

**UNIWERSYTET WARMIŃSKO-MAZURSKI
W OLSZTYNIE**

WYDZIAŁ MATEMATYKI I INFORMATYKI

Kierunek: Informatyka

Kacper Bańkowski

Łukasz Dąbrowski

**Serwis internetowy wspomagający wyszukiwanie
graczy do drużyny, w grze internetowej Tibia.**

Praca inżynierska wykonana
w Katedrze Centrum Diagnostyki Radiowej
Środowiska Kosmicznego pod kierunkiem
dr hab. Leszka Błaszkiwicza

Olsztyn 2021

**UNIVERSITY OF WARMIA AND MAZURY IN
OLSZTYN**

**FACULTY OF MATHEMATICS AND
COMPUTER SCIENCE**

Field of study: Computer Science, General Information Sciences

Kacper Bańkowski

Lukasz Dąbrowski

**Internet service aiding search for players to team, in
online game Tibia.**

Engineering Thesis written
in Chair of Space Radio-Diagnostics
Research Centre under
supervision of dr hab. Leszek Błaszkiwicz

Olsztyn 2021

Spis treści

1	Wprowadzenie	5
2	Historia gier MMO	7
2.1	Początki.....	7
2.2	Habitat i Neverwinter Nights jako pierwsze gry MMORPG.....	8
2.3	Ultima Online i Everquest.....	10
2.4	Tibia	12
3	Tibia obecnie	14
3.1	Przyrost liczby graczy.....	14
3.2	Popularność gry Tibia w Polsce i na świecie.....	15
4	Proces tworzenia serwisu	17
4.1	Pomysł	17
4.2	Diagram oraz scenariusze przypadków użycia	18
4.3	Wykorzystane narzędzia, biblioteki i ich opis.....	21
4.3.1	ASP.NET Core.....	21
4.3.2	Język programowania	21
4.3.3	Środowisko programistyczne	22
4.3.4	Entity Framework	22
4.3.5	Identity Framework.....	23
4.4	Pierwsze zaprogramowane funkcjonalności	24
4.5	Kontroler tablicy i logika tworzenia ogłoszeń.....	26
4.6	Podział na warstwy serwisu i repozytorium	28
4.7	Model postaci i przypisywanie postaci do konta	30
5	Finalna aplikacja	33
6	Podsumowanie	43
7	Spis rysunków	44
8	Źródła	46
9	Tabele	47

Streszczenie

Celem pracy było stworzenie serwisu internetowego typu LFG (Looking For Group) do gry Tibia, opartego o framework ASP.NET Core w wersji 3.1 w środowisku Visual Studio 2019.

W części teoretycznej opisana została historia gier MMO. Zaprezentowano też ważniejsze gry, wprowadzające rozwiązania, które stały się później wzorem powielanym w kolejnych produkcjach. Opisana jest też sama gra, której dotyczy serwis, jak i jej obecna sytuacja na świecie. Omówiono również wykorzystane technologie, narzędzia, język programowania oraz samo środowisko programistyczne i opisano czym ono jest.

W części praktycznej stworzony i opisany został serwis internetowy. Działanie serwisu zostało opisane, obok przedstawienia graficznego poszczególnych jego części.

Abstract

The aim of the thesis was creating LFG type internet service (Looking For Group) for game Tibia, ASP .NET Core-based in version 3.1 using Visual Studio 2019 programming environment.

Theoretical part describes the history of MMO genre and important examples of games which introduced solutions which later on became a pattern introduced to many bigger games. It also describes the game targeted by service, and it's current situation in world. It also describes technologies, tools, programming language and IDE which have been used and what is it.

Practical part is the creation of service, which is depicted in thesis with help of screenshots. Service's functionalities are described with screenshots showing specific parts.

1 Wprowadzenie

Gry MMORPG- Massively Multiplayer Online Role-Playing game, do którego należy również Tibia, to rodzaj komputerowych gier fabularnych, w których rozgrywka skupia się głównie na rozwijaniu kierowanej przez gracza postaci, jak i eksploracji świata przedstawionego.

Do powstania pierwowzorów dzisiejszych MMO doprowadziła próba przeniesienia papierowych gier RPG (Role playing game) na grunt komputerów. Pierwsze na rynku były więc gry tekstowe, gdzie dokonywane wybory, sprowadzały się do kliknięcia odpowiedniego okna tekstowego. Z czasem na znaczeniu zyskało przedstawienie graficzne podejmowanych przez gracza wyborów, aż w końcu wizualizacja poczynąń gracza stała się trzonem rozgrywki, pozwalając na precyzyjne sterowanie poczynaniami postaci w czasie rzeczywistym.[1]

Największą popularnością cieszą się obecnie takie gry MMORPG jak World of Warcraft, Final Fantasy XIV czy też The Elder Scrolls Online.[2] Ważną kwestią jest to, co gra ma do zaoferowania graczom - im bardziej gry te są rozbudowane, tym łatwiej jest im przyciągnąć nowych graczy. W tego typu tytułach gracze mogą ze sobą współpracować, aby wykonywać kolejne zadania, awansować w kolejnych poziomach doświadczenia i rozwijać dalej umiejętności swojej postaci, jak również możliwa jest rywalizacja między graczami.

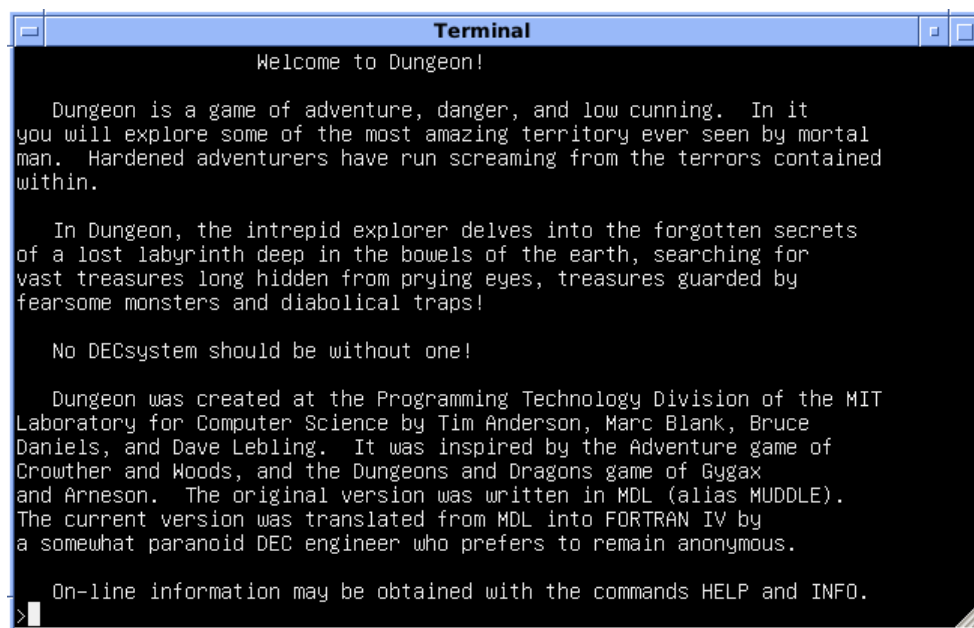
Rozgrywka stale idzie do przodu, gracze mogą obserwować rozwój swojej postaci i piąć się coraz wyżej w najróżniejszych rankingach, zawartych w danym tytule. Gry MMORPG są tytułami, w które można grać latami, stworzona i rozwijana w nich postać zostaje niejako awatarem swojego właściciela, a świat przedstawiony w grze, zostać może drugim życiem w którym można się zrelaksować i odreagować stres.

Zawartość tych gier najczęściej nie ogranicza się tylko do zawartości, którą można zająć się w pojedynkę. Różne gry wprowadzają różne mechanizmy, miejsca i wydarzenia, których wykonanie wymaga mniejszej lub większej grupy z którą trzeba współpracować. Wiele gier ma swoje fora internetowe gdzie można szukać potencjalnej drużyny. Równie popularne są serwisy LFG (Looking For Group) poświęcone danym grom, a służące właśnie, do łatwiejszego szukania ludzi, do grupy lub wspólnego grania we dwójkę, jak też umożliwiają wystawianie własnych ogłoszeń o poszukiwaniach osób do gry. Przykładowymi tego typu serwisami są między innymi destinylfg.net (poświęcone grze Destiny) lub poeparty.com (poświęcone grze Path Of Exile).

2 Historia gier MMO

2.1 Początki

Początków gatunku MMORPG można wypatrywać w grze, po której mało kto by się tego spodziewał, która nie posiadała ani żadnego trybu multiplayer, ani jakiegokolwiek systemu walki - Zork. Przed powstaniem dzisiejszych wciągających światów MMORPG, przygody online były ograniczone do świata przedstawionego za pomocą tekstu.



Rysunek 1: Zrzut ekranu z gry Zork (Źródło: <https://en.wikipedia.org/wiki/Zork>)[3]

We wczesnych latach 80., ludzie chcący odkrywać świat online wypełniony innymi ludźmi i stworami z którymi można było walczyć, mieli tylko jedną opcję do wyboru - MUD. Była to prosta gra online Multi-User Dungeon stworzona przez Roya Trubshawą.[5]



Rysunek 2: Zrzut ekranu z gry Multi-User Dungeon1 (Źródło: <https://en.wikipedia.org/wiki/MUD1>)[4]

2.2 Habitat i Neverwinter Nights jako pierwsze gry MMORPG

Habitat- gra online początkowo stworzona w 1985r. oraz udostępniona w 1986 jako beta test[7], okazała się być ogromnym krokiem do przodu w świecie gier online, również wyraźną inspiracją dzisiejszych, otwartych światów MMORPG, tworząc środowisko swobodne w takim stopniu, w jakim może się to już więcej nie powtórzyć w żadnej innej grze.



Rysunek 3: Zrzut ekranu z gry Habitat (Źródło: <http://www.gameclassification.com/EN/games/12005-Lucasfilms-Habitat/index.html>)[6]

Świat gry został zbudowany tak, by użytkownicy mogli odgrywać swój role-play jak tylko chcieli. Postacie w grze można było okraść, zabić oraz zrobić prawie

wszystko o czym można było wtedy pomyśleć, ale to, co najbardziej wyróżniało rozgrywkę, był prawie całkowity brak zasad, które jakkolwiek określałyby sposób gry. Środowisko było całkowicie uformowane przez graczy, a interakcje między postaciami kierowane wyłącznie przez samych użytkowników, stojących za nimi.

Po kilku latach od wypuszczenia Habitat, Stormfront Studios wydało grę znaną jako Neverwinter Nights, będące połączeniem graficznego role-playu oraz systemu walki opartego o Dungeons and Dragons.

W wielu aspektach gra ta, jest pierwszym przykładem tego, co dziś postrzegamy jako MMORPG. Przejawiała bardzo zbalansowany i przemyślany system walki, a także przeprowadzała turnieje walk PvP (Player versus Player). Wprowadziła również system gildii oraz ich wojen, dodając w ten sposób bardziej społeczne i oparte o współpracę drużynową elementy do rozgrywki.[5]



Rysunek 4: Zrzut ekranu z gry Neverwinter Nights (1991) (Źródło: <https://gameplay.pl/news.asp?ID=56094>)[8]

2.3 Ultima Online i Everquest

Ultima Online została wydana w 1997 roku, kiedy termin „MMORPG” jak i koncept tego gatunku były jeszcze nieznane i dopiero się formowały. Pomimo istniejących gier typu MUD czy MMO, żadna nie odbiła się głośnym echem i zdawało się, że nikt nie do końca wie jak stworzyć grę RPG, w której bawiłoby się naraz tysiące graczy.

Wszystko zmieniło się, gdy na rynku pojawiła się Ultima, która pobiła rekordy graczy i popularności, istniejące przed nią. Pomimo tego, że tytuł ten był niesamowicie trudny, zwłaszcza w początkach istnienia gry, Ultima została pierwszą grą MMO, która osiągnęła liczbę 100 000 subskrybentów, a ostatecznie sprzedało się ponad milion kopii.[10]



Rysunek 5: Zrzut ekranu z gry Ultima Online (Źródło: <https://www.engadget.com/2010-05-04-the-game-archaeologist-and-the-ultima-prize-the-history.html>)[9]

Jedną z najbardziej zapadających w pamięć cech Ultimy, był sposób, w jaki angażowała interakcje między graczami. Bez udogodnień takich jak czaty globalne, gracze musieli się kontaktować i spotykać bezpośrednio. Mogli ze sobą handlować, współpracować jak i okradać się, oszukiwać czy zabijać się nawzajem. Wraz z biegiem

czasu poszczególni gracze budowali swoją reputację i byli rozpoznawalni wśród innych jako bohaterowie lub złoczyńcy wśród społeczności pozostałych graczy.

Jeśli o Ultimie Online można powiedzieć, że niejako rozpowszechniła i zapoczątkowała gatunek MMORPG, to Everquest go jeszcze ulepszył. Wiele pomysłów które wprowadza, jak na przykład raidy wymagające wielu graczy do ukończenia, stały się schematami, powtarzającymi się w różnych grach tego gatunku.

Everquest jest też jednym z pierwszych MMO, które spopularyzowały tak zwany grinding - jest to powtarzalne wykonywanie jakiejś akcji, w celu zdobycia punktów doświadczenia lub innych nagród (na przykład zabijanie danej grupy stworów). W tamtym czasie grinding był wciągający. W celu zabijania silniejszych stworów, zmuszał on do współpracy z nieznanymi graczami oraz później był jedyną optymalną metodą zdobywania doświadczenia. Przez współpracę w takich ciężkich warunkach Everquest umożliwiał nawiązanie silnych relacji między graczami, nie tylko przez wspólne przetrwanie, ale także przez wspólne rozwijanie swoich postaci.[11]



Rysunek 6: Zrzut ekranu z gry Everquest (Źródło: <https://www.gry-online.pl/S026.asp?ID=9684>)[10]

2.4 Tibia

Tibia, która powstała jako projekt studencki, została wypuszczona w tym samym roku, co Ultima, w styczniu 1997 roku i jest obecnie jedną z niewielu wciąż aktywnych gier MMO z lat 90. oraz tych, które zachowały grafikę 2D.



Rysunek 7: Zrzut ekranu z wczesnej wersji gry Tibia (Źródło: <https://www.tibia-wiki.net/wiki/Historia>)[12]

W 1997 roku pojęcie gier MMO było nieznane prawie nikomu, poza najbardziej oddanym fanom gatunku. Po otwarciu gry minęło kilka dni zanim załogował się pierwszy gracz, ale z czasem liczba graczy była coraz większa.

Tibia stabilnie rosła przez pierwszy rok swojego istnienia, aż w końcu twórcy zdali sobie sprawę, że pierwszy, niewielki serwer nie spełniał już ich potrzeb. Większy serwer został uruchomiony w 1998 roku, a kod samej gry został przepisany od podstaw. Druga, ulepszona wersja gry została wypuszczona w formie bety 1999 roku i przyciągnęła około 150 graczy.



Rysunek 8: Zrzut ekranu z jednej z najnowszych wersji gry (Źródło: <https://www.tibia.com/abouttibia/?subtopic=screenshots>)[13]

Kiedy twórcy skończyli swoje studia zdecydowali, by zająć się swoją grą zawodowo i założyli firmę CipSoft. Zespół firmy założonej w 2001 roku, który składał się początkowo z 4 założycieli, urósł do 30 deweloperów w 2006 roku, aż do 89 ludzi pracujących nad grą dzisiaj.[14][15]

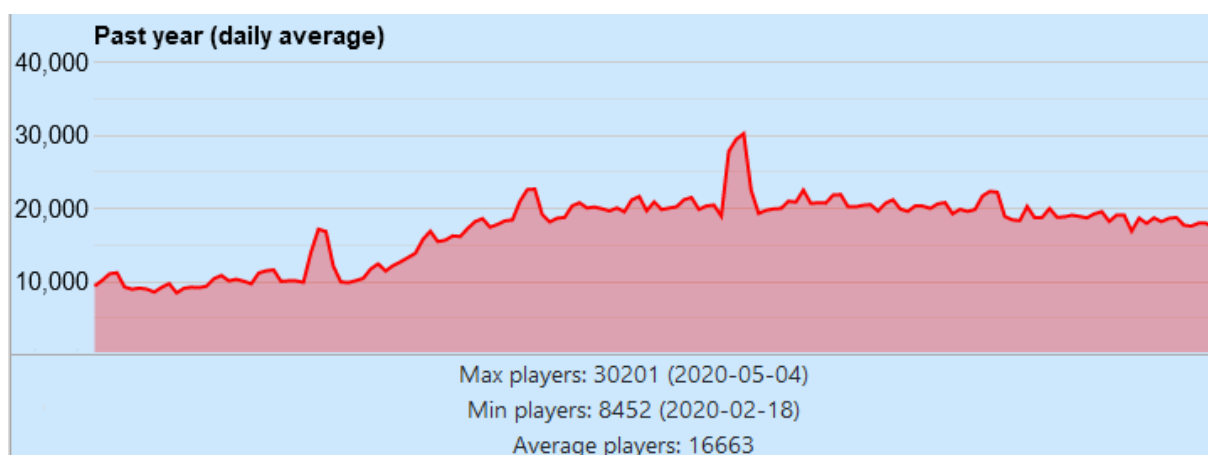
3 Tibia obecnie

3.1 Przyrost liczby graczy

Pomimo swojego wieku, Tibia wciąż jest całkiem popularna wśród graczy. W godzinach szczytu gromadzi trochę ponad 20 000 graczy online na swoich serwerach.

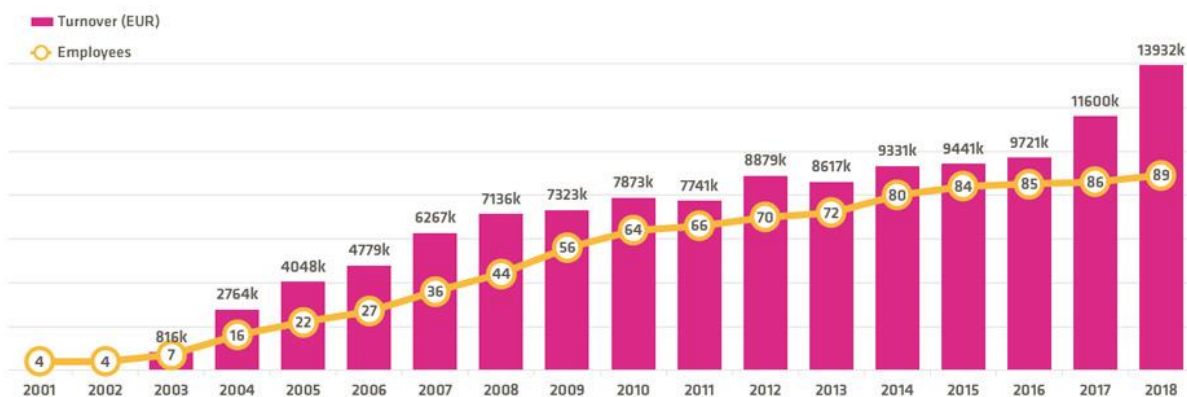
Poprzez ciągle aktualizacje gry, składające się z pomniejszych łatek w ciągu roku oraz 2 większych aktualizacji (letnie i zimowe aktualizacje) wprowadzających spore zmiany w grze, Tibia utrzymuje zainteresowanie swoich graczy dając ciągle kolejne cele do osiągnięcia użytkownikom.

W ostatnim roku zauważalny jest przyrost dziennej średniej liczby graczy, rozpoczynający się od lutego 2020 roku, co najprawdopodobniej jest skutkiem pandemii CoVid.



Rysunek 9: Średnia dzienna liczba graczy w przedziale luty-czerwiec 2020 (Źródło: <https://guildstats.eu/online-counter?lang=nl>)[16]

Popularność gry dobrze odzwierciedlają także przychody CipSoftu, które są w zdecydowanej większości oparte o Tibię, a także statystyki które podaje sama firma na oficjalnej stronie wskazując między innymi na to, że w 2018 roku średnio dziennie aktywnych użytkowników było 80 000, co stanowi ponad 8% więcej w stosunku do poprzedniego roku, a liczba użytkowników posiadających konto premium wzrosła o 12%[17]










Rysunek 10: Wzrost obrotów i liczby zatrudnionych osób w CipSoft w latach 2001-2018 (Źródło: <https://www.cipsoft.com/en/press/press-releases/165-result-financial-year-2018>)[18]

3.2 Popularność gry Tibia w Polsce i na świecie

Duża ilość graczy pamiętających poprzednie dziesięciolecie, kojarzy lepiej lub gorzej Tibię - albo przez negatywne stereotypy dotyczące graczy i prześmiewcze filmy na YouTube, albo przez to, że sami w tą produkcję grali. Przez wiele lat Tibia w Polsce była jedną z najpopularniejszych gier, zaraz obok Counter Strike'a, który również był wtedy szalenie popularny.

Pomimo tego, że prostota gry stanowiła obiekt wielu drwin, to właśnie ona i idące za nią bardzo niskie wymagania sprzętowe jak i wymagania dotyczące szybkości łącza internetowego przyczyniły się, że produkcja ta zdobyła bardzo dużą popularność wśród graczy. Gra zachęcała również tym, że można było grać za darmo, w przeciwieństwie do wielu innych gier MMORPG w tamtym czasie.

Obecnie Polacy stanowią drugą największą grupę użytkowników, wyprzedzani jedynie przez graczy z Brazylii.

Player nationality	Ilość - Procent
 - BR	8673 - 43.59%
 - PL	4036 - 20.29%
 - US	1019 - 5.12%
 - MX	865 - 4.35%
 - SE	788 - 3.96%
 - ES	525 - 2.64%
 - NL	496 - 2.49%
 - VE	418 - 2.1%
 - CA	361 - 1.81%
 - GB	345 - 1.73%
 - CL	335 - 1.68%
 - DE	285 - 1.43%
 - BE	152 - 0.76%
 - AU	147 - 0.74%
 - AR	141 - 0.71%
 - NO	117 - 0.59%
 - PT	112 - 0.56%

Rysunek 11: Liczba graczy z podziałem na kraj pochodzenia bazująca na użytkownikach portalu

Guildstat (Źródło: <https://guildstats.eu/census>)[19]

4 Proces tworzenia serwisu

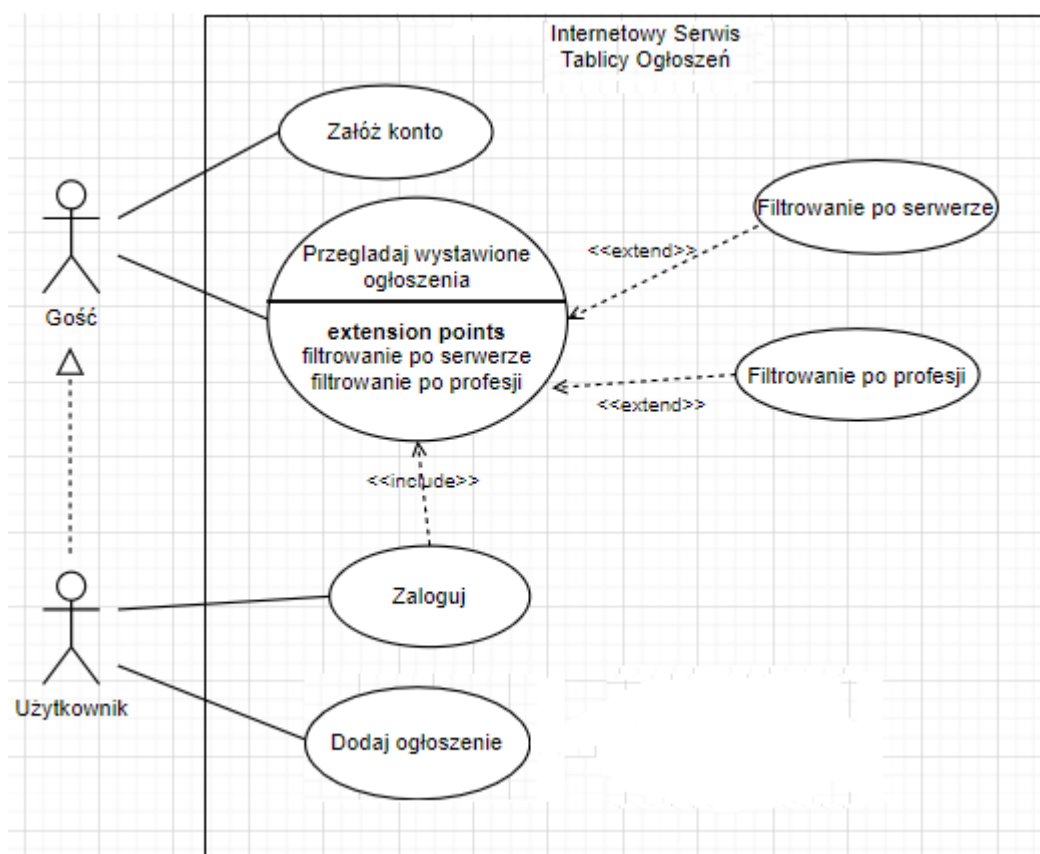
4.1 Pomysł

Większość zawartości dostępnej na wyższych poziomach w grze, nie jest dostępna dla pojedynczych graczy, lub byłaby kompletnie nieopłacalna do wykonania w pojedynkę. Wśród stron zajmujących się tematyką Tibii, brakuje takiej, która byłaby typowo nastawiona na szukanie i dobieranie sobie drużyny. Obecnie gracze korzystają głównie z forum na stronie gry, które samo w sobie również nie posiada wydzielonego działu gdzie można by było zamieścić i znaleźć posty o poszukiwaniach kogoś do wspólnej gry, przez co takie wpisy stają się mniej widoczne.

W samej grze znajduje się również prosty system, którego celem była pomoc w dobieraniu graczy, jednak wystawione tam pozycje przez graczy znikają po wylogowaniu się, przez co trzeba tworzyć swoje ogłoszenie na nowo za każdym razem po zalogowaniu się. Kolejną wadą tego systemu w grze jest jego znikoma popularność - często zdarza się, że w godzinach szczytu nie ma żadnego aktywnego poszukiwania.

Portal poświęcony samym ogłoszeniom, przez które szukamy kogoś do wspólnej gry, w których można zawrzeć informacje na temat tego w jakich godzinach gramy, w jakiej strefie czasowej się znajdujemy oraz jakimi językami się posługujemy mógłby bardzo ułatwić znalezienie kogoś do współpracy w grze.

4.2 Diagram oraz scenariusze przypadków użycia



Rysunek 12: Diagram przypadków użycia dla aktora Użytkownik oraz Gość. (Źródło: własne)

W systemie występuje dwóch aktorów:

- **Gość** - aktor, który może korzystać z możliwości przeglądania wystawionych ogłoszeń oraz zarejestrować w serwisie internetowym.
- **Użytkownik** – aktor, którym staje się Gość, po rejestracji w systemie. Posiada wszystkie możliwości gościa wraz z możliwością dodania oraz zarządzania ogłoszeniem po zalogowaniu w systemie.

Tabela 1 PU-1 Załóż konto

Lp.	PU-1
Nazwa:	Załącz konto
Aktor:	Gość
Warunki początkowe:	Gość posiada internetową skrzynkę pocztową i znajduje się na stronie głównej
Opis:	Gość chce zarejestrować się w systemie
Ścieżka główna:	1. Gość klika przycisk "Zarejestruj się" 2. Serwis internetowy wyświetla formularz rejestracji 3. Gość wpisuje adres e-mail oraz hasło i klika przycisk "Zarejestruj się" 4. Następuje zarejestrowanie w serwisie internetowym.
Ścieżka alternatywna:	3a. Gość wpisał adres e-mail, który został wcześniej użyty 4a. Serwis internetowy wyświetla komunikat, że podany adres e-mail został już wykorzystany
Warunki końcowe:	Rejestracja w systemie

Tabela 2 PU-2 Filtrowanie ogłoszeń po serwerze

Lp.	PU-2
Nazwa:	Filtrowanie ogłoszeń po serwerze
Aktor:	Gość, Użytkownik
Warunki początkowe:	Przeglądanie ogłoszeń wybierając serwer
Opis:	Gość, Użytkownik chcą znaleźć ogłoszenia na wybranym serwerze
Ścieżka główna:	1. Gość, Użytkownik wybierają interesujący ich serwer 2. Serwis internetowy wyświetla ogłoszenia na podanym serwerze
Ścieżka alternatywna:	
Warunki końcowe:	Wyświetlenie ogłoszeń spełniających podane kryterium wyszukiwania

Tabela 3 PU-3 Filtrowanie ogłoszeń po profesji

Lp.	PU-3
Nazwa:	Filtrowanie ogłoszeń po profesji
Aktor:	Gość, Kupujący
Warunki początkowe:	Przeglądanie ogłoszeń wybierając odpowiednią opcję z dostępnych profesji
Opis:	Gość, Użytkownik chcą znaleźć ogłoszenia z wybraną profesją
Ścieżka główna:	1. Gość, Użytkownik wybierają interesującą ich profesję 2. Serwis internetowy wyświetla ogłoszenia zawierające wybraną profesję
Ścieżka alternatywna:	
Warunki końcowe:	Wyświetlenie ogłoszeń spełniających podane kryterium wyszukiwania

Tabela 4 PU-4 Zaloguj

Lp.	PU-4
Nazwa:	Zaloguj
Aktor:	Użytkownik
Warunki początkowe:	Użytkownik został wcześniej zarejestrowany w systemie
Opis:	Użytkownik chce zalogować się do systemu
Ścieżka główna:	<ol style="list-style-type: none"> 1. Użytkownik klika w przycisk „Zaloguj się” w pasku nawigacyjnym 2. Serwis internetowy wyświetla formularz logowania 3. Użytkownik wpisuje adres e-mail oraz hasło i klika w przycisk „Zaloguj się” 4. 4. Serwis internetowy wyświetla widok ogłoszeń użytkownika
Ścieżka alternatywna:	<ol style="list-style-type: none"> 3a. Użytkownik wpisał niepoprawne hasło lub adres e-mail. 4a. Serwis internetowy wyświetla komunikat, że hasło lub adres e-mail został niepoprawnie wpisany
Warunki końcowe:	Użytkownik został zalogowany w systemie

Tabela 5 PU-5 Dodaj ogłoszenie

Lp.	PU-5
Nazwa:	Dodaj ogłoszenie
Aktor:	Użytkownik
Warunki początkowe:	Użytkownik zalogował się w systemie
Opis:	Użytkownik chce dodać ogłoszenie
Ścieżka główna:	<ol style="list-style-type: none"> 1. Użytkownik klika przycisk „Dodaj ogłoszenie” w pasku nawigacyjnym 2. Serwis internetowy wyświetla formularz dodania ogłoszenia 3. Użytkownik: <ol style="list-style-type: none"> a. Dodaje opis ogłoszenia b. Wybiera postać c. Wpisuje minimalne i maksymalne wymagania poziomowe 4. Użytkownik klika przycisk „Dodaj ogłoszenie” 5. Ogłoszenie zostaje dodane do bazy danych
Ścieżka alternatywna:	<ol style="list-style-type: none"> 3a. Użytkownik nie posiada żadnej zweryfikowanej postaci 4a. Serwis internetowy wyświetla komunikat o braku postaci do wyboru 5a. Błąd w systemie, ogłoszenie nie zostaje dodane
Warunki końcowe:	Użytkownik dodał ogłoszenie

4.3 Wykorzystane narzędzia, biblioteki i ich opis

4.3.1 ASP.NET Core

ASP.NET jest jednym z najbardziej udanych frameworków do tworzenia aplikacji internetowych, przygotowanych przez firmę Microsoft. Wraz z każdą aktualizacją pojawiają się nowe i rozszerzone funkcje, które pozwalają programistom wdrażać wysoce skalowalne i wydajne aplikacje internetowe.

Wraz z pojawieniem się technologii .NET Core możemy tworzyć aplikacje ASP.NET Core i wdrażać je na wielu systemach, tj. Windows, Linux oraz MacOS. Microsoft wraz ze społecznością programistyczną włożył ogromny wysiłek, w uczynienie systemu Linux idealną platformą do wdrażania aplikacji ASP.NET Core.[20]

.NET Core został stworzony przez Microsoft w roku 2016 i jego największymi zaletami są wieloplatformowość, a także open sourcowy kod źródłowy, który można znaleźć na githubie firmy Microsoft. .NET Core jest wieloplatformowy, to znaczy, że działa na dowolnym systemie operacyjnym (windows, linux, mac). Kolejną zaletą jest to, że ma większą wydajność niż zwykły .NET Framework.[21]

4.3.2 Język programowania

C# to nowoczesny język programowania stworzony i rozwijany przez Microsoft. Obecnie znajduje się w czołówce najpopularniejszych języków programowania na świecie. Używany jest głównie do tworzenia oprogramowania z wykorzystaniem platformy .NET.[22]

Język C# jest połączony z platformą .NET, która jest zarówno frameworkiem jak i środowiskiem uruchomieniowym. Kiedyś jedynym, głównym przeznaczeniem języka C# było pisanie aplikacji, które miał obsługiwać Windows, dziś, dzięki dostępności platformy .NET na takich systemach jak Linux, czy Mac, możliwe jest tworzenie ich właściwie na wszystkie platformy, co czyni z niego język bardzo uniwersalny. Dodatkowo uznawany jest za dosyć łatwy do przyswojenia.[23]

4.3.3 Środowisko programistyczne

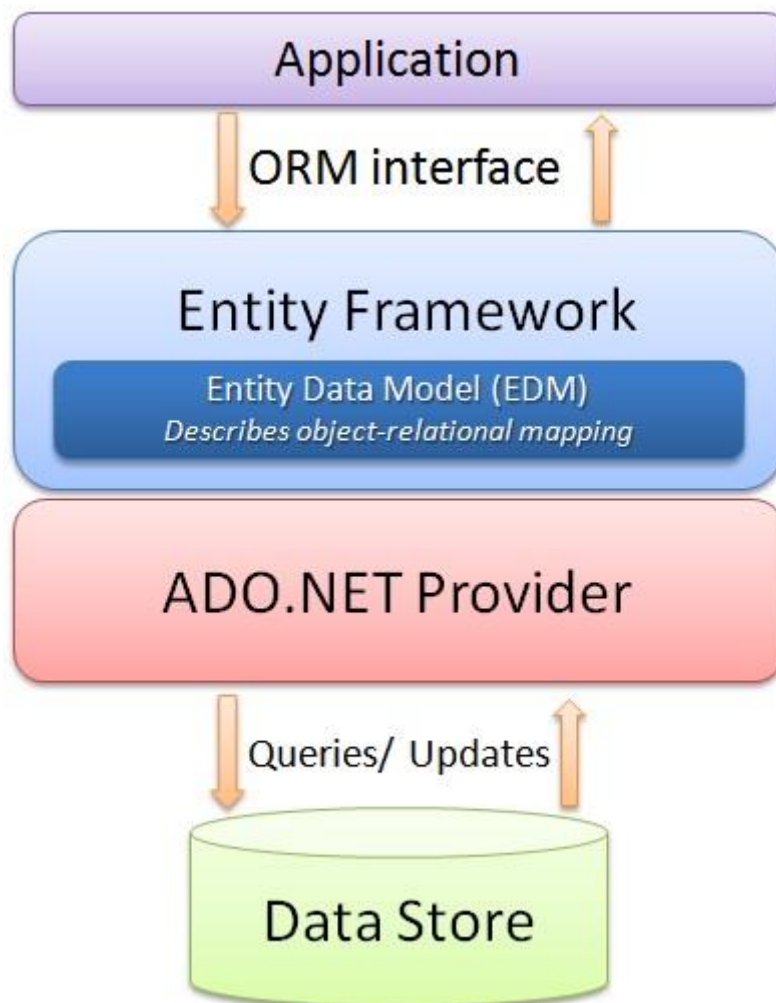
Środowisko programistyczne jest zbiorem aplikacji pozwalających na tworzenie, testy i modyfikacje oprogramowania. W skład środowiska programistycznego wchodzić może zaawansowany edytor kodu, system kontroli wersji, narzędzia do testowania, a także w wypadku niektórych języków także środowiska uruchomieniowe, które są potrzebne na przykład do języków z rodziny .NET i są niezbędne do uruchomienia programu.[24]

Najpopularniejszymi środowiskami do programowania w C# jest Rider od JetBrains oraz Visual Studio firmy Microsoft. W projekcie wybór padł na środowisko Visual Studio 2019.

4.3.4 Entity Framework

Entity Framework jest narzędziem mapowania obiektowo-relacyjnego (tzw. ORM-Object-relational mapping). Generuje obiekty biznesowe oraz encje zgodnie z tabelami baz danych. Obiekt biznesowy jest encją (entity) w wielowarstwowej aplikacji, która działa w połączeniu z dostępem do bazy danych oraz warstwą logiki biznesowej służącą do przesyłania danych. Z kolei entity odnosi się do czegoś co jest unikatowe i istnieje oddzielnie, np. tabela bazy danych.

Entity Framework pozwala na łatwe wykonywanie operacji CRUD (Create, Read, Update, Delete) czy też proste zarządzanie relacjami w bazie danych. Dzięki wykorzystaniu Entity Framework cała logika dostępu do bazy danych może być napisana w języku wyższego poziomu. [25]



Rysunek 13: architektura Entity Framework (Źródło: <https://www.codeproject.com/Articles/363040/An-Introduction-to-Entity-Framework-for-Absolute-B>)[26]

4.3.5 Identity Framework

Identity Framework jest narzędziem, które obsługuje funkcje logowania w ASP.NET aplikacji. Umożliwia tworzenie kont z informacjami logowania, które są przechowywane w usłudze Identity lub korzystać z zewnętrznych autoryzacji, obejmujących serwisy m.in. Facebook, Google, Microsoft.

Identity pozwala na kontrolę zapisu – możliwość wyboru, gdzie dane zostaną zapisane – m.in. SQL Azure, MS SQL, NOSQL. Zapewnia łatwe dodawanie dodatkowych informacji do profilu użytkownika, wsparcie dla ról w aplikacji.

4.4 Pierwsze zaprogramowane funkcjonalności

Tworzenie serwisu, zostało rozpoczęte, zaprogramowaniem działającego systemu logowania i rejestracji konta. Programowanie tych funkcji, w ASP.NET Core jest bardzo łatwe, dzięki wykorzystaniu Identity Framework.

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Register(RegisterViewModel model, string returnUrl = null)
{
    ViewData["ReturnUrl"] = returnUrl;
    if (ModelState.IsValid)
    {
        var user = new ApplicationUser { UserName = model.UserName };
        var result = await _userManager.CreateAsync(user, model.Password);
        if (result.Succeeded)
        {
            _logger.LogInformation("User created a new account with password.");

            await _signInManager.SignInAsync(user, isPersistent: false);
            _logger.LogInformation("User created a new account with password.");
            return RedirectToLocal(returnUrl);
        }
        AddErrors(result);
    }

    // If we got this far, something failed, redisplay form
    return View(model);
}
```

Rysunek 14: Logika rejestracji w serwisie (Źródło: własne)

Po otrzymaniu modelu danych, wysłanego z widoku rejestracji, do metody Register() kontrolera AccountController, wykonana zostaje metoda CreateAsync(), która w swoich parametrach przyjmuje model użytkownika, stworzony na podstawie wcześniej otrzymanego modelu oraz łańcuch znaków reprezentujący hasło.

Jeśli wszystko przebiegło pomyślnie, Identity Framework szyfruje hasło podane w formularzu rejestracji i tworzy konto, na które użytkownik zostaje od razu zalogowany. W wypadku błędów następuje ponowne wyświetlenie formularza rejestracji z wyszczególnionymi błędami, które wystąpiły.


```

[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Login(LoginViewModel model, string returnUrl = null)
{
    ViewData["ReturnUrl"] = returnUrl;
    if (ModelState.IsValid)
    {
        // This doesn't count login failures towards account lockout
        // To enable password failures to trigger account lockout, set lockoutOnFailure: true
        ApplicationUser user = _userManager.FindByNameAsync(model.Username).Result;
        var result = await _signInManager.PasswordSignInAsync(user, model.Password, model.RememberMe, lockoutOnFailure: false);
        if (result.Succeeded)
        {
            _logger.LogInformation("User logged in.");
            return RedirectToLocal(returnUrl);
        }
        if (result.IsLockedOut)
        {
            _logger.LogWarning("User account locked out.");
            // return RedirectToAction(nameof(Lockout));
            return Content("User locked out");
        }
        else
        {
            ModelState.AddModelError(string.Empty, "Invalid login attempt.");
            return View(model);
        }
    }

    // If we got this far, something failed, redisplay form
    return View(model);
}

```

Rysunek 15: Logika logowania w kodzie serwisu (Źródło: własne)

System logowania oraz rejestracji, wykorzystują Identity Framework do uwierzytelnienia użytkownika. Model danych wysłany z formularza do kontrolera, zawiera pola Username, Password oraz RememberMe. Po nazwie użytkownika zostaje wyszukane odpowiednie konto, a następnie, wykonane zostaje sprawdzenie zgodności podanego hasła, z jego zaszyfrowaną wersją w bazie danych.

Po poprawnym procesie uwierzytelnienia użytkownik zostaje zalogowany. W przeciwnym wypadku formularz logowania zostaje wyświetlony ponownie wraz z błędami, które wystąpiły.

4.5 Kontroler tablicy i logika tworzenia ogłoszeń

Następnym stworzonym elementem aplikacji, był `BulletinController`, odpowiadający za zarządzanie mechaniką ogłoszeń oraz powiązany z nimi model danych.

```
14 references
public class Advert : IEntity
{
    4 references
    public int Id { get; set; }
    1 reference
    public string AuthorName { get; set; }

    [Range(0,9999, ErrorMessage = "Value must be numeric from {1} to {2}")]
    [Required]
    [Display(Name = "Minimum teammate level")]
    0 references
    public int MinLevel { get; set; }

    [Range(0, 9999, ErrorMessage = "Value must be numeric from {1} to {2}")]
    [Required]
    [Display(Name = "Maximum teammate level")]
    0 references
    public int MaxLevel { get; set; }

    [Display(Name = "Description")]
    0 references
    public string Text { get; set; }

    0 references
    public string ApplicationUserId { get; set; }
    1 reference
    public ApplicationUser ApplicationUser { get; set; }
    2 references
    public GameCharacter GameCharacter { get; set; }
    2 references
    public int GameCharacterId { get; set; }
}
```

Rysunek 16: Model ogłoszenia wykorzystywany w aplikacji (Źródło: własne)

Pierwszymi metodami napisanymi w kontrolerze, zostały odpowiadające za wyświetlanie listy ogłoszeń oraz ich tworzenie. Początkowo, w celu łatwiejszego testowania poprawnego dodawania rekordów do bazy danych, nie została wprowadzona żadna logika weryfikacji danych.

Metoda `Board()` odpowiadała za wyświetlanie widoku, zawierającego wszystkie wystawione obecnie ogłoszenia, pasujące do wybranych parametrów filtrowania, a `Create()` za tworzenie nowych wpisów.

```

0 references
public IActionResult Board(BoardAdvertViewModel model)
{
    List<AdvertDTO> adsList = _advertsService.GetAllAdverts().Result;

    BoardAdvertViewModel viewModel = new BoardAdvertViewModel()
    {
        AdsList = adsList,
        FilterFormAdvert = model.FilterFormAdvert,
        VocList = GetVocationList(),
        WorldsList = GetGameWorldsList().worlds.allworlds
    };
    if (viewModel.FilterFormAdvert == null)
    {
        return View(viewModel);
    }
    else
    {
        adsList = _advertsService.GetFiltered(adsList, model.FilterFormAdvert);

        viewModel.AdsList = adsList;
        return View(viewModel);
    }
}

```

Rysunek 17: Metoda odpowiadająca za wyświetlenie listy wszystkich ogłoszeń (Źródło: własne)

```

[HttpPost]
0 references
public async Task<IActionResult> Create(CreateAdvertViewModel model)
{
    ApplicationUser user = _userManager.GetUserAsync(HttpContext.User).Result;

    if(_advertsService.CheckIfCharacterHasAdvert(model.SelectedCharacterId))
    {
        ModelState.AddModelError(string.Empty, "This character has advert already");
        model.OwnedCharactersList = _charactersService.GetUserCharacters(_userManager.GetUserAsync(User).Result.Id);
        return View(model);
    }

    var result = await _advertsService.Create(model.Ad, user, model.SelectedCharacterId);
    if (result.Succeeded)
    {
        return RedirectToAction("MyAdverts", new { userName = user.UserName });
    }

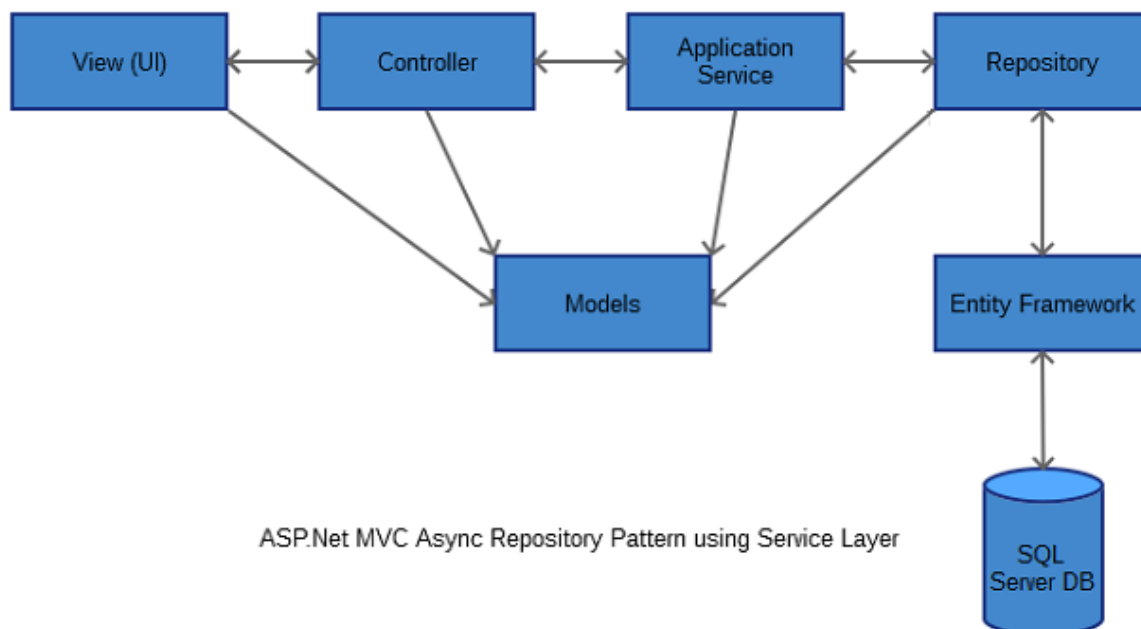
    ModelState.AddModelError(string.Empty, "This character does not exist");
    return View(model);
}

```

Rysunek 18: Metoda odpowiadająca za tworzenie nowych ogłoszeń (Źródło: własne)

4.6 Podział na warstwy serwisu i repozytorium

W celu organizacji struktury projektu po stworzeniu pierwszych funkcji aplikacji kod został podzielony na warstwy serwisu i repozytorium.



Rysunek 19: Wzorzec asynchronicznego repozytorium z wykorzystaniem warstwy serwisu w aplikacjach ASP.NET MVC (Źródło: <https://social.technet.microsoft.com/wiki/contents/articles/39621.asp-net-mvc-solution-architecture-project-with-async-repository-pattern-by-service-having-dependency-injection-by-unity-page-1.aspx>)[27]

Klasa repozytorium jest skupiona wyłącznie na operacjach CRUD (Create Read Update Delete) i nie powinna zawierać żadnej logiki biznesowej, która powinna być zawarta wyłącznie w warstwie serwisowej. Podstawowymi metodami zawartymi w repozytorium są przeważnie

- **Get()** - odpowiadająca za zwracanie konkretnego rekordu z bazy danych (najczęściej na podstawie przekazanym ID)
- **GetAll()** - odpowiadająca za zwracanie listy wszystkich rekordów z danej tabeli
- **Add()** – odpowiadająca za dodawanie nowych rekordów

- Update() – odpowiadająca za aktualizowanie rekordów w bazie
- Delete() – odpowiadająca za usuwanie rekordów z bazy

```

2 references
public interface IRepository<T> where T : class, IEntity
{
    6 references
    Task<List<T>> GetAll();
    5 references
    Task<T> Get(int id);
    3 references
    Task<T> Add(T entity);
    2 references
    Task<T> Update(T entity);
    3 references
    Task<T> Delete(int id);
}

```

Rysunek 20: Interfejs repozytorium implementujący sygnatury podstawowych metod CRUD (Źródło: własne)

Klasy serwisu z kolei odpowiadają za wszelką logikę biznesową zawartą w projekcie, taką jak na przykład weryfikacja danych w przesyłanych modelach, filtrowanie list, czy mapowanie obiektów modeli z bazy danych na odpowiadające im DTO (Data Transfer Object).

```

2 references
public async Task<List<AdvertDTO>> GetUserAdverts(string userName)
{
    List<Advert> adList = await advertRepository.GetAll();
    List<AdvertDTO> model = mapper.Map<List<AdvertDTO>>(adList).Where(x => x.AuthorName == userName).ToList();

    return model;
}

2 references
public List<AdvertDTO> GetFiltered(List<AdvertDTO> advertList, AdvertDTO filter)
{
    return advertList
        .Where(x => filter.GameCharacter.World != null ? x.GameCharacter.World == filter.GameCharacter.World : true)
        .Where(x => filter.GameCharacter.Vocation != null ? x.GameCharacter.Vocation == filter.GameCharacter.Vocation : true)
        .ToList();
}

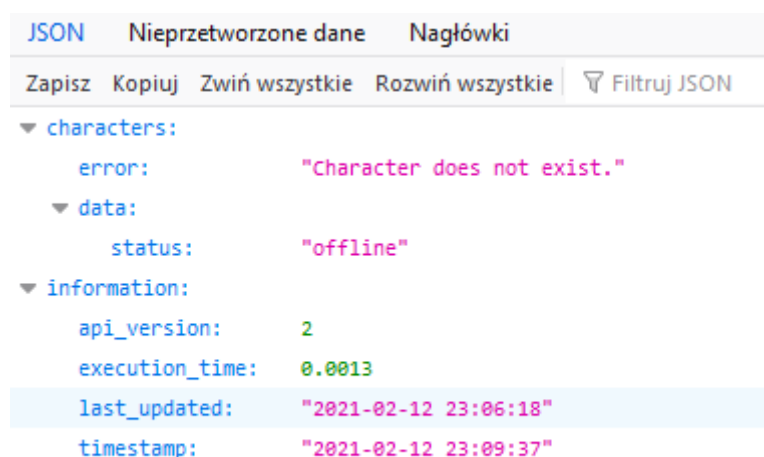
```

Rysunek 21: Przykładowe metody zawierające logikę biznesową w warstwie serwisu (Źródło: własne)

4.7 Model postaci i przypisywanie postaci do konta

Ostatnim fragmentem aplikacji bezpośrednio związanym z mechaniką tworzenia ogłoszeń była logika przypisywania postaci z gry do konta użytkownika. Dzięki wykorzystaniu wzorca podziału kodu na warstwy repozytorium i serwisu, wprowadzenie kolejnego modelu danych jest już znacznie łatwiejsze do zrealizowania, ze względu na fakt wykorzystania metod generycznych przez repozytorium.

Każde ogłoszenie w aplikacji dotyczy konkretnej postaci istniejącej w samej grze. W celu weryfikacji związanych z nimi danych wykorzystane zostało publiczne API, które dostarcza potrzebne informacje. Konto użytkownika nie może wystawić żadnego ogłoszenia jeśli nie posiada przypisanej do siebie przynajmniej jednej postaci.



Rysunek 22: Odpowiedź z [api.tibiadata.com](https://api.tibiadata.com/v2/characters/Saeqis.json) w wypadku nieistniejącej postaci (Źródło: <https://api.tibiadata.com/v2/characters/Saeqis.json>)[28]

JSONNieprzetworzone daneNagłówki

ZapiszKopiuujZwiń wszystkieRozwiń wszystkieFiltruj JSON

▼ characters:

▼ data:

name:"Ararat"

title:"None (15 titles unlocked)"

sex:"female"

vocation:"Master Sorcerer"

level:684

achievement_points:565

world:"Secura"

residence:"Roshamuul"

married_to:"Luchian"

▼ house:

name:"Darashia 8, Flat 05"

town:"Darashia"

paid:"2021-02-19"

world:"Secura"

houseid:0

▼ guild:

name:"Illuminatists"

rank:"Council of Seven"

▼ last_login:

▼ 0:

date:"2021-02-12 04:07:08.000000"

timezone_type:2

timezone:"CET"

comment:""

account_status:"Premium Account"

status:"offline"

Rysunek 23: Fragment odpowiedzi API zawierającej dane dotyczące przykładowej istniejącej postaci (Źródło: <https://api.tibiadata.com/v2/characters/Ararat.json>)[28]

Podstawowym problemem, któremu należało zapobiec, było niebezpieczeństwo wystawiania przez użytkowników ogłoszeń dotyczących postaci, których nie są właścicielami przez co mogliby się podszywać pod inne osoby. Zaprojektowany został mechanizm weryfikacji aby zapobiec wystąpieniu takiej sytuacji, polegający na generowaniu przez aplikację unikalnego GUID, który następnie należało wkleić do opisu postaci na stronie gry, co może zostać zrobione jedynie przez faktycznego właściciela danej postaci. Weryfikacja zgodności GUID wygenerowanego przez aplikację i tego wklejonego w opis postaci odbywa się przy pomocy zewnętrznego API udostępniającego między innymi opis postaci zamieszczony na oficjalnej stronie gry.

```

3 references
public Character GetCharacterDetailsIfExists(string charName)
{
    var characterRequest = new RestRequest(charName + ".json", DataFormat.Json);
    var characterResponse = tibiaApiClient.Get(characterRequest);
    Character characterApiResponse = JsonConvert.DeserializeObject<Character>(characterResponse.Content);

    return characterApiResponse;
}

```

Rysunek 24: Pobieranie danych istniejących postaci w kodzie aplikacji (Źródło: własne)

```

▼ characters:
  ▼ data:
    name:          "Cetyryzyna Di chlorowodorku"
    title:         "None (5 titles unlocked)"
    sex:           "female"
    vocation:      "Royal Paladin"
    level:         182
    achievement_points: 130
    world:         "Nefera"
    residence:     "Roshamuul"
  ▼ last_login:
    ▼ 0:
      date:        "2020-12-26 10:09:53.000000"
      timezone_type: 2
      timezone:    "CET"
      comment:     "c22ec938-2d96-4741-8ee5-e8e458f67c48"

```

Rysunek 25: Dane postaci na api.tibiadata po wklejeniu kodu weryfikacyjnego, do opisu postaci widocznego jako wartość wewnątrz comment (Źródło:

<https://api.tibiadata.com/v2/characters/Cetyryzyna%20Di%20Chlorowodorku.json>)[28]

```

3 references
public bool VerifyToken(GameCharacterDTO model)
{
    var character = GetCharacterDetailsIfExists(model.CharacterName);
    if(character.characters.data.comment != null)
    {
        if (character.characters.data.comment.Contains(model.VerificationToken))
        {
            return true;
        }
    }

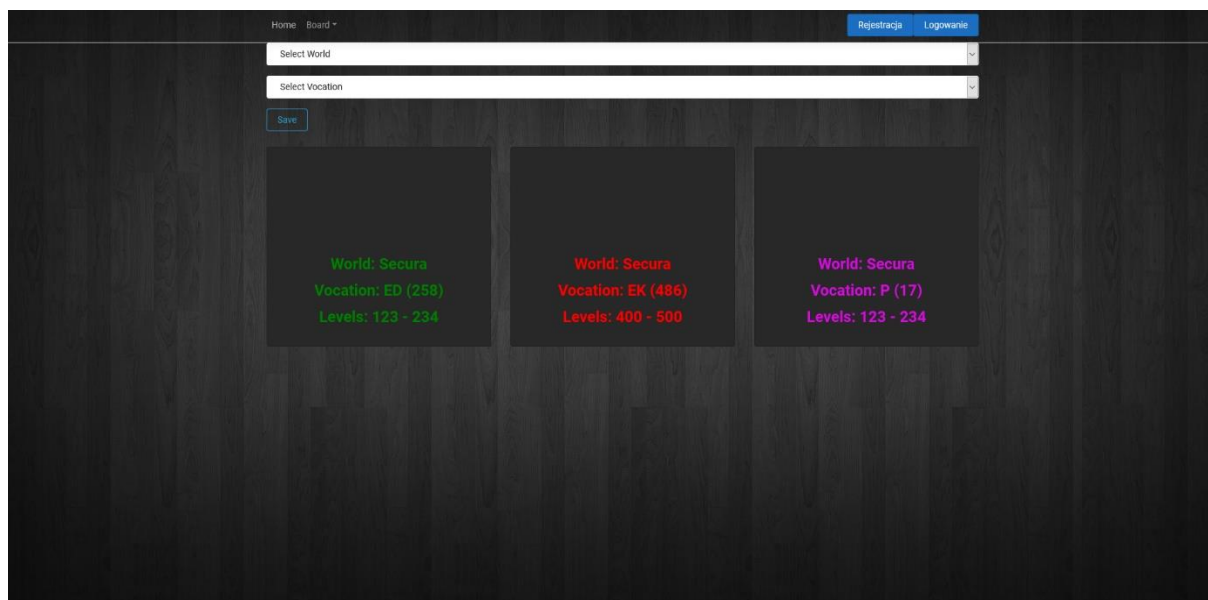
    return false;
}

```

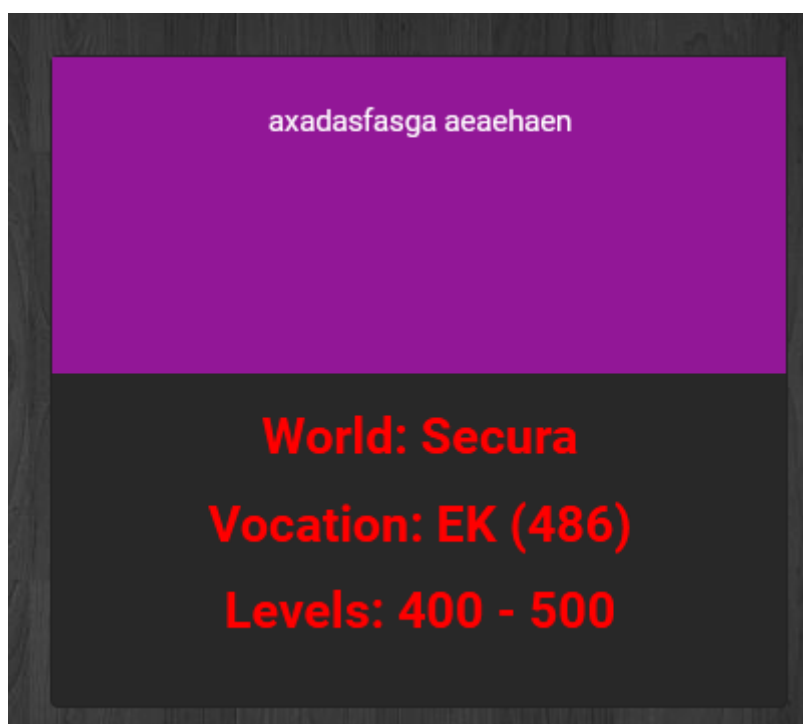
Rysunek 26: Sprawdzenie poprawności kodu weryfikacyjnego wklejonego w opis postaci na stronie gry (Źródło: własne)

5 Finalna aplikacja

Po wejściu na stronę aplikacji jesteśmy przenoszeni na stronę zawierającą wszystkie ogłoszenia, która została ustawiona jako domyślna lokalizacja.



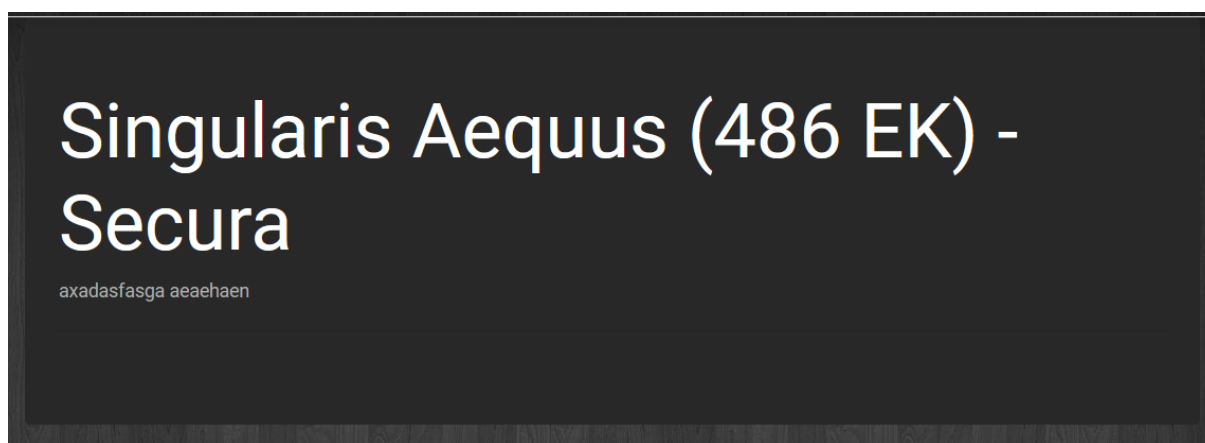
Rysunek 27: Strona główna aplikacji (Źródło: własne)



Rysunek 28: Wyświetlenie opisu po najechaniu myszką na kartę (Źródło: własne)

Na stronie tej w formie kart widzimy wszystkie aktualnie wystawione ogłoszenia graczy. Po najechnięciu myszką na jedną z kart można zobaczyć skrócony opis znajdujący się w ogłoszeniu (rys. 28).

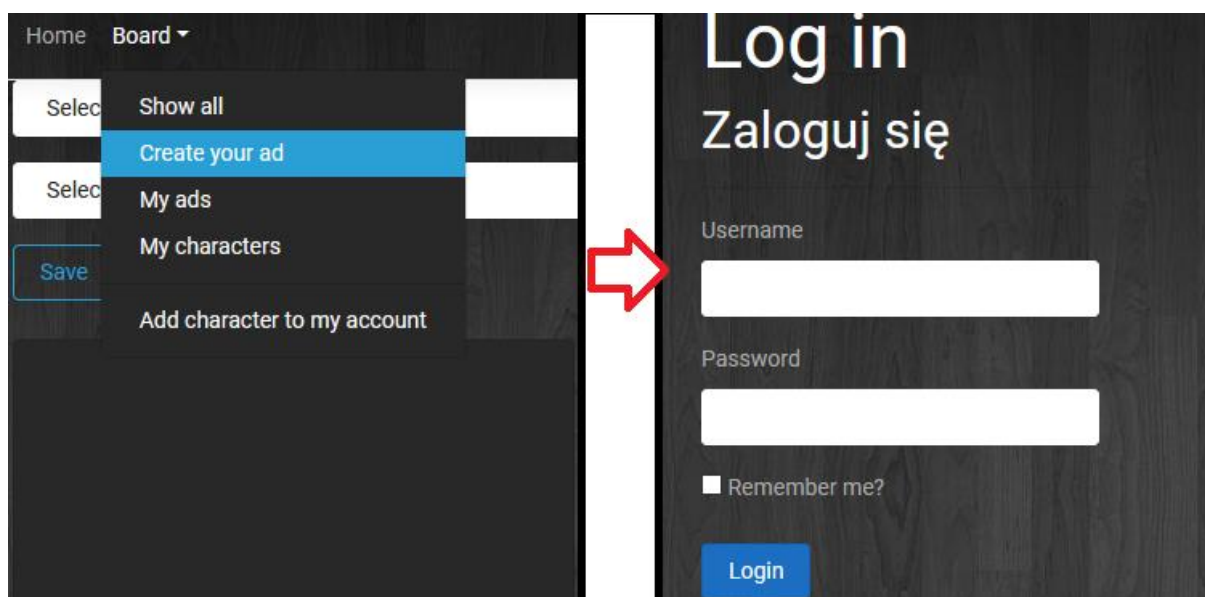
Po kliknięciu na wybraną kartę, przechodzimy do szczegółowego widoku, na którym widać nazwę postaci, dla której użytkownik szuka drużyny, w nawiasie zobaczyć można poziom, jak i profesję postaci oraz dalej serwer, na którym dana postać się znajduje. Poniżej tych informacji znajduje się opis, ustawiony przez osobę wstawiającą ogłoszenie (rys. 29). Są to wystarczające informacje, by z łatwością spotkać i skontaktować się z daną osobą już w grze.



Rysunek 29: Karta w widoku ze szczegółami ogłoszenia (Źródło: własne)

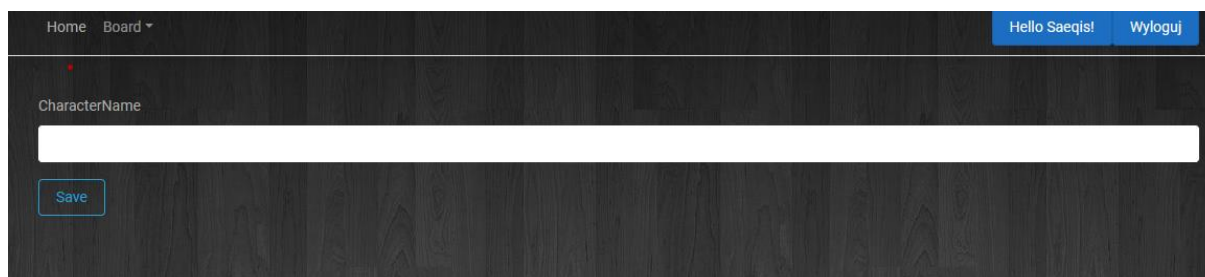
Aby wystawić swoje własne ogłoszenie w serwisie musimy najpierw założyć konto. Jeśli spróbujemy przejść do strony wstawiania ogłoszenia bez konta zostaniemy przeniesieni do widoku logowania się.

W przypadku braku konta należy je założyć. Rejestracja przebiega w bardzo prosty sposób. Aplikacja nie aktywuje kont poprzez wysyłanie kodów aktywacyjnych na maila. Każde nowe konto jest domyślnie traktowane jako nieaktywne, aż do momentu przypisania do niego przynajmniej jednej postaci ze swojego konta do gry. Po poprawnym przypisaniu postaci do konta, można bez problemu wstawić związane z nią ogłoszenie.



Rysunek 30: Próba stworzenia ogłoszenia bez posiadanego konta i widok logowania (Źródło: własne)

Po udanym założeniu konta, należy przypisać do niego postać ze swojego konta w grze, w tym celu używamy rozwijanego menu „*Board*” na górnej belce i przechodzimy pod odnośnik „*Add character to my account*”.



Rysunek 31: Pierwszy widok przypisywania postaci do konta (Źródło: własne)

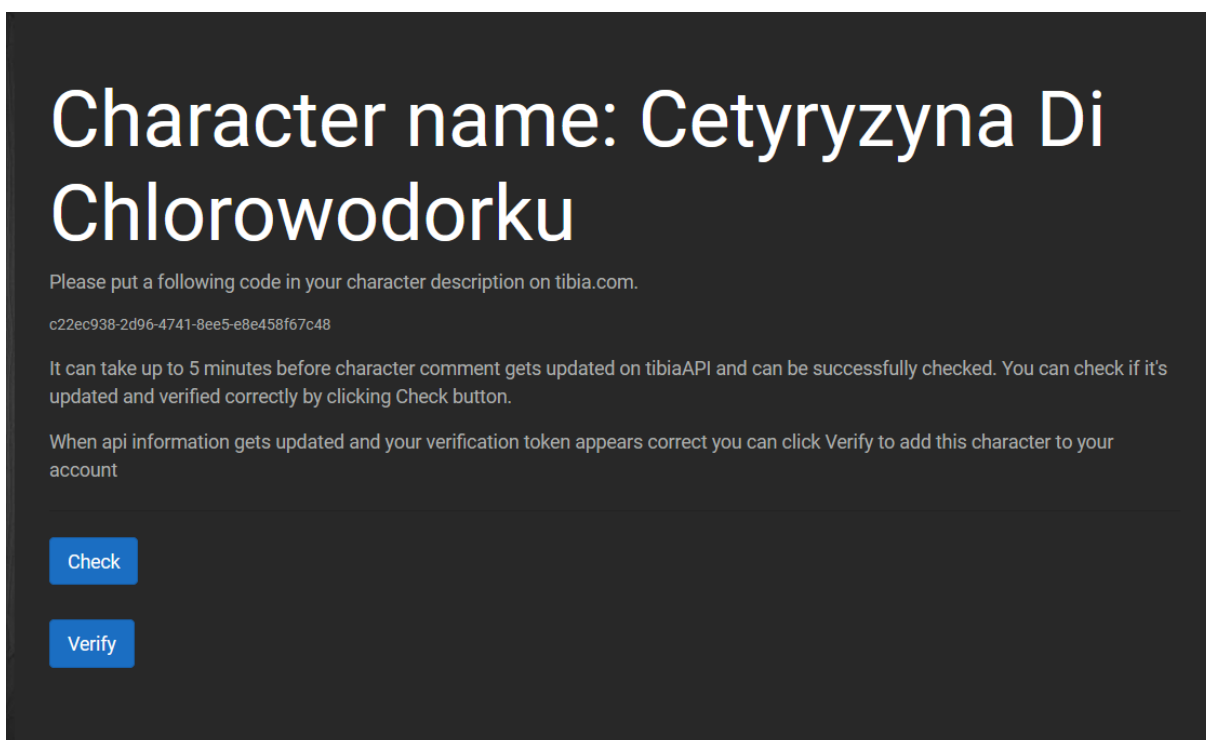
Na pierwszym widoku widnieje pasek w którym należy wprowadzić nazwę postaci którą chcemy przypisać do konta. Po wpisaniu nazwy i kliknięciu w przycisk „*Save*” następuje weryfikacja czy podana postać istnieje. W przypadku błędów zostaje wyświetlony stosowny komunikat, a pole, w którym była wprowadzona nazwa, zostaje wyczyszczone (rys. 32).



Rysunek 32: Komunikat błędu po próbie przypisania nieistniejącej postaci. (Źródło: własne)

Sprawdzenie istnienia podanej postaci, zostaje wykonane poprzez pobranie odpowiedzi z publicznego API, udostępniającego między innymi dane postaci lub komunikat błędu, jeśli dana postać nie istnieje.

Po wpisaniu nazwy istniejącej postaci, która nie została jeszcze przypisana do żadnego innego konta w serwisie, zostajemy przeniesieni do drugiego widoku, który przekazuje instrukcje, co należy zrobić, aby przypisać postać do swojego konta.



Rysunek 33: Drugi widok podczas przypisywania postaci do swojego konta (Źródło: własne)

Aby zweryfikować, że dana postać faktycznie należy do danego użytkownika, należy w celu weryfikacji, dodać wygenerowany kod do opisu postaci na stronie gry.

W tym celu, trzeba zalogować się na swoje konto do gry na stronie internetowej Tibia.com (rys. 34). Po zalogowaniu się na głównym widoku My Account widoczna jest

lista postaci (rys. 35) z której należy wybrać odpowiednią i kliknąć przy niej odnośnik Edit, który przeniesie nas do widoku szczegółowego, dotyczącego danej postaci gdzie również możemy wkleić wygenerowany kod w pole Comment (rys. 36).

Account Login

Email Address:

Login

Tibia Password:

Lost Account

If your Tibia Account is already connected to an authenticator, click on "Use Authenticator". A field will be displayed which allows you to provide your authenticator token along with your account data upon login. Otherwise, you will be asked for your authenticator token in the next step.

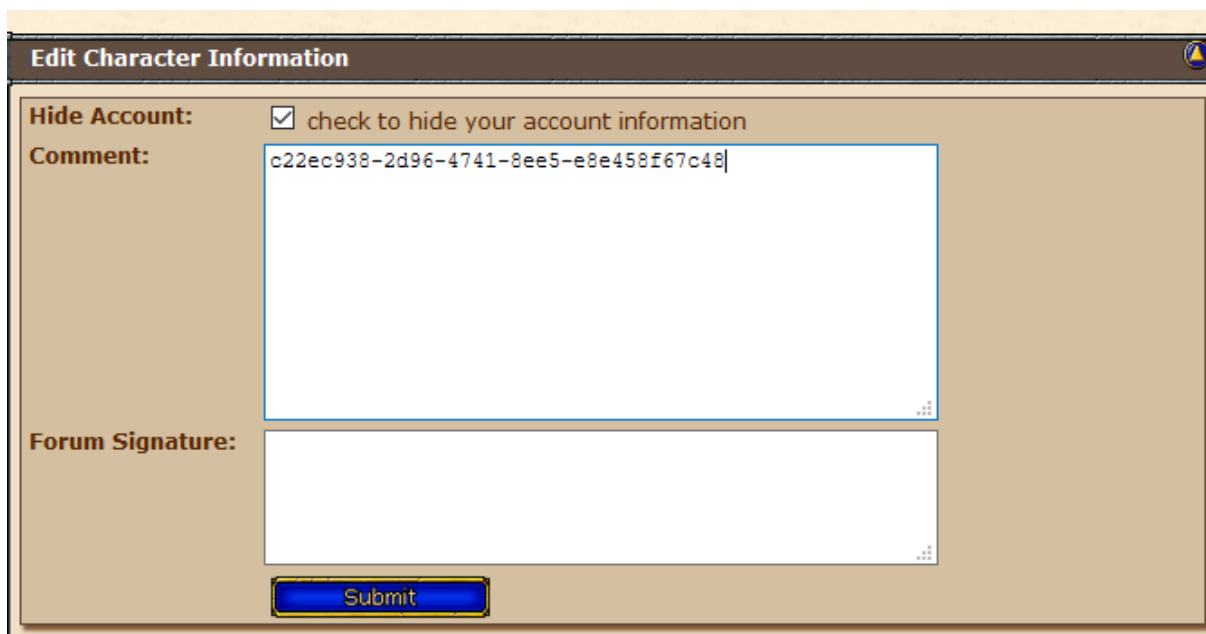
Use Authenticator

An authenticator is a security feature which helps to prevent any unauthorised access to your Tibia account! You can connect your account to an authenticator via your account management page.

Rysunek 34: Logowanie na konto gry na stronie internetowej.(Źródło: Tibia.com)[29]

2.	Bloody Pancake Elder Druid - Level 37 - On Secura		
3.	Brithiath Elder Druid - Level 258 - On Secura Guild Membership: Litul Ceiling Cat of the Lolcatz		
4.	Bus Driver Singu Elder Druid - Level 62 - On Zuna		
5.	Cetyryzyna Di chlorowodorku Royal Paladin - Level 182 - On Nefera		[Edit] [Delete]
6.	Dekarbo ksylacja Oksydanowa Master Sorcerer - Level 101 - On Secura Guild Membership: Litul Ceiling Cat of the Lolcatz		
7.	Donut of destruction Knight - Level 8 - On Antica		
8.	Dylsektyczka Elder Druid - Level 43 - On Peloria		
9.	Dysmutaza ponadtlenkowa Elder Druid - Level 36 - On Refugia		

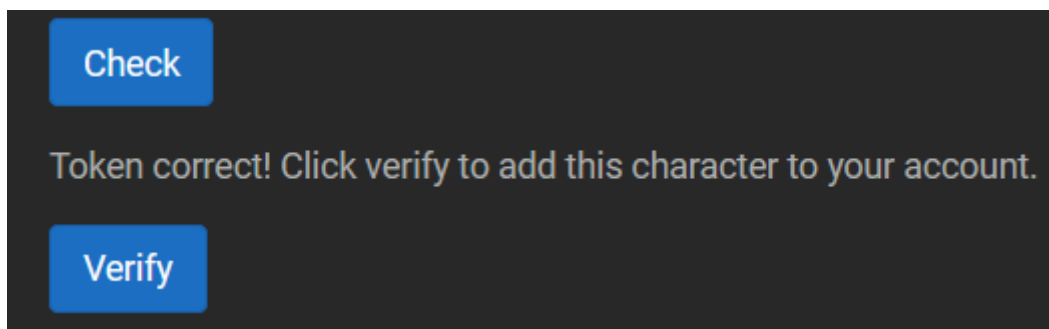
Rysunek 35 Widok listy postaci na stronie gry (Źródło: Tibia.com)[29]



Rysunek 36: Fragment szczegółowego widoku postaci w którym należy wkleić kod weryfikacyjny (Źródło: własne)

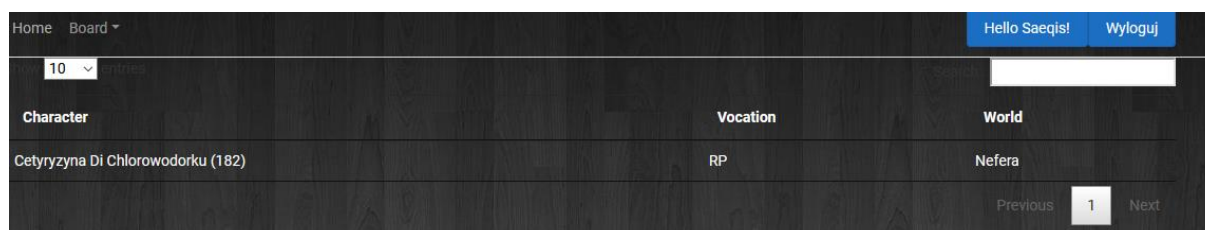
Po wykonaniu powyższych kroków, należy kliknąć na stronie aplikacji przycisk Check, który odpowiada za sprawdzenie czy informacje danej postaci zostały już odświeżone i poprawności kodu weryfikacyjnego. Jeśli sprawdzenie zwróci poprawny wynik, można wtedy kliknąć przycisk Verify, który wyśle dane na serwer i w przypadku poprawnego kodu przypisze postać do konta w serwisie, a w przypadku błędnego kodu odrzuci dane i trzeba będzie zacząć procedurę dodawania postaci od nowa. Jeśli sprawdzenie zwróci wynik błędny wyświetli stosowny komunikat oraz godzinę ostatniego sprawdzenia danych na API.

Weryfikacja wklejonego kodu odbywa się poprzez api.tibiadata, na którym odświeżenie danych może zająć do 5 minut, dlatego zmiany nie zawsze są zapisane od razu na API i sprawdzenie po wklejeniu poprawnego kodu może przez jakiś czas wyświetlać komunikat o błędzie. Należy wtedy poczekać do 5 minut i wykonać sprawdzenie ponownie.



Rysunek 37: Poprawne sprawdzenie kodu walidacyjnego w API po którym można przypisać postać do konta
(Źródło: własne)

Po poprawnym zweryfikowaniu kodu postać zostaje przypisana do konta w serwisie i użytkownik jest przenoszony do widoku, w którym zobaczyć można wszystkie postacie aktualnie przypisane do tego konta, razem z ważnymi danymi postaci, które będą wyświetlane później we wstawianym ogłoszeniu (Nazwa postaci, poziom postaci, profesja i serwer na którym postać się znajduje)



Rysunek 38: Widok wyświetlający postacie przypisane do aktualnego konta. (Źródło: własne)

Po udanym przypisaniu postaci do konta, możliwe jest wystawienie ogłoszenia dotyczącego danej postaci poprzez rozwinięcie menu „**Board**” i przejściu do odnośnika „**Create your ad**”. Po wyświetleniu się formularza w polu Description, można wpisać dowolny opis zawierające różne informacje, takie jak na przykład godziny w których dany użytkownik może być dostępny do wspólnej gry, znajomość języków czy inne dane według uznania (rys. 39). Z rozwijanego menu należy wybrać postać, której dotyczyć będzie ogłoszenie. W polach Minimum teammate level oraz Maximum teammate level wpisuje się minimalny i maksymalny poziom osoby, z którą użytkownik chce się dobrać do wspólnej gry. Po poprawnym wypełnieniu formularza (rys. 40) i kliknięciu „**Save**” ogłoszenie zostaje zapisane w bazie danych, przypisane do danego

konta i postaci, a użytkownik jest przenoszony do widoku wszystkich ogłoszeń opublikowanych z aktualnego konta (rys. 41). Zobaczyć tam można nazwę postaci której dotyczy dane ogłoszenie, jej profesję oraz fragment opisu. Po kliknięciu na daną pozycję w tabeli następuje przeniesienie do szczegółowego widoku danego ogłoszenia.

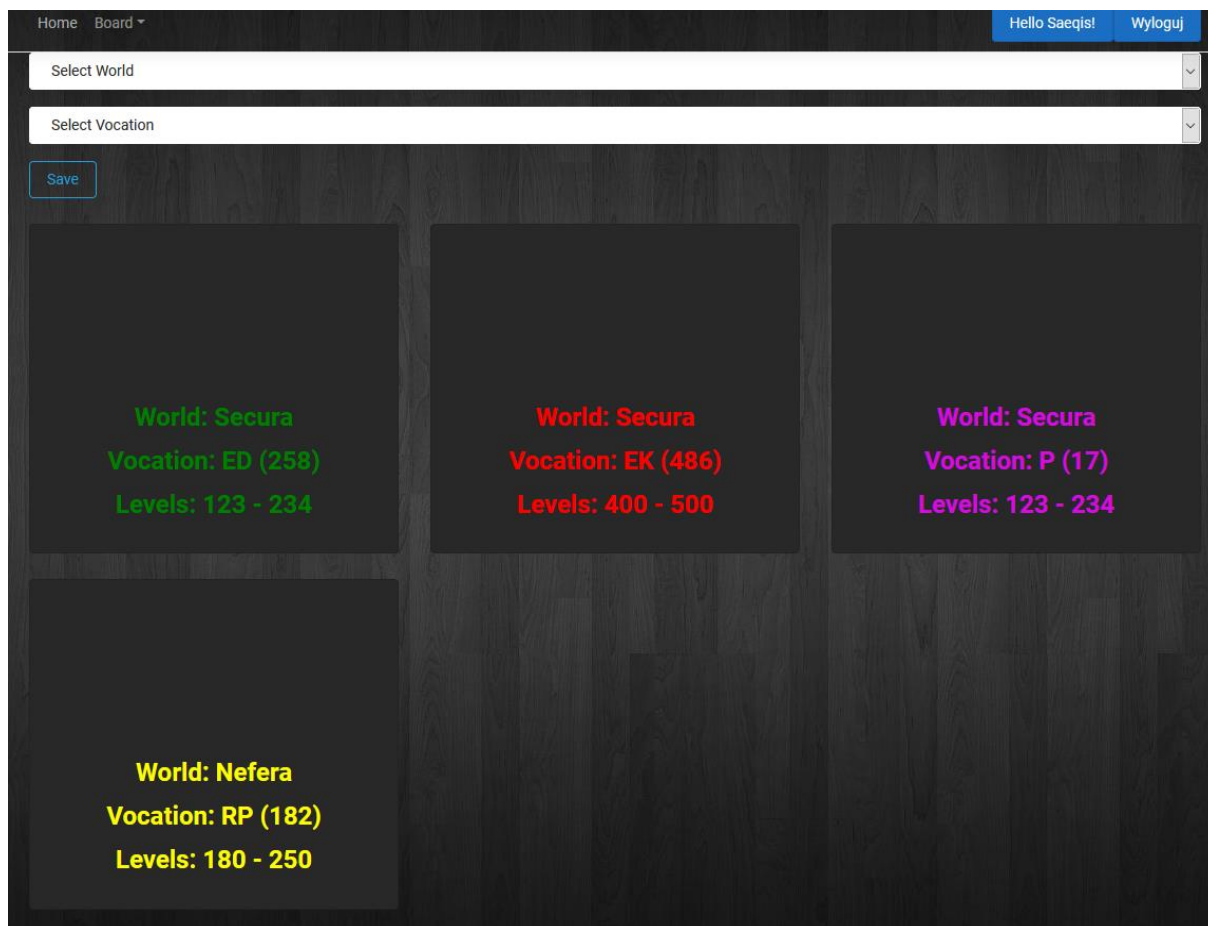
Rysunek 39: Formularz w widoku dodawania nowego ogłoszenia. (Źródło: własne)

Rysunek 40: Przykładowo wypełniony formularz tworzenia ogłoszenia. (Źródło: własne)

Character	Vocation	Advert text
Cetyryzyna Di Chlorowodorku (182)	RP	Template description

Rysunek 41: Widok ogłoszeń opublikowanych z danego konta. (Źródło: własne)

Po poprawnym opublikowaniu ogłoszenia można je zobaczyć na stronie głównej w formie karty wyświetlającej serwer (świat) gry, profesję i poziom postaci oraz poziomy jakimi osoba poszukująca jest zainteresowana.



Rysunek 42: Widok listy ogłoszeń po wstawieniu nowego ogłoszenia. (Źródło: własne)

Strona główna umożliwia również filtrowanie ogłoszeń wybierając profesję lub serwer gry na którym poszukiwana jest drużyna (rys. 43 i 44).



Rysunek 43: Rozwijane menu do wyboru filtrowania ogłoszeń. (Źródło: własne)



Rysunek 44: Widok ogłoszeń po wyborze jednej z opcji filtrowania. (Źródło: własne)

6 Podsumowanie

Spółeczność gry posługuje się głównie językiem angielskim. W języku tym, zarówno jest cały interfejs gry, przez co cała strona została napisana również w języku angielskim. Dzięki temu aplikacja może trafić do największego grona osób.

Większość czasu została poświęcona na poprawną integrację aplikacji z zewnętrznym API i zaprogramowaniu poprawnej weryfikacji postaci i pobierania informacji, co początkowo sprawiało problemy ze względu na tymczasowo nieważny certyfikat bezpieczeństwa na stronie przez czasem niemożliwe było łączenie się i operowanie na danych z API, co przełożyło się na ograniczoną ilość czasu potrzebnego do stworzenia funkcjonalności strony, z których większość opiera się na informacjach z API.

Dzięki zastosowaniu podejścia MVC i podziale kodu aplikacji na warstwy, serwisową i repozytorium, można bardzo łatwo rozszerzać funkcjonalności strony o dodatkowe elementy.

7 Spis rysunków

Rysunek 1: Zrzut ekranu z gry Zork	7
Rysunek 2: Zrzut ekranu z gry Multi-User Dungeon1	8
Rysunek 3: Zrzut ekranu z gry Habitat.....	8
Rysunek 4: Zrzut ekranu z gry Neverwinter Nights (1991)	9
Rysunek 5: Zrzut ekranu z gry Ultima Online	10
Rysunek 6: Zrzut ekranu z gry Everquest	11
Rysunek 7: Zrzut ekranu z wcześniejszej wersji gry Tibia.....	12
Rysunek 8: Zrzut ekranu z jednej z najnowszych wersji gry	13
Rysunek 9: Średnia dzienna liczba graczy w przedziale luty-czerwiec 2020	14
Rysunek 10: Wzrost obrotów i liczby zatrudnionych osób w CipSoft w latach 2001-2018	15
Rysunek 11: Liczba graczy z podziałem na kraj pochodzenia bazująca na użytkownikach portalu Guildstat	16
Rysunek 12: Diagram przypadków użycia dla aktora Użytkownik oraz Gość	18
Rysunek 13: Architektura Entity Framework.....	23
Rysunek 14: Logika rejestracji w serwisie	24
Rysunek 15: Logika logowania w kodzie serwisu	25
Rysunek 16: Model ogłoszenia wykorzystywany w aplikacji.....	26
Rysunek 17: Metoda odpowiadająca za wyświetlenie listy wszystkich ogłoszeń	27
Rysunek 18: Metoda odpowiadająca za tworzenie nowych ogłoszeń.....	27
Rysunek 19: Wzorzec asynchronicznego repozytorium z wykorzystaniem warstwy serwisu w aplikacjach ASP.NET MVC.....	28
Rysunek 20: Interfejs repozytorium implementujący sygnatury podstawowych metod CRUD.....	29
Rysunek 21: Przykładowe metody zawierające logikę biznesową w warstwie serwisu	29
Rysunek 22: Odpowiedź z api.tibiadata.com w wypadku nieistniejącej postaci	30
Rysunek 23: Fragment odpowiedzi API zawierającej dane dotyczące przykładowej istniejącej postaci	31
Rysunek 24: Pobieranie danych istniejących postaci w kodzie aplikacji.....	32
Rysunek 25: Dane postaci na api.tibiadata po wklejeniu kodu weryfikacyjnego, do opisu postaci widocznego jako wartość wewnątrz comment	32
Rysunek 26: Sprawdzenie poprawności kodu weryfikacyjnego wklejonego w opis postaci na stronie gry.....	32
Rysunek 27: Strona główna aplikacji.....	33
Rysunek 28: Wyświetlenie opisu po naciśnięciu myszką na kartę	33
Rysunek 29: Karta w widoku ze szczegółami ogłoszenia.....	34

Rysunek 30: Próba stworzenia ogłoszenia bez posiadanego konta i widok logowania	35
Rysunek 31: Pierwszy widok przypisywania postaci do konta.....	35
Rysunek 32: Komunikat błędu po próbie przypisania nieistniejącej postaci	36
Rysunek 33: Drugi widok podczas przypisywania postaci do swojego konta	36
Rysunek 34: Logowanie na konto gry na stronie internetowej	37
Rysunek 35: Widok listy postaci na stronie gry	37
Rysunek 36: Fragment szczegółowego widoku postaci w którym należy wkleić kod weryfikacyjny...	38
Rysunek 37: Poprawne sprawdzenie kodu walidacyjnego w API po którym można przypisać postać do konta	39
Rysunek 38: Widok wyświetlający postaci przypisane do aktualnego konta.....	39
Rysunek 39: Formularz w widoku dodawania nowego ogłoszenia.	40
Rysunek 40: Przykładowo wypełniony formularz tworzenia ogłoszenia.....	40
Rysunek 41: Widok ogłoszeń opublikowanych z danego konta	40
Rysunek 42: Widok listy ogłoszeń po wstawieniu nowego ogłoszenia	41
Rysunek 43: Rozwijane menu do wyboru filtrowania ogłoszeń	41
Rysunek 44: Widok ogłoszeń po wyborze jednej z opcji filtrowania	42

8 Źródła

- [1] <http://www.rankinggier.pl/gry-mmo-z-czym-to-sie-je/>
- [2] <https://bestreamer.com/gaming/most-played-mmorpg-2019/2/>
- [3] <https://en.wikipedia.org/wiki/Zork>
- [4] <https://en.wikipedia.org/wiki/MUD1>
- [5] <https://www.juegostudio.com/articles/history-of-mmo-games>
- [6] <http://www.gameclassification.com/EN/games/12005-Lucasfilms-Habitat/index.html>
- [7] [https://en.wikipedia.org/wiki/Habitat_\(video_game\)](https://en.wikipedia.org/wiki/Habitat_(video_game))
- [8] <https://gameplay.pl/news.asp?ID=56094>
- [9] <https://www.engadget.com/2010-05-04-the-game-archaeologist-and-the-ultima-prize-the-history.html>
- [10] <https://www.gry-online.pl/S026.asp?ID=9684>
- [11] <https://www.pcgamer.com/a-brief-history-of-mmo-games/>
- [12] <https://www.tibia-wiki.net/wiki/Historia>
- [13] <https://www.tibia.com/abouttibia/?subtopic=screenshots>
- [14] <https://www.engadget.com/2014-05-03-the-game-archaeologist-the-silent-world-of-tibia.html>
- [15] <https://www.cipsoft.com/en/14-company>
- [16] <https://guildstats.eu/online-counter?lang=nl>
- [17] <https://www.cipsoft.com/en/press/press-releases/165-result-financial-year-2018>
- [18] <https://www.cipsoft.com/en/press/press-releases/165-result-financial-year-2018>
- [19] <https://guildstats.eu/census>
- [20] <https://www.plukasiewicz.net/Artykuly/AspNetCoreFeatures>
- [21] <https://www.modestprogrammer.pl/szybkie-wprowadzenie-do-asp-net-core>
- [22] <https://testuj.pl/blog/10-pytan-programisty-temat-jezyka-csharp/>
- [23] <https://chrzanowski24.pl/strona-glowna/czym-jest-jezyk-c-oraz-gdzie-sie-go-uzywa-300434277>
- [24] <https://codecool.com/pl/wiedza/srodowisko-programistyczne-czym-jest-codecool-pl/>
- [25] <https://www.plukasiewicz.net/Artykuly/EntityFramework>
- [26] <https://www.codeproject.com/Articles/363040/An-Introduction-to-Entity-Framework-for-Absolute-B>

[27] <https://social.technet.microsoft.com/wiki/contents/articles/39621.asp-net-mvc-solution-architecture-project-with-async-repository-pattern-by-service-having-dependency-injection-by-unity-page-1.aspx>

[28] <https://api.tibiadata.com>

[29] <http://tibia.com>

9 Tabele

Tabela 1 PU-1 Załóż konto	19
Tabela 2 PU-2 Filtrowanie ogłoszeń po serwerze.....	19
Tabela 3 PU-3 Filtrowanie ogłoszeń po profesji.....	19
Tabela 4 PU-4 Zaloguj	20
Tabela 5 PU-5 Dodaj ogłoszenie.....	20