
# Stack Tecnológico

Grupo 7

ISPP-MapYourWorld

Devising a project


Alfonso Alonso, Alejandro Aragón, José María Baquero, Pablo Caballero, Ricardo Carreño, Franco Dell 'Aguila, Alberto Escobar, Jaime Gómez, Claudio González, Ángel Neria, Pablo Olivencia, Antonio Porcar, Alba Ramos, Pedro Pablo Santos, Manuel Vélez, Gonzalo García.

15/02/2025

| CONTROL DE VERSIONES | | | |
|---|---|---|---|
| VERSIÓN | FECHA | COMENTARIOS | AUTOR |
| v1 | 15/02/2025 | Creación del documento | Ángel Neria<br>Pedro Pablo Santos |
| V1.1 | 19/02/2025 | Revisión del documento | Antonio Porcar y Ricardo Carreño |

# Tabla de contenido

# 1. Introducción

El presente documento detalla la selección del stack tecnológico utilizado en el desarrollo del proyecto, justificando cada tecnología en función de su rendimiento, escalabilidad, facilidad de uso y compatibilidad con los requerimientos del sistema. Se ha priorizado la eficiencia en términos de desarrollo, mantenimiento y despliegue.

# 2. Backend

## 2.1. Tecnología seleccionada: Node.js con Express

Node.js ha sido elegido, como la base del backend por su rapidez en la ejecución de código, su ecosistema robusto de librerías y la facilidad de desarrollo en un entorno asíncrono y no bloqueante. Express, como framework, simplifica la creación de API REST y permite una integración eficiente con otras herramientas.

**Ventajas de Node.js y Express:**

- Compilación rápida, facilitando iteraciones ágiles en el desarrollo.
- Amplio ecosistema de librerías en npm, reduciendo tiempos de implementación.
- Flexibilidad y compatibilidad con otros lenguajes y tecnologías.
- Capacidad de manejar un alto volumen de peticiones concurrentes.

## 2.2. Comunicación interna entre microservicios: Apache Kafka

Para garantizar una comunicación eficiente entre microservicios, se ha optado por Apache Kafka, un sistema de mensajería distribuida que permite el procesamiento en tiempo real de eventos y la integración escalable de múltiples servicios.

**Razones para elegir Apache Kafka:**

- Manejo eficiente de grandes volúmenes de datos en tiempo real.
- Baja latencia en la comunicación entre servicios.
- Escalabilidad horizontal y alta disponibilidad.
- Soporte para integración con múltiples tecnologías.

# 3. Frontend

## 3.1. Tecnología seleccionada: React Native con Tailwind CSS

El desarrollo del frontend se ha basado en **React Native** debido a su capacidad de generar aplicaciones móviles nativas para Android e iOS con una sola base de código.

La integración con **Tailwind CSS** permite una estilización rápida y eficiente gracias al repertorio de componentes que posee.

**Ventajas de React Native:**

- Reutilización de código entre plataformas móviles.
- Gran comunidad y soporte en la web.
- Amplia cantidad de librerías y componentes predefinidos.
- Integración sencilla con backend basado en Node.js.

**Beneficios de usar Tailwind CSS:**

- Evita la sobrecarga de archivos CSS adicionales.
- Facilita la implementación de un diseño modular y escalable.
- Mejora la productividad del equipo gracias a su sintaxis simplificada.

# 4. Base de Datos

## 4.1. Tecnología seleccionada: PostgreSQL con la extensión PostGIS

Para el almacenamiento de datos, se ha optado por **PostgreSQL**, un sistema de gestión de bases de datos relacional altamente confiable y escalable. La integración con **PostGIS** permite realizar consultas geoespaciales avanzadas.

**Ventajas de PostgreSQL y PostGIS:**

- Soporte para datos espaciales, ideal para aplicaciones basadas en mapas y geolocalización.
- Escalabilidad y rendimiento óptimo en grandes volúmenes de datos.
- Capacidad de realizar análisis geoespaciales complejos.
- Fuerte comunidad y documentación extensa.

# 5. Despliegue

## 5.1. Plataforma seleccionada: Raspberry Pi

El despliegue del sistema se realizará en un entorno **Raspberry Pi**, lo que proporciona una infraestructura de bajo costo y consumo energético.

**Motivos para elegir Raspberry Pi:**

- Permite realizar pruebas en entornos locales antes de escalar.
- Bajo consumo energético, reduciendo costos operativos.
- Suficiente capacidad de procesamiento para aplicaciones ligeras.

- Flexibilidad para configuraciones personalizadas.

**Despliegue de la landing page:**

El despliegue de la landing page se realizará en Vercel. Se ha elegido esta solución por su simplicidad e idoneidad para desplegar una página web que únicamente está compuesta de una capa de frontend en React Native.

# 6. Pruebas y Validación

## 6.1. Pruebas en Backend

Para garantizar la estabilidad del backend, se definirán pruebas con **Jest**, un framework de testing ampliamente utilizado en el ecosistema de JavaScript.

**Beneficios de Jest:**

- Pruebas unitarias y de integración rápidas y eficientes.
- Fácil configuración con Node.js y Express.
- Soporte para mocks y pruebas asíncronas.

## 6.2. Pruebas en Frontend

El frontend será validado con **Jest**, **React Testing Library** y **Postman** para verificar la comunicación con el backend.

**Razones para esta selección:**

- **Jest:** Facilita la ejecución de pruebas unitarias en componentes React Native.
- **React Testing Library:** Mejora la confiabilidad de las pruebas en UI.
- **Postman:** Permite validar el correcto funcionamiento de las API REST.

# 7. Metodología de Implementación

Para optimizar el desarrollo y asegurar la calidad del código, se ha establecido una metodología que incluye las siguientes fases:

1. **Definición de módulos:** Se identifican las funcionalidades clave del sistema antes de repartir tareas.
2. **Distribución equitativa del equipo:** Se asignan desarrolladores de backend y frontend de manera balanceada.
3. **Validación interna:** Cada equipo prueba sus propias implementaciones antes de la entrega a testers.
4. **Revisión por el equipo de testing:** Se ejecutan pruebas exhaustivas antes del despliegue final.
5. **Despliegue progresivo:** Implementación gradual para minimizar riesgos.