UNIWERSYTET EKONOMICZNY W KRAKOWIE

KOLEGIUM NAUK O ZARZĄDZANIU I JAKOŚCI

INSTYTUT INFORMATYKI, RACHUNKOWOŚCI I CONTROLLINGU

KIERUNEK: INFORMATYKA STOSOWANA

SPECJALNOŚĆ: INŻYNIERIA OPROGRAMOWANIA

Yahor Lashko

Nr albumu: 219145

Badanie procesu tworzenia aplikacji internetowej na przykładzie frameworka Django

Praca licencjacka

Promotor
dr Paweł Konkol

Kraków 2023

Spis treści

Wstęp

W dzisiejszej erze cyfrowej aplikacje internetowe stały się integralną częścią naszego życia, zmieniając sposób, w jaki wchodzimy w interakcje, komunikujemy się i korzystamy z różnych usług. Wraz z szybkim rozwojem platform internetowych, zapotrzebowanie na wydajne i przyjazne dla użytkownika aplikacje internetowe gwałtownie wzrosło. W związku z tym zrozumienie procesu tworzenia takich aplikacji zyskało ogromne znaczenie.

Dana praca dyplomowa ma na celu zagłębienie się w zawiłości tworzenia aplikacji internetowych poprzez przeprowadzenie analizy procesu rozwoju, ze szczególnym naciskiem na użycie Django, potężnego frameworka internetowego napisanego w Pythonie. Aby zilustrować ten proces, zostanie przeanalizowany proces tworzenia strony internetowej poświęconej recenzowaniu restauracji, na której użytkownicy mogą publikować swoje restauracje i dzielić się swoimi doświadczeniami i opiniami na ich temat.

W pierwszym rozdziale pracy zostanie opisany proces inicjowania projektu, polegający na analizie podobnych usług, planowaniu wymagań i wyborze technologii do realizacji projektu. Drugi rozdział opisuje projektowanie aplikacji i jej komponentów: bazy danych, interfejsu użytkownika i opiera się na znaczeniu zarządzania danymi w tworzeniu aplikacji internetowej. Wykonanie projektu składa się z tworzenia projektu w kodzie, struktury projektu, tworzenia interfejsu użytkownika, bazy danych, a także mojego zrozumienia tematów specyficznych dla Django oraz problemów i rozwiązań, które napotkałem podczas tworzenia tego projektu. Trzecia część składa się z metod testowania projektu i sposobu jego wdrażania.

Aplikacje internetowe, takie jak platformy z recenzjami restauracji, odgrywają ważną rolę we współczesnym społeczeństwie, pomagając użytkownikom w podejmowaniu świadomych decyzji dotyczących lokali gastronomicznych. Oceniając rozwój strony internetowej opartej na Django, ta praca ma na celu rzucenie światła na podstawowe zasady, metodologie i wyzwania związane z tworzeniem takich aplikacji.

Django, znane ze swojej prostoty, wszechstronności i obszernej dokumentacji, oferuje programistom solidny framework, który przyspiesza proces tworzenia aplikacji internetowych. Przez pryzmat tego szeroko stosowanego frameworka, krok po kroku

zostanie zbadany proces tworzenia strony internetowej z recenzjami restauracji, od konceptualizacji do wdrożenia.

Rozdział 1
Inicjowanie projektu

Etap inicjowania projektu to początkowy etap w cyklu życia projektu, który obejmuje planowanie, analizę i przygotowanie projektu do jego uruchomienia. Jest to kluczowy etap, w którym określa się cele projektu, identyfikuje się jego zakres, ocenia się wykonalność, identyfikuje się interesariuszy oraz określa się zasoby i harmonogram projektu. Podczas etapu inicjowania projektu następuje zrozumienie problemu, który ma być rozwiązany, oraz określenie celów projektu.

Etap inicjowania projektu obejmuje również identyfikację interesariuszy, czyli osób lub grup, które mogą być zaangażowane lub mieć wpływ na projekt. Ważne jest zrozumienie ich potrzeb, oczekiwań i wpływu na projekt, co umożliwia lepsze zarządzanie nimi w trakcie realizacji projektu.

Wreszcie, w etapie inicjowania projektu określa się zasoby potrzebne do realizacji projektu, takie jak budżet, personel, technologia i inne narzędzia. Tworzony jest także wstępny harmonogram projektu, który określa sekwencję działań i ich planowane terminy. W rezultacie etap inicjowania projektu kończy się formalnym zatwierdzeniem projektu przez odpowiednie autorytety i uzyskaniem zielonego światła do przejścia do następnego etapu, którym jest planowanie projektu.

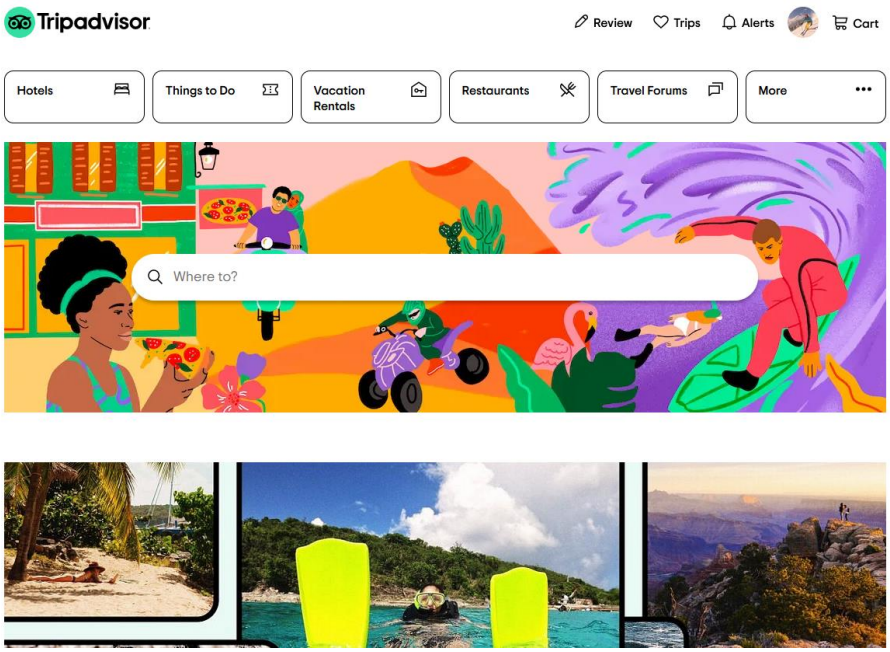
1.1 Funkcjonalna analiza istniejących rozwiązań

Podobne usługi są świadczone przez TripAdvisor i Google.

Tripadvisor to popularna strona internetowa dla podróżnych, która pozwala użytkownikom wyszukiwać i recenzować hotele, restauracje, zajęcia i inne firmy związane z podróżami. TripAdvisor został założony w 2000 roku i od tego czasu jest jednym z liderów w swojej dziedzinie. Funkcjonalność i wygląd serwisu zmieniały się wielokrotnie, aby dostosować się do obecnych standardów i przyciągnąć nowych klientów (Wikipedia contributors, 2023).

W tej chwili projekt strony ma minimalistyczny styl z całkowicie białym tłem i kanciastymi obrazkami. Interfejs jest dobrze zorganizowany i intuicyjny, z wyraźnymi wezwaniami do działania i pomocnymi podpowiedziami. Na stronach wszelkiego rodzaju informacje o danym miejscu, jak koszty, oceny, popularność i średni czas spędzony w danym miejscu. Dodatkowe informacje o obiekcie, takie jak bezpieczeństwo COVID czy dostępność dla osób niepełnosprawnych oraz najczęściej zadawane pytania także są dostępne na stronie. Recenzje w formie listy znajdują się na dole strony i zawierają ocenę ogólną, nazwę i zdjęcie profilowe recenzenta oraz tekst recenzji.

Jednym z kluczowych powodów sukcesu TripAdvisora jest wiarygodność - serwis podjął działania mające na celu ograniczenie fałszywych recenzji, np. stosując moderację kontentu. Pracownicy ręcznie i za pomocą narzędzi programowych rozpoznają nieodpowiednie recenzje i recenzje pozostawione przez konkurentów w celu zaniżenia oceny miejsca.

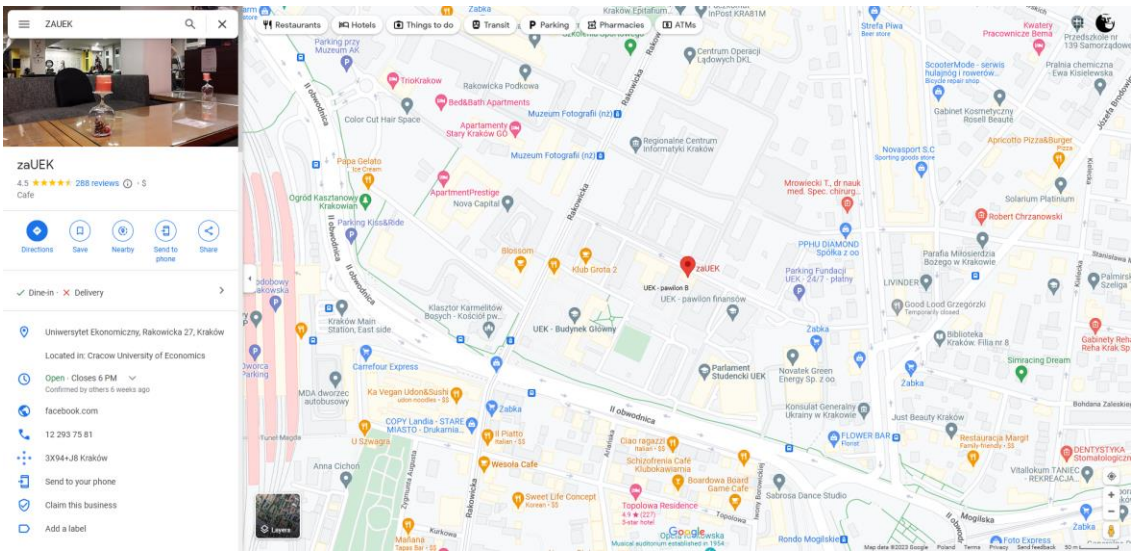


Rys. 1 - Strona internetowa tripadvisor.com.

Źródło: TripAdvisor https://www.tripadvisor.com/

Główne funkcje map Google to wyszukiwanie lokalizacji i nawigacja, ale pozwalają one także na wyszukiwanie różnych obiektów według zapytań i pozostawianie opinii na ich temat. Mapy Google, podobnie jak TripAdvisor, pozwalają na wyszukiwanie miejsc według różnych kryteriów, ale w przypadku map mamy znacznie mniejszą funkcjonalność: przeglądanie miejsc, wyświetlanie dodatkowych informacji przy wyborze miejsca,

przeglądanie i pozostawianie recenzji, różnego rodzaju integracje z serwisami zakupowymi i rezerwacyjnymi w celu szybkiego dostępu do tych funkcji bezpośrednio z Map Google.



Rys. 2 - Strona internetowa maps.google.com.

Źródło: Google Maps https://www.google.com/maps

1.2 Wymagania projektu. Poziomy biznesowy, użytkownika

Kompleksowy zestaw wymagań jest kluczowy dla każdego projektu oprogramowania. Wymagania identyfikują potrzeby i cele biznesowe produktu na wysokim poziomie. Wyjaśniają również cechy, funkcjonalność, zachowania i wydajność, których oczekują interesariusze.

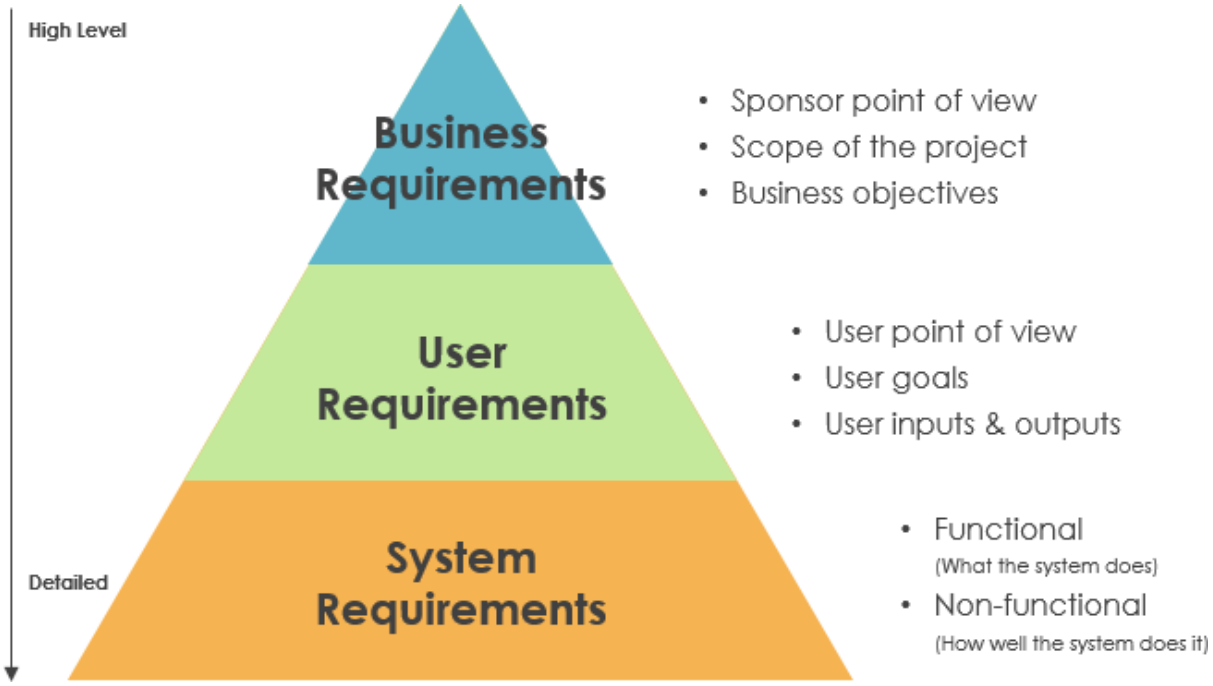
Wymagania dotyczące oprogramowania są sposobem na zidentyfikowanie i wyjaśnienie, dlaczego, co i jak w aplikacji biznesowej. Odpowiednio udokumentowane wymagania dotyczące oprogramowania tworzą mapę drogową, która prowadzi zespół programistów do szybkiego zbudowania właściwego produktu przy minimalnych kosztownych przeróbkach (Bigelow, 2020).

Aby zapewnić jasność i kontekst dla tej kwestii, wymagania są często kategoryzowane według 3 poziomów: biznesowy, użytkownika i produktu.

Wymagania biznesowe koncentrują się na wysokopoziomowych celach i zadaniach projektu z perspektywy biznesowej. Nakreślają one strategiczną wizję i zapewniają ramy dla projektu. Wymagania biznesowe zazwyczaj obejmują interesariuszy na poziomie wykonawczym lub kierowniczym.

Wymagania użytkownika koncentrują się na potrzebach, oczekiwaniach i doświadczeniach użytkowników końcowych lub klientów produktu. Wymagania te zapewniają wgląd w to, jak produkt powinien funkcjonować oraz jakie funkcje i możliwości powinien posiadać.

Wymagania produktowe koncentrują się na specyfikacjach technicznych i szczegółach niezbędnych do wdrożenia projektu. Wymagania te wypełniają lukę między wymaganiami biznesowymi a wymaganiami użytkowników, przekładając je na praktyczne wytyczne dla zespołu programistów (Project Management, b.r.; Bigelow, 2020).



Rys. 3 - Poziomy wymagań projektu.

Źródło: (Project Management: Different Types of Requirement)

Poziom biznesowy

- Klient musi mieć możliwość przeglądania i pozostawiania opinii o restauracjach,
- Klient musi mieć możliwość tworzenia i zarządzania profilem klienta,
- Klient musi mieć możliwość stworzenia profilu firmowego pod warunkiem istnienia profilu klienta,
- Pod warunkiem posiadania profilu klienta, możliwe jest pozostawianie recenzji restauracji,

- Pod warunkiem posiadania profilu firmy, użytkownik ma możliwość tworzenia strony restauracji
- Należy wdrożyć system moderowania recenzji i kont,
- Konta firm i strony restauracji muszą być moderowane przed ich utworzeniem.

Poziom użytkownika

- Umożliwić klientowi utworzenie profilu poprzez wprowadzenie nazwy, adresu e-mail, pseudonimu, hasła,
- Umożliwić klientowi zmianę szczegółów profilu, takich jak pseudonim, nazwa, zdjęcie profilowe,
- Umożliwić klientowi stworzenie profilu firmy poprzez wpisanie jej nazwy, numeru, adresu,
- Umożliwić klientom posiadającym profil klienta wystawianie opinii o restauracjach według jakości jedzenia, jakości obsługi, cen,
- Umożliwić klientowi posiadającemu profil firmowy tworzenie profili restauracji, wskazując nazwę, krótki i pełny opis, kuchnię, adres, współrzędne (dla widoku mapy) oraz zdjęcia,
- Strona powinna automatycznie rekomendować restauracje w zależności od restauracji, dla których użytkownik pozostawił recenzję i oceny w recenzji,
- Umożliwić klientowi oznaczenia recenzji innego klienta oceną pozytywną lub negatywną,
- Moderator powinien mieć możliwość usuwania recenzji i blokowania kont użytkowników,
- Pracownik serwisu powinien mieć możliwość moderowania (usuwania recenzji, moderowania profili firmowych, profili użytkowników).

1.3 Wymagania projektu. Poziom produktu

Tworzenie konta użytkownika:

- Pokaż stronę tworzenia użytkownika,
- Zweryfikuj dane, w przypadku niezgodności (adres e-mail jest zajęty, nazwa użytkownika jest zbyt długa, hasło jest zbyt proste) pokaż komunikat z prośbą o zmianę danych,
- Wyślij wiadomość e-mail z kodem weryfikacyjnym i poproś o wprowadzenie kodu, aby utworzyć konto,
- Utwórz konto.

Strona główna:

- Pokaż użytkownikowi bez profilu wybór popularnych restauracji w formie "Kart", a zalogowanemu na koncie kilka selekcji według parametrów takich jak miasto, kuchnia, liczba recenzji,
- Wyświetl filtry z możliwością wyboru kuchni, miasta, cen,
- Wyszukaj według nazwy i opisu.

Edycja konta użytkownika:

- Wyświetl formularz do edycji danych użytkownika,
- Sprawdź, czy dane wprowadzone przez klienta mogą być zapisane w systemie,
- Zapisz dane do systemu.

Tworzenie profilu firmy:

- Pokaż formularz do tworzenia profilu firmy,
- Zweryfikuj dane wprowadzone przez użytkownika,
- Utwórz prośbę o weryfikację przez moderatora.

Edycja profilu firmy:

- Pokaż formularz do tworzenia profilu firmy,

- Zweryfikuj dane wprowadzone przez użytkownika,
- Utwórz prośbę o weryfikację przez moderatora.

Strona usunięcia recenzji:

- Poproś o potwierdzenie usunięcia recenzji,
- Usuń recenzję.

Tworzenie profilu restauracji:

- Pokaż formularz tworzenia restauracji,
- Sprawdź w profilu użytkownika możliwość utworzenia strony restauracji,
- Zweryfikuj wprowadzone dane,
- Wprowadź dane do systemu.

Strona moderacji recenzji:

- Wyświetl listy recenzji z przeciskami do weryfikacji, usuwania, blokowania użytkownika,
- Pokaż listę recenzji w kolejności od najgorszej oceny,
- Wykonaj akcję moderatora zgodnie z wybraną opcją.

Strona moderacji restauracji:

- Pokaż listę kart restauracji do moderacji, tak jak klienci widzieliby je na stronie głównej, z opcją przejścia do strony restauracji,
- Pokaż dodatkowe elementy na stronie restauracji, jak przyciski do potwierdzenia lub odrzucenia, lub pole z powodem odrzucenia,
- W zależności od wybranej opcji, odrzuć lub potwierdź umieszczenie.

Strona moderacji profili firmowych:

- Pokaż listę profili firmowych przekazanych do weryfikacji, z odpowiednimi przyciskami "Potwierdź weryfikację", "Odrzuć weryfikację",
- Jeśli wybrano odmowę, wyświetl pole do wpisania przyczyny odmowy,

- W zależności od wyboru moderatora, dodaj do profilu firmy potwierdzenie weryfikacji lub odrzucenie.

Strona utworzenia recenzji:

- Pokaż formularz tworzenia recenzji - ocena w 3 kryteriach od 1 do 10: jakość jedzenia, jakość obsługi, ceny; pole na recenzję tekstową,
- Sprawdź, czy użytkownicy mogą napisać recenzję o restauracji,
- Umieść recenzję na stronie restauracji.

Strona restauracji:

- Pokaż nazwę restauracji, zdjęcia, pełny opis, widget mapy z lokalizacją restauracji, przycisk do umieszczenia recenzji (jeśli użytkownik ma recenzje dla tej restauracji, przycisk do usunięcia), lista recenzji. Zaznacz przycisk obok każdej recenzji, pozwalający na wystawienie jej oceny (pozytywna, negatywna),
- Pokaż przycisk zostaw recenzję, jeśli ten użytkownik nie utworzył strony tej restauracji i nie ma recenzji dla tej restauracji.

1.4 Użyte technologie

Tabela 1. Opis użytych technologii.

Technologia	Opis
Python/Django	Wykorzystane do stworzenia części backendowej aplikacji.
HTML/CSS/JavaScript	Wykorzystane do stworzenia części frontendowej aplikacji.
Docker	Wykorzystane jako platforma do tworzenia i uruchomienia uruchomienia instancji działającej aplikacji.
Bash	Wykorzystane do tworzenia skryptów wdrożeniowych.

Źródło – opracowanie własne

Python

Python to wysokopoziomowy język programowania ogólnego przeznaczenia, który służy między innymi do tworzenia aplikacji internetowych. Język ten skupia się na zwiększeniu produktywności programistów i czytelności kodu. Python obsługuje kilka paradygmatów programowania: strukturalny, obiektowy, funkcjonalny, imperatywny i aspektowy. Język charakteryzuje się dynamicznym typowaniem, automatycznym zarządzaniem pamięcią, pełną introspekcją, mechanizmem obsługi wyjątków, obsługą

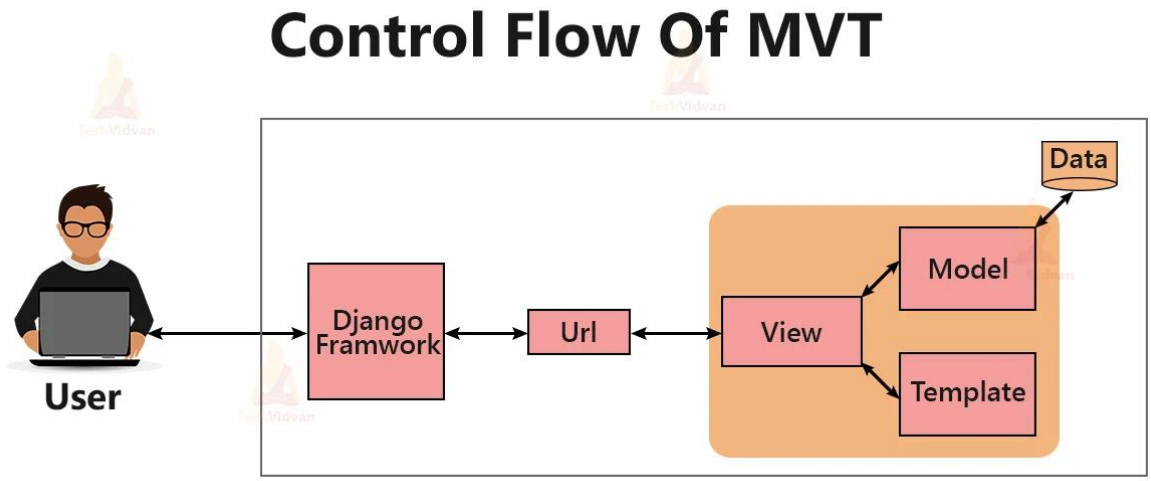
wielowątkowości oraz wygodnymi strukturami danych wysokiego poziomu. Kod Pythona jest zorganizowany w funkcje i klasy, które mogą być łączone w moduły, które z kolei mogą być łączone w pakiety. Python jest zwykle używany jako interpretowany, ale może być skompilowany do kodu bajtowego Java i MSIL (w ramach platformy .NET) (Downey, 2015; Team, 2021a).

Django

Django to framework do budowania aplikacji internetowych z wykorzystaniem języka programowania Python. Django powstał w 2005 roku, gdy web developerzy z Lawrence Journal-World zaczęli używać Pythona jako języka do budowania stron internetowych. A w 2008 roku pojawiło się pierwsze publiczne wydanie tego frameworka. Do dnia dzisiejszego wciąż się rozwija. Tak więc aktualną wersją frameworka w chwili pisania tego tekstu jest wersja 4.2, która została wydana w 2023 roku. Każde nowe wydanie frameworka ukazuje się średnio co 8 miesięcy. Ponadto cały czas wydawane są aktualizacje i łaty bezpieczeństwa.

Django jest dość popularny. Jest używany na wielu stronach, w tym Pinterest, PBS, Instagram, BitBucket, Washington Times, Mozilla i wiele innych. Django jest bezpłatny. Jest rozwijany jako open source, jego kod źródłowy jest otwarty i można go znaleźć w repozytoriach na GitHubie. Na Django można zbudować szeroką gamę aplikacji internetowych, od małych osobistych stron do wysoko obciążonych, złożonych usług internetowych. Django domyślnie oferuje funkcjonalność dla wielu typowych zadań, takich jak systemy uwierzytelniania, generowanie sitemap, itp., więc nie musimy wymyślać koła na nowo i po prostu wziąć komponenty, które są już dostarczane z frameworkiem. Django kładzie duży nacisk na bezpieczeństwo, więc framework pomaga programistom uniknąć wielu typowych problemów bezpieczeństwa, takich jak, na przykład, zastrzyki sql.

Django implementuje wzorzec architektury Model-View-Template, w skrócie MVT, który jest właściwie modyfikacją popularnego w programowaniu webowym wzorca MVC (Model-View-Controller) (Wikipedia contributors, 2023).



Rys. 4 - Przepływ kontroli MVT.

Źródło: (All You Need to Know About Django MVT Architecture).

Podstawowe elementy wzorca:

1. Dyspozytor URL: po otrzymaniu żądania na podstawie podanego adresu URL określa, który zasób powinien obsłużyć żądanie.
2. Widok: odbiera żądanie, przetwarza je i wysyła odpowiedź do użytkownika. Jeśli model i baza danych muszą być dostępne, aby przetworzyć żądanie, wtedy View wchodzi w interakcję z nimi. Do tworzenia odpowiedzi można wykorzystać szablony lub wzorce. W architekturze MVC kontrolery (nie widoki) odpowiadają temu komponentowi.
3. Model: opisuje dane wykorzystywane w aplikacji. Poszczególne klasy odpowiadają tabelom w bazie danych.
4. Szablon: reprezentuje logikę widoku w postaci wygenerowanego znacznika html. W MVC komponent ten odpowiada View, czyli widokom (Wikipedia contributors, 2023; Korsun & Korsun, 2023).

JavaScript

JavaScript to tekstowy język programowania używany do tworzenia interaktywnych stron internetowych. Jest to język wysokiego poziomu, co oznacza, że jest łatwy do czytania i pisania, i jest interpretowany, co oznacza, że nie musi być kompilowany przed uruchomieniem. JavaScript jest zorientowany obiektowo, co oznacza, że organizuje kod w obiekty, które są samodzielne i wielokrotnego użytku. JavaScript jest również dynamicznie typowany, co oznacza, że zmienne nie muszą być deklarowane z określonym typem danych.

W trakcie rozwoju języka, programiści JavaScript stworzyli biblioteki, frameworki i praktyki programistyczne i zaczęli używać go poza przeglądarkami internetowymi. Obecnie JavaScript można wykorzystywać zarówno do programowania po stronie klienta, jak i serwera. Chociaż jest najbardziej znany jako język skryptowy dla stron internetowych, wiele środowisk niebędących przeglądarkami również go używa, takich jak Node.js, Apache CouchDB oraz Adobe Acrobat. JavaScript jest prototypowym, wieloparadygmatowym, jednowątkowym, dynamicznym językiem, wspierającym style obiektowe, imperatywne i deklaratywne (np. programowanie funkcjonalne) (AWS, b.r.).

HTML/CSS

HTML to język znaczników hipertekstowych (HyperText Markup Language). Język ten służy do tworzenia stron internetowych. Jest on interpretowany (przetwarzany) przez przeglądarkę i prezentowany w formie czytelnej dla człowieka. HTML jest nieodłącznym elementem prawie każdej strony internetowej. HTML to przede wszystkim środek do logicznego oznaczania strony.

CSS jest językiem opisującym wygląd dokumentu, napisanym przy użyciu języka znaczników. Nazwa pochodzi od Cascading Style Sheets. Dzięki CSS web developer może ustawić dla strony i jej poszczególnych elementów różne kroje i wielkości czcionek, kolory elementów, wcięcia elementów od siebie, położenie poszczególnych bloków na stronie itp. HTML, CSS i JS współpracują ze sobą, tworząc pozytywne wrażenia użytkownika na każdej stronie. Podczas gdy HTML i CSS mogą głównie manipulować zawartością statyczną, mogą one integrować się z kodem JavaScript po stronie klienta, aby dynamicznie aktualizować zawartość (AWS, b.r.).

Docker

Docker to platforma open source, która umożliwia programistom budowanie, wdrażanie, uruchamianie, aktualizowanie i zarządzanie kontenerami - ustandaryzowanymi, wykonywalnymi komponentami, które łączą kod źródłowy aplikacji z bibliotekami systemu operacyjnego (OS) i zależnościami wymaganymi do uruchomienia tego kodu w dowolnym środowisku.

Kontenery są wykonywalnymi jednostkami oprogramowania, w których kod aplikacji jest pakowany, wraz z bibliotekami i zależnościami, we wspólny sposób, tak aby można go było uruchomić w dowolnym miejscu, niezależnie od tego, czy jest to desktop, czy chmura. Aby to osiągnąć, kontenery wykorzystują formę wirtualizacji systemu

operacyjnego, w której cechy systemu operacyjnego (w przypadku jądra Linux, mianowicie przestrzenie nazw i prymitywy cgroups) są wykorzystywane zarówno do izolowania procesów, jak i kontrolowania ilości zasobów procesora, pamięci i dysku, do których te procesy mają dostęp (Docker documentation, 2023).

W swoim rdzeniu, Docker pozwala uruchomić prawie każdą aplikację bezpiecznie izolowaną w kontenerze. Bezpieczna izolacja pozwala na uruchomienie wielu kontenerów na tym samym hoście w tym samym czasie. Lekka natura kontenera, który działa bez dodatkowego obciążenia hypervisora, pozwala osiągnąć więcej ze swojego sprzętu (Srivastav, b.r.).

Bash

Bash jest interpreterem języka poleceń dla systemu operacyjnego i jest kompatybilny z programami powłoki Bourne. Innymi słowy, zapewnia użytkownikom prosty i wydajny sposób interakcji z systemem i wykonywania różnych zadań, takich jak tworzenie plików, edycja tekstu i zarządzanie procesami. Polecenia są podstawowymi elementami składowymi Bash. Argumenty, zmienne, funkcje, potoki, przekierowania i symbole wieloznaczne są używane do modyfikowania i rozszerzania zachowania poleceń Bash. Skrypty Bash to programy napisane w języku Bash, które mogą automatyzować zadania, wykonywać złożone operacje i tworzyć niestandardowe narzędzia (Goyal, 2023).

Rozdział 2
Projektowanie, wykonanie

Etap projektowania to kolejny etap w cyklu życia projektu, który następuje po etapie inicjowania projektu. W tym etapie szczegółowo opracowuje się plany i strategie dotyczące realizacji projektu. Głównym celem etapu projektowania jest opracowanie kompletnego projektu, który uwzględnia wszystkie niezbędne informacje, rozwiązania techniczne i specyfikacje potrzebne do skutecznej realizacji projektu.

Etap wykonania projektu to jeden z kluczowych etapów w cyklu życia projektu, który następuje po etapie projektowania projektu. W tej fazie plany i strategie opracowane na etapie projektowania są wdrażane i realizowane.

Rola danych w aplikacji

W dzisiejszych aplikacjach internetowych dane odgrywają kluczową rolę w działaniu aplikacj: wpływają na wiele aspektów funkcjonowania tych aplikacji oraz na korzyści, jakie użytkownicy i przedsiębiorstwa mogą z nich czerpać. Każda aplikacja internetowa nie może zostać pozbawiona gromadzenia wszelkiego rodzaju danych klientów, takich jak dane osobowe dotyczące zachowania klienta podczas przeglądania, informacje o urządzeniu i przeglądarce użytkownika, a także różnorodne pliki cookies i dane do celów analitycznych.

Dane mogą pomóc zidentyfikować obszary napięć w przepływach użytkowników, określić, które projekty zapewniają najlepsze wyniki i określić, które prace będą miały największy wpływ na użytkowników. Informacje zwrotne mogą pomóc zidentyfikować wymagania użytkowników, o których możliwe, że nie pomyślano, a także możliwości rozszerzenia zakresu oferty w celu rozwiązania konkretnych problemów (The Council on Quality and Leadership, 2021; Ellingwood, 2017).

2.1 Pobieranie danych

First party / Third party

First party: Są to dane gromadzone przez firmę na temat jej własnych odbiorców, co oznacza, że śledzi ona sposób, w jaki ludzie korzystają z jej witryny. Żadna inna firma ani źródło zewnętrzne nie jest w to zaangażowane. Na przykład, przy robieniu zakupów online, witryna może gromadzić dane, takie jak oglądane elementy, ile użytkownik zapłacił i jak długo przebywał na każdej stronie.

Third party: Strony trzecie to zewnętrzne źródła, które nie są powiązane z odwiedzaną witryną. Ponieważ zbierają one dane z wielu witryn, firma może korzystać z zewnętrznych modułów śledzących na swojej stronie internetowej, aby uzyskać bardziej wszechstronny obraz odwiedzających witrynę. Na przykład ta sama witryna zakupów online może korzystać z zewnętrznych modułów śledzących, aby zrozumieć, w jaki sposób użytkownik robi zakupy w podobnych sklepach, jego dane demograficzne i inne zachowania związane z zakupami (What Is First-party Vs Third-party Data, b.r.).

Dane jawne / ukryte

Dane jawne to dane dostarczane bezpośrednio do witryny przez użytkownika poprzez wprowadzenie ich w polach tekstowych. Obejmuje to ustawienia, dane osobowe, takie jak imię i nazwisko, adres pocztowy, adres e-mail, konta w mediach społecznościowych, szczegóły płatności i inne. Ponieważ dane te są podawane bezpośrednio przez użytkownika, są one najbardziej wiarygodnym źródłem informacji o użytkowniku. Na stronach internetowych i w aplikacjach ten rodzaj danych jest wykorzystywany do obsługi podstawowych funkcji witryny, takich jak zarządzanie użytkownikami, dokonywanie płatności, rezerwacji, wysyłanie wiadomości itp.

Dane ukryte nie są wprowadzane bezpośrednio przez użytkownika, ale są uzyskiwane poprzez gromadzenie danych o użytkowniku i jego urządzeniu za pomocą skryptów. Może to obejmować analizy behawioralne, takie jak czas trwania sesji, odwiedzane strony lub profil urządzenia, ale może również obejmować wnioski wyciągnięte z dostarczonych danych, takie jak segment osobowości użytkownika, prawdopodobne harmonogramy pracy i snu. Ten rodzaj danych może być wykorzystany do analizy zachowania użytkownika na stronie, a także do identyfikacji użytkownika i jego aktywności na innych stronach internetowych (wymaga to uzyskania danych użytkownika od stron

trzecich). Dane ukryte pozwalają "spojrzeć na produkt z perspektywy użytkownika końcowego" i lepiej zrozumieć jego mocne i słabe strony (Douek, 2023).

"Odcisk palca", czyli sposób zidentyfikowania użytkownika przy użyciu tylko danych ukrytych

Specyficzny dla urządzenia "odcisk palca" to technologia gromadzenia danych, która służy do identyfikacji urządzenia i jego użytkownika w oparciu o unikalne cechy urządzenia i oprogramowania użytkownika. Unikalny odcisk palca może zawierać różne dane, w tym:

1. Identyfikator urządzenia: może to być numer interfejsu szeregowego (S/N) lub inny unikalny identyfikator urządzenia.
2. Informacje o systemie operacyjnym urządzenia.
3. Informacje o przeglądarce i jej ustawieniach.
4. Język i ustawienia regionalne: informacje o języku i ustawieniach regionalnych urządzenia.
5. Informacje o rozdzielczości ekranu.
6. Ustawienia systemowe: Informacje o ustawieniach systemowych, takich jak ilość pamięci RAM.
7. Inne funkcje, takie jak wersja zainstalowanych aplikacji, obecność lub brak niektórych sterowników itp.

Zestaw takich parametrów daje większą szansę na stwierdzenie, że jest to ten sam użytkownik podczas wyszukiwania ruchu użytkownika z identycznym zestawem parametrów, ale wirtualny "odcisk palca" nie może wykazać w 100% wiarygodnych wyników w identyfikacji użytkownika, więc musi być używany w połączeniu z inną, bardziej niezawodną metodą identyfikacji użytkownika.

Do zbierania danych potrzebnych do utworzenia unikalnego odcisku palca wykorzystywane są różne metody, w tym skrypty JavaScript i inne technologie, które są uruchamiane na urządzeniu użytkownika podczas odwiedzania strony internetowej. Dane zebrane za pomocą takich technik są przekazywane firmom zewnętrznym, które zajmują się analizą i targetowaniem reklam. Unikalny odcisk palca może być wykorzystywany do różnych celów, w tym do dostarczania spersonalizowanych treści, analizy zachowań użytkowników i ukierunkowanych reklam (Cover Your Tracks, b.r.; Crawford, 2021).

2.2 Przechowywanie danych

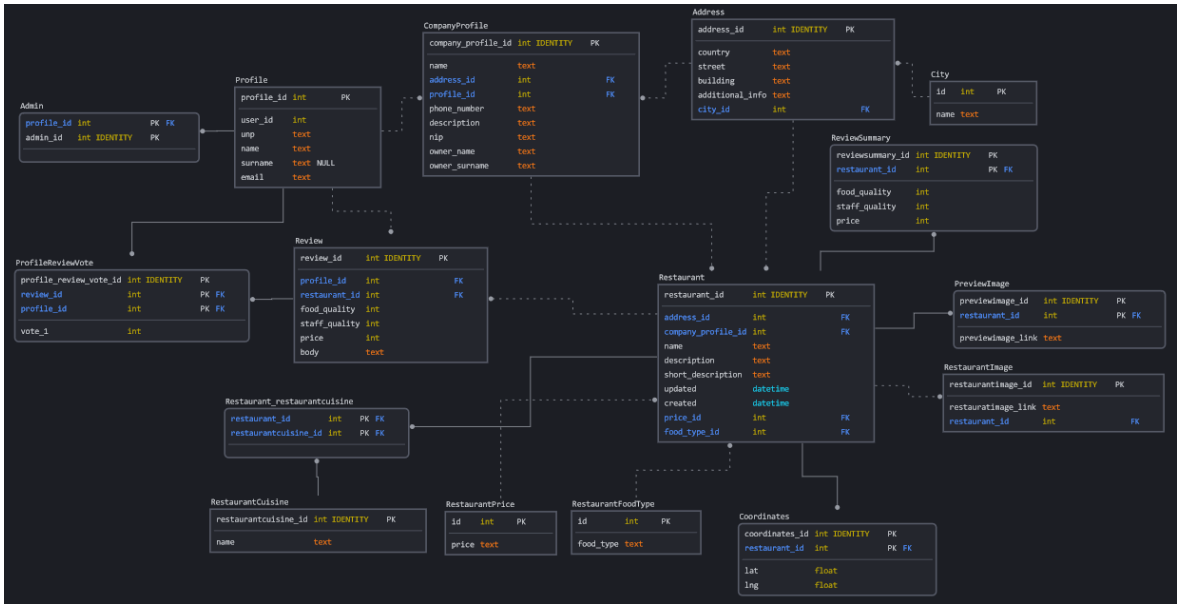
Przechowywanie danych wymaganych do prawidłowego działania witryny może odbywać się na różne sposoby, w zależności od konkretnego wdrożenia i zastosowanej technologii. Sposoby przechowywania danych obejmują:

1. System zarządzania bazą danych (DBMS) to oprogramowanie systemowe do tworzenia i zarządzania bazami danych. DBMS umożliwia użytkownikom końcowym tworzenie, ochronę, odczyt, aktualizację i usuwanie danych w bazie danych. Najbardziej rozpowszechniony typ platformy zarządzania danymi, DBMS zasadniczo służy jako interfejs między bazami danych a użytkownikami lub programami aplikacyjnymi, zapewniając, że dane są spójnie zorganizowane i pozostają łatwo dostępne (Mullins, 2022).
2. Przechowywanie plików w systemie plików serwera. Ta metoda przechowywania danych polega na tworzeniu plików zawierających dane w określonym formacie (np. przechowywanie danych o użytkownikach w plikach JSON) na serwerze. Ta metoda przechowywania danych może być przydatna w przypadku małych witryn z niewielką ilością informacji oraz do przechowywania danych, które nie nadają się do przechowywania w DBMS, np. obrazów.
3. Usługi przechowywania danych w chmurze, takie jak Amazon S3, Google Cloud Storage, Microsoft Azure. Dane w pamięci masowej mogą być zorganizowane na różne sposoby: w postaci bazy danych, pliku lub bloku. Usługi zapewniają podatność i odzyskiwanie danych, wirtualizację i dostępność pamięci masowej w chmurze, bezpieczeństwo pamięci masowej w chmurze, równoważenie obciążenia i migrację danych (AWS, b.r.).

W tym projekcie wybrano formę przechowywania danych w bazie danych SQL, ponieważ zapewnia ona wszystkie niezbędne funkcje do przechowywania danych na tego rodzaju stronie internetowej, a także może być łatwo przenoszona z lokalnej pamięci masowej do usług w chmurze, jeżeli będzie taka potrzeba, jednak przechowywanie danych w formie plików jest również wykorzystywane do przechowywania obrazów.

Kluczowymi obiektami przechowywanymi w bazie danych są profile użytkowników (tabela profile), restauracje (tabela restaurant), profile firm (tabela CompanyProfile) i recenzje użytkowników (tabela reviews). Dodatkowo zaprojektowane zostały tabele do

przechowywania profili administratorów, recenzji, zapytań o profile firm i adresów, a także zdjęć podglądowych i zdjęć restauracji. Układ projektu bazy danych jest następujący:



Rys. 5 - schemat projektu bazy danych.

Źródło: opracowanie własne

2.3 Wyświetlanie danych, czyli interfejs użytkownika

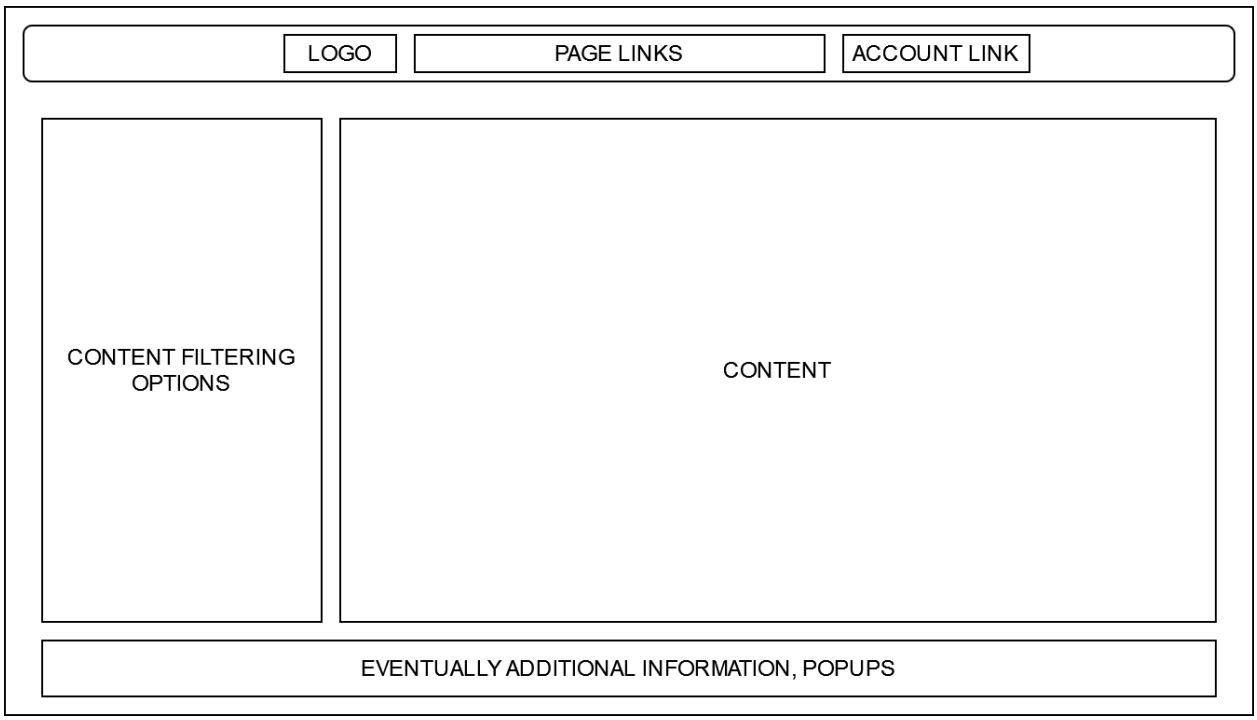
Architektura interfejsu użytkownika to połączenie między stroną internetową lub architekturą a jej użytkownikiem. Obejmuje każdy element projektu, który musi być obecny, aby użytkownicy mogli poruszać się po witrynie i podejmować pewne działania. Odnosi się do relacji między projektem a użytkownikiem podczas korzystania z niego.

Projekt strony jest niezwykle ważną częścią witryny internetowej. Wygląd ekranu powinien informować użytkownika o tym, jak z nim współdziałać i jakiego zachowania oczekiwać. Projektowanie ekranów jest komunikacją wizualną, mostem między wyglądem a interfejsem użytkownika.

Istnieje kilka podstawowych zasad tworzenia czytelnego interfejsu użytkownika:

1. Znane wzorce, znaczniki. Prawie każdy użytkownik korzysta z dziesiątek innych stron internetowych oprócz własnej, a wszystkie te witryny mają pewien rodzaj wzorca w swoim projekcie. To nie tylko upraszcza projekt strony, ale także pozwala użytkownikom szybciej znaleźć odpowiednie elementy interfejsu w znanych

miejscach. Wzór podobny do zamieszczonego niżej został wybrany dla tego projektu:



Rys. 6 - Schemat strony głównej.

Źródło: opracowanie własne

Ten rodzaj wzoru jest używany na wielu stronach internetowych opartych na treści, takich jak YouTube, a także na stronach sklepów wyszukiwarek, na przykład RTV Euro AGD (https://www.euro.com.pl/search.bhtml?keyword=B).

2. Użyteczność jest ważniejsza niż wygląd. Podczas gdy wygląd jest ważną częścią projektu strony, nie należy poświęcać użyteczności na rzecz wyglądu, ponieważ jednym z najważniejszych parametrów projektu jest czas między zrozumieniem przez użytkownika tego, co on chce zrobić na stronie, a odpowiedzią witryny. Obejmuje to zarówno intuicyjność interfejsu, jak i czas reakcji na bezpośrednie kliknięcia. Wszystkie elementy muszą być łatwo dostępne i dobrze widoczne na ekranie, w przeciwnym razie korzystanie z takiej strony będzie irytujące dla użytkownika.
3. Tylko niezbędne informacje. Strona powinna domyślnie zawierać tylko te informacje, których użytkownik potrzebuje do podjęcia dalszych działań, a dostarczanie dodatkowych informacji powinno być jedynie wynikiem zapytania o te konkretne informacje.

4. Koncentracja na elementach. Najczęściej używane elementy interfejsu użytkownika powinny być bardziej widoczne poprzez zwiększenie ich rozmiaru lub różnicy kolorów w stosunku do innych elementów na stronie. Jeśli wszystkie inne zasady są przestrzegane, ta nie będzie miała tak dużego znaczenia, ale nadal poprawi wrażenia użytkownika podczas interakcji ze stroną (Wilding, 1998).

Podsumowując, interfejs użytkownika odgrywa kluczową rolę w tworzeniu strony internetowej, ponieważ jest środkiem, za pomocą którego użytkownik wchodzi w interakcję z witryną i doświadcza jej. Dobrze zaprojektowany interfejs użytkownika może znacznie poprawić wrażenia użytkownika, ułatwiając mu nawigację, znajdowanie informacji i osiąganie celów na stronie. I odwrotnie, źle zaprojektowany interfejs użytkownika może frustrować użytkowników i powodować, że opuszczają witrynę. Dobry układ i wzór osiąga się poprzez harmonię wizualną, zapewniając wskazówki dotyczące funkcjonalności oraz przejrzystość i komunikację (White, 2022).

2.4 Podział kodu projektu na pliki, struktura projektu

Standardowa architektura Django składa się z czterech głównych komponentów:

1. Model (models.py) jest najważniejszą częścią aplikacji, która uzyskuje dostęp do danych na każde żądanie z dowolnej sesji. Każdy model jest standardową klasą Pythona. Mapowanie obiektowe (ORM) pozwala takim klasom na bezpośredni dostęp do baz danych. Gdyby nie było ORM, konieczne byłoby pisanie zapytań bezpośrednio w SQL. Model zapewnia lekki mechanizm dostępu do warstwy danych i hermetyzuje logikę biznesową oraz jest niezależny od aplikacji. Danymi można manipulować nawet z poziomu wiersza poleceń, bez konieczności korzystania z interfejsu użytkownika.
2. Widoki (views.py). Wykonują różne funkcje, w tym kontrolowanie zapytań użytkownika i dostarczanie kontekstu w zależności od roli użytkownika. View jest funkcją, która jest wywoływana w odpowiedzi na zapytanie adresu URL i zwraca kontekst.
3. Szablony (templates) są formą przedstawienia danych. Szablony Django są napisane w Django Template Language (DTL), który jest prostym i wydajnym językiem zaprojektowanym specjalnie do generowania HTML. Szablony DTL są plikami

tekstowymi, które używają specjalnych znaczników do wyznaczania miejsc, gdzie dane powinny być wstawione do HTML.

4. Mechanizm URL (urls.py) służy do zewnętrznego dostępu do widoków. Wyrażenia regularne wbudowane w adresy URL sprawiają, że mechanizm ten jest dość elastyczny. Pojedynczy widok może być skonfigurowany do wielu adresów URL, zapewniając dostęp do różnych aplikacji. Filozofia zakładek jest tutaj wspierana: adresy URL stają się samowystarczalne i zaczynają żyć niezależnie od widoku (T. Team, 2021).

Dla wygody dokonano również dalszego podziału logiki biznesowej na pliki:

1. forms.py - zawiera formularze do wprowadzania danych przez użytkownika i podstawowe metody walidacji danych, takie jak długość nazwy użytkownika, wykorzystuje również wbudowane metody walidacji Django, określając warunki walidacji w definicjach pól, np. max_length służy do sprawdzania długości adresu email: *email = forms.CharField(max_length=200).*
2. validators.py - zawiera metody walidacji wpisów danych, które wymagają implementacji dodatkowej logiki, takiej jak użycie zewnętrznych bibliotek lub własnych algorytmów.
3. services.py, selectors.py - zawierają większość logiki biznesowej projektu poprzez funkcje wywoływane z views.py. Służą również jako interfejs do bazy danych, usługi do tworzenia obiektów w bazie danych, selektory do zapytań do bazy danych. Przykładami funkcji z tych plików są funkcje do pobierania zawartości strony głównej dla konkretnego użytkownika, funkcja do tworzenia profilu użytkownika, funkcja do obliczania statystyk z recenzji użytkowników dla restauracji. Takie rozdzielenie poprawia czytelność kodu i ułatwia nawigację. Odciąża również plik views.py, w którym zwykle implementowana jest większość logiki.

2.5 Tworzenie bazy danych w Django models

Jeśli chodzi o wybór bazy danych dla aplikacji, Django oferuje wiele różnych opcji.

SQLite

Nazwa SQLite mówi sama za siebie - jest to "lekka" implementacja bazy danych SQL. W przeciwieństwie do PostgreSQL, MySQL i wielu komercyjnych baz danych, takich

jak Oracle lub MS SQL, baza danych SQLite nie jest samodzielnym serwerem, jest po prostu biblioteką, która implementuje interfejs dostępu do pliku dysku bazy danych. Podobnie jak inne "lekkie" implementacje zazwyczaj złożonych usług. SQLite świetnie nadaje się do szybkiego rozpoczęcia procesu rozwoju lub do małych witryn, w których nie trzeba instalować pełnego serwera internetowej bazy danych. Jednak po początkowej fazie szkolenia i w przypadku poważniejszych wdrożeń wymagane będzie coś bardziej zaawansowanego (Yigal, 2018).

PostgreSQL

PostgreSQL (często w skrócie "Postgres") to kompletny internetowy serwer baz danych z szerokim zakresem funkcji i długą historią rozwoju, będący jedną z wiodących aplikacji bazodanowych typu open source. Ta baza danych jest zalecana do użytku przez głównych deweloperów Django, którzy bardzo wysoko oceniają jej jakość. Aby móc wchodzić w interakcje z bazą danych Postgresql z poziomu kodu Pythona, potrzebna jest biblioteka psycopg2. Tworzenie baz danych i użytkowników w Postgres jest dość proste, Postgres domyślnie posiada oddzielne narzędzia wiersza poleceń, takie jak createuser i createdb, których nazwy mówią same za siebie (Yigal, 2018; Editor, 2019).

MySQL

Innym dominującym serwerem baz danych, rozpowszechnianym na zasadach open source, jest MySQL. W przeciwieństwie do niektórych innych serwerów baz danych, MySQL zawiera dwa wewnętrzne silniki baz danych z różnymi zestawami funkcji: pierwszy to MyISAM, który nie obsługuje transakcji i kluczy obcych, ale obsługuje wyszukiwanie pełnotekstowe. Drugi to InnoDB, który jest nowszy i ma więcej funkcji, ale obecnie nie obsługuje wyszukiwania pełnotekstowego. Istnieją również inne mechanizmy, ale te dwa są używane najczęściej.

Do organizowania interakcji z MySQL na platformie Django preferowana jest biblioteka python-mysql. Tworzenie baz danych w MySQL odbywa się zazwyczaj za pomocą wielofunkcyjnego narzędzia administracyjnego mysqladmin. Podobnie jak w przypadku Postgres, zanim będziemy mogli utworzyć nowe konto dla projektu na platformie Django, musimy ustalić nazwę i hasło superużytkownika bazy danych. Zazwyczaj superużytkownik nazywa się root i najczęściej nie posiada hasła początkowego (co należy poprawić, gdy tylko stanie się ono dostępne).

W przeciwieństwie do Postgres, zarządzanie użytkownikami w MySQL odbywa się całkowicie na poziomie bazy danych, więc aby utworzyć Django_user bazy danych, trzeba od razu użyć powłoki poleceń SQL (Yigal, 2018; Editor, 2019).

Na tym etapie zdecydowano się użyć SQLite ze względu na lekkość i łatwiejszą konfigurację, ale w przyszłości, jeśli trzeba będzie zorganizować osobny serwer dla bazy danych, możliwe jest przejście na inny typ bazy danych.

Przeniesienie bazy danych do models.py

Baza danych w Django jest tworzona przy użyciu modeli i ich relacji, podobnych do tabel i relacji w standardowym SQL. Aby stworzyć model, należy stworzyć klasę, która dziedziczy z klasy Django.db.models.Model i zawiera obiekty klasy Django.db.models.field działające jako kolumny do przechowywania danych. Na przykład, tabela restauracji w SQL wyglądałaby następująco:

```
CREATE TABLE restaurant (
  id INTEGER PRIMARY KEY,
  name TEXT NOT NULL,
  description TEXT NOT NULL DEFAULT '',
  company_profile_id INTEGER NOT NULL,
  address_id INTEGER NOT NULL,
  coordinates_id INTEGER NOT NULL
  updated DATETIME NOT NULL,
  created DATETIME NOT NULL,
  FOREIGN KEY (address_id)
    REFERENCES address (id)
      ON UPDATE CASCADE
      ON DELETE CASCADE,
  FOREIGN KEY (coordinates_id)
    REFERENCES coordinates (id)
      ON UPDATE CASCADE
      ON DELETE CASCADE,
  FOREIGN KEY (company_profile_id)
    REFERENCES company_profile (id)
      ON UPDATE CASCADE
      ON DELETE CASCADE
)
```

Rys. 7 - Kod tworzący tabelę restaurant w języku SQL.

Źródło: opracowanie własne

Implementacja w Django wygląda następująco:

```
class Restaurant(models.Model):
  company_profile = models.ForeignKey(CompanyProfile, on_delete=models.CASCADE, blank=False,
null=False)
  name = models.CharField(max_length=200)
  description = models.TextField(null=False, blank=False)
  short_description = models.TextField(max_length=625, null=False, blank=False, default='')
  cuisines = models.ManyToManyField(RestaurantCuisine, related_name='restaurant')
  address = models.OneToOneField(Address, on_delete=models.CASCADE, blank=False, null=False)
  coordinates = models.OneToOneField(Coordinates, on_delete=models.CASCADE, null=False)

  updated = models.DateTimeField(auto_now=True)
  created = models.DateTimeField(auto_now_add=True)
```

Rys. 8 - Kod tworzący tabelę restaurant w języku Python.

Źródło: opracowanie własne

Django models umożliwia podstawową weryfikację danych na poziomie implementacji struktury bazy danych i ma inny sposób implementacji powiązań między tabelami (modelami): Django automatycznie tworzy pola id dla każdego modelu, pozwala uzyskać odniesienie do instancji obiektu modelu (lub obiektów) powiązanego z innym modelem za pośrednictwem specjalnych klas pól, po prostu odwołując się do nazwy pola lub wartości atrybutu related_name, jeśli łącze jest inicjowane z żądanego modelu, i pozwala uniknąć ręcznego tworzenia tabel, które działają jako łącza wiele do wielu (w tym przykładzie taka tabela musiałaby zostać utworzona, aby połączyć się z tabelą cuisines).

2.6 Dane klienta: rejestracja, sprawdzanie poprawności

Django users

W implementacji schematu bazy danych w projekcie nie ma tabeli użytkowników, zamiast tego została ona zastąpiona tabelą profili połączoną jeden-do-jednego z wbudowaną w Django tabelą obsługi użytkowników.

System uwierzytelniania użytkowników Django obsługuje konta użytkowników, grupy, uprawnienia i sesje użytkowników oparte na plikach cookie.

System uwierzytelniania i autoryzacji pozwala weryfikować dane uwierzytelniające użytkownika i określać, jakie działania może on wykonywać. Ten framework zawiera wbudowane modele dla użytkowników i grup, sam system uprawnień, który określa, czy użytkownik może wykonać zadanie, w jakiej formie i wyświetlać dla autoryzowanych użytkowników oraz uzyskać dostęp do ograniczonej zawartości.

System uwierzytelniania w Django ma na celu być bardzo ogólny i nie zapewnia niektórych funkcji powszechnie spotykanych w systemach uwierzytelniania webowego. Rozwiązania niektórych z tych powszechnych problemów zostały zaimplementowane w pakietach firm trzecich (*Django,* b.r.):

1. Sprawdzanie siły hasła
2. Ograniczanie prób logowania
3. Uwierzytelnianie względem stron trzecich (na przykład OAuth)
4. Uprawnienia na poziomie obiektu

Weryfikacja danych

Walidacja po stronie klienta to wstępna weryfikacja danych wejściowych, która znacznie poprawia wrażenia użytkownika; wykrycie nieprawidłowych danych po stronie klienta pozwala użytkownikowi natychmiast je poprawić bez angażowania serwera w ten proces. Walidacja danych po stronie użytkownika może zapewnić lepsze wrażenia użytkownika, dając mu natychmiastową informację zwrotną, jeśli wprowadzi nieprawidłowe lub niekompletne dane. Jednak poleganie wyłącznie na walidacji frontendu nie jest bezpieczne, ponieważ złośliwi użytkownicy mogą łatwo ominąć lub manipulować sprawdzaniem poprawności. Użytkownik może po prostu wyłączyć JavaScript w przeglądarce, w takim przypadku logika walidacji nie zostanie uruchomiona. Użytkownicy mogą też użyć specjalnych narzędzi do modyfikacji kodu źródłowego aplikacji po stronie klienta: źródło strony i podstawowy kod JavaScript można łatwo zmanipulować, aby zrobić wszystko, czego może chcieć złośliwy użytkownik. Wreszcie, rzeczywiste żądanie przesłania danych można przechwycić, przechwycić, zmienić i otworzyć. Podczas odtwarzania złośliwy użytkownik może zmienić treść żądania, aby przesłać dowolne dane.

Jeśli walidacja odbywa się tylko na serwerze, dane użytkownika z zapytania zostaną wysłane do serwera w oryginalnej formie bez żadnych kontroli ze strony klienta, a walidacja i bezpieczeństwo żądania odbywa się na serwerze. W takim przypadku proces uzyskiwania danych użytkownika byłby chroniony, ale każda taka kontrola wymagałaby wysłania

żądania do serwera i otrzymania odpowiedzi, co jest czasochłonne i pogarsza wrażenia użytkownika podczas pracy ze stroną.

Rozwiązaniem tego problemu byłaby wstępna walidacja danych po stronie klienta i dodatkowa walidacja po stronie serwera, gdy użytkownik wysyła zapytanie. Na przykład, podczas tworzenia recenzji, głównym sprawdzeniem byłaby długość recenzji i zakres ocen:

```
<textarea class="new-review-textarea text-18"  maxlength="1000" name="body" required></textarea>
<input type="range" min="1" max="10" value="1" name="price" class="new-review-slider"
oninput="update_span(this.value, 'text-c')">
```

Rys. 9 - Kod tworzący pola input i textarea z ograniczonym zakresem wartości.

Źródło: opracowanie własne

W tym przypadku sprawdzany jest zakres ocen od 1 do 10, obecność opinii tekstowych i maksymalna długość opinii wynosząca 1000 znaków. Po stronie serwera jest więcej kontroli: podobnie jak po stronie użytkownika, musisz sprawdzić zakres ocen, dostępność recenzji tekstowej i jej długość, ale także to, czy użytkownik zalogował się na swoje konto, czy zostawia recenzję dla własnej restauracji i czy już zostawił recenzję dla tej restauracji:

```
def user_can_review_restaurant(user: User, restaurant: Restaurant):
    if restaurant.company_profile.profile.user == user:
        raise PermissionError(_('You cannot review your own restaurant'))
    if Review.objects.filter(profile=user.profile, restaurant=restaurant).exists():
        raise PermissionError(_('You have already reviewed this restaurant'))
```

Rys. 10 - Sprawdzanie dostępu do danych po stronie serwera.

Źródło: opracowanie własne

W normalnym przypadku użytkownik prawdopodobnie nie napotka takich błędów (zakładając, że wysłany interfejs jest dostosowany do statusu użytkownika), ale zapewni to dodatkową ochronę przed atakami.

Istnieją jednak również kontrole, których nie można sprawdzić bez odpytywania bazy danych, takie jak to, czy użytkownik o danej nazwie istnieje w bazie danych podczas tworzenia konta (Field Validations at Frontend or Backend? Which Is Better?, b.r.; Nahum, 2021).

AJAX, asynchroniczny transfer danych

AJAX to technologia wymiany danych z serwerem bez odświeżania strony. AJAX to skrót od Asynchronous JavaScript and XML.

Jak sama nazwa wskazuje, wymiana danych z serwerem jest asynchroniczna. Oznacza to, że nie wpływa ona na interakcję użytkownika ze stroną. Po wysłaniu żądania oznacza to, że użytkownik nadal wchodzi w interakcję ze stroną. Po otrzymaniu odpowiedzi z serwera uruchamiana jest funkcja zwrotna (callback), która reaguje na wynik, np. wyświetlając powiadomienie lub aktualizując liczbę polubień.

JQuery zapewnia kompletny zestaw opcji do stosowania AJAX. Za pomocą tych metod można zażądać kodu tekstowego, HTML, XML lub JSON z serwera, wysyłając żądanie HTTP Get lub Post. Wynikowe dane mogą być wyprowadzane w dowolnym elemencie HTML.

W tym projekcie zapytania AJAX zostały użyte w wielu komponentach, jednym z zastosowań jest powyższy przykład sprawdzania, czy nazwa użytkownika istnieje w bazie danych. Za każdym razem, gdy nazwa jest zmieniana, żądanie jest wysyłane do serwera, przekazując nazwę użytkownika i czekając na odpowiedź z serwera, czy użytkownik o tej nazwie może zostać utworzony. To podejście ma również swoje wady: zwiększa obciążenie serwera, ponieważ za każdym razem, gdy nazwa użytkownika jest zmieniana w ciągu znaków, nowe żądanie jest wysyłane do serwera, co generuje dużą liczbę żądań. Problem ten można częściowo rozwiązać, na przykład wykonując wstępne sprawdzenie długości nazwy po stronie klienta przed żądaniem, tak aby żądania ze zbyt krótkimi nazwami nie były wysyłane, ponieważ taka nazwa ze 100% prawdopodobieństwem spowoduje odpowiedź "Zbyt krótka nazwa użytkownika".

Poniżej znajduje się przykład użycia AJAX do walidacji nazwy użytkownika. Najpierw następuje podstawowa walidacja długości nazwy użytkownika po stronie klienta. Jeśli nazwa użytkownika nie przejdzie walidacji, żądanie walidacji nie zostanie wysłane do serwera, a użytkownik otrzyma komunikat, że nazwa użytkownika jest zbyt krótka. Jeśli nazwa jest wystarczająco długa, jest wysyłana do serwera w celu pełnego sprawdzenia poprawności.

```
$('#login-name').keyup(function () {
    const login = $('#login-name');
    const usernameStatus = $('#username-status');
    if (login.val().length < 3) {
        usernameStatus.removeAttr("hidden");
        usernameStatus.text("This username is too short");
        return ;
    }
    $.ajax({
        data: $(this).serialize(),
        url: "{% url 'validate_username' %}",
        success: function (response) {
            if (response.is_taken === true) {
                login.after()
                usernameStatus.text(response.resp_message);
            } else {
                usernameStatus.attr("hidden", "");
            }
        },
        error: function (response) {
            console.log(response.responseJSON.errors)
        }
    });
    return false;
});
```

Rys. 11 – Przykład użycia AJAX do sprawdzania poprawności nazwy użytkownika

Źródło: opracowanie własne

Opcjonalnie można zaimplementować mechanizm, który wysyła żądanie sprawdzenia nazwy dopiero po upływie, na przykład, ⅓ sekundy od ostatniej zmiany nazwy użytkownika, co, choć wydłużyłoby czas oczekiwania, pozwoliłoby na sprawdzenie nazwy użytkownika prawie zawsze po jej pełnym wprowadzeniu.

Ponadto żądania AJAX mogą być przydatne na stronie głównej i w żądaniach użytkowników, służąc do odświeżania treści na podstawie filtrów, wyszukiwania, a także dynamicznego ładowania treści, eliminując potrzebę przeskakiwania między stronami w celu załadowania pozostałych elementów.

2.7 Automatyczne tłumaczenie strony na inne języki

W przykładowym kodzie, sprawdzającym możliwość użytkownika do utworzenia strony restauracji wyświetlane błędy są najpierw przekazywane do funkcji _(), tą funkcją jest alias metody Django.utils.translation.gettext, która wykonuje funkcje tłumaczenia strony.

Żeby umożliwić tłumaczenie projektu Django, należy dodać hooki do kodu Pythona i szablonów. Te hooki nazywane są łańcuchami tłumaczeń. Dosłownie oznaczają one: "Ten tekst powinien zostać przetłumaczony na język użytkownika końcowego, jeśli tłumaczenie tego tekstu jest dostępne w tym języku". Najpierw wszystkie wyrażenia, które mają zostać automatycznie przetłumaczone, muszą zostać umieszczone w funkcji gettext. Następnie Django dostarcza narzędzia do wyodrębniania ciągów tłumaczeń do pliku wiadomości. Plik ten jest wygodnym sposobem dla tłumaczy na dostarczenie równoważnych ciągów tłumaczeń do języka docelowego. Gdy tłumacze wypełnią plik wiadomości, musi on zostać skompilowany. W tym procesie używany jest zestaw narzędzi GNU gettext (Django documentation, bd.).

```
<script src="{% static 'js/new_review.js' %}"></script>
<div class="main">
    <h2 class="text-24">{% trans "New review on" %} "{{ restaurant.name }}"</h2>
    <form method="post" action="">
        {% csrf_token %}
        <div class="new-review-flex">
            <div class="new-review-left">
                <h2 class="text-24 new-review-text">{% trans "Food" %}</h2>
                <h2 class="text-24 new-review-text">{% trans "Staff" %}</h2>
                <h2 class="text-24 new-review-text">{% trans "Price" %}</h2>
            </div>
            <div class="new-review-right">
```

Rys. 12 – Przykład użycia hooków w szablonie html

Źródło: opracowanie własne

Gdy to zostanie zrobione, Django przejmuje tłumaczenie aplikacji internetowych w locie na każdy dostępny język zgodnie z preferencjami językowymi użytkowników.

Plik wygenerowany przez narzędzie składa się z komunikatów, przykład komunikatu pokazano poniżej:

```
#: .\otzovik_app\templates\otzovik_app\login_page.html:15
#: .\otzovik_app\templates\otzovik_app\new_restaurant.html:56
#: .\otzovik_app\templates\otzovik_app\new_review.html:41
#: .\otzovik_app\templates\otzovik app\registration_page.html:44
#: .\otzovik_app\templates\otzovik_app\user_profile.html:20
msgid "Submit"
msgstr "Potwierdź"
```

Rys. 13 - Fragment pliku przechowującego tłumaczenia.

Źródło: opracowanie własne

Tutaj po znakach # znajdują się ścieżki, w których dana wiadomość została użyta, co pozwala na lepsze zrozumienie kontekstu tłumaczenia. Msgid to unikalny identyfikator komunikatu, którego tłumaczenie można wywołać za pomocą funkcji gettext z kodu Pythona lub bezpośrednio w języku szablonów Django za pomocą funkcji {* trans 'msgid' *}, msgstr to tekst tłumaczenia.

Django i18n zapewnia łatwe w użyciu i poręczne narzędzie do tłumaczenia elementów interfejsu, ale jego funkcjonalność nie pozwala na uwzględnienie różnych innych obiektów na ekranie i wykonywanie funkcji podczas tłumaczenia, ale dla tego projektu jest to wystarczające (Dalibor, 2023).

2.8 Komunikacja z użytkownikiem, obsługiwanie błędów

W tym projekcie szablony Django zostały wybrane jako format do tworzenia interfejsu użytkownika. Szablony Django to pliki, które definiują, jak powinien wyglądać kod HTML stron aplikacji internetowej. Używają one specjalnego języka szablonów, który pozwala wstawiać dane z kontekstu aplikacji, używać pętli, instrukcji warunkowych i innych konstrukcji.

Język szablonów Django ma kilka konstrukcji, które pozwalają wstawiać dane z kontekstu aplikacji, używać pętli, operatorów warunkowych i innych konstrukcji. Wspiera także dziedziczenie szablonów, co zmniejsza duplikację kodu i ponowne wykorzystanie istniejących szablonów. Podstawowe konstrukcje języka szablonów Django obejmują następujące elementy:

1. {{ variable }} - Ta konstrukcja jest używana do wyświetlania wartości zmiennej na stronie. Wewnątrz podwójnych nawiasów klamrowych można określić nazwę zmiennej z kontekstu aplikacji.
2. {% tag %} - ta konstrukcja służy do wykonywania różnych działań na stronie, takich jak kontrolowanie przepływu wykonywania, ustawianie wartości zmiennych itp. Wewnątrz nawiasów klamrowych można określić nazwę tagu i jego argumenty.
3. {% for element in list %} ... {% endfor %} - Konstrukcja ta służy do wykonywania pętli. Pozwala ona na wykonanie pętli przez wszystkie elementy na liście i wykonanie jakiejś akcji na każdym z nich.

4. {% if condition %} ... {% endif %} - Konstrukcja ta służy do wykonywania instrukcji warunkowej. Pozwala na wykonanie pewnych akcji tylko wtedy, gdy podany warunek jest prawdziwy.
5. {% extends "base_template.html" %} - Konstrukcja ta służy do dziedziczenia z innego szablonu. Pozwala utworzyć nowy szablon, który dziedziczy elementy i style z szablonu bazowego.
6. {% block nazwa_bloku %} ... {% endblock %} - Konstrukcja ta służy do tworzenia bloku, który może zostać zastąpiony w dziedziczonym szablonie. Pozwala to na tworzenie bardziej elastycznych szablonów, które można dostosować do konkretnych potrzeb.
7. {% url 'submit_name' arguments_url %} - Ta konstrukcja służy do tworzenia linków do innych stron w aplikacji. Pozwala na automatyczne wygenerowanie adresu URL dla danego widoku i przekazanie niezbędnych argumentów.
8. {% include 'template.html' %} - Ta konstrukcja służy do włączania treści z innego szablonu do bieżącego szablonu. Pozwala to na używanie wspólnych elementów interfejsu użytkownika na wielu stronach aplikacji bez konieczności duplikowania kodu.
9. {{ value|filter }} - Ta konstrukcja służy do zastosowania filtra do wartości zmiennej wyświetlanej na stronie. Filtry umożliwiają zmianę formatu, rozmiaru i typu wartości wyjściowej.
10. {% csrf_token %} - Ta konstrukcja służy do wstawiania tokena CSRF do formularza. Pozwala to chronić aplikację przed atakami typu cross-site request forgery.
11. {% with variable_name=value %} ... {% endwith %} - Ta konstrukcja służy do tworzenia tymczasowych zmiennych w szablonie. Pozwala ona na zapisanie wartości zmiennej podczas wykonywania bloku, po czym zmienna jest usuwana.
12. {% comment "Comment" %} - Ta konstrukcja służy do dodawania komentarzy do szablonu. Pozwala na pozostawienie notatek dla innych programistów bez wpływu na układ strony.

Zmienne w szablonie Django są używane do przekazywania danych z widoku do szablonu w celu wyświetlenia na stronie. Aby użyć zmiennych w szablonie Django, należy zdefiniować kontekst w widoku, który zawiera słownik z parami klucz-wartość, gdzie klucze są nazwami zmiennych, a wartości są danymi, które mają być mapowane na stronę.

W Django szablony mogą być używane zarówno na poziomie projektu, jak i aplikacji. Szablony na poziomie projektu są przechowywane w katalogu templates w

głównym folderze projektu, a szablony na poziomie aplikacji są przechowywane w katalogu templates wewnątrz folderu aplikacji.

W projekcie każdy szablon jest oddzielną stroną, z wyjątkiem tych, które są używane na wszystkich lub kilku stronach jednocześnie (element wyszukiwania, nagłówek, szablon główny). Szablon główny służy jako podstawowy układ dla wszystkich stron i zawiera globalne pliki stylów i skryptów, a także pasek nawigacyjny i blok do wyświetlania błędów na ekranie.

Views.py

Funkcje w pliku views.py służą do wysyłania szablonów jako stron html do użytkownika (utrzymywanie komunikacji użytkownika z witryną, utrzymywanie zapytań html). W standardowym szablonie projektu plik ten zawiera większość logiki strony, ale w tym projekcie ma zamiast tego jedną funkcję: obsługę błędów.

```
def edit_profile(request, pk):
    profile = get_profile(pk)
    context = {'type': 'edit', 'profile': profile}
    try:
        user_can_edit_profile(request.user, profile)
    except PermissionError as msg:
        messages.error(request, msg)
        return redirect('home')

    if request.method == 'POST':
        try:
            save_profile(request, pk)
            return redirect('user_profile', pk)
        except ValueError as msg:
            messages.error(request, msg)
            return render(request, 'otzovik_app/registration_page.html', context)
        except PermissionError as msg:
            messages.error(request, msg)
            return redirect('home')

    return render(request, 'otzovik_app/registration_page.html', context)
```

Rys. 14 - metoda edit_profile.

Źródło: opracowanie własne

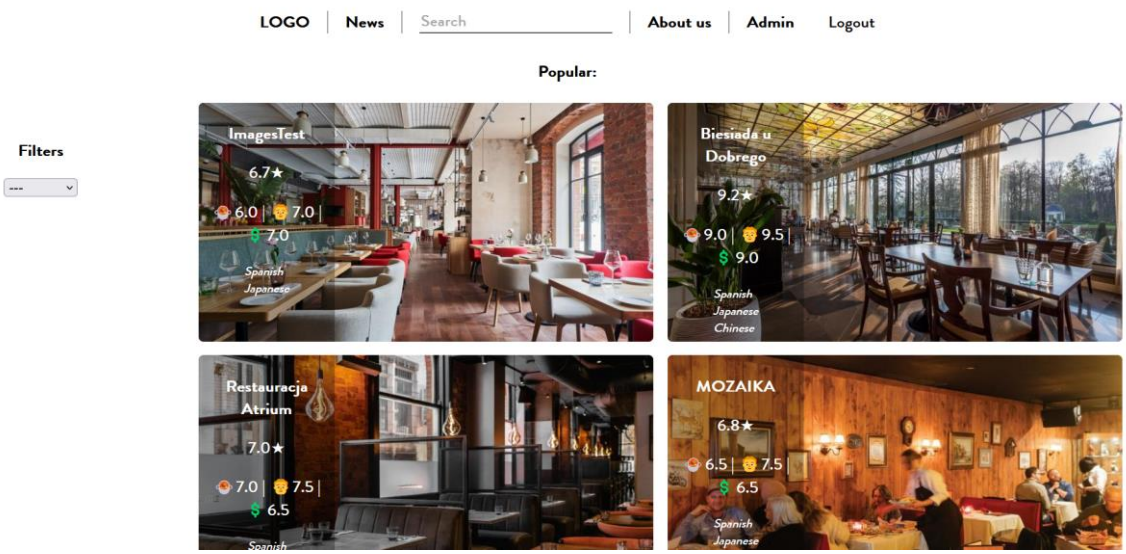
W tym przykładzie funkcja edit_profile jest odpowiedzialna za interakcję z użytkownikiem w zakresie zmiany jego danych profilowych. Gdy użytkownik wchodzi na stronę, funkcja sprawdza, czy użytkownik może modyfikować dane profilu (użytkownik może próbować wejść na stronę danych edit_profile dowolnego użytkownika). Jeśli pojawi się żądanie zmiany danych, istnieją 3 możliwe wyniki tego procesu:

1. Pomyślna zmiana danych: w tym przypadku dane zostają zmienione i użytkownik przechodzi do strony użytkownika.
2. ValueError. Ten typ błędu może wystąpić z powodu wprowadzenia przez użytkownika nieprawidłowych danych, np. zajętego adresu e-mail. W takim przypadku użytkownik powinien zostać przekierowany z powrotem na stronę modyfikacji danych, a błąd powinien zostać wyświetlony na ekranie.
3. PermissionError. Ten błąd może wystąpić, jeśli system odmówi modyfikacji danych, na przykład z powodu próby modyfikacji danych profilu innej osoby. W takim przypadku należy wyświetlić błąd i przekierować na stronę główną.

Podsumowując, takie przypisanie pliku views.py pozwala na łatwą i wydajną obsługę błędów oraz przyczynia się do lepszej czytelności kodu w porównaniu do umieszczenia całej logiki biznesowej w views.py.

Wygląd strony

Projekt strony internetowej opiera się na kartach restauracji; karty są głównym przedmiotem zainteresowania użytkownika, dlatego położono na nie szczególny nacisk:



Rys. 15 - strona główna.

Źródło: opracowanie własne

Użytkownik otrzymuje dane dotyczące ocen, nazwy, kuchni i ocen według kategorii, aby zdecydować czy kliknąć na kartę. Jeśli te dane są niewystarczające, po najechaniu na kartę pojawia się adres i opis miejsca. Na wszystkich stronach z linkami do restauracji, restauracje są prezentowane jako “karty”.

Rys. 16 - Karta restauracji po najechaniu myszą.

Źródło: opracowanie własne

Ogólny projekt strony został wybrany jako minimalistyczny, pozostawiając skupienie się na kartach restauracji, ponieważ jest to główna treść na stronie. Elementy interaktywne, takie jak pola wprowadzania i wyboru, przyciski i linki zostały również wizualnie wyróżnione.

2.9 Mechanizmy wyboru rekomendacji i wydajność

Obecnie systemy dopasowywania rekomendacji stały się jedną z najważniejszych części stron internetowych o różnej tematyce. Przykładem zastosowania takich systemów jest możliwość zatrzymania użytkownika na stronie poprzez dostarczenie dodatkowych informacji adekwatnych do jego preferencji i zawartości aktualnej strony internetowej. Najskuteczniejszy efekt uzyskuje się, gdy dla systemu rekomendacji dostępne są dane o zachowaniu użytkownika na stronie (statystyki odwiedzanych stron, czas przebywania, oceny i pozostawione komentarze).

Wybór rekomendacji składa się z 2 głównych części: filtrowania treści i sortowania treści. Filtrowanie treści polega na zmniejszeniu ilości treści wyświetlanych użytkownikowi, usuwając treści, które prawdopodobnie go nie zainteresują. Może to być wykonywane ręcznie przez użytkownika lub automatycznie. Na przykład na tej stronie należy określić swoje miasto rodzinne podczas tworzenia użytkownika. Filtr jest następnie automatycznie stosowany do wyboru restauracji, który usuwa wszystkie restauracje z wyjątkiem miasta, w którym znajduje się użytkownik. Utworzono również filtry, które umożliwiają ręczne wybieranie pożądanych cech restauracji, takich jak kuchnia czy cennik.

Automatyczne sortowanie restauracji

Automatyczne sortowanie treści na podstawie aktywności użytkownika odnosi się do procesu organizowania i ustalania priorytetów treści dla użytkowników na podstawie ich wcześniejszych zachowań, preferencji i interakcji na platformie. Wiąże się to z wykorzystaniem algorytmów i technik uczenia maszynowego do analizy danych dotyczących aktywności użytkownika i dostarczania spersonalizowanych rekomendacji treści.

Celem automatycznego sortowania treści jest poprawa doświadczenia użytkownika poprzez prezentowanie odpowiednich i angażujących treści osobom na podstawie ich zainteresowań i zachowań. Analizując aktywność użytkowników, platformy mogą zrozumieć preferencje użytkowników, takie jak rodzaj treści, z którymi się angażują, tematy, którymi są zainteresowani, oraz ich wzorce interakcji (polubienia, komentarze, udostępnienia itp.).

Najważniejsze cechy algorytmu sortowania treści:

1. Prędkość i szybkość reakcji: Jednym z kluczowych czynników jest to, jak szybko algorytm może przetwarzać dane o aktywności użytkownika i generować spersonalizowane rekomendacje treści. System powinien być w stanie analizować duże ilości danych w czasie rzeczywistym lub zbliżonym do rzeczywistego, aby dostarczać użytkownikom sugestie na czas. Szybszy czas przetwarzania prowadzi do bardziej płynnego doświadczenia użytkownika, zapewniając szybkie dostarczanie rekomendacji treści.

2. Skalowalność: Wydajność systemu powinna być skalowalna, aby poradzić sobie z rosnącą aktywnością użytkowników i ilością treści. Wraz ze wzrostem liczby użytkowników i ilości treści, algorytm powinien być w stanie wydajnie przetwarzać

i sortować dane bez znaczących opóźnień lub spowolnień. Wymaga to infrastruktury obliczeniowej, która może obsługiwać duże obciążenia i skalować się poziomo, aby sprostać zwiększonym wymaganiom.

3. Dokładność i precyzja: Wydajność algorytmu jest mierzona tym, jak dokładnie jest on w stanie przewidzieć preferencje użytkownika i dostarczyć odpowiednie rekomendacje treści.

Złożoność czasowa, Notacja dużego O

Kiedy mówi się o złożoności czasowej, mówi się o liczbie operacji. W celu ułatwienia obliczeń, różnica prędkości między operacjami jest zwykle pomijana. Dlatego też, pomimo faktu, że dzielenie liczb zmiennoprzecinkowych wymaga więcej pracy od procesora niż dodawanie liczb całkowitych, w teorii algorytmów obie operacje są uważane za równe pod względem złożoności.

W typowym użyciu notacja O jest asymptotyczna, to znaczy odnosi się do bardzo dużych x. W tym ustawieniu wkład terminów, które rosną "najszybciej", ostatecznie sprawi, że inne staną się nieistotne. W rezultacie można zastosować następujące zasady upraszczania:

1. Jeśli f(x) jest sumą kilku wyrażeń, jeśli istnieje jedno o największym tempie wzrostu, można je zachować, a wszystkie inne pominąć.

2. Jeśli f(x) jest iloczynem kilku czynników, można pominąć wszelkie stałe (czynniki w iloczynie, które nie zależą od x) (Wikipedia contributors, 2023b).

Każdy algorytm ma najgorszy, średni i najlepszy scenariusz, w zależności od tego, jak dobrze wybrano dane wejściowe. Są one często nazywane przypadkami. Najgorszy przypadek to taki, w którym dane wejściowe wymagają maksymalnej ilości czasu i pamięci. Najlepszy przypadek to dokładne przeciwieństwo najgorszego przypadku, czyli najlepsze dane wejściowe. Prawidłowo posortowana tablica, z którą algorytm sortujący nie musi nic robić. W przypadku wyszukiwania - gdy algorytm znajdzie właściwy element za pierwszym razem. Średni przypadek znajduje się między najlepszym a najgorszym przypadkiem. Często pasuje do najgorszego przypadku i zawsze jest gorszy niż najlepszy przypadek, jeśli najlepszy przypadek nie pasuje do najgorszego przypadku (GeeksforGeeks, 2023).

Techniki uczenia maszynowego nie zostały wykorzystane w tym projekcie ze względu na brak użytkowników do analizy, zamiast tego stworzono algorytm matematyczny

do wyboru rekomendacji. Głównym zadaniem algorytmu w tym projekcie jest znalezienie odpowiednich restauracji na podstawie kliknięć użytkownika i ocen innych lokali. Użytkownik powinien najpierw zobaczyć restauracje o najbardziej podobnych cechach do tych, które lubi i które mają dobre recenzje.

Aby porównać restauracje, wprowadzamy system abstrakcyjnych "punktów", które oceniają szansę, że użytkownikowi spodoba się dany lokal (więcej punktów - większa szansa). Do określenia liczby punktów brane są pod uwagę następujące dane: Liczba ocen użytkowników dla każdej kategorii cenowej, kategorii typu lokalu i kuchni, punktacja i popularność lokalu. Dla każdego użytkownika obliczana jest atrakcyjność każdego z atrybutów restauracji w kategoriach:

$$a = \sum_{x=0}^{n}(c_p * (r_x - b + c_r)), \text{ gdzie} \quad (1)$$

a - atrakcyjność kategorii

n - liczba recenzji

r_x - średnia ocena restauracji z recenzji danego użytkownika

c_p - czy restauracja z recenzji użytkownika zawiera kategorię (0 lub 1)

b - średnia ocena restauracji

c_r - kompensacja oceny (jest potrzebna, aby zrekompensować nieco niższe oceny niż średnia, ponieważ jeśli ocena jest nieco niższa niż średnia, uważa się, że użytkownik wykazał zainteresowanie miejscem)

Następnie można obliczyć liczbę punktów dla każdej restauracji:

$$s = \sum_{a=0}^{n_a}(c_a * s_a) + (r * r_c), \text{ gdzie} \quad (2)$$

n_a - liczba kategorii

c_a - czy kategoria należy do kategorii restauracji (0 lub 1)

s_a - liczba punktów dla kategorii (rezultat z wyrażenia 1)

r - średnia ocen restauracji

r_c - stała waga oceny restauracji

Dla wzoru 1, obliczającego atrakcyjność każdej kategorii *k* w *n* recenzjach i dynamicznie obliczającego średnią ocen restauracji *r*, złożoność algorytmu wynosi O(n(k+r)), ponieważ dla każdej recenzji należy obliczyć atrakcyjność kategorii dla wszystkich kategorii restauracji, przy czym wzór ten zawiera średnią ocen restauracji, która nie musi być obliczana dla każdej kategorii osobno. Skutkuje to kwadratowym wzrostem złożoności i nie da się w tym przypadku zoptymalizować samego algorytmu, ale możliwe jest przechowywanie pewnych danych, takich jak średnia ocen restauracji, oraz przechowywanie atrakcyjności każdej kategorii dla każdego użytkownika, co prawda będzie wymagało więcej pamięci i nie zmieni tempa wzrostu złożoności algorytmu, ale pozwoli na uruchamianie algorytmu nie podczas recenzowania restauracji, ale podczas tworzenia recenzji, co występuje rzadziej, a także zmniejszy bezwzględną złożoność algorytmu do O(nk).

W przypadku wzoru 2 nie ma sensu przechowywać wyników obliczeń w bazie danych, ponieważ wymagałoby to aktualizacji danych dla wszystkich użytkowników podczas tworzenia każdej nowej recenzji i ogromnych zasobów pamięci, ale przy użyciu przechowywanych danych ze wzoru 1, aby obliczyć ogólną atrakcyjność jednej restauracji, złożoność algorytmu będzie wynosić tylko O(k), ponieważ dla każdej kategorii *k* tej restauracji pobiera się już obliczone dane dotyczące atrakcyjności kategorii i średnich ocen z bazy danych i nie oblicza się ich na bieżąco.

2.10 Moderacja manualna i automatyczna

Moderowanie treści odnosi się do procesu monitorowania i regulowania treści generowanych przez użytkowników na różnych platformach internetowych, takich jak portale społecznościowe, fora lub społeczności internetowe. Wiąże się to z przeglądaniem i oceną treści przesyłanych przez użytkowników w celu zapewnienia, że są one zgodne z wytycznymi społeczności platformy, warunkami świadczenia usług i wymogami prawnymi.

Głównym celem moderacji treści jest utrzymanie bezpiecznego i pełnego szacunku środowiska online poprzez usuwanie lub filtrowanie treści, które są nieodpowiednie, obraźliwe, szkodliwe lub naruszają zasady platformy. Może to obejmować usuwanie treści zawierających mowę nienawiści, nękanie, groźby, materiały wulgarne lub graficzne, spam lub jakąkolwiek formę nielegalnych lub szkodliwych treści.

Moderacja treści może być wykonywana zarówno przez ludzkich moderatorów, jak i zautomatyzowane systemy, często wykorzystujące kombinację obu. Ludzcy moderatorzy przeglądają i podejmują decyzje dotyczące treści w oparciu o wcześniej ustalone wytyczne, podczas gdy zautomatyzowane systemy wykorzystują algorytmy i sztuczną inteligencję do identyfikowania potencjalnie problematycznych treści.

Moderacja manualna

Ręczna moderacja polega na ręcznym sprawdzaniu użytkowników i ich treści pod kątem reguł platformy. Dla pracowników zajmujących się moderacją treści w moim projekcie wystarczy mieć prosty profil użytkownika i ręczny wpis w arkuszu moderacji. W tym celu utworzono osobną stronę z 3 kategoriami: moderacja restauracji, moderacja recenzji i moderacja profilu firmy.

Zarówno w przypadku restauracji, jak i recenzji, użytkownicy mają możliwość ustawienia tak zwanych "flag" dla moderatorów (w przypadku recenzji flagi te to negatywna ocena recenzji, w przypadku restauracji jest to flaga do zgłaszania naruszeń). Jeśli istnieje co najmniej jedna flaga dla restauracji lub negatywna ocena recenzji, recenzja jest oznaczona do sprawdzenia, a po sprawdzeniu jest oznaczona jako sprawdzona, po czym nie może być oznaczona do sprawdzenia aż do następnej zmiany. Moderacja profili firmowych jest prostsza: profil firmowy jest moderowany raz po jego utworzeniu i nie może być później poddawany drugiemu procesowi moderacji.

Moderacja automatyczna

Automatyczna moderacja treści nie może zagwarantować pełnego bezpieczeństwa treści, ale znacznie zmniejszy ilość niedopuszczalnych treści. Polega ona na sprawdzaniu treści pod kątem zakazanych słów i fraz, można to zrobić za pomocą wyrażeń regularnych lub po prostu wyszukując znaki w ciągu znaków. Stworzenie takiego systemu wymaga wdrożenia dodatkowych metod weryfikacji różnych danych wprowadzanych przez użytkownika i wykorzystania ich podczas tworzenia i modyfikowania materiałów.

Rozdział 3
Testowanie, wdrażanie

W dziedzinie tworzenia aplikacji dwa krytyczne elementy wyróżniają się jako niezbędne do zapewnienia jakości, niezawodności i wydajności oprogramowania: testowanie i automatyczne wdrażanie. Testowanie służy jako kompleksowy proces oceny, który identyfikuje i naprawia błędy, weryfikuje wymagania i zapewnia, że aplikacja spełnia pożądane oczekiwania. Z kolei automatyczne wdrażanie usprawnia proces wdrażania, automatyzując wprowadzanie zmian w oprogramowaniu do środowisk produkcyjnych lub testowych. Inwestując w dokładne testy i stosując praktyki automatycznego wdrażania, programiści mogą dostarczać solidne i niezawodne rozwiązania programowe, które spełniają oczekiwania użytkowników. Te podstawowe elementy rozwoju aplikacji sprzyjają zadowoleniu użytkowników, ograniczają ryzyko, zwiększają wydajność i promują ciągłe doskonalenie.

3.1 Testowanie

Testowanie backendu to rodzaj testowania, które sprawdza warstwę aplikacji i bazy danych w architekturze 3-warstwowej. W przypadku złożonych aplikacji, takich jak ERP, testowanie backendu obejmuje sprawdzenie logiki biznesowej na poziomie aplikacji. W przypadku prostszych aplikacji, testowanie backendu sprawdza część serwerową lub bazę danych. Oznacza to, że dane wprowadzone do interfejsu zostaną sprawdzone w bazie danych. Bazy danych są sprawdzane pod kątem właściwości ACID, operacji CRUD, ich schematu, zgodności z regułami biznesowymi. Bazy danych są również sprawdzane pod kątem bezpieczeństwa i wydajności. Sprawdzana jest integralność danych, poprawność danych, testowane są funkcje, procedury i wyzwalacze. W testach wewnętrznych nie ma potrzeby korzystania z interfejsu graficznego. Można bezpośrednio przekazywać dane za pomocą przeglądarki z parametrami wymaganymi dla funkcji, aby uzyskać odpowiedź w pewnym domyślnym formacie. Na przykład XML lub JSON. Można również połączyć się bezpośrednio z bazą danych i testować dane za pomocą zapytań SQL (Deshpande, 2023).

Rodzaje testów funkcjonalnych

Testy modułowe. Testy modułowe działają na bardzo niskim poziomie, blisko kodu źródłowego aplikacji. Polegają one na testowaniu poszczególnych metod i funkcji klas, komponentów lub modułów wykorzystywanych w oprogramowaniu. Testy modułowe nie są zazwyczaj kosztowne w automatyzacji i mogą być wykonywane bardzo szybko przez serwer ciągłej integracji.

Testy integracyjne. Testy integracyjne sprawdzają, czy różne moduły i usługi używane przez aplikację dobrze ze sobą współpracują. Na przykład, można przetestować interakcję z bazą danych lub upewnić się, że mikrousługi współpracują ze sobą zgodnie z przeznaczeniem. Ten rodzaj testowania jest bardziej kosztowny, ponieważ wymaga różnych komponentów aplikacji do uruchomienia testów.

Testy funkcjonalne. Testy funkcjonalne koncentrują się na wymaganiach biznesowych aplikacji. Sprawdzają one jedynie wynik pewnej akcji i nie sprawdzają pośrednich stanów systemu podczas wykonywania tej akcji.

Czasami koncepcje testów integracyjnych i funkcjonalnych są mylone, ponieważ oba wymagają interakcji kilku komponentów. Różnica polega na tym, że test integracyjny musi po prostu upewnić się, że można wysyłać zapytania do bazy danych, podczas gdy test funkcjonalny będzie oczekiwał pobrania określonej wartości z bazy danych zgodnie z wymaganiami produktu (Vijay, 2023).

Narzędzia do testowania aplikacji Django

Django oferuje wiele możliwych narzędzi do testowania aplikacji:

Wbudowany framework testowy Django: Django zapewnia kompleksowy framework testowy jako część swojej podstawowej funkcjonalności. Obejmuje on narzędzia do tworzenia przypadków testowych, zarządzania testami, symulowania żądań HTTP, potwierdzania oczekiwanego zachowania i uruchamiania testów. Możesz wykorzystać framework testowy Django do pisania testów jednostkowych, integracyjnych i funkcjonalnych.

Przykładem takiego testu może być test sprawdzający czy możliwe jest utworzenie profilu użytkownika. Ponieważ funkcja używana do tworzenia użytkownika jest oddzielna i pobiera zmienną zapytania, najlepszym sposobem na przetestowanie tej funkcji jest symulacja wysłania zapytania typu post z wymaganymi danymi, a następnie sprawdzenie

czy użytkownik znajduje się w bazie danych za pomocą metody *exists*. W tym przypadku oczekiwanym kodem statusu zapytania będzie 302, ponieważ po udanym utworzeniu profilu następuje przekierowanie na stronę główną.

```
class CreateUserTest(TestCase):
    Wym
    def setUp(self) -> None:
        self.client = Client()
        City.objects.create(name="Poznan")

    Wym
    def test_create_user(self):
        response = self.client.post("/register/", {
            "username": "test_user_01",
            "password1": "testpass01",
            "password2": "testpass01",
            "name": "tname01",
            "surname": "tsurname01",
            "email": "test01@mail.com",
            "city": "Poznan"
        })

        self.assertEqual(response.status_code, 302)
        self.assertTrue(Profile.objects.filter(user__username="test_user_01").exists())
```

Rys. 17 - Przykładowy test modułowy z wykorzystaniem Django Test Framework.

Źródło: opracowanie własne

Ważne jest również, że testy powinny sprawdzać nie tylko pomyślne użycie aplikacji, ale także scenariusze, w których występują błędy. Ma to na celu poprawę odporności systemu i umożliwia wykrycie większej liczby potencjalnych błędów. Na przykład w tym przypadku można również przetestować proces tworzenia profilu z nieistniejącym miastem, chociaż nie jest możliwe wprowadzenie nieistniejącego miasta za pomocą formularza, przy użyciu narzędzi do modyfikacji kodu strony opisanych w sekcji "Weryfikacja danych" to może być możliwe i może to spowodować awarię serwera. Kod statusu zapytania powinien w tym przypdku wynosić 200(OK), ponieważ nie występuje przekierowania na stronę główną, a dodatkowo można też sprawdzić, czy danego profilu nie istnieje w bazie danych.

```python
class CreateUserNonExistentCityTest(TestCase):
    Wym
    def setUp(self) -> None:
        self.client = Client()
        City.objects.create(name="Poznan")

    Wym
    def test_create_user(self):
        response = self.client.post("/register/", {
            "username": "test_user_02",
            "password1": "testpass02",
            "password2": "testpass02",
            "name": "tname02",
            "surname": "tsurname02",
            "email": "test02@mail.com",
            "city": "Wroclaw"
        })

        self.assertEqual(response.status_code, 200)
        self.assertFalse(Profile.objects.filter(user__username="test_user_02").exists())
```

Rys. 18 - Test mający na celu sprawdzenie, że profil nie może zostać utworzony, jeśli w bazie danych nie ma podanego miasta.

Źródło: Opracowanie własne

Pytest: Pytest to popularny framework testowy dla języka Python, który zapewnia bardziej zwięzły i ekspresyjny sposób pisania testów w porównaniu do wbudowanego modułu unittest. Oferuje proste, ale potężne podejście do testowania, które koncentruje się na prostocie, czytelności i rozszerzalności. Zapewnia możliwość pisania testów jako funkcji, co oznacza, że znaczna część kodu boilerplate jest wyeliminowana, dzięki czemu kod jest bardziej czytelny i łatwiejszy w utrzymaniu. Pytest zapewnia również funkcjonalność w zakresie wykrywania testów oraz definiowania i używania *fixtures* (Sheth, 2022).

Coverage.py: Coverage.py jest narzędziem innej firmy, które mierzy zakres pokrycia testami w twojej bazie kodu. Pomaga zidentyfikować obszary aplikacji, w których brakuje pokrycia testami. Coverage.py może być zintegrowany z pakietem testów Django, aby generować raporty pokrycia, podkreślając, które części kodu są wykonywane przez testy. To narzędzie pomaga w zapewnieniu kompleksowego pokrycia testami aplikacji Django (Sheth, 2022).

Selenium: Selenium to zaawansowane narzędzie do automatyzacji testów przeglądarkowych. Pozwala symulować interakcje użytkownika i testować funkcjonalność aplikacji Django w rzeczywistym środowisku przeglądarki. Selenium można połączyć z frameworkiem testowym Django lub innymi frameworkami testowymi, takimi jak Pytest, w

celu pisania testów end-to-end (E2E), które zapewniają prawidłowe zachowanie w różnych przeglądarkach (Sheth, 2022).

Testowanie manualne

Testowanie manualne stron internetowych obejmuje proces ręcznego testowania zarówno frontendowych, jak i backendowych komponentów aplikacji internetowej. Wymaga to interwencji człowieka w celu interakcji z aplikacją, symulacji działań użytkownika i sprawdzenia jej zachowania.

Dla frontendu testerzy ręcznie weryfikują wygląd i funkcjonalność interfejsu użytkownika. Sprawdzają układ, responsywność, nawigację i elementy wizualne, takie jak przyciski, formularze i menu, aby upewnić się, że działają poprawnie w różnych przeglądarkach i urządzeniach. Wykonują również testy funkcjonalne, aby sprawdzić, czy komponenty frontendowe, takie jak formularze, przyciski i linki, działają zgodnie z oczekiwaniami. Wprowadzają prawidłowe i nieprawidłowe dane, przesyłają formularze i upewniają się, że uruchamiane są odpowiednie akcje. Sprawdzają również obsługę błędów i komunikaty walidacyjne. Testerzy sprawdzają, czy aplikacja internetowa działa płynnie na różnych przeglądarkach, systemach operacyjnych i urządzeniach. Ręcznie testują zachowanie i wygląd aplikacji, aby zidentyfikować wszelkie problemy z kompatybilnością i zapewnić spójne wrażenia użytkownika.

Dla części backendowej testerzy ręcznie weryfikują backendowe interfejsy API, wysyłając żądania i analizując odpowiedzi. Sprawdzają prawidłową funkcjonalność API, właściwą obsługę danych wejściowych, uwierzytelnianie i autoryzację. Testują również różne scenariusze, takie jak przypadki brzegowe i warunki błędu, aby upewnić się, że otrzymywane są odpowiednie odpowiedzi. Testerzy weryfikują dokładność, kompletność i spójność danych przetwarzanych i przechowywanych przez backend. Ręcznie weryfikują operacje wprowadzania, pobierania i manipulowania danymi, upewniając się, że backend wykonuje dokładne obliczenia, transformacje danych i operacje na bazie danych. Ręcznie testują backend pod kątem potencjalnych luk i słabości bezpieczeństwa. Sprawdzają prawidłowe szyfrowanie danych, mechanizmy uwierzytelniania, ochronę przed typowymi atakami, takimi jak SQL injection lub cross-site scripting (XSS) oraz przestrzeganie najlepszych praktyk bezpieczeństwa (Unadkat, 2023).

3.2 Wdrażanie

Systemy kontroli wersji

System kontroli wersji (VCS), znany również jako system kontroli kodu źródłowego, to oprogramowanie do śledzenia i zarządzania zmianami w systemie plików. VCS oferuje również narzędzia do współpracy, które umożliwiają udostępnianie zmian w systemie plików i powiązanie ich z innymi użytkownikami VCS. Podczas pracy na poziomie systemu plików, VCS śledzi dodawanie, usuwanie i modyfikowanie plików i katalogów. Repozytorium jest pojęciem związanym z VCS, miejscem, w którym przechowywana jest historia śledzenia systemu plików VCS. W przypadku poszczególnych plików kodu źródłowego, VCS śledzi dodawanie, usuwanie i zmiany linii tekstu w takich plikach. Popularne systemy kontroli wersji to Git, Mercurial, SVN i Preforce (*Perforce Software*, 2023).

Do tego projektu został wybrany VCS GitHub.

Projekt w serwisie GitHub jest przechowywany w repozytorium, będącym zbiorem wszystkich zmian w utworzonym kodzie. Jeśli nad projektem będzie pracować tylko jedna osoba, musi ona utworzyć nowe repozytorium. Jeśli nad projektem pracuje wielu deweloperów, każdy z nich sklonuje repozytorium pierwotnego twórcy. Wewnątrz repozytorium zmiany kodu są przechowywane jako gałęzie i commity.

Commit jest głównym obiektem deweloperskim, w którym przechowywane są wszystkie zmiany w kodzie podczas iteracji. Zasadniczo jest to lista wszystkich rzeczywistych zmian i link do poprzedniej wersji zatwierdzenia. Każde zatwierdzenie ma atrybuty: nazwę, datę utworzenia, autora i komentarze do bieżącej wersji.

Gałąź jest wskaźnikiem do zatwierdzenia z pewnymi zmianami. Na przykład dwóch programistów pobrało commit i każdy z nich zmodyfikował kod, tworząc nowy commit. Tworzy to dwie gałęzie projektu z różnym kodem: deweloper może wybrać, nad którym commitem chce pracować w następnej kolejności. Główna gałąź jest zwykle uważana za main lub master - programiści tworzą nowe gałęzie na jej podstawie. Możliwe jest również utworzenie nieograniczonej liczby gałęzi, dzięki czemu można wprowadzać nowe zmiany bez ingerencji w główny projekt.

Ustawienia deweloperskie a ustawienia produkcyjne

Ustawienia deweloperskie są używane podczas lokalnego developmentu aplikacji Django. Ich celem jest ułatwienie pracy programistów poprzez dostarczenie informacji debugujących, trybu developerskiego oraz umożliwienie szybkiego i elastycznego kodowania. Włączenie trybu debugowania (DEBUG=True) powoduje wyświetlanie szczegółowych informacji o błędach oraz traceback'u na stronach błędów. To ułatwia identyfikację problemów podczas developmentu, ale nie powinno być używane w środowisku produkcyjnym z powodu potencjalnych zagrożeń bezpieczeństwa.

Zmienne środowiskowe

Zmienne środowiskowe to z góry określone wartości, które są zwykle używane w celu zapewnienia możliwości konfigurowania wartości w kodzie spoza aplikacji.

Głównym celem zmiennych środowiskowych jest zminimalizowanie potrzeby modyfikowania i ponownego wdrażania aplikacji w przypadku zmian danych konfiguracyjnych. Na przykład w podanym przykładzie ze skryptu deploy.sh, jeśli zachodzi potrzeba zmiany trybu debugowania, nie ma potrzeby modyfikowania kodu źródłowego, przeprowadzania testów i ponownego wdrażania zmodyfikowanej aplikacji.

Modyfikowanie i udostępnianie kodu aplikacji może być skomplikowane i niesie ze sobą ryzyko wprowadzenia niezamierzonych błędów w środowisku produkcyjnym. Jeśli jednak *DEBUG* jest zdefiniowany przez zmienną środowiskową, a nie zakodowany na stałe w aplikacji, proces wprowadzania zmian staje się znacznie prostszy (Medlock, 2021).

Przypadki użycia zmiennych środowiskowych obejmują między innymi takie dane jak:

1. Tryb wykonania (np. produkcja, rozwój, inscenizacja itp.),
2. Nazwy domen,
3. Adresy URL/URI API,
4. Publiczne i prywatne klucze uwierzytelniające (bezpieczne tylko w aplikacjach serwerowych),
5. Grupowe adresy e-mail, np. dla działu marketingu, wsparcia, sprzedaży itp,
6. Nazwy kont usług.

Konteneryzacja Docker

Docker to oprogramowanie, które automatyzuje wdrażanie i zarządzanie aplikacjami w dedykowanym środowisku wirtualizacji. Praca wykonywana jest na poziomie systemu operacyjnego. Dzięki temu pakietowi oprogramowania aplikacje są pakowane w kontenery wraz z ich środowiskami i zależnościami, a użytkownik otrzymuje środowisko do zarządzania tymi kontenerami.

Główną cechą kontenerów jest ich stosunkowo krótki cykl życia. Każdy kontener może zostać zatrzymany, ponownie uruchomiony i, w razie potrzeby, zniszczony wraz ze wszystkimi znajdującymi się w nim danymi. Z tego powodu istnieje specjalna zasada projektowania aplikacji znana jako Stateless. Zaleca ona unikanie przechowywania ważnych danych w kontenerach.

Kontenery przechowują jedynie procesy i zależności wymagane do wykonania kodu. To sprawia, że uruchamianie jest wystarczająco szybkie. Lekka natura kontenerów oszczędza również miejsce na nośniku. Każdy proces ma swój własny kontener, który można bezpiecznie wyłączyć, na przykład w celu debugowania i aktualizacji. Nie ma to wpływu na działanie całego programu.

W wyniku konteneryzacji procesy są od siebie bezpiecznie odizolowane. Zwiększa to ogólne bezpieczeństwo całego systemu. Aplikacje działające wewnątrz kontenera nie ingerują w żaden sposób w główny system operacyjny, ponieważ nie mają do niego dostępu. Konteneryzacja umożliwia automatyczne wdrażanie aplikacji na różnych hostach (Docker documentation, 2023).

Do utworzenia obrazu docker z aplikacji można użyć pliku dockerfile. Plik dockerfile to plik konfiguracyjny opisujący instrukcje, które zostaną zastosowane podczas tworzenia obrazu Docker i uruchamiania kontenera. Plik Dockerfile jest tworzony w katalogu głównym projektu i nie ma rozszerzenia.

Dockerfile ma następującą logikę wypełniania:

1. Pierwszą instrukcją jest zawsze FROM, określająca obraz nadrzędny. Na przykład FROM python:latest.

2. Instrukcja RUN może akceptować łańcuch instrukcji systemu Linux, aby nie tworzyć niepotrzebnych warstw. Na przykład RUN apt-get update && apt-get install python3-pip -y && pip install --upgrade pip && pip install pipenv.

3. Instrukcja WORKDIR ustawia katalog roboczy kontenera. Na przykład WORKDIR /usr/src/app/. Kolejne polecenia RUN, CMD, ENTRYPOINT dziedziczą powiązanie WORKDIR.

4. Po nim zawsze następuje CMD, na przykład CMD [“python”, “manage.py”, “runserver”]. CMD dziedziczy wiązanie WORKDIR, więc manage.py będzie uruchamiany z folderu /usr/src/app/ (Docker documentation, 2023).

Plik dockerfile dla danego projektu wygląda następująco:

```
FROM python:3.10.6-alpine3.15

COPY . ./app
WORKDIR ./app

RUN pip install --no-cache-dir -r requirements.txt
EXPOSE 8000

CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
```

Rys. 19. Plik dockerfile służący do utworzenia obrazu zawierającego aplikację.
Źródło: opracowanie własne

W tym przykładzie aplikacja jest zainstalowana na systemie linux alpine przy użyciu python 3.10.6 w katalogu /app i działa z otwartym portem 8000 i przy użyciu ip 0.0.0.0 do pozwolenia połączenia z zewnątrz kontenera.

Skrypty wdrożeniowe

Do automatycznego wdrożenia aplikacji w tym projekcie wykorzystano skrypty bash wraz z docker-compose.

Bash to popularna powłoka systemu Unix i język poleceń, który zapewnia środowisko skryptowe do automatyzacji zadań. W zasadzie bash pozwala automatycznie w pełni kontrolować system Linux. Za pomocą Bash można pisać skrypty do wykonywania różnych czynności, takich jak konfigurowanie środowiska, instalowanie zależności, konfigurowanie sieci i wykonywanie poleceń Docker Compose.

Z kolei Docker Compose to narzędzie służące do definiowania i uruchamiania aplikacji Docker składających się z wielu kontenerów. Docker Compose upraszcza wdrażanie aplikacji składających się z wielu kontenerów. Używa pliku YAML do definiowania usług, sieci i woluminów wymaganych przez aplikację. Docker Compose zarządza orkiestracją kontenerów, w tym uruchamianiem, zatrzymywaniem i skalowaniem kontenerów, zapewniając spójność w różnych środowiskach. Razem, Bash i Docker Compose oferują potężną kombinację do wdrażania aplikacji.

```
cd /home/wym
mkdir -p rrat
cd ./rrat
mkdir -p db
cd ./db
wget -nc https://github.com/YLashko/notes/raw/main/db.sqlite3
sudo chown $(whoami):$(whoami) ./db.sqlite3
cd ../
echo 'version: "3.7"
services:
  rrat:
    image: wymm/rrat-thesis:0.2
    ports:
      - "8000:8000"
    environment:
      - DEBUG=True
    volumes:
      - /usr/local/rrat/db:/app/db
volumes:
  rrat:' > docker-compose.yml
sudo systemctl start docker
sudo nohup sudo docker-compose up
```

Rys. 20. Skrypt wdrożeniowy *deploy.sh*, zawierający skrypt docker-compose.

Źródło: opracowanie własne

Tabela 2. Wykorzystane polecenia do tworzenia pliku wdrożeniowego

Polecenie	Wyjaśnienie
cd	Zmienia bieżący katalog na podaną lokalizację.
mkdir	Tworzy nowy katalog o podanej nazwie.
wget	Pobiera pliki z internetu, korzystając z podanego URL-a.
docker-compose up	Uruchamia kontenery zdefiniowane w pliku docker-compose.yml.
nohup	Uruchamia polecenie, które kontynuuje działanie nawet po zamknięciu sesji terminala.
chown	Zmienia właściciela pliku lub katalogu.
systemctl start	Uruchamia usługę systemową za pomocą polecenia systemctl.
echo	Wyświetla podany tekst lub wartość zmiennej do SYSOUT. W danym przypadku służy do zapisania wartości do pliku za pomocą zmiany kanału SYSOUT na plik docker-compose.yml.

Źródło – opracowanie własne

Dla tego projektu utworzono skrypt wdrożeniowy *bash*, który tworzy katalog aplikacji rrat w katalogu użytkownika wym, a następnie tworzy folder dla bazy danych i pobiera podstawową wersję bazy danych, jeżeli nie ma jej w tym folderze (w przypadku hostingu w chmurze ta konfiguracja nie byłaby najlepszym rozwiązaniem, ponieważ usunięcie instancji maszyny spowodowałoby również usunięcie bazy danych, w tym celu należy utworzyć oddzielny serwer bazy danych). Skrypt docker-compose w tym przypadku jest zapisywany do pliku na maszynie docelowej za pomocą polecenia echo, a następnie uruchamiany za pomocą docker-compose up. Skrypt jest używany do skonfigurowania uruchomienia kontenera i w tym przypadku przekazuje port 8000 dla żądań do serwera Django i tworzy wolumin, który służy do utworzenia współdzielonego drzewa plików między kontenerem a maszyną, na której jest uruchomiony, w tym przypadku dla umożliwienia dostępu do katalogu bazy danych (Docker documentation, 2023).

Rozdział 4
Wnioski

Podsumowując analizę procesu tworzenia aplikacji internetowej oraz jej kluczowych komponentów, można wyróżnić kilka istotnych wniosków:

1. Planowanie jest kluczowe: Praca dyplomowa jednoznacznie podkreśla znaczenie dokładnego planowania dla sukcesu projektu aplikacji internetowej. Poprzez staranne definiowanie zakresu projektu, identyfikację wymagań biznesowych i wyznaczanie jasnych celów, tworzy solidne podstawy dla procesu rozwoju. Planowanie to fundament, na którym opiera się cały projekt.

2. Baza danych również jest kluczowym elementem: Badanie przeprowadzone w ramach tej pracy dowodzi, że stworzenie dobrze zaprojektowanej i zoptymalizowanej bazy danych jest kluczowym aspektem efektywnego zarządzania danymi.

3. Wartość projektowania interfejsu użytkownika (UI): Praca wykazuje, że projektowanie interfejsu użytkownika odgrywa fundamentalną rolę w tworzeniu atrakcyjnych i przyjaznych dla użytkownika aplikacji internetowej. Przestrzeganie zasad projektowania zorientowanego na użytkownika jest ważne dla stworzenia wizualnie atrakcyjnego i intuicyjnego interfejsu.

4. Integracja Ajax poprawia wrażenia użytkownika: Integracja technologii Ajax znacznie poprawia wrażenia użytkownika, umożliwiając dynamiczne aktualizacje i informacje zwrotne w czasie rzeczywistym. Ta praca dowodzi, że wykorzystanie tej technologii eliminuje potrzebę przeładowywania strony, co przekłada się na bardziej płynne i responsywne działanie aplikacji.

5. Walidacja danych gwarantuje wiarygodność: Walidacja danych jest kluczowym elementem zapewniającym wiarygodność i dokładność treści generowanych przez użytkowników. Niniejsza praca podkreśla, że wdrożenie solidnych mechanizmów walidacji danych jest niezbędne do ograniczenia ryzyka związanego z nieprawidłowymi lub złośliwymi danymi.

6. Moderacja treści jest istotna: Praca dyplomowa kładzie nacisk na kluczową rolę moderacji treści w utrzymaniu jakości i integralności aplikacji internetowej. Efektywne strategie moderacji są niezbędne, aby zapewnić, że treści generowane przez użytkowników są zgodne z wytycznymi społeczności.

7. Testowanie i wdrażanie to nieodzowne etapy: Ta praca podkreśla znaczenie dokładnych procedur testowych i procesu wdrażania podczas rozwoju aplikacji internetowej. Testowanie jest niezbędne do identyfikowania i rozwiązywania problemów funkcjonalnych oraz wydajnościowych.

Niniejsza praca podkreśla, że w procesie tworzenia aplikacji internetowych kluczowe jest dokładna analiza i dobre wykonanie wszystkich etapów, od planowania po testowanie, aby osiągnąć sukces i dostarczyć wartość użytkownikom.

Zakończenie

Podsumowując, niniejsza praca dyplomowa podjęła się kompleksowej analizy procesu tworzenia aplikacji internetowej przy użyciu Django, koncentrując się na rozwoju strony internetowej z recenzjami restauracji. Poprzez badanie różnych tematów, takich jak planowanie, wymagania biznesowe, tworzenie baz danych, projektowanie interfejsu użytkownika, walidacja danych, moderacja treści, integracja Ajaxa, testowanie, wdrażanie i innych, uzyskałem cenny wgląd w zawiłości związane z budowaniem aplikacji internetowej.

W pracy dyplomowej szczegółowo omówiono każdy z tych etapów i przedstawiono najlepsze praktyki oraz wyzwania, z jakimi można się spotkać podczas tworzenia aplikacji Django. Dodatkowo, przeprowadzono analizę porównawczą różnych narzędzi i technik, które można wykorzystać w trakcie procesu tworzenia aplikacji internetowych opartych na Django, co pozwoliło na identyfikację korzyści i ograniczeń związanych z wyborem tych technologii.

W zakończeniu, badanie procesu tworzenia aplikacji internetowej jest nie tylko fascynującym tematem, ale także kluczowym elementem w dzisiejszym świecie technologicznym. Framework Django oferuje programistom wiele możliwości i udogodnień, co sprawia, że jest popularnym wyborem w branży. Jednak, aby osiągnąć sukces w tworzeniu aplikacji internetowej, niezbędne jest zrozumienie procesu i przestrzeganie najlepszych praktyk. Moja praca dyplomowa ma na celu dostarczenie kompleksowej wiedzy na ten temat i mam nadzieję, że będzie stanowić cenny wkład w dziedzinie aplikacji internetowych opartych na Django.

Bibliografia

1. Eby, K. (b.r.). Everything You Need to Know About Gathering Project Requirements. *Smartsheet*. Pobrano 02.09.2023 z lokalizacji https://www.smartsheet.com/content/project-requirements
2. Wikipedia contributors. (2023). Tripadvisor. *Wikipedia*. Pobrano 02.09.2023 z lokalizacji https://en.wikipedia.org/wiki/Tripadvisor
3. *Project Management: Different Types of Requirement*. (b.r.). Pobrano 02.09.2023 z lokalizacji https://www.visual-paradigm.com/project-management/different-types-of-requirements/
4. Desmarais, F., CPA. (b.r.). Levels of software requirements. *www.linkedin.com*. Pobrano 02.09.2023 z lokalizacji https://www.linkedin.com/pulse/levels-software-requirements-fran%C3%A7ois-desmarais-cpa-cma/?trk=public_profile_article_view
5. Bigelow, S. J. (2020). What are the types of requirements in software engineering? *Software Quality*. Pobrano 02.09.2023 z lokalizacji https://www.techtarget.com/searchsoftwarequality/answer/What-are-requirements-types
6. Team, T. (2021c). *All you need to know about Django MVT Architecture.* (b.r.). TechVidvan. Pobrano 02.09.2023 z lokalizacji https://techvidvan.com/tutorials/djangos-mvt-architecture/
7. *What is Python? | Teradata*. (b.r.). Pobrano 02.09.2023 z lokalizacji https://www.teradata.com/Glossary/What-is-Python
8. Team, D. (2021). 12 Features of Python that make it The Most Popular Programming Language. *DataFlair*. Pobrano 02.09.2023 z lokalizacji https://data-flair.training/blogs/features-of-python/

9. Team, D. (2021). 12 Features of Python that make it The Most Popular Programming Language. *DataFlair*. Pobrano 02.09.2023 z lokalizacji https://data-flair.training/blogs/features-of-python/
10. Downey, A. B. (2015). *Think Python : How to Think Like a Computer Scientist*. Pobrano 02.09.2023 z lokalizacji https://openlibrary.org/books/OL26455778M/Think_Python
11. Wikipedia contributors. (2023). Django (web framework). *Wikipedia*. Pobrano 02.09.2023 z lokalizacji https://en.wikipedia.org/wiki/Django_(web_framework)
12. Korsun, J., & Korsun, J. (2023, February 10). Why We Use Django Framework & What Is Django Used For. *Software Development Blog & IT Tech Insights | Django Stars*. Pobrano 02.09.2023 z lokalizacji https://Djangostars.com/blog/why-we-use-Django-framework/
13. *What is JavaScript? - JavaScript (JS) Explained - AWS*. (b.r.). Amazon Web Services, Inc. Pobrano 02.09.2023 z lokalizacji https://aws.amazon.com/what-is/javascript/
14. Srivastav, P. (b.r.). *A Docker Tutorial for Beginners*. A Docker Tutorial for Beginners. Pobrano 02.09.2023 z lokalizacji https://docker-curriculum.com/
15. Kinsta. (2022, January 26). *What Is Nginx? A Basic Look at What It Is and How It Works*. Kinsta®. Pobrano 02.09.2023 z lokalizacji https://kinsta.com/knowledgebase/what-is-nginx/
16. The Council on Quality and Leadership. (2021, November 2). *12 Reasons Why Data Is Important - The Council on Quality and Leadership*. Pobrano 02.09.2023 z lokalizacji https://www.c-q-l.org/resources/guides/12-reasons-why-data-is-important/
17. Ellingwood, J. (2017). User Data Collection: Balancing Business Needs and User Privacy. *DigitalOcean*. Pobrano 02.09.2023 z lokalizacji https://www.digitalocean.com/community/tutorials/user-data-collection-balancing-business-needs-and-user-privacy

18. *What is first-party vs third-party data?* (b.r.). Pobrano 02.09.2023 z lokalizacji https://clearbit.com/resources/books/b2b-data/what-is-data
19. Douek, D. (2023, May 11). Explicit vs Implicit Data: Why it Matters for Pharma - phamax Digital. *phamax Digital*. Pobrano 02.09.2023 z lokalizacji https://phamax-digital.ch/academy/explicit-implicit-data/
20. *Cover Your Tracks*. (b.r.-b). Pobrano 02.09.2023 z lokalizacji https://coveryourtracks.eff.org/learn
21. Crawford, E. (2021). Website Tracking: Why and How Do Websites Track You? *CookiePro*. Pobrano 02.09.2023 z lokalizacji https://www.cookiepro.com/blog/website-tracking/
22. Mullins, C. S. (2022). database management system (DBMS). Data Management. Pobrano 02.09.2023 z lokalizacji https://www.techtarget.com/searchdatamanagement/definition/database-management-system
23. What is Cloud Storage? - Cloud Storage Explained - AWS. (b.r.). Amazon Web Services, Inc. Pobrano 02.09.2023 z lokalizacji https://aws.amazon.com/what-is/cloud-storage/
24. White, L. (2022). 5 Reasons Why a Good User Interface is Important. CODERSERA. Pobrano 02.09.2023 z lokalizacji https://codersera.com/blog/why-a-good-user-interface-is-important/
25. Wilding, C. (1998). *Practical GUI screen design. Pobrano 02.09.2023 z lokalizacji https://doi.org/10.1145/286498.286574*
26. Team, T. (2021). Django Project Structure and File Structure. TechVidvan. Pobrano 02.09.2023 z lokalizacji https://techvidvan.com/tutorials/Django-project-structure-layout/

27. Yigal, A. (2018, November 8). Sqlite vs. MySQL vs. PostgreSQL: A Comparison of Relational Databases. Logz.io. Pobrano 02.09.2023 z lokalizacji https://logz.io/blog/relational-database-comparison/
28. Editor. (2019, October 15). *Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch and others. AltexSoft.* Pobrano 02.09.2023 z lokalizacji https://www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/
29. Django. (b.r.). Django Project. Pobrano 02.09.2023 z lokalizacji https://docs.Djangoproject.com/en/4.2/topics/auth/
30. Field validations at frontend or backend? Which is better? *(b.r.). Stack Overflow.* Pobrano 02.09.2023 z lokalizacji https://stackoverflow.com/questions/54800075/field-validations-at-frontend-or-backend-which-is-better
31. Nahum, S. (2021). Input Validation: Client-Side & Server-Side Cybersecurity Deterrent. SecureCoding. Pobrano 02.09.2023 z lokalizacji https://www.securecoding.com/blog/input-validation/
32. Django. (b.r). *Django Project.* Pobrano 02.09.2023 z lokalizacji https://docs.Djangoproject.com/en/4.2/topics/i18n/
33. Dalibor. (2023, May 12). A Quick Guide to Django i18n. Pobrano 02.09.2023 z lokalizacji https://phrase.com/blog/posts/quick-guide-Django-i18n/
34. GeeksforGeeks. (2023). Analysis of Algorithms Big O analysis. GeeksforGeeks. Pobrano 02.09.2023 z lokalizacji https://www.geeksforgeeks.org/analysis-algorithms-big-o-analysis/
35. Dillon, A. (2003). *User interface design.* Pobrano 02.09.2023 z lokalizacji https://repository.arizona.edu/handle/10150/105299

36. Wikipedia contributors. (2023b). Big O notation. Wikipedia. Pobrano 02.09.2023 z lokalizacji https://en.wikipedia.org/wiki/Big_O_notation
37. Goyal, S. (2023). What Is Bash? Features, Concepts, Commands And More! *unstop.com*. Pobrano 02.09.2023 z lokalizacji https://unstop.com/blog/what-is-bash
38. *Try Docker Compose*. (2023, July 8). Docker Documentation. Pobrano 02.09.2023 z lokalizacji https://docs.docker.com/compose/gettingstarted/
39. (2023, July 8). Docker Documentation. Pobrano 02.09.2023 z lokalizacji https://docs.docker.com/
40. Vijay. (2023). Types of Software Testing: Different Testing Types with Details. *Software Testing Help*. Pobrano 02.09.2023 z lokalizacji https://www.softwaretestinghelp.com/types-of-software-testing/#1_Unit_Testing
41. Deshpande, D. (2023). Backend Testing – The Ultimate Guide You'll Ever Need. *Hurix Digital*. Pobrano 02.09.2023 z lokalizacji https://www.hurix.com/backend-testing-the-ultimate-guide-youll-ever-need/
42. Unadkat, J. (2023). Manual Testing Tutorial for Beginners | BrowserStack. *BrowserStack*. Pobrano 02.09.2023 z lokalizacji https://www.browserstack.com/guide/manual-testing-tutorial
43. What is a VCS? Overview of Version Control Software. (2023). *Perforce Software*. Pobrano 02.09.2023 z lokalizacji https://www.perforce.com/blog/vcs/what-is-version-control
44. Medlock, J. (2021, December 7). An Introduction to Environment Variables and How to Use Them. *Medium*. Pobrano 02.09.2023 z lokalizacji https://medium.com/chingu/an-introduction-to-environment-variables-and-how-to-use-them-f602f66d15fa

45. Sheth, H. (2022, May 14). Test Automation Using Pytest and Selenium WebDriver. *Medium*. Pobrano 02.09.2023 z lokalizacji https://himanshu-sheth.medium.com/test-automation-using-pytest-and-selenium-webdriver-955333b43997

Spis rysunków

Spis tabel