Politechnika Poznańska

Wydział Elektryczny

Instytut Automatyki i Inżynierii Informatycznej

PRACA DYPLOMOWA INŻYNIERSKA

**Michał Suchorzyński**

**System zarządzania treścią dla małych i średnich przedsiębiorstw**



Promotor: dr inż. Adama Meissnera

Poznań, styczeń 2018

**Streszczenie**

W pracy przedstawiono system zarządzania treścią dla małych i średnich przedsiębiorstw. Zadaniem aplikacji jest ułatwienie użytkownikowi tworzenie oraz utrzymywanie witryny internetowej firmy. Aplikacja składa się z czterech modułów: aplikacji administracyjnej, aplikacji webowej, bazy danych, usługi WEB API. Pierwsza część pracy dotyczy charakterystyki systemów zarządzania treścią oraz celu i zakresu pracy. W dalszej części znajduje się opis metodyki realizacji oraz szczegóły dotyczące realizacji aplikacji.

**Abstract**

The work presents a content management system for small and medium enterprises. The task of the application is to facilitate the creation and maintenance of the company's website. The application consists of four modules: administrative application, web application, database, WEB API service. The first part of the work concerns the characteristics of content management systems as well as the purpose and scope of work. In the following, there is a description of the implementation methodology and details of the implementation of the application.

**Spis treści**

[1. Wstęp 8](#_Toc504763099)

[2. Cel i zakres pracy 10](#_Toc504763100)

[3. Metody modelowania i realizacji 11](#_Toc504763101)

[4. Model systemu 14](#_Toc504763102)

[4.1 Architektura systemu 14](#_Toc504763103)

[4.2 Model bazy danych 15](#_Toc504763104)

[4.3 Moduły systemu 17](#_Toc504763105)

[5. Realizacja 23](#_Toc504763106)

[5.1 Moduł administracyjny 23](#_Toc504763107)

[5.2 Moduł strony WWW 27](#_Toc504763108)

[5.3 Moduł WEB API 28](#_Toc504763109)

[6. Użytkowanie 30](#_Toc504763110)

[7. Podsumowanie 35](#_Toc504763111)

[8. Bibliografia 36](#_Toc504763112)

# Wstęp

System zarządzania treścią *content management system* jest to system informatyczny, który umożliwia użytkownikowi bez znajomości wiedzy technicznej utworzenie, rozwijanie oraz utrzymywanie serwisu WWW. Przeznaczony jest także dla wykfalifikowanym programistą w celu przyspieszenia procesu produkcji strony internetowej. Występuje klika rodzajów systemów zarządzania treścią:

* *content management framework*
* *page-based systems*
* *module-based systems*
* *content object systems*.

Systemy zarządzania treścią składają się najczęściej z dwóch modułów:

* aplikacji do zarządzania treścią, która jest aplikacją administracyjną umożliwiającą użytkownikowi dodawanie , usuwanie i edytowanie zawartości tworzonej strony internetowej
* aplikacja do dostarczania treści , której zadaniem jest wyświetlanie zawartości zaprojektowanego projektu strony internetowej wcześniej stworzonego w aplikacji do zarządzania treścią.

Jednym z najpopularniejszych istniejących systemów zarządzania treścią jest darmowa open sourcowa aplikacja [WordPress](https://pl.wikipedia.org/wiki/WordPress), zbudowana w oparciu na technologiach PHP oraz MySQL. W początkowych etapach funkcjonowania była używany jako platforma blogowa. Zawiera wiele rozszerzeń dostarczających użytkownikowi nowe elementy strony WWW, lub zmieniających aspekty stylistyczne powstałej witryny. WordPress jest najczęściej stosowany do tworzenia zwykłych stron lub stron firmowych. Kolejnym systemem dostępnym na rynku jest aplikacja [Joomla!](https://pl.wikipedia.org/wiki/Joomla!). Również jest to darmowy open source system kontroli wersji oparty na technologiach PHP oraz MySQL [5]. Została utworzona z wykorzystaniem wzorca architektonicznego model-widok-kontroler. Jedną z głównych zalet tej aplikacji jest dostępność w wielu wersjach językowych.

Wybór tematu był ukierunkowany chęcią stworzenia systemu informatycznego, dzięki któremu użytkownik będzie miał możliwość w prosty sposób utworzyć oraz utrzymywać serwis WWW. Najważniejszym aspektem w trakcie projektowania systemu, było aby aplikacja nie wymagała od przyszłego użytkownika wiedzy technicznej na temat funkcjonowania tworzonego serwisu. W celu dalszego rozwijania systemu został on zaprojektowany tak aby proces dalszego rozbudowywania istniejących funkcji oraz dodawania nowych był prosty w realizacji.

# Cel i zakres pracy

Celem pracy jest skonstruowanie systemu zarządzania treścią przeznaczonych dla małych i średnich przedsiębiorstw. Zadaniem systemu jest tworzenie prostych witryn WWW służących jako wizytówki firm. Umożliwia on użytkownikowi w prosty sposób zaprojektowanie menu oraz podstron witryny internetowej. Każdy element strony może być projektowany od zera, poprzez nadawanie mu własnych kolorów, rozmiarów i innych właściwości. Możliwe jest również projektowanie wielu witryn w jednym miejscu, aby użytkownik w jednej aplikacji mógł tworzyć i administrować wiele projektów witryn niepowiązanych ze sobą. Po utworzeniu projektu produkt może być w dowolny sposób modyfikowany lub rozszerzany poprzez powiększanie istniejących elementów lub dodawanie nowych podstron. Głównymi funkcjami realizowanymi przez system są:

* tworzenie nowych projektów witryn lub usuwanie już istniejących
* dodawanie nowych podstron i menu do istniejących projektów lub usuwanie już istniejących
* zapisywanie zawartości projektów w bazie danych
* projektowanie zawartości menu
* projektowanie zawartości poszczególnych podstron
* dodawanie i usuwanie kontrolek do siatki kontrolek
* zmienianie rozmieszczenia kontrolek na siatce kontrolek
* zmienianie zawartości oraz wyglądu położonych kontrolek na siatce kontrolek
* dodawanie akcji przejścia miedzy poszczególnymi podstronami
* wyświetlanie witryny internetowej ustawionej w pliku konfiguracyjnym projektu lub wyświetlanie innych projektów poprzez zmianę parametrów w adresie URL.

W kolejnym rozdziale opisano metody modelowania i realizacji projektu. Zawiera on opis technologii wykorzystanych do realizacji poszczególnych modułów. Czwarty rozdział przedstawia model zaprojektowanego systemu, opisano w nim ogólny sposób działania systemu oraz opis poszczególnych modułów aplikacji. W piątym rozdziale zostały przedstawione szczegóły implementacji aplikacji. Kolejny rozdział opisuje interfejs użytkownika modułu aplikacji administracyjnej oraz przeprowadzone testy systemu. Ostatnie dwa rozdziały poświęcone są podsumowaniu pracy oraz literaturze wykorzystywanej do realizacji pracy.

# Metody modelowania i realizacji

Do utworzenia projektu wykorzystano dwa wzorce architektoniczne. Pierwszym z nich jest wzorzec *model–view–controler*, w którym struktura aplikacji jest dzielona na trzy główne warstwy:

* modelu w której przechowywane są elementy odpowiedzialne za implementacje logiki dla aplikacja. Często elementy modelu wykorzystywane są do odczytu i zapisu stanu aplikacji w bazie danych
* widoku przechowująca elementy odpowiedzialne za wyświetlanie interfejsu użytkownika. Najczęściej ten interfejs jest tworzony na podstawie stanu danych modelu
* kontrolera w której przechowywane są elementy odpowiedzialne za interakcje aplikacji z użytkownikiem. Obsługują model oraz decydują który widok i z jaką zawartością zostanie wyświetlony użytkownikowi.

Kolejnym rozpatrywany wzorzec architektoniczny nosi nazwę *model–view–viewmodel* [7], jest on wzorcem w którym struktura aplikacji jest dzielona na trzy główne warstwy:

* modelu w której przechowywane są elementy odpowiedzialne za implementacje logiki dla aplikacja. Często elementy modelu wykorzystywane są do odczytu i zapisu stanu aplikacji w bazie danych
* widoku przechowująca elementy odpowiedzialne za wyświetlanie interfejsu użytkownika. Najczęściej ten interfejs jest tworzony na podstawie stanu danych modelu
* modelu widoku która jest abstrakcją widoku aplikacji. W tej warstwie przechowywane są elementy odpowiedzialne za wiązanie danych modelu z wyświetlanymi wartościami użytkownikowi.

Głównym językiem wykorzystanym do implementacji poszczególnych modułów jest wieloparydygmatowy język programowania C# obejmujący silne typowanie, imperatywne, deklaratywne, funkcjonalne, ogólne, obiektowe i zorientowane komponentowo dziedziny programowania. Najczęściej jest wykorzystywany do programowania obiektowego opartego na klasach. Został stworzony i nadal jest rozwijany przez firmę Microsoft. Najnowsza wersja języka to C # 7.2, która została wydana w 2017 roku. Moduł administracyjny oparty jest na technologii Windows Presentation Foundation, która udostępnia model programistyczny umożliwiający programiście tworzenie nowoczesnych aplikacji desktopowych na systemy operacyjne Windows. Została wprowadzono na przez firmę Microsoft w roku 2006 jako przyszły następca dotychczasowego modelu Windows Forms. Umożliwia on pisanie aplikacji z wykorzystaniem wzorca projektowego MVVM. Wygląd layoutu modułu administracyjnego zbudowany jest w języku XAML bazującym na składni XML stworzonym przez firmę Microsoft [3]. W technologii WPF jest wykorzystywany do projektowania widoków aplikacji. Do realizacji moduł strony www został użyty ASP.NET Core. Jest to open source freamowork firmy Microsoft. Został wprowadzony w 2016 roku jako nowa generacja freamoworku ASP .NET. Umożliwia tworzenie aplikacji Web z wykorzystaniem wzorca projektowego MVC [1]. Dodatkowo w module prezentującym witrynę internetową dołączone zostały biblioteki Bootstrap zawierające zestaw narzędzi HTML, CSS i JS ułatwiającą tworzenie interfejsów użytkownika serwisów WWW. Umożliwiają one tworzenie responsywnych stron internetowych oraz wykorzystywanie wcześniej zaprojektowanych elementów widoku. Dodatkowo do obsługi specyficznych zdarzeń wykonujących się na stronie zostały napisane dodatkowe funkcje w język JavaScript, który jest skryptowym językiem programowania wspomagającym interakcje między użytkownikiem a stroną internetową. Jego wykorzystanie daję możliwość wizualnego urozmaicenia witryny poprzez dodanie animacji oraz dynamicznej zmiany zawartości layoutu bez konieczności przeładowywania strony. W projekcie został wykorzystany jQuery który jest jeden z wielu freamowork tego języka. Do zamodelowania bazy danych został zastosowany Object/Relational Mapping (O/RM) freamowork,który nosi nazwę Entity Framework Dzięki użyciu tej technologii ułatwiono dostęp i obsługę bazy danych. Pozwala ona zmapować tabele relacyjnej bazy danych do postaci obiektów klas aplikacji, dzięki czemu możliwe jest łatwe dokonywanie uaktualniania zawartości bazy danych. Przesyłane dane między modułami aplikacji są w formacie JSON. JavaScript Object Notation jest formatem wymiany danych w informatyce. Wykorzystuje się go do przesyłania informacji w sytuacjach gdy format przesyłanych danych musi być tekstem na przykład do przekazywania obiektów klasy w komunikacji klient serwer.

Implementacja systemu wymagała użycia dwóch środowisk programistycznych. Pierwszy z nich nazywa się Microsoft Visual Studio. Jest to zintegrowane środowisko programistyczne firmy Microsoft. Służy do tworzenia programów komputerowych, a także stron internetowych, aplikacji internetowych, usług internetowych i aplikacji mobilnych. Umożliwia tworzenie aplikacji z wykorzystaniem obiektowych języków programowania na przykład C#. Dzięki menadżerowi paczek NuGet programista ma łatwy dostęp do dodatkowych bibliotek języka. Kolejnym środowiskiem programistycznym jest narzędzie firmy Microsoft SQL Server Management Studio, w pracy wykorzystano go do wygenerowania schematu używanej w aplikacji bazy danych. Użytkownik tego programu ma możliwość wykorzystania graficznego interfejsu użytkownika w celu zarządzania bazami danych. Aplikacja pozwala zalogować się do serwera SQL aby tworzyć, edytować lub wykonywać inne operacje na bazie danych. Dodatkowo podczas realizacji projektu użyto system kontroli wersji, który umożliwia użytkownikowi przegląd postępów dotychczasowej pracy. Chroni również przed utrata dotychczasowych postępów w pracy. Najpopularniejszymi systemami dostępnymi na rynku są GIT oraz TFS. W ramach projektu wykorzystano system GIT.

Schematy systemu zostały wykonane poprzez użycie UML Unified Modeling Language, który jest językiem pół-formalnym wykorzystywanym do modelowana schematów systemów informatycznych [4]. Jego wykorzystanie umożliwiło zobrazowanie architekturę oraz zasadę działania systemu. Użyto narzędzia Visual Paradigm ułatwiającego modelowanie diagramów UML. W pracy znajdują się diagramy aktywności oraz diagram klas przedstawiający Poza obsługą modelowania zapewnia ono funkcje generowanie raportów oraz kodu.

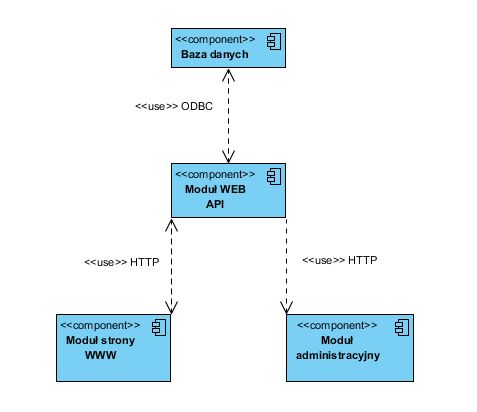
# Model systemu

## Architektura systemu

Do zbudowania systemu została wykorzystana trójwarstwowa architektura. Opiera się ona na rozdzieleniu aplikacji na trzy warstwy:

* warstwę danych
* warstwę logiki biznesowej
* warstwę prezentacji.

Powodem wykorzystania tej architektury byłą chęć wyeliminowania powielania kodu funkcjonalności obsługi bazy danych dla aplikacji administracyjnej i aplikacji klienckiej (modułu strony WWW). Część wykorzystywana do odczytywania i modyfikowania danych zawartych w bazie danych została przeniesiona do z warstwy prezentacji do warstwy logiki biznesowej. Na rysunku 3.1.1 została przedstawiony diagram komponentów systemu oraz w jaki sposób odbywa się komunikacja między modułami.



*Rysunek 3.1.1 Diagram komponentów systemu*

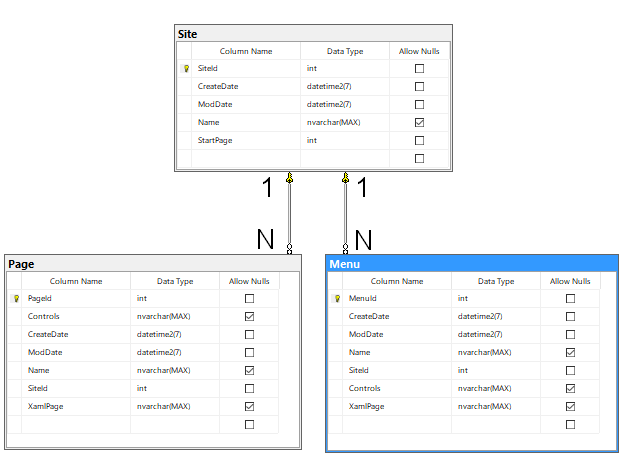
W warstwie danych znajduję się baza danych SQL przechowywująca informację o projektach stron internetowych stworzonych przez użytkownika w module administracyjnym. Obsługa operacji dokonywanych na tych danych odbywa się w warstwie logiki biznesowej, która w projekcie została zrealizowana jako moduł WEB API posiadający funkcje dokonujące zapisów, odczytów i modyfikacji tabel. Rodzaj wykonywanych operacji oraz dokładne działanie jest określane na podstawie requestów otrzymywanych z warstwy prezentacji. Po wykonaniu przez bazę danych zleconych operacji w responsie wysyłana jest informacja zwrotna zawierającego informacje o skutku wykonanej operacji. W warstwie prezentacji znajdują się moduł administracyjny oraz moduł strony WWW. W trakcie działania obie aplikacje wysyłają requesty do warstwy logiki biznesowej. Moduł administracyjny ma możliwość wysyłania zleceń odczytywania, modyfikowania i dodawania nowych rekordów w bazie danych, a moduł strony WWW tylko odczytuje wcześniej zapisane informacje o utworzonym projekcie strony internetowej.

## Model bazy danych

Baza danych wykorzystywana do przechowywania danych wygenerowanych w module administracyjnym składa się z trzech tabel:

* Site zawierającej witryny stworzone w aplikacji administracyjnej
* Page zawierającej podstrony witryn
* Menu zawierające panele Menu witryn.

Między tabelą Site oraz Page występuje relacja 1 do N, umożliwia to tworzenie witryn internetowych składających się z wielu podstron. Identyczna relacja występuje między tabelami Site oraz Menu. Na rysunku 3.2.1 został przedstawiony schemat bazy danych przedstawiający tabele wraz z ich kolumnami. Linie łączące elementy reprezentują powiązania między tabelami.



*Rysunek 3.2.1 Schemat bazy danych*

Baza danych została stworzona za pomocą technologii Entity Framework, wraz z wykorzystaniem podejścia Code-First [2] polegającym na implementacji klas modelu z których następnie generowane są za pomocą mechanizmów freamowork tabele bazy danych wraz z ich relacjami. Przedstawiony poniżej kod źródłowy zawiera implementacje modelu klasy Site reprezentującej pojedynczą witrynę projektowaną w module administracyjnym.

|  |
| --- |
| public class Site  {  [Key]  public int SiteId { get; set; }  public string Name { get; set; }  public DateTime CreateDate { get; set; }  public DateTime ModDate { get; set; }  public int StartPage { get; set; }  public virtual ICollection<Menu> Menus { get; set; }  public virtual ICollection<Page> Pages { get; set; }  } |

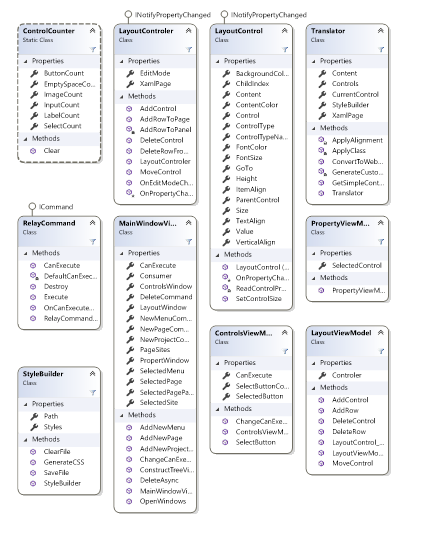
Przedstawiona klasa Site posiada właściwości:

* SiteId reprezentującą klucz główny tabeli. Została ona poprzedzona atrybutem [Key], który w podejściu Code-First oznacza że będzie to kolumna przechowywująca klucz główny
* Name reprezentującą kolumnę przechowywująca nazwę witryny
* CreateDate reprezentującą kolumnę zawierająca datę stworzenia witryny w module administracyjnym
* ModDate reprezentującą kolumnę odpowiadającą za przechowywanie daty ostatniej modyfikacji witryny dokonanej przez użytkownika
* StartPage reprezentującą kolumnę przechowywującą identyfikator podstrony która ma być wyświetlana jako strona startowa
* Menus reprezentująca wiersze tabeli Menu zawierające klucze obce do tego wiersza kolumny Site
* Pages reprezentującą ona wiersze tabeli Page zawierające klucze obce do tego wiersza kolumny Site.

W analogiczny sposób z wykorzystaniem podejścia CodeFirst zostały zamodelowane pozostałe tabele bazy danych czyli Page oraz Menu.

## Moduły systemu

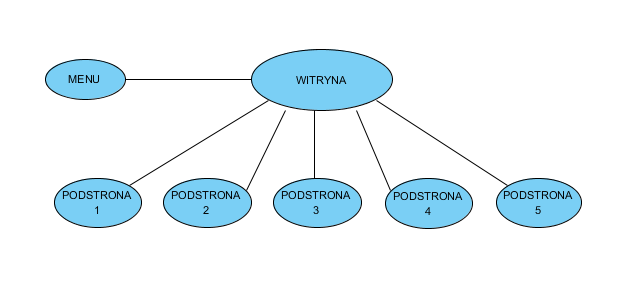
Moduł administracyjny został zrealizowany przy wykorzystaniu wzorca architektonicznego MVVM, jest to aplikacja utworzona poprzez wykorzystanie w technologii WPF. Na rysunku 3.3.1.1 zostały przedstawione klasy wchodzące w skład aplikacji administracyjnej.



Rysunek 3.3.1.1 Klasy aplikacji administracyjnej.

Głównymi funkcjami realizowanymi przez moduł administracyjny są:

1. Tworzenia instancji projektu witryny internetowej, na rysunku 3.3.1.2 przedstawiono schemat modelu budowy pojedynczej witryny internetowej. Widać na nim że pojedyncza witryna może składać się z jednego menu oraz podstron których liczba może być zmieniana dynamicznie.

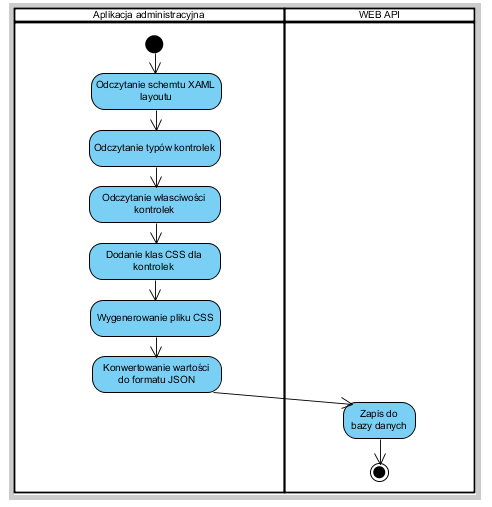


Rysunek 3.3.1.1 Model budowy witryny.

1. Wypełnianie zawartością pojedynczych kontrolek
2. Modelowanie wyglądu pojedynczych kontrolek. Dla utworzonych kontrolek definiowane są właściwości możliwe do edycji, odpowiadają one za:

* kolor tła
* kolor zawartości (dotyczy przycisków i pól do wprowadzania),
* kolor czcionki,
* rozmiar czcionki,
* akcja wykonywana po kliknięciu elementu (dotyczy przycisków i linków),
* wyrównanie elementów w pionie i poziomie,
* liczba zajmowanych kolumn przez element.

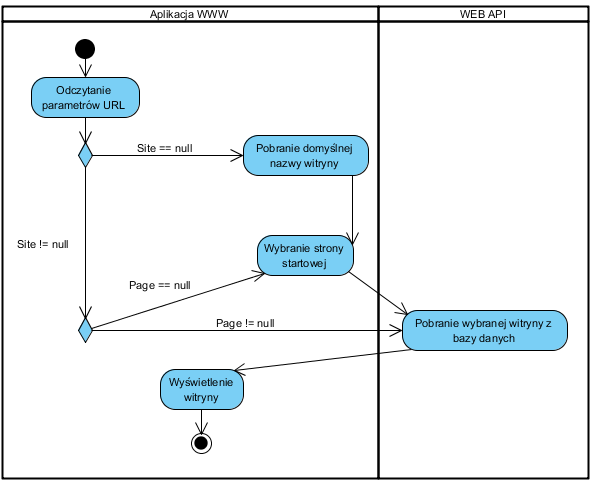
1. Modelowanie rozkładu kontrolek, schemat rozmieszczenia elementów został zastosowany tak jak w technologii Bootstrap [6]. Schemat strony internetowa dzielony jest na wiersze i kolumny. Jeden wiersz zawiera dwanaście kolumn w których mogą być umieszczane kontrolki. Z utworzonego schematu tworzona jest lista kontrolek która następnie jest zapisywana w postaci JSON w bazie danych podczas zapisu layoutu. Rysunek 3.3.1.3 przedstawia algorytm zapisu siatki kontrolek.



Rysunek 3.3.1.1 Kroki zapisu layoutu.

Moduł strony WWW zawiera aplikacje ASP.Net MVC tworząca witrynę internetową na podstawie schematu kontrolek wcześniej utworzonego w module administracyjnym Elementami tego modułu są :

* HomeController jest to kontroler odpowiedzialny za pobranie z modelu i przekazanie do widoku odpowiedniej strony internetowej zamodelowanej w aplikacji administracyjnej. Rysunek 3.3.2.1 przedstawia uproszczony schemat działania wybierania strony internetowej do załadowania:



Rysunek 3.3.2.1 Schemat wybierania aktualnej strony.

Na starcie ładowania witryny są pobierane z adresu URL parametry Site oznaczający nazwę witryny i Page oznaczający nazwę podstrony. Jeśli parametr Site nie istnieje z pliku konfiguracyjnego pobierana jest domyślna nazwa witryny która ma być ładowana na tej instancji projektu i jako podstrona do wyświetlenie ustawiana jest strona startowa. Jeśli parametr Site istnieje lecz parametr Page ma wartość pustą także jako podstrona wybierana jest strona startowa. Jeżeli oba parametry posiadają wartość wybierana jest podstrona o nazwie wartości Page znajdująca się w witrynie wartości Site. Po ustaleniu nazwy witryny i podstrony jest ona pobierana z bazy danych i wyświetlana w przeglądarce

* Index jest to widok który, odpowiada za generowanie layoutu kontrolek aktualnie ładowanej podstrony
* Layout jest również widokiem lecz odpowiada za generowanie layoutu menu aktualnie ładowanej witryny.

Ostatnim modułem systemy jest Web API. Zadaniem tego modułu jest obsługę operacji dokonywanych na bazie danych. Został zrealizowany w technologii WEB API MVC. Elementami tej części systemu są:

* SiteController odpowiadający za obsługę tabeli Site z bazy danych
* PageContrller odpowiadający za obsługę tabeli Page z bazy danych
* MenuController odpowiadający za obsługę tabeli Menu z bazy danych.

W tym module znajdują się również klasy modelu bazy danych oraz kontekst z którego za pomocą Entity Freamowork generowana jest baza danych SQL.

# Realizacja

## Moduł administracyjny

Moduł administracyjny składa się z kilku klas, których zadaniami jest obsługa głównych funkcji modułu. Pierwszą klasą jest LayoutControler odpowiadająca za obsługę siatki kontrolek w czasie modelowania zawartości witryny. Jest to klasa dziedzicząco po interfejsie INotifyPropertyChanged. Właściwościami występującymi w implementacji tego elementu są:

* StackPanel XamlPage przechowująca aktualny schemat XAML
* LayoutControl SelectedControl przechowująca aktualnie wybraną kontrolkę.

Operacje na siatce kontrolek obsługiwane przez klasę LayoutControler odbywają się za pomocą funkcji

* AddRowToPage(selected)która jest metoda dodająca wiersz dwunastokolumnowy do zawartości całej strony. Parametrem tej metody jest wybrana podczas kliknięcia kontrolka. Program na wstępie wyszukuje wiersz w której ona się znajduję, następnie dodawana jest pod nim nowy wiersz zawierający dwanaście pustych przestrzeni
* DeleteRowFromPage(selected)będącą metoda usuwającą wiersz wraz z wszystkimi kontrolkami które się w nim znajdują. Parametrem tej metody jest wybrana podczas kliknięcia kontrolka
* AddControl(toAdd,toDelete)odpowiadającą za dodawanie nowej kontrolkę do strony. Parametr toAdd reprezentuje kontrolkę z domyślnymi ustawieniami wybraną w ControlWindow. Drugi parametr o nazwie toDelete odpowiada wybranemu miejscu na siatce kontrolek. Jeżeli na wybranym obszarze nie znajduje się inny element następuje dodanie nowej kontrolki zajmującej jedną kolumnę
* DeleteControl(selected) jest metoda usuwającą kontrolkę ze strony. Wybrany element przekazywany w parametrze zostaje usuwanięty z siatki kontrolek, a w jego miejsce zostają dodane puste przestrzenie których ilość zależy od ilości zajmowanych kolumn przez wcześniejszą kontrolkę.

Kolejną klasą jest LayoutControl odpowiadająca za obsługę zmian ustawień wybranej kontrolki. Jest to klasa dziedzicząco po interfejsie INotifyPropertyChanged. Właściwościami występującymi w implementacji tego elementu są

* WebControlTypeEnum ControlType przechowująca typ kontrolki
* string ControlTypeName przechowująca nazwę typu kontrolki
* int Size przechowująca liczbę kolumn zajmowanych przez kontrolkę
* object Content przechowująca zawartość kontrolki
* Grid ParentControl przechowująca obiekt rodzica kontrolki na siatce.
* int ChildIndex przechowująca indeks kontrolki który zajmuje wśród dzieci rodzica
* string Value przechowująca wartość kontrolki
* Color? BackgroundColor przechowująca kolor tła kontrolki
* Color? FontColor przechowująca kolor tekstu kontrolki
* Color? ContentColor przechowująca kolor elementów kontrolki
* TextAlignment TextAlign przechowująca wyrównanie tekstu kontrolki
* HorizontalAlignment ItemAlign przechowująca wyrównanie kontrolki w poziomie
* VerticalAlignment VerticalAlign przechowująca wyrównanie kontrolki w pionie
* string GoTo przechowująca rodzaj przejścia po naciśnięciu kontrolki
* double FontSize przechowująca rozmiar czcionki teksty kontrolki

Modyfikacja rozmiaru kontrolki odbywa się za pomaca funkcji SetControlSize(newSize) ustawia ona ilość kolumn które zajmuję kontrolka. Podczas nadawania rozmiaru sprawdzane jest czy nowa wartość jest większa od zera oraz mniejsza od dwunastu, oraz czy położenie elementu oraz rozmieszczenie innych kontrolek pozwala na jego rozszerzenie. Podczas zmniejszania kontrolki następuje zmniejszenie ilości kolumn którą zajmuje a w pozostałe miejsce wstawiane są puste przestrzenie, natomiast gdy zwiększany jest rozmiar algorytm jest odwrotny. Kod źródłowy przedstawia działanie tej funkcji.

|  |
| --- |
| public bool SetControlSize(int newSize)  {  if (\_control == null || newSize <= 0 || newSize == this.Size)  return false;  if (this.ControlType == WebControlTypeEnum.row)  return false;  if (newSize > this.Size)  {  if (this.ChildIndex + newSize > 12)  return false;  for (int i = this.ChildIndex + 1; i < this.ChildIndex + newSize; i++)  {  LayoutControl child = new LayoutControl(\_parentControl.Children[i] as UserControl);  if (child.ControlType != WebControlTypeEnum.emptySpace)  {  return false;  }  }  var colDefinitions = \_parentControl.ColumnDefinitions.Where(x => \_parentControl.ColumnDefinitions.IndexOf(x) >= this.ChildIndex && \_parentControl.ColumnDefinitions.IndexOf(x) < this.ChildIndex + newSize);  \_parentControl.ColumnDefinitions[this.ChildIndex].Width = new GridLength(newSize, GridUnitType.Star);  for(int i = this.ChildIndex + 1; i < this.ChildIndex + newSize; i++)  {  \_parentControl.ColumnDefinitions[i].Width = new GridLength(0, GridUnitType.Star);  }  }  else  {  \_parentControl.ColumnDefinitions[this.ChildIndex].Width = new GridLength(newSize, GridUnitType.Star);  for (int i = this.ChildIndex + newSize; i < this.ChildIndex + \_size; i++)  {  \_parentControl.ColumnDefinitions[i].Width = new GridLength(1, GridUnitType.Star);  }  }  return true;  } |

Kolejną funkcją występującą w klasie LayoutControl jest ReadControlPropertyFromXaml(), która odpowiada za odczytywanie wartości właściwości kontrolki podczas procesu zmieniania wyglądu wybranego elementu. Zebrane dane są wykorzystywane w oknie PropertyWindow.

W module administracyjnym znajduje się również klasa Translator, odpowiadająca za przekształcenie struktury siatki kontrolek z jezyka XAML do listy obiektów z których każdy reprezentuje pojedynczą kontrolkę. Właściwościami występującymi w implementacji tego elementu są

* List<IWebControl> Controls zawierająca listę już przekształconych kontrolek
* LayoutControl CurrentControl przechowująca zawiera aktualnie tłumaczoną kontrolkę.

Operacje na przekształcania siatki kontrolek w listę obiektów odbywają się za pomocą funkcji

* ConvertToWebPage()będącej główna metoda tłumacząca schemat XAML na listę kontrolek
* GetSimpleControlFromXaml()odpowiedzialną za odczytywanie wartości właściwości kontrolki. Na podstawie wykrytego rodzaju kontrolki XAML tworzony jest odpowiedni obiekt elementu umieszczany na wynikową listę
* ApplyAlignment()odpowiedzialną za wykrycie klasy CSS nadających położenie w poziomie i pionie elementu HTML na podstawie właściwości kontrolki. Uzyskana wartość zapisywana jest do właściwości ClassName obiektu elementu
* GenerateCustomClass(selected)wykrywającą klasy CSS określających typ elementu HTML na podstawie właściwości kontrolki. Uzyskana wartość zapisywana jest do właściwości ClassName obiektu elementu.

Kolejną klasą tego modułu jest StyleBuilder odpowiadająca za wygenerowania stylów CSS dla kontrolek. Posiada ona tylko jedną właściwość string Styles przechowująca łańcuch wygenerowanych styli. Obsługa tworzenia pliku zawierającego style elementów podczas zapisu layout witryny odbywa się za pomocą funkcji

* GenerateCSS()tworzącej plik CSS zawierający style dla elementów w których określony został kolor: tła, czcionki lub elementu
* SaveFile zapisującej utworzony plik CSS w folderze witryny
* ClearFile()wykorzystywanej do czyszczenia pliku CSS zawierającego style dla kontrolek. Wywoływana jest przy każdym zapisie siatki kontrolek.

Ostatnim z głównych elementów modułu administracyjnego jest statyczna klasa ControlCounter zliczająca liczbę poszczególnych elementów na stronie. Wykorzystywana jest w klasie StyleBuilder aby wytworzyć osobne style dla każdej kontrolki. Właściwościami występującymi w implementacji tego elementu są

* int InputCount zawierająca liczbę wystąpień pól do wprowadzania na siatce kontrolek
* int LabelCount zawierająca liczbę wystąpień napisów na siatce kontrolek
* int ButtonCount zawierająca liczbę wystąpień przycisków na siatce kontrolek
* int LinkCount zawierająca liczbę wystąpień linków na siatce kontrolek
* int ImageCount zawierająca liczbę wystąpień obrazów na siatce kontrolek
* int EmptySpaceCount zawierająca liczbę wystąpień pustych komórek na siatce kontrolek.

## Moduł strony WWW

Moduł strony WWW zbudowany jest przy wykorzystaniu technologii .NET Core. Głównymi elementami występującymi w tym module są

* CostructSite(siteName, pageName) będącą główna funkcja tworząca witrynę internetową
* GetPageUrl() jest funkcją wykrywającą czy w adresie URL, zostały podane specjalne parametry
* CreateSimpleControl(control) jest to helper, czyli pomocnicza funkcja do generowania widoku. Służy do wygenerowania kontrolek layoutu. Poprzez rekurencyjne wykonywanie tej funkcji tworzone są elementy HTML na podstawie obiektu modelu typu WebControl przekazywanego w parametrze control tej funkcji. Kod źródłowy przedstawia rekurencyjne wywołanie tej funkcji

|  |
| --- |
| case WebControlTypeEnum.panel:  if (control.ChildrenControls != null)  {  for (int i = 0;  i< control.ChildrenControls.Count;  i++)  {  @CreateSimpleControl(control.ChildrenControls[i]);  }  }  break; |

* SetRowElementsHeight() jest to funkcja JavaScript uruchamiana podczas ładowania widoku strony. Jest to pomocnicza funkcja poprawiająca wygląd layoutu witryny. Zadaniem które wykonuje jest wyrównanie wysokości elementów HTML znajdujących się w elemencie o klasie row, jest to konieczne do działania funkcjonalności ustawiania pozycji elementów w poziomie.

## Moduł WEB API

Pierwszym elementem tworzącym ten moduł jest klasa dziedzicząco po klasie Controller nazywająca się MenuController. Odpowiada za obsługę operacji dokonywanych na tabeli Menu znajdującej się w bazie danych. Posiada ona właściwość DataContext Context zawierającej aktualny kontekst bazy danych. Obsługa tabeli Menu odbywa się za pomocą funkcji

* GetAll()zwracającej listę wszystkich elementów typu Menu
* GetById(id) zwracającej element typu Menu o podanym id
* GetByName(name) zwracającej listę elementów typu Menu posiadających podaną nazwę
* GetMenusForSite(id) zwracającej listę elementów typu Menu przypisanych do witryny o podanym id
* Create(item) dodającej nowy element typu Menu
* Update(menu) aktualizującej elementy typu Menu
* Delete(id) usuwającej elementów typu Menu o podanym id.

Kolejnym elementem tworzącym ten moduł jest klasa dziedzicząco po klasie Controller nazywająca się PageController. Odpowiada za obsługę operacji dokonywanych na tabeli Page znajdującej się w bazie danych. Posiada ona właściwość DataContext Context zawierającej aktualny kontekst bazy danych. Obsługa tabeli Page odbywa się za pomocą funkcji

* GetAll()zwracającej listę wszystkich elementów typu Page
* GetById(id) zwracającej element typu Page o podanym id
* GetByName(name) zwracającej listę elementów typu Page posiadających podaną nazwę
* GetPagesForSite(id) zwracającej listę elementów typu Page przypisanych do witryny o podanym id
* Create(item) dodającej nowy element typu Page
* Update(page) aktualizującej elementy typu Page
* Delete(id) usuwającej elementów typu Page o podanym id.

Ostatnim elementem tworzącym ten moduł jest klasa dziedzicząco po klasie Controller nazywająca się SiteController. Odpowiada za obsługę operacji dokonywanych na tabeli Site znajdującej się w bazie danych. Posiada ona właściwość DataContext Context zawierającej aktualny kontekst bazy danych. Obsługa tabeli Site odbywa się za pomocą funkcji

* GetAll()zwracającej listę wszystkich elementów typu Site
* GetById(id) zwracającej element typu Site o podanym id
* GetByName(name) zwracającej listę elementów typu Site posiadających podaną nazwę
* Create(item) dodającej nowy element typu Site
* Update(site) aktualizującej elementy typu Site
* Delete(id) usuwającej elementów typu Site o podanym id.

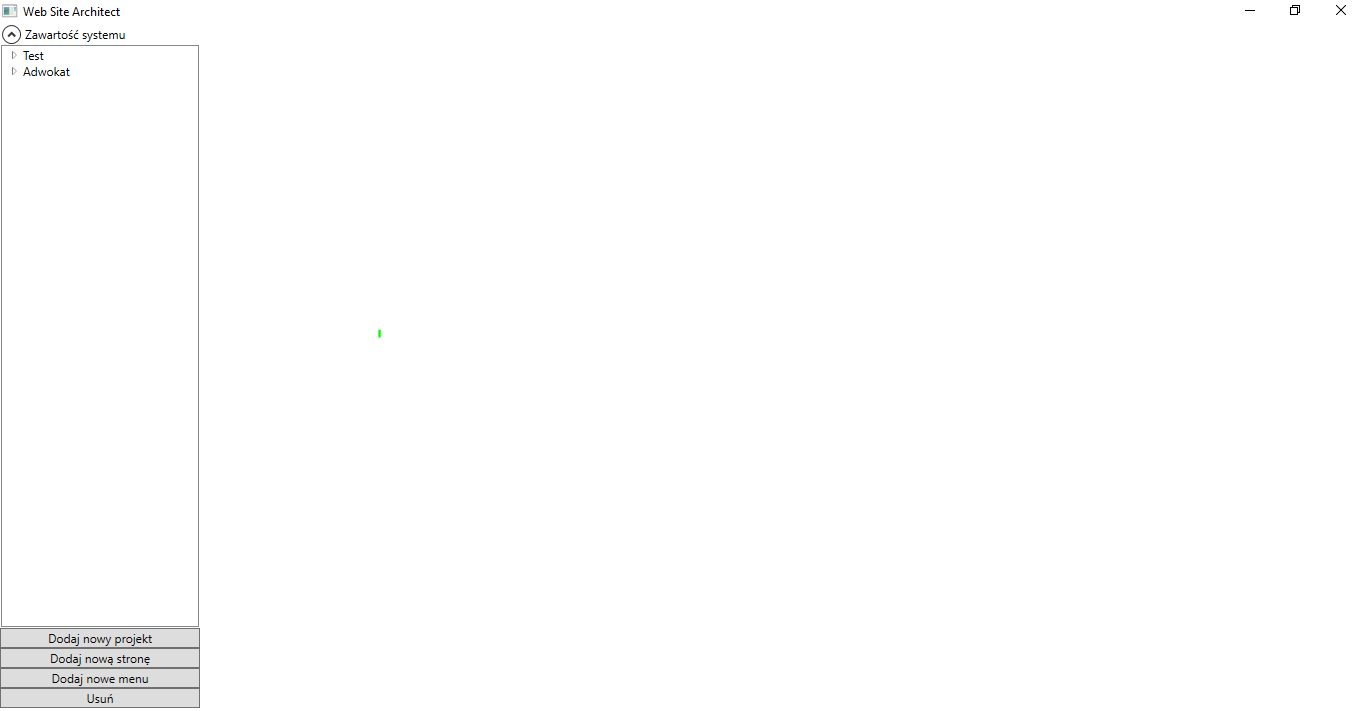
# Użytkowanie

Rozdział zawiera opis interfejsu użytkowania. W podrozdziale „Testy” zostały umieszczone informacje o przeprowadzonych testach aplikacji po zakończeniu etapu implementacji.

Aplikacja administracyjna składa się z czterech okien:

1. Okno główne – jest oknem startowym aplikacji. Rysunek 5.1.1 przedstawia wygląd okna. Znajdują się w nim dwa elementy:

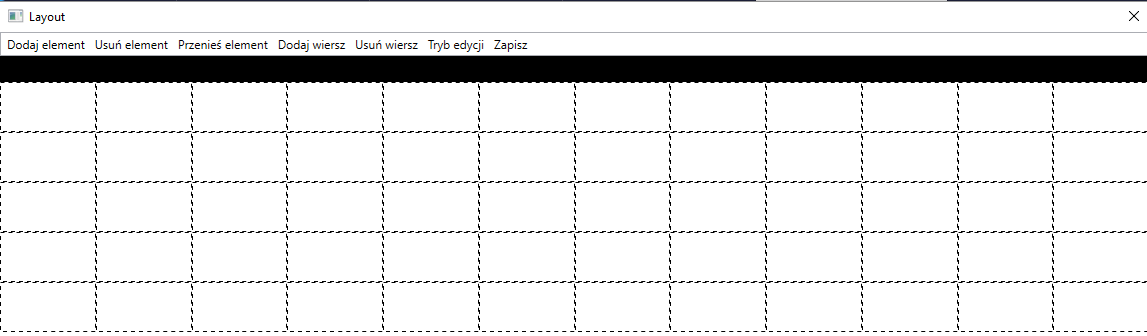
* Panelu zarządzania projektami w którym znajduje się drzewo projektów. Przy przechodzeniu przez elementy drzewa następuje otwarcie okna layoutu dla wybranego menu lub podstrony.
* Przycisków nawigacyjnych umożliwiających dodawanie nowych elementów do projektów oraz usuwaniem już istniejących.



Rysunek 5.1.1 Okno główne.

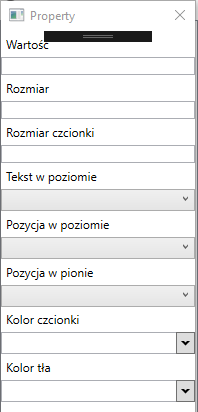
1. Okno layoutu - jest oknem projektowania menu oraz podstron witryny. Rysunek 5.1.2 przedstawia wygląd tego okna. Składa się z dwunastokolumnowej siatki na których umieszczane są kontrolki oraz pasku przycisków nawigacyjnych umożliwiających.

* Dodanie nowej kontrolki.
* Usunięcie istniejącej kontrolki.
* Przeniesienie elementu.
* Dodanie wiersza.
* Usunięcie wierszy.
* Włączenie trybu edycji kontrolki.
* Zapisanie schematu strony do bazy danych.



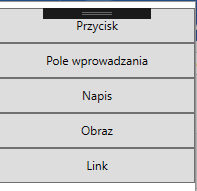
Rysunek 5.1.2 Okno layoutu.

1. Okno właściwości – jest oknem służącym do edytowania zawartości i wyglądu pojedynczej kontrolki.



Rysunek 5.1.3 Okno właściwości.

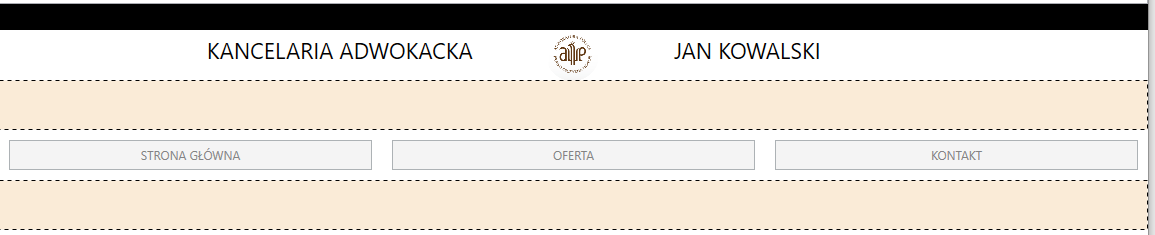
1. Okno kontrolek – jest oknem służącym do wybieranie kontrolki do dodania.



Rysunek 5.1.4 Okno kontrolek.

W ramach testów stworzono przykładowa stronę internetowa reklamująca kancelarie adwokacką składająca się ze strony głównej, podstrony opisującej ofertę oraz podstrony informującej o kontakcie:

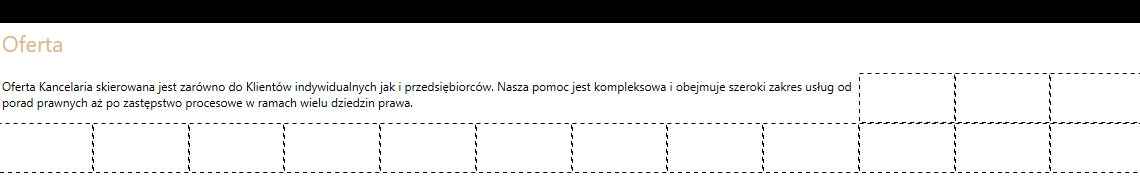
Projekt w aplikacji administracyjnej:



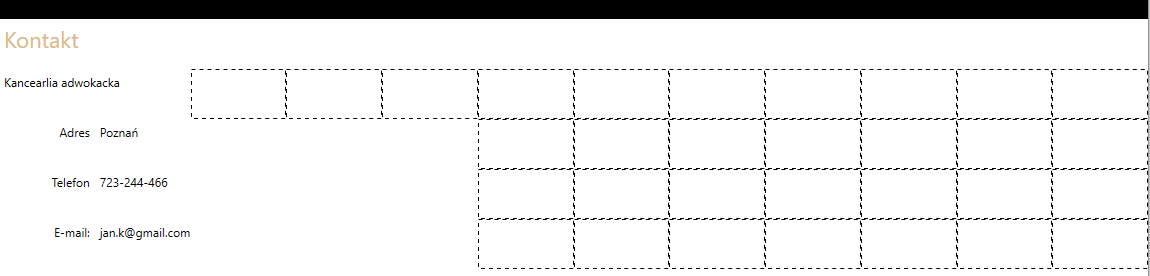
Rysunek 5.2.1.1 Layout menu.



Rysunek 5.2.1.2 Layout strony głównej.



Rysunek 5.2.1.3 Layout podstrony Oferta



Rysunek 5.2.1.4 Layout podstrony Kontakt

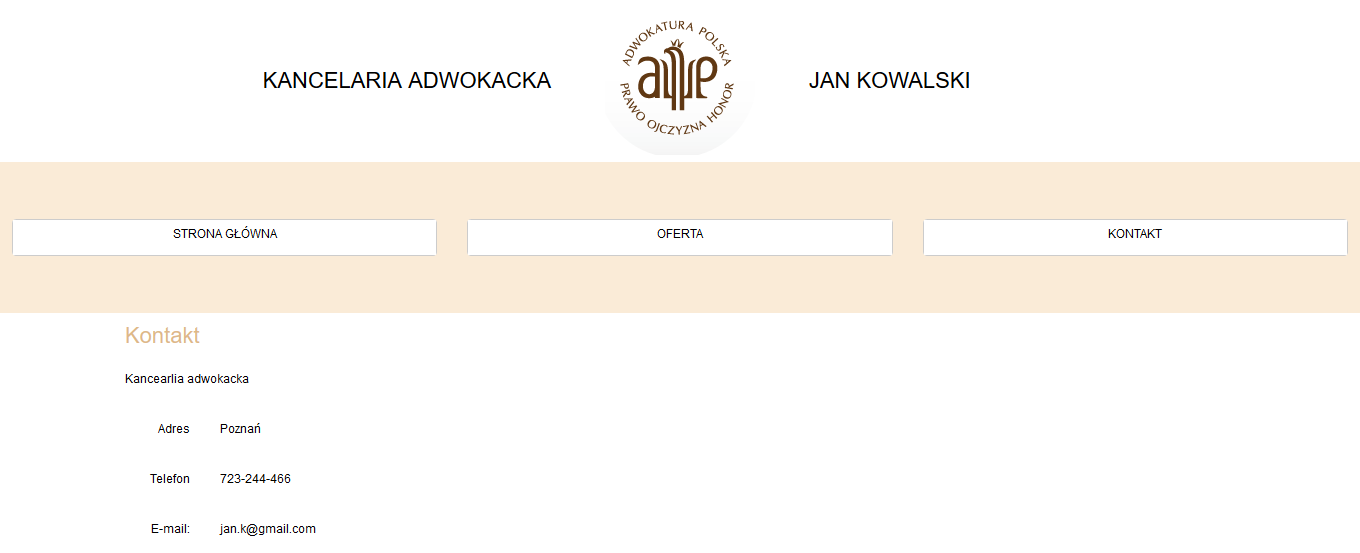
Efekt końcowy:



Rysunek 5.2.1.5 Strona główna.



Rysunek 5.2.1.6 Podstrona Oferta .



Rysunek 5.2.1.7Podstrona kontakt.

# Podsumowanie

W pracy zostały zrealizowane następujące cele:

* Stworzenie systemu zarzadzania treścią umożliwiającego tworzenie wiele stron internetowych w jednym miejscu
* stworzenie aplikacji nie wymagającej od użytkownika wiedzy technicznej
* stworzenie systemu opartego na najbardziej popularnych technologiach
* stworzenie systemu umożliwiającego szybkie stworzenie witryny internetowej
* zaprojektowanie i realizacja aplikacji umożliwiającej w łatwy sposób dodawanie nowych funkcjonalności oraz poszerzanie już istniejących.

Podczas tworzenia projektu napotkano na problemy podczas tworzenia aplikacji administracyjnej. Jedynym z problemów była serializacja layoutu kontrolek. Problem ten został rozwiązany za pomącą biblioteki NewtonSoft.Json, która umożliwiła zapisanie stanu siatki layoutu kontrolek do postaci formatu JSON oraz odczytanie stanu wcześniej wykonanej pracy podczas ponownego uruchamiania tworzonego projektu strony.

W przyszłości aplikacja może być rozbudowana o nowe funkcjonalności:

* dodanie nowych rodzajów kontrolek ( na przykład galerii zdjęć)
* dodanie możliwości projektowania różnego rozłożenia kontrolek dla mniejszych szerokości ekranu – umożliwiłoby to stworzenie bardziej przejrzystych layoutów dla urządzeń mobilnych
* możliwość dodawania animacji do strony www
* rozszerzenie aplikacji administracyjnej o autoryzację użytkownika
* dodanie do aplikacji gotowych szablonów stron
* zapis w bazie danych informacji o kontrolkach w postaci osobnych tabel dla każdego typu kontrolki.

# Bibliografia

[1] Freeman A., *Pro ASP.NET CORE MVC,* Apress, 2016.

[2] Lerman J., Miller R., *Programming Entity Framework: Code First*, O'Reilly Media, 2011.

[3] Matulewski J., *MVVM i XAML w Visual Studio 2015,* Helion, Gliwice, 2015.

[4] [Miles](http://lubimyczytac.pl/autor/56687/russ-miles) R., [Hamilton](http://lubimyczytac.pl/autor/88359/kim-hamilton) K., UML 2.0. Wprowadzenie, Helion.

[5] Shreves R***., Joomla! Biblia*, Helion*.***

[6] Documentation, [*https://getbootstrap.com*](https://getbootstrap.com)*,* [dostęp: 21.01.2018]

**[7] Wprowadzenie do wzorca projektowego MVVM, *https://msdn.microsoft.com****,* [dostęp: 21.01.2018]