Politechnika Poznańska

Wydział Elektryczny

Instytut Automatyki i Inżynierii Informatycznej

Michał Suchorzyński

System zarządzania treścią dla małych i średnich przedsiębiorstw

Praca inżynierska

napisana pod kierownictwem

dr inż. Adama Meissnera

Poznań, 2018

Poznan University of Technology

Faculty of Electrical Engineering

Institute of Control and Information Engineering

Michał Suchorzyński

A Content Management System for Small and Medium-sized Companies

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 project consists of four parts: an administrative application, a web application, a database, and a WEB API service. To run an administrative application, it is required to have a Microsoft Windows operating system, for the proper functioning of the website hosting is needed to support ASP .NET MVC technologies and the MS SQL database.

Streszczenie

W pracy przedstawiono system zarządzania treścią dla małych i średnich przedsiębiorstw. Zadaniem aplikacji jest ułatwienie użytkownikowi tworzenie oraz utrzymywanie witryny internetowej firmy. Projekt składa się z czterech części: aplikacji administracyjnej, aplikacji webowej, bazy danych, usługi WEB API. Do uruchomienia aplikacji administracyjnej wymagany jest posiadania systemu operacyjnego Microsoft Windows, do poprawnego działania witryny internetowej wymagany jest hosting wspierający technologie ASP .NET MVC oraz bazę danych MS SQL.

Spis treści

Wstęp

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 funkcjonalności oraz dodawania nowych był prosty w realizacji.

System zarządzania treścią

System zarządzania treścią - Content Management System (CMS) jest to system informatyczny, który umożliwia użytkownikowi bez znajomości wiedzy technicznej stworzenie, rozwijanie oraz utrzymywanie serwisu WWW. Występuje klika rodzajów systemów zarządzania treścią:

- Content Management Framework
- Page-based systems
- Module-based systems
- Content Object Systems

System zarządzania treścią składa się najczęściej z dwóch modułów:

- aplikacja do zarządzania treścią (CMA) jest to aplikacja administracyjna umożliwiająca użytkownikowi dodawanie , usuwanie i edytowanie zawartości tworzonej strony internetowej,
- aplikacja do dostarczania treści (CDA) jest to aplikacja wyświetlająca zawartość zaprojektowanego projektu strony internetowej wcześniej stworzonego w aplikacji administracyjnej.

Podobne systemy na rynku

Najczęstszymi rozwiązaniami dostępnymi na rynku są aplikacje stworzone w oparciu na języku PHP. Najpopularniejszymi systemami są;

- WordPress – jest to darmowy open sourcowy system zarządzania treścią oparty na technologiach PHP oraz MySQL. W początkowych etapach funkcjonowania był używany jako platforma blogowa. Zawiera wiele rozszerzeń zawierających nowe elementy strony WWW, lub zmieniające aspekty stylistyczne powstałej

witryny. Najczęściej jest stosowana do tworzenia zwykłych stron lub stron firmowych.

- Joomla! - jest to darmowy open source system kontroli wersji oparty na technologiach PHP oraz MySQL [5]. Oparty jest na wzorcu projektowym MVC model-widok-kontroler. Jedną z głównych zalet jest to że jest dostępna w wielu wersjach językowych.

1. Cel i zakres pracy

W rozdział zostanie omówione przeznaczenie oraz zadania projektowanego systemu zarządzania treścią. Przedstawione zostaną zrealizowane główne funkcjonalności oraz dodatkowo zostanie opisana ogólna struktura pracy.

1.1 Przeznaczenie i zadania projektowanego systemu

Przeznaczeniem systemu jest tworzenie prostych witryn WWW służących jako wizytówki firm. Umożliwia on użytkownikowi w prosty stron zaprojektowanie menu oraz podstron witryny internetowej. Każdy element strony może być projektowany od zera, poprzez nadawanie mu własnych kolorów rozmiarów i tym podobnych. Możliwe jest również projektowanie wielu witryn w jednym miejscu. Użytkownik w jednej aplikacji może tworzyć 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.

1.2 Zrealizowane funkcjonalności:

W ramach systemu istnieją funkcjonalności:

- 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.

1.3 Struktura pracy

Praca została podzielona na rozdziały:

- Wstęp – zawiera opis dziedziny podjętego tematu,
- Cel i zakres pracy – zawiera opis celów postawionych przed rozpoczęciem tworzenia systemu,

- Metody modelowania i realizacja – zawiera opis technologii wykorzystanych do realizacji projektu,
- Model systemu – zawiera opis architektury systemu oraz poszczególnych jego modułów,
- Realizacja – zawiera opis implementacji systemu,
- Użytkowanie – zawiera opis interfejsu użytkownika oraz przeprowadzone testy systemu.
- Podsumowanie – zawiera podsumowanie pracy,
- Bibliografia – zawiera spis wykorzystanej literatury w pracy.

2. Metody modelowania i realizacji

W rozdziale zostaną przedstawione wykorzystane technologie do realizacji systemu. Wzorce projektowe oraz technologie programistyczne wykorzystane do zaprojektowania systemu jak również środowiska programistyczne wykorzystywane do realizacji projektu.

2.1 Wzorce projektowe

2.1.1 MVC

Jest to wzorzec projektowy *Model–View–Controler*, w którym struktura aplikacji jest dzielona na trzy główne warstwy:

- Model – w tej warstwie przechowywane są elementy odpowiedzialne za implementacje logiki dla aplikacja. Często elementy modelu wykorzystywane są do odczytu i zapisu stanu aplikacji w bazie danych.
- Widok - w tej warstwie przechowywane są elementy odpowiedzialne za wyświetlanie interfejsu użytkownika. Najczęściej ten interfejs jest tworzony na podstawie stanu danych modelu.
- Kontroler - w tej warstwie przechowywane są elementy odpowiedzialne za interakcje aplikacji z użytkownikiem. Obsługują model oraz decydują który widok i z jaką zawartością zostanie wyświetlony użytkownikowi.

2.1.2 MVVM

Jest to wzorzec projektowy *Model–View–ViewModel*, [7] w którym struktura aplikacji jest dzielona na trzy główne warstwy:

- Model – w tej warstwie przechowywane są elementy odpowiedzialne za implementacje logiki dla aplikacja. Często elementy modelu wykorzystywane są do odczytu i zapisu stanu aplikacji w bazie danych.
- Widok - w tej warstwie przechowywane są elementy odpowiedzialne za wyświetlanie interfejsu użytkownika. Najczęściej wyświetlane są w nimi aktualne stany obiektów z modelu.
- Model widoku - jest abstrakcją widoku aplikacji, w tej warstwie przechowywane są elementy odpowiedzialne za wiązanie danych modelu z wyświetlanymi wartościami użytkownikowi.

2.2 Technologie programistyczne

2.2.1 C#

Jest to wieloparydygmatowy język programowania 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

2.2.2 WPF

Windows Presentation Foundation jest to model programistyczny umożliwiający programiście tworzenie nowoczesnych aplikacji desktopowych na systemy operacyjne Windows. Został wprowadzony przez firmę Microsoft w roku 2006 jako przyszły następca dotychczasowego modelu WinForms. Umożliwia on pisanie aplikacji z wykorzystaniem wzorca projektowego MVVM.

2.2.3 ASP.NET Core

Jest to freamowork open source firmy Microsoft. Został wprowadzony w 2016 roku jako nowa generacja freamowork ASP .NET. Umożliwia tworzenie aplikacji Web z wykorzystaniem wzorca projektowego MVC [1].

2.2.4 Bootstrap

Jest to zestaw narzędzi HTML, CSS i JS ułatwiającą tworzenie interfejsów użytkownika serwisów WWW. Umożliwia tworzenie responsywnych strony internetowe oraz wykorzystywanie wcześniej zaprojektowanych elementów widoku.

2.2.1 JavaScript

Jest to skryptowy język programowania wspomagającym interakcje między użytkownikiem a stroną internetową. Jego wykorzystanie daję możliwość urozmaicenia wizualnego witryny poprzez dodanie animacji oraz daje możliwość dynamicznej zmiany zawartości strony bez konieczności przeładowywania strony. W projekcie został wykorzystany jeden z wielu freamowork: jQuery.

2.2.2 Entity Framework

Jest to Object/Relational Mapping (O/RM) freamowork, który ułatwia dostęp i obsługę bazy danych. Pozwala zmapować tabele relacyjnej bazy danych do postaci

obiektów klas aplikacji, dzięki czemu możliwe jest łatwe dokonywanie uaktualniania zawartości bazy danych.

2.2.3 XAML

Extensible Application Markup Language jest językiem bazującym na składni XML stworzonym przez firmę Microsoft [3]. W technologii WPF jest wykorzystywany do projektowania widoku. Odpowiada za wizualną prezentacje aplikacji.

2.2.4 UML

Unified Modeling Language jest językiem pół-formalnym wykorzystywanym do modelowana schematów systemów informatycznych. Poprzez jego wykorzystanie można zobrazować architekturę oraz zasadę działania systemu [4].

2.2.5 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.

2.2.6 System kontroli wersji (*SVN*)

Zastosowanie systemu kontroli wersji 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.

2.3 Środowiska programistyczne

2.3.1 Microsoft Visual Studio

Jest to zintegrowane środowisko programistyczne (IDE) 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.

2.3.2 SQL Server Management Studio

Jest to narzędzie firmy Microsoft umożliwiające z wykorzystaniem graficznego interfejsu użytkownika zarządzanie bazami danych. Pozwala zalogować się do serwera SQL aby tworzyć lub edytować bazy danych.

2.3.3 Visual Paradigm

Jest narzędziem ułatwiającym modelowanie diagramów UML. Poza obsługą modelowania zapewnia generowanie raportów oraz funkcje generowanie kodu.

3. Model systemu

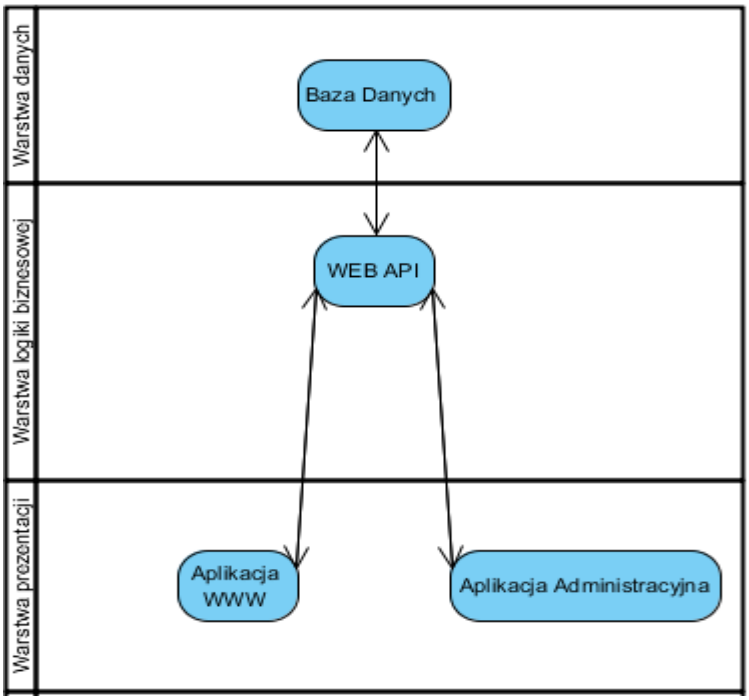
W tym rozdziale zostanie przedstawiona architektura pozioma i pionowa aplikacji. Zostanie opisana ogólna architektura całego systemu, model bazy danych oraz zasada funkcjonowania poszczególnych modułów projektu.

3.1 Architektura systemu

Do stworzenia 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 (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 schemat działania systemu po zastosowaniu tej architektury.



Rysunek 3.1.1 Schemat architektury pionowej

W warstwie danych znajduję się baza danych SQL przechowywująca informację o projektach stron internetowych stworzonych przez użytkownika w aplikacji

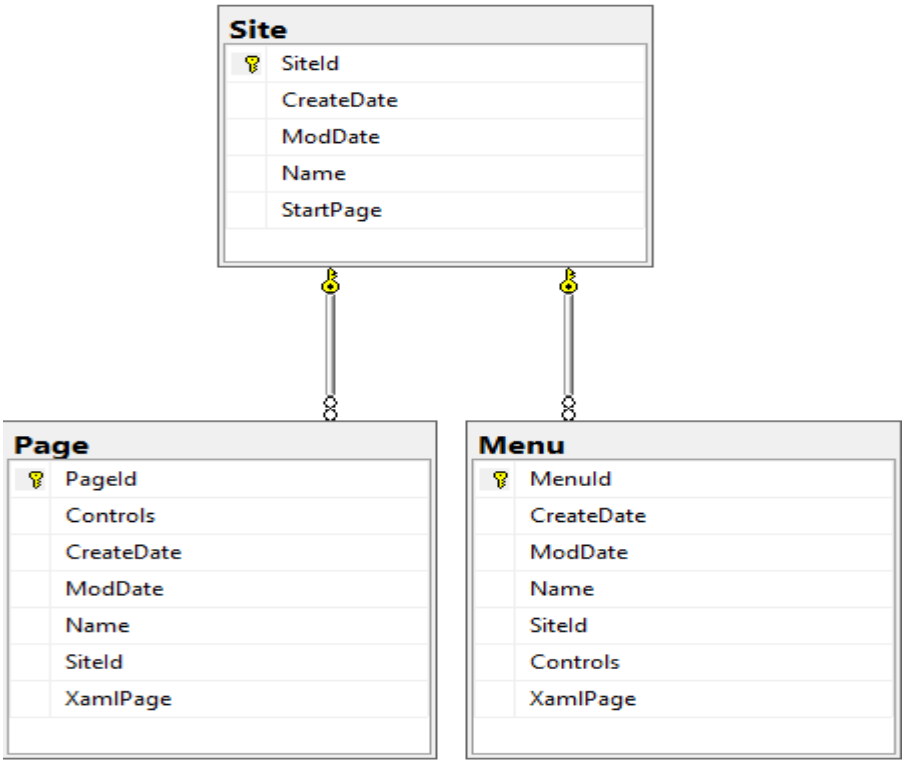
administracyjnej. Obsługa operacji dokonywanych na tych danych odbywa się w warstwie logiki biznesowej, która w projekcie została zrealizowana jako WEB API posiadający funkcje dokonujące zapisów, odczytów i modyfikacji tabel ich rodzaj oraz dokładne działanie jest określane na podstawie requestów otrzymywanych z warstwy prezentacji, po wykonaniu zleconej operacji w responsie wysyłana jest informacja zwrotna zawierającego informacje o skutku wykonanej operacji. W warstwie prezentacji znajdują się aplikacja administracyjna oraz strona WWW. W trakcie działania obie aplikacje wysyłają requesty do warstwy logiki biznesowej. Aplikacja administracyjna ma możliwość wysyłania zleceń odczytywania, modyfikowania i dodawania nowych rekordów w bazie danych, a strona WWW tylko odczytuje wcześniej zapisane informacje o utworzonym projekcie strony internetowej.

3.2 Model bazy danych

Baza danych składa się z trzech tabel:

- Site – zawierającej witryny stworzone w aplikacji administracyjnej,
- Page – zawierającej podstrony witryn,
- Menu – zawierające panele Menu witryn.

Na rysunku 3.2.1 został przedstawiony schemat bazy danych.



Rysunek 3.2.1 Schemat bazy danych

Baza danych została stworzona za pomocą technologii Entity Framework, umożliwia ona w podejściu Code-First [2], stworzenie najpierw klasy modelu z których następnie generowane są tabele bazy danych wraz z ich relacjami na Rysunku 3.2.2 został przedstawiona klasa `Site` reprezentująca pojedynczą witrynę projektowaną w aplikacji administracyjnej.

```
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; }
}
```

Rysunek 3.2.2 Model dla tabeli Site

Klasa `Site` posiada właściwości:

- `SiteId` – reprezentują ona kolumnę przechowywująca klucz główny tabeli, został on określony za pomocą atrybutu `[Key]`,
- `Name` – reprezentują ona kolumnę przechowywująca nazwę witryny,
- `CreateDate` – reprezentują ona kolumnę zawierająca datę stworzenia witryny
- `ModDate` – reprezentują ona datę ostatniej modyfikacji witryny,
- `StartPage` – reprezentują ona kolumnę przechowywującą identyfikator podstrony która ma być wyświetlana jako strona startowa,
- `Menus` – reprezentują ona wiersze tabeli `Menu` zawierające klucze obce do tego wiersza kolumny `Site`,
- `Pages` – reprezentują ona wiersze tabeli `Page` zawierające klucze obce do tego wiersza kolumny `Site`.

W analogiczny sposób zostały zamodelowane tabele `Page` oraz `Menu`

3.3 Moduły systemu

3.3.1 Aplikacja administracyjna

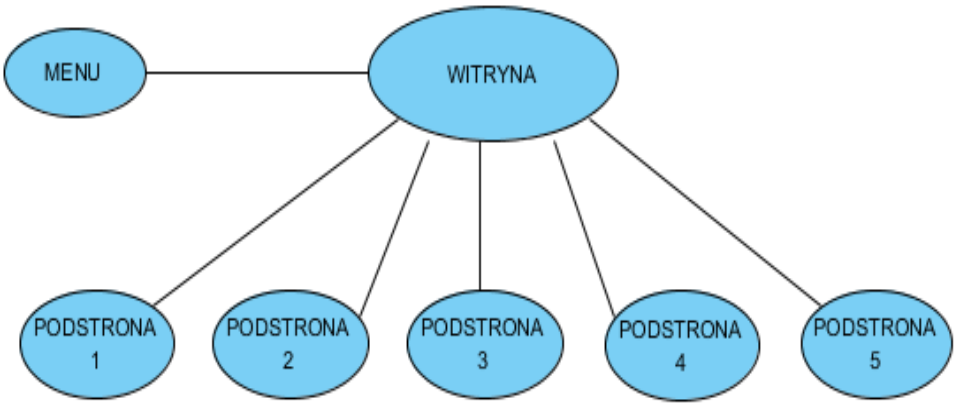
Jest to moduł zrealizowany przy wykorzystaniu wzorca projektowego MVVM, jest to aplikacja w technologii WPF.

Na rysunku 3.3.1.1 zostały przedstawione klasy wchodzące w skład aplikacji administracyjnej.

Rysunek 3.3.1.1 Klasy aplikacji administracyjnej.

Głównymi funkcjonalnościami aplikacji administracyjnej są:

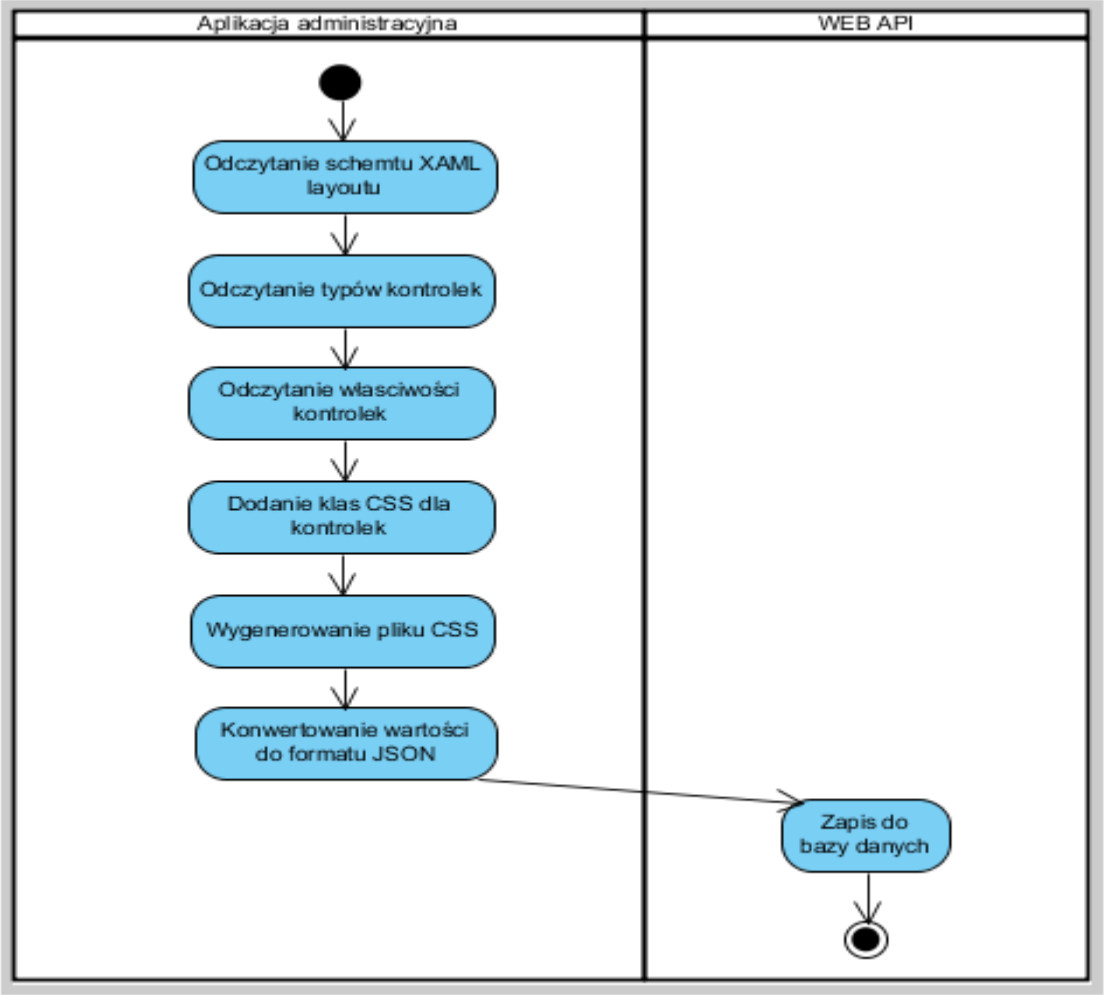
- tworzenia instancji projektu witryny internetowej, na rysunku 3.3.1.2 przedstawiono model budowy pojedynczej witryny internetowej.



Rysunek 3.3.1.1 Model budowy witryny.

Pojedyncza witryna składa się z jednego menu oraz podstron których liczba może być zmieniana dynamicznie.

- 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.



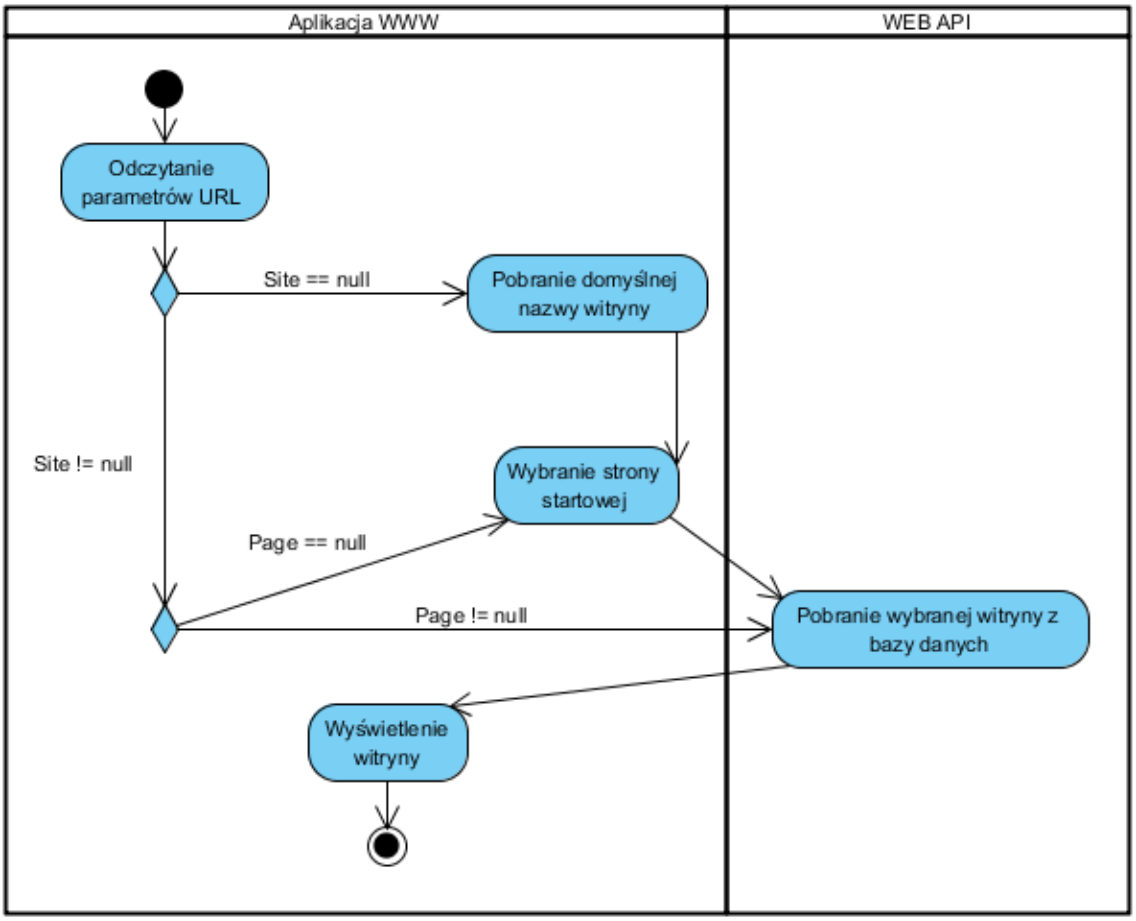
Rysunek 3.3.1.1 Algorytm zapisu layoutu.

- wypełnianie zawartością pojedynczych kontrolek
- 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.

3.3.2 Aplikacja WWW

Jest to moduł zawierający aplikacje ASP.Net MVC tworząca witrynę internetową Składa się z :

- HomeController – jest on 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.

- Widoku Index - odpowiada za generowanie layoutu kontrolek aktualnie ładowanej podstrony,
- Widok Layout - odpowiada za generowanie layoutu menu aktualnie ładowanej witryny.

3.3.3 WEB API

Jest to moduł odpowiedzialny za obsługę operacji dokonywanych na bazie danych. Został zrealizowany w technologii WEB API MVC. Składa się z trzech kontrolerów:

- SiteController – odpowiada za obsługę tabeli Site z bazy danych,
- PageContrller - odpowiada za obsługę tabeli Page z bazy danych,
- MenuController - odpowiada 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.

4. Realizacja

W tym rozdziale zostanie opisany elementy implementacji poszczególnych modułów projektu.

4.1 Aplikacja administracyjna

4.1.1 Klasa `LayoutControler`

Jest klasą dziedzicząco po interfejsie `INotifyPropertyChanged`. Odpowiada za obsługę siatki kontrolek w czasie modelowaniu zawartości witryny.

Zawiera właściwości:

- `StackPanel XamlPage` – przechowuje aktualny schemat XAML.
- `LayoutControl SelectedControl` – przechowuje aktualnie wybraną kontrolkę.

Zawiera funkcje:

- `AddRowToPage(selected)` – metoda dodająca dwunastokolumnowy wiersz do zawartości całej strony. Parametrem 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)` – metoda usuwający wiersz wraz z wszystkimi kontrolkami które się w nim znajdują. Parametrem jest wybrana podczas kliknięcia kontrolka,
- `AddControl(toAdd,toDelete)` – metoda dodająca nową kontrolkę do strony. Parametr `toAdd` reprezentuje kontrolkę z domyślnymi ustawieniami wybraną w `ControlWindow`. Parametr `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)` – metoda usuwającą kontrolkę ze strony. Wybrany element przekazywany w parametrze zostaje usuwany z siatki kontrolek, a w jego miejsce zostają dodane puste przestrzenie których ilość zależy od ilości zajmowanych kolumn przez wcześniejszą kontrolkę.

4.1.2 Klasa `LayoutControl`

Jest klasą dziedzicząco po klasie `INotifyPropertyChanged`. Odpowiada za obsługę zmian ustawień wybranej kontrolki.

Zawiera właściwości:

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

Zawiera funkcje:

`SetControlSize(newSize)` – jest funkcją ustawiającą 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;
        }
```

- `ReadControlPropertyFromXaml()` – jest to funkcja odpowiedzialna za odczytywanie wartości właściwości kontrolki podczas zmieniani wybranego elementu. Zebrane dane są wykorzystywane w oknie `PropertyWindow`.

4.1.3 Klasa `Translator`

Odpowiada za przekształcenie struktury siatki kontrolek z jezyka Xaml do listy klas reprezentujących pojedynczą kontrolkę.

Zawiera właściwości:

- `List<IWebControl> Controls` – zawiera listę już przekształconych kontrolek.
- `LayoutControl CurrentControl` – zawiera aktualnie tłumaczoną kontrolkę.

Zawiera funkcje:

- `ConvertToWebPage()` – jest to główna metoda tłumacząca schemat XAML na listę kontrolek,
- `GetSimpleControlFromXaml()` – jest to funkcja odpowiedzialna za odczytywanie wartości właściwości kontrolki. Na podstawie wykrytego rodzaju kontrolki XAML tworzony jest odpowiedni obiekt elementu.
- `ApplyAlignment()` – funkcja odpowiada 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)` – funkcja odpowiada za wykrycie klasy CSS określających typ elementu HTML na podstawie właściwości kontrolki. Uzyskana wartość zapisywana jest do właściwości `ClassName` obiektu elementu.

4.1.4 Klasa `StyleBuilder`

Odpowiada za obsługę operacji dokonywanych na tabeli `Menu` znajdującej się w bazie danych.

Zawiera właściwości:

- `string Styles` – zawiera łańcuch wygenerowanych styli.

Zawiera funkcje:

- `GenerateCSS()` – funkcja tworząca plik CSS zawierający style dla elementów w których określony został kolor: tła, czcionki lub elementu,

- `SaveFile ()` – funkcja zapisująca utworzony plik CSS w folderze witryny ,
- `ClearFile()` – funkcja wykorzystywana do czyszczenia pliku CSS zawierającego style dla kontrolek. Wywoływana jest przy zapisie siatki kontrolek.

4.1.5 Klasa `ControlCounter`

Jest klasą statyczną odpowiedzialną za zliczanie elementów na stronie. Wykorzystywana jest w klasie `StyleBuilder` aby wytworzyć osobne style dla każdej kontrolki.

Zawiera właściwość:

- `int InputCount` – zawiera liczbę wystąpień pól do wprowadzania na siatce kontrolek.
- `int LabelCount` – zawiera liczbę wystąpień napisów na siatce kontrolek.
- `int ButtonCount` – zawiera liczbę wystąpień przycisków na siatce kontrolek.
- `int LinkCount` – zawiera liczbę wystąpień linków na siatce kontrolek.
- `int ImageCount` – zawiera liczbę wystąpień obrazów na siatce kontrolek.
- `int EmptySpaceCount` – zawiera liczbę wystąpień pustych komórek na siatce kontrolek.

4.2 Aplikacja WWW

Najważniejszymi funkcjami są:

- `CostructSite(siteName, pageName)` – jest to główna funkcja tworząca witrynę internetową.
- `GetPageUrl()` – jest to funkcja wykrywająca czy w adresie URL, zostały podane specjalne parametry.
- `CreateSimpleControl(control)` – jest to `helper`, czyli pomocnicza funkcja do generowania widoku. Służy do generowania kontrolek layoutu. Poprzez rekurencyjne wykonywanie tej funkcji generowane 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.

4.3 WEB API

4.3.1 Klasa `MenuControler`

Jest klasa dziedzicząco po klasie `Controller`. Odpowiada za obsługę operacji dokonywanych na tabeli `Menu` znajdującej się w bazie danych.

Zawiera właściwość:

- `DataContext Context` – zawiera aktualny kontekst bazy danych.

Zawiera funkcje:

- `GetAll()` – zwraca listę wszystkich elementów typu `Menu`,
- `GetById(id)` – zwraca element typu `Menu` o podanym id,
- `GetByName(name)` – zwraca listę elementów typu `Menu` posiadających podaną nazwę,
- `GetMenusForSite(id)` – zwraca listę elementów typu `Menu` przypisanych do witryny o podanym id,
- `Create(item)` –dodaje nowy element typu `Menu`,
- `Update(menu)` – aktualizacja elementu typu `Menu`,
- `Delete(id)` –usuwa elementów typu `Menu` o podanym id.

4.3.2 Klasa `PageControler`

Jest klasa dziedzicząco po klasie `Controller`. Odpowiada za obsługę operacji dokonywanych na tabeli `Page` znajdującej się w bazie danych.

Zawiera właściwość:

- `DataContext Context` – zawiera aktualny kontekst bazy danych.

Zawiera funkcje:

- `GetAll()` – zwraca listę wszystkich elementów typu `Page`,
- `GetById(id)` – zwraca element typu `Page` o podanym id,
- `GetByName(name)` – zwraca listę elementów typu `Page` posiadających podaną nazwę,
- `GetPagesForSite(id)` – zwraca listę elementów typu `Page` przypisanych do witryny o podanym id,
- `Create(item)` –dodaje nowy element typu `Page`,
- `Update(menu)` – aktualizacja elementu typu `Page`,
- `Delete(id)` –usuwa elementów typu `Page` o podanym id.

4.3.3 Klasa `SiteControler`

Jest klasa dziedzicząco po klasie `Controller`. Odpowiada za obsługę operacji dokonywanych na tabeli `Site` znajdującej się w bazie danych.

Zawiera właściwość:

- `DataContext Context` – zawiera aktualny kontekst bazy danych.

Zawiera funkcje:

- `GetAll()` – zwraca listę wszystkich elementów typu `Site`,
- `GetById(id)` – zwraca element typu `Site` o podanym id,
- `GetByName(name)` – zwraca listę elementów typu `Site` posiadających podaną nazwę,
- `Create(item)` –dodaje nowy element typu `Site`,
- `Update(menu)` – aktualizacja elementu typu `Site`,
- `Delete(id)` –usuwa elementów typu `Site` o podanym id.

5. Użytkowanie

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

5.1 Interfejs użytkownika

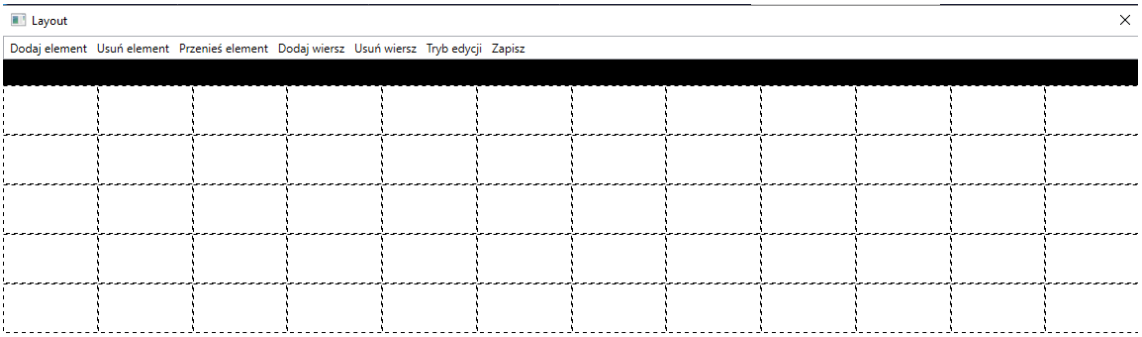
Aplikacja administracyjna składa się z czterech okien:

- 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.

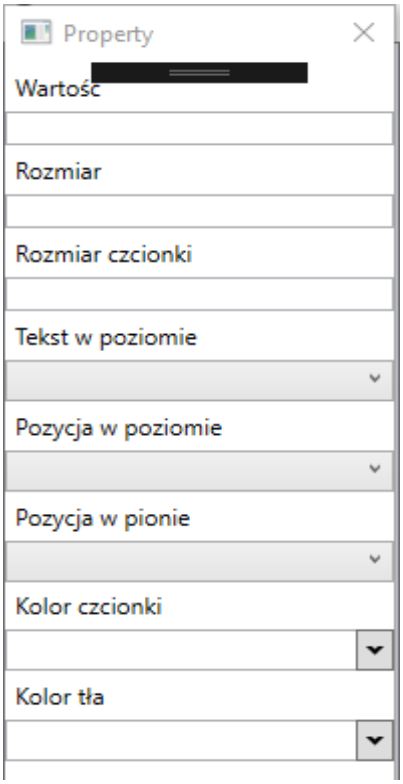
- Okno layoutu - jest oknem projektowania menu oraz podstron witryny. Rysunek 5.1.2 przedstawia wygląd tego okna. Składa się z dwunastokolumnowej siatki na których umieszczane są kontrolki oraz pasku przycisków nawigacyjnych umożliwiających:
    - Dodanie nowej kontrolki.
    - Usunięcie istniejącej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

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



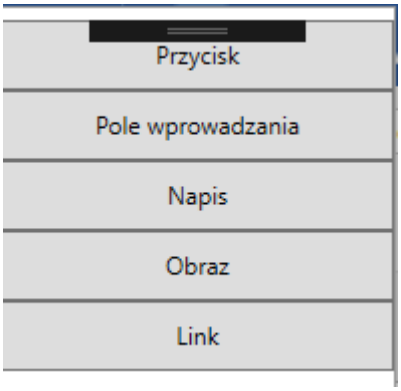
Rysunek 5.1.2 Okno layoutu.

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



Rysunek 5.1.3 Okno właściwości.

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



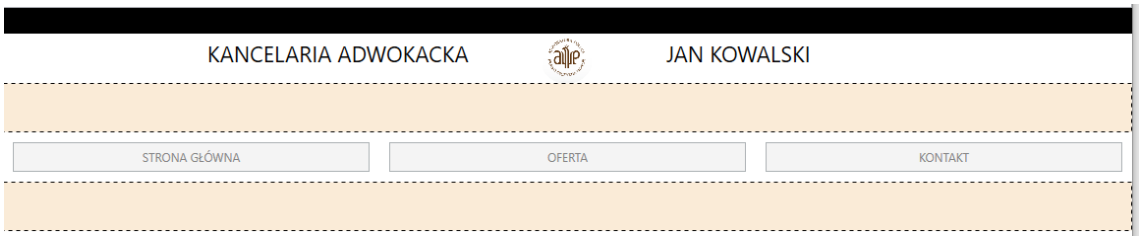
Rysunek 5.1.4 Okno kontrolek.

5.2 Testy

5.2.1 Przykładowa strona

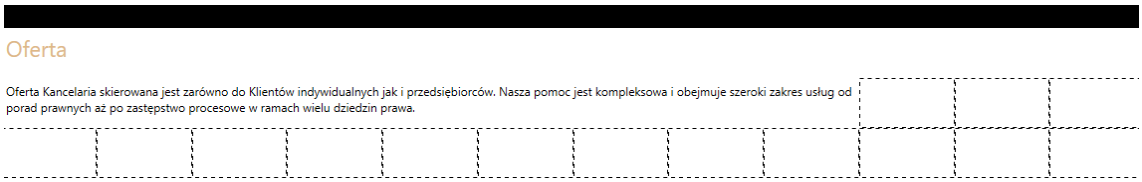
W ramach testów stworzono przykładowa stronę internetowa reklamująca kancelarie adwokacką składająca się ze strony głównej, podstrony opisującej ofertę oraz podstrony informującej o kontakcie:

Projekt w aplikacji administracyjnej:

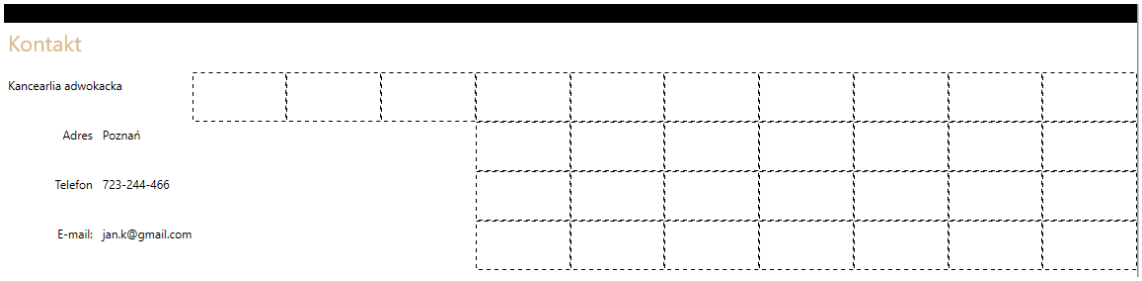


Rysunek 5.2.1.1 Layout menu.

Rysunek 5.2.1.2 Layout strony głównej.



Rysunek 5.2.1.3 Layout podstrony Oferta

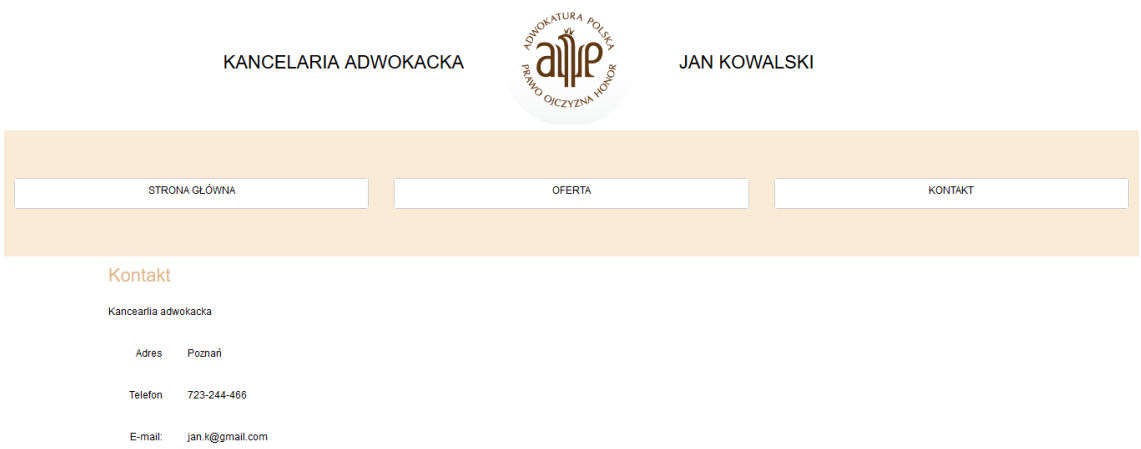


Rysunek 5.2.1.4 Layout podstrony Kontakt

Efekt końcowy:

Rysunek 5.2.1.5 Strona główna.

Rysunek 5.2.1.6 Podstrona Oferta .



Rysunek 5.2.1.7Podstrona kontakt.

6. Podsumowanie

6.1 Zrealizowane cele

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.

Podsumowując, realizacja projektu pracy inżynierskiej zakończyła się sukcesem

6.2 Napotkane problemy

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.

6.3 Kierunki rozwoju

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.

7. 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 R., Hamilton K., UML 2.0. Wprowadzenie, Helion.

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

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

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