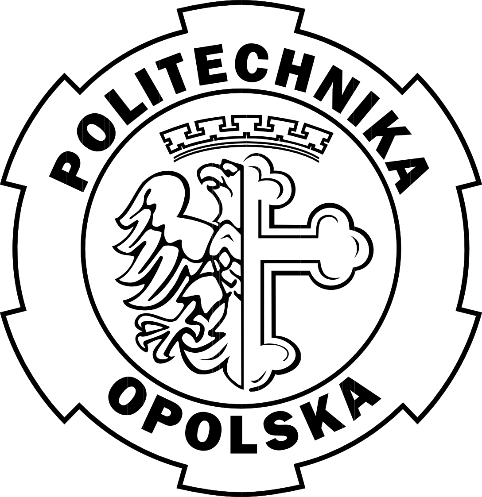
**Politechnika Opolska**

Wydział Elektrotechniki, Automatyki i Informatyki

Instytut Automatyki i Informatyki



**PRACA DYPLOMOWA**

Studia pierwszego stopnia

stacjonarne

Kierunek studiów

INFORMATYKA

**PROJEKT I REALIZACJA AUTORSKIEGO SYSTEMU**

**ZARZĄDZANIA TREŚCIĄ W ŚRODOWISKU ASP.NET MVC.**

|  |  |  |
| --- | --- | --- |
| Promotor:  dr inż. Anna ZATWARNICKA |  | Pracę wykonał:  Patryk LOTZWI  Numer indeksu: 84078 |

Opole, styczeń 2016

Spis treści

[1. WSTĘP 5](#_Toc437193566)

[2. CEL I ZAKRES PRACY 6](#_Toc437193567)

[3. TECHNOLOGIA ASP.NET 7](#_Toc437193568)

[3.1. Historia 7](#_Toc437193569)

[3.2. ASP.NET 5 7](#_Toc437193570)

[4. NARZĘDZIA I FRAMEWORKI 8](#_Toc437193571)

[4.1. Visual Studio 8](#_Toc437193572)

[4.2. Entity Framework 8](#_Toc437193573)

[4.3. Identity 8](#_Toc437193574)

[4.4. Unity 8](#_Toc437193575)

[5. WZORCE PROJEKTOWE I ARCHITEKTONICZNE 9](#_Toc437193576)

[5.1. MVC 9](#_Toc437193577)

[5.2. Repository 9](#_Toc437193578)

[5.3. Service 10](#_Toc437193579)

[5.4. Unit of Work 10](#_Toc437193580)

[5.5. Dependency Injection 10](#_Toc437193581)

[6. HOSTING 11](#_Toc437193582)

[6.1. Microsoft Azure 11](#_Toc437193583)

[6.2. Podejście klasyczne 11](#_Toc437193584)

[6.3. Docker 11](#_Toc437193585)

[7. FRONTEND 12](#_Toc437193586)

[7.1. AngularJS 12](#_Toc437193587)

[7.2. Bootstrap 12](#_Toc437193588)

[7.3. Pozostałe narzędzia 12](#_Toc437193589)

[8. TESTOWANIE 13](#_Toc437193590)

[9. PROJEKT APLIKACJI 14](#_Toc437193591)

[9.1. Wymagania funkcjonalne 14](#_Toc437193592)

[9.2. Diagram ERD 14](#_Toc437193593)

[9.3. Przypadki użycia 14](#_Toc437193594)

[10. WNIOSKI I UWAGI 15](#_Toc437193595)

[11. LITERATURA 16](#_Toc437193596)

# WSTĘP

Wraz z rozwojem technologii internetowych coraz więcej osób posiada własną stronę internetową. Celów jest wiele – począwszy od aspektów biznesowych, jak np. promocja własnej firmy, reklama świadczonych usług, poprzez różnego rodzaju serwisy informacyjne, aż po cele rozrywkowe takie jak portale społecznościowe i blogi.

Zakładając własną stronę użytkownik nie musi posiadać specjalistycznej wiedzy. Wystarczy podstawowa umiejętność obsługi komputera, przeglądarki, oraz podstawowe rozeznanie w obsłudze aplikacji webowych. Obecne technologie pozwalają na „wyklikanie” własnej strony w parę minut. Do wybory jest sporo darmowych szablonów, zwykle zmieniane jest tylko logo użytkownika i treść.

Prostota instalacji i obsługi własnej strony ciągnie za sobą pewne problemy. Z reguły im prostszy system, tym mniejszy wpływ użytkownika na zarządzanie stroną. W takiej sytuacji pomocne stają się systemy CMS – Content Management System. Są to zaawansowane aplikacje webowe, które pozwalają na tworzenie własnej strony, rozbudowywanie jej o nowe funkcjonalności, zarządzanie użytkownikami itp. Obsługa tego typu systemu wymaga już znacznie większego rozeznania osoby tworzącej stronę. Czasami systemy CMS okazują się za trudne w obsłudze dla zwykłych użytkowników. O ile dodawanie nowych funkcjonalności w postaci modułów nie wymaga jeszcze specjalistycznej wiedzy, to takie czynności jak edycja szablonu wymaga już znajomości języka HTML.

Sztuką jest stworzyć system, który będzie jednocześnie przystępny dla średnio doświadczonych użytkowników, a który będzie pozwalał tym bardziej zaawansowanych lub programistom w łatwy sposób go rozbudowywać. Istnieje wiele technologii, które pozwalają na tworzenie zaawansowanych aplikacji webowych. Jedną z nich jest ASP.NET w oparciu o model projektowy MVC. Struktura tego wzorca pozwala rozdzielić warstwę prezentacji od warstwy danych, jak również od logiki aplikacji, co w przypadku systemu CMS wydaje się bardzo użyteczne. Dlatego niniejsza praca opiera się właśnie na tej technologii.

# CEL I ZAKRES PRACY

Celem pracy jest stworzenie autorskiego systemu zarządzania treścią (CMS) w technologii ASP.NET MVC. Na rynku istnieje wiele rozwiązań w postaci systemów CMS, jednak większość z nich jest oparta o technologię PHP. Z kolei znaczna część systemów zarządzania treścią stworzonych w technologii ASP.NET jest płatna. Istnieje zatem zapotrzebowanie na tego typu system, który byłby dostępny na licencji Open Source.

Praca obejmuje stworzenie systemu, który będzie pozwalał na tworzenie, edycję i rozbudowę własnej strony internetowej. Poszczególne funkcjonalności będą zrealizowane w postaci modułów. Użytkownik będzie mógł instalować nowe moduły, a programiści będą mieli możliwość tworzenia własnych rozszerzeń. Kod całego systemu będzie otwarto-źródłowy, co pozwoli programistom lepiej zrozumieć jego działanie i może być przydatne przy tworzeniu zaawansowanych modułów. Praca obejmuje także przygotowanie i wdrożenie podstawowego szablonu, który będzie dostępny dla użytkownika od razu po instalacji systemu i który może być podstawą dla nowo tworzonej strony. Oprócz tego podstawowa wersja systemu będzie zawierała także moduły, które umożliwią m. in. wstawianie i kategoryzowanie treści, tworzenie galerii, sliderów, formularzy kontaktowych.

System będzie dostępny wraz z testami jednostkowymi, które będą informować o prawidłowym działaniu rdzenia systemu w przypadku jego modyfikacji. Projekt systemu będzie także zawierał modele UML, m. in. wymagania funkcjonalne, diagram związków encji oraz diagram przypadków użycia.

# TECHNOLOGIA ASP.NET

## Historia

ASP.NET to część platformy .NET pozwalająca na tworzenie dynamicznych stron, aplikacji i serwisów webowych. Pierwsza wersja tego frameworka pojawiła się w 2002 roku jako następstwo klasycznego ASP (Active Server Pages). Zawierała się w wydanym wtedy .NET Framework 1.0. Z początku tworzenie aplikacji na tę platformę opierało się na tzw. Web Forms, które z czasem zostały zastąpione przez ASP.NET MVC. To drugie, nowsze podejście z wykorzystaniem wzorca architektonicznego Model-View-Controller jest używane po dzień dzisiejszy.

## ASP.NET 5

Obecnie aktualna stabilna wersja ASP.NET to 4.6. Wersja 5 została zapowiedziana podczas konferencji Build 2014 w San Francisco. Zawiera ona duże zmiany względem poprzedniej wersji. Nowy ASP.NET oparty jest na .NET Core – wersji która działa nie tylko na Windows, ale także na systemach Linuksowych oraz Apple Mac. Platforma stała się także otwarto-źródłowa. W nowej wersji poszczególne moduły można doinstalować za pomocą pakietów nuget. Aplikacja może być hostowana nie tylko na IIS (Internet Integration Services), jak to było w poprzednich wersjach, ale umożliwia także tzw. self-hosting – można w prosty sposób napisać własną implementację serwera http. Możliwe jest też hostowanie aplikacji w kontenerach na platformie Docker.

Niniejszy projekt został napisany w oparciu o ASP.NET 5. Aplikacja bazuje na wersji RC (release candidate) nowego frameworka. Wydanie to jest stabilne i posiada rekomendację oraz wsparcie techniczne firmy Microsoft. Finalna wersja 5 ma zostać wydana w I kwartale 2016 roku.

# NARZĘDZIA I FRAMEWORKI

## Visual Studio

Microsoft Visual Studio to środowisko programistyczne do tworzenia aplikacji na platformę .NET. Składa się z zaawansowanego edytora z obsługą podpowiedzi (IntelliSense), kolorowania składni oraz licznych ułatwień dla programisty (np. Code Lens – można w prosty sposób przeskoczyć do implementacji danej metody, jak również podejrzeć kto ostatnio dokonywał zmian na danym fragmencie kodu w przypadku pracy zespołowej). Visual Studio posiada również kompilator i debugger, narzędzia do pracy z bazą danych oraz wiele więcej mniejszych narzędzi, które ułatwiają pracę z kodem. Istnieje możliwość instalacji rozszerzeń, które jeszcze bardziej wzbogacają środowisko.

## Entity Framework

Do mapowania obiektowo-relacyjnego aplikacja używa Entity Framework. Jest to znany i powszechnie używany w aplikacjach ASP.NET framework ORM. W projekcie użyto podejścia „Code-First”. Polega ono na tym, że w pierwszej kolejności tworzone są klasy POCO (Plain Old CLR Object), które odwzorowują encje, następnie na ich podstawie generowane są tabele w bazie. Do bardziej zaawansowanych relacji można użyć mechanizmu fluent api, który pozwala na ręczne mapowanie relacji.

## Identity

Aplikacja korzysta z frameworka Identity służącego do autentykacji użytkowników. Biblioteka ta jest obecnie integralną częścią .NET Framework i jest szeroko wykorzystywana zarówno w małych projektach, jak i tych klasy Enterprise. Autoryzacja opiera się na na loginie i haśle użytkownika. Dane są przechowywane w 5 tabelach:

AspNetUsers – dane użytkowników,

AspNetRoles – role

AspNetUserLogins – loginy użytkowników

AspNetUserClaims – uprawnienia użytkowników

AspNetUserRoles – role użytkowników

Jak można wywnioskować z powyższej rozpiski tabel, każdy użytkownik może mieć przypisaną rolę. Dodatkowo niezależnie od roli może posiadać specjalne uprawnienia. Można też przypisać kilka loginów do jednego użytkownika.

## Unity

W celu implementacji wzorca DI użyto frameworka Unity. Jest to część pakietu Enterprise Library firmy Microsoft, który jest powszechnie używany w projektach klasy Enterprise. Kontener pozwala na mapowanie interfejsów z klasami, jak również na użycie konkretnej implementacji danej klasy (wzorzec Singleton).

# WZORCE PROJEKTOWE I ARCHITEKTONICZNE

Wzorce projektowe i architektoniczne przyspieszają i ułatwiają proces wytwarzania oprogramowania. Nie wyznaczają one ściśle określonej metodyki, natomiast ukazują pewną ścieżkę, którą można podążać tworząc konkretne rozwiązanie i dostosowując je do własnych potrzeb. W niniejszej aplikacji zostało użytych kilka wzorców, które opisano w kolejnych punktach.

## MVC

Wzorzec architektoniczny MVC (Model-View-Controller) oddziela warstwę widoku, logiki oraz modelu danych systemu. Pomaga to w znaczący sposób przy tworzeniu bardziej złożonych systemów, gdyż kod jest wtedy znacznie bardziej czytelny. Podejście to pozwala uniknąć tzw. „spaghetti code”, gdzie większość kodu jest pisana strukturalnie i trzeba przedzierać się przez kolejne instrukcje warunkowe żeby zrozumieć jego działanie.

Dane są reprezentowane przez klasy POCO (Plain Old CLR Objects), które są analogicznym odpowiednikiem koncepcji POJO (Plain Old Java Objects) w języku Java zapoczątkowanej przez Martina Fowlera. Obiekty tych klas reprezentują dane w bazie danych, a same klasy są mapowane na tabele. Pojedynczy obiekt tego typu najczęściej zawiera tylko propercje, gdzie każda reprezentuje konkretną kolumnę w tabeli.

W części logicznej wykorzystuje się kontrolery. Kontroler jest to klasa która zawiera metody służące do obsługi żądań http. Każde metoda odpowiada konkretnemu adresowi url. Adresy są mapowane na podstawie zdefiniowanych ustawień routingu oraz poprzez konwencję nazw. Kontroler zawiera logikę aplikacji, nie powinien natomiast zawierać logiki biznesowej. W celu implementacji logiki biznesowej należy użyć dodatkowej warstwy abstrakcji. Metody w kontrolerze zwracają widok, przekazując do niego odpowiedni model danych.

Widoki służą do graficznej prezentacji interfejsu użytkownika. Pojedynczy widok składa się z pliku .cshtml, który jest kompilowany w postać natywnego kodu html zrozumiałego dla przeglądarki internetowej. Plik .cshtml oprócz konstrukcji strony zawiera elementy języka C#, które służą do umiejscowienia w nim danych z modelu. Wykorzystuje się w tym celu tzw. Razor Syntax. Jest to składnia, w której po umiejscowieniu znaku @ kod jest interpretowany jako język C#.

## Repository

Wzorzec projektowy Repository (Repozytorium) służy do wydzielenia warstwy abstrakcji odpowiedzialnej za operacje CRUD (Create, Read, Update, Delete) na bazie danych. Z reguły każda encja posiada swoje własne repozytorium. Repozytorium powinno być niezależne od konkretnej bazy danych, dlatego wykorzystuje się interfejsy, które standaryzują dostępność poszczególnych metod dla różnych implementacji. Ponieważ wiele metod w poszczególnych repozytoriach się powtarza, dobrym pomysłem jest utworzenie generycznego interfejsu bazowego. Dzięki temu można uniknąć powtarzalności kodu, zgodnie z ideologią DRY (Don’t Repeat Yourself).

## Service

Jak już zostało wspomniane w punkcie 5.1, kontroler nie powinien zawierać logiki biznesowej. Jednym z rozwiązań tego problemu jest wzorzec projektowy Service (Serwis). Jest to wydzielona warstwa abstrakcji, która zawiera wyłącznie logikę biznesową. Podobnie jak we wzorcu Repozytorium, zwykle tworzy się osobny serwis dla każdej encji. Serwis do operacji na bazie danych wykorzystuje odpowiadające mu repozytorium. Poszczególne metody serwisu mogą wykonywać bardziej złożone czynności, np. mogą zwracać konkretne rekordy z bazy w zależności od podanych uwarunkowań.

## Unit of Work

Użycie wzorców Repozytorium i Serwis znacząco ułatwia pracę z bazą danych, natomiast może spowodować niespójność danych. Może dojść do sytuacji, gdzie jedno repozytorium zapisze dane, a w międzyczasie inne też coś zmodyfikuje i pewne zmiany zostaną utracone. Aby temu zapobiec należy użyć wzorca Unit of Work. Wzorzec ten zapewnia transakcyjność operacji na bazie przy korzystaniu z wielu repozytoriów. Dzięki temu stan bazy jest zapisywany tylko raz. Dodatkowo wpływa też korzystnie na optymalizację, gdyż wszystkie zmiany są wysyłane i przetwarzane za jednym razem, a nie pojedynczo. Najprostszą formą implementacji tego wzorca jest utworzenie klasy, która będzie metodę Commit służącą do zatwierdzenia zmian. Dodatkowo klasa ta powinna implementować interfejs IDisposable, co zapewni możliwość użycia jej w strefie o ograniczonym zasięgu, osobno dla każdej metody serwisu.

## Dependency Injection

Wzorzec projektowy Dependency Injection (Wstrzykiwanie zależności) korzysta z paradygmatu IoC (Inversion of Control – odwrócenie sterowania) w celu uniezależnienia poszczególnych warstw abstrakcji od ich konkretnej implementacji. Architektura systemu, w którym wykorzystuje się DI opiera się na interfejsach, a poszczególne implementacje są wstrzykiwane poprzez kontener. Kontener może być plikiem konfiguracyjnym XML, może być też tworzony w specjalnej klasie konfiguracyjnej.

# HOSTING

## Microsoft Azure

Microsoft Azure to platforma chmurowa pozwalająca m. in. na hostowanie aplikacji webowych, tworzenie maszyn wirtualnych w chmurze, baz danych oraz wiele innych usług opartych o cloud computing. Usługa jest oparta na modelu subskrypcji – rozliczany jest zużycie zasobów poszczególnych maszyn i usług, czyli płaci się za to co się zużywa. Usługi są skalowalne –w dowolnym momencie można np. powiększyć bazę danych, dodać dodatkowe procesory lub pamięć do maszyny wirtualnej. Operacja tego typu trwa kilka minut. Z tego powodu wiele firm korzysta z tej platformy w komercyjnych projektach, gdyż jest to bardzo wygodne, poza tym można zaoszczędzić na własnej infrastrukturze.

W niniejszym projekcie został użyty hosting, jak również baza danych na platformie Azure. Zostały też utworzone maszyny wirtualne dla Dockera oraz na potrzeby implementacji Continuous Integration.

## Podejście klasyczne

Aplikacje ASP.NET można hostować na serwerze IIS, lub za pomocą self-hostingu ( w przypadku nowego ASP.NET 5). Serwer IIS wymaga systemu Windows Server. Aplikacja może zostać wdrożona automatycznie poprzez Visual Studio, jak również ręcznie za pomocą protokołu ftp. W przypadku pierwszej opcji baza danych sparowana z usługą hostingu jest automatycznie podpinana, w podejściu z ftp trzeba ręcznie skonfigurować połączenie poprzez edycje pliku web.config.

## Docker

Docker jest platformą, która pozwala na hostowanie aplikacji w tzw. kontenerach. Kontener jest to wydzielony obszar w systemie, który jest izolowany, posiada jednak dostęp do zasobów procesora, pamięci czy połączenia sieciowego. Docker powstał jako alternatywa dla klasycznego hostowania aplikacji, gdzie zwykle na jednej fizycznej maszynie jest uruchomionych kilka maszyn wirtualnych, z których każda hostuje daną aplikację. Docker pozwala uniknąć warstwy wirtualizacji, gdyż na jednym wszystkie aplikacje działają na fizycznej maszynie, są jednak podobnie jak maszyny wirtualne izolowane od siebie. Platforma ta świetnie nadaje się do mikroserwisów – aplikacji podzielonych na mniejsze jednostki, z których każda pełni określoną funkcję.

ASP.NET 5 pozwala na hostowanie na Dockerze. W tym celu należy pobrać specjalnie przygotowany kontener bazowy i użyć go jako podstawy dla własnej aplikacji.

# FRONTEND

## AngularJS

## Bootstrap

## Pozostałe narzędzia

# TESTOWANIE

# PROJEKT APLIKACJI

## Wymagania funkcjonalne

## Diagram ERD

## Przypadki użycia

# WNIOSKI I UWAGI

# LITERATURA

1. Żydzik K., Rak T., *C# 6.0 i MVC 5. Tworzenie nowoczesnych portali internetowych*, Helion, Gliwice 2015
2. Kalbarczyk D., Kalbarczyk A., *AngularJS. Pierwsze kroki*, Helion, Gliwice 2015
3. Gamma E., Helm R., Johnson R., Vlissides J., *Wzorce projektowe. Elementy oprogramowania obiektowego wielokrotnego użytku*, Helion, Gliwice 2010
4. Martin R. C., *Czysty kod. Podręcznik dobrego programisty*, Helion, Gliwice 2014
5. Hunt A., Thomas D., *Pragmatyczny programista. Od czeladnika do mistrza*, Helion, Gliwice 2011

Materiały Online:

<http://enterprisecraftsmanship.com/2015/04/13/dto-vs-value-object-vs-poco/>

<http://www.pzielinski.com/?p=281>

<http://www.codeproject.com/Articles/581487/Unit-of-Work-Design-Pattern>

<http://techbrij.com/generic-repository-unit-of-work-entity-framework-unit-testing-asp-net-mvc>

<http://www.codeproject.com/Articles/526874/Repositorypluspattern-cplusdoneplusright>