# Enrique Hernandez

+535-488-4414
enrique@swagup.com

**LinkedIn:** https://www.linkedin.com/in/jos%C3%A9-enrique-hern%C3%A1ndez-mart%C3%ADnez-189830b0/
**GitHub:** https://github.com/MrWest

Crea un chat donde al menos dos personas puedan chatear. Crearás las interfaces y todo lo necesario alojado en el servicio que te parezca mejor.

https://boxitas-chat-pi.vercel.app/

Hola a todos. Para desarrollar este app de chat, utilice React para frontend que es mi area de experticia, cree ademas un simple rest api usando json-server asi que lo mantuve todo en NodeJs frontend y backend solo buscando rapidez de desarrollo.

Tras una investigacion de las arquitecturas  best practices y herramientas para este tipo de aplicaciones de chat me decidi por Pusher que es una plataforma lider en la industria y provee 3rd party services de lo que llaman canales que en efecto proveen comunicaion en tiempo real tanto con servidores  como aplicaciones y dispositivos. En terminos de arquitectura de puede alcanzar altos niveles de escalabilidad en un sistema basado en websockets, y la posibilidad de usar el API de un tercero agiliza el resultado considerablemente.

Para completar la tarea necesite una forma rapida, sencilla y segura de lograr la autenticacion.  Para ello use la API de facebook, lo cual brinda incluso comodidad para los usuarios de prueba.

En general mi version del chat solo tiene la intención de demostrar lo que soy capaz de hacer en un fin de semana. El chat tiene las principales características, y todo lo necesario para crecer, si sigue el debate y el desarrollo quiza se le pueda agregar algunas pruebas. Pero igual si mi intencion fuese hacer un chat de con otros fines mas serios mas alla de una prueba igual mi approach hubiese sido usando Pusher API.

Tanto en la app frontend como en la backend se puede ver en la practica lo coherente y flexible que resulta arquitectonicamente el uso de Pusher. A continuacion se inicializa la instancia de pusher se subcribe al channel “chat”, y se le indica que hacer con los eventos “messages” y con los datos recibidos atraves de estos eventos.

```
172   const pusher = new Pusher(process.env.REACT_APP_PUSHER_KEY, {
173       cluster: process.env.REACT_APP_PUSHER_CLUSTER,
174       forceTLS: true
175     });
176
186         {...}
187       // when the component is mounted creates the Pusher object and subscribe to the channels
188       // chat channel  binded  to 'message' to watch for new messages on the current chat
189       useEffect(() => {
190
191             pusher.unsubscribe('chat'); // clean up the previuos channel subsccription if any
192             const channel = pusher.subscribe('chat');
193             channel.bind('message', data => {
194               if(((selectedContact.id === data.sender) && (currentUser.id === data.receiver)) ||
195               ((currentUser.id === data.sender) && (selectedContact.id === data.receiver)) )
196               {
197                   registerMessage(data);
198                   isLoading(false);
199               }
200             });
```

Luego en backend tenemos una implementacion sencilla y escalable. De momento velamos por ciertos tipos de request al API en ciertos tipos de endpoints y en dependencia de esto decidimos que trigger levantar y que info pasar en la notificacion. En el ejemplo a continuacion se espera un post al endpoint messages para trigger un evento de tipo “messages”, pasandole el request body que seria el propio mensaje. (tambien a “notify”)

```
15  const pusher = new Pusher({
16    appId: process.env.REACT_APP_PUSHER_ID,
17    key: process.env.REACT_APP_PUSHER_KEY,
18    secret: process.env.REACT_APP_PUSHER_SECRET,
19    cluster: process.env.REACT_APP_PUSHER_CLUSTER
20  });
21  // Custom middleware to access POST methids.
22  // Can be customized for other HTTP method as well.
23  server.use((req, res, next) => {
24
25    if (req.method === "POST" && req.path.includes('/messages')) {
26      // If the method is a POST echo back the name from request body
27      const payload = req.body;
28      pusher.trigger('chat', 'message', payload);
29      pusher.trigger('chat', 'notify', payload);
30      // res.json(payload);
31      next();
32    }
```

Notese que en esta implementacion que simula una API que vendria siendo la capa de aplicacion o de negocio de la aplicacion, se puede manejar la info que viene en el request hacer todas las consultas pertinentes y repartir el resultado a los listeners del evento. Un ejemplo aun mas sencillo seria notificar se una palabra o frase indebida a la administracion o a un user que ha sido mencionado en una conversacion

De momento es un chat al que se entra logueando por facebook, los usuarios que se registran por primera vez se quedan en el sistema, todo el mundo puede chatear con el resto de los usuarios, se manejan los estados de los usuarios online-offline el estado de los mensajes (leido), su agrupacion por tiempo (intervalo de 1 minuto), notificaciones (solo visuales) de nuevos mensajes, y “is typing”. Todo responsive.

## **Acerca de Pusher** http://pusher.com

Pusher le brinda un punto final simple para publicar eventos en tiempo real. utilizando su SDK del lado del servidor y la biblioteca del lado del cliente, puede agregar eventos en tiempo real a través de websockets / flash / longpolling sin escribir un solo punto final de polling. todo lo que necesita hacer es preocuparse por configurar canales y tener las listas de eventos para esos canales en el lado del cliente y publicar en el lado del servidor de canales.

Anteriormente, sus planes ofrecían conexiones máximas por plan muy poco convincentes Pero una visita rápida a su página de precios reveló que han aumentado generosamente sus conexiones máximas. El plan de negocios es de 199 / mes y ofrece 5000 conexiones simultáneas. eso es un trato.

Solo ofreceré un concern: algunos users de pushers han comentado que llega a ser algo lento alrededor de 10k conexiones. Incluso si una aplicación se prevé en un futuro que sea aun mayor que ese numero de 10k connetions, igual recomendaria comenzar con pusher, ya que la arquitectura esta tan desacoplada que no resultaria demasiado traumatico cambiar a otra 3rd party API o a alguna propia.