GRA MIEJSKA

Dokumentacja projektowa PZSP2

WERSJA 2

30.11.2024

Semestr 2024Z

Zespół nr 5 w składzie:
Maciej Kaniewski
Jędrzej Kędzierski
Michał Pieńkos
Marcin Sadowski

Mentor zespołu: prof. dr hab. inż. Wojciech Mazurczyk
Właściciel tematu: dr inż. Izabela Żółtowska
Współpraca: Fundacja Bo Warto, Renata Wardecka

Spis treści

1 Wprowadzenie

1.1. Cel projektu

Celem projektu jest stworzenie aplikacji wspomagającej Fundację Bo Warto w przygotowywaniu i przeprowadzaniu gier miejskich, w których, dzięki dłuższemu okresowi trwania i mniejszej potrzebie zaangażowania wolontariuszy, uczestnicy będą mogli brać udział w dogodnym terminie.

1.2. Wstępna wizja projektu

Aplikacja webowa będzie umożliwiała stworzenie scenariusza gry na podstawie przygotowanego wcześniej przez eksperta planu gry, a następnie uruchomienie instancji gry (na podstawie wybranego scenariusza) na dany okres czasu, bez konieczności ingerencji administratora podczas jej trwania, co umożliwi na uczestnictwo w grze osobom, które nie miałyby czasu w konkretnym terminie. Scenariusz będzie definiował kolejne miejsca, które uczestnik będzie musiał odwiedzić (w postaci tekstu w poleceniu lub mapki), oraz zadania do wykonania. Zagadki będą mogły być przekazywane w rozmaitych formach (tekst, obraz, wideo lub dźwięk). Status realizacji będzie określany za pomocą rozmaitych form weryfikacji, adekwatnych do typu zadania. Podczas gry użytkownik będzie mógł wyświetlić, ile grup wykonało dane zadanie. Po zakończeniu wszystkich zadań grupa zostanie poinformowana o zakończeniu gry. Zespół będzie mógł wykonywać na raz tylko jedno zadanie w kolejności w ustalonej podczas tworzenia scenariusz. Program nie pozwoli na przejście do zadania kolejnego przed zrobieniem wcześniejszego. Po zakończeniu gry administrator będzie mógł wyświetlić raport podsumowujący. Aby przystąpić do gry będzie należało stworzyć konto, a następnie dołączyć do gry, podając liczbę uczestników grających razem. Przy rejestracji gracz zostanie poproszony o podanie adresu email w celu poinformowania go o finale akcji. Zostanie również stworzone specjalne konto administratora, które będzie umożliwiało zarządzanie grą.

2 Metodologia wytwarzania

Organizacja pracy w projekcie:

- Komunikacja:
  - Zespołu ze sobą poprzez aplikację Discord oraz Messenger
  - Z mentorem poprzez Microsoft Teams
  - Z właścicielem tematu poprzez Microsoft Teams oraz pocztę elektroniczną
- Kod aplikacji przechowywany będzie na wydziałowym GitLabie pod adresem: https://gitlab-stud.elka.pw.edu.pl/msadows1/pzsp2-gra-miejska.
- Dokumentacja oraz minutki ze spotkań przechowywane będą w plikach na kanale zespołu w aplikacji Microsoft Teams.
- Definiowanie zadań oraz przydział osób do ich wykonania realizowany będzie na platformie GitLab.
- Tworzenie oprogramowania będzie przebiegać wedle założeń modelu kaskadowego.
- Konflikty w zespole będą rozwiązywane poprzez głosowanie, a w przypadku remisu odbędzie się losowanie.
- Rola skryby będzie pełniona przez różne osoby, zmieniając się w kolejności alfabetycznej.
- Role w zespole w ujęciu Belbina:
  - Shaper: Marcin Sadowski
  - Coordinator:

- Implementer: Marcin Sadowski, Michał Pieńkos
- Plant:
- Resource Investigator:
- Evaluator: Jędrzej Kędzierski, Maciej Kaniewski
- Team Worker: Marcin Sadowski, Michał Pieńkos, Maciej Kaniewski
- Completer/Finisher: Jędrzej Kędzierski
- Specialist: Marcin Sadowski, Michał Pieńkos, Jędrzej Kędzierski

3 Analiza wymagań

Aktorzy:

- Administrator – osoba zarządzająca systemem
- Specjalista – osoba odpowiedzialna za stworzenie planu gry
- Gracz – osoba lub grupa osób wspólnie uczestnicząca w grze

3.1. Wymagania użytkownika i biznesowe

Wymagania biznesowe:

- B1: Fundacja Bo Warto potrzebuje aplikacji webowej umożliwiającej przeprowadzenie gry terenowej.

Wymagania użytkownika:

- U1: Administrator będzie mógł zdefiniować scenariusz gry.
- U2: Administrator nie będzie musiał w czasie rzeczywistym nadzorować gry.
- U3: Administrator będzie mógł wygenerować raport przebiegu gry.
- U4: Administrator będzie mógł wysłać wiadomość do uczestników danej gry.
- U5: Administrator będzie mógł ` dane uczestników gry.
- U6: Gracz będzie mógł zarejestrować się do danej gry.
- U7: Gracz będzie mógł wyrejestrować się z danej gry.
- U8: Gracz będzie mógł rozwiązywać kolejne zadania zdefiniowane w scenariuszu gry.
- U9: Gracz będzie mógł wyświetlić statystyki, obrazujące, ile grup wykonało dane zadanie.

3.2. Wymagania funkcjonalne i niefunkcjonalne

Wymagania funkcjonalne:

- F1: Aplikacja pozwoli na definiowanie scenariusza gry poprzez dodawanie zadań. [U1]
- F2: Aplikacja pozwoli na definiowanie budowy zadania (rodzaj treści i sposób zaliczenia). [U1]
- F3: Aplikacja zapisze stworzony lub właśnie tworzony scenariusz w celu jego późniejszej edycji lub stworzeniu na jego podstawie gry. [U1]
- F4: Aplikacja pozwoli na podanie nazwy scenariusza, jego opisu oraz ewentualne zdjęcia np. mapy obszaru przebiegu gry przewidzianej w scenariuszu. [U1]
- F5: Aplikacja będzie automatycznie weryfikowała spełnienie warunków zaliczenia zadania gracza. [U2]
- F6: Aplikacja pozwoli na stworzenie gry według wybranego scenariusza oraz nadanie jej nazwy z zadanym czasem rozpoczęcia oraz końca na podstawie zdefiniowanego scenariusza. [U2]

- F7: Po ukończeniu przez grupę wszystkich zadań aplikacja poinformuje ją o zakończeniu gry. [U2]
- F8: Aplikacja nie pozwoli graczowi przejść do kolejnego zadania, jeśli obecne nie zostało ukończone. [U2]
- F9: Aplikacja wygeneruje raport z wybranymi z puli przez administratora informacjami: ile grup wykonało dane zadanie, ile było osób w grupie, ile osób było ogółem, ile grup się zarejestrowało, ukończyło grę, zrezygnowało oraz listę grup z ich nazwami wraz z czasem wykonania wszystkich zadań. [U3]
- F10: Aplikacja pozwoli na wysłanie wiadomości mailowej do zarejestrowanych uczestników danej gry. [U4]
- F11: Aplikacja pozwoli na usunięcie danych gracza z bazy danych. [U5]
- F12: Aplikacja pozwoli na wprowadzenie danych w celu rejestracji oraz potwierdzi oświadczenie o RODO (nazwa, hasło, adres email, liczba osób w zespole) [U4, U6]
- F13: Aplikacja pozwoli na wyrejestrowanie zespołu z danej gry. [U7]
- F14: Aplikacja zaprezentuje treść zadania graczowi. [U8]
- F15: Aplikacja umożliwi wprowadzenie odpowiedzi w formie wybrania odpowiedzi, wpisania jej lub określenie lokalizacji zespołu. [U8]
- F16: Aplikacja pozwoli na wyświetlenie, ile zespołów ukończyło dane zadanie. [U9]

Wymagania niefunkcjonalne:

- NF1: Aplikacja będzie mogła pracować na istniejącym serwerze fundacji (minimalizacja kosztów).
- NF2: Aplikacja nie będzie wymagała instalacji – aplikacja webowa.
- NF3: Aplikacja powinna szybko reagować na akcje użytkownika. Czas przesyłania odpowiedzi czy sprawdzania statystyk nie powinien przekraczać 2 sekund.
- NF4: Aplikacja powinna być dostosowana do uruchomienia na komputerze oraz telefonie.
- NF5: Interfejs powinien być możliwy do obsługi bez wcześniejszej jego znajomości.
- NF6: Użytkownicy powinni mieć dostęp tylko do funkcji i danych, które są do nich przypisane.
- NF7: Aplikacja powinna działać na popularnych przeglądarkach takich jak Google, FireFox, Safari, Opera czy Edge.
- NF8: Aplikacja powinna umożliwiać stworzenie scenariusza gry bez specjalistycznej wiedzy technicznej.

3.3. Przypadki użycia

Biznesowe przypadki użycia

PB1: Stworzenie scenariusza

Aktorzy: administrator (systemowy), specjalista (biznesowy)

Scenariusz główny:

1. Specjalista wymyśla plan gry miejskiej i przekazuje go administratorowi.

2. Administrator poprzez interfejs aplikacji tworzy scenariusz, nadając mu nazwę, opis oraz dodając kolejne zadania, definiując ich kolejność.

PB2: Stworzenie i rozpoczęcie gry

Aktorzy: administrator (systemowy), gracz (biznesowy, systemowy)

Scenariusz główny:

1. Administrator tworzy instancję gry na podstawie wybranego scenariusza określając jej nazwę, datę rozpoczęcia oraz zakończenia.
2. Gra staje się dostępna dla graczy od zdefiniowanej daty rozpoczęcia.
3. Gracz rejestruje się do gry.

PB3: Przebieg gry

Aktorzy: gracz (biznesowy, systemowy)

Scenariusz główny:

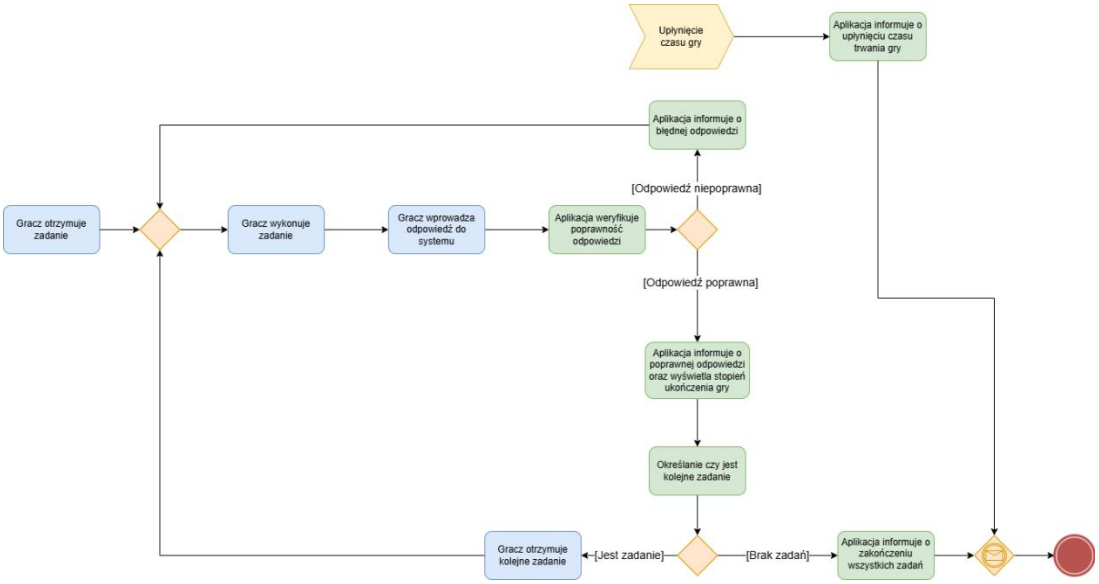
1. Aplikacja wyświetla graczowi zadanie do wykonania.
2. Gracz wykonuje zadanie.
3. Gracz wprowadza odpowiedź do systemu.
4. Aplikacja weryfikuje spełnienie warunków zaliczenia zadania.
5. Aplikacja informuje o poprawnym wykonaniu zadania oraz stopniu ukończenia całej gry.
6. W przypadku, gdy są dostępne kolejne zadania powrót do punktu 1. (Zilustrowane na Rysunek 1)
7. Aplikacja informuje gracza o zakończeniu wszystkich zadań.

Scenariusz alternatywny – błędna odpowiedź:

1. (1-4) Jak w scenariuszu głównym.
5. Aplikacja wyświetla komunikat o wprowadzeniu niepoprawnej odpowiedzi.
6. Powrót do kroku 2 scenariusza głównego.

Scenariusz alternatywny – upłynięcie czasu gry:

1. (1-4) Jak w scenariuszu głównym.
5. Aplikacja informuje gracza o upłynięciu czasu trwania gry oraz o stopniu ukończenia wszystkich zadań.



Rysunek 1 – Diagram aktywności PB3: Przebieg gry

PB4: Zakończenie gry

Aktorzy: administrator (systemowy), gracz (biznesowy, systemowy)

Scenariusz główny:

1. System wysyła do graczy informacje o miejscu i czasie finału.
2. System generuje raport z gry.
3. Administrator usuwa dane graczy.

Systemowe przypadki użycia

FU1: Tworzenie scenariusza

Wspiera procedurę PB1
Funkcja korzysta z FU3
Aktorzy: administrator

Scenariusz główny:

1. Administrator loguje się do systemu za pomocą funkcji FU3.
2. Administrator wybiera opcję tworzenia scenariusza lub kontynuacji jego tworzenia (następuje wtedy przejście do momentu, w którym skończył wcześniej edytować).
3. Administrator wpisuje nazwę scenariusza.
4. Administrator wpisuje opis scenariusza i ewentualnie załącza zdjęcie mapy obszaru, na którym ma odbyć się gra.
5. Administrator wybiera dodanie etapu (kolejność dodawania etapów, definiuje kolejność ich wykonywania).

6. Administrator wpisuje treść polecenia.
7. Administrator opcjonalnie dodaje dodatkową treść (tekst, zdjęcie, wideo lub audio).
8. Administrator wybiera typ oczekiwanej odpowiedzi (wybranie odpowiedniej opcji, wpisanie tekstu).
9. Administrator wybiera dodanie etapu (powrót do punktu 5) lub zapisanie scenariusza (przejście do punktu 10).
10. System zapisuje scenariusz do pliku tekstowego.

FU2: Rejestracja gracza

Wspiera procedurę: PB2
Funkcja korzysta z FU3
Aktorzy: gracz

Scenariusz główny:

1. Gracz wybiera opcję zarejestruj się.
2. System prezentuje formularz rejestracyjny.
3. Gracz wprowadza swoje dane do formularza.
4. Gracz wybiera zarejestruj się.
5. System weryfikuje poprawność danych.
6. System wyświetla komunikat o sukcesie.
7. System przekierowuje na stronę logowania FU3.

FU3: Logowanie

Wspiera procedurę: PB1, PB2, PB3, PB4
Aktorzy: administrator lub gracz

Scenariusz główny:

1. Użytkownik wybiera opcję logowania.
2. System wyświetla okno logowania.
3. Użytkownik wprowadza swoje dane logowania.
4. Użytkownik wybiera opcję zaloguj.
5. System weryfikuje dane logowania porównując je z danymi w bazie danych.
6. W przypadku logowania administratora system przekierowuje na panel administratora, natomiast w przypadku logowania gracza system przekierowuje na ekran główny gry zawierający nazwę gry, opis oraz zdjęcie.

Scenariusz alternatywny – niepoprawne dane logowania:

1. (1-5) Jak w scenariuszu głównym.
6. System wyświetla informację o wprowadzeniu niepoprawnych danych logowania.
7. Powrót do kroku 3 scenariusza głównego.

Scenariusz alternatywny – użytkownik zapomniał danych logowania:

1. (1-5) Jak w scenariuszu głównym.
6. System wyświetla informację o wprowadzeniu niepoprawnych danych logowania.
7. Użytkownik wybiera opcję: „nie pamiętam hasła”.
8. System wysyła na maila użytkownika nowe hasło.

FU4: Stworzenie i rozpoczęcie gry

Wspiera procedurę: PB2
Funkcja korzysta z FU3
Aktorzy: administrator

Scenariusz główny:

1. Administrator loguje się do systemu za pomocą funkcji FU3.
2. Administrator wybiera opcję stwórz grę.
3. Aplikacja otwiera przegląd scenariuszy.
4. Administrator przegląda listę i wybiera preferowany scenariusz.
5. System prezentuje ekran wyboru nazwy gry, daty rozpoczęcia i zakończenia.
6. Administrator wprowadza dane.
7. Administrator wybiera zaplanuj grę.
8. System tworzy grę i rozpoczyna ją w określonym czasie.

FU5: Uczestnictwo w grze

Wspiera procedurę: PB3
Funkcja korzysta z FU3
Aktorzy: gracz

Scenariusz główny:

1. Gracz loguje się do systemu za pomocą funkcji FU3.
2. Jeśli gracz nie zarejestrował się do gry to rejestruje swój zespół, podając liczbę członków.
3. Możliwość wyboru operacji [FU7, FU8]
4. Gracz wykonuje dostępne zadanie za pomocą funkcji FU6.
5. Powrót do kroku 3.

Scenariusz alternatywny – brak kolejnych zadań:

1. (1-3) Jak w scenariuszu głównym.
4. System informuje gracza o zakończeniu gry.

FU6: Wykonanie zadania

Wspiera procedurę: PB3
Funkcja specjalizująca FU5
Aktorzy: gracz

Scenariusz główny:

1. System wyświetla graczowi zadanie.
2. Gracz wykonuje zadanie.
3. Gracz wprowadza odpowiedź do systemu.
4. System weryfikuje odpowiedź porównując ją ze zdefiniowanymi kryteriami.
5. System wyświetla informację o poprawnym rozwiązaniu zadania.

Scenariusz alternatywny – błędna odpowiedź:

1. (1-4) Jak w scenariuszu głównym.
5. System wyświetla informację o niepoprawnym rozwiązaniu zadania.
6. Powrót do kroku 2 scenariusza głównego.

FU7: Wyświetlenie statystyk ukończenia zadań

Wspiera procedurę: PB3
Funkcja rozszerza funkcję FU5 w kroku 2. (w punkcie możliwość wyboru operacji)
Aktorzy: gracz

Scenariusz główny:

1. Gracz wybiera opcję wyświetl statystyki.
2. System prezentuje statystyki (ile zespołów ukończyło dane zadanie).
3. Gracz wybiera opcję powrót, aby zamknąć widok statystyk.

FU8: Wyrejestrowanie gracza

Wspiera procedurę: PB3
Funkcja rozszerza funkcję FU5 w kroku 2. (w punkcie możliwość wyboru operacji)
Aktorzy: gracz

Scenariusz główny:

1. Gracz wybiera opcję wyrejestruj.
2. System wyświetla zapytanie czy gracz jest pewny.
3. Gracz potwierdza chęć usunięcia konta.
4. System usuwa dane gracza z bazy danych.
5. System wyświetla stronę główną.

Scenariusz alternatywny – Wycofanie się gracza:

1. (1-2) Jak w scenariuszu głównym.

3. Gracz wybiera opcję wycofania się z usunięcia konta.
4. System zamyka okno.

FU9: Zakończenie gry

Wspiera procedurę: PB4
Funkcja korzysta z FU3
Aktorzy: administrator, czas

Scenariusz główny:

1. System sprawdza, że upłynął czas przebiegu gry terenowej.
2. System przestaje przesyłać kolejne zadania, zamiast tego wysyła informacje o upłynięciu czasu trwania gry.
3. Administrator loguje się do systemu za pomocą funkcji FU3.
4. Możliwość wyboru operacji [FU10, FU11]

FU10: Rozesłanie informacji o finale gry

Wspiera procedurę: PB4
Funkcja rozszerza funkcję FU9 w kroku 4. (w punkcie możliwość wyboru operacji)
Aktorzy: administrator, gracz

Scenariusz główny:

1. System wyświetla formularz informacyjny.
2. Administrator wypełnia formularz informacjami o miejscu i czasie finału gry.
3. Administrator wybiera opcję roześlij.
4. System rozsyła informacje zawarte w formularzu na adresy e-mail zarejestrowanych graczy.

FU11: Generowanie raportu

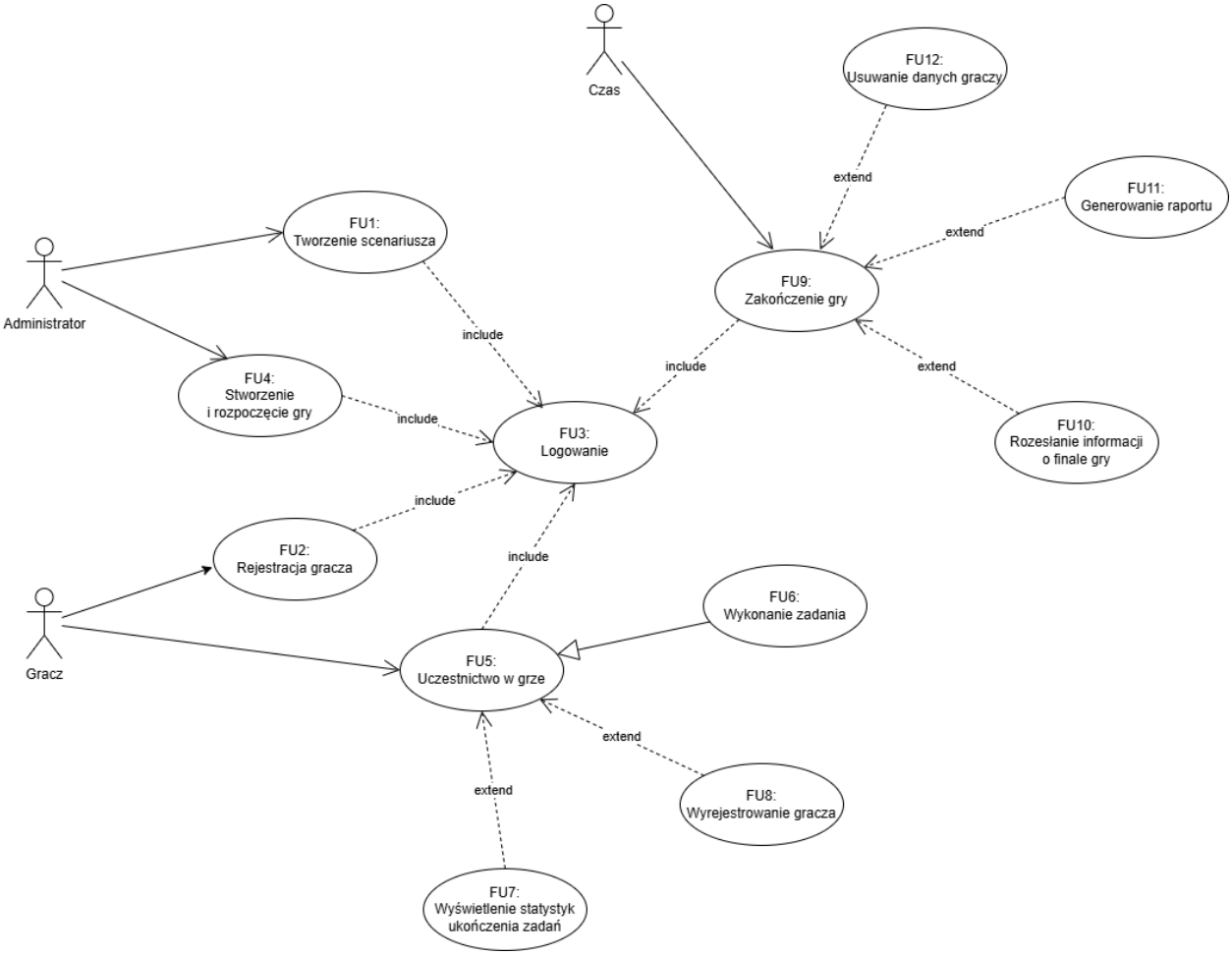
Wspiera procedurę: PB4
Funkcja rozszerza funkcję FU9 w kroku 4. (w punkcie możliwość wyboru operacji)
Aktorzy: administrator

Scenariusz główny:

1. Administrator wybiera opcję generuj raport gry.
2. Administrator wybiera dane do zawarcia w raporcie.
3. System przygotowuje statystyki dotyczące przebiegu gry.
4. System zapisuje statystyki do pliku pdf.

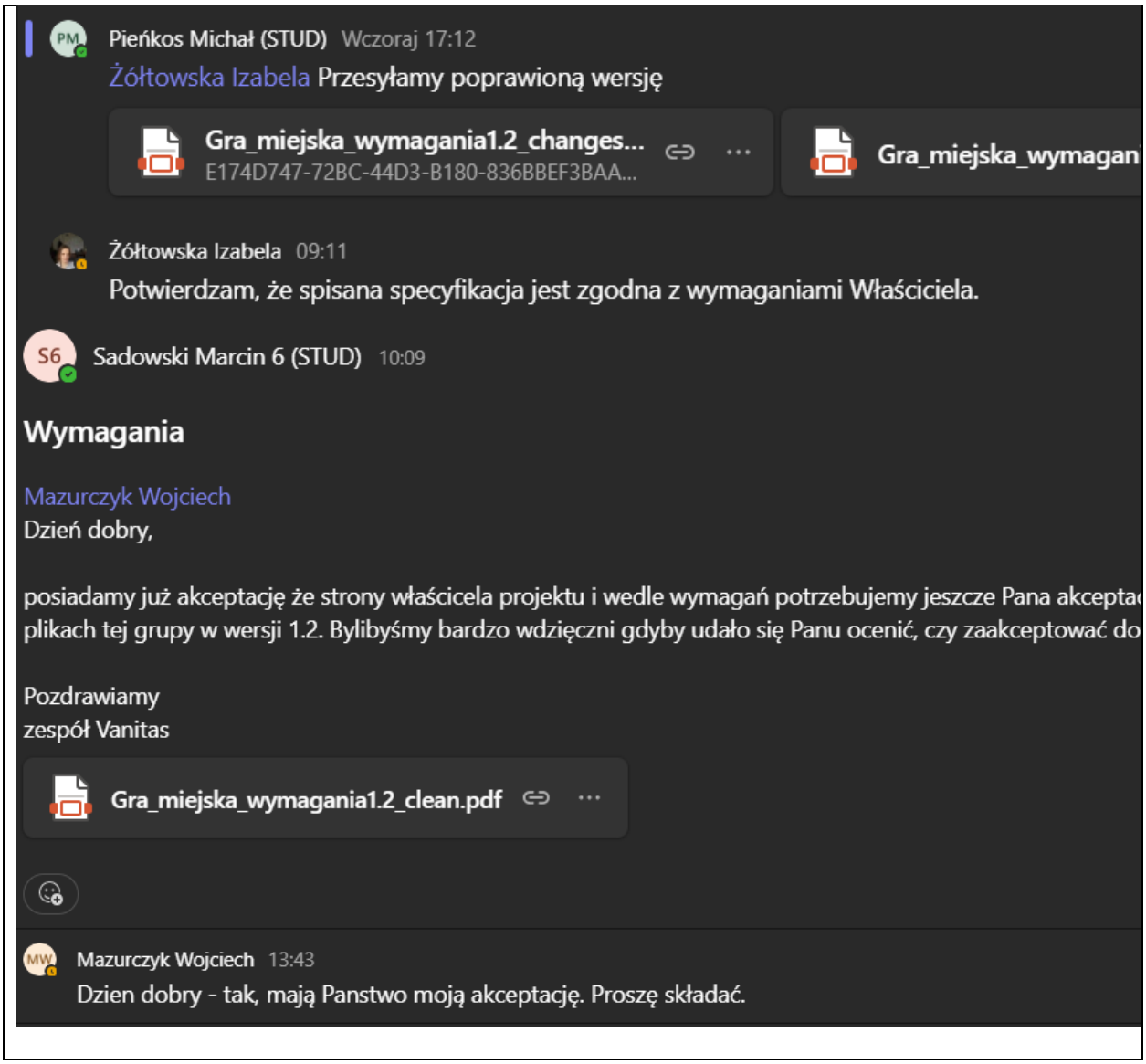
Diagram systemowych przypadków użycia

Relacje pomiędzy systemowymi przypadkami użycia zostały przedstawiony na Rysunek 2 – Diagram systemowych przypadków użycia



Rysunek 2 – Diagram systemowych przypadków użycia

3.4. Potwierdzenie zgodności wymagań



4 Definicja architektury

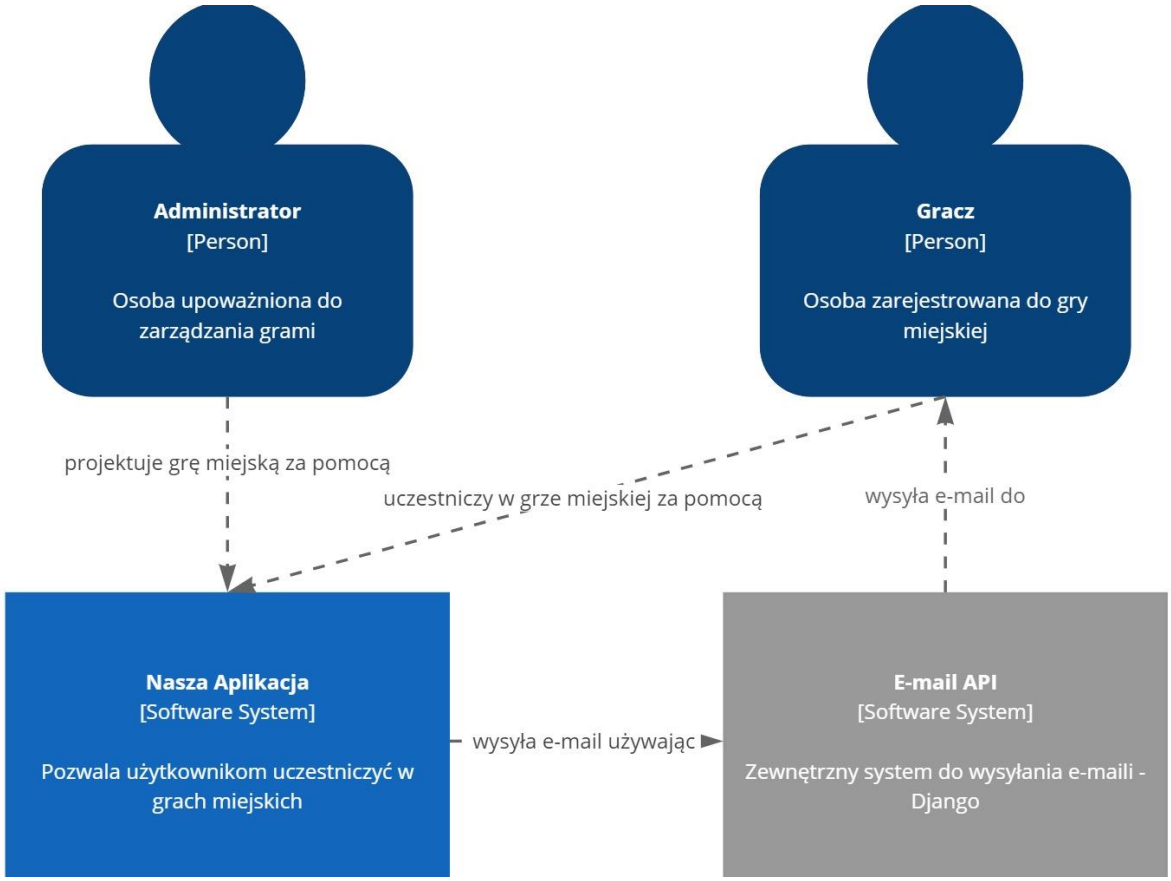
Zastosowana architektura: Architektura trójwarstwowa

Wizualizacja stworzona przy pomocy Miro: [Link do wizualizacji]

Plan struktury systemu przedstawiony za pomocą modelu C4:

Warstwa C1 – Kontekst (Rysunek 3):

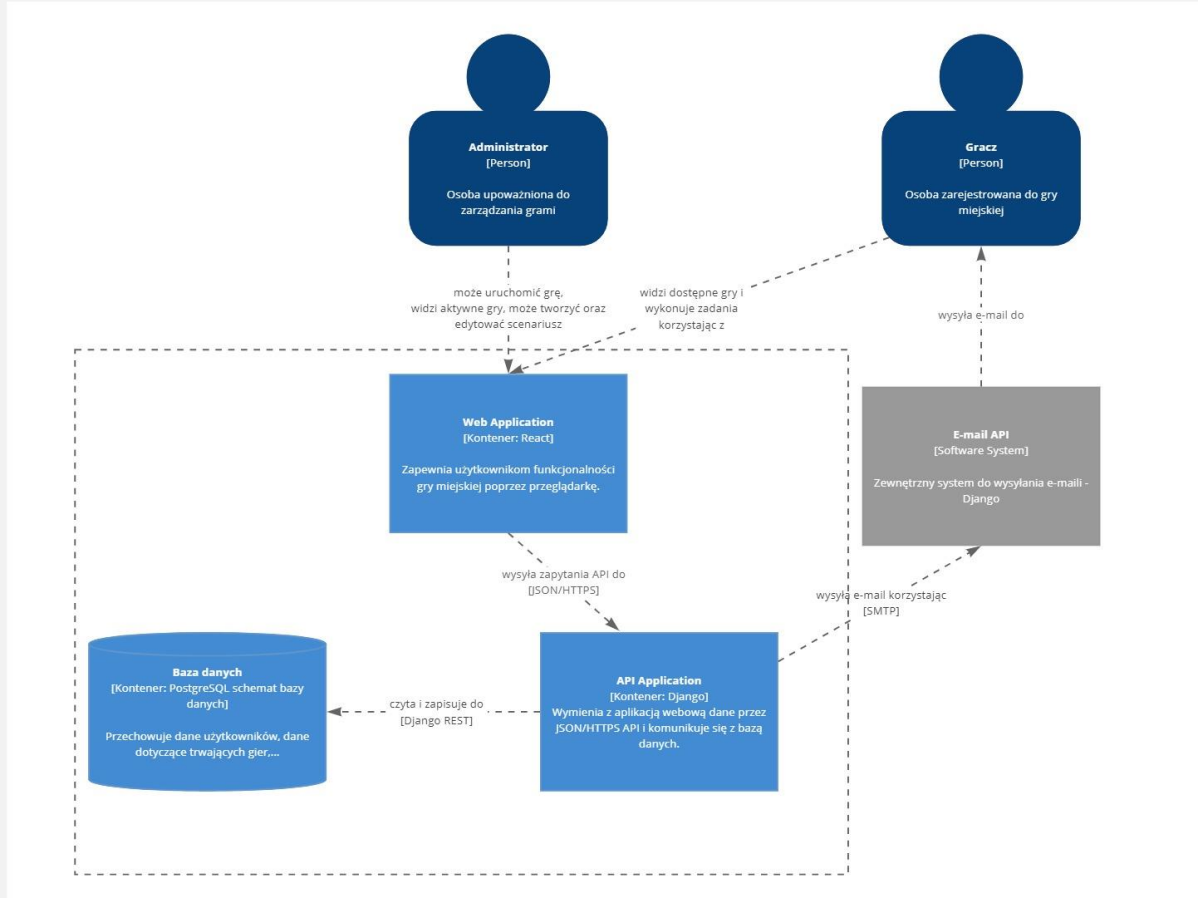
- W interakcję z naszą aplikacją będzie wchodził administrator (zarządzający grami) oraz użytkownik (uczestniczący w stworzonych grach)
- Do wysyłania e-maili do zarejestrowanych użytkowników z informacjami dotyczącymi zakończenia gry aplikacja będzie wykorzystywała E-mail API



Rysunek 3 - Kontekst

Warstwa C2 – Kontenery (Rysunek 4):

- Aplikacja będzie implementowała szablon architektoniczny architektury trójwarstwowej i składała się z trzech głównych elementów: bazy danych, serwera API i aplikacji przeglądarkowej. Komunikują się one między sobą za pomocą protokołu HTTP.

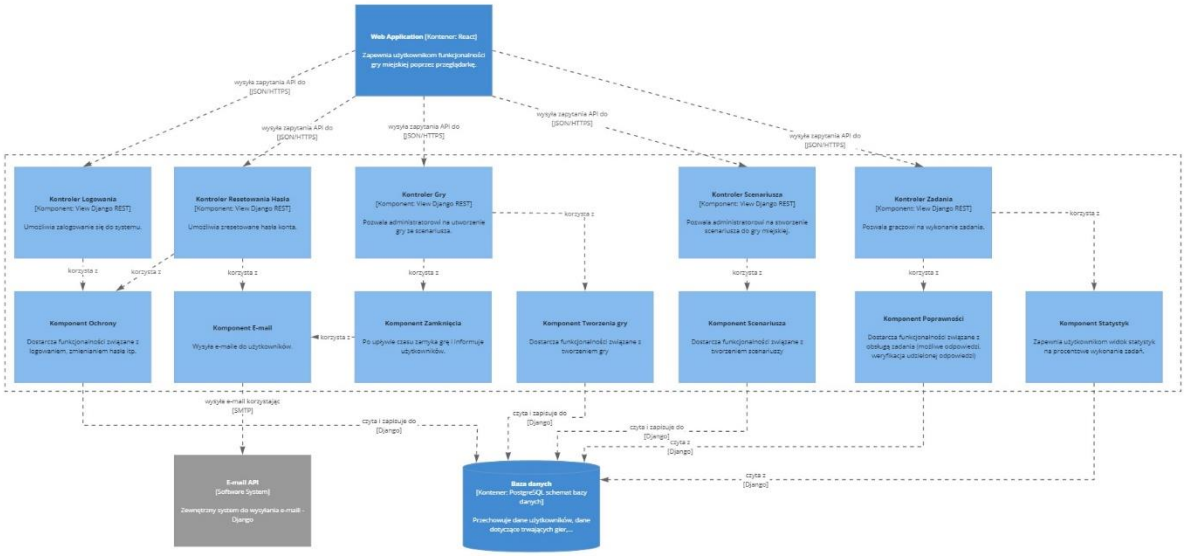


Rysunek 4 - Kontenery

Warstwa C3 – Komponenty:

Komponent backend API (Rysunek 5):

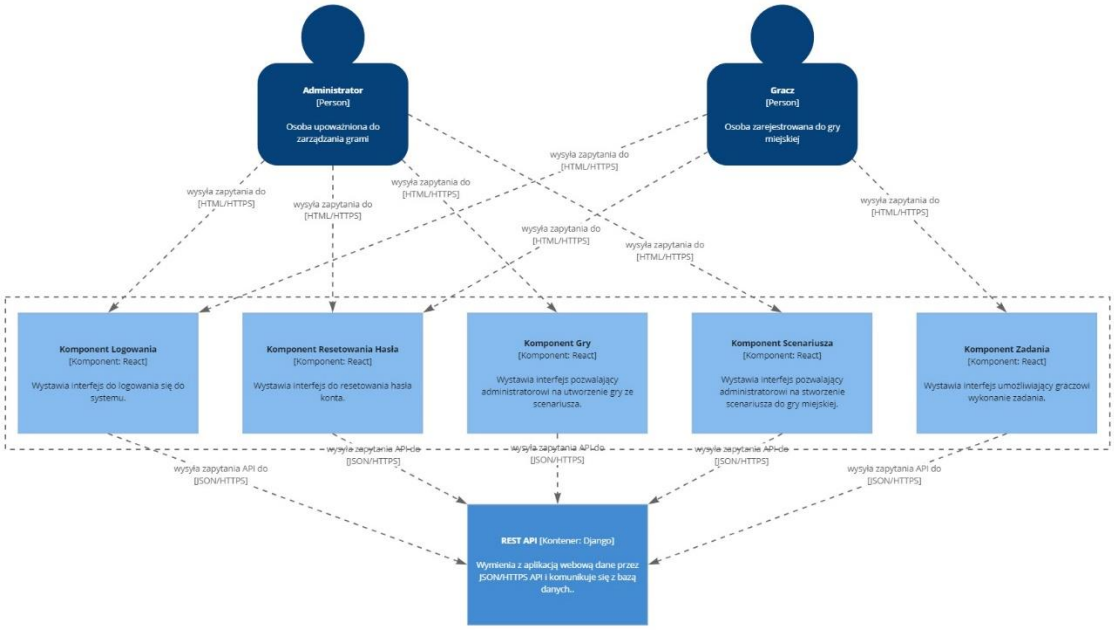
- Odpowiednie kontrolery wystawiają API do komunikacji z frontendem i w odpowiedzi na zapytania wywołują właściwe komponenty
- Komponenty implementują odpowiednie funkcjonalności i komunikują się z bazą danych



Rysunek 5 - Komponent backend

Komponent frontend (Rysunek 6):

- Komponent jest odpowiedzialny za “konwersację” z użytkownikiem. Ma za zadanie dostarczać mu interfejs dla poszczególnych czynności, a także reagować na ich wykonanie, wysyłając odpowiednie zapytania do backendu API.

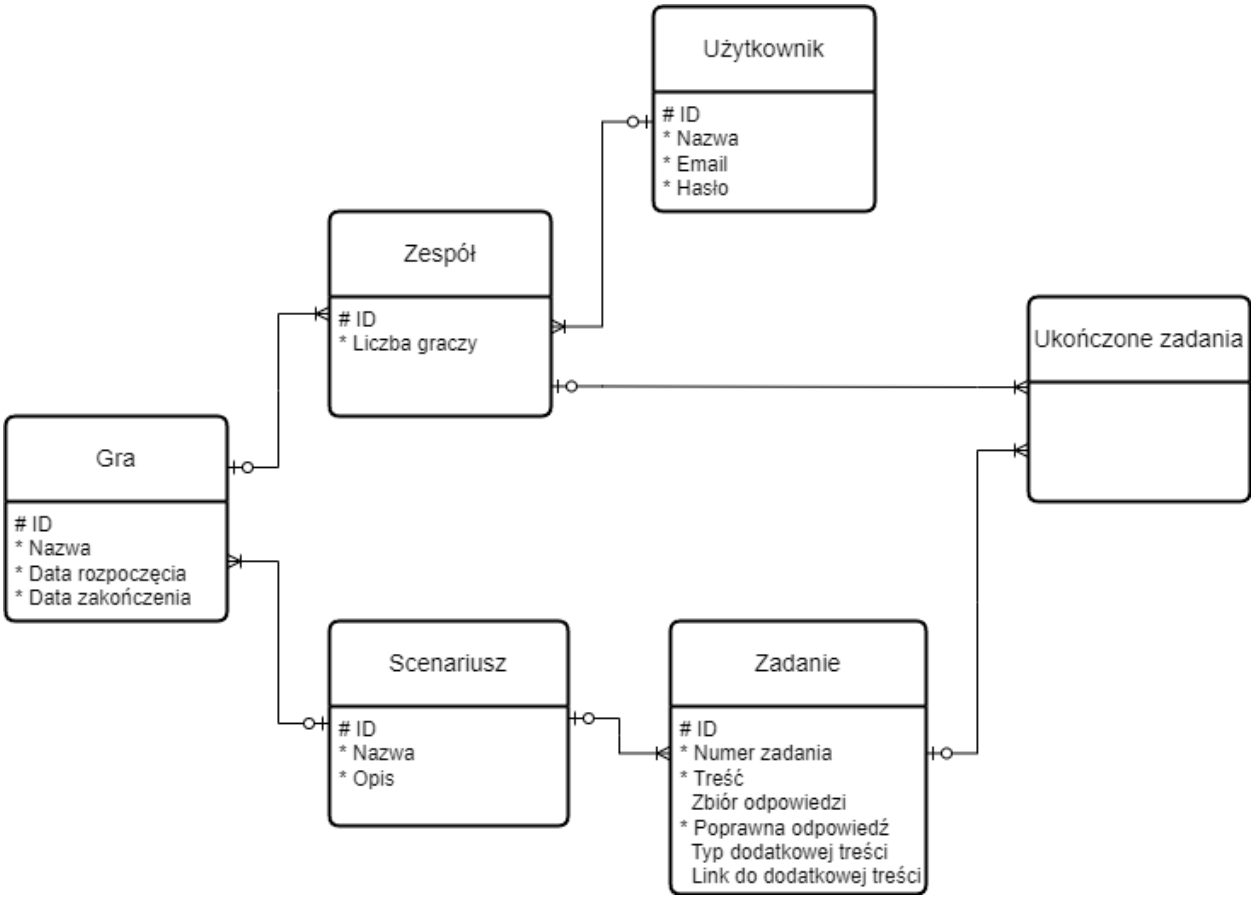


Rysunek 6 - Komponent frontend

5 Dane trwałe

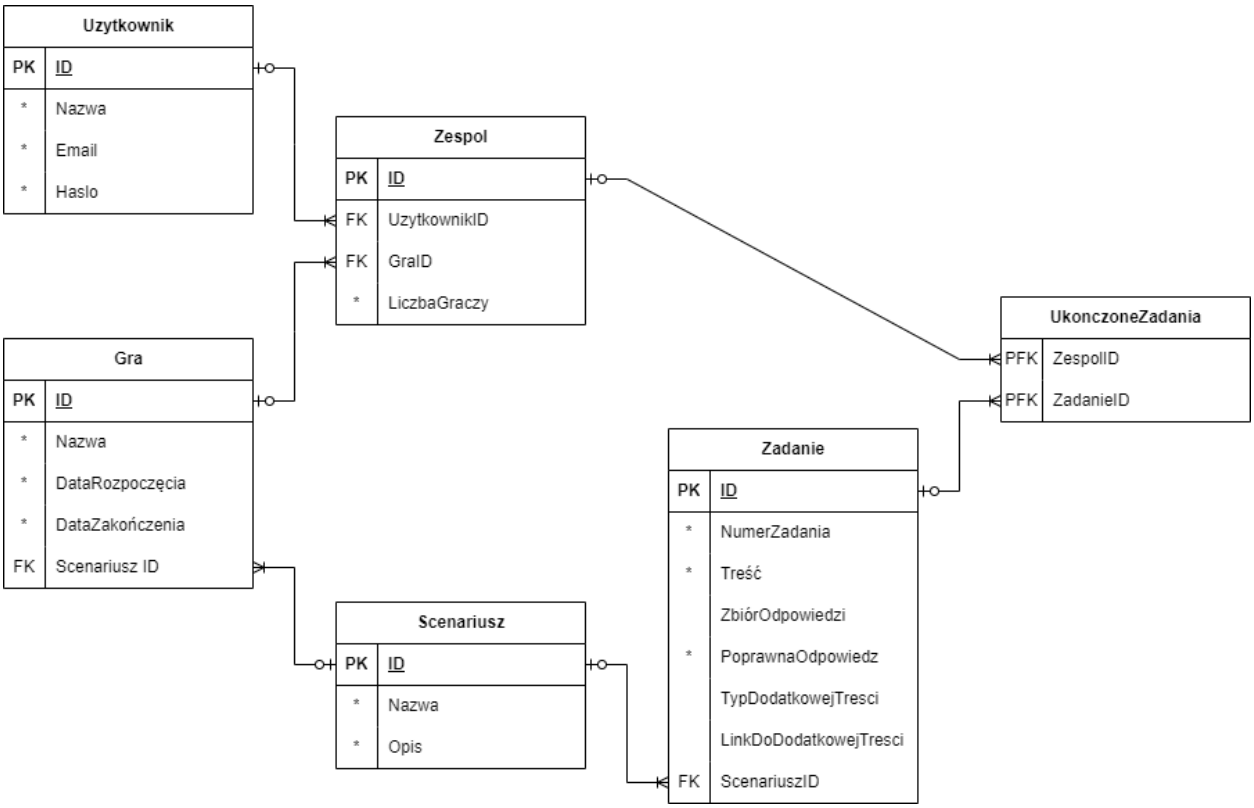
5.1. Modele danych

– Model ER dla bazy danych



Rysunek 7 – Model ER dla bazy danych

– Model relacyjny dla bazy danych



Rysunek 8 – Model relacyjny dla bazy danych

5.2. Przetwarzanie i przechowywanie danych

Podczas realizacji projektu planowane jest korzystanie z bazy danych PostgreSQL. Komunikacja z nią realizowana będzie przy pomocy Django ORM. Planowane jest stworzenie perspektywy zmaterializowanej, przedstawiającej statystyki gry (ile zespołów ukończyło dane zadanie), która będzie aktualizowana po zakończeniu zadania przez zespół.

6 Specyfikacja analityczna i projektowa

Link do repozytorium:

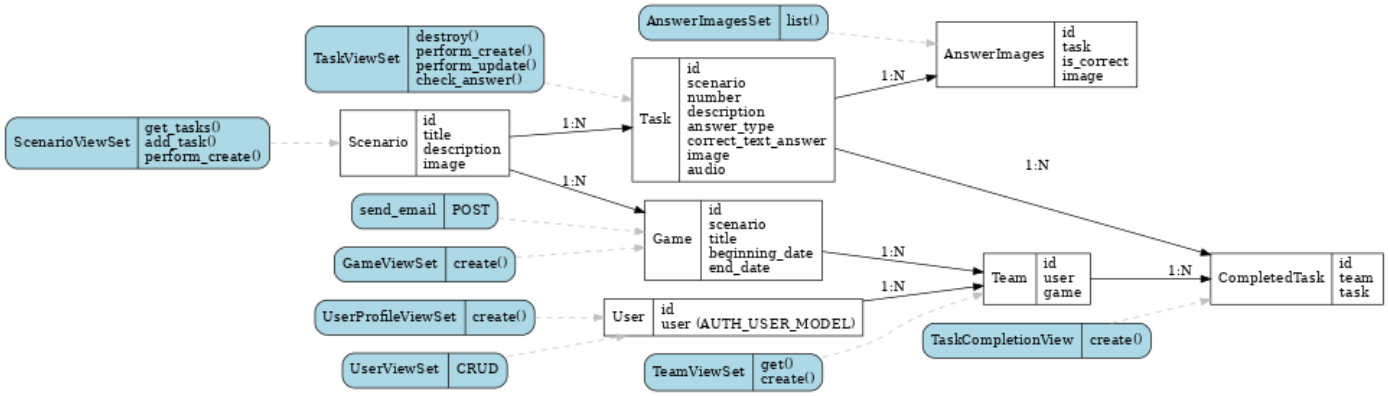
https://gitlab-stud.elka.pw.edu.pl/msadows1/pzsp2-gra-miejska

Obowiązkowo określenie metod realizacji: języki programowania, frameworki, środowisko programowania/ uruchamiania/ wdrażania, środowisko ciągłej integracji]

Realizacja projektu:

Projekt zrealizowany został przy pomocy języka programowania **Python.** Program został zrealizowany przy wsparciu frameworku **Django.** Cały kod pisany był w środowisku **VSCode** oraz uruchamiany przy pomocy **Dockera.**

Struktura projektu:



Rysunek 9 – Diagram klas w projekcie

Statystki projektu wygenerowane przy użyciu narzędzia *"cloc"*:

Language	files	blank	comment	code
SVG	4	2	2	82035
Python	32	417	106	1787
JSX	19	161	15	1470
CSS	6	87	5	494
YAML	1	14	0	39
Dockerfile	2	11	0	18
SUM:	64	692	128	85843

Rysunek 10 - Statystyki plików odpowiedzialnych za główną logikę backendu i frontendu

Language	files	blank	comment	code
JavaScript	18262	231075	432184	2857619
JSON	2350	82	0	358308
TypeScript	3546	25374	147922	211453
Markdown	1948	79372	878	203373
Python	714	33472	45957	170811
SVG	8	2	3	82127
HTML	46	35	38	8382
Text	52	222	0	5503
CSS	33	710	518	2584
YAML	164	223	115	2219
Bourne Shell	59	289	127	1623
JSX	19	161	15	1470
PowerShell	54	103	579	1274
CoffeeScript	16	386	40	966
DOS Batch	55	121	1	835
Windows Module Definition	5	83	0	451
INI	21	76	0	320
PHP	1	13	19	124
XML	7	0	1	98
EJS	2	4	0	81
make	6	35	20	79
Bourne Again Shell	2	11	1	43
Dockerfile	4	16	0	34
Nix	1	1	0	19
peg.js	1	14	5	16
SUM:	27376	371880	628423	3909812

Rysunek 11 - Pełne statystyki projektu

W projekcie znajduje się 82 testów jednostkowych, o których więcej informacji znajduje się w osobnym rozdziale.

7 Projekt standardu interfejsu użytkownika

Linki do przykładowych person stworzonych w programie Canva:

- Persona nr. 1 - Marek
- Persona nr. 2 - Katarzyna
- Persona nr. 3 - Marta

Link do przykładowych historyjek użytkowników stworzonych w programie Canva:

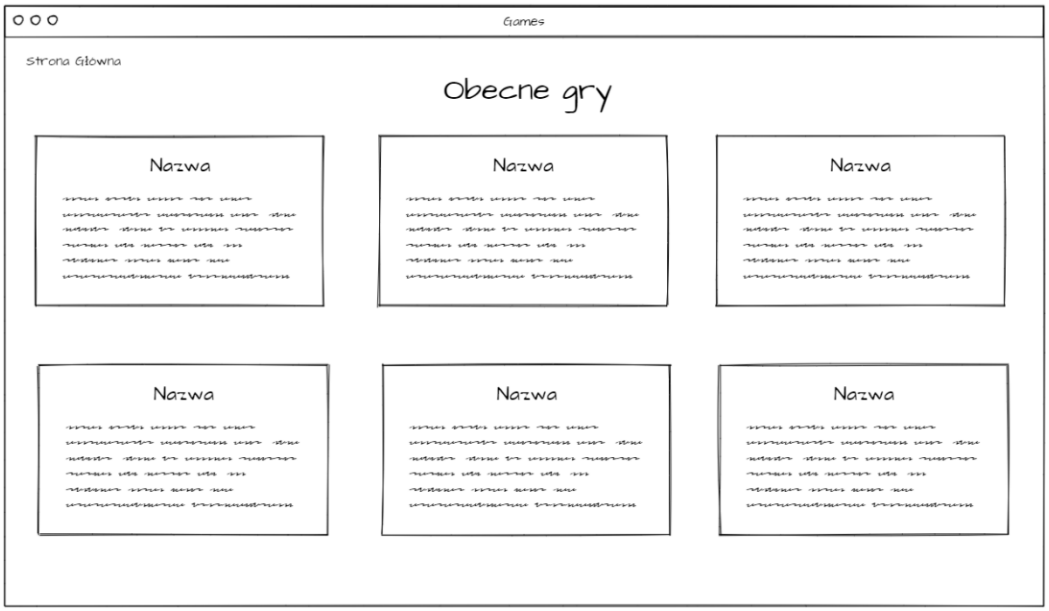
- Historyjki

Przykładowe widoki użytkownika stworzone w wireframepro: Link do wizualizacji

- Strona główna aplikacji – strona witająca użytkownika. Umożliwia przejście do ekranu logowania lub do listy obecnych gier.

Rysunek 12 - Strona główna aplikacji

- Lista obecnych gier – Umożliwia powrót do strony głównej lub przejście do wybranej gry.



Rysunek 13 - Lista obecnych gier

- Widok gry – Umożliwia przejście do panelu grania w grę lub powrót do strony głównej.

Rysunek 14- Widok gry

8 Specyfikacja testów

Pod względem błędów występujących w części backendu kod przechwytuje wszelkie błędy i daje odpowiednie odpowiedzi zawierające krótki opis oraz kod błędu odpowiadający błędom HTTP.

Dla części frontendu wszelkie błędy są wyświetlane, czy to z odpowiednim dopiskiem np. “Użytkownik już dołączył do tej gry.” przy ponownej próbie dołączenia do gry, do której już się dołączyło. W przypadku nieokreślonych błędów wyświetlany komunikat pokazuje kod i treść błędu. Co więcej w przypadku jakiegokolwiek błędu jest on logowany na konsolę.

Testy backendu przeprowadzone zostały przy pomocy biblioteki testowej zapewnione przez Django. Testowane jest tworzenie nowych rekordów w bazie danych z poprawnymi danymi oraz błędnymi będącymi polegającymi na próbie połączenia rekordów z innymi nieistniejącymi.

Testy frontendu przeprowadzane zostały przy pomocy różnych scenariuszy testowych, żeby sprawdzić, czy program działa zgodnie z założeniami lub czy wyświetlane są komunikaty jakie były przewidziane w przypadku określonych błędów.

Scenariusze testowe:

- Logowanie do nieistniejącego konta - Wybieramy “Zaloguj się”, wpisujemy nazwę użytkownika oraz hasło użytkownika, którego nie ma jeszcze w bazie danych. Klikamy “Zaloguj się”. Aplikacja wyświetli komunikat: “Nieprawidłowa nazwa użytkownika lub hasło. Spróbuj ponownie.”
- Rejestrowanie się do aplikacji - Wybieramy “Zarejestruj się”, wpisujemy nazwę użytkownika, którego nie ma w bazie danych, email oraz hasło. Klikamy “Zarejestruj się”. Aplikacja wyświetla komunikat: “Pomyślnie zarejestrowano.”
- Próba rejestracji z już istniejącą nazwą użytkownika - Wybieramy “Zarejestruj się”, wpisujemy nazwę użytkownika, która jest już w bazie danych, email oraz hasło. Klikamy “Zarejestruj się”. Aplikacja wyświetla komunikat: “Ta nazwa użytkownika jest już zajęta.”
- Logowanie do aplikacji - Wybieramy “Zaloguj się”, wpisujemy nazwę oraz hasło użytkownika, który już jest w bazie danych. Klikamy “Zaloguj się”. Aplikacja wyświetli komunikat: “Pomyślnie zalogowano”
- Pierwsze dołączanie do gry - Logujemy się na swoje konto. Wybieramy jedną ze stworzonych gier, w które wcześniej nie graliśmy i klikamy “Zagraj”. Wpisujemy liczbę osób w grupie grających w grę, a następnie klikamy “Dołącz do gry“. Aplikacja przekieruje nas do okna gry.
- Powrót do gry - Logujemy się na swoje kontro, wybieramy jedną ze stworzonych gier, do których dołączyliśmy wcześniej. Aplikacja przekieruje nas do okna gry.
- Granie w grę - Po dołączeniu do gry aplikacja wyświetla zadanie.
    - Zadanie z odpowiedzią tekstową - Wprowadzamy niepoprawną odpowiedź i klikamy “Zatwierdź odpowiedź”. Aplikacja wyświetli komunikat: “Niepoprawna odpowiedź” i nie pozwoli przejść dalej. Po wpisaniu poprawnej odpowiedzi i kliknięciu “Zatwierdź odpowiedź” aplikacja pozwoli przejść dalej.

    - Zadanie z odpowiedzią w formie obrazka – Klikamy w niepoprawny obraz, aplikacja pobierze nową listę obrazów, z czego jeden będzie poprawny, przy kilkukrotnym wyborze złej odpowiedzi aplikacja dobiera różne zestawy niepoprawnych odpowiedzi połączonych z jedną prawidłową odpowiedzią. Po wybraniu prawidłowego obrazka aplikacja pozwoli przejść do kolejnego zadania.
    - Zadanie z lokalizacją - Klikamy "Sprawdź lokalizację". Aplikacja wyświetli informację o współrzędnych geograficznych użytkownika. Po kliknięciu "Zatwierdź odpowiedź", jeśli użytkownik jest dostatecznie blisko docelowego miejsca, aplikacja pozwoli przejść do kolejnego zadania.
    - Po skończeniu się zadań aplikacja wyświetli komunikat z gratulacjami dla gracza.
- Dodanie scenariusza - Logujemy się jako administrator. Wybieramy "Scenariusze". Klikamy "+". Wpisujemy tytuł scenariusza, opis oraz wybieramy obraz. Klikamy "Dodaj scenariusz".
- Dodanie zadań różnego typu do scenariusza - Po dodaniu nowego scenariusza znajdujemy go na liście i wybieramy "Zadania". Wybieramy "+". Wpisujemy treść zadania, jako typ zadania wybieramy "tekst", wpisujemy poprawną odpowiedź oraz wybieramy obrazek pomocniczy. Klikamy "Dodaj zadanie". Aplikacja wyświetli dodane zadanie pod scenariuszem. Dodajemy kolejne zadanie, wybieramy jako typ odpowiedzi "Obraz". Wybieramy listę obrazów które posłużą jako odpowiedzi poprawne oraz listę obrazów, które posłużą jako odpowiedzi niepoprawne. Dodajemy kolejne zadanie, wybieramy typ odpowiedzi jako "lokalizacja". Wpisujemy po przecinku szerokość i wysokość geograficzną miejsca, do którego ma udać się gracz. Aplikacja pod scenariuszem wyświetli listę zadań.
- Edycja zadań - Wybieramy "Zadania" w danym scenariuszu. Klikamy "Edytuj" przy zadaniu, które chcemy zmienić. Wpisujemy nowe wartości zadania i klikamy "Edytuj zadanie". Aplikacja wyświetli listę zadań ze zmodyfikowanym zadaniem.
- Aktywacja gry – Przechodzimy do scenariuszy. Klikamy "Aktywuj grę" przy grze, którą chcemy uruchomić. Ustawiamy jej tytuł, czas rozpoczęcia oraz zakończenia. Klikamy "Zaplanuj grę". Po przejściu do zakładki "Gry" aplikacja wyświetli nam nowo utworzoną grę.

Testy części backendu zostały również przebadane przy pomocy narzędzia "coverage".

W celu uruchomienia narzędzia "coverage" należy:

- Wejść do kontenera odpowiadającego za backend: **docker exec -it pzsp2-gra-miejska-backend-1 sh** (**sh** jest dla Windowsa w przypadku Linuxa należy zamienić **sh** na **bash**)
- W kontenerze: **coverage run manage.py test**
- W celu wyświetlenia wyniku należy wpisać: **coverage report**
- W celu wygenerowania podgladu w html należy wpisać: **coverage html**
- W celu uruchomienia podglądu html należy uruchomić plik **index.html** znajdujący się w folderze **htmlcov** w backendzie.

Name	Stmts	Miss	Cover
backend/__init__.py	0	0	100%
backend/settings.py	37	0	100%
backend/urls.py	7	1	86%
manage.py	11	2	82%
tasks/__init__.py	0	0	100%
tasks/admin.py	1	0	100%
tasks/apps.py	4	0	100%
tasks/migrations/0001_initial.py	5	0	100%
tasks/migrations/0002_alter_task_number.py	4	0	100%
tasks/migrations/0003_scenario.py	4	0	100%
tasks/migrations/0004_task_scenario.py	5	0	100%
tasks/migrations/0005_alter_task_scenario.py	5	0	100%
tasks/migrations/0006_remove_task_scenario.py	4	0	100%
tasks/migrations/0007_task_scenario.py	5	0	100%
tasks/migrations/0008_game.py	5	0	100%
tasks/migrations/0009_rename_beggining_date_game_beginning_date.py	4	0	100%
tasks/migrations/0010_task_correct_answer.py	4	0	100%
tasks/migrations/0011_task_answer_type_alter_task_correct_answer.py	4	0	100%
tasks/migrations/0012_user.py	6	0	100%
tasks/migrations/0013_rename_correct_answer_task_correct_text_answer.py	4	0	100%
tasks/migrations/0014_correctimages_incorrectimages.py	5	0	100%
tasks/migrations/0015_remove_incorrectimages_task_answerimages_and_more.py	5	0	100%
tasks/migrations/0016_team_completedtask.py	5	0	100%
tasks/migrations/0017_remove_completedtask_team_completedtask_user.py	5	0	100%
tasks/migrations/0018_alter_completedtask_task_alter_completedtask_user.py	5	0	100%
tasks/migrations/0019_remove_completedtask_user_completedtask_team.py	5	0	100%
tasks/migrations/__init__.py	0	0	100%
tasks/models.py	39	0	100%
tasks/serializers.py	42	0	100%
tasks/tests.py	666	0	100%
tasks/urls.py	13	0	100%
tasks/views.py	256	24	91%
TOTAL	1165	27	98%

Rysunek 15- Wynik narzędzia ”coverage”

Opis realizacji testów:

1. Sprawdzanie poprawności danych wejściowych:
    - Brak wymaganych pól: Testy, które weryfikują, czy odpowiednie pola zostały uwzględnione w żądaniu (np. brak tytułu w tworzeniu scenariusza: test_create_scenario_missing_fields).
    - Nieprawidłowe wartości: Testy, które sprawdzają, czy aplikacja prawidłowo reaguje na dane wejściowe w nieprawidłowym formacie (np. niepoprawny typ odpowiedzi, jak w test_create_task_invalid_number).
2. Autoryzacja i uwierzytelnianie:
    - Testy dotyczące dostępu do zasobów w zależności od statusu uwierzytelnienia użytkownika (np. test_get_tasks_unauthenticated, test_add_task_to_scenario_unauthenticated). Tego rodzaju testy sprawdzają, czy

system odrzuca żądania, gdy użytkownik nie jest zalogowany lub nie ma odpowiednich uprawnień.
    - Błąd autoryzacji: Użycie force_authenticate w testach jednostkowych pozwala na symulowanie uwierzytelnienia użytkownika bez konieczności korzystania z rzeczywistego logowania.
3. Przetwarzanie wyjątków:
    - Testy weryfikujące, jak aplikacja reaguje na błędy, takie jak nieistniejące zasoby, np. test_create_game_invalid_scenario sprawdzający, czy aplikacja odrzuci próbę stworzenia gry z nieistniejącym scenariuszem (status HTTP 400 lub 404).
    - Obsługa nieistniejących zasobów: W przypadku nieistniejącego zasobu, system powinien odpowiedzieć z odpowiednim kodem błędu, np. HTTP_404_NOT_FOUND.
4. Obsługa plików i formatów odpowiedzi:
    - Testy związane z przesyłaniem plików, np. test_create_task_with_files, sprawdzają, czy aplikacja poprawnie obsługuje załączniki (np. obrazy, audio), a odpowiedź zawiera poprawne statusy HTTP.
    - Weryfikacja odpowiedzi JSON: Testy, które sprawdzają odpowiedzi w formacie JSON, czy zawierają one wymagane pola i wartości, np. test_send_email.
5. Sprawdzanie stanu bazy danych:
    - Testy, które sprawdzają, czy operacje na bazie danych (np. tworzenie lub usuwanie obiektów) zostały prawidłowo zrealizowane, takie jak test_create_scenario czy test_delete_task. Testy te upewniają się, że po wykonaniu operacji w bazie danych liczba rekordów jest zgodna z oczekiwaniami.
6. Zmiany w bazie danych i ich wpływ:
    - Testy takie jak test_shift_task_numbers oraz test_perform_update_new_number_greater weryfikują, jak aplikacja zachowuje się w przypadku zmiany danych (np. numerów zadań) i czy odpowiednia liczba rekordów zostaje zaktualizowana w bazie danych.
7. Kody odpowiedzi HTTP:
    - W testach używane są odpowiednie kody odpowiedzi HTTP (np. HTTP_201_CREATED, HTTP_400_BAD_REQUEST, HTTP_404_NOT_FOUND), aby upewnić się, że odpowiedź jest odpowiednia do sytuacji.
8. Zabezpieczenia i walidacja danych:
    - Walidacja wprowadzanych danych jest kluczowa w takich testach, jak test_perform_create_without_scenario oraz test_perform_create_with_invalid_number, gdzie weryfikowana jest obecność wymaganych pól oraz ich poprawność.

Funkcja porównująca odpowiedzi została dodatkowo przetestowana ręcznie w celu doboru parametru potwierdzającego poprawność odpowiedzi.

```
Dom poloni = Dom poloni   :   100.0
Dom poloni = Dom polONi   :   100.0
Dom poloni = Dom poloi    :   94.11764705882352
Dom poloni = DOM POLONI   :   100.0
Dom poloni = poloni dom   :   100.0
Dom poloni = Dom poloni starszej   :   100.0
a = b   :   0.0
czerwona szkoła = zielona szkoła   :   81.4814814814815
czerwona szkoła = szkoła zielona   :   81.4814814814815
szkoła = czerwona szkoła   :   100.0
czerwona szkoła = czkoła szerwona   :   85.71428571428572
czerwona szkoła = czerwona szkoła w Warszawie   :   100.0
```

Rysunek 16- Widok pierwszego zadania w danej grze (typ tekstowy)

9 *Wirtualizacja/konteneryzacja*

Projekt realizowany jest z wykorzystaniem kontenerów w technologii Docker. Struktura aplikacji opiera się na dwóch głównych kontenerach: jeden dedykowany jest backendowi, a drugi frontendowi. Kontener backendowy odpowiada za pobieranie wymaganych bibliotek oraz uruchamianie serwera aplikacji. Pliki źródłowe oraz wszystkie niezbędne zależności kopiowane są bezpośrednio z repozytorium projektu.

Podobnie działa kontener frontendowy, który instaluje biblioteki wymagane do działania aplikacji napisanej w React, a następnie uruchamia serwer obsługujący tę część systemu. Również tutaj pliki i zależności są pobierane z repozytorium, co zapewnia spójność kodu w obu środowiskach.

Całość projektu jest zarządzana za pomocą narzędzia Docker Compose, które umożliwia określenie portów, na których działają poszczególne kontenery, a także definiuje konfigurację bazy danych oraz jej integrację z backendem. Docker Compose obsługuje również zależności pomiędzy kontenerami, zapewniając ich poprawne działanie w skoordynowany sposób.

10 Bezpieczeństwo

Aplikacja wykorzystuje Django, które zapewnia kilka warstw bezpieczeństwa. Dzięki systemowi ORM, Django automatycznie zapobiega atakom typu SQL Injection, co chroni dane w bazie przed złośliwymi zapytaniami. Hasła użytkowników są przechowywane w bezpiecznej formie (jako hasze), co uniemożliwia ich odczyt w razie wycieku danych. Django wspiera również mechanizmy ochrony przed popularnymi atakami, takimi jak XSS, CSRF czy Clickjacking, automatycznie zabezpieczając formularze i strony.

Dodatkowo, aplikacja korzysta z Dockerów, co izoluje kontenery z backendem i frontendem, zwiększając bezpieczeństwo. Dane, takie jak hasła i klucze API, są przechowywane w zmiennych środowiskowych, co minimalizuje ryzyko ich ujawnienia.

Uwierzytelnianie jest zrealizowane przy pomocy biblioteki djoser z wykorzystaniem tokenów JWT.

11 Podręcznik użytkownika

Rejestracja:

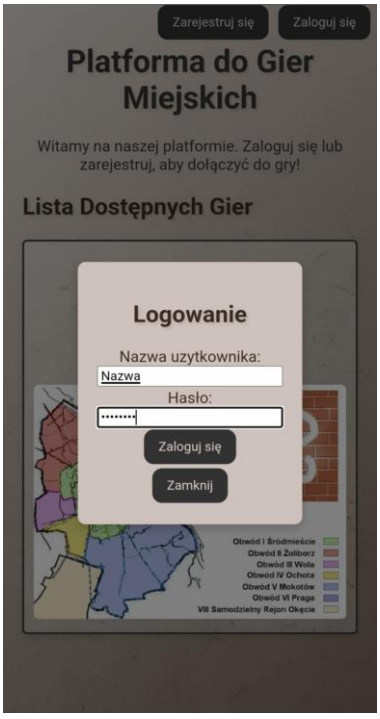
Rysunek 17- Widok strony głównej

Rysunek 18- Widok formularza rejestracyjnego

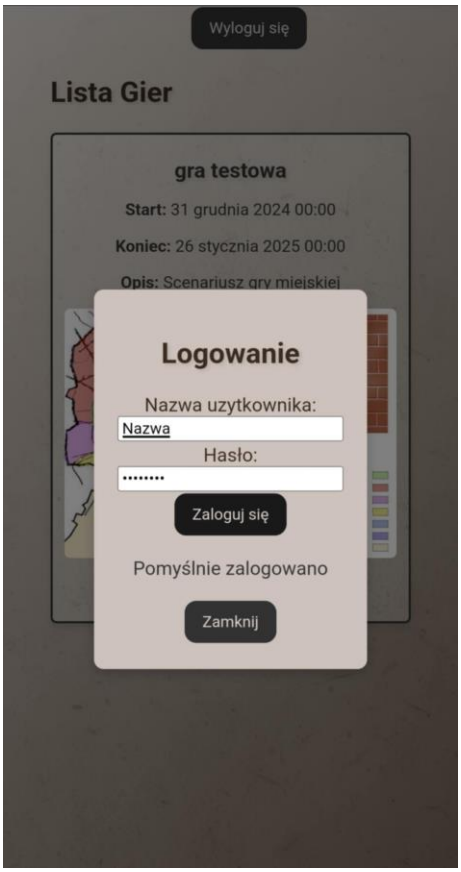
Rysunek 19- Widok formularza rejestracyjnego z informacją o statusie rejestracji

Logowanie:

Rysunek 20- Widok strony głównej



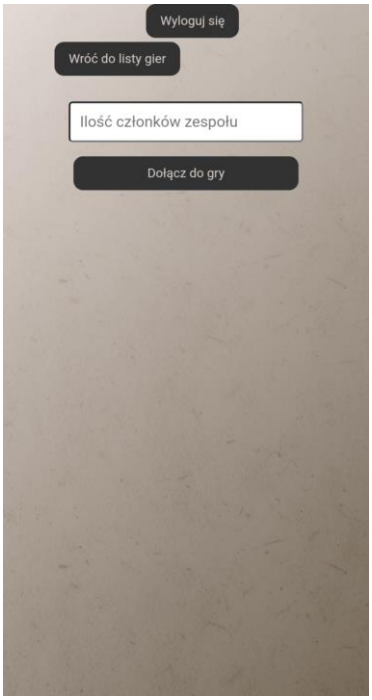
Rysunek 21- Widok formularza do logowania



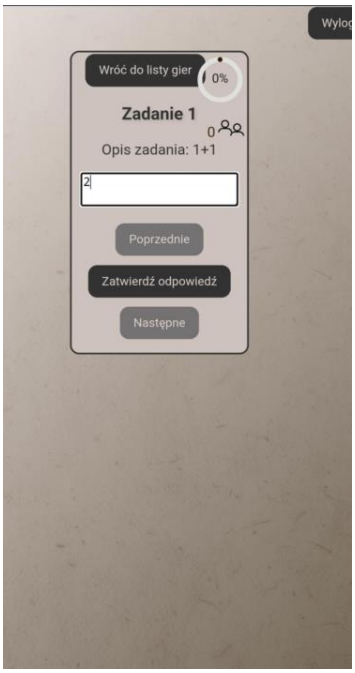
Rysunek 22- Widok formularza rejestracyjnego z informacją o statusie logowania

Dołączenie do gry:

Rysunek 23- Widok strony po zalogowaniu



Rysunek 24- Widok formularza dołączania do gry



Rysunek 25- Widok pierwszego zadania w danej grze

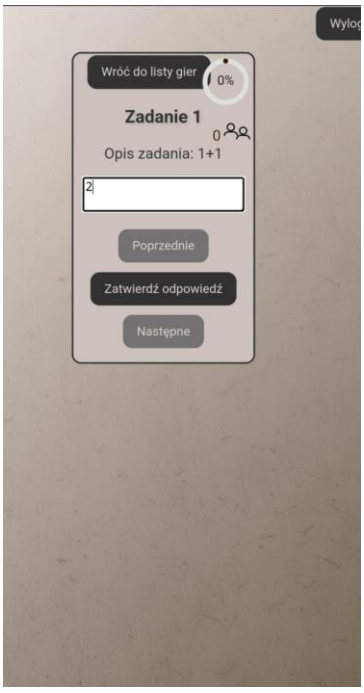
Powrót do wcześniej dołączonej gry:

Rysunek 26- Widok strony po zalogowaniu

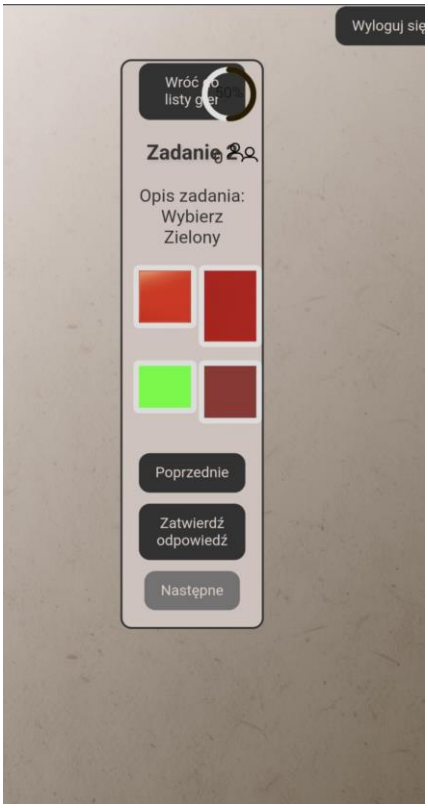
Rysunek 27- Widok pierwszego zadania w danej grze

Przechodzenie zadań:

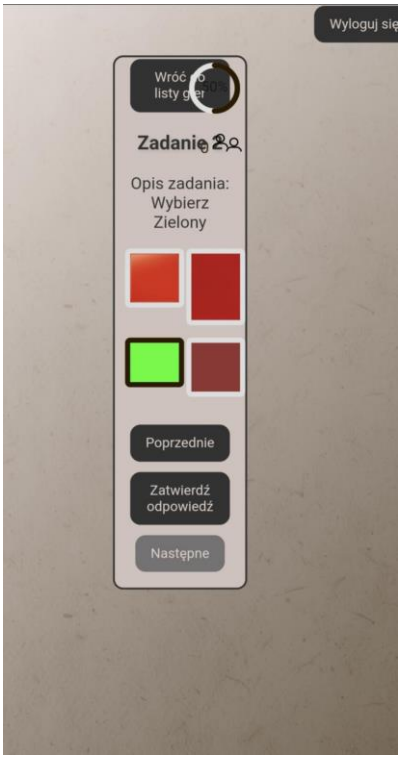
Rysunek 28- Widok pierwszego zadania w danej grze (typ tekstowy)



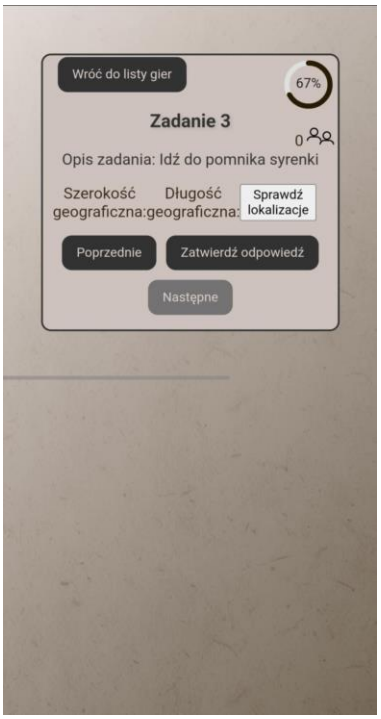
Rysunek 29- Widok wypełnionego pierwszego zadania w danej grze (typ tekstowy)



Rysunek 30- Widok drugiego zadania w danej grze (typ wyboru obrazka)



Rysunek 31- Widok wypełnionego drugiego zadania w danej grze (typ wyboru obrazka)



Rysunek 32- Widok trzeciego zadania w danej grze (typ lokalizacja)

12 Podręcznik administratora

12.1. Instrukcja budowy systemu:

Do uruchomienia aplikacji niezbędne jest oprogramowanie Docker. Po sklonowaniu repozytorium należy skopiować pliki:

- *.env.db.sample* i nadać mu nazwę *.env.db.dev*
- *.env.sample* i nadać mu nazwę *.env.dev*

W pliku *.env.dev* należy podmienić w dwóch miejscach <API_IP> na adres ip urządzenia, na którym uruchamiamy aplikację oraz ustawić swoje dane dla aplikacji, takie jak nazwa użytkownika i hasło dla bazy danych, czy klucz prywatny django. Po skonfigurowaniu aplikacji należy uruchomić terminal w głównym folderze. W terminalu uruchamiany komendę budującą kontenery:

docker compose build

Po zbudowaniu kontenerów uruchamiamy je:

docker compose up

Następnie należy uruchomić drugi terminal, również w głównym folderze. W terminalu wpisujemy komendę wykonującą migrację:

docker-compose exec backend python manage.py migrate

Po wykonaniu migracji aplikacja będzie dostępna pod adresem

http://<machine_ip_address>:5173

Aby utworzyć konto administratora należy w tym samym terminalu wywołać komendę:

docker compose exec backend python manage.py createsuperuser

A następnie wprowadzić pożądane dane.

13 Podsumowanie

[Krytyczna analiza osiągniętych wyników, mocne i słabe strony

Możliwe kierunki rozwoju]

Na obecny moment projekt nie posiada wszystkich funkcjonalności, które były zaplanowane. Nie da się wygenerować raportu z danej gry. Na obecną chwilę ponowna rozgrywka odbywa się przez wejście do gry z listy wszystkich dostępnych gier i nie da się zobaczyć tylko tych, w których bierze się udział.

Sama aplikacja bez istniejących gier wydaje się pusta i brakuje opisu organizacji czy celu na stronie głównej.

Aplikacja zapewnia dobre skalowanie interfejsu do telefonu oraz spójny wygląd wszystkich elementów na stronie. Obsługa platformy jest intuicyjna i pozwala graczowi skupić się na rozgrywce.

Co do kwestii wizualnych, to dla telefonu w zadaniach wskaźnik procentowy przysłania nazwę zadania.

Ostatecznie aplikacja realizuje praktycznie wszystko co było założone i na obecną chwilę da się z niej korzystać zgodnie z jej przeznaczeniem. Co więcej uważamy, że wygląd jest na tyle prosty, że da się po niej płynnie przemieszczać bez wcześniejszej znajomości.

14 Bibliografia

Django Documentation: *The web framework for perfectionists with deadlines*. [Online]. Dostępne na: https://docs.djangoproject.com/.

React Documentation: *A JavaScript library for building user interfaces*. [Online]. Dostępne na: https://react.dev/.

Zatwierdzam dokumentację.	…… . Data i podpis Mentora