19 czerwca 2023

Michal Piaszczyk

Agreagator Aktualności  
Praca zaliczeniowa na kierunku python developer wsb merito

# Cel powstania aplikacji

Celem powstania aplikacji jest ułatwienie śledzenia aktualności pojawiających się na wybranych stronach internetowych. Są to głównie strony rządowe oraz strony internetowe agencji odpowiedzialnych za wdrażanie w Polsce funduszy europejskich. Większość z nich nie obsługuje przystępnych kanałów śledzenia nowości pojawiających się na nich (np. kanałów RSS). Z tego powodu korzystanie z niniejszej aplikacji znacznie ułatwia bieżące zapoznawania się z informacjami prezentowanymi na zaimplementowanych do aplikacji stronach internetowych.

Stworzona przeze mnie aplikacja automatycznie, po uruchomieniu serwera pobiera zawartość stron internetowych, pobiera z nich newsy i przedstawia je w czytelny sposób za pomocą frameworka Django.

Lista obsługiwanych stron internetowych została przedstawiona w tabeli poniżej.

|  |  |  |
| --- | --- | --- |
| **Adres** | **Skrót wykorzystywany w aplikacji** | **Opis Strony** |
| //trzydoliny.eu' | 'LGD', | Lokalna Grupa Działania Trzy Doliny |
| //www.gov.pl/web/cppc/lista-aktualnosci' | 'CCCP', | Centrum Projektów Polska Cyfrowa |
| //www.gov.pl/web/uw-kujawsko-pomorski/wiadomosci2' | 'Urząd Wojewódzki', | Urząd Wojewódzki Województwa Kujawsko-Pomorskiego |
| //www.gov.pl/web/fundusze-regiony/aktualnosci1' | 'Ministerstwo Funduszy', | Ministerstwo Funduszy i Polityki Regionalnej |
| //www.gov.pl/web/sport/wiadomosci' | 'Ministerstwo Sportu', | Ministerstwo Sportu |
| //www.gov.pl/web/premier/wydarzenia' | 'Premier', | Strona internetowa premiera |
| //www.gov.pl/web/infrastruktura/aktualnosci' | 'Ministerstwo Infrastruktury', | Ministerstwo Infrastruktury |
| //www.gov.pl/web/edukacja-i-nauka/wiadomosci' | 'Ministerstwo Edukacji', | Ministerstwo Edukacji |
| //www.gov.pl/web/sport/dofinansowanie-zadan-z-funduszu-rozwoju-kultury-fizycznej-201948' | 'Fundusz Rozwoju Kultury Fizycznej', | Ministerstwo Sportu - Strona poświęcona środkom finansowym |
| //www.funduszeeuropejskie.gov.pl/strony/wiadomosci#/najnowsze=1' | 'Fundusze Europejskie', | Fundusze Europejskie - portal dla zaintreresowanych |
| //www.funduszeeuropejskie.gov.pl/strony/o-funduszach/fundusze-2021-2027/aktualnosci/#/domyslne=1' | 'Fundusze Europejskie -  Aktualności', | Fundusze Europejskie - portal dla zaintreresowanych |
| //www.polskacyfrowa.gov.pl/strony/o-programie/fundusze-europejskie-na-rozwoj-cyfrowy-2021-2027/aktualnosci/#/domyslne=1' | 'Fundusze na rozwój cyfrowy', | Fundusze Europejskie - portal dla zaintreresowanych |
| //www.poir.gov.pl/strony/o-programie/fe-dla-nowoczesnej-gospodarki/aktualnosci/#/domyslne=1' | 'Nowoczesna Gospodarka', | Fundusze Europejskie - portal dla zaintreresowanych |
| //www.rozwojspoleczny.gov.pl/strony/aktualnosci#/domyslne=1' | 'Rozwoj Społeczny', | Fundusze Europejskie - portal dla zaintreresowanych |
| //www.rozwojcyfrowy.gov.pl/strony/aktualnosci/#/domyslne=1' | 'Fundusze na rozwój cyfrowy aktualności', | Fundusze Europejskie - portal dla zaintreresowanych |
| //www.nowoczesnagospodarka.gov.pl/strony/aktualnosci/#/domyslne=1' | 'Nowoczesna Gospodarka Aktualności', | Fundusze Europejskie - portal dla zaintreresowanych |
| //www.feniks.gov.pl/strony/aktualnosci/#/domyslne=1' | 'Feniks', | Fundusze Europejskie - portal dla zaintreresowanych |
| //mojregion.eu/rpo/wiadomosci/' | 'Mojregion', | Fundusze Europejskie - portal dla zaintreresowanych |
| //mojregion.eu/prow/aktualnosci/' | 'PROW', | Fundusze Europejskie - portal dla zaintreresowanych |
| //efrwp.pl/' | 'EFRPW', | Europejski Fundusz Rozwoju Wsi Polskiej |
| //portal-sow.pfron.org.pl/opencms/export/sites/pfron-sow/sow/aktualnosci/' | 'SOW', | Portal PFRON |
| //bip.kujawsko-pomorskie.pl/' | 'BIP K-P' | BIP Województwa Kujawsko Pomorskiego |

# Instalacja

Pakiety niezbędne do właściwego używania aplikacji zostały zapisane w pliku requirements.txt znajdującym się w folderze głównym aplikacji.

W repozytorium GIT znajdują się następujące foldery:

net – jest to moja maszyna wirtualna

nws – katalog w którym znajduje się projekt i aplikacja Django

nws/nws – katalog projektu Django

nws/nius – katalog aplikacji Django

Do bezproblemowego korzystania z aplikacji konieczne jest właściwe ustawienie ścieżek do plików w następujących plikach:

nius/newsy.py

w wierszu 16 oraz 273

należy podać ścieżkę do pliku nws\nius\config.ini

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie



nius/views.py

w wierszu 64

należy podać ścieżkę do pliku nws\nius\config.ini

Zarządzanie aktualnościami następuje z poziomu strony /admin

# Funkcje agregujące aktualności

Istotą działania tej aplikacji jest pobieranie danych ze stron internetowych, zamienianie ich na ustandaryzowane listy słowników o następującej strukturze:

news = {'title':tytuł newsa,'date': data publikacji ,'link': link do strony internetowej zawierającej danego newsa,'desc': opis newsa znajdujący się na stronie głównej (o ile jest dostępny),'source': źródło z którego news został pobrany}

na podstawie listy słowników w kolejnym kroku nowe wpisy są automatycznie dodawane do modelu danych Django.

Poniżej znajduje się opis kodu znajdującego się w pliku newsy.py

config = configparser.ConfigParser()

config.read(config.ini')

start\_date\_str = config.get('DEFAULT', 'start\_date')

start\_date = datetime.strptime(start\_date\_str, '%Y-%m-%d')

Opis działania

1. Importowanie modułów configparser i datetime.
2. Tworzenie obiektu config klasy ConfigParser - obiekt ten jest używany do parsowania plików konfiguracyjnych w formacie INI.
3. Wywołanie metody read obiektu config z podaną ścieżką do pliku konfiguracyjnego. Ta metoda odczytuje i analizuje zawartość pliku konfiguracyjnego.
4. Wywołanie metody get obiektu config z dwoma argumentami: sekcją 'DEFAULT' i kluczem 'start\_date'. Ta metoda pobiera wartość odpowiadającą podanemu kluczowi z określonej sekcji pliku konfiguracyjnego.
5. Przypisanie odczytanej wartości daty jako tekst do zmiennej start\_date\_str.
6. Wywołanie funkcji strptime z dwoma argumentami: start\_date\_str - odczytany tekst reprezentujący datę i '%Y-%m-%d' - format daty, do którego zostanie przekonwertowany tekst.
7. Przypisanie skonwertowanej daty do zmiennej start\_date.

W efekcie, po wykonaniu tego fragmentu kodu, zmienna start\_date będzie zawierać wartość daty początkowej odczytanej z pliku konfiguracyjnego, w formacie datetime.

Przykład:

python

start\_date = datetime.datetime(2023, 6, 15, 0, 0)

Wartością start\_date jest obiekt datetime, który reprezentuje datę 15 czerwca 2023 roku.

Kroki te są używane w celu pierwszej selekcji newsów, które są dodawane do modelu danych. Po uruchomieniu wszystkich funkcji z pliku newsy.py program zapisuje do pliku config.ini aktualną datę, dzięki czemu po kolejnym uruchomieniu do modelu mogą zostać dodane jedynie newsy opublikowane na stronach internetowych nie wcześniej niż w dniu zapisanym w pliku config.ini

Zapis w kodzie wygląda następująco:

new\_start\_date = datetime.now().date()

config.set('DEFAULT', 'start\_date', new\_start\_date.strftime('%Y-%m-%d'))

with open(config.ini', 'w') as f:

    config.write(f)

Ogólne działanie wszystkich funkcji zawartych w pliku newsy.py

1. Analizowanie zawartości strony przy użyciu biblioteki BeautifulSoup.
2. Tam gdzie proste pobranie zawartości strony za pomocą BeautifulSoup nie było możliwe, wykorzystano bibliotekę Selenium, która uruchamia przeglądarkę internetową, aby załadować stronę internetową, a następnie przekazuje zawartość html do biblioteki BeautifulSoup.
3. Analiza przekazanego html w celu odnalezienia newsa, jego tytułu, opisu, daty publikacji i linku do newsa. Z uwagi na ograniczoną długość tego dokumentu nie będę opisywał poszczególnych stron internetowych. Każda z nich jest inna i inne tagi html odpowiadają za poszczególne infromacje.
4. Pobrane daty (tekstowe) zostały przekonwertowane na obiekty datetime
5. Odrzucenie wpisów starszych niż data pobrana z plku config
6. Zwrócenie listy słowników zawierających poszczególne newsy w przedstawionym wcześniej standardzie.

# Model Danych

W projekcie Djano wdrożono następujący model danych

class News(models.Model):

    tytul = models.CharField(max\_length=100)

    opis = models.TextField()

    url = models.URLField()

    data = models.DateTimeField()

    source = models.URLField()

    is\_new = models.BooleanField(default=True)

    day = models.DateField()

    def \_\_str\_\_(self):

        return self.tytul

    class Meta:

        verbose\_name\_plural ="Newsy"

Klasa News reprezentuje model danych dla wiadomości. Posiada następujące pola:

* tytul (CharField): pole tekstowe o maksymalnej długości 100 znaków, przechowuje tytuł wiadomości.
* opis (TextField): pole tekstowe o zmiennej długości, przechowuje opis wiadomości.
* url (URLField): pole przechowujące adres URL wiadomości.
* data (DateTimeField): pole przechowujące datę i czas wiadomości.
* source (URLField): pole przechowujące adres URL źródła wiadomości.
* is\_new (BooleanField): pole logiczne, domyślnie ustawione na True, określa czy wiadomość jest nowa.
* day (DateField): pole przechowujące tylko datę wiadomości.

Metoda \_\_str\_\_ zwraca tytuł wiadomości, co jest przydatne podczas reprezentowania obiektu w postaci tekstowej.

Klasa Meta wewnątrz modelu zawiera metadane. W tym przypadku, ustawiamy verbose\_name\_plural na "Newsy", co jest nazwą w liczbie mnogiej dla modelu w interfejsie administratora Django.

trzy\_doliny = trzy\_doliny()

rzadowe = rzadowe()

unijne = unijne()

moj = mojrerpo()

prow = mojprow()

efr = efr()

sow = sow()

bip = bip()

all\_list = [moj, prow, trzy\_doliny, rzadowe, unijne, efr, sow, bip]

newsy\_wszystkie=[]

for x in all\_list:

    newsy\_wszystkie.extend(x)

for news in newsy\_wszystkie:

    if not News.objects.filter(url=news['link']).exists():

        News.objects.create(

            tytul=news['title'],

            opis=news['desc'],

            url=news['link'],

            data=news['date'],

            source = news['source'],

            day = news['date']

        )

Powyższy fragment kodu wykonuje następujące operacje:

1. Wywołuje funkcje trzy\_doliny, rzadowe, unijne, mojrerpo, mojprow, efr, sow i bip i przypisuje ich wyniki do zmiennych o tych samych nazwach.
2. Tworzy listę all\_list, zawierającą wszystkie wcześniej otrzymane wyniki.
3. Inicjalizuje pustą listę newsy\_wszystkie.
4. Iteruje przez elementy listy all\_list i rozszerza newsy\_wszystkie o zawartość każdego elementu.
5. Dla każdej wiadomości w newsy\_wszystkie sprawdza, czy już istnieje w bazie danych na podstawie pola url. Jeśli nie istnieje, tworzy nowy obiekt klasy News za pomocą metody create() obiektu News.objects.
6. Przypisuje wartości pól tytul, opis, url, data, source i day wiadomości do odpowiednich pól nowo utworzonego obiektu News.

W efekcie, ten fragment kodu dodaje nowe wiadomości do bazy danych na podstawie danych otrzymanych z różnych źródeł.

# Wyświetlanie danych za pomocą frameworka Django

Routing w projekcie jest obsługiwany przez plik nws/urls.py

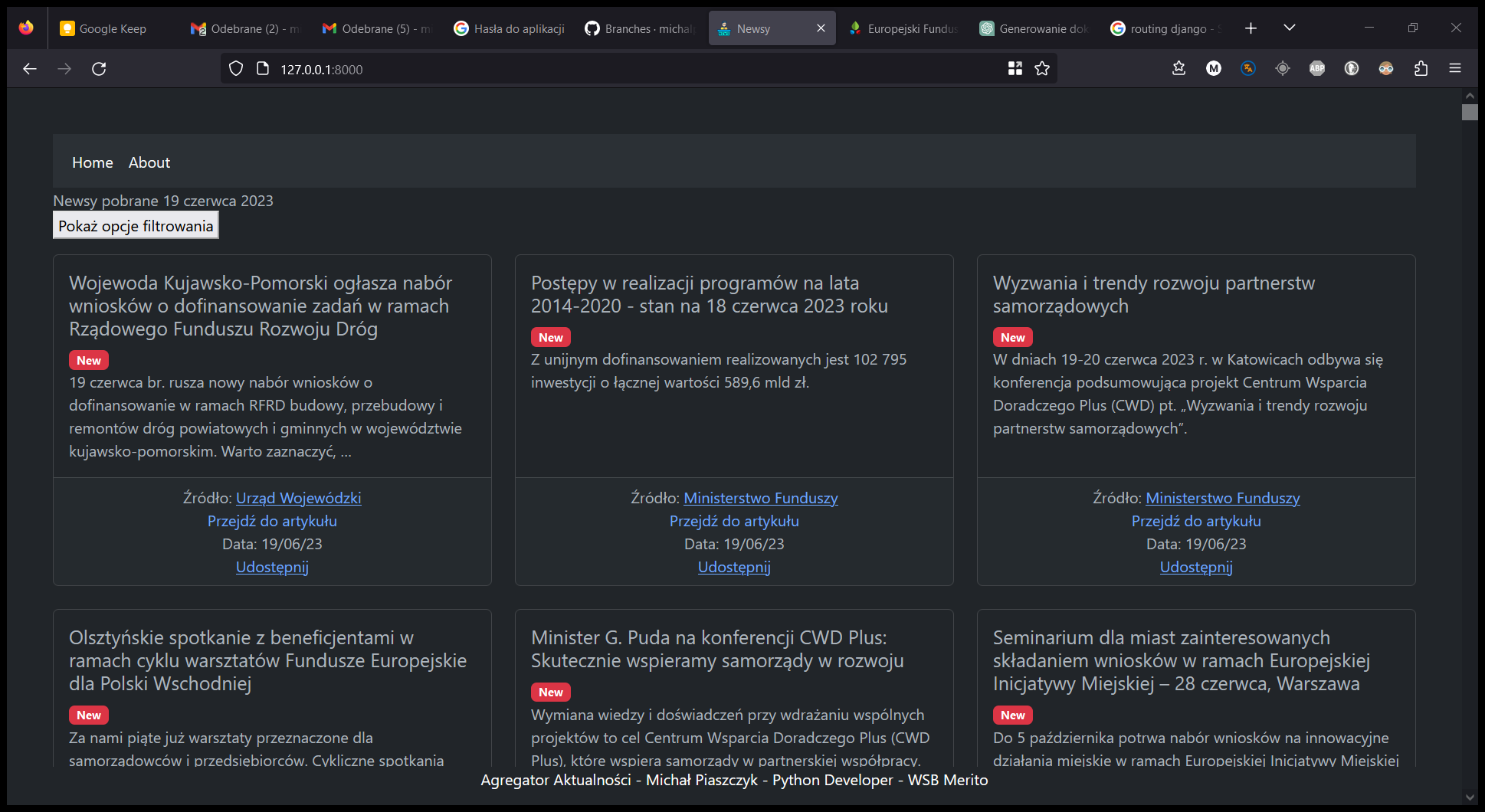
Widoku są zapisane w nius/views.py

Pliki html są zapisane w katalogu nius/templates

Do prezentowania newsów wykorzystano szablon bootstrap. Zastosowano motyw ciemny bootstrap, jako lepszy do przeglądania tekstu.

W celu czytelnej prezentacji wyników wykorzystano karty dostępne w bootstrap. Karty zostały ułożone po trzy w jednym rzędzie. Dzięki wykorzystaniu stopki karty, mają one tę samą szerokość w każdym wierszu. Zasługą wykorzystania bootstrap jest również autoresponsywność strony. Strona ma również nagłówek i stopkę, wykorzystano rozszerzenie każdej strony za pomocą pliku base.html

Przykładowa prezentacja strony



Każda karta zawiera:

* tytuł (pole title modelu),
* opis (pole opis modelu),
* artykuły z dzisiejszego dnia posiadają badge New, który znika po upływie jednego dnia

a w stopce

* Źródło (należy zwrócić uwagę że źródło jest skrótem pochodzącym z pliku constants.py),
* link do artykułu (pole url modelu)
* datę (pole date modelu odpowiednio sformatowane)
* możliwość przesłania artykułu mailen (patrz udostępnianie)

Badge „New” oraz prezentacja źródła w formie skróconej (a nie w pełnym adresie używanym w modelu) jest możliwe dzięki następującemu fragmentowi kodu w pliku views.py (definicja widoku home)

news\_with\_sources = []

    for news in news\_list:

        source = zrodla[news.source]

        news.is\_new = (now - news.data) < timezone.timedelta(days=1)

        news\_with\_sources.append({'news': news, 'source': source, 'new':news.is\_new})

Fragment kodu wykonuje operacje na liście wiadomości news\_list i tworzy nową listę news\_with\_sources zawierającą dodatkowe informacje o każdej wiadomości.

Dla każdej wiadomości następuje:

* Pobranie źródła wiadomości z zrodla na podstawie wartości pola source. Słownik zrodla został zaimportowany z pliku constants.py
* Obliczenie wartości pola is\_new dla wiadomości na podstawie różnicy między bieżącym czasem now a datą wiadomości data. Jeśli różnica jest mniejsza niż 1 dzień, to pole is\_new zostaje ustawione na True, w przeciwnym przypadku na False. Skorzystano tu z modułu timezone znajdującego się w django.utils

W pliku home.html zastosowano odpowiednią iterację po elementach listy news\_with\_sources aby wyświetlić wszystkie powyższe informacje.

W pliku base.html zdefiniowano nagłówek i stopkę każdej strony. Stopka zawiera informację o autorze i projekcie, nagłówek zawiera link do strony głównej oraz sekcję about z dodatkowymi informacjami o projekcie.

**Definicja widoku about**

def about(request):

    path = r'about.docx'

    file\_content = docx2txt.process(path)

    return render(request, 'about.html',{'file\_content': file\_content})

Widok about obsługuje żądanie widoku strony "about" i zwraca zawartość pliku DOCX do wyrenderowania w szablonie HTML.

Nastąpiło pobranie ścieżki do pliku DOCX i przypisanie jej do zmiennej path. Wywołanie funkcji docx2txt.process(path) w celu przetworzenia pliku DOCX. Funkcja ta zwraca tekstową reprezentację zawartości pliku. Następnie przypisanie przetworzonej zawartości pliku do zmiennej file\_content oraz zwrócenie wyniku renderowania szablonu HTML o nazwie 'about.html' za pomocą funkcji render(). Jako parametry przekazywane są: obiekt żądania request, nazwa szablonu i słownik kontekstu zawierający 'file\_content' jako klucz i wartość.

Dla przykładu działania modułu docx2txt, do sekcji about została wrzucona niniejsza dokumentacja.

Należy również zwrócić uwagę, że w projekcie zastosowano ikonę favicon dzięki czemu wygląda on bardziej profesjonalnie. Osiągnięto to dzięki dodaniu w pliku settings.py informacji o folderze static oraz odpowienie nazwanie pliku w tym folderze.

Ostatnią bardzo interesującą ciekawostką na stronie głównej możliwość podpatrzenia daty ostatniej aktualizacyjne newsów w polskiej lokalizacji, to znaczy widać datę 19 **czerwca** 2023 nie jak domyślnie Django przy wykorzystaniu modułu datetime 19 czerwiec 2023 roku. Zostało to osiągnięte dzięki następującemu fragmentowi kodu zawartego w pliku views.py w definicji widoku home

start\_date\_str = config.get('DEFAULT', 'start\_date')

    start\_date = datetime.strptime(start\_date\_str, '%Y-%m-%d').date()

    # Ustawienie lokalizacji na polską dla nazw miesięcy

    locale.setlocale(locale.LC\_TIME, 'pl\_PL.UTF-8')

    formatted\_date = babel.dates.format\_date(start\_date, 'd MMMM yyyy', locale='pl\_PL')

Kod wykonuje następujące kroki

1. Pobranie wartości daty początkowej z sekcji 'DEFAULT' pliku konfiguracyjnego. Klucz 'start\_date' wskazuje na odpowiednie pole w pliku konfiguracyjnym. Pobrana wartość jest przypisywana do zmiennej start\_date\_str.
2. Przekształcenie ciągu znaków start\_date\_str na obiekt datetime przy użyciu funkcji datetime.strptime(start\_date\_str, '%Y-%m-%d'). Format daty '%Y-%m-%d' odpowiada formatowi 'RRRR-MM-DD', w którym wartość daty jest zapisana w pliku konfiguracyjnym.
3. Wyodrębnienie samej daty z obiektu datetime poprzez wywołanie metody date(). Uzyskana data jest przypisywana do zmiennej start\_date.
4. Ustawienie lokalizacji na polską dla nazw miesięcy przy użyciu funkcji locale.setlocale(locale.LC\_TIME, 'pl\_PL.UTF-8'). Ta operacja pozwala na późniejsze sformatowanie daty w języku polskim.
5. Sformatowanie daty początkowej start\_date w określonym formacie przy użyciu funkcji babel.dates.format\_date(). Parametr 'd MMMM yyyy' określa format wyjściowy, gdzie 'd' oznacza dzień miesiąca, 'MMMM' oznacza nazwę miesiąca, a 'yyyy' oznacza rok. Parametr locale='pl\_PL' wskazuje na użycie polskiej lokalizacji. Sformatowana data jest przypisywana do zmiennej formatted\_date.

Formatted\_date zostało przekazane return funkcji oraz wyświetlone w pliku home.html

Początek formularza

Dół formularza

# Filtrowanie aktualności

Bardzo przydatną funkcjonalnością niniejszego projektu jest filtrowanie aktualności. Po kilku tygodniach użytkowania mam ich w bazie ponad tysiąc, więc odpowiednie filtrowanie może znacznie ułatwić zarówno ich przeglądanie jak i wyszukiwanie niezbędnych informacji.

Opcje filtrowania zostały ukryte w buttonie „Pokaż opcje filtrowania” na stronie głównej.

Obraz zawierający zrzut ekranu, tekst

Opis wygenerowany automatycznie

Dostępne opcje:

* Wyszukiwanie wartości w polu tytuł karty
* Wyszukiwanie newsów z daty po wskazanej
* Wyszukiwanie newsów z daty przed wskazaną
* Filtrowanie po źródle (co ważne nazwy filtrów pochodzą ze słownika w pliku constants.py)

Zostało to osiągnięte dzięki następującym krokom.

Do aplikacji w pliku settings.py dodano

'crispy\_forms',

    'crispy\_bootstrap4',

(wiersze 42 i 43)

W tym samym pliku dodano linię:

CRISPY\_TEMPLATE\_PACK = 'bootstrap4'

A w sekcji TEMPLATES dodano

'builtins': [

                'crispy\_forms.templatetags.crispy\_forms\_tags',

            ],

W pliku forms.py utworzono następujący formularz

class NewsFilterForm(forms.Form):

    tytul = forms.CharField(required=False)

    data\_od = forms.DateField(required=False)

    data\_do = forms.DateField(required=False)

    source = forms.MultipleChoiceField(choices=zrodla.items(), required=False, widget=forms.SelectMultiple(attrs={'class': 'form-control'}))

Formularz NewsFilterForm, jest używany do filtrowania danych w aplikacji. Formularz zawiera pola umożliwiające wprowadzenie kryteriów filtrowania dla tytułu, daty od, daty do oraz źródła.

Pola formularza:

1. tytul: Pole CharField umożliwiające wprowadzenie kryterium filtrowania dla tytułu. Jest opcjonalne (required=False), co oznacza, że nie jest wymagane w celu przesłania formularza.
2. data\_od: Pole DateField umożliwiające wprowadzenie kryterium filtrowania dla daty początkowej. Jest opcjonalne (required=False), co oznacza, że nie jest wymagane w celu przesłania formularza.
3. data\_do: Pole DateField umożliwiające wprowadzenie kryterium filtrowania dla daty końcowej. Jest opcjonalne (required=False), co oznacza, że nie jest wymagane w celu przesłania formularza.
4. source: Pole MultipleChoiceField umożliwiające wybór jednego lub więcej źródeł filtrowania. Opcje wyboru są dostarczane jako elementy słownika zrodla.items() zaimportowanego z pliku constants.py. Jest opcjonalne (required=False), co oznacza, że nie jest wymagane w celu przesłania formularza. Pole to jest renderowane jako pole SelectMultiple z atrybutami class: 'form-control', które wpływają na stylizację wyglądu.

Do obsługi formularza w pliku views.py w definicji funkcji home napisano następujący kod

form = NewsFilterForm(request.GET)

    news\_list = News.objects.all().order\_by('-data')

    if form.is\_valid():

        title = form.cleaned\_data['tytul']

        start\_date = form.cleaned\_data['data\_od']

        end\_date = form.cleaned\_data['data\_do']

        sources = form.cleaned\_data['source']

        if title:

            news\_list = news\_list.filter(tytul\_\_icontains=title)

        if start\_date:

            news\_list = news\_list.filter(data\_\_gte=start\_date)

        if end\_date:

            news\_list = news\_list.filter(data\_\_lte=end\_date)

        if sources:

            news\_list = news\_list.filter(source\_\_in=sources)

    helper = FormHelper()

    helper.form\_class = 'form-inline'

    helper.form\_method = 'GET'

    helper.form\_action = ''

    helper.layout = Layout(

        Row(

            Column('tytul', css\_class='form-group col-md-3 mb-0'),

            Column('data\_od', css\_class='form-group col-md-3 mb-0'),

            Column('data\_do', css\_class='form-group col-md-3 mb-0'),

            Column('source', css\_class='form-group col-md-3 mb-0'),

            css\_class='form-row align-items-center'

        ),

        Submit('submit', 'Filtruj', css\_class='btn btn-primary')

    )

Przetwarzanie formularza

1. form = NewsFilterForm(request.GET): Tworzy instancję formularza NewsFilterForm na podstawie danych przekazanych w parametrze request.GET. request.GET zawiera dane przesłane przez użytkownika z wykorzystaniem metody GET.
2. if form.is\_valid():: Sprawdza, czy dane przesłane przez użytkownika są poprawne. Jeśli formularz jest ważny, przechodzi do etapu filtrowania danych na podstawie wprowadzonych kryteriów.
3. title = form.cleaned\_data['tytul']: Odczytuje wartość pola tytul z formularza. cleaned\_data zawiera zvalidowane dane przesłane przez użytkownika.
4. start\_date = form.cleaned\_data['data\_od']: Odczytuje wartość pola data\_od z formularza.
5. end\_date = form.cleaned\_data['data\_do']: Odczytuje wartość pola data\_do z formularza.
6. sources = form.cleaned\_data['source']: Odczytuje wartość pola source z formularza.

Filtrowanie danych

Kod następnie filtruje listę news\_list na podstawie wprowadzonych kryteriów. Jeśli kryterium zostało wprowadzone przez użytkownika, odpowiednie filtry są zastosowane do listy.

* news\_list.filter(tytul\_\_icontains=title): Filtrowanie na podstawie tytułu, przy uwzględnieniu wielkości liter.
* news\_list.filter(data\_\_gte=start\_date): Filtrowanie na podstawie daty początkowej (data\_od) większej lub równej wartości start\_date.
* news\_list.filter(data\_\_lte=end\_date): Filtrowanie na podstawie daty końcowej (data\_do) mniejszej lub równej wartości end\_date.
* news\_list.filter(source\_\_in=sources): Filtrowanie na podstawie źródła (source), przy uwzględnieniu wartości wybranych przez użytkownika.

Lista jest przekazywana do pliku home.html który iteruje po liście wyświetlając odpowiednie newsy

Kod następnie definiuje helper (pomocnika) do renderowania formularza. Tworzona jest instancja FormHelper, która dostosowuje wygląd i zachowanie formularza.

Co ciekawe, filtry działają razem dodając kolejne ograniczenia do wyświetlanych kart.

Na końcu strony oraz pod wyborem filtra zastosowano buton Usuń filtr, który powraca do niefiltrowanej strony głównej

# Udostępnianie aktualności

Ostatnią funkcjonalnością strony jest przesłanie wybranego newsa mailem. Po naciśnięciu linku udostępnij, użytkownik jest przenoszony na stronę, gdzie musi podać email, na który ma być wysłana wiadomość

Obraz zawierający tekst, zrzut ekranu, Czcionka, Marka

Opis wygenerowany automatycznie

Po kliknięciu buttona Send zostaje przekierowany na stronę potwierdzającą wysłanie maila, z której może powrócić na stronę główną

Obraz zawierający tekst, zrzut ekranu, Czcionka, design

Opis wygenerowany automatycznie

Wysłana wiadomość wygląda następująco

Obraz zawierający tekst, zrzut ekranu, Czcionka, paragon

Opis wygenerowany automatycznie

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

Do osiągnięcia tego celu zastosowano następujące kroki

W pliku settings.py wprowadzono dane konta z którego ma być wysłana wiadomość email

EMAIL\_HOST = 'smtp.gmail.com'

EMAIL\_HOST\_USER = 'michal.na.wsb@gmail.com'

EMAIL\_HOST\_PASSWORD = 'jcbpatlxaftctuqt'

EMAIL\_PORT = 587

EMAIL\_USE\_TLS = True

Do prawidłowego działania należy w gmail ustawić weryfikację dwuetapową oraz hasło aplikacji.

Przygotowano stronę form.html z prostym formularzem do wprowadzenia adresu email

{% extends 'base.html' %}

{% block content %}

    <h1>Podaj adres email</h1>

    <form method="post">

        {% csrf\_token %}

        <br>email<br>

        <input type="email" name="email">

        <button type="submit">Send</button>

    </form>

{% endblock content%}

Oraz stronę sent.html zawierającą informację o wysłaniu maila i link do strony głównej.

W pliku urls.py zastosowano widok zawierający nr newsa oraz odniesienie do funkcji sendemail

path('<int:news\_id>/sendemail', nius.sendemail, name='sendemail'),

w pliku views.py zdefiniowano następujący widok

def sendemail(request, news\_id):

    news = get\_object\_or\_404(News, id = news\_id)

    if request.method == "POST":

        sub = news.tytul

        msg = f"{news.opis} \nPrzeczytaj więcej na stronie {news.url}"

        email = request.POST.get('email')

        send\_mail(

            sub, msg,'michal.na.wsb@gmail.com',

            [email]

        )

        return render(request, 'sent.html')

    return render(request, 'form.html')

Powyższy kod pobiera obiekt z bazy danych na podstawie jego id. Po kliknięciu na link udostępnij metodą żądania jest GET, dlatego użytkownik jest przekierowany do pliku form.html gdzie wypełnia pole email formularza. Następnie żądanie jest POST dlatego wykonywane są po kolei kroki instrukcji if. Jako temat wiadomości przyjmuje się tytuł pobranego newsa, treść maila jest kombinacją pola opis i url. Za pomocą funkcji send\_mail wysyłana jest wiadomość do wybranego użytkownika. W związku z tym, że pole formularza am typ email, użytkownik będzie mógł je wysłać do systemu jedynie po wprowadzeniu poprawnego adresu email.