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. Adam Meissner

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: administracyjnego, strony www, bazy danych oraz 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-sized companies. 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 and 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

System zarządzania treścią *CMS* (***c**ontent **m**anagement **s**ystem)* jest to system informatyczny, który umożliwia użytkownikowi bez znajomości wiedzy technicznej utworzenie, rozwijanie oraz utrzymywanie serwisu www. Przeznaczony jest także dla wykfalifikowanych programistów 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 utworzonego w aplikacji do zarządzania treścią.

Jednym z najpopularniejszych istniejących systemów zarządzania treścią jest darmowa aplikacja *open source 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!*. Również jest to darmowy *open source* system zarządzania treścią 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 utworzenia 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.

2. 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 administracyjnego oraz przeprowadzone testy systemu. Ostatnie dwa rozdziały poświęcone są podsumowaniu pracy oraz literaturze wykorzystywanej do realizacji pracy.

3. Metody modelowania i realizacji

Do utworzenia projektu wykorzystano dwa wzorce architektoniczne. Pierwszym z nich jest wzorzec MVC (***m**odel **v**iew **c**ontroler)*, 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 MVVM (***m**odel **v**iew **v**iew**m**odel)* [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 utworzony 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 *WPF* (**w**indows ***p**resentation **f**oundation)*, 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* utworzonym 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 *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 *O/RM* (***o**bject/**r**elational **m**apping) freamowork*,który nosi nazwę e*ntity 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*. ***j**avas**c**ript **o**bject **n**otation* 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* (***u****nified* ***m****odeling* ***l****anguage)*, 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*. Poza obsługą modelowania zapewnia ono funkcje generowanie raportów oraz kodu. W pracy znajdują się diagramy aktywności, komponentów oraz diagram klas przedstawiający budowę modułu administracyjnego.

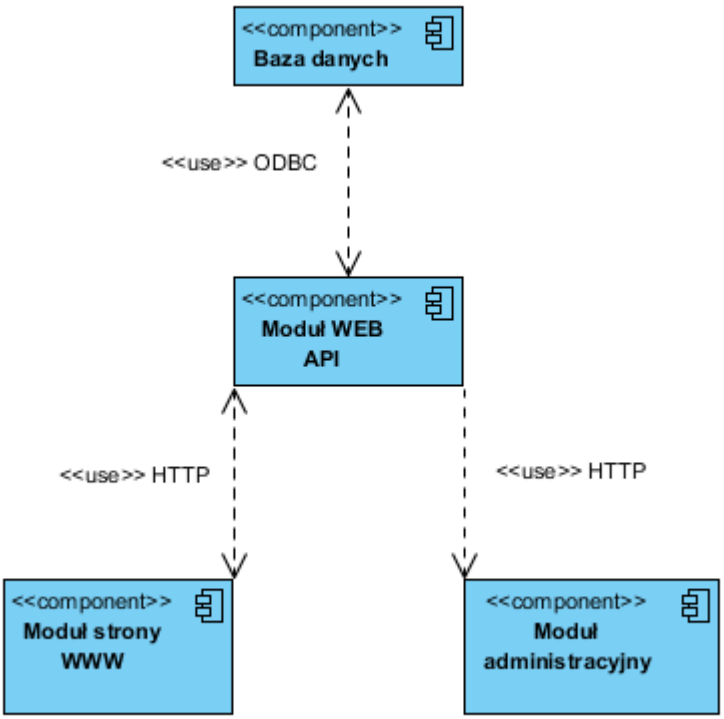
4. Model systemu

4.1 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 4.1 została przedstawiony diagram komponentów systemu oraz w jaki sposób odbywa się komunikacja między modułami.



Rys.4.1. Diagram komponentów systemu

W warstwie danych znajduję się baza danych *SQL* przechowywująca informację o projektach stron internetowych utworzonych przez użytkownika w module

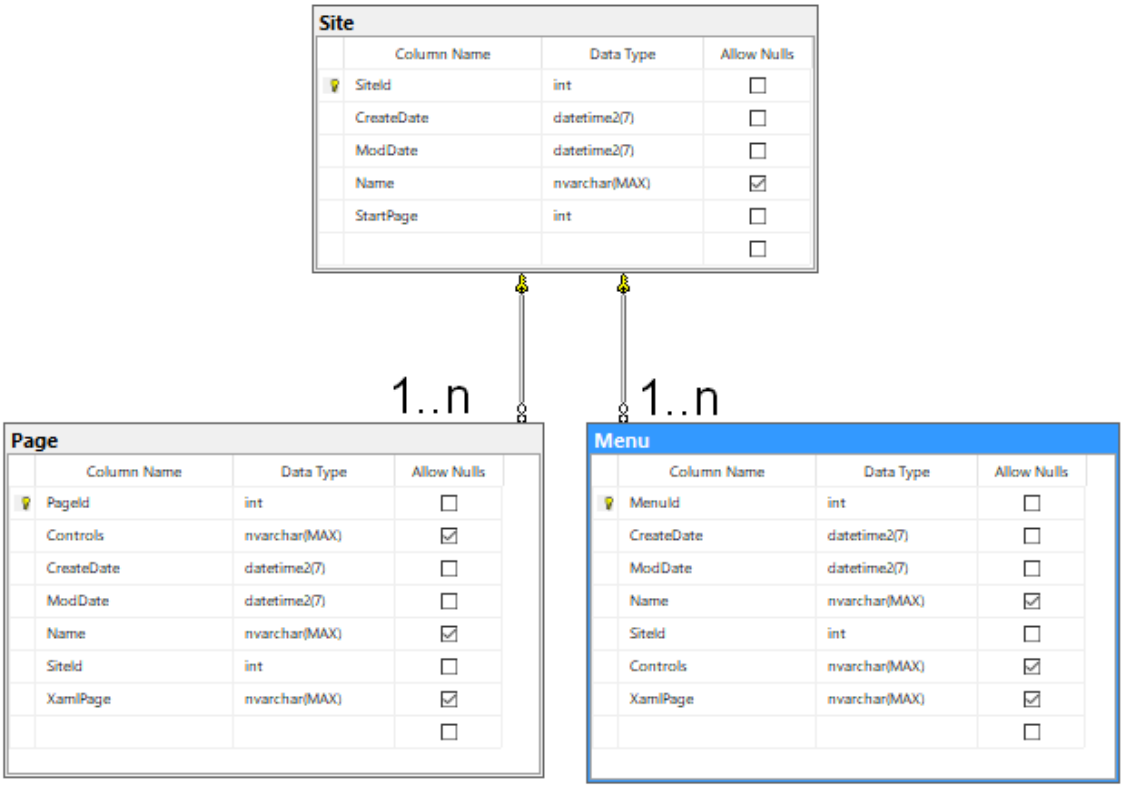
administracyjnym. Obsługa operacji dokonywanych na tych danych odbywa się w warstwie logiki biznesowej, która w projekcie została zrealizowana jako moduł *WEB API* posiadający funkcje dokonujące zapisów, odczytów i modyfikacji tabel. Rodzaj wykonywanych operacji oraz dokładne działanie jest określane 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.

4.2 Model bazy danych

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

- `Site` zawierającej witryny utworzone 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 został przedstawiony schemat bazy danych przedstawiający tabele wraz z ich kolumnami. Linie łączące elementy reprezentują powiązania między tabelami.



Rys.3.2. Schemat bazy danych

Baza danych została utworzona 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 utworzenia 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 *code-first* zostały zamodelowane pozostałe tabele bazy danych czyli `Page` oraz `Menu`.

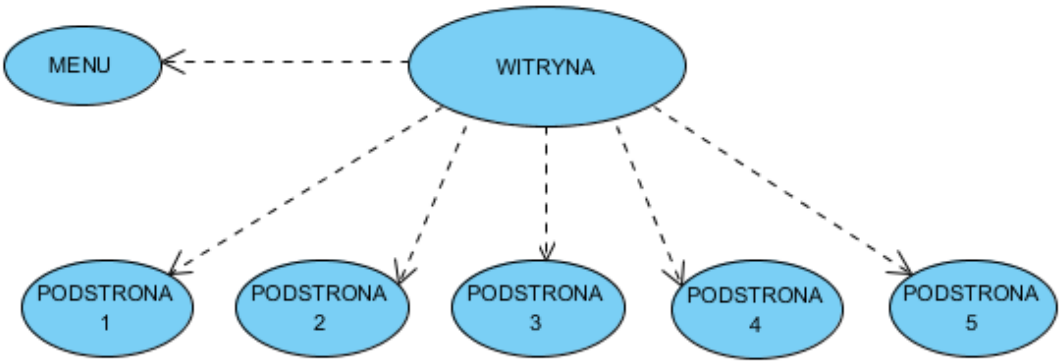
4.3 Moduły systemu

Moduł administracyjny został zrealizowany przy wykorzystaniu wzorca architektonicznego *MVVM*, jest to aplikacja utworzona w technologii *WPF*. Na rysunku 3.3 zostały przedstawione klasy wchodzące w skład aplikacji administracyjnej.

Rysunek 3.3. Klasy modułu administracyjnego

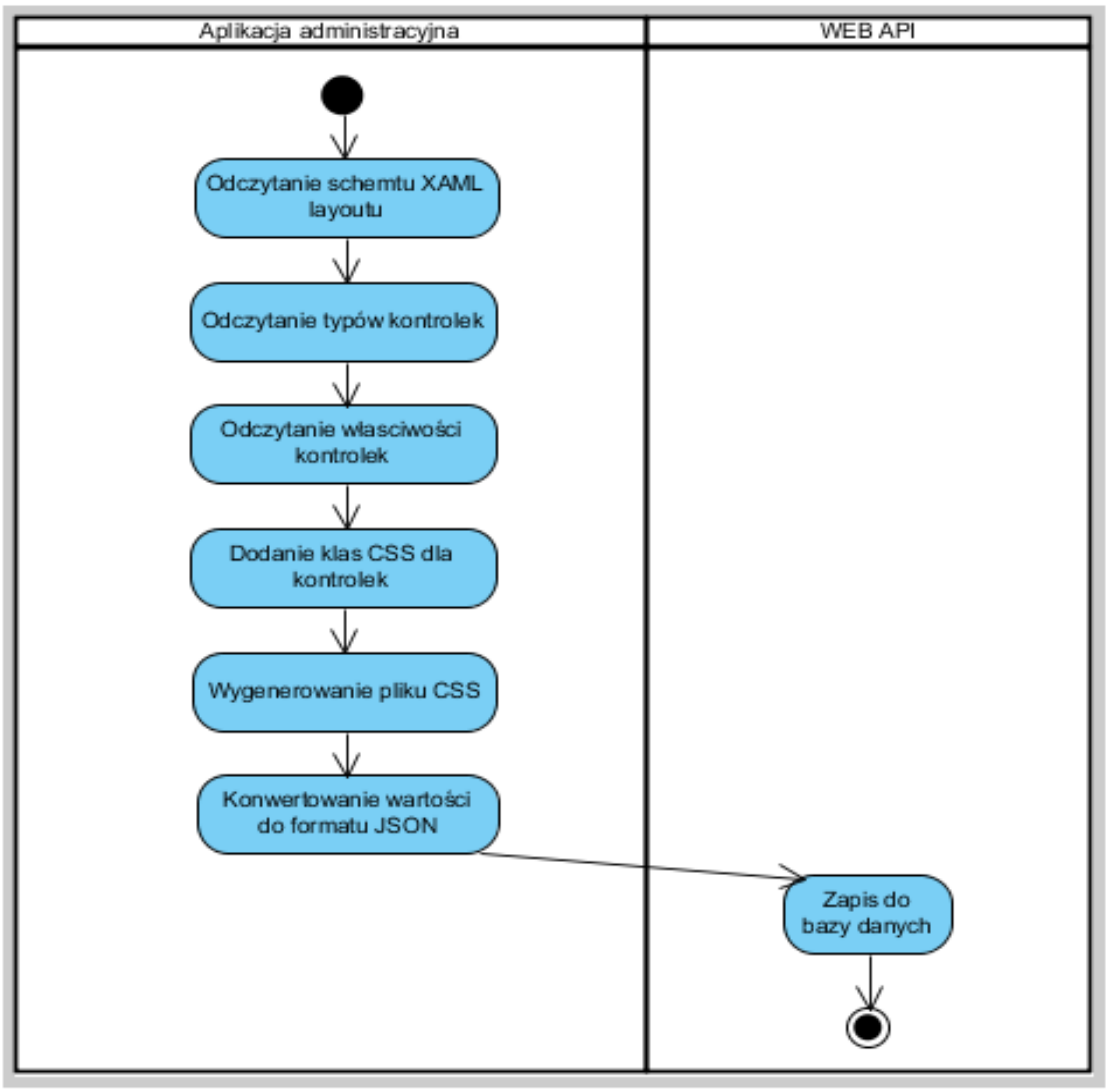
Poniżej wymienia się główne funkcje realizowane przez moduł administracyjny.

1. Tworzenia instancji projektu witryny internetowej, na rysunku 3.4. przedstawiono schemat tworzony we własnym standardzie ukazujący model budowy pojedynczej witryny internetowej. Owalne elementu reprezentują poszczególne komponenty. Kierunek strzałki łączącej te komponenty wskazuje na element podrzędny. Na schemacie widać, że pojedyncza witryna może składać się z jednego menu oraz wielu podstron których liczba może być zmieniana dynamicznie.



Rys.3.4. Model budowy witryny

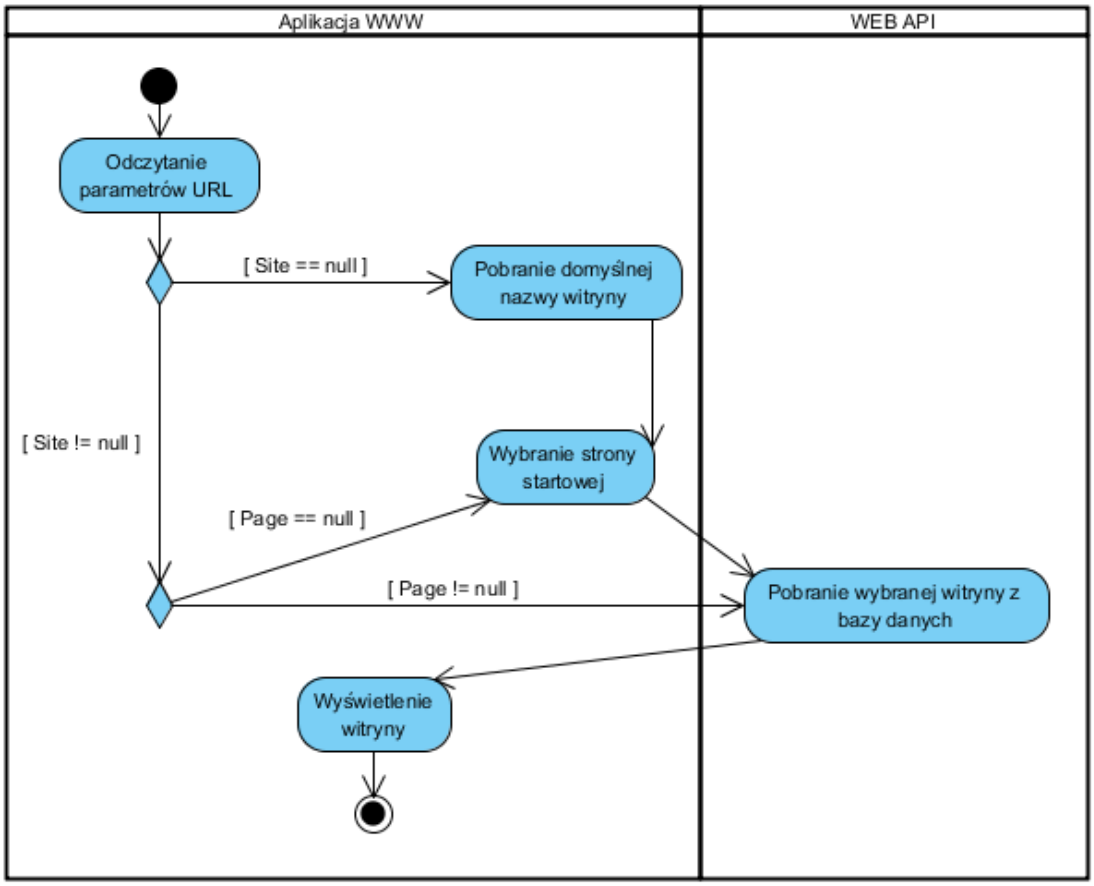
2. Wypełnianie zawartością pojedynczych kontrolek
3. Modelowanie wyglądu pojedynczych kontrolek. Dla utworzonych kontrolek definiowane są właściwości możliwe do edycji, odpowiadają one za:
   - kolor tła
   - kolor zawartości (dotyczy przycisków i pól do wprowadzania),
   - kolor czcionki,
   - rozmiar czcionki,
   - akcja wykonywana po kliknięciu elementu (dotyczy przycisków i linków),
   - wyrównanie elementów w pionie i poziomie,
   - liczba zajmowanych kolumn przez element.
4. 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 3.5 przedstawia diagram aktywności *UML* zapisu siatki kontrolek.



Rys.3.5. Kroki zapisu layoutu

Moduł strony WWW zawiera aplikacje *ASP.Net MVC* tworząca witrynę internetową na podstawie schematu kontrolek wcześniej utworzonego w module administracyjnym Poniżej wymieniono elementy wchodzące w skład tego modułu.

1. `HomeController` będący kontrolerem odpowiedzialny za pobranie z modelu i przekazanie do widoku odpowiedniej strony internetowej zamodelowanej w aplikacji administracyjnej. Rysunek 3.6 przedstawia diagram aktywności *UML* działania wybierania strony internetowej do załadowania.



Rys.3.6. Diagram aktywności 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.

2. `Index` jest to widok który, odpowiada za generowanie layoutu kontrolek aktualnie ładowanej podstrony.
3. `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*.

5. Realizacja

5.1 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 metodą 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 usunięty 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 języka *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 kontrolk0i *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.

5.2 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.

5.3 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`.

6. Użytkowanie

6.1 Wprowadzenie

Rozdział zawiera opis interfejsu użytkowania modułu administracyjnej, opisano w nim wszystkie okna panelu oraz przedstawiono funkcje jakie są w nich realizowane. Przedstawiono również wynik przeprowadzonych testów aplikacji po zakończeniu etapu implementacji. Testy dotyczyły utworzenia przykładowej strony internetową z wykorzystaniem projektowanego systemu zarządzania treścią.

6.2 Interfejs użytkownika

Aplikacja administracyjna składa się z czterech okien

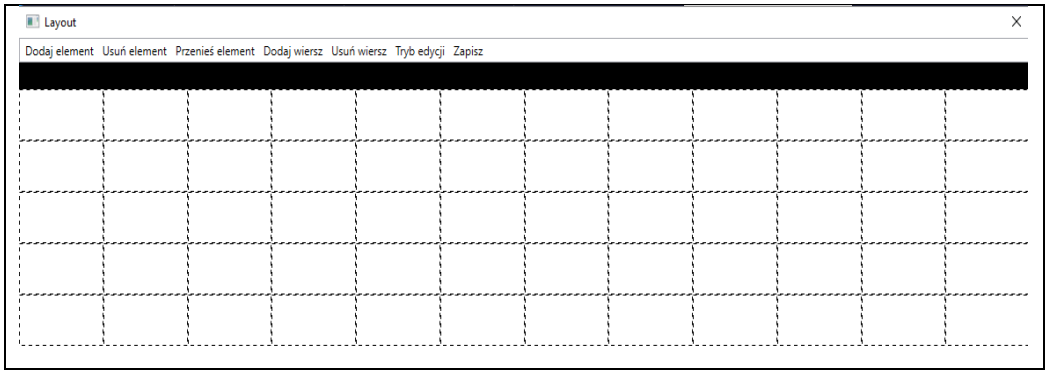
1. Okna głównego wyświetlanego jako okno startowe aplikacji. Rysunek 6.1 przedstawia wygląd tego okna. Znajdują się w nim dwa elementy:
    - panel zarządzania projektami w którym znajduje się drzewo projektów w którym przy przechodzeniu przez elementy drzewa następuje otwarcie okna layoutu dla wybranego menu lub podstrony.
    - przycisków nawigacyjnych umożliwiających dodawanie nowych elementów do projektów oraz usuwaniem już istniejących.

Rys.6.1. Okno główne

2. Okna layoutu służącego do projektowania menu oraz podstron witryny. Rysunek 6.2 przedstawia wygląd tego okna składającego się z dwunastokolumnowej siatki

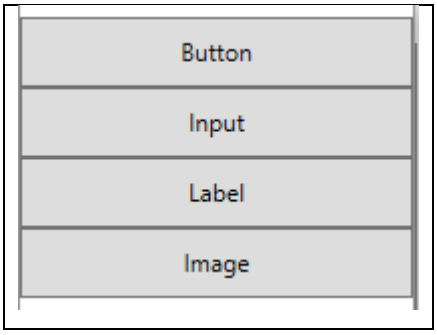
na których umieszczane są kontrolki oraz pasku przycisków nawigacyjnych umożliwiających

- dodanie nowej kontrolki
- usunięcie istniejącej kontrolki
- przeniesienie elementu
- dodanie wiersza
- usunięcie wierszy
- włączenie trybu edycji kontrolki
- zapisanie schematu strony do bazy danych.



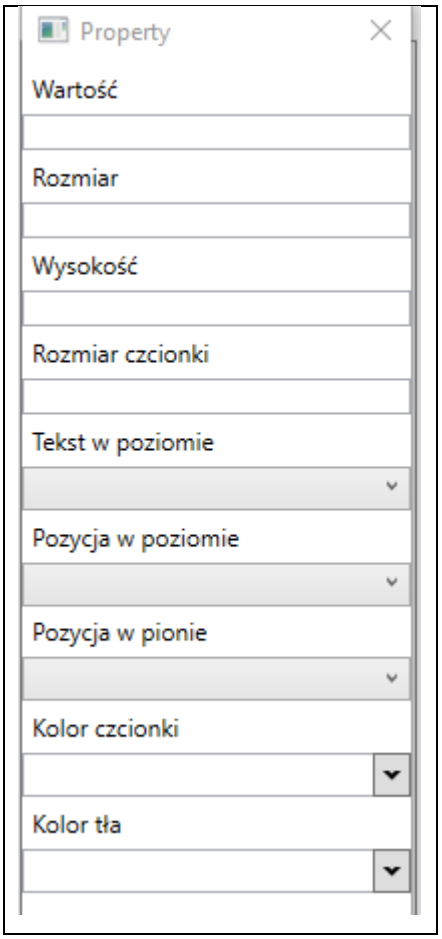
Rys.6.2. Okno layoutu

3. Okna kontrolek służącego do wybieranie nowych kontrolek do dodania na siatkę. Użytkownik po wybraniu trybu dodawania nowych kontrolek wybiera w tym oknie typ elementu, następnie umieszcza go w oknie layoutu. Rysunek 6.3 przedstawia wygląd tego okna.



Rys.6.3. Okno kontrolek

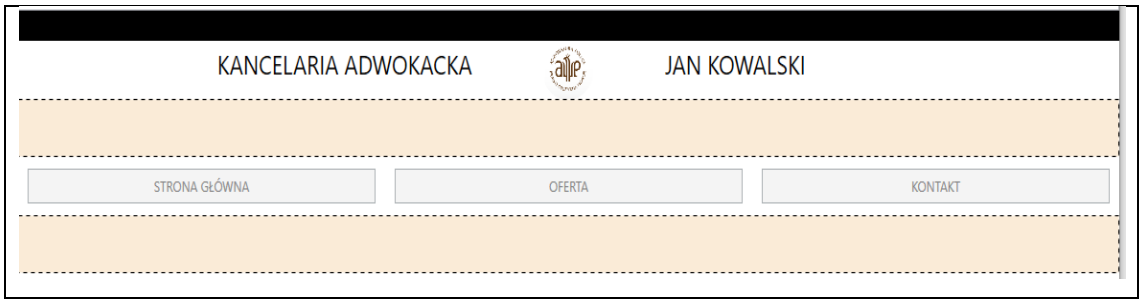
4. Okna właściwości służącego do edytowania zawartości i wyglądu pojedynczej kontrolki. Po wybraniu kontrolki na oknie layoutu następuje wypełnienie właściwości aktualnymi wartościami dla tej kontrolki. Użytkownik może edytować właściwości co powoduje aktualizacje siatki kontrolek. Rysunek 6.4 przedstawia wygląd tego okna.



Rys.6.4. Okno właściwości

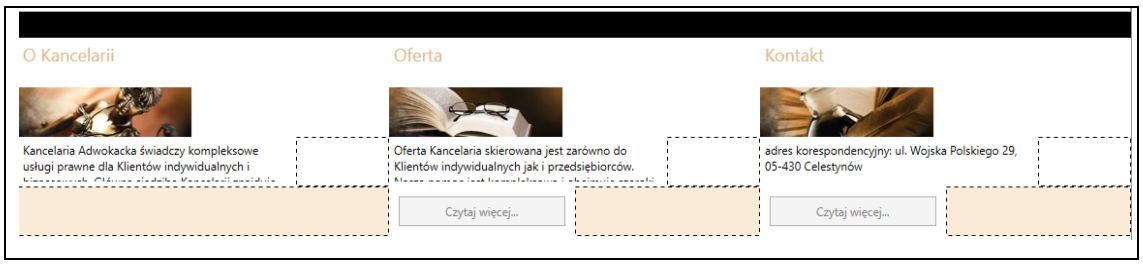
6.3 Testy

W ramach testów utworzono przykładowa stronę internetowa reklamująca kancelarie adwokacką składająca się ze strony głównej, podstrony opisującej ofertę oraz podstrony informującej o kontakcie z adwokatem. Rysunek 6.5 przedstawia wygląd ułożenia kontrolek na layoucie menu. Na siatce kontrolek położone są trzy przyciski, obraz i dwa napisy.



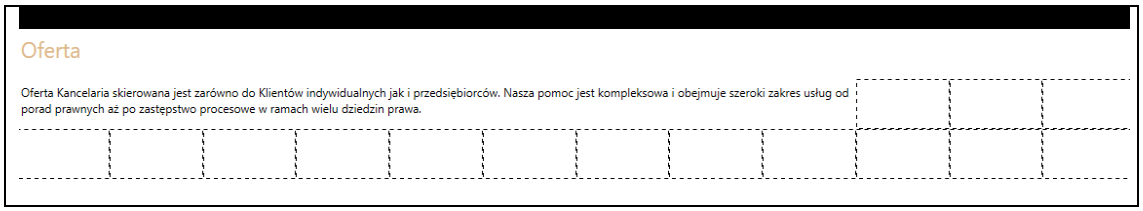
Rys.6.5. Layout menu

Rysunek 6.6 przedstawia wygląd ułożenia kontrolek na layoucie strony głównej. Na siatce kontrolek położone są dwa przyciski, trzy obrazy i sześć napisy o różnych kolorach i rozmiarze czcionki.



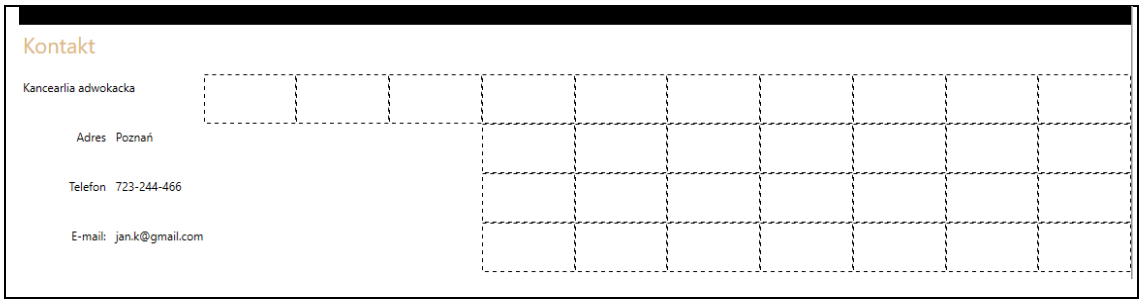
Rys.6.6. Layout strony głównej

Rysunek 6.7 przedstawia wygląd ułożenia kontrolek na layoucie podstrony „Oferta”. Na siatce kontrolek położone są dwa napisy o różnych kolorach i rozmiarach czcionki.



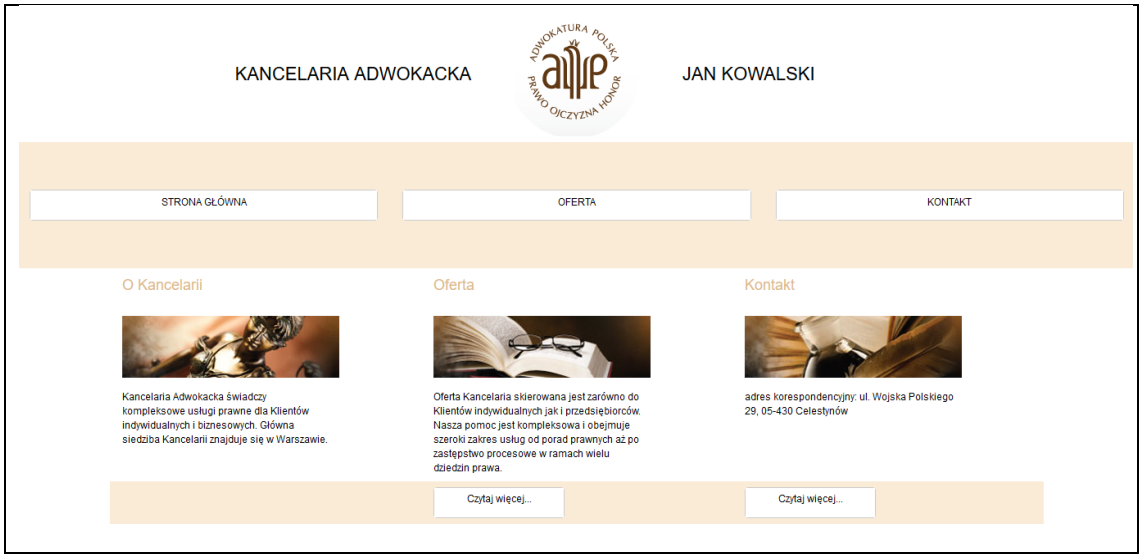
Rys.6.7. Layout podstrony „Oferta”

Rysunek 6.8 przedstawia wygląd ułożenia kontrolek na layoucie podstrony „Kontakt”. Na siatce kontrolek położone jest osiem napisów o różnych kolorach i rozmiarach czcionki



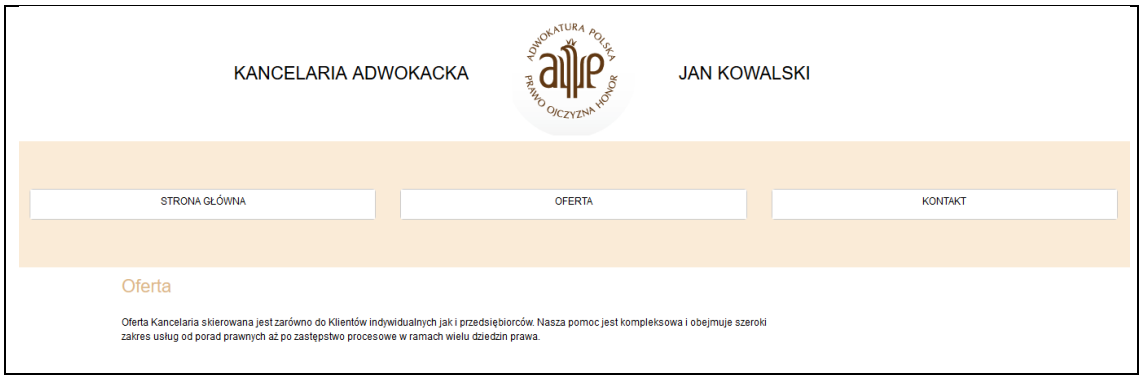
Rys.6.8. Layout podstrony „Kontakt”

Rysunek 6.9 przedstawia wygląd wygenerowanej strony internetowej. W górnej części witryny znajduje się menu z przyciskami przenoszącymi użytkownika na inne podstrony. Poniżej znajduje się layout strony głównej na którym również występują przyciski przenoszące do innych layoutów.



Rys.6.9 Strona główna

Rysunek 6.10 przedstawia wygląd podstrony „Oferta”. W górnej części witryny znajduje się menu z przyciskami przenoszącymi użytkownika na inne podstrony. Poniżej znajduje na temat oferty usług świadczonych przez kancelarie



Rys.6.10. Podstrona „Oferta”

Rysunek 6.11 przedstawia wygląd podstrony „Kontakt”. W górnej części witryny znajduje się menu z przyciskami przenoszącymi użytkownika na inne podstrony. Poniżej znajdują się dane kontaktowe kancelarii.

Rys.6.11. Podstrona „Kontakt”

7. Podsumowanie

W pracy zostały zrealizowane następujące cele:

- utworzenie systemu zarzadzania treścią umożliwiającego tworzenie wiele stron internetowych w jednym miejscu
- utworzenie aplikacji nie wymagającej od użytkownika wiedzy technicznej
- utworzenie systemu opartego na najbardziej popularnych technologiach
- utworzenie systemu umożliwiającego szybkie utworzenie 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 modułu administracyjnego. 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 utworzenie 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.

8. 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, 2012.

[5] Shreves R**.,** *Joomla! Biblia*, Helion, 2013.

[6] Dokumentacja biblioteki Bootstrap, `https://getbootstrap.com`, (dostęp: 21.01.2018).

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