GABRIELE ROMANATO

React: aggiungere il Bearer Token nelle richieste HTTP

React è una libreria JavaScript ampiamente utilizzata per la creazione di interfacce utente dinamiche e reattive. Quando si sviluppa un'applicazione web con React che si interfaccia con un server API sicuro, spesso è necessario autenticare le richieste HTTP utilizzando un token di accesso Bearer. In questo articolo, esploreremo come aggiungere un Bearer Token a tutte le richieste HTTP in un'applicazione React.

Ottenere il Bearer Token

Prima di tutto, è necessario ottenere un token di accesso Bearer dall'endpoint di autenticazione del server. Questo di solito avviene attraverso un processo di login, dove le credenziali dell'utente vengono inviate al server e, se valide, viene restituito un token di accesso.

```
// Esempio di richiesta di accesso e ottenimento del token
fetch('https://api.example.com/login', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({
    username: 'yourUsername',
    password: 'yourPassword',
  }),
})
  .then(response => response.json())
  .then(data => {
    const accessToken = data.accessToken;
    // Salvare il token di accesso in uno stato di React o in un sistema di gestione
dello stato
  })
  .catch(error => console.error('Errore durante il login:', error));
```

Creare un Wrapper per le Richieste HTTP

Una volta ottenuto il token di accesso, possiamo creare un wrapper per le richieste HTTP che includa automaticamente il token in ogni richiesta. Possiamo farlo utilizzando l'interceptor delle richieste di una libreria per le chiamate HTTP, come ad esempio Axios.

```
// Wrapper per le richieste HTTP con Axios
import axios from 'axios';

const api = axios.create({
  baseURL: 'https://api.example.com',
});

// Aggiungi un interceptor per tutte le richieste
api.interceptors.request.use(config => {
  // Recupera il token di accesso dallo stato di React o da un sistema di gestione dello
stato
```

```
  const accessToken = 'yourAccessToken';

  // Aggiungi il token di accesso all'header Authorization
  config.headers.Authorization = `Bearer ${accessToken}`;

  return config;
});

export default api;
```

Utilizzare il Wrapper nelle Chiamate API

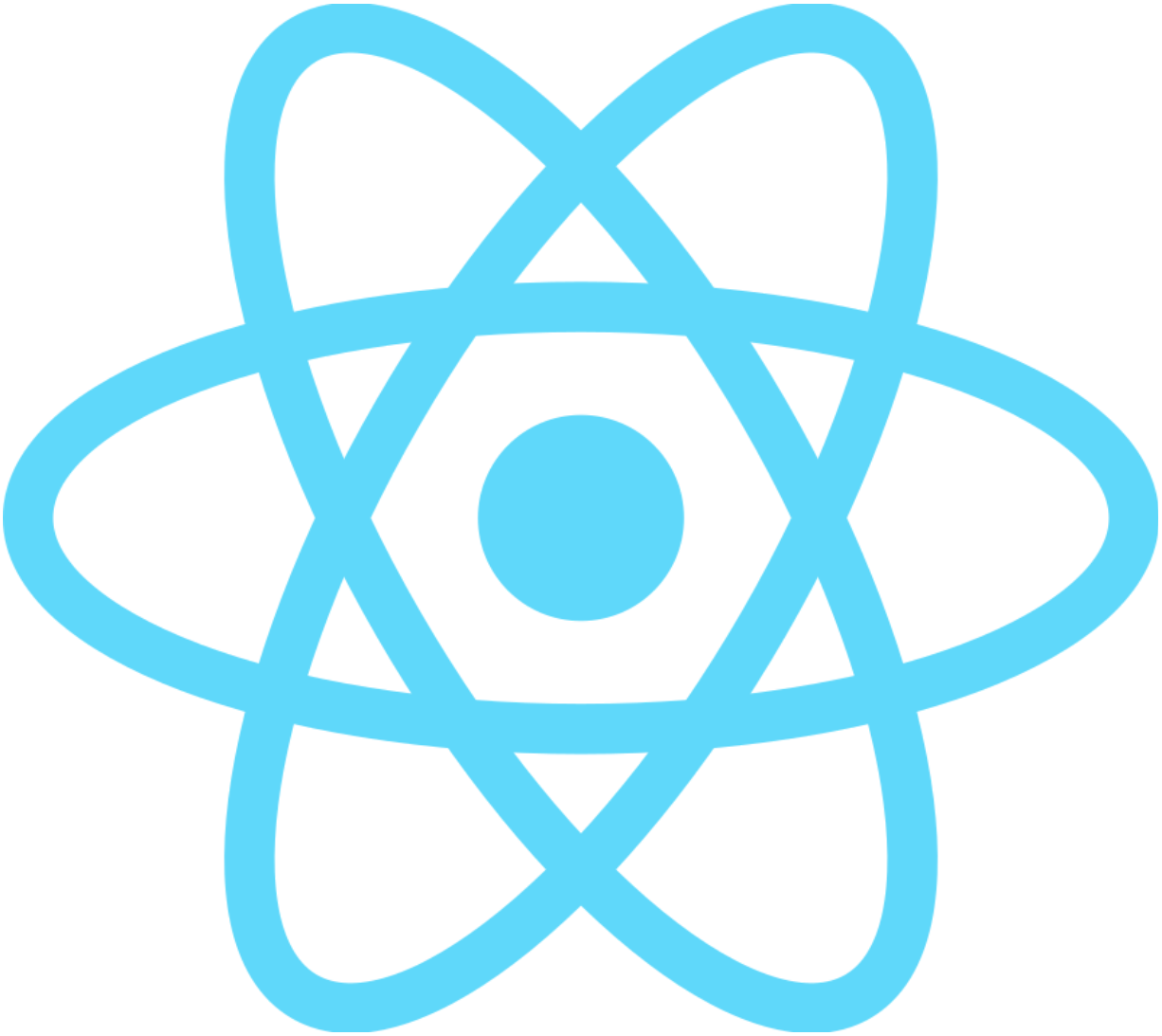
Ora che abbiamo creato il wrapper per le richieste HTTP, possiamo utilizzarlo nelle nostre chiamate API.

```
// Esempio di utilizzo del wrapper per le richieste HTTP
import api from './api'; // Importa il wrapper

// Effettua una richiesta GET
api.get('/data')
  .then(response => {
    // Gestisci la risposta
    console.log(response.data);
  })
  .catch(error => {
    // Gestisci gli errori
    console.error('Errore nella richiesta:', error);
  });
```

Con questo approccio, ogni richiesta HTTP effettuata con il wrapper Axios includerà automaticamente il token di accesso Bearer nell'header Authorization. Questo semplifica la gestione dell'autenticazione nelle tue applicazioni React, garantendo che ogni richiesta al server sia autorizzata correttamente.

Applicazioni Correlate



Book Store

Un'applicazione in React con Node.js e Fastify come backend.
DockerDocker ComposeNode.jsJavaScriptFastifyReact