GABRIELE ROMANATO

Subscription e notifiche in Supabase con Node.js e WebSocket

Il client JavaScript di Supabase funziona anche lato server se si utilizza Node.js per sviluppare il backend. Se vogliamo comunicare in tempo reale con il frontend per segnalare le avvenute modififche sulle tabelle del database, possiamo utilizzare le subscription di Supabase e i WebSocket.

Moduli NPM

Per prima cosa installiamo i moduli NPM fondamentali:

```
npm install @supabase/supabase-js ws --save
```

Il client Supabase

Creiamo quindi un modulo della nostra applicazione in cui andremo ad istanziare il client Supabase, ad esempio in `client/index.js`.

```
'use strict';

const { createClient } = require('@supabase/supabase-js');
const supabaseUrl = 'Supabase-URL';
const supabaseKey = 'Supabase-KEY';
const supabase = createClient(supabaseUrl, supabaseKey);

module.exports = {
  supabase,
};
```

Implementare le notifiche

A questo punto, nel file principale della nostra applicazione, dobbiamo implementare i seguenti passaggi:

1. Definire un'istanza di un server WebSocket.
2. Definire un elenco di client connessi evitando al suo interno di avere elementi duplicati.
3. Creare nel client Supabase un canale in cui restare in ascolto di uno specifico evento del database su una specifica tabella.
4. Inviare la notifica dell'avvenuto evento a tutti i client registrati al punto 2.

1. Creare il server WebSocket

Dopo aver importato il modulo `ws` e il client Supabase, creiamo un'istanza del server WebSocket.

```
'use strict';

const { WebSocketServer } = require('ws');
const { supabase } = require('./client');

const wss = new WebSocketServer({ noServer: true });
```

2. Gestione dei client connessi

Per avere un elenco di elementi univoci per rappresentare i client connessi, possiamo utilizzare il tipo di dati `Set` e i metodi ad esso correlati.

```
const clients = new Set();

wss.on('connection', (ws) => {
  clients.add(ws);

  ws.on('close', () => {
    clients.delete(ws);
  });
});
```

3. Gestire le subscription di Supabase

Per gestire le subscription occorre creare un canale di ascolto. Possiamo monitorare le operazioni di creazione di nuovi record andando a gestire i risultati delle query `INSERT` ottenendo come payload il record appena inserito.

```
const startSubscription = async () => {
  const channel = supabase
    .channel('posts_updates')
    .on(
      'postgres_changes',
      { event: 'INSERT', schema: 'public', table: 'posts' },
      (payload) => {
        // Accediamo al nuovo record inserito
      }
    )
    .subscribe();
};

startSubscription();
```

Diamo come prima cosa il nome al nostro canale (`posts_updates`), quindi specifichiamo l'evento della tabella, lo schema di riferimento e il nome della tabella. Ogni qualvolta viene inserito un nuovo record, il parametro della funzione di callback ci permetterà di accedere alla riga appena inserita.

4. Inviare le notifiche ai client

Sfruttando la funzione di callback del gestore di eventi vista in precedenza, tramite WebSocket possiamo facilmente inviare come stringa JSON l'oggetto che rappresenta il nuovo record inserito a tutti i client connessi.

```
const startSubscription = async () => {
  const channel = supabase
    .channel('posts_updates')
    .on(
      'postgres_changes',
      { event: 'INSERT', schema: 'public', table: 'posts' },
      (payload) => {
        for (const client of clients) {
          client.send(JSON.stringify(payload.new));
        }
      }
    )
    .subscribe();
};

startSubscription();
```

Conclusioni

La comunicazione in tempo reale tra backend e frontend in Supabase con Node.js viene enormemente semplificata se si utilizza una tecnologia specifica come i WebSocket.

Applicazioni Correlate

- 

Node.js Placeholder Image

Applicazione per la generazione con Node.js di immagini segnaposto.
DockerDocker ComposeNode.jsJavaScriptExpressJS

Node.js URL Shortener

Implementazione in Node.js di un sistema per l'abbreviazione degli URL.
DockerDocker ComposeNode.jsJavaScriptExpressJSMongoDB

- ### JavaScript App Hash Change

  Applicazione che sfrutta gli hash degli URL per gestire contenuto dinamico in JavaScript. DockerDocker ComposeNode.jsJavaScriptExpressJSMySQL