Instytut Informatyki
Kolegium Nauk Przyrodniczych
Uniwersytet Rzeszowski

Przedmiot:

Programowanie portali biznesowych

Dokumentacja projektu:

„Prosta aplikacja CRUD – uczelnie wyższe”

Prowadzący:
dr inż. Piotr Lasek

Wykonał:
Dominik Puchała

Rzeszów 2020

Spis treści

1. Opis projektu

Program to aplikacja webowa, która ma za zadania zarządzać ogólnodostępnymi danymi o uczelniach wyższych na całym świecie. Poszczególne uczelnie są określona poprzez zbiór danych i wyświetlane za pomocą listy. Użytkownik korzystający z aplikacji może dodać, usunąć oraz zmienić informację o danej uczelni za pomocą interfejsu użytkownika.

2. Zakres projektu

Stworzona została podstawowa wersja aplikacji. Wykonuję ona podstawowe założenia aplikacji CRUD. Zapisuję ona dane tylko w trakcie uruchomienia aplikacji, po ponownym uruchomieniu dane zostaną zresetowane i przywrócone do domyślnych. Powstał interfejs graficzny aplikacji, który ułatwia korzystanie z niej oraz testowanie programu. Dodatkowo powstał serwer, który umożliwia uruchomienie aplikacji.

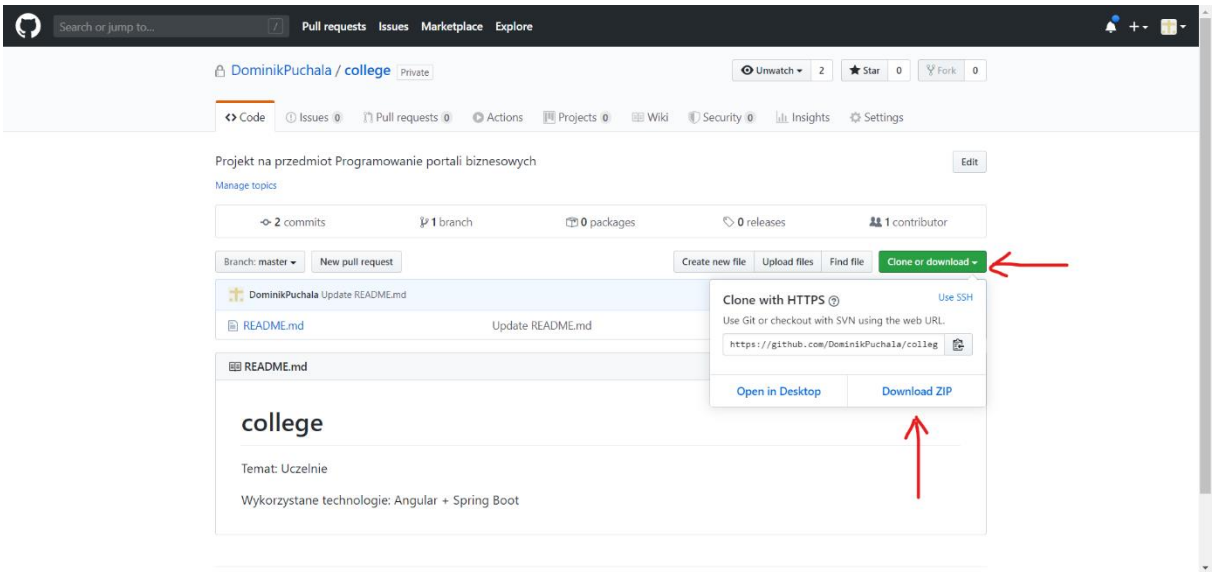
3. Wykorzystane technologie

Do stworzenie aplikacji zostały wykorzystane następujące technologie:

- Angular 9,
- Angular material,
- Maven,
- Spring Boot,
- Node.js,
- Express.

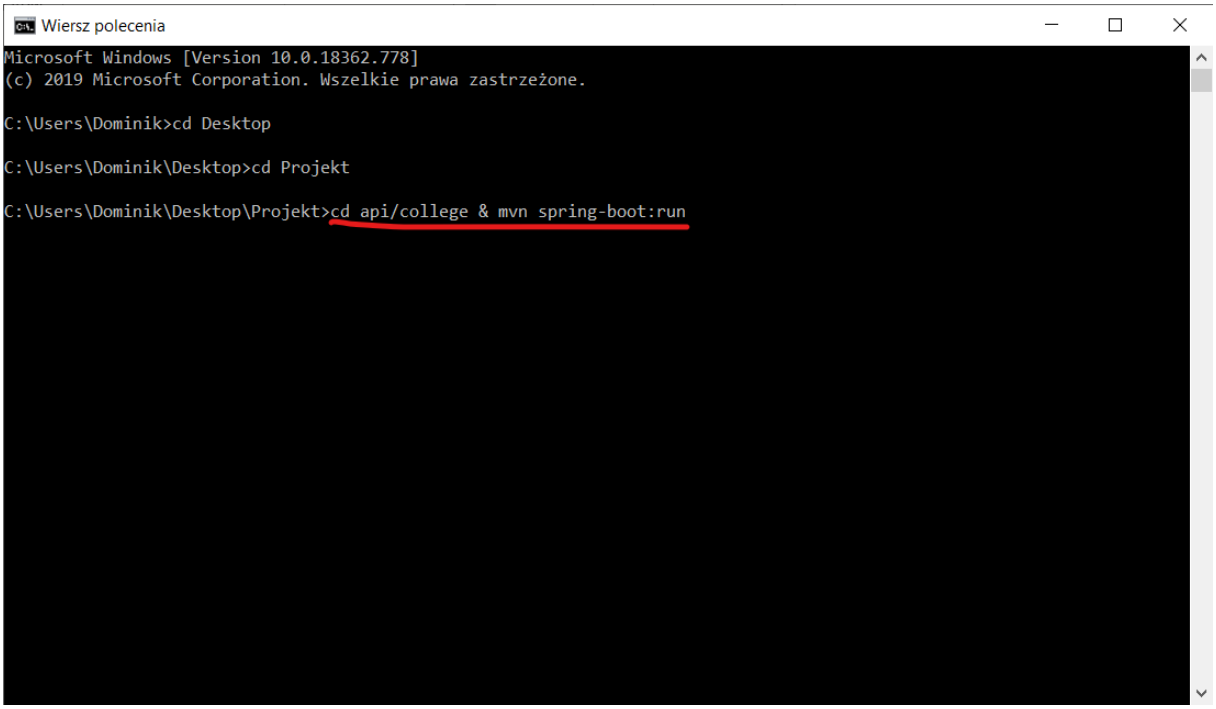
4. Przygotowanie środowiska

Projekt został zamieszczony w prywatnym repozytorium portalu „GitHub”. Aby uruchomić aplikację należy pobrać wszystkie pliki i rozpakować w dogodnym miejscu.

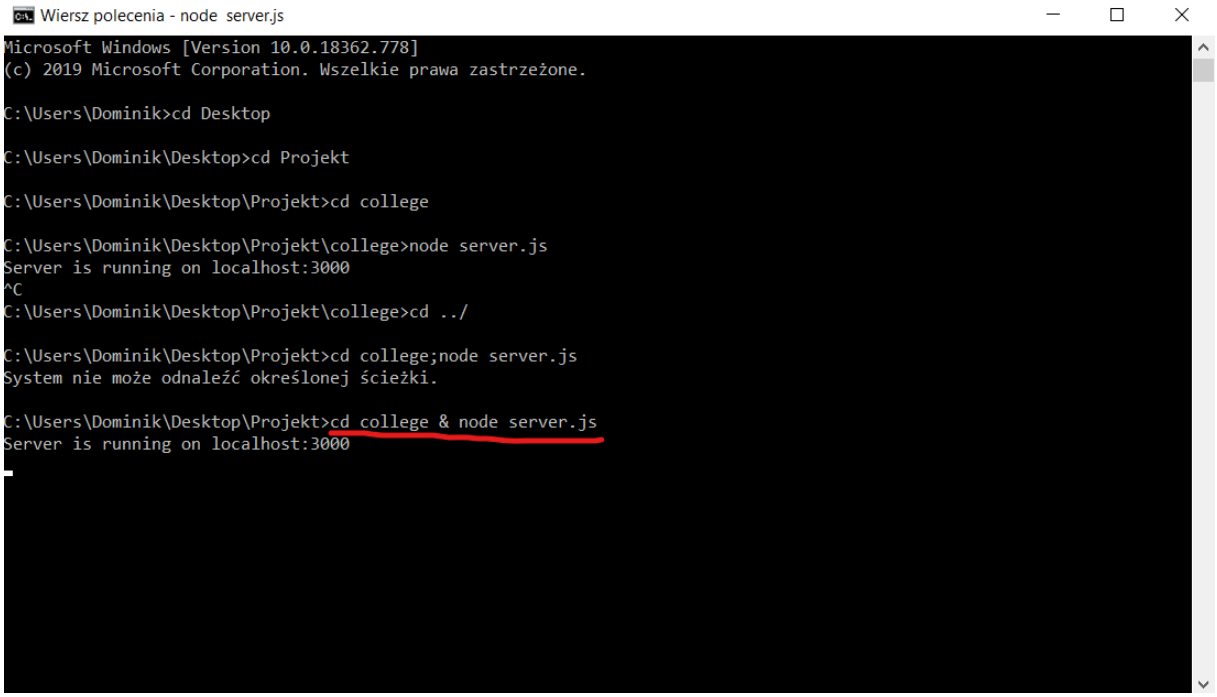


4.1 Windows

Po rozpakowaniu należy otworzyć dwie konsole w folderze z aplikacją. W wierszu poleceń wpisujemy „cd api/college”, a następnie „mvn spring-boot:run”. Aby druga komenda zadziałała, należy mieć zainstalowane narzędzie Maven (https://maven.apache.org/download.cgi).



W drugiej konsoli wpisujemy „cd college”, a następnie „node server.js”. Należy pamiętać również, że trzeba mieć zainstalowane narzędzie Node (https://nodejs.org/en/download/). Jeśli wszystko się udało powinniśmy zobaczyć komunikat „Server is running on localhost:3000”.

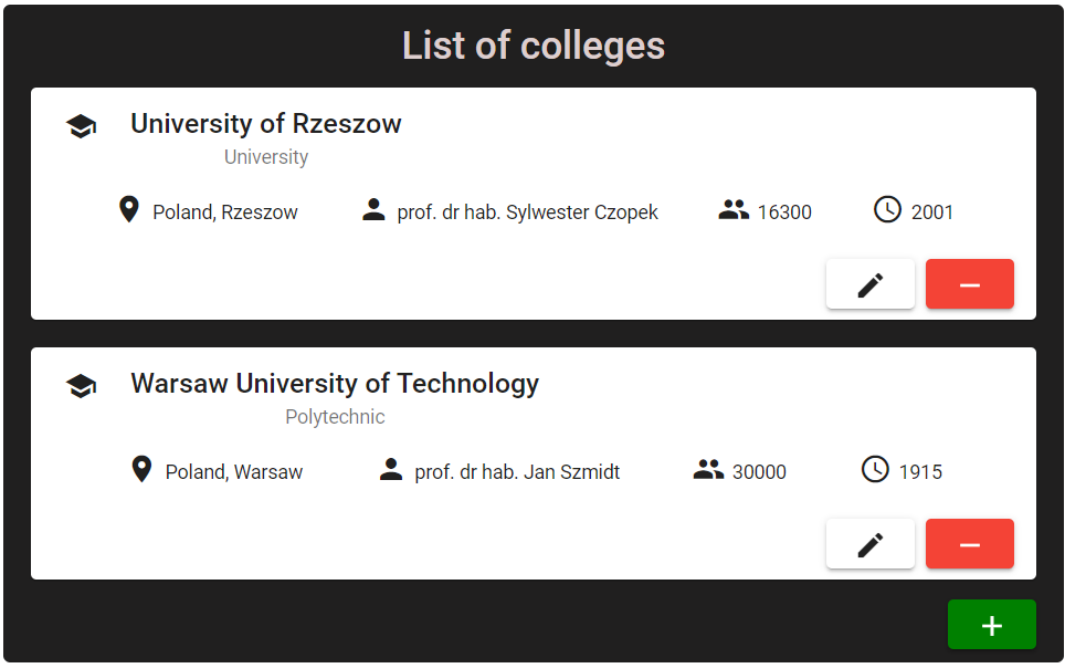


Teraz wystarczy otworzyć przeglądarkę i wpisać „localhost:3000”.

5. Interfejs aplikacji

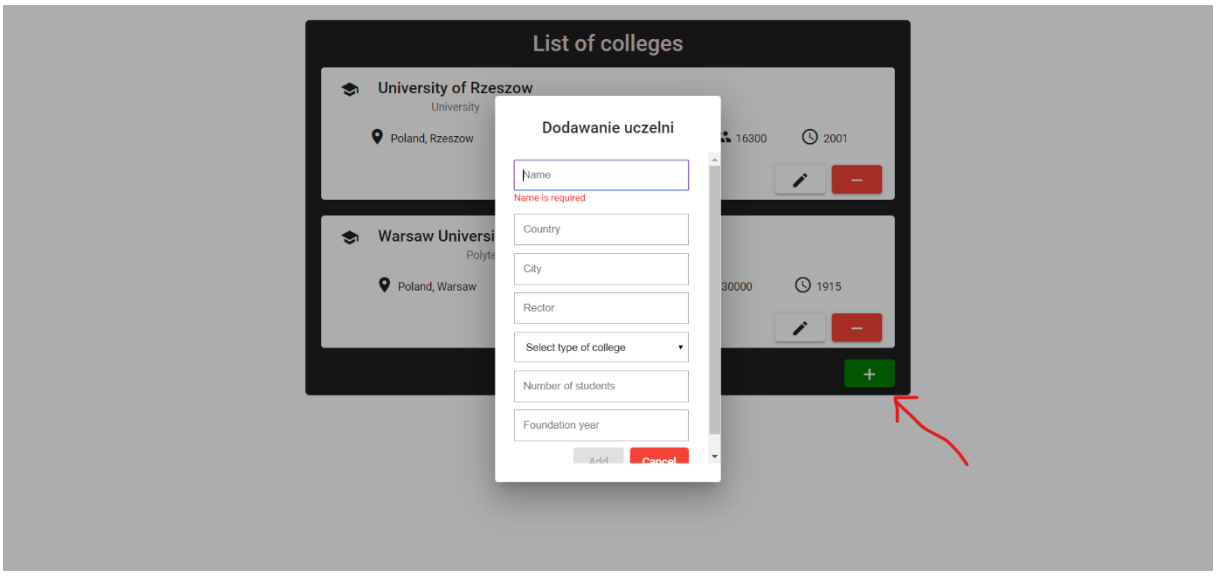
W aplikacji ustawione zostały dwie uczelnie początkowe, które pokazują się za każdym razem gdy uruchomiony zostanie program. Dane każdej uczelni są wypisywane w karcie, gdzie na górze widnieje nazwa uczelni oraz jej rodzaj. Niżej od lewej znajduję się: miejsce położenia, rektor uczelni, liczba studentów, rok powstania uczelni.

5.1 Strona główna



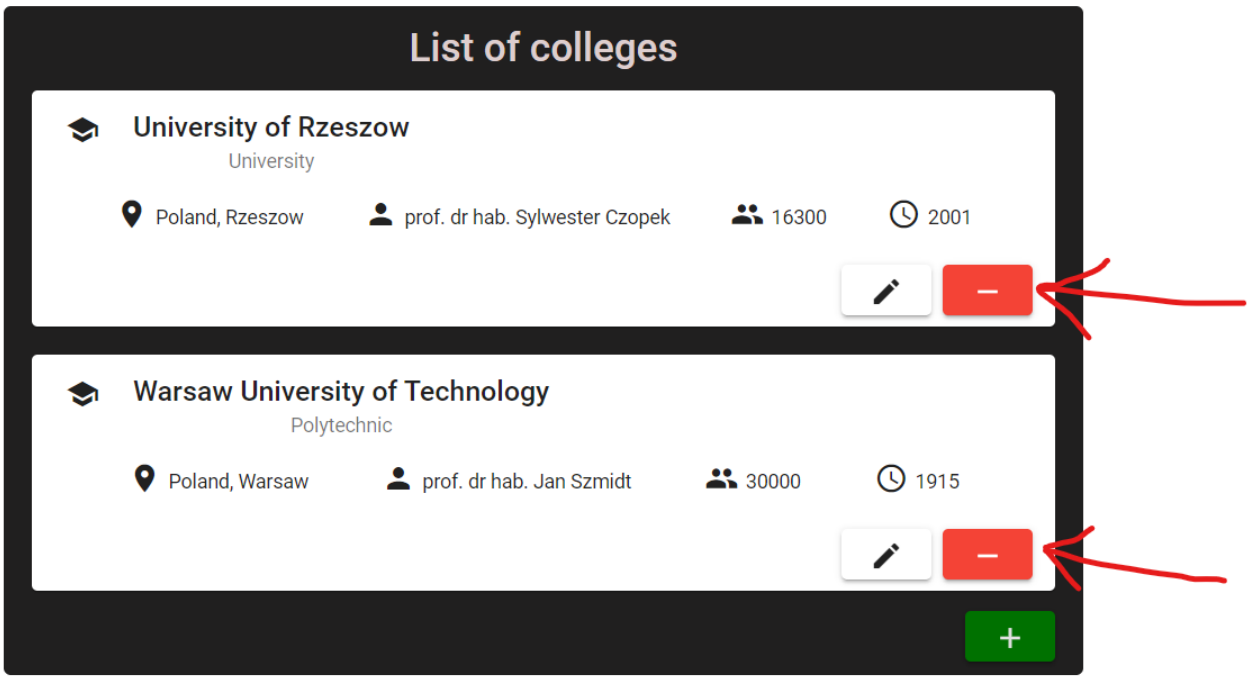
5.2 Dodawanie nowej uczelni

Aby dodać nową uczelnie należy kliknąć przycisk „+” znajdujący się w prawym dolnym rogu ekranu, a następnie wypełnić wszystkie pola zgodnie z opisanymi wymaganiami. Na koniec klikamy przycisk „Add”.



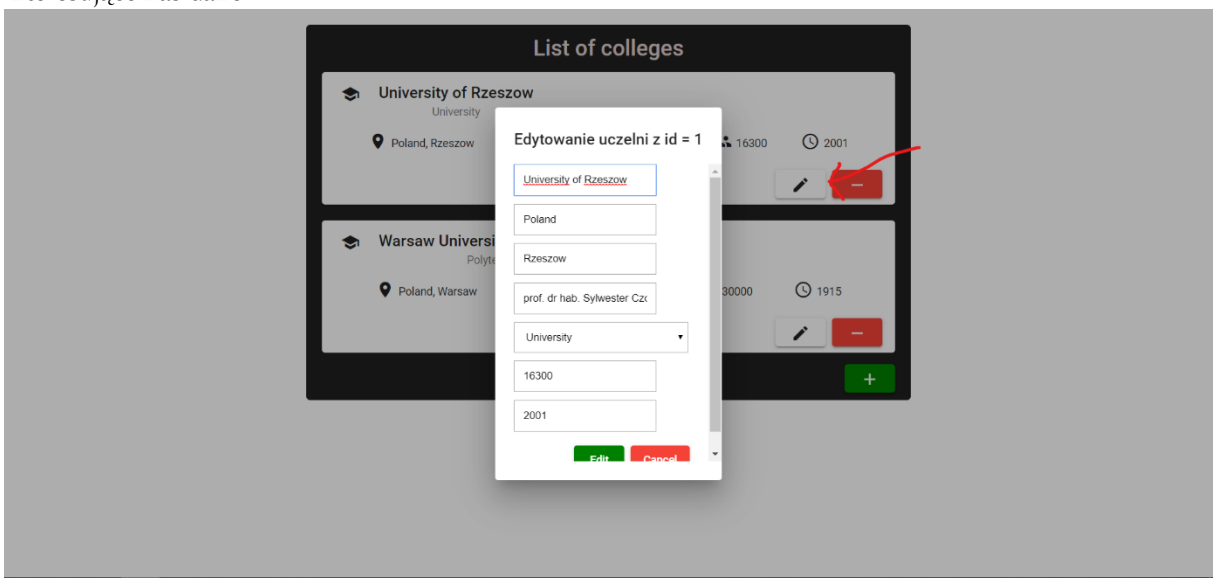
5.3 Usuwanie uczelni

Aby usunąć uczelnie z listy wystarczy kliknąć przycisk „-” znajdujący się w prawym dolnym rogu każdej karty.



5.4 Edytowanie uczelni

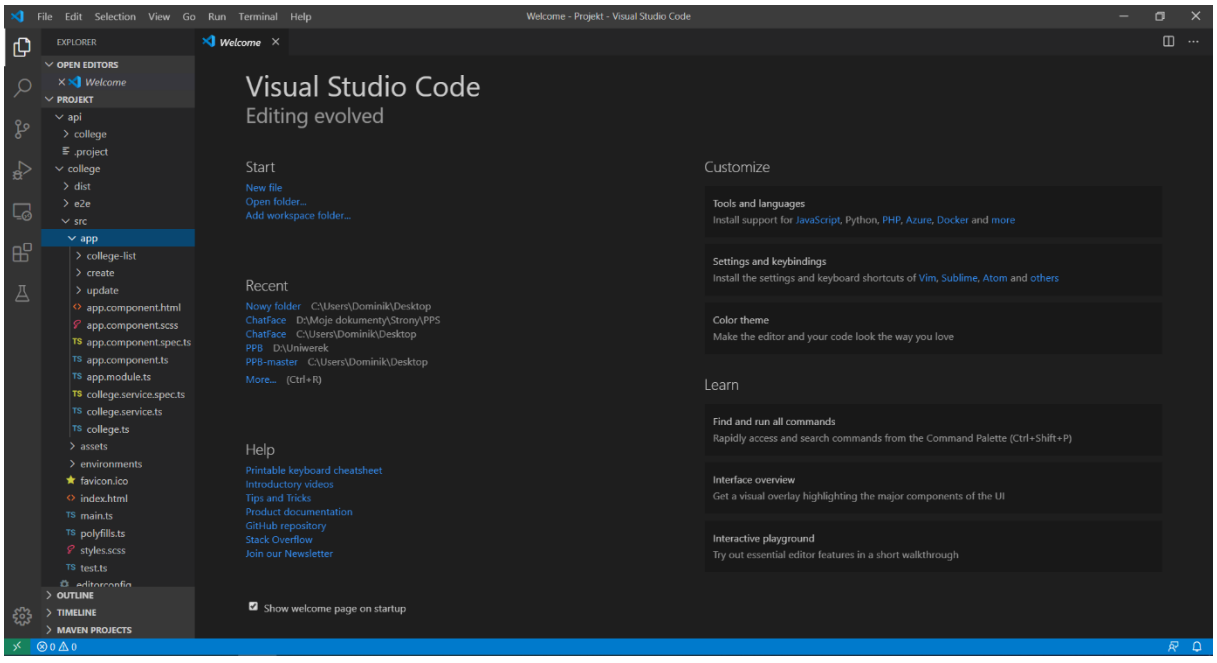
Aby edytować uczelnie należy kliknąć przycisk z obrazkiem ołówka, a następnie poprawić interesujące nas dane.



6. Kod aplikacji

6.1 Struktura plików

W projekcie znajdują się dwa główne foldery: „API” oraz „college”. Folder „API” zawiera część związaną z API oraz backendem aplikacji napisanym w Spring Boot. Drugi folder zawiera front aplikacji napisany przy pomocy Angulara. Znajduję się w nim jeden komponent rodzic „app” oraz trzy komponenty dzieci „college-list” w którym jest wyświetlana lista uczelni, „create”, czyli dialog, który służy do dodawania nowych uczelni oraz „update”, który jest również dialogiem, służącym natomiast do uaktualniania danych w istniejących już uczelniach. Oprócz tego do ważniejszych należy również plik „server.js”, który służy do tworzenia serwera za pomocą narzędzia Express oraz „college.service.ts”, gdzie została zaimplementowana logika aplikacji po stronie frontendu.



6.2 Serwis

W serwisie o nazwie „college.service.ts" umieszczona została cała logika frontendu. Znajduję się w niej sześć funkcji służących do: pobierania pojedynczego obiektu, pobierania wszystkich obiektów, dodawanie nowego obiektu, usuwanie obiektu, edycja obiektu oraz funkcja zwracająca listę dostępnych rodzajów uczelni wyższych.

```
33    getCollege(id: number) {
34      return this.http.get<College>(this.url + `/${id}`);
35    }
36
37    getColleges(): Observable<College[]> {
38      return this.http.get<College[]>(this.url);
39    }
40
41    addCollege(college: College) {
42      return this.http.post<College>(this.url, college);
43    }
44
45    deleteCollege(id: number) {
46      return this.http.delete<College>(this.url + `/delete/${id}`);
47    }
48
49    editCollege(college: College) {
50      return this.http.put<College>(this.url + `/put/${college.id}`, college);
51    }
52
53    getTypesOfColleges() {
54      return this.typesOfCollege;
55    }
56  }
57
```

6.3 Java Controller

API aplikacji zostało napisane w języku Spring Boot. Zostały zaimplementowane dwa obiekty początkowe reprezentujące uczelnie wyższe, tak aby można było je testować zaraz po uruchomieniu aplikacji

```
27      @PostConstruct
28      private void set() {
29          final College c1 = new College();
30          c1.setName("University of Rzeszow");
31          c1.setCountry("Poland");
32          c1.setCity("Rzeszow");
33          c1.setFoundationYear(2001);
34          c1.setNumberOfStudents(16300);
35          c1.setRector("prof. dr hab. Sylwester Czopek");
36          c1.setType("University");
37          this.cRepository.save(c1);
38
39          final College c2 = new College();
40          c2.setName("Warsaw University of Technology");
41          c2.setCountry("Poland");
42          c2.setCity("Warsaw");
43          c2.setFoundationYear(1915);
44          c2.setNumberOfStudents(30000);
45          c2.setRector("prof. dr hab. Jan Szmidt");
46          c2.setType("Polytechnic");
47          this.cRepository.save(c2);
48      }
```

W kontrolerze powstały również endpointy, które służą do łączenia się z backendem aplikacji.

```
50      @GetMapping
51      private ArrayList<College> getAllItems() {
52          return (ArrayList<College>) this.cRepository.findAll();
53      }
54      @GetMapping("/{id}")
55      private Optional<College> getItem(@PathVariable Long id) {
56          return this.cRepository.findById(id);
57      }
58      @PostMapping
59      private College addItem(@RequestBody College college) {
60          return this.cRepository.save(college);
61      }
62      @PutMapping("/put/{id}")
63      private College replaceEmployee(@RequestBody College newCollege, @PathVariable Long id) {
64
65        return cRepository.findById(id)
66          .map(college -> {
67            college.setName(newCollege.getName());
68            college.setCountry(newCollege.getCountry());
69            college.setCity(newCollege.getCity());
70            college.setRector(newCollege.getRector());
71            college.setType(newCollege.getType());
72            college.setNumberOfStudents(newCollege.getNumberOfStudents());
73            college.setFoundationYear(newCollege.getFoundationYear());
74            return cRepository.save(college);
75          })
76          .orElseGet(() -> {
77              newCollege.setId(id);
78              return cRepository.save(newCollege);
79          });
80      }
81      @DeleteMapping("/delete/{id}")
82      private void deleteItem(@PathVariable long id) {
83          this.cRepository.deleteById(id);
84      }
85
86  }
```

6.4 Klasa College

API aplikacji zawiera klasę, która służy do reprezentowania obiektu uczelni wyższej. Zawiera ona pola prywatne oraz gettery i setery do każdego pola.

```
9   public class College {
10      @Id
11      @GeneratedValue
12
13      private long id;
14      private String name;
15      private String country;
16      private String city;
17      private String rector;
18      private String type;
19      private int numberOfStudents;
20      private int foundationYear;
21
22      public long getId() {
23          return id;
24      }
25
26      public String getName() {
27          return name;
28      }
29
30      public String getCountry() {
31          return country;
32      }
33
34      public String getCity() {
35          return city;
36      }
37
```

7. Możliwości rozwoju aplikacji

Obecnie zaimplementowana została podstawowa wersja aplikacji. Jest jeszcze wiele możliwości rozwoju dla niej do których możemy zaliczyć między innymi: utworzenie bazy danych i połączenie się z nią przez API, poprawa interfejsu użytkownika, możliwość grupowania uczelni, możliwość wyszukiwania dodanych szkół wyższych oraz wiele innych. Aktualizację te są planowane w następnych wersjach oprogramowania.