■ Proyecto Flask - Código Python (Parte 1)

Total de archivos: 6

Archivo 1: ./app.py

```
from flask import Flask, jsonify■
from flask_cors import CORS■
import threading■
from datetime import datetime■
■
from config import TELEGRAM_BOT_TOKEN, TELEGRAM_CHAT_ID■
from models.monitor import MonitorMaquinas■
from services.telegram_service import enviar_notificacion_telegram■
■
# Importar rutas■
from routes.chat_routes import setup_chat_routes■
from routes.monitor_routes import setup_monitor_routes■
from routes.dashboard_routes import setup_dashboard_routes■
■
app = Flask(__name__)■
CORS(app)■
■
# Instancia global del monitor■
monitor = MonitorMaquinas()■
■
# Configurar rutas■
setup_chat_routes(app)■
setup_monitor_routes(app, monitor)  # Pasar la instancia del monitor■
setup_dashboard_routes(app)■
■
@app.route("/")■
def index():■
    return jsonify({■
        "mensaje": "■ MAAS Bot API está funcionando correctamente",■
        "version": "2.0",■
        "endpoints": {■
            "/preguntar": "POST - Enviar preguntas al asistente",■
            "/monitor/start": "POST - Iniciar monitoreo",■
            "/monitor/stop": "POST - Detener monitoreo", ■
            "/monitor/status": "GET - Estado del monitoreo",■
            "/dashboard/metricas": "GET - Métricas del dashboard",■
            "/health": "GET - Health check"■
        }■
    })■
■
@app.route("/health", methods=["GET"])■
def health_check():■
    return jsonify({■
        "status": "healthy",■
        "timestamp": datetime.now().isoformat(),■
        "service": "MAAS Bot API"■
    })■
■
def iniciar_aplicacion():■
    print("■ Iniciando aplicación MAAS Bot...")■
    mensaje_inicio = f"■ <b>MAAS Bot Iniciado</b>\n■ Sistema de monitorización listo\n■ {datetime.now().strftime('%Y-
    threading.Thread(target=lambda: enviar_notificacion_telegram(mensaje_inicio)).start()■
■
if __name__ == "__main__":■
    iniciar_aplicacion()■
    ■
    print("■ Servidor Flask iniciado en http://0.0.0.0:5000")■
    app.run(host="0.0.0.0", port=5000, debug=False)
```

Archivo 2: ./config.py

```
# =========================■
# Configuración■
# =========================■
MAAS_URL = "http://172.16.25.2:5240/MAAS"■
MAAS_API_KEY = "E9nebNWw3WhSejrkAL:Av2xxeCgHq2jeGL2rG:skcKZQp85vMdya2WubtERYXhMxf7pTty"■
GEMINI_API_KEY = "AIzaSyCVmwiNmBqMaYQrvGDMBzPY_GJwrDNynt4"■
TELEGRAM_BOT_TOKEN = "8436841267:AAF5oYG_FiKvDNi-vKGh_JjL4X_v3ReQUHo"■
TELEGRAM_CHAT_ID = "5786912071"■
■
CORTESIAS = ["gracias", "muchas gracias", "ok gracias", "thank you", "ok", "perfecto"]
```

Archivo 3: ./export_code_to_pdf.py

```
import os■
from reportlab.lib.pagesizes import A4■
from reportlab.pdfgen import canvas■
■
# ■ Ruta del proyecto Flask (misma carpeta que app.py)■
PROJECT_PATH = "./"■
■
# ■ Solo archivos Python■
VALID_EXTENSIONS = (".py",)■
■
# ■ Carpetas a ignorar■
IGNORE_DIRS = {"maas-env"}■
■
■
def obtener_archivos(directorio):■
    """Recorre el directorio de forma recursiva y obtiene todos los archivos .py, ignorando carpetas especificadas."""
    archivos_validos = []■
    for raiz, subdirs, archivos in os.walk(directorio):■
        # Filtrar subdirectorios que queremos ignorar■
        subdirs[:] = [d for d in subdirs if d not in IGNORE_DIRS]■
■
        for archivo in archivos:■
            if archivo.endswith(VALID_EXTENSIONS):■
                archivos_validos.append(os.path.join(raiz, archivo))■
    return archivos_validos■
■
■
def escribir_pdf(nombre_pdf, titulo, archivos):■
    """Genera un PDF con el contenido de los archivos Python."""■
    pdf = canvas.Canvas(nombre_pdf, pagesize=A4)■
    ancho, alto = A4■
■
    pdf.setFont("Helvetica-Bold", 14)■
    pdf.drawCentredString(ancho / 2, alto - 30, f"■ {titulo}")■
    pdf.setFont("Helvetica", 10)■
    pdf.drawString(30, alto - 50, f"Total de archivos: {len(archivos)}")■
■
    for i, ruta in enumerate(archivos, 1):■
        pdf.showPage()■
        pdf.setFont("Helvetica-Bold", 12)■
        pdf.drawString(30, alto - 40, f"Archivo {i}: {ruta}")■
        pdf.setFont("Courier", 8)■
■
        try:■
            with open(ruta, "r", encoding="utf-8", errors="ignore") as f:■
                contenido = f.readlines()■
        except Exception as e:■
            contenido = [f"■■ No se pudo leer el archivo: {e}"]■
■
        y = alto - 60■
        for linea in contenido:■
            if y < 40:  # Salto de página si se acaba el espacio■
                pdf.showPage()■
                y = alto - 40■
                pdf.setFont("Courier", 8)■
            pdf.drawString(30, y, linea[:130])  # Cortar líneas largas■
            y -= 10■
■
    pdf.save()■
    print(f"■ PDF generado: {nombre_pdf}")■
■
■
def main():■
    archivos = obtener_archivos(PROJECT_PATH)■
    mitad = (len(archivos) + 1) // 2■
    primera_mitad = archivos[:mitad]■
    segunda_mitad = archivos[mitad:]■
■
    escribir_pdf("Proyecto_Flask_Backend_Parte1.pdf",■
                 "Proyecto Flask - Código Python (Parte 1)", primera_mitad)■
    escribir_pdf("Proyecto_Flask_Backend_Parte2.pdf",■
                 "Proyecto Flask - Código Python (Parte 2)", segunda_mitad)■
■
■
if __name__ == "__main__":■
```

```
main()■
```

Archivo 4: ./routes/dashboard_routes.py

```
from flask import jsonify■
import asyncio■
from services.maas_client import obtener_metricas_dashboard■
■
def setup_dashboard_routes(app):■
    """Configura las rutas del dashboard"""■
    ■
    @app.route("/dashboard/metricas", methods=["GET"])■
    def obtener_metricas_dashboard_endpoint():■
        """Endpoint para obtener todas las métricas del dashboard"""■
        try:■
            loop = asyncio.new_event_loop()■
            asyncio.set_event_loop(loop)■
            metricas = loop.run_until_complete(obtener_metricas_dashboard())■
            return jsonify(metricas)■
        except Exception as e:■
            print(f"Error en endpoint /dashboard/metricas: {e}")■
            return jsonify({■
                "resumen": {},■
                "maquinas": [],■
                "red": {},■
                "alertas": [],■
                "rendimiento": {},■
                "error": str(e)■
            }), 500■
■
    @app.route("/dashboard/maquinas", methods=["GET"])■
    def obtener_maquinas_dashboard_endpoint():■
        """Endpoint para obtener detalle de máquinas"""■
        try:■
            loop = asyncio.new_event_loop()■
            asyncio.set_event_loop(loop)■
            metricas = loop.run_until_complete(obtener_metricas_dashboard())■
            return jsonify(metricas.get("maquinas", []))■
        except Exception as e:■
            print(f"Error en endpoint /dashboard/maquinas: {e}")■
            return jsonify({"error": str(e), "maquinas": []}), 500■
■
    @app.route("/dashboard/alertas", methods=["GET"])■
    def obtener_alertas_endpoint():■
        """Endpoint para obtener alertas activas"""■
        try:■
            loop = asyncio.new_event_loop()■
            asyncio.set_event_loop(loop)■
            metricas = loop.run_until_complete(obtener_metricas_dashboard())■
            return jsonify(metricas.get("alertas", []))■
        except Exception as e:■
            print(f"Error en endpoint /dashboard/alertas: {e}")■
            return jsonify({"error": str(e), "alertas": []}), 500
```

Archivo 5: ./routes/monitor_routes.py

```
from flask import jsonify■
import asyncio■
import threading■
■
def setup_monitor_routes(app, monitor):■
    """Configura las rutas del monitor"""■
    ■
    @app.route("/monitor/start", methods=["POST"])■
    def iniciar_monitor():■
        """Endpoint para iniciar el monitoreo"""■
        try:■
            if not monitor.monitoreo_activo:■
                threading.Thread(target=lambda: asyncio.run(monitor.iniciar_monitoreo()), daemon=True).start()■
                return jsonify({■
                    "estado": "Monitoreo iniciado",■
                    "mensaje": "■ Monitoreo de máquinas MAAS iniciado correctamente"■
                })■
            else:■
                return jsonify({■
                    "estado": "El monitoreo ya está activo",■
                    "mensaje": "El monitoreo ya se encuentra en ejecución"■
                })■
        except Exception as e:■
            return jsonify({"error": str(e)}), 500■
■
    @app.route("/monitor/stop", methods=["POST"])■
    def detener_monitor():■
        """Endpoint para detener el monitoreo"""■
        try:■
            monitor.detener_monitoreo()■
            return jsonify({■
                "estado": "Monitoreo detenido", ■
                "mensaje": "■■ Monitoreo de máquinas detenido correctamente"■
            })■
        except Exception as e:■
            return jsonify({"error": str(e)}), 500■
■
    @app.route("/monitor/status", methods=["GET"])■
    def estado_monitor():■
        """Endpoint para ver el estado del monitoreo"""■
        return jsonify({■
            "monitoreo_activo": monitor.monitoreo_activo,■
            "maquinas_monitoreadas": len(monitor.estados_anteriores),■
            "intervalo_segundos": monitor.intervalo,■
            "estado": "Activo" if monitor.monitoreo_activo else "Inactivo"■
        })
```

Archivo 6: ./routes/chat_routes.py

```
from flask import jsonify, request■
import asyncio■
from services.chat_service import responder_pregunta■
■
def setup_chat_routes(app):■
    @app.route("/preguntar", methods=["POST"])■
    def preguntar():■
        try:■
            data = request.get_json()■
            if not data:■
                return jsonify({"respuesta": "■ No se recibieron datos JSON"}), 400■
                ■
            pregunta = data.get("pregunta", "")■
            if not pregunta.strip():■
                return jsonify({"respuesta": "Por favor, ingresa una pregunta."}), 400■
            ■
            loop = asyncio.new_event_loop()■
            asyncio.set_event_loop(loop)■
            respuesta = loop.run_until_complete(responder_pregunta(pregunta))■
            return jsonify({"respuesta": respuesta})■
        except Exception as e:■
            return jsonify({"respuesta": f"■ Error procesando la pregunta: {e}"}), 500
```