Materia: Laboratorio de Sistemas Operativos y Redes

Informe final

Levantar aplicación de chat SpacebarChat

Profesor: José Luis Di Biase

Periodo: Primer cuatrimestre 2025

Alumnos: Deo Matias Francisco, Karottupullolil Ezequiel, Murua Ariel.

Introducción: ¿Qué es SpacebarChat?

Primero debemos hablar de **Discord**, que, por su parte, es una plataforma diseñada para intercambiar mensajes, comunicarse mediante voz, e incluso videollamadas; entre otras funciones, como crear canales para aceptar distintos miembros.

SpacebarChat es un proyecto de código abierto, actualmente **en desarrollo**, que busca replicar y ampliar la funcionalidad de la API cliente-servidor de **Discord** agregando la característica de ser completamente auto-hosteable, configurable, descentralizado, y extensible. Desarrollada enteramente utilizando **Typescript** como lenguaje de programación, compilado a **Javascript** e interpretado por **NodeJS**.

Descripción del proyecto

SpacebarChat está completamente desarrollado en un entorno **NodeJs** teniendo dos códigos fuentes separados para el frontend y para el backend:

- El frontend utiliza como stack tecnológico la librería web **ReactJs** con **VITE** para compilar archivos **TSX** a una **SPA (Single Page Application)**.
- El backend está estructurado completamente usando **ExpressJS** como **framework** web bajo la arquitectura de diseño **REST**ful API para la comunicación **Cliente-Servidor**, se puede encontrar una documentación de sus endpoints aca.

Prerrequisitos

La siguiente instalación de la aplicación es para entornos **Linux**. Como aclaración es necesario que el usuario tenga permisos de administración (es decir, sea **sudoer**) para toda la instalación.

Instalación de dependencias

Primero instalaremos las dependencias necesarias para configurar los servicios de red, comenzando por:

-**NGINX**, que actuará como **servidor web** y **proxy**;

-**libsqlite3-dev,** para la gestión de bases de datos **SQLite3** en Node.js;

-**curl** (programa para transferir datos a través de diversos protocolos), que utilizaremos para **instalar** Node.js;

-**git**, para clonar los repositorios del proyecto.

-**ufw**, para configurar el firewall.

```
# Primero actualizamos las dependencias y paquetes del sistema
sudo apt update && sudo apt upgrade -y
# Después instalamos las dependencias específicas del proyecto
sudo apt install build-essential libsqlite3-dev nginx git curl python3 g++
make ufw sqlite3
```

Después de instalar las dependencias básicas, utilizaremos git y make para compilar e instalar **Node.js 18.20.8** desde el código fuente. Esto se debe a que la versión disponible en los repositorios oficiales de Debian y Ubuntu es más reciente y **no compatible** con el servicio.

```
# Instalar NVM
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.3/install.sh |
bash
# Configurar
\. "$HOME/.nvm/nvm.sh"

# Instalar NodeJS 18.20.8
nvm install 18.20.8
# Config para bash
echo 'export PATH="$HOME/.npm-global/bin:$PATH"' >> ~/.bashrc
source ~/.bashrc
# Config para zsh
echo 'export PATH="$HOME/.npm-global/bin:$PATH"' >> ~/.zshrc
source ~/.zshrc
```

Una vez que **NodeJS** y **npm** están instalados completamos la instalación de dependencias instalando **pnpm** de forma global utilizando npm.

```
npm install -g pnpm@latest-10
```

Configuración del servicio de red

Levantar la api REST de Spacebar

Basándonos en la documentación oficial, para tener total funcionamiento del servicio de red **SpacebarChat** necesitamos tener una instancia de la **api REST** funcional y accesible por la **interfaz web**, para eso vamos a abrir una terminal y escribir los siguientes comandos:

```
# Clonar el repositorio oficial de SpacebarChat y moverse dentro
git clone https://github.com/spacebarchat/server && cd server
# Instalar las dependencias del servidor backend.
npm install
# Preparar la configuración del servicio.
npm run setup
```

Esto va dejar al servidor listo para ser **iniciado**, pero aún falta agregar algunas configuraciones específicas que van a permitir que **cualquier dispositivo** en la red pueda **acceder a nuestra aplicación** (se debe tener en cuenta conocer la **ip** de su máquina en la red y reemplazar **<HOST_IP>** por ésta; esto se puede lograr utilizando el comando **ip a**, entre otras formas). El protocolo podría variar dependiendo de cuál se intente utilizar, en este caso iremos con **http** y **ws.**

Por otra parte, la versión de la API que utilizaremos será la **versión 9**, pero esto es configurable para más viejas y nuevas versiones (de haberlas en el futuro).

Dependiendo de la escalabilidad, se aconseja utilizar otra base de datos que no sea la por defecto.

```
# Abrir el motor sqlite utilizando la base de datos del proyecto por
defecto
sqlite3 database.db
# configurar la ip del servidor backend con la versión 9
update config set value = '"http://<HOST_IP>/api/v9"' where key =
"api_endpointPublic";
# configurar la ip del gateway
update config set value = '"ws://<HOST_IP>"' where key =
"gateway_endpointPublic";
#Configurar la ip del cdn
update config set value = '"http://<HOST_IP>"' where key =
"cdn_endpointPublic";
#Salir de la db
.exit
# Iniciar servidor
npm run start
```

Una vez configurado los registros de la base de datos el servicio está listo para ser levantado usando **npm run start** dentro de la carpeta de **server** que se encargará de levantar tanto la Spacebar API, como el Gateway, el CDN Server y WebRTC.

Configurar NGINX

Ahora vamos a crear el archivos de configuración de NGINX para el servicio frontend:

```
# Quitamos el archivo por defecto de NGINX.
> sudo rm /etc/nginx/sites-available/default
# Creamos un nuevo archivo, donde se debe copiar el texto de abajo.
> sudo nano /etc/nginx/sites-available/default
--------------------------------------------------------------------------------
server {
       listen 80;
       root /var/www/html;
       index index.html;
       location = / {
              proxy_pass http://127.0.0.1:3001;
              proxy_http_version 1.1;
              proxy_set_header Upgrade $http_upgrade;
              proxy_set_header Connection "upgrade";
              proxy_set_header Host $host;
              proxy_read_timeout 86400s;
              proxy_send_timeout 86400s;
       }
       location / {
              try_files $uri /index.html;
       }
       location /api {
              proxy_pass http://127.0.0.1:3001;
              proxy_set_header Host $host;
              proxy_pass_request_headers on;
              add_header Last-Modified $date_gmt;
              add_header Cache-Control 'no-store, no-cache, must-revalidate,
proxy-revalidate, max-age=0';
              proxy_set_header  X-Real-IP $remote_addr;
              proxy_set_header  X-Forwarded-Proto $scheme;
              proxy_set_header  X-Forwarded-For $remote_addr;
              proxy_set_header  X-Forwarded-Host $host;
              proxy_no_cache 1;
              proxy_cache_bypass 1;

       proxy_http_version 1.1;
              proxy_set_header Upgrade $http_upgrade;
              proxy_set_header Connection "upgrade";
       proxy_read_timeout 86400s;
       proxy_send_timeout 86400s;
       }
}
```

Esta configuración habilita al servidor de NGINX escuchar el puerto 80, buscar sus archivos para dar el servicio en `/var/www/html`, tomando como index a un index.html. También configuramos un proxy para que todas las peticiones `/api` sobre este servidor se dirijan al servidor backend (alojado en el puerto **:3001**) a ser procesadas allí. También, cualquier petición a `/` enviará el cliente para que el usuario pueda utilizarlo.

Después de haber configurado el archivo de NGINX, vamos a **exponer** el **puerto 80** (usando **ufw** para configurar el firewall), verificar la configuración y **reiniciar el servicio** de NGINX:

```
sudo ufw allow 80/tcp
sudo nginx -t && sudo systemctl reload nginx
```

Servir la interfaz web

Una vez que **NGINX** está completamente **configurado**, procederemos a preparar la interfaz web de SpacebarChat para que sea servida por el NGINX. Esto consiste en **clonar el repositorio del cliente** web de **SpacebarChat**, **compilar** el sitio web y **mover** los archivos generados a la carpeta previamente creada **`/var/www/html`**.

Es necesario aclarar que el cliente se encuentra **en desarrollo** actualmente, lo cual lo hace algo inestable en ciertas versiones. Por esto, tomaremos la versión **0.1.2** del proyecto.

```
# Clonar el repositorio de SpacebarChat en una versión compatible
git clone --branch client-dev-v0.1.2 --single-branch
https://github.com/spacebarchat/client && cd client
# Instalar las dependencias de SpacebarChat
pnpm install
# Compilar la app a html/js/css en la carpeta dist
pnpm run build
# Vaciar la carpeta donde se encontrará el cliente
rm -r /var/www/html/*
# Mover los archivos compilados a NGINX
sudo mv dist/* /var/www/html
```

Iniciar NGINX

Una vez completado los pasos anteriores debemos **iniciar** NGINX para que provea la interfaz web, y así poder acceder y utilizar el servicio de red de **SpacebarChat.** Esto ofrecerá el cliente de SpacebarChat a la red, bajo el puerto **80**.

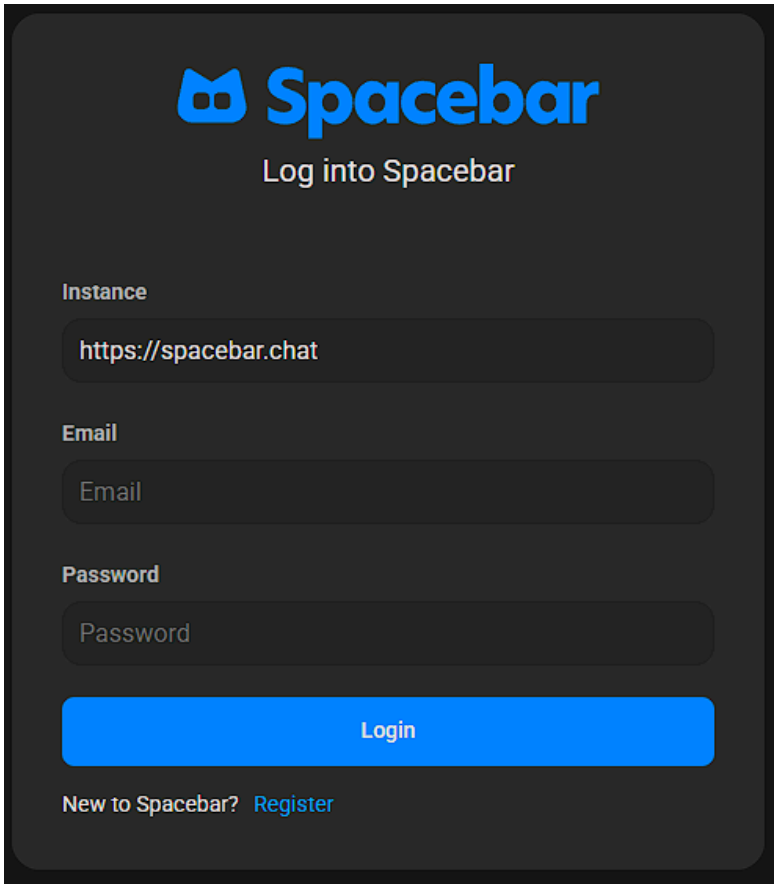
```
# Habilitar el servicio de NGINX
sudo systemctl enable nginx
# Reiniciar el servicio de NGINX
sudo systemctl reload nginx
# Verificar estado del servicio
sudo systemctl status nginx
```

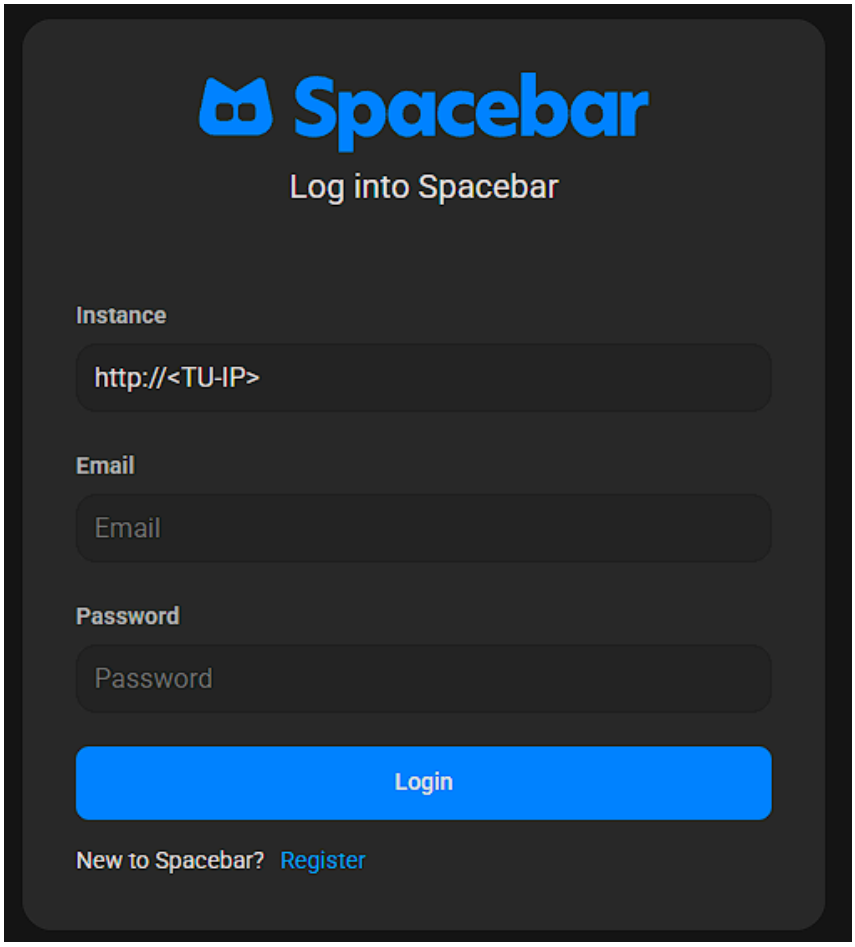
Guía de uso

Ahora, con el servidor backend y frontend funcionando, deberíamos poder entrar en el cliente utilizando **navegador**, entrando a la **URL http://<TU-IP>** (Al igual que pasos anteriores, se debe colocar en "<TU-IP>" la IP utilizada para configurar el servidor backend, y el protocolo respectivo). También, SpacebarChat provee un servicio de cliente propio al que se puede acceder desde aquí.

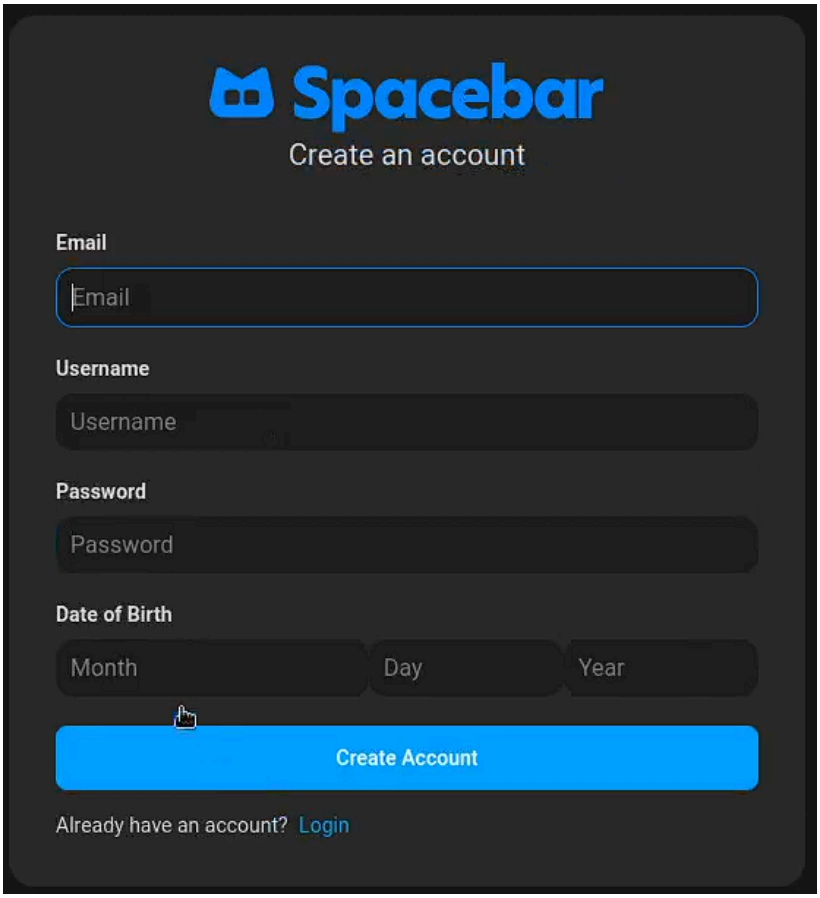
Aquí podremos entrar con una cuenta a la aplicación para utilizarla plenamente. Primero, deberemos cambiar el campo **Instance** arriba de todo, el cual intenta tomar un backend de SpacebarChat a través de una URL. Para utilizar el servidor que creamos anteriormente, habrá que colocar la misma URL del navegador: **http://<TU-IP>.**

De lo contrario, SpacebarChat ofrece un servidor backend suyo para utilizar.

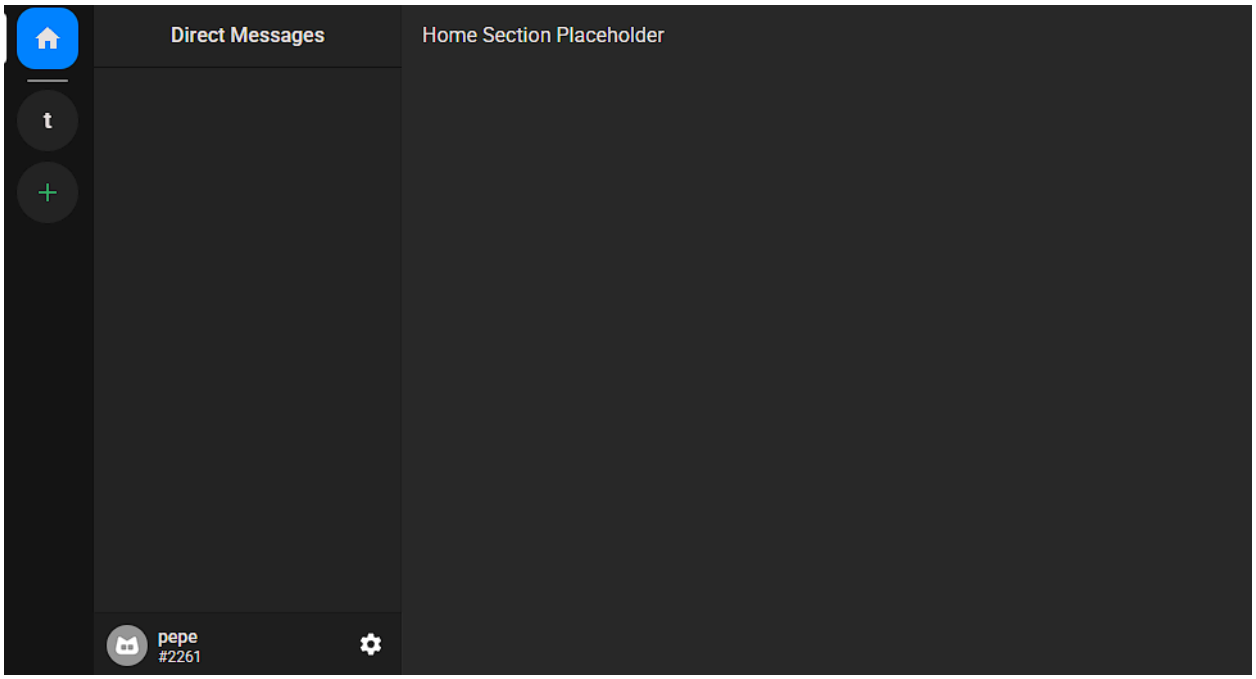




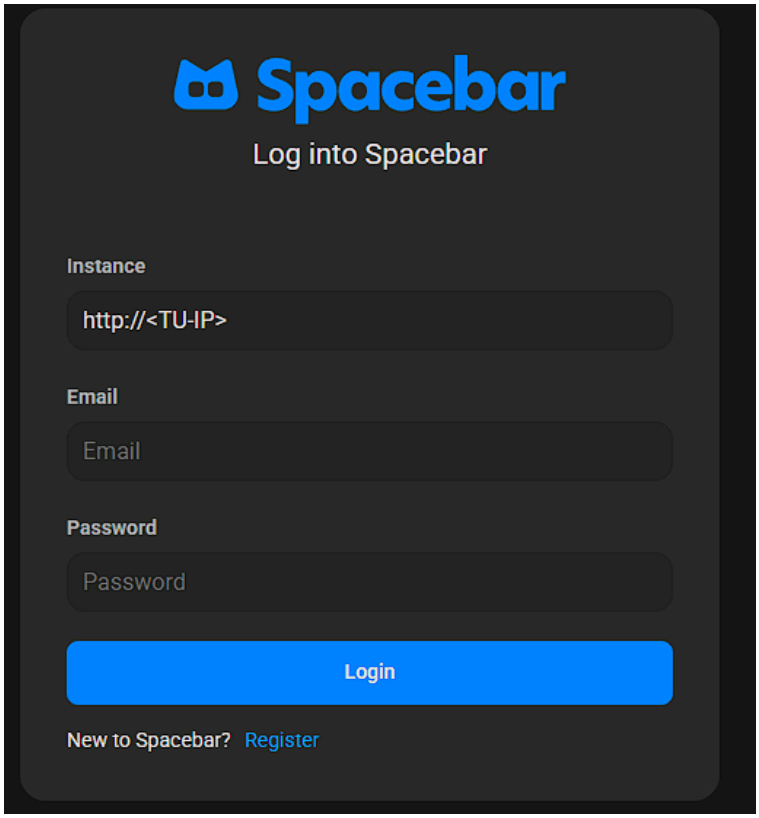
Luego, debemos crear una cuenta, **registrándose** con el botón debajo de “Login”, indicando un email, nombre de usuario, contraseña, y fecha de nacimiento, en ese orden.



Luego de todo esto, se debe dar al botón "**Create Account**" para finalizar la creación de la cuenta. Si todo va bien, debería iniciar sesión de manera automática.



Desde aquí se podrá **utilizar** la **aplicación** a su máximo esplendor. Se pueden crear canales con el botón de “+”, compartirlos a otros usuarios mediante URLs de invitación, y también cerrar sesión desde la rueda de configuración.



Luego, si se tiene una cuenta **ya creada**, se debe introducir email y luego contraseña (cambiando el campo instance por http://<TU-IP>, como en pasos anteriores).

Problemas durante el desarrollo del proyecto

NodeJS incompatible:

Por otra parte, encontramos que las versiones más recientes de NodeJS (npm) no eran compatibles con el proyecto en los repositorios de Debian y Ubuntu, por lo que nos vimos forzados a utilizar una versión más antigua (**18.20.8)**. Esto es un proceso más complejo de lo que pensábamos, y requerimos varias herramientas para lograrlo: **curl** (para instalar nvm) y **nvm** (para elegir la versión de NodeJS), además de configurar para que npm sea utilizable directamente desde la terminal, como se mostró anteriormente.

```
# Instalar NVM
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.3/install.sh |
bash
# Configurar
\. "$HOME/.nvm/nvm.sh"

# Instalar NodeJS 18.20.8
nvm install 18.20.8
# Cambiamos la ubicación de las dependencias globales
mkdir -p ~/.npm-global
npm config set prefix '~/.npm-global'
# Config para bash
echo 'export PATH="$HOME/.npm-global/bin:$PATH"' >> ~/.bashrc
source ~/.bashrc
# Config para zsh
echo 'export PATH="$HOME/.npm-global/bin:$PATH"' >> ~/.zshrc
source ~/.zshrc
```

Utilizando npm y pnpm:

En un principio, intentamos utilizar npm para instalar tanto las dependencias del cliente como del servidor, pero esto causó problemas de compatibilidad. Se debe instalar el **servidor** utilizando **npm**; pero luego, también nos dimos cuenta que el cliente **no** se puede instalar con **npm sino** con **pnpm**, por problemas similares.

Configuración del proxy:

Sin una configuración correcta del proxy, se hacía una redirección hacia una ip del estilo: http://<IP>/api//etc

Esto, provocaba que fallase el proceso y por defecto el cliente utilizara como **instancia** al **servidor backend del equipo de SpacebarChat**. Hubo entonces necesidad de continuar configurando el proxy para que pudiese funcionar correctamente, eliminando la barra extra.

```
server {
	listen 80;
	root /var/www/html;
	index index.html;
	location = / {
		proxy_pass http://127.0.0.1:3001;
		proxy_http_version 1.1;
		proxy_set_header Upgrade $http_upgrade;
		proxy_set_header Connection "upgrade";
		proxy_set_header Host $host;
		proxy_read_timeout 86400s;
		proxy_send_timeout 86400s;
	}
	location / {
		try_files $uri /index.html;
	}
	location /api {
		proxy_pass http://127.0.0.1:3001;
		proxy_set_header Host $host;
		proxy_pass_request_headers on;
		add_header Last-Modified $date_gmt;
		add_header Cache-Control 'no-store, no-cache, must-revalidate,
proxy-revalidate, max-age=0';
		proxy_set_header  X-Real-IP $remote_addr;
		proxy_set_header  X-Forwarded-Proto $scheme;
		proxy_set_header  X-Forwarded-For $remote_addr;
		proxy_set_header  X-Forwarded-Host $host;
		proxy_no_cache 1;
		proxy_cache_bypass 1;

	proxy_http_version 1.1;
		proxy_set_header Upgrade $http_upgrade;
		proxy_set_header Connection "upgrade";
	proxy_read_timeout 86400s;
	proxy_send_timeout 86400s;
	}
}
```