y la descripción detallada decía otro, desconcertando en este caso a Margarita.

A su vez, los usuarios nos dieron ideas de posibles implementaciones futuras, entre las cuales:

- Sincronización con álbumes de fotos del dispositivo.
- Posición relativa entre los objetos.
- Descripción del fondo de la imagen para tener más información del contexto.
- Vinculación de las personas identificadas que tengas en la galería mediante reconocimiento facial.

	Margarita	Víctor Alberto
Usaría esta aplicación muy frecuentemente.	5	4
Encuentro esta aplicación innecesariamente compleja.	2	3
Pienso que la aplicación es fácil de usar.	4	4
Creo que necesitaría ayuda de una persona especializada para poder usar esta aplicación.	1	2
He encontrado las funcionalidades de esta aplicación bien integradas.	2	2
Hay demasiadas inconsistencias en esta aplicación.	3	3
La mayor parte de la gente aprendería a usar la aplicación muy rápido.	5	4
He encontrado esta aplicación muy complicada de utilizar.	1	1
Me he sentido muy cómodo/a usando esta aplicación.	4	4
Necesitaba aprender muchas cosas antes de empezar con este sistema	2	2

Tabla 6.1: Resultados encuesta SUS

- Describir caras. Se mencionó la posibilidad de ajustar más los recortes, con el objetivo de obtener mejores descripciones de las personas.
- Identificar razas de perro, ya que actualmente la descripción se limita a la palabra *perro*.

6.4.1. SUS

Para finalizar la evaluación se les realizó un cuestionario de usabilidad SUS¹ que contiene las siguientes preguntas a las que tendrán que responder con una puntuación de 1 al 5 dependiendo de cuan de acuerdo estén. Podemos ver los valores obtenidos en la Tabla 6.1.

Las puntuaciones finales para cada uno de los usuarios son las siguientes:

- Margarita

Respuestas a las afirmaciones impares : $(5 + 4 + 2 + 5 + 4) - 1 * 5 = 15$

Respuestas a las afirmaciones pares : $25 - (2 + 1 + 3 + 1 + 2) = 16$

Cálculo SUS : $(15 + 16) * 2,5 = 77,5$

- Víctor Alberto

Respuestas a las afirmaciones impares : $(4 + 4 + 2 + 4 + 4) - 1 * 5 = 13$

Respuestas a las afirmaciones pares : $25 - (3 + 2 + 3 + 1 + 2) = 14$

Cálculo SUS : $(13 + 14) * 2,5 = 67,5$

¹<https://forms.gle/5t78bUcJC65qND8L6>

La media de puntuaciones SUS ha sido de 72,5, esta puntuación se tiene en cuenta en un rango entre 0 a 100. Por tanto podemos asumir que la aplicación tiene una buena usabilidad.

Como se puede ver, la puntuación de Víctor Alberto es inferior a la de Margarita. Esto podría deberse a que, como nos contó Víctor en la evaluación, usa el sistema de accesibilidad con más frecuencia que Margarita. Por lo tanto, conoce aplicaciones con muy buena usabilidad. Otro factor a tener en cuenta es que Víctor Alberto tiene menor visión que Margarita y es ciego de nacimiento, por tanto, necesita que todos los aspectos de usabilidad sean totalmente accesibles.

Capítulo 7

Conclusiones y Trabajo Futuro

En este capítulo se incluyen las conclusiones obtenidas tras la realización de nuestro trabajo, y se añaden algunas posibles mejoras a realizar en un futuro.

7.1. Conclusiones

Este proyecto partía con una base, el Trabajo de Fin de Grado anterior “*Desarrollo de una aplicación móvil para la generación de descripciones de imágenes para personas con discapacidad visual*” (Amor Sanz et al., 2023). El primer punto al que necesitábamos llegar una vez comenzado el desarrollo, era igualar en términos de funcionalidad lo que ya se consiguió el año pasado. Una vez llegamos a ese punto, comenzamos a implementar las mejoras que se habían planteado tanto en la evaluación final de nuestros compañeros como en las lluvias de ideas que se habían realizado durante las primeras fases del desarrollo. Es por eso que se implementaron las siguientes medidas:

- La diferenciación de objetos mediante el uso de sufijos numéricos. Esta medida se implementa a raíz de la dificultad de diferenciación entre distintas entidades con el mismo nombre. De esta manera si hay dos objetos iguales juntos, el usuario puede saber que son dos objetos distintos, así como ubicarlos de forma más precisa.
- Dependiendo de si la foto está tomada en horizontal o vertical se girará o no. Esto hace que se aproveche más la pantalla, y evita que los *bounding boxes* puedan ser demasiado pequeños.
- Para objetos que no se traten de personas se dice el color predominante.
- Si se pulsa sobre objetos solapados, se nombran todos estos, para que sean más sencillos de ubicar.
- Para las personas se dice el género de las mismas y una edad aproximada, siendo los rangos los siguientes:
 - Joven.

- Edad media.
 - Edad avanzada.
- Si se mantiene pulsado sobre un objeto, se obtendrá una descripción más detallada del mismo, lo que es útil para obtener más información respecto al objeto deseado.
 - Se ha añadido el mensaje “estás fuera de la imagen” y el mensaje “no hay ningún objeto” para que puedan tener conocimiento de qué están tocando en todo momento.
 - Poder realizar una fotografía mediante la cámara y usarla directamente en la aplicación. Esto es una gran ventaja, ya que si en algún momento el usuario quisiera identificar qué tiene delante o simplemente conocer el color de un objeto pueda hacerlo de forma rápida y sencilla.
 - Al cargar la imagen y después de decir la descripción inicial, se ha añadido una descripción cuantitativa que dice cuántos y qué objetos se han detectado.

Durante el proceso de desarrollo, fueron surgiendo ciertas limitaciones. Al principio, nos encontramos con la problemática de montar un servidor que funcionase bien. No encontramos muchas alternativas que se adaptaran a nuestro bajo presupuesto. De hecho, en el mismo *Firebase* se encuentran mejores alternativas, pero todas ellas con un precio algo más elevado. En segundo lugar, cuando iniciamos nuestra investigación acerca de cómo usar las distintas APIs que necesitábamos para cumplir con nuestros objetivos, nos encontramos que muchas de las que llamaban nuestra atención cobraban dinero por cada consulta. Este factor nos impidió en muchas ocasiones hacer uso de las mismas.

El estudio realizado sobre este tema nos ha llevado a tener una mayor comprensión de los campos como inteligencia artificial, APIs y aprendizaje automático. Entender cómo funciona la integración de todas estas funcionalidades, nos llevó al planteamiento de crear una aplicación modular. Es decir, una aplicación en la que de manera sencilla se puedan añadir funcionalidades evitando incompatibilidades entre las funciones ya integradas. Esto lo vemos como una gran ventaja, ya que cualquier persona puede investigar acerca de la aplicación y aportar su granito de arena. Facilitando de esta manera la expansión de la aplicación.

Por otro lado, este estudio ha logrado alcanzar los objetivos establecidos al inicio de este proyecto. En primer lugar y más importante se ha conseguido realizar una aplicación capaz de interpretar imágenes para personas con discapacidad visual y proporcionar información precisa sobre las mismas. Gracias a la colaboración de personas con esta discapacidad, pudimos hacer una evaluación y saber de primera mano que nuestra aplicación sirve para su principal propósito.

Nuestros resultados ofrecen perspectivas significativas sobre cómo pueden beneficiarse las personas con discapacidad visual. Se ha podido probar la aplicación en una evaluación como se ha comentado anteriormente. Las personas con discapacidad visual pueden conectar más con la imagen que tienen en pantalla debido a la gran variedad de información que reciben de la aplicación. Al proporcionarles una

herramienta que les permite acceder y comprender mejor el mundo visual, estamos facilitando su participación en una variedad de actividades diarias y fomentando su independencia y autonomía.

7.2. Trabajo Futuro

Tras la evaluación, y teniendo en cuenta la opinión y las ideas de los participantes, se han detectado las siguientes posibles mejoras:

- Añadir un mensaje auditivo que indique cuando se está cargando la descripción detallada. En las ocasiones en las que tardaba más de lo debido en cargar, puede desconcertar al usuario al no saber si llegó a pulsar bien.
- Modificar la forma de acceder a la descripción detallada, en vez de pulse largo, dar dos toques. Al estar acostumbrados al sistema de funcionamiento de *VoiceOver* donde las acciones se realizan pulsando dos toques, les resultaba más intuitivo que una pulsación larga.
- Poder aportar más información acerca de las distintas personas, como color de pelo, el color de piel, altura, estado de ánimo e intentar que los rangos de edad no sean tan amplios.
- Dar más información acerca de ciertos objetos. Por ejemplo en el caso del perro, su raza o color, en el caso de un coche la marca, modelo y/o tipo.
- Poder ubicar dónde está el objeto, con mensajes como “detrás de”, “al lado de”, etc.
- Que la descripción cuantitativa se diga por cuadrantes siguiendo una matriz de 3x3, para que los objetos detectados sean más sencillos de encontrar.
- En lugar de cargar las imágenes desde galería de manera individual, cargar un álbum y poder cambiar de foto desplazando hacia la izquierda o hacia la derecha.
- Permitir que detecte caras y se puedan almacenar para que se puedan reconocer en otras fotos.
- Ofrecer una descripción del fondo de la imagen.

Por otro lado, de cara a mejorar la experiencia del usuario, estas son las mejoras que creemos que sería necesario implementar:

- Solucionar los errores provocados en el servidor y en nuestra aplicación, con el fin de conseguir mayor robustez.
- Intentar reducir el tiempo de carga de las distintas informaciones. En algunas ocasiones llegaba a tardar varios segundos en hacerlo, lo que empeora la experiencia del usuario.

- Añadir funcionalidades para poder ubicar objetos de manera más precisa.
- Desarrollar una aplicación que sea completamente compatible con el *talkback*.
- Que los colores detectados sean más precisos.

Capítulo 8

Conclusions and Future Work

This chapter includes the conclusions obtained from our work and some possible improvements to be introduced in the future.

8.1. Conclusions

This project started with a base, the previous Final Degree Project “*Development of a mobile application for the generation of image descriptions for visually impaired people*”(Amor Sanz et al., 2023). The first point we needed to reach once we started development was to match in terms of functionality what was already achieved last year. Once we reached that point, we started to implement the improvements that had been raised both in the final peer review and in the brainstorming that had taken place during the early stages of the development. That is why the following measures were implemented:

- The difference between objects through the use of numerical suffixes. This measure is implemented because of the difficulty of differentiation between different entities with the same name. In this way, if there are two identical objects together, the user can know that there are two different objects, as well as locate them more precisely.
- It depends on whether the photo is taken horizontally or vertically, it will be rotated or not. This makes more use of the screen, and prevents the bounding boxes from being too small.
- For objects that are not people, the predominant colour is stated.
- If you click on overlapping objects, they are all named, so that they are easier to locate.
- For persons, the gender and an approximate age are given, the ranges being as follows:
 - Young.

- Middle age.
 - Advanced age.
- Holding down on an object will give a more detailed description of the object.
Useful for obtaining more information about the desired object.
 - The message “you are out of the picture” and the message “there is no object” have been added so that they can be aware of what they are touching at all times.
 - Being able to take a picture using the camera and use it directly in the application. This is a great advantage, because if at some point the user wants to know what is in front of him or simply to know the colour of an object, he can do it quickly and easily.
 - Loading the image and after saying the initial description, a quantitative description has been added, saying how many and which objects have been detected.

During the development process, certain limitations emerged. At the beginning, we were faced with the problem of setting up a well-functioning server. We did not find many alternatives that would suit our low budget. In fact, in the same *Firebase* you can find better alternatives. All of them with a somewhat higher price tag. Secondly, when we started our research about which ones and how to implement the different APIs we needed to meet our objectives, we found that many of the ones that caught our attention charged money for each query. This was a factor that prevented us from using them on many occasions. In other words, the biggest limitation we found was the budget.

The study carried out on this topic has led us to have a better understanding of fields such as artificial intelligence, APIs and machine learning. Understanding how the integration of all these functionalities works, led us to the approach of creating a modular application. In other words, an application in which functionalities can be added in a simple way, avoiding incompatibilities between the already integrated functions. We see this as a great advantage, as anyone can investigate the application and contribute their bit. Thus facilitating the expansion of the application.

Our results offer significant insights into how visually impaired people can benefit. The app has been tested in an evaluation as discussed above. Visually impaired people can connect more with the image on the screen due to the wide variety of information they receive from the app. It is a very positive thing to be able to have made an application for people with this disability. By providing them with a tool that allows them to access and better understand the visual world, we are facilitating their participation in a variety of daily activities and fostering their independence and autonomy.

8.2. Future Work

After evaluation, and taking into account the opinion and ideas of the participants, the following possible improvements have been identified:

- Add an audible message to indicate when the detailed description is loading. On occasions when it took longer than expected to load, it can be disconcerting for the user not to know if they got the right click.
- Modifying the way to access the detailed description, instead of a long press, give two taps. As they were used to the operation system of
- *VoiceOver* where the actions are performed by pressing two taps, they found it more intuitive than a long press.
- To be able to provide more information about different people, such as hair colour, skin colour, height, mood, and to try to keep that age ranges are not so wide..
- Give more information about certain objects. For example in the case of a dog, its breed or colour, in the case of a car the make, model and/or type.
- Be able to locate where the object is, with messages such as “behind”, “next to”, etc.
- That the quantitative description is said by quadrants following a 3x3 matrix. To make detected objects easier to find.
- Instead of uploading images from the gallery individually, upload an album and be able to switch between photos by scrolling left or right.
- Allow it to detect faces and store them so that they can be recognised in other photos.
- Provide a description of the background of the image.

On the other hand, in order to improve the user experience, these are the improvements that we believe should be implemented:

- Fix the errors caused in the server and in our application, in order to achieve greater robustness.
- Try to reduce the loading time of the different information. On some occasions it takes several seconds to do so, which worsens the user experience.
- Adding functionalities to be able to locate objects more precisely.
- Developing an application that is completely compatible with *talkback*.
- Fix the colour detection to be more accurate.

Capítulo 9

Contribuciones Personales

9.1. Wалид Bousnitra Bousnitra

A la hora de buscar un tema para el trabajo de fin de grado buscaba un tema que implicase un desarrollo completo de software. Cuando decidí elegir este tema una las razones que más influyó en mi decisión fue poder trabajar en un proyecto ingenieril, que a diferencia de otros temas se aleja de los aspectos mas teóricos y se centra en un producto funcional. Además con este proyecto conseguiría junto a mis compañero abordar un problema real y aportar nuestra solución. Otra de las razones ha sido poder hacer uso de la inteligencia artificial aplicada y no solo a nivel teórico.

Al estar en el itinerario de Tecnología Específica de Computación partía con los conocimientos básicos de los modelos de inteligencia artificial. Sin embargo antes de comenzar con nuestra investigación tuve que recordar la terminología dentro del mundo de la IA para poder buscar mejor la información necesaria. Por otro lado, el desarrollo para *Android* ha sido un mundo nuevo, ya que es una tecnología con la que nunca me había enfrentado. En *Android* se puede realizar código tanto en Java como en *Kotlin*, dado que Java es un lenguaje que estudiamos en profundidad durante la carrera, este era la mejor opción para poder empezar con los primeros prototipos lo antes posible. Finalmente y antes de empezar con el trabajo, había que leer en profundidad el Trabajo de fin de grado del curso anterior, incluyendo las entrevistas realizadas y pruebas de código.

Para mayor claridad en la contribución podemos hablar de dos partes principales, la contribución en la implementación del servidor, la contribución en la implementación del cliente y contribuciones generales:

- Servidor: Inicialmente y partiendo de las ideas del Trabajo de Fin de Grado (Amor Sanz et al., 2023) anterior probamos a crear un servidor Flask muy sencillo que con una estructura *request/reply* (Sección 5.1).

Una vez decidimos usar las tecnologías de *Firebase*, creé el proyecto donde se despliega el servidor e implementé una aplicación sencilla para poder comprobar la conexión con el servidor. Mientras que intentábamos avanzar con otros aspectos en el cliente, me informé sobre las distintas funcionalidades de

Firebase para poder ver cuales de ellas eran útiles para nosotros. Inicialmente encontré *Firebase Storage*, que parecía un buen camino a seguir para poder guardar imágenes en el servidor, y *Firebase Machine Learning* que era interesante ya que permitiría incluir modelos de IA en la lógica del servidor. Como explicamos en la memoria dejamos estas dos funcionalidades rápidamente ya que tienen una serie de limitaciones. Investigando un poco más encontré las *Cloud Functions* de Google que permite tener funciones aisladas en la nube. Esta funcionalidad nos ha sido de gran utilidad ya que permitía encapsular los servicios de terceros que usamos para los modelos. Una vez decidimos que esta sería la forma en la que implementaríamos el servidor comencé a probar con funciones simples ya que tuve que aprender JavaScript. El objetivo era conseguir recibir imágenes y enviarlas a *Hugging Face*. Ya que el cliente aun no estaba preparado empecé por una función de tipo *OnRequest* cuyo trigger es simplemente la conexión entre cliente y servidor. Con esta función y a través de la consola online pude comprobar como todo se conectaba correctamente. Investigando descubrí que una forma muy robusta de enviar información con el protocolo HTTPs era la codificación en Base64. Tanto Java como Javascript soporta esta codificación por lo que fue sencillo implementarla en el proyecto. Una vez teníamos una conexión sólida, incluí las consultas a los distintos modelos de *Hugging Face*. En este punto del desarrollo ya entendíamos lo suficiente las Cloud Functions para poder desplegar y eliminar con facilidad las distintas funciones que necesitábamos en el servidor.

En cuanto a funcionalidades más concretas, uno de los objetivos que teníamos era informar al usuario sobre el fondo de la imagen. Buscando en la plataforma de *Hugging Face* algún modelo con el que se pudiese abordar este problema encontré el modelo de *LiheYoung/depth-anything-small-hf*. Aunque este modelo no estaba disponible para realizar consultar merecía estudiarse para futuras versiones. Implementé un *Notebook*¹ con el esquema que comentamos en la sección 5.8.2. También intenté, sin éxito, desplegar esta funcionalidad en el servidor mediante la librería *transformers* de python.

- Cliente: Inicialmente partimos de una aplicación muy simple a la cual íbamos añadiendo funcionalidades. Una vez avanzamos con el servidor, para mantener una estructura robusta en el cliente, estructuré el proyecto de *Android*. Esta estructura no solo conseguía un proyecto más legible y organizado sino que encapsulaba las tres partes del código, los hilos, la imagen y el servidor. En el servidor apliqué la herencia de Java para implementar las consultas a los distintos modelos de *Hugging Face* y de esta forma todas las consultas al servidor heredan de la clase *Query*.

Buscando optimizar en tiempo las consultas, decidí implementar un Hilo para poder lanzar la consulta del modelo de detección de objetos a la vez que el modelo de descripción de imágenes. Para esto use la clase *Thread* de Java y así se podría ejecutar en otro hilo la consulta y solo hacia falta un *Join* antes de necesitar la información de la consulta. También implementé un segundo

¹https://colab.research.google.com/drive/1MdZSnbiAG5jFTj-bmtSP8K_NV3Ir5fj?usp=sharing

Hilo con el objetivo de lanzar un Hilo para cada objeto de la imagen pero esta implementación no se usó en la versión final del código.

Uno de los modelos que más trabajo supuso fue el de detección de objetos ya que a la clase *Identificador* se accedía de forma recurrente. En esta clase se gestionaba el *JSON* cuyos campos, por ejemplo las coordenadas, se usan en varias de las funcionalidades implementadas.

Un aspecto muy importante para la interfaz de la aplicación es la gestión de coordenadas para poder adaptar la imagen a cualquier dispositivo. Tuve que hacer los cálculos correspondientes. Lo primero que descubrí es que el ajuste a los límites de la pantalla no dependía de la orientación sino de la proporción de la imagen, teniendo esto en cuenta y contando con que para las imágenes horizontales hay que invertir los ejes obtuve 4 y cada uno tenía sus propios cálculos. Estos cálculos incluían redimensionar los *Bounding Boxes* para que se mostrasen correctamente en la pantalla del dispositivo y la transformación de las coordenadas donde pulsa el usuario a las coordenadas de la imagen. Relacionado con las coordenadas también adapté los cálculos dependiendo de la orientación de la imagen e incluí el control de los límites de la imagen.

En cuenta a los colores, el mayor obstáculo ha sido encontrar el color más próximo al color detectado en la imagen. Carlos en un primer momento realizó los cálculos manualmente pero investigando encontré el repositorio que finalmente usamos. Este repositorio incluía la clase *ColorUtils* que realizaba un cálculo más específico y permitía calcular el color mas próximo a los colores de la librería *Color* de Java.

- General: Durante todo el desarrollo hemos completado la memoria entre todos. Gracias a las indicaciones de nuestros tutores hemos corregido constantemente la memoria.

Aunque muchas veces las distintas implementaciones se realizasen de forma individual, durante todo el desarrollo hemos puesto en común todas las diferentes ideas para mejorar el trabajo.

Finalmente y gracias a Alberto y Raquel, que contactaron con Víctor Alberto y Margarita, pudimos probar la aplicación.

9.2. Sirma Sosa Fernández

Cuando hubo que determinar el tema para el Trabajo de Fin de Grado, para así concluir mis estudios en la carrera de ingeniería, tenía claro que se tenía que tratar de un trabajo que aunase gran parte de los conocimientos adquiridos durante toda la carrera. Además no quería que se tratase de un trabajo teórico, es por esto que al ver que existía la posibilidad de desarrollar una aplicación, de manera grupal, relacionadas con el mundo de la inteligencia artificial que está tan en auge, y que además podía aportar una solución para un problema real me decidió, junto con mis compañeros a escoger este tema, así que contactamos inmediatamente con nuestros tutores para decirles que estábamos encantados de realizar dicha aplicación. En

nuestra primera reunión con los tutores nos contaron que esta era, en cierto modo, la continuación de un Trabajo de Fin de Grado hecho el año pasado. Si bien es cierto que ya teníamos una base sobre la que empezar, el mundo de *Android* era completamente nuevo para mí ya que nunca había programado en dicho sistema operativo y tampoco nunca había hecho algo de esta magnitud. Por suerte durante la carrera Java se estudia en profundidad por lo que programar en sí no era nuevo si no más bien la lógica propia de una aplicación *Android*. Lo primero que se hizo fue leerse detenidamente el Trabajo de Fin de Grado (Amor Sanz et al., 2023) del que partíamos, además de visualizar las entrevistas con los usuarios y ver el código, con el fin de recabar información. Ahora sí, pasaré a detallar cual ha sido mi contribución durante el desarrollo de nuestra aplicación y para mayor claridad diferenciaré entre mis aportes al cliente y al servidor.

- Servidor: Para poder implementar nuestra aplicación lo primero que necesitábamos era un servidor, y basándonos en el Trabajo de Fin de Grado anterior (Amor Sanz et al., 2023) intentamos desarrollar el servidor en *Flask*. Realizamos un pequeño prototipo que permitía enviar mensajes al servidor y recibir una respuesta en el dispositivo móvil. Posteriormente intentamos enviar una imagen al servidor, para ello investigué *okHttp* que parecía cumplir con nuestros objetivos. Sin embargo no fui capaz de implementar *okHttp* y además mis compañeros ya estaban explorando otro camino, el uso de *Firebase*. Finalmente, nos decantamos por usar *Firebase* como servidor, y lo primero que hice fue buscar información sobre cómo conectar nuestra aplicación con el servidor, este paso era bastante sencillo. El siguiente objetivo era poder enviar una imagen a nuestro servidor. Waliq encontró la tecnología MLKit que implementa unos modelos IA, propios de *Firebase*, tanto de descripción de imágenes como de detección de objetos. Desarrollé una aplicación funcional y sencilla, que implementaba estos modelos. Bien es cierto que los modelos eran un poco pobres, sin embargo esta aplicación en cierto modo sirvió de base ya que entre otras cosas permitía acciones como coger una imagen de la galería o abrir la cámara para realizar una imagen. Sin embargo, como ya se ha dicho descartamos esta opción rápidamente. Se investigaron otras líneas como usar el *Cloud Storage* o los *buckets* de *Firebase* pero todas sin éxito hasta que se probó con las *Cloud Functions*. Estas permiten aislar las funciones de llamadas a las APIs en la nube, además el envío de imágenes, que era nuestro principal problema, resultaba bastante simple. De manera paralela al desarrollo del servidor, investigué distintas herramientas que podrían cumplir nuestros objetivos. Busqué información sobre *Microsoft Azure*, una plataforma muy completa de análisis de imágenes que se descartó por que no era gratuita; *Google Cloud Vision*, otra herramienta para analizar imágenes, también la descartamos ya que consideramos que la versión gratuita no sería suficiente para desarrollar la aplicación; también busqué información sobre los *datasets* de COCO y de OID por si era necesario para entrenar algún modelo y para saber cómo funcionaban, ya que la mayoría de modelos están entrenados con estos *datasets*; otra plataforma que investigué fue *Roboflow*, si bien permitía detectar objetos, entrenar los modelos y además parecía que tenía una fácil implementación, al estar también investigando con *Hugging Face* nos dimos cuenta de que esta última era

mucho más sencilla de implementar. Otra de las ventajas que nos proporcionaba *Hugging Face* era que había una gran variedad de modelos que satisfacían nuestros requisitos, manteniendo así una casuística similar al solo usar una plataforma, lo que podría reducir errores y ser más fácil de manejar. En *Hugging Face* encontramos todos los modelos que están actualmente incluidos en nuestra aplicación. A nivel de servidor, me encargué de desarrollar una lógica para evitar que se cierre la aplicación, ya que al tratar con *JSONs*, había veces que las APIs devolvía un error. Nuestra primera idea era devolver la llamada al servidor desde el cliente, pero esto se complicaba mucho, es por eso que decidí modificar el servidor para que si la respuesta de las APIs contenía el campo error se mandará otra vez la petición para así evitar que se cierre.

- Cliente: Nuestro primer prototipo, como ya mencioné en el apartado anterior, era el que incluía las tecnologías MLKit, de este solo nos quedamos con el botón que permitía cargar una imagen, y a partir de aquí fuimos desarrollando de acuerdo a nuestras necesidades. Desarrollamos primero la tecnología para obtener una descripción y traducirla, ya que esta era en inglés. Además añadí un botón que permitía reproducir la descripción cuando se quisiera. Cuando estuve implementada la detección de objetos, me encargué de añadir la lógica necesaria para que se añadieran sufijos a los objetos que eran iguales, para una mayor diferenciación, también junto con otro compañero desarrollamos la lógica para que se dibujasen los *Bounding Boxes*, que resultaría bastante útil para poder realizar pruebas. Otra funcionalidad que se desarrolló fue la descripción detallada, en esta mi principal participación fue implementar la lógica para diferenciar entre un toque, o un toque largo para invocar a esta funcionalidad, además implementé la distancia manhattan para poder llevar a cabo una diferenciación en caso de que se estuviera pulsando sobre más de un objeto. Otra de las funcionalidades que desarrollé fue el girar una imagen si esta era horizontal, así como añadir los TextToSpeech para mayor accesibilidad y de etiquetar los dos botones que tiene nuestra aplicación para su uso con el TalkBack. Otra de las cosas que implementé fue el poder usar las imágenes hechas con la cámara. Por otro lado también implementé la lógica de obtener la edad y género, con su respectiva llamadas a las APIs y todo lo que engloba, de las personas así como un mensaje para avisar de que esta información no está cargada y que tiene que esperar. Junto con otro de mis compañeros implementamos la descripción cuantitativa de objetos para que así los usuarios puedan saber qué objetos tienen que localizar en la imagen.

Durante todo el desarrollo, las correcciones sobre la memoria y escribir la misma se han desarrollado de manera conjunta siguiendo lo que nos decían nuestros tutores. A pesar de que en ocasiones las implementaciones se llevaron a cabo de manera individual, a lo largo de todo el proceso de desarrollo, hemos compartido todas nuestras ideas para mejorar nuestra aplicación.

Finalmente, una vez finalizada nuestra aplicación nuestros tutores, Raquel y Alberto, se pusieron en contacto con Victor Alberto y Margarita para poder llevar a cabo una evaluación en la que probaríamos nuestra aplicación con usuarios de verdad, obteniendo una información muy valiosa para futuras mejoras de la aplicación.

9.3. Carlos Martín Recio

En el marco de este proyecto en grupo, y echando la vista hacia atrás, me enorgullece saber que tomamos una buena decisión eligiendo este Trabajo de Fin de Grado. En mi caso, siempre había tenido en mente terminar mi grado en Ingeniería Informática desarrollando una aplicación móvil, al igual que el hecho de realizarlo en grupo. Esto último, fue la parte más sencilla del proyecto, ya que diría que contábamos todos con ello. Elegir el tema del proyecto fue quizás más complicado. Pero tras barajar varias opciones vimos esta temática como una oportunidad de ponernos en la piel de las personas invidentes. Personas que viven una vida normal, pero con limitaciones que no te llegas a plantear hasta que trabajas en un proyecto como este.

Al principio, creía que nos iba a quedar un poco grande. Lo creía ya que veía una responsabilidad inmensa sobre nuestros hombros. Este sentimiento se fue desvaneciendo durante los primeros meses, que aunque no dábamos pasos de gigante, yo notaba que la motivación del equipo era cada vez mayor. En mi cabeza ya no sólo se trataba de terminar la carrera, si no de intentar marcar una diferencia en el día a día de alguna persona.

Nos pusimos manos a la obra desde que tuvimos la primera tutoría con Raquel y Alberto. Tuvimos que hacer una investigación inicial sobre aplicaciones, algoritmos, modelos... Con el objetivo de ponernos en contexto sobre lo que íbamos a hacer, también estuvimos estudiando el Trabajo de Fin de Grado anterior, viendo las entrevistas que realizaron, así como los recursos que utilizaron, ya que el objetivo era conseguir una aplicación más completa este año. Los primeros pasos fueron complicados, ya que nos costó encontrar una plataforma donde montar el servidor. En un principio quisimos usar Flask, y aunque cada uno de nosotros montó un servidor de una manera diferente, no conseguíamos avanzar ya que no conseguíamos recibir la imagen en el servidor. Por mi parte estuve investigando distintas posibilidades para poder hacerlo, pero no conseguí que funcionara ninguna de ellas. Finalmente Walid y Sirma consiguieron montarlo en *Firebase*. Una vez que teníamos el servidor, lo siguiente era empezar con el desarrollo de la aplicación.

Durante las fases iniciales del desarrollo de la aplicación, mi función principal fue avanzar con la memoria e investigar posibles funcionalidades que hicieran la descripción de imágenes más detallada. A pesar de haber cursado un par de asignaturas de inteligencia artificial, los conocimientos que tenía no me parecían aplicables a nuestra aplicación, por lo que fue como empezar de cero en un primer momento. Durante este período de tiempo, descubrí la API ColorThief, la cual nos podía aportar información relativa a los colores, que era uno de los objetivos que nos habíamos propuesto para nuestra aplicación. Poco tiempo después informándome de otras opciones a la hora de describir colores encontré la biblioteca Palette, la cual parecía más sencilla de utilizar. Tras una comparación entre las dos opciones y consultarla con los demás miembros del equipo, se decidió entre todos que lo mejor sería implementar la biblioteca Palette. Una vez implementé la biblioteca Palette, al principio puse que mostrara el color de las imágenes por consola, para comprobar su correcto funcionamiento. De por sí, el color dominante devuelto se encontraba en hexadecimal, por lo que mi siguiente movimiento fue buscar una manera de convertir un color de hexadecimal a “palabra”, para que en las descripciones pudiera decir el

color de una manera comprensible para el usuario. Para esto se encontraron diversas clases ya creadas pero que no hacían una conversión muy precisa. Finalmente Wалиd encontró la clase `ColorUtils.java`.

Durante el proceso de implementación de la detección de colores con la biblioteca `Palette`, me di cuenta de un fallo muy grande en nuestra aplicación. Este fallo lastimaba su funcionamiento y había pasado desapercibido. Cada vez que se solicitaba la descripción detallada de los *Bounding Boxes* (o la descripción breve una vez implementada la detección de colores), se realizaba un recorte de la imagen original con el tamaño del *Bounding Box* en cuestión. Esto generaba un problema de recursos, ya que el recorte tardaba en realizarse. Se creó un `HashMap` en el que almacenar las imágenes ya recortadas, cada una de las cuales se identificaba con las coordenadas del *Bounding Box*. De esta manera, el recorte solo se realizaría una vez por cada *Bounding Box*. Por lo que ahora, cuando se llama a la función de recorte, primero se comprueba si la imagen asociada a esas coordenadas ya está en el `HashMap`.

Una vez terminada la aplicación, se realizó una evaluación con las personas entrevistadas el año pasado en la que se comprobó si las personas con discapacidad visual eran capaces de percibir y hacerse una idea mental de las imágenes que se les propusieron. Tras usar la aplicación se les hicieron una serie de preguntas con las que podían evaluar nuestra aplicación en términos de usabilidad, complejidad, etc.

Bajo mi punto de vista, desde el principio el equipo tuvo una gran cohesión, y aunque ha habido momentos en los que se han tenido diferentes opiniones, siempre se ha sabido llegar a un entendimiento. Esto, sumado a la buena comunicación entre todos los integrantes del equipo, ha hecho que sea sencillo trabajar en un proyecto tan complejo.

9.4. Daniel Leo Cansado

Cuando llegó el momento de elegir el tema del Trabajo de Fin de Grado para finalizar mis estudios en ingeniería informática, tenía claro que quería hacer una aplicación móvil porque cuando estuve un año estudiando en *L'Alma Mater Studiorum - Università di Bologna*, Italia, di la asignatura de aplicaciones móviles y me despertó tal interés que ya se me instauró la idea de acabar mis estudios con una de este tipo. Mi idea sentaba su base en hacerla de manera grupal ya que considero que un grupo de personas que se conoce y tiene un objetivo similar puede lograr grandes cosas. Cuando nos pusimos a ver qué tema encajaba más con nuestras ideas personales y grupales, no llegamos a terminar de leer las opciones que había cuando vimos que había un tema que, aparte de ser aplicación móvil, tenía el fin de poder ayudar a otras personas. Fuimos directamente a comunicarle a nuestros tutores, Raquel y Alberto, que nos encantaba la idea. Queríamos formar parte del proyecto para poder materializar esas necesidades que presentan las personas con discapacidad visual.

Una vez tuvimos la primera reunión, los tutores nos pusieron en contexto sobre qué tareas ya se habían realizado en este proyecto el año anterior. Nos dieron pautas sobre qué problemas tuvieron el año pasado a la hora de la implementación. Además, nos guiaron para empezar a investigar las nuevas tecnologías que existían en el mercado. Estudiamos el trabajo de fin de grado del año pasado para poder tener un

punto de partida.

Gracias a tener conocimientos sobre *Android Studio*, una ligera noción de *Google Firebase* y los problemas que existieron el año pasado con el servidor, propuse a nuestros tutores la idea de empezar otra alternativa de servidor para poder tener más de una opción por si acaso. Por otro lado, de forma paralela se intentó montar un servidor de la misma manera que el año pasado, corrigiendo los problemas que se encontraron el año pasado. Debido a que yo tenía más conocimientos sobre *Android Studio* y *Google Firebase*, me encargué de investigar cómo poner en práctica lo que había aprendido en Italia con estas nuevas funcionalidades que necesitábamos todos aprender. Mientras mis compañeros se encargaban de intentar montar el servidor en *Flask*, mi primer desempeño fue buscar información y aprender cómo poder subir y alojar imágenes en *Google Firebase*, ya que era algo que nunca había tocado. Inicialmente, tuve que investigar qué y cómo funcionaba *Cloud Storage*.

Dados los problemas con el servidor anterior, optamos por descartar esa opción. En su lugar, ayudé a mis compañeros a adentrarse en esta nueva ruta que estábamos explorando para implementar el servidor lo más rápido posible. Esto también les permitió familiarizarse más rápidamente con los conceptos básicos.

Una vez montado el servidor, cada uno se puso a investigar cómo con la aplicación y el servidor poder implementar las funcionalidades que necesitábamos. En primer lugar, estuve investigando sobre distintas APIs, que fueron *Imaga* y *Clarifai*. Dos APIs que de haberlas usado, hubieran venido como anillo al dedo, debido a la gran satisfacción que presentaban ante nuestras necesidades: presentaban detección de objetos, etiquetado, extracción del fondo, entre otras funcionalidades. Tuvimos que descartarlas porque su utilización gratuita no era suficiente para poder llevar a cabo nuestro desempeño. Esto me permitió empezar a tener noción de cómo funcionaban las APIs, algo que era totalmente nuevo para mí y en qué consistía ese proceso de investigar y aprender para poder utilizar esa información para tus propósitos. Mientras este proceso seguía en marcha, se encontró información sobre cómo se podía utilizar APIs con *Firebase*, que es a través de las *Cloud Functions*, por lo que nos pusimos a ver cuáles podíamos empezar a probar. Mis compañeros encontraron la primera que te creaba una descripción en inglés en *Hugging Face*. Al funcionar a la perfección con *Cloud Function*, nos focalizamos en buscar en dicha página.

Probando modelos de entrenamiento que pudieran traducir descripciones de una imagen, encontré el que ahora mismo tenemos implementado ya que fue el que mejor resultado nos daba a la hora de ejecutarse. Al mismo tiempo que íbamos probando distintos modelos y distintas APIs, había que investigar cómo hacer que las personas con discapacidades visuales pudieran utilizar la aplicación móvil de forma autónoma. Por consiguiente, me ocupé de buscar información acerca de qué tecnologías utilizan ellos para poder utilizar el móvil. Gracias a las entrevistas proporcionadas del año pasado y la investigación que hice, me puse a implementar dicha funcionalidad de manera personalizada para nuestra aplicación. Estuve intentándolo sin éxito cambiar la configuración de dicha accesibilidad que en *Android* se llama *Talkback* y en *iOS* *VoiceOver*, pero encontré el problema de que dicha funcionalidad va arraigada al SO del terminal y la única solución que vi era *rootear* cada terminal que fuera a utilizar la aplicación móvil. Concluí que no había una solución factible y razonablemente segura para los dispositivos con los que utilizábamos la aplicación. También estuvimos

ayudando a los compañeros encargados del recorte de la imagen a depurar el código ya que ha sido una de las partes más complejas y nos tuvimos que poner para poder sacarlo.

Como producto de haber dado la asignatura de Ingeniería del Software tanto aquí en España como en Italia, fui el encargado de realizar parte de los diagramas para hacer más visual cómo se conectan las distintas clases del código con sus partes más relevantes. Una de las funcionalidades que presenta nuestra aplicación móvil es la capacidad de, si es un objeto distinto a una persona, decirte el color, con sus limitaciones. En primer lugar, estuvimos buscando distintos modelos y APIs que de forma legítima y correcta pudieramos utilizar e implementar. Como posible, manejamos *Cloud Vision* por la alta conectividad con las distintas herramientas de *Google* que tiene pero una vez más el coste por utilizarla fue motivo de descarte. Pero uno de mis compañeros que también estaba encargado de buscar encontró una biblioteca de Java la cual está implementada en la aplicación. Entre los dos montamos la primera versión de la clase, que hemos ido puliendo entre todos a lo largo de las distintas pruebas que hemos ido realizando.

Por otro lado y para concluir, también estuve trabajando en la depuración del modelo del género de personas. Se añadió un mensaje informativo de forma auditiva, el cual ayuda a que cuando se pulsa sobre una persona y no se ha recibido la información adicional de la persona, te informa que si pulsas otra vez obtendrás más información.

Bibliografía

- AMOR SANZ, M., CHAVES LÓPEZ, A. y RUIZ GEA, V. Desarrollo de una aplicación móvil para la generación de descripciones de imágenes para personas con discapacidad visual. 2023. Trabajo de Fin de Grado, Universidad Complutense de Madrid.
- CARION, N., MASSA, F., SYNNAEVE, G., USUNIER, N., KIRILLOV, A. y ZAGORUYKO, S. End-to-end object detection with transformers. *CoRR*, vol. ab-s/2005.12872, 2020.
- IAKUBOVSKYI, D. Ageless faces: Using vision transformer for human age detection from facial images. 2023a. [Web; accedido el 14-05-2024].
- IAKUBOVSKYI, D. Men woman face images detection. 2023b. [Web; accedido el 14-05-2024].
- LI, J., LI, D., XIONG, C. y HOI, S. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. 2022. [Web; accedido el 14-05-2024].
- TIEDEMANN, J. The Tatoeba Translation Challenge – Realistic data sets for low resource and multilingual MT. En *Proceedings of the Fifth Conference on Machine Translation*, páginas 1174–1182. Association for Computational Linguistics, Online, 2020.
- YANG, L., KANG, B., HUANG, Z., XU, X., FENG, J. y ZHAO, H. Depth anything: Unleashing the power of large-scale unlabeled data. 2024.

