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

**Daniel Andres Marquez Torrez**

**Cod.: 20222020238**

**Carlos Steban Enciso Caballero**

**Cod.: 20222020249**

**Asignatura: Programación Avanzada**

**Docente: Ing. Jonathan Torres, PhD**

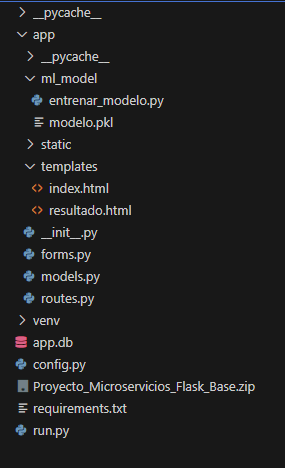
**Fecha: 07/07/2025**

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.

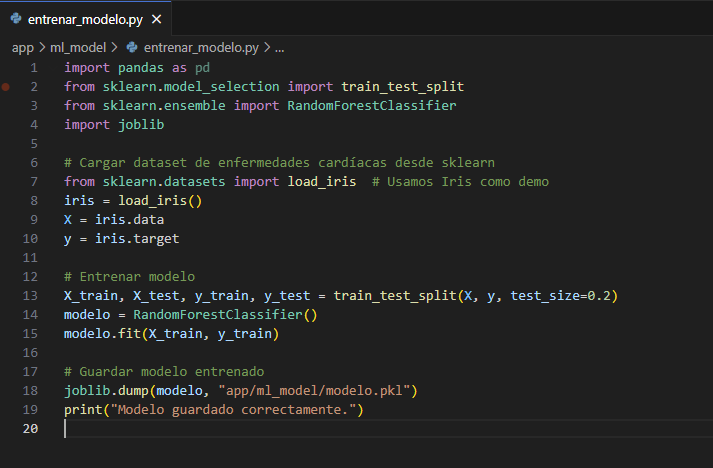
**2. Descripción de la Solución**

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**

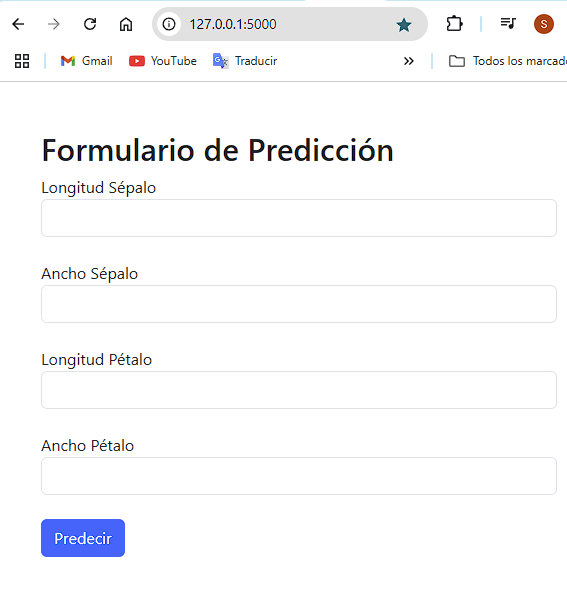


Aquí se visualiza el archivo entrenar\_modelo.py en VS Code. Muestra cómo se importan los datos, se entrena el modelo y se guarda con joblib.dump. Es clave para entender la base predictiva del proyecto.

Interfaz Web con Flask y Bootstrap

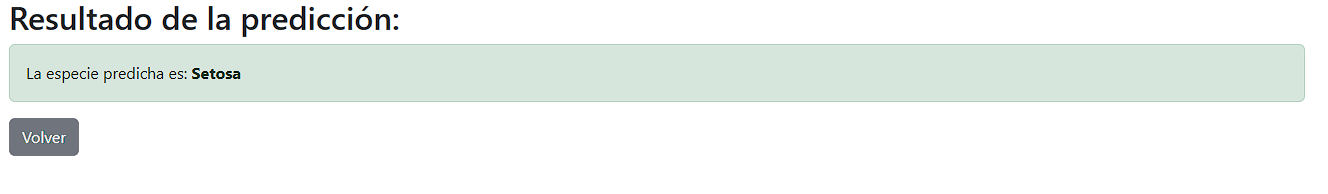
Se desarrolló una interfaz básica donde el usuario puede ingresar los datos requeridos (longitud y ancho de sépalo y pétalo). Al enviar el formulario, se ejecuta la predicción y se muestra el resultado.

**CAPTURA 3: formulario cargado en el navegador**



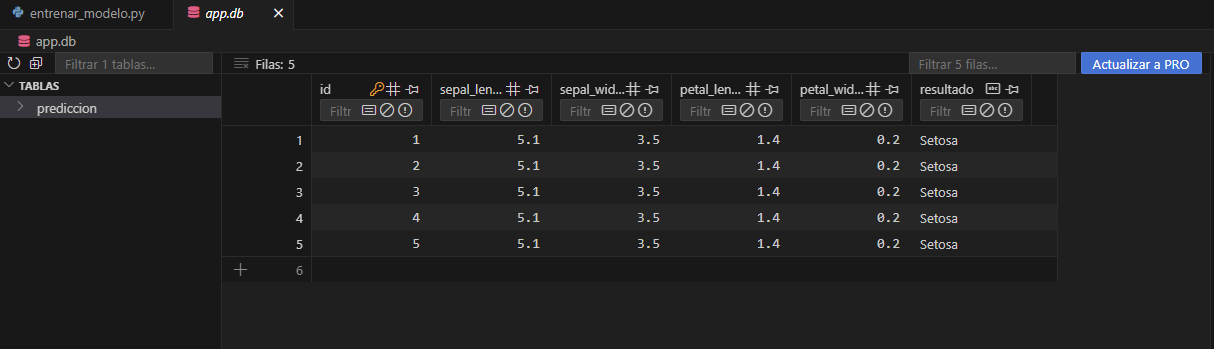
* Esta imagen muestra el formulario antes de ser enviado. Contiene campos numéricos para los cuatro parámetros requeridos. Refleja el punto de entrada de datos para la predicción.

**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**

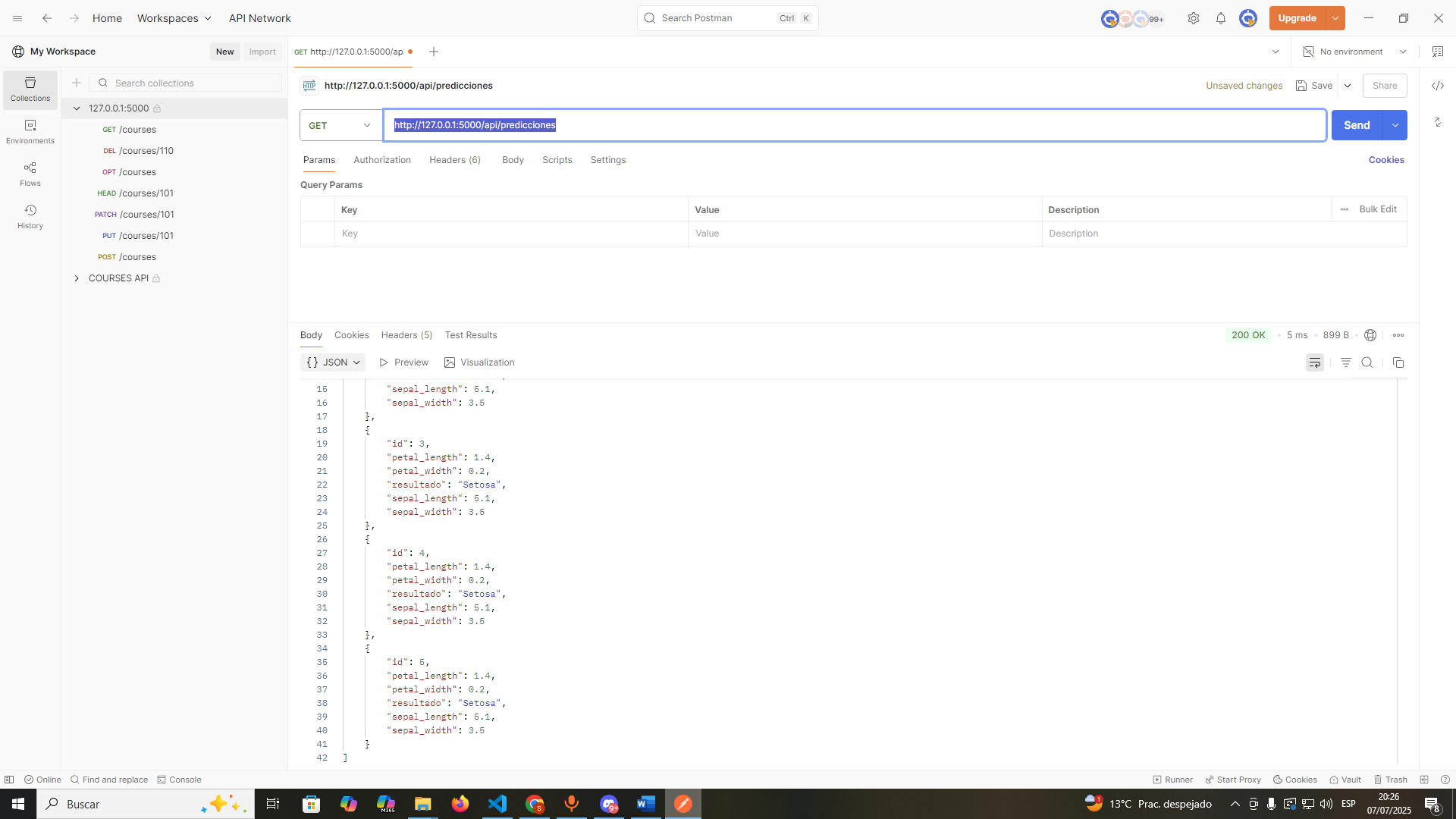
La aplicación expone dos rutas:

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

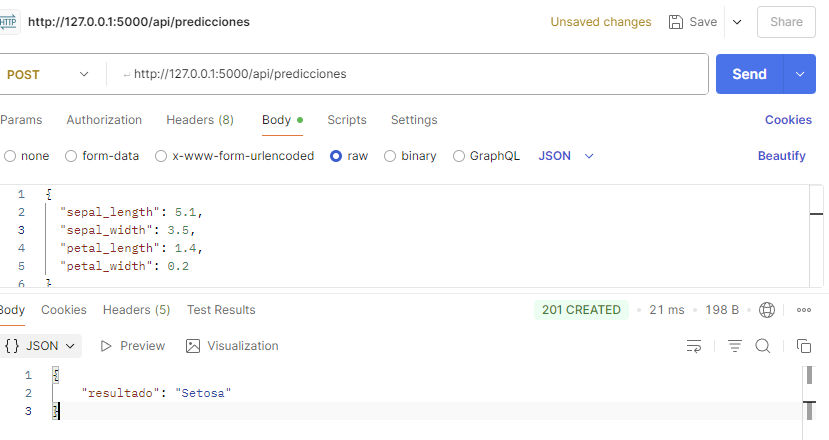
**CAPTURA 6: prueba en Postman del endpoint GET y POST**

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      }  ] |



**3. 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.

**4. Conclusiones**

Este proyecto nos ayudó muchísimo a reforzar lo que aprendimos durante el semestre. Pusimos en práctica temas como machine learning, manejo de bases de datos y desarrollo web con Flask, todo en un solo ejercicio que nos obligó a pensar y a solucionar errores reales que se presentan en el desarrollo de software.

Tuvimos que enfrentarnos a problemas como errores de contexto, rutas mal configuradas o plantillas que no se cargaban, y eso nos obligó a entender mejor cómo funciona Flask por dentro. Esas dificultades, aunque a veces frustrantes, fueron las que más nos enseñaron.

También confirmamos que Flask es una muy buena opción para proyectos pequeños o medianos, porque es flexible y se adapta fácil. Al unirlo con herramientas como SQLAlchemy, Jinja2 y scikit-learn, logramos construir una aplicación completa y funcional.

Sentimos que esta experiencia nos dio una base sólida para trabajar en proyectos más grandes a futuro. Ya no nos da miedo tocar cosas como APIs REST, formularios, modelos predictivos o manejar una base de datos. Incluso podríamos pensar en añadir más cosas como login de usuarios o subir este tipo de apps a la nube.

**5. Reflexión Técnica**

Este proyecto nos ayudó a aterrizar muchos de los conceptos que vimos a lo largo del curso. Al principio parecía complicado juntar cosas como modelos de machine learning con desarrollo web, pero poco a poco fuimos entendiendo cómo todo se conecta cuando se usan las herramientas adecuadas.

* Usar Flask nos pareció muy práctico, ya que no es pesado y permite construir aplicaciones web de manera sencilla, sobre todo si se quiere probar rápido una idea.
* Pudimos ver que integrar un modelo de machine learning entrenado con scikit-learn dentro de una app Flask es más fácil de lo que pensábamos, siempre y cuando se tenga claro cómo es el flujo de datos desde el formulario hasta la predicción.
* Trabajar con SQLAlchemy fue una gran ventaja, porque evitamos escribir consultas SQL manuales. Eso hizo que el código fuera más claro y más fácil de mantener.
* Durante el desarrollo nos enfrentamos a errores como TemplateNotFound o Working outside of application context. Aunque en el momento fue frustrante, al final nos ayudaron a entender lo importante que es tener bien estructurado el proyecto y entender cuándo y cómo se deben ejecutar ciertas partes del código.

A pesar de que el proyecto cumplió con los objetivos, sentimos que hay cosas que podríamos mejorar si tuviéramos más tiempo:

* Agregar un sistema de autenticación de usuarios con Flask-Login para que cada usuario pueda ver su historial.
* Hacer validaciones más completas en los formularios para evitar que el usuario meta datos incorrectos.
* Usar librerías como Chart.js para visualizar las predicciones en gráficos bonitos y fáciles de entender.

En resumen, fue un proyecto retador pero muy enriquecedor. Aprendimos haciendo, equivocándonos y corrigiendo, y eso nos dejó aprendizajes reales para el futuro.

**6. Enlaces de Entrega**

- <https://github.com/CAMUS3/PROYECTO-FINAL-P.A>

**7. Referencias**

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