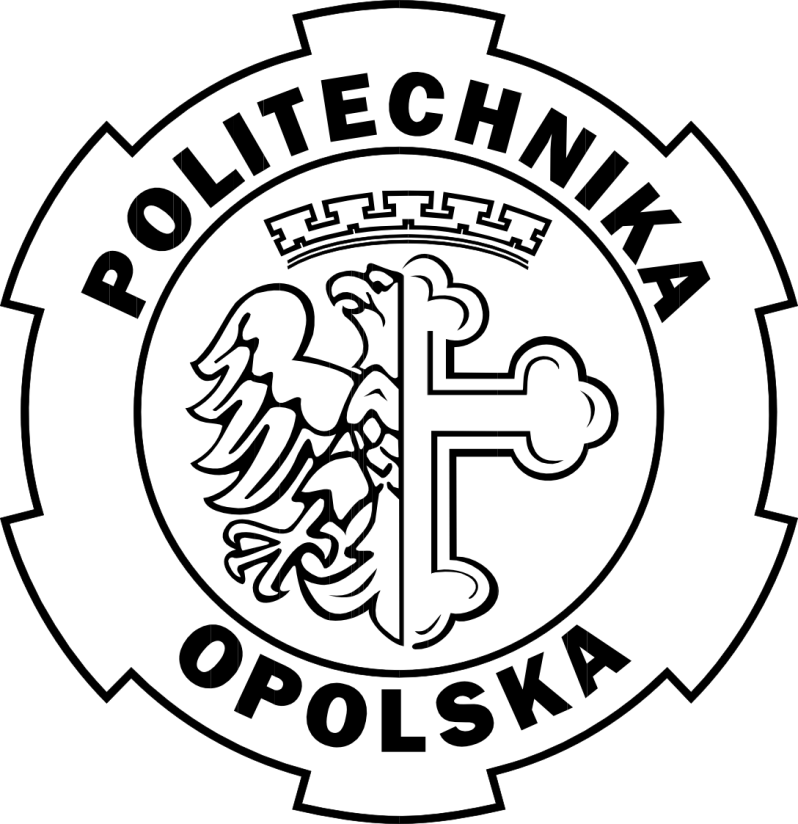
**Politechnika Opolska**

Wydział Elektrotechniki, Automatyki i Informatyki

Instytut Automatyki i Informatyki

Kierunek Informatyka

Studia Dzienne



**Praca dyplomowa inżynierska**

Projekt i implementacja systemu budowania i administrowania stronami internetowymi przez personel nietechniczny – część logiki biznesowej

|  |  |
| --- | --- |
| Promotor: | Autor: |
| dr inż. hab.Rafał Stanisławski, prof. PO | Kamil Ślusarczyk |

Opole, 2016

Spis treści

[1. Wprowadzenie 3](#_Toc441264231)

[2. Cel i zakres pracy 4](#_Toc441264232)

[3. Narzędzia wykorzystane w projekcie 6](#_Toc441264233)

[3.1. Serwer 6](#_Toc441264234)

[3.2. Platforma .NET 6](#_Toc441264235)

[3.3. Język C# 7](#_Toc441264236)

[3.4. LINQ 7](#_Toc441264237)

[3.5. Kontrola wersji 8](#_Toc441264238)

[3.6. Producteev 8](#_Toc441264239)

[3.7. RestSharp 8](#_Toc441264240)

[3.8. Wzorce projektowe 9](#_Toc441264241)

[3.8.1. Unit of work 9](#_Toc441264242)

[3.8.2. Wstrzykiwanie zależności 9](#_Toc441264243)

[3.8.3. Repozytorium 9](#_Toc441264244)

[3.8.4. MVC 10](#_Toc441264245)

[3.9. Visual Studio 10](#_Toc441264246)

[3.10. Autofac 10](#_Toc441264247)

[4. Projekt systemu współnie z Arturem Stelmachem 11](#_Toc441264248)

[4.1. Wymagania biznesowe 11](#_Toc441264249)

[4.2. Słownik pojęć 11](#_Toc441264250)

[4.2. Wymagania funkcjonalne 12](#_Toc441264251)

[4.2.1 Specyfikacja wymagań dla panelu administratora 12](#_Toc441264252)

[4.2.2. Specyfikacja wymagań dla panelu klienckiego 13](#_Toc441264253)

[4.2.3. Wymagania niefunkcjonalne 13](#_Toc441264254)

[4.3. Warstwy aplikacji 13](#_Toc441264255)

[4.4. Projekt bazy danych 14](#_Toc441264256)

[4.5. Diagramy sekwencji 17](#_Toc441264257)

[4.5.1. Rejestracja na usługę 17](#_Toc441264258)

[4.5.2. Edycja aktualności 17](#_Toc441264259)

[4.5.3. Usuwanie aktualności 18](#_Toc441264260)

[4.5.4. Wyświetlenie statystyk 19](#_Toc441264261)

[4.5.5. Skorzystanie z formularza kontaktowego 19](#_Toc441264262)

[4.5.6. Przekazanie plików na serwer 20](#_Toc441264263)

[5. Prezentacja systemu 20](#_Toc441264264)

[5.1. Ekran logowania 20](#_Toc441264265)

[6. Budowa systemu 30](#_Toc441264266)

[6.1. CRUD Logic 30](#_Toc441264267)

[6.2. Restore news 34](#_Toc441264268)

[6.3. Restore Page 35](#_Toc441264269)

[6.4. Gettery usług 35](#_Toc441264270)

[6.5. Sprawdzanie dostępności usług 38](#_Toc441264271)

[6.6. Moduł statystyk 39](#_Toc441264272)

[6.7. Moduł e-mail 43](#_Toc441264273)

[6.8. Moduł RestSharp 45](#_Toc441264274)

[7. Podsumowanie 46](#_Toc441264275)

[8. Bibliografia 47](#_Toc441264276)

# Wprowadzenie

Podwaliny dzisiejszych aplikacji webowych były rozwijane od XXX. W tamtych czasach były to jednak programy tworzone przez specjalistów i wykorzystywane zazwyczaj na użytek wąskiej specjalności naukowej. Wraz ze wzrostem dostępności Internetu aplikacje internetowe zaczęły być coraz częściej wykorzystywane do celów biznesowych (np. E-commerce) jak i rekreacyjnych. Łatwa dostępność do tego typu udogodnień sprawiła, że konieczne jest wprowadzenie w świat Internetu ludzi, którzy na codzień nie mają styczności ze światem technologii. Fakt ten rodzi wiele wyzwań przed twórcami aplikacji internetowych. Muszą oni zadbać aby efekt ich pracy był funkcjonalny, estetyczny i opłacalny.

Efekt poniższej pracy ma pozwolić osobom nietechnicznym nie tylko uczestniczenie w świecie internetu w roli odbiorcy, ale także i twórcy. Podczas fazy projektowej przyjęto, że wynik prac ma w sposób przyjazny dla użytkownika umożliwiać tworzenie stron dla firm usługowych, takich jak zakład fryzjerski czy mechaniczny. W trakcie implementacji założonych funkcjonalności skupiono się na odpowiedniej prezentacji możliwości panelu administracyjnego mając jednak na uwadze również złożoność funkcji.

Przedstawiona aplikacja została wykonana na platformie .NET z użyciem języka C# w wersji 5.0. Technologia ta jest rozwijana przez firmę Microsoft od roku XXX i wciąż zyskuje na popularności. Źródła tego faktu należy szukać w konswekwentnej obniżce cen rozwiązań serwerowych i rozwojowi darmowych bibliotek.

W celu poprawnej implementacji przyjęto wzorzec MVC (ang. Model-View-Controller). Dzięki zastosowaniu tego podejścia znacznie skrócono zarówno czas potrzebny na implementacje jak i potencjalną ilość błędów. Oddzielenie warstw, zgodne z zasadami przyjętymi w wytycznych wzorca MVC, poskutkowało skalowalnym i elastycznym kodem.

# Cel i zakres pracy

Celem pracy jest przedstawienie procesu projektowania i rozwóju aplikacji do tworzenia i administrowania stron internetowych przez personel nietechniczny. Przedstawiono szczegółowo wzorce projektowe oraz technologie, które zostały wykorzystane do rozwoju tego projektu. Zaprezentowano również korzyści płynące, z wyboru konkretnych narzędzi, które zauważono podczas procesu wdrażania aplikacji. Zwrócono też uwagę na korzyści, które odniósł użytkownik końcowy, który w naszym przypadku jest klient firmy usługowej. W poniższej pracy opisane są moduły i warstwy, które zostały zaimplementowane w celu realizacji zadania. Podczas tego procesu trzymano się ściśle wytycznych, które są zamieszczone przez firmę Microsoft w dokumentacji technicznej języka C# jak i tych, które firma Google zamieszcza odnośnie frameworku AngularJS.

W pracy zostaną przedstawione narzędzia i technologie, które zostały wykorzystane do stworzenia części logiki biznesowej aplikacji ServiceCMS. Opis ten będzie dotyczył zarówno panelu administracyjnego, jak i klienckiego. Narzędzia, które zostaną opisane, to platforma .NET, Enity Framework, Autofac, RestSharp oraz język C#.

Idea projektu zakładała stworzenie programu, który będzie przyjazny użytkownikomi nietechnicznemu. Z uwagi na to, konieczna była znajomość zarówno technologii programistycznych, używanych do implementacji logiki, jak i znajomość narzędzi, które odpowiadają za wykonanie estetcznej warstwy prezentacji. W celu spełnienia tych wymagań zdecydowano, że praca zostanie podzielona pomiędzy Artura Stelmacha i Kamila Ślusarczyka. Poniższa praca opisuje część logiki biznesowej. Kompletny opis warstwy prezentacji został przedstawiony w pracy pod tytułem Projekt i implementacja systemu budowania i administrowania stronami internetowymi przez personel nietechniczny – warstwa prezentacji, napisanej przez Artura Stelmacha.

Praca jest podzielona na trzy części. Pierwsza część jest częścią teoretyczną, opisującą historię, założenia projektu oraz narzędzia wykorzystane do jego implementacji. W części tej zostaną również poruszone wzorce programistyczne oraz zostanie opisany szereg faktów, które doprowadziły do wyboru konkretnych rozwiązań.

Kolejna część jest częścią praktyczną, która ukazuje szczegóły implementacyjne i opisuje podejście do problemów, które napotkano podczas fazy wdrażania funkcjonalności. Opis będzie podzielony zgodnie z modułową budową aplikacji i przedstawi zarówno część kodu źródłowego, jak i schematy graficzne projektu.

Podsumowanie pracy porówna założenia projektowe z wynikami, które osiągnięto. Zwrócono w nim uwagę na to, w jakim stopniu narzędzia i technologie spełniły wymagania projektowe. Opisano również przewagi, które zauważono podczas korzystania z określonych narzędzi, a które nie były znane podczas fazy projektowej.

Zakres pracy obejmuje X rozdziałów:

1. Wprowadzenie – opis historii

2. Cel i zakres pracy – ogólny opis celów, których realizację przyjęto na etapie fazy projektowej. Opisano tu również zakres poniższej pracy.

3.Narzędzia wykorzystane w projekcie – szczegółowy opis narzędzi i technologii, których użyto do implementacji wymagań funkcjonalnych systemu.

4. Projekt systemu – opis przypadków użycia oraz wymagań funkcjonalnych i niefunkcjonalnych

5. Prezentacja systemu – przedstawienie graficznego interfejsu użytkownika i ścieżek wykonania zaimplementowanych funkcjonalności

6. Budowa systemu – opis wybranych problemów programistycznych i przedstawienie sposobu ich rozwiązania za pomocą języka C# oraz JavaScript

7. Podsumowanie – porównanie przyjętych założeń z uzyskanymi wynikami. Zwrócono uwagę na wartość edukacyjną oraz rynkową stworzonego systemu.

# Narzędzia wykorzystane w projekcie

W obecnych czasach istnieje wiele narzędzi i technologii, które w bezproblemowy sposób umożliwiają rozwój aplikacji internetowych. Wybór jest najczęściej podyktowany ceną i dostępnością danego rozwiązania. Zdarzają się również przypadki, w których specyfika systemu (wydajność, dostępność) dyktuje wybór, który należy podjąć.

Do realizacji przedstawionej aplikacji użyto technologie firmy Microsoft. Narzędzia te były jeszcze stosunkowo niedawno wykorzystywane głównie do rozwoju dużych aplikacji biznesowych. Aktualnie ich popularność rośnie również w odniesieniu do małych i średnich projektów, co jest bezpośrednią pochodną obniżenia cen rozwiązań serwerowych jak i bezpłatnemu środowisku pracy Visual Studio.

W celu przedstawienia części logiki biznesowej zaprezentowano technologie związane z językiem C# i platformą .NET. Umożliwiły one implementacje zaprojektowanych funkcjonalności w sposób bezproblemowy i elastyczny, spełniając jednocześnie kryteria dotyczące bezpieczeństwa i wydajności.

## 3.1. Serwer

Założono, że aplikacja będzie rozwijana w środowisku lokalnym. Aby spełnić to wymaganie należało uruchomić serwer lokalny. W związku z faktem, że system był rozwijany w środowisku programistycznym Visual Studio, naturalnym wyborem był program IIS Express. Zawiera on bogaty zbiór usług internetowych. W projekcie wykorzystano powyższy program jedynie do roli serwera HTTP. Fakt wykorzystywania platformy .NET uniemożliwiał wykorzystanie innego rozwiązania.

## 3.2. Platforma .NET

Do implementacji systemu wykorzystano platformę .NET. Jest to technologia rozwijana przez firmę Microsoft. W jej skład wchodzi środowisko uruchomieniowe CLR (ang. Common Language Runtime) oraz bogaty zbiór bibliotek klas, które umożliwiają realizację aplikacji. Platforma .NET umożliwia programowanie w wielu językach (m.in. C#, VB, C++). Powyższy projekt zrealizowano w języku C# (szerzej opisany w pkt. 3.3.). Platforma .NET odpowiada za kod źródłowy i jego kompilację, oraz sposób alokacji pamięci.

Platformę .NET można wykorzystać do rozwoju aplikacji internetowych, desktopowych oraz systemów wbudowanych. Opisywany system jest systemem opartym o środowisko sieciowe. W skład opisywanej platformy zalicza się kompilatory języków wysokiego poziomu (C#, VB, C++) oraz kompilatory JIT i AoT (CIL do kodu maszynowego). Kompilatory platformy .NET nie są tematem tej pracy i z tego względu nie będą dokładnie opisane.

## 3.3. Język C#

Do wykonania aplikacji zastosowano język C#. Jest to silnie typowany, obiektowy język programowania rozwijany przez firmę Microsoft od 2000 roku. Jest jednym z języków, zaprojektowanych na CLR. WHAAAT? Umożliwia on budowę aplikacji z zachowaniem wydzielenia warstw abstrakcyjnych, co jest zgodne z paradygmatem programowania obiektowego. Dzięki zastosowaniu wzorców projektowcyh (patrz punkt 3.9.) możliwa była implementacja funkcjonalności zgodnie z zasadami SOLID. Wzorce umożliwiły również eliminację twardych zależności (patrz rozdział ….).

Na etapie projektu założono, że aplikacja będzie realizowana w metodologii code-first. Polega ona na implementacji klas języja C# i konwersji ich na tabele w bazie danych. W celu przeprowadzenia tej konwersji zastosowano Enity Framework, który jest narzędziem typu ORM (Object Relational Mapping). Jest to najpopularniejszy mapper dla platformy .NET dostępny na rynku. Alternatywą była biblioteka Nhibernate, która umożliwia bardzo podobny zbiór funkcjonalności, co EF. Wybór w tym przypadku był podyktowany świetnym wsparciem technologii EF, w porównaniu do Nhibernate.

## 3.4. LINQ

Aplikacja ServiceCMS zależy w dużym stopniu od bazodanowych operacji CRUD (ang. Create,Read,Update,Delete). Z racji tego, niezbędne było opracowanie procedury postępowania z kolekcjami danych. Zwrócono uwagę szczególnie na kwestie wydajności oraz bezawaryjności. Oczywistym wyborem w tym przypadku jest rozszrzenie LINQ (ang. Language Integrated Query). Jest to biblioteka zawarta w platformie .NET, która zawiera zbiór metod i rozszerzeń klas, przeznaczonych do operacji na kolekcjach. Dzięki wbudowanym mechanizmom optymalizacyjnym (np. Lazy Loading), spełnia ona założone normy wydajnościowe. Biblioteka LINQ jest w opisywanym przypadku rozwiązaniem bezkonkurencyjnym, przez co nie analizowano innych możliwych rozwiązań.

## 3.5. Kontrola wersji

Na etapie projektu podjęto decyzję o wykonaniu aplikacji w dwuosobowym zespole programistycznym. Z uwagi na to, niezbędna była organizacja pracy za pomocą systemu kontroli wersji (ang. Subversion Control.). Do przechowywania plików źródłowych wykorzystano serwis Github. Jest to najpopularniejsza platforma do obsługi wersji projektów informatycznych. Jako klienta wykorzystano aplikację SourceTree. Umożliwia ona zatwierdzanie zmian (ang. Commit) oraz powrót do wcześniejszych wersji (ang. Rollback). Przy doborze narzędzia rozważano wykorzystanie innych klientów wersjonowania plików. Były to między innymi TFS oraz konsola wersja Git. Z uwagi na brak istotnych różnic pomiędzy tymi rozwiązaniami, wybrano darmowe i proste narzędzie, które spełniło wszystkie założenia, przyjęte na etapie projektu.

## 3.6. Producteev

Podczas procesu implementacji niezbędny był podział pracy pomiędzy obu autorów. Przed fazą wdrożenia funkcjonalności, szczegółowo zaplanowano za które moduły będą odpowiedzialni poszczególni autorzy. Formalny podział został wykonany przy pomocy programu o nazwie Producteev. Służy on do przydzielania zadań osobom, które biorą udział w projekcie. Zadania posiadają statusy, które można zmieniać w zależności od procentowej wartości wykonania zadania. Możliwe jest również dodanie zadań podrzędnych (ang. Substask), które ułatwiają organizację przepływu informacji. Powyższe funkcjonalności uznano za wystarczająco do organizacji dwuosobowego zespołu programistycznego. Atutem Producteev jest również darmowa licencja, dzięki czemu nie wygenerowano dodatkowych kosztów.

## 3.7. RestSharp

RestSharp to darmowa biblioteka dla platformy .NET. Umożliwia ona wykonanie żądań REST do zewnętrznych serwerów w łatwy i elastyczny sposób. Zastosowanie tej biblioteki umożliwiło wykonanie żądań HTTP oraz deserializacje uzyskanych wyników bez konieczności samodzielnej implementacji. W opisywanym systemie jednym z wymagań było przechwycenie odpowiedzi w formacie JSON. RestSharp umożliwia realizację tego wymagania, poprzez implementację serializatora JSON. Dzięki bogatej i przejrzystej dokumentacji korzystanie z tej biblioteki nie powodowało jakichkolwiek problemów. Nie rozważano użycia innego rozwiązania – moduł RestSharp jest najpopularniejszym tego typu dodatkiem dla platformy .NET.

## 3.8. Wzorce projektowe

Poniżej opisano wzorce projektowe, które zostały użyte podczas implementacji systemu ServiceCMS.

## 3.8.1. Unit of work

Unit of work – wzorzec przeznaczony do ułatwienia procesu dostępu do repozytorium. Poprzez implementację powyższego wzorca wszystkie instancje generycznego repozytorium są zaimplementowane w jednej klasie.Wzorzec ten wprowadza również transakcje bazodanowe. Z uwagi na to, że ServiceCMS jest systemem w dużej mierze opierającym się na dostępie do warstwy bazodanowej, było to spore ułatwienie.

## 3.8.2. Wstrzykiwanie zależności

Wstrzykiwanie zależności – wzorzec architektoniczny, który umożliwia eliminację twardych powiązań pomiędzy komponentami i umieszczenie w ich miejsce powiązań luźnych (ang. Loose coupling). Zastosowanie tego wzorca radykalnie obnizyło czas, potrzebny na refaktoryzację kodu źródłowego. Dzięki implementacja wstrzykiwania zależności za pomocą biblioteki Autofac ograniczono zmianę sposobu wywoływania i sygnatur funkcji w przypadku zmiany wymagań funkcjonalnych. Autofac umożliwił również przeniesienie odpowiedzialności za tworzenie obiektów i ich łączenie do obiektów fabryki.

## 3.8.3. Repozytorium

Repozytorium – warstwa architektoniczna, której zadaniem jest odizolowanie warstwy danych od warstwy logiki biznesowej. Repozytorium działa na poziomie jednej klasy modelu. Dzięki odesparowaniu warstw uzyskano system, którego debugowanie podczas fazy testów było względnie bezproblemowe.

## 3.8.4. MVC

MVC – wzorzec architektoniczny, który organizuje strukturę projektów informatycznych. Został on zaprojektowany w 1979 roku. Wzorzec ten zakłada podział systemu na trzy warstwy:

-warstwa Modelu (odpowiedzialna za reprezentację logiki biznesowej aplikacji)

-warstwa widoku (sposób prezentacji wyglądu systemu i umożliwienie interakcji z użytkownikiem)

-warstwa kontrolera (otrzymuje żądania z warstwy widoku i wykonuje metody, związane z logiką aplikacji).

Powyższe warstwy są ze sobą ściśle powiązane. Podczas implementacji ściśle trzymano się zasad czystego kodu, co poskutkowało zmniejszeniem roli kontrolerów na rzecz wyizolowanych metod, związanych z logiką biznesową. Wzorzec MVC w ServiceCMS spełnił swoją funkcję i uprościł implementację oraz testowanie wykonywanego systemu.

## 3.9. Visual Studio

Do realizacji systemu wymagany było środowisko programistyczne. Naturalnym wyborem dla języka C# i platformy .NET jest Visual Studio. W opisywanym projekcie korzystano z wersji Community, wydanej w 2013 roku. Wybór wersji był uzależniony od postanowień licencyjnych firmy Microsoft. Środowisko programistyczne Visual Studio wraz z dodatkiem Resharper znacznie przyczyniło się do ułatwienia pracy z kodem źródłowym aplikacji. Wielokrotnie korzystano z wbudowanego debuggera, który śledził występujące błędy. Usługa Nuget umożliwiła korzystanie z zewnętrznych bibliotek (RESTSharp,Autofac). Z uwagi na wykorzystywany system operacyjny i charakter projektu niemożliwe było skorzystanie z innego rozwiązania.

## 3.10. Autofac

Autofac to kontener wstrzykiwania zależności (ang. Inversion of Control Container). Jest on udostępniany zgodnie z licencją open-source. W opisywanym systemie został wykorzystany do implementacji wzorca projektowego IoC. Autofac jest z założenia biblioteką pozbawioną zbędnych funkcjonalności. Rezultatem takiego podejścia jest stosunkowo prosty schemat implementacji wstrzykiwania zależności. Warto zauważyć, że Autofac realizuje procesy zarządzania czasem życia obiektów i rozwiązywania zależności bez udziału programisty. Dodatkowym atutem tej biblioteki jest bogata dokumentacja, która wyczerpująco opisuje wszelkie problemy, które można napotkać podczas korzystania z niej.

# Projekt systemu współnie z Arturem Stelmachem

Zgodnie z założeniami, przyjętymi na etapie projektu, zdecydowano o wspólnej realizacji aplikacji. Service CMS został zrealizowany wspólnie przez Kamila Ślusarczyka oraz Artura Stelmacha. Poniższy rozdział jest wspólny dla obu prac – zarówno powyższej, jak i pracy Artura Stelmacha pt. Projekt i implementacja systemu budowania i administrowania stronami internetowymi przez personel nietechniczny – warstwa prezentacji. Podczas implementacji ściśle trzymano się modułowego podziału systemu, który był podstawą do podziału pomiędzy autorów.

# 4.1. Wymagania biznesowe

ServiceCMS to aplikacja internetowa, dedykowana dla wszelakich firm usługowych. Aplikacja ma za zadanie wspomaganie w tworzeniu strony internetowej firmy. Najważniejszą funkcjonalnością aplikacji jest zautomatyzowanie rejestracji klientów na zdefiniowane wcześniej usługi. Ponadto aplikacja umożliwia zarządzanie treściami, które są reprezentowane przez strony i aktualności. Umożliwia również zarządzanie plikami i udostępnia możliwość przeglądania statystyk odwiedzin panelu klienckiego. Oprogramowanie może obsługiwać osoba nietechniczna z podstawową znajomością obsługi komputera.

## Słownik pojęć

**Administrator** – osoba wyznaczona do zarządzania treścią, która jest wyświetlana w panelu klienckim.

**Klient** – osoba odwiedzająca panel kliencki

**Panel administracyjny** – strona internetowa umożliwiająca administratorowi zarządzanie treścią

**Panel kliencki** – strona internetowa która jest wynikiem ustawień ustalonych w panelu administracyjnym

**Aktualność** – prosty typ treści, wyświetlany na głównej stronie panelu klienckiego

**Strona** – typ treści który może zawierać wiele elementów takich jak ilustracje i załączniki w postaci plików. Do strony może prowadzić link w postaci przycisku z menu głównego.

**Dostawca usługi** – byt związany z typem usługi. Może to być osoba, stanowisko lub narzędzie wykonujące usługę.

**Typ usługi** – opis usługi wykonywanej w firmie. Każdy typ może posiadać wiele faz usługi.

**Popup** – wiadomość wyświetlana klientowi przy pierwszej wizycie w panelu klienckim.

**Odbiorca newslettera** – adres email na który może być wysłany newsletter

## 4.2. Wymagania funkcjonalne

### 4.2.1 Specyfikacja wymagań dla panelu administratora

* Dodawanie, usuwanie i edytowanie aktualności
* Dodawanie, usuwanie i edytowanie kategorii aktualności
* Dodawanie, usuwanie i edytowanie artykułów/stron
* Dodawanie, usuwanie i edytowanie usług zarejestrowanych przez klienta
* Dodawanie, usuwanie i edytowanie aktualności
* Dodawanie, usuwanie i edytowanie dostawców usług
* Dodawanie, usuwanie i edytowanie typów usług
* Możliwość zarządzania przyciskami menu
* Dodawanie, usuwanie i edytowanie odbiorców newslettera
* Możliwość kompozycji i wysyłania newslettera
* Dodawanie, usuwanie plików
* Dodawanie, usuwanie i edytowanie popupów
* Możliwość tylko jednego aktywnego popupu
* Możliwość przeglądania statystyk odwiedzin dotyczących panelu klienckiego
* Możliwość edytowania ustawień
* Możliwość zmiany języka na polski lub angielski

### 4.2.2. Specyfikacja wymagań dla panelu klienckiego

* Wyświetlanie aktualności
* Wyświetlanie stron
* Nawigowanie za pomocą menu
* Rejestracja na usługę
* Wyświetlanie popupu przy pierwszej wizycie
* Zbieranie informacji o odwiedzinach

### 4.2.3. Wymagania niefunkcjonalne

* Aplikacja powinna mieć spójny i przejrzysty interfejs użytkownika
* Panel administracyjny powinien być w języku polskim jak i angielskim
* Zarówno panel kliencki jak i panel administracyjny powinien być poprawnie wyświetlany na rozdzielczościach nie mniejszych niż 1600x900
* Aplikacja powinna działać na najnowszych wersjach przeglądarek Chrome, Firefox i Opera

## Warstwy aplikacji

Aplikacja została podzielona na 4 warstwy : rdzenia, modułów, logiki biznesowej i prezentacji. Dzielenie aplikacji miało na celu uporządkowanie struktury aplikacji i zmniejszenie redundancji kodu. Poniżej przedstawiono każdą z warstw:

1. **Rdzeń** – warstwa odpowiadająca za połączenie i obsługę bazy danych, obsługę i logowanie wyjątków oraz zawierająca klasy wykorzystywane w całej aplikacji.
2. **Logika biznesowa** – warstwa zawierająca implementacje algorytmów logiki biznesowej takiej jak zapisywanie, edytowanie i usuwanie modeli wykorzystywanych w aplikacji.
3. **Moduły** – warstwa zawierająca wymienialne rozwiązania niezależne od reszty aplikacji np. obsługa zapisu i usuwania plików, łączenie z zewnętrznym serwisem lub obsługa kryptografii haseł.
4. **Prezentacja** – warstwa odpowiedzialna za interakcje z użytkownikiem aplikacji, zawiera wszystkie widoki wyświetlane użytkownikowi oraz akcje obsługujące działania użytkownik

## Projekt bazy danych

Aplikacja wykorzystuje relacyjną bazę danych SQL wygenerowaną przy pomocy Entity Framework zgodnie z metodologią CodeFirst. Każda z tabel bazy posiada klucz główny który zapewnia możliwość łączenia ze sobą dwóch lub więcej tabel. Baza danych została znormalizowana do 3 postaci normalnej, aby zapobiec rozrastaniu i nadmiarowości danych. W celu przyspieszenia przeszukiwania tabel zostały nałożone indeksy na każdy klucz główny.

Poniżej przedstawiono najważniejsze tabele i ich relacje:

**ServiceTypes –** tabela odwzorowująca typ usługi. Oprócz klucza głównego „Id” posiada kolumnę „Name” przechowującą nazwę danego typu usługi. Jest połączona relacją jeden do wielu z tabelą **ServicePhases** (typ usługi ma wiele faz).

**ServicePhases –** tabela przedstawiająca fazy usług. Posiada kolumnę „Order” które przechowuje informację o kolejności wykonywania. Tabela przechowuje również informacje o czasie trwania danej fazy (kolumna „DurationInMinutes”) oraz czasie opóźnienia (kolumna „DelayInMinutes”). Posiada również klucz obcy tabeli ServiceType, umożliwiający relację typu „jeden do wielu”.

**ServiceProviders –** przechowuje informacje o dostawcach usług. Jest połączona z tabelą **ServiceTypes** relacją wiele do wielu (dostawca usługi może wykonywać wiele usług a usługi mogą być wykonywane przez wielu dostawców) przy pomocy dodatkowej tabeli **ServiceProviderServiceTypes.**

**RegistratedServices –** odwzorowuje usługi zarejestrowane przez klientów. Połączona jest z tabelą **ServiceTypes** relacją jeden do wielu (typ usługi może być w wielu zarejestrowanych usługach) oraz z tabelą **ServiceProviders** również relacją jeden do wielu. Oprócz tego przechowuje podstawowe informacje o kliencie który zarejestrował się na usługę.

**Pages –** przedstawia strony edytowane przez administratora. Przy pomocy tabeli **PageFiles** tworzy relację wiele do wielu z tabelą **Files.** Kolumna „RestorePageId” odpowiada za połączenie tej tabeli z samą sobą. Dzięki temu baza danych umożliwia przechowywania historii modyfikacji danych stron.

**MenuButtons –** tabela przedstawiająca przyciski menu. Łączy się sama ze sobą dzięki czemu uzyskujemy strukturę drzewiastą która doskonale odwzorowuje menu. Oprócz tego posiada połączenie jeden do wielu z tabelą **Pages** co pozwala na nawigowanie pomiędzy stronami dzięki stworzonemu menu.

**News –** tabela podobna do tabeli **Pages**. Posiada połączenie wewnętrzne, co umożliwia przechowywanie historii. Oprócz tego za pośrednictwem tabeli **NewsNewsCategories** realizowana jest relacja wiele do wielu z tabelą **NewsCategories** która pozwala na filtrowanie aktualności za pomocą kategorii.

**Settings** – tabela przechowująca informacje o ustawieniach aplikacji. Posiada pole „Value” które może przechowywać dowolną wartość możliwą do zapisania oraz pole InputType dzięki któremu wiemy jaki typ kontrolki należy wyświetlić do edytowania danego ustawienia

Poniżej przedstawiono pełny diagram bazy danych:

## Diagramy sekwencji

Poniżej przedstawiono diagramy sekwencji najważniejszych akcji podejmowanych przez użytkownika.

### Rejestracja na usługę

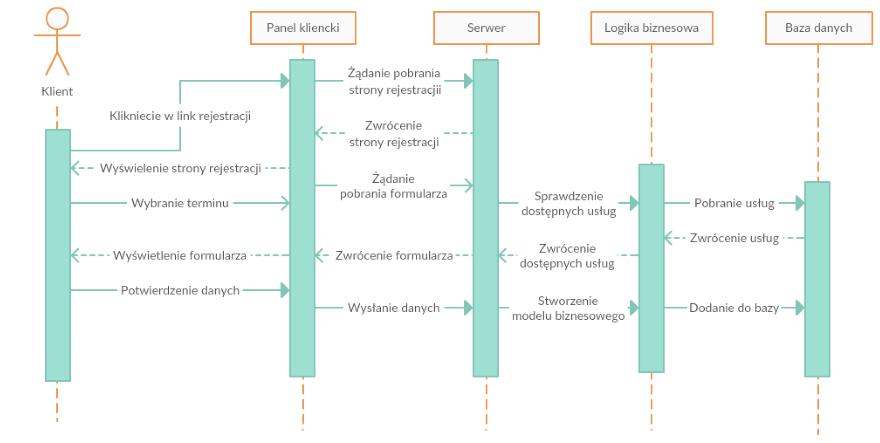
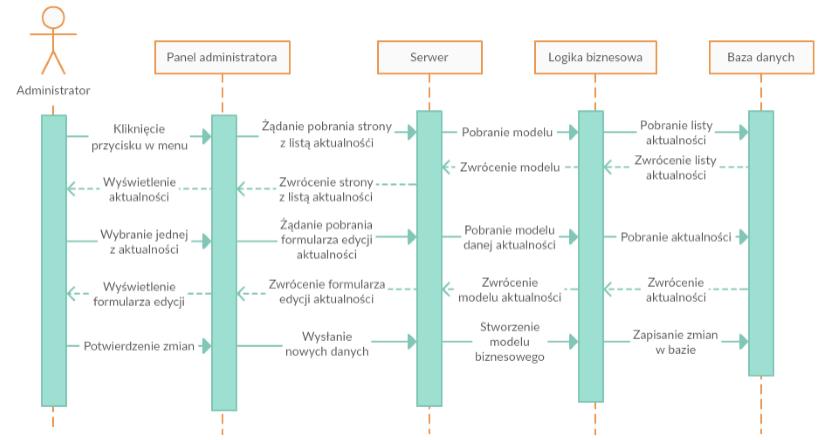


Diagram przedstawia podstawową funkcjonalność aplikacji, jaką jest rejestracja klienta na

usługę. Użytkownik poprzez kliknięcie w odpowiedni link sygnalizuję chęć rejestracji. Wyświetlana jest strona z kalendarzem i dostępnymi terminami usług. Użytkownik wybiera odpowiadający mu termin. Następnie wysyłane jest żądanie pobrania formularza z dostępnymi w tym terminie usługami. Kolejnym krokiem jest potwierdzenie przez użytkownika danych i zarejestrowanie usługi w bazie danych.

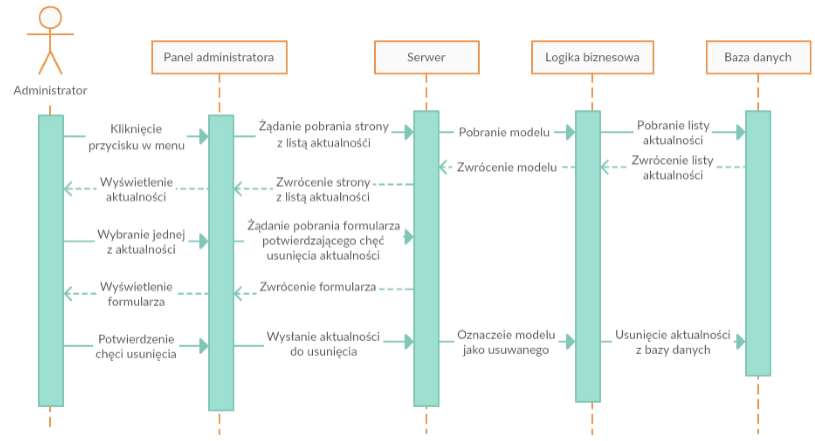
### Edycja aktualności



Ze względu na podobieństwo wszystkich funkcjonalności edycji i dodawania, aktualności, stron itd. ograniczono się do przedstawienia jednego przykładu w postaci edycji aktualności.

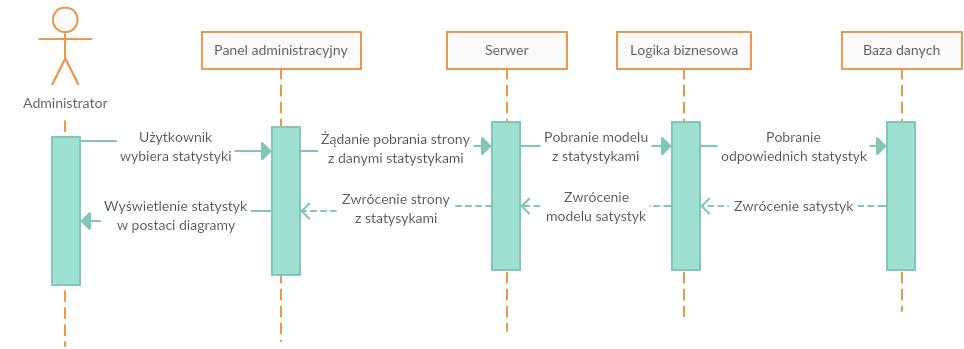
Użytkownik poprzez menu wybiera typ danych który chce edytować. Wyświetlana jest strona z listą wybranych wcześniej encji. Użytkownik wybiera jedną z encji którą chce edytować i wyświetlany jest mu formularz edycji z danymi. Następnie użytkownik potwierdza dokonanie zmian co skutkuje zapisaniem danych w bazie.

### Usuwanie aktualności



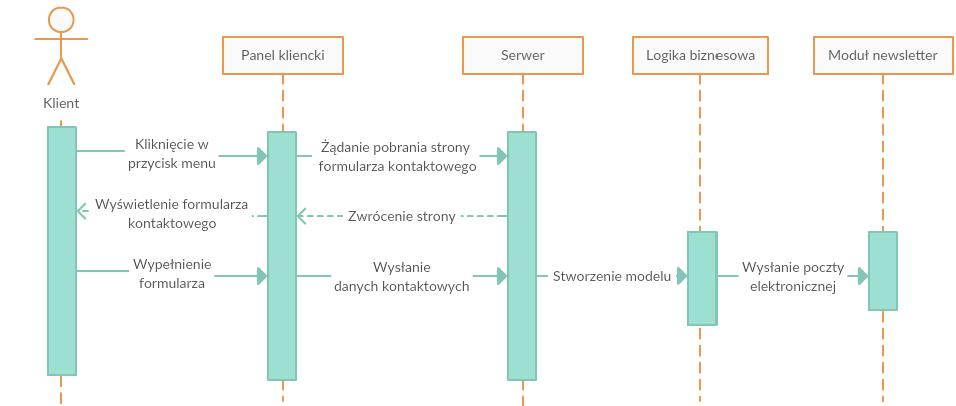
Usuwanie każdego typu danych jest realizowane w podobny sposób. Tak jak w przypadku edycji czy usuwania użytkownik wybiera typ danych do usunięcia. Następnie wyświetlana jest strona z listą encji. Użytkownik wybiera encję którą chce usunąć, wyświetlany jest mu formularz który pozwala na potwierdzenie chęci usunięcia. Po potwierdzeniu wysyłane jest żądanie usunięcie aktualności, która jest usuwana z bazy danych.

### Wyświetlenie statystyk



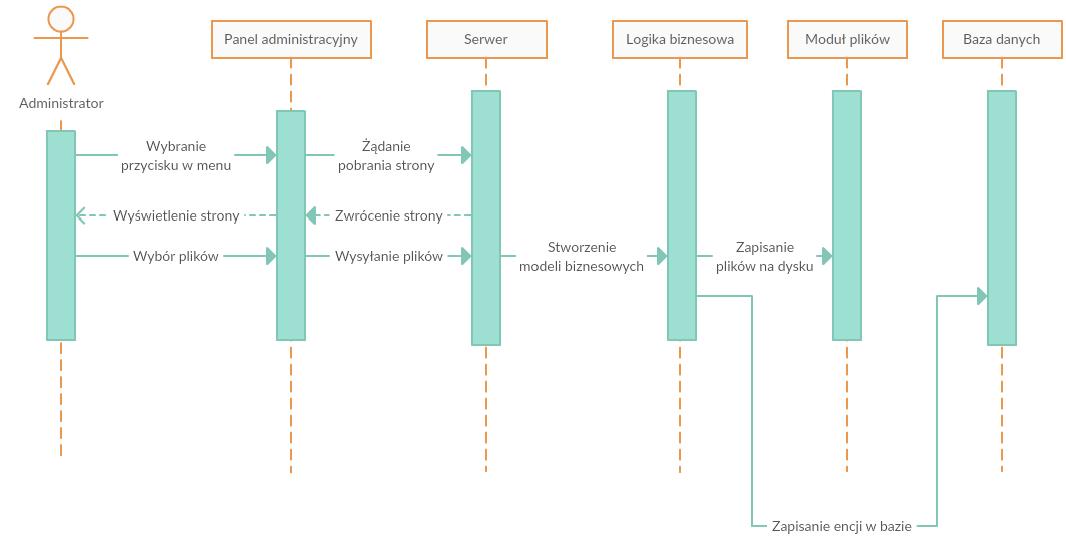
Administrator sygnalizuje chęć przeglądania statystyk odwiedzin, poprzez wybranie w menu odpowiedniego przycisku. Wysyłane jest żądanie pobrania strony z statystykami, następnie serwer pobiera odpowiednie dane z bazy danych i wyświetlana jest strona z danymi przedstawionymi w postaci odpowiedniego diagramu.

### Skorzystanie z formularza kontaktowego



Klient wybiera w menu odpowiedni przycisk. Wyświetlana jest mu strona z formularzem kontaktowym. Następnie klient wypełnia formularz danymi i potwierdza wysyłanie wiadomości. System przekazuje dane do modułu newslettera który tworzy i wysyła wiadomość poprzez serwer SMTP.

### Przekazanie plików na serwer



Administrator wybiera odpowiedni przycisk w menu. Następnie wybiera pliki które chce przekazać na serwer i potwierdza ten wybór. Pliki są zapisywane na dysku oraz tworzone są encje opisujące te pliki do zapisania w bazie danych .

# Prezentacja systemu

## 5.1. Ekran logowania

Poniżej zaprezentowano ekran logowania do panelu administracyjnego. Korzysta on ze standardowych rozwiązań, przeznaczonych dla aplikacji internetowych. Możliwe jest zaznaczenie checkbox-a w celu zapamiętania nazwy użytkownika i hasła.



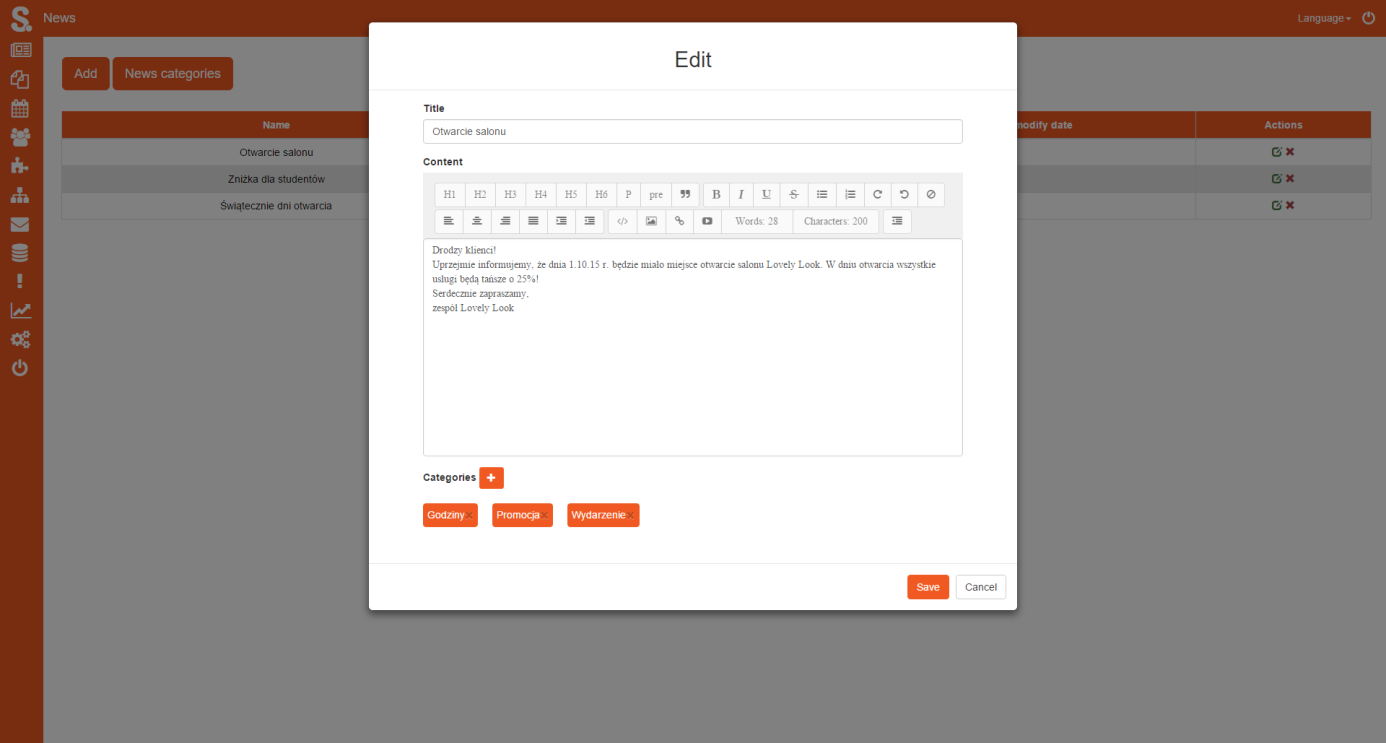
Poniżej zaprezentowano pulpit aplikacji administracyjnej. Jest on ekranem powitalnym, wyświetlanym zaraz po zalogowaniu się do systemu przez administratora. Pulpit ten jest zrealizowany z pomocą technologii Bootstrap. W trakcie implementacji zwrócono szczególną uwagę na estetyczność i łatwość obsługi, czego dowodem jest poniższa ilustracja.



Poniżej zaprezentowano menu boczne, które ukazuje się po najechaniu na ikony, zamieszczone na pulpicie administracyjnym (patrz pkt…). Ikony są spójne i zgodne z nowoczesnymi standardami projektowania interfejsu użytkownika. Dzięki wykorzystaniu ikon z serwisu *FontAwesome*możliwe było intuicyjne przedstawienie kolejnych funkcjonalności administratorowi.



Poniżej zamieszczono widok edycyjny encji News (ang. Aktualność). Ilustracja poniżej przedstawia możliwość zmiany tytułu, zawartości (łącznie z panelem edycji) oraz możliwość dodania kategorii aktualności za pomocą sekcji, umieszczonej u dołu formularza.



Poniżej zamieszczono ilustrację, przedstawiającą moduł obsługi plików. ServiceCMS umożliwia wrzucanie na serwer plików o dowolnych rozszerzeniach. Ich wielkość jest domyślnie ograniczona do 50 mB. Należy jednak zauważyć, że można zwiększyć tę wartość do maksymalnie 2 GB. Poniższa ilustracja ukazuje zakończony proces przesyłania plików, co ukazuje niebieski pasek postępu.



Poniżej przedstawiono widok edycji określonej strony. Podobnie jak w przypadku modyfikacji encji News, możliwa jest edycja tytułu strony i jej zawartości. Do strony można również załączyć pliki, umieszczone wcześniej na serwerze, za pomocą aplikacji Files.



Poniżej przedstawiono widok edycji encji MenuButton, która odzwierciedla przycisk menu na stronie klienckiej. Ułożenie hierarchiczne, zaprezentowane poniżej wskazuje na strukturę drzewiastą menu. Stopień zagnieżdżania poszczególnych elementów jest zrealizowany w koncepcji Drag n’ Drop. Kolejne elementy menu można usuwać, modyfikować bądź zmieniać ich kolejność.Krytyczną funkcjonalnością modułu MenuButton jest możliwość podpinania do nich encji Page w relacji jeden do wielu.



Poniżej przedstawiono odwzorowanie struktury przycisków menu po stronie klienckiej.



Poniżej zaprezentowano widok zarządzania newsletterem. Załączono dwie ilustracje, przedstawiające kolejno: zarządzanie listą odbiorców oraz ekran edycji wysyłanej wiadomości e-mail. W oknie tworzenia wiadomości możliwe jest przypisanie poszczególnych odbiorców, bądź całej ich listy do zbioru odbiorców.

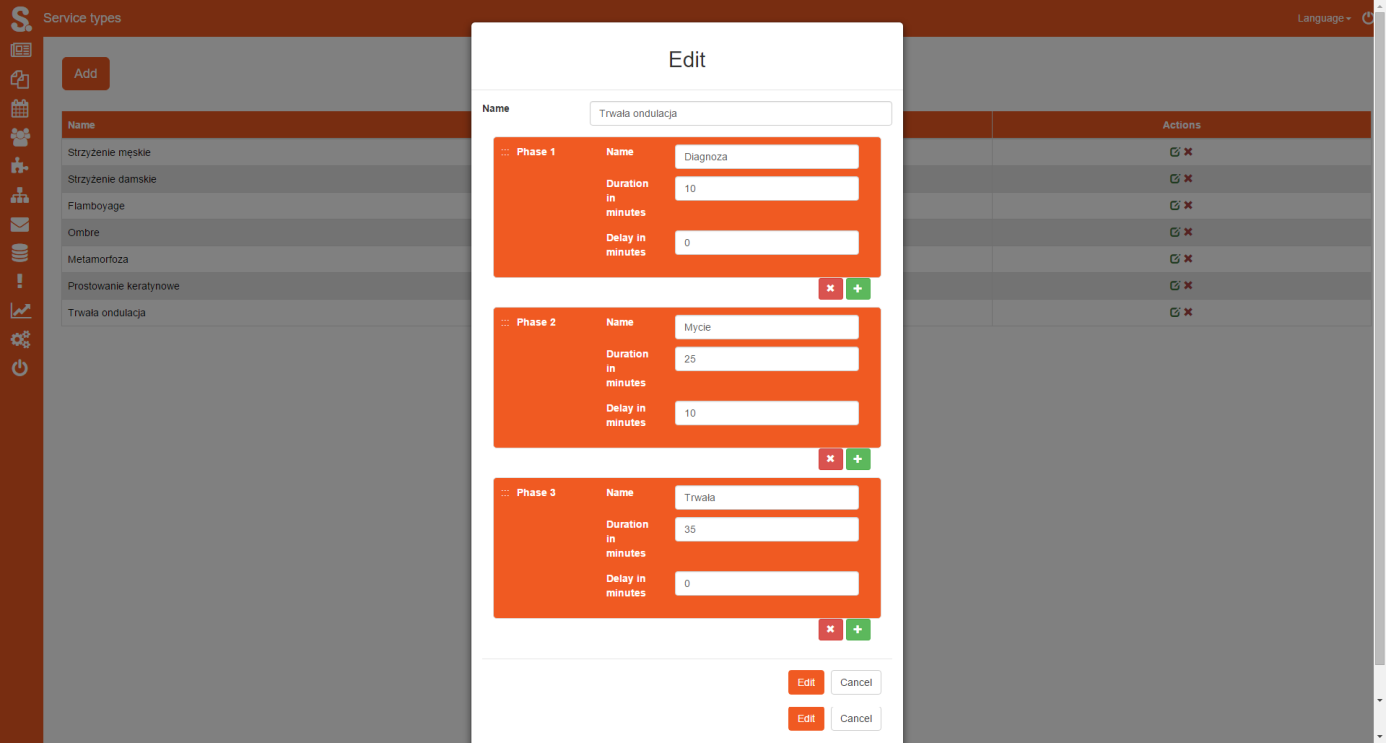




Poniżej zaprezentowano ekran edycji ustawień globalnych aplikacji ServiceCMS. Można w nim edytować takie właściwości systemu, jak: nazwa firmy, szczegóły konta e-mail, ustawienia paginacji aktualności oraz stan formularza kontaktowego, kalendarza usług i pop-up (aktywny/nieaktywny).



Poniżej zaprezentowano moduł zarządzania usługami. Istotna w przypadku tej funkcjonalności jest odwzorowana relacja bazodanowa – każda usługa może posiadać wiele faz. Faza ma określoną przez administratora nazwę, czas trwania i opóźnienie. Dzięki elastycznej implementacji dostępności usług możliwe jest zapisanie się na usługę podczas gdy inna osoba jest w trakcie trwania fazy z opóźnieniem. Fazy można reorganizować zgodnie z podejściem Drag n’ Drop.



Poniżej zaprezentowano panel edycji encji ServiceProvider (ang….). W poniższym formularzu można zmieniać nazwę oraz zakres usług dostępnych możliwych do wykonania przed określonego pracownika. Usługi definiuje się w module ServiceTypes. Po stronie klienckiej ma to wpływ na określenie możliwych wykonawców usług – do wybranego dostawcy usług (np. pracownik, stanowisko) można się zapisać na określoną usługę, tylko jeśli jest ona poprawnie zdefiniowana w panelu administracyjnym.



Poniżej zaprezentowano moduł Pop-up. Odpowiada on za jednorazowo wyświetlaną informację w panelu klienckim. Jego edycja jest spójna z procesem edycji aktualności, czy stron – za pomocą okna edycji można edytować tytuł oraz zawartość. W module Pop-up z założenia można uaktywnić jedną encję. Jest to odwzorowane w panelu klienckim za pomocą ekskluzywnego checkbox-a.



Poniżej zaprezentowano moduł Registrated Services. W panelu administratora widoczny jest lista zarejestrowanych usług. Widoczne są szczegóły usługi: czas utworzenia rejestracji, typ usługi, wybrany dostarczyciel usługi oraz dane klienta. Ostatnia informacja jest widoczna po najechaniu kursorem na ikonę klienta i prezentowana jest za pomocą tool-tipu. W module można również odwołać usługę.



Istotną funkcjonalnością w module Registrated Services jest potwierdzenie, które otrzymuje klient po zarejestrowaniu się na usługę. Jest to spersonalizowana wiadomość e-mail, która zostaje wysłana na adres e-mail, podany podczas procesu rejestracji.

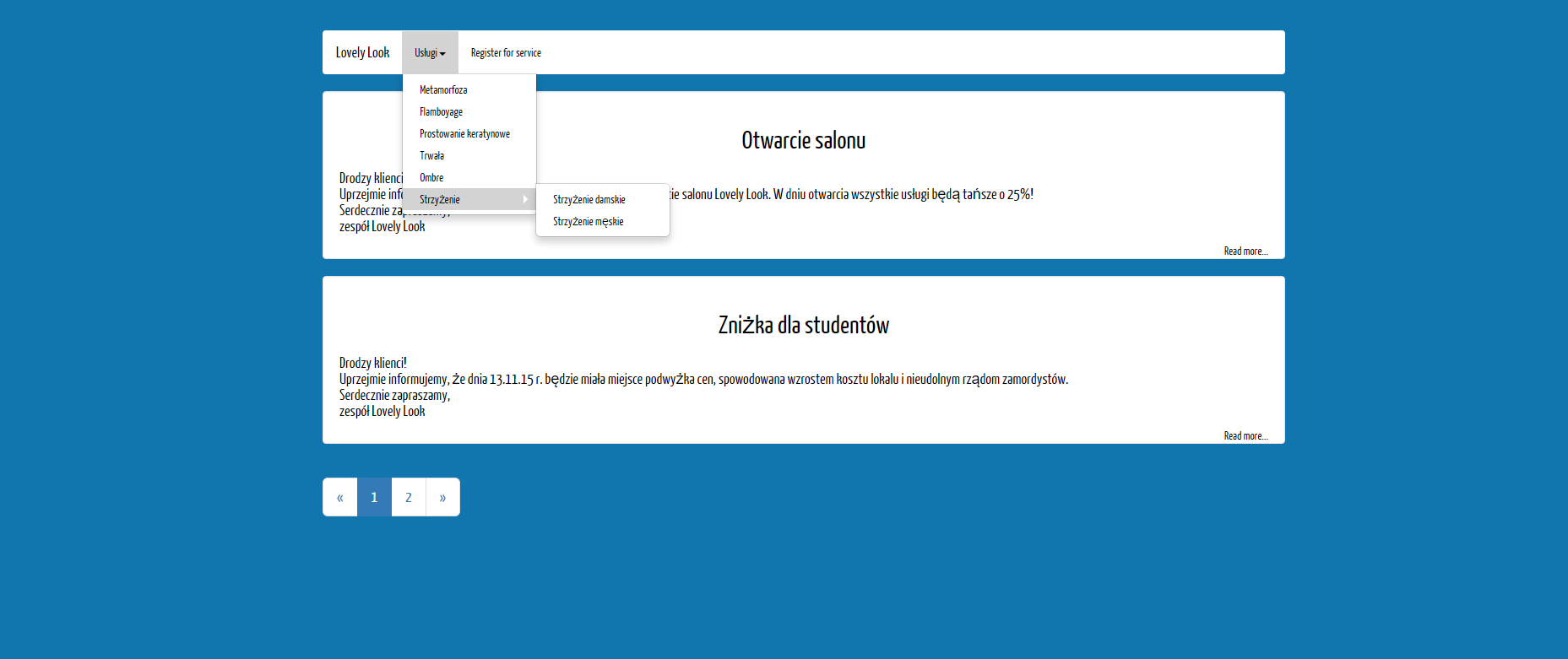


Poniżej zaprezentowano moduł Statistics. Generuje on statystyki, oparte o odwiedziny i rodzaj aktywności na stronie klienckiej. Jest to moduł wyjątkowo istotny, ponieważ realizuje on podstawowe założenia marketing automation. W celu doprecyzowania rodzaju i formy przedstawianej oferty administrator korzysta z danych, otrzymanych dzięki modułowi statystyk. Funkcjonalności pozwalają określić które strony są najczęściej wyświetlane w zależności od wybranych przedziałów czasowych. Administrator może określić również liczbę użytkowników portalu w zdefiniowanych dniach, miesiącach, czy latach. Ostatnią funkcjonalnością jest pobranie danych, dzielących użytkowników według krajów, z których nastąpiło żądanie HTTP.

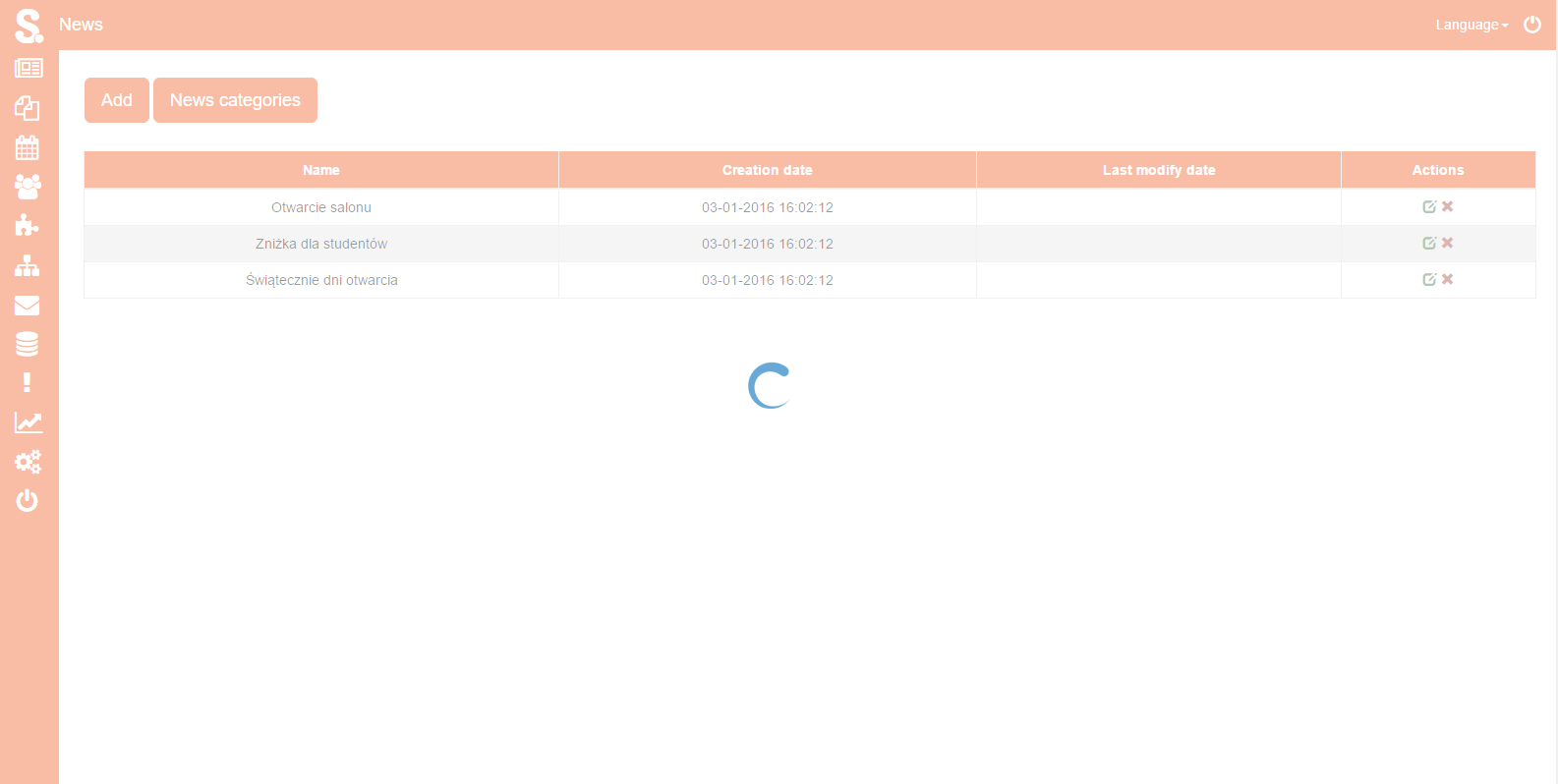




Poniżej zamieszczono ilustracje, przedstawiające panel kliencki aplikacji. Na ilustracji u góry widać rozwijalne menu z strukturą drzewiastą. Przyciski te prowadzą do stron, które zostały do nich podpięte w panelu administracyjnym. Aktualności, wprowadzone przez administratora są stronicowane zgodnie z ustawieniami, zapisanymi w module Settings.



Poniżej zamieszczono ekran ładowania, który jest wyświetlany w chwili pobierania danych z bazy danych. Proces ten wielokrotnie wymaga około 0.5-1 sekundy na realizację. Ekran ładowania, który jest przedstawiany administratorowi upewnia go, że system nie napotkał błędu i cały proces obsługi danych jest przeprowadzony zgodnie z oczekiwaniami.



# Budowa systemu

# CRUD Logic

Ze względu na duże podobieństwo operacji CRUD dla wszystkich encji, poniżej przedstawiono przykładową implementację dla encji User. Każda metoda opatrzona jest w blok try-catch, który gwarantuje odpowiednie zabezpieczenia tzw. information-flow. W przypadku nieprzewidzianego zachowania wyjątek zostanie zalogowany do pliku. Zostaje zwrócona również wiadomość o statusie operacji.



Metody o nazwie GetById oraz GetAll zawierają kod który realizuje pobieranie encji, bądź zbioru encji według określonych zasad. W przypadku braku encji o określonym kluczu głównym zwracany jest obiekt pusty. Podobnie w przypadku metody GetAll – tutaj zwracany jest lista, która może być pusta. Błędy są przechwytywane przez klasę Logger.

Operacja Insert umieszcza obiekt w bazie danych. Generyczne repozytorium przy udziale Entity Framework rzutuje obiekt języka C# na bazodanową krotkę. Umieszczenie w bazie danych następuje, gdy obiekt spełnia założenia, które zostały zdefiniowane przy jego tworzeniu - dba o to generyczne repozytorium.

Aktualizacja obiektu w bazie danych następuje, gdy obiekt spełnia założenia, które zostały zdefiniowane przy jego tworzeniu. Ponownie – operacja ta jest wykonywana przez instancję repozytorium generycznego.

Operacja usunięcia krotki z bazy danych jest zbieżna z metodami Insert oraz Update. Jeśli w bazie danych istnieje obiekt o przekazanej wartości klucza głównego generyczne repozytorium usuwa odpowiednią krotkę.

# Restore news



W celu impelementacji historii aktualności oraz stron napisano metodę powyższą metodę Update. Wbrew nazwie, metoda ta przeprowada proces umieszczenia w bazie danych nowego rekordu, który wskazuje na swojego poprzednika. Dzięki temu system pamięta wszelkie zmiany, które dokonano na encjach News oraz Page. Zależności bazodanowe zostały stworzone w postaci klucza obcego o nazwie RestoreNewsId. Metoda przypisuje wszystkie własności encji starszej do jej nowej wersji, łącznie z kategoriami aktualności.

# Restore Page



Metoda Update dla encji Page działa podobnie jak ta, zaimplementowana w punkcie… dla encji News. Tutaj również następuje umieszczenie nowego rekordu, wraz z przekazaniem do niego wszystkich własności obiektu pochodnego. W zaktualizowanej wersji strony przypisano również pliki, obecne w poprzedniej wersji.

# Gettery usług

Aplikacji ServiceCMS w dużej mierze opiera się na procesie rejestracji usług. Proces ten wymagał przeciążenia metody o nazwie GetAllServicesWithMatchingCriteria. Było to niezbędne do uporządkowania przepływu informacji w systemie.



Metoda GetAllServicesWithMatchingCriteria, która otrzymuje jako argument datę, zwraca listę usług, zarejestrowanych na określony czas.

Metoda GetAllServicesWithMatchingCriteria, która otrzymuje jako argument dostarczyciela, zwraca listę usług, wykonywanych przez określoną osobę, bądź na określonym stanowisku.

Metoda GetAllServicesWithMatchingCriteria, która otrzymuje jako argument dostarczyciela oraz datę, zwraca listę usług. Jej zawartość to część wspólna usług, wykonywanych przez określoną osobę, bądź na określonym stanowisku, w określonym czasie.

# Sprawdzanie dostępności usług



Metoda CheckAvailability sprawdza, czy dany typ usługi może być zarejestrowany na daną datę. Metoda wykorzystuje opisane wcześniej metody do sprawdzenia czy bloki czasowe się na siebie nie nakładają . Jeśli taka sytuacja zaistnieje, dany typ usługi jest oznaczany jako niemożliwy do wykonania.

# Moduł statystyk

Moduł statystyk jest szerzej opisany w rozdziale … Poniżej zostanie zaprezentowany kod, odpowiadający za realizację wymienionych tam funcjonalności. Moduł ten opiera się na filtrach - obiektach, zawartych w ASP.NET MVC. Umożliwiają one wywołanie zbioru metod podczas wykonywania żądań HTTP na określonych akcjach w określonych kontrolerach.



Klasa StatisticsFilter przechwytuje dane użytkownika, który korzysta z aplikacji klienckiej. Dane, które są przechowywane w obiekcie StatisticsInformationModel to adres IP, nazwa kontrolera i akcji, z jakiej korzystał, oraz data wykonania żądania.



Metoda AddEntry otrzymuje dane, przechwycone przez filtr statystyczny i umieszcza je w bazie, w celu późniejszego opracowania ich przez metody warstwy logiki modułu statystyk.

Klasa statyczna DistinctByHelper umożliwia wybranie rekordów bez potwarzania klucza, który określimy jako jeden z atrybutów metody. Jest to rozwiązanie wyjątkowo skalowalne, oparte na mechaniźmie generyczności. Użycie słowa kluczowego yield zapewnia poprawną zawartość zwracanej kolekcji.



Klasa statyczna BetweenDatesValidationHelper umożliwia zbudowanie obiektu Expression w celu umieszczenia go w wyrażeniu lambda. Zapytanie to odpowiada następnie za filtrowanie wyników, pobranych z bazy danych. Rozwiązanie to było koniecznie, aby umożliwić administratorowi określenie ram czasowych, z jakich mają być pokazane statystyki. Powyższa metoda umożliwia implementację czterech ścieżek, jakie można wykonać podczas korzystania z modułu statystyk.

Metoda GetUsersPerCountry odwołuje się do zewnętrznego serwisu za pomocą biblioteki RestSharp. Zwracany jest słownik, który zawiera nazwę kraju jako klucz, oraz liczbę użytkowników, którzy przypadają na ten kraj, jako wartość. Metoda GetUsersForEveryMonth zwraca ten sam obiekt, co metoda GetUsersPerCountry. Ustawienia regionalne są zaimplementowane w sposób twardy.



Powyżej zaprezentowano metody odpowiadające za sformatowanie wyników dla modułu statystyk. Dzięki nim, administrator może korzystać z estetycznych i jasnych danych. Ponownie, jak w przypadku reszty metod z modułu statystyk, wykorzystano słowniki danych.

# Moduł e-mail

Moduł MailManagement służy do obsługi newslettera oraz formularza kontaktowego, umieszczonego na stronie klienckiej. ServiceCMS posiada możliwość zdefiniowania danych uwierzytelniających dla e-maila właściciela zakładu usługowego. Dzięki temu, moduł ten jest możliwie najbardziej bezobsługowy oraz elastyczny.

Metoda SendMail umożliwia wysłanie maila do zbioru adresatów, bądź – w wersji przeciążonej – do jednego adresata. W ciele metody następuje definiowanie treści oraz tematu wiadomości. Wywoływana jest też metoda, konfigurująca klienta SMTP (ang..), który opisany jest poniżej. Cała operacja jest opatrzona blokiem try-catch, na wypadek, gdyby napotkano błędy związane z połączeniem internetowym.

W ciele metody ConfigureSmtpClient następuje konfiguracja klienta SMTP, który jest użyty do wysłania maila za pomocą modułu Newsletter oraz RegistratedServices. Konfiguracja klienta jest zależna od ustawień, pobranych z modułu Settings.

# Moduł RestSharp



Metoda statyczna GetIpInfo, korzystając z biblioteki RestSharp, pobiera kraj dla podanego w atrybucie metody adresu IP. Metoda jest zabezpieczone przed błędami z połączeniem internetowym za pomocą bloku try-catch, przechwytującego zarówno wyjątek połączenia, jak i błąd serializacji wyniku.



# Podsumowanie

W trakcie fazy projektowej założono, że system ServiceCMS będzie zaprojektowany do obsługi stron internetowych przez personel nietechnicznych. Założono, że strony te będą należeć do firm usługowych, gdzie pracownicy mają bezpośredni kontakt z klientem. Z tego względu na potrzeby prezentacji aplikacji wybrano zakład fryzjerski. Przed zaprojektowaniem aplikacji zasięgnięto porady u ekspertów dziedzinowych, w celu dokładnego określenia wymagań niefunkcjonalnych oraz funkcjonalnych. Miało to sprawić, że system będzie możliwie najbardziej zbliżony do wymagań rynkowych.

Implementacja systemu ServiceCMS przebiegła bez większych problemów. Poprawnie zaprojektowano architekturę warstwową aplikacji oraz z powodzeniem wdrożono wzorce projektowe, które znacząco obniżyły nakład pracy, potrzebny do kompletnego przetestowania aplikacji.

Po wykonaniu projektu można śmiało stwierdzić, że wykorzystane narzędzia spełniły swoje zadanie. W trakcie implementacji nie było przypadku, aby którekolwiek z wybranych narzędzi uniemożliwiało wdrożenie funkcjonalności. Zewnętrzne biblioteki były bezproblemowe w użyciu, a efekt ich pracy zgadzał się z przewidywaniami.

# Bibliografia

[1] Beck Kent, TDD. Sztuka tworzenia dobrego kodu, Helion, Gliwice 2014.

[2] Freeman Adam, ASP.NET MVC 5. Zaawansowane programowanie, Helion, Gliwice 2015.

[3] Fryźlewicz [Zbigniew](http://helion.pl/autorzy/zbigniew-fryzlewicz,zbigniewfryzlewicz.htm) , Bukowska [Ewa](http://helion.pl/autorzy/ewa-bukowska,ewabukowska.htm) , Nikończuk Daniel ASP.NET MVC 4. Programowanie aplikacji webowych, Helion, Gliwice 2013.

[4] Martin Robert C., Czysty kod. Podręcznik dobrego programisty, Helion, Gliwice 2009.

[5] Metsker Steven John, C#. Wzorce projektowe, Helion, Gliwice 2005.