



Politechnika Wrocławska

Wydział Informatyki i Zarządzania

kierunek studiów: Informatyka

specjalność: -

Praca dyplomowa - inżynierska

**Komputerowy system planowania przewozów
dla taksówek**

Mateusz Adamczyk

słowa kluczowe:
systemy transportowe
zastosowanie informatyki
planowanie

krótkie streszczenie:

Praca dotyczy systemu, który usprawni i ułatwi pracę pracownika centrali firmy taksówkowej w zakresie planowania kursów.

W pracy uwzględniono przegląd istniejących rozwiązań oraz projekt nowego systemu. Opisano również najważniejsze elementy implementacji. Praca zawiera instrukcję instalacji i użytkowania systemu.

opiekun pracy	dr inż. Krzysztof Chudzik
dyplomowej	<i>Tytuł/stopień naukowy/imię i nazwisko</i>	<i>ocena</i>	<i>podpis</i>

Do celów archiwalnych pracę dyplomową zakwalifikowano do:*

a) kategorii A (akta wieczyste)

b) kategorii BE 50 (po 50 latach podlegające ekspertyzie)

* niepotrzebne skreślić

pieczęć wydziałowa

Wrocław 2015

Streszczenie

Tytuł pracy inżynierskiej brzmi „Komputerowy system do planowania przewozów dla taksówek“. Celem niniejszej pracy było stworzenie systemu do automatycznego planowania kursów w firmie taksówkowej, który ułatwiłby pracę pracownika centrali. Pierwszym etapem było określenie wymagań projektowych, a więc wytłumaczenie jak powinien działać system. Po zapoznaniu się z istniejącymi rozwiązaniami na rynku oraz technologiami i technikami przydatnymi do stworzenia takiego systemu, sporządzone zostały założenia projektowe. Określone zostały tutaj wymagania funkcjonalne i нефункционалне systemu, architektura systemu oraz sposób realizacji. Kolejnym etapem było dokładne zaprojektowanie systemu, specyfikacji przypadków użycia, wyglądu interfejsu, modeli bazy danych, wybór wzorca architektonicznego oraz opis struktury oprogramowania. Opisany został również proces implementacji tego właśnie systemu. Wymienione zostały narzędzia wykorzystane podczas tworzenia systemu, opisano najważniejsze (w domniemaniu autora pracy) fragmenty kodu źródłowego, sporządzono instrukcje instalacji oraz użytkowania systemu dla przyszłych użytkowników. System został również przetestowany, co zostało opisane w oddzielnym rozdziale. Na koniec wymieniono wnioski autora na temat systemu oraz przykładowe ścieżki rozwoju systemu w przyszłości.

Abstract

The title of engineering work is „A computer transportation planning system for taxis“. The goal of this work was to create system for automatic planning courses for the taxi firm that would facilitate the work of an employee of head office. The first step was to define the project requirements, which explains how the system should work. It was needed to take a look at existing technical solutions on the market and technologies and techniques useful to create such a system and then I was drawn up the design assumptions. It was defined here functional and nonfunctional requirements, system architecture and how this is to be achieved. Next step was to prepare accurate system design, specification of use cases, the appearance of the interfaces, database models. It was chosen the architectural pattern and described the structure of the software. It was also described the process of the implementation of this particular system. It was written what tools was used to create a system. It was described the most important (in my opinion) pieces of the source code. It was drawn up instructions for installing and using the system for future users. The system has been tested, and the results it was described in a separate chapter. At the end it was written conclusions about the system and it was described exemplary development path in the future.

Spis treści

Strona tytułowa z oceną pracy	i
Streszczenie pracy w językach polskim i angielskim	ii
Spis treści	iv
Spis rysunków	vi
Spis tabel	vii
1 Wstęp	1
1.1 Wprowadzenie do problematyki	1
1.2 Geneza pracy	2
1.3 Cel pracy	2
1.4 Zakres pracy	2
2 Wymagania projektowe	5
3 Stan wiedzy i techniki w zakresie tematyki pracy	7
3.1 Wprowadzenie	7
3.2 Przegląd podobnych istniejących rozwiązań	7
3.3 Przegląd przydatnych technik i technologii	9
3.4 Podsumowanie i wnioski	10
4 Założenia projektowe	11
4.1 Przedmiot prac	11
4.2 Wymagania funkcjonalne	11
4.3 Wymagania нефункционалне	13
4.4 Opis podstawowej architektury systemu	13
4.5 Sposób realizacji	13
5 Projekt	15
5.1 Wprowadzenie	15
5.2 Przypadki użycia	15
5.3 Interfejs	22
5.4 Baza danych	29
5.4.1 Model logiczny	29
5.4.2 Model fizyczny	31
5.5 Wzorzec architektoniczny	32
5.5.1 Opis wzorca	32
5.5.2 Zastosowanie w systemie	32
5.6 Struktura oprogramowania	33
5.7 Podsumowanie	35
6 Implementacja	37

6.1	Wprowadzenie	37
6.2	Wykorzystane środowiska i narzędzia programistyczne	37
6.3	Kod źródłowy	38
6.4	Interfejs	43
6.5	Instalacja oprogramowania	44
6.6	Instrukcja użytkowania	45
6.7	Archiwizacja danych	49
6.8	Podsumowanie	50
7	Testy	51
7.1	Wprowadzenie	51
7.2	Testy poprawności działania	51
7.3	Testy wydajnościowe	52
7.4	Testy bezpieczeństwa	52
7.5	Podsumowanie	52
8	Podsumowanie pracy	53
8.1	Wykonane prace	53
8.2	Realizacja celu pracy	53
8.3	Wnioski	54
8.4	Sugestie odnośnie dalszych prac	54
	Bibliografia	55
	Oświadczenia	57
	CD/DVD-ROM	61

Spis rysunków

5.1	Diagram przypadków użycia systemu	16
5.2	Przypadek użycia „Modyfikacja planowania kursów - Dodanie parametru“	18
5.3	Przypadek użycia „Modyfikacja planowania kursów - Usuwanie parametru“	18
5.4	Przypadek użycia „Modyfikacja planowania kursów - Edycja parametru“ .	18
5.5	Przypadek użycia „Planowanie kursów“	19
5.6	Przypadek użycia „Utworzenie zamówienia“	20
5.7	Przypadek użycia „Anulowanie zamówienia“	21
5.8	Przypadek użycia „Modyfikacja zamówienia“	21
5.9	Tworzenie zamówienia przez klienta	22
5.10	Zatwierdzenie zamówienia przez klienta	23
5.11	Anulowanie zamówienia przez klienta	23
5.12	Dane podane podczas anulowania zamówienia są niezgodne z bazą danych	23
5.13	Podanie błędnych danych podczas tworzenia zamówienia przez klienta . . .	24
5.14	Logowanie	24
5.15	Lista taksówkarzy, którzy są obecnie w pracy	25
5.16	Lista kursów aktualnych i przyszłych	25
5.17	Modyfikacja wybranego kursu	26
5.18	Usunięcie wybranego kursu	26
5.19	Stworzenie nowego kursu	27
5.20	Planowanie kursów	27
5.21	Lista kursów	28
5.22	Lista klientów	28
5.23	Mapa	29
5.24	Diagram klas, model logiczny bazy danych	29
5.25	Diagram encji, model fizyczny bazy danych	31
5.26	Widok	33
5.27	Kontroler	33
5.28	Model	34
6.1	Fragment kodu: aktualizacja kursu	38
6.2	Struktura plików	39
6.3	Fragment kodu: tworzenie tablicy kosztów	40
6.4	Fragment kodu: obliczenie kosztu dojazdu	40
6.5	Fragment kodu: wywołanie i obsługa po zakończeniu algorytmu	42
6.6	Obecny wygląd formularza logowania	43
6.7	Obecny wygląd mapy	43
6.8	Instrukcja: Złożenie zamówienia na taksówkę	45
6.9	Instrukcja: Potwierdzenie złożenie zamówienia na taksówkę	45
6.10	Instrukcja: Anulowanie zamówienia na taksówkę	46
6.11	Instrukcja: Lista taksówkarzy w pracy	46

6.12 Instrukcja: Lista aktualnych i przyszłych kursów	47
6.13 Instrukcja: Modyfikacja wybranego kursu	47
6.14 Instrukcja: Lista wszystkich kursów	48
6.15 Instrukcja: Lista kursów wybranego klienta	49

Spis tabel

4.1	Tabela wymagań funkcjonalnych	12
5.1	Połączenia relacyjne	30
5.2	Słownik bazy danych	31
5.3	Znaczenie kluczy obcych	32

Rozdział 1

Wstęp

1.1	Wprowadzenie do problematyki	1
1.2	Geneza pracy	2
1.3	Cel pracy	2
1.4	Zakres pracy	2

1.1 Wprowadzenie do problematyki

Niemal każda funkcjonująca na rynku organizacja czy firma dąży do unowocześnienia dotychczas stosowanych metod rozwoju przedsięwzięcia. Musi ona zmagać się ze zmieniającymi się warunkami środowiska biznesowego, dostosowywać się do nich przy równoczesnym przodowaniu nad innymi, konkurencyjnymi firmami. Odpowiedzią na potrzeby niemal każdej korporacji jest informatyka, która proponuje nowoczesne rozwiązania, umożliwiające polepszenie warunków pracy i rozwój danego przedsiębiorstwa. Transport jest obecnie stale rozwijającym się obszarem biznesu, przez co zapotrzebowanie na usługi informatyków i oprogramowań komputerowych wzrasta. Firmy taksówkarskie jak prawie każde przedsiębiorstwo mają na celu maksymalizację zysków i zminimalizowanie strat. Głównym problemem w przypadku firm taksówkarskich przy dążeniu do tego celu jest m.in. zaplanowanie przewozów, czyli przydzielenie kursów odpowiednim pracownikom ze względu na zarobki, szybkość dojazdu do klienta i realizacji trasy, obsługa klientów w przypadku zajętości wszystkich taksówek. Aby możliwe było rozwiązanie problemu i zoptymalizowanie pracy taksówkarzy niezbędne jest przechowywanie informacji dotyczących kursów dokonanych przez taksówkarzy oraz aktualnego położenia taksówek i klientów, dzięki którym możliwe byłoby zaplanowanie przewozów. Nowatorskie podejście do zagadnień związanych z funkcjonowaniem i prosperowaniem na rynku usług taksówkarskich sprawi, że korzystanie z tego typu rozwiązań będzie bardziej zrozumiałe, szybkie w obsłudze, a przede wszystkim przyniesie większe zyski firmom, które zajmują się tą dziedziną usług. Zaplanowanie i stworzenie tego typu oprogramowania jest jak najbardziej uzasadnione ze względu na zapotrzebowanie na rynku oraz aktualność tematu. Firmy taksówkarskie dążą do rozwoju i unowocześnienia funkcjonujących systemów, więc powinny być zainteresowane kupnem oprogramowania.

1.2 Geneza pracy

Nieodłącznym elementem rozwoju współczesnej technologii i powstawania nowych rozwiązań technologicznych jest informatyka i proponowane przez nią metody i sposoby, które w dużym stopniu przyczyniają się do nieustannego postępu technologicznego. Problemy podejmowane przez specjalistów, sposób ich rozwiązania i kompetencje sprawiają, iż rynek wykazuje coraz większe zainteresowanie nowatorskimi metodami, które oferują, dzięki czemu obserwowane jest ogromne zapotrzebowanie na systemy ułatwiające pracę w wielu dziedzinach życia (między innymi w transporcie). Odpowiedzią na popyt jest system planowania przewozów dla taksówek. System który ułatwi pracę człowieka pracującego w centrali firmy pomagając mu określić kursy dla każdej taksówki. Największym problemem w przyporządkowaniu zleceń będzie dynamiczne napływanie i rezygnacja klientów z zamówień oraz niemożność określenia końcowej lokalizacji kursu. Trzeba również uwzględnić godziny pracy taksówkarzy i ich zarobki. Wszystkie te problemy powodują, że człowiek bez żadnej pomocy nie jest w stanie w sposób optymalny wykonać swojej pracy. Więc ważne jest stworzenie systemu, który dostarczy pracownikom w centrali firmy taksówkowej pomoc, której potrzebują.

1.3 Cel pracy

Celem pracy jest wykonanie projektu i prototypu system planowania przewozów dla taksówek.

1.4 Zakres pracy

Rozdział 1 jest rozdziałem wprowadzającym czytelnika w problematykę pracy w podrozdziale 1.1, opisującym genezę pracy w podrozdziale 1.2, jej cel (podrozdział 1.3) oraz zakres (podrozdział 1.4).

Rozdział 2 posiada opis wymagań projektowych systemu, wyjaśniających sposób w jaki system powinien działać.

Rozdział 3 ma na celu zapoznanie się z podobnymi rozwiązaniami i porównanie ich ze względu na technologie i funkcje (podrozdział 3.2), wyszukanie technologii i technik przydatnych do stworzenia systemu (podrozdział 3.3).

Rozdział 4 służy do opisu głównych założeń projektowych systemu. Znajdują się tutaj wymagania funkcjonalne w podrozdziale 4.2 oraz нефункционалне systemu (podrozdział 4.3). Na tym etapie została opisana podstawowa architektura systemu (4.4) oraz sposób realizacji (4.5).

Następny rozdział (5) w sposób dokładny opisuje projekt systemu, który zawiera przypadki użycia wraz z ich opisami (podrozdział 5.2), prototypy interfejsów (5.3), schemat i opis modelu logicznego i fizycznego bazy danych (5.4). Również tutaj został wybrany wzorzec architektoniczny systemu (podrozdział 5.5) oraz opisana została struktura oprogramowania (5.6).

Rozdział 6 ma za zadanie przybliżyć sposób implementacji systemu, wskazać środowiska oraz narzędzia użyte w procesie implementacji (podrozdział 6.2), wyjaśnić niektóre algorytmy i zastosowane rozwiązania (podrozdział 6.3). Pokazano zmiany jakie nastąpiły w interfejsie względem projektów (6.4). Opisane zostały tutaj również instrukcje wyjaśniające sposób zainstalowania (6.5) i użytkowania systemu (6.6).

Rozdział 7 służy do przedstawienia wyników testów wykonanych na systemie: testy poprawności działania (7.2), testy wydajnościowe (7.3), testy bezpieczeństwa (7.4). Ostatni rozdział 8 podsumowuje całą pracę, wskazane zostały tutaj wnioski (8.3) oraz propozycje rozwoju systemu (8.4).

Rozdział 2

Wymagania projektowe

System będzie produktem, który ułatwi zaplanowanie, podział kursów w firmie taksówkowej oraz umożliwi składanie zamówień przez klientów i wprowadzenie zamówień przez pracowników.

Użytkownicy będą dzielić się na cztery typy:

- administrator,
- pracownik centrali,
- klient.
- taksówkarz.

System ma być obsługiwany głównie przez pracowników centrali danej firmy taksówkowej. Klient to osoba niezalogowana, której celem jest zamówienie taksówki. Jego interakcja będzie ograniczać się do:

- wprowadzania nowego zamówienia
- możliwości anulowania wybranego zamówienia.

Taksówkarz będzie miał możliwość:

- zmiany swojego stanu, który określa czy jest on w czasie realizacji kursu, na przerwie, czy przyjął już zlecenie, jest obecnie wolny bądź nie ma go w pracy

Pracownik centrali natomiast będzie posiadał wszystkie funkcje klienta, dodatkowo będzie miał możliwość:

- modyfikacji wybranego zamówienia,
- wyświetlenia informacji o kursach, taksówkarzach i klientach, szczególnie:
 - wykazu obecnie realizowanych i następnych w kolejności kursach
 - listy oczekujących klientów.
- podglądać obecne pozycje taksówek, klientów i realizowane, planowane kursy na mapie, co pozwoli na wizualny podgląd obecnej sytuacji.

Administrator będzie posiadał funkcje pracownika, ponad to będzie mógł:

- dodawać i modyfikować dane pracowników (w tym taksówkarzy), również możliwe będzie dodanie nowego pracownika.

Zaplanowanie kursów ma się odbywać automatycznie przez system po każdorazowym dodaniu lub anulowaniu zamówienia, włączeniu się do pracy lub ukończeniu pracy przez taksówkarza. System powinien uwzględnić przy planowaniu aspekty dotyczące:

- odległości taksówki od klienta,
- kierunku kursu,
- zarobków pracowników,
- czasu pracy,
- statusu w firmie.

Mimo, że kursy będą automatycznie planowane, to możliwe będzie przydzielenie wybranej taksówki do zlecenia. Wszelkie informacje dotyczące pracowników, taksówek czy kursów będą trzymane w bazie danych.

System nie będzie obejmował przekazania zleceń do taksówkarzy, ponieważ na rynku istnieje wiele produktów, które skupiają się głównie na aspekcie komunikacji taksówkarza z centralą. Również wiele aplikacji mobilnych oferuje rozwiązanie problemu składania zamówień przez klientów, a więc ten aspekt pojawi się w pracy w wersji okrojonej.

System będzie aplikacją webową, a więc będzie pracował na serwerze WWW i komunikował się z wykorzystaniem przeglądarki internetowej. Pozwoli to na uniwersalny dostęp, bez względu na system operacyjny i miejsce użytkowania. Dane będą trzymane na serwerze, co zabezpieczy je przed utratą w razie awarii urządzenia. Dostęp do systemu musi być zabezpieczony tak, aby jedynie osoby upoważnione miały do niego dostęp, takie jak pracownik centrali czy kierownik firmy taksówkowej.

Wygląd systemu powinien być prosty, przejrzysty. Klientowi zostanie udostępniona jedna strona na której znajdzie się formularz do zamówienia taksówki oraz do anulowania zamówienia. Pracownik będzie korzystał z panelu administracyjnego w którym główne opcje znajdują się w menu po lewej stronie, natomiast opcje podrzędne na górze strony. W środku znajdzie się komponent, który będzie służył do obsługi wybranych opcji. Panel administratora będzie identyczny jak w przypadku pracownika, z dodanymi kilkoma opcjami.

System ten w znacznym stopniu zmniejszy wydatki firmy na paliwo, ponieważ kursy zostaną zaplanowane w sposób optymalny lub bliski optymalnemu pod względem odległości tras, jakie będą musieli pokonać taksówkarze. Również zarobki w firmie zostaną rozdzielone niemal równomiernie, między taksówkarzy o takim samym statusie, czego przyczyną będzie uwzględnienie aspektu zarobków i statusu pracowników podczas planowania kursów.

Rozdział 3

Stan wiedzy i techniki w zakresie tematyki pracy

3.1	Wprowadzenie	7
3.2	Przegląd podobnych istniejących rozwiązań	7
3.3	Przegląd przydatnych technik i technologii	9
3.4	Podsumowanie i wnioski	10

3.1 Wprowadzenie

Rozdział ten ma na celu zapoznanie się z podobnymi rozwiązaniami i porównanie ich ze względu na technologie i funkcje. Jest to bardzo ważna część pracy, ponieważ możemy dowiedzieć się w jaki sposób i jakimi technologiami próbowano poradzić sobie z takim samym lub podobnym problemem oraz możemy sprawdzić czy tworzenie systemu jest opłacalne, a może rozwiązań na obecnym rynku jest na tyle dużo, że nie ma przesłanek do uruchomienia projektu.

Pierwszy podrozdział opisuje systemy powiązane mniej lub bardziej z planowaniem przewozów dla taksówek i optymalizacją procesów biznesowych w korporacjach taksówkowych. Znajdziemy tutaj opisy kolejnych rozwiązań, których podejście do problemu jest czasem bardzo odmienne. Drugi podrozdział dotyczy technologii, jaka zostanie wykorzystana przeze mnie do stworzenia systemu. Rozszerzone informacje na temat systemów i technologii znajdują się w odsyłaczach w bibliografii.

3.2 Przegląd podobnych istniejących rozwiązań

Na rynku istnieje wiele systemów informatycznych, które usprawniają pracę w różnych firmach. Wiele firm logistycznych korzysta z rozwiązań informatycznych w celu optymalizacji tras lub wspomagania planowania i zarządzania transportem, zamówieniami, pracownikami. Również korporacje taksówkowe wykorzystują takie systemy w swojej pracy. Firma *Autocab*, działająca na arenie międzynarodowej, tworzyła przez ponad dwie dekady system *Ghost*

(„<http://www.autocabinternational.com/taxi-booking-system-software/ghost/ghost/>”), który umożliwia:

- rejestrację audio zamówień,
- podgląd kierowców na mapie,
- prowadzenie statystyk dotyczących połączeń,
- dostosowanie formularza zamówień,
- rejestrowanie zamówień przez pracowników centrali w systemie.

Lokalizacja taksówek na mapie uwzględnia obecny status kierowcy oraz rejestrowanie przebytych tras. Firma również tworzy urządzenia i aplikacje mobilne dla kierowców taksówek, na które przekazywane są zlecenia. Na stronie dotyczącej produktów nie znajduje się żadna informacja dotycząca wspomagania planowania kursów przez system, co prawdopodobnie oznacza, że nie posiada takowej funkcji i są nią obciążeni pracownicy centrali. Produkt *Ghost* oparty jest na technologii .net, c# i SQL.

Bardziej rozbudowanym systemem jest *esysTaxi* stworzony przez *ESYSCODER Dariusz Bączkowski* („<http://system-taxi.pl/>”). Składa się z:

- serwera map i *esysTaxi*,
- centrali telefonii VoIP,
- aplikacji dyspozytora,
- terminala.

Posiada bardzo wiele funkcji dotyczących między innymi:

- automatyczne planowanie kursów,
- zmiany algorytmu przydzielania kursów,
- podglądu taksówkarzy i klientów na mapie,
- składanie zamówień przez klienta na stronie internetowej.

System umożliwia zmiany algorytmu przydzielania kursów do taksówkarzy przez upoważnionego pracownika. Algorytmy zwracają głównie uwagę na podział na strefy i wybierają taxi ze strefy klienta lub wybór najbliższej taksówki z możliwością określenia najdłuższego możliwego dojazdu, połączone obydwie algorytmy oraz wybór taksówki najdłużej oczekującej na zlecenie, a znajdujące się w zakresie określonej odległości. Aplikacja dyspozytora umożliwia ciągły podgląd pozycji taksówek na mapie z zaznaczeniem stanu w jakim się znajdują, pozycji klientów. Mapa posiada możliwość zdefiniowania widoków, aby pracownik mógł szybko zmieniać podgląd na określone strefy. Obsłużone zostało również odbieranie zamówień telefonicznych, przez stronę WWW dostosowaną również do smartfonów od klienta przez aplikację. Do każdego telefonu zostają przypisane i zapamiętane składane zamówienia, aby pracownik mógł później szybko wybrać miejsce początku kursu z listy, co znacznie przyspiesza złożenie zamówienia oraz jest wykorzystywane połączeń obsługiwanych przez zdalnego dyspozytora IVR, który proponuje klientowi miejsca startowe lub połączenie z pracownikiem. Klient ma możliwość sprawdzenia ceny informacyjnej lub maksymalnej, w zależności od preferencji korporacji, jeśli poda adres docelowy kursu. Pracownik centrali ma dodatkowo możliwość sprawdzenia stanów taksówek, pozycji klientów w kolejce, informacji o dowolnym zleceniu, komunikacji z taksówkarzami, sprawdzenia

historii pracy taxi. System posiada jeszcze więcej funkcji, a więc jest on bardzo rozbudowany oraz wydaje się być mocno przemyślane pod względem biznesowym. Niestety firma nie udostępniła informacji o technologiach, które zostały wykorzystane do stworzenia produktu.

Istnieją również rozwiązania skupiające się głównie na aspekcie składania zamówienia przez klienta i przyjmowania zleceń przez kierowców taksówek. Zrzeszają one wiele korporacji taksówkowych, do których są przekazywane zlecenia. Najczęściej są to aplikacje mobilne takiej jak *Taxi5* („<http://www.taxi5.pl/>”) lub *heyTaxi* („<http://www.heytaxi.pl/>”), w których aby zamówić taksówkę wystarczy nacisnąć jeden przycisk na ekranie, a system wybierze najbliższą taksówkę i przekaże zamówienie do korporacji w której ta taksówka pracuje. Korporacja sama decyduje co zrobić z danym zleceniem. Aplikacje te obsługują również możliwość komunikacji między kierowcą, a klientem oraz automatycznie powiadamiają klienta o czasie do przyjazdu taksówki lub o tym, że taxi już czeka. Ważną funkcją jest też możliwość podglądu lokalizacji zarówno klientów jak i taksówek na mapie. Rozwiązania te są wykonane najczęściej pod systemy *Android*, *Windows Phone* i *iOS*.

3.3 Przegląd przydatnych technik i technologii

Obecnie możliwe jest połączenie niesamowicie wielu technologii do stworzenia systemu informatycznego. Wybór najlepszych jest trudny, ponieważ każda z nich posiada swoje zalety i wady.

System będzie aplikacją webową, a więc będzie pracował na serwerze WWW i komunikował się z wykorzystaniem przeglądarki internetowej. Pozwoli to na uniwersalny dostęp, bez względu na system operacyjny i miejsce użytkowania. Dane będą trzymane na serwerze, co zabezpieczy je przed utratą w razie awarii urządzenia. Baza danych zostanie stworzona w środowisku *MySQL*, ponieważ jest to lekkie i przyjazne środowisko. Serwer będzie stworzony w technologii *node.js*. Jest to środowisko programistyczne open source, którego głównym aspektem jest możliwość tworzenia aplikacji tworzonych zdarzeniami wykorzystującymi asynchroniczny system wejścia-wyjścia. Bardzo ważna właściwość dla mojego produktu, ponieważ operacje optymalizacji kursów, ciągłe obliczenia tras oraz inne funkcje mogą być czasochłonne i nie powinny przeszkadzać w użytkowaniu systemu pracownikowi lub klientom. Środowisko to jest zaprojektowane do tworzenia wysoce skalowalnych aplikacji internetowych, co może okazać się przydatne w przypadku większej ilości taksówek w firmie, bądź wielu zamówień klientów. Do wyznaczenia kursów wykorzystany zostanie algorytm genetyczny. Jest to metaheurystyka opierająca się na ewolucji biologicznej. Polega na generowaniu kolejnych populacji osobników, w taki sposób aby geny tych osobników stawały się coraz lepsze, czyli coraz bardziej dopasowane do środowiska. Jedną z zalet algorytmu genetycznego jest to, wynik staje się coraz lepszy z czasem, co oznacza, że im dłużej będzie działał tym wartości najlepsze będą bliżej wartości optymalnej. Zostanie on zaimplementowany w javascript. Oczywiście do stworzenia systemu zostanie wykorzystany *html5*, *css*, *javascript* wraz z frameworkiem *jQuery*. Są to podstawowe narzędzia wykorzystywane do tworzenia stron internetowych. Jeśli chodzi o techniki wyznaczania najlepszych rozwiązań, istnieje ich wiele, jednak ja zainteresowałem się obliczeniami ewolucyjnymi. „Obliczenia ewolucyjne to dział sztucznej inteligencji – dziedziny stosunkowo młodej, budzącej wiele kontrowersji. Spotykane w literaturze przedmiotu różne jej definicje sprowadzają się do określenia, że jest to próba modelowania aspektów ludzkiego rozumowania za pomocą komputerów, albo próba rozwiązywania za pomocą komputerów takich problemów, które są zwykle rozwiązywane przez (inteligentnego) człowieka”. Jako,

że na zajęciach poznałem algorytm genetyczny, postanowiłem wykorzystać go w moim problemie planowania przewozów dla taksówek. „Algorytm genetyczny (GA, ang. Genetic Algorithm) to ewolucja sztucznych „osobników“, z których każdy jest zakodowanym potencjalnym rozwiązaniem rozpatrywanego problemu. Osobniki te ewoluują w sztucznym środowisku, ponieważ należą do tego samego gatunku (koduują rozwiązanie tego samego zadania) to konkurują o zasoby w tym środowisku. Podobnie jak w biologii, o szansach przeżycia i wydania potomstwa przez sztucznych osobników decyduje „dobór naturalny“ – im lepiej jest przystosowany osobnik do danego środowiska, tym większe ma szanse przeżyć i wydać potomstwo. Potomstwo – podobnie jak w naturze – różni się od swoich rodziców dzięki działaniu specjalnie zaprojektowanych, wzorowanych na naturze, operatorów genetycznych (mutacji i krzyżowania). Środowisko odzwierciedla rozwiązywane przez algorytm genetyczny zadanie, zatem ocena przystosowania osobnika jest oceną jakości kodowanego przez niego rozwiązania. W ten sposób, w kolejnych pokoleniach populacji sztucznej osobników zaczynają dominować coraz to lepsze rozwiązania. Najlepszy osobnik, po rozkodowaniu, jest szukanym rozwiązaniem. „Algorytm ten działa tym lepiej im więcej „osobników“ posiada „populacja“.

3.4 Podsumowanie i wnioski

Ogólnie rzecz biorąc, informacje na temat systemów wykorzystywanych do zarządzania zamówieniami w firmach taksówkowych są ciężkie do znalezienia, bądź niechętnie udostępniane przez twórców. Jeśli chodzi o obsługę klienta, istnieje wiele systemów/aplikacji, które wspomagają zamówienie taksówki i mają na celu jak najszybszą obsługę klienta. Natomiast systemy, które znalazłem, uwzględniające pracę centrali korporacji albo nie posiadają funkcji optymalizowania/wspomagania planowania kursów, albo jest ono zrealizowane w sposób, który uważam za możliwy do ulepszenia. Sądzę, że komputerowy system planowania przewozów dla taksówek, który uwzględnia więcej czynników niż odległość od klienta może okazać się przydatny w firmach, aby nie tylko zmniejszać zużycie paliwa, ale również rozdzielać pracę w sposób jak najbliższy sprawiedliwemu pomiędzy taksówkarzy.

Rozdział 4

Założenia projektowe

4.1	Przedmiot prac	11
4.2	Wymagania funkcjonalne	11
4.3	Wymagania niefunkcjonalne	13
4.4	Opis podstawowej architektury systemu	13
4.5	Sposób realizacji	13

4.1 Przedmiot prac

Przedmiotem projektu jest stworzenie systemu, który usprawni planowanie przewozów dla taksówek. Jako planowanie określam podział kursów pomiędzy taksówkarzy w taki sposób aby zużycie paliwa było bliskie minimum oraz zarobki taksówkarzy były bliskie równemu zarobkom innych taksówkarzy o takim samym statusie w firmie. Czynność ta będzie realizowana automatycznie przez system, ale pracownik będzie miał możliwość ingerowania w podział kursów, ponieważ mogą pojawić się nieprzewidziane wcześniej, wyjątkowe ograniczenia, nie uwzględnione podczas planowania kursów. Pracownik będzie miał ciągły dostęp do potrzebnych danych, które usprawnią jego działania. Zarządzanie kontami pracowników centrali i taksówkarzy przypadnie Administratorowi. Prototyp systemu w sposób okrojony obejmie funkcje potrzebne klientowi i taksówkarzowi do realizacji zamówień.

4.2 Wymagania funkcjonalne

System udostępniony zostanie dla czterech typów użytkowników:

- administrator,
- pracownik centrali,
- klient.
- taksówkarz.

Wymaganie	Użytkownicy	Opis
Logowanie	Pracownik centrali, Administrator, Taksówkarz	Każdy pracownik i taksówkarz musi zalogować się do systemu przed rozpoczęciem pracy.
Zarządzanie kontami pracowników i taksówkarzy	Administrator	Możliwość dodawania i modyfikacji kont pracowników centrali i taksówkarzy do systemu.
Podgląd aktualnych i następnych w kolejności kursów	Pracownik centrali, Administrator	Podgląd listy aktualnych i następnych w kolejności kursów, aktualizowany każdorazową zmianą dotyczącą kursów.
Planowanie kursów		Dopasowanie klientów do taksówek. Najważniejszy proces w systemie. Uruchamiany automatycznie przez system zawsze po dodaniu nowego zamówienia, anulowaniu zamówienia i zmianie stanu taksówki.
Modyfikacja planowania kursów	Administrator	Określenie parametrów uwzględnionych podczas planowania oraz ich wag.
Wyświetlenie informacji o kursach	Pracownik centrali, Administrator	Podgląd listy kursów zrealizowanych, aktualnie realizowanych i przyszłych.
Wyświetlenie informacji o taksówkarzach	Pracownik centrali, Administrator	Podgląd listy taksówkarzy z wszystkimi danymi oraz taksówek obecnie przez nich używanych.
Wyświetlenie informacji o klientach	Pracownik centrali, Administrator	Podgląd listy kursów przypisanych do klientów
Wyświetlenie aktualnego stanu kursów na mapie	Pracownik centrali, Administrator	Ułatwienie nadzoru taksówkarzy poprzez wizualizację stanu taksówek, położenia taksówek i klientów oraz kursów aktualnych i zaplanowanych na mapie.
Wyświetlenie informacji o klientach	Pracownik centrali, Administrator	Podgląd listy kursów przypisanych do klientów
Utworzenie zamówienia	Pracownik centrali, Klient	Zamówienie taksówki do klienta poprzez formularz internetowy.
Anulowanie zamówienia	Pracownik centrali, Klient	Zaniechanie zamówienia poprzez formularz internetowy.
Modyfikacja zamówienia	Pracownik centrali	Manualne przypisanie taksówki do klienta.
Zmiana stanu pracy	Taksówkarz	Określenie stanu w jakim się znajduje taksówkarz.

Tablica 4.1: Tabela wymagań funkcjonalnych

Natomiast najważniejszym użytkownikiem jest pracownik centrali, a więc to on będzie posiadał najwięcej funkcji. Wszystkie procesy zostaną wymienione w tabeli 4.1 wraz z użytkownikami, którzy będą mieli możliwość realizacji funkcji i krótkim opisem celu ich działania. W przypadku braku użytkownika, oznacza to, że proces wykonywany będzie przez system. Dokładny opis przypadków użycia znajduje się w następnym rozdziale 5 Projekt.

4.3 Wymagania niefunkcjonalne

- System musi być dostępny dla każdego użytkownika używającego urządzenia z dostępem do internetu.
- System musi obsługiwać co najmniej 50 użytkowników w jednym czasie.
- Planowanie kursów nie powinno trwać dłużej niż 5 sekund, ponieważ taksówkarz musi otrzymać informacje o zleceniu jak najszybciej, aby nie przedłużać okresu oczekiwania klienta.
- System musi być dostępny w polskiej wersji językowej.
- Interfejs musi być przejrzysty, intuicyjny.

4.4 Opis podstawowej architektury systemu

Architektura systemu to architektura klient-serwer. Cechą tej architektury jest równoległe działanie dwóch procesów. Jeden z nich (serwer) realizuje usługi na rzecz drugiego (klient). Właściwość ta pozwala na równoczesne uruchomienie wielu środowisk z możliwością spójnego operowania danymi, niezależnie od środowiska, w którym te dane będą prezentowane, np. przeglądarka internetowa. Z usług serwera może korzystać wielu klientów równocześnie. Połączenie pomiędzy klientem a serwerem możliwe jest przez zastosowanie interfejsów i protokołów komunikacyjnych modelu *TCP/IP*.

4.5 Sposób realizacji

System będzie pracował na serwerze WWW i komunikował się z wykorzystaniem przeglądarki internetowej. Pozwoli to na uniwersalny dostęp, bez względu na system operacyjny i miejsce użytkowania. Dane będą trzymane na serwerze, co zabezpieczy je przed utratą w razie awarii urządzenia. Baza danych zostanie stworzona w środowisku *MySQL*, ponieważ jest to lekkie i przyjazne środowisko. Serwer będzie stworzony w technologii *Node.js*. Jest to środowisko programistyczne open source, którego głównym aspektem jest możliwość tworzenia aplikacji tworzonej zdarzeniami wykorzystującymi asynchroniczny system wejścia-wyjścia. Bardzo ważna właściwość dla mojego produktu, ponieważ operacje optymalizacji kursów, ciągłe obliczenia tras oraz inne funkcje mogą być czasochłonne i nie powinny przeszkadzać w użytkowaniu systemu pracownikowi lub klientom. Środowisko to jest zaprojektowane do tworzenia wysoce skalowalnych aplikacji internetowych, co może okazać się przydatne w przypadku większej ilości taksówek w firmie, bądź wielu zamówień klientów. Środowisko *Node.js* zostanie wsparte przez framework *Sails.js*, który wymusza użycie architektury *MVC*. Dzięki temu struktura plików i

kod systemu będą bardziej przejrzyste. Ponad to usprawnia on połączenie warstwy Modelu z bazą danych. Asynchroniczność zostanie wsparta również po stronie klienta przez framework *AngularJS*, który służy do szybkiego i łatwego tworzenia aplikacji internetowych zwanych *SPA* (*single page app*), czyli takich, w których zmiana widoku nie wymaga przeładowania strony oraz logika aplikacji zostaje przeniesiona z serwera na klienta. Do wyznaczenia kursów wykorzystany zostanie algorytm genetyczny. Jest to metaheurestyka opierająca się na ewolucji biologicznej. Polega na generowaniu kolejnych populacji osobników, w taki sposób aby geny tych osobników stawały się coraz lepsze, czyli coraz bardziej dopasowane do środowiska. Jedną z zalet algorytmu genetycznego jest to że, wynik staje się coraz lepszy z czasem, co oznacza, że im dłużej będzie działał tym wartości najlepsze będą bliżej wartości optymalnej. Zostanie on zaimplementowany w javascript. Oczywiście do stworzenia systemu zostanie wykorzystany html5, css, javascript wraz z frameworkiem *jQuery*. Są to podstawowe narzędzia wykorzystywane do tworzenia stron internetowych.

Rozdział 5

Projekt

5.1	Wprowadzenie	15
5.2	Przypadki użycia	15
5.3	Interfejs	22
5.4	Baza danych	29
5.4.1	Model logiczny	29
5.4.2	Model fizyczny	31
5.5	Wzorzec architektoniczny	32
5.5.1	Opis wzorca	32
5.5.2	Zastosowanie w systemie	32
5.6	Struktura oprogramowania	33
5.7	Podsumowanie	35

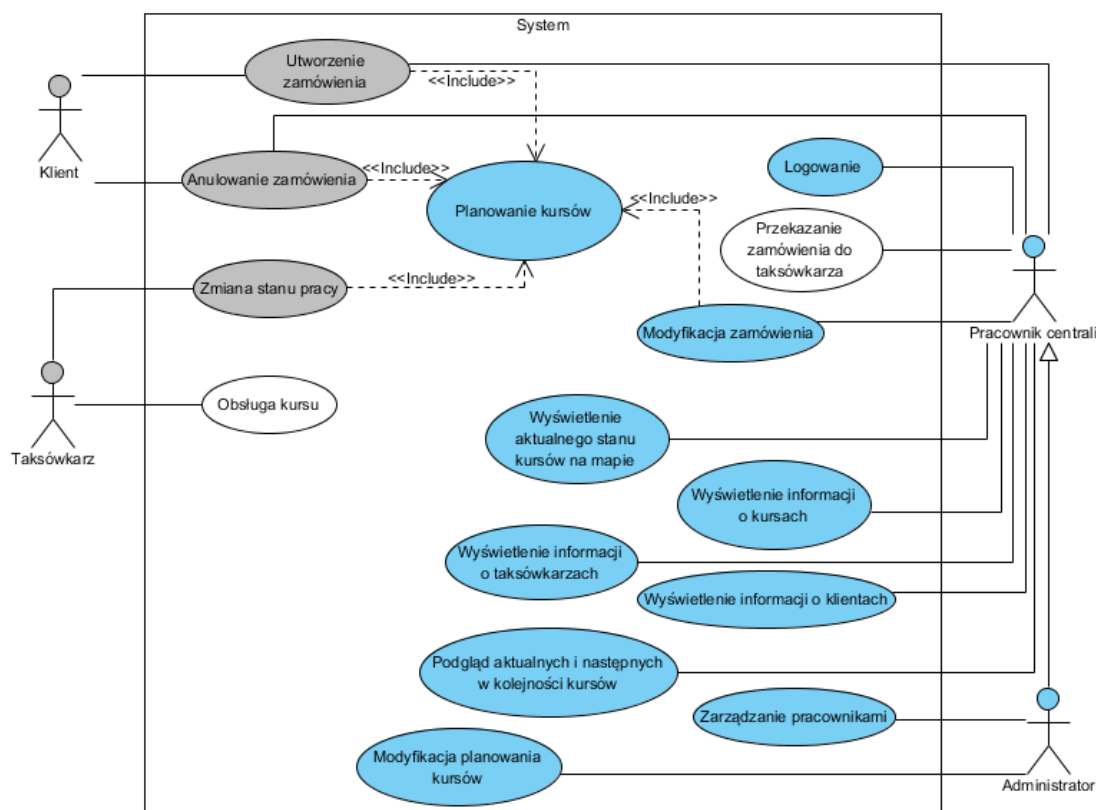
5.1 Wprowadzenie

W poprzednim rozdziale przyjęto założenia wymagań funkcjonalnych, niefunkcjonalnych, ogólną architekturę systemu oraz sposób realizacji oprogramowania systemu. Są one wystarczające do rozpoczęcia prac projektowych. Podrozdział „Przypadki użycia“ zawiera dokładny opis funkcji systemu. Wykorzystałem tam diagramy oraz opisy słowne, które w połączeniu powinny nakreślić właściwy obraz sytemu i procesów w nim zawartych. Następny podrozdział skupiony jest na wyglądzie systemu, który powinien w jak największym stopniu wpływać na wygodę korzystania z niego oraz być dostosowany do pracy w firmie taksówkowej. Przed ostatni podrozdział dotyczy bazy danych. Ukazałem tam schemat bazy, którą należy stworzyć, aby system zawierał wszystkie dane potrzebne do prawidłowego działania.

5.2 Przypadki użycia

Do opisu przypadków użycia posłużę się diagramem przypadków użycia oraz diagramów aktywności dla ważniejszych przypadków. Każdy przypadek zostanie również opisany, aby mógł on być właściwie zinterpretowany.

Na rysunku 5.1 przedstawiłem przypadki użycia które zostaną zaimplementowane w prototypie systemu (kolor niebieski), funkcje, których funkcjonalność zostanie w pewny sposób ograniczona (kolor szary) oraz funkcje pominięte (kolor biały), podobny podział dotyczy użytkowników. Wszystkie funkcje wymienione na diagramie, w zależności od użyt-



Rysunek 5.1: Diagram przypadków użycia systemu

kownika, powinny być dostępne do uruchomienia w każdej chwili korzystania z systemu z poziomu menu. Jeśli użytkownik zalogowany będzie jako pracownik centrali lub administrator powinien przez cały czas korzystania z systemu widzieć listę aktualnych i następnych w kolejności kursów w panelu na dole strony, który aktualizowany będzie z każdorazową zmianą dotyczącą kursów aktualnych i następnych w kolejności, czyli np. kiedy klient utworzy nowe zamówienie. Każdy pracownik, w tym administrator oraz taksówkarz musi zalogować się do systemu przed rozpoczęciem pracy, używając loginu bądź identyfikatora i hasła zapisanego w bazie danych. Po zalogowaniu pracownika centrali pojawi się informacja o aktualnych i przyszłych kursach oraz spis taksówkarzy będących w pracy.

Administrator będzie posiadał funkcje pracownika centrali, będzie w stanie zmieniać parametry planowania kursów oraz otrzyma możliwość zarządzania kontami pracowników i taksówkarzy. Oznacza to, że będzie mógł dodawać nowe konta oraz je modyfikować. Będzie się to odbywało poprzez przejście do odpowiedniej funkcji, gdzie zostaną wyświetlone listy pracowników i taksówkarzy. Nad listą powinien widnieć przycisk służący do utworzenia nowego konta, naciśnięcie go spowoduje wyświetlenie formularza z polami do wypełnienia, odpowiednimi do typu konta. Identyfikator taksówkarza będzie automatycznie wypełniany przez system. Poprawne wypełnienie formularza, spowoduje dodanie konta do bazy danych. Modyfikować konto będzie możliwe poprzez naciśnięcie przycisku

znajdującego się z boku każdego z kont na liście, co zadziała podobnie jak w przypadku tworzenia konta, z tą różnicą, że pola zostaną wypełnione danymi konta. Edycja kilku pól, takich jak login, identyfikator nie będzie możliwa.

Modyfikacja planowania kursów przez Administratora składa się z trzech procesów:

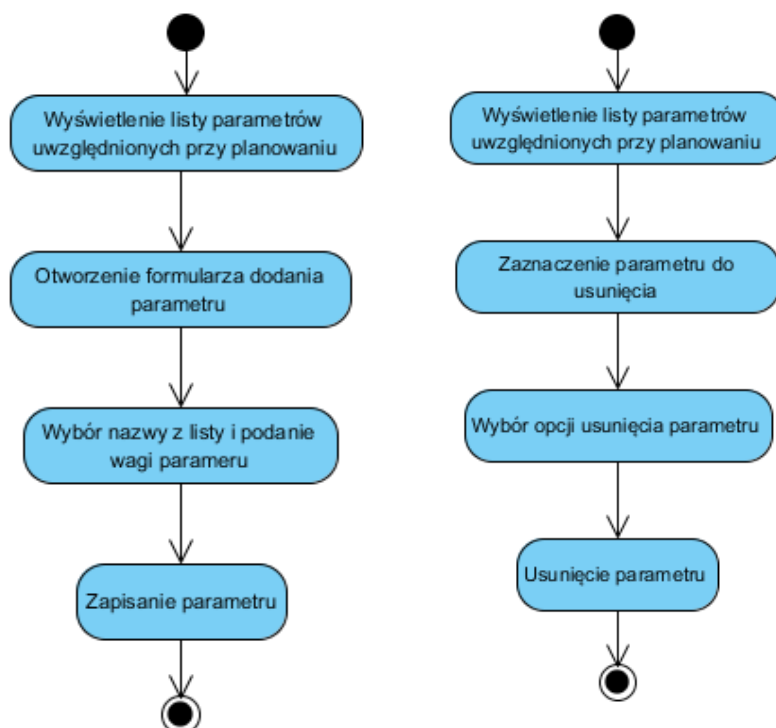
- dodania parametru (Rysunek 5.2)
- usunięcia parametru (Rysunek 5.3)
- edycji parametru (Rysunek 5.4)

Każdy z przypadków zaczyna się od wyświetlenia listy parametrów, które są obecnie uwzględniane podczas planowania. W przypadku dodania parametru, należy wybrać opcję do tego przeznaczoną, co spowoduje wyświetlenie formularza, w którym użytkownik będzie musiał wybrać parametr z listy dostępnych atrybutów, np. odległość od klienta, a następnie podać wagę parametru z zakresu od 1 do 10, gdzie im wyższa waga tym ważniejszy jest parametr podczas planowania kursów. Możliwe jest również usunięcie parametru, co oznacza, że parametr nie będzie brany pod uwagę lub edycja, polegająca na zmianie wagi parametru.

Planowanie kursów (Rysunek 5.5): Jest to najważniejszy i kluczowy proces w systemie. Uruchamiany automatycznie przez system zawsze po dodaniu nowego zamówienia, anulowaniu, modyfikacji zamówienia i zmianie stanu taksówki. System na początku procesu ma pobierać informacje dotyczące aktualnych, czyli niezrealizowanych i nierealizowanych, nieanulowanych, niezmodyfikowanych przez pracownika centrali kursów i aktywnych, czyli nie przypisanych do aktualnych kursów zmodyfikowanych przez pracownika taksówkach. Następnie dopasować taksówki do kursów stosując algorytm genetyczny, uwzględniając parametry planowania kursów, zapisać zmiany w bazie danych. Wynik powinien zaktualizować panel aktualnych kursów pracownika centrali oraz wyświetlić powiadomienie taksówkarzowi, który otrzymał nowe kursy. Ostatni element „Przesłanie powiadomień o kursach taksówkarzom“, jak widać na rysunku 5.5, zostanie pominięty w prototypie systemu.

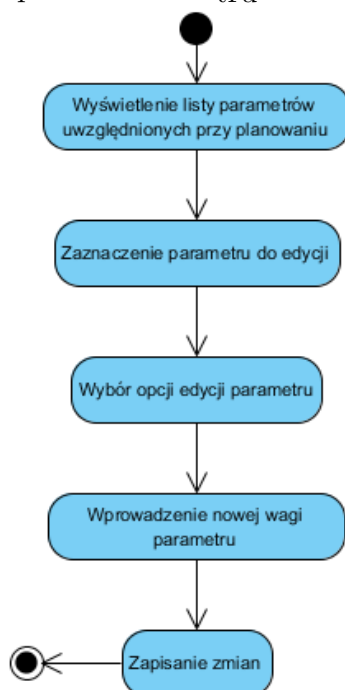
Wyświetlenie informacji o kursach, taksówkarzach i klientach odbywa się w podobny sposób: Pracownik poprzez naciśnięcie w odpowiednią pozycję w menu otwiera listę, którą zawiera dane zdefiniowane w bazie danych. Każdą listę można posortować i przefiltrować według danych, które się w niej znajdują. W przypadku kursów, domyślnie lista będzie posortowana według daty i czasu realizacji kursu oraz możliwe będzie szybkie przejście do kursów aktualnie realizowanych i następnych w kolejności poprzez naciśnięcie przycisku na stronie. Możliwe również będzie szybkie przejście do listy klientów oczekujących na kurs. Lista klientów ma zawierać informacje o wszystkich kursach realizowanych przez danego klienta i być pogrupowana według danego klienta. Lista taksówkarzy dodatkowo będzie zawierała dane o taksówkach, których używają.

Nadzór pracownikowi ułatwi również mapa, na której oznaczone zostaną pozycje i stany taksówek oraz klientów wraz z kursami planowanymi i realizowanymi, które będą oznaczone jako trasy. Kursy aktualne i zaplanowane jako następne powinny być rozróżnialne na mapie. Elementy zaznaczone na mapie będą mogły być filtrowane ze względu na taksówkarza (klienta), gdzie widoczne będą trasy realizowane przez taksówkarza (zamówione przez klienta), klienci (taksówkarze) powiązani z danymi trasami oraz sam tak-

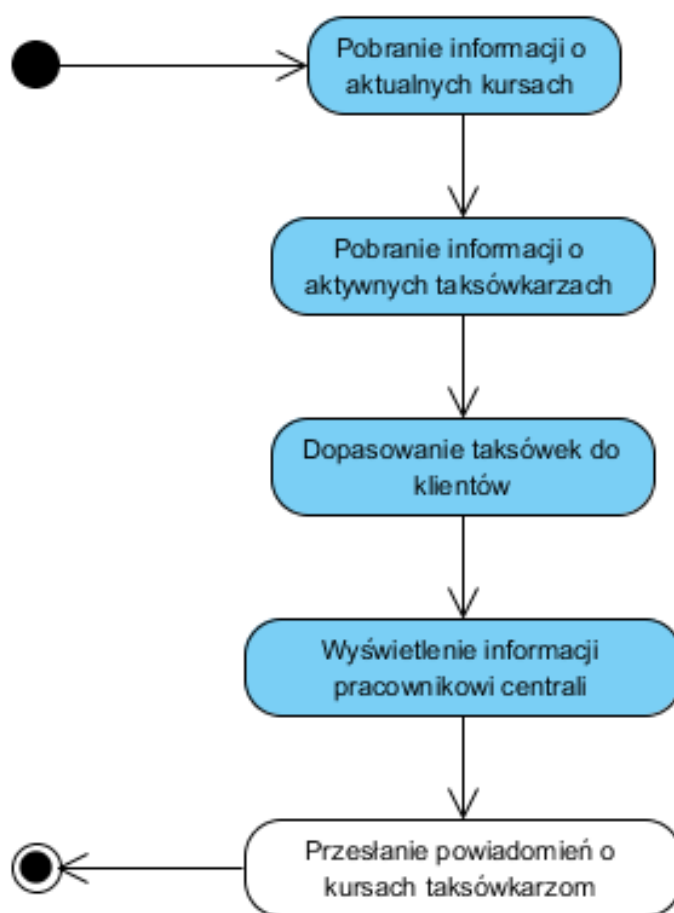


Rysunek 5.2: Przypadek użycia „Modyfikacja planowania kursów - Dodanie parametru”

Rysunek 5.3: Przypadek użycia „Modyfikacja planowania kursów - Usuwanie parametru”



Rysunek 5.4: Przypadek użycia „Modyfikacja planowania kursów - Edycja parametru”



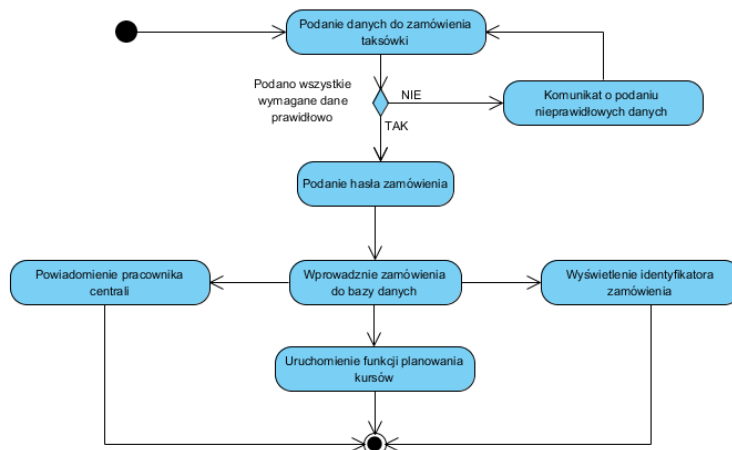
Rysunek 5.5: Przypadek użycia „Planowanie kursów“

sówkarz (klient).

Klient będzie mógł zamówić taksówkę poprzez formularz internetowy oraz dzwoniąc do centrali. W przypadku drugiej opcji pracownik centrali, który odbierze telefon wypełni identyczny formularz internetowy w celu zamówienia taksówki dla klienta, dodatkowo będzie miał podgląd na listę taksówkarzy. Formularz będzie posiadał pola:

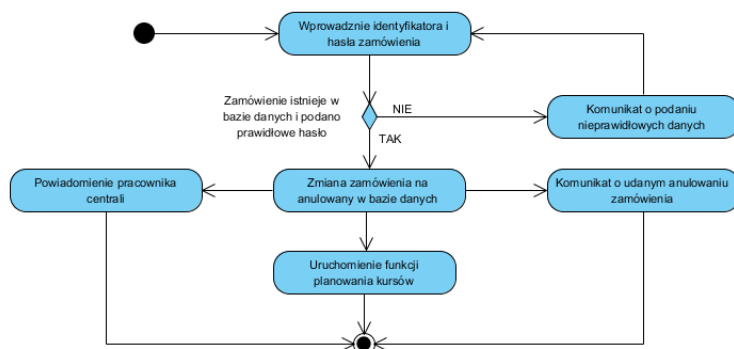
- Numer telefonu klienta
- Miasto
- Ulica
- Numer budynku
- Liczba pasażerów
- Typ taksówki - może zostać niewypełnione

Po prawidłowym wypełnieniu formularza system zapyta użytkownika o hasło zamówienia, potrzebne w razie chęci anulowania kursu. Jeśli użytkownik poda hasło, wtedy kurs zostanie zapisany do bazy danych i zostanie wyświetlony identyfikator kursu, również potrzebny do anulowania kursu. Natomiast system uruchomi funkcję planowania kursów. Anulowanie zamówienia również będzie możliwe przez formularz na stronie, do którego należy wpisać identyfikator kursu oraz hasło podane podczas tworzenia zamówienia. Jeśli podane dane są prawidłowe, system zmieni kurs jako anulowany i wywoła funkcję planowania kursów. Utworzenie i anulowanie zamówienia przedstawiłem na diagramach aktywności (Rysunek 5.6 i Rysunek 5.7). W przypadku pracownika anulowanie kursu ogranicza się do wybrania kursu z listy i wybrania opcji „Usuń”.

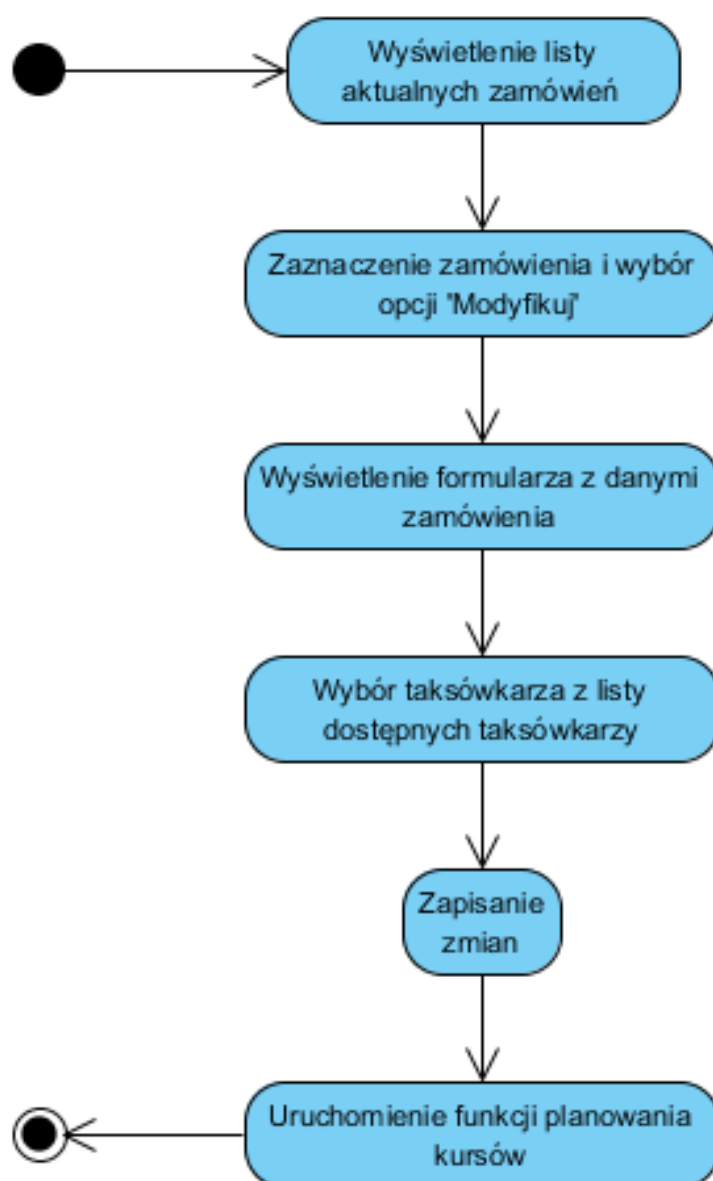


Rysunek 5.6: Przypadek użycia „Utworzenie zamówienia”

Modyfikacja zamówienia (Rysunek 5.8): W razie potrzeby pracownik centrali może modyfikować wybrany aktualny kurs, czyli przypisać wybraną przez siebie taksówkę do wybranego klienta, w przypadku gdyby chciał zmienić obecny stan. Taka funkcja dostępna będzie poprzez wybór kursu z panelu aktualnych kursów i naciśnięcie przycisku edycji. Następnie pracownik musi podać w formularzu identyfikator taksówkarza. Jednak wybór



Rysunek 5.7: Przypadek użycia „Anulowanie zamówienia“



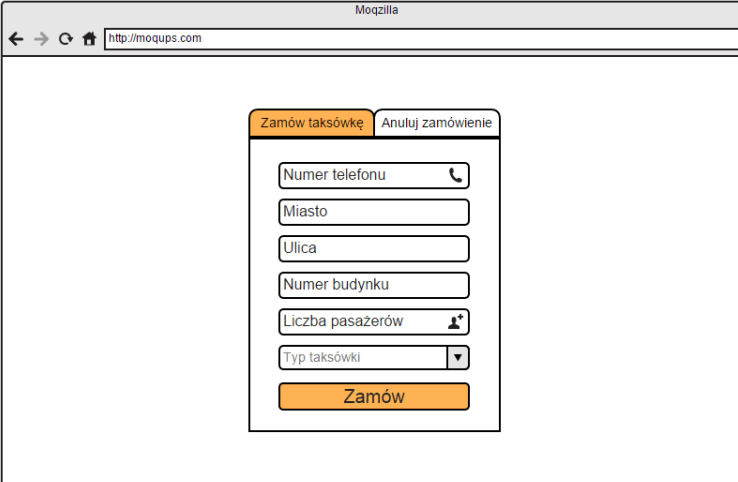
Rysunek 5.8: Przypadek użycia „Modyfikacja zamówienia“

taksówkarza będzie ograniczony. Taksówkarz nie może być w stanie „poza pracą” lub „na przerwie”. Po zatwierdzeniu modyfikacji system zapisze zmianę w bazie danych, przypisze do kursu pracownika jako zmieniającego i uruchomi funkcję planowania kursów.

5.3 Interfejs

System powinien mieć przejrzysty i prosty interfejs. Użytkownika nie powinny rozpraszać zbędne animacje czy kolory, tak żeby mógł dostrzec najważniejsze aspekty na ekranie. Pracownik musi mieć możliwość szybkiego przemieszczania się między funkcjami, dlatego też główne funkcje będą miały przypisane skróty klawiszowe, za pomocą których możliwe będzie ich uruchomienie.

Główną stroną serwisu będzie formularz zamówienia taksówki z możliwością przejścia do anulowania kursu (Rysunek 5.9). W oknie zatwierdzenia zamówienia (Rysunek 5.10) pole identyfikatora jest nieaktywne, nieedytowalne, przekazuje jedynie informacje klientowi. Natomiast hasło musi podać użytkownik. Te dwie informacje potrzebne są podczas zrezygnowania z zamówienia przez klienta, gdzie należy podać właśnie identyfikator i hasło. Jeśli podane dane są niezgodne z zapisanymi w bazie danych, system wyświetli odpowiedni komunikat (Rysunek 5.12). W przypadku niewypełnienia wymaganych pól formularzy lub podania niewłaściwego formatu danych, wyświetli się odpowiedni komunikat dla użytkownika oraz przycisk stanie się nieaktywny, co zostało pokazane na przykładzie formularzu tworzenia zamówienia na rysunku 5.13.

The image shows a web browser window with the address bar displaying 'http://moqups.com'. The page content features a central form for booking a taxi. At the top of the form are two buttons: 'Zamów taksówkę' (highlighted in orange) and 'Anuluj zamówienie'. Below these are several input fields: 'Numer telefonu' with a phone icon, 'Miasto', 'Ulica', 'Numer budynku', 'Liczba pasażerów' with a person icon, and a dropdown menu for 'Typ taksówki'. At the bottom of the form is an orange 'Zamów' button.

Rysunek 5.9: Tworzenie zamówienia przez klienta

Aby przejść do strony pracownika centrali, należy podać odpowiedni adres www w przeglądarce. Nie będzie możliwe przejście do logowania z głównej strony serwisu, aby nie zmylić klienta. Po zalogowaniu nastąpi przejście do listy taksówkarzy będących w pracy (Rysunek 5.15). Jak widzimy po lewej stronie znajduje się główne menu z zaznaczoną obecną funkcją, natomiast na górze widać podmenu. Filtry znajdują się tuż na kolumnami listy, do których są przypisane, co znacznie ułatwia znalezienie odpowiedniego filtru. Niektóre filtry umożliwiają wybór elementu z listy, ale możliwe jest również napisanie wartości z klawiatury. Jeśli wartości w kolumnach są liczbami lub czasem to często, ale nie zawsze nad nimi znajdują się dwa filtry w których podaje się zakres od-do. Listę można sortować poprzez naciśnięcie nazwy odpowiedniej kolumny. Listę można przewijać, znajdują się w niej wszystkie rekordy, ale są ładowane dynamicznie, co 25 rekordów. Opis ten

The screenshot shows a web browser window titled 'Moqups' with the address bar displaying 'http://moqups.com'. The main content area contains a form with two tabs at the top: 'Zamów taksówkę' (Order taxi) and 'Anuluj zamówienie' (Cancel order). The 'Zamów taksówkę' tab is active. Below the tabs, there is a message: 'Zapamiętaj identyfikator i hasło zamówienia. Potrzebne są w przypadku chęci rezygnacji z zamówienia.' (Remember the order identifier and password. Needed in case of wanting to cancel the order). Below this message are two input fields: 'Identyfikator' (Identifier) with the value '1212563' and 'Hasło' (Password) with a masked value. At the bottom of the form is an orange button labeled 'Zatwierdź' (Confirm).

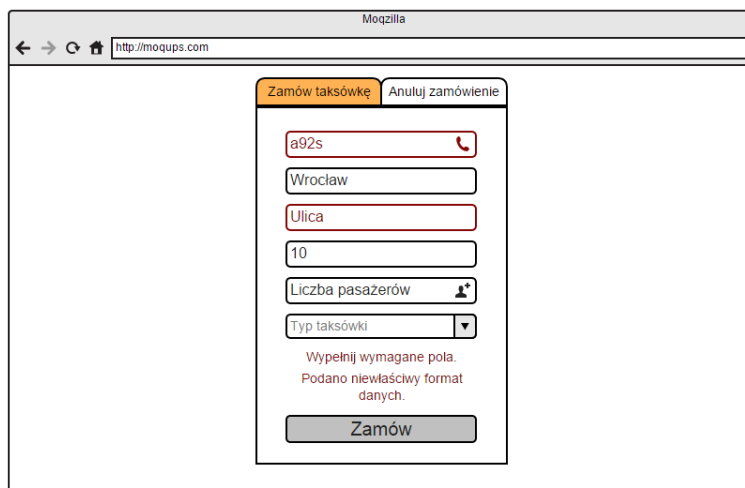
Rysunek 5.10: Zatwierdzenie zamówienia przez klienta

The screenshot shows the same web browser window as Figure 5.10. The 'Anuluj zamówienie' (Cancel order) tab is now active. The form contains the same 'Identyfikator' (Identifier) field with the value '1212563' and the 'Hasło' (Password) field with a masked value. At the bottom of the form is an orange button labeled 'Zrezygnuj z zamówienia' (Cancel order).

Rysunek 5.11: Anulowanie zamówienia przez klienta

The screenshot shows the same web browser window as Figure 5.11. The 'Anuluj zamówienie' (Cancel order) tab is active. The form contains the same 'Identyfikator' (Identifier) field with the value '1212563' and the 'Hasło' (Password) field with a masked value. At the bottom of the form is an orange button labeled 'Zrezygnuj z zamówienia' (Cancel order). Above the button, there is a red error message: 'Brak danych lub nieprawidłowe hasło.' (No data or incorrect password).

Rysunek 5.12: Dane podane podczas anulowania zamówienia są niezgodne z bazą danych

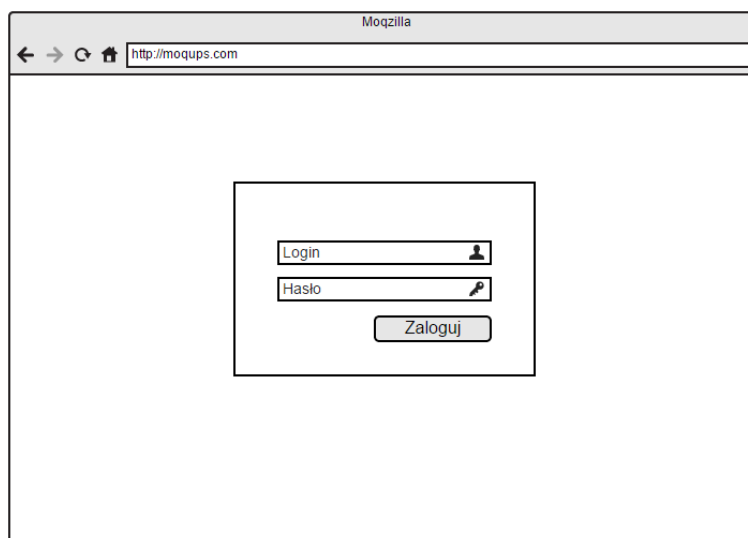


The screenshot shows a web browser window with the address bar displaying 'http://moqups.com'. The page contains a form for booking a taxi. At the top of the form are two buttons: 'Zamów taksówkę' (highlighted in orange) and 'Anuluj zamówienie'. Below these are several input fields: a phone number field containing 'a92s' with a red error message and a phone icon; a city field containing 'Wrocław'; a street name field containing 'Ulica' with a red error message; a number field containing '10'; a passenger count field with a person icon; and a dropdown menu for 'Typ taksówki'. Below the fields is a red error message: 'Wypełnij wymagane pola. Podano niewłaściwy format danych.' At the bottom of the form is a 'Zamów' button.

Rysunek 5.13: Podanie błędnych danych podczas tworzenia zamówienia przez klienta

dotyczy wszystkich list znajdujących się w serwisie, nie tylko tej dotyczącej taksówkarzy. Listę taksówkarzy wyróżnia to, że nad kolumną oznaczającą liczbę osób, którą pomieści auto istnieje tylko jeden filtr, który określa dolny zakres tej liczby. W podmenu widzimy dwie opcje, które służą do szybkiego filtrowania listy.

Na dole strony widoczny jest panel z aktualnymi i przyszłymi kursami (Rysunek 5.16),



The screenshot shows a web browser window with the address bar displaying 'http://moqups.com'. The page contains a login form with two input fields: 'Login' and 'Hasło' (Password). Below the fields is a 'Zaloguj' button.

Rysunek 5.14: Logowanie

który będzie widoczny zawsze podczas korzystania z serwisu przez pracownika centrali. Znajduje się tutaj lista tych kursów z danymi o statusie kursu, numerze telefonu klienta, liczbie osób, które taksówka ma przewieźć, identyfikatorze taksówkarza, czasie zamówienia kursu przez klienta i czasie odbioru klienta przez taksówkę. Pierwsza kolumna służy do zaznaczenia kursu, aby móc wybrany kurs zmodyfikować lub usunąć, poprzez naciśnięcie przycisków pod tabelą. Zostaną wtedy wyświetlone formularze przedstawione na rysunkach 5.17 i 5.18. W przypadku modyfikacji jedynym polem możliwym do zmiany jest „Taksówkarz“, umożliwia wybór odpowiedniego taksówkarza z listy, bądź wpisując identyfikator z klawiatury. W przypadku podania złego identyfikatora przycisk zatwierdzenia modyfikacji będzie nieaktywny. Pozostałe pola są wyłącznie w celu podania informacji

		Id	Stan	Status	Imię i Nazwisko	Standard	Ile
Kursy	Klienci	13256	Wolny	II	Marcin Nowak	Ekskluzywne	5
		11236	Przypisany do kursu	I	Mariusz Kowalski	Standardowe	9
		12092	Przypisany do kursu	II	Jan Nowak	Ekskluzywne	5
		11921	Realizuje kurs	I	Juliusz Nowak	Standardowe	5
		12121	Realizuje kurs	II	Jakub Korzeń	Sportowe	2
	11020	Przerwa	I	Magdalena Polak	Ekonomiczne	5	
	-	-	-	-	-	-	

Status	Klient	Taksówkarz	Czas zamówienia	Czas odbioru
<input checked="" type="radio"/> Oczekuje	22222222	11236	11:15	-
<input type="radio"/> Oczekuje	11111111	12092	11:09	-
<input type="radio"/> Realizowany	601782463	11921	11:05	11:08
<input type="radio"/> Realizowany	111222333	12121	11:00	11:04

Modyfikuj

Rysunek 5.15: Lista taksówkarzy, którzy są obecnie w pracy

o kursie. Jeśli pracownik zechce usunąć kurs, zostanie wyświetlony komunikat, na który zostaną wyświetlone dane kursu i przyciski do zatwierdzenia lub zaniechania usunięcia. Komunikat zakryje listę aktualnych i przyszłych kursów, aby zwrócić uwagę pracownika na to, że po usunięciu kursu, lista ta ulegnie zmianie.

Stworzenie nowego kursu przez pracownika centrali jest możliwe poprzez wybranie opcji

Status	Klient	Osoba	Taksówkarz	Czas zam.	Czas odb.	Adres odb.
<input checked="" type="radio"/> Oczekuje	222 222 222	2	11236	11:15	-	Wrocław Kościuszki 21
<input type="radio"/> Oczekuje	222 222 223	1	12092	11:09	-	Wrocław Prądyńskiego 49
<input type="radio"/> Realizowany	401 782 453	4	11921	11:05	11:08	Wrocław Grunwaldzka 13
<input type="radio"/> Realizowany	411 222 333	1	12121	11:00	11:04	Wrocław Miłosa 26
-	-	-	-	-	-	-

Modyfikuj Usuń

Rysunek 5.16: Lista kursów aktualnych i przyszłych

„Nowy kurs“ z głównego menu lub wybór opcji „Kursy“, a następnie naciśnięcie przycisku „+“ nad listą kursów lub wybranie opcji z podmenu „Stwórz“. Wszystkie trzy opcje przekierują pracownika do tego samego formularza, który widnieje na rysunku 5.19. Formularz zawiera te same pola, co w przypadku zamówienia taksówki przez klienta. Podanie złego formatu danych lub nie podanie wymaganych danych spowodują dezaktywację przycisku „Stwórz“.

Podczas działania funkcji planowania kursów lista kursów oraz lista aktualnych i przyszłych kursów jest zakrywana przez półprzezroczysty blok z napisem „Planowanie tras“ (Rysunek 5.20). Jest to komunikat dla pracownika, że za chwilę mogą zmienić się szczegóły zaplanowanych tras.

Listę tras widoczną na rysunku 5.21 można włączyć przez wybór opcji z głównego menu oraz przez wybór opcji z podmenu „Lista“. Jest to standardowa lista z głównymi szczegółami tras, ale po naciśnięciu na wybraną trasę pracownik może podejrzeć inne detale, takie jak:

- koszt,

Możesz

← → ↻ http://moqups.com

Modyfikuj Stwórz Lista

Nowy kurs

Kursy

Klenci

Taksówkarze

Mapa

Kurs #23013

Taksówkarz: 11236

Klient: 222 222 222

Status: Oczekuje

Osób: 2

Czas zamówienia: 11:15

Modyfikuj

Aktualne i przyszłe kursy

Od: Do: Od: Do: Od: Do: Od: Do:

▼ Status	▼ Klient	▼ Osób	▼ Taksówkarz	▼ Czas zamówienia	▼ Czas odbioru
○ Oczekuje	222 222 222	2	11236	11:15	-
○ Oczekuje	222 222 223	1	12092	11:09	-
○ Realizowany	601 782 483	4	11921	11:05	11:08
○ Realizowany	411 222 333	1	12121	11:00	11:04
-	-	-	-	-	-

Modyfikuj **Usuń**

Rysunek 5.17: Modyfikacja wybranego kursu

Możesz

← → ↻ http://moqups.com

W pracy Wszyscy

Nowy kurs

Kursy

Klenci

Taksówkarze

Mapa

Id	Stan	Status	Imię i Nazwisko	Standard	Ile	
▼ Id	▼ Stan	▼ Status	▼ Imię	▼ Nazwisko	▼ Standard auta	▼ Osób
13256	Wolny	II	Marcin	Nowak	Ekskluzywne	5
11236	Przypisany do kursu	I	Mariusz	Kowalski	Standardowe	9
12092	Przypisany do kursu	II	Jan	Nowak	Ekskluzywne	5
11921	Realizuje kurs	I	Juliusz	Nowak	Standardowe	5
12121	Realizuje kurs	II	Jakub	Korzeń	Sportowe	2
11020	Przena	I	Magdalena	Polak	Ekonomiczne	5
-	-	-	-	-	-	-

Czy na pewno usunąć wybrany kurs?

Id	Status	Klient	Osób	Taksówkarz	Czas zamówienia
23013	Oczekuje	222 222 222	2	11236	11:15

Usuń **Anuluj**

Rysunek 5.18: Usunięcie wybranego kursu

Nowy kurs

Klient: 222 222 222

Miasto: Wrocław

Ulica: Kościuszki

Numer budynku: 21

Liczba pasażerów: 2

Typ auta: Standardowy

Stwórz

Aktualne i przyszłe kursy

Status	Klient	Do	Taksówkarz	Do	Do	Do
Oczekuje	222 222 222	2	11236	11:15	-	-
Oczekuje	222 222 222	1	12092	11:09	-	-
Realizowany	601 782 483	4	11921	11:05	11:08	-
Realizowany	411 222 333	1	12121	11:00	11:04	-
-	-	-	-	-	-	-

Modyfikuj Usun

Rysunek 5.19: Stworzenie nowego kursu

Nowy kurs

Kursy

Klienci

Taksówkarze

Mapa

Aktualne i przyszłe kursy

Id	Status	Klient	Osób	Taksówkarz	Czes zam	Czes odb	Adres odb	Adres dost
30258	Oczekuje	222 222 222	2	11236	11:15	-	Wrocław Kościuszki 21	-
30259	Oczekuje	222 222 222	1	12092	11:09	-	Wrocław Prądyńskiego 49	-
30259	Oczekuje	222 222 222	1	12092	11:09	-	Wrocław Świstackiego 1	-
30254	Realizowany	601 782 483	4	11921	11:05	11:08	Wrocław Grunwaldzka 13	-
30252	Realizowany	411 222 333	1	12121	11:00	11:04	Wrocław Miłosza 25	-
30251	Zrealizowany	411 222 333	1	12121	11:00	11:04	Wrocław Jana Pawła II 5	Grunwaldzka 1

Planowanie tras

Modyfikuj Usun

Rysunek 5.20: Planowanie kursów

- anulowany - czy kurs został anulowany,
- zmieniający - pracownik centrali, który modyfikował jako ostatni dany kurs,
- czas do odbioru - czas od zgłoszenia do odbioru,
- czas do dostarczenia - czas od odbioru do dostarczenia do celu,
- całkowity czas - czas od zgłoszenia do dostarczenia do celu.

Po naciśnięciu w podmenu „Modyfikuj“ pracownik zobaczy listę ograniczoną do tras ze statusem „Oczekuje“ i możliwością zaznaczenia jednej z nich. Musi wybrać trasę i nacisnąć przycisk „Dalej“, wtedy ukaże się formularz znany z modyfikacji trasy wybranej z listy tras aktualnych i przyszłych (Rysunek 5.17).

Pracownik może również zobaczyć listę klientów (Rysunek 5.22). Tutaj zobaczymy trasy

ID	Status	Klient	Taksówkarz	Data	Czas zam.	Czas odb.	Czas dost.	Adres odb.	Adres dost.
30256	Oczekuje	222 222 222	2	11236	05-09-2015	11:15	-	Wrocław Kościuszki 21	-
30255	Oczekuje	222 222 222	1	12092	05-09-2015	11:09	-	Wrocław Prądnickiego 49	-
30255	Oczekuje	222 222 222	1	12092	05-09-2015	11:09	-	Wrocław Świdackiego 1	-
30254	Realizowany	501 782 453	4	11921	05-09-2015	11:05	11:08	Wrocław Grunwaldzka 13	-
30252	Realizowany	411 782 553	1	12121	05-09-2015	11:00	11:04	Wrocław Miłostwa 28	-
30251	Zrealizowany	412 221 353	1	12121	05-09-2015	11:00	11:04	Wrocław Jana Pawła II 3	Grunwaldzka 1

Koszt: - Anulowany: Nie Zmieniający: 1027 Czas do odbioru: 00:04 Czas do dostarczenia: 00:11 Czas całkowity: 00:15

Rysunek 5.21: Lista kursów

konkretnego klienta, którego numer telefonu podamy w polu „Klient“. Lista wygląda identycznie jak w przypadku zwykłej listy kursów, natomiast jest filtrowana od razu po kliencie.

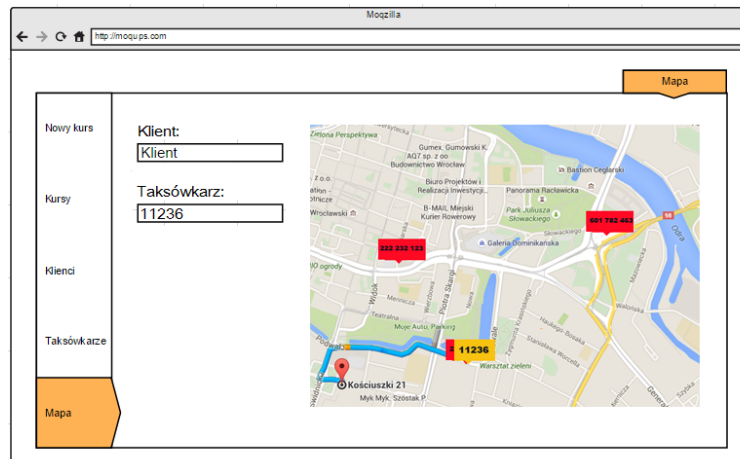
Ostatni interfejs dotyczy mapy kursów, przedstawiony na rysunku 5.23. Obiekty na ma-

ID	Status	Osób	Taksówkarz	Data	Czas zam.	Czas odb.	Czas dost.	Adres odb.	Adres dost.
30256	Oczekuje	2	11236	05-09-2015	11:15	-	-	Wrocław Kościuszki 21	-
30251	Zrealizowany	1	12121	02-09-2015	11:00	11:04	11:15	Wrocław Jana Pawła II 3	Grunwaldzka 1
30250	Zrealizowany	2	12121	24-08-2015	10:58	11:03	11:20	Wrocław Kościuszki 21	Plac dominikański 20

Koszt: - Anulowany: Nie Zmieniający: 1027 Czas do odbioru: 00:04 Czas do dostarczenia: 00:11 Czas całkowity: 00:15

Rysunek 5.22: Lista klientów

pie można przefiltrować podając identyfikator taksówkarza lub klienta. Kursy aktualnie realizowane zaznaczone są na mapie niebieską linią od adresu odbioru do pozycji taksówkarza. Taksówkarze zaznaczeni są żółtym znacznikiem z numerem identyfikatora, natomiast klienci na czerwonym znaczniku z numerem telefonu.

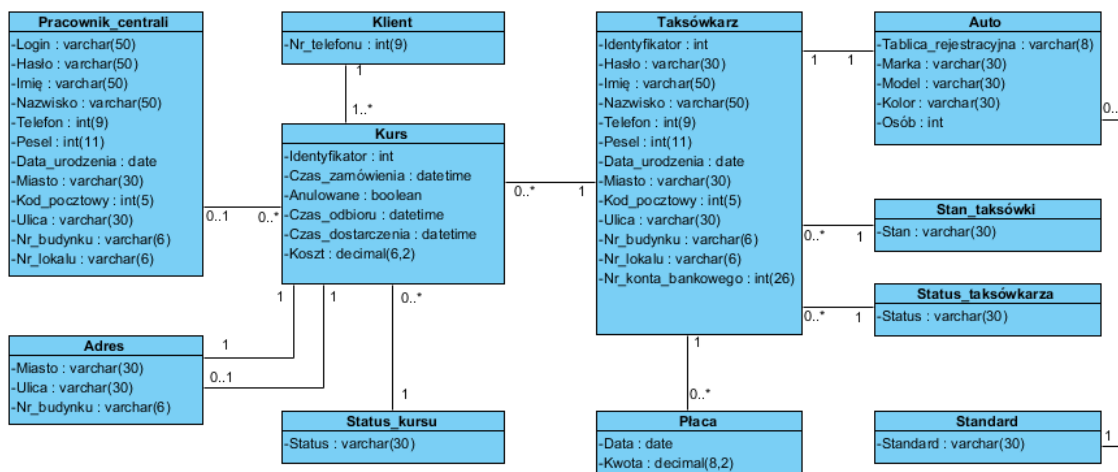


Rysunek 5.23: Mapa

5.4 Baza danych

5.4.1 Model logiczny

W tym podrozdziale przedstawię model logiczny bazy danych. Dane będą przetrzymywane w postaci znormalizowanej. Każda relacja będzie w 3 postaci normalnej. Baza danych będzie zawierała tabele ukazane na diagramie klas na rysunku 5.24. Na diagramie widać również atrybuty tabel wraz z ich typami i maksymalnym rozmiarem. Również widać połączenia między tabelami, które krótko opiszę w tabeli 5.1. Słownik bazy danych, przedstawia znaczenie tabel i atrybutów (Rysunek 5.2).



Rysunek 5.24: Diagram klas, model logiczny bazy danych

Nazwa tabeli	Krotność	Nazwa tabeli	Krotność	Opis
Pracownik centrali	0..1	Kurs	0..*	Kurs może być zmieniony przez wielu pracowników centrali, ale tylko ostatni zmieniający zostanie zapamiętany w bazie danych.
Klient	0..1	Kurs	1	Klient musi zrealizować co najmniej jeden kurs, aby zostać zapamiętany w bazie danych
Adres	1	Kurs	1	Kurs posiada adres miejsca, z którego klient został odebrany
Adres	0..1	Kurs	1	Kurs może posiadać adres miejsca, do którego klient został dowieziony
Status kursu	1	Kurs	0..*	Kurs musi mieć przypisany jeden ze statusów
Taksówkarz	1	Kurs	0..*	Kurs musi mieć przypisanego taksówkarza który zrealizuje, bądź zrealizował kurs
Taksówkarz	1	Auto	1	Taksówkarz musi posiadać auto, w którym wykonuje pracę
Taksówkarz	1	Stan taksówki	0..*	Taksówkarz musi mieć określony stan, w jakim obecnie się znajduje, pod względem pracy
Taksówkarz	1	Status taksówkarza	0..*	Taksówkarz musi mieć status w firmie taksówkowej
Taksówkarz	1	Płaca	0..*	Płatność za pracę muszą być przypisane do jednego taksówkarza
Standard	1	Auto	0..*	Każde auto musi mieć określony standard

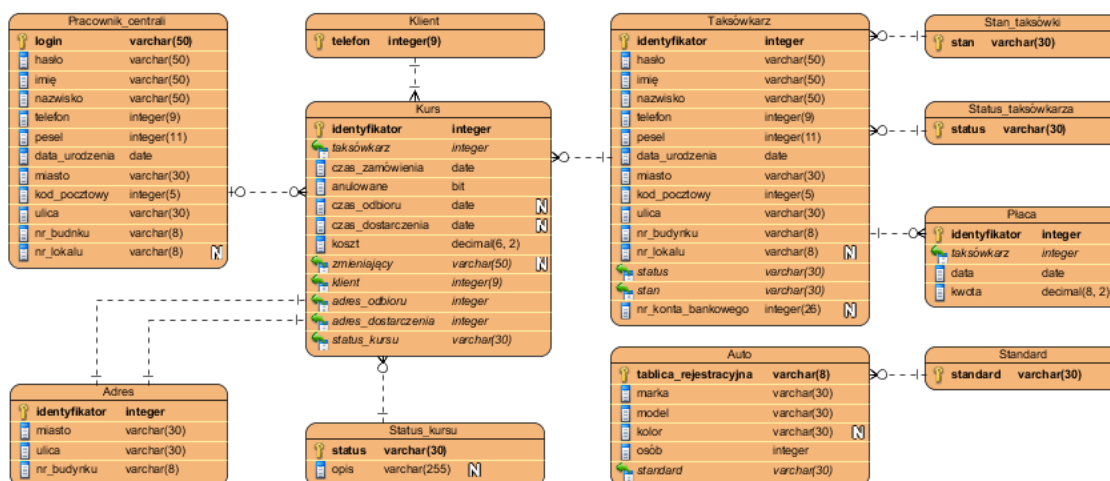
Tablica 5.1: Połączenia relacyjne

Nazwa	Opis
Stan taksówki	Określa stan w jakim może znajdować się taksówka, możliwe wartości to: wolny, przypisany do kursu, realizuje kurs, przerwa, poza pracą.
Status taksówkarza	Określa stopień taksówkarza w firmie.
Osób	Określa ile osób może pomieścić auto taksówkarza.
Standard	Określa standard auta taksówkarza. Możliwe wartości: ekskluzywne, ekonomiczne, sportowe, standardowe
Czas zamówienia	Data i czas złożenia zamówienia przez klienta.
Czas odbioru	Data i czas odbioru klienta przez taksówkę.
Czas dostarczenia	Data i czas dostarczenia klienta do celu.
Anulowane	Określa czy zamówienie zostało anulowane.
Status kursu	Określa czy kurs jest realizowany, zrealizowany lub oczekuje na realizację.
Płace	Tabela potrzebna do rejestrowania wypłat dla taksówkarzy.

Tablica 5.2: Słownik bazy danych

5.4.2 Model fizyczny

W systemie zostanie wykorzystana relacyjna baza danych. Co w znacznym stopniu przyspieszy dodawanie i modyfikowanie danych w stosunku do, np. hurtowni danych. Jest to ważny aspekt systemu, ponieważ dane będą często dodawane (tworzenie kursów) oraz modyfikowane (stan taksówek). Na modelu fizycznym przedstawiłem obligatoryjność danych, klucze podstawowe encji oraz klucze obce powstałe z połączeń relacyjnych encji (Rysunek 5.25). Poniżej widnieje również słownik bazy danych, przedstawiający znaczenie kluczy obcych (Rysunek 5.3).



Rysunek 5.25: Diagram encji, model fizyczny bazy danych

Nazwa	Opis
Miejsce odbioru	Adres miejsca, z którego klient został odebrany.
Miejsce dostarczenia	Adres miejsca, do którego klient został zawieziony.
Zmieniający	W przypadku modyfikacji kursu przez pracownika centrali, zostanie on przypisany do kursu, jako zmieniający.

Tablica 5.3: Znaczenie kluczy obcych

5.5 Wzorzec architektoniczny

5.5.1 Opis wzorca

W pracy zostanie wykorzystany wzorzec architektoniczny *MVC* (*Model-View-Controller*), który jest „wzorcem projektowym do tworzenia aplikacji internetowych”. „Wzorzec *Model-View-Controller* jest stworzony z trzech następujących części:

- *Model* - najniższa warstwa wzorca odpowiedzialna za utrzymywanie danych,
- *View* - odpowiedzialny za wyświetlanie całości lub części danych użytkownikowi,
- *Controller* - jest oprogramowaniem, które kontroluje interakcje między Modelem i Widokiem“.

5.5.2 Zastosowanie w systemie

Dla ułatwienia i wprowadzenia pewnego standardu wprowadzenia wzorca do systemu zastosowany będzie framework *Sails.js*, który jest dostosowany do wybranego wzorca *MVC*.

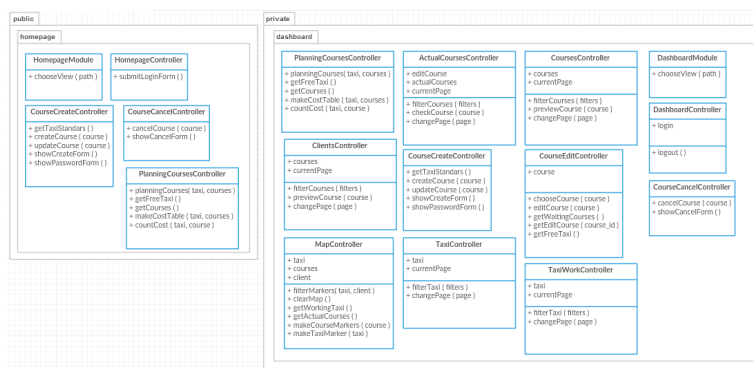
Warstwa Modelu zawarta zostanie w bazie danych stworzonej w technologii *MySQL*. Wszystkie tabele, które mają zostać stworzone wraz z kolumnami opisane zostały w poprzednim podrozdziale 5.4. Zgodnie z tabelami zostaną utworzone obiekty, które będą odwzorowywać wiersze w tabelach.

Warstwa Kontrolera zaimplementowana zostanie w technologii *Node.js* wraz z framework’iem *Sails.js*. Uruchomienie serwera jest bardzo proste i zostało dokładnie opisane na oficjalnej stronie narzędzia. Połączenie z bazą danych umożliwia dostępna biblioteka *sails-mysql*, służąca do łatwego połączenia serwera z bazą. Zostaną stworzone oddzielne moduły do zarządzania użytkownikami systemu, zarządzania kursami i taksówkarzami w firmie. Każdy z nich będzie posiadał pliki odpowiedzialne za komunikację z obiektami warstwy modelu związanych z modułem, czyli dostarczenie funkcji czytania, modyfikowania i dodawania danych do bazy. Nazwy tych plików będą nawiązywać do nazw obiektów. Warstwa Widoku realizowana będzie przez technologie *html5*, *css3*, *javascript* z wykorzystaniem frameworka *AngularJS*, który usprawni działanie asynchroniczne systemu po stronie klienta. Również ta warstwa zostanie podzielona na moduły, podobne jak w warstwie Kontrolera, jednak moduły podzielone zostaną ze względu na dostęp użytkowników,

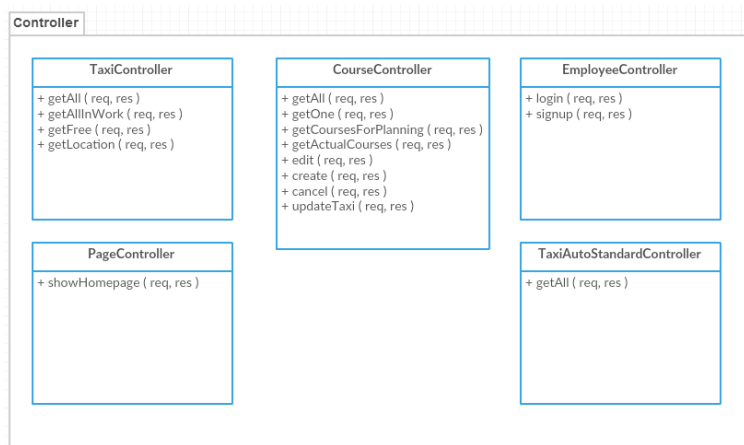
czyli publiczne, dla klientów i prywatne, dla pracowników centrali. Dokładny podział zamieszczony został na obrazkach 5.26, 5.27 i 5.28.

5.6 Struktura oprogramowania

Na poniższych rysunkach widać podział systemu ze względu na moduły i funkcje jakie wykonuje. Jak już zostało wspomniane wcześniej, w modelu występują obiekty, które



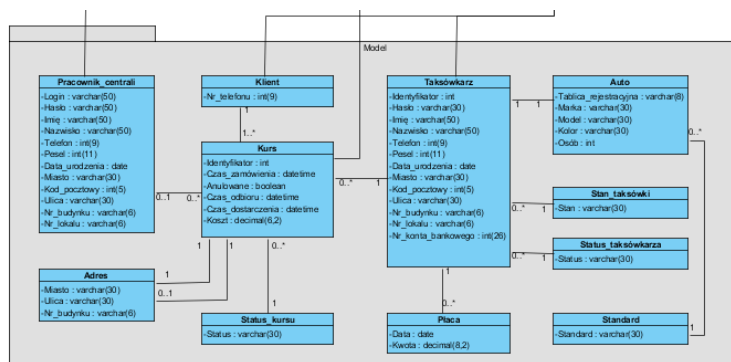
Rysunek 5.26: Widok



Rysunek 5.27: Kontroler

odzworowują wiersze w tabelach w bazie danych. Zostały opisane one dokładnie w poprzednim podrozdziale. W kontrolerze występują głównie funkcje manipulujące obiektami lub pobierające je i przekazujące do widoku. Na diagramie widoczne są czasami argumenty *req*, *res*, oznaczają one kolejno parametry otrzymane oraz odpowiedź, którą należy przekazać do widoku.

Zaczynając od klasy Kursów, czyli *CourseController*, która posiada funkcje pobierania kursów poprzez podanie w argumencie *req* obiekt z filtry, co spowoduje zwrócenie listy kursów, zgodnych z określonymi warunkami. Wydzielono również funkcje, które zwracają specyficzne kursy, np. lista kursów aktualnych *getActualCourses*, kursów do zaplanowania *getCoursesForPlanning*. Możliwe jest również dodanie kursu metodą *add* i anulowanie metodą *remove*. Ważną własnością jest planowanie, która jest dostępna właśnie w tym module oraz *modyfikacjaKursu*, gdzie jako argumenty wymagane jest podanie id kursu,



Rysunek 5.28: Model

który będzie zmieniony, *id taksówkarza*, który będzie go realizował i *id pracownika centrali*, który dokonuje tej modyfikacji.

Klasa *TaxiController* umożliwia zwrócenie listy taksówkarzy, którzy są w danej chwili wolni, czyli którzy są w pracy i nie realizują żadnego zlecenia, taksówkarzy, którzy są obecnie w pracy oraz wszystkich taksówkarzy. Możliwe jest również pobranie lokacji w jakiej znajduje się taksówkarz, jednak jest to jedynie testowa lokalizacja zapisana w bazie. W praktyce powinna być pobierana ona z osobnej aplikacji udostępniającej ową lokalizację.

Klasa *TaxiAutoStandardController* służy jedynie do pobrania wszystkich możliwych standardów aut. Klasa *EmployeeController* potrzebna jest do zalogowania i zarejestrowania pracownika centrali w systemie. Klasa *PageController* służy do wyboru strony głównej ze względu na to czy użytkownik jest zalogowany czy też nie.

Większość logiki systemu została przeniesiona do warstwy widoku. Została ona podzielona na dwa pakiety: *public/homepage* oraz *private/dashboard*. Pierwszy pakiet jest dostępny dla niezalogowanego użytkownika, a więc dla klientów firmy. Znajdują się tu klasy do tworzenia nowych zamówień: *CourseCreateController*, anulowania zamówień: *CourseCancelController* oraz do zarządzania wyświetlanymi szablonami: *HomepageModule* i logowania pracownika centrali do systemu: *HomepageController*. Jest tutaj również klasa odpowiedzialna za zaplanowanie kursów w razie stworzenia lub anulowania jakiegoś przez klienta. Pakiet drugi stworzony został dla zalogowanych użytkowników, a więc dla pracowników centrali. Klasy odpowiedzialne za wyświetlanie lub tworzenie, modyfikowanie i anulowanie są dostępne w tym pakiecie. Osobne klasy odpowiadają za wyświetlenie wszystkich czy też tylko aktualnych kursów. Klasa wyświetlająca kursy wybranego klienta zdefiniowana została jako *ClientsCourses*. Występują tutaj również klasy odpowiedzialne za wyświetlenie listy wszystkich taksówkarzy oraz tylko będących w pracy. Podobnie jak w pakiecie pierwszym, również tutaj znajduje się klasa do zaplanowania kursów i wygląda identycznie. Kolejny moduł dotyczy mapy. Posiada on interfejsy, które są wzorem obiektów, które mapa będzie obsługiwać. W przypadku tego systemu znaczniki będą taksówką lub adresami kursu, na co wskazuje typ znacznika. Mapa będzie obsługiwana przez *Google Maps*. Klasa mapy posiada wiele funkcji, służących do pokazania wszystkich lub wybranych tras i taksówkarzy. Umożliwia również dodanie trasy lub znacznika do zapamiętanych lub całkowite wyczyszczenie mapy.

Klasa *DashboardModule* służy do przełączania między widokami ładując jedynie szablony, nie całe strony. Dzięki niej strona posiada mechanizm *SPA*.

5.7 Podsumowanie

Funkcje systemu stworzone są głównie dla pracownika centrali, ponieważ on jest jego głównym użytkownikiem. Przypadki użycia są dość dokładne, aby zrozumieć ich znaczenie i działanie. Natomiast interfejsy dobrze obrazują system i doprecyzowują niektóre procesy. Treść rozdziału jest wystarczająca do rozpoczęcia implementowania systemu, który stanie się pomocnym narzędziem dla firm taksówkowych.

Rozdział 6

Implementacja

6.1	Wprowadzenie	37
6.2	Wykorzystane środowiska i narzędzia programistyczne	37
6.3	Kod źródłowy	38
6.4	Interfejs	43
6.5	Instalacja oprogramowania	44
6.6	Instrukcja użytkownika	45
6.7	Archiwizacja danych	49
6.8	Podsumowanie	50

6.1 Wprowadzenie

Rozdział ten ma przybliżyć sposób implementacji systemu, wskazać środowiska i narzędzia użyte w procesie implementacji, wyjaśnić niektóre algorytmy i zastosowane rozwiązania, pokazać również zmiany jakie nastąpiły w interfejsie względem fazy projektowania. Opisane zostały tutaj również instrukcje wyjaśniające sposób zainstalowania i użytkowania systemu. Są one wystarczające aby poprawnie korzystać z systemu.

6.2 Wykorzystane środowiska i narzędzia programistyczne

Do stworzenia systemu wykorzystane zostało środowisko deweloperskie *Atom*. Posiadające nieskończoną ilość pakietów, o których własności może rozszerzyć funkcje programu każdy z użytkowników. Środowisko to zostało stworzone do aplikacji zarówno desktopowych, jak również sieciowych. W podstawowej wersji wspierane jest najmocniej tworzenie z wykorzystaniem języka javascript, w tym środowisku node.js, którego użyłem w moim systemie. Użyłem pakietów wspierających pisanie kodu w użytych przeze mnie framework'ach: *Sails.js*, *AngularJS* oraz pakietu wbudowanego w program do zarządzaniem systemem kontroli wersji *Git*. Poza programem *Atom*, użyłem również narzędzia *Xampp*, aby uruchomić lokalny serwer wraz z bazą danych *MySQL*, na której zapisane są dane systemu.

6.3 Kod źródłowy

W tym podrozdziale przedstawię strukturę plików oraz istotne fragmenty kodu systemu. Dotyczyć będą one m.in. algorytmu planowania kursów, uruchomienia tego algorytmu, pobieranie, aktualizacji taksówkarza przypisanego do kursu. Struktura plików jest

```
//funkcja do aktualizacji taksówki przypisanej do kursu wraz ze stanem taksówkarza
updateTaxi: function( req, res ) {
  var id = req.body.id !== undefined ? req.body.id : 0;
  var taksowkarz = req.body.taksowkarz !== undefined ? req.body.taksowkarz : '';

  TaxiModel.update( { id: taksowkarz }, { stan: 'przypisany do kursu' } ).exec(function( err, updated ) {
  });

  CourseModel.update( { id: id }, { taksowkarz: taksowkarz } ).exec(function( err, updated ) {
    res.send( updated );
  });
}
```

Rysunek 6.1: Fragment kodu: aktualizacja kursu

dość przejrzysta, dzięki zastosowaniu framework'a *Sails.js*. Folder *api* odpowiada za część backend'ową systemu. Warstwa modelu, czyli deklaracje obiektów, będących odwzorowaniem encji w bazie danych znajdują się w folderze *api/models*, gdzie każdy plik nazwany jest angielskim odpowiednikiem nazwy encji z dopiskiem „Model”. Wszelka logika na serwerze znajduje się w plikach w folderze *api/controllers*. Każdy plik odpowiada jednemu z modeli, w pliku znajdują się funkcje potrzebne do tworzenia, usuwania, modyfikowania lub pobierania danych dotyczących modeli. W niektórych, znajdują się również funkcje zawierające modyfikacje na innych modelach niż te dla których zostały stworzone, np. zmiana stanu taksówkarza przy przypisaniu go do kursu i zmianie statusu tego kursu (Rysunek 6.1).

Prawie cała warstwa widoku, poza głównymi szablonami widoków, które znajdują się w folderze *views*, znajduje się w folderze *assets*. Mamy tutaj foldery odpowiedzialne za zdjęcia - *images*, pliki javascript - *js*, pliki css - *styles* oraz pliki szablonów - *templates*. Poza folderem *js*, pliki nie są katalogowane w innych podfolderach, ponieważ jest ich niewiele. Natomiast w folderze *js* występuje podział na pliki bibliotek zewnętrznych, posiadających potrzebne funkcje, w tym plik z moim algorytmem genetycznym pod nazwą *genetic*. Znajdują się one w folderze *dependencies*. W folderze *public* znajdują się pliki dostępne bez autoryzacji dostępu, a w nim folder *homepage* odpowiedzialny za stronę główną. W pliku *HomepageModule* zadeklarowany jest moduł framework'a *AngularJS* potrzebny do przełączania między widokami w sposób znany z aplikacji SPA. Do zarządzania tymi widokami stworzone są kontrolery framework'a *AngularJS*:

- *HomepageController* - logowanie pracownika centrali,
- *CourseCreate* - zamówienie kursu,
- *CourseCancel* - anulowanie kursu,
- *PlanningCourses* - planowanie kursów.

W folderze *js* znajduje się również folder *private*, w którym znajdują się pliki dostępne dla zalogowanego użytkownika. W nim jest folder *dashboard*, a tutaj podobnie jak w przypadku *homepage* znajduje się plik odpowiedzialny za przełączanie między widokami o nazwie *DashboardModule*. W folderze są również inne pliki:

- *ActualCoursesController* - pobranie i filtrowanie listy kursów, modyfikacja i anulowanie wybranego kursu,
- *ClientsController* - pobranie i filtrowanie listy kursów podanego klienta,
- *CourseCancelController* - anulowanie wybranego kursu,
- *CourseCreateController* - tworzenie nowego kursu,
- *CourseEditController* - modyfikacja wybranego kursu, czyli manualne przypisanie taksówkarza,
- *DashboardController* - pasek użytkownika,
- *MapController* - wyświetlanie mapy wraz z kursami i taksówkami z możliwością filtrowania,
- *PlanningCoursesController* - planowanie kursów,
- *TaxiController* - pobranie i filtrowanie listy wszystkich taksówkarzy,
- *TaxiWorkController* - pobranie i filtrowanie listy taksówkarzy będących w pracy.

W folderze głównym systemu znajdują się również foldery odpowiedzialne za konfigurację serwera, których omówienie można znaleźć na oficjalnej stronie framework'a *Sails.js* (<http://www.sailsjs.org>). Cała struktura omówionych plików pokazana została na ry-



Rysunek 6.2: Struktura plików

sunku 6.2.

```

//funkcja tworzenia tablicy kosztów
function makeCostTable( taxi, courses ) {
    var costTable = [];
    for( var i = 0; i < taxi.length; i++ ) {
        //utworz obiekt taksowki
        var taxiObject = new Taxi();
        //wypełnij tablice kosztów dojazdu do kursów
        for( var j = 0; j < courses.length; j++ ) {
            //jeśli posiada adres odbioru
            if( coursesFull[ courses[ j ] ].adres_odbioru ) {
                countCost( taxi[ i ], courses[ j ], taxiObject, courses[ j ], costTable, i, taxi.length, function() {
                    finish();
                } );
            }
            //jeśli nie posiada adresu odbioru
            else {
                taxiObject.cost[ courses[ j ] ] = 999999;
            }
        }
        //wypełnij pozostałe miejsca w tablicy kosztów dojazdu
        for( var j = courses.length; j < taxi.length; j++ ) {
            //przypisz liczbę ujemną jako brak kursu
            courses[ j ] = -1*j;
            taxiObject.cost[ -1*j ] = 999999;
        }
        //jeśli cała tablica kosztów dojazdu została wypełniona
        if( Object.size( taxiObject.cost ) == taxi.length ) {
            //dodaj obiekt taksowki do tablicy kosztów
            costTable[ i ] = taxiObject;
            finish();
        }
    }
}

//funkcja wewnętrzna do wywołania kontynuowania algorytmu
function finish() {
    //jeśli cała tablica kosztów dojazdu została wypełniona
    if( Object.size( costTable ) == taxi.length ) {
        //kontynuuj algorytm
        $( document ).trigger({
            type:"geneticStartAlgorithm",
            value:costTable
        });
        return costTable;
    }
}

```

Rysunek 6.3: Fragment kodu: tworzenie tablicy kosztów

```

//funkcja obliczania kosztu dojazdu taksowki do adresu odbioru kursu
function countCost( taxi, course, taxiObject, courseId, costTable, index, size, onComplete ) {
    //pobierz lokalizację taksowki
    $.http.get( "/taxi_location?id=" + taxi.id )
    .success( function( response ) {
        var location = response;
        //utworz adres url dla google maps API do pobrania odległości
        var url = 'https://maps.googleapis.com/maps/api/distancematrix/json?';
        //podaj adres początkowy
        url += 'origins=' + location.lat + ',' + location.lng;
        //podaj adres końcowy
        url += '&destinations=' + coursesFull[course].adres_odbioru.nr_budynku + '+'
            + coursesFull[course].adres_odbioru.ulica + '+' + coursesFull[course].adres_odbioru.miasto + '+Polska';
        //podaj klucz do google API
        url += '&key=AIzaSyBXTZogSxhfwrK_smDpOTFUBsWyoKW9ejU';

        //pobierz odległość taksowki od adresu odbioru z google maps API
        $.http.get( url ).success( function( response ) {
            var routeDistance = response.rows[0].elements[0].distance.value;

            //pobierz zysk taksowkarza w bieżącym miesiącu
            $.http.get( "/taxi_profit?id=" + taxi.id )
            .success( function( response ) {
                var profit = response;
                //zamien status taksowkarza na wartość liczbową
                var status = getValueOfTaxiStatus( taxi.status );
                //oblicz koszt dojazdu
                taxiObject.cost[ courseId ] = routeDistance / ( profit * status );
                //jeśli cała tablica kosztów dojazdu została wypełniona
                if( Object.size( taxiObject.cost ) == size ) {
                    //dodaj obiekt taksowki do tablicy kosztów
                    costTable[ index ] = taxiObject;
                    onComplete();
                }
            } );
        } );
    } );
}

```

Rysunek 6.4: Fragment kodu: obliczenie kosztu dojazdu

Algorytm planowania kursów jest algorytmem genetycznym, a dokładniej algorytmem posiadający mechanizmy typowe dla algorytmu genetycznego. W algorytmie zauważymy populacje (ang. population), w których zawarte są osobniki lub rozwiązania (ang. solutions), składające się z genów (ang. genes). Działanie algorytmu sprowadza się do powtarzania procesu generowania nowej populacji i wyznaczania najlepszego osobnika. Generowanie populacji polega na losowaniu osobników i mieszaniu ich genów, proces ten zwany jest krzyżowaniem, w celu utworzenia nowych osobników, następnie utworzone osobniki są mutowane, czyli zmieniane są dwa geny losowo wybrane.

W przypadku mojego systemu liczba genów jest uzależniona od liczby wolnych taksówek, natomiast samymi genami są identyfikatory kursów. W przypadku kiedy kursów jest mniej niż wyznaczona liczba genów, wstawiane są liczby ujemne, oznaczające brak kursu. Osobnik jest zbiorem genów w różnej kolejności. Kolejność ta ma bardzo duże znaczenie, ponieważ na jej podstawie określany jest najlepszy osobnik z populacji. Aby określić koszt każdego z osobników należy utworzyć tablicę kosztu, która zawiera obiekty taksówek, które z kolei posiadają tablicę kosztu dojazdu do każdego kursu. Koszt ten jest liczony nie tylko na podstawie odległości, ale również na podstawie zysków uzyskanych w danym miesiącu oraz statusu taksówkarza w firmie.

Wzór do obliczenia kosztu wygląda następująco:

$\text{Koszt} = \text{odległość} / (\text{zysk} * \text{status})$, gdzie: odległość - odległość taksówki od adresu początkowego kursu, zysk - zysk uzyskany przez taksówkarza w danym miesiącu, status - status taksówkarza w firmie, np. 1 lub $\frac{1}{2}$.

Koszt najmniejszy jest kosztem najlepszym. Czyli szukane jest rozwiązanie z jak najmniejszym kosztem. W przypadku kursów bez podanego adresu odbioru lub kiedy tablica taksówek jest większa od tablicy kursów, pozycje te w tablicy kosztów otrzymują dużą wartość, aby zwiększać sumę kosztów całego osobnika (rozwiązania) w algorytmie. Funkcja do tworzenia tablicy kosztów oraz wyliczania kosztu dojazdu widoczne są na rysunkach 6.3, 6.4.

Uruchomienie algorytmu planowania następuje po wielu procesach zachodzących w systemie, dlatego utworzyłem osobny kontroler odpowiedzialny za pełną logikę zaplanowania kursów nazwany *PlanningCoursesController*. Uruchomienie planowania wykonuje się poprzez zmianę widoku przy pomocy języka javascript (`window.location.href='#/courses/planning'`).

Wywołanie kontrolera powoduje pobranie niezbędnych danych o kursach oraz taksówkach. Ze względu na asynchroniczny charakter środowiska, dane pobierane są osobno, aby zmniejszyć czas, jednak wymaga to wywołania dalszego działania algorytmu w przypadku ukończenia każdej z funkcji. Odpowiada za to globalne wydarzenie o nazwie *geneticStart*, które wyzwala utworzenie tablicy kosztów, uruchamiane jeśli dane o kursach i taksówkach zostały już pobrane. Wymaga to pobrania danych odległości taksówek od adresów początkowych kursów przy pomocy *google maps API*, proces ten również działa w sposób asynchroniczny, dlatego wyzwolenie już właściwego algorytmu planowania i wyznaczenia najlepszej konfiguracji kursów z taksówkami oparte jest o wydarzenie globalne nazwane *geneticStartAlgorithm* w parametrze, którego przekazana jest tablica kosztów. Po ukończeniu algorytmu, należy zapisać zmiany w bazie danych i odświeżyć informacje wyświetlane na stronie do czego służy globalne wydarzenie *interfaceUpdate*. Wywołanie algorytmu oraz ogólna obsługa po zakończeniu algorytmu pokazana została na rysunku 6.5.

```

// Wywołanie globalnego wydarzenia geneticStart
$( document ).on( 'geneticStart', function() {
    //sprawdzenie czy ukończono pobieranie danych o taksówkach i kursach
    if( checkIndex.taxiDownloadComplete && checkIndex.coursesDownloadComplete ) {

        //dopasowanie rozmiaru tablicy kursów
        if( taxi.length < courses.length ) {
            var size = taxi.length;
            courses = courses.slice( 0, size );
        }

        //utworzenie tablicy kosztów
        makeCostTable( taxi, courses );

        $( document ).on( 'geneticStartAlgorithm', function( data ) {
            //przypisanie utworzonej tablicy kosztów do zmiennej
            var costTable = data.value;

            //utworzenie obiektu algorytmu genetycznego, parametry:
            //tablica identyfikatorów kursów, tablica kosztów, liczba osobników w populacji,
            //prawdopodobieństwo mutacji, maksymalna liczba pętli
            var geneticAlgorithm = new GeneticAlgorithm( courses, costTable, 500, 0.7, 300 );

            geneticAlgorithm.start();

            //pętla po kursach w najlepszym znalezionym rozwiązaniu
            for( var i = 0; i < geneticAlgorithm.population.best.genes.length; i++ ) {
                //sprawdzenie czy element jest kursem
                if( geneticAlgorithm.population.best.genes[i] > 0 ) {
                    //zapisanie zmian w bazie danych
                    $http.post( "/update_taxi_course", {
                        id: geneticAlgorithm.population.best.genes[i],
                        taksowkarz: taxi[i].id
                    } ).success( function( course, status ) {
                        //wyświetlenie listy kursów
                        window.location.href = '/#/courses';
                        //zaktualizowanie widoków
                        $( document ).trigger({
                            type:"interfaceUpdate"
                        });
                    });
                }
            }
        });
    }
});

```

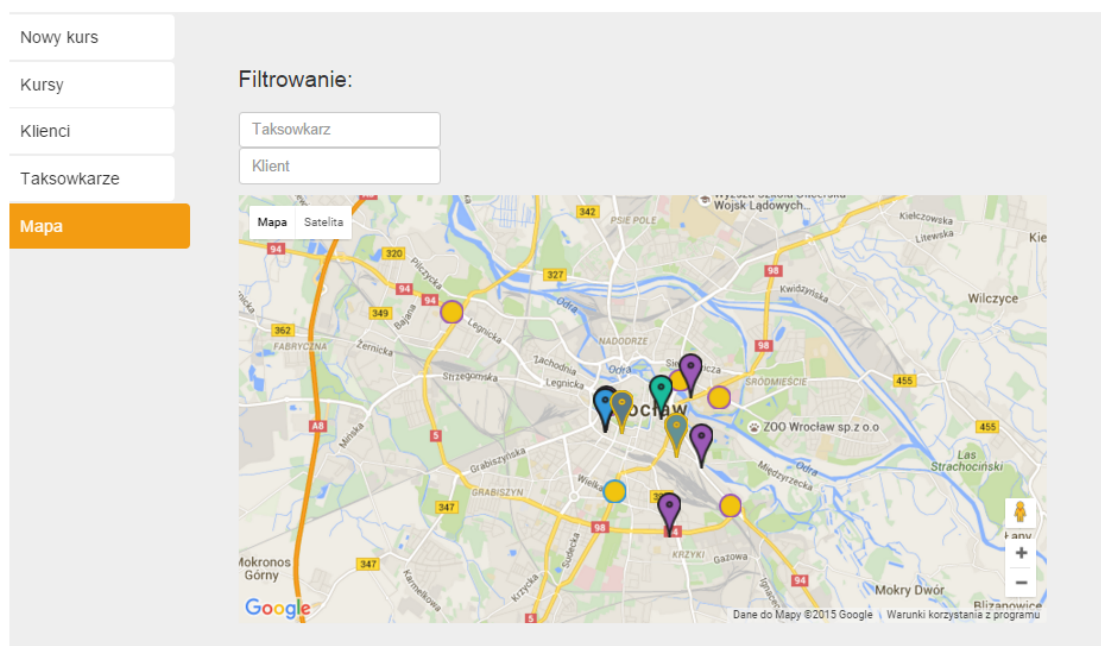
Rysunek 6.5: Fragment kodu: wywołanie i obsługa po zakończeniu algorytmu

6.4 Interfejs

Id	Stan	Status	Imię i nazwisko	Standard	Ile
2	przypisany do kursu	1/2	Jan Taksówkarz	sportowe	2
1	realizuje kurs	1	Adam Myszowski	ekskluzywne	4

Rysunek 6.6: Obecny wygląd formularza logowania

Pomiędzy fazą projektowania systemu, a fazą implementacji powstało kilka zmian dotyczących wyglądu interfejsów. Pierwszą zmianą jest wygląd formularza logowania pracownika (Rysunek 6.6). Formularz zamiast być oddzielnym widokiem, stał się fragmentem innych widoków, towarzyszący, na zmianę z formularzem wylogowania użytkownika, każdemu z widoków. Jest on widoczny po prawej stronie na górze strony. Przy zalogowaniu widoczny jest również login użytkownika. Zmiany nastąpiły również w wyglądzie widoku



Rysunek 6.7: Obecny wygląd mapy

mapy (Rysunek 6.7). Pola filtrowania zostały przeniesione nad mapę. Znaczniki na ma-

pie zmieniły wygląd. Teraz taksówka zaznaczona jest poprzez żółty okrąg. Obramowanie znacznika jest zależne od stanu taksówkarza.

- Wolny - wąskie obramowanie koloru szarego,
- przypisany do kursu lub realizuje kurs - szerokie obramowanie koloru zdefiniowanego dla kursu.

Znaczniki kursu są w kształcie odwróconej łyży. Ich kolor jest im nadawany losowo z puli kolorów. Znacznik dla adresu dostarczenia jest ciemniejszym odcieniem koloru dla adresu odbioru. Obramowanie znacznika jest zależne od statusu kursu. Jeśli kurs posiada status:

- oczekuje - wąskie obramowanie koloru szarego,
- realizowany - szerokie obramowanie koloru żółtego.

Po naciśnięciu na znaczniki wyświetlone zostaną informacje. W przypadku znacznika taksówki identyfikator taksówkarza oraz kursu, do którego jest przypisany lub który realizuje. W przypadku znacznika kursu, wyświetlony zostanie identyfikator kursu, numer telefonu klienta, adres odbioru lub dostarczenia, status kursu i identyfikator taksówkarza przypisanego do kursu.

6.5 Instalacja oprogramowania

Ten podrozdział obejmuje instalację i uruchomienie systemu na systemie operacyjnym *Windows 8.1*. Aby uruchomić system należy posiadać oprogramowanie:

- *Node.js*, które można pobrać ze strony „<https://nodejs.org/en/>“,
- *Sails.js*, które zainstalować można po pobraniu i zainstalowaniu *Node.js*, poprzez uruchomienie okna poleceń i wpisanie komendy: `npm -g install sails`
- *Xampp*, które można pobrać ze strony „<https://www.apachefriends.org/pl/download.html>“,

Należy uruchomić program *Xampp* i włączyć serwer *Apache* oraz *MySQL* przyciskiem „Start“. Następnie należy uruchomić przeglądarkę internetową i wpisać adres „<http://localhost/phpmyadmin>“, aby włączyć zarządzanie serwerem baz danych. Następnie należy utworzyć bazę danych o nazwie *system_taxi*. Kolejnym krokiem jest włączenie okna poleceń w wybranym folderze i wpisanie komendy `sails new system_taxi`. Utworzony zostanie przykładowy projekt *sails* w folderze *system_taxi*. Należy skopiować foldery bez plików z głównego folderu z załączonej płyty nazwanego *kod źródłowy*, wkleić je do utworzonego folderu i zastąpić istniejące pliki. Ostatnim krokiem jest otwarcie okna poleceń w folderze *system_taxi* i wpisanie komendy `sails lift`. W ten sposób system został uruchomiony. Wystarczy w przeglądarce wpisać adres „<http://localhost:1337>“. Możemy już złożyć zamówienie na stronie i je anulować jako klienta. Jednak aby móc się zalogować należy najpierw założyć konto Pracownika Centrali. Aby to zrobić, należy wpisać w przeglądarce adres „http://localhost:1337/signup_employee“ i wypełnić formularz. Strona do rejestracji została stworzona jedynie dla osób, które chcą wypróbować system. W praktyce taka strona nie byłaby dostępna dla każdego użytkownika.

6.6 Instrukcja użytkowania

W tym podrozdziale znajdują się instrukcje dla dwóch typów użytkowników: Klienta i Pracownika Centrali.

W obecnym stanie systemu klient ma możliwość składania nowego zamówienia i anulowania go w razie potrzeby. Aby złożyć zamówienie na taksówkę należy włączyć stronę systemu i nacisnąć przycisk „Zamów taksówkę” widoczny na środku ekranu. Wyświetli się formularz do zamówienia taksówki widoczny na rysunku 6.8 oznaczony cyfrą 1. Na-

The screenshot shows the 'SystemTaxi' web application. At the top, there is a navigation bar with the text 'SystemTaxi' on the left, a login field labeled 'Pracownik' with a password mask '.....' in the center, and a green 'Zaloguj' button on the right. Below the navigation bar, there are two tabs: 'Zamow' (highlighted in orange) and 'Anuluj'. A red box labeled '1' encloses a form for placing a taxi order. The form contains the following fields: 'Numer telefonu', 'Miasto', 'Ulica', 'Numer budynku', 'Liczba pasażerów', and a dropdown menu. At the bottom of the form is an orange 'Zamow' button. A red box labeled '2' encloses the 'Zamow' and 'Anuluj' tabs.

Rysunek 6.8: Instrukcja: Złożenie zamówienia na taksówkę

leży go wypełnić i nacisnąć przycisk „Zamów”. Ostatnie pole służące do określenia typu auta jest opcjonalne. Jeśli dane zostały wypełnione prawidłowo wyświetli się kolejny formularz (rysunek 6.9), w którym podany zostanie identyfikator zamówienia. Widoczne

The screenshot shows the 'SystemTaxi' web application. At the top, there is a navigation bar with the text 'SystemTaxi' on the left, a login field labeled 'Pracownik' with a password mask '.....' in the center, and a green 'Zaloguj' button on the right. Below the navigation bar, there are two tabs: 'Zamow' (highlighted in orange) and 'Anuluj'. A red box labeled '1' encloses a form for confirming the taxi order. The form contains the following fields: a text area with the message 'Zapamiętaj identyfikator i hasło zamówienia. Potrzebne są one w przypadku chęci rezygnacji z zamówienia.', a label 'Identyfikator:', a text input field containing '21', a label 'Hasło', a text input field, and an orange 'Zatwierdz' button at the bottom.

Rysunek 6.9: Instrukcja: Potwierdzenie złożenie zamówienia na taksówkę

jest w nim również pole do wpisania hasła zamówienia, potrzebne w razie gdyby klient chciał anulować zamówienie. Po wpisaniu hasła należy potwierdzić zamówienie naciskając na przycisk „Potwierdź”. Zamówienie zostanie przekazane do realizacji. Jeśli klient chce anulować zamówienie musi nacisnąć przycisk „Anuluj” w zakładce widocznej na rysunku

6.8 oznaczonej cyfrą 2. Wyświetli się formularz (rysunek 6.10), w którym należy podać identyfikator oraz hasło zamówienia i nacisnąć przycisk „Zrezygnuj z zamówienia“. Jeśli dane będą prawidłowo wypełnione i zamówienie nie będzie w trakcie lub po realizacji to zostanie ono anulowane.

SystemTaxi

Pracownik ***** Zaloguj

Zamow Anuluj

Identyfikator:

Identyfikator

Hasło

Zrezygnuj z zamówienia

Rysunek 6.10: Instrukcja: Anulowanie zamówienia na taksówkę

W przypadku pracownika centrali instrukcja jest bardziej obszerna. Po wejściu na stronę pracownik musi zalogować się do systemu używając posiadanego loginu i hasła. Po zalogowaniu użytkownik zobaczy listę taksówkarzy będących w pracy (rysunek 6.11).

SystemTaxi

Pracownik wyloguj

W pracy Wszyscy

Nowy kurs

Kursy

Klienci

Taksowkarze

Mapa

Id	Stan	Status	Imię i nazwisko	Standard	Ile
2	przypisany do kursu	1/2	Jan Taksówkarz	sportowe	2
1	realizuje kurs	1	Adam Myszkowski	ekskluzywne	4
4	przypisany do kursu	1	Krzysztof Jankowski	ekskluzywne	4
5	przypisany do kursu	1	Monik Kaminski	ekskluzywne	4
6	przypisany do kursu	1	Lech Zawada	ekskluzywne	4
3	przypisany do kursu	1	Maciej Leniwy	standardowe	4

Rysunek 6.11: Instrukcja: Lista taksówkarzy w pracy

Po lewej stronie znajduje się główne menu. Tutaj użytkownik może przechodzić między głównymi funkcjami systemu. Aktualnie aktywna funkcja oraz funkcja po najechaniu myszką podświetlona jest kolorem pomarańczowym.

Na górze po prawej stronie widoczne jest podmenu. Jego zawartość zmienia się w zależności od wyboru głównej funkcji. Tutaj również podświetlone na pomarańczowo są opcje aktywne lub na których znajduje się wskaźnik myszki.

Jeśli zjedziemy niżej, na dole ukaże się lista z kursami aktualnie realizowanymi lub do zrealizowania (rysunek 6.12). Po naciśnięciu na jeden z kursów o statusie „oczekuje“ pod tabelą pokażą się dwa przyciski, do modyfikacji i usunięcia wybranego kursu. Jeśli pracownik naciśnie przycisk „Usuń“ wyświetli się popup zasłaniający listę kursów z pytaniem czy aby na pewno należy usunąć kurs oraz dwa przyciski: „Usuń“, „Nie usuwaj“. Potwierdzenie usunięcia jest równoznaczne z anulowaniem kursu, nie będzie on faktycznie usunięty z bazy danych. Spowoduje to zamknięcie popup'u i uruchomienie planowania kursów. Po każdym planowaniu odświeżane są wszystkie listy oraz wyświetlana zostaje lista kursów z głównego menu.

Aktualne i przyszłe kursy

	Id	Status	Klient	Taksowkarz	Czas zamowienia	Czas odbioru
<input type="radio"/>	16	oczekuje	222	6	15:34	
<input checked="" type="radio"/>	19	oczekuje	234111234		23:17	
<input type="radio"/>	20	oczekuje	222333111		08:04	
<input type="radio"/>	21	oczekuje	221112333		22:16	
<input type="radio"/>	3	oczekuje	651928102		11:00	
<input type="radio"/>	15	oczekuje	222	4	02:23	
<input type="radio"/>	18	oczekuje	672188232	5	16:17	
<input type="radio"/>	5	oczekuje	222	2	01:40	
<input type="radio"/>	14	oczekuje	222	3	02:16	
<input type="radio"/>	1	realizowany	651928102	1	11:00	11:06

[Modyfikuj](#) [Usuń](#)

Rysunek 6.12: Instrukcja: Lista aktualnych i przyszłych kursów

SystemTaxi Pracownik [wyloguj](#)

[Modyfikuj](#) [Stworz](#) [Lista](#)

[Nowy kurs](#)
[Kursy](#)
[Klienci](#)
[Taksowkarze](#)
[Mapa](#)

Kurs#19

Taksowkarz:

Klient:

Status:

Osob:

Czas zamowienie:

Modyfikuj

Rysunek 6.13: Instrukcja: Modyfikacja wybranego kursu

Jeśli po wyborze kursu pracownik wybierze modyfikację zostanie wyświetlony formularz do manualnego przypisania taksówkarza do kursu (rysunek 6.13). Widoczne są tutaj różne informacje o kursie, jednak możliwa jest zmiana jedynie taksówkarza. Możliwe jest wybranie taksówkarzy, którzy są wolni lub przypisani do kursu, ale jeszcze go nie realizują. Duży przycisk „Modyfikuj” służy do zapisania zmiany i uruchomienia automatycznego planowania kursów. Kurs modyfikowany przez pracownika nie zostanie uwzględniony przy automatycznym planowaniu kursów.

W podmenu widoczna jest opcja „Stwórz”. Służy do wyświetlenia formularza znanego z zamawiania kursu przez klienta i działa dokładnie tak samo. Jediną różnicą jest brak pola do wpisania hasła przy anulowaniu zamówienia, ponieważ pracownik może anulować je bez podania hasła.

Id	Status kurs	Tak Nie Oznacz	Klient	Osob	Taksowkarz	Data	Czas zamowienia	Czas odbioru	Czas dostarczenia	Adres Odbioru	Adres Dostarcze
21	oczekuje	false	221112333	1		2015-11-24	22:16			Wroclaw, Kosciuszki 22	.
20	oczekuje	false	222333111	1		2015-11-19	08:04			Wroclaw, Kosciuszki 1	.
19	oczekuje	false	234111234	1		2015-11-18	23:17			Wroclaw, Traugutta 21	.
18	oczekuje	false	672188232	1	5	2015-11-18	16:17			Wroclaw, Krynicka 10	.
17	oczekuje	true	901837122	1		2015-11-18	16:01			Wroclaw, Dworcowa 20	.
16	oczekuje	false	222	1	6	2015-11-7	15:34			Wroclaw, Kosciuszki 2	.

Rysunek 6.14: Instrukcja: Lista wszystkich kursów

Po wyborze opcji „Lista” w podmenu zostanie wyświetlona lista wszystkich kursów (rysunek 6.14). Jest to domyślny widok przy wyborze opcji „Kursy” z głównego menu oraz po ukończeniu planowania kursów. Listę tę można filtrować poprzez wpisanie frazy w pola nad kolumnami tabeli. Wyświetlane są kursy pasujące do każdego z filtrów. Po naciśnięciu myszką na listę podświetlany jest kurs nad którym znajduje się wskaźnik myszy. Po naciśnięciu na kurs pod tabelą wyświetli się dodatkowa informacja na temat kursu. Wybór opcji „Nowy kurs” z głównego menu skutkuje wyświetleniem tego samego widoku, który był wyświetlany przy wyborze opcji „Stwórz” z podmenu. Opcja „Klienci” służy do wyświetlenia kursów wybranego klienta, którego numer telefonu pracownik podaje w polu „Klient” (rysunek 6.15). Działanie listy jest takie samo jak w przypadku wspomnianej wcześniej listy kursów.

Wracając do listy taksówkarzy, którą można wyświetlić poprzez wybór opcji „Taksówkarze” z menu głównego. Lista zawiera wszystkich taksówkarzy. Możliwe jest jej filtrowanie. W podmenu możliwy jest wybór opcji „W pracy”, który wyświetla listę taksówkarzy w pracy, tą samą, która jest wyświetlana przy zalogowaniu się do systemu.

Ostania opcja dotyczy mapy (rysunek 6.7). Na mapie zaznaczone są taksówki oraz kursy. Znaczniki taksówek zaznaczone są poprzez żółty okrąg. Obrazowanie znacznika jest zależne od stanu taksówkarza.

- Wolny - wąskie obramowanie koloru szarego,
- przypisany do kursu lub realizuje kurs - szerokie obramowanie koloru zdefiniowanego

Klient	Id kursu	Status	Osob	Taksowkarz	Data	Czas zamowienia	Czas odbioru	Czas dostarczenia	Adres Odbioru	Adres Dostarczenia
651928102	2	realizowany	2		2015-10-16	10:20	12:06		Wroclaw, Prądzynskiego 51	Wroclaw, Kościuszki 20
651928102	4	zrealizowany	2	1	2015-10-16	10:20	12:06	13:00	Wroclaw, Prądzynskiego 51	Wroclaw, Kościuszki 20
651928102	3	oczekuje	3		2015-10-15	11:00			Wroclaw, Prądzynskiego 51	Wroclaw, Kościuszki 20
651928102	1	realizowany	4	1	2015-10-15	11:00	11:06		Wroclaw, Prądzynskiego 51	Wroclaw, Kościuszki 20

Rysunek 6.15: Instrukcja: Lista kursów wybranego klienta

dla kursu.

Znaczniki kursu są w kształcie odwróconej łzy. Ich kolor jest im nadawany losowo z puli kolorów. Znacznik dla adresu dostarczenia jest ciemniejszym odcieniem koloru dla adresu odbioru. Obrazowanie znacznika jest zależne od statusu kursu. Jeśli kurs posiada status:

- oczekuje - wąskie obramowanie koloru szarego,
- realizowany - szerokie obramowanie koloru żółtego.

Po naciśnięciu na znaczniki wyświetlone zostaną informacje. W przypadku znacznika taksówki identyfikator taksówkarza oraz kursu, do którego jest przypisany lub który realizuje. W przypadku znacznika kursu, wyświetlony zostanie identyfikator kursu, numer telefonu klienta, adres odbioru lub dostarczenia, status kursu i identyfikator taksówkarza przypisanego do kursu. Możliwe jest filtrowanie taksówkarzy i kursów poprzez wpisanie identyfikatora taksówkarza lub klienta w pola filtrowania. Skutkuje to wyświetleniem jedynie kursów i taksówkarzy powiązanych z podanym taksówkarzem lub klientem.

6.7 Archiwizacja danych

Aby móc zarządzać bazą danych musiałem uruchomić serwer bazy danych. Ja wybrałem technologię *MySQL*, dla której istnieje wiele oprogramowań, m.in. oprogramowanie *Xampp* z wbudowanym modułem *phpmyadmin*. Właśnie to oprogramowanie zostało użyte do archiwizacji danych w systemie.

Aby uruchomić serwer bazy danych, należy zainstalować oprogramowanie *Xampp*, o czym wspominałem już w podrozdziale 6.5. Po uruchomieniu programu należy włączyć serwer *Apache* oraz *MySQL* przyciskiem „Start”. Następnie należy uruchomić przeglądarkę internetową i wpisać adres „http://localhost/phpmyadmin”. Wyświetlona zostanie strona, gdzie po lewej stronie widoczna jest lista baz danych utworzonych na serwerze. Aby utworzyć nową bazę danych należy nacisnąć na link „Nowa” na samej górze listy. Spowoduje to wyświetlenie formularza do wpisania nazwy bazy danych oraz wyboru metody porównywania napisów, dla potrzeb systemu wybrałem format „utf8_polish_ci”. Po naciśnięciu

„Utwórz“ baza danych zostanie utworzona na serwerze.

Możemy teraz wybrać bazę z listy po lewej stronie. Wyświetli się lista tabel utworzonych w tej bazie, jeśli baza nie zawiera żadnych tabel, zostanie wyświetlony odpowiedni komunikat. Pod tą listą znajduje się formularz do utworzenia nowej tabeli, gdzie podaje się jedynie nazwę i liczbę kolumn, liczbę tą można zmienić również później. Utworzenie tabeli następuje po naciśnięciu przycisku „Wykonaj“. W wyświetlonym widoku tabeli należy wypełnić informacje o kolumnach, jakie tabela ma posiadać. Możliwe jest również ustawienie wielu innych własności tabeli, np. domyślnego sortowania. Po wypełnieniu i zapisaniu możliwe jest dodawanie wierszy poprzez wybór opcji „Wstaw“ z menu znajdującego się na górze.

Ważną funkcją w *phpmyadmin* jest możliwość importowania i eksportowania baz danych, tabel czy wierszy. Aby wyeksportować bazę danych należy wybrać bazę z listy po lewej stronie, następnie wybrać opcję „Eksport“ z górnego menu i określić opcje eksportu. Jeśli chcemy zapisać całą bazę danych, należy zaznaczyć opcję „Szybko“, możemy też dostosować eksport do naszych wymagań, poprzez wybór opcji „Dostosuj“. Format zapisu bazy danych możliwy jest w polu „Format“. Domyślnie wybrana jest opcja „SQL“. Po naciśnięciu „Wykonaj“ zostanie zapisany plik w wybranym formacie na dysku komputera.

Importowanie bazy danych jest równie proste. Należy najpierw wyjść z bazy danych i wejść bezpośrednio na serwer naciskając na górze w link „Serwer: 127.0.0.1“. Następnie należy wybrać opcję „Import“ w górnym menu. W wyświetlonym widoku wybieramy plik z dysku komputera poprzez przycisk „Wybierz plik“. Po wyborze określamy kodowanie bazy danych, domyślnie „utf-8“ oraz format w jakim plik został zapisany. Na koniec naciskamy „Wykonaj“ i czekamy na zaimportowanie bazy danych na serwer. Narzędzie jest bardzo proste w użyciu, po zdobyciu niewielkiego doświadczenia.

6.8 Podsumowanie

Rozdział ten powinien wyjaśnić zasady działania systemu oraz sposób uruchomienia go. Instrukcje użytkownika w sposób dokładny opisują poruszanie się między funkcjami systemu i użytkownik nie powinien mieć problemu z użyciem ich. Faza implementacji wprowadziła kilka zmian w systemie względem fazy projektowania. Wynika to z spojrzenia na system z perspektywy czasu oraz z faktu niemożliwości przewidzenia wszystkich problemów, jakie mogą się pojawić w systemie. Oczywiście system mógłby i powinien być rozbudowany o wiele funkcji, o których wspomnę w późniejszym rozdziale. Jednak czas i liczba osób zaangażowanych w projekt uniemożliwiają wprowadzenia ich do systemu.

Rozdział 7

Testy

7.1	Wprowadzenie	51
7.2	Testy poprawności działania	51
7.3	Testy wydajnościowe	52
7.4	Testy bezpieczeństwa	52
7.5	Podsumowanie	52

7.1 Wprowadzenie

Ważną częścią tworzenia oprogramowania jest przetestowanie go, sprawdzenie czy wszystkie funkcje działają prawidłowo i czy jest ono wystarczająco bezpieczne, aby używać je w środowisku dla którego zostało stworzone. Ze względu na ważności informacji przetwarzanych i generowanych przez system, działanie jego powinno być prawidłowe w każdym wypadku. Ponieważ działanie całej firmy jest oparte na tym właśnie systemie. Testy jakie zamierzam przeprowadzić to testy poprawności działania testy wydajnościowe oraz testy bezpieczeństwa.

7.2 Testy poprawności działania

System został przetestowany na wszystkich najważniejszych przeglądarkach internetowych w systemie operacyjnym *Windows 8.1*:

- *Internet Explorer 11*,
- *Google Chrome 46*,
- *Firefox 42*,
- *Opera 33*,
- *Safari 5.1*.

Na każdej przeglądarce system działał prawidłowo. Dodatkowo każda z opisanych funkcji w podrozdziale 6.6 Instrukcja użytkownika została sprawdzona i każda z nich działała prawidłowo.

7.3 Testy wydajnościowe

Testy wydajnościowe zostały zrealizowany przy pomocy narzędzia *Apache JMeter*. Przetestowana zostanie szybkość odpowiedzi serwera ze względu na ilość wywołań strony. Parametry serwera są następujące:

- Procesor Intel Core i7 (2.00GHz - 2.90GHz)
- Pamięć RAM 6GB.

Wyniki testów: 1 użytkownik - 18ms 20 użytkowników - 34ms 50 użytkowników - 206ms 100 użytkowników - 444ms 200 użytkowników - 906ms Wynika z tego, że na stronie może przebywać około 100 użytkowników jednocześnie, wtedy korzystanie z niej będzie nadal wygodne i szybkie. Niestety przy 200 użytkownikach załadowanie strony wynosi już niemal sekundę, a więc zbliża się do granicy wygodnego użytkowania.

7.4 Testy bezpieczeństwa

Do testów bezpieczeństwa wykorzystałem narzędzie *Acunetix Web Vulnerability Scanner 10.0*. Program wykazał średni poziom bezpieczeństwa systemu. Powodem takiego wyniku jest podatność systemu na atak *clickjacking*, polegający na oszukaniu użytkownika sieci podkładając coś innego do kliknięcia nad rzecz na którą użytkownik faktycznie chce nacisnąć, w ten sposób wyłudzać inne działanie niż oczekuje użytkownik. Dodatkowo w formularzu logowania pole hasła jest uzupełniane automatycznie, co zostało uznane za niebezpieczne, ponieważ osoba postronna może zalogować się na zapamiętane przez przeglądarkę konto. Testy wykazały, że system może zostać użyty, jednak należałoby zadbać o lepszą jego ochronę.

7.5 Podsumowanie

Z przeprowadzonych testów wynika, że system działa poprawnie. Jest wydajny dla mniejszych firm taksówkowych, gdzie liczba klientów w jednym czasie nie przekracza 100 osób. Jednak ukazały również słabszą stronę, czyli mechanizmy bezpieczeństwa systemu, które można ulepszyć.

Rozdział 8

Podsumowanie pracy

8.1	Wykonane prace	53
8.2	Realizacja celu pracy	53
8.3	Wnioski	54
8.4	Sugestie odnośnie dalszych prac	54

8.1 Wykonane prace

Podczas realizacji pracy wykonałem następujące prace:

- zapoznałem się z istniejącymi rozwiązaniami dostępnymi na rynku,
- zaprojektowałem:
 - logikę działania systemu,
 - modele potrzebne do stworzenia systemu,
 - algorytm do planowania kursów,
 - interfejs aplikacji internetowej,
- stworzyłem prototyp systemu oparty o projekty,
- wykonałem potrzebne testy systemu,
- opisałem cały proces tworzenia systemu.

8.2 Realizacja celu pracy

Uważam, że praca jaką wykonałem jest w pełni zgodna z początkowymi założeniami, a stworzony system działa poprawnie, więc i cel pracy został zrealizowany. Praca ta jest dobrym początkiem do stworzenia w pełni sprawnego systemu do zarządzania firmą tak-sówkową.

8.3 Wnioski

Stworzenie systemu do planowania przewozów dla taksówek jest skomplikowanym projektem. Mnogość parametrów jakie należy uwzględnić nie tylko przy planowaniu kursów, ale również przy komunikacji między centralą, a taksówkarzem, przy priorytetowaniu procesów w systemie, tworzeniu wygodnego w użytkowaniu interfejsu, sprawia że jest to zadanie wymagające sporej ilości czasu na poprawne zaprojektowanie systemu. Potwierdza to moją tezę, że niemożliwe jest optymalne zaplanowanie przewozów dla taksówek i zarządzanie nimi wraz z taksówkarzami jedynie przez człowieka, potrzebna jest pomoc komputera z odpowiednim systemem. System, który stworzyłem, spełnia swoją rolę, czyli ułatwia pracę pracownika centrali w firmie taksówkowej. Jednak jego funkcje mogą zostać rozbudowane, tak aby praca ta stała się jeszcze łatwiejsza i dająca lepsze wyniki firmie.

8.4 Sugestie odnośnie dalszych prac

Jak już wcześniej wspominałem, system może być rozwinięty i ulepszony o nowe funkcje. Jedną z nich jest moduł do komunikowania się taksówkarza z systemem. Moduł ten umożliwiałby zalogowanie się taksówkarza do systemu, np. z urządzenia mobilnego. Dzięki temu taksówkarz automatycznie otrzymywałby informację o kursach, które zostały przypisane jemu do realizacji oraz posiadałby możliwość podglądu historii realizowanych przez niego kursów. Ponad to miałyby wbudowaną funkcję podglądu trasy na mapie do pobranej z systemu lokalizacji, w której miałby się stawić taksówkarz.

Moduł mapy w obecnym systemie również mógłby zostać rozbudowany, aby widoczne były nie tylko znaczniki, ale również trasy jakie pokonali taksówkarze, które byłyby archiwizowane w bazie danych. W ten sposób historia kursów była by jeszcze bardziej dokładna i mogła by być przeanalizowana w przyszłości pod kontem optymalności tras, jakimi jeżdżą taksówkarze.

Ważną funkcją dla klientów byłaby możliwość zamówienia taksówki na określoną godzinę, czego nie uwzględnia obecny system.

Aby system stał się kompletnym systemem do zarządzania firmą taksówkową należałoby stworzyć moduł dotyczące płatności i księgowości, aby nie był wymagany oddzielny system.

Bibliografia

- [1] Wydział Informatyki i Zarządzania Politechniki Wrocławskiej, *PROCEDURA DY-
PLOMOWANIA STUDENTÓW I i II ...*, [http://www.wiz.pwr.wroc.pl/477271,
1.dhtml](http://www.wiz.pwr.wroc.pl/477271,1.dhtml), dostęp 22.03.2015.
- [2] *LaTeX documentation*, <http://latex-project.org/guides/>, dostęp 22.03.2015.
- [3] Autocab Ghost features, <http://www.autocab.com/en-au/ghost.html>, dostęp 18.04.2015.
- [4] esysTaxi, <http://system-taxi.pl/>, dostęp 18.04.2015.
- [5] taxi5i, <http://www.taxi5.pl/>, dostęp 18.04.2015.
- [6] heyTaxi, <http://www.heytaxi.pl/start>, dostęp 18.04.2015.
- [7] *MySQL documentation*, <https://www.mysql.com/>, dostęp 18.04.2015.
- [8] *javascript documentation*, [https://developer.mozilla.org/en-US/docs/Web/
JavaScript](https://developer.mozilla.org/en-US/docs/Web/JavaScript), dostęp 18.04.2015.
- [9] Halina Kwaśnicka, 2002, *Sztuczna Inteligencja Nr 1, Algorytmy Ewolucyjne - Przy-
kłady Zastosowań*, Wrocław: Oficyna Wydawnicza Politechniki Wrocławskiej
- [10] *html5 documentation*, <http://www.w3.org/TR/html5/>, dostęp 18.04.2015.
- [11] *css documentation*, <http://www.w3.org/Style/CSS/>, dostęp 18.04.2015.
- [12] *jQuery documentation*, <https://jquery.com/>, dostęp 18.04.2015.
- [13] *Node.js*, <https://nodejs.org/>, dostęp 06.07.2015.
- [14] *AngularJS-Wprowadzenie* MRzepinski, [http://mrzepinski.pl/
angularjs-1-wprowadzenie.html](http://mrzepinski.pl/angularjs-1-wprowadzenie.html), dostęp 20.09.2015.
- [15] *Atom*, <https://atom.io/>, dostęp 01.09.2015.
- [16] *JMeter*, <http://jmeter.apache.org/>, dostęp 23.11.2015.
- [17] *Acunetix Web Vulnerability Scanner*, <http://www.acunetix.com/>, dostęp 26.11.2015.

Oświadczenia

POLITECHNIKA WROCŁAWSKA
WYDZIAŁ INFORMATYKI
I ZARZĄDZANIA
(nazwa jednostki organizacyjnej)

Mateusz Adamczyk
(imię i nazwisko)

Komputerowy system planowania przewozów dla taksówek
(tytuł pracy)

O Ś W I A D C Z E N I E

~~Wyrażam zgodę~~ (nie wyrażam zgody)* na udostępnienie mojej pracy dyplomowej,
~~doktorskiej*~~

.....
(podpis)

Wrocław, dnia

*) niepotrzebne skreślić



Politechnika
Wrocławska

załącznik do PO 33/2010

Mateusz Adamczyk

Imię i nazwisko studenta

203246

Numer albumu

93111100590

PESEL

Informatyka i Zarządzanie, Informatyka

Wydział, kierunek studiów

**Oświadczenie o samodzielności
wykonanej pracy / ~~części pracy dyplomowej~~*
~~licencjackiej~~ / inżynierskiej / ~~magisterskiej~~***

Ja, niżej podpisany Mateusz Adamczyk

student Wydziału Informatyki i Zarządzania Politechniki Wrocławskiej

oświadczam, że przedkładaną pracę dyplomową ~~licencjacką~~ / inżynierską / ~~magisterską~~ * /

~~część pracy dyplomowej licencjackiej / inżynierskiej / magisterskiej~~ *

pt.: Komputerowy system planowania przewozów dla taksówek

wykonałem samodzielnie, tzn. nie zlecałem opracowania pracy dyplomowej, ani jej części osobom trzecim, jak również nie kopiowałem i nie korzystałem z części lub całości pracy innych osób.

Jednocześnie przyjmuję do wiadomości, że w przypadku stwierdzenia popełnienia przeze mnie czynu polegającego na przypisaniu sobie autorstwa istotnego fragmentu lub innych elementów cudzej pracy lub ustalenia naukowego, Rada Wydziału stwierdzi nieważność postępowania w sprawie nadania mi tytułu zawodowego (art. 193 ustawy z dnia 27 lipca 2005 r. – Prawo o szkolnictwie wyższym, Dz. U. Nr 164 poz. 1365, z późniejszymi zmianami).

.....
Data

.....
Podpis studenta

* niepotrzebne skreślić

Wrocław, dnia

Mateusz Adamczyk
Imię i nazwisko studenta

2023246 93111100590
Numer albumu PESEL

Informatyka i Zarządzanie, Informatyka
Wydział, kierunek studiów

OŚWIADCZENIE

Oświadczam, że załączony (archiwalny) drukowany egzemplarz pracy dyplomowej/projektu inżynierskiego jest tożsamy z wersją załączoną na elektronicznym nośniku danych.

.....
podpis czytelny oświadczającego

CD/DVD-ROM

