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 6
System zarządzania treścią 6
Podobne systemy na rynku 6
1. Cel i zakres pracy 8
1.1 Przeznaczenie i zadania projektowanego systemu 8
1.2 Zrealizowane funkcjonalności: 8
1.3 Struktura pracy 8
2. Metody modelowania i realizacji 10
2.1 Wzorce projektowe 10
2.1.1 MVC 10
2.1.2 MVVM 10
2.2 Technologie programistyczne 11
2.2.1 C# 11
2.2.2 WPF 11
2.2.3 ASP.NET Core 11
2.2.4 Bootstrap 11
2.2.1 JavaScript 11
2.2.2 Entity Framework 11
2.2.3 XAML 12
2.2.4 UML 12
2.2.5 JSON 12
2.2.6 System kontroli wersji (*SVN*) 12
2.3 Środowiska programistyczne 12
2.3.1 Microsoft Visual Studio 12
2.3.2 SQL Server Management Studio 12
2.3.3 Visual Paradigm 13
3. Model systemu 14
3.1 Architektura systemu 14
3.1.1 Architektura pionowa 14
3.1.2 Architektura pozioma 15
3.2 Model bazy danych 15
3.3 Moduły systemu 17
3.3.1 Aplikacja administracyjna 17
3.3.2 Aplikacja WWW 17
3.3.3 WEB API 17
4. Realizacja 18
4.1 Aplikacja administracyjna 18
4.1.1 Klasa `LayoutControler` 18
4.1.2 Klasa `LayoutControl` 18
4.1.3 Klasa `Translator` 20
4.1.4 Klasa `StyleBuilder` 20
4.1.5 Klasa `ControlCounter` 21
4.2 Aplikacja WWW 21
4.3 WEB API 22
5. Użytkowanie 24
5.1 Instrukcja obsługi 24
5.2 Instrukcja instalacji 24
5.3 Testy 24
6. Podsumowanie 25

6.1 Zrealizowane cele 25
6.2 Napotkane problemy 25
6.3 Kierunki rozwoju 25
7. Bibliografia 26
8. Załączniki 27

Wstęp

Wybór tematu był ukierunkowany chęcią stworzenia systemu informatycznego, dzięki któremu użytkownik będzie miał możliwość w prosty sposób stworzyć 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.

Rodzaje systemów zarządzania treścią:

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

Podobne systemy na rynku

- WordPress – jest to darmowy open source system kontroli wersji 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! -

1. Cel i zakres pracy

W rozdział zostanie omówione przeznaczenie oraz zadania projektowanego systemu zarządzania treścią. Poruszane zostaną również aspekty zagadnienia które zostały zrealizowane podobnie lub pominięte w porównaniu z istniejącymi systemami na rynku. Zostaną również przedstawione funkcjonalności nie występujące w podobnych rozwiązaniach. 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 dowolnyy 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 instrukcje obsługę oraz instalacji stworzonego 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*, 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 od firmy Microsoft. Został wprowadzony w 2016 roku jako nowa generacja freamowork ASP .NET. Umożliwia tworzenie aplikacji Web z wykorzystaniem wzorca projektowego MVC.

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. 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.

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

3.1.1 Architektura pionowa

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 xxx została przedstawiony schemat działania systemu po zastosowaniu tej architektury.

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 w postaci responsa zawierającego informacje o skutku wykonanej operacji. W warstwie prezentacji znajdują się aplikacja administracyjna oraz strona WWW. W trakcie

działania wysyłają obie aplikacje 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 wcześniej utworzonym projekcie strony internetowej.

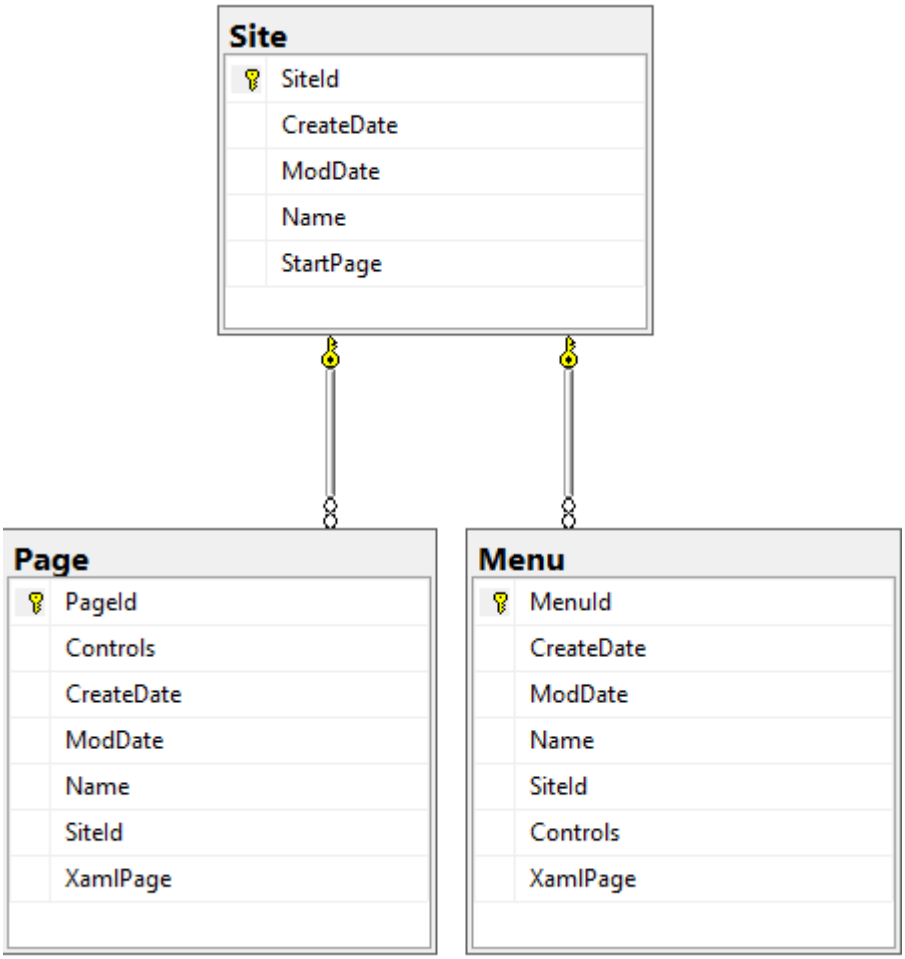
3.1.2 Architektura pozioma

3.2 Model bazy danych

Baza danych składa się z trzech tabel:

- Site – zawierającej witryny stworzone w aplikacji admnistracyjnej,
- 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, 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 okreśony 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

3.3.2 Aplikacja WWW

3.3.3 WEB API

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 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 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.

```
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 funkcje:

- `ConvaertToWebPage()` – zwraca listę wszystkich elementów typu `Menu`,
- `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 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.

4.2 Aplikacja WWW

Najważniejszymi funkcjami są:

- `CostructSite(siteName, pageName)` – jest to
- `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

Class `MenuControler` jest klasa dziedzicząco po klasie `Controller`. Odpowiada za obsługę operacji dokonywanych na tabeli `Menu` znajdującej się w bazie 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.

Class `PageControler` jest klasa dziedzicząco po klasie `Controller`. Odpowiada za obsługę operacji dokonywanych na tabeli `Page` znajdującej się w bazie 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.

Class `SiteControler` jest klasa dziedzicząco po klasie `Controller`. Odpowiada za obsługę operacji dokonywanych na tabeli `Site` znajdującej się w bazie 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 instrukcje obsługi projektu oraz instrukcje instalacji i konfiguracji aplikacji przed pierwszym uruchomieniem. W podrozdziale „Testy” zostały umieszczone informacje o przeprowadzonych testach aplikacji po zakończeniu etapu implementacji.

5.1 Instrukcja obsługi

5.2 Instrukcja instalacji

5.3 Testy

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 – umożliwiło by to stworzenie bardziej przejrzystych layoutów dla urządzeń mobilnych
- możliwość dodawania animacji do strony www
- rozszerzenie aplikacji administracyjnej o autoryzacje 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] Ducket J., *JavasScript i JQuery Interaktywne strony WWW dla każdego,* Helion, Gliwice, 2015

8. Załączniki

Płyta DVD zawierająca:

- obraz maszyny wirtualnej z aplikacją,
- tekst pracy w formacie PDF,
- tekst polityki prywatności.