



Politechnika Świętokrzyska

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI I INFORMATYKI

Mateusz Godlewski

Numer albumu: 85163

**Projekt i implementacja aplikacji internetowej
wspomagającej przeprowadzki z wykorzystaniem
narzędzi języka Python.**

**Praca inżynierska
na studiach I-go stopnia
na kierunku Informatyka**

Opiekun pracy dyplomowej:

dr inż. Andrzej Kułakowski

Katedra Zastosowań Informatyki

Kielce, 2020

POLITECHNIKA ŚWIĘTOKRZYSKA
WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI I INFORMATYKI

Zatwierdzam:

PROF. DR
ds. Kształcenia i Sprawy Studenckich
na Studiach Podyplomowych
Wydziału Elektrotechniki, Automatyki i Informatyki

Rok akademicki: 2019/20

Temat nr. 70/1876/2019/20

Dnia 05 07 2019 dr inż. Andrzej Stobiecki

ZADANIE NA PRACĘ DYPLOMOWĄ

Studiów pierwszego stopnia na kierunku INFORMATYKA

Wydano studentowi: **Godlewski Mateusz**

I. Temat pracy:

Projekt i implementacja aplikacji internetowej wspomagającej przeprowadzki z wykorzystaniem narzędzi języka Python.

II. Plan pracy:

1. Charakterystyka wybranych aplikacji internetowych wykonanych z wykorzystaniem języka Python.
2. Charakterystyka wybranych zagadnień wspomagania przeprowadzek.
3. Charakterystyka wybranych do realizacji zadania narzędzi i bibliotek programistycznych.
4. Opracowanie projektu aplikacji internetowej wspomagającej przeprowadzki.
5. Implementacja oprogramowania dla tworzonej aplikacji.
6. Testowanie działania wykonanej aplikacji internetowej.

III. Cel pracy:

Celem pracy jest stworzenie projektu i implementacji aplikacji internetowej wspomagającej przeprowadzki z wykorzystaniem narzędzi języka Python.

IV. Uwagi dotyczące pracy:

V. Termin oddania pracy: **koniec semestru zimowego - styczeń 2020**

VI. Konsultant:

Kierownik Zakładu

KIEROWNIK
Katedry Systemów Informatycznych
Wydziału Elektrotechniki, Automatyki i Informatyki

prof. dr hab. Aleksandra Jastrzebov

Opiekun pracy dyplomowej

(imię i nazwisko)

dr inż. Andrzej Kułakowski

Temat pracy dyplomowej celem jej wykonania otrzymałem(am):

Kielce, dnia 21.10.2019 r. Godlewski Mateusz
czytelny podpis studenta

Projekt i implementacja aplikacji internetowej wspomagającej przeprowadzki z wykorzystaniem narzędzi języka Python

Streszczenie

Celem mojej pracy dyplomowej było opracowanie projektu systemu wspomagającego przeprowadzki oraz jego implementacja w formie aplikacji internetowej. System umożliwia zarządzanie kontami użytkowników i firm, zarządzanie ogłoszeniami oraz wyszukiwarkę ogłoszeń bazującą na preferencjach użytkownika.

W niniejszej pracy zawarty jest przegląd podobnych serwisów internetowych lub aplikacji mobilnych oraz specyfikację wykorzystanych technologii oraz języków programowania.

Projekt został szczegółowo zilustrowany w postaci wykresów oraz diagramów, a implementacja została przedstawiona poprzez kody źródłowe oraz zrzuty ekranów poszczególnych funkcjonalności.

Słowa kluczowe: Aplikacja internetowa, Python, Django, Bootstrap, SQLite, Google Maps.

Project and implementation of relocation supporting web application using Python language utilities

Summary

Purpose of my thesis is development of relocation supporting system project and it's implementation in form of web application. System afford management user or relocation companies accounts, advert management as well search engine based on user preferences.

Present work contains review of similar web services and mobile app and specification of used technologies and programming languages.

Project was detailed illustrated in the form of charts and diagrams and implementation was presented by source codes and screenshots each functionalities.

Keywords: Web application, Python, Django, Bootstrap, SQLite, Google Maps.

SPIS TREŚCI

1. WSTĘP	11
2. CHARAKTERYSTYKA APLIKACJI INTERNETOWYCH STWORZONYCH ZA POMOCĄ NARZĘDZI JĘZYKA PYTHON	12
2.1 Nowoczesne aplikacje internetowe.....	12
2.2 Biblioteki i narzędzia języka Python	14
2.3 Aplikacje internetowe stworzone z wykorzystaniem bibliotek oraz narzędzi języka Python	16
3. CHARAKTERYSTYKA PROCESU PRZEPROWADZKI	18
3.1 Przeprowadzki	18
3.2 Nieruchomości	20
3.2.1 Rodzaje nieruchomości.....	20
3.2.1 Najem nieruchomości.....	21
3.3 Firmy przeprowadzkowe	22
3.4 Aplikacje internetowe wykorzystywane przy przeprowadzkach	23
4. PROJEKT APLIKACJI.....	26
4.1 Opis i założenia aplikacji.....	26
4.2 Opis funkcjonalności aplikacji	26
4.2.1 Typy użytkowników i przypadki użycia	27
4.2.2 Logowanie i autoryzacja	28
4.2.3 Zarządzanie ogłoszeniami w systemie	29
4.2.4 Zarządzanie firmami w systemie.....	31
4.2.5 Mechanizm wyszukiwania ogłoszeń.....	32
4.2.6 Administracja systemem	35
4.3 Architektura serwera aplikacji	36
5. IMPLEMENTACJA APLIKACJI	38
5.1 Wykorzystane narzędzia	38
5.1.1 Django	38
5.1.2 SQLite	39
5.1.3 REST	39
5.1.4 Google Maps API.....	40

5.1.5 Bootstrap	40
5.2 Struktura plików serwera aplikacji.....	41
5.3 Ustawienia serwera aplikacji.....	42
5.4 API serwera	44
6. TESTY APLIKACJI	XX
7. PODSUMOWANIE	XX

1. WSTĘP

Postęp technologiczny na przestrzeni ostatnich lat sprawił iż smartfony stały się dla ich posiadaczy narzędziem codziennego użytku. Wszechobecne aplikacje wykorzystywane są w niemal każdej dziedzinie życia. Służą nam w komunikacji, umożliwiają dostęp do informacji czy wiedzy, dostarczają rozrywki oraz pomagają w niektórych procesach życia.

Celem mojej pracy dyplomowej było opracowanie projektu systemu wspomagającego przeprowadzki oraz jego implementacja w formie aplikacji internetowej. System umożliwia zarządzanie kontami użytkowników i firm, zarządzanie ogłoszeniami oraz wyszukiwarkę ogłoszeń bazującą na preferencjach użytkownika. W niniejszej pracy zawarty jest przegląd podobnych serwisów internetowych i aplikacji mobilnych oraz specyfikację wykorzystanych technologii oraz języków programowania. Projekt został zilustrowany w postaci wykresów oraz diagramów, a implementacja została przedstawiona poprzez kody źródłowe oraz zrzuty ekranów poszczególnych funkcjonalności.

Dokument ten składa się z siedmiu rozdziałów przedstawiających najważniejsze aspekty pracy. Na wstępie przedstawiony został cel pracy oraz krótki opis poszczególnych rozdziałów. W rozdziale drugim pt. "Charakterystyka aplikacji internetowych stworzonych za pomocą narzędzi języka Python" przedstawiony został przegląd aplikacji internetowych przy których tworzeniu wykorzystane zostały biblioteki oraz dostępne frameworki języka Python z krótkim opisem tych technologii. Następny rozdział zatytułowany "Charakterystyka procesu przeprowadzki" zawiera opis zagadnień związanych z procesem przeprowadzek, nieruchomościami oraz firmami przeprowadzkowymi. W tej części znajduje się również przegląd dostępnych na rynku rozwiązań wykorzystywanych przy przeprowadzkach. Rozdział numer cztery "Projekt aplikacji" zawiera opis wymagań projektu, opis funkcjonalności systemu, diagramy przypadków użycia oraz opis wykorzystanej architektury serwera aplikacji. Następnie w rozdziale „Implementacja aplikacji” znajduje się przedstawienie wykorzystanych narzędzi oraz przedstawienie zaimplementowanych funkcjonalności systemu wraz z kodami źródłowymi oraz zrzutami ekranu stworzonego rozwiązania. Rozdział szósty "Testy aplikacji" poświęcony został opisowi testów stworzonego systemu. Ostatni, siódmy rozdział zawiera podsumowanie pracy.

2. CHARAKTERYSTYKA APLIKACJI INTERNETOWYCH STWORZONYCH ZA POMOCĄ NARZĘDZI JĘZYKA PYTHON

2.1 Nowoczesne aplikacje internetowe

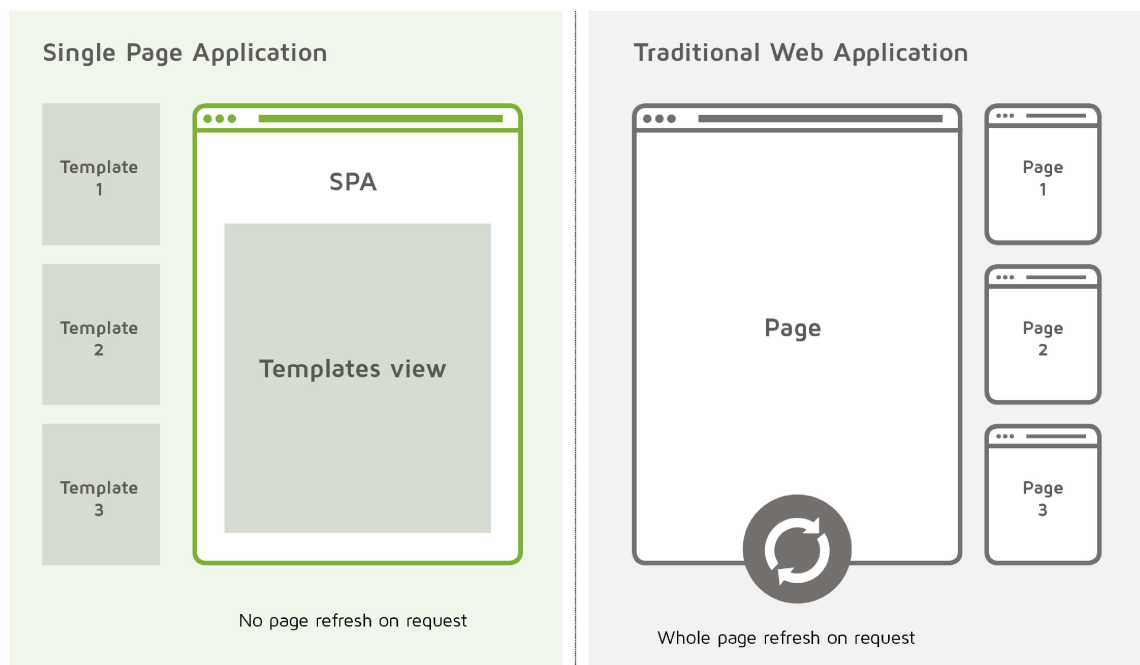
W dzisiejszych czasach, wraz z rozwojem technologii rosną w równie szybkim tempie oczekiwania wobec rozwiązań tworzonych z ich pomocą. Co za tym idzie twórcy aplikacji internetowych stoją przed nie lada wyzwaniem, albowiem produkty przez nich tworzone muszą spełniać bardzo wysoki poziom oczekiwania użytkowników i wymagania większe niż kiedykolwiek wcześniej. Oczekuje się że dzisiejsze aplikacje webowe będą dostępne całodobowo, bez przerwy, siedem dni w tygodniu, dostępne będą z każdego miejsca na świecie, oraz korzystać z nich można z większości urządzeń posiadających dostęp do internetu, niezależnie od oprogramowania urządzenia (wieloplatformowość) czy rozmiaru ekranu na którym aplikacja jest wyświetlana (responsywność). Obecnie gdy wzrasta wartość cyfrowych danych które stają się nową walutą w biznesie stawia się duży nacisk na bezpieczeństwo aplikacji sieciowych, a szczególnie tych które przetwarzają dane osobowe czy finansowe klienta lub użytkownika. Osoba korzystająca z usług musi mieć poczucie bezpieczeństwa oraz zaufanie do twórców aplikacji iż poufne dane będą zaszyfrowane i bezpiecznie przechowywane.

Tradycyjne podejście w tworzeniu aplikacji webowych zakłada niewielką ilość funkcjonalności po stronie klienta ale zamiast tego wszystkie funkcjonalności związane z nawigowaniem po stronach, zapytaniach do bazy czy całej logiki oddelegowane są na serwer aplikacji. Każda operacja wykonana przez użytkownika zostanie przetłumaczona na nowe żądanie sieci a wynikiem tej operacji jest nowo załadowana strona internetowa. Aplikacje tworzone według tego podejścia najczęściej tworzone w oparciu o architekturę MVC (ang. *Model View Controller*), gdzie każde nowe żądanie odpowiada operacji kontrolera, który przy współpracy z modelem zwraca widok przetwarzany do użytkownika [1]. Tradycyjne aplikacje wykorzystują na serwerach biblioteki takie jak:

- Spring (Java).
- Django (Python) .
- ASP.NET (C#).
- Ruby on Rails (Ruby).

Aktualnie nowoczesne aplikacje tworzone są w myśli aplikacji jednostronicowych SPA (ang. *Single Page Application*). Wykonują one większość logiki interfejsu użytkownika w przeglądarce internetowej, mogą wykonywać rozbudowane funkcje po stronie klienta które nie wymagają ponownego załadowania strony kiedy użytkownik podejmuje działania lub przemieszcza się pomiędzy obszarami aplikacji. Aplikacje SPA ładują się szybciej, pobierają oraz wykorzystują dane w tle. Interakcje użytkownika są bardziej płynne oraz responsywne, ponieważ funkcjonalności serwisu odciążone są od serwera aplikacji. Taki typ aplikacji umożliwia obsługę zaawansowanych funkcjonalności jak zapisywanie formularzy bez konieczności ich ponownego wczytywania czy przeciąganie i upuszczanie (ang. *Drag and drop*) które znacznie ułatwiają korzystanie z aplikacji [2]. Tworzenie nowoczesnych, jednostronicowych aplikacji jest możliwe z wykorzystaniem bibliotek języka JavaScript, takich jak:

- Angular
- React JS
- Node.js
- Vue.js

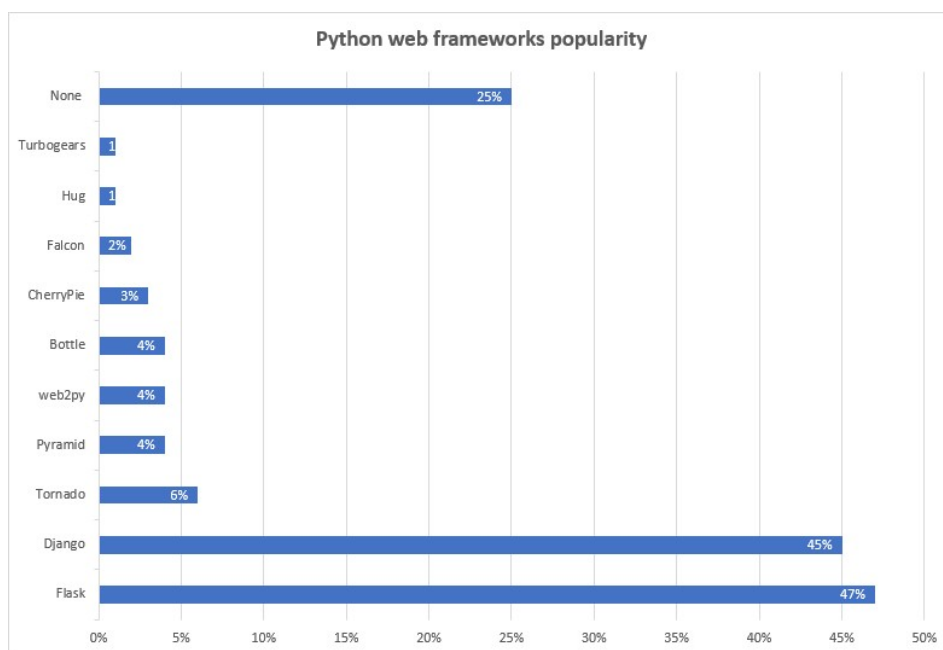


Rysunek 2.1 Porównanie SPA oraz tradycyjnych witryn internetowych

2.2 Biblioteki i narzędzia języka Python

Jednym z języków posiadających bogate biblioteki umożliwiające tworzenie nowoczesnych aplikacji internetowych jest język Python - język programowania wysokiego poziomu ogólnego przeznaczenia, którego ideą przewodnią jest czytelność i klarowność kodu źródłowego. Jego składnia cechuje się przejrzystością i zwięzłością. W tym języku zostało napisane wiele znanych i docenianych przez twórców aplikacji internetowych frameworków. Najczęściej wykorzystywane z nich przez developerów to:

- Django
- Flask
- Tornado
- Bottle



Rysunek 2.2 Popularność bibliotek webowych języka Python

Najpopularniejszymi z nich są bez wątpienia dwa pierwsze wyżej wymienione Django oraz Flask. Opinia programistów, specjalizujących się w tworzeniu aplikacji webowych właśnie w języku Python jest podzielona między oba z nich. Pomimo tego że każdy z nich wykorzystywany jest w tym samym celu, różnią się jednak od siebie pod kilkoma względami. Biblioteka Django udostępnia wiele gotowych rozwiązań takich jak panel administratora, zabezpieczenia, ORM czy wbudowany już system logowania, co dla niektórych programistów jest sporym usprawnieniem w procesie tworzenia oprogramowania. Porównywany Flask nie

posiada żadnych już wbudowanych udogodnień przez spowalnia pracę programistów i wymaga poświęcenia więcej czasu na stworzenie lub zaimplementowanie już gotowych bibliotek, lecz tworzy go „lżejszą” i bardziej konfigurowalną biblioteką w porównaniu do Django. Obie biblioteki oferują świetną wydajność działania tworzonej aplikacji, jednak Flask przoduje w wydajności mniejszych projektów, cechuje się on szybszym odczytem danych z bazy w porównaniu z Django, jednak konkurencyjne rozwiązanie jest wydajniejsze w kwestii większych, bardziej rozbudowanych aplikacji w których bardzo ważną rolę odgrywa skalowalność, dlatego też jest chętniej wybierane przez twórców dla których ważną cechą aplikacji jest jej wydajność sprastająca wzrastającej liczbie użytkowników. Każda z przedstawionych bibliotek doskonale sprawdza się w budowaniu aplikacji, ciężko jednoznacznie stwierdzić która z nich jest lepsza. To która z nich będzie sprawować się lepiej zależy od przeznaczenia tworzonego rozwiązania.

Tworzenie aplikacji przy użyciu języka Python oraz jego bibliotek niesie za sobą wiele korzyści takich jak:

- **Niski poziom wejścia** – osoby posiadające już doświadczenie programistyczne jak i zarówno osoby zaczynające pracę przy tworzeniu aplikacji nie będą miały problemów z nauką dzięki czytelnej i prostej składni języka.
- **Wizualizacja danych** – w prosty i szybki sposób można dokonać wizualizacji danych poprzez wygenerowanie rysunków i wykresów za pomocą takich bibliotek jak Matplotlib. Wizualizacja może pomóc przy rozwiązywaniu problemów natury programistycznej lub prezentowania użytkownikowi danych.
- **Bogate zasoby bibliotek** – biblioteki języka Python zawierają w sobie ogromną ilość gotowych, napisanych już funkcjonalności, dzięki czemu programista, korzystając z nich zaoszczędzi czas przy tworzeniu aplikacji.
- **Automatyzacja testów** – język Python jest jednym z najczęściej wybieranych języków przez testerów oprogramowania z racji na możliwości automatyzacji testów.
- **Uczenie maszynowe oraz sztuczna inteligencja** – to terminy których popularność w ostatnich czasach rośnie, wiele firm próbuje wykorzystać jak najwięcej możliwości dostarczanych przez uczenie maszynowe w swoich aplikacjach. Najbardziej zaawansowane w tej dziedzinie biblioteki powstały właśnie w języku Python, co znaczenie ułatwia ich wdrożenie w aplikacje napisane w tym języku.

2.3 Aplikacje internetowe stworzone z wykorzystaniem bibliotek oraz narzędzi języka Python

Z racji tego, iż aplikacje webowe cały czas ewoluują, co roku pojawiają się nowe narzędzia i frameworki różnych języków programowania umożliwiające tworzenie kompleksowych i zaawansowanych aplikacji internetowych spełniających wysokie oczekiwania użytkowników. Wiele dzisiejszych najpopularniejszych aplikacji oraz stron internetowych korzysta z pewnych funkcjonalności dostarczanych przez język Python i jego biblioteki, lub jest całkowicie napisana. Przykładem takich aplikacji są:

- **Instagram** – największy na świecie serwis do dzielenia się zdjęciami, który na swoich serwerach korzysta z frameworku Django. Inżynierowie Instagrama musieli stawić czoła wzrastającym w bardzo szybkim tempie liczbie użytkowników która w ostatnich latach wzrosła dwukrotnie i obecnie wynosi ponad 500 milionów klientów. W tej sytuacji priorytetem stało się postawienie nacisku na wydajność serwisu tak, aby usługa była dalej skalowalna i gotowa na kolejny przyrost użytkowników. Twórcy zdecydowali się, aby użyć na swoich serwerach Django, który pozwolił im na obsłużenie tej samej ilości użytkowników korzystając z tych samych zasobów sprzętowych [3].



Rysunek 2.3 Logo serwisu Instagram

- **Google** – jedna z najbardziej rozpoznawanych firm i zarazem stroną internetową na świecie. Google jest najczęściej używanym silnikiem wyszukiwania z udziałem rynkowym przekraczającym 75%. Inżynierowie Googla korzystają z języka Python do budowy wielu komponentów systemu Google (Google Aps Engine czy Youtube), lub do specjalistycznych narzędzi administracyjnych, analizy danych czy testowania tworzonego oprogramowania [4].

- **Spotify** – platforma umożliwiająca odsłuchiwanie konkretnych utworów lub albumów online bez opóźnienia związanego z buforowaniem. Aplikacja wystartowała w roku 2008, od tego czasu liczba użytkowników przekroczyła 75 milionów. Strona internetowa Spotify zbudowana jest na podstawie frameworka WordPress, natomiast reszta napisana jest w języku Python. Twórcy tej aplikacji przedstawili, iż Spotify składa się z wielu niezależnych serwisów, połączonych protokołami komunikacyjnymi, a 80% tych serwisów stworzona jest w języku Python. Zespół developerów Spotify intensywnie korzysta tego języka do analiz, zarówno w podejmowaniu decyzji jak i w samej aplikacji. Aby uprościć interakcje z serwisami przetwarzającymi zbiory danych wykorzystują oni bibliotekę Luigi. Biblioteka ta wykorzystywana jest z szeregiem algorytmów uczenia maszynowego w celu tworzenia radia oraz rekomendacji utworów, artystów czy albumów dla użytkowników Spotify [5].
- **Netflix** – wiodąca na świecie sieć telewizji internetowej z ponad 33 milionami użytkowników w 40 krajach którzy korzystają z ponad miliarda godzin programów telewizyjnych i filmów miesięcznie. Netflix w swoim blogu opowiada, iż coraz większa ilość programistów w firmie skłania się do używania właśnie języka Python. W dużej mierze język ten wykorzystywany jest w serwisach Netflix do przechowywania informacji za pomocą Python-memcached oraz Pycassa, zarządzania procesami dzięki Envoy odpytywania API dużych aplikacji przez funkcjonalności biblioteki requests, dostarczanie serwisów webowych z wykorzystaniem CherryPy oraz Bottle i przetwarzaniem danych z scipy [6].



Rysunek 2.4 Logo serwisu Netflix

3. CHARAKTERYSTYKA PROCESU PRZEPROWADZKI

3.1 Przeprowadzki

Przeprowadzka - zmiana miejsca zamieszkania na stałe lub na bliżej nieokreślony czas z jednego miejsca do drugiego. Relokacja miejsca zamieszkania może być uwarunkowana wieloma czynnikami, takimi jak edukacja, zdrowie czy zatrudnienie. Lecz warto też wspomnieć o powodach które w obecnych czasach wydają się być najczęściej spotykane [7]:

- **Chęć usamodzielnienia się.** Chęć bycia niezależnym od rodziców czy prawnych opiekunów jest jednym częstych powodów przeprowadzki u młodych osób. Mieszkanie samemu, z partnerem lub przyjaciółmi potrafi dostarczyć wiele doświadczeń i może być naprawdę przydatną życiową lekcją na której można nauczyć się odpowiedzialności czy oszczędności.
- **Chęć zmiany otoczenia.** Niektóre osoby bardzo szybko popadają w rutynę i nudę. Wiele zależy od prowadzonego stylu życia, preferencji czy potrzeb. Jednak niektórzy potrzebują powiewu świeżości i nowych wrażeń oraz zmian. Osoby takie najczęściej jako cel przeprowadzki obierają nowe miejsce, często daleko oddalone od poprzedniego miejsca pobytu. Zmiana miejsca zamieszkania do innego miasta czy regionu często odbywa się w celu poznawczym. Zmiana otoczenia również daje nowe możliwości zawodowe. Najczęściej na taki ruch mogą pozwolić sobie osoby które nie są stale związane z jednym miejscem pracy.
- **Zmiana perspektywy.** Niekiedy ktoś kto kiedyś zamieszkiwał większość swojego życia w jednym miejscu, dochodzi do wniosku po latach że potrzebuje czegoś innego. Taka sytuacja jest zależna od towarzyszących okoliczności i osobistych upodobań. Na przykład moment założenia rodziny, kiedy to zachodzi potrzeba zmiany miejsca mieszkania w celu lepszej perspektywy zawodowej zarówno dla siebie jak i dla przyszłego potomstwa. Innymi kwestiami może być lokalizacja pracy, przestrzeń wokół miejsca zamieszkania czy możliwości kulturowe.

Kupno mieszkania jest poważną decyzją podejmowaną na wiele lat. W ostatnich czasach wzrasta tendencja do kupowania mieszkań za gotówkę w celach inwestycyjnych, lecz nadal większość tych zakupów dokonywana jest na kredyt. Kredyt mieszkaniowy to zobowiązanie na kilka, kilkanaście czy czasem nawet kilkadziesiąt lat. Dlatego podejmując decyzję o zakupie mieszkania brane pod uwagę jest wiele czynników takich jak [8]:

- **Okolica i lokalizacja.** Lokalizacja mieszkania jest oczywiście najważniejszym czynnikiem który biorą pod uwagę osoby planujące zakup. Najpierw dokonywana jest selekcja miasta oraz okolicy, dopiero potem rozpatrywane są mieszkania według możliwości finansowych – takie wnioski płyną z badania "Finansowy Barometr ING. Jak Polacy kupują mieszkania i domy" opublikowanego w styczniu 2018 roku [9].
- **Cena.** Nie jest zaskakujące to że w większości przypadków cena ma decydujący wpływ na to czy wybrane mieszkanie zostanie zakupione. Cena była wskazana jako najważniejszy aspekt cytowanego w poprzednim punkcie badania ING [9] przez prawie 40 proc. respondentów. W przypadku posiadania ograniczonych możliwości finansowych kupujący musi iść na kompromis pomiędzy dogodniejszą lokalizacją a korzystniejszą ceną.
- **Powierzchnia i liczba pokoi.** Kolejnym istotnym czynnikiem z punktu widzenia nabywcy mieszkania jest kwestia wielkości nieruchomości oraz liczby pokoi. Na tą kwestię zwróciło uwagę około 25 proc. badanych w badaniu ING [9]. Niewątpliwie liczba pokoi czy powierzchnia jest kluczowa gdy nabywane lokum będzie zamieszkiwane przez wieloosobową rodzinę.

Wg jakich kryteriów zawęziłeś swój wybór mieszkania do kupna z dostępnej oferty?

Kryterium selekcji	Polska	Przeciętnie w Europie	Różnica
Okolica mieszkania	41%	23%	+18%
Zakres cenowy	35.5%	45%	-9,5%
Wielkość mieszkania	25%	28%	-3%
Liczba pokoi	23%	26%	-3%
Rodzaj budynku	18%	17%	+1%
Ogród lub balkon	17%	19%	-2%
Połączenia komunikacyjne	16%	17%	-1%

Źródło: Finansowy Barometr ING, Nieruchomości i kredyty hipoteczne, 2017

Rysunek 3.1 Kryteria wyboru mieszkania przez Polaków

3.2 Nieruchomości

Nieruchomości w świetle prawa określa się jako "część powierzchni ziemskiej stanowiące odrębny przedmiot własności (grunty), jak również budynki trwale z gruntem związane lub części takich budynków, jeżeli na mocy przepisów szczególnych stanowią odrębny od gruntu przedmiot własności". Definicja ta wskazuje iż nieruchomość jest nierozdzielnie związana z powierzchnią ziemską i prawem do własności gruntu lub/i budynku jak się na tym gruncie znajduje [10].

3.2.1 Rodzaje nieruchomości

Nieruchomość w świetle prawa można podzielić na trzy rodzaje [10]:

- **budynkowe** – według definicji nieruchomość budynkowa musi mieć budynki trwale związane z gruntem, które odnosząc się do przepisów szczególnych stanowią odrębny od gruntu przedmiot własności.
- **gruntowe** – to części powierzchni ziemskiej, które stanowią odrębny przedmiot własności. Do części składowych gruntu należą w szczególności budynki i inne urządzenia trwale z gruntem związane oraz drzewa i inne rośliny od chwili zasadzenia lub zasiania. Oznacza to, że właściciel gruntu jest właścicielem również wszystkich rzeczy ruchomych, które są trwale związane z gruntem.
- **lokalowe** – części budynków stanowiące odrębny przedmiot własności. Właściciel lokalu jest jednocześnie współwłaścicielem lub współużytkownikiem wieczystym gruntu, na którym posadowiony jest budynek. Do lokalu jako składowe części mogą przynależeć również inne pomieszczenia. Są to np. strych, piwnica, komórka, garaż.

Nieruchomości można podzielić również ze względu na ich przeznaczenie:

- nieruchomości mieszkalne
- nieruchomości rekreacyjne
- nieruchomości usługowe i handlowe (komercyjne)
- nieruchomości przemysłowe
- nieruchomości rolne
- nieruchomości specjalne
- nieruchomości leśne

3.2.2 Najem nieruchomości

Istnieje kilka prawnie sformułowane sposobów najmu mieszkania. Można całkowicie nabyć prawa do nieruchomości drogą zakupu, wynająć je na określony w umowie czas, nabyć je przez zasiedzenie, odziedziczyć w spadku czy w najrzadszym przypadku, zamienić się nimi. Do najczęściej zawieranych umów należą:

- **wynajem** - umowa, w której wynajmujący zobowiązuje się oddać lokal do używania najemcy, na czas oznaczony lub nieoznaczony, w zamian za wynagrodzenie w postaci czynszu płaconego przez najemcę. Wynajmujący ma obowiązek wydania najemcy lokalu w stanie przydatnym do umówionego użytku i utrzymywanie go w odpowiednim stanie przez cały czas trwania umowy na swój koszt. Najemca ma obowiązek płacenia czynszu, który może być ustalony w pieniądzach lub innych świadczeniach, a także używania rzeczy w sposób przewidziany w umowie i sprawowania nad nią pieczy. Stosunek najmu wygasa automatycznie z upływem okresu na jaki umowa została zawarta. Jeśli czas nie został oznaczony, umowa wygasa na skutek wypowiedzenia jednej ze stron [11].
- **sprzedaż** - umowa sprzedaży nieruchomości (w przeciwieństwie do ruchomości albo zwierząt) wymaga zachowania formy aktu notarialnego pod rygorem nieważności. Zgodnie z ustawą o nabywaniu nieruchomości przez cudzoziemców zakup nieruchomości w Polsce przez cudzoziemców może nastąpić wyłącznie za zezwoleniem ministra właściwego do spraw wewnętrznych. Nie dotyczy to jednak m.in. zakupu mieszkań. Przy zakupie nieruchomości obowiązują dwa rodzaje umów cywilnoprawnych: przedwstępna oraz sprzedaży-zakupu. Druga z nich musi zostać poświadczona aktem notarialnym, co zostało wspomniane wyżej. Od zakupu mieszkania obowiązuje zapłata podatku do Skarbu Państwa. Podatek ten wynosi 23% i jest pobierany od taksy notarialnej. Ponadto należy zapłacić również podatek od czynności cywilnoprawnych - 2% wartości mieszkania, oraz opłatę sądową za wpis do księgi wieczystej [12].

3.3 Firmy przeprowadzkowe

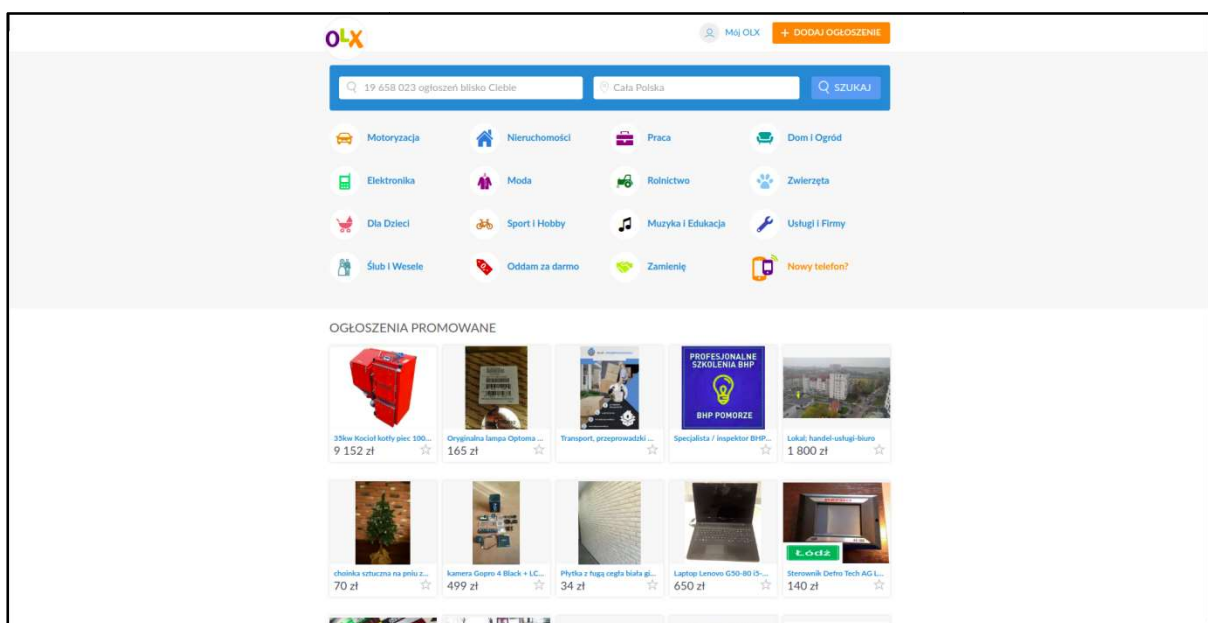
Przeprowadzka wiąże się często z transportem wielu dóbr materialnych, poczynając od walizek z odzieżą po elementy wyposażenia domu takie jak meble czy sprzęt RTV i AGD. W przypadku większej ilości przedmiotów, osoba przeprowadzająca się często korzysta z usług firm przeprowadzkowych.

Każda firma oferująca przeprowadzki jest również firmą transportową. Posiada ona sprawne pojazdy, ich ilość i typ jest używana zależnie od typu zlecenia przeprowadzki oraz potrzeb klienta. Samochody te posiadają usprawnienia które umożliwiają bezpieczny transport dóbr klienta. Bardzo ważnym elementem każdej firmy przeprowadzkowej jest kompetentna ekipa. Transportowane są często przedmioty o wysokiej wartości materialnej oraz sentymentalnej. Dlatego też ważne jest aby firma dysponowała solidnymi pracownikami którym można zaufać. Takich którym bez problemu można zawierzyć transport szkła, porcelany lustra, roślin, dzieł sztuki, antyków, dokumenty czy innych cennych dóbr. Każda firma przeprowadzkowa zajmuje się transportem przedmiotów między wskazanym miejscem a docelowym miejscem relokacji klienta. Firmy w swoich usługach posiadają również możliwość załadunku dóbr z miejsca wskazanego oraz ich zabezpieczenie za pomocą specjalnych narzędzi typu folia bąbelkowa, taśmy, worki, koce filcowe czy kartony. W przypadku mebli możliwy jest też ich demontaż w celu bezpieczniejszego i łatwiejszego transportu. Pracownikom można również powierzyć rozpakowywanie przywiezionych przedmiotów i ustawienie ich wedle uznania klienta. Wymagania klientów ciągle napędzają konkurencję wśród firm świadczących usługi przeprowadzki. Przedsiębiorstwo chcące zdobyć jak najwięcej klientów musi stawiać czoła rosnącym wymaganiom. Firmy starają się spełnić wszelkie oczekiwania klienta i zapewnić mu kompleksową przeprowadzkę. Co raz więcej zakładów oferuje przewóz zakupionych towarów ze sklepów budowlanych, meblowych, RTV i AGD [13].

3.4 Aplikacje internetowe wykorzystywane przy przeprowadzkach

Istnieje obecnie wiele serwisów internetowych oraz aplikacji mobilnych umożliwiających znalezienie mieszkania, domu lub pokoju do przeprowadzki czy zamówienia usługi transportowo przeprowadzkowej która zaoferuje swoje usługi do pomocy przy niej. Aplikacje te mogą służyć biurom nieruchomościowym czy osobom prywatnym na umieszczenie ogłoszenia w sieci, co ułatwia dotarcie do potencjalnego najemcy nieruchomości. Zarówno firmy transportowo przeprowadzkowe mogą reklamować w sieci swoje usługi. Przykładem takich serwisów są:

- **olx.pl** – serwis ogłoszeniowy o globalnym zasięgu. Działa on w ponad 40 krajach i jest obecnie największą firmą oferującą dostęp do platform ogłoszeniowych na rynku. Serwis działa w Polsce od 2014 roku jest on darmowy, pobierane są jedynie opłaty za ogłoszenia w niektórych kategoriach. Tworzenie ogłoszenia odbywa się za pomocą prostego formularza w którym umieszcza się opis wystawionego obiektu, jego cenę oraz zdjęcia. Użytkownicy mają dostęp do dyspozycji 15 głównych kategorii, jedną z nich jest "Nieruchomości". Użytkownicy aplikacji mają możliwość wyszukiwania ogłoszeń lub ich tworzenia w podkategoriach takich jak Mieszkania, Domy, Pokoje, Biura i Lokale jak i wiele innych związanych z nieruchomościami. Wyszukiwarka ogłoszeń posiada wiele pól które ułatwiają znalezienie nieruchomości według preferencji użytkownika [14].



Rysunek 3.2 Zrzut ekranu serwisu olx.pl

Każda podkategoria posiada odpowiednie dla siebie pola ogłoszenia, (np. Ogłoszenia mieszkań można wyszukiwać po rodzaju zabudowy, natomiast lokale i biura po ich przeznaczeniu).

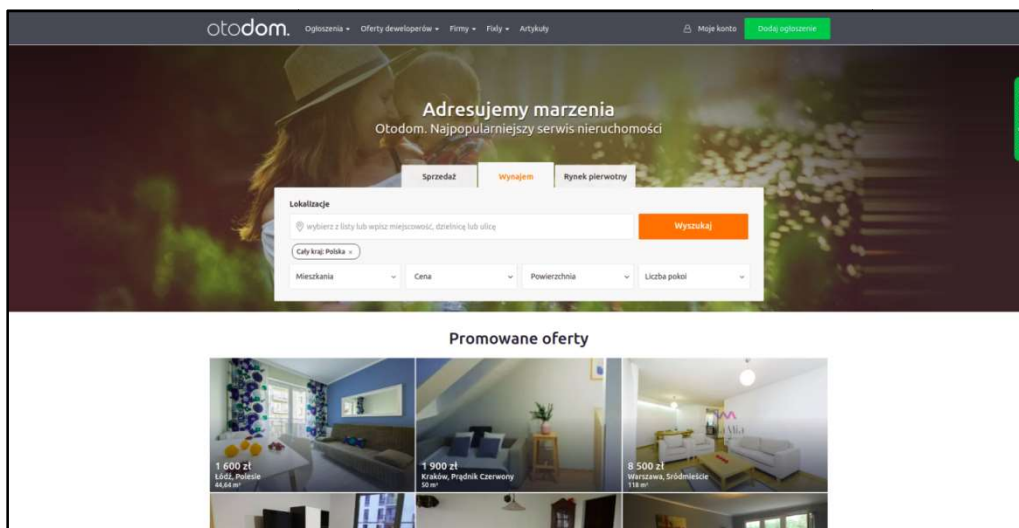
The screenshot shows the search interface for 'Biura i Lokale' (Offices and Localities) on the olx.pl website. At the top, there is a search bar with the placeholder 'Szukaj...', a location selector set to 'Cała Polska' (All Poland) with a distance of '+ 0 km', and a category dropdown menu currently showing 'Biura i Lokale'. Below these are two checkboxes: 'szukaj również w opisach' (search also in descriptions) and 'tylko ze zdjęciem' (only with photo). The main filter section includes a 'Wybierz kategorię' (Choose category) dropdown, a 'Przeznaczenie lokalu' (Office purpose) dropdown, and input fields for 'Cena od' (Price from), 'Cena do' (Price to), 'Pow. od' (Area from), and 'Pow. do' (Area to). There is also a 'Poziom' (Floor) dropdown. At the bottom right, there is a star icon for 'Obserwuj wyszukiwanie' (Watch search) and a blue 'SZUKAJ' (SEARCH) button.

Rysunek 3.3 Parametry wyszukiwania biur i lokali w serwisie olx.pl

The screenshot shows the search interface for 'Mieszkania' (Apartments) on the olx.pl website. The layout is similar to the previous one, but the category dropdown is set to 'Mieszkania'. The filter section includes a 'Wybierz kategorię' (Choose category) dropdown, a 'Rodzaj zabudowy' (Type of building) dropdown, and input fields for 'Cena od' (Price from), 'Cena do' (Price to), 'Pow. od' (Area from), and 'Pow. do' (Area to). Additionally, there are dropdowns for 'Poziom' (Floor), 'Liczba pokoi' (Number of rooms), and 'Umeblowane' (Furnished). At the bottom right, there is a star icon for 'Obserwuj wyszukiwanie' (Watch search) and a blue 'SZUKAJ' (SEARCH) button.

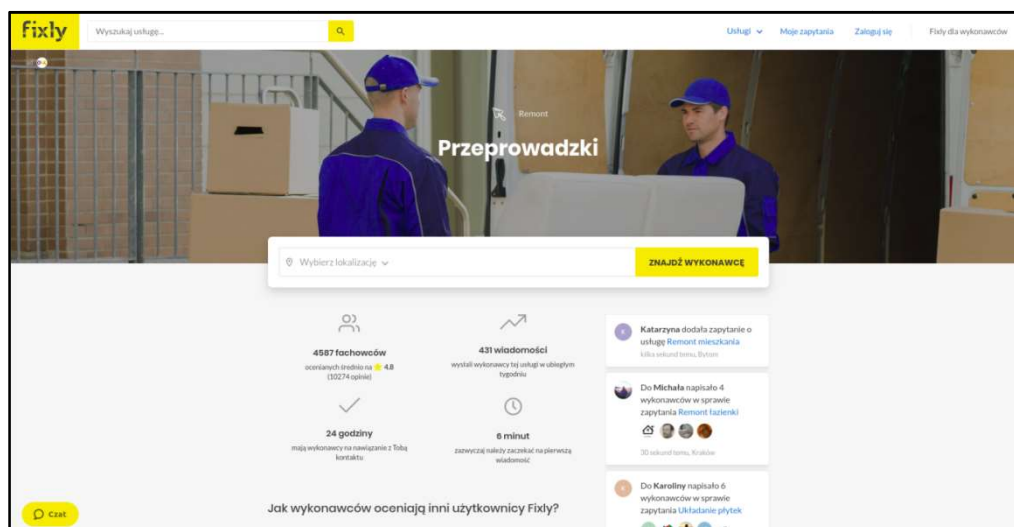
Rysunek 3.4 Parametry wyszukiwania mieszkań w serwisie olx.pl

- **otodom.pl** – serwis ogłoszeniowy dotyczący jedynie nieruchomości, działający w Polsce od 2006 roku. Umożliwia użytkownikom przeglądanie i umieszczanie ogłoszeń sprzedaży, wynajmu nieruchomości: mieszkań, domów, pokoi, działek etc. Dodawanie ogłoszeń do serwisu jest płatne, wysokość opłat jest uzależniona od tego czy klient wystawiający jest osobą prywatną, pośrednikiem nieruchomości czy deweloperem i od tego czy użytkownik chce dodatkowo promować ogłoszenie w serwisie. Funkcjonalność wyszukiwania jest podobna do serwisu olx.pl jednakże otodom.pl posiada mniej filtrów do wyszukiwania [15].



Rysunek 3.5 Zrzut ekranu serwisu otodom.pl

- **fixly.pl** – serwis ogłoszeniowy umożliwia odnalezienie lub zaoferowanie usług związanych z remontem, montażem, naprawą, budową, projektowaniem, transportem, przeprowadzką, usługami hydraulicznymi, elektrycznymi lub florystycznymi. Użytkownik chcący skorzystać z usług wybiera kategorie usług oraz swoją lokalizację, następnie odpowiada on na kilka pytań związanych z zamawianą usługą. Zapytanie stworzone przez użytkownika wysyłane jest do wszystkich firm i osób prywatnych świadczących dane usługi. Wykonawcy usług po zaakceptowaniu zapytania kontaktują się z klientem.



Rysunek 3.6 Zrzut ekranu serwisu fixly.pl

4. PROJEKT APLIKACJI

4.1 Założenia aplikacji

Stworzona aplikacja oferuje użytkownikowi pomoc w złożonym procesie przeprowadzki poprzez integrację tablicy ogłoszeń nieruchomości wraz z ogłoszeniami usług firm transportowo przeprowadzkowych. System dodatkowo posiada wyszukiwarkę ogłoszeń rozszerzoną o filtry lokalizacyjne umożliwiające odnalezienie lokum w dogodnym miejscu. Użytkownicy programu mogą zarządzać zarówno swoimi firmami jak i ogłoszeniami w systemie. Cały system posiada intuicyjny oraz przejrzysty interfejs umożliwiający wygodne użytkowanie programu.

Przed projektowaniem systemu zostały określone założenia które powinien on spełniać i na podstawie których można będzie przeprowadzić testy potwierdzające jego kompletność. Aplikacja ta powinna być:

- intuicyjna
- prosta w obsłudze
- funkcjonalna
- skalowalna
- bezpieczna

4.2 Opis funkcjonalności aplikacji

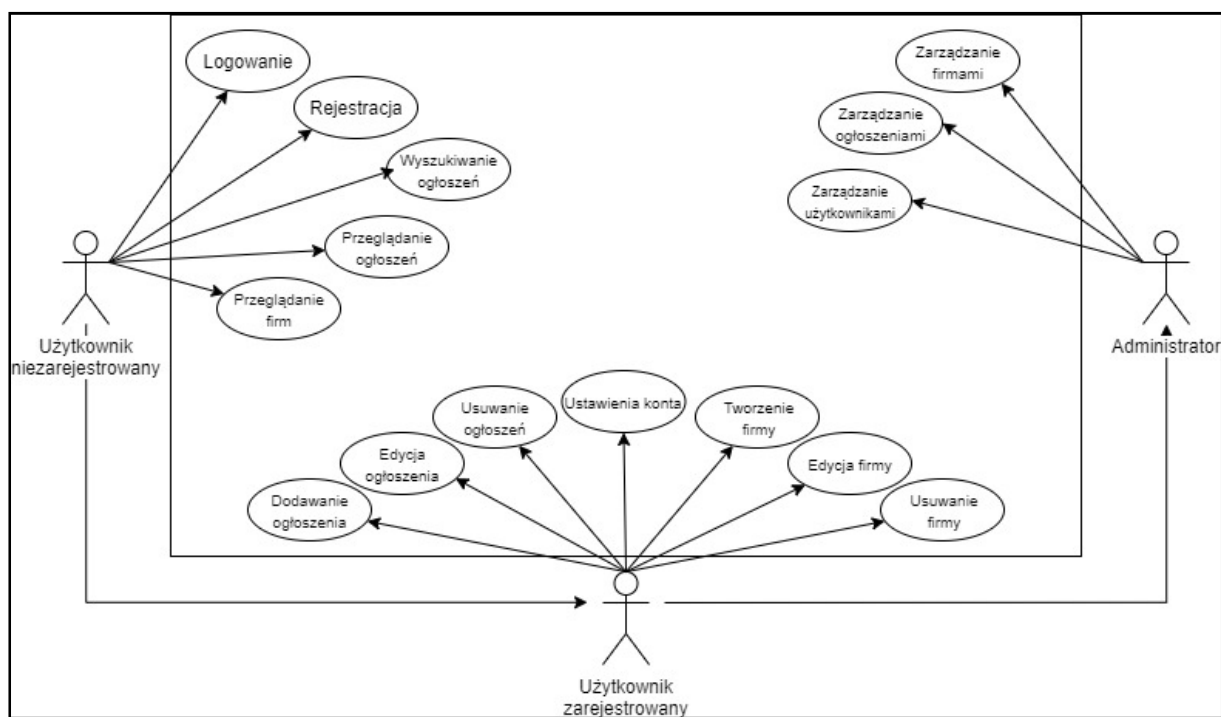
Tworzenie opisywanego systemu przebiegało zgodnie z uprzednio założonymi wymaganiami funkcjonalnymi które dotyczyły każdej ze sfer funkcjonalnościowej aplikacji. Punkty listy tych wymagań posłużyły jako punkty kontrolne podczas realizacji programu.

1. Określenie typu użytkowników i przypadków użycia.
2. Zaprojektowanie systemu logowania oraz autoryzacji w systemie.
3. Zarządzanie ogłoszeniami w systemie.
4. Zarządzanie firmami w systemie.
5. Zaprojektowanie mechanizmu wyszukiwania ogłoszeń.
6. Zapewnienie administratorowi możliwości zarządzania systemem.

4.2.1 Typy użytkowników systemu oraz przypadki użycia

W stworzonym systemie rozróżnia się trzy typy użytkowników korzystających z aplikacji:

- **Użytkownik niezarejestrowany (gość)** to użytkownik nie posiadający konta w systemie, lub użytkownik który nie zalogował się jeszcze na swoje konto. Ma on możliwość przeglądania firm oraz ogłoszeń w systemie.
- **Użytkownik zarejestrowany (klient)** to użytkownik zarejestrowany w systemie. Może tworzyć ogłoszenia i firmy oraz nimi zarządzać.
- **Administrator (superużytkownik)** administrator systemu. Posiada dostęp do zarządzania wszystkimi obiektami w aplikacji: użytkownikami, ogłoszeniami oraz firmami.



Obraz 4.1 Diagram przypadków użycia

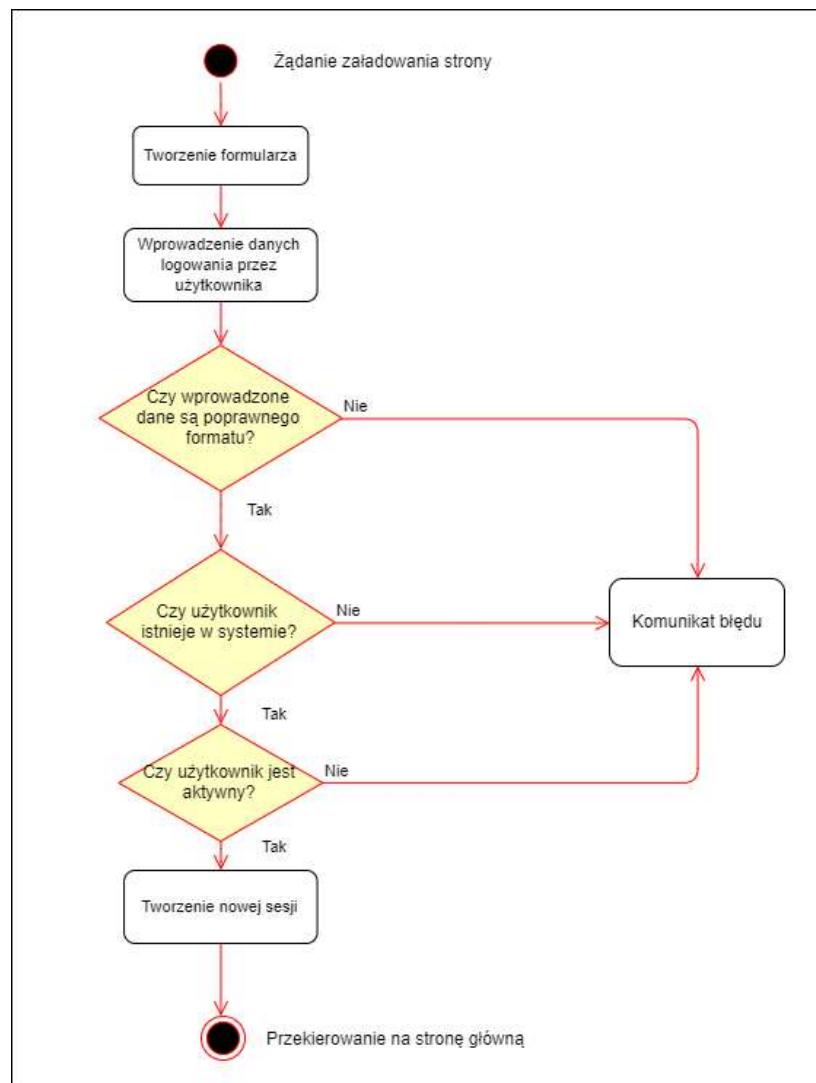
Dzięki takiemu rozkładowi ról użytkowników w systemie, jest on zabezpieczony przed ingerencją w dane z których korzysta aplikacja przez osoby nie będące użytkownikami systemu. Osoby niezalogowane nie mogą również dodawać obiektów w systemie co uniemożliwia generowanie dużej ilości nieistniejących ogłoszeń lub firm, tzw. spamu. Superużytkownik, czyli administrator w tym systemie będzie pełnił rolę zarządzającą, sprawdzając poprawność działania systemu oraz ingerencję w razie zaistniałych komplikacji.

4.2.2 Logowanie i autoryzacja

W projektowanym systemie ważnym aspektem jest system logowania, rejestracji oraz mechanizm sesji dla korzystających z niego użytkowników. Za tę funkcjonalność będzie odpowiedzialny domyślny, wbudowany moduł **django.auth**. Biblioteka ta zawiera możliwości konfiguracji wbudowanych już w sobie funkcjonalności.

Wbudowany moduł autoryzacji został odpowiednio skonfigurowany dla potrzeb tworzonego systemu i prezentuje się następująco:

- Logowanie do systemu jest możliwe po wprowadzeniu unikatowej nazwy użytkownika oraz hasła podanego podczas procesu rejestracji. Po udanym logowaniu w aplikacji tworzona jest sesja użytkownika, trwa ona aż do momentu wylogowania z systemu.



Rysunek 4.2 Proces logowania w systemie

- W nazwianiu do poprzedniego podrozdziału zatytułowanego „Typy użytkowników systemu oraz przypadki użycia” w systemie istnieją trzy grupy użytkowników, każda z tych grup posiada odpowiednie dla siebie prawa dostępu które determinują jakie funkcjonalności są dla nich dostępne.
- Hasła w systemie szyfrowane są za pomocą funkcji PBKDF2 która jest funkcją pochodnych klucza z przesuwającym kosztem obliczeniowym, wykorzystywanym w celu redukcji podatności na ataki typu brute force.
- Zarówno proces rejestracji jak i logowania zabezpieczony jest po stronie użytkownika za pomocą stosownych komunikatów pojawiających się podczas błędnego wprowadzania danych do formularzu.

4.2.3 Zarządzanie ogłoszeniami w systemie

Danymi obiektów w tworzonej aplikacji na których najczęściej wykonywane są operacje to dane ogłoszenia. Bardzo istotnym elementem był dobór parametrów opisujących nieruchomości które miały tworzyć wspólnie strukturę danych w bazie danych aplikacji. Po przeanalizowaniu funkcjonalności pod kątem wykonywanych na nich operacji powstał poniżej przedstawiony model ogłoszenia.

Advert
user: Integer
title: String
content: String
category: String
advert_type: String
create_date: Datetime
furnished: Boolean
city: String
address: String
price: Float
image: ImageField
map_url: String
map_coord_x: String
map_coord_y: String

Rysunek 4.3 Model ogłoszenia

Gdzie:

- **user** – właściciel ogłoszenia, każde pojedyncze ogłoszenie może posiadać jednego właściciela, lecz jeden użytkownik może posiadać wiele ogłoszeń.
- **title** – tytuł ogłoszenia.
- **content** - treść, opis ogłoszenia.
- **category** - kategoria ogłoszenia: dom, mieszkanie lub pokój.
- **advert_type** - typ umowy najmu nieruchomości, wynajem lub sprzedaż.
- **create_date** - data wystawienia ogłoszenia w systemie.
- **furnished** - informacja, czy nieruchomość jest umeblowana.
- **city** - miasto, w którym znajduje się nieruchomość.
- **adres** - ulica i numer domu zawartej w ogłoszeniu nieruchomości.
- **price** – cena najmu nieruchomości. W przypadku sprzedaży, konkretna kwota za całość, natomiast w przypadku wynajmu, kwota opłaty miesięcznej. Każde z ogłoszeń zawiera kwotę w walucie PLN.
- **image** - zdjęcie nieruchomości.
- **map_url** - adres lokalizacji nieruchomości w serwisie Google Maps.
- **map_coord_x** - szerokość geograficzna nieruchomości zawartej w ogłoszeniu.
- **map_coord_y** - długość geograficzna nieruchomości zawartej w ogłoszeniu.

Powyżej opisany model ogłoszenia zawiera najważniejsze informacje o nieruchomości w nim zawartej. Takie dane można zaprezentować potencjalnemu nabywcy lokum. Uwzględniając pola w modelu została zaprojektowana wyszukiwarka która zostanie przedstawiona bardziej szczegółowo w kolejnych rozdziałach Trzy ostatnie pola **map_url**, **map_coord_x** oraz **map_coord_y** umożliwiają wyszukiwanie nieruchomości bazując na jej lokalizacji geograficznej, oraz jej prezentację z wykorzystaniem interfejsu API oferowanym przez usługę Google Maps.

Aby zgodnie z diagramem przypadków użycia, zarejestrowany użytkownik systemu mógł zarządzać ogłoszeniami, w systemie zostały zaprojektowane cztery funkcjonalności umożliwiające wykonywanie operacji na danych:

- pobieranie informacji o istniejącym ogłoszeniu.
- tworzenie nowego ogłoszenia.
- edytowanie istniejącego ogłoszenia.
- usuwanie istniejącego ogłoszenia.

Każda z powyższych funkcji operuje na danych wprowadzonych do systemu przez jego użytkowników, dlatego każda z nich została zabezpieczona przed nieautoryzowanymi próbami ingerencji. Osoby niezalogowane w systemie nie mają prawa do modyfikacji danych, a klienci nie mają możliwości edycji oraz usuwania ogłoszeń których nie są właścicielami. Jeśli korzystająca z systemu osoba będzie próbowała naruszyć tę ochronę danych zostanie wyświetlony stosowny komunikat oraz nastąpi przekierowanie na stronę główną aplikacji.

4.2.4 Zarządzanie firmami w systemie

Zgodnie początkowymi założeniami w systemie oprócz tablicy ogłoszeń, obiektem biznesowym są również firmy przeprowadzkowo transportowe. Firmy oferujące swoje usługi w konkretnym obszarze wyświetlane są jako propozycje podczas przeglądania przez użytkowników ogłoszeń. Wspomniane firmy przeprowadzkowe posiadają również swoją osobną sekcję w systemie. Osoby korzystający z niego w celu własnie skorzystania z usług relokacyjnych mogą przejrzeć katalog firm będących w systemie.

Do przechowywania w aplikacji informacji o firmach została zaproponowana taka struktura która jest przystosowana do pozostałych modeli w systemie, użytkowników oraz ogłoszeń.

Company
user: Integer
name: String
location: String
logo: ImageField
tariff: ImageField

Rysunek 4.4 Model firmy przeprowadzkowej

Gdzie:

- **user** – właściciel firmy, który jest użytkownikiem systemu. Jedna firma może mieć jednego właściciela, a jeden użytkownik może być posiadaczem jednej firmy.
- **name** – nazwa firmy.
- **location** – miasto, w którym firma świadczy swoje usługi.

- **logo** – logo firmy.
- **tariff** – cennik firmy w formie obrazu.

Powyżej zaprojektowany model firmy przeprowadzkowej zawiera minimalne informacje które można zaprezentować potencjalnemu klientowi w systemie. Pole *location* zostało wykorzystane aby można było przyporządkować usługi danej firmy do nieruchomości znajdujących się w mieście w której przedsiębiorstwo świadczy swoje usługi. Aby uniknąć nadmiarowości danych, jako kontakt do konkretnej firmy został przyjęty kontakt do jej właściciela. Użytkownik będący posiadaczem firmy posiada możliwości jej zarządzania tak samo jak właściciel ogłoszenia może zarządzać swoim ogłoszeniem. Do wykonywania operacji na danych przedsiębiorstwa zostały zaprojektowane podobnie jak w przypadku ogłoszeń cztery funkcje: tworzącą, edytującą, usuwającą oraz pobierającą informację o danej firmie przeprowadzkowej. Podobnie jak w przypadku ogłoszeń zastosowane został również zabezpieczenia przed niepowołanymi żądaniami do danych. Jedynie właściciel firmy może modyfikować dane dotyczące przedsiębiorstwa którego jest właścicielem, pozostali użytkownicy systemu mogą jedynie wyświetlić o niej informację.

4.2.5 Mechanizm wyszukiwania ogłoszeń

Nawiązując do poprzednich podrozdziałów, jednym z najważniejszych obiektów w systemie są ogłoszenia. Przy dużej liczbie użytkowników aplikacji należy założyć iż liczba tworzonych ogłoszeń może być liczona w setkach a nawet tysiącach. Przeglądanie takiej ilości ogłoszeń w formie tablicy podzielonej na strony może być niewygodne, dlatego też jedną z głównych funkcjonalności systemu, oraz tą którą ten system wyróżnia spośród podobnych, istniejących na rynku rozwiązań jest mechanizm wyszukiwania ogłoszeń stworzonych w systemie.

Wyszukiwarka umożliwia zarówno użytkownikowi filtrowanie ogłoszeń na podstawie podstawowych parametrów nieruchomości takich jak przedziały cenowe, typy nieruchomości czy miasto, ale co najważniejsze i wyróżniające tworzoną aplikację od innych tego typu pozwala określić na jakich obiektach oraz w jakiej odległości od szukanego lokum zależy szukającemu. Taka funkcjonalność była możliwa dzięki oferowanym przez platformę Google Maps usługą Places API oraz Geocoding API.

Geocoding API to serwis umożliwiający uzyskanie współrzędnych geograficznych, szerokości i długości geograficznej na podstawie adresu w formie tekstowej oraz odwrotnie. Uzyskane w ten sposób wartości mogą zostać wykorzystane do prezentacji punktu lokalizacyjnego na mapie lub do innych obliczeń bazujących na współrzędnych [16].

Places API to usługa platformy Google Maps pozwalająca wyszukiwać informację na temat różnych miejsc czy obiektów zainteresowań na mapie, na podstawie ich typów, poziomu zainteresowania turystycznego czy ocen użytkowników. Możliwe jest również wyszukiwanie według odległości od podanej lokalizacji geograficznej lub adresu w formie tekstu [17].



Rysunek 4.5 Logo Geocoding API



Rysunek 4.6 Logo Places API

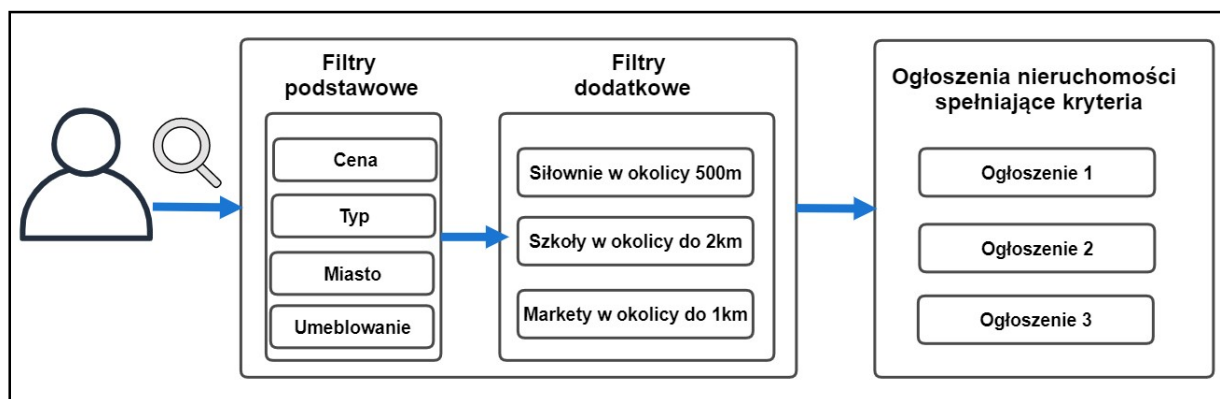
Wyszukiwanie ogłoszenia na podstawie jego odległości od podanych obiektów korzysta z obu powyżej przedstawionych serwisów. Pierwszy z nich na podstawie pól *city* oraz *address* modelu ogłoszenia (rysunek 4.3) uzyskuje za pomocą Geocoding API długość oraz szerokość geograficzną przechowywaną w polu *map_coord_x* oraz *map_coord_y* w obiekcie ogłoszenia. Tak przechowywane współrzędne nieruchomości posłużą do odnajdywania posiadłości oddalonych o konkretną odległość od konkretnych typów obiektów podanych jako parametry wyszukiwania przez użytkownika za pomocą Places API. Każda nieruchomość powinna zostać sprawdzona pod kątem podanych parametrów wykonując zapytanie do API o obiekty danego typu w zadanej odległości. Jeśli serwis zwróci jakieś wyniki, ogłoszenie powinno być brane pod uwagę, w przeciwnym wypadku lokum to nie pasuje to preferencji szukającego.

W tworzonej aplikacji istnieją dwa typy parametrów wyszukiwania:

- **Podstawowe** - to podstawowe parametry wyszukiwania, przy których nie jest wymagana żadna zaawansowana logika filtrowania ogłoszeń. Są nimi: miasto, kategoria nieruchomości, rodzaj najmu, umeblowanie czy zakres cenowy.
- **Dodatkowe** – to lista parametrów które są wykorzystywane do zapytania zewnętrznych serwisów w celu uzyskania danych służących do filtrowania ogłoszeń. Jest to lista typów obiektów i ich odległości od potencjalnego lokum.

Aby uniknąć sytuacji gdzie odpytywane będzie każde z ogłoszeń z systemu które mogłoby prowadzić do nadmiernej ilości zapytań serwisów Google Maps API a co za tym idzie wydłużało czas filtrowania ogłoszeń w aplikacji, wyszukiwanie zostało podzielone na dwa etapy:

- **Etap I** – filtrowanie ogłoszeń w systemie bazujące na parametrach podstawowych. Wykonwane jest ono za pomocą prostych zapytań SQL, a wynik tego etapu przekazywany jest do kolejnego, drugiego etapu wyszukiwania.
- **Etap II** – dane lokalizacyjne każdego z ogłoszeń ze zbioru będącego wynikiem pierwszego etapu wyszukiwania są wysyłane do serwisu Places API z zapytaniem o obiekty danego typu w konkretnej odległości zgodnie z preferencjami użytkownika. Jeśli ogłoszenie spełnia kryteria, w okolicy znajduje się obiekt tego typu jest ono wyświetlane użytkownikowi.

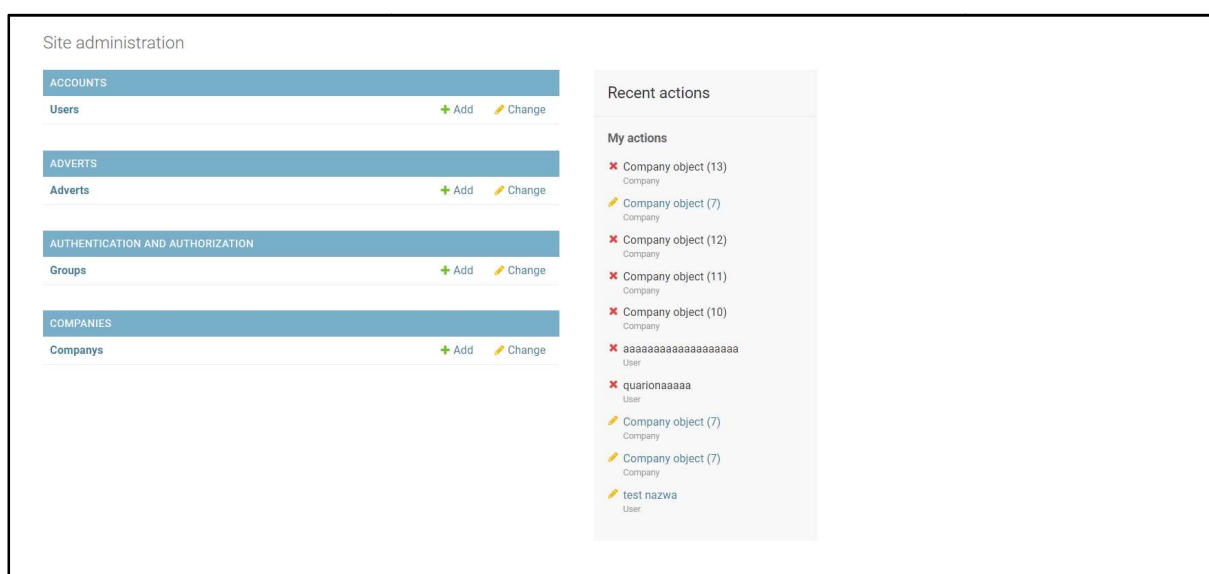


Rysunek 4.7 Proces wyszukiwania ogłoszenia w systemie

4.2.6 Administracja systemem

Wraz z rosnącą liczbą użytkowników aplikacji rośnie również ilość danych które trafiają do systemu. Aby w wydajny i szybki sposób zarządzać ciągle wzrastającą liczbą informacji aplikacja powinna posiadać panel administracyjny. Jest to graficzny interfejs umożliwiający administratorowi systemu zarządzanie obiektami na których system operuje.

Jako panel administratora w tworzonym systemie został wykorzystany gotowy już panel zaoferowany przez framework Django. Panel jest instalowany domyślnie wraz z utworzeniem aplikacji i łączy się on automatycznie z bazą danych gromadzącą informacje przetwarzane przez system.



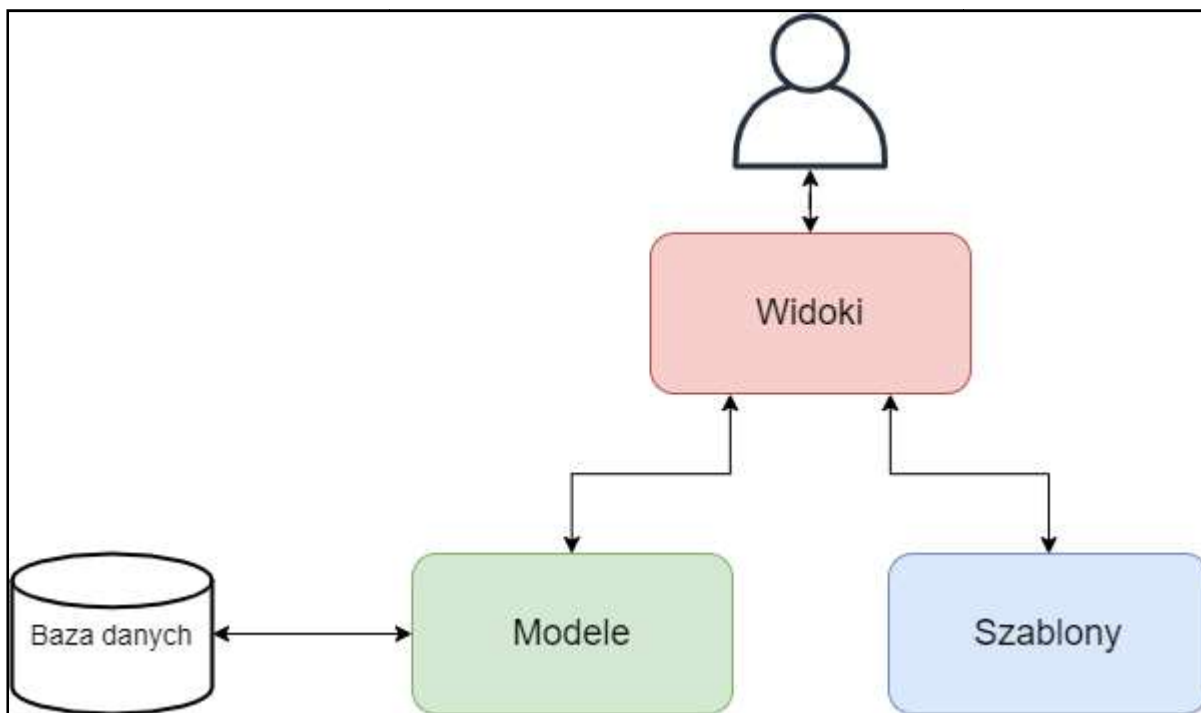
Rysunek 4.8 Panel administracyjny

Na powyższym rysunku przedstawiony jest domyślny panel administratora. Posiada on możliwość zarządzania użytkownikami, ogłoszeniami oraz firmami przeprowadzkowymi. Administrator bez bezpośredniej ingerencji w bazę danych aplikacji może zmienić wartość pola w dowolnym z obiektów systemu. Najważniejszą jednak kwestią jest możliwość administrowania kontami użytkowników z poziomu panelu użytkowników. Administrator aplikacji ma możliwość zmiany ról użytkowników, blokowania kont, przywracanie kont nieaktywnych lub wykonania innych czynności które dla zwykłego zarejestrowanego użytkownika są niedostępne bądź ograniczone. Panel ten jest konfigurowalny, poszczególne funkcjonalności mogą zostać usunięte z systemu, jeśli natomiast domyślny panel nie sprząta oczekiwaniom można poprzez modyfikację kodu źródłowego aplikacji dodać kolejne, nowe funkcjonalności poszerzające możliwości zarządzania systemem.

4.3 Architektura serwera aplikacji

Ważnym elementem projektowania aplikacji webowej stanowi wybór odpowiedniej dla niej architektury, szczególnie w przypadku aplikacji webowych wykorzystujących najnowsze technologie. Wybór ten bowiem ma istotny wpływ na łatwość w utrzymaniu aplikacji, jej skalowalność czy wydajność pod obciążeniem.

Aplikacja tworzony został w oparciu o architekturę MVT (ang. *Model View Template*, pol. *Model Widok Szablon*). Wybór tej architektury jest uwarunkowany jest używaną do tworzenia aplikacji biblioteką języka Python, Django. Architektura MVT rozdziela różne warstwy aplikacji co pozwala modyfikację wybranej warstwy, kiedy reszta z nich pozostaje niezmienna. Rozdział warstw zwiększa możliwość wydajnego rozwoju aplikacji poprzez wprowadzanie dodatkowych funkcjonalności oraz łatwość w utrzymaniu systemu.



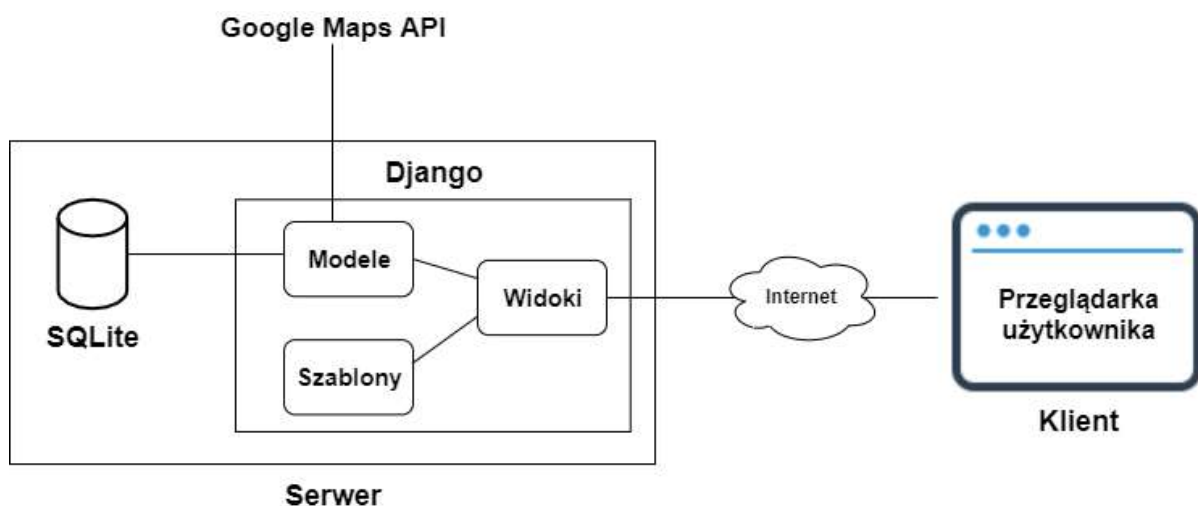
Rysunek 4.9 Schemat architektury MVT

Architektura MVT, jak można zauważyć na wyżej przedstawionym rysunku składa się z trzech głównych komponentów:

- **Modele** – jest to komponent architektury aplikacji odpowiedzialny za komunikację z bazą danych. W tym miejscu zdefiniowane są również modele obiektów wykorzystywanych w systemie, ich struktura oraz funkcjonalności ich dotyczące.

W przypadku tworzonego programu, w aplikacji wyróżnia nadmieniane już w poprzednich rozdziałach, trzy istniejące modele: model konta użytkownika, model ogłoszenia, model firmy przeprowadzkowej oraz dodatkowy model odpowiadający za integrację z Google Maps API. Tworzenie rekordów w bazie danych przechowywujące dane modelu aplikacji jest możliwe dzięki wbudowanej już w bibliotekę Django funkcjonalności ORM (ang. *Object-Relational Mapping*).

- **Widoki** – w tym komponencie zawarta jest cała logika biznesowa aplikacji. Widoki stanowią warstwę pośredniczącą między modelami a szablonami. Funkcje w nich zaimplementowane pobierają poprzez modele informacje o obiektach z bazy danych, wykonują na nich operacje, a następnie wynikowe dane wysyłają do szablonów. W tworzonej aplikacji każdy z modeli posiada co najmniej 4 widoki pozwalające na wykonywanie na nich operacji: tworzenia, pobierania, aktualizacji oraz usuwania.
- **Szablony** – warstwa która odpowiedzialna jest za prezentację danych w aplikacji. Dane wynikowe trafiają do komponentów szablonów gdzie wytwarzane są pliki języka HTML oraz DTL (ang. *Django Template Language*). Pliki te wysyłane są do przeglądarki użytkownika gdzie są prezentowane w formie strony internetowej. W tworzonym projekcie pliki różnią się od siebie w zależności od przeznaczenia, np. w szablonach odpowiadających za edycję poszczególnych obiektów systemu zawarte są napisane w języku HTML formularze.



Rysunek 4.10 Schemat blokowy aplikacji

5. IMPLEMENTACJA APLIKACJI

5.1 Wykorzystane narzędzia

5.1.1 Django

Django – darmowy framework typu open source, wykorzystywany do tworzenia aplikacji internetowych napisany całkowicie w języku Python. Pierwsza wersja została wydana w lipcu 2005 roku na licencji BSD (ang. *Berkeley Software Distribution*). Obecnie aktualna wersja frameworka to 3.0, natomiast tworzona aplikacja bazuje na wersji 2.2 tej biblioteki.

Głównym celem Django jest ułatwienie tworzenia złożonych stron internetowych opartych na bazie danych. Framework ten kładzie nacisk na możliwość ponownego użycia i „modułowość” komponentów, w celu zmniejszenia ilości kodu i szybkości rozwoju w myśl zasady DRY (ang. *Don't Repeat Yourself*, pol. *Nie powtarzaj się*). Django zapewnia również opcjonalny oraz konfigurowalny interfejs administracyjny służący do tworzenia, odczytu, modyfikacji oraz usuwania obiektów w aplikacji. Aplikacje internetowe tworzone z wykorzystaniem Django opierają się na architekturze MVT (ang. *Model View Template*), która została szczegółowiej opisana w rozdziale 4.3 *Architektura serwera aplikacji*. Biblioteka ta zapewnia wiele gotowych rozwiązań użytkownikowi takich jak gotowy panel administracyjny, system autoryzacji użytkowników, wbudowany ORM czy zabezpieczenia przed złośliwymi atakami przez co usprawnia pracę programisty i pozwala szybko tworzyć nowoczesne i rozbudowane aplikacje [18].

Oprócz wbudowanych funkcjonalności framework Django oferuje:

- Serializację oraz walidację formularzy.
- System szablonów wykorzystujący koncepcję dziedziczenia zapożyczoną z programowania obiektowego.
- Zaawansowany mechanizm cachowania.
- System serializacji który umożliwia odwzorowanie modelu w Django do reprezentacji w formacie JSON czy XML.
- Interfejs do wbudowanej platformy testów jednostkowych.
- Zaawansowany i elastyczny zestaw narzędzi do tworzenia webowych API.

5.1.2 SQLite

SQLite – system zarządzania bazą danych wraz z biblioteką implementującą go, obsługująca język SQL (ang. *Structured Query Language*). System ten jest dostępny na licencji public domain oraz został wydany w maju 2000 roku. Biblioteka ta implementuje silnik SQL, przez co umożliwia korzystanie z bazy danych bez konieczności uruchamiania osobnego procesu, przez co sprawia że SQLite idealnie nadaje się do wbudowanych systemów. Cała baza danych przechowywana jest w pojedynczym, binarnym pliku o rozszerzeniu .sqlite3 a pojedyncza tabela opiera się na osobnym B-drzewie[19]. W tworzonej aplikacji SQLite odpowiada za przechowywanie wszystkich danych na których operuje biblioteka Django.

Framework SQLite oferuje takie funkcjonalności jak:

- Zapytania zagnieżdżone
- Widoki
- Klucze obce
- Transakcje
- Wyzwalacze

5.1.3 REST

REST (ang. *Representational state transfer*, pol.*transfer stanu prezentacji*) to styl architektury oprogramowania opierający się o zbiór wcześniej określonych reguł opisujących jak zdefiniowane są zasoby aplikacji a także umożliwia do nich dostęp. REST umożliwia komunikację oraz wymianę danych między systemami oraz aplikacjami nie działającymi w obrębie jednej maszyny czy serwera. W tworzonym projekcie to właśnie REST umożliwia komunikację z usługami udostępnianymi przez Google Maps.

5.1.4 Google Maps API

Google Maps API - Interfejs programistyczny stworzony przez Google, umożliwiający uzyskiwanie informacji w postaci danych geograficznych, odległości czy zdjęć obiektów z których korzysta serwis Google Maps. Do korzystania z Google Maps API wymagany jest bezpłatny klucz który może uzyskać każdy użytkownik konta Google. Najnowsza wersja API umożliwia takie funkcjonalności jak:

- Możliwość geokodowania adresów
- Rysowanie polilinii
- Wyznaczenie trasy między lokazjami wraz z listą kroków
- Widok ulic

W tworzonej aplikacji zostały wykorzystane dwie w wielu usług Google Maps, Google Geocoding API oraz Places API, których zastosowanie w tworzonej aplikacji zostało szczegółowo opisane w rozdziale 4.2.5 *Mechanizm wyszukiwania ogłoszeń*. Komunikacja z owymi serwisami była możliwa poprzez wykorzystanie technologii REST [20].

5.1.5 Bootstrap

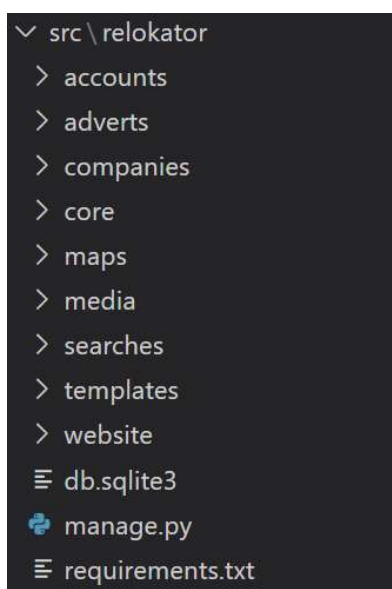
Bootstrap – biblioteka języka CSS, stworzona oraz rozwijana przez programistów serwisu Twitter, wydana na licencji MIT. Biblioteka ta zawiera zestaw przydatnych narzędzi umożliwiających tworzenie intuicyjnych, nowoczesnych oraz relatywnych stron internetowych. Głównie swoją funkcjonalność opiera na językach HTML oraz CSS, oraz opcjonalnie wykorzystuje do pewnych funkcjonalności język JavaScript. Bootstrap zawiera w sobie wiele gotowych szablonów tekstów, formularzy, przycisków, panelów nawigacyjnych oraz innych komponentów interfejsu [20].

W tworzonym projekcie biblioteka Bootstrap została wykorzystana do stworzenia całego interfejsu graficznego użytkownika. Do większości komponentów zostały wykorzystane gotowe klasy CSS, niektóre z nich zostały zmodyfikowane na potrzeby tworzonego rozwiązania. Wersja która została wykorzystana w projekcie to 4.3.1.

5.2 Struktura plików projektu

Z racji tego iż jako serwer aplikacji został wykorzystany framework Django narzucona została z góry ustalona struktura plików taka jak na rysunku 5.1. Większość z folderów w folderu głównego reprezentuje pojedynczą „aplikację” serwera Django. Każda z pojedynczych aplikacji w nazewnictwie biblioteki Django to moduł odpowiedzialny za konkretną funkcjonalność na serwerze. Aplikacje te są odczepialne, przez to raz napisaną aplikację można wykorzystać w innym projekcie Django.

Moduły **accounts**, **adverts** oraz **companies** zawierają w sobie definicję klas, metody operujące na obiektach oraz zbiór szablonów odpowiedzialnych kolejno za konta użytkowników, ogłoszenia oraz firm przeprowadzkowych. Aplikacja **core** to rdzeń serwera który zawiera jego pliki konfiguracyjne oraz jest odpowiedzialny za współpracę reszty modułów. Aplikacja **maps** służy do integracji serwera z Google Maps API. Folder **media** przechowuje pliki multimedialne przesyłane przez użytkowników systemu takie jak zdjęcia ogłoszenia, zdjęcia profilowe, loga firm przeprowadzkowych oraz ich cenniki. Moduł **searches** definiuje silnik wyszukiwania systemu, wykorzystuje on funkcjonalności modułu maps w celu dostarczenia wyszukiwarki ogłoszeń w systemie. Folder **templates** oraz **website** przechowują pliki HTML oraz CSS służące do tworzenia interfejsu użytkownika.



Rysunek 5.1 Struktura plików w projekcie

Aby uruchomić serwer należy zainstalować maszynie **Pythona** w wersji co najmniej **3.7** oraz system zarządzania pakietami Pythona 3 – **pip3**. Pierwszym krokiem jest instalacja pakietów poprzez wywołanie polecenia **pip3 install –r requirements.txt**, które zainstaluje wszystkie pakiety potrzebne do prawidłowego funkcjonowania serwera, których nazwy i wersje zostały właśnie umieszczone w pliku **requirements.txt**. Następnie w celu uruchomienia serwera wykorzystuje się polecenie **python3 manage.py runserver** z poziomu folderu na rysunku 5.1.

```
(venv) C:\Users\Mateusz\Documents\Praca Inżynierska\relokator\src\relokator>python manage.py runserver
Watching for file changes with StatReloader
INFO Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
January 21, 2020 - 17:52:16
Django version 2.2.3, using settings 'core.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Rysunek 5.2 Logi startowe serwera aplikacji

5.3. Ustawienia serwera

Ustawienia serwera są zdefiniowane w pliku **settings.py** który znajduje się w module będącym rdzeniem całego projektu – **core**. Plik uspokojnia współpracę między modułami, przechowuje globalne zmienne określające pracę serwera oraz połączenie z zewnętrznymi modułami serwera, takimi jak baza danych. Aby nowo dodana aplikacja mogła współpracować z serwerem musi ona zostać dodana do listy **INSTALLED_APPS** zdefiniowanych w pliku ustawień. Aplikacjami tymi mogą być zarówno te utworzone przez programistę jak zewnętrzne biblioteki oferowane przez framework Django.

```
INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "django_extensions",
    "accounts.apps.AccountsConfig",
    "website",
    "adverts",
    "searches",
    "maps",
    "companies",
]
```

Aby serwer mógł prawidłowo współpracować z bazą danych wymagane jest odpowiednie zdefiniowanie bibliotek umożliwiających tą współpracę w pliku ustawień oraz ścieżki do lokalizacji w której ma być przechowywany plik bazy danych.

```
DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.sqlite3",
        "NAME": os.path.join(BASE_DIR, "db.sqlite3"),
    }
}
```

W pliku settings.py zdefiniowane są również ścieżki dla plików w których korzysta serwer, takich jak statyczne pliki szablonów CSS czy lokalizację dla plików multimedialnych przesyłanych do serwera.

```
# login and logout redirection
LOGIN_REDIRECT_URL = "/"
LOGOUT_REDIRECT_URL = "/"

# static files location(CSS, JavaScript, Images)
STATIC_URL = "/static/"
```

W ustawieniach serwera istnieje również możliwość konfiguracji autoryzacji użytkowników w systemie oraz walidacji danych, jak na przykład hasło użytkownika. Na rysunku 5.7 został wskazany model **Accounts** jako model mający reprezentować użytkownika systemu oraz sposób autoryzacji. Lista **AUTH_PASSWORD_VALIDATORS** zawiera listę wbudowanych modułów Django odpowiedzialnych za walidację hasła wprowadzanego przez użytkownika podczas logowania do systemu.

```
# User Validation
AUTH_USER_MODEL = "accounts.Account"

# Password validation
AUTH_PASSWORD_VALIDATORS = [
    {"NAME": "django.contrib.auth.password_validation.UserAttributeSimilarityValidator"},
    {"NAME": "django.contrib.auth.password_validation.MinimumLengthValidator"},
    {"NAME": "django.contrib.auth.password_validation.CommonPasswordValidator"},
    {"NAME": "django.contrib.auth.password_validation.NumericPasswordValidator"},
]
```

5.4 API serwera

API (od ang. *Application programming interface*) to zestaw reguł w jaki sposób można uzyskać dostęp do zasobów serwera. W przypadku tworzonego systemu zasobami są pliki HTML. Aby otrzymać konkretny plik HTML (**szablon**) należy wywołać poprzez zapytanie do serwera **widok**. Wywołanie widoku jest możliwe poprzez wprowadzenie w adres przeglądarki konkretnej ścieżki URL (ang. *Uniform Resource Locator*), czyli ciągu znaków odpowiedzialnego za wywołanie odpowiedniego widoku na serwerze.

```
urlpatterns = [  
    path("", home),  
    path("admin/", admin.site.urls),  
    path("accounts/", include("django.contrib.auth.urls")),  
    path("accounts/", include("accounts.urls")),  
    path("adverts/new", advert_create_view),  
    path("adverts/", include("adverts.urls")),  
    path("companies/new", company_create_view),  
    path("companies/", include("companies.urls")),  
    path("search/", search_view),  
]
```

Na rysunku 5.8 została przedstawiona lista adresów URL pod którymi można uzyskać dostęp do poszczególnych zasobów systemu. Funkcja **path** przypisuje ciąg znaków który przyjmuje jako pierwszy argument do widoku np. do widoku tworzenia nowego ogłoszenia które można uzyskać po wejści pod adres *adverts/new*, jak i zarówno do kolejnych ścieżek zawartych w innych modułach projektu jak w przypadku adresu *accounts* gdzie za odpowiednie „oddelegowanie” do widoku będzie odpowiedzialny plik **urls.py** znajdujący się w module *accounts*. Taka struktura ścieżek porządkuje adresy odpowiednio do modułów których dotyczą i ułatwiają wykorzystanie gotowych już modułów poprzez dopisanie kolejnej linii w liście **urlpatterns** i wywołanie funkcji **path**, gdzie drugim argumentem będzie plik *urls.py* nowo dodanego do systemu modułu. Plik **urls.py** w module **core** zawiera ścieżki:

- "" – ścieżka do widoku strony głównej systemu.
- "admin" – ścieżka do panelu administracyjnego.
- "accounts/" – ścieżki odpowiedzialne za konta użytkowników w systemie.
- "adverts/" - ścieżki odpowiedzialne za ogłoszenia w systemie.
- "companies/" – ścieżki odpowiedzialne za firmy przeprowadzkowe.
- "search/" – ścieżka odpowiedzialna za wyszukiwanie ogłoszeń w systemie.

5.5 Nawigacja interfejsu

Aby umożliwić swobodne poruszanie się po systemie, na każdej ze stron został zaimplementowany pasek nawigacyjny. Panel ten umożliwia proste przechodzenie między widokami aplikacji bez konieczności ciągłego cofania się do strony głównej aplikacji.

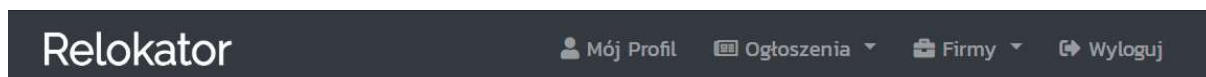
Zawartość nawigacji zmienia się zależnie od roli użytkownika w systemie, tak aby każdy z nich miał dostęp do odpowiednich funkcjonalności zgodnie z rysunkiem 4.1 *Diagram przypadków użycia*.

Użytkownik zalogowany na pasku nawigacji dostępne ma tylko przyciski umożliwiające rejestrację lub zalogowanie w systemie.



Rysunek 5.3 Pasek nawigacyjny użytkownika niezalogowanego

Użytkownik zalogowany w systemie, ma możliwość przemieszczania się za pomocą nawigacji między widokiem swojego profilu, ogłoszeniami i firmami przeprowadzkowymi. Ostatnim przyciskiem „Wyloguj” użytkownik zakończy swoją aktywną sesję w przeglądarce, wygłaszając się z aplikacji.



Rysunek 5.4 Pasek nawigacyjny użytkownika zalogowanego

Administrator systemu posiada te same funkcjonalności co zwykły, zarejestrowany użytkownik, lecz dodatkowo posiada on przycisk przekierowujący do panelu administracyjnego.



Rysunek 5.5 Pasek nawigacyjny administratora systemu

5.6 Autoryzacja użytkownika

W celach uwierzytelniania użytkownika w tworzonym systemie wykorzystany został wbudowany moduł frameworka Django, **django.auth**. Domyślny model zaoferowany przez ten moduł oferował model użytkownika który został rozszerzony o dodatkowe pola **phone** oraz **profile_image** na potrzeby tworzonego rozwiązania.

Obecnie model użytkownika w projekcie prezentowany jest przez klasę **Account** która dziedziczy po domyślnym modelu **AbstractUser** pochodzącym z wcześniej wspomnianego **django.auth**.

```
class Account(AbstractUser):

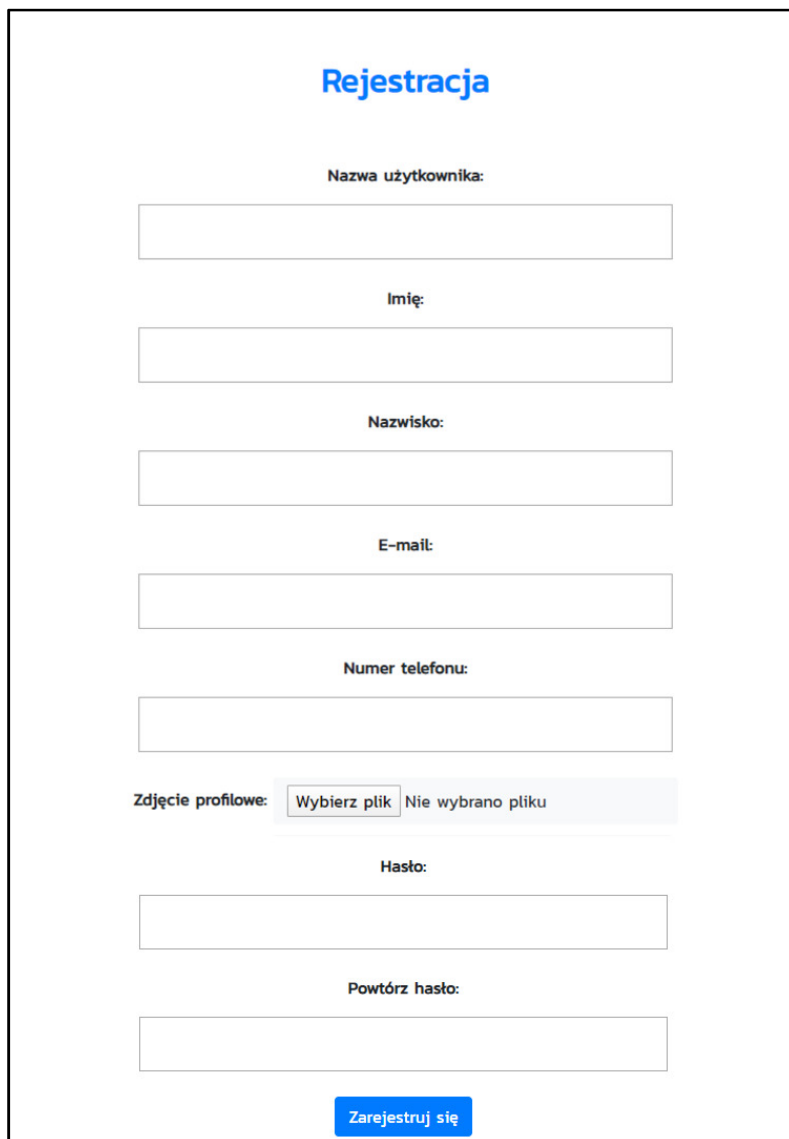
    username = models.CharField(
        max_length=50, unique=True, verbose_name="Nazwa użytkownika")
    first_name = models.CharField(
        max_length=100, unique=False, blank=False, verbose_name="Imię")
    last_name = models.CharField(
        max_length=100, unique=False, blank=True, verbose_name="Nazwisko")
    join_date = models.DateTimeField(default=timezone.now)
    email = models.EmailField(
        max_length=100, unique=True, blank=False, verbose_name="E-mail")
    phone_number = models.CharField(
        max_length=20, unique=True, blank=False, verbose_name="Numer telefonu")
    profile_image = models.ImageField(
        upload_to="profile_images/",
        blank=True,
        null=True,
        verbose_name="Zdjęcie profilowe",
    )

    USERNAME_FIELD = "username"

    REQUIRED_FIELDS = ["first_name", "email", "phone_number"]
```

Rejestracja

W celu założenia konta użytkownik niezalogowany musi przejść przez proces rejestracji. Rejestracja w systemie możliwa jest poprzez naciśnięcie przycisku „Zarejestruj się” w górnym pasku nawigacyjnym aplikacji. Następnie klient poprzez poprawne wypełnienie formularza rejestracyjnego tworzy w systemie swoje unikatowe konto.



Rejestracja

Nazwa użytkownika:

Imię:

Nazwisko:

E-mail:

Numer telefonu:

Zdjęcie profilowe: Wybierz plik Nie wybrano pliku

Hasło:

Powtórz hasło:

Zarejestruj się

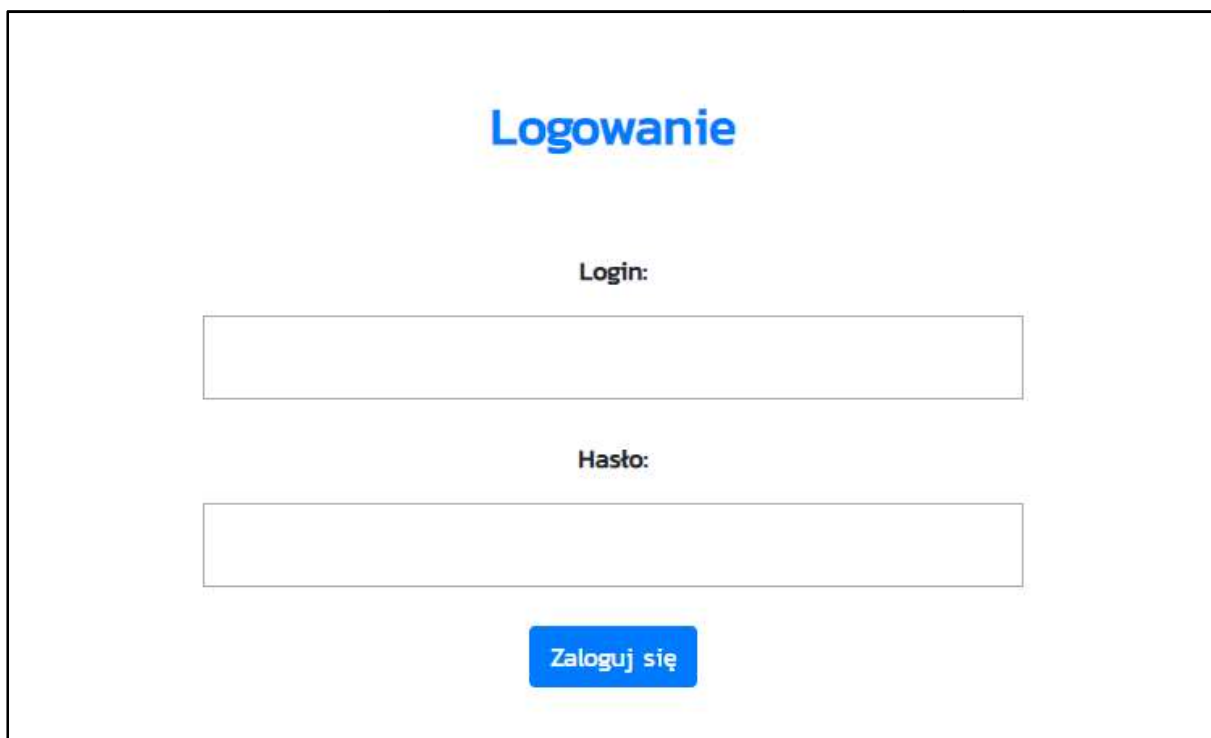
Rysunek 5.6 Formularz rejestracyjny

W przypadku gdy użytkownik poda wartość błędną lub nieodpowiedniego formatu (np. adres email nie będzie posiadał znaku „@”) do formularza zostanie on o tym powiadomiony poprzez wyskakujący komunikat. W przypadku podania zbyt słabego hasła lub zajętej nazwy użytkownika klient nie przejdzie dalej, a zostanie stosowanie poinformowany. Polami wymaganymi do wprowadzenia do formularza rejestracyjnego są

pola zawarte w liście **REQUIRED_FIELDS**, które odpowiadają kolejno imieniu, adresowi email oraz numerze telefonu w formularzu.

Logowanie

Logowanie do systemu odbywa się w panelu logowania dostępnym poprzez kliknięcie przycisku „Zaloguj się” na pasku nawigacji. Po przejściu do tego panelu użytkownik w celu zalogowania się musi podać hasło oraz nazwę użytkownika podawaną w procesie rejestracji. Logowanie jest możliwe poprzez wprowadzenie nazwy użytkownika, co zostało zaimplementowane ustawiając wartość stałej **USERNAME_FIELD** na nazwę pola klasy **username**.



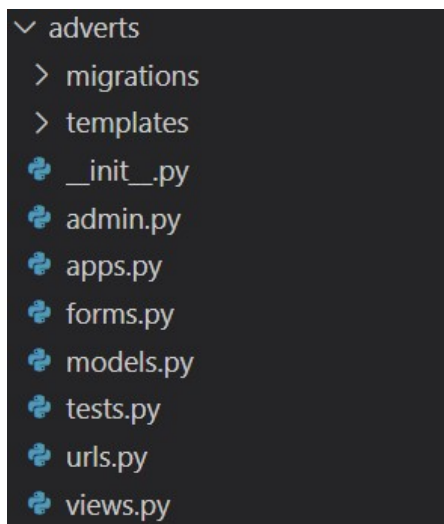
The image shows a login form with a light gray background. At the top, the word "Logowanie" is written in a large, blue, sans-serif font. Below it, the label "Login:" is centered in a smaller, dark gray font. Underneath is a white rectangular input field with a thin gray border. Below this field, the label "Hasło:" is centered in a smaller, dark gray font. Underneath is another white rectangular input field with a thin gray border. At the bottom center of the form is a blue rectangular button with the text "Zaloguj się" in white, sans-serif font.

Rysunek 5.7 Formularz logowania do systemu

Po poprawnym podaniu nazwy użytkownika oraz hasła, klient zostanie przekierowany na stronę główną aplikacji, natomiast w przypadku gdy użytkownik poda błędny login lub hasło, albo wprowadzi nazwę nieistniejącego użytkownika, zostanie poinformowany o błędzie poprzez wyskakujący komunikat.

5.7 Zarządzanie ogłoszeniami

Serwer aplikacji posiada trzy moduły odpowiadające za trzy główne modele na których wykonywane są operacje w tworzonym systemie, są to moduły: **accounts** (konta użytkownika), **adverts** (ogłoszenia) oraz **companies** (firmy przeprowadzkowe). W tym podrozdziale szczegółowo zostanie opisany moduł odpowiedzialny za ogłoszenia, albowiem ma on największy udział w funkcjonalności aplikacji.



Rysunek 5.8 Struktura plików w module adverts

W skład modułu **advert** wchodzi dwa foldery **migrations** oraz **templates**. Pierwszy z nich zawiera automatycznie generowane pliki Python zawierające historię migracji modułu. Natomiast drugi, templates przechowuje pliki szablony interfejsu użytkownika będące plikami języka HTML. Resztą plików są skrypty języka Python, każdy z nich odpowiedzialny jest za określoną funkcjonalność w module, najważniejsze pliki zostały opisane w kolejnych podrozdziałach.

5.7.1 Model

W pliku **models.py** przechowywana jest struktura modelu ogłoszenia w postaci klasy w języku Python. Klasa ta dziedziczy po klasie **Model** z modułu **django.db**, co umożliwia jej mapowanie dzięki wbudowanej funkcjonalności ORM frameworka Django na tabelę w bazie danych z której korzysta system. Dzięki zastosowaniu takiego dziedziczenia możliwe jest odwoływanie się do poszczególnych pól klasy jak do wartości w kolumnie o tej samej nazwie.

```

class Advert(models.Model):

    objects = AdvertManager()
    location = GoogleMaps()

    CATEGORY_CHOICES = (
        ("Dom", "Dom"),
        ("Mieszkanie", "Mieszkanie"),
        ("Pokój", "Pokój"),
    )

    TYPE_CHOICES = (
        ("Wynajem", "Wynajem"),
        ("Sprzedaż", "Sprzedaż")
    )

    user = models.ForeignKey(
        User, default=1, null=True, on_delete=models.CASCADE)
    title = models.CharField(max_length=120, verbose_name="Tytuł")
    content = models.TextField(null=True, blank=True, verbose_name="Opis")
    category = models.CharField(
        max_length=10, choices=CATEGORY_CHOICES, verbose_name="Kategoria")
    advert_type = models.CharField(
        max_length=10, choices=TYPE_CHOICES, verbose_name="Typ")
    create_date = models.DateTimeField(default=timezone.now)
    furnished = models.BooleanField(default=False, verbose_name="Umeblowanie")
    city = models.CharField(
        max_length=100, blank=False, verbose_name="Miasto")
    address = models.CharField(
        max_length=100, blank=False, verbose_name="Adres")
    price = models.PositiveIntegerField(
        blank=False, null=False, verbose_name="Cena")
    image = models.ImageField(
        upload_to="images/", blank=True, null=True, verbose_name="Zdjęcia")
    map_url = models.CharField(max_length=200, default="")
    map_coord_x = models.FloatField(default=0)
    map_coord_y = models.FloatField(default=0)

```

Zaprezentowana powyżej klasa **Advert** powstała na bazie zaprojektowanym w rozdziale 4.3.4 *Zarządzanie ogłoszeniami w systemie* modelu ogłoszenia. Każde z pól klasy jest zdefiniowane jako zmienna typu pochodzącego z modułu **django.models** co umożliwia jednocześnie określenie typu kolumny w tabeli przechowywującej dane obiektów tej klasy. Pola definiowane w tej klasie posiadają typy takie jak:

- **CharField** – zmienna przechowująca łańcuchy znaków, w konstruktorach określany jest maksymalna długość jaką może posiadać łańcuch.
- **TextField** – zmienna o podobnym jak w przypadku CharField zastosowaniu, lecz przechowywane ciągi mogą być większej długości.

- **DateTimeField** – zmienna przechowująca informację o dacie oraz czasie.
- **BooleanField** – pole przechowujące wartość logiczną, prawdę lub fałsz.
- **PositiveIntegerField** – pole przechowujące nieujemną liczbę całkowitą.
- **ImageField** – pole które umożliwia przechowywanie pojedynczego pliku obrazu.
- **FloatField** – zmienna przechowująca liczbę zmiennoprzecinkową.

Warto zwrócić uwagę na pole **user** które jest typu **ForeignKey**, oznacza to iż jest ono kluczem obcym w tabeli przechowującej obiekty ogłoszenia i odnosi się do klucza obcego w tabeli użytkowników co można zauważyć po przekazanej w pierwszym argumencie konstruktora klasie **User**. Domyślna wartość 1 w argumencie **default** oznacza że „osierocone” ogłoszenie zostanie automatycznie przypisane do użytkownika systemu o identyfikatorze 1, czyli administratora. Wartość **models.CASCADE** przekazana w parametrze **on_delete** sprawia iż usunięcie użytkownika z systemu spowoduje kaskadowe usunięcie należących do niego ogłoszeń. Do zmiennych odpowiadające za przechowywanie informacji o kategorii nieruchomości w ogłoszeniu (**category**) oraz rodzaju umowy najmu (**advert_type**), przekazane zostały do opcjonalnego argumentu **choices**, zbiory krotek określające jakie wartości mogą zostać przypisane do owej zmiennej. W przypadku kategorii nieruchomości mogą być to mieszkanie, dom lub pokój, natomiast w przypadku rodzajów umowy może być to wynajem lub sprzedaż. Zmienna **image** przechowuje w obiektach klasy **Advert** zdjęcie ogłoszenia. W celu przechowywania każdego ze zdjęć nowo dodanego obiektu w określonej lokacji na dysku, do argumentu **upload_to** został przekazany ciąg znaków wskazujący folder image jako docelowe miejsce składowania obrazów. Pola odpowiedzialne za składowanie informacji geograficznych nieruchomości zawartej w ogłoszeniu **map_url**, **map_coord_x** oraz **map_coord_y** zostały domyślnie zainicjalizowane z pustymi wartościami jako domyślnymi w konstruktorze. Taki zabieg został dokonany ponieważ wartość tych zmiennych dla poszczególnego obiektu klasy **Advert** będzie przypisywana przez funkcjonalności dostarczane z innych modułów. Każdy obiekt klasy **Advert** posiada również dwa pola **objects** oraz **location** które są obiektami innych klas tego modułu. Pierwsze z pól, **objects** jest obiektem klasy **AdvertManager** która odpowiedzialna jest m. in. za mechanizm wyszukiwania ogłoszenia w systemie, natomiast zmienna **location** jest obiektem klasy **GoogleMaps** która odpowiada za dane lokalizacji nieruchomości przy współpracy z serwisem **Google Maps API**. Klasy te opisane są w dalszej części rozdziału.

5.7.2 Widok

Aby możliwe było wykonywanie operacji na obiektach poprzednio opisywanej klasy **Advert**, w module muszą zostać zaimplementowane odpowiednie funkcjonalności w formie widoków. Widoki w danym module są funkcjami języka Python zamieszczonymi w pliku **views.py** które odpowiadają za wykonywanie większości logiki serwera. Jako argument przyjmują one żądanie HTTP, czyli zapytanie klienta o zasoby serwera, natomiast, zwracają one szablon wraz z zawartymi już w sobie wynikami obliczeń w formie odpowiedzi HTTP który jest potem przetwarzany przez przeglądarkę klienta i wyświetlany w formie strony internetowej. Wywołanie jednej z nich możliwe jest za pomocą API serwera przedstawianego w rozdziale 5.4 *API serwera*, poprzez wpisanie odpowiedniego adresu URL w adres przeglądarki przez klienta.

Dla klasy ogłoszeń został zaimplementowane 4 funkcje umożliwiające wykonanie operacji **tworzenia** nowego obiektu w systemie, **pobieranie** informacji o obiektach, **edycja** stworzonych już ogłoszeń czy **usuwanie** istniejących ogłoszeń. Każdy z tych widoków posiada odpowiedni dla siebie szablon w który umieszcza wyniki uzyskane podczas obliczeń.

Tworzenie ogłoszenia

Pierwszym z omawianych widoków to widok tworzący nowe ogłoszenie w systemie. Funkcja generująca formularz tworzenia ogłoszenia oraz umożliwiającą dodanie owego ogłoszenia do systemu ma nazwę **advert_create_view**.

```
@login_required
def advert_create_view(request):
    template_name = "adverts/adverts-create.html"
    form = AdvertModelForm(request.POST or None, request.FILES or None)

    if form.is_valid():
        obj = form.save(commit=False)
        obj.user = request.user
        form.save()
        form = AdvertModelForm()
    if request.method == "POST":
        return redirect(f"/accounts/{request.user}/adverts")

    context = {"form": form}
    return render(request, template_name, context)
```

Jeszcze nad deklaracją funkcji można zauważyć dekorator języka Python **@login_required**, który sprawdza czy użytkownik obecnie korzystający z systemu posiada aktywną sesję, czyli czy jest zalogowany. Dekorator ten pełni rolę zabezpieczenia przed tworzeniem ogłoszeń przez użytkowników którzy nie posiadają konta w aplikacji. Każda metoda początkowo jest wywoływana jest metodą GET, w celu otrzymania odpowiedniego szablonu. W każdym z widoków obecna jest zmienna **template_name** określająca lokalizację pliku HTML wraz z dołączonymi do niego danymi za zostać zwrócony w odpowiedzi funkcji. Do zmiennej **form** to obiekt klasy **AdvertModelForm** będący formularzem służącym tworzenia lub modyfikacji obiektów klasy **Advert**, klasa ta w sobie listę pól które mają zawierać się w formularzu oraz klasę dla której ma pełnić rolę formularza.

```
class AdvertModelForm(forms.ModelForm):  
  
    class Meta:  
        model = Advert  
        fields = [  
            "title",  
            "content",  
            "city", "address",  
            "category", "advert_type", "furnished",  
            "price",  
            "image",  
        ]  
  
        widgets = {'content': forms.Textarea(attrs={'rows':10, 'cols':70})}
```

Formularz jest inicjalizowany przez przekazanie do pierwszego argumentu żądania HTTP **request** w celu przekazania do funkcji takich informacji jak np. użytkownik który wysłał żądanie, a natomiast drugi argument przyjmuje pliki multimedialne, wysłane do tej funkcji metodą POST. W ten sposób zmienna **form** przechowuje pusty formularz który jest wyświetlany w szablonie użytkownikowi jako formularz to tworzenia ogłoszenia. Następnie do obiektu formularza do pola **user** przypisywany jest identyfikator obecnie zalogowanego użytkownika, co sprawia iż użytkownik tworzący ogłoszenie jest automatycznie ustawiany jako jego właściciel. Jeśli widok ten zostanie wywołany metodą POST, co odbywa się po naciśnięciu przycisku „Dodaj ogłoszenie” do systemu zostaje dodane owe ogłoszenie, a użytkownik zostaje przekierowany do widoku wszystkich swoich ogłoszeń. Funkcja **render**, jak wcześniej zostało wspomniane zwraca do przeglądarki klienta odpowiedź serwera w postaci szablonu HTML zawierającym w sobie, w tym przypadku pusty formularz przekazywany do funkcji **render** w zmiennej **context**.

Tworzenie nowego ogłoszenia

Tytuł:

Opis:

Kategoria: ----- ▼ Kategoria: ----- ▼ Umeblowanie: ☐

Miasto: Adres:

Cena: PLN

Zdjęcie:

Wybierz plik Nie wybrano pliku

Dodaj ogłoszenie

Rysunek 5.9 Formularz dodawania ogłoszenia

Edycja ogłoszenia

Aby właściciel ogłoszenia mógł **modyfikować** istniejące już w systemie ogłoszenia należało zaimplementować również funkcję która umożliwi mu swobodą zmianę wartości każdego z parametrów ogłoszenia. Funkcja ta nosi nazwę **adver_update_view**.

```
@login_required
def advert_update_view(request, advert_id:str):
    template_name = "adverts/adverts-update.html"
    advert = Advert.objects.get(id=advert_id)
    form = AdvertUpdateForm(request.POST or None, instance=advert)
    if form.is_valid():
        form.save()
    if request.user.username != str(advert.user):
        return render(request, "website/error_404.html")
```

```

if request.method == "POST":
    return redirect(f"/adverts/{advert_id}")

context = {
    "form": form,
    "title": "Edytujesz ogłoszenie '{}'.format(str(advert.title))"
}
return render(request, template_name, context)

```

Działanie tego widoku jest podobne do wcześniej omawianego widoku kreacji. Jedyną różnicą w implementacji jest to iż zmienna **form**, przechowuje obiekt klasy Advert przekazany do argumentu **instance** konstruktora. Obiekt ten natomiast został uzyskany na podstawie przekazanego do widoku argumentu **advert_id**, który umożliwił pobranie z bazy konkretnego obiektu. W tym widoku oprócz dekoratora **@login_required**, zostało zastosowane zabezpieczenie które w przypadku próby modyfikacji danych ogłoszenia przez użytkownika nie będącym jego właścicielem, przeniesie go na stronę błędu.

Edycja ogłoszenia

Tytuł:

Opis:

Kategoria: Kategoria: Umeblowanie: ☒

Miasto: Adres:

Cena: PLN

Zdjęcie:
 Currently: [images/photo-147585581690-80accde3ae2b_2EIBTTP_U5BEXmo.jpeg](#)
 Change: Nie wybrano pliku

Rysunek 5.10 Formularz edycji ogłoszenia

Wyświetlanie ogłoszeń

W celach prezentacji ogłoszenia dla osoby zainteresowanej najmem lokum zaimplementowany został widok pobierający informację o konkretnym ogłoszeniu oraz jego **wyświetlenie**. Za pobieranie informacji o ogłoszeniu odpowiedzialna jest funkcja **advert_detail_view**.

```
def advert_detail_view(request, advert_id:str):

    advert = get_object_or_404(Advert, id=advert_id)
    companies = Company.objects.filter(Q(location=advert.city))
    user = Account.objects.get(username=str(advert.user))

    context = {
        "advert": advert,
        "companies": companies,
        "phone_number": user.phone_number,
        "email": user.email,
    }
    return render(request, template_name, context)
```

Pierwszą kwestią którą można zauważyć to brak dektoratora **@login_required** nad funkcją odpowiedzialną za ten widok. Dektorator ten nie został zastosowany, ponieważ zgodnie z założeniami przedstawionymi w rozdziale *4.2.1 Typy użytkowników systemu oraz przypadki użycia*, przeglądanie ogłoszeń jest również dostępne dla niezarejestrowanych użytkowników systemu.

W celu pobrania z bazy danych odpowiedniego ogłoszenia wykorzystywany wartość argumentu **advert_id** który zostaje dalej przekazany do funkcji **get_object_or_404**, która zwraca ogłoszenie o podanym identyfikatorze, w przeciwnym wypadku, gdy takiego ogłoszenia nie ma, zwróci ona błąd. W kolejnej linii funkcji do zmiennej **companies** przypisywane są wszystkie firmy przeprowadzkowe które oferują swoje usługi w lokacji wyświetlanego ogłoszenia. Do ich pobrania wykorzystana jest wartość pola **city** danego ogłoszenia. Jako trzeci i ostatni obiekt który potrzebny jest do zaprezentowania ogłoszenia to użytkownik będący właścicielem ogłoszenia. Pobierany jest on na podstawie pola **user** ogłoszenia, a do prezencji ogłoszenia są wykorzystywane jego dane kontaktowe, adres email oraz numer telefonu.

Dom w Krakowie

800000 PLN

Nieruchomość: Dom

Rodzaj umowy: Sprzedaż

Umeblowanie: Tak

Opis

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

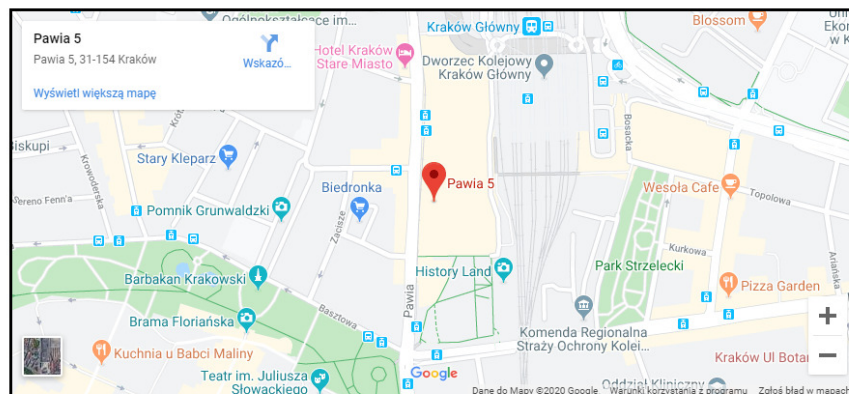
Zdjęcia



Lokalizacja

Miasto: Kraków

Adres: Pawia 5



Firmy przeprowadzkowe



Wystawiono: Oct. 27, 2019, 4:59 p.m.

Autor: admin

Edytuj

Usuń

Rysunek 5.11 Wyświetlenie szczegółów ogłoszenia

Usuwanie ogłoszenia

Jeśli ogłoszenie stanie się nieaktualne, użytkownik musi mieć możliwość usunięcia go z systemu. W tym celu został zaimplementowany widok usuwający ogłoszenie z systemu na życzenie jego właściciela. Funkcja odpowiedzialna za ten widok to **advert_delete_view**.

```
@login_required
def advert_delete_view(request, advert_id:str):

    template_name = "adverts/adverts-delete.html"

    advert = get_object_or_404(Advert, id=advert_id)

    if request.user.username != str(advert.user):
        return render(request, "website/error_404.html")

    if request.method == "POST":
        advert.delete()
        if advert.image:
            image_path = BASE_DIR + advert.image.url
            os.remove(image_path)

        return redirect(f"/accounts/{request.user}/adverts")

    context = {"object": advert}
    return render(request, template_name, context)
```

W celu zabezpieczenia przed niepożądanym usuwaniem ogłoszeń z systemu funkcja posiada wcześniej już opisywany dekorator **@login_required**. Podobnie jak w przypadku wyświetlania ogłoszenia czy jego podyfikacji, za pomocą wartości **advert_id** pobierany jest obiekt który ma zostać usunięty. Po wywołaniu tego widoku użytkownikowi ukazuje się poniższy komunikat.

Czy jesteś pewien że chcesz usunąć to ogłoszenie?

Tej akcji nie będzie można cofnąć!

Tak

Nie

Rysunek 5.12 Usuwanie ogłoszenia z systemu

Jeśli użytkownik zatwierdzi usunięcie poprzez naciśnięcie przycisku „Tak”, ogłoszenie to zostanie usunięte z systemu, a użytkownik zostanie przekierowany na widok wszystkich jego ogłoszeń. Natomiast w przypadku wcisnięcia przycisku nie, zostanie on powrotnie przekierowany na ogłoszenie które chciał usunąć.

5.8 Integracja z Google Maps API

W rozdziale 4.2.5 *Mechanizm wyszukiwania ogłoszeń* zaprezentowany został porojekt dwuetapowy mechanizm wyszukiwania ogłoszeń w systemie. Drugi etap tego mechanizmu bazuje na filtrowaniu obiektów ogłoszeń na podstawie danych geograficznych nieruchomości zawartej w ogłoszeniu. W celu pozyskania owych danych zaimplementowany został specjalny moduł **maps**.

W aplikacji **maps** serwera, całość funkcjonalności odpowiedzialnej za integrację wykonuje klasa **GoogleMaps**. Model ten odpowiada zarówno za pozyskiwanie współrzędnych geograficznych za pomocą usługi **Geocoding API**, znajdowanie obiektów określonego typu, w określonej odległości od szukanej przez użytkownika nieruchomości z wykorzystaniem **Places API** jak i generowanie linków oznaczających lokalizację nieruchomości na mapie.

Poniżej został przedstawiony kod klasy **GoogleMaps** znajdującej się w pliku **maps.py** w module **maps**.

```
class GoogleMaps(object):

    GOOGLE_MAPS_URL = "http://maps.google.com/maps"
    GOOGLE_GEOCODING_URL = "https://maps.googleapis.com/maps/api/geocode/json"
    GOOGLE_PLACE_URL = "https://maps.googleapis.com
        /maps/api/place/nearbysearch/json"

    def preprocess_address(self, city:str, address:str) -> str:
        return f'{address}+{city}'

    def get_location_map_url(self, city:str, address:str) -> str:
        return f'{self.GOOGLE_MAPS_URL}?
            q={self.preprocess_address(city, address)}&output=embed'

    def get_location_coords(self, city:str, address:str) -> int:
        try:

            response = requests.get(
                url = self.GOOGLE_GEOCODING_URL,
                params = {
                    'address': self.preprocess_address(city, address),
                    'key': 'AIzaSyC5rVKcoTfCep0GE7wnJc56P0ZfNbuLto8'
                }
            )
            geocode_data = response.json()
```

```

        x = geocode_data['results'][0]['geometry']['location']['lng']
        y = geocode_data['results'][0]['geometry']['location']['lat']

    except Exception as e:
        logging.error(f'GoogleMaps.get_location_coords() error -> {e}')
    else:
        return x, y

def filter_list_of_adverts(
    self,
    list_of_adverts:Iterable[object],
    object_type:str,
    radius:str
) -> Iterable[object]:

    try:

        filtered_list_of_adverts = []

        for advert in list_of_adverts:

            response = requests.get(
                url = self.GOOGLE_PLACE_URL,
                params = {
                    'location': f'{advert.map_coord_y},{advert.map_coord_x}',
                    'type': object_type,
                    'radius': radius,
                    'key': 'AIzaSyC5rVKcoTfCep0GE7wnJc56P0ZfNbuLto8'
                }
            )
            search_data = response.json()
            locations = [
                location["name"] for location in search_data["results"]
            ]

            if len(locations) > 0:
                filtered_list_of_adverts.append(advert)

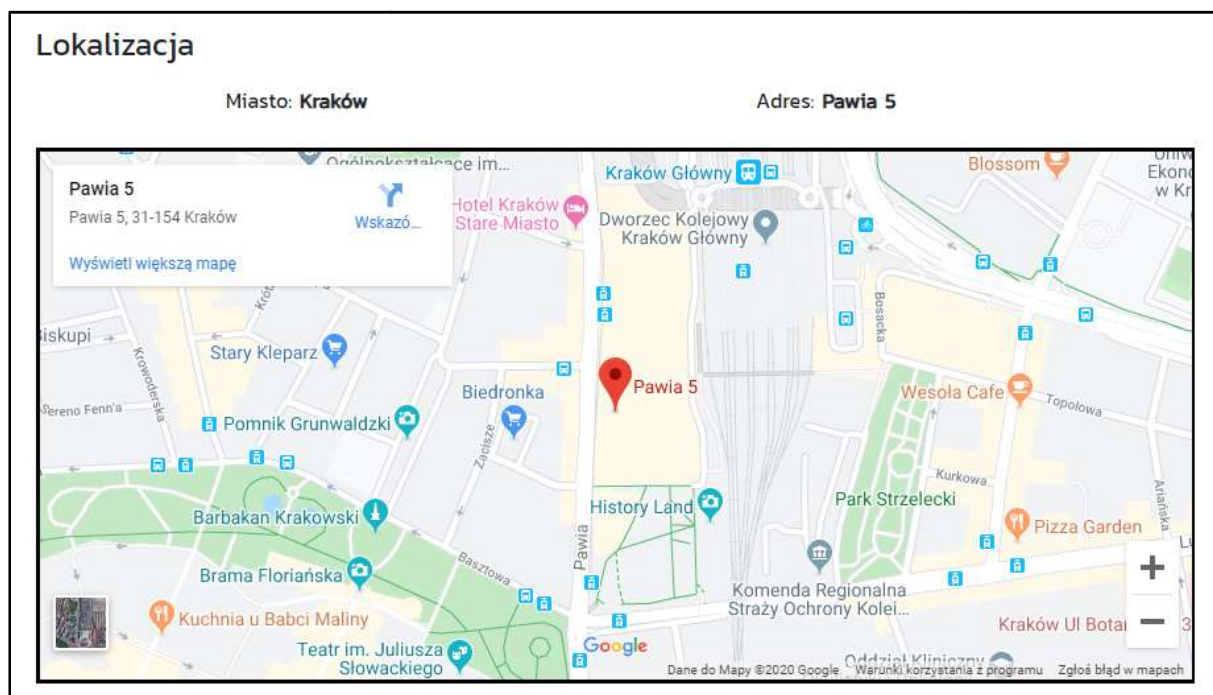
    except Exception as e:
        logging.error(f'GoogleMaps.filter_list_of_adverts() error -> {e}')
    else:
        return filtered_list_of_adverts

```

Klasa ta posiada trzy stałe przechowujące adresy URL poszczególnych serwisów Google Maps, są nimi: **GOOGLE_MAPS_URL**, **GOOGLE_GEOCODING_URL** oraz **GOOGLE_PLACE_URL**.

Metoda **preprocess_address** łączy dwa pojedyncze ciągi znaków będące reprezentacjami miasta oraz adresu nieruchomości w jeden długi ciąg znaków oddzielony znakami „+”, dzięki temu ciąg nadaje się do wykonania zapytania do serwisów Google Maps.

Funkcja **get_location_map_url** podobnie jak poprzednia funkcja, przyjmuje jako argumenty miasto oraz adres nieruchomości, jednak zwraca ona kompletny adres URL, który przekierowuje do serwisu map z oznaczonym na mapie w formie znacznika lokalację nieruchomości. Adres ten wykorzystywany jest w widoku przeglądania ogłoszenia w prezentacji lokalizacji nieruchomości na mapie w ogłoszeniu.



Rysunek 5.13 Mapa lokalizacji nieruchomości

Metoda **get_location_coords** odpowiada za pozyskanie szerokości oraz długości geograficznej na bazie adresu w formie tekstowej z wykorzystaniem **Google Geocode API**. Funkcja ta przyjmuje, jak poprzednio omawiane również adres i miasto w którym znajduje się nieruchomość. Ciąg znaków połączony za pomocą funkcji **preprocess_address** wysyłany jest dzięki bibliotece **request** metodą POST pod adres znajdujący się w stałej **GOOGLE_GEOCODING_URL**. W odpowiedzi na żądanie zostają zwrócone dane w postaci **JSON** (ang. JavaScript Object Notation). Dzięki metodzie **json()** dane te zostały przekonwertowane to słownika, z którego następnie została wydobyta wartość szerokości i długości geograficznej.

Obie poprzednio opisane metody `get_location_map_url` i `get_location_coords`, wywoływane są w momencie tworzenia nowego obiektu klasy `Advert`, a przeżyjniej w funkcji zapisującej ów obiekt do bazy danych, funkcji `save`.

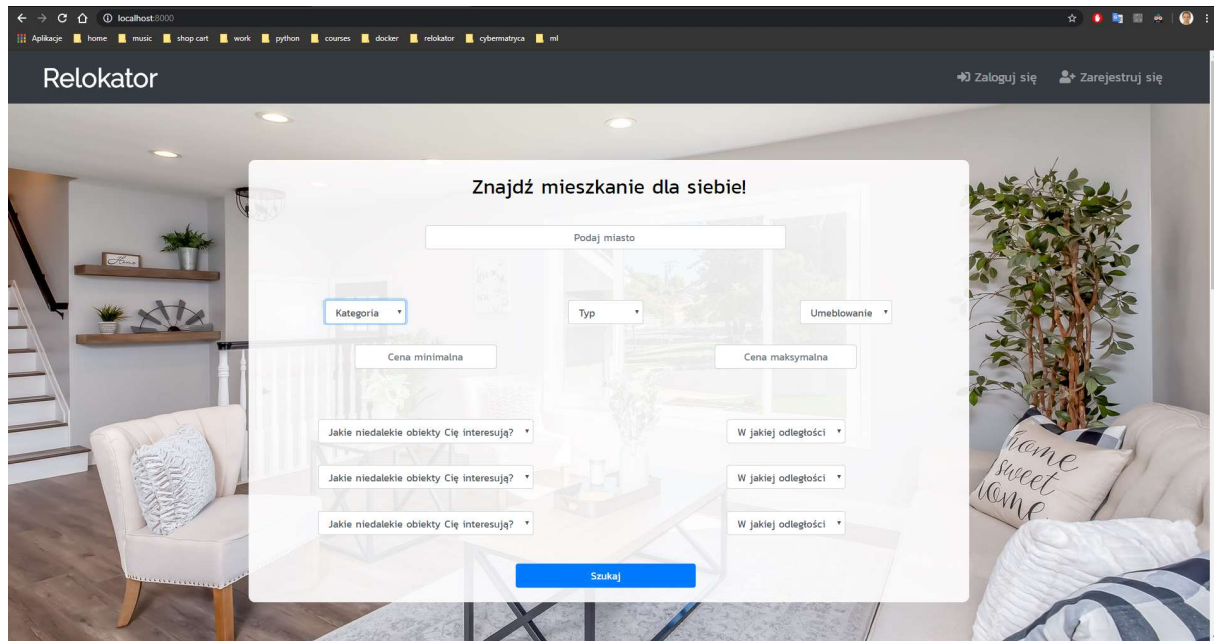
```
def save(self, *args, **kwargs):
    self.map_url = self.location.get_location_map_url(
        self.city, self.address
    )
    self.map_coord_x, self.map_coord_y = self.location.get_location_coords(
        self.city, self.address
    )
    super(Advert, self).save(*args, **kwargs)
```

Funkcja `get_location_map` zwracany wynik zapisuje do pola **map_url** zapisywanego obiektu, przyjmując wartości pól **city**, **address** jako argumenty wywołania. Podobna sytuacja zachodzi w przypadku funkcji `get_location_coords`, która zapisuje szerokość i długość geograficzną do pól **map_coord_x** oraz **map_coord_y** kreowanego obiektu bazując na wartościach pól **city** oraz **address**.

Ostatnią z metod klasy `GoogleMaps` jest funkcja **filter_list_of_adverts** wykorzystywana w drugim etapie wyszukiwania ogłoszeń. Danymi wejściowymi do danej metody jest lista ogłoszeń wynikowych pierwszego etapu, typ obiektów i odległość od których szukane nieruchomości mogą być maksymalnie oddalone. Iterując po obiektach w przekazanej jako argument funkcji liście wykonywane są zapytania do **Google Places API** gdzie jako parametry wysyłane są wartości pól **map_coord_x** oraz **map_coord_y** wraz wartościami zmiennych **object_type** i **radius** przekazywanymi jako argumenty do opisywanej funkcji. Po konwersji odpowiedzi serwera z formatu JSON na słownik otrzymywana jest lista obiektów. Jeśli w liście znajduje się co najmniej obiekt, oznacza to że ogłoszenie znajdujące się w obecnej iteracji spełnia parametry wyszukiwania i zostaje on dodany do listy ostatecznie zwracanych do użytkownika ogłoszeń **filtered_list_of_adverts**.

5.9 Wyszukiwanie ogłoszeń

Głównym procesem biznesowym aplikacji jest wyszukiwanie ogłoszeń w systemie. W tym celu, na stronie głównej aplikacji została zaimplementowana dwuetapowa wyszukiwarka, która filtruje ogłoszenia zarówno to podstawowych parametrach zawartych w ogłoszeniu jak: cena, kategoria mieszkania, typ najmu, umeblowanie, czy miasto jak i parametrach dodatkowych, jak odległość szukanych mieszkań od obiektów danego typu.



Rysunek 5.14 Wyszukiwarka ogłoszeń na stronie głównej aplikacji

Po wprowadzeniu przez użytkownika parametrów wyszukiwania ogłoszenia i naciśnięciu przycisku „Szukaj” wywoływany jest widok z modułu **searches** projektu, metoda o nazwie **search_view**. Funkcja ta została przedstawiona poniżej.

```
def search_view(request):  
  
    query = request.GET.get("q", None)  
  
    parameters = get_search_parameters(request)  
  
    if request.user.is_authenticated:  
        user = request.user  
    else:  
        user = None  
  
    context = {"query": query, "parameters": parameters}
```

```

if query is not None:

    SearchQuery.objects.create(user=user, query=query)

    # filtrowanie po parametrach bazowych (Etap I)
    adverts_list = Advert.objects.search(query=query, parameters=parameters)

    # filtrowanie po priorytetach lokalizacyjnych (Etap 2)
    if parameters['location_priority'] or
       parameters['location_priority_1'] or
       parameters['location_priority_2']:
        adverts_list = SearchQuery.location.filter_list_of_adverts(
            list(adverts_list),
            parameters["location_priority"],
            parameters["location_priority_radius"],
        )

    context["adverts_list"] = adverts_list
    context["counter"] = len(adverts_list)

return render(request, "searches/search-view.html", context)

```

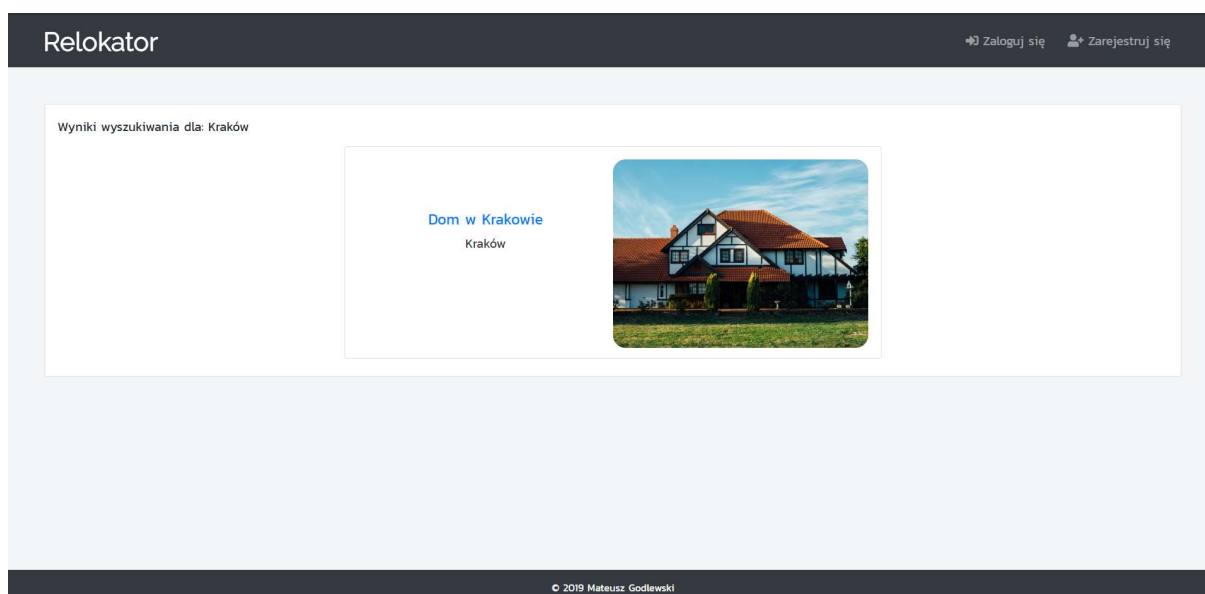
Przed rozpoczęciem filtrowania przygotowywane są odpowiednio wartości przekazanych przez użytkownika parametrów przez funkcję **get_search_parameters**, która jako argument przyjmuje żądanie wysłane przez użytkownika jako zapytanie, a zwraca słownik przetworzonych parametrów, metoda ta sprawdza czy wartości poszczególnych pól nie są puste oraz zmienia ich typ w celach dalszego przetwarzania. Po przygotowaniu parametrów, sprawdzane zostaje czy użytkownik który wysłał zapytanie jest użytkownikiem zalogowanym.

Następnie następuje przejście do właściwego etapu filtrowania ogłoszeń. Na początku, tworzony zostaje obiekt **SearchQuery**, który zawiera dane o pojedynczym zapytaniu i zapisuje je do bazy danych. Dzięki temu w bazie danych przechowywana jest historia zapytań. Następnie z bazy danych, z tabeli przechowującej dane ogłoszeń zostają pobierane te które spełniają podane przez użytkownika parametry podstawowe i zapisywane są do zmiennej **advert_list**, w ten sposób wykonywany jest pierwszy w dwóch etapów przedstawionych na rysunku 4.7. W etapie drugim lista **advert_list** przekazywana jest wraz z parametrami priorytetu lokalizacji zawierającymi typ obiektów oraz ich odległość od potencjalnych nieruchomości do metody **filter_list_of_adverts** klasy **GoogleMaps**, której

działanie zostało przedstawione w poprzednim podrozdziale. Przefiltrowana w ten sposób lista ogłoszeń zostaje użytkownikowi wyświetlona za pomocą szablonu `search_view.html`.

Dla przykładu został poniżej przedstawiony widok wyszukiwania umeblowanych domów na sprzedaż w Krakowie, których cena rozpoczyna się od 120 000 PLN znajdujących się nie dalej niż 1 kilometr od centrum handlowego oraz maksymalnie 3 kilometry do najbliższej przychodni lekarskiej.

Rysunek 5.15 Wyszukiwanie ogłoszeń z podanymi parametrami



Rysunek 5.16 Wyniki wyszukiwania ogłoszeń

6. TESTY APLIKACJI

7. PODSUMOWANIE

LITERATURA

- [1] L. Madeyski, M. Stochmialek, *Architektura nowoczesnych aplikacji internetowych*, Wydziałowy Zakład Informatyki, Wydział Informatyki i Zarządzania Politechnika Wrocławska.
- [2] <https://docs.microsoft.com/pl-pl/dotnet/architecture/modern-web-apps-azure/modern-web-applications-characteristics>
- [3] <https://instagram-engineering.com/web-service-efficiency-at-instagram-with-python-4976d078e366>
- [4] <https://quintagroup.com/cms/python/google>
- [5] <https://labs.spotify.com/2013/03/20/how-we-use-python-at-spotify/>
- [6] <https://medium.com/netflix-techblog/python-at-netflix-86b6028b3b3e>
- [7] <http://www.bagazowe.com.pl/dlaczego-ludzie-sie-przeprowadzaja>
- [8] <https://nieruchomosci.dziennik.pl/kupno-i-wynajem/artykuly/576246,mieszkanie-wybor-dom-nieruchomosci-pieniadze-deweloper.html>
- [9] <https://media.ing.pl/informacje-prasowe/926/pr/381312/kupujac-mieszkania-polacy-zwracaja-wieksza-uwage-na-okolice-niz-cene-w>
- [10] <https://mfiles.pl/pl/index.php/Nieruchomo%C5%9B%C4%87>
- [11] https://pl.wikipedia.org/wiki/Umowa_najmu
- [12] <https://pl.wikipedia.org/wiki/Nieruchomo%C5%9B%C4%87>
- [13] <https://blog.express-przeprowadzki.pl/7-cech-ktorymi-powinna-cechowac-sie-profesjonalna-firma-przeprowadzkowa/>
- [14] <https://pl.wikipedia.org/wiki/OLX>
- [15] <https://pl.wikipedia.org/wiki/Otodom>

[16] <https://developers.google.com/maps/documentation/geocoding/start>

[17] <https://developers.google.com/places/web-service/search>

[18] [https://en.wikipedia.org/wiki/Django_\(web_framework\)](https://en.wikipedia.org/wiki/Django_(web_framework))

[19] <https://pl.wikipedia.org/wiki/SQLite>

[20] https://pl.wikipedia.org/wiki/Mapy_Google

[21] [https://en.wikipedia.org/wiki/Bootstrap_\(front-end_framework\)](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework))

[Rysunek 2.1] <https://gearheart.io/media/images/SPA-1-768x454.original.png>

[Rysunek 2.2] <https://blog.soshace.com/wp-content/uploads/2019/10/Flask-vs.-Django.-Lets-Choose-the-Right-Framework-for-the-Job-Python-web-frameworks-popularity.png>

[Rysunek 2.3] <https://reliablecounter.com/blog/wp-content/uploads/2018/10/instagram.jpg>

[Rysunek 2.4] https://myapple.pl/uploads/image/file/1/c/0/big_78c50854-52b1-42f5-bb52-679065f321c0.png

[Rysunek 3.1] <https://d2xhqddaxyaju6.cloudfront.net/file/attachment-s/1219973/3d/s-640-x.jpg>

[Rysunek 3.2, rysunek 3.3, rysunek 3.4] <https://www.olx.pl/>

[Rysunek 3.5] <https://www.otodom.pl/>

[Rysunek 3.6] <https://fixly.pl/>

[Rysunek 4.5] <https://lh3.googleusercontent.com/EMzbsH0qJXweaoeOxPia96kx0u2h9b-QoEggREKjjsPwBdEn4cO7zBTai1cywpw4TvZrOgJTRPMc4GNdykXN-w>

[Rysunek 4.6] <https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSUf7uVtyL7pK9DSgl9r9Aw6aJ2TBflhcKhcJEuj9BrzA1u66dR&s>