

# Dokumentacja projektu: SzachUZ

---

## Spis treści:

1.	Specyfikacja wymagań.....	2
2.	Plan projektu.....	3
3.	Architektura systemu.....	5
4.	Protokoły spotkań i decyzje.....	7
5.	Projekty UI.....	11
6.	Dokumentacja Kodu.....	17

## **1. Specyfikacja projektu**

**Projekt SzachUZ** to kompleksowa, internetowa platforma służąca do gry w szachy w środowisku online. System został zaprojektowany tak, aby umożliwiać użytkownikom swobodną i intuicyjną rozgrywkę w dwóch głównych trybach: **PvE** (gracz kontra komputer) oraz **PvP** (gracz kontra gracz). Głównym założeniem projektu jest stworzenie w pełni funkcjonalnej aplikacji webowej, dzięki której gracze będą mogli rywalizować zarówno z algorytmem szachowym, jak i z innymi użytkownikami w czasie rzeczywistym, niezależnie od miejsca, w którym się znajdują.

Platforma przewiduje pełną obsługę kont użytkowników, obejmującą **rejestrację, logowanie oraz zarządzanie profilem**, co pozwoli na personalizację doświadczenia oraz zachowanie historii rozgrywek. Aplikacja będzie wspierać **wiele wersji językowych**, aby umożliwić korzystanie z niej szerszemu gronu odbiorców, a także zaoferuje różne **tryby wizualne**, takie jak motyw jasny i ciemny, zwiększające komfort korzystania w zależności od preferencji użytkownika i warunków oświetleniowych.

Kluczowym elementem projektu jest integracja z zaawansowanym **algorytmem szachowym**, zapewniającym możliwość gry przeciwko komputerowi w trybie PvE. Jednocześnie dostępny będzie tryb **PvP**, umożliwiający rywalizację bezpośrednio pomiędzy graczami — zarówno w formie zapraszania konkretnych znajomych do partii, jak i otwartych meczów z losowo dobranymi przeciwnikami.

Platforma wprowadza również rozbudowany **system rankingu**, oparty na punktach przyznawanych za wygrane i przegrane rozgrywki. Ranking umożliwi użytkownikom śledzenie własnych postępów, porównywanie wyników z innymi graczami oraz motywację do dalszego doskonalenia swoich umiejętności szachowych. Dodatkową funkcjonalnością będą **powiadomienia e-mail**, pozwalające użytkownikom otrzymywać informacje o zaplanowanych meczach, zaproszeniach do gry oraz wynikach zakończonych rozgrywek.

Po każdej partii użytkownik będzie miał możliwość wygenerowania **raportu w formacie PDF**, zawierającego szczegółowy zapis wykonanych ruchów, finalny wynik, czas trwania partii oraz dodatkowe informacje podsumowujące przebieg rozgrywki. Funkcja ta może być szczególnie przydatna dla osób analizujących swoje gry lub wykorzystujących platformę do nauki i treningu.

Projekt SzachUZ jest realizowany przez **zespoł studentów**, którego celem jest stworzenie nowoczesnej, stabilnej i w pełni funkcjonalnej platformy do gry w szachy online. Ostatecznym celem projektu jest dostarczenie użytkownikom narzędzia, które będzie nie tylko przyjazne w obsłudze, lecz także rozbudowane pod względem możliwości, wspierając zarówno rozgrywkę rekreacyjną, jak i bardziej zaawansowaną analizę partii.

## **2. Plan projektu**

Poniżej przedstawiono podział prac projektowych na sprinty wraz z zakresem funkcjonalnym realizowanym przez zespół. Każdy sprint obejmuje również bieżące uzupełnianie **dokumentacji projektu**.

---

### **Sprint 1: Podstawy projektu i fundamenty aplikacji**

**Okres:** 7 listopada – 20 listopada

**Cel główny:** przygotowanie warstwy front-endowej, konfiguracja środowiska oraz implementacja podstawowych funkcjonalności użytkownika.

#### **Zakres prac:**

- **Projekt graficzny aplikacji** – stworzenie layoutu, stylów oraz podstawowej identyfikacji wizualnej.
- **Logowanie i system użytkowników** – obsługa tworzenia kont, logowania i podstawowej autoryzacji.
- **Konfiguracja CI/CD** – przygotowanie pipeline'ów do automatycznego budowania i wdrażania aplikacji.
- **Obsługa wersji językowych** – implementacja języka polskiego i angielskiego.
- **Ciasteczka (cookies)** – obsługa zapisów preferencji użytkownika, sesji oraz danych technicznych.
- **Profil użytkownika** – podstawowy widok i możliwość modyfikacji danych użytkownika.
- **Motyw jasny/ciemny** – wprowadzenie przełącznika motywu oraz pełna obsługa obu wariantów.
- **Front-end (podstawowa warstwa UI)** – przygotowanie kluczowych widoków aplikacji (zakres zależny od osoby odpowiedzialnej).
- **Komunikacja serwer–baza danych** – przygotowanie backendu do wymiany danych z bazą.

## Sprint 2: Implementacja logiki szachowej – tryb solo (PvE)

**Okres:** 20 listopada – 25 grudnia

**Cel główny:** opracowanie i integracja logiki gry jednoosobowej, opartej na algorytmach szachowych lub sztucznej inteligencji.

**Zakres prac:**

- **Analiza algorytmów i SI do szachów** – przegląd dostępnych rozwiązań, ich możliwości i wymagań.
  - **Interakcja gracza z planszą szachową** – implementacja wykonywania ruchów, validacji i mechaniki gry.
  - **API do algorytmu szachowego / silnika SI** – stworzenie modułu komunikującego aplikację z AI.
  - **Poziomy trudności** – dodanie co najmniej kilku wariantów inteligencji przeciwnika.
- 

## Sprint 3: Tryb wieloosobowy (PvP) i funkcje społecznościowe

**Okres:** 7 stycznia – 22 stycznia

**Cel główny:** uruchomienie rozgrywek gracz vs gracz oraz dodanie funkcji wspierających rywalizację.

**Zakres prac:**

- **Ranking (top 10–20)** – wyświetlanie najlepszych graczy i mechanizm aktualizacji punktów.
- **Statystyki / przebieg partii w PDF** – generowanie raportu zawierającego zapis ruchów, wynik oraz czas gry.
- **Powiadomienia e-mail o zaplanowanych grach** – automatyczne wysyłanie informacji o nadchodzących meczach.
- **Matchmaking / wyszukiwanie przeciwnika** – dobieranie rywali w czasie rzeczywistym.
- **Tryb PvP** – pełna obsługa gry między użytkownikami.
- **Integracja z modelem algorytmu (opcjonalnie)** – jeśli będzie wymagane wsparcie w logice gry lub weryfikacji ruchów.

### **3. Architektura systemu**

Architektura projektowanego systemu została zaprojektowana w sposób modułowy i zgodny ze współczesnymi standardami tworzenia aplikacji webowych w języku Java. Rozwiązanie opiera się na technologiach **Jakarta EE 10**, co umożliwia łatwą integrację poszczególnych warstw systemu oraz zapewnia skalowalność i wysoką wydajność podczas przetwarzania żądań użytkowników.

System realizuje model architektoniczny **warstwowy**, obejmujący trzy podstawowe warstwy: warstwę prezentacji, warstwę logiki biznesowej oraz warstwę dostępu do danych.

---

#### **Warstwa prezentacji**

Warstwa frontendowa jest realizowana przy użyciu **Java Servlet Pages (JSP)**. JSP umożliwia generowanie dynamicznych treści HTML na podstawie danych dostarczanych z serwera aplikacji. Komunikacja między użytkownikiem a aplikacją odbywa się za pośrednictwem przeglądarki internetowej, która wysyła żądania HTTP do serwera.

W tej warstwie znajdują się również mechanizmy walidacji danych po stronie użytkownika oraz elementy zapewniające ergonomiczny interfejs użytkownika. JSP działa ściśle z servletami, które odpowiadają za obsługę żądań, sterowanie przepływem i komunikację z warstwą logiki biznesowej.

---

#### **Warstwa logiki biznesowej i komunikacja REST**

W celu realizacji logiki aplikacji oraz udostępniania funkcji systemu zewnętrznym modułom zastosowano **JAX-RS** — standard Jakarta EE służący do budowania usług REST. Endpointy REST są podstawowym sposobem wymiany danych między frontendem a backendem, jak i potencjalnymi klientami zewnętrznymi (np. aplikacjami mobilnymi).

Każdy endpoint odpowiada za obsługę konkretnej części funkcjonalności systemu, m.in. rejestrację użytkowników, operacje CRUD na danych domenowych czy współpracę z modułem autoryzacji.

W zakresie bezpieczeństwa zastosowano rozwiązania oparte na **Java MicroProfile** z wykorzystaniem **JWT (JSON Web Token)**. Tokeny zapewniają bezpieczne potwierdzanie tożsamości użytkownika po zalogowaniu oraz umożliwiają kontrolę dostępu do chronionych zasobów systemu. MicroProfile oferuje gotowy zestaw

rozszerzeń wspierających konfigurację, obsługę zabezpieczeń i komunikację w architekturach mikroserwisowych.

---

## Warstwa danych

Za przechowywanie danych odpowiada relacyjna baza danych **PostgreSQL**. Baza została dobrana ze względu na wysoką zgodność ze standardami SQL, dużą wydajność oraz rozbudowane możliwości zarządzania danymi. Aplikacja komunikuje się z bazą poprzez komponenty Jakarta EE odpowiedzialne za trwałość danych, zapewniając spójność informacji oraz obsługę transakcyjną.

Model danych został opracowany tak, aby możliwa była łatwa rozbudowa systemu i integracja z dodatkowymi modułami w przyszłości.

---

## Konteneryzacja i środowisko uruchomieniowe

Cały system został zintegrowany z platformą **Docker**, co umożliwia uruchamianie aplikacji i bazy danych w izolowanych kontenerach. Dzięki temu środowisko uruchomieniowe jest powtarzalne, łatwe w konfiguracji i niezależne od lokalnych ustawień programistów.

Architektura kontenerowa poprawia elastyczność wdrożeniową — system może być uruchamiany zarówno lokalnie, jak i na serwerach w chmurze. Oddzielenie kontenerów aplikacji oraz bazy danych zapewnia też lepszą kontrolę nad zarządzaniem zasobami.

## **4. Protoły spotkań i decyzje**

Realizacja projektu odbywała się w trybie zdalnym z wykorzystaniem platformy GitHub jako głównego narzędzia koordynacji prac zespołu. Z uwagi na charakter projektu studenckiego i ograniczoną dostępność czasową członków zespołu, spotkania nie odbywały się w formie regularnych sesji scrumowych, lecz w sposób asynchroniczny poprzez konsultacje online, komunikatory oraz komentarze w repozytorium.

Każdy z członków zespołu pracował nad wyznaczonymi zadaniami w dogodnym dla siebie czasie, co odzwierciedlane jest w historii zmian i aktywności GitHub Actions. Poniżej zestawiono kluczowe daty, prace i decyzje podjęte w trakcie rozwoju projektu.

---

### **6 listopada 2025 — Konfiguracja środowiska i automatyzacja**

Na tym etapie rozpoczęto przygotowania środowiska developerskiego. Uruchomiono pierwsze **GitHub Actions** odpowiedzialne za automatyczne testy oraz weryfikację poprawności komplikacji projektu. Decyzja: dalsza automatyzacja budowania systemu będzie rozwijana wraz z projektem.

---

### **18 listopada 2025 — Usprawnienia uruchamiania aplikacji**

Wprowadzono poprawki w plikach Docker Compose oraz dodano skrypty ułatwiające uruchamianie aplikacji lokalnie. Zaktualizowano plik README o instrukcje dla użytkowników korzystających z PowerShell. Decyzja: docelowe środowisko uruchamiane będzie w **kontenerach Dockera** w celu zapewnienia spójności konfiguracji między członkami zespołu.

W tym dniu również scalono pierwsze większe zmiany funkcjonalne poprzez **Merge Pull Request #4**.

---

### **21 listopada 2025 — Uporządkowanie repozytorium**

Wprowadzono zmiany porządkujące repozytorium, m.in. scalono gałąź **origin/master** oraz dodano linki do niezbędnych zasobów projektowych. Ustalono również podział prac dotyczących front-endu i backendu.

## **27 listopada 2025 — Dodanie kluczowych funkcjonalności**

Był to jeden z najbardziej intensywnych dni projektowych — scalono wiele gałęzi funkcjonalnych:

- **Merge PR #5** – pierwsza wersja interfejsu użytkownika (front-end)
- **Merge PR #6** – funkcjonalność profilu użytkownika
- **Merge PR #7 i #8** – obsługa logowania oraz stylizacja widoków

Decyzja: komponenty autoryzacji i interfejsu będą rozbudowywane iteracyjnie zgodnie z dalszym rozwojem projektu.

---

## **29 listopada 2025 — Dokumentacja techniczna**

Dodano pełen zestaw **diagramów UML**, obejmujących model danych oraz komponenty aplikacji. Decyzja: diagramy będą aktualizowane równolegle z implementacją, aby zapewnić ich zgodność ze stanem faktycznym aplikacji.

---

## **3 grudnia 2025 — Bezpieczeństwo i doświadczenie użytkownika**

Wprowadzono obsługę ciasteczek wykorzystywanych m.in. w procesie autoryzacji. Jest to krok w stronę pełnego wdrożenia **JWT MicroProfile** oraz poprawy UX użytkowników podczas logowania.

---

## **4 grudnia 2025 — Konfiguracja CI/CD**

Rozpoczęto prace nad konfiguracją potoku CI/CD z wykorzystaniem GitHub Actions oraz automatycznych wdrożeń na środowisko AWS. W tym okresie wystąpiły pierwsze problemy związane z poprawną konfiguracją workflowów oraz integracją gałęzi roboczych z gałęzią główną (**master**). Scalono Pull Request #13 dotyczący wstępnej konfiguracji mechanizmu CI/CD, który stał się bazą do dalszych usprawnień.

## **5 grudnia 2025 — Poprawki automatyzacji i rozwój profilu użytkownika**

Wprowadzono poprawki do konfiguracji GitHub Actions (Pull Requesty #15 oraz #16), co poprawiło stabilność procesu budowania i wdrażania aplikacji. Równolegle prowadzono prace nad funkcjonalnością profilu użytkownika ([userProfile](#)). Część wdrożeń zakończyła się niepowodzeniem z powodu problemów integracyjnych oraz błędów związanych z przekazywaniem tokenów autoryzacyjnych do serwera. W procesie weryfikacji zmian wykorzystano narzędzie Copilot Code Review.

---

## **8 grudnia 2025 — Rozszerzenie interfejsu użytkownika**

Scalono Pull Request #18, w ramach którego dodano nowy motyw graficzny interfejsu użytkownika (jasny oraz ciemny). Zmiany miały na celu poprawę czytelności oraz komfortu korzystania z aplikacji.

---

## **11 grudnia 2025 — Integracja zmian funkcjonalnych**

Scalono Pull Request #20, zawierający integrację wcześniej rozwijanych funkcjonalności aplikacji. Wdrożenie przebiegło poprawnie, co potwierdziło stabilność skonfigurowanego procesu CI/CD.

---

## **15 grudnia 2025 — Zabezpieczenie formularzy aplikacji**

Dodano mechanizm reCAPTCHA (Pull Request #21), który zwiększył bezpieczeństwo formularzy aplikacji oraz ograniczył możliwość nadużyć ze strony automatycznych botów.

---

## **16 grudnia 2025 — Poprawki konfiguracji Dockera**

Wprowadzono poprawki w konfiguracji środowiska kontenerowego Dockera (Pull Request #22), dotyczące obsługi kluczy oraz zmiennych środowiskowych. Zmiany wyeliminowały problemy występujące podczas wdrożeń aplikacji.

## **13 stycznia 2026 — Scalanie zmian końcowych**

Scalono Pull Request #30, obejmujący końcowe poprawki oraz porządkowanie kodu aplikacji. Wdrożenie zakończyło się powodzeniem, co potwierdziło stabilność projektu na etapie końcowym.

## 5. Projekty UI

SzachUZ

Strona Główna

Zaloguj się

motyw

Ranking najlepszych graczy

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.
- 10.

Rywalizuj ze znajomymi  
lub  
Trenuj samemu



Zostań królem SzachUZ

Zagraj teraz

SzachUZ

Panel główny (po zalogowaniu)

motyw

Trenuj z botem

Znajdź godnego przeciwnika

Rzuć wyzwanie znajomemu

Znajomi

Znajomy1

Znajomy2

Znajomy3

Znajomy4

Dodaj znajomego



Ekran Gry



Figury zbite Białe

Szachownica

Figury zbite Czarne

Historia ruchów



Profil użytkownika



Avatar

Ustawienia konta:

Zmień nick

Zmień awatar

Zmień hasło

Usuń konto

Punkty ligowe:

Nick gracza



## LOGOWANIE

e-mail:

hasło:

[Nie pamiętasz hasła?](#)

recaptcha

[Zaloguj się](#) [Zarejestruj się](#)

## Rejestracja

nazwa użytkownika:

e-mail:

hasło:

powtórz hasło:

[Masz już konto? Zaloguj się!](#)

recaptcha

[Zarejestruj się](#)

 SzachUZ

Strona Główna

Zaloguj się

motyw 

Ranking najlepszych graczy

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.
- 10.

Rywalizuj ze znajomymi  
lub  
Trenuj samemu



Zostań królem SzachUZ

Zagraj teraz

 SzachUZ

Panel główny (po zalogowaniu)

motyw 

Trenuj z botem

Znajdź godnego przeciwnika

Rzuć wyzwanie znajomemu

Znajomi

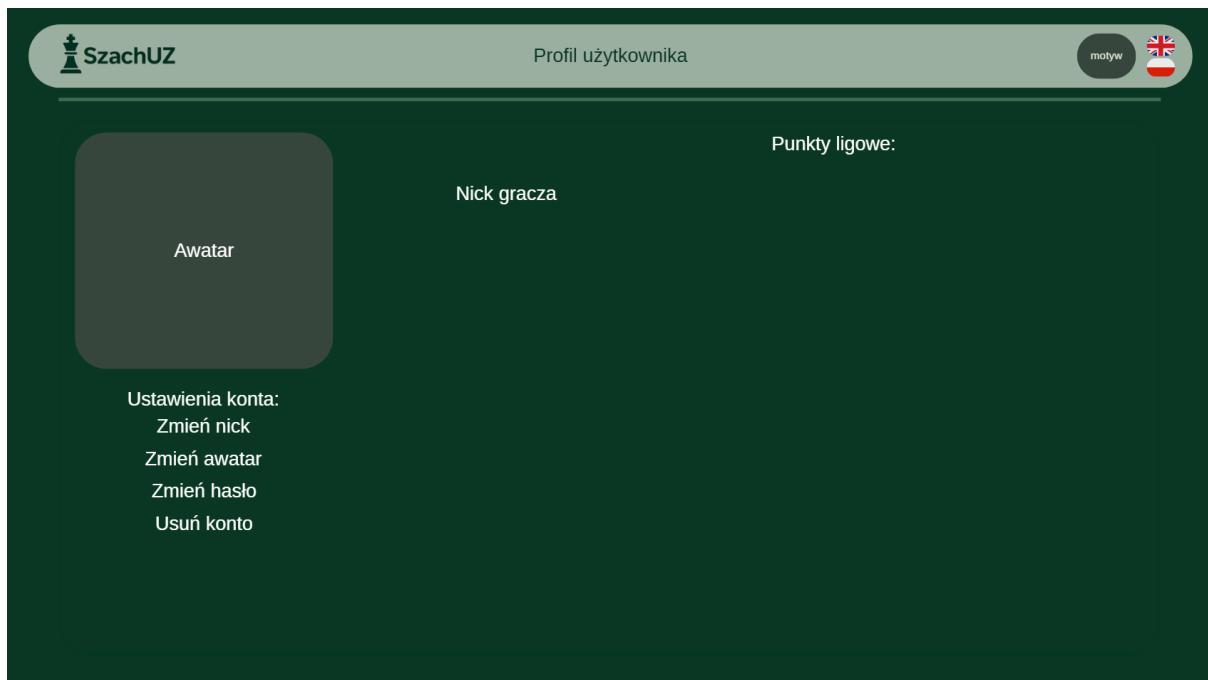
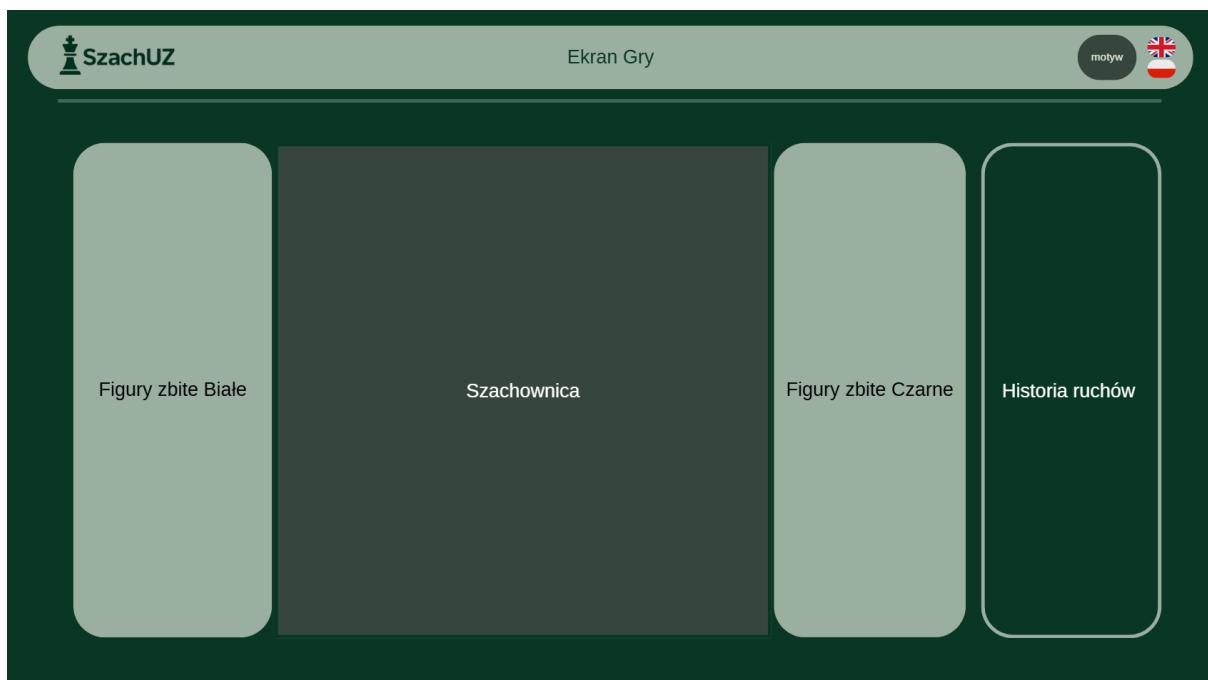
Znajomy1

Znajomy2

Znajomy3

Znajomy4

Dodaj znajomego



 SzachUZ

Logowanie



LOGOWANIE

e-mail:

hasło:

[Nie pamiętasz hasła?](#)

[Zaloguj się](#) [Zarejestruj się](#)

 SzachUZ

Rejestracja



Rejestracja

nazwa użytkownika:

e-mail:

hasło:

powtórz hasło:

[Masz już konto? Zaloguj się!](#)

## 6. Dokumentacja kodu

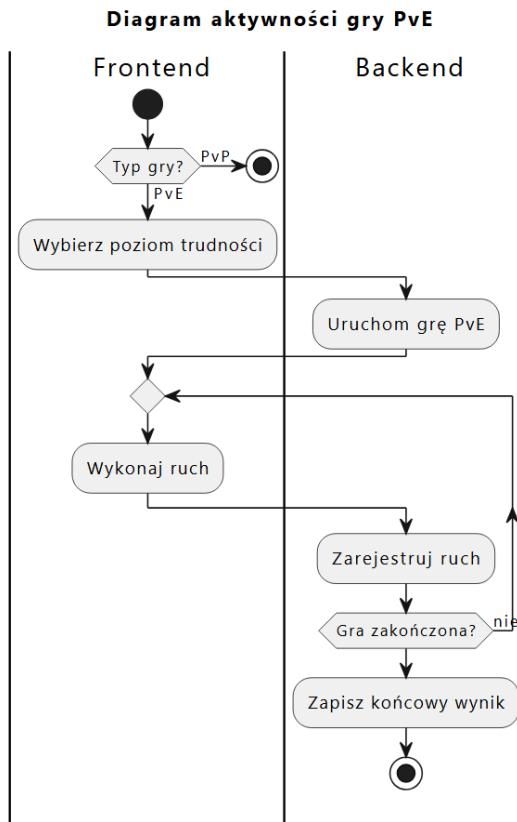
Celem niniejszej dokumentacji jest przedstawienie struktury implementacyjnej systemu bez umieszczania pełnych listingów kodu, które dostępne są w repozytorium GitHub. Dokumentacja kodu została przygotowana w postaci diagramów UML, które przedstawiają strukturę systemu oraz kluczowe procesy realizowane w aplikacji.

---

### Diagram aktywności gry PvE

Diagram przedstawia przebieg rozgrywki w trybie PvE z podziałem na warstwę frontendową oraz backendową. Proces rozpoczyna się od wyboru typu gry i poziomu trudności po stronie interfejsu użytkownika. Następnie backend inicjuje rozgrywkę oraz obsługuje rejestrowanie kolejnych ruchów gracza. Po zakończeniu gry zapisywany jest końcowy wynik.

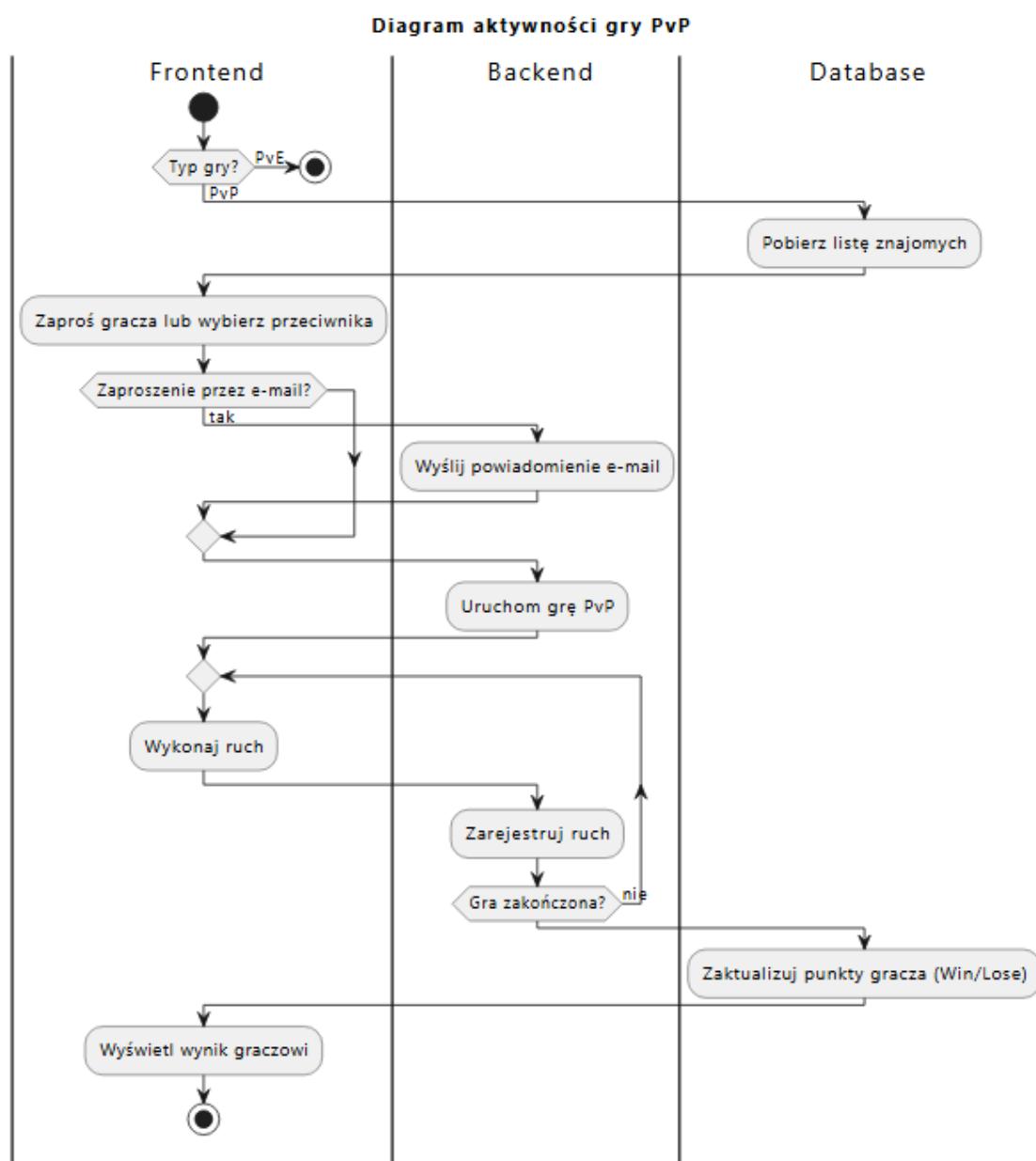
Diagram obrazuje cykliczny charakter rozgrywki oraz podział odpowiedzialności pomiędzy warstwą prezentacji a logiką serwera.



## Diagram aktywności gry PvP

Diagram prezentuje przebieg gry w trybie PvP z uwzględnieniem interakcji pomiędzy frontendem, backendem oraz bazą danych. Użytkownik może zaprosić przeciwnika lub wybrać go z listy znajomych. Backend obsługuje proces zaproszeń, uruchomienie rozgrywki oraz rejestrowanie ruchów graczy.

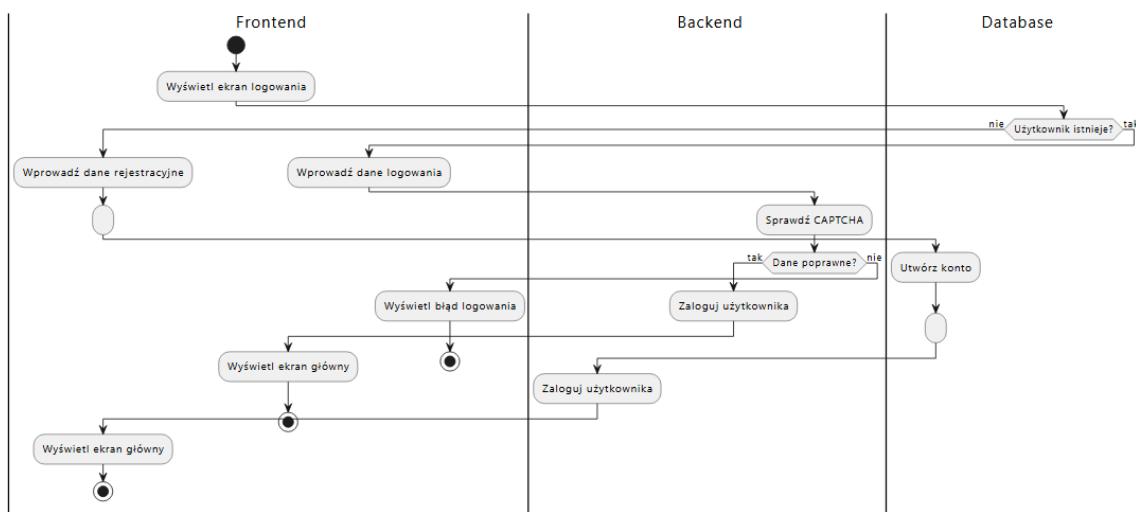
Po zakończeniu gry następuje aktualizacja punktów gracza w bazie danych oraz wyświetlenie wyniku w interfejsie użytkownika.



## Diagram aktywności logowania i rejestracji

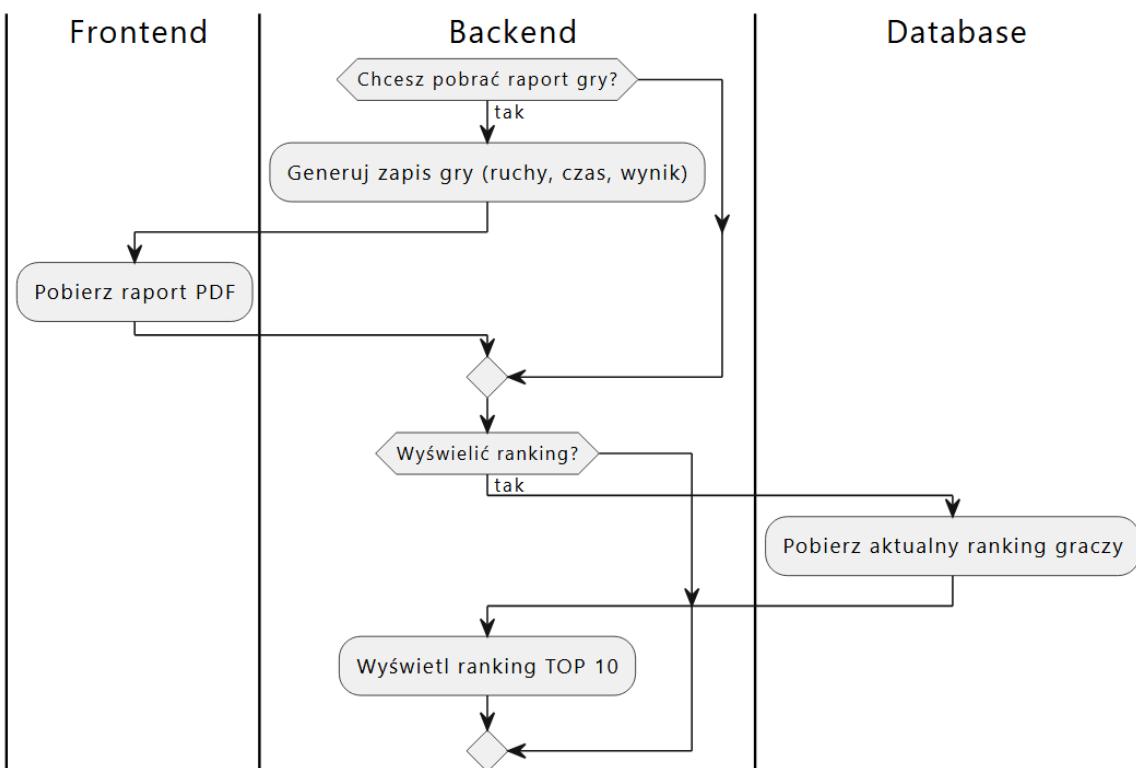
Diagram przedstawia proces logowania oraz rejestracji użytkownika w systemie. Użytkownik wprowadza dane uwierzytelniające w warstwie frontendowej, które następnie są weryfikowane przez backend. Proces obejmuje sprawdzenie poprawności danych, obsługę mechanizmu CAPTCHA oraz tworzenie konta w przypadku nowego użytkownika.

Po pomyślnej autoryzacji użytkownik zostaje przekierowany do ekranu głównego aplikacji.



## Diagram aktywności raportu gry i rankingu

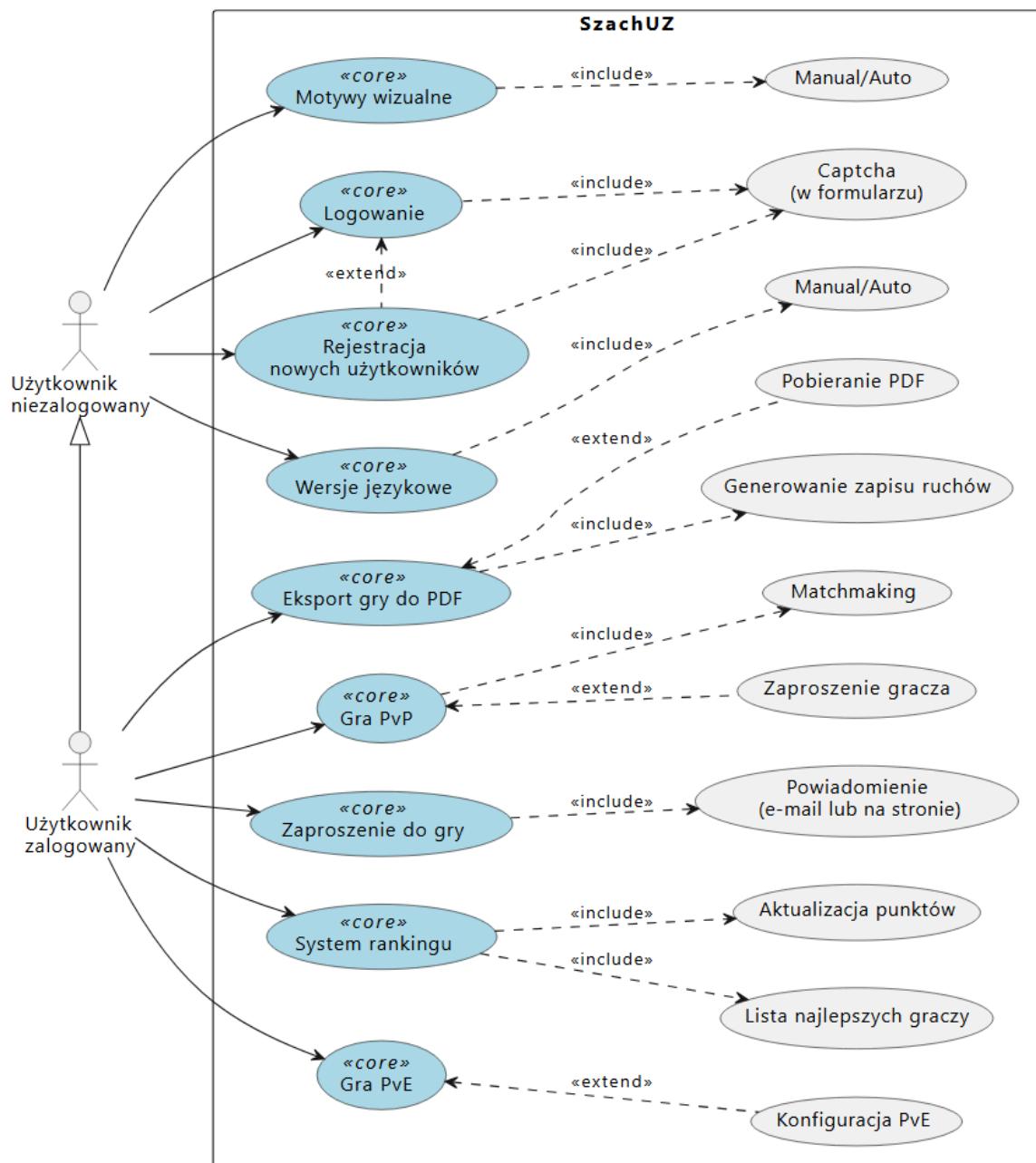
Diagram ilustruje proces generowania raportu z rozgrywki oraz wyświetlania rankingu graczy. Backend generuje zapis gry zawierający ruchy, czas trwania oraz wynik, który następnie może zostać pobrany przez użytkownika w formacie PDF. Alternatywnie system umożliwia pobranie aktualnego rankingu graczy z bazy danych i wyświetlenie listy najlepszych wyników.



## Diagram przypadków użycia systemu

Diagram przypadków użycia przedstawia główne funkcjonalności systemu dostępne dla użytkowników niezalogowanych oraz zalogowanych. Obejmuje on m.in. logowanie, rejestrację, rozgrywkę w trybie PvP i PvE, system rankingowy, zapraszanie graczy oraz eksport gry do pliku PDF.

Diagram pokazuje również zależności pomiędzy przypadkami użycia, takie jak relacje «include» i «extend», co pozwala na czytelne przedstawienie struktury funkcjonalnej aplikacji.



# Diagram klas systemu

Diagram klas prezentuje strukturę aplikacji od strony backendu, w tym główne encje domenowe, repozytoria oraz klasy odpowiedzialne za obsługę logiki biznesowej. Przedstawiono relacje pomiędzy użytkownikami, preferencjami, rankingiem oraz mechanizmami autoryzacji.

Diagram obrazuje podział odpowiedzialności pomiędzy warstwą dostępu do danych, logiką aplikacyjną oraz warstwą API.

