```
from flask import Flask, request, jsonify, send_file, send_from_directory
import os
import uuid
import subprocess
import time
import json
import socket
import threading
import signal
import base64
import requests
import shlex
import re
import unicodedata

app = Flask(__name__)

# === CONFIGURACIÃ“N DE RUTAS ===
STL_DIR = "/sdcard/3d_jobs"
os.makedirs(STL_DIR, exist_ok=True)

GEMINI_DIR = os.path.expanduser("~/3d-workstation/gemini")
os.makedirs(GEMINI_DIR, exist_ok=True)

GEMINI_CONFIG = f"{GEMINI_DIR}/gemini_config.json"
TEMP_NODE_SCRIPT = f"{GEMINI_DIR}/temp_gemini_request.js"

GITHUB_TOKEN_FILE = os.path.expanduser("~/3d-workstation/github_token.txt")

GITHUB_REPO = "txurtxil/3DIASrv"
GITHUB_API = "https://api.github.com"

active_processes = {}
process_lock = threading.Lock()

# === FUNCIONES AUXILIARES ===

def sanitize_name(prompt, max_len=40):
    prompt = unicodedata.normalize('NFKD', prompt).encode('ascii',
'ignore').decode('ascii')
    prompt = re.sub(r'[^a-zA-Z0-9\s\-_]', '', prompt)
    name = re.sub(r'[\s\-_]+', '_', prompt.strip())
    name = name.lower()
    if not name:
        name = "modelo"
    return name[:max_len]

def get_local_ip():
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    try:
```

```
        s.connect(('10.255.255.255', 1))
        IP = s.getsockname()[0]
    except Exception:
        IP = '127.0.0.1'
    finally:
        s.close()
    return IP

def get_battery_status():
    try:
        result = subprocess.run(["termux-battery-status"], capture_output=True,
text=True, timeout=3)
        if result.returncode == 0:
            data = json.loads(result.stdout)
            level = data.get("percentage", "?")
            status = data.get("status", "unknown").lower()
            icon = "ðŸ”Œ" if status == "charging" else "ðŸ”‹"
            return f"{icon} {level}%"
    except:
        pass
    return "ðŸ”‹ N/A"

def get_uptime():
    try:
        # Preferimos el binario directo si estÃ¡ presente, pero `uptime` solo suele
ser suficiente
        result = subprocess.run(["uptime"], capture_output=True, text=True,
timeout=3)
        if result.returncode == 0:
            # Ej: " 14:23:48 up 18:47,  load average..."
            output = result.stdout.strip()
            # Buscar "up X days, HH:MM" o "up HH:MM"
            import re
            match = re.search(r'up\s+((\d+)\s+day[s]?,\s+)?(\d+):(\d+)', output)
            if match:
                days = int(match.group(2)) if match.group(2) else 0
                hours = int(match.group(3))
                minutes = int(match.group(4))
                total_hours = days * 24 + hours
                if total_hours > 0:
                    return f"ðŸ•— {total_hours}h {minutes}m"
                else:
                    return f"ðŸ•— {minutes}m"
            # Si no coincide, fallback simple (como antes)
            parts = output.split()
            for i, p in enumerate(parts):
                if p == "up" and i + 1 < len(parts):
                    time_part = parts[i + 1].rstrip(',')
                    if ':' in time_part:
                        h, m = time_part.split(':')
```

```
                        hours, mins = int(h), int(m)
                        if hours > 0:
                            return f"ðŸ•— {hours}h {mins}m"
                        else:
                            return f"ðŸ•— {mins}m"
    except Exception as e:
        pass
    return "ðŸ•— N/A"

def get_hardware_stats():
    # RAM
    try:
        ram_output = subprocess.run(["free", "-m"], capture_output=True,
text=True).stdout.splitlines()
        ram = ram_output[1].split()
        ram_used = ram[2]
        ram_total = ram[1]
        ram = f"RAM: {ram_used}MB/{ram_total}MB"
    except:
        ram = "RAM: N/A"

    # CPU
    try:
        cpu_output = subprocess.run(["top", "-bn1"], capture_output=True,
text=True).stdout
        cpu_line = [line for line in cpu_output.splitlines() if "%Cpu" in line]
        if cpu_line:
            cpu_idle = float(cpu_line[0].split()[7])
            cpu_used = 100 - cpu_idle
            cpu = f"CPU: {cpu_used:.1f}%"
        else:
            cpu = "CPU: N/A"
    except:
        cpu = "CPU: N/A"

    # Disk
    try:
        disk_output = subprocess.run(["df", "-h", "/storage/emulated/0"],
capture_output=True, text=True).stdout.splitlines()
        disk = disk_output[1].split()
        disk_used = disk[2]
        disk_total = disk[1]
        disk = f"Disk: {disk_used}/{disk_total}"
    except:
        disk = "Disk: N/A"

    # Battery & Uptime
    battery = get_battery_status()
    uptime = get_uptime()
```

```
    return (
        f"<div
style='text-align:center;font-size:0.8em;color:#888;margin-top:-20px;margin-bottom:
15px;'>"
        f"{ram} | {cpu} | {disk} | {battery} | {uptime}"
        f"</div>"
    )

def setup_gemini_dependencies():
    try:
        subprocess.run(["node", "--version"], capture_output=True, check=True)
    except:
        print("Instalando Node.js...")
        subprocess.run(["pkg", "install", "nodejs", "-y"], check=True)

    if not os.path.exists(f"{GEMINI_DIR}/node_modules/@google/generative-ai"):
        print("Instalando @google/generative-ai...")
        subprocess.run(["npm", "init", "-y"], cwd=GEMINI_DIR, check=True)
        subprocess.run(["npm", "install", "@google/generative-ai"], cwd=GEMINI_DIR,
check=True)

def save_api_key(api_key, model="gemini-2.5-flash"):
    config = {"apiKey": api_key.strip(), "model": model}
    with open(GEMINI_CONFIG, "w") as f:
        json.dump(config, f, indent=2)
    return True

def has_valid_api_key():
    if not os.path.exists(GEMINI_CONFIG):
        return False
    try:
        with open(GEMINI_CONFIG, 'r') as f:
            config = json.load(f)
        return "apiKey" in config and config["apiKey"].startswith("AIza")
    except:
        return False

def get_gemini_config():
    if has_valid_api_key():
        try:
            with open(GEMINI_CONFIG, 'r') as f:
                return json.load(f)
        except:
            pass
    return {"apiKey": "", "model": "gemini-2.5-flash"}

def save_github_token(token):
    with open(GITHUB_TOKEN_FILE, "w") as f:
        f.write(token.strip())
    return True
```

```
def has_github_token():
    return os.path.exists(GITHUB_TOKEN_FILE) and os.path.getsize(GITHUB_TOKEN_FILE)
> 10

def get_github_token():
    if has_github_token():
        with open(GITHUB_TOKEN_FILE, "r") as f:
            return f.read().strip()
    return None

def generate_gemini_script(prompt, model="gemini-2.5-flash"):
    with open(TEMP_NODE_SCRIPT, "w") as f:
        f.write(f'''const {{ GoogleGenerativeAI }} =
require("@google/generative-ai");
const fs = require('fs');

async function run() {{
    try {{
        const config = JSON.parse(fs.readFileSync('{GEMINI_CONFIG}', 'utf8'));
        const genAI = new GoogleGenerativeAI(config.apiKey);
        const model = genAI.getGenerativeModel({{ model: '{model}' }});

        const fullPrompt = `Eres un experto en OpenSCAD para impresiÃ³n 3D FDM.
Genera SOLO cÃ³digo OpenSCAD vÃ¡lido para: {prompt}

REGLAS ESTRICTAS:
1. SOLO CÃ“DIGO OPENSCAD (nada de texto, explicaciones ni markdown)
2. Centrado en (0,0,0)
3. Paredes mÃ-nimas 2mm
4. $fn=24 en curvas
5. Usa comentarios // para partes clave
6. Compatible con OpenSCAD 2021+

SALIDA: Solo el cÃ³digo OpenSCAD, lÃ-nea 1 debe ser vÃ¡lida.
`;

        const result = await model.generateContent(fullPrompt, {{
            generationConfig: {{
                maxOutputTokens: 2048,
                temperature: 0.2,
                topP: 0.8
            }}
        }});

        const text = result.response.text().trim();

        if (!text.includes("module") && !text.includes("cube") &&
!text.includes("cylinder") && !text.includes("difference") &&
!text.includes("union")) {{
```

```
            throw new Error("No se detectÃ³ geometrÃ-a vÃ¡lida");
        }}

        console.log(text);
    }} catch (error) {{
        console.error("GEMINI_ERROR:" + error.message);
        process.exit(1);
    }}
}}

run().catch(console.error);
''')

def get_job_info(job_id):
    if not os.path.exists(STL_DIR):
        return None
    for entry in os.listdir(STL_DIR):
        job_dir = os.path.join(STL_DIR, entry)
        meta_path = os.path.join(job_dir, "metadata.json")
        if os.path.isdir(job_dir) and os.path.exists(meta_path):
            try:
                with open(meta_path, "r") as f:
                    meta = json.load(f)
                if meta.get("job_id") == job_id:
                    return {"job_id": job_id, "job_name": entry, "job_dir":
job_dir}
            except:
                continue
    return None

# === RUTAS ===

@app.route('/')
def home():
    api_key_missing = not has_valid_api_key()
    github_missing = not has_github_token()
    local_ip = get_local_ip()
    hardware = get_hardware_stats()
    has_git = has_github_token()
    gemini_config = get_gemini_config()
    current_model = gemini_config.get("model", "gemini-2.5-flash")

    return f'''<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>EstaciÃ³n 3D IA</title>
<style>
:root {{
```

```
    --primary: #4CAF50;
    --bg: #f5f5f5;
    --card: white;
    --error: #f44336;
    --success: #e8f5e8;
    --warn: #ff9800;
}}
body {{
    font-family: 'Segoe UI', sans-serif;
    margin: 0;
    padding: 15px;
    background: var(--bg);
}}
.container {{
    max-width: 900px;
    margin: 0 auto;
}}
.card {{
    background: var(--card);
    padding: 20px;
    border-radius: 16px;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
    margin-bottom: 20px;
}}
h1 {{
    color: var(--primary);
    text-align: center;
    margin: 0 0 25px;
    font-size: 1.8em;
}}
.api-key-form, .github-form {{
    display: flex;
    flex-direction: column;
    gap: 12px;
    margin-bottom: 20px;
}}
.api-key-form input, .github-form input, .model-select {{
    padding: 14px;
    font-size: 16px;
    border: 2px solid #ccc;
    border-radius: 8px;
}}
.api-key-form button, .github-form button {{
    background: var(--primary);
    color: white;
    padding: 14px;
    border: none;
    border-radius: 8px;
    font-weight: bold;
    cursor: pointer;
```

```
}}
.model-row {{
    display: flex;
    gap: 12px;
    margin-bottom: 20px;
}}
textarea {{
    width: 100%;
    height: 110px;
    padding: 14px;
    font-family: monospace;
    font-size: 15px;
    border: 2px solid var(--primary);
    border-radius: 8px;
    resize: vertical;
}}
button.main {{
    background: var(--primary);
    color: white;
    padding: 16px;
    font-size: 18px;
    border: none;
    border-radius: 8px;
    width: 100%;
    font-weight: bold;
    margin: 15px 0;
    cursor: pointer;
}}
button.main:disabled {{
    background: #aaa;
    cursor: not-allowed;
}}
.examples {{
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(140px, 1fr));
    gap: 10px;
    margin: 15px 0;
}}
.ex {{
    background: #e3f2fd;
    padding: 12px;
    text-align: center;
    border-radius: 8px;
    cursor: pointer;
    font-weight: bold;
    font-size: 0.9em;
    border: 2px solid #2196f3;
}}
.ex:hover {{
    background: #bbdefb;
```

```
    transform: translateY(-1px);
}}
.spinner {{
    border: 4px solid #f3f3f3;
    border-top: 4px solid var(--primary);
    border-radius: 50%;
    width: 32px;
    height: 32px;
    animation: spin 1s linear infinite;
    margin: 20px auto;
    display: none;
}}
@keyframes spin {{
    0% {{ transform: rotate(0deg); }}
    100% {{ transform: rotate(360deg); }}
}}
.result {{
    margin-top: 20px;
    padding: 18px;
    border-radius: 12px;
    display: none;
}}
.success {{
    background: var(--success);
    border: 2px solid #4CAF50;
}}
.error {{
    background: #ffeaea;
    border: 2px solid #f44336;
}}
.warn {{
    background: #fff3e0;
    border: 2px solid #ff9800;
}}
.actions {{
    display: flex;
    flex-wrap: wrap;
    gap: 10px;
    margin-top: 15px;
}}
.btn {{
    flex: 1;
    min-width: 120px;
    padding: 12px;
    text-align: center;
    color: white;
    text-decoration: none;
    border-radius: 6px;
    font-weight: bold;
}}
```

```
.btn-dl {{ background: #4CAF50; }} .btn-dl:hover {{ background: #45a049; }}
.btn-view {{ background: #2196f3; }} .btn-view:hover {{ background: #1976d2; }}
.btn-files {{ background: #ff9800; }} .btn-files:hover {{ background: #f57c00; }}
.btn-code {{ background: #9c27b0; }} .btn-code:hover {{ background: #7b1fa2; }}
.btn-logs {{ background: #607d8b; }} .btn-logs:hover {{ background: #455a64; }}
.btn-upload {{ background: #333; }} .btn-upload:hover {{ background: #111; }}
.btn-cancel {{ background: #f44336; }} .btn-cancel:hover {{ background: #d32f2f; }}
#logs {{
    margin-top: 15px;
    padding: 12px;
    background: #2d2d2d;
    color: #f0f0f0;
    border-radius: 8px;
    font-family: monospace;
    max-height: 250px;
    overflow-y: auto;
    display: none;
}}
.log {{
    margin: 4px 0;
    padding: 4px;
    border-left: 3px solid var(--primary);
}}
.log-error {{
    border-left-color: var(--error);
    color: #ffcdd2;
}}
.log-warn {{
    border-left-color: var(--warn);
    color: #ffcc80;
}}
.progress-container {{
    width: 100%;
    background: #ddd;
    border-radius: 8px;
    overflow: hidden;
    margin: 15px 0;
    height: 20px;
}}
.progress-bar {{
    height: 100%;
    width: 0%;
    background: var(--primary);
    transition: width 0.3s;
}}
.status-text {{
    text-align: center;
    font-size: 0.9em;
    color: #555;
    margin-top: 8px;
```

```
}}
.file-card {{
    background: #f9f9f9;
    padding: 12px;
    margin: 8px 0;
    border-radius: 8px;
    border: 1px solid #ddd;
    font-size: 0.9em;
}}
</style>
</head>
<body>
<div class="container">
    <div class="card">
        <h1>EstaciÃ³n 3D con IA + GitHub</h1>
        {hardware}
        <p style="text-align:center;font-size:0.9em;color:#666">
            Sube tus modelos a: <a href="https://github.com/{GITHUB_REPO}"
target="_blank">github.com/{GITHUB_REPO}</a>
        </p>

        <div class="api-key-form" style="display: {'none' if has_valid_api_key()
else 'flex'}">
            <input type="password" id="apiKey" placeholder="Pega tu API Key de
Gemini (AIza...)"/>
            <div class="model-row">
                <select id="geminiModel" class="model-select">
                    <option value="gemini-2.5-flash" {"selected" if current_model
== "gemini-2.5-flash" else ""}>gemini-2.5-flash (rÃ¡pido)</option>
                    <option value="gemini-2.5-pro" {"selected" if current_model ==
"gemini-2.5-pro" else ""}>gemini-2.5-pro (preciso)</option>
                </select>
                <button onclick="saveKey()">Guardar</button>
            </div>
        </div>

        <div class="github-form" style="display: {'none' if has_github_token() else
'flex'}">
            <input type="password" id="githubToken" placeholder="Pega tu GitHub
Token (ghp_...)"/>
            <button onclick="saveGithubToken()">Guardar GitHub Token</button>
        </div>

        <div id="mainForm" style="display: {'block' if has_valid_api_key() else
'none'}">
            <div class="model-row" style="margin-top: 0;">
                <select id="geminiModelMain" class="model-select"
onchange="updateModel()">
                    <option value="gemini-2.5-flash" {"selected" if current_model
== "gemini-2.5-flash" else ""}>âš¡ gemini-2.5-flash</option>
```

```
                    <option value="gemini-2.5-pro" {"selected" if current_model ==
"gemini-2.5-pro" else ""}>ðŸ§  gemini-2.5-pro</option>
                </select>
            </div>
            <div class="examples">
                <div class="ex" onclick="set('Caja 60x40x30mm tapa encajada')">Caja
HermÃ©tica</div>
                <div class="ex" onclick="set('Soporte telÃ©fono 60Â°')">Soporte
MÃ³vil</div>
                <div class="ex" onclick="set('Gancho pared 5kg')">Gancho 5kg</div>
                <div class="ex" onclick="set('Porta llaves 6x5mm')">Porta
Llaves</div>
            </div>
            <textarea id="prompt" placeholder="Describe tu pieza 3D (ej: caja
50x30x20mm con tapa)"></textarea>
            <button class="main" id="genBtn" onclick="generate()">Generar con
IA</button>
            <div class="spinner" id="spinner"></div>
            <div class="progress-container" id="progressContainer"
style="display:none;">
                <div class="progress-bar" id="progressBar"></div>
            </div>
            <div class="status-text" id="statusText"></div>
            <div id="result" class="result"></div>
            <div id="logs"></div>
        </div>
    </div>

    <div class="card" id="filesCard" style="display: none;">
        <h2>Explorador de Modelos</h2>
        <div id="filesList"></div>
    </div>
</div>

<script>
const localIP = "{local_ip}";
let currentJobId = null;
let progressInterval = null;
const hasGit = {str(has_git).lower()};

function set(t) {{
    document.getElementById('prompt').value = t;
}}

function updateModel() {{
    const model = document.getElementById('geminiModelMain').value;
    fetch('/set-model', {{
        method: 'POST',
        headers: {{ 'Content-Type': 'application/json' }},
        body: JSON.stringify({{ model: model }})
```

```
    }});
}}

function log(m, t = 'info') {{
    const l = document.getElementById('logs');
    const e = document.createElement('div');
    e.className = 'log ' + (t === 'error' ? 'log-error' : (t === 'warn' ?
'log-warn' : ''));
    e.textContent = '[' + new Date().toLocaleTimeString() + '] ' + m;
    l.appendChild(e);
    l.style.display = 'block';
    l.scrollTop = l.scrollHeight;
}}

function updateProgress(percent, status) {{
    const bar = document.getElementById('progressBar');
    const text = document.getElementById('statusText');
    const container = document.getElementById('progressContainer');
    bar.style.width = percent + '%';
    text.textContent = status;
    container.style.display = 'block';
}}

function startProgress() {{
    let percent = 0;
    updateProgress(0, "Iniciando...");
    progressInterval = setInterval(() => {{
        if (percent < 90) {{
            percent += Math.random() * 3;
            updateProgress(percent, "Procesando con IA...");
        }}
    }}, 800);
}}

function stopProgress() {{
    if (progressInterval) clearInterval(progressInterval);
    updateProgress(100, "Â¡Completado!");
    setTimeout(() => {{
        document.getElementById('progressContainer').style.display = 'none';
        document.getElementById('statusText').textContent = '';
    }}, 2000);
}}

async function saveKey() {{
    const k = document.getElementById('apiKey').value.trim();
    const m = document.getElementById('geminiModel').value;
    if (!k.startsWith('AIza')) return alert('API Key invÃ¡lida');
    const r = await fetch('/set-key', {{
        method: 'POST',
        headers: {{ 'Content-Type': 'application/json' }},
```

```
        body: JSON.stringify({{ key: k, model: m }})
    }});
    const d = await r.json();
    if (d.status === 'ok') {{
        document.querySelector('.api-key-form').style.display = 'none';
        document.getElementById('mainForm').style.display = 'block';
        log('Gemini Key y modelo guardados');
        loadFiles();
    }} else {{
        alert(d.error);
    }}
}}

async function saveGithubToken() {{
    const t = document.getElementById('githubToken').value.trim();
    if (!t.startsWith('ghp_')) return alert('GitHub Token invÃ¡lido');
    const r = await fetch('/set-github', {{
        method: 'POST',
        headers: {{ 'Content-Type': 'application/json' }},
        body: JSON.stringify({{ token: t }})
    }});
    const d = await r.json();
    if (d.status === 'ok') {{
        document.querySelector('.github-form').style.display = 'none';
        log('GitHub Token guardado');
        loadFiles();
    }} else {{
        alert(d.error);
    }}
}}

async function generate() {{
    const p = document.getElementById('prompt').value.trim();
    if (!p) return showError('Escribe un prompt');
    const b = document.getElementById('genBtn');
    const s = document.getElementById('spinner');
    const r = document.getElementById('result');
    document.getElementById('logs').innerHTML = '';
    b.disabled = true;
    s.style.display = 'block';
    r.style.display = 'none';
    currentJobId = null;
    startProgress();

    try {{
        const res = await fetch('/design', {{
            method: 'POST',
            headers: {{ 'Content-Type': 'application/json' }},
            body: JSON.stringify({{ prompt: p }})
        }});
```

```
        const d = await res.json();
        currentJobId = d.job_id || null;

        if (d.status === 'success') {{
            stopProgress();
            log('Ã‰xito: ' + d.job_name + ' (' + d.model + ') en ' + d.total_time);
            const uploadBtn = hasGit ? `<a href="#"
onclick="uploadJob('${{d.job_id}}'); return false;" class="btn btn-upload">Subir a
GitHub</a>` : '';
            r.innerHTML = `<div class="success">
                <h3>Â¡Listo!</h3>
                <p><strong>Nombre:</strong> ${{d.job_name}}</p>
                <p><strong>Modelo:</strong> ${{d.model}}</p>
                <p><strong>Tiempo:</strong> ${{d.total_time}}</p>
                <div class="actions">
                    <a href="${{d.download_url}}" class="btn btn-dl">Descargar
STL</a>
                    <a href="/view/${{d.job_id}}" class="btn btn-view"
target="_blank">Ver 3D</a>
                    <a href="/browse/${{d.job_id}}" class="btn btn-files"
target="_blank">Archivos</a>
                    <a href="${{d.openscad_url}}" class="btn btn-code"
target="_blank">CÃ³digo</a>
                    <a href="/logs/${{d.job_id}}" class="btn btn-logs"
target="_blank">Logs</a>
                    ${{uploadBtn}}
                </div>
            </div>`;
            loadFiles();
        }} else {{
            stopProgress();
            showError(d.error || 'Error desconocido');
        }}
    }} catch (e) {{
        stopProgress();
        showError(e.message);
    }} finally {{
        b.disabled = false;
        s.style.display = 'none';
        r.style.display = 'block';
    }}
}}

async function uploadJob(jobId) {{
    log('Subiendo a GitHub...', 'warn');
    try {{
        const r = await fetch('/upload-github/' + jobId, {{ method: 'POST' }});
        const d = await r.json();
        if (d.status === 'ok') {{
            log('Subido: ' + d.url);
```

```
            alert('Â¡Subido a GitHub!\\n' + d.url);
        }} else {{
            log(d.error, 'error');
            alert('Error: ' + d.error);
        }}
    }} catch (e) {{
        log(e.message, 'error');
    }}
}}

async function cancelJob() {{
    if (!currentJobId) return log('No hay proceso activo', 'warn');
    log('Cancelando...', 'warn');
    try {{
        await fetch('/cancel/' + currentJobId, {{ method: 'POST' }});
        log('Cancelado', 'warn');
    }} catch (e) {{
        log('Error al cancelar: ' + e.message, 'error');
    }}
}}

function showError(m) {{
    log(m, 'error');
    const cancelBtn = currentJobId ? '<button class="btn btn-cancel"
onclick="cancelJob()" style="margin-top:10px;">Cancelar Proceso</button>' : '';
    document.getElementById('result').innerHTML = `<div
class="error"><h3>Error</h3><p>${{m}}</p>${{cancelBtn}}</div>`;
    document.getElementById('result').style.display = 'block';
}}

async function loadFiles() {{
    const c = document.getElementById('filesCard');
    const l = document.getElementById('filesList');
    try {{
        const r = await fetch('/files');
        const f = await r.json();
        if (f.length === 0) {{
            l.innerHTML = '<p style="color:#888">No hay modelos aÃºn</p>';
        }} else {{
            l.innerHTML = f.map(j => {{
                const date = new Date(j.time * 1000).toLocaleString();
                const uploadLink = hasGit ? ` | <a href="#"
onclick="uploadJob('${{j.job_id}}'); return false;">Subir GitHub</a>` : '';
                return `<div class="file-card">
                    <strong>${{j.job_name}}</strong><br>
                    <small>${{date}}</small><br>
                    <a href="/view/${{j.job_id}}" target="_blank">Ver 3D</a> |
                    <a href="/download/${{j.job_id}}">STL</a> |
                    <a href="/browse/${{j.job_id}}">Archivos</a>
                    ${{uploadLink}}
```

```
                </div>`;
            }}).join('');
        }}
        c.style.display = 'block';
    }} catch (e) {{
        console.error(e);
    }}
}}

window.onload = () => {{
    if (!{str(api_key_missing).lower()}) loadFiles();
}};
</script>
</body>
</html>'''

@app.route('/set-key', methods=['POST'])
def set_key():
    data = request.json
    if not data or "key" not in data:
        return jsonify({"error": "Falta la API Key"}), 400
    key = data["key"].strip()
    model = data.get("model", "gemini-2.5-flash")
    if not key.startswith("AIza"):
        return jsonify({"error": "API Key invÃ¡lida"}), 400
    save_api_key(key, model)
    return jsonify({"status": "ok", "message": "API Key y modelo guardados"})

@app.route('/set-model', methods=['POST'])
def set_model():
    data = request.json
    if not data or "model" not in data:
        return jsonify({"error": "Falta el modelo"}), 400
    model = data["model"]
    if model not in ["gemini-2.5-flash", "gemini-2.5-pro"]:
        return jsonify({"error": "Modelo no soportado"}), 400
    config = get_gemini_config()
    save_api_key(config.get("apiKey", ""), model)
    return jsonify({"status": "ok", "model": model})

@app.route('/set-github', methods=['POST'])
def set_github():
    data = request.json
    if not data or "token" not in data:
        return jsonify({"error": "Falta el token"}), 400
    token = data["token"].strip()
    if not token.startswith("ghp_"):
        return jsonify({"error": "Token invÃ¡lido"}), 400
    save_github_token(token)
    return jsonify({"status": "ok", "message": "GitHub Token guardado"})
```

```
@app.route('/files')
def list_files():
    jobs = []
    if os.path.exists(STL_DIR):
        for entry in sorted(os.listdir(STL_DIR), reverse=True):
            job_dir = os.path.join(STL_DIR, entry)
            meta_path = os.path.join(job_dir, "metadata.json")
            if os.path.isdir(job_dir) and os.path.exists(meta_path):
                try:
                    with open(meta_path, "r") as f:
                        meta = json.load(f)
                    st = meta.get("timestamp", os.path.getmtime(job_dir))
                    jobs.append({
                        "job_id": meta["job_id"],
                        "job_name": entry,
                        "time": int(st)
                    })
                except:
                    continue
    return jsonify(jobs)

def run_gemini(job_id, job_name, prompt, log_path, scad_path):
    try:
        config = get_gemini_config()
        model = config.get("model", "gemini-2.5-flash")

        with open(log_path, "a") as f:
            f.write(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Modelo: {model}\n")
            f.write(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Iniciando
Node.js...\n")

        generate_gemini_script(prompt, model)

        process = subprocess.Popen(
            ["node", TEMP_NODE_SCRIPT],
            cwd=GEMINI_DIR,
            stdout=subprocess.PIPE,
            stderr=subprocess.PIPE,
            text=True,
            preexec_fn=os.setsid
        )

        with process_lock:
            active_processes[job_id] = process

        stdout, stderr = process.communicate()

        with open(log_path, "a") as f:
            f.write(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Node.js finalizado.
```

```
CÃ³digo: {process.returncode}\n")
            if stdout:
                f.write(f"STDOUT:\n{stdout}\n")
            if stderr:
                f.write(f"STDERR:\n{stderr}\n")

        with process_lock:
            active_processes.pop(job_id, None)

        if process.returncode != 0:
            error_msg = stderr or "Error desconocido"
            if "GEMINI_ERROR:" in error_msg:
                error_msg = error_msg.split("GEMINI_ERROR:")[-1].strip()
            raise Exception(f"Gemini fallÃ³: {error_msg}")

        scad_code = stdout.strip()
        lines = scad_code.splitlines()
        if lines and lines[0].strip().startswith('```'):
            lines = lines[1:]
        if lines and lines[-1].strip().startswith('```'):
            lines = lines[:-1]
        scad_code = '\n'.join(lines).strip()

        if not scad_code or len(scad_code) < 20:
            raise Exception("CÃ³digo OpenSCAD demasiado corto")

        with open(scad_path, "w") as f:
            f.write(scad_code)

        with open(log_path, "a") as f:
            f.write(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] CÃ³digo guardado.
Renderizando STL...\n")

        stl_path = scad_path.replace('.scad', '.stl')
        render_cmd = f"LC_ALL=C openscad -o {shlex.quote(stl_path)}
{shlex.quote(scad_path)}"
        render_result = subprocess.run(
            ["sh", "-c", render_cmd],
            capture_output=True,
            text=True
        )

        with open(log_path, "a") as f:
            f.write(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] OpenSCAD finalizado.
CÃ³digo: {render_result.returncode}\n")
            if render_result.stdout:
                f.write(f"OpenSCAD STDOUT:\n{render_result.stdout}\n")
            if render_result.stderr:
                f.write(f"OpenSCAD STDERR:\n{render_result.stderr}\n")
```

```
        if render_result.returncode != 0:
            raise Exception(f"OpenSCAD fallÃ³: {render_result.stderr[:300]}")

        with open(log_path, "a") as f:
            f.write(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Â¡Ã‰xito total!\n")

    except Exception as e:
        with open(log_path, "a") as f:
            f.write(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] ERROR EN HILO:
{str(e)}\n")
        raise

@app.route('/design', methods=['POST'])
def design():
    if not has_valid_api_key():
        return jsonify({"error": "API Key no configurada"}), 400
    data = request.json
    if not data or "prompt" not in data:
        return jsonify({"error": "Falta el prompt"}), 400
    prompt = data["prompt"].strip()
    if len(prompt) < 5:
        return jsonify({"error": "Prompt demasiado corto"}), 400

    config = get_gemini_config()
    model = config.get("model", "gemini-2.5-flash")

    job_id = str(uuid.uuid4())
    job_name_base = sanitize_name(prompt)
    suffix = job_id[:8]
    job_name = f"{job_name_base}_{suffix}"
    job_dir = os.path.join(STL_DIR, job_name)
    os.makedirs(job_dir, exist_ok=True)

    meta = {
        "job_id": job_id,
        "prompt": prompt,
        "job_name": job_name,
        "model": model,
        "timestamp": time.time()
    }
    with open(os.path.join(job_dir, "metadata.json"), "w") as f:
        json.dump(meta, f, indent=2)

    log_path = os.path.join(job_dir, "generation.log")
    scad_path = os.path.join(job_dir, "model.scad")

    with open(log_path, "w") as f:
        f.write(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Prompt: {prompt}\n")
        f.write(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Modelo: {model}\n")
        f.write(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Nombre del job:
```

```
{job_name}\n")
        f.write(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Iniciando generaciÃ³n (sin
timeout)...\n")

    try:
        setup_gemini_dependencies()
        start_time = time.time()
        thread = threading.Thread(target=run_gemini, args=(job_id, job_name,
prompt, log_path, scad_path))
        thread.start()
        thread.join()
        total_time = time.time() - start_time

        return jsonify({
            "status": "success",
            "job_id": job_id,
            "job_name": job_name,
            "model": model,
            "total_time": f"{total_time:.1f} segundos",
            "download_url": f"http://{get_local_ip()}:8080/download/{job_id}",
            "openscad_url": f"http://{get_local_ip()}:8080/openscad/{job_id}"
        })

    except Exception as e:
        with open(log_path, "a") as f:
            f.write(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] ERROR FINAL:
{str(e)}\n")
        return jsonify({"error": str(e)}), 500

@app.route('/cancel/<job_id>', methods=['POST'])
def cancel_job(job_id):
    with process_lock:
        process = active_processes.get(job_id)
        if process:
            try:
                os.killpg(os.getpgid(process.pid), signal.SIGTERM)
                info = get_job_info(job_id)
                if info:
                    log_path = os.path.join(info["job_dir"], "generation.log")
                    with open(log_path, "a") as f:
                        f.write(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] CANCELADO
POR USUARIO\n")
                return jsonify({"status": "cancelled"})
            except:
                pass
    return jsonify({"status": "not_found"}), 404

@app.route('/upload-github/<job_id>', methods=['POST'])
def upload_github(job_id):
    token = get_github_token()
```

```
    if not token:
        return jsonify({"error": "GitHub Token no configurado"}), 400

    info = get_job_info(job_id)
    if not info:
        return jsonify({"error": "Job no encontrado"}), 404

    job_dir = info["job_dir"]
    job_name = info["job_name"]
    files = {}

    for f in os.listdir(job_dir):
        path = os.path.join(job_dir, f)
        if os.path.isfile(path):
            with open(path, "rb") as fb:
                content = base64.b64encode(fb.read()).decode()
            files[f] = {"content": content}

    headers = {
        "Authorization": f"token {token}",
        "Accept": "application/vnd.github.v3+json"
    }

    url = f"{GITHUB_API}/repos/{GITHUB_REPO}/contents/{job_name}"
    for filename, filedata in files.items():
        file_url = f"{url}/{filename}"
        filedata["message"] = f"Subida automÃ¡tica - {job_name} ({info.get('model',
'gemini')})"
        filedata["branch"] = "main"
        r = requests.put(file_url, headers=headers, json=filedata)
        if r.status_code not in [200, 201]:
            return jsonify({"error": f"Error al subir {filename}: {r.text}"}), 500

    commit_url = f"https://github.com/{GITHUB_REPO}/tree/main/{job_name}"
    return jsonify({"status": "ok", "url": commit_url})

@app.route('/status')
def status():
    return jsonify({"status": "online", "message": "Servidor activo"})

@app.route('/download/<job_id>')
def download(job_id):
    info = get_job_info(job_id)
    if not info:
        return "Job no encontrado", 404
    stl_path = os.path.join(info["job_dir"], "model.stl")
    if os.path.exists(stl_path):
        return send_file(stl_path, as_attachment=True,
download_name=f"{info['job_name']}.stl")
    return "Archivo no encontrado", 404
```

```
@app.route('/view/<job_id>')
def view_3d(job_id):
    info = get_job_info(job_id)
    if not info:
        return "Job no encontrado", 404
    stl_path = os.path.join(info["job_dir"], "model.stl")
    if os.path.exists(stl_path):
        return send_file(stl_path, mimetype='model/stl')
    return "Modelo no encontrado", 404

@app.route('/openscad/<job_id>')
def view_openscad(job_id):
    info = get_job_info(job_id)
    if not info:
        return "Job no encontrado", 404
    scad_path = os.path.join(info["job_dir"], "model.scad")
    if os.path.exists(scad_path):
        with open(scad_path, 'r') as f:
            code = f.read()
        job_name = info["job_name"]
        return f'''<!DOCTYPE html>
<html>
<head>
<title>CÃ³digo - {job_name}</title>
<style>
body {{ font-family: monospace; padding: 20px; background: #1e1e1e; color: #f0f0f0;
}}
pre {{ background: #2d2d2d; padding: 20px; border-radius: 8px; overflow: auto; }}
.btn {{ display: inline-block; background: #4CAF50; color: white; padding: 10px
20px; text-decoration: none; border-radius: 5px; margin: 5px; }}
</style>
</head>
<body>
    <div style="text-align:center;margin-bottom:20px;">
        <h1>CÃ³digo OpenSCAD</h1>
        <p>ID: {job_id}<br>Nombre: {job_name}<br>Modelo: {info.get('model',
'gemini')}</p>
        <a href="/" class="btn">Volver</a>
    </div>
    <pre>{code}</pre>
</body>
</html>'''
    return "CÃ³digo no encontrado", 404

@app.route('/logs/<job_id>')
def view_logs(job_id):
    info = get_job_info(job_id)
    if not info:
        return "Job no encontrado", 404
```

```
    log_path = os.path.join(info["job_dir"], "generation.log")
    if os.path.exists(log_path):
        with open(log_path, 'r') as f:
            logs = f.read()
        job_name = info["job_name"]
        return f'''<!DOCTYPE html>
<html>
<head>
<title>Logs - {job_name}</title>
<style>
body {{ font-family: monospace; padding: 20px; background: #1e1e1e; color: #f0f0f0;
}}
pre {{ background: #2d2d2d; padding: 20px; border-radius: 8px; overflow: auto; }}
.btn {{ display: inline-block; background: #4CAF50; color: white; padding: 10px
20px; text-decoration: none; border-radius: 5px; margin: 5px; }}
</style>
</head>
<body>
    <div style="text-align:center;margin-bottom:20px;">
        <h1>Logs de GeneraciÃ³n</h1>
        <p>{job_name} ({info.get('model', 'gemini')})</p>
        <a href="/" class="btn">Volver</a>
    </div>
    <pre>{logs}</pre>
</body>
</html>'''
    return "Logs no encontrados", 404

@app.route('/browse/<job_id>')
def browse_files(job_id):
    info = get_job_info(job_id)
    if not info:
        return "Job no encontrado", 404
    job_dir = info["job_dir"]
    job_name = info["job_name"]
    files_html = ""
    for filename in os.listdir(job_dir):
        size = os.path.getsize(os.path.join(job_dir, filename))
        size_str = f"({size // 1024} KB)" if size > 1024 else f"({size} B)"
        files_html += f'<p><a href="/files/{job_id}/{filename}">{filename}</a>
{size_str}</p>'

    return f'''<!DOCTYPE html>
<html>
<head>
<title>Archivos - {job_name}</title>
<style>
body {{ padding: 20px; font-family: Arial; }}
.btn {{ display: inline-block; background: #4CAF50; color: white; padding: 10px
20px; text-decoration: none; border-radius: 5px; margin-bottom: 20px; }}
```

```
a {{ color: #1976d2; text-decoration: none; }}
a:hover {{ text-decoration: underline; }}
</style>
</head>
<body>
    <div style="margin-bottom:20px;">
        <h1>Archivos de {job_name} ({info.get('model', 'gemini')})</h1>
        <a href="/" class="btn">Volver</a>
    </div>
    {files_html}
</body>
</html>'''

@app.route('/files/<job_id>/<filename>')
def serve_files(job_id, filename):
    info = get_job_info(job_id)
    if not info:
        return "Job no encontrado", 404
    try:
        return send_from_directory(info["job_dir"], filename)
    except:
        return "Archivo no encontrado", 404

if __name__ == '__main__':
    local_ip = get_local_ip()
    print("EstaciÃ³n 3D con IA + GitHub")
    print(f"Accede: http://{local_ip}:8080")
    print(f"Modelos soportados: gemini-2.5-flash (rÃ¡pido) / gemini-2.5-pro
(preciso)")
    print(f"Sube tus modelos a: https://github.com/{GITHUB_REPO}")
    app.run(host='0.0.0.0', port=8080, threaded=True)
```