

QUIZZ GAME
CZERWIEC 2020

Quizz Game to projekt realizowany w ramach przedmiotu Kurs
wybranego języka programowania

Autorzy: Agnieszka Szmigulan, Sara Kubczak, Filip Kulasza, Jakub Amanowicz

1. Czym jest Quizz Game?

Quizz Game to darmowa gra pozwalająca użytkownikowi sprawdzić wiedzę w różnych dziedzinach. Jest to idealna gra do nauki i zabawy. Gracz odpowiada na pytania i zbiera punkty za prawidłowe odpowiedzi. Quizz Game to świetna rozrywka zarówno w pojedynkę jak i z przyjaciółmi dzięki systemowi rankingowemu. Obecnie dostępnych jest 6 kategorii pytań z dziedzin: Geografia, Rozrywka, Historia, Sztuka i literatura, Nauka i natura, Sport.

2. Gdzie znaleźć Quizz Game?

Gra dostępna jest na githubie pod linkiem: <https://github.com/tym1cla/Zziug>

3. Jak uruchomić Quizz Game?

Zainstaluj projekt w środowisku wirtualnym,

Następnie zainstaluj pakiety:

```
python3 -m pip install -r requirements.txt
```

Stwórz i zastosuj migrację:

```
python3 manage.py makemigrations  
python3. manage.py migrate
```

Odpal serwer:

```
python3 manage.py runserver
```

4. Główne funkcjonalności oraz ich autorzy:

- Rejestracja (Sara Kubczak)
- Logowanie (Sara Kubczak)
- Możliwość wyboru kategorii pytań (Filip Kulasza)
- Strona z informacją o Quizz Game (Agnieszka Szmigulan)
- Możliwość gry, zdobywania punktów (Jakub Amanowicz, Agnieszka Szmigulan)
- Możliwość dodawania pytań do gry jako superuser (Jakub Amanowicz)
- Wyświetlenie rankingu i punktacji top 8 graczy (Sara Kubczak, Filip Kulasza)

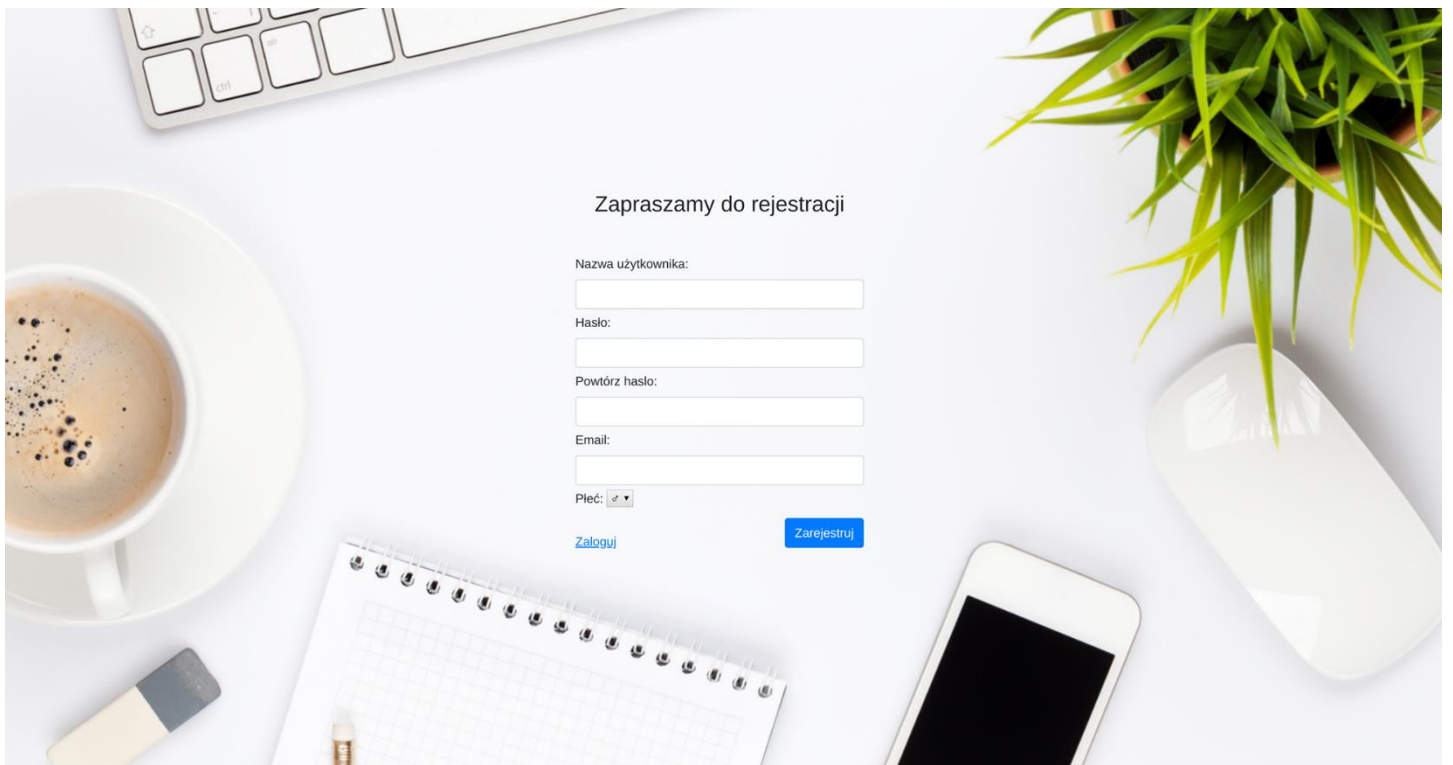
Dokładny udział osób w projekcie widać w commitach na github. Praca została podzielona równo między członków zespołu.

5. Szczegółowy opis aplikacji Quizz Game

5.1 Konfiguracja Django

Dostosowywanie uwierzytelniania, stworzenie widoków ([Quizz/game/views.py](#)), rozszerzenie domyślnego modelu użytkownika, formularze uwierzytelniania, szablony (templates) z wykorzystaniem jQuery i Bootstrapa.

5.2 Rejestracja



Użytkownik ma możliwość zarejestrowania się poprzez wypełnienie formularza.

- Formularz modelu dla modelu User: [Quizz/login/forms.py](#)
- Model użytkownika: [Quizz/login/models.py](#)

Kolumna user definiuje relację typu „jeden do jednego” i pozwala na powiązanie profilu z użytkownikiem. Użyliśmy klauzuli CASCADE dla parametru on_delete, aby po usunięciu użytkownika został również usunięty powiązany z nim profil.

- Szablony rejestracji: [Quizz/login/templates/login/register.html](#)

Ten szablon zawiera formularz, którego egzemplarz jest tworzony w widoku. Ponieważ formularz zostanie wysłany za pomocą metody POST, dołączamy znacznik szablonu {% csrf_token %} w celu zapewnienia ochrony przed atakami typu CSRF. Próbuąc wysłać formularz, pozostawiając niewypełnione jedno z pól, formularz jest uznawany za nieprawidłowy i zostanie wyświetlony komunikat błędu.

- Widoki rejestracji: [Quizz/login/views.py](#)

```
def register(request):
    if request.method == 'POST':
        register_form = RegisterForm(request.POST)
        message = "Sprawdź poprawność"
        if register_form.is_valid():
            username = register_form.cleaned_data.get('username')
            password1 = register_form.cleaned_data.get('password1')
            password2 = register_form.cleaned_data.get('password2')
            email = register_form.cleaned_data.get('email')
            sex = register_form.cleaned_data.get('sex')

            if password1 != password2:
                message = "Hasła różnią się"
                return render(request, 'register.html', locals())
            else:
                same_name_user = User.objects.filter(username=username)
                if same_name_user:
                    message = "Użytkownik o takiej nazwie już istnieje! "
                    return render(request, 'register.html', locals())

                same_email_user = User.objects.filter(email=email)
                if same_email_user:
                    message = "Podany mail jest już zarejestrowany! "
                    return render(request, 'register.html', locals())

                # create user and expand user profile
                new_user = User.objects.create_user(username=username, email=email, password=password1)
                user_profile = Profile(user=new_user, sex=sex)
                user_profile.save()

                return redirect(login_user)
        else:
            return render(request, 'register.html', locals())
    register_form = RegisterForm()
    return render(request, 'register.html', locals())
```

To jest kod widoku rejestracji użytkownika. Po wywołaniu widoku register przez żądanie GET za pomocą wywołania `register_form = RegisterForm()` tworzymy nowy egzemplarz formularza rejestracji i wyświetlamy go w szablonie. Kiedy użytkownik wyśle formularz przy użyciu żądania POST, przeprowadzane są następujące akcje:

- Utworzenie egzemplarza formularza wraz z wysłanymi danymi. Do tego celu służy polecenie `form = RegisterForm(request.POST)`.
- Sprawdzenie, czy formularz jest prawidłowy, za pomocą wywołania `form.is_valid()`.
- Jeżeli formularz jest nieprawidłowy, w szablonie wyświetlamy błędy wykryte podczas weryfikacji formularza (np. użytkownik nie wypełnił jednego z pól).
- Jeżeli wysłane dane są prawidłowe, tworzymy nowego użytkownika i wypełniamy jego profil. Następnie użytkownik zostaje przekierowany do formularza logowania.

5.3 Logowanie



Szablon logowania: [Quizz/login/templates/login/login.html](#)

Widok logowania: [Quizz/login/views.py](#)

Formularz logowania: [Quizz/login/forms.py](#)

Formularz będzie używany do uwierzytelnienia użytkownika na podstawie informacji przechowywanych w bazie danych

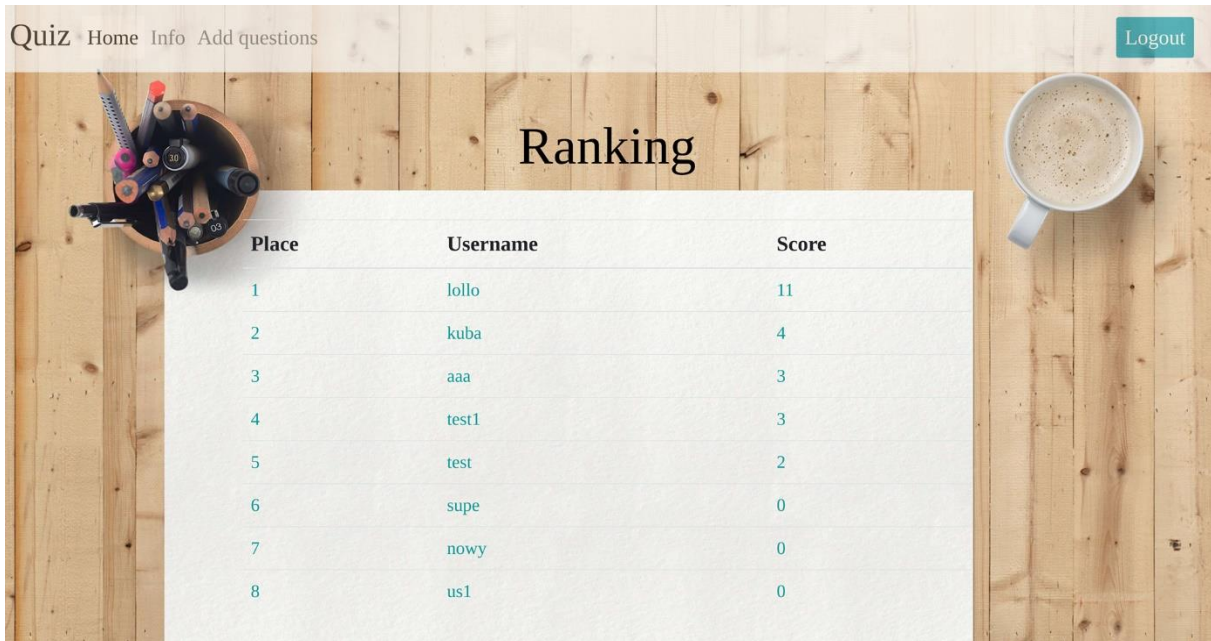
Formularz próbuje uwierzytelnić użytkownika i zgłasza błąd weryfikacji, gdy logowanie zakończy się niepowodzeniem.

Jeżeli podasz dane nieistniejącego użytkownika lub błędne hasło, Django wygeneruje komunikat o nieudanym logowaniu. Natomiast po podaniu prawidłowych danych uwierzytelniających Django przekieruje nas na main page.

Aby zalogować użytkownika, widok wykonuje poniższe akcje:

- Pobranie nazwy użytkownika i hasła z wysłanego formularza.
- Uwierzytelnienie użytkownika na podstawie danych przechowywanych w bazie danych.
- Sprawdzenie, czy konto użytkownika jest aktywne.
- Zalogowanie użytkownika w witrynie i rozpoczęcie uwierzytelnionej sekcji.

5.4 Ranking/Home



The screenshot shows a web application interface with a wooden background. At the top left, there are navigation links: Quiz, Home, Info, and Add questions. At the top right, there is a Logout button. The main heading is "Ranking". Below it is a table with three columns: Place, Username, and Score. The table lists the top 8 users and their scores.

Place	Username	Score
1	lollo	11
2	kuba	4
3	aaa	3
4	test1	3
5	test	2
6	supe	0
7	nowy	0
8	us1	0

Po zalogowaniu użytkownik zostaje przekierowany na stronę główną aplikacji. Znajduje się na niej ranking top 8 graczy wraz z ich wynikiem. Ranking znajduje się na stronie głównej w celu wyróżnienia najlepszych użytkowników.

W widoku [Quizz/login/views.py](#) pobieramy 8 najlepszych graczy wraz z ich wynikami i sortujemy od najlepszego.

```
@login_required
def index(request):
    if 'logout' in request.GET:
        logout(request)
        cache.clear()
        return redirect(login_user)

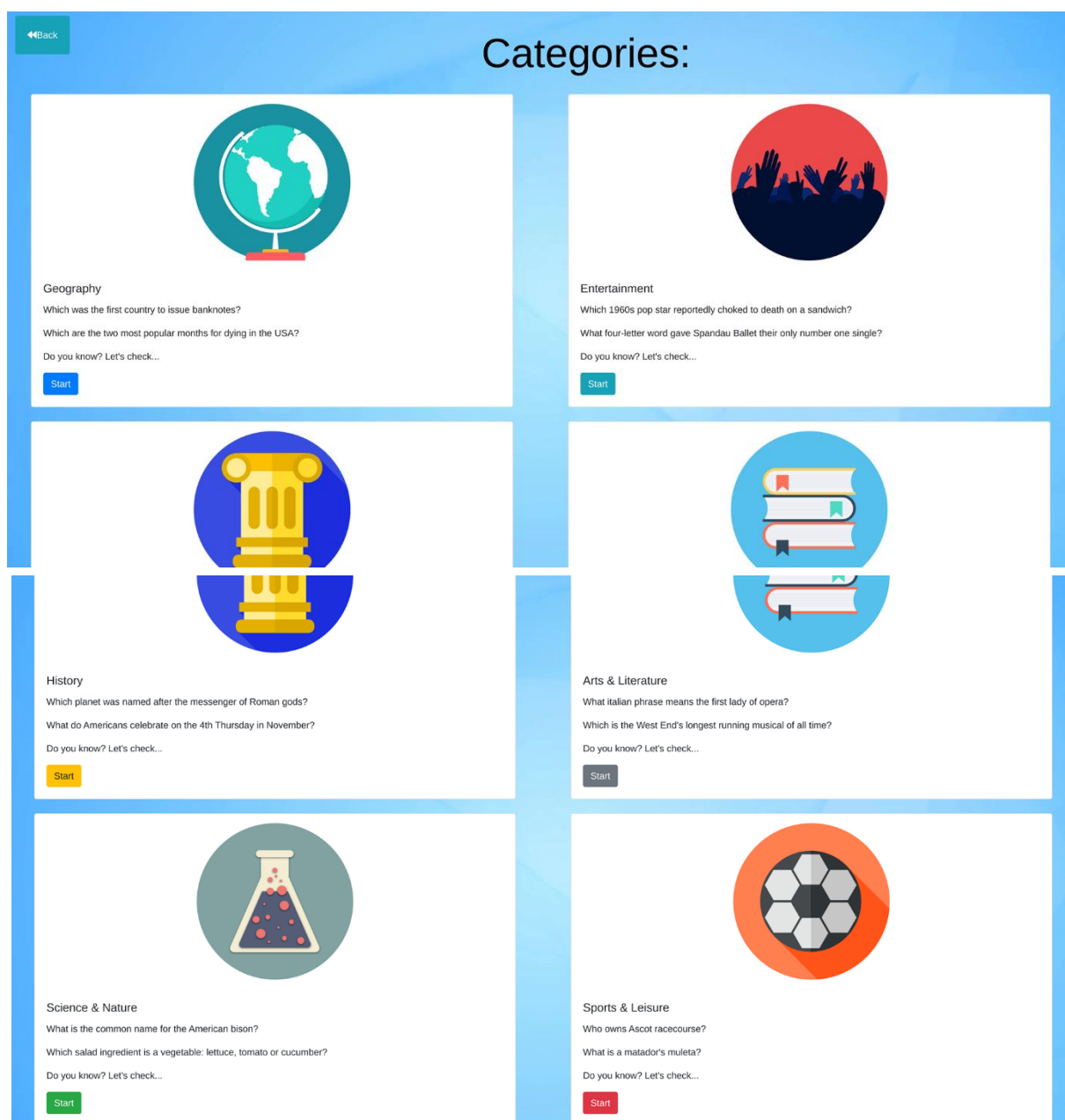
    user = Profile.objects.all().order_by('-score')[:8]

    score = Profile.objects.all().order_by('-score')[:8]

    context = {"user": user, "score": score}
    return render(request, 'index.html', context)
```

Aby następnie wyświetlić ich w szablonie: [Quizz/login/templates/index.html](#)

5.4 Kategorie pytań

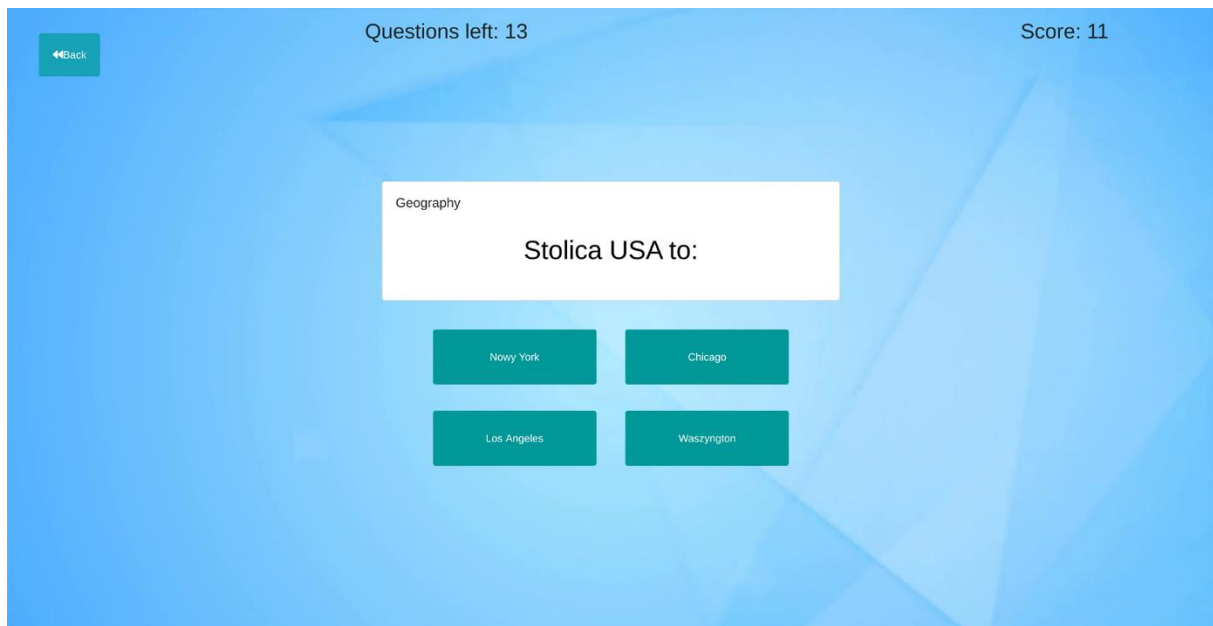


Użytkownik ma możliwość wyboru dziedziny pytań. Dostępnych jest 6 kategorii: Geografia, Rozrywka, Historia, Sztuka i literatura, Nauka i natura, Sport.

Szablon: Quizz/login/templates/login/categories.html

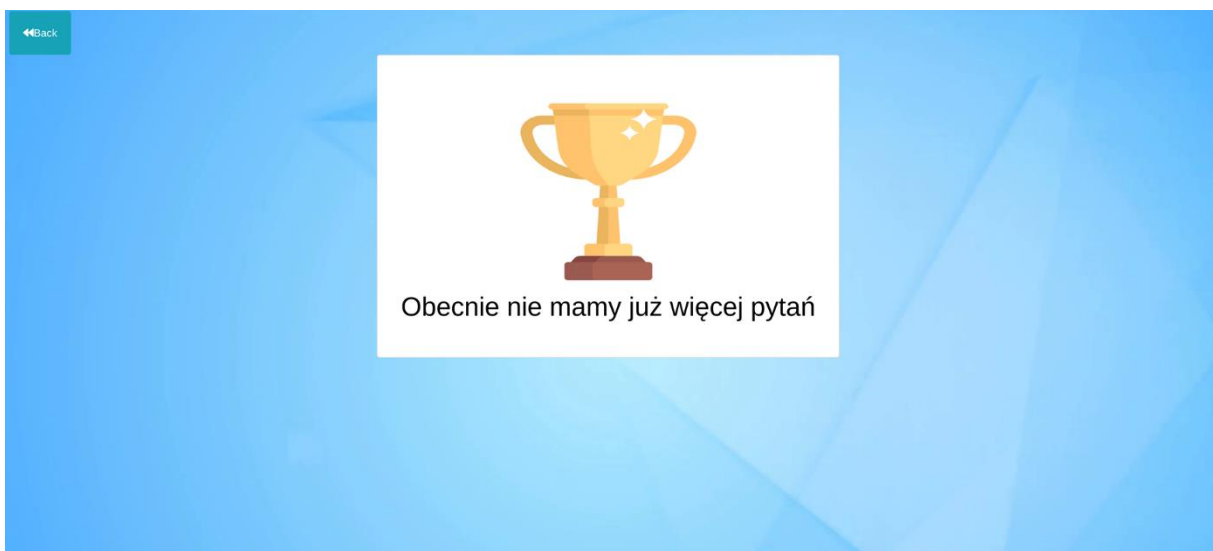
Modele: Quizz/category/models.py

5.5 Możliwość gry i zdobywania punktów



Po wybraniu danej kategorii użytkownik rozpoczyna grę. Na ekranie wyświetlone zostaje pytanie wraz z czterema odpowiedziami. Widzimy również nazwę bieżącej kategorii, wynik gracza oraz ile pytań zostało bez odpowiedzi w danej dziedzinie.

Po zaznaczeniu prawidłowej odpowiedzi wynik użytkownika wzrasta o 1 punkt a na ekranie pojawia się kolejne pytanie wybrane losowo. Gdy skończą się pytania na ekranie wyświetli się komunikat:



W sytuacji gdy użytkownik udzieli niepoprawnej odpowiedzi, pytanie znika i pojawia się losowo kolejne. Użytkownik ma możliwość przerwać grę klikając w button „back” i np. wybrać inną kategorię lub zobaczyć ranking.

Logika gry (pobieranie pytań, uzyskiwanie pytań na które użytkownik jeszcze nie odpowiedział, losowe wyświetlanie pytań, zbieranie wyniku gracza): [Quizz/category/views.py](#)

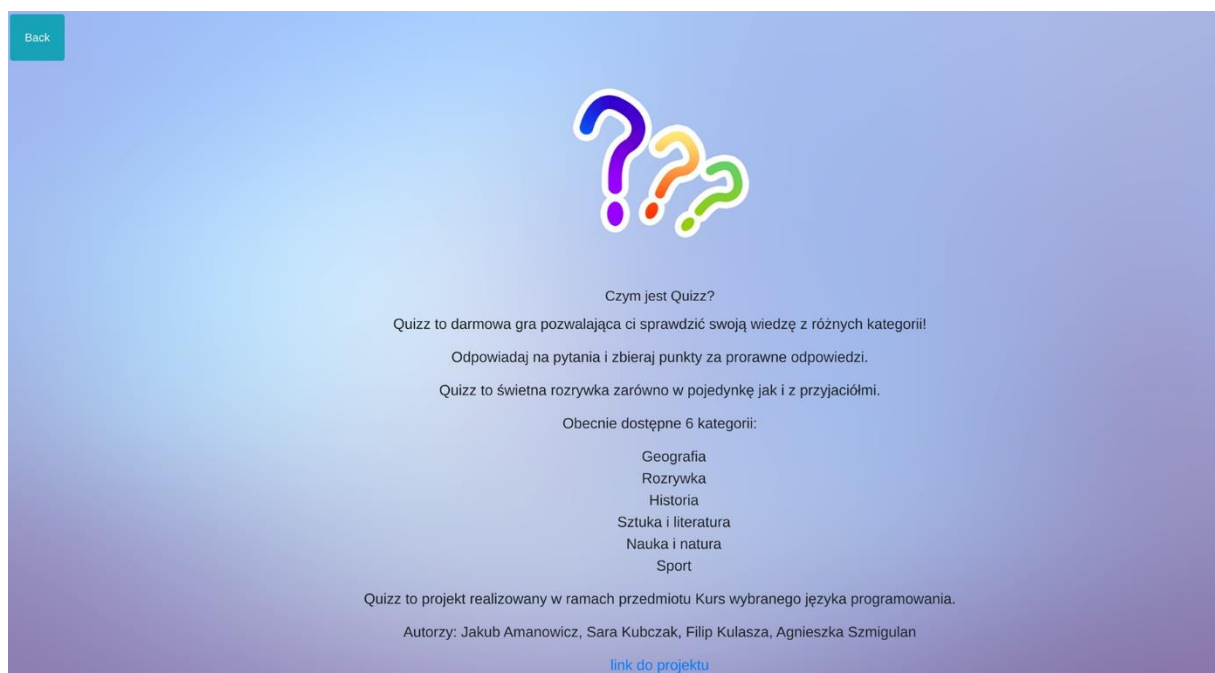
Modele pytań: [Quizz/category/models.py](#)

Formularze pytań: [Quizz/category/forms.py](#)

Szablon z pytaniem: [Quizz/login/templates/quizz.html](#)

Szablon(gdy pytania się skończą): [Quizz/login/templates/end.html](#)

5.6 Strona z informacją o Quizz Game



Każdy użytkownik w dowolnej chwili w zakładce „Info” może uzyskać informację na temat aplikacji, ilości dostępnych kategorii, autorach oraz linku do projektu.

Szablon: [Quizz/login/templates/login/info.html](#)

5.7 Możliwość dodawania pytań do gry jako superuser

The screenshot shows a web form for adding a new question. On the left, there are labels for the form fields: 'Kategoria' (Category), 'Pytanie' (Question), 'Odpowiedź 1' (Answer 1), 'Odpowiedź 2' (Answer 2), 'Odpowiedź 3' (Answer 3), 'Odpowiedź 4' (Answer 4), and 'Poprawna odpowiedź' (Correct answer). The 'Kategoria' field is a dropdown menu currently showing 'Geography'. The 'Pytanie' field is a large text area with the placeholder 'Enter question'. Below it are four input fields for answers, each with a placeholder: 'Enter first answer', 'Enter second answer', 'Enter third answer', and 'Enter fourth answer'. The 'Poprawna odpowiedź' field is a radio button. At the bottom right, there is a blue button labeled 'Zatwierdź' (Confirm).

Po zalogowaniu do naszej aplikacji przez superusera dostaje on możliwość dodania pytań do wybranej kategorii. Wystarczy, że na stronie głównej naciśnie „Add question” i zostanie przekierowany na stronę z formularzem. Superuser wybiera kategorię, zapisuje treść pytania oraz cztery odpowiedzi i zaznacza, która z nich jest poprawna. Wystarczy zatwierdzić i pytanie już jest w bazie.

Widok: [Quizz/category/views.py](#)

Szablon: [Quizz/login/templates/questions.html](#)

W kodzie dodawanie pytania wygląda to następująco([Quizz/category/views.py](#)):

```
def questions(request):
    form = QuestionForm()

    if request.method == "POST":
        form = QuestionForm(request.POST)
        if form.is_valid():
            # get posted data
            category = form.cleaned_data['category']
            question = form.cleaned_data['question']
            answer_1 = form.cleaned_data['answer_1']
            answer_2 = form.cleaned_data['answer_2']
            answer_3 = form.cleaned_data['answer_3']
            answer_4 = form.cleaned_data['answer_4']
            correct_answer = form.cleaned_data['correct_answer']

            # save data
            if category == 'Geography':
                question = Geography(question=question, answer_1=answer_1, answer_2=answer_2, answer_3=answer_3, answer_4=answer_4, correct_answer=correct_answer)
                question.save()

            if category == 'Entertainment':
                question = Entertainment(question=question, answer_1=answer_1, answer_2=answer_2, answer_3=answer_3, answer_4=answer_4, correct_answer=correct_answer)
                question.save()

            if category == 'History':
                question = History(question=question, answer_1=answer_1, answer_2=answer_2, answer_3=answer_3, answer_4=answer_4, correct_answer=correct_answer)
                question.save()

            if category == 'Literature':
                question = Literature(question=question, answer_1=answer_1, answer_2=answer_2, answer_3=answer_3, answer_4=answer_4, correct_answer=correct_answer)
                question.save()

            if category == 'Science':
                question = Science(question=question, answer_1=answer_1, answer_2=answer_2, answer_3=answer_3, answer_4=answer_4, correct_answer=correct_answer)
                question.save()

            if category == 'Sport':
                question = Sport(question=question, answer_1=answer_1, answer_2=answer_2, answer_3=answer_3, answer_4=answer_4, correct_answer=correct_answer)
                question.save()

        return render(request, 'questions.html', locals())
    else:
        return render(request, 'questions.html', locals())

return render(request, 'questions.html', locals())
```