Politechnika Łódzka

Wydział Elektrotechniki, Elektroniki, Informatyki i Automatyki

Instytut Informatyki Stosowanej

**PRACA DYPLOMOWA INŻYNIERSKA**

System do wspomagania zarządzania dydaktyką w jednostce naukowej

System supporting the teaching management process in a scientific institution

Wojciech Pełka

187783

Opiekun / opiekunowie pracy:  
Dr. Inż. Radosław Adamus

Łódź, Luty, 2017

**Spis Treści**

[1. Wstęp 4](#_Toc474086190)

[1.1. Problematyka pracy 4](#_Toc474086191)

[1.2. Cel pracy 4](#_Toc474086192)

[1.3. Przegląd literatury 4](#_Toc474086193)

[1.4. Zakres pracy 5](#_Toc474086194)

[1.5. Układ pracy 5](#_Toc474086195)

[2. Wymagania funkcjonalne systemu 6](#_Toc474086196)

[3. Użyte technologie przy rozwiązywaniu postawionego problemu 7](#_Toc474086197)

[3.1. Dlaczego .Net Framework i C# ? 7](#_Toc474086198)

[3.2. CLR – Common Language Runtime 9](#_Toc474086200)

[3.3. C# 11](#_Toc474086201)

[3.3.1. Wstęp 11](#_Toc474086202)

[3.3.2. Charakterystyka według ECMA 11](#_Toc474086203)

[3.3.3. Zmiany na przestrzeni wersji 12](#_Toc474086204)

[3.3.4. Składnia 12](#_Toc474086205)

[3.4. Microsoft Visual Studio 2015 (IDE) 13](#_Toc474086206)

[3.5. ASP.NET 14](#_Toc474086207)

[3.5.1. Wstęp 14](#_Toc474086208)

[3.5.2. IIS – Internet Information Service 15](#_Toc474086209)

[3.5.3. Razor 16](#_Toc474086210)

[3.5.4. ASP.NET MVC 16](#_Toc474086211)

[3.6. Microsoft SQL Server 17](#_Toc474086212)

[3.6.1. Wstęp 17](#_Toc474086213)

[3.6.2. SQL Server Management Studio 18](#_Toc474086214)

[3.7. ADO.NET 18](#_Toc474086215)

[3.7.1. Wstęp 18](#_Toc474086216)

[3.7.2. Entity Framework 19](#_Toc474086217)

[3.7.3. LINQ 21](#_Toc474086218)

[4. Projekt Systemu 23](#_Toc474086219)

[4.1. Baza Danych 23](#_Toc474086220)

[4.2. Struktura systemu 30](#_Toc474086221)

[4.2.1. Wstęp 30](#_Toc474086222)

[4.2.2. Model 30](#_Toc474086223)

[4.2.3. Widok 32](#_Toc474086224)

[4.2.4. Kontroler 32](#_Toc474086225)

[5. Implementacja kluczowych rozwiązań 33](#_Toc474086226)

[6. Podsumowanie 33](#_Toc474086227)

[7. Bibliografia 33](#_Toc474086228)

# Wstęp

## Problematyka pracy

Problem przedstawionego tematu pracy jest związany z procesem rozliczania godzin pracowników Politechniki Łódzkiej. Pracownicy prowadzący zajęcia dydaktyczne muszą być odpowiednio rozliczani ze swojej pracy. Każdy ma przypisaną określoną liczbę godzin na semestr, prowadzi prace dyplomowe, laboratoria czy wykłady. Każda taka aktywność odpowiada określonej liczbie godzin dydaktycznych. Pomimo przypisania pracownikowi wszystkich godzin na początku semestru, w trakcie jego trwania zajęcia mogą się zmieniać, co wpływa na zmianę przydziału godzin. Może to być konieczność utworzenia nowej grupy laboratoryjnej lub poprowadzenie zajęć w ramach zastępstwa za inną osobę. Obecnie wszystkie te zmiany w stosunku do pierwotnego planu są przesyłane do osoby tworzącej podsumowania i rozliczenia. Robi się to po ukończeniu semestru i zazwyczaj jest kilka iteracji zanim wszystkie niejaności zostaną rozwiązane.

## Cel pracy

Obecnie cały proces wspomniany w punkcie 1.1 Problematyka pracy nie jest zautomatyozwany przez co jest monotonny i trwa bardzo długo zanim osiągnie się zamierzony cel.

Celem pracy, który pozwoli rozwiązać przedstawiony problem jest zaprojektowanie i zaimplementowanie systemu z interfejsem webowym. System taki zautomatyzuje cały proces pozwalając na bierząco zarządzać i kontrolować godziny pracowników. System wykryje ewentualne niespójności i poinformuje o nich osobę, która jest odpowiedzialna za tworzenie raportów końcowych.

## Przegląd literatury

Niezbędnym elementem zarówno przy tworzeniu części pisemnej jak i praktycznej pracy jest literatura, która dostarcza niezbędną wiedzę pozwalącą stworzyć i opisać tworzoną aplikację. Wszystkie źródła zostały wymienione w punkcie 7. Bilbiografia. Bliżej należy przyjrzeć się dwóm pierwszym pozycją. Są to źródła drukowane. Pierwsza pozycja opisuje dokładnie platformę .Net jak i język C# oraz wszelkie elementy niezbędne przy tworzeniu każdej aplikacji jak np dostęp do bazy danych. Druga pozycja opisuje dokładnie platforma ASP.NET MVC, czyli platformę programistyczną do tworzenia stron internetowych w oparciu o platofrmę .NET i język C#.  
Pozostałe elementy to źródła internetowe, które stanowiły uzupełnienie tematów opisanych w wyżej wymienionych książkach.

## Zakres pracy

W Celu wykonania projektu należało podjąć następujące kroki

* Ustalenie wymagań systemu
* Skonfigurowanie środowiska programistycznego
* Skonfigurowanie serwera
* Wykonanie projektu bazy danych
* Utworzenie bazy danych
* Wypełnienie bazy danych danymi
* Zaprojektowanie interfejsu użytkownika
* Implementacja Interfejsu użytkownika
* Implementacja poszczególnych funkcjonalności
  + Logowanie/Rejestracja użytkowników
  + Implementacja systemu autentykacji użytkowników
  + Implementacja panelu pracownika i administratora
  + Zaprojektowanie systemu powiadomień
  + Implementacja systemu powiadomień
  + Implementacja systemu kontroli błędów

## Układ pracy

Zawartość pracy dyplomowej inżynierskiej przedstawia się następująco :

1. **Wstęp** – W tym rozdziale opisany został problem postawiony w pracy inżynierskiej, jego rozwiązanie oraz wykaz literatury wraz z zakresem i układem pracy.

2. **Wymaganie funkcjonalne systemu** – W tym rozdziale opisane zostały wymagania funkcjonalne systemu, jakie muszą być spełnione aby system mógł zostać zaakceptowany.

3. **Użyte technologie przy rozwiązywaniu postawionego problemu** – Rozdział ten stanowi część, technologiczną pracy. Została tutaj opisana platforma .Net wraz z językiem C# oraz wykorzystaniem go przy projektowaniu stron internetowych. W rozdziale zawarta jest dokładna specyfikacja języka jak i przedstawiony powód dla którego platforma jak i język zostały użyte przy projektowaniu systemu.

4. **Projekt systemu** – W tym rozdziale został opisany system oraz baza danych od strony implementacji, struktury oraz funkcjonowania.

5. **Implementacja kluczowych rozwiązań** – W tym rozdziale została opisana kluczowa część systemu czyli system powiadomień i komunikacji użytkownik – administrator.

6. **Podsumowanie** – W rozdziale tym zostało napisane krótkie podsumowanie pracy.

7. **Bibliografia** – Rozdział ten stanowi wykaz literatury wykorzystanej przy tworzeniu części pisemnej jak i praktycznej pracy dyplomowej.

# Wymagania funkcjonalne systemu

Aby system został zaakceptowany musi spełniać następujące wymagania

* System musi udostępniać funkcjonalność logowania się do systemu
* System musi udostępniać możliwość rejestracji użytkownika.
* System musi udostępniać możliwość przeglądania przypisanych przedmiotów do pracownika zarówno pracownikowi jak i administratorowi systemu
* System musi udostępniać możliwość dodawania i usuwania przypisanych przedmiotów oraz wprowadzania zmian w aktulany stan przypisania przedmiotów oraz godzin pracownika.
* System musi udostępniać wewnętrzny system powiadomień. Powiadomienia reprezentują prośby pracownika o zaakceptowanie wprowadzonych przez niego zmian.
* System musi udostępniać możliwość akceptowania bądź anulowania przez administratora systemu zmian jakie pracownik wprowadza w swoim przypisaniu godzin.
* System musi udostępniać pracownikowi możliwość wglądu w listę próśb wysłanych do administratora związanych z aktualnie wprowadzonymi zmianami wraz z możliwośćią anulowania wysłanej prośby o taką zmianę.

# Użyte technologie przy rozwiązywaniu postawionego problemu

## Dlaczego .Net Framework i C# ?

Aby opisać zalety, które przemawiają za użyciem platformy ASP.NET MVC do tworzenia aplikacji webowej (strony internetowej), należy powiedzieć ogólnie dlaczego używać technologii .Net i języka C#, zamiast innej konkurencyjnej technologii dostępnej na rynku. Język C# oraz platforma .Net mogą być używane nie tylko do tworzenia aplikacji internetowych, ale Również aplikacji typu desktop, systemów komputerowych, sterowników, gier i wielu innych.

Pierwszym elementem nad którym należy się zastanowić przed dokonaniem wybory technologii jest to, co tak na prawdę chcemy zaprojektować i zaimplementować. Kolejnym bardzo ważnym czynnikiem wyboru jest to, co lubimy w językach programowania i w jakich aspektach programowania czujemy się najlepiej. .NET Framework i język C# dostarczają potężne i elastyczne narzędzia programistyczne pracując jednocześnie na wysokim poziomie abstrakcji. Dzięki temu programista może skupić się na implementacji rozwiązań bezspośrednio związanych z problemem nie marnując czasu na projektowaniu algorytmów pobocznych problemów. Problemy takie pojawiają się w trakcie pracy programisty nad różnymi aspektami aplikacji. Problemy te w języku C# zostały rozwiązane i rozwiązania te zostały udostępnione w postaci narzędzi i bibliotek. Narzędzia te dodatkowo są proste tak samo w użyciu jak i w zrozumieniu.

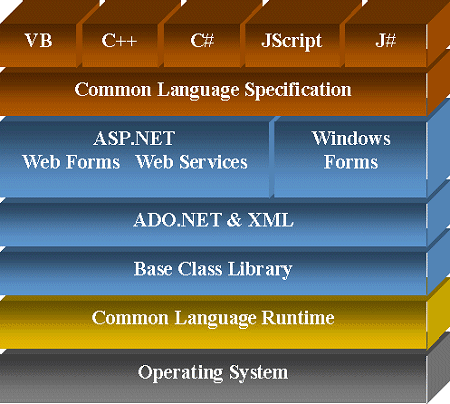
Znaczna ilość zalet języka C# bierze się ze wspieranych przez ten język technik programistycznych takich jak: elementy zorientowane obiektowo, generyczne oraz umożliwienie programowania funkcyjnego. Dodatkowo język wspiera zarówno typowanie statyczne jak i dynamiczne. C# dostarcza poteżne operacje na listach i zbiorach danych dzięki zintegrowanemu językowi zapytań (LINQ). Najnowsze wersje języka wporwadzają dodatkowo obsługę dla programowania asynchronicznego.

Jedne z najważnieszych zalet z używania języka C# wiążą sie z jego środowiskiem uruchomieniowym. Środowisko uruchomieniowe czyli inaczej CLR (Common Language Runtime) umożliwia sprawdzanie typów w trakcie działania programu, obsługę wyjątków, zarządzanie wątkami oraz automatyczne zarządzanie pamięcią (Garbage Collector), które zwalnia programistę z konieczności ręcznego zwalniania pamięci dla utworzonych obiektów.

W całej liście języków programowania jakie powstały wraz z rozwojem tej dziedziny znajdziemy wiele języków, które są doskonałe do momentu gdy musimy wykonać podstawową czynność. Operacje takie jak komunikacja z bazą danych, lub jak przechować ustawienia użytkownika w optymalny sposób C# rozwiązuje bardzo wydajnie.

Niezależnie od tego jak silne wyrażenia programistyczne dany język dostarcza, Również musi dostarczyć przystępny dla programisty pełny dostęp do serwisów różnych innych platform i dzięki .Net Framework, język C# jest na tym polu językiem przodującym. [1]

Poniższe zdjęcie przedstawia architekture platformy .Net, którą stanowią 4 niebieskie „pudełka” reprezentujące ASP.NET, Windows Forms, ADO.NET, XML wraz z podkomponentami.



Rys. 1 Architektura platformy .Net

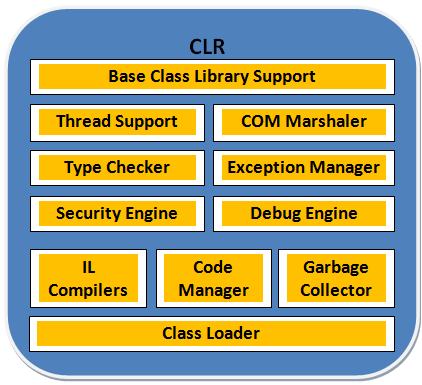
## CLR – Common Language Runtime

Jak zostało wspomniane w rodziale 3.1 zalety języka C# biorą się m.in. ze jego środowiska uruchomieniowgo czyli CLR (eng. Common Language Runtime).

Kierując się nazwą (common – wspólny) można wywnioskować, że CLR nie tylko wspiera język C#, ale każdy język należący do platformy .Net. Języki, które mogą wchodzić w skład platformy .Net to Visual Basic, F#, C++ (rozszerzony na potrzeby pracy w środowisku .Net), oraz open-sourcowe implementacje Ruby oraz Python które mogą pracować w .Net Framework (IronPython, IronRuby).

CLR posiada system CTS (eng. Common Type System), który umożliwa współprace różnych języków programowania ze sobą. Oznacza to, że biblioteki .Net są wspólne dla wszystkich języków wchodzących w skład tej platformy. Programy napisane w języku F# mogą używać bibliotek napisanych w języku C#, a biblioteki napisane w C# mogą korzystać z programów napisanych w języku Visual Basic itd.

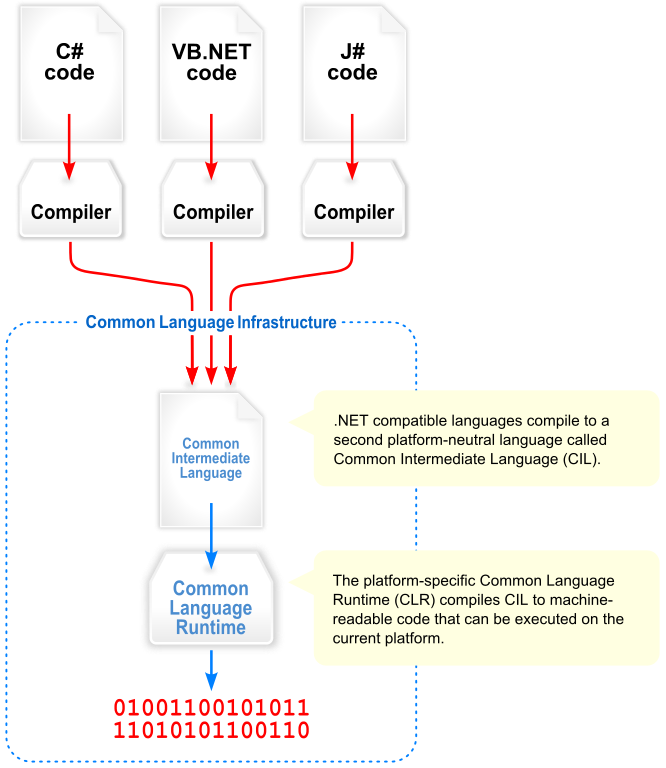
Na poniższym rysunku zamieszczona zostala infrastruktura CLR.



Rys 2. Infrastruktura CLR – Common Langauge Runtime

CLR jest jednym z elementów infrastruktury CLI (ang. Common Language Infrastructure). CLI jest to specyfikacja techniczna, która opisuje kod wykonywalny CIL (ang. Common Intermediate Langaute) oraz środowisko uruchomieniowe. Środowisko to kompiluje wspólny kod wykonywalny CIL do kodu maszynowego. To właśnie kod CIL pozwala na współpracę wielu języków ze sobą jako że każdy z tych języków jest do niego kompilowany.

Poniższy rysunek przedstawia infrastrykurę Platformy .Net, uwzględniając zależności pomiędzy wspomnianymi wcześniej elementami platformy czyli CLI, CIL, CLR, oraz to jak języki należące do platformy .Net współpracują razem, będąc tłumaczone na wspólny język CIL.



Rys 3: Infrastruktura CLI – Common Language Infrastructure [3]

## C#

### Wstęp

C# jest to wieloparadygmatowy język programowania obejmujący takie paradygmaty programowania jak : silne typowanie, programowanie imperatywne, deklaratywne, funkcyjne, generyczne, obiektowe, komponentowe. Został wymyślony przez firmę Microsoft w 2000 roku wraz z platformą .Net której miałbyć głównym językiem. Następnie został zaakceptowany przez standardy ECMA i ISO. Język C# jest jednym z języków stworzonych dla wspomnianego wyżej CLI (ang. Common Language Infrastructure).

Zespół który zajmuje się rozwojem tego języka jest prowadzony przez Anders’a Hejlsberga. Najnowszą wersją języka jest C# 6.0 wypuszczona w 2015 roku.

### Charakterystyka według ECMA

* Język C# został zaprojektowany tak aby był prosty, nowoczesny, uniwersalny i zorientowany obiektowo.
* Język i jego implementacja powinny dostarczać wsparcie dla celów inżynierskich takich jak : kontrola silnego typowania, sprawdzanie zakresu obiektów, sprawdzanie nieużuwanych zmiennych, i automatyczne zarządzanie pamięcią.
* Język jest przeznaczony do użycia przy programowaniu komponentów systemów które będą wdrążane na różne środowiska.
* Język C# musi zapewnić przenośność co jest jest bardzo ważne dla kodu źródłowego oraz programistów, w szczególności tych którzy są zapoznani z językiem C oraz C++.
* Język C# zapewnia wsparcie dla internacjonalizacji.
* Język C# był zaprojektowany tak aby możliwe było jego użycie przy pisaniu aplikacji serwerowych jak i systemów wbudowanych, od prostych do bardzo skomplikowanych.
* Aplikacje napisane w języku C# są ekonomiczne pod względem zużycia pamięci, jednak nie było zamiarem aby język ten dorównał pod tym względem językowi C oraz Assembler.

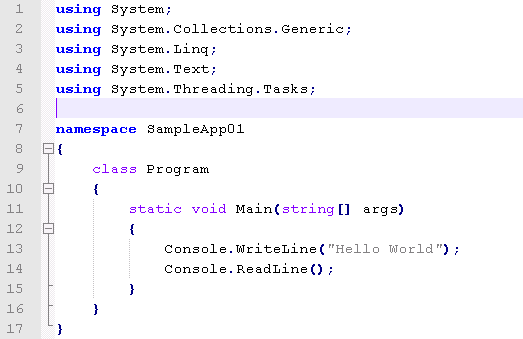
### Zmiany na przestrzeni wersji

|  |  |  |
| --- | --- | --- |
| Wersja C# | Data | Główne Zmiany |
| 2.0 | Listopad 2005 | Typy generyczne, Typy częściowe, metody anonimowe, iteratory, typy nullowalne, delegaty, klasy statyczne, |
| 3.0 | Listopad 2007 | Inicjalizatory obiektów i kolekcji, typy anonimowe, metody rozszerzeń, wyrażenia zapytań, drzewa zapytań, metody częściowe |
| 4.0 | Kwiecień 2010 | Dynamiczne wiązanie, argumenty domyślne, kowariancja i kontrawariancja |
| 5.0 | Sierpień 2012 | Metody asynchroniczne |
| 6.0 | Lipiec 2015 | Kompilator jako serwis, await w bloku try catch, inicjalizator słowników, statyczne importowanie obiektow w przestrzeniach nazw |
| 7.0 (future) | Future | Literały binarne, separatory liczbowe, funkcje lokalne, krotki, out var, |

### Składnia

Składnia języka C# jest zbliżona do składni języka C takich jak C++ czy Java. W szczególności :

* Średniki są używane na zakończenie instrukcji
* Klamry służą do grupowania instrukcji. Instrukcje są grupowane do metod, metody do klas a klasy do przestrzeni nazw.
* Przypisanie do zmiennej następuje za pomocą pojedynczego znaku równości, porównanie następuje za pomocą podwójnego znaku równości.
* Nawiasy kwadratowe są wykorzystywane w tablicach, zarówno do deklarowania ich jak i pobierani wartości.



Rys. 4 : Składnia języka C#

## Microsoft Visual Studio 2015 (IDE)

Visual Studio jest zintegrowanym środowiskiem programistycznym (ang. Integrated Development Environment). Istnieje wiele wersji oprogramowania poczynając od darmowej do użytku prywatnego do bardzo drogiej dla firm do użytku komercyjnego. Wszystkie wersje dostarczają podstawowych funkcjonalnosci niezbędnych do pracy progamisty czyli : edytor tekstowy, narzędzia do budowania aplikacji, debugger oraz narzędzia do tworzenia interfejsu użytkownika.

Każdy większy projekt będzie miał wiele plików kodu źródłowego, które w Visual Studio będą należeć do *projektu*. Wynikiem końcowym projektu po zbudowaniu go może być pojedynczy plik wykonywalny jak np pliki .exe. bądź wiele plików w zależności od rodzaju projektu. Niektóre typy projektów zwracają strony internetowe po zbudowaniu. Każdy projekt będzie zbudowany jako jedna jednostka, wiele plików serwisu internetowego bedzie zgrupowanych do jednostki jaką jest strona internetowa.

Pliki projektu zazwyczaj mają rozszerzenie .*proj* w zależności od języka. Dla przykładu projekty C# mają rozszerzenie .*csproj* natomiast projekt C++ używają rozszerzenia .*vcxproj*. Jeśli otworzymy te pliki za pomocą dowolnego edytora tekstowego odkryjemy, że pliki te to tak na prawdę pliki o formacie XML. Pliki te określają zawartość projektu oraz to w jaki sposób projekt ma zostać zbudowany.

Programista często będzie pracować z grupą projektów. Dla przykładu dobrą praktyką programistyczną jest pisanie testów. Większość testów nie musi być wdrążonych wraz z projektem, więc możemy zaimplementować testy automatyczne w innym projekcie. Innym przypadkiem jest sytuacja w której na przykład chcemy rozdzielić nasz projekt. Dla przykładu jeśli nasza aplikacja jest typu desktop, ale mamy również część webową i chcemy aby część kodu była wspólna dla obu aplikacji.

Visual Studio pomaga we współpracy projektów grupując je w tak zwaną *solucję* (ang. solution). Można przejrzeć aktualną solucję oraz jej projekty w oknie *Solution Explorer*.



Rys. 5 : Okno Solution Explorer. [1]

Visual Studio może załadować projekt tylko jeśli taki projekt jest członkiem danej solucji. Kiedy użytkownik tworzy nowy projekt można dodać go do istniejącej solucji, lub utworzyć dla niego nową. Jeśli użytkownik spróbuje otworzyć istniejący projekt, Visual Studio poszuka solucji do której tej projekt należy. Jeśli Visual Studio nie znajdzie poszukiwanej solucji to wymusi na użytkowniku jej utworzenie. Dzieje się tak ze wzgledu na to, że wiele operacji w Visual Studio dzieje się w zakresie danej solucji. Kiedy użytkownik buduje swój kod, tak na prawdę buduje on solucję z projektem lub projektami. Decyzja czy budować w trybie *dubug* czy *release* również odbywa się na poziomie solucji. Wyszukiwarka elementów również może szukać obiektów w całej solucji. Fizycznie solucja to plik z rozszerzeniem .*sln,* który również zawiera konfigurację jej zawartości. [1]

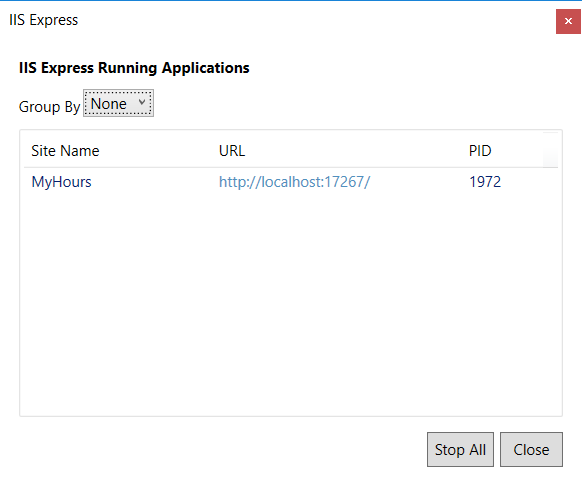
## ASP.NET

### Wstęp

Platforma .Net dostarcza wiele możliwości tworzenia aplikacji internetowych. Oczywiście przy tworzeniu stron internetowych również należy się dedykowanym pod to frameworkiem, którym właśnie jest ASP.NET. ASP.NET pokrywa w całości część technologiczną zarówno po stronie klienta jak i serwera, niezależnie czy pisze się serwisy internetowe czy interfejs użytkownika. Dla przykładu ASP.NET wspiera uruchamianie aplikacji po stronie serwera Microsoft jakim jest IIS (ang. Internet Information Serivce). Posiada on rozszerzalny moduł dla obłsługi zapytań oraz różne inne podstawowe mechanizmy jak np. autentykację użytkownika. Po stronie klienta natomiast można tworzyć interfejs użytkownika, co więcej ASP.NET udostępnia dwa silniki pozwalające tworzyć taki interfejs. Każdy z tych silników definiuje swoją składnie pisania stron internetowych. Starszy silnik Web Forms oraz nowszy, łatwiejszy silnik Razor nad który będzie szerzej opisany się w kolejnym rozdziale. Którego kolwiek z silników się nie wybierze, należy również zdecycować jak dana aplikacja webowa określa, które podstrony są pokazywane oraz jaki kod będzie te podstrony obsługiwał. Najprostrzym sposobem jest utworzenie struktury plików, której drzewo jest odzwierciedlone poprzez adres URL. Takie podejście jest bardzo łatwe w zrozumieniu jednak nie jest elastyczne. Np strony oparte na blogach czasami zawierają w swoich sciezkach daty np : *http://example.com/blog/2017/01/15/example* i nieefektywne by było tworzenie nowego folderu każdego miesiąca. ASP.NET udostępnia routing URL który powoduje, że struktura strony internetowej jest bardziej dynamiczna. Można używać tego w dowolnym projekcie ASP.NET. Istnieje jednak bardziej znany i bardziej przyjazny programiście sposób na osiągnięcie tego, poprzez projekt MVC (ang. Model View Controller).

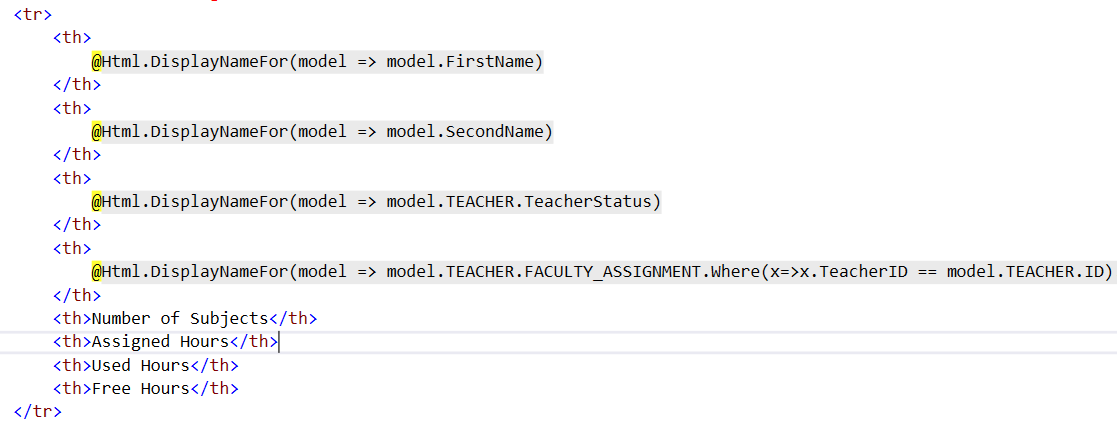
### IIS – Internet Information Service

IIS (ang. Internet Information Service) jest rozszerzalnym serwerem internetowym stworzonym przez firmę Microsoft. Wspiera takie protokoły internetowe jak : HTTP, HTTPS, FTP, FTPS, SMTP oraz NNTP. Według firmy Netcraft w 2014 był drugim najpopularniejszym serwerem zaraz po Apache HTTP Server.



Rys. 6 : Okno IIS Express z listą uruchomionych aplikacji

### Razor

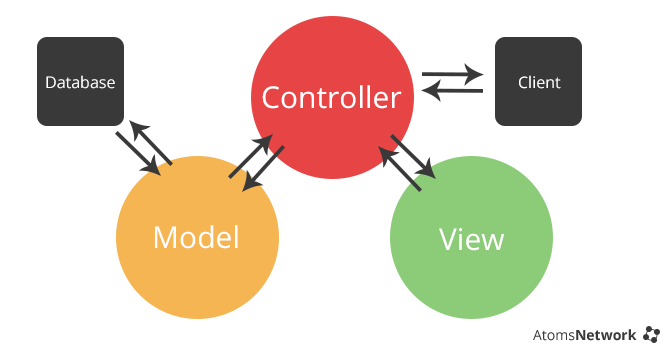
Razor jest to składnia do tworzenia stron HTML która pozwala na zamieszczenie kodu np C# na formularzach HTML. Kod ten będzie uruchamiany na swerwerze, dając kontrolę nad tym co się pojawia na stronie w trakcie jej działania.

Rys. 7 : Składnia Razor [1]

Na powższym zdjeciu można zauważyć wykorzystaną składnie Razor, elementy poprzedzone znakiem @ to własnie początek składni Razor i kod języka C#. [1]

### ASP.NET MVC

ASP.NET MVC tak na prawdę nie jest odrębna technologia od ASP.NET, to po prostu wykorzystanie wzorca MVC (ang. Model View Controller) przy tworzeniu stron w technologii ASP.NET. Model projektowy MVC pozwala na ustrukturyzowanie projektu dzieląc go na 3 wartswy, warstwę Modelu (Model) która odpowiada za dane które wyświetlane są na stronie internetowej, mogą to być dane z bazy danych. Widoku (View) który odpowiada za to co użytkownik widzi na ekranie, pliki .cshtml i wszystkie inne, które odpowiadają za interfejs użytkownika oraz Kontrolera (Controller), który jest warstwą pośrednią pomiędzy wartswią widoku a wartswą modelu i który odpowiada za logikę biznesową tego jak dane są wyświetlane w interfejsie. [2]



Rysunek 4 : Model MVC (Źródło [4])

## Microsoft SQL Server

### Wstęp

Microsoft SQL Server jest systemem zarządania relacyjną bazą danych wydanym przez firmę Microsoft. Jako serwer bazy danych udostępnia podstawowe funkcje przechowywania i otrzymywania danych. Dzieje się to za pośrednictwen zapytań wysyłanych przez aplikacje zewnętrzne uruchomione na tym samym komputerze lub innym komputerze w sieci włączając w to sieć internetową.

Microsoft wydał wiele wersji SQL Server’a. Zależnie od zapotrzebowania, poczynając od małych aplikacji uruchamianych na jednej maszynie po ogromne aplikacje internetowe, na których znajduje się wiele użytkowników.

Warstwa protokołu implementuje zewnętrzny interfejs dla SQL Server’a. Wszystkie operacje, które mogą być wywołane na SQL Serverze są obsługiwane poprzez Tabelaryczny Strumień Danych (ang. Tabular Data Stream). Jest on protokołem komunikacyjnym do transferu danych pomiędzy serwerem bazy danych a klientem. Został on wymyślony przez firmę Sybase w 1984 roku, a następnie ulepszony przez Microsoft na potrzeby SQL Server’a. Pakiety TDS mogą być pokryte innymi protokołami transportu danych m.in. TCP/IP, FIFO czy pamięć dzielona. Konsekwetnie dostęp do SQL Server’a jest możliwy dzięki nim. Dodatkowo, API SQL Server’a jest również udostpeniane poprzez serwisy internetowe.



Rysunek 5 : (Źródło [5])

### SQL Server Management Studio

SQL Server Management Studio (SSMS) jest narzędziem w postaci oprogramowania, wydanym razem z Micorosft SQL Server 2005. Używany jest do konfiguracji, zarządzania i administrowania wszystkich komponentów w Microsoft SQL Server. Narzędzie zawiera zarówno edytor skryptów SQL, jak i graficzne narzędzie, które pracuje ze wszystkimi obiektami i funkcjami serwera.

Główną funkcjonalnością SSMS jest eksplorator obiektow (ang. object explorer), który umożliwia użytkownikowi przeglądanie, wybieranie i działanie w kontekście każego obiektu serwerwa.

Microsoft wydał również wersję express, która jest do pobrania za darmo, jednak ostatnia wersja SQL Server Management Studio pozwala w pełni zarówno na połączenie sie jak i zarządzanie każdą instancją SQL Server'a. Microsoft umożłiwił również kompatybilność wsteczną z poprzednimi wersjami SQL Server'a, tak aby SSMS mógł obsługiwać starsze serwery. Poczynając od wersji 11, aplikacja została przepisana w technologii WPF dzięki czemu jest ona podobna do Micorosoft Visual Studio 2010. W czerwcu 2015, Microsoft zapowiedział wydanie przyszłej wersji SQL Server Management Studio, która będzie aplikacją niezależną już od silnika SQL Server.

## ADO.NET

### Wstęp

ADO.NET jest zestawem klas które udostępniają serwisy dostępu do danych dla programistów .Net. ADO.NET udostępnia bogaty zestaw komponentów do tworzenia aplikacji, które współdzielą swoje dane. Jest Integralnym elementem platformy .Net, dając dostęp do danych (relacyjnych, XML, i danych aplikacji). ADO.NET rozwiązuje wiele problemów programistycznych zwiazanych z operowaniem na danych, które pojawiają się przy tworzeniu aplikacji, narzędzi, języków czy przeglądarek internetowych. [6]

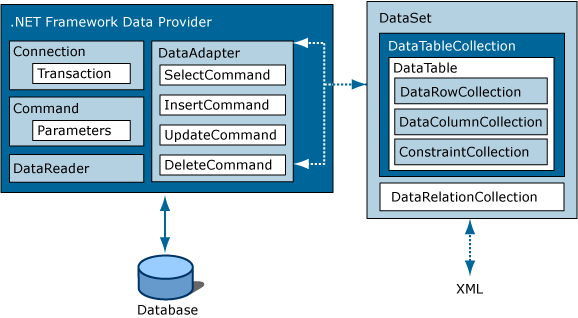
Poniższy rysunek przedstawia architekturę ADO.NET.

**Connection** – Połączenie do źródła danych.

**Command** – Odpowiada za wykonywanie zapytań na bazie danych.

**Data** **Reader** – odpowiada za otrzymywanie danych w formie tylko do odczytu

**Data** **Adapter** – działa na zasadzie „mostu” pomiedzy zbiorem danych (DataSet) a źródłem danych.



Rys. 6 : Architekura ADO.NET

### Entity Framework

Pisanie i zarządzanie kodem ADO.NET dostępu do danych może być czasochłonne. Microsoft udostepnił framework ORM nazwany Entity Framework do zautomatyzowania różnych czynności związanych z bazą danych. Według definicji Microsoft Entity Framework jest to framework do mapowania obiektowo-relacyjnego, który umożliwia programistom prace z relacyjnymi danymi jak z określonymi obiektami, eliminując potrzebę zagłębiania się w kod jaki trzeba było by napisać aby uzyskać wymagany dostęp do bazy.

Używając Entity Framework developer pisze zapytania w języku LINQ. Następnie otrzymuje dane jako obiekty silnie typowane, którymi może dowolnie manipulować. Implementacja Entity Framework dostarcza serwisy takie jak kontrola zmian, leniwe ładowanie (ang. Lazy Loading) czy tłumaczenie zapytań tak aby programista mógł skupić się na logice biznesowej aplikacji którą pisze zamiast na podstawowych elementach dostępu do danych.



Rys. 7 : Schemat działania Entity Framework

Entity Framework jest rozserzeniem do ADO.NET, który daje programiście zautomatyzowany dostep do danych oraz przechowywanie ich w bazie danych.

Entity Framework może być użyteczny w następujących trzech scenariuszach.

* Istnieje gotowa baza danych, na podstawie której Entity Framework generuje klasy domeny, które umożliwią dostęp do danych z bazy danych.
* Istnieją już napiasne przez progarmiste klasy domeny, na podstawie których Entity Framework tworzy bazę danych.
* Nie istnieją jeszcze klasy domeny ani baza danych, jednak istnieje schemat bazy danych zaprojektowany w odpowiednim narzędziu do tworzenia diagramów. Entity Framework na podstawie tak utworzonego diagramu, tworzy klasy domeny oraz bazę danych.



Rys. 8 : Scenariusze pracy Entity Framework [7]

Aby lepiej zrozumieć działanie Entity Framework, należy wyjaśnić czym jest ORM.

ORM jest narzędziem służącym do przechowywania danych z obiektów domeny dla relacyjnej bazy danych takiej jak Microsoft SQL Server. Dzieje się to w sposób całkowicie zautomatyzowany, bez potrzeby pisania kodu.

ORM zawiera trzy główne elementy :

* Obiekty klasy domeny
* Obiekty relacyjnej bazy danych oraz
* Informacje na temat mapowania, które Entity Framework używa przy tłumaczeniu obiektów domeny na obiekty bazy danych.

Dzięki ORM projekt domeny jest odseperowany od projektu bazy danych To powoduje z kolei że aplikacja jest utrzymywalna i rozszerzalna.

ORM również automatyzuje operacje CRUD (Create, Read, Update & Delete), tak aby programista nie musiał pisać ich ręcznie. [7]

Typowe narzędzie ORM generuje klasy do interakcji z bazą danych tak jak jest to pokazane na schemacie poniżej.



Rys. 9 : Działanie narzędzia O/RM. [7]

### LINQ

Zintegrowany język zapytań (ang. Language Integrated Query). Jest składnią do tworzenia zapytań w jezykach C# i VB.NET używany do zapisywania i otrzymyawnia danych z różnych źródeł danych. Zasada jego działania jest zbliżona do zasady działania języka SQL.

LINQ Jest zintegrowany z językami C# i VB.NET, eliminując nieścisłości pomiędzy jezykami programowania a bazą danych. Tak samo jak język SQL służy do operowania na danych w bazie danych, która jest również kontenerem na dane tak LINQ dostarcza interfejs do tworzenia zapytań dla różnych typów źródeł danych. Źródłami danych mogą być kolekcje obiektów (listy, słowniki), baza danych SQL, XML, serwisy webowe i wiele innych.

LINQ zawsze pracuje z obiektami tak aby programista mógł używać tych samych wzorców programowania przy tworzeniu zapytań.

Poniższy rysunek pokazuje jakie kolekcje danych wspiera LINQ. [8]



Rys. 10 : Kontenery danych wspierane przez LINQ [8]

Zaletami języka LINQ są [8]:

* **Uniwersalność** - Dzięki ustandaryzowaniu tworzenia zapytań programista nie musi uczyć się nowego języka dla każdego typu źródeł danych. Ten sam język istnieje dla list jak i dla obiektów bazodanowych. Można napisać ten sam kod dla wielu różnych kontenerów z danymi.
* **Czystość kodu** - dzięki LINQ znacznie zmniejsza się ilość napisanego kodu potrzebnego do operacji na kolekcji danych.
* **Czytelność kodu** - LINQ wprowadza łatwą w zrozumieniu składnię. Kod napisany w LINQ będzie zrozumiały i utrzymywalny.
* **Bezpieczeństwo** - pozwala na sprawdzanie typów obiektów podczas kompilacji.
* **IntelliSense** - LINQ udostępnia podpowiadanie składni w trakcie pisania kodu.
* **Elastyczność** - pisząc zapytanie w LINQ możemy okreslić w jaki sposób i w jakiej postaci chcemy otrzymać wybierane dane.

# Projekt Systemu

## Baza Danych

FACULTY – Tabela zawierająca dane reprezentujące wydziały na politechnice łódzkiej

CREATE TABLE [dbo].[FACULTY](

[ID] [int] IDENTITY(1,1) NOT NULL,

[Name] [varchar](50) NOT NULL,

[Description] [varchar](500) NOT NULL,

CONSTRAINT [PK\_FACULTY] PRIMARY KEY CLUSTERED ([ID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

FACULTY\_ASSIGNMENT – Tabela zawierająca dane reprezentujące przypisanie nauczyciela razem z grupą studentów do danego wydziału. Tabela ma klucze obce, które powiązane są z kluczami głównymi tabel FACULTY, STUDENT\_GROUP, TEACHER

CREATE TABLE [dbo].[FACULTY\_ASSIGNMENT](

[ID] [int] IDENTITY(1,1) NOT NULL,

[FacultyID] [int] NOT NULL,

[StudentGroupID] [int] NOT NULL,

[TeacherID] [int] NOT NULL,

CONSTRAINT [PK\_FACULTY\_ASSIGNMENT] PRIMARY KEY CLUSTERED ([ID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

ALTER TABLE [dbo].[FACULTY\_ASSIGNMENT] WITH CHECK ADD CONSTRAINT [FK\_FACULTY\_ASSIGNMENT\_FACULTY] FOREIGN KEY([FacultyID])

REFERENCES [dbo].[FACULTY] ([ID])

ALTER TABLE [dbo].[FACULTY\_ASSIGNMENT] WITH CHECK ADD CONSTRAINT [FK\_FACULTY\_ASSIGNMENT\_STUDENT\_GROUP] FOREIGN KEY([StudentGroupID])

REFERENCES [dbo].[STUDENT\_GROUP] ([ID])

ALTER TABLE [dbo].[FACULTY\_ASSIGNMENT] WITH CHECK ADD CONSTRAINT [FK\_FACULTY\_ASSIGNMENT\_TEACHER] FOREIGN KEY([TeacherID])

REFERENCES [dbo].[TEACHER] ([ID])

MAJOR – Tabela zawierająca dane reprezentujące kierunki na politechnice łódzkiej, przypisane do określonych wydziałów. Tabela zawiera klucz obcy powiązany z kluczem głównym tabeli FACULTY

CREATE TABLE [dbo].[MAJOR](

[ID] [int] IDENTITY(1,1) NOT NULL,

[Name] [varchar](200) NOT NULL,

[FacultyID] [int] NOT NULL,

CONSTRAINT [PK\_MAJOR] PRIMARY KEY CLUSTERED ([ID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

ALTER TABLE [dbo].[MAJOR] WITH CHECK ADD CONSTRAINT [FK\_MAJOR\_FACULTY] FOREIGN KEY([FacultyID])

REFERENCES [dbo].[FACULTY] ([ID])

NOTIFICATION\_STATUS – Tabel zawierająca dane reprezentujące statusy powiadomień

CREATE TABLE [dbo].[NOTIFICATION\_STATUS](

[ID] [int] IDENTITY(1,1) NOT NULL,

[Name] [varchar](20) NOT NULL,

[Description] [varchar](50) NOT NULL,

CONSTRAINT [PK\_NOTIFICATION\_STATUS] PRIMARY KEY CLUSTERED(ID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

SPECIALITY – Tabela zawierająca dane reprezentujące specjalności na Politechnice Łódzkiej, które przypisane są do określonego kierunku. Tabelza zawiera klucz obcy powiązany z kluczem głównym w tabeli MAJOR.

CREATE TABLE [dbo].[SPECIALITY](

[ID] [int] IDENTITY(1,1) NOT NULL,

[Name] [varchar](50) NOT NULL,

[Description] [varchar](500) NOT NULL,

[MajorID] [int] NOT NULL,

CONSTRAINT [PK\_SPECIALITY] PRIMARY KEY CLUSTERED([ID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

ALTER TABLE [dbo].[SPECIALITY] WITH CHECK ADD CONSTRAINT [FK\_SPECIALITY\_MAJOR] FOREIGN KEY([MajorID])

REFERENCES [dbo].[MAJOR] ([ID])

STUDENT\_GROUP – Tabela zawiera dane reprezentujące grupy studentów na Politechnice Łódzkiej.

CREATE TABLE [dbo].[STUDENT\_GROUP](

[ID] [int] IDENTITY(1,1) NOT NULL,

[Name] [varchar](50) NOT NULL,

[Description] [varchar](500) NOT NULL,

CONSTRAINT [PK\_STUDENT\_GROUP] PRIMARY KEY CLUSTERED ([ID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

STUDIES\_TYPE\_DICT – Tabea zawiera dane reprezentujące rodzaj studiów wraz ze stopniem (Dzienne, Zaoczne, 1 Stopień, 2 Stopień itd).

CREATE TABLE [dbo].[STUDIES\_TYPE\_DICT](

[ID] [int] NOT NULL,

[Name] [varchar](50) NOT NULL,

[Description] [varchar](50) NOT NULL,

CONSTRAINT [PK\_STUDIES\_TYPE\_DICT] PRIMARY KEY CLUSTERED ([ID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

SUBJECT – Tabelza zawierająca dane reprezentujące przedmioty na Politechnice Łódzkiej. Każdy przedmiot przypisany jest do odpowiedniej specjalności. Tabela zawiera informacje na temat maksymalnej ilości przypisanych godzin na przedmiot wraz z podziałem tych godzin na typ przedmiotu tj. Laboratoria, wykłady itp. Dla celów kontroli poprawności przypisanych godzin tabela zawiera również informacje o aktualnym wykorzystaniu godzin. Tabela posiada klucz obcy powiązany z kluczem głównym tabeli SPECIALITY.

CREATE TABLE [dbo].[SUBJECT](

[ID] [int] IDENTITY(1,1) NOT NULL,

[SubjectCode] [varchar](50) NOT NULL,

[Name] [varchar](100) NOT NULL,

[Language] [varchar](50) NULL,

[SpecialityID] [int] NOT NULL,

[NumberOfGroups] [int] NOT NULL,

[AssignedHours] [int] NOT NULL,

[UsedHours] [int] NOT NULL,

[LabHours] [int] NOT NULL,

[LecHours] [int] NOT NULL,

[SemHours] [int] NOT NULL,

[PrjHours] [int] NOT NULL,

[ExcHours] [int] NOT NULL,

[DipHours] [int] NOT NULL,

CONSTRAINT [PK\_SUBJECT] PRIMARY KEY CLUSTERED([ID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

ALTER TABLE [dbo].[SUBJECT] WITH CHECK ADD CONSTRAINT [FK\_SUBJECT\_SPECIALITY] FOREIGN KEY([SpecialityID])

REFERENCES [dbo].[SPECIALITY] ([ID])

SUBJECT\_ASSIGNMENT – Tabela zawierająca dane reprezentujące przypisanie przedmiotu do nauczyciela i grupy studentów. Tabela ta zawiera klucze obce które powiązane są z kluczami głównymi tabel TEACHER, STUDENT\_GROUP, SUBJECT, SUBJECT\_TYPE\_DICT, STUDIES\_TYPE.

Tabela zawiera również informacje o tym, czy dane przypisanie przedmiotu do nauczyciela jest zastępstwem za inną osobę.

CREATE TABLE [dbo].[SUBJECT\_ASSIGNMENT](

[ID] [int] IDENTITY(1,1) NOT NULL,

[Hours] [int] NOT NULL,

[Semester] [int] NOT NULL,

[TeacherID] [int] NOT NULL,

[IsSubstitute] [bit] NOT NULL,

[IsSubstituteDescription] [varchar](max) NULL,

[StudentGroupID] [int] NOT NULL,

[SubjectID] [int] NOT NULL,

[SubjectTypeID] [int] NOT NULL,

[StudiesTypeID] [int] NOT NULL,

[ReplacedName] [varchar](1000) NULL,

CONSTRAINT [PK\_SUBJECT\_ASSIGNMENT] PRIMARY KEY CLUSTERED ([ID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

ALTER TABLE [dbo].[SUBJECT\_ASSIGNMENT] ADD CONSTRAINT [DF\_SUBJECT\_ASSIGNMENT\_IsSubstitute] DEFAULT ((0)) FOR [IsSubstitute]

ALTER TABLE [dbo].[SUBJECT\_ASSIGNMENT] WITH CHECK ADD CONSTRAINT [FK\_SUBJECT\_ASSIGNMENT\_STUDENT\_GROUP] FOREIGN KEY([StudentGroupID])

REFERENCES [dbo].[STUDENT\_GROUP] ([ID])

ALTER TABLE [dbo].[SUBJECT\_ASSIGNMENT] WITH CHECK ADD CONSTRAINT [FK\_SUBJECT\_ASSIGNMENT\_STUDIES\_TYPE\_DICT] FOREIGN KEY([StudiesTypeID])

REFERENCES [dbo].[STUDIES\_TYPE\_DICT] ([ID])

ALTER TABLE [dbo].[SUBJECT\_ASSIGNMENT] WITH CHECK ADD CONSTRAINT [FK\_SUBJECT\_ASSIGNMENT\_SUBJECT] FOREIGN KEY([SubjectID])

REFERENCES [dbo].[SUBJECT] ([ID])

ALTER TABLE [dbo].[SUBJECT\_ASSIGNMENT] WITH CHECK ADD CONSTRAINT [FK\_SUBJECT\_ASSIGNMENT\_SUBJECT\_TYPE\_DICT] FOREIGN KEY([SubjectTypeID])

REFERENCES [dbo].[SUBJECT\_TYPE\_DICT] ([ID])

ALTER TABLE [dbo].[SUBJECT\_ASSIGNMENT] WITH CHECK ADD CONSTRAINT [FK\_SUBJECT\_ASSIGNMENT\_TEACHER] FOREIGN KEY([TeacherID])

REFERENCES [dbo].[TEACHER] ([ID])

SUBJECT\_ASSIGNMENT\_TEMP – Tabela zawierająca tymczasowe informacje o przypisaniu nauczyciela I grupy studentów do przedmiotu. Dane do tej tabeli dodawawne są tylko w przypadku jeśli nauczyciel z poziomu aplikacji wyśle prośbę do administratora o dodanie nowego bądź zmodyfikowanie istniejącego przedmiotu. Dane z tej tabeli usuwane są zaraz po odpowiedzi administratora. Jeśli prośba do administratora o dodanie przedmiotu będzie zaakceptowana, to odpowiednie dane reprezentujące ten przedmiot z tej tabeli przenoszone są do tabeli SUBJECT\_ASSIGNMENT, a następnie usuwane z SUBJECT\_ASSIGNMENT\_TEMP. Jeśli prośba ta zostanie odrzucona, to dane zostaną tylko usunięte z SUBJECT\_ASSIGNMENT\_TEMP. W przypadku modyfikowania danych w SUBJECT\_ASSIGNMENT, dane w tabeli SUBJECT\_ASSIGNMENT\_TEMP służą do porównania elementu po modyfikacji z elementem oryginalnym.

Struktura tej tabeli jest dokładną kopią struktury tabeli SUBJECT\_ASSIGNMENT.

TEACHER – Tabela zawierająca dane reprezentujące nauczyciela. Każdy nauczyciel jest przypisany do określonego wydziału, dlatego tabela zawiera klucz obcy powiązany z kluczem głównym tabeli FACULTY.

CREATE TABLE [dbo].[TEACHER](

[ID] [int] IDENTITY(1,1) NOT NULL,

[FirstName] [varchar](50) NOT NULL,

[SecondName] [varchar](50) NOT NULL,

[TeacherStatus] [varchar](50) NOT NULL,

[AssignedHours] [int] NOT NULL,

[FullName] [varchar](5000) NULL,

[FacultyID] [int] NULL,

CONSTRAINT [PK\_TEACHER] PRIMARY KEY CLUSTERED ([ID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

ALTER TABLE [dbo].[TEACHER] WITH CHECK ADD CONSTRAINT [FK\_TEACHER\_FACULTY] FOREIGN KEY([FacultyID])

REFERENCES [dbo].[FACULTY] ([ID])

USER – Tabela zawierająca informacje reprezentujące użytkownika. Tabela ta zawiera klucze obce powiązane z kluczami głównymi tabel USER\_TYPE, TACHER. Tabela zawiera również klucz obcy powiązany z kliczem głownym tabeli AspNetUsers. Wartość tego klucza generowana jest przez Entity Framework podczas rejestracji użytkownika.

CREATE TABLE [dbo].[USER](

[ID] [int] IDENTITY(1,1) NOT NULL,

[Name] [varchar](300) NOT NULL,

[FirstName] [varchar](100) NOT NULL,

[SecondName] [varchar](100) NOT NULL,

[PhoneNumber] [varchar](50) NULL,

[UserTypeID] [int] NOT NULL,

[TeacherID] [int] NULL,

[AspNetUserID] [nvarchar](128) NOT NULL,

CONSTRAINT [PK\_USER] PRIMARY KEY CLUSTERED ([ID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

ALTER TABLE [dbo].[USER] WITH CHECK ADD CONSTRAINT [FK\_USER\_AspNetUsers] FOREIGN KEY([AspNetUserID])

REFERENCES [dbo].[AspNetUsers] ([Id])

ALTER TABLE [dbo].[USER] WITH CHECK ADD CONSTRAINT [FK\_USER\_TEACHER] FOREIGN KEY([TeacherID])

REFERENCES [dbo].[TEACHER] ([ID])

ALTER TABLE [dbo].[USER] WITH CHECK ADD CONSTRAINT [FK\_USER\_USER\_TYPE] FOREIGN KEY([UserTypeID])

REFERENCES [dbo].[USER\_TYPE] ([ID])

USER\_NOTIFICATION – Tabela zawierająca dane reprezentujące powiadomienia jakie dostaje administrator systemu od użytkownika. Powiedomienia te, to prośby wysyłane przez użytkownika o dodanie nowych bądź modyfikacje lub usuniecie istniejących przedmiotów. Tabela zawiera informacje o typie powiadomienia (dodanie, usuniecie, modyfikacja), użytkownikowi który to powiadomienie wysłał oraz o przypisaniu przedmiotu, którego to powiadomienie dotyczy.

Tabela zawiera klucze obce, które powiązane są z kluczami głównymi tabel USER, NOTIFICATION\_STATUS, SUBJECT\_ASSIGNMENT\_TEMP, SUBJECT\_ASSIGNMENT.

CREATE TABLE [dbo].[USER\_NOTIFICATION](

[ID] [int] IDENTITY(1,1) NOT NULL,

[SenderID] [int] NOT NULL,

[UserID] [int] NOT NULL,

[Name] [nchar](200) NOT NULL,

[Description] [nchar](500) NOT NULL,

[StatusID] [int] NOT NULL,

[Date] [smalldatetime] NOT NULL,

[SubjectAssignmentTempID] [int] NULL,

[SubjectAssignmentID] [int] NULL,

[SubjectName] [varchar](100) NULL,

CONSTRAINT [PK\_UserNotification] PRIMARY KEY CLUSTERED ([ID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

ALTER TABLE [dbo].[USER\_NOTIFICATION] WITH CHECK ADD CONSTRAINT [FK\_USER\_NOTIFICATION\_NOTIFICATION\_STATUS] FOREIGN KEY([StatusID])

REFERENCES [dbo].[NOTIFICATION\_STATUS] ([ID])

ALTER TABLE [dbo].[USER\_NOTIFICATION] WITH CHECK ADD CONSTRAINT [FK\_USER\_NOTIFICATION\_SUBJECT\_ASSIGNMENT] FOREIGN KEY([SubjectAssignmentTempID])

REFERENCES [dbo].[SUBJECT\_ASSIGNMENT\_TEMP] ([ID])

ALTER TABLE [dbo].[USER\_NOTIFICATION] WITH CHECK ADD CONSTRAINT [FK\_USER\_NOTIFICATION\_SUBJECT\_ASSIGNMENT1] FOREIGN KEY([SubjectAssignmentID])

REFERENCES [dbo].[SUBJECT\_ASSIGNMENT] ([ID])

ALTER TABLE [dbo].[USER\_NOTIFICATION] WITH CHECK ADD CONSTRAINT [FK\_USER\_NOTIFICATION\_USER] FOREIGN KEY([SenderID])

REFERENCES [dbo].[USER] ([ID])

USER\_TYPE – Tabela zawierająca dane reprezentujące typ użytkownika (administrator, nauczyciel)

CREATE TABLE [dbo].[USER\_TYPE](

[ID] [int] IDENTITY(1,1) NOT NULL,

[Name] [varchar](50) NOT NULL,

[Description] [varchar](50) NOT NULL,

CONSTRAINT [PK\_USER\_TYPE] PRIMARY KEY CLUSTERED ([ID] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

AspNetRoles – Tabela zawierająca dane reprezentujące role użytkownika. Dane te są niezbędne do autentykacji i rozróżniania użytkowników pomiędzy administratora i nauczyciela.

CREATE TABLE [dbo].[AspNetRoles](

[Id] [nvarchar](128) NOT NULL,

[Name] [nvarchar](256) NOT NULL,

CONSTRAINT [PK\_dbo.AspNetRoles] PRIMARY KEY CLUSTERED ([Id] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

AspNetUserRoles – Tabela zawierająca dane reprezentujące przypisanie użytkownika do odpowiedniej roli za pomocą jego numeru ID. Tabela posiada klucze obce powiązane z kluczem głównym tabel AspNetUsers i AspNetRoles. Tabela ta odzwierciedla relacje wiele do wielu.

CREATE TABLE [dbo].[AspNetUserRoles](

[UserId] [nvarchar](128) NOT NULL,

[RoleId] [nvarchar](128) NOT NULL,

CONSTRAINT [PK\_dbo.AspNetUserRoles] PRIMARY KEY CLUSTERED ([UserId] ASC, [RoleId] ASC) WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]) ON [PRIMARY]

ALTER TABLE [dbo].[AspNetUserRoles] WITH CHECK ADD CONSTRAINT [FK\_dbo.AspNetUserRoles\_dbo.AspNetRoles\_RoleId] FOREIGN KEY([RoleId])

REFERENCES [dbo].[AspNetRoles] ([Id])

ON DELETE CASCADE

ALTER TABLE [dbo].[AspNetUserRoles] WITH CHECK ADD CONSTRAINT [FK\_dbo.AspNetUserRoles\_dbo.AspNetUsers\_UserId] FOREIGN KEY([UserId])

REFERENCES [dbo].[AspNetUsers] ([Id])

ON DELETE CASCADE

AspNetUsers – Tabela zawierająca dane reprezentujące użytkownika. Tabela ta jest używana przy rejestracji użytkownika. Zawiera hasło zaszyfrowane algorytmem haszującym. Tabela zawiera równierz inne dane, które mogą być użyte przy autentykacji użytkownika, takie jak potwierdzenie adresu email lub potwierdzenie numeru telefonu.

CREATE TABLE [dbo].[AspNetUsers](

[Id] [nvarchar](128) NOT NULL,

[Email] [nvarchar](256) NULL,

[EmailConfirmed] [bit] NOT NULL,

[PasswordHash] [nvarchar](max) NULL,

[SecurityStamp] [nvarchar](max) NULL,

[PhoneNumber] [nvarchar](max) NULL,

[PhoneNumberConfirmed] [bit] NOT NULL,

[TwoFactorEnabled] [bit] NOT NULL,

[LockoutEndDateUtc] [datetime] NULL,

[LockoutEnabled] [bit] NOT NULL,

[AccessFailedCount] [int] NOT NULL,

[UserName] [nvarchar](256) NOT NULL,

CONSTRAINT [PK\_dbo.AspNetUsers] PRIMARY KEY CLUSTERED([Id] ASC)

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY])

## 

## Struktura systemu

### Wstęp

### Model

Model stanowi wartstwę dostępu do danych. Reprezentuje on dane, które używane są w aplikacji. W projekcie głównym modelem jest model wygenerowant przez Entity Framework. Jest to plik z rozszerzeniem .edmx. Struktura tego modelu jest dokładnie taka sama jak strutkura tabel w bazie danych opisanych w punkcie 4.1. Baza Danych. Dodatkowo oprócz modelu Entity Framework, w aplikacji jest model pomocniczny AccountViewModel, używany przy rejestracji użytkownika. Jest to klasa, która składa w skład której wchodzą mniejsze klasy, reprezentujące odpowiednie modele, takie jak model do logowania (LoginViewModel) czy rejestracji (RegisterViewModel). Za ich pomocą przeprowadzana jest operacja logowania bądź rejestracji.

Przy projektowaniu systemu własciwe zaprojektowanie warstwy dostępu do danych oraz reprezentacji modelowej ORM ma ogromny wpływ na wydajność całej aplikacji. Operacje takie jak wykonanie zapytań lub dostęp do danych, zwłaszcza na dużej bazie danych (więcej niż 500 000 rekordów) są w logice biznesowej działaniami, które zużywają najwięcej zasobów.   
Oprócz tego, właściwe zaprojektowanie warstwy danych jest pomocne przy pisaniu kodu. Cały proces manipulacji danymi w oparciu o klasy domeny, które stanowią reprentacje tabel bazie danych jest łatwy i pozwala znacznie skrócić pisany kod.

Poniżej został umieszczony przykładowy fragment kodu, który pobiera dane wraz z przykładową reprezentacją tabeli przy pomocy klasy Entity Framework. Do pobrania danych użyty został kontekst danych, który reprezentuje kombinację wzorców Repository oraz Unit of Work, aby mógł być użyty do pobierania danych z bazy danych, które będą zwrócone jako obiekty.

public partial class TAM\_DBEntities : DbContext

{

public TAM\_DBEntities()

: base("name=TAM\_DBEntities")

{

}

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

throw new UnintentionalCodeFirstException();

}

public virtual DbSet<AspNetRoles> AspNetRoles { get;set; }

public virtual DbSet<AspNetUsers> AspNetUsers { get;set; }

public virtual DbSet<FACULTY> FACULTY { get;set; }

public virtual DbSet<FACULTY\_ASSIGNMENT> FACULTY\_ASSIGNMENT { get;set; }

public virtual DbSet<MAJOR> MAJOR { get;set; }

public virtual DbSet<NOTIFICATION\_STATUS> NOTIFICATION\_STATUS { get;set; }

public virtual DbSet<SPECIALITY> SPECIALITY { get;set; }

public virtual DbSet<STUDENT\_GROUP> STUDENT\_GROUP { get;set; }

public virtual DbSet<STUDIES\_TYPE\_DICT> STUDIES\_TYPE\_DICT { get;set; }

public virtual DbSet<SUBJECT> SUBJECT { get;set; }

public virtual DbSet<SUBJECT\_ASSIGNMENT> SUBJECT\_ASSIGNMENT { get;set; }

public virtual DbSet<SUBJECT\_ASSIGNMENT\_TEMP> SUBJECT\_ASSIGNMENT\_TEMP { get;set; }

public virtual DbSet<SUBJECT\_TYPE\_DICT> SUBJECT\_TYPE\_DICT { get;set; }

public virtual DbSet<sysdiagrams> sysdiagrams { get;set; }

public virtual DbSet<TEACHER> TEACHER { get;set; }

public virtual DbSet<USER> USER { get;set; }

public virtual DbSet<USER\_NOTIFICATION> USER\_NOTIFICATION { get;set; }

public virtual DbSet<USER\_TYPE> USER\_TYPE { get;set; }

}

Implementacja kontekstu bazy danych. Klasa musi dziedziczyć po klasie DbContext oraz zawiera kolekcje typu DbSet, które przechowują dane z bazy danych.

protected TAM\_DBEntities db = new TAM\_DBEntities();

var user = db.USER.Where(x => x.AspNetUserID == id).FirstOrDefault();

Do pobierania danych został użyty język LINQ. LINQ tłumaczy zapytania w postaci funkcji .NET do zapytań SQL, które wykonywane są bezpośrednio na bazie danych.  
W powuższym przykładnie do zmiennej user zostanie pobrany rekord z tabeli USER, gdzie wartość w kolumnie AspNetUserID będzie równa zmiennej id. Funkcja FirstOrDefault() zwraca pierwszy element z kolekcji pobranych danych, lub null jeśli dane nie zostały znalezione.

Zaletą tego podejścia jest to, że jeśli baza danych nie zwróci żadnych wartości na podstawie utworzengo zapytania LINQ, to funkcja zwróci null dzięki czemu łatwo potem sprawdzić czy zapytanie zwraca wartość. Następnie na podstawie prawdziwości tego warunku można wykonać odpowiednie działania.

public partial class USER

{

public USER()

{

this.USER\_NOTIFICATION = new HashSet<USER\_NOTIFICATION>();

}

public int ID { get; set; }

public string Name { get; set; }

public string FirstName { get; set; }

public string SecondName { get; set; }

public string PhoneNumber { get; set; }

public int UserTypeID { get; set; }

public Nullable<int> TeacherID { get; set; }

public string AspNetUserID { get; set; }

public virtual AspNetUsers AspNetUsers { get; set; }

public virtual TEACHER TEACHER { get; set; }

public virtual ICollection<USER\_NOTIFICATION> USER\_NOTIFICATION { get; set; }

public virtual USER\_TYPE USER\_TYPE { get; set; }

}

Reprezentacja tabeli USER przy pomocy klasy Entity Framework.

Dzięki kluczom obcym istnieje dostęp do odpowiednich danych za pomocą właściwości, które reprezentują te klucze obce.

Dla przykładu, jeśli został już pobrany użytkownika o danym ID to jednocześnie zostały od razu pobrane dane dla tego użytkownika powiązane z tabelami TEACHER, USER\_NOTIFICATION, AspNetUsers i USER\_TYPE. Entity Framework umożliwia to bez pisania dodaktowych zapytań.

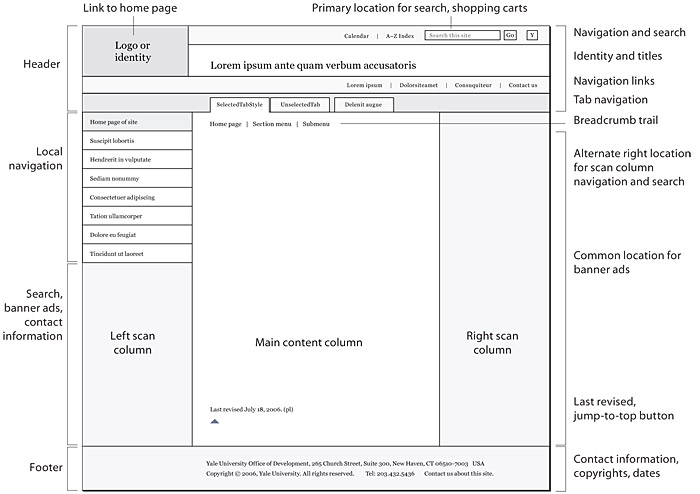
### Widok

Widok jest warstwą prezentacyjną. Dane, które reprezentowane sa przez warstwę modelu muszą być w jakiś sposób wyświetlane użytkownikowi, w postaci tabel, wykresów czy innych form prezentacji danych. W tym celu tworzy się widok czyli inaczej interfejs użytkownika. Interfejs użytkownika jest najważniejszym aspektem każdej aplikacji, jest częścią którą użytkownik widzi już od pierwszego uruchomienia oraz z którą wchodzi w interakcję przez cały okres jej użytkowania. Użytkownicy znacznie częściej oceniają aplikację po jej wyglądzie niż po jakości działania. Użytkownik mając tą samą aplikację działającą bardzo szybko ale z ubogim interfejsem użytkownka i działającą wolniej z bogatym interfejsem, wybierze tą z lepszym interfejsem. Z tego powodu powstały specjalne szkoły, które nauczają technik tworzenia interfesjów użytkownika, a ofert pracy dla tzw UI Designerów (projektantów interfejsów użytkownika) jest coraz więcej.

Przed przystąpieniem do projektowania logiki systemu, został zaprojektowany interfejs użytkownika. Pomimo, że głównym celem pracy nie bylo zaprojektowanie interfejsu a rozwiązanie problemu logiki biznesowej, to interfejs użytkownika nadal jest jej nieodłącznym elementem.

Istnieje kilka podstawowych wzorów interfejsów aplikacji webowych, które prezentują najlepszy układ komponentów, tak aby użytkownik odczuwał jak największy komfor z użytkowania go. Każdy z tych wzorów opiera się na jednym kanonicznym schemacie, który istnieje od początku stron internetowych. Oczywiście nie jest to przymusowe i osoba o odpowiedniej wiedzy i doświadczeniu, może samemu zaprojektować bardzo dobry interfejs.

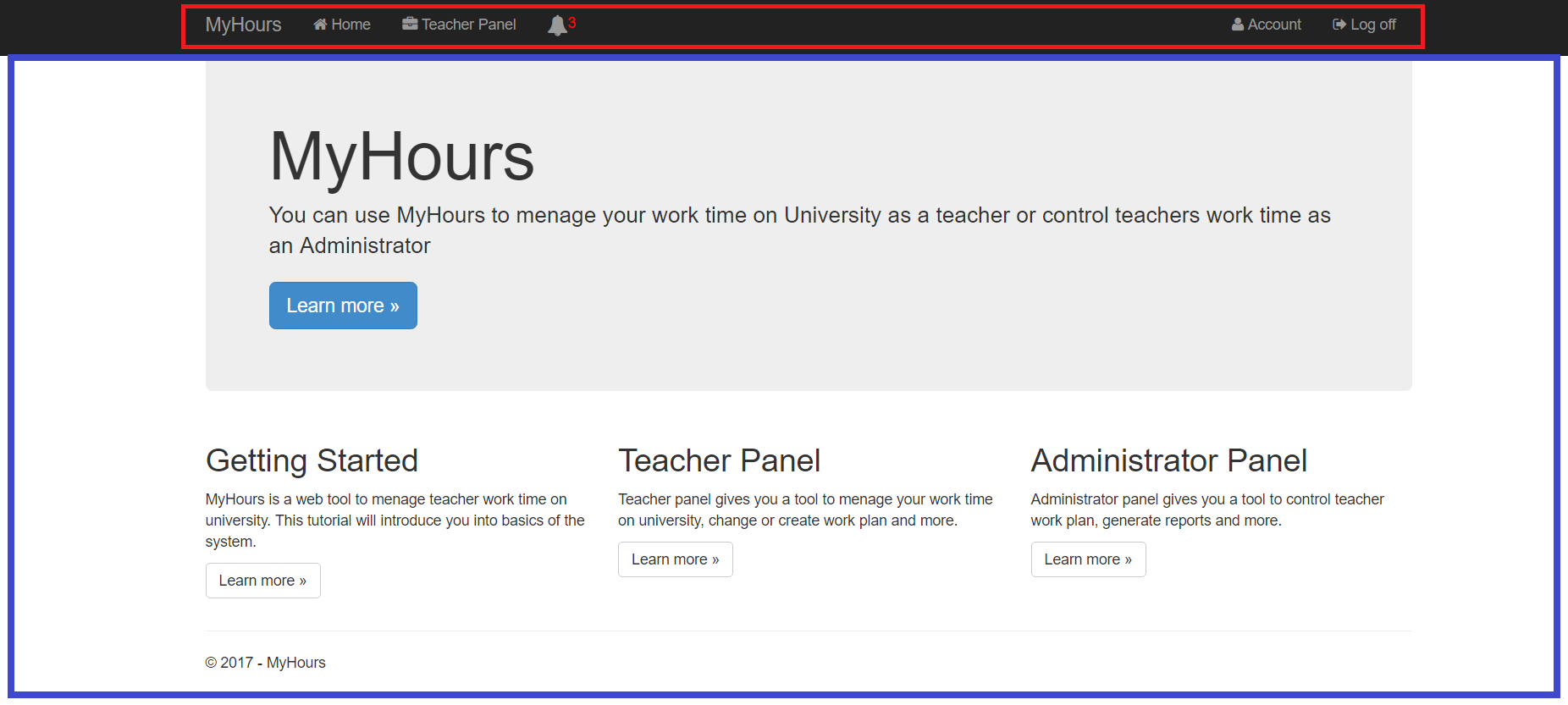
Na poniższym zdjęciu został umieszczony podstawowy wzorzec, który wykorzystywany jest przy projektowaniu interfejsów stron internetowych.



Rys. 11 : Podstawowy układ strony internetowej. [15]

W aplikacji równierz został wykorzystany wyżej wspomniany wzorzec, jednak ze względu na ilość funkcjonalności, menu użytkownika zostało ograniczone zaledwie do górnego panelu.

Na poniższym zrzucie ekranu przedstawiona jest strona główna systemu. Kolorem czerwonym został zaznaczony górny panel kontrolny użytkownika, natomiast kolorem niebieskim główną zawartość strony.



Rys. 12 : Strona główna systemu

Na widoki systemu składają się pliki z rozszerzeniem .cshtml. Każdy z plików odpowiada za inny widok. W projekcie znajdują się również tzw. widoki częściowe, czyli pomniejsze widoki, które są używane w innych widokach, może to być np jedna tabela która jest częścią jakiegoś większego widoku. Oprócz widoków częściowych są równierz widoki współdzielone przez wszystkie widoki w całej aplikacji. W przypadku prezentowanego systemu, niektóre widoki widoczne są tylko dla nauczyciela (Panel Nauczyciela) lub tylko dla administratora (Panel Administratora), widoki współdzielone wykorzystywane są nawet przez te widoki. Przykładem takiego widoku współdzielonego jest panel górny użytkownika zaznaczony kolorem czerwonym na Rys. 12 : Strona główna systemu. Rozwiązanie to pomogło uwspólnić logikę działania oraz sposób wyświetlania powiadomień dla każdego typu użytkownika.

### Kontroler

# Implementacja kluczowych rozwiązań

# Podsumowanie

# Bibliografia

1. Griffiths I.: C# 5.0 Programming. O’Reilly Media, Inc. USA:CA 2013.
2. Freeman A.: Pro ASP.NET MVC 5. Apress. USA:CA 2013.
3. Wikipedia - Common Language Infrastructure *https://en.wikipedia.org/wiki/Common\_Language\_Infrastructure* (12.28.2016)
4. Tutorialized – Fundamentals of an MVC Framework *http://www.tutorialized.com/tutorial/Fundamentals-of-an-MVC-Framework/81946* (12.29.2016)
5. PinalDave - *http://www.pinaldave.com/blogfolder/SQL\_Server.png* (12.20.2016)
6. Microsoft Developer Network – ADO.NET *https://msdn.microsoft.com/pl-pl/library/e80y5yhx(v=vs.110).aspx* (12.21.2016)
7. Entity Framework Tutorial – What is Entity Framework *http://www.entityframeworktutorial.net/what-is-entityframework.aspx* (12.15.2016)
8. TutorialsTeacher – LINQ Tutorial *http://www.tutorialsteacher.com/linq/linq-tutorials* (12.17.2016)
9. Microsoft – Moving Java Application to .Net https://msdn.microsoft.com/en-us/library/ms973842.aspx (02.05.2017)
10. Tech Strikers - .Net Common Language Runtime http://www.techstrikers.com/DotNet/dotnet-common-language-runtime.php (02.05.2017)
11. Eduonix – Understanding Syntax in C# Programming Lanugage https://www.eduonix.com/blog/system-programming/understanding-syntax-in-c-programming-language/ (02.05.2017)
12. Microsoft – Web Server (IIS) https://technet.microsoft.com/en-us/library/cc753433(v=ws.10).aspx (02.05.2017)
13. SlidePlayer – ADO.NET http://slideplayer.es/slide/1445039/ (02.05.2017)
14. plukasiewicz.net – C# - Entity Framework http://plukasiewicz.net/Artykuly/EntityFramework (02.05.2017)
15. Web Style Guide – Interface Design http://webstyleguide.com/wsg3/4-interface-design/3-interface-design.html (02.06.2017)