Arquitectura de Software (DAS)

Proyecto RecyBear

Índice

Introducción

Propósito

El propósito del proyecto de RecyBear consiste en facilitar el reciclaje mediante la digitalización del proceso de recolección y clasificación de materiales reciclables, integrando pesas digitales y un sistema de localización para puntos verdes.

Alcance

Este sistema permite a los usuarios registrar sus reciclajes mediante balanzas, visualizar sus datos en una interfaz amigable, y ubicar puntos verdes cercanos a través de un mapa interactivo. Los administradores gestionan los datos de los usuarios y puntos verdes desde una interfaz separada.

Se incentiva su uso con un sistema de metas y premios a los usuarios, esto se medirá con un sistema de puntos los cuales serán asignados dependiendo de la cantidad de puntos obtenidos por el reciclaje para poder ser usados dentro de la tienda de RecyBear.

Objetivos

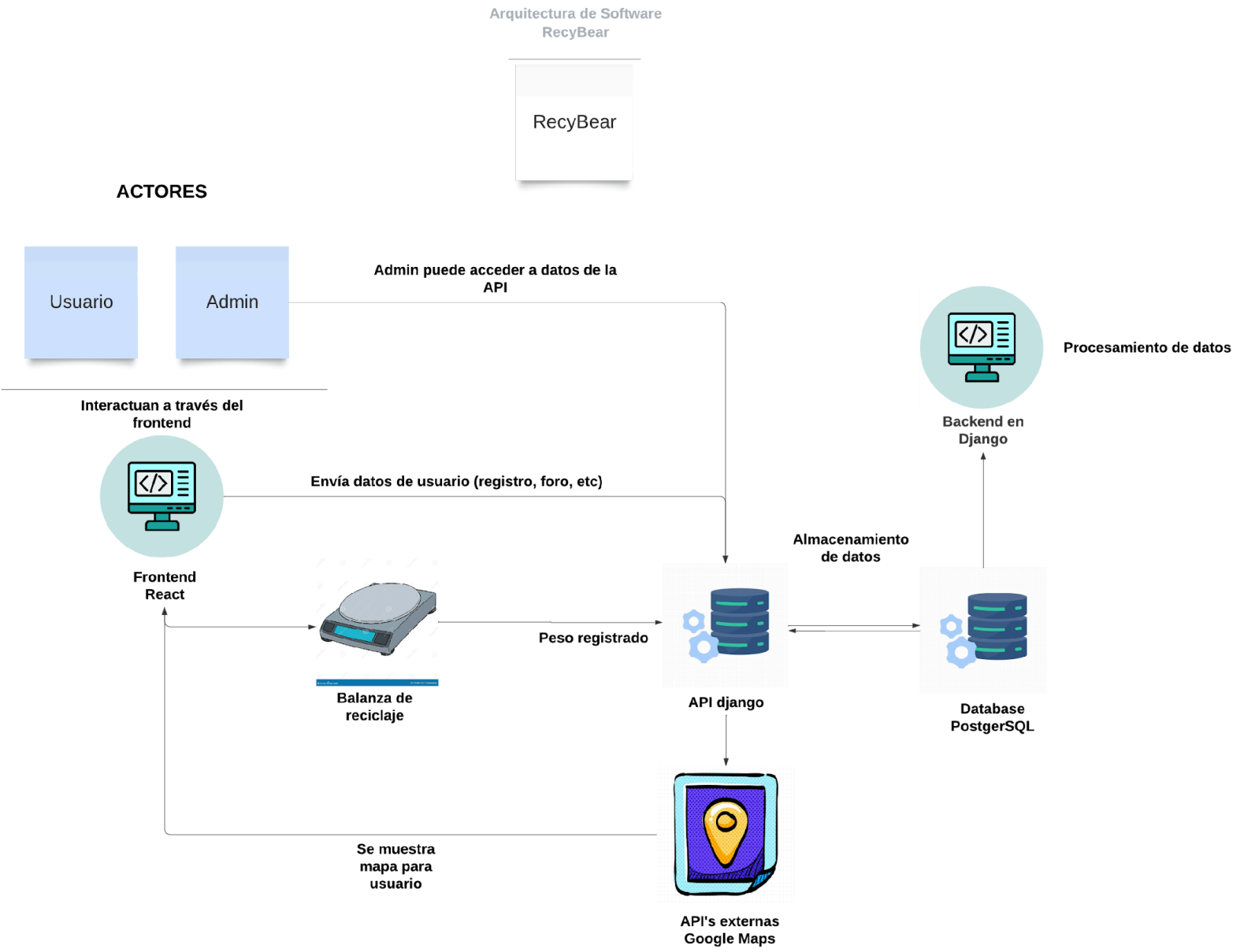
Se tienen en cuenta los siguientes objetivos principalmente:

A) Facilitar el registro digital del peso de materiales reciclados y de esta manera impactar de manera positiva al medio ambiente.
B) Proveer una interfaz intuitiva para usuarios y administradores para su uso.
C) Optimizar el acceso a información sobre puntos verdes y materiales reciclables para los usuarios.

Arquitectura General

Se muestra el diagrama realizado para la arquitectura del proyecto de RecyBear. Como explicación, se puede verificar que la arquitectura estará compuesto por:

a) **Frontend (React):** Interfaz de usuario para clientes y administradores.
b) **Backend (Django):** API para gestionar datos de usuarios, reciclaje y puntos verdes.
c) **Base de datos (PostgreSQL):** Almacenamiento de datos persistente.
d) **Balanza digital:** Dispositivo para registrar pesos de reciclaje en tiempo real.
e) **Integración de Google Maps:** Ubicación y visualización de puntos verdes.



Detalle de componente

1. Frontend.
    a) **Tecnología**: React
    b) **Responsabilidad**: Mostrar la interfaz de usuario y enviar solicitudes al backend.
    c) **Interacciones**:
    - Los usuarios envían datos de reciclaje y consultan información de puntos verdes.
    - Los usuarios pueden realizar envío de información de sus cuentas al ser creadas o modificadas, la creación de publicaciones en el foro o comentarios dentro de la misma.
    - Los administradores gestionan datos desde su panel.
2. Backend
    a) **Tecnología**: Django
    b) **Responsabilidad**: Procesar solicitudes, interactuar con la base de datos y coordinar la integración con balanzas y mapas
    c) **Endpoints principales**:
        - Registro de usuarios.
        - Envío de datos de reciclaje.
        - Gestión de puntos verdes.
3. Base de datos
    a) **Tecnología**: PostgreSQL
    b) **Responsabilidad**: Almacenar datos de usuarios, reciclajes, puntos verdes y registros de pesos.
4. API (Django)
    a) **Tecnología**: Django
    b) **Responsabilidad**: Conecta la plataforma con la base de datos, toda información subida y manejada por la plataforma pasa a través de la API y ésta la enviará a la base de datos.
5. Balanza Digital
    a) **Función**: Capturar el peso del material reciclable.
    b) **Tecnología**: Hardware, arduino y módulo HX711.
    c) **Interacción**: Los datos de la balanza se envían al frontend y luego al backend para su almacenamiento y procesamiento.
6. Integración de mapas
    a) **Herramientas**: API de Google Maps.
    b) **Función**: Visualización de los puntos verdes en tiempo real.

Principios de Diseño

Se tomaron en cuenta los siguientes puntos a mencionar para el diseño de la arquitectura de software presentada:

1. Escalabilidad: El sistema está diseñado para soportar un aumento de usuarios y puntos verdes en el futuro.
2. Modularidad: Cada componente puede modificarse o sustituirse sin afectar al resto del sistema.
3. Seguridad: Los datos de los usuarios se transmiten y almacenan de forma segura.

Con estos aspectos en mente, se decide que la arquitectura diseñada es la más óptima para cumplir con lo necesitado para el proyecto de RecyBear.

También, para justificar el razonamiento detrás del diseño, se toma en cuenta lo siguiente:

1. React para el frontend: Eficiencia en el desarrollo y alta personalización para interfaces. También es uno de los lenguajes mejor documentados lo que permitiría a los desarrolladores dar solución a inconvenientes o errores.
2. Django para el backend y la API: Lenguaje ideal para el manejo de la lógica compleja y las consultas a base de datos requerida para el desarrollo de RecyBear.
3. PostgreSQL como base de datos: Por su escalabilidad y manejo robusto de los datos relacionales que serían enviados.

De esta manera podemos razonar que se tomó la decisión correcta con el diseño de la arquitectura.

Requerimientos No Funcionales

Al diseñarse esta arquitectura, se tomaron en cuenta los siguientes requerimientos a tomar en cuenta, los cuales se proyectan serán cumplidos perfectamente:

1. Disponibilidad: Se espera que la plataforma se encuentre operativo el 99% del tiempo.
2. Rendimiento: Procesar datos de reciclaje y mapas en tiempo real sin retrasos perceptibles.
3. Escalabilidad: Soportar grandes cantidades de usuarios iniciales dentro de la plataforma.

Conclusión

Tomando en cuenta la información de este documento, se puede concluir que la arquitectura propuesta para el proyecto de RecyBear proporciona una solución integral para gestionar el flujo de información que se espera manejar dentro de la plataforma. Con el uso de las tecnologías como React para el frontend, Django para el backend y la API; y postgreSQL para la base de datos, esto permitirá una comunicación fluida entre los diferentes componentes, garantizando también que las funcionalidades con dispositivos externos como la balanza de peso y los servicios de Google Maps funcionen correctamente.

El diseño modular asegura también que cada componente del sistema sea independiente y escalable, permitiendo futuras expansiones en caso de ser requeridas o cambios sin comprometer la estabilidad del sistema. Además también permite priorizar aspectos como seguridad de datos, la disponibilidad constante del sistema y un rendimiento óptimo, tanto en escenarios de baja como de alta demanda de uso por parte de los usuarios.

En conclusión, la arquitectura propuesta para RecyBear satisface todas las necesidades del proyecto, también está alineada con los principios de diseño moderno. Ante esto, se puede decir que el proyecto se encuentra preparado para ofrecer una buena experiencia para sus usuarios de ser cumplido todo lo ya mostrado.