Projekt bloga

Patryk Kulpiński

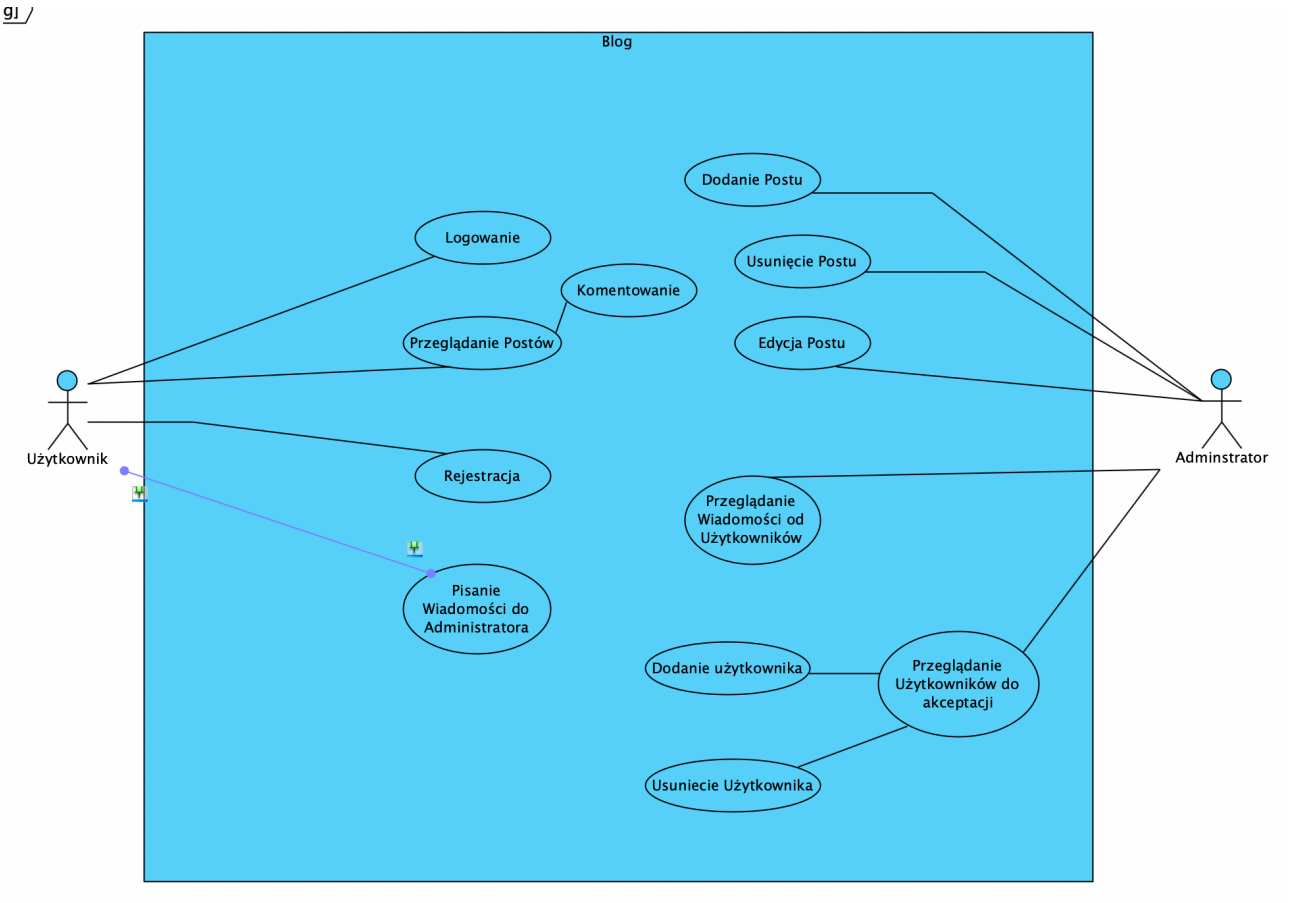
Opis aplikacji:

Aplikacja przedstawia system „bloga”. Jako właściciel/ administrator można dodawać/usuwać/edytować posty na stronie głównej. Administrator posiada dodatkowo możliwość autoryzowania użytkowników lub ich usuwania, przeglądania wiadomości. Komentarze dodawane jako gość posiadają nazwę anonim, przy komentowaniu z posiadaniem konta nazwa zostaje ustawiona na nazwę użytkownika.

Funkcjonalność systemu:

• dodawanie/edycja/usuwanie postów,
• graficzny edytor WYSIWYG dla postów uwzględniający proste znaczniki formatujące,
• system komentarzy dla niezarejestrowanych użytkowników,
• formularz kontaktowy,
• panel administracyjny z uwierzytelnianiem użytkowników,
• różne poziomy dostępu użytkowników (administrator, autor) wraz z zarządzaniem użytkownikami,
• system tagów / kategorii postów,
• wykorzystanie CAPTCHy w celu zabezpieczenia przed botami,
• wgrywanie plików na serwer.

Diagram Przypadków Użycia :



Użytkownik może logować się do systemu, napisać wiadomość do administratora, zarejestrować się jako użytkownik do akceptacji. Ma też opcje przeglądać post i zostawić pod nim komentarz.

Administrator może dodawać posty, edytować je oraz usunąć post. Może rownież przeglądać wiadomości od użytkowników i przeglądać prośby o dodanie użytkownika. Podczas przeglądania próśb może je zaakceptować bądź usunąć taką prośbę.

Opis przedstawienia postu wraz z funkcją dodania komentarza:

```python
def showSinglePost(request, id):
    post = get_object_or_404(Post, slug=id, ) #Pobranie z bazy danych postu o slugu zgodnym z zapytaniem
    posts = Post.objects.filter(slug=id) #Filtracja bazy danych po slugu
    comments = Comments.objects.filter(post=posts[0]) #Filtracja komentarzy do posta
    new_comment = None
    #print(posts)
    comment_form = CommentForm() #uruchomienie formularza do komentarzy
    if request.method == 'POST': # Jeżeli przkazana jest wartość POST
        comment_form = CommentForm(data=request.POST) # wpisać do formularza jej wartość
        if comment_form.is_valid(): #Sprawdzenie poprawności formularza
            new_comment = comment_form.save(commit=False) #Zapisanie formularza jako obiektu bazy danych bez dodania do bazy danych
            if request.user.is_authenticated: #Sprawdzenie czy użytkownik jest zalogowany
                new_comment.author = request.user.username #dodanie nazwy użytkownika przy jego zalogowaniu
            new_comment.post = get_object_or_404(Post, slug=id,) # Przypisanie komentarza do postu
            new_comment.save() #Dodanie do bazy danych komentarza
            return redirect(post.get_absolute_url()) #Powrót do postu z zapisanym komentarzem
        else:
            comment_form = CommentForm() #Jeżeli walidacja się nie powiodła usunięcie zawartości formularza
    return render(request, 'singlePost.html', {'posts': posts, 'comments': comments, 'comment_form': comment_form}) #wyświetlenie postu wraz z komentarzami
```

Algorytm przedstawia wyświetlanie postu na blogu wraz z dodawaniem komentarzy. Przy otwarciu danego postu, filtrowana jest baza danych by wyświetlić dany post i komentarze do danego posta. Przy dodawaniu komentarza sprawdzana jest poprawność formularza. Jeżeli użytkownik jest zalogowany do komentarza zostanie dodana jego nazwa użytkownika.

W formularzu komentarza dodane jest pole captchy oraz elementy body czyli sam komentarz.

```python
class CommentForm(forms.ModelForm):
    captcha = CaptchaField()

    class Meta:
        model = Comments
        fields = ('body',)
        widgets = {
            'body': TextInput(attrs={
                'class': "form-control",
                'style': 'max-width: 300px;',
                'placeholder': 'Skomentuj tutaj'
            })}
```