🚀 Guía de Instalación - IoT Sensor Platform

Esta guía te ayudará a instalar y configurar la plataforma IoT paso a paso.

📋 Tabla de Contenidos

🐳 Instalación Rápida con Docker (Recomendado)

Requisitos Previos

- Docker 20.10+
- Docker Compose 2.0+

Paso 1: Clonar o navegar al proyecto

```
cd /home/ubuntu/iot_sensor_platform
```

Paso 2: Ejecutar el script de inicio automático

```
./start_docker.sh
```

El script automáticamente:
- ✅ Crea el archivo `.env` si no existe
- ✅ Construye las imágenes Docker
- ✅ Levanta PostgreSQL y Django
- ✅ Ejecuta las migraciones
- ✅ Crea roles y permisos por defecto
- ✅ Te pregunta si quieres crear superusuario y datos de prueba

Paso 3: Acceder a la aplicación

- **API**: http://localhost:8000/api/
- **Admin**: http://localhost:8000/admin/
- **Documentación**: http://localhost:8000/api/docs/

Comandos Docker Útiles

```
# Ver logs en tiempo real
docker-compose logs -f django

# Ejecutar comandos Django
docker-compose exec django python manage.py <comando>

# Detener servicios
docker-compose down

# Reiniciar servicios
docker-compose restart

# Eliminar todo (incluyendo volúmenes)
docker-compose down -v
```

🖥️ Instalación Manual sin Docker

Requisitos Previos

- Python 3.11+
- PostgreSQL 15+
- pip
- virtualenv (opcional pero recomendado)

Paso 1: Crear entorno virtual

```
cd /home/ubuntu/iot_sensor_platform
python -m venv venv
source venv/bin/activate  # En Windows: venv\Scripts\activate
```

Paso 2: Instalar dependencias

```
pip install -r requirements.txt
```

Paso 3: Configurar PostgreSQL

En Linux/Mac:

```
# Iniciar PostgreSQL
sudo systemctl start postgresql

# Conectar como superusuario postgres
sudo -u postgres psql

# En el shell de PostgreSQL:
CREATE DATABASE iot_sensor_db;
CREATE USER iot_user WITH PASSWORD 'iot_password_123';
GRANT ALL PRIVILEGES ON DATABASE iot_sensor_db TO iot_user;
\q
```

En Windows:

```
# Desde pgAdmin o psql:
CREATE DATABASE iot_sensor_db;
CREATE USER iot_user WITH PASSWORD 'iot_password_123';
GRANT ALL PRIVILEGES ON DATABASE iot_sensor_db TO iot_user;
```

Paso 4: Configurar variables de entorno

```
cp .env.example .env
nano .env  # o usa tu editor favorito
```

Edita el archivo `.env` con tus configuraciones:

```
DEBUG=True
SECRET_KEY=tu-clave-secreta-aqui-cambiar-en-produccion
DB_HOST=localhost
DB_NAME=iot_sensor_db
DB_USER=iot_user
DB_PASSWORD=iot_password_123
```

Paso 5: Ejecutar el script de inicio

```
./start.sh
```

O ejecutar manualmente:

```
# Migraciones
python manage.py makemigrations
python manage.py migrate

# Crear roles y permisos
python manage.py crear_permisos_default
python manage.py crear_roles_default

# Crear superusuario
python manage.py crear_superuser
# Usuario: admin, Password: admin123

# (Opcional) Crear datos de prueba
python manage.py crear_datos_prueba

# Iniciar servidor
python manage.py runserver 0.0.0.0:8000
```

Paso 6: Acceder a la aplicación

- **API**: http://localhost:8000/api/
- **Admin**: http://localhost:8000/admin/
- **Documentación**: http://localhost:8000/api/docs/

🗄️ Configuración de la Base de Datos

PostgreSQL (Recomendado)

```
DB_NAME=iot_sensor_db
DB_USER=iot_user
DB_PASSWORD=tu_password_seguro
DB_HOST=localhost  # o 'postgres' si usas Docker
DB_PORT=5432
```

SQLite (Solo para desarrollo/pruebas)

Si quieres usar SQLite temporalmente, modifica `config/settings.py`:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

⚠️ **ADVERTENCIA**: SQLite no es recomendado para producción o grandes volúmenes de datos.

✅ Verificación de la Instalación

1. Verificar que el servidor esté corriendo

```
curl http://localhost:8000/api/
```

Deberías ver una respuesta JSON con los endpoints disponibles.

2. Verificar la autenticación

```
# Login
curl -X POST http://localhost:8000/api/auth/login/ \
  -H "Content-Type: application/json" \
  -d '{"username": "admin", "password": "admin123"}'
```

Deberías recibir tokens JWT.

3. Verificar la documentación

Abre en tu navegador: http://localhost:8000/api/docs/

Deberías ver la interfaz de Swagger UI.

4. Verificar el admin de Django

Abre en tu navegador: http://localhost:8000/admin/

Inicia sesión con: `admin` / `admin123`

5. Verificar roles y permisos

```
# Con Docker
docker-compose exec django python manage.py shell

# Sin Docker
python manage.py shell
```

En el shell de Python:

```
from api.models import Rol, Permiso
print(f"Roles: {Rol.objects.count()}")
print(f"Permisos: {Permiso.objects.count()}")
```

Deberías ver: `Roles: 3` y `Permisos: 13`

🔧 Comandos de Gestión

Crear Superusuario

```
# Método 1: Con valores por defecto
python manage.py crear_superuser

# Método 2: Con valores personalizados
python manage.py crear_superuser --username=myadmin --email=admin@example.com --pass-
word=mypassword

# Método 3: Interactivo
python manage.py createsuperuser
```

Crear Datos de Prueba

```
python manage.py crear_datos_prueba
```

Esto crea:
- 3 usuarios operadores (operador1, operador2, operador3)
- 1 usuario de solo lectura (viewer)
- 6 sensores de ejemplo
- 4 dispositivos de ejemplo
- Asignaciones de sensores a dispositivos
- 120 lecturas de ejemplo

Listar Comandos Disponibles

```
python manage.py help
```

Ejecutar Tests

```
python manage.py test
```

🐛 Solución de Problemas

Error: “ModuleNotFoundError: No module named ‘decouple’“

Solución: Instala las dependencias

```
pip install -r requirements.txt
```

Error: “could not connect to server: Connection refused”

Problema: PostgreSQL no está corriendo o no está accesible.

Solución:

```
# Linux
sudo systemctl status postgresql
sudo systemctl start postgresql

# Docker
docker-compose ps
docker-compose up -d postgres
```

Error: “relation ‘api_customuser’ does not exist”

Problema: Las migraciones no se han ejecutado.

Solución:

```
python manage.py makemigrations
python manage.py migrate
```

Error: “FATAL: password authentication failed”

Problema: Credenciales de base de datos incorrectas.

Solución:
1. Verifica el archivo `.env`
2. Asegúrate de que DB_USER y DB_PASSWORD coincidan con PostgreSQL
3. Recrea el usuario en PostgreSQL si es necesario

Error: “Port 8000 is already in use”

Problema: Ya hay un servidor corriendo en el puerto 8000.

Solución:

```
# Encontrar el proceso
lsof -i :8000

# Matarlo
kill -9 <PID>

# O usa otro puerto
python manage.py runserver 0.0.0.0:8001
```

Error: “Invalid HTTP_HOST header”

Problema: El host no está en ALLOWED_HOSTS.

Solución: Agrega tu host en `.env`:

```
ALLOWED_HOSTS=localhost,127.0.0.1,tu-ip,tu-dominio.com
```

Los logs no se están creando

Problema: El directorio `logs/` no existe.

Solución:

```
mkdir logs
python manage.py runserver
```

Error 401 Unauthorized en endpoints

Problema: Token JWT no válido o expirado.

Solución:
1. Haz login de nuevo para obtener un nuevo token
2. Verifica que estés enviando el header: `Authorization: Bearer {token}`
3. Verifica que el token no haya expirado (60 minutos por defecto)

Error 403 Forbidden

Problema: El usuario no tiene permisos para esa acción.

Solución:
1. Verifica que el usuario tenga el rol apropiado
2. Verifica que el rol tenga los permisos necesarios
3. Si eres superusuario, deberías tener acceso completo

🔐 Seguridad en Producción

Antes de desplegar en producción, asegúrate de:

1. **Cambiar el SECRET_KEY**:
   `python`
   `# Generar una nueva clave`
   `python -c 'from django.core.management.utils import get_random_secret_key; print(get_random_secret_key())'`
2. **Establecer DEBUG=False**:
   `env`
   `DEBUG=False`
3. **Configurar ALLOWED_HOSTS**:
   `env`
   `ALLOWED_HOSTS=tu-dominio.com,www.tu-dominio.com`

4. **Usar contraseñas seguras**:
   - Cambiar contraseñas de base de datos
   - Cambiar contraseña del superusuario

5. **Configurar HTTPS**:
   - Usar certificados SSL/TLS
   - Configurar nginx o Apache como proxy reverso

6. **Configurar CORS apropiadamente**:
   `env`
   `CORS_ALLOWED_ORIGINS=https://tu-frontend.com`

📊 Siguientes Pasos

Una vez instalado y funcionando:

1. **Explora la API**: http://localhost:8000/api/docs/
2. **Lee la documentación**: `API_DOCUMENTATION.md`
3. **Revisa el modelo de datos**: `MODELO_ER.md`
4. **Crea tus propios sensores y dispositivos**
5. **Integra con tus dispositivos IoT reales**

📞 Soporte

Si encuentras problemas no cubiertos en esta guía:

1. Revisa los logs:
   ```bash
   # Con Docker
   docker-compose logs -f django

# Sin Docker
tail -f logs/django.log
```

1. Verifica el archivo `README.md` para más información
2. Consulta la documentación de Django y DRF

⚠️ NOTA IMPORTANTE

Este localhost (127.0.0.1) se refiere al localhost de la computadora que está ejecutando el servidor Django, no a tu máquina local. Para acceder de forma local o remota, necesitarás desplegar la aplicación en tu propio sistema o servidor.

¡Disfruta construyendo tu plataforma IoT! 🚀