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 es una técnica de producción intensiva de plantas que se caracteriza por suministrar los nutrientes de forma controlada a través de una solución de nutrientes minerales.
- 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 [1].
- Mercado valuado en 411.88 millones de dólares (USD) en 2017 y una proyección de 752.57 millones para 2022 [2].
- 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 [3].
- 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.

Figura 2.1. Tizón temprano.

Figura 2.2. Moho en la hoja.

2. Objetivos

Figura 2.3. Virus del rizado amarillo.

Figura 2.4. Virus del mosaico.

Figura 2.5. Mancha en forma de blanco.

Figura 2.6. Araña roja.

Figura 2.7. Septoriosis.

Figura 2.8. Tizón tardío.

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.

2. Objetivos

Clase	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
Hoja sana	1,591
Total	16,419

Figura 2.9. Conjunto de datos.

3. Resumen

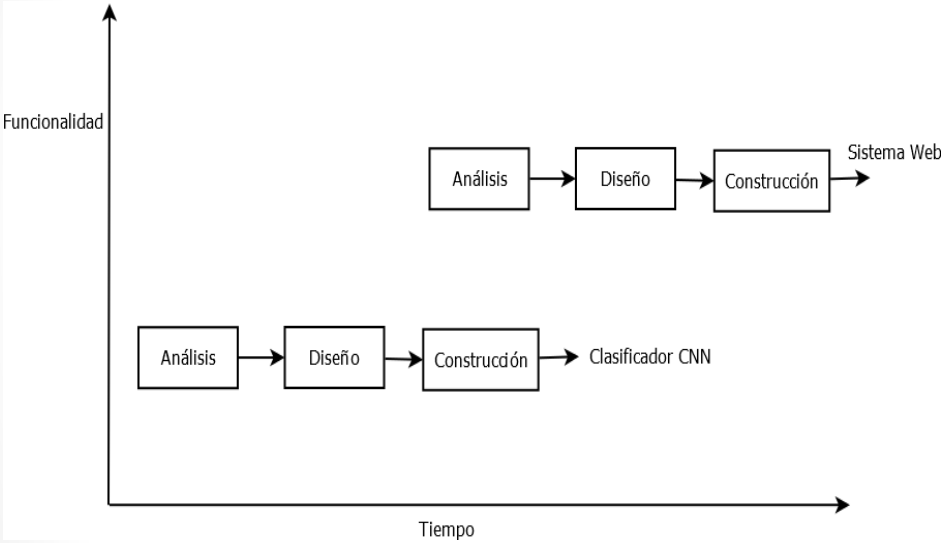

Figura 3.1. Metodología.

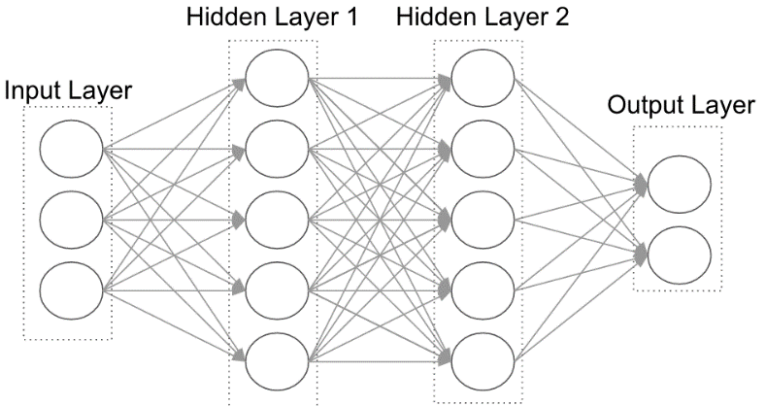

Figura 3.2. Algoritmo clasificador

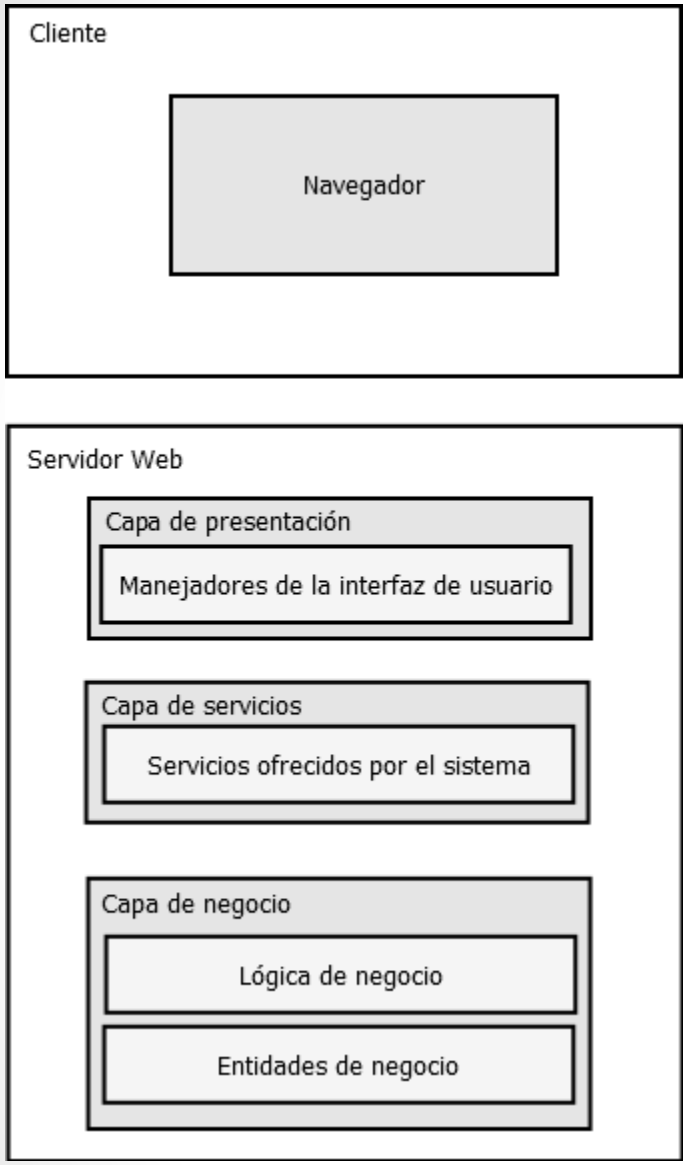

Figura 3.3. Arquitectura.

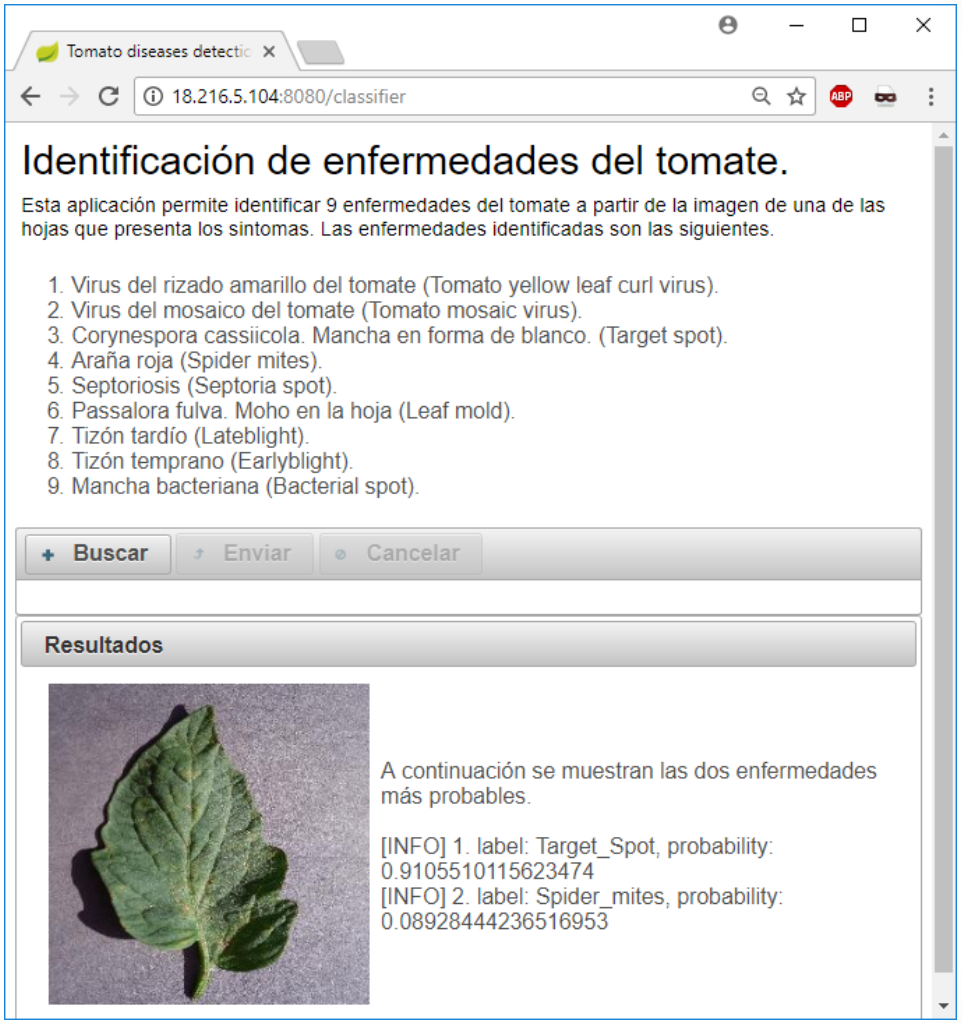

Figura 3.4. Interfaz de usuario.

4. Clasificador.

La clasificación de imágenes usando algoritmos de aprendizaje automático (*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.

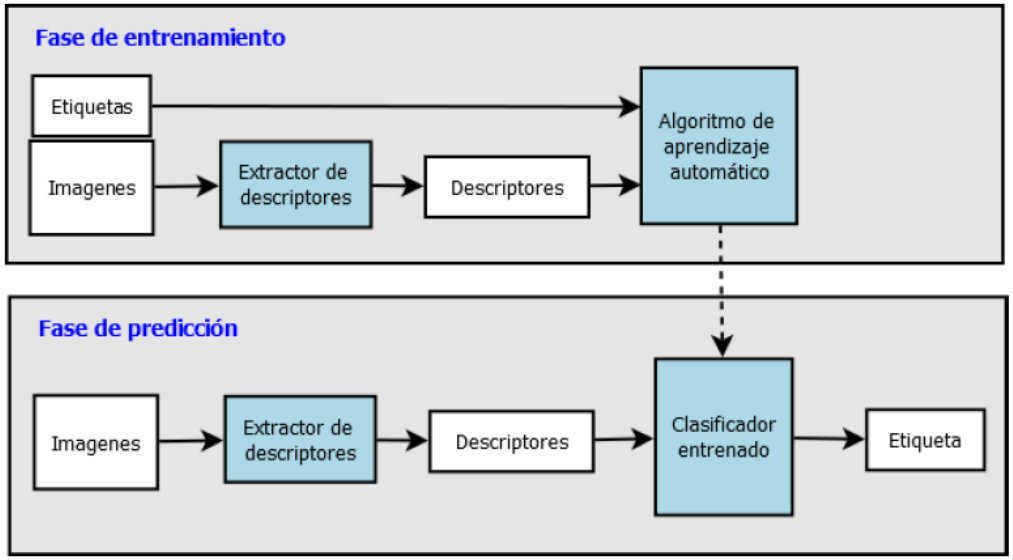

Figura 4.1. Fases de clasificación de imágenes en algoritmos de aprendizaje automático.

4. Clasificador.

En los algoritmos tradicionales de aprendizaje automático, 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 cambio, en los algoritmos de aprendizaje profundo (*deep learning*), la ingeniería de descriptores es un proceso automático. Es así como estos algoritmos prometen resultados más precisos comparados con los algoritmos de aprendizaje automático, con menos o incluso sin ingeniería de descriptores.

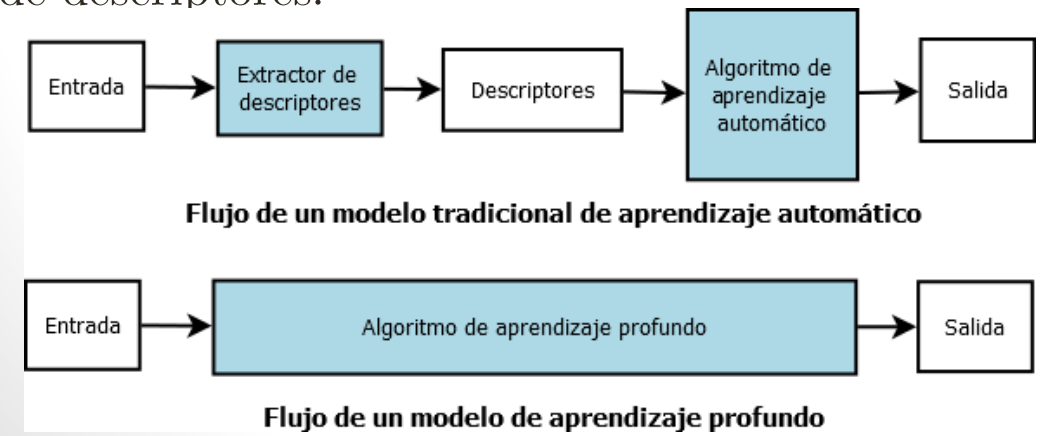

Figura 4.2. Diferencias entre la clasificación con modelos de aprendizaje automático y aprendizaje profundo.

4. Clasificador.

El algoritmo clasificador de este prototipo es una red neuronal convolucional (CNN por sus siglas en inglés), que forma parte de los algoritmos de aprendizaje profundo.

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, como lo demuestran sus resultados en el ILSVRC (*ImageNet Large Scale Visual Recognition Challenge*).

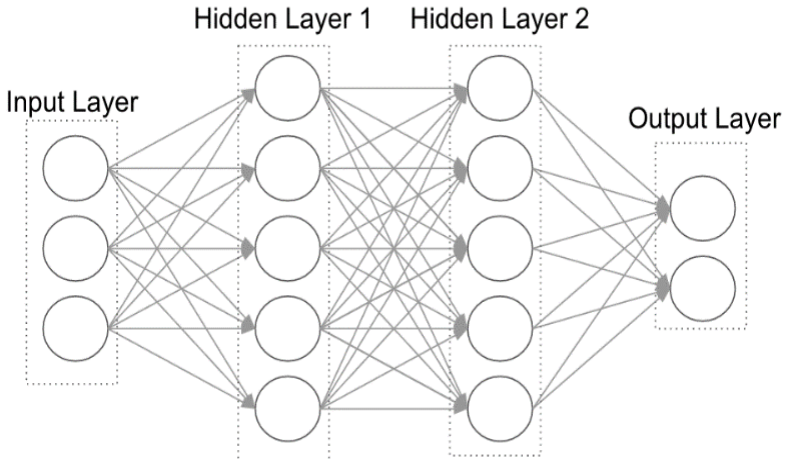

Figura 4.3. Red neuronal unidireccional con dos capas ocultas.

4. Clasificador.

La arquitectura de una CNN está compuesta por capas tres tipos de capas:

- Capa convolucional. Extracción de descriptores.
- Capa de agrupamiento. Reducir el tamaño de la representación.
- Capa completamente conectada. Clasificador.

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

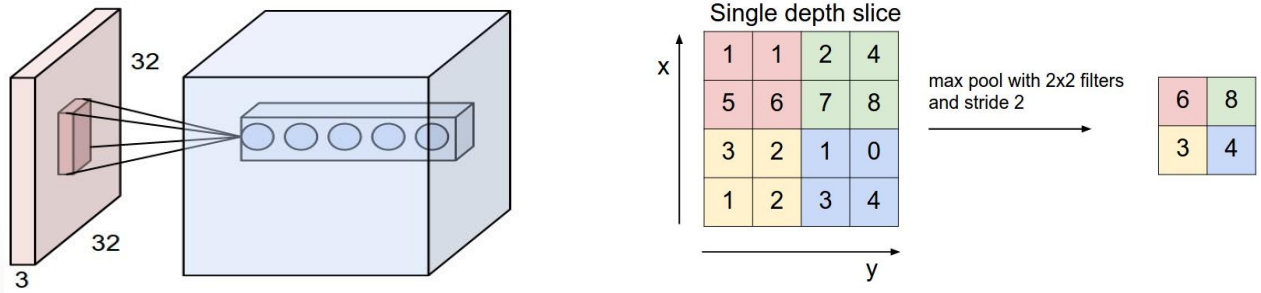

Figura 4.4. Capa convolucional (izquierda) y capa de agrupamiento (derecha).

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.

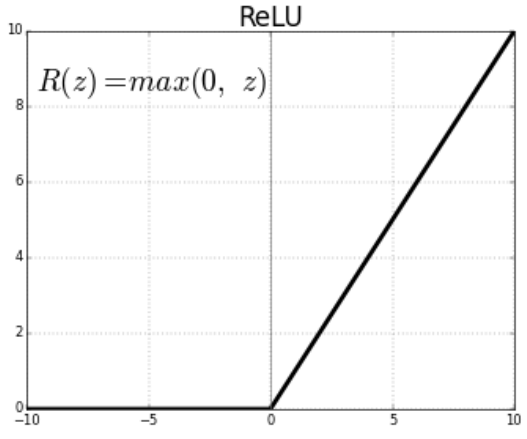

Figura 4.5.

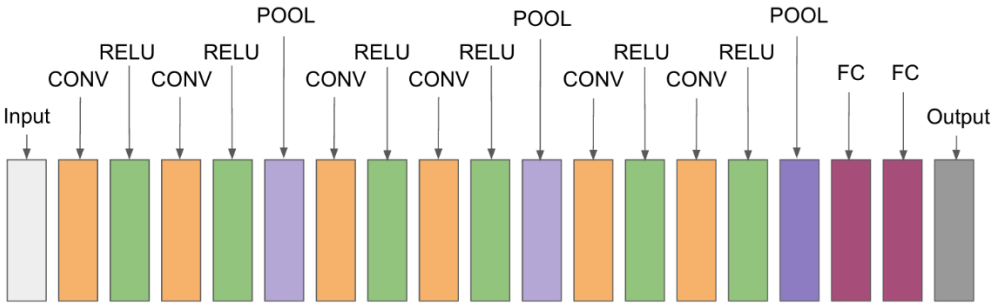

Figura 4.6. Ejemplo de arquitectura CNN.

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 clase se muestra en la tabla siguiente.

Clase	**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
Hoja sana	1,591
Total	16,419

4. Clasificador.

Entrenamiento del clasificador.

- El entrenamiento del clasificador se realizó con el framework de aprendizaje profundo Caffe.
- La máquina utilizada tiene las siguientes características: 4 GB de RAM, procesador Intel i3, sin GPU compatible con Caffe.
- Se seleccionó la arquitectura AlexNet. Está arquitectura tuvo un tiempo estimado de entrenamiento de 7 días. En cambio la GoogLeNet tuvo un tiempo estimado de entrenamiento de 91 días!
- La AlexNet alcanzó una eficiencia superior al 98% después de 8 mil iteraciones de entrenamiento, en un periodo de una semana.

4. Clasificador.

Es posible realizar predicciones usando el clasificador con C++, Python o Matlab, teniendo como requisito contar con 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 da mayor flexibilidad al desarrollo de aplicaciones que incluyan un clasificador entrenado con Caffe. Por está razón se decidió realizar las predicciones con el módulo dnn de OpenCV.

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 de la figura 2.9, 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 seguirá los estándares CSS3, XHTML 1.0 Transitional y ECMAScript 5 para el desarrollo de la interfaz gráfica.

5. Aplicación Web.

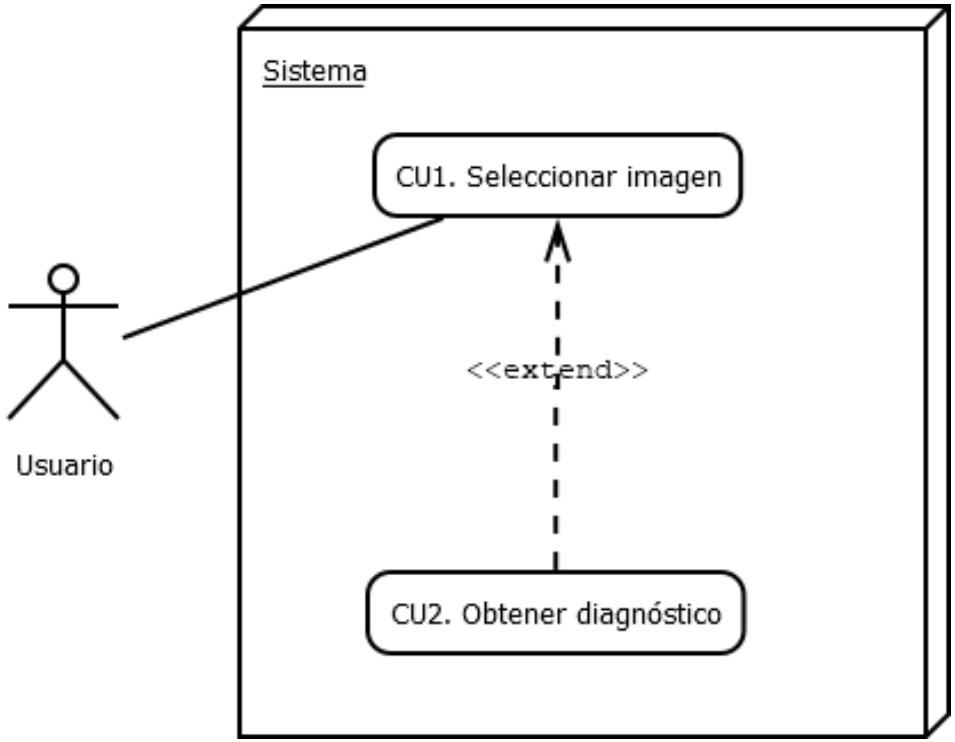

Figura 5.1. Modelo de casos de uso de la aplicación Web.

5. Aplicación Web.

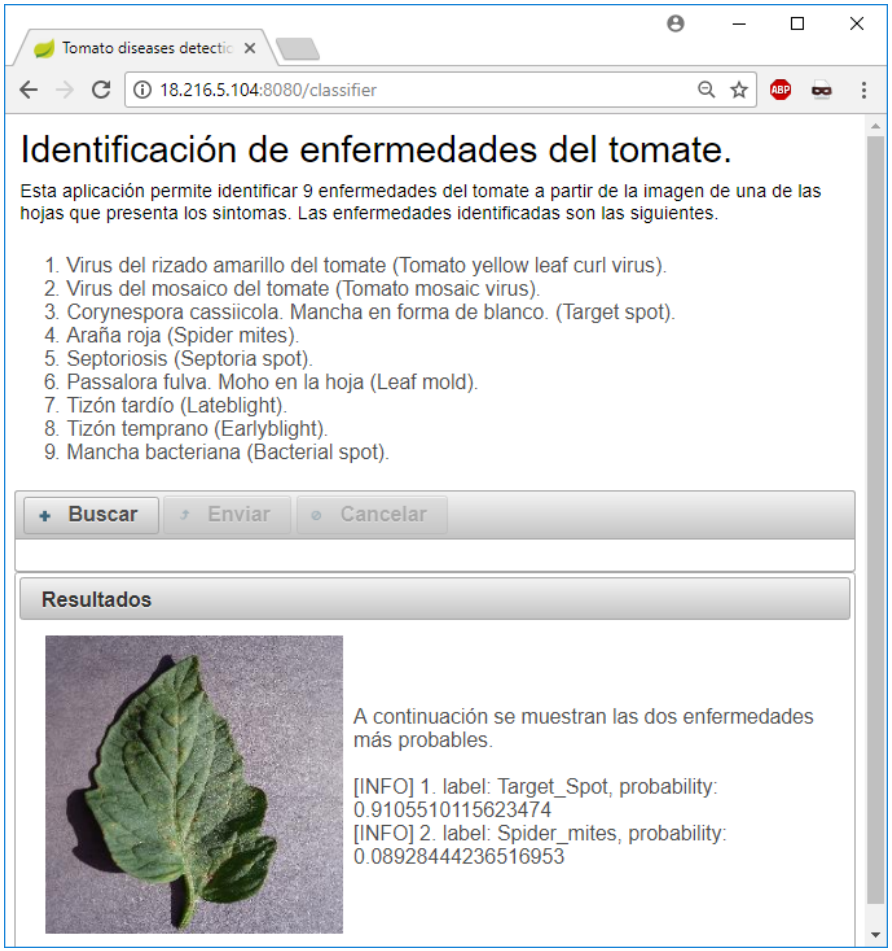

Figura 5.2. Diseño de la interfaz de usuario.

5. Aplicación Web.

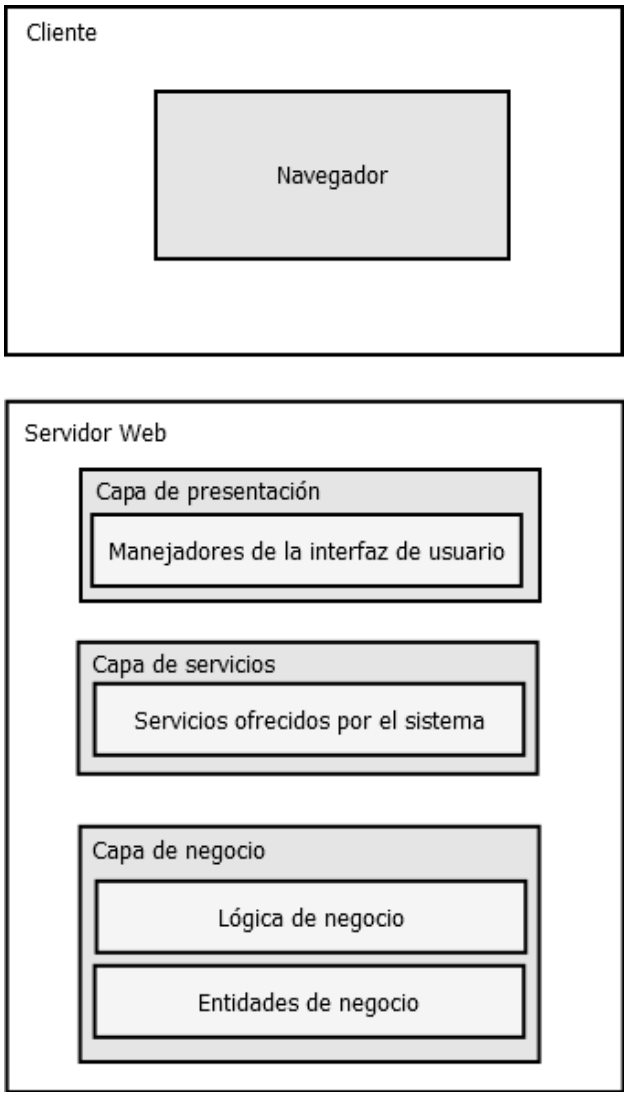

Figura 5.3. Arquitectura de la aplicación Web.

5. Aplicación Web.

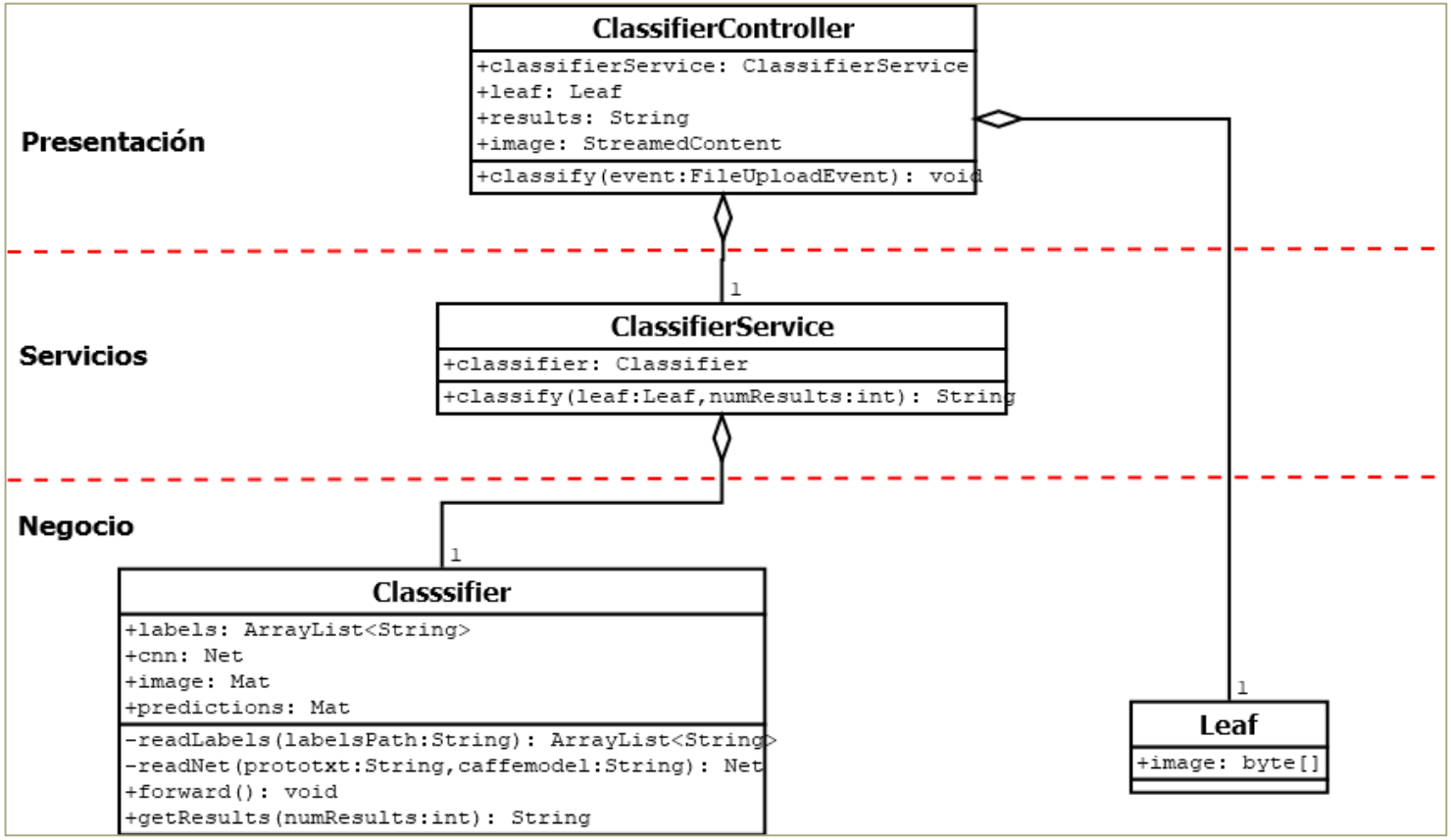

Figura 5.4. Diagrama de clases de la aplicación Web.

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.

Pruebas.

Número de prueba	Requisito	Descripción de la prueba	Resultado extensión esperado
1	RF01	Seleccionar una imagen del sistema de archivos local del dispositivo.	Es posible seleccionar imágenes con *.jpg, *.png y *.bmp.
2	RF01	Seleccionar un archivo con extensión distinta a las permitidas.	No es posible seleccionar archivos con extensión distinta a las permitidas.
3	RF02	Seleccionar una imagen cuya clase sea previamente conocida (etiquetada) y clasificarla.	El sistema clasifica la imagen en alguna de las nueve enfermedades o en hoja sana.
4	RF02	Seleccionar una imagen que no corresponda con ninguna de las diez clases válidas.	El sistema debe indicar que la imagen no corresponde a ninguna de las enfermedades identificables u a una hoja sana.
5	RNF01	Medir la eficiencia de clasificación con imágenes del conjunto de prueba.	La eficiencia de clasificación es superior al 90%.
6	RF02	Medir el tiempo de respuesta del sistema Web.	El tiempo promedio de clasificación es menor a cinco segundos, para al menos 10% de imágenes del conjunto de prueba.

6. Resultados.

Resultados.

Número de prueba	Resultado	Observaciones
1	Se permitió la selección de archivos extensiones *.jpg, *.png y *.bmp.	
2	No se permitió la selección de archivos con extensión distinta a las permitidas.	
3	El sistema clasificó la imagen de forma correcta y mostró los dos resultados más probables.	
4	El sistema clasifica algunas de las imágenes en clases válidas.	Se debería agregar al clasificador una clase específica para aquellas imágenes no reconocidas.
5	La prueba de eficiencia (98.82%) fue presentada al final de la sección 5.3.	
6	El tiempo promedio de clasificación con imágenes del subconjunto de pruebas fue de 582 ms.	Solo el proceso de clasificación tarda en promedio 115 ms. El resto consiste en la transmisión de datos entre cliente y servidor.

7. Conclusiones.

En comparación con otros trabajos en el área de detección de enfermedades en plantas (tabla inferior [4]), la metodología seguida en este prototipo obtuvo mejores resultados, teniendo en también en cuenta el número de enfermedades detectables (nueve).

Estudio	Descriptor	Clasificador	Conjunto de datos	Precisión
Prasad, Peddoju y Ghosh 2016 [5]	GWT, GLCM	KNN	297 imágenes, 5 enfermedades	93.00
Mokhtar et al. 2015 [6]	GLCM	SVM	800 imágenes, 2 enfermedades	99.80
Semary et al. 2015 [7]	Momentos de color, GLCM, descomposición wavelet	SVM	177 imágenes, 12 enfermedades	92.00
Dandawate y Kokare 2015 [8]	SIFT	SVM	120 imágenes, 2 enfermedades	93.79
Raza et al. 2015 [9]	Estadísticas locales, globales	SVM	71 imágenes, 2 enfermedades	89.93

8. Trabajo a futuro.

- Probar y comparar arquitecturas distintas de CNN en busca de una mayor eficiencia de clasificación.
- Ampliar el número de plantas y enfermedades identificables.
- Usar las imágenes que los usuarios proporcionan para incrementar el tamaño del conjunto de datos, con el objetivo de mejorar la eficiencia de clasificación.
- Desarrollo de una aplicación móvil.

9. Referencias

[1] SAGARPA. Hidroponía rústica. [En línea] Disponible en: http://www.sagarpa.gob.mx
[2] Research and Markets. Global Hydroponics Market - Forecasts from 2017 to 2022. [En línea] Disponible en: https://www.researchandmarkets.com.
[3] Sánchez F. Entrevista con Félipe Sanchez del Castillo, investigador de la Universidad Autónoma Chapingo. Recuperado de http://www.2000agro.com.mx.
[4] Brahimi M. et al. Deep Learning for Tomato Disease: Classification and Symtoms Visualization. 2017.
[5] Prasad S. et al. Multi-resolution mobile vision system for plant leaf disease diagnosis. 2015.
[6] Mokhtar U. et al. SVM-Based detection of tomato leaves diseases. 2015.
[7] Semary N. et al. Fruit-based tomato grading system using features fusión and support vector machine. 2015.
[8] Dandawate Y., Kokare R. An automated approach for classification of plant diseases towards development of futuristic decision support system in Indian perspective 2015.
[9] Raza S. et al. Automatic detection of diseased tomato plants using thermal and stereo visible light images. 2015

Gracias