DOCUMENTO TÉCNICO DE REQUISITOS

Sistema SIVET v2.0

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
Institución:	Universidad de Nariño

Pasto, Nariño, Colombia

Índice

1. Introducción

Este Documento Técnico de Requisitos (DTR) define las especificaciones técnicas, requisitos del sistema y arquitectura para la aplicación software "Sistema de Visualización Energética Transaccional"(SIVET), versión 2.0. El documento establece los fundamentos técnicos para el desarrollo, implementación y mantenimiento del sistema.

1.1. Propósito del Documento

Este documento sirve como especificación técnica autorizada para:

- Definir la arquitectura técnica del sistema
- Establecer requisitos funcionales y no funcionales
- Especificar tecnologías y herramientas de desarrollo
- Definir modelos de datos y APIs
- Establecer estándares de calidad y rendimiento
- Servir como base para la implementación y testing

1.2. Alcance del Sistema

SIVET es una plataforma web integral para monitoreo y análisis de sistemas energéticos que incluye:

- Visualización de datos históricos de consumo y generación
- Análisis de indicadores energéticos y ambientales
- Integración con sistemas SCADA externos
- Procesamiento asíncrono de datos
- Generación de reportes y exportación de datos
- Sistema de autenticación y gestión de usuarios

1.3. Audiencia Técnica

- **Arquitectos de Software**: Para diseño de sistema y toma de decisiones técnicas
- **Desarrolladores**: Para implementación de funcionalidades
- **Ingenieros de DevOps**: Para configuración de infraestructura
- **Ingenieros de QA**: Para definición de criterios de testing
- **Administradores de Sistemas**: Para despliegue y mantenimiento

2. Arquitectura del Sistema

2.1. Arquitectura General

El sistema SIVET implementa una arquitectura desacoplada con los siguientes componentes principales:

- **Frontend**: Aplicación React.js con interfaz de usuario responsiva
- **Backend**: API REST Django con procesamiento asíncrono

- **Base de Datos**: PostgreSQL para datos relacionales
- **Caché**: Redis para sesiones y broker de tareas
- **Procesamiento**: Celery para tareas asíncronas
- **Integración**: Proxy SCADA para datos externos

2.2. Patrones de Diseño

- **MVC**: Modelo-Vista-Controlador en el backend Django
- **Component-Based**: Arquitectura de componentes en React
- **Repository Pattern**: Abstracción de acceso a datos
- **Service Layer**: Lógica de negocio encapsulada
- **Observer Pattern**: Notificaciones y eventos del sistema

2.3. Comunicación entre Componentes

- **HTTP/HTTPS**: APIs RESTful entre frontend y backend
- **JSON**: Formato de intercambio de datos estándar
- **WebSockets**: Para actualizaciones en tiempo real (futuro)
- **Redis Pub/Sub**: Comunicación entre workers de Celery

3. Requisitos Funcionales del Sistema

3.1. RF-001: Gestión de Autenticación y Autorización

- **RF-001.1**: El sistema debe permitir autenticación mediante tokens JWT
- **RF-001.2**: El sistema debe implementar control de acceso basado en roles (RBAC)
- **RF-001.3**: El sistema debe permitir gestión de sesiones múltiples
- **RF-001.4**: El sistema debe implementar rate limiting para prevenir abuso
- **RF-001.5**: El sistema debe registrar eventos de seguridad para auditoría

3.2. RF-002: Gestión de Datos Energéticos

- **RF-002.1**: El sistema debe calcular automáticamente KPIs energéticos mensuales
- **RF-002.2**: El sistema debe procesar datos de consumo eléctrico en tiempo real
- **RF-002.3**: El sistema debe analizar generación de inversores solares
- **RF-002.4**: El sistema debe calcular balance energético neto
- **RF-002.5**: El sistema debe generar alertas por umbrales de consumo

3.3. RF-003: Análisis de Datos Meteorológicos

- **RF-003.1**: El sistema debe procesar datos de estaciones meteorológicas
- **RF-003.2**: El sistema debe calcular correlaciones clima-energía
- **RF-003.3**: El sistema debe generar pronósticos de generación solar
- **RF-003.4**: El sistema debe analizar patrones climáticos históricos

3.4. RF-004: Integración SCADA

- **RF-004.1**: El sistema debe sincronizar metadatos de dispositivos
- **RF-004.2**: El sistema debe obtener mediciones históricas
- **RF-004.3**: El sistema debe manejar fallos de conectividad SCADA
- **RF-004.4**: El sistema debe validar integridad de datos recibidos

3.5. RF-005: Generación de Reportes

- **RF-005.1**: El sistema debe generar reportes en formato CSV
- **RF-005.2**: El sistema debe permitir filtrado por rangos de fechas
- **RF-005.3**: El sistema debe exportar datos en múltiples formatos
- **RF-005.4**: El sistema debe programar reportes automáticos

4. Requisitos No Funcionales

4.1. RNF-001: Rendimiento

- **RNF-001.1**: Tiempo de respuesta de API ¡200ms para operaciones CRUD
- **RNF-001.2**: Tiempo de carga de página ¡3 segundos
- **RNF-001.3**: Procesamiento de 1M+ registros sin degradación
- **RNF-001.4**: Soporte para 1000+ usuarios concurrentes
- **RNF-001.5**: Rendimiento de gráficos a 60 FPS

4.2. RNF-002: Escalabilidad

- **RNF-002.1**: Escalabilidad horizontal mediante balanceo de carga
- **RNF-002.2**: Manejo de 10x incremento en volumen de datos
- **RNF-002.3**: Arquitectura modular para extensibilidad
- **RNF-002.4**: Caché distribuido para alta disponibilidad
- **RNF-002.5**: Base de datos particionable por tiempo

4.3. RNF-003: Seguridad

- **RNF-003.1**: Encriptación HTTPS/TLS 1.3 en producción
- **RNF-003.2**: Tokens JWT con expiración configurable
- **RNF-003.3**: Validación de entrada en todas las APIs
- **RNF-003.4**: Logging de eventos de seguridad
- **RNF-003.5**: Protección contra ataques OWASP Top 10

4.4. RNF-004: Disponibilidad

- **RNF-004.1**: Disponibilidad del 99.5 % en producción
- **RNF-004.2**: Recuperación automática de fallos transitorios
- **RNF-004.3**: Backup automático de base de datos
- **RNF-004.4**: Monitoreo continuo del sistema
- **RNF-004.5**: Plan de recuperación ante desastres

4.5. RNF-005: Mantenibilidad

- **RNF-005.1**: Cobertura de testing mínima del 80 %
- **RNF-005.2**: Documentación técnica completa
- **RNF-005.3**: Logging estructurado para debugging
- **RNF-005.4**: Gestión de dependencias con versiones fijas
- **RNF-005.5**: Código siguiendo estándares PEP 8 y ESLint

5. Especificaciones Técnicas

5.1. Stack Tecnológico Backend

- **Framework Principal**: Django 5.2.4 con Python 3.9+
- **API Framework**: Django REST Framework 3.16.0
- **Base de Datos**: PostgreSQL 12+ con psycopg2-binary 2.9.10 **Caché y Broker**: Redis 6.2.0
- **Tareas Asíncronas**: Celery 5.5.3 con django-celery-beat 2.8.1
- **Documentación API**: drf-spectacular 0.28.0
- **Seguridad**: django-ratelimit 4.1.0, CORS headers
- **Validación**: Validadores personalizados para datos energéticos

5.2. Stack Tecnológico Frontend

- **Framework Principal**: React.js 19.1.0
- **Lenguaje Base**: JavaScript (ES6+) con módulos ES6
- **Estilos**: Tailwind CSS 3.4.17
- **Visualización**: Chart.js 4.5.0 con react-chartjs-2 5.3.0
- **Enrutamiento**: React Router v6
- **Estado**: React Hooks y Context API
- **HTTP Client**: Fetch API nativa con interceptores
- **Testing**: Jest, React Testing Library

5.3. Herramientas de Desarrollo

- **Control de Versiones**: Git con flujo GitFlow
- **Entorno Virtual**: Python venv para aislamiento
- **Gestor de Paquetes**: pip para Python, npm para Node.js
- **Linting**: ESLint, Prettier para JS; Black, Flake8 para Python
- **Testing**: Jest, React Testing Library, Django Test Framework
- **CI/CD**: GitHub Actions para integración continua
- **Contenedores**: Docker y Docker Compose
- **Monitoreo**: Sentry para errores, Prometheus para métricas

6. Modelos de Datos

6.1. Modelos de Indicadores Energéticos

- **MonthlyConsumptionKPI**: KPIs mensuales pre-calculados
- **DailyChartData**: Datos agregados diariamente para gráficos
- **ElectricMeterConsumption**: Consumo acumulado por medidor
- **InverterGeneration**: Generación de inversores solares
- **WeatherStationData**: Datos meteorológicos agregados

6.2. Modelos de Infraestructura SCADA

- **Institution**: Instituciones que poseen dispositivos
- **DeviceCategory**: Categorías de dispositivos
- **Device**: Dispositivos específicos con metadatos
- **Measurement**: Mediciones históricas en formato JSON
- **TaskProgress**: Seguimiento de tareas asíncronas

6.3. Modelos de Autenticación

- **User**: Usuarios del sistema con perfiles
- **UserSession**: Sesiones activas con metadatos
- **UserProfile**: Perfiles extendidos con configuraciones

7. APIs y Endpoints

7.1. Authentication API

- **POST /api/auth/login/**: Autenticación de usuario
- **POST /api/auth/logout/**: Cierre de sesión
- **GET /api/auth/profile/**: Obtener perfil de usuario
- **PUT /api/auth/profile/**: Actualizar perfil de usuario
- **GET /api/auth/sessions/**: Listar sesiones activas

7.2. Indicators API

- **GET /api/indicators/monthly-kpi/**: KPIs mensuales
- **GET /api/indicators/daily-chart/**: Datos diarios para gráficos
- **GET /api/indicators/electric-consumption/**: Consumo eléctrico
- **GET /api/indicators/inverter-generation/**: Generación de inversores
- **GET /api/indicators/weather-data/**: Datos meteorológicos

7.3. SCADA Proxy API

- **GET /api/scada/institutions/**: Listar instituciones
- **GET /api/scada/device-categories/**: Categorías de dispositivos
- **GET /api/scada/devices/**: Dispositivos por institución
- **GET /api/scada/measurements/**: Mediciones históricas
- **POST /api/scada/sync/**: Sincronizar metadatos

7.4. Reports API

- **POST /api/reports/generate/**: Generar reporte
- **GET /api/reports/download/id/**: Descargar reporte
- **GET /api/reports/history/**: Historial de reportes
- **POST /api/reports/schedule/**: Programar reporte automático

8. Configuración del Sistema

8.1. Configuración de Base de Datos

Listing 1: Configuración PostgreSQL

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': os.getenv('DB_NAME', 'sivet_db'),
        'USER': os.getenv('DB_USER', 'postgres'),
        'PASSWORD': os.getenv('DB_PASSWORD'),
        'HOST': os.getenv('DB_HOST', 'localhost'),
        'PORT': os.getenv('DB_PORT', '5432'),
        'OPTIONS': {
            'sslmode': 'require' if not DEBUG else 'disable',
        },
    }
}
```

8.2. Configuración de Redis y Celery

Listing 2: Configuración Celery

```
REDIS_URL = os.getenv('REDIS_URL', 'redis://localhost:6379/0')
CELERY_BROKER_URL = REDIS_URL
CELERY_RESULT_BACKEND = REDIS_URL
CELERY_ACCEPT_CONTENT = ['json']
CELERY_TASK_SERIALIZER = 'json'
CELERY_RESULT_SERIALIZER = 'json'
CELERY_TIMEZONE = 'America/Bogota'
```

8.3. Configuración de Seguridad

Listing 3: Configuración de Seguridad

```
SECRET_KEY = os.getenv('SECRET_KEY')
DEBUG = os.getenv('DEBUG', 'False').lower() == 'true'
ALLOWED_HOSTS = os.getenv('ALLOWED_HOSTS', '').split(',')
```

```
# Configuraci n CORS
CORS_ALLOWED_ORIGINS = [
    "http://localhost:3000",
    "https://sivet.example.com",
]

# Configuraci n de autenticaci n
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'authentication.authentication.JWTAuthentication',
    ],
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated',
    ],
}
```

9. Requisitos de Infraestructura

9.1. Requisitos de Hardware

- **Servidor de Aplicación**: 4+ núcleos CPU, 8GB+ RAM, 50GB+ almacenamiento
- **Servidor de Base de Datos**: 4+ núcleos CPU, 16GB+ RAM, 100GB+ almacenamiento SSD
- **Servidor de Caché**: 2+ núcleos CPU, 4GB+ RAM, 20GB+ almacenamiento
- **Red**: Conexión estable con ancho de banda mínimo 10 Mbps

9.2. Requisitos de Software

- **Sistema Operativo**: Ubuntu 20.04+ LTS, CentOS 8+, Windows Server 2019+
- **Python**: 3.9+ (recomendado 3.11+)
- **Node.js**: 18+ LTS
- **PostgreSQL**: 12+ con extensiones de optimización
- **Redis**: 6+ con persistencia habilitada
- **Nginx**: 1.18+ para proxy inverso
- **Docker**: 20.10+ para contenedorización

9.3. Requisitos de Red

- **Protocolo**: HTTPS/TLS 1.3 para producción
- **Puertos**: 80 (HTTP), 443 (HTTPS), 5432 (PostgreSQL), 6379 (Redis)
- **Firewall**: Configuración para permitir tráfico web y base de datos
- **SSL**: Certificados válidos de autoridad certificadora reconocida

10. Monitoreo y Logging

10.1. Configuración de Logging

Listing 4: Configuración de Logging

```
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'formatters': {
        'verbose': {
            'format': '{levelname} {asctime} {module} {process:d} {thread:
               d} {message}',
            'style': '{',
        },
    },
    'handlers': {
        'file': {
            'level': 'INFO',
            'class': 'logging.handlers.RotatingFileHandler',
            'filename': 'logs/sivet.log',
            'maxBytes': 1024*1024*10,  # 10MB
            'backupCount': 5,
            'formatter': 'verbose',
        },
        'console': {
            'level': 'DEBUG',
            'class': 'logging.StreamHandler',
            'formatter': 'verbose',
        },
    },
    'loggers': {
        'django': {
            'handlers': ['file', 'console'],
            'level': 'INFO',
            'propagate': True,
        },
        'sivet': {
            'handlers': ['file', 'console'],
            'level': 'DEBUG',
            'propagate': True,
        },
    },
}
```

10.2. Métricas de Monitoreo

- **Métricas de Aplicación**: Tiempo de respuesta, tasa de errores, throughput
- **Métricas de Base de Datos**: Conexiones activas, consultas lentas, uso de memoria

- **Métricas de Sistema**: CPU, memoria, disco, red
- **Métricas de Negocio**: Usuarios activos, reportes generados, datos procesados

11. Testing y Calidad

11.1. Estrategia de Testing

- **Testing Unitario**: Cobertura mínima del 80 % para backend y frontend
- **Testing de Integración**: APIs, base de datos, servicios externos
- **Testing de Rendimiento**: Carga, estrés, escalabilidad
- **Testing de Seguridad**: Vulnerabilidades OWASP, autenticación, autorización
- **Testing de Usabilidad**: Experiencia de usuario, accesibilidad

11.2. Herramientas de Testing

- **Backend**: Django Test Framework, pytest, factory-boy
- **Frontend**: Jest, React Testing Library, Cypress
- **APIs**: Postman, Newman para testing automatizado
- **Base de Datos**: pytest-django, testcontainers
- **Cobertura**: coverage.py para Python, istanbul para JavaScript

12. Despliegue y DevOps

12.1. Configuración de Docker

Listing 5: Dockerfile para Backend

```
FROM python:3.11-slim

WORKDIR /app

# Instalar dependencias del sistema
RUN apt-get update && apt-get install -y \
    postgresql-client \
    && rm -rf /var/lib/apt/lists/*

# Copiar requirements e instalar dependencias Python
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copiar c digo de la aplicaci n
COPY . .

# Exponer puerto
EXPOSE 8000
```

```
# Comando de inicio
CMD ["gunicorn", "--bind", "0.0.0.0:8000", "core.wsgi:application"]
```

12.2. Configuración de Nginx

Listing 6: Configuración Nginx

```
server {
    listen 80;
    server_name sivet.example.com;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name sivet.example.com;

    ssl_certificate /etc/ssl/certs/sivet.crt;
    ssl_certificate_key /etc/ssl/private/sivet.key;

    # Configuraci n SSL moderna
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512;

    # Archivos est ticos
    location /static/ {
        alias /var/www/sivet/static/;
        expires 1y;
        add_header Cache-Control "public, immutable";
    }

    # Proxy a Django
    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

13. Glosario Técnico

Término	Definición
API	Interfaz de Programación de Aplicaciones para comunicación entre sistemas
Celery	Sistema de colas de tareas asíncronas para Python
Django	Framework web de alto nivel para Python
JWT	JSON Web Token para autenticación segura
KPI	Indicador Clave de Rendimiento para medir éxito de procesos
PostgreSQL	Sistema de gestión de base de datos relacional de código abierto
React	Biblioteca de JavaScript para construir interfaces de usuario
Redis	Base de datos en memoria para caché y broker de mensajes
SCADA	Sistema de Control y Adquisición de Datos en tiempo real
SIVET	Sistema de Visualización Energética Transaccional

14. Conclusiones

Este Documento Técnico de Requisitos establece las especificaciones técnicas fundamentales para el desarrollo e implementación del sistema SIVET v2.0. Las especificaciones definidas garantizan que el sistema cumpla con los estándares de calidad, rendimiento y seguridad requeridos para un entorno de producción empresarial.

La arquitectura desacoplada, las tecnologías modernas seleccionadas y los requisitos no funcionales establecidos posicionan a SIVET como una solución tecnológica robusta y escalable para el monitoreo y análisis de sistemas energéticos.

¡SIVET: Transformando la Visualización Energética del Futuro!