



Universidad de Guadalajara
Centro universitario de ciencias exactas e ingenierías



Docente: Michel Emanuel López Franco.

Programación para internet.

Reporte final Sistema de Riego Inteligente + Flora-Net



Integrantes del equipo:

Miriam Paola Rodríguez Martín
Maritza Medina Hernández

21 de Mayo del 2024

1. Definición del Proyecto

Introducción

En el vasto mundo de la botánica, la diversidad de flores es asombrosa y encantadora. Cada variedad lleva consigo una historia única, características distintivas y un atractivo estético que las distingue. Sin embargo, para los entusiastas de la naturaleza y los amantes de las flores, identificar y clasificar estas maravillas botánicas puede resultar todo un desafío.

Es en este contexto donde la convergencia entre la inteligencia artificial y la pasión por la botánica da lugar a una fascinante empresa: la creación de un clasificador de flores basado en IA. Este proyecto tiene como objetivo principal proporcionar una herramienta innovadora y práctica que permita a los usuarios identificar fácilmente diferentes tipos de flores a través de imágenes, aprovechando las capacidades de aprendizaje automático y las tecnologías web modernas.

En este proyecto, creamos una aplicación web que utiliza inteligencia artificial para clasificar diferentes tipos de flores basándose en imágenes proporcionadas por el usuario. Además de integrar un sistema de riego a la inteligencia artificial, el cual ayudará al usuario a regar su planta de acuerdo con la humedad de cada planta.

Descripción

"FloraNet" es un nombre atractivo y descriptivo que combina "Flora", que se refiere a la vegetación de una región específica o en este caso, las flores, con "Net", que puede asociarse con redes neuronales o redes de computadoras. Aquí hay algunas razones por las que este nombre podría ser adecuado para un proyecto de clasificación de flores con inteligencia artificial. Se integró , el Sistema de Riego Inteligente propuesto utiliza tecnología de inteligencia artificial para determinar el tiempo de riego adecuado para diferentes variedades de flores de jardín. La idea principal es crear un sistema que utilice una tarjeta Arduino para controlar el riego de las plantas de manera automatizada.

Objetivo

Desarrollar un sistema de inteligencia artificial capaz de detectar y clasificar diferentes tipos de flores, medir la temperatura del suelo y activar el riego de manera automatizada y eficiente, optimizando el cuidado de las plantas. Este proyecto es relevante tanto desde un punto de vista tecnológico, al integrar tecnologías avanzadas de IA y sensores, como desde una perspectiva social, al promover prácticas de jardinería y agricultura más sostenibles y precisas.

Este objetivo subraya la combinación de tecnologías innovadoras para resolver problemas prácticos, beneficiando tanto a los jardineros y agricultores como al medio ambiente.

Funcionamiento:

- **Captura de Imágenes:** Se introduce una imagen de la flor a regar.
- **Procesamiento de Imágenes:** La IA integrada en el sistema procesa las imágenes para identificar la variedad de la flor, utilizando un modelo de clasificación previamente entrenado.
- **Medición de la Humedad del Suelo:** Mientras se procesan las imágenes, el sensor de humedad del suelo continúa monitoreando la humedad del suelo alrededor de las plantas.
- **Determinación del Tiempo de Riego:** La IA, basada en la variedad de la flor identificada y la humedad del suelo, determina el tiempo de riego óptimo para la planta en cuestión.
- **Control de Riego:** El Arduino activa los actuadores de riego durante el tiempo determinado, asegurando que las plantas reciban la cantidad adecuada de agua.

Beneficios:

- **Eficiencia de Riego:** El sistema proporciona la cantidad justa de agua requerida por cada planta, evitando el desperdicio y garantizando un crecimiento saludable.
- **Automatización:** La automatización del riego reduce la necesidad de intervención humana, lo que resulta en un mantenimiento más fácil y conveniente del jardín.
- **Personalización:** La capacidad de identificar diferentes variedades de plantas permite adaptar el riego a las necesidades específicas de cada especie. **Clasificando** imagen por medio de la IA y además decide si se debe o no regar la tierra, dependiendo de la humedad obtenida con el Arduino y el sensor de humedad

El Desafío Botánico: Clasificación de flores con IA

El núcleo de nuestro proyecto reside en la capacidad de un algoritmo de inteligencia artificial para analizar y reconocer patrones en imágenes de flores, permitiendo así su clasificación precisa en categorías predefinidas. Nos enfrentamos al desafío de entrenar un modelo que pueda distinguir entre cinco tipos populares de flores: Margarita, Diente de león, Rosa, Tulipán y Girasol. Para ello, nos apoyaremos en bibliotecas de Python ampliamente reconocidas en el campo del aprendizaje automático, como TensorFlow y scikit-learn.

La alianza entre tecnología y diseño: Creando una interfaz intuitiva

Entendemos que la usabilidad y la estética son elementos cruciales para garantizar una experiencia satisfactoria para el usuario. Por lo tanto, además de desarrollar un potente clasificador de flores, nos comprometemos a diseñar una interfaz de usuario atractiva y fácil de usar. Utilizando HTML, CSS y JavaScript, crearemos una plataforma web que permita a los usuarios cargar imágenes de flores, recibir una clasificación precisa y visualizar el porcentaje de confianza asociado a cada predicción.

Modelo de IA con TensorFlow

- Utilizamos TensorFlow.js, para crear y desplegar nuestro modelo de IA directamente en el navegador web del usuario.
- Durante el proceso de entrenamiento, la red neuronal aprendió automáticamente a reconocer patrones y características distintivas en las imágenes de las flores, lo que le permite clasificar nuevas imágenes con una precisión razonable, por medio de etiquetas en las imágenes.
- Este modelo de IA es una red neuronal convolucional (CNN) previamente entrenada para la clasificación de imágenes de flores. El modelo ha sido entrenado utilizando un conjunto de datos de flores que contiene miles de imágenes de diferentes tipos de flores, cada una etiquetada con su clase correspondiente.

Componentes:

Sensor de Humedad del Suelo: Se utilizará un sensor de humedad del suelo para medir continuamente el nivel de humedad en la tierra alrededor de las plantas.

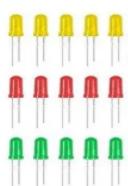
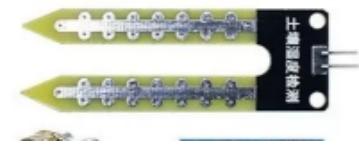
Tarjeta Arduino: Se utilizará una tarjeta Arduino como el cerebro del sistema para recopilar datos de los sensores, procesar la información y controlar el riego.

Actuadores de Riego.

2. Materiales Necessarios

Para la realización de este proyecto se usaron los siguientes materiales:

- Arduino UNO
- Sensor de humedad
- Leds
- Jumpers
- Resistencias
- LM393 comparador de voltaje



Disponibilidad y coste

Todos los materiales presentados son accesibles, ya que todos se encuentran en línea, los precios varían, pero los costos son accesibles a unos \$500 aproximadamente o menos, en total considerando las horas de trabajo, investigación y desarrollo del proyecto podemos aproximar un valor a la venta del proyecto de 16,875 pesos mexicanos.

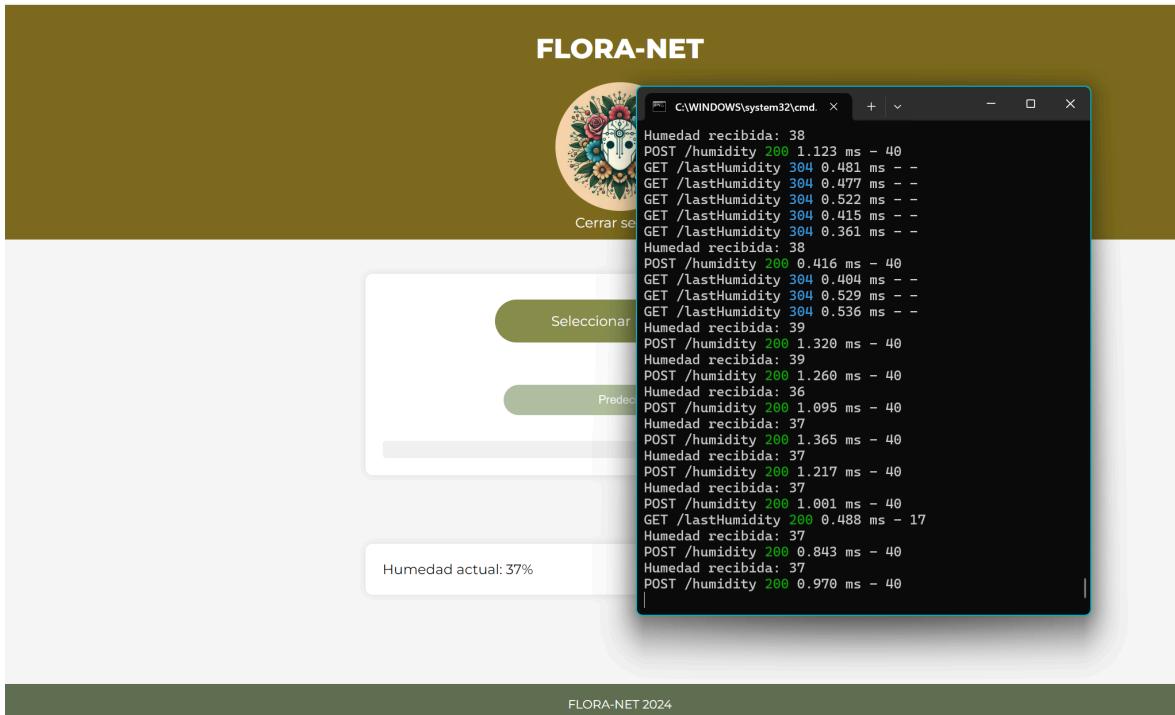
3. Configuración del Hardware

- Conexión y Configuración de Sensores**

Conectamos el sensor de humedad a la PC para obtener los datos de humedad de la tierra.



Datos obtenidos:



4. Almacenamiento de Datos

- Estructura de la Base de Datos**
- La base de datos está bien estructurada, incluye los datos sobre el registro de inicios de sesión, entre otras cosas.

The screenshot shows the MySQL Workbench interface. On the left, the 'Navigator' pane shows the 'SCHEMAS' section with 'datos' selected, and the 'Tables' section with 'usuarios' selected. The 'Columns' section lists 'id', 'email', 'password', and 'username'. The 'Script' tab contains SQL code for creating the table, inserting data, and altering it. The 'Result Grid' tab shows the data from the 'usuarios' table, which includes columns 'id', 'email', 'password', and 'username'. The 'Object Info' and 'Session' tabs are also visible at the bottom.

id	email	password	username
6	pao@gmail.com	12345	Paola
9	maritza@gmail.com	12345	Maritza
12	usuario@gmail.com	12345	usuario
13	usuario2@gmail.com	12345	usuario2
14	usuario3@gmail.com	12345	usuario3
15	usuario4@gmail.com	12345	usuario4
16	gerardo@gmail.com	12345	Gerardo
17	1@gmail.com	12345	1
18	2@gmail.com	2	2

- **Conexión y Almacenamiento de Datos**

Parte de código que almacena los datos correctamente en la base de datos.

```
// Configuración de La conexión a La base de datos
const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '*****', // Reemplaza 'tu_contraseña' con La contraseña de tu base de datos
  database: 'datos' // Reemplaza 'nombre_de_tu_base_de_datos' con el nombre de tu base de datos
});

app.use(session({
  secret: 'secreto', // Clave secreta para firmar la cookie de sesión
  resave: false,
  saveUninitialized: false
}));

// Conectar a La base de datos
db.connect((err) => {
  if (err) {
    throw err;
  }
  console.log('Conexión exitosa a la base de datos MySQL');
});
```

5. Desarrollo del Modelo de IA

- **Preparación de Datos**

- Los datos están correctamente limpiados y preparados para el entrenamiento del modelo.

```
[ ] import pathlib
dataset_url = "https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz"
data_dir = tf.keras.utils.get_file('flower_photos', origin=dataset_url, untar=True)
data_dir = pathlib.Path(data_dir)

↳ Downloading data from https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz
228813984/228813984 [=====] - 3s 0us/step

[ ] roses = list(data_dir.glob('roses/*'))
print(roses[0])
PIL.Image.open(str(roses[0]))
```



```
❶ img_height,img_width=180,180
batch_size=32
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="training",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)

↳ Found 3670 files belonging to 5 classes.
Using 2936 files for training.

[ ] val_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)

↳ Found 3670 files belonging to 5 classes.
Using 734 files for validation.
```

① 0 s se ejecutó 10:51 p.m.



- **Selección y Entrenamiento del Modelo**

- Se ha seleccionado un modelo de machine learning adecuado y se ha entrenado correctamente con datos históricos.

```
⌚ Epoch 1/10
/usr/local/lib/python3.10/dist-packages/keras/src/backend.py:5727: UserWarning: ``sparse_categorical_crossentropy` received `from_logits=True` , but the `output` argument is
    output, from_logits = _get_logits()
92/92 [=====] - 185s 1s/step - loss: 1.4158 - accuracy: 0.4176 - val_loss: 1.0981 - val_accuracy: 0.5545
Epoch 2/10
92/92 [=====] - 102s 1s/step - loss: 1.0495 - accuracy: 0.5739 - val_loss: 1.0549 - val_accuracy: 0.5899
Epoch 3/10
92/92 [=====] - 108s 1s/step - loss: 0.8776 - accuracy: 0.6689 - val_loss: 0.9303 - val_accuracy: 0.6403
Epoch 4/10
92/92 [=====] - 105s 1s/step - loss: 0.6565 - accuracy: 0.7609 - val_loss: 0.9106 - val_accuracy: 0.6703
Epoch 5/10
92/92 [=====] - 105s 1s/step - loss: 0.4829 - accuracy: 0.8236 - val_loss: 1.0284 - val_accuracy: 0.6512
Epoch 6/10
92/92 [=====] - 106s 1s/step - loss: 0.2691 - accuracy: 0.9087 - val_loss: 1.2230 - val_accuracy: 0.6444
Epoch 7/10
92/92 [=====] - 103s 1s/step - loss: 0.1770 - accuracy: 0.9445 - val_loss: 1.2991 - val_accuracy: 0.6444
Epoch 8/10
92/92 [=====] - 112s 1s/step - loss: 0.1184 - accuracy: 0.9653 - val_loss: 1.5079 - val_accuracy: 0.6294
Epoch 9/10
92/92 [=====] - 104s 1s/step - loss: 0.0798 - accuracy: 0.9802 - val_loss: 1.5265 - val_accuracy: 0.6553
Epoch 10/10
92/92 [=====] - 101s 1s/step - loss: 0.0848 - accuracy: 0.9792 - val_loss: 1.5955 - val_accuracy: 0.6703
```

- **Validación y Ajuste del Modelo**

- El modelo ha sido validado y ajustado adecuadamente para mejorar su precisión.

```
▶ num_classes = 5

model = Sequential([
    layers.Lambda(lambda x: x / 255.0, input_shape=(img_height, img_width, 3)),
    layers.Conv2D(6, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(num_classes,activation='softmax')
])

[ ] model.compile(optimizer='adam',
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])

[ ] epochs=10
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs
)
```

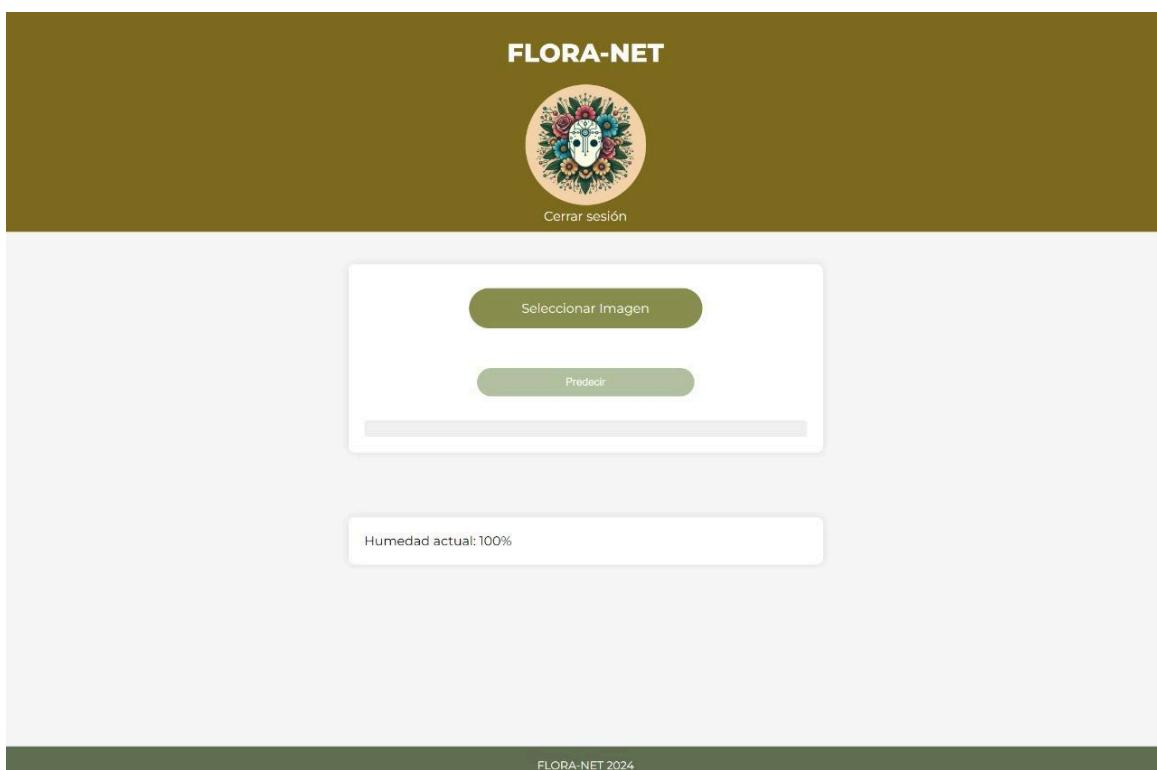
6. Desarrollo de la Interfaz de Usuario

• Diseño de la Interfaz

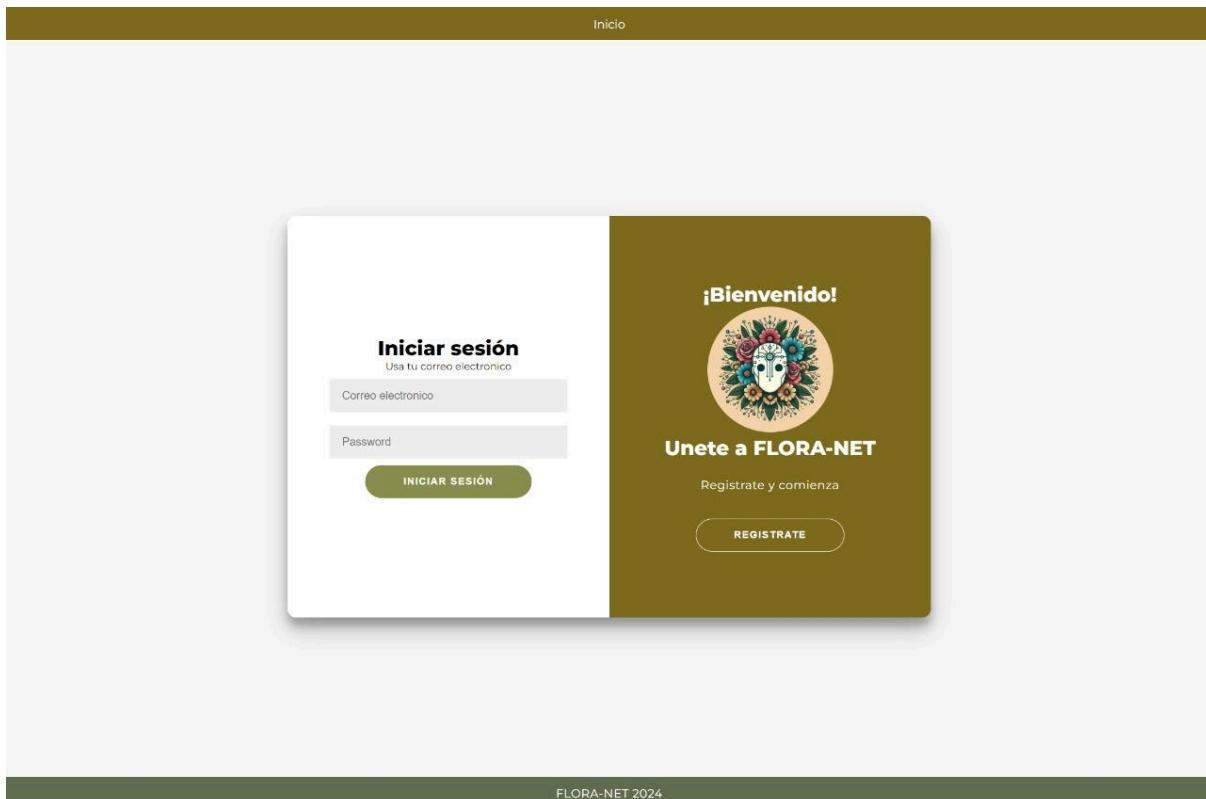
Página de inicio:



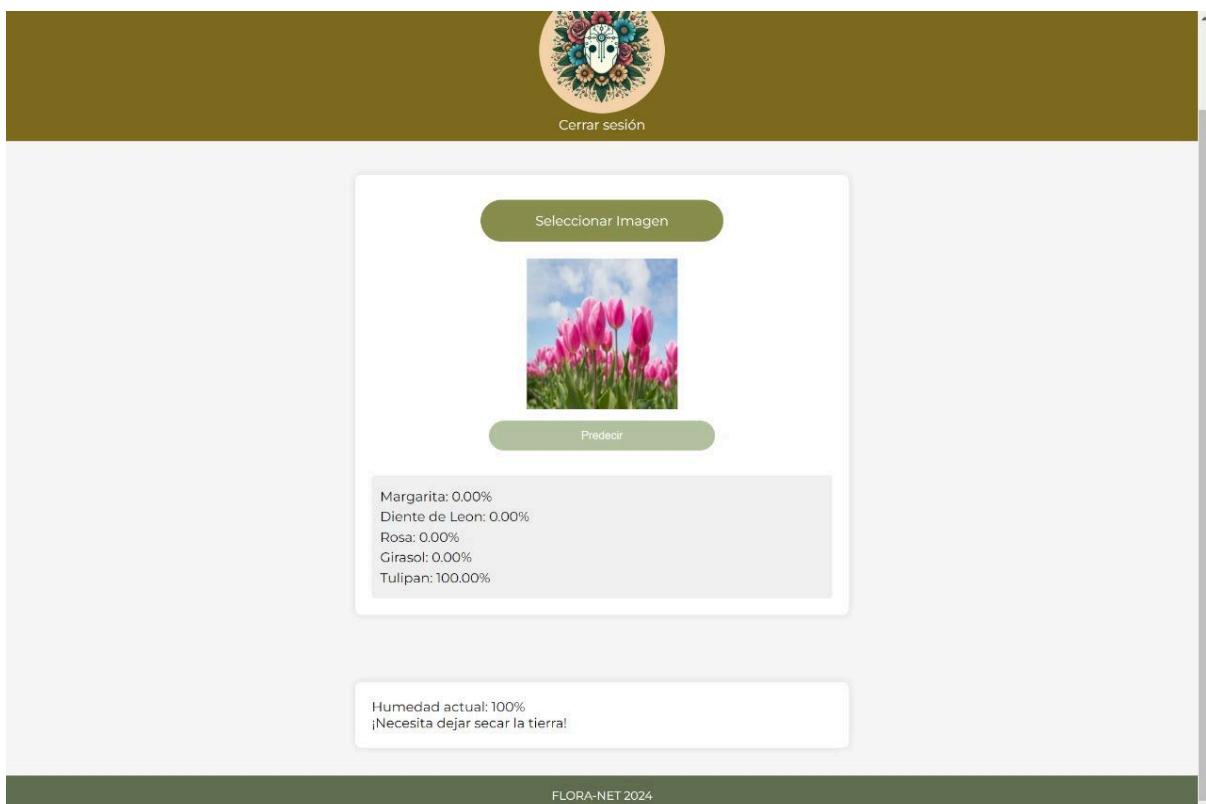
Predicción de clasificación de flor y detección de la humedad en tiempo real:



Log-in:

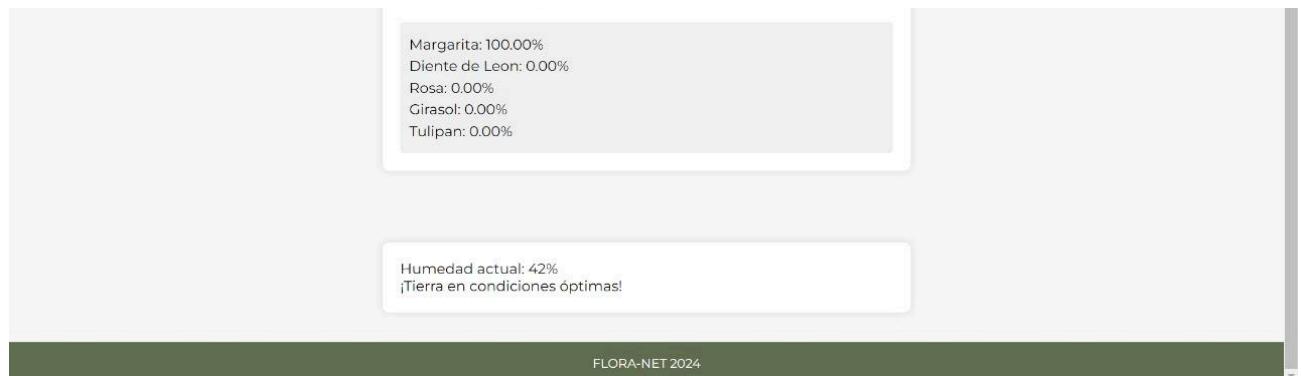


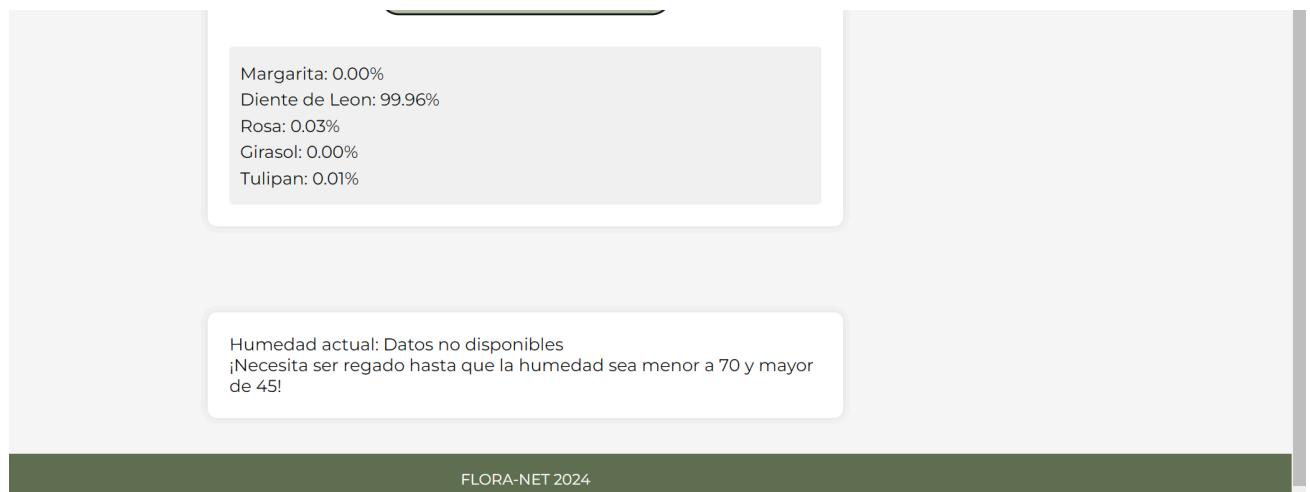
Recomendación de riego según la humedad:



7. Pruebas y Validación

Cada que el sensor de humedad se mueve de ambiente cambia también su medición y el comportamiento de la página web:





8. Documentación y Presentación

- Documentación Técnica

Diagrama de la arquitectura de la aplicación:

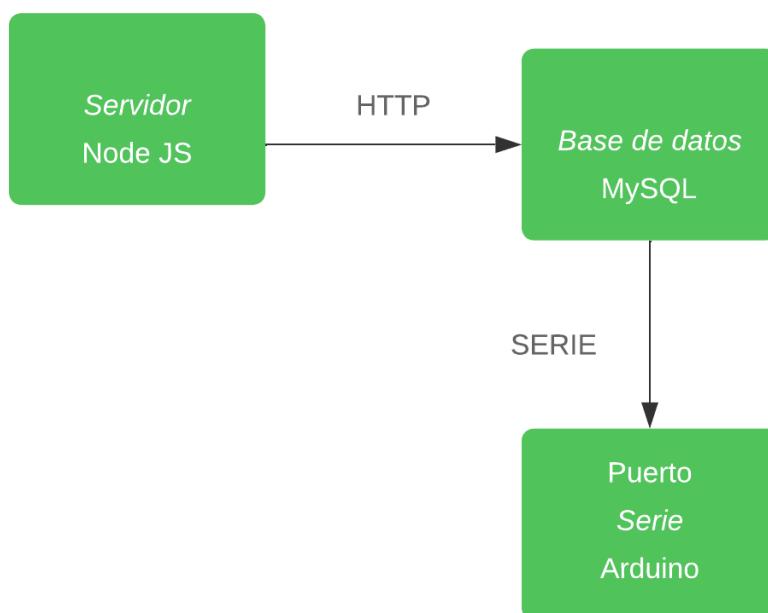


Diagrama de flujo de datos

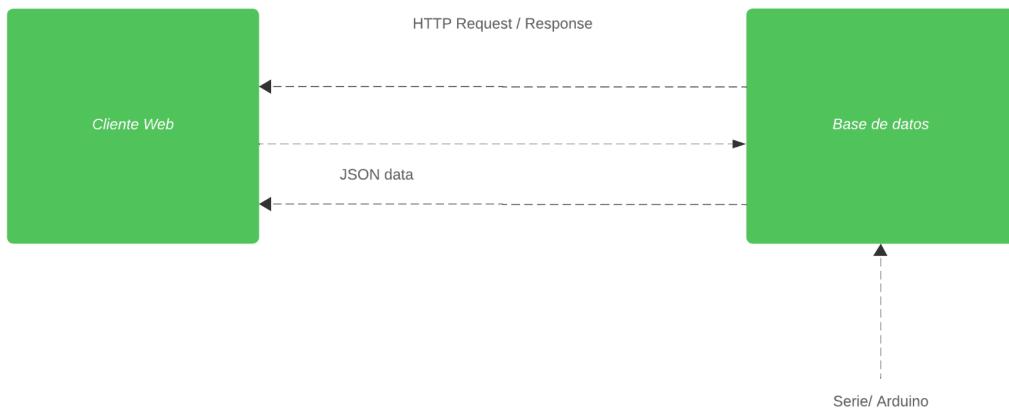
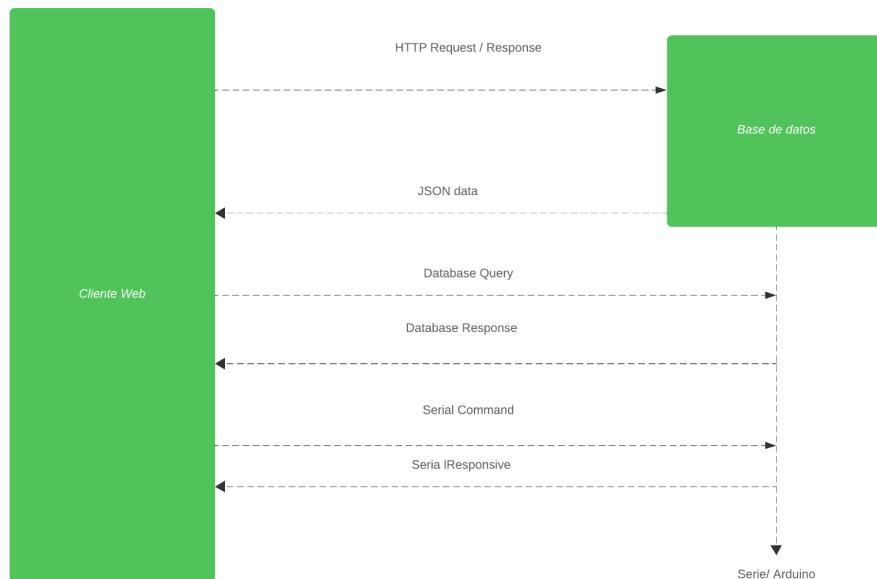


Diagrama de secuencia



- **Presentación**

<https://docs.google.com/presentation/d/1ucUN9id6HCt3ljY9Qh4DOXsEI0Htjm07V4lvqJQSIVg/edit?usp=sharing>