Justificación del Proyecto

El cambio climático, la escasez de recursos naturales y el crecimiento de la huella ecológica hacen urgente la adopción de hábitos sostenibles en la vida cotidiana. Sin embargo, muchas personas no cuentan con herramientas personalizadas que los motiven, orienten y acompañen en este proceso. En este contexto, SmartEcoWatch surge como una solución tecnológica orientada a promover hábitos ecológicos de manera práctica y automatizada.

Este proyecto se justifica por tres ejes principales:

1. Tendencia creciente de wearables: El uso de relojes inteligentes y dispositivos vestibles está en auge, con usuarios cada vez más familiarizados con recibir notificaciones, alertas y estadísticas en tiempo real.
2. Falta de aplicaciones con enfoque ambiental: Aunque existen apps de salud o productividad, hay un vacío en el mercado de soluciones enfocadas en fomentar conductas responsables con el planeta de forma accesible y lúdica.
3. Capacidad tecnológica actual: Con herramientas modernas como Node.js, Express, Prisma y PostgreSQL (Supabase), es posible construir una API robusta, escalable y segura, mientras que Kotlin con Jetpack Compose permite desarrollar experiencias nativas fluidas para Wear OS.

Descripción Técnica General

El sistema estará compuesto por dos capas principales:

1. Frontend (Cliente Wearable)

- Tecnología: Kotlin + Jetpack Compose para Wear OS.
- Funcionalidades clave:
    - Registro e inicio de sesión.
    - Visualización de hábitos diarios asignados.
    - Recordatorios programados.
    - Recomendaciones ecológicas adaptadas al perfil.
    - Interacción rápida (ej. marcar hábito como completado).

2. Backend (API REST)

- Tecnología: Node.js + Express + Prisma ORM.
- Base de datos: Supabase (PostgreSQL).
- Arquitectura:
    - Modular: separación clara por rutas, controladores, servicios y repositorios.
    - Escalable: lista para usar contenedores, middlewares y autenticación por tokens.
- Endpoints esperados:

- /users, /habits, /profile, /recommendations, /interactions
- Gestión de usuarios, hábitos, historial, recomendaciones personalizadas, etc.

3. Modelo de datos (Prisma)

Incluye entidades clave como:

- User, Habit, UserHabit, Recommendation, Interaction, Profile.
- Relaciones bien definidas para permitir consultas eficientes, estadísticas por usuario y mejoras progresivas en la personalización.

Seguridad y Privacidad

El sistema usará tokens JWT para autenticación segura en el wearable. Los datos del usuario estarán cifrados y protegidos bajo buenas prácticas de seguridad web y políticas claras de privacidad, cumpliendo con los lineamientos recomendados por Supabase y la OWASP.

Escalabilidad y Mantenibilidad

El uso de Prisma ORM facilita el mantenimiento del esquema, relaciones y migraciones en la base de datos. La arquitectura basada en capas del backend permite reemplazar componentes individuales sin afectar el resto del sistema. La interfaz en Kotlin es reactiva y optimizada para rendimiento en dispositivos de bajo consumo, como los smartwatches.

Pruebas y Validación

Se planean pruebas unitarias (Jest) para los servicios y middlewares, pruebas de integración para los endpoints principales, y pruebas de experiencia de usuario en el emulador y dispositivos reales con Wear OS.