UNIWERSYTET GDAŃSKI
Wydział Matematyki, Fizyki i Informatyki

Kamil Pek
nr albumu: 231 050

TrainCMS — system zarządzania treścią witryny internetowej

Praca licencjacka na kierunku:
INFORMATYKA
Promotor:
dr W. Bzyl

Gdańsk 2017

Streszczenie

W pracy przedstawiono wersję deweloperską systemu zarządzania treścią witryny internetowej „TrainCMS". W trakcie pracy zaimplementowano publikowanie artykułów, kategoryzację, wyświetlanie listy kategorii na pasku nawigacji, kalendarz wydarzeń oraz kanał RSS. Stworzono User Interface, który wyświetla wszystkie artykuły na stronie głównej, niezależnie od kategorii w kolejności malejącej od daty dodania oraz kalendarz wydarzeń. Do artykułów i wydarzeń w kalendarzu zaimplementowano możliwość załączania ilustracji oraz dodawania komentarzy.

Zaimplementowano panel administratora do zarządzania artykułami, kategoriami, komentarzami, tagami, użytkownikami i kalendarzem, zakładkami, komponentami strony głównej oraz do podglądu statystyk.

Przy implementacji użyto technologie najnowszych wersji Ruby, Ruby on Rails, ZURB Foundation, jQuery Turbolinks, Plataformatec Devise, CarrierWave, RMagick, reCAPTCHA, CKEditor, Chartkick, Prawn, RSS.

Projekt wdrożono w serwisie `heroku.com` i jest dostępny pod adresem: https://traincms.herokuapp.com/.

Kod źródłowy dostępny jest w serwisie `github.com` pod adresem: https://github.com/kamilpek/traincms/.

Słowa kluczowe

cms, ruby on rails, calendar, comments, tags, rss

Spis treści

Wprowadzenie

Podczas kilkuletniej pracy z najpopularniejszymi aplikacjami w tej kategorii, takimi jak Joomla i WordPress nabyłem doświadczenie oraz swój pogląd na to jak ma wyglądać system zarządzania treścią (ang. Content Managment System, CMS). Naturalnym stało się więc stworzenie własnego systemu, przy okazji prezentując jak najszerszą część umiejętności nabytych w trakcie trwania studiów.

Istniejące systemy są często wybierane między innymi przez lokalne serwisy informacyjne, przedsiębiorstwa i instytucje, dlatego w swoim systemie zawarłem funkcjonalności, które na pewno przydadzą się różnym podmiotom w skutecznym zaistnieniu w Internecie.

Podczas tworzenia interfejsu użytkownika i administratora, kierowałem się głównie ergonomią użytkowania i przedstawieniem możliwości jakie prezentuje system w jak najbardziej przystępny sposób tak, aby początkujący użytkownik mógł poruszać się w sposób intuicyjny po aplikacji.

ROZDZIAŁ 1

Wstęp i opis problemu

1.1. Porównanie dostępnych rozwiązań z systemem TrainCMS

Na rynku systemów zarządzania treścią znajdziemy sporo różnych rozwiązań. W dalszej części rozdziału przybliżę i porównam z systemem TraincMS dwa najbardziej popularne produkty, będzie to Joomla i WordPress[1]. Systemy różnią się od siebie pod wieloma względami. Rozwiązanie przedstawione przeze mnie jakim jest TrainCMS różni się przede wszystkim technologią wykonania, gdyż oba wcześniej wspomniane systemy wyprodukowane są technologii języka PHP i bazy danych MySQL, gdzie mój system opiera się na frameworku Ruby On Rails i bazie danych PostgreSQL.

1.1.1. Joomla!

Joomla jest to system zarządzania treścią, napisany w języku PHP, wykorzystujący do swojego działania system zarządzania bazą danych MySQL, rozpowszechniana jest na licencji GPL. Nazwa Joomla w języku suahili oznacza razem.

System ten oferuję obsługę wielu kont użytkownika, wyszukiwarkę zaimplementowaną w User Interface, tworzenie wydruków artykułów, dołączanie ilustracji do artykułu, komentowanie artykułów przez niezalogowanych użytkowników. Wymienione funkcjonalności pokrywają się z możliwościami stworzonego przeze mnie systemu.

[1]Istnieje jeszcze jeden bardzo popularny system zarządzania treścią – Drupal. Podobnie jak oba opisane wyżej systemy, wyprodukowany został w technologii języka PHP i jest udostępniony na otwartej licencji.

TrainCMS posiada także inne możliwości, których nie oferuję Joomla w wersji podstawowej, jest to kalendarz wydarzeń, dodawanie załączników, generowanie dokumentów pdf zawierających artykuły, przedstawienie statystyk w formie graficznej, karuzela ilustracji wyróżnionych artykułów. Natomiast niektóre z rozwiązań zostały rozszerzone względem Joomli są to komentarze, które w projekcie TrainCMS rejestrują adres IP autora komentarza.

Znajdziemy także w Joomli funkcje, których nie posiada mój system. Jednym z takich rozwiązań jest tworzenie struktury menu w formie drzewiastej. Kolejnym rozwiązaniem jest możliwość zmiany szablonu frontu strony i szablonu zaplecza witryny. Główną funkcjonalnością Joomli jest możliwość łatwego rozszerzania możliwości strony za pomocą pluginów i komponentów. Podczas porównywania obu systemów należy pamiętać, że Joomla jest produktem z wieloletnim doświadczeniem na rynku, tworzonym przez zespół programistów z całego świata. Rozwiązania oparte na Joomli znajdują zastosowanie głównie przy dużych witrynach.

Rysunek 1.1: Przykładowa strona wykonana w Joomla!.

Źródło: commons.wikimedia.org

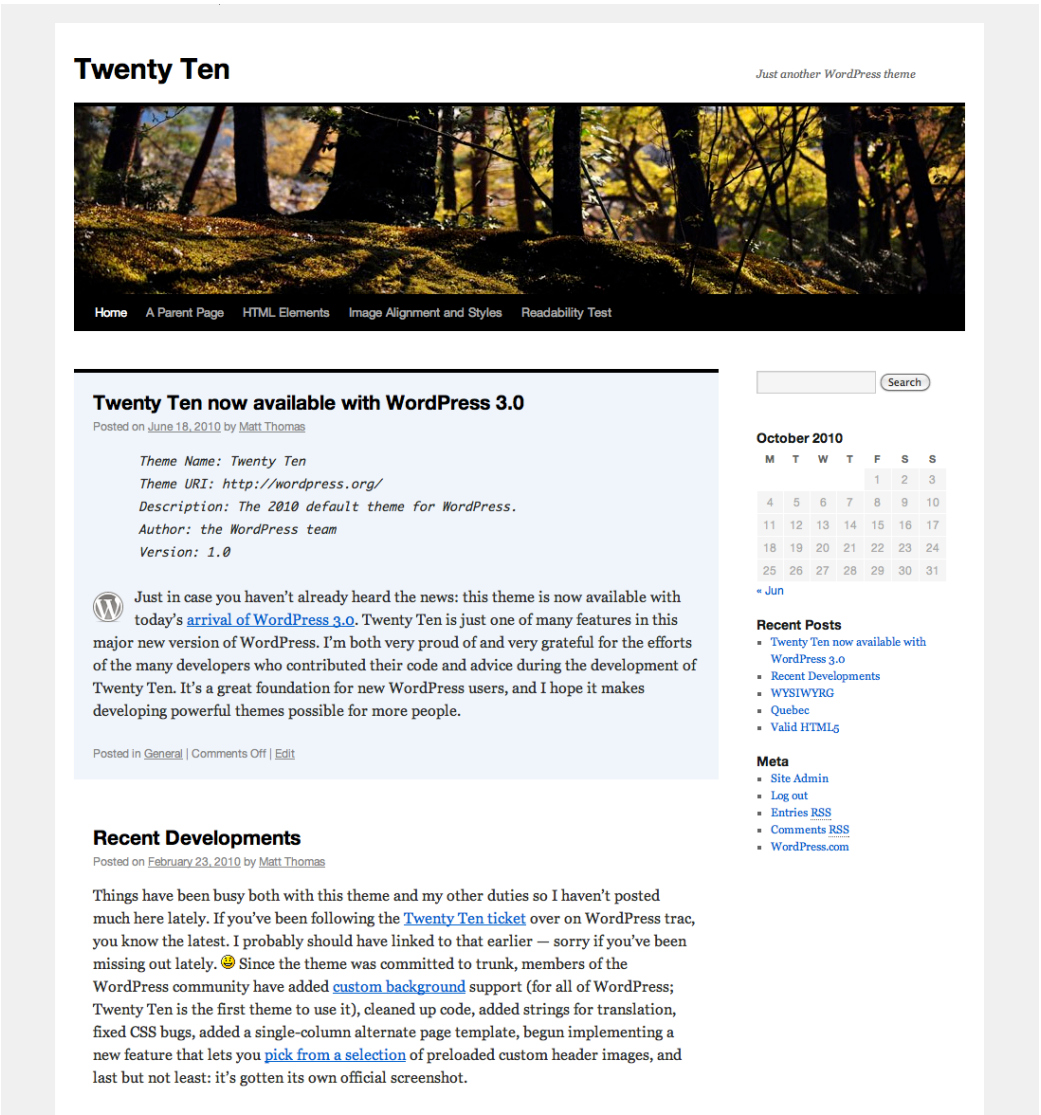
1.1.2. WordPress

WordPress jest systemem zarządzania treścią napisanym w języku PHP, wykorzystujący systemem zarządzania bazą danych MySQL i jest dystrybuowany na licencji GPL.

System WordPress jest zdecydowanie mniej rozbudowany w porównaniu do Joomli. Oferuje on takie funkcjonalności jak podstawową kategoryzację, tagowanie i komentowanie artykułów, obsługę wielu kont użytkownika, odrębny interfejs dla użytkownika gościa, zwykłego użytkownika i administratora oraz podgląd statystyk jest również w pełny responsywny. Wszystkie wymienione funkcjonalności pokrywają się z zaimplementowanymi w systemie TrainCMS.

W TrainCMS znajdziemy także inne możliwości, których nie oferuję WordPress w wersji podstawowej, jest to kalendarz wydarzeń, dodawanie załączników, generowanie dokumentów pdf zawierających artykuły oraz karuzela ilustracji wyróżnionych artykułów. Natomiast niektóre z rozwiązań zostały rozszerzone względem Joomli są to komentarze, które w projekcie TrainCMS rejestrują adres IP autora komentarza.

Należy w tym miejscu wspomnieć, że główną funkcjonalnością WordPress jest łatwość instalacji i zmiany wielu dostępnych szablonów strony. WordPress jest produktem z utartą pozycja na rynku systemów zarządzania treścią, który podobnie jak Joomla tworzony jest przez zespół programistów z całego świata. Witryny obsługiwane przez WordPress to głównie blogi.



Rysunek 1.2: Przykładowa strona wykonana w WordPress.

Źródło: commons.wikimedia.org

1.2. Możliwości zastosowania praktycznego

System TrainCMS został opracowany w taki sposób, aby sprostać wielu wymaganiom różnych użytkowników. Oferuje sporo możliwości, które przypadną do gustu każdemu i będą zarazem bardzo przydatne w codziennej pracy nad własna witryną Internetową. Reasumując, możliwości serwisu ogranicza jedynie wyobraźnia administratora.

1.2.1. Strona wizytówka

W celu stworzenia optymalnej i efektownej strony wizytówki należałoby uruchomić tryb statycznej strony głównej. W tymże celu utworzymy zakładkę, którą oznaczymy jako strona główna. Ilość pozostałych zakładek jest dowolna. Może się też zdarzyć potrzeba prowadzenia minibloga lub prostych aktualności firmy, tutaj posłużymy się kategoriami i artykułami. Łącza do kategorii będą wyswietlona na górnym pasku nawigacji co ułatwi poruszanie się po stronie. Po odpowiednim według operatora strony rozmieszczeniu informacji, możemy przejść do podglądu statystyk, które w tym przypadku mogą wyświetlić informację na przykład o tym, która sekcji informacji jest najbardziej popularna.

1.2.2. Internetowe portfolio

Każda osoba tworząca w Internecie portfolio swojej działalności zamierza przyciągnąć w ten sposób jak największą liczbę nowych klientów. Aby skutecznie rozwiązać ten problem, proponuję, każde dzieło zaprezentować w osobnym artykule. Natomiast informacje, które autor chciałby, aby były zawsze łatwo dostępne, umieścić w przygotowanych do tego zakładkach, do których to łącza będą wyswietlane na górnym pasku nawigacji. Można też przyjąć inne podejście do tego tematu, otóż ustawić stronę główną jako stronę statyczną, następnie utworzyć kategorię, do której łącze, podobnie jak do zakładek ukaże się na górnym pasku nawigacji, w której to umieścimy dzieła swojej działalności.

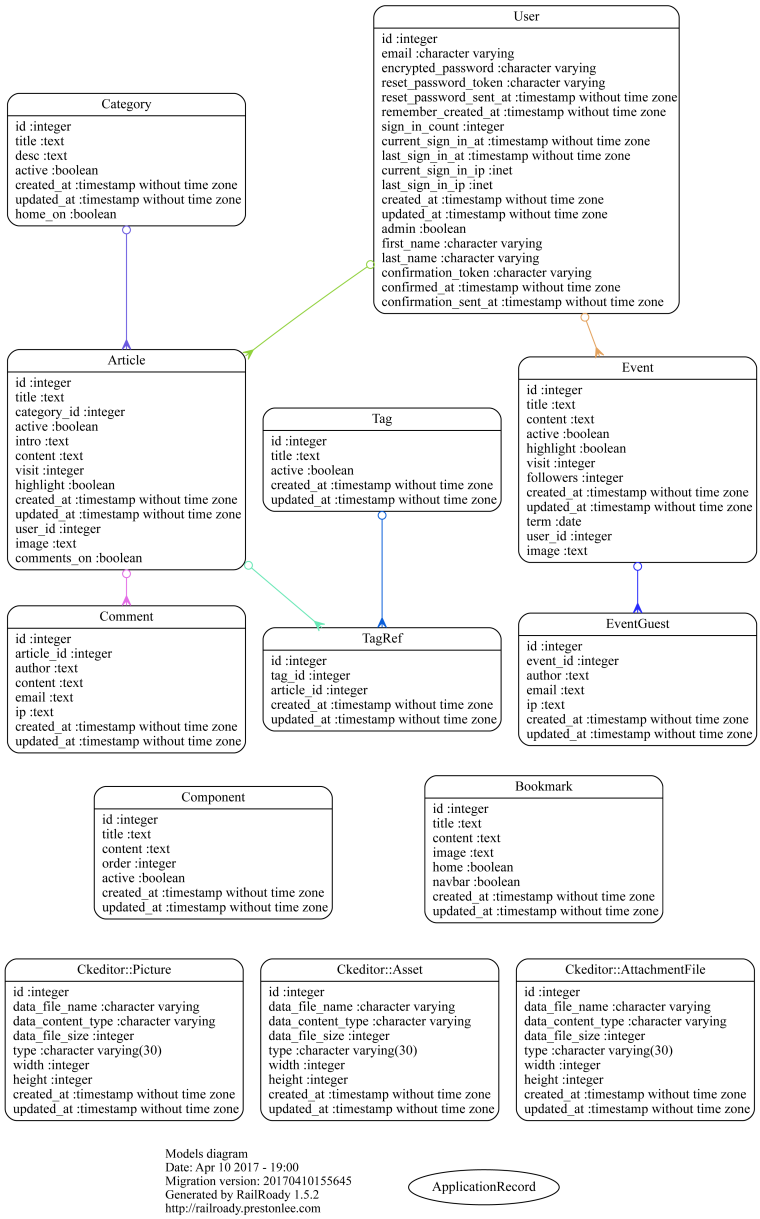
1.2.3. Serwis informacyjny

W tym rozwiązaniu znajdą zastosowanie wszystkie zaimplementowane w systemie funkcjonalności. Większość rozwiązań została wyprofilowana właśnie na tego typu zastosowania. Głównym szkieletem jest w tym przypadku możliwość tworzenia wielu kategorii, gdzie redaktor takiego serwisu, będzie mógł z pełną łatwością organizować wszystkie tematy poruszane na portalu i jednocześnie wszystkie artykuły z każdej kategorii będą wyświetlane na stronie głównej. Gorące tematy będzie można oznaczać jako wyróżnione i tym sposobem będą przez cały widoczne na szczycie karuzeli. Gość odwiedzający serwis z łatwością wejdzie w interakcję ze stroną poprzez system komentarzy, operator serwisu będzie mógł korzystać z przejrzystych statystyk i za ich pomocą analizować pracę portalu oraz planować dalszy jego rozwój. Z pomocą dla nowych gości przyjdą tagi, dzięki którym będzie można szybko wyszukać artykuły poruszające dany temat. Łatwiejsze stanie się planowanie różnego rodzaju imprez za pomocą wbudowanego kalendarza wydarzeń. Autor piszący artykuły dla serwisu nie będzie musiał zagłębiać się w panel zaplecza, na stronie głównej po zalogowaniu znajdzie skróty do najważniejszych funkcji takich jak nowy artykuł, lista własnych artykułów oraz lista komentarzy pod tymi artykułami. Jeżeli autor zechce, ma możlwiość wyłączenia komentarzy. Jeżeli nadejdzie taka potrzeba, możemy skorzystać z zaimplemenotwanego mechanizmu zakładek, które to, po utworzeniu wyświetlone będą na górnym pasku nawigacji.

ROZDZIAŁ 2

Projekt i analiza

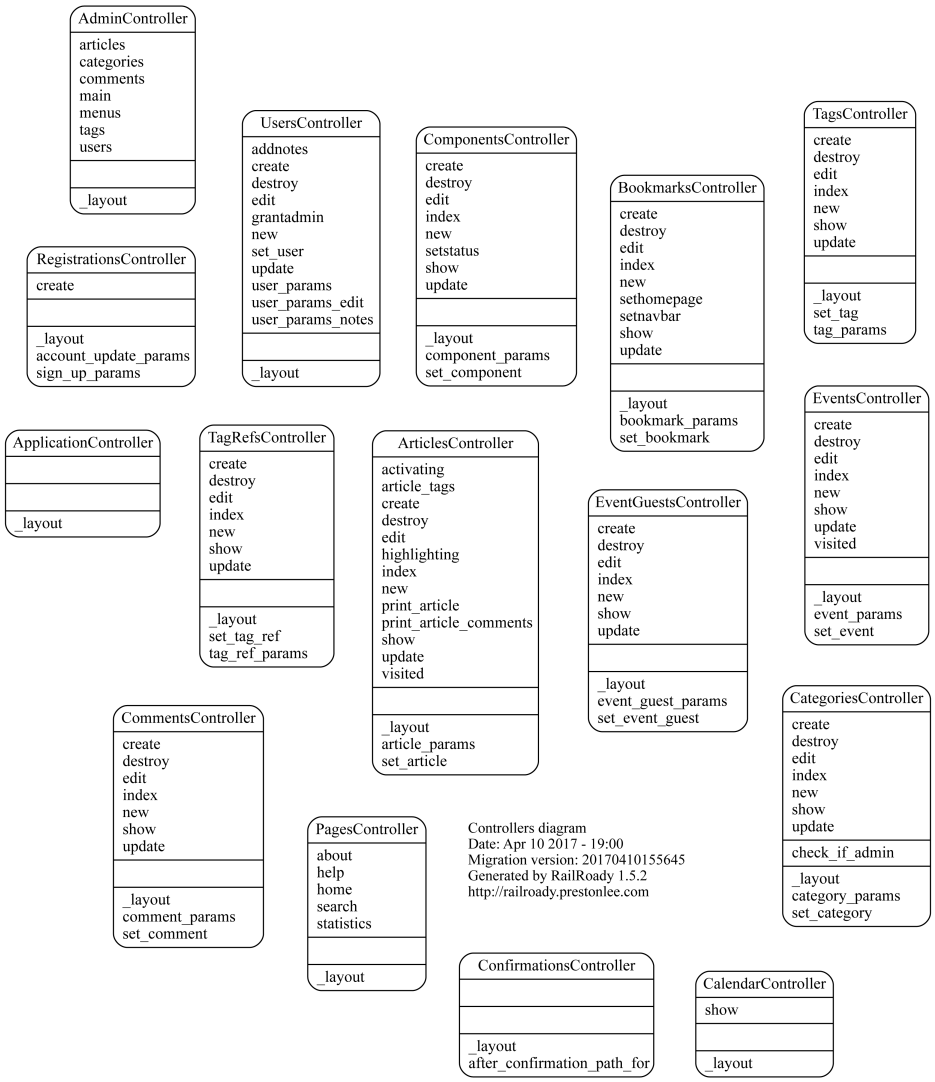
2.1. Diagram związków encji



Rysunek 2.1: Diagram związków encji.

Źródło: Opracowanie własne za pomocą Gemu RailRoady [4].

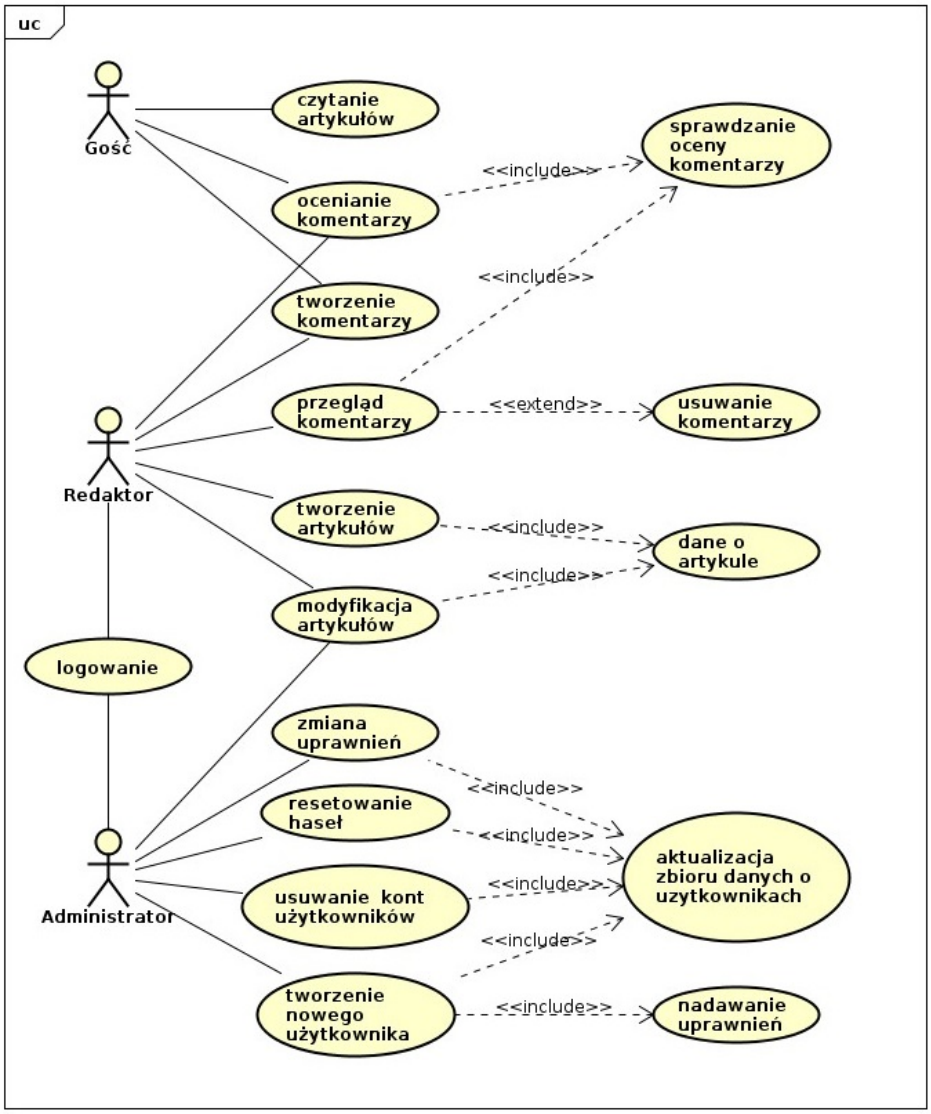
2.2. Diagram kontrolera danych



Rysunek 2.2: Diagram kontrolera danych.

Źródło: Opracowanie własne za pomocą Gemu RailRoady.

2.3. Diagram Przypadków Użycia

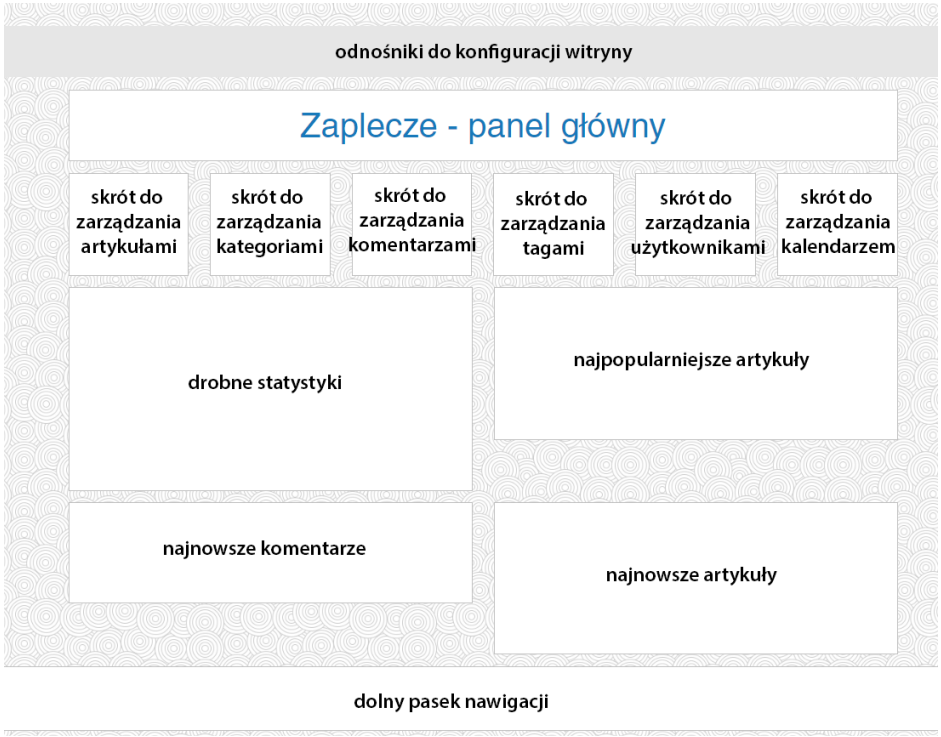


Rysunek 2.3: Diagram Przypadków Użycia.

Źródło: Opracowanie własne za pomocą programu Astah [5].

2.4. Projekt interfejsu użytkownika

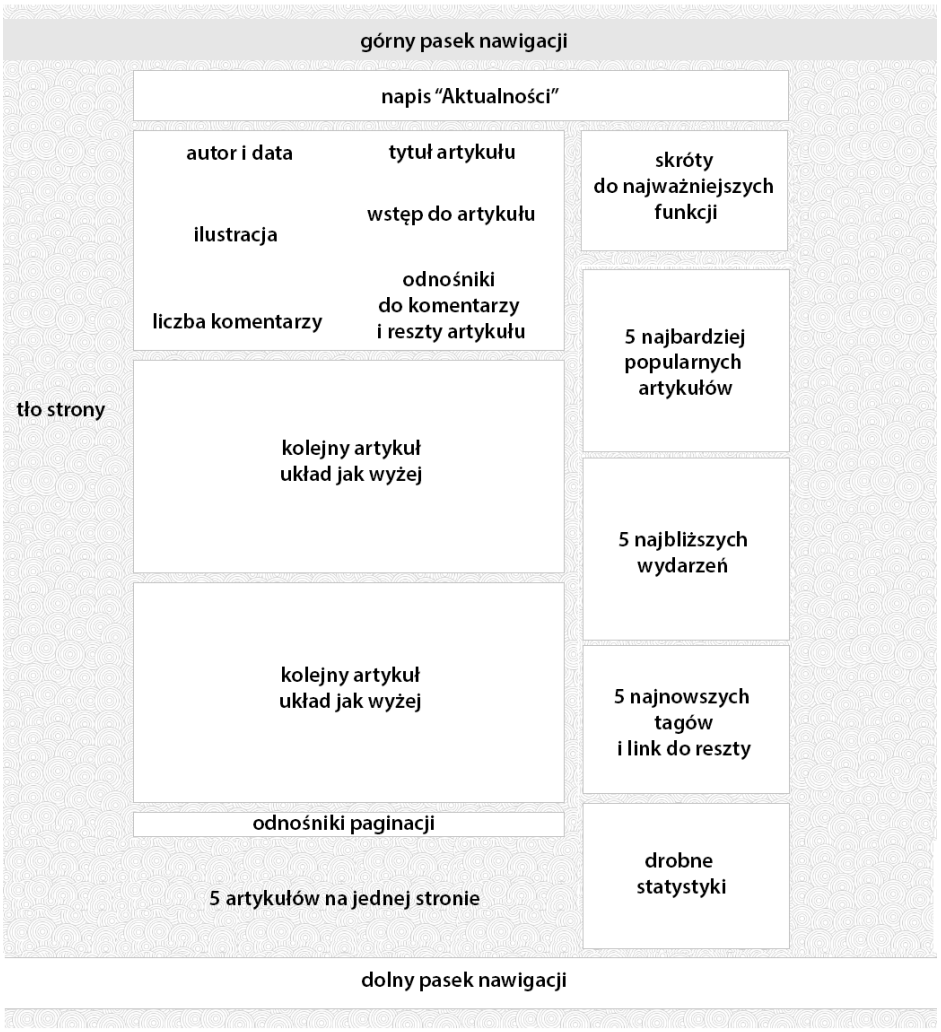
2.4.1. Panel Administracyjny



Rysunek 2.4: Projekt interfejsu użytkownika. Panel Administracyjny.

Źródło: Opracowanie własne

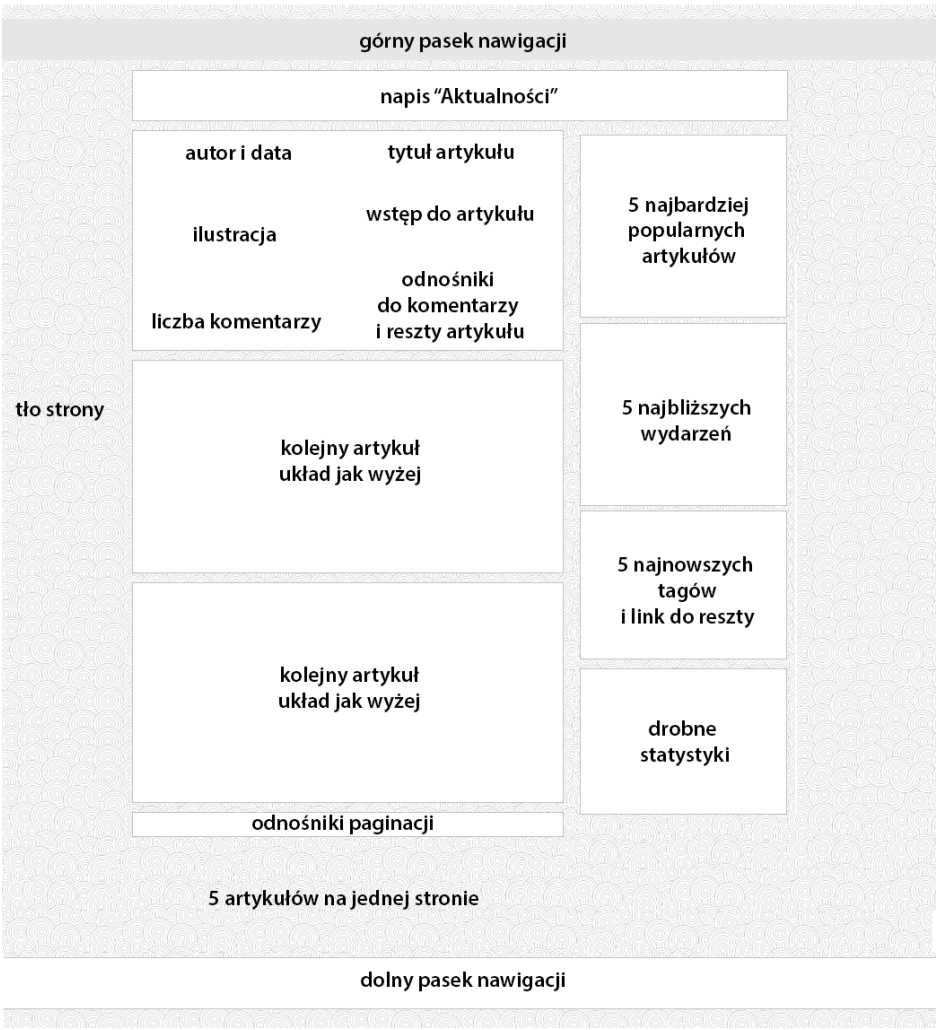
2.4.2. Widok Redaktora



Rysunek 2.5: Projekt interfejsu użytkownika. Widok Redaktora.

Źródło: Opracowanie własne

2.4.3. Widok Gościa



Rysunek 2.6: Projekt interfejsu użytkownika. Widok Gościa.

Źródło: Opracowanie własne

ROZDZIAŁ 3

Implementacja

3.1. Architektura rozwiązania – Ruby on Rails

Głównym rusztowaniem całego systemu jest framework Ruby On Rails [1] [2] [3] [6]. Odpowiada on za całość frontendu i backendu. Do swojego działania używa bazy PostgreSQL. W dalszych podrozdziałach przybliżę jak wyglądała implementacja poszczególnych elementów systemu za pomocą Ruby On Rails. Framework dystrybuowany jest na licencji MIT License[1].

3.1.1. Artykuły i Kategorie

Podstawą jednostką na stronie jest artykuł, który zawsze zawiera się w jednej z uprzednio utworzonych kategorii. Artykuł posiada atrybuty takie jak tytuł, wstęp, treść główną, numer kategorii do której został przypisany, ilustrację, znacznik aktywności, znacznik wyróżnienia, znacznik komentarzy, liczbę wyświetleń oraz znaczniki czasowe – data, czas utworzenia i edycji. Chciałbym przybliżyć niektóre z atrybutów, pierwszym z nich będą znaczniki aktywności, wyróżnienia i komentarzy, które kolejno oznaczają informacje o tym, czy artykuł jest aktywny co przekłada się na to, że będzie wyświetlony na stronie głównej oraz spisie artykułów danej kategorii, kolejny atrybut determinuję to, czy artykuł zostanie wyświetlony na karuzeli ilustracji na szczycie strony głównej, ostatni ze znaczników pozwala dezaktywować moduł komentarzy w przypadku, gdyby zaszła konieczność, aby przy pewnym artykule nie miałoby być komentarzy. Liczba wyświetleń jest sumaryczną wartością wszystkich odsłon artykułu, obliczanie polega na pobraniu liczby

[1]MIT License – Jest to jedna z najmniej rozbudowanych licencji oprogramowania i zarazem jest jedna z najbardziej liberalnych. Umożliwia nieograniczone prawo do użytkowania, kopiowania, modyfikowania i publikacji, również sprzedaży wersji pierwotnej bądź zmodyfikowanej oprogramowania. Jedynym wymaganiem jest umieszczenie informacji o autorze i licencji.

wyświetleń z bazy danych, następnie zwiększeniu jej o jeden oraz aktualizacji wartości w bazie danych. Znaczniki czasowe są automatycznie dodawane przez Rails. Zarówno na stronie głównej, jak i na liście artykułów danej kategorii wyświetlany jest tytuł, wstęp do artykułu, ilustracja, autor, data utworzenia, liczba komentarzy. Po przejściu do artykułu zobaczymy pełną treść, w tym celu właśnie zostały zaimplementowane dwa oddzielne atrybuty. Zaimplementowana została również wyszukiwarka artykułów, gdzei słowem kluczowym wyszukiwania jest tytuł artykułu. Algorytm wyszukiwania ignoruje wielkość liter oraz pozwala szukać za pomocą fragmentów wyrazów. Artykuły mogą dodawać jedynie zalogowaniu użytkownicy. Po przejściu do artykułu możemy go wydrukować dzięki specjalnie do tego przygotowanej formatce optymalizującej miejsce na stronie kartki papieru.

Kategoria natomiast posiada atrybuty takie jak: tytuł, opis, znacznik aktywności, znacznik strony głównej oraz znaczniki czasowe – data, czas utworzenia i edycji. Wszystkie kategorie oznaczone jako aktywne wyświetlane są na górnym pasku nawigacji, po przejściu w odnośnik do danej kategorii zobaczymy opis kategorii oraz listę wszystkich artykułów przypisanych do tejże kategorii. Istnieje również taki atrybut jak znacznik strony głównej, który określa to czy artykuły przypisane do kategorii będą wyświetlane na stronie głównej. Do strony głównej może być przypisane kilka kategorii.

3.1.2. Komentarze

Do każdego artykułu możemy dodawać komentarze. Ta funkcjonalność udostępniona jest dla gości odwiedzających stronę, a co za tym idzie, aby dodać komentarz nie jest wymagane logowanie. W celu dodania komentarza musimy podać swój adres e-mail, który jednak nie będzie weryfikowany, jest to powszechnie stosowana praktyka. W bazie danych zapisywany jest także adres IP autora komentarza. Każdy komentarz można ocenić w skali plus/minus. Obok treści komentarza wyświetlana jest wartość oceny, która może być również ujemna. Jeden odwiedzający może ocenić jeden komentarz jeden raz, informacja o tym fakcie zapisywana jest w ciasteczkach.

3.1.3. Tagi

W celu dodatkowej kategoryzacji oraz łatwiejszego znajdowania poszukiwanych przez odwiedzających treści, zaimplementowano tagi artykułów. Po utworzeniu artykułu, możemy przejść do formularza dodawania tagów, w którym za pomocą listy rozwijanej wybieramy dopasowane tagi, w tym samym miejscu, jeżeli nie znajdziemy poszukiwanych przez siebie tagów, możemy dodać swój tag i przypisać go do artykułu.

3.1.4. Kalendarz Wydarzeń

Zaimplementowany został również kalendarz wydarzeń, który wyświetla dodane wydarzenia w formie klasycznego kalendarza ściennego, podzielonego na pojedyncze miesiące. Każde wydarzenie, na wzór artykułu, posiada atrybuty takie jak tytuł, treść, termin wydarzenia, ilustrację, znacznik aktywności, znacznik wyróżnienia, liczbę wyświetleń oraz znaczniki czasowe – data, czas utworzenia i edycji. Znaczniki aktywności i wyróżnienia są odpowiedzialne odpowiednio za wyświetlanie wydarzenia na kalendarzu oraz na liście pod kalendarzem.

Goście odwiedzający stronę mogę zapisywać się do wybranego przez siebie wydarzenia, polega to na podaniu swojego imienia oraz adresu e-mail, ponadto w tle do bazy danych trafia również adres IP osoby deklarującej dołączenie do wydarzenia. Wydarzenia mogą tworzyć jedynie zalogowani użytkownicy.

3.1.5. Zakładki

W celu uporządkowania statycznych informacji prezentowanych na stronie, zaimplementowano zakładki. Za pomocą atrybutów przypisanych do każdej zakładki możemy określać tytuł, treść, ilustrację, znacznik strony głównej, znacznik paska nawigacji oraz znaczniki czasowe – data, czas utworzenia i edycji. Znacznik strony głównej decyduje o tym, czy dana zakładka będzie pełniła rolę strony głównej. Natomiast znacznik paska nawigacji determinuje fakt wyświetlenia odnośnika do zakładki na górnym pasku nawigacji. Aby zakładka została wyświetlona na stronie głównej konieczne jest zaznaczenie tylko jednej zakładki w przypadku, gdy zostaną więcej niż dwie, wtedy strona główna przybierze formę dynamiczną.

3.1.6. Strona główna

Strona główna może zostać skonfigurowana dwojako. Pierwszy sposób polega na wyświetlaniu listy artykułów oraz jej stronicowaniu za pomocą Gema o nazwie `will_paginate`[8]. Jak wcześniej wspomniałem na stronie głównej wyświetlone zostaną tylko artykuły aktywne z aktywnych kategorii. Na szczycie głównej witryny pojawi się również karuzela z wyróżnionymi artykułami a i wydarzeniami. Drugim sposobem aranżacji strony głównej jest jest statyczna wersja organizowana za pomocą wyżej opisanych zakładek. Na stronie głównej możemy umieszczać także komponenty, są to ramki z pewną treścią. Na stałe zostały osadzone cztery komponenty zawierające listę popularnych artykułów, listę najbliższych wydarzeń, listę najnowszych tagów oraz drobne statystyki. Komponenty za wzór innych modułów strony posiadają znacznik aktywności, który określa czy dany komponent będzie wyświetlony na stronie głównej.

3.1.7. Nawigacja

Nawigacja po stronie zrealizowana jest za pomocą dwóch pasków nawigacji, górnego i dolnego.

Na górnym pasku znajdziemy odnośniki do strony głównej, wszystkich kategorii oznaczonych jako te, które mają się znaleźć na pasku, zakładek, które podobnie jak kategorie muszą być oznaczone jako dostępne z poziomu paska nawigacji. Znajduje się tam również odnośnik do kalendarza wydarzeń oraz wyszukiwarki artykułów.

Rysunek 3.1: Przykładowy górny pasek nawigacji.

Źródło: Opracowanie własne

Natomiast na dolnym pasku nawigacji zwanym stópką, znajduję się w widoku dla niezalogowanych użytkowników klauzula Copyright, odnośnik do planszy pod tytułem „o projekcie", odnośnik do prostej pomocy oraz odnośnik do panelu logowania. Po zalogowaniu z uprawnieniami redaktora znajdziemy dodatkowo odnośnik do statystyk, natomiast po zalogowaniu z uprawnieniami administratora zyskamy odnośnik do zaplecza. Dla wszystkich zalogowanych użytkowników na prawym końcu dolnego paska nawigacji znajduje się odnośnik do panelu zmiany hasła oraz przycisk wylogowania.

Rysunek 3.2: Przykładowy dolny pasek nawigacji.

Źródło: Opracowanie własne

3.1.8. Kanał RSS

Kolejną zaimplementowaną w systemie funkcjonalnością jest agregator kanału RSS. Dzięki niemu fani witryny mogą dodać sobie link RSS do swojego czytnika i mieć zawsze dostęp do najnowszych informacja publikowanych na stronie.

Implementacja polegała na stworzeniu dwóch plików, jednego w standardzie Atom i jednego w standardzie RSS. Następnie wypełnieniu ich kodem wyświetlającym tytuł artykułu, nagłówek, autora oraz odnośnik do pełnej treści artykułu. Oba pliki zostały zwalidowane i są w pełni zgodne z obowiązującymi wersjami obu standardów. Kod generatora spływu wiadmości w standardzie Atom

```
1  atom_feed do |feed|
2    feed.title "TrainCMS - Artykuly"
3    feed.updated @articles.maximum(:updated_at)
4    @articles.order("created_at desc").each do |article|
5      feed.entry article do |entry|
6        entry.title article.title
7        entry.content sanitize(article.intro, :tags => {})
8        entry.author do |author|
9          author.name
10         User.where(id:article.user_id).pluck(:email).last
11       end
12     end
13   end
14 end
```

Listing 3.1: Kod generatora spływu wiadmości w standardzie Atom

3.2. ZURB Foundation

ZURB Foundation [9] jest responsywnym frameworkiem frontendu. Został stworzony w 2011 roku i dystrybuowany jest na licencji MIT License. W systemie TrainCMS odpowiada za frontend.

3.2.1. Instalacja

Dołączenie Foundation do projektu Ruby On Rails polega na zainstalowaniu Gema o nazwie `foundation-rails`. Następnie przeprowadzeniu automatycznej instalacji za pomocą polecenia

```
1 <table class="stack"></table>
```

Listing 3.2: Przykładowa tabela

oraz dodaniu odpowiednich zapisów w plikach kaskadowych arkuszy stylów i plikach skryptów JavaScript.

3.2.2. Użycie

Podczas budowy całego systemu zarządzania treścią strony internetowej korzystałem z bogatej biblioteki komponentów jaką oferuje framework Foundation. Każda zaimplementowana tabela otrzymała klasę

```
1 <table class="stack"></table>
```

Listing 3.3: Przykładowa tabela

która odpowiedzialna jest za wyświetlanie tabeli w mobilnym widoku jako stos kolumn, wiersze tabeli wyświetlane są na przemian kolor biały z kolorem grafitowym, tę funkcję również zawdzięczamy Foundation. Wszystkie odnośniki posiadają klasę

```
1 <%= link_to 'odnosnik', odnosnik_path,
2         class:'button' %>
```

Listing 3.4: Przykladowy przycisk

co pozwala na wyświetlania każdego odnośnika w formie prostokątnego przycisku. Wszystkie odnośniki nie są przyciskami o tej samej wielkości, odnośniki zawarte w tabeli mają dodatkową klasę

```
<%= link_to 'maly odnosnik', maly_odnosnik_path,
        class:'tiny button' %>
```

Listing 3.5: Przykładowy mały przycisk

dzięki której przycisk staję bardzo mały i w elegancki sposób wkomponowuje się w wiersze tabeli. Również do nawigacji podczas stronicowania wykorzystano metodę renderowania w stylu Foundation. Ogólna konwencja graficzna opiera się na siatce, opisanej za pomocą znaczników `div`. Każda sekcja wszystkich stron poszczególnych modułów całego systemu zapisana jest w znaczniku `div`, który otrzymuje za każdym razem klasę

```
<div class="callout"></div>
```

Listing 3.6: Przykładowy div

Dzięki, której treść wyświetlana jest na białym eleganckim prostokącie z ostrymi rogami. Strona główna została podzielona za pomocą siatki znaczników `div` na kilka sekcji. Formularze wprowadzania danych wykorzystują klasę `div`

```
<div class="input-group">
        <span class="input-group-label">Tytul</span>
         <%= f.text_field :title, type:"text",
                class:"input-group-field" %>
</div>
```

Listing 3.7: Przykładowe pole tekstowe

która pozwala na wyświetlanie etykiety i samego pola w jednej linii, oszczędzając tym samym miejsce, prezentując stronę w jeszcze bardziej czytelny sposób.

3.2.3. Ikony

Bardzo ciekawą funkcjonalnością frameworku Foundation jest możliwość dodawania ikon w kodzie strony. Polega to na zainstalowaniu Gemu o nazwie Foundation Icon Fonts on SASS for Rails [10], potrzebne do tego będzie dodanie wpisu do pliku Gemfile oraz dodania linii kodu do pliku `application.css.scss` znajdującego się w katalogu `app/assets/stylesheets/`:

```
1 @import 'foundation-icons';
```

Listing 3.8: Kod dołączający zbiór ikon Foundation Icons do aplikacji

Na koniec w celu wyświetlenia ikony na stronie, należy dodać kod, którego wynikiem będzie ikona kalendarza wielkości 24 punktów:

```
1 <font size="24"><i class="fi-calendar"></i></font>.
```

Listing 3.9: Przykładowa ikona ze zbioru ikon Foundation Icons

Rysunek 3.3: Ikona kalendarza ze zbioru ZURB Foundation Icons.

Źródło: Opracowanie własne

3.3. CarrierWave, CKEditor, Cloudinary

3.3.1. CarrierWave i Cloudinary

CarrierWave [11] jest to Gem usprawniający obsługę plików o różnych rozszerzeniach dla aplikacji w Ruby, natomiast Cloudinary jest to usługa oferująca przechowywanie plików na bezpłatnym serwerze hostingowym, dodatkowo o tej samej nazwie istnieje Gem, który obsługuje całą tę funkcjonalność z poziomu aplikacji Ruby. Oba rozwiązania są ze ściśle sobą powiązane, ale mogą też działać samodzielnie. Gemy udostępnione są na licencji MIT License.

Gem należy dodać do pliku gemfile. Następnie utworzyć uploader za pomocą polecenia:

```
1 $ rails generate uploader Avatar
```

Listing 3.10: Polecenie generujące plik uploadera

które wygeneruje plik, w którym to możemy przeprowadzić konfigurację. Dodatkowo do swojego pełnego działania potrzebuje pakiet RMagick, który możemy zainstalować za pomocą polecenia systemowego:

```
1 $ sudo apt-get install imagemagick libmagickwand-dev
```

Listing 3.11: Polecenie instalujące oprogramowanie RMagick

Przed przejściem do dalszych kroków, potrebne będzie konto w serwisie Clodinary, z tego też serwisu po zalogowaniu pobieramy plik konfiguracyjny przygotowany dla aplikacji napisanych w Ruby, zapisujemy go w katalogu `/config`. Do każdego z plików uploadera, należy dodać dwie linie

```
1 include CarrierWave::Rmagick
2 include Cloudinary::CarrierWave.
```

Listing 3.12: Framgent zawartości pliku uploadera

W celu zachowania porządku na serwerze usługi Cloudinary w każdym pliku uploadera możemy dodać linię, która oznacza tagiem każdy załadowany przez nas plik:

```
process :tags => ['random_tag']
```

Listing 3.13: Przykładowy tag dla pliku

Aby wyświetlić załadowany plik, na przykład obraz należy dodać linię o treści:

```
<%= image_tag @article.image.url %>
```

Listing 3.14: Kod wyświetlający obraz

3.3.2. CKEditor

CKEditor [12] jest edytorem WYSIWYG[2], który umożliwia łatwą i przejrzystą edycję tekstu w oknie przeglądarki, możliwościami zbliżonymi do edytora tekstu klasy Microsoft Word. Gem udostępniony jest na licencji MIT License.

W celu instalacji należy dodać gem do pliku gemfile. W drugim kroku należy dodać do pliku `config/initializers/ckeditor.rb` linie:

```
Ckeditor.setup do |config|
  config.cdn_url =
        "//cdn.ckeditor.com/4.6.1/basic/ckeditor.js"
end
```

Listing 3.15: Framgent zawartości pliku ckeditor.rb

[2]ang. what you see is what you get skrót oznaczający „to co widzisz, to otrzymasz", stosowany w technikach komputerowych do określenia rozwiązań pozwalających uzyskać już podczas produkcji tekstu wynik wielce zbliżony lub niemalże identyczny do finalnego efektu.

Następnie w pliku `/app/views/layouts/application.html.html` linię o następującej treści:

```
1 <%= javascript_include_tag "chartkick" %>
```

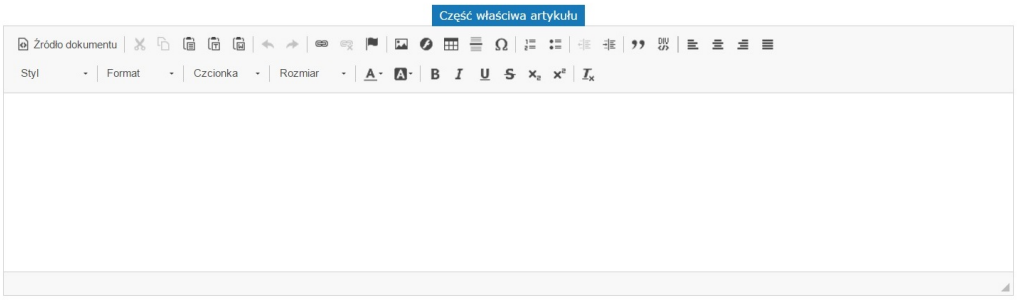
Listing 3.16: Framgent zawartości pliku application.html.rb

W miejscu, w którym chcemy użyć ten komponent dodajemy linię o treści:

```
1 <%= f.cktext_area :content, placeholder:"Content" %>
```

Listing 3.17: Kod uruchamiajacy edytor

Rozwiązanie to ściśle współpracuje z przedstawionym wyżej rozwiązaniem publikacji załączników, na przykładzie artykułów, możemy nie tylko dodać główną ilustrację publikacji, ale także za pomocą CKEditor dodać kilka innych załączników, nie tylko obrazków, które także znajdą się na serwerze usługi Cloudinary.



Rysunek 3.4: Edytor CKEditor.

Źródło: Opracowanie własne

3.4. Prawn

Prawn [13] jest to Gem generujący pliki w formacie PDF. Udostępniano został na licencji GPL.

Aby zainstalować Gem w naszym projekcie, należy go dodać do pliku Gemfile. W celu utworzenia plików PDF należy utworzyć klasę w kontrolerze, która będzie dziedziczyła z klas komponentu:

```
1 class ArticleOnePdf < Prawn::Document
```

Listing 3.18: Deklaracja klasy generującej plik PDF

Wykorzystanie Prawn umożliwia bardzo precyzyjne, pod względem rozmieszczania poszczególnych elementów, projektowanie dokumentów. Do generowanych dokumentów możemy dodawać obrazy, tabelę i wiele innych elementów. Podczas generowania precyzujemy rozmiar oraz orientację strony.

```
1  class ArticleOnePdf < Prawn::Document
2    def initialize(article)
3      super()
4      @article = article
5
6      move_down 10
7      photo = "#{Rails.root}/public#{@article.image.url}"
8      image photo, :width => 400
9
10     move_down 10
11     font("SourceSansPro-Bold.ttf", size: 14) do
12       text "#{remove_html(@article.intro)}"
13   end
14 end
```

Listing 3.19: Kod generujacy dokment zawierający ilustrację i wstęp do artykułu

3.5. Chartkick

Chartkick [14] jest gemem, który z pewnością wzbogaci wizualnie każdy projekt, w którym się znajdzie. Głównym zadaniem gema jest generowanie wykresów. Pierwszy raz został opublikowany w 2013 i teraz udostępniany jest na licencji MIT License.

Aby zainstalować Gem należy dodać go do pliku Gemfile, następnie w pliku `application.js` dodać linię

```
//= require chartkick
```

Listing 3.20: Framgent zawartości pliku application.js

natomiast w pliku `layouts/application.html` dodać linię:

```
<%= javascript_include_tag "//www.google.com/jsapi",
"chartkick" %>
```

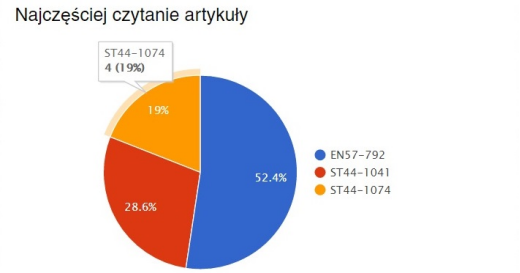
Listing 3.21: Framgent zawartości pliku application.html.rb

Bardzo ciekawym wykresem jest wykres o nazwie `pie_chart`. Można go umieścić w następujący sposób:

```
<%= pie_chart Article.group(:title).sum(:visit) %>
```

Listing 3.22: Kod generujący wykres kołowy

Generuje on bardzo przejrzysty obrazek z wykresem kołowym:



Rysunek 3.5: Wykres kołowy.

Źródło: Opracowanie własne

3.6. reCAPTCHA

W celu rozwiązania problemu zabezpieczenia systemów wdrożonych w ogólnodostępnej sieci Internet przed różnymi formami złośliwego oprogramowania, a szczególnie robotów spamujących za pomocą formularzy zawartych na stronach internetowych, wdrożyłem rozwiązanie o nazwie reCAPTCHA produkcji Google za pomocą Gemu o tej samej nazwie [15] dystrybuowanego na licencji MIT License.

Instalacja polega na dodaniu wpisu do pliku Gemfile, zarejestrowaniu strony na serwerach Google w celu pobrania kodu strony i kodu sekretnego, które to należy dodać do pliku `/config/initializers/recaptcha.rb`. Następnie dodaniu do formalarza linii kodu:

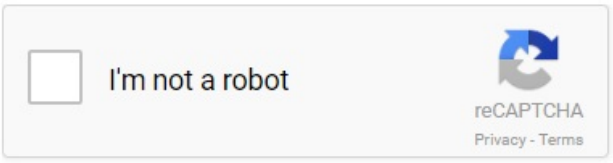
```
<%= recaptcha_tags %>
```

Listing 3.23: Kod wyświetlający formularz reCAPTCHA

Na koniec należy dodać do pliku kontrolera poniższy fragment kodu:

```
if verify_recaptcha(model: @user) && @user.save
  redirect_to @user
else
  render 'new'
end
```

Listing 3.24: Kod kontrolera weryfikujący reCAPTCHA



Rysunek 3.6: Formularz weryfikacyjny reCAPTCHA.

Źródło: Opracowanie własne

3.7. Devise

W celu organizacji uprawnień i dostępu elementów strony przez upoważnione osoby zaimplementowano obsługę użytkowników. Całość polega na tym, aby dodać do naszego projektu Gem Devise[16], który jest udostępniony na licencji MIT License. Natomiast jego instalacja polega na dodaniu go do pliku Gemfile, następnie użyciu generatora wbudowanego w Gem za pomocą polecenia powłoki:

```
1 $ rails generate devise:install
```

Listing 3.25: Polecenie instalujące Gem Devise w naszym projekcie

Kolejnym krokiem jest wygenerowanie modelu za pomocą następnego polecenia powłoki:

```
1 $ rails generate devise users
```

Listing 3.26: Polecenie generujące modeul użytkowników w naszym projekcie

Na koniec instalacji należy wykonać migracje bazy danych.

W przypadku TrainCMS, na tym nie zakończyła się implementacja obsługi użytkowników. Wprowadzono uprawnienia administratora, polegało to na stworzeniu metody w kontrolerze użytkowników, która to modyfikuje pole w bazie danych. Następnie w miejscach do których ma mieć dostęp tylko administrator, dane pole jest sprawdzane pod kątem występowania twierdzenia, wówczas użytkownik otrzymuje dostęp. Stworzono dodatkową podstronę w panelu administratora w celu zarządzania użytkownikami, na tejże stronie możemy usunąć danego użytkownika, nadać mu uprawnienia administratora, przejść do edycji jego danych oraz przejść do formularza dodawania użytkowników. W tym miejscu należy wspomnieć o założenie, które twierdzi, że tylko administrator może dodawać użytkowników, nikt nie może się sam zarejestrować.

3.8. rQRcode i Prawn/QRCode

Aby ułatwić promocję strony zaimplementowana została funkcjonalność generowania kodów QR, w tym celu z pomocą przychodzi Gem rQRcode[17] i Gem Prawn/QRCode[18]. Pierwszy Gem jest głównym silnikiem kodów QR w naszym projekcie, natomiast drugi Gem odpowiada za poprawne osadzanie kodu w dokumencie PDF. Instalacja polega na dodaniu obu dodatków do pliku Gemfile. Następnie należy dodać do pliku `app/helpers/application.rb` linię:

```
require 'rqrcode'
```

Listing 3.27: Kod generujący kod QR

Generowanie kodu QR polega na wywołaniu w kodzie strony polecenia:

```
qrcode = RQRCode::QRCode.new( ‘random ’text )
```

Listing 3.28: Kod generujący kod QR

W systemie TrainCMS kody QR odpowiadają za prezentację odnośnika do strony danego artykułu. Umieszczone są one na wydruku artykłu oraz w pliku PDF z artykułem.

W celu wyświetlenia wygenerowanego kodu QR należy go skonwertować do formatu PNG za pomocą metody `as_png`:

```
qrimage = qrcode.as_png
```

Listing 3.29: Kod konwertujący kod QR

Następnie wyświetleniu go na stronie za pomocą polecenia:

```
<img src="<%=@qrimage.to_data_url %>" />
```

Listing 3.30: Kod wyświetlający kod QR na stronie HTML

Aby umieścić kod QR w dokumencie PDF nalży dodać linię:

```
render_qr_code(qrcode)
```

Listing 3.31: Kod wyświetlający kod QR w dokumencie PDF

Wygenerowany kod QR.

Zeskanuj, aby przejść do artykułu na stronie:

Rysunek 3.7: Wygenerowany kod QR.

Źródło: Opracowanie własne

3.9. Geocoder

Kolejnym ciekawym dodatkiem do projektu jest Gem Geocoder[19], udostępniany na licencji MIT license. Pozwala on na tak zwane geokodowanie, czyli ustalanie za pomocą na przykład adresu urzędowego współrzędnych geograficznych, tyczy się to również adresów IP z którymi radzi sobie nieco gorzej aniżeli z urzędowymi adresami.

Instalacja polega na dodaniu wpisów do pliku Gemfile. Każdy model, który ma zostać poddany geokodowaniu musi posiadać kolumny przeznaczone na przechowywanie długości i szerokości geograficznej. Następnie w pliku danego modelu dodajemy dwie linie kodu:

```
1 geocoded_by :adress
2 after_validation :geocode
```

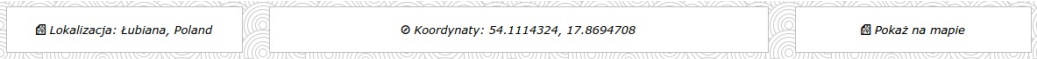
Listing 3.32: Kod odpowiadający za geokodowanie

Aby wyświetlić zlokalizowane koordynaty korzystamy z poleceń:

```
1 <%= @event.latitude %>, <%= @event.longitude %>
```

Listing 3.33: Kod wyświetlający koordynaty

Funkcjonalność geokodowania została wdrożona przy kalendarzu wydarzeń, podczas tworzenia wydarzenia podajemy jego adres, a Gem ustala jego koordynaty, następnie w podglądzie danego wydarzenia uzyskujemy odnośnik do mapy z zaznaczoną pozycją geograficzną.



Rysunek 3.8: Adres, koordynaty i odnośnik do mapy przy wydarzeniu

Źródło: Opracowanie własne

Innym miejscem wdrożenia geokodowania jest moduł komentarzy. Gem na podstawie adresu IP ustala przybliżone współrzędne geograficzne, jednak należy pamiętać, że jest to tylko informacja o sugerowanym położeniu w rzeczywistości zgadza się tylko kraj co i tak nie jest wiążące zwłaszcza kiedy użytkownik korzysta z usług VPN.

Rysunek 3.9: Koordynaty jako odnośnik do mapy przy komentarzach

Źródło: Opracowanie własne

3.10. cookies_eu

Zgodnie z nowelizacja Prawa Telekomunikacyjnego uchwalonego dnia 16 listopada 2012 każdy operator strony internetowej zobowiązany jest do umieszczenia stosownej informacji o przechowywaniu i operowaniu na plikach cookies, potocznie zwanymi ciasteczkami. Z pomocą w wykonaniu tych obostrzeń przychodzi Gem cookies_eu[20], udostępniany na licencji MIT license.

Instalacja polega na dodaniu wpisu do pliku Gemfile, następnie dodaniu do pliku app/assets/javascripts/application.js jednej linii kodu o następującej treści:

```
1 //= require cookies_eu
```

Listing 3.34: Kod odpowiadający dodanie bibliotek javascript

Natomiast do pliku app/asstes/stylesheets/aplication.css linii kodu o treści:

```
1 *= require cookies_eu
```

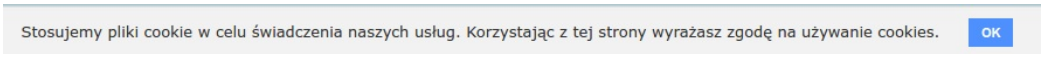
Listing 3.35: Kod odpowiadający dodanie bibliotek kaskadowych arkuszy stylów

Na koniec dodajemy linię, która będzie odpowiadać za renderowania paska ze stosowną informacją na dole strony:

```
1 <%= render 'cookies_eu/consent_banner' %>
```

Listing 3.36: Kod renderujący pasek z informacją o ciasteczkach

Wynik końcowy prezentuje się w następujący sposób:



Rysunek 3.10: Belka informująca o plikach cookies.

Źródło: Opracowanie własne

Bibliografia

[1] John Elder. Learn Ruby On Rails For Web Development: Learn Rails The Fast And Easy Way!. Codemy.com; 1 edition (January 19, 2015).

[2] Dan Chak. Enterprise Rails. O'Reilly Media; 1 edition (November 3, 2008).

[3] Użytkownicy Wikibooks. Ruby. Wikibooks; 1 edition (February 17, 2008).

[4] Oficjalna dokumentacja - Gem RailRoady. http://railroady.prestonlee.com/ (dostęp 23.04.2017)

[5] Oficjalna dokumentacja Aplikacji Astah. http://astah.net/tutorials (dostęp 23.04.2017)

[6] Oficjalna dokumentacja frameworku Ruby on Rails. http://guides.rubyonrails.org/ (dostęp 23.04.2017)

[7] Oficjalna dokumentacja API Ruby on Rails. http://api.rubyonrails.org/ (dostęp 23.04.2017)

[8] Oficjalny opis - Gem will_pagintate. http://www.rubydoc.info/gems/will_paginate/ (dostęp 23.04.2017)

[9] Oficjalna dokumentacja frameworku Foundation for Sites. http://foundation.zurb.com/sites/docs/ (dostęp 23.04.2017)

[10] Oficjalna dokumentacja - Gem Foundation Icon. http://www.rubydoc.info/gems/foundation-icons-sass-rails/ (dostęp 23.04.2017)

[11] Oficjalna dokumentacja - Gem CarrierWave. https://github.com/carrierwaveuploader/carrierwave/wiki (dostęp 23.04.2017)

[12] Oficjalna dokumentacja - Gem CKEditor for Rails. https://github.com/galetahub/ckeditor/ (dostęp 23.04.2017)

[13] Oficjalna dokumentacja - Gem PrawnPDF.
http://prawnpdf.org/api-docs/2.0/ (dostęp 23.04.2017)

[14] Oficjalna dokumentacja - Gem Chartkick.
https://github.com/ankane/chartkick/ (dostęp 23.04.2017)

[15] Oficjalna dokumentacja - Gem reCAPTCHA.
https://github.com/ambethia/recaptcha/ (dostęp 23.04.2017)

[16] Oficjalna dokumentacja - Gem devise.
https://github.com/plataformatec/devise/ (dostęp 23.04.2017)

[17] Oficjalna dokumentacja - Gem rQRcode.
http://www.rubydoc.info/gems/rqrcode/ (dostęp 23.04.2017)

[18] Oficjalna dokumentacja - Gem Prawn/QRCode.
http://www.rubydoc.info/gems/prawn-qrcode/ (dostęp 23.04.2017)

[19] Oficjalna dokumentacja - Gem Geocoder.
http://www.rubydoc.info/gems/geocoder/ (dostęp 23.04.2017)

[20] Oficjalna dokumentacja - Gem cookies_eu.
http://www.rubydoc.info/gems/cookies_eu/ (dostęp 23.04.2017)

Zakończenie

Podczas pracy nad projektem zrealizowałem wszystkie założone wcześniej cele, jedynie nie udało się osiągnąć pełnej responsywności w zakresie widoku kalendarza wydarzeń. Dzięki temu zyskałem duże doświadczenie w pracy nad średniej wielkości projektami informatycznymi. Do pracy wykorzystałem niemalże wszystkie nabyte w trakcie trwania studiów umiejętności. Koncepcja na rozwój projektu obejmuje rozszerzenie funkcjonalności systemu o możliwość dodawania komponentów z biblioteki Polymer.

DODATEK A

Płyta CD z kodem projektu

Płyta CD

DODATEK B

Płyta CD z plikami pracy licencjackiej

Płyta CD

Spis rysunków

Spis kodów źródłowych

Oświadczenie

Ja, niżej podpisany oświadczam, iż przedłożona praca dyplomowa została wykonana przeze mnie samodzielnie, nie narusza praw autorskich, interesów prawnych i materialnych innych osób.

.....	.....
data	podpis