Politechnika Wrocławska
Wydział Informatyki i Telekomunikacji

Kierunek: **TIN**
Specjalność: **TIU**

PRACA DYPLOMOWA INŻYNIERSKA

Projekt oraz implementacja strony internetowej dla szczepu harcerskiego wraz z systemem zarządzania treścią

Radosław Łuckoś

Opiekun pracy
Dr. Inż. Michał Kowal

Słowa kluczowe: aplikacja webowa, JavaScript, React, MongoDB

WROCŁAW 2021

Streszczenie pracy inżynierskiej

Niniejsza praca obejmuje cały proces tworzenia strony internetowej dla szczepu harcerskiego wraz z dedykowanym systemem zarządzania treścią od sformułowania założeń, przez projekt aż po publikację aplikacji. Pracę można podzielić na dwie części.

Pierwsza część pracy skupia się na dokładnym przedstawieniu procesu projektowania tej aplikacji. Faza ta zaczyna się od sformułowania założeń projektu oraz planowanych funkcjonalności. Następnie przedstawiane są wykorzystane technologie i narzędzia. Kolejnym etapem jest omówienie projektu całej aplikacji, zarówno od strony graficznej jak i od strony strukturalnej.

Druga część pracy dotyczy już implementacji aplikacji. Znajdują się tam szczegółowe opisy zastosowanych rozwiązań oraz prezentowane są efekty końcowe.

Pracę podsumowuje ostatni rozdział zawierający wnioski oraz perspektywy ewentualnego dalszego rozwoju aplikacji.

Brief thesis summary

This engineer’s thesis covers all of the process of the development of the web app for the scout’s tribe including dedicated content management system, from the conception through the projecting phase up till deployment. The thesis can be divided into two parts.

First of them concentrates on detailed presentation of the projecting process. It starts with the couched assumptions and planned features. Then, the technology stack and tool set are presented. After that there is a report of the whole projecting process of the app, from both the graphical side and the app structure.

Second part pertain to the implementation process. There are detailed descriptions of the applied solutions there and the end results are presented.

In the last chapter there are some conclusions and perspectives of further development of the app mentioned.

Spis treści

1. Wstęp

W pierwszym rozdziale pracy przedstawiono temat pracy oraz dokładnie określono postawione jej cele oraz zakres, który obejmuje.

1.1 Uzasadnienie wyboru tematu pracy

Podczas wymyślania tematu pracy inżynierskiej autorowi zależało na dwóch podstawowych warunkach.

Po pierwsze, aby był to projekt mieszczący się w szerokich ramach programowania webowego, które znajduje się w kręgu zainteresowań autora i z którym wiąże on swoje plany zawodowe. Dlatego tematem pracy jest budowa aplikacji webowej.

Po drugie autor chciał, aby projekt był jak najbardziej praktyczny oraz by odpowiadał na konkretną potrzebę. Miało to na celu stworzenie aplikacji, która zostanie realnie wykorzystana, a nie zostanie tzw. „pracą do szuflady". Jako że autor jest instruktorem Związku Harcerstwa Polskiego, a środowisko, w którym działa, dynamicznie się rozwija i potrzebuje reklamy, by przyciągnąć kolejnych potencjalnych członków, uznano, iż jest to doskonałe pole do wykorzystania w pracy. To pozwoliło ukształtować końcową wersję tematu.

1.2 Aplikacja webowa a strona internetowa [1]

Przed rozpoczęciem pracy nad projektem aplikacji należało sobie uświadomić czym jest aplikacja internetowa, gdyż już samo to sformułowanie określa pewne wymagania, które projekt musi spełniać. Zamiast jednak rozpocząć od dzielących te dwa pojęcia różnic, zaczęto od cech wspólnych. Mianowicie w obu przypadkach poruszamy się w obrębie programów, z których korzystanie wymaga dostępu do sieci. To odróżnia te dwie rzeczy od na przykład aplikacji desktopowych czy mobilnych, które tego wymagania nie muszą spełniać. Sprawia to, iż zarówno aplikacje internetowe jak i strony internetowe są niezależne od systemu operacyjnego, a do korzystania z nich potrzebujemy jedynie przeglądarki internetowej, co czyni je bardzo łatwo dostępnymi.

Strona internetowa jest jednak pojęciem o wiele węższym niż aplikacja. Strona internetowa to uporządkowany zbiór statycznych elementów, połączonych ze sobą za pomocą nawigacji i linków. Do jej budowy zazwyczaj wystarczą jedynie technologie frontendowe takie jak HTML, CSS oraz JavaScript. Strona internetowa ma jednak głównie charakter informacyjny. Przedstawia użytkownikowi pewną treść oferując użytkownikowi niewiele ponadto.

Aplikacja internetowa, oprócz przekazania pewnego zestawu informacji, ma charakter interaktywny, co oznacza możliwość interakcji użytkownik – aplikacja. Użytkownik jest w stanie np. logować się do aplikacji, wprowadzać i przesyłać do niej dane, dokonywać płatności itd. To zdecydowanie poszerza wachlarz zastosowań biznesowych aplikacji webowych, czyniąc je najpopularniejszym typem programów komputerowych w dzisiejszych czasach.

1.3 Cele oraz zakres pracy

Ostatnim zadaniem przed przystąpieniem do realizacji tematu pracy było określenie konkretnych celów oraz zakresu, który praca ma obejmować. Zostały one przedstawione w następnych akapitach.

1.3.1 Cele

Poniżej wyszczególniono cele tej pracy:

- Stworzenie aplikacji webowej dla szczepu harcerskiego pełniącą funkcję informacyjno-promocyjną, umożliwiającą zarządzanie treścią oraz uaktualnianie jej przez kadrę szczepu,
- Wykorzystanie oraz poszerzenie wiedzy i umiejętności z zakresu programowania webowego w języku JavaScript oraz technologiach z nim związanych,
- Poszerzenie wiadomości o bazach danych oraz tworzeniu interfejsów programistycznych typu REST,
- Nauka łączenia oprogramowania po stronie klienta (frontend) z częścią znajdującą się po stronie serwera (backend),
- Stworzenie od podstaw projektu programistycznego z zastosowaniem nowoczesnych technologii zarówno frontendowych jak i backendowych.

1.3.2 Zakres prac

Zakres pracy inżynierskiej obejmuje cały proces powstania aplikacji. Poszczególne jego elementy to:

- sformułowanie założeń projektu,
- projekt struktury strony internetowej wraz z systemem zarządzania treścią,
- projekt graficzny strony internetowej,
- projekt REST API oraz bazy danych,
- wybór technologii do użycia w projekcie,
- implementacja,
- deployment aplikacji.

2. Założenia projektu

Niniejszy rozdział dokładnie omawia poszczególne założenia oraz wymagania postawione tworzonej aplikacji.

Głównym założeniem pracy jest zbudowanie aplikacji webowej, która będzie pełniła funkcję promocyjną oraz informacyjną, dając jednocześnie możliwość dynamicznego zarządzania jej treścią przez administratorów.

Z tej przyczyny istnieją 2 poziomy dostępu do aplikacji: użytkownik oraz administrator. Użytkownik ma dostęp jedynie do statycznej części aplikacji pełniącej funkcję informacyjną oraz promocyjną. Użytkownik nie ma dostępu do panelu admistracyjnego. Natomiast administrator ma możliwość zarządzania wyświetlanymi na stronie danymi poprzez dedykowany panel.

2.1 Wymagania funkcjonalne

Poszczególne wymagane funkcjonalności aplikacji wypisano przy użyciu metody znanej z oprogramowania tworzonego w metodyce zwinnej, czyli tzw. User Stories. [2] Metoda ta polega na nieformalnym wypisaniu poszczególnych funkcjonalności i opisaniu ich z perspektywy użytkownika końcowego (w tym przypadku są to użytkownik oraz administrator). Do opisu używa się języka nietechnicznego, tak by był on zrozumiały nie tylko dla programistów, ale i dla klienta. Celem stosowania tej metody jest wyartykułowanie potrzeb klienta w postaci konkretnych funkcjonalności oprogramowania, niezbędnych do realizacji projektu.

- Jako użytkownik mogę uzyskać interesujące mnie informacje o szczepie, jego drużynach i działalności.
- Jako użytkownik mogę przeczytać relacje z minionych wydarzeń.
- Jako użytkownik mogę zapoznać się z aktualnymi ogłoszeniami dotyczącymi działalności szczepu.
- Jako użytkownik mogę zobaczyć lokalizację Harcerskiego Ośrodka Wodnego „Stanica”, na którym odbywają się zbiórki.
- Jako użytkownik mogę sprawdzić termin zbiórek.
- Jako użytkownik mogę sprawdzić dane kontaktowe do kadry lub skontaktować się bezpośrednio przez stronę internetową przy pomocy formularza kontaktowego.
- Jako użytkownik mogę zapoznać się z sekcją FAQ, by znaleźć odpowiedzi na nurtujące mnie pytania.
- Jako użytkownik mogę obejrzeć galerię zdjęć.
- Jako administrator mam możliwość zalogowania się do panelu administracyjnego.
- Jako administrator mogę zarządzać kontami innych administratorów, mających dostęp do panelu administracyjnego. Mogę tworzyć je i usuwać.
- Jako administrator mogę zarządzać relacjami oraz ogłoszeniami na stronie, dodawać je, edytować oraz usuwać.
- Jako administrator mogę edytować dane mojego konta.

3. Wybór technologii oraz narzędzi

Do stworzenia całego projektu został wykorzystany wieloparadygmatowy język skryptowy JavaScript. Jest on ulubionym językiem programowania autora i wiąże on z nim plany zawodowe, co zaważyło na tym, iż został on w projekcie. [3] Standardem języka JavaScript jest ECMAScript. Od 2012 roku wszystkie nowoczesne przeglądarki całkowicie obsługują standard ECMAScript 5.1. W czerwcu 2015 miała miejsca aktualizacja standardu do ES6 (ECMAScript 6). Od tego czasu standardy wychodzą cyklicznie raz do roku.

Nie bez znaczenia w kwestii wyboru języka pozostają jego zdecydowane zalety na tle innych języków. Jest to najpopularniejszy język, jeśli chodzi o zastosowanie w budowie aplikacji webowych oraz stron internetowych i jeden z popularniejszych języków w rozrachunku ogólnym. Według indeksu TIOBE [4] znajduje się on na 7 miejscu w rankingu popularności języków programowania na świecie. Za popularnością idzie wysoki poziom społeczności zbudowanej wokół języka, co z kolei ułatwia radzenie sobie z napotkanymi trudnościami.

Poza jego popularnością do zdecydowanych zalet tego języka należy jego uniwersalność – można go użyć zarówno po stronie klienta jak i po stronie serwera. Są także rozwiązania pozwalające na użycie tego języka do budowy aplikacji desktopowych (np. Electron.js) oraz natywnych.

Kolejną rzeczą przemawiającą za tym językiem jest mnogość bibliotek oraz szybkość jego rozwoju gwarantujący stały wzrost zapotrzebowania w zakresie tego języka w przyszłości oraz prostota składni. Jest to zatem język, który jest z pewnością należy do języków przyszłościowych.

3.1 Wybór narzędzi i technologii użytych po stronie klienta (Frontend)

Wybór technologii, z której skorzystano przy realizacji części frontendowej, uzależniony był od zastosowanego podejścia. Rozważane były dwie możliwości.

[5] Pierwsza to podejście typu MPA (ang. Multi Page App). Charakteryzuje je obecność wielu plików HTML odpowiadających za osobne widoki aplikacji, co oznacza każdorazowe przeładowanie strony i wysłanie zapytania na serwer, w momencie zmiany widoku (np. zmiana podstrony). Wpływa to negatywnie na UX (ang. User Experience). Ze względu na konieczność pobierania nowego pliku HTML przy każdorazowej zmianie strony spada szybkość ładowania treści i wyświetlania jej co może prowadzić do zbytnich opóźnień w reakcji systemu na działanie użytkownik jak na dzisiejsze standardy.

Przeciwieństwem tego podejścia jest podejście SPA (ang. Single Page App). Pozwala ono na użycie tylko jednego pliku HTML oraz renderowanie pożądanych widoków aplikacji bez przeładowywania całej aplikacji. Sprawia to, iż nie jest potrzebne ładowanie widoków po stronie serwera, cała treść dociera do użytkownika już przy pierwszym załadowaniu, a jedyną formą komunikacji z serwerem jest wymiana informacji na poziomie warstwy danych. Sprawia to, iż aplikacja reaguje na działania użytkownika bez zbędnych opóźnień. Oczywiście wadą tego typu podejścia jest wydłużenie czasu załadowania pierwszej treści, ze względu na konieczność przesłania całego kodu już za pierwszym razem, jednak przy dzisiejszej technologii nie ma to zbyt odczuwalnego wpływu na UX aplikacji. Da się również podzielić kod (ang. Code splitting), tak aby możliwie skrócić czas załadowania pierwszej treści i jeszcze poprawić UX omawianej aplikacji.

[6] Aplikacja po stronie klienta powstała z użyciem Reacta w wersji 17.0.2. Jest to biblioteka JavaScriptu szeroko wykorzystywana do tworzenia interfejsów użytkownika.

React jest bardzo lekki, elastyczny i wydajny. Dzięki użyciu abstrakcyjnej kopii DOMu (Dokument Object Model), czyli tak zwanego Virtual DOM, React dokonuje poszczególnych zmian bez zmiany reszty interfejsu co sprawia, iż jest uważany za jedno z najszybszych, o ile nie najszybsze rozwiązanie tego rodzaju. Na jego wydajność i szybkość wpływa też możliwość wielokrotnego wykorzystania poszczególnych komponentów. Przez wrażliwość kryteriów algorytmów pozycjonujących na lekkość oraz szybkość renderowanie strony React bardzo pozytywnie oddziałuje na SEO (ang. search engine optimization). Dzięki swojej popularności oraz stałemu rozwojowi kod w nim pisany jest stabilny. Warto również wspomnieć o jednokierunkowym przepływie danych, który sprawia, że aplikacja jest mniej podatna na błędy oraz łatwiejsza w debugowaniu, ze względu na dużą kontrolę nad przepływem danych. Do ostylowania komponentów użyto kaskadowych arkuszy stylów (CSS) wraz z preprocesorem SCSS. Poprawia on zdecydowanie czytelność kodu CSS oraz poszerza jego możliwości o tak użyteczne funkcjonalności jak np. zmienne czy też zagnieżdżanie kolejnych reguł css.

3.2 Backend [7]

Po stronie serwera wykorzystany został framework technologii Node.js - Express.js. Node.js to środowisko uruchomieniowe dla języka JavaScript po stronie serwera. Express.js jest bardzo minimalistycznym oraz elastycznym frameworkiem ułatwiającym i automatyzującym operacje związane z API, routingiem, żądaniami http oraz wieloma innych. Jednym z jego głównych zalet i celem powstania zarazem jest oszczędność czasu oraz zdecydowana poprawa czytelności przy pisaniu kodu. Kolejną zaletą jest jego skalowalność w pracy nad projektem.

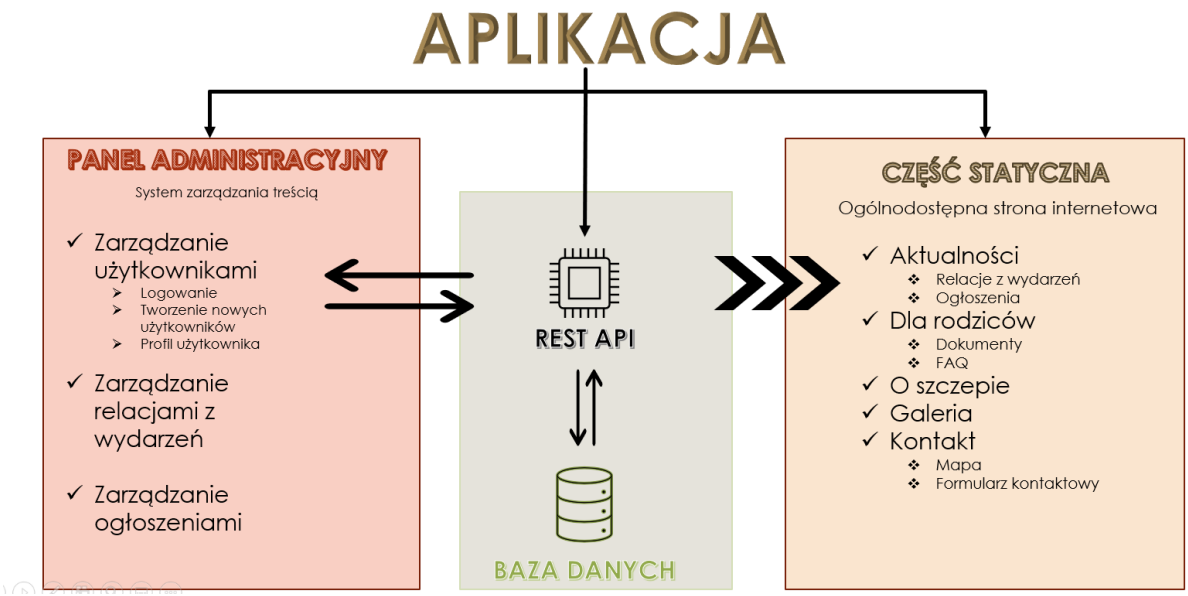
3.3 Środowisko bazodanowe [8]

Przy wyborze bazy danych wybór dotyczył w pierwszej kolejności rodzaju bazy danych, która byłaby najodpowiedniejsza. Dane wynikające z określonych w rozdziale 2 funkcjonalności nie są ze sobą powiązane. Każda grupa danych może być zatem składowana oddzielnie co pozwala wykluczyć użycie relacyjnych baz danych. Optymalnym rozwiązaniem jest tu użycie bazy danych typu dokumentowego. Dokumentowe bazy danych przechowują dane w postaci dokumentów strukturalnych (np. w formacie JSON, lub XML). Ich zaletami krótszy czas pobierania niż w przypadku baz relacyjnych, możliwość zmiany struktury danych bez ingerencji w już zebrane dane, uniwersalność oraz prostota (brak potrzeby używania języka SQL). Wśród baz dokumentowych najpopularniejszą jest baza MongoDB i to właśnie ona została wybrana do wykorzystana w aplikacji.

4. Projekt aplikacji

Rozpoczynany w tym miejscu rozdział zawiera szczegółowy opis zaprojektowanych kolejnych części aplikacji. Odpowiednie podrozdziały skupiają się na wybranych fragmentach procesu tworzenia projektu aplikacji, zaczynając od jej struktury i przechodząc kolejno przez jej elementy składowe takie jak interfejs graficzny, baza danych oraz API.

4.1 Opis struktury aplikacji



Rysunek 1. Schemat struktury aplikacji

Po stronie klienta aplikacja składa się z dwóch głównych części: treści dostępnej dla każdego użytkownika oraz panelu administracyjnego, do którego należy się zalogować, aby uzyskać dostęp. Z poziomu panelu administracyjnego możemy zmieniać dane składowane w bazie danych wykorzystując do tego REST API. API dostarcza danych do strony statycznej oraz panelu administracyjnego. Panel administracyjny podzielony jest na trzy podstrony odpowiadające za obsługę poszczególnych funkcjonalności:

- zarządzanie kontami administratorów,
- zarządzanie ogłoszeniami,
- zarządzanie relacjami z wydarzeń.

Część statyczna jest ogólnodostępna dla każdego zainteresowanego i składa się zaś z pięciu podstron:

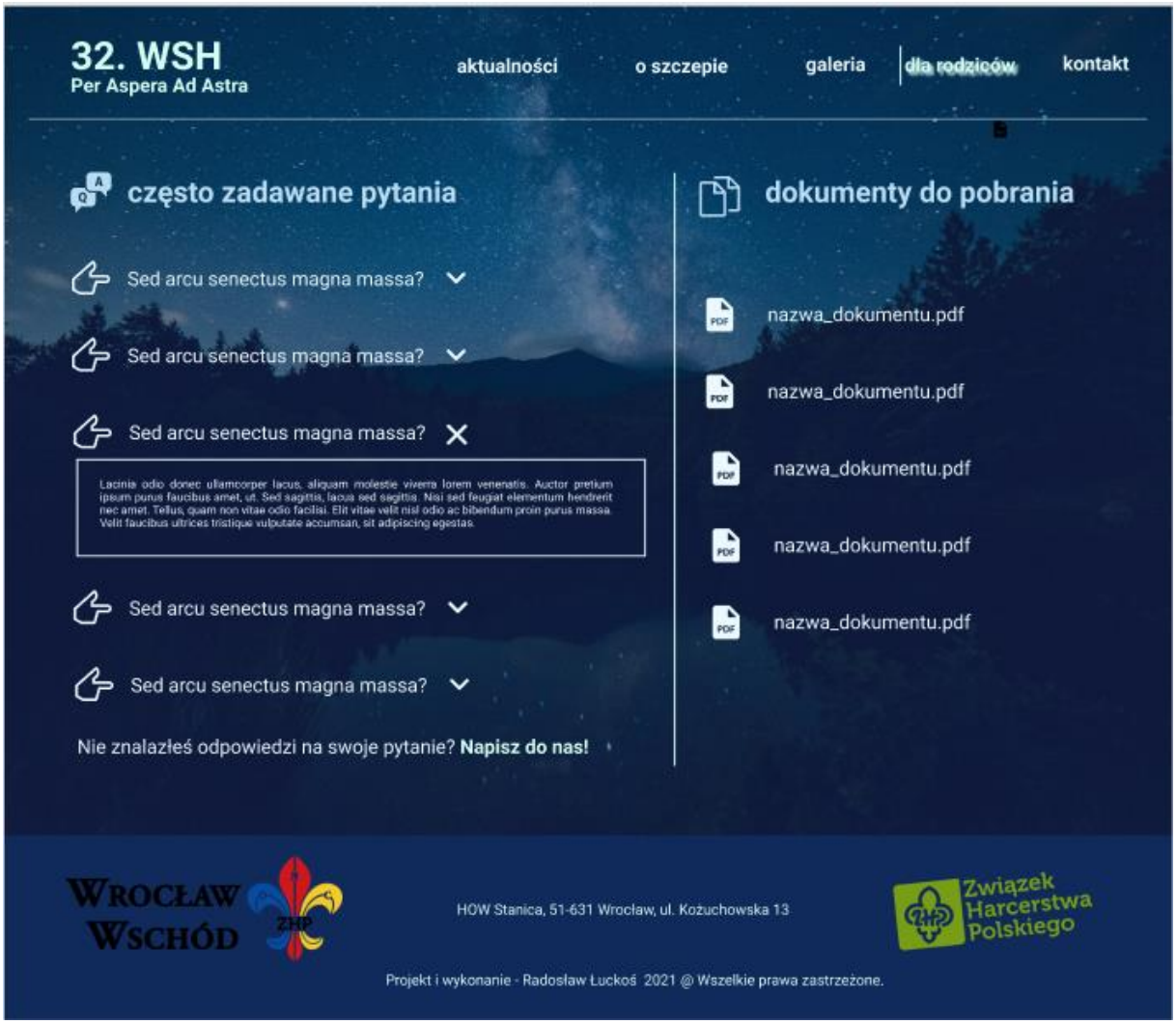
- Aktualności - to strona główna. Znajdują się na niej ogłoszenia, relacje oraz slajder z wybranymi zdjęciami. Pełni głównie funkcję promocyjną.
- O szczepie – Ta podstrona zawiera informacje o szczepie. Znajdują się tu opisy każdej z drużyn działających w szczepie oraz informacje o kadrze.
- Galeria – To podstrona z najciekawszymi zdjęciami mająca na celu pokazać najlepsze momenty działalności szczepu w sposób przyjemny dla oka.
- Dla rodziców – Na tej podstronie rodzic może znaleźć sekcję FAQ z najczęstszymi pytaniami zadawanymi przez rodziców oraz wzory dokumentów do pobrania.
- Kontakt – Podstrona z danymi kontaktowymi do kadry oraz mapą z zaznaczonym miejscem odbywania się zbiórek. Znajduje się tu także formularz kontaktowy do mailowego kontaktu bezpośredniego z kadrą szczepu.

4.2 Projekt graficzny interfejsu

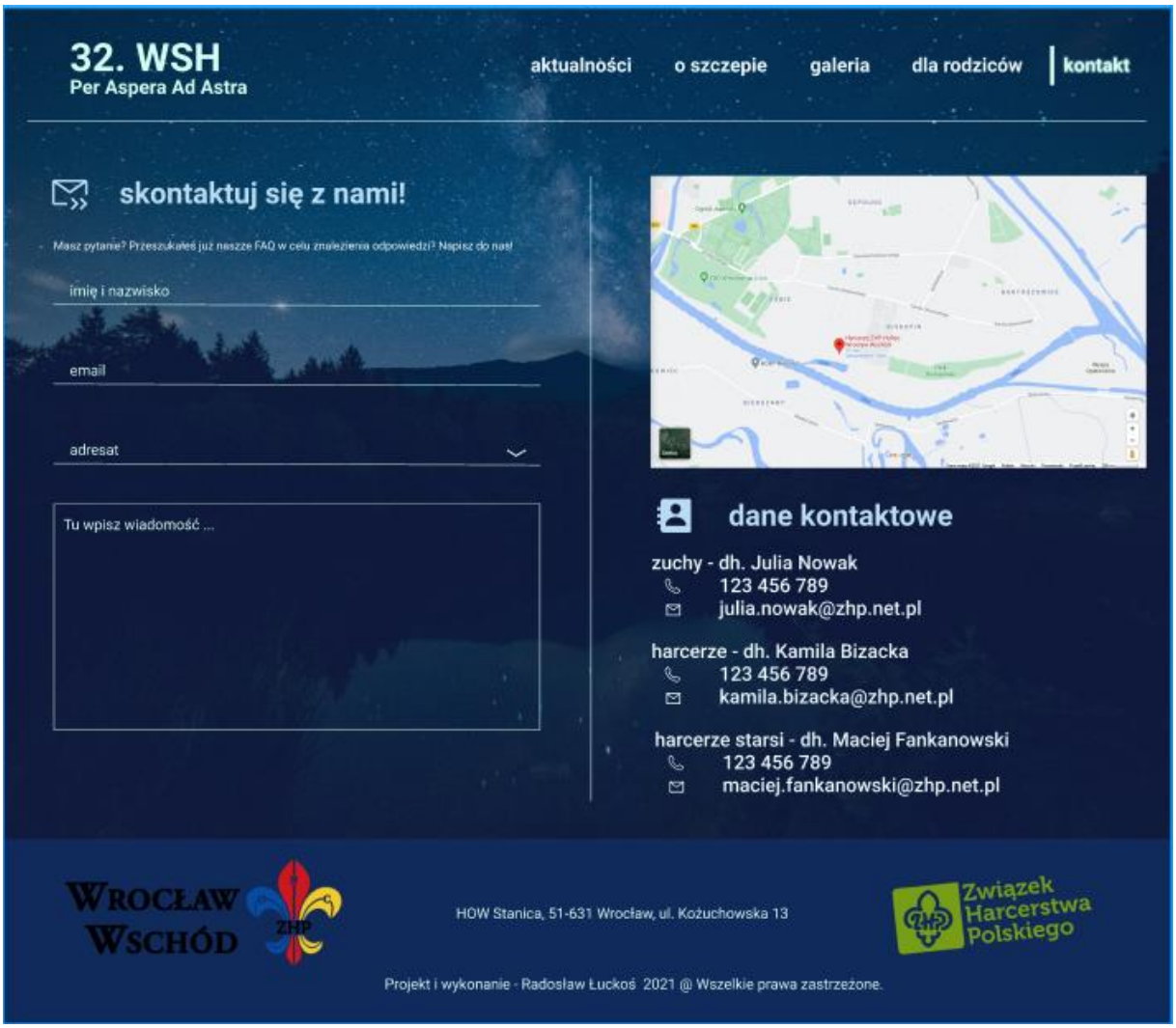
Projekt graficzny powstał jedynie dla desktopowej wersji części statycznej aplikacji, ze względu na skomplikowany layout oraz jego kluczową rolę dla pełnionej przez tę część aplikacji. W przypadku wersji dla mniejszych ekranów układ treści nie wymagał szczególnego projektowania, ze względu na fakt, iż przewidywany kolumnowy układ treści jest dosyć podstawowym układem. Część administracyjna aplikacji nie została zaprojektowana, gdyż do jej wykonania zaplanowano wykorzystanie biblioteki UI, ze względu na mniejszą rolę designu. Także galeria została w projekcie graficznym pominięta, gdyż była ona planowana do zrobienia w stylu „masonry grid" przy użyciu zewnętrznej biblioteki. Poszczególne zaprojektowane widoki przedstawione zostały na Rysunkach 2. - 5. Wyszarzone fragmenty obrazują obszar, który nie jest widoczny od razu, a dopiero po scrollowaniu.

Rysunek 2. Projekt widoku strony głównej

Rysunek 3. Projekt graficzny widoku podstrony "O szczepie"



Rysunek 4. Projekt graficzny widoku podstrony "Dla rodziców"



Rysunek 5. Projekt graficzny widoku "Kontakt"

4.3 Projekt bazy danych

Baza danych potrzebna do obsługi aplikacji nie musi być bardzo skomplikowana. To w rzeczywistości trzy proste kolekcje, w których przetrzymywane będą poszczególne dokumenty odpowiednich kategorii zawierające dane typu JSON.

I tak kolejne kolekcje to:

„Ads” – kolekcja zawierająca dokumenty JSON z danymi ogłoszeń. Każdy z dokumentów zawiera takie dane jak: id ogłoszenia, tytuł ogłoszenia, deadline, a więc termin, do którego odnosi się ogłoszenie oraz treść ogłoszenia. Przykładowy dokument tej kolekcji przedstawia Rysunek 6.


```
_id: ObjectId("61b3c2c52af9f4f48236e9e3")
title: "Przykładowe ogłoszenie 1"
body: "Przykładowa treść"
deadline: 2021-11-11T00:00:00.000+00:00
__v: 0
```


Rysunek 6. Przykładowy dokument kolekcji "Ads"

„Stories” – kolekcja, w której składowane są dokumenty z danymi relacji z wydarzeń. Każda z relacji zawiera takie pola jak: id relacji, tytuł, data, opis, tag (definiujący która z drużyn szczepu zaangażowana była w dane wydarzenie). Przykładowy dokument należący do tej kolekcji został przedstawiony na Rysunku 7.


```
_id: ObjectId("61b3c40c2af9f4f48236e9e9")
title: "Przykładowa relacja"
desc: "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla imperdi..."
tag: "HS"
date: 2021-12-10T21:18:04.171+00:00
__v: 0
```


Rysunek 7. Przykładowy dokument kolekcji "Stories"

„Users” – kolekcja, w której składowane są dokumenty z danymi użytkowników. Każdy obiekt dotyczący konkretnego użytkownika zawiera takie dane jak: imię, nazwisko, adres mailowy, hasło(zaszyfrowane), jednostka(drużyna), funkcje, numer telefonu oraz id. Przykładowy dokument tej kolekcji przedstawia Rysunek 8.


```
_id: ObjectId("61b3c4752af9f4f48236e9f0")
name: "Użytkownik"
lastname: "Testowy"
email: "test.admin@zhp.net.pl"
password: "$2a$10$zfl80Y16yweKKk.7e6KvneBZNHJQ8aPXrukwDrldgQfjUem8D7FBG"
unit: "32 WDSH Pegaz"
functions: Array
  0: "drużynowy"
phone: 123456789
__v: 0
```


Rysunek 8. Przykładowy dokument kolekcji "Users"

4.4 Projekt REST API [9]

API w odniesieniu to programowania webowego jest rodzajem interfejsu programistycznego, w którym do komunikacji między aplikacjami znajdującymi się na innych serwerach sieciowych używa się protokołów sieciowych, takich jak na przykład http.

REST (ang. Representational State Transfer) to pewien zbiór wymogów/reguł określających definicję zasobów oraz dostępu do nich, pozwalających na budowę wydajnych, niezawodnych i skalowalnych systemów wymiany danych typu klient-serwer. Do wymiany informacji między klientem a serwerem używane są metody wspomnianego wcześniej protokołu http takie jak na przykład POST, GET, DELETE czy UPDATE. Cykl postępowania z REST API wygląda następująco:

1. Klient przygotowuje żądanie do wysłania na konkretny adres (endpoint).
2. Klient wysyła żądanie (request).
3. Serwer otrzymuje żądanie, przetwarza je po czym zwraca odpowiedź (response).
4. Klient przetwarza odpowiedź.

Zaletami stosowania REST API w aplikacjach webowych jest możliwość łatwego skalowania systemu, oddzielenia warstwy prezentacji od warstwy logicznej naszych aplikacji oraz uniwersalność pozwalająca na użycie jednego API przez kilka aplikacji.

Do zaimplementowania pełni planowanych funkcjonalności z wykorzystaniem REST API konieczne jest wystawienie odpowiednich endpointów przedstawionych w Tabelach 1– 3.

Tabela 1. Endpointy dotyczące zarządzania użytkownikami

Nazwa	Metoda	Opis
LoginUser	POST	Przyjmuje obiekt z loginem i hasłem, służy do zweryfikowania użytkownika i umożliwienia procesu logowania. W przypadku udanej weryfikacji danych odpowiedzi zwraca token użytkownika i jego dane. W przypadku nieudanej weryfikacji tożsamości użytkownika zwraca błąd.
RegisterUser	POST	Jako argument przyjmuje obiekt z danymi użytkownika. Po sprawdzeniu otrzymanych danych zapisuje nowego użytkownika w bazie danych lub zwraca błąd. Endpoint zabezpieczony przed dostępem osób nieuprawnionych.
GetUsers	GET	Zwraca obiekty wszystkich użytkowników dostępnych w bazie danych. Endpoint zabezpieczony przed dostępem osób nieuprawnionych.
DeleteUser	DELETE	Jako parametr przyjmuje id użytkownika do usunięcia. Weryfikuje id, po czym usuwa użytkownika w przypadku powodzenia weryfikacji lub zwraca błąd. Endpoint zabezpieczony przed dostępem osób nieuprawnionych.
UpdateUser	PATCH	Jako parametr przyjmuje id użytkownika do edycji. Weryfikuje id, po czym podmienia dane przekazane w ciele zapytania w bazie danych w przypadku powodzenia weryfikacji lub zwraca błąd w przypadku niepowodzenia. Endpoint zabezpieczony przed dostępem osób nieuprawnionych.

Tabela 2. Endpointy dotyczące zarządzania relacjami z wydarzeń

Nazwa	Metoda	Opis
CreateStory	POST	Przyjmuje obiekt z danymi nowej relacji. Po sprawdzeniu otrzymanych danych zapisuje nową relację w bazie danych lub zwraca błąd. Endpoint zabezpieczony przed dostępem osób nieuprawnionych.
GetStories	GET	Zwraca obiekty wszystkich relacji dostępnych w bazie danych.
DeleteStory	DELETE	Jako parametr przyjmuje id relacji do usunięcia. Weryfikuje id, po czym usuwa relację w przypadku powodzenia weryfikacji lub zwraca błąd. Endpoint zabezpieczony przed dostępem osób nieuprawnionych.
UpdateStory	PATCH	Jako parametr przyjmuje id relacji do edycji. Weryfikuje id, po czym podmienia dane przekazane w ciele zapytania w bazie danych w przypadku powodzenia weryfikacji lub zwraca błąd w przypadku niepowodzenia. Endpoint zabezpieczony przed dostępem osób nieuprawnionych.

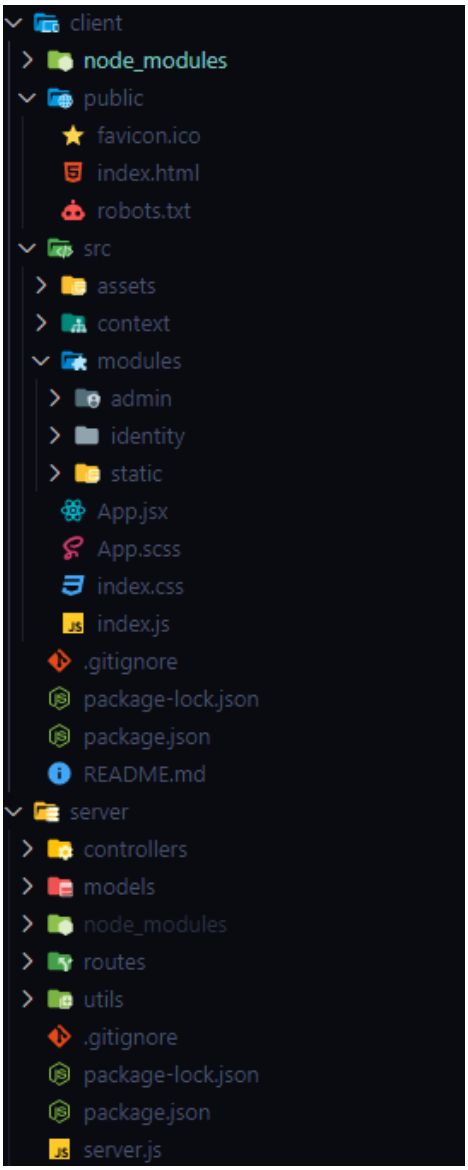
Tabela 3. Endpointy dotyczące zarządzania ogłoszeniami

Nazwa	Metoda	Opis
CreateAd	POST	Przyjmuje obiekt z danymi nowego ogłoszenia. Po sprawdzeniu otrzymanych danych zapisuje nowe ogłoszenie w bazie danych lub zwraca błąd. Endpoint zabezpieczony przed dostępem osób nieuprawnionych.
GetAds	GET	Zwraca obiekty wszystkich ogłoszeń dostępnych w bazie danych.
DeleteAd	DELETE	Jako parametr przyjmuje id ogłoszenia do usunięcia. Weryfikuje id, po czym usuwa ogłoszenie w przypadku powodzenia weryfikacji lub zwraca błąd. Endpoint zabezpieczony przed dostępem osób nieuprawnionych.
UpdateAd	PATCH	Jako parametr przyjmuje id ogłoszenia do edycji. Weryfikuje id, po czym podmienia dane przekazane w ciele zapytania w bazie danych w przypadku powodzenia weryfikacji lub zwraca błąd w przypadku niepowodzenia. Endpoint zabezpieczony przed dostępem osób nieuprawnionych.

5.Implementacja

Przechodząc do implementacji należy zaznaczyć, iż do kontroli wersji użyto systemu git. Pierwszym krokiem było utworzenie projektu oraz założenie specjalnego repozytorium. Repozytorium zostało założone w serwisie Github i jest dostępne pod adresem: https://github.com/radoslaw-luckos/wsh-mern.git

Struktura ogólna katalogów projektu przedstawiona została na Rysunku 9.



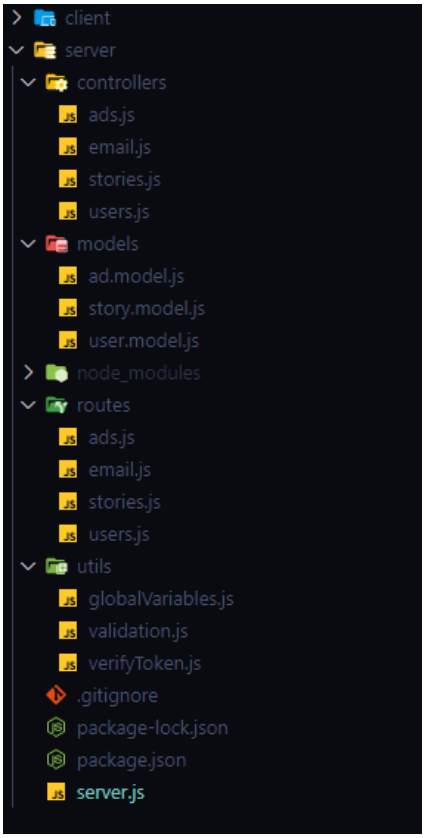
Rysunek 9. Struktura katalogów i plików projektu

Projekt składa się z dwóch głównych katalogów: **client** oraz **server**. Pierwszy z nich, **client**, zawiera pliki aplikacji front-end natomiast backend znajduje się w folderze **server**.

5.1 Implementacja REST API

Proces developmentu aplikacji rozpoczęto od zaimplementowania API niezbędnego do implementacji dalszych części aplikacji.

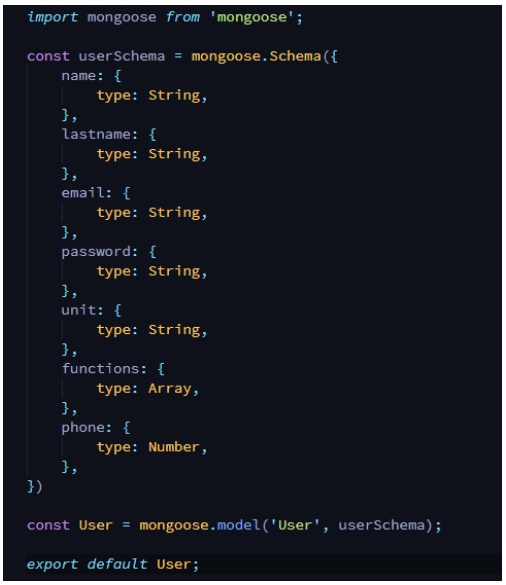
5.1.1 Struktura REST API



Rysunek 10. Struktura plików REST API

Rysunek 10. Przedstawia strukturę plików aplikacji backend, a więc naszego REST API.

Plikiem głównym aplikacji działającej po stronie serwera jest plik server.js. To tu znajduje się konfiguracja serwera oraz konfiguracja połączenia z bazą danych. Folder models zawiera schematy definiujące strukturę danych, które są zapisywane w bazie danych. Przykładem takiego modelu jest model użytkownika przedstawiony na Rysunku 11.



```
import mongoose from 'mongoose';

const userSchema = mongoose.Schema({
    name: {
        type: String,
    },
    lastname: {
        type: String,
    },
    email: {
        type: String,
    },
    password: {
        type: String,
    },
    unit: {
        type: String,
    },
    functions: {
        type: Array,
    },
    phone: {
        type: Number,
    },
})

const User = mongoose.model('User', userSchema);

export default User;
```

Rysunek 11. Przykładowy model danych zapisywany w bazie danych MongoDB

W folderze **routes** zdefiniowane wszystkie endpointy, które są wystawione przez REST API. Ścieżki dotyczące poszczególnych obszarów tematycznych (np. zarządzanie użytkownikami) znajdują się w jednym pliku (Rysunek 12).

```
const router = express.Router();

router.post('/register', auth, registerUser);
router.post('/login', loginUser);
router.get('/', auth, getUsers);
router.get('/:userId', auth, getUser);
router.delete('/delete/:userId', auth, deleteUser)
router.patch('/update/:userId', auth, updateUser)

export default router;
```

Rysunek 12. Przykład definicji poszczególnych endpointów w obszrze zarządzania użytkownikami

Foldery **utils** oraz **controllers** to foldery zawierające definicje konkretnych funkcji/zmiennych użytych w innych plikach. W przypadku folderu controllers są to funkcje wywoływane w momencie żądania dot. konkretnego endpointu, a ich celem jest obsługa żądania. Folder utils zawiera funkcje użytkowe oraz zmienne pomocnicze, używane wielokrotnie. Taka struktura plików pozwala w szybki i prosty sposób wprowadzać zmiany w już istniejących endpointach, czy też funkcjonalnościach, ale też pozwala na łatwe rozszerzanie naszego API poprzez wdrożenie nowych funkcjonalności.

5.1.2 Bezpieczeństwo REST API [10]

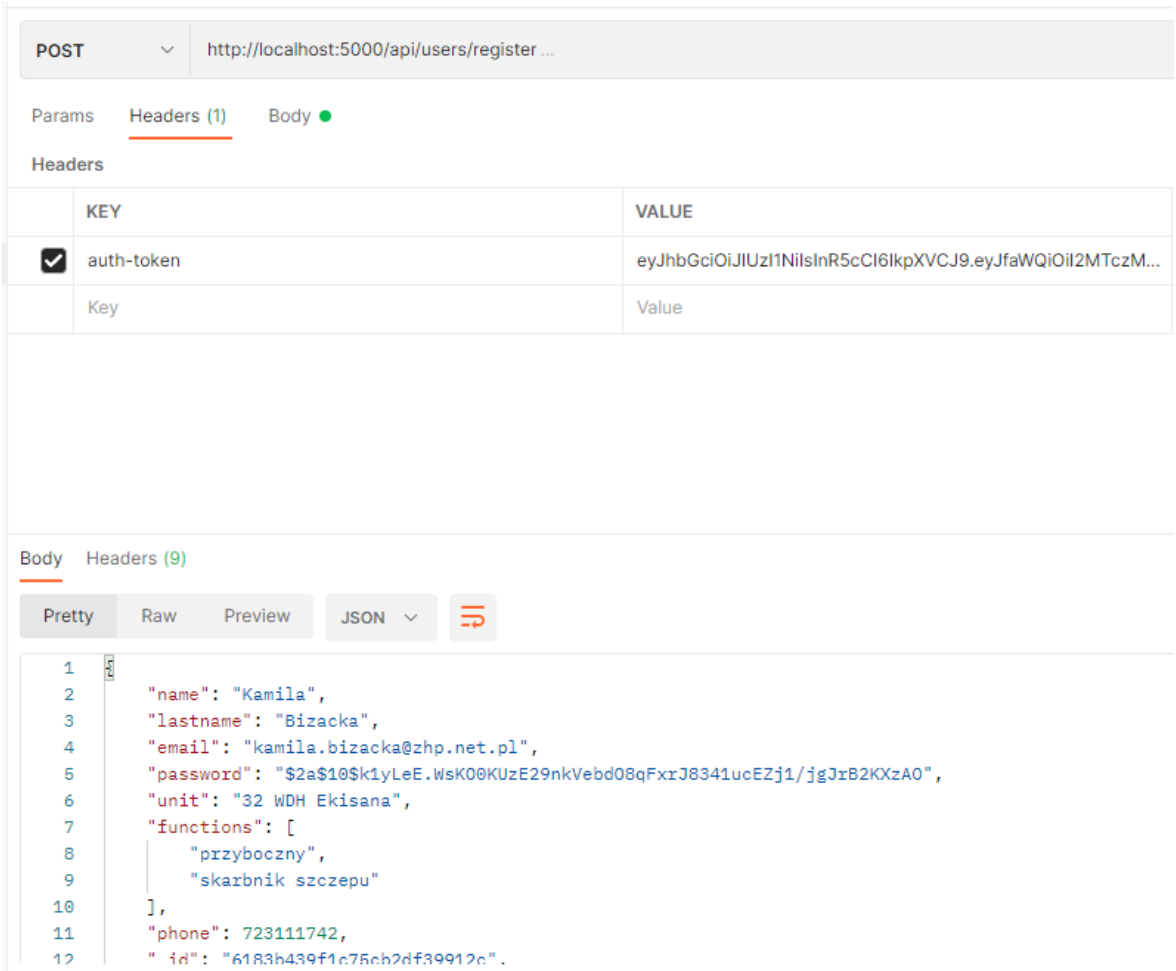
Kolejnym aspektem mojego rozwiązania, które moim zdaniem zasługuje na uwagę jest kwestia zabezpieczenia API. Dzieje się to na dwa sposoby.

Pierwszą kwestią jest przechowywanie haseł. Przechowywanie haseł w bazie danych w postaci jawnej w oczywisty sposób naraża aplikację na różne zagrożenia. Pozyskanie takiego hasła nie stanowi żadnego problemu. Nie jest to zatem optymalne podejście.

Jeśli chodzi o przechowywanie hasła w postaci niejawnej to istnieją dwa sposoby: zaszyfrowanie i zahashowanie hasła. W przypadku zaszyfrowania hasła problematycznym staje się przechowywanie klucza szyfrującego. Dlatego optymalną metodą wydaje się być przetrzymywanie hasła zahashowanego, do którego uzyskania najczęściej stosuje się funkcje skrótu. Funkcje te są funkcjami jednokierunkowymi, co oznacza, że na podstawie wyniku funkcji nie można obliczyć danych wejściowych.

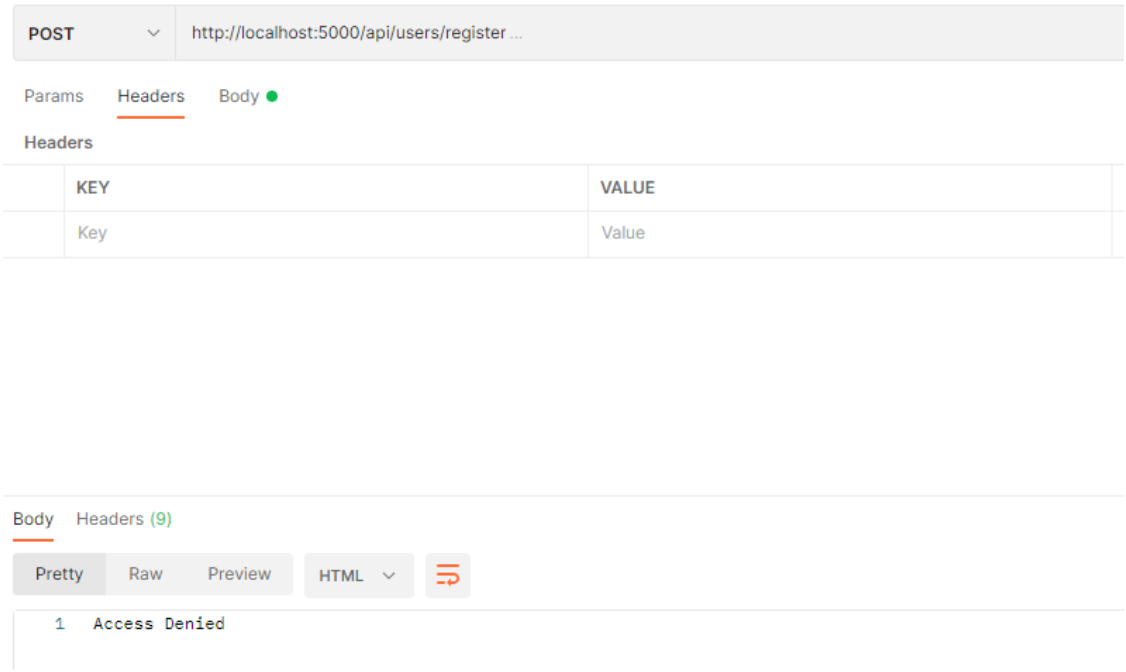
W tym celu wykorzystano bibliotekę bcrypt, pozwalającą skutecznie zabezpieczyć hasło przed odczytaniem algorytmem szyfrującym Blowfish, w sposób przyjemny dla developera.

Kolejna kwestia to zabezpieczenie wrażliwych ścieżek przed dostępem osób nieuprawnionych. W tym celu zastosowano JSON-Web-Token (JWT). Podczas logowania użytkownikowi zostaje przydzielony token, który następnie musi zostać umieszczony jako nagłówek każdego z zabezpieczonych w ten sposób żądań. Po zaimplementowaniu tego rozwiązania poprawną odpowiedź uzyskujemy tylko w momencie umieszczenia poprawnego nagłówka przed wysłaniem zapytania (Rysunek 13.).



Rysunek 13. Poprawne zapytanie z nagłówkiem auth-token oraz zwracana odpowiedź

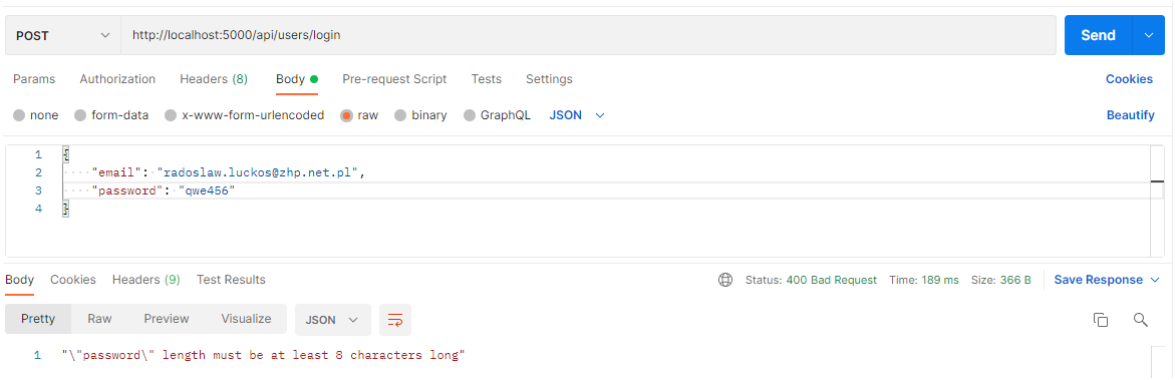
W przeciwnym wypadku, tzn. w przypadku braku nagłówka, endpoint zwraca odpowiedź „Access Denied” (Rysunek 14.).



Rysunek 14. Zapytanie bez nagłówka oraz zwracana odpowiedź o braku dostępu

5.1.3 Walidacja oraz testowanie API

Kolejnym ważnym aspektem pisania REST API była walidacja danych które użytkownik przesyła na serwer, tak by do bazy danych były zapisywane jedynie odpowiednio sformatowane dane. W przypadku złego formatu danych zwracana jest odpowiedź o statusie 400 – Bad request wraz z wiadomością określającą błąd (Rysunek 15.).



Rysunek 15. Przykładowy błąd zwracany przy walidacji wprowadzonych danych

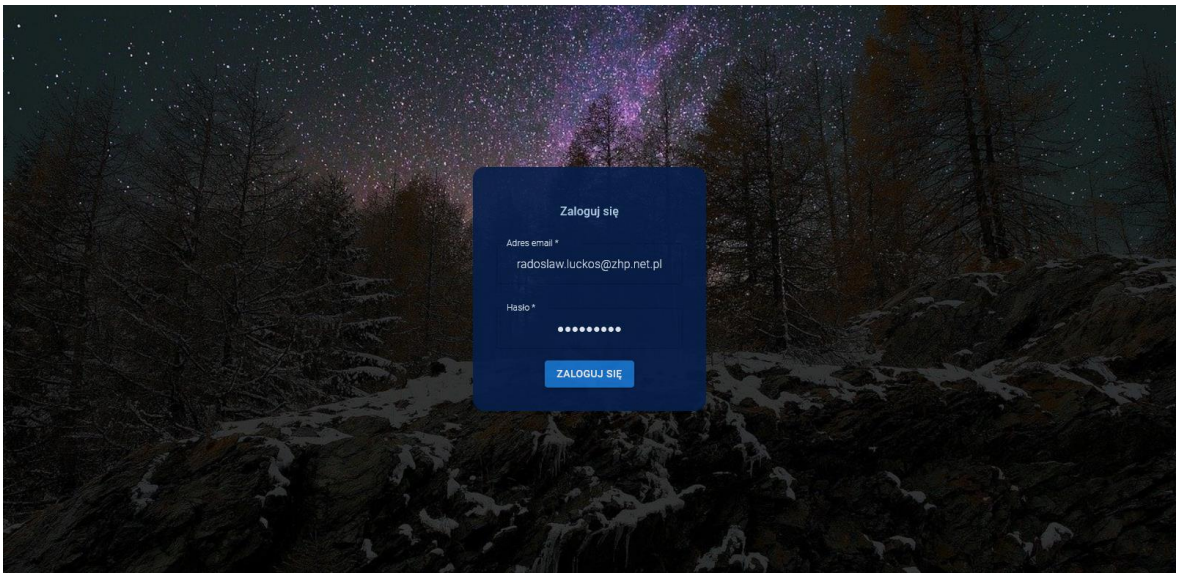
Jako że w trakcie powstawania poszczególnych endpointów nie istniała jeszcze część aplikacji działająca w przeglądarce, do testowania działania implementowanych funkcjonalności zostało użyte narzędzie Postman. Umożliwia ono wysyłanie żądań http na

serwer, na którym działa nasza aplikacja. Pozwala to nie tylko przetestować działanie implementowanych rozwiązań, ale także ułatwia debugowanie błędów.

5.2 Implementacja panelu administracyjnego

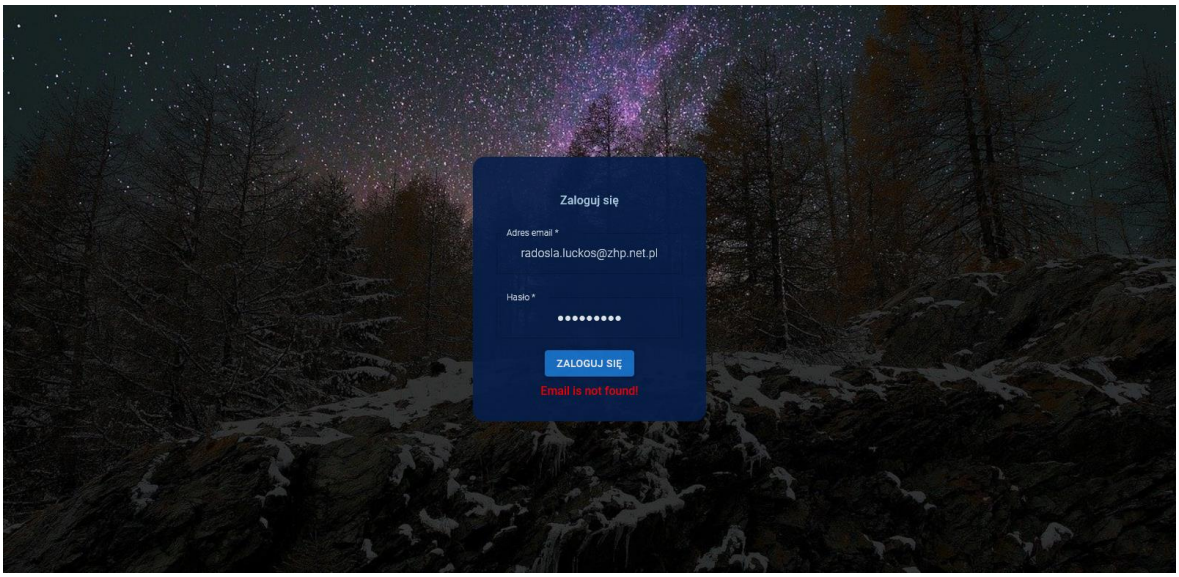
Podstawową funkcją panelu administracyjnego jest udostępnienie zweryfikowanemu użytkownikowi możliwości zarządzania treścią dostępną dla wszystkich użytkowników oraz zarządzania kontami innych administratorów. Taka funkcja wymaga wdrożenie przede wszystkim intuicyjnego, prostego i klarownego interfejsu. Nie ma tu potrzeby stosowania zbyt skomplikowanego layoutu. Jest to idealna sytuacja do wykorzystania biblioteki UI, pozwalającej na zbudowanie prostego układu treści z gotowych komponentów. W tym konkretnym przypadku zastosowano MUI, dedykowaną Reactowi wersję biblioteki Material UI.

Aby otrzymać dostęp do panelu administracyjnego użytkownik musi się zalogować. Aby to zrobić musi ręcznie dopisać do adresu URL strony ścieżkę /admin. Wtedy następuje sprawdzenie, czy w przeglądarce zapisany jest ważny token JWT, pozwalając ominąć proces logowania. Jeśli nie, użytkownik jest automatycznie przekierowywany do widoku logowania (Rysunek 16.).



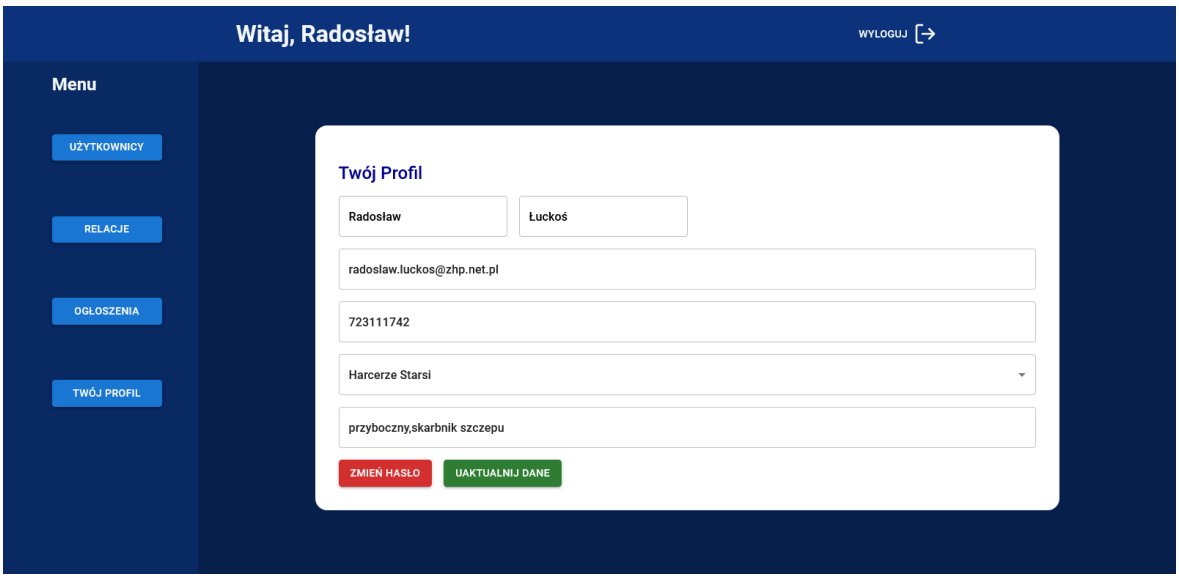
Rysunek 16. Widok logowania

Aby się zalogować użytkownik musi podać poprawny adres email oraz odpowiadające mu hasło. Po wprowadzeniu danych oraz zatwierdzeniu ich poprzez kliknięcie przycisku zaloguj, zostaje wysłane zapytanie logowania na serwer. W przypadku niepowodzenia wyświetlany jest odpowiedni komunikat (Rysunek 17.).



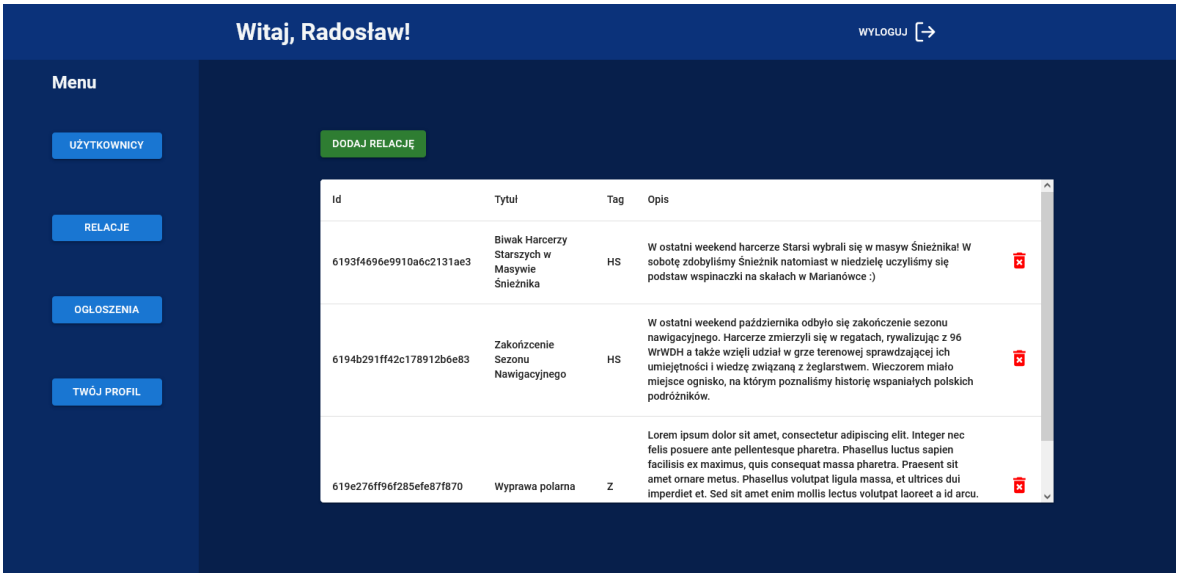
Rysunek 17. Błąd logowania

Pierwszym widokiem po zalogowaniu się jest widok profilu zalogowanego użytkownika. Jest to de facto formularz. Wszystkie jego pola oprócz imienia i nazwiska są edytowalne. Zmiany zatwierdza się klikając przycisk „Uaktualnij dane". Przycisk „Zmień hasło" powoduje dodanie do formularza pola do wprowadzenia nowego hasła.



Rysunek 18. Widok profilu zalogowanego użytkownika

W panelu dostępne są 4 zakładki. Oprócz zakładki twój profil, która powoduje przekierowanie do omówionego wcześniej widoku, znajdują się tam takie zakładki jak Użytkownicy, Relacje oraz Ogłoszenia. Każdy z widoków jest zbudowany w bardzo podobny sposób. Na Rysunku 19. przedstawiono wygląd zakładki Relacje. Odpowiada ona za zarządzanie relacjami z wydarzeń, które są wyświetlane potem na stronie z aktualnościami. Widok składa się z tabeli, w której znajdują się poszczególne relacje (wraz z możliwością ich usunięcia poprzez naciśnięcie czerwonego przycisku) oraz zielonego przycisku „Dodaj relację" który wyświetla formularz dodania nowej relacji.



Rysunek 19. Przykładowa zakładka panelu administracyjnego

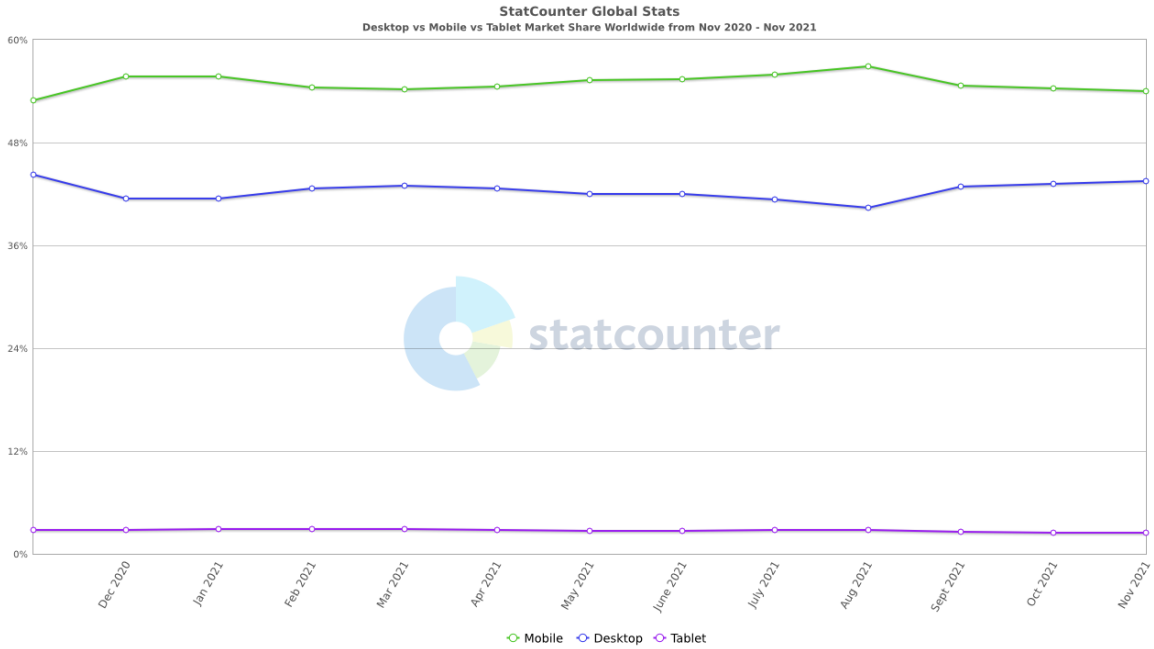
Reszta zakładek panelu administracyjnego, a więc „Użytkownicy” oraz „Ogłoszenia” zbudowane są identycznie i odpowiadają za zarządzanie kolejno użytkownikami (kontami z prawami administratora) oraz ogłoszeniami.

Oprócz tych zakładek w panelu znajduje się jeszcze nagłówek, który wita zalogowanego użytkownika oraz pozwala na wylogowanie się, po kliknięciu odpowiedniego przycisku.

5.3 Implementacja części statycznej

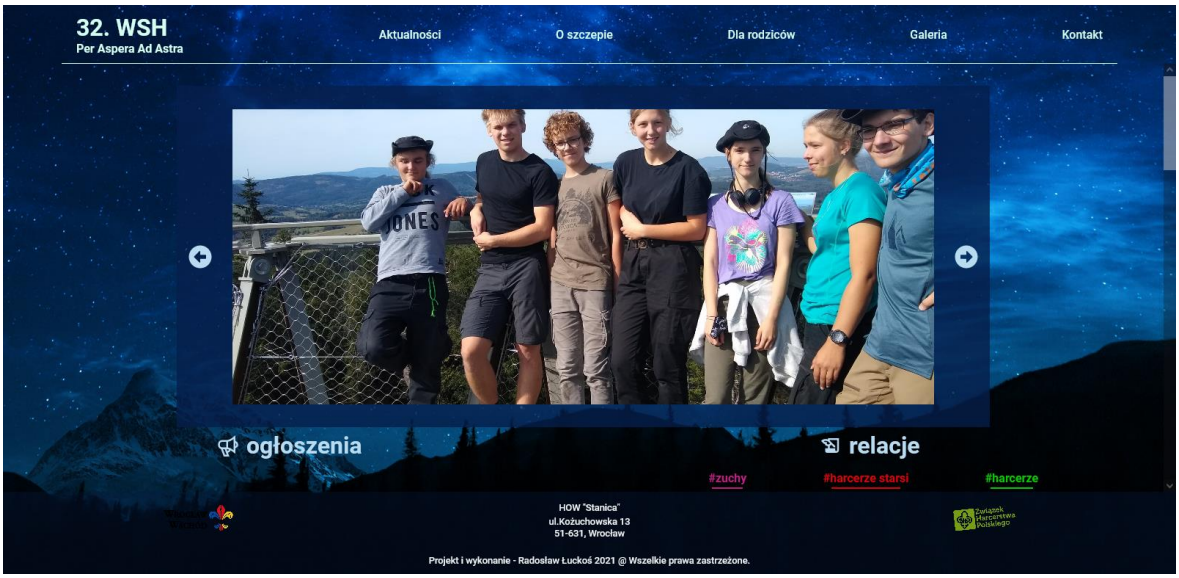
W przeciwieństwie do panelu administracyjnego interfejs graficzny ogólnodostępnej części aplikacji jest bardzo istotny. Pełni ona bowiem funkcję promocyjną i informacyjną, a co za tym idzie sposób prezentacji treści jest tutaj fundamentalny. Dlatego w tym przypadku zrezygnowano z użycia biblioteki UI na rzecz arkuszy styli pisanych ręcznie.

Bardzo ważnym aspektem implementacji interfejsu graficznego jest kwestia podejścia. Istnieją powiem dwa alternatywne podejścia. Pierwsze – desktop-first – polega na wykonaniu najpierw interfejsu dla wyższych rozdzielczości, tak by widoki prezentowały się dobrze na dużych ekranach. Następnie dopisujemy kolejne style tak by funkcjonalność strony przy wyświetlaniu na małych ekranach nie została zbyt okrojona i zachowała miarę możliwości pełną funkcjonalność.



Rysunek 20. Statystyki użycia poszczególnych typów urządzeń do przeglądania Internetu

Jednak według serwisu statcounter[12] udział urządzeń mobilnych w statystyce ruchu sieciowego wg urządzenia przekracza 50% (w listopadzie 2021 jest to już 56,46% - 53,98% dla smartfonów oraz 2,48% dla tabletów), co przedstawia wykres na Rysunku 20. To oznacza, że niewłaściwym byłoby traktowanie użytkowników urządzeń mobilnych jako użytkowników drugiej kategorii. Dlatego podejście mobile-first zyskuje na popularności. Polega ono na dostarczeniu jak najlepszego UX (ang. User Experience – do…świadczenie użytkownika) dla użytkowników urządzeń mobilnych i stopniową dalszą rozbudowę aplikacji o nowe, dodatkowe funkcjonalności wraz ze wzrostem rozdzielczości ekranu. Pozwala to na uniknięcie zdecydowanych różnic w jakości odbioru naszego interfejsu przez użytkownika i to właśnie to podejście zostało zastosowane w arkuszach stylów w tym projekcie.

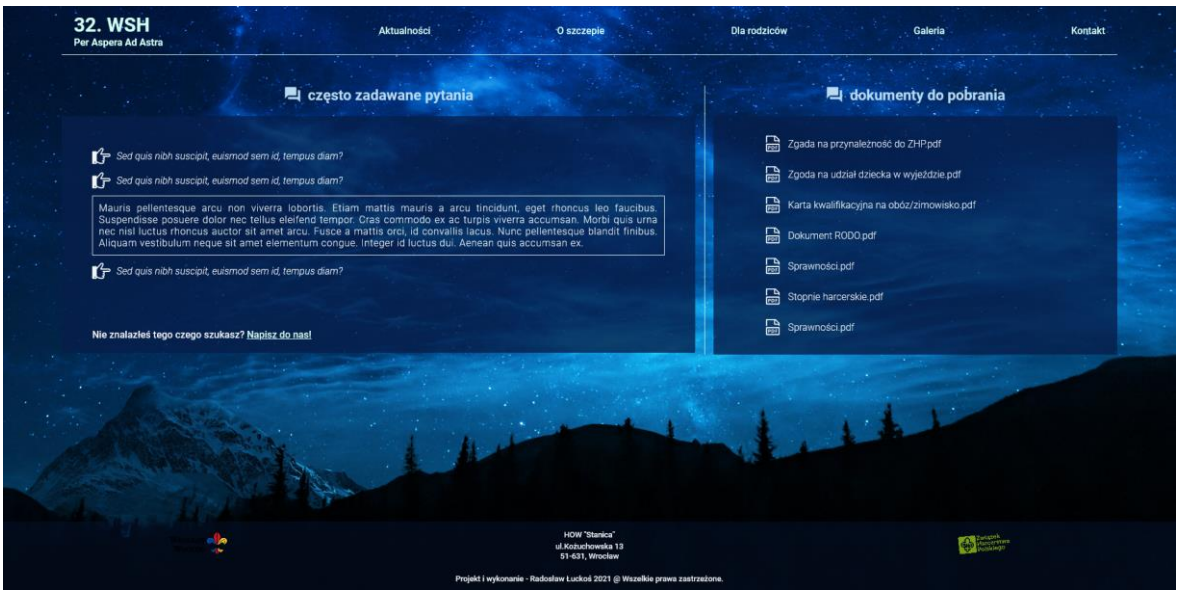


Rysunek 21. Strona główna – Aktualności

Na stronie głównej aplikacji (Rysunek 21.) miejsce na pierwszym planie zajmuje slider z wybranymi, zmieniającymi się zdjęciami. Oprócz tego znajdują się tam pobierane z bazy danych relacje oraz ogłoszenia. Relacje można filtrować ze względu na jednostkę, której ta relacja dotyczy. Domyślnie wszystkie filtry są zaznaczone. Klikając na któryś z nich, użytkownik może dany filtr odznaczyć co spowoduje ponowne załadowanie listy bez obiektów z odznaczonym hasztagiem.

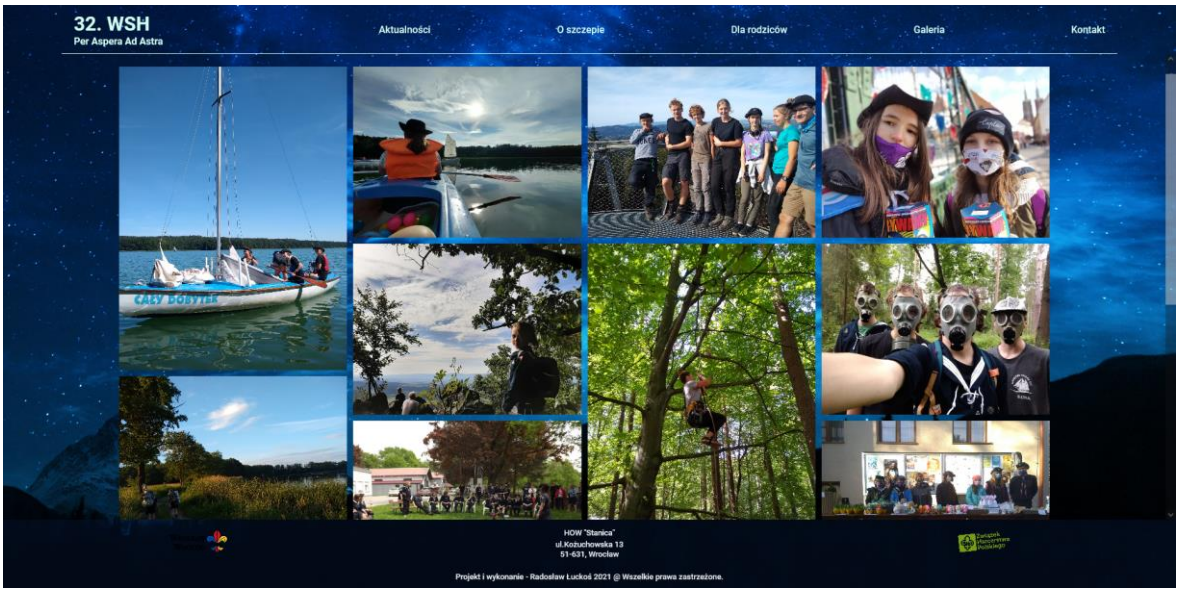
W zakładce o szczepie użytkownik znajdzie szczegółowe informacji dotyczące zarówno szczepu jak i podstawowe informacje o kadrze.

Istotną zakładką jest zakładka „dla rodziców” (Rys. 22), w której znajdują się dwie sekcje: FAQ oraz sekcja z wzorami dokumentów dla rodziców. Każde z pytań sesji FAQ można rozwinąć wyświetlając odpowiedź. Pod sekcją FAQ znajduje się odnośnik do formularza kontaktowego. Druga z sesji to lista dokumentów. Po kliknięciu każdego z nich odpowiedni dokument jest pobierany.



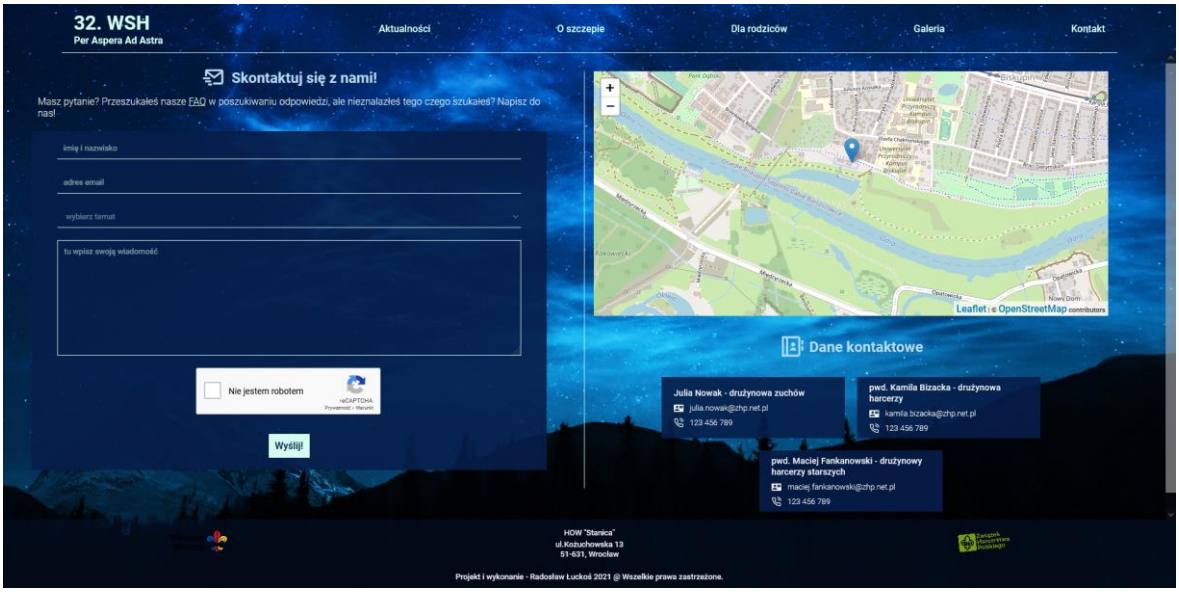
Rysunek 22. Widok podstrony "dla rodziców"

Kolejna zakładka to galeria zdjęć. Zdjęcia w galerii mają różne wymiary. Aby zachować spójność całego layoutu wykorzystano rozwiązanie typu „masonry grid”. Jest to sposób ułożenia zdjęć tak by dopasować je do wymiarów elementu HTML, w którym galeria ma się zawierać. Do implementacji tego rozwiązania wykorzystano bibliotekę macy.js, która automatycznie analizuje wymiary zdjęć, układając je tak by galeria zachowała spójność oraz atrakcyjny wygląd. Rozwiązanie to pozwala również na wdrożenie pełnej responsywności, co oznacza dostosowanie widoku galerii do aktualnej rozdzielczości ekranu.



Rysunek 23. Galeria zdjęć

Ostatnia podstrona dotyczy wszelkich form kontaktu z kadrą. Znajdują się tam dane kontaktowe do drużynowych formularz kontaktowy oraz mapa z zaznaczoną lokalizacją Harcerskiego Ośrodka Wodnego „Stanica”, na którym odbywają się zbiórki. Do integracji aplikacji z mapą OpenStreetMap użyto biblioteki React-leaflet. Istotnym jest również fakt zabezpieczenia formularza przed spamem [11] przy pomocy zabezpieczenia ReCAPTCHA. Jest to w pełni zautomatyzowany algorytm pozwalający odróżnić ludzi od botów i zabezpieczający aplikację przed spamem z ich strony.



Rysunek 24. Widok zakładki "Kontakt"

5.4 Deployment

Ostatnim etapem developmentu aplikacji jest jej deployment, a więc umieszczenie jej na serwerach, tak by była dostępna online.

Proces ten składa się z dwóch części: deploy frontendu oraz backendu. Jako pierwszy został wykonany deploy backendu. Aby operacja ta się powiodła należało dokonać następujących korekt w kodzie programu:

1. Dla zabezpieczenia aplikacji zmienne globalne takie jak adres URL do połączenia aplikacji z bazą danych lub hasło do tokena JWT umieszczono w pliku środowiskowym .env by nie był ogólnodostępny
2. Dodano nową domenę (domenę części frontendowej) do obsługiwanych przez mechanizm CORS umożliwiający współdzielenie zasobów między domenami.
3. Dodano domyślną ścieżkę „powitalną" by aplikacja na serwerze uruchamiała się poprawnie.
4. Dodano plik Procfile z określonym poleceniem, które ma zostać wywołane w momencie umieszczenia aplikacji na serwerze.

Po dokonaniu tych korekt część backendowa aplikacji została umieszczona na serwerze. Wykorzystano do tego serwis Heroku. Backend aplikacji jest dostępny online pod adresem: https://mern-32wsh.herokuapp.com/.

Przygotowanie części frontendowej do deploymentu również wymagało kilku zmian. Wymienione one zostały poniżej:

1. Zmiana adresu URL, na który wysyłane są zapytania http na adres aplikacji umieszczonej na Heroku.
2. Dodanie pliku _redirects w celu poprawienia routingu.

Po dokonaniu poprawek aplikacja została zbudowana przy użyciu polecenia **„npm build"** i wysłana na serwer. Skorzystano z usługi serwisu netlify. Aplikacja dostępna jest pod adresem https://32wsh.netlify.app/.

6.Podsumowanie i wnioski

Na podsumowanie całej pracy składają się dwa elementy. Pierwsze to krótkie podsumowanie efektów końcowych pracy. Drugą częścią są perspektywy dalszego rozwoju aplikacji.

6.1 Efekt końcowy

Efekt końcowy wielomiesięcznej pracy to aplikacja, która spełnia wszystkie początkowe założenia. Wymagania funkcjonalne oraz niefunkcjonalne zostały zrealizowane. Zarówno administratorzy jak i odbiorcy mają dostęp do wszelkich planowanych funkcjonalności.

Niektóre elementy interfejsu różnią się jednak od pierwotnego projektu graficznego. Przyczyną takiego stanu rzeczy jest nieumiejętne zaplanowanie layoutu. Proces budowy aplikacji to proces bardzo złożony. Zazwyczaj nad projektem pracuje kilka osób. Jedną z nich jest właśnie UI/UX designer, który posiada odpowiednie kompetencje, aby w odpowiedni sposób zaprojektować interfejs graficzny aplikacji.

Kolejnym ważnym dla mnie doświadczeniem było zbudowanie od podstaw REST API. Pozwoliło to poszerzyć moje pole widzenia o kolejną ciekawą gałąź web developmentu, zwiększając wszechstronność mojego spektrum umiejętności.

Dodatkowo, tworząc tę aplikację korzystałem z najnowszych technologii. Nowoczesne języki są na tyle dynamicznie rozwijane, iż ciągły proces nauki i podążania za nowościami jest w pracy programisty jest kwestią fundamentalną.

Cały proces budowy aplikacji uświadomił mi, jak ważnym jest etap projektowania jej. Efekt swojej pracy uznaję za bardzo zadowalający. Doświadczenie, które zyskałem zdecydowanie rozwinęło mnie jako web developera upewniając w przekonaniu, iż jest to droga zawodowa, którą chcę podążać. Jednocześnie widzę wiele perspektyw dalszego rozwoju tej aplikacji.

6.1 Perspektywy rozwoju aplikacji

Pierwszym kierunkiem rozwoju aplikacji jaki dostrzegam jest zdecydowane rozszerzenie panelu administracyjnego o dodatkowe funkcjonalności, takie jak:

- Dodawanie nowych pytań do sekcji FAQ
- Dodawanie dokumentów do pobrania
- Edycja większej ilości danych dostępnych na stronie internetowej szczepu

Kolejnym aspektem rozwoju aplikacji, który jest warty rozważenia jest kwestia galerii zdjęć. W tym momencie zdjęcia są statycznie załadowane na stronie a administrator by je zmienić musi prosić o pomoc developera. Dobrym rozwiązaniem byłoby wdrożenie rozwiązania pozwalającego trzymać zdjęcia w bazie danych i mieć dostęp do zarządzania nimi z poziomu panelu administracyjnego.

Ostatnim ciekawym pomysłem, który chciałbym tu wyróżnić, jest wdrożenie funkcji zdalnego zapisania dziecka na wydarzenia takie jak obozy, zimowiska, biwaki etc. Pozwoliłoby to na rozszerzenie relacji rodzic-szczep poprzez aplikacji i ułatwiłoby to wiele procedur.

7. Bibliografia

[1] Blog internetowy firmy StudioSoftware, 10 września 2020, dostępny online pod adresem: https://studiosoftware.pl/blog/aplikacja-webowa-a-strona-internetowa-poznaj-3-glowne-roznice/ (ostatni dostęp: 18.11.2021)

[2] Max Rehkopf, witryna internetowa, Atlassian Agile Coach, dostępna online pod adresem: https://www.atlassian.com/agile/project-management/user-stories (ostatni dostęp: 17.11.2021)

[3] Autorzy MDN, przewodnik MDN Web Docs, dostępny online pod adresem: https://developer.mozilla.org/pl/docs/Web/JavaScript (ostatni dostęp: 17.11.2021)

[4] Indeks TIOBE, dostępny online pod adresem: https://www.tiobe.com/tiobe-index/ (ostatni dostęp: 12.11.2021)

[5] Przemysław Ćwik, blog internetowy firmy KISS Digital, 15 stycznia 2021, dostępny online pod adresem: https://kissdigital.com/pl/blog/single-page-application-jak-dziala-spa-i-czym-sie-rozni-od-mpa (ostatni dostęp: 18.11.2021)

[6] Mikołaj Żywczok, blog internetowy firmy Just Join IT, 21 listopada 2019, dostępny online pod adresem: https://geek.justjoin.it/zalety-wady-reactjs (ostatni dostęp: 12.11.2021)

[7] Dokumentacja Frameworka Express.js, dostępna online pod adresem: https://expressjs.com/ (ostatni dostęp: 10.12.2021)

[8] Jarosław Żeliński, blog IT-Consulting, 4 marca 2021, dostępny online pod adresem: https://it-consulting.pl/autoinstalator/wordpress/glossary/baza-dokumentowa/ (ostatni dostęp 10.12.2021)

[9] Blog internetowy devszczepaniak.pl, 25 sierpnia 2018, dostępny online pod adresem: https://devszczepaniak.pl/wstep-do-rest-api/ (ostatni dostęp: 5.12)

[10] Blog internetowy ioerror.pl, 10 czerwca 2014, dostępny online pod adresem: https://ioerror.pl/post/przechowywanie-hasel/ (ostatni dostęp: 2.12.2021)

[11] Lindsay Liedke, 28 lipca 2019, blog internetowy kaliforms, dostępny online pod adresem: https://kaliforms.com/blog/form-spam/ (ostatni dostęp: 25.11)

8. Wykaz rysunków