🐍 GUÍA TEÓRICO-PRÁCTICA: FRAMEWORKS WEB EN PYTHON

INSTITUTO UNIVERSITARIO DE TECNOLOGÍA INDUSTRIAL RODOLFO LOERO ARISMENDI

ASIGNATURA: Programación III con Python | SECCIÓN: 4DA
UNIDAD: III y IV (Desarrollo Web con Python)
FECHA: 23 de Julio de 2026

📚 PARTE I: FUNDAMENTOS TEÓRICOS

1. ¿QUÉ ES UN FRAMEWORK WEB?

Un FRAMEWORK WEB es un conjunto de herramientas y bibliotecas que facilitan el desarrollo de aplicaciones web en Python.

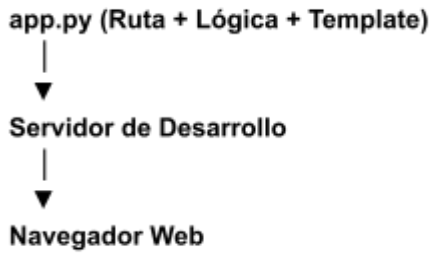
Ventajas principales:
✅ Estructura organizada del proyecto
✅ Seguridad integrada
✅ Manejo automático de rutas y URLs
✅ Conexión simplificada con bases de datos
✅ Reutilización de código.

2. COMPARATIVA: FLASK VS DJANGO VS FASTAPI

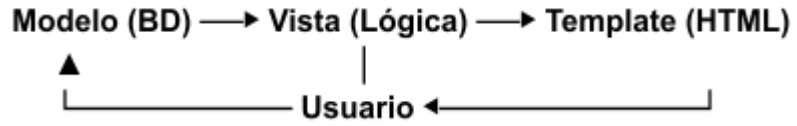
Característica	Flask	Django	FastAPI
Tipo	Micro-framework	Framework completo	Micro-framework moderno
Curva de aprendizaje	Baja	Media-Alta	Media
Velocidad	Buena	Buena	Excelente (más rápido)
Base de datos	Requiere extensión (SQLAlchemy)	Incluida (ORM)	Requiere extensión (SQLAlchemy)
Panel Admin	Requiere extensión	Incluido por defecto	Requiere extensión
Documentación automática	No	No	Sí (Swagger UI)
Validación de datos	Manual	Con forms	Automática (Pydantic)
Ideal para	Proyectos pequeños, APIs simples	Proyectos grandes, CMS, E-commerce	APIs modernas, microservicios
Año de creación	2010	2005	2018

3. **ARQUITECTURA DE CADA FRAMEWORK**

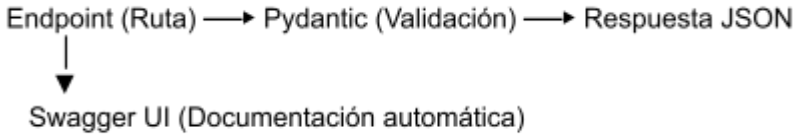
Flask (Minimalista)



Django (MVT - Model View Template)



FastAPI (Moderno con Pydantic)



4. **¿CUÁNDO USAR CADA UNO?**

Escenario	Framework Recomendado	Razón
API REST rápida y moderna	FastAPI	Validación automática, documentación Swagger
Sitio web con panel admin	Django	Admin incluido, ORM potente
Proyecto pequeño o prototipo	Flask	Sencillo, pocas dependencias
Microservicios	FastAPI	Alto rendimiento, async/await
E-commerce o CMS	Django	Seguridad, escalabilidad
Aprendizaje inicial	Flask	Curva de aprendizaje suave

💻 PARTE II: GUÍA PRÁCTICA - FASTAPI

🚀 EJERCICIO: "MI PRIMER PROYECTO CON FASTAPI"

Objetivo: Configurar el entorno de trabajo con FastAPI y crear un servidor web básico que ejecute en localhost.
Ponderación: 10% (Actividad Formativa)

PASO 1: VERIFICAR INSTALACIÓN DE PYTHON

```
# Abre la terminal o CMD y ejecuta:
python --version

# Debe mostrar Python 3.8 o superior
# Si no está instalado, descargar desde:
https://www.python.org/downloads/
```

PASO 2: INSTALAR FASTAPI Y UVICORN

```
# FastAPI es el framework
# Uvicorn es el servidor que ejecuta FastAPI

pip install fastapi
pip install uvicorn

# Verificar instalación
python -c "import fastapi; print(fastapi.__version__)"
```

PASO 3: CREAR ARCHIVO PRINCIPAL

1. Crea una carpeta llamada proyecto_fastapi_[tuapellido]
2. Dentro, crea un archivo llamado main.py
3. Escribe el siguiente código:

```
# ============================================
# MI PRIMER PROYECTO FASTAPI - IUTIRLA
# Estudiante: [Tu Nombre y Apellido]
# Cédula: V-[Tu Cédula]
# ============================================

from fastapi import FastAPI

# Crear la aplicación FastAPI
app = FastAPI(
    title="Mi Primera API",
    description="Proyecto de clase - Programación III",
    version="1.0.0"
)

# Ruta raíz (endpoint principal)
@app.get("/")
def leer_raiz():
    return {
        "mensaje": "¡Hola! Bienvenido a FastAPI",
        "estudiante": "[Tu Nombre]",
        "cedula": "V-[Tu Cédula]"
    }

# Ruta de saludo personalizado
@app.get("/saludo/{nombre}")
def saludar(nombre: str):
    return {"mensaje": f"Hola {nombre}, esto es FastAPI"}

# Punto de entrada del programa
if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="127.0.0.1", port=8000)
```

PASO 4: EJECUTAR EL SERVIDOR

```
# Navega a la carpeta de tu proyecto
cd proyecto_fastapi_[tuapellido]

# Ejecutar el servidor
python main.py

# O también puedes usar:
# uvicorn main:app --reload
```

Deberías ver algo similar a esto:

```
INFO:     Started server process [12345]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://127.0.0.1:8000
```

PASO 5: VERIFICAR EN EL NAVEGADOR

1. Abre tu navegador web
2. Ingresa a: `http://127.0.0.1:8000`
3. Deberías ver un JSON con tu mensaje de bienvenida

📄 ***Documentación Automática (Swagger UI):***

- Ingresa a: `http://127.0.0.1:8000/docs`
- Verás la documentación interactiva generada automáticamente
- Puedes probar los endpoints desde allí

PASO 6: CAPTURA DE PANTALLA (ENTREGABLE)

Toma captura de pantalla y/o foto de:

1. La terminal mostrando el servidor ejecutándose
2. El navegador en `http://127.0.0.1:8000`
3. La documentación Swagger en `http://127.0.0.1:8000/docs`

`4.` enviar la foto al grupo de whatsapp.

📊 NORMAS DE EVALUACIÓN

CRITERIOS DE APROBACIÓN

Criterio	FastAPI	Puntos
Instalación correcta	FastAPI + Uvicorn instalados	2 pts
Estructura de archivos	Carpeta + `main.py`	2 pts
Servidor ejecuta	Levanta en puerto 8000	3 pts
Ruta raíz funciona	`http://127.0.0.1:8000/` responde	2 pts
Personalización	Nombre y cédula en el código	1 pt
Capturas de pantalla	3 capturas (terminal, web, docs)	3 pts
Código limpio	Comentarios, indentación	2 pt
TOTAL		15 pts

📁 ESTRUCTURA DE ENTREGA

Carpeta Comprimida (.zip)

```
ApellidoNombre_Frameworks.zip
│
├── FastAPI/
│   ├── proyecto_fastapi_[apellido]/
│   │   ├── main.py
│   │   └── capturas/
│   │       ├── terminal_fastapi.png
│   │       ├── navegador_fastapi.png
│   │       └── swagger_docs.png
│   │
├── Flask/
│   ├── proyecto_flask_[apellido]/
│   │   ├── app.py
│   │   └── capturas/
│   │       ├── terminal_flask.png
│   │       └── navegador_flask.png
│   │
└── README.txt (con nombre, cédula y sección)
```

📞 RECURSOS DE APOYO

Recurso	Enlace
Documentación FastAPI	https://fastapi.tiangolo.com/
Documentación Flask	https://flask.palletsprojects.com/
Documentación Django	https://docs.djangoproject.com/
Python Oficial	https://docs.python.org/3/
Grupo de Clase	WhatsApp/Telegram de la sección 4DA