Universidad Mariano Gálvez de Guatemala Facultad de Ingeniería de Sistemas de la Información

Semestre IV

Curso: Programación II

Ing. Juan Pablo Israel Escobar Castro

Logotipo

El contenido generado por IA puede ser incorrecto.

**Plataforma web de streaming con Flask**

Lesly Yuliza Amaya López 0909-24-1059

Douglas Lisandro Chamalé Sam 0909-24-4732

Kevin Ivan Martinez Cartagena 0909-23-16182

Jesus Rodrigo De Leon Tobar 0909-24-17053

Puerto Barrios, Izabal

Índice

[INTRODUCCIÓN 3](#_Toc212277236)

[OBJETIVOS 4](#_Toc212277237)

[JUSTIFICACIÓN 5](#_Toc212277238)

[Qué es Flask y sus características. 6](#_Toc212277239)

[Ventajas 8](#_Toc212277240)

[Desventajas 8](#_Toc212277241)

[Historia de Flask 9](#_Toc212277242)

[¿Flask es un frontend o un backend? 10](#_Toc212277243)

[¿Qué es un Framework Web? 10](#_Toc212277244)

[Preparación de Flask para su uso 10](#_Toc212277245)

[Ejecución de Flask en producción 13](#_Toc212277246)

[PROYECTO 14](#_Toc212277247)

[Descripción 14](#_Toc212277248)

[Características 14](#_Toc212277249)

[Tecnologías Utilizadas 15](#_Toc212277250)

[Instalación 15](#_Toc212277251)

[Ejecutar la aplicación: 16](#_Toc212277252)

[Detalles del Código 17](#_Toc212277253)

[Ejemplo de Diagrama de Flujo 19](#_Toc212277254)

[CONCLUSION 20](#_Toc212277255)

[REFERENCIAS BIBLIOGRAFICAS 21](#_Toc212277256)

# INTRODUCCIÓN

En la actualidad, las plataformas de streaming se han convertido en una de las principales formas de consumo de contenido digital, permitiendo a los usuarios acceder de manera rápida, sencilla y personalizada a películas, series y música desde cualquier dispositivo con conexión a internet.

El proyecto fue desarrollado utilizando el lenguaje de programación Python y el framework Flask, que permite crear aplicaciones web ligeras y eficientes mediante el uso de rutas, plantillas y conexión a bases de datos. A través de este trabajo se busca poner en práctica conceptos fundamentales como la estructura cliente-servidor, manejo de formularios, gestión de usuarios, plantillas dinámicas y almacenamiento de información en una base de datos.

La implementación de este proyecto no solo refuerza las habilidades técnicas en el desarrollo web, sino que también fomenta la comprensión del flujo de una aplicación completa, desde el diseño de la interfaz hasta la programación del backend. Con ello, se pretende demostrar la integración de herramientas modernas para construir soluciones tecnológicas útiles, escalables y orientadas a la experiencia del usuario.

# OBJETIVOS

**Objetivo General**

* Desarrollar una aplicación web de streaming funcional utilizando el framework Flask de Python

**Objetivos Específicos**

* Diseñar la estructura de la aplicación web con Flask.
* Gestionar listas de reproducción y contenido multimedia.
* Aplicar buenas prácticas de programación y organización de rutas.
* Realizar pruebas de funcionamiento y corrección de errores para asegurar la estabilidad del sistema.

# JUSTIFICACIÓN

La creación de una plataforma de streaming permite comprender la integración de bases de datos, manejo de sesiones y rutas en Flask, fortaleciendo los conocimientos en desarrollo web y aplicando conceptos de backend y frontend de manera práctica**.**

# Qué es Flask y sus características.

Flask es un framework web ligero de Python que permite crear aplicaciones web rápidamente y con un mínimo de código repetitivo. Admite enrutamiento, plantillas con Jinja2 y extensiones para mayor funcionalidad, lo que lo hace ideal para API y proyectos web pequeños y medianos.

Flask se conoce como un microframework, lo que significa que incluye solo las herramientas básicas necesarias para crear una aplicación web. Esto deja la mayoría de las decisiones arquitectónicas en manos del desarrollador, a diferencia de los frameworks full-stack como [Django](https://builtin.com/software-engineering-perspectives/django)  , que incluyen funciones y convenciones integradas.

* **Ligero y flexible:** A diferencia de frameworks más grandes como Django, Flask no impone un montón de dependencias o decisiones de arquitectura. Esto permite a los desarrolladores elegir las herramientas y bibliotecas que mejor se adapten a las necesidades específicas de su proyecto.
* **Servidor de desarrollo y depurador integrados:** Incluye un servidor web incorporado que facilita el desarrollo y las pruebas, además de un depurador para ayudar a encontrar y corregir errores.
* **Motor de plantillas Jinja2:** Utiliza Jinja2 para la creación de plantillas, lo que permite generar HTML dinámico de manera sencilla.
* **Despacho de solicitudes RESTful:** Facilita la creación de aplicaciones web y de API RESTful que pueden ser utilizadas por otras aplicaciones.
* **Manejo de cookies seguras:** Admite cookies seguras (sesiones del lado del cliente) para gestionar la información de la sesión de manera segura.
* **Extensible:** Ofrece una gran cantidad de extensiones (propias y de terceros) que se pueden agregar para ampliar su funcionalidad, por ejemplo, para integrar una base de datos, validar formularios o manejar la autenticación.
* **Pruebas unitarias:** Incluye soporte para pruebas, lo que permite a los desarrolladores escribir fácilmente pruebas unitarias y de integración.
* **Comunidad activa y buena documentación:** Cuenta con un ecosistema de soporte muy sólido, con una comunidad activa y una documentación completa que ayuda a los desarrolladores a resolver dudas y aprender rápidamente.
* **Basado en WSGI:** Cumple con la especificación WSGI (Web Server Gateway Interface), lo que le permite interactuar con una amplia variedad de servidores web.
* **Adecuado para múltiples proyectos:** Su flexibilidad lo hace ideal para una variedad de proyectos, desde pequeñas aplicaciones y prototipos hasta APIs y sistemas de nivel empresarial.

## Ventajas

* **Simplicidad.** Flask es fácil de instalar y configurar.
* **Flexibilidad.** El microframework permite a los desarrolladores elegir las tecnologías y herramientas que desean utilizar en sus proyectos.
* **Extensibilidad.** Flask permite ampliar la funcionalidad mediante complementos y módulos que se pueden integrar fácilmente en el proyecto.
* **Comunidad activa.** Flask es uno de los frameworks más utilizados para Python, por lo que cuenta con una gran comunidad de desarrolladores.

## Desventajas

* **Falta de soluciones listas para usar.** Inicialmente, Flask solo proporciona un conjunto mínimo de funcionalidades. Si un programador necesita funcionalidades más amplias, como autenticación de usuarios, tendrá que agregar bibliotecas adicionales o implementarlas por sí mismo.
* **Falta de soporte nativo para subprocesos múltiples.** Flask fue diseñado como un framework de un solo subproceso. Para manejar aplicaciones web de subprocesos múltiples, se deben instalar bibliotecas adicionales.
* **Limitaciones en la capacidad de escalar.** Si un proyecto comienza a crecer y volverse más complejo, puede haber dificultades para mantener la funcionalidad necesaria.

En resumen, Flask es ideal para proyectos pequeños, es perfecto para prototipos e ideas rápidas. Sin embargo, rara vez se utiliza en proyectos grandes y no es adecuado para la programación asíncrona.

## Historia de Flask

Armin Ronacher desarrolló Flask y lo lanzó en 2010. Lideró un grupo internacional de entusiastas de Python llamado Pocoo, que también creó el kit de herramientas Werkzeug WSGI y el motor de plantillas Jinja2 sobre el que se basa Flask.



Las tecnologías principales del marco Flask incluyen:

* [WSGI](https://wsgi.readthedocs.io/en/latest/) (Web Server Gateway Interface) es el estándar de desarrollo de aplicaciones web Python comúnmente utilizado que define las interacciones entre servidores web y aplicaciones web.
* [Werkzeug](https://werkzeug.palletsprojects.com/en/2.2.x/) es un kit de herramientas que utiliza WSGI para implementar, enviar solicitudes y manejar respuestas.
* J [inja2](https://jinja.palletsprojects.com/en/3.1.x/) es una plantilla web de Python que permite a los usuarios combinar una plantilla con una fuente de datos para generar páginas web dinámicas.

## ¿Flask es un frontend o un backend?

Flask es un framework para el backend. Está diseñado para gestionar el código de una aplicación que se ejecuta en el servidor, como la gestión de solicitudes, bases de datos y la creación de API. Flask puede servir archivos frontend al cliente, como HTML, CSS y JavaScript, pero su función principal no es construir y diseñar el frontend.

## ¿Qué es un Framework Web?

Website Application Framework, o simplemente "Web Framework", es un grupo de herramientas y módulos que permite a los desarrolladores de aplicaciones web crear aplicaciones sin preocuparse por cuestiones de bajo nivel como la gestión de subprocesos, protocolos, etc.

# Preparación de Flask para su uso

Paso a: Si Python aún no está allí, instálelo.

Python (especialmente Python 3.6+) ya debería estar instalado en tu ordenador antes de configurar Flask. Sigue estos pasos para comprobar si Python ya está instalado:

Dominio:

*python –version*

Paso b: Implementar un entorno virtual (no es obligatorio pero se recomienda).

Debes usar un entorno virtual para gestionar los recursos de tu proyecto Flask. Esto mantiene los requisitos de tu proyecto separados y evita que entren en conflicto con los de otros proyectos.

Dominio:

*pip install virtualenv*

Para crear un nuevo entorno virtual para su proyecto:

virtualenv venv

**Instalación de Flask**

Una vez que su entorno virtual esté en funcionamiento, puede instalar Flask usando pip, el administrador de paquetes de Python.

Dominio:

*pip install Flask*

Instalar la versión estable más reciente de Flask es fácil con este comando. De la misma manera, puedes especificar el número de versión de Flask si lo necesitas:

*pip install Flask==2.0.1*

**Verificación de la instalación del matraz**

Ejecute este comando para ver si Flask se instaló correctamente:

*python -m flask --version*

**Cree una aplicación Flask simple**

Creemos una aplicación sencilla para asegurarnos de que todo funcione ahora que Flask está instalado.

1. Crea un archivo Python llamado app.py para tu aplicación Flask:

*from flask import Flask*

*app = Flask(\_\_name\_\_)*

*@app.route('/')  
def hello\_world():  
return 'Hello, World!'*

*if \_\_name\_\_ == '\_\_main\_\_':  
app.run(debug=True)*

b. Ejecute su aplicación Flask:

* Vaya al directorio donde está almacenado su archivo app.py en su terminal.
* Utilice el siguiente comando para ejecutar el programa:

*python app.py*

c. Pruebe su aplicación:

*Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)*

# Ejecución de Flask en producción

Flask no debe entregarse con el servidor de desarrollo habitual en producción. En su lugar, debe entregarse con un servidor web listo para producción, como Gunicorn o uWSGI. Para instalar Gunicorn:

*pip install gunicorn*

*To run the app use this command:*

*gunicorn -w 4 app:app*

# PROYECTO

## Descripción

Mini Streaming es una aplicación web desarrollada con Flask que permite a los usuarios visualizar una colección de videos organizados en tres categorías: Flask, Python y Anime. Los videos están embebidos desde enlaces de Google Drive. Además, los usuarios pueden enviar valoraciones a través de un formulario, las cuales se almacenan en una base de datos SQLite. Los administradores pueden ver las valoraciones enviadas en una página dedicada.

El proyecto es una demostración de una aplicación web simple, ideal para aprender conceptos de desarrollo web con Flask, manejo de bases de datos SQLite y diseño de interfaces con HTML/CSS.

### Características

* Visualización de videos organizados por categorías (Flask, Python, Anime).
* Formulario para enviar valoraciones con nombre, calificación (0-5 estrellas) y comentarios.
* Página de administración para ver todas las valoraciones almacenadas.
* Interfaz responsiva con un diseño moderno y scroll horizontal para los videos.
* Base de datos SQLite para almacenar valoraciones.

## Tecnologías Utilizadas

* **Python 3.x**: Lenguaje principal.
* **Flask**: Framework web para la creación de rutas y manejo de solicitudes.
* **SQLite**: Base de datos ligera para almacenar valoraciones.
* **HTML/CSS**: Para la estructura y estilo de la interfaz.
* **Jinja2**: Motor de plantillas para renderizar contenido dinámico.

## Instalación

Sigue estos pasos para ejecutar el proyecto localmente:

git clone https://github.com/DLCS-001/proyecto-de-progra-2-con-flask.git

cd proyecto-de-progra-2-con-flask

Crear un entorno virtual (opcional, pero recomendado)

python -m venv venv

source venv/bin/activate # En Windows: venv\Scripts\activate

Instalar dependencias: Crea un archivo requirements.txt con lo siguiente

flask==3.0.3

Luego, instala las dependencias:

pip install -r requirements.txt

## Ejecutar la aplicación:

python app.py

La aplicación estará disponible en http://localhost:5000.

**Uso**

1. **Página principal (/)**:
   * Muestra videos organizados en tres categorías: Flask, Python y Anime.
   * Cada video se embebe desde Google Drive y se presenta en una tarjeta con título y descripción.
   * Incluye un enlace para acceder al formulario de valoración.
2. **Formulario de valoración (/formulario)**:
   * Los usuarios pueden ingresar su nombre, una calificación (0-5 estrellas) y un comentario sobre su experiencia.
   * Al enviar el formulario, se redirige a una página de agradecimiento (/gracias).
3. **Página de agradecimiento (/gracias)**:
   * Muestra un mensaje de confirmación tras enviar una valoración.
   * Incluye un enlace para volver a la página principal.
4. **Panel de administración (/admin/valoraciones)**:
   * Muestra una tabla con todas las valoraciones almacenadas (nombre, estrellas, comentario).
   * Accesible directamente sin autenticación (puede mejorarse con un sistema de login).

## Detalles del Código

**app.py**

* **Descripción**: Configura la aplicación Flask, define las rutas y maneja la lógica principal.
* **Rutas**:
  + /: Renderiza la página principal con videos organizados por categorías.
  + /formulario: Maneja el formulario de valoración (GET para mostrar, POST para procesar).
  + /gracias: Muestra un mensaje de agradecimiento tras enviar una valoración.
  + /admin/valoraciones: Muestra todas las valoraciones almacenadas.
* **Funciones clave**:
  + drive\_embed\_url: Convierte enlaces de Google Drive en URLs embebibles.
* **Estructura de datos**: Los videos están almacenados en un diccionario VIDEOS con categorías como claves.

**formulario.py**

* **Descripción**: Contiene funciones para interactuar con la base de datos SQLite.
* **Funciones**:
  + crear\_tabla: Crea la tabla valoraciones si no existe, con campos id, nombre, estrellas y experiencia.
  + guardar\_valoracion: Inserta una nueva valoración en la base de datos.

**static/estilos.css**

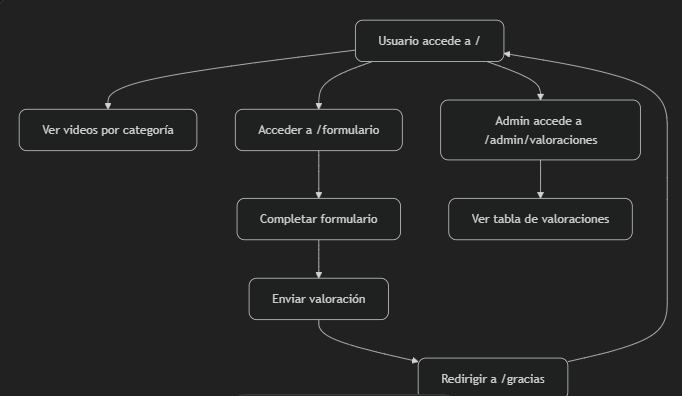
* **Descripción**: Define los estilos para la interfaz, usando variables CSS para colores, un diseño responsivo y efectos visuales (hover, scroll horizontal).
* **Características**:
  + Diseño oscuro con colores contrastantes.
  + Tarjetas de video con efecto de escala al pasar el cursor.
  + Formulario y tabla estilizados para una experiencia consistente.

**Plantillas (templates/)**

* **index.html**: Muestra los videos en secciones por categoría, usando un diseño de grid con scroll horizontal.
* **formulario.html**: Contiene un formulario para enviar valoraciones, con campos validados (requeridos).
* **ver\_valoraciones.html**: Muestra una tabla con las valoraciones almacenadas.

## Ejemplo de Diagrama de Flujo

A continuación, un diagrama simple de la interacción del usuario con la aplicación (usando Mermaid):



# CONCLUSION

**Mini Streaming** es una aplicación web desarrollada con Flask que demuestra la integración de tecnologías como Python, SQLite, HTML, CSS y Jinja2 para crear una plataforma funcional y accesible. Con su capacidad para mostrar videos organizados por categorías, recolectar valoraciones de usuarios y ofrecer un panel de administración, el proyecto refleja los fundamentos del desarrollo web moderno. Desarrollado por el \*\*Grupo #2\*\* para el curso de Programación 2, este proyecto no solo cumple con los objetivos académicos, sino que también sirve como base para futuras mejoras, como la adición de autenticación, filtros de búsqueda o pruebas unitarias. Mini Streaming es un ejemplo práctico de cómo combinar tecnologías web para crear una experiencia de usuario interactiva y educativa. ¡Clona el repositorio y explora sus posibilidades!

# REFERENCIAS BIBLIOGRAFICAS

* Metwalli, S. A. (2025, 5 septiembre). *What is flask?* Built In. https://builtin.com/software-engineering-perspectives/flask
* Pandey, A. (2025, 16 mayo). *Flask Python: A Comprehensive Guide to Building Web Applications*. Edureka. https://www.edureka.co/blog/flask-python/
* *¿Qué es Flask y cómo crear rápidamente sitio web usándolo?* (s. f.). https://codica.la/blog/que-es-flask-y-como-crear-un-sitio-web-rapido-con-el