



Politechnika Gdańska
Wydział Elektroniki, Telekomunikacji i
Informatyki



Katedra/Zakład: Inżynieria oprogramowania
Kierunek studiów: Informatyka
Specjalność: -
Rodzaj studiów: stacjonarne
Imię i nazwisko: MATEUSZ KOWALSKI
Numer albumu: 125632

Katedra/Zakład: Systemy geoinformatyczne
Kierunek studiów: Informatyka
Specjalność: -
Rodzaj studiów: stacjonarne
Imię i nazwisko: TYMOTEUSZ GACH
Numer albumu: 125568

Katedra/Zakład: Algorytmy i modelowanie systemów
Kierunek studiów: Informatyka
Specjalność: -
Rodzaj studiów: stacjonarne
Imię i nazwisko: RAFAŁ LEWANDOWSKI
Numer albumu: 125664

PROJEKT DYPLOMOWY
INŻYNIERSKI

Temat projektu:

Serwis internetowy przeznaczony dla urządzeń przenośnych i desktopowych.

Zakres projektu:

Aplikacja internetowa wspomagająca tworzenie prostych stron internetowych na urządzenia desktopowe z automatyczną konwersją na urządzenia mobilne.

Potwierdzenie przyjęcia projektu:

Opiekun projektu

.....
.....

Tytuł, imię i nazwisko

Kierownik Katedry/Zakładu

.....
.....

Tytuł, imię i nazwisko

Gdańsk, 12.12.2012r.

Spis treści

1.	Wstępny plan projektu	5
1.1.	Wstęp.....	5
1.2.	Cele projektu	5
1.3.	Zespół projektu.....	6
2.	Plan projektu.....	8
2.1.	Organizacja projektu.....	8
2.2.	Aspekty techniczne	8
2.2.1.	Metody, narzędzia i techniki	8
2.2.2.	Dokumentacja oprogramowania	10
2.3.	Komunikacja	11
2.3.1.	Wewnętrzna komunikacja.....	11
2.3.2.	Komunikacja z promotorem	11
2.4.	Zarządzanie ryzykiem.....	11
2.4.1.	Czynniki ryzyka wychodzące ze środowiska zespołu	11
2.4.2.	Strategia minimalizacji ryzyka	12
2.5.	Plan zapewnienia jakości	12
2.5.1.	Opinie i audyty.....	12
2.5.2.	Testy.....	13
2.5.3.	Zgłaszanie problemów i działania naprawcze	13
2.6.	Harmonogram	14
3.	Model przypadków użycia	15
3.1.	Diagram przypadków użycia	15
3.2.	Opis aktorów	15
3.2.1.	Administrator strony.....	15
3.2.2.	Użytkownik desktopowy	16
3.2.3.	Użytkownik mobilny.....	16
3.3.	Opis przypadków użycia.....	16
3.3.1.	Dodanie elementów z „toolbox’a”	16
3.3.2.	Dodanie podstrony	16
3.3.3.	Edytowanie istniejących elementów	17
3.3.4.	Edytowanie podstron.....	17
3.3.5.	Konfiguracja elementów	18

3.3.6.	Konfiguracja punktu na mapie.....	19
3.3.7.	Konfiguracja ustawień administratora	19
3.3.8.	Logowanie	19
3.3.9.	Przeglądanie wersji desktopowej	20
3.3.10.	Przeglądanie wersji mobilnej.....	20
3.3.11.	Zmiana kolejności elementów na stronie	20
3.3.12.	Zmiana koloru	20
4.	Architektura systemu	22
4.1.	Architektura	22
4.2.	Dane.....	23
4.3.	Komunikacja JavaScript – ASP.NET	24
5.	Baza danych	25
5.1.	Struktura bazy danych	25
5.2.	Opis tablic bazy danych i ich atrybutów.	25
6.	Opis implementacji	27
6.1.	Interfejs i działanie systemu.....	27
6.2.	Logowanie do systemu DotNetCms.....	38
6.3.	Zmiana kolorystyki strony.....	39
6.4.	Dodawanie i edytowanie podstron.	40
6.5.	Zmiana hasła redaktora strony.....	41
6.6.	Dodawanie i rozmieszczanie elementów z paska narzędzi.....	41
6.7.	Dodawanie i edycja tekstu za pomocą edytora oraz implementacja nowej kontrolki	42
6.8.	Użycie gadżetu zakładka (Tab)	43
6.9.	Tworzenie narzędzia Accordion	44
6.10.	Dodawanie, edycja i skalowanie zdjęć.....	44
6.11.	Dodawanie mapy i rozmieszczanie punktów	46
6.12.	Umieszczanie podowiedzi o edycji gadżetów	47
6.13.	Implementacja instalatora	47
7.	Opis instalacji	48
7.1.	Przygotowanie środowiska	48
7.2.	Instalacja	49
7.3.	Uruchamianie	49
8.	Raport końcowy	51
8.1.	Osiągnięte rezultaty.....	51

8.2.	Podział wykonanej pracy między członków grupy projektowej.....	51
8.3.	Podsumowanie.....	54
9.	Bibliografia.....	56

1. Wstępny plan projektu

1.1. Wstęp

Temat projektu: Serwis internetowy przeznaczony dla urządzeń desktopowych i mobilnych (ang. Web Internet services for mobile and desktop devices).

Serwis DotNetCms jest to aplikacja typu CMS (Content Management System – z ang. System Zarządzania Treścią) służąca do łatwego i szybkiego tworzenia stron. Oddziela ona treść (zawartość informacyjną strony) od wyglądu (sposobu jej prezentacji). Projekt kierowany jest do małych firm i osób prywatnych, które chcą posiadać własną elementarną stronę WWW w formie wizytówki. Mogą w niej umieszczać podstawowe informacje o swojej firmie, produktach, czy formie kontaktu. Aplikacja wykorzystuje przyjazny dla użytkownika system Drag&Drop w celu umieszczenia kontrolerek na stronie. Dostosowana jest w taki sposób, by w atrakcyjny sposób wyświetlać podstawowe informacje o swojej działalności biznesowej.

Wraz z coraz bardziej rozwijającą się technologią mobilną, dużą częścią użytkowników Internetu są posiadacze urządzeń przenośnych. Jedną z najbardziej atrakcyjnych i wyróżniającą się od innych aplikacji tego typu funkcjonalnością projektu jest dostosowywanie zawartości strony do wielkości okna przeglądarki internetowej w urządzeniu. W ten sposób użytkownik mobilny korzysta ze specjalnego interfejsu aplikacji, który jest kompatybilny z jego urządzeniem.

1.2. Cele projektu

- Stworzenie przyjaznego i prostego w obsłudze narzędzia.
- Stworzenie aplikacji, która pozwala na:
 - Przedstawienie wizytówki swojej firmy w Internecie dla osób, które są laikami w dziedzinie tworzenia stron WWW.
 - Stworzenie strony internetowej bez większej wiedzy o technologii.
 - Możliwość edytowania wyglądu interfejsu strony, bez konieczności ingerencji w kod HTML czy CSS.
 - Możliwość łatwej zmiany treści strony.
 - Dostosowanie strony do urządzeń mobilnych.
 - Łatwe edytowanie serwisu dla osób, które chciałyby zmienić aplikację na poziomie kodu HTML, JavaScript czy CSS.
- Zmniejszenie kosztów tworzenia strony. Klient naszej aplikacji dużo zaoszczędzi robiąc własną stronę samemu, a nie zlecając jej stworzenie innej firmie.

1.3. Zespół projektu

- **Tymoteusz Gach** - programista
Doświadczenie: IHS (praktykant, programista w technologii .NET), Gdańska Fundacja Kształcenia Menadżerów (tworzenie dedykowanej aplikacji internetowej), student.
Umiejętności: znajomość platformy .NET, MySQL, Android, XHTML, Zend Framework.
Odpowiada za: logika aplikacji.
- **Mateusz Kowalski** – programista
Doświadczenie: firma EXE (młodszy programista baz danych), hiszpańska firma INDRA (praktyki: programista i analityk w Business Intelligence), student.
Umiejętności: znajomość platformy .Net, MSSQL, Android, jQuery.
Odpowiada za: model danych, logika aplikacji, raportowanie.
- **Rafał Lewandowski** – programista (kierownik projektu)
Doświadczenie: firma Starsoft S.C. (programista .NET), hiszpańska firma QueHoteles.com (programista PHP/MySQL/JavaScript), student.
Umiejętności: znajomość platformy .NET, PHP, JavaScript, MS SQL, MySQL.
Odpowiada za: funkcjonalność, logika aplikacji.
- **Promotor:** dr inż. Jerzy Demkowicz

Podział obowiązków przedstawia poniższy macierz odpowiedzialności.

	Mateusz Kowalski	Tymoteusz Gach	Rafał Lewandowski
Zarządzanie	O	O	O
Szkielet	O	O	O
Dodawanie elementów	O	O	O
Accordion			O
Zakładki (Taby)	O		
Edytor tekstu			O
Nowa kontrolka w edytorze		O	
Dodawanie zdjęć	O		
Ładowanie zdjęć na serwer			O
Dostosowywanie zdjęć do urządzenia		O	
Mapy		O	
Interfejs	O		
Zmiana koloru	O		
Model danych	O		O
Podstrony	O		O
Serwer			O
Logowanie			O
Zarządzanie administratorem			O
Ajax	O	O	O

DotNetCMS

Dokumentacja	O		
Instalator			O
Testowanie	O	O	O

Diagram 1: Macierz odpowiedzialności. (Źródło: opracowanie własne)

2. Plan projektu

2.1. Organizacja projektu

2.1.1. Projektanci

Projekt wykonywały 3 osoby o takich samych rolach. Każdy członek grupy miał wyznaczone zadania, ustalone między sobą, za które był odpowiedzialny. Projektanci mieli obowiązek dbania o funkcjonalność i jakość wykonania swoich zobowiązań. Musieli integrować swoje rozwiązania w sposób zrozumiały dla każdego członka grupy projektowej. Każdy mógł zmienić dowolny fragment systemu.

Aplikacja była programem wysokiego ryzyka, ponieważ nie było wiadomo do końca, co i jak prawidłowo zrobić, dlatego projekt był wykonywany w metodyce ekstremalnego programowania. Projektanci po wspólnych spotkaniach decydowali, kto i co robi. Po wykonaniu wyznaczonych powinności i skomunikowaniu się z resztą grupy, ustalane były następne zadania wobec wykonawców aplikacji. Nigdy nie projektowano z góry, ponieważ ciężko było przewidzieć jaka architektura będzie najlepsza dla danego problemu, dlatego tworzyło się ją w miarę rozszerzenia programu.

Częstą formą współpracy między realizatorami projektu było programowanie w parach. Polegało to na tym, że jedna osoba pracowała przy komputerze i pisała kod, a druga zgłaszała poprawki i swoje sugestie. Technika ta umożliwiała szybkie wyłapywanie błędów, oraz poprawę, jakości kodu, co w efekcie dawało taką samą ilość kodu, natomiast był on bardziej dopracowany i przejrzysty.

2.1.2. Opiekun projektu.

Opiekun projektu był odbiorcą systemu. Definiował swoje wymagania i inicjalizował nowe działania. Musiał być dla członków zespołu organizatorem, negocjatorem i strategiem. Do jego obowiązków należało również przewidywanie skutków poszczególnych posunięć realizowanych przez członków zespołu.

2.2. Aspekty techniczne

2.2.1. Metody, narzędzia i techniki

Wybór narzędzi i technologii w celu realizacji projektu był ustosunkowany do preferencji członków zespołu. Również w znacznej mierze był narzucony osiągalnością narzędzi. Priorytetem był dobór narzędzi ogólnie dostępnych z możliwością jego nieodpłatnego wykorzystania w ramach realizacji aplikacji.

Główne narzędzia wykorzystywane podczas etapu analizy to aplikacja ArgoUML i pakiet Microsoft Office. ArgoUML jest to aplikacja pozwalająca na programowanie obiektowe w języku UML. Argumentami przemawiającymi za wyborem tego oprogramowania było doświadczenie w korzystaniu tego narzędzia na innych przedmiotach realizowanych w trakcie studiowania. Aplikacja jest bezpłatna i można ją pobrać ze strony <http://argouml.tigris.org/>.

Zdecydowanie większa ilość narzędzi była wykorzystywana w fazie implementacji projektu. Aplikacja, jaką jest serwis internetowy, była realizowana w technologii ASP.Net z wykorzystaniem biblioteki jQuery. Z powodu braku doświadczenia w języku Visual Basic i małej popularności J#, które również są obsługiwane przez ASP.Net, implementowano w języku C#, choć zdecydowanie większą część logiki aplikacji napisano w języku JavaScript. Środowiskiem programistycznym, które zostało wybrane w celu implementacji, było Microsoft Visual Studio 2012, będące w czasie realizacji projektu najnowszą wersją tego narzędzia dla programistów. Użycie wspomnianego produktu było możliwe dzięki programowi DreamSpark Premium for Academic Institutions, który pozwala studentom na darmowe pobranie i wykorzystanie oprogramowania w celach edukacyjnych. Na potrzebę aplikacji serwer został założony w usłudze IIS oraz baza danych w SQL Server Compact Edition, które są kompatybilne z resztą środowiska programistycznego.

Narzędzia:

- Framework:
 - .Net 4.0 – platforma programistyczna opracowana przez firmę Microsoft.
 - Gumbo – Responsive Web Design.
- Serwer:
 - IIS – zbiór usług internetowych dla systemów rodziny Microsoft.
- Baza danych:
 - SQL Server Compact Edition 4.0 – kompaktowa baza danych stworzona przez firmę Microsoft dla aplikacji działających na urządzeniach mobilnych i desktopowych.
- Biblioteki:
 - jQuery – biblioteka programistyczna dla języka JavaScript.
 - jHtmlArea – wizualny edytor HTML.
- Komunikacja:
 - Gadu – Gadu – komunikator internetowy.
 - Skype – komunikator internetowy umożliwiający wideo konferencje i przesyłanie plików.
 - Teamviewer – narzędzie do zdalnej kontroli i zdalnym współdzieleniem pulpitu między komputerami.
- Dokumentacja:
 - Microsoft Office 2010 – pakiet aplikacji biurowych.
 - Adobe Acrobat – program do przeglądania plików PDF.
- Współdzielenie dokumentów i kodu:

- GitHub – <https://github.com/InzynierkaKsg/> hostingowy serwis internetowy wykorzystujący system kontroli wersji Git.
 - Dropbox – usługa polegająca na udostępnianiu przestrzeni dyskowej w celu synchronizacji plików między komputerami.
- Tworzenie kodu:
 - Microsoft Visual Studio 2012 – zintegrowane środowisko programistyczne.
- Wspomaganie modelowania:
 - ArgoUML – aplikacja przeznaczona do programowania obiektowego w języku UML.
- Wspomaganie testowania:
 - Microsoft Visual Studio 2012 – zintegrowane środowisko programistyczne.
 - Google Chrome – przeglądarka internetowa rozwijana przez firmę Google.
 - Nokia E7 z zainstalowaną przeglądarką OperaMobile – smartphone.
 - Nokia 710 z wbudowaną przeglądarką Internet Explorer – smartphone.
- Wspomaganie zarządzania:
 - Microsoft Office 2010 – pakiet aplikacji biurowych.

2.2.2. Dokumentacja oprogramowania

- Opisuje stworzenie i działanie projektu.
- Podaje opis organizacji i elementy zarządzania projektem.
- Wykonana przez 1 osobę (Mateusz Kowalski). Informacje na temat realizacji zadań wykonanych przez innych członków zespołu zbierane były wraz z rozwojem dokumentacji.
- Jeden dokument - nie było potrzeby wersjonowania.
- Zgodnie z metodyką ekstremalnego programowania, dokumentacja była realizowana na końcu tworzenia projektu, z wyjątkiem punktów rozdziałów 1 i 2, to jest planów projektu.

2.3. Komunikacja

2.3.1. Wewnętrzna komunikacja

Komunikacja w zespole, głównie, odbywała się drogą internetową poprzez korzystanie z komunikatorów takich jak Skype czy Gadu-Gadu, jak również z wykorzystaniem poczty elektronicznej. Dla szybkiego przesyłania plików został założony współdzielony, dostępny dla każdego członka zespołu folder, obsługiwany za pomocą programu Dropbox. W ten efektywny sposób można było przysyłać pomoce do projektu. W razie szczególnych problemów projektanci korzystali z narzędzia TeamViewer, które pozwala na zdalną kontrolę i zdalne współdzielenie pulpitu przez połączenie internetowe. Repozytorium założone na portalu <http://www.github.com> pozwalało na zarządzanie zmianami w kodzie źródłowym w trakcie rozbudowy projektu, oprócz tego w plikach źródłowych były umieszczane specjalne komentarze.

Poza drogą elektroniczną zespół spotykał się kilka razy w tygodniu, ze wcześniej ustaloną datą i miejscem spotkania, w celu ustalenia postępów oraz problemów, które pojawiały się w trakcie realizacji projektu, a także wspólnego pisanie kodu. Z powodu tego, że każdy członek teamu studiuje na innej katedrze, co oznaczało odmienne godziny trwania zajęć podczas semestru, nie było możliwości ustalenia stałego terminu spotkań.

2.3.2. Komunikacja z promotorem

Zespół spotykał się z promotorem raz w tygodniu o wcześniej ustalonej i zaakceptowanej przez wszystkich członków grupy godzinie. Spotkania miały na celu naprowadzić studentów na sposób w jaki powinni zrealizować projekt. Na każdym spotkaniu sprawdzana była obecność studentów oraz omówienie z promotorem postępu pracy.

Promotor miał dostęp do repozytorium, więc w każdej chwili mógł nadzorować poziom postępów. W nagłych wypadkach członkowie zespołu komunikowali się z promotorem drogą mailową i vice versa.

2.4. Zarządzanie ryzykiem

2.4.1. Czynniki ryzyka wychodzące ze środowiska zespołu

Największym czynnikiem zagrażającym zrealizowaniu projektu był termin jego oddania. Ostateczna data oddania serwisu, 11 grudnia 2012, nie mogła zostać przekroczona. Zespół był mało doświadczony w realizacji projektów takiej skali i planowania. Dodatkowo zadania związane z innymi przedmiotami podczas 7 semestru przyczyniały się do ograniczenia czasowego dla realizacji wyznaczonych zadań.

Kolejnym elementem zagrażającym projektowi była mało optymalna organizacja współpracy między członkami zespołu. Niestety, każdy z projektantów studiował na różnych katedrach:

- Mateusz Kowalski – Inżynieria oprogramowania
- Rafał Lewandowski – Algorytmy i modelowanie systemów
- Tymoteusz Gach – Systemy geoinformatyczne.

Różny plan zajęć ograniczał możliwości spotkań w celu omówienia aspektów tworzenia projektu i współpracy. Dodatkowo dwóch z trzech współtwórców miało obowiązki wobec firm, w których byli zatrudnieni, co w dużym stopniu zawężało operowanie wolnym czasem.

Kolejnym ryzykiem był brak doświadczenia. Projektanci, studenci oraz ludzie dopiero rozpoczynający pracę nie byli biegli w organizacji i tworzeniu takiego typu projektu. Możliwe, że wybór metodyki ekstremalnego programowania nie był odpowiedni do realizacji tej aplikacji przez te konkretne osoby.

2.4.2. Strategia minimalizacji ryzyka

Na cotygodniowych spotkaniach oceniane były postępy prac. Na tychże spotkaniach, po uwzględnieniu harmonogramu, podejmowano decyzje o zwiększeniu lub zmniejszeniu zakresu produktu. Opinie opiekuna projektu pomogły w ustaleniu wartości ryzyka w jakim projekt aktualnie się znajdował.

2.5. Plan zapewnienia jakości

2.5.1. Opinie i audyty

W ramach spotkań z opiekunem projektu, promotor oceniał powstającą pracę oraz sugerował różne funkcjonalności serwisu, których część zostało wdrożonych do systemu. Nadzorował pracę wraz z harmonogramem i wyznaczał terminy poszczególnych etapów projektu oraz przewidywał skutki decyzji zgłaszanych przez projektantów. Pomogło to w lepszej organizacji pracy i zarządzaniu projektem.

2.5.2. Testy

Wraz z rozwojem aplikacji przeprowadzone były testy jednostkowe, czyli testy na poziomie pojedynczych wydzielonych elementów technicznych projektu. Uwagi, które przekraczały zakres założeń były na bieżąco przekazywane osobie odpowiedzialnej za dany element wdrożenia. Wszystkie błędy były korygowane w jak najszybszym czasie, a testy ponawiane - aż do momentu wyeliminowania wszystkich nieprawidłowości.

Końcowe integralne testy wszystkich funkcjonalności były przeprowadzane za pomocą przeglądarki Google Chrome, która była w czasie realizacji projektu najbardziej popularną przeglądarką na świecie. Dodatkowo aplikacja była sprawdzana przez emulator Windows Phone udostępnioną przez narzędzie Microsoft Visual Studio 2012, w smartphon'ie Nokia E7 wykorzystując przeglądarkę Opera Mobile oraz Nokia 710 z wbudowaną przeglądarką Internet Explorer.

W ramach testów zabezpieczeń, w ASP.NET zostały dokonane usługi autoryzacji i uwierzytelnienia, które współpracują wspólnie z IIS.

2.5.3. Zgłaszanie problemów i działania naprawcze

Ze względu na małą wielkość projektu i bardzo dobrze skomunikowany i znający się zespół, członkowie zdecydowali nie zakładać systemu obsługi zgłoszeń. Każdy z członków zespołu, gdy zauważył jakiś błąd, starał się skomunikować się pozostałymi osobami grupy projektowej lub od razu naprawić powstały problem. Każdy z problemów był rozwiązywany na ile było to możliwe oraz na bieżąco.

2.6. Harmonogram

Harmonogram prac został przedstawiony w poniższym diagramie Gantta.

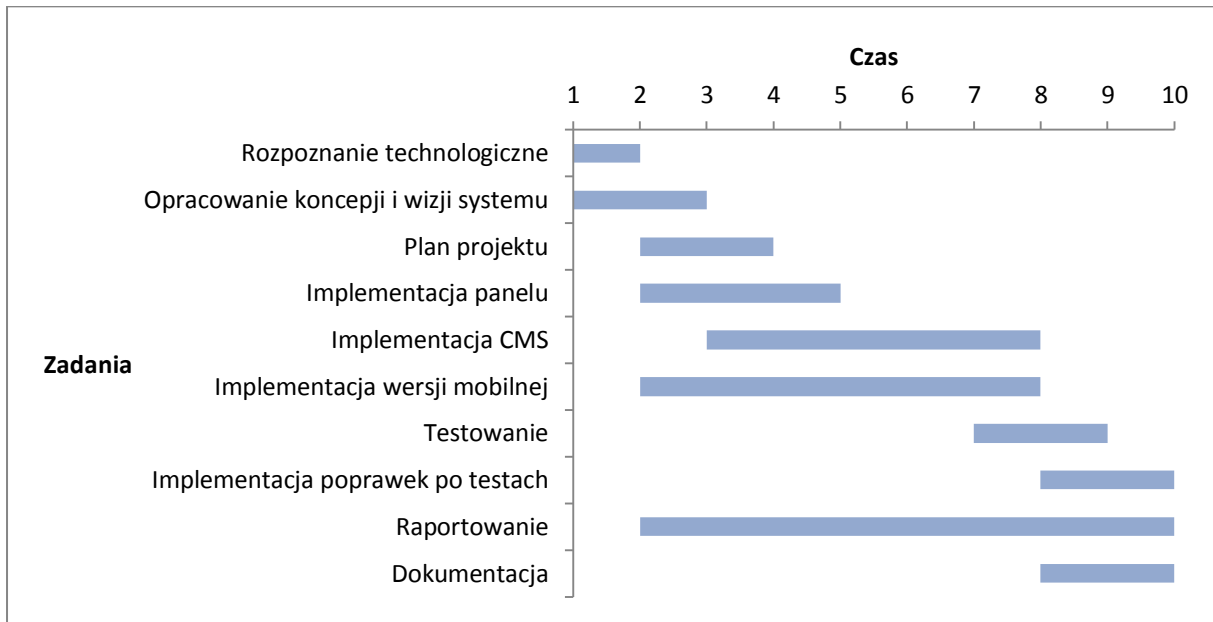
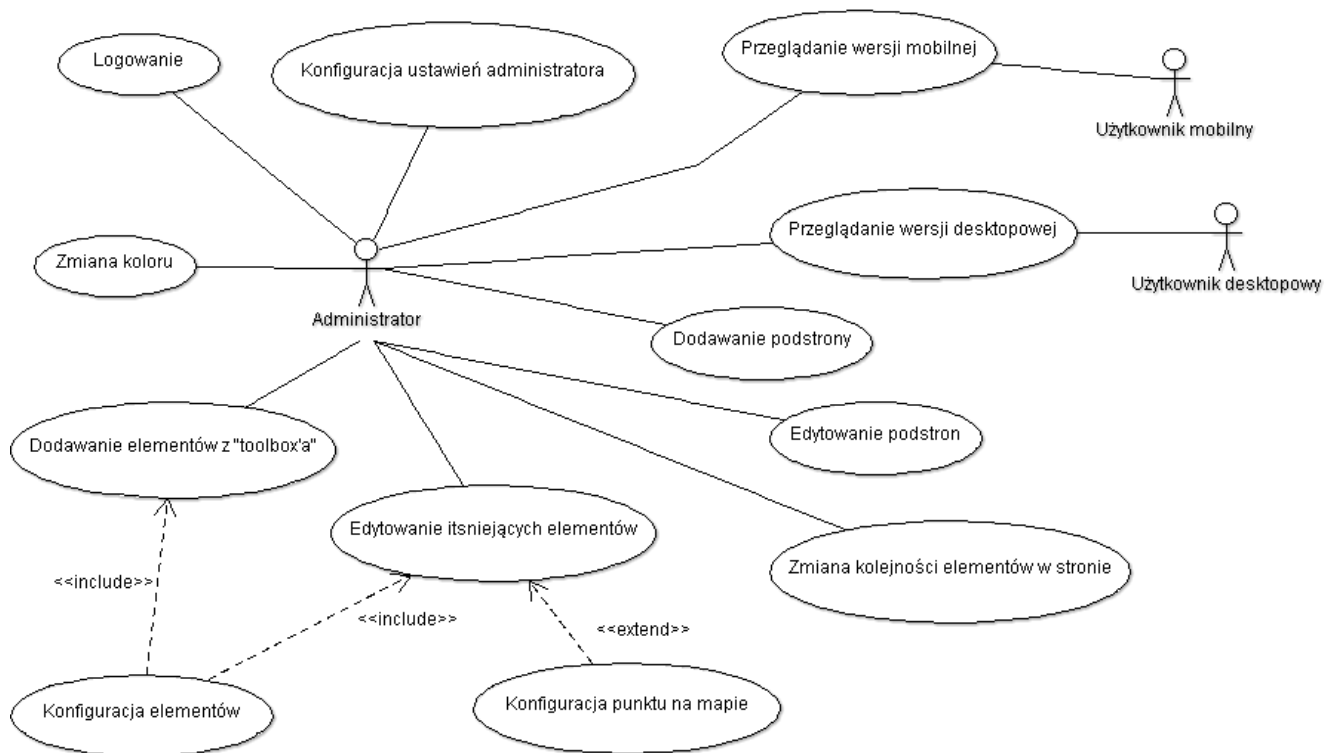


Diagram 2: Harmonogram. (Źródło: opracowanie własne)

3. Model przypadków użycia

3.1. Diagram przypadków użycia



Rysunek 1: Diagram przypadków użycia (Źródło: opracowanie własne).

3.2. Opis aktorów

3.2.1. Administrator strony

Redaktor systemu, główny użytkownik administracyjnej części serwisu. Po zalogowaniu jako administrator ma dostęp do wszystkich możliwości, jakie daje mu projekt. Głównym zadaniem administratora jest dbanie o wygląd i jakość merytoryczną serwisu internetowego. Decyduje o ilości podstron znajdujących się w serwisie. Może on dla każdej podstrony dodawać, edytować i usuwać elementy znajdujące się w systemie oraz wprowadzać zmiany nad treścią według własnego uznania lub wyznaczonych mu wytycznych. Administrator może się logować jedynie z urządzenia desktopowego.

3.2.2. Użytkownik desktopowy

Jeden z docelowych użytkowników systemu korzystający z urządzeń stacjonarnych. Potencjalny klient, który chciałby uzyskać informacje na temat firmy i jej produktów.

3.2.3. Użytkownik mobilny

Jeden z docelowych użytkowników systemu korzystający z urządzeń mobilnych. Potencjalny klient, który w szybkim czasie chciałby uzyskać informacje na temat firmy i jej produktów. Strona automatycznie dostosowuje się tak, żeby użytkownik w łatwy sposób, niezależnie od urządzenia (smartphone, tablet), mógł korzystać z funkcjonalności serwisu.

3.3. Opis przypadków użycia.

3.3.1. Dodanie elementów z „toolbox’a”.

3.3.1.1. Warunki

1. Użytkownik zalogowany.
2. Pokazany „toolbox”.

3.3.1.2. Ograniczenia

Dotyczy dolnego toolboxa.

3.3.1.3. Przebieg

1. Użytkownik wybiera dany element i metodą Drag&Drop umieszcza element w danym kontenerze.
2. System wyświetla okno dialogowe z konfiguracją danego elementu (przypadek użycia: Konfigurowanie elementów).

3.3.2. Dodanie podstrony

3.3.2.1. Warunki

1. Użytkownik zalogowany.
2. Pokazany „toolbox”.

3.3.2.2. Ograniczenia

Brak.

3.3.2.3. Przebieg

1. Użytkownik naciska przycisk „Add Page”.
2. System wyświetla okno dialogowe z formularzem dla nowej podstrony.
3. Użytkownik wpisuje nazwę strony.
4. Użytkownik naciska przycisk w oknie dialogowym:

- a. Użytkownik naciska przycisk „Add”. W przypadku, gdy użytkownik naciśnie przycisk „Add” bez wpisania nazwy, system wyświetla podpowiedź oraz podświetla input na czerwono. Dopóki input nie zostanie wypełniony, system nie reaguje na przycisk „Add”.
 - b. Użytkownik naciska przycisk „Cancel”. W tym przypadku system zamknie okno dialogowe bez dokonania żadnych zmian.
5. System zapisuje do bazy danych nową podstronę i odświeża pasek nawigacyjny.

3.3.3. Edytowanie istniejących elementów

3.3.3.1. [Warunki](#)

1. Użytkownik zalogowany.
2. Przynajmniej jeden element dodany.

3.3.3.2. [Ograniczenia](#)

Brak.

3.3.3.3. [Przebieg](#)

1. Użytkownik najeżdża na dany element myszką.
2. System wyświetla „tooltip”, w którym jest zawarta informacja jak dostać się do trybu edycji danego elementu.
3. Użytkownik naciska lub naciska podwójnie, w zależności od danego elementu.
 - a. W przypadku mapy pojawia się przycisk „Click to delete the map” na 1,4 sekundy. Użytkownik może nacisnąć na mapę w celu dodania punktu. Wtedy otwiera się okno dialogowe, które pozwala na dodawanie punktu na mapie (przypadek użycia: Konfiguracja punktu na mapie).
 - b. Otwiera się okno dialogowe z konfiguracją danego elementu (przypadek użycia: Konfigurowanie elementów).

3.3.4. Edytowanie podstron

3.3.4.1. [Warunki](#)

1. Użytkownik zalogowany.
2. Wyświetla się formularz z poprawnie istniejącymi już podstronami.

3.3.4.2. [Ograniczenia](#)

Brak.

3.3.4.3. [Przebieg](#)

1. Użytkownik naciska przycisk „Pages”.
2. System otwiera okno dialogowe, w którym wyświetlona jest lista istniejących już podstron.
3. Użytkownik dokonuje zmian poprzez: edytowanie nazw istniejących podstron lub usuwanie działających podstron.
4. Użytkownik naciska przycisk w oknie dialogowym:
 - a. Użytkownik naciska przycisk „Save”. W przypadku, gdy użytkownik naciśnie przycisk „Save” pozostawiając puste inputy, system wyświetla podpowiedź oraz podświetla input na czerwono. Dopóki wszystkie inputy nie zostaną wypełnione, system nie reaguje na przycisk „Save”.

- b. Użytkownik naciska przycisk „Cancel”. W tym przypadku system zamknie okno dialogowe bez dokonania żadnych zmian.
5. System zapisuje do bazy danych zastosowane zmiany i odświeża pasek nawigacyjny.

3.3.5. Konfiguracja elementów

3.3.5.1. [Warunki](#)

1. Użytkownik zalogowany.
2. W przypadku trybu edycji poprawnie wypełnione formularze.

3.3.5.2. [Ograniczenia](#)

Nie dotyczy gadżetu z mapą.

3.3.5.3. [Przebieg](#)

1. System wyświetla okno dialogowe odpowiednie dla danego elementu wybranego przez użytkownika.
 - a. Element „Text”:
 - i. System wyświetla edytor tekstu.
 - Nowy tekst: W edytorze tekstu wyświetlony napis „Click to edit”.
 - Edycja istniejącego tekstu: W edytorze wyświetlone wszystkie elementy edytowanego tekstu.
 - ii. Użytkownik dokonuje zmian w tekście z możliwością korzystania ze wszystkich funkcjonalności, jakie daje edytor tekstu.
 - b. Elementy „Accordion”, „Tab1”, „Tab2”:
 - i. System wyświetla formularz.
 - Nowy „Accordion”, „Tab1”, „Tab2”: System wyświetla formularz z 1 pustym input’em i przyciskami na dodanie i usunięcie input’a.
 - Edycja istniejącego „Accordion’a”, „Tab’a1”, „Tab’2”: System wyświetla odpowiednią ilość wypełnionych inputów do ilości zakładek w zakładkach i „accordion’ach”, wraz z przyciskiem na dodanie nowego inputa i przyciskami na usunięcie każdego z inputów. Edycja kart pozwala na zamianę inputów miejscami poprzez metodę Drag&Drop.
 - ii. Użytkownik dokonuje zmian poprzez zmianę tytułów zakładek, tworzenie nowych zakładek lub usuwanie ich.
 - c. Element „Picture”:
 - i. System wyświetla formularz dający użytkownikowi wybór pobierania zdjęcia: z pliku lub wpisując adres URL.
 - ii. Użytkownik wybiera opcje i dokonuje zmiany w formularzu.
2. Użytkownik naciska przycisk w oknie dialogowym:
 - a. Użytkownik naciska przycisk „Save”. W przypadku, gdy użytkownik naciśnie przycisk „Save” pozostawiając puste inputy, system wyświetla podpowiedź oraz podświetla input na czerwono. Dopóki wszystkie inputy nie zostaną wypełnione, system nie reaguje na przycisk „Save”.

- b. Użytkownik naciska przycisk „Delete” (tylko w trybie edycji). System usuwa dany element ze strony. W przypadku, gdy użytkownik będzie chciał usunąć logo (edycja zdjęć), lub tekst znajdujący się w „Tab’ach” i „Accordion’ach” (edytor tekstu), system wyświetli informacje o nieusuwalności danych elementów.
 - c. Użytkownik naciska przycisk „Cancel”. W tym przypadku system zamknie okno dialogowe bez dokonania żadnych zmian.
3. System zapisuje do bazy danych zastosowane zmiany i odświeża dany element.

3.3.6. Konfiguracja punktu na mapie

3.3.6.1. [Warunki](#)

1. Użytkownik zalogowany.
2. Formularz wyświetla prawidłowe dane o punkcie.

3.3.6.2. [Ograniczenia](#)

Możliwość dodania tylko 1 punktu.

3.3.6.3. [Przebieg](#)

1. System wyświetla formularz, w którym zostają wyświetlone informacje na temat punktu: państwo, stan / województwo / land, adres.
2. Użytkownik może napisać komentarz o danym punkcie.
3. Użytkownik naciska przycisk w oknie dialogowym.
 - a. Użytkownik naciska przycisk „Save”. Punkt zostaje dodany do mapy.
 - b. Użytkownik naciska przycisk „Remove”. Punkt zostaje usunięty z mapy.

3.3.7. Konfiguracja ustawień administratora

3.3.7.1. [Warunki](#)

1. Użytkownik zalogowany.

3.3.7.2. [Ograniczenia](#)

Brak.

3.3.7.3. [Przebieg](#)

1. Użytkownik naciska przycisk „Admin”.
2. System wyświetla okno dialogowe z formularzem do zmiany hasła użytkownika.
3. Użytkownik wpisuje dane do formularza.
4. Użytkownik naciska przycisk w oknie dialogowym.
 - a. Użytkownik naciska przycisk „Save”. Hasło zostaje zmienione i zapisane do bazy danych. W przypadku gdy użytkownik nie wpisze nic do formularza lub wpisze dane nieprawidłowo, system wyświetli komunikat o popełnionym błędzie.
 - b. Użytkownik naciska przycisk „Cancel”. Operacja zmiany hasła zostaje anulowana.

3.3.8. Logowanie

3.3.8.1. [Warunki](#)

1. Warunek początkowy; Użytkownik niezalogowany.

2. Warunek końcowy; Użytkownik zalogowany.

3.3.8.2. [Ograniczenia](#)

Brak.

3.3.8.3. [Przebieg](#)

1. Administrator znajduje link do podstrony logowania i wysyła żądanie zalogowania.
2. Administrator wpisuje login i hasło. W przypadku błędu logowanie jest przerywane.

3.3.9. Przeglądanie wersji desktopowej

3.3.9.1. [Warunki](#)

Brak.

3.3.9.2. [Ograniczenia](#)

Przeglądarka musi obsługiwać JavaScript.

3.3.9.3. [Przebieg](#)

1. System wyświetla stworzoną stronę WWW.
2. Użytkownik przegląda stronę internetową.

3.3.10. Przeglądanie wersji mobilnej

3.3.10.1. [Warunki](#)

Brak.

3.3.10.2. [Ograniczenia](#)

Przeglądarka w urządzeniu musi obsługiwać JavaScript.

3.3.10.3. [Przebieg](#)

1. System wyświetla stronę WWW w wersji przystosowanej do urządzenia.
2. Użytkownik przegląda stronę internetową.

3.3.11. Zmiana kolejności elementów na stronie

3.3.11.1. [Warunki](#)

1. Użytkownik zalogowany.
2. Dodane przynajmniej 2 elementy.

3.3.11.2. [Ograniczenia](#)

Możliwa tylko modyfikacja w pionie.

3.3.11.3. [Przebieg](#)

1. Użytkownik łapie dany element i metodą Drag&Drop przesuwa element na wybrane przez siebie miejsce.
2. System zapisuje zmianę kolejności elementów w bazie danych i odświeża zawartość content'u.

3.3.12. Zmiana koloru

3.3.12.1. [Warunki](#)

1. Użytkownik zalogowany.
2. Pokazany poprawny kolor w label'u.

3.3.12.2. [Ograniczenia](#)

Brak.

3.3.12.3. [Przebieg](#)

1. Użytkownik naciska przycisk „Color”.
2. System wyświetla okno dialogowe, w którym znajdują się slidery RGB oraz label informujący o kolorystyce aplikacji. Suwaki mają wartość odpowiadającą współrzędnych barw R (red z ang. czerwony), G (green z ang. zielony) oraz B (blue z ang. niebieski).
3. Użytkownik przesuwa slidery według własnego uznania, szukając odpowiedniego koloru.
4. Wraz z przesunięciem suwaka system zmienia kolor tła label’u odpowiadającym wartościom slider’ów.
5. Użytkownik naciska przycisk w oknie dialogowym:
 - a. Użytkownik naciska przycisk „Save”.
 - b. Użytkownik naciska przycisk „Cancel”. W tym przypadku system zamknie okno dialogowe bez dokonania żadnych zmian.
6. System zapisuje nowe wartości RGB do bazy danych i koloruje wszystkie elementy znajdujące się na stronie.

4. Architektura systemu

4.1. Architektura

Motorem napędowym aplikacji jest technologia ASP.NET, a konkretnie WebForms. Każdy WebForm składa się z dwóch plików. Pierwszy plik z rozszerzeniem .ASPX odpowiada za wygląd strony oraz za to, co się na niej mieści. Natomiast drugi plik o tej samej nazwie, w naszym wypadku z rozszerzeniem .ASPX.CS (ponieważ korzystamy z języka C#), odpowiada za logikę strony oraz jej przetwarzanie.^[1]

Architektura aplikacji, podobnie jak wielu innych CMS'ów, opiera się na jednej stronie głównej, która posiada specjalnie wydzielone kontenery, wypełniane danymi z bazy.

Przykładowy, kod korzystający z kontenerów:

```
<div class="row">
  <div class="twelve columns">
    <nav class="pretty navbar clearfix" id="prettynav">
      <h1 class="logo">
        <a class="notDeleteable tooltip" href="#" id="logo">
          <%= PageLogo%>
        </a>
      </h1>
      <a id="tog" href="#" class="toggle" data-for="#prettynav"> ul">
        
      </a>
      <ul id="menuNav">
        <%= PagesCollection%>
      </ul>
    </nav>
  </div>
</div>

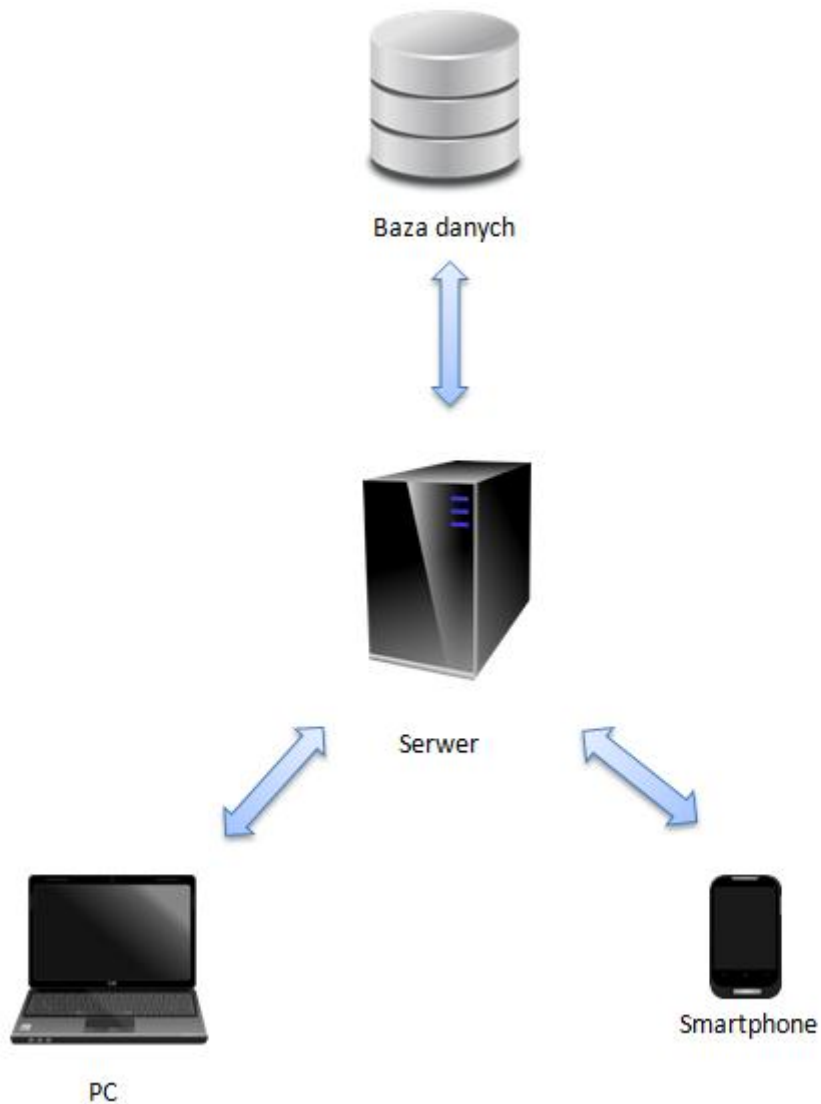
<div class="row">
  <div class="twelve columns" id="content">
    <ul id="contentUL">
      <%= PageContent %>
    </ul>
  </div>
</div>

<div id="currentPage" style="display: none"><%= PageId%></div>
```

W tym przypadku, kontenery te to „PageLogo”, „PagesCollection”, „PageContent” oraz „PageId”. Odpowiadają one odpowiednio za renderowanie logo strony, kolekcji przycisków do poszczególnych podstron, zawartości podstrony oraz do przetrzymywania identyfikatora danej podstrony.

Wymienione kontenery, przy ładowaniu strony zamieniane są na odpowiednie znaczniki HTML pobrane z bazy danych. Dzięki takiemu podejściu, w celu stworzenia nowej podstrony, nie musimy tworzyć osobnego pliku, lecz jedynie nowy rekord w bazie danych, który wypełni odpowiednie miejsca na stronie głównej.

DotNetCMS



Rysunek 2: Schemat architektury systemu. (Źródło: opracowanie własne, grafika pobrana z <http://openclipart.org>)

4.2. Dane

Do stworzenia bazy danych została wykorzystana technologia ADO.NET Entity Framework. Skorzystano z podejścia Model-First, tzn. najpierw stworzono model danych, a następnie na jego podstawie wygenerowano bazę danych. Jako kontener na przechowywanie danych, wybrano Microsoft SQL Server Compact Edition.

Jest to relacyjna baza danych przeznaczona dla aplikacji na urządzenia mobilne jak i desktopowe.

Bazę tę wybrano dlatego, że jest to baza danych w pliku, która do swojego działania nie potrzebuje serwera bazodanowego. Pozwala to na pominięcie procesu konfiguracji przez użytkownika. Z tego powodu jej możliwości są ograniczone w porównaniu do bazy danych umieszczonej na serwerze, jednak dla potrzeb naszej aplikacji jej funkcjonalność jest wystarczająca. Z uwagi na wymienione korzyści i prostotę użycia jest ona wyborem wręcz idealnym.

4.3. Komunikacja JavaScript – ASP.NET^[1]

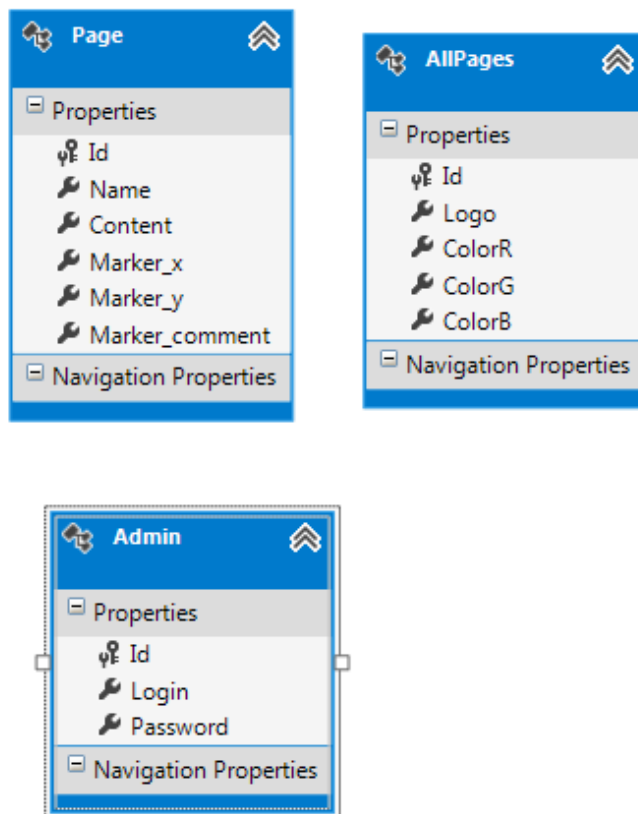
Jako, że aplikacja korzysta zarówno z JavaScript'u po stronie klienta jak i ASP.NET po stronie serwera, należało znaleźć sposób, aby połączyć ze sobą te dwie technologie. Z pomocą przychodzą WebServices czyli usługi internetowe. Web serwisy to tak naprawdę klasy, które udostępniają na zewnątrz swoje metody poprzez protokół http. Poprzez usługę internetową udostępniono następujące metody:

- ChangeLogin – służy do zmiany loginu administratora
- ChangePassword – służy do zmiany hasła administratora
- CreatePage – służy do stworzenia podstrony
- DeletePage – służy do usunięcia podstrony
- AddMarker – służy do zapisania dodanego do mapy punktu
- GetMarker – pobieranie punktu w celu dodania go do mapy
- DeleteMarker – usuwanie punktu
- GetColorB – pobieranie składnika niebieskiego z modelu barw RGB
- GetColorG - pobieranie składnika zielonego z modelu barw RGB
- GetColorR – pobieranie składnika czerwonego z modelu barw RGB
- GetContent – pobieranie zawartości podstrony
- GetLastPage – pobieranie ostatnio dodanej strony
- GetLogo – pobieranie logo
- LogOut – wylogowanie administratora
- SaveContent – zapisanie treści podstrony
- SaveLogo – zapisanie logo
- UpdateColor – aktualizacja koloru
- UpdateLogo – aktualizacja logo
- UpdatePage – aktualizacja zawartości strony

5. Baza danych

5.1. Struktura bazy danych

Zostały stworzone dwie bazy danych - jedna do przechowywania konfiguracji wyglądu strony, natomiast druga do przechowywania loginu i hasła administratora.



Rysunek 3: Diagram modelu bazy danych(Źródło: opracowanie własne).

5.2. Opis tablic bazy danych i ich atrybutów.

AllPagesSet	Elementy, które znajdują się na każdej podstronie. W tej tabeli występuje przez cały czas tylko 1 rekord. Zmieniają się tylko wartości atrybutów	
ID	INT	Identyfikator.
Logo	NAVCHAR(4000)	Znacznik wraz atrybutem src, gdzie wartość atrybutu odpowiada ścieżce dostępu do log.
colorR	INT	Wartość barwy czerwonej modelu RGB.
colorG	INT	Wartość barwy zielonej modelu RGB.
colorB	INT	Wartość barwy niebieskiej modelu RGB.

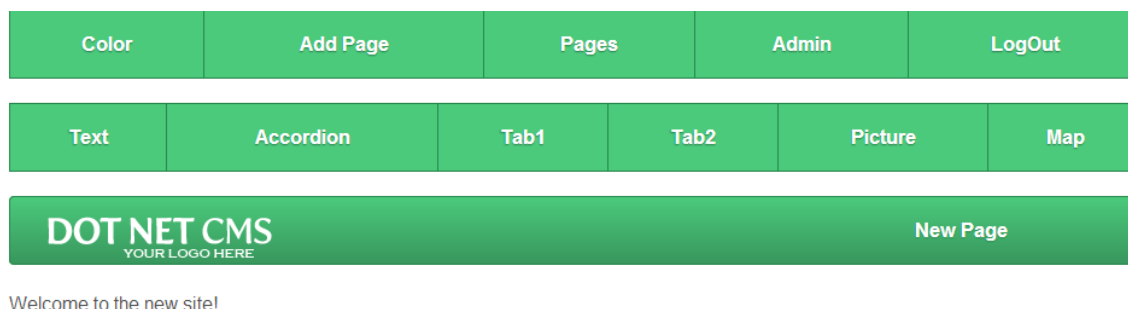
PageSet	Wszystkie elementy, które zostały dodane przez redaktora strony, w zależności od podstrony.	
ID	INT	Identyfikator.
Name	NAVCHAR(4000)	Nazwa podstrony.
Content	nText	Kod HTML, w którym znajdują się wszystkie elementy danej podstrony oraz ich konfiguracje.
Marker_x	NAVCHAR(4000)	Szerokość geograficzna punktu na mapie wyrażona w liczbie zmiennoprzecinkowej.
Marker_y	NAVCHAR(4000)	Długość geograficzna punktu na mapie wyrażona w liczbie zmiennoprzecinkowej.
Marker_comment	NAVCHAR(4000)	Komentarz dodany do punktu na mapie

AdminSet	Użytkownicy, którzy mają dostęp do pełnej funkcjonalności aplikacji – redaktorzy stron.	
ID	INT	Identyfikator.
Login	NAVCHAR(4000)	Login administratora.
Password	NAVCHAR(4000)	Hasło administratora.

6. Opis implementacji

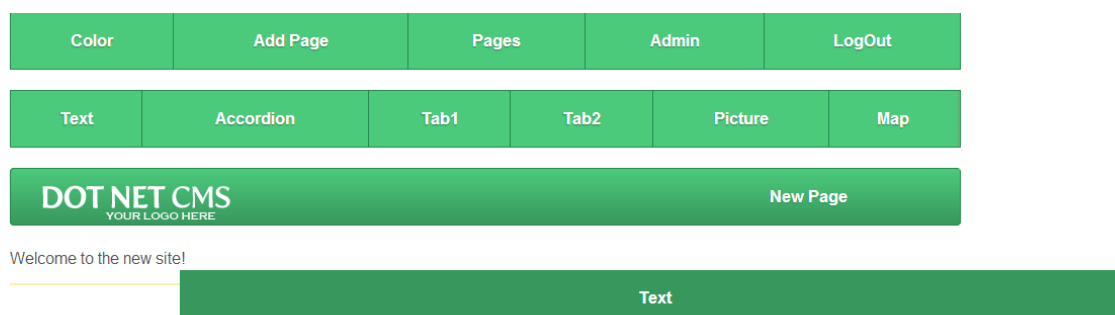
6.1. Interfejs i działanie systemu

Po zalogowaniu wyświetli nam się strona administracyjna, która udostępnia pełną funkcjonalność projektu. W tej chwili stworzona jest tam jedna podstrona „New page” wraz z tekstem „Welcome to the new site!”.



Rysunek 4: Okno strony początkowej (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)

Podstawową funkcją programu jest dodawanie elementów. Należy wtedy wybrać z dolnego toolbox'a gadżet, który chce się wrzucić na stronę i przeciągnąć w wybrane przez siebie miejsce. System podświetli wtedy dostępne miejsca.

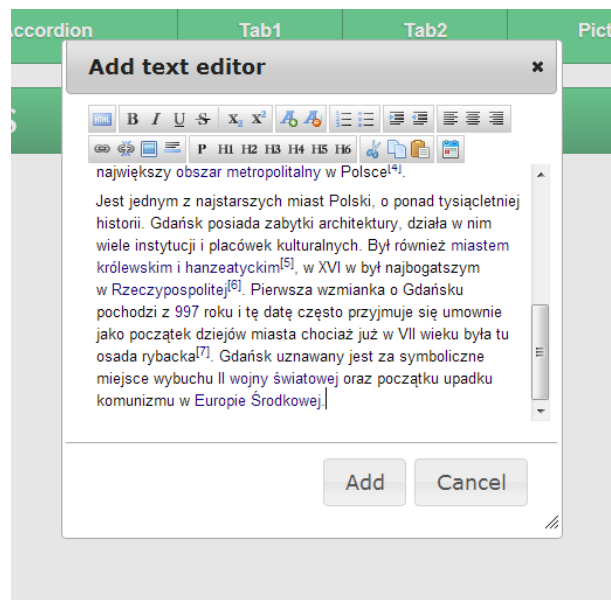


Rysunek 5: Dodawanie elementu do strony (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)

W tym przypadku został wybrany nowy tekst. Wraz opuszczeniem elementu otworzy się okno dialogowe, w które można załączyć treść według uznania. Okno ma wiele możliwości:

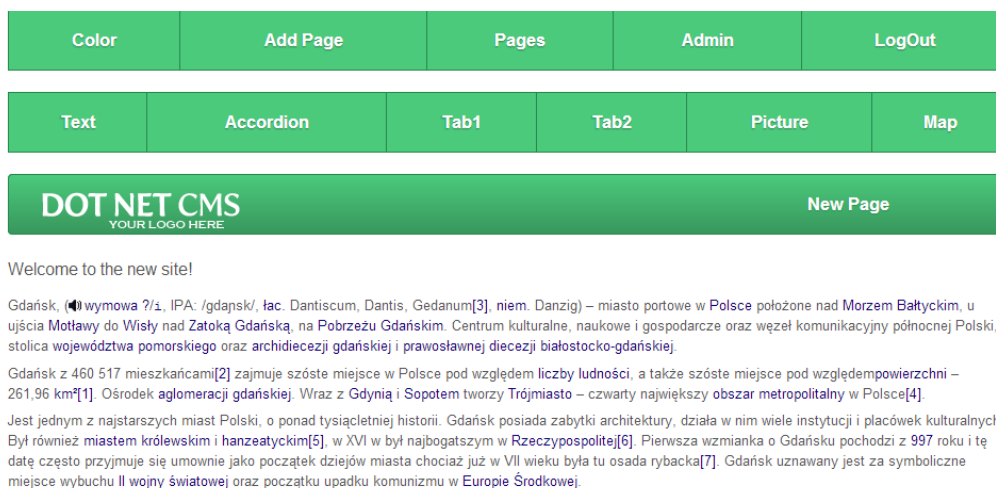
- Wyświetlenie źródła HTML naszej treści.
- Opcje pogrubienia, kursywy, podkreślenia i skreślenia.
- Zwiększanie i zmniejszanie czcionki.
- Numeracja, punktowanie.
- Paragrafy.
- Dodawanie i usuwanie linków, zdjęć i linii poziomej.
- Nagłówki.
- Wycinanie, kopiowanie, wklejanie.
- Dodawanie dzisiejszej daty.

Na początku w edytorze wyświetlony jest napis „Click to edit”, który przypomina o tym, że później można edytować tekst naciskając na obszar tekstu myszką. Dla celów prezentacyjnych umieścimy część artykułu z Wikipedii o Gdańsku, który zawiera w sobie link.



Rysunek 6: Okno dialogowe edytora HTML (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)

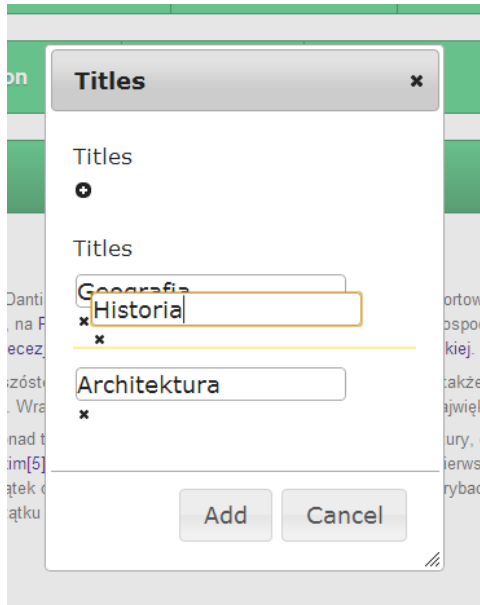
DotNetCMS



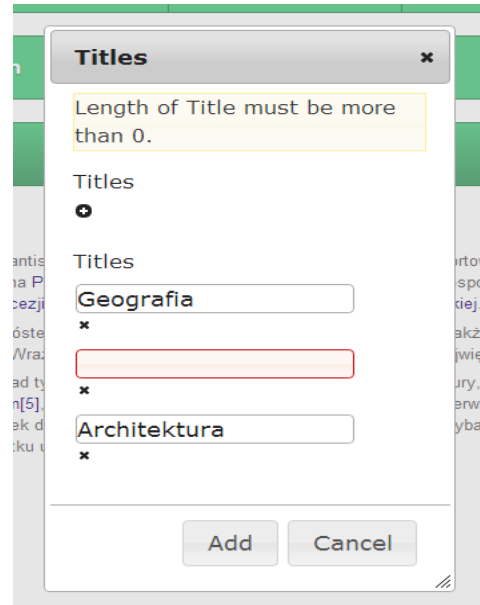
Rysunek 7: Dodany tekst do strony (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)

Po dodaniu tekstu, zawartość wyświetliła się na naszej stronie tak, jak widać to na rys 6. Oczywiście treść można zmieniać według własnych upodobań. Po naciśnięciu na tekst znowu otworzy się okno dialogowe z edytowanym tekstem, w którym można dokonać zmian.

Kolejne elementy, które można dodawać to „Accordion”, „Tab1”, „Tab2”. Są to dynamicznie tworzone zakładki różniące się przede wszystkim wyglądem. Gdy dodamy jeden z tych elementów, otworzy się okno dialogowe, które będzie od nas wymagało wpisania tytułów zakładek. Ilość zakładek jest zmienna. Użytkownik w prosty sposób może dodawać lub usuwać tytuły naciskając na odpowiednie przyciski. Dodatkowo kolejność zakładek może być zmieniana. Wystarczy złapać wybrany input i przeciągnąć go w odpowiednie miejsce na liście metodą Drag&Drop. Gdy redaktor strony zapomni wpisać którykolwiek z tytułów, system wyświetli stosowną informację, nie dając tym samym możliwości dodania nowych zakładek.



Rysunek 8 (po lewej): Zmiana kolejności zakładek. (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)



Rysunek 9 (po prawej): Walidacja danych wpisanych do formularza (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)

Dodane zakładki mają w sobie tekst „Click to edit.”. Nie jest to nic innego, jak edytowalny tekst, tego samego typu co ten, który był dodawany na początku prezentacji. Tak więc po naciśnięciu na obszar tekstu otworzy nam się okno dialogowe, w którym można zmienić tekst. Tym razem również zostanie tam umieszczony tekst o Gdańsku, pobrany z Wikipedii, ale wraz ze zdjęciem. Teksty w zakładkach są nieusuwalne. System rozróżnia tekst, który znajduje się w zakładce i poza nią.



Welcome to the new site!

Gdańsk, (wymowa [?]/i/, IPA: /gdąnsk/, łac. Dantiscum, Dantis, Gedanum[3], niem. Danzig) – miasto portowe w Polsce położone nad Morzem Bałtyckim, u ujścia Motławy do Wisły nad Zatoką Gdańską, na Pobrzeżu Gdańskim. Centrum kulturalne, naukowe i gospodarcze oraz węzeł komunikacyjny północnej Polski, stolica województwa pomorskiego oraz archidiecezji gdańskiej i prawosławnej diecezji białostocko-gdańskiej.

Gdańsk z 460 517 mieszkańcami[2] zajmuje szóste miejsce w Polsce pod względem liczby ludności, a także szóste miejsce pod względem powierzchni – 261,96 km²[1]. Ośrodek aglomeracji gdańskiej. Wraz z Gdynią i Sopotem tworzy Trójmiasto – czwarty największy obszar metropolitalny w Polsce[4].

Jest jednym z najstarszych miast Polski, o ponad tysiącletniej historii. Gdańsk posiada zabytki architektury, działa w nim wiele instytucji i placówek kulturalnych. Był również miastem królewskim i hanzeatyckim[5], w XVI w. był najbogatszym w Rzeczypospolitej[6]. Pierwsza wzmianka o Gdańsku pochodzi z 997 roku i tę datę często przyjmuje się umownie jako początek dziejów miasta chociaż już w VII wieku była tu osada rybacka[7]. Gdańsk uznawany jest za symboliczne miejsce wybuchu II wojny światowej oraz początku upadku komunizmu w Europie Środkowej.



Średniowiecze

(..)

W 1410, po klęsce zakonu krzyżackiego w bitwie pod Grunwaldem, biedota miejska wszczęła rozruchy, w których wymordowano gości Zakonu izacięznych, których rannych przywieziono po bitwie[12]. Gdańska rada postanowiła uznać władzę Władysława Jagiełły, za co miasto uzyskało liczne przywileje. Rok później Jagiełło w wyniku traktatu w Toruniu uwolnił Gdańsk od złożonej mu przysięgi. Gdańsk dotknęły represje ze strony Krzyżaków.

14 marca 1440 Gdańsk przystąpił do Związku Pruskiego. 11 lutego 1454, po 146 latach, zakończył się okres panowania Krzyżaków w Gdańsku. 6 marca tego roku król Kazimierz IV Jagiellończyk na wniosek poselstwa Związku Pruskiego, wcielił Gdańsk do Polski, udzielając mu jednocześnie przywileju bicia własnej monety. Gdańsk został zwolniony z prawa nabrzeżnego, a także dopuszczono przedstawicieli ziem pruskich do elekcji króla Polski. Gdańsk przystąpił do wojny trzynastoletniej w 1455; gdańszczanie złożyli hołd Kazimierzowi Jagiellończykowi, a 25 maja 1457 miasto otrzymało Wielki Przywilej, zapewniający swobodny przywóz towarów Wisłą z Polski, Litwy i Rusi bez konieczności kontroli oraz inne przywileje, które miały wynagrodzić miastu wkład w wojnę. Zawarcie w 1466 pokoju



Rysunek 10: Dodany tekst do zakładki razem ze zdjęciem (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)

Dodane zakładki również można edytować. Za każdym razem, gdy najedzie się myszką na umieszczoną na stronie zakładkę wyświetli się informacja, w jaki sposób można edytować element. W tym przypadku jest to „Double click to edit tab”, co oznacza, że trzeba nacisnąć podwójnie na zakładkę, żeby ją z edytować. Inaczej niż w odniesieniu do tekstu, który edytowało się poprzez 1 kliknięcie. Każdy umieszczony element po za treścią tekstową wyświetla instrukcje edycji.

datę często przyjmuje się umownie jako początek dziejów miasta chociaż już w VII wieku była tu osada rybacka[7]. Gdańsk uznawany jest za symboliczne miejsce wybuchu II wojny światowej oraz początku upadku komunizmu w Europie Środkowej.



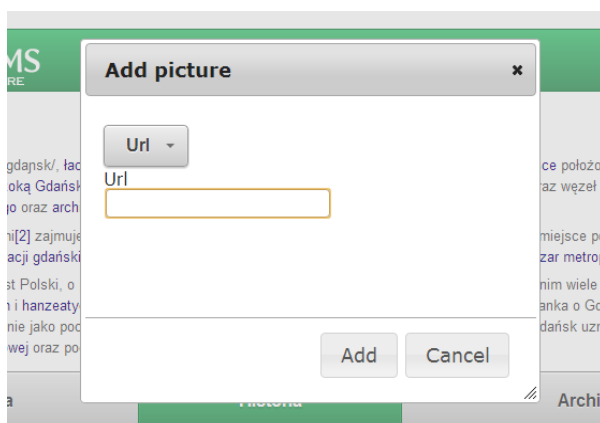
Click to change content / Double click to edit Tab.

W 1410, po klęsce zakonu krzyżackiego w bitwie pod Grunwaldem, biedota miejska wszczęła rozruchy, w których wymordowano gości Zakonu izacięznych, których rannych przywieziono po bitwie[12]. Gdańska rada postanowiła uznać władzę Władysława Jagiełły, za co miasto uzyskało liczne przywileje. Rok później Jagiełło w wyniku traktatu w Toruniu uwolnił Gdańsk od złożonej mu przysięgi. Gdańsk dotknęły represje ze strony Krzyżaków.

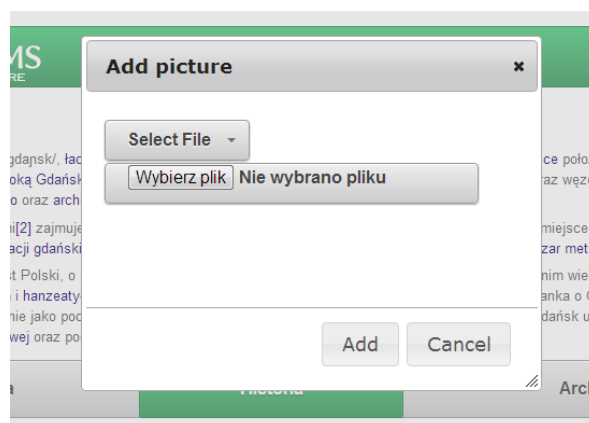


Rysunek 11: Podpowiedź (Tooltip) jak z edytować dany element. (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)

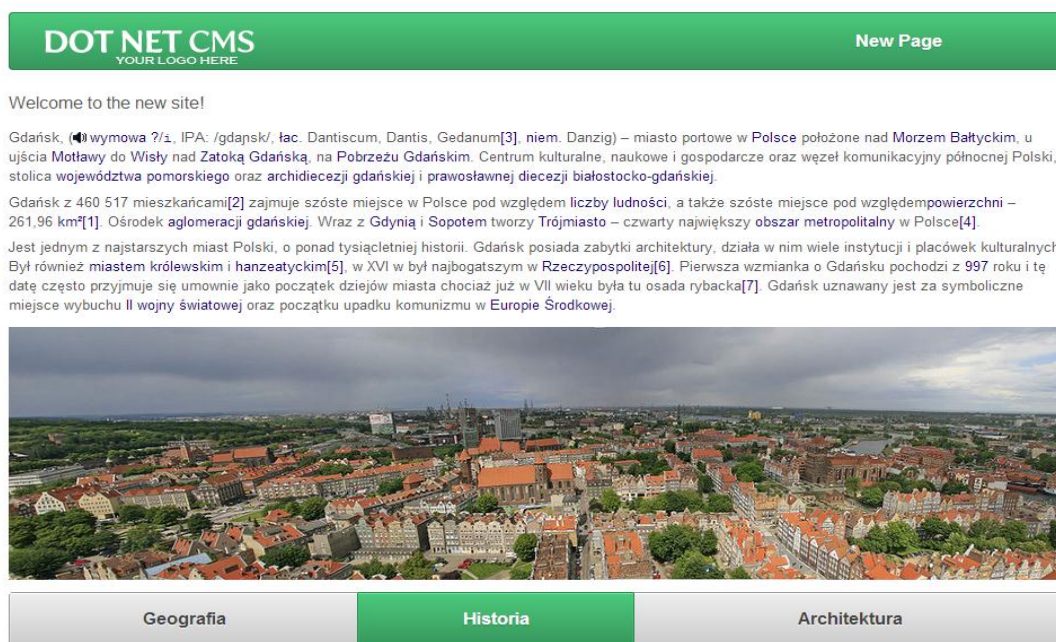
Nierozłącznym elementem stron internetowych są pliki graficzne. Projekt uwzględnia dodawanie samych zdjęć. Po wybraniu z dolnego toolbox'a elementu „Picture” i umieszczeniu go na stronie, system wyświetli dialog, który pozwoli na dodanie strony. Administrator strony ma do wyboru dwie opcje: dodanie umieszczonej już w Internecie grafiki podając jej adres URL lub pobierając obraz z pliku. Niemożliwe jest dodanie zdjęcia o pustym adresie, lecz dodanie o złym już tak. W momencie, gdy wybrano nieodpowiedni obraz lub wpisano zły adres można z edytować ilustrację, naciskając podwójnie na element i wpisać dobry adres lub wybrać odpowiedni plik. Dla demonstracji tej funkcjonalności dodamy obrazek wpisując adres obrazka z wybranego artykułu Wikipedii o Gdańsku.



Rysunek 12 (po lewej): Dodawanie zdjęć: URL (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)



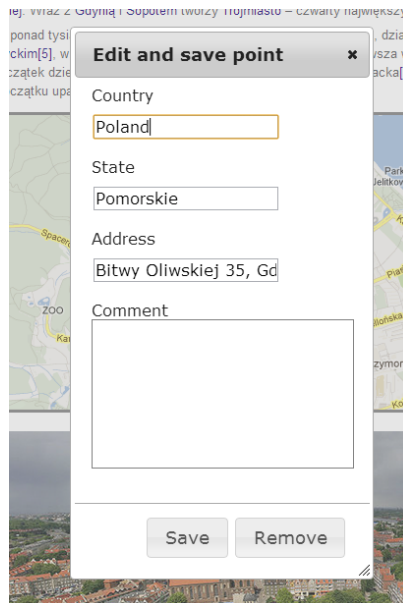
Rysunek 13 (po prawej): Dodawanie zdjęć pobierając z pliku (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)



Rysunek 14: Dodane zdjęcie do strony (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)

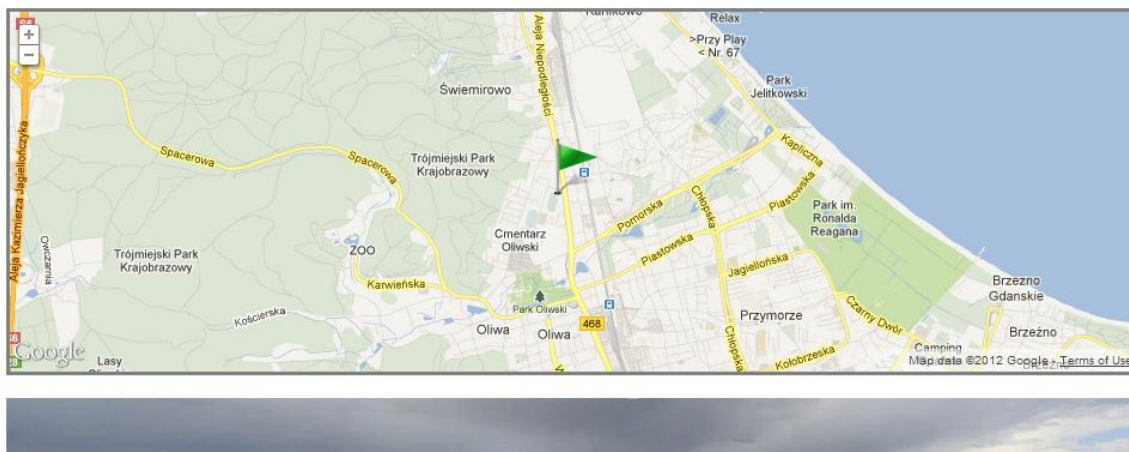
Podobnie jest jeżeli chodzi o zakładki - w momencie najechania na zdjęcie kursorem, pojawi się podpowiedź informująca o tym, jak wejść w tryb edycji zdjęcia.

Kolejny element, jaki można dodać, to gadżet, który zawiera w sobie mapę udostępnioną przez firmę Google. Redaktor strony może umieścić punkt (znacznik) w dowolnym miejscu na mapie. Zastosowanie tej funkcjonalności może być różne. Autorzy projektu, dodali tę funkcję z myślą o wskazaniu miejsc siedzib firm, o których może być strona. W tym momencie dodamy punkt w pewnym miejscu Gdańska.



Rysunek 15: Dodany punktu do mapy (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)

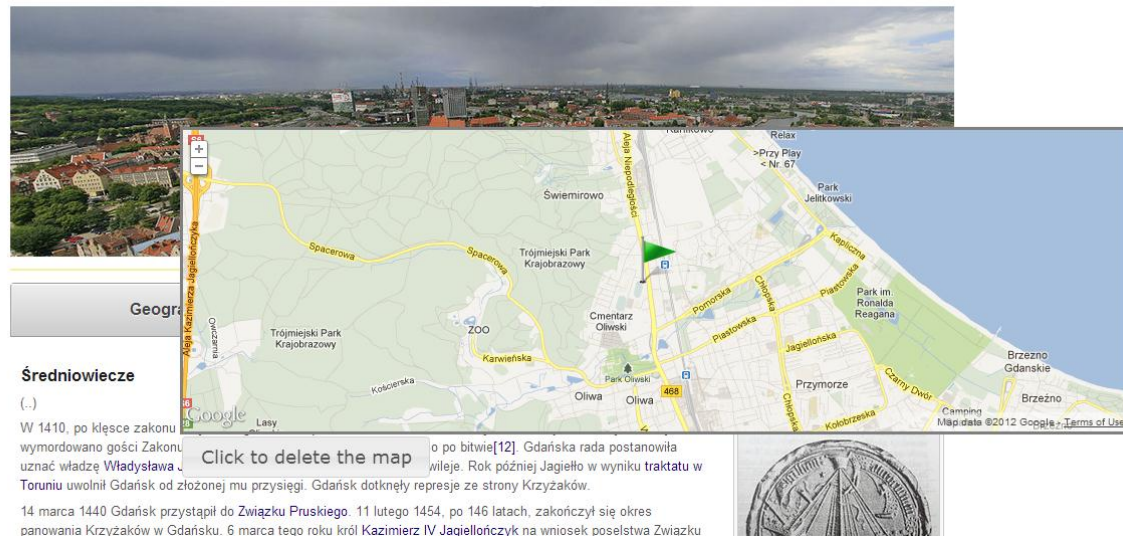
Ważną częścią projektu jest umieszczenie punktu na mapie. W tym celu użyto biblioteki Google Maps API. Punkt umieszczony jest na symbolizującym miejsce wybuchu II wojny światowej oraz początku upadku komunizmu w Europie Środkowej.



Rysunek 16: Dodana mapa z punktem (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)

W momencie, gdy administrator strony uzna, że kolejność elementów jest nieodpowiednia, może w każdej chwili zmienić porządek komponentów na stronie. Wystarczy złapać kursorem dany element i przeciągnąć go, metodą Drag&Drop, w wybrane przez siebie miejsce.

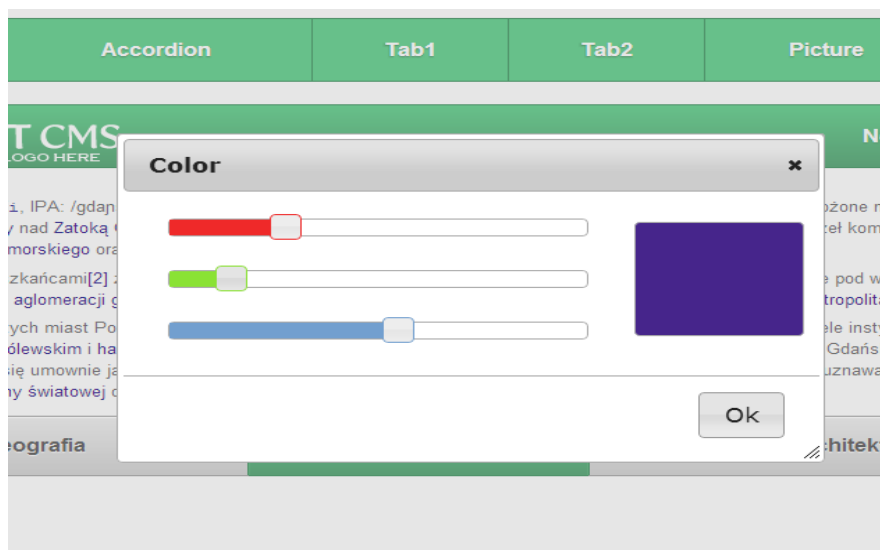
Długość historii miasta Gdańsk, w tym w tym najdłuższym w Rzeczypospolitej, pierwsza wzmianka o Gdańsku pochodzi z 997 roku i tę datę często przyjmuje się umownie jako początek dziejów miasta chociaż już w VII wieku była tu osada rybacka[7]. Gdańsk uznawany jest za symboliczne miejsce wybuchu II wojny światowej oraz początku upadku komunizmu w Europie Środkowej.



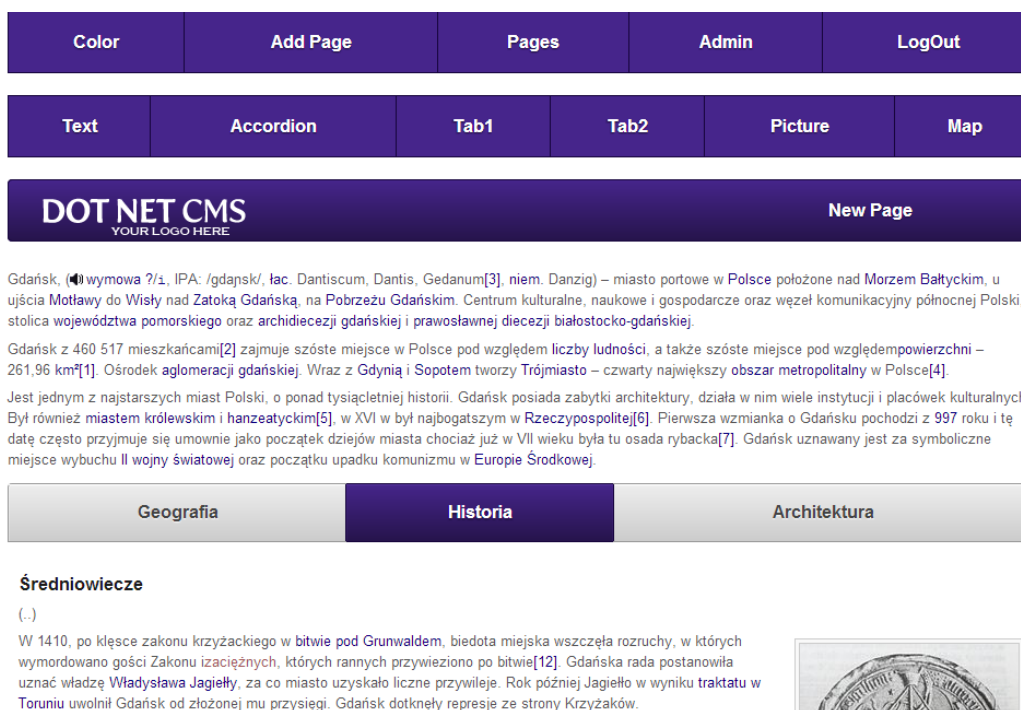
Rysunek 17: Przenoszenie elementów (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)

W tej chwili pokazowa strona posiada już kilka elementów. W każdej sekundzie zarządca strony może uznać, że część treści na stronie jest już niepotrzebna. By usunąć element trzeba wejść w tryb edycji. Jednym z dostępnych przycisków w oknie konfiguracji jest opcja „Delete” – czyli usunąć, który kasuje dany element ze strony. Jedynym wyjątkiem jest gadżet „Map”. Gdy najedzie się myszką na ten komponent, pojawi się przycisk „Click to delete map”, który usuwa mapę ze strony. W przypadku strony pokazowej został pozostawiony tekst „Welcome”, który automatycznie został utworzony na początku istnienia strony. Napis „Welcome” jest zwykłym tekstem, więc należy na niego kliknąć, po czym otworzy się okno dialogowe stworzone do edycji.

Podstawową funkcją systemu jest zmiana kolorystyki. Redaktor strony w prosty sposób może zmienić barwy wybranych elementów na stronie. Aby zmienić kolor elementów należy nacisnąć przycisk „Color” znajdujący się w górnym toolbox’ie. Otworzy się wówczas okno dialogowe z 3 sliderami odpowiadającymi wartości barw modelu RGB. Administrator przysuwając suwaki zmienia kolorystykę istniejących elementów na stronie.



Rysunek 18: Wybór kolorystyki strony (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)



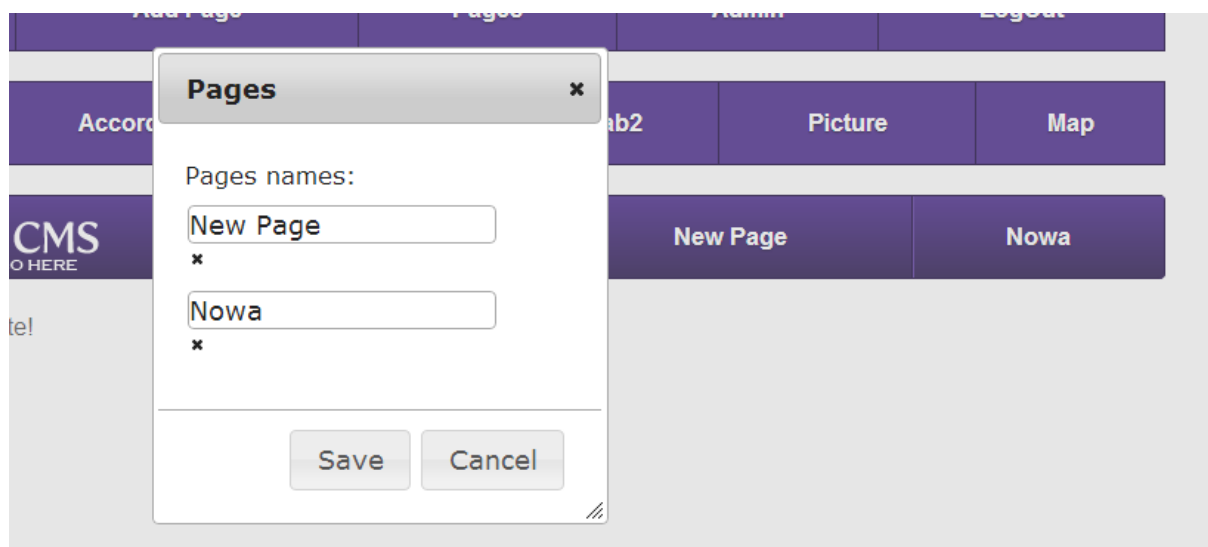
Rysunek 19: Zmiana kolorystyki elementów na stronie (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)

Tworzenie serwisów internetowych wiąże się z tworzeniem kolejnych podstron. Aplikacja umożliwia tę opcję po naciśnięciu przycisku „Add Page”. Otworzy się okno dialogowe, w którym należy wpisać nazwę nowej podstrony. System od razu wyświetli link do nowej podstrony w pasku nawigacyjnym.



Rysunek 20: Dodana nowa podstrona (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)

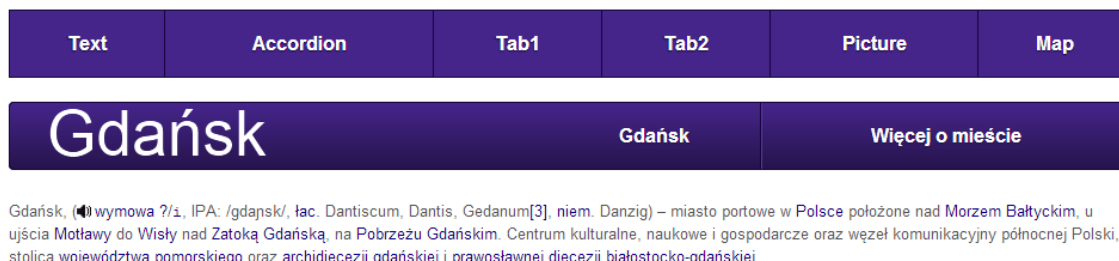
Może wydarzyć się taka sytuacja, w której redaktor strony dodał stronę przez przypadek lub źle wpisał nazwę podstrony. Aby zarządzać utworzonymi podstronami należy nacisnąć przycisk „Pages”. Aplikacja wyświetli listę ze wszystkimi podstronami znajdującymi się w serwisie. Aby zmienić tytuł strony należy wpisać odpowiednią nazwę do wybranego inputa. Usunięcie polega na naciśnięciu ikonki zamykającej (czarnego krzyżyka) znajdującej się pod inputami z nazwami podstron. Aby anulować zmiany wystarczy nacisnąć przycisk „Cancel”, znajdujący się w oknie dialogowym.



Rysunek 21: Edycja nazw istniejących podstron (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)

Dla celów prezentacyjnych, nazwy podstron zostaną ustawione na „Gdańsk”, „Więcej o mieście”.

W niemalże każdej stronie internetowej występuje logo. Aplikacja DotNetCms pozwala na wstawienie własnego logo. Wystarczy zlokalizować oraz edytować je poprzez podwójne kliknięcie. Należy pamiętać, że logo ma ograniczoną wysokość, dlatego należy stworzyć taką grafikę, która będzie pasowała do paska nawigacyjnego. Zalecane jest stworzenie obrazka z przezroczystym tłem. Dla celów prezentacyjnych zostało utworzone logo pasujące do reszty strony pokazowej – napis Gdańsk z przezroczystym tłem.



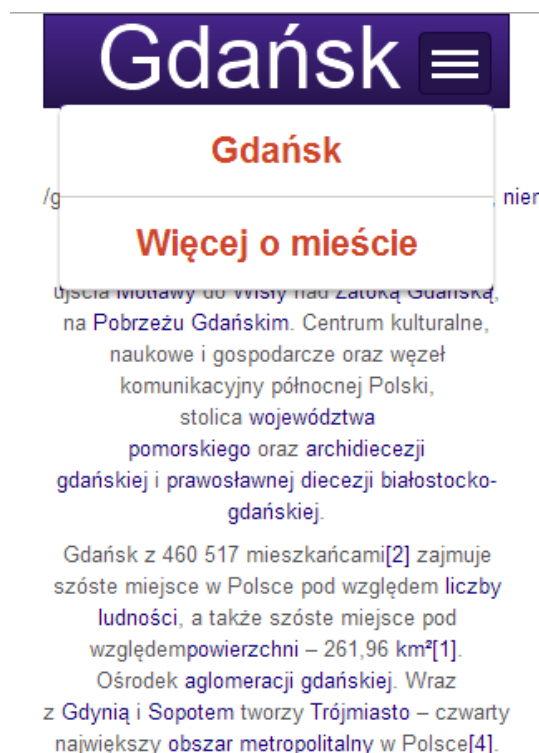
Rysunek 22: Dodana nowa logo do strony (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)

W celu ułatwienia pracy administratora i zabezpieczenia dostępu do strony administracyjnej, redaktor powinien zmienić hasło. Należy wtedy nacisnąć przycisk „Admin”, który wyświetla okno dialogowe pozwalające na zmianę hasła. Po każdej edycji strony administrator powinien pamiętać o wylogowaniu się poprzez naciśnięcie przycisku „Log out”.

Teraz można podziwiać efekty pracy administratora, zarówno w wersji desktopowej jak i na mobilnej stronie.



Rysunek 23: Wersja desktopowa (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)



Rysunek 24: Wersja mobilna z otwartą rozwijaną listą (podstrony) (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)

6.2. Logowanie do systemu DotNetCms

Aby włączyć mechanizm uwierzytelnienia użytkownika, należy zdefiniować w pliku Web.config tag <authentication>. Należy również ustawić parametr mode="Forms", w celu uaktywnienia uwierzytelniania opartego o formularze.^[1]

Aby zablokować dostęp do katalogu, należy zdefiniować tag <authorization> w pliku Web.config, znajdujący się w danym katalogu. Do zdefiniowania reguły dostępu do katalogu, jedynie dla zalogowanego użytkownika, posłużono się tagiem <deny> z parametrem users="?".

Do zalogowania użytkownika użyto gotowego komponentu .Net Framework 4.0 o nazwie „Login”. Kontrolka ta posiada zdarzenie „Authenticate”, które sprawdza, czy wprowadzony przez użytkownika ciąg znaków (login i hasło) zgadza się z odpowiednim wpisem w bazie danych. Jeśli dane są identyczne użytkownik zostaje zalogowany, a w przeciwnym wypadku wyświetlany jest stosowny komunikat. Zdarzenie „Authenticate” wywoływane jest za każdym razem, gdy użytkownik kliknie przycisk „Log In”.


6.3. Zmiana kolorystyki strony

Elementarną funkcją systemu jest zmiana kolorystyki elementów znajdujących się na stronie. Odbyna się to przy zastosowaniu funkcji napisanej po stronie klienta z wykorzystaniem stylów CSS z Gumby Framework oraz Web serwisu, dzięki któremu skrypt komunikuje się z bazą danych.

Na początku, w momencie gdy użytkownik naciśnie przycisk „Color” uruchamia się okno dialogowe, które wyświetla trzy suwaki. Wartość sliderów odpowiada wartości barw modelu RGB. Pobierane są one z bazy danych z tabeli AllPagesSet (atrybuty: colorR, colorG, colorB). Po dokonaniu zmian przez redaktora strony uruchamia się funkcja JavaScript `changeColor()` znajdująca się w pliku `color.js`.

Funkcja wykorzystuje trzy parametry `red`, `green`, `blue` odpowiadające wartościom barw - kolejno czerwonej, niebieskiej i zielonej. Zmienne te zostają przetwarzane w barwę postaci szesnastkowej (na przykład kolor biały posiada oznaczenie: `#ffffff`) przy pomocy funkcji `hexFromRGB()`. Na podstawie tej barwy, która staje się głównym zabarwieniem kolorystyki strony, tworzone są kolory podrzędne wykorzystywane w takich elementach jak cień linków, obramowania elementów czy gradientu i tym podobne. W zależności od tego czy główny kolor wybrany wcześniej przez administratora strony jest ciemny czy jasny, dobierane są barwy drugorzędne.

Następnie sprawdzane jest na jakiej przeglądarce wyświetlona jest strona. Elementy były kolorowane w taki sposób aby każda przeglądarka mogła interpretować, według swoich zasad, zmianę stylów CSS. Kontrola została ograniczona do:

- Google Chrome 
- Mozilla Firefox 
- Opera 
- Pozostałe przeglądarki (Internet Explorer itd.).  ...

Pierwsze 2 przeglądarki były, podczas realizacji projektu, najbardziej popularnymi programami obsługującymi strony internetowe na świecie. W urządzeniach przenośnych bardzo często używana jest mobilna wersja Operry.

Kolorowanie elementów na stronie wykonuje się podczas każdej zmiany kolorystyki oraz przy każdym załadowaniu strony.

Zmiana koloru wymagała rozwinięcia gotowego rozwiązania jakim jest Gumby Grid Responsive Framework. Poza edycją pliku ui.css, która polegała na usunięciu niektórych funkcji, które nie współgrały z metodą kolorującą, należało również zredagować skrypt gumby.js. W pliku tym znajduje się logika struktury Gumby. W celu integracji reakcji na zmianę aktualnej zakładki „Tab2” należało dodać funkcję kolorującą zmieniane zakładki na barwę przezroczystą - zakładki nieaktywne, oraz barwę odpowiadającą wyborowi redaktora strony – zakładki aktywne.

6.4. Dodawanie i edytowanie podstron.

Po stronie klienta (JavaScript):

Dodawanie nowych podstron po stronie klienta, czyli używając języka JavaScript, które miało zastosowanie wizualne, nie było skomplikowane, ponieważ polegało to wyłącznie na wpisaniu nowej strony do bazy danych i przeładowaniu paska nawigacyjnego poprzez technologię Ajax.

Edytowanie podstron wymagało ustalenia wszystkich nazw podstron i umieszczenie ich do listy inputów. Dzieje się to poprzez zlokalizowanie wszystkich tagów w pasku nawigacyjnym i pobraniu ich wewnętrznego kodu HTML (.innerHTML), w tym przypadku nazw podstron. Reakcja na naciśnięcie przycisku „Save” powoduje pobranie nowych, bądź niezmienionych nazw i aktualizację rekordu w bazie danych. W momencie usunięcia jakiegokolwiek inputa poprzez naciśnięcie ikonki zamykającej, zostaje automatycznie usunięta podstrona z bazy, o ile naciśnie się później przycisk „Save”. W przeciwnym razie wszelkie zmiany nie zostaną dokonane.

Po stronie serwera (ASP):

Nazwy podstron przechowywane są w bazie danych w tabeli PageSet. W celu dodania nowej podstrony, konieczne jest stworzenie nowego rekordu w bazie danych, a następnie wypełnienie go odpowiednimi danymi. Natomiast w przypadku edycji nazw podstron potrzebna jest jedynie aktualizacja istniejących rekordów w bazie danych. Zarówno do dodawania nowych podstron jak i edytowania już istniejących wykorzystano odpowiednie metody Webservice. Metody to odpowiednio CreatePage, która jako parametr przyjmuje nazwę nowej strony, oraz UpdatePage, która jako parametry przyjmuje identyfikator modyfikowanej strony oraz zmienioną nazwę.

6.5. Zmiana hasła redaktora strony

Istotnym elementem projektu jest możliwość zmiany hasła do logowania przez administratora w celu zabezpieczenia administracyjnej części strony przed niepowołanym dostępem. Jest to wręcz konieczny zabieg ze względu na to, że po zainstalowaniu aplikacji standardowym hasłem jest wyraz „admin”. Aby zmienić hasło należy zalogować się do administracyjnej części strony i kliknąć guzik „Admin”, a następnie po pojawieniu się okna dialogowego wpisać nowe hasło oraz przepisać je powtórnie, w celu wykrycia możliwej pomyłki przy jego wprowadzaniu. Gdy oba wpisane hasła nie są takie same, wyświetla się komunikat o błędzie. W przeciwnym wypadku ze skryptu po stronie klienta wywoływana jest metoda „ChangePassword” zaimplementowana w Web serwisie, gdzie następnie generowany jest skrót hasła za pomocą algorytmu MD5, który jest później zapisywany do bazy danych. Od tego momentu administrator strony musi logować się za pomocą nowego hasła.

6.6. Dodawanie i rozmieszczanie elementów z paska narzędzi

Cała aplikacja opiera się na metodyce Drag&Drop, co oznacza, że użytkownik łapie kursorem dany element i przeciąga, po czym upuszcza w dowolnie wybranym przez siebie dostępnym miejscu. Dzieje się to dzięki interakcjom Draggable oraz Sortable, które są zaimplementowane w bibliotece jQuery UI. Pierwsza z nich uruchamia funkcję przeciągającą na jakimkolwiek elemencie DOM. Dzięki opcji „connectToSortable” gadżety z dolnego Toolboxa mogą być przenoszone do listy „contentUL”.^[2]

```
$("#toolbox > li").draggable({
    connectToSortable: '#contentUL',
    containment: "document",
    revert: "invalid",
    helper: "clone"
});
```

Druga interakcja pozwala grupie elementów DOM na bycie sortowanym. Klikając i przeciągając element do nowego miejsca na liście, pozostałe będą się dostosowywać do nowej kolejności. Funkcja użyta w projekcie posiada w sobie opcję „recive”, która dopełnia komunikację z toolbox’em i uruchamia odpowiednie okna dialogowe w celu konfiguracji nowo dodanych elementów.^[2]

```
$("#contentUL").sortable({
  placeholder: "ui-state-highlight",
  update: function (event, ui) {
    if (event.handleObj.namespace == 'sortable') {
      WebService.SaveContent($('#contentUL').html(), $('#currentPage').text());
    }
  },
  receive: function (event, ui) {
    var newItem = $(this).data().sortable.currentItem;
    newItem.html("");
    switch (ui.item.attr('id')) { ... }
  });
```

6.7. Dodawanie i edycja tekstu za pomocą edytora oraz implementacja nowej kontrolki

W aplikacji jako edytor tekstu/HTML wykorzystano komponent jHtmlArea – strona domowa projektu <http://jhtmlarea.codeplex.com/>.

jHtmlArea jest to lekki, prosty i łatwo rozszerzalny graficzny (WYSIWYG- What You See Is What You Get) edytor HTML zbudowany w oparciu o bibliotekę jQuery. Komponent ten pozwala w łatwy sposób edytować HTML w przeglądarce (Edit-In place), nie zaglądając do kodu źródłowego, jednak w dalszym ciągu to umożliwiając.

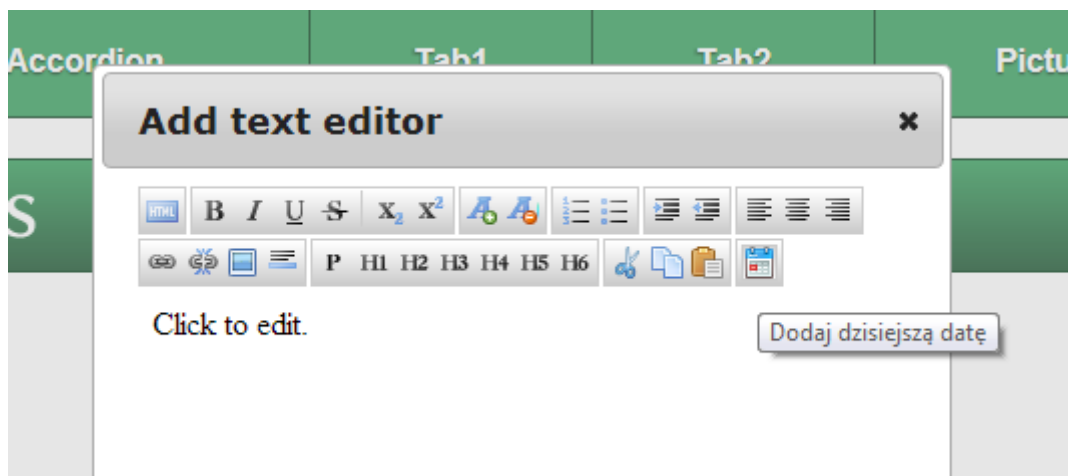
W sieci można znaleźć wiele tego typu edytorów, jednak zdecydowano się na wybór jHtmlArea z kilku ważnych powodów. Po pierwsze jest on całkowicie darmowy, posiada otwarty kod źródłowy, wydany na licencji Microsoft Public License. jHtmlArea jest prosty w obsłudze i konfiguracji. Jest lekki, waży zaledwie 23.6kb wraz z kaskadowymi arkuszami stylów oraz plikami graficznymi. Obsługuje wszystkie najważniejsze przeglądarki internetowe takie jak: Internet Explorer 7/8/9+, Firefox 3+, Safari 4+ oraz Google Chrome. Również bardzo istotne jest to, że jHtmlArea posiada kompletną dokumentację, dzięki czemu nie było problemów z osadzeniem komponentu w aplikacji.

Szczegóły dotyczące działania jHtmlArea, można znaleźć w dokumentacji pod linkiem: <http://jhtmlarea.codeplex.com/documentation>.

Ważnym elementem projektu jest możliwość rozbudowywania i dostosowywania aplikacji do własnych potrzeb przez osoby bardziej doświadczone w technologiach internetowych takich jak HTML, CSS czy JavaScript. Nie jest więc problemem dodanie własnej kontrolki do edytora tekstu. W projekcie DotNetCms dla przykładu został dodany komponent, dzięki któremu do tekstu zostaje automatycznie wstawiona aktualna data. Tworząc tę kontrolkę w skrypcie edytora – jHtmlArea-0.7.5.js trzeba było najpierw rozpoznać odpowiednie miejsce do wprowadzenia zmian, a potem należało stworzyć tablicę miesięcy, następnie pobrać numer dnia, miesiąca i roku, a później wstawić do zawartości edytora tekst wygenerowany na podstawie pobranych danych i tablicy miesięcy (np. 12 stycznia 2012). Tworzona kontrolka posiada również odpowiedni identyfikator, dzięki któremu w pliku arkusza stylów możemy dodać swoją własną ikonkę do edytora.

```
[{
  css: "datebutton",
  text: "Dodaj dzisiejszą datę",
  action: function (btn) {
    var months = new Array("styczeń", "luty", "marzec", "kwiecień", "maj",
      "czerwiec", "lipiec", "sierpień", "wrzesień", "październik", "listopad",
      "grudzień");
    var date = new Date();
    var currentDay = date.getDate();
    var currentMonth = date.getMonth();
    var currentYear = date.getFullYear();

    this.pasteHTML(currentDay + " " + months[currentMonth] + ", " +
      currentYear + "r.");
  }
}]
```



Rysunek 25: Nowa kontrolka z dzisiejszą datą (Źródło: opracowanie własne, przykładowa strona WWW projektu DotNetCms)

6.8. Użycie gadżetu zakładka (Tab)

Podstawową funkcją systemu jest zarządzanie treścią poprzez tworzenie zakładki (Tab'ów). W projekcie istnieją dwie możliwości dodania tego gadżetu: zakładki wywołane przy pomocy jQuery UI oraz zakładki kompatybilne z Gumbo Framework. Różnią się one przede wszystkim layoutem.

Aby można było dodać jakiegokolwiek taby należało najpierw ustalić tytuły. Każda zakładka musi mieć unikalne id w podstronie (na różnych podstronach id mogą się powtarzać). Skrypt na początku odnajdywał wszystkie zakładki i stwarzał tablicę `idTabs`, która informowała o tym, które id są już zajęte. Id w każdej nowej zakładce było ustawione o najniższej wartości, to znaczy, jeżeli na stronie istniały już zakładki o id „Tab1”, „Tab2”, „Tab4”, nowy tab otrzymywał id o wartości „Tab3”. Reakcja na ikonki usunięcia i dodawania inputów formularzy powodowały zmiany w tablicy `idTabs`. W formularzu wszystkie inputy można było przenosić poprzez zastosowanie interakcji `sortable()` z biblioteki jQuery UI, która została opisana w punkcie 6.7.

Następnym krokiem było pobranie wszystkich nazw z formularza i przypisanie ich do odpowiednich zakładek. Wygenerowany kod HTML był zależny od wyboru tabów przez użytkownika.

W przypadku zastosowania zakładek „Tab1”, które są obsługiwane przez jQuery UI, należało zastosować funkcje wywołania interakcji w tabach `.tab()`.^[2] Każde takie wywołanie musiało być wykonane podczas przeładowania strony lub głównego content'u.

6.9. Tworzenie narzędzia Accordion

Zastosowanie Accordion'ów wygląda bardzo podobnie do tabów, lecz różnią się wyglądem i mogą być zamykane. Klikając na nagłówki można zwinąć lub rozwinąć treści, które podzielone są na sekcje logiczne podobnie jak karty. Podstawowym znacznikiem HTML jest seria nagłówków `<H3>` oraz `div` zawartości więc treść może być użyteczna bez korzystania z JavaScript.

W tworzeniu Accordiona wykorzystany był skrypt generujący Taby. Ustawienie nazw nagłówków opiera się na tej samej logice co ustawienie tytułów zakładek. Wystarczyło jedynie zmienić generator HTML tak, aby mógł umieszczać na stronie zarówno zakładki jak i accordion.

Również podobnie jak w przypadku zakładek, jako narzędzie wykorzystujące funkcje biblioteki jQuery, wymagane jest inicjowanie funkcji `.accordion()`^[2] przy każdym wywołaniu strony lub przeładowaniu centralnego contentu.

6.10. Dodawanie, edycja i skalowanie zdjęć

Zdjęcia, obrazki, rysunki - inaczej pliki graficzne, są nierozłącznymi elementami stron internetowych. Tak jak większość gadżetów w projekcie DotNetCms odbywa się to przy użyciu utworzonego w oknie dialogowym formularza. Główną funkcją napisaną w JavaScript jest zarządzanie formą wyboru pobrania obrazka. Program umożliwia 2 opcje.

- Ładowanie zdjęć z URL.
- Pobieranie zdjęć z pliku i wrzucanie ich na serwer.

Zadaniem skryptu jest rozróżnienie, którą opcję redaktor strony wybrał. W formularzu jest wyświetlana lista rozwijana, w której można wybrać opcje poboru grafiki. Tak jak w każdym formularzu następuje walidacja danych. Serwis sprawdza czy adres nie jest pusty, ale nie przewiduje czy podany adres jest prawidłowy. W momencie dodawania zdjęć z pliku następuje odświeżenie zawartości strony przy pomocy funkcji napisanej w Ajaxie.

Unikalną grafiką na stronie jest logo, ponieważ znajduje się ono na każdej podstronie. Jest to jedyna grafika, którą można edytować, ale nie można jej całkowicie usunąć ze strony. Element ten jest klasy „notDeletable” przez co skrypt wie, że nie można tego obrazka skasować ze strony. W przypadku braku chęci posiadania loga na stronie zalecane jest stworzenie grafiki z przezroczystym tłem i załadowanie jej na serwer. Ten efekt sprawia, że strona wygląda jakby tego loga nie posiadała.

Jedną z możliwości aplikacji DotNetCms jest uploadowanie zdjęć bezpośrednio na serwer. Funkcjonalność ta została zrealizowana przy pomocy Generic Handler. Gdy użytkownik zaakceptuje wybór zdjęcia klikając przycisk „Add” na okienku „Add picture”, generowane jest asynchroniczne żądanie (przy pomocy AJAX’a)^[3] do FileUploaderHandler.ashx. Handler ten przy pomocy klasy HttpContext wyłuskuje odpowiednie dane wybranego zdjęcia, a następnie dzięki metodzie SaveAs klasy HttpPostedFile, zapisuje zdjęcie na serwerze.

```
$.ajax({
    type: "POST",
    url: "../FileUploaderHandler.ashx",
    contentType: false,
    processData: false,
    data: data,
    error: function () {
        alert("There was error uploading files!");
        bValid = false;
    }
});
```

Ważnym założeniem projektu jest wyświetlanie w przeglądarce zdjęcia dostosowanego do rodzaju urządzenia, za pomocą którego użytkownik końcowy przegląda stronę. Dla urządzeń mobilnych nie jest pożądane ładowanie zdjęć o dużej rozdzielczości i rozmiarze ze względu na szybkość i limity transferu danych, a także z uwagi na aspekt czysto praktyczny – na małym w stosunku do ekranu komputera wyświetlaczu i tak nie będzie można cieszyć się z oglądania tak dużych zdjęć. Aby dostosować plik graficzny do odpowiedniego urządzenia, po wysłaniu wybranego przez administratora obrazu, na serwerze jest on przetwarzany i tworzone są jego 3 kopie. Każdą cechuje inna rozdzielczość i jakość. Pierwszą kopią jest oryginalne zdjęcie, druga to wersja średnia – stworzona z myślą o tabletach oraz trzeci wariant stworzony dla smartphon'ów. Każda wersja obrazu jest dodawana na serwer z odpowiednim przedrostkiem - „original”, „medium” bądź „small”. Kiedy osoba uruchamia stronę, JavaScript wykrywa rozmiar okna przeglądarki i zmienia źródła plików obrazu w tagach na odpowiednie, zanim cała zawartość strony zostanie wczytana. W ten sposób unikamy ładowania dużych zdjęć na urządzeniach mobilnych.

6.11. Dodawanie mapy i rozmieszczanie punktów

Tworząc narzędzie mapy posługujemy się dodatkiem do jQuery Mobile korzystającym z Google Maps Api w wersji trzeciej. Po przeciągnięciu kontrolki mapy z paska narzędzi w skrypcie addMapAdmin.js wywołana zostaje funkcja, która dodaje do zawartości strony znacznik <div> zawierający kod HTML mapy wygenerowany dzięki skorzystaniu z interfejsu map Google. Aby dodać znacznik do mapy redaktor strony klika w wybranym przez siebie miejscu, po czym pojawia się okno, gdzie może dodać swój komentarz oraz zobaczyć dokładne dane adresowe klikniętego punktu. Jego koordynaty oraz komentarz są przesyłane za pomocą webserwisu na serwer, gdzie są zapisywane w bazie danych. Kiedy użytkownik końcowy wchodzi na podstronę zawierającą mapę, w funkcji JavaScriptu zostaje wywołane żądanie pobrania współrzędnych znacznika oraz komentarza, które korzysta z metody POST w ramach protokołu HTTP za pomocą technologii ajax i webserwisu, po czym serwer wyszukuje w bazie odpowiedni punkt i wysyła odpowiedź w formacie JSON. Korzystając z odebranych z serwera koordynat, znacznik jest dodawany dzięki api Google za pomocą JavaScriptu do mapy.

Administrator może również z łatwością usunąć punkt z mapy klikając na niego i wybierając opcję „Remove” z nowo utworzonego okna dialogowego. Kiedy punkt zostaje usunięty następuje wywołanie metody webserwisu ze skryptu działającego po stronie klienta, która usuwa wybrany punkt z bazy (ustawia koordynaty i komentarz na wartość null). Oprócz tego mapa, tak jak każdy inny dodany komponent, może zostać skasowana z zawartości strony. Zostaje wtedy wywołana metoda „remove” z biblioteki jQuery na elemencie div, który odpowiada za przechowywanie mapy. Cały kod HTML, który zawiera, łącznie z nim samym, zostaje usunięty z zawartości strony.

6.12. Umieszczanie podowiedzi o edycji gadżetów

Na początku myślą przewodnią dodania tooltip'ów było skorzystanie z efektownego rozwiązania, które można pobrać stronie <http://www.menucool.com/tooltip/javascript-tooltip>. Jednak zbyt usilne komplikowanie kodu źródłowego i wiążące się z tym reklamy przyczyniły się do porzucenia tego pomysłu.

Zdecydowano się więc na własną implementację tej funkcjonalności z użyciem biblioteki jQuery UI. Rozwiązanie nie okazało się trudne. Wystarczyło bowiem dodać atrybut „tittle” do nowo tworzonych elementów i nadać im odpowiednią klasę. Wartość tego atrybutu była definiowana zależnością od dodawanego elementu. Następnie należało jedynie wywołać funkcje z wyżej wymienionej biblioteki: .tooltip(). Trzeba było pamiętać, że strona klienta nie mogła w sobie zawierać podpowiedzi jak edytować gadżety na stronie. Dlatego w pliku Default.aspx została napisana funkcja, która usuwa wszystkie tytuły edytowalnych elementów. Znajomość klasy zdecydowanie ułatwiła wyznaczenie, które tytuły w kodzie HTML należało usunąć.

6.13. Implementacja instalatora

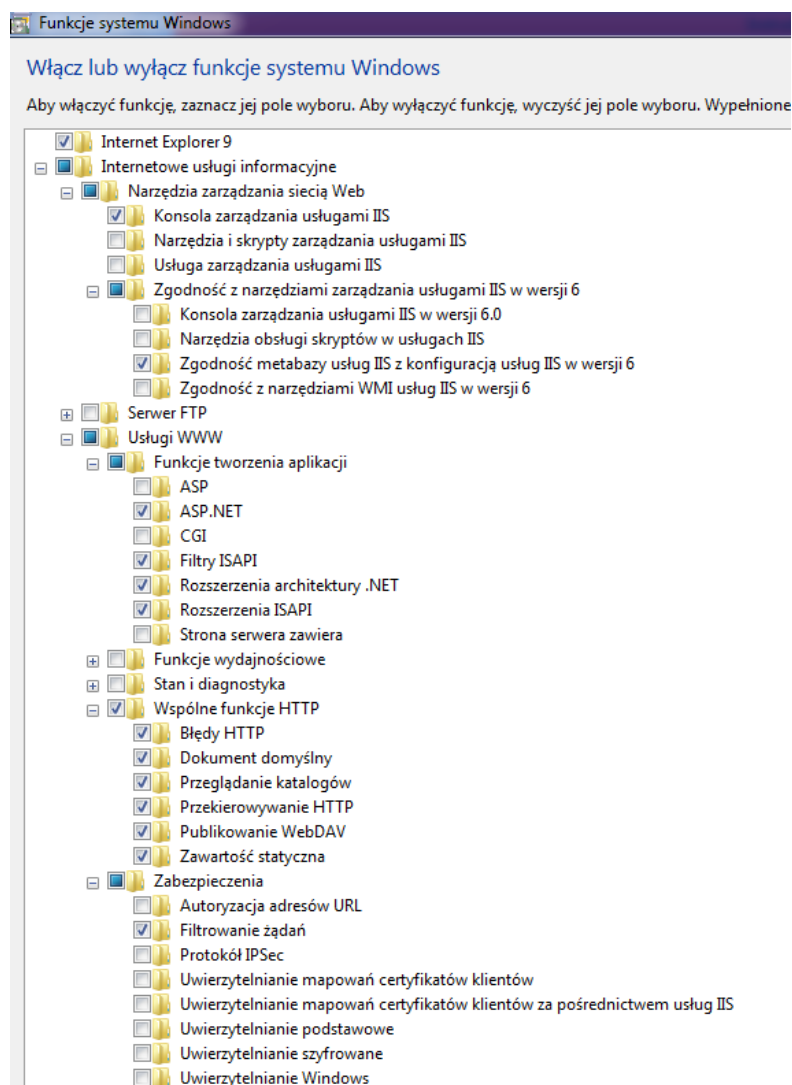
Jako instalator wykorzystano szablon projektu „Web Setup Project”, który jest dostępny w Microsoft Visual Studio 2012. Jest to oddzielny projekt dodany do solucji aplikacji. Jako „Project Output” projektu instalatora wybrano wszystkie pliki z projektu DotNetCms, zostały również ustawione „Launch Condition” czyli warunki poprawnego uruchomienia instalatora.^[1] Jedynym warunkiem uruchomienia instalatora jest posiadanie zainstalowanego serwera IIS (Internet Information Services) w wersji co najmniej 6.0. W przypadku gdy użytkownik nie posiada danego oprogramowania, wyświetlany jest stosowny komunikat.

7. Opis instalacji

7.1. Przygotowanie środowiska

Do uruchomienia aplikacji potrzebny jest .Net Framework 4.0 (który należy pobrać ze strony <http://www.microsoft.com/pl-pl/download/details.aspx?id=17851>), SQL Server CE 4.0 (który należy pobrać ze strony <http://www.microsoft.com/en-us/download/details.aspx?id=30709>) oraz serwer Internetowych Usług Informacyjnych IIS. Pliki instalacyjne IIS oraz wsparcie techniczne, można znaleźć pod adresem <http://www.iis.net/>.

Uwaga, użytkownicy Windows 7 nie muszą pobierać serwera IIS, gdyż jest on dołączony do systemu operacyjnego. Należy kliknąć Start -> Panel Sterowania -> Programy. Następnie kliknąć znajdujący się po lewej stronie napis „Włącz lub wyłącz funkcje systemu Windows”. Otworzy się okno „Funkcje systemu Windows”. Przykładowa konfiguracja serwera IIS przedstawiona została na rysunku poniżej:

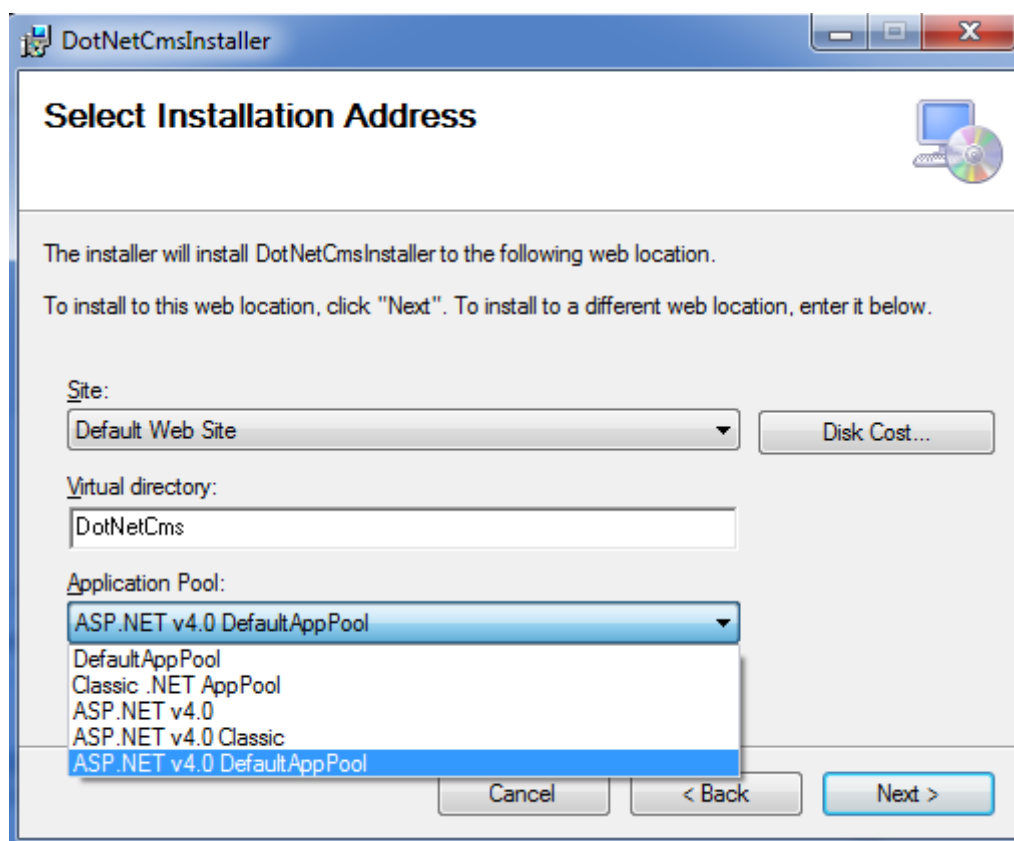


Rysunek 26: Konfiguracja serwera IIS (Źródło: opracowanie własne)

7.2. Instalacja

Po pomyślnym zainstalowaniu potrzebnego oprogramowania, można przystąpić do wdrożenia aplikacji. W tym celu należy uruchomić instalator klikając dwa razy na plik Setup.exe.

Gdy zostanie wyświetlone okno podobne do poniższego, należy podać nazwę wirtualnego katalogu oraz z listy „Application Pool” wybrać ASP.NET v4.0 DefaultAppPool. Następnie należy kliknąć Next i kontynuować instalację.



Rysunek 27: Wybór Application Pool w instalatorze (Źródło: opracowanie własne)

7.3. Uruchamianie

Po prawidłowym zakończeniu procesu instalacji, aplikacja jest dostępna w „Menadżerze Internetowych Usług Informacyjnych”. W celu uruchomienia menadżera można skorzystać z wyszukiwarki systemowej, bądź przejść do: Panel Sterowania -> System i zabezpieczenia -> Narzędzia administracyjne.

Przed pierwszym uruchomieniem należy zmienić uprawnienia dostępu do katalogów:

- C:\inetpub\wwwroot\[Nazwa_wirtualnego_katalogu]\App_Data
- C:\inetpub\wwwroot\[Nazwa_wirtualnego_katalogu]\images.

Należy kliknąć prawym przyciskiem myszy na katalog i wybrać właściwości, następnie wybrać zakładkę zabezpieczenia i kliknąć przycisk Edytuj. Po wybraniu użytkownika należy zezwolić mu na pełną kontrolę i zapisać zmiany.

Po wykonaniu wszystkich czynności można uruchomić aplikację w oknie przeglądarki wpisując adres [http://localhost/\[Nazwa wirtualnego katalogu\]](http://localhost/[Nazwa wirtualnego katalogu]), bądź przejść do menadżera IIS i po wybraniu odpowiedniej aplikacji kliknąć przycisk „Przeglądaj”. Panel administracyjny dostępny jest pod adresem [http://localhost/\[Nazwa wirtualnego katalogu\]/Admin/Admin.aspx](http://localhost/[Nazwa wirtualnego katalogu]/Admin/Admin.aspx)

Domyślne dane uwierzytelniające:

Login: admin

Hasło: admin

Po zalogowaniu możliwa jest zmiana hasła administratora. Login można zmienić jedynie modyfikując wpis bezpośrednio w bazie danych.

8. Raport końcowy

8.1. Osiągnięte rezultaty

Grupa projektowa stworzyła łatwą w użyciu aplikację typu CMS, generującą proste strony internetowe. Zespół w łatwy i efektywny sposób rozwiązał problem automatycznej zamiany zawartości strony na wersję mobilną wykorzystując dosyć nową technologię jaką jest Responsive Web Design.

Projekt DotNetCms może służyć jako szkielet dla dużo większej aplikacji. Przy odpowiedniej rozbudowie programu można stworzyć ciekawe i pokaźne narzędzie, które będzie mogło się odnaleźć na rynku aplikacji informatycznych. Coraz szybciej rozwijająca się technologia mobilna sprawia, że serwis staje się tym bardziej atrakcyjny, a możliwości rozwoju oraz dodawania nowych gadżetów są niemalże nieograniczone.

Główne elementy możliwej rozbudowy:

- Zwiększenie ilości gadżetów, które mogą poprawić atrakcyjność i czytelność strony (przykłady: Baza danych z produktami, formularz kontaktowy, który wysyła maile, Paypal).
- Rozwinięcie wbudowanych już gadżetów (przykład: możliwość dodania dynamicznej galerii zdjęć).
- Poprawa funkcjonalności wdrożonych rozwiązań (przykład: edycja pliku CSS, zamiast ciągłego kolorowania elementów przy przeładowywaniu stron).
- Stworzenie nowych funkcjonalności (kolorowanie tła, ingerencja w kod JavaScript na wysokości interfejsu aplikacji, zarządzanie bazą danych bez konieczności uruchamiania specjalistycznego oprogramowania).
- Ulepszenie funkcjonalności wersji mobilnej strony.
- Testy na większej ilości przeglądarek mobilnych i desktopowych.

8.2. Podział wykonanej pracy między członków grupy projektowej

Każdy z członków grupy projektowej miał przydzielony pewien zakres odpowiedzialności. Rafał Lewandowski zajął się częścią serwerową wraz z połączeniem jej ze stroną klienta oraz intuicyjną instalacją projektu. Mateusz Kowalski skupił się na implementacji po stronie klienta, spójności aplikacji oraz dokumentacji projektu. Zadania Tymoteusza Gach opierały się na zwiększeniu funkcjonalności serwisu oraz poprawy jego atrakcyjności. Ponadto każdy z projektantów odpowiedzialny był za wspólne zarządzanie, organizacją pracy, testowanie oraz dbanie o spójność repozytorium.

Bardziej szczegółowy przydział do zadań został przedstawiony jest na poniższym diagramie:

Mateusz Kowalski	<p>Zarządzanie</p> <ul style="list-style-type: none">• Wyznaczanie prac.• Kontrola.• Nadzorowanie repozytorium. <p>Szkielet</p> <ul style="list-style-type: none">• Stworzenie szkieletu, na którym są dodawane wszystkie elementy.• Integracja Gumbo Framework z projektem.• Dodawanie elementów.• Umożliwienie dodania elementów do strony.• Możliwość przemieszczania elementów. <p>Zakładki (Taby)</p> <ul style="list-style-type: none">• Dodawanie tab'ów.• Edycja tab'ów.• Dostosowanie gadżetu „Tab2” do kolorystyki strony.• Wykorzystanie jHtmlArea do zawartości tab'ów. <p>Dodawanie zdjęć</p> <ul style="list-style-type: none">• Dodawanie zdjęć url.• Implementacja interfejsu umożliwiająca dodawanie zdjęć z pliku.• Edycja zdjęć i usuwanie zdjęć. <p>Interfejs</p> <ul style="list-style-type: none">• Inicjalizowanie wyglądu formularzy do konfigurowania elementów aplikacji.• Walidacja formularzy.• Zabezpieczenie przed możliwymi błędami użytkownika• Dodanie podpowiedzi informujących o możliwych akcjach do wykonania.• Podpowiedzi są widoczne jedynie w panelu administratora. <p>Zmiana koloru</p> <ul style="list-style-type: none">• Kolorowanie wybranych elementów na stronie.• Wygenerowanie drugorzędnych kolorów cieni, obramowań itp., elementów, na które najedzie się myszką.• Edycja plików CSS w Gumbo Framework potrzebne do kolorowania. <p>Model danych</p> <ul style="list-style-type: none">• Projekt struktury bazy danych. <p>Podstrony</p> <ul style="list-style-type: none">• Wizualne tworzenie podstron w aplikacji.• Edycja podstron.• Wspomaganie Rafała Lewandowskiego z połączeniem części serwerowej ze skryptem tworzącym podstrony (Programowanie w parach). <p>Ajax</p> <ul style="list-style-type: none">• Automatyczne odświeżenie wszystkich edytowanych elementów.• Powtórna inicjalizacja elementów jQuery UI. <p>Dokumentacja</p> <ul style="list-style-type: none">• Tworzenie i kreowanie dokumentacją.• Integracja raportów innych projektantów z dokumentacją. <p>Testowanie</p> <ul style="list-style-type: none">• Tworzenie testów.
-------------------------	--

	<ul style="list-style-type: none"> • Zgłaszanie nieprawidłowości.
Tymoteusz Gach	<p>Zarządzanie</p> <ul style="list-style-type: none"> • Wyznaczanie prac. • Kontrola. • Nadzorowanie repozytorium. <p>Szkielet</p> <ul style="list-style-type: none"> • Stworzenie szkieletu, na którym są dodawane wszystkie elementy. • Implementacja Gumby Framework i integracja z projektem. <p>Dodawanie elementów.</p> <ul style="list-style-type: none"> • Umożliwienie dodania elementów do strony. • Możliwość przemieszczania elementów. <p>Nowa kontrolka w edytorze</p> <ul style="list-style-type: none"> • Edycja gotowego rozwiązania, jakim jest jHtmlArea i dodanie nowej kontrolki „Dzisiejsza data”. <p>Dostosowanie zdjęć do urządzenia</p> <ul style="list-style-type: none"> • Rozpoznanie wielkości okna urządzenia. • Dobór zdjęć z odpowiednią rozdzielczością do okna urządzenia. • Tworzenie 3 kopii zdjęć z różną jakością i rozdzielczością po stronie serwera. <p>Mapy</p> <ul style="list-style-type: none"> • Implementacja jQuery Ui Map, która korzysta z Google API. • Dodawanie map. • Dodawanie punktów do map. • Zapis lokalizacji i punktów w bazie. • Edycja i usuwanie map. • Edycja punktu na mapie. <p>Testowanie</p> <ul style="list-style-type: none"> • Tworzenie testów. • Zgłaszanie nieprawidłowości. <p>Ajax</p> <ul style="list-style-type: none"> • Użycie Ajaxa przy dodawaniu map.
Rafał Lewandowski	<p>Zarządzanie</p> <ul style="list-style-type: none"> • Wyznaczanie prac. • Kontrola. • Nadzorowanie repozytorium. <p>Szkielet</p> <ul style="list-style-type: none"> • Stworzenie szkieletu, na którym są dodawane wszystkie elementy. • Implementacja Gumby Framework i integracja z projektem. <p>Dodawanie elementów.</p> <ul style="list-style-type: none"> • Umożliwienie dodania elementów do strony. • Możliwość przemieszczania elementów. <p>Accordion</p> <ul style="list-style-type: none"> • Dostosowanie skryptów tworzenia i edycji tab'ów, tak aby działał też na Accordion'ach. <p>Edytor tekstu</p> <ul style="list-style-type: none"> • Implementacja wizualnego edytora HTML – jHtmlArea i integracja z projektem. <p>Ładowanie zdjęć na serwer</p> <ul style="list-style-type: none"> • Implementacja ładowania zdjęć z pliku.

Model danych
<ul style="list-style-type: none">• Implementacja struktury bazy danych.
Podstrony
<ul style="list-style-type: none">• Implementacja i zapis do bazy danych nowych podstron.
Serwer
<ul style="list-style-type: none">• Implementacja części serwerowej.• Integracja aplikacji z napisaną wcześniej częścią po stronie klienta (HTML, JavaScript).• Autoryzacja administratora.• Zapisywanie wszystkich elementów do bazy danych przez Webservice.• Zapisywanie kolorystyki serwisu przez Webservice.• Pobieranie zawartości contentu i kolorystyki strony przez Webservice.
Logowanie
<ul style="list-style-type: none">• Implementacja i interfejs logowania.
Zarządzanie administratorem
<ul style="list-style-type: none">• Interfejs podstrony do ustawień administratora.• Edytowanie administratora.
Ajax
<ul style="list-style-type: none">• Użycie asynchronicznej funkcji Ajaxu do ładowania zdjęć na serwer.
Instalator
<ul style="list-style-type: none">• Implementacja instalatora.• Konfigurowanie IIS.
Testowanie
<ul style="list-style-type: none">• Tworzenie testów.• Zgłaszanie nieprawidłowości.

Diagram 3: Podział obowiązków na członków zespołu. (Źródło: opracowanie własne)

8.3. Podsumowanie

Dla projektantów była to pierwsza tak duża aplikacja o takim stopniu zaangażowania. Pozwoliła na dogłębne utrwalenie technologii .NET i jQuery, z którymi wcześniej byli już obeznani. Zupełnie nowym aspektem tworzenia oprogramowania było planowanie ryzyka, zarządzanie współpracą i korzystanie z takich narzędzi jak repozytorium. Zdecydowanie ważnym elementem projektowania był harmonogram oraz spotkania, na których omawiało się efekty swoich prac. Pozwoliło to na lepsze zarządzanie ryzykiem wiążącym się z realizacją pracy inżynierskiej, która była według planu dyplomantów aplikacją wysokiego ryzyka. Podczas ostatniego spotkania członków zespołu dyskutowano na temat tego, co było robione w sposób nieprawidłowy, a także o możliwościach zapobiegania temu w przyszłości.

Ustalono, że korzystanie z gotowych rozwiązań wiąże się także z wymaganiami. Każde, choć nie wiadomo jak efektywne rozwiązanie, zazwyczaj jest niezrozumiałe bez dobrze napisanej dokumentacji. Często wydajniej jest napisać własną solucję niż skorzystać z innych bez zrozumienia. Tak samo jest z własnym oprogramowaniem. Warto napisać dobrą dokumentację, która pozwoli na bezproblemowy powrót do analizowania stworzonego samodzielnie projektu.

9. Bibliografia

1. Microsoft ASP.NET 4 Step by Step; George Shepherd; Microsoft 2010
2. JQUERY: NOVICE TO NINJA; Earl Castledine & Craig Sharkie; Sitepoint 2012
3. <http://api.jquery.com/jquery.ajax/>