Librus-Scrapper

Dokumentacja Projektu

Spis treści

1. Opis projektu

Projekt ma na celu automatyczne pobieranie danych z systemu Librus Skrypt napisany w języku JavaScript i uruchamiany przez rozszerzenie Tampermonkey w przeglądarce, zbiera dane, następnie te są przetwarzane i wysyłane do zewnętrznej bazy danych za pomocą REST API.

Projekt został stworzony jako zadanie zaliczeniowe do szkoły.

2. Technologie

JavaScript

JavaScript to najważniejszy język programowania używany w tym projekcie. Dzięki swojej elastyczności i możliwości działania bezpośrednio w przeglądarce, jest idealnym wyborem do wyciągania danych z kodu HTML(Web Scrappingu) w połączeniu z rozszerzeniem Tampermonkey. JavaScript jest językiem skryptowym, który działa po stronie klienta, co oznacza, że jest wykonywany bezpośrednio na komputerze użytkownika za pomocą przeglądarki.

Tampermonkey

Tampermonkey to rozszerzenie przeglądarki, które umożliwia użytkownikom tworzenie i uruchamianie własnych skryptów JavaScript na stronach internetowych. W tym projekcie rozszerzenie to odgrywa kluczową rolę, ponieważ to dzięki niemu możliwe jest automatyczne uruchamianie skryptu na stronach internetowych , co pozwala na pobranie danych bez potrzeby ingerowania w serwer Librus, czy też prób przechwytywania pakietów wysyłanych do klienta.

REST API

REST API (Representational State Transfer Application Programming Interface) to interfejs, który pozwala na komunikację między skryptem a zewnętrzną bazą danych za pomocą endpointów, komunikacja odbywa się z pomocą protokołu HTTP/S. REST API w tym projekcie jest używane do przesyłania danych pobranych z systemu Librus do bazy danych.

JSON

JSON (JavaScript Object Notation) to format danych używany w projekcie do przesyłania informacji między skryptem a API. Jest to lekki format wymiany danych, który jest łatwy do odczytu i późniejszej walidacji podczas operacji na danych

ASP.NET

ASP.NET to platforma stworzona przez Microsoft do budowy aplikacji internetowych i API opierająca się o język .NET(C#). W tym projekcie język ten zostanie wykorzystany do stworzenia REST API, i relacyjnej bazy danych MySql poprzez automatyczną migracje czyli autoamtyczne genrowanie bazy danych na podstawie schematu danych używanych w API, dzięki czemu mamy pewność że baza będzie w pełni kompatybilna z REST API

Baza danych MySql

Baza danych zostanie wygenerowana za pomocą systemu migracji na platformie ASP.NET w języku MySql. W projekcie baza danych nie będzie łączyć się bezpośrednią z skryptem JavaScript, komunikacja będzie się odbywać za pomocą wcześniej wspomnianego REST API

3. Architektura rozwiązania

Projekt składa się z 2 głównych modułów

Moduł Web Scrappingu

Moduł Web Scrappingu zostanie opracowany w jeżyku JavaScript, skrypty te będą obsługiwane przez rozszerzenie do przeglądarki internetowej tampermonkey. Język JS udostępnia wiele prostych funkcjonalności takich jak obiekt *document* za pomocą którego z łatwością można manipulować kodem strony HTML jak i co w tym przypadku ważniejsze ekstraktować z niego informacje.

Moduł REST API

Moduł ten jest opracowany w języku .NET na platformie ASP, będzie on odpowiedzialny za komunikacje z bazą jak i jest utworzenie na podstawie struktury danych zawartych w modelach danych. W tym module będzie odbywać się cała walidacja danych odebranych z pomocą protokołu HTTP/S wysłanych z Modułu Web Scrappingu.

4. Struktura Bazy Danych

Struktura bazy danych może ulec zmianie podczas realizacji projektu z racji na czynniki które nie zostały przemyślane w okresie pisania dokumentacji i co ważniejsze, przyszłe zmiany w strukturze mogą być spowodowane jakością wykonania serwisu Librus.

Przykładowa struktura bazy danych:

Students:

- student_id: unikalny identyfikator
- first_name: imię ucznia
- last_name: nazwisko ucznia
- class: klasa, do której należy uczeń

Grades:

- grade_id: unikalny identyfikator oceny
- student_id relacja z tabelą Students
- subject: przedmiot, z którego została wystawiona ocena
- grade: ocena (np. 5, 4, 3, itd.)
- date: data wystawienia oceny

Attendance:

- attendance_id: unikalny identyfikator obecności
- student_id: odniesienie do tabeli Students
- date: data obecności
- status: status (obecny, nieobecny, spóźniony)
- lesson_nr: numer lekcji z którego wystawiona została obecność

5. Instalacja i konfiguracja

5.1 Wymagania

- Przeglądarka obsługująca Tampermonkey (np. Google Chrome, Firefox)
- Konto w systemie Librus
- Klucz API do bazy danych

5.2 Instalacja

Pobierz i zainstaluj rozszerzenie Tampermonkey dla swojej przeglądarki:

- Wersja dla przeglądarek opartych na chromium (np. Chrome, Opera, Brave) https://chromewebstore.google.com/detail/tampermonkey/dhdgffkkebhmkfjojejmpbldmpobfkfo?hl=pl&pli=1
- Wersja dla przeglądarki Firefox https://addons.mozilla.org/pl/firefox/addon/tampermonkey/

Dodanie skryptu do Tampermonkey:

Sposób pierwszy:

- Otwórz Tampermonkey klikając na jego ikonę w pasku narzędzi przeglądarki.
- Wybierz opcję "Utwórz nowy skrypt".
- Usuń przykładowy kod w edytorze i wklej swój skrypt.

Sposób drugi:

- Otwórz Tampermonkey klikając na jego ikonę w pasku narzędzi przeglądarki.
- W panelu Tampermonkey wybierz "Narzędzia".
- Kliknij przycisk "Import".
- wybierz plik JS skryptu z komputera.

6. Proces działania

6.1 Web Scraping

Skrypt uruchomiony w przeglądarce (przez Tampermonkey) automatycznie w zależności od strony pobiera dane, takie jak:

- przykład 1
- przykład 2
- przykład 3

Pobrane dane są następnie przetwarzane w formacie JSON.

6.2 Wysyłanie danych do bazy danych

Po przetworzeniu, dane są przesyłane do bazy danych za pomocą żądań HTTP (np. POST, PUT) przy użyciu komunikacji z REST API

7. Obsługa błędów

W trakcie działania skryptu mogą pojawić się różne błędy np. wynikające z problemów z połączeniem z REST API, czy też walidacją i parsowaniem danych już po ich wysłaniu przez skrypt JS

Obsługa błędów HTTP:

Każde żądanie http wykonane przez skrypt JavaScript jest zawarte w klauzuli try-catch, a odpowiedzi serwera są sprawdzane pod kątem kodów błędów (np. 404, 500). W razie niepowodzenia skrypt wyświetla odpowiednie komunikaty.

```
fetch(API_URL, options)
  .then(response => {
    if (!response.ok) {
      throw new Error('Network response was not ok ' + response.statusText);
    }
    return response.json();
  })
  .catch(error => console.error('There was a problem with the fetch operation:', error));
```

8. Bezpieczeństwo

- Bezpieczne połączenia - Wszelkie żądania do API są realizowane z użyciem protokołu HTTPS, co zapewnia bezpieczeństwo transmisji.
- Ochrona danych - Projekt używa tylko danych dostępnych jawnie na stronie librus, bez ingerowania w serwer strony czy przechwytywania pakietów.

9. Wymagania systemowe

- Przeglądarka internetowa z zainstalowanym Tampermonkey
- Połączenie internetowe
- Dostęp do API bazy danych

10. Autorzy

Projekt został opracowany przez: Bartosz Ujma.

11. Licencja