POLITECHNIKA KOSZALIŃSKA

WYDZIAŁ ELEKTRONIKI I INFORMATYKI

STUDIA STACJONARNE, SEMESTR VI, KIERUNEK INFORMATYKA

ROK AKADEMICKI 2015 / 2016

**DOKUMENTACJA Z PRZEDMIOTU „PROJEKT ZESPOŁOWY CASE”**

**TEMAT PROJEKTU**

**APLIKACJA BAZODANOWA DO OBSŁUGI FIRMY TRANSPORTOWEJ**

http://b10.com.pl/images/main-track.png

**wykonawcy**

Damian Berda

Jakub Wachowiak

Łukasz Stolarczyk

Marcin Chruścicki

Michał Motykowski

Spis Treści

[1. Protokół z zebrania założycielskiego zespołu programistycznego „aplikacja bazodanowa do obsługi firmy transportowej” zwanego dalej zespołem, odbytego w dniu 01.03.2016 r. w Koszalinie. 5](#_Toc452312094)

[2. Instrumentarium zespołu projektowego. 6](#_Toc452312095)

[3. Opis wymagań klienta. 6](#_Toc452312096)

[3.1. Model klienta. 6](#_Toc452312097)

[3.2. Wymaganie funkcjonalne. 6](#_Toc452312098)

[3.3. Wymagania niefunkcjonalne (ograniczenia). 6](#_Toc452312099)

[4. Wstępny harmonogram prac. 7](#_Toc452312100)

[5. Specyfikacja wymagań. 7](#_Toc452312101)

[5.1. Wstęp. 7](#_Toc452312102)

[5.1.1. Cele. 7](#_Toc452312103)

[5.1.2. Definicje, skróty. Opis. 7](#_Toc452312104)

[5.2. Ogólny opis. 8](#_Toc452312105)

[5.2.1. Przydatność projektowanej aplikacji. Walory użytkowe. 8](#_Toc452312106)

[5.2.2. Możliwości projektowanego oprogramowania. 8](#_Toc452312107)

[5.2.3. Ogólne ograniczenia. 8](#_Toc452312108)

[5.2.4. Charakterystyka użytkowników. 8](#_Toc452312109)

[5.2.5. Środowisko operacyjne. 8](#_Toc452312110)

[5.3. Specyfika wymagań. 8](#_Toc452312111)

[5.3.1. Wymagania funkcjonalne. 8](#_Toc452312112)

[5.3.2. Wymagania niefunkcjonalne (ograniczenia). 9](#_Toc452312113)

[5.4. Diagram przypadków użycia. 9](#_Toc452312114)

[6. Raport dla zleceniodawcy. 9](#_Toc452312115)

[6.1. Cele przedsięwzięcia. 9](#_Toc452312116)

[6.2. Zakres przedsięwzięcia. 9](#_Toc452312117)

[6.3. Systemy zewnętrzne. 9](#_Toc452312118)

[6.4. Szkicowy opis wymagań. 10](#_Toc452312119)

[6.4.1. Wymagania funkcjonalne. 10](#_Toc452312120)

[6.4.2. Wymagania niefunkcjonalne (ograniczenia). 10](#_Toc452312121)

[6.5. Szkicowy opis rozwiązań. 10](#_Toc452312122)

[6.6. Szkicowy opis modelu systemu. 10](#_Toc452312123)

[6.7. Kosztorys projektu. 10](#_Toc452312124)

[6.8. Wstępny harmonogram prac. 11](#_Toc452312125)

[7. Harmonogram prac. 13](#_Toc452312126)

[8. Wizytówka. 14](#_Toc452312127)

[9. Logo. 14](#_Toc452312128)

[10. Model konceptualny bazy danych. 15](#_Toc452312129)

[11. Model fizyczny bazy danych. 15](#_Toc452312130)

[12. Diagram klas. 16](#_Toc452312131)

[13. Diagram czynności. 18](#_Toc452312132)

[14. Diagramy sekwencji. 19](#_Toc452312133)

[15. Metodologia. Narzędzia CASE. 21](#_Toc452312134)

[16. Kosztorys. 22](#_Toc452312135)

[17. Narzędzia RAD. 23](#_Toc452312136)

[18. Testy przeprowadzone w ramach projektu. 23](#_Toc452312137)

[1. Streszczenie. 27](#_Toc452312138)

[2. Wprowadzenie. 28](#_Toc452312139)

[2.1. Cel. 28](#_Toc452312140)

[2.2. Zakres. 28](#_Toc452312141)

[2.3. Definicje, akronimy, skróty. 28](#_Toc452312142)

[2.4. Odsyłacze. 28](#_Toc452312143)

[2.5. Opracowanie. 28](#_Toc452312144)

[3. Projekt. Standardy, konwencje, procedury. 28](#_Toc452312145)

[3.1. Standardy projektowe. 28](#_Toc452312146)

[3.2. Standardy dokumentacyjne. 28](#_Toc452312147)

[3.3. Konwencje nazewnicze. 29](#_Toc452312148)

[3.4. Standardy programistyczne. 29](#_Toc452312149)

[3.5. Narzędzia. 29](#_Toc452312150)

[4. Specyfikacja komponentów. 29](#_Toc452312151)

[4.1. Klasa “Autokary”. 29](#_Toc452312152)

[4.2. Klasa “Kierowcy”. 30](#_Toc452312153)

[4.3. Klasa “Kursy”. 30](#_Toc452312154)

[4.4. Klasa “Miejscowosci”. 30](#_Toc452312155)

[4.5. Klasa “Przystanki”. 30](#_Toc452312156)

[4.6. Klasa “Rejestr\_przejazdow”. 30](#_Toc452312157)

[4.7. Klasa “Rozklad\_Jazdy”. 30](#_Toc452312158)

[4.8. Klasa “Trasa”. 30](#_Toc452312159)

[4.9. Klasa “checkbox”. 30](#_Toc452312160)

[4.10. Klasa “Loguj”. 30](#_Toc452312161)

[4.11. Klasa “Main”. 30](#_Toc452312162)

[4.12. Klasa “ConnectionToDB”. 30](#_Toc452312163)

[4.13. Klasa ”FillingTables”. 31](#_Toc452312164)

[4.15. Biblioteka “sqljdbc4”. 31](#_Toc452312165)

[5. Załączniki. 31](#_Toc452312166)

[5.1. Harmonogram prac. 31](#_Toc452312167)

[5.2. Interfejs aplikacji. 32](#_Toc452312168)

[5.2.1. Okno powitalne. 32](#_Toc452312169)

[5.2.2. Logowanie do aplikacji. 32](#_Toc452312170)

[5.2.3. Aplikacja. 32](#_Toc452312171)

[5.2.4. Rozkład jazdy. 33](#_Toc452312172)

[5.2.5. Okno dialogowe. 34](#_Toc452312173)

# Protokół z zebrania założycielskiego zespołu programistycznego „aplikacja bazodanowa do obsługi firmy transportowej” zwanego dalej zespołem, odbytego w dniu 01.03.2016 r. w Koszalinie.

W zebraniu wzięły udział następujące osoby:

1. Damian Berda.
2. Jakub Wachowiak.
3. Łukasz Stolarczyk.
4. Marcin Chruścicki.
5. Michał Motykowski.

Uczestnicy zebrania podjęli indywidualną decyzję o przystąpieniu do zespołu programistycznego „aplikacja bazodanowa do obsługi firmy transportowej” zwanego dalej zespołem. Po zapisaniu obecności wszystkich członków oraz wypełnieniu deklaracji członkowskich zespołu przystąpiono do wyboru kierownika zespołu. Podczas głosowania jednogłośnie wybrany został Marcin Chruścicki. Zebrani upoważnili nowo wybranego przedstawiciela do reprezentowania interesów zespołu. Po wybraniu przedstawiciela przystąpiono do ustalenia postanowień ogólnych.

Postanowienia ogólne:

1. Odwołanie przewodniczącego może odbyć się po zgłoszeniu tzw. wotum nieufności względem wyżej wymienionej osoby oraz zatwierdzenie usunięcia poprzez demokratyczne głosowanie.
2. Każdy członek zespołu zobowiązany jest do wykonywania zadań projektowych określonych w harmonogramie zadań.
3. Każdy członek zespołu zobowiązany jest do przestrzegania postanowień ogólnych.
4. Data zakończenia projektu została wstępnie ustalona na dzień 7 VI 2016 r.

Następnie przystąpiono do określenia celów projektu, jak również obowiązków członków zespołu.

Cele projektu:

1. Zaliczenie przedmiotu „Projekt Zespołowy CASE”.
2. Wykonanie projektu oprogramowania, które posłuży zarówno dyżurnemu ruchu PKS, jak i pasażerom.
3. Nauka pracy w zespole projektowym.
4. Rozwiązanie zespołu po pomyślnym zaliczeniu projektu. Przewidywany termin to 7 czerwca 2016 r.

Obowiązki członków zespołu:

1. Damian Berda (grafik).
2. Jakub Wachowiak (analityk, projektant).
3. Łukasz Stolarczyk (dokumentalista).
4. Marcin Chruścicki (kierownik, programista).
5. Michał Motykowski (tester).

# Instrumentarium zespołu projektowego.

W ramach pracy zespołowej zostało stworzone repozytorium projektu o nazwie „DamberPol” na serwerach GitHub.com i Trello.com. Do repozytorium dodawane będą kolejne wersje aplikacji oraz dokumentacji, z możliwością przeglądania przez wszystkich członków zespołu projektowego.

Żeby wspomóc i usprawnić wspólną pracę nad projektem, członkowie zespołu będą kontaktować się za pomocą takich aplikacji, jak Skype i Facebook Messenger.

Narzędzia wykorzystywane podczas pracy projektowej:

* ArgoUML – producent University of California, licencja bezpłatna. (<http://argouml.tigris.org>)
* Eclipse – producent Eclipse Foundation, licencja Eclipse Public License. ([http://www.eclipse.org](http://www.eclipse.org/))
* Microsoft Office Excel – producent Microsoft, licencja komercyjna. (<http://www.office.microsoft.com>)
* Microsoft Office Word – producent Microsoft, licencja komercyjna. (<http://www.office.microsoft.com>)
* Microsoft SQL Server – producent Microsoft, licencja Microsoft EULA. ([http://www.microsoft.com/sql](http://www.microsoft.com/sql/))
* Microsoft SQL Server Management Studio Express – producent Microsoft, licencja Microsoft EULA. (<https://www.microsoft.com/en-us/server-cloud/products/sql-server>)
* Mozilla Firefox – producent Mozilla Corporation, licencja Mozilla Licensing. (<http://www.mozilla.org>)

# Opis wymagań klienta.

# Model klienta.

Klientem jest Państwowa Komunikacja Samochodowa w Koszalinie. Działalność tej instytucji polega na gromadzeniu pewnego rodzaju danych i udostępnianiu jej części pasażerom za pośrednictwem lokalnej aplikacji.

# Wymaganie funkcjonalne.

* Sprawdzanie połączeń przez pasażerów.
* Sporządzanie raportów (np. tygodniowych, miesięcznych) na temat ilości pasażerów korzystających z danego kursu.
* Zarządzanie (np. dodawanie, usuwanie) listą zatrudnionych kierowców i wykorzystywanych autokarów.
* Administrowanie danymi – wprowadzanie / usuwanie podstawowych obiektów bazy danych.

# Wymagania niefunkcjonalne (ograniczenia).

* Dla każdego kierunku (np. Koszalin – Gdańsk) istnieją kursy wyznaczające realizację danego połączenia (np. data, czas przyjazdu / odjazdu).
* Na danej trasie (np. Koszalin – Gdańsk) istnieją kursy pośrednie (np. Słupsk – Gdańsk).
* Pasażerowie mają możliwość sprawdzenia połączenia poprzez podanie miejscowości początkowej i końcowej oraz dnia, w którym zamierzają podróżować.
* Dyżurny powinien mieć możliwość sporządzenia odpowiednich raportów.

# Wstępny harmonogram prac.

luty:

* Założenie grupy projektowej.
* Uzgodnienie wstępnych założeń projektu.

marzec:

* Określenie wymagań funkcjonalnych projektu.
* Tworzenie dokumentacji (diagram klas, przypadków użycia, funkcjonalności).

kwiecień:

* Zaprojektowanie graficznego interfejsu użytkownika.
* Programowanie kolejnych etapów projektu.
* Konsultacje z klientem (prowadzącym zajęcia).

maj:

* Dokończenie prac nad projektem.
* Testowanie i naprawa ewentualnych błędów w projekcie.
* Finalizacja prac związanych z projektem.

czerwiec:

* Zaprezentowanie aplikacji.
* Zakończenie prac nad projektem.

# Specyfikacja wymagań.

# Wstęp.

Celem dokumentu jest specyfikacja obsługi bazodanowej aplikacji w firmie transportowej.

# Cele.

* Zaliczenie przedmiotu „projekt zespołowy CASE”.
* Wykonanie projektu aplikacji wspomagającej funkcjonowanie Państwowej Komunikacji Samochodowej.
* Nauka pracy w zespole projektowym.
* Rozwiązanie zespołu po pomyślnym zakończeniu projektu.
* Usprawnienie działania firmy.

# Definicje, skróty. Opis.

* Dodawanie kierowców / autokarów – dodawanie zatrudnionych kierowców i wykorzystywanych autokarów w firmie.
* Dyżurny ruchu – wykwalifikowany pracownik, którego obowiązkiem jest prowadzenie ruchu.
* Pasażer – osoba podróżująca pojazdem.
* PKS – Państwowa Komunikacja Samochodowa, firma, dla której opracowywany jest projekt.
* Połączenie – łączność między miastami utrzymywana za pomocą środka lokomocji.
* Sporządzanie raportów – jaki kierowca jest w trasie, jakim jedzie kursem, jaki autokar prowadzi.
* Usuwanie kierowców / autokarów – usuwanie zwolnionych kierowców i niewykorzystywanych autokarów w firmie.

# Ogólny opis.

# Przydatność projektowanej aplikacji. Walory użytkowe.

Aplikacja będzie zaprojektowana w taki sposób, aby użytkownik z podstawowymi umiejętnościami obsługi komputera, nie miał jakichkolwiek problemów z obsługą rozkładu jazdy, a dyżurny z nadzorem ruchu.

# Możliwości projektowanego oprogramowania.

Aplikacja będzie oferowała:

* Dodawanie autokarów.
* Dodawanie kierowców.
* Dodawanie cyklicznych raportów nt. ilości pasażerów korzystających z danego kursu.
* Sprawdzanie rozkładu jazdy przez pasażerów.
* Usuwanie autokarów.
* Usuwanie kierowców.

# Ogólne ograniczenia.

Brak ograniczeń.

# Charakterystyka użytkowników.

* Dyżurny ruchu – osoba zarządzająca aplikacją, posiada największe uprawnienia.
* Pasażer – osoba posiadająca ograniczone przywileje, posiada możliwość sprawdzenia rozkładu jazdy autobusów.

# Środowisko operacyjne.

Każde środowisko obsługujące wirtualny silnik Java.

# Specyfika wymagań.

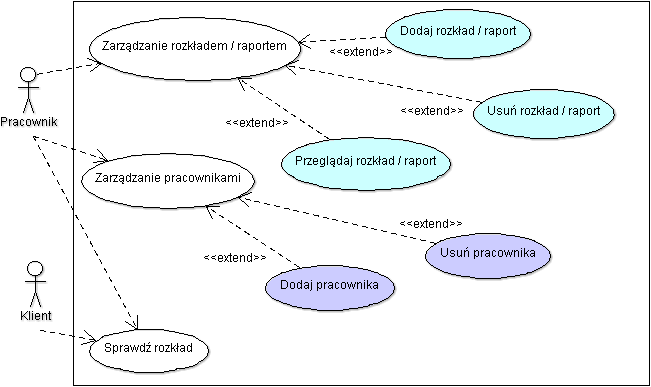
# Wymagania funkcjonalne.

* Dodawanie i usuwanie autokarów z bazy danych.
* Dodawanie i usuwanie kierowców z bazy danych.
* Dodawanie i usuwanie z bazy danych informacji o kursie.
* Logowanie do bazy.
* Wyświetlanie rozkładu jazdy.

# Wymagania niefunkcjonalne (ograniczenia).

* Pasażer nie ma uprawnień do modyfikacji rozkładu.
* Użytkownik korzysta ze środowiska obsługującego wirtualny silnik Java.

# Diagram przypadków użycia.



# Raport dla zleceniodawcy.

# Cele przedsięwzięcia.

Celem projektu jest stworzenie aplikacji wspomagającej działanie firmy transportowej. Program ma na celu ułatwić pracownikom zarządzanie danymi, a czytelnikom zapewnić możliwość przeglądania wybranych zasobów, rozkładu jazdy.

# Zakres przedsięwzięcia.

* Określenie wymagań ze strony klienta.
* Analiza i projektowanie systemu.
* Realizacja projektu.
* Tworzenie dokumentacji.
* Testowanie.
* Wdrożenie aplikacji.

# Systemy zewnętrzne.

Aplikacja będzie zgodna z systemem operacyjnym Windows (XP, Vista, Windows 7, Windows 8, Windows 10).

Oprogramowanie będzie współpracować z systemem bazodanowym MS SQL.

# Szkicowy opis wymagań.

# Wymagania funkcjonalne.

* Zarządzanie rozkładem jazdy.
* Zarządzanie kontami użytkowników.
* Zdalne przeglądanie zasobów rozkładu jazdy.

# Wymagania niefunkcjonalne (ograniczenia).

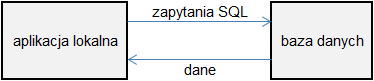
* Weryfikacja użytkownika.

# Szkicowy opis rozwiązań.

Realizacja aplikacji lokalnej zostanie wykonana w technologii Java. Do przygotowania bazy danych wykorzystany zostanie Microsoft SQL Server oraz Microsoft SQL Server Management Studio.

# Szkicowy opis modelu systemu.

System będzie się składał z aplikacji lokalnej. Aplikacja będzie się łączyć z bazą danych Microsoft Server.



# Kosztorys projektu.



# Wstępny harmonogram prac.

Zadanie 1 (29.02 – 01.03). Uzgodnienie tematu projektu, wstępna umowa z klientem. Utworzenie grupy projektowej, ustalanie zasad współpracy z członkami grupy. Utworzenie protokołu z zebrania założycielskiego.

Zadanie 2 (08.03 – 15.03). Utworzenie harmonogramu prac. Opracowanie dokumentacji wymagań użytkownika. Raport klienta, opracowanie raportu dla zleceniodawcy (zakres przedsięwzięcia; przygotowanie szkicowego opisu wymagań, modelu systemu, proponowanych rozwiązań, oszacowanie kosztów projektu oraz daty zakończenia prac nad projektem).

Zadanie 3 (15.03 – 22.03). Określenie wymagań funkcjonalnych projektu, opracowanie specyfikacji wymagań (opracowanie słownika pojęć, przygotowanie listy wymagań funkcjonalnych i niefunkcjonalnych, zidentyfikowanie aktorów oraz ich funkcjonalności, przeprowadzenie identyfikacji przypadków użycia, opracowanie diagramu przypadków użycia). Opracowanie stylu firmowego projektu (logo, wizytówka). Zaprojektowanie modelu bazy danych (model konceptualny i model fizyczny), stworzenie bazy danych na potrzeby projektu.

Zadanie 4 (22.03 – 29.03). Wybór odpowiednich narzędzi CASE. Opracowanie zestawu diagramów (przypadków użycia, klas, przepływu danych).

Zadanie 5 (29.03 – 01.04). Wykończenie prac nad stylem firmowym projektu (logo, wizytówka).

Zadanie 6 (01.04 – 06.04). Zaprojektowanie graficznego interfejsu użytkownika. Opracowanie słownika danych zawierający specyfikację modelu.

Zadanie 7 (12.04 – 19.04). Opracowanie diagramu związków encji, diagramu przejść stanów, diagramu aktywności, diagramu sekwencji, diagramu współpracy, diagramu komponentów. Analiza kosztorysu. Konsultacje z klientem.

Zadanie 8 (19.04 – 26.04). Wybór odpowiednich narzędzi do szybkiego rozwijania aplikacji (ang. RAD). Ciąg dalszy prac nad projektem (zaprogramowanie GUI, podstawowych funkcjonalności).

Zadanie 9 (26.04 – 03.05). Prace nad programowaniem dalszych funkcjonalności systemowych.

Zadanie 10 (03.05 – 13.05). Wstępne testowanie aplikacji. Naprawa ewentualnych błędów.

Zadanie 11 (17.05 – 27.05). Zakończenie programowania projektu. Przeprowadzenie szczegółowych testów. Finalizacja projektu. Sprawdzenie i zamknięcie dokumentacji. Przygotowanie projektu do zaprezentowania klientowi.

Zadanie 12 (31.05 – 03.06). Prezentacja projektu klientowi. Zakończenie prac nad projektem.

# Harmonogram prac.

Zadanie 1 (29.02 – 01.03). Uzgodnienie tematu projektu, wstępna umowa z klientem. Utworzenie grupy projektowej, ustalanie zasad współpracy z członkami grupy. Utworzenie protokołu z zebrania założycielskiego.

Zadanie 2 (08.03 – 15.03). Utworzenie harmonogramu prac. Opracowanie dokumentacji wymagań użytkownika. Raport klienta, opracowanie raportu dla zleceniodawcy (zakres przedsięwzięcia; przygotowanie szkicowego opisu wymagań, modelu systemu, proponowanych rozwiązań, oszacowanie kosztów projektu oraz daty zakończenia prac nad projektem).

Zadanie 3 (15.03 – 22.03). Określenie wymagań funkcjonalnych projektu, opracowanie specyfikacji wymagań (opracowanie słownika pojęć, przygotowanie listy wymagań funkcjonalnych i niefunkcjonalnych, zidentyfikowanie aktorów oraz ich funkcjonalności, przeprowadzenie identyfikacji przypadków użycia, opracowanie diagramu przypadków użycia). Opracowanie stylu firmowego projektu (logo, wizytówka). Zaprojektowanie modelu bazy danych (model konceptualny i model fizyczny), stworzenie bazy danych na potrzeby projektu.

Zadanie 4 (22.03 – 29.03). Wybór odpowiednich narzędzi CASE. Opracowanie zestawu diagramów (przypadków użycia, klas, przepływu danych).

Zadanie 5 (29.03 – 01.04). Wykończenie prac nad stylem firmowym projektu (logo, wizytówka).

Zadanie 6 (01.04 – 06.04). Zaprojektowanie graficznego interfejsu użytkownika. Opracowanie słownika danych zawierający specyfikację modelu.

Zadanie 7 (12.04 – 19.04). Opracowanie diagramu związków encji, diagramu przejść stanów, diagramu aktywności, diagramu sekwencji, diagramu współpracy, diagramu komponentów. Analiza kosztorysu. Konsultacje z klientem.

Zadanie 8 (19.04 – 26.04). Wybór odpowiednich narzędzi do szybkiego rozwijania aplikacji (ang. RAD). Ciąg dalszy prac nad projektem (zaprogramowanie GUI, podstawowych funkcjonalności).

Zadanie 9 (26.04 – 03.05). Prace nad programowaniem dalszych funkcjonalności systemowych.

Zadanie 10 (03.05 – 13.05). Wstępne testowanie aplikacji. Naprawa ewentualnych błędów.

Zadanie 11 (17.05 – 27.05). Zakończenie programowania projektu. Przeprowadzenie szczegółowych testów. Finalizacja projektu. Sprawdzenie i zamknięcie dokumentacji. Przygotowanie projektu do zaprezentowania klientowi.

Zadanie 12 (31.05 – 03.06). Prezentacja projektu klientowi. Zakończenie prac nad projektem.

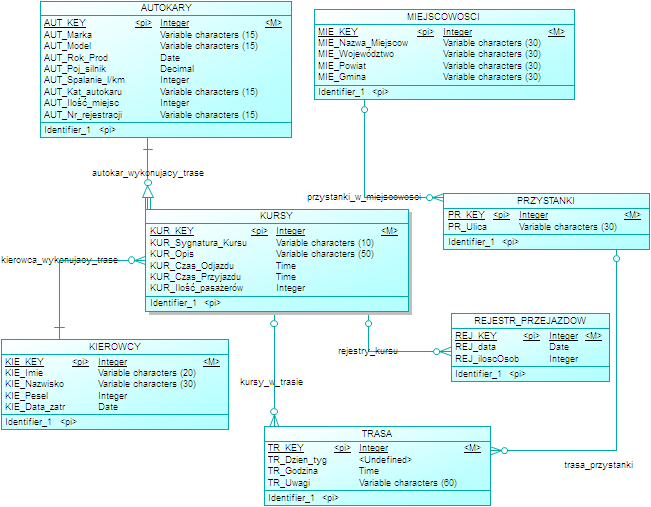
# Wizytówka.



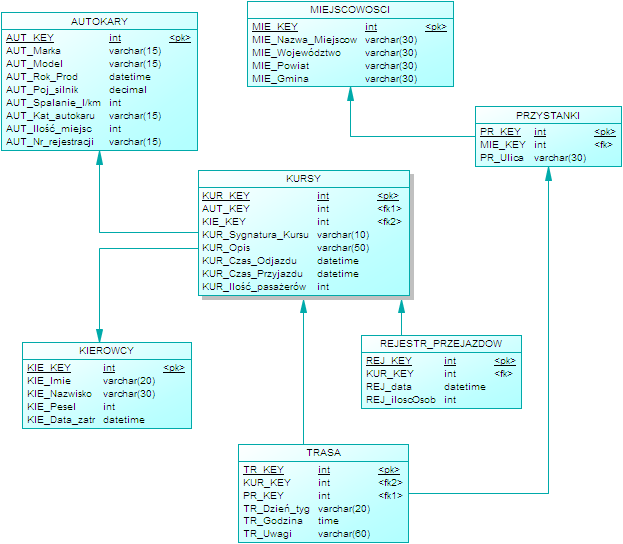
# Logo.



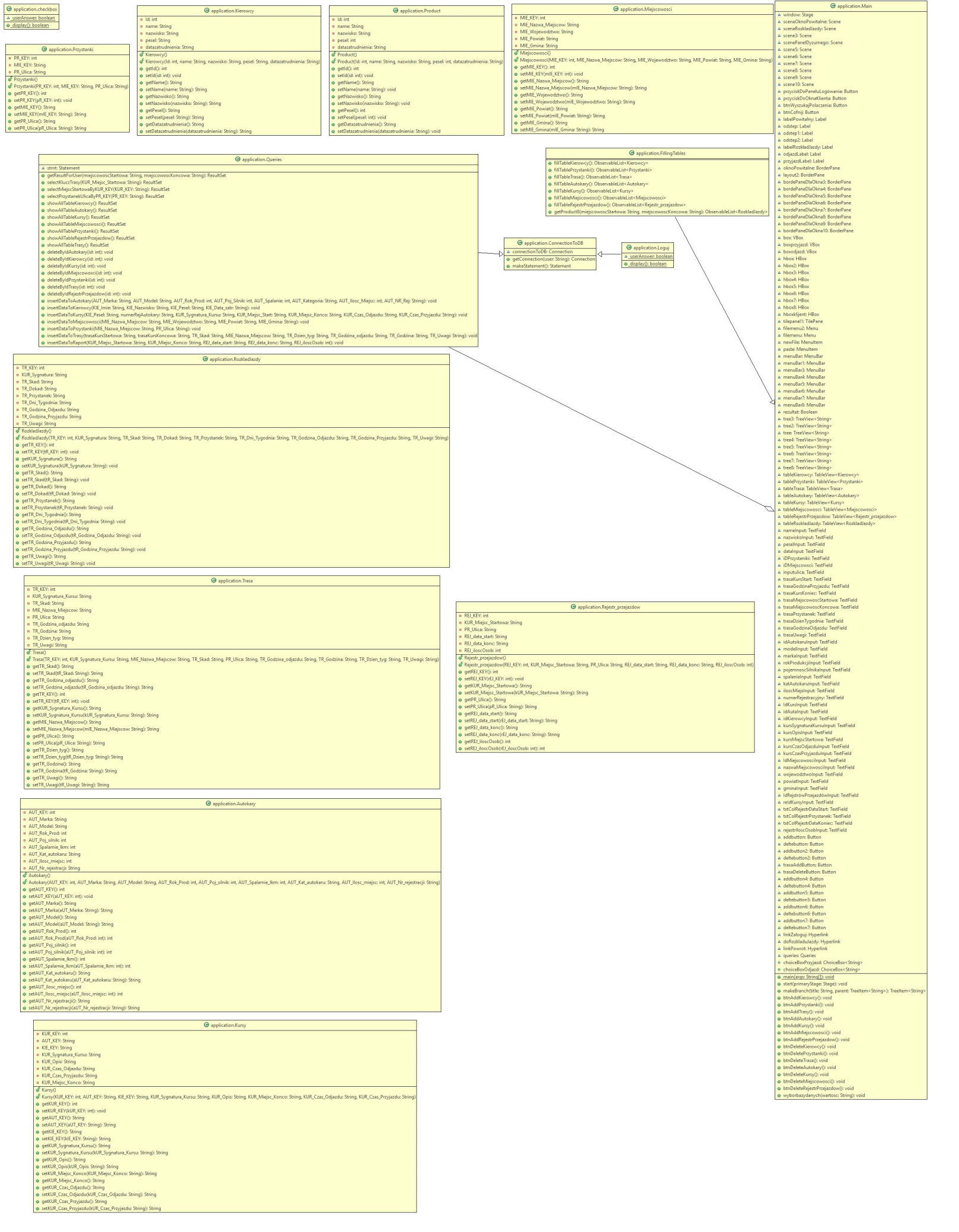
# Model konceptualny bazy danych.



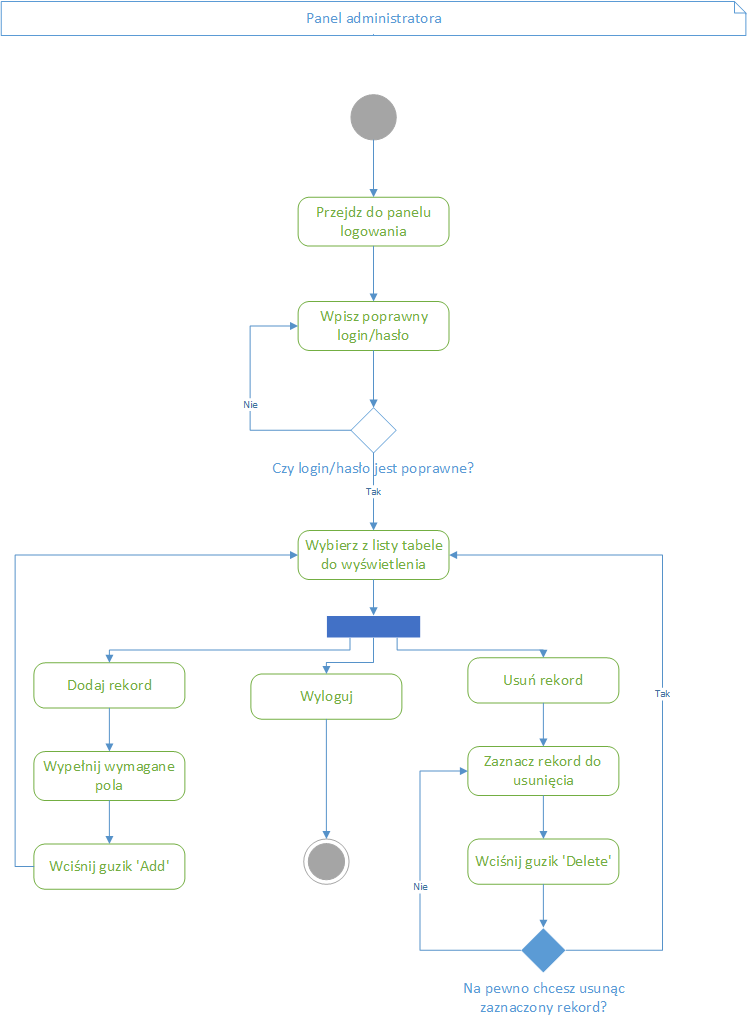
# Model fizyczny bazy danych.

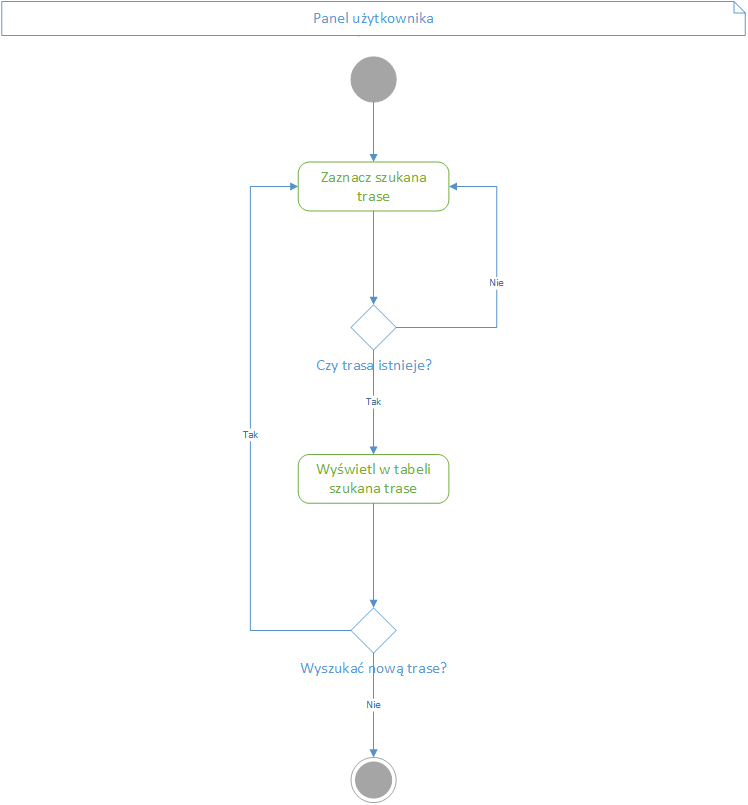


# Diagram klas.

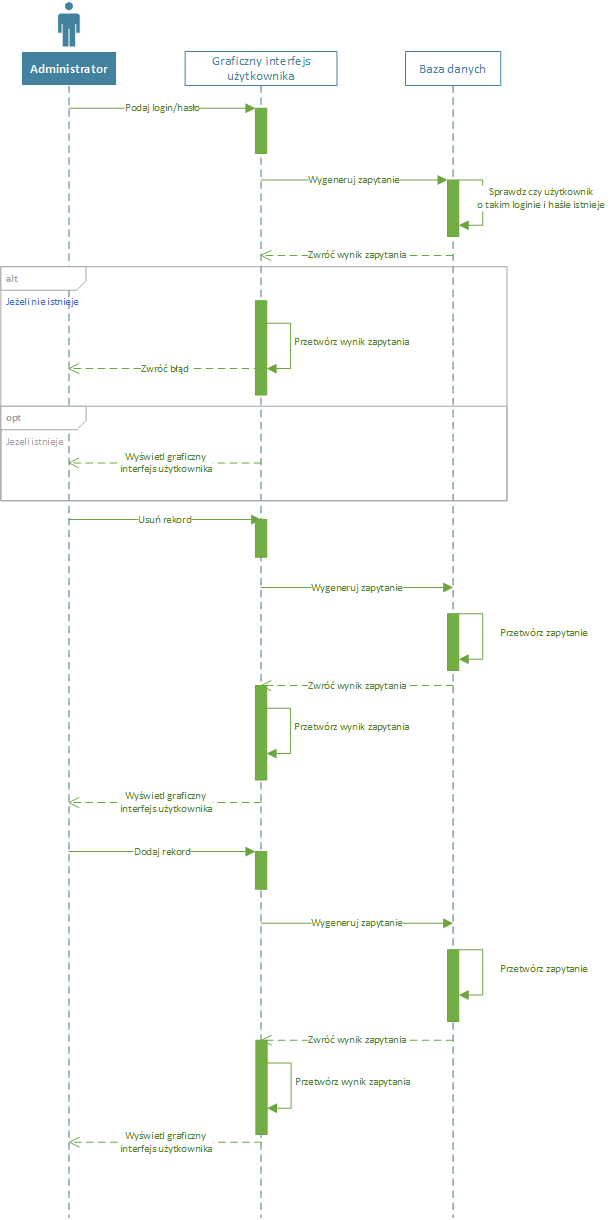


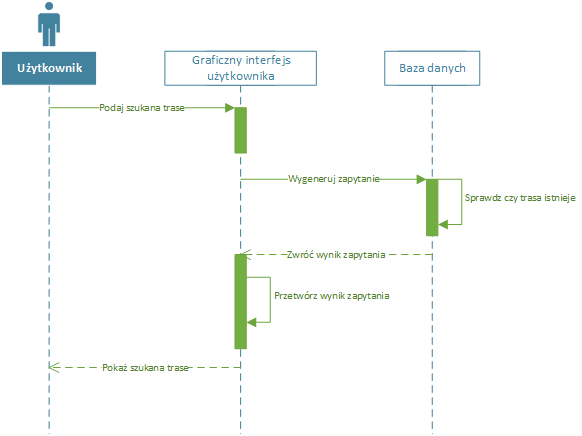
# Diagram czynności.





# Diagramy sekwencji.





# Metodologia. Narzędzia CASE.

Mianem narzędzi CASE (ang. Computer Aided Software Engineering) określa się systemy komputerowe, przeznaczone do wspomagania rutynowych czynności procesu tworzenia oprogramowania. To właśnie dzięki nim projekty tworzy się precyzyjniej, a praca nad diagramami, weryfikowanie ich poprawności oraz śledzenie wykonanych testów jest jeszcze szybsze i prostsze.

Narzędzia CASE zastosowane w projekcie:

* ArgoUML – producent University of California, licencja bezpłatna. (<http://argouml.tigris.org>)
* Eclipse – producent Eclipse Foundation, licencja Eclipse Public License. ([http://www.eclipse.org](http://www.eclipse.org/))
* Microsoft Office Excel – producent Microsoft, licencja komercyjna. (<http://www.office.microsoft.com>)
* Microsoft Office Word – producent Microsoft, licencja komercyjna. (<http://www.office.microsoft.com>)
* Microsoft SQL Server – producent Microsoft, licencja Microsoft EULA. ([http://www.microsoft.com/sql](http://www.microsoft.com/sql/))
* Microsoft SQL Server Management Studio Express – producent Microsoft, licencja Microsoft EULA. (<https://www.microsoft.com/en-us/server-cloud/products/sql-server>)
* Mozilla Firefox – producent Mozilla Corporation, licencja Mozilla Licensing. (<http://www.mozilla.org>)

# Kosztorys.

Poniżej zamieszczono wydatki uwzględnione w 4-miesięcznym procesie wdrażania projektu.



# Narzędzia RAD.

Rapid Application Development (RAD) to metodologia polegająca na udostępnieniu programiście dużych możliwości prototypowania oraz dużego zestawu gotowych komponentów (np. zapewniających dostęp do bazy danych). Umożliwia to uzyskanie pewnego efektu już w pierwszych krokach programistycznych, jednocześnie stanowi poważne zagrożenie dla projektów o większych rozmiarach ze względu na łatwość nieprzemyślanego modyfikowania.

Podczas pracy nad projektem wykorzystane zostało narzędzie Eclipse (framework) do tworzenia aplikacji w języku Java. Narzędzie to zostało wybrane z uwagi na nabyte wcześniej doświadczenie z tym oprogramowaniem.

# Testy przeprowadzone w ramach projektu.

testy modelu bazy danych

|  |  |  |
| --- | --- | --- |
| **testowany element** | **środowisko** | **rezultat** |
| encje | Sybase PowerDesigner 12.5 | pozytywny |
| połączenia między encjami | Sybase PowerDesigner 12.5 | pozytywny |
| klucze | Sybase PowerDesigner 12.5 | pozytywny |
| relacje pomiędzy encjami | Sybase PowerDesigner 12.5 | pozytywny |
| typy danych atrybutów encji | Sybase PowerDesigner 12.5 | pozytywny |
| dodawanie rekordów do tabel | Microsoft SQL Server 2014 | pozytywny |
| generowanie modelu fizycznego bazy danych na podstawie modelu konceptualnego | Sybase PowerDesigner 12.5 | pozytywny |

testy bazy danych / zgodności ze specyfikacją

|  |  |  |  |
| --- | --- | --- | --- |
| **nazwa testu** | **środowisko** | **warunek spełnienia** | **rezultat, komentarz** |
| model konceptualny – zachowanie logiki i zgodność ze specyfikacją wymagań | Sybase PowerDesigner 12.5 | * obecność tabeli „AUTOKARY”; * obecność tabeli „KIEROWCY”; * obecność tabeli „KURSY”; * poprawnie przypisane typy danych do atrybutów encji; | pozytywny |
| wygenerowanie modelu fizycznego na podstawie modelu konceptualnego | Sybase PowerDesigner 12.5 | * stworzenie działającego modelu fizycznego; | negatywny – w tabeli „TRASA” atrybut „TR\_Dzien\_tyg” ma nieprzypisany typ danych |
| model fizyczny – poprawność przypisania kluczy | Sybase PowerDesigner 12.5 | * każda tabela musi posiadać klucz podstawowy; * odpowiednio przypisane klucze obce; | pozytywny |
| wygenerowanie bazy danych na podstawie modelu fizycznego (poprawionego) | Sybase PowerDesigner 12.5 | * wygenerowanie kodu SQL; | pozytywny |
| baza danych – dodawanie rekordów do tabel | Microsoft SQL Server 2014 | * dodanie pojedynczego rekordu do każdej tabeli; | pozytywny |

testy gotowego produktu

|  |  |  |
| --- | --- | --- |
| **nazwa testu** | **warunki spełnienia** | **rezultat testu, komentarz** |
| GUI – menu główne | - możliwość przejścia do rozkładu jazdy  - możliwość przejścia do logowania | pozytywny |
| GUI – rozkład jazdy | - opcja powrotu do menu głównego  - możliwość wyboru miejsca startowego  - możliwość wyboru miejsca docelowego  - poprawne wyświetlenie kursów odpowiadających kryterium wyszukiwania | pozytywny |
| GUI – okno logowania | - możliwość wpisania loginu  - możliwość wpisania hasła  - obecność guzika „zaloguj” | pozytywny,  brak informacji o wpisaniu błędnego loginu lub hasła, hasło jest niezakodowane |
| GUI – trasy | - wyświetlenie wcześniejszych rekordów z tabeli TRASA  - dodanie rekordu do tabeli  - usunięcie rekordu z tabeli | pozytywny |
| GUI – przystanki | - wyświetlenie wcześniejszych rekordów z tabeli PRZYSTANKI  - dodanie rekordu do tabeli  - usunięcie rekordu z tabeli | pozytywny |
| GUI – kierowcy | - wyświetlenie wcześniejszych rekordów z tabeli KIEROWCY  - dodanie rekordu do tabeli  - usunięcie rekordu z tabeli | pozytywny |
| GUI – autokary | - wyświetlenie wcześniejszych rekordów z tabeli AUTOKARY  - dodanie rekordu do tabeli  - usunięcie rekordu z tabeli | pozytywny |
| GUI – kursy | - wyświetlenie wcześniejszych rekordów z tabeli KURSY  - dodanie rekordu do tabeli  - usunięcie rekordu z tabeli | pozytywny |
| GUI – miejscowości | - wyświetlenie wcześniejszych rekordów z tabeli MIEJSCOWOSCI  - dodanie rekordu do tabeli  - usunięcie rekordu z tabeli | pozytywny |
| GUI – rejestr przejazdów | - wyświetlenie wcześniejszych rekordów z tabeli REJESTR\_PRZEJAZDOW  - dodanie rekordu do tabeli  - usunięcie rekordu z tabeli | pozytywny |

DOKUMENT DETALICZNY PROJEKTU

dla firmy transportowej

wersja 1.0

# Streszczenie.

Niniejszy dokument detaliczny projektu (DDP) prezentuje szczegóły pracy zespołu projektowego, nad stworzeniem aplikacji bazodanowej wspomagającej organizację pracy w firmie transportowej. Pierwsza część dokumentu zawiera opis ogólnych założeń projektowych, druga zaś opisuje wykorzystane w projekcie komponenty.

SPIS TREŚCI

1. Streszczenie
2. Wprowadzenie
   1. Cel
   2. Zakres
   3. Definicje, akronimy, skróty
   4. Odsyłacze
   5. Opracowanie
3. Projekt. Standardy, konwencje, procedury
   1. Standardy projektowe
   2. Standardy dokumentacyjne
   3. Konwencje nazewnicze
   4. Standardy programistyczne
   5. Narzędzia
4. Specyfikacja komponentów
   1. Klasa „Autokary”
   2. Klasa „Kierowcy”
   3. Klasa „Kursy”
   4. Klasa „Miejscowosci”
   5. Klasa „Przystanki”
   6. Klasa „Rejestr\_przejazdow”
   7. Klasa „Rozklad\_jazdy”
   8. Klasa „Trasa”
   9. Klasa „checkbox”
   10. Klasa „Loguj”
   11. Klasa „Main”
   12. Klasa „ConnectionToDB”
   13. Klasa „FillingTables”
   14. Klasa „Queries”
   15. Biblioteka „sqljdbc4”
5. Załączniki
   1. Harmonogram prac
   2. Interfejs aplikacji
      1. Okno powitalne
      2. Logowanie do aplikacji
      3. Aplikacja
      4. Rozkład jazdy
      5. Okno dialogowe

# Wprowadzenie.

# Cel.

Dokument detaliczny projektu ma za zadanie przedstawić szczegółowo sposób realizowanych prac. Zgromadzone są w nim wszystkie informacje odnośnie budowy i działania oprogramowania. Określa on również założenia projektu, standardy, narzędzia i komponenty wchodzące w skład aplikacji.

# Zakres.

Założeniem projektu jest stworzenie bazodanowej aplikacji *DamberPol*, służącego do wspomagania organizacji pracy w firmie transportowej. System ma służyć do przechowywania różnego rodzaju informacji, m.in. rozkładu jazdy. Aplikacja pozwoli na magazynowanie informacji o kierowcach, pasażerach oraz pracownikach. Umożliwi pasażerom przeglądanie rozkładu jazdy.

# Definicje, akronimy, skróty.

* Pasażer – niezarejestrowany osoba firmy transportowej, nieposiana własnego loginu i hasła do systemu, posiada możliwość przeglądania zasobów rozkładu jazdy.
* Pracownik – pracownik firmy, posiada własny identyfikator i hasło do logowania do systemu, posiada możliwość przeglądania zasobów (raporty, rozkłady jazdy, kierowcy zatrudnieni w firmie itp.), dodawania, edytowania już istniejących pozycji.

# Odsyłacze.

Ustawa z dn. 29 sierpnia 1997 r. o ochronie danych osobowych (Dz. U. 1997 nr 133 poz. 883 z późn. zm.).

# Opracowanie.

Wyżej wymieniony dokument powstał na bazie specyfikacji wymagań systemu określonych podczas pierwszych etapów projektowania systemu. Zawiera on definicje standardów, które będą przestrzegane podczas realizacji projektu. Dalsza część dokumentu zawiera informacje o narzędziach, modułach i komponentach systemu oraz interfejsie graficznym oprogramowania.

# Projekt. Standardy, konwencje, procedury.

# Standardy projektowe.

Z uwagi na niewielkie doświadczenie zespołu dot. prac projektowych, wykorzystaliśmy tzw. model przyrostowy tworzenia oprogramowania. Model ten gwarantuje częste kontakty z klientem, brak konieczności zdefiniowania z góry całości wymagań, wczesne wykorzystanie przez klienta fragmentów systemu, możliwość elastycznego reagowania na opóźnienia realizacji fragmentu – przyspieszenie prac nad inną / innymi częściami. Wyżej wymienione punkty sprawiają, że ryzyko porażki prac nad całym projektem jest dużo mniejsze.

# Standardy dokumentacyjne.

Wszystkie dokumenty projektu stworzone zostały na podstawie jednego firmowego szablonu (papieru firmowego). W trakcie tworzenia dokumentacji zastosowany został jednolity, przejrzysty język, który jest wspólny dla całego dokumentu. Podczas programowania aplikacji zostały zastosowane komentarze – powodują one, że kod oprogramowania jest bardziej zrozumiały, a także umożliwiają łatwą edycję kodu aplikacji nawet przez osoby niezwiązane z pracą projektową.

# Konwencje nazewnicze.

Zastosowane w projekcie słownictwo jest ukierunkowane na prostotę i jednoznaczność. W implementacji została zachowana konwencja nazewnictwa Java.

# Standardy programistyczne.

System został zaprojektowany i stworzony w oparciu o język programowania Java. W projekcie będzie wykorzystywane podejście obiektowe do programowania oraz wykorzystywany będzie wzorzec projektowy MVC (Model-View-Controller). Aplikacja będzie stworzona zgodnie z wcześniejszymi założeniami i wcześniej ustalonym diagramem klas. Zaletami takiego podejścia jest brak zależności modelu od widoków aplikacji oraz łatwość dodawania i modyfikowania istniejących widoków bez wpływu na kluczową część systemu.

# Narzędzia.

Do zrealizowania oprogramowania wykorzystano język Java. Środowisko RAD w projekcie zastosowano w postaci aplikacji Eclipse. Podczas tworzenia dokumentacji posługiwaliśmy się oprogramowaniem StarUML i oprogramowaniem firmy Microsoft Office 2007. Do stworzenia bazy danych wykorzystaliśmy narzędzie „PowerDesigner” firmy Sybase.

Pełna lista narzędzi wykorzystywanych podczas pracy nad systemem i dokumentacją znajduje się poniżej.

* ArgoUML – producent University of California, licencja bezpłatna. (<http://argouml.tigris.org>)
* Eclipse – producent Eclipse Foundation, licencja Eclipse Public License. ([http://www.eclipse.org](http://www.eclipse.org/))
* Microsoft Office Excel – producent Microsoft, licencja komercyjna. (<http://www.office.microsoft.com>)
* Microsoft Office Word – producent Microsoft, licencja komercyjna. (<http://www.office.microsoft.com>)
* Microsoft SQL Server – producent Microsoft, licencja Microsoft EULA. ([http://www.microsoft.com/sql](http://www.microsoft.com/sql/))
* Microsoft SQL Server Management Studio Express – producent Microsoft, licencja Microsoft EULA. (<https://www.microsoft.com/en-us/server-cloud/products/sql-server>)
* Mozilla Firefox – producent Mozilla Corporation, licencja Mozilla Licensing. (<http://www.mozilla.org>)

# Specyfikacja komponentów.

# Klasa “Autokary”.

Klasa zawiera settery oraz gettery, które umożliwiają uzupełnienie tabel wartościami pobranymi z bazy danych z encji Autokary.

# Klasa “Kierowcy”.

Klasa zawiera settery oraz gettery, które umożliwiają uzupełnienie tabel wartościami pobranymi z bazy danych z encji Kierowcy.

# Klasa “Kursy”.

Klasa zawiera settery oraz gettery, które umożliwiają uzupełnienie tabel wartościami pobranymi z bazy danych z encji Kursy.

# Klasa “Miejscowosci”.

Klasa zawiera settery oraz gettery, które umożliwiają uzupełnienie tabel wartościami pobranymi z bazy danych z encji Miejscowosci.

# Klasa “Przystanki”.

Klasa zawiera settery oraz gettery, które umożliwiają uzupełnienie tabel wartościami pobranymi z bazy danych z encji Przystanki.

# Klasa “Rejestr\_przejazdow”.

Klasa zawiera settery oraz gettery, które umożliwiają uzupełnienie tabel wartościami pobranymi z bazy danych z encji Rejestr\_Przejazdow.

# Klasa “Rozklad\_Jazdy”.

Klasa zawiera settery oraz gettery, które umożliwiają uzupełnienie tabel wartościami pobranymi z bazy danych z encji Autokary.

# Klasa “Trasa”.

Klasa zawiera settery oraz gettery, które umożliwiają uzupełnienie tabel wartościami pobranymi z bazy danych z encji Autokary.

# Klasa “checkbox”.

Klasa odpowiadająca za wyświetlanie dodatkowego okna podczas próby usunięcia rekordu.

Okno ma za zadanie potwierdzić chęć usunięcia rekordu przez użytkownika.

# Klasa “Loguj”.

Klasa odpowiedzialna za wyświetlenie okna, które pozwala na zalogowanie się do aplikacji poprzez wpisanie poprawnego loginu oraz poprawnego hasła.

# Klasa “Main”.

Główna klasa odpowiedzialna za cale działanie aplikacji.

# Klasa “ConnectionToDB”.

Klasa odpowiedzialna za połączenie z bazą danych za pomocą poprawnie skonfigurowanego sterownika sqljdbc4.

# Klasa ”FillingTables”.

Klasa odpowiedzialna za wypełnianie tabel po wcześniejszym wywołaniu metody odpowiedzialnej za połączenie z bazą danych.

* 1. Klasa ”Queries”.

Klasa zawierająca metody, które realizują zapytania do bazy danych jak np. pobranie całej tabeli, wprowadzenie do niej danych.

# Biblioteka “sqljdbc4”.

Biblioteka odpowiedzialna za możliwość połączenia z bazą danych SQL Server.

# Załączniki.

# Harmonogram prac.

Zadanie 3 (15.03 – 22.03). Zaprojektowanie modelu bazy danych (model konceptualny i model fizyczny), stworzenie bazy danych na potrzeby projektu.

Zadanie 6 (01.04 – 06.04). Zaprojektowanie graficznego interfejsu użytkownika. Opracowanie słownika danych zawierający specyfikację modelu.

Zadanie 8 (19.04 – 26.04). Wybór odpowiednich narzędzi do szybkiego rozwijania aplikacji (ang. RAD). Ciąg dalszy prac nad projektem (zaprogramowanie GUI, podstawowych funkcjonalności).

Zadanie 9 (26.04 – 03.05). Prace nad programowaniem dalszych funkcjonalności systemowych.

Zadanie 10 (03.05 – 13.05). Wstępne testowanie aplikacji. Naprawa ewentualnych błędów.

Zadanie 11 (17.05 – 27.05). Zakończenie programowania projektu. Przeprowadzenie szczegółowych testów. Finalizacja projektu. Sprawdzenie i zamknięcie dokumentacji. Przygotowanie projektu do zaprezentowania klientowi.

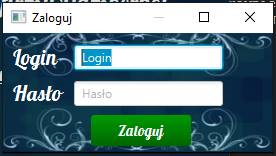
Zadanie 12 (31.05 – 03.06). Prezentacja projektu klientowi. Zakończenie prac nad projektem.

# Interfejs aplikacji.

# Okno powitalne.



# Logowanie do aplikacji.



# Aplikacja.



# Rozkład jazdy.



# Okno dialogowe.

