Sprawozdanie z Pracy Projektowej

Dominik Gwóźdź
Kamil Rudowski
Aleksander Craven

8.06.2024

Spis treści

1 Opis funkcjonalny systemu

System to forum internetowe poświęcone tematyce anime, stworzone w celu umożliwienia użytkownikom dyskusji na tematy związane z anime. Użytkownicy mogą zakładać nowe wątki, odpowiadać na istniejące posty, rejestrować się, logować, zmieniać język interfejsu oraz przeglądać najnowsze i najpopularniejsze wątki. Administratorzy mają możliwość zarządzania wątkami oraz postami, a także monitorowania aktywności użytkowników.

1.1 Główne funkcjonalności

- **Rejestracja i logowanie użytkowników**: Użytkownicy mogą zakładać konta oraz logować się do systemu, aby uzyskać dostęp do pełnych funkcji forum. Proces rejestracji wymaga podania unikalnego adresu e-mail i hasła, a także nazwy użytkownika.
- **Zakładanie nowych wątków i dodawanie postów**: Zalogowani użytkownicy mogą tworzyć nowe wątki w wybranych kategoriach oraz odpowiadać na istniejące posty. Każdy wątek zawiera tytuł i treść, a posty mogą być edytowane lub usuwane przez ich autorów.
- **Przeglądanie kategorii i wątków**: Użytkownicy mogą przeglądać dostępne kategorie oraz wątki, sortować je według popularności, daty i liczby odpowiedzi. Każda kategoria zawiera listę wątków, które można filtrować i przeszukiwać.
- **Zmiana języka interfejsu**: System umożliwia użytkownikom zmianę języka interfejsu, co zwiększa dostępność dla międzynarodowych użytkowników. Dostępne języki to polski i angielski, a wybór języka jest zapisywany w preferencjach użytkownika.
- **Wyświetlanie najnowszych i najpopularniejszych wątków**: Na stronie głównej wyświetlane są najnowsze oraz najpopularniejsze wątki. Lista najpopularniejszych wątków jest sortowana według liczby odpowiedzi i wyświetleń.
- **Edycja i usuwanie postów**: Autorzy postów oraz administratorzy mogą edytować i usuwać posty, co zapewnia moderację treści. Edycje są zapisywane, a użytkownicy są powiadamiani o zmianach w wątkach, które obserwują.
- **Logowanie akcji użytkowników**: Rejestrowanie kluczowych akcji użytkowników takich jak logowanie, wylogowanie, tworzenie wątków i postów. Te dane są przechowywane w bazie danych i mogą być używane do analizy aktywności użytkowników.

2 Opis technologiczny

Projekt został zrealizowany przy użyciu nowoczesnych technologii, które zapewniają skalowalność, wydajność oraz łatwość wdrażania.

2.1 Użyte technologie

- **Python 3.12**: Język programowania używany do tworzenia logiki aplikacji. Python jest znany z czytelności kodu i dużej liczby bibliotek wspierających rozwój aplikacji webowych.

- **Django 5.0.6**: Framework webowy, który zapewnia solidną podstawę do budowy aplikacji internetowych. Django oferuje wbudowane mechanizmy uwierzytelniania, administrowania, ORM oraz wiele innych narzędzi ułatwiających tworzenie aplikacji.
- **Django REST Framework**: Narzędzie do tworzenia API, które ułatwia integrację z innymi systemami. DRF umożliwia łatwe tworzenie endpointów API, serializację danych oraz zarządzanie uwierzytelnianiem i autoryzacją.
- **MySQL**: Relacyjna baza danych używana do przechowywania danych użytkowników, wątków oraz postów. MySQL jest wydajny, skalowalny i dobrze wspierany przez Django.
- **Bootstrap 4**: Framework CSS, który ułatwia tworzenie responsywnych i atrakcyjnych interfejsów użytkownika. Bootstrap zawiera gotowe komponenty CSS i JavaScript, które przyspieszają tworzenie nowoczesnych interfejsów.
- **JavaScript**: Używany do dynamicznych interakcji w przeglądarce, takich jak zmiana języka interfejsu i obsługa formularzy. JavaScript wraz z jQuery umożliwia tworzenie interaktywnych i responsywnych elementów strony.
- **Git**: System kontroli wersji używany do zarządzania kodem źródłowym oraz współpracy zespołowej. Git umożliwia śledzenie zmian w kodzie, zarządzanie wersjami oraz współpracę wielu programistów nad jednym projektem.

2.2 Architektura systemu

Architektura systemu została zaprojektowana w sposób modułowy, co ułatwia jej rozbudowę oraz utrzymanie. System składa się z następujących warstw:

- **Warstwa prezentacji**: Odpowiada za interfejs użytkownika, wykorzystując HTML, CSS (Bootstrap) oraz JavaScript. Ta warstwa jest odpowiedzialna za wyświetlanie danych oraz obsługę interakcji użytkownika.
- **Warstwa logiki aplikacji**: Realizowana przy użyciu Django, zawiera logikę biznesową oraz kontrolery. Ta warstwa zarządza przetwarzaniem danych, obsługą zapytań HTTP oraz logiką aplikacji.
- **Warstwa dostępu do danych**: Odpowiada za komunikację z bazą danych MySQL. Ta warstwa obejmuje modele Django, które definiują strukturę bazy danych oraz operacje CRUD.
- **API**: Implementowane za pomocą Django REST Framework, umożliwia integrację zewnętrznych aplikacji oraz mobilnych klientów. API oferuje endpointy do zarządzania wątkami, postami oraz użytkownikami.

3 Wyszczególnione wdrożone zagadnienia kwalifikacyjne

1. **Framework MVC** (Wykład VI): W projekcie wykorzystano framework Django, który bazuje na architekturze Model-View-Controller (MVC). Model zarządza logiką danych, View odpowiada za prezentację, a Controller obsługuje interakcje użytkownika.

2. **Framework CSS** (Wykład II): Do stylizacji aplikacji użyto Bootstrap 4, który jest popularnym frameworkiem CSS. Bootstrap ułatwia tworzenie responsywnych i atrakcyjnych wizualnie interfejsów użytkownika.

3. **Baza danych** (Wykład VIII): W projekcie wykorzystano bazę danych MySQL do przechowywania informacji o użytkownikach, wątkach i postach. Baza danych jest zarządzana za pomocą ORM Django, co upraszcza operacje na danych.

4. **Dependency manager** (Wykład III): Zarządzanie zależnościami realizowane jest za pomocą pliku 'requirements.txt', który zawiera wszystkie wymagane biblioteki Pythona. Użycie narzędzia 'pip' umożliwia łatwe instalowanie zależności.

5. **HTML** (Wykład II): Szkielet aplikacji internetowej został stworzony przy użyciu języka HTML. Struktura dokumentu HTML umożliwia organizację treści i elementów interfejsu użytkownika.

6. **CSS** (Wykład II): Stylizacja aplikacji internetowej została wykonana za pomocą CSS, z wykorzystaniem Bootstrap. CSS umożliwia definiowanie wyglądu elementów HTML, takich jak kolory, czcionki i układ.

7. **JavaScript** (Wykład XI): JavaScript został użyty do dodania interaktywności do aplikacji internetowej. Skrypty JavaScript obsługują dynamiczne zmiany w interfejsie, takie jak walidacja formularzy i zmiana języka interfejsu.

8. **Routing** (Wykład V): W projekcie zaimplementowano routing za pomocą systemu URL Django, co umożliwia tworzenie tzw. pretty URLs. Każda strona ma przyjazny dla użytkownika adres URL, co poprawia nawigację po stronie.

9. **ORM** (Wykład IX): W projekcie wykorzystano mapowanie obiektowo-relacyjne (ORM) Django do zarządzania bazą danych. ORM umożliwia operacje na danych przy użyciu obiektów Pythona, co upraszcza dostęp do bazy danych.

10. **Uwierzytelnianie** (Wykład X): Zaimplementowano mechanizmy uwierzytelnienia użytkowników, takie jak rejestracja, logowanie i wylogowanie. Proces uwierzytelniania jest zabezpieczony za pomocą hashowania haseł.

11. **Lokalizacja** (Wykład XV): Aplikacja umożliwia przełączanie języka interfejsu, co jest realizowane za pomocą mechanizmów lokalizacji Django. Użytkownicy mogą wybrać język interfejsu zgodnie ze swoimi preferencjami.

12. **Mailing** (Wykład XVII): Aplikacja obsługuje wysyłanie mejli, na przykład do potwierdzania rejestracji. Mailing jest realizowany za pomocą biblioteki Django Email, która umożliwia wysyłanie wiadomości e-mail z aplikacji.

13. **Formularze** (Wykład II): Przesyłanie danych do aplikacji odbywa się przez formularze, które są walidowane zarówno po stronie klienta, jak i serwera. Formularze umożliwiają użytkownikom interakcję z aplikacją i przesyłanie danych.

14. **Konsumpcja API** (Wykład XIII): Aplikacja korzysta z zewnętrznego API, takiego jak WeatherApi, do pobierania danych o rekomendacjach anime. Konsumpcja API jest realizowana za pomocą żądań HTTP.

15. **Publikacja API** (Wykład XI): Wystawiono własne API za pomocą Django REST Framework. API umożliwia zewnętrznym aplikacjom dostęp do danych o wątkach na forum.

16. **RWD** (Wykład XII): Frontend aplikacji jest responsywny dzięki zastosowaniu Bootstrap. Responsywny design zapewnia poprawne wyświetlanie aplikacji na różnych urządzeniach, takich jak komputery, tablety i smartfony.

17. **Logger** (Wykład XV): Zaimplementowano logowanie akcji w systemie, co umożliwia śledzenie aktywności użytkowników i diagnozowanie problemów. Logi są przechowywane w bazie danych i mogą być analizowane przez administratorów.

4 Instrukcja lokalnego i zdalnego uruchomienia systemu

4.1 Lokalna instalacja

1. Sklonuj repozytorium projektu:

```
git clone https://github.com/dominog125/PPSI-Project.git
```

2. Utwórz i aktywuj wirtualne środowisko:

```
python -m venv venv
source venv/bin/activate  # Na Windows: venv\Scripts\activate
```

3. Zainstaluj zależności:

```
pip install -r requirements.txt
```

4. Skonfiguruj bazę danych:

   - Upewnij się, że MySQL jest zainstalowany i uruchomiony.
   - Utwórz bazę danych:

```
CREATE DATABASE anime_forum CHARACTER SET UTF8;
```

   - Skonfiguruj połączenie do bazy danych w settings.py.

5. Wykonaj migracje bazy danych:

```
python manage.py makemigrations
python manage.py migrate
```

6. Uruchom serwer:

```
python manage.py runserver
```

5 Wnioski projektowe

Projekt został zrealizowany zgodnie z wymaganiami. Implementacja funkcji takich jak cache'owanie, API, międzynarodowość oraz logowanie akcji użytkowników znacznie poprawiła wydajność i funkcjonalność systemu.

Projekt dostarczył cenne doświadczenie w pracy z Django, REST Framework, MySQL i innymi technologiami, które będą przydatne w przyszłych projektach.