Universidad Nacional de Colombia - sede Bogotá
Facultad de Ingeniería
Departamento de Sistemas e Industrial
Curso: **Ingeniería de Software 1 (2016701)**
Estudiantes: Juan Camilo D'Aleman Rodriguez, David Ladino, Jorge Cuadrado, Smith Yañez.

Colombia Raíces - Tecnologías

1. Lenguaje De Programación Y Framework Principal

Se ha seleccionado JavaScript con Electron.js como framework principal.

Criterio	Electron.js + React	Java + JavaFX	C# + WPF	Python + Tkinter
Desarrollo Multiplataforma	✅ Excelente	⚠️ Bueno	❌ Solo Windows	⚠️ Limitado
Ecosistema de Librerías	✅ Muy Rico	✅ Rico	⚠️ Medio	⚠️ Limitado
Rendimiento	⚠️ Medio	✅ Alto	✅ Alto	❌ Bajo
Interfaz Moderna	✅ Excelente	⚠️ Bueno	⚠️ Bueno	❌ Limitado
Comunidad y Soporte	✅ Muy Grande	✅ Grande	⚠️ Medio	⚠️ Medio
Facilidad de Mantenimiento	✅ Alta	⚠️ Media	⚠️ Media	❌ Baja

Ventajas de Electron.js + React

- **Desarrollo Unificado**: Una sola base de código para todas las plataformas
- **Tecnologías Web Familiares**: HTML, CSS, JavaScript conocidos
- **Ecosistema Rico**: Miles de librerías disponibles en NPM
- **Interfaz Moderna**: Capacidad de crear UIs atractivas con CSS/Tailwind
- **Desarrollo Rápido**: Prototipado y desarrollo ágil
- **Debugging Familiar**: Herramientas de desarrollo web conocidas

Desventajas de Electron.js + React

- **Consumo de Memoria**: Mayor uso de RAM que aplicaciones nativas
- **Tamaño del Ejecutable**: Archivos más grandes (incluye Chromium)
- **Rendimiento**: Ligeramente inferior a aplicaciones nativas

- **Dependencia de Node.js**: Requiere runtime de Node.js

Justificación De La Selección

Para las Necesidades del Proyecto

- **Multiplataforma**: Colombia Raíces debe funcionar en Windows, macOS y Linux
- **Interfaz Atractiva**: El turismo requiere interfaces visualmente llamativas
- **Mapas Interactivos**: Integración fácil con librerías como Leaflet
- **Desarrollo Rápido**: Tiempo limitado del proyecto académico

Para las Capacidades del Equipo:

- **Conocimiento Previo**: JavaScript es más accesible para principiantes
- **Curva de Aprendizaje**: Menor tiempo de adaptación
- **Recursos de Aprendizaje**: Abundante documentación y tutoriales

Para los Objetivos del Curso:

- **Arquitectura Monolítica**: Electron permite una estructura monolítica clara
- **Patrones de Diseño**: Implementación clara de MVC, Repository, Singleton
- **Pruebas**: Excelente soporte para testing con Jest
- **Ciclo de Desarrollo**: Facilita todas las fases del desarrollo

2. Base De Datos Relacional

Se ha seleccionado SQLite para base de datos

Criterio	SQLite	PostgreSQL	MySQL	SQL Server
Facilidad de Instalación	✅ Sin instalación	❌ Compleja	❌ Compleja	❌ Compleja
Portabilidad	✅ Archivo único	❌ Servidor requerido	❌ Servidor requerido	❌ Servidor requerido
Rendimiento (Pequeña escala)	✅ Excelente	⚠️ Bueno	⚠️ Bueno	⚠️ Bueno
Concurrencia	⚠️ Limitada	✅ Excelente	✅ Excelente	✅ Excelente
Soporte SQL	✅ Completo	✅ Completo	✅ Completo	✅ Completo

Criterio	SQLite	PostgreSQL	MySQL	SQL Server
Tamaño	✅ Muy pequeño	❌ Grande	❌ Grande	❌ Grande
Costo	✅ Gratis	✅ Gratis	✅ Gratis	❌ Licencia

Ventajas de SQLite

- **Sin Configuración**: No requiere instalación ni configuración de servidor
- **Portabilidad Total**: Base de datos en un solo archivo
- **Rendimiento**: Excelente para aplicaciones de escritorio
- **Integridad**: Transacciones ACID completas
- **Simplicidad**: Ideal para proyectos académicos
- **Respaldos**: Fácil copiar/restaurar (un solo archivo)

Desventajas de SQLite

- **Concurrencia Limitada**: No ideal para múltiples usuarios simultáneos
- **Funciones Avanzadas**: Menos características que PostgreSQL
- **Escalabilidad**: Limitada para grandes volúmenes de datos
- **Administración**: Herramientas de administración limitadas

Justificación De La Selección

Para Las Necesidades Del Proyecto:

- **Aplicación de Escritorio**: No requiere servidor centralizado
- **Datos Locales**: Experiencias, comunidades y reservas almacenadas localmente
- **Simplicidad**: Prototipo funcional sin complejidad de servidor
- **Integridad**: Soporte completo para relaciones y constraints

Para las Capacidades del Equipo:

- **Facilidad de Uso**: No requiere conocimientos de administración de BD
- **Debugging**: Fácil visualización y modificación de datos
- **Aprendizaje**: Enfoque en SQL sin complejidades de configuración

Para los Objetivos del Curso:

- **Arquitectura Monolítica**: BD embebida perfecta para monolito
- **Desarrollo Completo**: Permite demostrar normalización y diseño
- **Pruebas**: Fácil crear bases de datos de prueba
- **Documentación**: Esquemas claros y documentables

3. Bibliotecas Y Herramientas Complementarias

Frontend (Renderer Process)

React.js

Propósito: Biblioteca para construcción de interfaces de usuario

Justificación:

- Componentes reutilizables para formularios y cards
- Manejo de estado eficiente
- Ecosistema maduro con hooks y context
- Integración natural con Electron

Tailwind CSS

Propósito: Framework CSS utility-first para diseño

Justificación:

- Desarrollo rápido de interfaces atractivas
- Tema personalizable para identidad colombiana
- Componentes responsivos sin CSS personalizado
- Menor tamaño final que Bootstrap

React Router

Propósito: Navegación entre páginas en SPA

Justificación:

- Manejo de rutas para diferentes tipos de usuario
- Protección de rutas autenticadas
- Navegación fluida sin recargas

Leaflet + React-Leaflet

Propósito: Mapas interactivos con OpenStreetMap

Justificación:

- Mapas gratuitos sin API keys

- Marcadores personalizables para experiencias
- Integración fácil con React
- Funcionalidad offline potencial

Backend (Main Process)

Knex.js

Propósito: Query builder para SQLite

Justificación:

- Abstracción sobre SQL crudo
- Migraciones y schemas versionados
- Prevención de inyección SQL
- Soporte completo para SQLite

Bcrypt

Propósito: Encriptación de contraseñas

Justificación:

- Estándar de la industria para hashing
- Resistente a ataques de fuerza bruta
- Fácil integración con Node.js
- Seguridad probada

Desarrollo y Pruebas

Jest
Propósito: Framework de testing

Justificación:

- Pruebas unitarias e integración
- Mocking avanzado para servicios
- Cobertura de código integrada
- Sintaxis clara y documentada

React Testing Library

Propósito: Testing de componentes React

Justificación:

- Pruebas enfocadas en comportamiento de usuario
- Integración perfecta con Jest
- Mejores prácticas de testing
- Simulación de interacciones

Babel

Propósito: Transpilador JavaScript

Justificación:

- Soporte para JavaScript moderno
- Compatibilidad con versiones anteriores
- Transpilación de JSX
- Optimizaciones automáticas

Empaquetado y Distribución
Electron Builder

Propósito: Empaquetado de aplicaciones Electron

Justificación:

- Instaladores nativos para cada plataforma
- Configuración declarativa
- Auto-updater integrado
- Optimización de tamaño