Guía Técnica Interna: Proyecto LGG MRI Segmentation

1. Objetivo y Arquitectura (AWS + Docker)

El objetivo es desplegar una app de segmentación de tumores antes del 10 de Diciembre.

Arquitectura definida:

- Frontend: Streamlit (Puerto 8501). Muestra resultados y conecta con S3.
- Backend: Flask (Puerto 5000). Procesa inferencia y gestiona BBDD.
- Datos: AWS S3. Almacena imágenes .tif (Bucket privado).
- Infraestructura: AWS EC2 (Ubuntu) orquestando contenedores con Docker Compose.

2. Flujo de Trabajo Git (¡ESTRICTO!)

Reglas de oro para no romper el proyecto. Asignad roles YA.

Rol: SENIOR (Dueño del Repo)

```
# 1. Crear repo y ramas base
git init
git checkout -b main
git checkout -b develop
git push origin main develop
```

Rol: JUNIOR (Colaborador)

```
# 1. Hacer Fork en GitHub
# 2. Clonar TU fork
git clone https://github.com/TU_USUARIO/repo.git
# 3. Conectar con el repo del Senior (Upstream)
git remote add upstream https://github.com/SENIOR_USER/repo.git

# 4. Trabajar día a día (Feature Branch)
git checkout -b feature/mi-funcionalidad
# ... trabajar ...
git add . && git commit -m 'Avance X'
git push origin feature/mi-funcionalidad
# 5. Ir a GitHub y abrir Pull Request hacia 'upstream/develop'
```

Guía Técnica Interna: Proyecto LGG MRI Segmentation

3. Configuración Docker (Copiar y Pegar)

Estos archivos deben estar en la RAÍZ del proyecto.

Archivo: Dockerfile

```
FROM python:3.9-slim

# Dependencias sistema para OpenCV
RUN apt-get update && apt-get install -y libglib2.0-0 libsm6 libxext6 libxrender-dev && rm -rf
/var/lib/apt/lists/*

WORKDIR /app
COPY requirements.txt .
# Instalar dependencias Python
RUN pip install --no-cache-dir -r requirements.txt

COPY . .
EXPOSE 5000
EXPOSE 8501
CMD ["bash"]
```

Archivo: docker-compose.yml

```
version: '3.8'
services:
  backend:
    build: .
    container_name: mri-backend
    command: python src/backend/app.py
    ports: ["5000:5000"]
    volumes: ["./data:/app/data"]

  frontend:
    build: .
    container_name: mri-frontend
    command: streamlit run src/frontend/ui.py --server.port 8501 --server.address 0.0.0.0
    ports: ["8501:8501"]
    environment:
      - API_URL=http://backend:5000/api/predict
      - AWS_DEFAULT_REGION=us-east-1
    depends_on: ["backend"]
```

Guía Técnica Interna: Proyecto LGG MRI Segmentation

4. Desarrollo Backend (Flask) - src/backend/app.py

Esqueleto obligatorio para cumplir requisitos del PDF.

```
from flask import Flask, request, jsonify
import datetime

app = Flask(__name__)

# 1. Endpoint PREDICCIÓN (Body Param)
@app.route('/api/predict', methods=['POST'])
def predict():
    if 'file' not in request.files:
        return jsonify({'error': 'No file'}), 400
    file = request.files['file']
    # Lógica de modelo aquí...
    return jsonify({'mask': 'base64_string', 'has_tumor': True})

# 2. Endpoint HISTORIAL (Path Param)
@app.route('/api/history/<patient_id>', methods=['GET'])
def history(patient_id):
    # Consulta BBDD filtrando por ID
    return jsonify({'patient': patient_id, 'scans': 5})

# 3. Endpoint LOGS (Query Param)
@app.route('/api/logs', methods=['GET'])
def logs():
    date = request.args.get('date')
    # Filtrar logs por fecha
    return jsonify({'date': date, 'logs': []})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Guía Técnica Interna: Proyecto LGG MRI Segmentation

5. Desarrollo Frontend y AWS S3

El frontend en Streamlit (src/frontend/ui.py) debe usar 'boto3' para leer las imágenes directamente desde el bucket S3, SIN descargar todo el dataset.

```
import boto3
import requests
import streamlit as st

# Cliente S3 (Usa permisos implícitos de EC2, no hardcodear keys)
s3 = boto3.client('s3')
BUCKET = 'nombre-de-vuestro-bucket'

# Listar imágenes
response = s3.list_objects_v2(Bucket=BUCKET, Prefix='samples/')
files = [obj['Key'] for obj in response.get('Contents', [])]

selected = st.selectbox('Elige imagen de S3', files)
if st.button('Analizar'):
    # Descargar en memoria
    obj = s3.get_object(Bucket=BUCKET, Key=selected)
    img_data = obj['Body'].read()
    # Enviar a Flask
    files = {'file': (selected, img_data)}
    res = requests.post('http://backend:5000/api/predict', files=files)
```

6. Checklist de Despliegue (Día 8/9)

1. AWS S3: Crear bucket, subir carpeta 'samples/' con 10 .tif.
2. AWS IAM: Crear Rol 'EC2-S3-Access' con permiso 'AmazonS3ReadOnlyAccess'.
3. AWS EC2: Lanzar Ubuntu t2.medium. Asignar el Rol IAM creado.
4. Security Group: Abrir puertos 22, 5000, 8501.
5. Instalación (SSH):
   > sudo apt update && sudo apt install docker.io docker-compose -y
   > sudo usermod -aG docker ubuntu
   > git clone [URL_REPO]
   > cd [REPO]
   > docker-compose up -d --build