

UNIWERSYTET JAGIELŁOŃSKI
Wydział Matematyki i Informatyki

kierunek: Informatyka

Klaudia Olejniczak, Damian Misiura, Oleksandr Kuzhel

Teiru MMO (Massively multiplayer online real-time strategy)

Projekt i implementacja .

Teiru MMO (Massively multiplayer online real-time strategy)

Project and implementation .

Kraków 2015

Opracowano zgodnie z ustawą z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych
(Dz.U. 1994 nr 24 poz. 83 ze zm. Tekst jedn.: Dz.U 2006 nr 90 poz. 631 ze zm.)

Abstrakt

Celem tej pracy będzie pokazanie etapu projektowania oraz implementacji gry typu massively multiplayer, czyli gry wieloosobowej Teiru MMO.

Niniejsza praca skupi się głównie na trzech elementach : GUI (Graficzny interfejs użytkowy) , budowie klienta oraz serwerze.

Graficzny interfejs użytkowy to kluczowy element dla każdego programu, pośredniczący pomiędzy programem a użytkownikiem. Szczególnie ważny w przypadku programów biznesowych przeznaczonych dla osób które niekoniecznie posiadają doświadczenie informatyczne.

Serwer jest jedną z najważniejszych części każdej gry wieloosobowej. Programy tego typu przeważnie wykorzystują architekturę typu klient-serwer. Takie rozwiązanie ma zarówno wady jak i zalety które są później szczegółowo rozpatrzone.

Aplikacja kliencka jest niezbędną do rozpoczęcia rozgrywki oraz komunikacji z serwerem gry. Dzięki niej użytkownik może rozpocząć interakcję z innymi graczami tejże gry, walczyć z nimi, tworzyć nowe zorganizowane grupy. Gra oferuje rozwój postaci, wykonywanie ciekawych wyzwań.

Na początku pracy zostanie omówiony pokrótko obraz całego projektu, następnie przedstawienie teorii językaUML i jego użyteczności w procesie modelowania systemu.

Zostanie przedstawiony szczegółowy model gry z naciskiem na warstwę GUI, klienta i serwera. Następnie omówione zostanie proces implementacji, na końcu krótkie podsumowanie projektu .

Abstract

<.... DO przetłumaczenia>

Spis treści

1. Wstęp.	3
2. Krótka charakterystyka gry – Teiru MMO.	4
3. Teoria	5

3. Model interfejsu graficznego.	8
3.1 Diagram przypadków użycia.	9
3.2 Diagram klas.	13
3.3 Diagram sekwencji.	15
3.4 Diagram stanów.	
4. Model aplikacji klienckiej	
5. Model części serwerowej	
4. Implementacja.	
5. Obsługa programu.	
5.1 Ekran logowania.	
5.2 Ekran rejestracji.	
5.3 Ekran menu głównego i opcje.	
5.4 Ekran wyboru i tworzenia postaci.	
6. Podsumowanie.	

Bibliografia

Spis rysunków

Spis tabel

Spis załączników

< DO OGARNIECIA >

Rozdział 1

Wstęp

Gry komputerowe towarzyszą nam wiernie od roku 1947, w którym to powstał pierwszy analogowy symulator pocisku rakietowego. Wraz z rozwojem komputerów, popularność zdobywały gry komputerowe, a później w dobie internetu gry wieloosobowe typu MMO, (ang. multiplayer), pozwalające na rywalizacje wielu graczy z różnych stron świata.

Ta praca ma za zadanie opisać etapy projektowania i implementacji projektu gry typu MMO. Przede wszystkim zostanie zaprezentowany projekt interfejsu graficznego, samego klienta oraz serwera.

Aplikacja jest budowana na bazie pakietu narzędzi Unity oraz technologii .NET. Wykorzystywane w niej są algorytmy bezpiecznego połączenia z serwerem oraz algorytmy typu znajdowanie najkrótszej drogi bądź optymalnego zarządzania obiektami w grze. Serwer został napisany w języku Java z wykorzystaniem bibliotek Netty[1], wersja 3.6.10 i MySql Connector[2] , wersja 5.1.34. Komunikacja po stronie klienta została napisana w języku C# z wykorzystaniem biblioteki graficznej Unity[3] , wersja 4.6. Diagramy przedstawione w pracy zostały stworzone za pomocą Lucidchart[5], a model bazy danych za pomocą Vertabelo[6].

Pierwszy rozdział skupi się głównie na projekcie interfejsu graficznego. Stworzenie odpowiedniego GUI dla aplikacji nie jest prostym zadaniem, dlatego w tej kwestii bardzo ważne jest określenie grupy odbiorców do którego dana aplikacja będzie skierowana. Drugi rozdział skupi się na warwie aplikacji klienckiej , będzie opisywać część logiczną aplikacji, strukturę modułów zbudowanych w tejże aplikacji oraz ich wykorzystanie. Trzeci rozdział będzie głównie poświęcony projektowaniu części serwerowej tak aby zapewnić jak najbardziej wydajność połączenie pomiędzy graczami. Czwarty rozdział będzie skupiony na procesie implementacji naszego zaprojektowanego systemu. Na samym końcu pracy zostanie zaprezentowane podsumowanie oraz rozdział poświęcony obsłudze samego programu.

Rozdział 2

2.1 Krótka charakterystyka gry – Teiru MMO.

Celem projektu jest napisanie gry wieloosobowej czasu rzeczywistego. Podzieliliśmy projekt na 3 części : część kliencka , część serwerowa i interfejs graficzny.

Gra taktyczna stworzona w przestrzeni dwuwymiarowej, oparta głównie na rozgrywkach pomiędzy graczami. Posiada **tryb dla pojedynczego gracza** (ang. single player) z jedną kampanią pozwalającą użytkownikowi zapoznać się z zasadami panującymi w grze oraz **trybem dla wielu graczy** z rozgrywką prowadzoną w sieci Internet (ang. MMO - Massively multiplayer online), który to tryb jest głównym elementem aplikacji.

Single player jest to krótki zarys fabularny mający wprowadzić gracza w najważniejsze rzeczy związane z częścią MMO, takie jak **klasy postaci** i ich różnice, system prowadzenia walk, **zadania** (ang. quests) oraz związane z nimi nagrody, takie jak **złoto** (ang. gold) i **EXP** (ang. Experience points) oraz trochę informacji fabularnych . A na końcu sposób rozwijania postaci ,którzy przychodzi wraz podniesieniem poziomu postaci (ang. level up).

Postać stworzona przez gracza może zostać wybrana z jednej z **czterech dostępnych klas postaci**. Są to **Ranger**, **Mage**, **Knight** oraz **Mystiq**. Każda klasa postaci ma swoje zalety jak i wady, każda też wyróżnia się stylem walki. Przykładowo Mage włada magią w celach ofensywnych, Knight z drugiej strony zawiera dużą ilość obrony i walczy wręcz.

Postaci z czasem mogą awansować na wyższe poziomy, aby odblokować nowe zaklęcia, umiejętności oraz rozdzielić nowo dobyte punkty statystyk. Statystyk w grze jest sześć, są to:

- | | |
|-----------------------|--------------------------------------------------------------------------------------|
| • Strength | – zwiększa możliwy ładunek gracza oraz ilość obrażeń zadawanych w walce wręcz |
| • Dexterity | – określa skuteczność ataków dystansowych oraz ich obrażenia |
| • Constitution | – wytrzymałość, określa ilość punktów życia postaci |
| • Intelligence | – zwiększa ilość punktów statystyk i umiejętności przyznawanych na każdy nowy poziom |
| • Wisdom | – mądrość, zwiększa siłę zaklęć |
| • Charisma | - zwiększa odporność na magiczne obrażenia |

Gracz pierwszopoziomowy otrzymuje wartość 8 każdej ze statystyk, aby zwiększyć każdą z nich należy rozdysponować punkty statystyk otrzymywane za każdy nowy poziom. Kolejno 1 punkt statystyki od 9 do 14, po 14 punkcie statystyk potrzebne są 2 punkty aby awansować. Co każdy poziom postać ma możliwość wyboru nowych zaklęć bądź indywidualnych umiejętności, wyspecyfikowanych dla każdej klasy.

Postać ma możliwość tworzenia ekipunku, zarządzania nim, zbierania nowych przedmiotów z zabitych potworów, nakładania nowych elementów ekipunku, kupna nowych przedmiotów oraz sprzedawania niepotrzebnych przedmiotów.

Gra posiada czat, w którym wszyscy gracze mogą rozmawiać ze sobą. Czat składa się z podczatów: czatu gry, czatu ogłoszeń bądź czatu logu. Gracze mogą tworzyć własne kanały do rozmowy poprzez odpowiednią akcję w grze. Gildie również posiadają swój odrębny czat, tak samo jak walki.

Gra posiada linie zadań (and. quests), które mają za zadanie pomóc graczowi rozwijać jego postać , poprzez zdobywanie punktów doświadczenia (and. experience points) . Pojedynek są oparte o zasadę roll d20. Istnieje kilka rodzajów pojedynków : **PvP**(Player versus Player) walki pomiędzy uczestnikami, **GvG** (Guild versus Guild) walki pomiędzy zespołami 4 osobowymi.

Walka jest podzielona na 3 etapy.

Etap ustawienia - ustalenie kolejki według inicjatyw graczy.

Etap działania – walczący działają w kolejności inicjatyw(od większej do mniejszej).

Etap końcowy – po tym jak każdy z walczących rozegra turę , gracz z większą inicjatywą działa jeszcze raz, a inne etapy powtarzają się aż do końca starcia. W trakcie tury gracz może **wykonąć ruch , akcję całorundową , akcję darmową lub wykonać akcję standardową**.

Sukces w walce zależy w dużej mierze od współczynników bojowych. Krótka lista współczynników bojowych jest przedstawiona poniżej.

Test ataku – próba trafienia przeciwnika , podejmowana jest każdym graczem w swojej turze. Jeśli posiadana przez gracza premia do ataku razem z wynikiem rzutu kostką **k20 przekracza klasę pancerza przeciwnika to przeciwnik otrzymuje obrażenia**.

Premia do ataku – składa się z bazowej premii do ataku(BAB) i z modyfikatora z Siły(dla ataki bronią do walki wręcz) . **Bazowa premia do ataku + kara z zasięgu + modyfikator ze Zręczności (dla ataki bronią do walki dystansowej)**.

Premia z Siły – do obrażeń zadanych w wyniku trafienia przeciwnika bronią do walki wręcz stosuje się modyfikator z Siły.

Klasa Pancerza – to **10 + premia z tarczy + premia z pancerze + modyfikator ze Zręczności**.

Pokazuje to jak trudno wyprowadzić cios który zada graczowi obrażenia.

Punkty wytrzymałości(ang. Heath Points) – przy -10 punktach wytrzymałości postać ginie. Jeśli postać ma nie mniej niż -9 punktów wytrzymałości i nie więcej niż -1 ona zostaje umierająca. Kiedy postać ma 0 punktów wytrzymałości to jest okaleczona.

Okaleczony – (0 punktów wytrzymałości)gracz traci możliwość wykonywania akcji całorundowej i nie może wykonać akcję ruchu i akcję standardową w jednej turze(po winien wybrać jedną z nich). Po zakończeniu tury postać traci jeden punkt wytrzymałości i zostaje umierający.

Umierający – (od -1 do -9 punktów wytrzymałości)umierający bohater nie ma prawa do żadnych działań i każdą rundę traci jeden punkt wytrzymałości , dopóki nie umrze lub dopóki ktoś go nie wyleczy.

Martwy – (-10 i mnie punktów wytrzymałości)grać traci możliwość być wyleczonym i już nie może wrócić do pojedynku.

Leczenie – jeśli podczas leczenia postaci punkty wzrosną powyżej 0 , postać może funkcjonować tak jakby nigdy nie była okaleczona lub umierająca.

Rzuty obronne – postać może być zaatakowana magią lub innym niezwykłym atakiem. Wtedy postać ma prawo do rzutu obronnego który pozwoli jej częściowo lub w całości unicestwić efekt ataku. Wynik rzutu obronnego to rzut kostką k20 + premia (w zależności od klasy, poziomu i atrybutu) .

Rodzaje akcji są podzielone na 3 różne klasy. Podział na klasy odbywa się według długości akcji.

Akcja standardowa – to każde wykonane przez gracza działanie , takie jak atak wręcz lub dystansowy lub jedno z dostępnych zaklęć.

Akcja ruchu – postać zmienia swoją pozycję.

Akcja całorundowa – to działanie które postać wykonuje przez całą rundę.

Niezależnie od kampanii, każdy gracz rozpoczyna grę jedną ze stworzonych postaci . Od tej pory ma pełną kontrolę nad losami swojej postaci, zgodnie z możliwościami przedstawionymi mu w trybie single player

2.2 Technologie i narzędzia.

Serwer został napisany w języku Java z wykorzystaniem bibliotek Netty[1], wersja 3.6.10 i MySql Connector[2] , wersja 5.1.34. Komunikacja po stronie klienta została napisana w języku C# z wykorzystaniem biblioteki graficznej Unity[3] , wersja 4.6.

2.3 Wymagania sprzętowe aplikacji:

Procesor:	dowolny procesor Dual Core, minimum 1,2 GHz
RAM:	minimum 512 MB
Grafika:	karta graficzna z minimum 256 MB pamięci RAM oraz obsługuje DirectX w wersji minimum 9c
HDD:	minimum 1GB miejsca dostępnego na dysku
Pozostałe:	dla trybu multiplayer wymagane stałe połączenie internetowe, klawiatura, myszka.
System:	Windows XP/7/8/8.1/...

Rozdział 3

Teoria.

Rosnąca popularność programowania obiektowego zmieniła metody modelowania systemów

informatycznych. Modelowanie strukturalne zostało zastąpione przez modelowaniem obiektowym. Standardem w tej dziedzinie stał się **język UML (Unified Modelling Language)** - graficzny system wizualizacji, specyfikowania oraz dokumentowania składników systemów informatycznych. W notacji UML modele bądź obiekty przedstawiane są jako różnego rodzaju diagramy. Reprezentowane są one przez obiekty które przedstawione są jako kształty bądź obrazki zawierające odpowiednią treść.

3.1 Język UML

Opis systemu wykonany za pomocą języka UML jest **jednoznaczny**, co bardzo ułatwia napisanie kodu źródłowego w oparciu o modele. Narzędzia do modelowania obiektowego umożliwiają wygenerowanie szkieletu klas i obiektów, a po odpowiednim zintegrowaniu ze środowiskiem programistycznym - pozwalają na dwukierunkową synchronizację modelu z kodem źródłowym.

3.1.1 Diagram przypadków użycia oraz scenariusze przypadków użycia

Diagram przypadków użycia opisuje zachowanie systemu z punktu widzenia użytkownika. Dla deweloperów oprogramowania ten typ diagramów okazuje się być bardzo przydatnym narzędziem, dzięki niemu metodą prób i błędów można zobaczyć założenia użytkowników wobec tworzonego systemu. Diagramy te służą ukazaniu tego co chce użytkownik wobec systemu.

Scenariusz przypadków użycia pokazuje oczekiwanie użytkownika względem konkretnej wykonywanej akcji, proces jej wykonywania oraz etapy w których owa akcja jest wykonywana. Przykładem jest proces kupna jachtu.

2.1.1. Diagram stanów

Diagram stanów opisuję zmianę stanu obiektów względem wykonywanej akcji. Z wykonaniem każdego etapu tej akcji zmienia się stan obiektu bądź modułu systemu, ich zmiany pokazane są na diagramie. Dla programistów jest to ważny diagram ponieważ mogą zauważać **zmianę zachowania obiektu** wobec wykonywanej czynności na nim.

2.1.2. Diagram klas

Diagram klas opisuje wszystkie obiekty tworzone w obecnym module, ich powiązania, na

przykład zawieranie się w sobie bądź związki dziedziczenia. Obiekty mają określone typy, funkcje oraz metody. Każdy element modułu przedstawiany na diagramie ma określoną formę, nie opisywane są konkretne obiekty ale jedynie ich abstrakcyjne formy. Dla programisty jest to szczególnie ważne ponieważ diagram ten pokazuje mu jak zbudowany jest system i jakie są w nim zależności. Również dzięki niemu programista jest w stanie łatwo zaimplementować te powiązania między obiektami.

2.1.3. **Diagram czynności**

Diagram czynności pokazuje jak zachowuje się system w odniesieniu do obecnie wykonującej się operacji.

Jest bardzo podobny do diagramu stanów, jednak zamiast opisywać zachowanie jednego obiektu **opisuje zachowania wielu obiektów**, między którymi nawiązywana jest komunikacja. Pokazuje on programistom sposób działania pewnych czynności, ułatwiając implementację opisywanej diagramem czynności.

2.1.4. **Diagram sekwencji**

Diagram sekwencji prezentuje kolejność wykonania etapów pewnej akcji, **przepływ sterowania informacjami pomiędzy modułami systemu oraz szablon opisywanego algorytmu**. Nie jest uwzględniany dostęp do danych, operacje na danych oraz innych funkcji związanych z komunikacją.

2.1.5. **Diagram komunikacji**

Diagram skupia się na obiektach które wchodzą w skład interakcji oraz komunikatach wymienianych przez nich. Aspekt czasowy ma małe znaczenie, z tego powodu umieszczenie obiektów na diagramie ma cel łatwego opisania relacji między nimi. Diagram opisuje w jaki sposób komponenty komunikują się ze sobą podczas wykonywania danej akcji.

2.1.6. **Diagram komponentów**

Diagram komponentów pokazuje **podział systemu na mniejsze podsystemy logiczne**.

Podsystemy te są połączone między sobą poprzez wspólne funkcje bądź wspólne zapotrzebowanie na daną porcję danych. Komponent to fizyczny moduł kodu, zwykle pakiet. Musi on posiadać swoją unikalną nazwę. Można tworzyć grupy komponentów zwane pakietami, które mogą zostać podsystemami.

2.1.7. Diagram wdrożenia

Diagram wdrożeń pokazuje powiązania pomiędzy oprogramowaniem a sprzętem fizycznym. Wykorzystywane są przy modelowaniu dużych systemów informatycznych.

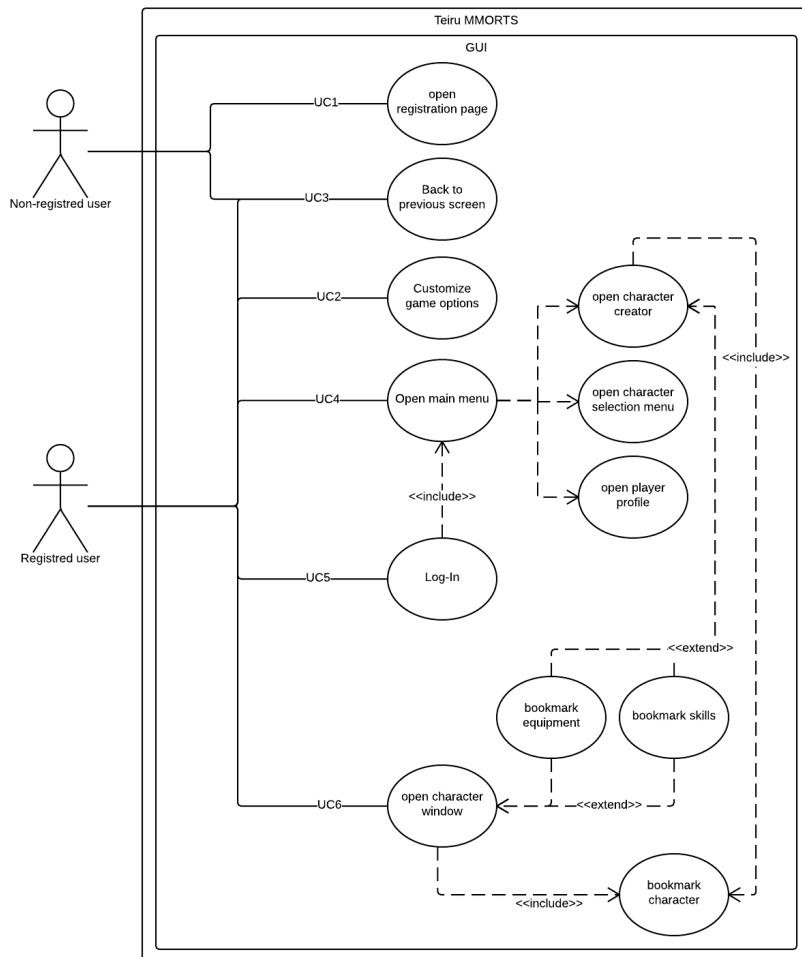
Rozdział 4

Model interfejsu graficznego.

Celem tego rozdziału jest zaprezentowanie modelu użytkowego interfejsu graficznego w

grze komputerowej Teiru MMO.

4.1 Przypadki użycia GUI



Rys.1 Diagram przypadków użycia – GUI

Przypadek użycia	Scenariusz
ID: UC1 Nazwa: Open registration	Scenariusz główny: 1. User wybiera opcje rejestracji.

<p>page</p> <p>Aktorzy: Non-Registered user</p>	<ol style="list-style-type: none"> 2. System prezentuje formularz rejestracji w nowym oknie. 3. User wypełnia formularz. 4. System validuje dane. 5. System po udanej rejestracji informuje użytkownika o tym fakcie. <p>Wyjątki i rozszerzenia:</p> <p>3.A User nie uzupełnił wszystkich wymaganych pól w formularzu.</p> <ol style="list-style-type: none"> 1. System informuje gracza o problemie. 2. Przejdź do kroku 3. <p>4.A User próbuje zarejestrować nazwe/email, która już istnieje w bazie.</p> <ol style="list-style-type: none"> 1. System informuje gracza o problemie. 2. Przejdź do kroku 3. <p>5.A User nie przechodzi procesu uwierzytelnienia i nie zostaje zarejestrowany o czym informuje go system.</p>
<p>ID: UC2</p> <p>Nazwa: Customize game options</p> <p>Aktorzy: Registered</p>	<p>Scenariusz główny:</p> <ol style="list-style-type: none"> 1. Użytkownik uaktywnia button „game options” . 2. System wyświetla okno z ustawieniami. 3. Użytkownik może dostosować opcje gry do swoich wymagań. 4. Użytkownika zapisuje swoje zmiany . 5. System zapisuje zmiany. 6. Okno opcji zostaje zamknięte automatycznie.

	Wyjątki i rozszerzenia:
	<p>4.A</p> <ol style="list-style-type: none"> 1. User anuluje swoje decyzje poprzez przycisk "back". 2. Krok 6.
ID: UC3 Nazwa: Back to previous screen Aktorzy: Registered user, non-registered user	<p>Scenariusz główny:</p> <ol style="list-style-type: none"> 1. User uaktywnia przycisk „back” na dowolnym ekranie. 2. User zostanie cofnięty do poprzedniego ekranu.
ID: UC4 Nazwa: Open main menu Aktorzy: Registered user	<p>Scenariusz główny:</p> <ol style="list-style-type: none"> 1. User otwiera główne menu. 2. System wyświetla graczu okno z menu. 3. User dokonuje akcji <p>Wyjątki i rozszerzenia:</p> <p>1.A Menu główne zostaje wyświetlone też zaraz po zalogowaniu się gracza.</p> <p>3.A Kreator postaci - Gracz zostaje odesłany do ekranu tworzenia postaci.</p> <p style="padding-left: 40px;">A 1. Gracz tworzy postać. 2. Rozpoczyna grę.</p> <p style="padding-left: 40px;">B Gracz wraca do ekranu menu głównego.</p> <p>3.B Wybór postaci – Gracz może wybrać postać którą chce grać.</p> <p style="padding-left: 40px;">A 1. Wybiera postać. 2. Rozpoczyna grę.</p>

	<p>B Gracz wraca do ekranu menu głównego.</p> <p>3.C Profil gracza – Gracz może zobaczyć swój profil, ewentualnie edytować.</p> <ol style="list-style-type: none"> 1. Edytuje 2. A. Zapisuje efekty. B. Anuluje akcje. 3. Wraca do menu głównego.
ID: UC5 Nazwa: Log In Aktorzy: Registered user	<p>Scenariusz główny:</p> <ol style="list-style-type: none"> 1. User wybiera opcje wylogowania. 2. User loguje się do gry. <p>Wyjątki i rozszerzenia:</p> <p>1.A 1. User rezygnuje z logowania i opuszcza program. 2. Program zostaje zamknięty.</p> <p>2.A 1. User wpisuje dane do formularza 2. A. Dane są niedpoprawne, zostaje wyświetlone ostrzeżenie. B. Dane są poprawne, użytkownik zostaje przekierowany do strony głównej.</p>
ID: UC6 Nazwa: Open character window Aktorzy: Registered user	<p>Scenariusz główny:</p> <ol style="list-style-type: none"> 1. User otwiera menu kreatora postaci. 2. System wyświetla graczowi okno z formularzem. 3. Użytkownik uzupełnia formularz i kliką „dalej”. 4. System wyświetla graczowi ekran wyboru skilli.

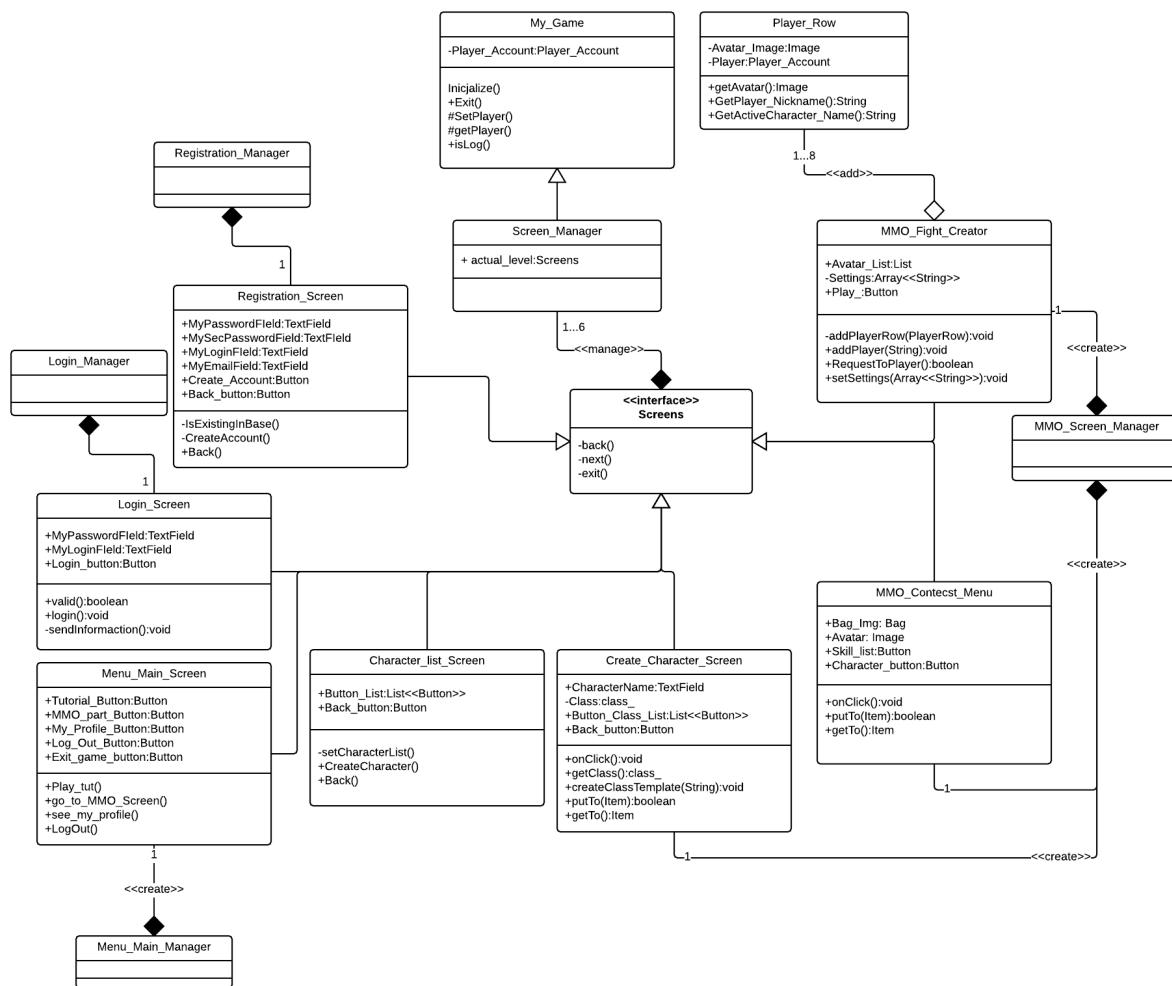
	<p>5. Gracz wybiera skill i wybiera kolejny krok</p> <p>6. System wyświetla graczowi zakładkę ekwipunek.</p> <p>7. Gracz wybiera ekwipunek.</p> <p>8. Zapisuje swoją nową postać.</p>
Wyjątki i rozszerzenia:	
1.A	<p>1. User rezygnuje z tworzenia postaci.</p> <p>2. User zostaje przekierowany do main menu.</p>
2.A	<p>1. User nie wykorzystuje wszystkich punktów statystyk.</p> <p>2. System wyświetla ostrzeżenie o tym.</p>
8.A	<p>1. User rezygnuje z tworzenia postaci.</p> <p>2. User zostaje przekierowany do main menu.</p>

Tab. 1 Tabela do Diagramu przypadków użycia (Rys. 1).

3.3 Diagram klas dla modelu GUI

3.3.1 Diagram klas dla ogólnej budowy ekranów.

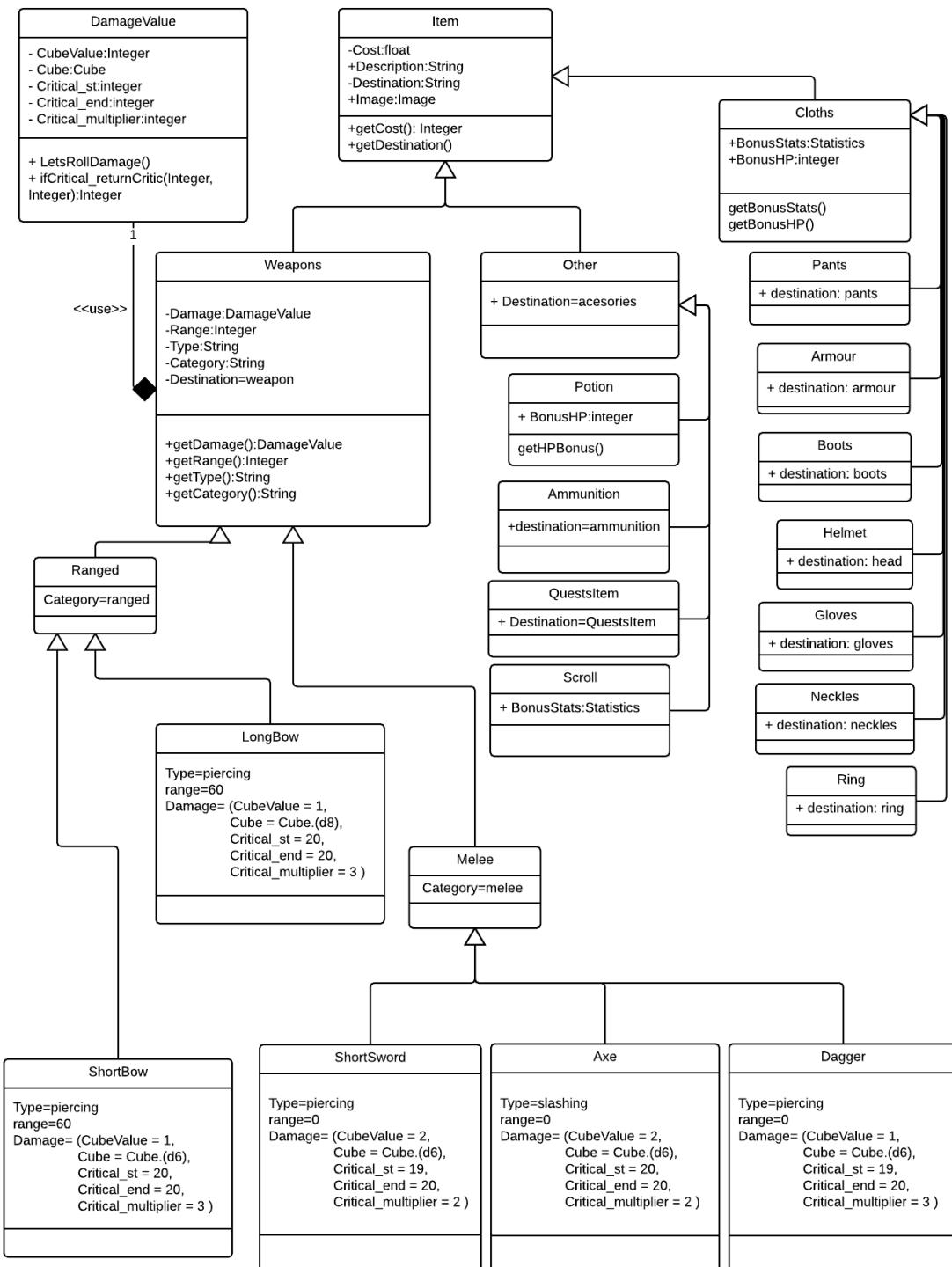
Każdy ekran programu, który ma na celu wyświetlanie elementów oraz formularzy posiada „menadżera” odpowiadającego za sterowanie zachowaniem ekranów oraz walidacje formularzy.



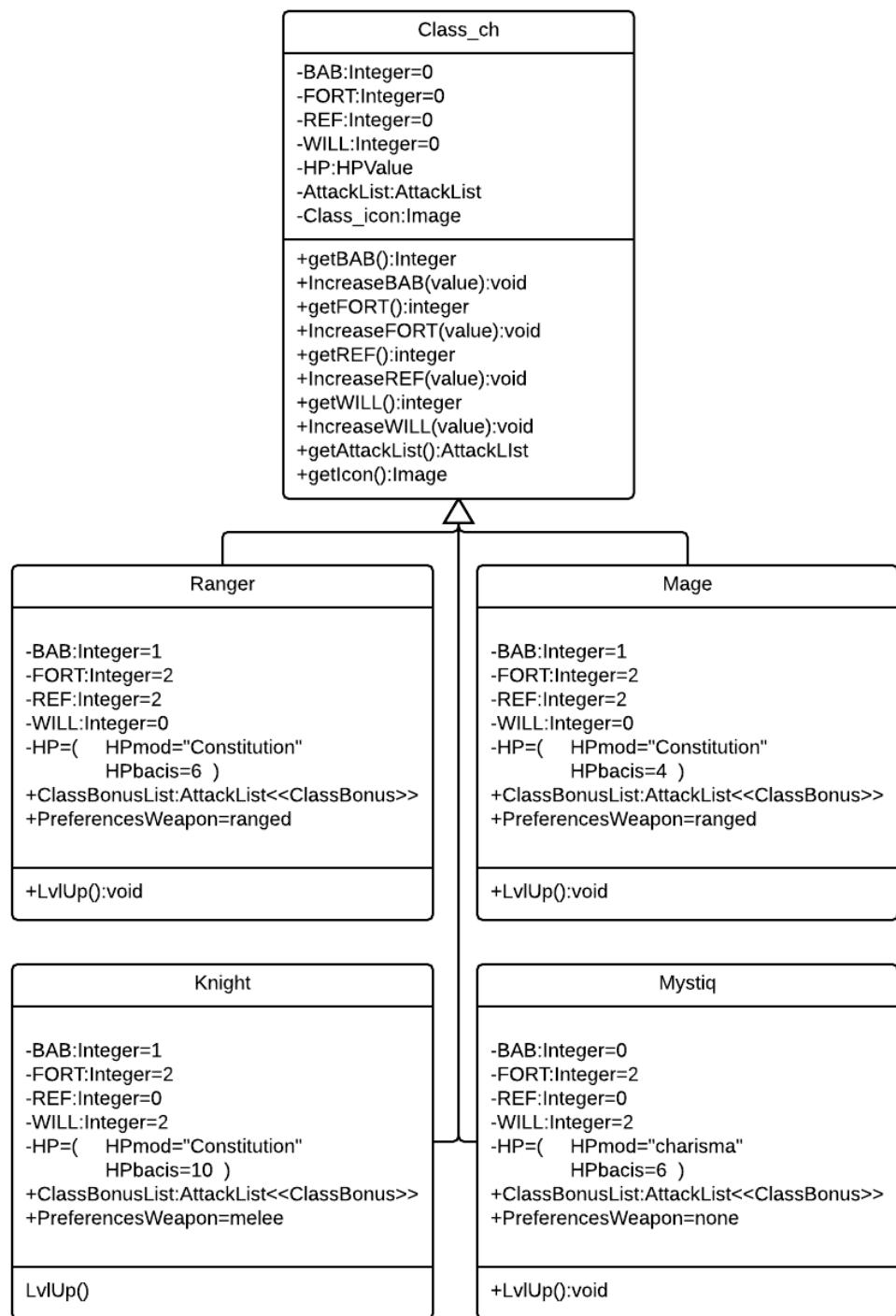
Rys.2 Diagram klas dla ogólnej budowy ekranów w programie.

3.3.2 Diagram klas dla elementów wyświetlanych i użytkowych w GUI .

Elementy te są wykorzystywane w menu kontekstowym oraz kreatorze postaci.



Rys.3 Diagram klas dla elementów „Item” .

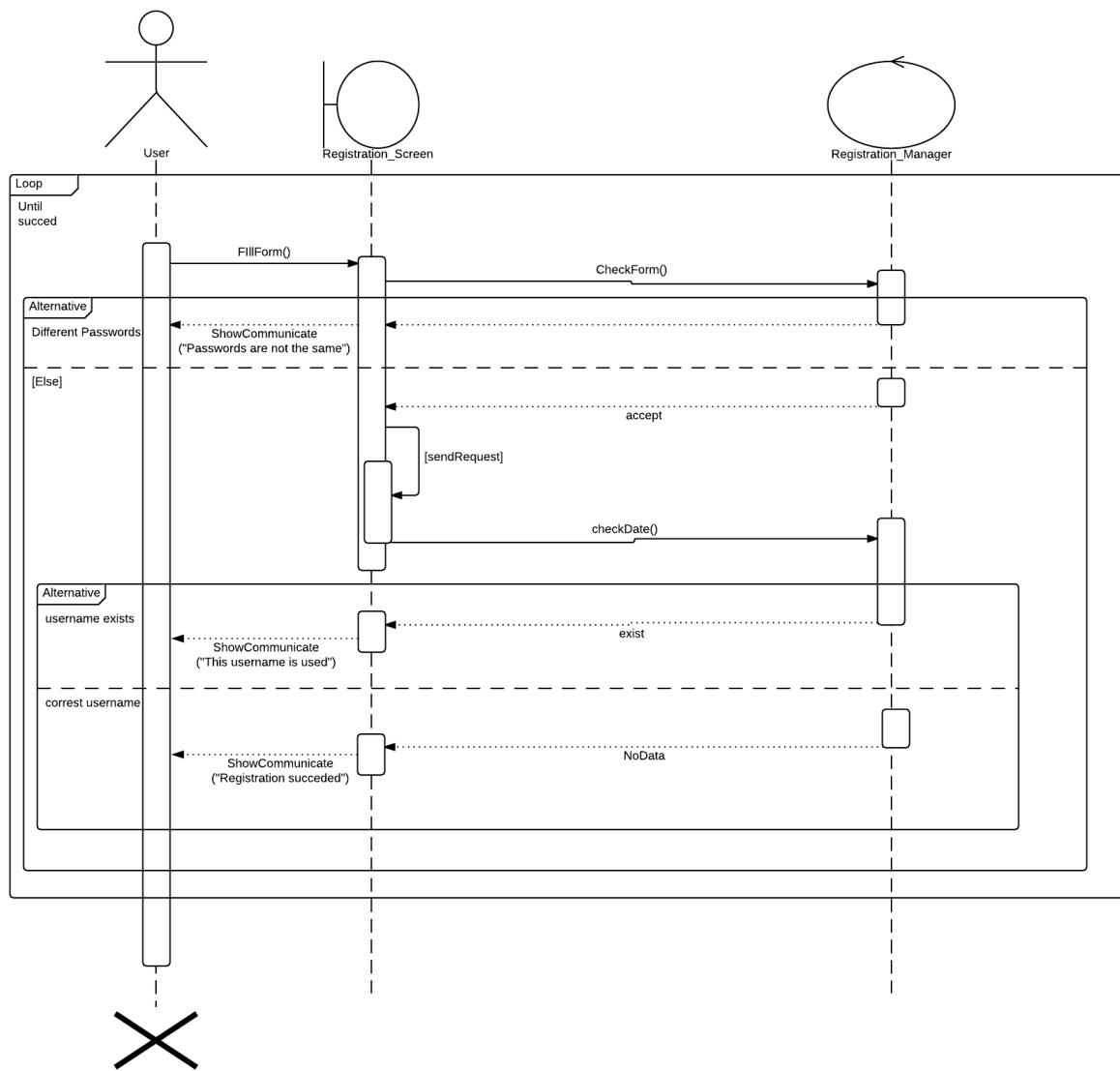


Rys.4 Diagram klas dla elementów „Class_ch” .

3.4 Diagram sekwencji

3.4.1 Diagram sekwencji – Rejestracja

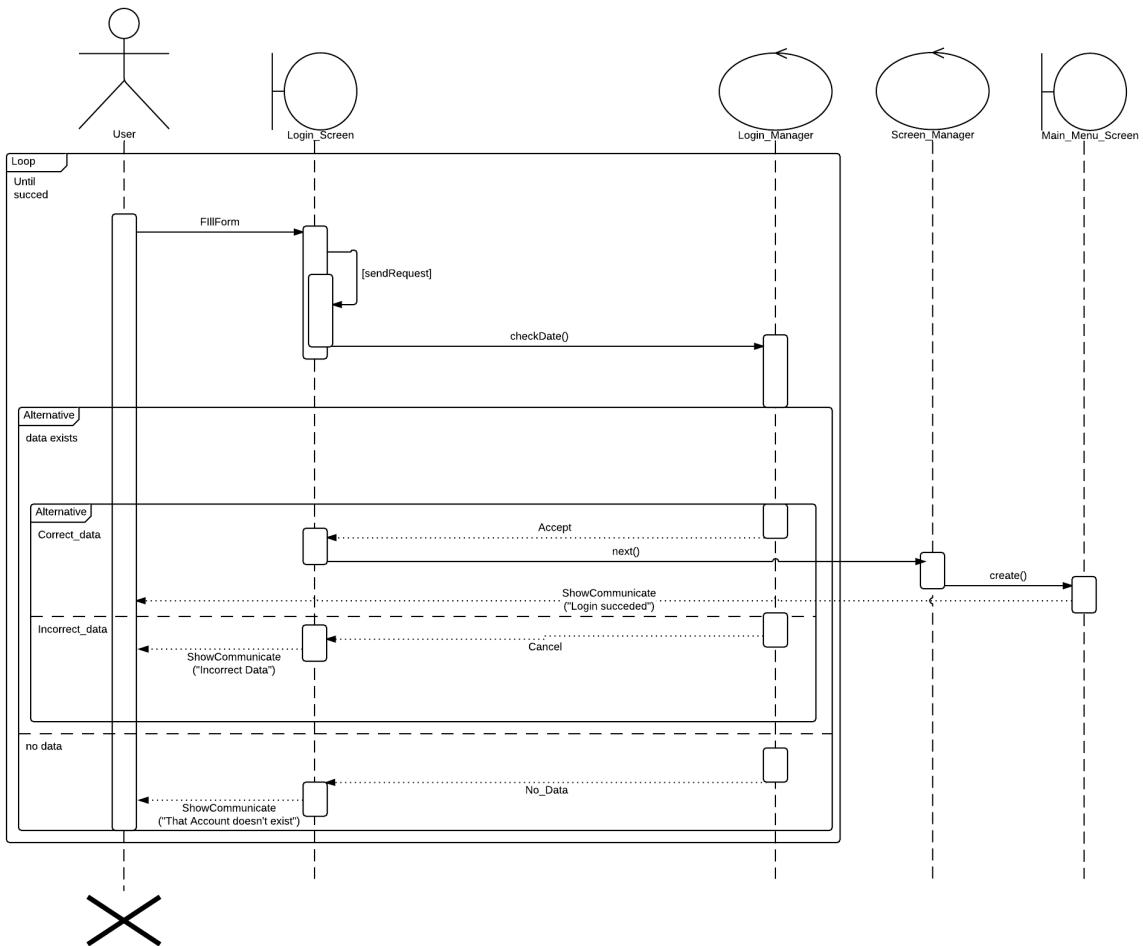
Poniżej przedstawiony jest przebieg procesu rejestracji w obrębie GUI.



Rys.5 Diagram sekwencji – Rejestracja

3.4.2 Diagram sekwencji – Logowanie

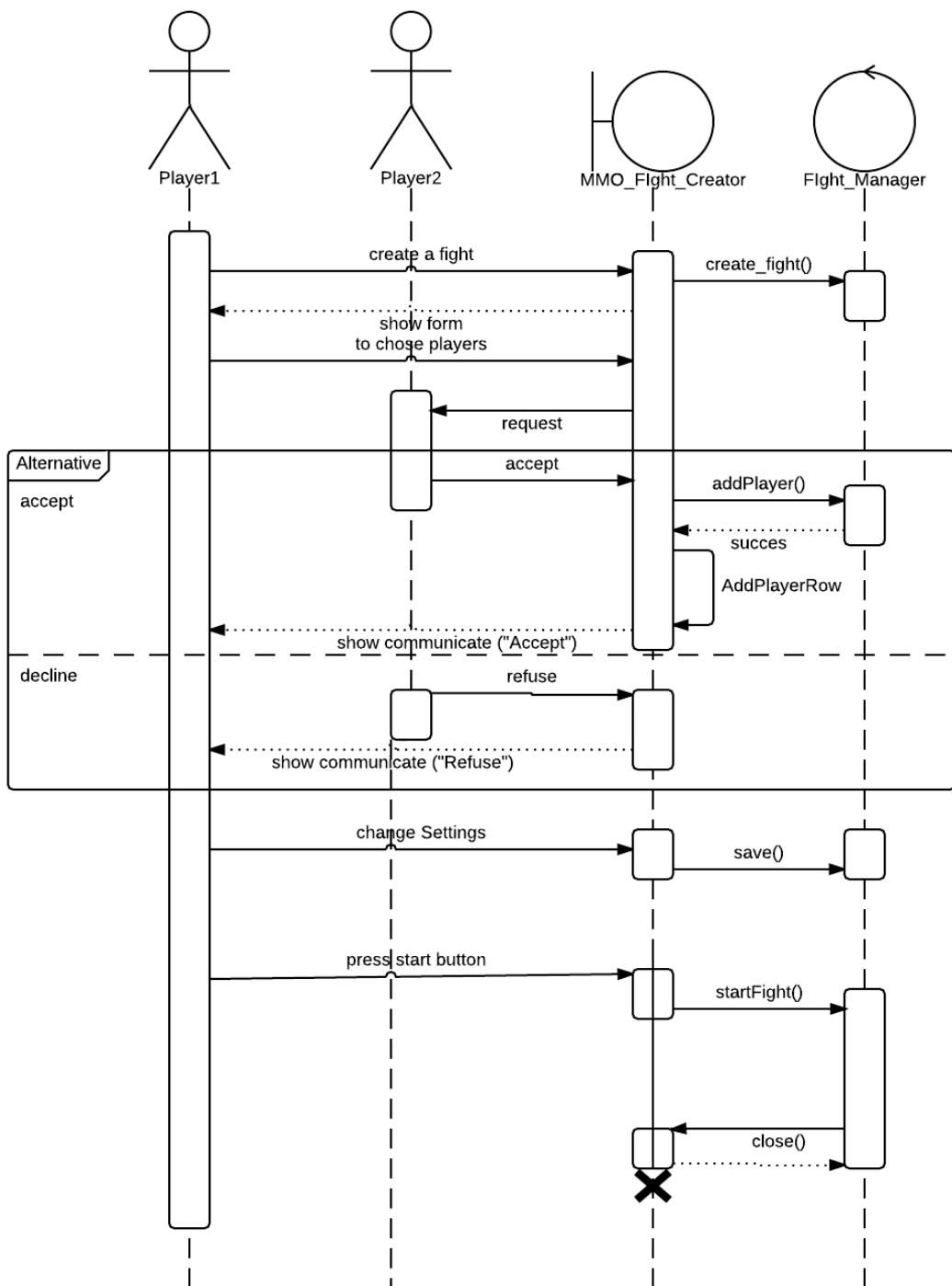
Poniżej przedstawiony jest przebieg procesu logowania w obrębie interfejsu użytkowego.



Rys.6 Diagram sekwencji – Login

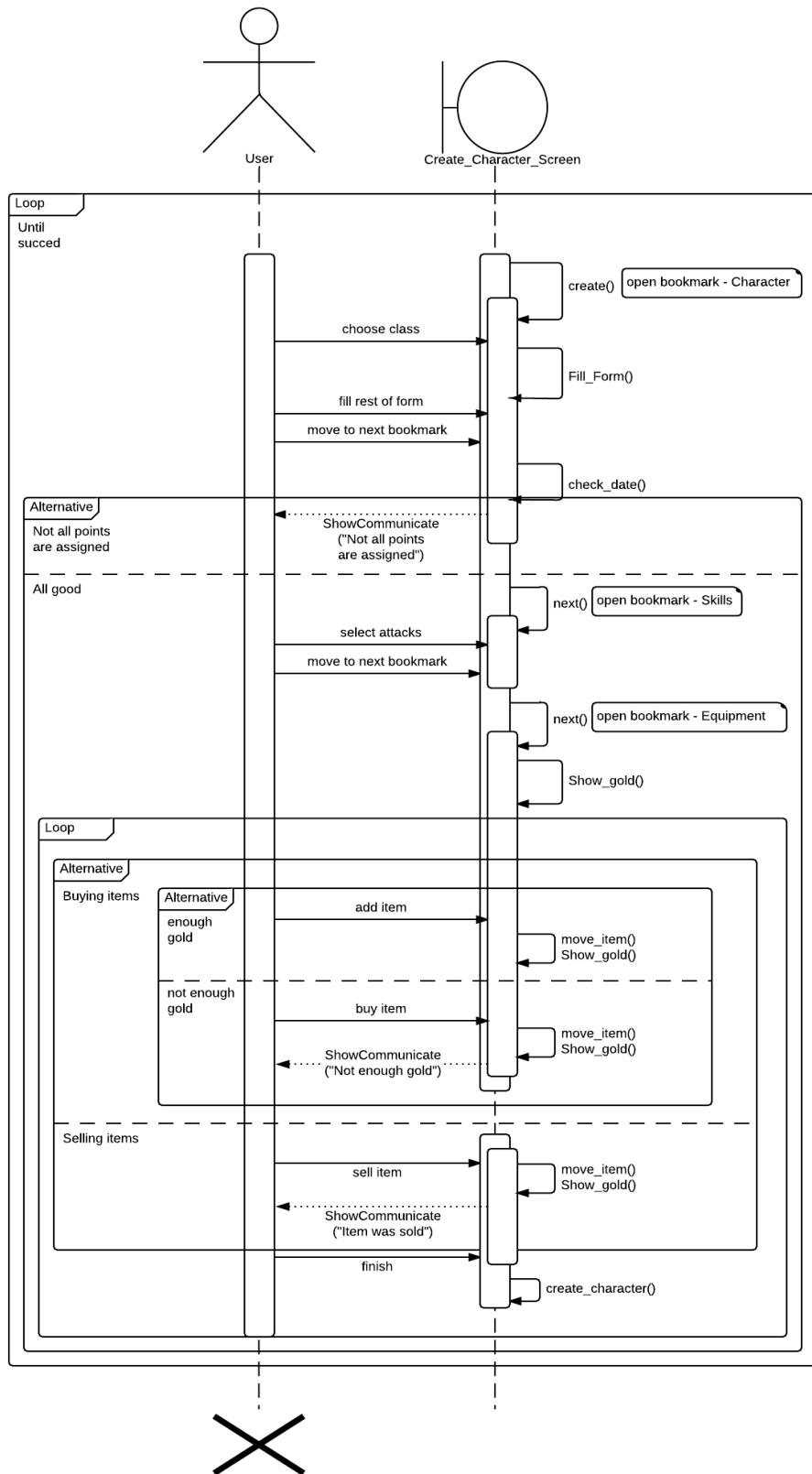
3.4.3 Diagram sekwencji – Tworzenie walki w okienku kreatora walk.

Poniżej przedstawiony jest przebieg procesu tworzenia pojedynku w obrębie interfejsu użytkowego w okienku kreatora walk.



Rys.7 Diagram sekwencji – Tworzenie walki PvP/TvT/GvG

3.4.4 Diagram sekwencji – Kreator postaci.



Rys.8 Diagram sekwencji – Kreator postaci

3.5 Diagram stanów.

3.5.1 Diagram stanów - Okna

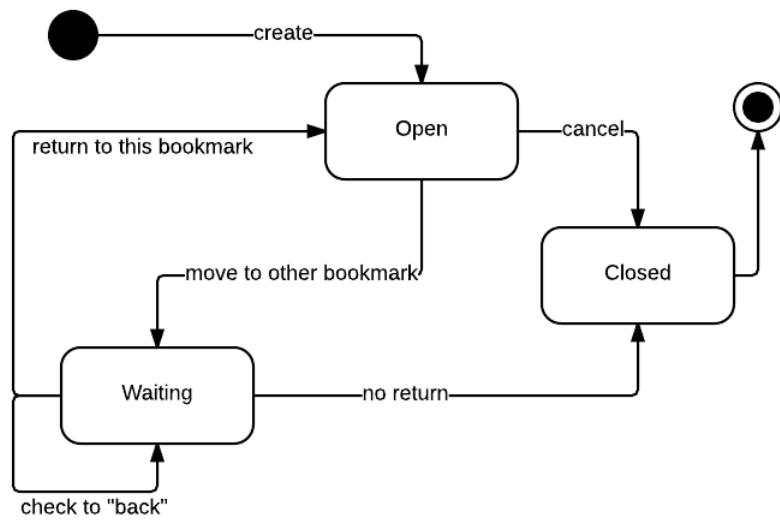


Rys.9 Diagram stanów – Okna

3.5.2 Diagram stanów - Zakładki w oknie kreatora postaci.

Zakładki mogą być w trzech stanach, w zależności od tego co się właśnie dzieje w oknie.

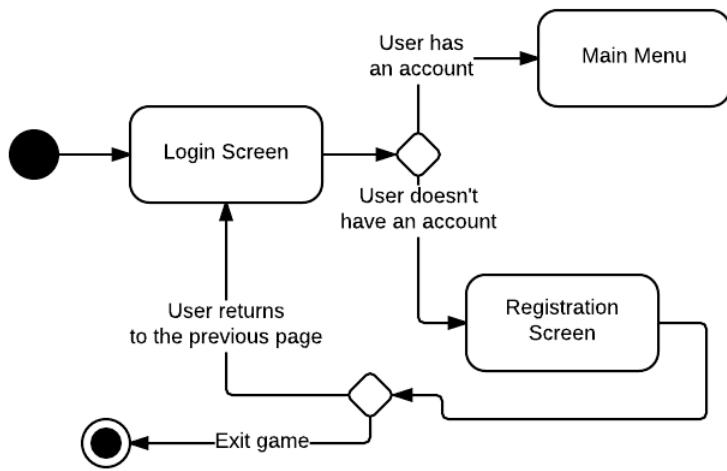
Weźmy pod uwagę pierwszą zakładkę kreatora „Character”, gdy użytkownik ją używa jest ona w stanie „open”, gdy idzie do następnej przechodzi w stan „waiting”, ponieważ za pomocą przycisku „back”, ciągle można do niej wrócić. I na końcu w stanie close, gdy okno zostało zamknięte.



Rys.10 Diagram stanów – zakładki w oknie kreatora postaci.

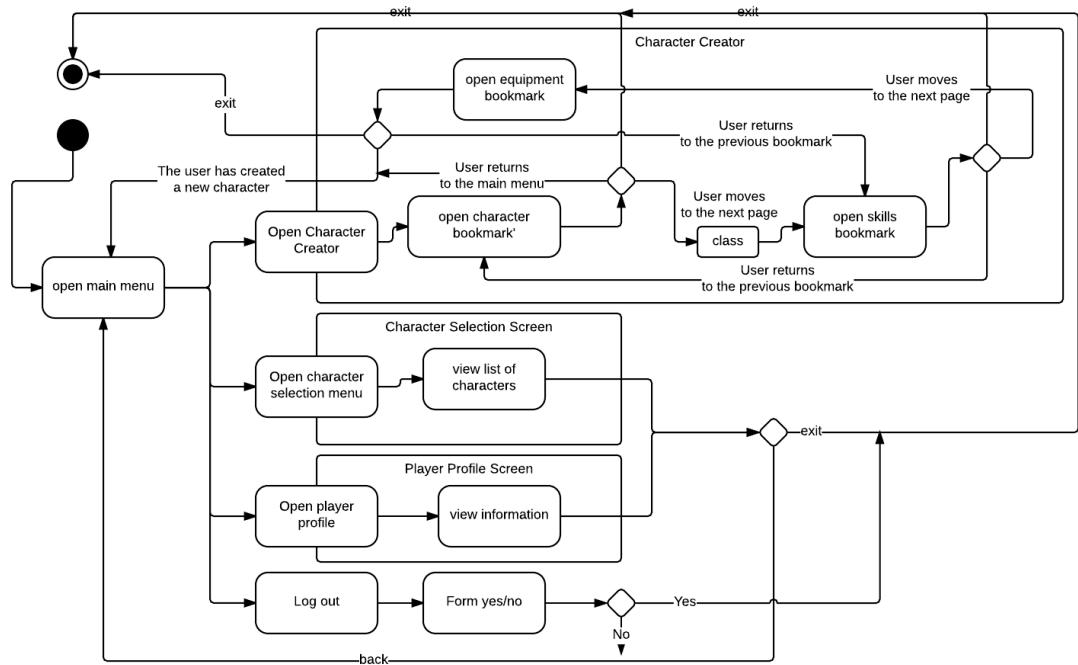
3.6 Diagram aktywności

3.6.1 Diagram aktywności - Dostępność do programu



Rys.12 Diagram aktywności – Dostępność do programu(Logowanie i Rejestracja).

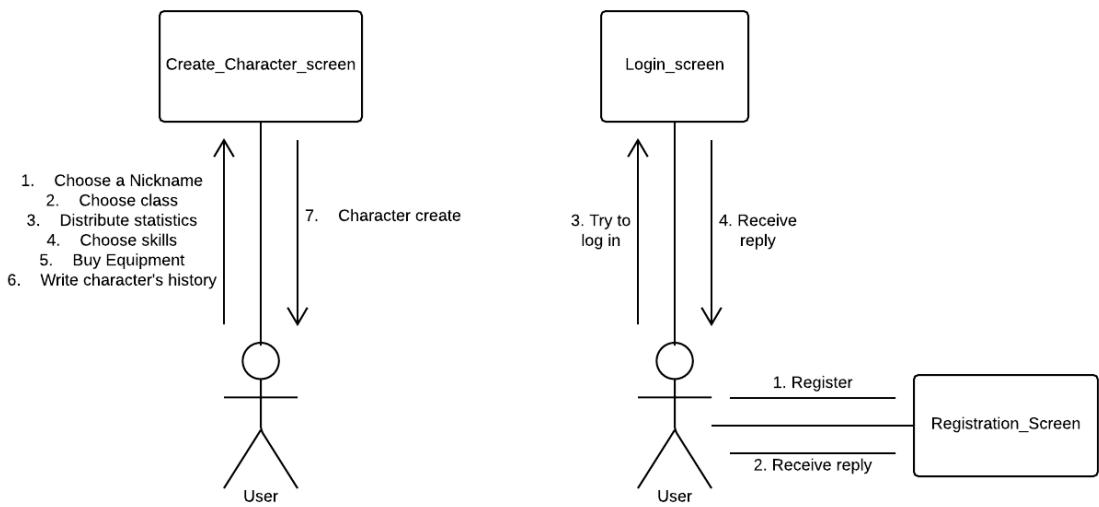
3.6.2 Diagram aktywności - Menu główne, kreator postaci, menu wyboru postaci, profil gracza.



Rys.13 Diagram aktywności – Menu główne i inne ekrany.

3.6 Diagram komunikacji/kooperacji.

Diagramy komunikacji/kooperacji mają na celu pokazać tutaj kolejność wykonywania działań przez użytkownika. W tym wypadku jasno widać, że użytkownik wprowadza dane, bądź wykonuje jakieś operacje w GUI, następnie te informacje zostają przetwarzane w innej części systemu, a interfejs wyświetla odpowiedź systemu.



Rys.14 Diagramy komunikacji/kooperacji

– Tworzenie postaci

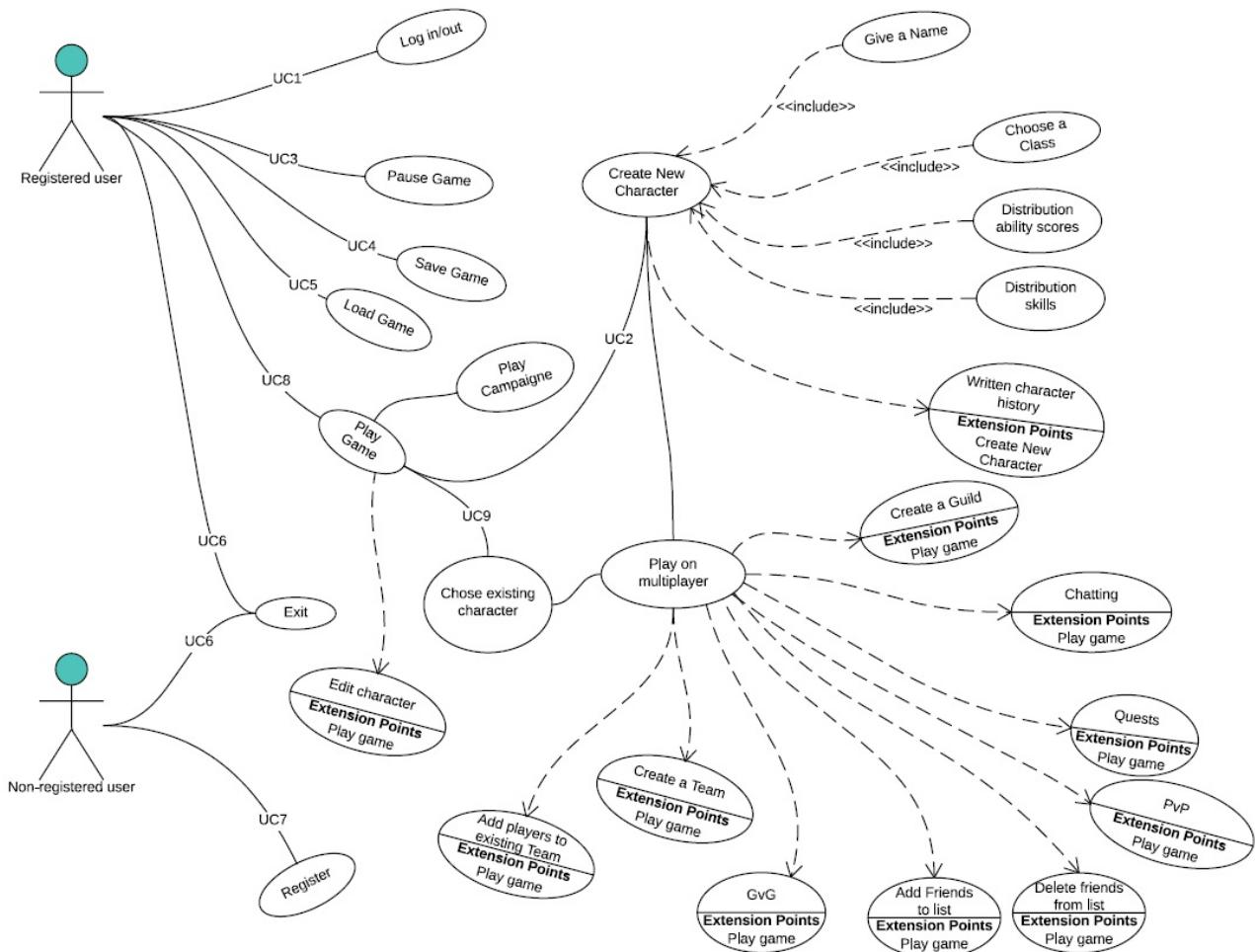
Rys.15. Diagramy komunikacji/kooperacji

– Rejestracja/Logowanie

3.7 Krótkie podsumowanie.

Model interfejsu graficznego stawia przed nami dosyć spore wyzwanie przynajmniej na początku, gdy nie jesteśmy w stanie określić efektu końcowego innych części projektu. Jednak jako warstwa pośrednicząca między klientem, a systemem jest warta uważnego zaprojektowania, gdyż dzięki temu jako programiści jesteśmy w stanie pominąć najbardziej problematyczny etap. Etap w którym to posiadającą całą funkcjonalność systemu, wielu programistów nie potrafi stworzyć wygodnego i przystosowanego dla każdego interfejsu graficznego.

2.2. Przypadki użycia aplikacji klienckiej



Rysunek 1 - Przypadek użycia aplikacji klienckiej przez użytkownika

Użytkownik zarejestrowany bądź niezarejestrowany może prowadzić interakcje z aplikacją kliencką. Użytkownik niezarejestrowany ma możliwość rejestracji bądź wyjścia z gry, funkcje gry są zablokowane dla niego do momentu rejestracji.

Gracz który pomyślnie się zalogował może zacząć grę, wybrać tryb gry (single player lub multi player) i rozpocząć grę. Przy wyborze trybu pojedynczego gracza gracz może rozpoczęć kampanię przygotowującą go do gry wieloosobowej. Jeśli wybierze tryb wieloosobowy musi posiadać do gry stworzoną postać którą może stworzyć teraz bądź wybrać istniejącą postać. Po czym rozpoczyna normalne czynności związane z grą wieloosobową na przykład rozmowa z innymi graczami, handel, polowanie na stwory na mapie gry itd..

Przykładowe scenariusze użycia gry przez użytkownika:

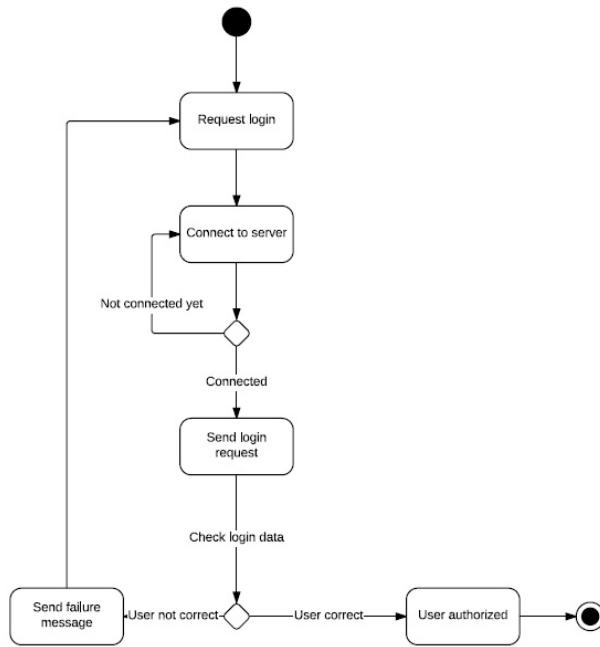
Przypadek użycia	Scenariusz
<p>ID: UC1</p> <p>Nazwa: wylogowanie użytkownika z serwera</p> <p>Aktorzy: Logged user</p>	<p>Scenariusz główny:</p> <ol style="list-style-type: none">Użytkownik wybiera opcje wylogowania.System prezentuje zapytanie, czy użytkownik chce dalej się wylogować.System zapisuje aktualny stan gry/postaci użytkownika.System wylogowuje użytkownika z gry. <p>Wyjątki i rozszerzenia:</p> <p>1.A User chce się wylogować w trakcie aktywnej walki.</p> <p> 1.A.1 System informuje gracza o problemie i wyświetla mu formularz z zapytaniem o poddanie pojedynku.</p> <p> 1.A.2 Przejdź do kroku 3.</p>
<p>ID: UC2</p> <p>Nazwa: Stworzenie nowej postaci</p> <p>Aktorzy: Registered</p>	<p>Scenariusz główny:</p> <ol style="list-style-type: none">Użytkownik chce stworzyć nową postać.System wyświetla formularz z zapytaniem o imię postaci.System sprawdza czy takie imię postaci figuruje w bazie graczy w trybie multi player.User wybiera rasę, klasę oraz rozdaje punkty statystyk dla postaci.System sprawdza czy gracz wydał wszystkie punkty.Gracz musi wybrać umiejętności swojej nowej postaci.User może napisać historię swojej postaci.

	<p>Wyjątki i rozszerzenia:</p> <p>3.A User chce nadać postaci imię, które już figuruje w bazie postaci w trybie multi player.</p> <p>3.A.1 System informuje gracza o problemie.</p> <p>3.A.2 Użytkownik musi wybrać nowe imię dla postaci.</p> <p>3.A.3 Przejdź do kroku 3.</p> <p>5.A User nie rozdał wszystkich punktów statystyk.</p> <p>5.A.1 System informuje gracza o problemie.</p> <p>5.A.2 Użytkownik musi rozdać resztę punktów.</p> <p>5.A.3 Powrót do kroku 7.</p>
<p>ID: UC3</p> <p>Nazwa: Rejestracja nowego użytkownika</p> <p>Aktorzy: non-registered user</p>	<p>Scenariusz główny:</p> <ol style="list-style-type: none"> User wybiera opcję rejestracji. System prezentuje formularz rejestracji. User wypełnia formularz. System sprawdza, czy wszystkie wymagane pola są uzupełnione. System sprawdza dane. System rejestruje nowego użytkownika. <p>Wyjątki i rozszerzenia:</p> <p>3.A Użytkownik nie uzupełnił wszystkich wymaganych pól w formularzu.</p> <p>3.A.1 System informuje gracza o problemie.</p> <p>3.A.2 Przejdź do kroku 3.</p> <p>5.A Użytkownik próbuje zarejestrować nazwę lub adres e-mail, która już istnieje w bazie.</p> <p>5.A.1 System informuje gracza o problemie.</p> <p>5.A.2 Przejdź do kroku 3.</p> <p>6.A Użytkownik przechodzi proces uwierzytelniania i zostaje zarejestrowany.</p>

	<p>6.B Użytkownik nie przechodzi procesu uwierzytelnienia i nie zostaje zarejestrowany.</p>
ID: UC4 Nazwa: Rozpoczęcie nowej gry Aktorzy: Registered user	<p>Scenariusz główny:</p> <ol style="list-style-type: none"> 1. Użytkownik wybiera rodzaj rozgrywki. 2. Użytkownik rozpoczyna nową grę. <p>Wyjątki i rozszerzenia:</p> <ol style="list-style-type: none"> 1.A Użytkownik rozpoczyna kampanię. 1.B Użytkownik rozpoczyna rozgrywkę w trybie multi player. <ol style="list-style-type: none"> 1.B.1 Użytkownik rozmawia z innym graczem poprzez czat. 1.B.2 Użytkownik tworzy drużynę. <ol style="list-style-type: none"> 1.B.2.1 Użytkownik dodaje lub usuwa gracza z drużyny. 1.B.2.2 Użytkownik proponuje pojedynek między drużynami. 1.B.3 Użytkownik tworzy gildię. 1.B.3.1 Użytkownik dodaje lub usuwa gracza z gildii. 1.B.3.2 Użytkownik proponuje pojedynek między gildiami. 1.B.4 Użytkownik pojedynkuje się z innym graczem. 1.B.5 Użytkownik pobiera lub anuluje zadanie. 1.B.6 Użytkownik dodaje lub usuwa gracza z listy przyjaciół. 1.C Użytkownik tworzy nową postać. 1.D Użytkownik wybiera postać już istniejącą.

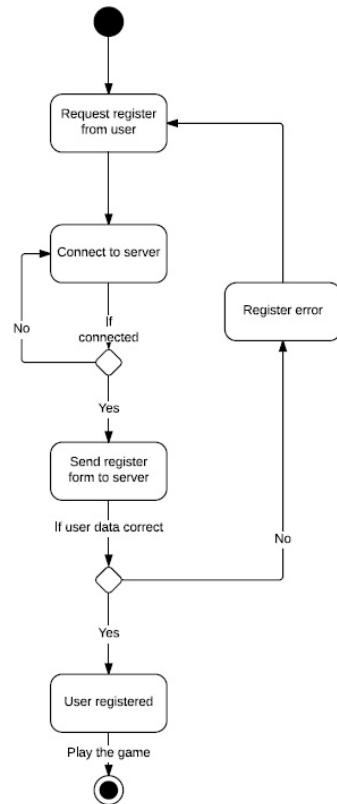
2.3. Diagramy czynności ważniejszych funkcji programu

Poniżej przedstawione diagramy opisują ważne części systemu, bez których gra wieloosobowa nie ma sensu:



Rysunek 2 - Diagram czynności dla logowania do serwera

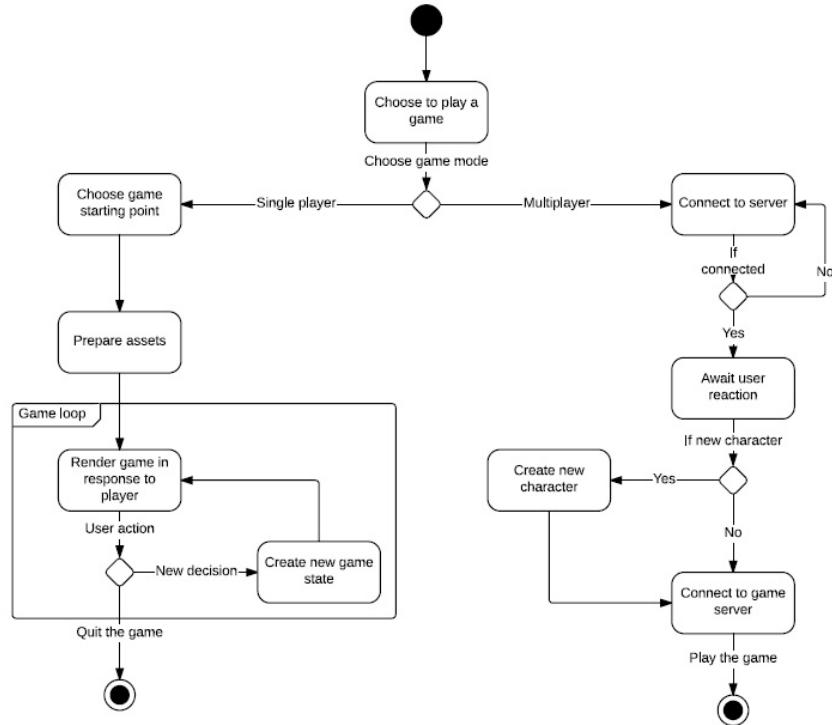
Diagram przedstawia proces logowania do systemu. Użytkownik żąda zalogowania, po czym aplikacja kliencka loguje się do serwera, przesyła mu żądanie logowania. Jeśli serwer odpowie błędem, operacja logowania jest przywracana do początku.



Rysunek 3 - Diagram czynności dla rejestracji nowego konta

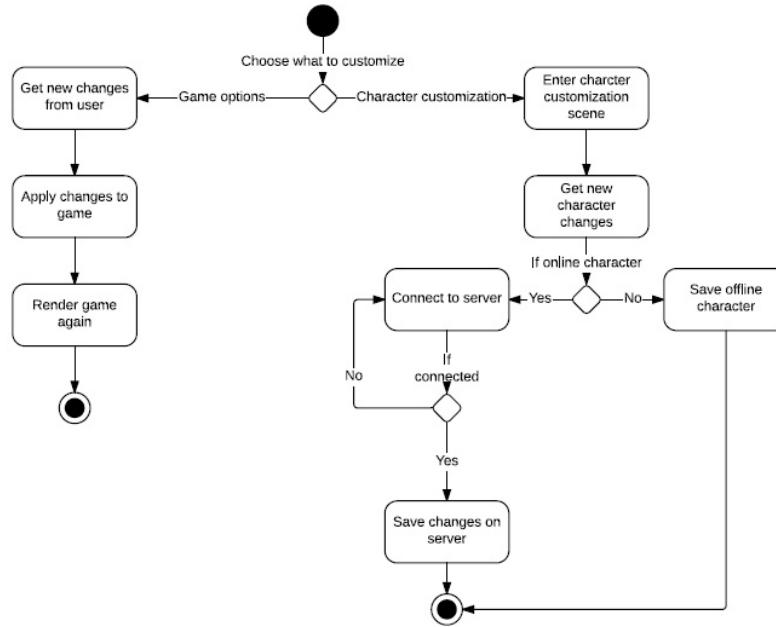
Użytkownik żąda stworzenia nowego konta w serwisie. Aplikacja tworzy szablon żądania po czym wysyła je na serwer. Ten po odebraniu danych sprawdza czy dane są poprawne i czy

użytkownik już nie istnieje w serwisie. Jeśli nastąpił jakikolwiek błąd, serwer zwraca ten błąd i cała procedura rejestracji jest powtarzana od nowa. W przeciwnym przypadku rejestracja jestkończona sukcesem i użytkownik może korzystać z gry.



Rysunek 4 - Diagram czynności dla rozpoczęcia gry

Użytkownik ma opcję wyboru trybu gry: single player mode and multi player mode. Gdy wybierze single player mode ma możliwość wyboru punktu rozpoczęcia rozgrywki (nową kampanię bądź zapisany stan gry), następnie aplikacja kliencka przygotowuje potrzebne zasoby do stworzenia gry i tworzy pętlę gry (game loop) gdzie oczekuje na decyzje gracza i dostosowuje narysowany obraz do jego decyzji. Jeśli gracz zdecyduje się na tryb wieloosobowy, musi się połączyć z serwerem. Po nawiązaniu połączenia klient oczekuje na decyzję gracza jaką postacią rozpocznie grę. Następnie użytkownik jest łączony z serwerem gry i następuje rozpoczęcie rozgrywki sieciowej.



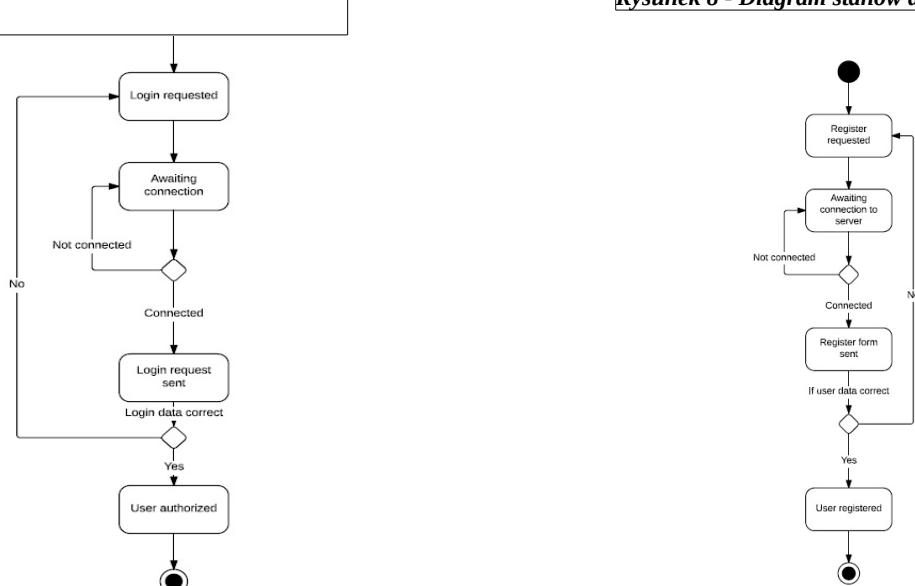
Rysunek 5 - Diagram czynności dla dostosowania elementów gry

Użytkownik musi wybrać co chce dostosować w grze, jeśli to są same gry to wybiera spośród opcji w menu opcji nowe ustawienia i zapisuje je bądź anuluje. Klient wtedy odczytuje nowe ustawienia i ponownie tworzy środowisko gry dostosowane do nowych ustawień. Jeśli wybrane zostanie dostosowanie postaci wtedy użytkownik dobiera interesujące go opcje, zmienia statystyki bądź umiejętności postaci, po czym w zależności czy to postać z kampanii bądź gry wieloosobowej zmiany zostają zapisane odpowiednio na komputerze klienta bądź na serwerze gry.

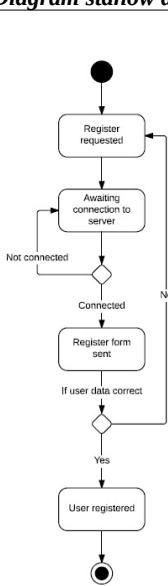
2.4. Diagramy stanów ważniejszych funkcji programu

Diagramy stanów opisują stan klienta podczas wykonywania ważnych czynności w programie, jest to bardzo przydatne dla programisty ponieważ pokazuje zachowanie obiektu względem czynności, co ułatwia implementację.

Rysunek 7 - Diagram stanów dla logowania



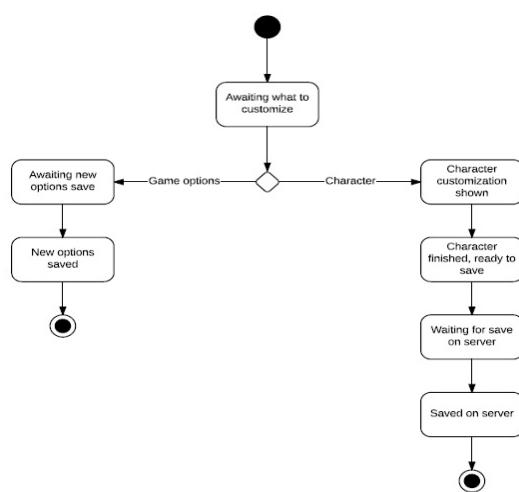
Rysunek 8 - Diagram stanów dla rejestracji



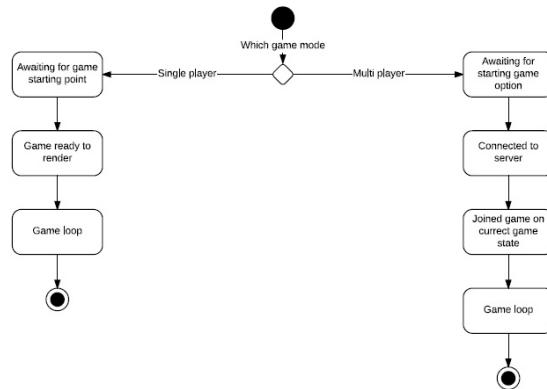
Rysunek 7 pokazuje stan klienta podczas procesu logowania. Klient pierw otrzymuje żądanie o logowanie do serwisu, po czym oczekuje na połaczenie z serwerem. Aplikacja wysyła żądanie dla serwera z logowaniem i w zależności od otrzymanej odpowiedzi akceptuje użytkownika do gry bądź odrzuca go i zaczyna proces logowania od nowa.

Rysunek 8 ukazuje stan klienta podczas rejestracji nowego użytkownika. Klient otrzymuje żądanie rejestracji klienta, po czym łączy się z serwerem. Następnie zostaje wysłane żądanie o rejestrację nowego użytkownika do serwera. Jeśli się powiedzie nowy użytkownik zostaje stworzony w bazie, w przeciwnym wypadku cały proces jest ponawiany.

Rysunek 9 - Diagram stanów dla dostosowanie aplikacji



Klient oczekuje na decyzję użytkownika o tym co on chce dostosować. Jeśli wybrał opcje gry, klient oczekuje na zapisanie zmian, po czym zapisuje je i tworzy środowisko gry dostosowane do nowych ustawień. Jeśli zostało wybrane dostosowanie postaci, klient oczekuje na stworzenie zmian, po czym zapisuje zmiany na komputerze użytkownika i zapisuje zmiany na serwerze.

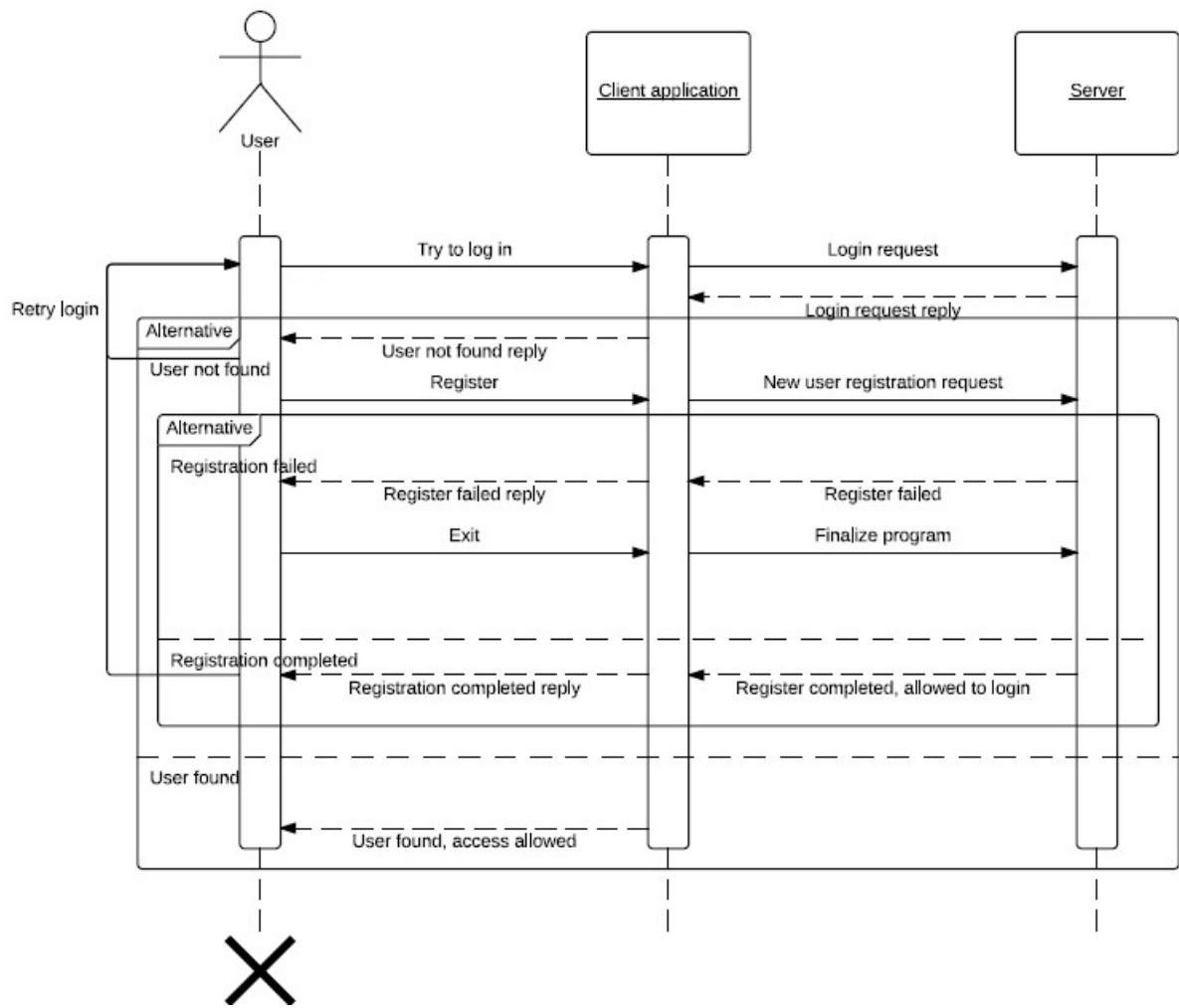


Rysunek 10 - Diagram stanu rozpoczęcia gry

Klient początkowo oczekuje na wybranie przez użytkownika trybu gry. Jeśli gracz wybrał tryb pojedynczego gracza klient oczekuje na wybranie punktu startego (nową grę lub wznowienie poprzedniej rozgrywki), po czym przygotowuje grę do działania i przechodzi do pętli gry (game loop). Jeśli wybrany został tryb wieloosobowy, gra oczekuje na wybranie postaci przez gracza, następnie łączy się z serwerem, dołącza do aktualnego stanu gry serwera. Klient przechodzi wtedy do pętli gry.

2.5. Diagramy sekwencji

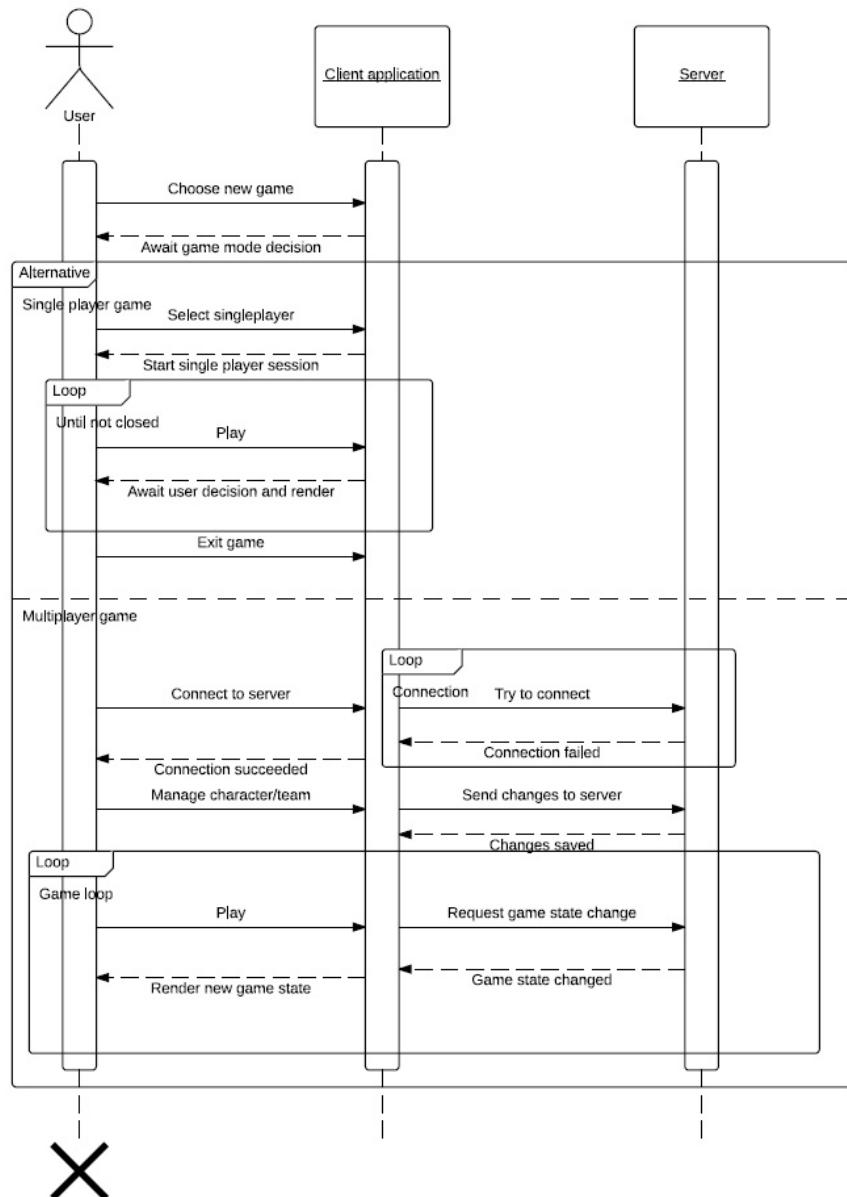
Diagramy sekwencji pokazują przepływ sterowania przez moduły programu oraz jak przepływają dane pomiędzy nimi. Programista dzięki tym diagramom potrafi odtworzyć łatwość w tworzeniu metod aby zachować opisany przepływ danych i sterowania.



Rysunek 11 - Diagram sekwencji dla logowania i rejestracji

Użytkownik najpierw próbuje zalogować się do aplikacji, co odbiera poprawnie aplikacja kliencka i przygotowuje odpowiednie żądanie dla serwera. Serwer odpowiada aplikacji, w zależności od odpowiedzi istnieją dwa przypadki rozpatrzenia sytuacji.

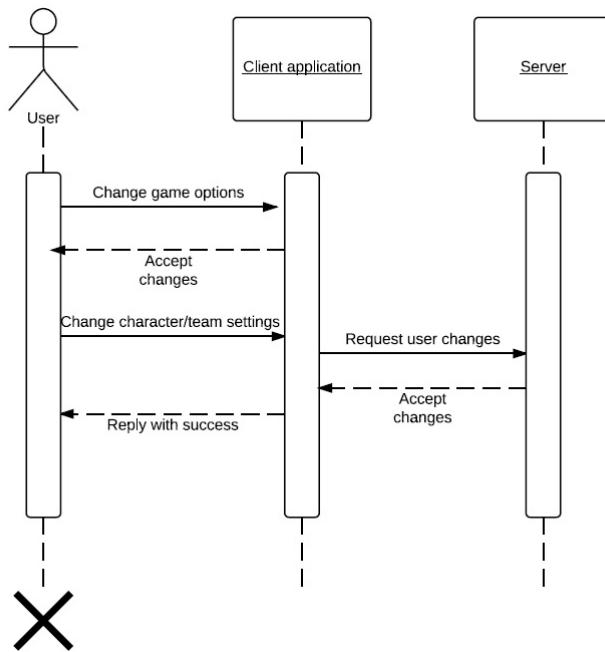
1. Użytkownik nie został znaleziony, wtedy użytkownik dostaje możliwość rejestracji na serwis. Jeśli uda mu się pomyślnie wypełnić formę rejestracyjną, aplikacja kliencka wysyła żądanie o dodanie nowego użytkownika, jeśli z jakiegoś powodu serwer odrzuci prośbę rejestracji, gracz może albo ponowić próbę lub wyjść z gry. Po pomyślnym zarejestrowaniu się użytkownik dostaje możliwość korzystania z gry na nowym koncie.
2. Użytkownik został rozpoznany i dostał możliwość korzystania z gry.



Rysunek 12 - Diagram sekwencji dla rozpoczęcia gry

Gracz początkowo wybiera nowy tryb gry, na co oczekuje aplikacja kliencka. W zależności który tryb gry wybierze klient sterowanie przeprowadzone jest w inny sposób.

1. Jeśli wybierze tryb pojedynczego gracza, aplikacja kliencka nie łączy się z serwerem a tworzy nową sesję gry single player i wykonuje pętlę gry (game loop) w której dostosowuje środowisko gry do decyzji gracza.
2. Jeśli wybrany zostanie tryb wieloosobowy aplikacja kliencka łączy gracza z serwerem i po przygotowaniu swojej postaci bądź drużyny gracz rozpoczyna rozgrywkę sieciową.

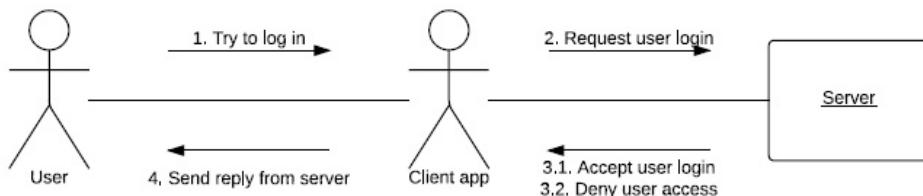


Rysunek 13 - Diagram sekwencji dla dostosowania gry

Gracz może dostosować pod siebie zarówno opcje gry jak i zmienić ustawienia swoich postaci i drużyny. W przypadku zmiany opcji gry nie dochodzi do połączenia z serwerem i zmiany są zapisywane lokalnie. W przypadku zmiany opcji drużyny bądź postaci aplikacja kliencka łączy się z serwerem aby zapisać nowe zmiany wprowadzone przez użytkownika.

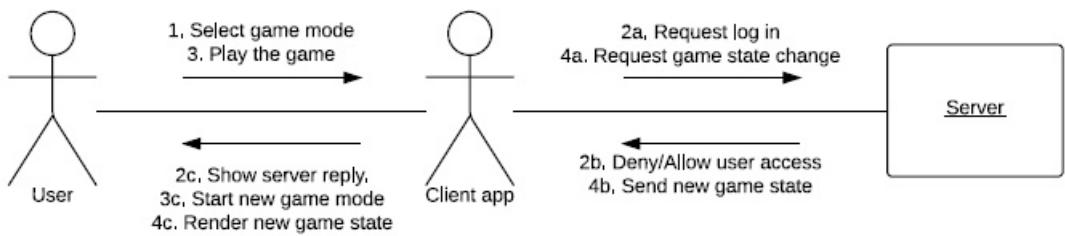
2.6. Diagramy komunikacji dla ważniejszych funkcji programu

Diagramy komunikacji obrazują jak wygląda komunikacja między elementami systemu dla danej akcji. W tych diagramach zostało założone że aplikacja kliencka jest aktorem.



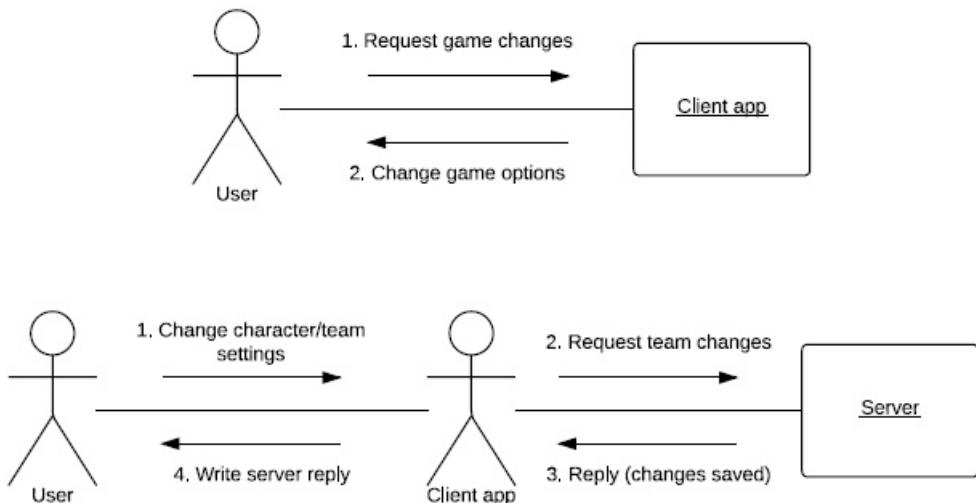
Rysunek 14 - diagram komunikacji dla logowania

Klient przesyłając formularz logowania do klienta komunikuje się z aplikacją kliencką. Następnie zostaje stworzone żądanie logowania w programie które zostaje wysłane do serwera, ten po odebraniu żądania wysyła odpowiedź z powrotem do aplikacji. Zostaje wyświetlony komunikat w zależności od odpowiedzi serwera.



Rysunek 15 - diagram komunikacji dla rozpoczęcia gry

Gracz komunikuje się z aplikacją poprzez wybranie trybu gry, w przypadku trybu multi player (podpunkty a,b i c) aplikacja ta łączy się z serwerem aby zalogować użytkownika. Serwer wysyła odpowiedź do programu, po czym zostaje ona wyświetlona dla użytkownika.

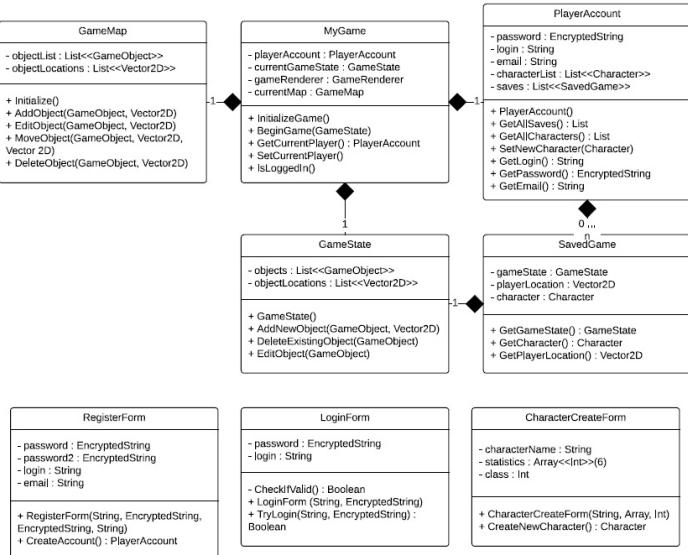


Rysunek 16 - diagram komunikacji dla dostosowania aplikacji

Gracz może zażądać zmian ustawień gry bądź jego postaci lub drużyny. W przypadku zmian opcji gry komunikacja odbywa się tylko z aplikacją gry. W przypadku zmian postaci lub drużyny gracz wprowadza zmiany w aplikacji klienckiej, a ta przesyła informację o zmianie do serwera który wprowadza je.

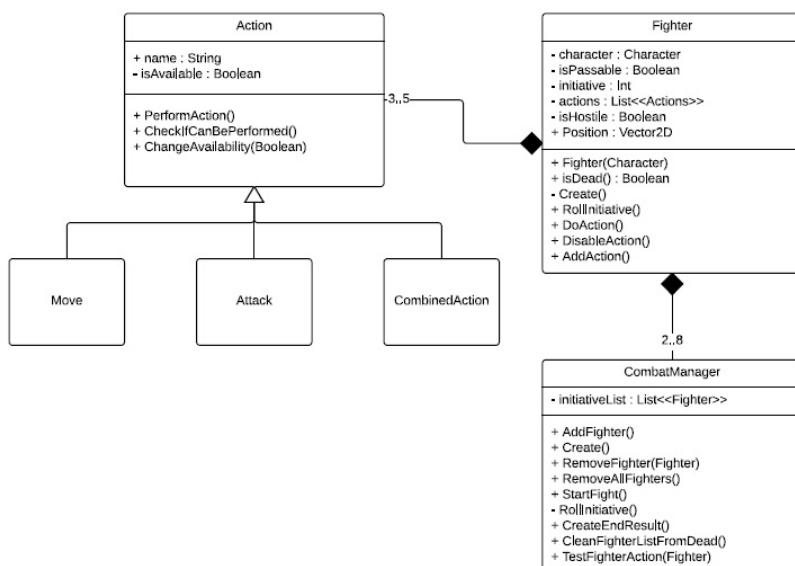
2.7. Diagramy klas

Diagramy klas pokazują podział funkcji systemu na poszczególne klasy, które odpowiadają ich odpowiednikom podczas implementacji systemu. Programista powinien śledzić te diagramy aby utworzyć żądaną strukturę programu.



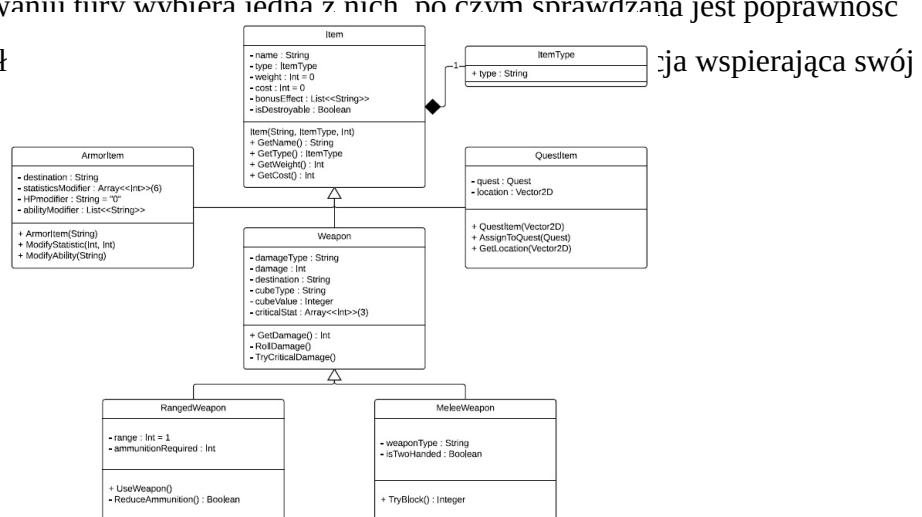
Rysunek 17 - Diagram klas dla mechanizmu gry

Klasa MyGame obsługuje cały mechanizm gry, zarządzając położeniem obiektów na mapie oznaczona klasą GameMap oraz rysowaniem ich na ekranie za pomocą renderera wbudowanego w pakiet Unity. Klasa ta może też sprawdzić czy obecny gracz ma możliwość obsługi gry i czy jest zalogowany. Sam gracz jest oznaczony klasą PlayerAccount, posiada swoje hasło, login, adres e-mail, listę postaci (mogą być zarówno lokalne bądź w trybie online) oraz listę zapisanych stanów gry. Stan gry (GameState) jest zapisany w tej liście wraz z lokalizacją gracza oraz postacią wykorzystaną na mapie. Sam stan gry to lista obiektów oraz ich lokalizacji na mapie. Można dodawać oraz usuwać obiekty z mapy oraz je edytować. Do komunikacji z serwerem klient wykorzystuje formy do kolejno: rejestracji, logowania oraz tworzenia postaci. Formy te są szyfrowane i bezpiecznie wysyłane na serwer oczekując odpowiedzi od serwera.



Rysunek 18 - Diagram klas dla wydarzeń podczas walki w grze

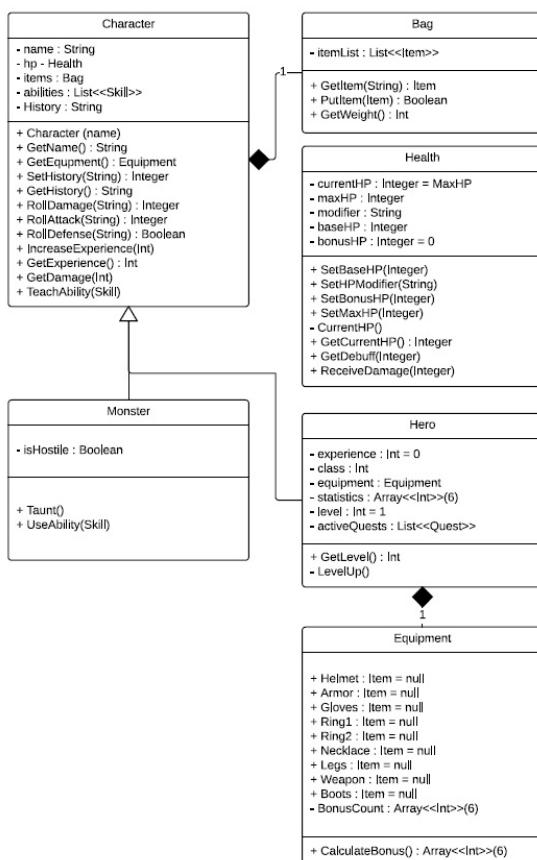
Walka startowana jest przez klasę CombatManager zawierającą listę wojowników (Fighters), zarządza listą priorytetów w walce oraz decyduje o tym, jaki rezultat jest stworzony przez dane akcje. Klasa może zarządzać wojownikami, czyścić pole walki z martwych wojowników, tworzyć rezultat walki i wyświetlać go. Wojownik zawiera daną postać, sprawdza czy można przejść przez nią, czy jest sojusznikiem bądź wrogiem. Każdy wojownik zawiera też inicjatywę – obliczaną ze statystyk oraz umiejętności postaci oraz listę akcji które może wykonać. Wojownik przy wykonywaniu turnu wybiera jedną z nich po czym sprawdzana jest poprawność akcji. Akcja jest to przykład zespołu.



Rysunek 19 - Diagram klas przedmiotów używanych w grze

Klasa Item jest ogólnym pojęciem przedmiotu, każdy obiekt poza terenem, postaciami, przeszkodeami i punktami kluczowymi zadań są przedmiotami. Przedmiotem może być na przykład mikstura życia regenerująca część życia użytkownikowi. Zagwarantowane jest to przez bonusowy efekt jako pole w tej klasie. Każdy przedmiot ma swoją nazwę, wagę, koszt oraz typ. Przedmioty zadaniowe (QuestItem) nie mogą być zniszczone, posiadają ich cel podany przez wektor dwuwymiarowy oraz zadanie do którego należą. Przedmiot który jest swojego rodzaju pancerzem (ArmorItem) posiada miejsce w którym powinno się znaleźć (hełm, pancerz, nogawice, buty, amulet, dwa pierścienie, rękawice). Każdy przedmiot tego typu może modyfikować statystyki gracza, jego ilość punktów życia. Rzadsze przedmioty mają też możliwość edycji umiejętności gracza. Bronie podzielone są na długodystansowe oraz bronie do walki wręcz. Każda broń ma swoje obrażenia oraz szansę na trafienie krytyczne (obliczanie przez kość).

Każda broń ma szansę na nie, obrażenia wywołane przez nie może się różnić w zależności od broni (zwykle jest to 150% bazowych obrażeń). Bronie dystansowe posiadają zasięg, mogą być używane przez postaci nie znajdujące się przy przeciwniku. Używają też amunicji, jeśli gracz nie posiada takiej ten typ broni jest zablokowany. Broń do walki wręcz może być jednoręczna oraz dwuręczna. Modyfikuje to jej obrażenia. Można też zablokować cios przeciwnika jeśli obliczony rzut kostki będzie korzystny.



Postaci (Character) w tej grze posiadają własną nazwę, ilość punktów życia, przedmioty przy sobie, znane przez nie umiejętności oraz historię opisaną przez tekst. Postać ma też swoją bazową wartość obrażeń, którą może wylosować

przez rzut kostką sześcienną. Postać może też zdobywać punkty doświadczenia, dla zwykłych postaci nie mają one znaczenia jednak dla bohaterów (Hero) mają one duże znaczenie. Worek (Bag) to przedmioty posiadane przez gracza. Worek ma limit maksymalnie 20 przedmiotów jednocześnie lub określonego udźwigu postaci określonego przez statystykę Strength. Życie (Health) to ilość punktów postaci, determinowana przez bonusy napływające z opancerzenia postaci, umiejętności bądź statystyki Constitution.

Rysunek 19 - Diagram klas przedmiotów używanych w grze

Po otrzymaniu obrażeń postać może dostać modyfikator życia w postaci mniejszej ilości zadawanych obrażeń, otrzymywania silniejszych ciosów bądź zmniejszonej inicjatywy. Postaci dzielą się na dwa typy:

1. Potwory, są one sterowane przez boty wbudowane w grę. Mają swoje bazowe umiejętności, określoną ilość obrażeń. Nie każdy potwór musi być wrogiem, z reguły jednak są wrogo nastawione do gracza. Za ich zabicie gracz może zdobyć przedmioty należące do ich worka.
2. Bohaterów, jest to typ postaci kierowany przez gracza. Bohater ma swój ekwipunek, klasę postaci do której należy, statystyki rozdzielane od początku gry. Każdy bohater ma swoją ilość punktów doświadczenia która wraz z wykonywaniem zadań, wygrywaniem walk bądź uczestnictwie w losowych wydarzeniach wzrasta i może osiągać nowe poziomy. Bohater posiada też listę zadań do wykonania, zostają one usunięte z listy po wykonaniu. Ekwipunek gracza składa się z broni oraz maksymalnie ośmiu unikalnych typowo przedmiotów pancerzowych.

1.2 Baza danych.

Baza danych to zbiór danych zapisanych zgodnie z określonymi regułami. W naszym przypadku to dane cyfrowe zapisywane zgodnie z zasadami przyjętymi dla danego programu komputerowego specjalizowanego do gromadzenia i przetwarzania danych.

1.2.1 Relacyjne bazy danych.

W bazach relacyjnych wiele tablic danych może współpracować ze sobą (są między sobą powiązane). Bazy relacyjne posiadają wewnętrzne języki programowania, wykorzystujące zwykle SQL do operowania na danych, za pomocą których tworzone są zaawansowane funkcje obsługi danych. Relacyjne bazy danych (jak również przeznaczony dla nich standard SQL) oparte są na kilku prostych zasadach:

- 1.Wszystkie wartości danych oparte są na prostych typach danych.
- 2.Wszystkie dane w bazie relacyjnej przedstawiane są w formie dwuwymiarowych tabel(w matematycznym żargonie noszących nazwę „relacji”). Każda tabela zawiera zero lub więcej wierszy i jedną lub więcej kolumn („atrybutów”). Na każdy wiersz składają się jednakowo ułożone kolumny wypełnione wartościami, które z kolei w każdym wierszu mogą być inne.
- 3.Po wprowadzeniu danych do bazy, możliwe jest porównywanie wartości z różnych kolumn, zazwyczaj również z różnych tabel, i scalanie wierszy, gdy pochodzące z nich wartości są zgodne. Umożliwia to wiązanie danych i wykonywanie stosunkowo złożonych operacji w granicach całej bazy danych.
- 4.Wszystkie operacje wykonywane są w oparciu o algebrę relacji, bez względu na położenie wiersza tabeli. Wiersze w relacyjnej bazie danych przechowywane są w porządku zupełnie

dowolnym – nie musi on odzwierciedlać ani kolejności ich wprowadzania, ani kolejności ich przechowywania.

5.Z braku możliwości identyfikacji wiersza przez jego pozycję pojawia się potrzeba obecności jednej lub więcej kolumn niepowtarzalnych w granicach całej tabeli, pozwalających odnaleźć konkretny wiersz. Kolumny te określa się jako „klucz podstawowy” (ang. primary key) tabeli.

1.2.2 Klucze.

Klucze to zbiory atrybutów mających określona właściwość. Dzięki nim, możemy jednoznacznie identyfikować każdy pojedynczy wiersz. Znajomość pojęć kluczy podstawowych i obcych jest niezbędna do tworzenia zapytań, odwołujących się do wielu tabel. Dalej przedstawię najczęściej spotykane typy klucze.

Superklucz to dowolny podzbiór atrybutów, identyfikujący jednoznacznie każdy wiersz. Każda RELACJA (tabela) może zawierać wiele takich kluczy. Szczególnym przypadkiem jest superklucz składający się ze wszystkich atrybutów (kolumn) danej tabeli.

Klucz kandydujący to dowolny z superkluczy, mogący zostać kluczem podstawowym. W implementacji bazy danych, w praktyce nie istnieje jako niezależny osobny (zmaterializowany byt) jako taki. Jest to tylko założenie teoretyczne.

Klucz podstawowy(primary key) to wybrany (zazwyczaj najkrótszy), jednoznacznie identyfikujący każdy, pojedynczy wiersz, zbiór atrybutów (kolumn) danej relacji (tabeli). Jest to pierwszy z wymienionych do tej pory kluczy, który ma faktyczne, fizyczne odwzorowania w implementacji bazy danych. Każda tabela może mieć tylko jeden taki klucz.

Klucz obcy to atrybut lub zbiór atrybutów, wskazujący na klucz główny w innej relacji. Klucz obcy to nic innego jak związek, relacja między dwoma tabelami. Definicja klucza obcego, pilnuje aby w tabeli powiązanej, w określonych atrybutach, znaleźć się mogły tylko takie wartości które istnieją w tabeli docelowej jako klucz główny. Klucz obcy może dotyczyć również tej samej tabeli.

1.2.3 Powiązania pomiędzy tabelami.

Związek 1:1(ang. *One to one*) - każdy wiersz z tabeli A może mieć tylko jednego odpowiednika w tabeli B (i na odwrót). Ten rodzaj relacji może być postrzegany jako podzielenie tabeli na dwie (bo relacja jest jeden do jeden). Stosowany np. wtedy, gdy zbiór dodatkowych atrybutów jest określony tylko dla wąskiego podzbioru wierszy w tabeli podstawowej.

Związek 1:N(ang. *One to many*) - jest to najczęściej spotykana relacja. Określamy w niej że każdy element ze zbioru A (wiersz tabeli A), może być powiązany z wieloma elementami zbioru B.

Związek N:M(ang. *Many to many*) - realizowana jest zawsze jako dwie relacje 1:N. Zatem jeśli chcemy między dwoma tabelami zamodelować związek N:M **potrzebujemy trzecią tabelę – łącznikową**.

1.2.4 MySQL.

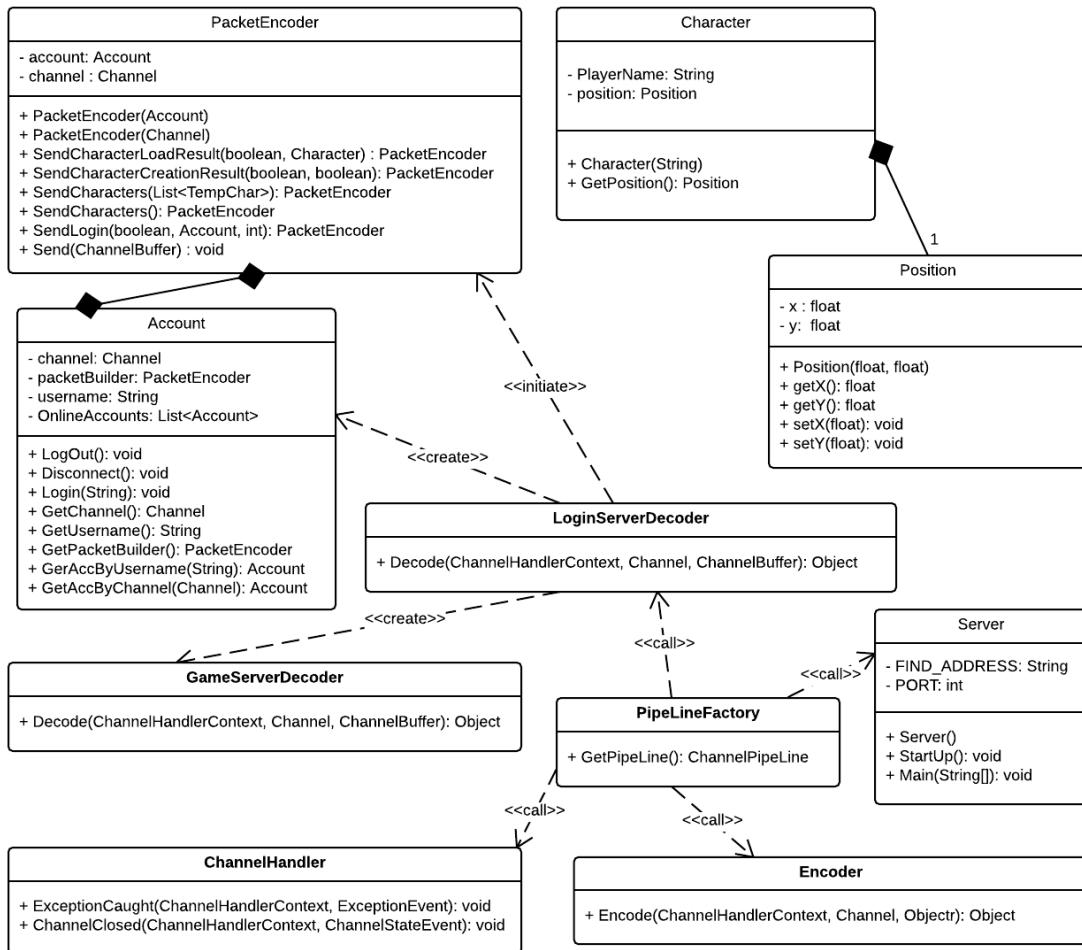
MySQL – wolnodostępny system zarządzania relacyjnymi bazami danych. MySQL rozwijany jest przez firmę Oracle. Wcześniej przez większość czasu jego tworzeniem zajmowała się szwedzka firma MySQL AB. MySQL AB została kupiona 16 stycznia 2008 roku przez Sun Microsystems, a ten 27 stycznia 2010 roku przez Oracle. W międzyczasie Monty Widenius (współtwórca MySQL) stworzył MariaDB – alternatywną wersję opartą na licencji GPL. MariaDB jest oparta na tym samym kodzie bazowym co MySQL i dąży do utrzymania kompatybilności z jej poprzednimi wersjami.

2. Model części serwerowej

2.1. Opis serwera

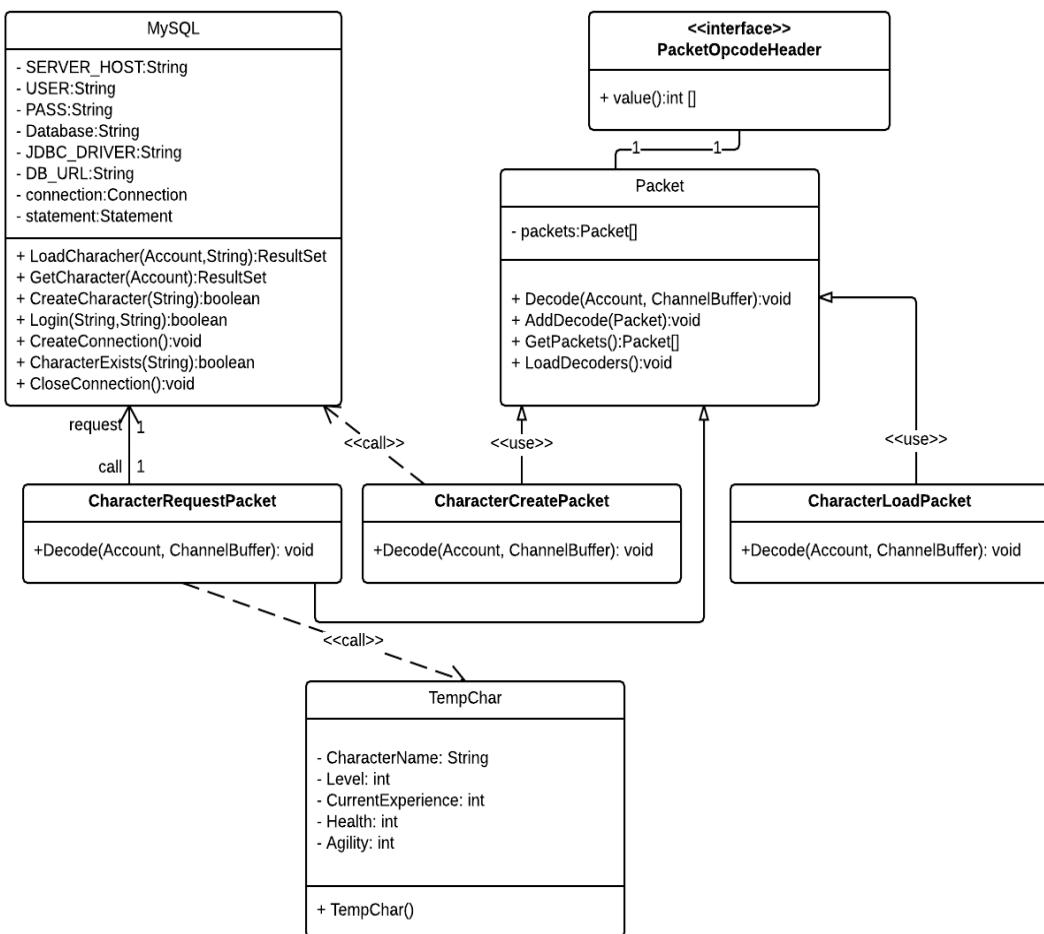
Serwer w Teiru MMO powinien pełnić kilka bardzo ważnych funkcji. Przedewszystkim powinien pozwalać użytkownikom się logować/rejestrować , walczyć i wspólnie swobodnie zwiedzać świat gry. Niżej , za pomocą diagramów spróbuje przedstawić to jak działa serwer.

2.1.1 Diagram klas – serwer



Rys. 1: Diagram klas - część 1

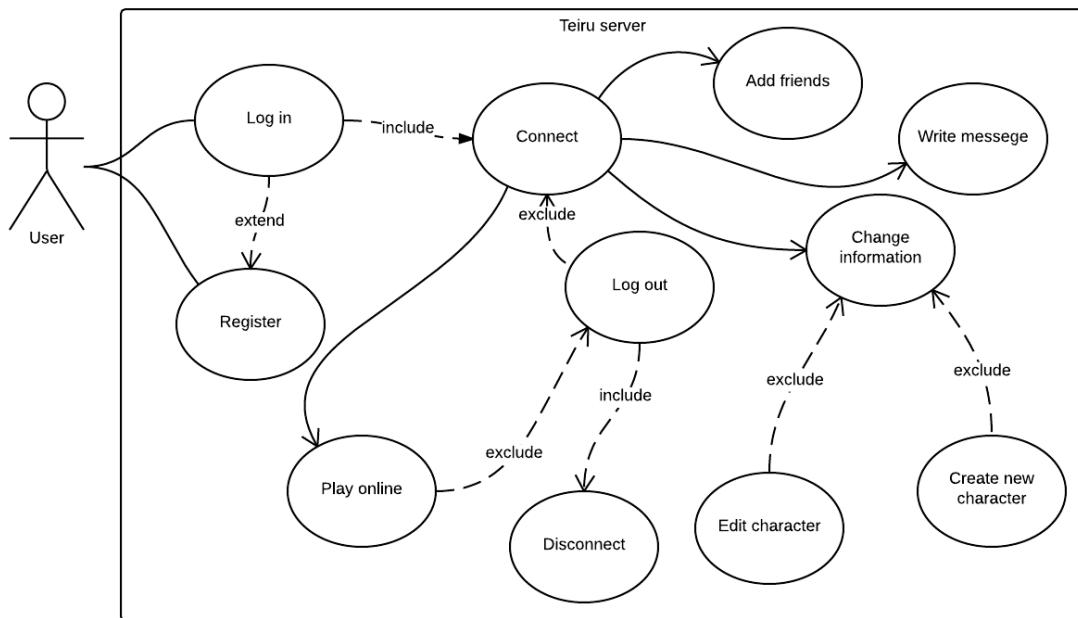
Powyżej została przedstawiona część diagramu klas odpowiadająca za logowanie. Serwer ciągle oczekuje na żądania użytkowników . Kiedy klient wysyła żądania serwer odszyfrowuje otrzymany komunikat i sprawdza w bazie danych wprowadzoną przez użytkownika informacje. Proces rejestracji odbywa się podobnie , jedyną różnicą jest to że serwer sprawdza dostępność wprowadzonego identyfikatora użytkownika(ang. Username) i wprowadzonej poczty elektronicznej(ang. Email). Po udanym zalogowaniu, żądania użytkownika przestaje przyjmować i opracowywać LoginServerDecoder i zaczyna GameServerDecoder, który odpowiada już za proces gry.



Rys. 2: Diagram klas - część 2

Obsługa gracza podczas gry w większej części odbywa się po stronie klienta. Serwer tylko odpowiada na żądania klientów, aktualizuje i przechowuje dane , takie jak pozycja każdej postaci gracza, poziom, punkty wytrzymałości i inne.

2.1.2 Diagram przypadków użycia – serwer



Rys. 3: Diagram przypadków użycia

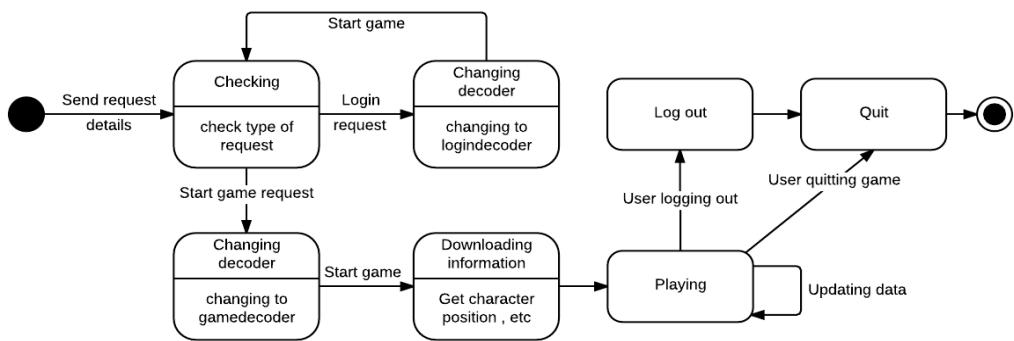
Jak już pisałem wcześniej diagram przypadków użycia opisuje system z punktu widzenia użytkownika. Na diagramie możemy zobaczyć dostępne dla użytkownika działania związane z serwerem. Następne kilka scenariuszy przypadków użycia przedstawiają parę możliwych dla użytkownika scenariuszy działania.

Nazwa: Zaloguj się Aktorzy: Niezalogowany użytkownik	Główny scenariusz: <ol style="list-style-type: none"> Użytkownik wypełnia formularz logowania. Użytkownik potwierdza wpisane dane. System sprawdza otrzymane dane. Użytkownik uzyskuje dostęp do swojego konta. Rozszerzenia: <ol style="list-style-type: none"> Nieodpowiednie dane(zabronione znaki, zbyt krótkie hasło , etc)
-----------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>a. Użytkownik wpisuje poprawne dane.</p> <p>b. Użytkownik potwierdza wpisane dane.</p> <p>4a. Nieprawidłowa nazwa użytkownika lub hasło.</p> <p>a. Użytkownik ponownie wpisuje dane.</p> <p>b. System sprawdza otrzymane dane.</p>
<p>Nazwa: Stwórz nową postać</p> <p>Aktorzy: Zalogowany użytkownik</p>	<p>Główny scenariusz:</p> <ol style="list-style-type: none"> 1. Użytkownik poprawnie się loguje 2. Użytkownik wybiera opcję „Create new character” 3. Użytkownik wybiera klasę postaci i rozdaje dostępne punkty umiejętności. 4. Użytkownik potwierdza swój wybór. 5. Nowa postać zostaje stworzona i odpowiednia informacja zapisana w bazie danych. <p>Rozszerzenia:</p> <ol style="list-style-type: none"> 1a. Wystąpił problem z logowaniem <ol style="list-style-type: none"> a. Użytkownik ponownie próbuje się zalogować. 2a. Zbyt dużo postaci <ol style="list-style-type: none"> a. Użytkownik usuwa jedną z już istniejących postaci. 5a . Wystąpił problem z połączeniem <ol style="list-style-type: none"> a. Użytkownik zostaje poproszony o ponowne potwierdzenia wyborów.
<p>Nazwa: Edytuj istniejącą postać</p> <p>Aktorzy: Zalogowany użytkownik</p>	<p>Główny scenariusz:</p> <ol style="list-style-type: none"> 1. Użytkownik poprawnie się loguje 2. Użytkownik wybiera jedną ze swoich postaci 3. Użytkownik zmienia dane. 4. Użytkownik potwierdza zmiany. 5. Zmiany zostają zapisane w bazie danych. <p>Rozszerzenia:</p> <ol style="list-style-type: none"> 1a. Wystąpił problem z logowaniem <ol style="list-style-type: none"> a. Użytkownik ponownie próbuje się zalogować. 2a. Nie ma żadnej postaci.

	<p>a. Użytkownik tworzy nową postać (patrz scenariusz „Stwórz nową postać”)</p> <p>5a . Wystąpił problem z połączeniem</p> <p>a. Użytkownik zostaje poproszony o ponowne potwierdzenia zmian</p>
<p>Nazwa: Zacznię grać</p> <p>Aktorzy: Zalogowany użytkownik</p>	<p>Główny scenariusz:</p> <ol style="list-style-type: none"> 1. Użytkownik poprawnie się loguje 2. Użytkownik wybiera tutorial lub online grę 3. Użytkownik zaczyna grać. <p>Rozszerzenia:</p> <ol style="list-style-type: none"> 2a. Użytkownik wybiera tutorial. <ol style="list-style-type: none"> a. Bezzwłocznie zaczyna grę. 2b. Użytkownik wybiera online. <ol style="list-style-type: none"> a. Tworzy lub wybiera nową postać. b. Zaczyna grę.

2.1.3 Diagram stanów - serwer

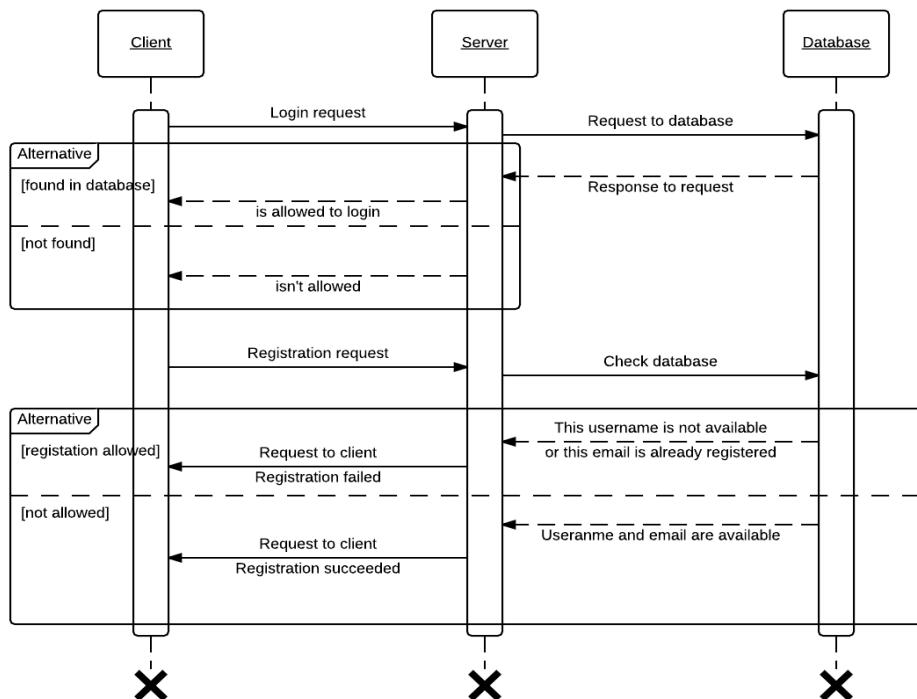


Rys. 4: Diagram stanów

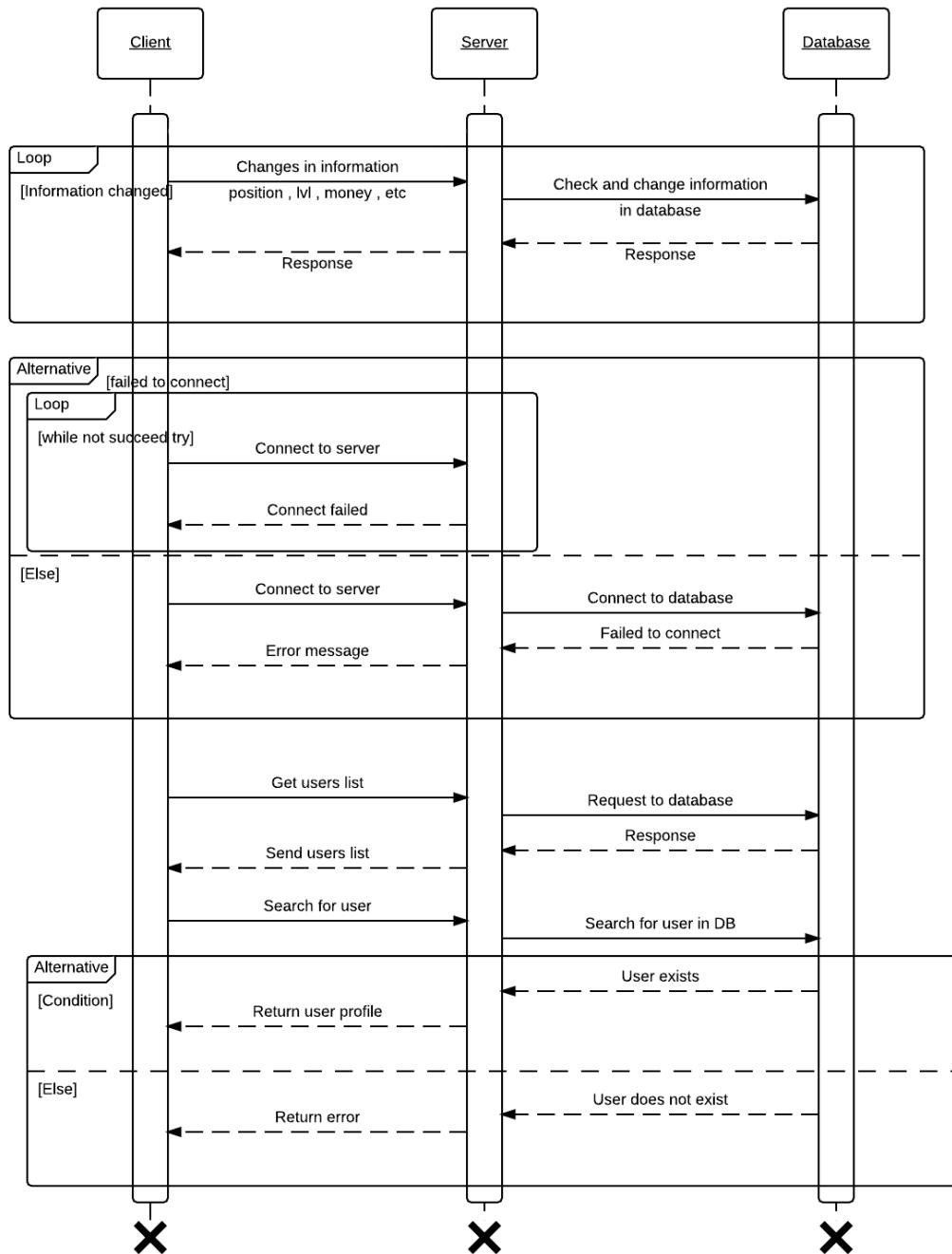
Najlepszym sposobem na scharakteryzowanie tego że w systemie zachodzą zmiany jest to że obiekty systemu zmieniają stan. Zmiana stanów obiektów jest pokazana na Rys.4. Na diagramie są opisane prawie wszystkie możliwe zmiany stanów, które są związane z serwerem.

W zależności od tego czy użytkownik chce się zalogować, czy chce zacząć grę, serwer zmienia swój stan w odpowiedzi na te zdarzenia. Podczas gry żaden obiekt nie zmienia swojego stanu, tylko zaktualizowane dane automatycznie zapisują się do bazy danych.

2.1.4 Diagram sekwencji - serwer



Rys. 5: Diagram sekwencji - Logowanie i rejestracja, część 1



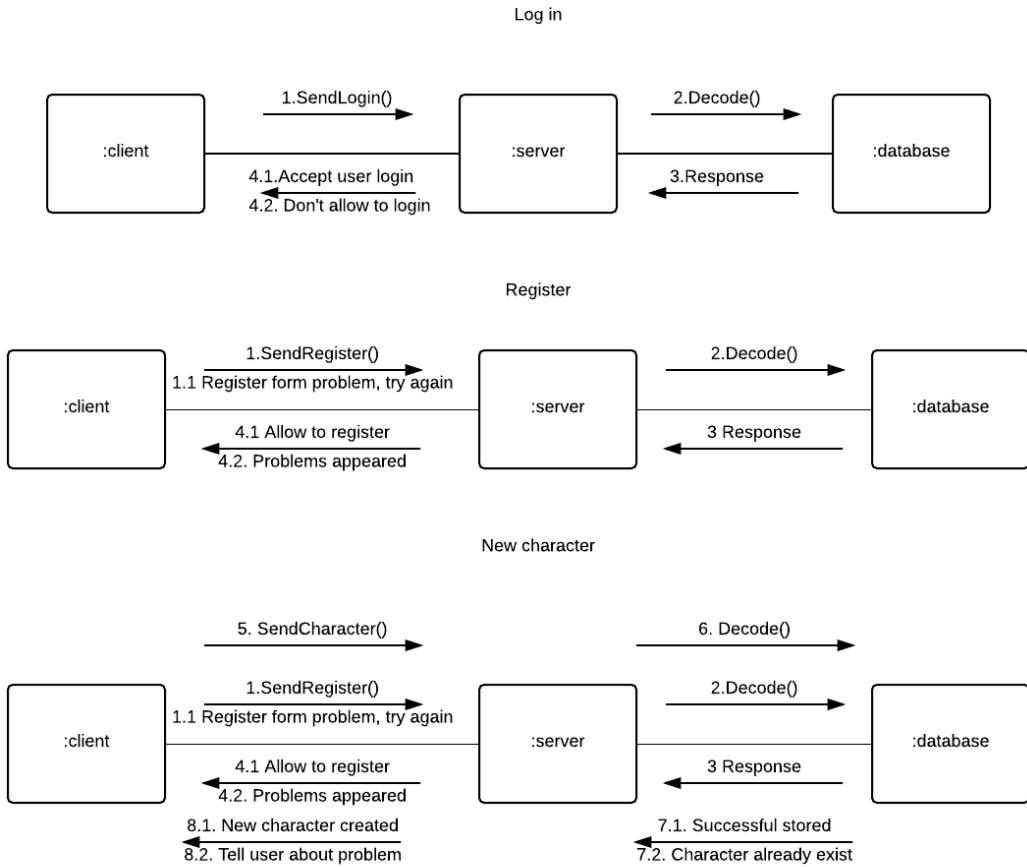
Rys. 6: Diagramy sekwencji - łączenie się z bazą danych, zmiana informacji i dodawanie przyjaciół, część 2

Diagram sekwencji(przebiegu) pokazuje jak , w zależności od czasu, odbywa się komunikacja między obiektami. Na przedstawionym powyżej diagramie opisane są komunikacje odbywające się między klientem , serwerem i bazą danych podczas logowania i rejestracji. Diagram opisuje to co już było wcześniej rozpatrzone, ale pod innym kątem co pozwala lepiej zrozumieć działanie systemu(w tym przypadku serwera).

Diagram przedstawiony na Rys.6 przedstawia komunikacje odbywające się podczas edytowania

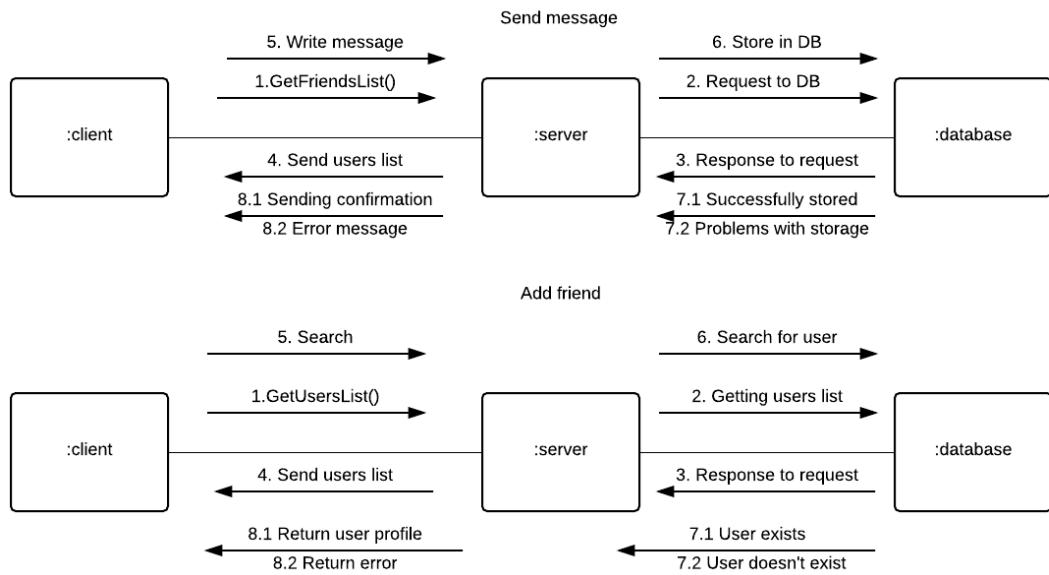
informacji, łączenia się z serwerem i dodawania nowych przyjaciół. Łączenie się z serwerem i bazą danych odbywa się bezpośrednio, po każdym niepowodzeniu występuje próba ponownego łączenia się z serwerem/bazą danych. Po pewnym czasie klient-serwer przestaje próbować się połączyć z serwerem/bazą , co nie pozwala programowi się zapętlić. Dodając przyjaciół użytkownik wyszukuje ich wpisując imię użytkownika, po czym serwer zwraca wynik wyszukiwania. Jeśli użytkownik o takim imieniu użytkownika istnieje to może być dodany do przyjaciół

2.1.5 Diagram komunikacji - serwer



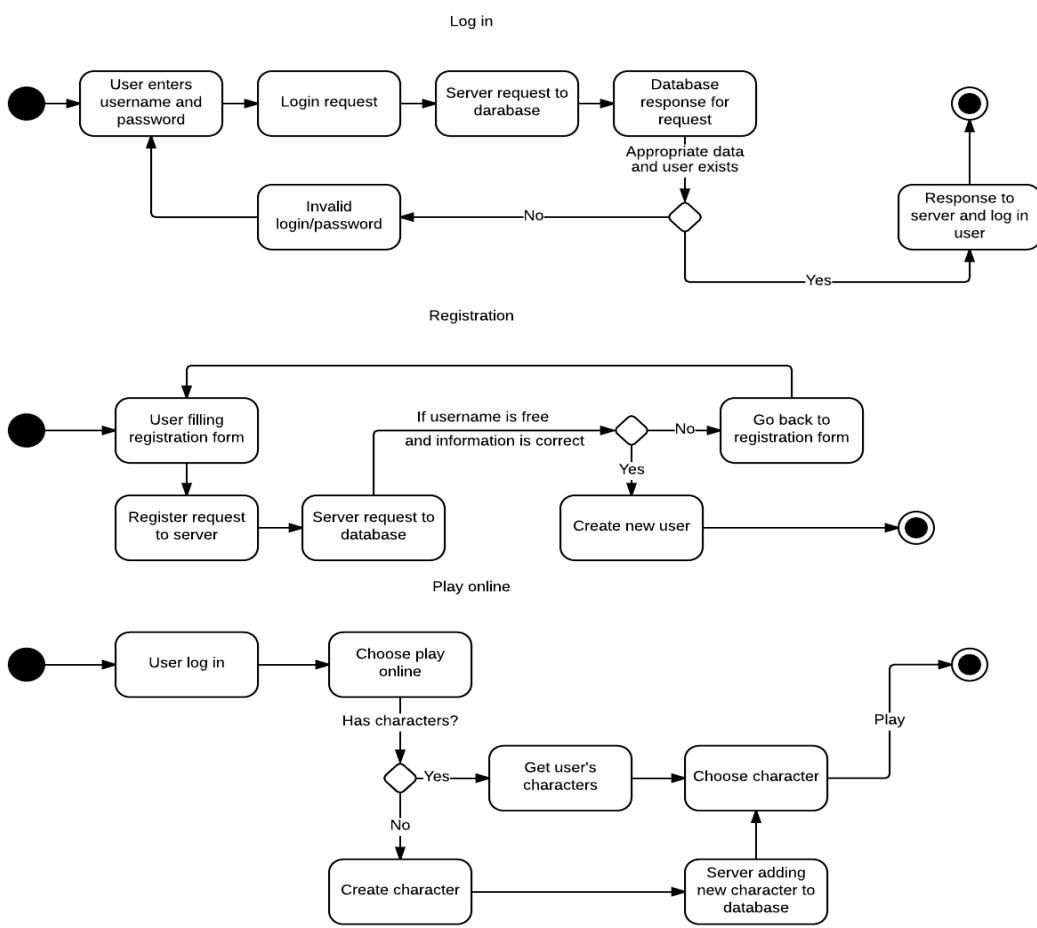
Rys. 7: Diagramy komunikacji - logowanie , rejestracja i tworzenie nowej postaci, część 1

Diagram komunikacji oprócz powiązań między obiektami pokazuje jeszcze komunikaty przesyłane między nimi. Diagramy te pozwalają na sortowanie komunikatów otrzymywanych przez obiekt. Na diagramach powyżej możemy zobaczyć jak obiekty komunikują się między sobą podczas logowania , rejestracji lub tworzenia nowej postaci. Wszystkie komunikaty są oznaczone liczbami w kolejności ich przesyłania między obiektami. Najwięcej komunikatów widzimy między klientem, a serwerem, gdyż baza danych uczestniczy w opisanych komunikacjach tylko poprzez przesyłanie danych do serwera. Na Rys. 8 przedstawiłem komunikowanie się obiektów podczas wysyłania wiadomości lub dodawania przyjaciół.



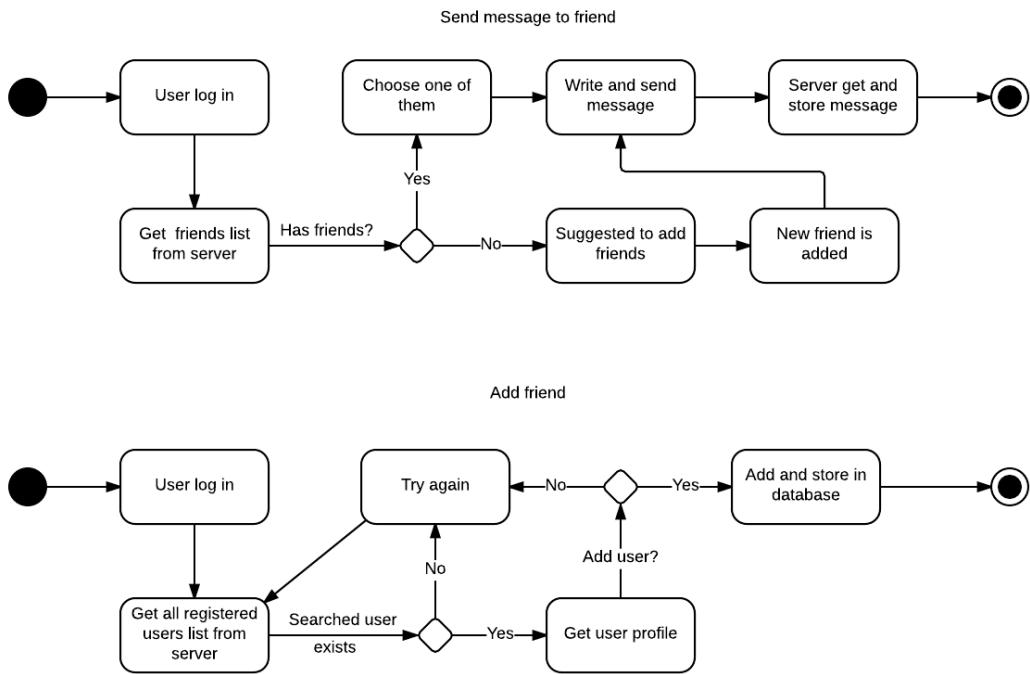
Rys. 8: Diagramy komunikacji - wysyłanie wiadomości i dodawanie przyjaciół, część 2

2.1.6 Diagram czynności – serwer



Rys. 9: Diagramy czynności - część 1

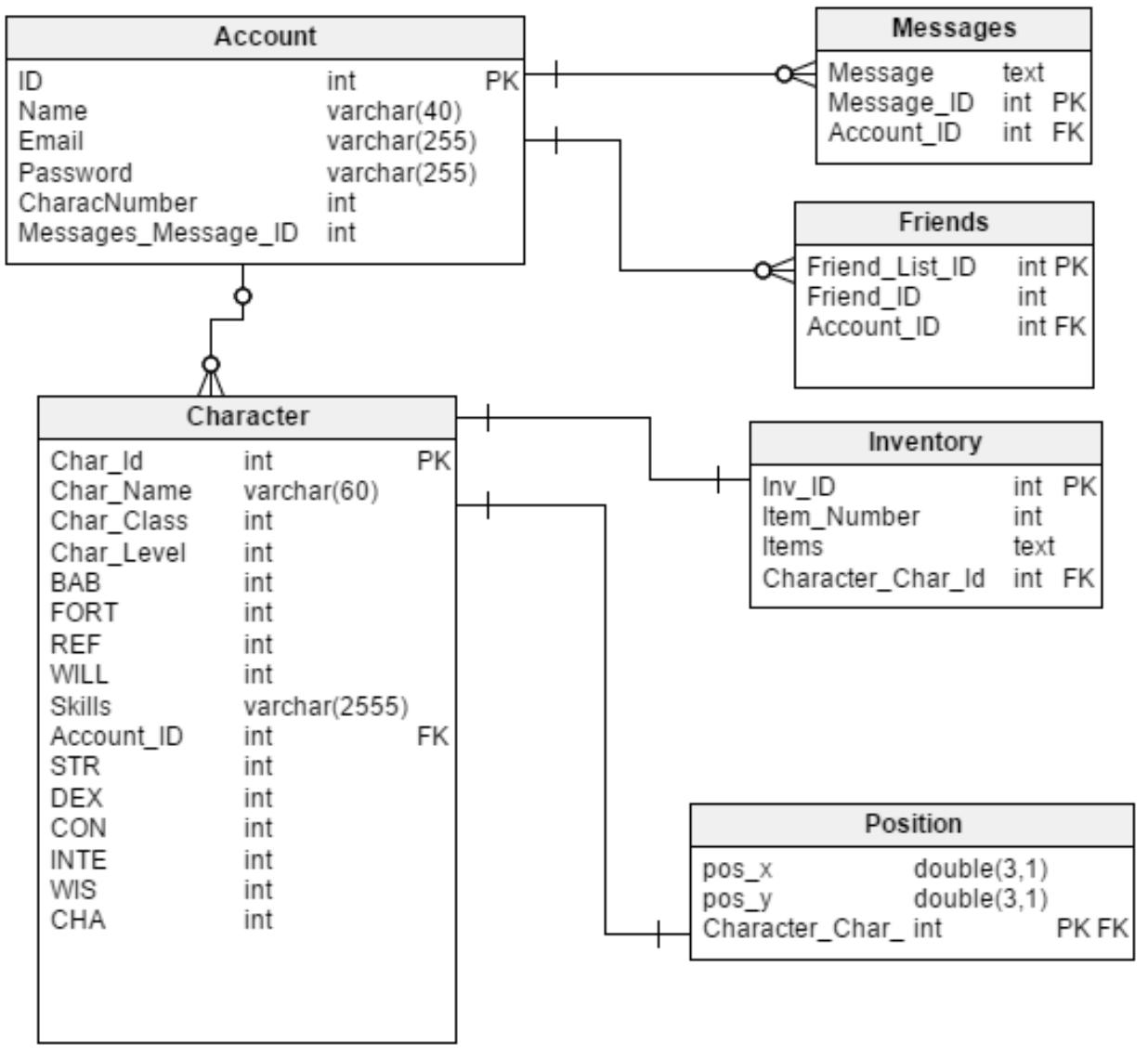
Diagram czynności opisuje kroki, punkty rozgałęzienia i podejmowania decyzji. Diagram czynności to rozwinięcie diagramu stanów, gdzie czynności były opisane tylko napisami na strzałkach. Więc dla lepszego zrozumienia tego jak działa system, diagramy czynności są niezbędne.



Rys. 10: Diagramy czynności - część 2

Na diagramach powyżej są opisane rejestracja, logowanie i gra. Możemy zobaczyć jakie decyzje może podjąć użytkownik i jakie możliwości on otrzymuje po podjęciu tej czy innej decyzji. Na Rys. 10 opisałem wysyłanie wiadomości do przyjaciół i dodawanie nowych przyjaciół. Tak jak potrzebne dane(wszyscy zarejestrowani użytkownicy i lista przyjaciół użytkownika) są zapisane w bazie danych to każdy raz klient otrzymuje wszystkie dane z serwera.

2.2 Model bazy danych



Rys. 11: Model bazy danych

W bazie przechowujemy informację o koncie użytkownika. Razem z tym utrzymujemy w bazie informację o otrzymanych i wysłanych wiadomościach i listę przyjaciół użytkownika. Jak widać na Rys. 11 każdy użytkownik ma postać . Przechowujemy w bazie informacje o postaciach (charakterystyki, pozycję i przedmioty postaci). W bazie przechowujemy tylko niezbędną informację co pozwala bazie obsługiwać więcej żądań jednocześnie(tak jak część informacji zostaje po stronie klienta i klient nie musi łączyć się z bazą dla jej otrzymania).

Rozdział 5

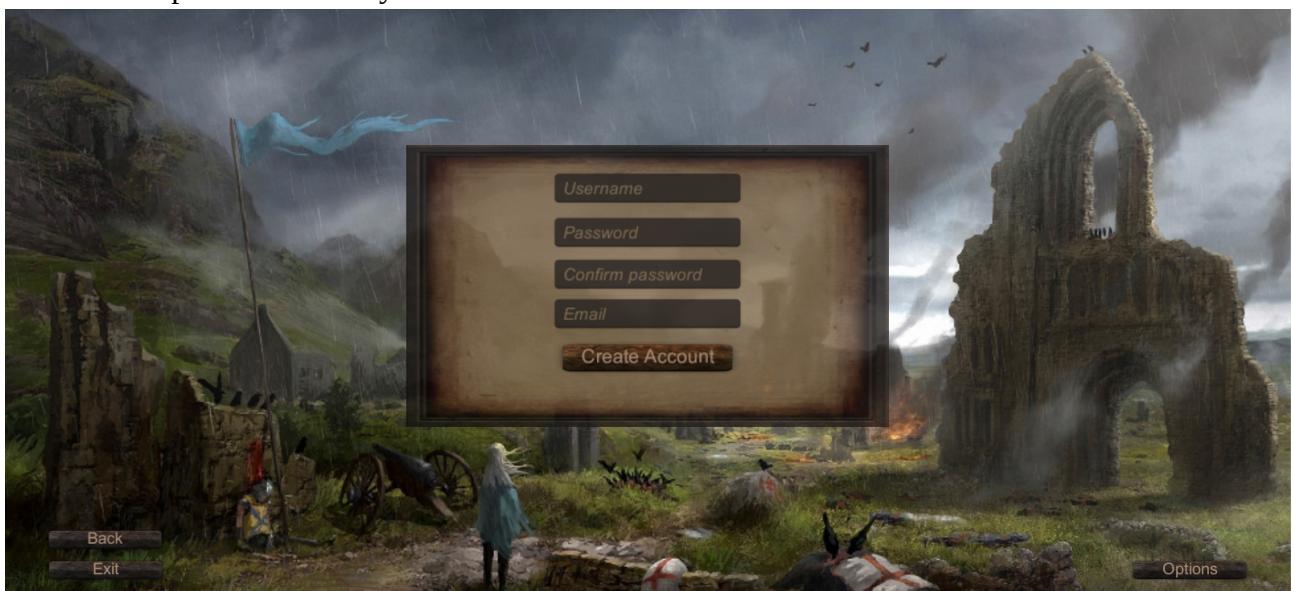
Screeny i opisy.

W tym rozdziale zaprezentuje efekt końcowy projektu w postaci screenów stworzone interfejsu użytkowego wraz z opisami.

5.1 Dostępność do programu.

5.1.1 Rejestracja

Użytkownik, który chce mieć dostęp do gry musi się zarejestrować, jest to wymagane do użytkowania programu. Aby proces rejestracji zakonczył się powodzeniem, gracz musi uzupełnić formularz odpowiednimi danymi



*Rys. * Ekran Rejestracji*

Username – nazwa konta gracza, musi być unikalna, więc w razie zaistnienia już takowej w bazie danych, użytkownikowi zostanie zwrócony komunikat błędu

Password – Hasło, strzegące dostępu do konta gracza, dla potwierdzenia, że użytkownik się nie pomylił, musi powtórnie je wpisać w polu *Confirm password*.

Email – Adres email, wymagany, aby zapobiec tworzeniu przez gracza wielu kont . Też musi być unikalny, dla każdego gracza.

5.1.2 Logowanie

Użytkownik, który posiada już swoje konto, może dokonać procesu logowania i rozpocząć swoją

przygode.



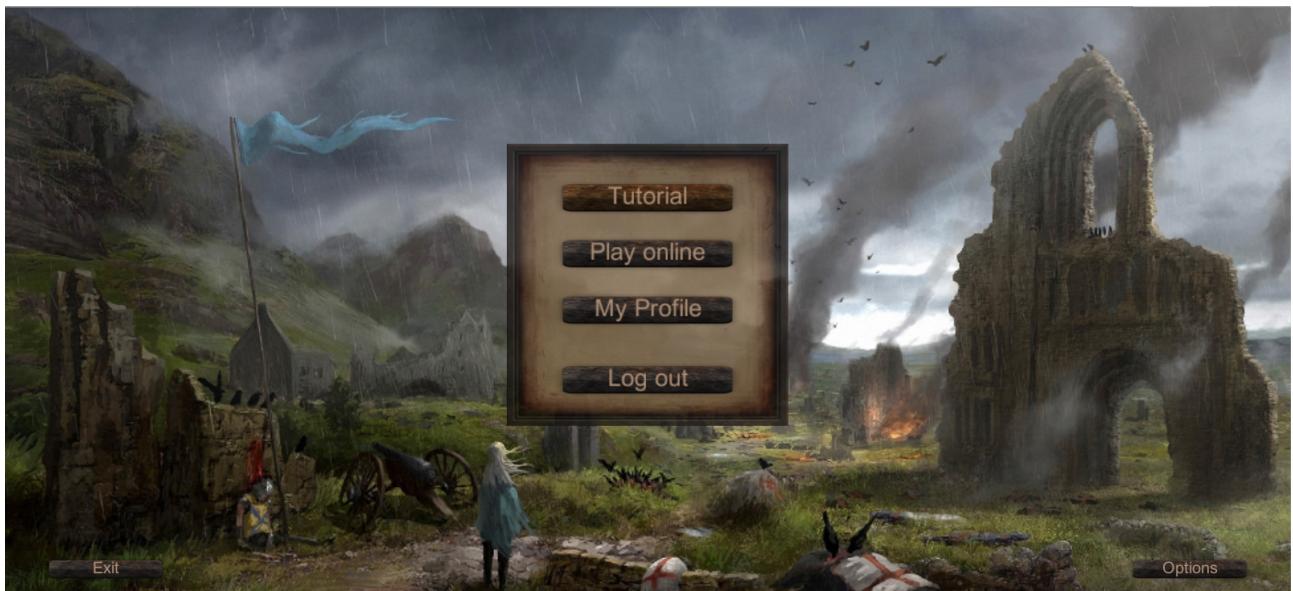
Rys. * Ekran Logowania

Email – wprowadzony uwcześnie przez użytkownika przy procesie rejestracji.

Password – wybrane uwcześnie przez użytkownika, ma zweryfikować, czy gracz loguje się na jego własne konto.

Registration – gdy użytkownik nie posiada jeszcze własnego konta, ten przycisk wysyła go do ekranu z formularzem rejestracji.

5.2 Menu Głównego



Rys. 4 Ekran Menu Głównego

Tutorial – przycisk prowadzi do menu.

Play online – przycisk prowadzi do menu z aktualnymi postaciami gracza, oraz gdzie

może stworzyć nowe.

My Profile – otwiera stronę, gdzie user może zobaczyć informacje o jego profilu, oraz zmienić hasło.

Log out – wylogowywuje gracza z gry.

5.3 Profil gracza.

W tym oknie gracz może zobaczyć dane swojego konta, oraz je zmienić.



Rys. * Ekran profilu gracza.

Username – nazwa konta gracza, musi być unikalna, więc w razie zaistnienia już takowej w bazie danych, użytkownikowi zostanie zwrócony komunikat błędu

Email – Adres email, wymagany, aby zapobiec tworzeniu przez gracza wielu kont. Też musi być unikalny, dla każdego gracza.

Avatar – Ikona która będą widywać gracze np. w oknie kreatora walki.

5.4 Wybór Postaci.



Gracz ma do wyboru, może stworzyć nową postać, bądź ją wybrać z dostępnej listy.

Rys.

* *Ecran wyboru postaci.*

Character Name – ten button przekierowuje gracza do okna kreatora postaci.

„Character's Name” - button z imieniem postaci (już istniejącej) rozpoczyna rozgrywkę właśnie tą postacią.

5.5 Tworzenie Postaci.

Zanim gracz rozpocznie swoją przygodę na multi, musi stworzyć nową postać.



Rys. 4 *Ecran tworzenia postaci.*

Ikony klas – są tą przyciski wyboru klasy, sprawiają że formularz statysyk zostaje

automatycznie uzupełniony.

Formukarz – kolejne elementy odpowiadają za statystyki postaci,

Character's Name – w tym miejscu gracz nadaje imię swojej postaci.