



Politechnika Wrocławska

Wydział Informatyki i Zarządzania
kierunek studiów: Informatyka

Praca dyplomowa - inżynierska

Forum dyskusyjne

Paweł Bernecki

słowa kluczowe:

forum

aplikacja webowa

Django Rest Framework, Vue.js

krótkie streszczenie:

Wstęp do niniejszej pracy obejmuje wprowadzenie do tematyki forów dyskusyjnych, oraz omówienie celu i zakresu pracy. Następnie pokazane zostały istniejące na rynku rozwiązania i przedstawiona została specyfikacja wymagań. W dalszej części pracy opisany został proces wytwarzania oprogramowania, a także zestawienie użytych narzędzi i technologii.

opiekun pracy dyplomowej
	<i>Tytuł/stopień naukowy/imię i nazwisko</i>	<i>ocena</i>	<i>podpis</i>
Ostateczna ocena za pracę dyplomową			
Przewodniczący Komisji egzaminu dyplomowego
	<i>Tytuł/stopień naukowy/imię i nazwisko</i>	<i>ocena</i>	<i>podpis</i>

Do celów archiwalnych pracę dyplomową zakwalifikowano do:*

a) kategorii A (akta wieczyste)

b) kategorii BE 50 (po 50 latach podlegające ekspertyzie)

* niepotrzebne skreślić

pieczęćka wydziałowa

Wrocław 2019

Streszczenie

Celem niniejszej pracy jest projekt i implementacja systemu umożliwiającego szybkie utworzenie spersonalizowanego forum. Efektem pracy jest aplikacja webowa zaimplementowana przy użyciu platform programistycznych Vue.js oraz Django Rest Framework, zgodna z założeniami idei Single-page Application i stylem architektonicznym REST.

Praca została podzielona na sześć rozdziałów, z których pierwszym jest wstęp, obejmujący wprowadzenie do tematyki forów dyskusyjnych oraz omówienie celu i zakresu pracy. W drugim rozdziale pokazane zostały istniejące na rynku rozwiązania używane jako inspiracje przy tworzeniu pracy. Trzeci rozdział obejmuje specyfikację wymagań na tworzone oprogramowanie, między innymi: słownik pojęć, opis problemu oraz zdefiniowane przypadki użycia. W czwartym rozdziale uwaga poświęcona zostaje projektowi aplikacji i elementom takim jak baza danych czy prototypy interfejsów. Piąty rozdział opisuje etap implementacji, wyszczególniając użyte technologie oraz narzędzia, a także opisując wybrane zagadnienia programistyczne związane z wytwarzaną aplikacją. Szósty rozdział poświęcony został etapowi testowania wytworzonego systemu.

Pracę zamyka strona poświęcona podsumowaniu, bibliografia oraz spis tabel, rysunków i listingów kodu.

Abstract

The goal of this thesis is design and implementation of a system capable of creating a personalized message board. The result of this thesis is a web application implemented using Vue.js and Django Rest Framework, consistent with the idea of Single-page Applications and REST software architectural style.

This thesis has been divided into six chapters, of which the first one is the preface, consisting of an introduction to the subject of message boards and a description of the goal and scope of the thesis. The second chapter features solutions existing on the market, used as inspirations in the process of creating the software. The third chapter includes a software requirements specification, complete with a conceptual dictionary, description of the problem and defined use cases. In the fourth chapter the attention is devoted to the project of the application, namely to elements such as database and user interface mockups. The fifth chapter describes the process of implementation, specifying the technologies and tools used, as well as highlighting selected interesting programming issues linked with the produced software. The sixth chapter has been devoted to the testing stage of the developed application.

The thesis is completed with a summary and a bibliography, as well as lists of images, tables and code snippets used through the text.

Spis treści

1. Wstęp.....	3
1.1. Wprowadzenie do tematyki, motywacja	3
1.2. Cel i zakres pracy	3
2. Przegląd literatury w zakresie tematyki pracy	4
2.1. Przegląd istniejących rozwiązań	4
2.1.1. Reddit	4
2.1.2. StackOverflow	4
2.1.3. 4chan/4channel	5
2.2. Podsumowanie	5
3. Specyfikacja wymagań.....	7
3.1. Słownik pojęć.....	7
3.2. Opis problemu, wizja systemu	7
3.2.1. Użytkownicy.....	8
3.2.2. Historyjki użytkowników	8
3.3. Wymagania funkcjonalne i нефункционалне	9
3.4. Przypadki użycia	9
3.4.1. Opis przypadków użycia	10
3.4.2. Diagram przypadków użycia	15
3.5. Reguły biznesowe	16
4. Projekt aplikacji.....	17
4.1. Prototypy interfejsu użytkownika	17
4.2. Architektura aplikacji	21
4.3. Baza danych	22
5. Implementacja	23
5.1. Przegląd technologii	23
5.1.1. Django	23
5.1.2. Django Rest Framework.....	23
5.1.3. Vue.js.....	24
5.1.4. Inne warte uwagi technologie.....	24
5.2. Narzędzia programistyczne	24
5.3. Opis implementacji funkcjonalności	25
5.3.1. Punkty końcowe API.....	25
5.3.2. Obsługa plików graficznych w systemie	27

5.3.3. Autoryzacja.....	27
5.3.4. Panel administracyjny	28
5.4. Prezentacja interfejsu aplikacji.....	29
6. Testowanie	30
6.1. Testy jednostkowe.....	30
6.2. Testy systemowe	31
7. Podsumowanie	32
8. Bibliografia.....	33
9. Załączniki	34

1. Wstęp

1.1. Cel i zakres pracy

Celem tworzonej pracy jest projekt i implementacja systemu umożliwiającego szybkie utworzenie spersonalizowanego forum. Oprogramowanie jest skierowane do osób które chciałyby uruchomić własne forum, jak i do osób będących potencjalnymi użytkownikami. Oprogramowanie ma umożliwiać tworzenie tablic, postów i komentarzy, publikowania treści przez zalogowanych użytkowników i możliwość przesyłania plików graficznych. Aplikacja powinna także oddawać w ręce administratora możliwość dodawania, edycji oraz usuwania tablic, postów i komentarzy.

Zakres pracy obejmuje przegląd literatury, projekt oraz implementację aplikacji zapewniającej standardowe mechanizmy forum dyskusyjnego, strukturę tablica-post-komentarz, możliwość crosspostingu, oraz interfejs nastawiony na prezentowanie plików graficznych.

W głównej części pracy skupię się na opisaniu realizowanego procesu wytwarzania oprogramowania, zgodnego z modelem kaskadowym: Model ten zakłada wykonanie w ustalonej kolejności wcześniej zaplanowanych etapów. Określenie wymagań zrealizowane zostanie poprzez przygotowanie opisu problemu i wizji systemu, a następnie wymagań na system. W ramach etapu analizy przygotowany zostanie diagram przypadków użycia, opisy przypadków oraz makiety interfejsu. W etapie projektowania aplikacji zaprojektowana zostanie architektura aplikacji oraz baza danych wraz z modelem fizycznym. W ramach etapu implementacji przedstawiony zostanie opis zastosowanych rozwiązań programistycznych, interfejsu użytkownika oraz użytych narzędzi programistycznych. Etap testowania złożony będzie z opisu przeprowadzonych testów jednostkowych, integracyjnych oraz systemowych.

1.2. Wprowadzenie do tematyki, motywacja

W dobie Internetu, tradycyjne środki komunikacji i wymiany informacji zostały w znacznej mierze zastąpione przez portale społecznościowe, komunikatory internetowe i wszelkiego rodzaju fora. Począwszy od prostych grup dyskusyjnych, poprzez fora obrazkowe aż po serwisy skupione na zbieraniu linków do informacji opublikowanych w Internecie, mechanizm forum jest jednym z najpopularniejszych archetypów stron internetowych.

Idea forum jest wyjątkowo prosta, co daje twórcom możliwości rozwijania schematu i tworzenia nowych iteracji oprogramowania, integrujących nowe funkcjonalności, mechanizmy i sposoby użytkowania. W pewnych aspektach bazowa idea forum jako grupy dyskusyjnej ogranicza kreatywność twórców, ale nawet nie ingerując w podstawowe mechanizmy działania takiej strony, trudno wyobrazić sobie limit modyfikacji, którym można ją poddać aby poprawić jakość doświadczeń odwiedzających ją użytkowników.

W ramach tworzonej pracy zaprojektowana oraz zaimplementowana została aplikacja webowa bazująca na idei forum dyskusyjnego, łącząca w sobie także wybrane cechy forów obrazkowych oraz wyżej wymienionych serwisów skupionych na agregowaniu linków do informacji w Internecie.

Główną motywacją do podjęcia takiego tematu pracy inżynierskiej jest duży stopień dowolności w kwestiach wyglądu i działania wytwarzanej aplikacji. Integrując pomysły z różnych rodzajów forów internetowych, mam możliwość wyboru tych które uważam za wartościowe i odrzucenia tych nieprzydatnych. To przedsięwzięcie okazało się również dobrą okazją do poznania i opanowania dwóch interesujących platform programistycznych – Django Rest Framework bazujący na Django oraz Vue.js.

2. Przegląd literatury w zakresie tematyki pracy

2.1. Przegląd istniejących rozwiązań

Mimo iż wszystkie fora z założenia służą do wymiany informacji i poglądów, wiele z najpopularniejszych i najbardziej znanych serwisów tego typu posiada wyjątkowe dla siebie funkcjonalności, wyróżniające je spośród reszty. Poniżej opisane zostaną najbardziej znane rozwiązania istniejące na rynku. Są to fora, które proponują ciekawe pomysły na polepszenie doświadczeń użytkownika i zostały po części wykorzystane jako inspiracja dla wytwarzanej aplikacji.

2.1.1. Reddit

Reddit jest aktualnie najpopularniejszym forum internetowym na świecie. Założony w 2005 roku, jest rozwijany już od 13 lat i zatrudnia ponad 200 osób. Strona pozwala na przeglądanie publikowanej na niej treści przez odwiedzających bez konieczności logowania czy rejestracji. Zalogowani użytkownicy uzyskują możliwość tworzenia społeczności, postów czy komentarzy oraz wchodzenia w interakcję z innymi użytkownikami, jak i z tworzoną przez nich treścią. Reddit zapewnia swoim użytkownikom możliwość prowadzenia dyskusji na dowolne tematy, ale głównym rodzajem publikowanej treści są linki i wszelkiego rodzaju media, udostępniane przez użytkowników.

Każdy post tworzony jest na wybranym „Subreddicie”, czyli swoistej tablicy tematycznej, tworzonej i moderowanej przez związanych z nią użytkowników. Post zbudowany jest z tytułu, treści – linków, mediów lub tekstu, oraz wyniku – wartości punkowej w systemie dziesiętnym. Po opublikowaniu postu, każdy widzący go użytkownik, może zdecydować się zmodyfikować jego wynik o jeden, dodatni lub ujemny punkt. Taki sposób oceniania treści zapewnia zwiększoną jakość doświadczeń ludzi odwiedzających stronę, ponieważ możemy upewnić się, że w pierwszej kolejności zobaczą oni posty które podobały się największej liczbie użytkowników. Jedną z poważniejszych wad takiego sposobu zarządzania treścią, jest drastycznie zmniejszona waga nowo tworzonych postów w porównaniu do tych o wysokich wynikach. Dlatego Reddit udostępnia kilka opcji sortowania treści:

- Według liczby punktów, z ustalonego przedziału czasu – najlepsze posty
- Według prędkości z jaką post zyskuje punkty – najszybciej rozwijające się posty
- Według czasu dodania – ostatnio dodane posty
- Według stosunku dodatnich i ujemnych punktów – najbardziej kontrowersyjne posty

Podsumowując, Reddit wprowadził dużo mechanizmów, które wpłynęły pozytywnie na jakość doświadczeń użytkownika odwiedzającego stronę, niezależnie od tego czy jest zalogowany czy nie.

2.1.2. StackOverflow

StackOverflow jest forum utworzonym w roku 2008 przez Jeffa Atwooda i Joela Spolsky’ego, należącym do sieci forów tematycznych Stack Exchange Network. Podstawowym założeniem serwisu StackOverflow jest utworzenie platformy pozwalającej użytkownikom na swobodną wymianę informacji. Publikowane przez użytkowników posty zostają opatrzone odpowiednimi etykietami kategoryzującymi ich treść i muszą być zgodne z narzuconym zestawem zasad i ograniczeń. Zgodnie z ideą serwisu, tworzone posty muszą mieć charakter pytania, a publikowane w odpowiedzi na nie komentarze powinny być próbami udzielenia odpowiedzi na zadane w poście pytanie.

Treść publikowana na stronie może być edytowana przez użytkowników serwisu w celu poprawy błędów lub zwiększeniu wartości merytorycznej postu czy komentarza. Dodatkowo strona jest moderowana przez liczną grupę moderatorów, co w połączeniu z aktywną bazą użytkowników udzielających odpowiedzi na zadawane na stronie pytania, zmienia główną funkcjonalność serwisu z forum dyskusyjnego na internetową bazę wiedzy specjalistycznej.

Dane statystyczne z roku 2013 pokazały, że aż 75% użytkowników zadało tylko jedno pytanie oraz aż 65% użytkowników udzieliło tylko jednej odpowiedzi [7]. Dane te w znacznym stopniu potwierdzają teorię, mówiącą o tym, że przeciętny użytkownik strony StackOverflow jest zainteresowany tylko uzyskaniem odpowiedzi na interesujące go pytanie, a bycie częścią aktywnej społeczności użytkowników nie ma dla niego znaczenia.

2.1.3. 4chan/4channel

Serwis 4channel, do niedawna istniejący w sieci pod domeną 4chan.org jest najpopularniejszym na świecie anglojęzycznym forum obrazkowym, wzorowanym na powstałym w 1999 japońskim forum 2channel. Z powodu bariery językowej, 2channel został prześcignięty przez 4channel, który stał się szybko międzynarodowym symbolem anonimowej komunikacji wśród młodych ludzi.

Głównymi cechami wyróżniającymi fora z rodziny chan/channel jest anonimowość użytkowników, brak konieczności zakładania konta do tworzenia postów i komentarzy, tematyczne tablice oraz duży stopień wykorzystania grafiki w komunikacji między użytkownikami. Użytkownik może dołączyć do każdego tworzonego postu i komentarza wybrane zdjęcie. Interfejs użytkownika również jest w znacznej mierze nastawiony do oglądanie zdjęć i ułatwia użytkownikowi ich przeglądanie.

Tego typu forum zostało nazwane forum obrazkowym, ale jednocześnie pozostaje także rodzajem forum dyskusyjnego. Charakterystyczna struktura Tablica/Post/Komentarz pozwala na łatwe prowadzenie rozmów przez użytkowników, ale z racji anonimowości użytkowników, konwersacje mogą przebiegać w zawiły sposób, przy obecności większej liczbie aktywnych rozmówców. W tym celu na tablicach których tematyka wspiera prowadzenie debat, wprowadzone zostały tymczasowe identyfikatory, unikalne dla danego IP w danym poście. Takie rozwiązanie ułatwiło rozmówcom wzajemne rozpoznawanie się podczas konwersacji, jednocześnie pozostawiając im anonimowość.

Porównując fora anonimowe do tych, gdzie treść tworzą zalogowani użytkownicy, można dojść łatwo do wniosku, że wartość merytoryczna czy rozrywkowa tworzonych postów jest znacząco większa w tych, gdzie twórcy postów narażeni są na krytykę.

2.2. Podsumowanie

Wymienione rozwiązania prezentują odmienne podejścia do archetypu forów dyskusyjnych i posiadają kilka mechanizmów, które warto wykorzystać w tworzonej w ramach niniejszej pracy aplikacji.

Reddit posiada mechanizm udostępniania postów pomiędzy tablicami, który pozwala na publikowanie treści dotyczącej kilku różnych tematów na odpowiednich tematycznych tablicach. Takie rozwiązanie wciąż wymaga jednak od użytkownika wybrania głównej tablicy, na której osadzony zostanie tworzony post. W celu wykorzystania korzyści prezentowanych przez stojącą za tym mechanizmem ideę, w rozwijanej aplikacji udostępniona zostanie możliwość przypisania wielu tablic do jednego postu.

StackOverflow pozwala użytkownikom edytować posty i komentarze publikowane na stronie przez innych użytkowników. Biorąc pod uwagę naukowo-informacyjny charakter tego serwisu i wysoki poziom moderacji treści zamieszczanej na stronie, takie rozwiązanie sprawdza się tam dobrze. Mając na uwadze brak informacji o przyszłym charakterze forum i bazie jego

użytkowników, w tworzonej aplikacji oddajemy możliwość edytowania postów i komentarzy administratorowi.

Komunikacja między użytkownikami w serwisie 4chan/4channel realizowana jest nie tylko przy użyciu tekstu, ale także plików graficznych, załączonych w publikowanych postach i komentarzach. Jest to ściśle związane z wyglądem interfejsu oraz łatwością przeglądania z jego pomocą treści graficznych. Z tego powodu, układ interfejsu w projektowanej aplikacji posiada pewien stopień podobieństwa do tego działającego w serwisie 4chan/4channel.

3. Specyfikacja wymagań

Specyfikacja wymagań jest dokumentem opisującym wymagania na produkt. W tym dokumencie, klient zawiera informacje dotyczące charakterystyki pożądanego produktu. Poprawne przygotowanie specyfikacji wymagań, pozwala znacząco obniżyć szansę na wystąpienie nieporozumień wynikających z odmiennych wizji produktu według klienta i dostawcy, a tym samym zmniejsza całkowity koszt wytworzenia produktu.

3.1. Słownik pojęć

Na potrzeby projektu aplikacji sporządzony został słownik pojęć, zawierający pojęcia używane w pracy.

- Crossposting – publikowanie jednego postu na wielu tablicach.
- Forum dyskusyjne – aplikacja webowa pozwalająca na wymianę informacji i poglądów między użytkownikami
- Komentarz – Odpowiedź na post, publikowana przez użytkownika pod wybranym postem. Składa się z treści i opcjonalnego pliku graficznego.
- Moderacja – nadzorowanie treści w celu zapewnienia jej zgodności z zasadami obowiązującymi na forum.
- Post – Tworzona przez użytkownika wypowiedź, złożona z tytułu, treści i opcjonalnego pliku graficznego, pod którym użytkownicy mogą tworzyć komentarze.
- Tablica – Tworzony przez administratora, tematycznie ograniczony kontener przechowujący posty o odpowiedniej tematyce.
- Tematyka tablicy – Temat, któremu poświęcona jest konkretna tablica. Ustalany przez administratora przy tworzeniu tablicy.

3.2. Opis problemu, wizja systemu

Produkt skierowany jest do osób chcących w prosty sposób utworzyć spersonalizowane forum dyskusyjne.

Tworzona aplikacja ma za zadanie zapewnienie klientowi możliwości szybkiego uruchomienia w pełni konfigurowalnego forum dyskusyjnego, o schemacie zmodyfikowanym o wybrane mechanizmy zaczerpnięte z forów obrazkowych i serwisów skupionych na agregowaniu przez użytkowników linków z Internetu. Kluczowymi cechami tworzonego produktu są:

- Dopasowanie interfejsu do prezentowania zdjęć w wygodny sposób
- Możliwość Crosspostingu – zamieszczania jednego postu na wielu tablicach
- Przechowywanie zdjęć w systemie plików, z adresami przechowywanymi w bazie danych

3.2.1. Użytkownicy

Użytkownik, to osoba do której skierowany jest nasz system. Na bazie jego potrzeb, jesteśmy w stanie stworzyć wymagania na system, dlatego też jednoznaczne określenie użytkowników naszej aplikacji traktujemy priorytetowo. Wszystkich wymienionych użytkowników można zaliczyć do abstrakcyjnej grupy „Użytkowników systemu”.

Nazwa	Opis	Odpowiedzialności
Administrator	Osoba posiadająca konto użytkownika z prawem moderowania treści publikowanej na stronie, zarządzaniem tablicami, postami i komentarzami oraz dostępem do panelu administracyjnego	Przeglądanie tablic, postów i komentarzy, tworzenie, edycja i usuwanie tablic, postów oraz komentarzy.
Gość	Osoba nie posiadająca konta użytkownika	Zakładanie konta
Niezałogowany użytkownik	Osoba niezałogowana, posiadająca konto użytkownika	Logowanie się w aplikacji
Zalogowany użytkownik	Osoba posiadająca konto użytkownika z prawem tworzenia postów i komentarzy	Przeglądanie tablic, postów i komentarzy, tworzenie postów i komentarzy, edytowanie własnego profilu

Tabela 3.1. Użytkownicy aplikacji

3.2.2. Historyjki użytkowników

Historyjka użytkownika [6] jest krótkim tekstem opisującym konkretną potrzebę użytkownika. Sporządzenie zestawu historyjek ma na celu zebranie wymagań od użytkownika. Historyjki nie skupiają się na sposobie implementacji oczekiwanej przez użytkownika funkcjonalności, zamiast tego poświęcając uwagę jedynie na samej potrzebie. Pojedyncza historyjka powinna dotyczyć jednej potrzeby. W przypadku gdy opisywana potrzeba jest zbyt skomplikowana, historyjka powinna zostać rozbita.

Historyjki są powszechnie używane z racji ich prostego przekazu, zrozumiałego niezależnie od rodzaju wykształcenia lub branży. Częstą strukturą używaną do tworzenia historyjek jest :„Jako <użytkownik> potrzebuję <potrzeba>”.

Historyjka użytkownika
Jako dowolny użytkownik systemu potrzebuję możliwości nawigowania po treści publikowanej na stronie.
Jako gość potrzebuję możliwości założenia konta.
Jako niezałogowany użytkownik potrzebuję możliwości zalogowania się na swoje konto.
Jako zalogowany użytkownik potrzebuję możliwości tworzenia postów.
Jako zalogowany użytkownik potrzebuję możliwości tworzenia komentarzy.
Jako zalogowany użytkownik potrzebuję możliwości załączania zdjęć do publikowanych postów i komentarzy.
Jako administrator potrzebuję możliwości tworzenia tablic.
Jako administrator potrzebuję możliwości moderowania treści publikowaną przez użytkowników .
Jako administrator potrzebuję możliwości edycji informacji o tablicach.

Tabela 3.2. Historyjki użytkowników

3.3. Wymagania funkcjonalne i нефункционалне

Wymagania funkcjonalne [6] to informacje mówiące o oczekiwanych możliwościach tworzonej aplikacji. Dotyczą one funkcjonalności udostępnionych przez aplikację i są tworzone na podstawie wymagań użytkowników.

Wymagania funkcjonalne
FR1. Aplikacja umożliwia rejestrację konta użytkownika
FR2. Aplikacja umożliwia logowanie się na konto użytkownika za pomocą loginu i hasła
FR3. Aplikacja umożliwia tworzenie, przeglądanie, edycję i usuwanie tablic
FR4. Aplikacja umożliwia tworzenie, przeglądanie, edycję i usuwanie postów
FR5. Aplikacja umożliwia tworzenie, przeglądanie, edycję i usuwanie komentarzy
FR6. Aplikacja umożliwia przysyłanie plików graficznych
FR7. Aplikacja umożliwia powiązanie postu z wieloma tablicami
FR8. Tworzenie postów i komentarzy może być realizowane tylko przez zalogowanych użytkowników i administratora.
FR9. Tworzenie tablic może być realizowane tylko przez administratora.

Tabela 3.3 Wymagania funkcjonalne

Wymagania нефункционалне są informacjami o ograniczeniach aplikacji i dotyczą całości tworzonego produktu.

Wymagania нефункционалне
NFR1. Aplikacja jest dostępna przez Internet.
NFR2. Aplikacja działa poprawnie na przeglądarkach Firefox, wersja 64 oraz Chrome, wersja 71
NFR.3 Rodzaj systemu operacyjnego użytkowników nie powinien mieć wpływu na poprawne funkcjonowanie aplikacji.

Tabela 3.4. Wymagania нефункционалне

3.4. Przypadki użycia

Przypadek użycia [6] jest opisem serii interakcji pomiędzy aktorami. Aktorami mogą być tutaj użytkownicy lub części tworzonej aplikacji. Często przypadek użycia obejmuje szerszy zakres niż historyjka użytkownika.

Typowym schematem opisu przypadku użycia jest: Tytuł, główny scenariusz, scenariusz alternatywny. W głównym scenariuszu mówimy o sekwencji kroków opisującej standardowy przebieg wykonywanej czynności, podczas gdy w scenariuszu alternatywnym wspominamy o możliwych odstępstwach od standardowego przebiegu danej czynności.

3.4.1. Opis przypadków użycia

Identyfikator	P001
Nazwa	Rejestracja
Cel	Założenie konta użytkownika w systemie
Aktor	Gość
Scenariusz	<ol style="list-style-type: none">1. Gość chce dokonać rejestracji2. Aplikacja wyświetla formularz tworzenia konta3. Gość uzupełnia formularz swoimi danymi4. System stwierdza poprawność danych i przekierowuje na stronę główną
Scenariusz alternatywny	Wejście z punktu 4: <ol style="list-style-type: none">1. System stwierdza niepoprawność danych i zwraca komunikat o błędzie

Tabela 3.5. Opis przypadku użycia P001

Identyfikator	P002
Nazwa	Logowanie
Cel	Zalogowanie się na konto użytkownika w systemie
Aktor	Niezalogowany użytkownik
Scenariusz	<ol style="list-style-type: none">1. Niezalogowany użytkownik chce się zalogować2. Aplikacja wyświetla formularz logowania3. Niezalogowany użytkownik uzupełnia formularz swoimi danymi4. System stwierdza poprawność danych i przekierowuje na stronę główną
Scenariusz alternatywny	Wejście z punktu 4: <ol style="list-style-type: none">1. System stwierdza niepoprawność danych i zwraca komunikat o błędzie

Tabela 3.6. Opis przypadku użycia P002

Identyfikator	P003
Nazwa	Przeglądanie tablic
Cel	Zapoznanie się ze zbiorem istniejących tablic
Aktor	Użytkownik systemu
Scenariusz	<ol style="list-style-type: none">1. Użytkownik chce zapoznać się ze zbiorem istniejących tablic2. Aplikacja wyświetla listę istniejących tablic
Scenariusz alternatywny	Wejście z punktu 2: <ol style="list-style-type: none">1. P004 – Przeglądanie postów na tablicy

Tabela 3.7. Opis przypadku użycia P003

Identyfikator	P004
Nazwa	Przeglądanie postów na tablicy
Cel	Zapoznanie się ze zbiorem postów na danej tablicy
Aktor	Użytkownik systemu
Scenariusz	<ol style="list-style-type: none"> 1. Użytkownik chce zapoznać się ze zbiorem postów na danej tablicy 2. Użytkownik wybiera interesującą go tablicę 3. Aplikacja wyświetla listę postów należących do wybranej tablicy
Scenariusz alternatywny	Wejście z punktu 3: <ol style="list-style-type: none"> 1. P005 – Przeglądanie komentarzy postu

Tabela 3.8. Opis przypadku użycia P004

Identyfikator	P005
Nazwa	Przeglądanie komentarzy postu
Cel	Zapoznanie się ze zbiorem komentarzy dotyczących danego postu
Aktor	Użytkownik systemu
Scenariusz	<ol style="list-style-type: none"> 1. Użytkownik chce zapoznać się ze zbiorem komentarzy dotyczących danego postu 2. Użytkownik wybiera interesujący go post 3. Aplikacja wyświetla listę komentarzy dotyczących wybranego postu

Tabela 3.9. Opis przypadku użycia P005

Identyfikator	P006
Nazwa	Tworzenie postu
Cel	Utworzenie nowego postu na wybranej tablicy
Aktor	Zalogowany użytkownik
Scenariusz	<ol style="list-style-type: none"> 1. Zalogowany użytkownik chce utworzyć nowy post 2. Aplikacja wyświetla formularz tworzenia nowego postu 3. Zalogowany użytkownik uzupełnia formularz odpowiednią treścią 4. System stwierdza poprawność danych i przekierowuje do nowo utworzonego postu
Scenariusz alternatywny	Wyjście z punktu 4: <ol style="list-style-type: none"> 1. System stwierdza niepoprawność danych i zwraca komunikat o błędzie

Tabela 3.10. Opis przypadku użycia P006

Identyfikator	P007
Nazwa	Tworzenie komentarza
Cel	Utworzenie nowego komentarza pod wybranym postem
Aktor	Zalogowany użytkownik
Scenariusz	<ol style="list-style-type: none"> 1. Zalogowany użytkownik chce utworzyć nowy komentarz 2. Aplikacja wyświetla formularz tworzenia nowego komentarza 3. Zalogowany użytkownik uzupełnia formularz odpowiednią treścią 4. System stwierdza poprawność danych i przekierowuje do postu nowo utworzonego komentarza
Scenariusz alternatywny	<p>Wyjście z punktu 4:</p> <ol style="list-style-type: none"> 1. System stwierdza niepoprawność danych i zwraca komunikat o błędzie

Tabela 3.11. Opis przypadku użycia P007

Identyfikator	P008
Nazwa	Przeglądanie listy postów
Cel	Zapoznanie się ze zbiorem postów istniejących w systemie
Aktor	Administrator
Scenariusz	<ol style="list-style-type: none"> 1. Administrator chce zapoznać się ze zbiorem postów istniejących w systemie 2. Aplikacja wyświetla listę postów
Scenariusz alternatywny	<p>Wejście z punktu 2:</p> <ol style="list-style-type: none"> 1. P009 – Modyfikowanie postu

Tabela 3.12. Opis przypadku użycia P008

Identyfikator	P009
Nazwa	Modyfikowanie postu
Cel	Modyfikacja istniejącego postu
Aktor	Administrator
Scenariusz	<ol style="list-style-type: none"> 1. Administrator chce zmodyfikować istniejący post 2. System wyświetla formularz edycji postu 3. Administrator edytuje zawartość formularza odpowiednią treścią 4. System stwierdza poprawność danych i przekierowuje do widoku przeglądania postów
Scenariusz alternatywny	<p>Wyjście z punktu 4:</p> <ol style="list-style-type: none"> 1. System stwierdza niepoprawność danych i zwraca komunikat o błędzie

Tabela 3.13. Opis przypadku użycia P009

Identyfikator	P010
Nazwa	Przeglądanie listy komentarzy
Cel	Zapoznanie się ze zbiorem komentarzy istniejących w systemie
Aktor	Administrator
Scenariusz	1. Administrator chce zapoznać się ze zbiorem komentarzy istniejących w systemie 2. Aplikacja wyświetla listę komentarzy
Scenariusz alternatywny	Wejście z punktu 2: 1. P011 – Modyfikowanie komentarza

Tabela 3.14. Opis przypadku użycia P010

Identyfikator	P011
Nazwa	Modyfikowanie komentarza
Cel	Modyfikacja istniejącego komentarza
Aktor	Administrator
Scenariusz	1. Administrator chce zmodyfikować istniejący komentarz 2. System wyświetla formularz edycji komentarza 3. Administrator edytuje zawartość formularza odpowiednią treścią 4. System stwierdza poprawność danych i przekierowuje do widoku przeglądania listy komentarzy
Scenariusz alternatywny	Wyjście z punktu 4: 1. System stwierdza niepoprawność danych i zwraca komunikat o błędzie

Tabela 3.15. Opis przypadku użycia P011

Identyfikator	P012
Nazwa	Przeglądanie listy tablic
Cel	Zapoznanie się ze zbiorem tablic istniejących w systemie
Aktor	Administrator
Scenariusz	1. Administrator chce zapoznać się ze zbiorem tablic istniejących w systemie 2. Aplikacja wyświetla listę tablic
Scenariusz alternatywny	Wejście z punktu 2: 1. P013 – Tworzenie tablicy 2. P014 – Modyfikowanie komentarza

Tabela 3.16. Opis przypadku użycia P012

Identyfikator	P013
Nazwa	Modyfikowanie tablicy
Cel	Modyfikacja istniejącej tablicy
Aktor	Administrator
Scenariusz	<ol style="list-style-type: none"> 1. Administrator chce zmodyfikować istniejącą tablicę 2. System wyświetla formularz edycji tablicy 3. Administrator edytuje zawartość formularza odpowiednią treścią 4. System stwierdza poprawność danych i przekierowuje do widoku przeglądania listy tablic
Scenariusz alternatywny	<p>Wyjście z punktu 4:</p> <ol style="list-style-type: none"> 1. System stwierdza niepoprawność danych i zwraca komunikat o błędzie

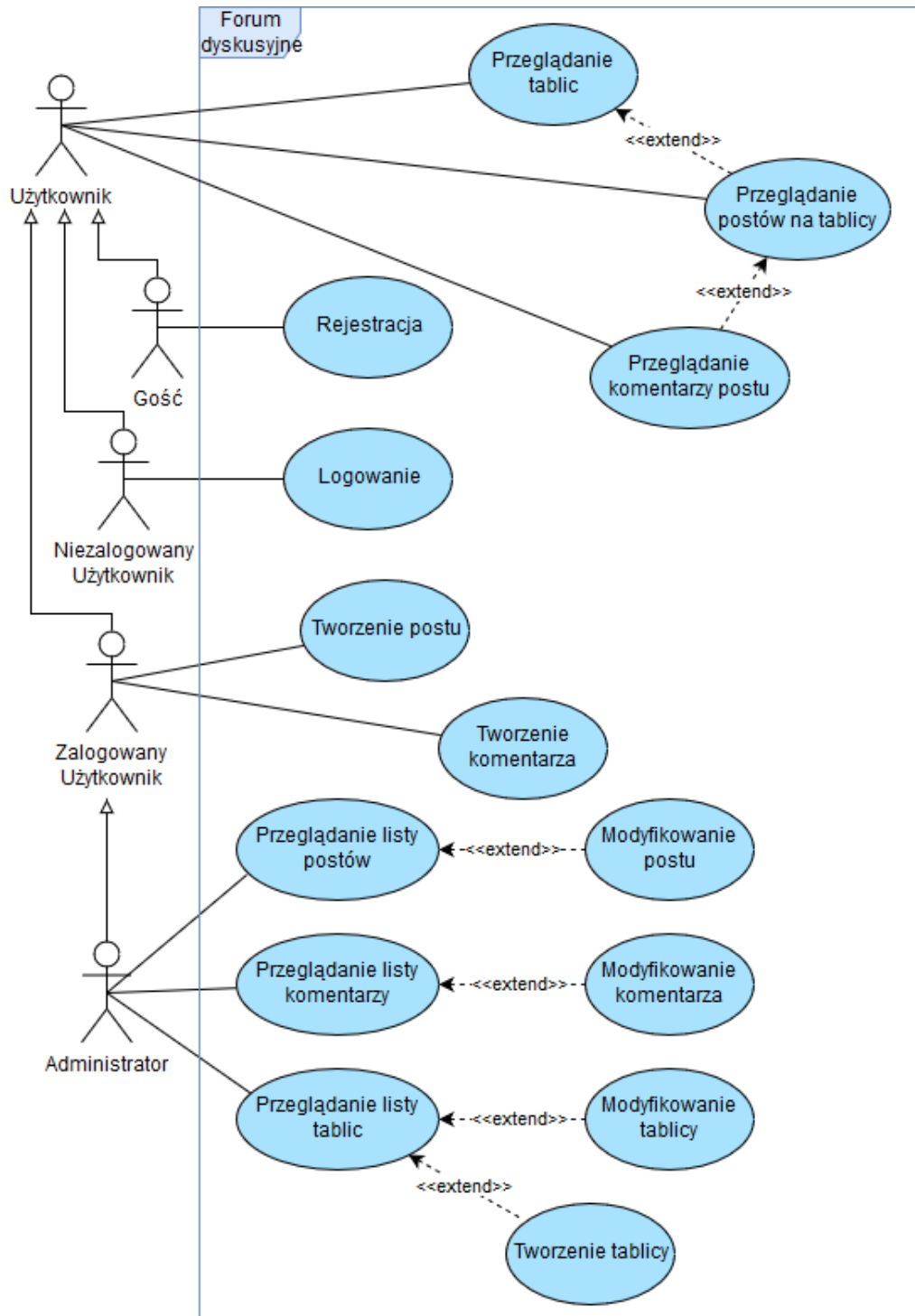
Tabela 3.17. Opis przypadku użycia P013

Identyfikator	P014
Nazwa	Tworzenie tablicy
Cel	Utworzenie nowej tablicy
Aktor	Administrator
Scenariusz	<ol style="list-style-type: none"> 1. Administrator chce utworzyć nową tablicę 2. System wyświetla formularz tworzenia tablicy 3. Administrator uzupełnia formularz odpowiednią treścią 4. System stwierdza poprawność danych i przekierowuje do widoku przeglądania listy tablic
Scenariusz alternatywny	<p>Wyjście z punktu 4:</p> <ol style="list-style-type: none"> 1. System stwierdza niepoprawność danych i zwraca komunikat o błędzie

Tabela 3.18. Opis przypadku użycia P014

3.4.2. Diagram przypadków użycia

Diagram przypadków użycia [6] reprezentuje interakcje użytkowników z systemem. Używając prostych rysunków reprezentujących aktorów i elips zawierających nazwy przypadków użycia, jesteśmy w stanie utworzyć czytelną reprezentację czynności wykonywanych przez użytkowników w aplikacji. Dodatkowo użyte mogą zostać połączenia typu <<extend>> w celu reprezentacji ciągów czynności wymagających specyficznych okoliczności.



Rys. 3.1. Diagram przypadków użycia

3.5. Reguły biznesowe

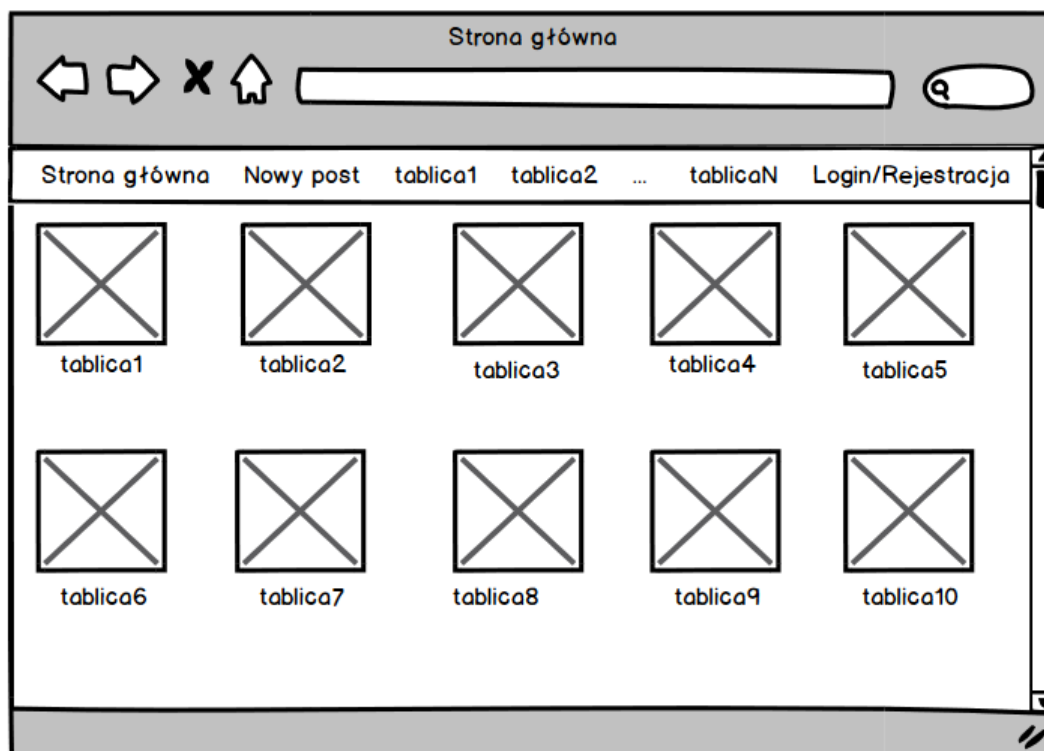
Na potrzeby projektu utworzony został zbiór reguł biznesowych, zapisanych w języku naturalnym.

- REG001. Zalogowany użytkownik jest osobą korzystającą z forum, posiadającą możliwość tworzenia i publikowania treści w postaci postów i komentarzy
- REG002. Tablica jest ograniczonym tematycznie zbiornikiem przechowującym opublikowane posty.
- REG003. Post jest połączeniem tytułu, treści i opcjonalnego pliku graficznego, opublikowanym przez użytkownika na wybranych tablicach
- REG004. Komentarz jest wypowiedzią utworzoną przez użytkownika pod postem, zawierającą treść i opcjonalny plik graficzny.
- REG005. Zalogowany użytkownik może publikować wiele postów.
- REG006. Zalogowany użytkownik może nie publikować żadnego postu.
- REG007. Post musi być publikowany przez użytkownika.
- REG008. Post może być utworzony przez co najwyżej jednego użytkownika.
- REG009. Zalogowany użytkownik może publikować wiele komentarzy.
- REG0010. Zalogowany użytkownik może nie publikować żadnego komentarza.
- REG0011. Komentarz musi być publikowany przez użytkownika.
- REG0012. Komentarz może być utworzony przez co najwyżej jednego użytkownika.
- REG0013. Post może mieć wiele komentarzy.
- REG0014. Post nie musi mieć żadnego komentarza.
- REG0015. Komentarz musi być przypisany do postu.
- REG0016. Komentarz może być przypisany do co najwyżej jednego postu.
- REG0017. Post musi być utworzony na tablicy.
- REG0018. Post może zostać opublikowany na dowolnej liczbie tablic.
- REG0019. Tablica może przechowywać wiele postów.
- REG0020. Tablica nie musi przechowywać żadnego postu.

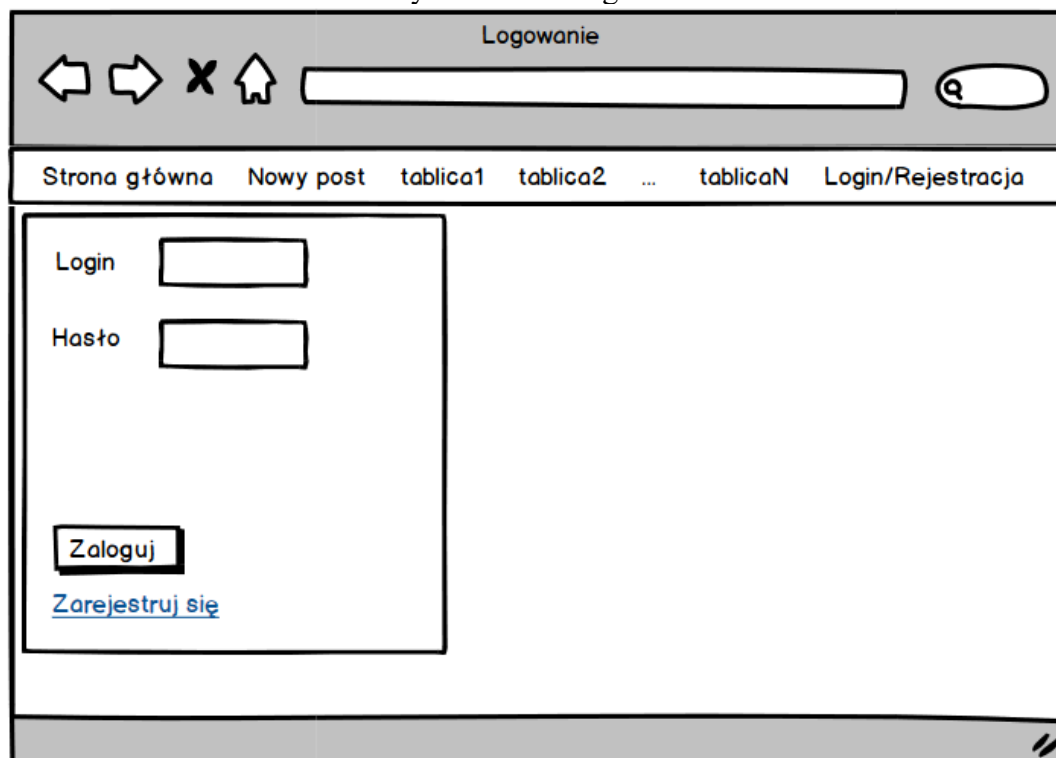
4. Projekt aplikacji

4.1. Prototypy interfejsu użytkownika

Prototypy interfejsu użytkownika są graficznymi makietami pozwalającymi na przedstawienie wyglądu przyszłego systemu. Tworzenie ich pozwala również na weryfikację zebranych wymagań w celu ich dopracowania.



Rys. 4.1. Strona główna



Rys. 4.2. Logowanie

←

→

✕

🏠

Rejestracja

Strona główna

Nowy post

tablica1

tablica2

...

tablicaN

Login/Rejestracja

Login

Hasło

Adres email

Zarejestruj

[Zaloguj się](#)

Rys. 4.3. Rejestracja

←

→

✕

🏠

Tworzenie nowego komentarza

Strona główna

Nowy post

tablica1


tablica2

...

tablicaN

Login/Rejestracja

Treść



Wybierz zdjęcie

Utwórz komentarz


Rys. 4.4. Tworzenie nowego komentarza

Tworzenie nowego postu

Strona główna Nowy post tablica1 tablica2 ... tablicaN Login/Rejestracja

Tytuł

Tresc





Wybierz zdjęcie

Tablice

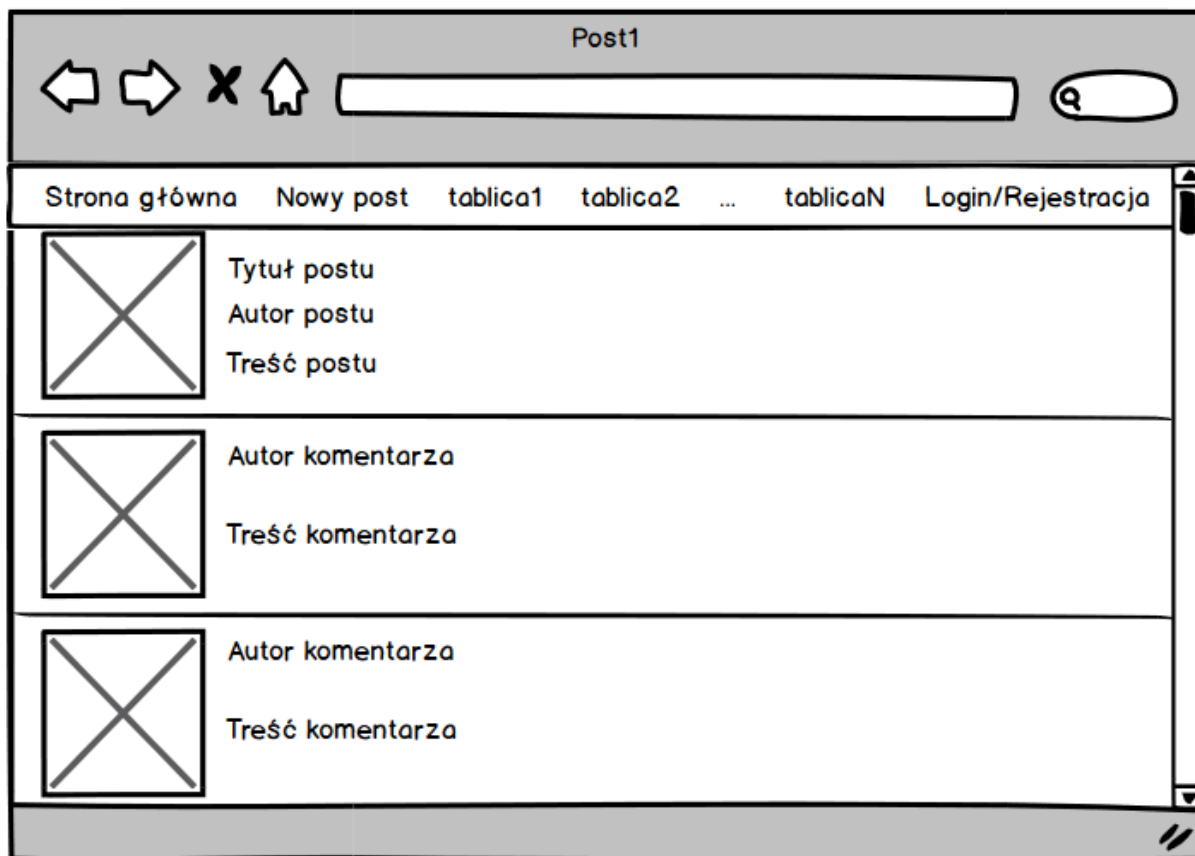
Rys. 4.5. Tworzenie nowego postu

Tablica1

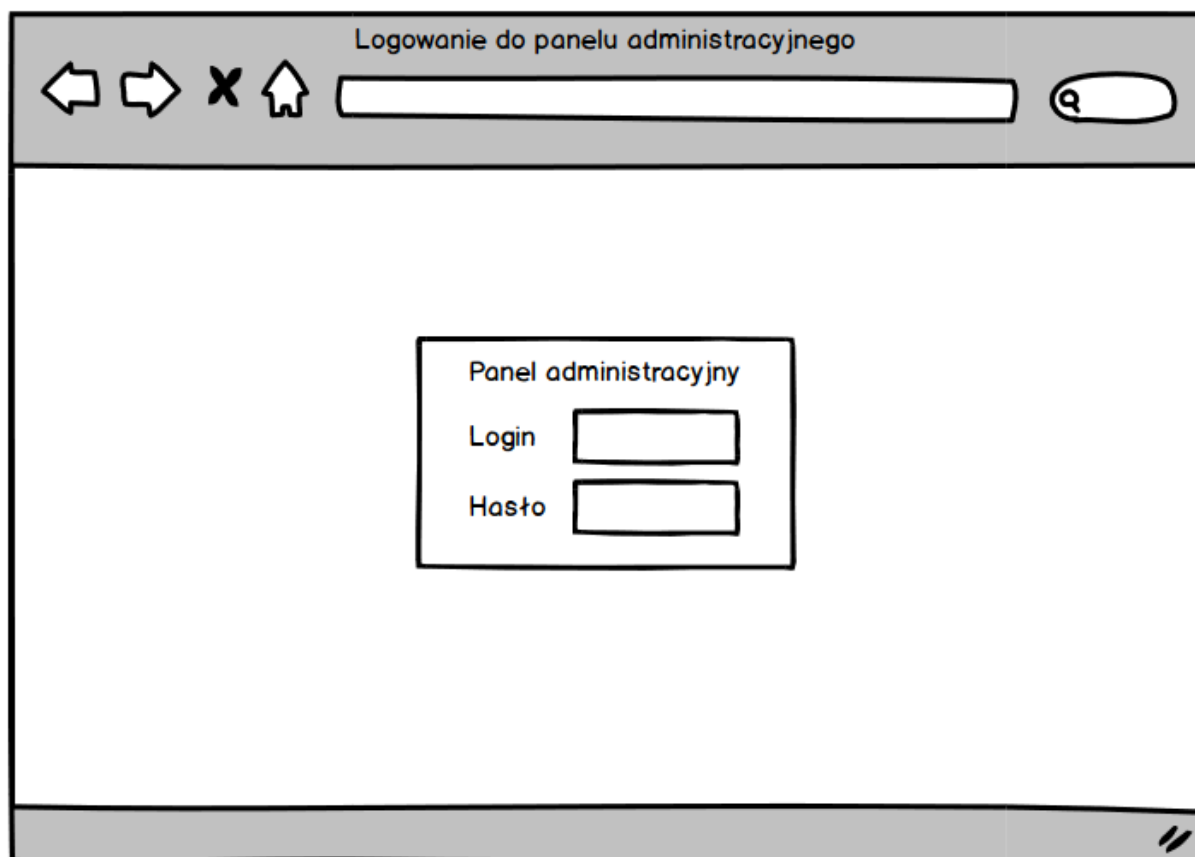
Strona główna Nowy post tablica1 tablica2 ... tablicaN Login/Rejestracja

Nazwa tablicy	
Opis tablicy	
	Tytuł postu Autor postu Treść postu
	Tytuł postu Autor postu Treść postu

Rys. 4.6. Przeglądanie postów



Rys. 4.7. Przeglądanie komentarzy



Rys.4.8. Okno logowania panelu administracyjnego

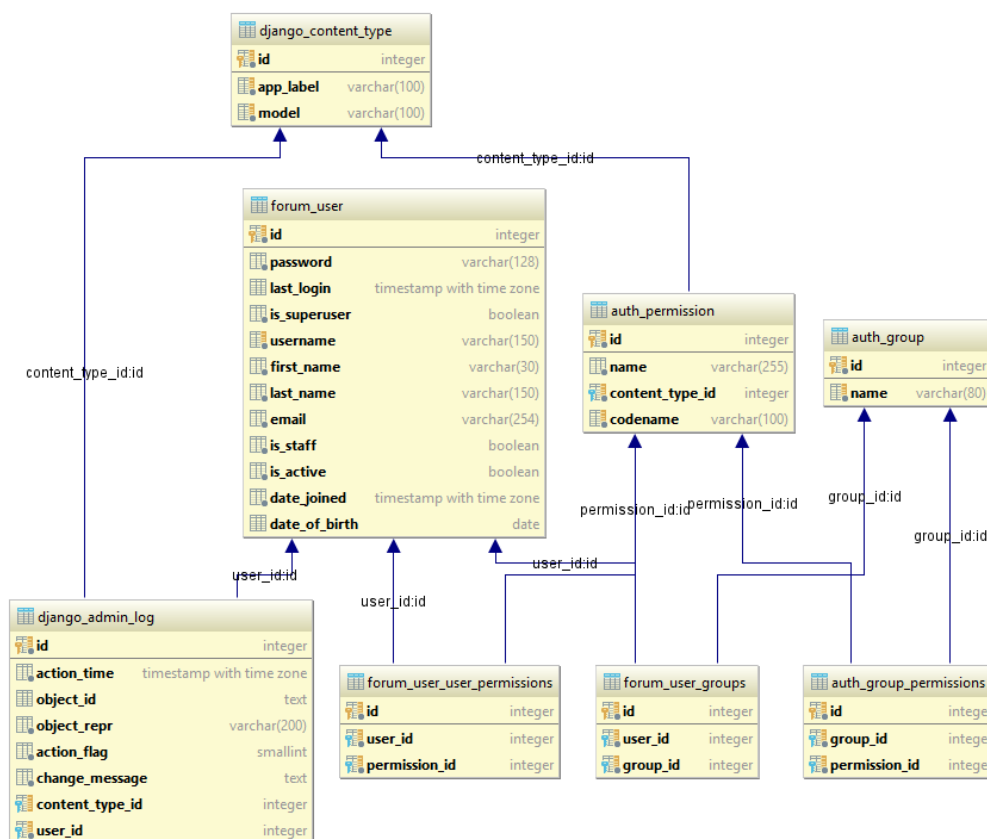
4.2. Architektura aplikacji

Przy tworzeniu opisywanej aplikacji wykorzystane zostały podejścia Single Page Application i stylu architektonicznego REST. W ten sposób wyodrębnione zostały dwa oddzielne elementy tworzonego forum: pisany przy użyciu Django i Django Rest Framework interfejs programowania aplikacji, oraz tworzony z użyciem platformy programistycznej Vue.js interfejs klienta działający pod postacią aplikacji webowej.

Całość zrealizowana jest zgodnie ze wzorcem architektonicznym MVVM, generując widoki w aplikacji używając komponentów Vue, operujących na danych otrzymywanych z serwera[1]. MVVM jest akronimem angielskiego wyrażenia Model-View-ViewModel. Ten wzorzec architektoniczny zakłada zastąpienie kontrolera z tradycyjnej architektury Model-View-Controller mechanizmem ViewModel. Jest on swojego rodzaju konwerterem wartości, to znaczy jest odpowiedzialny za zarządzanie obiektami otrzymywanymi od modelu.

4.3. Baza danych

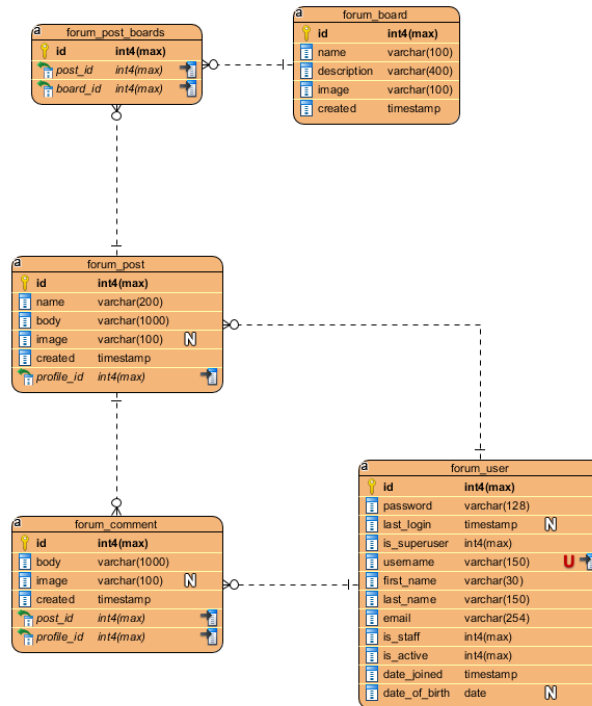
Platforma programistyczna Django pozwala programistom na posługiwanie się modelami – Pythonowymi obiektami, które reprezentują tradycyjne klasy w relacyjnej bazie danych [4]. Django udostępnia także gotowy zestaw klas dotyczących użytkowników systemu, pozwalających na wysoki stopień konfigurowalności. [11]



Rys.4.9. Schemat dostarczanego przez Django modułu autoryzacyjnego

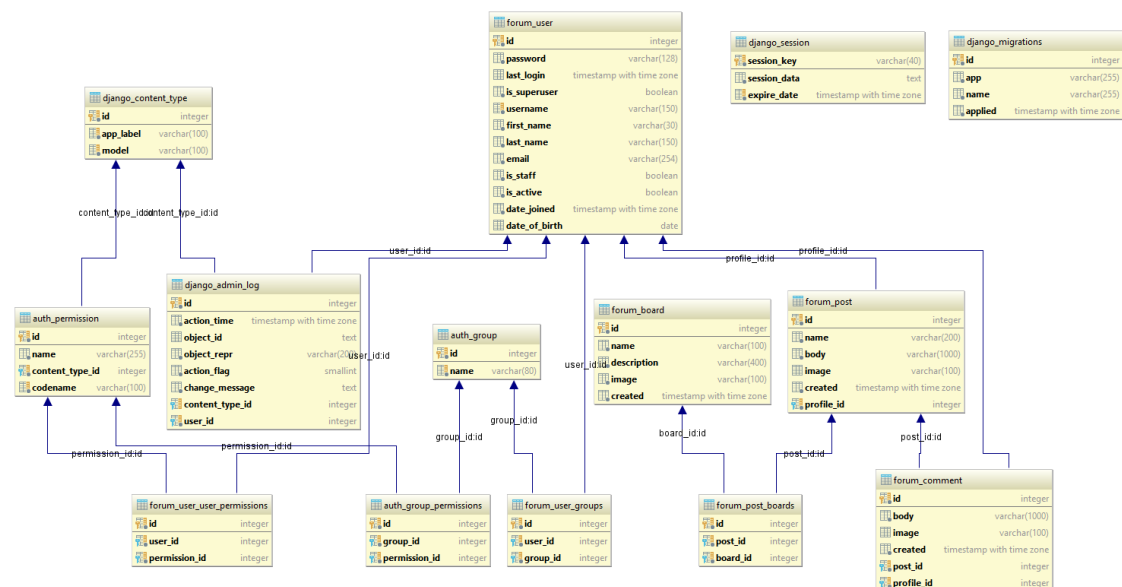
Baza danych wykorzystana w tworzonej aplikacji jest bazą relacyjną, więc jednym z pierwszych kroków na etapie projektowania aplikacji było przygotowanie diagramu ERD [3] przedstawiającego encje i połączenia między nimi.

Django pozwala na automatyczne generowanie klas asocjacyjnych w relacjach wiele do wielu, dodatkowo zmniejszając zakres pracy związany z tworzeniem bazy danych. Klasy te zostaną faktycznie utworzone w bazie i znajdują się na diagramie ERD oraz schemacie bazy danych, ale znika konieczność odwoływania się do nich podczas programowania logiki tworzonego systemu.



Rys. 4.10. Diagram ERD

Na bazie powyższego diagramu ERD, utworzonego z uwagą na naturę dostarczanego przez Django modułu autoryzacyjnego, jesteśmy w stanie utworzyć kompletną bazę danych o następującym schemacie:



Rys. 4.11. Schemat utworzonej bazy danych

5. Implementacja

5.1. Przegląd technologii

W tym podrozdziale opisane zostaną najważniejsze technologie wykorzystane przy tworzeniu aplikacji webowej wytwarzanej w ramach niniejszej pracy. Poprzez technologie rozumiane są platformy programistyczne, oprogramowanie pośrednie, systemy zarządzania bazą danych czy też biblioteki. Przy wyborze technologii najwięcej uwagi poświęcone zostało cechom takim jak: dostępność materiałów do nauki, jakość dokumentacji oraz możliwość wykorzystania ich bez konieczności uiszczania opłat. Kluczowe było również to, aby wybrane technologie należały do czołówki najczęściej używanych i najbardziej docenianych technologii w ich odpowiednich kategoriach.

5.1.1. Django

Django [11] jest wydaną w roku 2005 platformą programistyczną operującą w języku Python. Rozwijany przez Django Software Foundation, niezależną organizację non-profit, framework Django jest regularnie aktualizowany i ulepszany, z najnowszą wersją 2.1.4 wydaną pod koniec roku 2018.

Kluczową zaletą frameworku Django jest prędkość z jaką można utworzyć za jego pomocą prototyp i finalną wersję aplikacji webowej. Dodatkowo Django automatycznie generuje kompletny panel administracyjny z możliwością personalizacji i dostosowania do własnych potrzeb. Do dyspozycji dewelopera oddaje także dużą liczbę paczek, bibliotek i oprogramowania pośredniego, które rozwiązują często spotykane problemy i zaspakajają najczęściej występujące potrzeby. Twórcy postarali się także, aby zapewnić wysoki poziom bezpieczeństwa, skupiając się na przeciwdziałaniu wstrzykiwaniu kodu SQL, atakom typu XSS, polegającym na wstrzykiwaniu do przeglądarki ofiary kodu javascript [2], czy też atakom CSRF, polegającym na zmuszeniu przeglądarki ofiary do wykonania nieautoryzowanego żądania HTTP [8]. Jednak jedną z największych zalet frameworku Django jest bogata dokumentacja oraz związana z nim, rozległa, aktywna społeczność internetowa.

Pomimo prostego procesu tworzenia aplikacji webowych, Django zapewnia wysoki stopień skalowalności – największe strony działające aktualnie na frameworku Django posiadają miliony użytkowników i obsługują setki żądań na sekundę. Do najbardziej znanych należą: Instagram, Bitbucket, Disqus, Mozilla, NASA oraz The Washington Post.

Django umożliwia również programistom pracę na modelach i tym samym sprawia, że w ciągu całego procesu wytwarzania oprogramowania, deweloperzy nie muszą ani razu korzystać z języka SQL do operowania bazą danych. Jest to duże udogodnienie i ogromny plus dla wszystkich którzy wolą posługiwać się językiem Python.

5.1.2. Django Rest Framework

Django Rest Framework[12] jest wydajną i elastyczną platformą programistyczną napisaną w języku Python, służącą do budowania webowych interfejsów programowania aplikacji zgodnych ze stylem REST. Jest to framework typu open-source, utworzony przez programistę, Toma Christie i rozwijany przez niego od lat, z pomocą licznych ochotników.

Django Rest Framework pozwala na tworzenie API i ułatwia ten proces poprzez wprowadzenie prostego widoku w przeglądarce do poruszania się po nim. Wszystkie aspekty tworzonego interfejsu programowania aplikacji są konfigurowalne, a w przypadku generycznych zadań, Django Rest Framework oddaje nam do dyspozycji gotowe metody pozwalające na ich zrealizowanie, bez konieczności pisania rozwiązań od zera.

Jako że Django Rest Framework, w odróżnieniu od samego Django, nie należy do ścisłej czołówki najpopularniejszych frameworków webowych, rozmiar i aktywność społeczności

związanej z nim jest proporcjonalnie mniejsza. Zapewnia on jednak dostęp do obszernej dokumentacji, wielu poradników oraz przykładów, pozwalających opanować podstawy i techniki użytkowania go.

5.1.3. Vue.js

Vue.js[20] jest platformą programistyczną języka JavaScript umożliwiającą budowanie interfejsów użytkownika. Została ona stworzona przez firmę Vue Technology, założoną przez Evana You. Vue.js jest technologią typu open-source, co ma znaczący wpływ na rozmiar i aktywność związanej z nią społeczności.

Vue.js jest frameworkiem wykorzystywanym do projektów o różnych rozmiarach i naturach i jest dla wielu ciekawą alternatywą dla platformy React, ponieważ przy użyciu dedykowanego rozwiązania do obsługi routingu, projektowaną stronę można przekształcić w SPA.

„SPA” jest tutaj akronimem angielskiego wyrażenia „single-page application” oznaczającego aplikację jednostronicową. Jest to sposób projektowania stron zyskująca w ciągu ostatnich lat sporo uwagi, zakładający istnienie jednego, niezmiennego szablonu, do którego ładowane są dane w zależności od okoliczności. Cały ten proces, wraz z trasowaniem (routingiem) realizowany jest po stronie klienta. Podejście to różni się znacząco od standardu w tworzeniu stron internetowych, według którego oddzielne widoki są konkretnymi plikami wysyłanymi z serwera do klienta w celu ich wyświetlenia.

Flagową mechaniką platformy Vue.js są „single file components”, komponenty pozwalające na grupowanie kodu źródłowego aplikacji w oddzielne pliki zgodnie z przyjętą konwencją. Każdy komponent posiada unikalną nazwę i składa się z trzech podstawowych elementów: szablonu przechowującego zdefiniowaną w języku znaczników HTML strukturę, części zawierającej skrypty w języku JavaScript odpowiedzialne za dynamiczne zarządzanie komponentem, oraz części przechowującej styl komponentu, zdefiniowany przy pomocy języka CSS/SCSS [1].

5.1.4. Inne warte uwagi technologie

- Axios – [9] bazujący na obietnicach klient HTTP, wykorzystywany do pochłaniania API
- Django Rest Swagger – [13] biblioteka służąca do wizualizacji i łatwego korzystania z API
- Vue Material – [19] biblioteka pozwalająca zintegrować styl Material Design i Vue.js
- Vuex – [21] biblioteka pozwalająca na zarządzanie stanem w aplikacjach Vue.js

5.2. Narzędzia programistyczne

Ważnym krokiem poprzedzającym zawsze etap implementacji jest proces dobierania odpowiednich narzędzi do wyznaczonego zadania. W przypadku zadań programistycznych, oznacza to wybór programów i środowisk programistycznych, które usprawnią proces wytwarzania oprogramowania. Do realizowanego projektu wybrane zostały następujące narzędzia programistyczne:

5.2.1. DataGrip

DataGrip[10] jest rozwijanym przez firmę JetBrains, zintegrowanym środowiskiem graficznym baz danych, znacznie usprawniającym pracę z bazami danych SQL. Przyjazny interfejs, zestaw profesjonalnych narzędzi i zaawansowany mechanizm autouzupełniania kodu są niektórymi z powodów, dla których DataGrip jest ciekawą alternatywą dla systemów

zarządzania baz danych oferowanych przez twórców konkretnych składni SQL. W tworzonej pracy głównymi zastosowaniami programu były: wizualizacja struktury bazy danych oraz możliwość szybkiego podglądu danych znajdujących się w bazie danych.

5.2.2. Git

Git[15] jest zdecentralizowanym systemem kontroli wersji. Rozwijany na zasadach wolnego oprogramowania, oddaje w ręce programistów narzędzie ułatwiające pracę nad projektami rozwijanymi przez grupy programistów. Git jest obecnie najpopularniejszym systemem kontroli wersji na świecie.

5.2.3. ESLint

ESLint[14] narzędziem przeprowadzającym statyczną analizę kodu, polegającą na automatycznym wykonywaniu testów na kodzie, w celu znalezienia w nim błędów popełnionych przez programistę. ESLint dba także o utrzymanie spójnej stylistyki kodu, czytelność tworzonych metod oraz zgodność z ogólnie przyjętymi konwencjami strukturyzowania kodu.

5.2.4. PyCharm

PyCharm[18], podobnie do DataGrip, jest rozwijanym przez firmę JetBrains, zintegrowanym środowiskiem graficznym. PyCharm dostosowany jest do pracy z językiem Python, ale nie pozwala też na wygodną pracę z dowolnymi platformami programistycznymi, pod warunkiem obecności odpowiednich wtyczek. Narzędzie to pozwala na usprawnienie procesu implementacji oferując programistom zaawansowany mechanizm autouzupełniania kodu, bogaty zestaw skrótów klawiaturowych oraz wbudowaną integrację systemu kontroli wersji Git.

5.3. Opis implementacji funkcjonalności

5.3.1. Punkty końcowe API

Aby klient mógł pozyskiwać z serwera dane, udostępnione zostały mu odpowiednie punkty końcowe interfejsu programowania aplikacji (API). Utworzone API definiuje to, jakie operacje pozwala wykonać tworzona aplikacja [5]. Dzięki jego wykorzystaniu, aplikacja po stronie klienta nie potrzebuje informacji o klasach ani metodach oferujących dane usługi, a jedynie adresu URL na serwerze.

Aby utworzyć punkt końcowy API, konieczne było zdefiniowanie klas Serializer, tłumaczących dane w zależności od kontekstu na struktury JSON lub obiekty Python, oraz klas ViewSet, specyfikujących zachowanie i charakter tworzonych punktów końcowych. Po utworzeniu wyżej wymienionych elementów, konieczne jest opisanie Routingu, czyli zmapowanie ścieżek do poszczególnych klas ViewSet zwracających požądane dane.

Django Rest framework udostępnia gotowe domieszki (mixin) które można wykorzystać przy implementacji generycznych funkcjonalności, co pozwoliło na szybkie utworzenie części punktów końcowych. Zdefiniowane zostały także metody obsługujące żądania POST i pozwalające na tworzenie nowych obiektów.

Na kolejnej stronie zaprezentowane zostały fragmenty kodu przedstawiające elementy potrzebne do utworzenia punktu końcowego API pozwalającego na pozyskanie listy Postów za pomocą metody GET.

```

class Post(models.Model):
    boards = models.ManyToManyField('Board', related_name='posts')
    user = models.ForeignKey(User, default=1, on_delete=models.SET_DEFAULT)
    title = models.CharField(max_length=200)
    body = models.CharField(max_length=1000)
    image = models.FileField(upload_to='%Y/%m/%d/', null=True, blank=True)
    score = models.IntegerField(default=0)
    created = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return 'Id{} {}, by {}'.format(self.id, self.title, self.user)

```

Rys. 5.1. Implementacja modelu Post

```

class PostSerializer(serializers.ModelSerializer):
    class Meta:
        model = Post
        fields = '__all__' #('id', 'user', 'name', 'body', 'image')

```

Rys. 5.2. Implementacja klasy Serializer dla modelu Post

```

class PostViewSet1(viewsets.ViewSet, mixins.CreateModelMixin):
    serializer_class = PostSerializer

    def list(self, request, board_pk=None):
        queryset = Post.objects.filter(boards=board_pk)
        serializer = PostSerializer(queryset, many=True)
        return Response(serializer.data)

    def retrieve(self, request, pk=None, board_pk=None):
        queryset = Post.objects.filter(pk=pk, boards=board_pk)
        post = get_object_or_404(queryset, pk=pk)
        serializer = PostSerializer(post)
        return Response(serializer.data)

```

Rys. 5.3. Implementacja klasy ViewSet dla modelu Post

```

board_router = routers.NestedSimpleRouter(router, r'boards', lookup='board')
board_router.register(r'posts', views.PostViewSet1, base_name='posts')

```

Rys. 5.4. Zmapowanie ścieżki do punktu końcowego modelu Post

5.3.2. Obsługa plików graficznych w systemie

Jednym z wyzwań programistycznych związanych z realizowaną pracą, była niewątpliwie implementacja mechanizmów związanych z obsługą plików graficznych. Tematy takie jak sposoby przechowywania plików w systemie, czy obsługa procesu ich tworzenia oraz usuwania okazały się zagadnieniami opisanymi w dokumentacji Django Rest Framework w minimalnym stopniu. Proces implementacji tych funkcjonalności wymagał znacznej ilości czasu poświęconego na zaznajamianie się z powiązanymi tematycznie materiałami znalezionymi na forach programistycznych.

W tworzonej aplikacji, zdjęcia przesyłane przez użytkowników, zapisywane są w dynamicznie tworzonych, odpowiednio nazywanych folderach w systemie plików, a do bazy danych trafiają ich adresy. Tego typu rozwiązanie wymagało zapewnienia odpowiednich metod automatycznie usuwających pliki, których adresy zostaną usunięte z bazy danych, przykładowo w efekcie modyfikacji zdjęcia reprezentującego daną tablicę.

Po stronie klienta, do przesyłania zdjęć wykorzystana została biblioteka Vue-picture-input, a wybrane zdjęcie zostaje zakodowane w base64 i wysłane wraz z treścią publikowanego posta/komentarza na serwer.

```
if kwargs['image'] is not None:
    uri = DataURI(kwargs['image'])
    _, extension = uri.mimetype.split("/")
    new_name = ''.join(random.choices(string.ascii_uppercase + string.digits, k=100))
    data = ContentFile(uri.data)
    instance.image.save('{0}-{1}'.format(new_name, extension), data)
```

Rys. 5.5. Obsługa zdjęcia wysłanego metodą POST po stronie serwera

5.3.3. Autoryzacja

Podczas gdy platforma programistyczna Django oferuje zadowalające rozwiązania w kwestii autoryzacji, w rozwijanej aplikacji wykorzystany został aktualnie zyskujący popularność mechanizm JWT – Json Web Token [17]. Pozwala on na wyeliminowanie istnienia sesji użytkownika na serwerze, zastępując ją rozwiązaniem bezstanowym o następującym działaniu:

1. Użytkownik wysyła wypełniony formularz logowania
2. System weryfikuje otrzymane dane logowania i w przypadku gdy są poprawne, wysyła w odpowiedzi token
3. Uzyskany token zostaje zapisany po stronie klienta i może być używany do potwierdzania tożsamości użytkownika. Gwarantuje on również dostęp do części aplikacji dostępnych jedynie dla zalogowanych użytkowników.

JWT składa się z trzech podstawowych części:

- Header – informacja o tym jaki algorytm szyfrujący został wykorzystany przy tworzeniu tokenu
- Payload – Dane dotyczące autoryzowanego użytkownika i daty ważności wysłanego tokenu
- Signature – Podpis cyfrowy składający się z zaszyfrowanego połączenia części Header i Payload.

Implementując mechanizm JWT w tworzonej aplikacji wykorzystana została biblioteka Simplejwt[16], umożliwiająca tworzenie i weryfikację tokenów JWT.

```

path('api/token/', TokenObtainPairView.as_view(), name='token_obtain_pair'),
path('api/token/refresh/', TokenRefreshView.as_view(), name='token_refresh'),
path('api/token/verify/', TokenVerifyView.as_view(), name='token_verify')

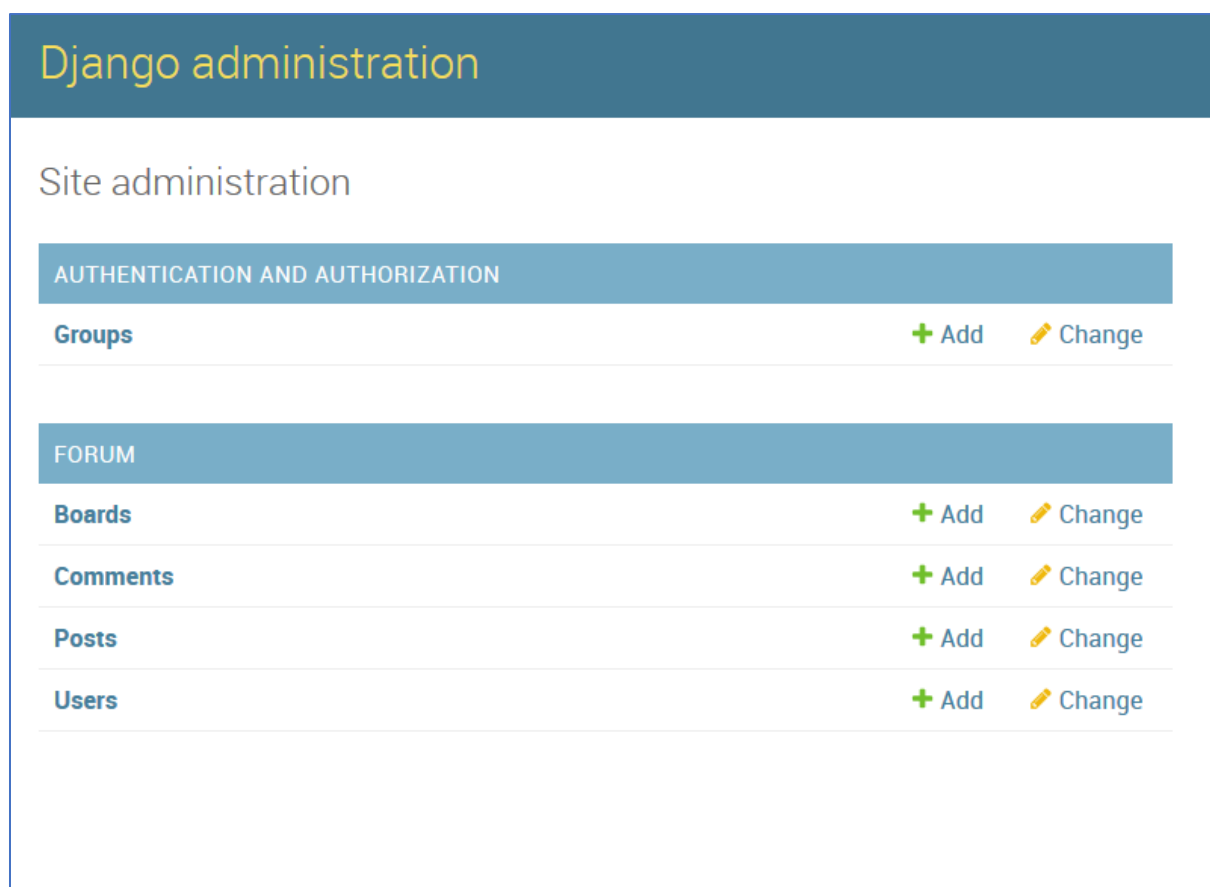
```

Rys. 5.6. Punkty końcowe API wykorzystywane do obsługi JWT

5.3.4. Panel Administracyjny

Jedną z największych zalet platformy programistycznej Django jest automatycznie tworzona strona zawierająca panel administracyjny. Konfigurowalna z poziomu pliku odpowiedzialnego za ustawienia Django i generowana w oparciu o zdefiniowane modele, strona ta oddaje w ręce administratorów narzędzia pozwalające na wykonywanie operacji CRUD na bazie danych powiązanego systemu.

Strona ta pozwala na realizację przypadków użycia wykonywanych przez administratora, opisanych na etapie specyfikacji wymagań.



Rys. 5.7. Panel administracyjny Django

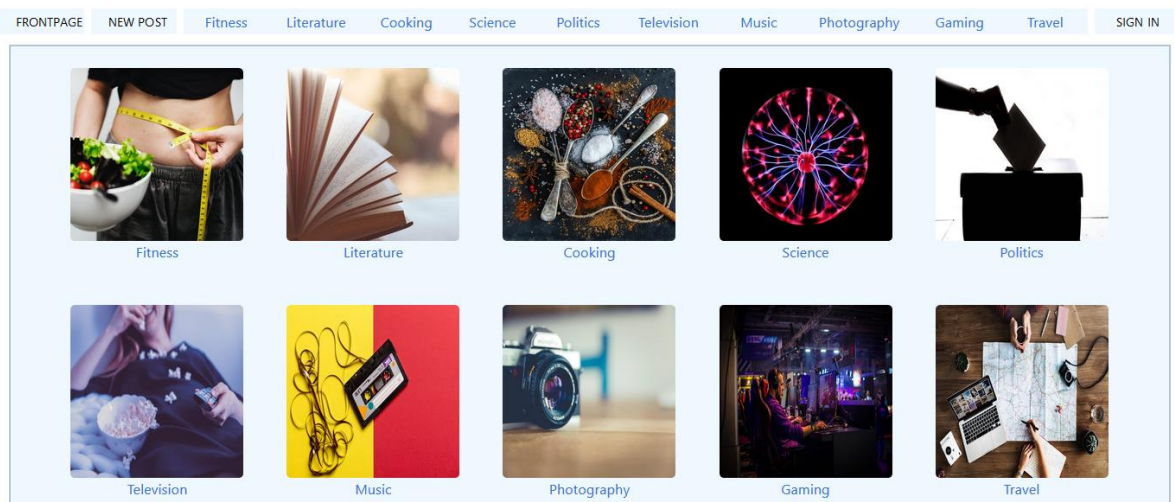
5.4. Prezentacja interfejsu aplikacji

Poniżej przedstawione zostały przykłady ilustrujące wygląd aplikacji podczas realizacji wybranych przypadków użycia.

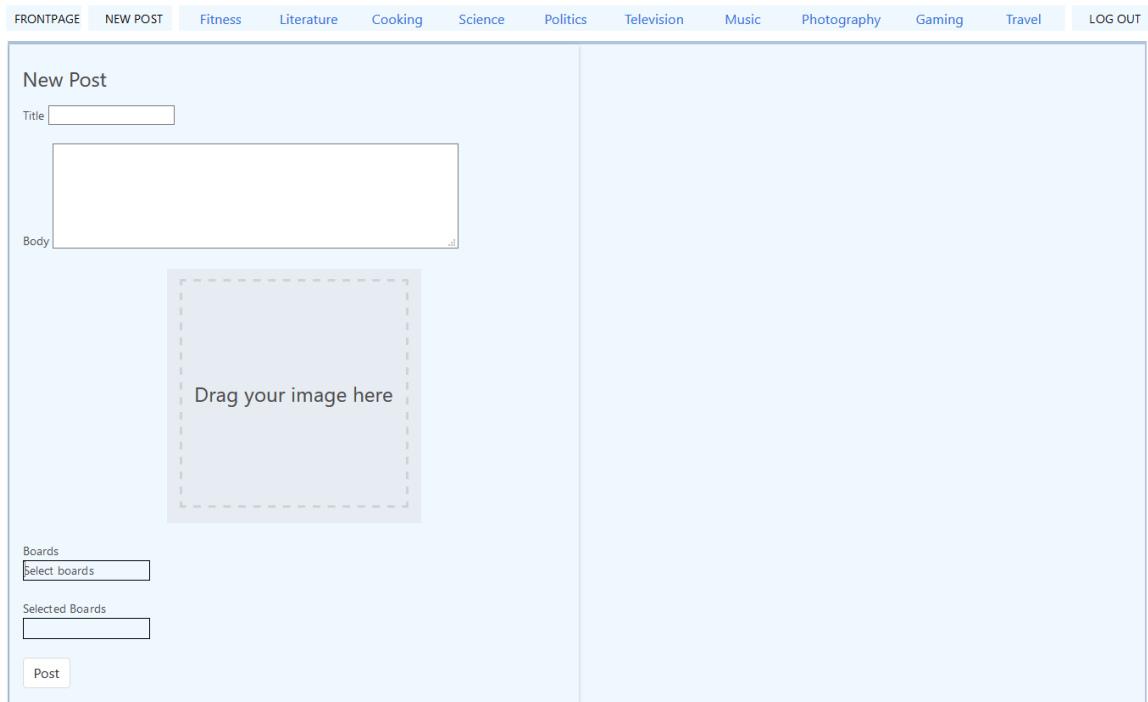


The screenshot shows the login interface of the application. At the top, there is a navigation bar with links: FRONTPAGE, NEW POST, Fitness, Literature, Cooking, Science, Politics, Television, Music, Photography, Gaming, Travel, and SIGN IN. Below the navigation bar, there is a light blue box containing the login form. The form has two input fields: 'Username' and 'Password'. Below the 'Password' field, there is a 'Login' button and a 'Sign up' link.

Rys. 5.9. Realizacja przypadku użycia P02 - Logowanie



Rys. 5.8. Realizacja przypadku użycia P03 – Przeglądanie tablic



The screenshot shows the 'New Post' form in the application. At the top, there is a navigation bar with links: FRONTPAGE, NEW POST, Fitness, Literature, Cooking, Science, Politics, Television, Music, Photography, Gaming, Travel, and LOG OUT. Below the navigation bar, there is a light blue box containing the 'New Post' form. The form has a 'Title' input field and a 'Body' text area. Below the 'Body' text area, there is a dashed box with the text 'Drag your image here'. At the bottom left of the form, there is a 'Boards' section with a 'Select boards' dropdown menu and a 'Selected Boards' input field. Below the 'Selected Boards' input field, there is a 'Post' button.

Rys. 5.9. Realizacja przypadku użycia P06 – Tworzenie postu

6. Testowanie

Testowanie jest działaniem mającym na celu potwierdzenie, że system nie zawiera błędów oraz odkrycie ewentualnych luk w oprogramowaniu. Prawdopodobieństwo wystąpienia niedociągnięć w systemie rośnie wraz z poziomem skomplikowania tworzonego systemu. W celu weryfikacji poprawności działania implementowanej aplikacji, przeprowadzone zostały testy jednostkowe oraz testy systemowe.

6.1. Testy jednostkowe

Testy jednostkowe weryfikują poprawne działanie konkretnych części kodu. Najczęściej, testy te sprawdzają działanie metod. Tradycyjnie, test składa się z kodu wywołującego testowaną metodę i walidującego zwrócony przez nią wynik [6].

Do testowania tworzonego oprogramowania wykorzystane zostały narzędzia dostarczone przez Django i Django Rest Framework. Platformy te dostarczają biblioteki pozwalające na między innymi: zasymulowanie przeglądarki internetowej, wysyłanie oraz odbieranie żądań HTTP czy pracę na tymczasowej bazie danych.

Przeprowadzone testy pozwoliły na uzyskanie stopnia pokrycia kodu wynoszącego 89%. Wynik ten został pozyskany poprzez użycie biblioteki Coverage.py. W wykonanych testach dokonane zostało 11 asercji.

Poniżej przedstawiona została implementacja testu pozwalającego na weryfikację poprawnego działania systemu w przypadku próby utworzenia nowego postu.

```
class PostTests(APITestCase):
    def setUp(self):
        User.objects.create(username='test', password='test')

    def test_post_create(self):
        url = '/api/posts/'
        data = {"user": "test",
                "title": "test",
                "body": "test",
                "image": None,
                "boards": [
                    {"id": 1}
                ]}
        response = self.client.post(url, data, format='json')
        self.assertEqual(response.status_code, status.HTTP_201_CREATED)
        self.assertEqual(Post.objects.count(), 1)
        self.assertEqual(Post.objects.get().title, 'test')
```

Rys. 6.1. Przykład implementacji testu jednostkowego

6.2. Testy systemowe

Utworzony system został poddany testom systemowym, czyli interakcjom użytkownika z systemem. Działają one na zasadzie czarnej skrzynki, co oznacza, że osoba wykonująca test nie musi wiedzieć jak zaimplementowany jest system. Przeprowadzane jako testy manualne, testy systemowe wykorzystują uprzednio przygotowane instrukcje, w celu weryfikacji odpowiedzi systemu na konkretne działania użytkownika.

Poniżej przedstawione zostały przykłady instrukcji wykonanych testów systemowych.

Numer kroku	Akcja	Oczekiwany wynik
1	Użytkownik z paska nawigacji wybiera zakładkę „Sign Up”	Wyświetlona zostaje strona rejestracji zawierająca formularz tworzenia konta użytkownika
2	Użytkownik wypełnia formularz danymi: Username: test Email: test@example Password: test	Brak
3	Użytkownik wybiera przycisk [Sign Up]	Wyświetlona zostaje strona główna
4	Użytkownik z paska nawigacji wybiera zakładkę „Sign In”	Wyświetlona zostaje strona logowania zawierająca formularz logowania
5	Użytkownik loguje się za pomocą danych: Username: test Password: test	Brak
6	Użytkownik wybiera przycisk [Sign In]	Wyświetlona zostaje strona główna

Tabela 6.1. Test systemowy - Rejestracja i logowanie

Numer kroku	Akcja	Oczekiwany wynik
1	Użytkownik loguje się za pomocą danych: Username: test Password: test	Wyświetlona zostaje strona główna
2	Użytkownik z paska nawigacji wybiera zakładkę „New Post”	Wyświetlony zostaje formularz tworzenia nowego postu
3	Użytkownik wypełnia formularz danymi: Title: test Body: test Boards: test	Brak
4	Użytkownik wybiera przycisk [Post]	Wyświetlony zostaje nowo utworzony post

Tabela 6.2. Test systemowy – Tworzenie nowego postu

7. Podsumowanie

W ramach realizacji pracy przedstawiony został proces tworzenia projektu i implementacji aplikacji webowej o funkcjonalności forum dyskusyjnego. We wstępie wyznaczony został cel pracy, następnie sporządzony został przegląd istniejących rozwiązań oraz utworzona została specyfikacja wymagań. Zostały w niej opisane przypadki użycia oraz sporządzone zostały reguły biznesowe, a także zaprezentowana została struktura bazy danych. W dalszej części pracy opisano etap implementacji, skupiający się na wyszczególnieniu użytych technologii i narzędzi oraz prezentujący stronę techniczną wybranych zagadnień dotyczących wytwarzania aplikacji. Pracę zamknął rozdział mówiący o testach, którym poddany został rozwijany system. Cel pracy zdefiniowany we wstępie został osiągnięty i spełnione zostały wymagania funkcjonalne oraz нефункционалне.

Zaimplementowane zostały główne funkcjonalności wyróżnione na etapie przeglądu istniejących rozwiązań, a wytworzone oprogramowanie spełnia wymagania wyszczególnione w ramach specyfikacji wymagań. Działanie systemu zostało zweryfikowane za pomocą testów jednostkowych oraz systemowych. W pracy spełnione zostały wszystkie wymagania funkcjonalne i нефункционалне, co potwierdzone zostało poprzez przeprowadzenie testów na różnych środowiskach - na aktualnych wersjach przeglądarek Firefox i Chrome, przy użyciu systemów operacyjnych Linux oraz Windows.

Dostarczone oprogramowanie pozwala na publikowanie przez użytkowników postów oraz komentarzy oraz umożliwia administratorowi dodawanie, edycję oraz usuwanie tablic, postów oraz komentarzy. Użytkownicy mają możliwość publikowania jednego postu na wielu tablicach, a utworzony interfejs pozwala na wygodne przeglądanie plików graficznych oraz tekstu.

Po zaczerpnięciu inspiracji z wymienionych w przeglądzie rozwiązań istniejących na rynku systemów, wytworzone oprogramowanie najbardziej przypomina wyglądem serwis 4chan/4channel, z racji podobnej idei stojącej za wyglądem interfejsu. Podczas gdy zaimplementowana aplikacja posiada elementy zaczerpnięte z serwisów Reddit oraz StackOverflow, z racji braku mechanizmu oceniania postów i komentarzy, utworzona aplikacja nie posiada wysokiego stopnia podobieństwa do żadnego z nich. Finalnie, opisywane oprogramowanie można umieścić pomiędzy serwisami Reddit i 4chan/4channel, z racji występowania w nich tematycznych tablic i nastawienia na prowadzenie dyskusji pod publikowanymi postami.

Niewykluczona jest możliwość dalszego rozwoju wytworzonego oprogramowania. Możliwymi kierunkami rozwoju dla powstałej aplikacji jest wprowadzenie mechanizmów takich jak:

- system oceniania postów i komentarzy
- personalizowane profile użytkowników
- prywatne tablice wymagające autoryzacji w celu ich przeglądania
- zagnieżdżone komentarze
- poprawa aspektów wizualnych
- możliwość logowania za pomocą konta Google+ lub konta Facebook
- możliwość wysyłania prywatnych wiadomości do użytkowników

8. Bibliografia

- [1] Filipova, O. (2016) *Learning Vue.js 2*. Packt Publishing
- [2] Fogie, S., Grossman, J., Hansen, R., Rager, A., Petkov, P. (2007) *XSS Attacks: Cross Site Scripting Exploits and Defense* Syngress Publishing
- [3] Fowler, M. (2005) *UML w kropelce 2.0*. Oficyna Wydawnicza LTP
- [4] Raaj, S. (2018) *Building APIs with Django and Django Rest Framework*. Agiliq Info Solutions India Pvt Ltd
- [5] Richardson, L., Ruby, S., & Amundsen, M. (2013). *RESTful Web APIs*. O'Reilly Media Inc.
- [6] Stephens, R. (2015). *Beginning Software Engineering, 1st Edition*. Wrox Press
- [7] Wang, S., Lo, D., Jiang, L., (2013) *An Empirical Study on Developer Interactions in StackOverflow*. SAC 2013: Proceedings of the 28th annual ACM Symposium on Applied Computing: Coimbra, Portugal, 18-22 March 2013
- [8] Zeller, W., Felten, E. (2008) *Cross-Site Request Forgeries: Exploitation and Prevention* Princeton University
- [9] Axios (2018) <https://github.com/axios/axios>
- [10] DataGrip. (2018) <https://www.jetbrains.com/datagrip/>
- [11] Django documentation (2018) <https://docs.djangoproject.com/en/2.1/>
- [12] Django Rest Framework API Guide (2018) <https://www.django-rest-framework.org>
- [13] Django Rest Swagger (2018) <https://github.com/marcgibbons/django-rest-swagger>
- [14] ESLint (2018) <https://eslint.org/>
- [15] Git. (2018). Pobrano z lokalizacji Git: <https://git-scm.com/>
- [16] JSON Web Token authentication plugin for the Django REST Framework. (2018) <https://github.com/davesque/django-rest-framework-simplejwt>
- [17] JSON Web Tokens (2018) <https://jwt.io/introduction/>
- [18] PyCharm. (2018) <https://www.jetbrains.com/pycharm/>
- [19] Vue Material. (2018) <https://vuematerial.io/>
- [20] Vue.js Guide. (2018) <https://vuejs.org/v2/guide/>
- [21] What is Vuex? (2018) <https://vuex.vuejs.org/>

9. Załączniki

Spis listingów kodu

Rys. 5.1. Implementacja modelu Post	26
Rys. 5.2. Implementacja klasy Serializer dla modelu Post	26
Rys. 5.3. Implementacja klasy ViewSet dla modelu Post	26
Rys. 5.4. Zmapowanie ścieżki do punktu końcowego modelu Post	26
Rys. 5.5. Obsługa zdjęcia wysłanego metodą POST po stronie serwera	27
Rys. 5.6. Punkty końcowe API wykorzystywane do obsługi JWT	28
Rys. 6.1. Przykład implementacji testu jednostkowego	30

Spis rysunków

Rys. 3.1. Diagram przypadków użycia	15
Rys. 4.1. Strona główna	17
Rys. 4.2. Logowanie	17
Rys. 4.3. Rejestracja	18
Rys. 4.4. Tworzenie nowego komentarza	18
Rys. 4.5. Tworzenie nowego postu	19
Rys. 4.6. Przeglądanie postów	19
Rys. 4.7. Przeglądanie komentarzy	20
Rys. 4.8. Okno logowania panelu administracyjnego	20
Rys. 4.9. Schemat dostarczanego przez Django modułu autoryzacyjnego	21
Rys. 4.10. Diagram ERD	22
Rys. 4.11. Schemat utworzonej bazy danych	22
Rys. 5.7. Panel administracyjny Django	28
Rys. 5.8. Realizacja przypadku użycia P06 – Tworzenie postu	29
Rys. 5.9. Realizacja przypadku użycia P02 - Logowanie	29
Rys. 5.10. Realizacja przypadku użycia P03 – Przeglądanie tablic	29

Spis tabel

Tabela 3.1. Użytkownicy aplikacji	8
Tabela 3.2. Historyjki użytkowników	8
Tabela 3.3 Wymagania funkcjonalne	9
Tabela 3.4. Wymagania нефункционалне	9
Tabela 3.5. Opis przypadku użycia P001	10
Tabela 3.6. Opis przypadku użycia P002	10
Tabela 3.7. Opis przypadku użycia P003	10
Tabela 3.8. Opis przypadku użycia P004	11
Tabela 3.9. Opis przypadku użycia P005	11
Tabela 3.10. Opis przypadku użycia P006	11
Tabela 3.11. Opis przypadku użycia P007	12
Tabela 3.12. Opis przypadku użycia P008	12
Tabela 3.13. Opis przypadku użycia P009	12
Tabela 3.14. Opis przypadku użycia P010	13
Tabela 3.15. Opis przypadku użycia P011	13
Tabela 3.16. Opis przypadku użycia P012	13

Tabela 3.17. Opis przypadku użycia P013	14
Tabela 3.18. Opis przypadku użycia P014	14
Tabela 6.1. Test systemowy - Rejestracja i logowanie	31
Tabela 6.2. Test systemowy – Tworzenie nowego postu	31