

DemoGraphicVisualization

Dokumentacja techniczna

Spis treści

1. Wymagania systemowe.....	3
2. Opis projektu.....	3
3. Uruchomienie programu.....	4
4. Wykorzystane technologie.....	6
5. Wykorzystane narzędzia.....	7
6. Architektura.....	8
6.1 Aplikacja serwerowa.....	8
6.2 Aplikacja interfejsu użytkownika.....	9
6.3 Schemat przepływu danych.....	10
6.4 Endpointy WebAPI.....	11
6.5 Pobieranie danych w aplikacji interfejsu użytkownika.....	15
7. RestAPI Eurostat.....	17
8. Widoki interfejsu użytkownika.....	18
9. Podsumowanie.....	21

1. Wymagania systemowe

- System Operacyjny Windows 10 w wersji 64-bitowej.
- Zainstalowane środowisko uruchomieniowe Microsoft .NET Core SDK w wersji co najmniej 3.1.1.
- Zainstalowane środowisko uruchomieniowe Node.js w wersji co najmniej 14.15.1.

2. Opis projektu

Celem projektu jest aplikacja webowa przedstawiająca wizualizacje danych statystycznych dotyczących demografii Unii Europejskiej. Dane przedstawione są na różnego rodzaju wykresach oraz grafach. Na poszczególnych podstronach istnieje możliwość filtrowania wartości pod względem narodowości oraz roku kalendarzowego.

Projekt dzieli się na aplikację serwerową zrealizowaną w formie WebAPI utworzonego w technologii .NET Core oraz interfejs użytkownika utworzony przy pomocy Vue.js. Aplikacje te komunikują się za pomocą protokołu HTTP. Aplikacja serwerowa pobiera dane z serwisu Eurostat, wykorzystując udostępnione punkty końcowe REST API.

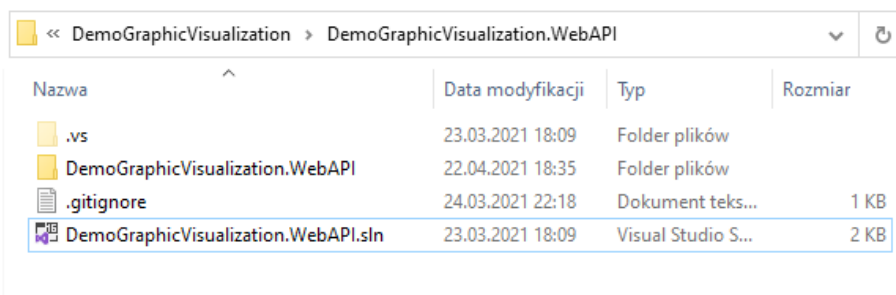
3. Uruchomienie programu

Przed przystąpieniem do uruchamiania projektu należy upewnić się, że poniższe punkty są zrealizowane:

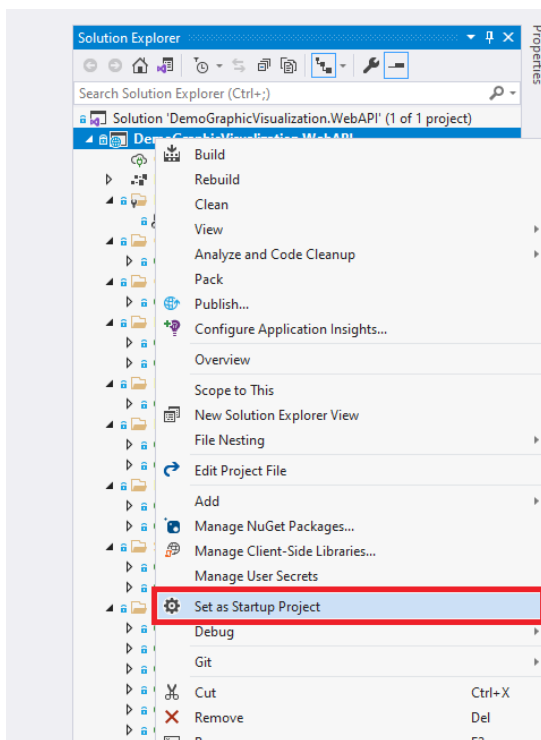
- zainstalowane IDE Microsoft Visual Studio 2019,
- zainstalowany program Powershell,
- zainstalowane środowisko programistyczne Microsoft .NET Core SDK 3.1,
- zainstalowane środowisko programistyczne Node.js.

Proces uruchamiania programu prezentuje się następująco:

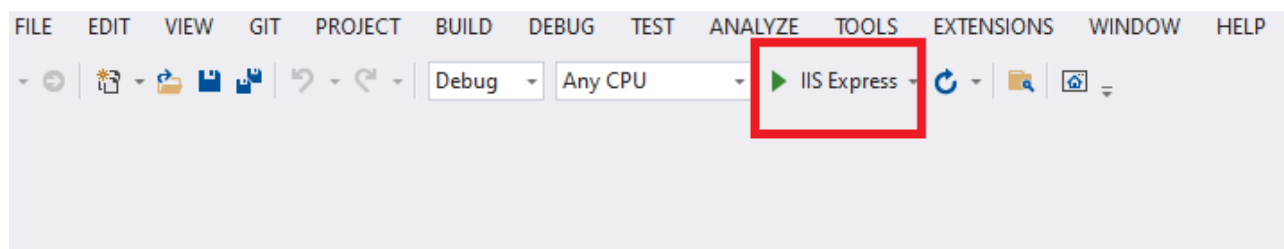
1. Plik `DemoGraphicVisualization.WebAPI.sln` znajdujący się w folderze `DemoGraphicVisualization.WebAPI` należy otworzyć z prawami administratora w programie Microsoft Visual Studio 2019.



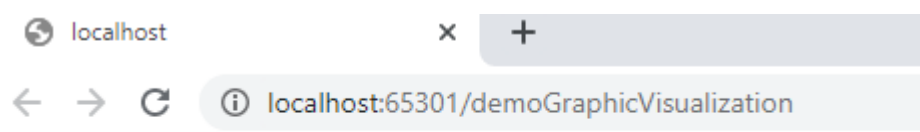
2. W Solution Explorer należy kliknąć prawym przyciskiem myszy na projekt `DemoGraphicVisualization.WebAPI` i wybrać `Set as Startup Project`.



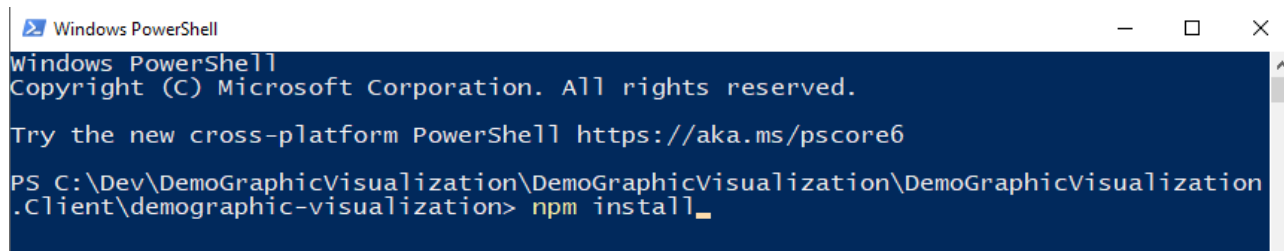
3. Uruchomić WebAPI poprzez wciśnięcie F5 lub kliknięcie zielonej strzałki na górnym pasku narzędzi.



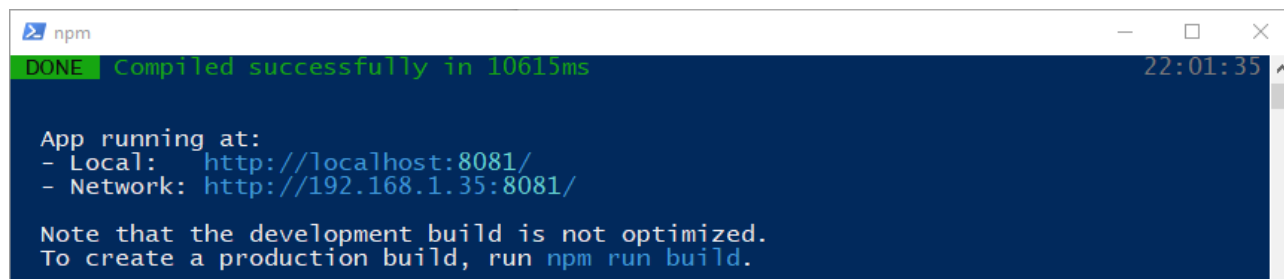
Aplikacja zacznie działać lokalnie na porcie 65301.



4. Uruchomić Powershell i przejść do katalogu DemoGraphicVisualization\DemoGraphicVisualization\DemoGraphicVisualization.Client\demographic-visualization. Następnie wykonać komendę `npm install`, która zainstaluje lokalnie zewnętrzne moduły opisane w pliku `package.json`.



5. Po pomyślnie zakończonej instalacji należy wykonać komendę `npm run serve`. Uruchamia ona lokalnie aplikację interfejsu użytkownika na porcie 8081.



4. Wykorzystane technologie

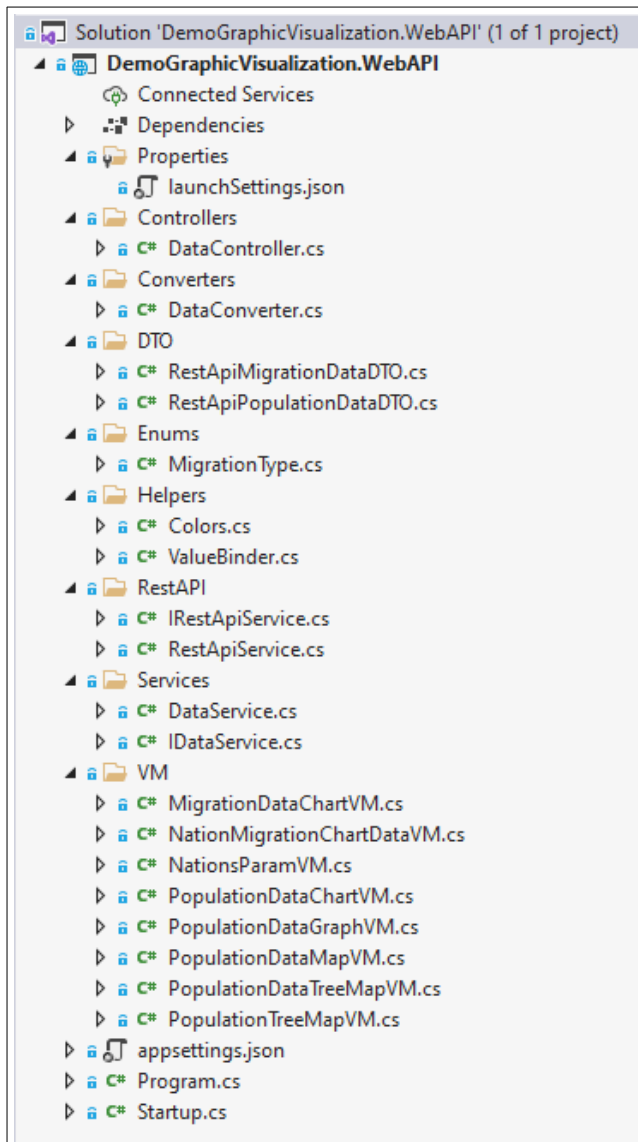
- .Net Core 3.1 – środowisko programistyczne dostarczające możliwości tworzenia różnego rodzaju projektów, m.in.: WebAPI, mikroserwisy, Windows serwisy, aplikacje desktopowe. Używanymi językami programowania są C#, F# oraz Visual Basic. Oprogramowanie utworzone przy użyciu tej technologii są wieloplatformowe.
- LINQ – część technologii Microsoft .NET umożliwiająca wykonywanie zapytań na obiektach, składnią przypominających język SQL.
- Vue.js 2 – framework JavaScriptowy służący do tworzenia webowych aplikacji interfejsu użytkownika.
- Axios – biblioteka JavaScriptowa pozwalająca na komunikację z aplikacją WebAPI za pomocą protokołu HTTP.
- npm – rejestr programistyczny pozwalający na pobieranie oraz publikowanie pakietów kodu. Umożliwia łatwe pobieranie dodatkowych bibliotek JavaScriptowych oraz zarządzanie ich wersjami.
- Node.js – środowisko uruchomieniowe oprogramowania napisanego w języku JavaScript. Pozwala na hostowanie zarówno aplikacji interfejsu użytkownika jak i backendowych części systemów.

5. Wykorzystane narzędzia

- Visual Studio 2019
- Visual Studio Code
- Powershell
- Postman

6. Architektura

6.1 Aplikacja serwerowa



Controllers – zawiera klasę realizującą komunikację HTTP z interfejsem użytkownika.

Converters – metody serializujące dane do postaci wymaganej w kontrolkach UI.

DTO – Data Transfer Objects – klasy, do których serializowane są dane przychodzące z RestAPI.

Enums – typy wyliczeniowe.

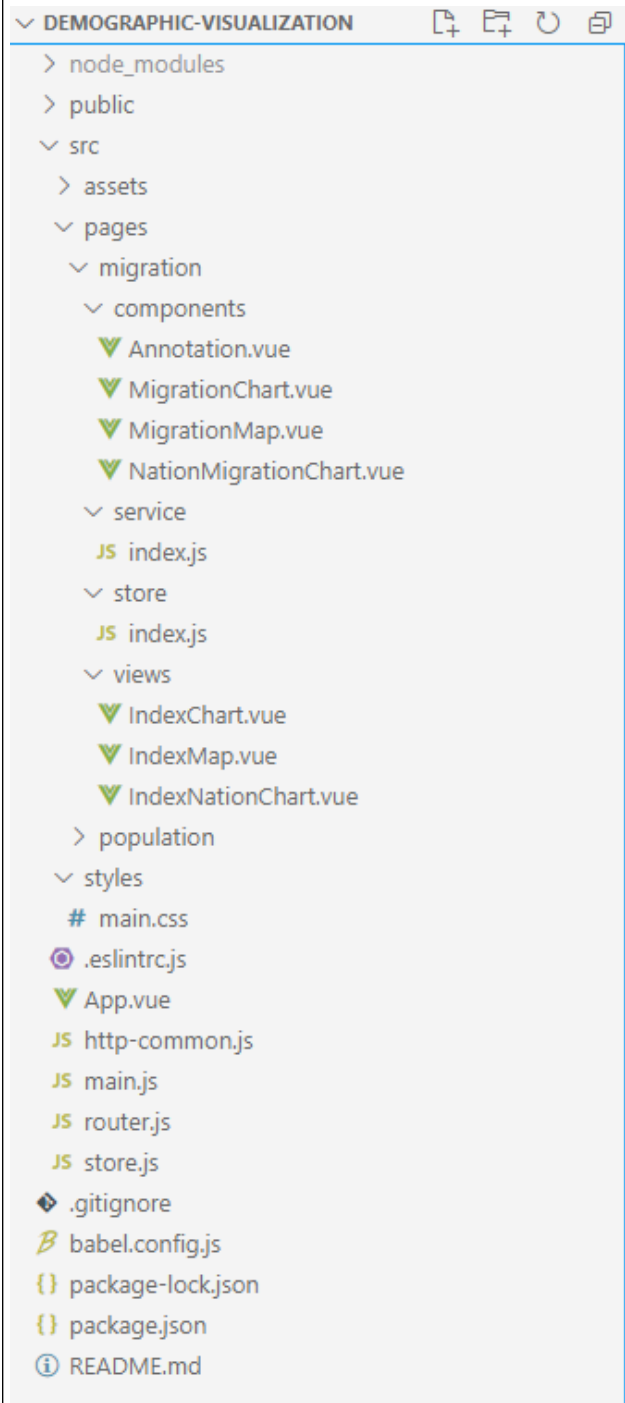
Helpers – pomocnicze metody.

RestAPI – interfejs oraz implementacja serwisu realizującego komunikację z serwerem RestAPI.

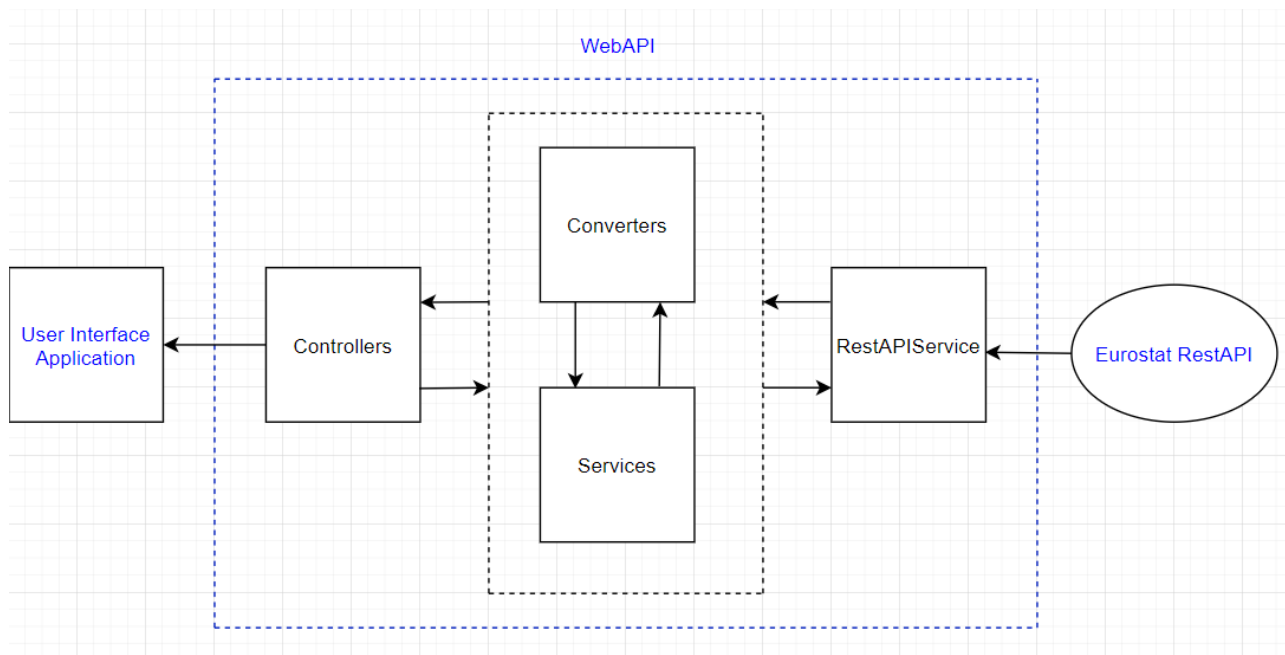
Services – interfejs oraz implementacja serwisu odpowiadającego za warstwę logiki aplikacji.

VM – View Models – klasy, do których serializowane są dane wysyłane do interfejsu użytkownika.

6.2 Aplikacja interfejsu użytkownika

 <p>The screenshot shows a file explorer for a project named 'DEMOGRAPHIC-VISUALIZATION'. The file tree is as follows:</p> <ul style="list-style-type: none">node_modulespublicsrc<ul style="list-style-type: none">assetspages<ul style="list-style-type: none">migration<ul style="list-style-type: none">components<ul style="list-style-type: none">Annotation.vueMigrationChart.vueMigrationMap.vueNationMigrationChart.vueservice<ul style="list-style-type: none">index.jsstore<ul style="list-style-type: none">index.jsviews<ul style="list-style-type: none">IndexChart.vueIndexMap.vueIndexNationChart.vuepopulationstyles<ul style="list-style-type: none">main.css.eslintrc.jsApp.vuehttp-common.jsmain.jsrouter.jsstore.js.gitignorebabel.config.jspackage-lock.jsonpackage.jsonREADME.md	<p>main.js/App.vue - pliki inicjujące pracę aplikacji.</p> <p>http-common.js – konfiguracja połączenia z WebAPI.</p> <p>router.js – plik konfiguracyjny biblioteki Vue Router.</p> <p>store.js – plik konfiguracyjny biblioteki Vuex.</p> <p>styles – pliki CSS.</p> <p>pages – podstrony aplikacji z podziałem na moduły.</p> <p>components – komponenty używane na widokach.</p> <p>service – metody realizujące połączenie z punktami końcowymi WebAPI.</p> <p>store – klasy realizujące funkcjonalność biblioteki Vuex – akcje, mutacje, gettersy do zarządzania danymi.</p> <p>views – widoki – podstrony aplikacji.</p>
--	---

6.3 Schemat przepływu danych



Aplikacja dzieli się na część interfejsu użytkownika oraz części backendowej – Web API. Łączy się ona z Rest API serwisu Eurostat. Przepływ danych wygląda następująco:

1. Otrzymanie żądania danych w kontrolerze WebAPI.
2. Wywołanie metody w serwisie odpowiadającej danemu zapytaniu.
3. Wywołanie metody w serwisie Rest API wysyłającej żądanie do serwisu Eurostat.
4. Przekazanie danych do serwisu, gdzie mapowane są do obiektu.
5. Przekazanie utworzonego obiektu do konwertera, w którym dane są serializowane do struktury wymaganej przez kontrolki graficzne w interfejsie użytkownika.
6. Przefiltrowanie danych, jeśli dane zapytanie to umożliwia.
7. Przesłanie danych do kontrolera i następnie do interfejsu użytkownika.

6.4 Endpointy WebAPI

WebAPI pracuje na porcie 65301. Z aplikacją interfejsu użytkownika komunikuje się za pomocą następujących punktów końcowych:

1. Pobranie danych o populacji w danym kraju według roku do pokazania na wykresie.

URI:	api/data/getPopulationDataToChart
HTTP Method:	GET
Params:	year: string
Return value:	List<PopulationDataChartVM> PopulationDataChartVM: { Nation: string, Population: long, Year: string }

2. Pobranie danych o populacji w danym kraju według roku do pokazania na mapie wektorowej.

URI:	api/data/getPopulationDataToMap
HTTP Method:	GET
Params:	year: string
Return value:	Dictionary<string, long>

3. Pobranie danych o populacji w danym kraju według roku do pokazania na wykresie tree map.

URI:	api/data/getPopulationDataToTreeMap
HTTP Method:	GET
Params:	year: string
Return value:	List<PopulationTreeMapVM> PopulationTreeMapVM: { Name: string, List<PopulationDataTreeMapVM>: { Name: string, Value: long } } }

4. Pobranie danych o populacji w danym kraju do grafu.

URI:	api/data/getPopulationDataToGraph
HTTP Method:	GET
Params:	-
Return value:	<pre>PopulationDataGraphVM PopulationDataGraphVM: { List<GraphNodeVM>: { Id:string, Group:int }, List<GraphLinkVM>: { Source: string, Target: string, Population: long } }</pre>

5. Pobranie danych o migracjach według wybranego roku i krajów.

URI:	api/data/getMigrationDataToChart
HTTP Method:	GET
Params:	<pre>NationsParamVM: { Year:string, SelectedNations:string[] }</pre>
Return value:	<pre>List<MigrationDataChartVM> MigrationDataChartVM: { Nation: string, Immigration: long, Emigration: long }</pre>

6. Pobranie krajów do wyboru w select boxie.

URI:	api/data/getNationsToLookup
HTTP Method:	GET
Params:	-
Return value:	List<ValueBinder<string, string>>

7. Pobranie danych o migracji lub imigracji według wybranego kraju.

URI:	api/data/getNationMigrationDataToChart
HTTP Method:	GET
Params:	nation:string, migrationType:MigrationTypeEnum
Return value:	List<NationMigrationChartDataVM> NationMigrationChartDataVM: { Year: string, Migration: long, Change: double }

8. Pobranie danych o wskaźniku napaści na 100 osób według kraju.

URI:	api/data/getAssaultsDataToChart
HTTP Method:	GET
Params:	nation:string
Return value:	List<AssaultsDataChartVM> AssaultsDataChartVM: { Country: string, Migration: long, Assaults: double }

9. Pobranie danych o średniej ilości migracji w EU według roku.

URI:	api/data/getMigrationAverage
HTTP Method:	GET
Params:	migrationType: int
Return value:	List<MigrationAverageVM> MigrationAverageVM: { Average: double, Year: string }

10. Pobranie danych o przewidywanej ilości lat przeżytych w zdrowiu według kraju.

URI:	api/data/getHealthyLifeDataToChart
HTTP Method:	GET
Params:	nation:string
Return value:	List<HealthyLifeDataChartVM> HealthyLifeDataChartVM: { Country: string, ExpectedLife: double, Year: string }

6.5 Pobieranie danych w aplikacji interfejsu użytkownika

Do połączenia z endpointami WebAPI aplikacja kliencka używa biblioteki Axios. Poniżej zaprezentowana jest zawartość pliku http-common.js będącego konfiguracją połączenia.

```
1import axios from "axios";
2
3export default axios.create({
4  baseURL: "http://localhost:65301/api",
5  headers: {
6    "Content-type": "application/json",
7  },
8});
9
```

Za pomocą metody create() tworzony obiekt pozwalający na połączenia HTTP. Jako parametr przekazany jest obiekt zawierający adres, pod którym dostępny jest serwer, oraz rodzaj przesyłanych danych. Konfiguracja ta wykorzystywana jest do wywoływania konkretnych requestów, których przykłady zaprezentowane są poniżej.

```
1 import axios from "@/http-common";
2 const data = "/data";
3
4 class homeService {
5   getMigrationDataToChart(param) {
6     return axios.post(`${data}/getMigrationDataToChart`, param);
7   }
8   getMigrationDataToMap(year) {
9     return axios.get(`${data}/getMigrationDataToMap/${year}`);
10  }
11  getNationMigrationDataToChart(nation, migrationType) {
12    return axios.get(
13      `${data}/getNationMigrationDataToChart/${nation}/${migrationType}`
14    );
15  }
16  getNationsToLookup() {
17    return axios.get(`${data}/getNationsToLookup`);
18  }
19  getAssaultsDataToChart(nation) {
20    return axios.get(`${data}/getAssaultsDataToChart/${nation}`);
21  }
22  getHealthyLifeDataToChart(nation) {
23    return axios.get(`${data}/getHealthyLifeDataToChart/${nation}`);
24  }
25  getMigrationAveragesData(migrationType) {
26    return axios.get(`${data}/getMigrationAverage/${migrationType}`);
27  }
28}
29
30export default new homeService();
```

Klasa homeService przechowuje metody pozwalające na wysyłanie requestów. Na obiekcie zaimportowanym z pliku http-common.js wywoływane są metody HTTP – GET i POST – wraz

z parametrami zawierającymi informację o ścieżce endpointu oraz ewentualnych argumentach wywołania.

Metody te wywoływane są w akcjach znajdujących się w storach. Otrzymane dane przekazywane są bezpośrednio do widoku UI w postaci Promisa lub zapisywane są w zmiennych znajdujących się w sekcji state.

Poniżej zaprezentowano przepływ danych pomiędzy akcjami, mutacjami oraz sekcją state.

```
1 const actions = {
2   setMigrationChartData: ({ commit, state }) => {
3     service.getMigrationDataToChart(state.param).then((response) => {
4       commit("SET_MIGRATION_CHART_DATA", response.data);
5     });
6   },
7   setMigrationDataToMap: ({ commit }, year) => {
8     return new Promise((resolve, reject) => {
9       service.getMigrationDataToMap(year).then(
10         (response) => {
11           resolve(response);
12         },
13         (error) => {
14           reject(error);
15         }
16       );
17     });
18   },
19 }
```

```
1 const mutations = {
2   updateField,
3   SET_MIGRATION_CHART_DATA(state, payload) {
4     state.migrationChart = payload;
5   },
6   SET_NATION_MIGRATION_CHART_DATA(state, payload) {
7     state.nationMigrationChart = payload;
8   },
9   SET_NATIONS_TO_LOOKUP(state, payload) {
10    state.nationsLookup = payload;
11  },
12  SET_ASSAULTS_CHART_DATA(state, payload) {
13    state.assaults = payload;
14  },
15  SET_HEALTHY_LIFE_CHART_DATA(state, payload) {
16    state.healthyLife = payload;
17  },
18  SET_MIGRAITON_AVERAGES_DATA(state, payload) {
19    state.migrationAverages = payload;
20  },
21};
```

```
1const state = {
2  migrationChart: [],
3  nationsLookup: [],
4  nationMigrationChart: [],
5}
```


7. RestAPI Eurostat

Dane przetwarzane w systemie pochodzą z serwisu internetowego Eurostat (<https://ec.europa.eu/eurostat>). Adresy URI do punktów dostępowych można wygenerować za pomocą Query Buildera dostępnego pod adresem <https://ec.europa.eu/eurostat/web/json-and-unicode-web-services/getting-started/query-builder>. W tym celu należy pobrać identyfikator tabeli, z której zamierzamy pobrać dane oraz uzupełnić konfigurator generowania zapytania. Lista tabel wraz z ich opisem znajduje się pod adresem <https://ec.europa.eu/eurostat/web/main/data/database>.

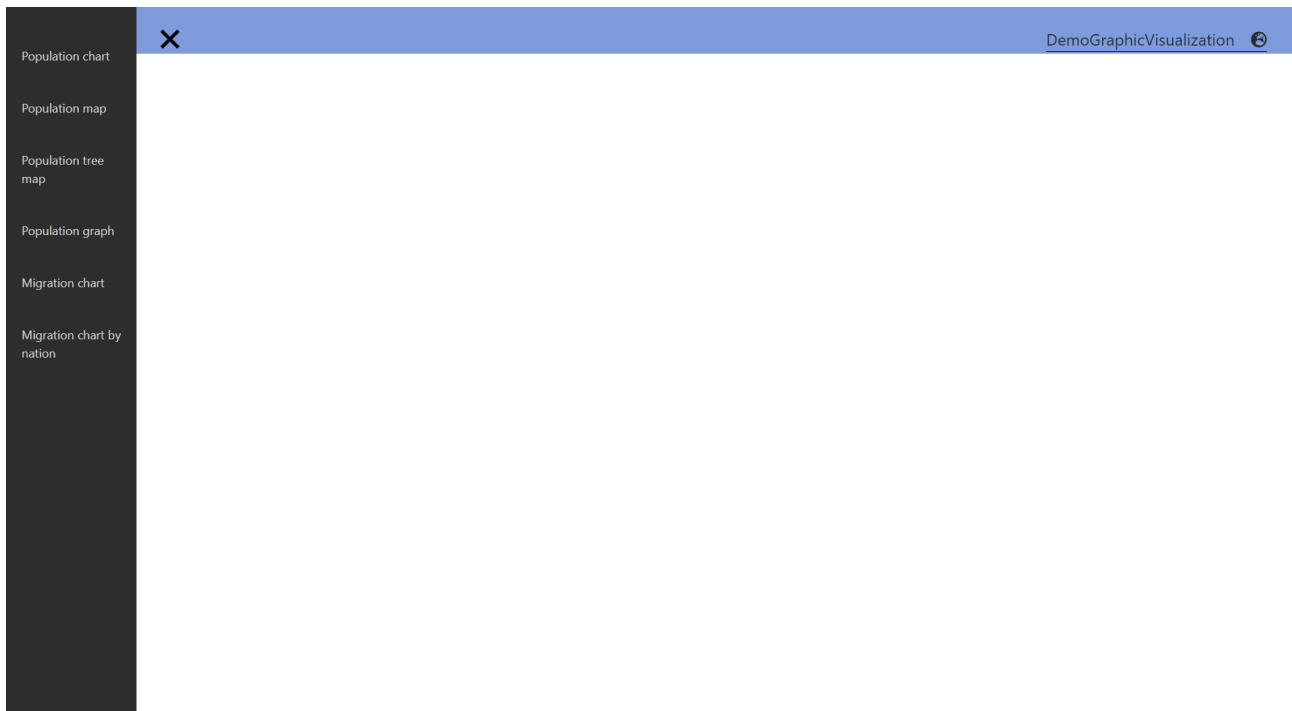
Wygenerowany URI jest wykorzystywany w metodach serwisu RestApiService.

Poniższy kod prezentuje przykładowe wykorzystanie URI i wysłanie żądania danych.

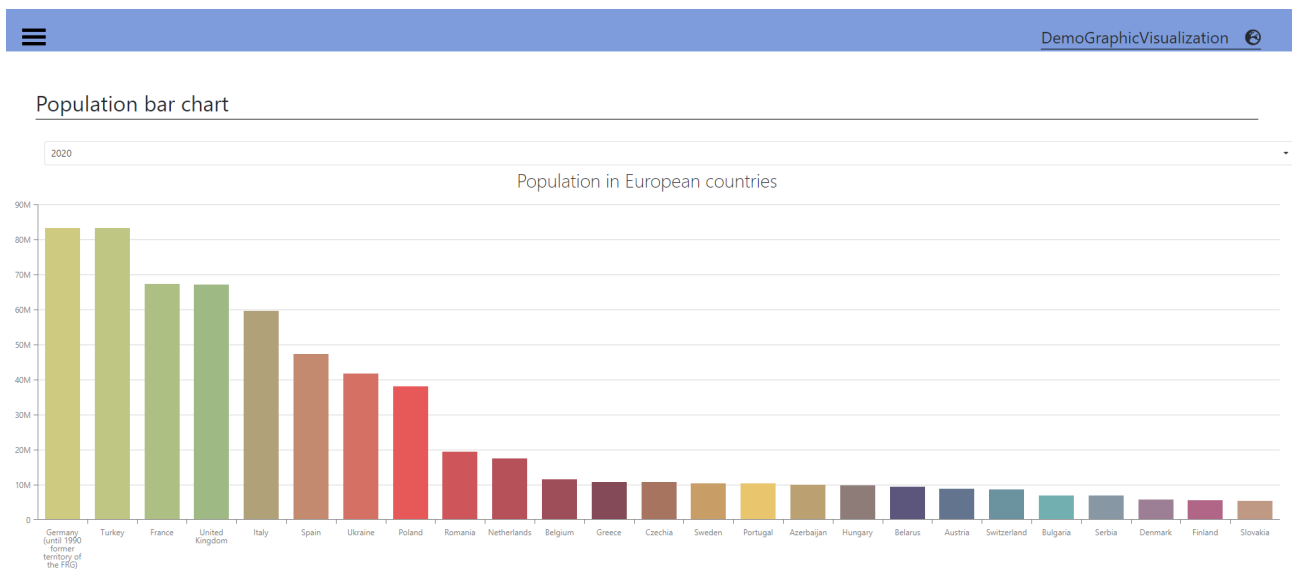
```
1 public RestApiPopulationDataDTO GetPopulationData ()
2 {
3     IRestClient restClient = new RestClient ();
4     IRestRequest restRequest = new RestRequest
5         ("http://ec.europa.eu/eurostat/wdds/rest/data/v2.1/json/en/tps00001?
precision=1");
6     restRequest.AddHeader("Accept", "application/json");
7
8     IRestResponse<RestApiPopulationDataDTO> restResponse =
restClient.Get<RestApiPopulationDataDTO>(restRequest);
9
10    if (restResponse.IsSuccessful)
11    {
12        return restResponse.Data;
13    }
14    else
15    {
16        return null;
17    }
18 }
```

8. Widoki interfejsu użytkownika

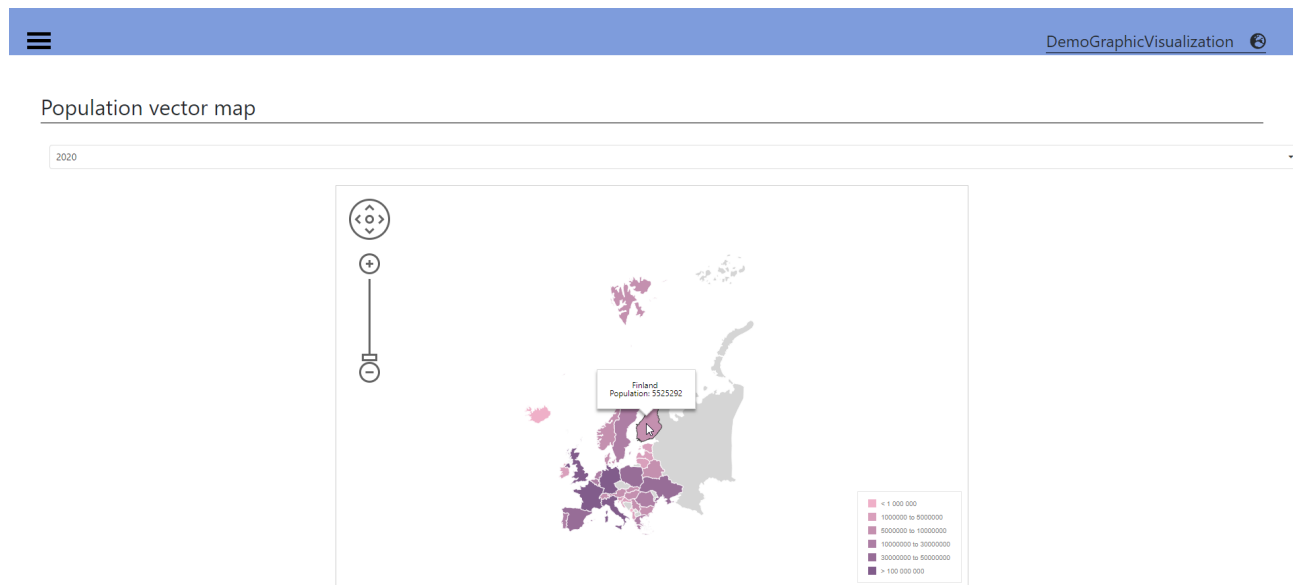
Poszczególne podstrony aplikacji interfejsu użytkownika wybiera się z bocznego menu nawigacyjnego



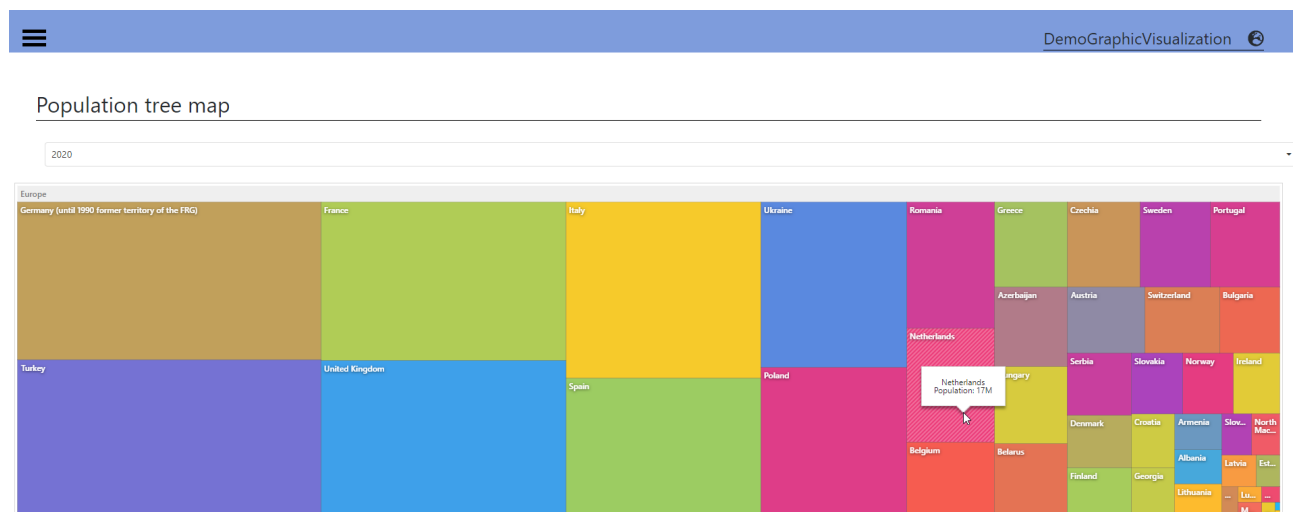
1. Wykres kolumnowy prezentujący populację krajów według wybranego roku.



2. Mapa wektorowa prezentująca populację krajów według wybranego roku.



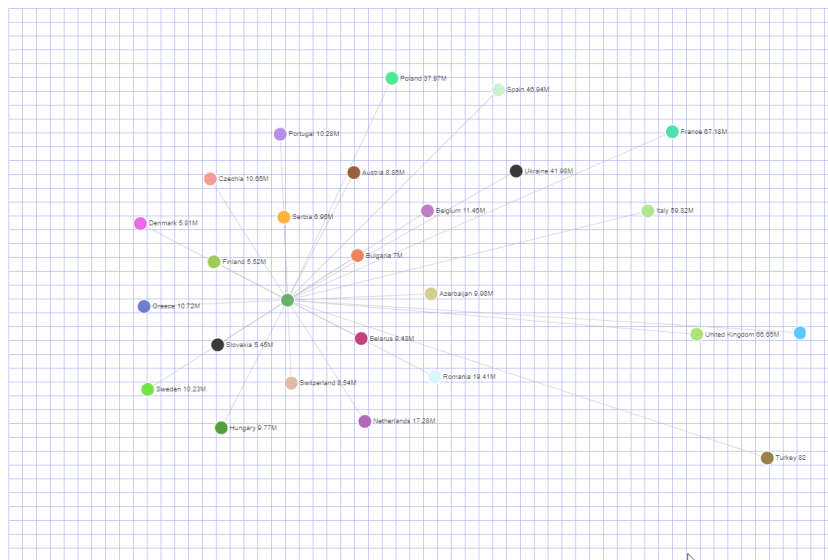
3. Wykres tree map prezentujący populację krajów według wybranego kraju.



4. Graf prezentujący populację 25 krajów w roku 2019. Im dłuższa linia łącząca węzeł przedstawiający dany kraj z środkiem grafu, tym większa populacja.

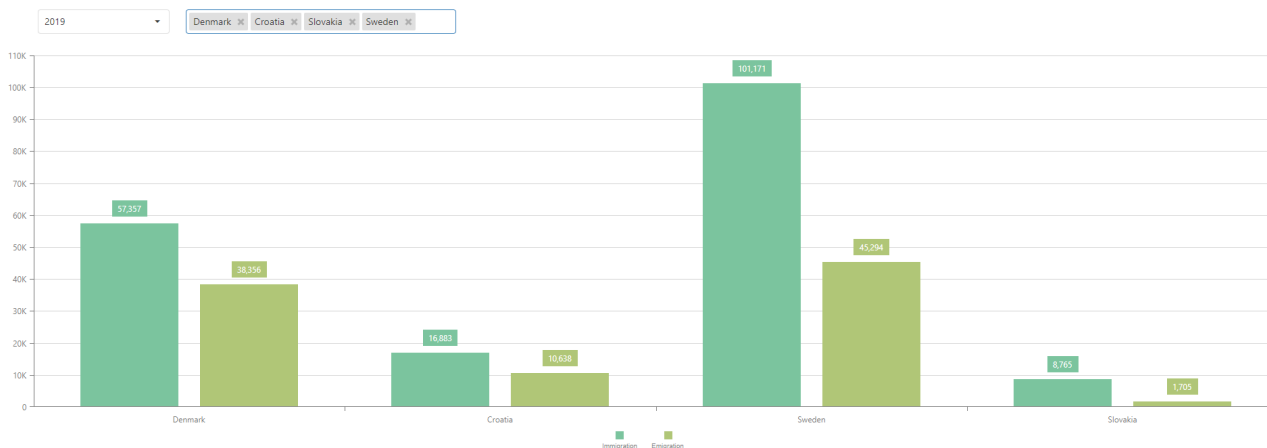
Population force directed graph

Top 25 countries by population (year 2019)



5. Wykres kolumnowy prezentujący porównanie emigracji i imigracji w wybranym roku w wybranych krajach. Z listy rozwijalnej można wybrać konkretny rok. W tag boxie można wybrać kraje, których dane mają być ukazane na wykresie.

Migration bar chart



6. Wykres kolumnowy prezentujący zmiany imigracji/emigracji na przestrzeni lat według wybranego kraju. W listach rozwijalnych można wybrać rok, z którego mają pochodzić dane oraz rodzaj migracji – emigracja lub imigracja. Ponadto na wykresach liniowych przedstawiono wskaźnik ilości napaści na 100 osób oraz przewidywań odnośnie ilości przeżytych w zdrowiu lat.



9. Podsumowanie

Celem projektu było opracowanie aplikacji z warstwą interfejsu użytkownika w postaci aplikacji przeglądarkowej, która pozwala na przeglądanie zwizualizowanych danych demograficznych dotyczących Unii Europejskiej. Dane te pobierane są z RestAPI serwisu Eurostat. W ramach projektu przygotowano wizualizację danych dotyczących populacji, imigracji, emigracji oraz warunków życia państw UE. Dane zaprezentowano w postaci mapy wektorowej, grafu, wykresu typu tree map oraz wykresów słupkowych wraz seriami danych w postaci wykresu liniowego. Po stronie backendu informacje zostały odpowiednio przygotowane do wyświetlenia oraz dokonano obliczeń takich jak średnie, procentowe zmiany trendów, sortowanie, filtrowanie. Cele projektu zostały zrealizowane.