チャットUI用APIチュートリアル［プレビュー版］

はじめに

本書は、本サービスが提供するチャットUI用APIによりWebアプリケーションにログイン機能とチャット機能を実装する方法についてチュートリアル形式で説明します。

本書の対象読者は以下を想定しています。

- 本サービスを用いてWeb画面にチャット機能を実装するフロントエンド開発者

なお、本書ではReactを用いたサンプルコードを記載しておりますが、REST API にて機能提供しており他のWebアプリケーションフレームワークでもご利用いただくことができます。

> チャットUI用APIは現在プレビュー版です。利用をご希望の場合には問い合わせ窓口よりチャットUI用APIを利用したい旨を事前にお申し出ください。

チャットUI用APIとは

管理ポータルで管理するユーザによりログインして取得するJWT用いて認証を行い、対話機能を利用するためのAPIです。本API用いることでWebアプリケーションにログイン機能とチャット機能を実装することができます。

サポートしているAPIとリクエスト・レスポンスの詳細については、APIリファレンスをご参照ください。

チュートリアルの流れ

以下の手順で説明します。なお、事前にNode.jsをインストールし利用可能な状態になっていることが前提となります。

1. Reactプロジェクトの作成と必要なライブラリのインストール
2. 認証設定をアプリケーションに組み込み
3. ログイン機能とチャット機能の実装

Reactプロジェクトの作成と必要なライブラリのインストール

Reactプロジェクトの作成後、ログイン機能の実装に必要な Microsoft Authentication Library (MSAL) を導入します。以下のコマンドを実行します。

```
// Reactプロジェクトの作成
npm create vite@latest chat-ui-api-tutorial -- --template react-ts

// Microsoft Authentication Library (MSAL) を導入
cd chat-ui-api-tutorial
npm install @azure/msal-browser @azure/msal-react
```

認証設定をアプリケーションに組み込み

main.tsx においてアプリケーションに認証のためのMSALの設定をMsalProviderを用いて組み込みます。以下は実装例です。ソースコード中の `xxxx` の箇所は申請により払い出された文字列を指定します。

```
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.tsx'
import { PublicClientApplication } from '@azure/msal-browser';
import { MsalProvider } from '@azure/msal-react';

const msalConfig = {
```

```
  auth: {
    clientId: 'xxxx',
    authority: 'xxxx',
    knownAuthorities: ['xxxx'],
    redirectUri: 'https://xxxx'
  },
  cache: {
    cachelocation: "SessionStorage",
    storeAuthStateInCookie: false
  }
}
export const msalInstance = new PublicClientApplication(msalConfig)

createRoot(document.getElementById('root')!).render(
  <MsalProvider instance={msalInstance}>
    <App />
  </MsalProvider>
)
```

ログイン機能とチャット機能の実装

App.tsx においてMSALライブラリから提供されるuseMsalおよびuseIsAuthenticatedによりログイン機能を実装します。さらに、acquireTokenSilent関数によりアクセストークンを取得し、APIの認証情報としてHTTPヘッダ `Authorization` に設定します。以下は実装例です。ソースコード中の `xxxx` の箇所は申請により払い出された文字列を指定します。

```
import React, { useEffect, useState } from 'react';
import './App.css';
import { useMsal, useIsAuthenticated } from "@azure/msal-react";

const REACT_APP_AADB2C_SCOPES='xxxx'
const V1_CHAT_URL = 'https://api.genai-api.nec-cloud.com/genai-ui-api/v1/chat'

export async function callChatAPI(token: string, userContent: string,
                                  historyId: string, model: string):
                                  Promise<{answer: string, historyId: string}> {
  const defaultResult = {
      answer: '',
      historyId: ''
  }
  const response = await fetch(V1_CHAT_URL, {
    method: 'POST',
    headers: {
      'authorization': 'Bearer ' + token,
      'Content-Type': 'application/json',
      'x-nec-genai-client-id': 'sample-user'
    },
    body: JSON.stringify({
      userContent,
      historyId,
      model
    })
  })
  return await response.json()
}

function App() {
  const { instance, accounts } = useMsal();
  const isAuthenticated = useIsAuthenticated();
```

```
  const [ msg, setMsg ] = useState<string>('');
  const [ answer, setAnswer ] = useState<string>('(ここに回答が表示されます)');

  const getAccessToken = async () => {
    const accessTokenRequest = {
      scopes: (REACT_APP_AADB2C_SCOPES || "")?.split(","),
      account: accounts[0],
    };

    try {
      const tokenResponse = await instance.acquireTokenSilent(accessTokenRequest);
      return tokenResponse.accessToken;
    } catch (error) {
      console.error('Failed to get access token.');
      throw error;
    }
  }

  return (
    <div className="App">
      {isAuthenticated ? (
          <>
            <h1>Chat API サンプル</h1>
            <div>
              <label htmlFor='msg'>質問：</label>
              <textarea id='msg'
                value={msg}
                onChange={(event) => setMsg(event.target.value)}
                placeholder='任意の文章を入力'
                rows={1}
              />
              <button onClick={async ()=>{
                const response = await callChatAPI(
                    await getAccessToken(), msg, 'new', 'cotomi-fast-v2.0');
                setAnswer(response.answer);
              }}>送信</button>
            </div>
            <div>
              <span>回答：</span><span>{answer}</span>
            </div>
            <hr/>
            <div>
              <button onClick={() => instance.logoutRedirect(
                  { postLogoutRedirectUri: window.location.origin })}>
                ログアウト
              </button>
            </div>
          </>
        ) : (
          <>
            <h1>ログイン画面</h1>
            <div>
              <p>以下のボタンを押してログイン</p>
              <button onClick={() => instance.loginRedirect({scopes: []})}>
                ログイン
              </button>
            </div>
          </>
```

```
        )
      }
    </div>
  );
}

export default App;
```

以下はここまでの実装例を `$ npm run dev` などのコマンドにより動作した場合の画面です。

参考：実装例のトップ画面

参考：ログインボタン押下後の認証画面

参考：実装例の対話画面