INSTITUTO TECNOLÓGICO SUPERIOR DE MISANTLA

INGENIERÍA EN SISTEMAS COMPUTACIONALES

DETECCIÓN DE PLAGAS Y ENFERMEDADES DEL LIMÓN PERSA MEDIANTE UNA APLICACIÓN MÓVIL CON APRENDIZAJE PROFUNDO A PARTIR DE REDES NEURONALES CONVOLUCIONALES

TESIS

QUE PRESENTA:

ALONSO HERNÁNDEZ MORA

MSC. IRAHAN OTONIEL JOSE GUZMAN

DIRECTOR

DR. SERGIO FABIAN RUIZ PAZ

DR. ROBERTO ANGEL MELENDEZ ARMENTA

CODIRECTOR

MISANTLA, VERACRUZ DICIEMBRE, 2021

Dedicatoria

Dedico este trabajo principalmente a Dios, que me dio la oportunidad de llegar hasta este momento tan importante de mi formación profesional.

A mi padre Alejandrino, a mi madre Celestina quienes con su amor, paciencia y esfuerzo me han permitido llegar a cumplir hoy un sueño más, gracias por inculcar en mí el ejemplo de esfuerzo y valentía, de no temer las adversidades porque Dios está conmigo siempre.

A mi esposa y mi hija por el amor que me han dado hasta el momento y por todo el apoyo para lograrlo.

Agradecimientos

Agradezco al Instituto Tecnológico Superior de Misantla, por haberme brindado tantas oportunidades y enriquecerme en conocimiento.

De igual forma agradezco al maestro Irahan Otoniel José Guzmán, por haberme guiado, no solo en la elaboración de este proyecto, sino a lo largo de mi carrera universitaria y haberme brindado el apoyo para desarrollarme profesionalmente y seguir cultivando mis valores.

Gracias DIOS mi fortaleza, mi escudo por siempre estar en esos momentos de dificultades, cuando todo parecía perdido me tendiste tu mano y me guardaste de todos los peligros que existen en el camino, sabes que todo mi agradecimiento para ti es sincero.

Resumen

Ante la creciente demanda de limón por parte de Estados Unidos, ha ido en aumento el área cultivada de este fruto y cada vez más agricultores se unen la producción de este, sin embargo, ante la falta de información referente al control de plagas y enfermedades existentes en este tipo de cultivo los agricultores normalmente se enfrentan a diversas problemáticas al iniciar y mantener la siembra de este tipo de fruto.

En el siguiente trabajo se presentan los resultados obtenidos de las actividades del desarrollo de una aplicación móvil Android, capaz de realizar la detección de plagas y enfermedades en el limón persa, utilizando técnicas de aprendizaje profundo y redes neuronales convolucionales.

Para el desarrollo de la aplicación móvil, se utilizó el IDE Android Studio y el lenguaje de programación Kotlin. Además, la red neuronal está desarrollada con Python y utiliza la librería de Tensorflow para poder ser utilizada en el sistema operativo Android utilizando la librería Tensorflow Lite.

El propósito del desarrollo de la aplicación no es solo realizar la detección de plagas y enfermedades en el limón persa, si no también proporcionar información relevante sobre estas, a los productores de limón persa del país.

El resultado obtenido de este trabajo es una aplicación móvil Android que es capaz de realizar la detección de plagas y enfermedades en el limón persa utilizando aprendizaje profundo mediante redes neuronales convolucionales. Tras su desarrollo la aplicación fue probada en el campo real y aunque cumple con su tarea, el margen de error de las predicciones es considerado alto, ya que es del 15% aproximadamente, esto se debe principalmente a la falta de datos para el entrenamiento del modelo de aprendizaje profundo.

Índice

Índice de figuras

Índice de tablas

Capitulo 1. Antecedentes

1.1 Introducción

México es actualmente uno de los mayores productores de limón en el mundo con alrededor de 209, 129 ha plantadas, de las cuales el 50% de estas es de limón persa y esta cifra va en aumento ante la necesidad de satisfacer la demanda nacional como internacional. (USDA, 2020)

En el cultivo de limón persa se presentan diversos problemas, pero uno de los principales es la correcta detección de plagas y enfermedades y el buen control de estas. A pesar de que la inteligencia artificial (IA) se ha utilizado en cultivos como el aguacate para mejorar su producción y calidad, en cuanto al limón persa la situación es distinta ya que, en el estado de Veracruz, el principal productor de limón persa del país, se ha realizado poca inversión en tecnologías que implementen el uso de IA como apoyo a los citricultores para lograr tener un mejor control de plagas y enfermedades de este cultivo.

Hoy en día es necesario desarrollar herramientas que apoyen a los citricultores a realizar distintas tareas que beneficien la producción de limón persa del país, en este caso que beneficien al realizar una correcta detección de plagas y enfermedades en el limón persa, para lograr tener un mejor control sobre estas y obtener mejor calidad de frutos. La importancia de una buena detección de las plagas y enfermedades en el cultivo de limón persa es de vital importancia, ya que se asegura el buen control de estas, evitando que los productores recurran al uso de productos que ya no se encuentran permitidos para este tipo de cultivos, debido a los residuos tóxicos peligrosos para el ser humano.

En este trabajo se demuestra que mediante el uso de tecnologías móviles se puede brindar un mayor apoyo a los productores de limón persa del país, ofreciendo una herramienta capaz de realizar una correcta detección de plagas y enfermedades y ofreciendo información relevante acerca de estas.

El documento está constituido por 4 capítulos:

- Capítulo 1. En este capítulo se plasma el planteamiento del problema, los antecedentes de este, la delimitación y los limitantes del problema, etc.
- Capítulo 2. En este capítulo se describen los fundamentos teóricos de la investigación.
- Capítulo 3. En este capítulo se describe el procedimiento de las actividades realizadas para el desarrollo de la aplicación móvil.
- Capítulo 4. En este capítulo se muestran los resultados obtenidos al final del proyecto y se explican las conclusiones a las que se ha llegado.

1.2 Planteamiento del problema

En el área de cultivo de limón persa es muy importante que los productores tengan un buen control sobre las plagas y enfermedades, esto para obtener mejores beneficios al ofrecer frutos de buena calidad para el mercado nacional e internacional.

Con el aumento en la demanda de este fruto, el número de productores va en incremento, sin embargo, es común que cuando un productor recién inicia en el cultivo, no tiene el suficiente conocimiento para lograr identificar las plagas y enfermedades propiciando que éste no tome las medidas adecuadas por lo tanto se generen daños mayores y no tome las acciones adecuadas tales como:

- Utilizar productos que no están permitidos para el control de plagas o enfermedades en el limón persa.
- Aplicar productos no efectivos sobre la plaga o enfermedad.
- No aplicar ningún producto, provocando que la plaga o enfermedad se extienda por la huerta o a otras adyacentes.

La aplicación móvil propuesta para solucionar esta problemática está enfocada en realizar una correcta detección de plagas y enfermedades en el limón persa, esto con el fin de ayudar

a los productores a tener un mejor control sobre estas. Con esto los productores estarán mejor informados y podrán ofrecer mejores frutos, sobre todo para el mercado internacional que es el que mejor calidad exige.

1.3 Justificación

Según el Departamento de Agricultura de los Estados Unidos (USDA, por sus siglas en inglés) México es el segundo mayor productor de Limón Persa en el mundo y cuenta con un poco más de 209, 000 hectáreas plantadas.

Ante la creciente demanda de este cítrico, principalmente por parte de Estados Unidos y algunos países del continente europeo, el área de cultivo ha ido aumentando tal como se muestra en la figura 1.1 ocasionando que muchos agricultores opten por cultivar limón persa, sin embargo, muchos de ellos no cuentan con la suficiente información respecto a las enfermedades y plagas que este tipo de cultivos pueden llegar a presentar, ocasionando que recurran al uso de pesticidas que se encuentran prohibidos en el cultivo de limón persa.

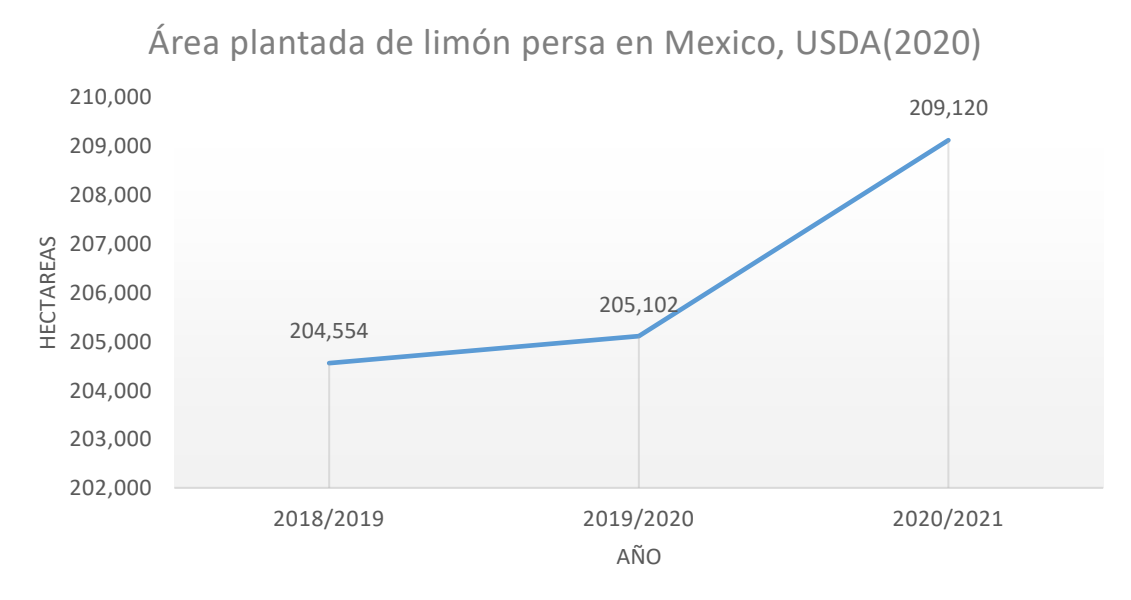

Figura 1.1 Aumento del área plantada de limón persa en México

A pesar de que México es el segundo mayor productor de limón persa y su cultivo dentro del país tiene una antigüedad de aproximadamente 50 años, existe poca información sobre el uso de la tecnología móvil en esta área.

Ante lo mencionado anteriormente surge la idea de realizar una aplicación para dispositivos móviles con sistema operativo Android, que ayude a los agricultores de México a realizar una correcta detección de plagas y enfermedades en el cultivo de limón persa y ofrezca información relevante sobre estas y sobre los plaguicidas que no se encuentran permitidos para uso en este tipo de cultivos.

1.4 Objetivos

1.4.1 Objetivo general

Desarrollar una aplicación móvil Android de detección de plagas y enfermedades en el limón persa de México empleando aprendizaje profundo a partir de redes neuronales, con el fin de facilitar a los citricultores esta labor.

1.4.2 Objetivos específicos

- Recopilar información acerca de plagas y enfermedades del limón persa.
- Crear un conjunto de datos de imágenes de plagas y enfermedades del limón persa para entrenar y validar un modelo de clasificación de imágenes.
- Desarrollar un modelo de clasificación de imágenes con Python y Tensorflow que realice la predicción de plagas y enfermedades en el limón persa.
- Crear una base de datos NoSQL para almacenar la información sobre plagas y enfermedades del limón persa.
- Desarrollar una API REST que permita realizar la conexión entre la aplicación móvil y la base de datos.

1.5 Hipótesis

La aplicación de detección de plagas y enfermedades del limón persa apoya a los citricultores mexicanos a realizar una correcta detección y control de estas.

1.6 Delimitación del proyecto

1.6.1 Delimitación del espacio

La aplicación de detección de plagas y enfermedades del limón persa está desarrollada para apoyar a los productores de limón persa de México.

1.6.2 Delimitación del tiempo

La información sobre plagas y enfermedades del limón persa utilizada durante el desarrollo del proyecto está basada en los últimos 5 años.

1.7 Limites

- La falta de imágenes sobre plagas y enfermedades del limón persa, para el entrenamiento del modelo de aprendizaje profundo, ocasiona que el margen de error en las predicciones sea de un 10% aun después de aplicar técnicas para solucionar el overfitting.
- Las imágenes capturadas con la aplicación deberán ser tomadas en un ambiente en el que no exista exceso de luz que pueda generar reflejos en la fruta, ya que esto podría alterar la predicción del modelo de clasificación.
- El modelo de clasificación solo detecta 4 tipos de plagas en los frutos del limón persa.
- La información sobre plagas y enfermedades estudiadas en el proyecto son solo algunas de las más conocidas en México.
- La aplicación está disponible solo para dispositivos que ejecuten el sistema operativo Android en una versión igual o superior a la 5.0 (Lollipop).

Capitulo 2. Estado del arte

Durante el desarrollo de este proyecto se observaros distintos sistemas que hacen uso de técnicas de Deep Learning para la detección de plagas y enfermedades que se manifiestan en diferentes cultivos agrícolas, entre ellos, el limón.

2.1 Trabajos relacionados

2.1.1 Enfocados a la comparación de algoritmos

Balzhoyt Roldan et al. (2019) en su artículo titulado *Detección de enfermedades en el sector agrícola utilizando Inteligencia Artificial* exponen varios trabajos de investigación en donde se utilizan técnicas de inteligencia artificial para la posible detección de enfermedades en cultivos agrícolas a través de la obtención de características de las hojas y los frutos de la planta.

Balzhoyt Roldan et al. (2019) exponen que se utilizaron cámaras digitales para capturar imágenes de las hojas o las partes donde es visible el daño causado por la enfermedad, así como un conjunto de imágenes en la web, estas sirvieron de base para el entrenamiento del modelo. Los algoritmos elegidos como clasificadores fueron Fuzzy logic, SVM, Bayes, KNN, ANN, CNN.

También se realizó la comparación de la precisión obtenida de cada uno de los algoritmos en distintos cultivos según cada autor, llegando a la conclusión de que, en caso de contar con los datos suficientes, las redes neuronales convolucionales obtendrían mejores resultados que otros algoritmos, en caso contrario, lo mejor sería utilizar arboles de decisión con lógica difusa, ya que es el algoritmo con mejores resultados después de las redes neuronales convolucionales.

Jefferson Aimacaña y Alexander Columba (2021) realizaron un análisis comparativo de distintos algoritmos de Machine Learning para la detección de plagas en distintos cultivos,

entre los cuales los más destacados son la regresión logística, las redes neuronales convolucionales, los bosques aleatorios y SVM.

Durante su trabajo de investigación, Aimacaña y Columba entrenan cada uno de los algoritmos con distintos tipos de cultivos (papa, maíz, tomate y manzana) y enfermedades. Al final ponen a prueba cada algoritmo y obtienen que con la regresión logística se obtiene un 79% de precisión, con los bosques aleatorios un 82%, con SVM un 78% y con las redes neuronales convolucionales un 97%, demostrando que estas últimas son el mejor clasificador, aunque consuman más recursos computacionalmente.

2.1.2 Enfocados a la detección de plagas en cultivos distintos al limón

Nilda Espinoza Villafuerte (2019) realizo una investigación sobre el diagnostico automático de café trillado con broca mediante el procesamiento de imágenes con técnicas de Deep Learning.

Durante su investigación Espinoza uso la librería OpenCV, Scikit-Learn, Keras y el lenguaje Python, para la creación de los modelos utilizados en la clasificación del daño ocasionado por la broca del café.

Espinoza entreno dos modelos clasificadores, una Maquina de Vectores de Soporte (SVM) y una Red Neuronal Convolucional (CNN), ambas entrenadas con 6000 imágenes con una resolución de 480x640. Al final ambos modelos fueron probados con 50 imágenes, demostrando que el modelo SVM es mejor clasificando, para este caso, con un 96% de precisión, ya que el modelo CNN solo obtuvo un 90%.

Mauro Ezequiel Pereyra (2020) realizo una investigación sobre la detección de plagas y enfermedades en cultivos mediante Machine Learning y propuso una solución basada en la creación de una aplicación móvil que hace uso de un modelo de detección de objetos en tiempo real a través de la cámara de un dispositivo móvil.

El entrenamiento del modelo de clasificación (CNN) se realizó con ayuda del lenguaje de programación Python y la librería de Tensorflow. Para mejorar la precisión del modelo se

utilizó aprendizaje por transferencia con ayuda de un modelo previamente entrenado del repositorio de Tensorflow. El modelo resultante fue probado en una aplicación móvil Android en donde demostró buena precisión a la hora de clasificar las plagas en las hojas de tomate con hasta un 95%.

2.1.3 Enfocados a la detección de plagas en el limón

Javier Berger et al. (2019) en su artículo titulado Identificación de síntomas de Huanglonbing en hojas de cítricos mediante técnicas de Deep Learning expone el desarrollo de una aplicación móvil que utiliza técnicas de Deep Learning para la identificación de síntomas de Huanglonbing (HLB) y carencias nutricionales en hojas de árboles cítricos.

Para el entrenamiento del modelo de clasificación se utilizó un conjunto de datos que contenía de 1200 imágenes de 640x480 pixeles y este contenía cuatro categorías, asintomáticas, carencia de magnesio, carencia de zinc y HLB.

Berger utilizo aprendizaje por transferencia, para crear su modelo de clasificación, con ayuda de la librería Tensorflow y el lenguaje de programación Python. Utilizo dos modelos previamente entrenados, Inception V3 y MobileNet V0.5, y comparo los resultados de cada uno.

La aplicación que utilizaron para probar los modelos entrenados fue TFmobile, un proyecto de Android Studio provisto por Tensorflow, dando como resultado una APK para el modelo Inception y otro para MobileNet. El resultado que obtuvieron fue un 90.75% de predicciones correctas para el modelo MobileNet y un 90.5% para Inception.

Con los resultados que obtuvo Berger, llega a la conclusión de que con un dispositivo móvil dotado de una aplicación que utilice métodos de Deep Learning se pueden reconocer síntomas de HLB y otras carencias nutricionales en el limón con hasta un 90% de precisión que, si bien no es una precisión alta, se puede mejorar al agregar más imágenes de hojas en distintas posiciones y con distintos fondos, para lograr una mejor generalización de cada categoría clasificada.

José Luis Carranza Flores (2020) propone un sistema basado en técnicas de visión artificial que realice la detección automática del daño causado por el minador de los cítricos en la hoja de limón. Este sistema fue realizado con ayuda de la herramienta MATLAB y el algoritmo KNN. Su proceso de clasificación consta de dos pasos previos antes de pasar las imágenes al algoritmo, la conversión de sistema de color RGB a HLS y la segmentación. Cabe resaltar que los resultados que obtuvo el sistema propuesto por Carranza fueron bastante buenos, logrando un 98% de precisión en el reconocimiento de los daños causados por el minador de los cítricos.

2.2 Comparación de trabajos

Tabla 2.1 Trabajos relacionados

Autor	Título y año	Algoritmo	Resultados
Nilda Espinoza	Diagnóstico automatizado para la clasificación de café trillado con broca mediante procesamiento de imágenes con Deep Learning	CNN y SMV	Creo dos modelos para diagnosticar el café trillado con broca, observando que en este caso la Maquina de Vectores de Soporte obtuvo mejores resultados, 96% de precisión, que la red neuronal convolucional que solo obtuvo un 90%.
Javier Berger et al.	Identificación de síntomas de Huanglongbing en hojas de cítricos mediante técnicas de Deep Learning	CNN	Realizo la comparación de dos modelos de Tensorflow (MobileNet e Inception) utilizados en aprendizaje por transferencia, logrando obtener hasta un 90% de precisión con ambos modelos y optando por el modelo MobileNet debido a

			su menor tamaño y mayor velocidad a la hora de clasificar.
Mauro Ezequiel	Detección de enfermedades y plagas en cultivos mediante Machine Learning	CNN	Creo una aplicación móvil que hace uso de una modelo de detección de plagas y enfermedades en las hojas de tomate, alcanzando hasta un 90% de precisión.
José L. Carranza	Detección automática del daño causado por el Minador de los cítricos en la hoja de limón Mexicano	KNN	Tras su investigación obtuvo un modelo que realiza la detección del daño causado por el minador de los cítricos con hasta un 98% de precisión.
Alonso Hernandez	Detección de plagas y enfermedades del limón persa mediante una aplicación móvil con aprendizaje profundo a partir de redes neuronales convolucionales	CNN	Tras su investigación creo un modelo de detección de plagas y enfermedades en el limón persa con hasta un 85% de precisión, el cual se podría mejorar al obtener más imágenes de cada plaga.
Jefferson Aimacaña y Alexander Columba	Análisis comparativo de algoritmos de Machine Learning para la detección de plagas en los cultivos representativos de la sierra ecuatoriana	Regresión logística, CNN, SVM y Random Forest	Tras comparar los cuatro algoritmos en distintos cultivos y con distintas plagas, lograron observar que en todos los casos las redes neuronales convolucionales superaron a los demás algoritmos alcanzando un 97% de precisión promedio a diferencia de los demás que

			solo alcanzaron un 80% en promedio.
Balzhoyt Roldan et al.	Detección de enfermedades en el sector agrícola utilizando Inteligencia Artificial	Fuzzy logic, SVM, Bayes, KNN, ANN, CNN	En su investigación observaron el desempeño de cada uno de los algoritmos en distintos cultivos, llegando a la conclusión de que las redes neuronales convolucionales son la mejor opción a la hora de clasificar enfermedades siempre y cuando se tenga la cantidad de datos necesaria y el equipo de cómputo, en caso contrario lo mejor sería utilizar arboles de decisión con lógica difusa, aunque no apliquen técnicas de visión artificial.

2.3 Conclusión

Como se pudo observar en los trabajos presentados, el uso de las técnicas de Deep Learning es ampliamente usado en la detección de plagas y enfermedades de distintos cultivos. También se puede notar que las redes neuronales convolucionales son un algoritmo que puede ofrecer una precisión bastante buena para este caso, sin embargo, el hecho de entrenarla resulta muy costoso computacionalmente y se requiere una gran cantidad de datos y tiempo para hacerlo, aun así, no deja de ser uno de los algoritmos más utilizados y con una predicción más cercana a la de un experto humano.

Capitulo 3. Marco teórico

3.1 Metodología

3.1.1 Metodología de desarrollo móvil

La metodología propuesta para el desarrollo de 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 6M, la ingeniería de software educativo con modelado orientado por objetos (ISE-OO), y principalmente en los valores de las metodologías agiles. (Gasca *et al*, 2014)

La metodología se encuentra enmarcada en cinco fases, denominadas:

- Análisis. En esta fase se analizan los requerimientos de las personas o entidad para la cual se desarrolla el servicio móvil "Cliente".
- Diseño. En esta fase se plasma el pensamiento de la solución mediante diagramas o esquemas.
- Desarrollo. En esta fase se implementa el diseño en un producto de software.
- Pruebas de funcionamiento. En esta fase se verifica el funcionamiento de la aplicación en diferentes escenarios y condiciones.
- Entrega. En esta fase se realiza la entrega del ejecutable, el código fuente, la documentación y el manual del sistema, siempre y cuando la depuración haya terminado y se hayan atendido todos los requerimientos de última hora del cliente.

3.1.2 Metodologías de desarrollo ágil

Son aquellas metodologías que tienen la capacidad de adaptar las formas de trabajo a las condiciones del proyecto, consiguiendo inmediatez y flexibilidad.

Este tipo de metodologías busca proporcionar en un corto tiempo piezas pequeñas de sistemas de software en función para mejorar la satisfacción del cliente. (RedHat, 2020)

A continuación, se muestran las metodologías agiles más usadas:

- Scrum. Está basada en una estructura de desarrollo incremental. Esta metodología sigue la estrategia de desarrollo incremental de los marcos de desarrollo ágiles.
- Extreme Programming XP. Su principal objetivo es apoyar las relaciones entre empleados y clientes, por lo que es una herramienta útil para startups o empresas en proceso de consolidación.

  Las fases que atraviesa son:

  - Planificación del proyecto.
  - Diseño del proyecto.
  - Codificación.
  - Pruebas para comprobar el funcionamiento de los códigos.
- Kanban. Conocida como Tarjeta Visual. Consiste en elaborar un diagrama de 3 columnas de tareas pendientes, en proceso o finalizadas. Debe estar disponible para todos los miembros del equipo de trabajo. (Toledo, 2020)

3.1.3 Metodologías de desarrollo tradicional

Como su nombre lo indica, son aquellas metodologías que se han usado toda la vida. Buscan que el desarrollo de software sea predecible y eficiente al imponer disciplina a su proceso.

Estas metodologías están caracterizadas por definir total y rígidamente los requisitos al inicio de los proyectos de ingeniería de software. Al contrario de las metodologías agiles, estas son poco flexibles y no permiten realizar cambios.

Las principales metodologías tradicionales son:

- Waterfall. Sus etapas se organizan de arriba abajo. Las funciones que se desarrollan obedecen un riguroso orden.

- Prototipado. se basa en la construcción de un prototipo de software que se construye rápidamente para que los usuarios puedan probarlo y aportar feedback.
- Espiral. Es una combinación de Waterfall y Prototipado, esta metodología añade el concepto de análisis de riesgo.

  Tiene 4 etapas:

  - Planificación.
  - Análisis de riesgo.
  - Desarrollo del prototipo.
  - Evaluación del cliente.
- Incremental. Se construye el producto final de manera progresiva. Cada etapa incremental agrega una nueva funcionalidad.
- Diseño rápido de aplicaciones (RAD). Permite desarrollar software de alta calidad en poco tiempo. Tiene costos más altos y desarrollo más flexible.

3.1.4 Ágil vs Tradicional

En la siguiente tabla 3.1 se muestran algunas diferencias entre las metodologías agiles y tradicionales.

Tabla 3.3.1 Metodología ágil vs tradicional

Metodología ágil	Metodología tradicional
Orientada a pequeños proyectos	Orientada a proyectos de cualquier tamaño
Equipos pequeños (<10 personas)	Efectivas con equipos grandes o dispersos
Orientada a proyectos de poca duración	Orientada a proyectos de cualquier duración
Flexibilidad en el contrato	Contrato prefijado
Se esperan cambios en el proyecto	No se esperan cambios importantes en el proyecto
Poca documentación	Mucha documentación

3.2 Deep Learning

El Deep Learning (Aprendizaje profundo) es uno de los métodos de aprendizaje de la inteligencia artificial (IA) y pertenece a un subcampo del Machine Learning. Con este se puede entrenar a una computadora para que realice tareas como las que realizan los seres humanos, como predicciones, identificación de imágenes, etc. (Alonso, 2020)

El Deep Learning destaca por no requerir de reglas programadas previamente, ya que le mismo sistema es capaz de aprender por su cuenta para efectuar una tarea a través de una fase previa de entrenamiento.

También se caracteriza por estar compuesto de redes neuronales artificiales entrelazadas para el procesamiento de información.

3.2.1 Redes neuronales

Una red neuronal es un modelo que emula el modo en que el cerebro humano procesa información.

Una red neuronal está compuesta normalmente por tres capas:

- Capa de entrada (Input). Está compuesta por neuronas que asimilan los datos de entrada.
- Capa oculta (Hidden). Es la red que realiza el procesamiento de información y hacen los cálculos intermedios. Puede haber una o varias.
- Capa de salida (Output). Es la red que toma la decisión o realiza alguna conclusión aportando datos de salida.

La red neuronal aprende examinando los registros individuales, generando una predicción para cada registro y realizando ajustes cuando realiza una predicción incorrecta. Este proceso puede iterar muchas veces y la red sigue mejorando sus predicciones hasta haber alcanzados los criterios de parada. (IBM, 2020)

3.2.2 Redes neuronales convolucionales

Una red neuronal convolucional (CNN) es una arquitectura de red para aprendizaje profundo que aprende directamente de los datos, sin la necesidad de extraer características manualmente.

Las CNN son particularmente útiles para encontrar patrones en imágenes para reconocer objetos, caras y escenas. También son bastante eficaces para clasificar datos sin imágenes, como audio, series temporales y señales.

Al igual que otras redes neuronales, una CNN está compuesta por una capa de entrada, una capa de salida y muchas capas intermedias ocultas. Además de estas tres capas las CNN también incluyen dos capas extra que son la capa de convolución y la capa de pooling. (MathWorks, 2021)

La capa convolucional aplica distintos filtros sobre una imagen de entrada y crea nuevos volúmenes.

La capa de pooling realiza la reducción de las representaciones obtenidas de manera que estas se vuelvan más pequeñas y manejables computacionalmente, reduciendo así el número de parámetros necesarios. (Parada, 2021)

3.2.3 Aplicaciones del Deep Learning

El Deep Learning tiene especial aplicación en el área de medicina y en el mercado financiero, pero poco a poco otros sectores se unen diseñando sistemas de Deep Learning.

Los sistemas de Deep Learning se utilizan en:

- Traductores inteligentes.
- Lenguaje natural hablado y escrito.
- Reconocimiento de voz.
- Interpretación semántica.
- Reconocimiento facial.

- Visión computacional. (SmartPanel, 2018)

3.3 Android

Android es un sistema operativo basado en Linux y otros softwares de código abierto. Fue diseñado para dispositivos móviles con pantalla táctil, es por eso que se le puede encontrar en teléfonos inteligentes o tabletas, aunque también es usado en otros dispositivos como relojes inteligentes, televisores e incluso en los sistemas multimedia de algunos modelos de coches. (Adeva, 2021)

Figura 3.1 Logo del sistema operativo Android

La figura 3.1 muestra el logo actual del sistema operativo Android.

3.3.1 Historia

El sistema operativo Android fue creado por la compañía Android Inc., la cual fue fundada en el año 2003 por Andry Rubin, Rich Miner, Nick Sears y Chris White. En un principio Android fue anunciado como un sistema operativo para las cámaras y que permitía conectarlas con el PC sin necesidad de utilizar cables.

Sin embargo, el uso de Android en cámaras no era un tema con mucho futuro, por lo que decantaron por los teléfonos móviles. En el año 2005, Google se interesa por esta compañía y decide comprarla e incorporar a sus fundadores a las filas de la compañía.

Dos años después, el 5 de noviembre de 2007, se anunció la primera versión del sistema operativo Android, llamada Apple Pie. Sin embargo, los primeros dispositivos con esta versión comenzaron a aparecer hasta el año 2008.

Desde entonces y durante estos últimos años, el desarrollo de Android no ha parado y han sido muchas las versiones lanzadas del sistema con mejoras a nivel de rendimiento y seguridad.

3.3.2 Versiones y actualizaciones

Desde su lanzamiento hasta la actualidad Android ha recibido un gran número de actualizaciones como se muestra en la tabla 2.2. Hasta la versión 9 estas actualizaciones han recibido el nombre de un postre o dulce e iban por orden alfabético.

Tabla 3.3.2 Versiones y actualizaciones de Android

Nombre de código	Versión	Fecha de lanzamiento	API
Android Apple Pie	1.0	23 de septiembre de 2008	1
Android Banana Bread	1.1	9 de febrero de 2009	2
Android Cupcake	1.5	25 de abril de 2009	3
Android Donut	1.6	15 de septiembre de 2009	4
Android Eclair	2.0 – 2.1	26 de octubre de 2009	5 - 7
Android Froyo	2.2 – 2.2.3	20 de mayo de 2010	8
Android Gingerbread	2.3 – 2.3.7	6 de diciembre de 2010	9 – 10
Android Honeycomb	3.0 - 3.2.6	22 de febrero de 2011	11 – 13
Android Ice Cream Sandwich	4.0 – 4.0.5	18 de octubre de 2011	14 – 15
Android Jelly Bean	4.1 – 4.3.1	9 de julio de 2012	16 – 18
Android Kitkat	4.4 – 4.4.4	31 de octubre de 2013	19 – 20
Android Lollipop	5.0 – 5.1	12 de noviembre de 2014	21 – 22
Android Marshmallow	6.0 – 6.0.1	5 de octubre de 2015	23
Android Nougat	7.0 – 7.1.2	15 de junio de 2016	24 – 25
Android Ore	8.0 – 8.1	21 de agosto de 2017	26 – 27
Android Pie	9.0	6 de agosto de 2018	28
Android 10	10.0	3 de septiembre de 2019	29
Android 11	11.0	8 de septiembre de 2020	30
Android 12	12.0	Agosto de 2021	31

3.4 Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones Android y está basado en IntelliJ IDEA. Este IDE fue anunciado en la conferencia Google I/O en mayo del año 2013 reemplazando a eclipse como IDE oficial para el desarrollo de aplicaciones Android. Su primera versión estable fue publicada en diciembre de 2014. Actualmente el logo de este software es el que se muestra en la figura 3.2. (Android, 2021)

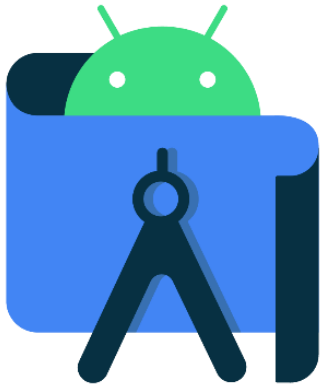

Figura 3.2 Logo actual de Android Studio

3.4.1 Funciones

Android Studio además de ofrecer un potente editor de códigos y herramientas para desarrolladores de IntelliJ, también ofrece más funciones que aumentan la productividad cuando se desarrollan aplicaciones Android como las siguientes:

- Un sistema de compilación flexible basado en Gradle.
- Un emulador rápido y cargado de funciones.
- Un entorno unificado donde puedes desarrollar para todos los dispositivos Android.
- Aplicación de cambios para insertar cambios de código y recursos a la app en ejecución sin reiniciarla

- Integración con GitHub y plantillas de código para ayudarte a compilar funciones de apps comunes y también importar código de muestra
- Variedad de marcos de trabajo y herramientas de prueba
- Herramientas de Lint para identificar problemas de rendimiento, usabilidad y compatibilidad de versiones, entre otros
- Compatibilidad con C++ y NDK
- Compatibilidad integrada con Google Cloud Platform, que facilita la integración con Google Cloud Messaging y App Engine (Android, 2021)

3.5 Kotlin

Kotlin es un lenguaje de programación estático, de código abierto y admite la programación funcional y orientada a objetos. Su sintaxis es similar a otros lenguajes de programación, como C#, Java y Scala. Cuenta con variantes que se orientas a la Java Virtual Machine (Kotlin/JMV), JavaScript (Kotlin/JS) y el código nativo (Kotlin/Native). La figura 3.3 muestra el logo actual de este lenguaje de programación. (Android, 2019)

Figura 3.3 Logo actual del lenguaje de programación Kotlin

3.5.1 Kotlin para Android

Kotlin se utiliza en el desarrollo de aplicaciones Android desde 2019. Al usar Kotlin en el desarrollo Android se pueden obtener los siguientes beneficios:

- Menos código combinado con mayor legibilidad. Se dedica menos tiempo a escribir código y a trabajar para entender código de otros.
- Lenguaje y entorno maduro. Desde su creación en el año 2011, Kotlin se ha desarrollado continuamente, no solo como lenguaje sino como un entorno con herramientas robustas. Ahora está perfectamente integrado en Android Studio y es utilizado por muchas empresas para el desarrollo de aplicaciones Android.
- Soporte de Kotlin en Android Jetpack y otras bibliotecas. Las extensiones KTX añaden características del lenguaje Kotlin, como corutinas, lambdas, etc.
- Interoperabilidad con Java. Kotlin se puede utilizar junto con Java sin la necesidad de migrar todo el código a Kotlin.
- Soporte Multiplataforma. Kotlin se puede usar no solo en el desarrollo Android sino también en iOS, backend y aplicaciones webs.
- Código seguro. Menos código y mejor legibilidad conduce a menos errores.
- Fácil aprendizaje. Kotlin es muy fácil de aprender, especialmente para los desarrolladores Java.
- Comunidad grande. Kotlin cuenta con un gran apoyo y muchas contribuciones por parte de su comunidad, que está creciendo en todo el mundo.

3.6 Spyder

Spyder (Scientific Python Development Enviroment) anteriormente conocido como Pydee, es un entorno de desarrollo integrado escrito en Python y desarrollado para científicos, ingenieros y analistas de datos que usen este lenguaje de programación, su logo actual es el mostrado en la figura 3.4. (Spyder, 2021)

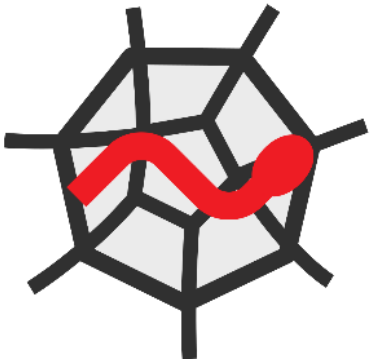

Figura 3.4 Logo actual de Spyder

3.6.1 Componentes

- Editor. Es multilenguaje y ofrece varias herramientas para mejorar la experiencia de edición.
- Consola. Ejecución de código por línea, celda o archivo.
- Explorador de variables. Explore y edite las variables durante la ejecución de un archivo.
- Graficas. Explore, amplié copie y guarde las figuras que cree.
- Depurador. Seguimiento de pasos de la ejecución de código de forma interactiva.
- Ayuda. Muestra información acerca de cualquier objeto dentro del código.

3.6.2 Plugins

- Spyder Notebook. Permite el uso de cuadernos Jupyter dentro de Spyder.
- Spyder Terminal. Muestra una terminal virtual dentro de la ventana principal de Spyder.
- Spyder Unittest. Integra marcos populares de pruebas unitarias con Spyder, lo que permite ejecutar conjuntos de pruebas y ver los resultados en el IDE. (Spyder, 2021)

3.7 Python

Python es un lenguaje de programación interpretado cuya principal filosofía es ser legible por cualquier persona con conocimientos básicos de programación. Este lenguaje de programación está orientado a objetos y es comparable a Perl, Ruby, Schema y Java.

La figura 3.5 muestra el logo actual del lenguaje de programación Python, este se caracteriza por tener una sintaxis elegante y un tipado dinámico, lo cual lo convierte en un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en muchas áreas. (Python, 2019)

Figura 3.5 Logo actual del lenguaje de programación Python

3.7.1 Características

- Usa una sintaxis elegante, haciendo a los programas escritos fácil de leer.
- Es un lenguaje fácil de usar.
- Tiene una biblioteca estándar que admite muchas tareas de programación, como la conexión a servidores web, búsqueda de texto con expresiones regulares, lectura y modificación de archivos.
- El modo interactivo de Python facilita la prueba de pequeños fragmentos de código.
- Se amplía fácilmente añadiendo nuevos módulos en un lenguaje compilado como C o C++.
- Es un software libre. También se puede modificar y distribuir libremente.

3.8 Tensorflow

Tensorflow es una plataforma de código abierto de extremo a extremo para el aprendizaje automático. Fue desarrollada por Google para satisfacer las necesidades de sistemas capaces de construir y entrenar modelos de redes neuronales.

Tensorflow cuenta con un ecosistema integral y flexible de herramientas, bibliotecas y recursos de la comunidad que permite a los investigadores innovar con el aprendizaje automático y, a los desarrolladores, compilar e implementar con facilidad aplicaciones con tecnología de aprendizaje automático.

Tensorflow desarrollado para su uso interno en Google por el equipo de Google Brain antes de ser lanzado bajo la licencia de código abierto Apache 2.0 el 9 de noviembre de 2015. Su actual logo es el mostrado en la imagen 3.6.

Figura 3.6 Logo actual de Tensorflow

3.8.1 Características

- Compilación sencilla de modelos. Compile y entrene modelos de aprendizaje automático con facilidad con ayuda de API intuitivas y de alto nivel como Keras, con ejecución inmediata, que permite una iteración de modelos y una depuración fácil.
- Producción de AA solido en cualquier parte. Entrene e implemente modelos en la nube, de forma local, en el navegador o en el dispositivo, sin importar el lenguaje que se use.
- Importante experimentación para la investigación. Arquitectura simple y flexible para llevar las nuevas ideas del concepto al código. (Tensorflow, 2021)

3.9 API REST

Una API de transferencia de estado representacional (REST), o API RESTful, es una interfaz de programación de aplicaciones (API o API web), esta fue creada por Roy Fielding y esta se ajusta a los límites de la arquitectura REST y permite la interacción con los servicios web de RESTful.

Una API es un conjunto de definiciones y protocolos, esta se utiliza para desarrollar e integrar el software de las aplicaciones.

REST es cualquier interfaz entre sistemas que usen HTTP para obtener datos o generar operaciones sobre esos datos. Es una alternativa a otros protocolos estándar de intercambio de datos como SOAP (Simple Object Access Protocol), que dispone de gran capacidad, pero es más complejo. (RedHat, 2021)

3.9.1 Características

- Protocolo cliente/servidor sin estado. Cada petición HTTP contiene toda la información necesaria para ejecutarla, con esto ni el cliente ni el servidor necesitan recordar ningún estado precio para satisfacerla.
- Operaciones. Las operaciones más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son cuatro:
    - GET (leer y consultar).
    - POST (crear).
    - PUT (modificar).
    - DELETE (eliminar).
- Los objetos REST siempre se manipulan a partir de la URI. La URI es el único identificador de cada recurso de ese sistema REST, esta nos facilita acceder a la información para su modificación o borrado.

- Interfaz uniforme. Se aplican acciones concretas (GET, POST, PUT, DELETE) sobre los recursos para la transferencia de datos en un sistema REST. Con esto se facilita la existencia de una interfaz uniforme que sistematiza el proceso con la información.
- Sistema de capas. Arquitectura jerárquica entre los componentes.
- Hipermedios. El concepto de hipermedia fue acuñado por Ted Nelson en 1965 y explica la capacidad de una interfaz de desarrollo de aplicaciones de proporcionar al cliente y al usuario los enlaces adecuados para ejecutar acciones concretas sobre los datos.

  El uso del principio HATEOAS (Hypermiedia As The Engine Of Application State) es obligatorio para cualquier API REST, de no usarlo no se considera una verdadera API REST. Este principio define que cada vez que se realice una petición al servidor y este devuelva una respuesta, parte de la información que contendrá serán los hipervínculos de navegación asociada a otros recursos del cliente.

3.9.2 Ventajas

- Separación entre cliente y servidor. El protocolo REST separa totalmente la interfaz de usuario del servidor y el almacenamiento de datos. Esto es una ventaja a la hora de portar la interfaz a otro tipo de plataformas, también aumenta la escalabilidad de los proyectos.
- Visibilidad, fiabilidad y el servidor. La separación entre cliente y servidor tiene una ventaja evidente y es que cualquier equipo de desarrollo puede escalar el producto sin excesivos problemas. Se puede migrar a otros servidores o realizar todo tipo de cambios en la base de datos, siempre y cuando los datos de cada una de las peticiones se envíen de forma correcta. Esta separación facilita tener en servidores distintos el Frontend y el Backend y eso convierte a las aplicaciones en productos más flexibles a la hora de trabajar.
- La API REST siempre es independiente del tipo de plataformas o lenguajes. La API REST se adapta al tipo de sintaxis o plataformas con las que se esté trabajando. Con

API REST se pueden tener servidores PHP, Java, Python o NodeJS. Lo único que es indispensable es que las respuestas a las peticiones se hagan siempre en el lenguaje de intercambio de información usado, normalmente XML o JSON. (BBVA, 2016)

3.10 Node.js

Node.js es un entorno de tiempo de ejecución orientado a eventos asíncronos y diseñado para crear aplicaciones network escalables. Este incluye todo lo necesario para ejecutar programas escritos en JavaScript. Fue creado por los desarrolladores originales de JavaScript, quienes lo transformaron de algo que solo podía ejecutarse en el navegador en algo que se podría ejecutar en los ordenadores como si de aplicaciones independientes se tratara. Node.js y JavaScript se ejecutan en el motor de tiempo de ejecución JavaScript V8.

La figura 3.7 muestra el logo actual de node.js.

Figura 3.7 Logo actual de node.js

3.10.1 Ventajas

- Incorporar JavaScript en su plataforma.
- Node.js se desarrolla en un entorno de tiempo de ejecución de fuente libre.
- Cuenta con un modelo de entrada y salida impulsado por eventos.
- Los administradores y los usuarios combinan estrategias de cifrado similares para crear muchas aplicaciones de Internet altamente competitivas.

- Node.js es la plataforma de software más utilizada en este momento estando por encima en entornos de ejecución y lenguajes de programación como PHP y C a la vez que tiene un tiempo de ejecución (tiempo real) muy superior.
- Con la tecnología de Node.js se pueden enviar archivos de gran peso. (Lucas, 2019)

3.11 JavaScript

JavaScript es un lenguaje de programación ligero, interpretado, o compilado justo a tiempo con funciones de primera clase. A pesar de ser mejor conocido como un lenguaje de scripting para páginas web, es un lenguaje de programación basada en prototipos, multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y declarativa. Su logo actual es el mostrado en la figura 3.8. (MDN, 2021)

Figura 3.8 Logo actual del lenguaje de programación JavaScript

JavaScript se ejecuta del lado del cliente de la web, y puede ser utilizado para estilizar y programar cómo se comportan las páginas web cuando ocurre un evento.

Al contrario de lo que se cree, JavaScript no es "Java interpretado". Intencionalmente, su sintaxis es similar a la de C++ y Java, esto para reducir la cantidad de conceptos nuevos para aprender el lenguaje.

JavaScript puede funcionar como lenguaje orientado a objetos. Los objetos se crean mediante programación, adjuntando métodos y propiedades a objetos que de otro modo en tiempo de ejecución estarían vacíos, a diferencia de las definiciones de clases sintácticas comunes en lenguajes compilados. (MDN, 2019)

Las capacidades dinámicas de JavaScript incluyen:

- Construcción de objetos en tiempo de ejecución.
- Listas de parámetros variables.
- Variables de función.
- Creación dinámica de scripts.
- Introspección de objetos.
- Recuperación de código fuente.

3.12 MongoDB

MongoDB es un sistema de base de datos distribuido NoSQL orientado a documentos, es de código abierto y está basado en C++. Está diseñada para facilitar el desarrollo y el escalado. Su logo actual se muestra en la figura 3.9

Figura 3.9 Logo actual de MongoDB

3.12.1 Características

Las características principales de MongoDB se describen a continuación:

- Consultas ad hoc. Con MongoDB se puede realizar cualquier tipo de consulta.
- Indexación. El concepto de índices en MongoDB es bastante similar al empleado en base de datos relacionales. La diferencia es que cualquier documento puede ser indexado y añadir múltiples índices secundarios.
- Replicación. MongoDB soporta el tipo de replicación primario-secundario.

- Balanceo de carga. MongoDB tiene la capacidad de ejecutarse de manera simultánea en múltiples servidores, ofreciendo un balanceo de carga o servicio de replicación de datos, de modo que se puede mantener al sistema funcionando en caso de un fallo de hardware.
- Almacenamiento de archivos. Mongo puede ser utilizado como un sistema de archivos.
- Ejecución de JavaScript del lado del servidor. MongoDB tiene la capacidad de realizar consultas utilizando JavaScript, haciendo que estas sean enviadas directamente a la base de datos para ser ejecutadas. (Robledano, 2019)

3.12.2 Ventajas y desventajas

- Ventajas
    - Validación de documentos.
    - Motores de almacenamiento integrado.
    - Menor tiempo de recuperación ante fallos.
- Desventajas
    - No es una solución adecuada para aplicaciones con transacciones complejas.
    - Aun es una tecnología joven.
    - No tiene un reemplazo para las soluciones de herencia.

3.13 Heroku

Heroku es una plataforma como servicio basada en un sistema de contenedores administrado, con servicios de datos integrados y un poderoso ecosistema, para implementar y ejecutar aplicaciones modernas. Heroku permite a las empresas construir, entregar, supervisar y alojar aplicaciones en la nube. Su logo actual es el mostrado en la figura 3.10 (Heroku, 2021)

Figura 3.10 Logo actual de Heroku

3.13.1 Características

- En Heroku el código corre siempre dentro de un dyno que es el que proporciona a la plataforma la capacidad de cómputo, es un proceso que puede usarse para ejecutar contenido web, para ejecutar procesos batch.
- Los dynos garantiza la escalabilidad en caso de que una aplicación se convierta en viral (automáticamente se levantan varios dynos)
- Los Dynos pueden ser de tres tipos: web, worker o cron.
    - WEB: se encarga del desarrollo de la aplicación web.
    - WORKER: ejecuta la base de datos.
    - CRON: se emplea para procesos de corta vida o conexiones Secure Shell (interprete de órdenes seguro).
- Los dynos aíslan de comunidades SSL, enrutamiento o blanqueo.
- Los dynos son transparentes y pueden ser levantados en otras máquinas de manera transparente.
- Heroku es “poliglota”, es decir, Heroku permite la utilización de diferentes lenguajes de programación.
- Heroku internamente se apoya en GitHub, pero no es necesario realizar pagos adicionales.
- A la hora de trasladar cualquier aplicación a Heroku, hay que adaptar cualquier tipo de grabación de fichero a filesystem que estuviéramos haciendo, y pasarlo a otros servicios Amazon.

- Las aplicaciones Java en Heroku, no necesitan un contenedor servlets.
- Heroku incluye Logplex, asumiendo que un log es un stream de eventos.

3.13.2 Ventajas

- Heroku es gratuito para aplicaciones de poco consumo.
- Permite el uso de diferentes lenguajes de programación.
- Es una plataforma fácil de usar.
- Integra varios servicios dentro de su estructura.
- Las actualizaciones en Heroku no afectan a nuestra plataforma informática.
- Se puede tener acceso desde cualquier lugar y dispositivo compatible con la computación en la nube.

3.13.3 Desventajas

- Es necesario contratar un servicio de base de datos externo y un hosting.
- Poca personalización y mínima optimización cuando se requiere más infraestructura (Gratis).
- La disponibilidad de las aplicaciones está limitada a la disponibilidad del acceso a internet.

3.14 Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS.

A pesar de ser un editor de texto ligero es muy potente. Tiene soporte incorporado para JavaScript, TypeScript y Node.js, además cuenta con un rico ecosistema para otros lenguajes

como, C++, C#, Java, Pyhton, PHP, Go, etc. Su logo actual es el mostrado en la figura 3.11. (Microsoft, 2021)

Figura 3.11 Logo actual de Visual Studio Code

3.14.1 Características

- IntelliSense. Visual Studio Code ofrece un autocompletado que proporciona terminaciones inteligentes basadas en tipos de variables, definiciones de funciones y módulos importados.
- Debugging. Imprimir sentencia de depuración es cosa del pasado. En VS Code el código se depura directamente desde el editor.
- Comandos Git. Con VS Code se puede trabajar con Git y otros proveedores de SCM. La revisión de diffs, archivos de etapas y confirmaciones se pueden realizar directamente desde el editor.
- Extensible y personalizable. VS Code permite la instalación de extensiones para agregar nuevos lenguajes, temas, depuradores y para realizar la conexión a servicios adicionales. (Microsoft, 2021)

Capitulo 4. Procedimiento

4.1 Metodología

La metodología utilizada durante el desarrollo del proyecto se muestra en la figura 4.1 y está conformada de 4 etapas: planeación, diseño, desarrollo y pruebas.

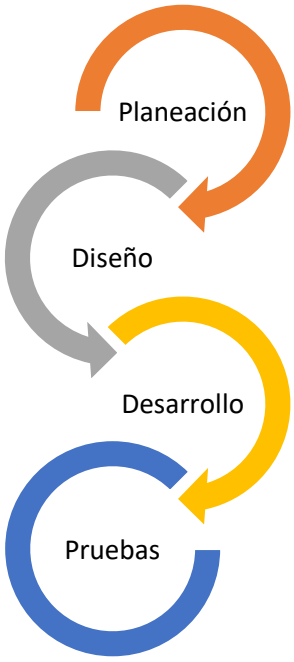

Figura 4.1 Metodología utilizada en el desarrollo del proyecto

4.1.1 Planeación

- Se realizó un análisis sobre las plagas más comunes en la región.
- Se recopilo información e imágenes sobre plagas y enfermedades del limón persa.
- Se recopilo información acerca de productos no permitidos en los cultivos de limón persa.
- Se analizaron los principales requerimientos para poder plantear una solución.

4.1.2 Diseño

- Diseño del modelo de clasificación de imágenes. Se realizó el diseño de la estructura de la red neuronal utilizada para la clasificación de imágenes.
- Diseño de la API REST. Se realizó el diseño de un servicio web para la conexión de la aplicación móvil con la base de datos.
- Diseño de la aplicación móvil. Se realizó el diseño de la aplicación móvil tomando en cuenta las funciones principales requeridas.

4.1.3 Desarrollo

- Desarrollo del modelo de clasificación de imágenes con Tensorflow.
- Desarrollo de la API REST con Node.js.
- Desarrollo de la aplicación móvil en Android Studio.

4.1.4 Pruebas

Se realizaron pruebas para verificar que no existieran errores en el funcionamiento de la aplicación móvil, que las respuestas por parte del servicio web sean los indicados y que la base de datos almacenara los datos de manera correcta.

4.2 Hardware y software

4.2.1 Hardware

Para el desarrollo del proyecto se utilizó una computadora portátil, sus especificaciones se muestran en la tabla 4.1.

Tabla 4.4.1 Especificaciones técnicas de hardware

Descripción	Equipo
Marca	Dell

Modelo	G3 15
Almacenamiento	480 GB SSD
Procesador	Intel Core i7 octava generación
RAM	16 GB
Tarjeta de video	NVIDIA GeForce GTX 1050Ti
Sistema operativo	Windows 11

4.2.2 Software

El software requerido para el desarrollo de la aplicación móvil se muestra en la tabla 4.2.

Tabla 4.4.2 Software utilizado en el desarrollo

Nombre	Versión	Dirección de descarga
Android Studio	4.2.2	https://developer.android.com/android-studio/descarga
Visual Studio Code	1.57.1	https://code.visualstudio.com/docs/?dv=win64user
Anaconda3	2020.11	https://repo.anaconda.com/archive/Anaconda3-2021.05-Windows-x86_64.exe
Node.js	15.12.0	https://nodejs.org/dist/v14.17.2/node-v14.17.2-x64.msi
Python	3.9.5150	https://www.python.org/ftp/python/3.9.6/python-3.9.6-amd64.exe

4.3 Análisis de requerimientos

4.3.1 Requerimientos funcionales

- La aplicación realizara la detección de plagas y enfermedades del limón persa.
- La aplicación mostrara información acerca de distintas plagas y enfermedades.
- La aplicación permite la búsqueda de plagas y enfermedades mediante el nombre de estas.
- La aplicación realizara maneje de datos a través de una API REST.

4.3.2 Requerimientos no funcionales

- La aplicación se debe adaptar de acuerdo a las dimensiones de la pantalla del dispositivo móvil.
- La aplicación se podrá ejecutar en versiones de Android 5 o superior.
- La aplicación debe tener una interfaz intuitiva para los usuarios.

4.4 Diseño del modelo de clasificación de imágenes

Para el diseño del modelo de clasificación de imágenes de plagas y enfermedades en el limón persa se utilizó la arquitectura básica de una red neuronal convolucional como se muestra en la figura 4.2.

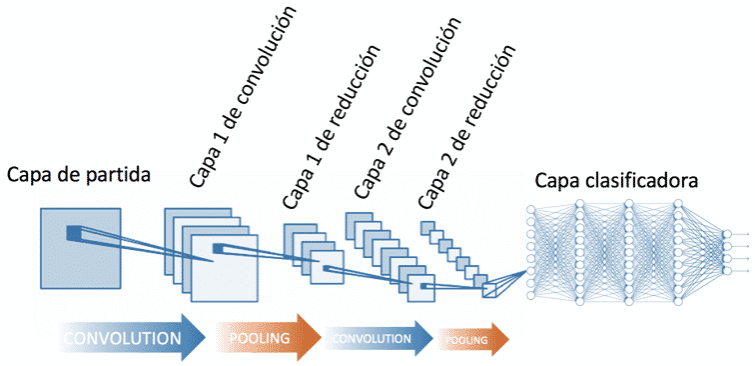

Ilustración 4.2 Arquitectura de una red neuronal convolucional

4.5 Desarrollo del modelo de clasificación de imágenes

4.5.1 El entorno

El desarrollo del modelo de clasificación de imágenes se realizó en el IDE Spyder, el cual se encuentra incluido en Anaconda3. El lenguaje utilizado fue Python.

Las principales librerías utilizadas son:

- Tensorflow.
- Tensorflow Hub.

4.5.2 Preparación de los datos

Para el desarrollo del modelo de clasificación de imágenes se utilizó un conjunto de datos compuesto por 5, 389 imágenes de 5 diferentes clases:

- Acaro Tostador
- Limón Sano
- Mancha Sectorial
- Piojo Harinoso
- Piojo Rojo

A su vez este conjunto de datos se dividió en dos, un conjunto de entrenamiento y otro de prueba, el primero compuesto por el 80% de las imágenes y el segundo por el 20% restante.

4.5.3 Overfitting

Debido a que el conjunto de imágenes utilizado para el entrenamiento del modelo de clasificación es pequeño, este tiende a caer en el problema de sobreajuste (Overfitting).

Para solucionar este problema se utilizó Transfer Learning (aprendizaje por transferencia) y Data Augmentation (aumento de datos).

Con Data Augmentation se aumentó el número de imágenes del conjunto de datos, ya que esta técnica permite agregar copias ligeramente modificadas de los datos existentes. Algunas de las modificaciones agregadas son las siguientes:

- Rotación

- Desplazamiento a lo ancho.
- Desplazamiento a lo largo.
- Zoom.
- Volteo vertical
- Entre otros.

Por otro lado, se utilizó Transfer Learning mediante el uso de un modelo previamente entrenado, en este caso se usó el modelo ImageNet classification with MobileNet V2, este admite imágenes de tamaño 224 x 224, por lo que las imágenes en el conjunto de datos se redimensionaron para poder ser compatibles con este modelo. Para usar Transfer Learning, debemos guardar el modelo previamente entrenado que se haya elegido y se debe guardar como una capa de Keras, esto para posteriormente agregarla al modelo que estemos creando.

4.5.4 El modelo

Una vez que se haya aplicado Data Augmentation al conjunto de datos y el modelo para realizar Transfer Learning se haya guardado como una capa de Keras, se puede proceder a la creación del modelo.

En este caso el modelo es bastante simple, se trata de un modelo secuencial con dos capas:

- Capa 1. Es la capa en donde se encuentra el modelo que se usara para aplicar Transfer Learning.
- Capa 2. Es una capa densa que funciona como capa de salida, recibe 2 argumentos:
    - El número de valores de salida, en este caso 5.
    - La función de activación, en este caso softmax.

Para compilar el modelo se utilizó:

- El optimizador Adam.

- Categorical Crossentropy como función de pérdida. Para calcular la pérdida de entropía cruzada entre las etiquetas y las predicciones.
- Accuracy como métrica, para calcular la frecuencia con la que las predicciones son iguales a las etiquetas.

Con los datos preparados, el modelo creado y compilado se procede a entrenarlo, el entrenamiento se hizo durante 10 épocas.

4.5.5 Validación

La figura 4.3 muestra un gráfico de los valores de la precisión durante el entrenamiento y la validación. Como se aprecia el modelo alcanza una precisión alta durante el entrenamiento llegando a 98% mientras que en la validación solo obtiene un valor de 87% que después disminuye hasta un 84%. Esto indica que nuestro modelo es un poco malo a la hora de realizar predicciones.

Figura 4.3 Precisión del modelo durante el entrenamiento y validación

4.5.6 De Tensorflow a Tensorflow Lite

Ya que el modelo está pensado para utilizarse en una aplicación Android, este se debe convertir en un modelo Tensorflow Lite que sea compatible con los dispositivos móviles.

4.6 Desarrollo de la API REST

4.6.1 Entorno

Para el desarrollo de la API se utilizó el editor de código Visual Studio Code, el lenguaje de programación JavaScript y el entorno de ejecución Node.js.

Las principales dependencias utilizadas para el desarrollo son:

- Express
- Bcryptjs
- JsonWebToken
- Moongose

4.6.2 Base de datos

La base de datos utilizada para el desarrollo de la API fue MongoDB Atlas, ya que permite el almacenamiento de datos en la nube y es compatible con Node.js.

Dentro de la base de datos existen 2 colecciones principales users y plagas a continuación, se muestra los atributos dentro de estas.

- Users
    - Status. Estado de la cuenta del usuario.
    - Username. Nombre con el que se reconoce al usuario.
    - Email. Correo electrónico del usuario.
    - Password. Contraseña de acceso del usuario.

    - Code. Código de verificación de cuenta.

- Plagas
    - Name. Nombre de la plaga.
    - imgURL. Dirección web en donde está ubicada la imagen de la plaga.
    - Description. Breve descripción de la plaga.
    - Damage. Daños que ocasiona.
    - Prevention. Medidas de prevención.

4.6.3 Rutas

La tabla 4.3 muestra las rutas disponibles dentro de la API REST.

Tabla 4.4.3 Rutas de la API

Ruta	Operación	Esquema	Descripción
/users	GET	Users	Devuelve todos los usuarios en la base de datos.
/users/:token	GET	Users	Devuelve el usuario correspondiente al token.
/users/:token	PUT	Users	Actualiza los datos del usuario correspondiente al token.
/users/:token	DELETE	Users	Elimina los datos del usuario correspondiente al token.
/auth/signup	POST	Users	Registra un usuario en la base de datos.
/auth/signin	POST	Users	Verifica si existe el usuario requerido y si sus datos son correctos.
/plagas	GET	Plagas	Devuelve todas las plagas de la base de datos.
/plagas/:name	GET	Plagas	Devuelve la plaga que coincida con el nombre dado.
/plagas/	POST	Plagas	Agrega una nueva plaga a la base de datos

/plagas/:id	PUT	Plagas	Actualiza los datos de la plaga correspondiente al id.
/plagas/:id	DELETE	Plagas	Elimina los datos de la plaga correspondiente al id.

4.6.4 Despliegue

Una vez que el desarrollo de la API ha terminado, se procede a desplegarla para poder utilizarla en la aplicación móvil, esto se hace con ayuda de Heroku.

El despliegue con Heroku es bastante sencillo, ya que proporciona una herramienta para realizar el despliegue desde la consola, Heroku CLI. Con este lo único que se debe hacer es inicializar un proyecto con Git, realizar un Commit y un Push a la rama master de nuestra aplicación Heroku.

4.7 Diseño y desarrollo de la aplicación móvil

4.7.1 Diseño

El diseño de la aplicación móvil se hizo lo más simple posible, esto para obtener una interfaz que permita a los usuarios aprovechar las funciones de la aplicación de una forma fácil e intuitiva.

Las interfaces principales son las siguientes:

- Login.
- Registro.
- Home.
- Detección de plagas.
- Información de plaga.
- Plagas.

- Productos no permitidos.

Otras interfaces:

- Información de la app.
- Preguntas frecuentes.

4.7.2 Desarrollo

El desarrollo de la aplicación móvil se dividió en cuatro etapas:

1. Creación de interfaces. Durante esta fase los diseños previamente realizados se crearon utilizando las herramientas de Android Studio, en este caso XML (Extensible Markup Language).
2. Configuración del servicio web. Durante esta fase se realizó la conexión a la base de datos, esto mediante del uso de la API REST creada para el proyecto. También se crearon los modelos necesarios para el manejo de los datos de usuarios y plagas. Las peticiones a la API REST se hicieron con ayuda de la dependencia Retrofit y Gson Converter.
3. Carga de modelo Tensorflow Lite. En esta fase se realizó la carga del modelo Tensorflow Lite, entrenado para la clasificación de plagas y enfermedades en el limón persa, y el archivo de texto en el que se encuentran las etiquetas de las posibles plagas detectadas.

   Para utilizar el modelo se utilizaron las dependencias de Tensorflow-Lite.
4. Programación. Durante esta fase se realizó la programación del comportamiento de las interfaces con el servicio web, como la forma en la que los datos se manejaran para poder almacenarse en la base de datos, como mostrar datos en las interfaces y cómo manejar las imágenes que se capturen y se procesen con el modelo de clasificación.

Capitulo 5. Resultados y conclusiones del proyecto

5.1 Resultados de la aplicación móvil

En el presente trabajo se propuso desarrollar una aplicación móvil Android como apoyo a los citricultores mexicanos en la correcta detección de plagas y enfermedades en el limón persa.

A continuación, se describen los resultados obtenidos del presente trabajo.

5.1.1 Ingreso

Para que los usuarios puedan acceder al sistema estos deben proporcionar su correo electrónico y la contraseña establecida durante el proceso de registro. La figura 5.1 muestra la interfaz de Login dentro de la aplicación móvil desarrollada.

Figura 5.1 Captura de pantalla del Login

Una vez que los usuarios ingresan sus datos de acceso, el sistema se encarga de comprobar que estos se encuentren en la base de datos, si existen otorgara el acceso al sistema, de lo contrario mostrara un error indicando el motivo por el cual no se permitió el acceso.

5.1.2 Registro

Para el proceso de registro los usuarios deben ingresar un nombre de usuario, correo electrónico y una contraseña, tal como muestra la figura 5.2 que es la correspondiente a la interfaz de registro de usuario.

El usuario y el correo electrónico son únicos, por lo que no podrán existir dos cuentas con el mismo nombre de usuario o correo electrónico, es por eso que antes de poder registrar un nuevo usuario en la base de datos, el sistema verifica que no exista otro usuario con alguno de estos datos.

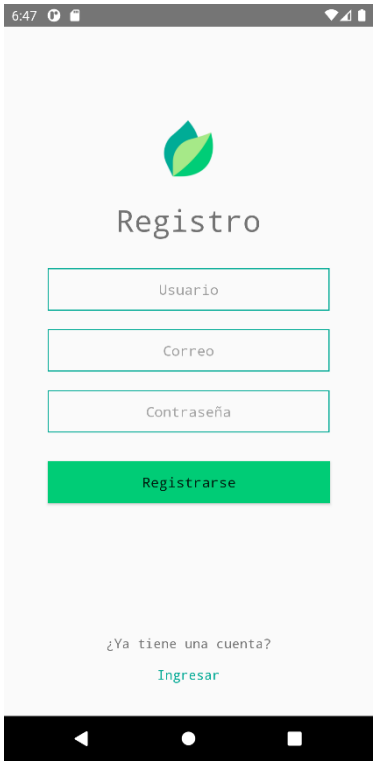

Figura 5.2 Captura de pantalla del registro

Una vez que el usuario se registra se le envía un mensaje al correo electrónico proporcionado para que pueda validar su cuenta y acceder a las funciones de la aplicación.

5.1.3 Pantalla principal

La figura 5.3 muestra la interfaz de la pantalla principal de la aplicación móvil esta pantalla es bastante simple y se compone de solo dos elementos los cuales son:

- Un área en la que el usuario puede hacer clic para acceder a la función principal de la app, el detector de plagas y enfermedades.
- Un menú, que muestra todas las funciones de la aplicación, que son:
    - Detección de plagas.
    - Plagas.
    - Productos prohibidos.
    - Preguntas frecuentes.
    - Información.

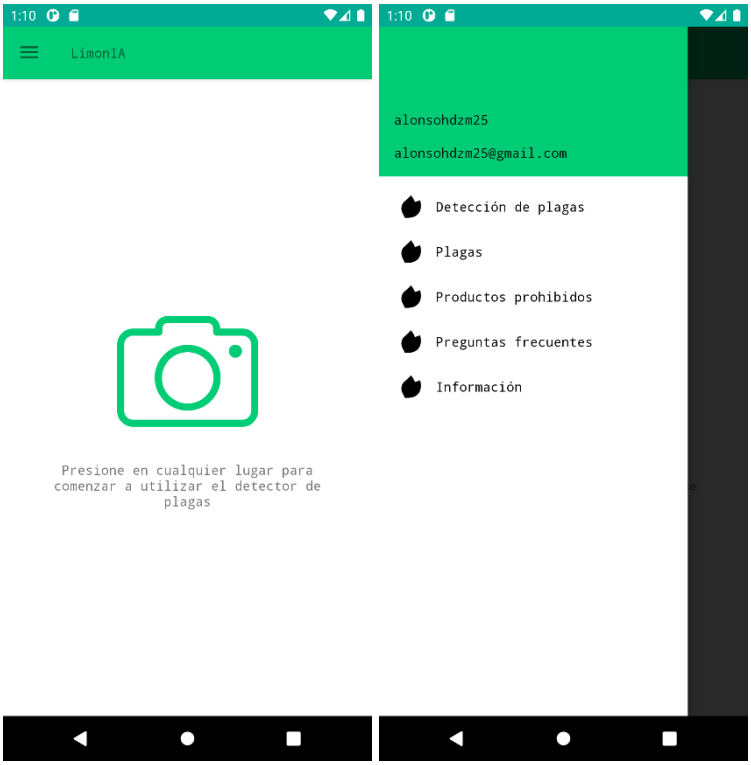

Figura 5.3 Capturas de la pantalla principal

5.1.4 Detección de plagas

La pantalla de detección de plagas y enfermedades hace uso de la cámara, esto para poder obtener una imagen de la plaga o enfermedad acerca de la cual se quiere obtener información o se desea conocer. La interfaz de esta pantalla se muestra en la figura 5.4.

Figura 5.4 Captura de pantalla del detector de plagas

Tras capturar la imagen, esta se procesa según los requerimientos del modelo de Tensorflow y una vez esta lista se realiza la detección de la plaga que podría encontrarse en la imagen.

Este proceso devuelve la etiqueta de la plaga detectada, en este caso el nombre, con este se realiza una consulta al servidor para que devuelva la información respecto a la plaga y se pueda mostrar al usuario.

5.1.5 Resultado de la detección de plagas

Una vez la plaga o enfermedad sea detectada se mostrará información acerca de esta, tal como se aprecia en la figura 5.5, la información que se mostrara al usuario es la siguiente:

- Nombre de la plaga o enfermedad.
- Una breve descripción acerca de esta.
- Los daños que provoca.
- La forma en la que se puede prevenir.

Figura 5.5 Captura de pantalla del resultado de la detección

Esta pantalla es reutilizada por la interfaz en donde se muestra la lista de plagas registradas en la base de datos.

5.1.6 Plagas

En esta pantalla el usuario puede visualizar las plagas y enfermedades del limón persa que se encuentran registradas en la base de datos, como se muestra en la figura 5.6. Desde aquí puede acceder a la información de cada una.

Esta pantalla tiene dos elementos principales:

- Una barra de búsqueda. En esta el usuario puede ingresar el nombre de la plaga o enfermedad acerca de la cual desea obtener más información.
- Una lista de plagas. En la que el usuario puede visualizar las plagas y enfermedades disponibles y consultar su información.

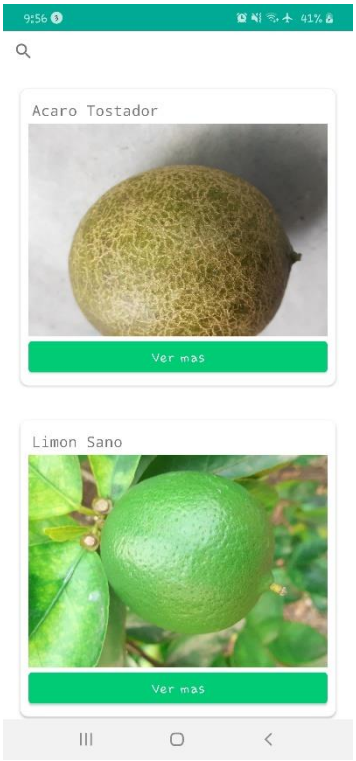

Figura 5.6 Captura de la pantalla de Plagas

5.1.7 Productos prohibidos

Esta pantalla es una lista de los productos prohibidos para uso en el cultivo de limón persa, esta muestra el ingrediente activo y el nombre comercial del producto.

Para mayor facilidad también permite realizar la búsqueda de algún producto mediante su nombre.

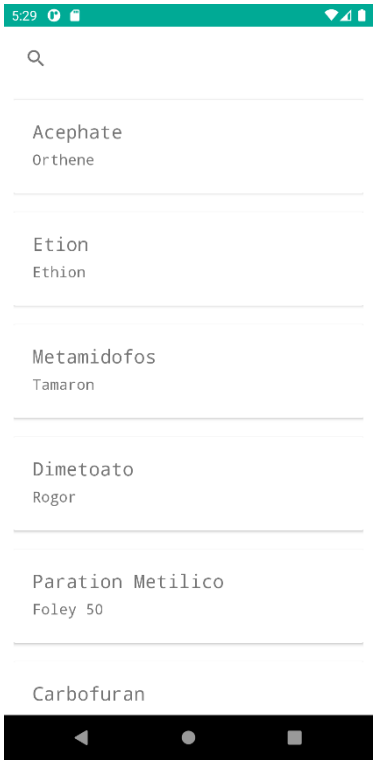

Figura 5.7 Captura de pantalla de productos prohibidos

La figura 5.7 muestra la interfaz de la pantalla en donde se muestran los productos prohibidos para el cultivo de limón persa.

5.2 Conclusiones

El buen control de plagas y enfermedades en el limón persa es de vital importancia para los productores de este, ya que al tener un buen control se evita que las huertas sean afectadas y disminuyan su productividad, algo que sería un impacto negativo para los productores.

El proyecto realizado en el Instituto Tecnológico Superior de Misantla, permitió desarrollar una herramienta útil para el área de agricultura enfocada a la producción de limón persa, ya que permite a los productores realizar una correcta detección de plagas y enfermedades en sus huertas.

En base a lo desarrollado en este proyecto se llegó a las siguientes conclusiones:

- La implementación de esta aplicación será de gran beneficio para un gran número de productores, ya que podrán tener acceso a una herramienta que facilite el proceso de detección de plagas y enfermedades del limón persa, a información sobre productos no permitidos para este tipo de cultivos y a plagas y enfermedades de las cuales no tengan conocimiento.
- Después de realizar pruebas en el campo real se observó que, aunque el objetivo de detectar plagas y enfermedades en el limón persa se cumple, existen ocasiones en las que la predicción no es correcta, esto se debe principalmente a la falta de datos de entrenamiento para nuestro modelo de predicción.

Como un proyecto a futuro la adición de nuevas funcionalidades que ayuden a los productores de limón persa a realizar otras tareas primordiales de esta área como el registro de fumigaciones, abonados y cortes permitirá aprovechar al máximo esta aplicación ofreciendo una mejor experiencia para estos.

Referencias bibliográficas

Alonso, Rodrigo (2020). "IA, Machine Learning y Deep Learning, ¿cuál es la diferencia?". *HardZone*. Obtenido de la Red Mundial el 23 de junio de 2021, https://hardzone.es/tutoriales/rendimiento/diferencias-ia-deep-machine-learning/

Android (2021). "Introducción a Android Studio". *Android Developers*. Obtenido en la Red Mundial el 23 de junio de 2021, https://developer.android.com/studio/intro?hl=es-419

Android (2019). "Descripción general de Kotlin". *Android Developers*. Obtenido en la Red Mundial el 23 de junio de 2021, https://developer.android.com/kotlin/overview?hl=es

Adeva, Roberto (2021). "¿Qué es Android?: todo sobre el sistema operativo de Google". *ADSL Zone*. Obtenido en la Red Mundial el 23 de junio de 2021, https://www.adslzone.net/reportajes/software/que-es-android/

Berger, J., Preussler, C., & Agostini, J. P. (2019). "Identificación de síntomas de Huanglongbing en hojas de citricos mediante tecnicas de deep learning". *Congreso Argentico de Agroinformatica*, 90-103.

BBVA (23 de marzo de 2016). "API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos". *BBVA API Market.* Obtenido de la Red Mundial el 24 de junio de 2021, https://www.bbvaapimarket.com/es/mundo-api/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos/

Carranza, José L. "Detección automática del daño causado por el Minador de los cítricos en la hoja de limón mexicano", *Tesis de Maestría*, Instituto Tecnológico de Acapulco, 2020.

Collado, Christian (2021). "Versiones de Android: de la primera a la última versión de Android". *Andro4All.* Obtenido en la Red Mundial el 23 de junio de 2021, https://andro4all.com/guias/android/versiones-android-historia

Espinoza Villafuerte, Nilda. "Diagnóstico automatizado para la clasificación de café trillado con broca mediante procesamiento de imágenes con Deep Learning". *Tesis de grado*, 2019.

Gasca Mantilla, Maira Cecilia, & Camargo Ariza, Luis Leonardo, & Medina Delgado, Byron (2014). "Metodología para el desarrollo de aplicaciones móviles*". Tecnura*. Obtenido

en la Red Mundial el 23 de junio de 2021, https://www.redalyc.org/articulo.oa?id=257030546003

Heroku (2021). “The Heroku Platform”. *Heroku.* Obtenido en la Red Mundial el 23 de junio de 2021, https://www.heroku.com/platform

IBM (2020). “El modelo de redes neuronales”. *IBM*. Obtenido de la Red Mundial el 23 de junio de 2021, https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-neural-model

Intagri (2018). “La Producción de Limón en México”. *Intagri*. Obtenido en la Red Mundial el 23 de junio de 2021, https://www.intagri.com/articulos/frutales/la-produccion-de-limon-en-mexico

ITSM (2019). “Oferta académica”. Instituto Tecnológico Superior de Misantla. Obtenido en la Red Mundial el 23 de junio de 2021, https://misantla.tecnm.mx/

Kotlin (2021). “Kotlin for Android”. Kotlin. Obtenido de la Red Mundial el 23 de junio de 2021, https://kotlinlang.org/docs/android-overview.html

Lucas, Jesús (4 de septiembre de 2019). “¿Qué es NodeJS y para qué sirve?”. *OpenWebinars.* Obtenido de la Red Mundial el 23 de junio de 2021, https://openwebinars.net/blog/que-es-nodejs/

MathWorks (2021). “Redes neuronales convolucionales”. *MathWorks*. Obtenido de la Red Mundial el 26 de junio de 2021, https://la.mathworks.com/discovery/convolutional-neural-network-matlab.html

Microsoft (2021). “Getting Started”. *Visual Studio Code.* Obtenido de la Red Mundial el 23 de junio de 2021, https://code.visualstudio.com/docs

MND (30 de junio de 2021). “Acerca de JavaScript”. *Mozilla Developers.* Obtenido de la Red Mundial el 23 de junio de 2021, https://developer.mozilla.org/es/docs/Web/JavaScript/About_JavaScript

NodeJS (2021). “Acerca de Node.js”. *Node.js.* Obtenido de la Red Mundial el 24 de junio de 2021, https://nodejs.org/es/about/

Roldan Ortega, B., Roshan Biswal, R., & Sanchez De La Cruz, E. (2019). "Detección de enfermedades en el sector agrícola utilizando Inteligencia Artificial". *Research in Computing Science*, 419-427.

Parada, Pascual (2021). "¿Qué son las Redes Neuronales Convolucionales?". *IEBSchool*. Obtenido de la red mundial el 26 de junio de 2021, https://www.iebschool.com/blog/redes-neuronales-convolucionales/

Pereyra, Mauro E. "Detección de enfermedades y plagas en cultivos mediante Machine Learning". *Tesis de grado*, 2020.

Python (2019). "Beginners Guide". Python. Obtenido de la Red Mundial el 23 de junio de 2021, https://wiki.python.org/moin/BeginnersGuide/Overview

RedHat (2020). "¿Qué es la metodología ágil?". *RedHat*. Obtenido en la Red Mundial el 23 de junio de 2021, https://www.redhat.com/es/devops/what-is-agile-methodology

RedHat (2021). "¿Qué es una API de REST?". RedHat. Obtenido de la Red Mundial el 24 de junio de 2021, https://www.redhat.com/es/topics/api/what-is-a-rest-api

Robledano, Ángel (28 de octubre de 2019). "¿Qué es MongoDB?". *OpenWebinars*. Obtenido de la Red Mundial el 24 de junio de 2021, https://openwebinars.net/blog/que-es-mongodb/

Santander (21 de diciembre de 2020). "Metodologías de desarrollo de software: ¿qué son?". *Santander*. Obtenido en la Red Mundial el 23 de junio de 2021, https://blog.becas-santander.com/es/metodologias-desarrollo-software.html

SmartPanel (2018). "¿Qué es el Deep Learning?". *SmartPanel*. Obtenido de la Red Mundial el 23 de junio de 2021, https://www.smartpanel.com/que-es-deep-learning/

Spyder (2021). "Welcome to Spyder's Documentation". *Spyder*. Obtenido en la Red Mundial el 23 de junio de 2021, https://www.spyder-ide.org/

Tensorflow (2021). "¿Por qué Tensorflow?". *Tensorflow*. Obtenido de la Red Mundial el 23 de junio de 2021, https://www.tensorflow.org/about

Toledo, Rogelio (14 de enero de 2020). "Metodologías agiles: ¿Qué son?". *Grupo Cibernos*. Obtenido en la Red Mundial el 23 de junio de 2021,

https://www.grupocibernos.com/blog/conoce-las-metodologias-de-desarrollo-agil-mas-usadas#:~:text=Metodolog%C3%ADas%20%C3%A1giles%3A%20%C2%BFQu%C3%A9%20son%3F,se%20ahorra%20tiempo%20y%20costes

Osoyo, Ariel y Elms Rhiannon (2020). “Citrus Annual”. *United States Deparment of Agriculture*. Obtenido de la Red Mundial el 26 de junio de 2021, https://apps.fas.usda.gov/newgainapi/api/Report/DownloadReportByFileName?fileName=Citrus%20Annual_Mexico%20City_Mexico_12-15-2020

ANEXOS

El diseño de la interfaz de la aplicación móvil de detección de plagas y enfermedades en el limón persa se realizó en un software de diseño que ofrece suficientes herramientas para realizar el prototipeado de aplicaciones móviles y web, el nombre de este software es AdobeXD.

Ilustración 1 Diseño de la interfaz de Login

Ilustración 2 Diseño de interfaz de registro de usuario

Ilustración 3 Diseño de interfaz de página principal

Ilustración 4 Diseño de la interfaz de detección de plagas

Ilustración 5 Diseño de la interfaz de información de la plaga detectada

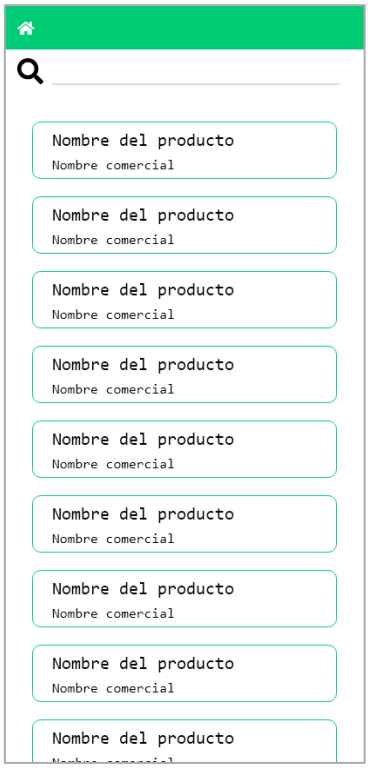

Ilustración 6 Diseño de la interfaz de productos no permitidos

Ilustración 7 Diseño de interfaz de plagas

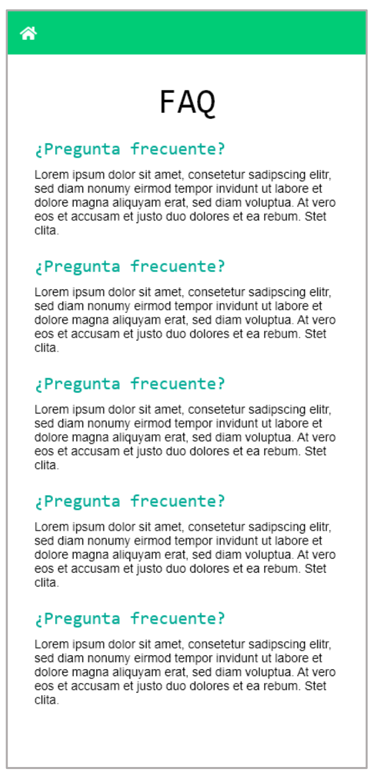

Ilustración 8 Diseño de la sección de preguntas frecuentes