Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

ingweb-tx-1

Tecnologías Web

Autor:	Toral Pallás, Héctor - 798095
Grado:	Ingeniería Informática
Curso:	2022-2023

18 de noviembre de 2022

Índice

1. Resumen

En este informe se analizan aspectos positivos y negativos de diferentes protocolos, librerías y frameworks para concluir cuando y/o porque si o no deberían usarse estos.

En el informe se saca como conclusión que la librería de tRPC y el protocolo Web Sockets pueden sustituirse por otras alternativas, en el primer caso por el uso de una web api normal con Next js o si se quiere seguir el modelo de rpc con grpc. Por otro lado, el caso de Web Sockets se concluye que se debe tener muy clara la decisión de implementar la lógica entera del web socket en vez de usar alternativas como socket io. También se menciona Next JS como framework para hacer desarrollo fullstack y el hook desarrollado por los mismos creadores useSWR. Este segundo se considera una de las mejores opciones para implementar y simular experiencias de tiempo real con el patrón de optimistic UI mostrando datos generados en el cliente incluso antes de haberlos guardado en la base de datos. En cuanto al caso de Next js se concluye que para ciertos casos de desarrollo suele ser una buena opción debido al amplio abanico de opciones y posibilidades que ofrece para crear proyectos pequeños y medianos de forma rápida y sencilla sin necesidad de gran configuración permitiendo al desarrollador una gran granularidad en cuanto a la estrategia de renderizado de sus componentes.

2. tRPC

Introducción

tRPC o llamada procedimiento remoto de typescript es una librería que permite construir y consumir fácilmente APIs seguras sin esquemas ni generación de código en proyectos de javascript o typescript.

Comparativa

Ventajas:

- Seguridad al usar tipos estáticos y autocompletado en el cliente, entradas, salidas y errores.
- Sin generación de código, aumento del tiempo de ejecución ni canalización de compilación.
- Agnóstico al framework.
- Soporte de suscripciones (beta)
- Solicitudes por lotes: Las solicitudes realizadas al mismo tiempo se pueden combinar automáticamente en una sola.

Desventajas:

- Solución válida para proyectos que usen javascript o typescript.
- El autocompletado de tipos solo es válido para monorepos.

Conclusiones

tRPC puede ser una solución válida para realizar un mecanismo de RPC aunque viendo como esta planteado en la página oficial, si se decide adoptar esta solución en entornos como Next js no tendría tanto sentido ya que no aportaría ninguna ventaja que marcara la diferencia entre usar una web API con una documentación que acompañase a los endpoints implementados.

3. useSWR

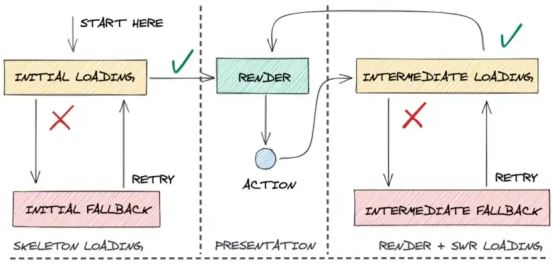

Figura 1: ejemplo del workflow de useSWR

Introducción

useSWR es una librería para la obtención de datos. Su nombre deriva de "stale while revalidate"(obsoleto mientras se revalida) y consiste en una estrategia de invalidación de caché popularizada por HTTP RFC 5861. SWR es una estrategia para devolver primero los datos en caché (obsoletos), luego envíe la solicitud de recuperación (revalidación), y finalmente entregue los datos actualizados.

Comparativa

useSWR permite con una sola línea de código simplificar la obtención de datos en tu proyecto, además de incluir las siguientes características:

- Obtención de datos rápida, ligera y reutilizable
- Caché integrada y deduplicación de solicitudes
- Experiencia en tiempo real
- Agnóstico al transporte y al protocolo
- soporte para SSR / ISR / SSG

SWR cubre aspectos de velocidad, corrección, y estabilidad para ayudar a los desarrollador a construir experiencias mejores mediante:

- "Polling on interval"
- Dependencia de los datos
- Revalidation on focus"
- Revalidation on network recovery"
- Mutación local (Optimistic UI)
- "Smart error retry"
- "Pagination and scroll position recovery"
- React Suspense"

Como aspecto negativo se podría decir que solo se puede usar en proyectos de React ya que es un hook pensado para ser usado en este framework.

Conclusiones

useSWR es una opción a tener en cuenta para aquellas situaciones donde se pretenda realizar una interfaz de usuario que requiera del patrón de Optimistic UI permitiendo simular una experiencia de tiempo real mientras por debajo se van cargando todos los datos para cachearlos en el navegador.

De esta manera, podemos simplificar el código evitando el uso de useEffects y useStates a la vez que mejoramos la experiencia de usuario teniendo siempre los datos más recientes.

4. WebSocket

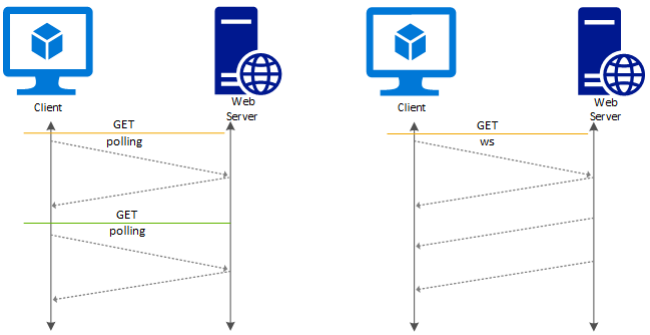

Figura 2: long polling vs web sockets

Introducción

Web Socket es un protocolo que proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP.

Con este protocolo, se puede enviar mensajes a un servidor y recibir respuestas controladas por eventos sin tener que consultar al servidor para recibir una respuesta.

Comparativa

Ventajas:

- Eliminación de la necesidad de una nueva conexión con cada solicitud, reduciendo drásticamente el tamaño de cada mensaje (sin cabeceras HTTP). Esto favorece el ahorro de ancho de banda, mejora la latencia y hace que los WebSockets sean menos exigentes en el lado del servidor en comparación con HTTP.
- La flexibilidad es inherente al protocolo permitiendo la implementación de nuevos protocolos/funcionalidades a nivel de aplicación.
- Permite transferir datos sin que el cliente los solicite. Esta característica es ideal en escenarios en los que el cliente necesita reaccionar rápidamente ante un evento.

Desventajas:

- A diferencia de HTTP, Web Sockets tiene estado, lo que puede suponer un problema a gran escala, porque requiere del servidor para almacenar información sobre cada conexión individual y mantener la información de estado.
- Los Web Sockets no se recuperan automáticamente cuando se cierran las conexiones esto es una característica que tienes que implementar tú mismo.
- Algunos entornos (con servidores proxy) bloquean las conexiones además, algunos navegadores todavía no admiten Web Sockets.

Conclusiones

Para elegir la opción de una implementación de Web Sockets desde 0 deberíamos haber descartado previamente que nuestro uso no puede ser realizado por alguna otra librería que implemente ya Web Sockets ya que de lo contrario estaríamos perdiendo dinero en un trabajo que ya han realizado otras personas. En aquellos casos en los que necesitamos mas granularidad como podría ser el caso de las reconexiones o el tema de la escala, debería de preguntarse si no se puede escalar el servicio de otra forma y una vez entonces, apostar por la opción que sea más conveniente.

5. Socket IO

Introducción

Socket IO es una librería basada en eventos para aplicaciones web en tiempo real. Permite la comunicación bidireccional en tiempo real entre clientes web y servidores. En otras palabras sería una implementación sobre el protocolo de Web Sockets.

Comparativa

Ventajas:

- Soporta la multiplexación a través de espacios de nombres. Hacer uso de los espacios de nombres le permite minimizar el número de conexiones TCP utilizadas, y ahorrar puertos de socket en el servidor.
- Permite que el lado del servidor emita eventos de forma flexible a todos los clientes conectados. También puede emitir eventos a un subconjunto de clientes mediante la función de salas.
- Ofrece la opción de HTTP long-polling como alternativa, que es útil en entornos que no soportan Web Sockets.
- Proporciona un mecanismo configurable de latido de corazón que le permite detectar si una conexión está viva o no. Además, si un cliente se desconecta, se reconecta automáticamente.

Desventajas:

- No proporciona garantías sólidas de ordenación de los mensajes, especialmente cuando se utiliza el transporte de long-polling.
- Viene con características de seguridad nativas limitadas. Por ejemplo, no proporciona cifrado de extremo a extremo, y no ofrece un mecanismo para generar y renovar tokens para la autenticación.
- No es compatible con ninguna otra implementación de Web Socket.
- No está diseñado para trabajar en varias regiones. Esto puede dar lugar a problemas como el aumento de la latencia (si sus usuarios están en diferentes regiones), e incluso el tiempo de inactividad del sistema.

Conclusiones

Socket IO sería una buena opción para aquellos proyectos que requieran de un flujo de datos en tiempo real entre cliente y servidor y, cuyos requisitos puedan ser satisfechos en su totalidad por dicha librería debido al impedimento/incompatibilidad de esta con otras implementaciones de Web Sockets.

Si se cumplen estas características, el usar Socket IO puede suponer una simplificación en el uso del protocolo permitiendo a los desarrolladores centrarse en la lógica de su aplicación.

6. NEXT js

Introducción

Next js es el framework de React para producción de código abierto creado por Vercel que habilita funcionalidades como SSR, SSG, CSR, ISR y enrutado dinámico.

Comparativa

Ventajas:

- Permite escoger la estrategia de renderizado de cada página.
- Rápido testeo de ideas en el mercado. (MVP)
- Tiempos de carga más rápidos debido al uso de la optimización de imagenes, el generador de código estático, ejecutarse en el Edge(CDN)...
- Encontrar desarrolladores para Next JS es sencillo ya que sigue muchos de los principios del stack MERN.
- Compresión de los archivos servidos por defecto.

Desventajas:

- No da la opción de escoger el tipo de enrutado y sigue su filosofía "page-based". (Podría considerarse a su vez como una ventaja)
- Es un monolito al no separar la parte del cliente de la lógica de la aplicación.

Conclusiones

Next js es una muy buena opción a la hora de desarrollar aplicaciones web de tamaño pequeño/medio debido a que para empezar trae todos los componentes necesarios para crear un producto profesional por defecto.

En primer lugar, te ofrece una estructura de archivos que ayuda al desarrollador a mantener el repositorio limpio, optimizaciones en los envíos de mensajes gracias a su configuración con webpack($\leq$ v12) y turbopack ($\geq$ 13), la red que mantiene (vercel) para ofrecer el contenido estático y ejecutar ciertos "scripts"desde ahí sin tener que llegar al servidor (Edge), la granularidad para elegir el tipo de renderizado de cada pagina, etc.

El principal de los inconvenientes sería que durante la vida del producto, el código fuera creciendo de forma descontrolada, provocando así un mantenimiento tedioso y complicado, esto podría ser solucionado fácilmente separando la parte más de "backend"de la del "frontendïmplementando por ejemplo un servidor con Express js ya que al correr ambos sobre Node, el código es 90 % compatible en ambas direcciones.

7. Anexo

En el anexo se exponen algunos de los enlaces visitados a la hora de redactar el informe:

- trpc
- useSWR
- Web Sockets API
- WS RFC 6455
- Socket IO
- nextjs
- gRPC
- HTTP RFC 5861
- Comet
- Scaling Socket.IO
- An introduction to Socket.IO
- Build a full-stack TypeScript app using tRPC and React
- What is tRPC?
- Figura 1
- Figura 2