

GUÍA DE DESPLIEGUE - GAMC BIG DATA DASHBOARD

PROYECTO: GAMC Big Data Dashboard - Sistema de Análisis y Predicción con Machine Learning

VERSIÓN: 1.0

FECHA: Diciembre 2024

ÍNDICE

- [1. Requisitos del Sistema](#)
- [2. Instalación Local](#)
- [3. Configuración](#)
- [4. Ejecución](#)
- [5. Endpoints de API](#)
- [6. Flujo de Entrenamiento y Predicciones](#)
- [7. Despliegue en Producción](#)
- [8. Solución de Problemas](#)

1. REQUISITOS DEL SISTEMA

1.1 Requisitos Mínimos

Sistema Operativo: - Windows 10/11, macOS 10.15+, o Linux (Ubuntu 20.04+)

Software Requerido: - **Python 3.8+**: Lenguaje del backend - **Node.js 18+**: Runtime para el frontend - **npm 9+** o **yarn 1.22+**: Gestor de paquetes Node.js - **Git**: Control de versiones

Servicios Externos: - **MongoDB Atlas**: Base de datos en la nube (cuenta gratuita disponible) -

Supabase: Autenticación (opcional, puede usar autenticación local)

1.2 Recursos del Sistema

Mínimo: - RAM: 4 GB - Disco: 2 GB libres - CPU: 2 cores

Recomendado: - RAM: 8 GB - Disco: 5 GB libres - CPU: 4 cores

2. INSTALACIÓN LOCAL

2.1 Diagrama del Flujo de Despliegue

```
flowchart TD
    A[Clonar Repositorio] --> B[Configurar Backend]
    B --> C[Instalar Dependencias Python]
    C --> D[Configurar Variables Entorno]
    D --> E[Configurar Frontend]
    E --> F[Instalar Dependencias Node.js]
    F --> G[Ejecutar Backend]
    G --> H[Ejecutar Frontend]
    H --> I[Sistema Funcionando]

    B --> B1[Crear entorno virtual]
    B1 --> B2[Activar entorno virtual]
    B2 --> C

    E --> E1[cd frontend]
    E1 --> F
```

2.2 Paso 1: Clonar Repositorio

```
# Clonar el repositorio
git clone https://github.com/Mystbeast1234263/BigDataProyect-ImplementadoConMachineLearning.git
cd BigDataProyect-ImplementadoConMachineLearning
```

Verificación:

```
# Verificar estructura
ls -la
# Deberías ver: backend/, frontend/, docs/, README.md
```

2.3 Paso 2: Configurar Backend

2.3.1 Crear Entorno Virtual

Windows (PowerShell):

```
# Crear entorno virtual
python -m venv venv

# Activar entorno virtual
.\venv\Scripts\Activate.ps1
```

Windows (CMD):

```
python -m venv venv
venv\Scripts\activate
```

Linux/macOS:

```
python3 -m venv venv
source venv/bin/activate
```

Verificación:

```
# Deberías ver (venv) al inicio del prompt
which python # Linux/macOS
where python # Windows
```

2.3.2 Instalar Dependencias

```
# Instalar dependencias Python
pip install --upgrade pip
pip install -r backend/requirements.txt
```

Dependencias Principales: - **fastapi**: Framework web - **uvicorn**: Servidor ASGI - **pymongo**: Cliente MongoDB - **pandas**: Procesamiento de datos - **scikit-learn**: Machine Learning - **matplotlib**, **seaborn**: Visualizaciones - **python-dotenv**: Variables de entorno

Verificación:

```
pip list | grep fastapi
# Debería mostrar: fastapi x.x.x
```

2.4 Paso 3: Configurar Frontend

```
# Navegar a carpeta frontend
cd frontend

# Instalar dependencias Node.js
npm install
# O con yarn: yarn install
```

Dependencias Principales: - `react`: Biblioteca UI - `react-router-dom`: Navegación - `axios`: Cliente HTTP - `recharts`: Gráficos interactivos - `tailwindcss`: Framework CSS

Verificación:

```
npm list react
# Debería mostrar: react@18.x.x
```

2.5 Paso 4: Configurar Variables de Entorno

Crear archivo `.env` en la raíz del proyecto:

```
# Desde la raíz del proyecto
cd ..
touch .env # Linux/macOS
# O crear manualmente en Windows
```

Contenido del archivo `.env`:

```
# MongoDB Atlas
MONGO_URI=mongodb+srv://usuario:password@cluster.mongodb.net/database?retryWrites=true&w=majority

# Supabase (Opcional)
SUPABASE_URL=https://tu-proyecto.supabase.co
SUPABASE_SERVICE_ROLE_KEY=tu-service-role-key

# JWT Secret
JWT_SECRET_KEY=tu-clave-secreta-muy-segura-aqui-minimo-32-caracteres

# Backend
BACKEND_URL=http://localhost:8000
FRONTEND_URL=http://localhost:5173
```

Obtener MongoDB URI: 1. Crear cuenta en [MongoDB Atlas](#) 2. Crear un cluster (gratis disponible) 3. Crear usuario de base de datos 4. Whitelist IP (0.0.0.0/0 para desarrollo) 5. Copiar connection

string y reemplazar `<password>` con tu contraseña

Obtener Supabase (Opcional): 1. Crear cuenta en [Supabase](#) 2. Crear proyecto 3. Ir a Settings → API 4. Copiar URL y Service Role Key

Verificación:

```
# Verificar que el archivo existe
cat .env # Linux/macOS
type .env # Windows
```

3. CONFIGURACIÓN

3.1 Configurar MongoDB Atlas

1. **Crear Base de Datos:**

Nombre sugerido: `gamc_bigdata`

Crear Colecciones (se crean automáticamente al insertar datos):

4. `air_sensors`

5. `sound_sensors`

`underground_sensors`

Configurar Índices (opcional, se crean automáticamente): `javascript` // En MongoDB Compass o shell `db.air_sensors.createIndex({deviceName: 1, time: 1}, {unique: true})`
`db.sound_sensors.createIndex({deviceName: 1, time: 1}, {unique: true})`
`db.underground_sensors.createIndex({deviceName: 1, time: 1}, {unique: true})`

3.2 Configurar Supabase (Opcional)

Si usas Supabase para autenticación:

Crear Tabla de Usuarios: `sql CREATE TABLE users (id UUID PRIMARY KEY DEFAULT uuid_generate_v4(), email TEXT UNIQUE NOT NULL, password_hash TEXT NOT NULL, created_at TIMESTAMP DEFAULT NOW());`

Configurar JWT Secret:

3. Usar el mismo `JWT_SECRET_KEY` del `.env`

4. EJECUCIÓN

4.1 Opción 1: Ejecutar Todo Automáticamente (Recomendado)

Windows:

```
# Ejecutar script que inicia backend y frontend
.\start_all.bat
```

PowerShell:

```
.\start_all.ps1
```

Linux/macOS:

```
# Crear script similar o ejecutar manualmente
```

4.2 Opción 2: Ejecutar por Separado

4.2.1 Terminal 1 - Backend

```
# Activar entorno virtual (si no está activo)
# Windows:
.\venv\Scripts\Activate.ps1
# Linux/macOS:
source venv/bin/activate

# Navegar a backend
cd backend

# Ejecutar servidor
python -m uvicorn main:app --reload --host 0.0.0.0 --port 8000
```

Verificación: - Abrir navegador: <http://localhost:8000/docs> - Deberías ver la documentación interactiva de Swagger

4.2.2 Terminal 2 - Frontend

```
# Navegar a frontend
cd frontend

# Ejecutar servidor de desarrollo
npm run dev
# O con yarn: yarn dev
```

Verificación: - Abrir navegador: <http://localhost:5173> - Deberías ver el dashboard

4.3 Verificar que Todo Funciona

1. **Backend:** <http://localhost:8000/docs>
2. Deberías ver la documentación de Swagger

Probar endpoint: `GET /api/sensors/air/data`

Frontend: <http://localhost:5173>

5. Deberías ver el dashboard principal
6. Iniciar sesión (crear cuenta si es necesario)

5. ENDPOINTS DE API

5.1 Autenticación

Login

```
POST http://localhost:8000/api/auth/login
Content-Type: application/json

{
  "email": "usuario@ejemplo.com",
  "password": "contraseña"
}
```

Respuesta:

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_type": "bearer",
  "user": {
    "email": "usuario@ejemplo.com"
  }
}
```

Registro

```
POST http://localhost:8000/api/auth/register
Content-Type: application/json
```

```
{
  "email": "nuevo@ejemplo.com",
  "password": "contraseña_segura"
}
```

5.2 Sensores

Obtener Datos

```
GET http://localhost:8000/api/sensors/air/data?days_back=30&limit=1000
Authorization: Bearer {token}
```

Parámetros: - `days_back`: Días hacia atrás (default: 30) - `limit`: Límite de registros (default: 5000)
- `date_from`: Fecha inicio (YYYY-MM-DD, opcional) - `date_to`: Fecha fin (YYYY-MM-DD, opcional)

Respuesta:

```
{
  "success": true,
  "data": [
    {
      "time": "2024-12-01T10:00:00",
      "deviceName": "sensor_001",
      "co2_ppm": 450,
      "temperatura_c": 22.5,
      "humedad_percent": 65
    }
  ],
  "total": 1000
}
```


Subir Archivo

```
POST http://localhost:8000/api/sensors/air/data
Authorization: Bearer {token}
Content-Type: multipart/form-data

file: [archivo.csv]
```

5.3 Machine Learning

Entrenar Modelo

```
POST http://localhost:8000/api/ml/train
Authorization: Bearer {token}
Content-Type: application/json

{
  "sensor_type": "air",
  "metric": "co2_ppm",
  "date_from": "2024-11-15",
  "date_to": "2024-12-30",
  "model_type": "auto",
  "test_size": 0.2,
  "random_state": 42
}
```

Respuesta:

```
{
  "success": true,
  "model_key": "air_co2_ppm_random_forest",
  "model_type": "random_forest",
  "metrics": {
    "accuracy": 0.925,
    "precision": 0.91,
    "recall": 0.89,
    "f1_score": 0.90,
    "confusion_matrix": [[850, 45, 5], [30, 120, 10], [5, 15, 80]],
    "confusion_matrix_labels": ["normal", "warning", "critical"]
  },
  "all_models_metrics": {
    "random_forest": {...},
    "logistic_regression": {...},
    "decision_tree": {...}
  }
}
```

Predicción por Fecha

```
POST http://localhost:8000/api/ml/predict/regression
Authorization: Bearer {token}
Content-Type: application/json

{
  "sensor_type": "air",
  "metric": "co2_ppm",
  "prediction_date": "2024-12-15",
  "model_key": "air_co2_ppm_random_forest"
}
```

Respuesta:

```
{
  "success": true,
  "model_key": "air_co2_ppm_random_forest",
  "metric": "co2_ppm",
  "sensor_type": "air",
  "prediction_date": "2024-12-15",
  "predictions": [
    {
      "timestamp": "2024-12-15T08:00:00",
      "predicted_class": "normal",
      "predicted_value": 450.5,
      "confidence": 0.92,
      "probabilities": {
        "normal": 0.92,
        "warning": 0.06,
        "critical": 0.02
      }
    }
  ],
  "total_predictions": 3
}
```

Predicción por Mes

```
POST http://localhost:8000/api/ml/predict/regression
Authorization: Bearer {token}
Content-Type: application/json

{
  "sensor_type": "air",
  "metric": "co2_ppm",
  "prediction_month": "2024-12",
  "model_key": "air_co2_ppm_random_forest"
}
```

Listar Modelos Entrenados

```
GET http://localhost:8000/api/ml/models
Authorization: Bearer {token}
```

Obtener Métricas

```
GET http://localhost:8000/api/ml/metrics/air_co2_ppm_random_forest
Authorization: Bearer {token}
```

5.4 Predicciones

Predicciones de Serie de Tiempo

```
GET http://localhost:8000/api/predictions/air?metric_name=co2_ppm&horizon_hours=24&model_type=lin
Authorization: Bearer {token}
```

Análisis de Salud

```
GET http://localhost:8000/api/predictions/air/health?days_back=7
Authorization: Bearer {token}
```

6. FLUJO DE ENTRENAMIENTO Y PREDICCIONES

6.1 Flujo Completo

```
sequenceDiagram
    participant U as Usuario
    participant F as Frontend
    participant B as Backend
    participant M as MongoDB
    participant ML as ML Service
    participant FS as File System

    U->>F: Accede a ML Wizard
    F->>B: GET /api/sensors/air/data
    B->>M: Consulta datos disponibles
    M-->>B: Retorna datos
    B-->>F: Datos disponibles
    F-->>U: Muestra verificación de datos
```

```

U->>F: Configura entrenamiento
F->>B: POST /api/ml/train
B->>M: Carga datos históricos
M-->>B: Datos para entrenamiento
B->>ML: Prepara datos (features)
ML->>ML: Entrena 3 modelos
ML->>FS: Guarda mejor modelo .pkl
ML->>FS: Guarda métricas .json
ML->>FS: Exporta matriz confusión .png
ML-->>B: Métricas y resultados
B-->>F: Respuesta con métricas
F-->>U: Muestra resultados (Paso 4)

U->>F: Selecciona modelo entrenado
U->>F: Configura predicción (fecha/mes)
F->>B: POST /api/ml/predict/regression
B->>FS: Carga modelo .pkl
B->>M: Carga datos para contexto
M-->>B: Datos históricos
B->>ML: Genera predicciones
ML-->>B: Predicciones con valores
B-->>F: JSON con predicciones
F-->>U: Muestra gráficos (Paso 5)

```

6.2 Pasos Detallados para Entrenar un Modelo

1. **Verificar Datos** (ML Wizard Paso 1):
2. Seleccionar tipo de sensor (Air, Sound, Underground)
3. Seleccionar métrica (CO₂, Temperatura, etc.)

El sistema verifica disponibilidad de datos

Configurar Entrenamiento (ML Wizard Paso 2):

6. Seleccionar rango de fechas (opcional)
7. Configurar parámetros (test_size, random_state)

Click en "Entrenar Modelo"

Ver Resultados (ML Wizard Paso 3-4):

10. Progreso del entrenamiento
11. Métricas de los 3 modelos
12. Modelo seleccionado (mejor F1-Score)
13. Matriz de confusión

Gráficos de rendimiento

Hacer Predicciones (ML Wizard Paso 5):

16. Seleccionar modelo entrenado
17. Elegir tipo de predicción:
 - **Por Fecha:** Fecha específica (3 predicciones horarias)
 - **Por Mes:** Mes completo (predicciones diarias)
 - **Por Semana:** Similar a mes, con rango de 7 días
18. Ver gráficos interactivos
19. Filtrar por clase (normal/warning/critical)

6.3 Ejemplo Completo: Entrenar y Predecir CO₂

Paso 1: Entrenar Modelo

```
# Usando curl (o Postman/Thunder Client)
curl -X POST http://localhost:8000/api/ml/train \
-H "Authorization: Bearer {token}" \
-H "Content-Type: application/json" \
-d '{
  "sensor_type": "air",
  "metric": "co2_ppm",
  "date_from": "2024-11-15",
  "date_to": "2024-12-30",
  "model_type": "auto"
}'
```

Resultado Esperado: - Modelo guardado: `backend/models/air_co2_ppm_random_forest.pkl` - Métricas guardadas: `backend/models/air_co2_ppm_random_forest_metrics.json` - Matriz de confusión: `backend/models/air_co2_ppm_random_forest_confusion_matrix.png` - F1-Score: ≥ 0.85 (85%)

Paso 2: Hacer Predicción

```
curl -X POST http://localhost:8000/api/ml/predict/regression \
-H "Authorization: Bearer {token}" \
-H "Content-Type: application/json" \
-d '{
  "sensor_type": "air",
```

```
"metric": "co2_ppm",  
"prediction_date": "2024-12-15",  
"model_key": "air_co2_ppm_random_forest"  
'
```

Resultado Esperado: - 3 predicciones horarias (8:00, 14:00, 20:00) - Valores numéricos (**predicted_value**) - Clases predichas (**predicted_class**) - Confianza (**confidence**)

7. DESPLIEGUE EN PRODUCCIÓN

7.1 Opción 1: Railway (Recomendado para Inicio)

1. **Crear Cuenta:** <https://railway.app>
2. **Conectar Repositorio:** GitHub/GitLab
3. **Configurar Variables de Entorno:** En el dashboard de Railway
4. **Desplegar:** Railway detecta automáticamente Python/Node.js

Ventajas: - Despliegue automático desde GitHub - SSL automático - Escalado fácil

7.2 Opción 2: VPS (Hostinger, DigitalOcean, etc.)

7.2.1 Configurar Servidor

```
# Actualizar sistema  
sudo apt update && sudo apt upgrade -y  
  
# Instalar Python, Node.js, Nginx  
sudo apt install python3 python3-pip nodejs npm nginx -y  
  
# Instalar PM2 para backend  
sudo npm install -g pm2
```

7.2.2 Desplegar Backend

```
# Clonar repositorio  
git clone https://github.com/Mystbeast1234263/BigDataProyect-ImplementadoConMachineLearning.git  
cd BigDataProyect-ImplementadoConMachineLearning
```

```
# Configurar entorno virtual
python3 -m venv venv
source venv/bin/activate
pip install -r backend/requirements.txt

# Configurar .env
nano .env # Editar variables de entorno

# Ejecutar con PM2
cd backend
pm2 start "python -m uvicorn main:app --host 0.0.0.0 --port 8000" --name gamc-backend
pm2 save
pm2 startup # Seguir instrucciones
```

7.2.3 Desplegar Frontend

```
# Build de producción
cd frontend
npm install
npm run build

# Configurar Nginx
sudo nano /etc/nginx/sites-available/gamc
```

Configuración Nginx:

```
server {
    listen 80;
    server_name tu-dominio.com;

    # Frontend
    location / {
        root /ruta/a/frontend/dist;
        try_files $uri $uri/ /index.html;
    }

    # Backend API
    location /api {
        proxy_pass http://localhost:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

```
# Habilitar sitio
sudo ln -s /etc/nginx/sites-available/gamc /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

7.2.4 Configurar SSL (Let's Encrypt)

```
sudo apt install certbot python3-certbot-nginx -y
sudo certbot --nginx -d tu-dominio.com
```

8. SOLUCIÓN DE PROBLEMAS

8.1 Backend No Inicia

Problema: `ModuleNotFoundError` o errores de importación

Solución:

```
# Verificar que el entorno virtual está activo
which python # Debería mostrar ruta a venv

# Reinstalar dependencias
pip install --upgrade pip
pip install -r backend/requirements.txt --force-reinstall
```

8.2 MongoDB No Conecta

Problema: `pymongo.errors.ServerSelectionTimeoutError`

Soluciones: 1. Verificar que la IP está en whitelist de MongoDB Atlas (0.0.0.0/0 para desarrollo)
2. Verificar credenciales en `.env` 3. Verificar que el cluster está activo en MongoDB Atlas

```
# Probar conexión
python -c "from pymongo import MongoClient; import os; from dotenv import load_dotenv; load_dotenv"
```

8.3 Frontend No Muestra Datos

Problema: Errores 401 (No autorizado) o 404

Soluciones: 1. Verificar que el backend está ejecutándose 2. Verificar que el token JWT es válido
3. Verificar URL del backend en `frontend/src/services/api.js`


```
// frontend/src/services/api.js
const API_URL = import.meta.env.VITE_API_URL || 'http://localhost:8000';
```

8.4 Modelo ML No Entrena

Problema: "No se encontraron datos" o "Datos insuficientes"

Soluciones: 1. Verificar que hay datos en MongoDB para el sensor y métrica seleccionados 2. Ampliar rango de fechas 3. Verificar que la métrica existe en los datos

```
# Verificar datos en MongoDB
# Usar MongoDB Compass o shell
db.air_sensors.countDocuments({})
db.air_sensors.findOne()
```

8.5 Matriz de Confusión No se Exporta

Problema: Error al guardar imagen PNG

Solución:

```
# Verificar permisos de escritura
ls -la backend/models/
chmod 755 backend/models/

# Verificar que matplotlib está instalado
pip install matplotlib seaborn
```

8.6 Puerto Ya en Uso

Problema: Address already in use

Solución:

```
# Windows: Encontrar proceso
netstat -ano | findstr :8000
taskkill /PID <PID> /F

# Linux/macOS: Encontrar proceso
lsof -i :8000
kill -9 <PID>

# O cambiar puerto
```

```
python -m uvicorn main:app --port 8001
```

CONTACTO Y SOPORTE

Para problemas o preguntas: - **Repositorio:** [GitHub](#) [Issues](#) - **Documentación:** Ver [docs/INFORME_ML.md](#)
