

LABORATORIUM: SYSTEMY ZARZĄDZANIA TREŚCIĄ

CMS PIZZERIA

16 stycznia 2019

Szymon Kołodziejczak

Spis treści

1	Opis i schemat ogólnej struktury systemu CMS	2
2	Opis poszczególnych modułów systemu	2
3	Opis ról i uprawnień	3
4	Diagram funkcjonalności	3
5	Schemat struktury bazy danych wraz ze szczegółowym opisem	4
6	Opis zastosowanych technologii i rozwiązań	6
7	Instrukcja instalacji	7
8	Dokumentacja implementacji nowych modułów systemu.	8
9	Skrócona instrukcja użytkownika, umożliwiająca wdrożenie osoby do obsługi i zarządzania systemem CMS	11
10	Link do repozytorium projektu	11

1 OPIS I SCHEMAT OGÓLNEJ STRUKTURY SYSTEMU CMS

CMS jest stroną-wizytówką dla Pizzeri. System nie skupia się na płatności online jedynie na prezentowaniu treści przygotowanych przez administratora systemu. Cała treść jest generowana wyłącznie przez autoryzowaną osobę w panelu administracyjnym - użytkownik / gość jest jedynie odbiorcą treści. Możliwe jest jednak skontaktowanie się z administratorem poprzez formularz kontaktowy. W tym celu nie jest wymagane żadne konto w systemie.

Ogólna struktura systemu jest oparta na prostych zasadach: bazowa aplikacja/moduł implementuje wygląd (układ strony) oraz postać nawigacji. Wszystkie dodatkowe aplikacje są podstronami dodającymi właściwą treść (poprzez implementację bazowego layoutu). W kolejnym podpunkcie wszystkie zaimplementowane moduły systemu zostały dokładnie opisane.

2 OPIS POSZCZEGÓLNYCH MODUŁÓW SYSTEMU

Projekt składa się z 6 aplikacji - poniżej każda z nich została krótko scharakteryzowana.

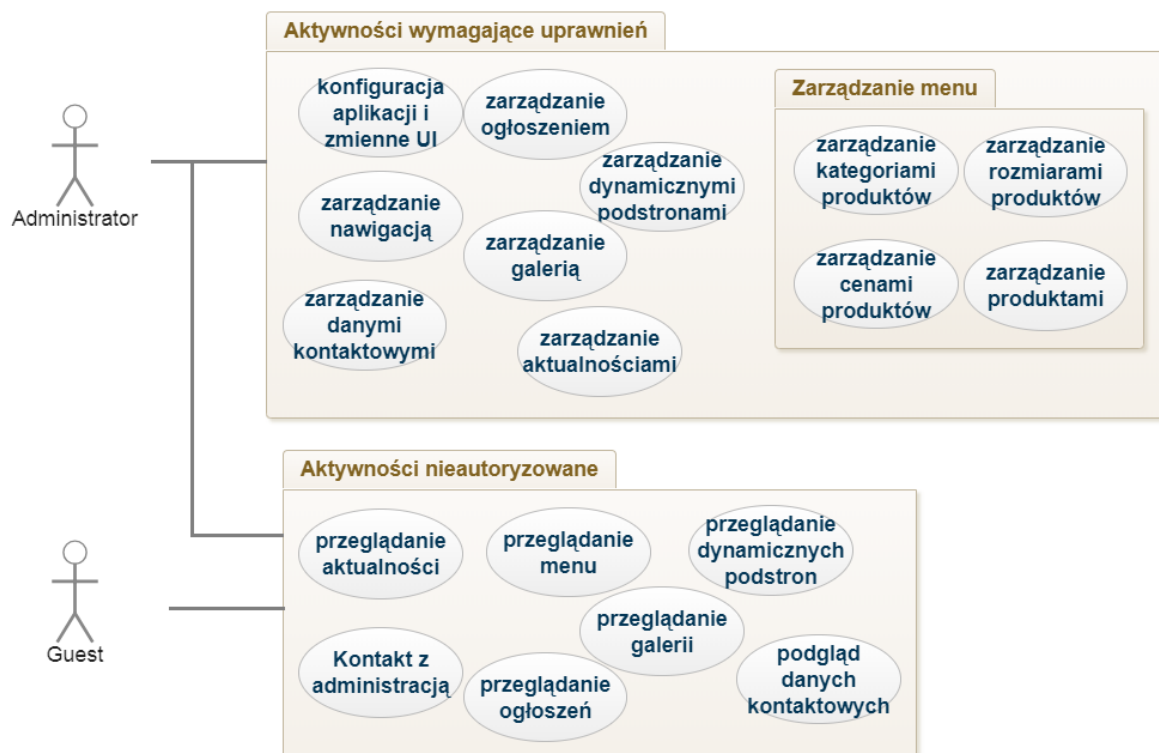
- Moduł bazowy - odpowiada za podstawową konfigurację strony: tytuł, logo, strona główna, tło, kolory i rozmiar czcionki. Poza konfiguracją, pozwala na definiowanie ogłoszenia oraz zarządzanie nawigacją.
- Moduł aktualności - definiowanie aktualności. Oferuje widok wylistowanych aktualności jak i podgląd konkretnej aktualności. Widok wylistowanych aktualności wspiera paginację.
- Moduł galerii - pozwala administratorowi na dodawanie zdjęć wraz z ich opisem. Wszystkie zdjęcia są wylistowane w jednym widoku w formie kart-kafelków.
- Moduł menu - definiowanie kategorii produktów, samych produktów oraz ich cen w konkretnym rozmiarze. Podane dane są prezentowane za pomocą tabeli w zakładce menu. Widok menu jest poprzedzony widokiem wyboru kategorii, a sama tabela menu może być poprzedzona tekstem jak również zawierać stópkę.
- Moduł kontakt - bazuje na dwóch tabelach Contact i Message. Dane podane w konfiguracyjnej tabeli Contact wygenerowane zostaną w widoku. Podstrona jest typowa: zawiera formularz kontaktowy, podstawowe informacje w celu skontaktowania się oraz mapę. Wiadomości wysłane poprzez formularz trafiają do tabeli message - podgląd w panelu administracyjnym tej tabeli jest w istocie podglądaniem skrzynki pocztowej".

- Moduł dynamicznych podstron - administrator może stworzyć dowolną podstronę składającą się z tytułu oraz treści. Treść definiuje się poprzez Rich Text Editor, który umożliwia m.in. formatowanie czy dodawanie zdjęć - tak więc administrator ma pełną swobodę w definiowaniu treści podstrony. Stworzoną stronę można podpiąć do nawigacji lub np. ustawić jako stronę startową.

3 OPIS RÓL I UPRAWNIENÍ

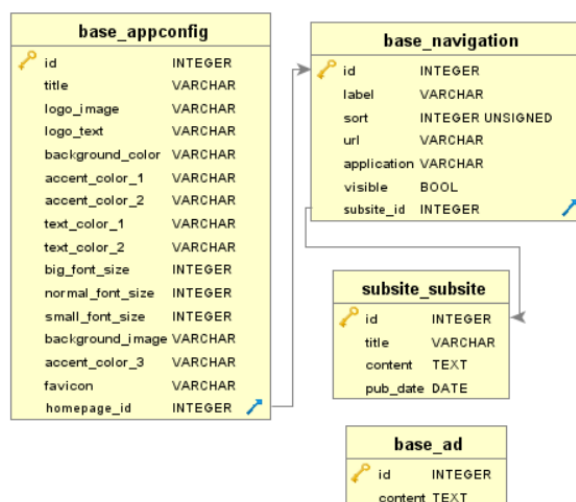
Aplikacja ma charakter rozbudowanej strony-wizytówki. Zgodnie z diagramem funkcjonalności wyróżnia się dwie role: administratora i gościa-użytkownika. Aplikacja nie przewiduje możliwości rejestracji konta zwykłego użytkownika, ani żadnych aktywności, które mogłyby być zarezerwowane dla niego. Za całą treść i jej prezentację odpowiada administrator.

4 DIAGRAM FUNKCJONALNOŚCI



Rysunek 1: Diagram przypadków użycia systemu.

5 SCHEMAT STRUKTURY BAZY DANYCH WRAZ ZE SZCZEGÓŁOWYM OPISEM



Rysunek 2: Schemat bazy danych bazowej aplikacji i jej zależności.

Tabela base-appconfig zawiera podstawową konfigurację strony. Pola tabeli można pogrupować na dwie kategorie:

- UI - kolory i rozmiar tekstu, tło, logo, favicon.
- reszta: tytuł strony, nazwa firmy, strona główna.

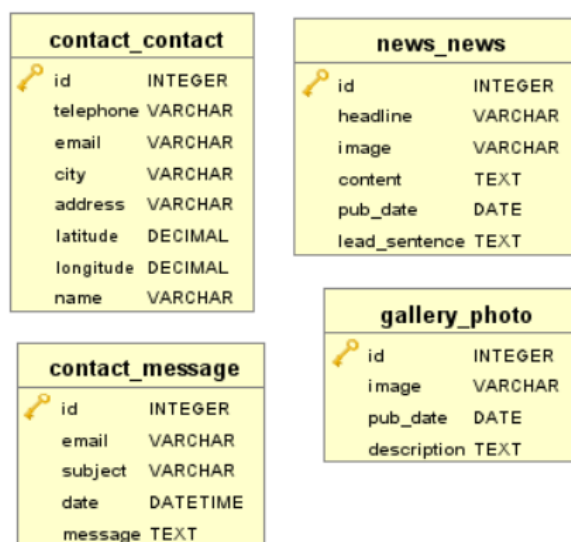
Tabela base-ad pozwala na dodanie jednego wpisu będącego ogłoszeniem tekstowym. Jedyne wpisy można usunąć bądź zedytować. Jego obecność decyduje o wygenerowaniu dodatkowego bloku w layoutcie strony.

Aplikacja bazowa implementuje nawigację. Wpisy nawigacji mogą odwoływać się do podstrony (relacja 0 lub jeden do wielu). Subsite-subsite, czyli aplikacja podstrony, jest jedyną aplikacją z zależnościami. Technicznie, mogłaby działać bez tej zależności, jednak zamysł projektowy był taki aby routing był obsługiwany w jednym miejscu (bazowej aplikacji).

W panelu administracyjnym administrator może utworzyć wpis nawigacji dotyczący wyłącznie podstrony. Pole 'application' jest ukryte w formularzu zarządzania, a jego obsługa jest zarezerwowana wyłącznie dla programisty. Wpis dotyczący aplikacji nie może zostać usunięty, jednak jego ukrycie będzie miało takie same konsekwencje dla użytkownika końcowego.

Podstrona składa się z tytułu oraz treści. Typ danych TEXT budzi wątpliwości atrakcyjności takiego rozwiązania, jednak Użytkownik mając do dyspozycji RichTextEditor może wstawiać zdjęcia, tabelę czy formatować tekst.

Część aplikacji (galeria, kontakt, news) jest zupełnie niezależnych, tabele powstałe na ich potrzeby nie wykazują żadnych relacji (rysunek 3).

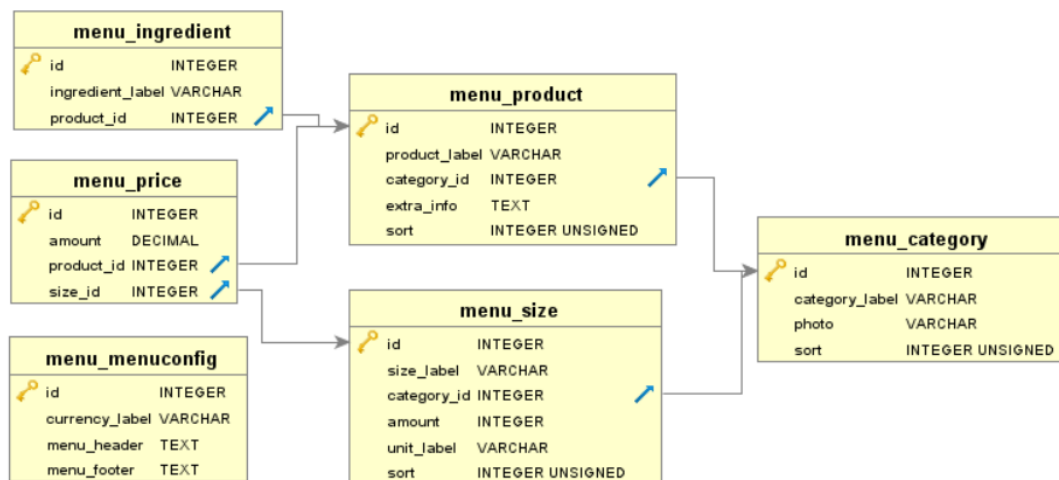


Rysunek 3: Część tabeli bazy danych poszczególnych aplikacji jest niezależnych.

Tabela contact-message jest miejscem do którego trafiają wiadomości po wypełnieniu formularzu na podstronie kontakt. Tabela contact-contact jest tabelą która zgodnie z założeniami przyjmuje tylko jeden wpis - dane kontaktowe organizacji (adres, telefon, email) wypełnione w panelu administracyjnym zostaną wyrenderowane na podstronie kontaktu.

Tabela news-news - aktualność składa się z: nagłówek, daty, zdjęcia, wstępu oraz właściwego tekstu. Treść "content" jest obsługiwana przez zewnętrzną aplikację tinyMCE, w tabeli jest to html przechowywany jako typ danych TEXT.

Galeria - zdjęcie oraz podpis. Warto wspomnieć, że w bazie danych nie są przechowywane zdjęcia, zapisywane są jedynie metadane dotyczące pliku. Fizyczny plik znajduje się w konkretnym katalogu projektu wskazanym podczas definiowania modelu.



Rysunek 4: Tabele używane przez aplikację menu.

Aplikacja menu jest najbardziej złożoną aplikacją w kontekście schematu bazy danych. Do jej implementacji użyto 6 tabel. Są to kolejno: składniki, cena, produkt, rozmiar, kategoria oraz konfiguracja.

Tabela menu-menuconfig pozwala skonfigurować tekst znajdujący się nad i pod wizualizacją produktów. W zamyśle autora może to posłużyć do podania dodatkowych informacji dotyczących produktów, promocji itp. To również tutaj administrator ustawia walutę w której prezentowane są ceny.

Produkt musi zostać przypisany do konkretnej kategorii. Rozmiary definiowane są dla konkretnej kategorii produktu. Pizza w rozmiarze XXL będzie miała inną cenę niż w rozmiarze M. Cena jest więc zależna od produktu i jego rozmiaru. Składniki są powiązane z produktem relacją wielu do jednego.

6 OPIS ZASTOSOWANYCH TECHNOLOGII I ROZWIĄZAŃ

Projekt nie korzysta z żadnych dodatkowych frameworków po stronie front-endu. HTML, CSS zostały w całości zaprojektowane przez autora. Autor w tym celu skorzystał z najnowszych technologii CSS - takich jak zmienne czy system flexbox. Skorzystano z metodologii BEM - szereg prostych zasad definiowania reguł CSS, których to trzymanie się usprawnia pracę i zwiększa czytelność kodu.

Do strony dodano Google Analytics – internetowe narzędzie do analizy statystyk serwisów WWW, udostępniane przez firmę Google. To narzędzie pozwala na generowanie około 80 rodzajów raportów związanych z zbieraniem danych dotyczących ruchu internetowego;

istnieje możliwość tworzenia segmentów użytkowników według źródła ruchu lub zachowania w serwisie.

Po stronie serwera skorzystano z frameworka Django, który dostarcza zdecydowaną ilość gotowych rozwiązań (panel administracyjny, autoryzacja, ORM). Do django doinstalowano następujące paczki:

- django-phonenumbers-field (walidacja numeru telefonu),
- django-tinymce (aplikacja rich text editor - bogaty edytor tekstowy pozwalający m.in na formatowanie tekstu czy dodawanie załączników)
- django-colorful (definiowanie koloru - dodaje obsługę definiowania koloru przy użyciu natywnej kontrolki HTML w panelu administracyjnym)
- django-admin-sortable2 (sortowanie w panelu administracyjnym przy pomocy drag-drop)

Dane konfiguracyjne aplikacji są przechowywane jako jedyny wpis tabeli, który może być wyłącznie edytowany. Ścieżki otwierania takich konfiguracji są nadpisane i domyślnie od razu przekierowują do edycji jedynego elementu.

Po dodaniu nowego rozmiaru lub nowego produktu tworzone są wpisy w tabeli cen (wykorzystanie django-signal jako alternatywa triggerów bazodanowych). Można je tylko i wyłącznie edytować. Panel administracyjny jest często wzbogacony w filtry widoku czy edycję liniową.

Za routing aplikacji odpowiada administrator modyfikując wpisy nawigacji. Wpisy dotyczące wbudowanych aplikacji są dodawane programistycznie i nie można usunąć, można je jedynie ukryć. Wpisy dotyczące dynamicznych podstron można dowolnie edytować. Kolejność wpisów nawigacji decyduje tylko i wyłącznie o kolejności renderowania w menu.

7 INSTRUKCJA INSTALACJI

Cały projekt jest zaimplementowany w Django. Django to framework napisany w Python, dzięki czemu aplikacja będzie działać na różnych platformach. Python jest konieczny do uruchomienia projektu - wymagana jest jego wcześniejsza instalacja w dowolnej wersji (2.7 - 3.x).

W celu zainstalowania aplikacji należy pobrać kod dostępny w publicznym repozytorium github - link znajduje się w ostatniej sekcji dokumentacji. Kod pobrany bezpośrednio lub

przez git wymaga zainstalowania kilku zależności przed jego uruchomieniem. W tym celu należy wpisać następujące komendy:

```
pip install django-phonenumbers-field
pip install phonenumbers
pip install Pillow
pip install django-tinymce
pip install django-colorful
pip install django-admin-sortable2
```

Wszystkie zależności zostały już opisane we wcześniejszej sekcji. Po ich zainstalowaniu należy przejść do katalogu projektu i wpisać następującą komendę uruchamiającą serwer:

```
python manage.py runserver.
```

Administrator nowej instancji produktu ma już w pełni gotową stronę-wizytówkę wypełnioną danymi tak więc zna jej potencjał już od samego początku pracy. Każda aplikacja projektu ma kod odpowiedzialny za wygenerowanie przykładowych danych. Całą treść definiuje się w panelu administracyjnym. Każda aplikacja prezentuje jedynie treści wygenerowane przez administratora, nie ma w niej fragmentów stricte hard-coded.

Repozytorium posiada bazę danych w której istnieje administrator o następujących danych logowania:

- login: admin,
- hasło: admin

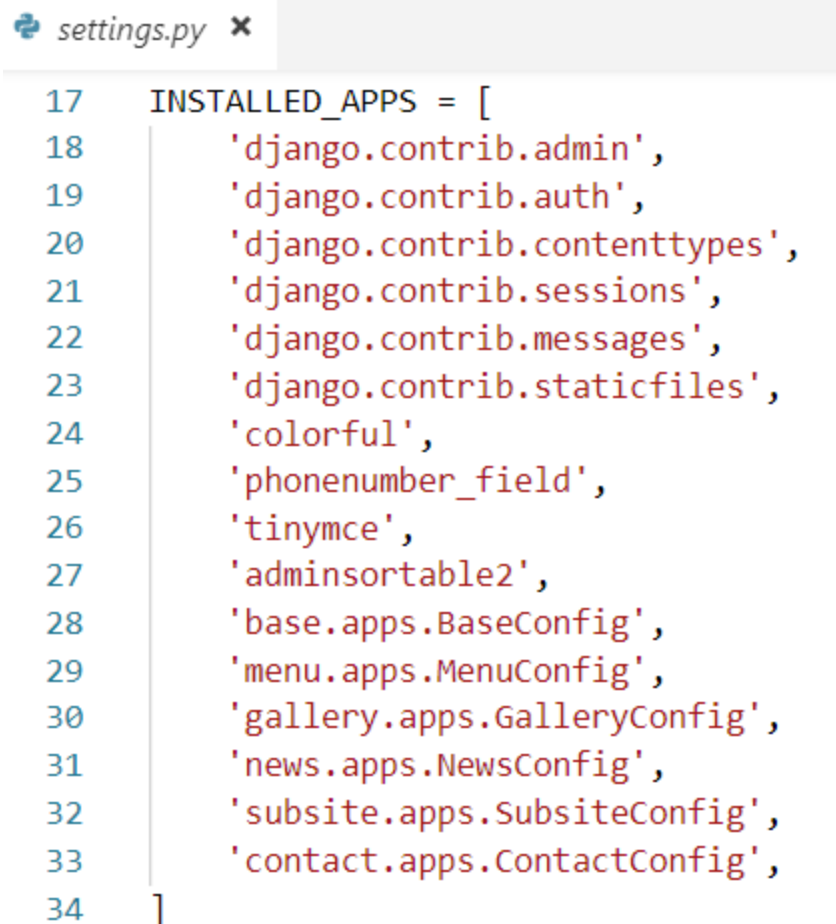
Po zalogowaniu się mamy dostęp do edycji i zarządzania całą treścią strony. W celu stworzenia dodatkowego administratora należy wpisać następującą komendę:

```
python manage.py createsuperuser
```

a następnie postępować wg. instrukcji wyświetlanej w konsoli.

8 DOKUMENTACJA IMPLEMENTACJI NOWYCH MODUŁÓW SYSTEMU.

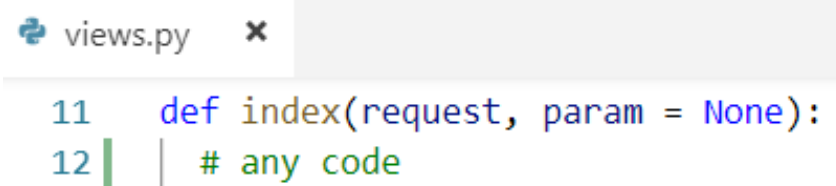
Nowo utworzoną aplikację należy dodać do listy zainstalowanych aplikacji w pliku settings.py projektu (rysunek 5).



```
settings.py x
17  INSTALLED_APPS = [
18      'django.contrib.admin',
19      'django.contrib.auth',
20      'django.contrib.contenttypes',
21      'django.contrib.sessions',
22      'django.contrib.messages',
23      'django.contrib.staticfiles',
24      'colorful',
25      'phonenumbers_field',
26      'tinymce',
27      'adminsortable2',
28      'base.apps.BaseConfig',
29      'menu.apps.MenuConfig',
30      'gallery.apps.GalleryConfig',
31      'news.apps.NewsConfig',
32      'subsite.apps.SubsiteConfig',
33      'contact.apps.ContactConfig',
34  ]
```

Rysunek 5: Lista zarejestrowanych aplikacji projektu.

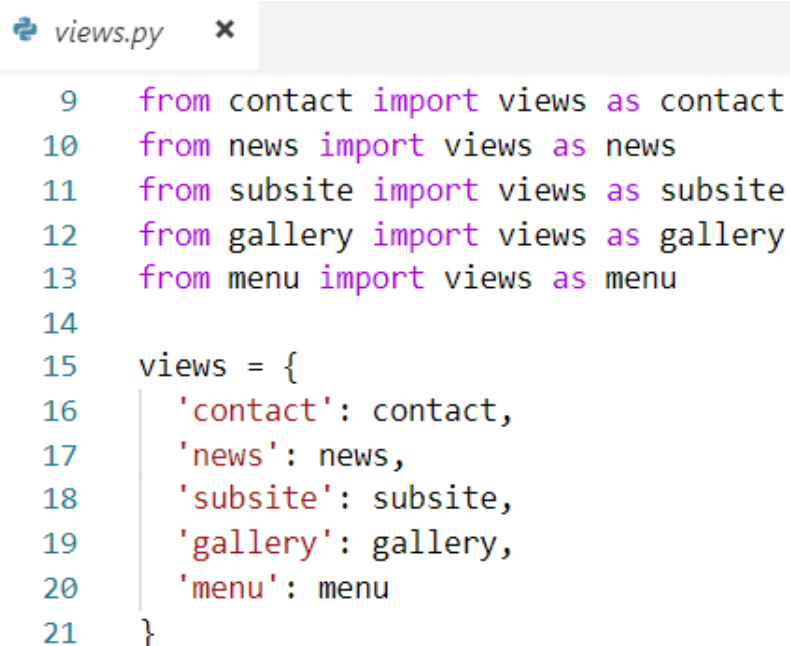
Następnie, w pliku views utworzonej aplikacji należy zaimplementować funkcję index przyjmującą dwa parametry: request oraz param (rysunek 6).



```
views.py x
11  def index(request, param = None):
12  |  |  # any code
```

Rysunek 6: Funkcja startowa wywoływana przez aplikację bazową. Param to parametr url.

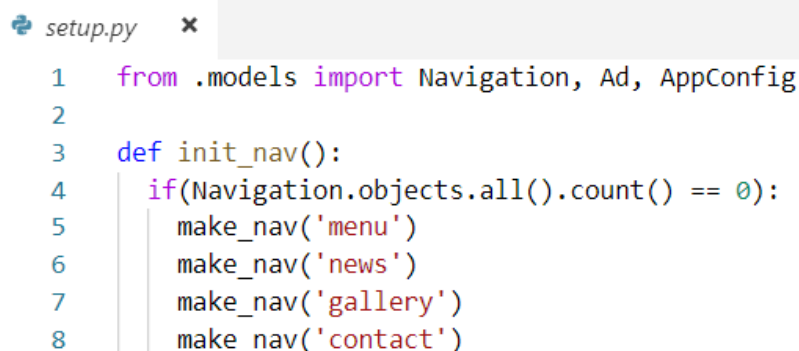
Aby zdefiniowana funkcja index została wywołana przez aplikację bazową należy ją 'zarejestrować' w pliku base.views. Wspomniany proces polega na rozszerzeniu słownika 'views' w następujący sposób: klucz to nazwa aplikacji, wartość to referencja do pliku views nowo stworzonej aplikacji (rysunek 7):



```
views.py x
9  from contact import views as contact
10 from news import views as news
11 from subsite import views as subsite
12 from gallery import views as gallery
13 from menu import views as menu
14
15 views = {
16     'contact': contact,
17     'news': news,
18     'subsite': subsite,
19     'gallery': gallery,
20     'menu': menu
21 }
```

Rysunek 7: Rejestracja funkcji startowej dodawanej aplikacji.

Finalnie, aby aplikacja mogła "urzeć światło dzienne" musi znaleźć się na liście nawigacji. Dodanie takiego wpisu może odbywać się jedynie programistycznie (rysunek 8). W tym celu należy wzbogacić implementację funkcji "init-nav()" o kolejne wywołanie "make-nav(nazwaAplikacji)". Nazwa aplikacji musi być identyczna jak ta podana w słowniku (rysunek 7).



```
setup.py x
1  from .models import Navigation, Ad, AppConfig
2
3  def init_nav():
4      if(Navigation.objects.all().count() == 0):
5          make_nav('menu')
6          make_nav('news')
7          make_nav('gallery')
8          make_nav('contact')
```

Rysunek 8: Rejestracja funkcji startowej dodawanej aplikacji.

Po tych krokach, po przejściu pod adres `host:port/page/nazwaAplikacji` zostanie wywołana funkcja `index` nowo utworzonej aplikacji, a sam adres znajdzie się na liście nawigacji.

9 SKRÓCONA INSTRUKCJA UŻYTKOWNIKA, UMOŻLIWIAJĄCA WDRO- ŻENIE OSOBY DO OBSŁUGI I ZARZĄDZANIA SYSTEMEM CMS

Po zainstalowaniu aplikacji użytkownik ma dostęp do już w pełni wypełnionej danymi strony. Po przejściu na adres url `host/admin/` i zalogowaniu się w panelu administracyjnym można edytować dowolną treść. Wszystkie poprzednie sekcje uzupełniają wiedzę potrzebną do wdrożenia i mogą posłużyć za dalszy poradnik.

10 LINK DO REPOZYTORIUM PROJEKTU

https://github.com/kolodziejczak-sz/cms_pizza