# Projekt na zaliczenie z przedmiotu Systemy informacji przestrzennych GIS

## Raport końcowy

## Skład osobowy

Grzegorz Gojska 174173  
Jakub Konkel 187207

## Wstęp

Projekt obejmował stworzenie serwisu, który pokazuje na mapie stacje benzynowych różnych sieci, z funkcjonalnością m.in. przeglądania stacji na mapie, wyszukiwania stacji według różnych kryteriów, dodawanie ocen i/lub komentarzy do stacji.

### Zaimplementowane przypadki użycia wraz z krótkim opisem

Zrealizowano następujące przypadki użycia:

- Wyznaczanie trasy między dwoma punktami i liczenie jej kosztu na podstawie spalania samochodu.

### Techniczna realizacja projektu

Na projekt składają się: strona www (front-end) oraz back-end do przetwarzania i odpowiedzi na zapytania ze strony www. Front-end został zrealizowany z wykorzystaniem HTML, CSS i JavaScript, bez żadnych dodatkowych bibliotek, z wyjątkiem biblioteki JavaScript i CSS OpenLayers. Back-end został zrealizowany w Pythonie z użyciem frameworka webowego Flask. Jako bazę danych wykorzystano Sqlite, a do komunikacji z bazą użyto frameworka ORM SqlAlchemy.

Back-end zrealizowano według wzorca architektonicznego Model-View-Controller, ale z nieco odmienną terminologią, przyjętą we frameworku Flask - Model-View-Template:

* **model** - model to klasa reprezentująca pewien byt w domenie biznesowej. W tym projekcie domeną są stacje benzynowe, a modelami: użytkownik, stacja benzynowa, paliwo z ceną, komentarz z oceną. Jest to główna jednostka informacji w projekcie. Odpowiada modelowi z MVC.
* **view** - tutaj trafiają żądania użytkownika. Widok odpowiada za przetwarzanie i odpowiedź na żądania użytkownika. W toku przetwarzania pobiera model i przekazuje pobrany model do szablonu (template), żeby wyrenderować odpowiedź - jeżeli odpowiedź powinna zostać zwrócona jako strona HTML. Niekiedy widoki używają również formularzy - są to klasy służące do opisania formularzy wykorzystwanych na stronie do wprowadzenia danych. Odpowiadają za logikę formularza - m.in. za jego walidację. Formularze przekazane do szablonu renderowane są jako formularze HTML. W tym projekcie zaimplementowano również niewielkie REST API, które zwraca odpowiedź w formacie JSON, w takim przypadku szablon nie jest renderowany. Widok jest odpowiednikiem kontrolera z MVC.
* **template** - szablon strony HTML, ta część wzorca jest odpowiedzialna za renderowanie stron WWW z wykorzystaniem szablonów oraz przekazanych przez kontroler danych, i zwrócenie wyrenderowanego szablonu. Jest to odpowiednik widoku z MVC.

### Użyte rozwiązania geoprzestrzenne

Poniżej opisano użyte rozwiązania geoprzestrzenne oraz funkcje biblioteki OpenLayers, wykorzystane od ich implementacji:

- Routing do wyznaczania najkrótszej trasy między dwoma punktami

### Realizacja wymagań

W projekcie postawiono i zrealizowano następujące wymagania:

* strona www wyświetla się poprawnie na dowolnej, nowoczesnej przeglądarce internetowej na komputerze osobistym (dostosowanie wyglądu do obsługi urządzeń przenośnych nie jest przewidziane)
* strona www zgodna ze standardami HTML/XHTML, CSS konsorcjum W3
* back-end stworzony w oparciu o wzorzec MVC
* hasła użytkowników przechowywane w bazie danych w bezpieczny sposób (hash + salt)
* baza danych zabezpieczona przed atakami SQL Injection
* zróżnicowanie widoku graficznego markera na mapie w zależności od sieci, do której należy dana stacja benzynowa

Jedno postawione wymaganie nie zostało zrealizowane:

* bezpieczne uwierzytelnianie użytkownika (oparte na tokenie)

Zamiast uwierzytelniania opartego na tokenie zaimplementowano uwierzytelnianie oparte na sesji: po udanym zalogowaniu klient otrzymuje informacje o ID swojej sesji, wygenerowane przez back-end, i zapisuje je do pliku cookie. W następnych żądaniach do strony klient posługuje się plikiem cookie w celu uwierzytelnienia, a back-end używa tych informacji do uzyskania informacji o kliencie. Co do uwierzytelniania opartego na tokienie, to jest ono szczególnie użyteczne w przypadku REST API, które nie obsługuje plików cookie, a informacje uwierzytelniające muszą być przesyłane w każdym żądaniu, o ile dany endpoint API tego wymaga. W tym projekcie jest tylko jeden endpoint REST API, który służy do pobierania informacji o stacjach benzynowych, i nie wymaga uwierzytlenienia. Z tego powodu nie zaimplementowano uwierzytelniania opartego na tokenie.