

Proyecto de Desarrollo - Herramientas

Juan David Ocampo - 38402

David Rincón Toro - 20992

Kevin López Salazar - 28855

Sebastián Salazar Güiza - 906

Bryan Cartagena - 5928

Presentado a:

Sandra Hurtado Gil

Ingeniería de Software 3

Universidad de Caldas

Facultad de Ingenierías e Inteligencia Artificial

Manizales, Colombia

2025

Tabla de Contenidos

Tabla de Contenidos.....	2
1. Introducción.....	3
2. Criterios de Evaluación.....	3
3. Matriz Comparativa.....	4
a. Frontend.....	4
Framework seleccionado: React Native.....	4
Justificación técnica:.....	4
Conclusión:.....	5
b. Backend.....	5
Framework seleccionado: Nest.js.....	6
Justificación técnica:.....	6
En comparación:.....	6
Conclusión:.....	7
4. Conclusión general.....	7

1. Introducción

Para garantizar el éxito del desarrollo del proyecto, se realizó un análisis comparativo de diferentes herramientas de frontend y backend. Se definieron criterios técnicos y funcionales relevantes, asignando pesos según su importancia para el equipo de desarrollo. Finalmente, se seleccionaron las herramientas con mayor puntaje total: React Native para el frontend y un framework minimalista de Node.js (Express.js) para el backend.

2. Criterios de Evaluación

Criterio	Descripción	Peso (%)
Facilidad de uso	Qué tan rápido puede adoptarse por el equipo.	20
Comunidad y soporte	Disponibilidad de documentación y foros activos.	20
Escalabilidad	Capacidad de crecer con nuevas funcionalidades.	15
Rendimiento	Eficiencia en la ejecución y carga.	15
Compatibilidad	Integración con otros módulos o librerías.	10
Mantenimiento	Facilidad para actualizar y corregir errores.	10
Popularidad / Estabilidad	Nivel de adopción en la industria.	10

3. Matriz Comparativa

a. Frontend

Criterio	Peso	React Native	Flutter	Ionic
Facilidad de uso	20	4	3	3
Comunidad y soporte	20	5	4	3
Escalabilidad	15	4	4	3
Rendimiento	15	4	5	3
Compatibilidad	10	5	4	4
Mantenimiento	10	4	3	3
Popularidad / Estabilidad	10	5	4	3
Puntaje total (ponderado)	100	4.4 / 5	3.9 / 5	3.2 / 5

Framework seleccionado: React Native

Justificación técnica:

- **Desarrollo nativo multiplataforma:** React Native permite crear aplicaciones móviles para Android e iOS desde una sola base de código en **JavaScript**, optimizando el tiempo y esfuerzo del desarrollo.
- **Alto rendimiento:** Aunque Flutter ofrece gran velocidad, React Native logra un rendimiento muy competitivo gracias al uso de componentes nativos y una arquitectura optimizada.
- **Ecosistema unificado:** Su compatibilidad con **Node.js** y bibliotecas de JavaScript facilita la integración con el backend y otras herramientas.
- **Amplia comunidad:** Desarrollado por Meta, cuenta con una comunidad global muy activa y constante actualización.
- **Interfaz moderna:** Facilita la creación de interfaces fluidas, con estilos personalizables y soporte de librerías visuales populares.

Conclusión:

React Native alcanzó el mayor puntaje ponderado (4.6/5), destacando por su equilibrio entre rendimiento, compatibilidad y facilidad de integración con el entorno [Node.js](#).

b. Backend

Criterio	Descripción	Peso (%)	Express.js (Node.js)	NestJS (Node.js)	Fastify (Node.js)	Flask (Python)	AdonisJS (Node.js)	Spring Boot (Java)
Facilidad de uso	Facilidad de aprendizaje y rapidez en la construcción de APIs.	20	5	4	4	3	4	2
Comunidad y soporte	Tamaño de la comunidad, documentación y soporte activo.	20	5	5	4	4	3	4
Escalabilidad	Capacidad de crecer con el proyecto sin pérdida de rendimiento.	15	4	5	4	3	4	5
Rendimiento	Velocidad y eficiencia en el manejo de peticiones.	15	5	5	5	3	4	3
Compatibilidad	Integración con librerías, bases de datos y servicios externos.	10	5	5	4	4	4	3

Mantenimiento	Facilidad para actualizar dependencias y gestionar errores.	10	4	5	4	5	3	3
Popularidad / Estabilidad	Nivel de adopción y madurez del framework.	10	5	5	4	4	3	5
Puntaje total (ponderado)		100	4.7 / 5	4.8 / 5	4.3 / 5	3.8 / 5	3.7 / 5	3.9 / 5

Framework seleccionado: Nest.js

Justificación técnica:

- **Estructura Escalable:** Impone una arquitectura modular (Módulos, Servicios, Controladores) que evita el desorden típico de los proyectos grandes en Node.js.
- **Seguridad de Tipos:** Construido con TypeScript de forma nativa. Captura errores antes de la ejecución, garantizando confiabilidad en producción.
- **Código Reutilizable:** Inyección de Dependencias (DI) nativa. Facilita las pruebas unitarias y el desacoplamiento de la lógica de negocio..
- **Patrones de Diseño:** Inspirado en frameworks maduros (Angular, Spring). Ofrece un camino claro y estándar para desarrolladores de software empresarial.
- **Motor Intercambiable:** Puede usar Express.js o Fastify como motor subyacente. Permite optimizar el rendimiento sin reescribir la lógica.

En comparación:

- **Express.js** es más simple pero se desorganiza rápidamente en proyectos grandes.
- **Fastify** es muy rápido, pero no ofrece la estructura ni los patrones de diseño de NestJS.

- **Flask y Spring Boot** usan otros lenguajes, lo que complica la uniformidad del stack (frontend/backend).
- **AdonisJS** es completo, pero con una comunidad y adopción mucho menores que NestJS.

Conclusión:

NestJS es la opción más conveniente para proyectos que requieren escalabilidad, orden y mantenibilidad a largo plazo. Ofrece la **combinación perfecta** entre la **flexibilidad** de Node.js y la **robustez** de un framework de nivel empresarial, garantizando un **desarrollo sólido y profesional**.

4. Conclusión general

Tras la evaluación comparativa, las herramientas seleccionadas para el desarrollo del proyecto son:

Área	Herramienta seleccionada	Justificación principal
Frontend	React Native	Desarrollo móvil nativo multiplataforma, alta compatibilidad con Node.js y fuerte comunidad.
Backend	Nest.js	Framework robusto y flexible que ofrece una estructura completa para mejorar el mantenimiento y escalabilidad.

Estas herramientas ofrecen una arquitectura moderna, escalable y coherente, permitiendo un desarrollo ágil con un único lenguaje (**TypeScript**) en todo el stack tecnológico.