PROYECTO DESARROLLO PORTALVYP.COM

Introducción

Este documento detalla una serie de librerías y herramientas recomendadas para optimizar y mejorar el proyecto. La integración de estos paquetes adicionales tiene como objetivo mejorar significativamente aspectos clave como la seguridad, el rendimiento y la experiencia del usuario. Si bien tecnologías fundamentales como **React.js**, **React Native**, **Next.js**, **NestJS**, **TypeORM**, **TailwindCSS** y **TypeScript** forman la base del proyecto, las librerías sugeridas complementarán y potenciarán el funcionamiento general de la aplicación. Además, se entrega el **diseño**, la **lógica** y la **base de datos** necesarias para la construcción de la plataforma, proporcionando una estructura sólida y coherente para su desarrollo. A continuación, se especifican los **8 puntos más destacados** en los que se debe trabajar, con un enfoque secuencial que asegura el cumplimiento de los objetivos en un plazo de **6 semanas**.

---

1. **Infraestructura y Seguridad** (Backend, caché, CDN, Servidor, seguridad):

Librerías y Tecnologías recomendadas:

1. **Frontend**: **React** y **Next.js** ofrecen una base sólida para el desarrollo de aplicaciones web modernas y escalables. **Next.js** proporciona características como renderizado del lado del servidor (SSR) y la generación de sitios estáticos (SSG), optimizando tanto la velocidad como el SEO. **TailwindCSS**: Un framework de CSS altamente eficiente que permite un diseño modular y totalmente personalizable, lo que facilita la creación de interfaces responsivas y atractivas.
2. **Backend**: **TypeScript** y **NestJS**: Esta combinación garantiza un backend estructurado y tipado, lo que mejora la seguridad del código, reduce los errores y aumenta la escalabilidad del sistema. **NestJS** estábasado en Express.js pero proporciona una arquitectura más robusta y modular.
3. **Base de Datos: SQL/PostgreSQL**: Una base de datos relacional ideal para manejar relaciones complejas y consultas eficientes. PostgreSQL ofrece robustez y flexibilidad, con soporte avanzado para tipos de datos y extensiones. Se debe realizar una migración desde MySQL a PostgreSQL para aprovechar sus ventajas. La implementación en Clever-Cloud facilita la gestión y escalabilidad en la nube.
4. **Repositorio GitHub**: Almacenar y versionar el código con **GitHub** es esencial para la colaboración en equipo y el control de versiones. Además, GitHub Actions permite la integración continua y el despliegue automático.
5. **Hosting del Frontend**: **Netlify**: Proporciona un hosting fácil de usar y de alta calidad para aplicaciones frontend. Permite despliegues rápidos, integraciones de CI/CD y soporte para funciones serverless.
6. **Hosting del Backend**: **Vercel**: Ideal para aplicaciones basadas en Next.js. Vercel ofrece un despliegue optimizado, escalabilidad automática y una excelente integración con el frontend. La integración de **Swagger** en Vercel proporciona documentación interactiva de la API, lo que facilita la comprensión y uso de los endpoints por parte de los desarrolladores y usuarios.
7. **Gestión de Archivos**: **Cloudinary**: Solución ideal para almacenar, procesar y servir imágenes y documentos, con características avanzadas como optimización de imágenes, soporte de transformaciones y una API flexible.
8. **Redis, Memcached, CDN, etc.:** Usados como sistemas de caché en el backend con NestJS. Redis es altamente eficiente para almacenar datos de acceso frecuente, lo que optimiza considerablemente los tiempos de respuesta del servidor y aligera la carga sobre la base de datos. Es crucial configurarlo adecuadamente en el backend para almacenar datos temporales de alta demanda, como los resultados de búsquedas o sesiones de usuario.

Resumen:

Para mejorar el rendimiento de la aplicación, se recomienda usar Redis para optimizar el backend y reducir la carga de consultas repetidas, almacenando cachés de consultas pesadas y datos temporales. Desplegar en plataformas como Vercel o Netlify permitirá integración continua y escalabilidad automática, mejorando la disponibilidad y el rendimiento del frontend. Cloudinary es ideal para optimizar y almacenar medios, asegurando tiempos de carga rápidos sin perder calidad de imagen. En Next.js, se deben usar técnicas como dynamic import() para cargar componentes de manera diferida y pre-fetching de rutas para mejorar la navegación. La purga de CSS no utilizado en TailwindCSS y la optimización de la compilación con herramientas como PostCSS y CSSnano contribuirán a reducir el tamaño de los archivos y mejorar los tiempos de carga. Para tareas que pueden ejecutarse en segundo plano, como el procesamiento de imágenes o correos electrónicos, se recomienda usar Bull o Agenda. En NestJS, la validación de datos con class-validator y class-transformer garantiza que los datos entrantes cumplan con los requisitos antes de procesarlos. El uso de JWT es ideal para la autenticación y la gestión de roles. Utilizar un CDN, como el de Cloudinary, mejorará la latencia al servir archivos desde servidores cercanos al usuario. Además, herramientas de monitoreo como New Relic o Datadog ayudan a identificar cuellos de botella y optimizar el rendimiento en tiempo real. Finalmente, habilitar HTTP/2 y GZIP en el servidor mejorará la velocidad de carga y la eficiencia en la transmisión de datos.

---

2. **Autenticación y Panel de Control** (Gestión de usuarios, roles, panel de administración):

Librerías y Tecnologías recomendadas:

1. **Supabase**: Proporciona un conjunto completo de herramientas para gestionar la autenticación y seguridad, incluyendo:
    - OAuth2 con proveedores como Google, Facebook y Apple.
    - Hash de contraseñas de manera segura utilizando algoritmos adecuados como bcrypt o Argon2.
    - Autenticación basada en JWT (tokens) con soporte de renovación de tokens.
    - Control de acceso basado en roles (RBAC) para gestionar permisos de usuario.
2. **express-rate-limit**: Limita el número de intentos de inicio de sesión por dirección IP para prevenir ataques de fuerza bruta, asegurando que solo un número determinado de intentos sean permitidos en un tiempo específico.
3. **Helmet.js**: Asegura la aplicación mediante la configuración de cabeceras HTTP de seguridad, protegiendo contra ataques comunes como XSS (Cross-Site Scripting), Clickjacking y CSRF (Cross-Site Request Forgery).
4. **React Table**: Librería para la gestión y visualización de datos tabulares en el panel administrativo. Permite crear tablas personalizadas y dinámicas, con características como filtrado, ordenación, paginación y agrupación de datos, lo cual es ideal para mostrar grandes volúmenes de información de manera eficiente.
5. **Recharts o Chart.js**: Ambas son librerías populares para mostrar gráficos y estadísticas en el panel administrativo.
    - **Recharts** es fácil de usar y se integra bien con React.
    - **Chart.js** es más flexible y tiene una mayor variedad de gráficos, permitiendo visualizar datos como el rendimiento de ventas, tráfico de usuarios o estadísticas generales.
6. **NestJS RBAC**: Librería de Role-Based Access Control (RBAC) para gestionar los roles y permisos de los usuarios en el backend. Con NestJS RBAC, puedes establecer diferentes niveles de acceso para distintos tipos de usuarios (administradores, usuarios estándar, etc.) y proteger las rutas y recursos según los roles.

Resumen:

El sistema de roles y paneles personalizados del portal inmobiliario está diseñado para gestionar diferentes niveles de acceso y funciones según el tipo de usuario. El **Administrador** tiene acceso completo a todas las funciones, incluyendo la gestión de usuarios, propiedades, pagos, reportes, configuraciones globales, seguridad y un sistema de favoritos para controlar las interacciones de los usuarios. El **Agente** tiene permisos para publicar, modificar y destacar propiedades según el plan contratado, gestionar estadísticas personalizadas, comunicarse con los usuarios, recibir notificaciones y gestionar sus propios favoritos. Los **Usuarios** pueden publicar propiedades (si

tienen habilitado el permiso), gestionar su perfil, interactuar con propiedades guardadas como favoritas y recibir alertas sobre actualizaciones, además de comunicarse con agentes o propietarios a través de los canales disponibles.

Se recomienda considerar la **Autenticación Multifactor (MFA)** como una mejora futura, especialmente para roles con acceso a funciones sensibles como administradores o agentes, agregando una capa extra de seguridad.

En este contexto, el **panel administrativo** (el área privada) se gestiona mediante librerías como **React Table** para mostrar datos tabulares, **Recharts** o **Chart.js** para representar visualmente estadísticas y métricas, y **NestJS RBAC** para controlar los roles y permisos de acceso. Además, se sugiere dividir el panel en módulos, implementar **lazy loading** para mejorar el rendimiento, y gestionar la seguridad de roles de manera adecuada en el backend. También se recomienda añadir funcionalidades como **filtrado, búsqueda avanzada y notificaciones** dentro del panel, lo cual contribuye a una administración eficiente y optimizada del área privada de los usuarios.

---

3. Diseño Inicial (UX/UI):

Librerías y Tecnologías recomendadas:

1. **Framer Motion o React Native Reanimated 2**: Esta librería se utiliza para crear animaciones fluidas y atractivas en la interfaz de usuario. Permite agregar transiciones suaves entre elementos, así como animaciones interactivas que mejoran la percepción del usuario. Framer Motion es para React (web), y React Native Reanimated 2 es para React Native (app móvil). Si planeas implementar animaciones en ambas plataformas, sería adecuado mantener ambas.
2. **NextUI** (para la **web**) y **React Native Gesture Handler** (para la **app móvil**): Librería para crear interfaces de usuario modernas y elegantes en React.. NextUI es excelente para interfaces web modernas y limpias, y React Native Gesture Handler es la librería recomendada para gestionar gestos en dispositivos móviles. No se sustituyen, ya que cada una está diseñada para una plataforma específica.
3. **React Tour**: Esta librería permite guiar a los usuarios nuevos a través de recorridos interactivos. Es útil para la introducción de nuevas funcionalidades o para educar a los usuarios sobre cómo utilizar ciertas partes de la aplicación.
4. **react-scroll**: Librería para crear transiciones suaves cuando los usuarios navegan entre secciones de una página. Aporta una experiencia de navegación más fluida y agradable.
5. **react-tooltip**: Librería para mostrar sugerencias, tips y pequeños detalles en forma de pop-ups cuando el usuario interactúa con elementos específicos. Es útil para mejorar la accesibilidad y proporcionar explicaciones adicionales sin sobrecargar la interfaz.
6. **i18next**: Herramientas para la internacionalización (i18n) en el frontend. Ambas librerías permiten que tu aplicación soporte múltiples idiomas, adaptando contenido, fechas, monedas y formatos según el idioma elegido. **i18next** es más flexible y tiene soporte para traducciones en tiempo real, mientras que **react-intl** se integra más fácilmente con aplicaciones React.
7. **@nestjs/i18n**: Librería para internacionalización en el backend, específicamente para aplicaciones construidas con NestJS. Permite gestionar las traducciones y configuraciones regionales de manera eficiente, asegurando que el backend pueda devolver contenido localizado según el idioma del usuario.
8. **react-aria**: Una librería de accesibilidad para mejorar la experiencia de usuarios con discapacidades. Proporciona componentes accesibles y adaptativos que cumplen con los estándares WCAG (Web Content Accessibility Guidelines). Esto incluye características como navegación por teclado, descripciones de elementos para lectores de pantalla, entre otros.

Resumen:

Realiza pruebas de rendimiento regulares en las páginas que usan animaciones y efectos visuales para garantizar que no afecten la experiencia del usuario. Asegúrate de que las animaciones sean accesibles, permitiendo pausarlas o desactivarlas para usuarios con sensibilidades. Implementa carga bajo demanda (lazy loading) para librerías

como Framer Motion y React Tour, cargándolas solo cuando el usuario interactúe con áreas relevantes. Utiliza animaciones con moderación para evitar distracciones y ralentización, asegurándote de que sean fluidas sin afectar el rendimiento. Las interacciones, como tooltips o recorridos de React Tour, deben ser no invasivas, permitiendo al usuario decidir cuándo iniciarlas o finalizarlas. En dispositivos móviles, reduce las animaciones complejas para mejorar la experiencia debido a las limitaciones de rendimiento. Utiliza **i18next** o **react-intl** para gestionar la internacionalización en el frontend, permitiendo soporte multilingüe con adaptaciones para contenido, fechas y monedas. Para el backend, **@nestjs/i18n** permite manejar traducciones y configuraciones regionales. Para mejorar la accesibilidad, usa **react-aria**, que asegura componentes accesibles conforme a los estándares WCAG, y **axe-core** para realizar auditorías automáticas de accesibilidad en el frontend. Además, considera usar **gettext** o **Poedit** para gestionar traducciones de manera eficiente y facilita el cambio de idiomas en el frontend para una mejor experiencia de usuario.

---

4 **Búsqueda y Anuncios** (Optimización de búsquedas, mapas**, p**ublicación, Formularios, validación):

Librerías y Tecnologías recomendadas:

1. **ElasticSearch**: Para implementar una búsqueda optimizada con tiempos de respuesta rápidos, especialmente útil para manejar grandes volúmenes de datos y proporcionar resultados de búsqueda instantáneos.
2. **React Query**: Para manejar el estado de los datos y la paginación de las búsquedas de manera eficiente, asegurando que se carguen solo los datos necesarios en cada momento.
3. **Infinite Scroll**: Similar a las redes sociales, donde los datos se cargan progresivamente a medida que el usuario desliza hacia abajo, sin necesidad de cambiar de página. Esto ofrece una experiencia más fluida y cómoda, ya que el usuario solo tiene que seguir desplazándose sin preocuparse por la paginación.
4. **Leaflet.js**: Para integrar mapas interactivos utilizando OpenStreetMap. Es una librería ligera y altamente personalizable.
5. **react-leaflet-draw**: Para permitir a los usuarios dibujar áreas personalizadas en el mapa, como zonas de interés, filtros geográficos, etc.
6. **React Hook Form**: Para manejar los formularios de filtros de búsqueda de manera eficiente, reduciendo el re-renderizado innecesario de los componentes.
7. **next-seo**: Para optimización SEO avanzada, como la gestión de metadatos dinámicos y el contenido enriquecido en el footer.
8. **Formik o React Hook Form**: Estas librerías están diseñadas para manejar formularios, lo cual es esencial para la creación y publicación de anuncios de propiedades. Te permiten gestionar de manera eficiente el estado de los formularios, lo que puede mejorar la experiencia de los usuarios al publicar anuncios.
9. **Zod**: Para la validación declarativa de los formularios, asegurando que los datos de entrada sean correctos antes de enviarlos.
10. **Redis o ElasticSearch**: Para automatizar las recomendaciones de propiedades basadas en el comportamiento del usuario o análisis de datos históricos. **Redis** se puede utilizar como un sistema de cache para mejorar la velocidad de las recomendaciones, mientras que **ElasticSearch** es ideal para búsquedas complejas en grandes volúmenes de datos.
11. **NextUI Table**: Para crear tablas de gestión de propiedades, con funcionalidades avanzadas como paginación, filtrado, ordenamiento y edición en tiempo real.
12. **Supabase con Row-Level Security (RLS)**: Para gestionar roles y permisos en el backend, asegurando que solo los usuarios autorizados puedan acceder a ciertos recursos o realizar determinadas acciones.

Resumen:

La plataforma incluirá un mapa interactivo basado en OpenStreetMap, con precios actualizados en tiempo real y opciones de búsqueda personalizadas. Habrá varios tipos de buscadores, un módulo de ordenación y opciones avanzadas como búsqueda por mapa, voz y geolocalización. Además, se optimizará el SEO del footer y se

ofrecerán páginas personalizadas para agencias, herramientas de financiamiento y simuladores de hipotecas. Los anuncios incluirán enlaces a propiedades relacionadas y recomendaciones automatizadas mediante algoritmos.

Se gestionarán formularios de publicación con herramientas como Formik o React Hook Form, validación con Zod y recomendaciones mediante Redis o ElasticSearch. La plataforma usará NextUI Table para gestionar propiedades con paginación y filtrado en tiempo real. Supabase con Row-Level Security controlará roles y permisos en el backend.

ElasticSearch es recomendado para búsquedas complejas en grandes volúmenes de datos, mientras que Redis es útil para almacenar recomendaciones populares. Se implementará paginación en el servidor para mejorar el rendimiento y asegurar que solo usuarios autorizados puedan gestionar propiedades y administrar la plataforma.

---

5. Monetización y Pagos (Interfaz, conexión, pantallas de Pago):

Librerías y Tecnologías recomendadas:

1) **Stripe SDK**: Para integrar pagos y suscripciones. Stripe ofrece una solución completa para procesar pagos en línea, incluyendo soporte para tarjetas de crédito, débito, y más. Además, permite gestionar suscripciones recurrentes, una funcionalidad clave para los modelos de negocio basados en membresías o pagos periódicos.
2) **Stripe Webhooks**: Para gestionar notificaciones de pagos. Stripe envía notificaciones a través de Webhooks cada vez que se realiza un pago o cuando hay cambios en el estado de una suscripción. Es esencial para mantener la coherencia de los datos en tiempo real y realizar acciones automáticas, como actualizar el estado del usuario o enviar confirmaciones de pago.
3) **React-Stripe-JS** : Integra las facturas pagadas, pendientes y suscripciones en el área privada del perfil del usuario. Estas librerías facilitan la visualización de pagos y suscripciones de Stripe en la interfaz de React, utilizando Stripe SDK o API para obtener y mostrar la información de manera fluida y en tiempo real.
4) **Supabase**: Para almacenar el historial de facturación de cada usuario y asegurar que la información se muestre correctamente en su perfil privado.
5) **Google AdSense**: Para gestionar anuncios publicitarios. Esta API permite crear, gestionar y optimizar campañas publicitarias a través de Google Ads, ideal para monetizar la plataforma mediante publicidad dirigida.
6) **Firmafy**: Para la firma digital de documentos de transacciones. Firmafy ofrece una solución sencilla para firmar digitalmente documentos, permitiendo a los usuarios completar transacciones sin necesidad deencuentros físicos.
7) **Math.js**: Para cálculos complejos en el simulador hipotecario. Math.js es una librería matemática poderosa que permite realizar operaciones y cálculos de alto rendimiento. Es ideal para crear simuladores hipotecarios precisos, que pueden incluir operaciones como tasas de interés, plazos de pago, y amortización de préstamos.
8) **Chart.js** o **Recharts**: Para visualizar gráficos de simulaciones y estadísticas. Ambas librerías permiten generar gráficos interactivos y dinámicos para mostrar datos como la evolución de una hipoteca, la comparación entre diferentes tasas de interés, o la distribución de pagos a lo largo del tiempo.

Resumen:

Se integrará Stripe para pagos seguros, habilitando pagos recurrentes y suscripciones tanto para particulares como profesionales. El modelo de monetización abarcará tanto la web como la app, con opciones de resaltado y promoción de anuncios mediante pagos para colocar anuncios en posiciones destacadas o con etiquetas como "Urgente" o "Oferta Especial". Los profesionales podrán suscribirse a planes mensuales, trimestrales o anuales para mayor visibilidad. También se ofrecerán servicios adicionales de pago, como informes de propiedades y análisis de

mercado, y un modelo freemium, donde las funcionalidades básicas son gratuitas y las avanzadas están disponibles mediante suscripción.

Se incluirán paquetes de promoción para resaltar anuncios y acciones de marketing, así como comisiones por transacciones o acuerdos cerrados a través de la plataforma. Además, los usuarios de la app podrán acceder a funciones exclusivas mediante suscripción, y se podrán mostrar anuncios publicitarios en la aplicación para generar ingresos sin afectar la experiencia del usuario. Si estás manejando pagos recurrentes o tareas a largo plazo, implementa **Bull** con una estrategia de reintentos para asegurarte de que los pagos fallidos se intenten nuevamente sin que el sistema quede bloqueado.

Para el simulador hipotecario, se recomienda utilizar Math.js, una librería matemática potente para realizar cálculos complejos como tasas de interés, plazos de pago y amortización de préstamos. Para la visualización de los datos, Chart.js o Recharts son opciones ideales, ya que permiten generar gráficos interactivos que muestran la evolución de la hipoteca, la comparación entre diferentes tasas de interés y la distribución de pagos. Se sugiere implementar actualizaciones en tiempo real en los gráficos, de modo que al cambiar una variable como la tasa de interés o el plazo de la hipoteca, los gráficos se actualicen automáticamente sin necesidad de recargar la página. Además, el uso de Web Workers para ejecutar los cálculos en un hilo separado puede mejorar el rendimiento, evitando que la interfaz de usuario se congele durante los procesos de cálculo. Es importante optimizar los gráficos para que sean interactivos, lo que permitirá a los usuarios explorar diferentes escenarios y visualizar cómo las variables afectan los resultados en tiempo real.

---

6. **Interacciones y Comunicación** (Chat, mail, WhatsApp, notificaciones).

Librerías y Tecnologías recomendadas:

1. **Socket.io:** Para implementar un chat en tiempo real utilizando WebSockets. Esta librería permite una comunicación bidireccional en tiempo real entre el servidor y el cliente, ideal para aplicaciones de mensajería instantánea.
2. **Clientify:** Para el envío de correos electrónicos a través de SMTP. Permite configurar y enviar correos electrónicos con facilidad, utilizando servicios como Gmail, SendGrid o servidores SMTP personalizados.
3. **WonlineSMS y WhatsBoot:** Para el envío de alertas y mensajes por SMS o WhatsApp. Ambas API permiten integrar fácilmente la funcionalidad de mensajería en la plataforma.
4. **Push.js:** Para implementar notificaciones push en la web. Esta librería permite enviar alertas a los usuarios incluso cuando no están activamente en la aplicación.
5. **react-datepicker o NextUI con DatePicker:** Para implementar un calendario de visitas o cualquier otro tipo de selección de fechas en la interfaz. Es muy útil para gestionar la disponibilidad de propiedades o agendar citas.
6. **NestJS EventEmitter:** Para manejar eventos del sistema, como alertas o acciones desencadenadas por ciertos eventos (ej. nuevos mensajes, actualizaciones de propiedades, etc.). Permite una gestión eficiente de eventos en el backend.
7. **hbs (Handlebars):** Para crear plantillas de correo electrónico personalizadas. Permite generar contenido dinámico y estructurado en los correos electrónicos, haciendo que las notificaciones sean más atractivas y funcionales. Usa variables dinámicas en las plantillas para proporcionar información personalizada y relevante para cada usuario (ej. detalles de la propiedad, horarios de visitas, etc.).

Resumen:

Los usuarios podrán marcar propiedades como favoritas y recibir notificaciones sobre cambios en su precio, disponibilidad o cualquier otra modificación relevante. También podrán configurar alertas personalizadas según criterios como rango de precios, ubicación y tipo de propiedad, recibiendo notificaciones por correo electrónico, WhatsApp, notificaciones push en la app o SMS. El sistema de contacto mediante SMTP permitirá que los usuarios se comuniquen directamente con agentes o propietarios desde los formularios de contacto en la web o la app. Este

sistema también gestionará correos electrónicos relacionados con los anuncios, detalles de propiedad y agencias. Se implementarán plantillas personalizadas para correos electrónicos y WhatsApp, mejorando la experiencia del usuario. Además, el sistema de captación de contactos guardará la información de los compradores interesados para enviarles automáticamente actualizaciones sobre nuevas propiedades que coincidan con sus preferencias.

Se habilitará un chat interno mediante WebSockets, permitiendo la comunicación en tiempo real entre usuarios y agentes inmobiliarios, tanto en la web como en la app, con la opción de guardar las conversaciones para futuras referencias. Se implementará un sistema de colas para el envío de correos masivos y notificaciones SMS/WhatsApp, para evitar la sobrecarga del servidor y garantizar que los mensajes se envíen de manera eficiente.

Si el sistema de chat en tiempo real con Socket.io va a manejar grandes volúmenes de mensajes, se evaluará la posibilidad de distribuir las conexiones a través de múltiples instancias de servidor utilizando Redis o alguna solución similar para manejar la escalabilidad. Para **Push.js**, asegúrate de que las notificaciones sean relevantes y no abusivas, ya que las notificaciones excesivas pueden generar una mala experiencia para el usuario.

Resumen:
El sistema propuesto para un portal inmobiliario incluye una amplia gama de tecnologías de inteligencia artificial y machine learning para mejorar tanto la experiencia del usuario como la eficiencia del proceso inmobiliario. Se integra un asistente virtual con Rasa para responder consultas sobre propiedades, un motor de búsqueda avanzado con ElasticSearch y spaCy para búsquedas semánticas, y un modelo de predicción de valor de mercado utilizando TensorFlow.js y scikit-learn. También se implementan descripciones automáticas de propiedades con Hugging Face, un sistema de recomendaciones personalizadas basado en el comportamiento de los usuarios, y recorridos virtuales interactivos con Three.js y A-Frame. Además, se utiliza Prophet y big data para el análisis predictivo de tendencias del mercado y la personalización del feed de propiedades. Otras funciones incluyen simulación de mejoras para propiedades con ARKit, un sistema de subastas inmobiliarias online con blockchain, un sistema colaborativo de valoración de propiedades con machine learning, y un marketplace para conectar usuarios con profesionales inmobiliarios.

---

7. **Aplicaciones Móviles y Notificaciones** (React Native, FCM):

Librerías y Tecnologías recomendadas:

1) **Firebase Cloud Messaging (FCM)** : para notificaciones push en tiempo real en aplicaciones móviles y web. Firebase es ideal para enviar notificaciones push masivas de manera eficiente. FCM tiene un plan gratuito con envío ilimitado de notificaciones push, lo que lo convierte en una opción viable para gestionar la mensajería y notificaciones en tiempo real de forma escalable.
2) **Firebase Cloud Messaging (FCM):** FCM es ideal para enviar notificaciones push tanto a aplicaciones móviles (iOS y Android) como a aplicaciones web. Puedes usarlo para enviar notificaciones de manera uniforme a ambas plataformas desde un único servicio.
3) **Socket.IO (para chats en tiempo real y envío de audios)** : Para implementar comunicación en tiempo real, como chats, y permitir el envío de audios o mensajes multimedia. Socket.IO es perfecto para gestionar mensajes en tiempo real sin sobrecargar el servidor, manteniendo conexiones persistentes de baja latencia entre el cliente y el servidor.
4) **Axios** : Para realizar solicitudes HTTP entre la aplicación y el backend. Axios se usa para gestionar las peticiones de datos, incluyendo la integración con las notificaciones push, y proporciona un manejo de errores robusto y tiempos de espera configurables.
5) **Supabase** : Como base de datos, autenticación y servicio de tiempo real. Supabase ofrece integración sencilla con PostgreSQL, websockets para actualizaciones en tiempo real y es ideal para manejar la base de datos relacional y almacenar los datos de usuarios, chats y notificaciones. Además, se puede integrar con FCM para gestionar las notificaciones push.

6) **React Navigation** : Para la navegación dentro de la aplicación móvil. Gestiona eficientemente el flujo de pantallas y rutas en aplicaciones móviles, con soporte para navegación por pilas, cajones, pestañas y más.

Resumen:
La estrategia planteada consiste en aprovechar el código de la web y conectar tanto la aplicación móvil como la web mediante una API RESTful. La web estará desarrollada usando Next.js para el frontend y NestJS para el backend, lo que permitirá centralizar la lógica de negocio y reutilizar gran parte del código. Este enfoque optimiza el desarrollo y asegura coherencia entre ambas plataformas, además de mejorar el rendimiento. En cuanto a la aplicación móvil, se utilizará React Native, lo que garantizará una experiencia nativa para el usuario mientras se conecta a la misma API RESTful que la web. Esto no solo facilita el mantenimiento a largo plazo, sino que también asegura una experiencia fluida y optimizada, comparable a aplicaciones populares como Idealista o Fotocasa. Además, se recomienda el uso de herramientas como Firebase Cloud Messaging para las notificaciones push, Socket.IO para la comunicación en tiempo real, Axios para las solicitudes HTTP, y Supabase como base de datos y servicio en tiempo real, lo que contribuirá a un rendimiento eficaz y a una experiencia de usuario coherente entre la web y la aplicación móvil.

---

8. **SEO y Estrategias de Indexación** (next-seo, next-sitemap, Schema.org):

Librerías y Tecnologías recomendadas:

1. **next-seo**: Esta librería te permite manejar los metadatos SEO dinámicamente en las páginas de tu aplicación Next.js, como el título, la descripción, palabras clave y otros datos importantes para el SEO. Puedes generar metadatos personalizados para cada página para mejorar su visibilidad en motores de búsqueda.
2. **next-sitemap**: Librería para generar automáticamente sitemaps XML para tu aplicación, lo que facilita la indexación de tus páginas en los motores de búsqueda. Los sitemaps ayudan a los motores de búsqueda a encontrar todas las páginas importantes de tu sitio y mejorar la indexación.
3. **Schema.org o Microdata**: Implementa datos estructurados en tu sitio usando el vocabulario de Schema.org **o** Microdata para mejorar la comprensión de tu contenido por parte de los motores de búsqueda y aumentar la visibilidad en los resultados de búsqueda enriquecidos. Puedes usarlo para marcar propiedades, eventos, ofertas, valoraciones y otros tipos de contenido relevante.
4. **Helmet.js**: Librería que permite la gestión de los encabezados HTTP y la seguridad en la aplicación, incluyendo la configuración de cabeceras como X-Content-Type-Options, Content-Security-Policy, Strict-Transport-Security y otros encabezados críticos para la seguridad y la optimización SEO.
5. **Lighthouse**: Herramienta de Google para medir y mejorar el rendimiento del frontend. Lighthouse ofrece auditorías sobre accesibilidad, SEO, rendimiento y mejores prácticas, lo que te permite identificar áreas de mejora en tu aplicación web. Usa Lighthouse de manera regular durante el desarrollo para asegurar que la aplicación cumpla con los estándares de rendimiento.

Resumen:

Asegúrate de ajustar los metadatos SEO generados por *next-seo* al contenido específico de cada página, utilizando técnicas como la carga diferida para evitar sobrecargar el renderizado de páginas dinámicas. Configura *next-sitemap* para indexar las URLs relevantes y excluir páginas de baja relevancia. Implementa datos estructurados en páginas clave (propiedades, artículos) y mantenlos actualizados para mejorar la visibilidad en motores de búsqueda. Utiliza *Helmet.js* para gestionar estrategias de caché, evitando que contenido obsoleto sea indexado. Aplica lazy loading para recursos no esenciales, como imágenes, para mejorar la carga de la página sin afectar el SEO. Optimiza la aplicación para dispositivos móviles, ya que Google prioriza estos sitios en la indexación, y realiza auditorías SEO periódicas con herramientas como Google Search Console para asegurar una correcta indexación y rendimiento. Es importante utilizar Lighthouse periódicamente para medir el rendimiento y realizar las optimizaciones necesarias, lo que también mejora el SEO y la experiencia en dispositivos móviles.