Requisitos de software

Nombre del proyecto: Rentabilidad de energía solar

Versión: 0.1

Creadores : Semillero de ciencia de datos

Indice

1. Introducción

1.1. Alcance del Producto

El presente documento define los requerimientos para el desarrollo de un backend en Python destinado a la captura, almacenamiento y visualización de datos solares. El sistema permitirá recibir datos en formato JSON a través de una API desplegada en un servidor, procesarlos y almacenarlos en una base de datos para su posterior análisis y visualización.

1.2. Valor del Producto

Este backend proporcionará una solución eficiente para la gestión de datos solares, facilitando la captura automática de información desde aplicaciones externas. Al centralizar y organizar los datos, se mejora la capacidad de análisis y toma de decisiones basadas en datos precisos y actualizados, contribuyendo así a optimizar el uso de recursos solares y a impulsar investigaciones en el campo de la energía renovable.

1.3. Público Objetivo

El sistema está diseñado para:

- **Estudiantes de ciencia de datos:** Que requieren acceder y analizar datos solares para sus estudios.
- **Estudiantes de sistemas:** Que integrarán la API para enviar datos desde diversas fuentes.
- **Clientes y empresas consumidoras de energía solar :** Encargados del mantenimiento y supervisión del backend y su infraestructura.

1.4. Uso Previsto

El backend se utilizará para:

- **Recepción de Datos:** Capturar datos solares enviados desde aplicaciones externas mediante solicitudes POST HTTPS.
- **Almacenamiento de Datos:** Guardar la información recibida en una base de datos segura y estructurada.
- **Procesamiento y Validación:** Asegurar que los datos cumplen con los formatos y tipos esperados antes de su almacenamiento.
- **Visualización y Consulta:** Proveer endpoints para que los usuarios puedan consultar y visualizar los datos almacenados.
- **Seguridad y Control de Acceso:** Implementar mecanismos de autenticación y autorización para proteger la integridad y confidencialidad de los datos.

1.5. Descripción General

El sistema consta de una API desarrollada en Python utilizando un framework adecuado (como FastAPI, Flask o Django, o cualquier otro). La API se desplegará en un servidor VPS, que actuará como el punto de recepción de las solicitudes POST enviadas por la aplicación logger. Los datos recibidos serán validados y almacenados en una base de datos relacional (PostgreSQL o MySQL).. El sistema estará diseñado para ser escalable, seguro y fácil de mantener, permitiendo futuras expansiones y mejoras según las necesidades del proyecto.

2. Requisitos de la Interfaz Externa

3.1. Requisitos de la Interfaz de Usuarios

- **Documentación de la API:**
    - Implementación de Swagger/OpenAPI para facilitar la comprensión y el uso de los endpoints disponibles.
    - Documentación detallada sobre los formatos de solicitud y respuesta, así como ejemplos de uso.

3.2. Requisitos de la Interfaz de Hardware

- **Servidor VPS:**
    - Especificaciones mínimas recomendadas:
        - **CPU:** 2 núcleos
        - **RAM:** 4 GB
        - **Almacenamiento:** 10 GB
        - **Conectividad:** Conexión a Internet de alta velocidad con soporte para HTTPS.
    - Sistema Operativo: Preferiblemente Linux (Ubuntu 20.04 LTS o superior).

    - Acceso SSH seguro para la gestión y despliegue de la aplicación.
- **Base de Datos:**
    - Servidor de base de datos alojado en el mismo VPS o en un servicio gestionado compatible.
    - Respaldo automático de datos para garantizar la integridad y disponibilidad.

3.3. Requisitos de la Interfaz de Software

- **Lenguaje de Programación:** Python 3.8 o superior.
- **Framework:** FastAPI, Flask o Django para el desarrollo de la API.
- **Base de Datos:** PostgreSQL o MySQL.
- **Dependencias y Librerías:**
    - ORM (e.g., SQLAlchemy para Flask/FastAPI, Django ORM para Django).
    - Bibliotecas para manejo de JSON y validación de datos (e.g., Pydantic).
    - Herramientas de autenticación JWT (e.g., PyJWT).
- **Sistema Operativo del Servidor:** Linux (preferiblemente Ubuntu 20.04 LTS o superior).
- **Contenedores y Orquestación (opcional):** Docker para la gestión de contenedores y despliegue.

3.4. Requisitos de la Interfaz de Comunicación

- **Protocolo de Comunicación:**
    - HTTPS para todas las comunicaciones entre la aplicación logger y la API, garantizando la seguridad de los datos en tránsito.
- **Endpoints de la API:**
    - **Recepción de Datos:**
        - **URL:** `https://tu-dominio.com/api/recepcion-datos`
        - **Método HTTP:** POST
        - **Formato de Datos:** JSON
        - **Ejemplo de Solicitud:**
          json

Consideraciones Adicionales

Despliegue de la API:

- **Servidor VPS:** Es recomendable utilizar un proveedor de VPS confiable como AWS EC2, DigitalOcean, o Linode. Configurar el servidor con medidas de seguridad adecuadas, incluyendo firewall, actualizaciones automáticas y acceso restringido.
- **Configuración del Dominio:** Apuntar el dominio registrado al servidor VPS mediante la configuración de DNS adecuada. Implementar certificados SSL mediante Let's Encrypt o proveedores similares para asegurar las comunicaciones.

Base de Datos:

- **Elección de la Base de Datos:** PostgreSQL es altamente recomendable por su robustez y características avanzadas, aunque MySQL también es una opción viable.
- **Migraciones y Versionado:** Utilizar herramientas como Alembic (para SQLAlchemy) o el sistema de migraciones de Django para gestionar cambios en el esquema de la base de datos de manera controlada.

Mantenimiento y Monitoreo:

- **Monitoreo de Rendimiento:** Utilizar herramientas como Prometheus y Grafana para monitorear el rendimiento de la API y la base de datos.
- **Gestión de Logs:** Implementar un sistema de logging estructurado para facilitar la depuración y el análisis de incidentes, utilizando herramientas como ELK Stack (Elasticsearch, Logstash, Kibana).

Escalabilidad:

- **Contenedorización:** Utilizar Docker para facilitar el despliegue y la escalabilidad de los componentes del sistema.