



# SIMPLE CHECKERS

**Autorzy:**

Przemysław Janiszewski, Julia Kluk, Jakub Kleszcz

## Spis treści

<b>1. SPECYFIKACJA PROJEKTU:</b> .....	<b>3</b>
1.1 KONCEPCJA I OPIS:.....	3
1.2 PODZIAŁ SYSTEMU:.....	3
1.3 PRZEPŁYW DANYCH:.....	4
<b>2. WYBRANA METODYKA:</b> .....	<b>6</b>
<b>3. SŁOWNIK POJĘĆ:</b> .....	<b>7</b>
<b>4. DIAGRAM PRZYPADKÓW UŻYCIA</b> .....	<b>8</b>
<b>5. WYMAGANIA FUNKCJONALNE:</b> .....	<b>16</b>
<b>6. WYMAGANIA NIEFUNKCJONALNE:</b> .....	<b>17</b>
<b>7. DIAGRAM WYMAGAŃ:</b> .....	<b>18</b>
<b>8.ZASTOSOWANE TECHNOLOGIE:</b> .....	<b>19</b>
<b>9. DIAGRAM AKTYWNOŚCI:</b> .....	<b>20</b>
<b>10. DIAGRAM KLAS:</b> .....	<b>31</b>
<b>11. DIAGRAM SEKWENCJI:</b> .....	<b>34</b>
<b>12. DIAGRAM KOMPONENTÓW:</b> .....	<b>44</b>
<b>13. DIAGRAM WDROŻEŃ</b> .....	<b>45</b>
<b>14. PRZYKŁADY WSPÓŁPRACY ZESPOŁU:</b> .....	<b>46</b>
<b>15. TESTY JEDNOSTKOWE:</b> .....	<b>48</b>
<b>16. REFAKTORYZACJA KODU:</b> .....	<b>50</b>
<b>17. WZORZEC PROJEKTOWY:</b> .....	<b>54</b>
<b>18. ZASADY GRY:</b> .....	<b>55</b>
<b>19. SZTUCZNA INTELIGENCJA:</b> .....	<b>56</b>
<b>20. INSTRUKCJA OBSŁUGI PROGRAMU – POMOC:</b> .....	<b>58</b>
<b>21. PODSUMOWANIE:</b> .....	<b>61</b>

## 1. Specyfikacja projektu:

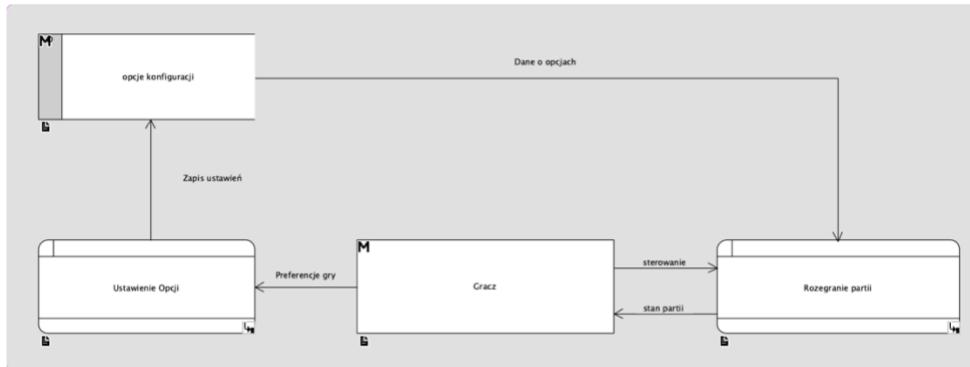
### 1.1 Koncepcja i opis:

- **Plansza:** szachownica 8x8 pól
- **Pionki:** zwykłe i królowe – o jednakowym motywie, dla danego gracza
- **Gracze:** uczestnicy gry, których zadaniem jest pozbyć się pionków przeciwnika lub doprowadzić go do sytuacji, gdy nie posiada on ruchu (zakleszczenia) – przestrzegając przy tym określonych zasad.
- **Zasady Gry:** opisane szczegółowo w punkcie **§18**
- **Interfejs użytkownika:** służy do komunikacji między graczem, a systemem
- **Mechanizm gry:** zajmuje się obsługą ruchów wykonywanych przez graczy
- **Algorytm sztucznej inteligencji (AI)** – pozwala komputerowi podejmować decyzję o swoich ruchach na podstawie analizy obecnego stanu gry (funkcja ocenяająca pozycje).
- **Tryb Gry:** możliwość gry z komputerem lub lokalnie z graczem nr.2
- **Opcje personalizacji:** możliwość wybrania motywu: pionków, planszy, trybu gry.
- **Statystyki i rankingi:** System pozwala na gromadzenie informacji o rozegranych partiach oraz opracowywanie na ich podstawie rankingu.
- **Logowanie i rejestracja:** możliwość zakładania konta, oraz logowania się do niego w celu zapisu statystyk

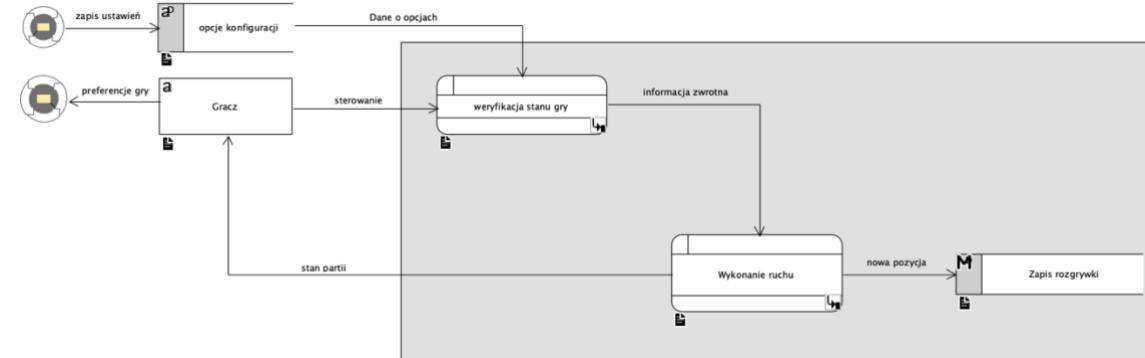
### 1.2 Podział systemu:

- **Podsystem gry** -> mechanizm rozgrywki (pętla gry wywołująca się określoną ilość razy na sekundę)
- **Podsystem interfejsu** -> wyświetlanie aktualnego stanu gry, umożliwienie wybierania ruchów, komunikacja między użytkownikiem a programem
- **Podsystem AI** -> podejmowanie decyzji o wykonanych ruchach w trybie gry z komputerem
- **Podsystem bazy danych** -> zbieranie informacji oraz udostępnianie ich odpowiednim modułom
- **Podsystem menu głównego** -> możliwość rozpoczęcie rozgrywki, wejście w opcje, przejście do sekcji: credits (informacje o autorach) lub wylogowania się
- **Podsystem menu opcji** -> możliwość wejścia do archiwum, obejrzenia rankingu, zmiany parametrów rozgrywki lub konta.
- **Podsystem rankingu i statystyk** -> możliwość obejrzenia zapisu ostatnich partii lub swojej pozycji w rankingu

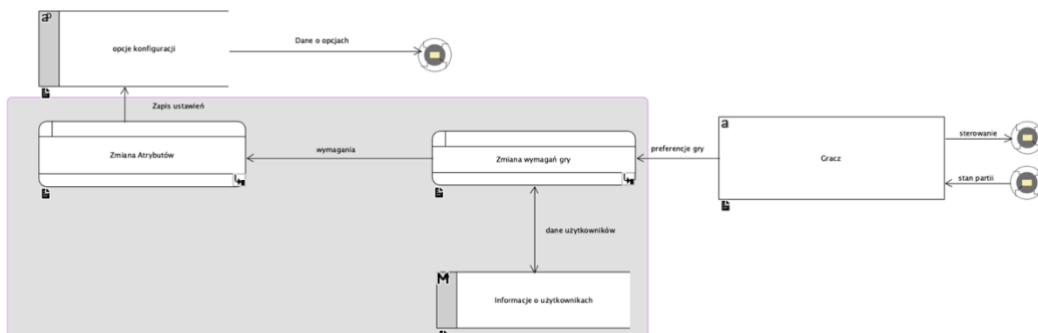
### 1.3 Przepływy danych:



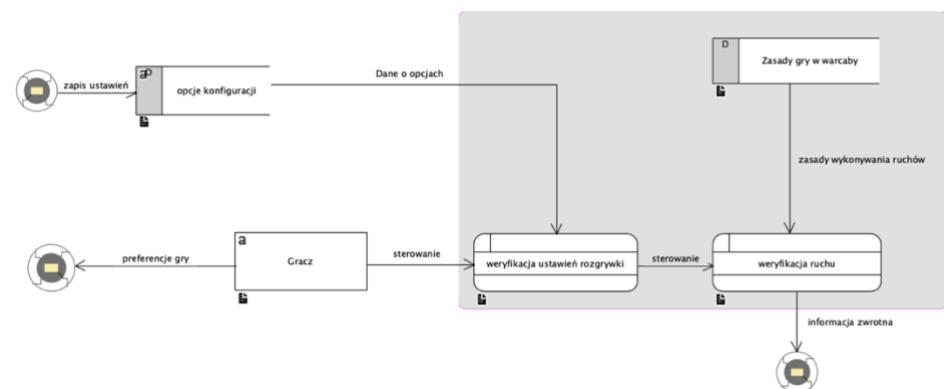
§ Poziom 0



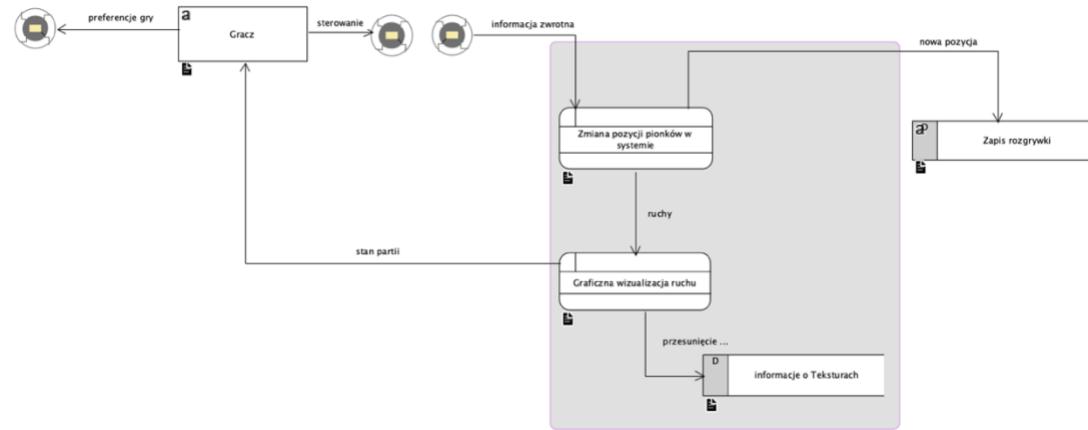
§ Poziom 1.1 – dekompozycja procesu: rozegranie partii



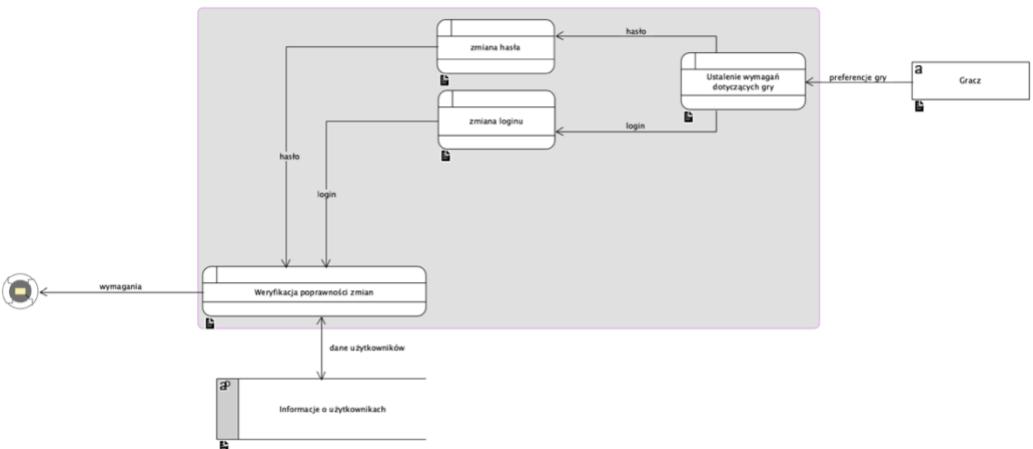
§ Poziom 1.2 – dekompozycja procesu: ustawienie opcji



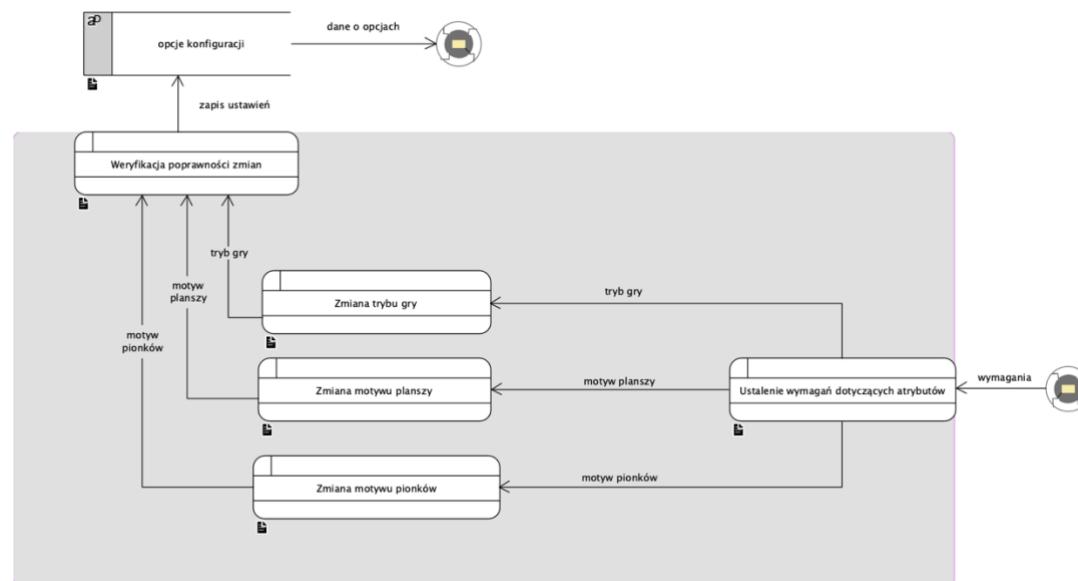
§ Poziom 2.1 – dekompozycja procesu: weryfikacja stanu gry



§ Poziom 2.2– dekompozycja procesu: wykonanie ruchu



§ Poziom 2.3– dekompozycja procesu: zmiana wymagań gry



§ Poziom 2.4– dekompozycja procesu: zmiana atrybutów

## 2. Wybrana metodyka:

**Kanban (Agile)** - Kanban to metodyka należąca do rodziny Agile, pozwalająca zarządzać projektami. Polega na rozdzieleniu etapów pracy nad projektem oraz możliwości obserwacji jego postępów. Funkcjonalność tą umożliwia tablica (jap. Kanban), w której poszczególne zadania posiadające jeden z 3 statusów:

- „do zrobienia”
- „w trakcie”
- „zrobione”

Taki podział pozwala lepiej zrozumieć ideę projektu oraz uniknąć etapów o niskiej wydajności pracy.

**W tablica Kanban mogą znaleźć się również inne, użyteczne informacje takie jak:**

- przydatne linki
- komentarze
- dokumenty
- obrazki

### Główne zalety metodyki:

- Pełna przejrzystość procesu
- Możliwość określania czasu i priorytetu zadań
- Zwiększenie wydajności zespołu
- Eliminacja marnotrawstwa
- Ciągłość pętli(informacji) zwrotnych

### Główne wady metodyki:

- Konieczność poświęcenia czasu na rozplanowanie projektu
- Wymaga dużej elastyczności

TO DO	IN PROGRESS	DONE

Najprostsza, przykładowa tablica Kanban

### Realizacja metodyki:

Dzięki użyciu platformy: <https://trello.com/>

### 3. Słownik pojęć:

- **PvP**- tryb gry: gracz vs gracz
- **PvE** – tryb gry: gracz vs środowisko (system, komputer)
- **Zakleszczenie** – sytuacja podczas rozgrywania partii, gdy dany gracz pomimo posiada pionków, nie posiada możliwych ruchów do wykonania
- **Diagonal\_down ( $c_1$ )** – identyfikator przekątnej dolnej
- **Diagonal\_up ( $c_2$ )** – identyfikator przekątnej górnej
- **Waga** – ustalona doświadczalnie stała, pomagająca określić najlepszy ruch w pozycji
- **Funkcja heurystyczna** – funkcja starająca się znaleźć najbardziej optymalny ruch w danej pozycji
- **Zero sum game** – gra w której jeden gracz zyskuje kosztem drugiego (np. Warcaby)
- **Min-max algorytm** – algorytm polegający na maksymalizowaniu swojego wyniku i minimalizowaniu wyniku przeciwnika
- **AI (artificial intelligence)** – sztuczna inteligencja

## 4. Diagram Przypadków użycia

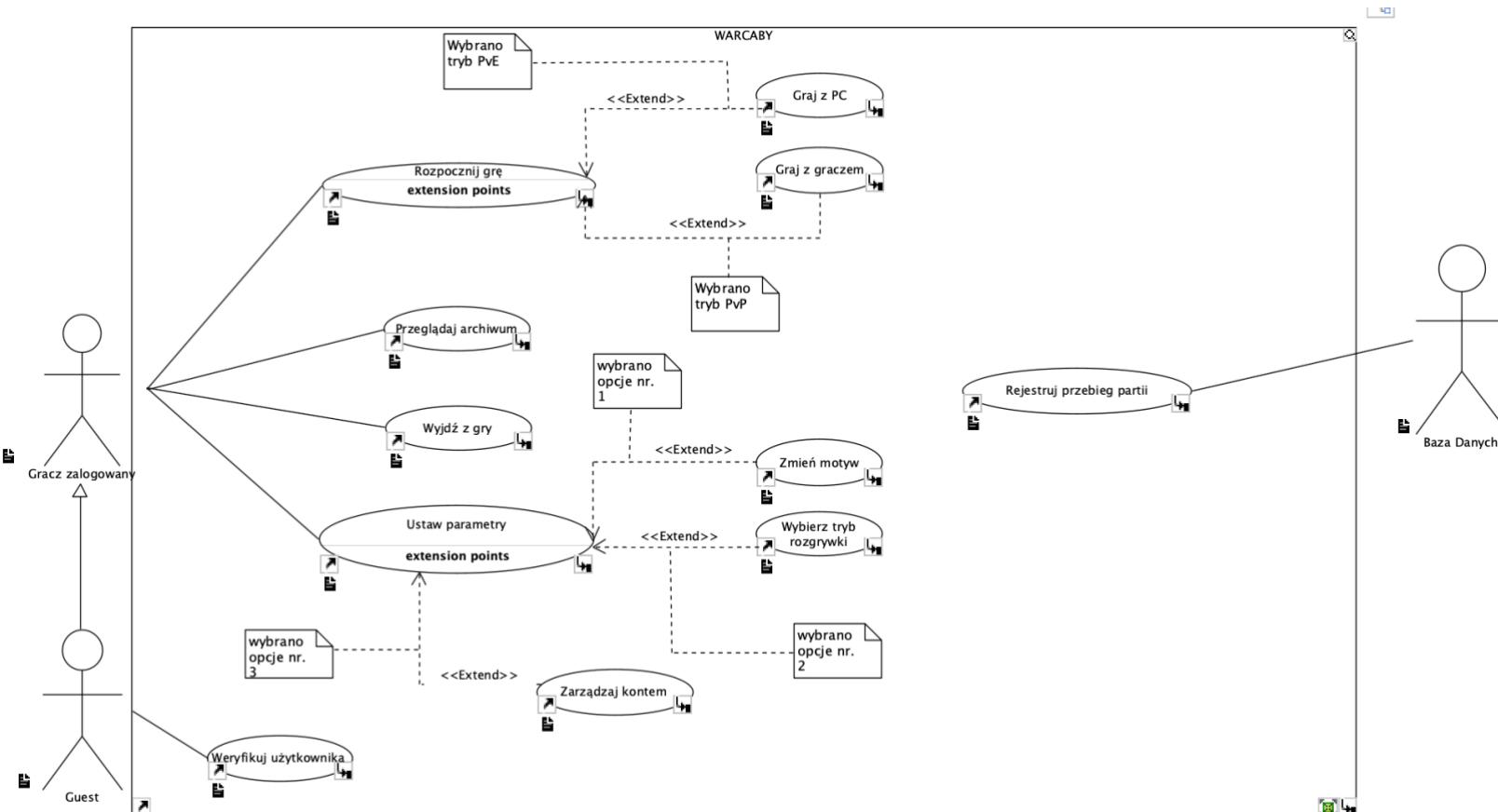


Diagram przypadków użycia (PU)

## ■ 1. Weryfikuj użytkownika

ID: UC16

- 1) Przypadek użycia: "weryfikuj użytkownika" pozwala na **utworzenie konta gracza** (login i hasło), dzięki któremu będzie się on mógł logować do swojego konta, zapisywać swoje wyniki i rozgrywać mecze.
- 2) W przypadku, gdy użytkownik posiada aktywne konto, może zalogować się na nie.
- 2) **Niepowodzenie** przypadku uniemożliwi rozgrywanie partii warcabowych – system nie przewiduje możliwości grania jako gość.

**Przypadek użycia zawiera:**

- Scenariusze: główny, alternatywny i wyjątku
- Szczegóły
- Wymagania

<b>Uzasadnienie</b>	Brak pomyślnej weryfikacji uniemożliwia skorzystanie z aplikacji
<b>Główni aktorzy</b>	Guest
<b>Level</b>	User
<b>Complexity</b>	Medium
<b>Use Case Status</b>	Complete
<b>Implementation Status</b>	Partially Complete
<b>Preconditions</b>	<ul style="list-style-type: none"><li>– Menu rejestracji lub</li><li>– Menu logowania</li></ul>
<b>Post-conditions</b>	– Menu Rozgrywki
<b>Author</b>	Przemysław Janiszewski
<b>Assumptions</b>	Działający podsystem rejestracji/logowania

### 1.1. Scenariusze

#### 1.1.1. Scenariusz główny

1. Użytkownik tworzy nowe konto.
2. Użytkownik podaje: login i hasło.
3. Dane zostają zapisane w bazie danych
4. Użytkownik zostaje zalogowany i może korzystać już z funkcjonalności konta.

#### 1.1.2. Scenariusz alternatywny a)

1. Użytkownik posiada aktywne konto.
2. Użytkownik podaje login i hasło.
3. Dane zostają zweryfikowane z danymi w bazie danych.
4. Wprowadzone dane są poprawne.

5. Użytkownik zostaje zalogowany i może korzystać już z funkcjonalności konta.

#### 1.1.3. Scenariusz alternatywny b)

1. Użytkownik posiada aktywne konto.
2. Użytkownik podaje login i hasło.
3. Dane zostają zweryfikowane z danymi w bazie danych.
4. Wprowadzone dane nie są poprawne.
5. Użytkownik podejmuje ponowną próbę podania danych.

#### 1.1.4. Scenariusz wyjątku

1. W momencie weryfikacji użytkownika występuje niezidentyfikowany błąd.
2. Informacja o błędzie zostaje przechwycona przez system.
3. Aplikacja wyłączy się, gdzie nie możliwe jest jej dalsze działanie.

## ■ 1. Ustaw parametry

ID: UC04

1) Przypadek użycia "Ustaw parametry" daje możliwość zmiany ustawień gry

2) Średni priorytet - Gracz wpływa na własności rozgrywki

### Przypadek użycia zawiera:

- Scenariusze: główny, alternatywny i dwa scenariusze wyjątku
- Szczegóły
- Wymagania
- Plan testów

<b>Uzasadnienie</b>	Zmiana parametrów powoduje modyfikację przebiegu rozgrywki
<b>Główni aktorzy</b>	Gracz zalogowany
<b>Level</b>	User
<b>Complexity</b>	Medium
<b>Use Case Status</b>	Base
<b>Implementation Status</b>	Complete
<b>Preconditions</b>	- Użytkownik jest w Menu pobocznym (opcje)
<b>Post-conditions</b>	- Ustawienia zapisały się - Użytkownik znajduje się w Menu pobocznym (opcje)
<b>Author</b>	Julia Kluk
<b>Assumptions</b>	Zapisywanie danych działa poprawnie

### 1.1. Scenariusze

#### 1.1.1. Scenariusz Główny

1. Użytkownik otwiera menu opcji (atrybuty)
2. Użytkownik wybiera jedną z 3 opcji zmiany parametrów
3. Użytkownik zapisuje zmianę ustawień i wychodzi

#### 1.1.2. Scenariusz Alternatywny

1. Użytkownik otwiera menu opcji (atrybuty)
2. Użytkownik wybiera jedną z 3 opcji zmiany parametrów
3. Użytkownik nie zapisuje zmian ustawień i wychodzi

#### 1.1.3. Scenariusz Wyjątku

1. W momencie zmiany parametrów występuje niezidentyfikowany błąd.
2. Informacja o błędzie zostaje przechwycona przez system.
3. Funkcjonalność wyłączy się, gdy nie możliwe jest jej dalsze działanie

## ■ 1. Zarządzaj kontem

ID: UC05

1) Przypadek użycia "Zarządzaj kontem" daje dwie możliwości wprowadzenia zmian na koncie:

- zmiana nicku
- zmiana hasła

2) Funkcja wprowadza dodatkowo opcje "wyloguj się", gdyby użytkownik nie chciał w tej chwili korzystać z używanego konta

3) Niepowodzenie funkcji skutkuje pozostawieniem konta bez zmian, ciągle zalogowanego

### Przypadek użycia zawiera:

- Scenariusze: główny, wyjątku oraz dwa scenariusze alternatywne
- Szczegóły
- Wymagania
- Plan testów

<b>Uzasadnienie</b>	Funkcja daje możliwość edycji danych na zalogowanym koncie
<b>Główni aktorzy</b>	Gracz zalogowany
<b>Level</b>	User
<b>Complexity</b>	Medium
<b>Use Case Status</b>	Complete
<b>Implementation Status</b>	Partially Complete
<b>Preconditions</b>	- Użytkownik znajduje się w menu opcji gry
<b>Post-conditions</b>	- Użytkownik dokonał zmiany parametrów konta
<b>Author</b>	Julia Kluk
<b>Assumptions</b>	Użytkownik ma dostęp do funkcjonalności zalogowanego konta

### 1.1. Scenariusze

#### 1.1.1. Scenariusz Główny

1. Użytkownik kliką przycisk wyloguj się.
2. Użytkownik zostaje wylogowany z konta.
3. Użytkownik traci dostęp do funkcji dostępnych dla zalogowanego użytkownika

#### 1.1.2. Scenariusz Alternatywny 1

1. Użytkownik przechodzi do podsystemu zmiany hasła.
2. Użytkownik podaje nowe hasło.

## ● 1. Wybierz tryb rozgrywki

ID: UC08

1) Przypadek użycia "Wybierz tryb rozgrywki" pozwala graczowi **mechanizm na zasadzie którego będzie toczyła się partia**. Użytkownik może rozgrywać partie w trybie Gracz vs Gracz (PvP) lub Gracz vs Środowisko (PvE)

2) Niepowodzenie tej funkcjonalności daje **mniejsze możliwości dostosowania rozgrywki** pod żądane preferencje.

### Przypadek użycia zawiera:

- Scenariusze: główny, alternatywny i wyjątku
- Szczegóły
- Wymagania
- Plan testów

<b>Uzasadnienie</b>	Wybieranie trybu rozgrywki ma istotny wpływ na jej przebieg
<b>Główni aktorzy</b>	Gracz Guest
<b>Level</b>	User
<b>Complexity</b>	High
<b>Use Case Status</b>	Complete
<b>Implementation Status</b>	Partially Complete
<b>Preconditions</b>	- Użytkownik znajduje się w Opcjach Gry
<b>Post-conditions</b>	- Ustawienia zostały zapisane - Tryb gry został określony przez użytkownika
<b>Author</b>	Julia Kluk
<b>Assumptions</b>	- Każda partia odbywa się w wybranym trybie - Użytkownik ma możliwość wyboru tryby rozgrywki

## 1.1. Scenariusze

### 1.1.1. Scenariusz Główny

1. Użytkownik ma podgląd na możliwe tryby rozgrywki.
2. Gracz wybiera jedną z opcji
3. Użytkownik dokonuje zapisu wybranych przez siebie parametrów.
4. Użytkownik wraca do odpowiedniego podmenu.

### 1.1.2. Scenariusz Alternatywny

1. Użytkownik przegląda możliwe tryby rozgrywki
2. Użytkownik nie zmienia trybu rozgrywki
3. Użytkownik wraca do odpowiedniego podmenu.

### 1.1.3. Scenariusz Wyjątku

1. W momencie ustawiania parametrów występuje niezidentyfikowany błąd.
2. Informacja o błędzie zostaje przechwycona przez system.
3. Funkcjonalność wyłączy się, gdy nie możliwe jest jej dalsze działanie.

## ■1. Zmień motyw

ID: UC06

- Przypadek użycia "Zmień motyw" daje możliwość wyboru jednego z pośród dwóch motywów graficznych aplikacji.

Możliwe wybory:

- tryb jasny
- tryb ciemny

2) Niepowodzenie skutkuje **zostaniem przy ostatnim trybie**

**Przypadek użycia zawiera:**

- Scenariusze: główny, alternatywny i wyjątku
- Szczegóły
- Wymagania
- Plan testów

<b>Uzasadnienie</b>	Funkcjonalność wpływa jedynie na wygląd tekstur
<b>Główni aktorzy</b>	Gracz Guest
<b>Level</b>	User
<b>Complexity</b>	Medium
<b>Use Case Status</b>	Complete
<b>Implementation Status</b>	Partially Complete
<b>Preconditions</b>	- Użytkownik znajduje się w Opcjach Gry (Atrybuty)
<b>Post-conditions</b>	- Ustawienia zostały zapisane - Interfejs zmienił się
<b>Author</b>	Julia Kluk
<b>Assumptions</b>	Użytkownik ma możliwość zmiany motywu

### 1.1. Scenariusze

#### 1.1.1. Scenariusz Główny

- Użytkownik ma wgląd w zakres możliwych do wyboru motywów
- Gracz wybiera jedną z dostępnych opcji
- Interfejs zmienia się według wybranego trybu

#### 1.1.2. Scenariusz Alternatywny

- Użytkownik ma wgląd w zakres możliwych do wyboru motywów
- Użytkownik nie zmienia motywu
- Ustawienia pozostają niezmienione

#### 1.1.3. Scenariusz Wyjątku

- W momencie ustawiania parametrów występuje niezidentyfikowany błąd.
- Informacja o błędzie zostaje przechwycona przez system.
- Funkcjonalność wyłączy się, gdy nie możliwe jest jej dalsze działanie.

## ■1. Wyjdź z gry

ID: UC03

- Przypadek użycia: "Wyjdź z gry" pozwala na **wyjście z gry**, gdy gracz ma taką potrzebę.

**Przypadek użycia zawiera:**

- Scenariusze: główny, alternatywny i wyjątku
- Szczegóły
- Wymagania
- Plan testów

<b>Uzasadnienie</b>	Gracz musi mieć możliwość wyjścia z programu
<b>Główni aktorzy</b>	Gracz zalogowany
<b>Level</b>	User
<b>Complexity</b>	Low
<b>Use Case Status</b>	Complete
<b>Implementation Status</b>	Partially Complete
<b>Preconditions</b>	- Gracz chce zakończyć działanie aplikacji
<b>Post-conditions</b>	- Użytkownik zamknie okno gry - powrót do ekranu startowego systemu operacyjnego
<b>Author</b>	Jakub Kleszcz
<b>Assumptions</b>	- Gracz ma możliwość wyjścia z gry

### 1.1. Scenariusze

#### 1.1.1. Scenariusz Główny

- Gracz podejmuje decyzję o wyjściu z aplikacji
- Gracz przechodzi do menu głównego aplikacji
- Gracz wylogowuje się (przy pomocy odpowiedniego przycisku)
- Gracz wychodzi z gry (przy pomocy odpowiedniego przycisku)

#### 1.1.2. Scenariusz Wyjątku

- Gracz decyduje się na zakończenie gry
- Gracz przechodzi do menu głównego aplikacji
- W momencie wykonywania akcji występuje niezidentyfikowany błąd.
- Informacja o błędzie zostaje przechwycona przez system.
- Program zostaje przerwany, gdy niemożliwe jest jego dalsze działanie.

## ● 1. Przeglądaj archiwum

ID: UC07

1) Przypadek użycia: "Przeglądaj archiwum" pozwala na **sprawdzenie zapisów rozegranych partii**.

### Przypadek użycia zawiera:

- Scenariusze: główny, alternatywny i wyjątku
- Szczegóły
- Wymagania
- Plan testów

<b>Uzasadnienie</b>	Gracz ma możliwość sprawdzenia zapisów z poprzednio rozgrywanych partii
<b>Główni aktorzy</b>	Gracz zalogowany
<b>Level</b>	User
<b>Complexity</b>	Medium
<b>Use Case Status</b>	Complete
<b>Implementation Status</b>	Partially Complete
<b>Preconditions</b>	– Użytkownik znajduje się w odpowiednim podmenu
<b>Post-conditions</b>	– Dostęp do zapisu ostatnio rozegranych partii
<b>Author</b>	Jakub Kleszcz
<b>Assumptions</b>	Gracz ma możliwość dostępu do zapisu ostatnio rozegranych partii

### 1.1. Scenariusze

#### 1.1.1. Scenariusz Główny

1. Gracz chce przeprowadzić analizę jednej z ostatnich partii
2. Gracz wybiera interesującą go rozgrywkę
3. Gracz przegląda wybrany przez siebie zapis partii

#### 1.1.2. Scenariusz Wyjątku

1. W momencie wykonywania akcji występuje niezidentyfikowany błąd.
2. Informacja o błędzie zostaje przechwycona przez system.
3. Funkcjonalność wyłączy się, gdy niemożliwe jest jej dalsze działanie.

## ● 1. Rozpoczni grę

ID: UC02

1) Przypadek użycia: "Rozpocznij grę" pozwala na **rozpoczęcie rozgrywki** przez zalogowanego gracza.

2) Niepowodzenie w rozpoczęciu gry skutkuje z pozostaniem w menu gry.

### Przypadek użycia zawiera:

- Scenariusze: główny, alternatywny i wyjątku
- Szczegóły
- Wymagania
- Plan testów

<b>Uzasadnienie</b>	Gracz musi mieć możliwość rozpoczęcia rozgrywki
<b>Główni aktorzy</b>	Gracz zalogowany
<b>Level</b>	User
<b>Complexity</b>	High
<b>Use Case Status</b>	Base
<b>Implementation Status</b>	Scheduled
<b>Preconditions</b>	– Użytkownik jest w Menu Głównym
<b>Post-conditions</b>	– Zakończenie partii warcabowej z wybranym przeciwnikiem – Powrót użytkownika do Menu Głównego albo opcja rewanżu
<b>Author</b>	Jakub Kleszcz
<b>Assumptions</b>	Gracz chce rozpocząć gre w Warcaby

### 1.1. Scenariusze

#### 1.1.1. Scenariusz główny

1. Gracz decyduje się rozpoczęć rozgrywkę w warcaby
2. Gracz wybiera tryb gry PvE
3. Rozpoczęcie partii w warcaby w wybranym trybie

#### 1.1.2. Scenariusz alternatywny

1. Gracz decyduje się rozpoczęć rozgrywkę w warcaby
2. Gracz wybiera tryb gry PvP
3. Rozpoczęcie partii w warcaby w wybranym trybie

#### 1.1.3. Scenariusz wyjątku

1. W momencie rozpoczęcia gry występuje niezidentyfikowany błąd.
2. Informacja o błędzie zostaje przechwycona przez system.
3. Gra wyłączy się, gdy niemożliwe jest jej dalsze działanie

## ■1. Graj z PC

ID: UC13

1) Przypadek użycia: "Graj z PC" pozwala na rozpoczęcie gry z PC.

2) Niepowodzenie przypadku uniemożliwi rozgrywanie partii warcabowej z PC.

**Przypadek użycia zawiera:**

- Scenariusze: główny, alternatywny i wyjątku
- Szczegóły
- Wymagania
- Plan testów

<b>Uzasadnienie</b>	Gracz ma możliwość rozpoczęcia gry z komputerem
<b>Level</b>	User
<b>Complexity</b>	High
<b>Use Case Status</b>	Complete
<b>Implementation Status</b>	Partially Complete
<b>Preconditions</b>	Uruchomienie gry w określonym trybie: <a href="#">Rozpocznij gre</a>
<b>Post-conditions</b>	- Rozegranie poprawnej partii w trybie PvE
<b>Author</b>	Jakub Kleszcz
<b>Assumptions</b>	Gracz może rozpocząć rozgrywkę w trybie gry z komputerem

### 1.1. Scenariusze

#### 1.1.1. Scenariusz Główny

1. Gracz wybiera opcję rozgrywki z PC
2. Gracz rozgrywa niezakłóconą partię w wybranym trybie
3. Jedna z stron wygrywa poprzez zbitie wszystkich pionków przeciwnika
4. Na ekranie przez określony czas wyświetla się informacja o wygranej określonej z stron

#### 1.1.2. Scenariusz Alternatywny

1. Gracz wybiera opcję rozgrywki z PC
2. Gracz rozgrywa niezakłóconą partię w wybranym trybie
3. Jedna z stron wygrywa poprzez doprowadzenie do zakleszczenia się przeciwnika (nie ma on dozwolonego ruchu)
4. Na ekranie przez określony czas wyświetla się informacja o wygranej określonej z stron

#### 1.1.3. Scenariusz Wyjątku

1. W momencie gry w trybie PvE występuje niezidentyfikowany błąd.
2. Informacja o błędzie zostaje przechwycona przez system
3. Gra wyłączy się, gdy nie możliwe jest jej dalsze działanie
4. Postępy gracza zostają utracone

#### 1.1.4. Scenariusz Alternatywny 2

1. Gracz wybiera opcję rozgrywki z graczem
2. Gracz rozgrywa niezakłóconą partię w wybranym trybie
3. Obie strony przez 15 ruchów (po każdej z stron) nie wykonują bicia
4. Na ekranie przez określony czas wyświetla się informacja o remisie

## ■1. Graj z graczem

ID: UC14

1) Przypadek użycia: "Graj z graczem" pozwala na rozpoczęcie gry z drugim graczem.

2) Niepowodzenie przypadku uniemożliwi rozgrywanie partii warcabowej innym graczem.

**Przypadek użycia zawiera:**

- Scenariusze: główny, alternatywny i wyjątku
- Szczegóły
- Wymagania
- Plan testów

<b>Uzasadnienie</b>	Gracz ma możliwość rozpoczęcia rozgrywki z drugim graczem
<b>Level</b>	User
<b>Complexity</b>	Medium
<b>Use Case Status</b>	Complete
<b>Implementation Status</b>	Partially Complete
<b>Preconditions</b>	- Uruchomienie gry w określonym trybie: <a href="#">Rozpocznij gre</a>
<b>Post-conditions</b>	- Rozegranie poprawnej partii w trybie PvP
<b>Author</b>	Jakub Kleszcz
<b>Assumptions</b>	Gracz może rozpoczęć rozgrywkę w trybie gry z innym graczem

### 1.1. Scenariusze

#### 1.1.1. Scenariusz Główny

1. Gracz wybiera opcję rozgrywki z graczem
2. Gracz rozgrywa niezakłóconą partię w wybranym trybie
3. Jedna z stron wygrywa poprzez zbitie wszystkich pionków przeciwnika
4. Na ekranie przez określony czas wyświetla się informacja o wygranej określonej z stron

#### 1.1.2. Scenariusz Alternatywny

1. Gracz wybiera opcję rozgrywki z graczem
2. Gracz rozgrywa niezakłóconą partię w wybranym trybie
3. Jedna z stron wygrywa poprzez doprowadzenie do zakleszczenia się przeciwnika (nie ma on dozwolonego ruchu)
4. Na ekranie przez określony czas wyświetla się informacja o wygranej określonej z stron

#### 1.1.3. Scenariusz Wyjątku

1. W momencie gry w trybie PvP występuje niezidentyfikowany błąd.
2. Informacja o błędzie zostaje przechwycona przez system
3. Gra wyłączy się, gdy nie możliwe jest jej dalsze działanie
4. Postępy gracza zostają utracone
5. Gracz wybiera opcję rozgrywki z graczem

#### 1.1.4. Scenariusz Alternatywny 2

1. Gracz wybiera opcję rozgrywki z graczem
2. Gracz rozgrywa niezakłóconą partię w wybranym trybie
3. Obie strony przez 15 ruchów (po każdej z stron) nie wykonują bicia
4. Na ekranie przez określony czas wyświetla się informacja o remisie

## ■ 1. Rejestruj przebieg partii

ID: UC12

- 1) Przypadek użycia: "Rejestruj przebieg partii" powoduje dynamiczny **zapis przebiegu rozgrywki** do odpowiedniej sekcji bazy danych
- 2) Niepowodzenie zapisu **nie spowoduje awarii aplikacji**, lecz uniemożliwi odczyt statystyk gracza.

**Przypadek użycia zawiera:**

- Scenariusze: główny i wyjątku
- Szczegóły
- Wymagania
- Plan testów

<b>Uzasadnienie</b>	Błąd rejestracji przebiegu partii uniemożliwia odczytanie jej przebiegu
<b>Główni aktorzy</b>	• Baza Danych
<b>Level</b>	Summary
<b>Complexity</b>	Medium
<b>Use Case Status</b>	Complete
<b>Implementation Status</b>	Partially Complete
<b>Preconditions</b>	<ul style="list-style-type: none"><li>- Partia rozpoczęła się/użytkownik zaczął rozgrywkę</li></ul>
<b>Post-conditions</b>	<ul style="list-style-type: none"><li>- Partia zakończyła się</li><li>- Zebrano zakładane dane o przebiegu partii, podczas trwania rozgrywki</li></ul>
<b>Author</b>	Przemysław Janiszewski
<b>Assumptions</b>	Rejestracja przebiegu partii została poprawnie zainicjowana

### 1.1. Scenariusze

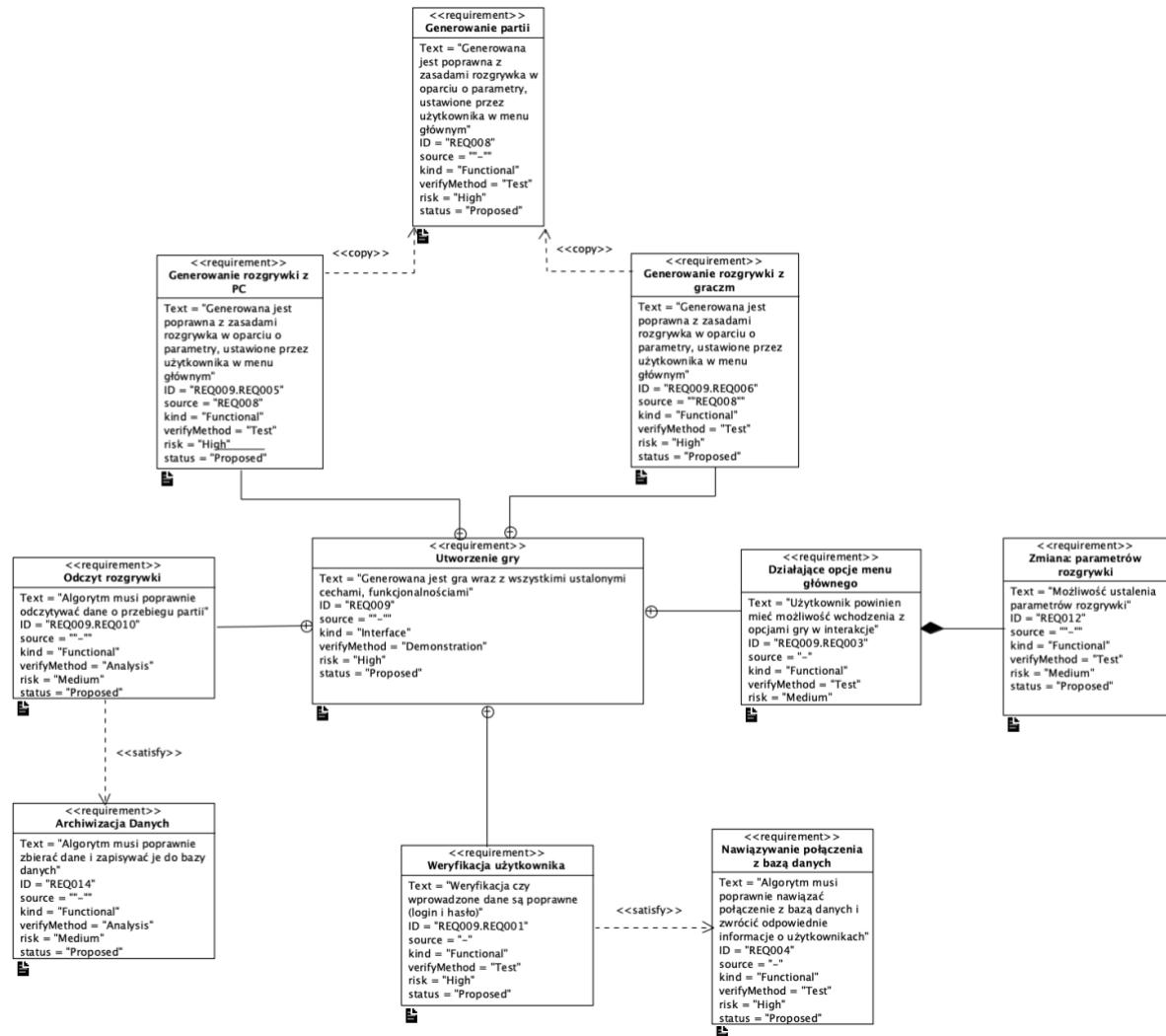
#### 1.1.1. Scenariusz Główny

1. Baza danych czeka na wykonanie ruchu przez jednego z graczy.
2. Baza danych pobiera w sposób dynamiczny informacje o ruchach wykonanych przez użytkownika.
3. Baza danych archiwizuje zebrane dane.

#### 1.1.2. Scenariusz wyjątku

1. W momencie pobierania danych występuje niezidentyfikowany błąd.
2. Informacja o błędzie zostaje przechwycona przez system.
3. Funkcjonalność wyłączy się, gdy nie możliwe jest jej dalsze działanie.

## 5. Wymagania funkcjonalne:



**1) Utworzenie prawidłowej rozgrywki** jest możliwe wyłącznie, gdy:

- Użytkownik został zweryfikowany
- Odczyt rozgrywki przebiega poprawnie
- Wszystkie opcje menu głównego działają poprawnie
- Program potrafi generować zarówno poprawną rozgrywkę z PC i/lub z Playerem

**2) Poprawna weryfikacja** jest możliwa, wyłącznie, gdy uda się nawiązać niezakłócone połączenie z bazą danych i odczytać z niej dane

**3) Analiza rozgrywki** może przebiegać poprawnie, wyłącznie, gdy archiwizacja danych będzie przeprowadzana w poprawny sposób

**4) Żeby wszystkie opcje menu działały poprawnie,** poszczególne jego składowe, także muszą działać poprawnie: opcja zmiany motywu, czy opcja wyboru poziomu trudności

**5) Generowanie rozgrywki z PC lub z Playerem** jest tożsame z wymaganiem docelowym

-> Generowanie rozgrywki [wymaganie docelowe, tylko do odczytu]

Diagram wymagań funkcjonalnych

## 6. Wymagania niefunkcjonalne:

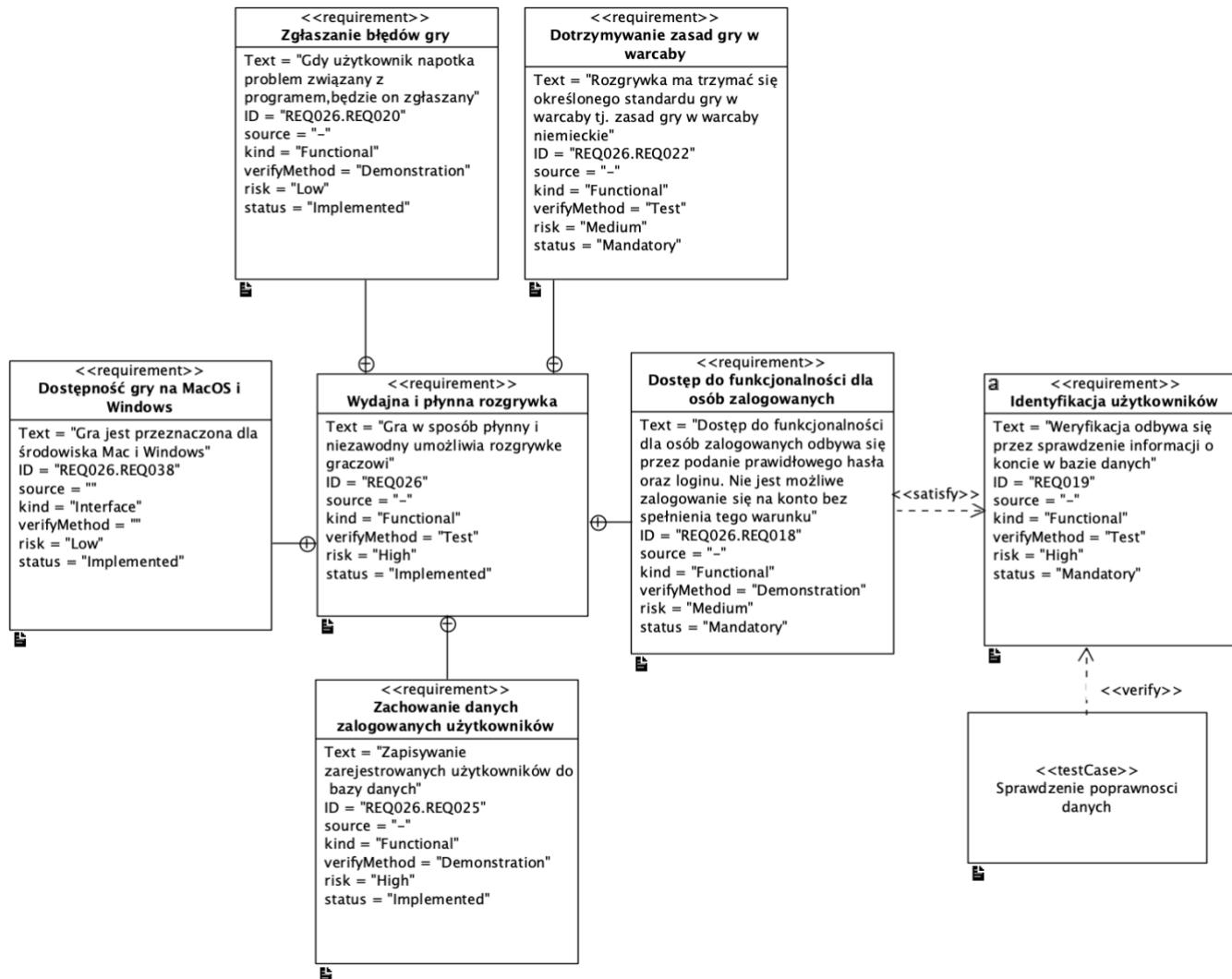
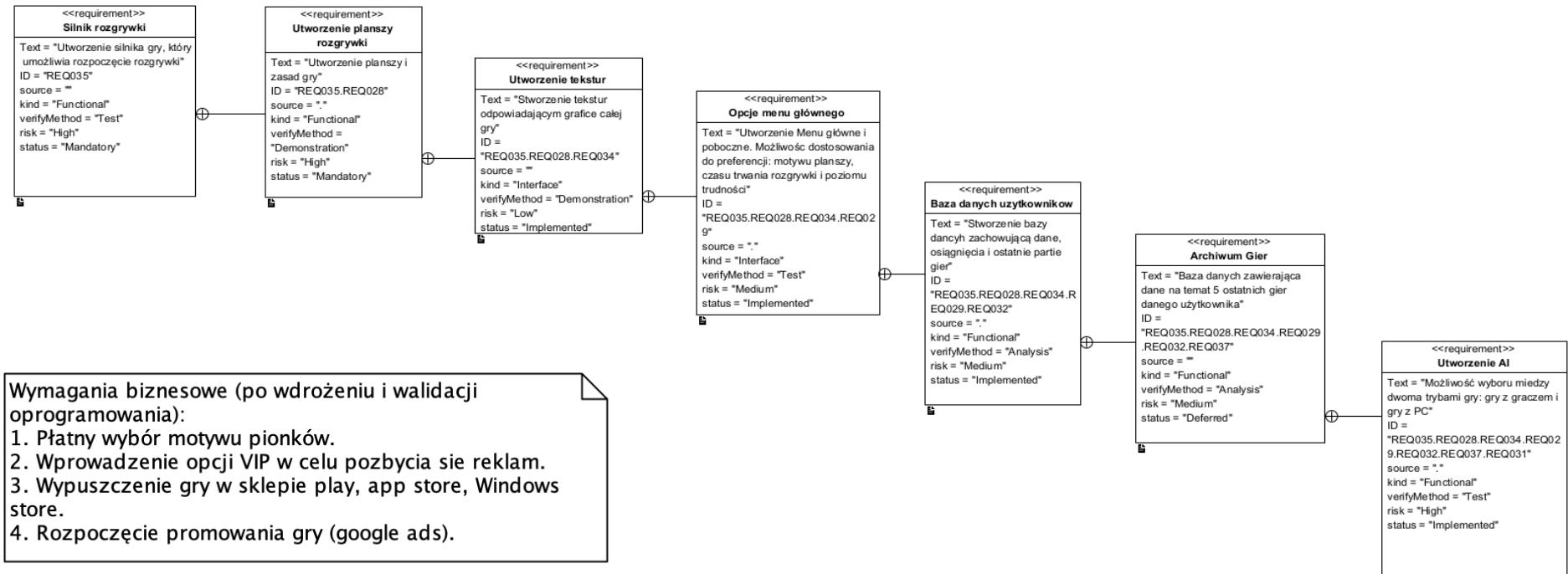


Diagram wymagań niefunkcjonalnych

## 7. Diagram Wymagań:



Przez cały okres tworzenia projektu, kierowaliśmy się powyższymi priorytetami. Były one na bieżąco aktualizowane, ale i tak niektórych nie udało się wykonać w 100%, z powodu zbyt małej ilości czasu.

Kod był tworzony według naszych preferencji, według powyższej hierarchii.

Wymagania biznesowe, których realizacje mamy na celu przeprowadzić w przyszłości, mają na celu pomóc nam zmonetyzować aplikację i przyniesienie zysku z stworzonego oprogramowania.

## 8.Zastosowane Technologie:



Simple and Fast Multimedia Library

Biblioteka dostarczająca prosty interfejs do obsługi modułów: systemu, okna, grafiki, audio oraz sieci.

Here is a list of all modules:

Audio module	Sounds, streaming (musics or custom sources), recording, spatialization
Graphics module	2D graphics module: sprites, text, shapes, ..
Network module	Socket-based communication, utilities and higher-level network protocols (HTTP, FTP)
System module	Base module of SFML, defining various utilities
Window module	Provides OpenGL-based windows, and abstractions for events and input handling

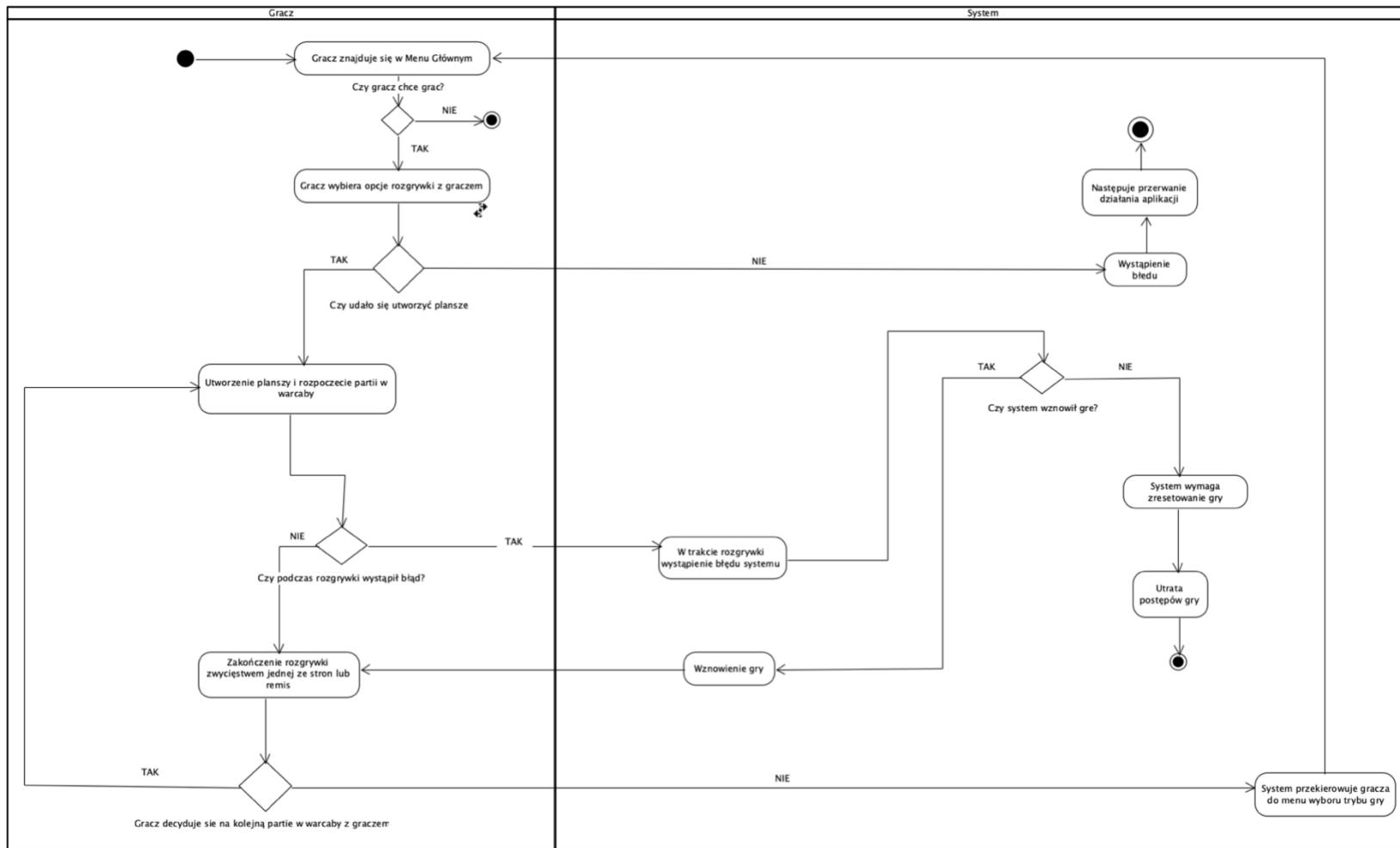


Clion - wieloplatformowe zintegrowane środowisko programistyczne języków C/C++ produkcji spółki JetBrains.

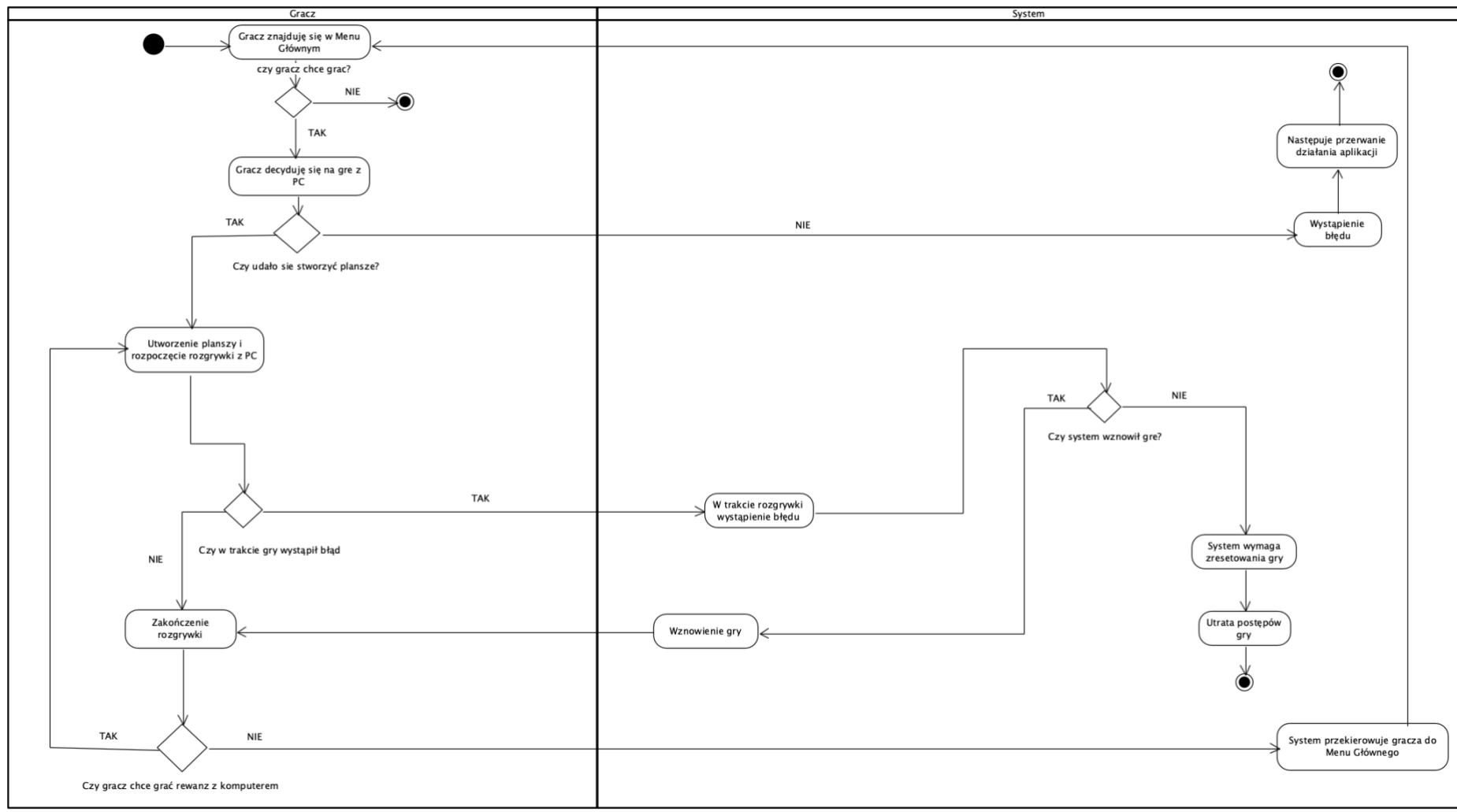


GIMP - zapewnia narzędzia potrzebne do obróbki obrazów wysokiej jakości.

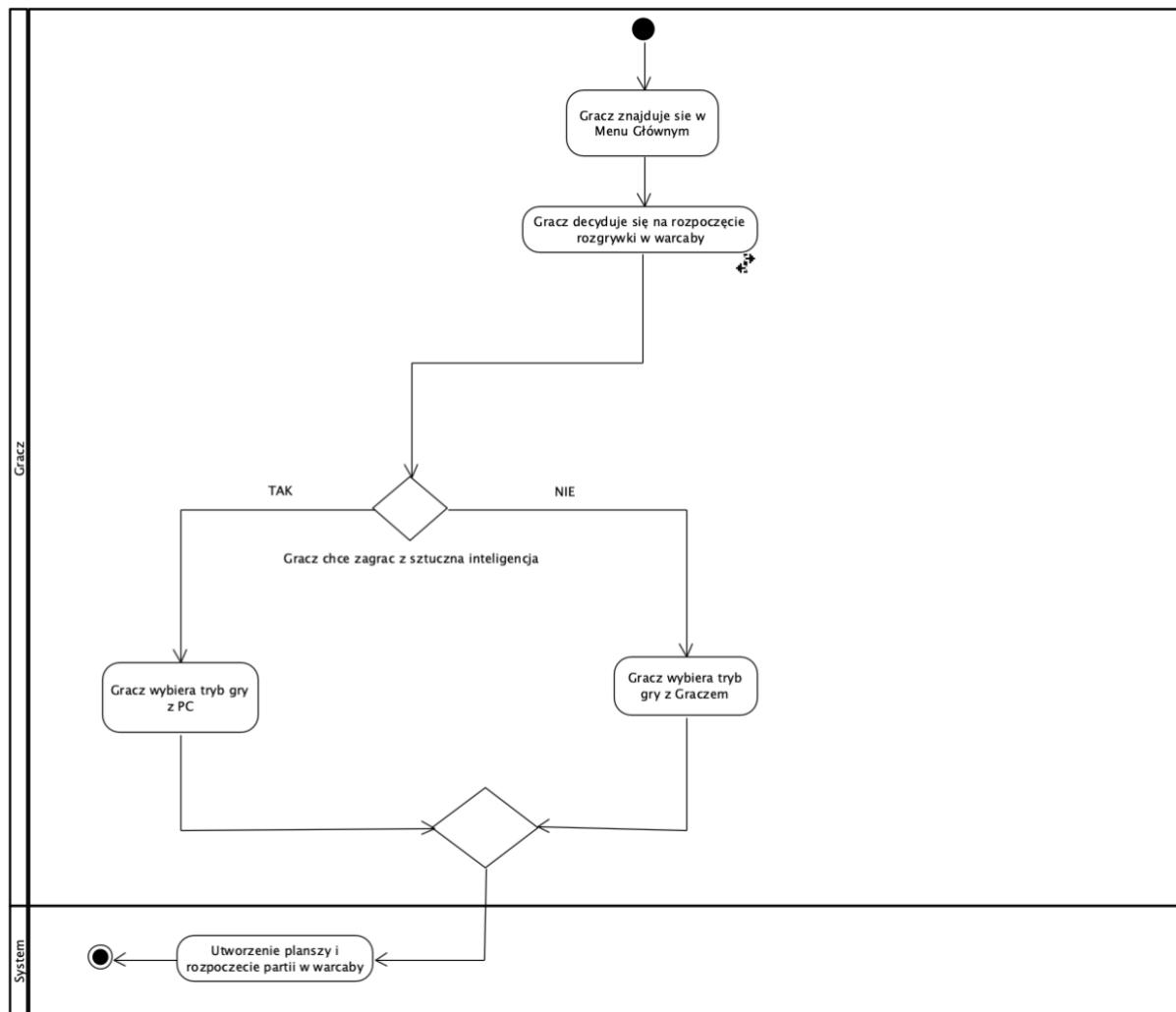
## 9. Diagram aktywności:



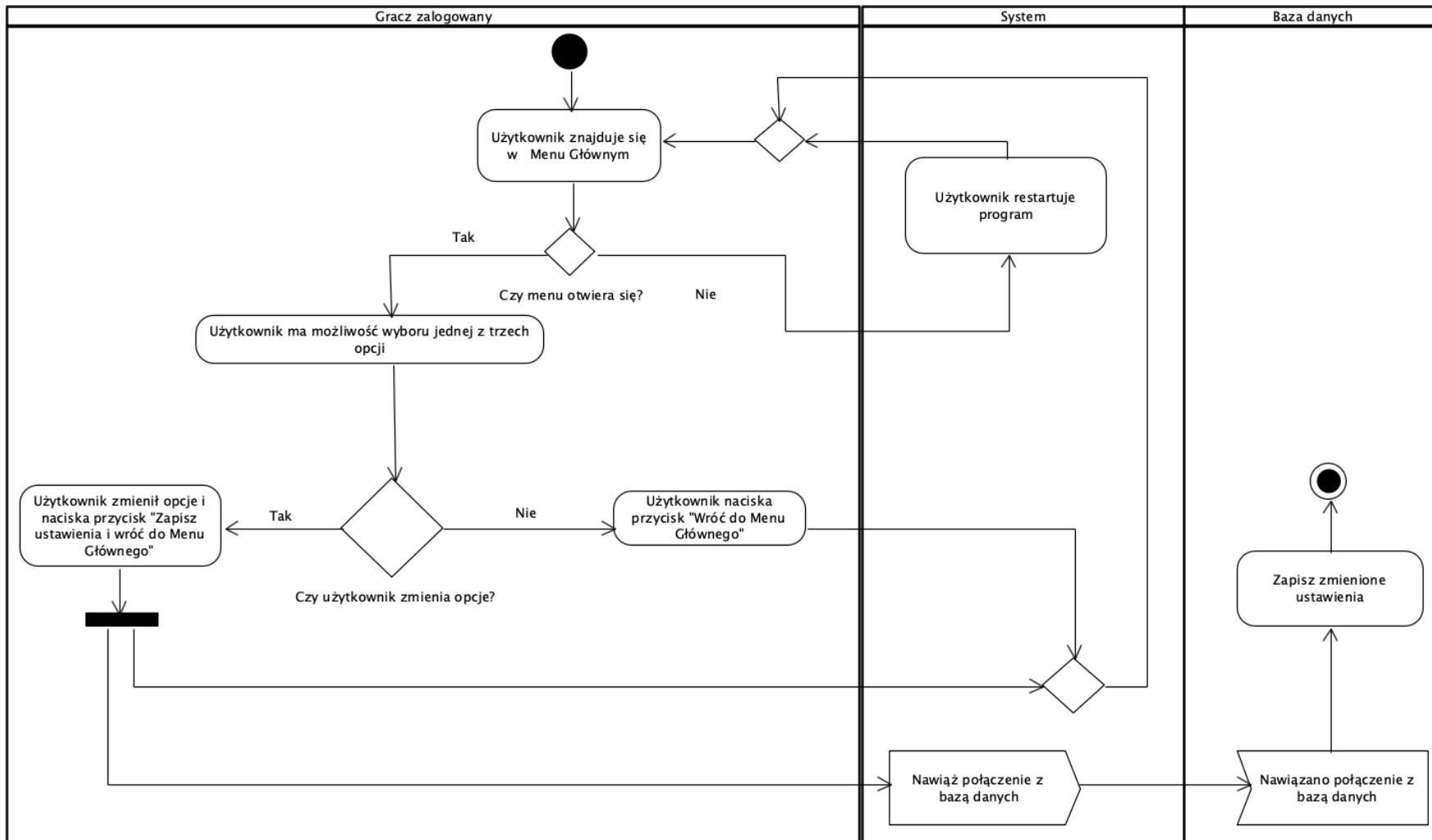
Graj z graczem



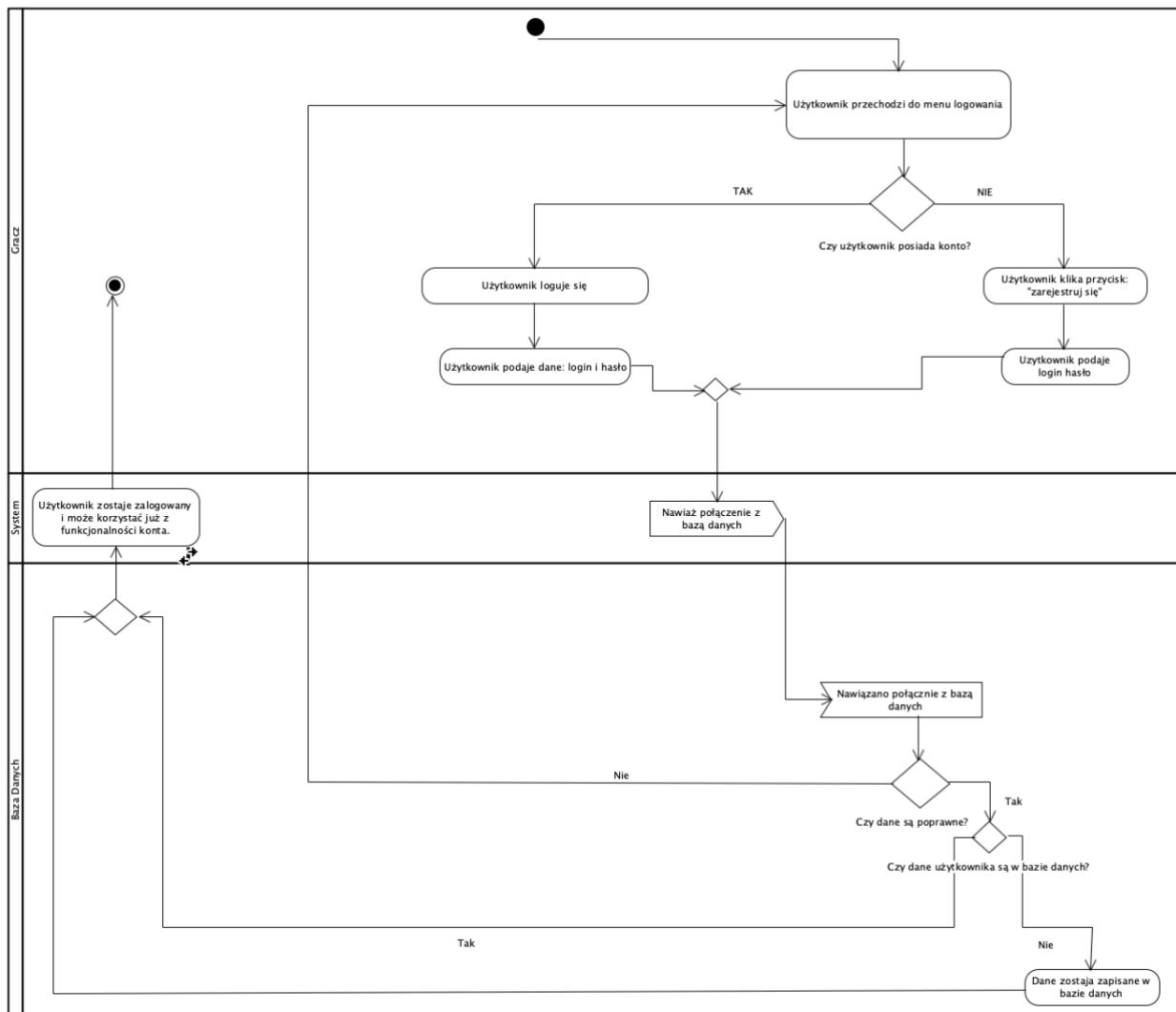
## Graj z PC



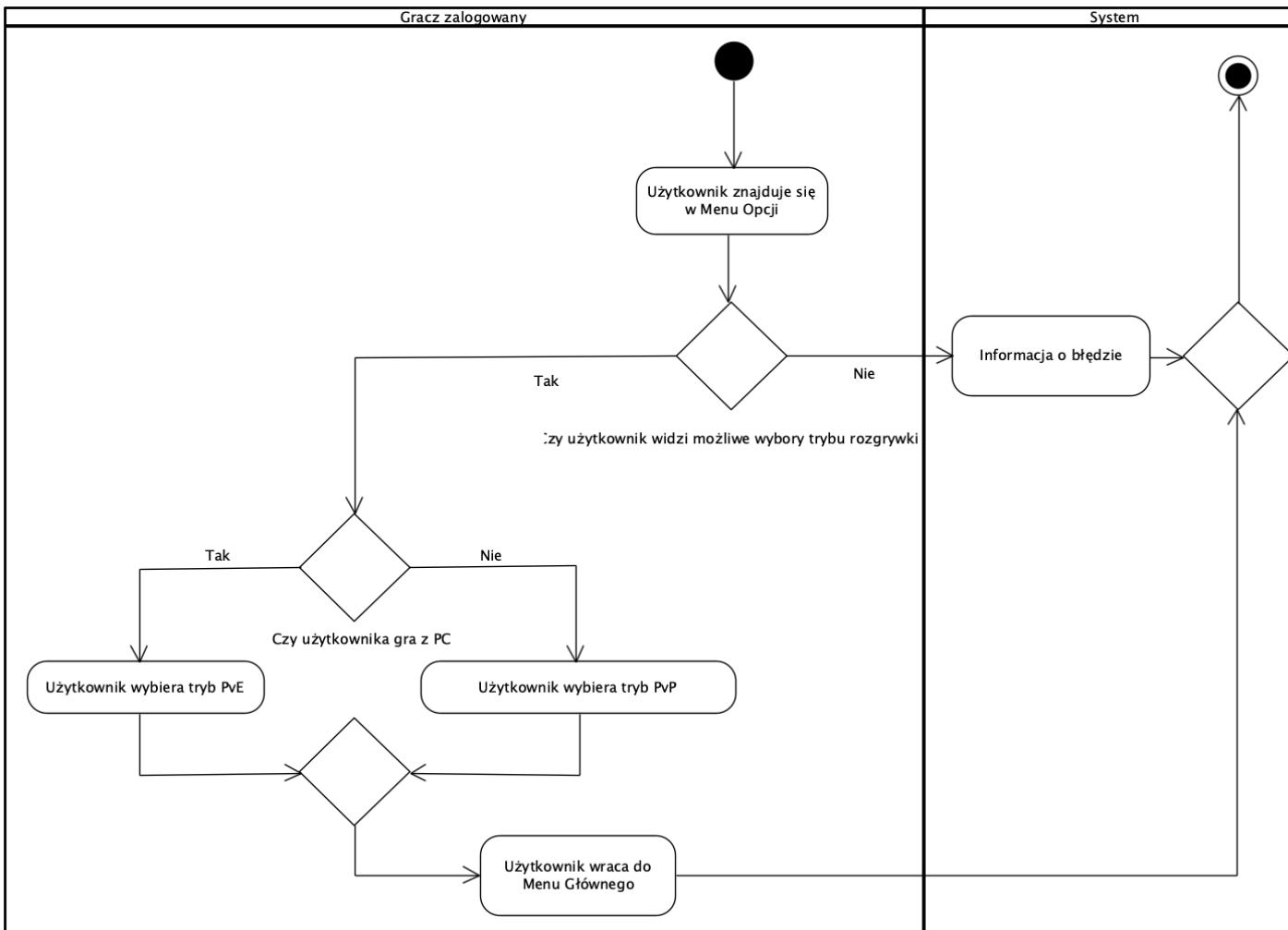
Rozpocznij grę



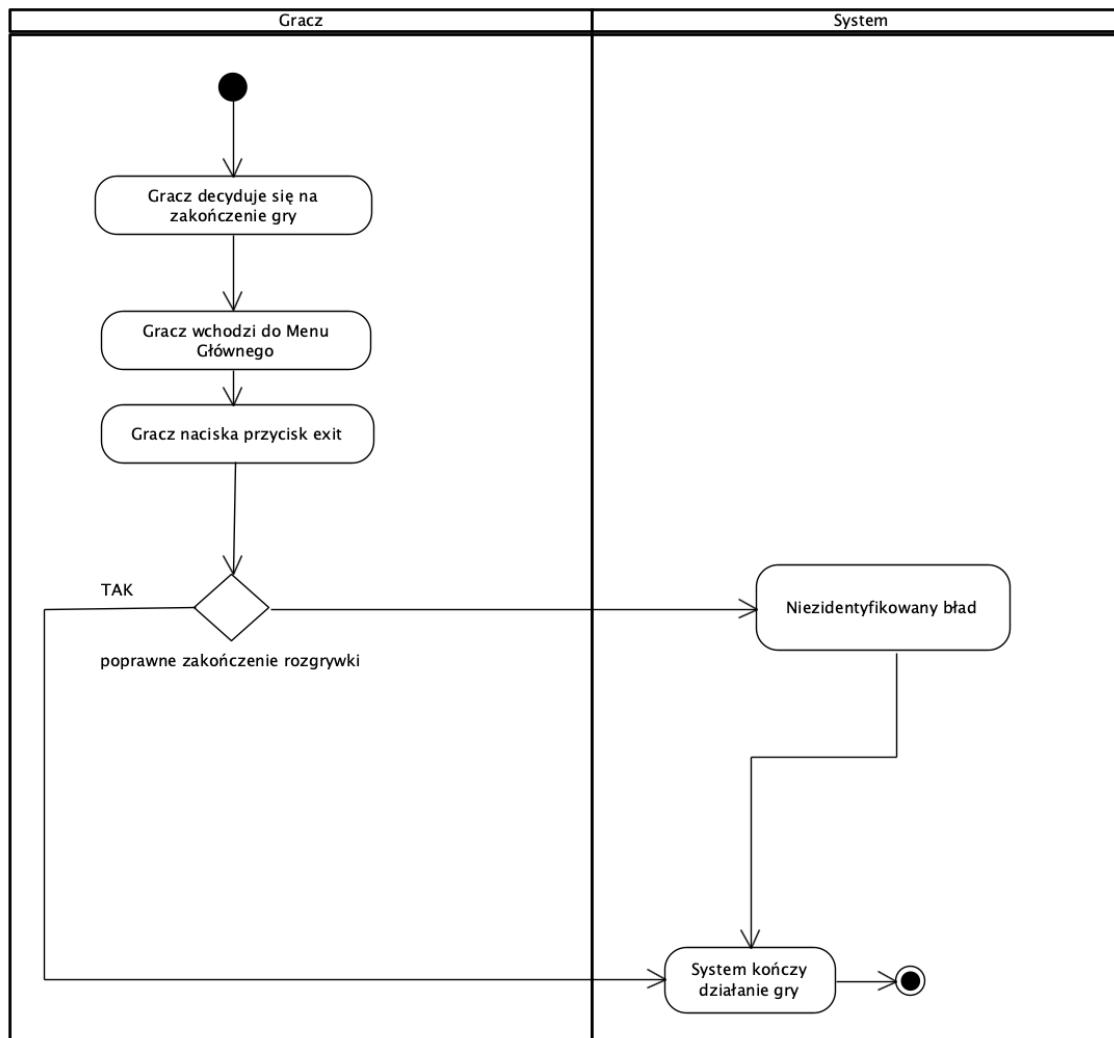
## Ustaw parametry



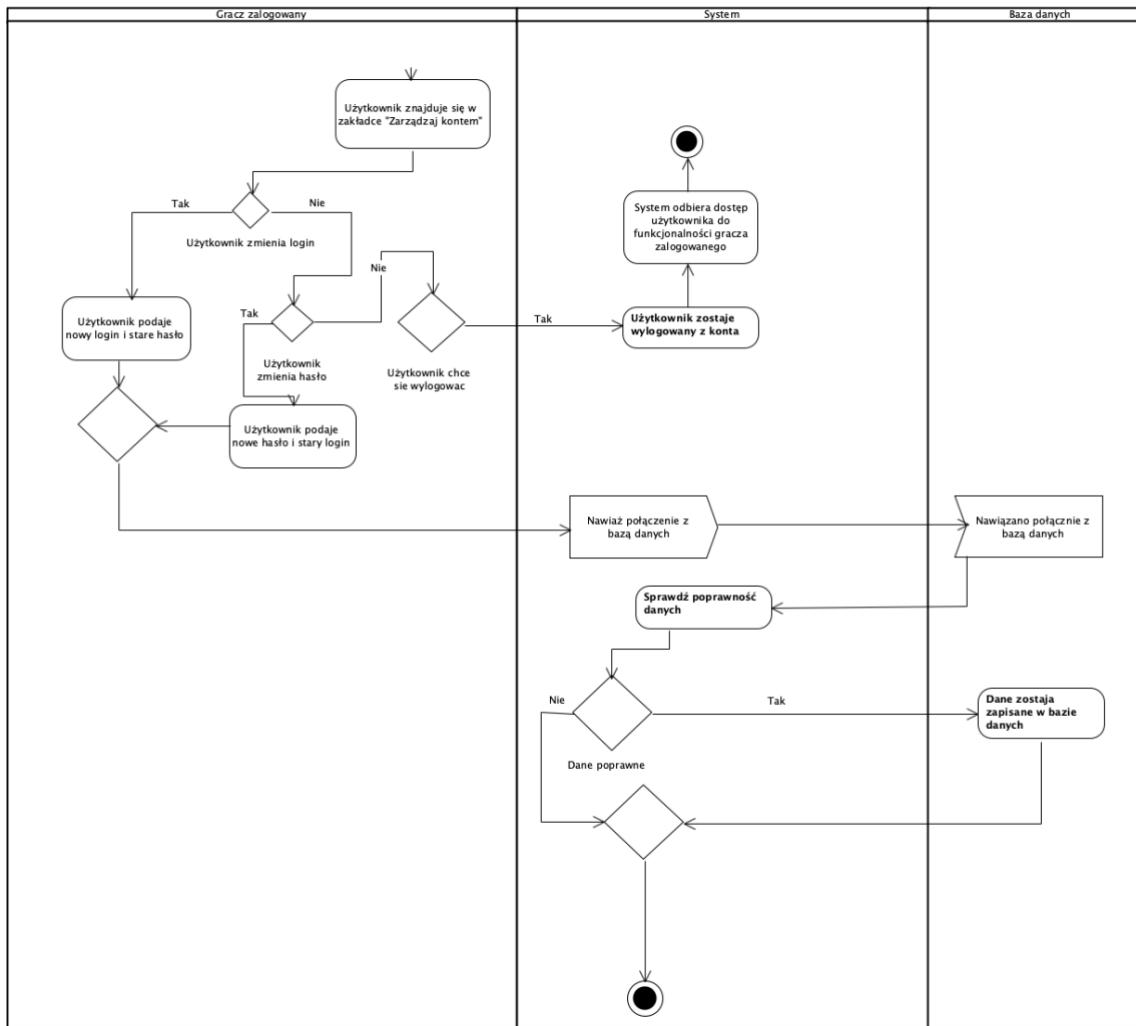
## Weryfikuj użytkownika



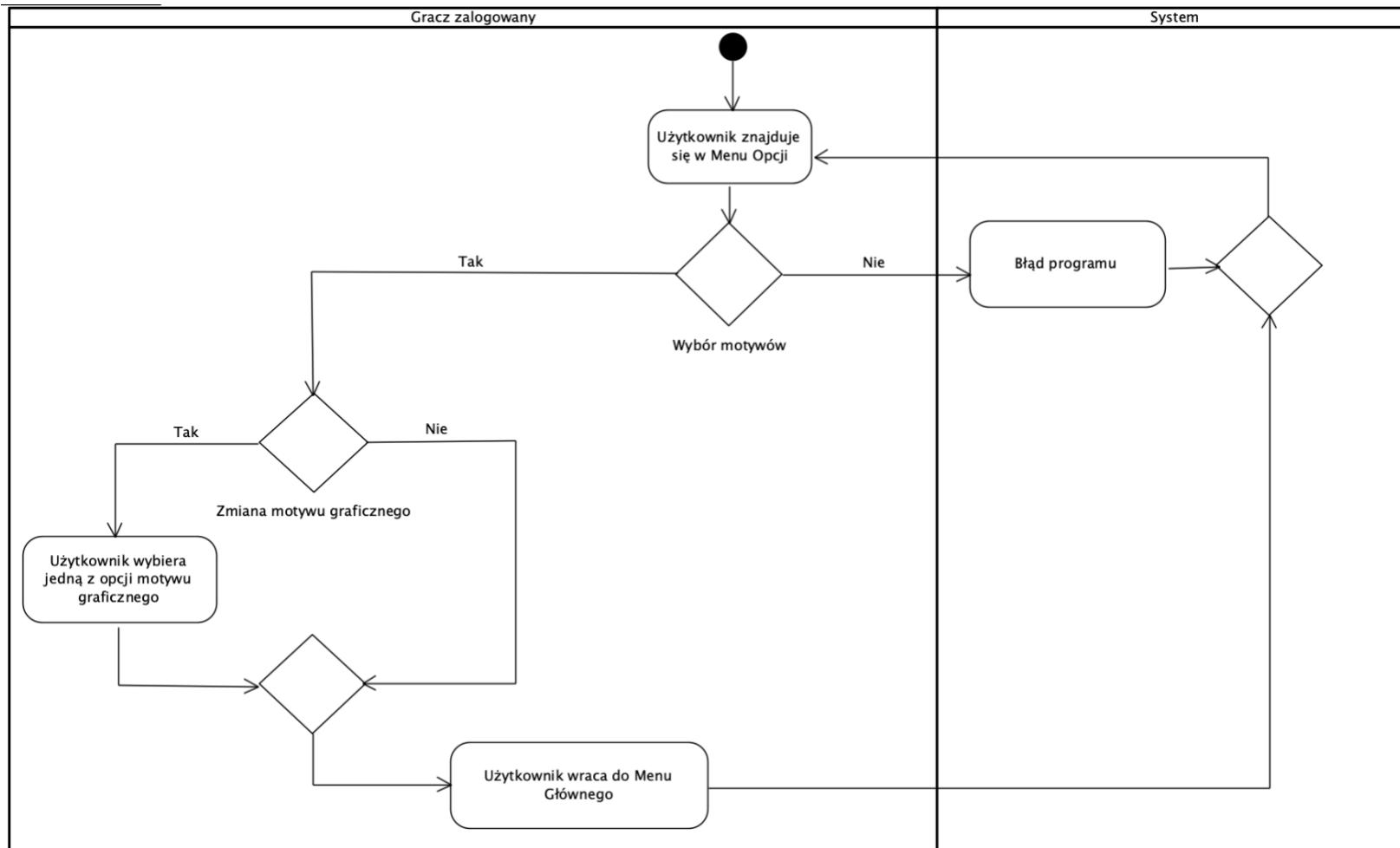
Wybierz tryb rozgrywki



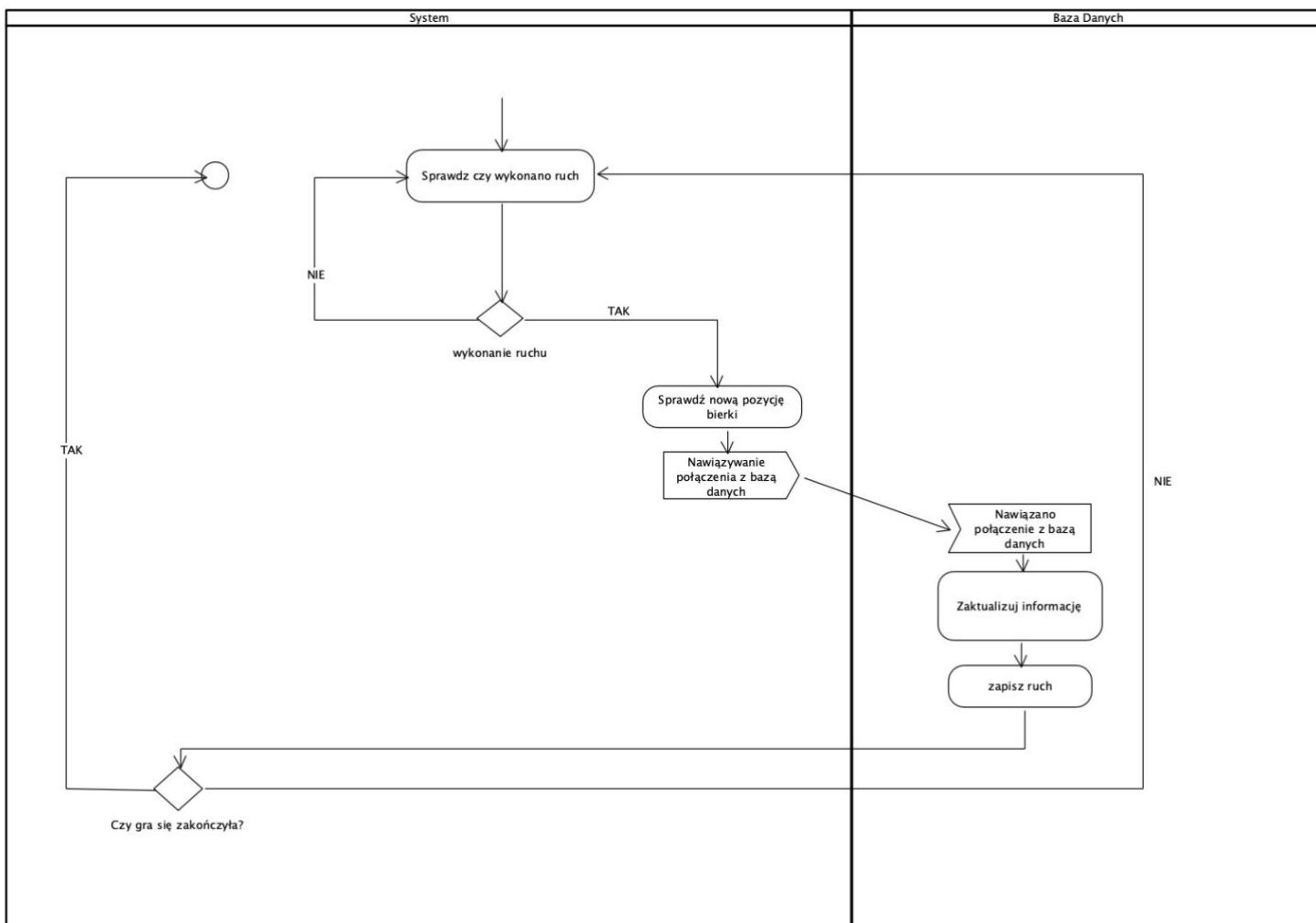
Wyjdź z gry



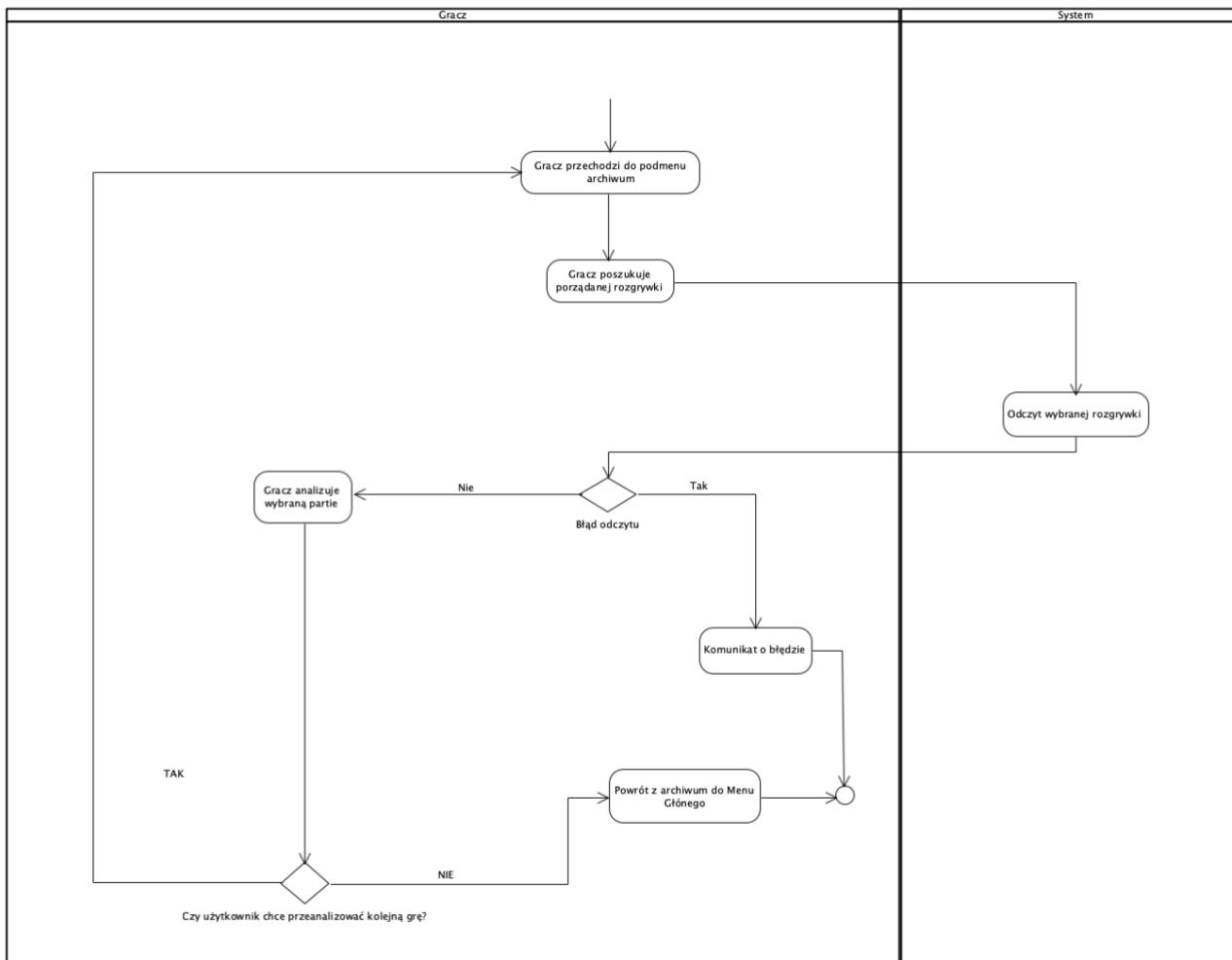
## Zarządzaj kontem



## Zmień motyw

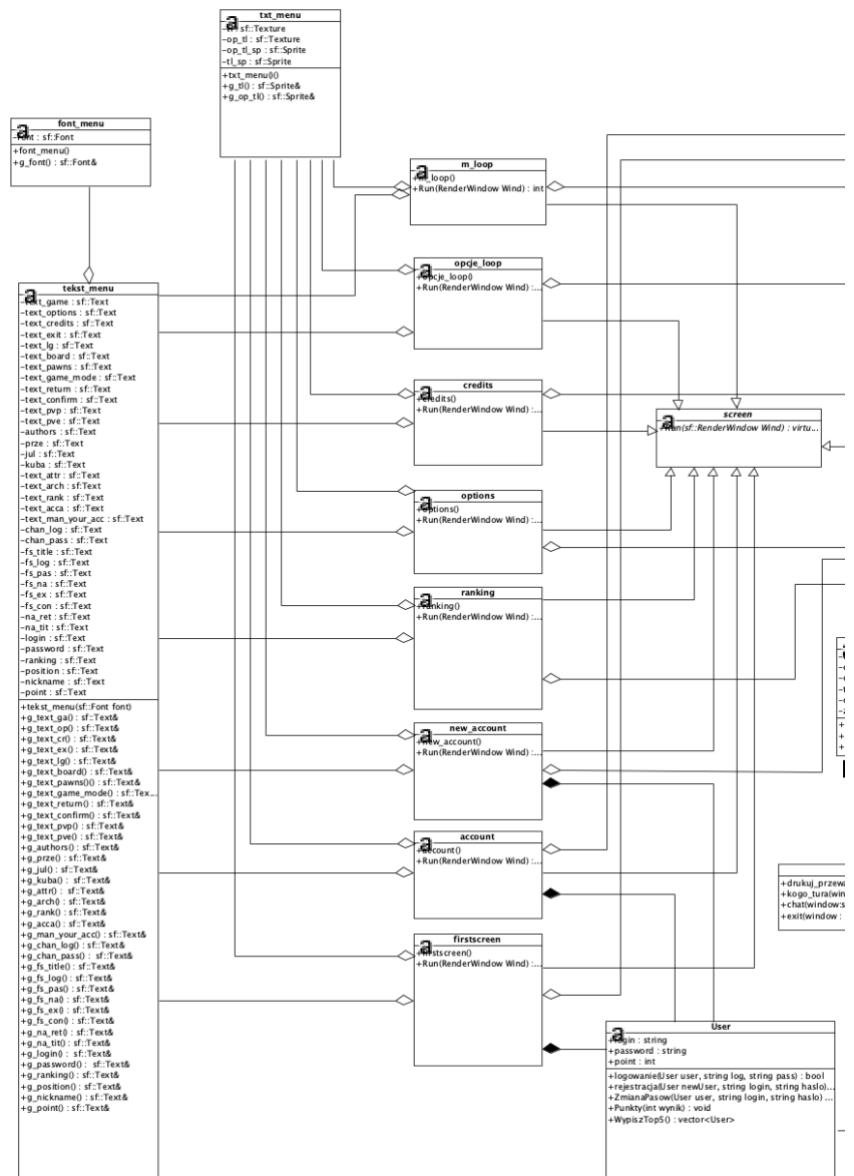


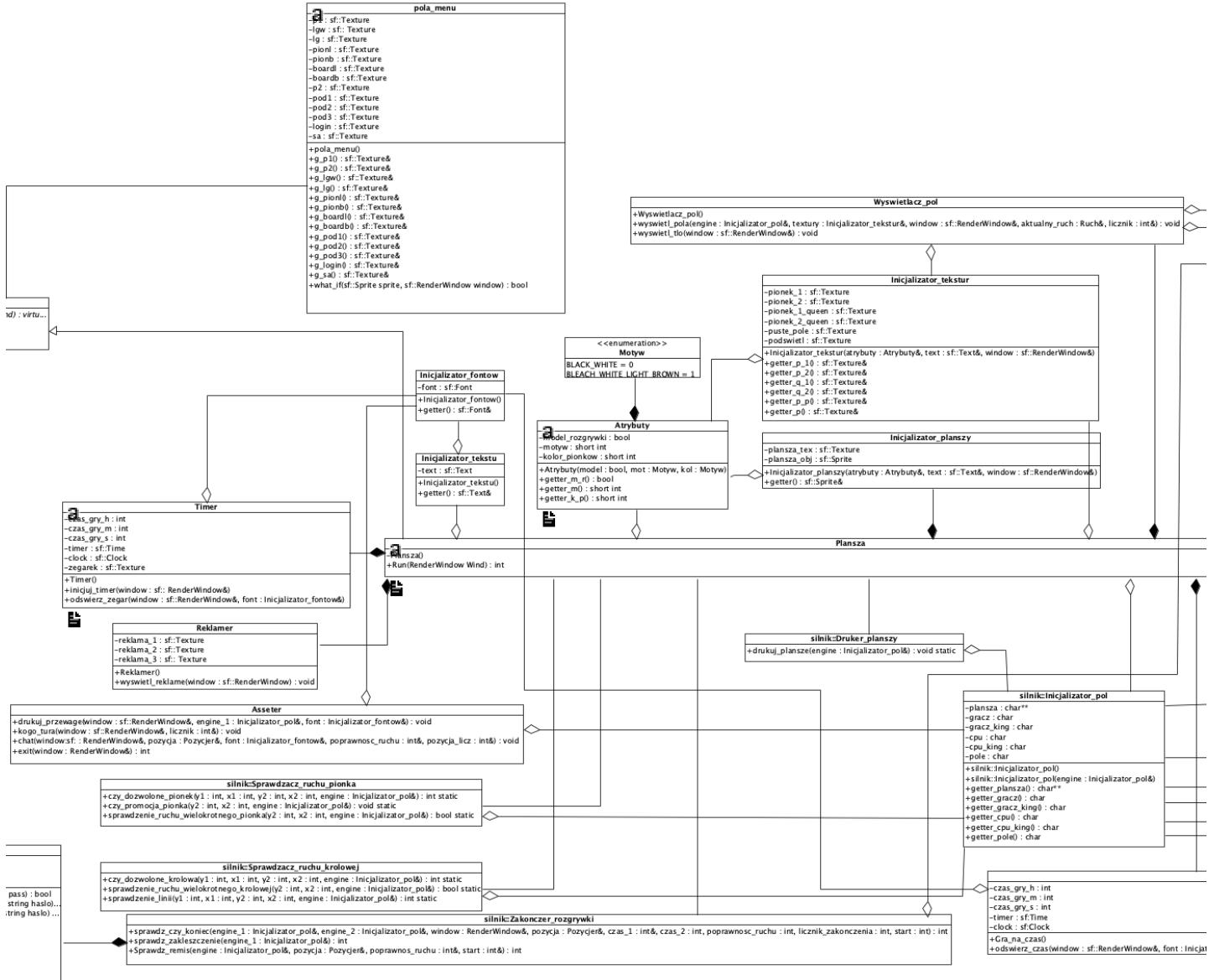
Rejestruj przebieg partii



## Przeglądaj archiwum

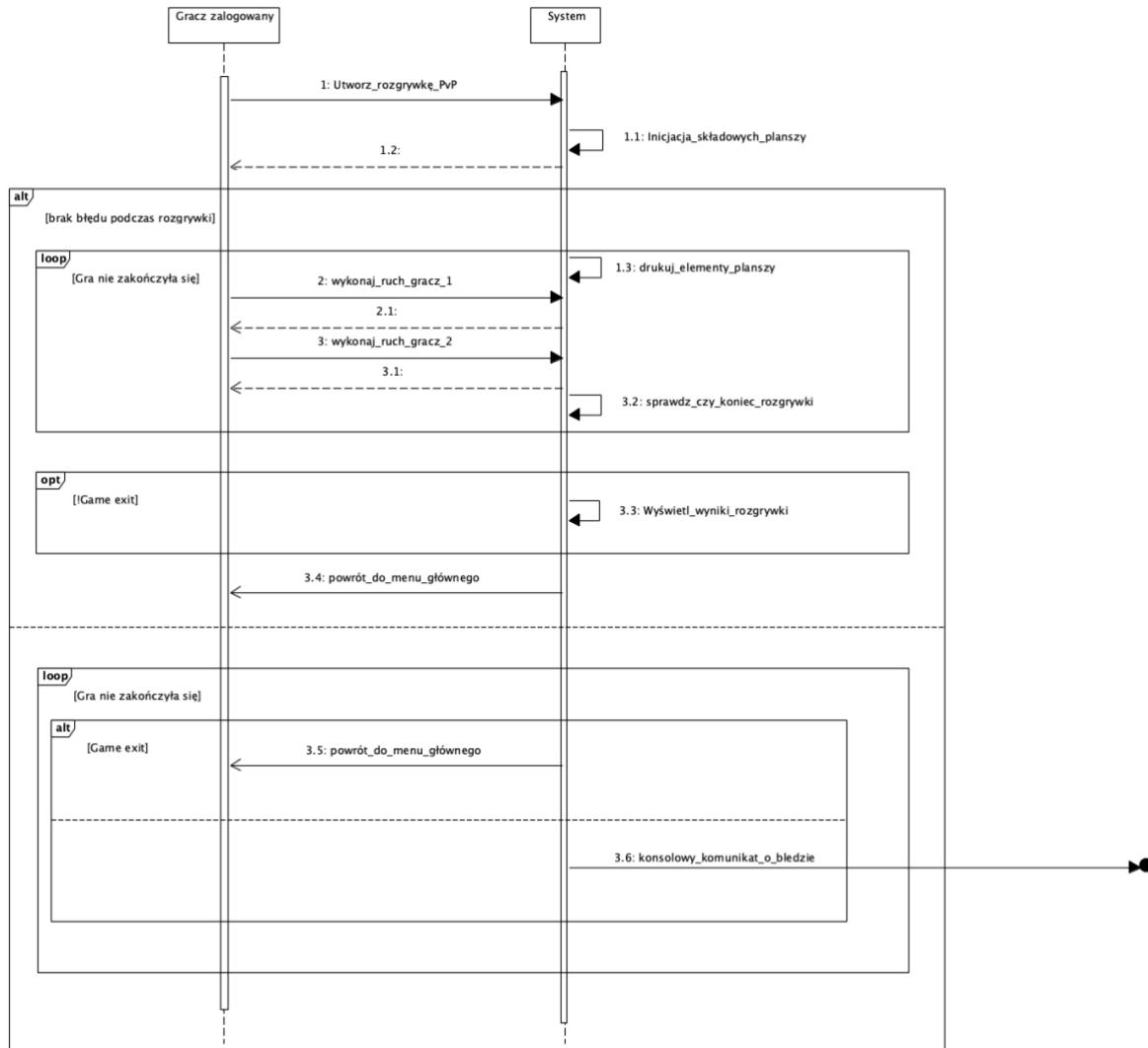
## 10. Diagram Klas:



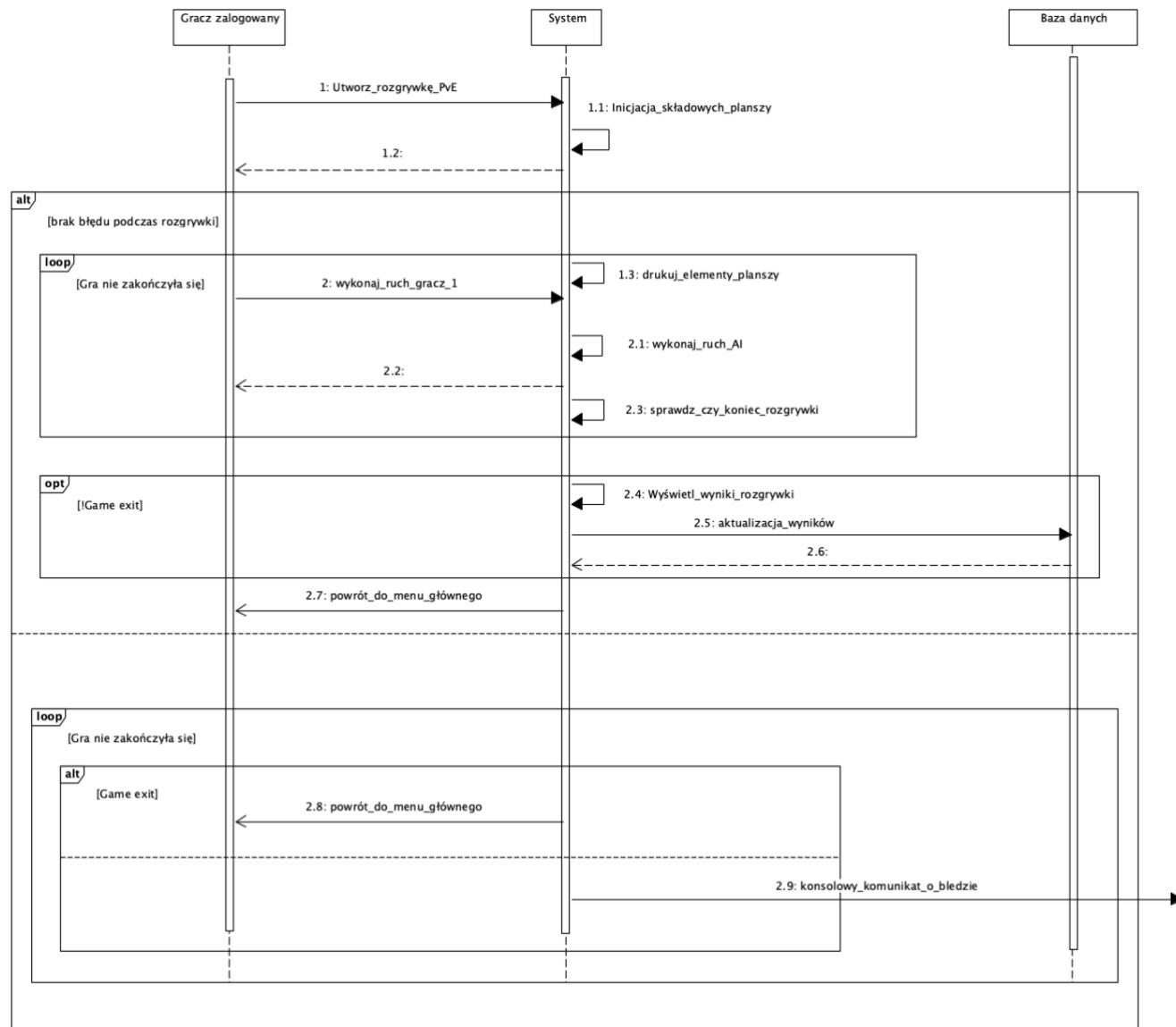




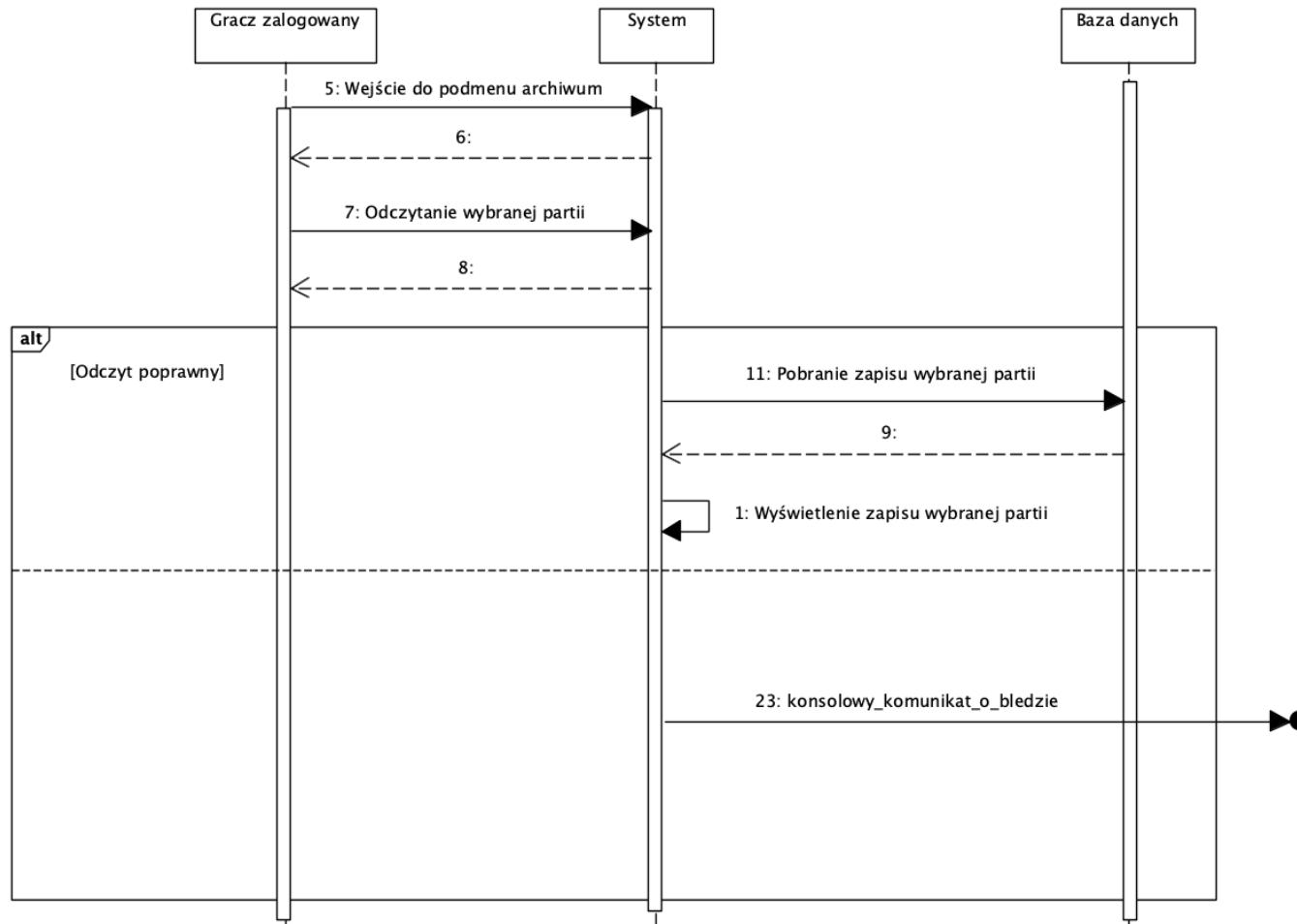
## 11. Diagram Sekwencji:



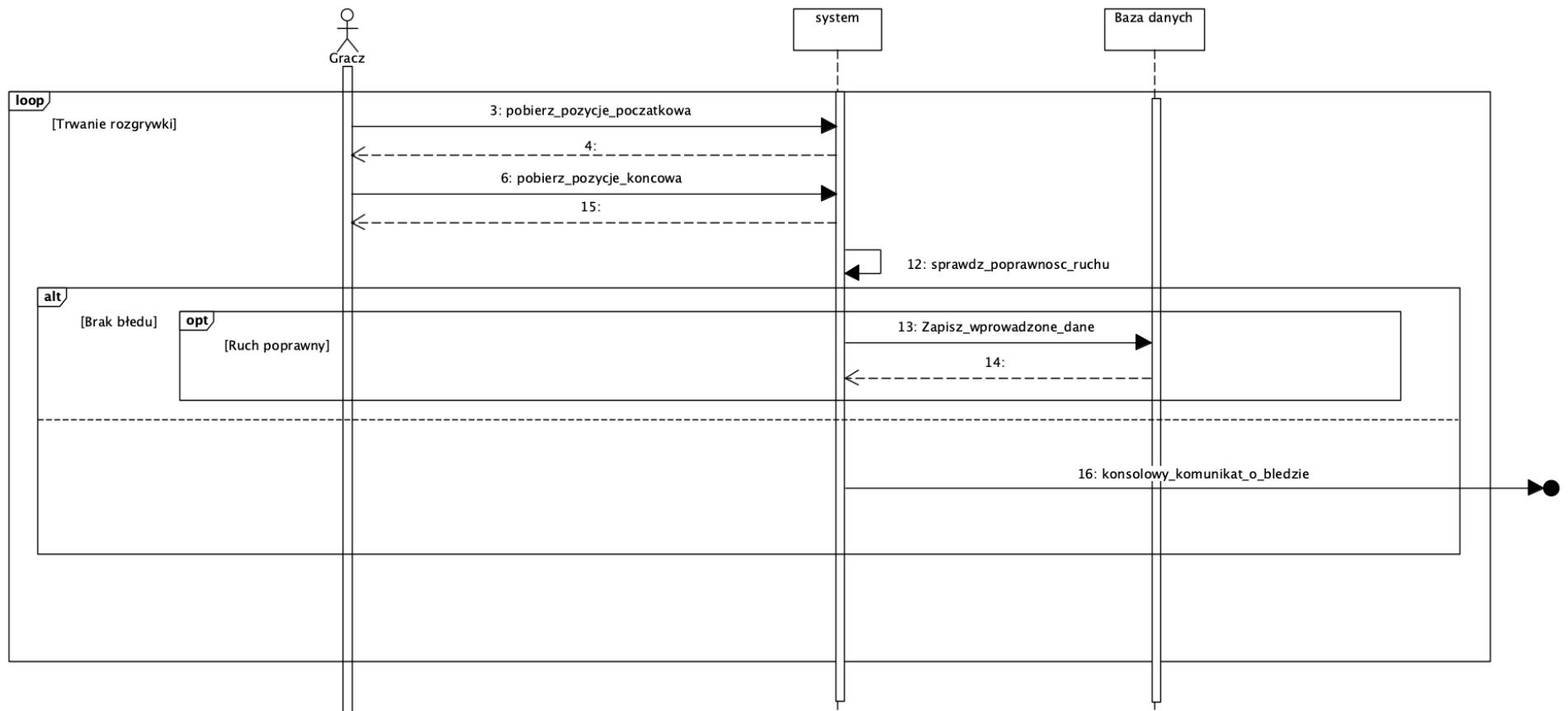
Graj z graczem



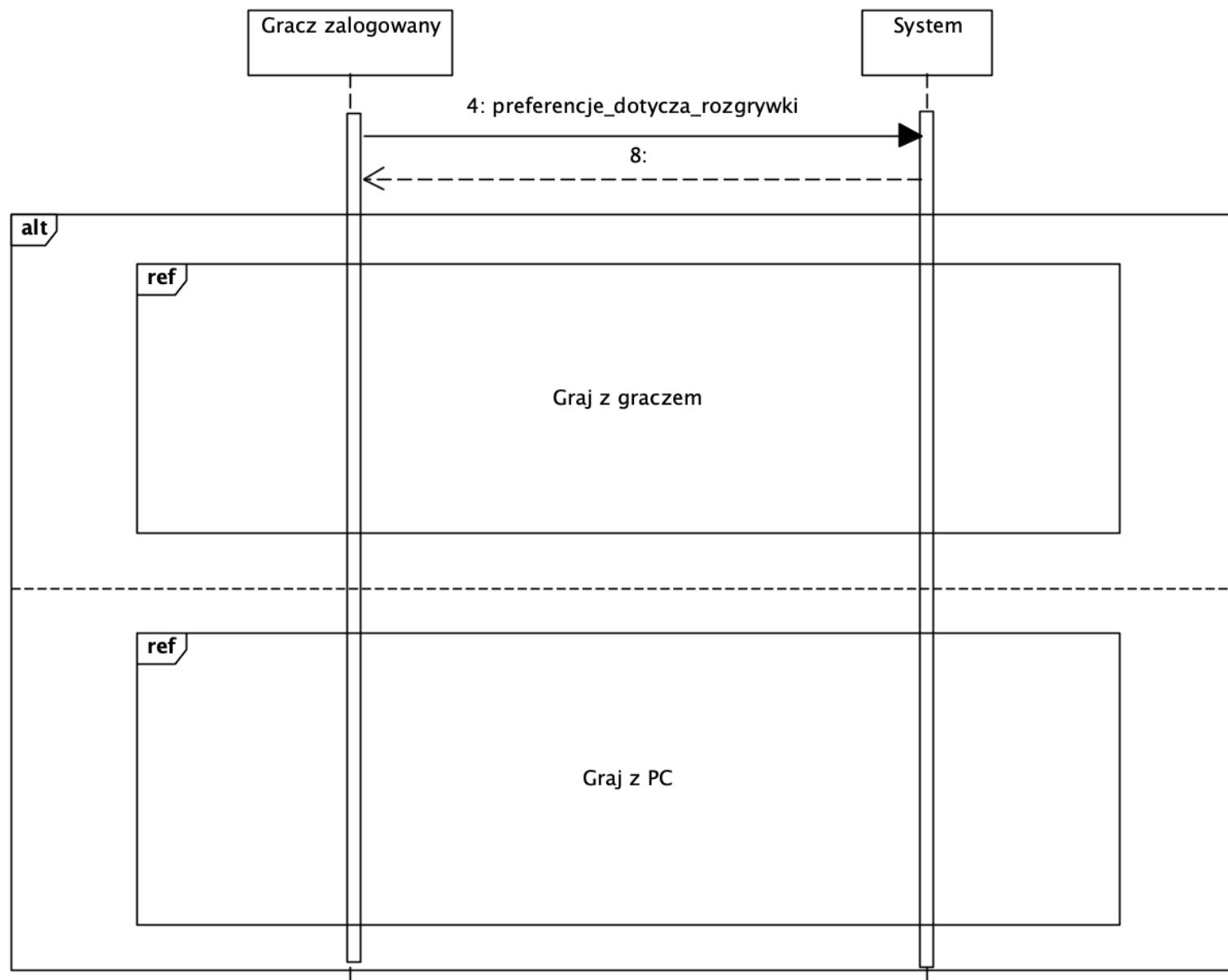
Graj z PC



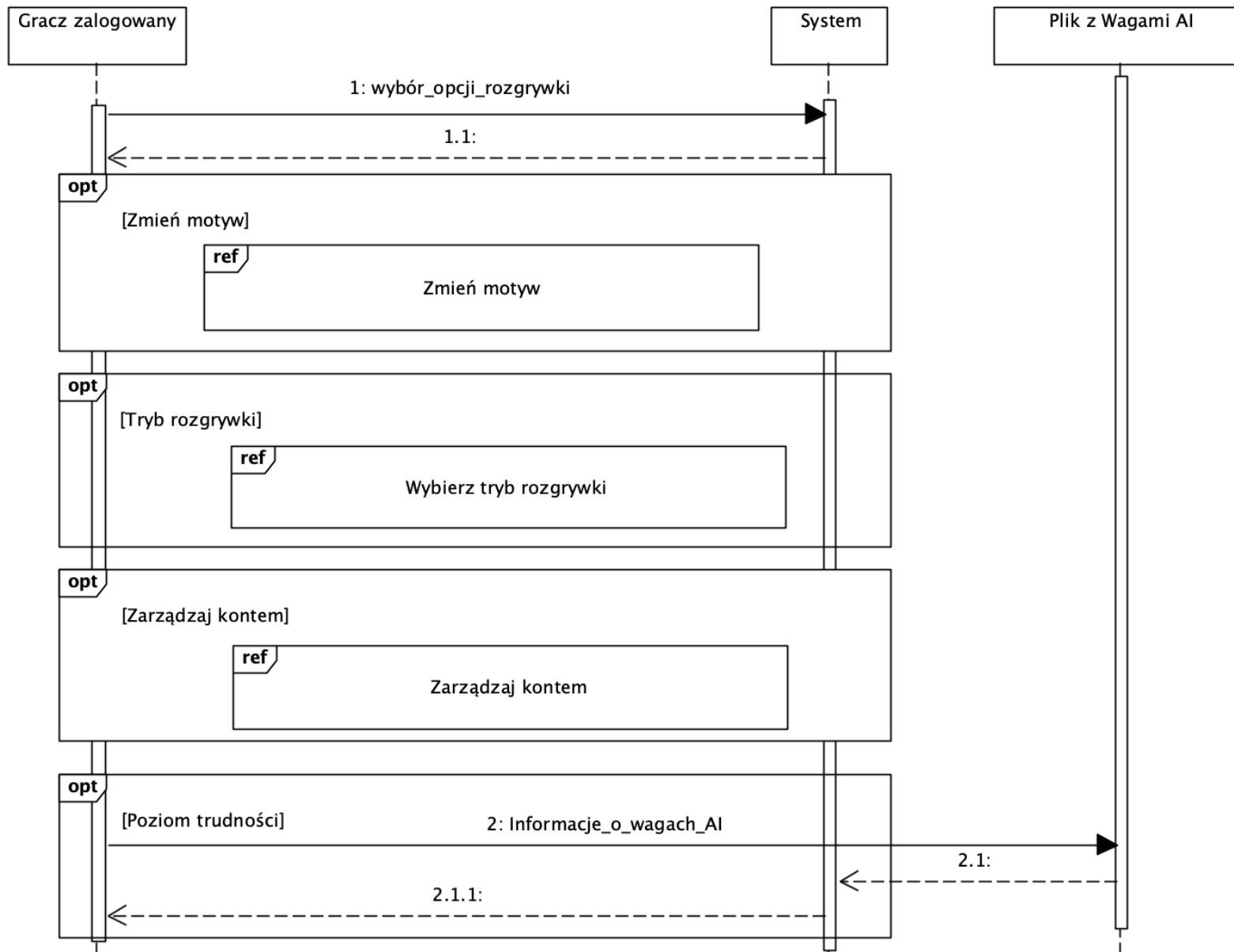
Przeglądaj Archiwum



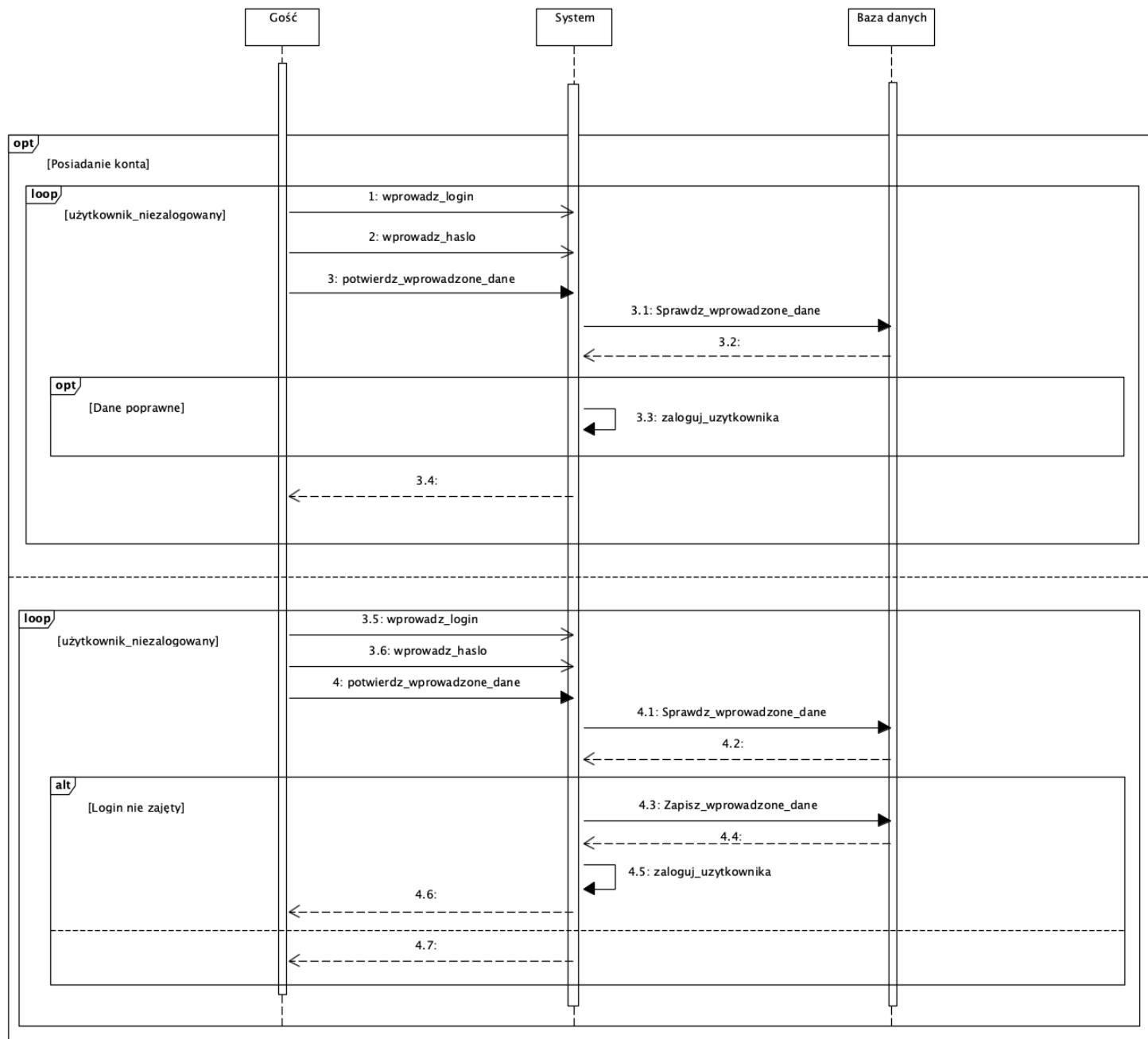
Rejestruj przebieg partii



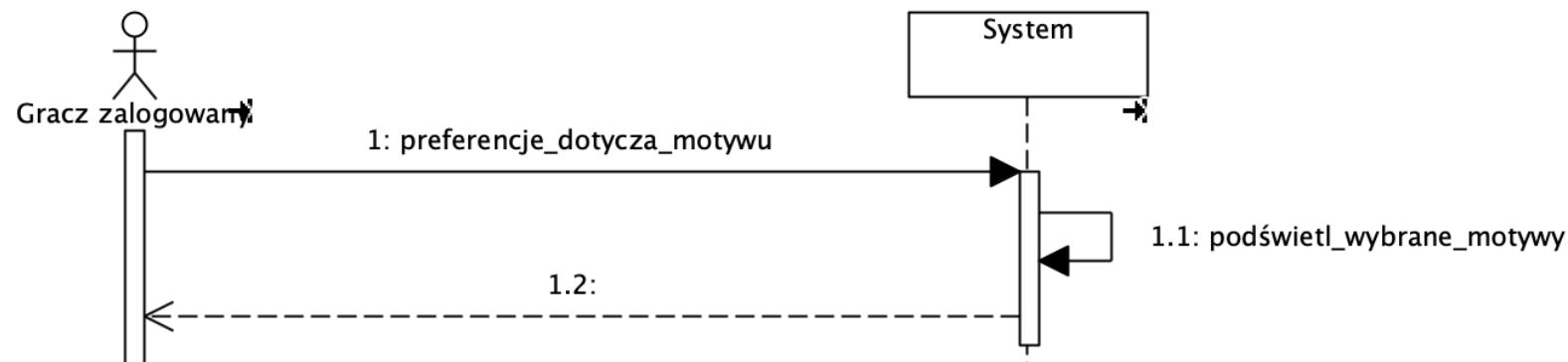
Rozpocznij grę



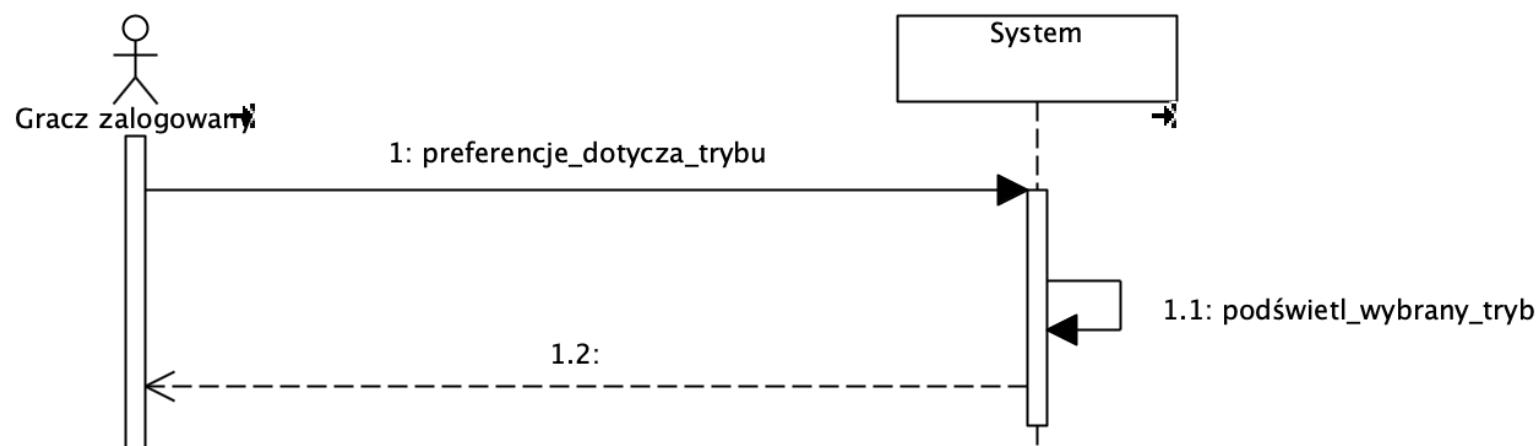
Ustaw parametry



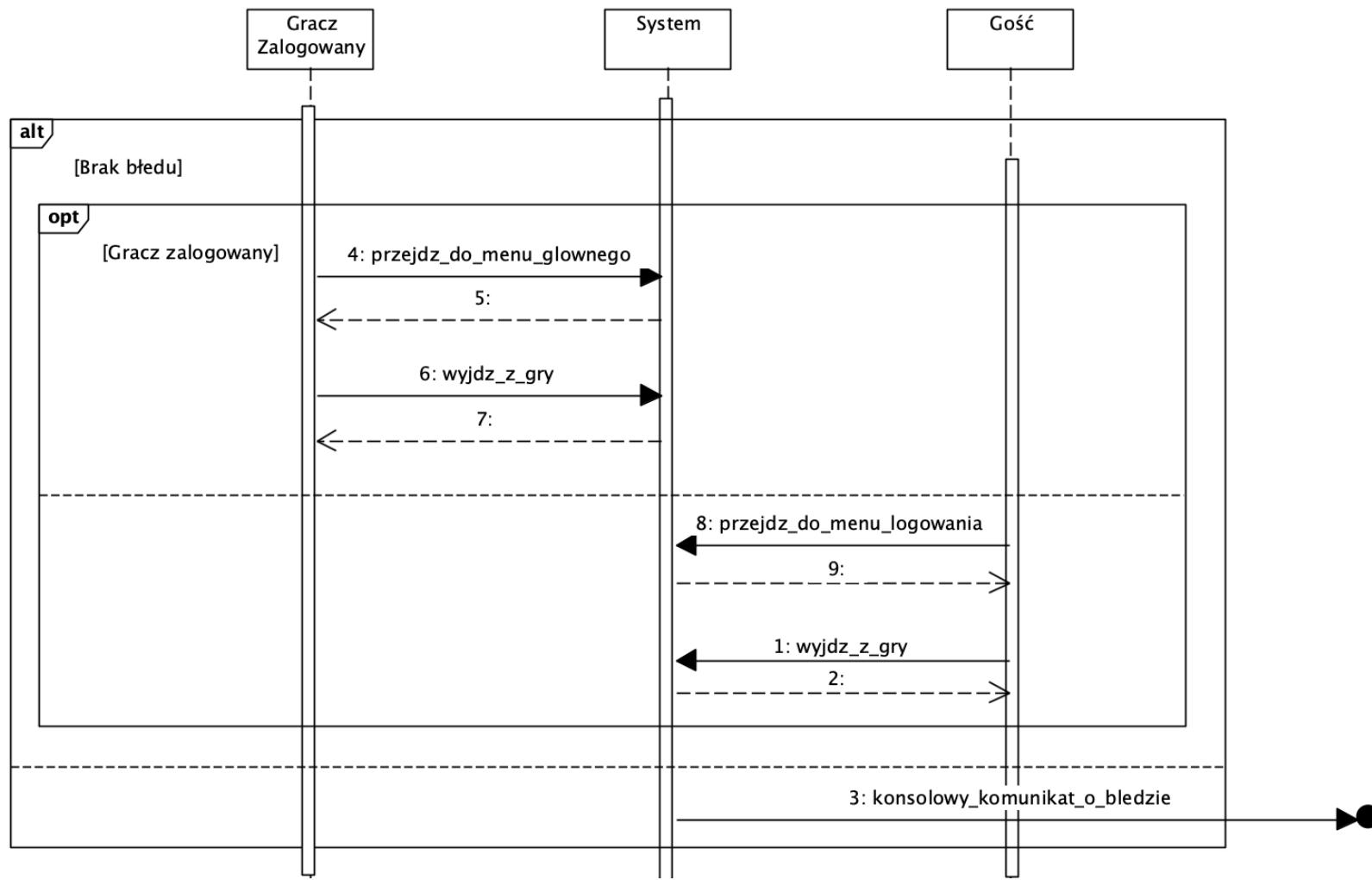
Weryfikuj użytkownika



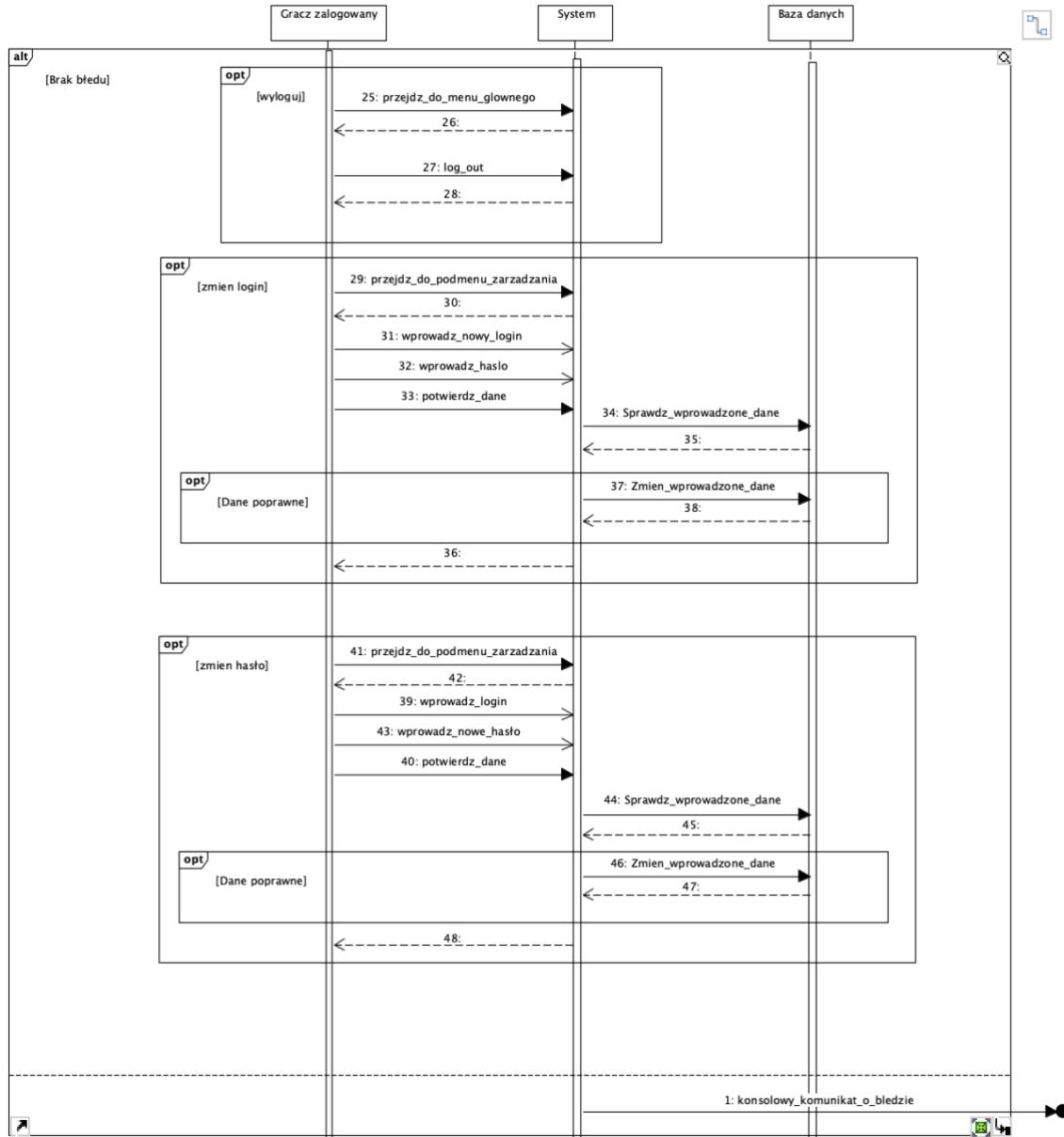
### Zmień motyw



### Wybierz tryb rozgrywki

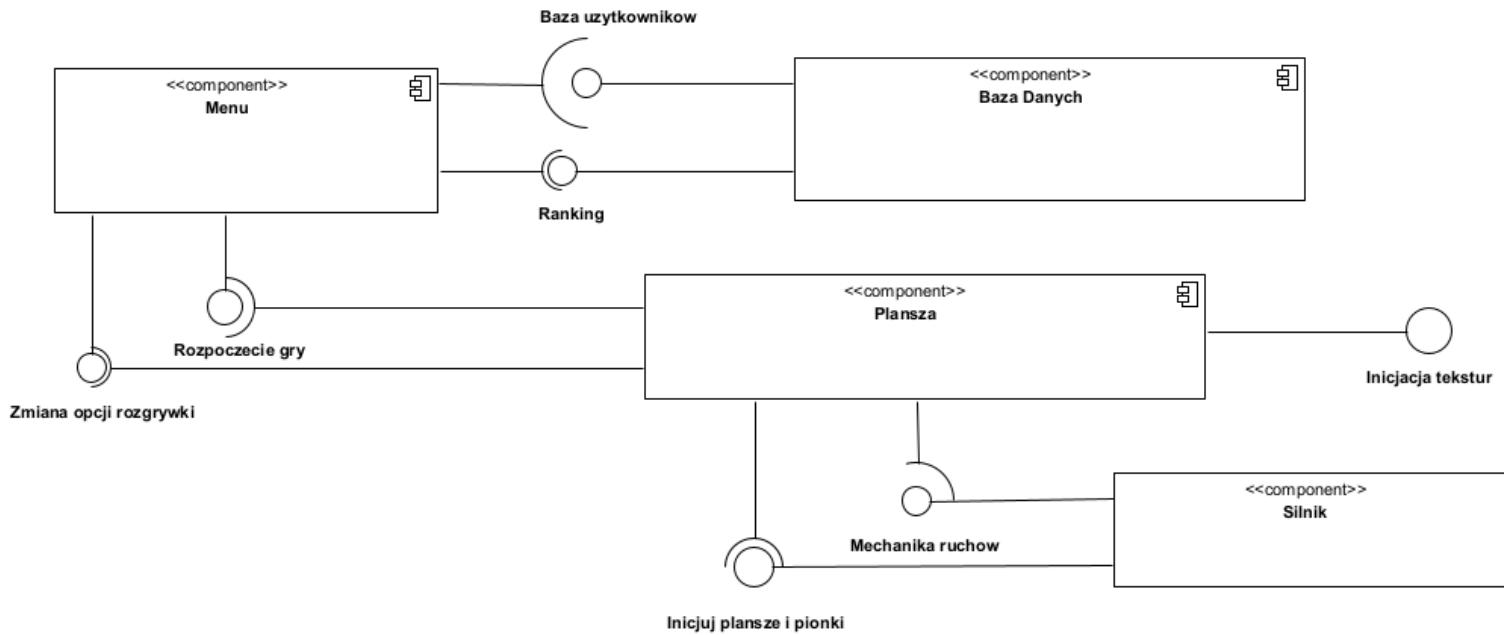


Wyjdź z gry



Zarządzaj kontem

## 12. Diagram komponentów:



**Komponenty:**  
-Silnik  
-Baza danych  
-Plansza  
-Menu

Diagram komponentów służy do ukazania organizacji pomiędzy komponentami. Prezentuje system na wyższym poziomie abstrakcji niż diagram klas. Ukazuje zależności między głównymi częściami systemu, który komponent wymaga jakiego interfejsu, dostarczanego przez inny.

## 13. Diagram wdrożeń

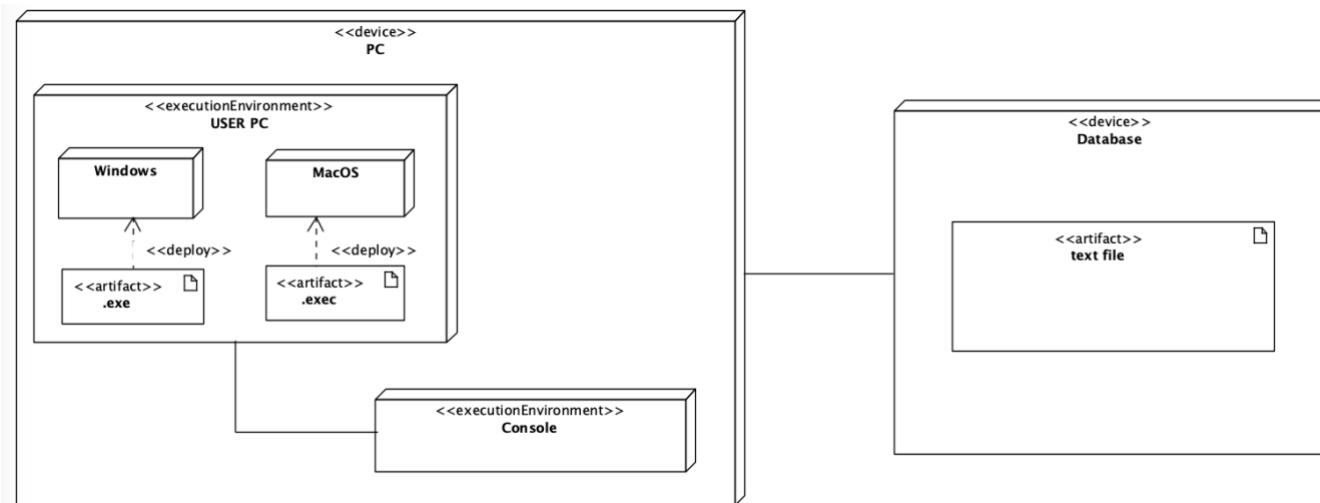


Diagram wdrożeń

Nasza aplikacja jest aplikacją desktopową – przeznaczoną dla komputerów PC. Jej uruchomienie możliwe jest zarówno w środowisku systemu operacyjnego Microsoft Windows jak i MacOS. Z tych powodów konieczne było wygenerowanie dwóch plików wykonywalnych – które następnie poprzez konsole, będzie można uruchomić na urządzeniu.



Plik wykonywalny MS Windows



Plik wykonywalny MacOS (UNIX)



Aplikacja po wdrożeniu – gotowa do użytku

Dzięki zastosowaniu komunikacji pomiędzy plikiem tekstowym, a programem możemy zbierać niezbędne dane potrzebne do realizacji licznych funkcjonalności, które zostały zaimplementowane przez nasz zespół.

## 14. Przykłady współpracy zespołu:

The screenshot shows the main Trello board for the 'Checkers' project. It features four columns: 'TO DO', 'IN PROGRESS', 'ARCHIVE - DONE', and 'OTHERS'. The 'TO DO' column contains tasks related to gathering information about game progress and MySQL database integration. The 'IN PROGRESS' column includes tasks for optimizing and commenting on the game logic. The 'ARCHIVE - DONE' column lists completed tasks like menu optimization and debouncing. The 'OTHERS' column contains links to repositories and disk drives.

Główna tablica Trello

This screenshot shows a Trello board titled 'Checkers - documentation (improvement)'. It has four columns: 'TO DO', 'IN PROGRESS', 'ARCHIVE - DONE', and 'REFERENCE MANUAL'. The 'TO DO' column lists tasks such as creating class diagrams and component diagrams. The 'IN PROGRESS' column includes tasks for creating a glossary and sequence diagrams. The 'ARCHIVE - DONE' column contains completed tasks like creating a title page and a table of contents. The 'REFERENCE MANUAL' column provides a link to the full documentation version.

Tablica poprawy dokumentacji

Wersje projektu :

Rev.	Uczestnik	Czas Daty
170	juliaklu	2022 Nov 26, 19:56:16
169	juliaklu	2022 Nov 26, 19:54:38
168	Przemekjan	2022 Nov 24, 15:44:59
167	Przemekjan	2022 Nov 24, 15:41:05
166	Przemekjan	2022 Nov 24, 15:34:10
165	Przemekjan	2022 Nov 20, 17:15:14
164	JKL	2022 Nov 17, 16:30:04
163	juliaklu	2022 Nov 17, 16:26:13
162	Przemekjan	2022 Nov 17, 16:25:50
161	Przemekjan	2022 Nov 16, 22:23:56
160	juliaklu	2022 Nov 16, 20:51:29
159	juliaklu	2022 Nov 16, 20:33:55
158	Przemekjan	2022 Nov 15, 14:02:39
157	Przemekjan	2022 Nov 15, 14:00:13
156	Przemekjan	2022 Nov 15, 11:45:13
155	Przemekjan	2022 Nov 15, 11:02:45
154	JKL	2022 Nov 13, 19:19:47
153	JKL	2022 Nov 13, 19:19:33
152	Przemekjan	2022 Nov 12, 23:42:42
151	juliaklu	2022 Nov 12, 19:31:39
150	juliaklu	2022 Nov 12, 19:19:24
149	juliaklu	2022 Nov 12, 17:22:17
148	Przemekjan	2022 Nov 12, 13:13:17
147	Przemekjan	2022 Nov 11, 22:49:18

Project log – Visual Paradigm

- o Commits on Jan 7, 2023
  - WarcabyKuba**  
 **JKEniu** committed yesterday
- o Commits on Jan 7, 2023
  - Add files via upload**  
 **theworstplayer3** committed 14 hours ago
- o Commits on Dec 23, 2022
  - Delete Method\_definitions.cpp**  
 **przemek890** committed 2 weeks ago

Github repository commits

## 15. Testy jednostkowe:

### 1) Przykład:

```
//usuwanie liter jesli backspace - TEST JEDNOSTKOWY czy dziala
else if(event.text.unicode==8&&a.length()>0){
    if(event.text.unicode==8&&a.length()>0){
        a.pop_back();
        cout<<"Test zakonczony pomyslnie"=><endl;
    }
    else cout<<"Test zakonczony niepomyślnie"=><endl;
}
```

Test jednostkowy sprawdza czy przy naciśnięciu klawisza BACKSPACE, podczas wprowadzania loginu przy logowaniu, w programie usuwany jest ostatni znak w stringu.

Jeżeli ostatni znak jest usunięta konsola wyświetla informacje o teście zakończonym pomyślnie. W innym przypadku konsola informuje o błędnie zakończonym teście.

Wynik testu

```
Test zakonczony pomyslnie
Test zakonczony pomyslnie
```

2) Przykład:

```
//dodawanie do stringa - TEST JEDNOTKOWY czy dziala
if(event.text_unicode!=8 && b.length()<24){
    if(event.text_unicode!=8 && b.length()<24){
        b+=event.text_unicode;
        gwiazdko+='*';
        cout<<"Test zakonczony pomyslnie"=><endl;
    }
    else cout<<"Test zakonczony niepomyślnie"=><endl;
}
```

Wynik testu

```
Test zakonczony pomyslnie
```

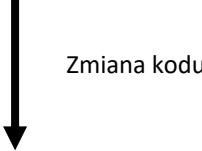
Test jednostkowy sprawdza czy przy naciśnięciu dowolnego klawisza na klawiaturze (oprócz BACKSPACE), podczas wprowadzania hasła, w programie dodany zostanie znak do stringa, który tworzy hasło, jak i do stringa, który odpowiada za ilość wypisywanych gwiazdek na ekran.

Jeżeli znak zostanie dodany do obu stringów konsola wyświetla informacje o teście zakończonym pomyślnie. W innym przypadku konsola informuje o błędnie zakończonym teście.

## 16. Refaktoryzacja kodu:

### 1) Przykład:

```
/// --> silnik::Zakonczer_rozgrywki:  
void silnik::Zakonczer_rozgrywki::sprawdz_czy_koniec(Inicjalizator_pol& engine_1, Inicjalizator_pol& engine_2, sf::RenderWindow& window) {  
    int licznik_gracz_1 = 0;  
    int licznik_gracz_2 = 0;  
    for(int i=0;i<8;i++) {  
        for (int j = 0; j < 8; j++) {  
            if (engine_1.getter_plansza()[i][j] == 120 || engine_1.getter_plansza()[i][j] == 88) licznik_gracz_1++;  
            else if (engine_2.getter_plansza()[i][j] == 111 || engine_2.getter_plansza()[i][j] == 79) licznik_gracz_2++;  
        }  
    }  
  
    int zakleszczenie_1 = silnik::Zakonczer_rozgrywki::sprawdz_zakleszczenie(engine_1);  
    int zakleszczenie_2 = silnik::Zakonczer_rozgrywki::sprawdz_zakleszczenie( & engine_2);  
  
    if(licznik_gracz_1 == 0 || licznik_gracz_2 == 0 || zakleszczenie_1 == 0 || zakleszczenie_2 == 0 ) {  
        int static licznik;  
        sf::Sprite pole_wynikowe;  
        sf::Texture text;  
        if(licznik_gracz_1 == 0 || zakleszczenie_1 == 0) text.loadFromFile( filename: "./win/win_2.png");  
        else text.loadFromFile( filename: "./win/win_1.png");
```



```
/// --> silnik::Zakonczer_rozgrywki:  
void silnik::Zakonczer_rozgrywki::sprawdz_czy_koniec(Inicjalizator_pol& engine_1, Inicjalizator_pol& engine_2,sf::RenderWindow& window) {  
  
    int zakleszczenie_1 = silnik::Zakonczer_rozgrywki::sprawdz_zakleszczenie(engine_1);  
    int zakleszczenie_2 = silnik::Zakonczer_rozgrywki::sprawdz_zakleszczenie( & engine_2);  
  
    if(zakleszczenie_1 == 0 || zakleszczenie_2 == 0 ) {  
        int static licznik;  
        sf::Sprite pole_wynikowe;  
        sf::Texture text;  
        if(zakleszczenie_1 == 0) text.loadFromFile( filename: "./win/win_2.png");  
        else text.loadFromFile( filename: "./win/win_1.png");
```

**Funkcjonalność** odpowiedzialna za zliczanie ilości figur każdego z graczy, w celu sprawdzenie czy gracz X nie wygrał z graczem Y – z powodu braku figur, **została permanentnie usunięta**.

Było to możliwe dzięki zauważeniu, że jest ona realizowana również poprzez funkcje obsługi zakleszczeń – Gracz Y nie ma możliwości wykonania ruchu, zarówno gdy nie ma pionków, jak i został zablokowany przez gracza X.

## 2) Przykład:

```

void AI::Podazaj_za_krolowa(AI_ruch& ruch,silnik::Inicjalizator_pol& engine,float& ocena) {
    // Jezeli na planszy jest krolowa to ją śledz by uniknac z nia konfrontacji (czy nie jesteś z nia na jednej przekatnej)
    int x_pom1 = ruch.getter_x2();
    int y_pom1 = ruch.getter_y2();
    while(y_pom1 >= 0 && x_pom1 <= 7) { // Kierunek NE
        if(engine.getter_plansza()[y_pom1--][x_pom1+] == engine.getter_cpu_king()) {
            if(engine.getter_plansza()[ruch.getter_y1()][ruch.getter_x1()] == engine.getter_gracz()) ocena += wg_15;
            else if(engine.getter_plansza()[ruch.getter_y1()][ruch.getter_x1()] == engine.getter_gracz_king()) ocena += wg_16;
            return;
        }
    }
    int x_pom2 = ruch.getter_x2();
    int y_pom2 = ruch.getter_y2();
    while(y_pom2 <= 7 && x_pom2 <= 7) { // Kierunek SE
        if(engine.getter_plansza()[y_pom2++][x_pom2+] == engine.getter_cpu_king()) {
            if(engine.getter_plansza()[ruch.getter_y1()][ruch.getter_x1()] == engine.getter_gracz()) ocena += wg_15;
            else if(engine.getter_plansza()[ruch.getter_y1()][ruch.getter_x1()] == engine.getter_gracz_king()) ocena += wg_16;
            return;
        }
    }
    int x_pom3 = ruch.getter_x2();
    int y_pom3 = ruch.getter_y2();
    while(y_pom3 >= 0 && x_pom3 >= 0 ) { // Kierunek NW
        if(engine.getter_plansza()[y_pom3--][x_pom3-] == engine.getter_cpu_king()) {
            if(engine.getter_plansza()[ruch.getter_y1()][ruch.getter_x1()] == engine.getter_gracz()) ocena += wg_15;
            else if(engine.getter_plansza()[ruch.getter_y1()][ruch.getter_x1()] == engine.getter_gracz_king()) ocena += wg_16;
            return;
        }
    }
    int x_pom4 = ruch.getter_x2();
    int y_pom4 = ruch.getter_y2();
    while(y_pom4 <= 7 && x_pom4 >= 0 ) { // Kierunek SW
        if(engine.getter_plansza()[y_pom4++][x_pom4-] == engine.getter_cpu_king()) {
            if(engine.getter_plansza()[ruch.getter_y1()][ruch.getter_x1()] == engine.getter_gracz()) ocena += wg_15;
            else if(engine.getter_plansza()[ruch.getter_y1()][ruch.getter_x1()] == engine.getter_gracz_king()) ocena += wg_16;
            return;
        }
    }
}

```

### Zmiana kodu

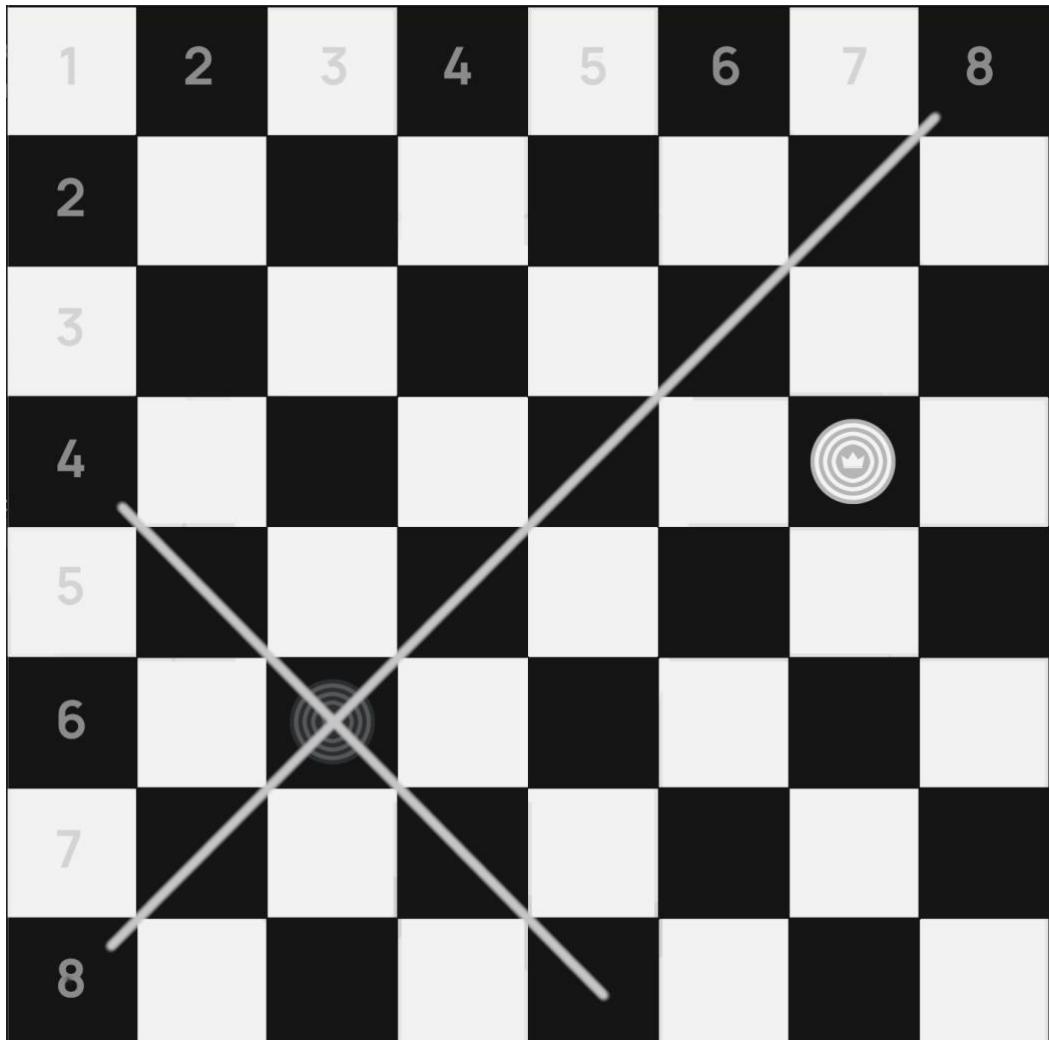
```

void AI::Podazaj_za_krolowa(AI_ruch& ruch,silnik::Inicjalizator_pol& engine,float& ocena) {
    // Jezeli na planszy jest krolowa to ją śledz by uniknac z nia konfrontacji (czy nie jesteś z nia na jednej przekatnej)
    int licznik = 0;
    for(int i=0;i<8;i++) {
        for(int j=0;j<8;j++) {
            if(engine.getter_plansza()[i][j] == engine.getter_cpu_king()){
                int diagonal_up_king = i+j; // identyfikator przekatnej górnej
                int diagonal_down_king = i-j; // identyfikator przekatnej dolnej

                int diagonal_up_pawn_2 = ruch.getter_y2() + ruch.getter_x2();
                int diagonal_down_pawn_2 = ruch.getter_y2() - ruch.getter_x2();

                if(diagonal_up_king == diagonal_up_pawn_2 || diagonal_down_king == diagonal_down_pawn_2) licznik++;
            }
        }
    }
    if(licznik == 1) { // Jezeli po wykonanym ruchu jesteś na jednej linii z królową
        if(engine.getter_plansza()[ruch.getter_y1()][ruch.getter_x1()] == engine.getter_gracz()) ocena += wg_15;
        else if(engine.getter_plansza()[ruch.getter_y1()][ruch.getter_x1()] == engine.getter_gracz_king()) ocena += wg_16;
    }
}

```



Kod odpowiedzialny za sprawdzenie, czy twój pionek, nie znajduje się na jednej przekątnej z wrogą królową, **został w znacznym stopniu usprawniony**. Było to możliwe, dzięki wykorzystaniu własności funkcji liniowej:

$$\begin{cases} y = x + c_1 \\ y = -x + c_2 \end{cases}$$

Gdzie  $c_1$  i  $c_2$  to odpowiednio: `diagonal_down` i `diagonal_up`

Jeżeli  $c_1$  lub  $c_2$  dla naszej figury jest takie samo jak dla królowej przeciwnika, po wykonanym ruchu – to ruch ten jest karany poprzez odjęcie odpowiedniej ilości punktów (w funkcji oceniającej pozycje dla potrzeb AI).

### 3) Przykład:

```
int x1,y1,x2,y2;
sleep(1);
while(!(sf::Mouse::isButtonPressed( button: sf::Mouse::Left)));
if (sf::Mouse::isButtonPressed( button: sf::Mouse::Left)) {
    x1 = static_cast<int>((sf::Mouse::getPosition( relativeTo: window).x) / 175); // pobranie pozycji poczatkowej
    y1 = static_cast<int>((sf::Mouse::getPosition( relativeTo: window).y) / 175); // pobranie pozycji poczatkowej
    cout << "poz_1: " << y1 << " " << x1 << endl;
    sleep(1);
    while (!sf::Mouse::isButtonPressed( button: sf::Mouse::Left)); // oczekuj na kolejna pozycje
    x2 = static_cast<int>((sf::Mouse::getPosition( relativeTo: window).x) / 175); // pobranie pozycji koncowej
    y2 = static_cast<int>((sf::Mouse::getPosition( relativeTo: window).y) / 175); // pobranie pozycji koncowej
    cout << "poz_2:" << y2 << " " << x2 << endl;
    x = x2;
    y = y2;
}

int x1,x2,y1,y2;
if (sf::Mouse::isButtonPressed( button: sf::Mouse::Left)) {
    if(licznik % 2 == 0) {
        x1 = static_cast<int>((sf::Mouse::getPosition( relativeTo: window).x) / (175/resolution_mode)); // pobranie pozycji poczatkowej
        y1 = static_cast<int>((sf::Mouse::getPosition( relativeTo: window).y) / (175/resolution_mode)); // pobranie pozycji poczatkowej

        if(x1 < 0 || x1 > 7 || y1 < 0 || y1 > 7) return 0; // pole poza plansza

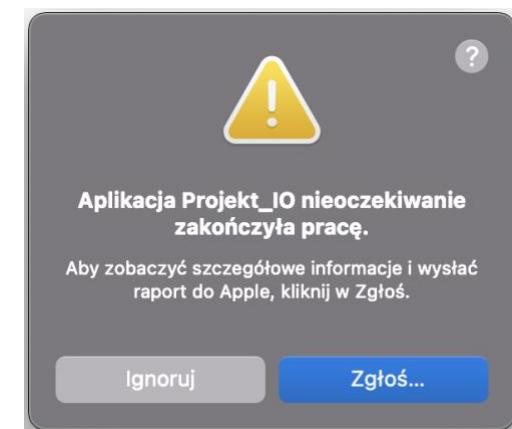
        cout << "poz_1: " << y1 << " " << x1 << endl;
        while (sf::Mouse::isButtonPressed( button: sf::Mouse::Left)); // -----
        return 1;
    }
    else if(licznik % 2 == 1) {
        x2 = static_cast<int>((sf::Mouse::getPosition( relativeTo: window).x) / (175/resolution_mode)); // pobranie pozycji koncowej
        y2 = static_cast<int>((sf::Mouse::getPosition( relativeTo: window).y) / (175/resolution_mode)); // pobranie pozycji koncowej

        if(x2 < 0 || x2 > 7 || y2 < 0 || y2 > 7) return 0; // pole poza plansza

        cout << "poz_2:" << y2 << " " << x2 << endl;
        while (sf::Mouse::isButtonPressed( button: sf::Mouse::Left)); // -----
        return 1;
    }
}
return 0; // nie udalo sie pobrać poprawnego ruchu
```

Zmiana kodu

Dzięki usunięciu funkcji sleep() oraz zmianie sposobu pobieranie pozycji, uzyskana została **większa płynność gry** (pętla główna wykonuje się w sposób ciągły poprzez zastosowanie liczników). Umożliwiło to również, pozbycie się błędu: „nieodpowiadającej aplikacji”, który sprawiał, że program kończył pracę po dłuższym czasie bezczynności ze strony gracza.



Błąd wyeliminowany poprzez zmianę struktury kodu

## 17. Wzorzec projektowy:

Wybrany wzorzec projektowy to **wzorzec budowniczy** z grupy wzorców kreacyjnych. Został zastosowany w celu oddzielenia proces budowania od ostatecznego wyniku produktu. Jesteśmy w stanie uzyskać w ten sposób **zasadę pojedynczej odpowiedzialności (SRC)**. Ułatwia on (przy dużej ilości parametrów w konstruktorze) ustawienie konkretnych, potrzebnych w danym momencie atrybutów do stworzenia obiektu. Sprawia to, że jesteśmy bardziej elastyczni przy dynamicznym określaniu parametrów konstruktora.

```
class Builder{  
private:  
    bool _model_rozgrywki;  
    Motyw _motyw;  
    Motyw _kolor_pionkow;  
public:  
    Builder& setModel(bool mod);  
    Builder& setMotyw(Motyw _mot);  
    Builder& setKolor(Motyw _kolorPion);  
    Atrybuty build();  
};
```

```
class Atrybuty : public Builder{  
private:  
    bool model_rozgrywki;  
    short int motyw;  
    short int kolor_pionkow;  
public:  
    Atrybuty(bool model,Motyw mot,Motyw kol);  
    bool getter_m_r();  
    short int getter_m();  
    short int getter_k_p();  
};
```

Klasa Atrybuty

Klasa Builder

```
Builder build;  
int tryb = OdczytModel();  
build.setModel(mod: tryb);  
int pion = OdczytKolor();  
int plansz = OdczytMotyw();  
if(pion==1){  
    motPion = BLEACH_WHITE_LIGHT_BROWN;  
    build.setKolor(kolorPion: motPion);  
}else if(pion==0){  
    motPion = BLACK_WHITE;  
    build.setKolor(kolorPion: motPion);  
}  
if(plansz == 1){  
    motPlansz = BLEACH_WHITE_LIGHT_BROWN;  
    build.setMotyw(mot: motPlansz);  
}else if(plansz == 0){  
    motPlansz = BLACK_WHITE;  
    build.setMotyw(mot: motPlansz);  
}  
  
Atrybuty at = build.build();
```

Stworzenie atrybutów o zadanych wartościach

Definicje metod w klasie Builder:

- Metody set ustawiające wartości
- Metoda build tworząca

```
/// ---> Builder do Atrybutow:  
Builder& Builder::setModel(bool mod){  
    _model_rozgrywki = mod;  
    return *this;  
}  
Builder& Builder::setMotyw(Motyw _mot){  
    _motyw = _mot;  
    return *this;  
}  
Builder& Builder::setKolor(Motyw _kolorPion){  
    _kolor_pionkow = _kolorPion;  
    return *this;  
}  
Atrybuty Builder::build(){  
    Atrybuty* atr = new Atrybuty(_model_rozgrywki, kol:_motyw, mot:_kolor_pionkow);  
    return *atr;  
}
```

## **18. Zasady gry:**

### **OGÓLNE:**

- Zasady gry są ściśle wzorowane są na warcabach niemieckich – często spotykanych podczas rozgrywania internetowych partii online.
- Wymiary planszy 8x8 jednostek
- Oznaczenia osi planszy [1-8]

### **WSTĘPNE:**

- Grę zaczyna gracz z jaśniejszym motywem pionków
- Wygrywa gracz, który wyeliminuje wszystkie pionki przeciwnika, doprowadzi do sytuacji, w której nie będzie miał on możliwości ruchu lub sprawi, że przeciwnikowi skończy się ustalony czas na grę.
- Remis występuje w momencie wykonania 15 ruchów, bez żadnego bicia, przez obie strony.

### **ZASADY WYKONYWANIA RUCHU:**

- Gracz poruszać może się wyłącznie po ciemnych polach planszy
- Pionek może poruszać się wyłącznie o jedno pole do przodu  
(wyjątek od tej reguły stanowi bicie)
- Pionek, który osiągnie koniec planszy staje się królową
- Królowa może poruszać się o dowolną liczbę pól, zarówno do przodu, jak i do tyłu (o ile na jej drodze nie ma akurat innych figur)

### **BICIA:**

- Bicia są obowiązkowe
- Bicie królową ma wyższy priorytet niż bicie pionkiem
- Należy wykonać w danym ruchu dowolne bicie o najwyższym priorytecie (o ile takowe występuje)
- Bicie pionkiem jest możliwe tylko do przodu, gdy pole za przeciwnikiem jest wolne (na to pole ląduje po biciu figura). Następnie może wykonać kolejne bicie w tej samej turze rozgrywki.
- Bicie królową jest możliwe, gdy pole za przeciwnikiem jest wolne. (na to pole ląduje po biciu figura). Następnie może wykonać kolejne bicie, jeżeli na którejkolwiek z przekątnych takowe występuje.

## 19. Sztuczna Inteligencja:

```
AI_ruch wybierz_najlepszy_ruch(list<AI_ruch>& lista_ruchow_1,silnik::Inicjalizator_pol& engine,float& genaral_rate,list<AI_ruch>& lista_ruchow_2) {
    ///////////////////////////////////////////////////////////////////
    AI_ruch najlepszy_ruch;
    silnik::Inicjalizator_pol eng_temp(engine);
    float ocena_najlepszego_ruchu = -INFINITY; // kazdy ruch jest lepszy od jego braku;
    for(std::list<AI_ruch>::iterator iter_1=lista_ruchow_1.begin(); iter_1 != lista_ruchow_1.end(); iter_1++) {
        float ocena_1 = AI::ocen_ruch( &*iter_1,engine); // ruch AI
        if(!lista_ruchow_2.empty()) {
            for(std::list<AI_ruch>::iterator iter_2=lista_ruchow_2.begin(); iter_2 != lista_ruchow_2.end(); iter_2++) {
                float ocena_2 = AI_gracz::ocen_ruch( &*iter_2,engine); // ruch Gracza
                float ocena = ocena_1 - ocena_2;
                if(ocena_najlepszego_ruchu < ocena ) {
                    najlepszy_ruch = *iter_1;
                    ocena_najlepszego_ruchu = ocena;
                }
            }
        }
        else {
            if(ocena_najlepszego_ruchu < ocena_1 ) {
                najlepszy_ruch = *iter_1;
                ocena_najlepszego_ruchu = ocena_1;
            }
        }
    }
    genaral_rate = ocena_najlepszego_ruchu;
    return najlepszy_ruch;
}
```

Zmienna **ocena** jest maksymalna, gdy ocena\_1 AI jest maksymalna i ocena\_2 gracza jest minimalna

W naszej aplikacji do stworzenia rozgrywki z sztuczną inteligencją (komputerem) został użyty **algorytm min-max**. Jest on stosowany w grach o sumie zerowej (**zero-sum game**), gdzie jedna strona zyskuje, kosztem drugiej. W przypadku partii warcabowej, służy więc do wyboru najlepszego ruchu w danej pozycji.

Jego idea polega na symulacji potencjalnych posunięć, a następnie ocenie aktualnego stanu rozgrywki, zarówno dla gracza jak i komputera, by na ich podstawie **dobrać odpowiednią strategię ruchu**.

Z racji, że jego zaawansowana wersja opiera się na skomplikowanej strukturze drzewiastej, której przeszukiwanie zajmuje dużo czasu i jest skomplikowane w implementacji, zastosowaliśmy **funkcję heurystyczną**, opartą na przeszukiwaniu jedynie jeden ruch w głąb danej pozycji. Uwzględnia ponadto wiele różnych przypadków, jednakże nadal wymaga udoskonalenia i dobrania optymalnych wag na drodze doświadczalnej.

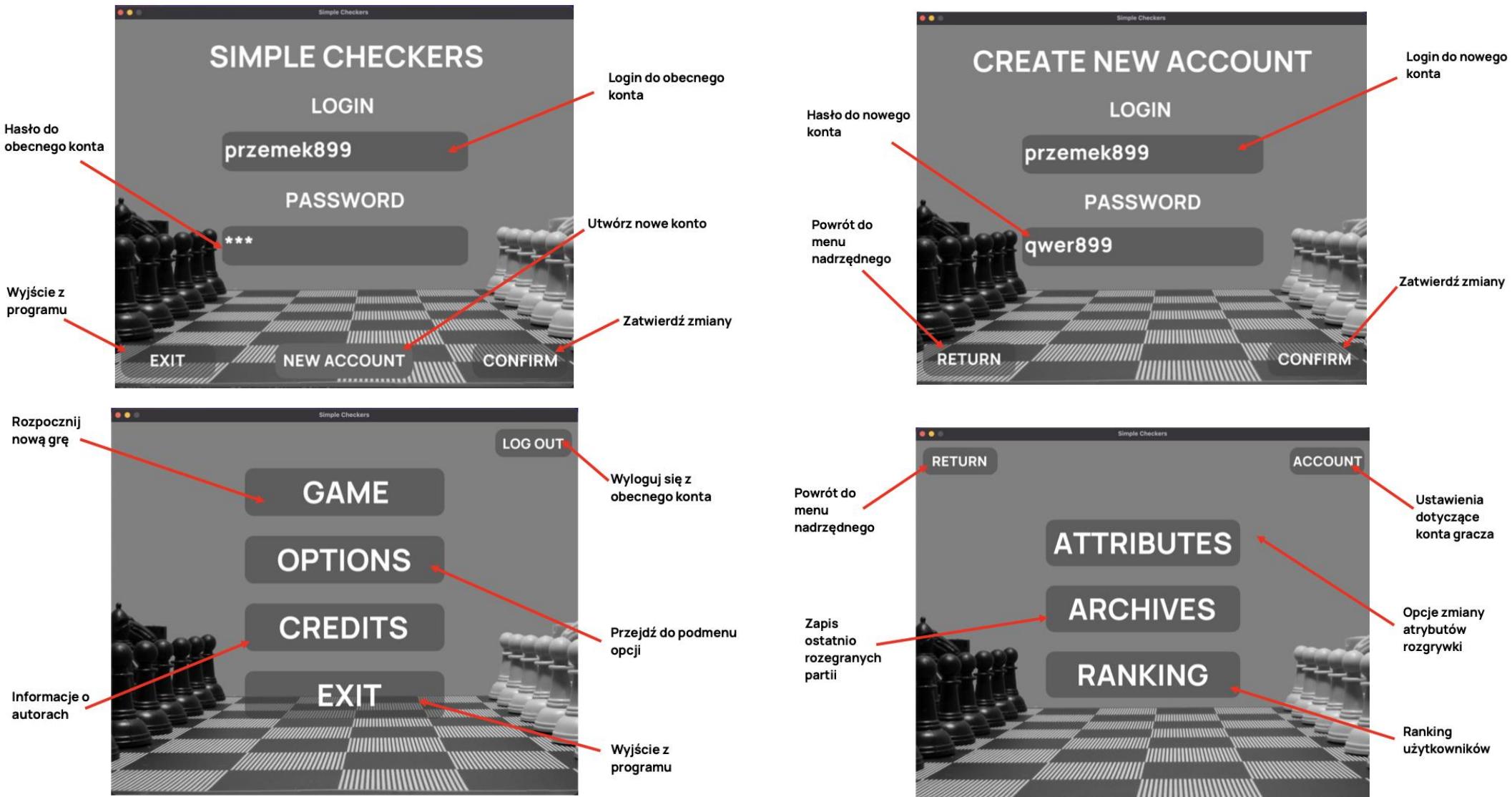
Wagi dobrane w sposób doświadczalny dla poszczególnych sytuacji, które mogą zajść na warcabnicy:

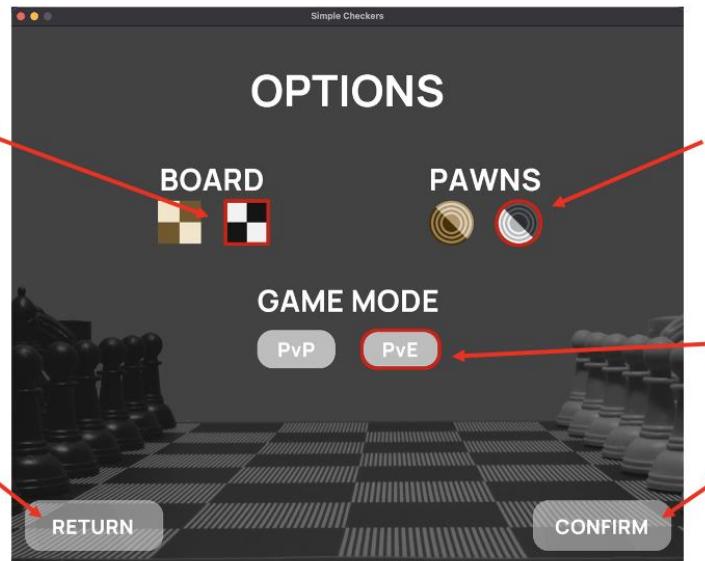
```
#define wg_1 0.25          // Grupowanie się (jeden pionek)
#define wg_2 0.10          // Grupowanie się (dwa pionki)
#define wg_3 0.35          // Pionek ma wolną drogę, by zostać królową
#define wg_4 1.50          // Nagroda za bicie wielokrotne pionkiem
#define wg_6 0.50          // Nagroda za ruch pionkiem do środka planszy
#define wg_7 -0.25         // Ruch na kraniec planszy pionkiem
#define wg_9 -0.50         // Kara za nie bronienie ostatniej linii przez pionka
#define wg_15 0.35         // Nagroda za uniknięcie bicia piona

/// KRÓLOWA:
#define wg_5 1.50          // Nagroda za bicie wielokrotne królową
///      wg_5/2           // Nagroda za bicie królowa (jednokrotne)
#define wg_10 0.20          // Nagroda za ruch królową do środka planszy
#define wg_11 -0.10         // Ruch na kraniec planszy Królową
#define wg_12 0.50          // Nagroda za pulapkę na piona
#define wg_16 0.95          // Nagroda za uniknięcie bicia królowej

/// GENARAL:
#define wg_13 -0.90         // Kara za podejście pod bicie
#define wg_14 -1.80         // Kara za podejście pod bicie wielokrotne
```

## 20. Instrukcja obsługi programu – pomoc:





**RANKING**

POSITION	NICKNAME	POINTS
1.	bbb	340
2.	aaa	220
3.	przemek899	85
4.	login	10
5.	login1	10

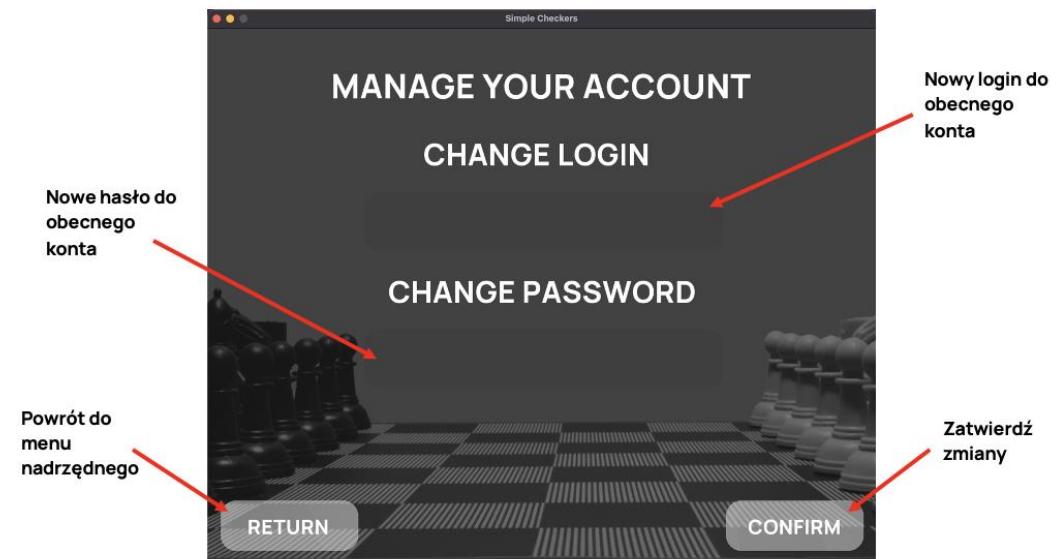
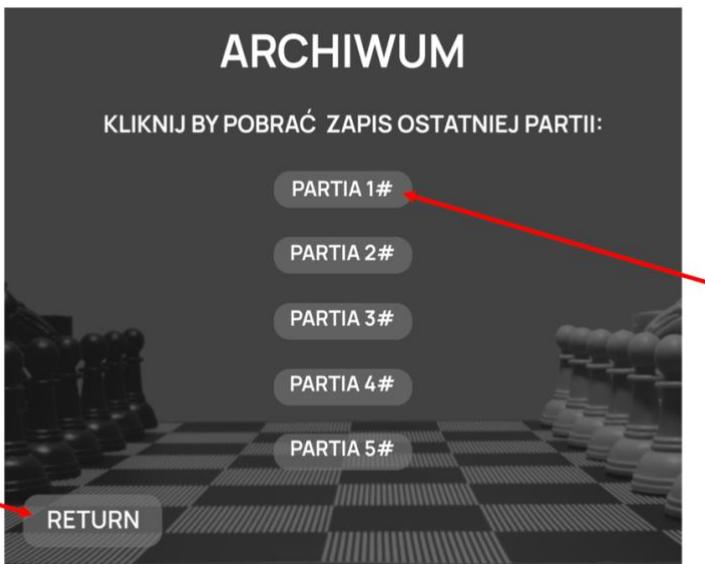
Wynik osiągnięty przez gracza

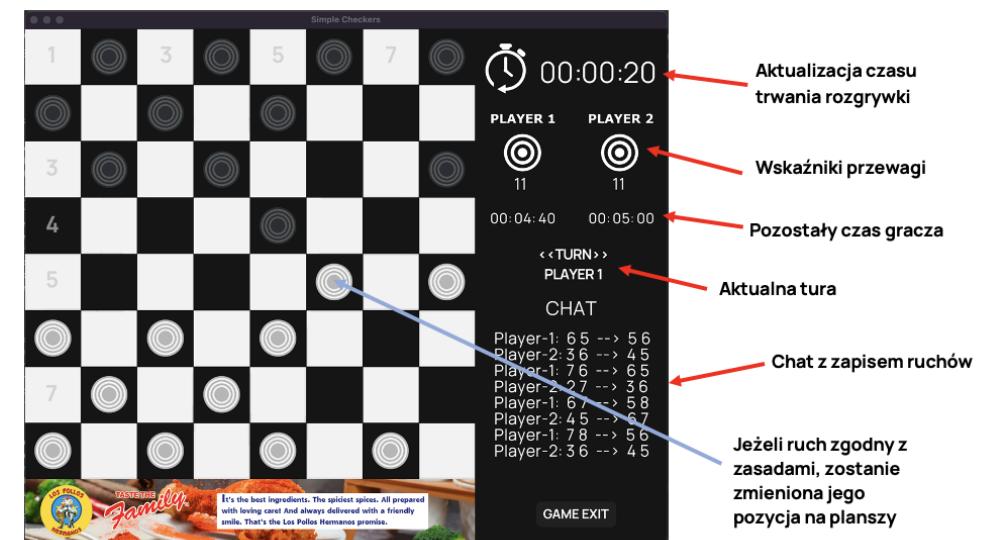
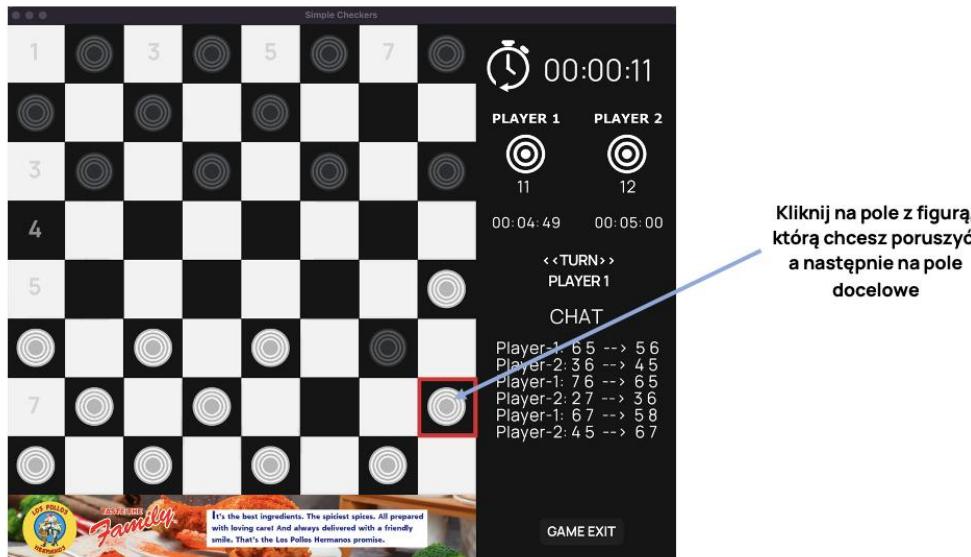
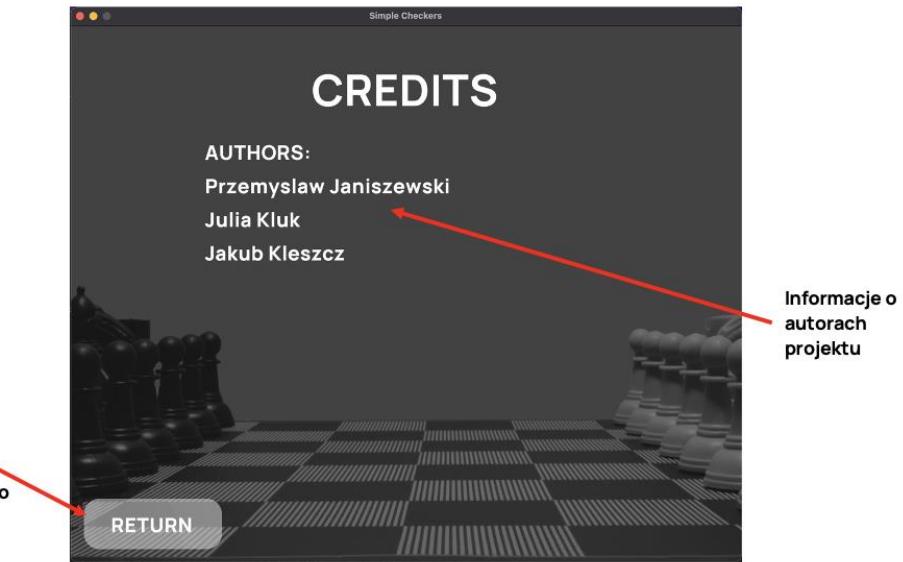
Nazwa gracza

Pozycja w rankingu

Powrót do menu nadzędnego

RETURN





## 21. Podsumowanie:

- Celem naszego projektu było stworzenie systemu, który umożliwiłby rozgrywanie graczowi partii warcabowych oraz zapisywanie jego postępów, by zgodnie z mottem gry – „stał się on mistrzem”
- Wykorzystaliśmy podczas jego tworzenia język C++ , C i język skryptowy Cmake.
- Przeznaczona jest ona na komputery PC z systemem operacyjnym MS Windows, jak i MacOS.

Z racji ograniczonej ilości czasu, nie udało nam się zrealizować wszystkich naszych pomysłów, lecz w przyszłości planujemy udoskonalać projekt m.in. poprzez:

- Umożliwienie graczom rozgrywki online
- Udoskonalenie modułu AI
- Wydanie aplikacji na urządzenia mobilne

Podczas tworzenia aplikacji, występowły drobne przeszkody, które dzięki współpracy zespołu, udawało nam się pokonywać. Staraliśmy się na bieżąco sprawdzać, czy poszczególne części kodu, nie mają ukrytych bugów – przeprowadzając m.in. testy na produkcji, czy wybiórczo – testy jednostkowe.

Planujemy także cały czas utrzymywać stworzony przez nas kod, oraz naprawiać błędy zgłasiane przez osoby korzystające z naszego oprogramowania.

**PODZIAŁ PRACY NAD APLIKACJĄ:**

<b>Przemysław Janiszewski</b>	<b>Julia Kluk</b>	<b>Jakub Kleszcz</b>
- silnik gry (rozgrywka według zasad gry)	- menu główne, menu opcji	-Ranking 5 najlepszych użytkowników
- tryb PvP	- zmiana atrybutów	-Rejestracja nowych graczy
- tryb PvE	- zarządzanie kontem użytkownika	-Logowanie użytkownika do systemu
- tekstury: pionki,plansza,reklamy,wdrożenie	- ranking graczy	-Zmiana danych gracza
- timer	- logowanie	
- chat	- rejestracja nowych użytkowników	
- oznaczenie ruchu - podświetlenie	- obsługa okien w programie	
- wskaźnik przewagi	- obsługa pętli (menu główne i poboczne)	
- pętla poboczna rozgrywki	- tekstury do menu (menu główne i poboczne)	
- obsługa okna rozgrywki		
- gra na czas		

**PODZIAŁ PRACY NAD TWORZENIEM/ POPRAWĄ DOKUMENTACJI:**

<b>Przemysław Janiszewski</b>	<b>Julia Kluk</b>	<b>Jakub Kleszcz</b>
- strona główna/logo/spis treści	Diagram klas	-Wzorzec projektowy
- specyfikacja projektu/przepływ danych	Testy jednostkowe	-Wymagania niefunkcjonalne
- metodyka i jej opis	Diagram wymagań niefunkcjonalnych	-Diagram wymagań (priorytetów)
- diagram PU wraz z opisem	Diagramy aktywności	-Diagram komponentów
- wymagania funkcjonalne		-Diagram klas - odpowiadająca część
- diagram klas – system rozgrywki		-Diagram sekwencji
- diagram wdrożenia/wdrożenie		-Diagram aktywności
- przykłady współpracy (trello/github)		-Wymagania biznesowe
- przykłady refaktoryzacji		
- Zasady gry, opis działania AI i menu		