MANUAL DE INSTALACIÓN

Aplicación Software SIVET

Sistema de Visualización Energética Transaccional

Versión:	2.0
Fecha:	27 de agosto de 2025
Autor(es):	Brayan Stiven López Méndez Wilson Olmedo Achicanoy Martínez
Proyecto:	Sistema de Visualización Energética Transaccional
Estado:	100 % de avance - Fase de maduración

Pasto, Nariño, Colombia

Índice

1 Introducción

Este manual proporciona instrucciones detalladas para la instalación y configuración del **Sistema de Visualización Energética Transaccional (SIVET)** en un entorno de desarrollo local. SIVET es una plataforma web integral diseñada para representar datos históricos, métricas en tiempo real y proyecciones energéticas, integrando múltiples fuentes como sistemas SCADA y estaciones meteorológicas.

1.1 Características del Sistema

- **Arquitectura Desacoplada**: Backend Django + Frontend React
- **Integración SCADA**: Conexión segura a sistemas externos
- **Procesamiento Asíncrono**: Tareas Celery para cálculos automáticos
- **Base de Datos PostgreSQL**: Almacenamiento robusto de datos
- **Redis**: Broker para tareas y caché
- **Autenticación Avanzada**: Sistema de tokens y gestión de sesiones

1.2 Estado del Proyecto

El proyecto SIVET se encuentra en un **100 % de avance**, completando la fase de maduración y refinamiento. Todas las funcionalidades principales están implementadas y funcionando correctamente.

2 Requisitos del Sistema

Antes de iniciar el proceso de instalación, verifique que su equipo cuente con el siguiente software:

2.1 Requisitos Mínimos del Sistema

- **Sistema Operativo**: Windows 10/11, macOS 10.15+, Ubuntu 18.04+
- **Memoria RAM**: Mínimo 8GB, recomendado 16GB
- **Almacenamiento**: Mínimo 10GB de espacio libre
- **Procesador**: Intel i5/AMD Ryzen 5 o superior

2.2 Software Obligatorio

- **Git** (versión 2.30+)
  - Verificar instalación: `git --version`
  - Descarga: https://git-scm.com/downloads
  - Configuración inicial: `git config --global user.name "Su Nombre"`
- **Node.js** (versión 18.x o superior)
  - Verificar: `node -v` y `npm -v`
  - Descarga: https://nodejs.org/
  - **ADVERTENCIA:** No instalar Node.js 20+ en Windows, puede causar problemas de compatibilidad
- **Python** (versión 3.9+)
  - Verificar: `python3 --version`, `pip3 --version`
  - Descarga: https://www.python.org/downloads/
  - **CONSEJO:** En Windows, marcar "Add Python to PATH"durante la instalación
- **PostgreSQL** (versión 12+)
  - Descarga: https://www.postgresql.org/download/
  - Crear base de datos: `createdb sivet_db`
  - Usuario por defecto: `postgres`
- **Redis** (versión 6.0+)
  - Windows: Descargar desde https://github.com/microsoftarchive/redis/releases
  - macOS: `brew install redis`
  - Ubuntu: `sudo apt-get install redis-server`

2.3 Software Opcional pero Recomendado

- **Visual Studio Code**: Editor de código con extensiones Python y React
- **Postman**: Cliente API para pruebas de endpoints
- **DBeaver**: Cliente de base de datos gráfico
- **Redis Desktop Manager**: Cliente gráfico para Redis

3 Proceso de Instalación

3.1 Preparación del Entorno

Antes de comenzar, asegúrese de tener todos los permisos necesarios y cree un directorio de trabajo:

```
# Crear directorio de trabajo
mkdir C:\Projects\SIVET
cd C:\Projects\SIVET

# Verificar herramientas instaladas
git --version
node --version
npm --version
python --version
pip --version
```

3.2 Clonar el Repositorio

```
# Clonar el repositorio
git clone https://github.com/Bura-hub/SIVET_App.git
cd SIVET_App

# Verificar estructura del proyecto
dir
```

3.3 Configuración de la Base de Datos

3.3.1 Instalación de PostgreSQL

1. Descargar e instalar PostgreSQL desde la página oficial
2. Durante la instalación, anotar la contraseña del usuario `postgres`
3. Crear la base de datos SIVET:

```
# Conectar a PostgreSQL
psql -U postgres

# Crear base de datos
CREATE DATABASE sivet_db;

# Verificar creaci n
\l
```

```
# Salir
\q
```

3.3.2 Configuración de Redis

1. Instalar Redis según su sistema operativo
2. Verificar que el servicio esté ejecutándose:

```
# Windows (PowerShell como administrador)
redis-server --service-start

# Verificar estado
redis-cli ping
# Debe responder: PONG
```

3.4 Configuración del Backend (Django)

3.4.1 Entorno Virtual Python

```
# Navegar al directorio del proyecto
cd SIVET_App

# Crear entorno virtual
python -m venv env_lumen

# Activar entorno virtual (Windows)
env_lumen\Scripts\activate

# Activar entorno virtual (Linux/macOS)
source env_lumen/bin/activate

# Verificar activaci n
where python
# Debe mostrar la ruta del entorno virtual
```

3.4.2 Instalación de Dependencias

```
# Actualizar pip
python -m pip install --upgrade pip

# Instalar dependencias desde requirements.txt
pip install -r requirements.txt
```

```
# Verificar instalaci n
pip list
```

3.4.3 Configuración de Variables de Entorno

Crear archivo `.env` en el directorio raíz:

Listing 1: .env

```
# Configuraci n Django
SECRET_KEY='django-insecure-su-clave-secreta-aqui-12345'
DEBUG=True
ALLOWED_HOSTS=localhost,127.0.0.1

# Base de datos PostgreSQL
DATABASE_URL=postgresql://postgres:su_password@localhost:5432/sivet_db

# Redis
REDIS_URL=redis://localhost:6379/0

# Credenciales SCADA (requeridas)
SCADA_USERNAME=su_usuario_scada
SCADA_PASSWORD=su_password_scada

# Configuraci n de correo (opcional)
EMAIL_HOST=smtp.gmail.com
EMAIL_PORT=587
EMAIL_USE_TLS=True
EMAIL_HOST_USER=su_email@gmail.com
EMAIL_HOST_PASSWORD=su_app_password
```

3.4.4 Configuración de Base de Datos

Editar `core/settings.py` para configurar la base de datos:

Listing 2: core/settings.py - Configuración de BD

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'sivet_db',
        'USER': 'postgres',
        'PASSWORD': 'su_password',
        'HOST': 'localhost',
        'PORT': '5432',
```

```
    }
}

# Configuraci n Redis para Celery
REDIS_URL = os.getenv('REDIS_URL', 'redis://localhost:6379/0')
CELERY_BROKER_URL = REDIS_URL
CELERY_RESULT_BACKEND = REDIS_URL
```

3.4.5 Migraciones y Superusuario

```
# Aplicar migraciones
python manage.py makemigrations
python manage.py migrate

# Crear superusuario
python manage.py createsuperuser

# Verificar instalaci n
python manage.py check
```

3.5 Configuración del Frontend (React)

3.5.1 Instalación de Dependencias

```
# Navegar al directorio frontend
cd frontend

# Instalar dependencias
npm install

# Verificar instalacion
npm list --depth=0
```

3.5.2 Configuración de Variables de Entorno

Crear archivo `frontend/.env`:

Listing 3: frontend/.env

```
# URL del backend Django
REACT_APP_API_URL=http://localhost:8000/api

# Configuraci n de desarrollo
REACT_APP_ENV=development
```

```
REACT_APP_DEBUG=true

# URLs de servicios externos
REACT_APP_SCADA_URL=http://localhost:8000/api/scada
REACT_APP_WEATHER_URL=http://localhost:8000/api/weather
```

3.5.3 Configuración de Tailwind CSS

Verificar que Tailwind CSS esté configurado correctamente:

```
# Verificar configuraci n de Tailwind
npx tailwindcss --help

# Construir CSS de producci n
npm run build
```

4 Ejecución de la Aplicación

4.1 Iniciar Servicios de Base de Datos

4.1.1 PostgreSQL

```
# Windows (PowerShell como administrador)
net start postgresql-x64-15

# Verificar estado
sc query postgresql-x64-15
```

4.1.2 Redis

```
# Windows
redis-server --service-start

# Verificar conexi n
redis-cli ping
```

4.2 Servidor Backend Django

4.2.1 Primera Ejecución

```
# Activar entorno virtual
env_lumen\Scripts\activate
```

```
# Verificar configuraci n
python manage.py check

# Ejecutar servidor de desarrollo
python manage.py runserver 0.0.0.0:8000
```

4.2.2 Verificación del Backend

- **Admin Django**: http://localhost:8000/admin/
- **API Root**: http://localhost:8000/api/
- **Documentación API**: http://localhost:8000/api/schema/swagger-ui/

4.3 Servidor Frontend React

```
# En nueva terminal, navegar al frontend
cd frontend

# Iniciar servidor de desarrollo
npm start
```

4.4 Acceso a la Aplicación

- **Frontend**: http://localhost:3000/
- **Backend API**: http://localhost:8000/api/
- **Admin Django**: http://localhost:8000/admin/

4.5 Credenciales de Prueba

- **Usuario**: `admin` (creado durante la instalación)
- **Contraseña**: La que configuró durante `createsuperuser`

5 Configuración de Celery

5.1 Configuración del Worker

```
# Activar entorno virtual
env_lumen\Scripts\activate
```

```
# Ejecutar worker de Celery
celery -A core worker --loglevel=info --pool=solo
```

5.2 Configuración del Beat (Programador)

```
# En nueva terminal, activar entorno virtual
env_lumen\Scripts\activate

# Ejecutar Celery Beat
celery -A core beat --loglevel=info
```

5.3 Verificación de Tareas

- **Monitoreo**: http://localhost:8000/admin/django_celery_beat/periodictask/
- **Estado del Worker**: Verificar logs de Celery
- **Tareas Programadas**: Configurar en Django Admin

6 Configuración Adicional

6.1 Ajuste de Puertos

Si los puertos por defecto están ocupados:

```
# Frontend en puerto 3001
cd frontend
set PORT=3001 && npm start

# Backend en puerto 8001
cd ..
python manage.py runserver 0.0.0.0:8001
```

6.2 Configuración de CORS

Para desarrollo local, verificar configuración en `core/settings.py`:

Listing 4: Configuración CORS

```
CORS_ALLOWED_ORIGINS = [
    "http://localhost:3000",
    "http://127.0.0.1:3000",
    "http://localhost:3001",
    "http://127.0.0.1:3001",
]
```

```
CORS_ALLOW_CREDENTIALS = True
```

6.3 Configuración de Logging

Crear directorio de logs y configurar en `core/settings.py`:

Listing 5: Configuración de Logging

```
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        'file': {
            'level': 'INFO',
            'class': 'logging.FileHandler',
            'filename': 'logs/sivet.log',
        },
    },
    'loggers': {
        'django': {
            'handlers': ['file'],
            'level': 'INFO',
            'propagate': True,
        },
    },
}
```

7 Despliegue en Producción

7.1 Requisitos de Producción

- **Servidor Web**: Nginx o Apache **Servidor WSGI**: Gunicorn o uWSGI
- **Base de Datos**: PostgreSQL con optimizaciones
- **Caché**: Redis con persistencia
- **SSL**: Certificados HTTPS válidos
- **Monitoreo**: Logs centralizados y alertas

7.2 Configuración de Nginx

Archivo de configuración básico:

Listing 6: nginx.conf

```
server {
    listen 80;
    server_name su-dominio.com;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl;
    server_name su-dominio.com;

    ssl_certificate /path/to/cert.pem;
    ssl_certificate_key /path/to/key.pem;

    location /static/ {
        alias /path/to/sivet/static/;
    }

    location /media/ {
        alias /path/to/sivet/media/;
    }

    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

7.3 Configuración de Gunicorn

Archivo de configuración:

Listing 7: gunicorn.conf.py

```
bind = "127.0.0.1:8000"
workers = 3
worker_class = "gevent"
worker_connections = 1000
max_requests = 1000
max_requests_jitter = 50
timeout = 30
keepalive = 2
```

7.4 Script de Inicio del Sistema

Crear servicio systemd (Linux) o script de Windows:

Listing 8: sivet.service

```
[Unit]
Description=SIVET Django Application
After=network.target

[Service]
Type=simple
User=www-data
WorkingDirectory=/path/to/sivet
Environment=PATH=/path/to/sivet/env_lumen/bin
ExecStart=/path/to/sivet/env_lumen/bin/gunicorn core.wsgi:application
Restart=always

[Install]
WantedBy=multi-user.target
```

8 Solución de Problemas Comunes

8.1 Errores de Base de Datos

Error: Connection refused

- Verificar que PostgreSQL esté ejecutándose
- Comprobar credenciales en `.env`
- Verificar puerto (5432 por defecto)

Error: Database does not exist

```
# Conectar a PostgreSQL
psql -U postgres

# Crear base de datos
CREATE DATABASE sivet_db;

# Verificar
\l
```

8.2 Errores de Redis

Error: Redis connection failed

- Verificar que Redis esté ejecutándose
- Comprobar puerto (6379 por defecto)
- Verificar configuración en `.env`

8.3 Errores del Frontend

Error: Categoría no encontrada

- Verificar consola del navegador
- Comprobar respuesta de la API
- Verificar nombres de categorías en la base de datos

Error: Network Error / Failed to fetch

- Verificar que el backend esté ejecutándose
- Comprobar URL en `frontend/.env`
- Verificar configuración CORS
- Comprobar firewall y antivirus

8.4 Errores de Dependencias

Error: Module not found

```
# Frontend
cd frontend
rm -rf node_modules package-lock.json
npm install

# Backend
cd ..
env_lumen\Scripts\activate
pip install -r requirements.txt
```

8.5 Problemas con Entornos Virtuales Python

Error: Virtual environment not activated

```
# Verificar activaci n
where python

# Activar manualmente
env_lumen\Scripts\activate

# Verificar pip
pip --version
```

Entorno virtual dañado

```
# Eliminar entorno da ado
deactivate
rmdir /s env_lumen

# Crear nuevo entorno
python -m venv env_lumen
env_lumen\Scripts\activate
pip install -r requirements.txt
```

8.6 Problemas de Puertos

Error: Port already in use

```
# Verificar procesos en puertos
netstat -ano | findstr :8000
netstat -ano | findstr :3000

# Terminar proceso espec fico
taskkill /PID <PID> /F

# Usar puertos alternativos
python manage.py runserver 8001
npm start -- --port 3001
```

9 Mantenimiento y Actualizaciones

9.1 Actualizaciones del Sistema

```
# Actualizar c digo
git pull origin main

# Actualizar dependencias backend
env_lumen\Scripts\activate
pip install -r requirements.txt

# Actualizar dependencias frontend
cd frontend
npm install

# Aplicar migraciones
cd ..
python manage.py migrate

# Reiniciar servicios
# (Reiniciar Django, Celery y frontend)
```

9.2 Backup de Base de Datos

```
# Crear backup
pg_dump -U postgres sivet_db > backup_$(date +%Y%m%d_%H%M%S).sql

# Restaurar backup
psql -U postgres sivet_db < backup_file.sql
```

9.3 Limpieza de Logs

```
# Limpiar logs antiguos
del logs\*.log

# Limpiar archivos temporales
del *.pyc
rmdir /s __pycache__
```

10 Recursos Adicionales

10.1 Documentación Técnica

- **Django Documentation**: https://docs.djangoproject.com/
- **React Documentation**: https://react.dev/

- **PostgreSQL Documentation**: https://www.postgresql.org/docs/
- **Redis Documentation**: https://redis.io/documentation
- **Celery Documentation**: https://docs.celeryproject.org/

10.2 Herramientas de Desarrollo

- **Postman**: Cliente API para pruebas
- **DBeaver**: Cliente de base de datos
- **Redis Desktop Manager**: Cliente Redis
- **Visual Studio Code**: Editor con extensiones útiles

10.3 Comunidad y Soporte

- **Stack Overflow**: Etiquetas Django, React, PostgreSQL
- **GitHub Issues**: Reportar bugs y solicitar features
- **Discord/Slack**: Canales de la comunidad

11 Conclusión

Este manual proporciona una guía completa y detallada para la instalación y configuración del sistema SIVET en un entorno de desarrollo local. Siguiendo estos pasos cuidadosamente, debería poder tener el sistema funcionando correctamente y acceder a todas las funcionalidades implementadas.

11.1 Estado del Proyecto

SIVET se encuentra en un **100 % de avance**, completando exitosamente la fase de maduración y refinamiento. Todas las funcionalidades principales están implementadas, probadas y funcionando correctamente:

- **Backend Django**: API REST completa con autenticación JWT
- **Frontend React**: Interfaz de usuario responsiva y moderna
- **Base de Datos**: Modelos optimizados y migraciones aplicadas
- **Integración SCADA**: Conexión segura con sistemas externos
- **Procesamiento Asíncrono**: Tareas Celery para cálculos automáticos
- **Reportes**: Generación automática de informes en múltiples formatos

11.2 Verificación de Instalación Exitosa

Para confirmar que la instalación fue exitosa, verifique que pueda:

- **Acceder al Admin Django**: `http://localhost:8000/admin/`
- **Ver la API funcionando**: `http://localhost:8000/api/`
- **Acceder al Frontend**: `http://localhost:3000/`
- **Conectar a PostgreSQL**: Base de datos accesible
- **Conectar a Redis**: Servicio de caché funcionando
- **Ejecutar Celery**: Worker procesando tareas

11.3 Próximos Pasos Recomendados

Una vez que SIVET esté funcionando correctamente:

- **Configurar Datos SCADA**: Sincronizar metadatos de dispositivos
- **Importar Datos Históricos**: Cargar información energética existente
- **Configurar Usuarios**: Crear cuentas para el equipo de trabajo
- **Personalizar Dashboards**: Adaptar visualizaciones a necesidades específicas
- **Configurar Alertas**: Establecer notificaciones automáticas
- **Documentar Procesos**: Crear guías de usuario para el equipo

11.4 Soporte y Mantenimiento

Para obtener soporte técnico o reportar problemas:

- **Email de Soporte**: bura.vent@gmail.com
- **Repositorio GitHub**: `https://github.com/Bura-hub/SIVET_App.git`
- **Documentación Técnica**: Disponible en la carpeta `Docs/`
- **Logs del Sistema**: Revisar archivo `logs/sivet.log`

ADVERTENCIA: Recuerde siempre activar el entorno virtual antes de ejecutar comandos Python y verificar que todos los servicios (PostgreSQL, Redis) estén ejecutándose antes de iniciar la aplicación.

CONSEJO: Para un despliegue en producción, considere usar Docker para contenerización y herramientas de monitoreo como Prometheus y Grafana para supervisar el rendimiento del sistema.

CONSEJO: Mantenga actualizadas las dependencias del proyecto ejecutando regularmente `pip install -r requirements.txt` y `npm install` en el frontend.

¡SIVET está listo para transformar su visualización energética!

Sistema de Visualización Energética Transaccional - Versión 2.0
Universidad de Nariño - Facultad de Ingeniería
2025 - Proyecto de Grado Completado