Componente	Tecnología Sugerida	Justificación
Front-End	**React.js** + TailwindCSS	React es eficiente para renderizado dinámico de componentes, y Tailwind permite estilos rápidos y personalizados.
Back-End	**Node.js** + Express.js	Rápido, ligero y con buena integración para APIs REST. Ideal para manejar peticiones al API del clima.
Base de Datos (opcional)	**MongoDB** o **LocalStorage**	Si solo se necesita historial o favoritos simples, LocalStorage es suficiente. Para algo más robusto, MongoDB.
API del clima	**OpenWeatherMap API**	Gratuita, confiable, bien documentada. Permite obtener clima actual, íconos, humedad y descripción.

#	Tarea	Descripción	Prioridad	Responsable	Estado
1	Inicialización del repositorio	Crear repositorio en GitHub, definir ramas de desarrollo.	Alta	Dev Líder	✅ Completo
2	Configuración de entorno	Instalar Node.js, crear estructura de carpetas, configurar Vite o Create	Alta	Dev Front	✅ Completo
3	Diseño de interfaz	Crear el mockup de la interfaz (input, botón de búsqueda, contenedor del	Media	UX/UI	✅ Completo
4	Desarrollo del Front-End básico	Input de ciudad, botón buscar, contenedor para resultados.	Alta	Dev Front	✅ Completo
5	Conexión a OpenWeather API	Obtener la API key, probar en Postman, integrar fetch o axios en el front.	Alta	Dev Back	✅ Completo
6	Renderizar datos del clima	Mostrar ícono, temperatura, humedad y descripción en pantalla.	Alta	Dev Front	✅ Completo
7	Manejo de errores	Mostrar mensajes si la ciudad no existe o falla la API.	Media	Dev Front	✅ Completo
8	Diseño responsivo	Adaptar el diseño para móviles y escritorio.	Baja	Dev Front	✅ Completo
9	Persistencia local (opcional)	Guardar ciudades buscadas recientemente usando LocalStorage.	Baja	Dev Back	✅ Completo
10	Pruebas funcionales	Probar distintas ciudades, errores comunes, conectividad.	Alta	QA	✅ Completo
11	Documentación del proyecto	README con instalación, uso y tecnologías usadas.	Media	Dev Líder	✅ Completo
12	Despliegue en línea	Deploy en Vercel, Netlify o GitHub Pages.	Alta	Dev Front	✅ Completo