POLITECHNIKA KOSZALIŃSKA

**Baza PKS**

Dokumentacja z przedmiotu

systemy zarządzania relacyjnymi

bazami danych

**Dawid Gawiński**

**U-8421**

**Spis treści**

1. [Opis zadania](#page3)

2. [Wyznaczenia kategorii użytkowników i funkcji systemu dla każdej z tych kategorii](#page3)

3. [Stworzenie diagramu przypadków użycia (UML)](#page3)

4. [Opis słownika danych](#page4)

5. [Opis modelu konceptualnego](#page11)

6. [Opis modelu fizycznego](#page12)

7. [Kody SQL do tworzenia tabel bazy danych](#page13)

8. [Kody zapytań SQL do bazy danych wraz z opisem](#page15)

9. [Opis aplikacji lokalnej , diagramy klas aplikacji](#page16)

a) [Opis funkcjonalności](#page16)

b) [Wybrana technologia](#page17)

c) [Diagram klas](#page18)

d) [Opis struktury programu](#page21)

e) [Opis najważniejszych funkcji programu](#page28)

[10. Opis aplikacji webowej](#page31)

a) [Opis funkcjonalności](#page31)

b) [Wybrana technologia](#page31)

c) [Graf przejść](#page32)

d) [Opis struktury programu](#page33)

[10. Wnioski](#page39)

**1. Opis zadania**

Zadania w ramach przedmiotu z „Zarządzania relacyjnymi bazami danych” polegały na zaprojektowaniu bazy danych, utworzeniu aplikacji lokalnej oraz internetowej, która ma zarządzać tą bazą oraz dokumentacji.

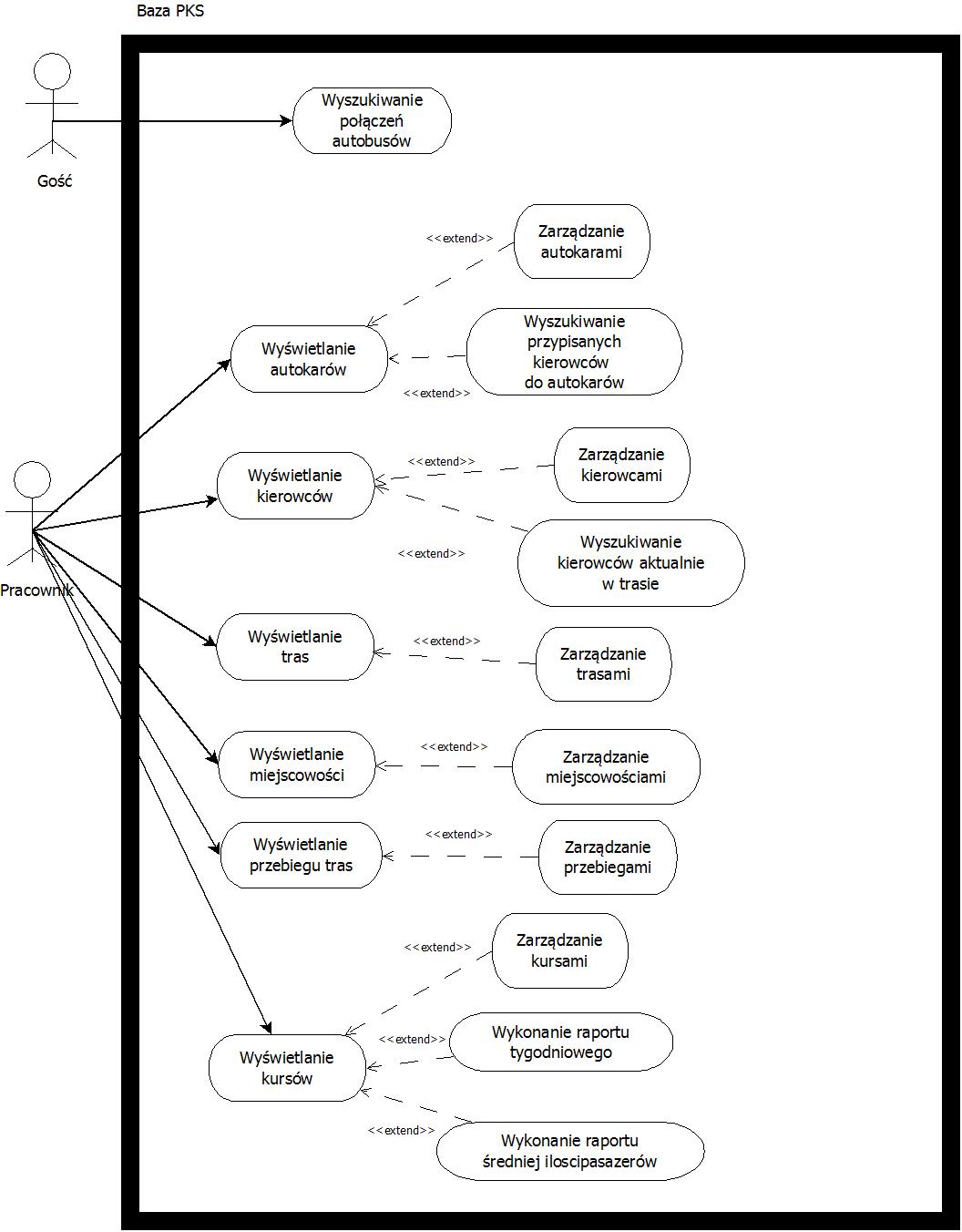
Mój projekt przeznaczony jest dla ludzi szukających połączeń autobusów PKS według wybranych przystanków, daty oraz czasu i dla pracowników, by w sposób łatwy byli w stanie zarządzać tymi połączeniami.

**2. Wyznaczenia kategorii użytkowników i funkcji systemu dla każdej z tych kategorii**

W oby dwóch aplikacjach zostały utworzone następujące kategorie użytkowników:

* użytkownik niezalogowany, który ma on dostęp do wyszukiwania połączeń autobusów potrzebnych dla tego użytkownika;
* użytkownik zalogowany, który jest pracownikiem firmy i ma możliwość edytowania wszystkich tabel.

**3. Stworzenie diagramu przypadków użycia (UML)**



**4. Opis słownika danych**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *ZBIÓR ENCJI „Kierowca”* | | | | |
| **Nazwa** | **Opis** | **Typ** | **Czy złożony?** | **Dziedzina atrybutu** |
| ID\_kierowcy | Numer jednoznacznie identyfikujący kierowcę | Liczba całkowita | Nie | Zbiór dwucyfrowych liczb całkowitych bez powtórzeń |
| Imie\_Nazwisko | Imię i nazwisko kierowcy | Znakowy | Tak, Imię Nazwisko | Zbiór łańcuchów znakowych opisujących imię i nazwisko kierowcy |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *ZBIÓR ENCJI „Autokar”* | | | | |
| **Nazwa** | **Opis** | **Typ** | **Czy złożony?** | **Dziedzina atrybutu** |
| ID\_autokaru | Numer jednoznacznie identyfikujący autokar | Liczba całkowita | Nie | Zbiór dwucyfrowych liczb całkowitych bez powtórzeń |
| Nr\_rejestr | Numer rejestracyjny autokaru | Znakowy | Nie | Zbiór łańcuchów znakowych opisujących numer rejestracyjny autokaru |
| Marka | Marka autokaru | Znakowy | Nie | Zbiór łańcuchów znakowych opisujących markę autokaru |
| Model | Model autokaru | Znakowy | Nie | Zbiór łańcuchów znakowych opisujących model autokaru |
| Liczba\_miejsc | Zmienna identyfikująca liczbę miejsc w autokarze | Liczba całkowita | Nie | Zbiór dwucyfrowych liczb całkowitych określających liczbę miejsc w autokarze |

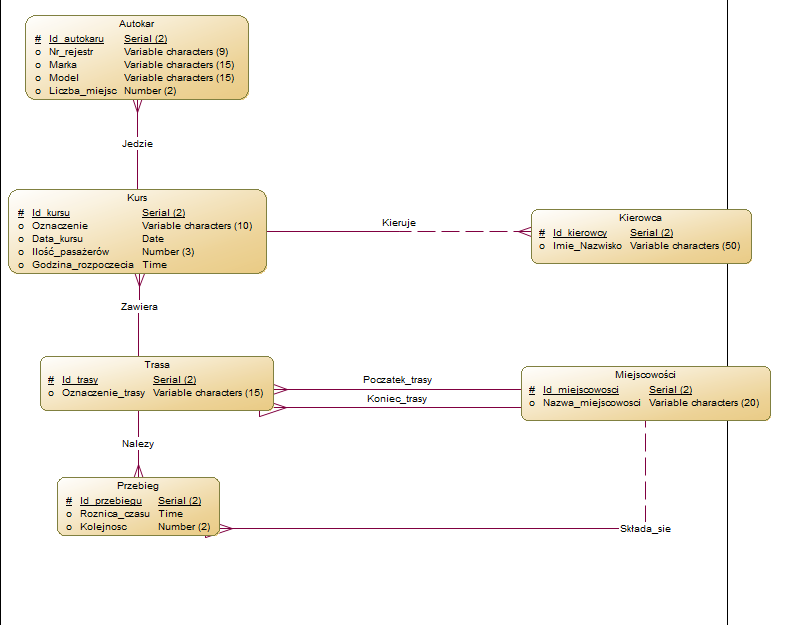
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *ZBIÓR ENCJI „Przebieg”* | | | | |
| **Nazwa** | **Opis** | **Typ** | **Czy złożony?** | **Dziedzina atrybutu** |
| ID\_przebiegu | Numer jednoznacznie identyfikujący przebieg pomiędzy miejscowościami | Liczba całkowita | Nie | Zbiór dwucyfrowych liczb całkowitych bez powtórzeń |
| Roznica\_czasu | Różnica czasu pomiędzy daną miejscowością a miejscowością początkową | Czas | Nie | Zbiór łańcuchów znakowych opisujących imię i nazwisko pracownika |
| Kolejnosc | Zmienna identyfikująca kolejność występowania przystanku na trasie | Liczba całkowita | Nie | Zbiór dwucyfrowych liczb całkowitych określających kolejność przystanków |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *ZBIÓR ENCJI „Miejscowości”* | | | | |
| **Nazwa** | **Opis** | **Typ** | **Czy złożony?** | **Dziedzina atrybutu** |
| ID\_miejscowosci | Numer jednoznacznie identyfikujący miejscowość | Liczba całkowita | Nie | Zbiór dwucyfrowych liczb całkowitych bez powtórzeń |
| Nazwa\_miejscowosci | Nazwa miejscowości w której znajduję się przystanek | Znakowy | Tak | Zbiór łańcuchów znakowych opisujących nazwę miejscowości |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *ZBIÓR ENCJI „Trasa”* | | | | |
| **Nazwa** | **Opis** | **Typ** | **Czy złożony?** | **Dziedzina atrybutu** |
| ID\_trasy | Numer jednoznacznie identyfikujący trasę | Liczba całkowita | Nie | Zbiór dwucyfrowych liczb całkowitych bez powtórzeń |
| Oznaczenie\_trasy | Oznaczenie trasy | Znakowy | Nie | Zbiór łańcuchów znakowych opisujących trasę |

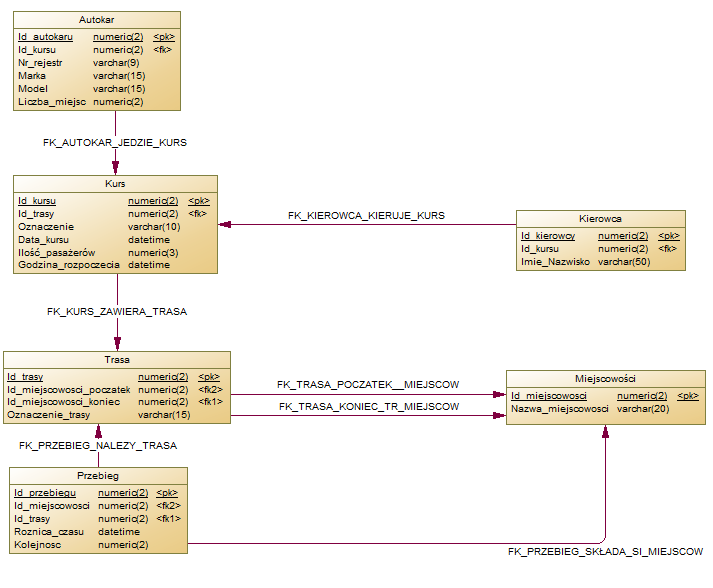
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *ZBIÓR ENCJI „Kurs”* | | | | |
| **Nazwa** | **Opis** | **Typ** | **Czy złożony?** | **Dziedzina atrybutu** |
| ID\_kursu | Numer jednoznacznie identyfikujący kurs | Liczba całkowita | Nie | Zbiór dwucyfrowych liczb całkowitych bez powtórzeń |
| Oznaczenie | Oznaczenie kursu | Znakowy | Nie | Zbiór łańcuchów znakowych opisujących kurs |
| Data\_kursu | Data wystąpienia kursu | Data | Tak  Dzień  Miesiąc  Rok | Data opisująca datę wystąpienia kursu |
| Ilosc\_pasazerow | Ilość pasażerów, która skorzystała z danego kursu | Liczba całkowita | Nie | Zbiór trzycyfrowych liczb całkowitych określających liczbę pasażerów |
| Godzina\_rozpoczecia | Godzina rozpoczęcia kursu na przystanku początkowym | Czas | Nie | Zbiór godzin z minutami określających rozpoczęcie danego kursu |

**5. Opis modelu konceptualnego**





**6.Opis modelu fizycznego**



**7. Kody SQL do tworzenia tabel bazy danych**

if exists (select 1

from sysindexes

where id = object\_id('Autokar')

and name = 'Jedzie\_FK'

and indid > 0

and indid < 255)

drop index Autokar.Jedzie\_FK

go

if exists (select 1

from sysobjects

where id = object\_id('Autokar')

and type = 'U')

drop table Autokar

go

if exists (select 1

from sysindexes

where id = object\_id('Kierowca')

and name = 'Kieruje\_FK'

and indid > 0

and indid < 255)

drop index Kierowca.Kieruje\_FK

go

if exists (select 1

from sysobjects

where id = object\_id('Kierowca')

and type = 'U')

drop table Kierowca

go

if exists (select 1

from sysindexes

where id = object\_id('Kurs')

and name = 'Zawiera\_FK'

and indid > 0

and indid < 255)

drop index Kurs.Zawiera\_FK

go

if exists (select 1

from sysobjects

where id = object\_id('Kurs')

and type = 'U')

drop table Kurs

go

if exists (select 1

from sysobjects

where id = object\_id('Miejscowości')

and type = 'U')

drop table Miejscowości

go

if exists (select 1

from sysindexes

where id = object\_id('Przebieg')

and name = 'Składa\_sie\_FK'

and indid > 0

and indid < 255)

drop index Przebieg.Składa\_sie\_FK

go

if exists (select 1

from sysindexes

where id = object\_id('Przebieg')

and name = 'Nalezy\_FK'

and indid > 0

and indid < 255)

drop index Przebieg.Nalezy\_FK

go

if exists (select 1

from sysobjects

where id = object\_id('Przebieg')

and type = 'U')

drop table Przebieg

go

if exists (select 1

from sysindexes

where id = object\_id('Trasa')

and name = 'Poczatek\_trasy\_FK'

and indid > 0

and indid < 255)

drop index Trasa.Poczatek\_trasy\_FK

go

if exists (select 1

from sysindexes

where id = object\_id('Trasa')

and name = 'Koniec\_trasy\_FK'

and indid > 0

and indid < 255)

drop index Trasa.Koniec\_trasy\_FK

go

if exists (select 1

from sysobjects

where id = object\_id('Trasa')

and type = 'U')

drop table Trasa

go

/\*==============================================================\*/

/\* Table: Autokar \*/

/\*==============================================================\*/

create table Autokar (

Id\_autokaru numeric(2) identity,

Id\_kursu numeric(2) not null,

Nr\_rejestr varchar(9) null,

Marka varchar(15) null,

Model varchar(15) null,

Liczba\_miejsc numeric(2) null,

constraint PK\_AUTOKAR primary key nonclustered (Id\_autokaru)

)

go

/\*==============================================================\*/

/\* Index: Jedzie\_FK \*/

/\*==============================================================\*/

create index Jedzie\_FK on Autokar (

Id\_kursu ASC

)

go

/\*==============================================================\*/

/\* Table: Kierowca \*/

/\*==============================================================\*/

create table Kierowca (

Id\_kierowcy numeric(2) identity,

Id\_kursu numeric(2) null,

Imie\_Nazwisko varchar(50) null,

constraint PK\_KIEROWCA primary key nonclustered (Id\_kierowcy)

)

go

/\*==============================================================\*/

/\* Index: Kieruje\_FK \*/

/\*==============================================================\*/

create index Kieruje\_FK on Kierowca (

Id\_kursu ASC

)

go

/\*==============================================================\*/

/\* Table: Kurs \*/

/\*==============================================================\*/

create table Kurs (

Id\_kursu numeric(2) identity,

Id\_trasy numeric(2) not null,

Oznaczenie varchar(10) null,

Data\_kursu datetime null,

Ilość\_pasażerów numeric(3) null,

Godzina\_rozpoczecia datetime null,

constraint PK\_KURS primary key nonclustered (Id\_kursu)

)

go

/\*==============================================================\*/

/\* Index: Zawiera\_FK \*/

/\*==============================================================\*/

create index Zawiera\_FK on Kurs (

Id\_trasy ASC

)

go

/\*==============================================================\*/

/\* Table: Miejscowości \*/

/\*==============================================================\*/

create table Miejscowości (

Id\_miejscowosci numeric(2) identity,

Nazwa\_miejscowosci varchar(20) null,

constraint PK\_MIEJSCOWOŚCI primary key nonclustered (Id\_miejscowosci)

)

go

/\*==============================================================\*/

/\* Table: Przebieg \*/

/\*==============================================================\*/

create table Przebieg (

Id\_przebiegu numeric(2) identity,

Id\_miejscowosci numeric(2) not null,

Id\_trasy numeric(2) not null,

Roznica\_czasu datetime null,

Kolejnosc numeric(2) null,

constraint PK\_PRZEBIEG primary key nonclustered (Id\_przebiegu)

)

go

/\*==============================================================\*/

/\* Index: Nalezy\_FK \*/

/\*==============================================================\*/

create index Nalezy\_FK on Przebieg (

Id\_trasy ASC

)

go

/\*==============================================================\*/

/\* Index: Składa\_sie\_FK \*/

/\*==============================================================\*/

create index Składa\_sie\_FK on Przebieg (

Id\_miejscowosci ASC

)

go

/\*==============================================================\*/

/\* Table: Trasa \*/

/\*==============================================================\*/

create table Trasa (

Id\_trasy numeric(2) identity,

Id\_miejscowosci\_poczatek numeric(2) not null,

Id\_miejscowosci\_koniec numeric(2) not null,

Oznaczenie\_trasy varchar(15) null,

constraint PK\_TRASA primary key nonclustered (Id\_trasy)

)

go

/\*==============================================================\*/

/\* Index: Koniec\_trasy\_FK \*/

/\*==============================================================\*/

create index Koniec\_trasy\_FK on Trasa (

Id\_miejscowosci\_koniec ASC

)

go

/\*==============================================================\*/

/\* Index: Poczatek\_trasy\_FK \*/

/\*==============================================================\*/

create index Poczatek\_trasy\_FK on Trasa (

Id\_miejscowosci\_poczatek ASC

)

go

* 1. **Kody zapytań SQL do bazy danych wraz z opisem**

**1.Otrzymać przebieg trasy kursu pomiędzy punktem startowym a końcowy**

Select m.Nazwa\_miejscowosci

From Trasa AS t

Inner join Kurs AS k

ON t.Id\_trasy=k.Id\_trasy

Inner join Przebieg AS p

on p.Id\_trasy=t.Id\_trasy

Inner join Miejscowości AS m

on m.Id\_miejscowosci=p.Id\_miejscowosci

Where( t.Id\_miejscowosci\_poczatek!=p.Id\_miejscowosci OR t.Id\_miejscowosci\_koniec!=p.Id\_miejscowosci ) and k.Id\_kursu=2

Order by p.Kolejnosc

**2.Otrzymać kierowców którzy są aktualnie w trasie**

Select k.Imie\_Nazwisko

From Kierowca AS k

Inner Join kurs AS ku

ON k.Id\_kursu=ku.Id\_kursu

WHERE ku.Data\_kursu=CONVERT(date,SYSDATETIME()) and ku.Godzina\_rozpoczecia<CONVERT(time,SYSDATETIME())

**3.Otrzymać raport tygodniowy na temat ilości pasażerów korzystających z danego kursu**

Select SUM(Ilość\_pasażerów) AS Tygodniowy

FROM Kurs

WHERE Data\_kursu<CONVERT(date,SYSDATETIME()

**4.Otrzymać listę połączeń pomiędzy miejscowościami dla różnych kursów**

Select k.Id\_kursu,m.Nazwa\_miejscowosci,k.Godzina\_rozpoczecia,p.Roznica\_czasu

From Kurs AS k

INNER JOIN Trasa as t

ON k.Id\_trasy=t.Id\_trasy

INNER JOIN Przebieg AS p

ON p.Id\_trasy=t.Id\_trasy

INNER JOIN Miejscowości AS m

ON m.Id\_miejscowosci=p.Id\_miejscowosci

WHERE p.Id\_miejscowosci=5 OR p.Id\_miejscowosci=1

Order by p.Roznica\_czasu

**5.Otrzymać raport na temat ilości pasażerów korzystających z danego kursu**

Select (SUM(Ilość\_pasażerów)/COUNT(Id\_kursu)) AS Tygodniowy

FROM Kurs AS k

INNER JOIN Trasa AS t

on k.Id\_trasy=t.Id\_trasy

WHERE Data\_kursu<=CONVERT(date,SYSDATETIME()) and k.Id\_kursu=2

**6.Otrzymać raport harmonogramu ruchu pojazdu po trasie, skojarzoną z wybranym kursem**

Select m.Nazwa\_miejscowosci,k.Godzina\_rozpoczecia,p.Roznica\_czasu

From Trasa AS t

Inner join Kurs AS k

ON t.Id\_trasy=k.Id\_trasy

Inner join Przebieg AS p

on p.Id\_trasy=t.Id\_trasy

Inner join Miejscowości AS m

on m.Id\_miejscowosci=p.Id\_miejscowosci

Where( t.Id\_miejscowosci\_poczatek!=p.Id\_miejscowosci OR t.Id\_miejscowosci\_koniec!=p.Id\_miejscowosci ) and k.Id\_kursu=2

Order by p.Kolejnosc

**7.Otrzymać raport pro przeznaczenie kierowców pojazdom**

Select k.Imie\_Nazwisko

FROM Kierowca AS k

INNER JOIN Kurs AS ku

ON ku.Id\_kursu=k.Id\_kursu

INNER JOIN Autokar AS a

on a.Id\_kursu=k.Id\_kursu

**9. Opis aplikacji lokalnej , diagramy klas aplikacji**

**a) Opis funkcjonalności**

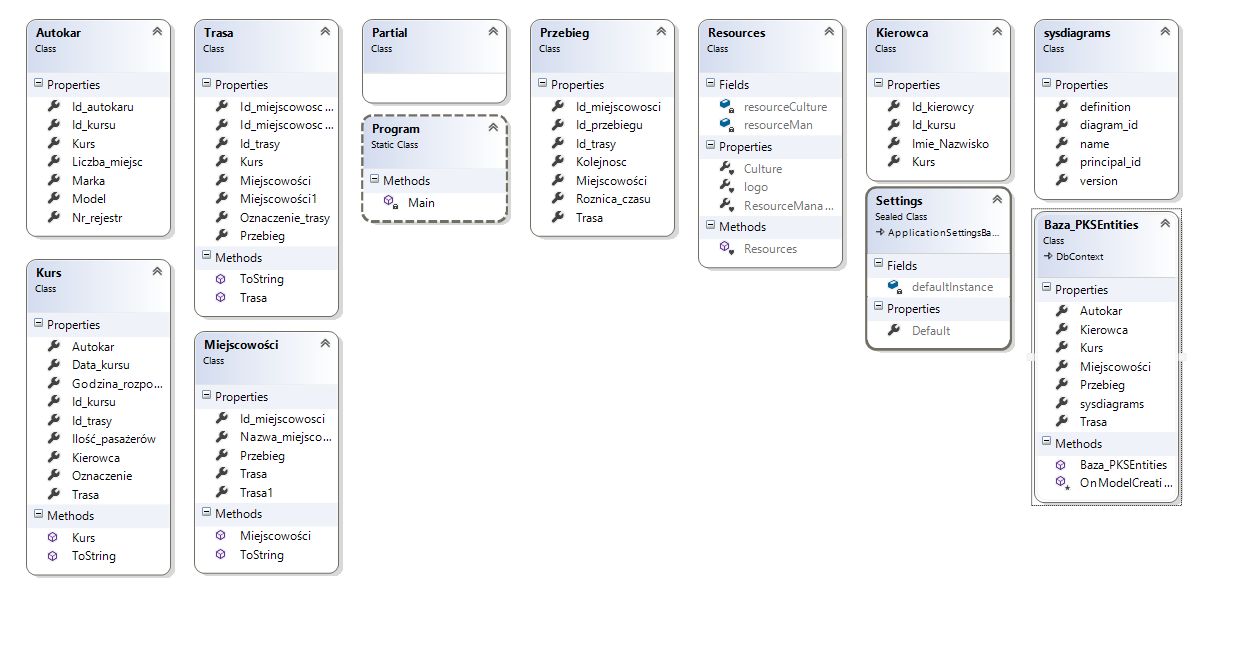
Aplikacja lokalna udostępnia następujące funkcjonalności:

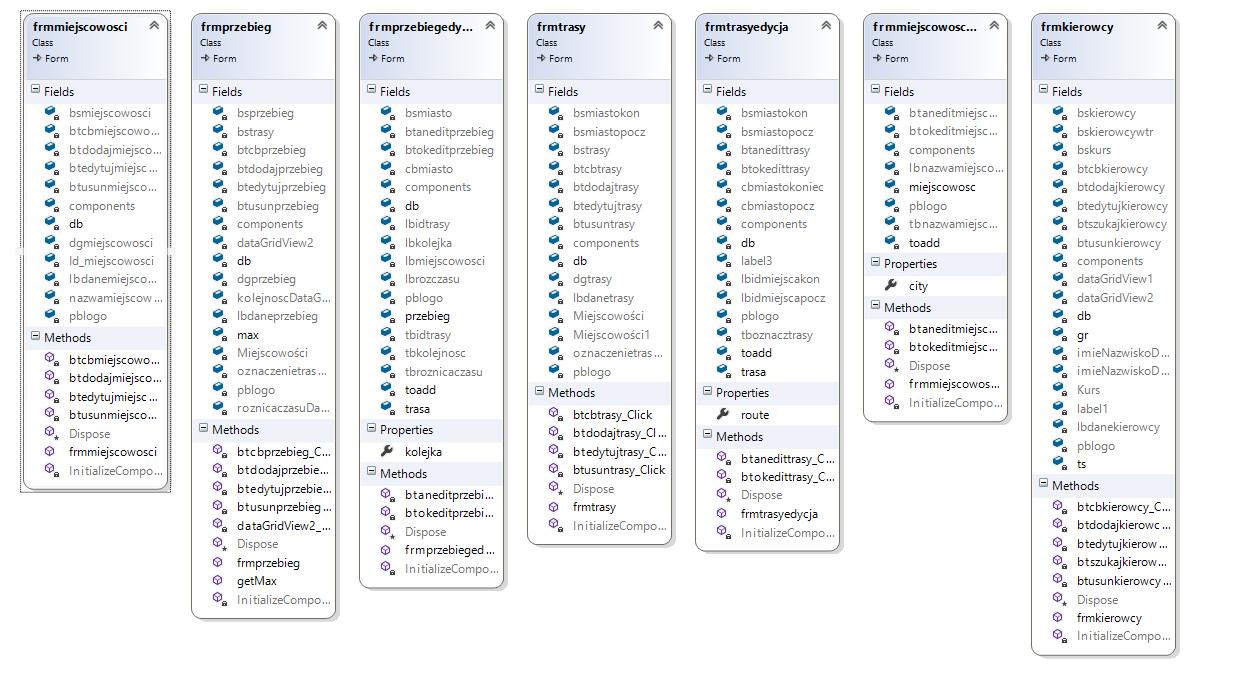
* Przeglądanie połączeń autokarów, po uprzednim podaniu dodatkowych parametrów;
* Administrowanie autokarami (usuwanie, edycja, dodawanie) przez upoważnionego pracownika;
* Wyszukiwanie przypisanych kierowców do wybranego autokaru;
* Administrowanie kierowcami (usuwanie, edycja, dodawanie) przez upoważnionego pracownika;
* Wyszukiwanie kierowców, którzy są w aktualnej chwili w trasie;
* Administrowanie trasami (edycja, usuwanie) przez upoważnionego pracownika;
* Administrowanie miejscowościami (edycja, usuwanie) przez upoważnionego pracownika;
* Administrowanie przebiegami pomiędzy przystankami (edycja, usuwanie) przez upoważnionego pracownika;
* Administrowanie kursami (edycja, usuwanie) przez upoważnionego pracownika;
* Generowanie raportu tygodniowego związanego z liczbą pasażerów danego kursu;
* Generowanie raportu liczącego średnią liczbę pasażerów od początku istnienia danego kursu

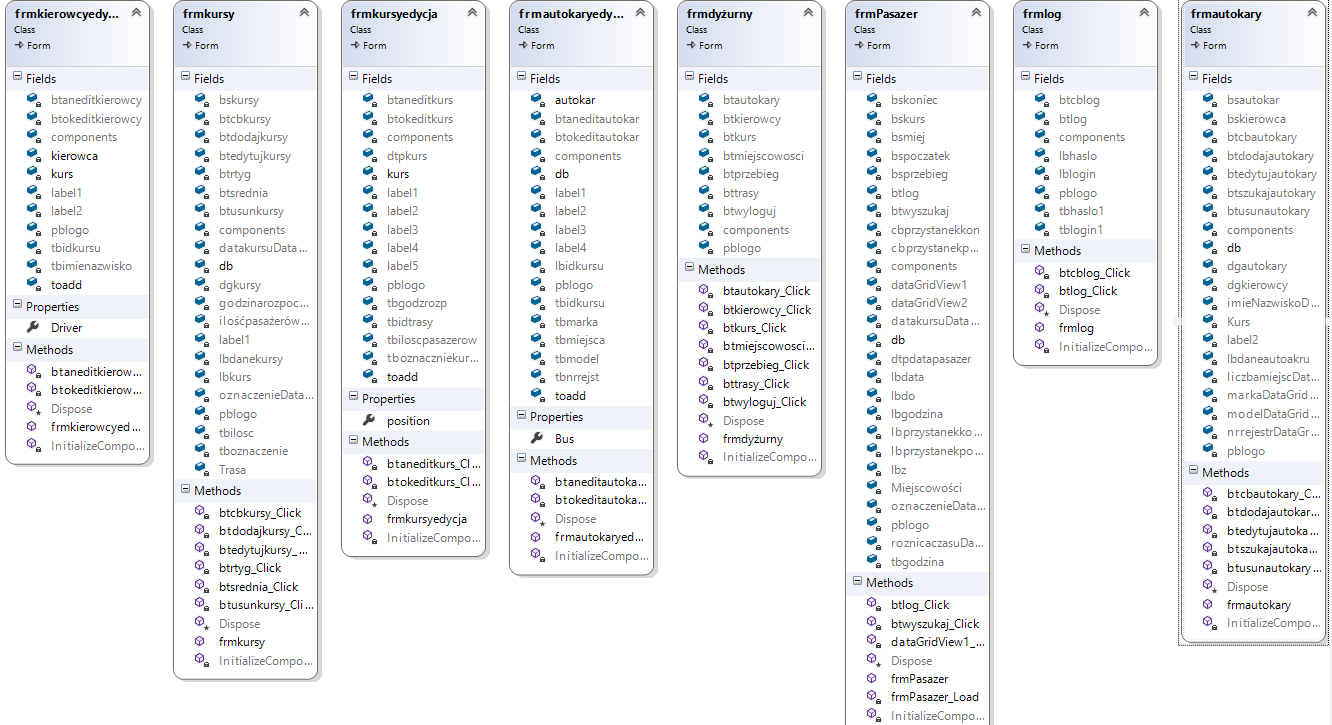
**b) Wybrana technologia**

Aplikacja została zrealizowana w C# przy użyciu środowiska Microsoft Visual Studio 2015 oraz następującej biblioteki:

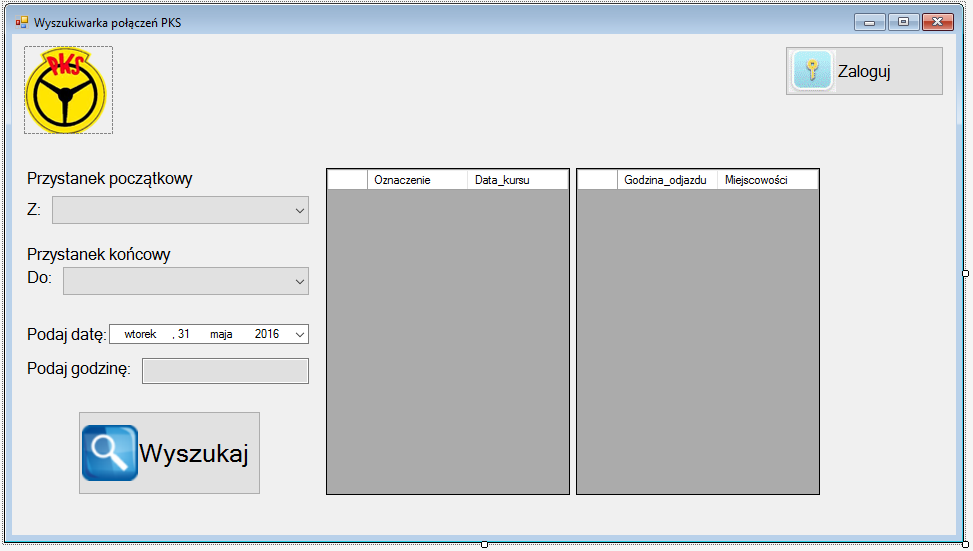
* Entity Framework 6.0 w celu połączenia bazy danych z aplikacją;

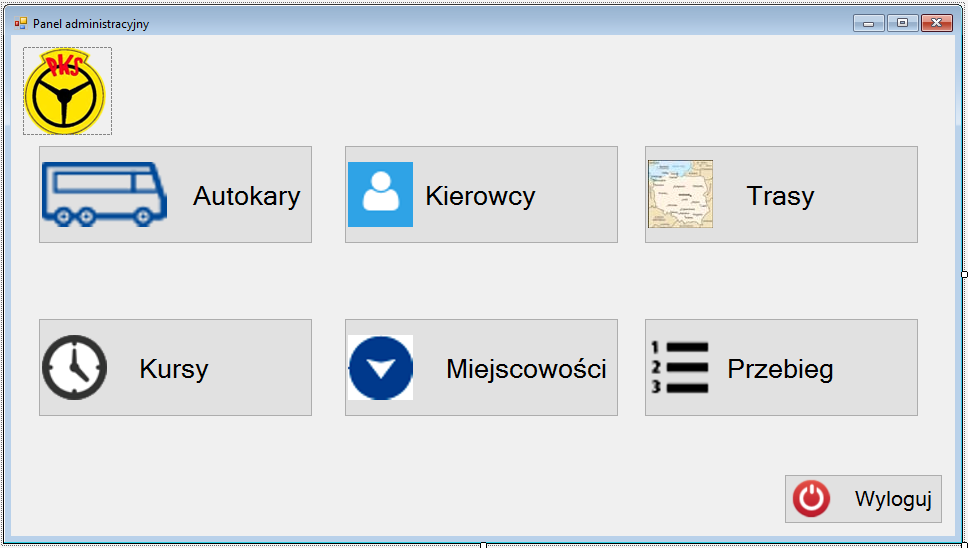
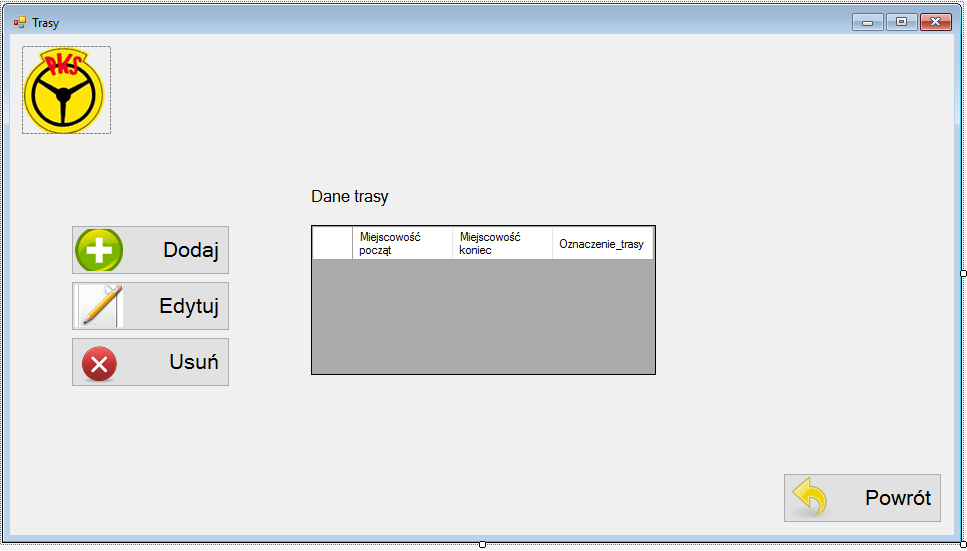
**c) Diagram klas**

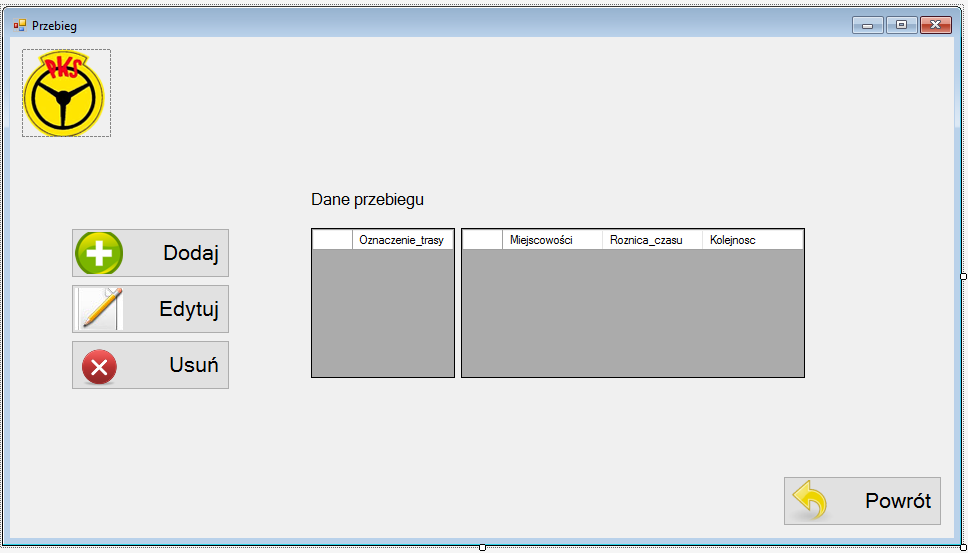
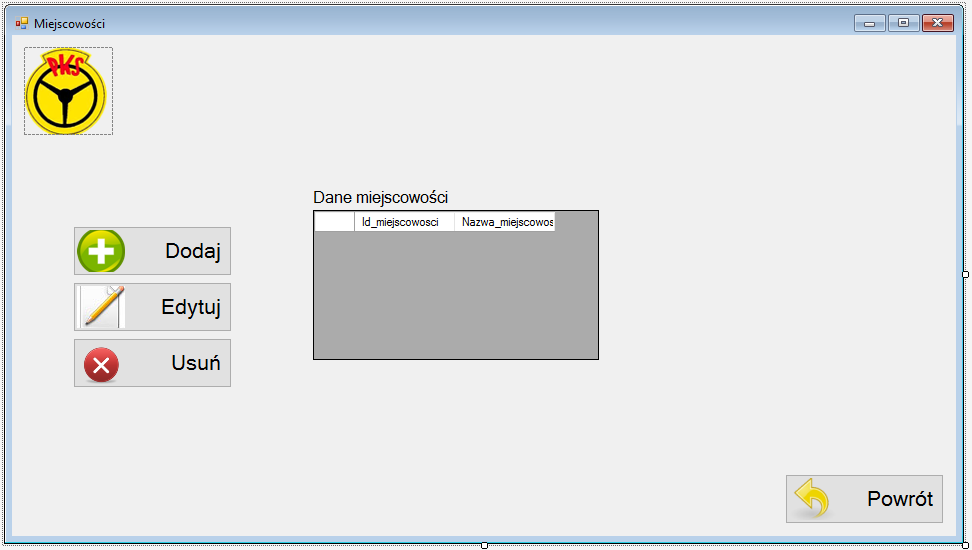
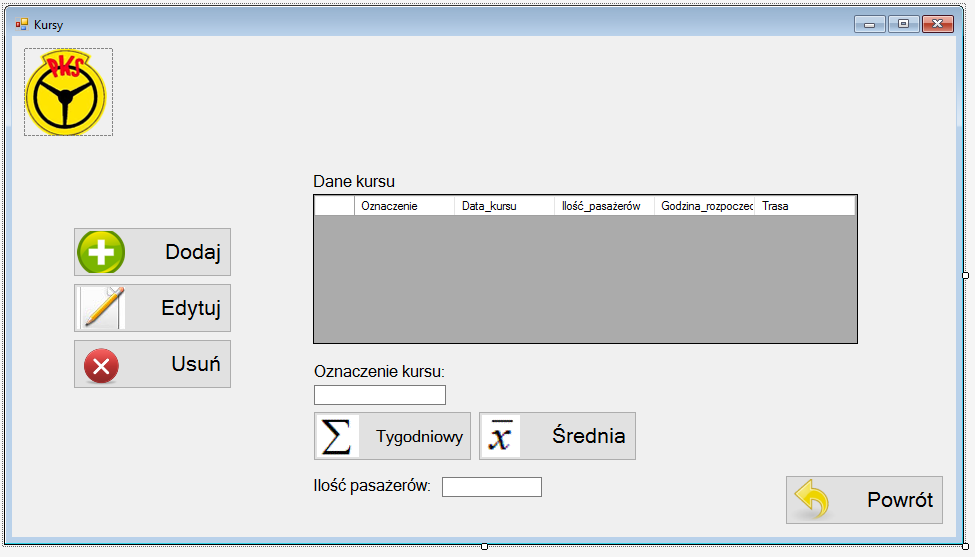


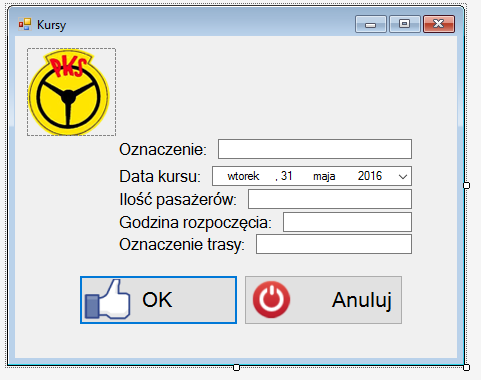


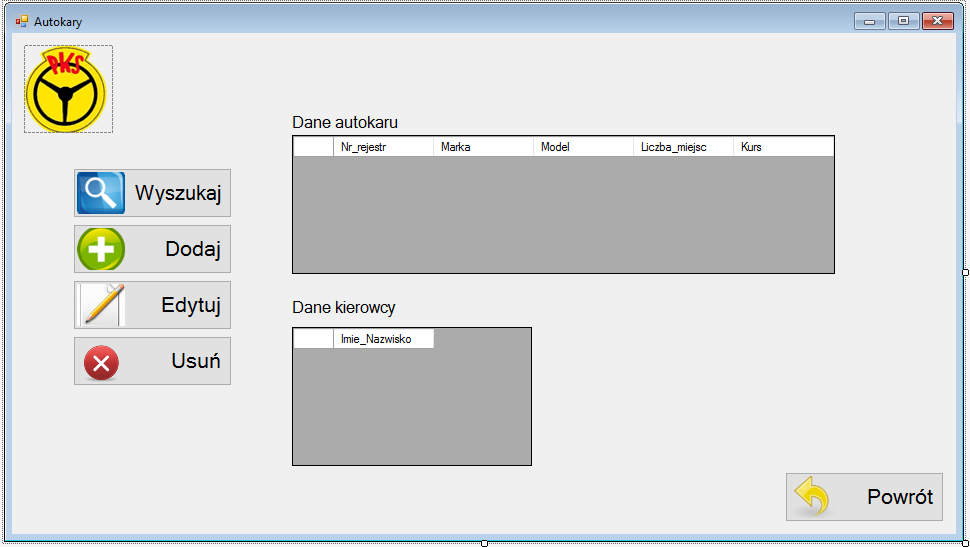
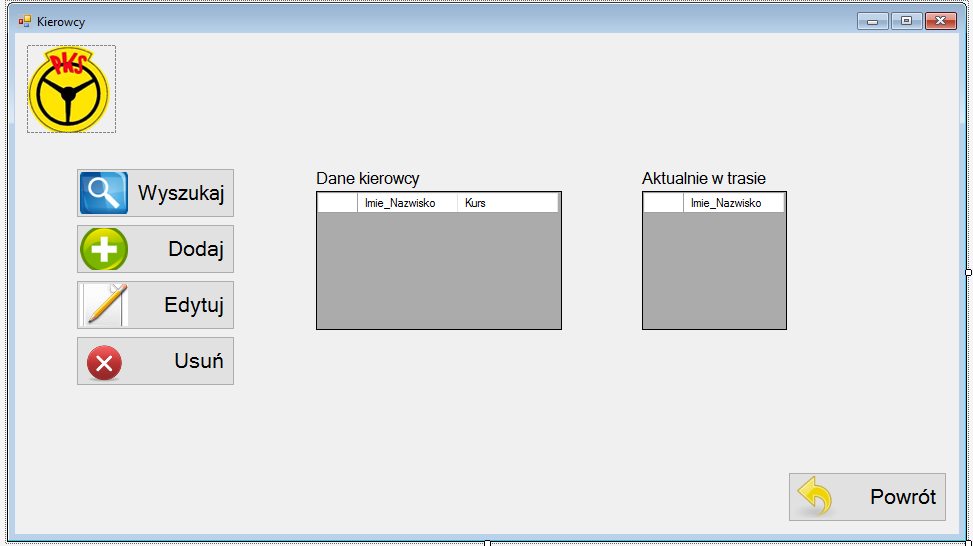
**d) Opis struktury programu**



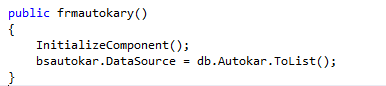




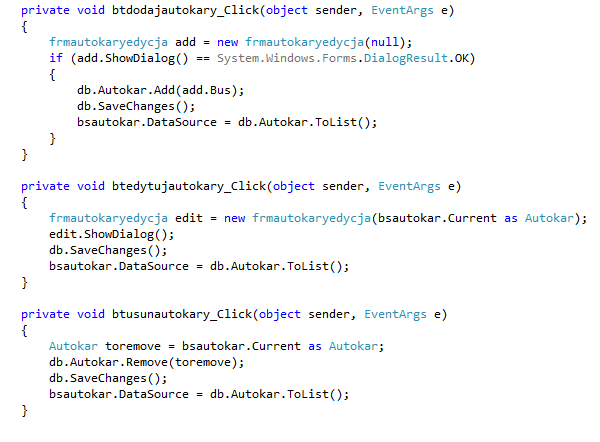




**e) Opis najważniejszych funkcji programu**



**Rysunek Funkcja pobierająca wartości z bazy danych**

**Rysunek Fragment kodu odpowiedzialny za usuwanie wiesza z bazy danyc****h**

1. **Opis aplikacji webowej**

**a) Opis funkcjonalności**

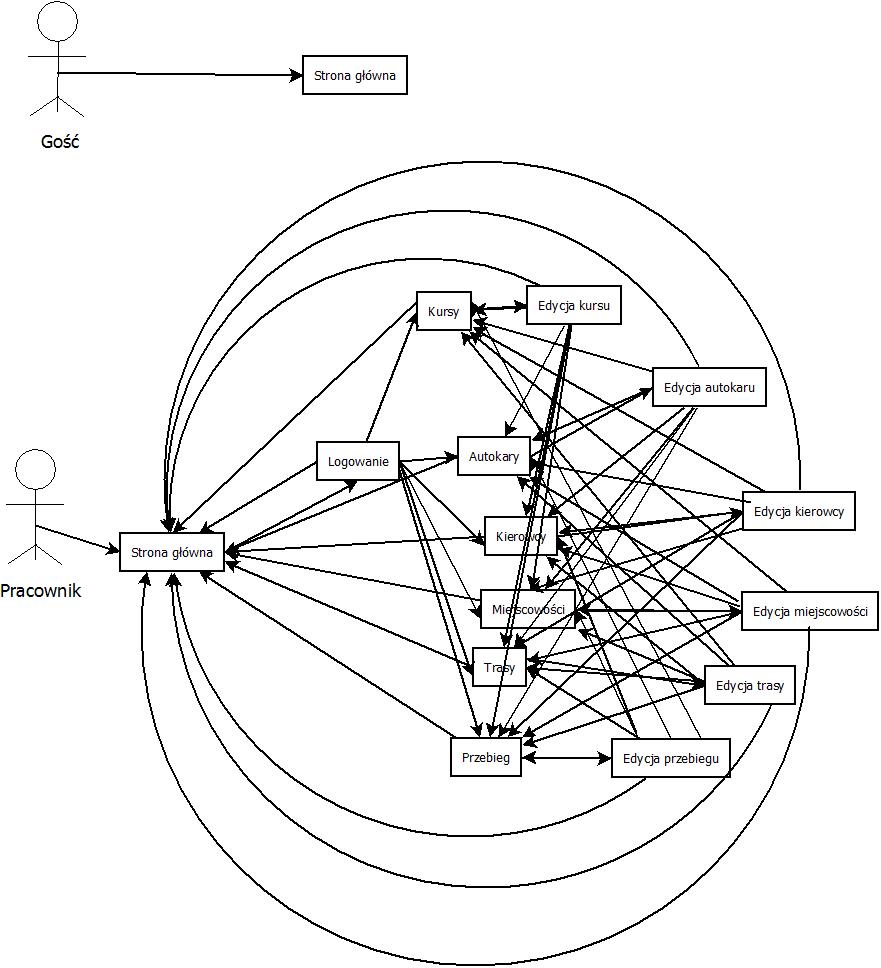
Aplikacja webowa udostępnia następujące funkcjonalności:

* Przeglądanie połączeń autokarów, po uprzednim podaniu dodatkowych parametrów;
* Administrowanie autokarami (usuwanie, edycja, dodawanie) przez upoważnionego pracownika;
* Wyszukiwanie przypisanych kierowców do wybranego autokaru;
* Administrowanie kierowcami (usuwanie, edycja, dodawanie) przez upoważnionego pracownika;
* Wyszukiwanie kierowców, którzy są w aktualnej chwili w trasie;
* Administrowanie trasami (edycja, usuwanie) przez upoważnionego pracownika;
* Administrowanie miejscowościami (edycja, usuwanie) przez upoważnionego pracownika;
* Administrowanie przebiegami pomiędzy przystankami (edycja, usuwanie) przez upoważnionego pracownika;
* Administrowanie kursami (edycja, usuwanie) przez upoważnionego pracownika;
* Generowanie raportu tygodniowego związanego z liczbą pasażerów danego kursu;
* Generowanie raportu liczącego średnią liczbę pasażerów od początku istnienia danego kursu

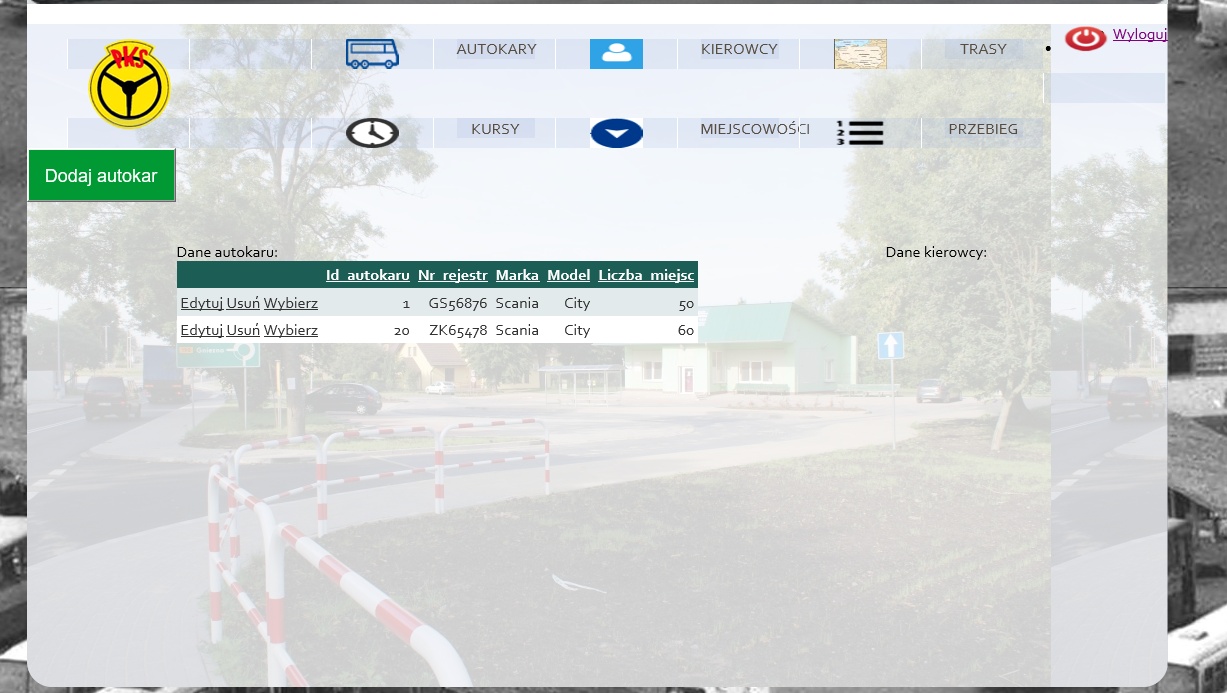
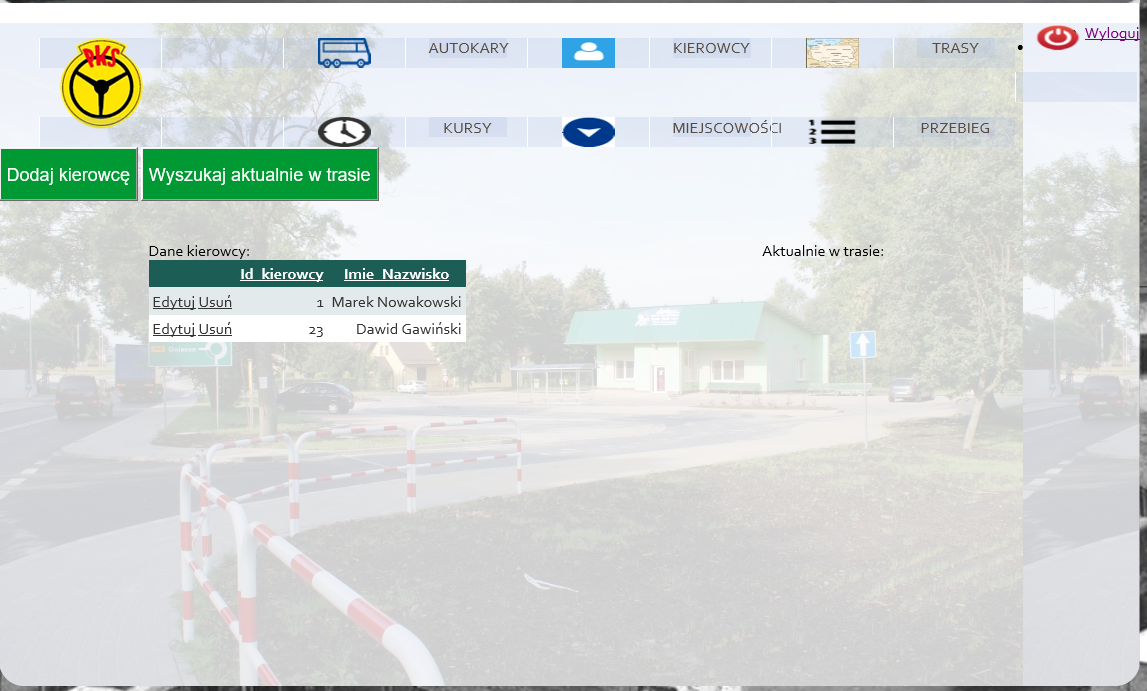
**b) Wybrana technologia**

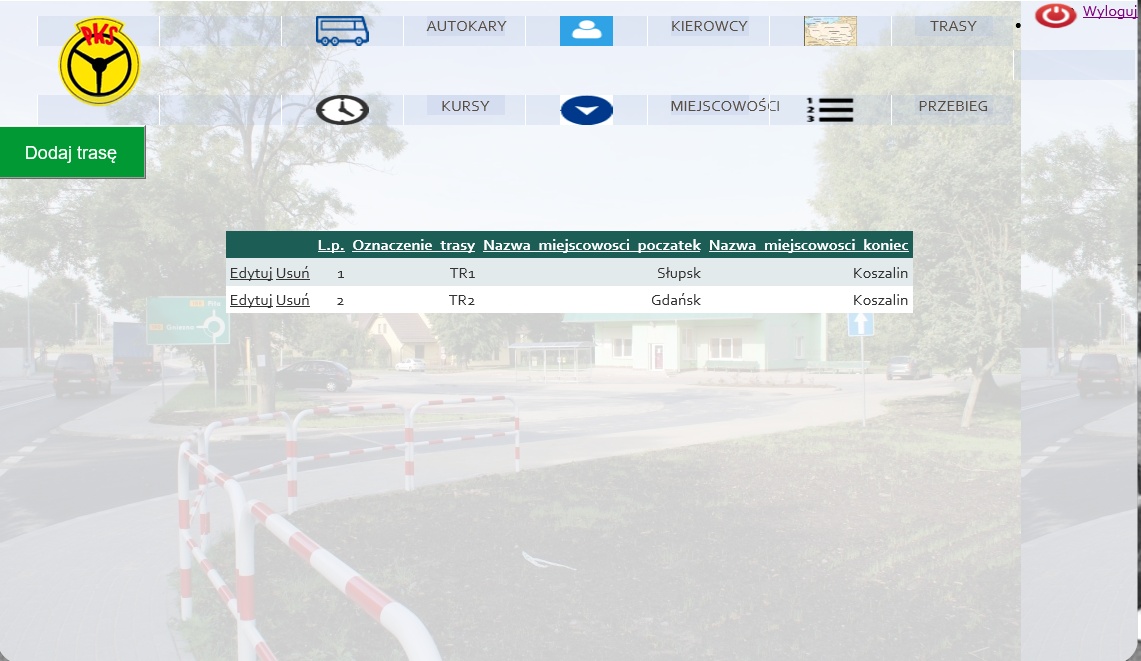
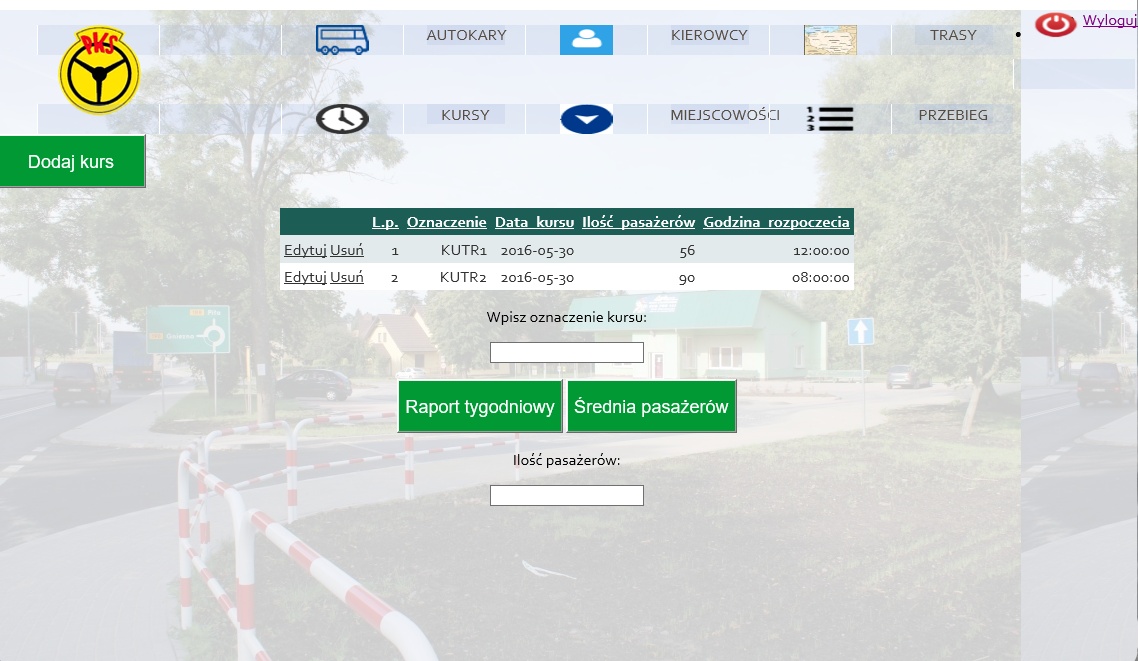
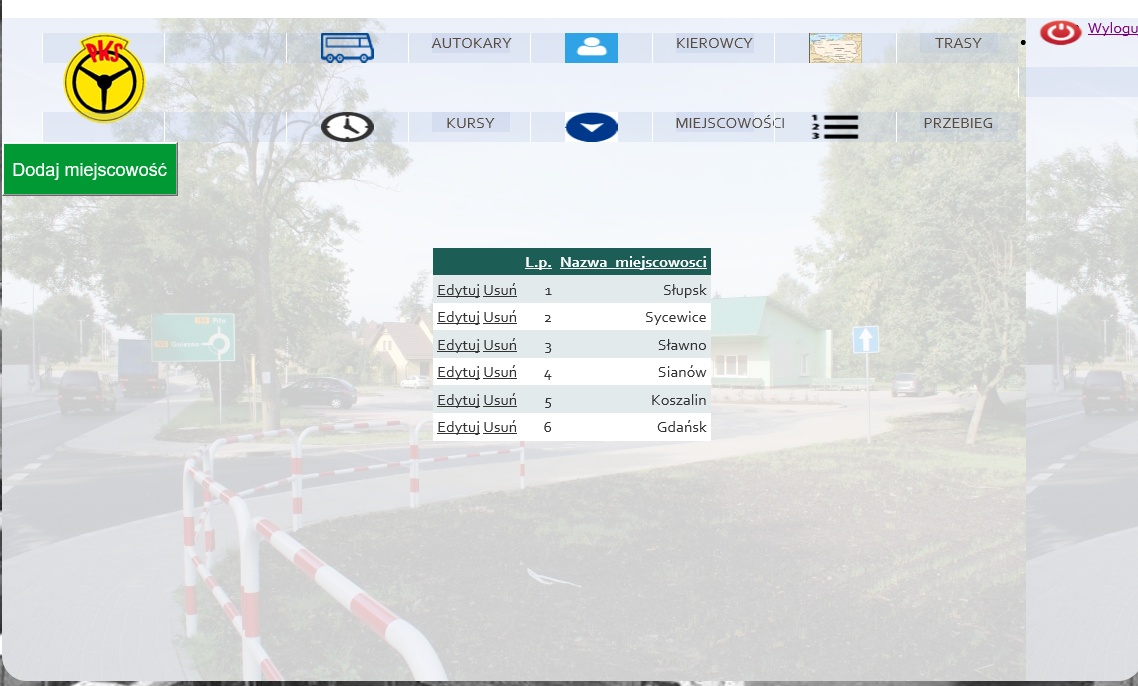
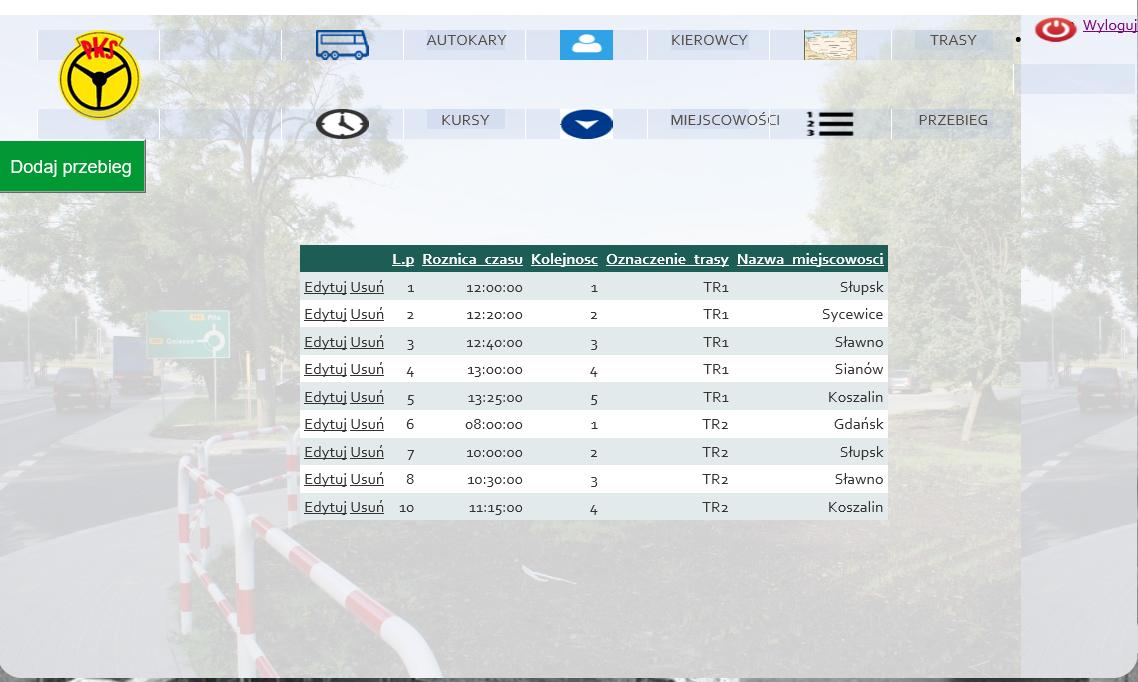
Aplikacja została zrealizowana w asp.net WebForms przy użyciu środowiska Visual Studio 2015

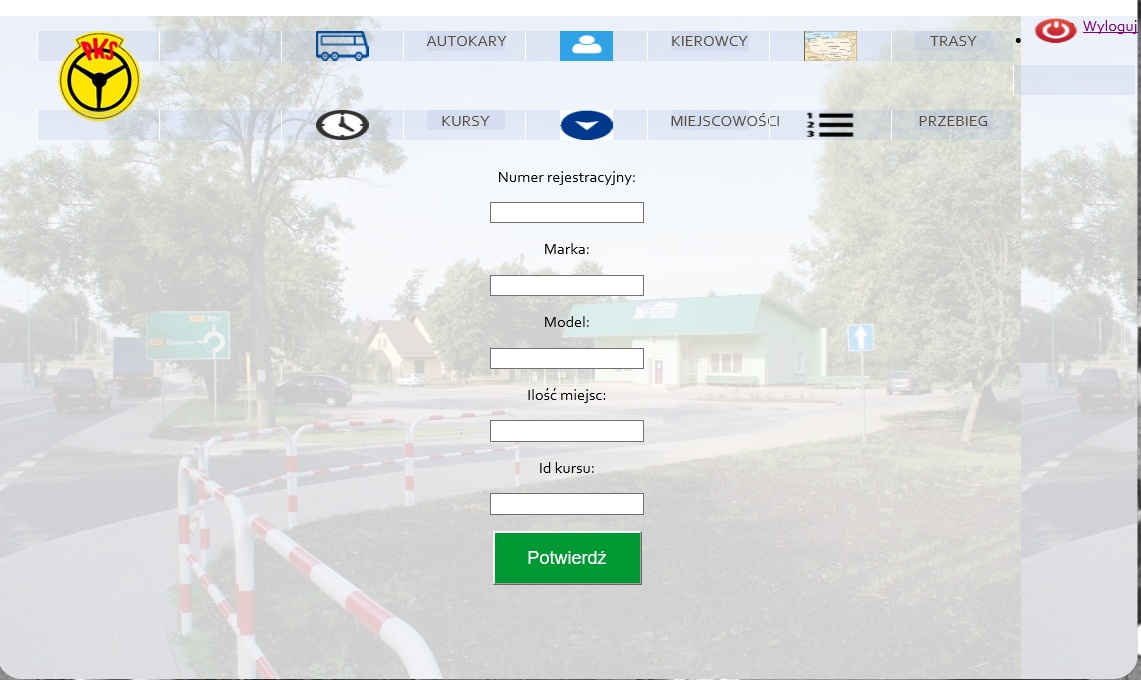
**c) Graf przejść**



**d) Opis struktury programu**







**10. Wnioski**

Poprawne zaprojektowanie bazy danych umożliwiło mi w sposób łatwy napisanie aplikacji lokalnej i internetowej.

Moja baza danych została postawiona na serwerze Microsoft SQL Server 2014. Wcześniej poznaliśmy Sybase, który według mnie jest o wiele gorszy. Microsoft SQL Server pozwala nam w prostszy sposób zarządzać danymi i je przeglądać.

Program PowerDesigner w bardzo łatwy sposób pozwala nam zaprojektować model konceptualny naszej bazy oraz pozwala wygenerować nam z niego model fizyczny i gotowy skrypt SQL. W przypadku Sybase skrypt nie zawsze wykonywany jest w 100%, natomiast w przypadku Microsoft SQL Server skrypt wykonywany jest poprawnie, a więc ułatwia nam to pracę.

Aplikacja lokalna napisana w języku C# do poprawnego działania potrzebowała dołączenia odpowiedniej biblioteki do połączenia z bazą danych. Prace przy pisaniu aplikacji przebiegały bez problemu, wcześniej napisane kwerendy działały poprawnie.

Technologia ASP.NET jest bardzo dobrą technologią do tworzenia aplikacji internetowych. Pozwala na bardzo proste zbudowanie szkieletu strony oraz zarządzanie użytkownikami zalogowanymi i niezalogowanymi. Technologia posiada także role, którymi możemy nadawać odpowiednie uprawnienia danym użytkownikom.

Aplikacja internetowa napisana w technologii asp.net WebForms była znacznie łatwiejsza do zaimplementowania niż ta w C#. Poprawne napisanie kontrolerów umożliwiało mi wygenerowanie potrzebnych widoków, takich jak okna dodawania nowych wierszy do bazy czy wyświetlania encji. Dzięki narzuconemu z góry frameworkowi kod był znacznie lepiej zorganizowany, każdą funkcjonalność można o wiele szybciej odszukać niż tą w C#. Utworzenie aplikacji w ASP.NET przy użyciu WebForms sprawiło, że wszelkie potrzebne biblioteki do połączenia z bazą danych (entity framework) zostały zainstalowane automatycznie przy tworzeniu projektu.

Według mojej oceny napisanie aplikacji w ASP.NET było znacznie łatwiejsze niż napisanie jej w C#. Język ten jest zdecydowanie bardziej intuicyjny, a zaimplementowane w nim rozwiązania sprawiają, że niektóre rzeczy można wykonać szybciej, lepiej i łatwiej.