**IMPLEMENTACIÓN DE CHATS Y COMUNICACIÓN EN UNA APLICACIÓN**

Firebase

Google poseía una base de datos en tiempo real que utiliza WebSockets y MongoDB como su pila de tecnología central. El servicio le permite guardar datos en una base de datos no-SQL y luego registrarse para recibir actualizaciones en tiempo real cuando los datos cambian. También ofrecen servicios para notificaciones push y almacenamiento de archivos. También hay un marco de mensajería de código abierto de terceros disponible para iOS y Android.

Firebase tiene una biblioteca llamada GeoFire que le permite conocer sus aplicaciones.

Pusher/PubNub

He agrupado estos juntos porque son bastante similares. Proporcionan infraestructura y tecnología de transmisión en tiempo real. Puede establecer canales entre grupos de dispositivos y transmitir datos entre ellos. No proporcionan ningún medio para autenticar a los usuarios, por lo que necesitaría un servidor de aplicaciones.

Quickblox

Quickblox está dirigido específicamente al mercado de mensajería instantánea y utilizan un servidor XMPP en el backend. Dado que utilizan XMPP, este servicio tiene muchas funciones de mensajería integradas, como listas de privacidad e indicadores de escritura. Proporcionan una aplicación de mensajería de ejemplo razonable pero básica.

Layer

La capa está a medio camino entre Quickblox y PubNub / Pusher. Ofrecen una plataforma de transmisión dirigida a la mensajería instantánea. Tienen algunas funciones de mensajería como el bloqueo básico, pero esto no es una implementación de mensajería a gran escala como Quickblox. Tienen una biblioteca de componentes de UI que se pueden usar para ayudar a construir una aplicación completa: Atlas.

Si usa un backend como servicio, siempre tendrá que crear el cliente de chat usted mismo. Si opta por una de las opciones de propósito más general, como Firebase, también tendrá que crear su propio protocolo de mensajería.

Estos servicios le ahorrarán tiempo cuando se trata de la implementación del servidor en tiempo real, la configuración del servidor y la escala.

XMPP

Otra opción es alojar un servidor XMPP usted mismo. Hay una serie de servidores XMPP de código abierto disponibles:

> ejabberd

> Prosody

> OpenFire

Los servidores XMPP le proporcionarán una gran cantidad de funciones de mensajería listas para usar. Por lo general, no incluyen ningún reconocimiento de ubicación, pero esto podría resolverse configurando el servidor para alojar la base de datos de usuarios en un servidor separado, lo que le permitiría agregar tablas adicionales para manejar la geolocalización. Una alternativa sería utilizar la presencia para actualizar dispositivos relevantes cuando la ubicación de un usuario cambie en cierta cantidad.

Para crear el cliente, recomendaría usar Smack. Como XMPP es un estándar abierto, su cliente debe poder trabajar con servidores y clientes de XMPP de terceros.

Si no desea crear el cliente usted mismo, hay una serie de proyectos de código abierto (en su mayoría, bajo licencias de copia libre) que podrían actuar como punto de partida. Xabber sería un ejemplo, pero puede encontrar más buscando. Alternativamente, podría usar un proyecto de código abierto comercial this.

Construyendo el servidor usted mismo

La última opción es construir el servidor de mensajería usted mismo. Esto tiene la ventaja de que puede agregar las funciones que necesita y mantenerlo razonablemente simple. Un buen enfoque sería utilizar Symfony y el paquete WebSocket para el back-end en tiempo real y Android WebSockets para el cliente. Usando este enfoque, tendría mucha flexibilidad al implementar la funcionalidad de ubicación geográfica.