1. Tytuł projektu:

CMS Blogowy Portal - System Zarządzania Treścią dla Blogów

2. Krótki opis działania projektu:

CMS Blogowy Portal to platforma służąca do zarządzania treścią blogową, oferująca różnorodne funkcjonalności dostosowane do potrzeb użytkowników o różnych rolach. System umożliwia rejestrację i logowanie użytkowników, którzy mogą korzystać z rozbudowanych możliwości związanych z tworzeniem oraz zarządzaniem treścią, takich jak publikacja artykułów, komentowanie, ocenianie wpisów i przeglądanie statystyk portalu.

Użytkownicy posiadają przypisane role, które określają ich uprawnienia: administratorzy mogą zarządzać wszystkimi aspektami systemu, autorzy skupiają się na tworzeniu artykułów, a moderatorzy dbają o jakość dyskusji, zatwierdzając lub usuwając komentarze i oceny. Czytelnicy mają możliwość interakcji z treścią poprzez dodawanie komentarzy i ocen, natomiast analitycy mogą analizować statystyki portalu, takie jak liczba wyświetleń artykułów czy poziom aktywności użytkowników.

a. Role użytkowników i ich uprawnienia:

i. Administrator:

Posiada wszystkie uprawnienia, w tym możliwość zarządzania użytkownikami, treścią, komentarzami, ocenami oraz dostępem do statystyk.

ii. Autor:

Może dodawać i edytować własne artykuły, organizować je w kategoriach oraz przypisywać do nich tagi.

iii. Czytelnik:

Może dodawać i edytować swoje komentarze oraz oceny pod artykułami.

iv. Moderator:

Odpowiada za moderację treści, w tym zatwierdzanie lub usuwanie komentarzy oraz ocen.

v. Analityk:

Ma dostęp do statystyk portalu, takich jak liczba wyświetleń artykułów, liczba komentarzy, oceny oraz ogólne dane o aktywności użytkowników.

b. Kluczowe funkcjonalności systemu:

i. Rejestracja:

Nowi użytkownicy mogą się zarejestrować, aby uzyskać dostęp do określonych funkcjonalności portalu.

ii. Logowanie i wylogowywanie:

Zalogowani użytkownicy mogą korzystać z pełni funkcjonalności portalu odpowiednich do ich ról.

iii. **Zarządzanie wpisami:**

Użytkownicy z rolą autora mogą dodawać oraz edytować artykuły. Artykuły mogą być kategoryzowane.

iv. **Komentarze i oceny:**

Czytelnicy mogą dodawać komentarze oraz oceny do artykułów. Moderatorzy mają uprawnienia do zatwierdzania lub usuwania komentarzy oraz ocen, aby zapewnić odpowiedni poziom dyskusji.

v. **Statystyki:**

Analitycy mogą przeglądać statystyki dotyczące aktywności użytkowników, takich jak liczba odwiedzin artykułów oraz interakcje takie jak komentarze i oceny.

vi. **Moderacja:**

Moderatorzy mogą usuwać lub zatwierdzać komentarze i oceny, dbając o jakość i kulturę dyskusji.

3. Autor:

Kamil Matyja K80 nr. albumu 138332

4. Specyfikacja wykorzystanych technologii

a. C#:

Wiodący język programowania obiektowego w projekcie, pozwalający na wydajne i nowoczesne zarządzanie logiką aplikacji.

b. ASP.NET Core 8.0:

Użyty jako platforma do tworzenia aplikacji internetowych, umożliwiający szybkie i wydajne działanie aplikacji serwerowej.

c. Entity Framework Core:

ORM (Object-Relational Mapper) używany do zarządzania bazą danych SQLite, umożliwiający abstrakcję i łatwe zarządzanie operacjami bazodanowymi.

d. Baza danych SQLite:

Lekka relacyjna baza danych, która pozwala na łatwe zarządzanie lokalnymi danymi bez potrzeby instalacji dodatkowych serwerów.

e. Warstwa prezentacji (frontend):

i. **Razor Pages:**

Używany do renderowania widoków HTML po stronie serwera, co upraszcza tworzenie dynamicznych stron internetowych.

ii. **HTML/CSS i JavaScript:**

Standardowe technologie do budowy interfejsu użytkownika.

f. **Narzędzia i środowisko developerskie:**
   i. **JetBrains Rider:**

   Wykorzystane jako główne IDE wspierające szybkie i efektywne tworzenie oprogramowania.

   ii. **Linux Ubuntu 24.10:**

   System operacyjny używany podczas tworzenia i testowania projektu.

g. **System zarządzania zależnościami i budowaniem .NET CLI:**

Narzędzie do obsługi budowania projektu, zarządzania zależnościami oraz operacji na bazach danych (jak migracje).

h. **Kontrola wersji Git:**

System kontroli wersji do zarządzania kodem i historią rozwoju projektu.

5. Instrukcje pierwszego uruchomienia projektu

Aby uruchomić projekt po raz pierwszy, należy wykonać poniższe kroki:

a. **Klonowanie repozytorium:**
   i. **Sklonuj kod źródłowy projektu na swój lokalny komputer przy pomocy komendy:**

   bash git clone https://github.com/kamilmatyja/Galak-Pizza.git

b. **Przygotowanie środowiska:**
   i. **Upewnij się, że masz zainstalowaną następującą wersję .NET SDK (lub wyższą):**
   ii. **NET Core 8.0.x możesz instalować ją zgodnie z oficjalną dokumentacją .NET.**
c. **Przygotowanie bazy danych:**
   i. **Wykonaj polecenia w poniższej kolejności:**
      1. **Usunięcie starej bazy danych (jeśli istnieje):**

      dotnet ef database drop

      2. **Utworzenie schematu bazy danych na podstawie migracji:**

      dotnet ef database update

d. **Instalacja zależności:**

Wszystkie niezbędne pakiety są zapisane w pliku projektowym '.csproj'. Użyj poniższej komendy, aby pobrać i zainstalować odpowiednie zależności:

dotnet restore

e. **Uruchomienie projektu:**

Uruchom serwer aplikacji za pomocą poniższej komendy:

dotnet run

f. **Serwer będzie dostępny na domyślnym porcie (np. 'http://localhost:5000').**
g. **Dostęp do portalu:**

Skorzystaj z przeglądarki i przejdź pod adres lokalny, aby uzyskać dostęp do aplikacji.

6. Opis struktury projektu

a. **Projekt został logicznie podzielony na kilka warstw, zgodnie z zasadami architektury MVC:**
   i. **Controllers:**

      Kontrolery zarządzające logiką aplikacji i komunikacją pomiędzy widokami a modelem.

   ii. **Models:**

      Modele odzwierciedlające główne jednostki aplikacji i strukturę bazy danych.

   iii. **Views:**

      Warstwa prezentacji, składająca się z widoków Razor, renderujących dynamiczne strony HTML dla użytkowników.

   iv. **Data:**

      Warstwa konfiguracji Entity Framework, odpowiadająca za dostęp do danych, konfigurację modeli oraz ich relacji.

      1. **Migrations:**

         Katalog zawierający pliki migracji Entity Framework, zarządzające strukturą bazy danych.

   v. **Enums:**

      Przechowuje wszystkie enumeratory (Enums), które definiują zbiory stałych używanych w systemie. Przykładem może być 'UserRoles' określający predefiniowane role użytkowników: Administrator, Autor, Moderator, Czytelnik, Analityk.

   vi. **Areas:**

Jest to zarezerwowany katalog obsługujący funkcjonalności związane z autoryzacją i uwierzytelnianiem użytkowników. Obejmuje mechanizmy logowania, rejestracji, zarządzania kontem oraz przypisywania ról użytkownikom. W szczególności zawiera ustawienia i widoki konfiguracyjne obsługi Identity Framework, opcjonalnie dostosowane do systemu ról w projekcie.

7. Wylistowane wszystkie modele

a. ErrorViewModel:

Model używany do obsługi informacji o błędach w systemie.

i. 'RequestId' (string?):

Identyfikator żądania, który pomógł w śledzeniu występujących błędów.

ii. 'ShowRequestId' (bool):

Właściwość logiczna wskazująca, czy 'RequestId' powinno być wyświetlane.

b. EntryModel:

Model reprezentujący wyświetlenia w systemie.

i. 'Id' (int):

Unikatowy identyfikator wpisu.

ii. 'PageId' (int):

Identyfikator powiązanej podstrony.

iii. 'Page' (PageModel):

Odniesienie do podstrony, do której wpis został przypisany.

iv. 'UserId' (int?):

Identyfikator użytkownika, który utworzył wpis.

v. 'User' (UserModel?):

Odniesienie do użytkownika tworzącego wpis.

vi. 'CreatedAt' (DateTime):

Data utworzenia wpisu.

c. PageContentModel:

Model reprezentujący zawartość podstron.

i. **'Id' (int):**

   Unikatowy identyfikator zawartości.

ii. **'PageId' (int):**

   Identyfikator podstrony, dla której zawartość została utworzona.

iii. **'Page' (PageModel):**

   Odniesienie do powiązanej podstrony.

iv. **'Type' (ContentTypesEnum):**

   Typ zawartości (np. tekst, obraz).

v. **'Value' (string):**

   Treść zawartości (np. tekst lub URL obrazu).

vi. **'Order' (int):**

   Kolejność wyświetlania zawartości na stronie.

d. **UserModel:**

Model reprezentujący użytkowników w systemie.

i. **'Id' (int):**

   Unikatowy identyfikator użytkownika.

ii. **'IdentityUserId' (string):**

   Identyfikator użytkownika w systemie ASP.NET Identity.

iii. **'IdentityUser' (IdentityUser):**

   Integracja z ASP.NET Identity dla funkcji uwierzytelniania.

iv. **'CreatedAt' (DateTime):**

   Data rejestracji użytkownika.

v. **'Role' (UserRolesEnum):**

   Rola przypisana użytkownikowi (np. Administrator, Autor, Czytelnik).

vi. **'Pages' (ICollection):**

   Lista podstron, które utworzył użytkownik.

vii. **'Entries' (ICollection):**

Lista wpisów przypisanych użytkownikowi.

viii. **'Comments' (ICollection):**

Lista komentarzy napisanych przez użytkownika.

ix. **'Ratings' (ICollection):**

Lista ocen przypisanych do użytkownika.

e. **RateModel:**

Model reprezentujący oceny użytkowników.

i. **'Id' (int):**

Unikatowy identyfikator oceny.

ii. **'PageId' (int):**

Identyfikator podstrony, do której odnosi się ocena.

iii. **'Page' (PageModel):**

Odniesienie do podstrony, do której przypisana jest ocena.

iv. **'UserId' (int):**

Identyfikator użytkownika, który wystawił ocenę.

v. **'User' (UserModel):**

Odniesienie do użytkownika, który wystawił ocenę.

vi. **'CreatedAt' (DateTime):**

Data wystawienia oceny.

vii. **'Rating' (RatingsEnum):**

Wartość oceny (np. 1-5 gwiazdek).

viii. **'Status' (InteractionStatusesEnum):**

Status interakcji (np. zatwierdzony, oczekujący na moderację).

f. **PageModel:**

Model reprezentujący podstronę w systemie.

i. **'Id' (int):**

   Unikatowy identyfikator podstrony.

ii. **'UserId' (int):**

   Identyfikator użytkownika, który stworzył podstronę.

iii. **'User' (UserModel):**

   Odniesienie do użytkownika tworzącego podstronę.

iv. **'ParentPageId' (int?):**

   Identyfikator podstrony nadrzędnej (jeśli istnieje).

v. **'ParentPage' (PageModel?):**

   Odniesienie do podstrony nadrzędnej.

vi. **'CategoryId' (int):**

   Identyfikator kategorii przypisanej do podstrony.

vii. **'Category' (CategoryModel):**

   Odniesienie do kategorii, do której należy podstrona.

viii. **'CreatedAt' (DateTime):**

   Data utworzenia podstrony.

ix. **'Link' (string?):**

   Unikalny link do podstrony.

x. **'Title' (string):**

   Tytuł podstrony.

xi. **'Description' (string):**

   Treść podstrony.

xii. **'Keywords' (string):**

   Słowa kluczowe związane z podstroną (SEO).

xiii. **'Image' (string):**

   Ścieżka do obrazu przypisanego do podstrony.

xiv. **'Entries' (ICollection):**

Lista wpisów podstrony.

xv. **'Comments' (ICollection):**

Lista komentarzy podstrony.

xvi. **'Ratings' (ICollection):**

Lista ocen przypisanych do podstrony.

xvii. **'Contents' (ICollection):**

Lista zawartości podstrony.

g. **CommentModel:**

Model reprezentujący komentarze dodawane do podstron.

i. **'Id' (int):**

Unikatowy identyfikator komentarza.

ii. **'PageId' (int):**

Identyfikator podstrony, do której dodano komentarz.

iii. **'Page' (PageModel):**

Odniesienie do podstrony, na której znajduje się komentarz.

iv. **'UserId' (int):**

Identyfikator użytkownika, który dodał komentarz.

v. **'User' (UserModel):**

Odniesienie do użytkownika, który dodał komentarz.

vi. **'CreatedAt' (DateTime):**

Data utworzenia komentarza.

vii. **'Description' (string):**

Treść komentarza.

viii. **'Status' (InteractionStatusesEnum):**

Status komentarza (np. zatwierdzony, oczekujący na moderację).

h. **CategoryModel:**

Model reprezentujący kategorie podstron.

i. **'Id' (int):**

Unikatowy identyfikator kategorii.

ii. **'UserId' (int):**

Identyfikator użytkownika, który utworzył kategorię.

iii. **'User' (UserModel):**

Odniesienie do użytkownika tworzącego kategorię.

iv. **'CreatedAt' (DateTime):**

Data utworzenia kategorii.

v. **'Name' (string):**

Nazwa kategorii.

vi. **'Pages' (ICollection):**

Lista podstron przypisanych do tej kategorii.

8. Wylistowane kontrolery wraz z metodami

a. **UserController:**

Zarządza użytkownikami w aplikacji.

i. **Index(UserRolesEnum? role): IActionResult:**

Wyświetla listę użytkowników z opcjonalnym filtrowaniem po roli.

ii. **Details(int? id): IActionResult:**

Wyświetla szczegóły konkretnego użytkownika.

iii. **Create(): IActionResult:**

Wyświetla widok do utworzenia nowego użytkownika.

iv. **Create(UserModel userModel): Task:**

Tworzy nowego użytkownika (POST).

v. **Edit(int? id): IActionResult:**

Wyświetla widok do edycji użytkownika.

vi. **Edit(int id, UserModel userModel): Task:**

Edytuje istniejącego użytkownika (POST).

vii. **Delete(int? id): IActionResult:**

Wyświetla widok potwierdzenia usunięcia użytkownika.

viii. **DeleteConfirmed(int id): Task:**

Usuwa użytkownika z bazy danych (POST).

b. **RateController:**

Zarządza ocenami stron.

i. **Index(int? pageId, int? userId, RatingsEnum? rating, InteractionStatusesEnum? status): IActionResult:**

Wyświetla listę ocen z możliwością filtrowania.

ii. **Details(int? id): IActionResult:**

Wyświetla szczegóły konkretnej oceny.

iii. **Create(): IActionResult:**

Wyświetla widok do utworzenia nowej oceny.

iv. **Create(RateModel rateModel): Task:**

Tworzy nową ocenę (POST).

v. **Edit(int? id): IActionResult:**

Wyświetla widok do edycji oceny.

vi. **Edit(int id, RateModel rateModel): Task:**

Edytuje istniejącą ocenę (POST).

vii. **Delete(int? id): IActionResult:**

Wyświetla widok potwierdzenia usunięcia oceny.

viii. **DeleteConfirmed(int id): Task:**

Usuwa ocenę z bazy danych (POST).

c. **PageController:**

Zarządza stronami w CMS.

i. **Home(string? link): IActionResult:**

   Wyświetla stronę główną lub konkretną stronę pod linkiem.

ii. **AddRating(int pageId, RatingsEnum rating): Task:**

   Pozwala użytkownikowi ocenić stronę (POST).

iii. **DeleteRating(int pageId): Task:**

   Usuwa ocenę użytkownika (POST).

iv. **PushComment(int pageId, int? commentId, string description): Task:**

   Dodaje lub edytuje komentarz na stronie (POST).

v. **DeleteComment(int pageId, int commentId): Task:**

   Usuwa komentarz użytkownika (POST).

vi. **Index(int? userId, int? parentPageId, int? categoryId, string? title): IActionResult:**

   Wyświetla listę stron z możliwością filtrowania.

vii. **Details(int? id): IActionResult:**

   Wyświetla szczegóły konkretnej strony.

viii. **Create(): IActionResult:**

   Wyświetla widok do utworzenia nowej strony.

ix. **Create(PageModel pageModel, List<string?> contentValues, List contentTypes): Task :**

   Tworzy nową stronę (POST).

x. **Edit(int? id): IActionResult:**

   Wyświetla widok do edycji strony.

xi. **Edit(int id, PageModel pageModel, List<string?> contentValues, List contentTypes): Task :**

   Edytuje istniejącą stronę (POST).

xii. **Delete(int? id): IActionResult:**

   Wyświetla widok potwierdzenia usunięcia strony.

xiii. **DeleteConfirmed(int id): Task:**

Usuwa stronę z bazy danych (POST).

d. **EntryController:**

Rejestruje wizyty na stronach.

i. **Index(int? pageId, int? userId): IActionResult:**

Wyświetla listę wejść przypisanych do stron i użytkowników.

ii. **Details(int? id): IActionResult:**

Wyświetla szczegóły konkretnego wejścia.

iii. **Create(): IActionResult:**

Wyświetla widok do utworzenia nowego wejścia.

iv. **Create(EntryModel entryModel): Task:**

Tworzy nowe wejście (POST).

v. **Edit(int? id): IActionResult:**

Wyświetla widok do edycji wejścia.

vi. **Edit(int id, EntryModel entryModel): Task:**

Edytuje istniejące wejście (POST).

vii. **Delete(int? id): IActionResult:**

Wyświetla widok potwierdzenia usunięcia wejścia.

viii. **DeleteConfirmed(int id): Task:**

Usuwa wejście z bazy danych (POST).

e. **CommentController:**

Zarządza komentarzami użytkowników na stronach.

i. **Index(int? pageId, int? userId, string? description, InteractionStatusesEnum? status): IActionResult:**

Wyświetla listę komentarzy z możliwością filtrowania.

ii. **Details(int? id): IActionResult:**

Wyświetla szczegóły konkretnego komentarza.

iii. **Create(): IActionResult:**

Wyświetla widok do utworzenia nowego komentarza.

iv. **Create(CommentModel commentModel): Task:**

Tworzy nowy komentarz (POST).

v. **Edit(int? id): IActionResult:**

Wyświetla widok do edycji komentarza.

vi. **Edit(int id, CommentModel commentModel): Task:**

Edytuje istniejący komentarz (POST).

vii. **Delete(int? id): IActionResult:**

Wyświetla widok potwierdzenia usunięcia komentarza.

viii. **DeleteConfirmed(int id): Task:**

Usuwa komentarz z bazy danych (POST).

f. **CategoryController:**

Zarządza kategoriami dostępnych stron.

i. **Index(string? name, int? userId): IActionResult:**

Wyświetla listę kategorii z możliwością filtrowania.

ii. **Details(int? id): IActionResult:**

Wyświetla szczegóły konkretnej kategorii.

iii. **Create(): IActionResult:**

Wyświetla widok do utworzenia nowej kategorii.

iv. **Create(CategoryModel categoryModel): Task:**

Tworzy nową kategorię (POST).

v. **Edit(int? id): IActionResult:**

Wyświetla widok do edycji kategorii.

vi. **Edit(int id, CategoryModel categoryModel): Task:**

Edytuje istniejącą kategorię (POST).

vii. **Delete(int? id): IActionResult:**

Wyświetla widok potwierdzenia usunięcia kategorii.

viii. **DeleteConfirmed(int id): Task:**

Usuwa kategorię z bazy danych (POST).

9. Opis systemu użytkowników

a. **Użytkownik niezalogowany:**
   i. Przeglądanie stron:
      1. Może przeglądać publicznie dostępne podstrony.
      2. Nie może oceniać stron, komentować, ani korzystać z funkcji wymagających logowania.
   ii. Komentarze i oceny:
      1. Nie ma możliwości dodawania, edycji lub usuwania komentarzy i ocen.
   iii. Wyświetlenia:
      1. Wyświetlenia podstron są rejestrowane.

b. **Czytelnik (Zarejestrowany użytkownik):**
   i. Przeglądanie stron:
      1. Może przeglądać wszystkie strony dostępne dla danej roli użytkownika.
      2. Może przeglądać komentarze i oceny na stronach.
   ii. Oceny stron:
      1. Może dodawać oceny.
   iii. Komentowanie:
      1. Może dodawać komentarze do stron.
      2. Może edytować własne komentarze.
      3. Nie może usuwać komentarzy innych użytkowników.

c. **Autor:**
   i. Oprócz działań dostępnych dla czytelnika:
      1. Może tworzyć nowe podstrony.
      2. Może edytować podstrony.
      3. Może tworzyć nowe kategorie.
      4. Może edytować kategorie.

d. **Analityk:**
   i. Analityk to rola skoncentrowana na analizie danych zebranych w systemie. W tej roli użytkownik może:
      1. Przeglądać historię wizyt użytkowników oraz statystyki ruchu.
      2. Analizować oceny stron.
      3. Filtrować i analizować komentarze na stronach.

e. **Moderator:**
   i. Moderator ma uprawnienia do moderowania treści generowanych przez użytkowników. Oprócz działań dostępnych dla analityka:
      1. Może edytować, zatwierdzać oraz usuwać komentarze użytkowników.
      2. Może modyfikować lub usuwać oceny, które naruszają zasady (np. spam, nieodpowiednie treści).

f. **Administrator:**
   i. Administrator ma pełne uprawnienia do zarządzania wszystkimi elementami systemu.
   ii. Zarządzanie użytkownikami:

1. Może przeglądać wszystkich użytkowników.
2. Może przeglądać szczegóły użytkowników.

iii. Może tworzyć nowych użytkowników.
iv. Może edytować użytkowników i ich role.
v. Może usuwać użytkowników.

10. Krótka charakterystyka najciekawszych funkcjonalności

a. **WYSIWYG Edytor:**
   i. Wbudowany edytor WYSIWYG umożliwia użytkownikom edytowanie i formatowanie treści stron w prosty sposób, bez znajomości HTML.
   ii. Edytor pozwala na łatwe dodawanie formatowania tekstu, obrazków i innych multimediów.
b. **Kondycyjny Formularz Dodawania/Edycji Podstron:**
   i. Obsługa dynamicznego formularza pozwala na:
      1. Dodawanie wielu sekcji jednocześnie.
      2. Przesuwanie sekcji (zmiana kolejności).
      3. Usuwanie wybranych sekcji.
   ii. Daje autorom maksymalną elastyczność w tworzeniu i edytowaniu podstron.
c. **Zapis Zdjęć w Formacie Base64:**
   i. W procesie dodawania lub edycji formularza zdjęcia są tymczasowo przechowywane w formacie Base64.
   ii. Zapobiega to utracie zdjęć podczas błędnej walidacji formularza, dzięki czemu po ponownym wyświetleniu formularza użytkownicy nie muszą ponownie wgrywać zdjęcia.
d. **Przechwytywanie Momentu Rejestracji i Automatyczne Tworzenie Ról w UserModel:**
   i. Podczas rejestracji użytkownika automatycznie przypisywana jest odpowiednia rola:
      1. Pierwszy zarejestrowany użytkownik otrzymuje rolę Administratora.
      2. Kolejni użytkownicy rejestrujący się w systemie otrzymują rolę Czytelnika.
e. **Wyświetlanie Podstron z Wszystkimi Zależnościami:**
   i. Każda wyświetlana podstrona prezentuje wszystkie istotne dane, takie jak:
      1. Rodziców (strony nadrzędne).
      2. Dzieci (podstrony podrzędne).
      3. Kategorię, datę publikacji.
      4. Powiązane komentarze i oceny od użytkowników.
f. **Ukrywanie Przycisków dla Użytkowników Bez Danych Uprawnień:**
   i. System dynamicznie ukrywa przyciski i linki, do których użytkownik nie ma dostępu.
   ii. Poprawia to bezpieczeństwo i estetykę aplikacji, ograniczając dostęp do nieautoryzowanych miejsc.
g. **Filtrowanie na Wszystkich Listach:**
   i. Wszystkie widoki list (np. użytkownicy, strony, komentarze, oceny) posiadają zaawansowany system filtrowania.
   ii. Umożliwia wyszukiwanie i sortowanie danych według odpowiednich parametrów (np. nazwa, użytkownik, kategoria).
h. **Walidacja Linku Podstrony:**
   i. System waliduje linki podstron pod różnymi względami:
      1. Czy link nie zawiera wykluczonych słów.
      2. Czy link jest unikalny (nie jest już zajęty).

3. Czy zawiera tylko dozwolone znaki.

i. **Automatyczne Przekierowanie Przy Braku Strony Głównej:**
   i. Jeśli system nie posiada podstrony oznaczonej jako strona główna (link = 'null'),
   ii. Automatycznie przekierowuje użytkownika do strony z listą wszystkich stron.
j. **Widoczność Komentarzy Zależna od Statusu:**
   i. Komentarze o różnych statusach mają różną widoczność:
      1. Dla użytkowników widoczne są tylko komentarze zaakceptowane.
      2. Komentarze o innych statusach (np. oczekujące, odrzucone) widzi tylko ich właściciel.