Projekt Scraper'a – Hurtownie danych

Członkowie projektu:

Mikołaj Męderski
Krzysztof Kowalik
Tomasz Nowakowski

Celem projektu jest zaprojektowanie i stworzenie aplikacji pobierającej, konwertującej i zapisującej dane pobrane z wskazanej strony internetowej. Docelową stroną internetową, z której scraper pobierze dane jest otoDom (https://www.otodom.pl/).

Ogólne informacje

Aplikacja ma za zadanie pobrać dane, przekonwertować je do czytelnego dla niej formatu i zapisać je w zdefiniowanej wcześniej bazie danych. Proces powinien być automatyczny, podzielony odpowiednio na 3 fazy. Faza pierwsza to pobranie interesujących użytkownika danych, w fazie drugiej dane pobrane w fazie pierwszej poddawane są obróbce i sprowadzane do stanu pozwalającego zapisać je do bazy danych, faza trzecia to eliminacja powtarzających się danych oraz ich zapis do bazy danych. Aplikacja powinna generować raporty przy każdej fazie i zależności od preferencji użytkownika powinna wykonać cały cykl pracy i wygenerować końcowy raport, lub wykonywać następujące po sobie fazy tylko po uprzednim zatwierdzeniu ich przez użytkownika oraz generować raport przy końcu każdej fazy.

Wymagania funkcjonalne.

Opis ogólny

Aplikacja powinna składać się z trzech modułów opisanych szczegółowo poniżej.
Pierwszy moduł wybiera stronę internetową, z której to pobierze dane i przygotuje je do następnego kroku. Wygeneruje on też raport ile danych udało mu się pobrać. Drugi moduł podda pobrane dane obróbce w celu konwersji ich do docelowego formatu, dzięki któremu możliwe będzie wprowadzenie danych do bazy danych. Moduł wygeneruje również raport ile danych uległo konwersji. Trzeci moduł usunie powtarzające się dane i zapisze pozostałe dane w bazie danych oraz wygeneruje końcowy raport oraz zestawienie wszystkich trzech raportów.

Użyte technologie

Aplikacja została podzielona na trzy części technologiczne. Pierwsza z nich odpowiada za interfejs użytkownika, napisana jest przy użyciu technologii frontend'owych (JavaScript, HTML5, CSS3). Druga jest częścią wykonawczą. Jest to API napisane przy użyciu języka C#. Odpowiada za pobieranie danych ze strony OTODOM, transformuje dane na docelowy format, oraz komunikuje się z bazą danych. Ostatnia - trzecia to sama baza danych.

Technologie użyte na froncie

Aplikacja po stronie frontend'u została stworzona w JavaScript, HTML5 oraz CSS3. W celu optymalizacji oparta jest na bibliotece ReactJS w wersji 16.12.0. W celu budowania aplikacji w wersji produkcyjnej oraz developerskiej używamy WebPack 4.41.0 pozwalający tworzyć paczki JavaScript

złożone z reużywalnych i łatwo zarządzalnych modułów, możliwych do odczytania przez przeglądarkę dopiero po ich skompilowaniu w jeden plik. Kolejną technologią używa w projekcie jest Babel 7.6.0 pozwalający pisać kod używając najnowszych feature'ow dodanych miedzy innymi w specyfikacji ES6 oraz nowszych. Kod jest kompliowany do standardów możliwych do odczytania nawet przez starsze przeglądarki. Do wykonania interfejsu użytkownika używamy Material-ui jednej z najpopularniejszych bibliotek UI dedykowanych dla ReactJS. Dodatkowo biblioteka material-ui pozawala na użycie material-table, która zapewnia obsługę wyświetlania tabel w przyjazny dla użytkownika sposób.

Technologie użyte w API

Aplikacja została napisana w języku C# 6.0 i używa .NET Framework w wersji 4.6.1. Głowną technologia odpowiadającą za komunikacje miedzy interfejsem użytkownika, a API jest ASP.NET Web API w wersji 2.0. Jest to technologia, która używa protokołu http, a dane wysyłane są w formacie JSON. W celu wykonywania operacji scrappingu, użyta została paczka Html Agility Pack (HAP) w wersji 1.11.17. Pomaga ona wyciągać potrzebne dane z dokumentu HTML zawierającego kod strony OTODOM. Po udanej transformacji danych, są one zapisywane do pliku .json, w tym celu użyta została paczka Newtonsof.Json w wersji 12.0.3. Pomaga ona serializować, oraz deserializować obiekty języka C#. Ostatnią technologią użytą w tej części aplikacji jest paczka CsvHelper w wersji 12.2.2, która pozwala eksportować obiekty języka C# do plików .csv.

Technologie użyte w bazie danych

Instalacja aplikacji

Front-end:

W celu zainstalowania części front-end'owej należy:

- Zainstalować NodeJs z podanego linku w najnowszej wersji rekomendowanej na oficjalnej stronie:

  https://nodejs.org/en/

- Posiadać dowolny edytor tekstu - rekomendowany Visual Studio Code, który można pobrać z oficjalnej strony:

  https://code.visualstudio.com/

- Korzystając z wiersza poleceń przechodzimy do katalogu:

  ~\WebScrapper\webscrapper-frontend

- Z wiersza poleceń uruchamiamy komendę

  npm install

Po zakończeniu procesu instalacji wszystkie niezbędne do uruchomienia pakiety zostaną zainstalowane lokalnie na komputerze.

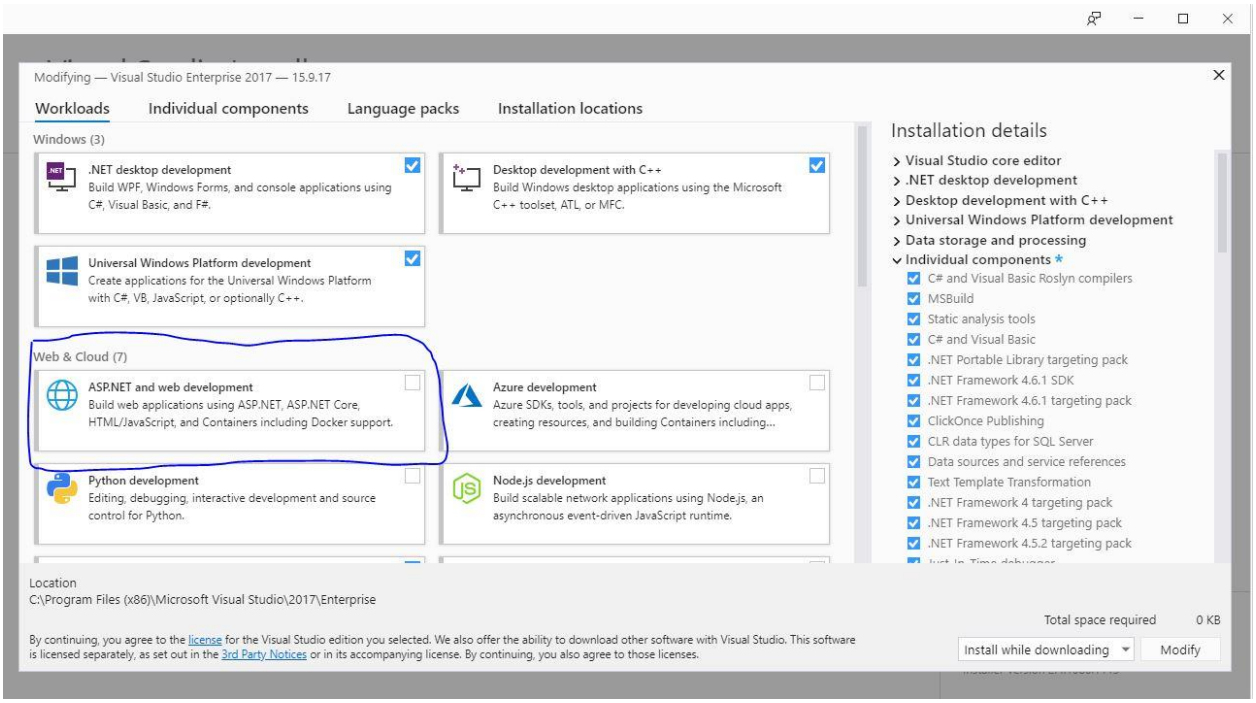
Back-end:

W celu zainstalowania API obsługującego front-end aplikacji należy:

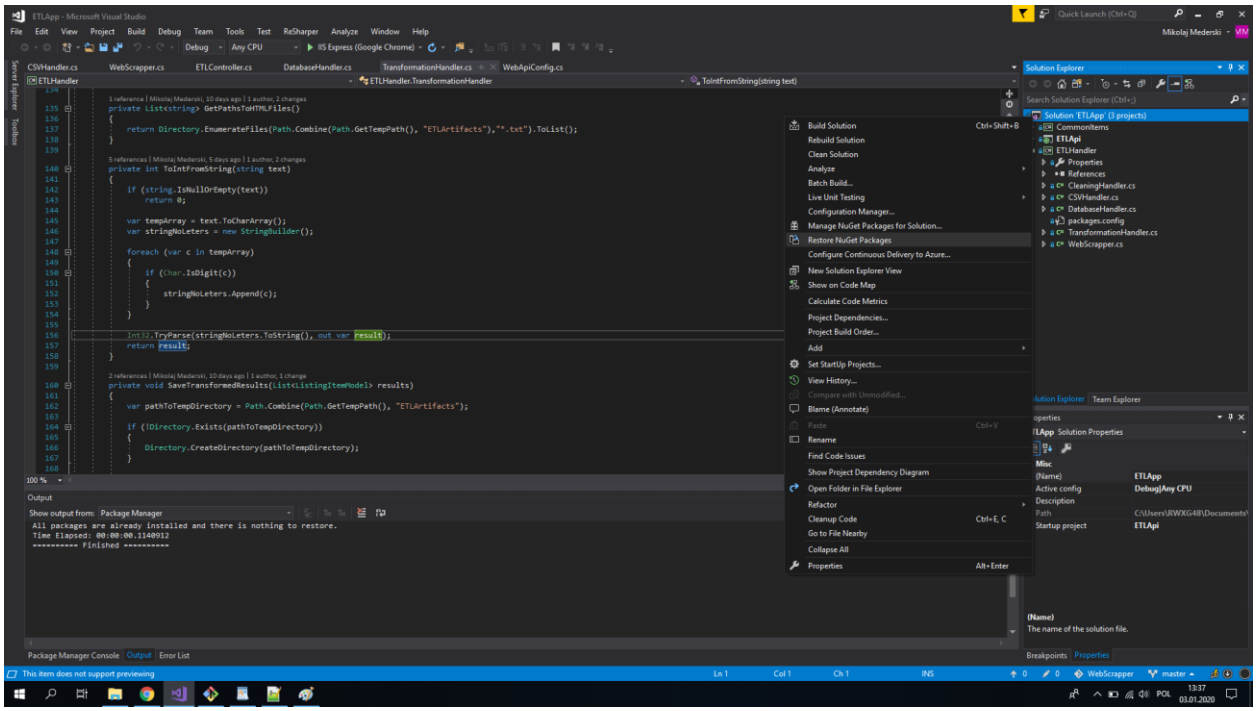
- Pobrać installer Visual Studio wchodząc w link:

  https://visualstudio.microsoft.com/pl/downloads/

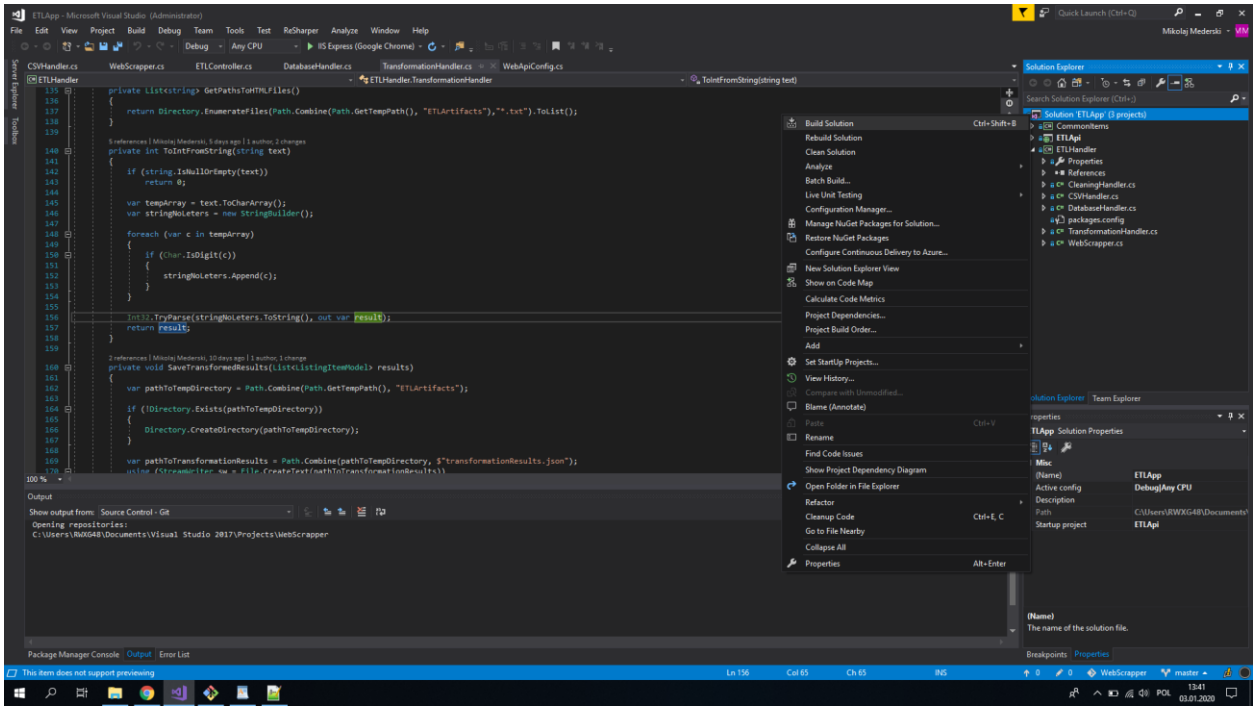
- Zainstalować Visual Studio 2017 wybierając dodatkowo ASP.NET and Web Development w zakładce Workload



- Otworzyć plik "ETLApp.sln" w programie Visual Studio 2017 wybierając kolejno w lewym górnym rogu ekranu File -> Open -> Project/Solution
- W oknie "Solution Explorer" kliknąć prawym przyciskiem myszy "Solution 'ELTApp'" i wybrać z menu opcję "Restore NuGet Packages"



- Ponownie kliknąć prawym przyciskiem myszy "Solution 'ELTApp'" i wybrać opcje Build Solution



Baza danych:

Dokładny opis modułów

Moduł pierwszy - jest on odpowiedzialny za scrappowanie. Aktywowany jest po wysłaniu prośby przez interfejs użytkownika. Aby moduł mógł zadziałać poprawnie, wymagane jest podanie poprawnego adresu URL strony OTODOM, zawierającej wyniki wyszukiwania. Adres ten zapisywany jest jako pole w klasie scrappującej. Następnym krokiem jest wysłanie prośby do serwisu OTODOM, o kod źródłowy strony, do której prowadzi wcześniej wspomniany link URL. Po otrzymaniu odpowiedzi, kod ten przekazywany jest do modułu drugiego (opisanego niżej w tym dokumencie), w celu uzyskania linków URL do pojedynczych ofert, znajdujących się na stronie wyszukiwarki ogłoszeń. Po otrzymaniu listy linków, klasa scrappująca wysyła prośby o kody źródłowe konkretnych ogłoszeń. Próśb tych wysłanych jest tyle, ile ogłoszeń znajdowało się na stronie wyszukiwarki. Ostatnią funkcjonalnością tego modułu jest zapisanie otrzymanych wyników do plików tekstowych, znajdujących się w folderze C:\Users\USER\AppData\Local\Temp (gdzie USER to nazwa użytkownika komputera). Zapis ten odbywa się jedynie wtedy gdy użytkownik wybierze funkcje Extract. Podczas funkcji ETL nie zachodzi potrzeba zapisywania danych do zewnętrznych plików. Dane zapisane w plikach tekstowych mogą następnie zostać odczytane przez moduł drugi, podczas transformacji.

Moduł drugi - odpowiedzialny za transformacje. Posiada on dwie główne funkcjonalności:

- Transformacje danych na format, który pozwala na zapis do bazy danych
- Wyciąganie adresów URL poszczególnych ogłoszeń zawartych na stronie wyszukiwani serwisu OTODOM.

Pierwsza funkcjonalność działa na dwa sposoby, w zależności od ścieżki która wybrał użytkownik. Jeśli wybrana została ścieżka pojedynczych kroków (osobno Extract, osobno Transform i osobno Load) wtedy dane potrzebne do transformacji (kod HTML ogłoszeń) odczytywany jest z plików powstałych podczas działania modułu pierwszego. Jeśli wybrana została ścieżka automatyczna (wszystkie kroki ETL wykonywane są jeden po drugim, bez ingerencji użytkownika) dane do transformacji przekazywane są bezpośrednio z poprzedniego modułu.
Gdy moduł ma już dostęp do danych, kod źródłowy ogłoszeń jest odczytywany przez Html Agility Pack, a następnie wyciągane są z niego części kodu odpowiedzialne za część tytułową ogłoszenia, oraz część posiadającą dane szczegółowe (np. rodzaj ogrzewania, rok budowy budynku). Te dwie sekcje są następnie przekazywane do funkcji odpowiedzialnej za wyciągnięcie konkretnych wartości liczbowych, bądź informacji tekstowych i stworzenie obiektu C#, posiadającego interesujące nas pola.
Kończąc swe działanie, na podstawie ścieżki wybranej przez użytkownika moduł zapisuje dane po transformacji, jeśli użytkownik wybrał ścieżkę pojedynczych kroków. Dane zapisywane są w pliku o formacie „.json”.
Druga funkcjonalność wykorzystywana jest przez moduł pierwszy. Funkcja otrzymuje kod źródłowy strony, na której wylistowane są ogłoszenia. Następnie tworzy listę pojedynczych ogłoszeń i z każdego ogłoszenia wyciąga jego adres URL. Lista adresów zwracana jest do modułu pierwszego.

Moduł trzeci - odpowiedzialny za komunikacje z bazą danych.

Moduł pomocniczy - wykonuje on operacje nie związane z procesem ETL. Zawiera on

- Eksport danych do pliku o formacie „.csv”
- Eksport pojedynczego ogłoszenia do pliku o formacie „.txt”
- Usuwanie plików powstałych podczas procesu ETL.

Tabele klas i atrybutów

Backend

Listing Item Model

Atrybuty:

Nazwa	**Typ**	**Opis**
Title	string	Pole przechowujące tytuł ogłoszenia
Rooms	int	Pole przechowujące ilość pokoi w nieruchomości
Area	string	Pole przechowujące powierzchnie nieruchomości
Price	string	Pole przechowujące cenę nieruchomości
Bond	int	Pole przechowujące kaucję za nieruchomość
BuildingType	string	Pole przechowujące rodzaj nieruchomości (np. Blok)
Windows	string	Pole przechowujące typ okien w nieruchomości (np. plastikowe)
BuiltIn	int	Pole przechowujące rok budowy nieruchomości
HeatingType	string	Pole przechowujące rodzaj ogrzewania (np. miejskie)
Materials	string	Pole przechowujące rodzaj materiałów budulcowych
Floor	int	Pole przechowujące piętro na którym znajduje się nieruchomość
Address	string	Pole przechowujące lokalizacje nieruchomości
FloorsInBuilding	int	Pole przechowujące ilość pięter w budynku

Cleaning Handler

Funkcje Składowe:

Nazwa	**Wartość Zwracana**	**Parametry**	**Opis**
DeleteArtifacts	Void	Brak	Funkcja usuwająca pliki powstałe podczas procesu ETL

CSVHandler

Funkcje Składowe:

Nazwa	**Wartość Zwracana**	**Parametry**	**Opis**
ExportToCSV	Void	List<ListingItemModel> - Lista ofert	Tworzy plik ".csv" z danymi, które zostały przekazane w parametrze
ExportSingleToTxt	Void	ListingItemModel - pojedyncza oferta	Tworzy plik ".txt" z danymi oferty przekazanej w parametrze
ValidateOfferTitle	string	string title	Zamienia znaki typu "," na znak "_", w celu stworzenia nazwy pliku na podstawie tytułu oferty
GetPathToFile	string	string fileName	Zwraca ścieżkę, w której powstanie plik z wyeksportowanymi danymi

DatabaseHandler

Funkcje składowe:

Nazwa	**Wartość Zwracana**	**Parametry**	**Opis**

TransformationHandler

Funkcje składowe:

Nazwa	**Wartość Zwracana**	**Parametry**	**Opis**
GetListOfProducts	List<ListingItemModel>	bool needSave, List<string> rawHtmlList	Zwraca listę ofert pod postacią obiektów ListingItemModel
GetAdsUrls	List<string>	string rawHtml	Zwraca wszystkie URLe ogłoszeń zawartych w HTMLu przekazanym w parametrze
GetOfferDetails	ListingItemModel	HtmlNode details HtmlNode title	Tworzy obiekt ListingItemModel przypisując wartości

			pól na podstawie sekcji HTML przekazanych w parametrze
GetPathsToHTMLFiles	List<string>	Brak	Zwraca lokalizacje plików zawierających kod HTML ofert
ToIntFromString	int	string text	Przerabia string w którym znajdują się cyfry na int ignorując litery. Np “30 zł” -> “30”
SaveTransformedResults	Void	List<ListingItemModel> results	Tworzy plik formatu “.json” zawierający listę ofert w postaci ListingItemModel

WebScrapper

Atrybuty:

Nazwa	**Typ**	**Opis**
_url	readonly string	Zawiera adres witryny OTODOM
_httpClient	HttpClient	Instancja klasy HttpClient, wykorzystywana do komunikacji internetowej
_transformer	TransformationHandler	Instancja klasy TransformationHandler opisanej powyżej

Funkcje składowe:

Nazwa	**Wartość Zwracana**	**Parametry**	**Opis**
KONSTRUKTOR: WebScrapper	Brak	String url	Przypisuje wartosc pola _url, oraz tworzy nowe instancje HttpClient i TransformationHandler
GetRawHtmls	List<string>	bool needSave	Pobiera kod źródłowy znajdujący się pod adresem z pola _url, a nastepnie pobiera kod źródłowy pojedynczych ofert
SaveRawHtmlToText	Void	string result int documentNumber	Tworzy plik o nazwie “rawHtmlX.txt” gdzie X to liczba ze zmiennej documentNumber. Zawartością pliku jest

			ciąg literalny result

ETLController

Funkcje składowe:

Nazwa	**Wartość Zwracana**	**Parametry**	**Opis**
Get	IHttpActionResult	JsonBodyModel model	Wywołuje akcje Extract
Transform	IHttpActionResult	Brak	Wywołuje akcje Transform
Load	IHttpActionResult	Brak	Wywołuje akcje Load
FullETL	IHttpActionResult	JsonBodyModel model	Wywołuje po kolei akcje Extract, Transform, Load
ExportToCSV	IHttpActionResult	List<ListingItemModel> offers	Wywołuje eksport do ".csv" danych przekazanych w parametrze offers
CleanDb	IHttpActionResult	Brak	Wywołuje czyszczenie bazy danych
ExportSingleToTxt	IHttpActionResult	ListingItemModel offer	Wywołuje eksport do ".txt" oferty przekazanej w parametrze offers
GetAllRecords	IHttpActionResult	Brak	Wywołuje metodę wyciągającą całą zawartość bazy danych

JsonBodyModel

Atrybuty:

Nazwa	**Typ**	**Opis**
url_adress	string	Adres witryny OTODOM

Frontend

Funkcje składowe:

Nazwa	**Wartość Zwracana**	**Parametry**	**Opis**
handleSubmit	IHttpActionRequest	Submit Event	Wysyłanie formularza z podanym adresem url, zapobieżenie domyślnej akcji przeglądarki po wysłaniu formularza.

postUrl	function	String url	Wysyła adres url do API w celu uruchomienia procesu Extract.
getAfterTransform	IHttpActionRequest	Brak	Uruchomienie metody odpowiedzialnej za pobieranie danych z domeny otodom.
fetchData	IHttpActionRequest	Brak	Uruchomienie procesu Transform, wyświetlenie statusu zapytania po wysłaniu zapytania.
getAfterLoad	IHttpActionRequest	Brak	Uruchomienie metody Load zwracającej dane otrzymane po scrapowanieu i wyświetlenie ich w przeglądarce.
fullETLProcess	IHttpActionRequest	Brak	Uruchomienie metod odpowiedzialnych za Export, Transform, Load automatycznie jedna po drugiej jeśli poprzenia zakończyła się sukcesem i wyświetlenie rezultatów.
exportToCsv	IHttpActionRequest	Brak	Export tabeli do pliku .csv.
exportSingleElement	IHttpActionRequest	rowData	Export pojedynczego wiersza z tabeli do pliku .txt.
cleadDb	IHttpActionRequest	Brak	Czyszczenie bazy danych

Baza danych