

JĘZYKI INTERNETOWE

LABORATORIUM nr 1

Temat: Wprowadzenie do podejścia *CodeFirst* z zastosowaniem *Entity Framework 6* oraz *MVC 5*

Laboratorium zawiera opis początkowego etapu tworzenia przykładowej aplikacji internetowej, która będzie umożliwiać przechowywanie informacji o przykładowego Uniwersytetu. Zostanie wykorzystana technologia *ASP.NET* (ang. *Active Server Pages .NET*), w połączeniu z wzorcem *MVC 5* (ang. *Model-View-Controller*). Ponadto zakłada się zastosowanie podejścia *CodeFirst*, poprzez wykorzystanie *Entity Framework ver. 6*.

Laboratorium obejmuje m.in.:

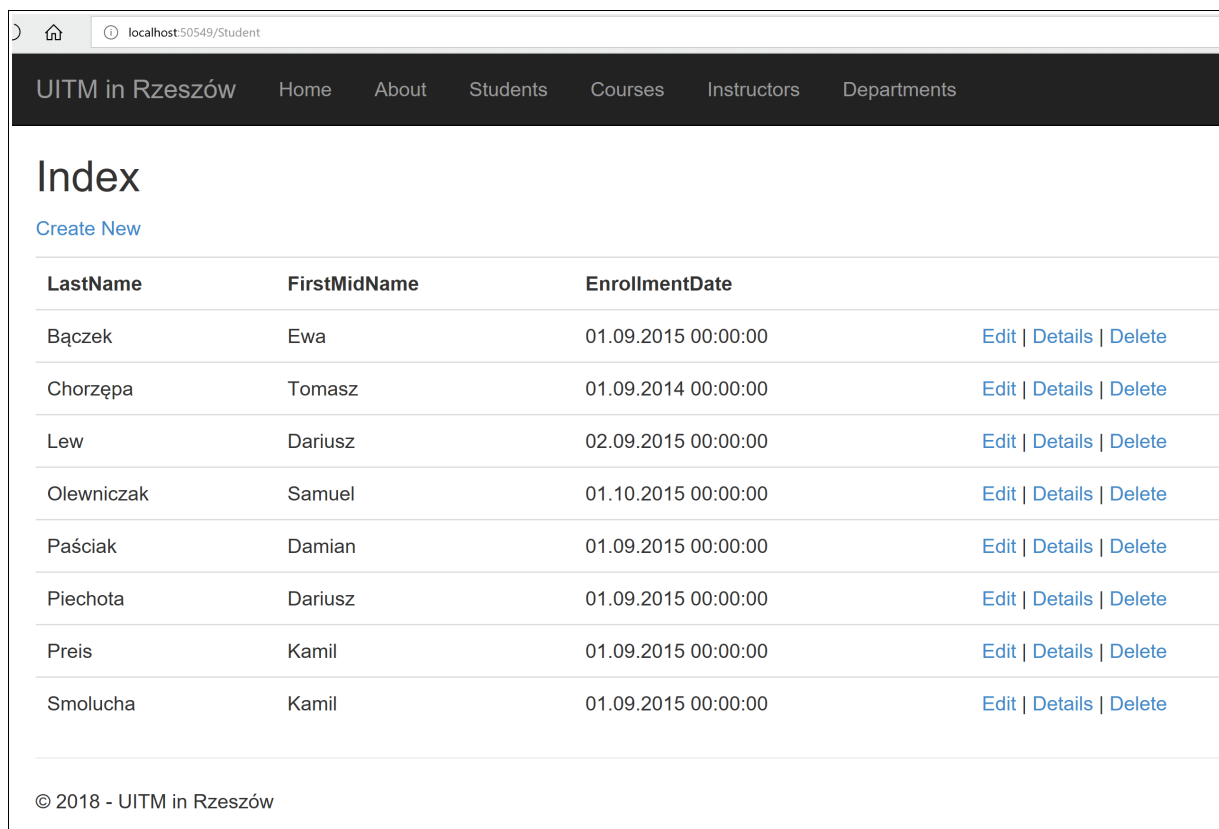
- omówienie założeń początkowych tworzonej aplikacji internetowej,
- utworzenie nowego projektu przy wykorzystaniu domyślnego wzorca *ASP.NET MVC 5*,
- instalację *Entity Framework* w wersji *6*,
- utworzenie pierwszego (prostego) modelu danych, umożliwiającego przypisanie studentów do wybranych zajęć,
- utworzenie kontekstu dostępu do bazy danych,
- zapis do bazy danych przykładowych rekordów, oraz
- utworzenie kontrolera oraz widoków do wyświetlania, edycji oraz usuwania studentów.

Tworzona aplikacja web'owa – od pierwszego do ostatniego laboratorium – będzie wykonywana przy użyciu:

- zintegrowanego środowiska programistycznego Microsoft Visual Studio 2015, oraz
- serwera bazodanowego *Microsoft SQL Server 2014*.

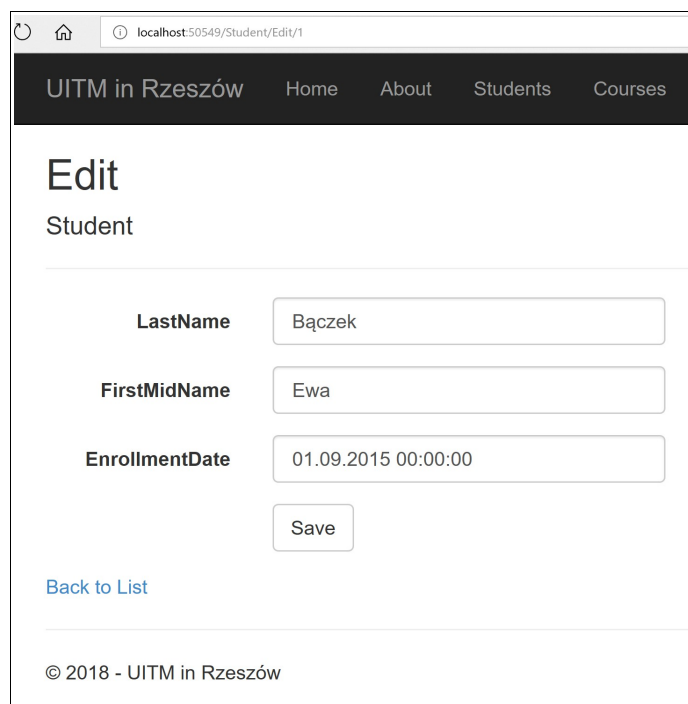
Samodzielne wykonanie zadań z laboratorium będzie możliwe w domu z zastosowaniem (niekomercyjnym) np. środowiska programistycznego *Microsoft Visual Studio 2017 Community* oraz serwera bazodanowego *Microsoft SQL Server 2016 Express*.

Przykładowy wygląd aplikacji do wykonania na laboratorium



| UITM in Rzeszów | | | |
|---|--------------|---------------------|---|
| Home About Students Courses Instructors Departments | | | |
| Index | | | |
| Create New | | | |
| LastName | FirstMidName | EnrollmentDate | |
| Bączek | Ewa | 01.09.2015 00:00:00 | Edit Details Delete |
| Chorzępa | Tomasz | 01.09.2014 00:00:00 | Edit Details Delete |
| Lew | Dariusz | 02.09.2015 00:00:00 | Edit Details Delete |
| Olewniczak | Samuel | 01.10.2015 00:00:00 | Edit Details Delete |
| Paściak | Damian | 01.09.2015 00:00:00 | Edit Details Delete |
| Piechota | Dariusz | 01.09.2015 00:00:00 | Edit Details Delete |
| Preis | Kamil | 01.09.2015 00:00:00 | Edit Details Delete |
| Smolucha | Kamil | 01.09.2015 00:00:00 | Edit Details Delete |
| © 2018 - UITM in Rzeszów | | | |

Rys.1. Wyświetlanie listy wszystkich studentów



| UITM in Rzeszów | |
|------------------------------|--|
| Home About Students Courses | |
| Edit Student | |
| LastName | <input type="text" value="Bączek"/> |
| FirstMidName | <input type="text" value="Ewa"/> |
| EnrollmentDate | <input type="text" value="01.09.2015 00:00:00"/> |
| | <input type="button" value="Save"/> |
| Back to List | |
| © 2018 - UITM in Rzeszów | |

Rys.2. Panel edycji danych studenta

Wprowadzenie do wzorca MVC

Wzorzec **MVC** (ang. *Model-View-Controller*) może być postrzegany pod kątem (i) wzorca architektonicznego oraz/lub (ii) wzorca projektowego. Mianowicie, pod pojęciem wzorca architektonicznego należy rozumieć wielowarstwową architekturę aplikacji, natomiast wzorzec projektowy oznacza zdarzeniowy charakter (model) aplikacji. Z zastosowania wzorca MVC w procesie tworzenia aplikacji wynikają następujące cechy:

- systematyzacja kodu oraz określenie przejrzystej struktury programu,
- rozdzielenie warstwy zawierającej dane oraz logikę biznesową od warstwy prezentacji, oraz
- z uwagi na systematyzację kodu ułatwione jest jego testowanie.

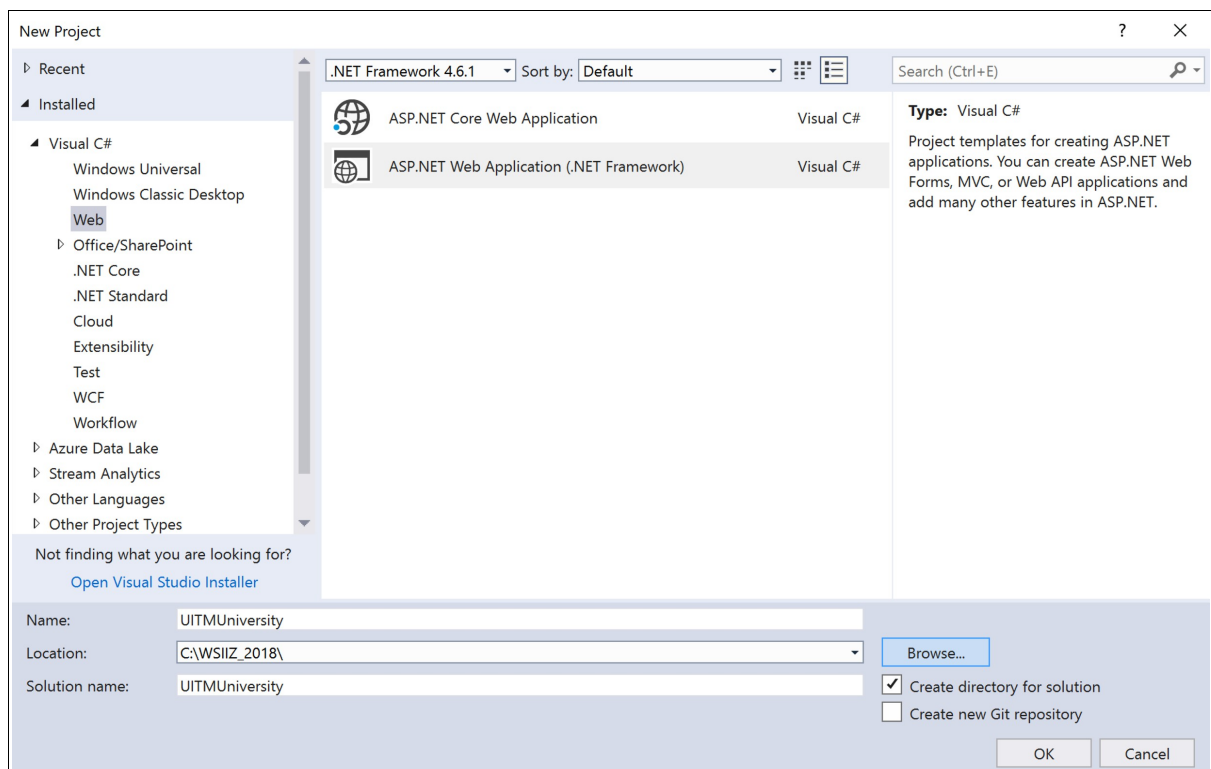
Model oznacza niejako obiekty (klasy), w tym pod kątem (i) implementacji logiki biznesowej związanej z zachowaniem stanu obiektu (np. w pliku, w bazie danych, etc.) oraz (ii) właściwości umożliwiających odzwierciedlenie ich rzeczywistego znaczenia.

Widok jest komponentem odpowiedzialnym za tworzenie interfejsu użytkownika w oparciu o przekazane dane modelu, czyli widok umożliwia wizualizację danych zapisanych w modelu.

Kontroler obejmuje komponenty odpowiedzialne za reakcje na akcje wykonywane przez użytkownika. W odpowiedzi na działania wykonywane przez użytkownika kontrolery dokonują operacji na modelu oraz wybierają i tworzą widok, umożliwiając wizualizację efektu działań użytkownika.

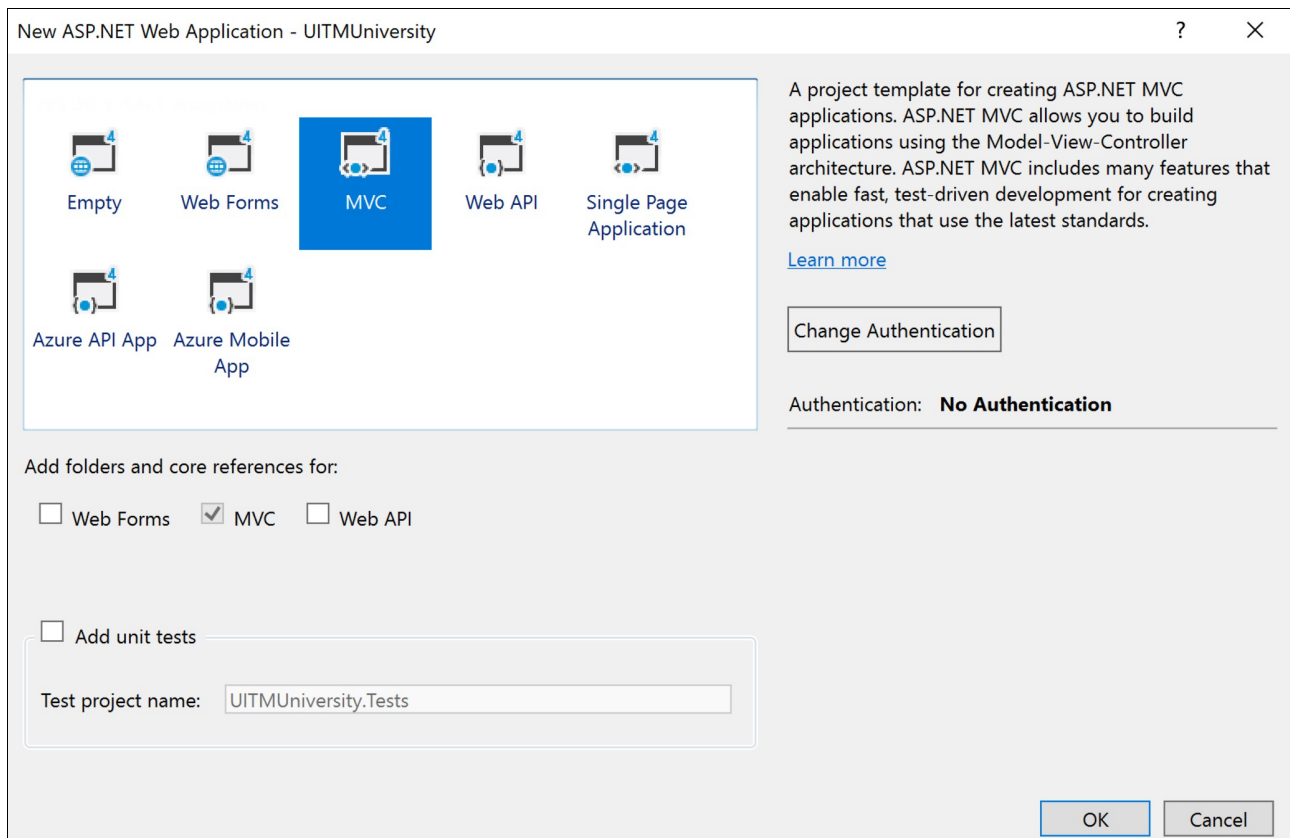
Tworzenie aplikacji ASP.NET MVC

Otworzyć program **Microsoft Visual Studio 2015**, a następnie utworzyć nowy projekt (typu **Visual C#** → **Web**) o nazwie **UITMUniversity** (Rys.3).

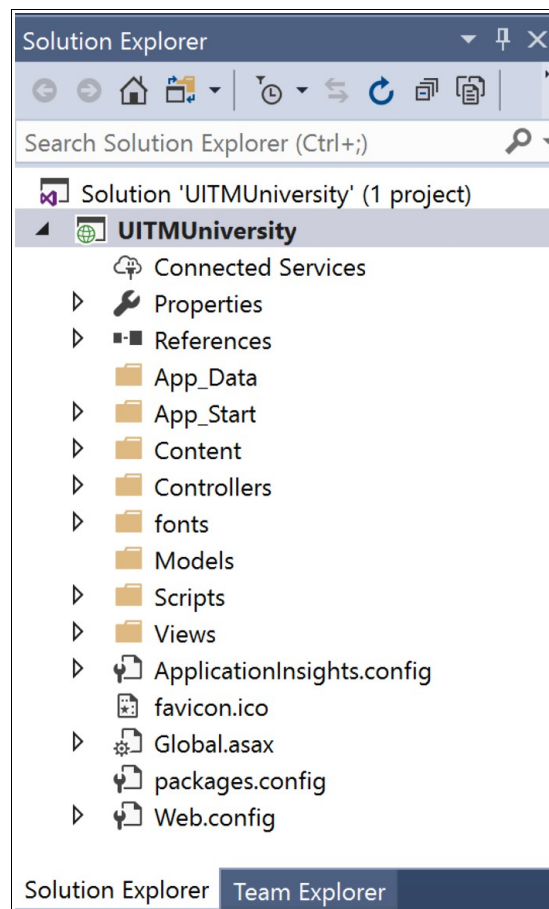


Rys.3. Okno tworzenia aplikacji webowej

W kolejnym oknie (Rys. 4) wybrać zastosowanie wzorca **MVC** oraz zmienić typ uwierzytelniania wybierając opcję **No Authentication**.

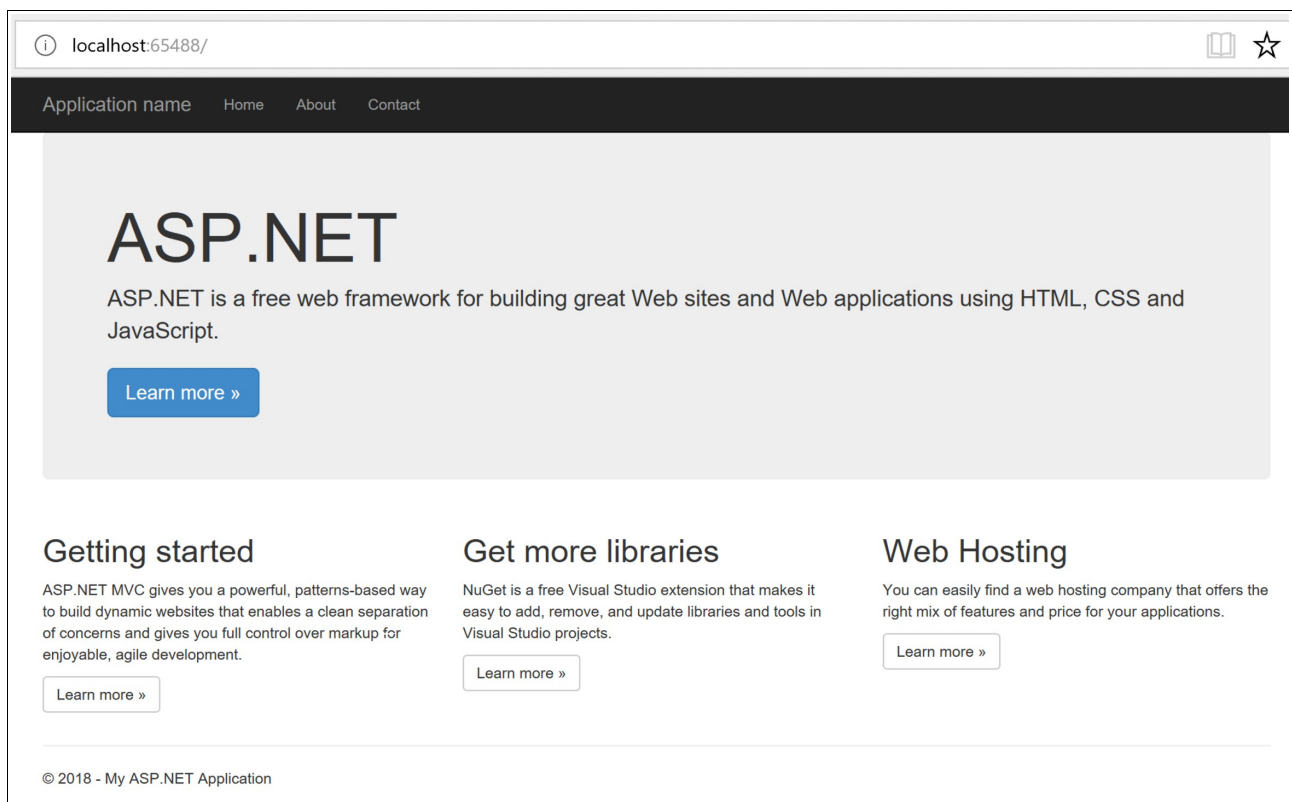


Rys.4. Okno *New ASP.NET Project*

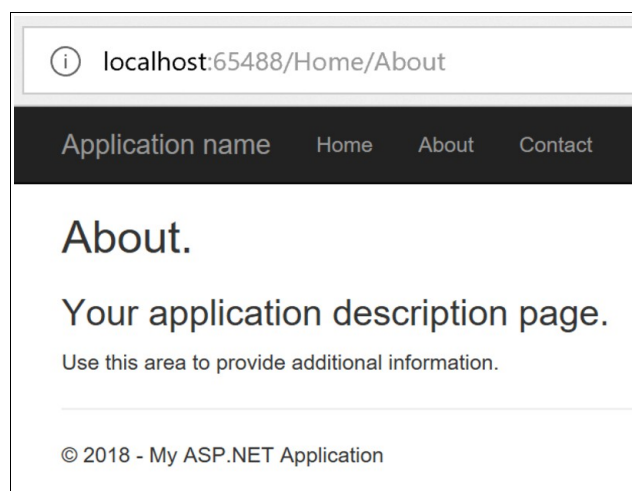


Rys.5. Domyślna struktura folderów aplikacji web w oknie *Solution Explorer*'a

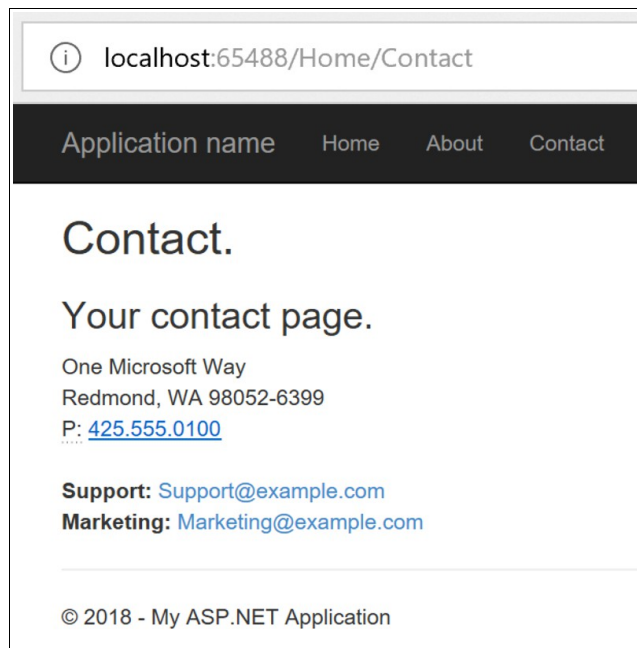
Uruchomić projekt (**Ctrl + F5**) (patrz Rys.6-8).



Rys.6. Strona *Index* domyślnej aplikacji



Rys.7. Strona *About* domyślnej aplikacji



Rys.8. Strona *Contact* domyślnej aplikacji

Zmienić wygląd template'ki, poprzez edycję pliku *_Layout.cshtml* (z katalogu */Views/Shared*).

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - UITM in Rzeszów</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-
toggle="collapse" data-target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("UITM in Rzeszów", "Index", "Home", new { area
= "" }, new { @class = "navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("About", "About", "Home")</li>
                    <li>@Html.ActionLink("Students", "Index", "Student")</li>
                    <li>@Html.ActionLink("Courses", "Index", "Course")</li>
                    <li>@Html.ActionLink("Instructors", "Index",
"Instructor")</li>
                    <li>@Html.ActionLink("Departments", "Index",
"Department")</li>
                </ul>
            </div>
        </div>
    </div>
    <div class="container body-content">
```

```

        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - UITM in Rzeszów</p>
        </footer>
    </div>

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>

```

Zmienić plik ***Index.cshtml*** (*/Views/Home*).

```

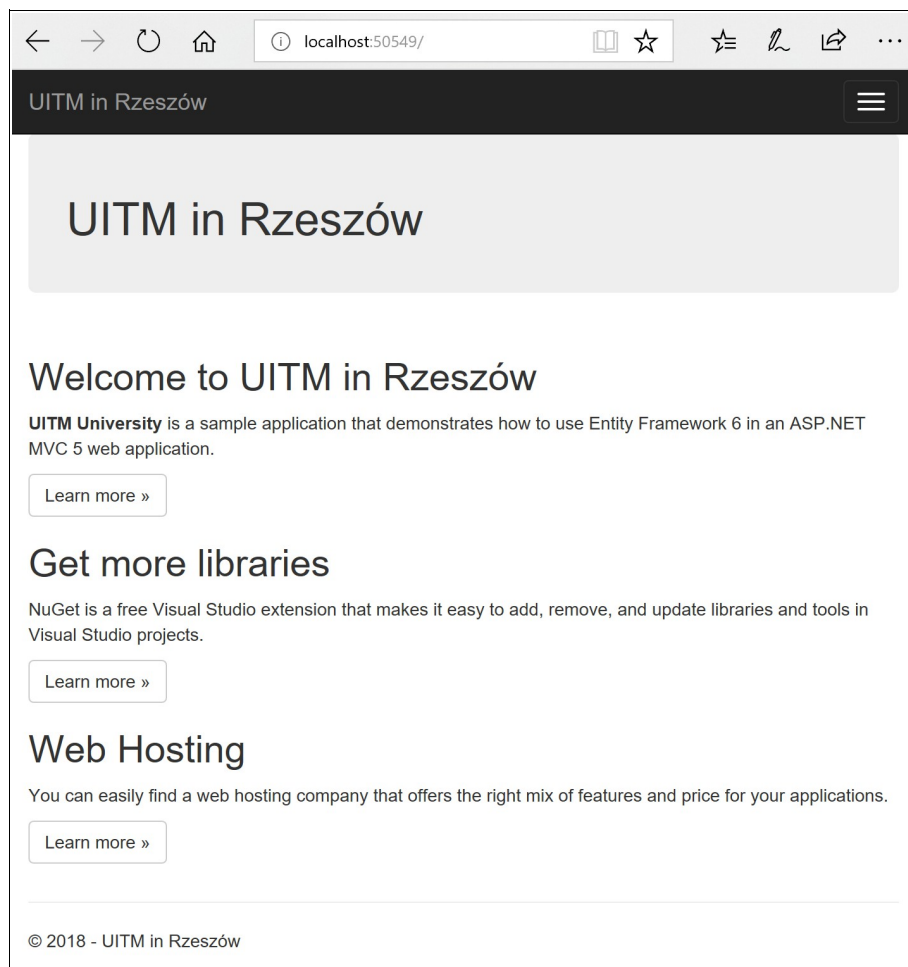
@{
    ViewBag.Title = "Home Page";
}

<div class="jumbotron">
    <h1>UITM in Rzeszów</h1>
</div>

<div class="row">
    <div class="col-md-4">
        <h2>Welcome to UITM in Rzeszów</h2>
        <p>
            <b>UITM in Rzeszów</b> is a sample application that
            demonstrates how to use Entity Framework 6 in an
            ASP.NET MVC 5 web application.
        </p>
        <p><a class="btn btn-default" href="http://www.wsiz.rzeszow.pl">Learn
more &raquo;</a></p>
    </div>
    <div class="col-md-4">
        <h2>Get more libraries</h2>
        <p>NuGet is a free Visual Studio extension that makes it easy to add,
remove, and update libraries and tools in Visual Studio projects.</p>
        <p><a class="btn btn-default" href="https://go.microsoft.com/fwlink/?
LinkId=301866">Learn more &raquo;</a></p>
    </div>
    <div class="col-md-4">
        <h2>Web Hosting</h2>
        <p>You can easily find a web hosting company that offers the right mix
of features and price for your applications.</p>
        <p><a class="btn btn-default" href="https://go.microsoft.com/fwlink/?
LinkId=301867">Learn more &raquo;</a></p>
    </div>
</div>

```

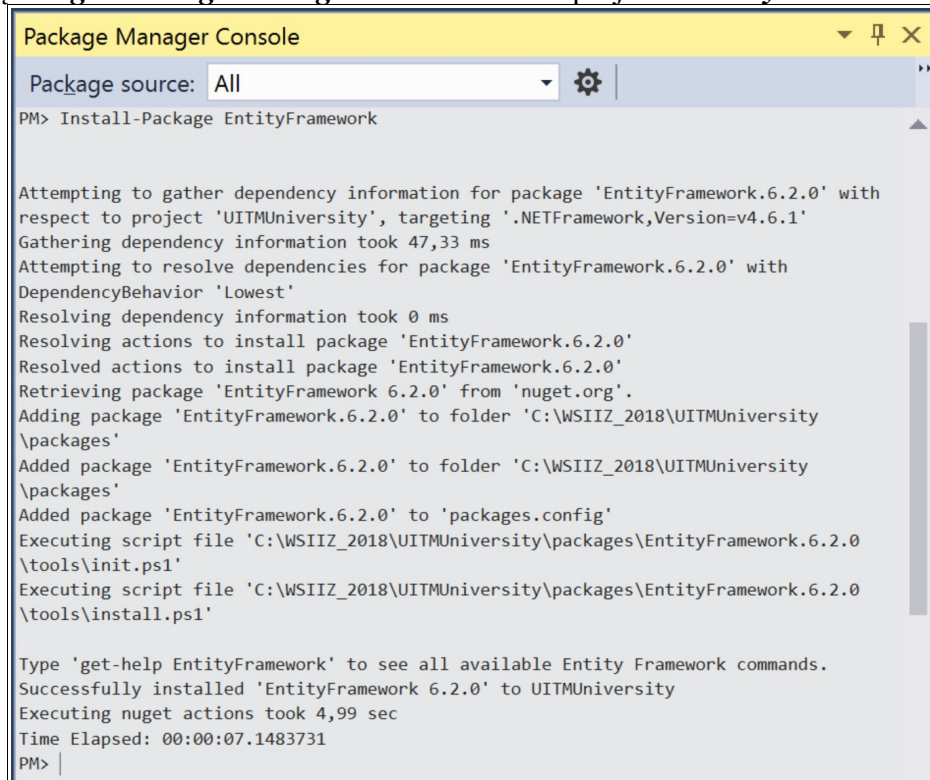
Ponownie uruchomić aplikację.



Rys.9. Wygląd strony *Index* po modyfikacji layout'u oraz pliku *Index.cshtml*

Instalacja Entity Framework

Wykorzystując *Nuget Package Manager* zainstalować w projekcie *Entity Framework*.



```
Package Manager Console
Package source: All
PM> Install-Package EntityFramework

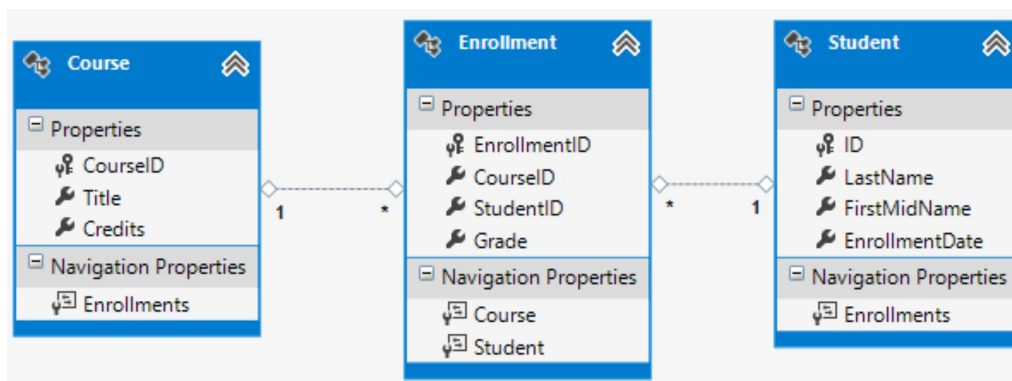
Attempting to gather dependency information for package 'EntityFramework.6.2.0' with
respect to project 'UITMUniversity', targeting '.NETFramework,Version=v4.6.1'
Gathering dependency information took 47,33 ms
Attempting to resolve dependencies for package 'EntityFramework.6.2.0' with
DependencyBehavior 'Lowest'
Resolving dependency information took 0 ms
Resolving actions to install package 'EntityFramework.6.2.0'
Resolved actions to install package 'EntityFramework.6.2.0'
Retrieving package 'EntityFramework 6.2.0' from 'nuget.org'.
Adding package 'EntityFramework.6.2.0' to folder 'C:\WSIIZ_2018\UITMUniversity
\packages'
Added package 'EntityFramework.6.2.0' to folder 'C:\WSIIZ_2018\UITMUniversity
\packages'
Added package 'EntityFramework.6.2.0' to 'packages.config'
Executing script file 'C:\WSIIZ_2018\UITMUniversity\packages\EntityFramework.6.2.0
\tools\init.ps1'
Executing script file 'C:\WSIIZ_2018\UITMUniversity\packages\EntityFramework.6.2.0
\tools\install.ps1'

Type 'get-help EntityFramework' to see all available Entity Framework commands.
Successfully installed 'EntityFramework 6.2.0' to UITMUniversity
Executing nuget actions took 4,99 sec
Time Elapsed: 00:00:07.1483731
PM>
```

Rys.10. Instalacja EntityFramework z wykorzystaniem *Package Manager Console*

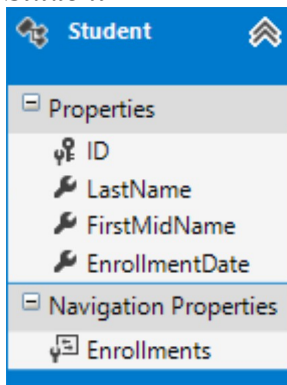
Tworzenie modelu danych

Utworzyć nieskomplikowany model danych, zawierający informacje o studentach i wybranych przez nich kursach (Rys.9).



Rys.11. Początkowy model danych

Student



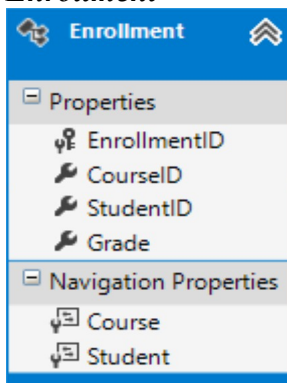
Klasa ***Student*** - w folderze ***Models*** utworzyć plik z klasą ***Student.cs***, po czym uzupełnić plik o kod zamieszczony poniżej:

```
using System;
using System.Collections.Generic;

namespace UITMUniversity.Models
{
    public class Student
    {
        public int ID { get; set; }
        public string LastName { get; set; }
        public string FirstMidName { get; set; }
        public DateTime EnrollmentDate { get; set; }

        public virtual ICollection<Enrollment> Enrollments
        { get; set; }
    }
}
```

Enrollment



Klasa ***Enrollment*** - w folderze ***Models*** utworzyć plik z klasą ***Enrollment.cs***, po czym uzupełnić plik o kod zamieszczony poniżej:

```
namespace UITMUniversity.Models
{
    public enum Grade
    {
        A, B, C, D, F
    }

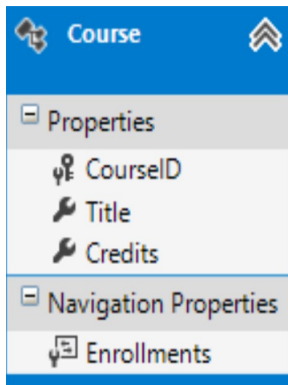
    public class Enrollment
    {
        public int EnrollmentID { get; set; }
        public int CourseID { get; set; }
        public int StudentID { get; set; }
        public Grade? Grade { get; set; }
    }
}
```

```

        public virtual Course Course { get; set; }
        public virtual Student Student { get; set; }
    }
}

```

Course



Klasa ***Course*** - w folderze ***Models*** utworzyć plik z klasą ***Course.cs***, po czym uzupełnić plik o kod zamieszczony poniżej:

```

using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;

namespace UITMUniversity.Models
{
    public class Course
    {
        [DatabaseGenerated(DatabaseGeneratedOption.None)]
        public int CourseID { get; set; }
        public string Title { get; set; }
        public int Credits { get; set; }

        public virtual ICollection<Enrollment> Enrollments { get; set; }
    }
}

```

Utworzenie kontekstu bazy danych

W projekcie dodać nowy folder **DAL** (*Data Access Layer*), po czym utworzyć w nim plik **UniversityContext.cs**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data.Entity;
using System.Data.Entity.ModelConfiguration.Conventions;
using UITMUniversity.Models;

namespace UITMUniversity.DAL
{
    public class UniversityContext : DbContext
    {
        public UniversityContext() : base("UniversityContext")
        {
        }

        public DbSet<Student> Students { get; set; }
        public DbSet<Enrollment> Enrollments { get; set; }
        public DbSet<Course> Courses { get; set; }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();
        }
    }
}
```

Wstawienie danych testowych do bazy danych

Do folderu **DAL** dodać kolejny plik (**UniversityInitializer.cs**), który spowoduje dodanie do bazy przykładowych danych po uruchomieniu aplikacji.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data.Entity;
using UITMUniversity.Models;

namespace UITMUniversity.DAL
{
    public class UniversityInitializer :
        System.Data.Entity.DropCreateDatabaseIfModelChanges<UniversityContext>
    {
        protected override void Seed(UniversityContext context)
        {
            var students = new List<Student>
            {
                new
                Student{FirstMidName="Ewa", LastName="Bączek", EnrollmentDate=DateTime.Parse("2015-09-01")},
                new
                Student{FirstMidName="Tomasz", LastName="Chorzępa", EnrollmentDate=DateTime.Parse
```

```

("2014-09-01")),
    new
Student{FirstMidName="Dariusz",LastName="Lew",EnrollmentDate=DateTime.Parse("20
13-09-01")},
    new
Student{FirstMidName="Samuel",LastName="Olewniczak",EnrollmentDate=DateTime.Par
se("2012-09-01")},
    new
Student{FirstMidName="Damian",LastName="Paściak",EnrollmentDate=DateTime.Parse(
"2012-09-01")},
    new
Student{FirstMidName="Dariusz",LastName="Piechota",EnrollmentDate=DateTime.Pars
e("2015-09-01")},
    new
Student{FirstMidName="Kamil",LastName="Preis",EnrollmentDate=DateTime.Parse("20
13-09-01")},
    new
Student{FirstMidName="Kamil",LastName="Smółucha",EnrollmentDate=DateTime.Parse(
"2015-09-01")}
};

students.ForEach(s => context.Students.Add(s));
context.SaveChanges();
var courses = new List<Course>
{
    new Course{CourseID=1050,Title="Chemistry",Credits=3},
    new Course{CourseID=4022,Title="Microeconomics",Credits=3},
    new Course{CourseID=4041,Title="Macroeconomics",Credits=3},
    new Course{CourseID=1045,Title="Calculus",Credits=4},
    new Course{CourseID=3141,Title="Trigonometry",Credits=4},
    new Course{CourseID=2021,Title="Composition",Credits=3},
    new Course{CourseID=2042,Title="Literature",Credits=4}
};
courses.ForEach(s => context.Courses.Add(s));
context.SaveChanges();
var enrollments = new List<Enrollment>
{
    new Enrollment{StudentID=1,CourseID=1050,Grade=Grade.A},
    new Enrollment{StudentID=1,CourseID=4022,Grade=Grade.C},
    new Enrollment{StudentID=1,CourseID=4041,Grade=Grade.B},
    new Enrollment{StudentID=2,CourseID=1045,Grade=Grade.B},
    new Enrollment{StudentID=2,CourseID=3141,Grade=Grade.F},
    new Enrollment{StudentID=2,CourseID=2021,Grade=Grade.F},
    new Enrollment{StudentID=3,CourseID=1050},
    new Enrollment{StudentID=4,CourseID=1050},
    new Enrollment{StudentID=4,CourseID=4022,Grade=Grade.F},
    new Enrollment{StudentID=5,CourseID=4041,Grade=Grade.C},
    new Enrollment{StudentID=6,CourseID=1045},
    new Enrollment{StudentID=7,CourseID=3141,Grade=Grade.A},
};
enrollments.ForEach(s => context.Enrollments.Add(s));
context.SaveChanges();
}
}
}

```

Ustawienia EF do utworzenia bazy danych typu LocalDB

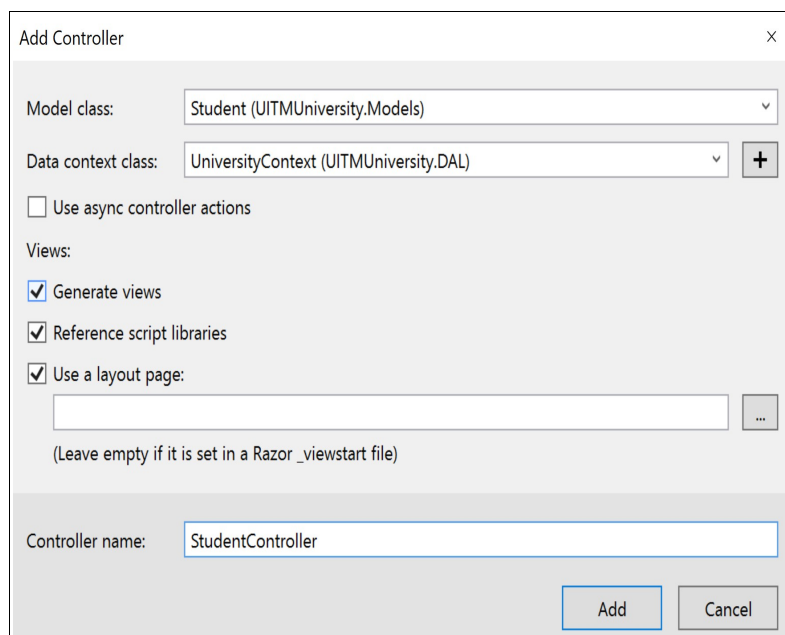
W pliku **Web.config** uzupełnić następujące wpisy

```
<entityFramework>
  <contexts>
    <context type="UITMUniversity.DAL.UniversityContext, UITMUniversity">
      <databaseInitializer type="UITMUniversity.DAL.UniversityInitializer,
UITMUniversity" />
    </context>
  </contexts>
  <defaultConnectionFactory
type="System.Data.Entity.Infrastructure.SqlConnectionFactory,
EntityFramework" />
  <providers>
    <provider invariantName="System.Data.SqlClient"
type="System.Data.Entity.SqlServer.SqlProviderServices,
EntityFramework.SqlServer" />
  </providers>
</entityFramework>
```

```
<connectionStrings>
  <add name="UniversityContext" connectionString="Data
Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=UITMUniversity1;Integrated
Security=SSPI;" providerName="System.Data.SqlClient"/>
</connectionStrings>
<appSettings>
  <add key="webpages:Version" value="3.0.0.0" />
  <add key="webpages:Enabled" value="false" />
  <add key="ClientValidationEnabled" value="true" />
  <add key="UnobtrusiveJavaScriptEnabled" value="true" />
</appSettings>
```

Kontroler oraz widoki dla studentów

Utworzyć kontroler *StudentController* (oraz widoki – dodawane automatycznie) dla studentów (patrz Rys.12).



Add Controller

Model class: Student (UITMUniversity.Models)

Data context class: UniversityContext (UITMUniversity.DAL)

☐ Use async controller actions

Views:

☒ Generate views

☒ Reference script libraries

☒ Use a layout page:

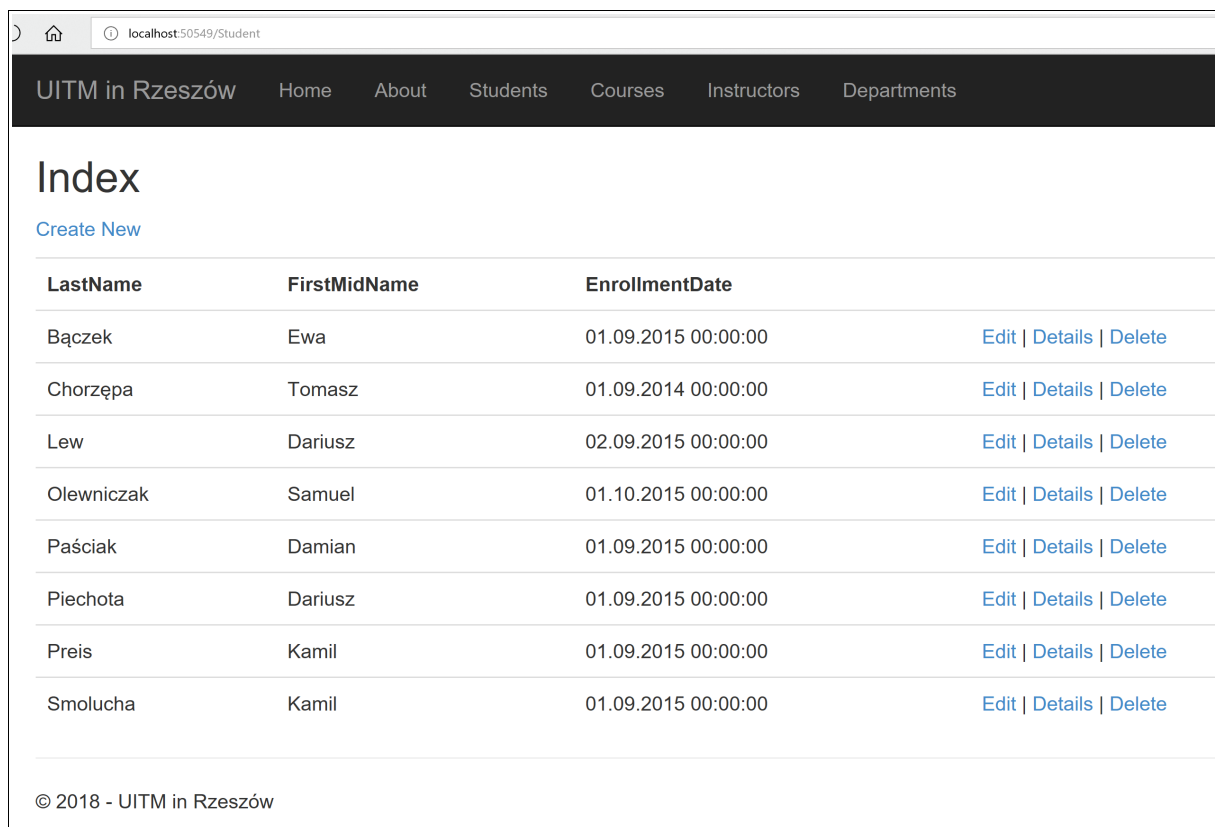
(Leave empty if it is set in a Razor _viewstart file)

Controller name: StudentController

Add Cancel

Rys.12. Kontroler dla klasy *Student*

Skompilować i uruchomić ponownie aplikację, po czym (i) wyświetlić listę studentów (Rys.13) oraz (ii) spróbować edytować dane wybranej osoby (Rys.14).



| LastName | FirstMidName | EnrollmentDate | |
|------------|--------------|---------------------|---|
| Bączek | Ewa | 01.09.2015 00:00:00 | Edit Details Delete |
| Chorzępa | Tomasz | 01.09.2014 00:00:00 | Edit Details Delete |
| Lew | Dariusz | 02.09.2015 00:00:00 | Edit Details Delete |
| Olewniczak | Samuel | 01.10.2015 00:00:00 | Edit Details Delete |
| Paściak | Damian | 01.09.2015 00:00:00 | Edit Details Delete |
| Piechota | Dariusz | 01.09.2015 00:00:00 | Edit Details Delete |
| Preis | Kamil | 01.09.2015 00:00:00 | Edit Details Delete |
| Smolucha | Kamil | 01.09.2015 00:00:00 | Edit Details Delete |

© 2018 - UITM in Rzeszów

Rys.13. Wyświetlanie listy studentów z bazy danych

UITM in Rzeszów Home About Students Courses

Edit

Student

LastName

FirstMidName

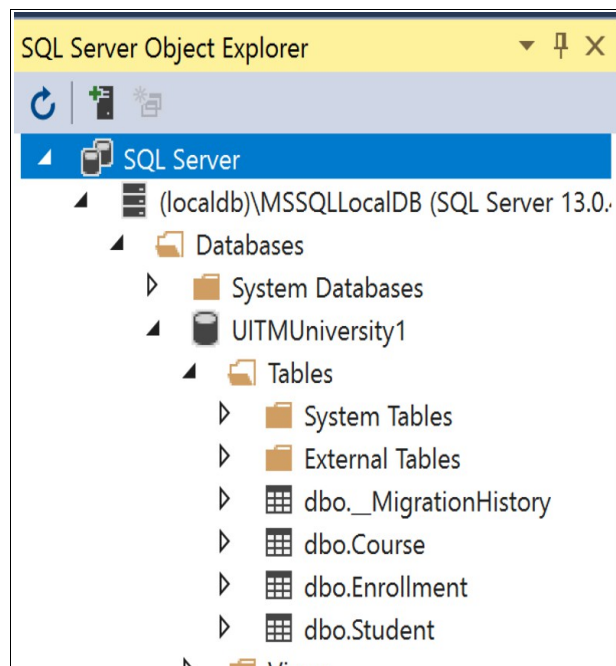
EnrollmentDate

[Back to List](#)

© 2018 - UITM in Rzeszów

Rys.14. Edycja danych studenta

Korzystając z okna **SQL Server Object Explorer**'a zobaczyć, że baza danych została utworzona (Rys.15) oraz przejrzeć dane zapisane w tabeli Student (Rys.16).



Rys.15. Struktura bazy danych w **SQL Server Object Explorer**

| dbo.Student [Data] | | StudentController.cs | Web.config | |
|--------------------|------|----------------------|--------------|---------------------|
| | | Max Rows: | 1000 | |
| | ID | LastName | FirstMidN... | EnrollmentDate |
| ▶ | 1 | Bączek | Ewa | 01.09.2015 00:00:00 |
| | 2 | Chorzępa | Tomasz | 01.09.2014 00:00:00 |
| | 3 | Lew | Dariusz | 01.09.2013 00:00:00 |
| | 4 | Olewniczak | Samuel | 01.09.2012 00:00:00 |
| | 5 | Paściak | Damian | 01.09.2012 00:00:00 |
| | 6 | Piechota | Dariusz | 01.09.2015 00:00:00 |
| | 7 | Preis | Kamil | 01.09.2013 00:00:00 |
| | 8 | Smołucha | Kamil | 01.09.2015 00:00:00 |
| * | NULL | NULL | NULL | NULL |

Rys.16. Zawartość tabeli *Student*