UNIVERSIDAD PÚBLICA DE EL ALTO

CARRERA INGENIERÍA DE SISTEMAS

PROYECTO DE GRADO

APLICACIÓN MOVIL EN EL RECONOCIMIENTO DE PLAGAS EN PLANTACIONES DE CAFÉ APLICANDO REDES NEURONALES CONVOLUCIONALES

CASO: Empresa "APEN" Asociación de Productores de Café Nogalani

Para Optar al Título de Licenciatura en Ingeniería de Sistemas

MENCIÓN: INFORMÁTICA Y COMUNICACIONES

Postulante: Judith Pillco Ticona

Tutor Metodológico: Lic. Ing. Helen Fanny Suntura Escobar

Tutor Especialista: M.SC. Lic. Patricio José Ilaluque Vargas

Tutor Revisor: Lic. Rosa Patricia Nina Chura

EL ALTO – BOLIVIA

2024

DECLARACIÓN JURADA DE
AUTENTICIDAD Y RESPONSABILIDAD

Yo, Judith Pillco Ticona, estudiante de Taller de Grado II, de la Carrera de Ingeniería de Sistemas de la Universidad Pública de El Alto, identificado(a) con CI. 6982460 y Registro Universitario: 14000322.
Declaro bajo juramento que:

1. Soy autor del Trabajo de Grado titulado:
“Aplicación móvil en el reconocimiento de plagas en plantaciones de Café Aplicando redes neuronales convolucionales”
El mismo que presento bajo la modalidad de Proyecto de grado.

2. El texto de mi proyecto de grado respeta y no vulnera los derechos de terceros, incluidos los derechos de propiedad intelectual. En tal sentido, declaro que este Proyecto de Grado no ha sido plagiado total ni parcialmente, para lo cual he respetado las normas internacionales de citas y referencias de las fuentes consultadas.

3. El texto del Proyecto de Grado que presento no ha sido publicado ni presentado antes en cualquier medio electrónico o físico.

4. Por tanto, declaro que mi Proyecto de Grado cumple con todas las normas de la carrera de Ingeniera de Sistemas de la Universidad Pública de El Alto.

El incumplimiento de lo declarado da lugar a responsabilidad del declarante, en consecuencia; a través del presente documento asumo frente a terceros, la carrera de Ingeniera de Sistemas de la Universidad Pública de El Alto toda responsabilidad que pueda derivarse por el Proyect de Grado presentado.

El Alto, Noviembre de 2024

……
JUDITH PILLCO TICONA
C.I. 6982460 LP.

DEDICATORIA

Este proyecto va dedicado a Dios y a toda y las que depositaron un voto de confianza en mis capacidades en el transcurrir de los años en mi preparación y formación como profesional. Gracias por cada gesto de aliento en los momentos de dificultades.

A mis familiares y parientes de los cuales he llegado a recibir un apoyo incondicional. En especialmente a mi Familia mi hijo Gael Roland, esposo Rolando Tola Ticona y a mi Madre Martha Ticona Condori, quien siempre ha sido mi fuente de inspiración, luz y compañía, quien siempre me acompaña en mi largo caminar como una presencial invisible que guía mis días.

AGRADECIMIENTO

En primer lugar, agradezco a Dios por darme la fuerza, la salud y la sabiduría necesarias para completar este proyecto. Sin Su guía y bendiciones, no habría sido posible alcanzar este logro.

A mi madre y esposo por el deseo de superación y amor que me brindan cada día por animarme y no darme por vencida, no estaría aquí sin su guía y apoyo en la cual fueron una fuente de motivación para seguir adelante.

RESUMEN

El presente trabajo de grado se desarrolló en el Departamento de la Paz, del Municipio la Asunta, en la Asociación de Productores Ecológicos Nogalani, el problema que se encontró fue que los agricultores de café tienen la dificultad de detectar las plagas que afectan a sus plantaciones de café.

La producción del cultivo de café enfrenta un gran desafío debido a las plagas que afectan su producción y calidad. El reconocimiento temprano y preciso de estas plagas es esencial para implementar acciones de control efectivas y minimizar las pérdidas. Este proyecto propone una solución basada en una aplicación móvil que emplea redes neuronales convolucionales (CNN).

La aplicación permite a los agricultores capturar imágenes de las plantas afectadas, las cuales son procesadas mediante un modelo de CNN entrenado para reconocer dos tipos de plagas que son roya y arañita roja los más comunes del sector. Esta herramienta está diseñada para funcionar de manera eficiente en dispositivos móviles, optimizando el uso de recursos y asegurando una alta precisión en el reconocimiento bajo diferentes condiciones de imagen.

Se implementaron pruebas de estrés para evaluar el rendimiento de la aplicación en distintos escenarios de carga, además de pruebas de calidad (caja negra y caja blanca) para asegurar su robustez y facilidad de uso.

Este desarrollo representa un avance en el uso de inteligencia artificial aplicada a la agricultura, ofreciendo a los productores una herramienta accesible y poderosa para el monitoreo y control de plagas en tiempo real.

Palabras Claves: CNN, Plagas, Precisión, Aplicación, Rendimiento.

ABSTRACT

This thesis was developed in the Department of La Paz, in the Municipality of La Asunta, in the Association of Ecological Producers Nogalani. The problem that was found was that coffee farmers have difficulty detecting pests that affect their coffee plantations.

Coffee production faces a great challenge due to pests that affect its production and quality. Early and accurate recognition of these pests is essential to implement effective control actions and minimize losses. This project proposes a solution based on a mobile application that uses convolutional neural networks (CNN).

The application allows farmers to capture images of affected plants, which are processed by a CNN model trained to recognize two types of pests, which are rust and red spider, the most common in the sector. This tool is designed to work efficiently on mobile devices, optimizing the use of resources and ensuring high recognition accuracy under different image conditions.

Stress tests were implemented to evaluate the application's performance in different load scenarios, as well as quality tests (black box and white box) to ensure its robustness and ease of use.

This development represents a step forward in the use of artificial intelligence applied to agriculture, offering producers an accessible and powerful tool for monitoring and controlling pests in real time.

Keywords: CNN, Pests, Precision, Application, Performance.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

LISTADO DE ABREVIATURAS

CNN: Redes Neuronales Convolucionales

APEN: Asociación de Productores de Café Nogalani

ISE- OO: Herramienta que permite controlar y conocer los usuarios y dispositivos de una red

ISO/IEC: Norma más conocida del mundo para sistemas de gestión de la seguridad de la información.

JDK: Paquete de software que permite a los desarrolladores crear aplicaciones basados en Java.

RFN: Requisitos no funcionales.

IDE: Entorno de Desarrollo integrado

PIP: Sistema de gestión de paquetes

OPENCV: Biblioteca libre de visión artificial

ISTQB: Certificación de pruebas de software

XAML: Lenguaje declarativo

MOCKS: Función real de un programa

IU: Interfaz de usuario

ML: Aprendizaje automático

APP: Programa informático

CAPÍTULO I

MARCO PRELIMINAR

1. CAPÍTULO I - MARCO PRELIMINAR

1.1 INTRODUCCIÓN

Las aplicaciones móviles en la actualidad son muy utilizadas gracias a las facilidades de acceso a internet, así como los avances tecnológicos de teléfonos inteligentes, estos cuentan con sistemas operativos que facilitan desarrollar aplicaciones gratuitas que se pueden instalar en un dispositivo móvil sin ningún problema. Al realizar un análisis de los beneficios que ofrece la tecnología, se propuso una aplicación móvil con redes neuronales para los productores cafetaleros APEN la aplicación brindara la detección de enfermedades más comunes en plantas.

En el proyecto se identificó el problema, los antecedentes y la situación actual del problema. Con lo cual se determinan los objetivos tanto generales y específicos para luego obtener las justificaciones, la metodología, las herramientas que utilizaran.

Actualmente los productores de APEN. Tienen dificultad de identificar con exactitud las enfermedades que atacan a las plantaciones de café, a la causa identificada se propone una App móvil incorporado el sistema de inteligencia artificial que usará redes neuronales convolucionales para apoyar a los agricultores en el tratamiento de la planta de café y permitirá tomar mejores decisiones a los agricultores en la detección de enfermedades.

Ante la situación y aprovechando los grandes beneficios que brinda el uso de la tecnología, el proyecto realizado se enfoca en ayudar a que esta situación

mejore para los caficultores, un método más eficiente, con ello brindar un servicio óptimo en la agricultura.

Posteriormente se desarrollará la aplicación móvil, paso a paso con el uso de las diferentes herramientas de software y un almacenamiento de base de datos donde se encontrará la información requerida para los cafetaleros.

Finalmente, por lo último se describirá los resultados, de la misma manera las conclusiones y recomendaciones sobre el uso de la aplicación móvil.

1.2 ANTECEDENTES

En antecedentes, se desglosa los antecedentes institucionales y Antecedentes afines que aportan al proyecto de grado.

1.2.1 Antecedentes Institucionales

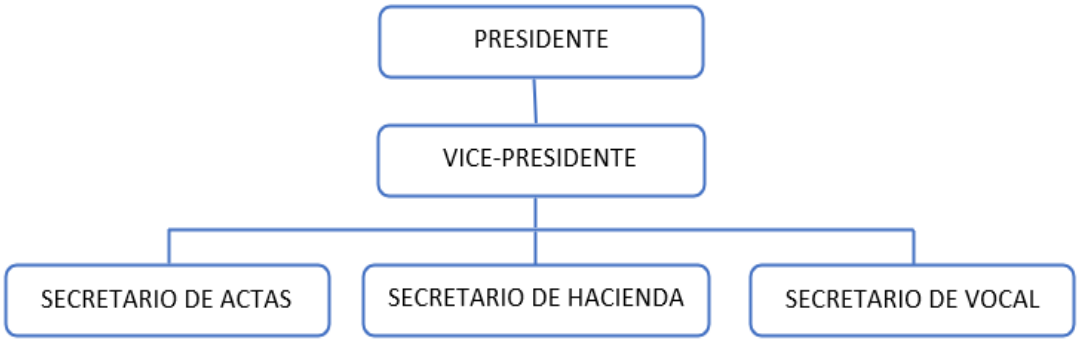
La asociación APEN fue creada el 12 de junio de 2004, dedicando al acopio del café guinda de los productores socios, posterior la exportación a países europeos del grano de oro. También la asociación tiene la certificación del producto orgánico, trabajando con la fundación internacional UNODC que apoyo la certificación de la producción ecológico grano de café. Un 28 y 29 de julio del 2015, fue experimentado y reconocidos, evaluaron y la aprobación de la certificación. La Asociación de Productores café ecológico Nogalani (APEN) realizado por (UNODC, 2024, p. 5).

1.2.1.1 Objetivo General

- Llegar a mejorar los ingresos y auto abastecimiento en los productores a través de una agricultura sostenible.
- Sostener la producción de manera indefinida.
- Alimentar a toda la población.

  Ofrecer una vida digna para las familias de las diferentes comunidades.

***Figura 1**: Organigrama de la Asociación APEN*



Nota: Organigrama de la Asociacion, APEN, 2004,Plan Estrategico de de la Asociacion.

1.2.2 Antecedentes Afines al Proyecto

En los antecedentes afines al proyecto se separa en antecedentes internacionales, antecedentes nacionales y antecedentes locales.

1.2.2.1 Antecedentes Internacionales

Los antecedentes Internacionales son:

Tesis: “Aplicación Móvil usando redes neuronales convolucionales para la detección de enfermedades en las plantas de café en el distrito Chirino, provincia de San Ignacio, departamento de Cajamarca” escrito por (Flores Mendoza & Carhuajulca, 2023) de la Universidad nacional Pedro Ruiz Gallo. La presente investigación tiene como importancia, la implementación de un sistema de inteligencia artificial que usará redes neuronales convolucionales, para apoyar a los agricultores en el tratamiento de la planta de café, y permitirá tomar mejores decisiones a los agricultores en la detección de enfermedades teniendo una mayor información de cada una de ellas para un mejor uso y correcto de agroquímicos.

- Tesis: “Aplicativo móvil para la detección de enfermedad del mosaico común en la yuca (Manihot Esculenta) utilizando redes neuronales convolucionales” escrito por Diaz Holgado y Villalba (2021) de la Universidad nacional amazónica de Madre de Dios. El sistema consta de una aplicación móvil que, a través de la cámara del celular, pueda identificar las características que posee la hoja en estudio y determinar si estas características, son las que posee una hoja infectada con la enfermedad del mosaico.

1.2.2.2 Antecedentes Nacionales

Las aplicaciones móviles en la actualidad son muy utilizadas gracias a las facilidades de acceso a internet, así como los avances tecnológicos de teléfonos inteligentes, estos cuentan con sistemas operativos que facilitan el desarrollo de

aplicaciones gratuitas que se pueden instalar en un dispositivo móvil sin ningún problema. Al realizar un análisis de los beneficios que ofrece la tecnología, se propuso una aplicación móvil con redes neuronales para los productores cafetaleros APEN la aplicación brindara la detección de enfermedades más comunes en plantas.

En el proyecto se identificó el problema, los antecedentes y la situación actual del problema. Con lo cual se determinan los objetivos tanto generales y específicos para luego obtener las justificaciones, la metodología, las herramientas que utilizaran.

Actualmente los productores de APEN. Tienen dificultad de identificar con exactitud las enfermedades que atacan a las plantaciones de café, a la causa identificada se propone una App móvil incorporado el sistema de inteligencia artificial que usará redes neuronales convolucionales para apoyar a los agricultores en el tratamiento de la planta de café y permitirá tomar mejores decisiones a los agricultores en la detección de enfermedades.

Ante la situación y aprovechando los grandes beneficios que brinda el uso de la tecnología, el proyecto realizado se enfoca en ayudar a que esta situación mejore para los caficultores, un método más eficiente, con ello brindar un servicio óptimo en la agricultura.

Posteriormente se desarrollará la aplicación móvil, paso a paso con el uso de las diferentes herramientas de software y un almacenamiento de base de datos donde se encontrará la información requerida para los cafetaleros.

Finalmente, por lo último se describirá los resultados, de la misma manera las conclusiones y recomendaciones sobre el uso de la aplicación móvil.

- Tesis: “Clasificación mediante redes neuronales y conglomerados no jerárquicos de las condiciones de vida de los hogares de Bolivia “escrito por Rivero(2014) de la Universidad Mayor de San Andrés. Logra distinguir una cara en una calle mal iluminada, leer entre líneas en una conversación, o discernir un susurro en una sala ruidosa; pero lo más impresionante de todo es su capacidad aprender a representar la información necesaria para desarrollar tales habilidades sin instrucciones explícitas que dirijan este proceso.

- Tesis: “Detector de mentiras mediante el análisis de estrés de voz usando redes neuronales recurrentes “escrito por Carrasco (2022) de la Universidad Mayor de San Andrés.El presente trabajo tiene como finalidad realizar un detector de mentiras analizando el estrés de voz con redes neuronales recurrentes para la detección del engaño.

1.2.2.3 Antecedentes Locales

- Tesis: “Biometría facial basada en redes neuronales aplicando el algoritmo de viola-Jones “escrito por Ticona (2020) de la Universidad Pública de el Alto. En la construcción del sistema se aplican redes neuronales, y en específico, la red base radial, en el cual presenta diferentes usabilidades. También se utilizará para su construcción MatLab (laboratorio de matrices) que es una herramienta de software matemátic

1.3 PLANTEAMIENTO DEL PROBLEMA

En el planteamiento del problema se desglosa el problema principal y los problemas secundarios.

1.3.1 Problema Principal

La asociación de productores ecológicos Nogalani (APEN) del municipio la Asunta del departamento de la Paz, los agricultores de café, presentan dificultades en las detecciones de las plagas, mediante el uso de la percepción visual la cual tienen la dificultad al identificar el tipo de plaga y trae como consecuencia gastos excesivos en la producción de café y el uso erróneo de los agroquímicos causado la mortalidad en plantas de café.

1.3.2 Problemas Secundarios

Tras observar el cultivo de plantaciones de café, socios de APEN se pudieron identificar los siguientes problemas:

- **Impacto Económico en las Familias**

Las enfermedades en las plantas de café disminuyen la producción, lo cual afecta significativamente los ingresos de las familias de caficultores. Esta reducción en los ingresos compromete el sustento familiar y limita la capacidad de los agricultores para invertir en mejores prácticas agrícolas.

- **Reducción en el Rendimiento y Mortalidad de las Plantaciones**

Las enfermedades en los cultivos de café no solo disminuyen el rendimiento, sino que también provocan un aumento en la tasa de mortalidad de las plantas.

Esto agrava la situación al reducir las áreas productivas disponibles y obliga a los agricultores a reemplazar continuamente las plantas afectadas.

- **Aplicación Incorrecta de Agroquímicos**

La dificultad de los agricultores para identificar correctamente las plagas lleva a la aplicación de agroquímicos inadecuados. Esta práctica no solo aumenta los costos de producción, sino que también puede causar un daño adicional a las plantas y al medio ambiente, empeorando el problema de salud en el cultivo.

- **Dependencia en métodos tradicionales y limitación en la toma de decisiones**

La ausencia de herramientas precisas para el diagnóstico de plagas obliga a los agricultores a depender de la percepción visual y de su experiencia, métodos que pueden ser ineficaces en situaciones de mayor complejidad. Esto limita su capacidad para tomar decisiones informadas y proactivas para el manejo de las plantaciones

1.3.3 Formulación del Problema

¿De qué manera el aplicativo móvil hará el uso de redes neuronales convolucionales permitirá el apoyo a los caficultores en el tratamiento de las enfermedades en las plantas de café de los productores de APEN?

1.4 OBJETIVOS

1.4.1 Objetivo General

Desarrollar una aplicación móvil para el sistema Android que permita a los productores de café Nogalani (A.P.E.N.) detectar de manera precisa las plagas que afectan sus plantaciones de café, mediante redes neuronales convolucionales a fin de analizar y reconocer las imágenes de las hojas de café.

1.4.2 Objetivos Específicos

- Investigar los tipos de plagas que afectan a las plantaciones de café, productores del Municipio la Asunta en la Asociación de APEN.
- Desarrollar el entrenamiento de imágenes en el modelo de red neuronal convolucional (APEN) en la detección de plagas en la hoja de café.
- Desarrollar la aplicación móvil modelo de red neuronal convolucional (CNN), que permitir a los agricultores a tomar mejores decisiones informadas sobre el manejo de cultivo de café.
- Desarrollar la interfaz de la aplicación móvil con el modelo de red neuronal convolucional (CNN), que no se visualizara la precisión de reconocimiento de plagas.

1.5 JUSTIFICACIONES

1.5.1 Justificación Técnica

El desarrollo será para el sistema Android con la compatibilidad en teléfono móvil aplicando de un sistema de inteligencia artificial que se utilizará redes

neuronales convolucionales. La aplicación tendrá la capacidad de capturar la imagen de las plantas de café, donde estará entrenado para procesar y para posterior indicarnos el resultado

1.5.2 Justificación Económica

Actualmente los caficultores son amenazados por diferentes tipos de enfermedades, por diferentes factores, que incide en la productividad de la planta y la calidad del grano, afectando al productor cafetalero en sus ingresos y calidad de vida. La aplicación móvil que se propone la implementación que estará entrenado para detectar enfermedades de las plantas de café, en el instante tendrás el resultado de la enfermedad que presenta para luego el agricultor que pueda tratarlo la enfermedad en el corto tiempo, relativamente será la mejora de las plantas de café, con un mayor rendimiento en su producción del grano. El beneficio económico será para el agricultor por la mejora en su producción, con un incremento en la cantidad de grano y una mejora en la calidad del producto.

1.5.3 Justificación Social

Mediante la implementación de la aplicación móvil con inteligencia artificial para detectar enfermedades de las plantaciones de café, relativamente tendrá la mejora de las plantas y el producto del café, obteniendo el resultado de grano de oro. Con la calidad del producto que se ofrecerá a la sociedad, de los distintos beneficios y propiedades que tiene el producto de café en el consumo humano.

1.6 METODOLOGIA

1.6.1. Metodología del Desarrollo

La metodología Mobile-D, una propuesta de desarrollo en aplicaciones móviles, se fundamenta en la experiencia de investigaciones previas en aplicaciones móviles, la evaluación del potencial de éxito para servicios de tercera generación denominada 6 M, la ingeniería de software educativo con modelado orientado por objetos (ISE-OO), y principalmente en los valores de las metodologías agiles.

Fases a seguir para el desarrollo del sistema son:

- **Análisis**: En esta fase se analizan las peticiones o requerimientos de las personas o entidad para la cual se desarrolla el servicio móvil “cliente”, el propósito es definir las características del mundo o entorno de la aplicación.
- **Diseño:** El objetivo de esta etapa es plasmar el pensamiento de la solución mediante diagramas o esquemas, considerando la mejor alternativa al integrar aspectos técnicos, funcionales, sociales y económicos.
- **Desarrollo:** El objetivo de esta fase es implementar el diseño en un producto en software.
- **Pruebas de funcionamiento:** El objetivo de esta fase es verificar el funcionamiento de la aplicación en diferentes escenarios y condiciones.

- **Entrega**

Terminada la depuración de la aplicación y atendidos todos los requerimientos de última hora del cliente se da por finalizada la aplicación y se procede a la entrega del ejecutable, el código fuente, la documentación y el manual del sistema.

1.6.2 Métricas de Calidad

Se evaluará la calidad de software usando la norma ISO/IEC 25000 SQuaRE(Software Product Quality Requirements and Evaluation), establece un marco de trabajo para la evaluación y mejora de la calidad de software y servicios de TI. Proporciona una serie de requisitos y recomendaciones que permiten a las organizaciones garantizar la calidad de sus productos y servicios, así como satisfacer las necesidades y expectativas de sus clientes (Akismet, 2024, p. 5).

- Beneficios de la norma ISO 25000
- Mejor calidad
- Mayor satisfacción del cliente
- Competitividad
- Reducción de costo

1.6.3 Estimación de Costos

Para la estimación de costos del software se aplicará el Modelo COCOMO II. Es un modelo de estimación cuantitativa que utiliza un enfoque analítico basado en la descomposición del proyecto y el análisis individual de sus componentes.

Proporciona una estimación aproximada del tiempo, el costo y los recursos necesarios para desarrollar un proyecto de software (Software, 2021, p. 7).

Ofrece tres niveles de detalle para la estimación:

Básico: Ofrece una estimación inicial simple basada en el tamaño del proyecto y algunos factores de ajuste.

Intermedio: Agrega más factores de ajuste y considera aspectos como la experiencia del equipo y la complejidad del proyecto.

Detallado: Proporciona un análisis más profundo, incluyendo una amplia gama de factores que influyen en el costo y esfuerzo del proyecto.

1.6.4 Seguridad de Software

La seguridad del software se centra en identificar y evaluar los posibles riesgos que podrían afectar negativamente al software y, por ende, causar fallos en todo el sistema. Si estos riesgos se pueden detectar tempranamente durante el proceso de ingeniería del software, es posible especificar características de diseño que permitan eliminar o controlar estos riesgos potenciales. Se trata de anticipar y abordar los riesgos de seguridad desde las etapas iniciales del desarrollo del software para garantizar su integridad y confiabilidad.

1.6.5 Pruebas de Software

1.6.5.1 Caja Negra

En los estándares para Software Testing definidos por ISTQB, las técnicas de pruebas de caja negra son utilizadas para realizar pruebas funcionales, basadas

en las funciones o características del sistema y su interacción con otros sistemas o componentes. Se pueden utilizar técnicas basadas en especificación para identificar las condiciones y casos de prueba a partir de la funcionalidad del software (Atom, 2016, p. 12).

Técnicas de Pruebas de Caja Negra:

- Partición de equivalencias
- Análisis de valores borde
- Tablas de decisión
- Transición entre estados
- Pruebas de casos de uso

1.6.5.2 Caja Blanca

La caja blanca es una categoría de las pruebas de software que se refiere a los métodos de comprobación del funcionamiento de la estructura interna y el diseño del software. Contrasta con las pruebas de caja negra, que no se ocupan de las operaciones internas del software, sino que sólo comprueban sus resultados externos, consiste en probar la estructura interna y el diseño de un programa, en contraposición a los resultados externos o la experiencia del usuario final que se prueban en la prueba de caja negra (Thomas, 2024, p. 9).

1.6.4.3 Pruebas de Estrés

Las pruebas de esfuerzo validan el comportamiento de la aplicación bajo condiciones de carga máxima. El objetivo de esta prueba es identificar los errores

como fugas de memoria o problemas de sincronización, que aparecen sólo en condiciones de carga máxima. Las pruebas de esfuerzo le ayudan a encontrar y resolver los cuellos de botella. Los resultados de las pruebas de estrés destacan los componentes que fallan primero y estos resultados pueden ayudar a los desarrolladores a hacer que estos componentes sean más robustos y eficientes. Cada caso diferirá en el volumen del estímulo a aplicar sobre la aplicación (cantidad de usuarios, cantidad de peticiones, etc.), el tiempo que durará cada estímulo y la duración total del experimento, entre otras variables (CANO, 2018, p. 7).

1.6.4.4 Pruebas de Accesibilidad

Las pruebas de accesibilidad se definen como un tipo de prueba de software realizada para garantizar que la aplicación que se está probando sea utilizable por personas con discapacidades como audición, daltonismo, personas mayores y otros grupos desfavorecidos (Hamilton, 2024, p. 2).

1.7 OTRAS METODOLOGÍAS

1.7.1 Método Cualitativo

El proyecto titulado “aplicación móvil con reconocimiento de plagas en plantaciones de café aplicando redes neuronales convolucionales”, se desarrolló la documentación, con la metodología de investigacion cualitativa, porque implica la recopilación y analiza las informaciones, conceptos, opiniones o experiencias, sobre el problema que tienen los productores de café.

1.7.2 Redes Neuronales

En inteligencia artificial la red neuronal es un método que enseña a las computadoras a procesar datos que utiliza los nodos en una estructura de capas que se parece al cerebro humano, las redes neuronales tratan de resolver problemas complicadas como el reconocimiento de rostros.

1.7.3 Redes Neuronales Convolucionales

Las redes neuronales convulsiónales (CNN) son usadas para trabajar con imágenes ya que son un tipo de algoritmo de Deep learning, son capaces de detectar objetos simples en las primeras capas, y con los procesos de entrenamiento poco a poco se van especializando en reconocer formas u objetos más complejos (Llano Carmon, 2021, p. 6).

1.8 HERRAMIENTAS

1.8.1 Lenguaje Para el Desarrollo en Android

En el desarrollo del sistema se utilizarán las siguientes herramientas:

Python:

Python es un lenguaje de programación, se trata de un lenguaje orientado a objetos, fácil de interpretar y con una sintaxis que permite leerlo de manera semejante a como se lee el inglés. Es un lenguaje interpretado, esto significa que el código de programación se convierte en bytecode y luego se ejecuta por el intérprete, en este caso, es la máquina virtual de Python (Duque, 2017, p. 4).

Pycharm

Es un entorno de desarrollo integrado (IDE) desarrollado por JetBrains, una empresa conocida por crear herramientas de desarrollo de alta calidad. Este IDE está diseñado específicamente para programadores que trabajan con el lenguaje de programación Python (Casero, 2024, p. 4)..

Pip

Es el sistema de gestión de paquetes que se utiliza para instalar y administrar las bibliotecas externas en Python, es importante tener en cuenta que Pip no siempre es perfecto.

TensorFlow

Se trata de una librería de código libre para Machine Learning (ML). Fue desarrollado por Google para satisfacer las necesidades a partir de redes neuronales artificiales. TensorFlow te permite construir y entrenar redes neuronales para detectar patrones y razonamientos usados por los humanos (Fidalgo, 2022, p. 11).

Keras

Está escrita en el lenguaje de programación de Python, es una de las API de redes neuronales de alto nivel, su principal motor de back-end es tensorflow, se estructura en dos tipos de modelos los cuales son modelo Sequential y la clase Model.

Open Cv

Open (Open Computer Visión), “es una librería que se usa para el procesamiento de imágenes, que son usadas principalmente para aplicaciones de visión por el computador en tiempo real, de esta manera ayuda a detectar y reconocer objetos” (Mogena Soler, 2014, p.13).

Kotlin

Es un lenguaje de programación usado para el desarrollo de aplicaciones móviles, facilita el manejo de código y la reducción de las líneas de código, la cual permite la inclusión de este lenguaje por la compatibilidad con java la cual permite la reutilización de código (Martines, 2018, p. 8).

Android Studio

Este entorno de desarrollo integrado (IDE) está basado en la herramienta de IntelliJ IDEA, y cuenta con un potente editor de códigos y multitud de funciones que permiten una mayor productividad durante el desarrollo de la aplicación. También ofrece un flexible sistema de compilación, un emulador de gran rapidez y herramientas para identificar problemas de compatibilidad, rendimiento o usabilidad (Caceres, 2024, p. 2).

Java

Es un lenguaje de programación orientada a objetos que cuenta con una gran comunidad de desarrolladores, así como una gran cantidad de librerías que ayudan a cumplir diferentes funciones. Es el lenguaje de programación más utilizados ya que tiene compatibilidad con muchas plataformas.

1.9 LIMITES Y ALCANCES

1.9.1 Limites

- La aplicación móvil se limita a los agricultores cafetaleros.
- La ejecución de la aplicación se limita a dispositivos móviles que cuentes con plataforma Android versión 4.2 para adelante, con cámaras de buena calidad.
- La aplicación se limita a detectar las enfermedades de roya y Phoma o Ácaros en las plantas de café.

1.9.2 Alcances

El presente trabajo de la aplicación podrá facilitar a la organización de caficultores que están asociados a la institución, por lo cual busca incrementar socios que aporten el grano de oro y tengan buen ingreso.

El estudio de este proyecto se realizó en el área rural, puede ser el uso a nivel nacional| e internacional a todas las personas que requieran del aplicativo móvil desarrollado para los productores de caficultores.

1.10 APORTES

Gracias al avance de la tecnología se puede automatizar y mejorar los procesos en la agricultura, la aplicación desarrollará un aporte muy importante sobre la detección de enfermedades más comunes, a los productores dedicados en la agricultura de café como lo que se está aplicando con el proyecto elaborado

en la cooperativa de APEN será un gran beneficio en la producción y un gran desarrollo económicamente.

CAPÍTULO II

MARCO TEÓRICO

CAPÍTULO II MARCO TEÓRICO

2.1 PLANTACIÓN DE CAFÉ

El café (coffea), es un grano que se cosecha de un arbusto llamado cafeto, crece en climas cálidos, siendo una de las bebidas por su consumo más populares a nivel mundial, sin embargo, los cafetos son muy vulnerables a diversas plagas y enfermedades (Coral, 2012, p. 4).

2.1.1 Tipos de Plantaciones de Café

Se clasificará en tres niveles Internacional, nivel Nacional y nivel Local se detallará las variedades de plantas de café.

2.1.1.1 Tipos de Plantas a Nivel Internacional

El cultivo de café es una actividad agrícola de gran importancia económica y social a nivel mundial (Davis, 2015, p. 8).

Tabla 1: *Tipos de Plantas a Nivel Internacional*

NOMBRE	DESCRIPCIÓN	VARIEDAD
Caturra	Es una planta de porte bajo, altura promedio de 1.80 metros, con eje principal grueso y entrenudos cortos	
Catuai	El Catuaí es una variedad de porte bajo, pero un poco más alta que Caturra, con una altura promedio de 2.25 metros, las ramas laterales	

	forman un ángulo cerrado de 45 grados con el tallo principal, con entrenudos cortos.	
Pache comun	Planta de porte bajo, con una altura promedio de 1.80 metros; el crecimiento de las bandolas primarias forma un ángulo de 60 grados con el tallo principal, con buena ramificación	
Pache colis	Es una planta de porte muy bajo, con altura entre 0.80 a 1.25 metros; característica que le permite resistir a los vientos moderados.	
Pacamara	Esta variedad, en condiciones adecuadas y con buen mantenimiento puede llegar a producir 50 quintales pergaminos secos por manzana (71 quintales pergaminos secos por hectárea).	
Catimor	Estas variedades son muy precoces, productivas y exigentes en el manejo agronómico, especialmente en la nutrición. Evidencian una mayor susceptibilidad a la enfermedad ojo de gallo (Mycena citricolor).	
Lempira	Planta de porte bajo, brotes bronce, de alta productividad (50 a 70 quintales pergaminos secos por manzana), con buena adaptabilidad en zonas de 800 a 1,400 metros sobre el nivel del mar (2,600 a 4,600 pies sobre el nivel del mar).	
Catimor	Planta de porte bajo, compacta, semejante a la variedad Caturra, con brotes de color verde y bronce.	

Castillo	Es una planta de porte bajo, ligeramente más alta y con ramas más largas que la variedad Caturra, vigorosa y de alta productividad.	

2.1.1.2 Tipos de Plantas a Nivel Nacional

Diferentes variedades de café producidas en Bolivia que potencian la oferta de café del país. Los bolivianos son apreciados por su sabor limpio, frutal y dulce calidad aromática (Mamani, 2018, p. 5).

Tabla 2:*Tipos de Plantas a Nivel Nacional*

NOMBRE	DESCRIPCIÓN	VARIEDAD
Borbón de frutos rojos	Es de tamaño medio, con una estructura frondosa y ramas delgadas que pueden necesitar soporte. Su crecimiento es vertical y las hojas son de un verde brillante, más grandes que las de otras variedades	
Borbón de frutos amarillos	Produce cerezas de color amarillo en lugar de las rojas	
Caturra	Es una planta de café de tamaño medio y crece en altitudes entre los 800 y 1,500 metros sobre el nivel del mar.	

Catuai	Esta variedad es muy apreciada por su resistencia a enfermedades y su capacidad de adaptación a diferentes condiciones climáticas y geográficas	
Icatu	Esta variedad es popular en regiones productoras de café debido a su adaptabilidad, resistencia y capacidad de obtener buenos rendimientos, además de ser relativamente fácil de cultivar y cosecha	
Java	Esta variedad es muy apreciada por su perfil de sabor único y sus características de cultivo.	
Castillo	Es una variedad muy resistente a las enfermedad de roya que afecta a las hojas de las plantas de café, que aparecen en forma de manchas de color amarillo-naranja.	

2.1.1.3 Tipos de Plantas de Café en la Comunidad de Nogalani

Las variedades de planta de café más cultivadas en el Municipio de Asunta comunidad Nogalani son:

Tabla 3: *Tipos de Plantas de Café en la Comunidad de Nogalani*

NOMBRE	DESCRIPCIÓN	VARIEDAD
Caturra	Es una variedad de café más famosas del mundo y una de las más cultivadas en nuestro país.	
Catuai rojo	Esta variedad es muy apreciada por su resistencia a enfermedades y su capacidad de adaptación.	

2.1.2 Plagas de la Planta de Café

En general, los insectos debilitan los granos de café y reducen la densidad. Además, las picaduras de estos pueden facilitar una infección secundaria de los cafetos por parte de hongos y otros microorganismos. La infestación de insectos no solo disminuye el rendimiento, sino que también puede tener un efecto importante en el perfil del café, con la reducción de la calidad en cuanto al sabor y el aroma (Contreras, 2020, p. 99).

Tabla 4: *Plagas de la Planta de Café*

PLAGAS	SÍNTOMAS	DAÑOS
Broca	Perforación del fruto por el área de la corona, construyendo un túnel hacia el interior para depositar sus huevos y alimentarse.	

Minador de hoja	Manchas con formas irregulares formadas por las larvas que penetran la hoja para alimentarse de sus tejidos.	
Nematodos	Nudosidades necrosadas en las raíces laterales del cafeto.	
Planococus Cochinilla del café	Caída prematura de las hojas, deformación.	
Oligonychus Acaro rojo del cafeto	Cuando introducen el estilete en la epidemia del haz de la hoja y destruyen las células de las cuales se alimentan, succionando contenido celular de las hojas del café.	

Meloidogyne Nematodo agallador del cafeto	Retraso en el crecimiento, plantas con amarillamiento, entrenudos cortos y defoliación.	

2.1.3 Metodología Para el Uso de Agroquímicos

Para usar agroquímicos de manera segura y eficiente, se deben seguir algunas recomendaciones:

- **Minador de Hoja**

Tabla 5: *Agroquímicos para la Plaga de Minador de Hoja*

PRODUCTO	DOSIS
Oxicloruro de Cobre (50%) a 3.5 Kg /ha	6 g /litro de agua
Óxidos de cobre (50%) o hidrocidos de cobre (50 %) a 2.5 Kg /ha	6 g/litro de agua

Nota: Plaga de Minador de hoja, Almengor, 2012,Manejo integrado de plagas en el cultivo de café.

- **Planococus**

Tabla 6*: Agroquímicos para la plaga de Planococus de Hoja*

Producto	Dosis
Captan	1g/litro de agua
Benlate (50%)	1g/litro de agua

Nota: Plaga de Planococus, Almengor, 2012,Manejo integrado de plagas en el cultivo de café.

- **Nematodos**

Tabla 7: *Agroquímicos para la plaga de Nematodos de Hoja*

PRODUCTO	DOSIS
Oxicloruro de cobre 58.8%	1g/litro de agua

Nota: Plaga de Nematodos, Almengor, 2012,Manejo integrado de plagas en el cultivo de café.

- **Oligonychus**

Tabla 8: *Agroquímicos para la plaga de Oligonychus de Hoja*

PRODUCTO	DOSIS
Bangot (50%)	1g/litro de agua

Nota: Plaga de oligonychus, Almengo, 2012, Manejo integrado de plagas en el cultivo de café.

2.1.4 Control Biológico

Tabla 9: *Control Biológica*

Plagas	Control Biológico
Minador de hojas	o Evitar el uso de abonos nitrogenados o Crianza y liberación de parasitoides
Planococus **Cochinilla del Café**	o Usar fertilizantes adecuados y verificar un mejor control del suelo en el pH.
Nematodos	o Evitar sembrar café donde se sembró anteriormente o Favorecer la aireación o Las plantas de sombra no deben tener más de 20 años. o Aplicaciones de oxicloruro de cobre.
Oligonychus **Arañita roja**	o Se realiza a través de la acción de ácaros depredadores de la familia Phytoseiidae además de otros enemigos naturales como los Coccinellidae

2.1.5 Aplicación Móvil

Es una herramienta tecnológica que emplea inteligencia artificial (IA) para identificar y diagnosticar de manera rápida y precisa las enfermedades y plagas que afectan a las plantaciones de café, todo a través de la cámara de un teléfono inteligente

2.1.6 Aplicación de Visión Artificial

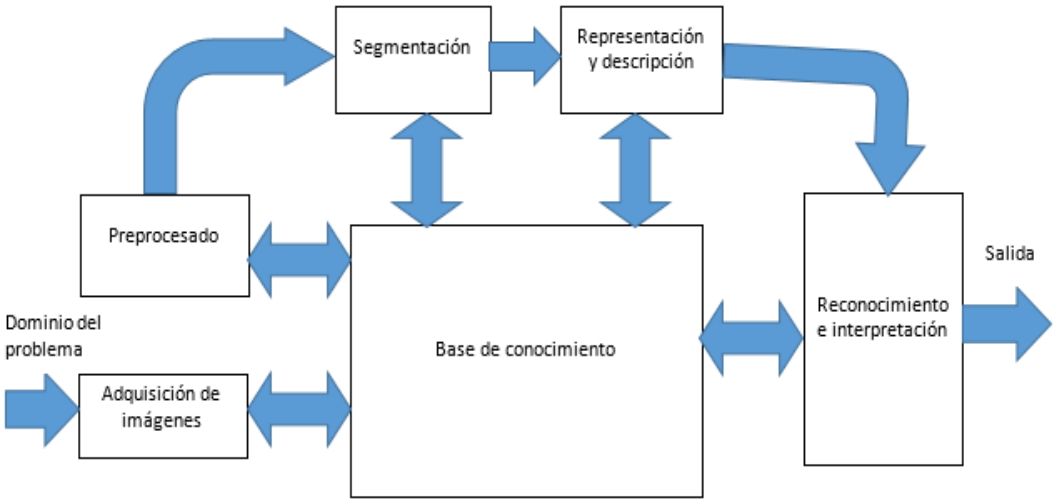
Según (Sullcahuaman, 2021) afirma que la visión artificial es la utilización de las técnicas adecuadas de las cuales permiten el análisis y el procesamiento de cualquier tipo de información en el campo de la inteligencia artificial a través de imágenes digitales. Los procesos que utiliza la visión artificial para realizar los análisis de las imágenes son los siguientes procesos: adquisición de la imagen, pre procesamiento, segmentación, representación y descripción, reconocimiento e interpretación.

2.1.7 Procesamiento de Imágenes Digitales

Según (Parra-Plaza, 2015) define, “el procesamiento de imágenes digitales como la representación de diferentes estrategias y métodos que se utilizan para analizar, modificar y/o extraer cualquier tipo de información”.

Para (Vilet, 2005) afirma que “el procesamiento de imágenes digitales representa un papel muy importante porque está construido sobre bases probabilísticas y matemáticas ya que está basado para desarrollar juicios visuales subjetivos para tener un mejor entendimiento con la percepción humana”.

Figura 2: *Procesamiento de Imágenes Digitales*



Nota: Procesamiento de imágenes, Parra-Plaza, 2015, Entrenamiento de CNN.

2.1.7.1 Adquisición de la imagen de la Planta de Café

En esta etapa se captura la muestra mediante algún dispositivo o algún sensor y en la cual la calidad de las imágenes tomadas depende de la precisión de la digitalización de este dispositivo (Parra-Plaza, 2015, p. 3).

2.1.7.2 Pre procesamiento

"En esta etapa posterior a la adquisición de la imagen se trata de mejorar la calidad de la imagen utilizando técnicas como la mejora en el contraste de imagen" (Parra-Plaza, 2015, p. 15).

2.1.7.3 Segmentación Semántica

"Diríamos que esta etapa es la más importante porque en este proceso se aplican los algoritmos que se va a utilizar para separar la imagen en sus componentes, buscando así una mejora en la búsqueda de resultados" (Parra-Plaza, 2015, p. 15).

2.1.7.4 Representación y Descripción

“En esta etapa se permite obtener la información de interés y adecuada para la identificación de patrones y rasgos, por ejemplo, las manchas de las hojas” (Parra-Plaza, 2015, p. 15).

2.1.7.5 Reconocimiento e Interpretación

“Finalmente, en la última etapa se basa principalmente en la interpretación de resultados de las fases anteriores comparándolos con referentes en base de datos, buscando similitudes en correspondencias basadas en patrones” (Parra-Plaza, 2015, p. 16).

2.1.8 Procesamiento de Imagen OpenCv

OpenCv (Open Computer Visión), “es una librería que se usa para el procesamiento de imágenes, que son usadas principalmente para aplicaciones de visión por el computador en tiempo real, de esta manera ayuda a detectar y reconocer objetos” (Mogena Soler, 2014, p.9)

Esta librería cuenta con un amplio conjunto de algoritmos de visión por computador y aprendizaje automático con un más de 2500 algoritmos optimizados, entre las principales funcionalidades de opencv tenemos las siguientes:

- Procesamiento de imágenes (Brillo, Contraste, Umbral)
- Detección de objetos
- Captura de imágenes en tiempo real.

Entre los principales lenguajes de programación es compatible con C++, C, Python, Java, y los sistemas operativos de Linux, Windows, Android y mac0s.

2.1.9 Machine Learning

Machine learning o aprendizaje automático, es un proceso computacional que para lograr una tarea deseada se debe utilizar datos de entrada sin estar codificado, logrando un resultado concreto. Estos algoritmos se vuelven mejores mediante la repetición para lograr una tarea deseada adaptándose a su arquitectura. (El Naqa & J.Murphy, 2015, p. 4).

El objetivo de este algoritmo es optimizar los datos de entrada para producir un resultado deseado partiendo de datos no vistos, el entrenamiento es la parte fundamental del aprendizaje automático en la cual es un buen algoritmo que a medida que entrena aprende de sus errores mientras procesa los datos.

2.1.10 Visión general de aprendizaje automático

- El aprendizaje automático: puede dividirse por su naturaleza de etiquetado de las siguientes formas:
- Aprendizaje supervisado: Es usado para mapear datos de entrada a partir de muestras desconocidas donde los datos de salida son etiquetados.
- Aprendizaje no supervisado: Es usado para la entrada de datos al sistema de aprendizaje.

- Aprendizaje semi-supervisado: está formado por una parte de datos que parcialmente está etiquetada se utiliza para inferir datos no etiquetados, es la unión entre un aprendizaje supervisado y uno no supervisado.

2.2. METODOLOGIA MOVIL - D

La metodología Mobile-D, una propuesta de desarrollo en aplicaciones móviles, se fundamenta en la experiencia de investigaciones previas en aplicaciones móviles, la evaluación del potencial de éxito para servicios de tercera generación denominada 6 M, la ingeniería de software educativo con modelado orientado por objetos (ISE-OO), y principalmente en los valores de las metodologías agiles (Medina, 2017, p. 11).

Fases a seguir para el desarrollo del sistema son:

- **Análisis**

En esta fase se analizan las peticiones o requerimientos de las personas o entidad para la cual se desarrolla el servicio móvil “cliente”, el propósito es definir las características del mundo o entorno de la aplicación.

- **Diseño**

- El objetivo de esta etapa es plasmar el pensamiento de la solución mediante diagramas o esquemas, considerando la mejor alternativa al integrar aspectos técnicos, funcionales, sociales y económicos.

- **Desarrollo**

El objetivo de esta fase es implementar el diseño en un producto en software.

Pruebas de funcionamiento

El objetivo de esta fase es verificar el funcionamiento de la aplicación en diferentes escenarios y condiciones.

Entrega

Terminada la depuración de la aplicación y atendidos todos los requerimientos de última hora del cliente se da por finalizada la aplicación y se procede a la entrega del ejecutable, el código fuente, la documentación y el manual del sistema.

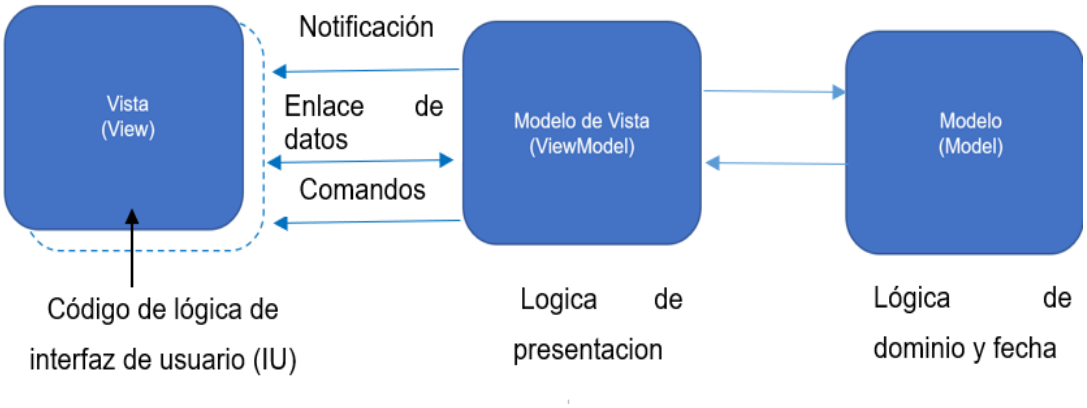
2.3 MODELO DE REQUERIMIENTO

2.3.1. Modelo Vista - Vista Modelo (Model View-View Model)

MVVM es un patrón de diseño arquitectónico que facilita la separación entre la interfaz gráfica de usuario (front-end) y la lógica de negocio (back-end) en cuanto a su desarrollo se refiere. Es decir, que permite desarrollarlos de forma independiente.

Para hacer esto posible, MVVM ha de contar con los siguientes componentes:

Figura 3: *Arquitectura de MVVM*



Nota: Arquitectura de Modelo vista – Vista Modelo, Alberto Cardona López, 2019,Desarrollo de Software.

Donde, además de los componentes habituales (views y models), encontramos:

El ViewModel es una interfaz o intermediario entre el back-end y el front-end que se encarga de preparar los datos que recibe del back-end (transformándolos o disponiendo de ellos en distintas formas) de forma que facilite a la capa visual los datos consumir y presentar los datos de forma cómoda. Se puede entender como que un view model es, en efecto, un modelo para una vista. Es práctica extendida el asociar cada vista con un único view model.

Notifications y Data Binding se refieren a las notificaciones que se envían entre el ViewModel y la vista cuando alguno de los dos ha actualizado algún objeto que esté enlazado (que tenga un binding entre el ViewModel y la vista).

Commands como forma de encapsular las acciones disponibles para el usuario. Basta con hacer referencia a éstos en la vista e implementarlos después

en el ViewModel. Incluso si no se implementan en absoluto, no tendrá lugar ningún error, si bien esas acciones no surtirán efecto alguno.

2.3.1.1 Ventaja del Patrón MVVM

Un ViewModel proporciona un único depósito de estados y políticas de presentación, lo que mejora la reutilización del Modelo (desacoplándolo de las Vistas) y facilita el reemplazamiento de las Vistas (al eliminar políticas específicas de presentación, de las Vistas)

Un diseño MVVM mejora la facilidad de realizar Testing (Pruebas Unitarias especialmente) de la aplicación. Al separar la lógica de las vistas y controles visuales, podemos crear fácilmente pruebas unitarias que se encarguen exclusivamente del Modelo y del ViewModel (Puesto que las Vistas serán normalmente solo XAML, sin "code-behind"). Adicionalmente, MVVM también facilita la implementación de MOCKs en la Capa de Presentación, porque habiendo desacoplado las Vistas del Modelo y situando la lógica cliente en los ViewModels, esa lógica es fácilmente sustituible por MOCKS (simulación de ejecución), que es fundamental para las pruebas de aplicaciones complejas.

El patrón MVVM ofrece un diseño desacoplado. Las Vistas solo tienen referencia al View Model y el View Model referencia solo al Modelo. El resto lo realiza el databinding y Commands de la infraestructura.

2.3.1.2 Desventajas del Patrón MVVM

La relación típica entre un View Model y sus correspondientes Vistas son normalmente “una a muchas”, pero hay situaciones en las que eso no es cierto. En

general, cualquier lógica de negocio cliente y lógica de negocio de validación de entrada de datos (seguimiento de selección de elementos, etc.) debe implementarse en el View Model.

Hay situaciones donde un View Model es “consciente” de otro View Model dentro de la misma aplicación. Esas situaciones aparecen cuando hay una relación maestro-detalle entre dos View Models o cuando un View Model representa a un elemento suelto (por ejemplo, la representación visual de un único Cliente).

2.4. ESTIMACION DE COSTOS

2.4.1. Cocomo II

Para la estimación de costos del software se aplicará el Modelo COCOMO II.

Es un modelo de estimación cuantitativa que utiliza un enfoque analítico basado en la descomposición del proyecto y el análisis individual de sus componentes. Proporciona una estimación aproximada del tiempo, el costo y los recursos necesarios para desarrollar un proyecto de software (Software, 2021, p. 7).

Ofrece tres niveles de detalle para la estimación:

Básico: Ofrece una estimación inicial simple basada en el tamaño del proyecto y algunos factores de ajuste.

Intermedio: Agrega más factores de ajuste y considera aspectos como la experiencia del equipo y la complejidad del proyecto.

Detallado: Proporciona un análisis más profundo, incluyendo una amplia gama de factores que influyen en el costo y esfuerzo del proyecto.

2.4.1.1 Ecuación del Modelo de Cocomo II

Las ecuaciones que se utilizan son:

(1) Cantidad de Líneas de Código en Miles KLDC

$$KLDC = \text{LDC}/1000 \quad (1)$$

Donde:

$KLDC =$ Cantidad de líneas de código en miles

$LDC =$ Líneas de código

(2) Esfuerzo en Personas - Mes (ED)

$$\text{ED} = \text{A} * (\text{KLDC})^{B} \quad (2)$$

Donde:

$ED =$ Esfuerzo requerido por el proyecto en personas-mes

$KLDC =$ Cantidad de líneas de código en miles

$A =$ Coeficiente Cocomo II

$B =$ Coeficiente Cocomo II

(3)Tiempo de Desarrollo en Mes (TD)

$$TD = C * (ED)^{D} \quad (3)$$

Donde:

$TD =$ Tiempo de desarrollo

$ED =$ Esfuerzo requerido por el proyecto.

C y D = Constantes con valores definidos en una tabla.

(4) Numero de Programadores Necesarios

$$NP = \frac{ED}{TD} \quad (4)$$

Donde:

$NP =$ Numero de Programadores

$ED =$ Esfuerzo requerido por el proyecto en personas-mes

$TD =$ Tiempo de desarrollo

(5) Costo del Proyecto

$$CS = NP * SM * TD \quad (5)$$

Donde:

$CS =$ Costo del software

$SM =$ Sueldo mes

$NP =$ Numero de programadores

$TD =$ Tiempo de desarrollo

2.5.1.2 Modelo de Cocomo II

Se utiliza para obtener una primera aproximación rápida del esfuerzo, y hace uso de la siguiente tabla de constantes para calcular distintos aspectos de costos

Tabla 10: *Modelo Básico*

PROYECTO DE SOFTWARE	A	B	C	D
Orgánico	2,4	1,05	2,5	0,38
Semicopado	3,0	1,12	2,5	0,35
Empotrado	3,6	1,20	2,5	0,32

Se puede observar que a medida que aumenta la complejidad del proyecto (modo), las constantes aumentan de 2.4 a 3.6, que corresponde a un incremento del esfuerzo del personal. Hay que utilizar con mucho cuidado el modelo básico puesto que se obvian muchas características del entorno

2.4.1.3 Ventajas de COCOMO II

Este modelo se usa en las etapas tempranas de un proyecto de software, cuando se conoce muy poco del tamaño del producto a ser desarrollado, de la naturaleza de la plataforma, del personal a ser incorporado al proyecto o detalles específicos del proceso a utilizar. Este modelo podría emplearse tanto en productos desarrollados en sectores de Generadores de Aplicación, Sistemas Integrados o Infraestructura.

Las ventajas del modelo de estimación de costos COCOMO II son:

- Es fácil de realizar y de interpretar
- Tiene pocas variables
- Se acerca a la realidad en la mayoría de los casos

2.5.1.4 Desventajas del Modelo Cocomo II

- No saca resultados fiables en proyectos demasiado pequeños
- La elección de las variables es muy subjetiva depende de la persona que realiza el estudio.(Alban &Galarza, 2017, p. 10)

2.5 MÉTRICAS DE CALIDAD

Son todas las métricas de software que definen de una u otra forma la calidad del software. Tales como exactitud, estructuración o modularidad, pruebas, mantenimiento, reusabilidad, cohesión del módulo, acoplamiento del módulo, etc. Estas son los puntos críticos en el diseño, codificación, pruebas y mantenimiento. (Garcia & Palafox, 2010, p.8)

Para el presente proyecto tomaremos en cuenta la métrica de calidad ISO/IEC 25010 hace parte de la familia ISO 25000. Es una norma que está centrada hacia la usabilidad, en el cual se determinan las características de calidad que se deben tener en cuenta en el momento de evaluar las propiedades de un producto.

En términos matemáticos se define así:

6) Predicción Positiva

$$P = \frac{VP}{VP + FP} \qquad (6)$$

Donde:

$\boldsymbol{P}$ = Exactitud de las predicciones positivas.

$\boldsymbol{VP}$ =Verdadero positivo.

$\boldsymbol{FP}$ = Falsos positivos.

2.5.1 Funcionalidad

Representa la capacidad del producto software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas de los usuarios cuando el producto se usa en las condiciones especificadas. Esta característica se subdivide a su vez en las siguientes sub características:

Completitud funcional. Grado en el que el conjunto de funcionalidades del producto cubre todas las tareas y los objetivos de usuario especificados.

Corrección funcional. Capacidad del producto o sistema para proveer resultados exactos cuando es usado por los usuarios especificados.

Pertinencia funcional. Capacidad del producto software para proporcionar un conjunto de funciones que facilitan la consecución de tareas y objetivos de usuario especificados.

Fórmula para ajustar puntos de función, que considera factores como la complejidad técnica, el rendimiento y la usabilidad.

(7) Formula de Ajustes

$$PF = conteo\ total\ [0.65 + 0.01 * \sum fi] \quad (7)$$

Donde:

$\sum(fi) =$ Sumatoria de los valores de los factores de ajustes

$PF =$ Ajustes

Con el límite superior establecido en la suma de los factores de ajuste de acuerdo a los parámetros propuestos, se realiza el cálculo del valor siguiente en la secuencia, este proceso es fundamental para mantener la coherencia en la evaluación de la complejidad del sistema.

(8) Punto Función Máxima

$$PFmax = Cuenta\ Total[0.65 + (0.01 * \sum fi)] \quad (8)$$

Donde:

$\sum(fi)$ = Sumatoria de los valores de los factores de ajuste.

$PFmax$ = Punto función máxima.

Con estos factores podemos encontrar el valor de la funcionalidad.

(9) Formula de la Funcionalidad

$$F = PF/(PF\ max) * 100 \quad (9)$$

Donde:

F = Funcionalidad

PF = Pruebas función

PF = Punto máximo

2.5.2 Confiabilidad

La confiabilidad de un sistema se refiere a su capacidad para mantener un nivel de funcionamiento adecuado durante un período de prueba establecido que implica considerar aspectos como la madurez del sistema. Aplicaremos las fórmulas correspondientes, teniendo en cuenta las probabilidades de fallas y éxitos debido a varios factores.

- Utilización de recursos. Grado en que la cantidad y tipos de recursos utilizados por el producto al llevar a cabo su función bajo condiciones determinadas no exceden lo especificado.

- Capacidad. Grado en que el producto cumple los requisitos relativos a límites máximos para un parámetro (ítems almacenados, usuarios concurrentes, ancho error de banda de comunicaciones).

  Considerando que se tiene en cuenta que:

  (10) Probabilidad de Fallas

$$P(T =< t) \quad => PF \qquad (10)$$

Donde:

$PF = Probabilidad\ de\ fallas$

(11) Probabilidad de Trabajo Sin Fallas

$$P(T > t) = 1 - F(t) => PT \qquad (11)$$

Donde:

$PT =$ Probabilidad de trabajo sin fallas

Se considera la siguiente función al calcular la confiabilidad del sistema.

12)Función de la Confiabilidad del Sistema MT

$$F(t) = f * e^{-\mu * t} \qquad (12)$$

Donde:

$f =$ Funcionalidad del sistema

$\mu =$ Probabilidad de error

$t =$ Tiempo de prueba del sistema

$F(t) =$ Confiabilidad

2.5.3 Usabilidad

Al evaluar la usabilidad de un sistema, es importante considerar el factor humano, asegurando que el software cumpla con los requisitos y expectativas del usuario, por lo tanto, se lleva a cabo una evaluación mediante encuestas dirigidas a los usuarios del sistema.

Estas encuestas permiten recopilar información valiosa sobre la experiencia del usuario, su satisfacción y la facilidad de uso percibida que se utiliza luego para calcular la usabilidad del sistema, aplicando una fórmula diseñada específicamente para medir la eficacia y la experiencia del usuario en la interacción con el software.

(13) Función para Determinar la Usabilidad MT

$$FU = \frac{\left(\frac{\sum valor}{n} * 100\right)}{5} \qquad (13)$$

Donde:

$FU =$ Función usabilidad

$\sum \text{valor} =$ Sumatoria de los valores de la usabilidad del sistema

$n =$ Cantidad de valores de la usabilidad del sistema

2.5.4 Eficiencia

Capacidad del producto software para que el usuario interactúe mediante su interfaz intercambiando información para completar determinadas tareas.

Para determinar la eficiencia, se emplea la fórmula siguiente.

(14) Fórmula de la Eficiencia MT

$$E = \frac{\sum \boldsymbol{x_i}}{\boldsymbol{n}} * \frac{100}{\boldsymbol{n}} \qquad (14)$$

Donde:

$\sum x_i$ = Sumatoria de los valores de eficiencia

n = Numero de preguntas

E = Eficiencia

2.5.5 Mantenibilidad

Esta métrica evalúa la cantidad de trabajo requerido para realizar cambios en el sistema, ya sea para corregir errores o agregar nuevas funcionalidades. El estándar IEEE94 recomienda utilizar el índice de madurez del sistema como indicador de su estabilidad. Por lo tanto, la ecuación para este índice es la siguiente:

15) Cálculo del Nivel de Madurez del Sistema

$$IMS = \frac{[Mt - (Fa + Fc + Fd)]}{Mt} \qquad (15)$$

Donde:

Mt = Numero de modulos total nde la versión actual

Fa = Numero de módulos de la versión actual que se añadieron

Fc = Numero de modulo de la versión actual que se cambiaron

Fd = Numero de modulo de la versión anterior que se eliminaron en la versión actual

IMS = Nivel de madurez del sistema

2.5.6 Portabilidad

La capacidad de un software para ser transferido de un entorno a otro incluye varios aspectos importantes:

- **Adaptabilidad**

Habilidad del software para ajustarse a diferentes entornos sin requerir modificaciones adicionales.

- **Facilidad de Instalación**

Esfuerzo requerido para instalar el software en un entorno específico.

- **Conformidad**

Verificación de que el software cumple con estándares o convenciones de portabilidad.

- **Capacidad de Sustitución**

Facilidad y esfuerzo necesarios para sustituir el software con otro producto que tenga funciones similares.

El sistema, desarrollado con Laravel, puede implementarse en cualquier servidor con Apache y las herramientas necesarias.

Como la tecnología web es ejecuta fácilmente en cualquier dispositivo con conexión a Internet y un navegador.

(16)Fórmula de la Portabilidad MT

$$P = 1 - (\frac{NDA}{NDI}) \qquad (16)$$

Donde:

P = Probabilidad

NDA = Numero de dia para aportar el sistema

NDI = Numero de dias para implementar el sistema

2.6 PRUEBAS DE SOFTWARE

Las pruebas de software son el proceso de evaluar y verificar que un producto o aplicación de software hace lo que se supone que debe hacer. Entre los beneficios de unas buenas pruebas se incluyen la prevención de errores y la mejora del rendimiento.

2.6.1 Caja Blanca

La caja blanca es una categoría de las pruebas de software que se refiere a los métodos de comprobación del funcionamiento de la estructura interna y el diseño del software. Contrasta con las pruebas de caja negra, que no se ocupan de las operaciones internas del software, sino que sólo comprueban sus resultados externos, consiste en probar la estructura interna y el diseño de un programa, en contraposición a los resultados externos o la experiencia del usuario final que se prueban en la prueba de caja negra (Thomas, 2024, p. 10).

2.6.2 Pruebas de Caja Negra

Las pruebas de caja negra, una forma de prueba que se realiza sin conocimiento de los componentes internos de un sistema, se pueden realizar para evaluar la funcionalidad, la seguridad, el rendimiento y otros aspectos de una aplicación. Análisis dinámico de código es un ejemplo de pruebas automatizadas de seguridad de caja negra. Los evaluadores de caja negra definen casos de prueba e interactúan con el software como lo haría un usuario para validar que hace lo que debería, como debería (Zaptest, 2024, p. 7).

2.63 Pruebas de Estrés

Las pruebas de esfuerzo validan el comportamiento de la aplicación bajo condiciones de carga máxima. El objetivo de esta prueba es identificar los errores como fugas de memoria o problemas de sincronización, que aparecen sólo en condiciones de carga máxima. Las pruebas de esfuerzo le ayudan a encontrar y resolver los cuellos de botella. Los resultados de las pruebas de estrés destacan los componentes que fallan primero y estos resultados pueden ayudar a los desarrolladores a hacer que estos componentes sean más robustos y eficientes. Cada caso diferirá en el volumen del estímulo a aplicar sobre la aplicación (cantidad de usuarios, cantidad de peticiones, etc.), el tiempo que durará cada estímulo y la duración total del experimento, entre otras variables (CANO, 2018, p. 11).

2.6.4 Pruebas de Accesibilidad en Dispositivo Móvil

Las pruebas de accesibilidad te permiten experimentar tu app desde la perspectiva del usuario y encontrar problemas de usabilidad que podrías pasar por alto. Las pruebas de accesibilidad pueden revelar oportunidades para que tu app sea más potente y versátil para todos los usuarios, incluidos los que tienen discapacidades.

Para obtener los mejores resultados, usa todos los enfoques descritos en este documento:

2.7 HERRAMIENTAS A DESARROLLAR

2.7.1 Lenguaje Para el Desarrollo

En el desarrollo del sistema se utilizarán las siguientes herramientas:

Python: Python es un lenguaje de programación, se trata de un lenguaje orientado a objetos, fácil de interpretar y con una sintaxis que permite leerlo de manera semejante a como se lee el inglés. Es un lenguaje interpretado, esto significa que el código de programación se convierte en bytecode y luego se ejecuta por el intérprete, en este caso, es la máquina virtual de Python (Duque, 2017, p. 5).

Pycharm: Es un entorno de desarrollo integrado (IDE) desarrollado por JetBrains, una empresa conocida por crear herramientas de desarrollo de alta calidad. Este IDE está diseñado específicamente para programadores que trabajan con el lenguaje de programación Python (Casero, 2024, p. 3).

Interpretes: Es un programa que analiza y ejecuta simultáneamente un programa escrito en un lenguaje fuente.

Pip: Es el sistema de gestión de paquetes que se utiliza para instalar y administrar las bibliotecas externas en Python, es importante tener en cuenta que Pip no siempre es perfecto.

Keras: Es una librería que funciona como una interfaz de alto nivel para Tensorflow, CNTK o Theano, y que al momento de programar reduce significativamente la cantidad de código requerida para implementar un modelo. Esto hace más rápido el proceso de desarrollo de diferentes modelos Deep Learning.

Lo anterior quiere decir que al momento de programar usaremos la sintaxis definida por Keras, pero al momento de ejecutar el código Keras se encarga de hacer la traducción a la sintaxis usada por Tensorflow o Pytorch.

TensorFlow Lite: TensorFlow Lite es una biblioteca multiplataforma de aprendizaje automático optimizada para ejecutar modelos de aprendizaje automático en dispositivos perimetrales, incluidos dispositivos móviles iOS y Android.

TensorFlow Lite es el motor principal del Kit de AA para ejecutar modelos de aprendizaje automático. El ecosistema de TensorFlow Lite cuenta con dos componentes que facilitan el entrenamiento y la implementación de modelos de aprendizaje automático en dispositivos móviles:

Model Maker es una biblioteca de Python que facilita el entrenamiento de modelos de TensorFlow Lite con tus propios datos y con solo unas pocas líneas de código. No necesitas experiencia en aprendizaje automático.

La Biblioteca *de tareas* es una biblioteca multiplataforma que facilita la implementación de modelos de TensorFlow Lite con solo unas pocas líneas de código en tus apps para dispositivos móviles.

Open Cv: Open (Open Computer Visión), “es una librería que se usa para el procesamiento de imágenes, que son usadas principalmente para aplicaciones de visión por el computador en tiempo real, de esta manera ayuda a detectar y reconocer objetos” (Mogena Soler, 2014, p. 8).

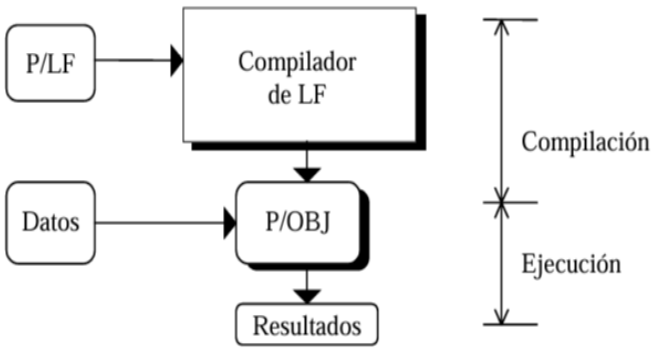
Kotlin: Es un lenguaje de programación usado para el desarrollo de aplicaciones móviles, facilita el manejo de código y la reducción de las líneas de código, la cual permite la inclusión de este lenguaje por la compatibilidad con java la cual permite la reutilización de código (Martines, 2018, p. 5).

Android Studio: Este entorno de desarrollo integrado (IDE) está basado en la herramienta de IntelliJ IDEA, y cuenta con un potente editor de códigos y multitud de funciones que permiten una mayor productividad durante el desarrollo de la aplicación. También ofrece un flexible sistema de compilación, un emulador de gran rapidez y herramientas para identificar problemas de compatibilidad, rendimiento o usabilidad (Caceres, 2024, p. 7).

Compiladores: Es un compilador es un programa que traduce código escrito en un lenguaje de programación (llamado fuente) a otro lenguaje (conocido como objeto).

Los compiladores, a diferencia de los intérpretes, transforman el programa a un programa equivalente en un código objeto (fase de compilación), y en un segundo paso generan los resultados a partir de los datos de entrada (fase de ejecución).

Figura 4: *Esquema General de un Compilador*



Nota: Jose Emilio Labra Guayo, 2017, Proceso del Compilador.

Java Development Kit (JDK):

Es el intérprete de Java. Ejecuta el bytecode a partir de los archivos class.

2.8.1.1 Estructura y Componentes de Keras

Keraz utiliza la misma estructura de componentes que TensorFlow y la amplia. Algunos de los componentes más importantes de Keras incluyen:

- **Capas:** Las capas el bloque de construcción fundamental de Keras. Se utilizan para construir modelos de aprendizaje automático.
- **Modelos:** Los modelos son un conjunto de capas que se utilizan para construir modelos de aprendizaje automático.
- **Optimizadores:** Se utilizan para ajustar los parámetros de los modelos de aprendizaje automático durante el entrenamiento.
- **Callbacks:** Son funciones que se ejecutan durante el entrenamiento de un modelo, permitiendo la customización del proceso de entrenamiento.
- **Métricas:** Son funciones que utilizan para evaluar el rendimiento de un modelo de aprendizaje automático

2.8.1.2 Estructura y Componentes de TensorFlow

TensorFlow tiene sus funcionalidades gracias al uso de diferentes componentes claves de TensorFlow que incluyen:

- **Capas Neuronales:** Las capas son los bloques de construcción fundamentales de los modelos de aprendizaje automático en TensorFlow. Hay diferentes tipos de capas para las diferentes tareas.

Modelos: Los modelos en TensorFlow son la presentación matemática de un problema de aprendizaje automático.

- **Procesamiento de Datos:** Es un paso crucial en el desarrollo de modelos de aprendizaje automático que proporciona herramientas para la limpieza, transformación y normalización de los datos.
- **Algoritmo de Entrenamiento:** Proporciona una variedad de algoritmos de entrenamiento, como el algoritmo de gradiente descent Adam, RMSProp, entre otros, que utilizan para ajustar los parámetros del modelo durante el entrenamiento.

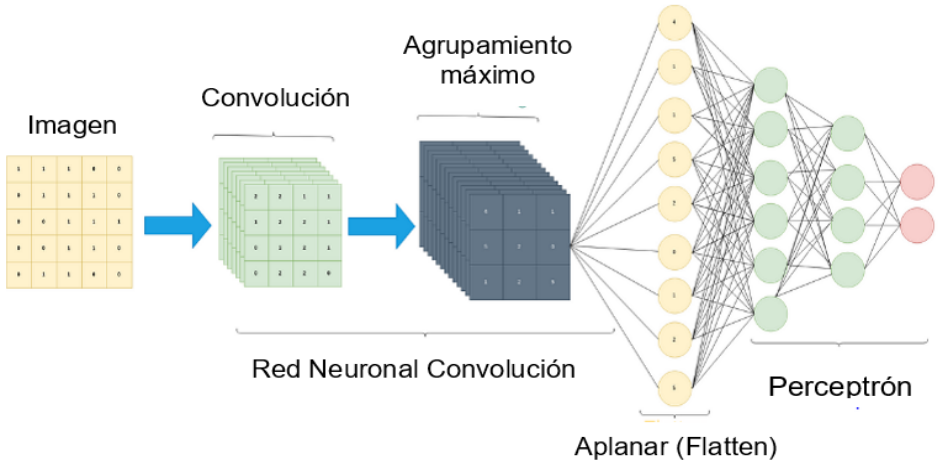
Este codelab se enfoca en TFLite. Los conceptos y los bloques de código que no son relevantes para TFLite y la detección de objetos no se explican y se proporcionan para que simplemente los copies y los pegues (Fidalgo, 2022, p. 12).

2.8.1.3. Redes Neuronales Convolucionales

Las redes neuronales convolucionales (CNN) son usadas para trabajar con imágenes ya que son un tipo de algoritmo de *Deep Learning*, son capaces de detectar objetos simples en las primeras capas, y con los procesos de entrenamiento poco a poco se van especializando en reconocer formas u objetos más complejos (Llano-Carmona, 2021, p. 8).

Para las extracciones de características de los objetos se utiliza la composición de capas convolucionales, capas de agrupación y capas totalmente conectadas.

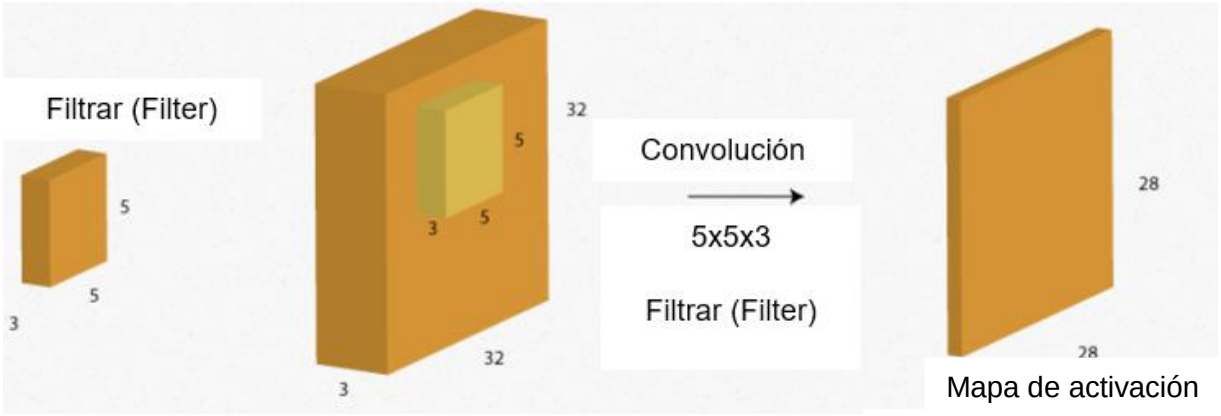
Figura 5: *Redes Neuronales Convolucionales*



Nota: Redes Neuronales Convoluciones, Llano-Carmona, 2021,Capas convolucionales hacia el perceptron

Las capas de convolución se realizan por la multiplicación de la matriz de datos de entrada por una matriz bidimensional de filtros ya que son operaciones lineales, el filtro se utiliza para detectar características en la parte de la entrada de datos, en la cual dicho filtro permite el desplazamiento de derecha a izquierda y de arriba hacia abajo obteniendo un resultado de una matriz bidimensional o también llamada *feature map*, es muy importante el entrenamiento del modelo de nuestra red neuronal porque aprenderá cuales son los valores adecuados para nuestro filtro (Llano-Carmona, 2021, p. 9).

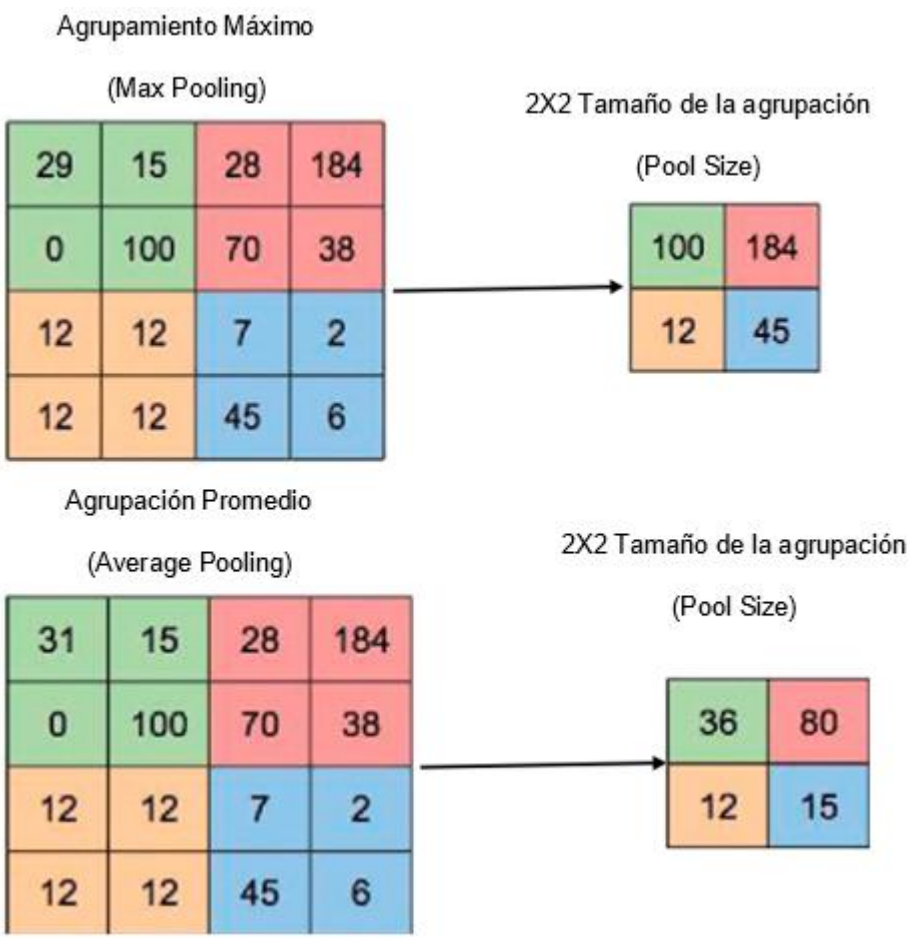
Figura 6: *Capa de Convolución*



Nota: Capa de convolucion, Llano Carmona, 2021, Capa de convolucion

Las capas de agrupación o también llamadas pooling layers son utilizadas para reducir las dimensiones del alto y el ancho de la entrada de la siguiente capa convolucional. Es utilizada para la reducción de sobrecarga de cálculo ya que al utilizar muchas capas las redes neuronales convolucionales el coste computacional es muy elevado para el procesamiento de los parámetros y esto nos conlleva a reducir el tamaño, aunque vamos a perder información, pero nos ayudará para realizar el sobreajuste. (Llano-Carmona, 2021, p. 12)

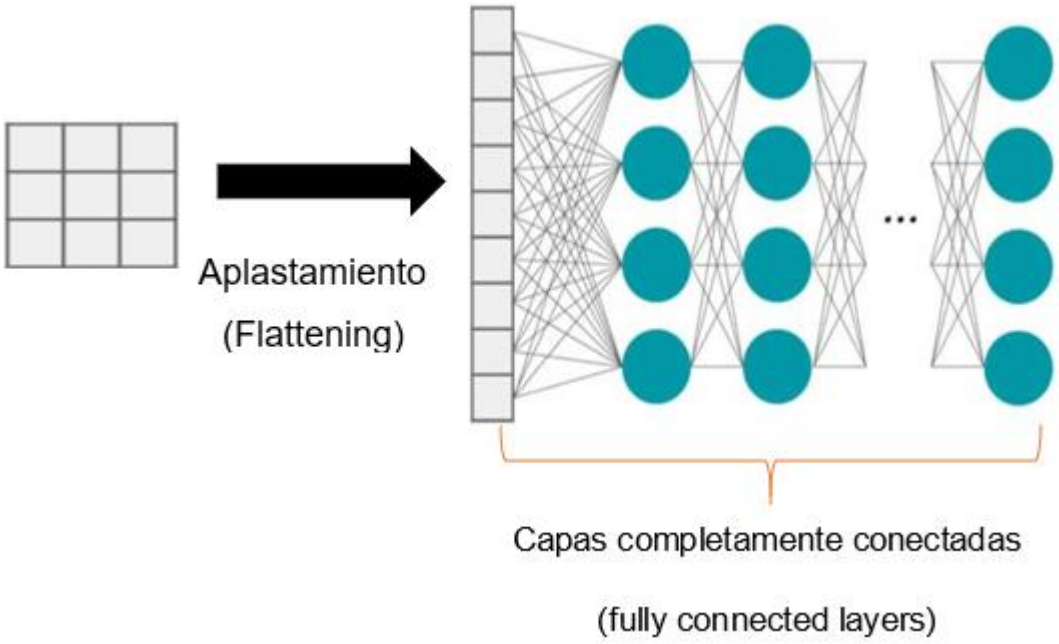
Figura 7: *Max Pooling y Abaraje Pooling*



Nota: Max Pooling y Abaraje Pooling, Llano-Carmona, 2021, Operaciones de agrupacion de Redes Neuronales Convolucionales.

Las capas totalmente conectas o fully-connected layers conectan a una sola matriz en una sola dimensión llamada flatten, que es conectada con la capa de salida que tendrá las neuronas con las clases que estamos clasificando en la cual se va a predecir la etiqueta correcta.

Figura 8: *Capa Flatten*



Nota: Capa Flatten,Llano-Carmona, 2021, Arquitectura Generica de un Red Neuronal.

CAPÍTULO III

MARCO APLICATIVO

CAPÍTULO III - MARCO APLICATIVO

3. INTRODUCCION

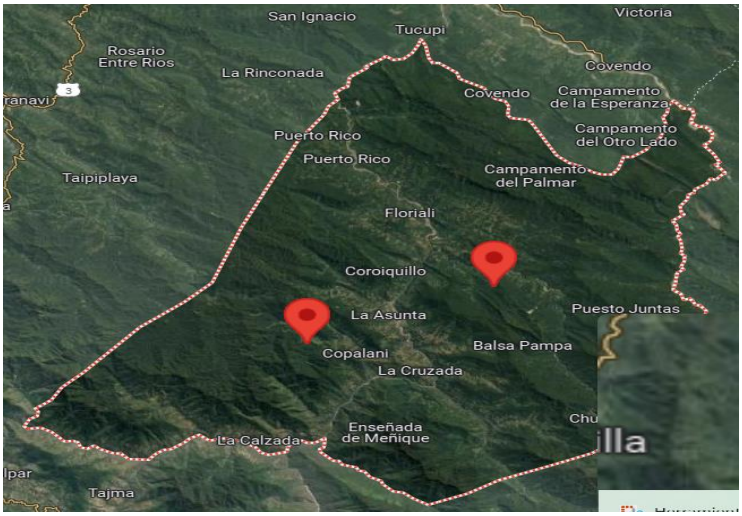
En este capítulo se describe la forma de organización y descripción del anterior capítulo, se hará el uso de todas las herramientas mencionadas y metodologías descritas anteriormente, que nos servirá para el desarrollo de la aplicación y como se utilizará en el contexto agrícola.

3.1. GENERALIDADES DE LA SOCIACIÓN APEN

3.1.1. Ubicación Geográfica

La comunidad Nogalani, está ubicado a una altura 1800 m.s.n.m., dentro el municipio la Asunta, de la provincia sud yungas del departamento de La Paz. Teniendo un clima muy agradable considerado como cálido, y apto para la producción de café.

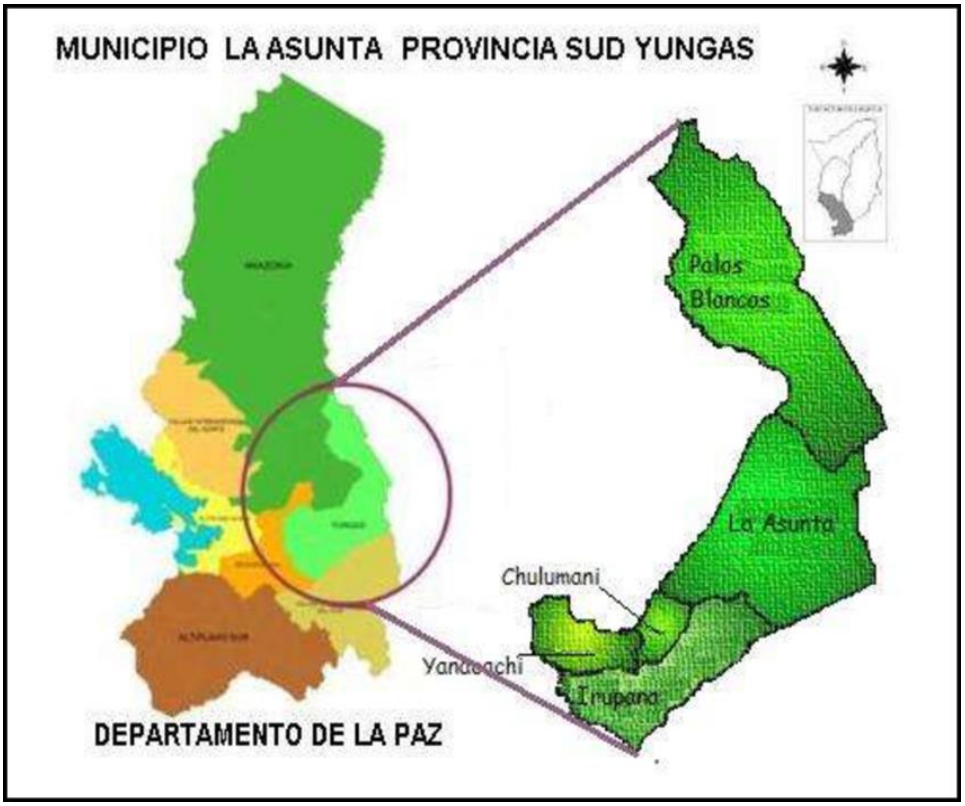
Figura 9: *Ubicación Geográfica*



Nota: Google Maps, 2024, Ubicación geografica del Municipio la Asunta.

El municipio de la Asunta cuenta con distintos tipos de producción, como cultivos de café, coca, plátanos, cítricos y otros.

Figura 10: *Ubicación Geográfica del Municipio La Asunta*



Nota: Ubicación del municipio de La Asunta, 2018, Provincia Sud Yungas.

3.2 DESARROLLO DE LA METODOLOGÍA MOVIL -D

En el desarrollo de la metodología se aplicará las fase o etapas para el desarrollo de la aplicación móvil.

3.2.1 Análisis de Requerimiento

En esta fase es comprender en profundidad las plagas que afectan a las plantaciones de café y establecer una base de datos adecuada con imágenes relevantes para el entrenamiento del modelo. También se busca identificar los

desafíos específicos de la aplicación y definir las características clave del modelo que se desarrollará.

Tabla 11:*Plagas de la Planta de Café*

NOMBRE	DESCRIPCION	SINTOMAS
Arañita roja (Tetranychus urticae)	• Defoliación y estrés en la planta • Daño follaje • Reducción en la productividad	
Roya (Hemileia vastatrix)	• Manchas amarillas o naranjas, con presencia de polvo fino amarillo • Defoliacion	
Hoja sana	Son verdes durante todo el año	

3.2.1.1. Requerimientos Funcionales

Se describe todas las características que se requiere para el desarrollo del sistema aplicativo generado con CNN.

Tabla 12: *Requerimientos Funcionales*

N°	PASOS DE INGRESO A APP	OBSERVACIONES
F1	Acceso a la aplicación móvil Agrocafe.	El usuario tendrá el acceso a todas funciones de la APP.
F2	Escaneo de imagen por cámara	Realizara el escaneo con la cámara del equipo en hojas de las de café.
F3	Selección de imágenes	Realizara la selección de imágenes de hojas de café de la galería del equipo.
F4	Información de las plantas de café	El agricultor tendrá el acceso a la información rápida a las plagas de la planta de café.
F5	Información APEN y manual de siembra	Información de la institución Asociación de Productores Ecológicos de Café Nogalani.
F6	Manual de carga de imagen	Ajustara automáticamente en el modelo de CNN.
F7	Modelo de modelo de CNN	Reconocimiento de plagas
F8	Módulo de resultados	Información sobre las plagas detectadas

3.2.2. Requerimientos No Funcionales

Los Requerimientos no funcionales (RFN), se describen cualidades y características del sistema que el usuario puede visualizar para ser efectivo, eficiente y adecuado, a continuación, se muestra la siguiente tabla.

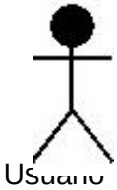
Tabla 13: *Requisitos no Funcionales*

N°	NO FUNCIONALES
FN1	La aplicación debe tener funcionamiento eficaz de la detección de las enfermedades mediante las fotos o imágenes
FN2	El dispositivo móvil debe constar con el sistema Android.
FN3	El dispositivo móvil debe ser de la clase gama media o gama alta.
FN4	La calidad de la cámara o fotografías deben de ser de equipos con buena calidad de cámara.
FN5	El dispositivo móvil debe de contar con espacio disponible para la aplicación

3.2.3 Definición de Actores

En la sección que se muestra se verá un desglose detallado de las responsabilidades y funciones vinculadas al actor de nuestra aplicación móvil.

Tabla 14 : *Lista de Actor*

ACTOR	FUNCIONES
 Usuario	• Acceso a la aplicación • Verificación de iconos • Acceso a la cámara • Acceso a galería de imágenes • Carga de la interfaz de la enfermedad correcta en tiempo real • Generación del tipo de enfermedad • Verificación a la información de APEN

3.2.4 Descripción de Funciones

Tabla 15: *Descripción de Funciones*

ACTOR	FUNCIONES
 USUARIO	El usuario tendrá el acceso a todas las opciones de la aplicación móvil.

3.3 DISEÑO DE LA APLICACIÓN

3.3.1 Diagrama de Casos de Uso

Los diagramas de casos de uso se emplean para representar las interacciones entre el usuario y la aplicación. A continuación, se muestran los casos de uso del sistema:

Figura 11: *Caso de Uso General*

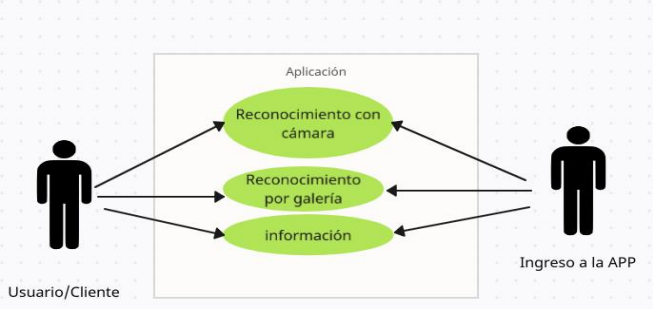


Tabla 16: *Caso de Uso General*

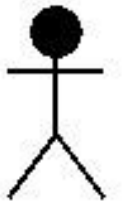
CASOS DE USO	CASO DE USO GENERAL
 Usuario	Usuario, Escaneo de imagen, selección de imagen, información rápida, información
Descripción	El usuario ingresará a la APP, tendrá opciones de realizar la búsqueda de diferentes iconos que formarán de diferente manera primero.

Figura 12:*Proceso de Captura*

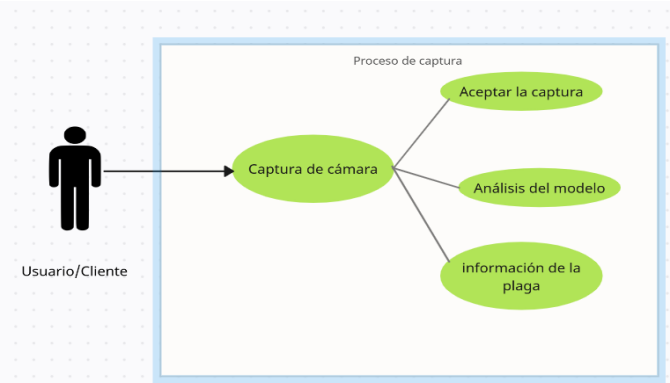


Tabla 17: *Procesamiento de Captura de Imagen*

CASO DE USO	CASO DE USO EN PROCESAMIENTO
Usuario	Caficultor
Tipo	Primario esencial en el procesamiento de imagen mediante Inteligencia Artificial
Descripción	Ingresando a la APP adicionalmente deberá ingresar a la captura de imagen y recepción el resultado esperado.

Figura 13:*Imagen por Galería*

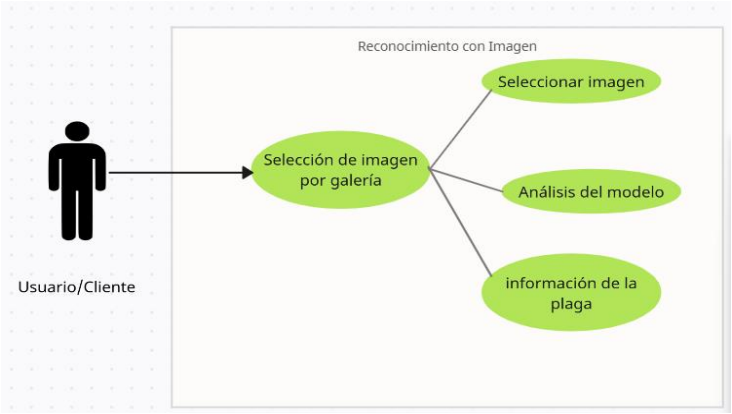


Tabla 18:*Imagen por Galería*

CASO DE USO	CASO DE USO EN PROCESAMIENTO
Usuario	Caficultor
Tipo	Primario esencial en el procesamiento de imagen mediante Inteligencia Artificial
Descripción	Ingresando a la APP adicionalmente deberá ingresar a galería seleccionar una imagen y recepción el resultado esperado

Figura 14:*Procesamiento de Carga de Imagen*

Tabla 19:*Procesamiento de Carga de Imagen*

CASO DE USO	CASO DE USO CARGA DE IMAGEN
Usuario	Caficultor
Descripción	Al seleccionar una imagen se realizará el proceso de análisis del dato.

Figura 15:*Procesamiento de Imagen en el Modelo de CNN*

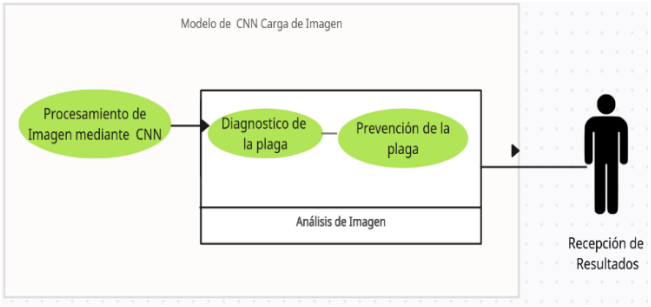


Tabla 20:*Procesamiento de Imagen de Modelo CNN*

CASO DE USO	CASO DE USO DEL MODELO CNN
Usuario	Caficultor
Descripción	El dato optado se analizará en el modelo entrenado decepcionando un resultado.

Figura 16:*Módulo de resultado*

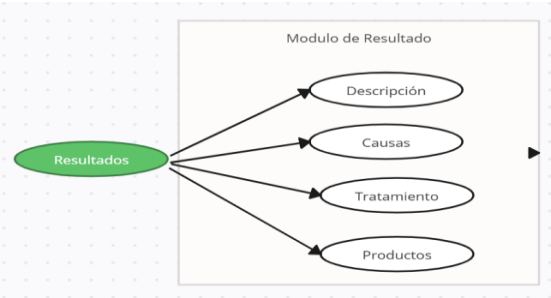


Tabla 21:*Módulo de Resultado*

CASO DE USO	CASO DE USO DEL RESULTADO
Usuario	Caficultor
Descripción	El resulta tendrá diferentes tipos de información para el agricultor

Figura 17:*Información de APEN*

Tabla 22: Información de APEN

CASO DE USO	CASO DE USO DE INFORMACION
Usuario	Caficultor
Descripción	En la opción información se encuentran archivos de la Asociación.

El agricultor podrá informase sobre el manejo de plantación de café, a partir de preparación del suelo, elección de la variedad, producción de plántulas(vivero), trasplante al campo, poda y manejo de la planta, cosecha y beneficios del secado del producto del café.

***Figura 18**: Manual de Cultivo*

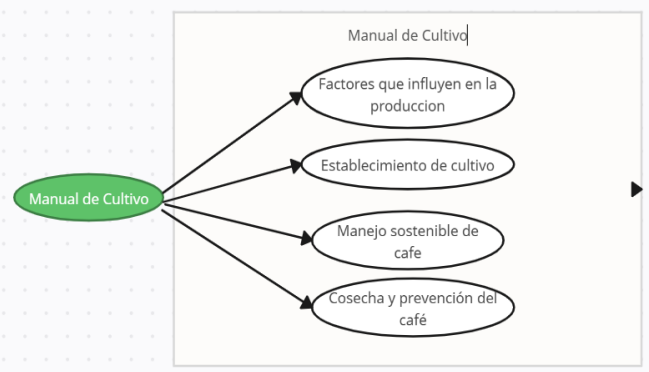


***Tabla 23:** Manual de Cultivo*

CASO DE USO	MANUAL DE CULTIVO
Usuario	Caficultor
Descripción	El agricultor podrá beneficiarse del manual de cultivo para mejorar el producto primario del café, optando mejores resultados en la familia caficultora.

Figura 19:*Información de la Cooperativa*

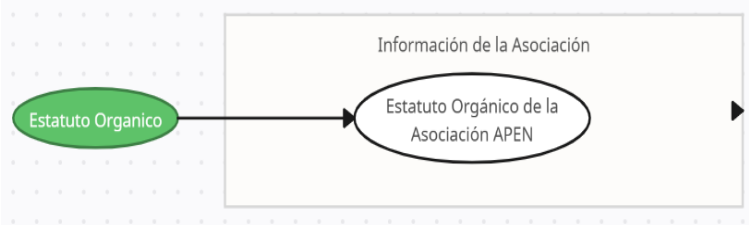


Tabla 24: *Información de la Cooperativa*

CASO DE USO	INFORMACION DE APEN
Usuario	Caficultor
Descripción	Información necesaria de la Cooperativa

3.4 REQUISITOS PREVIOS DEL DESARROLLO DE LA APLICACION

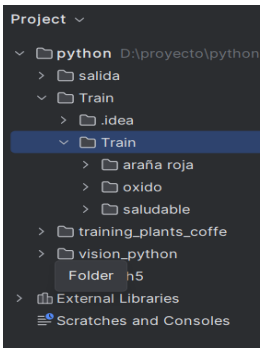
El sistema aplicativo se desarrolló en PyCharm (IDE) para desarrollar el código del modelo, con un lenguaje de programación de Python en su versión 3.12.4, TensorFlow en su versión de 2.6.3 y Keras.

Para el entrenamiento de este modelo se adquirió una laptop con un procesador Core I5 12th y una tarjeta gráfica Nvidia GeForce MX230, además, de un celular con Android 10.0 para el procesamiento de las imágenes.

3.4.1 Requisitos del Entorno de Trabajo

El presente proyecto fue desarrollado en primer lugar en lenguaje de programación Python, y el entorno de desarrollo integrado de Pycharm, se realizó la siguiente estructura de directorios:

Figura 20: *Requisitos del Entorno de Trabajo*



El directorio data se dividió en 3 sub directorios como punto de inicio de la aplicación, en las cuales son test, train, training_plants_coffe, visión_python, el sub directorio train (carga de imágenes), el sub directorio training_plants_coffe (direcciona datos y comprime model.h5 todas las imágenes), y el sub directorio visión_python (pruebas), tienen datos de 3 ,2 son enfermedades y 1 es hoja saludable que afectan a las hojas de café, siendo sub divididas en carpetas que tienen los siguientes nombres: Hoja Saludable, Arañita Roja y Roya de Hoja.

3.4.2 Procesamiento de Datos

Los datos procesados contienen una gran variedad de imágenes de hoja de café para la detección de las enfermedades. En el preprocesamiento de los datos se han eliminado detalles no deseados y se han mejorado características de los síntomas de las hojas del café.

El conjunto de datos se divide en tres partes: la primera parte contiene imágenes originales de las hojas de café recortadas de los síntomas de las hojas, segunda dataset Entrenamiento, y dataset Validación, y por último las imágenes comprimidas de los Test.

3.4.3 Descripción de Conjunto de Imágenes

El conjunto total de datos contiene 2100 imágenes que son divididas entre las 3 diferentes clases de hojas entre ellas tenemos: Hoja saludable, Phoma de hoja, Roya de hoja y hoja saludable.

En el entrenamiento de la CNN se utilizó un conjunto de imágenes de un total de 700 imágenes recortadas en sus partes asintomáticas de las hojas que están etiquetadas según diferentes clases entre ellas son: Hoja saludable, phoma de hoja y Roya de Hoja todas las imágenes tienen una dimensión de 128x 128 pixeles.

3.4.4 Presentación de Red Neuronal

El entrenamiento de la red neuronal se utilizó el modelo secuencial para clasificar el presente modelo que ha sido implementado para usar las librerías de Keras, TensorFlow en lenguaje de Python.

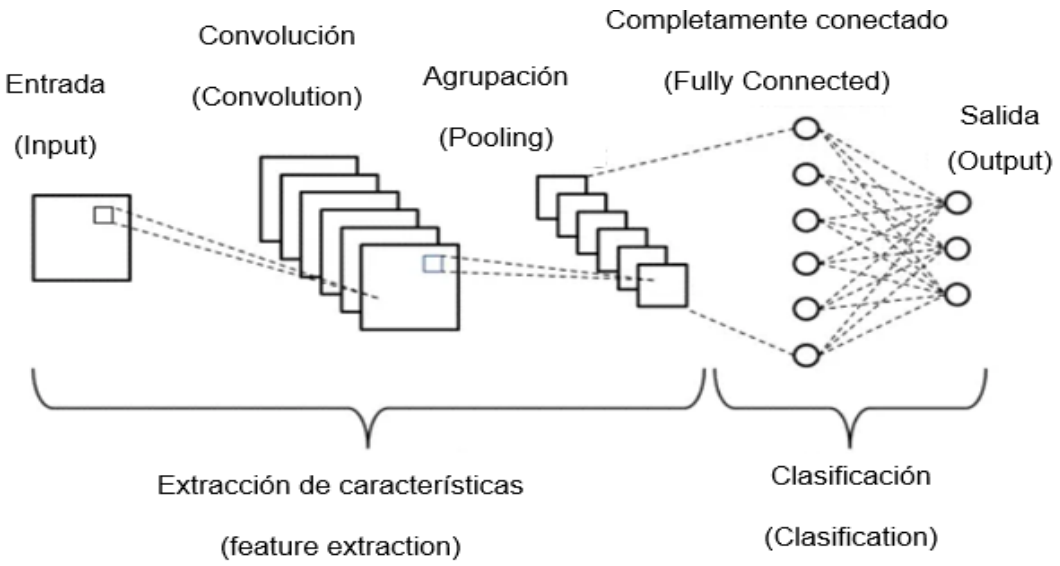
El algoritmo utilizado en la presente investigación es el algoritmo de ADAM, su rendimiento a comparación a otros algoritmos, es mucho más rápido en el entrenamiento o la inferencia del modelo aprendizaje automático para actualizar los parámetros, las imágenes proporcionadas para el entrenamiento son introducidas en lotes de 32 y un valor de épocas de 20, para el entrenamiento se utilizan imágenes de un tamaño de 128x128 píxeles pertenecientes a las 3 clases diferentes.

3.4.5 Arquitectura de la Red Neuronal Convolucional (CNN)

Se describe el proceso de la CNN a seguir el entrenamiento que compone la entrada de una capa o unas varias capas de convoluciones, una o varias capas de

agrupamiento y una o varias capas densamente conectadas. La capa de entrada consiste en un vector donde cada pixel de la imagen termina siendo una característica

Figura 21: *Arquitectura de la red Neuronal Convolucional*



Nota: Arquitectura Comun, Kumar, 2020,Redes Neuronales Convolucionales

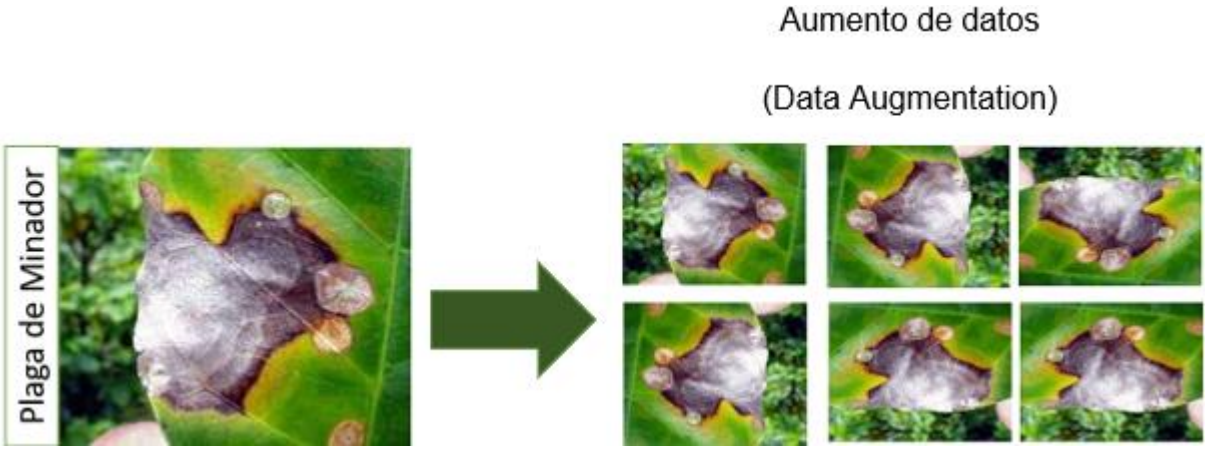
3.4.6 Datos de Aumentación

Es una técnica que permite incrementar el dataset, en caso tengamos una cantidad pequeña de imágenes. Data augmentation se utilizó para mejorar el sobreajuste y el aprendizaje de nuestra red neuronal.

Se usó esta técnica porque proporciona muchas formas de mejorar las imágenes como lo son: girar la imagen original, cambiar las condiciones de iluminación, recortarla de forma diferente de modo que la imagen pueda generar diferentes submuestras.

La biblioteca Keras, proporciona una clase llamada ImageDataGenerator, que se encarga de aplicar transformaciones espaciales a las imágenes.

Figura 22: *Data Augmentation*



3.4.7 Desarrollo del Software

En el desarrollo se detallará la codificación del software aplicando redes neuronales convolucionales basado en los modelos y diseños previamente desarrollados.

Al scanear una imagen en tiempo real mediante redes neuronales convolucionales, verificará la información el caficultor, no podrá almacenar datos obtenidos del scaneo captura de imagen.

3.4.8 Estructura de Herramientas para el Desarrollo

Para la creación de la aplicación móvil Agrocafe, se ha utilizado como entorno de desarrollo Android studio, y Pycharm, para el modelo se ha realizado en el lenguaje de programación Python, se ha utilizado Keras con TensorFlow como bibliotecas de software, Open CV se utilizó como procesamiento de imágenes en tiempo real y Pip administrador de paquetes de las bibliotecas.

Figura 23:*Estructura de Herramientas para el Desarrollo*

3.4.9. Pycharm (IDE)

Facilitará la escritura, prueba y depuración del código de Python relacionado con el modelo de aprendizaje automático y procesamiento de imágenes.

3.4.9.1 Modelo

Este modelo está basado en una arquitectura en la cual veremos un mayor número de capas convolucionales y se implementó la técnica de Early stopping.

3.4.9.2 Datos

Para la importación de los datos del dataset, se utilizó la clase que proporciona Keras llamada ImageDataGenerator, en donde se va aprovechar para reescalar los píxeles entre los valores 0 y 1. Dicha clase ofrece un método llamado Flow-from-directory a los que vamos a pasar los siguientes parámetros: la ruta de nuestro directorio, el tamaño de nuestras imágenes (128x128), los canales de color

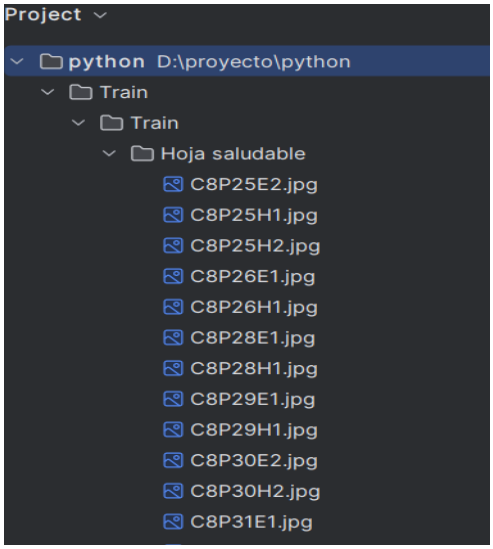
RGB (3 canales), el tamaño de los lotes o “batch size” (32) y el tipo de clase (“categorical”).

En la clase categorical se aplica en el dataset el llamado “one-hot encoding”, que asigna la etiqueta correspondiente a cada imagen en lugar de un valor numérico. Este proceso se llevó a cabo tanto en las imágenes de entrenamiento como en las imágenes de validación y test.

3.4.9.3 Procesamiento del Modelo

El funcionamiento de Pychar IDE se encuentran subcarpetas almacenando los datos que contienen campos que se asignan valores, estas fotos se almacenan en colección, que son contenedores de las fotos que utiliza la aplicación móvil para la organización.

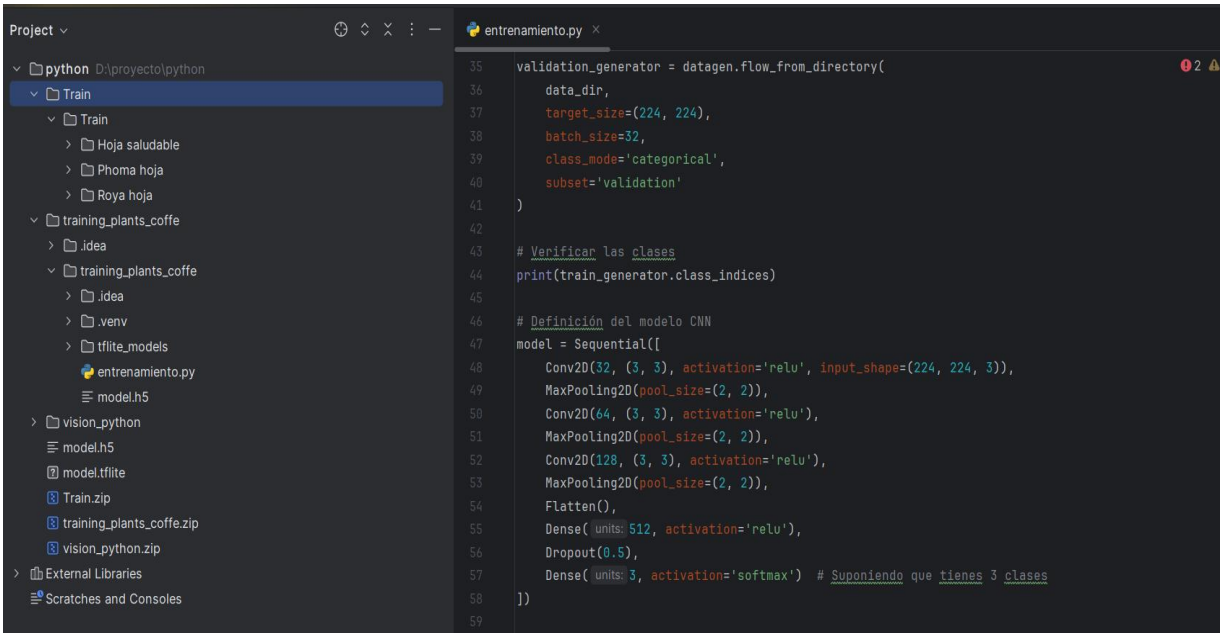
Figura 24:Procesamiento del Modelo



3.4.9.4 Entrenamiento

Para comprimir el model.h5 se importó diferentes tipos de librerías se genera datos de entrenamiento y validación verificando las clases.

Figura 25: *Entrenamiento del Modelo*



3.4.9.5 Entrenamiento del Modelo

La biblioteca Keras tiene una función llamada Fit, la cual permitió entrenar a nuestro modelo con los siguientes parámetros:

Primeramente, se agregó un generador de imágenes de entrenamiento, redirecciona el directorio de datos, carga datos de entrenamiento y validación la cual ayudo con el número de pasos por lotes “batch size” en cada época, seguido agregamos un generador de imágenes de validación con su respectivo parámetro llamado "validation steps" equivalente a “steps_per_epoch”, por último el parámetro de “epochs” nos indica el número de pasadas de todo el conjunto de entrenamiento

la cual lo asignamos el valor de 20 y finalmente “model.save” se guarda el modelo entrenado direccionando “E:/proyecto/python/model.h5”.

Por lo tanto, se agregó la técnica de regularización llamada “Early Stopping”, esta técnica permite entrenar grandes épocas de entrenamiento y se detiene una vez que el modelo del conjunto de datos visualiza que el rendimiento del entrenamiento no mejora la ganancia, en nuestro caso hemos monitorizado el valor de “val_loss”, agregando un parámetro de “patience” con un valor de 3, esto quiere decir que debe haber alguna mejora en nuestros indicadores en 3 épocas.

Figura 26: *Entrenamiento*

```
17/17 ━━━━━━━━━━━━━━━━━━━━ 42s 2s/step - accuracy: 0.3238 - loss: 3.1568 - val_accuracy: 0.4062 - val_loss: 1.0464
Epoch 2/20
 1/17 ━━━━━━━━━━━━━━━━━━━━ 23s 1s/step - accuracy: 0.4062 - loss: 1.0798C:\Users\WINDOWS 11 PRO\AppData\Local\Progra
  self.gen.throw(value)
17/17 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.4062 - loss: 1.0798 - val_accuracy: 0.5455 - val_loss: 1.0303
Epoch 3/20
17/17 ━━━━━━━━━━━━━━━━━━━━ 39s 2s/step - accuracy: 0.4316 - loss: 1.0565 - val_accuracy: 0.4297 - val_loss: 1.0754
Epoch 4/20
17/17 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.4062 - loss: 1.0866 - val_accuracy: 0.6364 - val_loss: 1.0855
Epoch 5/20
17/17 ━━━━━━━━━━━━━━━━━━━━ 39s 2s/step - accuracy: 0.4207 - loss: 1.0658 - val_accuracy: 0.3906 - val_loss: 1.0399
Epoch 6/20
17/17 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.4375 - loss: 1.0916 - val_accuracy: 0.4545 - val_loss: 0.9801
Epoch 7/20
17/17 ━━━━━━━━━━━━━━━━━━━━ 41s 2s/step - accuracy: 0.4304 - loss: 1.0433 - val_accuracy: 0.4062 - val_loss: 1.1357
Epoch 8/20
17/17 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.5000 - loss: 1.0332 - val_accuracy: 0.6364 - val_loss: 0.7619
Epoch 9/20
17/17 ━━━━━━━━━━━━━━━━━━━━ 40s 2s/step - accuracy: 0.4587 - loss: 1.0170 - val_accuracy: 0.4844 - val_loss: 0.9916
Epoch 10/20
17/17 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.4062 - loss: 0.9986 - val_accuracy: 0.2727 - val_loss: 0.9563
Epoch 11/20
17/17 ━━━━━━━━━━━━━━━━━━━━ 39s 2s/step - accuracy: 0.5143 - loss: 0.9683 - val_accuracy: 0.4453 - val_loss: 1.0251
Epoch 12/20
17/17 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - accuracy: 0.4062 - loss: 0.9764 - val_accuracy: 0.5455 - val_loss: 0.9769
Epoch 13/20
17/17 ━━━━━━━━━━━━━━━━━━━━ 39s 2s/step - accuracy: 0.4048 - loss: 1.0094 - val_accuracy: 0.4453 - val_loss: 1.0541
Epoch 14/20
17/17 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - accuracy: 0.4062 - loss: 1.0516 - val_accuracy: 0.5455 - val_loss: 0.9888
```

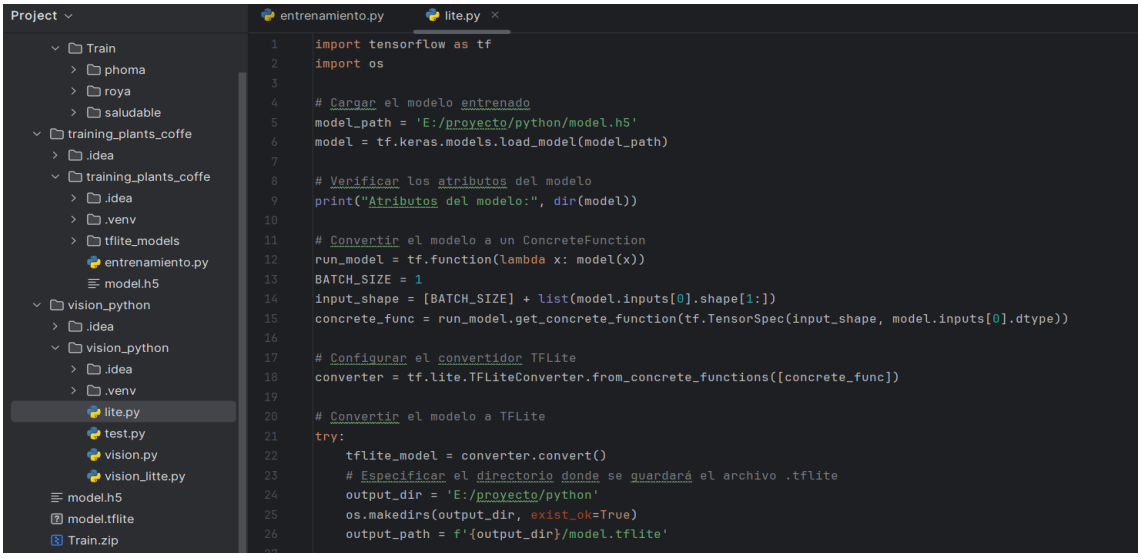
3.4.9.6 Proceso del modelo entrenado

Para el procesamiento de datos de entrada y de salida, se ha utilizado la biblioteca de TensorFlow, en la cual la Api del intérprete de TensorFlow ejecuta el modelo de aprendizaje automático en el dispositivo utilizando los tensores de forma de ByteBuffer, en la cual los datos son difíciles de manipular y depurar, siendo el objetivo final la cuantización de objetos de entrada y salida dando un resultado a una probabilidad de un punto flotante entre 0 y 1.

3.4.9.7 Carga del Modelo Entrenado

Primero se realizó la carga del modelo entrenado con" model_path", se verifica los atributos del modelo, convierte el modelo a un ConcreteFunction, configura el convertidor de TFLite" converter", convertir el modelo a TensorFlow Lite" tflite_model", especificar el directorio donde se guardará el archivo.tflite, guardar el modelo convertido en el directorio especificado.

Figura 27:*Carga del Modelo Entrenado*



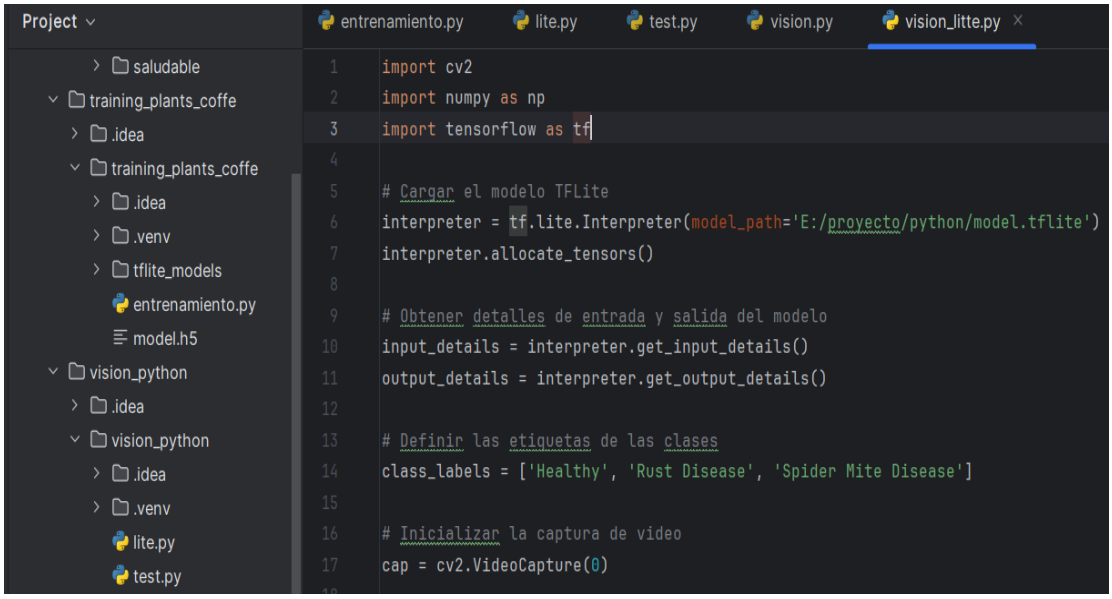
3.4.9.8 Proceso del Modelo a TensorFlow Lite

El proceso que se debe a seguir:

- Cargar el modelo TFLite
- Obtener detalles de entrada y salida del modelo
- Definir las etiquetas de las clases
- Inicializar la captura de video
- Procesar la captura del fotograma
- Configurar el tensor de entrada del modulo
- Obtener la salida del modelo
- Mostrar el resultado

3.4.9.9 Proceso del Modelo a TensorFlow Lite

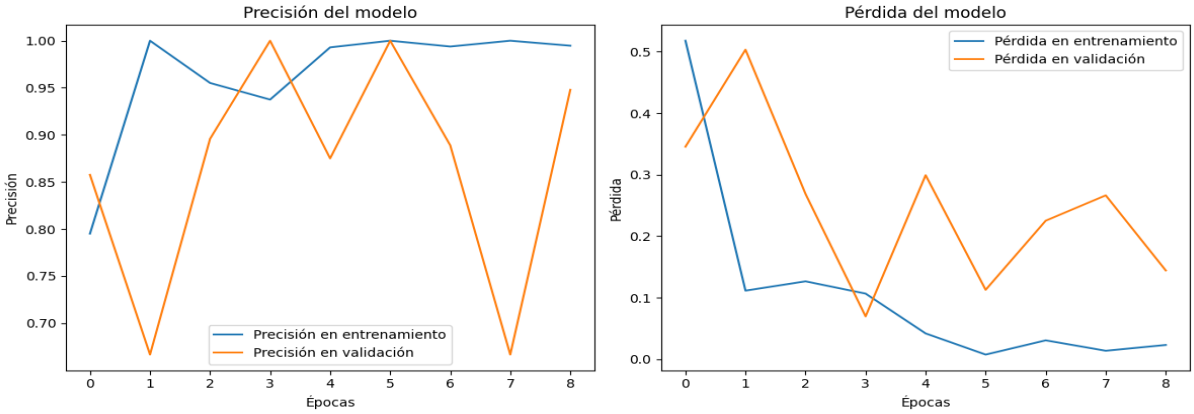
Figura 28: *Proceso del Modelo a TensorFlow Lite*



3.4.9.10 Grafica de Precisión

Es una gráfica que se utiliza para evaluar el rendimiento de modelos de clasificación, especialmente en conjuntos de datos desequilibrados. La curva muestra la relación entre precisión y épocas que se va realizar el entrenamiento.

Figura 29: *Grafica de Predicción*



3.4.9.10.1 Precisión del Modelo

- **Precisión en Entrenamiento (Línea Azul)**

Esta línea muestra la exactitud del modelo en el conjunto de datos de entrenamiento.

Observamos que la precisión en el entrenamiento se mantiene alta (por encima de 0,9) en la mayoría de las épocas. Esto sugiere que el modelo está "aprendiendo" bien estos datos y es capaz de reconocer patrones en ellos.

Sin embargo, cuando la precisión en el entrenamiento es tan alta de manera constante, esto podría ser una señal de sobreajuste (overfitting), especialmente si el modelo no mantiene un desempeño similar en los datos de validación.

- **Precisión en Validación (Línea Naranja)**

Esta línea muestra qué tan bien el modelo predice correctamente en un conjunto de datos de validación, es decir, datos que el modelo no ha visto antes.

Notamos que la precisión en la validación fluctúa bastante. Esta variabilidad (subidas y bajas) significa que el modelo no es consistente en sus predicciones para datos nuevos, lo cual sugiere una falta de generalización.

3.4.9.10.2 Perdida del Modelo

- **Pérdida en Entrenamiento (Línea Azul)**

La pérdida representa el “error” del modelo, es decir, cuán lejos están sus predicciones de los valores reales.

La pérdida en el entrenamiento disminuye constantemente a lo largo de las épocas, lo cual es una buena señal en general, ya que indica que el modelo está aprendiendo y mejorando su precisión en el conjunto de entrenamiento.

- **Pérdida en Validación (Línea Naranja)**

La pérdida en validación no disminuye de forma constante; en su lugar, muestra subidas y bajadas.

Las fluctuaciones en la pérdida de validación, incluso cuando la pérdida en entrenamiento sigue bajando, indican que el modelo no está siendo capaz de predecir bien en datos nuevos.

3.4.10 Procesamiento en Android Studio

En Android estudio se realizará la ejecución de tareas computacionales como la inferencia de modelos, pre procesamiento de imágenes o datos, y la generación de resultados del modelo entrenado.

3.4.10.1 Configuración del Entorno de Desarrollo

Android Studio es el entorno de desarrollo integrado (IDE) seleccionado para implementar la aplicación, gracias a sus herramientas de integración específicas para Android y su capacidad de manejo de procesamiento gráfico y redes neuronales. Se configuraron los siguientes elementos en Android Studio:

- Instalación de dependencias
- Gradle scripts

3.4.10.2 Integración de Redes Neuronales Convolucionales

El procesamiento de las redes neuronales convolucionales (CNN) para el reconocimiento de plagas se lleva a cabo utilizando un modelo previamente entrenado que está optimizado para ejecutarse en Android. Los pasos clave incluyen:

- Conversión del modelo
- Cargar el modelo en la aplicación

  Copiar el archivo **t**flite en el directorio assets de tu proyecto Android.

Figura 30: *Vista del Proyecto y Modelo Entrenado*

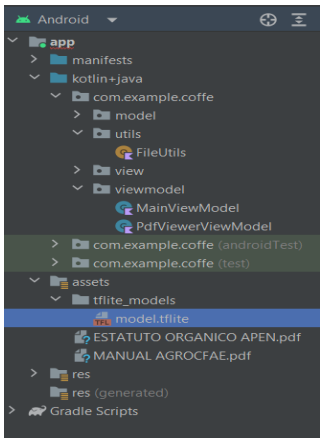
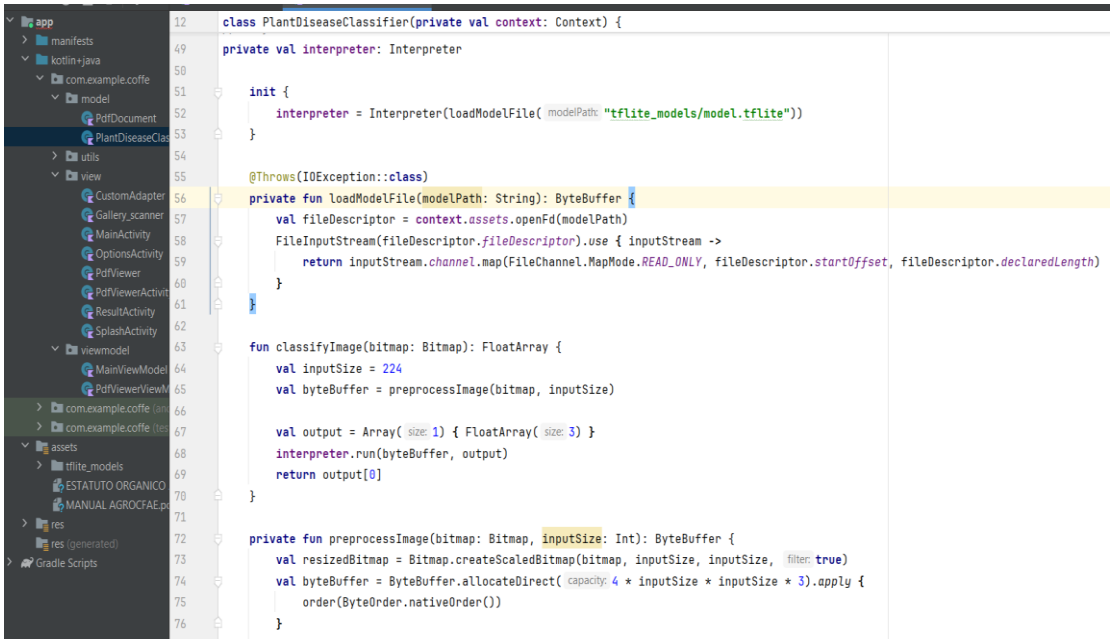
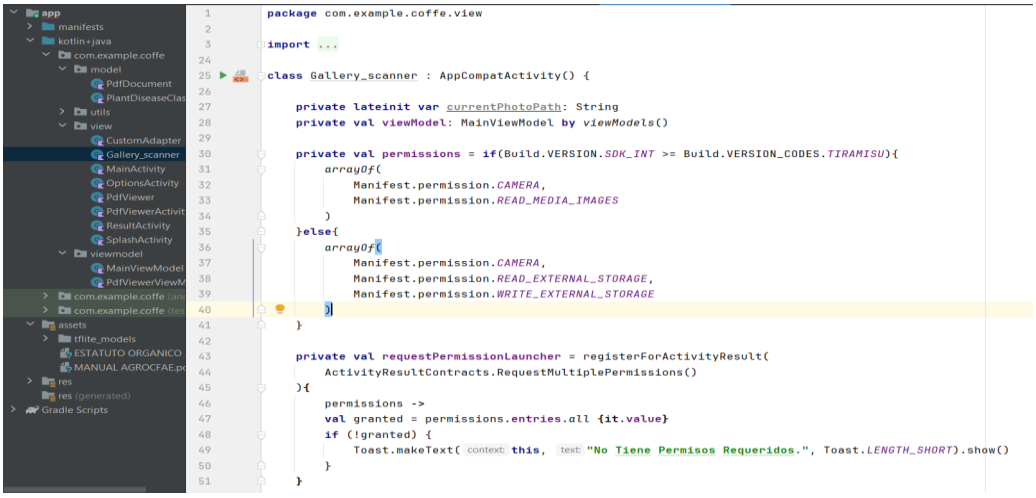


Figura 31: *Carga de Archivos del Modelo*



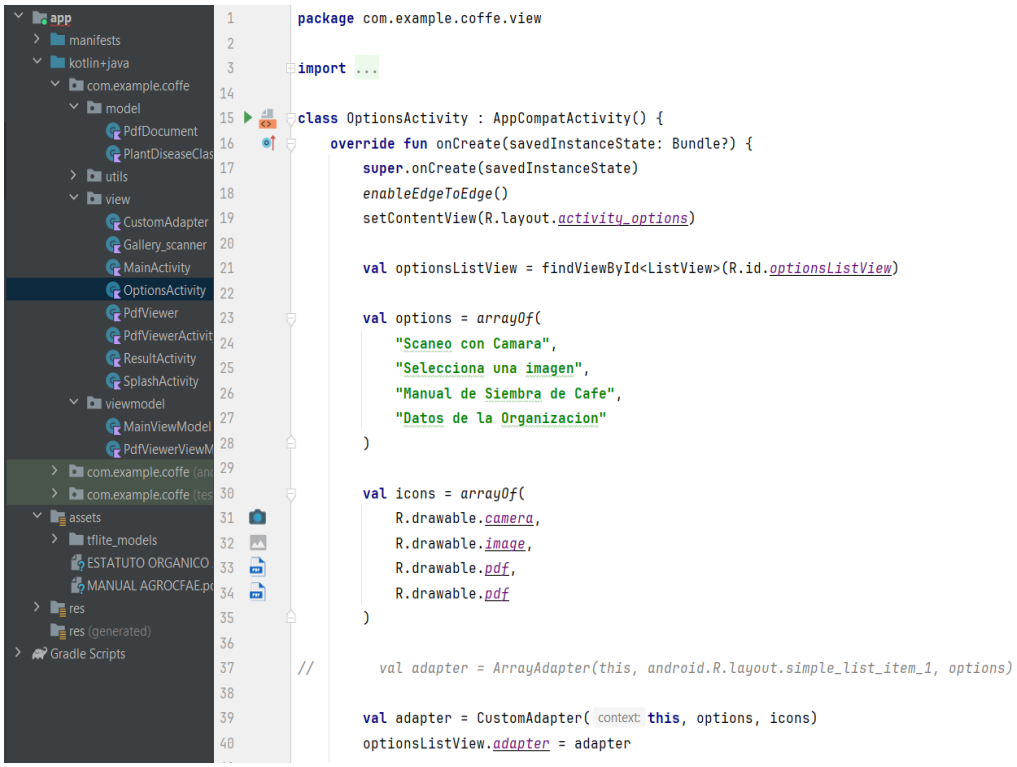
```
class PlantDiseaseClassifier(private val context: Context) {
private val interpreter: Interpreter

    init {
        interpreter = Interpreter(loadModelFile( modelPath: "tflite_models/model.tflite"))
    }

    @Throws(IOException::class)
    private fun loadModelFile(modelPath: String): ByteBuffer {
        val fileDescriptor = context.assets.openFd(modelPath)
        FileInputStream(fileDescriptor.fileDescriptor).use { inputStream ->
            return inputStream.channel.map(FileChannel.MapMode.READ_ONLY, fileDescriptor.startOffset, fileDescriptor.declaredLength)
        }
    }

    fun classifyImage(bitmap: Bitmap): FloatArray {
        val inputSize = 224
        val byteBuffer = preprocessImage(bitmap, inputSize)

        val output = Array( size: 1) { FloatArray( size: 3) }
        interpreter.run(byteBuffer, output)
        return output[0]
    }

    private fun preprocessImage(bitmap: Bitmap, inputSize: Int): ByteBuffer {
        val resizedBitmap = Bitmap.createScaledBitmap(bitmap, inputSize, inputSize,  filter: true)
        val byteBuffer = ByteBuffer.allocateDirect( capacity: 4 * inputSize * inputSize * 3).apply {
            order(ByteOrder.nativeOrder())
        }
```

Figura 32: *Permisos de la Cámara*



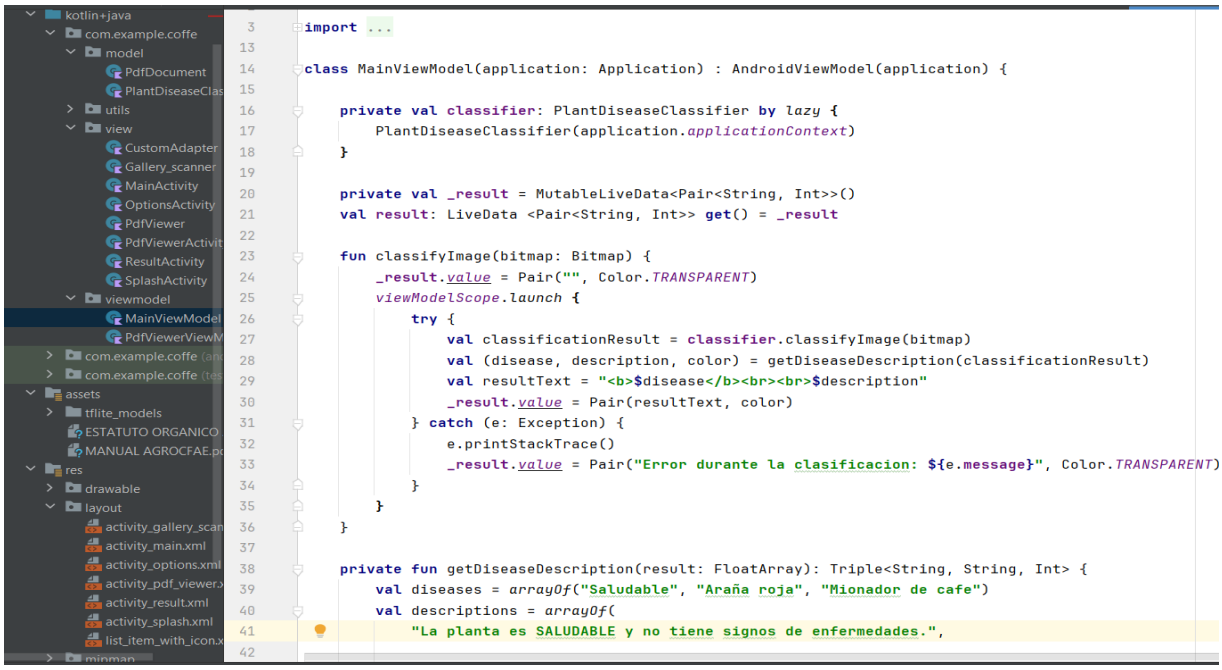
```
package com.example.coffe.view

import ...

class Gallery_scanner : AppCompatActivity() {

    private lateinit var currentPhotoPath: String
    private val viewModel: MainViewModel by viewModels()

    private val permissions = if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU){
        arrayOf(
            Manifest.permission.CAMERA,
            Manifest.permission.READ_MEDIA_IMAGES
        )
    }else{
        arrayOf(
            Manifest.permission.CAMERA,
            Manifest.permission.READ_EXTERNAL_STORAGE,
            Manifest.permission.WRITE_EXTERNAL_STORAGE
        )
    }

    private val requestPermissionLauncher = registerForActivityResult(
        ActivityResultContracts.RequestMultiplePermissions()
    ){
        permissions ->
        val granted = permissions.entries.all {it.value}
        if (!granted) {
            Toast.makeText( context: this,  text: "No Tiene Permisos Requeridos.", Toast.LENGTH_SHORT).show()
        }
    }
```

Figura 33: *Listado de las Opciones*



```
package com.example.coffe.view

import ...

class OptionsActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_options)

        val optionsListView = findViewById<ListView>(R.id.optionsListView)

        val options = arrayOf(
            "Scaneo con Camara",
            "Selecciona una imagen",
            "Manual de Siembra de Cafe",
            "Datos de la Organizacion"
        )

        val icons = arrayOf(
            R.drawable.camera,
            R.drawable.image,
            R.drawable.pdf,
            R.drawable.pdf
        )

//        val adapter = ArrayAdapter(this, android.R.layout.simple_list_item_1, options)

        val adapter = CustomAdapter( context: this, options, icons)
        optionsListView.adapter = adapter
```

Figura 34:*Vista del Modelo de Resultados*

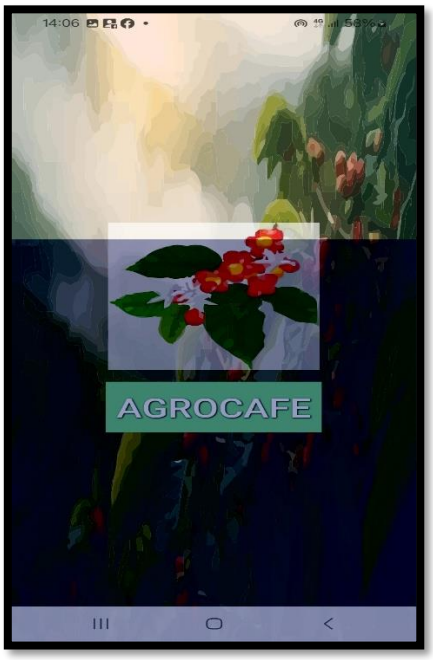


Res contiene los recursos sin código de una aplicación, como imágenes, cadena de IU.

3.4.10.3 Icono de la Aplicación

La aplicación móvil se llama AGROCAFE. Es una aplicación que detecta las enfermedades que presenta las plantas de café, está disponible para Android, esta aplicación móvil el agricultor o la persona que inicia la APP, carga automáticamente.

Figura 35: *Icono Principal*



3.4.10.4 **Menú Principal de la Aplicación**

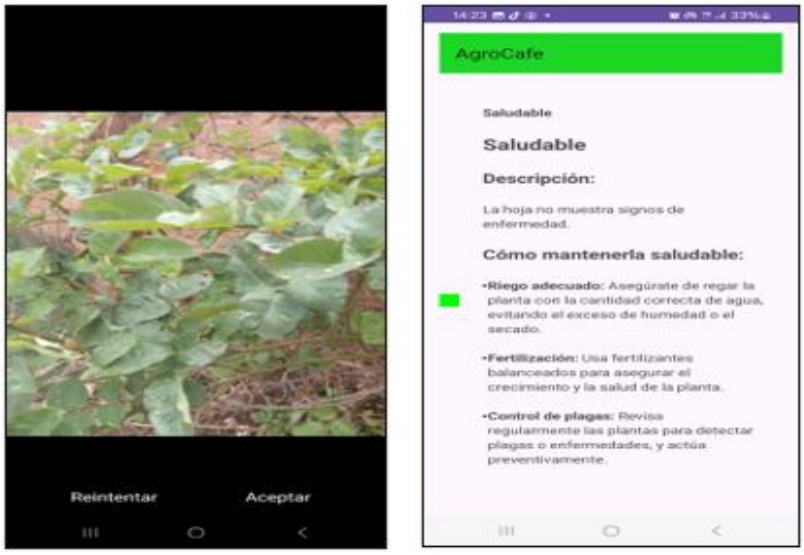
La pantalla principal de la aplicación móvil AGROCAFE mostrara tres listas de funciones que son:

Figura 36: *Menú Principal*

3.4.10.5 Reconocimiento de cámara

En la opción realizara el ingreso a la cámara del teléfono móvil para obtener una fotografía de la hoja de café con enfermedad, se recomienda que tenga una buena iluminación para el reconocimiento.

Figura 37: Reconocimiento de Cámara



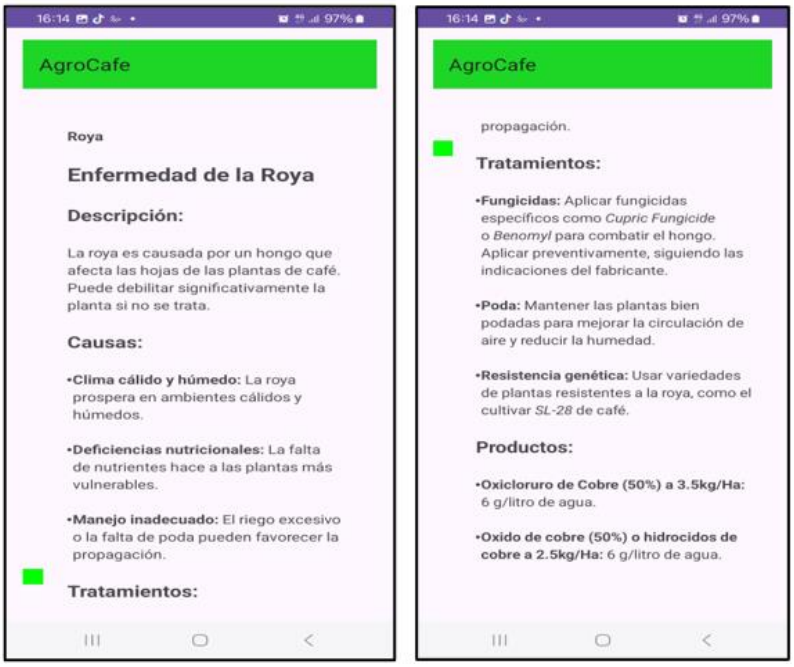
3.4.10.6 Reconocimiento con Imagen Guardada

En la opción la aplicación podrá ingresar a la galería del teléfono móvil, para seleccionar la imagen correspondiente de una hoja de café con enfermedad.

Figura 38: *Reconocimiento de la Galería de Imágenes*

Luego de pronosticar la enfermedad podrá mostrar las recomendaciones para tratar la enfermedad.

Figura 39:*Pronóstico de la Plaga de Roya*

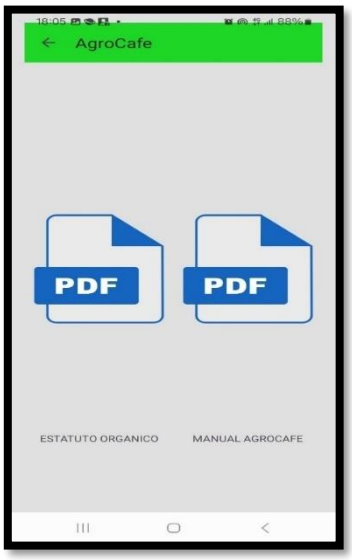


En la opción información contiene el manual de la planta de café y Estatuto Orgánico de APEN.

Figura 40: Menú de Información

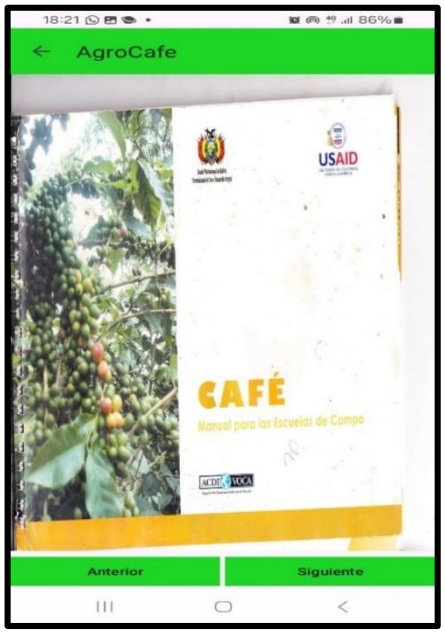
3.4.10.7 Manual de la Producción de Café

Figura 41: *Información en PDF*



En la opción tendrá un manual del campo para la producción de café en documento de PDF.

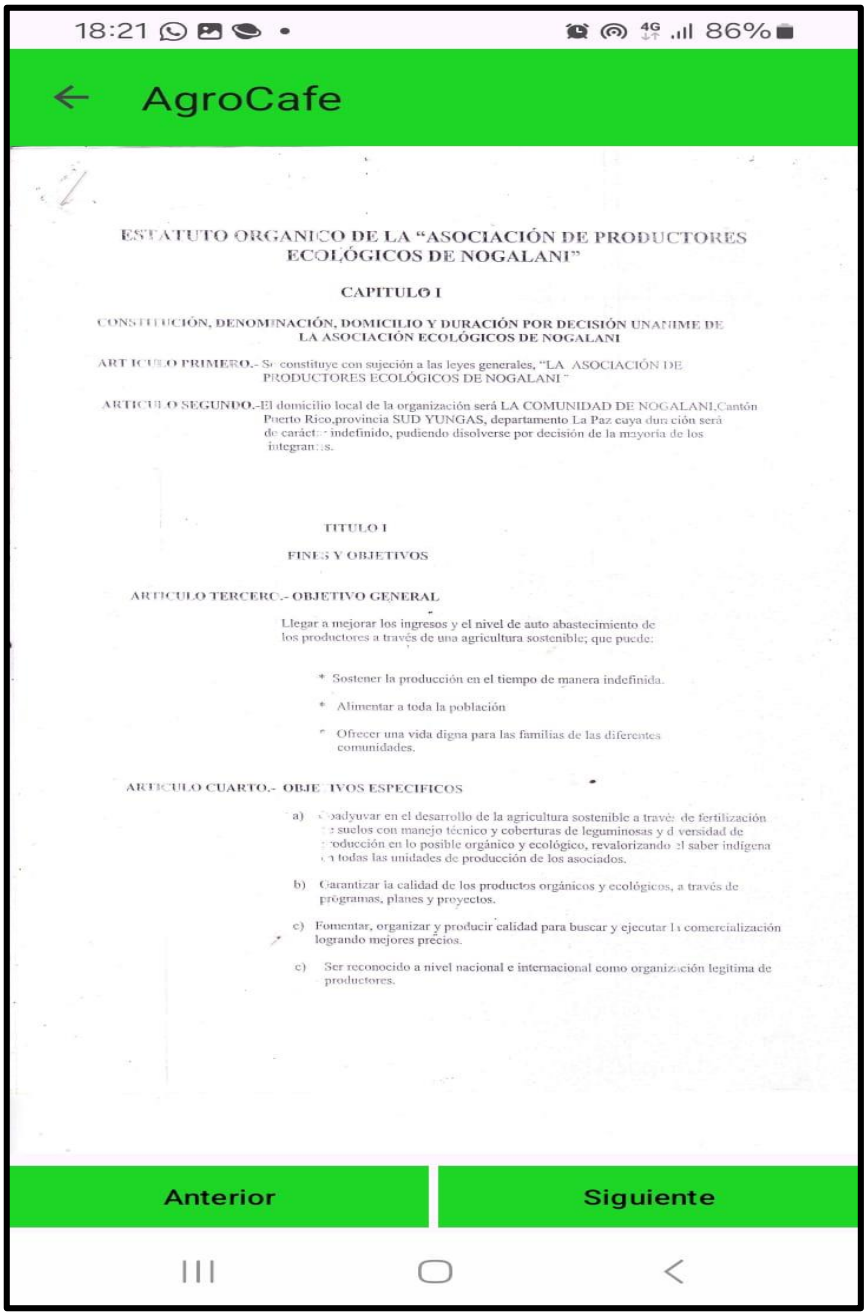
Figura 42*: Manual del Campo*



3.4.10.8 Datos de la Organización

En la opción estará los datos de la organización Asociación de Productores Ecológicos Nogalani, su estatuto orgánico en documento de PDF.

Figura 43:*Datos de la Organización*



CAPÍTULO IV

CALIDAD, COSTOS Y

CAPÍTULO IV CALIDAD, COSTO Y SEGURIDAD

4.1 METRICA DE CALIDAD DE SOFTWARE

La calidad del software es el desarrollo de software basado en estándares con la funcionalidad y el rendimiento total que satisfaga los requerimientos del cliente, todo desarrollador pone el máximo esfuerzo y dedicación, pero no siempre llega a la perfección en el producto terminado.

Al detallar estas métricas, el capítulo ofrecerá una comprensión integral de cómo evaluar y asegurar la excelencia de la aplicación en todos estos ámbitos clave.

4.2 MÉTRICAS DE CALIDAD - ESTÁNDAR ISO/IEC 25000

Para el presente Proyecto de Grado, se han implementado métricas de calidad que permiten evaluar diversos parámetros del sistema. Estas métricas son fundamentales para especificar de manera ordenada las características y atributos del software. Entre los diversos modelos y criterios de calidad para productos de software, uno de los más destacados es el ISO 25000. Este estándar ofrece un marco integral para evaluar la calidad del software y se utiliza para medir varios aspectos, incluyendo la funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad del sistema, de acuerdo al siguiente detalle:

4.2.1 Eficiencia

Es la capacidad de un sistema o modelo para producir resultados precisos y valiosos utilizando la menor cantidad posible de recursos, tiempo y esfuerzo. Para determinar su nivel, se considera una escala especifica como referencia.

Tabla 25:*Escala de Valores de Eficiencia*

ESCALA	VALOR
Excelente	5
Bueno	4
Aceptable	3
Deficiente	2
Pésimo	1

Tabla 26:*Evaluación de Eficiencia*

PREGUNTA	PORCENTAJE
¿La distribución y estilo de la interfaz permite hacer captura por cámara?	4
¿La aplicación es fácil de utilizar?	5
¿En la aplicación Agrocafe permite seleccionar una imagen por galería?	5
¿La usabilidad de la aplicación bueno?	5
¿El manual de la plantación de café es bueno?	3
TOTAL	**22**

(17)Ecuación Eficiencia

$$E = \frac{\sum x_i}{n} * \frac{100}{n} \qquad (17)$$

Donde:

$\sum x_i$ = Sumatoria de los valores de eficiencia

n = Número de preguntas

E = Eficiencia

$$E = \frac{22}{5} * \frac{100}{5} = 88\%$$

4.2.2 Portabilidad

La capacidad de la aplicación para ser transferido de un entorno a otro incluye varios aspectos importantes:

(18)Ecuación de Portabilidad

$$Portabilidad = 1 - (\frac{número\ de\ día\ para\ aportar\ el\ sistema}{número\ de\ dias\ para\ implementar\ el\ sistema}) \qquad (18)$$

Donde:

$$Portabilidad = 1 - \left(\frac{1}{12}\right) = 0.91 * 100\% = 91\%$$

Al analizar se ha obtenido el resultado que refleja el rendimiento global de la aplicación para asegura que la aplicación cumple con los estándares de la calidad y funcionalidad.

4.2.3 Usabilidad

Al evaluar la usabilidad de una aplicación es importante considerar el factor humano, asegurando que el software cumpla con los requisitos y expectativas del usuario.

(19)Ecuación de Usabilidad

$$FU = (\frac{\sum valor}{n} * 100) \quad (19)$$

Donde:

$\sum_{\text{valor}}$ = Sumatoria de los valores de la usabilidad del sistema

n = Cantidad de valores de la usabilidad del sistema

FU = Usabilidad

Tabla 27:*Indicadores de Facilidades de Uso*

ESCALA	VALOR
Muy bueno	5
Bueno	4
Regular	3
Malo	2
Pésimo	1

Tabla 28:*Usabilidad de la Aplicación*

PREGUNTA	VALOR
¿Puede Utilizar con facilidad la aplicación?	4
¿El manejo de la aplicación es complicado?	4
¿Te facilita la aplicación en detección de plagas?	4
¿El sistema cuenta con interface agradable a la vista?	4
¿Le parece complicada las funciones de la aplicación?	5
¿Se hace difícil o dificultoso aprender a manejar la aplicación?	5
¿Los resultados que proporciona la aplicación facilitan el trabajo?	4
¿Durante el uso de la aplicación se produjo errores?	3
TOTAL	**33**

A partir de los datos recabados a través del cuestionario, llevamos a cabo un análisis exhaustivo para evaluar la usabilidad de la aplicación.

$$FU = \frac{(\frac{33}{8} * 100)}{\mathbf{5}} = 82\%$$

4.2.4 Confiabilidad

La confiabilidad de una aplicación se refiere a su capacidad para mantener un nivel de funcionamiento adecuado durante un periodo de prueba establecido que implica considerar aspectos como la madurez del sistema.

(20)Ecuación de Confiabilidad

$$(t) = f * e^{-\mu * t} \qquad (20)$$

Donde:

f	=	Funcionalidad de la aplicación
μ	=	Probabilidad de error de la aplicación
t	=	Tiempo de prueba del sistema
(t)	=	Confiabilidad

$$F(t) = 0.85 * e^{-\frac{1}{10} * 16} = 0.17 * 100 = 17\%\ probabilidad\ de\ fallas$$

$$Confiabilidad = P(T \leq \mathrm{t}) = 1 - F(t)$$

$$Confiabilidad = P(T \leq \mathrm{t}) = 1 - 0.17 = 0.83$$

$$Confiabilidad = P(T \leq \mathrm{t}) = 83\%$$

4.2.5 Funcionalidad

Se llevará a cabo una exhaustiva evaluación para determinar si el software cumple con los requisitos funcionales establecidos. Este proceso garantiza que el programa posea las características esenciales para satisfacer las demandas y expectativas del usuario, asegurando su plena funcionalidad.

Tabla 29:*Número de Usuarios de Entrada*

ENTRADAS DE USUARIO	
Administración de usuarios	20
TOTAL	**20**

Tabla 30:*Número de Usuarios de Salidas*

SALIDAS DE USUARIO	
Administración de usuarios	20
TOTAL	**20**

Tabla 31:*Parametros de Medición*

PARÁMETROS DE MEDICIÓN	CANTIDAD
Número de entradas de Usuario	20
Número de salidas de usuario	20
Número de archivos	2
Numero de interfaz externa	8
TOTAL	50

Después de contar los parámetros de función, se procede al cálculo de los puntos de función sin ajustar en la tabla siguiente, este proceso implica sumar las diversas métricas recolectadas.

Tabla 32:*Índice de Usabilidad*

IMPORTANCIA	BAJO	MEDIO	ESENCIAL
¿Se requiere imágenes con buena calidad?	X		5
¿Es crítico el rendimiento?		X	4
¿La aplicación será ejecutado en un dispositivo móvil?			5
¿Se requiere una entrada interactiva para la aplicación?		X	4
¿Se requiere que la aplicación tenga múltiples ventanas?		X	5
¿Los datos generados en la aplicación son útiles?		X	4
¿Son complejas las entradas del entrenamiento de datos?			4
¿Es complejo el procesamiento interno del sistema?		X	4
¿Se ha diseñado el código para ser reutilizado?		X	4
¿Se ha diseñado la aplicación para facilitar al agricultor el trabajo y ayudarlos a encontrar la información?		X	5
¿Se ha diseñado la aplicación para ser fácilmente utilizada por el agricultor?			5
TOTAL			**49**

Fórmula para ajustar puntos de función, que considera factores como la complejidad técnica, el rendimiento y la usabilidad.

(21)Ecuación de Funcionalidad

$$PF = conteo\ total * \left[0.65 + 0.01 * \sum fi\right] \quad (21)$$

Donde:

$\sum(fi)$ = Sumatoria de los valores de los factores de ajuste.

$Conteo\ total$ = Número de parámetros del punto función

PF = Ajuste

Se obtiene la ponderación de los índices de usabilidad.

$\sum(f_i) = 49$

$$PF = 50 * [0.65 + (0.01 * 49)] = 57$$

A continuación, calculamos el PF ideal

$$PFideal = 50 * [0.65 + (0.01 * 59)] = 62$$

Entonces la adecuación funcional de la aplicación es:

$$Adecuacion\ Funcional = \left(\frac{PF}{PFideal}\right) * 100 =$$

$$Adecuacion\ Funcional = \left(\frac{57}{62}\right) * 100 = 91.93\%$$

Con el resultado de los cálculos realizado se puede interpretar de que 91.93% es funcionalidad determina que la aplicación responde de manera adecuada a las funcionalidades requeridas.

4.2.6 Mantenibilidad

Esta métrica evalúa la cantidad de trabajo requerido para realizar cambios en el sistema, ya sea para corregir errores o agregar nuevas funcionalidades. El estándar IEEE94 recomienda utilizar el índice de madurez del sistema como indicador de su estabilidad. Por lo tanto, la ecuación para este índice es la siguiente:

(22)Ecuación de Mantenibilidad

$$IMS = \frac{[Mt - (Fa + Fc + Fd)]}{Mt} \qquad (22)$$

Donde:

Mt = Número de módulos total de la versión actual

Fa = Número de módulos de la versión actual que se añadieron.

Fc = Número de módulos de la versión actual que se cambiaron.

Fd = Número de módulos que se eliminaron en la versión anterior a la actual.

IMS = Madurez del sistema

Tabla 33: Indicadores de Mantenibilidad

DESCRIPCIÓN	VALOR
Mt	9
Fa	0
Fc	1
Fd	0

A hora calculamos el IMS, usando los valores obtenidos:

$$IMS = \frac{[9 - (0 + 1 + 0)]}{9} = 0.88 * 100 = 88\%$$

El rendimiento global de la aplicación, se detalla de cada aspecto relevante para asegurar que el sistema cumple con los estándares de calidad y funcionalidad esperada.

Tabla 34:*Resultado Obtenido de la calidad*

CRITERIOS EVALUADOS	RESULTADOS
Confiabilidad	83 %
Usabilidad	92 %
Eficiencia	92%
Portabilidad	91 %
Funcionalidad	91,93%
Mantenibilidad	88%
TOTAL	**89,65%**

Con el resultado obtenido se llega a la conclusión de que el 89,93% del sistema es de calidad.

4.3. ESTIMACIÓN DEL COSTO COCOMO II

Para estimar el costo de una aplicación móvil que utiliza redes neuronales convolucionales (CNN) para el reconocimiento de plagas en plantaciones de café, puedes desglosar el proceso en varias etapas, considerando tanto los aspectos de desarrollo de software como los de entrenamiento de modelos de inteligencia. artificial. Aquí te detallamos los pasos:

Definir los requerimientos de la aplicación

Funciones Principales: Se identificará las funcionalidades de la aplicación como captura de imágenes, procesamiento de imágenes con el modelo, visualización de resultados y opciones de retroalimentación.

Interfaz de usuario (UI): se refiere al nivel de dificultad y detalle involucrado en el diseño, desarrollo y funcionamiento de la interfaz de usuario

Compatibilidad: La aplicación solo será desarrollado únicamente para Android.

Estimación de Costo de Desarrollo de Software

La estimación del costo de desarrollo de software para una aplicación de reconocimiento de plagas en plantaciones de café que utiliza redes neuronales convolucionales (CNN) implica calcular el esfuerzo, tiempo y recursos necesarios para cada etapa del desarrollo, considerando el tipo de tecnología y los requerimientos específicos del proyecto.

4.4. COCOMO II

En la siguiente tabla se detalla el número de líneas de código desarrollada, se realizó el uso de librerías que reduce significativamente el esfuerzo necesario ya que muchos componentes complejos, como el procesamiento de imágenes y las redes neuronales convolucionales, están pre construidos y optimizados en las librerías.

Tabla 35: *Números de Línea de Código*

Lenguaje de programación	Líneas de programación
Python	582
Kotlin	2138
Java	2880

Total	**3582**

A continuación, aplicamos las fórmulas de esfuerzo, tiempo calendario y personal requerido para lo cual utilizaremos las siguientes formulas:

(23) Línea de Código

$$LDC = Python + Kotlin + Java \quad (23)$$

$$LDC = \text{ 582+1800+1200= 3582 líneas}$$

En Cocomo II, la métrica de tamaño se mide en miles de líneas de código (Kilo-Líneas of Code). Entonces para obtener KLDC, dividimos el total de líneas de código (LDC) entre 1000.

(24) Líneas de Código en Limites

$$KLDC = \frac{LDC}{1000} \quad (24)$$

Donde:

KLDC=?

LDC=3582 líneas

$$KLDC = 3582/1000 = 3{,}6$$

(25)Esfuerzo en Personas-Mes (ED)

$$ED = A(KLDC)^B \quad (25)$$

Donde:

ED =?

KLDC=Cantidad de líneas de código en limites

A = 2,4

B = 1,05

$$ED = 2{,}4 * (3{,}6)^{1{,}05} = 9{,}2\ personas - mes$$

(26) Tiempo de Desarrollo en Mes (TD)

$$TD = C(ED)^{D} \quad (26)$$

Donde:

TD=?

ED=Esfuerzo requerido por el proyecto.

C = 2,5

D=0,38

$$TD = 2{,}5 * (9{,}2)\text{^}0{,}38 = 5{,}81$$

El tiempo de desarrollo es aproximadamente 5,81 meses.

(27) Numero de Programadores Necesarios

$$NP = \frac{ED}{TD} \quad (27)$$

Donde:

NP = ?

ED = 9,2 personas-mes

TD = 5,81 meses

$$\mathrm{NP} = \frac{9{,}2}{5{,}81} = 1{,}58$$

Se necesitará aproximadamente 1,58 programadores para el desarrollo del proyecto.

(28) Costo de Software

$$CS = NP * Salario\ Mensual * TD \qquad (28)$$

Donde:

CS =?

NP =1,58 programadores

TD=5,81 meses

Salario mensual= 2500 bs.

$$CS = 1{,}58 * 2500 * 7 = 27.650\ bolivianos$$

Los modos de desarrollo de Cocomo II son tres categorías que representan diferentes niveles de complejidad y tamaño del proyecto de software. cada modo tiene valores específicos para los coeficientes A, B, C y C.

***Tabla 36**: Valores de Modelo Básico*

Modo	A	B	C	D
Orgánico	2,4	1,05	2,5	0,38
Semi-acoplado	2,94	1,12	2,5	0,35
Empotrado	3,6	1,20	2,5	0,32

4.5. PRUEBAS DEL SISTEMA

4.5.1 Pruebas de Caja Blanca

Las pruebas se basan en la inspección y evaluación del código para asegurarse de que los procesos internos del software estén funcionando correctamente.

En la aplicación móvil que se utiliza Redes Neuronales Convolucionales para detectar plagas en las plantaciones de café, se enfocó en los siguientes aspectos.

Tabla 37: *Pruebas de Inspección y Evaluación*

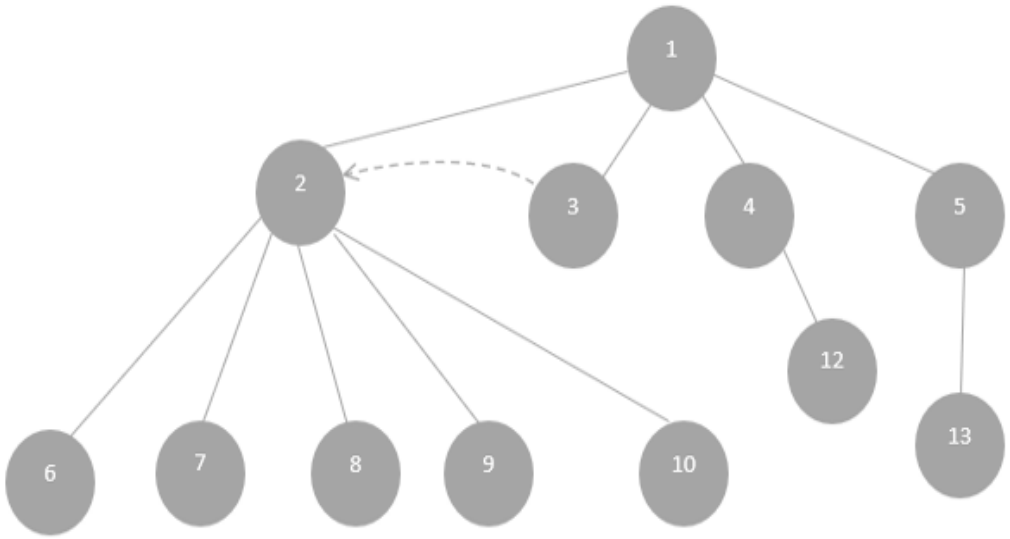
COBERTURA DE SENTENCIA	**AL CARGAR LA IMAGEN SE EJECUTA SOLO UNA VEZ**
Cobertura de decisiones	Se evalúa “plaga detectada” vs “plaga no detectada”
Cobertura de ciclos	Probar casos con diferentes números de imágenes.

Cobertura de caminos	Verificar las decisiones en el modelo y las rutas que pasan por diferentes decisiones.

4.5.1.1 Módulos Basados en Funcionalidades

Agrupa las funcionalidades en módulos que pueden ser desarrolladas y probados de forma independiente.

Figura 44:*Módulo de Funcionalidad*



- Inicio de sesión (1)
- Reconocimiento con cámara (2)
- Reconocimiento con imágenes (3)

- Manual de siembra de café (4)
- Información de la Asociación APEN (5)
- Módulo de carga de imágenes (6)
- Módulo de procesamiento (7)
- Módulo de modelo de CNN (8)
- Módulo de clasificación (9)
- Módulo de resultados (10)
- Módulo de interfaz de usuario (11)
- Vista del manual (12)
- Vista del estatuto de APEN (13)

Después de construir el grafo determinaremos su complejidad de siglo utilizando la siguiente formula:

(29) Complejidad Ciclo Matica

$$VG = A - N + 2 \quad (29)$$

Donde:

VG= La complejidad ciclo matica.

A=Número de aristas (flojos en el grafo de control)

N= Número de nodos

Donde:

A=12

N=13

$$VG = 12 - 13 + 2 = 1$$

La complejidad ciclo matica es 1 por lo que necesitamos 1 de prueba independientes para cubrir todos los caminos.

Camino 1: El usuario ingresa a la aplicación y selecciona reconocimiento con cámara, scanear el tipo de enfermedad ocasionado por la plaga.

Camino 2: El usuario ingresa a la aplicación y selecciona reconocimiento con imagen guardada scanear la hoja dañada.

Camino 3: El usuario inicia la aplicación y selecciona manual de siembra de café se informará el agricultor.

Camino 4: El usuario inicia a la aplicación y selecciona datos de la organización se dirige a la documentación de APEN.

Tabla 38: Descripción de Cada Módulo

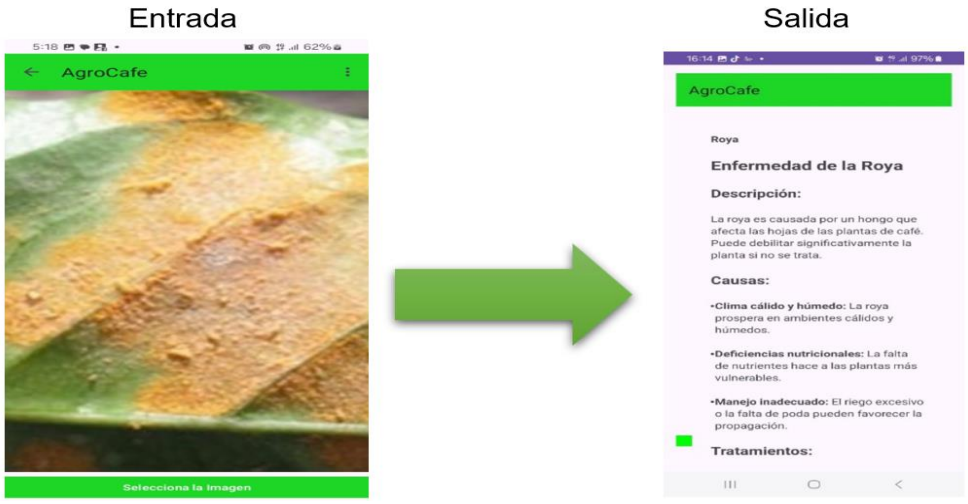
MODULO	DESCRIPCIÓN
Módulo de carga de imágenes	Maneja la carga de imágenes desde el dispositivo.
Módulo de procesamiento	Realiza el pre procesamiento de las imágenes (redimensionado, normalización, etc.).

Módulo de modelo de CNN	Contiene la implementación y carga del modelo de red neuronal convolucional. Realiza la clasificación de las imágenes usando el modelo de CNN.
Módulo de clasificación	Realiza la clasificación de las imágenes usando el modelo de CNN.
Módulo de resultados	Muestra los resultados de la clasificación al usuario.
Módulo de interfaz de usuario	Maneja el almacenamiento y recuperación de datos de usuario.
Módulo de gestión de datos	Diseña y gestiona la interfaz de usuario de la aplicación.

4.5.2. Prueba de Caja Negra

En la prueba se enfoca en probar la funcionalidad desde la perspectiva del usuario, sin necesidad de conocer el código interno con diferentes pruebas.

Figura 45: *Casos de Prueba Basado en Escenario de Usuario*



La aplicación identifica la enfermedad ocasionada por la plaga correctamente en la mayoría de las pruebas con imágenes de baja calidad con confianza de 62% muestra un nivel de certeza razonable, aunque puede ser inferior al de imágenes.

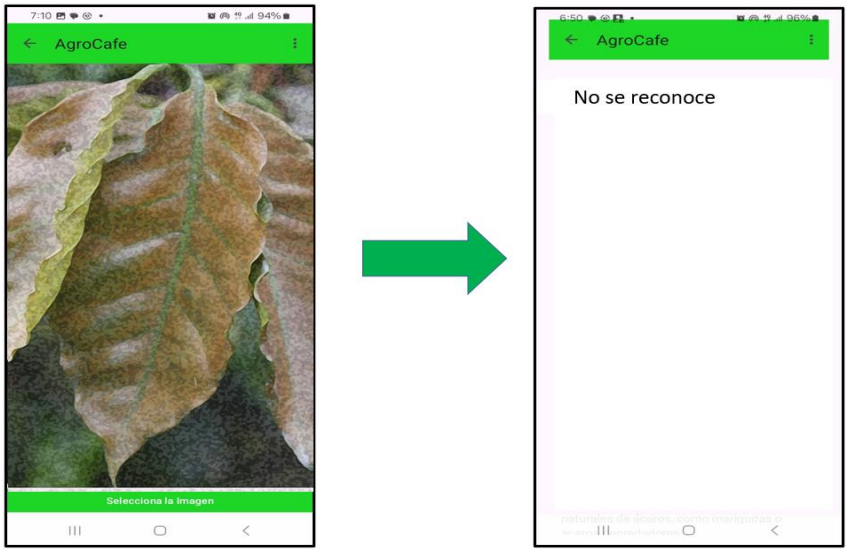
El resultado de las pruebas con imágenes de baja calidad depende de varios factores, como la capacidad del modelo de redes neuronales convolucionales (CNN) para manejar variaciones en las imágenes, la calidad de los datos de entrenamiento y la optimización del modelo para condiciones difíciles.

Variaciones en la Calidad de Imagen:

- Imagen desenfocada

  La confianza en la predicción es baja debido a la calidad de la imagen.

  Figura 46: *Imagen Desenfocada*



- Condiciones de baja iluminación

  La confianza en la predicción es baja debido a la iluminación de la imagen.

Figura 47: *Imagen de Baja Iluminación*

Según la variación de la calidad de imagen, el valor de confianza será de 0,4 o 40% el resultado será erróneo.

4.5.3. Pruebas de Estrés

Las pruebas de estrés se emplean para evaluar el rendimiento de la aplicación es una manera de entender cómo responder bajo carga intensa o condiciones extremas. Este tipo de pruebas ayuda a evaluar el rendimiento de la aplicación y la estabilidad del modelo de redes neuronales cuando se enfrenta a escenarios de alto uso o condiciones difíciles.

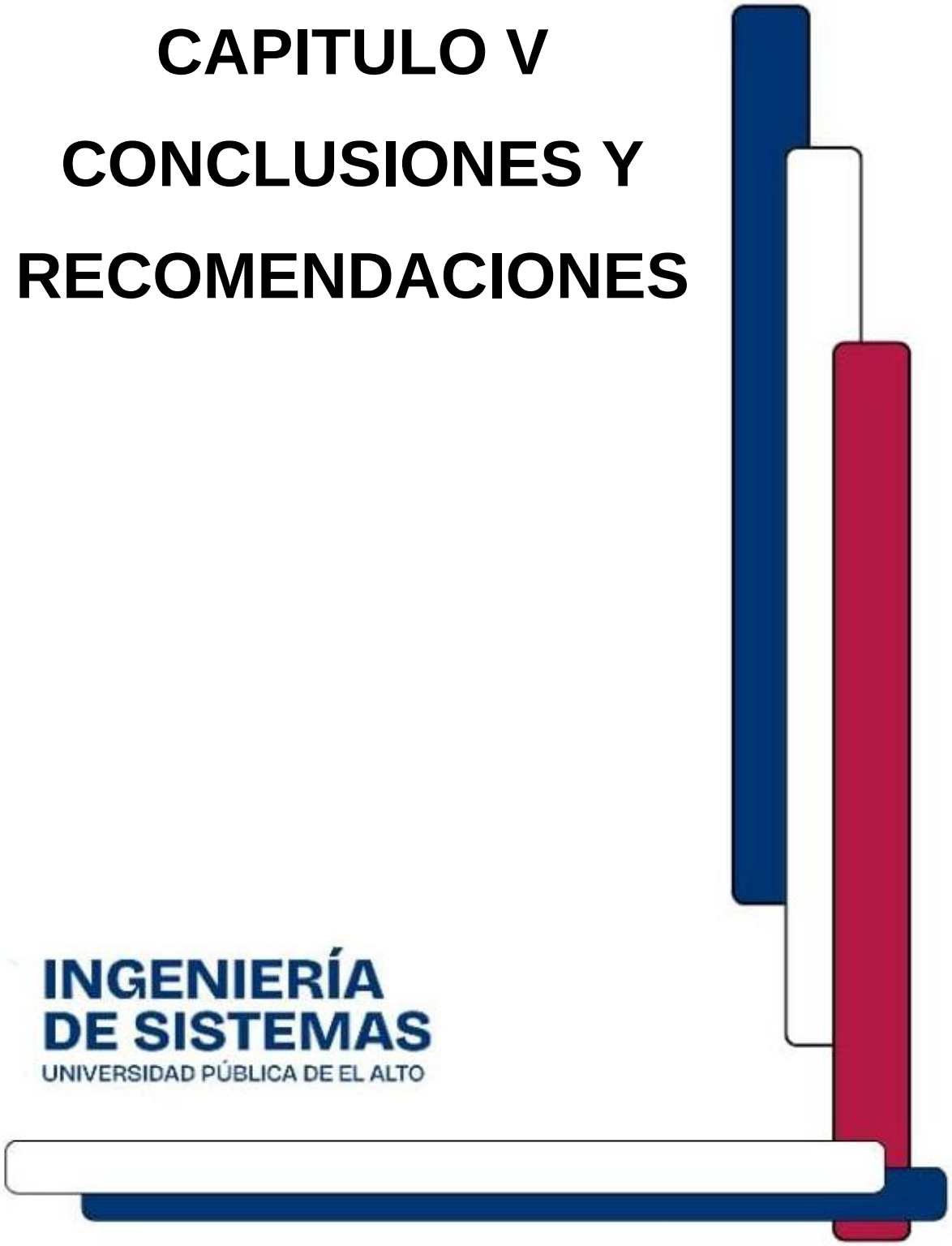
Los pasos para aplicar las pruebas, de la aplicación esté completamente instalada en el dispositivo móvil y configurada para procesar imágenes utilizando el modelo CNN.

Generación de gráficos y análisis de resultados.

Tabla 39: *Prueba de Estrés*

NUMERO DE IMÁGENES	TIEMPO DE RESPUESTA PROMEDIO	USO DE CPU (%)	USO DE MEMORIA (%)	BATERÍA CONSUMIDA
10	2 segundos	50%	60%	5%
50	6 segundos	70%	75%	15%
100	12 segundos	90%	85%	25%

CAPITULO V
CONCLUSIONES Y RECOMENDACIONES



CAPÍTULO V CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

Durante la investigación sobre las plagas más comunes dentro el Municipio la Asunta se detectó como arañita roja y roya de hoja, también se dio un paso fundamental fortalecer a los productores de café en el manejo de plagas, protegiendo la calidad y el rendimiento de sus cultivos y apoyando el desarrollo económico y ambiental.

El entrenamiento de imágenes se desarrolló con el modelo de red Neuronal Convolucional (CNN) con las herramientas de TensorFlow y Keras. La diversidad de las imágenes recopiladas, que incluyen variaciones de iluminación y estado de las plantas, contribuyó a mejorar la capacidad del modelo para reconocer plagas en diferentes condiciones.

La aplicación Agrocafe cumple las funciones de la detección en las hojas de café prediciendo la hoja sanas y no sanas. Las hojas no sanas de las plantas de café se clasifican en 2 clases: Roya de la hoja, arañita roja, que son estreses bióticos que ocurren comúnmente en las plantas de café, siendo evaluadas las plagas como arañita roja y roya de hoja, por la muestra aleatoriamente.

Las pruebas realizadas el rendimiento y precisión del modelo de la aplicación Agrocafe muestra la precisión de reconociendo en porcentaje para la seguridad del usuario y algunas recomendaciones marcadas por color verde.

5.2 RECOMENDACIONES

Para obtener la predicción de la enfermedad de la planta de café, el agricultor debe situarse en la planta y tomar fotos de las hojas de las plantas de café que se visualicen algún tipo de plaga.

Para hacer una predicción aún mejor y lograr una mayor precisión para una amplia variedad de plagas en las plantas de café, se requieren variables adicionales como la temperatura, la humedad del aire, variedades de plantas de café, el momento de la plantación y el clima de la ubicación.

La aplicación fue optimizada para dispositivos Android, garantizando un diseño accesible y funcional, adaptado a las condiciones del entorno agrícola. Además, se priorizó la simplicidad y claridad, permitiendo a los usuarios realizar la detección de plagas de forma rápida y precisa. La disposición de los elementos de la interfaz, tiene 4 listados de funciones que son la captura de imagen, selecciones de imagen de la galería guardada, manual de la producción de café para el campo y por el ultimo datos de la asociación (A.P.E.N.).

Como trabajo posterior se puede sugerir cambios necesarios en nuestra red neuronal para que realice un registro en los cultivos de café, en que épocas del año hay más posibilidades que puedan atacar las enfermedades.

Como trabajo fututo, pretendemos explorar los requisitos y cambios necesarios en nuestra red neuronal para poder acomodar otras nuevas enfermedades en las plantas de café y también tener en cuenta los factores ambientales variables.

Como un trabajo futuro desarrollar la Versión 2 de la Aplicación Móvil, utilizar roles, mejorar el modelo de Detección de Plagas con Red Neuronal Mejorada, utilizando modelo de redes neuronales más profundos o híbridos también llamadas sistemas multi -red que no solo detectan plagas si no también que identifiquen diferentes estados de infestación y los tipos de daños que causan.

BIBLIOGRAFIA

6. **BIBLIOGRAFIA**

(APEN), A. d. (2004). ESTATUTO ORGANICO.

Abu Mettleq, A. S.-N. (2019). *Un sistema basado en reglas para el diagnóstico de enfermedades del café.*

Akismet. (2024). Normas ISO. *Normas ISO 25000*.

Alberto Cardona López, J. A. (2019). *Diseño e implementación de un sistema de información para una.*

Almengor. (2019). *Recomendaciones de la época para el control de la Broca del Fruto del Cafeto - Hypothenemus hampei - y el Minador de la Hoja - Leucoptera coffeella.* Asociacion nacional del cafe.

Atom. (2016). *La oficina de proyectos de informatica.*

Caceres, A. (2024). *Scoreapps.* Madrid.

CANO, M. (2018). *Pruebas de cargo y estres*. Obtenido de https://www.paradigmadigital.com/dev/pruebas-carga-estres-que-son-cuando-usarlas/

Carrasco, C. S. (2022). *Detector de mentiras mediante el analisis de estres de voz usando redes neuronales recurrentes.* La Paz.

Casero, A. (2024). *Keepcoding.*

Chunnu, K., & Shah, P. (2018). *Aplicattion in Android App Development-A Study.*

Contreras, B. T. (2020). *Manual de cultivo de cafe.* La Paz.

Coral, L. M. (2012). Manejo integrado de plagas en el cultivo de cafe. En L. M. Coral. Peru.

Davis, A. P. (2015). *Guia de variedades de cafe.* Guatemala.

Derwin, S. W. (2021). Sistema experto para la detección de enfermedades en plantas de café. *Revista Universidad y Sociedad*, 162.

Diaz Holgado, R., & Villalba, V. (2021). *Aplicativo movil para la deteccion de la enfermedad del mosaico comun en la yuca.* Puerto Maldonado.

Duque, R. G. (2017). *PYTHON.*

El-Naqa & Murphy. (2015). What is Machine Learning? Springer International Publishing Switzerland.

Fidalgo, I. A. (2022). *Introduccion al Machine Learning con Tensorflow.* Santiago.

Flores Mendoza, J. C., & Carhuajulca, M. (2023). *aplicacion movil usando redes neuronales convulcionales para la deteccion de enfermedades en las plantas de cafe.* Peru.

Jose Emilio Labra Guayo, J. M. (2017). *Interpretes y diseño de lengaje de programacion.*

Kumar, M. G. (2020). *Disease Detection in Coffee Plants Using Convolutional Neural Network. International Conference on Communication and Electronics.*

Llano Carmon, M. d. (2021). *Artificial Neurotransmitter. dpto. Ciencias de la computacion e Inteligencia Artificial,1-12.*

Llano-Carmona. (2021). Artificial Neurotransmitter. . *dpto. Ciencias de la Computación e Inteligencia Artificial,*.

Mamani, P. P. (2018). Tippos de cafe.

Martines, J. C. (2018). *Mejorado con Inteligencia Artificial.*

Medina, M. C. (2017). Metodologia para el desarrollo de aplicaciones moviles.

Mogena-Soler. (2014). Informe de OpenCV y Tratamiento de Imágenes.

Parra-Plaza. (2015). Intérprete de lenguaje de signos para la comunicación de personas con discapacidad auditiva empleando procesamiento de imágenes. *Revista de Investigaciones - Universidad del Quindío*.

Rivero, L. F. (2014). *Clasificacion mediante redes neuronales y conglomerados no gerarquicos de las condiciones de vida de los hogares.* La Paz.

Software, G. (2021). *https://sg.com.mx/revista/34/estimaci%C3%B3n-costos*.

Sullcahuaman, A. (2021). *Visión Artificial.*

Thomas, t. M. (2024). *Zaptest.*

Ticona, B. S. (2020). *Biometria facial basada en redes neuronales aplicando el algoritmo de viola – Jones.* El Alto-Bolivia.

UNODC. (2024). La UNODC promueve la certificación de capacidades de caficultores en La Asunta. Obtenido de https://www.unodc.org/bolivia/es/La-UNODC-promueve-la-certificacion-de-capacidades-de-caficultores-en-La-Asunta.html

Vilet, M. (2005). Procesamiento Digital de Imágenes. *San Luis Potosí: Facultad de Ingeniería UASLP*.

ANEXOS

INGENIERÍA
DE SISTEMAS
UNIVERSIDAD PÚBLICA DE EL ALTO

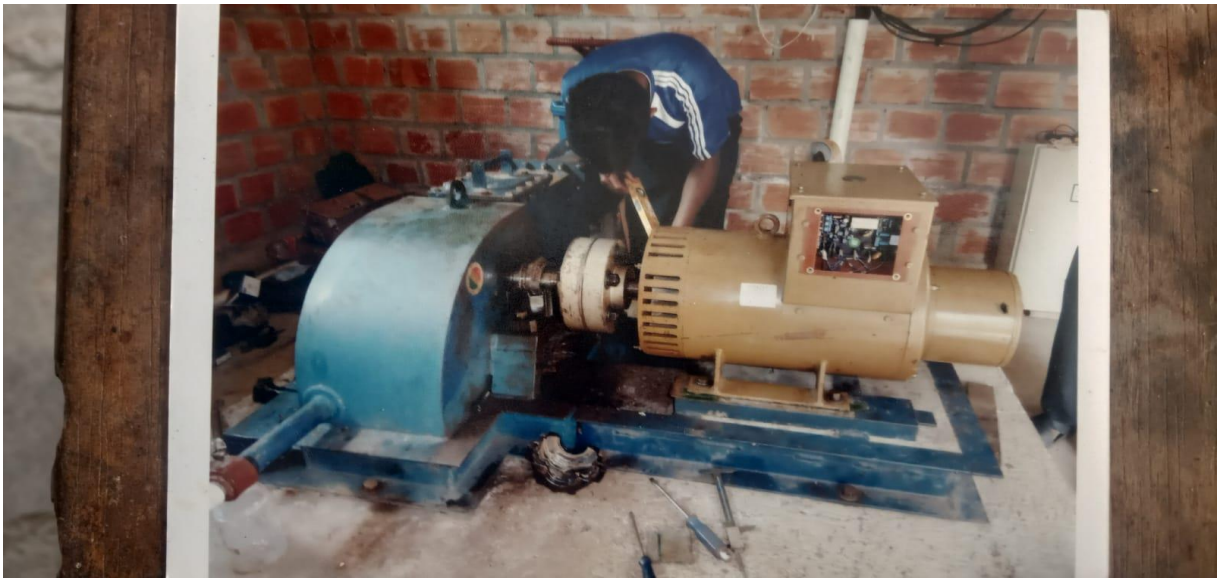
7.ANEXOS

ANEXO 1

ASOCIACION PRODUCTORES DE CAFÉ NOGALANI

ANEXO 2

PROCESAMIENTO DE PELADORA DE CAFÉ



ANEXO 3

ASOCIACION DE PRODUCTORES DE CAFÉ NOGALANI

ANEXO 4

VENTA DEL PRODUCTO DE CAFÉ PROCESADO

ANEXO 5

CULTIVO DE CAFE

ANEXO 6

ETAPA DE FLORECIMIENTO DEL CAFÉ

ANEXO 7

RECECADOR DE CAFE

MANUAL DE USUARIO

Aplicación móvil para la detección de plagas en plantas de café

(Versión 1.0)

1.Requisitos de la Aplicación

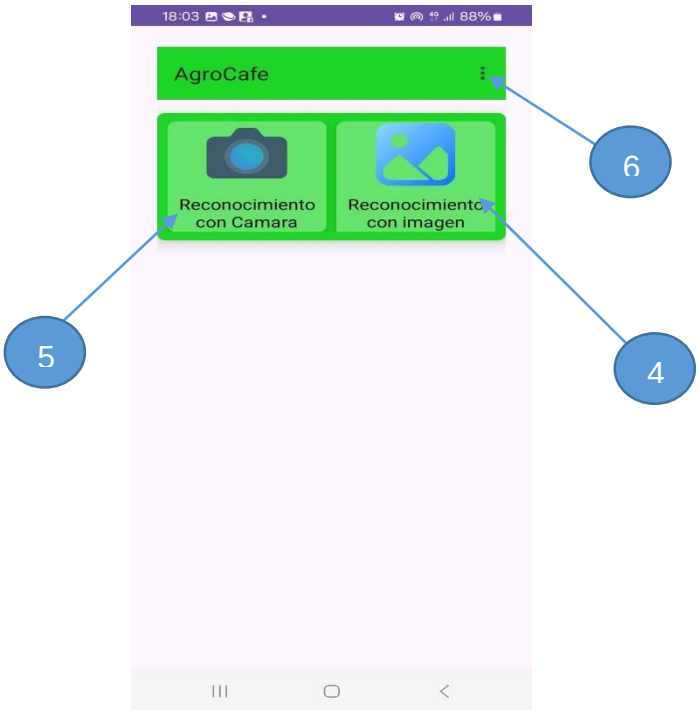
- ✓ Sistema operativo: Android
- ✓ Espacio de almacenamiento: Al menos 170 MB.
- ✓ Cámara del dispositivo: Buena resolución de cámara.
- ✓ Conexión a Internet: No es necesario.

2.Inicio de la Aplicación

En el inicio de cesión de la aplicación esperar unos según para el ingreso a la siguiente Pantalla.

3.Navegacion por la Interfaz

El usuario podrá navegar a diferentes opciones de la aplicación.

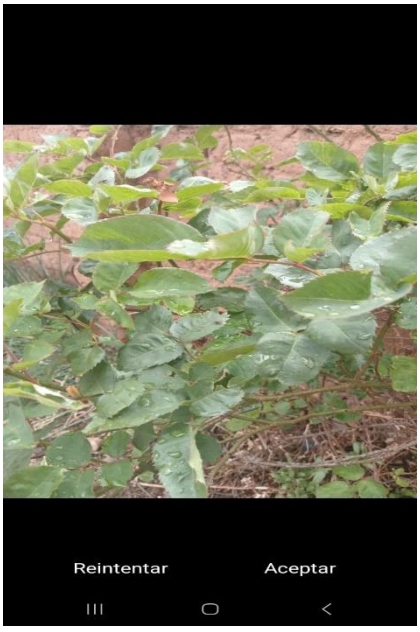


4.Uso de la Aplicación

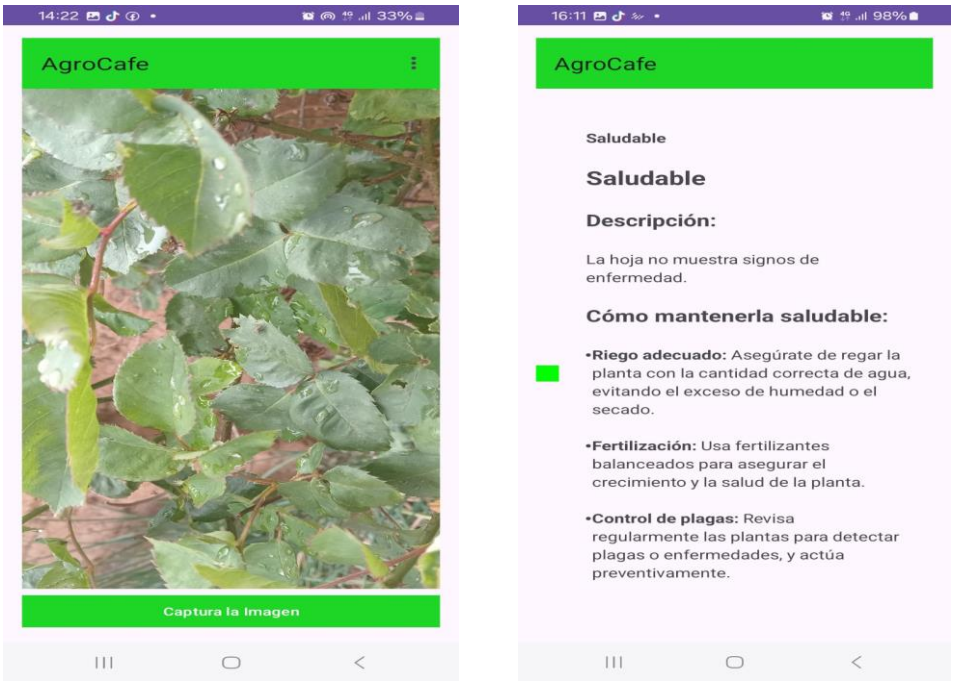
Ingresar a la opción de reconocimiento con imagen, se seleccionará una hoja de café con plaga y esperar 5 segundos para el análisis de la aplicación.

4.1 Análisis de la enfermedad de roya

5.Reconocimiento con Cámara



5.1 Análisis de reconocimiento con cámara.



6. La Opción de Información Generara Información de la Asociación APEN en PDF.