

WOJSKOWA AKADEMIA TECHNICZNA

im. Jarosława Dąbrowskiego

WYDZIAŁ CYBERNETYKI



PRACA DYPLOMOWA

STACJONARNE STUDIA I°

Temat: **PROJEKT I IMPLEMENTACJA APLIKACJI
DO WSPOMAGANIA PLANOWANIA WYJAZDÓW
TURYSTYCZNYCH**

Autor:

Sylwia PYTKO

Promotor pracy:

**dr hab. inż.
Kazimierz WORWA**

Warszawa 2018

OŚWIADCZENIE

*„Wyrażam zgodę na udostępnianie mojej pracy przez
Archiwum WAT”.*

Dnia

.....

(podpis)

Pracę przyjąłem

promotor pracy

dr hab. inż. Kazimierz Worwa

Spis treści

Wstęp	7
I. Ogólna charakterystyka komputerowego wspomaganie planowania wyjazdów turystycznych.....	8
1. Opis dziedziny	8
2. Rola informatyki w turystyce.....	9
II. Specyfikacja wymagań	11
1. Wymagania biznesowe	11
2. Dziedzinowy słownik pojęć	12
3. Wymagania funkcjonalne	14
4. Wymagania niefunkcjonalne	17
III. Projekt aplikacji.....	18
1. Diagram encji.....	18
2. Diagram przypadków użycia	19
2.1. Aktorzy.....	19
2.2. Opis przypadków użycia	19
3. Baza danych	24
4. Diagram klas	25
5. Podział na warstwy	26
6. Struktura drzewa plików aplikacji	28
IV. Opis implementacji.....	30
1. Użyte technologie i charakterystyka wykorzystanych narzędzi	30
1.1. Użyte wzorce projektowe.....	30
1.2. Użyte technologie serwera aplikacji	30
1.3. Użyte technologie serwera klienta	32
1.4. System kontroli wersji GIT	34
2. Usługi aplikacji	35
3. Szczegółowy opis implementacji wybranej usługi.....	37
4. Interfejs graficzny aplikacji	40
4.1. Single-page app	40

4.2. Wybrane panele interfejsu graficznego	41
5. Opracowanie i realizacja planu testowania	47
Podsumowanie	49
Bibliografia	50
Spis rysunków	51

Wstęp

Celem pracy jest opracowanie projektu i implementacji aplikacji wspomagającej planowanie wyjazdów turystycznych. Jej głównym przeznaczeniem będzie pomoc użytkownikowi w zaplanowaniu swoich podróży turystycznych w aspektach takich jak zaplanowanie miejsc, które użytkownik chce odwiedzić, jak również noclegu oraz możliwych atrakcji i aktywności, które turysta będzie chciał umieścić w swoim planie na spędzenie czasu w danym celu podróży. Aplikacja umożliwi użytkownikowi zanotowanie potrzebnych mu informacji w przejrzystej i wygodnej do edycji formie. Zapewni również dodatkową formę wizualizacji szczegółów, taką jak mapy ukazujące wprowadzone miejscowości oraz terminarz z graficzną reprezentacją zaplanowanych aktywności użytkownika.

Praca składa się z czterech rozdziałów, z którym pierwszym jest opis dziedziny, jaką jest zagadnienie turystyki oraz opis istniejących rozwiązań w zakresie komputerowego wspomagania turystyki. Następnie przedstawiony będzie rozdział dotyczący specyfikacji wymagań tworzonej aplikacji uwzględniając opisowe wymagania biznesowe na aplikację do wspomagania planowania wyjazdów turystycznych oraz szczegółowe wymagania funkcjonalne i нефункционалне. Kolejnym, trzecim, rozdziałem pracy jest projekt aplikacji zawierający w sobie odpowiednie diagramy UML i ich opis, takie jak diagram przypadków użycia, diagram encji, diagram klas oraz opis podziału aplikacji na warstwy. Po tym następuje rozdział czwarty, który zawiera opis implementacji zaprojektowanej aplikacji. W rozdziale tym zostanie przedstawiony także opis wybranych technologii użytych do realizacji implementacji, a następnie zostaną opisane utworzone usługi aplikacji realizujące jej funkcjonalności. Aby przedstawić schemat wykonywania aplikacji i sposób implementacji zostanie ukazany szczegółowy opis implementacji jednej przykładowej usługi. Następnie będzie przedstawiony przygotowany interfejs graficzny użytkownika poprzez zaprezentowanie wybranych ekranów interfejsu użytkownika. Pracę podsumują testy aplikacji oraz wynikające z nich wnioski.

I. Ogólna charakterystyka komputerowego wspomagania planowania wyjazdów turystycznych

1. Opis dziedziny

W dokumencie o tytule "Terminologia turystyczna. Zalecenia WTO" Światowa Organizacja Turystyki UNWTO (ang. United Nations World Tourism Organization) opublikowała definicję turystyki. Według tego dokumentu definicja ta brzmi:

„Turystyka obejmuje ogół czynności osób, które podróżują i przebywają w celach wypoczynkowych, służbowych lub innych nie dłużej niż jeden rok bez przerwy poza swoim codziennym otoczeniem, z wyłączeniem wyjazdów, w których głównym celem jest działalność zarobkowa wynagradzana w odwiedzanej miejscowości.”¹

Oznacza to, że turystyką jest każda zmiana miejsca pobytu osoby na krócej niż rok o ile nie jest to wyjazd związany z podjęciem stałej pracy zarobkowej w danym miejscu docelowym. Istotne jest również, aby dany wyjazd i pobyt odbywały się dobrowolnie.

Osoby zmieniające swoje miejsce pobytu w jakimkolwiek zakresie nazywamy podróżnymi, natomiast tylko te osoby, które spełniają definicję podaną przez UNWTO są uczestnikami ruchu turystycznego. Takie osoby nazywane są odwiedzającymi. Odwiedzających można podzielić na dwie kategorie ze względu na długość pobytu. Są to odwiedzający jednodniowi oraz turyści – odwiedzający, który spędza przynajmniej jeden nocleg w miejscowości pobytu².

Produkt aplikacji internetowej wspomagającej planowanie wyjazdów turystycznych, którego projekt i implementacja jest celem tej pracy skupiać się będzie na zagadnieniu turysty i jego potrzeb. Turysta bowiem jest w założeniu głównym od-

¹ Terminologia turystyczna. Zalecenia WTO. Organizacja Narodów Zjednoczonych, World Tourism Organization, Instytut Turystyki, Warszawa 1995, s. 5–7

² Podstawy turystyki Tom 1. Barbara Cymańska-Garbowska, Barbara Steblik-Włazłak, WSiP Warszawa 2014, s. 11-12

biorcą i użytkownikiem przygotowywanego systemu a funkcjonalności aplikacji mają zapewnić mu pomoc przy planowaniu swoich wyjazdów.

2. Rola informatyki w turystyce

Wraz z rozwojem informatyki nastąpiła rewolucja w branży turystycznej. Wprowadziły ją Komputerowe Systemy Rezerwacyjne CRS (ang. Computer Reservation Systems). Używanie ich w biurach i agencjach turystycznych oraz rozszerzenie ich o wiele aspektów planowania podróży takich jak środek transportu, hotele czy inne bilety na atrakcje turystyczne spowodowało znaczny wzrost efektywności branży turystycznej poprzez zwiększenie elastyczności i poprawienie czasu transakcji między usługodawcą a klientem. Obecnie jest to jeden z najważniejszych elementów funkcjonowania planowania i organizacji ruchu turystycznego oraz obsługi turystów.

Wraz z rozwijającą się turystyką międzynarodową największe znaczenie mają systemy o zasięgu globalnym. Nazywane są one Globalnymi Systemami Dystrybucji GDS (ang. Global Distribution Systems)³.

Największe z nich są to

- AMADEUS – stworzony dzięki współpracy kilku dużych linii lotniczych takich jak francuskie Air France, niemiecka Lufthansa, hiszpańska Iberia i szwedzki SAS. Jest liderem GDS na rynku europejskim
- GALILEO – drugi najważniejszy GDS w Europie
- SABRE – uruchomiony przez American Airlines. Obejmuje głównie USA gdzie jest liderem w udzielaniu rezerwacji
- APOLLO – wykorzystywany przez takie linie lotnicze jak United Airlines, US Airlines, Air Canada, British Airlines lub SAS. Działa głównie na rynku USA

Wymienione GDS są systemami służącymi usługodawcom w branży turystycznej takim jak liniom lotniczym, sieciom hoteli oraz agencjom biur podróży.

³ Kuczek Zygmunt (red.), *Kompendium pilota wycieczek*, Proksenia Kraków 2005 s.22-23

Aby pomóc jednak bezpośrednio klientowi w wyborze odpowiedniej usługi, lotu czy hotelu dla siebie powstało również wiele aplikacji internetowych oraz mobilnych służących w celu wyszukiwania najlepszej oferty. Dzięki nim użytkownik bezpośrednio sam wyszukuje, decyduje i dokonuje rezerwacji bez udziału biura podróży czy wyspecjalizowanego agenta.

Większość aplikacji do wspomagania planowania wyjazdów turystycznych dostępnych dla użytkowników Internetu są to właśnie różnego rodzaju wyszukiwarki. Takie aplikacje często umożliwiają jednocześnie wyszukiwanie możliwych lotów, biletów autobusowych, wynajmu samochodu i rezerwacji pokoju hotelowego przy posiadaniu tylko jednego konta gdzie gromadzą się informacje o wszystkich tych rezerwacjach.

Istnieją również aplikacje, które w swojej funkcjonalności oferują także utworzenie terminarzy lub harmonogramów odnalezionych poprzez wyszukiwanie wydarzeń lub aktywności możliwych do wykonania w popularnych turystycznie miejscowościach.

Moim zdaniem niewiele jest jednak aplikacji dedykowanych bezpośrednio pod użytkownika chcącego zaplanować cały swój wyjazd niezależnie od jednej wyszukiwarki. Celem tej pracy jest stworzenie aplikacji, w której użytkownik ma dowolność w planowaniu swoich wyjazdów zarówno pod względem miejscowości docelowych, które nie muszą być znanymi miejscami turystycznymi jak i atrakcji i aktywności, których definicja zależy tylko od upodobań użytkownika. Jednocześnie aplikacja ma umożliwić czytelne zapisanie tak ważnych informacji jak rezerwacja lub plan rezerwacji miejsca noclegowego.

II. Specyfikacja wymagań

1. Wymagania biznesowe

Aplikacja ma na celu wspomóc użytkownika w planowaniu wyjazdów turystycznych zwanych tu również podróżą. Każdy zalogowany użytkownik może dodawać swoje wyjazdy i zarządzać ich szczegółami.

W ramach jednego wyjazdu użytkownik ma możliwość określić nazwę wyjazdu jego główny cel oraz daty, w których go planuje. Jeśli główny cel podróży jest miastem, regionem czy krajem użytkownik otrzyma podpowiedzi nazw tych lokalizacji przy rozpoczęciu uzupełniania pola formularza dodawania wyjazdu. Daty użytkownik ma możliwość wybrać z pojawiającego się kalendarza.

Każdy wyjazd może zawierać w sobie dodatkowo jeden lub wiele celów podróży. Wiąże się to z udostępnieniem użytkownikowi szerszych możliwości zarządzania planowanym wyjazdem.

Szczegółowe informacje dotyczące konkretnego celu podróży to jego nazwa oraz daty, w których użytkownik planuje przebywać w danym miejscu. Jeśli nazwa celu podróży jest nazwą miejscowości zostanie podpowiedziana poprzez mechanizm autouzupełniania. Dodatkowo w trakcie wyświetlania szczegółów celu zostanie wyświetlona mapa wycelowana na podaną miejscowość. Dodając daty przebywania w danym miejscu jest możliwość wybrania daty z kalendarza.

Z celem podróży łączą się również takie aspekty planowania jak informacje o noclegu oraz planowane atrakcje, które użytkownik chciałby wykonać przebywając w danym miejscu w trakcie swojego wyjazdu.

Informacje dotyczące noclegu, jakie użytkownik może zapisać w aplikacji to nazwa i cena, jest również miejsce na dowolne notatki użytkownika.

Atrakcja jest rozumiana, jako każda aktywność, którą zaplanuje sobie użytkownik. Jest ona definiowana przez swoją nazwę. Można jej również zapisać cenę oraz dodatkowe notatki. Atrakcja może być umiejscowiona w czasie poprzez swoją datę i godzinę rozpoczęcia oraz końca. Takie określenie atrakcji umożliwia ukazanie jej na terminarzu.

Terminarz jest przedstawiany, jako kalendarz w formacie tygodniowym lub miesięcznym według uznania użytkownika. Na terminarzu umiejscowione są wszystkie dodane atrakcje związane z celem podróży, które mają datę początkową i końcową. Użytkownik ma możliwość manipulowania terminarzem poprzez przesuwanie i rozciąganie graficznie ukazanych atrakcji na kalendarzu. Takie działania powodują zmianę daty i godziny zapisanych atrakcji.

Logowanie użytkownika do aplikacji jest zapewnione poprzez unikalny login użytkownika oraz odpowiadające mu hasło znane tylko użytkownikowi. Tylko zalogowany użytkownik może planować i zapisywać swoje wyjazdy w tej aplikacji.

Rejestracja konta użytkownika odbywa się poprzez podanie podstawowych danych takich jak email, login oraz podwójnie wpisane hasło. Login oraz email muszą być unikalne w systemie. Email powinien spełniać warunek składania się z nazwy użytkownika znaku '@' oraz części domenowej.

Aplikacja internetowa uzyskała nazwę „TripPlanner”. Jej interfejs użytkownika jest wyświetlany w języku angielskim.

2. Dziedzinowy słownik pojęć

Lista najczęściej używanych pojęć w systemie przedstawiona w formie słownika tłumaczącego wybrane pojęcia przez sposób opisowy.

Istnieje potrzeba stworzenia słownika pojęć, aby utrzymać integralność zrozumienia znaczenia tych pojęć tak samo przez każdą osobę mającą styczność z systemem lub niniejszą dokumentacją wytworzonego systemu.

Dodatkowo w poniższym słowniku zostanie zawarte tłumaczenie występujących haseł na język angielski, który jest językiem interfejsu użytkownika aplikacji, zarówno jak, w którym aplikacja została zaimplementowana

- **Aplikacja internetowa** – tu: aplikacja działająca poprzez przeglądarkę internetową służąca do wspomagania wyjazdów turystycznych, pojęcie występuje również pod nazwą utworzonej aplikacji „TripPlanner”
- **Użytkownik** – osoba korzystająca z aplikacji „TripPlanner”, użytkownik niezalogowany nie może korzystać z funkcjonalności aplikacji. (ang. User)

- **Logowanie** – autoryzacja zarejestrowanego użytkownika poprzez podanie loginu oraz hasła (ang. Log in)
- **Rejestracja** – utworzenie konta użytkownika poprzez podanie unikalnego loginu i adresu email oraz uzupełnienie hasła umożliwiającego późniejszą autoryzację użytkownika (ang. Register in)
- **Wyjazd turystyczny** – Zmiana miejsca pobytu użytkownika odbywająca się do określonego miejsca lub miejsc w określonym czasie. Posiada główny cel będący miejscowością, regionem lub krajem. Wymienna nazwa tego pojęcia to „Podróż” (ang. Trip)
- **Cel wyjazdu** – miejsce lub miejsca do których użytkownik się udaje w ramach wyjazdu i będzie przebywał przez określoną ilość czasu. Najczęściej określona jako nazwa miejscowości będąca destynacją lub jedną z destynacji podróży. Wymienna nazwa tego pojęcia to „Destynacja” (ang. Destination)
- **Nocleg** – miejsce w którym użytkownik zatrzyma się w celu spoczynku w danej destynacji wyjazdu. (ang. Lodging)
- **Atrakcja** – dowolna aktywność użytkownika w danej destynacji. Może być to wydarzenie, miejsce lub dowolnie inny plan spędzenia czasu przez użytkownika. Np. „Muzeum Narodowe” lub „Zwiedzanie starówki miasta”. Atrakcja jest definiowana poprzez jej nazwę oraz czas rozpoczęcia oraz zakończenia. Wymienna nazwa tego pojęcia to „Aktywność” lub „Wydarzenie” (ang. Event)
- **Terminarz** – zaplanowane przez użytkownika aktywności przedstawione w formie kalendarza z widokiem tygodniowym lub miesięcznym, na którym widnieją graficznie zaprezentowane aktywności w podanym dniu i godzinie rozpoczęcia i zakończenia wydarzenia. Wymienna nazwa tego pojęcia to „Kalendarz” (ang. Scheduler)

3. Wymagania funkcjonalne

- Wszystkie funkcjonalności oprócz rejestracji i logowania dostępne są tylko dla użytkownika zalogowanego
- Rejestracja to umożliwienie użytkownikowi założenia konta w aplikacji. Odbywa się to poprzez podanie unikalnego loginu, hasła oraz emaila użytkownika
- Logowanie zachodzi poprzez podanie unikalnego loginu i znanego tylko użytkownikowi hasła. Dzięki zalogowaniu użytkownik ma dostęp do wszystkich funkcji systemu a także do zapisanych wcześniej przez siebie danych i informacji
- Użytkownik ma możliwość wylogowania z aplikacji, aby chronić swoje dane przed innymi użytkownikami przeglądarki
- Funkcjonalność przeglądania swoich wyjazdów służy użytkownikowi do oglądania listy stworzonych przez siebie wyjazdów a także ich szczegółów, włączając w to mapę wycentrowaną na odpowiednie miejsce, jeśli główny cel wyjazdu jest miejscowością
- Szczegóły wyjazdu składają się z: nazwy wyjazdu, głównego celu wyjazdu, daty startu i daty końca
- Funkcjonalność zarządzania wyjazdem składająca się z trzech możliwości: dodania nowego wyjazdu, edycji wyjazdu lub usunięcia wyjazdu
- Funkcjonalność przeglądania celów wyjazdów służy użytkownikowi do oglądania listy stworzonych przez siebie celów danego wyjazdu a także ich szczegółów, włączając w to mapę wycentrowaną na odpowiednie miejsce, jeśli cel wyjazdu jest miejscowością
- Szczegóły celu wyjazdu składają się z nazwy celu wyjazdu i dat początkowej i końcowej
- Jeden wyjazd może mieć jeden lub wiele celów podróży

- Funkcjonalność zarządzania celami wyjazdu składająca się z trzech możliwości: dodania nowego celu wyjazdu do danej podróży, edycji konkretnego celu wyjazdu lub usunięcia celu wyjazdu z listy
- Użytkownik może edytować nocleg w destynacji wyjazdu
- Szczegóły noclegu składają się z nazwy noclegu, ceny i dodatkowych notatek użytkownika
- Jeden cel wyjazdu ma jeden nocleg
- Funkcjonalność przeglądania atrakcji zapewnia użytkownikowi przegląd listy dodanych przez siebie atrakcji w wybranej destynacji oraz szczegółów tych atrakcji
- Szczegóły atrakcji składają się z nazwy atrakcji, daty i godziny rozpoczęcia oraz końca, ceny oraz dodatkowych notatek
- Jeden cel wyjazdu może mieć wiele atrakcji
- Lista atrakcji w danej destynacji może być modyfikowana przez użytkownika poprzez dodawanie nowych atrakcji, edycję lub usuwanie istniejących atrakcji
- Przeglądanie terminarza odbywa się w formie przeglądania kalendarza z widokiem tygodniowym lub miesięcznym w zależności od preferencji użytkownika. Na danym kalendarzu widoczne są graficzne reprezentacje atrakcji umiejscowione w odpowiednim dniu i, jeśli jest to widok tygodniowy - godzinach. Użytkownik może zmieniać widok na poprzednie lub przyszłe tygodnie lub miesiące.
- Użytkownik ma możliwość zmiany dat i godzin atrakcji poprzez manipulowanie graficznymi reprezentacjami atrakcji na terminarzu. Może to nastąpić poprzez odpowiednie przesuwanie lub rozciąganie atrakcji zgodnie z intuicją obsługi interaktywnych kalendarzy
- Login musi być unikalny w systemie
- Nie można zarejestrować użytkownika podając email, który już wcześniej został użyty przy formularzu rejestracji
- Hasło jest kodowane poprzez algorytm base64

- Aby wydarzenie ukazało się w terminarzu jego data rozpoczęcia musi być wcześniejsza niż zakończenia
- Wszystkie pola wyboru lub zmiany daty są opatrzone kalendarzem, z którego można wybrać odpowiednią datę
- Pola formularza przedstawiające główny cel podróży lub nazwa jednego z listy celi podróży posiadają funkcjonalność autouzupełniania oferowaną przez firmę Google. Jeśli wpisywana wartość jest częścią nazwy miejscowości zostanie ona podpowiedziana użytkownikowi.
- Jeśli użytkownik wybrał podpowieź autouzupełniania nazwy miejscowości przy wyświetlaniu szczegółów celu podróży zostanie przedstawiona mapa wycentrowana na daną miejscowość.
- Liczba wyjazdów użytkownika jest nieograniczona
- Liczba destynacji jednego wyjazdu jest nieograniczona
- Liczba atrakcji jednego celu wyjazdu jest nieograniczona
- Nazwy są ograniczone do 30 znaków.
- Notatki są ograniczone do 256 znaków
- Daty są przedstawiane w formacie rok-miesiąc- dzień godzina : minuta
- Przy usunięciu wyjazdu usuwają się wszystkie jego informacje łącznie z celami wyjazdu i należącymi do nich szczegółami
- Przy usunięciu celu wyjazdu usuwają się wszystkie jego informacje łącznie z atrakcjami oraz noclegiem w celu wyjazdu
- Operacja usunięcia jest nieodwracalna
- Przed usunięciem wyjazdu oraz celu wyjazdu użytkownik zostanie poproszony o potwierdzenie chęci wykonania tego zadania
- Język interfejsu użytkownika to język angielski
- Ekran główny użytkownika niezalogowanego zawiera tylko zachętę do logowania lub rejestracji
- Ekran główny użytkownika zalogowanego zawiera zachętę do dodawania nowych wyjazdów i listę istniejących wyjazdów użytkownika

- Po wybraniu konkretnego wyjazdu użytkownikowi zostaną przedstawione szczegóły wyjazdu i lista celów wyjazdu.
- Po wybraniu konkretnego celu wyjazdu użytkownikowi zostaną przedstawione szczegóły celu wyjazdu.
- Ekran szczegółów wyjazdu będzie zawierał wszystkie informacje związane z danym celem wyjazdu w tym: szczegóły wyjazdu, listę atrakcji, terminarz i informacje o noclegu

4. Wymagania niefunkcjonalne

- Interfejs użytkownika dostosowujący się do wielkości okna przeglądarki lub urządzenia obsługującego przeglądarkę
- Możliwość obsługi wielu użytkowników jednocześnie
- Ochrona kont użytkowników: hasła kodowane algorytmem base64, token autoryzacyjny przydzielany metodą JWT
- Odporność na błędy użytkownika

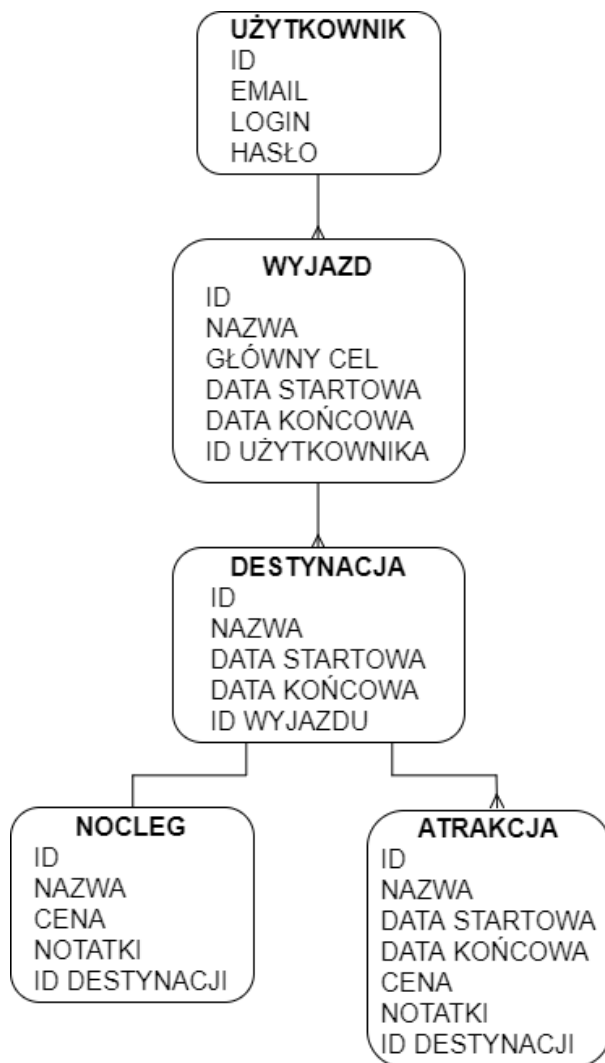
III. Projekt aplikacji

1. Diagram encji

Diagram encji przedstawia encje wykorzystywane w budowie i działaniu systemu. Pokazuje atrybuty konkretnych encji oraz ich relacje między encjami. Możliwe relacje to 1:1, 1:wielu lub wiele do wielu.

W przypadku tego systemu relacje wyglądają następująco:

- Użytkownik może mieć wiele wyjazdów.
- Wyjazd może zawierać wiele destynacji.
- Destynacja ma jeden nocleg
- Destynacja może zawierać wiele atrakcji



Rysunek III-1 Diagram encji. Źródło: Opracowanie własne

2. Diagram przypadków użycia

2.1. Aktorzy

Aktor – rola, którą pełni osoba lub system korzystająca z systemu w odwołaniu do przypadków użycia.

Jedynym aktorem w przypadku tego systemu jest Użytkownik. Niezalogowany użytkownik nie może korzystać z funkcjonalności systemu. Dopiero po rejestracji oraz zalogowaniu użytkownik może wykonywać dodatkowe akcje w systemie.

System nie zawiera administratora, ponieważ jest on poświęcony tylko i wyłącznie użytkownikowi- klientowi. Dowolny użytkownik sieci internetowej w założeniu może skorzystać z usług aplikacji „TripPlanner”. Działania użytkownika nie są w żaden sposób kontrolowane, nie ma również potrzeby wprowadzania administratora w celu zarządzania kontami użytkowników

2.2. Opis przypadków użycia

Przypadki użycia pokazują funkcjonalności oferowane przez system, które może wykonać aktor.

Przypadki użycia:

- **Rejestruj się** – umożliwienie użytkownikowi utworzenia konta w aplikacji, co da mu możliwość korzystania z funkcjonalności systemu które dostępne są tylko dla zalogowanych użytkowników. Rejestracja jest dokonywana poprzez podanie unikalnego loginu oraz emaila oraz podwójnego wpisania hasła użytkownika.
- **Loguj się** – autoryzacja użytkownika poprzez podanie unikalnego loginu oraz tylko znanemu użytkownikowi hasła. Umożliwia użytkownikowi korzystanie z wszystkich funkcjonalności systemu. Każdy zalogowany użytkownik otrzymuje unikalny token służący do autoryzacji.
- **Wyloguj się** – działanie mające na celu zabezpieczenie informacji podanych przez użytkownika przed niepowołanym dostępem innego człowieka lub sys-

temu w czasie gdy zainteresowany użytkownik nie korzysta z aplikacji osobście.

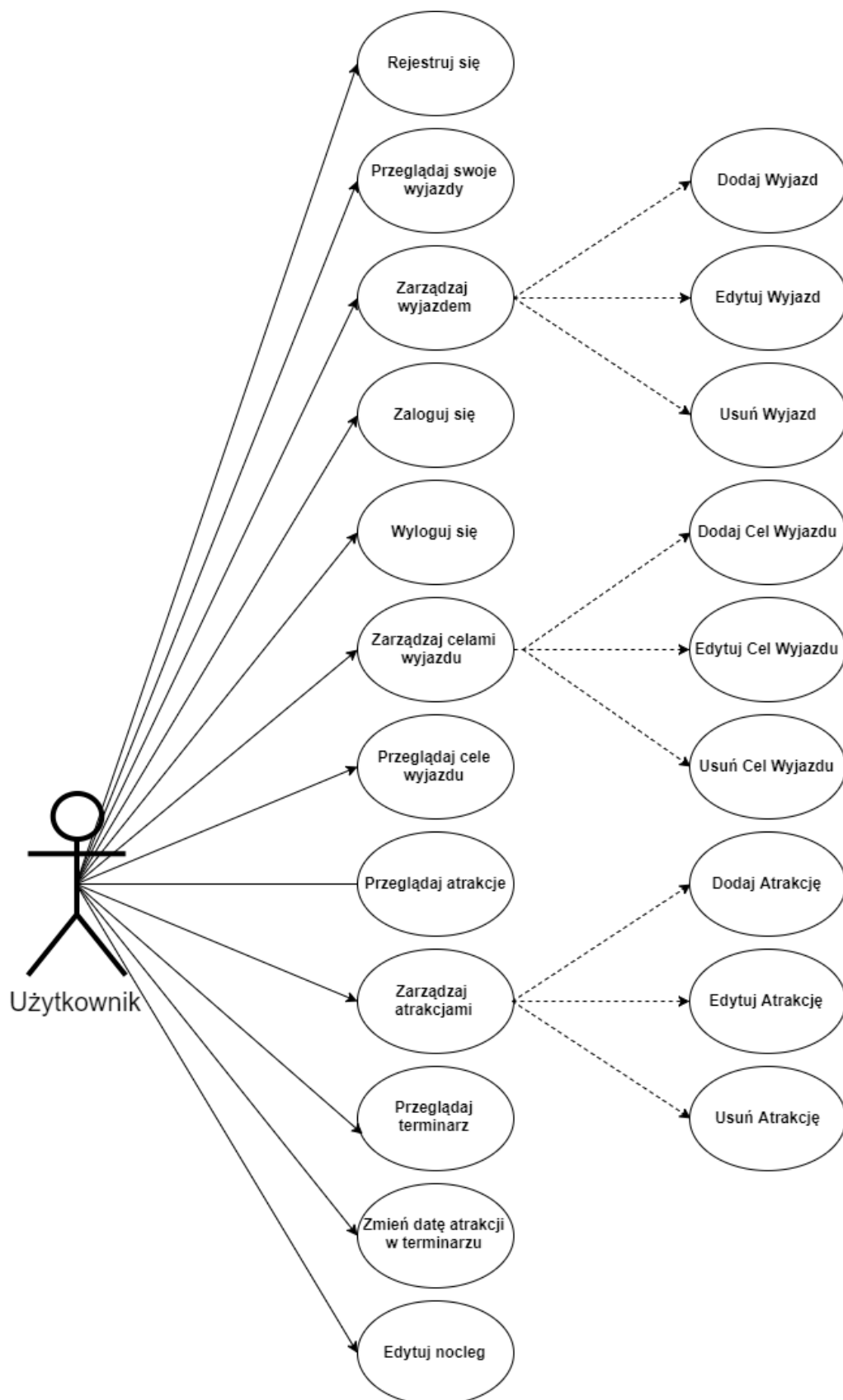
- **Przeglądaj swoje wyjazdy** – użytkownik może przeglądać listę dodanych przez siebie wyjazdów jak również oglądać szczegóły konkretnego wyjazdu łącznie z mapą wycelowaną na główny cel wyjazdu, jeśli jest ono nazwą miejscowości, regionu lub państwa.
- **Zarządzaj wyjazdem** – na tę funkcjonalność składają się trzy części:
 - Dodaj wyjazd – utworzenie nowego wyjazdu poprzez podanie nazwy, głównego celu podróży oraz przedziału dat, w których wyjazd ma się odbyć.
 - Edytuj wyjazd – edycja stworzonego wcześniej wyjazdu poprzez zmianę nazwy, głównego celu podróży oraz przedziału dat, w których wyjazd ma się odbyć.
 - Usuń wyjazd – usunięcie stworzonego wcześniej wyjazdu z listy wyjazdów użytkownika. Przy usunięciu wyjazdu usuwają się wszystkie informacje z nim powiązane w tym szczegóły wszystkich celów danego wyjazdu.
- **Przeglądaj cele wyjazdu** – użytkownik może przeglądać listę dodanych przez siebie destynacji wyjazdu dla danej podróży, może również oglądać szczegóły konkretnego celu wyjazdu łącznie z mapą wycelowaną na cel wyjazdu, jeśli jest ono nazwą miejscowości.
- **Zarządzaj celem wyjazdu** – na tę funkcjonalność składają się trzy części:
 - Dodaj cel wyjazdu – utworzenie nowego celu wyjazdu poprzez podanie nazwy np. miejscowości oraz przedziału dat, w których użytkownik planuje przebywać w danym miejscu.
 - Edytuj cel wyjazdu – edycja stworzonego wcześniej celu wyjazdu poprzez edycję nazwy oraz przedziału dat, w których użytkownik planuje przebywać w danym miejscu.
 - Usuń cel wyjazdu – usunięcie stworzonego wcześniej celu wyjazdu z listy celów wyjazdów należącej do danej podróży. Przy usu-

nięciu celu wyjazdu usuwane są również wszystkie informacje powiązane z danym celem wyjazdu w tym informacje o noclegu oraz atrakcjach w danym celu wyjazdu.

- **Przeglądaj atrakcje** – użytkownik może przeglądać listę dodanych przez siebie atrakcji w destynacji wyjazdu oraz szczegóły tych atrakcji.
- **Zarządzaj atrakcjami** – na tą funkcjonalność składają się trzy części:
 - Dodaj atrakcję – utworzenie nowej atrakcji poprzez podanie nazwy oraz daty startowej i końcowej, w których dana atrakcja ma się planowo rozpocząć oraz zakończyć. Użytkownik może również zanotować cenę powiązaną z atrakcją oraz dowolne inne notatki, które są mu potrzebne w związku z daną aktywnością.
 - Edytuj atrakcję – edycja stworzonej wcześniej atrakcji poprzez zmianę nazwy lub dat startowej i końcowej. Użytkownik może również zmienić cenę powiązaną z atrakcją oraz dowolne inne notatki, które są mu potrzebne w związku z daną aktywnością.
 - Usuń atrakcję – usunięcie utworzonej wcześniej atrakcji. Powoduje usunięcie wszystkich jej szczegółów oraz zniknie jej graficzna reprezentacja z terminarza.
- **Przeglądaj terminarz** – użytkownik może oglądać swój terminarz przedstawiony w formie kalendarza.
 - Użytkownik ma do wyboru dwa widoki terminarza: tygodniowy – przedstawiający siedem wybranych dni od soboty do niedzieli lub miesięczny – przedstawiający daty z jednego wybranego miesiąca.
 - Użytkownik może zmieniać widok na kolejne lub poprzednie tygodnie lub miesiące w zależności od ustawionego typu widoku kalendarza.
 - Na terminarzu użytkownik widzi graficznie przedstawione reprezentacje swoich Atrakcji ustawione w odpowiednim dniu. Dodatkowo w widoku tygodniowym wydarzenia ustawione są w odpowiednich godzinach w dniu. Ich reprezentacja rozpoczyna się w

rzędzie przedstawiającym godzinę startu a kończy na rzędzie przedstawiającym godzinę końcową atrakcji.

- **Zmień datę atrakcji w terminarzu** – Użytkownik może zmieniać datę lub godzinę startu i końca oraz czasu trwania wydarzenia poprzez reakcję z graficznie przedstawionymi wydarzeniami w swoim terminarzu celu wyjazdu.
 - Atrakcje przedstawione są w terminarzu, jako prostokąty ustawione w odpowiednim dniu oraz w przypadku widoku tygodniowego między odpowiednimi godzinami startu i końca wydarzenia. Użytkownik ma możliwość zmiany tych dat poprzez reakcję z grafikami.
 - Poprzez przesuwanie grafik użytkownik może zmieniać daty i godziny wydarzenia pozostawiając jego całkowity czas trwania niezmieniony.
 - Można również zmienić czas trwania wydarzenia poprzez rozciąganie i zmniejszanie reprezentującego je prostokąta. W przypadku widoku tygodniowego zmieniać godzinę startu lub końca. Lub w przypadku kalendarza z widokiem miesięcznym przedłużyć lub skrócić czas trwania wydarzenia o ilość pełnych dni.



Rysunek III-2 Diagram przypadków użycia. Źródło: Opracowanie własne

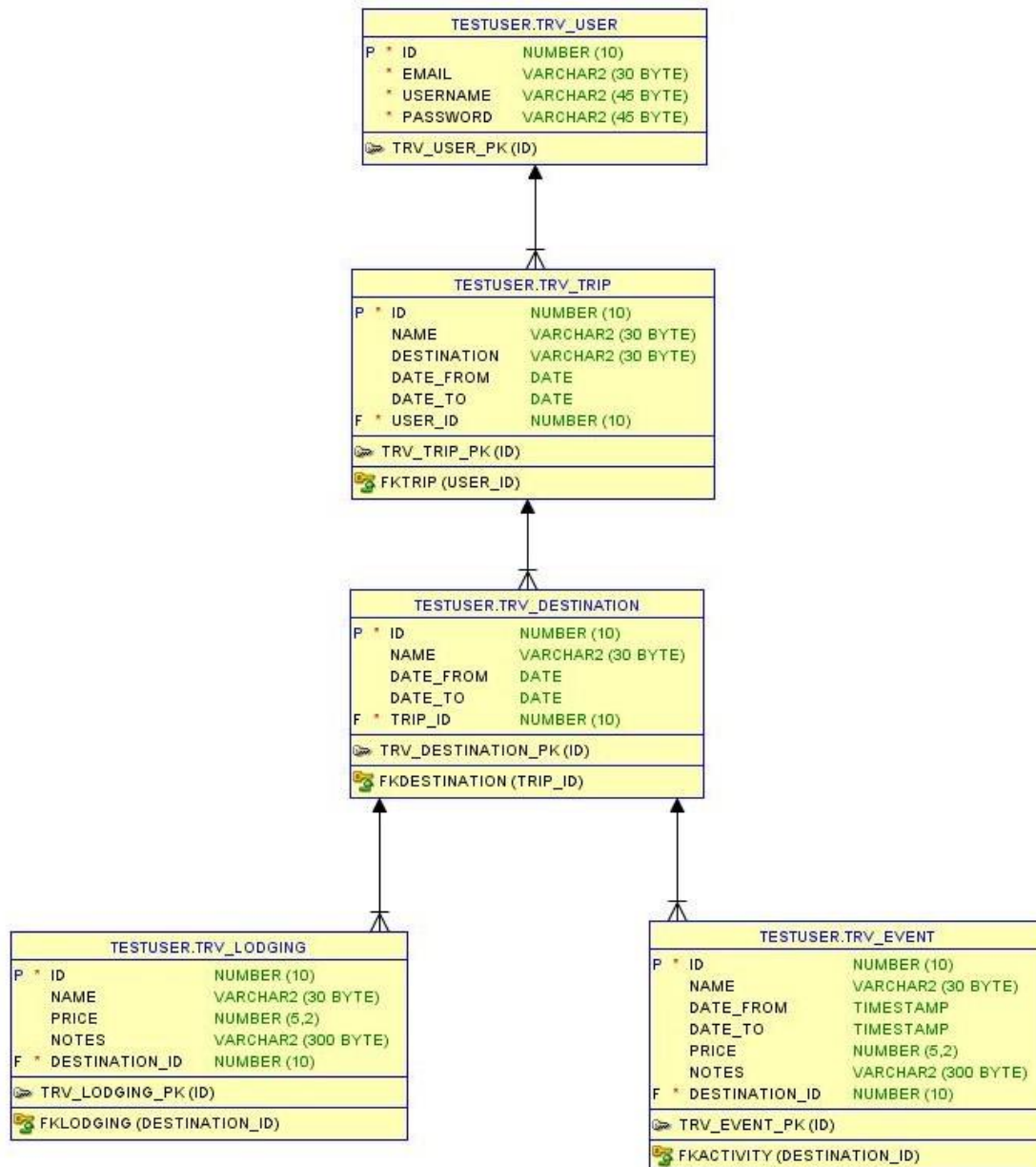
3. Baza danych

Wszystkie dane systemu są przetrzymywane na bazie danych Oracle. Na podstawie diagramu encji został stworzony model danych przechowywany w relacyjnej bazie danych. Encje w bazie danych zostały utworzone poprzez napisanie kodu wykorzystując zapytania SQL.

Klucze główne są tworzone automatycznie poprzez stworzone sekwencje. Klucze obce są to klucze główne innych encji.

Relacyjny diagram encji ERD (Rysunek III-3) jest przedstawiony w języku angielskim, ponieważ przy pomocy tego języka opisowego została zaimplementowana baza danych. I od tego momentu cały kod aplikacji jest napisany używając języka angielskiego.

Nazwy tabel oraz kolumn odpowiadają bezpośrednio nazwom encji oraz atrybutów w diagramie encji. Nazwy kolumn dodatkowo na modelu posiadają przedrostek „testuser”, jest to nazwa użytkownika bazy danych, w której przechowywane są dane tabele.



Rysunek III-3 Model ERD. Źródło: Opracowanie własne

4. Diagram klas

Rysunek III-4 przedstawia fragment diagramu klas. Na tym rysunku pominięto atrybuty i metody klas, aby zachować przejrzystość i czytelność diagramu.

Klasy te są stworzone tak, aby obiekty odpowiadały encjom w relacyjnej bazie danych. Dodatkowo w związku z podziałem aplikacji na warstwy została wprowadzona odpowiednia konwencja nazewnicza.

- Klasy z przyrostkiem Controller należą do warstwy prezentacji i służą do wystawiania punktów końcowych sieci. Są one kontrolerami Rest.

języku Java. Osobno zaś jest baza danych stworzona na oprogramowaniu firmy Oracle w języku SQL.

Główne warstwy podzielone są na trzy kolejne:

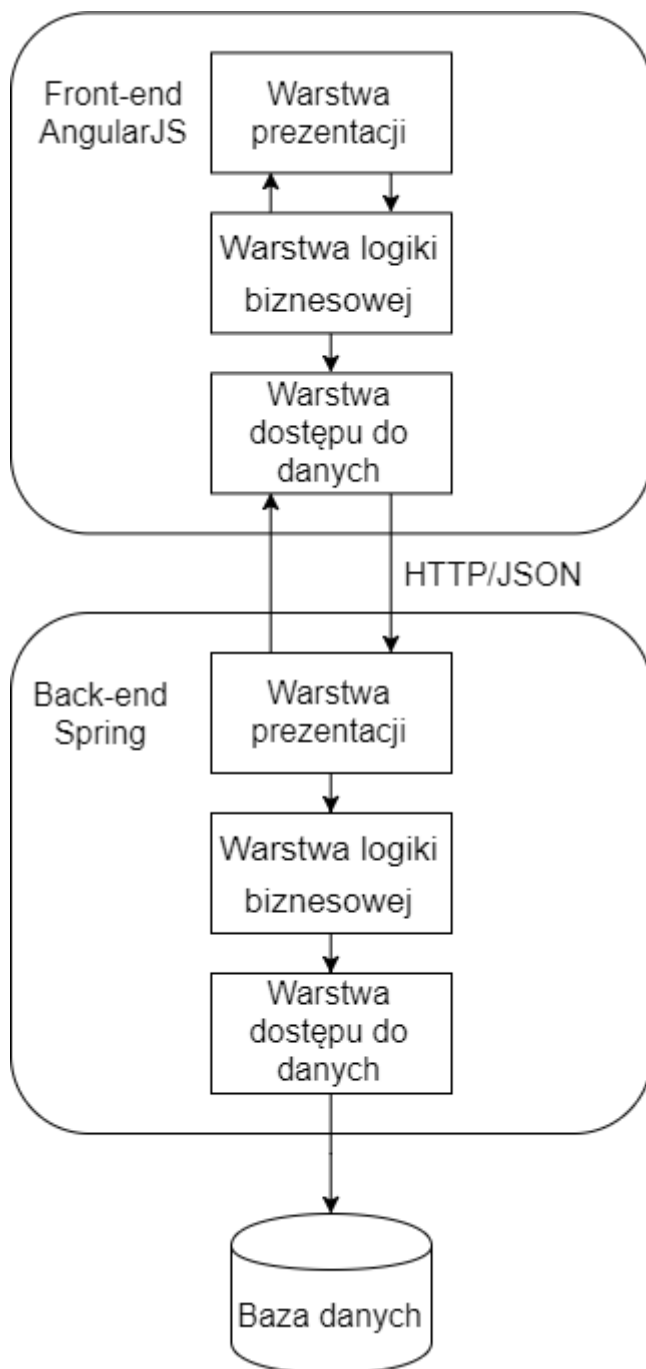
Frontend:

- Warstwa prezentacji – czyli widok – jest odpowiedzialna za prezentowanie interfejsu graficznego wraz ze wszystkimi danymi dla użytkownika w przeglądarce. Kontaktuje się tylko z warstwą logiki biznesowej wysyłając i odbierając dane dzięki dwustronnemu połączeniu zmiennych zapewnionym przez szkielet AngularJS.
- Warstwa logiki biznesowej – czyli kontrolery – są odpowiedzialne za odpowiednie przetwarzanie i tłumaczenie działań użytkownika oraz przekazywania do widoku odpowiednich danych zwróconych przez serwisy. Łączy się z warstwą prezentacji dzięki dwustronnemu połączeniu zmiennych zapewnionym przez szkielet AngularJS. Prosi o przetwarzanie żądań przez serwisy.
- Warstwa dostępu do danych – czyli serwisy – wykonują zapytania kontrolerów. Wysyłają żądania http do serwera aplikacji.

Backend:

- Warstwa prezentacji – definiuje, które serwisy logiki biznesowej odpowiadają którym zapytaniom http url oraz jakie parametry z żądań http zostaną odczytane.
- Warstwa logiki biznesowej – zawiera całą logikę biznesową, jakie dane zostaną przyjęte i jak przetworzone oraz jakie dane oraz w jakiej formie zostaną zwrócone.
- Warstwa dostępu do danych – zgodnie z wzorcem projektowym DAO warstwa ta łączy się z bazą danych i mapuje odpowiednie wartości uzyskane przez zapytania na obiekty. Udostępnia również mapowanie obiektowo-relacyjne które jest wykorzystywane do dodawania lub edycji rekordów w bazie danych. W tej warstwie pomaga technologia Hibernate

Rysunek III-5 przedstawia model reprezentujący podział na warstwy projektowanej aplikacji internetowej.



Rysunek III-5 Model podziału aplikacji na warstwy. Źródło: Opracowanie własne

6. Struktura drzewa plików aplikacji

W czasie implementacji serwer klienta został podzielony, zgodnie ze szkieletem AngularJS, na moduł główny zawierający wiele mniejszych modułów. W praktyce każdy widok, zwany panelem jest powiązany z osobnym modułem. W ten sposób

każdy widok posiada własny kontroler, który obsługuje działania użytkownika i przekazuje żądania do odpowiednich serwisów. Drzewo plików obsługiwanych przez środowisko Webstorm ponadto zostało podzielone względem tematyki. Inną metodę zastosowano do drzewa plików serwera aplikacji obsługiwanego przez IntelliJ, które jest podzielone na moduły zgodne z warstwami aplikacji. W obu przypadkach zostały zastosowane takie metody, które czynią kod aplikacji czytelniejszą.

IV. Opis implementacji

1. Użyte technologie i charakterystyka wykorzystanych narzędzi

1.1. Użyte wzorce projektowe

1.1.1. MVC

Wzorzec projektowy używany przy budowie aplikacji posiadających graficzny interfejs użytkownika. Zakłada podział aplikacji na trzy główne części⁴:

- Model – reprezentuje problem lub logikę aplikacji najczęściej, jako formę modelu danych w raz z operującymi na nich funkcjami wykorzystywanych przez logikę aplikacji, którymi zajmuje się problem będący tematyką aplikacji
- Widok – opisuje jak przekazać użytkownikowi, czyli jak wyświetlić na interfejsie graficznym wszystkie informacje, które chciałby i może zobaczyć klient aplikacji. Przekazuje działania użytkownika do kontrolera
- Kontroler – zarządza przejmując informacje od użytkownika przekazane przez widok, tłumaczy je i odpowiednio oddziałuje na model. Oraz pobiera informacje od modelu, co powinno być przedstawione poprzez widok biorąc pod uwagę działania użytkownika oraz wynik przetwarzania modelu..

1.2. Użyte technologie serwera aplikacji

1.2.1. Spring

Szkielet (ang. Framework) tworzenia aplikacji przy użyciu języka Java Enterprise Edition. Jest jednym z najbardziej znanych i darmowych frameworków w tym języku programowania. Głównymi zaletami Springa są⁵:

- Używanie wzorca projektowego Inversion of Control służącym na między innymi wstrzykiwaniu zależności. Czyli przeniesieniu sterowania wykony-

⁴ Yener Murat, Theedom Alex, *Professional Java® EE Design Patterns*, John Wiley & Sons, Inc. 2015

⁵ Walls Craig, *Spring w akcji*, Helion 2015

wania aplikacji na framework i tylko wywoływaniu kodu stworzonego przez programistę. Ułatwia to w znaczący sposób pisanie kodu w szczególności w dużym stopniu zmniejsza ilość kodu, który należy napisać.

- Wykorzystuje mechanizm JDBC Java Database Connectivity, aby ułatwić dostęp do danych przechowywanych w bazie danych. Dzięki takiemu podejściu nie trzeba pisać bezpośrednio zapytań sql. Zmniejsza to ilość kodu oraz ułatwia obsługę błędów.

1.2.2.Hibernate

Framework pomagający w tworzeniu warstwy dostępu do danych⁶. Zapewnia on mapowanie obiektowo-relacyjne, czyli tłumaczenie obiektów obiektowych zawierających w sobie referencje na inne obiekty na encje o charakterze relacyjnym.

Zintegrowany ze springiem poprzez warstwę dostępu do danych przy użyciu mechanizmu JDBC. Ułatwia pracę poprzez wbudowane metody.

1.2.3.Oracle Database

System zarządzania relacyjnymi bazami danych, na którym przy pomocy języka SQL została stworzona baza danych aplikacji. Z bazami danych Oracle⁷ związane są między innymi dwie aplikacje do zarządzania nimi: Oracle SQL Developer oraz Oracle SQL Developer Data Modeler. Te dwa programy współpracują ze sobą umożliwiając stworzenie schematu bazy danych z ERD Entity-Relationships diagram modelu związków encji lub odwrotnie za pomocą programowania zwrotnego – z istniejącej już bazy danych można stworzyć Entity-Relationships Diagram.

1.2.4.Swagger

Framework ułatwiający dokumentację i testowanie wystawianych usług serwera aplikacji⁸. Poprzez wstrzyknięcie go do aplikacji w łatwy sposób można kontrolować poprawność działania wystawianych usług. Dodatkowo automatycznie tworzy dokumentację usług oraz informuje, jakie parametry są przyjmowane przez usługę, np. JSON o jakiej strukturze powinien zostać przesłany.

⁶ Yogesh Prajapati, *Java Hibernate Cookbook*, Packt Publishing 2015

⁷ McLaughlin Michael, *Oracle Database 12c. Programowanie w języku PL/SQL*, Helion 2015

⁸ <https://swagger.io>

1.2.5.IntelliJ IDEA

Środowisko programistyczne dla języka Java⁹. Program niezwykle przyjazny dla dewelopera. Obsługuje takie technologie i frameworki jak Spring i Hibernate oraz współpracuje z systemem kontroli wersji GIT

1.3.Użyte technologie serwera klienta

1.3.1.AngularJS

Framework skierowany dla serwera klienta oparty na języku JavaScript¹⁰. Wspiera wykorzystywanie wzorca Model-Widok-Kontroler w aplikacjach internetowych. Jego celem jest oddzielenie manipulacji w widoku aplikacji od jej logiki.

- AngularJS wykorzystuje dwukierunkowe wiązanie danych, które polega na wyłapywaniu zmian w modelu w widoku i przekazywaniu ich do kontrolera. Oraz odwrotnie, jeśli zmiany na modelu zostaną wykonane za interwencją kontrolera widok zostanie o tym poinformowany i się zmieni.
- Ten framework zapewnia również dużą ilość użytecznych tagów html, dzięki którym widok staje się bardziej responsywny na działania użytkownika lub zmiany w modelu.

1.3.2.Bootstrap

Framework CSS zawierający wiele narzędzi służących do łatwiejszego tworzenia interfejsu graficznego aplikacji internetowych¹¹. Służy głównie do stylizacji elementów występujących w widoku stron internetowych za pomocą używania gotowych klas.

1.3.3.HTML5

Hipertekstowy język znaczników HyperText Markup Language wykorzystywany do tworzenia dokumentów takich jak strony internetowe umożliwiając dodawanie takich elementów jak hiperłącza, paragrafy, listy lub multimedia¹².

⁹ Hudson Orsine Assumpcao, *Getting Started with IntelliJ Idea*, Packt Publishing 2013

¹⁰ Kalbarczyk Dariusz, Kalbarczyk Arkadiusz, *Angular JS. Pierwsze kroki*, Helion 2015

¹¹ Rahman Syed Fazle, *Bootstrap. Tworzenie interfejsów stron WWW. Technologia na start*, Helion 2015

¹² MacDonald Matthew, *HTML5. Nieoficjalny podręcznik*, Helion 2013

1.3.4.NodeJS

Środowisko uruchomieniowe aplikacji internetowych w tym głównie serwerów klienta stworzonych w języku JavaScript¹³. Wspomaga aplikacje napisane w idei programowania zdarzeniowego z asynchronicznym systemem wejścia-wyjścia. Dodatkowo jest dobrym managerem pakietów współpracującym przez Npm.

1.3.5.Bower

Menager pakietów klienckich¹⁴. Ułatwia prostą instalację różnych paczek, które można następnie wstrzyknąć do tworzonej aplikacji.

1.3.6.Webstorm

Środowisko programistyczne dla języka JavaScript¹⁵. Umożliwia również pisanie kodu w językach HTML oraz CSS. Współpracuje z frameworkiem AngularJS, środowiskiem Node.js oraz systemem kontroli wersji GIT.

1.3.7.Dodatkowe biblioteki

Opis wybranych dodatkowych bibliotek powiązanych z frameworkiem AngularJS:

- **Route** – głównym celem jest łączenie linków URL do kontrolerów oraz widoków HTML. Serwis \$route obserwuje lokalizacje url i mapuje ścieżkę. Biblioteka ta udostępnia również serwis pozwalający na pobieranie parametrów z adresu URL
- **Cookies** – umożliwia dostęp do czytania i zapisu ciasteczek przeglądarki. Ciasteczka służą do zapisywania informacji na temat bieżącej sesji przeglądarki, bardzo często informacje o użytkowniku i jego działaniach.
- **Materials**¹⁶ – biblioteka komponentów interfejsu graficznego zawierająca dużą ilość użytecznych komponentów możliwych do użycia zarówno, jako znaczniki html jak i wspaniale wspomagających responsywność widoku aplikacji. Biblioteka ta wykorzystuje inne biblioteki takie jak Animate do

¹³ Basarat Ali Syed, *Beginning Node.js*, Springer 2014

¹⁴ <https://bower.io>

¹⁵ Rosca Stefan, Patin Den, *WebStorm Essentials*, Packt Publishing 2015

¹⁶ <https://material.angularjs.org>

generowania animacji swoich komponentów oraz Messages do wykonywania między innymi wyskakujących okienek, dialogów i wiadomości.

- **Map**¹⁷ – biblioteka przedstawiająca dyrektywy używające Google Maps API. Biblioteka ta stworzona bezpośrednio pod AngularJS przedstawia proste tagi html zapewniające szybkie zaprezentowanie mapy google z dowolnymi parametrami, takimi jak miejsce wycentrowania, stopień zbliżenia czy nawet typ mapy.
- **Daypilot**¹⁸ – komponent html oferujący dynamiczny terminarz. Zawiera wsparcie w języku JavaScript. Komponent ten umożliwia zaprezentowanie kalendarza w formie przedstawiającej różne okresy czasu, takie jak miesiąc, tydzień lub dzień. Można na nim przedstawiać wydarzenia zawarte między konkretnymi dniami lub godzinami. Biblioteka wspiera dynamiczną współpracę z użytkownikiem poprzez pozwolenie mu na interakcję z wyświetlonymi wydarzeniami w kalendarzu. Użytkownik ma możliwość przeciągać wydarzenia na inną datę oraz rozciągać je zmieniając ramy czasowe wydarzeń. To wszystko współpracuje z techniką AJAX (asynchronicznym JavaScript i XML) pozwalając na zapisanie każdej wprowadzonej w ten sposób zmiany na wydarzeniu.

1.4. System kontroli wersji GIT

Oprogramowanie pomagające w śledzeniu zmian w tworzonych dokumentach tekstowych. Umożliwia w łatwy sposób kontrolowanie zmian oraz łączeniu ich z dobrze działającym kodem.

Program GIT najlepiej sprawdza się przy pracy grupowej, kiedy to najbardziej pożądanym jest rozdzielenie i zrównoleglenie zadań przez programistów. Jednak przy jednoosobowym projekcie również jest przydatny, aby kontrolować postępy w pracy i w razie potrzeby udostępnia szybki powrót do ostatniej poprawnie działającej wersji aplikacji.

¹⁷ <https://ngmap.github.io>

¹⁸ <https://www.daypilot.org>

2. Usługi aplikacji

Aplikacja poprzez interfejs graficzny działający w przeglądarce internetowej udostępnia usługi serwera. Usługi te zostały pogrupowane według ustalonych encji.

Występuje, więc pięć podgrup:

- Użytkownik (User)
 - Register – usługa typu POST - służy do rejestracji użytkownika, zwraca status 200 gdy rejestracja zakończyła się sukcesem
 - Login – usługa typu POST - służy do logowania użytkownika – zwraca id użytkownika, jego login oraz unikalny token
 - checkUsername – usługa typu GET – służy do sprawdzenia unikalności loginu podanego w parametrze. Zwraca status 200, gdy login jest unikalny. Zwraca status 409, gdy taki login już istnieje w bazie
 - checkEmail – usługa typu GET – służy do sprawdzenia unikalności emaila podanego w parametrze. Zwraca status 200, gdy email nie występuje w bazie. Zwraca status 409, gdy taki login już istnieje w bazie
- Wyjazd (Trip)
 - getUserTrips/{user_id} - usługa typu GET – zwracająca wszystkie wyjazdy należące do użytkownika, którego id zostało podane w parametrze user_id
 - getTrip/{id} - usługa typu GET – zwraca szczegóły wyjazdu o danym id podanym w parametrze
 - addUpdateTrip – usługa typu POST – W parametrze przyjmuje obiekt w formacie JSON przekazujący informacje o wyjeździe. Jeśli wyjazd o takim id już istnieje w bazie zostanie on uaktualniony, jeśli taki klucz główny jeszcze nie istnieje rekord z podanymi danymi zostanie utworzony w tabeli
 - deleteTrip/{id} - usługa typu GET – powoduje usunięcie z tabeli w bazie danych rekordu wyjazdu o podanym w parametrze id

- Cel wyjazdu (Destination)
 - `getTripDestinations/{trip_id}` - usługa typu GET – zwracająca wszystkie destynacje wyjazdu, którego id zostało podane w parametrze `trip_id`
 - `getDestination/{id}` - usługa typu GET – zwraca szczegóły destynacji o danym id podanym w parametrze
 - `addUpdateDestination` – usługa typu POST – W parametrze przyjmuje obiekt w formacie JSON przekazujący informacje o destynacji. Rekord w bazie danych zostanie dodany lub uaktualniony w zależności czy już istnieje.
 - `deleteDestination/{id}` - usługa typu GET – powoduje usunięcie z tabeli w bazie danych rekordu wyjazdu o podanym w parametrze id
- Nocleg (Lodging)
 - `getDestinationLodging/{destination_id}` - usługa typu GET – zwracająca informacje o noclegu danego wyjazdu, którego id zostało podane w parametrze `trip_id`
 - `getLodging/{id}` - usługa typu GET – zwraca szczegóły noclegu o danym id podanym w parametrze
 - `addUpdateLodging` – usługa typu POST – W parametrze przyjmuje obiekt w formacie JSON przekazujący informacje o noclegu, rekord noclegu zostanie uaktualniony w bazie
 - `deleteLodging/{id}` - usługa typu GET – powoduje usunięcie z tabeli w bazie danych rekordu wyjazdu o podanym w parametrze id
- Atrakcja (Event)
 - `getDestinationEvents/{destination_id}` - usługa typu GET – zwracająca informacje o atrakcjach danego wyjazdu, którego id zostało podane w parametrze `destination_id`
 - `getEvent/{id}` - usługa typu GET – zwraca szczegóły atrakcji o danym id podanym w parametrze
 - `addUpdateEvent` – usługa typu POST – W parametrze przyjmuje obiekt w formacie JSON. Jeśli rekord o takim id już istnieje w bazie

zostanie on uaktualniony, jeśli taki klucz główny jeszcze nie istnieje
 rekord z podanymi danymi zostanie utworzony w tabeli

- deleteEvent/{id} - usługa typu GET – powoduje usunięcie z tabeli w bazie danych rekordu atrakcji o podanym w parametrze id

3. Szczegółowy opis implementacji wybranej usługi

Na podstawie jednej wybranej funkcjonalności przedstawię cały schemat działania aplikacji przechodząc przez wszystkie warstwy. Jako przykładową funkcjonalność zostało wybrane dodawanie nowego wyjazdu.

W założeniach początkowych użytkownik jest zalogowany.

Użytkownik z widoku panelu głównego klika na obiekt przycisku

```
<a href="" class="btn btn-primary" data-toggle="modal" data-  
target="#addEditTrip" ng-click="addDestination()">Start planning a  
new trip</a>
```

Wykonuje się funkcja „addDestination()” zawarta w kontrolerze która wysyła informacje „addTrip” do \$rootScope o zasięgu dostępnym dla całej aplikacji

```
$scope.addDestination = function () {  
    $rootScope.$broadcast('addTrip');  
}
```

Informację tą łapie kontroler modułu dodawania i edycji wyjazdu i ustawia atrybuty tamtejszej zmiennej tak, aby była pusta i posiadała user_id użytkownika, który ją wywołał pobierany z ciasteczka.

```
$rootScope.$on('addTrip', function () {  
    $scope.trip={  
        "id": 0,  
        "name": "",  
        "date_from": "",  
        "date_to": "",  
        "user_id": $cookies.get("userId")  
    }  
});
```

Otwiera się okno modalne, w którym użytkownik podaje Nazwę, Główny cel, który może wybrać z autouzupełniającej się podpowiedzi oraz dat, które może wybrać z pojawiających się kalendarzy.

```

<div class="modal-body">
  <form>
    <div class="form-group">
      <div class="row">
        <label>Name: </label>
        <input type="text" ng-model="trip.name"/> <br/>
      </div><div class="row">
        <label>Main destination: </label>
        <input places-auto-complete size=80
          ng-model="trip.destination"
          types="['(cities)', region]"
          on-place-changed="placeChanged()" /> <br/>
      </div>
    </div>
    <div layout-gt-xs="row">
      <div flex-gt-xs>
        <label>Date from:</label>
        <md-datepicker ng-model="trip.date_from" name="dateFrom"
          md-placeholder="Enter date"></md-datepicker>
      </div>
      <div flex-gt-xs>
        <label>Date to :</label>
        <md-datepicker ng-model="trip.date_to" md-placeholder="Enter
date"
          md-min-date="ctrl.dateFrom"></md-datepicker>
      </div>
    </div>
  </form>
</div>
<div class="modal-footer">
  <button type="button" class="btn btn-default" data-
dismiss="modal">Close</button>
  <button type="button" class="btn btn-primary" data-dismiss="modal" ng-
click="saveTrip()">Save changes</button>

```

Następnie użytkownik zapisuje swoje zmiany, którą to akcję przejmuje kontroler modułu.

Kontroler włącza serwis \$http, który to wysyła żądanie POST na url serwera aplikacji załączając, jako nagłówek token dostępu oraz parametr ze zmiennymi, które podał użytkownik w formacie JSON.

```

$scope.saveTrip = function() {
  var req = {
    method: 'POST',
    url: "http://localhost:8080/api/trip/addUpdateTrip",
    headers: {"access_token": $cookies.get("access_token")},
    data: $scope.trip
  };
  $http(req).then(function(data) {
    $rootScope.$broadcast('tripAdded');
  });
}

```

W tym momencie żądanie przejmuje serwer aplikacji w warstwie prezentacji. Na-

stępuje to w REST kontrolerze zmapowanym na odpowiednie url. Jest to endpoint definiujący usługę i jej przyjmowane parametry.

```
@RestController
@RequestMapping(value="/api/trip")
public class TripController {

    @Autowired
    private TripService tripService;

    @RequestMapping(value = "/addUpdateTrip", method = RequestMethod.POST)
    public void updateTrip(@RequestBody TripDto tripDto) {
        tripService.addUpdateTrip(tripDto);
    }
}
```

Metoda kontrolera przyjmuje parametry zgodne z atrybutami obiektu TripDto z warstwy logiki biznesowej.

```
@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class TripDto {
    private Long id;
    private String name;
    private String destination;
    private Date date_from;
    private Date date_to;
    private Long user_id;
}
```

Następnie uruchamiana jest metoda addUpdateTrip() serwisu TripService

```
public void addUpdateTrip(TripDto tripDto) {
    User user = userRepository.findById(tripDto.getUser_id());
    Trip trip = tripAssembler.fromDto(tripDto);
    trip.setUser(user);
    Trip savedTrip = tripRepository.save(trip);
}
```

Ta natomiast wywołuje metodę udostępnioną przez Hibernate w warstwie dostępu do danych, aby znaleźć i zmapować użytkownika o odpowiednim id. Jest to potrzebne, ponieważ JSON przechowuje tylko informację o id użytkownika a obiekty przechowują referencje na inne obiekty.

```
@Repository
public interface UserRepository extends JpaRepository<User, Long> {
    User findById(Long id);
}
```

Następnie tworzony obiekt za pomocą wzorca budowniczego.

```

@Override
public Trip fromDto(TripDto input) {
    return Trip.builder()
        .id(input.getId())
        .name(input.getName())
        .destination(input.getDestination())
        .date_from(input.getDate_from())
        .date_to(input.getDate_to())
        .build();
}

```

Typ skonstruowanego obiektu jest typem opartym bezpośrednio na tabeli w bazie danych.

```

@Entity
@Table(name="TRV_TRIP")
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class Trip implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
"SEQ_TRV_TRIP")
    @SequenceGenerator(name = "SEQ_TRV_TRIP", sequenceName = "SEQ_TRV_TRIP")

    @Column(name = "ID")
    private Long id;

    @ManyToOne
    private User user;

    @Column(name = "NAME")
    private String name;

    @Column(name = "DESTINATION")
    private String destination;

    @Column(name = "DATE_FROM")
    private Date date_from;

    @Column(name = "DATE_TO")
    private Date date_to;
}

```

Pozwala to na mapowanie obiektowo-relacyjne i zapisanie obiektu na rekord w bazie danych używając prostej metody save().

4. Interfejs graficzny aplikacji

4.1. Single-page app

Interfejs graficzny aplikacji został stworzony zgodnie z praktyką single-page app, co oznacza, że przeglądarka tylko raz, na początku ładuje cały widok strony. Następnie przy jakiegokolwiek zmianie lub działaniu użytkownika zostaje uaktual-

niony tylko ten moduł lub fragment, na którym zmiana nastąpiła. Zmniejsza to w znaczący sposób ilość żądań i informacji przesyłanych do przeglądarki z serwera klienta a za tym zwiększa prędkość działania aplikacji.

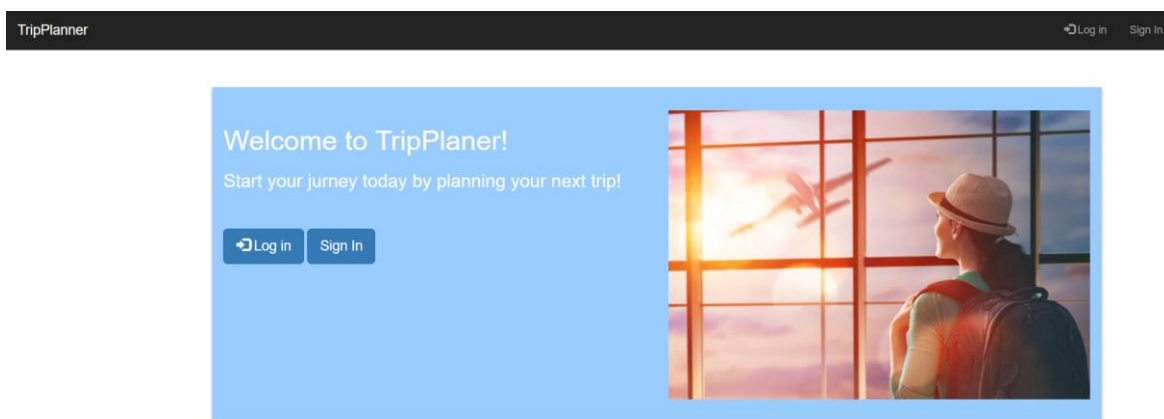
4.2. Wybrane panele interfejsu graficznego

W tym podrozdziale zostaną przedstawione wybrane panele lub fragmenty paneli interfejsu graficznego aplikacji, które użytkownik widzi w oknie przeglądarki.

Interfejs został stworzony z myślą, aby był czytelny i łatwy w użyciu przez wszystkich użytkowników Internetu. Jest on responsywny oraz odporny na błędy użytkownika. Został on napisany w najbardziej powszechnym języku w Internecie, jakim jest język angielski¹⁹.

Jako, że w założeniu dokumentacja będzie w wersji drukowanej monochromatycznie, więc dla uzyskania lepszej czytelności rysunków wyłączono obraz tła w interfejsie graficznym aplikacji na czas tworzenia zrzutów ekranu potrzebnych do załączenia w niniejszej dokumentacji aplikacji.

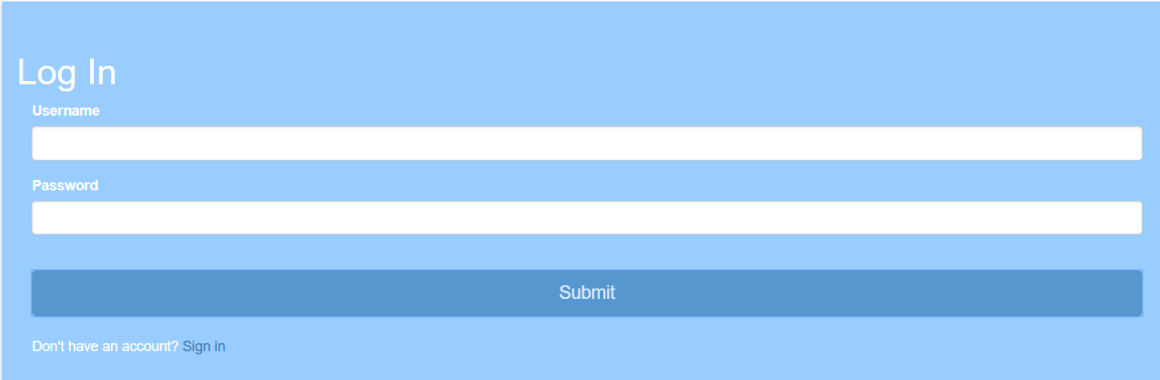
Grafiki użyte w aplikacji pochodzą ze strony <http://freedesignfile.com> i posiadają licencję Uznanie autorstwa 3.0, które udziela prawa na kopiowanie i rozpowszechnianie utworu w dowolnym medium i formie przy zachowaniu odpowiedniego oznakowania.



Rysunek IV-1 Interfejs graficzny: Ekran główny niezalogowanego użytkownika Źródło: Opracowanie własne

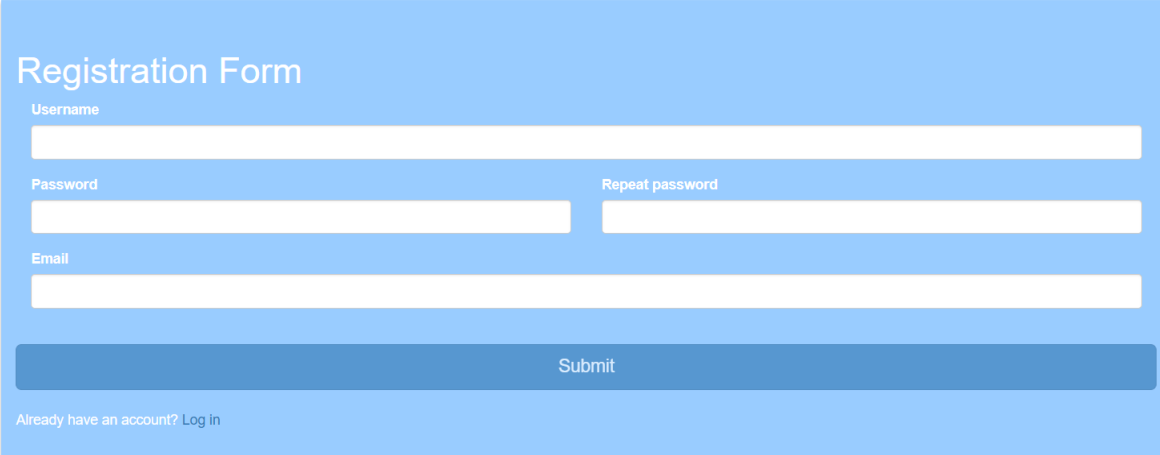
¹⁹ https://w3techs.com/technologies/overview/content_language/all

Główny ekran użytkownika zalogowanego (Rysunek IV-1) prezentuje zachętę do skorzystania z aplikacji poprzez zalogowanie lub rejestrację.

The image shows a 'Log In' form with a light blue background. At the top left, the text 'Log In' is displayed in a large, white font. Below it, there are two white input fields: the first is labeled 'Username' and the second is labeled 'Password'. A dark blue 'Submit' button is centered below the password field. At the bottom left, there is a link that says 'Don't have an account? Sign in'.

Rysunek IV-2 Interfejs graficzny: Formularz logowania Źródło: Opracowanie własne

W formularzu logowania (Rysunek IV-2) użytkownik musi dokonać interakcji z zarówno polem do wprowadzania loginu jak i hasła, aby uaktywnić przycisk zatwierdzający i dokonujący logowania.

The image shows a 'Registration Form' with a light blue background. At the top left, the text 'Registration Form' is displayed in a large, white font. Below it, there are three white input fields: 'Username', 'Password', and 'Email'. The 'Password' field is followed by a 'Repeat password' field. A dark blue 'Submit' button is centered below the email field. At the bottom left, there is a link that says 'Already have an account? Log in'.

Rysunek IV-3 Interfejs graficzny: Formularz rejestracji Źródło: Opracowanie własne

W formularzu logowania (Rysunek IV-3) użytkownik musi uzupełnić wszystkie pola, aby uaktywnić przycisk zatwierdzający i dokonujący rejestracji. Dodatkowo pola te muszą być wypełnione prawidłowo pod względem ilości wprowadzanych znaków. Dodatkowo, jeśli hasło powtórzone nie zgadza się z pierwszym hasłem lub email nie spełnia schematu standardowego adresu email użytkownikowi zostanie wyświetlona wiadomość pod polem formularza o rodzaju błędu. Jeśli okaże się, że dany login lub email istnieją już w systemie użytkownik otrzyma o tym informację i będzie zmuszony edytować dane pole w formularzu.



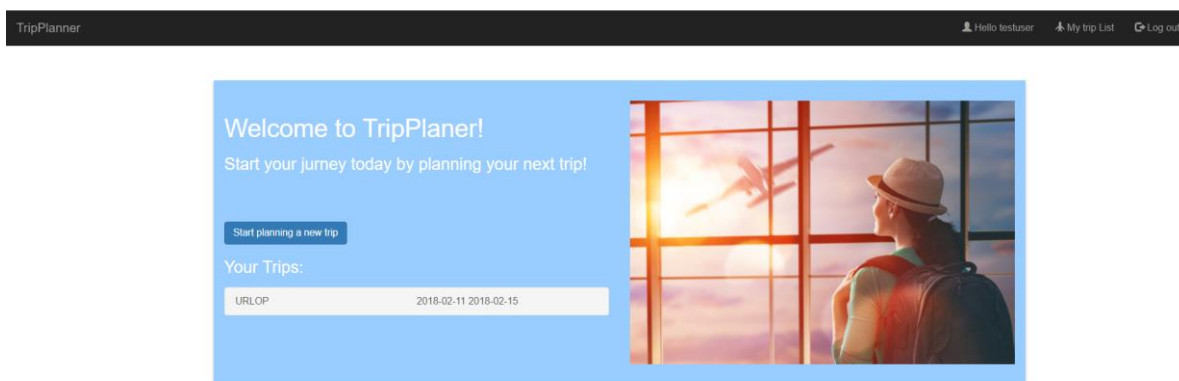
Rysunek IV-4 Interfejs graficzny: Menu główne niezalogowanego użytkownika Źródło: Opracowanie własne



Rysunek IV-5 Interfejs graficzny: Menu główne zalogowanego użytkownika Źródło: Opracowanie własne

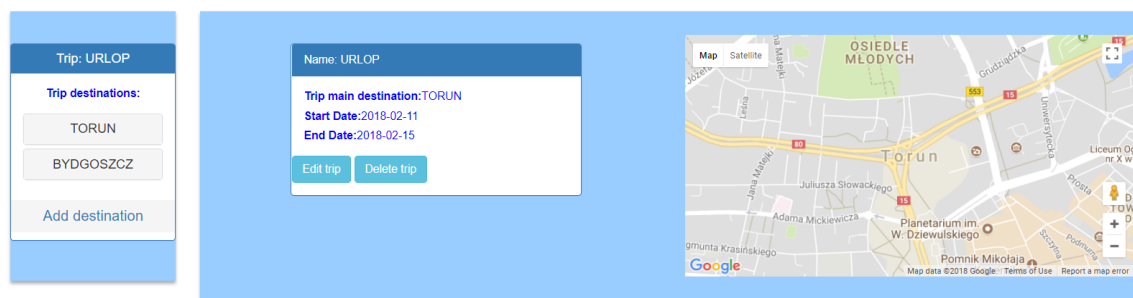
W górnej jego części interfejsu znajduje się menu, które zachowuje swoją pozycję względem okna przeglądarki w czasie przewijania w dół zawartości strony. Jest ono widoczne niezależnie od tego czy użytkownik aplikacji jest zalogowany czy też nie.

Jednak w zależności od tego faktu, przed zalogowaniem użytkownik będzie miał tylko dwie możliwości na głównym menu: Zalogować się lub Zarejestrować się (Rysunek IV-4). Te przyciski prześlą go do odpowiednich paneli z formularzami. Natomiast użytkownik zalogowany otrzyma przycisk kierujący go na ekran główny lub powodujący wylogowanie z aplikacji (Rysunek IV-5). Dodatkowo zawsze widoczne jest logo po lewej stronie przekierowujące na stronę główną.



Rysunek IV-6 Interfejs graficzny: Ekran główny zalogowanego użytkownika Źródło: Opracowanie własne

Główny ekran użytkownika zalogowanego (Rysunek IV-6) oferuje mu podgląd listy jego wyjazdów, które po kliknięciu przekierują go w szczegóły danego wyjazdu. Użytkownik na tym ekranie otrzymuje również zachętę i przycisk do stworzenia nowych wyjazdów.



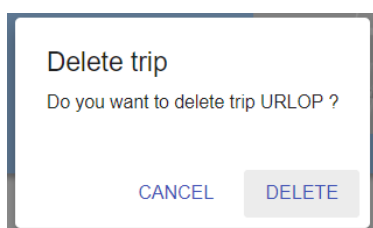
Rysunek IV-7 Interfejs graficzny: Szczegóły wyjazdu Źródło: Opracowanie własne

Rysunek IV-7 przedstawia panel szczegółów wyjazdu a także menu boczne przedstawiające listę celów danego wyjazdu.

Do menu bocznego dotyczącego listy celów wyjazdu została zastosowana podobna funkcjonalność jak do menu głównego. Menu to pozostaje w tym samym miejscu względem okna przeglądarki w trakcie przewijania szczegółów celu wyjazdu. Menu to przedstawia Listę celów danego wyjazdu wraz z opcjami przełączania paneli i jest ono widoczne, gdy użytkownik wybrał jeden z jego wyjazdów.

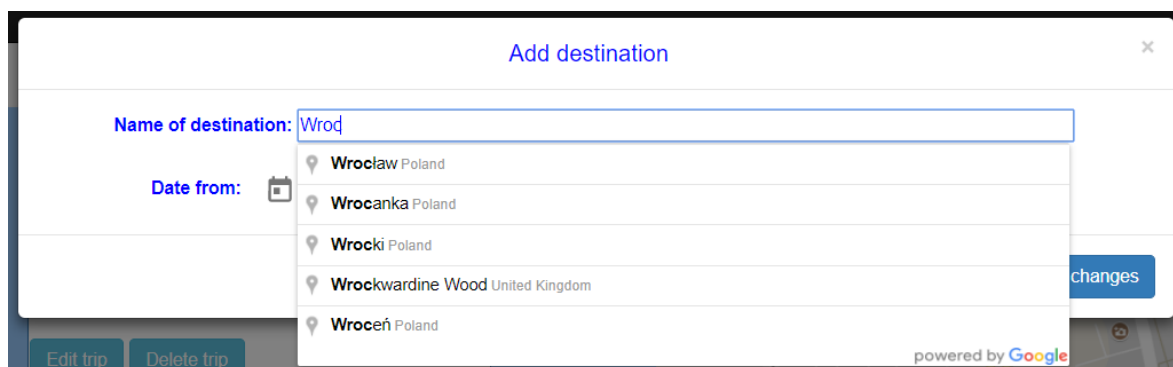
Panel przedstawiający szczegóły wyjazdu zawiera również mapę generowaną przez Google Maps API. Jeśli główny cel wyjazdu jest nazwą miejscowości mapa zostanie wycentrowana na daną miejscowość. Zbliżenie mapy można zmieniać, podobnie jak ustawiać inne ustawienie jej centrum.

Ekran szczegółów celu wyjazdu wygląda analogicznie podobnie do ekranu szczegółów wyjazdu.



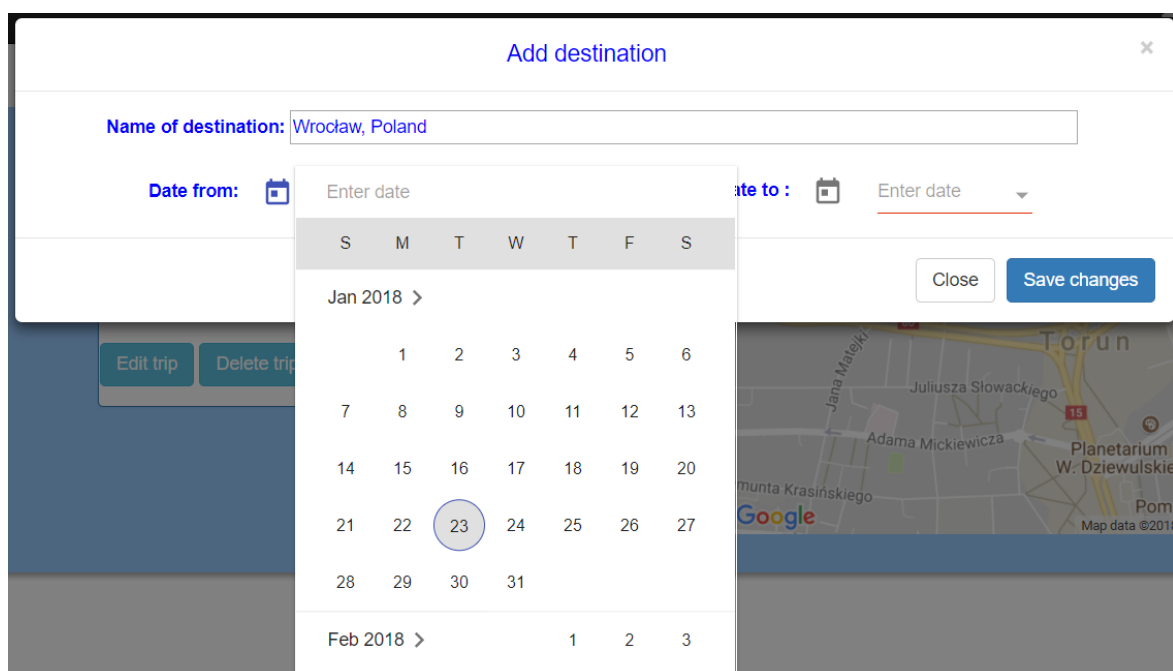
Rysunek IV-8 Interfejs graficzny: Okienko dialogu potwierdzenia usunięcia obiektu Źródło: Opracowanie własne

Przed usunięciem obiektu takiego jak wyjazd lub cel wyjazdu użytkownik otrzyma okienko żądające potwierdzenia, że użytkownik chce wykonać dane działanie (Rysunek IV-8). Zapobiega to błędom i irytacji użytkownika z powodu poświadczonych nieświadomie błędów ponieważ funkcjonalność usunięcia wyjazdu lub celu wyjazdu jest nieodwracalna i usuwa również wszystkie szczegóły powiązane z tymi obiektami.



Rysunek IV-9 Interfejs graficzny: Okno modalne dodawania celu wyjazdu – autouzupełnianie podpowiedzi Źródło: Opracowanie własne

Dodawanie i edycja wyjazdu lub celu wyjazdu odbywa się poprzez pojawiające się dodatkowe okno. W czasie wpisywania celu wyjazdu dzięki usłudze Google użytkownik uzyskuje podpowiedzi miast zgodnych z autouzupełnianiem wpisanych przez niego danych (Rysunek IV-9).

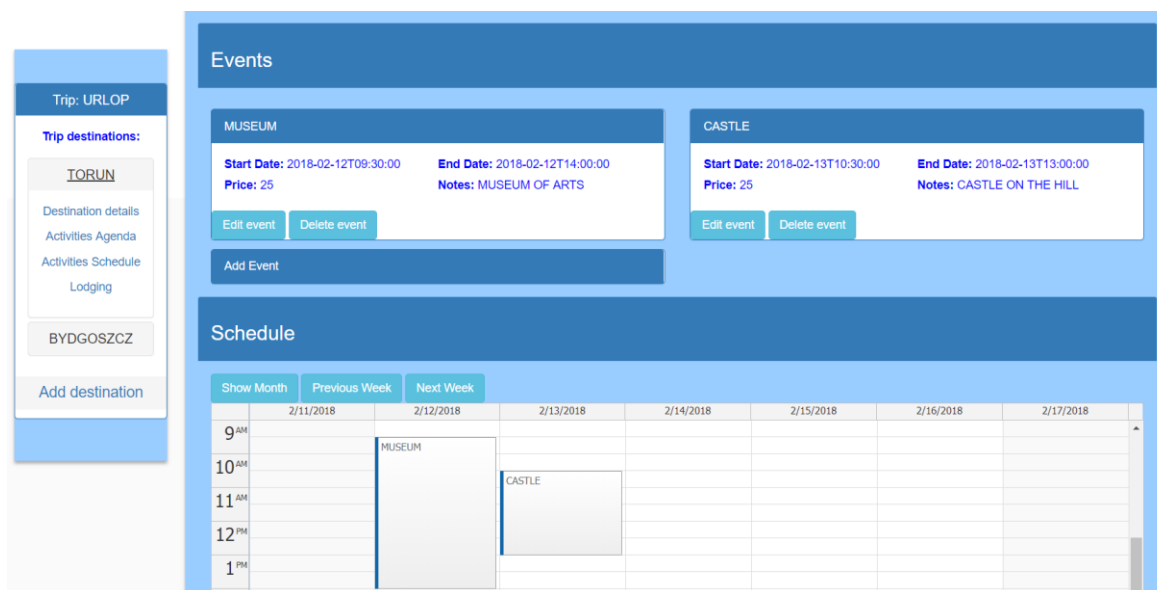


Rysunek IV-10 Interfejs graficzny: Okno modalne dodawania celu wyjazdu – wybór daty z kalendarza Źródło: Opracowanie własne

Wprowadzanie dat użytkownik może wykonać poprzez wybór odpowiedniej daty z pojawiających się kalendarzy tak jak na przykładzie dodawania nowego celu wyjazdu (Rysunek IV-10).

Ekran edycji celu wyjazdu jest tym samym ekranem, co ekran dodawania celu wyjazdu z tą różnicą, że dane w polach formularza będą uzupełnione aktualnymi wartościami atrybutów celu wyjazdu.

Ekran dodawania i edycji wyjazdu wygląda analogicznie podobnie do ekranu dodawania i edycji szczegółów celów wyjazdu.



Rysunek IV-11 Interfejs graficzny: Panel atrakcji i Panel terminarza Źródło: Opracowanie własne

Rysunek IV-11 przedstawia fragment ekranu szczegółów celu wyjazdu ukazując panel listy atrakcji oraz fragment terminarza.

Panel atrakcji pokazuje listę atrakcji danego celu wyjazdu wraz z jego szczegółami. Każdą atrakcję można edytować z osobna oraz usunąć. Usunięcie jej lub edycja powoduje również natychmiastowe zmiany w terminarzu. Użytkownik ma możliwość również dodawania nowych atrakcji. Formularz dodania lub edycji atrakcji ma analogicznie podobny układ do pozostałych formularzy dodawania lub edycji z tą jednak różnicą, że nie pojawia się on w dodatkowo wyskakującym oknie modalnym. Ukaże się on w miejscu panelu atrakcji w zamian za listę atrakcji. Funkcjonalności tego formularza pozostają jednak bez zmian w porównaniu do poprzednich.

Poniżej listy atrakcji na ekranie znajduje się panel z terminarzem. Terminarz zawiera graficzną reprezentację atrakcji z powyżej wypisanej listy. Są one umieszczone w kolumnie odpowiadającej odpowiedniemu dniu tygodnia oraz na przestrzeni odpowiednich godzin. Atrakcje te reagują na działania użytkownika. Po przez przesunięcie ich metodą przeciągnij i upuść można zmieniać ich daty pozostawiając łączny czas trwania niezmiennym. Można również zmieniać konkretnie jedynie godzinę startu lub końca, co powoduje również zmianę czasu trwania wyda-

rzenia poprzez rozciąganie i zmniejszanie reprezentującego je prostokąta w górę i w dół łapiąc za górną i dolną krawędź – w przypadku widoku tygodniowego - co zmienia godzinę startu lub końca lub poprzez zmianę szerokości prostokąta łapiąc za prawą lub lewą krawędź – w przypadku kalendarza w widoku miesięcznym – aby przedłużyć czas trwania wydarzenia o ilość pełnych dni.

Wszystkie szczegóły celu wyjazdu znajdują się na jednym ekranie. To oznacza, że użytkownik przesuwając zawartość ekranu z góry do dołu otrzyma po kolei panele szczegółów celu, atrakcji, terminarza i noclegu. Jeśli rozdzielczość ekranu monitora użytkownika na to pozwala może on oglądać kilka paneli jednocześnie. Menu boczne zawsze znajduje się w tym samym miejscu wobec okna przeglądarki. Zawiera ono rozsuwane menu celu wyjazdu. Dzięki narzędziu scrollspy biblioteki Bootstrap użytkownik klikając na jeden z odnośników do paneli zostanie przeniesiony na wysokość strony zawierającej ten panel. Ułatwia to użytkownikowi korzystanie z przeglądania szczegółów wyjazdu zapewniając, że musi wykonać tylko jeden krok zamiast przesuwania ekranu, aby zobaczyć interesujący go fragment.

5. Opracowanie i realizacja planu testowania

Testy utworzonej aplikacji odbyły się, jako testy jednostkowe przeprowadzane równoległe z implementacją aplikacji. Po zaimplementowaniu każdej kolejnej funkcjonalności była ona testowana, jak również zgodnie z zasadą testów regresyjnych były testowane również wcześniej przygotowane i przetestowane funkcjonalności aplikacji.

Po ukończeniu fazy implementacji całej aplikacji przygotowano schemat planu testowania polegający na opracowaniu listy kroków działań użytkownika tak, aby przetestować funkcjonalności aplikacji. Plan ten służy również do prezentacji możliwości aplikacji zostały więc pominięte w nim niektóre przypadki użycia takie jak przeglądanie szczegółów obiektu, ponieważ od ich pozytywnego przebiegu zależą kolejne kroki użytkownika. W planie testów nie uwzględniono również danych, jakie użytkownik ma wpisywać w trakcie wypełniania formularzy. Zostało to pozostawione do swobodnej decyzji użytkownika.

Plan testu użytkownika wyglądał następująco:

1. Rejestracja nowego użytkownika
2. Logowanie użytkownika
3. Dodanie nowego wyjazdu
4. Przeglądanie szczegółów wyjazdu
5. Edycja wyjazdu
6. Dodanie nowego celu wyjazdu
7. Przeglądanie szczegółów celu wyjazdu
8. Edycja celu wyjazdu
9. Dodanie atrakcji
10. Edycja atrakcji
11. Edycja daty atrakcji poprzez terminarz
12. Edycja noclegu
13. Usunięcie atrakcji
14. Usunięcie celu wyjazdu
15. Usunięcie wyjazdu

Realizacja planu przebiegła z wynikiem pozytywnym, wszystkie funkcjonalności systemu działają tak jak jest to oczekiwane. Użytkownik wykonując wskazane mu kroki przechodzi szczęśliwą ścieżkę. Oznacza to, że wszystkie usługi aplikacji działają tak, jak użytkownik chce, bez żadnych błędów. Dowodzi to więc, że aplikacja została zaprojektowana i zaimplementowana zgodnie z ustaloną na początku specyfikacją wymagań opartą na stworzonych wymaganiach biznesowych dla aplikacji wspomagającej planowanie wyjazdów turystycznych.

Podsumowanie

Zaplanowane cele pracy zostały zrealizowane. Celem pracy było stworzenie projektu i implementacji aplikacji wspomagającej planowanie wyjazdów turystycznych. Zgodnie z tym zaprojektowano aplikację, która umożliwi zalogowanemu użytkownikowi planowanie różnorodnych aspektów swojej podróży.

W ramach funkcjonalności aplikacji zalogowany użytkownik może dodawać nowe wyjazdy oraz edytować istniejące. W ramach jednego wyjazdu użytkownik może dodawać cele wyjazdu oraz planować szczegóły pobytu w danym miejscu. Przy wypełnianiu pola formularza dodawania destynacji, która może być miejscowością, użytkownikowi pokaże się podpowiedź pasujących do już wpisanego tekstu miejscowości. Jeśli cel wyjazdu jest miejscowością w szczegółach destynacji zamieszczona będzie mapa wycentrowana na dane miejsce. Wszystkie pola dodawania lub edycji dat opatrzone są kalendarzem, z którego można wybrać odpowiednią datę. Użytkownik ma możliwość dodawania planowanych aktywności w danej destynacji. Jeśli dana aktywność jest umieszczona w konkretnym przedziale czasowym zostanie ona również zaprezentowana na terminarzu. Terminarz przedstawiany jest, jako kalendarz w formacie tygodniowym lub miesięcznym. Terminarz ten jest interaktywny z użytkownikiem poprzez manipulację ustawieniem graficznych reprezentacji atrakcji w kalendarzu. Atrakcje można przesuwać na inne dni lub godziny a także rozciągać je w czasie. Użytkownik w ramach celu wyjazdu również może zanotować planowany nocleg w danej miejscowości.

Zgodnie ze specyfikacją wymagań i projektem aplikacji zaimplementowano wszystkie funkcjonalności, jako aplikację internetową. Frontend został stworzony przy pomocy frameworku AngularJS a backend został zakodowany używając frameworku Spring. Aplikacja została zaprojektowana zgodnie z architekturą wielowarstwową.

Po wykonaniu testów nad aplikacją potwierdzono, że wszystkie funkcjonalności zostały zaimplementowane i działają zgodnie z wymaganiami. Oznacza to, że cel pracy, którym był projekt i implementacja aplikacji do wspomagania planowania wyjazdów turystycznych został zrealizowany.

Bibliografia

1. Basarat Ali Syed, *Beginning Node.js*, Springer 2014
2. Cymańska-Garbowska Barbara, Steblik-Właźlak Barbara, *Podstawy turystyki Tom 1.*, WSiP, Warszawa 2014, s. 11-12
3. Hudson Orsine Assumpcao, *Getting Started with IntelliJ Idea*, Packt Publishing 2013
4. Kalbarczyk Dariusz, Kalbarczyk Arkadiusz, *Angular JS. Pierwsze kroki*, Helion 2015
5. Kuczek Zygmunt (red.), *Kompendium pilota wycieczek*, Proksenia, Kraków 2005 s.22-23
6. MacDonald Matthew, *HTML5. Nieoficjalny podręcznik*, Helion 2013
7. McLaughlin Michael, *Oracle Database 12c. Programowanie w języku PL/SQL*, Helion 2015
8. Rahman Syed Fazle, *Bootstrap. Tworzenie interfejsów stron WWW. Technologia na start*, Helion 2015
9. Organizacja Narodów Zjednoczonych, World Tourism Organization, *Terminologia turystyczna. Zalecenia WTO.*, Instytut Turystyki, Warszawa 1995, s.5–7
10. Rosca Stefan, Patin Den, *WebStorm Essentials*, Packt Publishing 2015
11. Yener Murat, Theedom Alex, *Professional Java® EE Design Patterns*, John Wiley & Sons, Inc. 2015
12. Yogesh Prajapati, *Java Hibernate Cookbook*, Packt Publishing 2015
13. Walls Craig, *Spring w akcji*, Helion 2015
14. <https://bower.io>
15. <https://www.daypilot.org>
16. <https://material.angularjs.org>
17. <https://ngmap.github.io>
18. <https://swagger.io>

Spis rysunków

Rysunek III-1 Diagram encji. Źródło: Opracowanie własne

Rysunek III-2 Diagram przypadków użycia. Źródło: Opracowanie własne

Rysunek III-3 Model ERD. Źródło: Opracowanie własne

Rysunek III-4 Diagram klas. Źródło: Opracowanie własne

Rysunek III-5 Model podziału aplikacji na warstwy. Źródło: Opracowanie własne

Rysunek IV-1 Interfejs graficzny: Ekran główny niezalogowanego użytkownika
Źródło: Opracowanie własne

Rysunek IV-2 Interfejs graficzny: Formularz logowania Źródło: Opracowanie
własne

Rysunek IV-3 Interfejs graficzny: Formularz rejestracji Źródło: Opracowanie
własne

Rysunek IV-4 Interfejs graficzny: Menu główne niezalogowanego użytkownika
Źródło: Opracowanie własne

Rysunek IV-5 Interfejs graficzny: Menu główne zalogowanego użytkownika
Źródło: Opracowanie własne

Rysunek IV-6 Interfejs graficzny: Ekran główny zalogowanego użytkownika
Źródło: Opracowanie własne

Rysunek IV-7 Interfejs graficzny: Szczegóły wyjazdu Źródło: Opracowanie własne

Rysunek IV-8 Interfejs graficzny: Okienko dialogu potwierdzenia usunięcia obiektu
Źródło: Opracowanie własne

Rysunek IV-9 Interfejs graficzny: Okno modalne dodawania celu wyjazdu –
autouzupełnianie odpowiedzi Źródło: Opracowanie własne

Rysunek IV-10 Interfejs graficzny: Okno modalne dodawania celu wyjazdu –
wybór daty z kalendarza Źródło: Opracowanie własne

Rysunek IV-11 Interfejs graficzny: Panel atrakcji i Panel terminarza Źródło:
Opracowanie własne