



Documento Técnico: Selección de Tecnologías

Elaborado por:

Team Sobrecupo

David Ladino

Jorge Cuadrado

Camilo D'Aleman

Smith Yañez

PROYECTO - CLEAN CODE

LENGUAJE Y FRAMEWORK DE PROGRAMACIÓN

Las tecnologías elegidas para este proyecto son: Electron.js (JavaScript + Node.js) con React.js para el front-end.

<i>Tecnología</i>	<i>Ventajas</i>	<i>Desventajas</i>
Electron.js (seleccionada)	<ul style="list-style-type: none">- Permite crear apps de escritorio multiplataforma con tecnologías web (HTML, CSS, JS).- Gran comunidad y documentación.- Fácil integración con React y Node.js.- Curva de aprendizaje baja para programadores con conocimientos básicos de JS.- Ideal para equipos con experiencia heterogénea en lenguajes.	<ul style="list-style-type: none">- Mayor consumo de memoria que apps nativas.- No tan eficiente para apps que requieren alto rendimiento gráfico.
Java + JavaFX	<ul style="list-style-type: none">- Muy estable, multiplataforma nativa, excelente para apps robustas.- Buen soporte a patrones de diseño.	<ul style="list-style-type: none">- Curva de aprendizaje más alta.- El equipo no domina Java.- Desarrollo más lento.



C# + WPF

- Excelente rendimiento en Windows.
- Buen soporte de herramientas (Visual Studio).
- No multiplataforma.
- Curva de aprendizaje mayor para el equipo.
- Requiere entorno Windows.

Python + PyQt

- Rápida creación de prototipos.
- Gran comunidad en IA y procesamiento de datos.
- Interfaz menos moderna.
- Mayor dificultad para empaquetar ejecutables multiplataforma.

Se optó por [Electron.js](https://electronjs.org/) por las siguientes razones:

1. El equipo no tenía un lenguaje de programación en común, entonces, se optó por JS que es más accesible para el aprendizaje colaborativo.
2. Permite una interfaz moderna con React.js y se ajusta al objetivo de hacer la app atractiva para usuarios turísticos.
3. Según la estructura de la aplicación (monolítica) y el tamaño de la aplicación, el consumo de recursos no representa, inicialmente, un problema.
4. Se alinea con los principios de ingeniería de software, tiene una documentación robusta e integración con otras herramientas para futuras funcionalidades/modificaciones/escalabilidad.

BASE DE DATOS

La base de datos empleada es SQLite.

<i>Base de datos</i>	<i>Ventajas</i>	<i>Desventajas</i>
SQLite (seleccionada)	<ul style="list-style-type: none">- Ligera y embebida, ideal para aplicaciones de escritorio.- No requiere servidor.- Excelente para sistemas pequeños y medianos.- Fácil de integrar con Electron.	<ul style="list-style-type: none">- No apta para sistemas de alta concurrencia o escalabilidad grande.
MySQL	<ul style="list-style-type: none">- Amplio soporte y escalabilidad moderada.	<ul style="list-style-type: none">- Requiere servidor, aumenta complejidad de despliegue.
PostgreSQL	<ul style="list-style-type: none">- Muy robusta, soporte avanzado de datos.	<ul style="list-style-type: none">- Excesiva para un proyecto pequeño y local.

La tecnología de base de datos seleccionada es perfecta según las especificaciones dadas para el proyecto y tiene una simbiosis con el lenguaje y framework seleccionados.



BIBLIOTECAS Y HERRAMIENTAS COMPLEMENTARIAS

1. *React.js*: para crear una interfaz moderna e interactiva, facilitando una buena UX para turistas.
2. *JWT (JSON Web Tokens)*: para autenticación local segura (opcional si se requiere manejo de perfiles de usuarios).
3. *Git + GitHub*: control de versiones y colaboración en equipo, imprescindible para el trabajo colaborativo.
4. *Copilot AI*: asistencia para acelerar el desarrollo, especialmente útil dado el bajo nivel de experiencia del equipo.

El uso de estas herramientas responde a la necesidad pedagógica del curso (buenas prácticas: Clean Code, testing) y mejora la colaboración entre un equipo con experiencia dispar.

ALINEACIÓN CON LOS OBJETIVOS DEL CURSO Y NECESIDADES DEL PROYECTO

1. Principios de Ingeniería de Software: La combinación Electron + React + SQLite permite aplicar patrones de diseño, testing, y Clean Code de forma pedagógica y realista para un equipo novato.
2. Capacidades del equipo: JS es más accesible y con Copilot se reduce la curva de aprendizaje.
3. Objetivos del proyecto: La interfaz moderna atraerá a usuarios interesados en turismo comunitario; la arquitectura monolítica facilita el despliegue rápido en zonas con poca conectividad.