Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Ingeniería del Software

Aplicación móvil para la clasificación de residuos reciclables mediante Machine Learning

Alumno:
Diego Fernández de Valderrama Domínguez

Tutores:
Diego García Álvarez

Al tiempo, que nos da peso.

Agradecimientos

Gracias a mis padres por leer una y otra vez este TFG sin entender la mitad de lo que sucede en él, y por ayudarme con sus opiniones durante el desarrollo del mismo.

Resumen

En la actualidad, el reciclaje se ha convertido en una de las principales preocupaciones ambientales debido a la creciente cantidad de residuos que se generan en todo el mundo. El aumento de la población y el consumo de recursos naturales ha llevado a una producción masiva de residuos, lo que ha generado una serie de problemas como la contaminación del aire, del agua y del suelo, la emisión de gases de efecto invernadero y la pérdida de biodiversidad.

En consecuencia, mejorar el reciclaje se ha vuelto cada vez más importante en la sociedad moderna. El reciclaje permite reducir la cantidad de residuos que se envían a los vertederos y, por lo tanto, disminuir el impacto ambiental. Asimismo, el reciclaje contribuye a la conservación de los recursos naturales, ya que se pueden reutilizar materiales como el papel, el plástico, el vidrio y el metal en la fabricación de nuevos productos.

El objetivo de este trabajo consiste en ofrecer una solución parcial a la problemática mediante el desarrollo de una aplicación para dispositivos móviles Android, que simplifique la labor de reciclaje para aquellas personas que no están familiarizadas con ella. Asimismo, la aplicación estará diseñada para adaptarse a posibles cambios en la distribución de contenedores, con el fin de ofrecer una solución siempre vigente y eliminar cualquier duda que pudiera surgir en el proceso de reciclaje.

Abstract

Nowadays, recycling has become one of the main environmental concerns due to the increasing amount of waste generated worldwide. The increase in population and consumption of natural resources has led to a massive production of waste, resulting in a number of problems such as air, water and soil pollution, emission of greenhouse gases and loss of biodiversity.

Consequently, improving recycling has become increasingly important in modern society. Recycling reduces the amount of waste sent to landfills and therefore reduces the environmental impact. Recycling also contributes to the conservation of natural resources, as materials such as paper, plastic, glass and metal can be reused in the manufacture of new products.

The aim of this work is to offer a partial solution to the problem through the development of an Android mobile application that simplifies the task of recycling for those who are not familiar with it. The application will also be designed to adapt to possible changes in the distribution of containers, in order to offer a solution that is always up to date and eliminate any doubts that the recycling process may raise.

Índice general

Lista de Figuras

Lista de Tablas

Glosario de términos

A | B | C | D | F | G | H | J | M | P | S | T

A

Agile for One

Especificación de la metodología Ágil. XIX

API

Las API son conjuntos de definiciones y protocolos que se utilizan para diseñar e integrar el software de las aplicaciones.Suele considerarse como el contrato entre el proveedor de información y el usuario, donde se establece el contenido que se necesita por parte del consumidor (la llamada) y el que requiere el productor (la respuesta). XIX

API REST

Una API de REST, o API de RESTful, es una interfaz de programación de aplicaciones (API o API web) que se ajusta a los límites de la arquitectura REST y permite la interacción con los servicios web de RESTful. XIX

B

Base de Datos Relacional

Una base de datos relacional es un tipo de base de datos que almacena y proporciona acceso a puntos de datos relacionados entre sí. Las bases de datos relacionales se basan en el modelo relacional, una forma intuitiva y directa de representar datos en tablas. En una base de datos relacional, cada fila en una tabla es un registro con una ID única, llamada clave. Las columnas de la tabla contienen los atributos de los datos y cada registro suele tener un valor para cada atributo, lo que simplifica la creación de relaciones entre los puntos de datos. XIX

BLOB

BLOB son las siglas de Binary Large Object o, en español, objeto binario grande. Es un término que se usa para almacenar un elemento grande de datos en una base de datos que está en código binario. Este código binario es legible para el software. 73, XIX

C

Captcha

(Completely Automated Public Turing test to tell Computers and Humans Apart) es un tipo de prueba diseñada para determinar si el usuario es humano o un programa de computadora automatizado. Por lo general, implica la presentación de una imagen o un desafío visual o auditivo que un humano puede resolver fácilmente pero que un programa de computadora no puede. Los Captchas se utilizan a menudo en sitios web para prevenir el spam, los ataques de bots y otras formas de actividad malintencionada.. XIX

CSV

(Comma Separated Values) es un formato de archivo de texto utilizado para almacenar y transmitir datos tabulares en forma de filas y columnas. Cada línea del archivo representa una fila de datos y los valores de cada columna están separados por comas. CSV es un formato comúnmente utilizado para intercambiar datos entre diferentes aplicaciones, sistemas y plataformas, ya que es fácil de leer y escribir con herramientas de programación y hojas de cálculo.. XIX

D

DatSet

Su traducción a nuestra lengua sería "conjunto de datos" y es una colección de datos habitualmente tabulada. XIX

Debugging

Es el proceso de encontrar y corregir errores o fallos en un programa informático. Esto implica la identificación de problemas en el código fuente, la ejecución de pruebas para reproducir y aislar el error, y la realización de ajustes necesarios para solucionarlo. El debugging es una parte importante del desarrollo de software y es esencial para garantizar la calidad y fiabilidad de un programa.. XIX

Docker

Docker es una tecnología de código abierto que permite el libre despliegue de aplicaciones, así como todo lo relacionado con ellas, en contenedores software sin importar el sistema operativo de la máquina que se esté utilizando. Los contenedores permiten crear, implementar, ejecutar, copiar y trasladar aplicaciones con facilidad. Estas aplicaciones se pueden obtener o compartir bajo el nombre de imagen. XIX

Dockerizar

Dockerizar se refiere a la implementación de Docker para empaquetar una aplicación (software), para luego distribuirla y ejecutarla a través de los contenedores. También se le conoce como contenerizar aplicaciones. XIX

F

FireBase

Firebase de Google es una plataforma en la nube para el desarrollo de aplicaciones web y móvil. Está disponible para distintas plataformas (iOS, Android y web), con lo que es más rápido trabajar en el desarrollo. XIX

Framework

Conjunto de herramientas, librerías, estándares y convenciones que se utilizan para facilitar el desarrollo de software y aplicaciones web. Proporciona una estructura básica para el desarrollo de software, lo que permite a los desarrolladores enfocarse en la lógica de la aplicación en lugar de tener que preocuparse por tareas comunes y repetitivas, como la gestión de bases de datos, la autenticación de usuarios o el manejo de solicitudes HTTP. Los frameworks son útiles para mejorar la productividad, la calidad y la escalabilidad de los proyectos de software, y pueden ser utilizados en una amplia variedad de lenguajes de programación.. XIX

G

GPU

(Graphics Processing Unit, por sus siglas en inglés) es un tipo de procesador especializado en el procesamiento de gráficos y cálculos matemáticos intensivos en paralelo. Las GPU se utilizan principalmente en aplicaciones que requieren una gran cantidad de cálculos paralelos, como la renderización de gráficos en videojuegos y películas, el aprendizaje automático y la minería de criptomonedas. Las GPU tienen una arquitectura altamente optimizada para procesamiento paralelo y están diseñadas para manejar grandes cantidades de datos simultáneamente, lo que las hace muy eficientes en la realización de tareas que requieren un alto grado de procesamiento numérico.. XIX

H

Hostear

El hosting (o alojamiento web) es el servicio que provee espacio en servidores de internet para que los sitios web puedan estar disponibles en línea. El proveedor de hosting ofrece diferentes planes y recursos, como ancho de banda, espacio en disco, correos electrónicos, bases de datos y seguridad, entre otros, para que los propietarios de los sitios web puedan publicar y mantener sus contenidos en la web de manera eficiente y segura.. XIX

J

Jupyter

Jupyter es un entorno interactivo de programación que permite la creación de documentos que combinan código ejecutable, texto explicativo, visualizaciones y otros elementos multimedia en un solo lugar. Es utilizado por científicos de datos, investigadores y programadores para desarrollar y compartir código, experimentos y análisis de datos de manera colaborativa. Jupyter soporta varios lenguajes de programación como Python, R y Julia, entre otros.. XIX

M

Machine Learning

El Machine Learning es una disciplina del campo de la Inteligencia Artificial que, a través de algoritmos, dota a los ordenadores de la capacidad de identificar patrones en datos masivos y elaborar predicciones (análisis predictivo). Este aprendizaje permite a los computadores realizar tareas específicas de forma autónoma, es decir, sin necesidad de ser programados. XIX

Mensajería Push

Es un servicio de notificación que permite a las aplicaciones móviles y sitios web enviar mensajes directamente a los dispositivos de los usuarios, incluso cuando la aplicación o sitio web no está en uso. Estos mensajes pueden incluir actualizaciones de noticias, ofertas especiales, recordatorios y otra información relevante para los usuarios. Los usuarios pueden optar por recibir notificaciones push al descargar una aplicación o visitar un sitio web y conceder permisos para recibir notificaciones.. XIX

P

POST

Es un método de petición en HTTP que trabaja en conjunto con otro método llamado GET. Estos emplean escenas de transmisión de datos a páginas, los cuales están diseñados para la lectura sin dificultad alguna. De esta forma, Tanto GET como POST son capaces de usar los mismos datos. Aunque surgen diversas diferencias según la importancia de cada uno, pero los desarrolladores web prefieren utilizar este método. XIX

Pup-Up

Pop-up es un término que se utiliza comúnmente para describir ventanas emergentes en una interfaz de usuario. Estas ventanas suelen aparecer de forma inesperada y pueden mostrar información adicional o solicitar una acción del usuario. Los pop-ups pueden ser utilizados para una variedad de propósitos, como mostrar mensajes de alerta, publicidad, formularios de registro o suscripción, entre otros. . XIX

S

SPAM

Es el envío masivo e indiscriminado de correos electrónicos no solicitados, mensajes de texto, publicaciones en redes sociales u otros medios digitales con fines publicitarios, fraudulentos o maliciosos. El objetivo del spam es llegar a la mayor cantidad de personas posible, aunque muchas veces estas personas no tienen ningún interés en los productos o servicios que se promocionan. El spam puede ser molesto e invasivo, y en algunos casos puede contener enlaces a sitios web maliciosos o intentar robar información personal.. XIX

T

TPU

TPU son las siglas en inglés de "Tensor Processing Unit", que se traduce al español como "Unidad de Procesamiento Tensorial". Es un tipo de procesador especializado diseñado por Google para acelerar el entrenamiento de redes neuronales y otros algoritmos de aprendizaje profundo en su plataforma de computación en la nube, Google Cloud. Los TPUs son particularmente efectivos para trabajar con grandes conjuntos de datos y redes neuronales profundas debido a su capacidad para realizar cálculos matemáticos de alta precisión con gran velocidad y eficiencia energética.. XIX

Siglas

D | H | S

D

DB

Base de Datos. XIX

H

HDD

Disco Duro. XIX

S

SO

Sistema Operativo. XIX

Capítulo 1

Introducción

1.1. Contexto

El presente Trabajo de Fin de Grado de la titulación de Grado en Ingeniería del Informática(Mención en Ingeniería del Software) de la escuela de Ingeniería Informática de la Universidad de Valladolid, se centra en buscar una salida a la falta de concienciación por parte de la población respecto al correcto reciclaje.

La contaminación y la necesidad de una mejor gestión de los residuos que se generan en la vida diaria es un problema del cual poco a poco la sociedad se ha ido haciendo más consciente. Ante esto han ido surgiendo soluciones parciales a estos problemas. Una de ellas es el uso eficaz del reciclado en el que, en una primera fase, los residuos son desechados en distintos compartimentos/contenedores que permiten y facilitan el trabajo para su uso o destrucción, de modo que dependiendo de la materia que compone el producto se destina a uno u otro tipo de contenedor: vidrio, papel, orgánico, plásticos u otros residuos, entre otros.

La mayor sensibilización, concienciación y necesidad de resolver el problema que genera las basuras ha aumentado la complejidad de esta labor de clasificación, por la presencia de una mayor variedad de contenedores. Algunos ciudadanos han manifestado confusión respecto al sistema de reciclaje. Con el objetivo de optimizar, agilizar y simplificar el proceso de clasificación de residuos, y de este modo, evitar confusiones derivadas del desconocimiento del contenedor adecuado para algunos desechos, se ha propuesto el desarrollo de una aplicación móvil fácil de manejar basada en imágenes y técnicas de *Machine Learning*. Dicha herramienta tendría como finalidad facilitar la correcta clasificación de los residuos por parte de los ciudadanos, contribuyendo a que el proceso de reciclaje pueda llevarse a cabo de manera comprensible y accesible para todos.

1.2. Objetivos

Dentro de este contexto, los objetivos en el desarrollo de una aplicación de clasificación de basuras buscan abordar la problemática de la acumulación y gestión inadecuada de residuos, los cuales representan una amenaza constante para el medio ambiente y la salud pública. En este sentido, el objetivo principal de la aplicación es fomentar una cultura de reciclaje y cuidado del medio ambiente, brindando una herramienta efectiva que facilite la clasificación de los residuos en diferentes categorías, como papel, plástico, vidrio y orgánicos, entre otros.

He aquí una lista mas detallada de estos:

1. Mejorar la eficiencia y precisión en la separación de residuos mediante el uso de tecnologías avanzadas como la visión por computadora y el aprendizaje automático.
2. Concienciar a los usuarios sobre la importancia del reciclaje y la adecuada disposición de los residuos, mediante la implementación de una interfaz amigable y educativa.
3. Contribuir al desarrollo sostenible y a la reducción de la huella de carbono, incentivando la separación y el reciclaje de los residuos.
4. Entrenamiento de una red neuronal basada en *Machine Learning Visual* que permita la clasificación automática de los residuos en diferentes categorías a partir de imágenes capturadas por la cámara del dispositivo móvil.
5. Desarrollo de este modelo en una aplicación móvil, para facilitar su uso a los usuarios.
6. Recolección de opiniones y datos por parte de los usuarios, para poder mejorar futuras experiencias.
7. Facilitar la investigación en el campo de la visión por computadora y el aprendizaje automático aplicado al procesamiento de imágenes de residuos.
8. Proporcionar una plataforma para la recolección y el análisis de datos sobre la cantidad y el tipo de residuos generados por los usuarios, lo que puede ser utilizado para estudios y análisis de impacto ambiental.
9. Brindar una herramienta educativa para la enseñanza de la importancia del reciclaje y la adecuada gestión de residuos en el ámbito académico y de investigación.
10. Fomentar la innovación y el desarrollo de soluciones tecnológicas que contribuyan al desarrollo sostenible y la protección del medio ambiente.

1.3. Estructura de la memoria

Este documento se estructura de la siguiente forma:

Capítulo 1 Introducción: El presente Capítulo.

Capítulo 2 Requisitos y planificación: En este capítulo se describen los requisitos y objetivos del proyecto, así como la planificación y metodología a seguir para el desarrollo del proyecto.

Capítulo 3 Análisis: En este capítulo se realizará un análisis exhaustivo desde una perspectiva de ingeniería de software de las tecnologías empleadas con *Machine Learning Visual*, API REST y aplicaciones móviles. Se evaluarán los diferentes métodos y herramientas disponibles para su implementación y se identificarán las mejores prácticas en cuanto a arquitectura, integración y diseño de software.

Capítulo 4 Tecnologías utilizadas: En este capítulo se describirán las tecnologías y herramientas utilizadas para el desarrollo del proyecto, incluyendo el lenguaje de programación, la plataforma y las bibliotecas utilizadas.

Capítulo 5 Diseño: En este capítulo se presentará el diseño detallado de la solución propuesta, incluyendo la arquitectura de software, la integración de las diferentes tecnologías y la descripción de cómo se implementará la API REST y la aplicación móvil.

Capítulo 6 Implementación y pruebas: En este capítulo se describirá la implementación del proyecto, incluyendo la programación y los casos de pruebas para verificar el correcto funcionamiento de la API REST y la aplicación móvil.

Capítulo 7 Seguimiento del proyecto: En este capítulo se describirán los diferentes procesos de seguimiento y control de calidad del proyecto, incluyendo la revisión y mejora continua del código y la documentación.

Capítulo 8 Conclusiones: En este capítulo se presentarán las conclusiones y resultados obtenidos durante el desarrollo del proyecto, así como las recomendaciones y posibles mejoras para futuros desarrollos.

Anexo A Manuales: En este anexo se incluirán los manuales de usuario y técnico necesarios para el uso y desarrollo del proyecto.

Anexo B Resumen de enlaces adicionales: En este anexo se incluirán los enlaces a los recursos adicionales utilizados durante el desarrollo del proyecto, incluyendo tutoriales, documentación y otros materiales relevantes.

Capítulo 2

Motivación y Estado del Tema

2.1. Introducción

La clasificación de basura es un proceso importante que ayuda a reducir la cantidad de residuos que van a los vertederos, lo que a su vez reduce la contaminación del medio ambiente. Con el uso de la tecnología de *Machine Learning Visual*, se ha logrado mejorar la eficiencia de este proceso mediante la automatización de la clasificación de los diferentes tipos de basura. En este documento, se discutirá el estado actual de las aplicaciones de clasificación de basura mediante *Machine Learning Visual*, incluyendo los desafíos actuales y las oportunidades futuras.

2.2. Motivación

Existen varias motivaciones principales para crear una aplicación móvil que permita a la gente clasificar tipos de residuos y saber en qué contenedor deben ir. Algunas de estas motivaciones son:

- **Conciencia ambiental:** Una de las motivaciones principales para crear esta aplicación es fomentar la conciencia ambiental en la población. Al informar a las personas sobre cómo clasificar adecuadamente sus residuos, se puede reducir el impacto ambiental y promover prácticas sostenibles.
- **Reducción de residuos:** Al clasificar correctamente los residuos, se puede reducir la cantidad de desechos que terminan en vertederos y que pueden contaminar el medio ambiente. Al utilizar la aplicación, las personas pueden aprender a separar sus residuos y así contribuir a la reducción de residuos.

- **Cumplimiento de regulaciones:** En algunos países, existen regulaciones y leyes que exigen que los residuos sean clasificados adecuadamente antes de su disposición. Por lo tanto, la aplicación puede ser una herramienta útil para ayudar a las personas a cumplir con estas regulaciones.
- **Facilitar el proceso de clasificación:** La clasificación adecuada de residuos puede ser confusa y abrumadora para muchas personas. Al proporcionar una aplicación que simplifica el proceso, se puede fomentar una mayor participación y ayudar a las personas a clasificar correctamente sus residuos.

2.3. Estado del Tema

En la actualidad, la disponibilidad de una aplicación móvil que ofrezca al usuario funcionalidades similares a las planteadas por este proyecto es muy limitada. Por lo tanto, para investigar opciones similares, se requiere una mayor flexibilidad, focalizando en cualquier tipo de aplicación de *Machine Learning* enfocada en el reciclaje, en lugar de centrarse exclusivamente en una aplicación móvil. A continuación se presentan algunos ejemplos:

- **Green Machine:** ha desarrollado un algoritmo de aprendizaje automático que utiliza cámaras para escanear materiales plásticos a medida que bajan por una cinta transportadora. El algoritmo identifica el tipo de plástico y envía señales a chorros de aire para que soplen el plástico y lo depositen en el contenedor adecuado para su reciclaje.
- **AMP Robotics:** ha desarrollado un robot que utiliza el aprendizaje automático para identificar y retirar materiales contaminados o no reciclables de un flujo de reciclaje. El robot utiliza cámaras para escanear los materiales y un algoritmo de aprendizaje automático para determinar si el artículo es reciclable o no. A continuación, el robot utiliza una ventosa para retirar el artículo no reciclable de la cinta transportadora.
- **TrashBotZero de CleanRobotics:** Este robot está diseñado para clasificar y separar los materiales reciclables de los residuos de los vertederos mediante sensores avanzados, cámaras y algoritmos de aprendizaje automático. De este modo, TrashBotZero puede ayudar a reducir la contaminación en los flujos de reciclaje, aumentar las tasas de reciclaje y promover un medio ambiente más limpio. Esta tecnología ya se ha implantado en diversas instalaciones, como aeropuertos, centros de convenciones y campus universitarios.

2.4. Desafíos Actuales

A pesar de los avances logrados en la clasificación de basura mediante *Machine Learning Visual*, todavía existen algunos desafíos que deben superarse. Uno de los mayores desafíos es la variabilidad en la calidad de las imágenes de basura proporcionadas. Las imágenes de basura pueden estar borrosas, mal iluminadas o tener un fondo complejo, lo que puede dificultar la identificación de los diferentes tipos de basura.

Otro desafío importante es la necesidad de datos de entrenamiento de alta calidad y cantidad suficiente. Los modelos de *Machine Learning* requieren grandes conjuntos de datos para entrenar de manera efectiva, y obtener estos datos puede ser un proceso costoso en tiempo.

Capítulo 3

Conceptos generales del Aprendizaje Automático

3.1. ¿Qué es el Aprendizaje Automático o *Machine Learning*?

El *Machine Learning* es un subcampo de la Inteligencia Artificial que se ocupa del diseño y desarrollo de algoritmos y modelos que permiten a las máquinas aprender de los datos y mejorar su rendimiento en tareas específicas con el tiempo, sin ser programadas explícitamente para cada tarea. El objetivo del *Machine Learning* es desarrollar sistemas capaces de tomar decisiones, hacer predicciones o identificar patrones en los datos sin intervención humana.

Algunos de los tipos de *Machine Learning* son:

- **Aprendizaje supervisado:** se proporciona al modelo una serie de ejemplos de entrada y salida esperada, para que pueda aprender a relacionarlos y hacer predicciones en nuevos datos similares.
- **Aprendizaje no supervisado:** el modelo se entrena con datos no etiquetados, para encontrar patrones, estructuras o grupos en los datos.
- **Aprendizaje por refuerzo:** el modelo aprende a través de la interacción con un entorno, recibiendo recompensas o castigos por sus acciones.

3.2. ¿Qué son las Redes Neuronales Artificiales?

Las redes neuronales son un tipo de modelo de *Machine Learning* inspirado en el cerebro humano y su capacidad para procesar información. Las redes neuronales se componen de capas de neuronas artificiales, que reciben entradas, procesan información y generan una salida.

Cada neurona se conecta a otras neuronas de la capa anterior o siguiente mediante conexiones llamadas pesos, que se ajustan durante el entrenamiento para mejorar el rendimiento del modelo. Las redes neuronales se pueden utilizar para una variedad de tareas, como reconocimiento de voz, visión por computadora, procesamiento del lenguaje natural, entre otras.

3.2.1. Neurona

Una neurona es la unidad básica de procesamiento en las redes neuronales y está diseñada para imitar el comportamiento de las neuronas biológicas que se encuentran en el cerebro. En una red neuronal, una neurona recibe entradas, las procesa y produce una salida.

La estructura de una neurona biológica consta de tres partes principales: las dendritas, el cuerpo celular y el axón. Las dendritas son las ramificaciones que se extienden desde el cuerpo celular y reciben las entradas de otras neuronas o del entorno. El cuerpo celular es la parte central de la neurona, donde se lleva a cabo el procesamiento de la información. El axón es una extensión larga y delgada que transporta la salida de la neurona a otras neuronas o al entorno.

Cada entrada que llega a una neurona tiene asociado un peso, que representa la fuerza de la conexión entre la neurona de entrada y la neurona de procesamiento. El procesamiento de la información en la neurona se lleva a cabo mediante una función de activación, que combina las entradas ponderadas con los pesos y produce una salida.

Existen diferentes tipos de funciones de activación, siendo una de ellas la función sigmoide, que comprime la salida de la neurona en un rango entre 0 y 1. Otras funciones de activación comunes incluyen la función tangente hiperbólica, que comprime la salida en un rango entre -1 y 1, y la función ReLU (Rectified Linear Unit), que es una función lineal simple que se utiliza para la activación de las redes neuronales profundas.

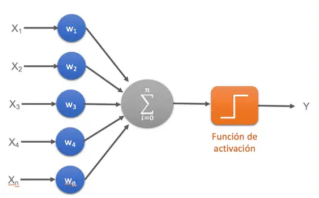


Figura 3.1: Ejemplo de Neurona - Fuente: [1]

El proceso de entrenamiento de una red neuronal implica ajustar los pesos de las conexiones entre las neuronas para minimizar una función de pérdida, que mide la diferencia entre la salida de la red y la salida deseada. El algoritmo de entrenamiento más común es el descenso de gradiente, que utiliza la derivada de la función de pérdida con respecto a los pesos para actualizar los pesos en la dirección que minimiza la pérdida.

3.2.2. Capas de la red

Una capa es un conjunto de neuronas o unidades que trabajan juntas para procesar la información de entrada y producir una salida. En general, una red neuronal está formada por múltiples capas, cada una de las cuales realiza una tarea específica en el procesamiento de la información.

La forma en que funciona una capa de la red depende del tipo de capa y su configuración. Estas operaciones matemáticas se llevan a cabo utilizando pesos y sesgos que se ajustan durante el entrenamiento de la red.

Existen varios tipos de capas que se utilizan comúnmente en las redes neuronales. A continuación, se describen algunos de ellos desde la perspectiva de la estructura de la red:

1. **Capa de entrada:** Esta capa es la primera capa de la red y recibe la entrada. En el caso de las redes neuronales para el procesamiento de imágenes, la entrada puede ser una imagen, y en el caso de las redes neuronales para el procesamiento de texto, la entrada puede ser una secuencia de palabras. Cada unidad en la capa de entrada representa una característica de la entrada.

2. **Capa oculta:** Esta capa procesa la entrada recibida de la capa anterior y produce una salida que se utiliza como entrada para la siguiente capa. Puede haber varias capas ocultas en una red neuronal, y cada capa puede tener un número diferente de neuronas y una configuración diferente. En general, se utilizan funciones de activación no lineales en las unidades de la capa oculta para introducir no linealidades en la red y permitir una mayor capacidad de modelado.

3. **Capa de salida:** Esta capa produce la salida final de la red. En el caso de las redes neuronales para el procesamiento de imágenes, la salida puede ser una clasificación de la imagen, y en el caso de las redes neuronales para el procesamiento de texto, la salida puede ser una clasificación de la secuencia de palabras. El número de unidades en la capa de salida depende del tipo de problema que se esté resolviendo. En este TFG, la capa de salida de la red tendría un número de unidades igual al número de clases de basura que se quieren clasificar.

4. **Capa de convolución:** Esta capa se utiliza comúnmente en las redes neuronales convolucionales (CNN) para el procesamiento de imágenes. La capa de convolución aplica filtros a la entrada y produce una salida convolucionada. Los filtros se ajustan durante el entrenamiento para detectar características específicas en la entrada. (ver 3.2.4 Operación de convolución)

5. **Capa recurrente:** Esta capa se utiliza comúnmente en las redes neuronales recurrentes (RNN) para el procesamiento de secuencias de tiempo, como el procesamiento de lenguaje natural. La capa recurrente permite que la red tenga memoria a largo plazo al mantener un estado oculto que se actualiza en cada paso de tiempo.

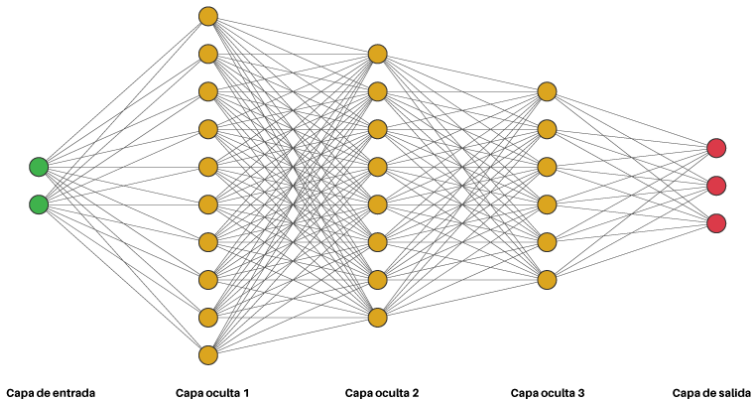


Figura 3.2: Ejemplo de Capas en una Red Neuronal - Fuente: [2]

En resumen, una capa en una red neuronal es un conjunto de neuronas que procesan la información de entrada y producen una salida. Las capas pueden ser de diferentes tipos y tienen diferentes configuraciones dependiendo del problema que se esté resolviendo.

3.2.3. Funciones de transferencia o activación

Una función de transferencia es una función matemática que se aplica a la salida de cada neurona en una capa de la red neuronal. El objetivo de una función de transferencia es generar una activación en función de la entrada en la salida de cada neurona, lo que permite a la red neuronal modelar funciones más complejas y no lineales.

Una función de transferencia toma como entrada la suma ponderada de las entradas a la neurona y el sesgo (bias) asociado con la neurona, y produce una salida.

La función de transferencia típicamente tiene un rango limitado de valores de salida, lo que significa que la salida se acota dentro de un rango específico. Esto es importante porque limita la magnitud de la salida de cada neurona, lo que puede ayudar a prevenir la divergencia durante el entrenamiento de la red neuronal.

Existen diferentes tipos de funciones de transferencia utilizadas en las redes neuronales. Algunos de los más comunes son:

- **Función de transferencia sigmoide:** Esta función, derivable en todo su dominio, produce una salida suave y no lineal que tiene un rango limitado de valores. La función sigmoide se utiliza a menudo en redes neuronales para clasificación binaria.

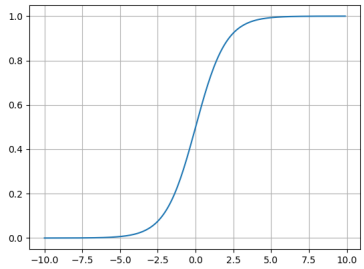


Figura 3.3: Gráfica función de activación Sigmoide

- **Función de transferencia ReLU (Rectified Linear Unit):** Esta función produce una salida que es cero si la entrada es negativa, y la entrada misma si es positiva. La función ReLU se utiliza a menudo en redes neuronales para tareas de clasificación.

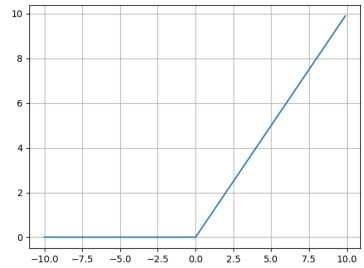


Figura 3.4: Gráfica función de activación ReLU

- **Función de transferencia softmax:** Esta función se utiliza en redes neuronales para la clasificación multiclase. La función softmax toma un vector de entrada y calcula la exponencial de cada valor en el vector. A continuación, normaliza la salida dividiendo cada valor exponencial por la suma de todos los valores exponenciales. La salida final es un vector de la misma dimensión que la entrada original, pero en el que cada valor representa la probabilidad de que ese valor sea la clase correcta.

Fuente: [8]

En resumen, la función de transferencia es una herramienta esencial en el modelado de redes neuronales porque permite a la salida de las neuronas modelar funciones complejas. La elección de la función de transferencia adecuada puede ser crítica para el rendimiento de la red en una tarea específica.

3.2.4. Operación de convolución

La operación de convolución es una técnica matemática utilizada en el procesamiento de señales, imágenes y otros tipos de datos que tienen una estructura similar. Se utiliza principalmente en redes neuronales convolucionales (CNN), que son un tipo de arquitectura de red neuronal especializada en procesamiento de imágenes y videos.

La convolución en una red neuronal es una operación matemática que aplica un filtro (también llamado *kernel*) a una imagen de entrada con el objetivo de extraer características importantes. El filtro es una pequeña matriz de números que se desliza sobre la imagen de entrada, multiplicando cada elemento del filtro por el valor correspondiente de la imagen de entrada y sumando los resultados. El resultado de esta operación es un valor que se coloca en una nueva matriz, llamada mapa de características o *feature map*.

El proceso de convolución se puede entender mejor a través de los siguientes pasos:

1. **Selección del filtro:** El filtro es una matriz de números que se utiliza para extraer características específicas de la imagen de entrada. El tamaño del filtro se determina por el usuario y se elige en función de las características que se desean extraer.
2. **Deslizamiento del filtro:** El filtro se desliza sobre la imagen de entrada, comenzando generalmente desde la esquina superior izquierda de la imagen y moviéndose de izquierda a derecha y de arriba a abajo. En cada paso, el filtro se superpone con una pequeña sección de la imagen de entrada y se realiza la operación de multiplicación y suma.

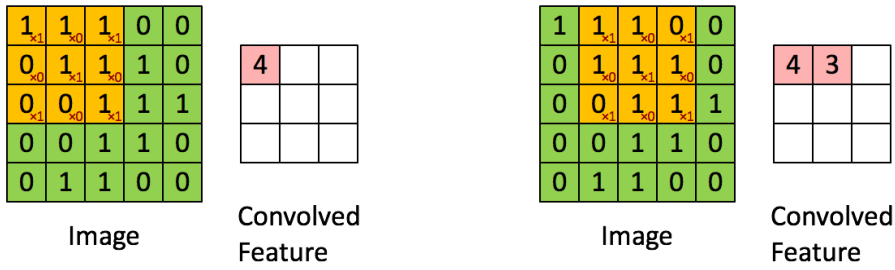


Figura 3.5: Convoluting a 5x5x1 image with a 3x3x1 kernel - Fuente: [3]

3. **Aplicación de la función de activación:** Después de realizar la operación de convolución, se aplica una función de activación a los valores resultantes. Esta función introduce no linealidades en la red neuronal, lo que aumenta su capacidad de modelar funciones complejas.
4. **Submuestreo o *pooling*:** Una vez que se han generado los mapas de características, se utiliza una técnica de submuestreo para reducir su tamaño y, por lo tanto, disminuir el coste computacional de la red. La técnica de submuestreo más común es el pooling, que toma una ventana (por ejemplo, 2x2) y selecciona el valor máximo o promedio de

los elementos en la ventana. Esto reduce el tamaño de la matriz de características, pero mantiene las características más importantes.

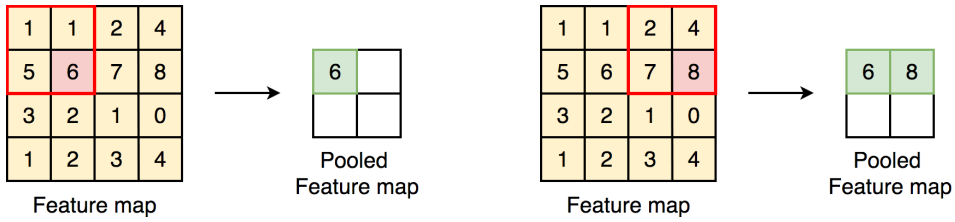


Figura 3.6: Ejemplo de Max Pooling - Fuente: [4]

5. **Repetición del proceso:** El proceso de convolución se repite varias veces, utilizando diferentes filtros en cada iteración. Cada iteración extrae características más complejas de la imagen de entrada. En este sentido, se va añadiendo capas y cada capa utiliza la salida de la capa anterior como entrada, de manera que las características extraídas se vuelven más complejas en cada capa sucesiva.

Filtros

La determinación de los filtros o *kernels* que se utilizan en las diferentes iteraciones del proceso de convolución es una tarea importante en el diseño de redes neuronales. La elección de los filtros y su estructura se basa en una combinación de conocimiento previo del problema que se está resolviendo y técnicas de optimización.

En general, los filtros utilizados en una red neuronal se aprenden a partir de los datos mediante el entrenamiento de la red. Durante el entrenamiento, la red ajusta los pesos de los filtros para minimizar la función de pérdida, que mide la diferencia entre la salida de la red y la salida deseada para un conjunto de datos de entrenamiento dado.

En el caso de las redes neuronales convolucionales, los filtros se aplican a las salidas de cada capa, y las capas más profundas de la red utilizan filtros más complejos que se construyen a partir de combinaciones de filtros más simples en capas anteriores. Cada capa extrae características más complejas de la imagen de entrada, y la red completa puede aprender a reconocer patrones y características cada vez más sofisticados en la imagen.

3.2.5. Entrenamiento

El entrenamiento de una red neuronal implica ajustar los pesos y las conexiones entre las neuronas de la red de manera que la salida de la red sea lo más cercana posible a la salida deseada para un conjunto de datos de entrada. El proceso de entrenamiento se realiza en varias iteraciones, llamadas épocas, y consiste en los siguientes pasos:

1. **Inicialización de los pesos:** Los pesos se inicializan aleatoriamente antes del entrenamiento.
2. **Propagación hacia adelante:** Los datos de entrada se pasan a través de la red neuronal, capa por capa, calculando la salida de cada neurona.
3. **Cálculo de la función de pérdida:** La función de pérdida se utiliza para calcular la diferencia entre la salida de la red y la salida deseada para los datos de entrenamiento.
4. **Propagación hacia atrás:** Los errores se propagan hacia atrás a través de la red, capa por capa, utilizando el algoritmo de retropropagación para ajustar los pesos de la red.
5. **Actualización de pesos:** Los pesos se actualizan utilizando un algoritmo de optimización, como el descenso de gradiente, que minimiza la función de pérdida.
6. **Repetición:** Los pasos 2 a 5 se repiten para un número predeterminado de épocas o hasta que la función de pérdida alcance un valor mínimo.

Durante el entrenamiento, la red neuronal ajusta los pesos y las conexiones entre las neuronas para minimizar la función de pérdida y mejorar su capacidad para realizar tareas específicas. Después del entrenamiento, la red puede utilizarse para predecir la salida de datos nuevos o desconocidos.

3.3. Hiperparámetros del entrenamiento

3.3.1. Optimizador

En el contexto de las redes neuronales, un optimizador es un algoritmo utilizado para ajustar los pesos y sesgos de las conexiones entre las neuronas de una red neuronal, con el objetivo de minimizar una función de pérdida. La función de pérdida es una medida que cuantifica la diferencia entre la salida predicha por la red neuronal y la salida deseada para un conjunto de datos de entrenamiento.

El proceso de entrenamiento de una red neuronal implica encontrar los valores óptimos de los pesos y sesgos de las conexiones entre las neuronas. Esto se logra mediante un proceso iterativo llamado descenso de gradiente, que implica calcular la derivada de la función de pérdida con respecto a los pesos y sesgos, y ajustar los valores de los pesos y sesgos en la dirección opuesta al gradiente de la función de pérdida.

El descenso de gradiente puede ser muy ineficiente en grandes conjuntos de datos, ya que puede llevar a una convergencia muy lenta o incluso a un estancamiento en mínimos locales subóptimos. Es por eso que se utilizan optimizadores que agregan modificaciones al proceso de descenso de gradiente para mejorar la velocidad y calidad de la convergencia.

Algunos optimizadores populares utilizados en el entrenamiento de redes neuronales son:

- **Gradiente descendente (GD):** Es el optimizador más básico, que simplemente actualiza los pesos y sesgos en la dirección opuesta al gradiente de la función de pérdida. Este enfoque puede ser ineficiente en grandes conjuntos de datos.
- **Gradiente descendente estocástico (SGD):** Utiliza pequeñas muestras de datos (lotes) en lugar de todo el conjunto de datos, lo que lo hace más eficiente que GD.
- **Adagrad:** Ajusta automáticamente la tasa de aprendizaje para cada peso y sesgo, utilizando un historial acumulado de los gradientes para cada uno. Es útil para problemas en los que los gradientes varían ampliamente en magnitud.
- **Adadelta:** Similar a Adagrad, pero utiliza un promedio móvil del historial acumulado de los gradientes para adaptar la tasa de aprendizaje en lugar de utilizar una suma acumulada. Es más robusto que Adagrad para problemas con gradientes ruidosos.
- **Adam:** Combina las ideas de Adagrad y RMSProp para adaptar la tasa de aprendizaje para cada peso y sesgo, utilizando estimaciones del primer y segundo momento de los gradientes. Es uno de los optimizadores más populares y eficientes en la actualidad.
- **AdamW:** Es una variante de Adam que utiliza una regularización de peso de decaimiento de peso para evitar el sobreajuste durante el entrenamiento.
- **RMSProp:** Utiliza una media móvil ponderada de los gradientes para adaptar la tasa de aprendizaje de cada peso y sesgo. Es útil para problemas en los que los gradientes varían ampliamente en magnitud y dirección.
- **Momentum:** Agrega un término de momento a la actualización de los pesos y sesgos, lo que acelera el proceso de entrenamiento y puede ayudar a evitar mínimos locales subóptimos.
- **Nesterov Accelerated Gradient (NAG):** Una variante de Momentum que utiliza una estimación del siguiente paso de los pesos y sesgos para evaluar el gradiente en lugar de la estimación actual.

Hay muchos otros optimizadores disponibles, cada uno con sus propias ventajas y desventajas. En general, el uso de un optimizador adecuado puede mejorar significativamente la velocidad y calidad del proceso de entrenamiento de una red neuronal, lo que se traduce en una mejor capacidad de generalización y rendimiento en el conjunto de datos de prueba.

3.3.2. Épocas

Un *epoch* o época es una pasada completa a través de todo el conjunto de datos de entrenamiento durante el proceso de entrenamiento de un modelo de aprendizaje automático. Durante una época, el algoritmo de aprendizaje automático utiliza los datos de entrenamiento para ajustar los pesos y los parámetros del modelo, con el objetivo de mejorar su capacidad para hacer predicciones precisas.

En otras palabras, una época se refiere a una iteración completa de entrenamiento del modelo, en la que se utilizan todos los datos de entrenamiento disponibles para ajustar el

modelo. Después de cada época, el modelo se evalúa en un conjunto de datos de validación para medir su capacidad para generalizar y hacer predicciones precisas en datos nuevos.

El número de épocas que se necesitan para entrenar un modelo depende del tamaño del conjunto de datos de entrenamiento, la complejidad del modelo y otros factores. En general, se recomienda ajustar el número de épocas durante el proceso de entrenamiento para encontrar un equilibrio entre la precisión del modelo y el tiempo de entrenamiento. Un numero excesivo de estas sobre el mismo *DataSet* puede dar lugar a Overfitting (ver 3.4.2).

3.3.3. Batch (lote)

Los datos de entrenamiento siempre se dividen en pequeños lotes para superar el problema que podría surgir debido a las limitaciones de espacio de almacenamiento de un sistema informático. Estos lotes más pequeños pueden introducirse fácilmente en el modelo de aprendizaje automático para entrenarlo. Este proceso de dividirlo en partes más pequeñas se denomina *Batch* en el aprendizaje automático. Este procedimiento se conoce como época, cuando todos los lotes se introducen en el modelo para entrenarlo a la vez.

3.3.4. Número de neuronas por capa

Este hiperparámetro se establece antes de que se entrene la red y puede afectar significativamente el rendimiento y la eficiencia de la red. En general, una mayor cantidad de neuronas por capa aumentará la capacidad de la red para modelar datos complejos y puede mejorar el rendimiento en conjuntos de datos complejos, pero también puede hacer que la red sea más lenta y difícil de entrenar.

3.3.5. Número de capas

Al igual que con el número de neuronas por capa, el número de capas puede tener un impacto significativo en el rendimiento de la red. En general, una red neuronal más profunda (con más capas) puede aprender representaciones más complejas de los datos de entrada y puede ser más eficaz en la identificación de patrones y relaciones en los datos. Sin embargo, una red neuronal más profunda también puede ser más difícil de entrenar y puede ser propensa a problemas como el Overfitting (ver 3.4.2).

3.4. Otros conceptos

3.4.1. Data Augmentation

Los modelos de *Machine Learning* pueden requerir millones de parámetros (Los cuales no suelen estar disponibles). La técnica de *Data Augmentation* (aumento de datos) se utiliza para

aumentar la cantidad de datos de entrenamiento disponibles para un modelo, creando nuevas instancias de datos a partir de los datos existentes. Esto se logra mediante la aplicación de transformaciones a los datos de entrenamiento existentes, como rotaciones, desplazamientos, cambios de escala y recorte, entre otras.

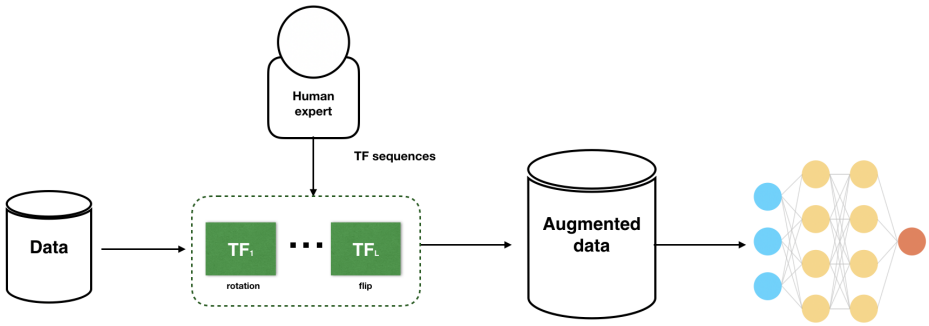


Figura 3.7: Heuristic data augmentations apply a deterministic sequence of transformation functions tuned by human experts. The augmented data will be used for training downstream models - Fuente: [5]

En el caso de los modelos de clasificación de imágenes, estas son algunas de las técnicas empleadas:

- Padding.
- Rotación aleatoria.
- Re-escalado.
- Giro horizontal y vertical.
- Traslación (la imagen se desplaza a lo largo de las direcciones X, Y).
- Recortar.
- Zoom.
- Modificar los colores.
- Cambiar el contraste.
- Añadir ruido.
- Forzar una perspectiva.

Es importante destacar que esta solo es una breve lista de la multitud de técnicas que existen para esto.

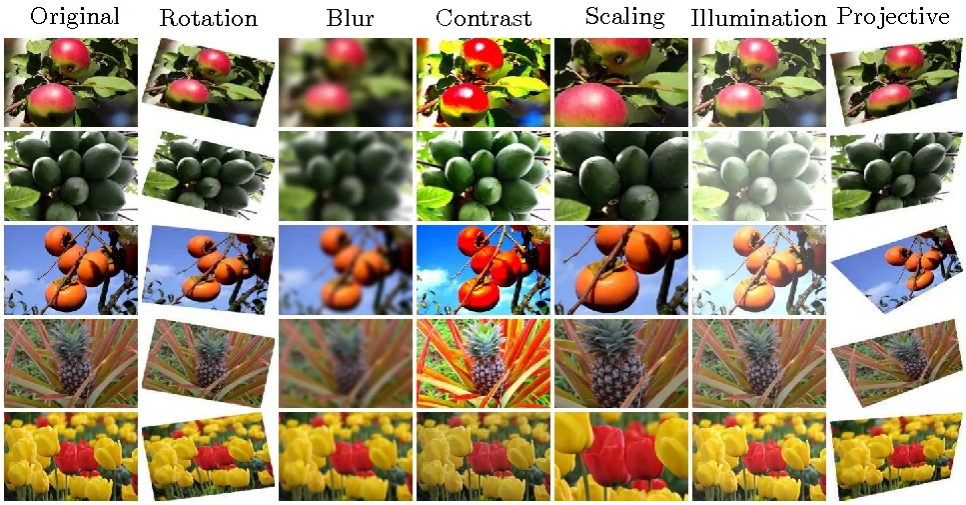


Figura 3.8: Image Data Augmentation techniques - Fuente: [6]

3.4.2. Overfitting

El sobreajuste o overfitting [9] es un problema común que puede ocurrir durante el entrenamiento de una red neuronal para clasificar imágenes. Se produce cuando la red neuronal se ajusta demasiado a los datos de entrenamiento y pierde su capacidad para generalizar y hacer predicciones precisas sobre nuevos datos.

En otras palabras, la red neuronal memoriza los ejemplos de entrenamiento en lugar de aprender a reconocer patrones generales en los datos. Esto puede llevar a una precisión baja en los datos de prueba y una precisión alta en los datos de entrenamiento.

3.4.3. Sesgos (Bias)

Los sesgos (o *bias*, en inglés) se refieren a las distorsiones sistemáticas en el proceso de aprendizaje de un modelo, que pueden llevar a la toma de decisiones incorrectas o injustas.

Estas distorsiones pueden ser introducidas por diversos factores, incluyendo la selección de características (features) o atributos del conjunto de datos de entrenamiento, la elección del algoritmo de aprendizaje automático, la cantidad y calidad de los datos de entrenamiento, y los supuestos subyacentes en el modelo.

Es importante tener en cuenta que los sesgos no son necesariamente malos en sí mismos. De hecho, los sesgos son un elemento esencial en el aprendizaje humano y en la toma de decisiones, y pueden ser útiles en ciertos contextos. Sin embargo, cuando se trata de machine learning, es importante ser consciente de los posibles sesgos y trabajar para minimizarlos para asegurar que los modelos sean precisos en su toma de decisiones.

Capítulo 4

Planificación

4.1. Introducción

La planificación de proyectos informáticos es un proceso sistemático que implica la definición de objetivos, la identificación de requisitos, la asignación de recursos y la definición de un plan de acción para lograr los objetivos del proyecto en un plazo determinado. De entre las múltiples opciones que se plantean a la hora de llevar un proyecto de este estilo, se ha decidido emplear el marco de trabajo *Agile for one*.

Hay varias razones por las que un marco de trabajo ágil como *Agile for one* puede ser una buena opción para un TFG de Ingeniería Informática:

- **Flexibilidad:** Al permitir una mayor flexibilidad en la aplicación de los valores y principios ágiles, *Agile for one* permite a los equipos adaptarse a los requisitos y circunstancias únicos de cada proyecto, incluyendo un TFG.
- **Enfoque en la entrega:** *Agile for one* se enfoca en la entrega continua de productos funcionales, lo que puede ser especialmente útil para un TFG donde se requiere demostrar el progreso y los resultados a lo largo del tiempo.
- **Mejora continua:** La filosofía de mejora continua de *Agile for one* permite a los equipos revisar y mejorar constantemente su proceso, lo que puede ser útil en un TFG donde se requiere una evolución constante y una optimización del producto final.

Agile for one puede ofrecer un enfoque flexible, enfocado en la entrega, colaborativo y orientado a la mejora continua que puede ser especialmente útil en un TFG de Ingeniería del Software.

4.2. Planificación

En este Trabajo de Fin de Grado (TFG), el Diagrama de Gantt se ha empleado para organizar y planificar el trabajo necesario para completar el proyecto. En este caso, el Diagrama de Gantt se ha utilizado para identificar las diferentes fases del proyecto, establecer plazos para cada fase y asignar responsabilidades a los miembros del equipo.

En el TFG en cuestión, el Diagrama de Gantt se ha empleado para planificar las diferentes fases del proyecto, que incluyen introducción, gestión del proyecto, su desarrollo y la redacción del informe final entre muchos otros. Cada una de estas fases se ha dividido en sub-tareas más pequeñas y se ha establecido un plazo para cada una.

A continuación se muestra el Diagrama de Gantt del presente proyecto (Ver. 4.1 Diagrama de Gantt del proyecto).

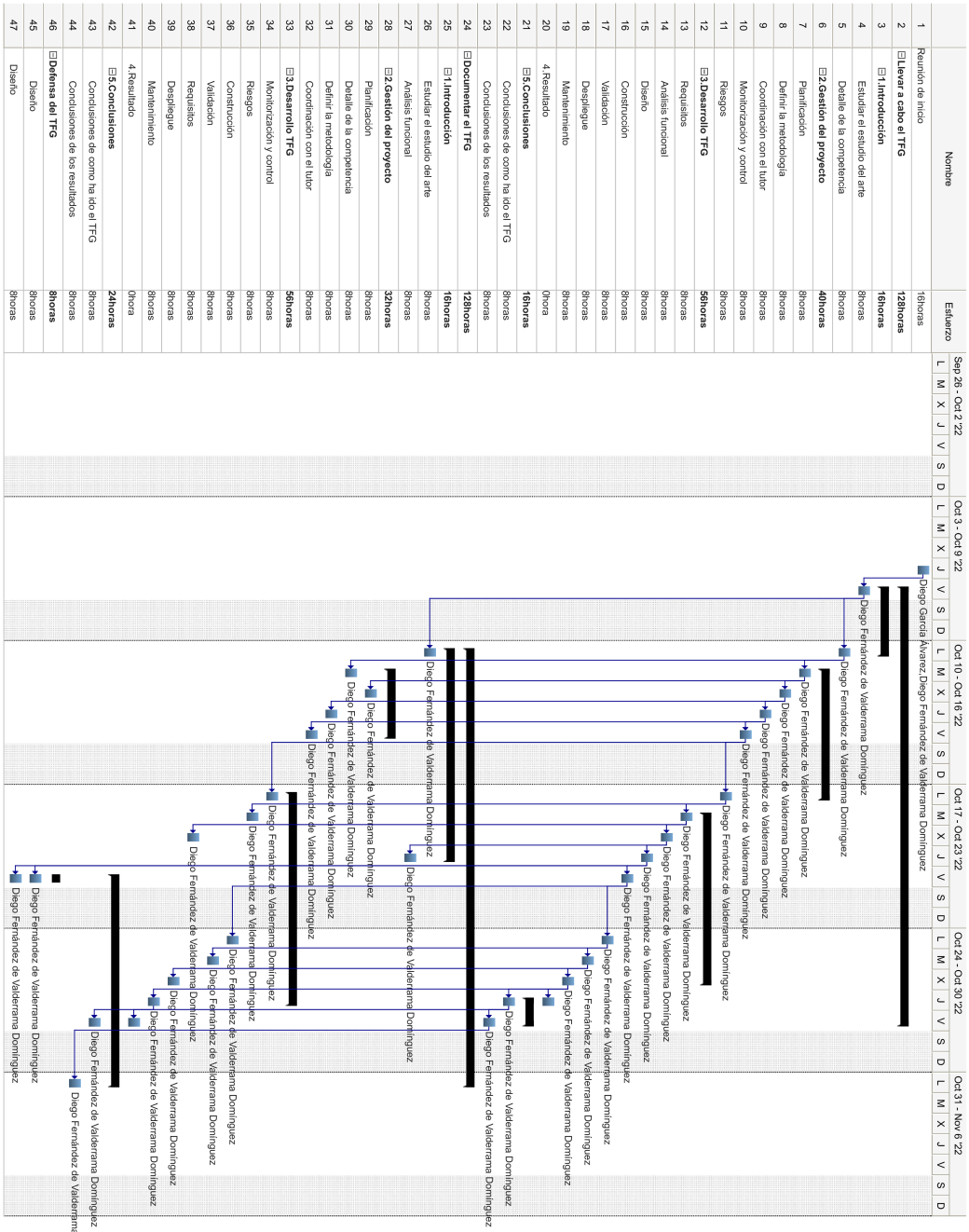


Figura 4.1: Diagrama de Gantt del proyecto

4.3. Seguimiento de las Tareas

Un Tablero Kanban es una herramienta visual que se utiliza para gestionar proyectos y tareas de forma eficiente. En este, as tareas se organizan en columnas que representan los diferentes estados del proceso de trabajo, desde "por hacer" hasta "en progreso" y "terminado". Cada tarea se representa mediante una tarjeta que contiene información sobre la tarea, como su descripción, prioridad, dueño, fecha de vencimiento, entre otros. El objetivo es tener una visión clara y en tiempo real del estado del trabajo, para poder tomar decisiones informadas sobre cómo avanzar y mejorar continuamente.

Trello es una herramienta en línea que utiliza el concepto de tablero Kanban para la gestión de proyectos y tareas. En Trello, los usuarios pueden crear tableros personalizados con columnas que representan los diferentes estados del proceso de trabajo, y las tarjetas se utilizan para representar las tareas y la información asociada a ellas.

Una de las ventajas de Trello es que es muy fácil de usar y configurar, lo que lo convierte en una herramienta accesible para equipos de todos los tamaños y niveles de experiencia. Es por esto y mas motivos que se ha decidido emplear Trello para este proyecto. A continuación se muestra un fragmento del Trello de este proyecto (Ver. 4.2 Tablero Kanban: Trello del Proyecto).

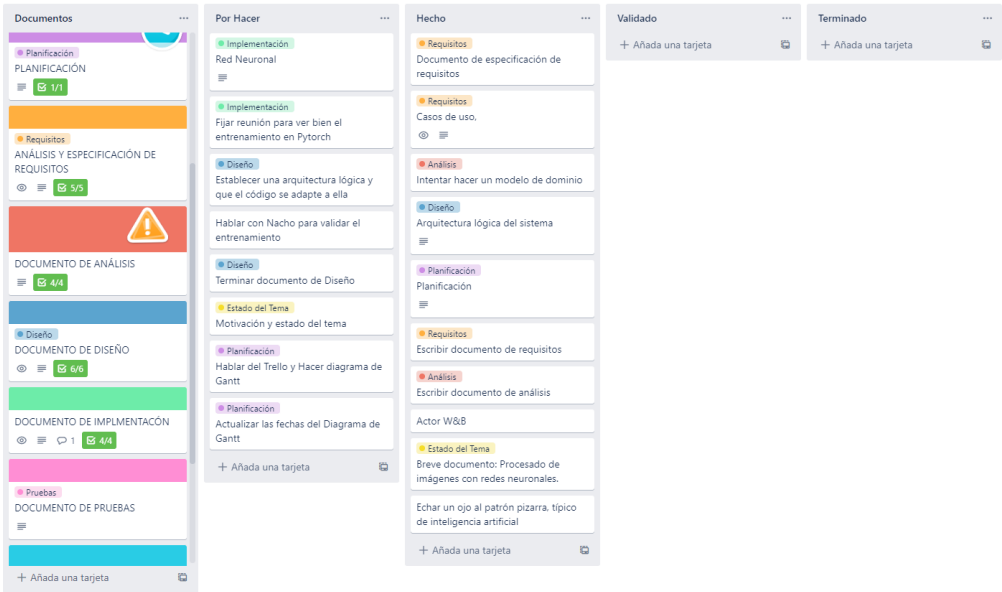


Figura 4.2: Tablero Kanban: Trello del Proyecto

4.4. Análisis de Riesgos

4.4.1. Introducción

La planificación de riesgos es un proceso crucial dentro de la gestión de proyectos que tiene como objetivo identificar, analizar y evaluar los posibles riesgos que pueden afectar el éxito del proyecto, así como desarrollar estrategias para prevenir, mitigar o responder a esos riesgos.

Durante la planificación de riesgos, el equipo del proyecto identifica los riesgos potenciales que podrían impactar negativamente el proyecto y evalúa su probabilidad de ocurrencia y su impacto en el proyecto. Con base en esta evaluación, se desarrollan planes de respuesta a los riesgos, que pueden incluir acciones para evitar o reducir los riesgos, así como planes de contingencia para manejarlos si se materializan.

Para el desarrollo del Análisis de Riesgos se ha provisto de un documento conteniendo una lista de todos los planteados. (ver 4.4.2 Riesgos Detectados) He aquí el formato que esta lista sigue:

- **ID:** Identificador único asignado a cada riesgo.
- **Nombre:** Es el nombre del riesgo en cuestión, que debe ser breve pero descriptivo para facilitar su identificación.
- **Descripción:** Es una descripción detallada del riesgo, que incluye información relevante como su origen, alcance y posibles consecuencias.
- **Categoría:** Es la categoría o tipo de riesgo al que pertenece, como puede ser hardware, software, formación, personal, entre otros.
- **Vulnerabilidad:** Es la debilidad o fallo en el sistema, proceso o activo que hace posible la aparición del riesgo.
- **Amenaza:** Es el evento o suceso que puede materializar el riesgo, como un incendio, un robo, una falla en el sistema, entre otros.
- **Probabilidad:** Es la posibilidad o frecuencia con la que puede ocurrir el evento que materializa el riesgo (Se puede ver en 4.1 Matriz de Riesgos).
- **Impacto:** Es el efecto o consecuencia que tendría la materialización del riesgo, en términos de pérdidas financieras, daños a la reputación, interrupción de operaciones, entre otros (Se puede ver en 4.1 Matriz de Riesgos).
- **Riesgo Total, exposición o nivel de riesgo:** Es la combinación de la probabilidad y el impacto del riesgo, y se utiliza para evaluar la importancia o prioridad de cada riesgo en la lista.

PROB/IMP	BAJA	MEDIA	ALTA
BAJO	BAJO	BAJO	MEDIO
MEDIO	BAJO	MEDIO	ALTA
ALTO	MEDIO	ALTA	ALTA

Tabla 4.1: Matriz de Riesgos

- **Estado del Riesgo:** Es el estado actual del riesgo, si está aceptado, derivado, abierto, cerrado, mitigado, entre otros.
- **Acciones de mitigación:** Son las medidas preventivas que se pueden tomar para reducir la probabilidad de que ocurra el riesgo o para minimizar su impacto en caso de que se materialice.
- **Acciones correctivas:** Son las medidas que se toman después de que el riesgo se ha materializado, con el fin de recuperar la normalidad lo antes posible, aprender de la experiencia y prevenir su repetición en el futuro.

4.4.2. Riesgos Detectados

RSK01:	Falta de Tiempo
Descripción:	La estimación de tiempo frente al trabajo a realizar no fue la adecuada.
Categoría:	Formación.
Vulnerabilidad:	Incumplimiento de los plazos de entrega.
Amenaza:	Al ser la primera vez que se realiza un proyecto de tal calibre, es posible estimar mal el tiempo a dedicar al mismo.
Probabilidad:	Media.
Impacto:	Bajo.
Riesgo Total:	Bajo.
Estado:	Abierto.
Acciones de mitigación	
1. Consultar al Tutor para obtener mejores estimaciones de tiempo. **2.** Plantear el proyecto como prototipo, justificando asi que ciertas partes no estén implementadas.	
Acciones correctivas	
1. Replanificar el proyecto con menos objetivos. **2.** Plantear como objetivo la entrega en la convocatoria extraordinaria.	

Tabla 4.2: Riesgo 01: "Falta de Tiempo"

RSK02:	Enfermedad
Descripción:	Se contrae una enfermedad durante el periodo de desarrollo.
Categoría:	Personal.
Vulnerabilidad:	Incumplimiento de los plazos de entrega.
Amenaza:	La contracción de una enfermedad imposibilita realizar el trabajo acorde a lo previsto.
Probabilidad:	Baja.
Impacto:	Bajo.
Riesgo Total:	Bajo.
Estado:	Abierto.

Acciones de mitigación

1. Plantear el proyecto como prototipo, justificando asi que ciertas partes no esten implementadas.

2. Negociar un menor alcance dada la situación.

Acciones correctivas

1. Replanificar el proyecto con menos objetivos.

2. Plantear como objetivo la entrega en la convocatoria extraordinaria.

Tabla 4.3: Riesgo 02: "Enfermedad"

RSK03:	Tutor no disponible
Descripción:	El tutor no responde apenas a los correos del alumno.
Categoría:	Personal.
Vulnerabilidad:	Desconocimiento sobre si el progreso es adecuado.
Amenaza:	La falta de contacto por parte del Tutor dificulta la realización de un correcto trabajo.
Probabilidad:	Baja.
Impacto:	Bajo.
Riesgo Total:	Bajo.
Estado:	Abierto.

Acciones de mitigación

1. Negociar con el coordinador de título un cambio de tutor o de plazos.

2. Contactar al tutor a través del coordinador de título.

Acciones correctivas

1. Negociar con el coordinador de título un cambio de tutor o de plazos.

Tabla 4.4: Riesgo 03: "Tutor no disponible"

RSK04:	Pérdida de todo el progreso
Descripción:	Se borran todos los archivos referentes al TFG.
Categoría:	Hardware.
Vulnerabilidad:	Debido a un fallo de HDD, se pierden todos los contenidos de este.
Amenaza:	Se pierde todo el progreso realizado hasta la fecha debido a un fallo de almacenamiento.
Probabilidad:	Baja.
Impacto:	Bajo.
Riesgo Total:	Mitigado.

Acciones de mitigación

1. Realización de copias de seguridad frecuentemente.

2. Trabajar sobre la nube.

Acciones correctivas

1. Replanificar el proyecto con menos objetivos.

2. Plantear como objetivo la entrega en la convocatoria extraordinaria.

3. Intentar recuperar partes desde documentos adjuntos de correos o similares.

Tabla 4.5: Riesgo 04: "Pérdida de todo el progreso"

RSK05:	Baja del Tutor no notificada
Descripción:	El tutor pasa a estar de baja, y esta no se notifica adecuadamente.
Categoría:	Personal.
Vulnerabilidad:	Desconocimiento sobre si el progreso es adecuado.
Amenaza:	La falta de contacto por parte del Tutor dificulta la realización de un correcto trabajo.
Probabilidad:	Baja.
Impacto:	Bajo.
Riesgo Total:	Bajo.
Estado:	Derivado.
Acciones de mitigación	
1. Procurar un conocimiento pleno de los criterios de evaluación y similares acerca del TFG.	
Acciones correctivas	
1. Negociar con la UVA un cambio de tutor o de plazos.	

Tabla 4.6: Riesgo 05: "Baja del Tutor no notificada"

RSK06:	Cambios en el enfoque o alcance del proyecto
Descripción:	El enfoque o alcance del proyecto pueden cambiar durante el proceso de investigación y desarrollo del TFG.
Categoría:	Software.
Vulnerabilidad:	Desconocimiento de los límites del trabajo a realizar.
Amenaza:	El proyecto puede sufrir retrasos y aumentar el esfuerzo si se producen cambios significativos en el enfoque o alcance.
Probabilidad:	Media.
Impacto:	Medio.
Riesgo Total:	Medio.
Estado:	Aceptado.
Acciones de mitigación	
Establecer un proceso sólido de gestión de cambios. Asegurarse de que se documenten y comuniquen adecuadamente todos los cambios en el enfoque o alcance del proyecto.	
Acciones correctivas	
Realizar un seguimiento de los cambios en el enfoque o alcance del proyecto y ajustar la planificación y el esfuerzo en consecuencia.	

Tabla 4.7: Riesgo 06: "Cambios en el enfoque o alcance del proyecto"

RSK07:	Dificultades para encontrar y acceder a fuentes de información
Descripción:	Puede resultar difícil encontrar y acceder a fuentes de información relevantes para el proyecto.
Categoría:	Formación.
Vulnerabilidad:	Dificultad para saber si el proyecto dispone de una base sólida.
Amenaza:	Las dificultades para encontrar y acceder a fuentes de información pueden afectar la calidad y la profundidad del proyecto.
Probabilidad:	Media.
Impacto:	Bajo.
Riesgo Total:	Bajo.
Estado:	Aceptado.

Acciones de mitigación

1. Realizar una búsqueda exhaustiva de fuentes de información relevantes.
2. Utilizar técnicas de búsqueda efectivas para encontrarlas.

Acciones correctivas

1. Utilizar fuentes alternativas.
2. Ampliar el enfoque del proyecto si es necesario.

Tabla 4.8: Riesgo 07: "Dificultades para encontrar y acceder a fuentes de información"

RSK08:	Problemas de comunicación o colaboración con el tutor o el director
Descripción:	Puede haber dificultades para establecer una buena comunicación o colaboración con el tutor o el director del proyecto.
Categoría:	Personal.
Vulnerabilidad:	Desconocimiento sobre los límites del proyecto.
Amenaza:	Los problemas de comunicación o colaboración pueden afectar la calidad y la efectividad del proyecto y pueden generar estrés y presión innecesarios.
Probabilidad:	Baja.
Impacto:	Alto.
Riesgo Total:	Medio.
Estado:	Mitigado.

Acciones de mitigación

1. Establecer una comunicación clara y regular con el tutor o el director.

2. Asegurarse de que se comprendan y respeten los roles y responsabilidades de cada uno.

Acciones correctivas

1. Solucionar los problemas de comunicación o colaboración de manera rápida y eficiente.

Tabla 4.9: Riesgo 08: "Problemas de comunicación o colaboración con el tutor o el director"

RSK09:	Problemas de plagio o fraude académico
Descripción:	El estudiante puede incurrir en plagio o fraude académico al utilizar material o ideas de otras fuentes sin dar debida atribución o al presentar trabajo como propio cuando no es así.
Categoría:	Personal.
Vulnerabilidad:	Gran riesgo de perder toda validez en el proyecto.
Amenaza:	El plagio o el fraude académico pueden tener graves consecuencias para el estudiante, como la pérdida de credibilidad y la expulsión del programa de estudios.
Probabilidad:	Baja.
Impacto:	Alto.
Riesgo Total:	Medio.
Estado:	Cerrado.
Acciones de mitigación	
1. Asegurarse de entender y seguir las normas y políticas de plagio y fraude académico de la institución y utilizar herramientas de verificación de plagio para detectar y evitar problemas.	
Acciones correctivas	
1. Identificar y solucionar los problemas de plagio o fraude académico de manera rápida y eficiente.	

Tabla 4.10: Riesgo 09: "Problemas de plagio o fraude académico"

RSK10:	Problemas con la calidad del trabajo
Descripción:	El trabajo del estudiante puede tener problemas de calidad, como errores o falta de profundidad en la investigación o en la presentación del proyecto.
Categoría:	Personal.
Vulnerabilidad:	Riesgo de reducir la puntuación final.
Amenaza:	Los problemas de calidad pueden afectar la valoración del trabajo y pueden requerir un esfuerzo adicional para corregirlos.
Probabilidad:	Media.
Impacto:	Medio.
Riesgo Total:	Medio.
Estado:	Cerrado.
Acciones de mitigación	
1. Establecer una planificación detallada y realista. **2.** Asegurarse de tener suficiente tiempo para revisar y corregir el trabajo.	
Acciones correctivas	
Revisar y corregir los problemas de calidad de manera rápida y eficiente.	

Tabla 4.11: Riesgo 10: "Problemas con la calidad del trabajo"

4.5. Recursos Necesarios

- **Equipo de desarrollo:** Para el desarrollo, programación y mantenimiento de la aplicación, se requiere la colaboración de expertos en software, diseñadores de interfaz de usuario y especialistas en experiencia de usuario.
- **Datos:** Para ajustar el modelo neuronal para la clasificación de diferentes tipos de basura, como papel, plástico, vidrio, orgánico, metales, etc. La obtención de datos puede requerir la colaboración con empresas de reciclaje, organizaciones gubernamentales o privadas que se dediquen al reciclaje, o incluso el desarrollo de una estrategia de recolección de datos propios.
- **Hardware:** Podría ser necesario contar con servidores para alojar la aplicación y realizar el procesamiento de datos en tiempo real. También se podría requerir de dispositivos móviles (teléfonos inteligentes o tabletas) para probar la aplicación y asegurarse de que funciona de manera efectiva.
- **Base de conocimiento:** Para que la aplicación pueda reconocer y clasificar correctamente los distintos tipos de basura, es posible que se requiera una base de conocimiento sobre los diferentes materiales y cómo se deben clasificar. Esta información se podría obtener de expertos en el tema o a través de investigación y recopilación de datos.

- **Financiación:** El proyecto podría requerir inversión para cubrir los costes de desarrollo, marketing y promoción de la aplicación. En este sentido, se podría considerar la posibilidad de buscar financiación a través de programas de apoyo a startups o inversores privados.
- **Pruebas y evaluación:** Para garantizar la calidad de la aplicación, se podrían requerir pruebas y evaluaciones exhaustivas en diferentes situaciones y contextos. Esto podría requerir la colaboración de expertos en el tema o la contratación de consultores especializados.
- **Plan de marketing:** Para asegurarse de que la aplicación llegue a su público objetivo, se podría requerir la elaboración de un plan de marketing que incluya estrategias de publicidad y promoción en redes sociales, plataformas de distribución de aplicaciones, etc.
- **Seguridad y privacidad:** Dado que la aplicación estaría recopilando información sobre los usuarios y su comportamiento en relación a la clasificación de basura, es importante garantizar la seguridad y privacidad de los datos. Para ello, se podría requerir la implementación de medidas de seguridad y el cumplimiento de regulaciones y leyes de protección de datos.
- **Capacitación y soporte:** Para garantizar que los usuarios puedan utilizar la aplicación de manera efectiva, se podría requerir la elaboración de material de capacitación y soporte técnico para responder a las preguntas y dudas que puedan surgir.
- **Alianzas estratégicas:** Podría ser beneficioso establecer alianzas estratégicas con organizaciones o empresas relacionadas con la gestión de residuos y reciclaje, lo que podría ayudar a promocionar la aplicación y mejorar la calidad de la información y datos que se utilizarían en el proyecto.

4.6. Presupuesto

El presupuesto es un elemento crucial en cualquier proyecto, y en el caso de una aplicación de clasificación de basura no es diferente. El coste del desarrollo de la aplicación dependerá de varios factores, como el tamaño y complejidad del proyecto, los recursos necesarios, el equipo de desarrollo, los materiales y herramientas, entre otros. Por lo tanto, es importante contar con un análisis claro de los costes para garantizar que se cuente con la financiación adecuada.

Un presupuesto bien elaborado puede ayudar a establecer una visión clara del alcance del proyecto, y a definir las prioridades en términos de gastos. Los costes que podrían estar incluidos en el presupuesto de una aplicación de clasificación de basura podrían incluir:

- **Costes de desarrollo:** En una primera instancia se acepta este como nulo, ya que estrictamente hablando el desarrollo esta hecho bajo el contexto de un TFG.

- **Costes de Hardware:** El presupuesto podría incluir el coste de cualquier hardware necesario para el proyecto.

  - **Coste energético:** Todo desarrollo se hace en o a través de un dispositivo electrónico, teniendo consecuentemente asignado un gasto eléctrico

    Asumiendo un costo de 0,28442 €/kWh

Fase	Consumo (Wh)	Tiempo de Computo(h)	Total(€)
Investigación	200	45	2,55978
Desarrollo	250	60	4,2663
Ejecución y Pruebas	250	30	2,13315
Total	-	135	8,98923

Tabla 4.12: Estimación Consumo Energético

  - **Coste DataSet:** Para el entrenamiento de Modelos se requiere de DataSet, y estos pueden suponer un coste. En este caso se ha decidido emplear un DataSet gratuito:

    Trashnet [10]

  - **Coste Base de Datos:** El hecho de alojar una base de datos puede suponer un coste añadido, sobre todo teniendo en cuenta los criterios de disponibilidad.

    Para la implementación de estas tecnologías se ha decidido emplear Google Firebase. He aquí una estimación de costes mensuales basada en la estimación de Google [11]:

Nº Instalaciones	**Costes de lectura/ escritura ($)**	**Costes de almacenamiento/ redes ($)**	**Total ($)**
Pequeña (50k instalaciones)	11,10	1,04	12,14
Medio (1M instalaciones)	261,90	30,12	292,02
Largo (10M instalaciones)	2637,90	313,62	2951,52

Tabla 4.13: Estimación Costes Base de Datos

- **Coste Servidores:** Similar al anterior punto, el alojar servicios con plena disponibilidad pueden suponer costes añadidos. Para la estimación de estos costes se ha estudiado el plan de precios ofrecido por amazon AWS para alojar API REST [12]:

Nº solicitudes mensuales	Costes por millón($)
Primeras 333M	3,50
Siguientes 667M	2,80
Siguientes 19B	2,38
Mas de 20B	1,51

Tabla 4.14: Estimación Costes Servidores

- **Coste del Entrenamiento del Modelo:** El entrenamiento de un Modelo puede resultar extensivo tanto monetariamente como temporalmente. Es por esto que esto coste no puede ser ignorado. Para el entrenamiento del Modelo, se utilizara el servicio de computación ofrecido por Google (Google Colab). Escogeremos el plan de pago por unidad de computo, al cual deberemos sumarle un mínimo de 51,12€ al mes, ya que este plan nos ofrece plenas capacidades y mejores unidades de computo [13].

Nº Unidades Informáticas	Coste(€)
100	11,19
500	51,12

Tabla 4.15: Estimación Costes Entrenamiento

- **Otros Costes:** Estos costos quedarán pendientes de estimar.
  - **Costes de marketing y promoción:** El presupuesto podría incluir los costes asociados con la promoción de la aplicación, como la publicidad en línea, eventos de promoción y la producción de materiales de marketing.
  - **Costes de capacitación:** El presupuesto podría incluir los costes asociados con la capacitación de personal capaz de filtrar tipos de residuos.

Es importante destacar que el presupuesto no debe ser visto simplemente como un gasto necesario, sino como una inversión en el éxito del proyecto. Al tener una comprensión clara de los costes y la capacidad de financiamiento, se puede tomar decisiones informadas y estratégicas sobre cómo asignar los recursos para lograr los objetivos del proyecto.

También es importante destacar que la mayoría de estos presupuestos no han tenido repercusión durante el desarrollo del presente TFG debido a la menor escala que requiere un entorno de desarrollo en comparación a uno de producción.

Capítulo 5

Análisis y especificación de requisitos

5.1. Introducción

El análisis y especificación de requisitos es un proceso en el desarrollo de software que tiene como objetivo entender y definir de manera clara los requisitos y expectativas del cliente o usuario final. Esto incluye la identificación de los objetivos del sistema, la funcionalidad deseada, los límites y restricciones, y cualquier otro aspecto relevante que influya en el diseño y desarrollo del software. Este proceso es esencial para garantizar que el software cumpla con las necesidades y expectativas de los usuarios y para evitar errores y malentendidos en el futuro.

5.2. Descripción Detallada del Sistema

El sistema a desarrollar comprenderá múltiples componentes que interactúan entre sí, generando un ciclo.

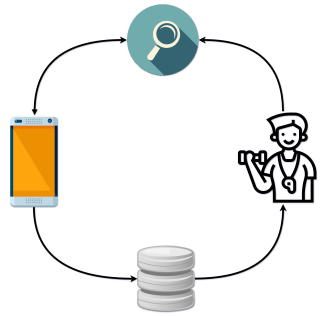


Figura 5.1: Diagrama Simplificado Funcionamiento del Sistema

Para ilustrar esta idea, se considera una aplicación móvil que captura imágenes las cuales pueden ser enviadas para su clasificación como basura y/o almacenadas en una base de datos (Si el usuario determina que la clasificación es incorrecta, puede proporcionar una nueva clasificación).

Las imágenes reportadas por los usuarios pueden filtrarse para generar nuevos conjuntos de imágenes que se utilizarán posteriormente para mejorar el modelo de clasificación de imágenes. De esta manera se completa el ciclo mencionado previamente.

El Sistema a desarrollar se va descomponer en cinco subsistemas:

- **Ajuste del Modelo:** Esta parte del sistema será la encargada de generar y ajustar un Modelo de clasificación de imágenes exportable. Para ello deberá poder leer datasets de imágenes correctamente etiquetadas. Esta parte del sistema deberá presentar la suficiente flexibilidad para admitir cambios en su DataSet (Esto puede ser tanto una ampliación del previamente existente, o la generación de nuevos tipos de residuos a clasificar).
- **API REST de clasificación de Imágenes:** Esta deberá ser capaz de leer un modelo de Clasificación de Imágenes previamente generado por el Entrenador del Modelo, y dada una imagen ser capaz de retornar la clasificación. Para la realización de este micro servicio se debe poder escuchar peticiones web ("POST") que contengan la imagen a clasificar y retornar como parte de esta la clasificación pertinente.
- **Aplicación Móvil:** Deberá ser capaz de tomar fotos desde la cámara del propio dispositivo móvil o de su galería y enviárselo a la previamente mencionada API REST, y una vez recibida la respuesta de la clasificación por parte de esta, mostrarla al usuario. Adicionalmente, esta aplicación debe ser capaz de poder gestionar los reportes de usuarios respectivos a imágenes incorrectamente clasificadas y enviarlos a una Base de Datos, para que puedan ser filtradas adecuadamente. Para la realización de reportes el usuario deberá estar autenticado, para así evitar el SPAM, o tipos similares de ataques, que puedan restar credibilidad a estos reportes masivos.
- **Base de Datos para la administración de imágenes reportadas.** En esta base de datos se debe poder almacenar las imágenes reportadas por los usuarios desde la

aplicación móvil, y que estas contengan un identificador del artífice del reporte junto a la clasificación considerada correcta por el usuario y la clasificación dada por el modelo. Por otro lado esta Base de Datos, debe almacenar datos de usuarios, como el numero de reportes realizados, y cuantos de estos han sido verificados como correctos.

- **Aplicación Web para filtrado y retroalimentación al entrenador del modelo.** Este servicio Web deberá ser capaz de leer los datos almacenados en la Base de Datos previamente mencionada, y mediante una interfaz de usuarios de fácil manejo permitir a un administrador del sistema filtrar las imágenes como correcta o incorrectamente clasificadas para así, tras reunir suficientes, poder reentrenar al Modelo inicial. Es importante destacar que el administrador podrá filtrar los usuarios por el porcentaje de certeza de sus reportes, y marcar usuarios como no validos si así lo desea, para ignorar sus reportes de ahora en adelante. En la versión actual del proyecto esta funcionalidad no se encuentra implementada.

Es importante destacar que siempre que sea posible, estos subsistemas deberán ser *dockerizados*, para así poder facilitar la ejecución en distintas maquinas y tener una mayor flexibilidad para futuras modificaciones.

5.3. Especificación de requisitos Software

5.3.1. Requisitos Funcionales

R.F.01 **Clasificar Basuras:** El Sistema clasificará una imagen, retornando la estimación de a que categoría pertenece.

R.F.02 **Certeza de Usuario:** El Sistema proveerá el porcentaje de acierto en sus reportes del Usuario.

R.F.03 **Autenticación:** El Usuario podrá autentificarse en el Sistema.

R.F.04 **Reportar Imagen:** El Usuario podrá reportar una imagen como, incorrectamente clasificada tras esta haber sido clasificada por el Sistema.

R.F.05 **Notificar mejoras:** El Usuario podrá generar un reporte con posibles mejoras a realizar a la aplicación.

R.F.06 **Almacenar en Galería:** El Usuario podrá decidir si desea guardar todas las imágenes clasificadas en la galería.

R.F.07 **Ajustar Modelo:** El Entrenador podrá entrenar el modelo dado un DataSet de Imágenes previamente clasificadas.

R.F.08 **Validar Modelo:** El Entrenador podrá validar la certeza que presentan los modelos generados mediante DataSets de pruebas.

R.F.09 **Exportar Modelo:** El Entrenador podrá exportar modelos ya entrenados.

R.F.10 **Consultar Modelo:** El Entrenador podrá consultar las estadísticas de modelos ya entrenados o entrenándose.

R.F.11 **Invalidar Usuario:** El Administrador podrá marcar a al Usuario como no valido bajo su juicio.

R.F.12 **Validar Propuestas:** El Administrador podrá validar los reportes generados por el Usuario como correcto o incorrecto.

R.F.13 **Exportar nuevo DataSet:** El Administrador podrá exportar el nuevo DataSet generado con los datos de reportes para su posterior uso por el Entrenador.

5.3.2. Requisitos No Funcionales

R.N.F.1 **Respuesta Rápida:** El Sistema deberá responder rápido a la clasificación de una imagen.

R.N.F.2 **Plena Disponibilidad:** El Sistema deberá estar disponible en todo momento.

R.N.F.3 **Interfaz Sencilla:** El Sistema deberá presentar una interfaz de usuario sencilla de entender y usar.

R.N.F.4 **Rápida Autenticación:** El Sistema deberá proveer un método de autenticación que no presente trabas al Usuario.

R.N.F.5 **Uso Eficiente de la Batería:** El Sistema deberá ser eficiente con el uso de la batería del dispositivo.

R.N.F.6 **Aplicación ligera:** El Sistema deberá proveer de una aplicación que no ocupe demasiado espacio del almacenaje.

R.N.F.7 **Poco calor:** El Sistema deberá proveer una aplicación que no caliente en exceso el dispositivo móvil.

R.N.F.8 **Cualquier dispositivo:** El Sistema deberá proveer una aplicación que funcione adecuadamente en dispositivos de gama baja.

5.4. Diagrama de Casos de Uso

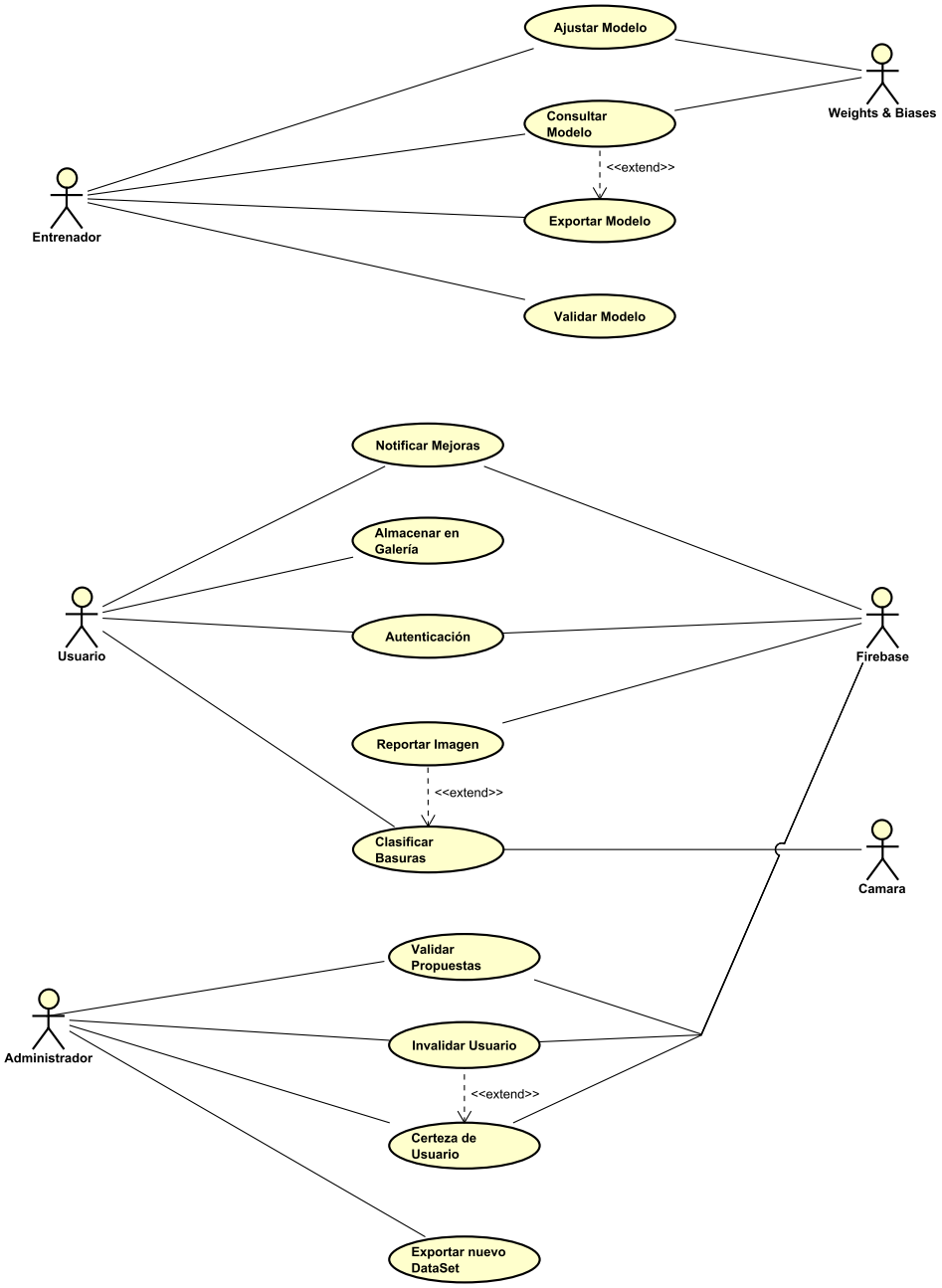


Figura 5.2: Diagrama de Casos de Uso del sistema

La Figura 5.2 muestra el Diagrama UML de casos de uso del Sistema, especificando tanto actores principales como secundarios y la relación entre ellos.

5.5. Actores. Descripción de cada uno de los actores del sistema

- **Usuario:** Aquel que emplea la aplicación móvil, no dispone de conocimiento acerca del funcionamiento e implementación del Sistema. Su objetivo principal es lograr discernir a que contenedor pertenece un residuo en concreto.
- **Entrenador:** Dispone de un gran conocimiento acerca de la implementación del Sistema, pero no conoce la implementación de la aplicación móvil. Es el encargado de entrenar el modelo de *Machine Learning Visual*, para lo cual se apoya en el uso de imágenes previamente clasificadas o etiquetadas.
- **Administrador:** No tiene por que tener ningún conocimiento de la implementación del Sistema, pues su única tarea es el filtrado de las imágenes reportadas por los usuarios y determinar si estos son validos para seguir teniendo en cuenta sus reportes.
- **Cámara:** Actor de tipo dispositivo, aporta información acerca del sistema. Provee los fotogramas captados por la cámara del dispositivo Android.
- **Firebase:** Actor de tipo Sistema, es el encargado de almacenar todo tipo de dato necesario para la aplicación y de gestionar la autenticación del Usuario.
- **Weights & Biases:** Actor de tipo Sistema, es el encargado de almacenar todas las estadísticas de los modelos y mostrárselas en un formato visible al Entrenador.

5.6. Descripción Casos de Uso

Los casos de uso son la descripción de una acción o actividad que puede ser realizada dentro del sistema. El objetivo de la identificación y descripción de los casos de uso es definir las posibles acciones más relevantes que forman parte del sistema y como este se debe comportar durante su realización, para ello se especificará el flujo normal del caso de uso, la realización exitosa de una acción o actividad, y aquellos flujos alternativos ante imposibilidades de realización o fallos, tanto del sistema como de los actores, a la hora de realizar cada uno de los casos de uso.

En esta sección se describirán los casos de uso identificados en la Figura 5.2.

Caso de uso:	Clasificar Basuras
Descripción:	El caso de uso permitirá clasificar una imagen, retornando la estimación de a que categoría pertenece.
Actores:	Usuario & Cámara.
Pre-condiciones:	
Post-condiciones:	El Sistema retornara una clasificación al Usuario.
Extend:	5.6 Descripción del caso de uso "Reportar Imagen"

Secuencia normal

1. El caso de uso comienza cuando el Usuario desea clasificar una imagen.

2. El Sistema solicita al usuario tomar una imagen para clasificarla.

3. El Usuario toma la imagen de la galería o de la Cámara.

4. El Sistema clasifica y retorna la clasificación de la imagen.

5. El Sistema pregunta al usuario si la imagen ha sido correctamente clasificada y el caso de uso finaliza.

Excepciones

4.a) Si el Sistema no puede realizar la acción muestra un mensaje de error y el caso de uso finaliza.

5.a) El Usuario solicita reporta la imagen como incorrectamente clasificada y comienza el caso de uso 5.6 Descripción del caso de uso "Reportar Imagen".

Tabla 5.1: Descripción del caso de uso "Clasificar Basuras"

Caso de uso:	Mostrar Certeza de Usuario
Descripción:	El caso de uso permitirá ver el porcentaje de certeza en los reportes de un Usuario.
Actores:	Administrador & Firebase.
Pre-condiciones:	El Sistema deberá contener reportes de usuarios.
Post-condiciones:	El Sistema retornara el porcentaje de acierto de un Usuario.

Secuencia normal

1. El caso de uso comienza cuando el Administrador desea conocer el porcentaje de acierto de un Usuario.

2. El Sistema solicita a Firebase una lista de todos los usuarios.

3. El Sistema muestra al Administrador una lista de todos los usuarios.

4. El Administrador selecciona un usuario.

5. El Sistema muestra el porcentaje de acierto de acierto del usuario en concreto.

6. El Sistema pregunta si el usuario sigue siendo valido.

7. El Administrador responde.

8. El Sistema confirma su validez y el caso de uso finaliza.

Excepciones

8.a) Si el Administrador decide invalidar al Usuario comienza el caso de uso 5.11 Descripción del caso de uso "Invalidar Usuario".

Tabla 5.2: Descripción del caso de uso "Mostrar Certeza de Usuario"

Caso de uso:	Almacenar en Galería
Descripción:	El caso de uso permitirá al Usuario determinar si desea guardar todas las imágenes en la galería.
Actores:	Usuario.
Pre-condiciones:	Se deben de haber garantizado permisos de almacenamiento.
Post-condiciones:	El Sistema almacenará el ajuste de guardar en la galería.

Secuencia normal

1. El caso de uso comienza cuando el Usuario decide almacenar todas las fotos tomadas en la galería.

2. El Sistema muestra al Usuario una confirmación de la acción.

3. El Usuario confirma y el caso de uso finaliza.

4. El Sistema actualiza los ajustes y el caso de uso finaliza.

Excepciones

4.a) Si el Usuario no confirma el caso de uso finaliza.

Tabla 5.3: Descripción del caso de uso "Almacenar en Galería"

Caso de uso:	Consultar Modelo
Descripción:	El caso de uso permitirá ver multiples estadísticas de un modelo al Entrenador.
Actores:	Entrenador & Weights & Biases.
Pre-condiciones:	El Sistema deberá contener al menos un modelo.
Post-condiciones:	El Sistema retornara multiples estadísticas del modelo.

Secuencia normal

1. El caso de uso comienza cuando el Entrenador desea conocer las estadísticas de un modelo.

2. El Sistema muestra al Entrenador una lista de todos los modelos.

3. El Entrenador selecciona un modelo.

4. El Sistema solicita a Weights & Biases las estadísticas de este modelo.

5. El Sistema muestra las estadísticas.

Tabla 5.4: Descripción del caso de uso "Consultar Modelo"

Caso de uso:	Autenticación
Descripción:	El caso de uso permitirá autentificarse en el Sistema, para poder reportar imágenes y notificar mejoras.
Actores:	Usuario &FireBase.
Pre-condiciones:	El Usuario no debe de estar autentificado en el Sistema.
Post-condiciones:	El Sistema da acceso al Usuario a todos los casos de uso que requieran de estar autentificado.

Secuencia normal

1. El caso de uso comienza cuando el Usuario desea o requiere autenticarse en el Sistema.

2. El Sistema solicita al usuario unos credenciales.

3. El Usuario provee los credenciales.

4. El Sistema envia los credenciales a la Firebase.

5. La Firebase valida los credenciales.

6. El Sistema marca a el Usuario como autentificado y el caso de uso finaliza.

Excepciones

5.a) Si los credenciales no son validos vuele al paso 1.

Tabla 5.5: Descripción del caso de uso "Autenticación"

Caso de uso:	Reportar Imagen
Descripción:	El caso de uso permitirá reportar una imagen como incorrectamente clasificada.
Actores:	Usuario & Firebase.
Pre-condiciones:	El Usuario debe de estar autentificado en el Sistema. El Usuario debe de haber clasificado una basura
Post-condiciones:	El Sistema almacenará el reporte generado.

Secuencia normal

1. El caso de uso comienza cuando el Usuario desea comunicar que una imagen ha sido incorrectamente clasificada.

2. El Sistema muestra una lista de todas las posibles clasificaciones.

3. El Usuario selecciona una clasificación.

4. El Sistema acepta el reporte, lo envia a Firebase y el caso de uso finaliza.

Excepciones

4.a) Si hay un problema de conexión el caso de uso finaliza.

Tabla 5.6: Descripción del caso de uso "Reportar Imagen"

Caso de uso:	Notificar Mejoras
Descripción:	El caso de uso permitirá reportar posibles mejoras a realizar a la aplicación.
Actores:	Usuario & Firebase.
Pre-condiciones:	El Usuario debe de estar autentificado en el Sistema.
Post-condiciones:	El Sistema almacenará el reporte generado.

Secuencia normal

1. El caso de uso comienza cuando el Usuario desea comunicar cualquier tipo de mejora a los desarrolladores.

2. El Sistema solicita al usuario un reporte escrito.

3. El Usuario genera el reporte.

4. El Sistema acepta el reporte, lo envía a Firebase y el caso de uso finaliza.

Excepciones

4.a) Si hay un problema de conexión el caso de uso finaliza.

Tabla 5.7: Descripción del caso de uso "Notificar Mejoras"

Caso de uso:	Ajustar Modelo
Descripción:	El caso de uso permitirá entrenar un modelo dado un DatSet de Imágenes previamente clasificadas.
Actores:	Entrenador.
Pre-condiciones:	
Post-condiciones:	El Sistema generara un modelo entrenado.

Secuencia normal

1. El caso de uso comienza cuando el Entrenador quiere ajustar un Modelo.

2. El Sistema solicita al Entrenador un DataSet.

3. El Entrenador provee el DataSet.

4. El Sistema pregunta al Entrenador si desea reentrenar un modelo y el caso de uso finaliza.

5. El Entrenador provee el modelo a reentrenar.

6. El Sistema reentrena el modelo.

7. El Sistema muestra el resultado exitoso y el caso de uso finaliza.

Excepciones

4.a) Si el Entrenador no decide reentrenar un modelo el Sistema genera uno nuevo y el caso de uso finaliza.

7.a) Si el sistema no puede realizar la acción muestra un mensaje de error y el caso de uso finaliza.

Tabla 5.8: Descripción del caso de uso "Ajustar Modelo"

Caso de uso:	Validar Modelo
Descripción:	El caso de uso permitirá validar la certeza que presenta un modelos mediante DataSets de pruebas.
Actores:	Entrenador.
Pre-condiciones:	
Post-condiciones:	El Sistema obtendrá el porcentaje de acierto de un modelo.

Secuencia normal

1. El caso de uso comienza cuando el Entrenador quiere comprobar la certeza de un Modelo.

2. El Sistema solicita al Entrenador un modelo.

3. El Entrenador provee el modelo.

4. El Sistema solicita al Entrenador un DataSet.

5. El Entrenador provee el DataSet.

6. El Sistema valida el modelo.

7. El Sistema muestra el resultado exitoso y el caso de uso finaliza.

Excepciones

7.a) Si el sistema no puede realizar la acción muestra un mensaje de error y el caso de uso finaliza.

Tabla 5.9: Descripción del caso de uso "Validar Modelo"

Caso de uso:	Exportar Modelo
Descripción:	El caso de uso permitirá exportar un Modelo ya entrenado.
Actores:	Entrenador & Weights & Biases.
Pre-condiciones:	El Modelo habrá sido entrenado previamente.
Post-condiciones:	El Sistema generará un Modelo exportable.
Extend:	5.4 Descripción del caso de uso "Consultar Modelo"

Secuencia normal

1. El caso de uso comienza cuando el Entrenador exportar un Modelo.

2. El Sistema comprueba si el Entrenador desea consultar mas datos acerca del Modelo.

3. El Sistema muestra las opciones de exportación del mismo.

4. El Entrenador las selecciona.

5. El Sistema genera un Modelo y el caso de uso finaliza.

Excepciones

2.a) Si el Entrenador desea consultar mas datos comienza el caso de uso 5.4 Descripción del caso de uso "Consultar Modelo".

5.a) Si el sistema no puede realizar la acción muestra un mensaje de error y el caso de uso finaliza.

Tabla 5.10: Descripción del caso de uso "Exportar Modelo"

Caso de uso:	Invalidar Usuario
Descripción:	El caso de uso permitirá marcar a un Usuario como no valido.
Actores:	Administrador & Firebase.
Pre-condiciones:	El Sistema contiene Usuarios que han hecho reportes.
Post-condiciones:	El Sistema obtendrá el porcentaje de acierto de un modelo.

Secuencia normal

1. El caso de uso comienza cuando el Administrador desea invalidar a un Usuario en el Sistema.

2. El Sistema solicita a la Firebase un listado con todos los Usuarios.

3. El Sistema muestra al Administrador un listado con todos los Usuarios y su porcentaje de acierto.

4. El Administrador selecciona un Usuario.

5. El Sistema muestra todos los reportes realizados por el Usuario.

6. El Sistema pide confirmación para marcar como no valido al Usuario.

7. El Administrador provee la confirmación.

8. El Sistema marca el Usuario como no valido y el caso de uso finaliza.

Excepciones

8.a) Si el Administrador decide no marcarlo como no valido, el caso de uso finaliza.

Tabla 5.11: Descripción del caso de uso "Invalidar Usuario"

Caso de uso:	Validar Propuestas
Descripción:	El caso de uso permitirá validar los reportes generados por los Usuarios.
Actores:	Administrador & Firebase.
Pre-condiciones:	Existen reportes presentes en el Sistema.
Post-condiciones:	El Sistema actualizara su DataSet con los reportes validados.

Secuencia normal

1. El caso de uso comienza cuando el Administrador desea validar un reporte.

2. El Sistema solicita a Firebase un listado de reportes de Usuario.

3. El Sistema muestra al Administrador un reporte aleatorio de un Usuario.

4. El Administrador lo marca como valido o no.

5. El Sistema se actualiza y el caso de uso finaliza.

Tabla 5.12: Descripción del caso de uso "Validar Propuestas"

Caso de uso:	Exportar nuevo DataSet
Descripción:	El caso de uso permitirá exportar un DataSet generado a partir de reportes de Usuarios previos.
Actores:	Administrador.
Pre-condiciones:	El Sistema contiene reportes de Usuario validados por el Administrador.
Post-condiciones:	El Sistema generará un DataSet de imágenes correctamente etiquetadas.

Secuencia normal

1. El caso de uso comienza cuando el Administrador quiere exportar un nuevo DataSet.
2. El Sistema muestra las opciones de exportación del mismo.
3. El Administrador las selecciona.
4. El Sistema genera un DataSet y el caso de uso finaliza.

Tabla 5.13: Descripción del caso de uso "Exportar nuevo DataSet"

Capítulo 6

Análisis

6.1. Introducción

El análisis en la ingeniería de software es un proceso en el cual se examina un sistema, aplicación o software con el objetivo de entender su estructura, funcionamiento y requisitos. Este análisis permite identificar posibles problemas y oportunidades de mejora, así como definir un plan para el desarrollo del software. También se utiliza para validar que el software cumpla con los requisitos y expectativas del cliente y de los usuarios finales. A continuación se mostrarán una serie de diagramas que desglosan este análisis.

6.2. Modelo de Dominio

El diagrama de clases del dominio (Figura 6.1) muestra las entidades importantes dentro del dominio y cómo están relacionadas entre sí. Estas entidades pueden incluir objetos del mundo real, como personas, lugares, cosas y procesos, así como conceptos abstractos, como clases y relaciones.

En general, el objetivo del diagrama de clases del dominio es proporcionar una visión general de la estructura del sistema, lo que ayuda a los desarrolladores y los stakeholders a comprender mejor el dominio de aplicación, lo que a su vez ayuda a garantizar que el sistema esté diseñado de manera efectiva para satisfacer las necesidades de los usuarios y cumplir con los requisitos del negocio.

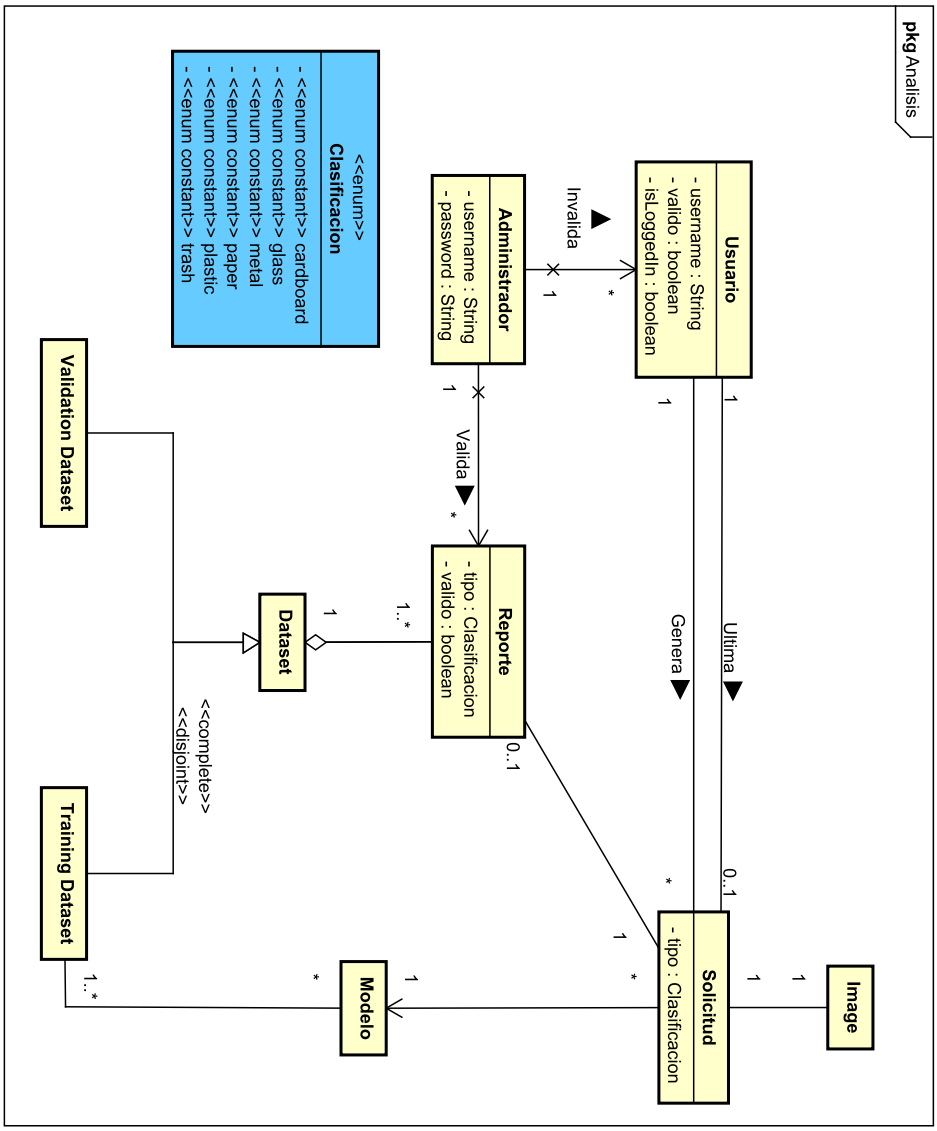


Figura 6.1: Diagrama de Modelo de Dominio

6.3. Clases de Análisis

El diagrama de Clases de Análisis (Figura 6.2) itera sobre el anterior 6.1 Diagrama de Modelo de Dominio añadiendo a este funciones necesarias para poder aproximar los Casos de Uso.

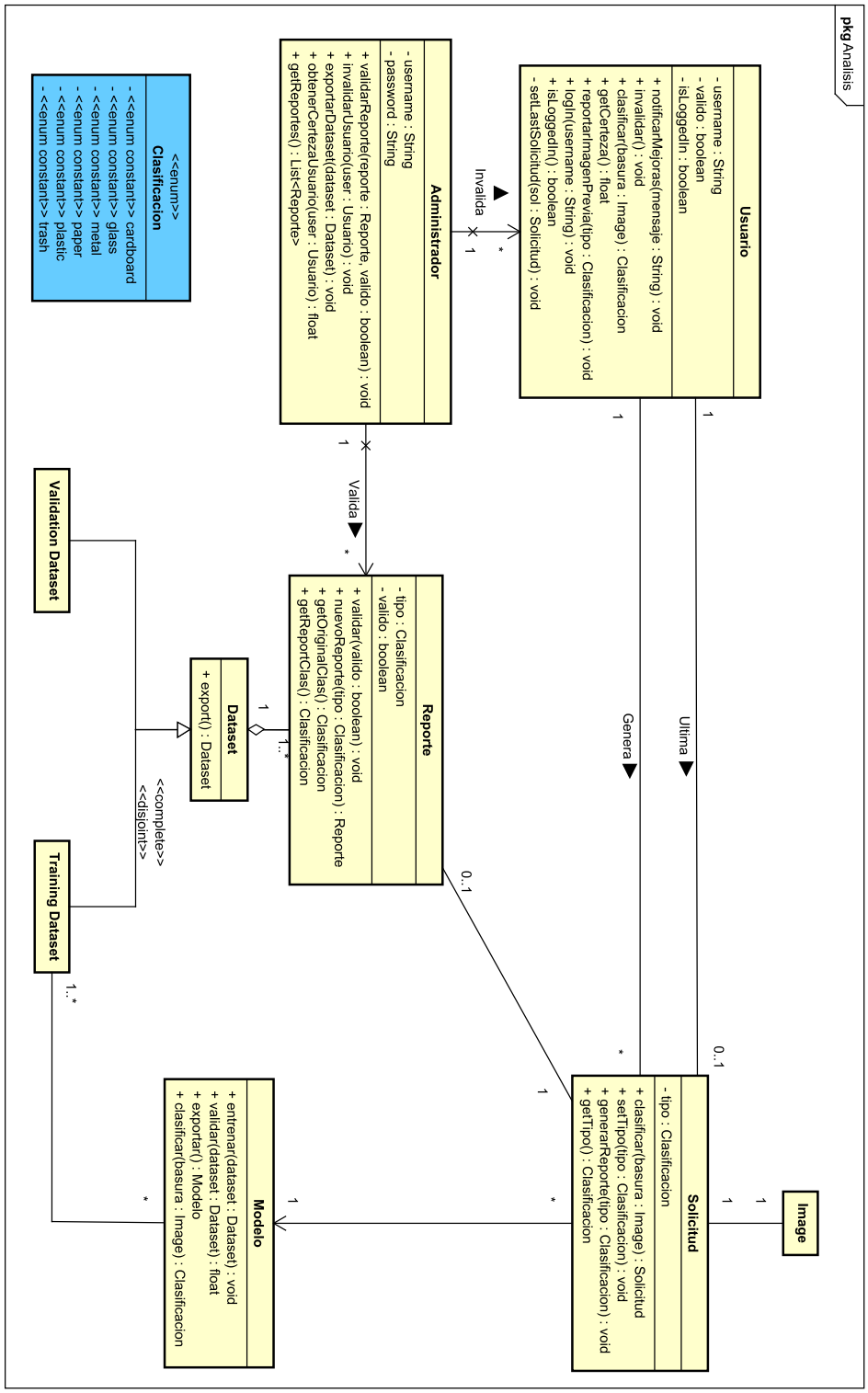


Figura 6.2: Diagrama de Clases de Análisis

6.4. Diagrama Conceptual de la Base de Datos

En la siguiente Figura 6.3 podemos ver de manera conceptual como se plantean relacionar los distintos tipos de datos a almacenar en la Base de Datos.

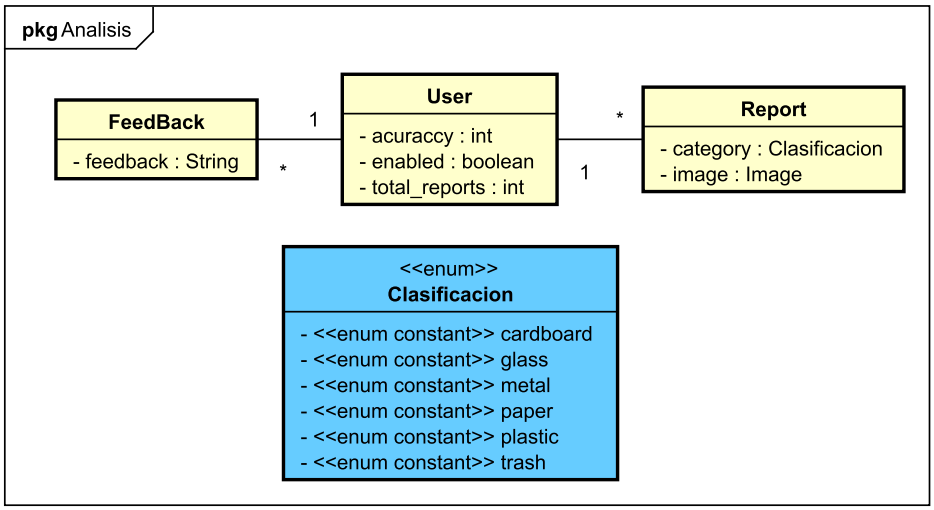


Figura 6.3: Diagrama Conceptual de la Base de Datos

6.5. Realización de Casos de Uso de Análisis

6.5.1. Caso de Uso Clasificar Imagen

En la siguiente Figura 6.4 se muestra el diagrama de secuencia a seguir para que el Usuario pueda clasificar una Image. Se debe destacar que el Usuario puede determinar si la Imagen provendrá de la Galería o no (La Cámara).

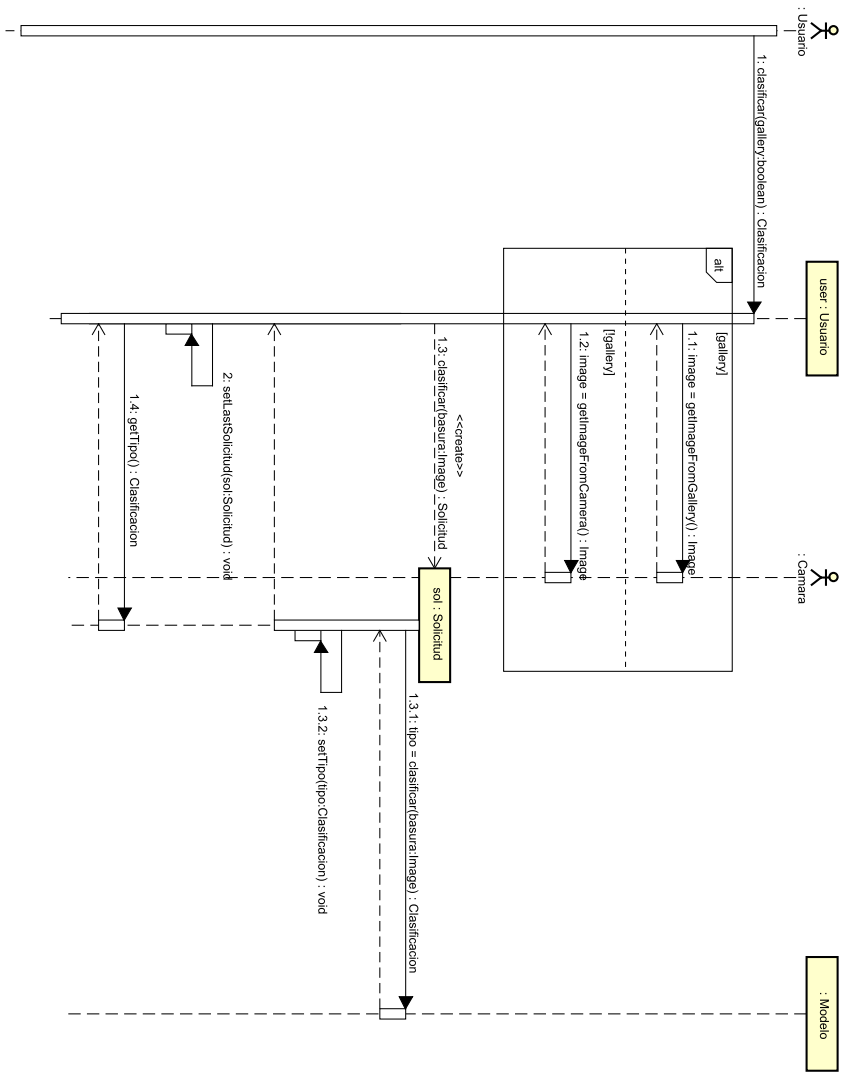


Figura 6.4: Diagrama de Caso de Uso Clasificar Imagen

6.5.2. Caso de Uso Reportar Imagen

En la siguiente Figura 6.5 se muestra el diagrama de secuencia a seguir para que el Usuario pueda reportar una Imagen como incorrectamente clasificada. Se debe destacar que el Usuario debe de haber generado una Solicitud de clasificación previamente.

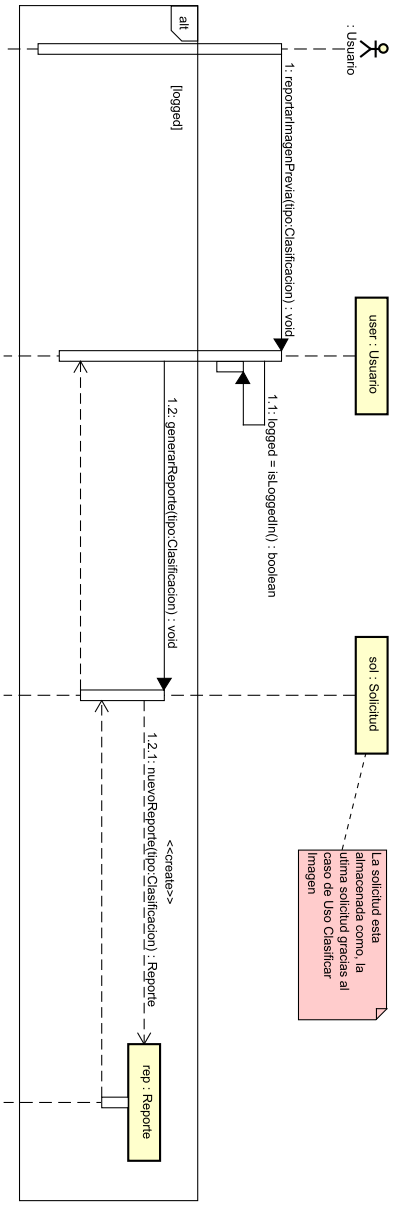


Figura 6.5: Diagrama de Caso de Uso Reportar Imagen

6.5.3. Caso de Uso Validar Propuestas

En la siguiente Figura 6.6 se muestra el diagrama de secuencia a seguir para que el Administrador pueda validar o no un reporte del Usuario. Se debe destacar que es la API FireBase la que contiene todos los reportes.

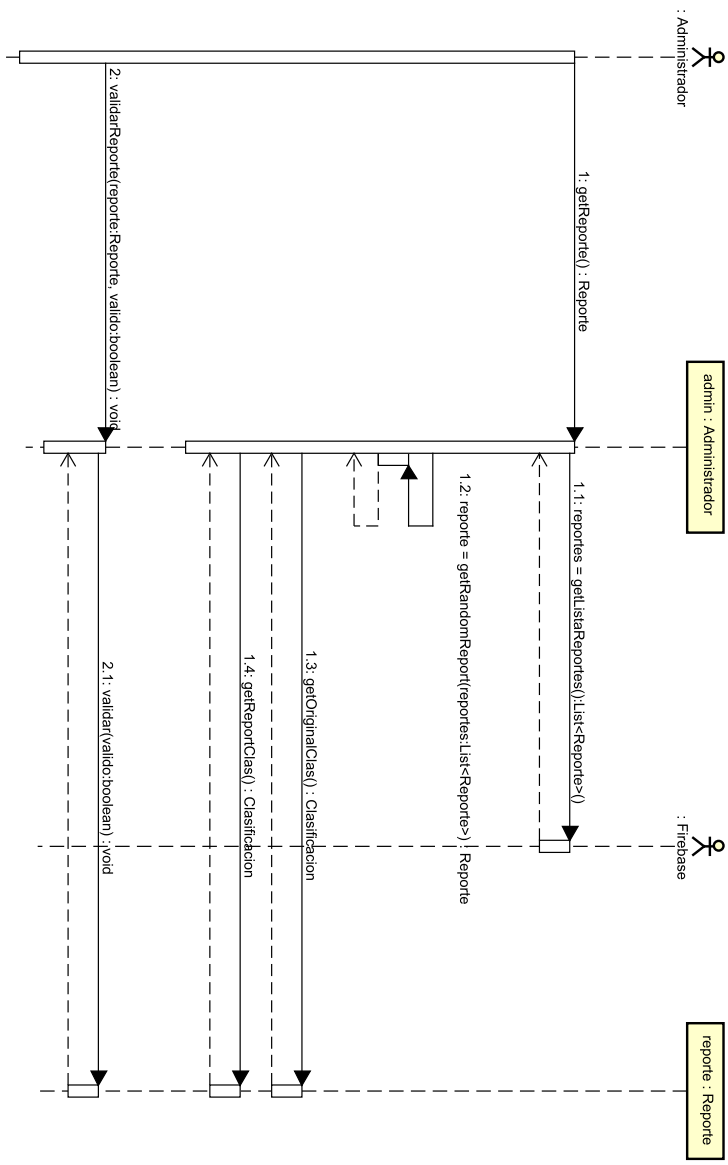


Figura 6.6: Diagrama de Caso de Uso Validar Propuestas

6.5.4. Caso de Uso Invalidar Usuario

En la siguiente Figura 6.7 se muestra el diagrama de secuencia a seguir para que el Administrador pueda marcar un Usuario como no valido. Se debe destacar que tras obtener la certeza del Usuario y todos sus reportes el criterio de invalidación sera puramente el del Administrador.

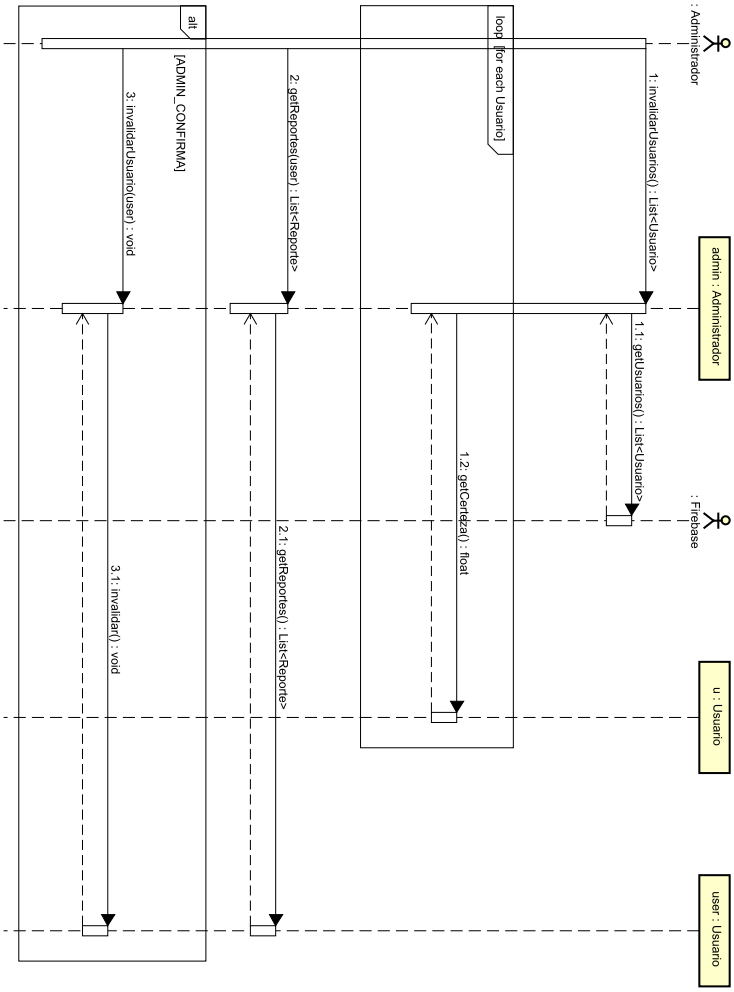


Figura 6.7: Diagrama de Caso de Uso Invalidar Usuario

Capítulo 7

Diseño

7.1. Introducción

El Diseño en la Ingeniería de Software es una fase crítica del desarrollo de software en la que se definen y planifican las soluciones técnicas para abordar los requisitos y objetivos de un proyecto de software. En esta fase, los ingenieros de software trabajan en la identificación y modelado de componentes, sistemas y arquitecturas, con el fin de garantizar que el software cumpla con los estándares de calidad y sea escalable, mantenible y fácilmente modificable. Además, en el diseño se determinan las interfaces y puntos de interacción entre los componentes, y se describen detalladamente las características técnicas y funcionales del software a desarrollar.

7.2. Arquitectura Lógica del sistema

7.2.1. Aplicación Móvil

La arquitectura lógica empleada en este sistema es MVVM.

La arquitectura de un sistema MVVM (Model-View-ViewModel) se compone de tres capas principales: la capa del modelo (Model), la capa de la vista (View) y la capa del modelo de vista (ViewModel).

- **Modelo (Model):** Es el componente que se encarga de representar los datos y la lógica de negocio del sistema. En esta capa se definen las entidades, clases y métodos que representan los objetos y operaciones que interactúan con los datos. El modelo es independiente de la vista y el modelo de vista, y se comunica con ellos a través de interfaces o eventos.

- **Vista (View):** Es el componente que se encarga de presentar la información al usuario y de recibir las interacciones que este realiza en la interfaz gráfica. La vista está compuesta por los elementos gráficos (botones, campos de texto, etiquetas, etc.) y los eventos asociados a ellos. En el patrón MVVM, la vista se enlaza a los datos mediante un modelo de vista, lo que permite que la vista se actualice automáticamente cuando cambian los datos.
- **Modelo de Vista (ViewModel):** Es el componente que actúa como intermediario entre la vista y el modelo. El modelo de vista se encarga de procesar los datos del modelo para presentarlos en la vista de forma adecuada, y de recibir las interacciones del usuario y transformarlas en acciones que afectan al modelo. El modelo de vista es el responsable de mantener el estado de la vista y de notificar a esta cuando cambian los datos del modelo.

La arquitectura lógica de un sistema MVVM se basa en una separación clara de responsabilidades entre las capas del modelo, la vista y el modelo de vista. Esto permite un desarrollo más modular, escalable y fácil de mantener, ya que cada capa se encarga de tareas específicas y tiene una interfaz clara para comunicarse con las otras capas.

7.2.2. Ajuste del Neuronal

Esta sección del sistema se adhiere al patrón Blackboard, también conocido como patrón de Pizarra, una arquitectura de software diseñada específicamente para abordar problemas complejos que no admiten una solución precisa de antemano. El patrón se fundamenta en la colaboración de un grupo de expertos que trabajan juntos para resolver un problema y utilizan una pizarra o tablero negro para compartir información y actualizarla a medida que avanzan en la solución.

En este caso, el patrón Blackboard es utilizado por una sola persona para explorar y probar múltiples parámetros, ajustes y conjuntos de datos diferentes para el entrenamiento de un modelo en el marco del presente trabajo de fin de grado. Este enfoque permite tener un enfoque sistemático y riguroso para probar diversas opciones y analizar los resultados obtenidos.

De esta manera, se puede lograr una comprensión más completa de las posibilidades y limitaciones del modelo en cuestión, lo que conduce a una toma de decisiones más informada y precisa.

7.2.3. Arquitectura de Microservicios

A continuación se expone en que consiste la arquitectura de microservicios que se ha empleado a lo largo de este proyecto.

Una arquitectura de microservicios es un enfoque de diseño de software en el que una aplicación se divide en pequeños servicios independientes, cada uno de los cuales cumple una

función específica. Cada servicio se ejecuta de forma autónoma y se comunica con otros servicios a través de interfaces bien definidas, generalmente mediante protocolos de comunicación ligeros, como HTTP o protocolos de mensajes.

Cada microservicio se desarrolla, implementa, escala y se mantiene de forma independiente de otros servicios de la aplicación, lo que permite una mayor flexibilidad y agilidad en el desarrollo y la entrega de software. Además, esta arquitectura permite que los equipos trabajen en diferentes servicios simultáneamente, lo que acelera el tiempo de desarrollo.

Una arquitectura de microservicios también puede mejorar la escalabilidad y la resiliencia de una aplicación, ya que los servicios pueden ser escalados individualmente según la demanda y los fallos en un servicio no afectan necesariamente a otros servicios de la aplicación.

Sin embargo, esta arquitectura también puede ser más compleja de diseñar y mantener que una arquitectura monolítica, y puede requerir habilidades y herramientas adicionales para gestionar el despliegue y la gestión de múltiples servicios.

7.3. Componentes e interfaces

A continuación se muestran los diagramas relativos a la implementación de tanto la aplicación móvil (Ver. 7.1 Diagrama de Clases de la Aplicación Android), como del servicio REST (Ver. 7.2 Diagrama de Clases del servicio REST de clasificación de Imágenes).

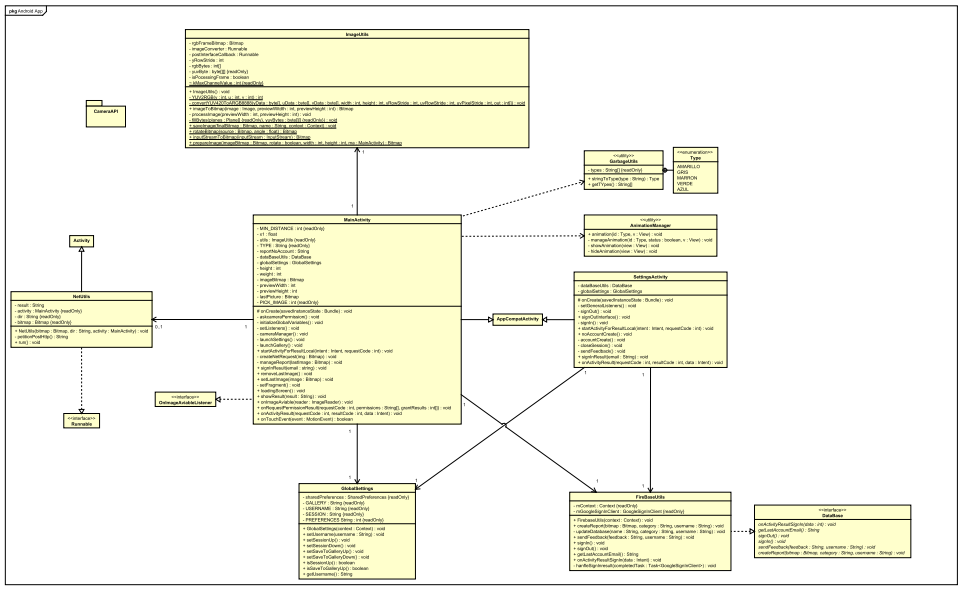


Figura 7.1: Diagrama de Clases de la Aplicación Android

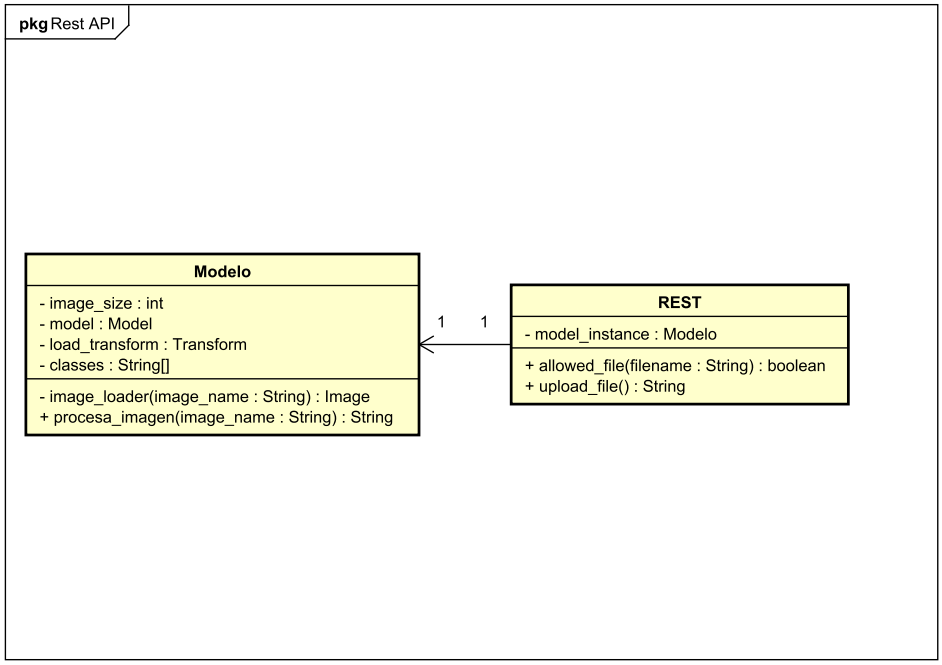


Figura 7.2: Diagrama de Clases del servicio REST de clasificación de Imágenes

7.4. Arquitectura física del sistema

Dada la naturaleza prototípica de este TFG se van a presentar dos diagramas de despliegue distintos: El primero siendo el del proyecto en su estado actual (7.3 Diagrama de Despliegue Actual), y el segundo siendo el planeado una vez finalizado el proyecto (7.3 Diagrama de Despliegue Actual). Este último esta sujeto a cambios y solo pretende dar una ligera idea de como se implementaría.

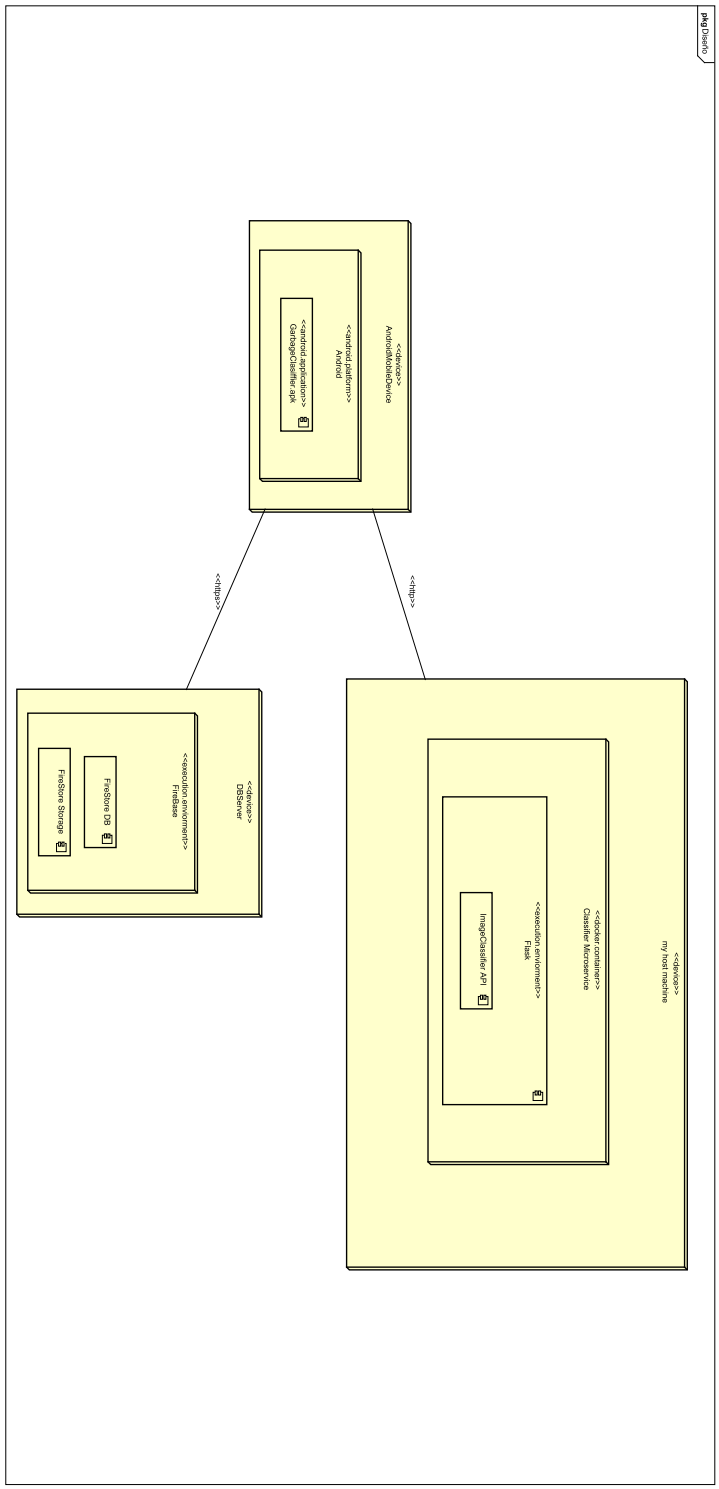


Figura 7.3: Diagrama de Despliegue Actual

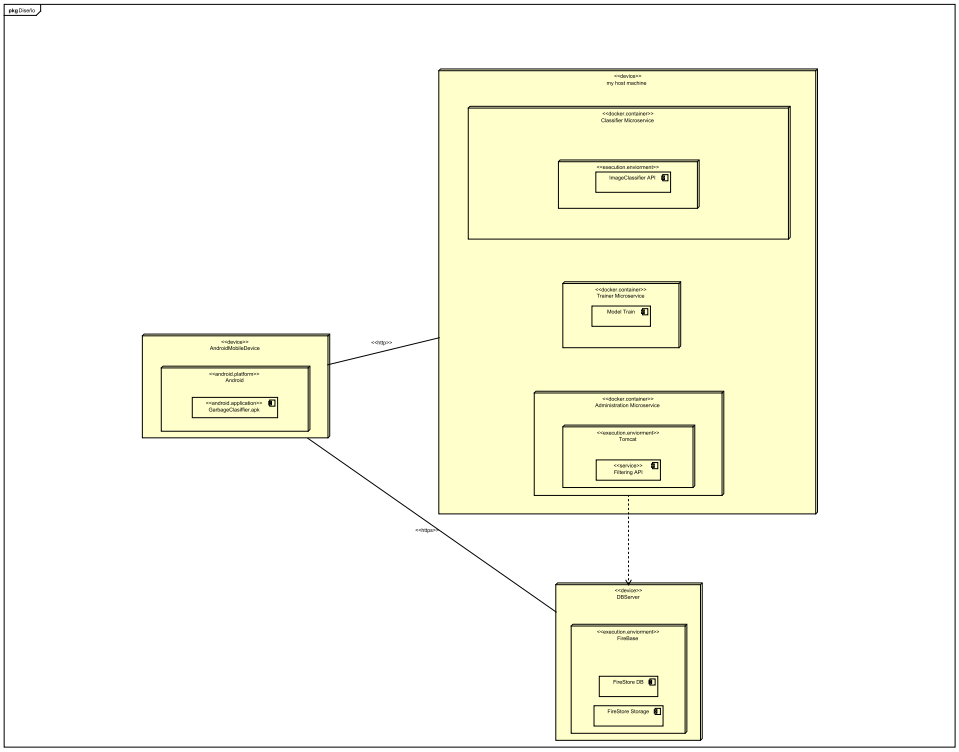


Figura 7.4: Prototipo de Diagrama de Despliegue Planeado

Como se puede apreciar el sistema se encuentra *Dockerizado* en medida de lo posible, así podremos eliminar todo tipo de problema relacionado con el entorno de ejecución. Por otro lado también es destacable la presencia de, en el caso ideal, tres micro-servicios los cuales facilitan el mantenimiento del sistema, no forzando a apagar el mismo de manera entera para poder hacer cambios a solo un fragmento de este.

Otro campo a destacar es la separación del servicio de Base de Datos de Firebase, que al ser externo, no depende de ningún tipo de *Dockerización* o similares.

7.5. Realización de Casos de Uso de Diseño

7.5.1. Realización del Caso de Uso "Clasificación de Basura"

En la Figura 7.5 se muestra la realización del caso de uso 5.1 Descripción del caso de uso "Clasificar Basuras".

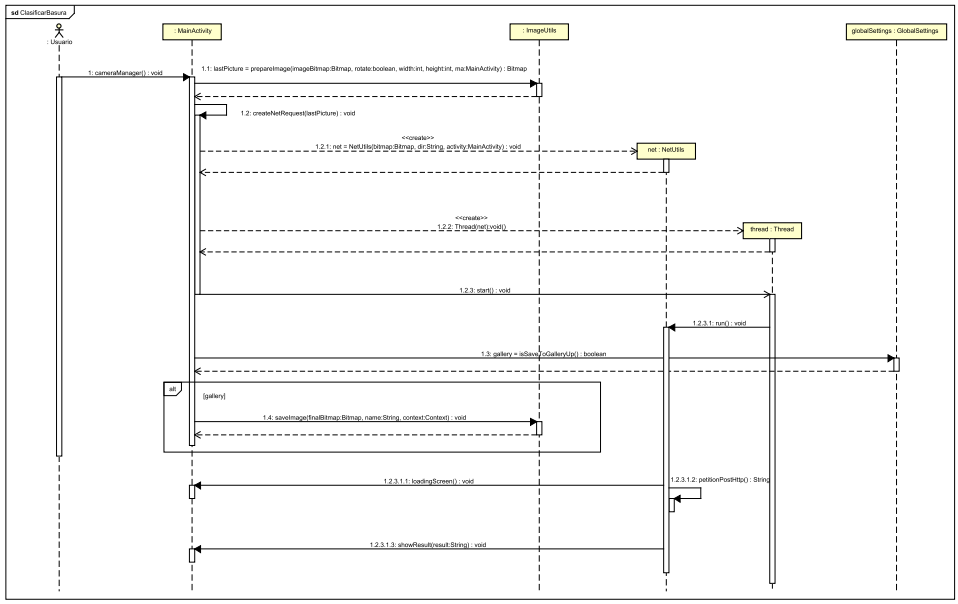


Figura 7.5: Diagrama de Secuencia Clasificación de Basura

7.6. Diseño de la interfaz Gráfica

A continuación se muestra de manera simplificada el flujo y el diseño de la aplicación, mostrando cada una de las vistas y sub vistas de estas (Ver 7.6 Diagrama de Secuencia de la Interfaz).

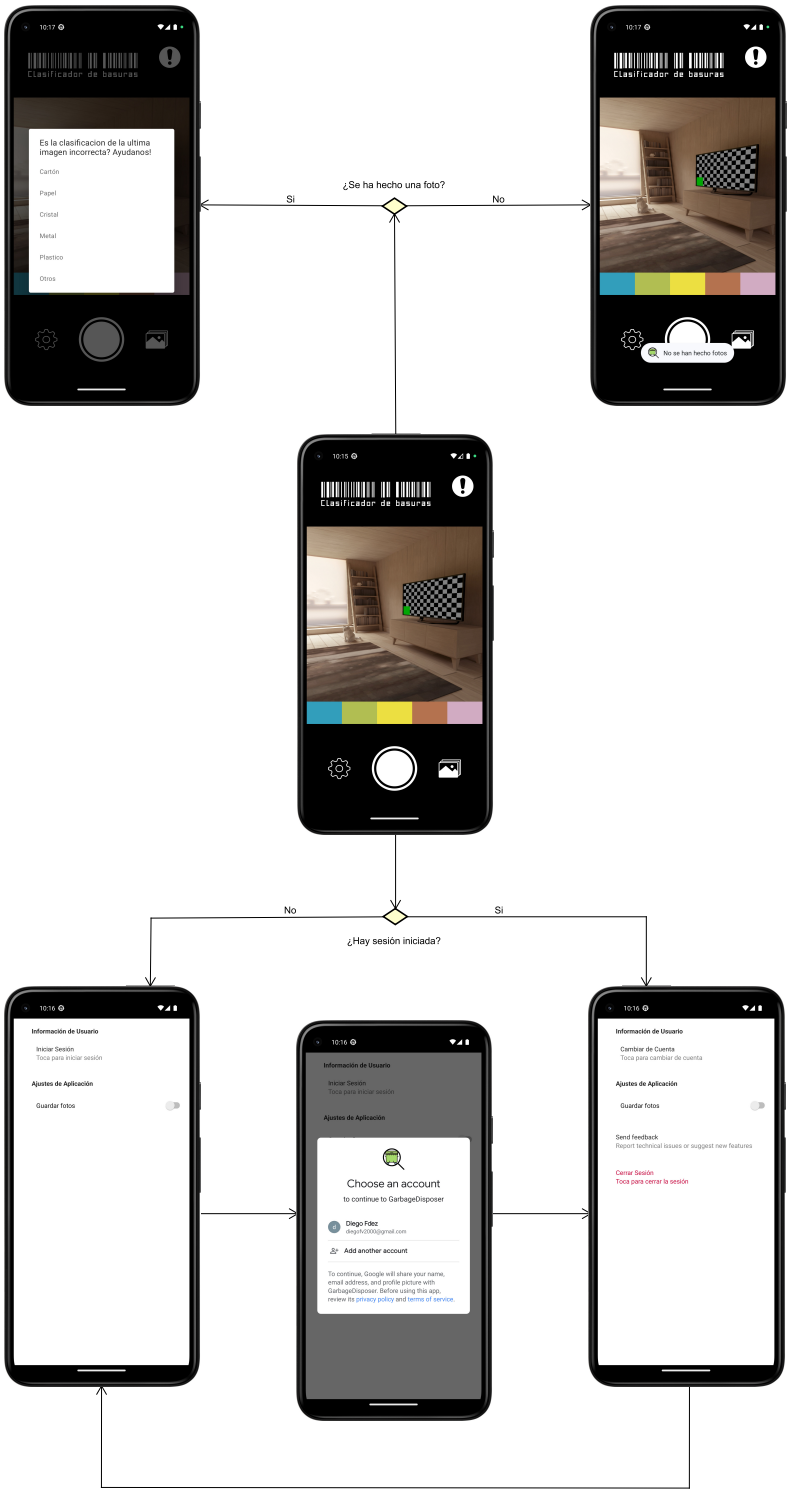


Figura 7.6: Diagrama de Secuencia de la Interfaz

7.7. Diseño de la base de datos

En este trabajo se ha decido emplear los servicios de Bases de Datos ofrecidos gratuitamente por Google. En concreto se han decido emplear Google FireBase FireStore en combinación con Google FireBase Storage (Ver. 8.7), siendo esta ultima para almacenar las Imágenes, ya que el modelo de Base de Datos de Google no admite BLOB o cualquier otro formato para almacenar imágenes. Otra de las motivaciones principales para emplear esta tecnología ha sido la asequibilidad que presentan para incorporarse desde Android Studio.

Aun que no cabe duda de la gran flexibilidad y comodidad que presenta el uso de estas tecnologías, se presentan un gran inconveniente a la hora de realizar un diseño de una Base de Datos tradicional, o relacional. Esto es debido a que las Bases de Datos de FireBase no son relacionales. Así pues, la implementación realizada de la misma no es relacional, pero se podría y se usa como una relacional en la practica.

Es por eso que se presentan dos diagramas respecto a la Base de Datos: El primero siendo el mas fidedigno a la verdadera implementación de esta (7.7 Diagrama no relacional de Base de Datos), y el segundo con ligeras modificaciones para entender como se relacionan las tablas entre si (7.7 Diagrama no relacional de Base de Datos).

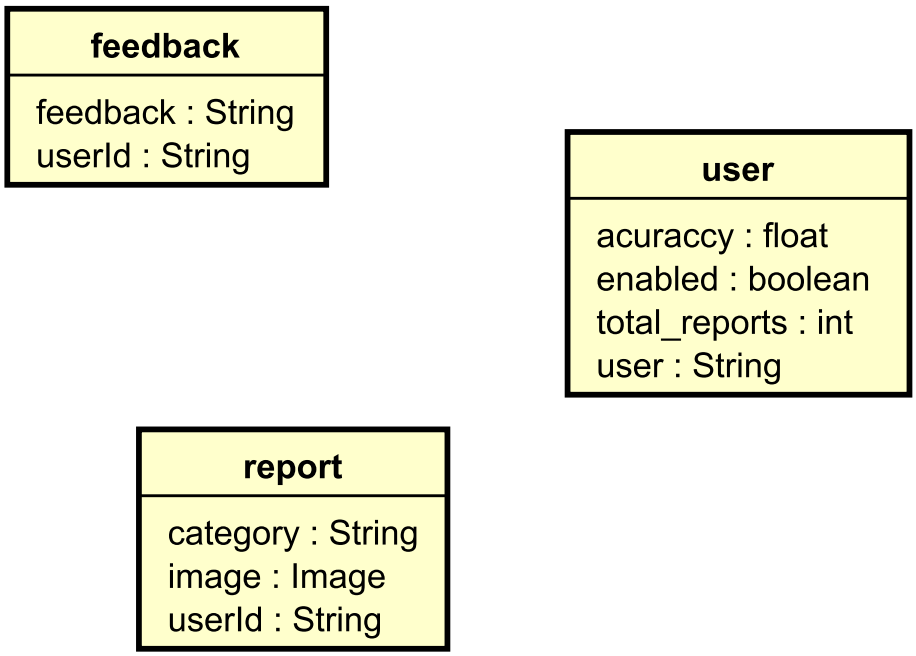


Figura 7.7: Diagrama no relacional de Base de Datos

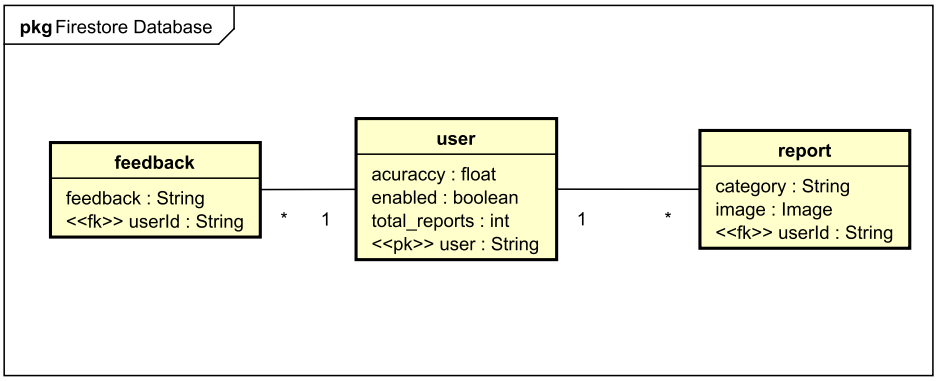


Figura 7.8: Diagrama relacional de Base de Datos

Como se puede apreciar la única diferencia es que en la Figura 7.7. cada tabla posee un id de Usuario, mientras que en el diseño relacional esta se omite, ya que se sobreentiende con la relación. De esta manera además, podemos especificar multiplicidades para lograr un mayor entendimiento de la funcionalidad.

Capítulo 8

Tecnologías utilizadas

8.1. Introducción

A continuación se mostrará un resumen de las tecnologías empleadas durante el desarrollo del proyecto, acompañándolas de una breve justificación del porque de su elección.

8.2. UML

UML (Unified Modeling Language) es un lenguaje de modelado gráfico, y un estándar que se utiliza para representar y documentar los componentes de un sistema software. Se ha utilizado como una herramienta para modelar el diseño en proyectos de software debido a su capacidad para describir de manera clara y visual las relaciones entre los diferentes componentes de un sistema, incluyendo clases, objetos, componentes y procesos.

En cuanto a su uso en este documento, UML se ha usado para representar y documentar el diseño y la arquitectura del sistema a desarrollar. Además, se ha utilizado para modelar las decisiones de diseño, explicando las razones detrás de cada elección y cómo se abordaron los problemas técnicos y de requisitos.

En resumen, UML es un lenguaje de modelado valioso en el proceso de desarrollo de software debido a su capacidad para representar y documentar de manera clara y visual los componentes de un sistema, facilitar la comunicación y la toma de decisiones y ayudar a identificar y resolver problemas de diseño desde una etapa temprana.

8.3. Selección del optimizador adecuado

El optimizador escogido ha sido el Adam, siendo el motivo detrás de este puramente funcional, pues tras revisar múltiples comparativas este siempre resultaba ser el mas rápido y además el que mayor porcentaje de acierto suele producir. (Si se desea conocer mas al respecto recomiendo la lectura de [14])

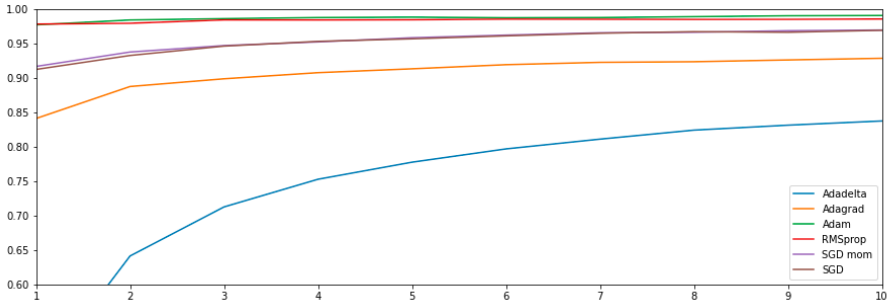


Figura 8.1: Comparación de Optimizadores

8.4. Pytorch

Para la elaboración del modelo de *Machine Learning* se decidió el uso del lenguaje de programación Python, ya que este es el que presenta un mayor abanico de opciones en lo que a librerías de *Machine Learning* concierne. Pese a que este paso resulta evidente, la elección de una librería dentro de la multitud de ellas que se ofrecen es mas complejo.

- TensorFlow
- Keras
- Pytorch
- Pandas
- Theano
- Scikit-learn
- SciPy

Estas solo son algunas de las opciones que Python presenta. Dada el temprano comienzo del TFG decidí probar a hacer diferentes prototipos con distintas librerías e ir determinando cual se adaptaría mejor a las necesidades de mi proyecto y a mis habilidades y conocimientos actuales. Así finalmente se decidió por usar Pytorch, ya que esta era una librería de la que fuí capaz de encontar multiples ejemplos y variada documentación.

Algunas de las ventajas de Pytorch son:

- Facilidad de aprendizaje.
- Una fuerte comunidad.
- Sencillez para la depuración de programas.
- Permite transportar modelos a tarjetas GPUs y dispositivos móviles. (Se hablará en detalle en 9.1.3 ¿Por qué usar una API REST?)

8.4.1. TorchVision

Dentro de la librería Pytorch, la dependencia encargada del *Machine Learning Visual* es TorchVision

EfficienNet

A la hora de entrenar una red neuronal para la clasificación de imágenes es usualmente mejor utilizar redes ya pre-entrenadas. Estas redes suelen estar entrenadas con un subconjunto de la base de datos ImageNet, y ya de por si suelen ser capaces de clasificar objetos en cientos de categorías.

En este caso vamos a emplear una red neuronal previamente entrenada para hacer lo que se conoce como *Transferencia del aprendizaje*: Utilizar las capas de una red ya entrenada para, con un nuevo pequeño conjunto de datos, acelerar el aprendizaje. (Si el conjunto de datos nuevo es demasiado grande puede que no acelere el aprendizaje).

En cuanto al porque de la elección de EfficientNet-B7 como el modelo pre-entrenado a emplear; la respuesta es sencilla, como podemos ver en la siguiente Figura 8.2, esta red presenta las siguientes ventajas:

- **Alta precisión:** EfficientNetB7 ha sido entrenado en una gran cantidad de datos de imágenes y ha obtenido un alto rendimiento en diferentes conjuntos de datos de referencia para la clasificación de imágenes. Por lo tanto, si estás buscando una red neuronal que tenga una alta precisión en la clasificación de imágenes, EfficientNetB7 puede ser una buena opción.
- **Eficiencia computacional:** EfficientNetB7 ha sido diseñado para ser altamente eficiente en términos de uso de recursos computacionales. Esto significa que puede realizar la clasificación de imágenes de manera rápida y precisa, sin necesidad de grandes cantidades de recursos de hardware. Si buscas una red neuronal que tenga un buen rendimiento pero que también sea eficiente en el uso de recursos, EfficientNetB7 podría ser una buena elección.

A continuación se muestra una comparación del rendimiento de múltiples redes neuronales de clasificación de imágenes relativas múltiples datasets (Para mas detalles ver este estudio [15])

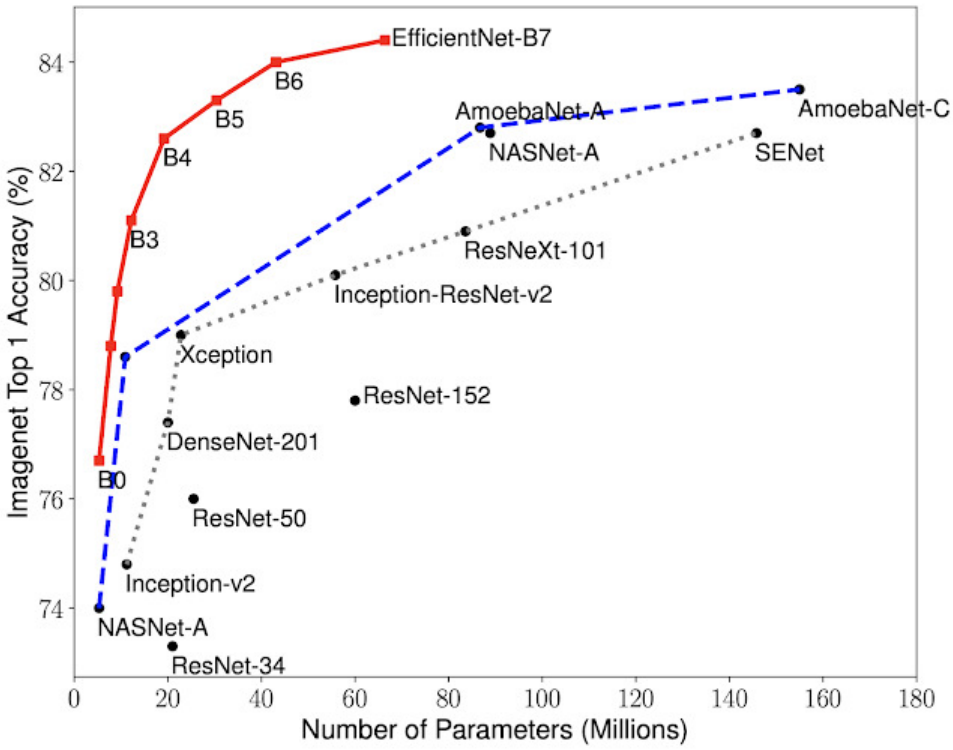


Figura 8.2: EfficientNet Performance - Fuente: [7]

ImageNet Transformation

Como se ha mencionado en el punto anterior la red neuronal EfficientNet-B7 esta entrenada sobre datasets de ImageNet, y es por ello que se ha decidió aplicar las mismas transformaciones a las imágenes a procesar.

Una transformación es un conjunto de modificaciones que se realizan a una imagen para cambiar propiedades de ella, estas pueden ser muy diversas y van desde normalización, hasta cambio de brillo, recorte o rotación. El objetivo de estas es múltiple:

- ***Data Augmentation.*** Ver 3.4.1 Data Augmentation.
- **Uniformidad de Datos:** Es gracias a estas transformaciones, que somos capaces de poder uniformar imágenes que provengan de distintas cámaras, con distintas resoluciones y relaciones de aspecto, aun que no es una solución perfecta nos ayuda en buena medida.

- ***Progresive Resizing*:** Es una técnica de aceleración del aprendizaje durante los primeros *epochs* y como su nombre indica, consiste en entrenar la red neuronal con inicialmente imágenes de pequeña resolución e ir incrementando esta según avanzamos los *epochs*. Aun que termino por descartarse su uso en este proyecto, se puede leer mas acerca de esto aquí [16].

8.4.2. Weights & Biases

Esta herramienta es uno de las principalmente empleadas para poder determinar si el rendimiento de los modelos y los métodos con los que se entrenan son adecuados; tomando el papel de *Pizarra* en el patrón BlackBoard (ver 7.2.2).

Este repositorio permite automatizar la recolección de datos durante el entrenamiento de los modelos. Así, de esta manera, podemos comparar que métodos han sido mas efectivos y poder sacar conclusiones sobre como continuar el entrenamiento del modelo. En cuanto a los datos que esta recolecta son cualquiera que nosotros queramos, siempre y cuando sea un tipo elemental de dato.

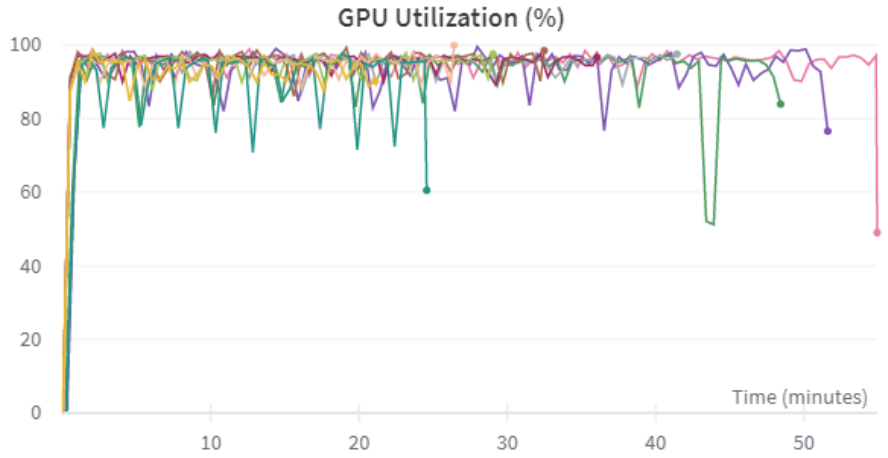
```
wandb.init(
      # Set the project where this run will be logged
      project = "New Dataset"
      # Track hyperparameters and run metadata
      ,config={
      "optimizer": "Adam",
      "model": "EfficientNet7",
      "learning_rate": learning_rate,
      "beta_1": beta_1,
      "beta_2": beta_2,
      "batch_size": batch_size,
      "epochs": epochs,
      "num_workers": num_workers,
      "image_size": image_size,
      "train_image_size": train_image_size,
      })
```

Figura 8.3: Ejemplo de entrada de datos Weights & Biases

Otra de las ventajas que supone el uso de esta herramienta es la facilidad para extraer los datos en formato CSV, o directamente generar gráficas desde la Web de la API. Un ejemplo

sencillo de aplicación de esta API durante la ejecución es asegurarse de que se maximice el uso de la GPU durante la fase de entrenamiento, de manera que se garantice la utilización de todos los recursos disponibles para acelerar el proceso de entrenamiento.

En las Figuras 8.4 y 8.5 siguientes podemos ver como, el paso de distintos *epochs* (Cada linea es una sucesión de 10 *epochs*) logramos maximizar el uso de GPU y su memoria. Pese a que no es la forma mas clara de mostrar estos datos en un gráfica, es el formato decidido pues es el que genera Weights & Biases y su objetivo principal es el ilustrar como se han ido realizando estos análisis según iban pasando las ejecuciones.



broadcast

Figura 8.4: Ejemplo de uso de GPU

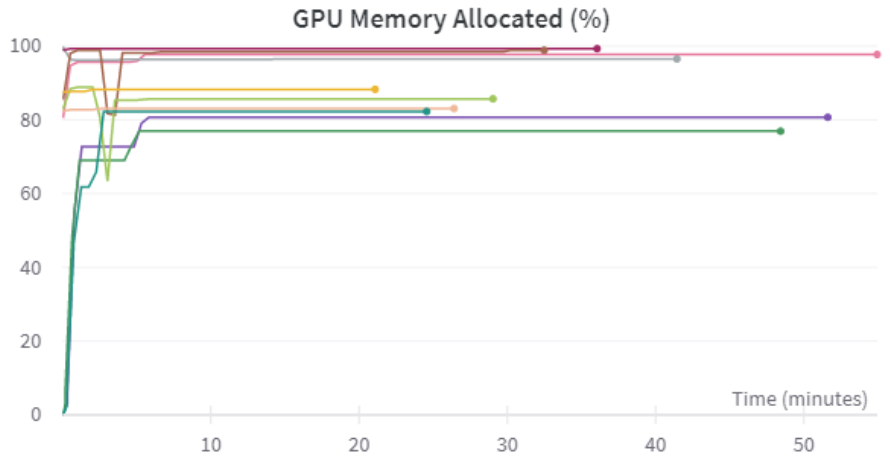


Figura 8.5: Ejemplo de uso de Memoria de la GPU

Otra de las multiples ventajas que ofrece esta herramienta es la de *generación de reportes*, mediante esta utilidad podremos generar colecciones de gráficos datos y textos para poder estudiarlos de manera mas cómoda, en la siguiente Figura 8.6 se muestra un ejemplo de un reporte generado tras entrenar un modelo usando 8.4.1 Progresive Resizing.

Comparación Tamaño Imagen - Batch

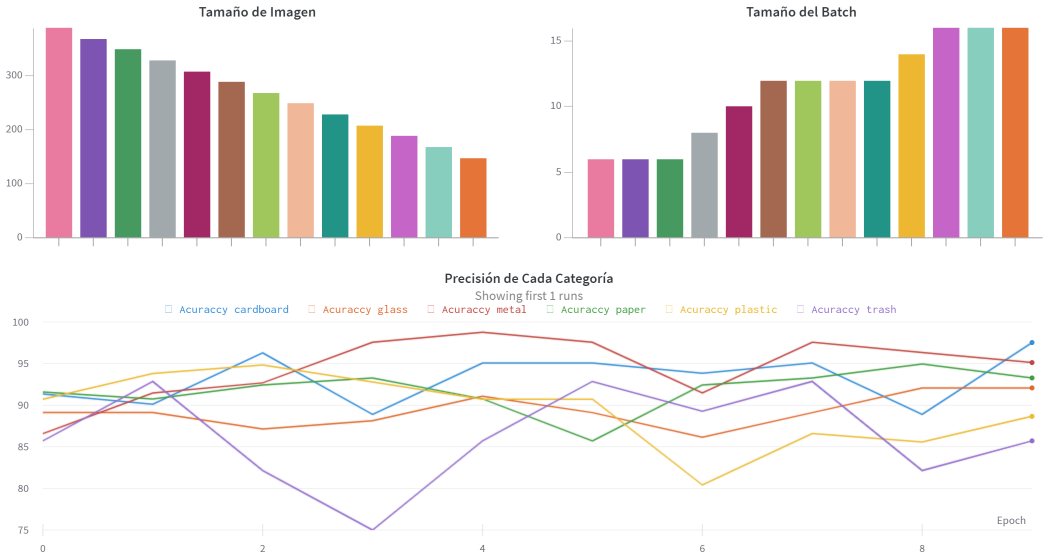


Figura 8.6: Reporte Weights & Biases

8.5. Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones Android. Se basa en IntelliJ IDEA, un entorno de desarrollo integrado Java para software, e incorpora sus herramientas de edición de código y para desarrolladores.

Para apoyar el desarrollo de aplicaciones dentro del sistema operativo Android, Android Studio utiliza un sistema de compilación basado en Gradle, un emulador de Android, plantillas de código e integración con GitHub. Cada proyecto en Android Studio tiene una o más modalidades con código fuente y archivos de recursos. Estas modalidades incluyen módulos de aplicaciones Android, módulos de bibliotecas y módulos de Google App Engine.

Es por esto y mas motivos que el desarrollo de esta aplicación Android se ha decidido llevar acabo en este IDE.

8.5.1. Camera2 API

Esta API para Android, permite obtener fotogramas en directo desde la Cámara de nuestro dispositivo móvil. Esto presenta una gran ventaja a la hora del diseño, pues el hecho de no tener que depender solo de la galería o de llamadas a la propia aplicación de Cámara del sistema permite generar una interfaz mucho mas personalizada y que se adapte a las necesidades actuales de la aplicación.

Sin embargo, esta mayor personalización viene con un coste adicional en la complejidad de su implementación, pues ahora se debe entender como administrar cada fotograma procesado por la cámara, entender mayores detalles de esta (Por ejemplo, las cámaras de los dispositivos Android soportan una lista prefijada de resoluciones y será necesario adaptarse a ellas) y hacer todo esto de manera que no sea demasiado costoso en términos de computación, pues bien un proceso costoso aunque puede realizarse, puede tener un impacto en la batería y temperatura del dispositivo (Cosa que es preferible evitar ya que reduciría el uso de estas aplicaciones en cualquier caso).

8.5.2. SharedPreferences

SharedPreferences es una librería de Android que permite guardar información, como opciones de usuario, de manera simple y eficiente en un archivo llamado "SharedPreferences". Es una forma de almacenar pequeñas cantidades de datos en un formato key-value, que pueden ser accedidos y modificados por diferentes componentes de la aplicación, como actividades o servicios.

SharedPreferences es una forma fácil y segura de guardar información que deba persistir entre sesiones de la aplicación, como preferencias de usuario, estados de la aplicación, o información necesaria para mantener el estado de la aplicación entre reinicios.

En resumen, SharedPreferences es una herramienta útil para guardar y acceder a pequeñas cantidades de información en una aplicación de Android, y es una alternativa más sencilla y ligera que otras opciones de almacenamiento persistente, como SQLite o archivos en el sistema de archivos.

8.6. Flask

Flask es un framework minimalista escrito en Python el cual permite alojar servicios web. Es por esta sencillez, y no mayor motivo, que se ha decido emplear esta tecnología para alojar el API y el servicio encargada de la clasificación de imágenes.

8.7. FireBase

Parafraseando a FireBase [17].

Firebase proporciona un conjunto de herramientas que permiten "construir, mejorar y hacer crecer aplicaciones", abarcando una amplia gama de servicios que normalmente deberían ser construidos por los desarrolladores, pero que prefieren evitar debido a que desean enfocarse en la experiencia de la aplicación en sí. Una variedad de servicios están cubiertos, incluyendo análisis, autenticación, bases de datos, configuración, almacenamiento de archivos, mensajería push, entre otros. Estos servicios se alojan en la nube y se expanden sin esfuerzo por parte del desarrollador.

Esta aplicación se focalizará en el uso de FireBase Storage y FireBase DataBase, de los cuales se habla en mas detalle a lo largo del documento.

8.8. Docker

Docker es una herramienta de software que permite la creación de aplicaciones en contenedores aislados del sistema operativo anfitrión, lo que facilita la portabilidad y evita posibles problemas de dependencias de software. La virtualización que ofrece Docker permite que las aplicaciones se puedan ejecutar en cualquier entorno que tenga instalado Docker.

Además, Docker es especialmente útil en el diseño de aplicaciones basadas en microservicios. Con esta arquitectura, cada servicio puede ser actualizado, escalado o eliminado de manera independiente sin afectar al resto del sistema. Docker permite detener y reiniciar los contenedores individuales sin afectar al resto del sistema, lo que significa que se pueden realizar tareas de mantenimiento o actualización de forma más rápida y sencilla.

En resumen, Docker es una herramienta que ofrece una mayor portabilidad, flexibilidad y modularidad para las aplicaciones, lo que facilita su desarrollo y mantenimiento. Docker es una solución útil para desarrolladores y administradores de sistemas que buscan una forma más eficiente de manejar aplicaciones complejas.

8.9. Pipenv

Pipenv es una herramienta de gestión de dependencias para aplicaciones Python que combina pip y virtualenv. Permite definir, instalar y administrar las dependencias del proyecto en un solo lugar, lo que facilita la creación de un entorno virtual de desarrollo aislado y reproducible.

Al utilizar Pipenv, los desarrolladores pueden acelerar el proceso de desarrollo al evitar problemas de compatibilidad de versiones, simplificar la instalación de dependencias y automatizar tareas comunes como la activación de entornos virtuales y la instalación de dependencias.

8.10. Google Colab

Colab, también conocido como "Colaboratory" es un servicio ofrecido por Google el cual permite programar y ejecutar código Python con un funcionamiento similar al de Jupyter Notebook.

8.10.1. CUDA Cores

Es comúnmente aceptado que las unidades de procesamiento gráfico (GPUs) tienen un desempeño sobresaliente en la aceleración de ciertos tipos de tareas, entre ellas el aprendizaje automático (*Machine Learning*), donde el uso de una GPU puede marcar una gran diferencia en la velocidad de ejecución.

En el caso particular de la librería Pytorch, se permite la aceleración a través del uso de GPU, pero esta solo es compatible con el lenguaje CUDA, el cual solo es compatible con tarjetas gráficas de la marca Nvidia. Afortunadamente, las GPU ofrecidas por Google son de la marca Nvidia, lo que permite aprovechar esta ventaja.

8.10.2. Google Drive Storage

Google Colab provee al desarrollador con una librería para comunicarse con Google Drive, esto es muy cómodo pues permite almacenar datos con la garantía de que no se perderán. El único inconveniente con esto, es que estamos limitados al tamaño que tengamos contratado con Google (15GB por defecto).

8.10.3. Algunas de las ventajas de esta tecnología son:

- No requiere configuración.
- Acceso a GPUs sin coste adicional.
- Permite compartir contenido fácilmente.
- No supone ningún coste en su versión gratuita, generalmente más que suficiente para proyectos de pequeña escala.
- Consecuentemente, ahorro energético al no tener que ejecutar localmente los códigos.
- Incluye una sintaxis especial de comentarios que permite ajustar valores de variables del código mediante controles externos (TextFields, Sliders, Checkboxs...)
- Facilidad para hacer Debugging.
- Permite el uso de TPUs.

8.10.4. Desventajas, en su versión gratuita

- Requiere la atención del programador durante su ejecución, pues puede saltar un Captcha, que de no ser respondido detendrá la ejecución.
- El uso de GPU, aunque gratuito, esta limitado por tiempo de cómputo.
- El uso de GPU, también viene condicionado por cuantas usuarios estén usándolas en ese momento.
- Dificultad para encontrar el Hardware en el que se esta ejecutando el código.
- Perdida total del estado de la máquina: todos los Archivos del S.O. a excepción del propio código, puesto que este no se almacena ahí.
- La CPU ofrecida sólo dispone de un núcleo.
- Tienes asignadas unas horas de computo máximas diarias.
- Estas son aleatorias y no son visibles por el usuario hasta que las hayas agotado.

Capítulo 9

Implementación y pruebas

9.1. Documento de Implementación

En la presente sección daremos mas detalles acerca de como se han implementado distintas partes del Software, acompañándolos de detalles mas técnicos como puede ser las versiones de librerías empleadas y la organización del mismo código en los distintos sub-sistemas.

9.1.1. ¿Qué tipos de residuos se van a clasificar?

La elección de las categorías sobre las que se ha entrenado el modelo vienen dadas por el DataSet empleado, en este la etiquetación provista abarca:

- **Cartón**
- **Papel**
- **Cristal**
- **Metal**
- **Plástico**
- **Otros**

Asi pues, se entrena el Modelo acorde a estas clasificaciones y una vez realizado el entrenamiento estas clasificaciones se agrupan en otros grupos, véase:

- **Contenedor Azul:** Bajo este grupo se encuentra tanto el *Papel* como el *Cartón*.

- **Contenedor Verde:** Bajo este grupo se encuentra la categoría *Cristal*.
- **Contenedor Gris:** Bajo este grupo se encuentra la categoría de *Metal* y *Plástico*.
- **Contenedor Amarillos:** Bajo este grupo se encuentra lo clasificados como *Otros*.
- **Contenedor Marrón:** Pese a que este este plenamente implementado, el modelo actual no clasifica nada relativo a este.

Es importante tener en cuenta que el motivo detrás de esta agrupación de clasificaciones es la de lograr una mayor flexibilidad, de manera que esta aplicación pueda ser empleada en regiones que tienen distintos criterios de reciclaje. Así teniendo una gran especificidad por parte de la clasificación del modelo esta aplicación podrá ser fácilmente ajustada (Dividido en los contenedores necesarios) para cuadrar con los requisitos de reciclaje de cada area.

Es este el motivo detrás de que cuando se reporte una imagen, las categorías ofrecidas sean con las que se ha entrenado el modelo y no los distintos tipos de contenedores.

9.1.2. Cómo y cada cuándo procesar imágenes

Al inicio del desarrollo, se centró en la idea de una aplicación que clasificaba continuamente los objetos en la cámara del dispositivo. Sin embargo, este método fue desechado debido a su alto costo en términos de consumo de batería y sobrecalentamiento del dispositivo móvil.

Finalmente, se optó por un enfoque más convencional en el que el usuario debe presionar un botón para procesar una imagen, lo que resolvió los problemas mencionados anteriormente.

9.1.3. ¿Por qué usar una API REST?

En primera instancia, es lógico pensar que la forma más conveniente de integrar el *Modelo de Aprendizaje Automático* en una aplicación sería incorporándolo directamente en la propia aplicación. Sin embargo, esto plantea varios problemas:

- **El tamaño de la aplicación** aumentaría significativamente, ya que estos modelos pueden ocupar varios gigabytes, lo que haría que la aplicación sea menos accesible para muchos usuarios.
- **La velocidad de procesamiento** del modelo estaría limitada por el dispositivo móvil, lo que lo haría menos accesible para dispositivos de gama baja.
- **Uso de la batería:** el cálculo de modelos tan complejos en un dispositivo con batería podría agotar considerablemente la duración de la batería.
- **Mantenimiento:** una arquitectura basada en microservicios ofrece una mayor flexibilidad de mantenimiento, sin necesidad de apagar todo el sistema para realizar el mantenimiento de una sola parte.

9.1.4. Formato de las peticiones a la API REST

```
POST /api/upload HTTP/1.1
Host: www.example.com
Content-Type: multipart/form-data; boundary=*****
Connection: Keep-Alive
Cache-Control: no-cache

--*****
Content-Disposition: form-data; name="file";filename="example.jpg"

[Contenido binario del archivo JPEG]
--******--
```

Figura 9.1: Ejemplo Petición API REST

9.1.5. Versiones del Software empleado

Nombre	Versión
compileSDK	**33**
androidx.appcompat	1.5.1
com.google.android.material	1.7.0
androidx.constraintlayout	2.1.4
androidx.legacy	1.0.0
androidx.preference	1.2.0
junit	4.13.2
androidx.test.ext	1.1.4
androidx.test.espresso	3.5.0
androidx.lifecycle	2.5.1
com.google.android.material	<latest-version>
com.google.firebase	31.1.1
com.google.android.gms	20.4.0

Tabla 9.1: Dependencias Aplicación Android.

Nombre	Versión
Python	**3.10.0**
certifi	2022.12.7
charset-normalizer	2.1.1
click	8.1.3
colorama	0.4.6
efficientnet-pytorch	0.7.1
enum-compat	0.0.3
flask	2.2.2
future	0.18.2
idna	3.4
itsdangerous	2.1.2
jinja2	3.1.2
markupsafe	2.1.1
numpy	1.23.5
packaging	22.0
pillow	9.3.0
psutil	5.9.4
requests	2.28.1
torch	1.13.1
torch-model-archiver	0.7.0
torch-workflow-archiver	0.2.6
torchserve	0.7.0
torchvision	0.14.1
typing-extensions	4.4.0
urllib3	1.26.13
waitress	2.1.2
werkzeug	2.2.2
wheel	0.38.4

Tabla 9.2: Dependencias API Rest.

9.1.6. Organización del código

Aplicación Android

El código se organiza principalmente 4 paquetes y dos clases adicionales:

- **Activities:** Este paquete contiene las clases relativas a todas las vistas de este proyecto, en nuestro caso solo son dos: MainActivty y SettingsActivty

- **CameraApi:** Contiene el código necesario para usar correctamente Camera2 API mencionado en REFERENCIA, ver https://hamzaasif-mobileml.medium.com/getting-frames-of-live-camera-footage-in-android-using-camera2-api-52cf4437f5fd.
- **Database:** Contiene la interfaz de acceso a la base de datos y su implementación. En este caso es la implementación con FireBase.
- **Utils:** Diversas colecciones de funciones organizadas temáticamente.
  - **ImageUtils:** Contiene funciones para realizar todo tipo de conversiones a imágenes, convertirlas a BitMaps e incluso guardarlas en la galería del dispositivo Android.
  - **NetUtils:** Es la encargada de hacer peticiones asíncronas a la API Rest, de no ser asíncrona la app se congelaría hasta conseguir una respuesta.
  - **GarbageUtils:** Almacena los distintos tipos de basura como Enum y provee conversión String $\longrightarrow$ Enum.

C_0 **AnimationManager:** Es la clase encargada de llevar acabo todas las animaciones que suceden en MainActivity.

C_1 **GlobalSettings:** Apoyándose en el uso de SharedPreferences, provee a la aplicación de una clase con llamadas uniformes a esta.

En cuanto al resto de documentos de este proyecto, estos de adhieren a la estructura estándar de Android Studio, dentro de la carpeta "res" encontramos multitud de recursos para la aplicación, como puede ser el diseño de las vistas, las fuentes empleadas, o svg's.

API REST

La organización de esta sección es inmediata pues solo cuenta con dos clases:

C_0 **REST:** Es la clase encargada de lanzar la API y su funcionalidad se reduce a escuchar ("POST") y responder a ellos.

C_1 **Modelo:** Es cargado al iniciar la API y se encarga de alimentar las imágenes al modelo y interpretar la respuesta.

Por otro lado esta API, cuenta con dos carpetas adicionales:

- **Models:** En esta carpeta se almacenan todos los posibles modelos que puede emplear la API. Para cambiar el modelo, actualmente, es necesario reiniciar la API.
- **Resources:** Su objetivo es puramente de Debugging, en esta carpeta se almacenan todas las fotos de peticiones enviadas al servidor.

Entrenador del Modelo

La organización de clases en este caso es inexistente, pues todo se encuentra dentro de un mismo archivo. Dada la naturaleza de Google Colab y como se suelen entrenar estos Modelos, esto se adhiere a lo estándar.

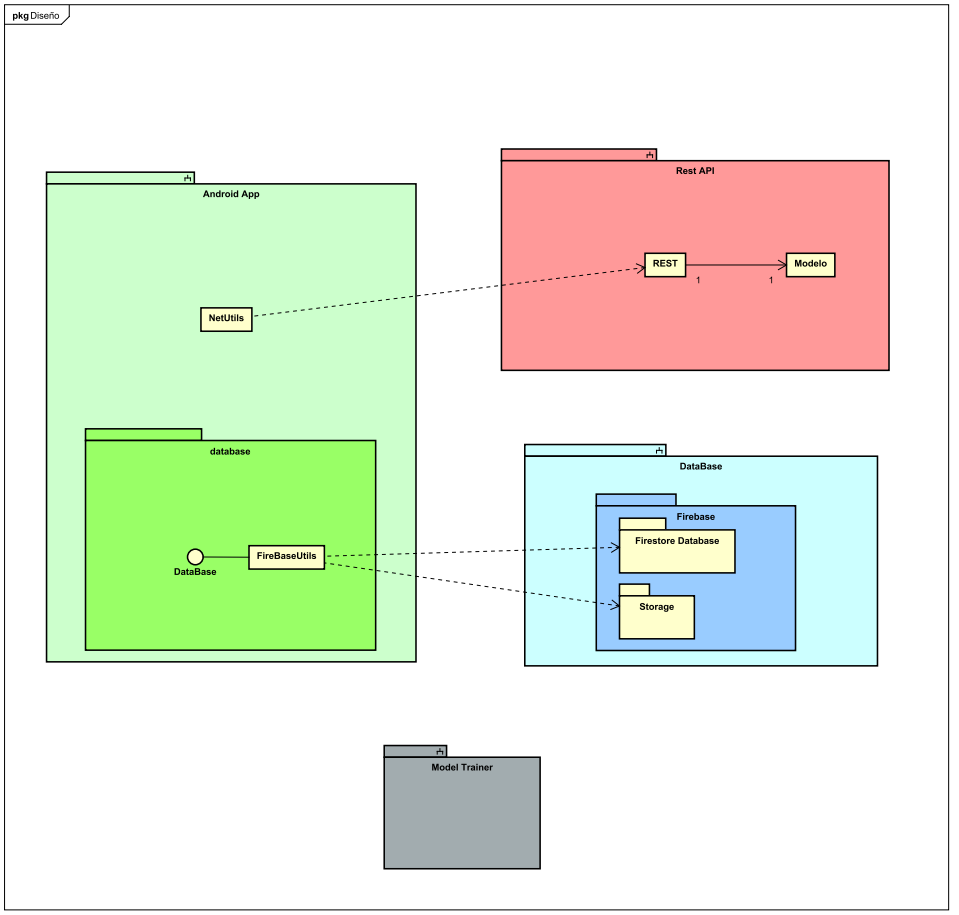


Figura 9.2: Diagrama Simplificado Organización General del Código

9.1.7. Buenas practicas empleadas

- **Creación de Interfaz para el acceso a la Base de Datos.** La creación de una interfaz nos da mucha mas flexibilidad a la hora de poder cambiar la Base de Datos en un futuro o la forma en la que esta se accede. Solo tendremos que proveer una clase que implemente dicha interfaz.
- **Encapsulación del uso de SharedPreferences.** El hecho de convertir el uso de esta librería a una clase, nos permite tener un estándar sobre que valores son modificables

y sus métodos para ello, además de simplificar la sintaxis.

9.2. Documento de Pruebas

Identificador:	5.1 Descripción del caso de uso "Clasificar Basuras" (Cámara).
Pre-condiciones:	Se han otorgado permisos de Cámara a la aplicación.
Entrada:	Foto tomada desde el dispositivo.
Salida Esperada:	Se mostrará una clasificación en pantalla.
Escenario	
1. Pulsar el botón central inferior con forma circular. **2.** Se mostrará una pantalla de carga.	

Tabla 9.3: Descripción de Prueba "Clasificar Basura (Cámara)"

Identificador:	5.1 Descripción del caso de uso "Clasificar Basuras" (Galería).
Pre-condiciones:	Se han otorgado permisos de acceso al almacenamiento a la aplicación.
Entrada:	Foto tomada de la galería del dispositivo.
Salida Esperada:	Se mostrará una clasificación en pantalla.
Escenario	
1. Pulsar el botón inferior derecho con forma de colección de fotos. **2.** Se mostrará una selector de Sistema de Galería. (Siempre y cuando el sistema cuente con más de uno) **3.** Seleccionar un lector de Galería. **4.** Seleccionar una imagen de la Galería.	

Tabla 9.4: Descripción de Prueba "Clasificar Basura (Galería)"

Identificador:	5.3 Descripción del caso de uso "Almacenar en Galería".
Pre-condiciones:	Se han garantizado permisos de almacenamiento a la aplicación.
Salida Esperada:	Se encontrará en la Galería la foto hecha.

Escenario

1. Pulsar en el botón con forma de engranaje de la esquina inferior izquierda.
2. Activar el campo "Guardar Fotos".
3. Volver a la vista principal pulsando la tecla de atrás en el dispositivo Android.
4. Tomar una foto, pulsando el botón circular del centro inferior de la aplicación.
5. Cerrar la aplicación y abrir la Galería de Android

Tabla 9.5: Descripción de Prueba "Almacenar en Galería"

Identificador:	5.6 Descripción del caso de uso "Reportar Imagen".
Pre-condiciones:	Se ha mandado una foto al servidor.
Entrada:	Categoría de basura: "Papel".
Salida Esperada:	Imagen reportada
Requisitos:	9.4 Descripción de Prueba "Clasificar Basura (Galería)" o 9.3 Descripción de Prueba "Clasificar Basura (Cámara)" 9.7 Descripción de Prueba "Autenticarse"

Escenario

1. Pulsar el botón en la esquina superior derecha con forma de exclamación.
2. Seleccionar la opción "Papel".

Tabla 9.6: Descripción de Prueba "Reportar Imagen"

Identificador:	5.5 Descripción del caso de uso "Autenticación".
Pre-condiciones:	No hay sesión iniciada en la aplicación.
Entrada:	Cuenta de Google generada para pruebas.
Salida Esperada:	La sesión estará iniciada y la vista de ajustes cambia por "Vista Sesión Ajustes"

Escenario

1. Pulsar en el botón con forma de engranaje de la esquina inferior izquierda.

1. Pulsar el campo de texto "Iniciar Sesión".

1. Seleccionar una cuenta de Google.

Tabla 9.7: Descripción de Prueba "Autenticarse"

Capítulo 10

Seguimiento del proyecto

Capítulo 11

Conclusiones

11.1. Recapitulación de los Objetivos

Los objetivos principales de este proyecto son dispares, abarcando desde el entrenamiento de un modelo de *Machine Learning* a su *hosteo* y posterior creación de una aplicación Android.

En cuanto a la tasa de cumplimiento de cada uno de los objetivos:

- **Modelo de *Machine Learning*:** Este se ha conseguido generar satisfactoriamente dejando espacio de mejora en futura iteraciones.
- ***Hosteo* del Modelo:** Este ha podido ser creado de manera satisfactoria cumpliendo plenamente todos los objetivos de: Portabilidad, Modularidad (Arquitectura de micro servicios) y facilidad de mantenimiento, sin necesidad de tener que reiniciar todo el sistema.
- **Aplicación Android:** La aplicación ha podido ser implementada con éxito, logrando hacer una aplicación que, mediante peticiones de red, se pueda comunicar con el servicio de *hosteo* del modelo de *Machine Learning* y lograr mostrárselo al usuario de una manera sencilla.

11.2. Implicaciones de los resultados

Gracias a la creación de una aplicación de esta modalidad, se ha podido identificar una nueva fuente de datos para creación de DataSets de todo tipo.

La mayor complicación planteada por este Modelo de Negocio, es la del correcto entrenamiento inicial de un Modelo de *Machine Learning*, que con un DataSet reducido inicial complica la tarea.

Una vez superada esta barrera y gracias a implicación de Usuarios iniciales dispuestos a generar multitud de reportes acerca los incorrectos comportamientos del Modelo de *Machine Learning*, se podrá generar Modelos que progresivamente se irán convirtiendo en más funcionales (Y gracias a la dispersa naturaleza de las correcciones de usuarios habrá casi una garantía de evitar el 3.4.2 Overfitting).

11.3. Limitaciones y recomendaciones

Como se ha mencionado previamente, el desarrollo de aplicaciones de esta naturaleza plantea un principal problema inicial. Este problema es, como en todo proyecto de *Machine Learning* la adquisición inicial de un suficiente conjunto de datos para entrenar estos Modelos.

Una recomendación es continuar con la estructura planteada en este proyecto, centrándose en un desarrollo por micro-servicios que facilite el mantenimiento y actualización de una aplicación de una naturaleza tan dinámica (Requiere de constantes actualizaciones de sus partes, sobre todo en la fase inicial).

11.4. Reflexiones

A continuación se presenta una lista de las reflexiones más relevantes que surgieron durante la realización del proyecto.

- **Agrupación de residuos.** La clasificación de estos es cada día mas compleja y presenta un gran abanico de opciones dependiendo de la región en la que se focalice. Por lo tanto, resulta crucial entrenar modelos con clasificaciones adicionales con el fin de poder categorizarlos en grupos, basándose en la legislación de reciclaje vigente en la región (ver 9.1.1).
- **Entrenamiento de Modelos.** El entrenamiento de cualquier tipo de modelo implica un proceso iterativo de prueba, error y repetición. Por lo tanto, se recomienda que la planificación del entrenamiento se realice con flexibilidad y se establezcan fechas límite amplias para permitir la exploración de múltiples técnicas de entrenamiento y garantizar un resultado óptimo.
- **Diseño auto explicativo.** La creación de una aplicación fácil de utilizar para cualquier usuario, especialmente en una aplicación de gran alcance cuyo público objetivo abarca todas las edades, puede plantear un desafío significativo y no debe ser subestimado.
- **Validación de Usuarios sencilla.** Continuando de lo previamente mencionado, la creación de un sistema de validación/autenticación de usuarios que elimine la presencia

de SPAM en el sistema de "reportes" de la aplicación y presente sencillez al usuario, debe ser planteado de manera sencilla, de aquí la recomendación de utilizar un sistema de Autenticación en vez de un sistema de Sesiones, que es mucho más complejo de implementar.

Apéndice A

Manuales

A.1. Manual de despliegue e instalación

A continuación se muestra los pasos a seguir para poder ejecutar todo el Sistema; se han intentado reducir los pasos a un mínimo.

A.1.1. Servicio REST

1. Instalación de Docker: Primeramente deberemos dirigirnos a la página oficial de "Docker" en instalarnos su ultima versión estable. Una vez descargado el ejecutable, nos limitaremos a seguir las instrucciones del instalador.

2. Ejecución de Docker: Una vez instalado, solo debemos ejecutar el comando "docker compose up". Para ello nos dirigiremos a la carpeta "ServicioMachineLearning Dockerizado", y dentro de esta ejecutaremos el comando previamente mencionado. En el caso de "Windows" se ha provisto de un archivo llamado "Ejecutar en Docker.bat", el cual ejecutara este comando por nosotros.

A.1.2. Aplicación Movil

Esta parte es bastante sencilla pues solo debemos abrir la carpeta "App" con Android Studio, y una vez en ella ejecutarla. Si se desea generar un nuevo archivo ".apk" solo debemos dirigirnos a la sección llamada: "Build" →"Build Bundle(s) / APK(s)" →"Build APK(s)". Y el "apk" será generado.

A.2. Manual de mantenimiento

A.2.1. Servicio REST

Para el mantenimiento se ha decido emplear "pipenv" en lugar de "Docker" dado que este es mucho mas rápido de recompilar y ejecutar:

1. **Instalación de "Python":** Primeramente deberemos instalar Python. Este método varía en función del S.O.
2. **Instalación de "pipenv":** Bastará con ejecutar el comando "pip install pipenv".
3. **Configuración de "pipenv":** Deberemos dirigirnos al directorio "ServicioMachine-Learning Dockerizado" y desde ahi abrir un terminal:
   a) **Localizar la ubicación** de nuestra instalación de "Python". En Windows podemos emplear el comando "where python".
   b) **Establecer la versión** de "Python" en "pipenv": Para ello ejecutaremos el comando "pipenv –python (la ubicación de nuestro python)".
   c) **Actualizar las dependencias:** Deberemos ejecutar "pipenv update".
4. **Ejecutar el Servicio:** Para ello solo deberemos de ejecutar el comando "pipenv run python -u REST.py", o en el caso de Windows ejecutar el archivo "LocalRun.bat" que se ha facilitado.
5. **Comprobar funcionamiento:** Para ello podemos servirnos de herramientas que nos permitan generar peticiones HTTP, en este caso se usará "Postman".
   a) **Seleccionar Petición tipo "POST".**
   b) **Introducimos la IP:** Para obtenerla nos dirigiremos a la consola en la que se esta ejecutando el Servicio.

```
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.1.200:5000
```

Figura A.1: Ejemplo Log Consola Servicio REST

   c) **Cuerpo de la Petición:** Nos iremos a la sección "Body", y en el campo KEY bajo el nombre "file" seleccionaremos el tipo "File" y adjuntaremos una imagen "jpg" en el campo value.
   d) **Enviar la Petición:** Pulsar el botón "Send".

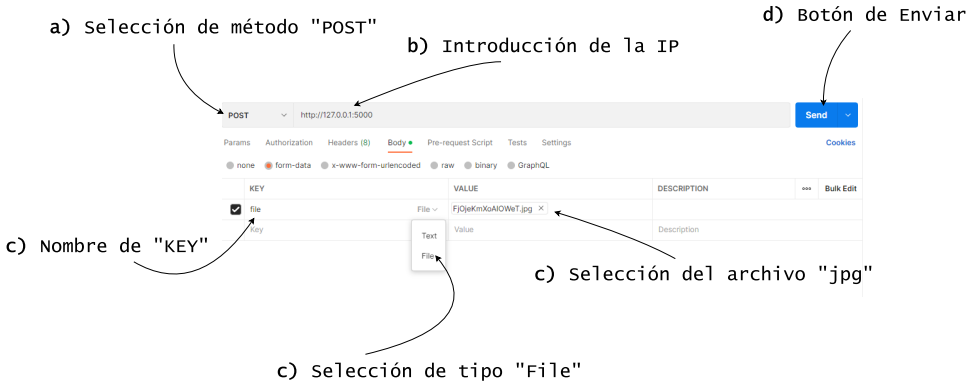


Figura A.2: Configuración Postman

Una vez ejecutados estos pasos, podremos hacer las modificaciones que consideremos pertinentes. Es importante destacar que para la instalación de dependencias en "pipenv" debemos hacerlo mediante el comando "pipenv install (nombre de la dependencia)".

También es importante destacar que si queremos que los cambios realizados a las dependencias de "pipenv" se proyecten en "Docker" deberemos de ejecutar el comando "pipenv requirements >requirements.txt", o de nuevo en el caso de Windows ejecutar el archivo "Actualizar Dependencia.bat" facilitado.

A.2.2. Aplicación Movil

Para el mantenimiento de la aplicación Android, se ha de tener un buen conocimiento sobre como se ha implementado y en especial el funcionamiento de Camera2API. Si damos esto por hecho lo único importante a destacar es el fichero "settings.xml" situado dentro de res/values: Este documento contiene los datos respectivos al uso de la API REST. Concretamente desde ese documento se puede especificar: Las dimensiones de la Imagen enviada a la API y la dirección IP de la misma.

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <item name="myUrl" type="string">http://192.168.1.200:5000/</item>
    <item name="image_height" type="string">512</item>
    <item name="image_width" type="string">512</item>
</resources>
```

Figura A.3: Datos configurable mediante Settings.xml

A.2.3. Modelo de Machine Learning

En cuanto al uso/mantenimiento del entrenador del Modelo todo esta suficientemente autodocumentado. Estos son los puntos principales en los que se divide este código:

1. **Instalaciones:** Se instalan todas las dependencias necesarias, en este caso:
   - **EfficienetB7.**
   - **W&B.**
2. **Descompresión del ZIP:** Se encarga de descomprimir un ZIP situado en "Google Drive".
3. **Estructuración en carpetas:** Estructura el anterior ZIP en carpetas.
4. **Copiamos el DataSet al entorno Local:** Copia directamente el DataSet de "Google Drive" ya estructurado, sin necesidad de descomprimirlo y organizarlo previamente.
5. **Entrenamiento**
   a) **Datos del entrenamiento:** Configuración de los datos mencionados en 8.3 Ejemplo de entrada de datos Weights & Biases
   b) **Nombre del proyecto:** Manda los datos previos a W&B y se establece el nombre del proyecto.
   c) **Decidimos si cargar un modelo viejo o no.**
   d) **Selección de Optimizador.**
   e) **Guardamos el Archivo:** Se exporta el modelo generado.
6. **Caso de Uso**
   a) **Cargamos el Modelo.** Esta sección autocontenida permite probar el Modelo. (Asegurarse que la imagen subida es un .jpg)

A.3. Manual de usuario

A continuación se mostrarán la serie de pasos a llevar acabo para lograr la primera clasificación usando la Aplicación "GarbageDisposer".

Primeramente, y una vez instalada la aplicación, se debe de abrir y aparecerá un Pop-Up pidiendo al Usuario permisos sobre la camara. A esto habrá que aceptar bajo criterio personal (Un solo uso, o siempre que la aplicación se esté en uso).

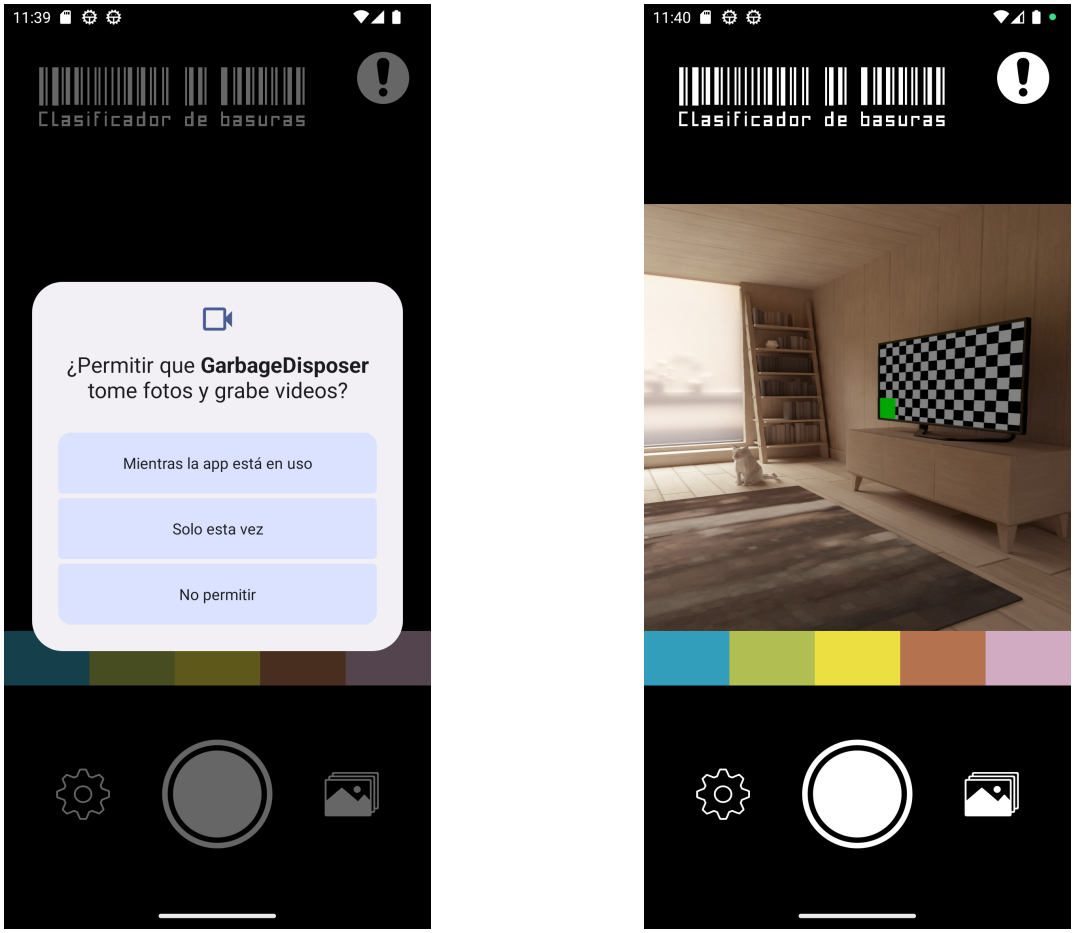


Figura A.4: Instrucciones: Garantizar acceso a Cámara

A continuación lo único que habrá que hacer es pulsar el botón circular blanco del centro inferior de la pantalla. Una vez hecho esto aparecerá un símbolo de carga y se esperará a que la aplicación retorne una clasificación. Una vez esta retorne, la tarea planteada habrá sido realizada con éxito. (La clasificación provista por la aplicación se mostrará en pantalla con una ligera animación justo debajo de la imagen en vivo de la cámara).

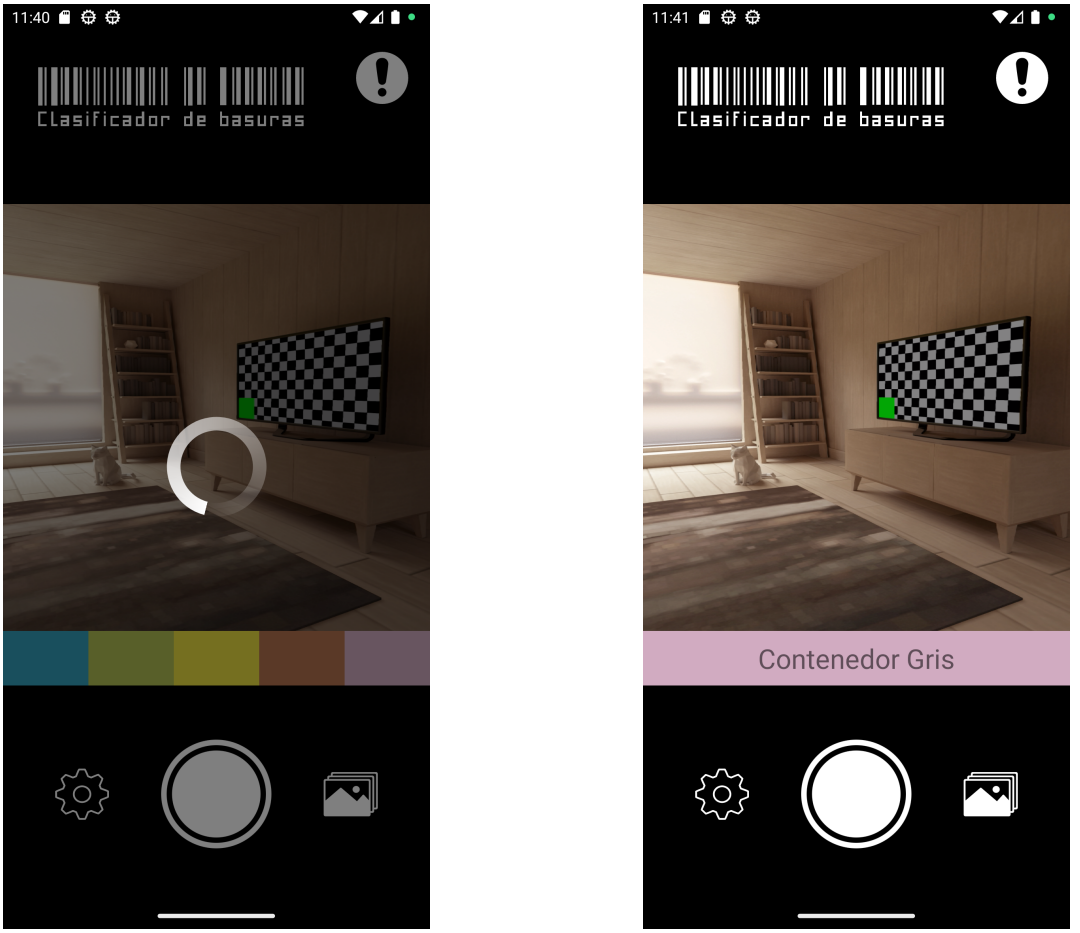


Figura A.5: Instrucciones: Clasificar Imagen

Es una vez realizada esta tarea cuando el Usuario podrá, si lo desea, reportar la clasificación provista como incorrecta, para lo cual tendrá que pulsar el símbolo de exclamación de la esquina superior derecha y pulsar en la categoría considerada correcta por el Usuario. (De no estar autenticado en el sistema se le solicitará al Usuario que lo haga)

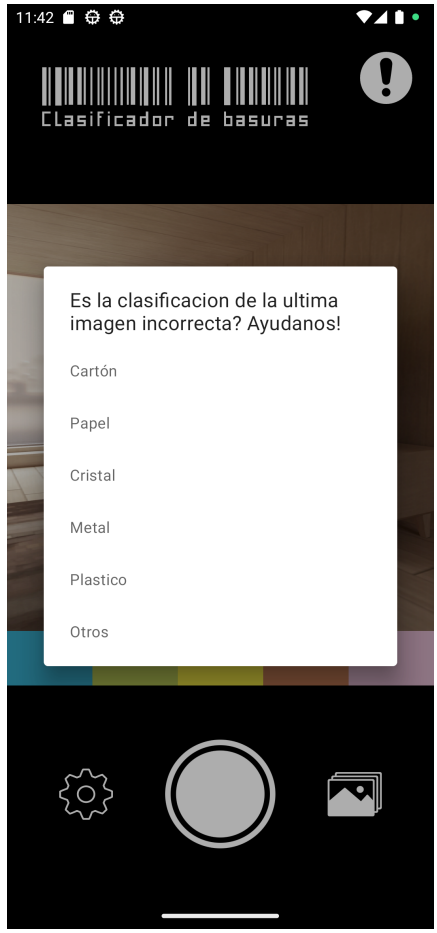


Figura A.6: Instrucciones: Reportar Imagen

Bibliografía

[1] Diego Calvo. Clasificación de redes neuronales artificiales. https://www.diegocalvo.es/clasificacion-de-redes-neuronales-artificiales/. Accessed:2023-02-15.

[2] Codificando Bits. Que es una red neuronal. https://www.codificandobits.com/blog/que-es-una-red-neuronal/. Accessed:2023-02-15.

[3] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 Accessed:2023-02-15.

[4] Parva Shah. Do we really need the pooling layer in our cnn architecture? https://www.linkedin.com/pulse/do-we-really-need-pooling-layer-our-cnn-architecture-parva-shah/. Accessed:2023-02-15.

[5] Sharon Y. Li. Automating data augmentation: Practice, theory and new direction. https://ai.stanford.edu/blog/data-augmentation/. Accessed:2023-02-15.

[6] Amey Gondhalekar. Data augmentation — is it really necessary? https://medium.com/analytics-vidhya/data-augmentation-is-it-really-necessary-b3cb12ab3c3f. Accessed:2023-02-15.

[7] Efficientnet: Rethinking model scaling for convolutional neural networks. https://arxiv.org/pdf/1905.11946.pdf, 2020.

[8] Jahaziel Ponce. Funciones de activiación y como puedes crear la tuya usando pytorch. https://jahazielponce.com/funciones-de-activacion-y-como-puedes-crear-la-tuya-usando-python-r-y-tensorflow Accessed:2023-02-15.

[9] Overtraining. https://reality.ai/machine-learning-projects-3-ways-to-make-it-succes#:~:text=Overtraining%20is%20when%20a%20machine,data%20that%20is%20too%20homogenous.

[10] garythung. Trashnet. https://github.com/garythung/trashnet. Accessed:2023-02-15.

[11] Google Firebase. Estimacion costes firebase. https://firebase.google.com/docs/firestore/billing-example?hl=es-419. Accessed:2023-02-15.

[12] Amazon Web Services. Precio host api rest. https://docs.aws.amazon.com/whitepapers/latest/best-practices-api-gateway-private-apis-integration/cost-optimization.html. Accessed:2023-02-15.

[13] Google Colab. Pricing training. https://colab.research.google.com/signup. Accessed:2023-02-15.

[14] Ayush Gupta. A comprehensive guide on deep learning optimizers. https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/. Accessed:2023-02-15.

[15] Efficientnet: Improving accuracy and efficiency through automl and model scaling - google ai blog. https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html. Accessed: 2023-02-08.

[16] Progresive resizing. https://www.fast.ai/posts/2018-04-30-dawnbench-fastai.html.

[17] Doug Stevenson. What is firebase? the complete story, abridged. https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0. Accessed:2023-09-02.

[18] Universidad de Valladolid. Proyecto docente del trabajo de fin de grado 2020-2021 (Mención Tecnologías de la Información). https://alojamientos.uva.es/guia_docente/uploads/2019/545/46977/1/Documento.pdf. Accessed: 2021-4-14.

[19] TT Inkscape. Inkscape. https://inkscape.org/es/, 2003. Accessed: 2021-4-5.

[20] Change Vision. Astah* professional. https://tecnicos.blogs.inf.uva.es/astah-professional/, 2020. Accessed: 2021-5-20.

[21] J Hammersley, J. & Lees-Miller. Overleaf. https://es.overleaf.com/, 2012. Accessed: 2021-2-15.

[22] Recycling image classification. http://web.cecs.pdx.edu/~singh/rcyc-web/dataset.html. Accessed: 2022-06-10.

[23] AgaMiko/waste-datasets-review: List of image datasets with any kind of litter, garbage, waste and trash. https://github.com/AgaMiko/waste-datasets-review. Accessed: 2022-06-10.

[24] Chris Fotache. Object detection and tracking in pytorch. https://towardsdatascience.com/object-detection-and-tracking-in-pytorch-b3cf1a696a98. Accessed: 2022-06-10.

[25] Chris Fotache. Training yolo for object detection in pytorch with your custom dataset — the simple way. https://towardsdatascience.com/training-yolo-for-object-detection-in-pytorch-with-your-custom-dataset-the-simple- Accessed: 2022-06-10.

[26] LearnOpenCV. Introduction to pytorch. https://learnopencv.com/pytorch-for-beginners-basics/. Accessed: 2022-08-10.

[27] Azure. Remove bg. https://dev.to/azure/opencv-10-lines-to-remove-the-background-in Accessed: 2022-08-10.

[28] Layan Alabdullatef. Complete guide to adam optimization. https://towardsdatascience.com/complete-guide-to-adam-optimization-1e5f29532c3d. Accessed: 2022-10-10.

[29] torch.optim — pytorch 1.12 documentation. https://pytorch.org/docs/stable/optim.html. Accessed: 2022-10-10.

[30] Davide Giordano. 7 tips to choose the best optimizer. https://towardsdatascience.com/7-tips-to-choose-the-best-optimizer-47bb9c1219e. Accessed: 2022-10-10.

[31] Lukas Melas-Kyriazi. lukemelas/EfficientNet-PyTorch: A PyTorch implementation of EfficientNet and EfficientNetV2 (coming soon!). https://github.com/lukemelas/EfficientNet-PyTorch. Accessed: 2022-10-10.

[32] Avinash. Pre-trained machine learning models vs models trained from scratch. https://heartbeat.comet.ml/pre-trained-machine-learning-models-vs-models-trained-from-scratch-63e079ed648f. Accessed: 2023-02-22.

[33] What is the difference between (objective / error / criterion / cost / loss) function in the context of neural networks? https://datascience.stackexchange.com/questions/10250/what-is-the-difference-between-objective-error-criterion-cost-loss-fun. Accessed: 2023-02-22.

[34] PyTorch/XLA. PyTorch on XLA devices. https://pytorch.org/xla/release/1.12/index.html. Accessed: 2023-02-22.

[35] Getting started with pytorch on cloud tpus. https://colab.research.google.com/github/pytorch/xla/blob/master/contrib/colab/getting-started.ipynb#scrollTo=42avAvSg17by. Accessed: 2023-02-22.

[36] Image classification via fine-tuning with efficientnet. https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/. Accessed: 2023-02-22.

[37] How does the "number of workers"parameter in pytorch dataloader actually work? https://stackoverflow.com/questions/53998282/how-does-the-number-of-workers-parameter-in-pytorch-dataloader-actually-work. Accessed: 2023-02-22.

[38] Speed up model training – pytorch lightning 1.8.0rc0 documentation. https://pytorch-lightning.readthedocs.io/en/latest/guides/speed.html. Accessed: 2023-02-22.

[39] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Himanshu Mehta, Tianxing Duan, Dylan Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, et al. The effect of image resolution on deep learning in radiography. https://pubs.rsna.org/doi/epdf/10.1148/ryai.2019190015, 2019.

[40] Baeldung. Relation between learning rate and batch size. https://www.baeldung.com/cs/learning-rate-batch-size, 2022.

[41] Mendeley Data. Data augmentation. https://data.mendeley.com/datasets/n3gtgm9jxj/2, 2022.

[42] H. S. Lee. Comparison of top-1 error on cifar10, cifar100, svhn, and imagenet dataset. https://www.researchgate.net/figure/Comparison-of-top-1-error-on-CIFAR10-CIFAR100-SVHN-and-ImageNet-dataset_tbl1_329945702, 2018.

[43] PyTorch. Autoaugment — torchvision main documentation. https://pytorch.org/vision/main/generated/torchvision.transforms.AutoAugment.html#torchvision.transforms.AutoAugment, 2022.

[44] Anand Borad. Image classification with efficientnet: Better performance with computational efficiency. https://datamonje.medium.com/image-classification-with-efficientnet-better-performance-with-computational-effici 2022.

[45] PyTorch. Efficientnetv2 — torchvision main documentation. https://pytorch.org/vision/main/models/efficientnetv2.html, 2022.

[46] Mingxing Tan and Quoc V Le. Efficientnetv2: Smaller models and faster training. *arXiv preprint arXiv:2104.00298*, 2021.

[47] Models and pre-trained weights — torchvision main documentation. https://pytorch.org/vision/main/models.html.

[48] Efficientnetv2 - weights & biases. https://wandb.ai/wandb_fc/pytorch-image-models/reports/EfficientNetV2--Vmlldzo2NTkwNTQ.

[49] Google. efficientnetv2. https://github.com/google/automl/tree/master/efficientnetv2, 2022. GitHub repository.

[50] Eddie. Image net preprocessing using torch transforms. https://stackoverflow.com/questions/67185623/image-net-preprocessing-using-torch-transforms, 2022.

[51] Normalize data: Component reference - azure machine learning — microsoft learn. https://learn.microsoft.com/en-us/azure/machine-learning/component-reference/normalize-data.

[52] AndroidKt. Confidence of answer. https://androidkt.com/use-saved-pytorch-model-to-predict-single-and-multiple-images/.

[53] Adam. Easily install jupyter notebook on ubuntu (step by step). https://adamtheautomator.com/install-jupyter-notebook-on-ubuntu/.

[54] Unknown. The blackboard pattern for autonomous navigation. https://dzone.com/articles/the-blackboard-pattern-for-autonomous-navigation, 2022. Accessed: 28/10/2022.

[55] Unknown. Swiftui mvvm with practical examples. https://blog.techchee.com/swiftui-mvvm-with-practical-examples/, 2022. Accessed: 28/10/2022.

[56] Unknown. mvvm-swiftui-architecture.png. https://blog.techchee.com/wp-content/uploads/2021/02/mvvm-swiftui-architecture.png, 2022. Accessed: 28/10/2022.

[57] Unknown. How to manage your python projects with pipenv. https://thoughtbot.com/blog/how-to-manage-your-python-projects-with-pipenv, 2022. [Online; accessed 6-December-2022].

[58] Unknown. Python and rest apis: Interacting with web services - real python. https://realpython.com/api-integration-in-python/, 2022. [Online; accessed 6-December-2022].

[59] Hamza Asif. Getting frames of live camera footage as bitmaps in android using camera2 api. https://hamzaasif-mobileml.medium.com/getting-frames-of-live-camera-footage-in-android-using-camera2-api-52cf4437f5fd, 2022. [Online; accessed 7-December-2022].

[60] Unknown. Live camera footage in android with camera2 api — camera2 api tutorial — camera2 api android studio. https://www.youtube.com/watch?v=HWHpE18t-fc, 2022. [Online; accessed 7-December-2022].

[61] Unknown. Sending files using post with httpurlconnection. https://stackoverflow.com/questions/11766878/sending-files-using-post-with-httpurlconnection, 2022. [Online; accessed 10-December-2022].

[62] Tipos de residuos. https://www.europapress.es/castilla-y-leon/noticia-ayuntamiento-valladolid-asegura-capitales-todos-contenedores-posibles-to
html#:~:text=Se%20trata%20del%20azul%20(papel,instalado%20para%20los%20envases%20ligeros. Accessed: 2022-11-12.

[63] Paleta. https://www.color-hex.com/color-palette/29800. Accessed: 2022-11-12.

[64] 7-frameworkscomponentes.pdf. https://campusvirtual.uva.es/pluginfile.php/3423175/mod_resource/content/1/7-frameworksComponentes.pdf. Accessed: 2022-12-13.

[65] Cómo obtener información de perfil — authentication — google developers. https://developers.google.com/identity/sign-in/android/people. Accessed: 2022-12-18.

[66] Uml class diagrams. http://www.cs.utsa.edu/~cs3443/uml/uml.html. Accessed: 2022-12-24.

[67] Why pytorch is the deep learning framework of the future. https://blog.paperspace.com/why-use-pytorch-deep-learning-framework/. Accessed: 2023-02-08.

[68] Efficientnet: Rethinking model scaling for convolutional neural networks. https://arxiv.org/abs/1905.11946v5. Accessed: 2023-02-08.

[69] Efiicenet b7. https://arxiv.org/pdf/1905.11946v5.pdf. Accessed: 2023-02-08.

[70] Redes neuronales profundas preentrenadas - matlab & simulink - mathworks españa. https://es.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html. Accessed: 2023-02-08.

[71] Simplilearn. What is epoch in machine learning? https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-epoch-in-machine-learning#:~:text=An%20epoch%20is%20when%20all,dataset%20takes%20around%20an%20algorithm. Accessed:2023-09-02.