Institutito Politécnico Nacional

Escuela Superior de Cómputo

Prototipo de aplicación para la detección de deficiencia de nutrientes en cultivos de hidroponía.

Presenta:
Edgar Rodrigo Arredondo Basurto

Directores
Ing. Eduardo Gutiérrez Aldana
Dr. José Félix Serrano Talamantes

1. Identificación del problema

- La hidroponía permite cultivar especies para el consumo humano en regiones donde no existen tierras de cultivo o donde el clima no es favorable para el cultivo tradicional de ciertas especies.
- Mercado valuado en 411.88 millones de dólares (USD) en 2017 y una proyección de 752.57 millones para 2022 [1].
- En México el 60% de los invernaderos de hidroponía que se han instalado han fracasado ante el desconocimiento de productores, la falta de capacitación de técnicos y de mercado [2].
- Uno de los principales problemas es la identificación y tratamiento de enfermedades en las plantas, derivadas de bacterias, insectos, virus, deficiencia de nutrientes, etcétera.

2. Objetivos

Objetivo general.

Diseñar y desarrollar el prototipo de una aplicación de visión por computadora que analiza imágenes de hojas de tomate con una anormalidad visible y realiza un diagnóstico de una posible enfermedad, bajo un subconjunto predefinido de enfermedades del tomate

- Virus del rizado amarillo del tomate.
- Virus del mosaico del tomate.
- Corynespora cassiicola. Mancha en forma de blanco.
- Araña roja.
- Septoriosis.
- Passalora fulva. Moho en la hoja.
- Tizón tardío.
- Tizón temprano.
- Mancha bacteriana.

2. Objetivos

Objetivos particulares

- Entrenar el modelo de clasificación de imágenes con el conjunto de datos predefinido.
- Realizar pruebas de eficiencia del clasificador, obteniendo un resultado superior al 90%.
- Implementar un sistema web en el que se aloje el clasificador y permita realizar identificaciones de enfermedades a los usuarios.

3. Resumen

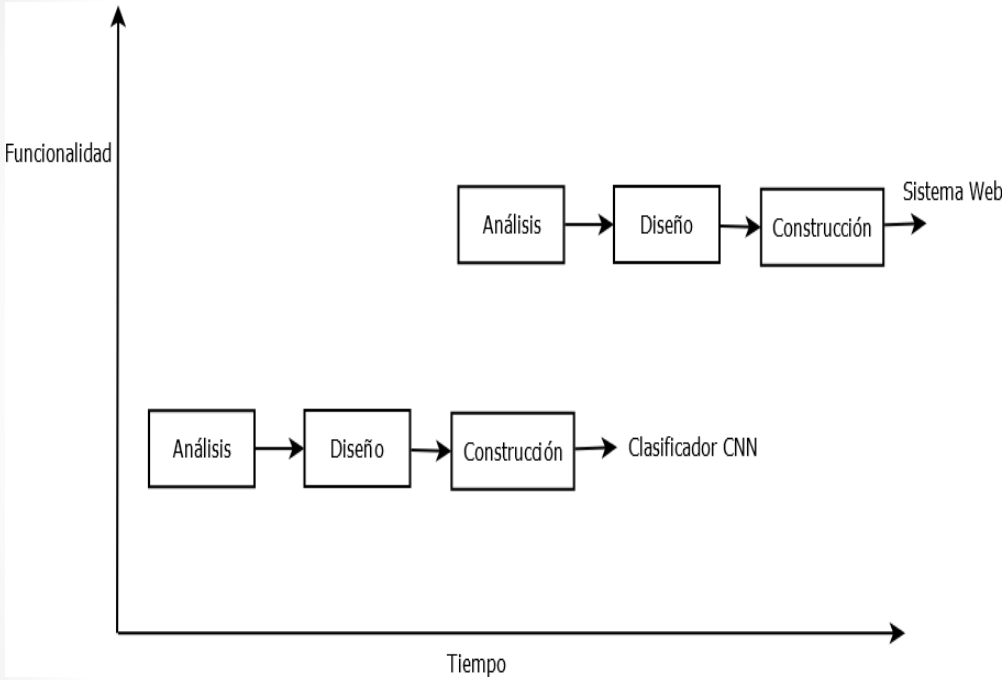

Figura 3.1. Metodología.

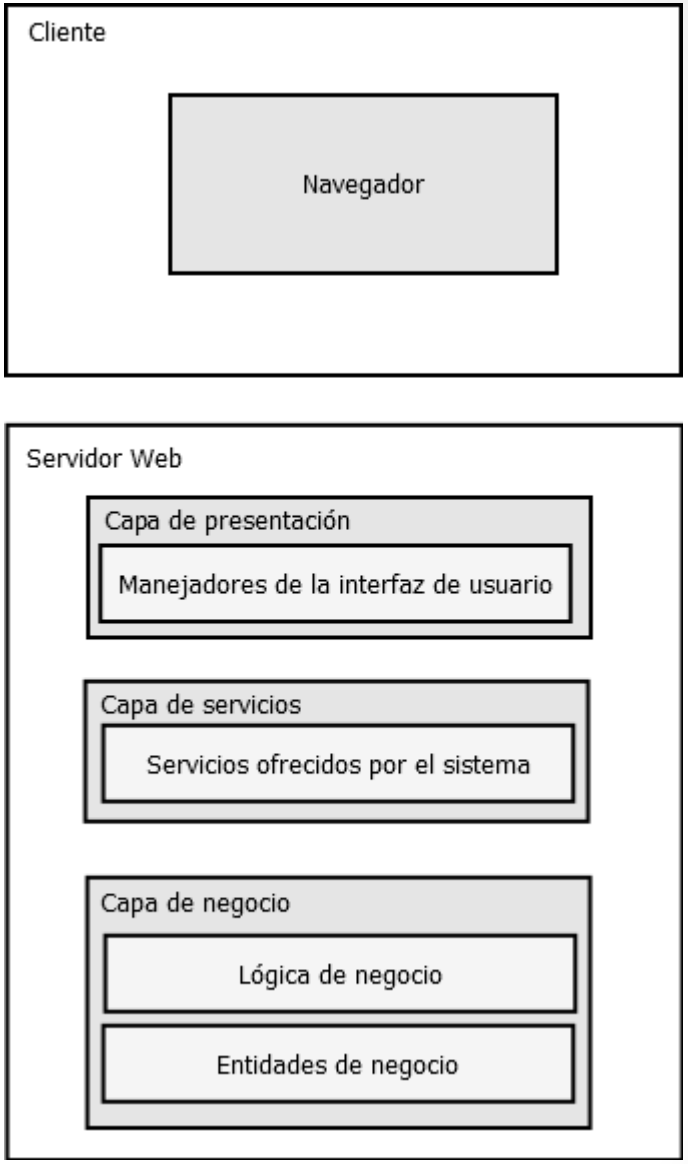

Figura 3.2. Arquitectura.

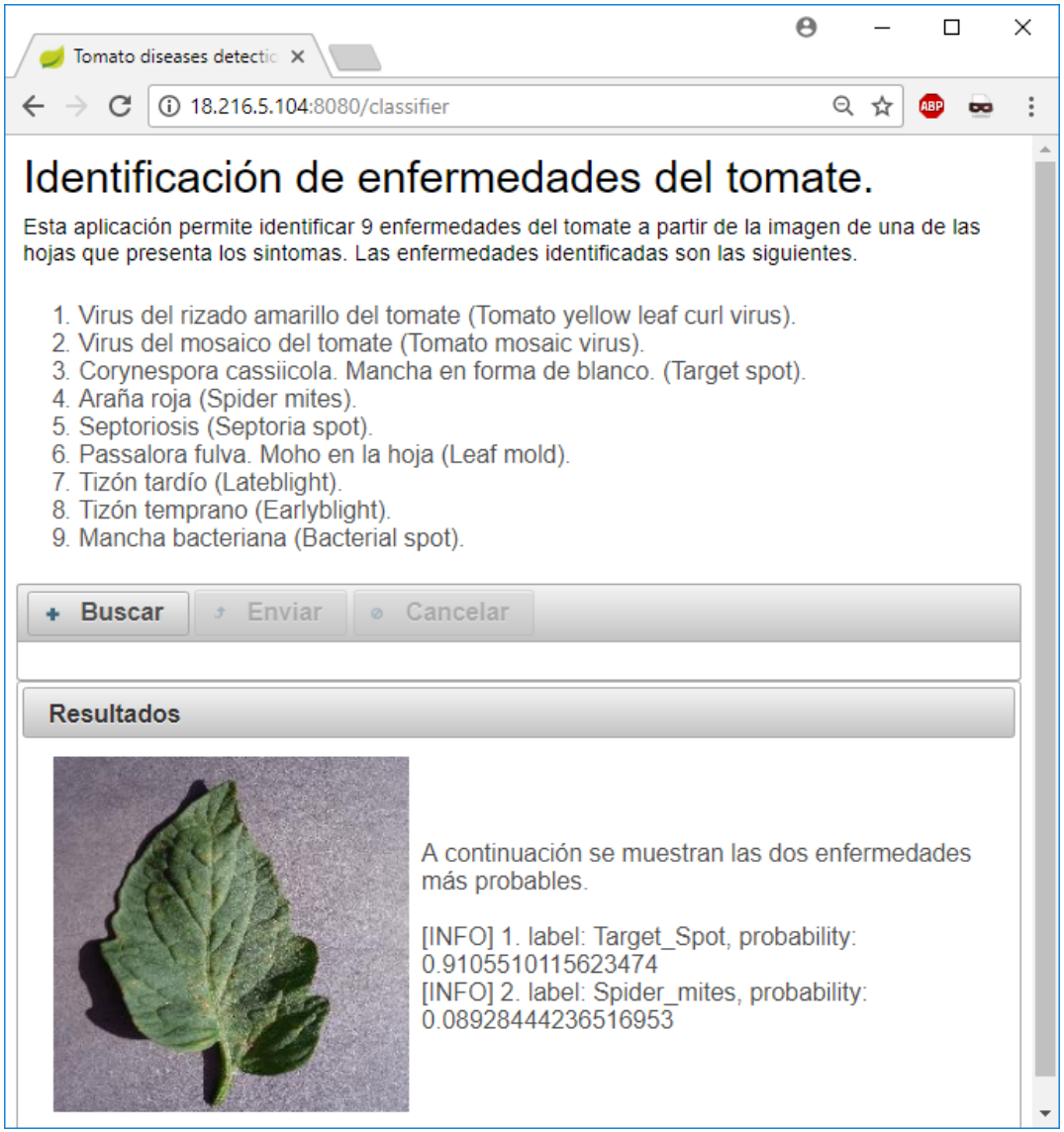

Figura 3.3. Interfaz de usuario.

4. Clasificador.

La clasificación de imágenes usando algoritmos de machine learning está compuesta de dos fases:

- Fase de entrenamiento. Se entrena el algoritmo usando un conjunto de datos pre-clasificados (muestras etiquetadas).
- Fase de predicción. Se utiliza el algoritmo entrenado para predecir la etiqueta de imágenes fuera del conjunto de entrenamiento.

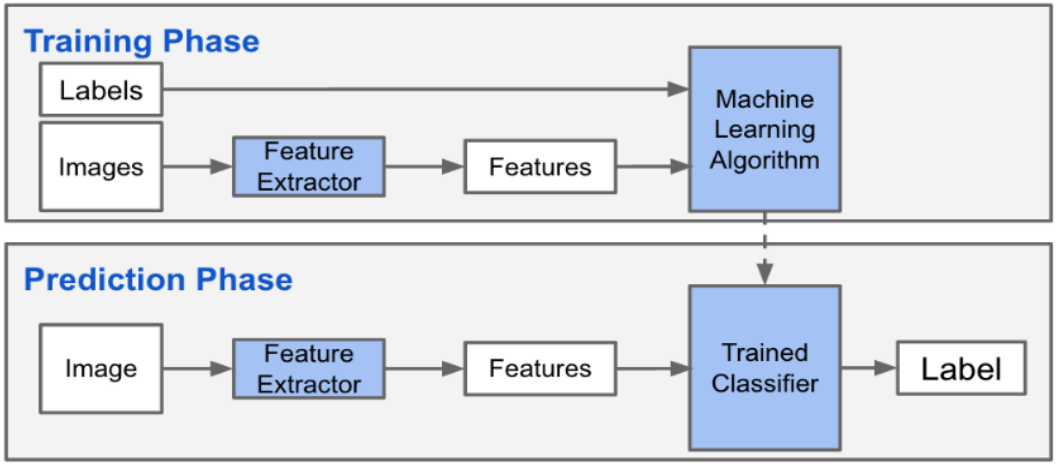

Figura 4.1. Fases de clasificación de imágenes en algoritmos de machine learning.

4. Clasificador.

En los algoritmos tradicionales de machine learning, la ingeniería de descriptores es un proceso manual, es decir, la selección y extracción debe ser diseñada e implementada por el programador.

La ingeniería de descriptores es costosa, consume tiempo importante y requiere de cierta experiencia. Un descriptor mal seleccionado condena al fracaso el resto del clasificador.

En los algoritmos de deep learning, la ingeniería de descriptores es un proceso automático. Es así como estos algoritmos prometen resultados más precisos comparados con los algoritmos de machine learning, con menos o incluso sin ingeniería de descriptores.

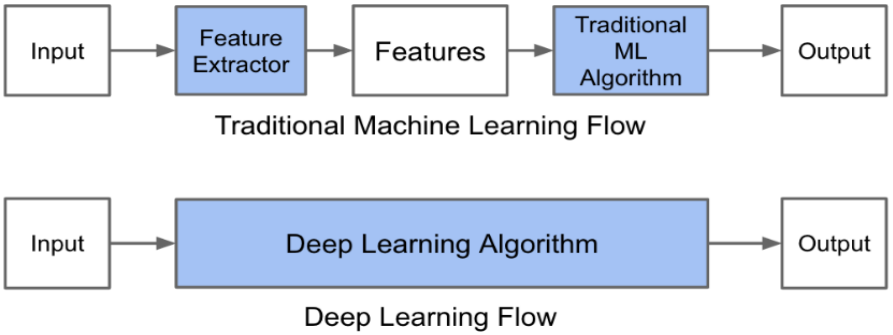

Figura 4.2. Diferencias entre la clasificación con modelos de machine learning y deep learning.

4. Clasificador.

El algoritmo clasificador de este prototipo es una red neuronal convolucional (CNN), que forma parte de los algoritmos de deep learning.

Esta red neuronal es una red unidireccional con múltiples capas ocultas y que asume que la entrada es una imagen, por lo que tienen buen desempeño en tareas de reconocimiento visual.

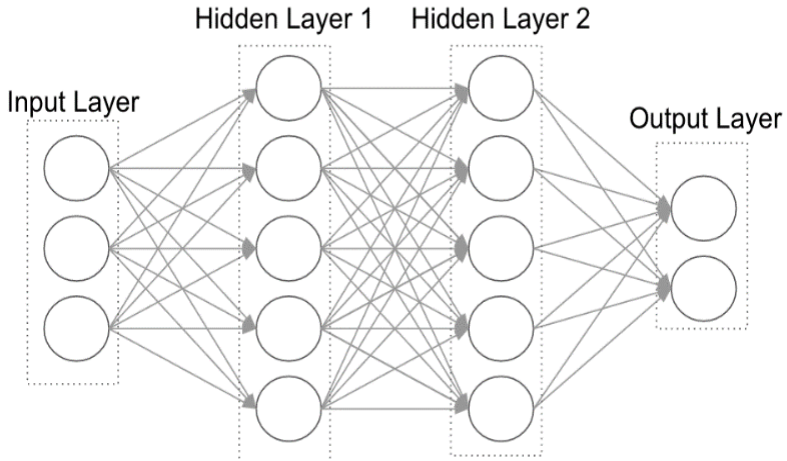

Figura 4.3. Red neuronal unidireccional con dos capas ocultas.

4. Clasificador.

La arquitectura de una CNNs está compuesta por capas tres capas:

- Capa convolucional (CONV layer). Extracción de descriptores.
- Capa de agrupamiento (pooling layer). Reducir el tamaño de la representación.
- Capa conectada completamente (Fully-connected layer). Clasificador.

Estas capas permiten a la red codificar determinadas propiedades de la imagen.

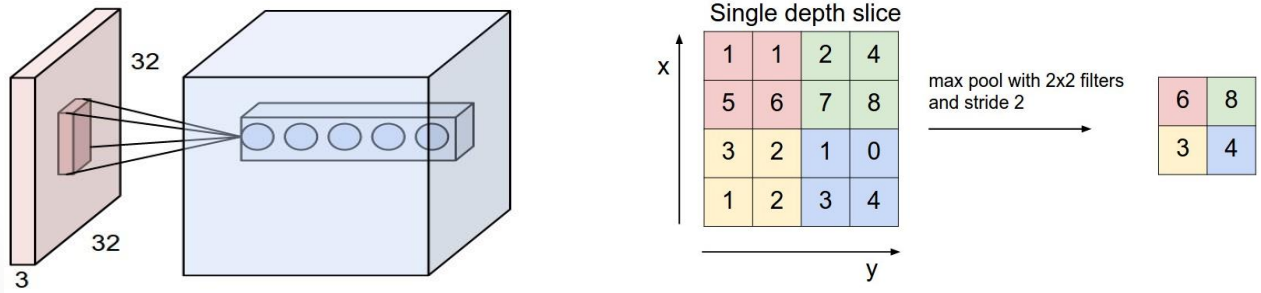

Figura 4.4. Capa convolucional (izquierda) y capa de agrupamiento (derecha).

4. Clasificador.

La arquitectura más sencilla de una red neuronal convolucional comienza con una capa de entrada (imágenes) seguida de una secuencia de capas convolucionales y de agrupamiento, terminando con capas completamente conectadas. Las capas convolucionales usualmente están seguidas de una capa de funciones de activación ReLU.

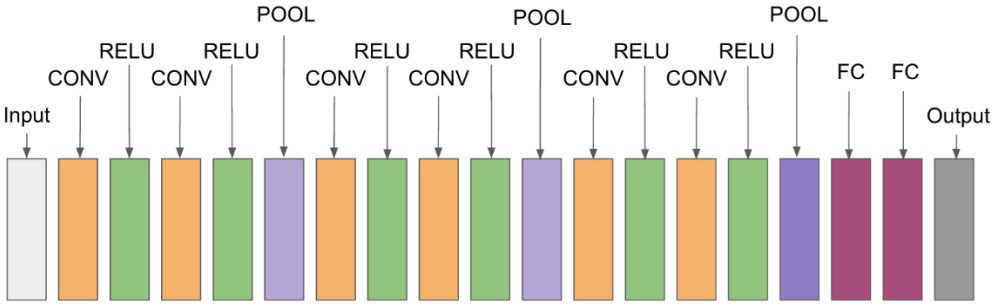

Figura 4.5. Ejemplo de arquitectura CNN.

4. Clasificador.

Existen distintas arquitecturas con un nombre asignado. Las más comunes son las siguientes:

- LeNet. Desarrollada por Yann LeCun en la década de 1990. Con esta arquitectura se desarrollaron las primeras aplicaciones exitosas de redes convolucionales.
- AlexNet. Es el primer trabajo que popularizo las redes convolucionales en la visión por computadora, desarrollado por Alex Krizhevsky, Ilya Sutskever y Geoff Hinton. La AlexNet fue inscrita en el reto ImageNet ILSVRC challenge en 2012 y superó de forma significativa al segundo clasificado.
- ZF Net. Una mejora de la AlexNet que resultó ganadora del ILSVRC 2013, desarrollada por Matthew Zeiler y Rob Fergus.
- GoogLeNet. La ganadora del ILSVRC 2014, desarrollada por Szegedy et al de Google.

4. Clasificador.

El conjunto de datos para el proceso de entrenamiento y prueba consistió de 16,419, 80% para entrenamiento y el restante para pruebas. La cantidad de imágenes por cada enfermedad se muestra en la tabla siguiente.

Enfermedad	Número de imágenes
Virus del rizado amarillo del tomate (Tomato yellow leaf curl virus)	4032
Virus del mosaico del tomate (Tomato mosaic virus)	325
Corynespora cassiicola. Mancha en forma de blanco (Target spot)	1,356
Araña roja (Spider mites)	1,628
Septoriosis (Septoria spot)	1,723
Passalora fulva. Moho en la hoja (Leaf mold)	904
Tizón tardío (Lateblight)	1,781
Tizón temprano (Earlyblight)	952
Mancha bacteriana (Bacterial spot)	2,127
Total	14,828

4. Clasificador.

Es posible realizar predicciones usando el clasificador con C++, Python o Matlab, teniendo como requisito tener Caffe y las bibliotecas correspondientes para cada lenguaje instaladas.

OpenCV 3.3.0 incluye un modulo para deep learning (dnn) que también permite realizar predicciones, pero sin la necesidad de tener instalado Caffe, lo que también añade Java como otra opción para realizar predicciones con el clasificador entrenado.

5. Aplicación Web.

Requisitos funcionales.

RF01. Selección de imagen.
Nivel de madurez: Alta.
Prioridad: Media.
Descripción: El sistema permitirá seleccionar una imagen del sistema de archivos local del dispositivo del usuario.

RF02. Clasificación.
Nivel de madurez: Alta.
Prioridad: Alta.
Descripción: El sistema clasificará la imagen en alguna de las nueve enfermedades descritas en la tabla 3.1 o en la clase "Hoja sana", indicando la probabilidad de que esa sea la enfermedad correcta.

5. Aplicación Web.

Requisitos no funcionales.

RNF01. La eficiencia de clasificación deberá ser superior al 90%.
RNF02. El clasificador identificará la clase de la imagen de entrada en un tiempo no mayor a cinco segundos.
RNF05. El sistema será desarrollado para un ambiente Web.
RNF06. El sistema podrá usarse a través del navegador Chrome v.64

5. Aplicación Web.

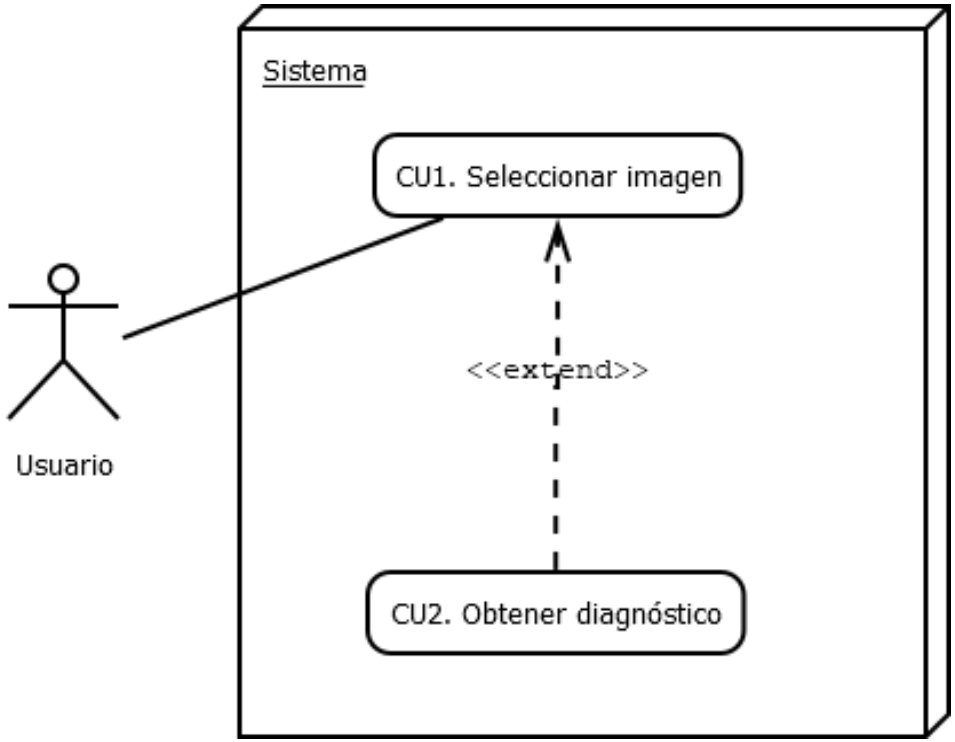

Figura 5.1. Modelo de casos de uso de la aplicación Web.

5. Aplicación Web.

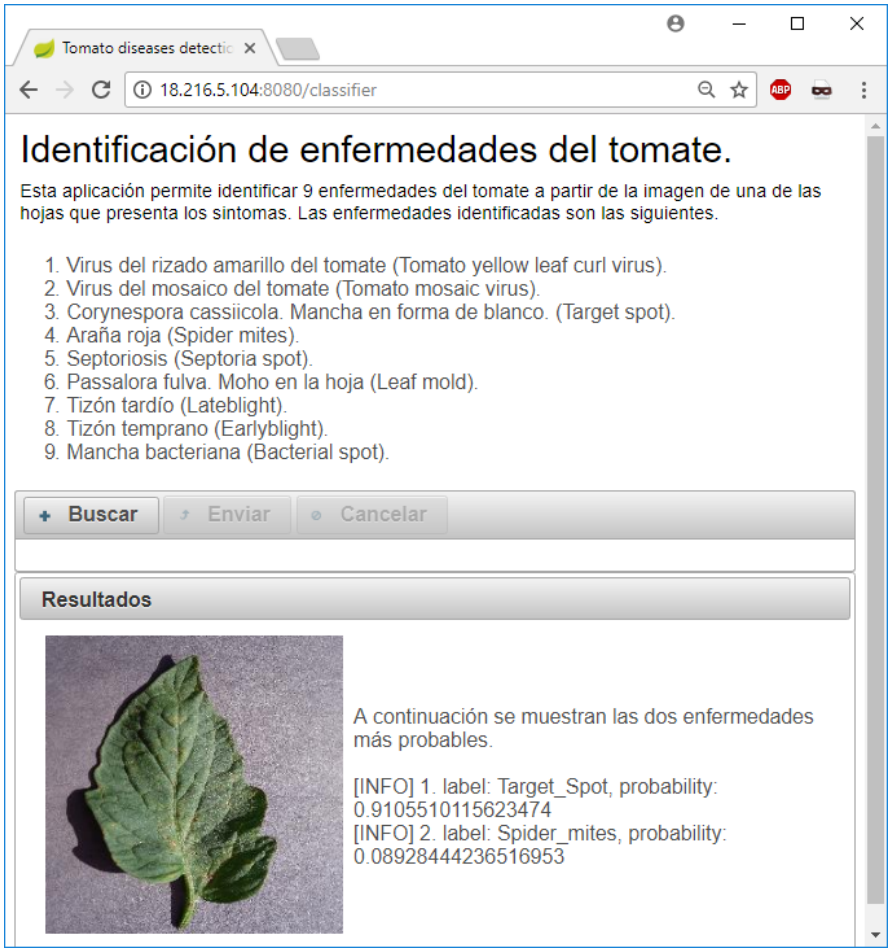

Figura 5.2. Diseño de la interfaz de usuario.

5. Aplicación Web.

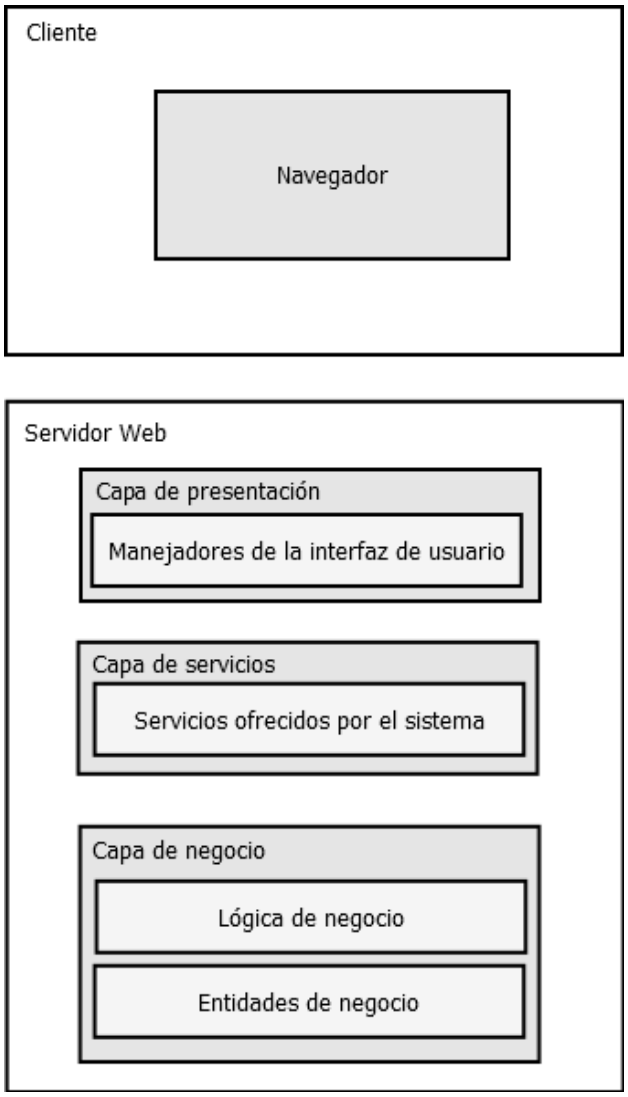

Figura 5.3. Arquitectura de la aplicación Web.

5. Aplicación Web.

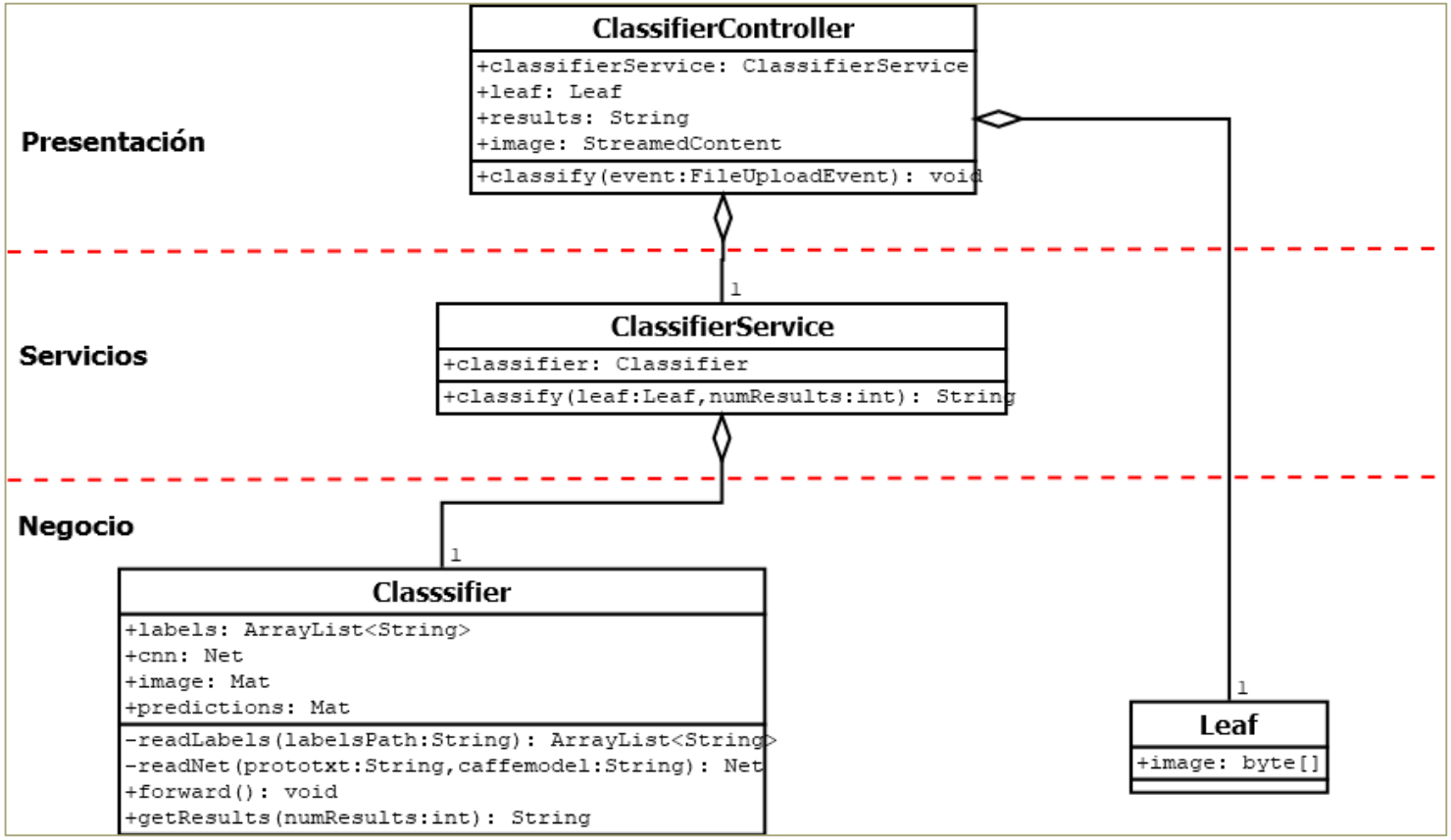

Figura 5.4. Diagrama de clases de la aplicación Web.

5. Aplicación Web.

Las tecnologías de desarrollo de la aplicación fueron las siguientes:

- Java. Lenguaje de desarrollo.
- OpenCV. Para la etapa de clasificación.
- Maven y Spring boot. Para el desarrollo y administración de dependencias del proyecto.
- JavaServer Faces (JSF) y PrimeFaces. Para el desarrollo de las interfaces de usuario y la comunicación entre el cliente y la capa de presentación del servidor.

La versión final de la aplicación consiste en un jar que contiene todas las dependencias incluyendo un servidor tomcat, a excepción de las bibliotecas nativas (*.so, *.dll) de OpenCV para Java. La ubicación de dichas bibliotecas se indica con la bandera `Djava.library.path` del comando java al ejecutar el jar.

6. Resultados.

A continuación se presentan los resultados finales obtenidos.

Eficiencia de clasificación:
Tiempo de respuesta:
...

7. Conclusiones.

...

8. Trabajo a futuro.

...

9. Referencias

[1] Research and Markets. Global Hydroponics Market - Forecasts from 2017 to 2022. [En línea] Disponible en: https://www.researchandmarkets.com.
[2] Sánchez F. Entrevista con Félipe Sanchez del Castillo, investigador de la Universidad Autónoma Chapingo. Recuperado de http://www.2000agro.com.mx.

Gracias