**WYŻSZA SZKOŁA**

**INFORMATYKI i ZARZĄDZANIA**

**z siedzibą w Rzeszowie**

**WYDZIAŁ INFORMATYKI STOSOWANEJ**

**Kierunek: INFORMATYKA**

**Specjalność: Programowanie**

Oleksandr Lisivka

Nr albumu 53410

***Serwis oglószeń do podroży***

Promotor: dr Marek Jaszuk

**PRACA DYPLOMOWA INŻYNIERSKA**

**Rzeszów 2018**

Ja niżej podpisany/a oświadczam, że składana przeze mnie praca dyplomowa pt. „...........................” została przygotowana samodzielnie.

Oświadczam również, że praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna ze złożoną wersją elektroniczną.

.........................................................

data czytelny podpis autora

Oświadczam, że niniejsza praca została przygotowana pod moim kierunkiem i stwierdzam, że spełnia ona warunki do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

................................................................

data czytelny podpis promotora

Оглавление

[**Wstęp** 5](#_Toc509535151)

[**Uzasadnienie wyboru tematu pracy** 5](#_Toc509535152)

[**Cel pracy** 6](#_Toc509535153)

[**Aktualny stan wiedzy dotyczący problemu (zagadnienia) będącego przedmiotem pracy** 6](#_Toc509535154)

[**Zakres pracy** 6](#_Toc509535155)

[**Omówienie układu pracy, krótka charakterystyka poszczególnych rozdziałów** 7](#_Toc509535156)

[**Część główna** 8](#_Toc509535157)

[**Część teoretyczna** 8](#_Toc509535158)

[**Front-end** 8](#_Toc509535159)

[**HTML5** 8](#_Toc509535160)

[**CSS3** 8](#_Toc509535161)

[**SCSS** 9](#_Toc509535162)

[**Bootstrap** 9](#_Toc509535163)

[**JavaScript** 9](#_Toc509535164)

[**JQuery** 9](#_Toc509535165)

[**Back-end** 10](#_Toc509535166)

[**Node.js** 10](#_Toc509535167)

[**Npm** 10](#_Toc509535168)

[**REST API** 11](#_Toc509535169)

[**Baza danych** 11](#_Toc509535170)

[**Relacji pomiędzy danymi** 11](#_Toc509535171)

[**Git oraz Github** 11](#_Toc509535172)

[**Visual Studio Code oraz Sourcetree** 12](#_Toc509535173)

[**Część praktyczna** 13](#_Toc509535174)

[**Specyfikacja wymagań** 13](#_Toc509535175)

[**Wymagania funkcjonalne** 13](#_Toc509535176)

[**Wymagania niefunkcjonalne** 13](#_Toc509535177)

[**Diagram przypadków użycia** 14](#_Toc509535178)

[**Mockup** 14](#_Toc509535179)

[**Git** 14](#_Toc509535180)

[**Tworzęnie aplikacji** 15](#_Toc509535181)

[**Front-end** 15](#_Toc509535182)

[**JavaScript** 15](#_Toc509535183)

[**CSS** 16](#_Toc509535184)

[**Back-end** 16](#_Toc509535185)

[**Npm pakiety** 16](#_Toc509535186)

[**body-parser** 16](#_Toc509535187)

[**connect-flash** 17](#_Toc509535188)

[**ejs** 17](#_Toc509535189)

[**express** 17](#_Toc509535190)

[**express-session** 18](#_Toc509535191)

[**method-override** 18](#_Toc509535192)

[**moment** 18](#_Toc509535193)

[**mongoose** 18](#_Toc509535194)

[**nodemon** 19](#_Toc509535195)

[**passport** 19](#_Toc509535196)

[**passport-local** 19](#_Toc509535197)

[**passport-local-mongoose** 19](#_Toc509535198)

[**FILL ME google map module** 19](#_Toc509535199)

[**Routing** 20](#_Toc509535200)

[**Baza danych - Mongoose NONSQL database** 20](#_Toc509535201)

[**Relacji pomiędzy tabelami** 20](#_Toc509535202)

[**Autoryzacja** 20](#_Toc509535203)

[**Authentyfikacja oraz Middleware** 20](#_Toc509535204)

[**Tok projektu** 21](#_Toc509535205)

[**Sprawdzenie działania aplikacji** 21](#_Toc509535206)

[**Demonstrowanie alikacji i jej działania** 21](#_Toc509535207)

[**Napotykane problemy** 21](#_Toc509535208)

[**Możliwości do przyszłego rozwinięcia aplikacji** 21](#_Toc509535209)

**Wstęp**

W raz z wynalezieniem internetu wsiat się zmiania i się rozwija co raz szybciej. Cały świat jest powiązany z każdą jego częścią za pomocą połączenia internetowego. Jednosześnie z tym w każdym momencie mająć dostęp do sięci również masz dostemp do wszystkich jego serwisów ułatwiających życie. Komunikacja od czasów jej wynależenia w życiu codziennym odgrywa najważniejszą role polączająć ludzi. Do wynależenia internetu ludzie komunikowały się najpierw ustnie póżniej pisemnie i już mogły przekazywać istotną dla nich informację na odłegłość. Progres ze swojej koleji w metodach przekazywania informacji wystąpił odnośnie nie tak dawno licząć od początkó istorii. I teraz mająć za sobą podgruncie komunikacja ma możliwość lączyć nie tylko osobnych ludzi ale i kompanie i nawet urądzenia (Internet of things). Praca i byt codzinny coraz większej ilości osób i przedsięmbiorstw jest związany za pomocą tego cuda - globalnego systemu połączeń sieciowych. Jednocześnie z tym ilość uzależnionych od internetu też co raz rośnie ale to nie jest niczym dziwnym kiedy idziesz w nogę z czasem. Komputery, laptopy, telefony, zegarki, stacji gier, konsole i wiele innych rzeczy komunikują się z innymi rzeczami czy nawet serwarami i ilość danych codzięnnie rośnie.

W dniach dzisięjszych żeby dostać to czego chcesz nie potrzebujesz dużo. Również jak i komunikowanie sprawa wymiany czy póżniej sprzedaży rzeczy lub usługi pojawiła się bardzo dawno a teraz ma swoj kolejny pik. Teraz żeby kupić rzecz nie koniecznie musisz iść gdzieś i jej szukać, wybierać czy porównywać. W ostatnich latach się rozwija sprzedaż za pomocą internetu. Sklepy, serwisy internetowe, salony, fabryki i temy podobne rzeczy można odnaleść na stronach WWW(World Wide Web).Co raz więcej towarów i usług dostępne w postaci elektronicznej. Dość prosty i zrozumiały z ekonomii wyraz: popyt powoduje propozycję. Tak i z serwisami internetowymi. Popularyzacja serwisów internetowych powoduję powiększenie ich ilości. Prosty i zrozumialy wygląd tranzakcji znajomy wszystkim ze swiatu rzeczywistego został przeniesiony do internetu. Teraz każdy użytkownik może występować jako sprzedawca czy jako konsument. Ze swojej strony to powoduję przyrost serwisów w każdej branży i dziedzinie.

Dla przykładu żeby teraz dostać się z punktu A do punktu B wystarczy pobrać czy otworzyć mapę i jechać za marszrutem z listy proponowanych. Albo wyszukać i wybierać samochód co jedzie za potrzebnym tobie kierunkiem i umawiać się z kierowcem żeby oczekiwał że pojedziesz z nim, kiedy i gdzie jest z min spotkanie. Jeździć że przy pomocy koleji też nie jest tak trudno. Dane o przyjazdach, wyjazdach i cenach są dostępne każdemu i w każdej chwili w internecie. Loty samolotem też są podobne. Wpisująć miasto wylotu oraz docelowe zostaniesz proinformowany o rejsach i kierónkach za jakimi oni lecą. Kupujesz bilet i jesteś gotowy do podróży – wsiadasz i lecisz. Na podwtawie tych danych teraz dzialają biura podróży. Mająć dostemp do danych o wylotach czy odjazdach i przyjazdach busów czy innych środków transportu wybierają skąd i dokąd i rezerwują hotel dla wybranej osoby na wybrany okres czasu. Czyli podają informacje hotelowi który może się znajdować na przeciwnej stronie kuli zięmnej. I oni mająć taki komunikat już oczekują że w czasie rezerwacji przyjedzie wybrana ilość osób.

Są również serwisy na których można zarezerwować miszkanie w określonym terminie albo orendować pokój. Jest to dość prostym i użytecznym kiedy planujesz sobie wyjazd do innego miasta. Takie serwisy są dobrą zamianą rezerwacji hotelów kiedy nie planujesz albo nie masz możliwości wydawać dużo za nocleg w innym miaście czy panstwie. Ale czy są serwysy pozwalające do rezerwacji kulturnej programy wycieczki. Jęzeli porównywać z hotelami to są serwisy gdzie być mógł to zrobić ale na przykład jako dodatek od biura podroży. Serwis na kórym można by było samemu powiedzieć czego szukasz i zaplanować to wspolnie z kimś albo dołonczyć się do kogoś w już zaplanowanej podróży, czy zarezerwować wobie wycieczka po mieście u tego kto ją proponję jeszcze niema.

**Uzasadnienie wyboru tematu pracy**

Potrebności na rynku rosną co raz szybciej. Dowodem tego jest każdy popujarny produkt. Czasami żeby dostać produkt jednym z pierwszych ludzie stoją w kolelce dniami i noczami. Żeby nie przegapić moment jakiegoś zapotrzebowania na rynku potrzebno cały czas myślić na przód. Serwis dla podrożujących gdzie użytkownicy by moli znaleść kogoś do wspolnego wyjazdu czy żeby znalść przewodnika po mieście do którego jadą był by bardzo przydatny podrożującym. W drugą stronę ludzi by moli sami dodawać ogłoszenia i proponować innym się do nich dołonczyć. W danym momencie rynek jeszcze nie ma podobnego produktu w swojej branży, temu on by był wybrany jako temat mojej pracy dyplomowej.

**Cel pracy**

Celem pracy było by stwożyć serwis dla pomocy podrożującym na którym oni by mogli dodawać ogłoszenia o swoich planach na wyjazd do którego potrzebna jest większa kompania lub dodania komentaży w odpowiedź do danego ogłoszenia.

Praca w swoją kolej była by podzielona na stwożenie stron internetowych do danego serwisu oraz web serwisu po stronie którego wykonywałybysię działania na stronach. Serwis powinienen zawierać możliwości logowania, dodowania ogłoszeń, edytowania ogłoszeń i ich usuwania przeż osobe co ich dodała. Komentaże równiż piwinni mieć podobną funkcjonalność. Strony internetowe powinne być responsywne i mieć prostą strukture żeby użytkownicy się w niej nie gubili oraz żeby strona nie była przepęłniona kontentem który był by trudnym w ogarnięciu (nie był by nieżbędnym)dla wykonania konkretnego działania na stronie(User Experience).

**Aktualny stan wiedzy dotyczący problemu (zagadnienia) będącego przedmiotem pracy**

Przed rozpocięciem danej pracy ja(Oleksand Lisivka) miałem doczynienie ze stronami internetowymi podczas praktyk zawodowych od uczelni. Pod czas danych praktyk nauczyłem się i pogłembiłem poznania w części Front-end’owej, czyli z HTML5, CSS(SCSS, SASS, Bootstrap), JavaScript(jQuery). Odnośnie wiedzy niezbędnej do napisania części serwerowej w wybranym przezemnie języky(JavaScript w środowisku Node.js), to takiej wiedzy nie posiadałęm. Na uczelni nauczyłem się języka C# i wzorzeca architektoniczny służącego do organizowania struktury aplikacji MVC i mółbym wykonać pracę na podstawie danej więdzy ale wybrałem inny język żeby ciągle uczyć się czegoś nowegoktóry i aby się rozwijać ponieważ w dziedzinie Technologij informacyjnych języki, techniki, standarty, i technologie ciągłe się zmieniają i potrzebno iść razem z nimi. Typ dazy danych (NONSQL) był wybrany z tego samego powodu - żeby powiększyć więdze profesjonalną.

**Zakres pracy**

Cały system można podzielić na części które muszą być zrealizowane w kolejności w której muszą być dodawane do serwisu:

1. REST Routing
2. Tworzęnie stron
3. Twożenie ogłoszenia
4. Konfigurowanie bazy danych i podpinanie jej do aplikacji żeby dać możliwość zapisywania danych do niej
5. Dodawanie komentarzy
6. Logowanie
7. Autentyfikacja
8. Edytowanie lub usunięcie ogłoszenia przez go wsaściciela
9. Edytowanie lub usunięcie komentaży przez ich wsaścicieli
10. Wyświetlenie komunikatów dla użytkownika oraz rozszerzenie istniejącego funkcjunału
11. Responsywność stron do różnych rozszeżeń ekranu.
12. Podtzymywanie różnych wersii przeglądarek internetowych.

Dana lista swkazuje na kolejność realiowanych zadań ze wzgłedu na ich ważność dla realizacji danego projektu. Idąć z góry do dołu tworzymy system od śkieletu do gotowego produktu.

Na przykład żeby każdy użytkownik mógł by utworzyć ogłoszenie on musi najpierw przejść na odpowiednią strone i się zalogować. Z punktu widzenia serwisu bez logowania dla wykonania rządania do serwera z informacją o ogłoszeniu iżytkownik musiał by przejść się przez 4 pierwsz części z listy, czyli otwożyć strone pod potrzebnym adresem(REST Routing). Po tym użytkownikowi musi wyświetlić się strona z treścią która odpowiada adresowi(Tworzęnie stron). Uzytkownik tworzy odpowiedznie rządanie do serwera z treścią ogłoszenia i po stronie serwera po przetwarzaniu danych musi się utworzyć ogłoszenie(Twożenie ogłoszenia oraz Konfigurowanie bazy danych i podpinanie jej do aplikacji żeby dać możliwość zapisywania danych do niej). Dalej dla rozszeżenia funcjanalności dodajemy po stronie serwera odpowiednie ścieżki routingu i strony co by się renderowały po przejściu na nich dla dodawania komentaży i ścieżki do oczekujące na wchadzące od użytkownika dane o komentazu(Dodawanie komentaży). Więcej szczegulowo informacja zostanie podana w kolejnych rozdziałach pracy.

**Omówienie układu pracy, krótka charakterystyka poszczególnych rozdziałów**

Praca dla latwości jej przetważania zostanie podzielona za treścią na części (np. część wstępu, część główna pracy, zakończenie, literatura, streszczenie pracy, załączniki.). Wstęp oraz część glówna zostaną podzielone na działy(np w przypadku części głuwnej na część teoretyczą oraz część praktyczną). Żeby działy ze swojej koleji zostały się więcej formatowanymi żeby odrazu więdzieć do częgo należy część pracy w uśczegulonych poddziałach została by podana śczegulniejsza informacja(np. część frontendowa część backendowa).

Charakterystyka nie potrzebuję śczegulnego omusienia z tego powodu że o treści każdej z części było by wiadomo odrazu z miejsca w którym ona się znajduje(np. Część teoretyczna -> Front-end -> Html5 miała by w sobię teorije o HTML5 co się odnosi do części frontendowej).

**Część główna**

Dany rozdział bezpośrednio odpowiada za opisanie nażędzi wykorzystanych przy tworzeniu danej pracy oraz za przebieg pracy pod czas jej realizacji od początku do konca. Z tego powodu została ona podzielona na 2 działy różnące się pomiędzy sobą.

**Część teoretyczna**

Poniżej są opisane tehnologie oraz narzędzia które zostały użyte przy napisaniu danej pracy dyplomowej. Niektóre są wymagane dla jej realizacji(np. HTML5 jest teraz standartem w napisaniu stron internetowych i pisać strone dla przykładu na tabelach nie było by dotrzymaniem się standartów) a niektóre zostały wybrane ze wzgłendów na łatwość w użyciu oraz możliwości które daję danę nażedzie, ciągle wsparcie i rozwijająnie(np. Bootstrap, jQuery czy npm).

Dwie najważniejsze części projektu to oczywiście front-end i back-end. Front-end odpowieda za komunikowanie się z użytkownikiem, pobieranie danych od użytkownika, czy ich wyświetlenie oraz przekazanie ich do back-endu. Back-end jest mózgiem i sercem aplikacji ponieważ w nim na podstawie danych wykonują się określone zadania. To nie znaczy że front-end jest mniej ważny. Dużo czego można zrobić za pomocą JavaScripu ale żeby nikt nie mał na to wpływu i żeby to było chroniono od innych żeczy ważniejszę są zaimplementowane po stronie serwera. Front-end w tej części występuję jak pyszy deser go głównego dania czym jest część serwerowa, jako graficzna reprezentacja dla aplikacji konsolowej. Taki typ podzielenia obowiązków set dość typowym i popularnym pośród oprogramowania danego typu. Tak podzielona praca daje osobie tworzącej strone wolne rzęce i jednocześnie dataa osoba nie koniecznie musi znać cech serwera i co się na nim dzieje, a programista serwera nie potrzebuje znajomości HTML i CSS bo zajmuję się nie wyglądem a tylko funkcjonalnością.

**Front-end**

Fornt-end jest częścią aplikacji która odpowiada za komunikację z użytkownikiem przy pomocy interfejsu graficznego. Dość często front-end jest używany jako pośrednik pomiędzy użytkownikiem systemu a aplikacją na serwererze. W systemach takiego typu front-end posługuje dla wyświetlenia informacji oraz pobierania informacji od użytkownika. Dość proste strony czy np. emaili mogą nie mieć serwera czy w przypadku emaili nawet scryptów. Takie strony wykorzystywane do reprezentacji danych. Są również i typy stron pomiędzy dwoma powyższymy. Takie strony nie są zbyt skomplikowane żeby niezbędny był serwer ale i nie są statyczne. Takie strony napisane są przy pomocy języka JavaScript dla dodawania interaktywności na strony. Więczej szczęgulowo każdy z elementów jest opisany poniżej.

**HTML5**

HTML albo ang. HyperText Markup Language jest językiem znaczników do markowania, strukturyzacji i prezentowania treści w Internecie. Cyfra 5 w nazwie wskazuję na aktualną wersję – piątą, dlatego wykorzystanie języku ze standartami ostatniej wersji zostało nazwane HTML5 co połączyło w sobie normy i standarty HTML oraz XHTML. HTML był stworzony żeby opisać strukturę informacji zawartych wewnątrz strony internetowej w postaci modelu oiektowego(DOM). HTML z tekstu który ma buduję hiperłącza, akapity, nagłówki i temu podobne rzeczy. Bez dodawania CSS strona z otwierających i zamykających się tegów ma swoją strukture ale nie ma żadnego wyglądu. Jest to „ściana tekstu” dla maszyny żeby ona wiedziała kiedy jedna go część się kończy a kolejna się zaczyna.

Struktura modelu DOM (ang. Document Object Model) przedstawia dokument HTML w postaci struktóry drzewa. Model DOM definiuje obiekty i właściwości wszystkich elementów HTML oraz metody dostępu do nich. Korzystając się z DOMu przy pomocy JavaScriptu można łatwo dodać zmienić lub usunąc element.HTML.

**CSS3**

Kaskadowe arkusze stylów(Cascading Style Sheets) jest językiem służącym do opisywania wyglądu, czyli wyświetlenia strony internetowej. CSS jest listą regul które wskazują przeglądarce w jaki sposób dana strona musi być wyświetlona i jak wyglądają wybrane elementy. CSS jest używany dla określenia kolorów, czcionek i innych aspektów wizualizacji stron internetowych. Był on stworzony żeby trzymać osobno strukture dokumentu a jego stylowanie. Z tego powodu strony na których stylowanie jest wpisane liniowo do kodu HTML musą być dawno napisane i tego się nie zmienia żeby trzymać kod jednolitnym. Czytać takie strony staje drudno kiedy przykladowo na jedą linię HTML przychodzi się po 3 stylowania. Również użycie klas pomogło by trzymać kod suchym i nie powtarzającym się. Osobny plik ze stylowaniem może być użyty na różnych stronach co ułatwia o wiele ich napisanie.

**SCSS**

FILL ME

**Bootstrap**

Bootstrap(1) jest frameworkiem dla CSS, stworony przez programistów Twittera. Glówną cechą bootstrapu jest responsywność oraz już gotowe klasy które w latwy i szybki sposób pomagają przestylować stronę internetową. Bootstrap napisany jest w taki sposób żeby strona również dobrze wygłądała w większościi najpopularniejszych przeglądarek, takich jak Google Chrome, Mozilla Firefox, Internet Explorer, Edge, Opera i Safari. Bótstrap jest łatwy w użyciu i pomaga zaośczędziś dużo czasu ns stulowaniu stron. Dla działania danego frameworku niezbędnym jest wykorzystanie biblioteki jQuery. Dodając trzy referencje do swojego kodu (pierwsza do stylowania, druga do jQuery i trzecia do scryptów Bootstrapowych) do pobranych plików lub z CDN serwisów można używać go klas w każdej chwili.

**JavaScript**

JavaScript jest wieloplatformowym objekto orijentowanym językiem scryptowym i jest trzecim ale nie za ważnością tytanem front-endu. JavaScript jest bardzo wielofunkcojalnym poniważ jest wykorzystywany w przeglądarkach, po stronie serwera(NodeJS), mobilnych ta desktopowych aplikacjach. W przeglądarkach JavaScript występuję jak na stronach tak i w web rozrzerzeniach. Strona bez wykorzystania JS wygląda statychną i nie zmieniającą się ale z dodawaniem JS wzsystko się zmienia. Dla przykładu JS jest używany do stworzenia scenariuszy za którymi strona się zachowóję co daje możliwość na stronie współdziałać z końcowym użytkownikiem na go użądzeniu. Również dobrze i często wykorzystywany do walidowania formólarzy na stronie, wymieniać się danymi z serwerem, zmieniać struktóre ta widoki stron.

Czasami przy stylowaniu strony potrzebno wybrać jakieś elementy ze strony ale stylowanie przez CSS nie daje takich możliwości do przestylowania ich i JS jest jedynym wyjściem. Z drugiej strony tak jak HTML CSS oraz JavaScript są różne za swoim znaczeniem warto by przymać ich osobno o ile się to daję. Niestety nie wszystkie przegliądarki dzialają jednakowo i temu jak i niektóre stylowania tak i niektóre scrypty mogą nie działać wszędzie, temu przy napisaniu kodu warto uważać na przęgądarki i ich werję. Większość problem można rozwiązać prototypowaniem (JS jest językiem z koncepcją prototypów temu podtzymowanie objektów jest takie samo i się nie różni od tradyctjnych języków OOP) Takie problemy są z czasem rozwiązywanę po przez standartyzację.

Standartem ostatniej wersji JS jest ECMAScript wersji 6. ECMAScript jest ustandaryzowany przez ECMA obiektowy skryptowy język programowania. Specyfikacja go znana jest jak ECMA-262. ECMA-262 definiuje semantykę języka ECMAScrpt(ktrego implementacją jest JavaScript oraz innę), oraz np. podstawowe typy danych (String, Boolean, Number, Object itd.) i obiekty (np. Math, Array).

**JQuery**

JQuery(2) jest szybką, niewięlką i bardzo popularną biblioteką języka JavaScrip. Powiedzieć popularna przy okraśleniu może nie wystarczyć ponieważ jQuery jest najpopujarniejszą biblioteką języka JavaScript i wykorzystuję się na ponad polowie najpopularniejszych stron internetowych. Pomaga ona w wybieraniu objektów DOMu oraz go manipulacją. Animajca, tworzenie eventów i ich obsługa jest też o wiele latwiejsza z jQuery. Ilość kodu napisanego z jQuery jest mniejszą, temu i latwiejszą w go czytaniu. Nie koniecznie zawsze potrzedno używać jQuery ale licenzja MIT pozwala każdemu używać danej biblioteki za darmo w każdej chwili.

**Back-end**

Back-end na przeciwnictwo front-endowi nie działa po to żeby kontaktować się z użytkownikiam tylko po to żeby generosać kod maszynowy dla procesorów z różnymy przyznaczeniami. Front-end wytępuje pośrednikiem pomiędzy back-endem a użytkownikiem koncowym, temu programista części fronentedowej jest odpowiedzialny za wyróżnienie aplikacji wykonanej przez programistę części backendowej. Dla przykładu mająć do porównania dwa serwisy internetowe dla sprzedaży back-endowe części mogły by mieć wiecej wspólnego pomiędzy sobą w kodie źródłowym aniż frontendowe. Oczywiście back-end mógł by być napisany w różnych językach i z użyciem różnych platform ale funkcjonalność napisana po stronie back-endu miała by większe podobięstwo aniż wygląd po stronie front-endu. Tak jak front-end dostępny każdeny użytkownikowi jakiejś strony to back-end odpowiadający za działalność całej aplikacji, i jej funkcjonal jest ukryty od większości. Przyczyn na to jest wiele od tego że korzystająć się ze strony nie potrzebujesz wiedzieć co się dzieje „pod spodem” takiej strony żeby nadał normalnie jej używać kończąć polityką bezpieczęśtwa. Końcowy użytkownik nawet nie zawsze może wiedzieć że coś się zmieniło po stronie serwera, dla przykładu kiedy w 2011 Twitter zmienieł swają część back-endową z Ruby on Rails na Java końcowe użytkownicy mająć dotyczenie tylko z front-endem nawet nie zauważyli różnicy.

Programowanie back-endowe to programowanie, którego celem jest implementacja strony serwera integracja bazy danych i połączenie jej ze stroną użytkownika(front-end). Mówiąc najprościej, frontend przesyła informacje i polecenia od użytkownika do serwera, który z kolei je przetwarza.

**Node.js**

Node.js(3) jest to środowisko uruchomieniowe zaprojektowanedo twożenia aplikacij internetowych napisanych w języku Javascript po stronie serwera. Działa ona na Chrome V8 silniku z otwartym kodem od Googla który również jest wykorzystywany w Google Chrome. Jest to szkielet do budowy aplikacji. Jako Framework albo platforma programistyczna definiuje on bazową strukturę aplikacji oraz ogólny mechanizm jej działania. Node.js ma w sobie zestaw komponentów i bibliotek do wykonywania – npm(node pachage manager). Programista tworząc aplikację, dostosowuję już gotowe pakiety do wymagań realizowanego projektu, tworząc w ten sposób gotową aplikację.

Node.js ma architekturę sterowaną zdarzeniami, zdolną do asynchronicznych operacji wejścia / wyjścia(I / O). Taka arhitektóra ma na celu optymalizację skalowalności i efektywności w aplikacjach internetowych z wieloma operacjami wejścia / wyjścia, również w aplikacjach internetowych działających w czasie rzeczywistym.

Badania robione przez Stack Overflow pokazują że w 2017 za rezultatami danych badań JavaScript jest najpopulrniejszym językiem programistycznym(4), a w kategorii Platformy, biblioteki oraz inne technologie wygrywa Node.js(5). Czyli Node.js na dzień dzisięjszy jest najpopularniejszą wśrod deweloperów technologią. JavaScript przy tym że co roku jest coraz więcej popularniejszy wspiera tym samym i Node.js w go popularności. Netflix. PayPal, Yahoo!, Walmart i wiele mniej wiadomych w strefie technologii firm wykorzystują Node jako korporacyjna platformę internetową.

JavaScript występując w większości aplikacji internetowych jest bardzo popularnym. Z tego powodu uniwersalność kodu ze wzgłedu na technologie jest ważnym plusem korzystania z JS i na serwerze, o którym należy pamiętać.

Pojawienie się asynchronicznej oraz oczekującej konstrukcji(async / await) całkowicie zmieniło podejście do pisania asynchronicznego kodu bo taki kod za swoim działaniem przypomina zwykły kod synchroniczny, zarówno pod względem wyglądu, jak i zachowania. Pomaga to w rozwiązaniu problemu zwanego callback hell kiedy w oczekiwaniu na wykonanie rządania musimy napisać kod który zarównie przewiduwuję porząke oraz powodzenie rządania, i ile więncej jest takich rządań asynchronicznych jedna po drugiej, tym więcej rośnie i gromadzi się kod związany z ich obsługa.

**Npm**

Node.js ma domyślny manager pakietów napisany całkowicie w języku JavaScript dla używania przez każdego w każdej chwili. Npm(6) jest największym na tą chwile zgromadzeniem pakietów w sieci internet. W tym momęcie są dostępę ponad 470 000 bezopłatnych pakietów. Taka ilość pakietów kiedy chcesz coś zbudować pierwsze aplikacje może przeszkadzać bo wybór każdego komponentu do swoich potrzeb i do dopasowania ich z innymi komponentami też potrzebuję czasu.

Żeby dla przykladu zbudować REST API serwis z routingiem potrzebny jest jeden dwa pakiety. Im więcej funkjanalności zostaję dodao do tego tym więcej pakietów z gotowymi pakietami js zostają dodane.

**REST API**

RESTfull Web Api(Applikation Programming Interface)(7) jest aplikacją, która może być manipulowana przy użyciu protokołu HTTP i formatu JSON lub XML. Pomaga to w żeby powiązać między sobą różnę części araz różne aplikacje. Web api pomaga w kontaktówaniu się web intefejsam i aplikacjąm pomiędzy sobą. Protokół HTTP zawiera metody dla manipulacji typu : „GET”, “POST”, “PUT”, “DELETE”, które są majwaznięszymi. Arhitektóra REST odnośnie krórej była zrobiona dana praca bazująć się na danych metodach pozwala na manipulowanie po przez routing.

Roouting wygląda następująco:

|  |  |
| --- | --- |
| Akcja | HTTP Metod |
| INDEX | GET |
| SHOW | GET |
| CREATE | POST |
| NEW | GET |
| EDIT | GET |
| UPDATE | PUT |
| DESTROY | DELETE |

Każda akcja ma swoję znaczenie które jak można się domyślić z nazwy odpowiada za:

Wyświetlenie wsystkiego, konkretnego ekzemplaża, twoznie nowego objektu, pokażywanie formy dla do twożenia, pokażywanie formy dla zmiany objektu, go zmiana i ostatnie – go usunięcie.

**Baza danych**

Bazą danych jest zbiur danych zapisany za pełnymi regulami. Jest to gromadzenie danych i istnieje to od dawna - przepisy osób do obremu mieszkaniowego czy liczenie towaru co został dostarczony z innego brygadą morską po portu ma dużo wsólnego a mianowicie regulowanie tych danych. Mająć liste produktów dostarczonych moglibyśmy zobaczyć nazwe produktu, ilość może jeczę cene, datę dostawy i skąd pochodzi, i tak dalej. Zapis danych z pojawą komputerów się zmienił i nie są to już książki czy osobne kartki tylko dane w postaci cyfrowej. Również dobrze można by było zapisywać danę i na żółtych karteczkach tylko czu jest to efektywno?

Pojawienie się baz danych, tabel i połaczeń pomiędzynimi ułatwia póżniejszy dostęp do tych danych. Np. każty rekord, czyli pojedynczy element ma 4 pola przechowujące dane o użytkowniku - imie, nazwisko, adres i miasto. Twożenie osobnej tabeli z miastamy ustandartyzowało by wybór miasta i póżniejsze odczytanie danych np. wszystkich mieszkańców miasta za wyborem za daną kryterią.

**Relacji pomiędzy danymi**

Relację pomiędzy tabelami mogą być 2 typów, jeden do jednego albo jeden do wielu. Różnią się połaczeniami pomiędzy danymi. Np. do jedenej osoby jęst przypisany jeden numer indefikacyjny platnika i również pod jednym numerem platnika istnieje tylko jedna osoba( 1 - 1) albo inny przykład jedna osoba jest przypisana do jednego miasta, ale w danym miastu jest dużo mieszkańców(1 - \*).

**Git oraz Github**

Git jest systemem kontroli wersij który ułatwią życie każdemu kto się z niego kożysta. Git pomaga zapisywać wszystkie wersje plików z odniesięniem do każdej zmiany w pliku. W każdym momencie maż dostemp do każdej swojej, podtwierdzonej zapisywaniem do gitu, zmiany wprowadzonej do plików. Programista mając problemy z kompilowaniem kodu po przez wprowadzony błed może zobaczyć jakie zmiany byli wprowadzone do danego kodu. Pracująć w zespole git, mając odgałęzienie, pomaga twożyć niezależnę cześci projektu wspólnie a przy zapisywaniu i wysyłaniu zmian każdy by mógł je odebrać i mieć do użycia. Późniel gałęzięnia można połączyć razem alo cofną zmiany w kodzie do poprzedniej wersji. Użycie gitu pozwala latwo śledzieć za zmianami co się zdażyły w plikach i ich nazwach.

Github jest usługą internetową, co działa na podstawie systemu kontroli wersji, za pomocą której można przechowywać i łtwo się dzielić repozytoriami z innymi. Można na nim twożyć prywatne oraz publiczne projekty do których mieli by dostęm albo wybrane osoby albo wszyscy chętni.

**Visual Studio Code oraz Sourcetree**

Visual Studio Code jest dystrybuowany bezpłatnie i jest dostępny w wersjach dla platform Windows, Linux i OS X. Jest wykożystywany do napisania kodu i byławybrany ze wzgłedu na wygodność oraz terminał wbudowany do programu. Wykożystywanie wbudowanego terminału pozwalało na efektywniejsze wykożystanie czasu przy każdym odpalaniu serwera, po zmianach do niego wprowadzonych. Mięć osobno terminał a IDE było nie zbyt wygodnym bo potrzeowało ciągłego przełączania się pomiędzy nimi.

Edytor również zawiera wbudowany debugger, narzędzia do pracy z Git,który nawet pokazuję zmianiy od ostatniego comitu, narzędzia do refaktoryzacji, nawigację po kodzie, autodopęłnienia typowych konstrukcji i wskazówki kontekstowe. Produkt wspiera rozwój platform ASP.NET i Node.js i jest pozycjonowany jako lekkie rozwiązanie, które pozwala na pracę bez w pełni zintegrowanego środowiska programistycznego. Wśród obsługiwanych języków i technologii: JavaScript, C ++, C #, TypeScript, jade, PHP, Python, XML, Batch, F #, DockerFile, Coffee Script, Java, HandleBars, R, Objective-C, PowerShell, Luna, Visual Basic, Markdown JSON, HTML, CSS, LESS i SASS, Haxe.

Sourcetree jest darmowy klient Git dla systemów Windows i Mac. Upraszcza on sposób interakcji z repozytoriami Git. Poprzez prosty Git GUI pomaga w wizualizacji i zarządzaniu swoimi repozytoriami. Sourcetree jest nażedziem do wprowadzenia zmian do repozytoriumu gitu na Githubie czy innych aplikacjach pomagających w tym.

**Część praktyczna**

Poniżej jest opisany przebieg realizacji praktycznej części pracy dyplomowej który by zawierał opisanie specyfikacji wstępnych do programu, twożenie repuzutoriumu, twożenie aplikacji z podziałem na części front-endowe oraz back-endowe, demonstrowanie alikacji i jej działania oraz omówienia napotykanych problemów i możliwych dodatków dla rozwinięcia aplikacji.

**Specyfikacja wymagań**

W tym rozdziale opisane są wymogi do serwisu który zostanie zaprogramowany oraz przypadki użycia danego serwisu.

**Wymagania funkcjonalne**

Wymagania funkcyjne określiają zakres możliwości oraz operacje które dany serwis musi posiada dla wykonania. Wymagania funkcjonalne muszą określić w jaki sposub system realizuję postawione cele oraz warunki przy których dana funkcjonalność musi istnieć. Zdefiniowanie i opisanie wamagań funkcjonalnych pomaga programistom zrozumieć zadanie do zaprojektowania.

Dany web serwis musi posiadać:

1. Załogowani użytkownicy musą mięć możliwość twożenia ogłoszeń;
2. Każdy użytkownik musi mieć możliwość załogowania się;
3. Username użytkownika musi być unikałnym;
4. Do każdego ogłoszenia niezbędnę są nąstęone danne:
   1. Stwożono przez;
   2. Kiedy stwożono;
   3. Nażwa podróży jest wymagana;
   4. Teks opisujący podruż jest wymagany;
   5. Cena podróży jest wymagana;
      1. Cena podrózy musi być do .01;
      2. Cena podrózy musi być w zakresię od 0 do nieskończoności;
   6. Obrazek jest wymagany;
   7. Użytkownik nie może wprowadzić do system żadnych plików nie będących obrazkami;
   8. Dozwolone rozrzeżenia obrazku to .png .jpeg .jpg;
5. Dane podawane użytkownikami przy logowaniu twożeniu i etydowaniu czegokolwiek muszą być zapisane w bazie banych;
6. Każdy użytkownik może zostawiać komentaże do ogłoszeń;
7. Ogłoszenia od niezalogowanych użytkowników muszą być wyróżnione w jakiś sposób;
8. Każde ogłoszenie musi mieć możliwość edytowania i usunięcia;
9. Każdy komentaż musi mieć możliwość edytowania i usunięcia;
10. Tyłko zalogowani użytkownicy muszą mieć możliwość edytowania i usunięcia ogłoszenia i komentaża;
11. Tylko twórca ogłoszenia musi mieć możliwość edytowania i usunięcia ogłoszenia;
12. Tylko twórca komentaża musi mieć możliwość edytowania go;
13. Tylko twórca ogłoszenia i twórca komentaża musą mieć możliwość usunięcia komenta;
14. Niezalogowani użytkowniki nie mogą edytować czy usuwać komentaży przez nich stwożone i nie dostają możliwości edytowania czy usunięcia anonimowych komentaży nawet po zalogowaniu.

**Wymagania niefunkcjonalne**

Wymagania niefunkcjonalne – są to warunki które nie stosują się funkjanalności ale równiaż muszą być spęłnione i opisują ograniczenia dla systemu, przy zachowaniu których system powinien nadał prawidłowo realizować swoje założenia funkcje.

1. System ma być uwzgłedniający i przewidywający możliwę błedy i mieć ich obługę;
2. System musi starać się zapewniać dostępność 24 godziny 7 dni w tygodniu przez cały rok oprócz czasu niezbędnego dla aktualizacji, możliwej naprawy oraz testowania systemu;
3. Kod źródłowy projektu musi mieć repozytorium i być zamieszczony w systemie kontroli wersji Git;
4. Baza danych do wykożystania w systemi jest MongoDB typu NonSQL;
5. Pakiety wykożystywane z NPM mogą być dowolne aby tylko kompatebilne ze sobą i łatłe w użyciu i obsłudze;
6. System musi korzystać z następujących technologii oraz standardów:
   1. WebAPI;
   2. RESTfull Web serwise;
   3. Npm
   4. MongoDB;
   5. Bootstrap;
   6. JQuery
7. System musi się rozwijać przyrostowo żeby w każdym momencie jakaś część już była gotowa a nie że potrzebno by było czekać do samego końca żeby wszysko zadziałało.
8. System musi być łatwy w użyciu;
9. System musi być user friendly i być zrozumiałym każdemu użytkownikowi.
10. System powinienen wydawać komunikaty użytkownikowi żeby użytkownik wiedział co się czieję z systemem;
11. System musi mieć łatwy do przeczytania oraz modyfikowania kod i być zoptymlizowanym;
12. Kod musi być DRY o ile się da, żeby był czytelny oraz łatwy w modyfikacji;
13. Dodatkowym plusem by było dodawania AJAXu dla awtomatycznej zmiany na stronie bez konieczności jej przładowania.

**Diagram przypadków użycia**

FILL ME

**Mockup**

FILL ME

**Git**

Przy rozbudowaniu kodu najlepiej kożystać się z gita. Daje to możliwość dość prosto panować nad wersjami kodu i mieć z głowy dublowanie plików po to żeby zmiany wprowadzone ni zepsuły już działającego kodu bez możliwości cofania się. Z tego powodu przy twożeniu danego projektu był używany git. Po stwożeniu plików i żeby dodać ję do repozytoriumu używynę są następujące polecenia(8)

* git init
* git status
* git add .
* git status
* git commit –m „first commit into git, adding files to git”
* git status
* git log

git init – inicjalizuje że w tym folderu będzie wykożystiwany git

git status – zodaczyć status, co jest zrobiono

git add . – dodać wszystkie pliki do systemu kontroli wersij

git commit –m „first commit into git, adding files to git” – zapisywanie plików do gitu gdzie –m oznacza że zostanie dodany text commitu

git log – pokazuje commity danego gitu

żeby wysłać dane do GitHabu dodajęmy jeszcze następne polecenia

* git remote add origin http://ścieżka do gitu
* git status
* git push –u origin master

git remote add origin wskazuję na repozytorium gitu

git push wysyła zmiany na serwer

**Tworzęnie aplikacji**

Po planowani, zdefiniowaniu wymagań które potrzebno spełnić, założeniu gitu i instalacli środowisk do napisania kodu można zaczynać twożyć aplikację. Poniżej opisanie został produkt końcowy oraz czasami elementy przejściowe, ponieważ kod nie był napisany z góry do dołu odrazu takim jaki został na końcu tylko był rozwijany ewolucyjno-przyrostowo. Również opisanie zostało podzielone na części (front-end oraz back-end ewentualne podsumowania).

**Front-end**

Część frontendowa aplikacji nie jest dużą ponieważ funkcjonalność jest zaimplementowana po stronie serwera. Rolą frontendu jest przesyłania danych do serwera orazodebranie ich i wyświetlenie na stronie użytkownikowi.

Żeby nie twożyć specjalnych skryptów do odbierania danych po stronie klienta(czyli front-endowej) i tym samym zmniejszyć obciążenie na stronę przy jej ładowaniu, oraz ukryć od użytkownika końcowego wszystko co nie jest niezbędnymdo pokazywania, wybrałem inny metod – przekazywanie danych po stronie serwera przy renderowaniu strony. Wygląda to następującą po stronie serwera kiedy idzie żądanie do serwera, serwer go odbiera i wykonuję swoję czynności z przygotowania danych, czyli szuka danych w bazię i np. w przypadku ogłoszenia odrazu populuję komentaży oraz obrazek do ogłoszenia za ID komentażów ta obrazka przypisanych danemu ogłoszeniowi i poszukiwaniu i odzytainu komentaży a obrazku w ich dokumentach(bazie danych odnąszącej się do nich). Po wyszukiwaniu wszystkich danych serwer w odpowiedź na rządanie renderuje użytkownikowi stronę i przekazuję do niej dane co pobrał z dazy.

Dla łatwości napisania stron twożyłem je przy pomocy pakietu ejs który pomagał w przygotowaniu plików końcowych użytkownikowi. Czyli kiedy twożyłem strone populowałem dane przekazane z serwera w odpowiednie im miejsca. Kod który musiałem dodać do strony wygłądał w postaci:

<%= dane z serwera których muszę się dostać i pokazać w danym miejscu %>

<% kod JavaSriptu który był wykonany przy przygotowaniu strony użytkownikowi przed jej wyświetleniem %>

Równiż wykożystywałem <% include ściezka do pliku który potrzebno dodać%> dla podpinania plików. Było to wykorzystane żeby kod był jednakowy na każdej serderowanej stronie dez konieczności podmiany w każdym pliku zmian które potzrebno by było wprowadzić oraz żeby nie dublować kodu(DRY) i trzymać ko łatłym do modyfikacji.

Jak widać na odrazkach poniżej tak wygłąda strony napisane w połączeniu z ejs chociaż przy otwieraniu kodu już z przeglądarki (view sorce) kod już wygląda nako normalny HTML bez żadnych <% %> czy <%= %> jak w przypadku plików źródłowych a kody z plików dodawanych po przez <% include %> wyglądają jak by znajdowały się w tym pliku od początku. Przy przygotowaniu strony pliki ejs konwertują się do normalnych plików typu HTML.

FILL ME WITH IMAGES

**JavaScript**

Po stronie klienta nie jest dużo kodu JavaScripu ponieważ jak mówiłem powyżej wszystko co można było zrobić po stronie serwera było zrobiono po stronie serwera. Temu po stronie klienta była dodana tylko walidacja.

Walidacja wygląda następującą:

FILL ME WITH IMAGES

Jest to wykorzystywane dla validowania obrazku dodanego przez użytkownika do ogłoszenia. Typ obrazku musi być z obrzaru dozwolonych, czyli .png, .jpg, jpng. Walidacja działa następującą że przy dodawaniu obrazku trigeruję się funkcja walidacyjna która sprawdza rozszeżenie pliku i nie pozwala dodawać plików rozszeżenie których jest inne od opisanych powyżej.

Kod walidacji można by było zamienić na JQuery validate(9) i stwożyć walidacje dla każdego pola a nie tylko dla obrazka żeby użytkownik ni wpisywał danych w niekorektnej postaci, ale tak jak danych nie jest dużo i od użytkowników załeży jeżeli oni chcą żeby ktoś odpisał na ich ogłoszenie i miał prawidłowe dane do kontaktu z nimi jakie dane oni wprowadzają. Na danym etapie tego wystarczy ale można by było dodać to do listy rzeczy możliwych dla przyszłego rozwoju aplikacji.

**CSS**

Każda strona musi mieć stylowanie żeby mieć wygląd nie tylko tekstowy ale więcej wizualny co ułatwiało by życie. Wygląd strony zależy od kaskadowych arkuszy stylów co jej stylóją w odpowiedności do reguł napisanych w nich. Dla stylowania stron w danym serwisie był używany bootstrap ponieważ stwożone już gotowe kłasy są łatwe w użyciu oraz stylują strone w miare dobrze. Używa się bootstrapu po przez podpinanie jego do kaskadowych arkuszy stylów oraz jeżeli potrzebna jest interakcja związana z bootstrapem dodawane równiż JQuery oraz JavaScript bootstrapu. Dla ich używania w odpowiednich miescach wpisywałem klasy albo inne żeczy co już są napisane przez Twiter bootstrap. Oprócz bootstrapu dla dodawania wlasnych stylów podpięty był wlasny niewięlki plik ze stylowaniem.

**Back-end**

Część back-endowa jest główną ponieważ cały funkcjonal jest wykonany po jej stronie. O ile aplikacja była rozwijana, były dodawane nowe pakiety dla nowej funkcjonalności i nowych możliwiści. W aplikacji używane są midleware dla ustaleń w aplikacji i jej działania.

Middleware jest częścią aplikacji co pomaga w komunikowaniu pomiędzy jej częściami. Np. przed tym jak dać dostemp użytkowniku do edytowania czy usunięcia ogłoszenia czy komentarza przy pomocy pośredniego sprawdzenia sprawdzamy czy użytkownik ma do tego uprawienia, czy dane ogłoszenie zostało wtwożone przez niego a w przypadku komentarza czy może on jest twórcą ogłoszenia.

Tak jak dana część jest dużą, została ona podziełona na mniejszę częsci dla opisania wykorzystanych części i czym oni posługują w danej aplikacji. Daje to że każda część zostanie chociaż by częściowo wyróżniona ta opisana. Póżniej jako podsumowaniem do tego wystąpi opis toku oraz podsumowanie projektu.

**Npm pakiety**

Poniważ część serwerowa aplikacji byłą pisana w Node.js to najłatwiej było pisać używająć już gotowych narzędzi gotowych do użycia i latwych w ustaleniach. Każdy pakiet daję jakąś przygotowaną część funkjanalności. Wszystkie pakiety użyte w danej aplikacji są opisanę w package.json do danej aplikacji. Jest to plik za pomocą którego można łatwo przenieść daną aplikację w inne miejsce i tam zainstalować.

Kiedy tyko zaczyna się napisanie kodu w Node zaczyna się ono od intalacli package.json. komenda „npm init” zaczyna instalacje i pyta się o startowe wartości takie jak name, version, description, main, author, licence. Służą oni do tego, żeby było wiadomo czym jest dany pakiet i jakie ma dependencies. Żeby przenieść serwer po kopjowaniu plików aplikacji i package.json również potrzebno napisać „npm init” i pakiety zostaną zainstalowane.

Żeby dodać inny pakiet do aplikacji potrzebno w folderze gdzie jest package.json napisać następne polecenie „npm i –s nazwa\_pakietu”, gdzie i jest skrótem od install a flaga s odpowiada –save co dodaję dany pakiet do pliku package.json do objektu dependencies dany pakiet.

Niektóre pakiety np. nodemon zowtały zainstałowane globalnie (-g przy instalacji) i nie zostały dodane do pliku package.

Wybrane pokiety definiują od części funkcjonalność programu. Poniżej zostaną krótko opisane pakiety używane w mojej pracy dyplomowej oraz w jakim celu oni służą.

**body-parser**

Dany pakiet(10) pomaga w parsowaniu danych otzymanych ze strony internetowej. Jest to middleware który parsuje dane z żądania wysłanego do serwera i robi ich dostępnymi za pomocą wlaćsiwości body (req.body). po podpinaniu danego pakietu dane z formularzów teraz łatwe do osiągania (req.body.name\_atribute\_from\_form).

Instaluje się poprzez:

* npm install body-parser --save

W aplikacji jest wykorzystywany poprzez:

* const bodyParser = require("body-parser");
* app.use(bodyParser.urlencoded({
* extended: true
* }));

Daje to parsowanie URLów I udostępnienie ich w req.body

**connect-flash**

Pakiet connect-flash(11) jest specjalnym obszarem sesji służącym do przechowywania wiadomościktóre są zapisywane w pamięci flash i kasowane po wyświetleniu użytkownikowi. Czyli np. przy logowaniu użytkownikowi po wprowadzeniu nieprawidłowego hasła zastanie pokazany komunikat z wybranym tekstem(w danym przypadku że hasło jest nie prawidłowe) ale po odswieżeniu strony dany komunikat już nie jest aktualnym i wygasa. Flash jest zwykle używany w połączeniu z przekierowaniami, dzięki czemu wiadomość jest dostępna dla następnej strony, która ma być renderowana. Deklaruje się tekst i typ komunikatu bezpośrednio przed przekierowaniem.

Instaluje się poprzez:

* npm install connect-flash --save

W aplikacji jest wykorzystywany poprzez:

* const flash = require('connect-flash');
* app.use(flash());
* req.flash('info', 'Flash mesaage text')
* res.redirect('/');

Daje to że po przekierowaniu tekst był by dostępny po przez req.flash(‘info’) i mógł by być przsłany do strony docelowej użytkownikowi dla wyświetlenia.

**ejs**

Ejs pakiet(12) pomaga przy przekazywaniu danych do strony pod czas jej renderowania dla końcowego użytkownika. Działa to poprzez specjalne symbole co dają możliwość wykorzystywać JavaScript przy napisaniu stron. Dla napisania JavaScriptu oraz wyświetlenia objektów przekazanych z serwera używanę są konstrukcję typu <% JS code %> oraz <%= JS object to be render as text value  %>. Również ejs daję możliwość użycia innych plików ejs jako części siębie. Czyli po dołączeniu kodu <% include ./partials/header %> dodaję plik header.ejs z folderu partials na miejsce linii z include. Pozwala to nie dublować kodu i przy edycji i dodawaniu zmian dodać je tylko w jednym miejscu.

Instaluje się poprzez:

* npm install ejs --save

W aplikacji jest wykorzystywany poprzez:

* app.set("view engine", "ejs");

Daje to że serwer podczas renderowania oczekuję że strony domyślnie mają rozszeżenia .ejs

**express**

Express(13) jest szybkim, minimalistycznym frameworkiem sieciowym dla Node.js. Express pomaga z napisaniem routingu oraz przy przekierowaniach w HTTP. Jest to jeden z podstawowych pakietów w npm i jest w 10 najpopularniejszych pakietów. Cały REST routing danej aplikacji bazuję się na danym pakiecie.

Instaluje się poprzez:

* npm install express --save

W aplikacji jest wykorzystywany poprzez:

* const express = require('express');
* const app = express();
* app.get('/', function (req, res) {
* res.send('Hello World');
* })
* app.listen(3000)

Daje to że przykładowo aplikacja oczekuję na lokalnym porcie 3000 i po przejsciu do „/” na danym porcie zostanie wyświetlony text „Hello World”.

**express-fileupload**

Przy pomocy express-fileupload(14) po przesłaniu pliku do aplikacji plik będzie dostępny z req.files. W danej aplikacji kiedy użytkownik dodaje plik z obrazkiem do ogłoszenia dany plik jest odczytany po stronie serwera.

Instaluje się poprzez:

* npm install express-fileupload --save

W aplikacji jest wykorzystywany poprzez:

* const fileUpload = require('express-fileupload');

**express-session**

Express-session(15) pomaga w zidentyfikowaniu użytkownika poprzez twożenie sesji. Dane sesji są przechowywane po stronie serwera.

Instaluje się poprzez:

* npm install express-session --save

W aplikacji jest wykorzystywany poprzez:

* const session = require ( 'express-session' );
* app.use(session({
* secret: 'keyboard cat',
* resave: false,
* saveUninitialized: true,
* }))

Daje to że użytkownik po załogowaniu i do wyłogowaniu był by znanym użytkownikiem i miał by za sobą dane danej sesji.

**method-override**

Tak jak HTML formy w danym momencie nie podtrzymują „PUT” i „DELETE” metod(16) temu method-override(17) pomaga w teożeniu RESTfull routingu z obejściem danego. Jeżeli ustawić method=”POST” ale action troche modyfikować to po stronie serwera method-override zmieni typ requesty na PUT albo DELETE odpowiednio.

Instaluje się poprzez:

* npm install method-override --save

W aplikacji jest wykorzystywany poprzez:

* const methodOverride = require('method-override');
* app.use(methodOverride('\_method'));

Daje to że każde PUT rządanie z dopisem „\_method=PUT” na końcu atrybutu action formy zostanie traktowane jako PUT request, i tak samo z rządaniami typu DELETE.

**moment**

Moment(18) jest lekką biblioteka JavaScript do analizowania, sprawdzania poprawności, manipulowania i formatowania dat. W przypadku danej aplikacji dla wyświetlenia jak dawno temu było stwożone ogłoszenie albo komentaż biblioteka moment pomada w formatowaniu danych pokazywanych użytkownikowi.

Instaluje się poprzez:

* npm install moment --save

W aplikacji jest wykorzystywany poprzez:

* const moment = require('moment');
* res.locals.moment = moment;

Daje to że przy rennderowaniu strony użytkownikowy po stwożeniu ogłoszenia czy komentaża było by widać że jest stwożony kilka sekund temu, a póżniej ile minut, dodzin itd. temu.

**mongoose**

Mongoose(19) to narzędzie do modelowania obiektów MongoDB przeznaczone do pracy w środowisku asynchronicznym. Czyli mongoos posługuję do działania z bazą danych MogoDB. Opisanie bazy danych da jej schematów jest w jednym z następnyh działów.

Instaluje się poprzez:

* npm install mongoose --save

W aplikacji jest wykorzystywany poprzez:

* const mongoose = require("mongoose");
* mongoose.connect("DB\_path");

Daje to możliwość działania na dazię danych dla odczytywania, zapisywania oraz mobyfikowania i usunięcia danych z bazy.

**nodemon**

Nodemon(20) jest pakietem co pomaga przy twożeniu oprogramowania. Serwer podczas go napisania potrzebno by było uruchamiać ponownie co raz po wprowadzaniu zmian. Zamiast robić to ręcznie można dodać nodemona który będzie oglądał pliki w katalogu, w którym nodemon został uruchomiony, a jeśli jakiekolwiek pliki ulegną zmianie, nodemon automatycznie zrestartuje aplikację noda. Dany pakiet nie jest w pliku package.json ponieważ jest instalowany globalnie.

Instaluje się poprzez:

* npm install -g nodemon

Daje to możliwość do automatycznego ponownego uruchamiania serwera po zmianach po przez polecenie „nodemon  nazwa\_serwera”.

**passport**

Passport(21) jest pakietem, który pomaga w autentyfikowaniu użytkowników. Passport wykorzystuje koncepcję strategii do autentyfikowanie żądań. Strategie mogą obejmować weryfikację nazwy użytkownika i poświadczeń hasła. Passport jest dobry tym że nie nażuca żadnych shematów dazy danych co pozwala podejmować decyzję programistowi systemu. Passport można by było połaczyć z logowaniem poprzez Google czy Facebook ta innych, ale dla danego projektu została wybrana lokalna strategia logowania.

Instaluje się poprzez:

* npm install passport --save

W aplikacji jest wykorzystywany poprzez:

* const passport = require('passport');
* app.use(passport.initialize());
* app.use(passport.session());
* passport.use(new LocalStrategy(User.authenticate()));
* passport.serializeUser(User.serializeUser());
* passport.deserializeUser(User.deserializeUser());

Daje to że passport wykorzystuje lokalną strategie dla autentyfikowaniu użytkowników.

**passport-local**

Passport-local(22) przedstawia lokalną strategie logowania przy użyciu nazwy użytkownika i hasła.

Instaluje się poprzez:

* npm install passport-local --save

W aplikacji jest wykorzystywany poprzez:

* const LocalStrategy = require('passport-local');
* passport.use(new LocalStrategy(User.authenticate()));

Daje to możliwość logowania się lokalnie, bez wykorzystania innych profilów użytkowników z innych intancij.

**passport-local-mongoose**

Passport-local-mongoose (23) jest plaginem dla konfiguracji mongoose dla używania passportu

Instaluje się poprzez:

* npm install passport-local-mongoose --save

W aplikacji jest wykorzystywany poprzez:

* const passportLocalMongoose = require('passport-local-mongoose');
* User.plugin(passportLocalMongoose);

**FILL ME google map module**

text

**Routing**

Routing w aplikacji jest bazowym funkcjonałem bez którego manipulowanie serwisem było by dość skomplikowane. Routing był robiony bazująć shematu arhitektóry REST, gdzie dla każdej operacji(INDEX, SHOW, CREATE, NEW, EDIT, UPDATE, DESTROY) jest wlasny routing z właściwym dla niego funkcjonałem. Przykladowy routing wygłąda następująco:

* app.get("/", (req, res) => {
* res.render("landing", {pagename: "Allx Travel"});
* });

Gdzie “/” to adres na naszym serwerze(w danym przydadku localhost:3005/(FILL ME IF WRONG)), req to parametry wejściowe do routingu (parametry z formularza również przyhodzą tutaj req.body.parametr) a res parametry wyjściowe. Res.render odpowaida za renderowanie strony i przyjmóje dwa parametry, pierwszy to nazwa pliku co zostanie wyświetlony i jest to landing.ejs, gdzie rozszeżenie .ejs jest domyślnie oczekiwane w aplikacji a drugi jest to objekt który przyjmuję parametry co zostaną przekazane do strony, czyli przy renderowaniu strony ejs podstawi zamiast <%= pagename %> text Allx Travel.

Routing został podzielony oraz zapisany do różnych plików ze wzgłedu na obsługiwaną funkcjonalność, czyli np. w pliku comments ścieżki odpowiadające za twożenie i edytowanie komentarza co jest logicznym. Dzielenie na osobne pliki pomaga podtrzymywać kod czytelnym i nie przegromadzonym oraz latwiejszym w edycji bo za nazwą pliku można zrozumieć jaki routing on podtszumuję.

**Baza danych - Mongoose NONSQL database**

Dla projektu została wybrana baza danych typu NONSQL, oznacza to że baza charakteryzuje się dużą skalowalnością, wydajnością oraz brakiem ściśle zdefiniowanej struktury obsługiwanych baz danych(np dokument ma 100 zadeklarowanych pól ale każdy wiersz może mieć od 0 do 100 pół o ile nie są wymagane) na przeciwięstwo baz danych typu SQL, gdzie struktura odrazu jest stałą i każdy wiersz(pojedynczy objekt) ma te pola zawsze(np. tabela ma 100 zadeklarowanych kołumn to i każdy wiersz ma 100 kołumn odpowiednio). Wykorzystanie bazy typu NONSQL daje możliwość zaoszczęzać przestszeń co jest bardzo istotne w przypadku wielkiej ilości danych.

**Relacji pomiędzy tabelami**

Relacje pomiędzy dokumentami mogą być albo 1 – 1 albo 1 - \*. W danej aplikacji są cztery różne dokumenty opisujące polączenia pomiędzy nimi i ich danymi.

* Każdy użytkownik może utworzyć wiele ogłoszeń, ale jedno ogłoszenie może mieć tylko jednogo autora(1 - \*);
* Każdy użytkownik może utworzyć wiele komentaży do ogłoszzenia, ale jeden komentaż może mieć tylko jednogo autora(1 - \*);
* Każde ogłoszenie ma jeden obrazek do go przedstawienia oraz ten odrazek jest przypisany tylko do jednego zdefiniowanego ogłoszenia(1 - 1);
* Każde ogłoszeni może mieć duzo komentaży, ale jeden komentaż może być tylko w jednym ogłoszeniu(1 - \*).

Za zdefinuiwanie połączeń pomiedzy dokumentami oraz zdefiniowanie pól które te dokumenty mają odpowiadają shematy, ich wygląd można zobaczyć poniżej.

FILL ME WITH IMAGES

**Autoryzacja**

Za autoryzajce użytkownika w systemie odpowiadają 3 dołącone pakiety I są to

* passport;
* passport-local;
* passport-local-mongoose.

Pasport zapełnia aplikacje funkcjonalnością do logowania użytkownika, passport-local mówi passpotowi że dla logowania używamy lokalną strategie a passport-local-mongoose odpowiada za dodawanie username, pol hash i salt, aby zapisać nazwę użytkownika, hashowane hasło i wartość salt.

Całą autoryzacje można wykonać po przez konfiguracje danych pakietów oraz używanie gotowych funkcji, bez napisania własnego kodu co odpowiadał by za łogowanie i rejestrowanie użytkowników w systemie.

**Authentyfikacja oraz Middleware**

Część oprogramowań pośredniczących była wykorzystywana z dołączonych pakietów ale niestety nie wszystko można się zrobić w taki sposób ponieważ każda aplikacja na wlasną funkcjolnalność i przewidywane miejsca ich wywołania za kolejnością, temu do danej aplikacji musiałem napisać cztery wlasne midleware odnoszących się do uprawień użytkownika ale żeby wiedzieć kim jest dany użytkownik on musi by zaauteryzowany oraz zaautentyfikowany. Poniżej są opisane znaczenia działania czterech midelware:

* Dla tego żeby dodawać, edytować albo usuwać ogłoszenie użytkownik musi być załogowany do systemu;
* Dla tego żeby edytować albo usuwać ogłoszenie użytkownik musi być autorem danego ogłoszenia;
* Dla tego żeby edytować komentarz użytkownik musi być autorem danego komentarza;
* Dla tego żeby usuwać komentarz użytkownik musi być autorem danego komentarza albo autorem ogłoszenia do którego przypisany jest komentarz.

FILL ME WITH IMAGES

Po wykonaniu danych sptardzań program był wywoływany dalej w swoim rządaniu a jeżeli nie to rządanie zostawało odrzucone. Przystąpienie do kolejnej części było po przez polecenie next().

**Tok projektu**

Praca była wykonywana zgodnie z zakresem pracy ponieważ tam zostały opisane i dzefiniowane poszczegulne zadanie do realizacji oraz ich kolejność.

1. REST Routing

Na samym początku było pobieranie pokietów przez npm i ich podpinanie w aplikacji. Dane działanie występowało na stałe przy rozszeżeniu funkcjonalności. W zależności od tego jaka akcja musiała być dodana załeżał jej funkconał, ale najpierw dane zdażenie musiało by mieć własną ścieżke marszutyzacji. Np. na samym początku projektu po intalowaniu, użyciu pakietów i wybraniu na jakim porcie musi działać aplikacja(również dobrze mogła by działać nie lokalnie a na hostowanym serwerze pod wykupiąną domeną), pierwszym co było napisane to była ścieżka „/” odpowiadająca za strone główną aplikacji. Na samym początku przy testowanie jakiejkolwiek ścieżki zaczynałem od wysyłania res.send(„text”) żeby zobaczyć że dana ścieżka działa i sprawdziałem jej działanie. Czyli na początku kod wyglądał jako:

* var express = require('express');
* var app = express();
* app.get('/', function(req, res){
* res.send('hello world');
* });
* app.listen(3005, function (){
* console.log("Server Started");
* console.log("Port 3005");
* });

Przy odwarciu w przeglądarce danego portu i przejściu na jego ścieżke „/” wiedziałem text 'hello world' co oznaczało że ścieżka działa i można iść dalej.

1. Tworzęnie stron

Kolejnym krokiem było twożenie strony co była wyświetłona użytkownikowi przy przejściu do danej strony. Najpierw to była strona html w tekstowej postaci z tekstem co muwił nie wiele więcej, że jest to strona glówna i tego było wystarczająco, bez żadnych stylowań, póżniej w kolejnych krokach była ona stylowana i wyglądała co raz lepiej. Dla wyświetlenia strony linija z res.send była podmieniona na res.render ze ścieżką do pliku, ale tak jak plik był w tym samym miejscu to było to proste.

1. Twożenie ogłoszenia

Dałej rozwijąjąć aplikacje dodałem ścieżke i strone dla ogłoszenia. Zaczałem od ścieżek „NEW” i „CREATE”, ponieważ odpowiadają oni za pokazawie strony dla stwożenia ogłoszenia oraz dodawania danego ogłoszenia. Na początku wystarczyło formularza co wysyłał danę a po stronie serwera ich wyświetlenia w konsoli. Przy twożeniu stron zauważono było że cześci początku z dodaną nawigacją i końca pliku z podpinanym bootstrapem również takie samo jak i w przypadku strony głownej temu dodany i używany został pakiet ejs co pomógł wynieść części wsólne do jednego wspólnego pliku.

1. Konfigurowanie bazy danych i podpinanie jej do aplikacji żeby dać możliwość zapisywania danych do niej

Po odebraniu danych przesłanych ze stromy i wyświetleniu ich wystąpiła konieczność gdzieś ich przechowywać. Mongoose – pakiet npm odpowiadający za polączenie z bazą mongoDB którą również pobrałem i dopasowałem dla zapisywania danych z serwera. Po wykonaniu danych działań stwożyłem shemat dla ogłoszenia co miał w sobie pola dla podania autora, nazwy oraz tekstu ogłoszenia. Po stronie serwera dodane zostało zapisywanie do bazy oraz nowe ścieżki dla odczytywania z bazy. „INDEX” odpowiadał za wyświetlenie wszystkisz ogłoszeń a „SHOW” za wyświetlenie ściegulnego wybranego ogłoszenia. Tak jak w danym kroku było najwiencej kodu do napisania od początku projektu do danego monentu to zauważono było że przy każdej zmianie nawet drobnej, czy literówce potrazebno było odnowa odpalać serwer żeby zmiany zostały dodane do aplikacji. Wyjściem było dodanie pakietu nodemon co pomagał w tym i automatycznie restertował serwer odnowa przy zmianiach w pliakch .js. Ułatwiało to pisanie kodu bez ciąłego odpalania serwera odnowa. Zmieniło się polecenie dla odpalania z „node app.js” na „nodemon app.js”.

1. Dodawanie komentarzy

Dalej miało sens dodawanie komentarzy do ogłoszeń temu zdefiniowany został schemat dla bazy danych oraz napisane zostały ścieżkie oraz strona dla obsługi dodawania komentarzy. Tak jak komentarzy nie mają wlasnej strony i niema strony do pojedynczego wyświetlenie komentarza to i ścieżki „INDEX” oraz „SHOW” nie są potrzedne. Natomiast komentarzy są pokazywane na stronie ogłoszenie powieważ do niej i mają bezpośrednie odnoszenie. Wygląda to w następujący sposub. Tak jak w ogłoszeniu oprósz parametrów zwykłych pole z komentarzami jest powiązane z komentarzami w polączenie 1 - \* to wyszukująć za parametrem id zapisanym w ogłoszeniu można znałeść komentarzy. To wszystko jest robiono przy wyszukiwaniu ogłoszenia w bazie danych. Jako dodatkowy parameter przy odczytywaniu objektu ogłoszenia zadany jest .populate("comments") co wyszukuję za parametrem id komentarza. Na samej stronie był napisany JavaScript co przechodził po każdym objekcie i wyświetłał każdy osobny komentarz. Tak jak iłość stron się gromadziła to potrzebno było podzielić ich na logicznę części i tak została stworzony folder view oraz podfoldery za ich znaczeniem.

1. Logowanie

Przed dodawaniem możliwości usunięcia oraz edycji ogłoszenia czy komentarza potrzebno by było wiedzieć kto dane ogłoszenie czy komentarz napisał. Czyli potrzebno było zaimplementować shemat bazy dla użytkownika oraz pobrać pakiety z npm ta użyć ich w aplikacji. Przy pomocy pakietów passport oraz dodatków do niego to było bardzo prosto, ponieważ opisywanie tego w jaki sposób podępnąć to wszystko razem było na stronie pakietów. Po zaimplementowaniu ścieżek logowania oraz rejestracji urzytkownika ilość kodu była podziełona, jak i w przypadku stron i folderu view, był stworzony folder routes i jego podkatałogi.

1. Autentyfikacja

Dany problem też był rozwiązany przy pomocy pakietów passport ta innych. Żeby wiedzieć inforwacje o urzytkowniku co stworzył ogłoszenie czy komentarz ich shematy oraz ścieżki ułegli zmienie dla przehowywania danej informacji.

1. Edytowanie lub usunięcie ogłoszenia przez go wsaściciela

Po dodawaniu odłoszenia pasowało by dodać funkjonalność dla obsługi go edytowania. Po dodawaniu odpowiednich ścierzek dla edycji objektu w bazie na podwtawie id ogłoszenia dodane zostały middleware co sprawdzały czy użytkownik co teraz jest załogowany, lub nie mam uprawnienia dla edycji lub usunięcia. Najpierw był napisany middleware co sprawdza czy użytkownik jest załogowany a póżniej i funkcji sprawdzające za parametrem id autora ogłoszenia.

Po przejściu midleware użytkownik miał dostęp do scieżek „EDIT”, „UPDATE” ta „DELETE”.

1. Edytowanie lub usunięcie komentaży przez ich wsaścicieli

Edytowanie ta usinięcia koentarzów różnią się uprawnieniami. Edytować ogłoszenie może tylko załogowany użytkownik co był już załogowany przy tworzeniu ogłoszenia i jest go autorem. Również dana osoba może usuwać ogłoszenie. Jeszcze uprawnienia dla usunięcia komentarzy ma autor ogłoszeń.

1. Wyświetlenie komunikatów dla użytkownika oraz rozszerzenie istniejącego funkcjunału

Po tworzeniu funkcjonalnej części pasowało by ją rozrzeżyć, temy dodawane zosrały nowe pola do ogłoszeń, furmulaży z nimi związane oraz shematy w bazie danych. Teraz do ogłoszenia dodawany jest obrazek co jest zapisywany po stronie serwera, mapa google z jej api, kalendarz dla wyboru dat wyjazdu ta przyjazdu oraz jak i z komentarz ma teraz date twotzenia. Obrazek został zapisywany na dysk twardy a nie do bazy danych z tego powodu że dla zapisu obrazku w bazu pasowało by go przekonwertować do postaci base64 ale wysoka ilość zapytań do serwera powodowała by to że każdy raz przy reagowaniu na rządanie obrazek musiał by być konwertowany do base64 albo z powrotem i obcążenie po stronie serwera było by większe niż w przypadku zapisywania na dysk. Wszystkie te dane pomogą w zdefiniowaniu punktu oraz daty wyjazdu. Obrazek był by pokazany na stronie „INDEX” dla wyśczegulnienia ogłoszenia nie tylko tekstem a i grafiką równiez.

Dla tego żeby system w jakiś sposub komunikowanikował się z użytkownikiem i użytkownik się nie gubił w nim, nie wiedząc co się dzieje, do użytkownika muszą być wysyłane komunikaty z serwera z informacją informującą o toku reakcji na zdażenie po stronie użytkownika końcowego. W celu tego podpinany został pakiet connect-flash co i odpowiada za dane komunikaty. Komunikaty zostały by wyświetlowe użytkownikowi w razie przewidywanych blędów albo w odpowiedź na zakończone działania, ale już po przeladowaniu strony nie byli by widoczne. Odbyła się w następujący sposób że przed przekierowaniem do innej strony definiuję się komunikac co zostaję wyświetlomy w konkretny, miejscu na stronie.

1. Responsywność stron do różnych rozszeżeń ekranu.

Do zakończeniu dodawania funkcjonału strone go reprezentujące zostały ostyłowane. Styłowanie musiało być odpowiednie do różnych rozmiarów ekranów ponieważ im mniejszy jest ekran tym mniej elementów mogło by być pokazywane na zednym ekranie użytkownika.

Dla styłowania był używany bootstrap ponieważ ma on już gotowe klasy ze zdefiniowanym stylowaniem oraz system grid co pomógł przy rozmieśczeniu elementów przy różnej rożdzielczości. Np google map co została dodana znajduję się po prawej stronie kiedy rozdzielciość ekranu na to pozwala, a jak nie przechodzi wdół pod głowną część ogłoszenia ale nad komentarze i zostaje ukryta. Po kliknięciu na przycisk zostaję ona pokazywana albo ukrywana z powtotem. To co nie było w bootstrapie albo specyficzne stylowanie do stron było napisano rzęcznie w pliku ze stylowaniem. @media odpowiada za stylowanie przy zdefiniowanym rozszerzeniu.

1. Podtzymywanie różnych wersii przeglądarek internetowych.

Żeby podrzymywać różne przeglądarki (również IE) zowtała wybrana 3 wersja bootstrapu chociaż na moment napisania pracy wersja 4 była dostępna do użycia. Było ty wybrano z tego powodu że w wersji 4 bootstrap był robiony na display flex którego IE nie podtrzymuje. Wersja 3 jest optymalną w danym przypadku żeby podtrzymywać kompatyblność z większą ilością rzeglądarek.

Po wykonaniu danych kroków przojekt był gotowy i spełniający wymagania.

**Sprawdzenie działania aplikacji**

FILL ME

**Demonstrowanie alikacji i jej działania**

FILL ME WITH FINAL IMAGES

**Napotykane problemy**

Kiedy się uczysz czegoś nowego to ciągle masz jakieś problemy przy programowaniu czegoś bo nie wiesz w jaki wlaściwie sposób było by najlepiej dany problem rozwiązać. Np przy napisaniu pragramu miałem problemy z następnymi rzeczami:

Po instalacji mając probmemy z uruchamieniam serwera przy użyczi nodemona. Skożystałem się z rozwiązania znalezianego na Stack Overflow(FILL ME WITH LINK) które pomogło.

Miałem problemy z podpinaniem google api do slasnej aplikacji przy pgraniczeniach klucza do localhostu temu zostało utworzono 2 kluczę, gdzie jeden był do przesyłania geodanych i ich opracowania a drugi przy wyświetleniu przy pomocy JavaScriptu wyników na mapie.

Inne drobniejsze problemy konfiguracyjne najczęściej zostały rozwiązane po przeczytaniu dokumentacji do pakietu.

**Możliwości do przyszłego rozwinięcia aplikacji**

Tak jak funkcjonał danej pracy opierał się na głównie na założeniach wstępnych ale również od części rozwijął mół by on również mieć następujące czynności ale na danym etapie nie było potrzeby w zaimplementowaniu danych rzeczy:

JQuery Validate

Validowanie formularza przed wysyłką mógło by być zrealizowane przez JQuery Validate. Przy pomocy JQery Validate można by było zdefiniować wymagane reguły oraz tekst walidaćji do wybranych pól.

User profile

Również jak i ogłoszenia urzytkownik mógł by mieć wlasny kabinet z ustawieniami, możliwością zmiany hasła i temu podobnych rzeczy, ale zwiększyło by to kod minimum o jedną trzecią. Dany funkcjonał był by preferowany w dodawaniu kolejnym przy rozwinięciu aplikacji.

Admin profile

Dobrzą praktyką by było mieć użytkownika z prawami administratora dla zarządzania systemem oraz ogłoszeniami i kontrolowaniem innych użytkowników.

Double opt-in

FILL ME

Zakończenie

Literatura

1. <https://getbootstrap.com/>
2. <https://jquery.com/>
3. <https://nodejs.org/en/>
4. <https://insights.stackoverflow.com/survey/2017#technology>
5. <https://insights.stackoverflow.com/survey/2017#technology-frameworks-libraries-and-other-technologies>
6. <https://www.npmjs.com/>
7. <http://sd.blackball.lv/library/RESTful_Web_APIs_(2013).pdf> - книжка
8. <https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf> - git
9. <https://jqueryvalidation.org/>
10. <https://www.npmjs.com/package/body-parser>
11. <https://www.npmjs.com/package/connect-flash>
12. <https://www.npmjs.com/package/ejs>
13. <https://www.npmjs.com/package/express>
14. <https://www.npmjs.com/package/express-fileupload>
15. <https://www.npmjs.com/package/express-session>
16. <https://softwareengineering.stackexchange.com/questions/114156/>
17. <https://www.npmjs.com/package/method-override>
18. <https://www.npmjs.com/package/moment>
19. <https://www.npmjs.com/package/mongoose>
20. <https://www.npmjs.com/package/nodemon>
21. <https://www.npmjs.com/package/passport>
22. <https://www.npmjs.com/package/passport-local>
23. <https://www.npmjs.com/package/passport-local-mongoose>
24. <https://stackoverflow.com/a/17976504> - nodemon help

**Zalecana zawartość treści pracy**

Wstęp

Kilkustronicowa analiza ogólna wprowadzająca w realizowane zagadnienie, zarysowująca problematykę pracy i użyteczność wynikającą z jej realizacji (uzasadnienie realizowanej problematyki). W ramach wstępu należy wyszczególnić następujące elementy:

* uzasadnienie wyboru problematyki/tematu pracy,
* cel pracy
* aktualny stan wiedzy dotyczący problemu (zagadnienia) będącego przedmiotem pracy,
* zakres pracy,
* omówienie układu pracy, to jest krótka charakterystyka poszczególnych rozdziałów.

Pod pojęciem celu pracy dyplomowej na studiach pierwszego stopnia należy rozumieć nie tyle samo rozwiązanie problemu, ile korzyści jakie ono przyniesie (cel to przyszły efekt podjętego działania - rozwiązywania problemu - który działanie to uzasadnia). Cel pracy powinien być tak sformułowany, aby możliwa była ocena efektywności działań podjętych w ramach jego rozwiązywania oraz stopnia osiągnięcia założonego celu.

Część główna pracy (podzielona na numerowane rozdziały, ewentualnie podrozdziały)[[1]](#footnote-1)

Część główna pracy powinna być dostosowana do specyfiki i przedmiotu rozwiązywanego problemu (zagadnienia) i składać się z części teoretycznej i praktycznej.

W części teoretycznej autor powinien dokonać charakterystyki problematyki pracy, analizy aktualnego stanu rzeczy z określeniem pojawiającego się problemu oraz wskazać potencjalne kierunki jego rozwiązania z uzasadnieniem wyboru jednego z nich.

W części praktycznej autor powinien wykazać się znajomością adekwatnych do kierunku studiów procedur (metodologii) rozwiązywania problemów, umiejętnością określania stopnia osiągniecia założonego celu pracy oraz wyciągania na tej podstawie wniosków na temat uzyskanych wyników i efektywności podjętych działań.

Zakończenie

Zakończenie powinno zawierać ocenę skuteczności podjętych działań i uzyskanych efektów pod kątem stopnia osiągniecia założonego celu pracy.

Literatura

Należy podać wszystkie źródła, które wykorzystano w pracy, w tym również - oprócz aktów prawnych, wydawnictw zwartych i ciągłych - adresy stron www.

**I. ZASADY FORMATOWANIA TEKSTU PRACY**

* Plik z pracą dyplomową należy zapisać jako dokument programu Word 97-2003 (\*.doc).
* Nie należy wstawiać nagłówków i stopek z oddzielną treścią, np. tytułem rozdziału.
* Każdą część pracy należy rozpoczynać od nowej strony.
* Nie należy stawiać kropki po tytule danej części pracy (w tym rozdziału/podrozdziału) oraz po tytule pracy.
* Numeracja stron - obowiązkowo automatyczna (w stopce, wyrównana do środka strony, Czcionka Times New Roman 12 pkt.). Pierwszą stroną pracy jest strona tytułowa i od niej należy zacząć numerowanie, ale nie umieszczać na niej numeru strony.
* Automatyczny spis treści – na trzeciej stronie pracy dyplomowej.
* Czcionka - Times New Roman, rozmiar 12 pkt., grubość normalna.
* Odstępy między znakami – standardowe (0 pkt.).
* Pomiędzy wyrazami nie należy wstawiać więcej niż jednego znaku spacji.
* Odstępy między wierszami - pojedyncze.
* Tekst justowany obustronnie.
* Włączona opcja automatycznego dzielenia wyrazów.
* Jeżeli dany akapit ma znaleźć się na nowej stronie należy zastosować funkcję podziału strony (kombinacja klawiszy: *Ctrl + Enter*).
* Na końcu wiersza nie mogą występować pojedyncze litery lub spójniki. W celu przeniesienia spójnika do kolejnego wiersza należy użyć tzw. twardej spacji (kombinacja klawiszy: *Ctrl* + *Shift* + *spacja*). „Twardą spację” należy wstawić bezpośrednio po spójniku, a przed następującym po nim wyrazem (powoduje to „przyklejenie” spójnika do wyrazu).
* Zabrania się wstawiania na końcu wiersza (w środku zdania) zwykłych oraz „miękkich” enterów.
* Pomiędzy wyrazem a następującym po nim znaku przestankowym (przecinek, kropka, średnik, dwukropek) nie należy wstawiać znaku spacji.
* Wyrazy (zdania) pisane w nawiasie nie mogą być oddzielone od tych nawiasów spacjami.
* Wyliczenia w tekście - za pomocą automatycznego wypunktowania lub numerowania.
* Praca musi być wydrukowana dwustronnie - w zakładce *Plik/Ustawienia strony/Marginesy* należy zaznaczyć „marginesy lustrzane” (margines wewnętrzny - 2 cm, margines zewnętrzny, górny i dolny - 1,5 cm).

**II. STRONA TYTUŁOWA PRACY DYPLOMOWEJ**

Strona tytułowa pracy dyplomowej powinna zawierać:

1. logo w kolorze i nazwę szkoły wg podanego wzoru (czcionka Times New Roman, bold, 16 pkt.),
2. nazwę wydziału, kierunek oraz specjalność (czcionka Times New Roman, bold, 16 pkt.),
3. imię i nazwisko autora (autorów) pracy, nr albumu (czcionka: Times New Roman, 15 pkt.) oraz tytuł pracy dyplomowej (czcionka: Times New Roman, *kursywa*, bold, 20 pkt.),
4. tytuł/stopień naukowy, imię i nazwisko promotora (czcionka: Times New Roman, 12 pkt.),

*UWAGA! Oprawa pracy dyplomowej zawiera okienko o wymiarach 15,5 x 6,5 cm, (umieszczone centralnie w stosunku do marginesów bocznych oraz w odległości 12,5 cm od dolnej krawędzi strony), w którym powinny się zmieścić informacje zawarte w pkt. c) i d).*

1. napis PRACA DYPLOMOWA MAGISTERSKA (lub LICENCJACKA/INŻYNIERSKA), (czcionka: Times New Roman, bold, 19 pkt),
2. napis Rzeszów oraz rok złożenia pracy dyplomowej, np. Rzeszów 2018 (czcionka: Times New Roman, bold, 14 pkt.).

Na kolejnych stronach znajduje się wzór strony tytułowej pracy dyplomowej - wersja dla prac przygotowanych w języku polskim (na przykładzie kierunku Informatyka) oraz wersja dla prac przygotowanych w języku angielskim (na przykładzie kierunku Filologia).

1. dziekan wydziału może w drodze decyzji określić bardziej szczegółowe wytyczne dotyczące zawartości pracy obowiązujące dla danego kierunku studiów [↑](#footnote-ref-1)