##### Informe del Proyecto Final: Experimentación con Microservicios Web Usando Flask y Python

Daniel Andres Marquez Torrez

Carlos Steban Enciso Caballero

Asignatura: Programación Avanzada

Docente: Ing. Jonathan Torres, PhD

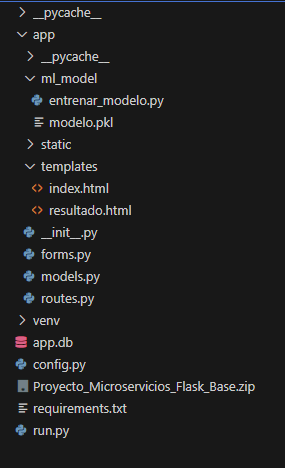
Fecha: [07/07/2025]

# 1. Introducción

Durante el desarrollo del curso de Programación Avanzada, se abordaron diferentes conceptos esenciales para el desarrollo de aplicaciones modernas, entre ellos: persistencia de datos, concurrencia, comunicación en red y desarrollo web con microservicios.  
  
En el presente informe se documenta la implementación y mejora de una aplicación web usando Flask. El objetivo del proyecto fue aplicar al menos dos temáticas vistas en clase, concretamente:  
  
- Uso de un modelo de machine learning entrenado con scikit-learn.  
- Persistencia de datos mediante base de datos SQLite usando SQLAlchemy.  
  
La aplicación resultante permite realizar predicciones sobre especies de flores Iris mediante un formulario web. Los resultados se almacenan en una base de datos y se pueden consultar desde una interfaz o una API REST.

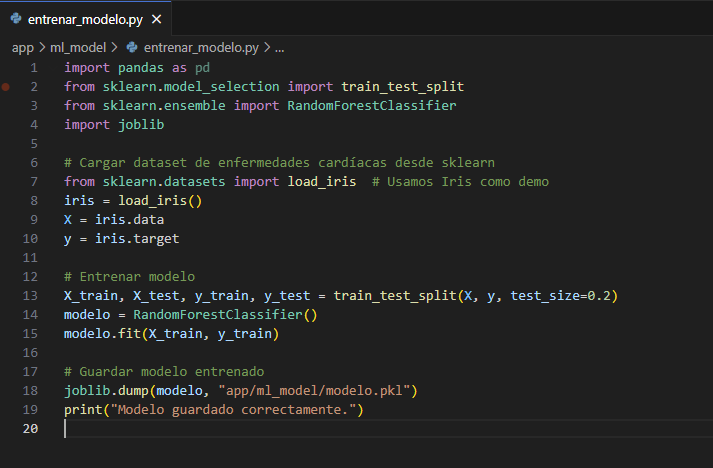
# 2. Descripción de la Solución

La solución se basa en el microframework Flask y está compuesta por los siguientes elementos:  
  
- Modelo ML entrenado y serializado con joblib.  
- Formularios WTForms para la entrada de datos.  
- Persistencia de resultados con SQLAlchemy y SQLite.  
- Interfaces HTML con Bootstrap.  
- API REST que permite consultar y crear predicciones vía JSON.

**> [CAPTURA 1: estructura del proyecto en VS Code]**

* Esta captura muestra la organización del proyecto. Se observan las carpetas app/, que contiene el código principal; templates/, con los archivos HTML; ml\_model/, que almacena el modelo entrenado; y archivos como run.py y config.py. Sirve para comprender cómo se estructura una aplicación Flask bien organizada.

**> [CAPTURA 2: código de entrenamiento del modelo]**

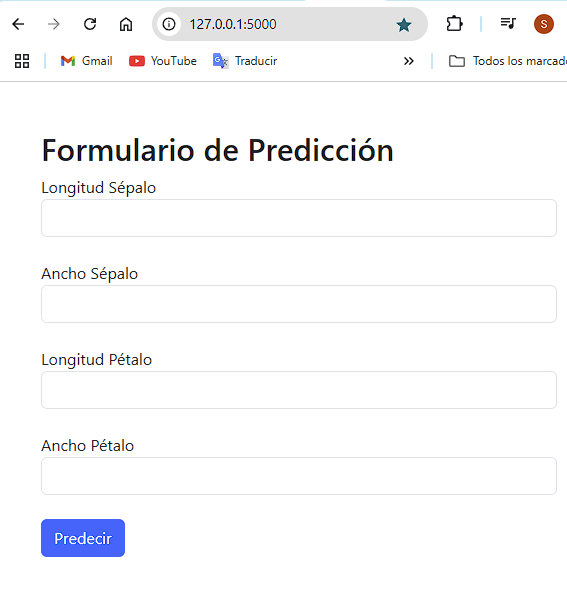


Aquí se visualiza el archivo entrenar\_modelo.py en VS Code. Muestra cómo se importan los datos, se entrena el modelo y se guarda con joblib.dump. Es clave para entender la base predictiva del proyecto.

Interfaz Web con Flask y Bootstrap

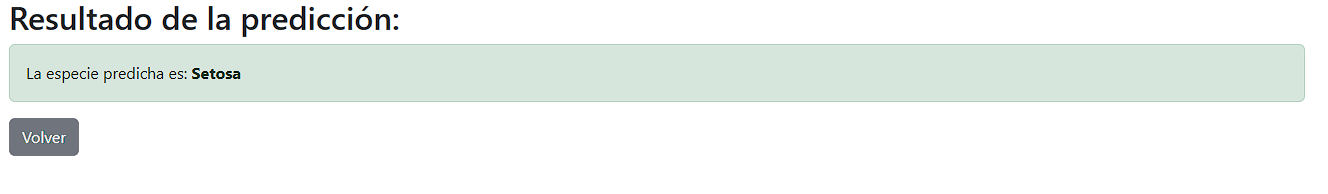
Se desarrolló una interfaz básica donde el usuario puede ingresar los datos requeridos (longitud y ancho de sépalo y pétalo). Al enviar el formulario, se ejecuta la predicción y se muestra el resultado.

**> [CAPTURA 3: formulario cargado en el navegador]**



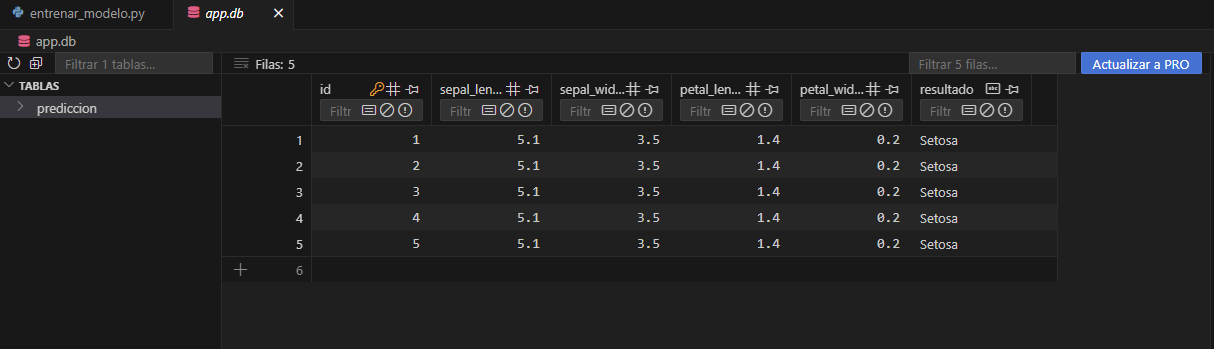
* Esta imagen muestra el formulario antes de ser enviado. Contiene campos numéricos para los cuatro parámetros requeridos. Refleja el punto de entrada de datos para la predicción.

**> [CAPTURA 4: resultado de la predicción mostrado en pantalla]**



Una vez enviado el formulario, esta pantalla muestra la especie predicha. Es resultado de la ejecución del modelo con los datos proporcionados por el usuario.

**> [CAPTURA 5: base de datos abierta en SQLite Browser o VS Code]**



**Almacenamiento de resultados**

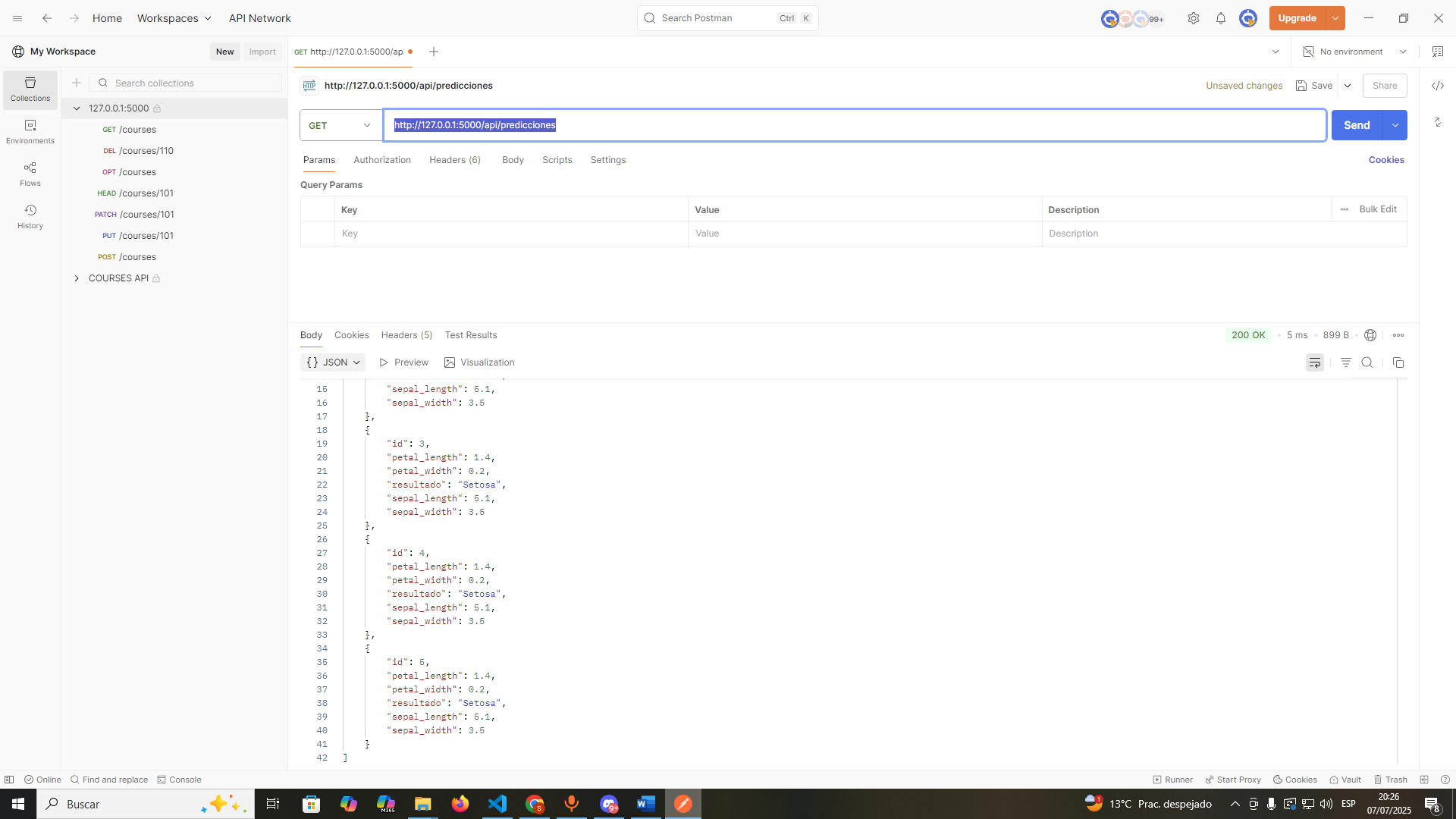
Cada predicción se guarda en la base de datos app.db usando SQLAlchemy. La tabla prediccion almacena las medidas ingresadas y el resultado.

**API REST**

La aplicación expone dos rutas:

* GET /api/predicciones: devuelve todas las predicciones en formato JSON.
* POST /api/predicciones: permite crear una nueva predicción desde una petición JSON.

**[**\*\*[**CAPTURA 6: prueba en Postman del endpoint GET y POST** Esta captura está dividida en dos pruebas que se deben realizar usando la herramienta Postman, que permite enviar peticiones HTTP a la API de la aplicación sin necesidad de una interfaz gráfica. A continuación se explican los pasos específicos para cada una:



|  |
| --- |
| [      {          "id": 1,          "petal\_length": 1.4,          "petal\_width": 0.2,          "resultado": "Setosa",          "sepal\_length": 5.1,          "sepal\_width": 3.5      },      {          "id": 2,          "petal\_length": 1.4,          "petal\_width": 0.2,          "resultado": "Setosa",          "sepal\_length": 5.1,          "sepal\_width": 3.5      },      {          "id": 3,          "petal\_length": 1.4,          "petal\_width": 0.2,          "resultado": "Setosa",          "sepal\_length": 5.1,          "sepal\_width": 3.5      },      {          "id": 4,          "petal\_length": 1.4,          "petal\_width": 0.2,          "resultado": "Setosa",          "sepal\_length": 5.1,          "sepal\_width": 3.5      },      {          "id": 5,          "petal\_length": 1.4,          "petal\_width": 0.2,          "resultado": "Setosa",          "sepal\_length": 5.1,          "sepal\_width": 3.5      }  ] |

> [CAPTURA 6: prueba en Postman del endpoint GET y POST]

> Comentario: ver descripción en el informe original.

> [CAPTURA 7: tabla del historial visible en el navegador]

### 4. Puntos de Discusión

* La importancia del contexto de aplicación en Flask fue evidente al trabajar con la base de datos. Inicialmente, se cometieron errores por ejecutar operaciones de SQLAlchemy fuera del contexto.
* El uso de plantillas Jinja2 fue una experiencia útil para entender cómo separar lógica de presentación.
* Al trabajar con formularios WTForms, surgieron retos con validaciones y compatibilidad entre campos numéricos y los valores que el modelo espera.
* El entrenamiento y carga del modelo scikit-learn mostró cómo se puede reutilizar ML en entornos de producción.
* Las pruebas con la API REST permitieron experimentar con herramientas como Postman y mejorar la comprensión del formato JSON y los métodos HTTP.

**5. Conclusiones**

El desarrollo de este proyecto me permitió poner en práctica múltiples conocimientos adquiridos a lo largo del curso, integrando machine learning, bases de datos y desarrollo web. A través de la implementación, enfrenté y resolví problemas reales como errores de contexto, rutas no encontradas y errores de plantilla, lo que reforzó mi aprendizaje.

El uso de Flask demostró ser una herramienta versátil y adecuada para proyectos que requieren prototipado rápido y escalabilidad. La combinación con herramientas como SQLAlchemy, Jinja2 y scikit-learn ofreció una visión integral del desarrollo moderno con Python.

Considero que esta experiencia aporta una base sólida para futuros proyectos más complejos, incluyendo el uso de microservicios reales, autenticación de usuarios y despliegue en la nube.

# 3. Reflexión Técnica

Este proyecto permitió consolidar el uso práctico de varios conceptos fundamentales del desarrollo moderno con Python:  
  
- El uso de Flask como microframework es ágil y potente, especialmente para servicios REST y prototipos rápidos.  
- La integración entre ML y aplicaciones web es directa cuando se entiende el flujo de datos.  
- Usar SQLAlchemy simplifica el acceso a base de datos relacionales sin escribir SQL directamente.  
- Manejar errores como TemplateNotFound o Working outside of application context me ayudó a entender la importancia de la estructura del proyecto y el contexto de la aplicación.  
  
Aunque el proyecto fue funcional, se podrían agregar mejoras como:  
- Autenticación de usuarios (por ejemplo, con Flask-Login).  
- Validaciones más robustas en el formulario.  
- Visualización de gráficas con Chart.js.

# 4. Enlaces de Entrega

- Repositorio código fuente: [Pegar enlace aquí]

- Video explicativo (2 min): [Pegar enlace aquí]

# 5. Referencias

- Flask: https://flask.palletsprojects.com/  
- SQLAlchemy: https://docs.sqlalchemy.org/  
- Scikit-learn: https://scikit-learn.org/  
- Curso Programación Avanzada, Universidad Distrital