AKADEMIA FINANSÓW I BIZNESU VISTULA

Wydział Informatyki, Grafiki i Architektury

Kierunek studiów Informatyka

Specjalność Lic INŻ spec. Inżynieria technologii internetowych

Vladyslav Kopaihorodskyi

Numer albumu: **62267**

Tworzenie aplikacji internetowej dla skrótu linków przy użyciu frameworka Django

Praca inżynierska

napisana pod kierunkiem

mgr inż. Maria Baczyńska-Wilkowska

Warszawa, 2026

Spis treści

Streszczenie i słowa kluczowe

Streszczenie pracy dyplomowej

Celem niniejszej pracy dyplomowej jest zaprojektowanie oraz implementacja aplikacji internetowej służącej do skracania adresów URL, zrealizowanej przy użyciu framework'a Django. W ramach pracy opracowano system umożliwiający tworzenie skróconych linków, ich przechowywanie w bazie danych oraz przekierowywanie użytkowników do oryginalnych adresów.

Aplikacja została wyposażona w mechanizmy rejestracji i uwierzytelniania użytkowników, a także w zestaw statycznych oraz dynamicznych podstron. W pracy omówiono architekturę aplikacji webowych, zasady działania framework'a Django oraz proces implementacji i uruchomienia systemu.

Słowa kluczowe: Django, aplikacja internetowa, skracanie linków, Python, URL shortener, backend, baza danych.

Diploma thesis abstract

The aim of this engineering thesis is to design and implement a web application for shortening URLs using the Django framework. The project includes the development of a system that allows users to generate short links, store them in a database, and redirect users to original URLs.

The application provides user registration and authentication mechanisms as well as static and dynamic web pages. The thesis discusses the architecture of web applications, the principles of the Django framework, and the implementation and deployment process of the developed system.

Keywords: Django, web application, URL shortening, Python, backend, database.

1. ANALIZA WŁAŚCIWOŚCI FUNKCJONALNYCH I UŻYTKOWYCH APLIKACJI INTERNETOWYCH

W niniejszym rozdziale przedstawiona została analiza funkcjonalna oraz użytkowa aplikacji internetowych, ze szczególnym uwzględnieniem systemów przeznaczonych do skracania adresów URL. Omówione zostaną kluczowe aspekty wpływające na użyteczność takich systemów, ich architekturę funkcjonalną oraz decyzje projektowe podejmowane na etapie planowania i implementacji aplikacji. Rozdział stanowi podstawę teoretyczną do dalszych prac projektowych opisanych w kolejnych częściach pracy.

1.1 Właściwości funkcjonalne i użytkowe aplikacji internetowych (UI/UX)

Właściwości funkcjonalne aplikacji internetowej do skracania linków obejmują zestaw operacji, które użytkownik może wykonać za pomocą systemu. Podstawową funkcjonalnością jest możliwość wprowadzenia długiego adresu URL oraz uzyskania jego skróconej wersji w możliwie najkrótszym czasie. Proces ten powinien być prosty, intuicyjny oraz nie wymagać od użytkownika dodatkowych działań, takich jak rejestracja czy logowanie, o ile nie jest to konieczne.

Z punktu widzenia użytkownika kluczowe znaczenie ma również czytelność interfejsu oraz szybkość reakcji systemu. Elementy interfejsu użytkownika powinny być rozmieszczone w sposób logiczny i spójny, a komunikaty zwrotne – jednoznaczne i zrozumiałe. W aplikacjach skracających linki istotne jest również umożliwienie łatwego kopiowania wygenerowanego skrótu oraz jego dalszego wykorzystania.

Aspekty UX obejmują także dostosowanie aplikacji do różnych urządzeń, w tym komputerów oraz urządzeń mobilnych. Responsywność interfejsu wpływa bezpośrednio na komfort użytkowania oraz dostępność systemu dla szerokiego grona odbiorców.

1.2 Rozważanie rozwiązań projektowych przy tworzeniu aplikacji webowych

Na etapie projektowania aplikacji internetowej do skracania linków konieczne jest rozważenie różnych rozwiązań architektonicznych oraz technologicznych. Jednym z kluczowych wyborów jest decyzja dotycząca struktury aplikacji oraz sposobu obsługi żądań użytkowników. W przypadku aplikacji webowych często stosuje się architekturę opartą na podziale na warstwę prezentacji, logiki biznesowej oraz dostępu do danych.

Istotnym elementem projektowym jest również wybór metody generowania skróconych linków. Możliwe podejścia obejmują wykorzystanie losowych ciągów znaków, algorytmów opartych na kodowaniu liczbowym lub skrótów kryptograficznych. Każde z tych rozwiązań posiada swoje zalety oraz ograniczenia, które należy uwzględnić w kontekście planowanego zastosowania systemu.

Rozważania projektowe obejmują także kwestie związane z bezpieczeństwem, takie jak walidacja danych wejściowych, ochrona przed nadużyciami oraz zapewnienie poprawności przekierowań. Odpowiednie zaprojektowanie tych mechanizmów wpływa na niezawodność oraz stabilność aplikacji.

1.3 Zalety i wady wybranych rozwiązań funkcjonalnych

Zastosowane w projekcie rozwiązania funkcjonalne charakteryzują się szeregiem zalet, do których należy zaliczyć prostotę obsługi, szybkość działania oraz możliwość łatwej rozbudowy systemu. Wykorzystanie nowoczesnych frameworków webowych pozwala na skrócenie czasu implementacji oraz zapewnienie wysokiej jakości kodu.

Do zalet należy również zaliczyć modularność systemu, która umożliwia wprowadzanie nowych funkcjonalności bez konieczności gruntownej przebudowy istniejącej aplikacji. Mechanizmy automatycznego zarządzania bazą danych oraz obsługi żądań HTTP znacząco upraszczają proces tworzenia aplikacji.

Potencjalne wady mogą obejmować ograniczenia wydajnościowe w przypadku bardzo dużej liczby użytkowników lub skróconych linków. W takich sytuacjach konieczne może być zastosowanie dodatkowych mechanizmów optymalizacyjnych, takich jak cache'owanie danych czy skalowanie infrastruktury serwerowej.

2. WŁAŚCIWOŚCI I CECHY TECHNOLOGII WYKORZYSTYWANYCH DO TWORZENIA APLIKACJI INTERNETOWYCH W OPARCIU O DJANGO

Rozdział drugi poświęcony jest analizie technologii wykorzystanych do realizacji projektu aplikacji internetowej. Szczególną uwagę zwrócono na framework Django, jego architekturę oraz narzędzia wspierające proces tworzenia, testowania i wdrażania aplikacji webowych. Omówione zostaną również dodatkowe biblioteki i rozwiązania technologiczne wspomagające rozwój systemu.

2.1 Zalety frameworka Django w kontekście aplikacji webowych

Django jest jednym z najpopularniejszych frameworków webowych opartych na języku Python. Jego główną zaletą jest kompleksowość, która pozwala na szybkie tworzenie aplikacji internetowych bez konieczności korzystania z dużej liczby zewnętrznych bibliotek. Framework oferuje wbudowane mechanizmy obsługi baz danych, system routingu URL oraz narzędzia do zarządzania użytkownikami.

W kontekście aplikacji do skracania linków Django umożliwia łatwą implementację mechanizmu przekierowań oraz przechowywanie danych w relacyjnej bazie danych. Zastosowanie wbudowanego systemu ORM upraszcza operacje na danych oraz zwiększa bezpieczeństwo aplikacji poprzez eliminację ryzyka błędów związanych z ręcznym tworzeniem zapytań SQL.

Django kładzie również duży nacisk na bezpieczeństwo, oferując ochronę przed najczęściej występującymi zagrożeniami w aplikacjach webowych, takimi jak SQL Injection czy Cross-Site Scripting.

Tabela 2.1 Porównanie wybranych frameworków webowych

Cecha	Django	Flask	Laravel
Język programowania	Python	Python	PHP
Typ frameworka	Kompleksowy	Minimalistyczny	Kompleksowy
ORM	Wbudowany	Zewnętrzny	Wbudowany
Panel administracyjny	Tak	Nie	Tak
Bezpieczeństwo	Wysokie	Zależne od implementacji	Wysokie
Zastosowanie	Średnie i duże projekty	Małe aplikacje	Aplikacje biznesowe

Źródło: Opracowanie własne.

Jak przedstawiono w **tabeli 2.1**, framework Django wyróżnia się kompleksowym podejściem do tworzenia aplikacji internetowych. W porównaniu do rozwiązań minimalistycznych, takich jak Flask, Django oferuje większą liczbę wbudowanych mechanizmów, co wpływa na szybkość realizacji projektu oraz bezpieczeństwo systemu. W kontekście aplikacji do skracania linków wybór Django jest uzasadniony możliwością dalszej rozbudowy systemu.

2.1.1 Porównanie Django z innymi frameworkami webowymi

Framework Django często porównywany jest z innymi popularnymi rozwiązaniami wykorzystywanymi do tworzenia aplikacji internetowych, takimi jak Flask, Ruby on Rails czy Laravel. Każdy z tych frameworków posiada odmienne założenia projektowe oraz zakres oferowanych funkcjonalności, co wpływa na sposób ich zastosowania w konkretnych projektach.

Django wyróżnia się podejściem typu „batteries included", co oznacza, że dostarcza szeroki zestaw wbudowanych narzędzi umożliwiających szybkie rozpoczęcie pracy nad projektem. W porównaniu do Flask, który jest frameworkiem minimalistycznym, Django oferuje gotowe

rozwiązania w zakresie obsługi baz danych, uwierzytelniania użytkowników oraz panelu administracyjnego.

W kontekście aplikacji do skracania linków wybór Django uzasadniony jest potrzebą stabilności, bezpieczeństwa oraz możliwości dalszej rozbudowy systemu. Framework ten sprawdza się szczególnie w projektach, które mogą być rozwijane w przyszłości o dodatkowe funkcjonalności.

2.2 Narzędzia i biblioteki wspierające Django

Ekosystem Django obejmuje szeroką gamę narzędzi i bibliotek, które wspierają proces rozwoju aplikacji internetowych. Jednym z najczęściej wykorzystywanych rozszerzeń jest Django REST Framework, umożliwiający tworzenie interfejsów API zgodnych z architekturą REST. Zastosowanie tego rozwiązania pozwala na łatwą integrację aplikacji z innymi systemami oraz oddzielenie warstwy backendowej od frontendowej.

Istotnym elementem jest także panel administracyjny Django, który umożliwia zarządzanie danymi aplikacji bez konieczności tworzenia dedykowanego interfejsu administracyjnego. Narzędzie to znacząco przyspiesza proces testowania oraz zarządzania systemem.

Dodatkowe biblioteki dostępne w ekosystemie Django umożliwiają implementację mechanizmów uwierzytelniania, obsługę formularzy oraz walidację danych wejściowych, co ma kluczowe znaczenie dla bezpieczeństwa i stabilności aplikacji.

2.3 Narzędzia wdrożeniowe i optymalizacyjne

Podczas wdrażania aplikacji internetowej istotną rolę odgrywają narzędzia umożliwiające zarządzanie środowiskiem uruchomieniowym. Konteneryzacja z wykorzystaniem technologii Docker pozwala na uruchamianie aplikacji w spójnym i powtarzalnym środowisku, niezależnie od platformy systemowej.

W przypadku dalszego rozwoju aplikacji możliwe jest zastosowanie narzędzi takich jak Redis czy Celery, które umożliwiają obsługę zadań asynchronicznych oraz zwiększenie wydajności systemu.

Rozwiązania te pozwalają na skalowanie aplikacji oraz jej przygotowanie do obsługi większej liczby użytkowników.

2.3.1 Bezpieczeństwo aplikacji internetowych opartych na Django

Bezpieczeństwo stanowi jeden z kluczowych aspektów projektowania aplikacji internetowych dostępnych publicznie. Framework Django oferuje szereg wbudowanych mechanizmów zabezpieczających aplikację przed najczęściej spotykanymi zagrożeniami, takimi jak ataki typu SQL Injection, Cross-Site Scripting czy Cross-Site Request Forgery.

Dzięki automatycznej obsłudze zapytań do bazy danych przez ORM, Django minimalizuje ryzyko wstrzykiwania nieautoryzowanych zapytań SQL. Ponadto framework wymusza stosowanie bezpiecznych praktyk programistycznych, takich jak walidacja danych wejściowych oraz odpowiednie zarządzanie sesjami użytkowników.

Zastosowanie tych mechanizmów ma istotne znaczenie w kontekście aplikacji do skracania linków, które są narażone na próby nadużyć oraz generowanie złośliwych adresów URL.

3. ZAŁOŻENIA PROJEKTOWE I REALIZACJA APLIKACJI INTERNETOWEJ

W niniejszym rozdziale opisano założenia projektowe aplikacji internetowej do skracania linków oraz proces jej implementacji. Przedstawione zostaną decyzje techniczne, architektura systemu oraz etapy realizacji projektu.

3.1 Założenia techniczne projektu

Projekt aplikacji został oparty na frameworku Django oraz języku Python. System zaprojektowano w architekturze klient–serwer, gdzie backend odpowiada za przetwarzanie danych i obsługę logiki biznesowej, natomiast frontend zapewnia interakcję z użytkownikiem. Do przechowywania danych wykorzystano relacyjną bazę danych.

Założenia techniczne obejmują również zapewnienie bezpieczeństwa danych oraz stabilności działania systemu. Projekt przewiduje możliwość dalszej rozbudowy aplikacji o dodatkowe funkcjonalności.

Tabela 3.1 Wymagania funkcjonalne i niefunkcjonalne aplikacji

Typ wymagania	Opis
Funkcjonalne	Generowanie skróconych adresów URL
Funkcjonalne	Obsługa przekierowań do adresów docelowych
Funkcjonalne	Walidacja danych wejściowych
Niefunkcjonalne	Wysoka dostępność systemu
Niefunkcjonalne	Bezpieczeństwo danych
Niefunkcjonalne	Skalowalność aplikacji

Źródło: Opracowanie własne.

Tabela 3.1 przedstawia zestaw podstawowych wymagań, które muszą zostać spełnione przez projektowaną aplikację internetową. Uwzględnienie zarówno wymagań funkcjonalnych, jak i

niefunkcjonalnych pozwoliło na zaprojektowanie systemu stabilnego, bezpiecznego oraz dostosowanego do potrzeb użytkowników końcowych.

3.2 Implementacja funkcjonalności aplikacji

Proces implementacji obejmował stworzenie modeli danych, widoków oraz szablonów interfejsu użytkownika. Szczególną uwagę zwrócono na poprawność generowania skróconych linków oraz obsługę przekierowań. Każdy skrócony adres URL jest jednoznacznie powiązany z adresem oryginalnym.

Aplikacja została zaprojektowana w sposób umożliwiający łatwe testowanie oraz utrzymanie kodu. Zastosowanie dobrych praktyk programistycznych wpłynęło na czytelność oraz jakość implementacji.

3.3 Testowanie i wdrożenie aplikacji

Testowanie aplikacji obejmowało sprawdzenie poprawności działania poszczególnych funkcjonalności oraz stabilności systemu. Przeprowadzone testy potwierdziły poprawne działanie aplikacji zgodnie z założeniami projektowymi.

Wdrożenie aplikacji polegało na konfiguracji środowiska serwerowego oraz uruchomieniu systemu w warunkach zbliżonych do produkcyjnych. Aplikacja została przygotowana do dalszego rozwoju oraz eksploatacji.

3.4 Analiza wymagań funkcjonalnych i niefunkcjonalnych

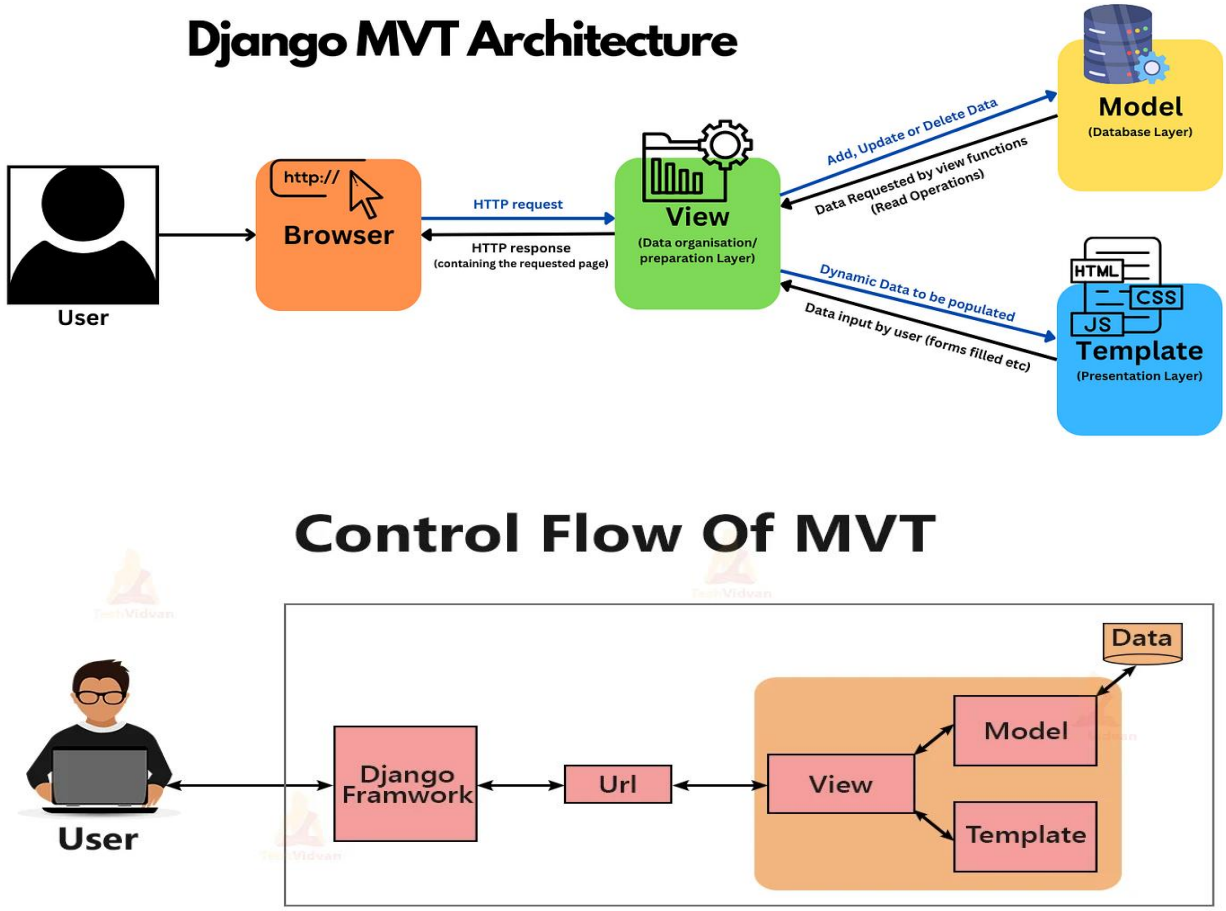
Analiza wymagań stanowi istotny etap realizacji każdego projektu informatycznego. W przypadku aplikacji internetowej do skracania linków zidentyfikowano zarówno wymagania funkcjonalne, jak i niefunkcjonalne, które muszą zostać spełnione przez system.

Do wymagań funkcjonalnych zaliczono m.in. możliwość skracania adresów URL, obsługę przekierowań oraz walidację danych wejściowych. Wymagania niefunkcjonalne obejmowały natomiast aspekty związane z wydajnością, bezpieczeństwem oraz dostępnością aplikacji.

Dokładne określenie wymagań pozwoliło na zaprojektowanie systemu spełniającego oczekiwania użytkowników oraz zapewniającego stabilne działanie aplikacji.

4. TWORZENIE APLIKACJI INTERNETOWEJ W OPARCIU DJANGO: OD PROJEKTU DO WDROŻENIA

W niniejszym rozdziale przedstawiono szczegółowy proces tworzenia aplikacji internetowej do skracania linków z wykorzystaniem frameworka Django. Rozdział obejmuje etapy projektowania architektury systemu, implementacji funkcjonalności oraz interfejsu użytkownika, a także testowania, optymalizacji i wdrożenia aplikacji. Opisane zagadnienia stanowią praktyczną realizację założeń projektowych przedstawionych we wcześniejszych rozdziałach pracy.



Rysunek 4.1 - Architektura aplikacji internetowej w oparciu o wzorzec MVT w frameworku Django

Źródło: Opracowanie własne na podstawie dokumentacji frameworka Django.

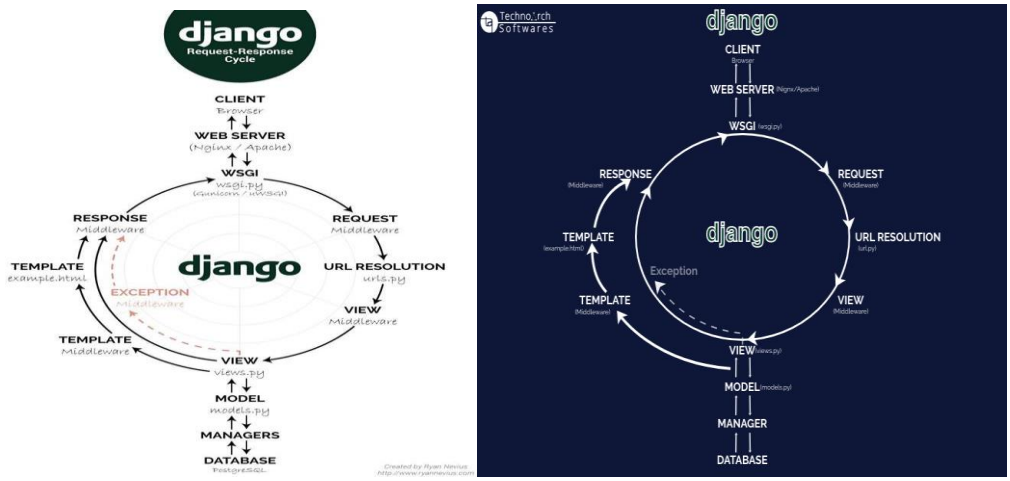
4.1 Projektowanie i architektura aplikacji internetowej

Proces tworzenia aplikacji internetowej do skracania linków rozpoczęto od zaprojektowania architektury systemu, której celem było zapewnienie czytelności, skalowalności oraz łatwości dalszego rozwoju aplikacji. W projekcie zastosowano wzorzec architektoniczny MVT (Model–View–Template), charakterystyczny dla frameworka Django, umożliwiający wyraźny podział odpowiedzialności pomiędzy poszczególne komponenty systemu.

Modele danych odpowiadają za przechowywanie kluczowych informacji związanych z funkcjonowaniem aplikacji, takich jak oryginalne adresy URL, wygenerowane skróty oraz liczba przekierowań. Dzięki wykorzystaniu systemu ORM dostępnego w Django możliwe było zdefiniowanie struktury bazy danych w sposób obiektowy, co znacząco ułatwiło zarządzanie danymi oraz zapewniło spójność logiczną systemu.

Widoki aplikacji realizują logikę biznesową, obejmującą obsługę żądań użytkowników, generowanie skróconych linków oraz mechanizm przekierowań do adresów docelowych. Warstwa szablonów odpowiada natomiast za prezentację danych oraz interakcję z użytkownikiem końcowym. Taki podział umożliwia łatwe utrzymanie kodu oraz jego dalszą rozbudowę bez konieczności ingerencji w całą strukturę aplikacji.

Architektura systemu została zaprojektowana w sposób modułowy, co pozwala na stopniowe dodawanie nowych funkcjonalności, takich jak system kont użytkowników czy zaawansowane statystyki wykorzystania linków. Zastosowanie frameworka Django umożliwiło szybkie stworzenie stabilnego szkieletu projektu, zapewniającego wysoką jakość kodu oraz zgodność z dobrymi praktykami programistycznymi.

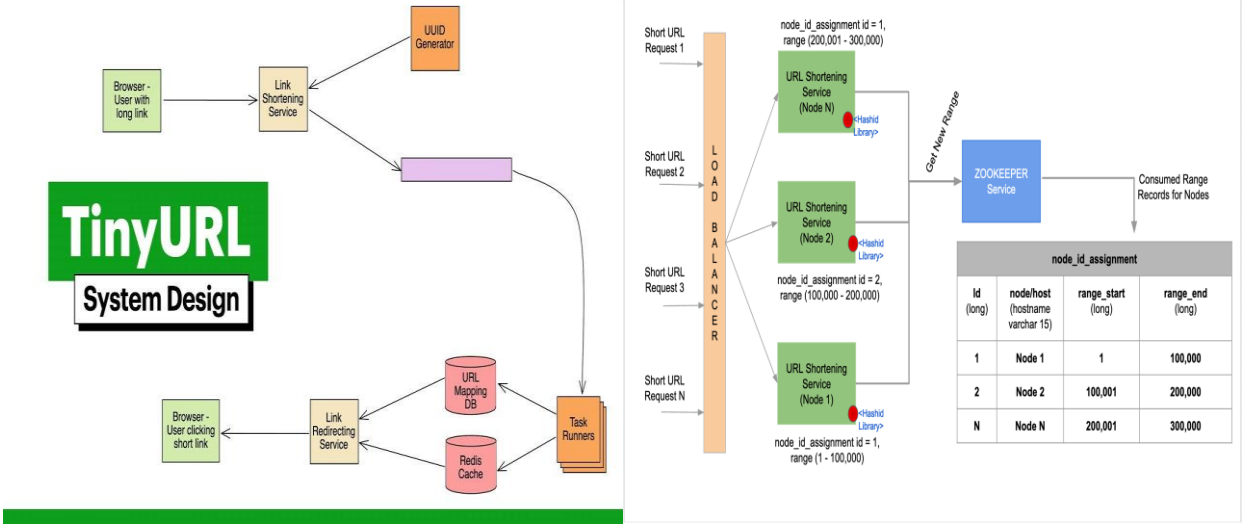


Rysunek 4.2 - Cykl obsługi żądania HTTP w aplikacji opartej na frameworku Django

Źródło: Opracowanie własne na podstawie dokumentacji Django.

Na rysunku **4.2** - przedstawiono schemat obsługi żądania HTTP w aplikacji Django. Proces ten obejmuje odebranie żądania od użytkownika, jego przetworzenie przez warstwę widoków oraz wygenerowanie odpowiedzi, która następnie przekazywana jest do przeglądarki użytkownika. Takie podejście zapewnia przejrzystość działania systemu oraz ułatwia jego debugowanie.

4.2 Rozwój funkcjonalności i interfejsu użytkownika



Rysunek 4.3 - Schemat procesu generowania i obsługi skróconych adresów URL

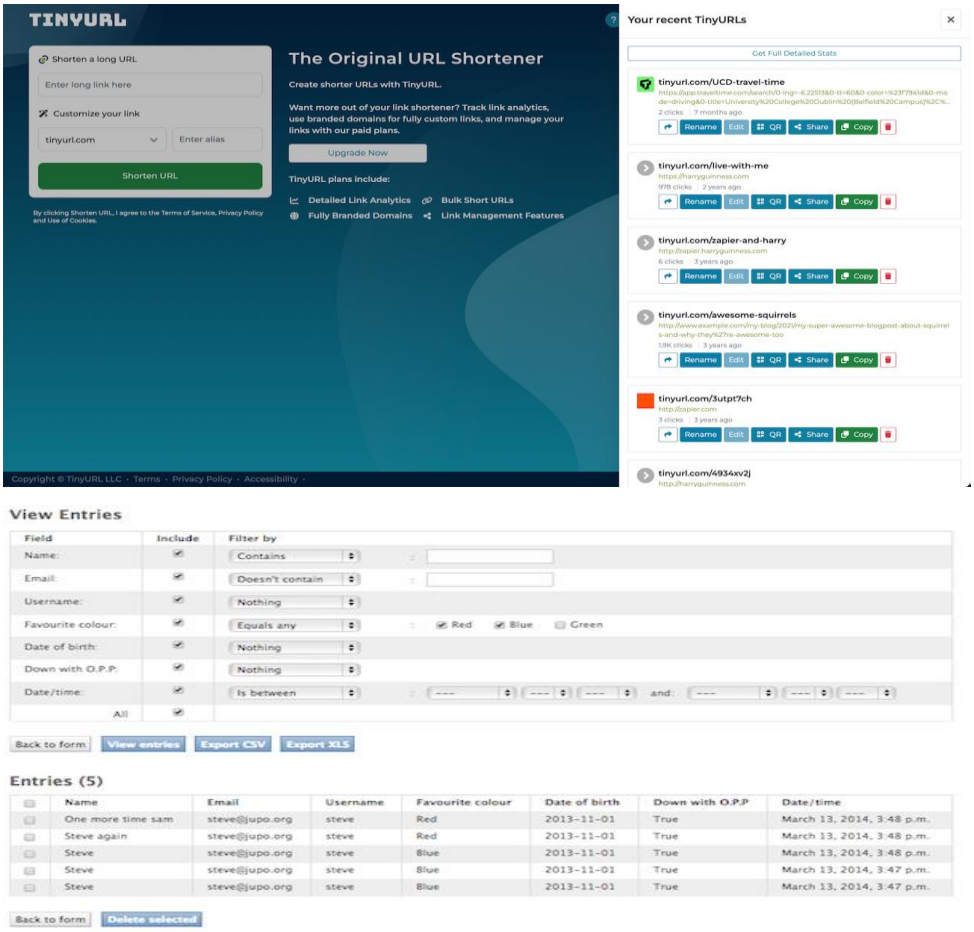
Źródło: Opracowanie własne.

Etap implementacji aplikacji obejmował rozwój kluczowych funkcjonalności systemu, które stanowią podstawę działania aplikacji do skracania linków. Najważniejszym elementem było stworzenie mechanizmu generowania unikalnych skrótów dla wprowadzanych przez użytkownika adresów URL. Proces ten polega na wygenerowaniu losowego lub algorytmicznie utworzonego identyfikatora, który następnie zapisywany jest w bazie danych wraz z odpowiadającym mu adresem docelowym.

Istotnym elementem implementacji była obsługa formularzy umożliwiających użytkownikowi wprowadzenie adresu URL. W projekcie zastosowano wbudowane mechanizmy Django do walidacji danych wejściowych, co pozwoliło na sprawdzenie poprawności wprowadzanych adresów oraz zapobieganie zapisowi nieprawidłowych danych w bazie danych. Takie podejście zwiększa bezpieczeństwo aplikacji oraz poprawia jej niezawodność.

Interfejs użytkownika został zaprojektowany w sposób prosty i intuicyjny, z naciskiem na minimalizm oraz czytelność. Głównym celem było umożliwienie użytkownikowi szybkiego skrócenia linku bez konieczności wykonywania zbędnych operacji. Interfejs prezentuje wygenerowany skrócony adres w sposób jednoznaczny, umożliwiając jego łatwe skopiowanie lub bezpośrednie wykorzystanie.

Podczas projektowania interfejsu zwrócono również uwagę na obsługę błędów oraz komunikaty informacyjne wyświetlane użytkownikowi. Jasne i zrozumiałe komunikaty zwiększają komfort korzystania z aplikacji oraz minimalizują ryzyko nieprawidłowego jej użycia, co jest szczególnie istotne w aplikacjach dostępnych publicznie.



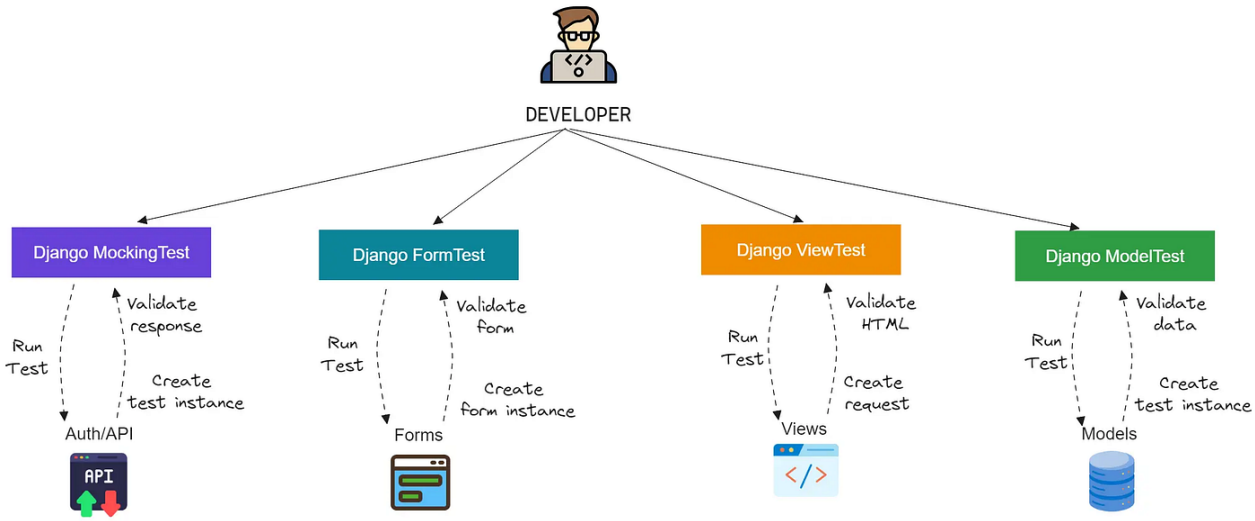
Rysunek 4.4 - Przykładowy interfejs użytkownika aplikacji do skracania linków

Źródło: Opracowanie własne.

4.3 Testowanie, optymalizacja i wdrożenie aplikacji

Po zakończeniu etapu implementacji aplikacja została poddana procesowi testowania, którego celem było sprawdzenie poprawności działania wszystkich kluczowych funkcjonalności systemu. Przeprowadzono testy jednostkowe, umożliwiające weryfikację poprawności działania poszczególnych komponentów aplikacji, takich jak generowanie skróconych linków, zapisywanie danych w bazie oraz obsługa przekierowań.

Dodatkowo wykonano testy funkcjonalne, które pozwoliły ocenić działanie aplikacji z perspektywy użytkownika końcowego. Testy te obejmowały m.in. wprowadzanie poprawnych i niepoprawnych adresów URL, sprawdzenie reakcji systemu na błędy oraz weryfikację poprawności przekierowań do adresów docelowych. Uzyskane wyniki potwierdziły poprawne działanie aplikacji zgodnie z założeniami projektowymi.



Rysunek 4.3 Etapy testowania oraz wdrażania aplikacji internetowej

Źródło: Opracowanie własne.

Po zakończeniu testów przeprowadzono optymalizację aplikacji pod kątem wydajności oraz stabilności działania. Optymalizacja obejmowała m.in. usprawnienie zapytań do bazy danych oraz uproszczenie logiki obsługi żądań HTTP. Następnie aplikacja została przygotowana do wdrożenia w środowisku produkcyjnym.

Proces wdrożenia obejmował konfigurację serwera aplikacyjnego, bazy danych oraz środowiska uruchomieniowego. Po wdrożeniu przeprowadzono końcowe testy stabilności, potwierdzające gotowość aplikacji do dalszej eksploatacji oraz rozwoju.

5. DOKUMENTACJA INSTALACYJNA I UŻYTKOWA APLIKACJI INTERNETOWEJ

Rozdział piąty poświęcony jest dokumentacji aplikacji internetowej, obejmującej zarówno instrukcję instalacji systemu, jak i opis sposobu korzystania z aplikacji przez użytkownika końcowego. Odpowiednia dokumentacja stanowi istotny element każdego projektu informatycznego, umożliwiając jego prawidłowe wdrożenie, eksploatację oraz dalszy rozwój.

5.1 Dokumentacja instalacyjna aplikacji

Dokumentacja instalacyjna zawiera szczegółowy opis kroków niezbędnych do uruchomienia aplikacji internetowej w środowisku lokalnym lub produkcyjnym. Proces instalacji obejmuje przygotowanie środowiska programistycznego, instalację wymaganych bibliotek oraz konfigurację bazy danych.

W projekcie zastosowano standardowe narzędzia dostępne w ekosystemie Django oraz języka Python, co znacząco upraszcza proces instalacji. Dzięki jasno określonym wymaganiom systemowym możliwe jest szybkie uruchomienie aplikacji bez konieczności posiadania zaawansowanej wiedzy administracyjnej.

5.2 Dokumentacja użytkowa aplikacji

Dokumentacja użytkowa została opracowana z myślą o użytkownikach końcowych aplikacji. Opisuje ona sposób korzystania z podstawowych funkcjonalności systemu, takich jak wprowadzanie adresów URL, generowanie skróconych linków oraz ich dalsze wykorzystanie.

Instrukcja użytkowa została napisana w sposób prosty i zrozumiały, co umożliwia szybkie zapoznanie się z działaniem aplikacji. Odpowiednie komunikaty oraz intuicyjny interfejs użytkownika minimalizują ryzyko błędów podczas korzystania z systemu.

5.3 Konserwacja i aktualizacja systemu

Konserwacja aplikacji obejmuje regularne monitorowanie jej działania oraz aktualizację wykorzystywanych bibliotek i zależności projektowych. Aktualizacje mają na celu poprawę bezpieczeństwa systemu, eliminację potencjalnych błędów oraz wprowadzenie nowych funkcjonalności.

Projekt aplikacji został przygotowany w sposób umożliwiający łatwe wprowadzanie zmian oraz rozbudowę systemu. Modularna struktura kodu pozwala na efektywne zarządzanie projektem w dłuższej perspektywie czasowej.

5.4 Znaczenie dokumentacji w projektach informatycznych

Dokumentacja techniczna oraz użytkowa odgrywa kluczową rolę w projektach informatycznych, umożliwiając prawidłowe wdrożenie oraz utrzymanie systemu. Dobrze opracowana dokumentacja ułatwia zrozumienie architektury aplikacji oraz zasad jej działania.

W kontekście projektów akademickich dokumentacja stanowi również potwierdzenie poprawności realizacji założeń projektowych oraz umiejętności autora w zakresie opisu rozwiązań technicznych.

6. MOŻLIWOŚCI ROZWOJU I ROZSZERZENIA APLIKACJI INTERNETOWEJ

W niniejszym rozdziale omówiono potencjalne kierunki dalszego rozwoju aplikacji internetowej do skracania linków. Przedstawione propozycje rozbudowy systemu uwzględniają zarówno aspekty funkcjonalne, jak i technologiczne, które mogą zwiększyć użyteczność oraz skalowalność aplikacji w przyszłości.

6.1 Rozszerzenie funkcjonalności aplikacji

Jednym z możliwych kierunków rozwoju aplikacji jest wprowadzenie mechanizmu rejestracji i logowania użytkowników. Dzięki temu możliwe byłoby przypisywanie skróconych linków do konkretnych kont użytkowników oraz zarządzanie nimi w bardziej zaawansowany sposób.

Dodatkową funkcjonalnością mogłoby być generowanie statystyk dotyczących liczby wejść na poszczególne linki, a także analiza danych takich jak czas czy lokalizacja użytkowników. Rozszerzenia te zwiększyłyby wartość aplikacji oraz jej zastosowanie w celach analitycznych.

6.2 Skalowalność i optymalizacja wydajności

W przypadku wzrostu liczby użytkowników konieczne może być zastosowanie mechanizmów zwiększających wydajność systemu. Do takich rozwiązań należą m.in. cache’owanie danych, optymalizacja zapytań do bazy danych oraz wykorzystanie zewnętrznych usług przechowywania danych.

Zastosowanie narzędzi takich jak Redis czy systemy kolejkowe pozwala na obsługę większej liczby zapytań bez pogorszenia jakości działania aplikacji. Skalowalność systemu jest istotnym aspektem w kontekście dalszego rozwoju projektu.

6.3 Integracja z innymi systemami

Kolejnym możliwym kierunkiem rozwoju aplikacji jest integracja z innymi systemami informatycznymi za pomocą interfejsów API. Dzięki temu aplikacja mogłaby być wykorzystywana jako usługa zewnętrzna przez inne aplikacje webowe lub mobilne.

Integracja z innymi systemami zwiększa elastyczność rozwiązania oraz umożliwia jego zastosowanie w szerszym kontekście biznesowym i technologicznym.

6.4 Możliwość komercyjnego wykorzystania aplikacji

Aplikacja internetowa do skracania linków może znaleźć zastosowanie nie tylko w celach edukacyjnych, ale również komercyjnych. Systemy tego typu wykorzystywane są w marketingu internetowym, analizie ruchu sieciowego oraz zarządzaniu kampaniami reklamowymi.

Rozbudowa aplikacji o dodatkowe funkcjonalności, takie jak analiza statystyk czy integracja z narzędziami analitycznymi, umożliwiłaby jej wykorzystanie w środowisku biznesowym.

PODSUMOWANIE

Niniejsza praca inżynierska poświęcona była projektowi i realizacji aplikacji internetowej do skracania linków przy użyciu frameworka Django. Celem pracy było zaprojektowanie oraz zaimplementowanie systemu spełniającego określone wymagania funkcjonalne i techniczne, a także umożliwiającego dalszą rozbudowę w przyszłości.

W kolejnych rozdziałach pracy omówiono zagadnienia teoretyczne związane z aplikacjami internetowymi oraz zastosowanymi technologiami, a następnie przedstawiono proces realizacji projektu – od etapu projektowania architektury, poprzez implementację funkcjonalności, aż po testowanie i wdrożenie aplikacji. Zastosowanie frameworka Django pozwoliło na stworzenie stabilnego i bezpiecznego systemu, który spełnia założenia projektowe.

Zrealizowana aplikacja stanowi praktyczny przykład wykorzystania nowoczesnych technologii webowych oraz potwierdza możliwość efektywnego tworzenia aplikacji internetowych przy użyciu frameworków wysokiego poziomu. Praca umożliwiła autorowi rozwinięcie umiejętności z zakresu projektowania systemów informatycznych, programowania oraz analizy problemów inżynierskich.

Osiągnięte rezultaty potwierdzają, że założone cele pracy zostały zrealizowane, a stworzony system może stanowić podstawę do dalszego rozwoju oraz wykorzystania w praktycznych zastosowaniach.

Ponadto realizacja pracy umożliwiła praktyczne zastosowanie wiedzy teoretycznej zdobytej w trakcie studiów oraz rozwinięcie umiejętności analitycznego podejścia do problemów inżynierskich. Proces tworzenia aplikacji obejmował pełny cykl życia oprogramowania, od analizy wymagań, poprzez projektowanie i implementację, aż po testowanie i wdrożenie systemu.

Zdobyte doświadczenie może zostać wykorzystane w przyszłych projektach informatycznych oraz stanowi solidną podstawę do dalszego rozwoju kompetencji zawodowych autora w obszarze tworzenia aplikacji webowych.

W pracy przedstawiono proces projektowania i realizacji aplikacji internetowej do skracania linków przy użyciu frameworka Django. Zrealizowany projekt umożliwił praktyczne zastosowanie wiedzy z zakresu programowania webowego oraz inżynierii oprogramowania. Stworzona aplikacja spełnia założone wymagania funkcjonalne i techniczne oraz stanowi solidną podstawę do dalszej rozbudowy.

BIBLIOGRAFIA

1. **Django Software Foundation.** (2024). *Official Django Documentation*. Dostęp online: https://docs.djangoproject.com/ (dostęp: 10.06.2026).
2. **Django REST Framework.** (2024). *Official Documentation*. Dostęp online: https://www.django-rest-framework.org/ (dostęp: 10.06.2026).
3. **Python Software Foundation.** (2024). *Python Documentation*. Dostęp online: https://docs.python.org/ (dostęp: 10.06.2026).
4. **Sommerville, I.** (2016). *Software Engineering*. Pearson Education.
5. **Gamma, E., Helm, R., Johnson, R., Vlissides, J.** (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.