

POLITECHNIKA ŚWIĘTOKRZYSKA
Wydział Elektrotechniki, Automatyki i Informatyki

Anna Tutaj

Numer albumu: 070125

Opracowanie gry sieciowej w środowisku Microsoft .NET

Praca dyplomowa magisterska
na kierunku Informatyka

Opiekun pracy dyplomowej:
dr inż. Katarzyna Rutczyńska-Wdowiak
Katedra Systemów Informatycznych

Kielce, 2017

POLITECHNIKA ŚWIĘTOKRZYSKA
WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI I INFORMATYKI

Zatwierdzam:
Dziekan Wydziału

Rok akademicki: 2016/2017
Temat nr... 30MVI/1389/2016 N7

Dnia 15.05.2017r.

PROBIEŻKAM
ds. Kształcenia i Szkół Studenckich
na Studiach Stacjonarnych
Wydziału Elektrotechniki, Automatyki i Informatyki

dr inż. Andrzej Stobiecki

ZADANIE NA PRACĘ DYPLOMOWĄ
studiów drugiego stopnia na kierunku Informatyka

Wydano studentowi: **inż. Tutaj Anna**

I. Temat pracy: Opracowanie gry sieciowej w środowisku Microsoft.NET

II. Plan pracy:

1. Wprowadzenie.
2. Omówienie Transmission Control Protocol (TCP).
3. Omówienie aplikacji typu klient-serwer oraz wzorca MVC.
4. Charakterystyka programów narzędziowych wykorzystanych w aplikacji.
5. Opracowanie założeń programu komputerowego.
6. Wybrane rozwiązania implementacyjne.
7. Testy programu komputerowego.
8. Podsumowanie.

III. Cel pracy:

Celem pracy dyplomowej magisterskiej jest projekt i implementacja gry sieciowej z wykorzystaniem .NET Framework.

IV. Uwagi dotyczące pracy:

V. Termin oddania pracy: **30 czerwca 2017**

VI. Konsultant:.....

Kierownik Zakładu/Katedry

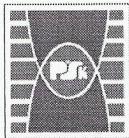
dr inż. Paweł Sitek

Opiekun pracy dyplomowej

dr inż. Katarzyna Rutczyńska-Wdowiak

Temat pracy dyplomowej celem jej wykonania otrzymałem(am):

Kielce, dnia 15.05.2017 r. Anna Tutaj
czytelny podpis dyplomanta



Politechnika Świętokrzyska

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI I INFORMATYKI

Kielce, dnia 15.06.2014r.

Anna Tutaj
Imię i nazwisko studenta

040125.....

nr albumu

ul. Jana Pawłowskiego 22 163 25-430 Kielce
Adres zamieszkania

Informatyka, systemy informacyjne, II, stacjonarne
Kierunek, specjalność, rok studiów, rodzaj studiów (stacjonarne/niestacjonarne)

dr inż. Krzysztof Rutkiewicz - Wdowski
Opiekun pracy dyplomowej inżynierskiej/magisterskiej*

OŚWIADCZENIE

Przedkładając w roku akademickim 2013/14. opiekunowi pracy dyplomowej inżynierskiej/magisterskiej*, powołanemu przez Dziekana Wydziału Elektrotechniki Automatyki i Informatyki Politechniki Świętokrzyskiej, pracę dyplomową inżynierską/magisterską* pod tytułem:

Opracowanie gry sieciowej na środowisku Microsoft .NET

oświadczam, że:

- 1) przedstawiona praca dyplomowa inżynierska/magisterska* została opracowana przeze mnie samodzielnie, stosownie do wskazówek merytorycznych opiekuna pracy,
- 2) przy wykonywaniu pracy dyplomowej inżynierskiej/magisterskiej* wykorzystano materiały źródłowe, w granicach dozwolonego użytku wymieniając autora, tytuł pozycji i miejsce jej publikacji,
- 3) praca dyplomowa inżynierska/magisterska* nie zawiera żadnych danych, informacji i materiałów, których publikacja nie jest prawnie dozwolona,
- 4) przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem stopnia zawodowego/naukowego w wyższej uczelni,
- 5) niniejsza wersja pracy jest identyczna z załączoną treścią elektroniczną (na CD i w systemie Archiwum Prac Dyplomowych).

Przyjmuję do wiadomości, iż w przypadku ujawnienia naruszenia przepisów ustawy o prawie autorskim i prawach pokrewnych, praca dyplomowa inżynierska/magisterska* może być unieważniona przez Uczelnię, nawet po przeprowadzeniu obrony pracy.

Zostałem uprzedzony:

- 1) o odpowiedzialności karnej wynikającej z art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t. j. Dz. U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, videogram lub nadanie.”,
- 2) odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t. j. Dz. U. z 2012 r. poz. 572, z późn. zm.) „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwany dalej sądem koleżeńskim”

Prawdziwość powyższego oświadczenia potwierdzam własnoręcznym podpisem.

Anna Tutaj
czytelny podpis studenta

* niepotrzebne skreślić

Streszczenie

Niniejsza praca dyplomowa zawiera opis procesu projektowania gry sieciowej. Do projektowania użyto Visual Studio 2015 wraz z .NET framework, bazę danych MS SQL oraz narzędzia służące do rysowania grafiki - Adobe Photoshop oraz Paint Tool SAI. Inspiracją do zaprojektowania gry karcianej była gra pt. „Dobble”. Gracz posiada możliwości: rejestracji, logowania, tworzenia pokoju do gry o wybranych parametrach, komunikowania się z innymi graczami, przeprowadzania rozgrywki, zdobywania punktów.

Słowa kluczowe: gra, sieciowa, Visual, Studio, MS, SQL, baza, danych, TCP, MVC

Summary

This diploma thesis contains a description of a process of designing a network game. Visual Studio 2015 with .NET framework, MS SQL database and tools for drawing graphics - Adobe Photoshop and Paint Tool SAI have been used in order to design game. Inspiration to project a card game was a game entitled „Dobble”. Player can: sign in, log in, create an room with selected parameters, chatting, playing, gain points.

Keywords: game, network, Visual, Studio, MS, SQL, database, TCP, MVC

SPIS TREŚCI

1. WSTĘP.....	10
2. PRZEGŁĄD WSPÓŁCZESNYCH GIER SIECIOWYCH.....	12
2.1 Omówienie gatunków gier komputerowych.....	12
2.2 Przegląd gier sieciowych	14
3. NARZĘDZIA WYKORZYSTANE DO ZAPROJEKTOWANIA GRY DABBLE.....	19
3.1 Visual Studio.....	19
3.1.1 Instalacja Visual Studio 2015.....	19
3.1.2 Interfejs Visual Studio 2015.....	21
3.2 Programy grafiki rastrowej.....	22
3.2.1 Adobe Photoshop CS6.....	22
3.2.2 Paint Tool Sai.....	23
3.3 Microsoft SQL Server 2012.....	24
4. ZAŁOŻENIA, STRUKTURA I REALIZACJA GRY.....	26
4.1 Założenia projektowe.....	26
4.2. Architektura aplikacji.....	28
4.2.1 Architektura typu klient – serwer	28
4.2.2 Transmission Control Protocol (TCP).....	29
4.2.3 Wzorzec Model View Controller (MVC).	32
4.3 Baza danych.....	37
4.4 Serwer.....	42
4.4.1 Prezentacja widoku.....	42
4.4.2 Algorytm generowania talii i system rozdawania kart.....	43
4.4.3 Opracowanie komunikatów.....	50
4.5 Klient.....	53
4.5.1 Prezentacja widoków.....	53
4.5.2 Omówienie funkcji kontrolera.....	59
5. TESTY GRY SIECIOWEJ.....	62
5.1 Warunki techniczne przeprowadzonych testów.....	62
5.2 Przeprowadzane testy.....	62
6. ZAKOŃCZENIE.....	78
LITERATURA.....	79

1. WSTĘP

Temat pracy magisterskiej został wybrany ze względu na wzrastające zainteresowanie zagadnieniem gier zarówno komputerowych, jak i planszowych. Łącząc te dwie kategorie można zaprojektować na podstawie gry karcianej grę komputerową. Niniejsza praca jest przykładem na opracowanie gry komputerowej opartej na grze karcianej. Przełożenie zasad i pomysłu rozgrywki na wersję komputerową pozwoliło opracować własną grę pt. „Dabble”. Według raportu e-zakupów w 2015 opracowanym przez Ceneo.pl wraz z partnerami - e-Commerce Polska Izba Gospodarki Elektronicznej oraz TNS (który w Polsce jest jedną z czołowych firm w dziedzinie custom research) w kategorii gry planszowe z dziedziny multimedia Rebel – Dobble otrzymało miano najlepszego produktu [5].

Wnioskując po wyborze konsumentów ten rodzaj rozgrywki odpowiada potrzebom graczy. Nie ma ograniczenia wiekowego, ani nie jest skierowana do określonej płci, przez co jest uniwersalna. Ważnym aspektem jest towarzystwo, ponieważ jest to gra skierowana dla przynajmniej dwóch osób. W dzisiejszych czasach ludzie potrzebują przebywania z kimś, a z drugiej strony są coraz bardziej ograniczeni czasowo oraz poprzez odległość. Trudniej jest znaleźć czas, by się z kimś spotkać, zatem rozwiązaniem jest komunikowanie się poprzez sieć. Zaprojektowana gra na potrzeby pracy magisterskiej łączy dwa aspekty - zalety najpopularniejszej gry planszowej Rebel Dobble i szerokie możliwości aplikacji sieciowej.

Projekt magisterski oprócz spełnienia wcześniej omawianych potrzeb użytkownika pozwolił na sprawdzenie i rozwijanie umiejętności autora z zakresu programowania, projektowania, korzystania z bazy danych, oraz opracowywania grafiki.

Cel pracy obejmował architekturę, projekt i realizację gry sieciowej w środowisku Microsoft .NET z wykorzystaniem bazy danych MS SQL. Korzystano z języka C# w edytorze Visual Studio 2015. Ponadto do wykonywania własnych grafik zostały wykorzystane programy Adobe Photoshop CS6 oraz Paint Tool SAI.

Rozdział 1 stanowi wprowadzenie w tematykę pracy, sformułowano w nim cel i zakres pracy. Uzasadniono dlaczego wybrano opracowanie gry sieciowej jako temat pracy.

Rozdział 2 to omówienie istniejących rodzajów gier: zręcznościowych, przygodowych, fabularnych, strategicznych, symulacyjnych, sportowych, logicznych, edukacyjnych oraz wymienienie ich gatunków, których jest ponad 40. Zawiera przegląd istniejących na rynku gier sieciowych oparty o ranking najpopularniejszych gier PC. Dodatkowo omówione zostały gry przypominające projektowaną grę Dabble - Gwint Wiedźmińska gra karciana i Heartstone ze względu na prowadzenie rozgrywki online poprzez pojedynek z innymi graczami za pomocą kart, a także ze względu na system pokoi o różnych parametrach i czat serwis www.kurnik.pl.

Rozdział 3 stanowi opis narzędzi programistycznych wykorzystanych do wykonania projektu. Każdy podrozdział został podzielony na trzy części, z których pierwsza stanowi wprowadzenie do programu, druga część opisuje proces instalacji programu, a trzecia omówienie interfejsów programów. Do zaprojektowania gry zostały użyte: Visual Studio, które udostępnia środowisko IDE, system zarządzania bazą danych Microsoft SQL Server 2012, programy grafiki rastrowej Adobe Photoshop i Paint Tool SAI.

Rozdział 4 dotyczy najważniejszej części pracy, ponieważ zawiera założenia projektowe, rozwiązania implementacyjne i realizację gry, a także prezentuje działanie aplikacji na zrzutach ekranu. Dodatkowo zawiera zagadnienia teoretyczne z zakresu wzorca MVC, protokołu TCP, architektury klient – serwer. Przedstawia model bazy danych oraz przykładowo wprowadzone rekordy. Zawiera opis algorytmu generowania talii oraz systemu rozdawania kart.

Rozdział 5 to testy gotowego już produktu z uwzględnieniem warunków technicznych komputerów, na których były uruchomiane aplikacje serwera oraz klientów. Zawiera zrzuty ekranów prezentujące możliwości aplikacji zarówno serwera, jak i klienta. Pokazuje jak przebiega proces przykładowej rozgrywki, a także jak wyglądają okna rozgrywek dla różnych wartości parametrów wybieranych podczas zakładania pokoju.

Rozdział 6 jest zakończeniem i podsumowaniem pracy. Zwraca uwagę na zalety oraz wady aplikacji. Oprócz tego zawiera szersze spojrzenie na projekt i przemyślenia dotyczące dalszego jego rozwoju.

2. PRZEGŁĄD WSPÓŁCZESNYCH GIER SIECIOWYCH

2.1 Omówienie gatunków gier komputerowych

W dzisiejszych czasach zarówno gry, jak i filmy, książki, czy muzyka są rodzajem produktów, które często nie mogą być jednoznacznie sklasyfikowanie. Wyróżnić można 8 rodzajów i ponad 40 gatunków gier komputerowych. Są to m.in. gry [4]:

- Zręcznościowe: gry arkadowe, platformówki, bijatyki, strzelanki, FPSy, wyścigi, filmy interaktywe, gry muzyczne.
- Przygodowe: tekstówki, point'n'click (klasyczne gry przygodowe).
- Fabularne: roguelike, MUDy (Multi-User Dungeon), hack'n'slashe (action-RPG), RPGi (włącznie z dungeon crawlery), spac-simy (symulatory kosmiczne), MMORPGi.
- Strategiczne: gry turowe, RTSy (strategia czasu rzeczywistego, w tym RTWG), MMORTSy, strategie przeglądarkowe (browser-based), gry taktyczne, strategie ekonomiczne, managery sportowe.
- Symulacyjne: symulatory lotnicze, symulatory samochodowe, symulatory społeczne, symulatory życia zwierząt, ubieranki i makeover'y.
- Sportowe: ze względu na wybraną dyscyplinę sportową.
- Logiczne: gry planszowe, gry karciane, logiczno-zręcznościowe, wojny rdzeniowe i pochodne.
- Edukacyjne: ten rodzaj gier łączy silny aspekt edukacyjny.

Jak już wcześniej wspomniano, trudno jest zaklasyfikować wybraną grę do jednego gatunku. Powstają także hybrydy międzyrodzajowe np. fabularno-strategiczne, czy symulacyjno-zręcznościowe. Wraz z upływem czasu wybrana hybryda może zostać włączona na poziomie gatunku. Niemniej jednak żadna klasyfikacja nie może być zamknięta i próba klasyfikacji danego tytułu jest często jedynie umowną kwestią.

Omówienie rodzajów gier skupi się na ośmiu rodzajach. Pierwszym są gry symulacyjne, które charakteryzują się głównie tym, że w wirtualnej rzeczywistości odzwierciedlają charakter rzeczywistych zdarzeń. Mogą skupiać się na jednej czynności np. jazda samochodem, lot samolotem albo odzwierciedlać bardziej złożone procesy, takie jak budowa miasta (np. seria Sim City), paru rozrywek (np. RollerCoaster Tycoon), czy symulacja całego życia (np. seria The Sims).

Drugi rodzaj gier, to gry strategiczne, które skupiają się na logice i wymagają perspektywicznego myślenia. Wygrana nie zależy od szczęścia, czy przypadku, ale od strategii, umiejętności planowania i przewidywania skutków dokonywanych wyborów. Strategie wojenne wymagają umiejętności dowodzenia i kierowania oddziałami zbrojnymi podczas konfliktu zbrojnego. Gracz opowiada się za jedną ze stron. RTSy to gry czasu rzeczywistego, przez co wymagają więcej zręczności niż planowania taktycznego. Strategia w grach ekonomicznych polega na planowaniu i zarządzaniu firmą, parkiem rozrywki, miastem, państwem, ośrodkiem wypoczynkowym, biorąc pod uwagę uwarunkowania wpływające na rynek finansowy i populację ludzi. Zadaniem gracza jest inwestowanie w celu osiągnięcia jak najlepszego efektu finansowego. Trzeci rodzaj gier to gry zręcznościowe, gdzie należy się wykazać nie tylko refleksem, ale i sprawnością manualną poprzez operowanie przyciskami klawiatury, joystika, czy innego kontrolera. Gry zręcznościowe charakteryzują się dynamiczną rozgrywką, w których akcja toczy się szybko. Myślenie nie jest ważnym aspektem, jak w przypadku gier strategicznych. Liczy się szybkość działania. Klasycznym przykładem jest wcielanie się w bohatera i widok kamery z perspektywy pierwszej osoby (FPP – ang. First-person perspective), jak w przypadku Call of Duty, Counter Strike, Medal of Honor. Gracze utożsamiają się z bohaterem, oglądając świat jego oczami. Ułatwia to wcielanie się w daną rolę i eksplorację otoczenia. Czwarty rodzaj to gry przygodowe, które często zawierają elementy zagadek, rebusów. Są zbliżone do gier logicznych. Różnią się często widokiem kamery – w przypadku gier przygodowych jest to zwykle FPP, które ułatwia wcielanie się w rolę. Dodatkowo takie gry posiadają rozbudowaną fabułę, która często jest skomplikowana i zawiera wątki detektywistyczne. Celem gracza jest odwiedzanie miejsc, rozmowa z postaciami, zdobywanie informacji, przedmiotów. Seria Tomb Raider jest przykładem gry przygodowej z kamerą TPP (Third-person perspective). Gry logiczne natomiast skupiają się wyłącznie na opracowaniu rozgrywki wymagającej myślenia w rodzaju gry w szachy. Szóstym rodzajem są gry fabularne tzw. RPG (ang. Role play game). Jak sama nazwa wskazuje, wymogiem jest, aby gracz wcielił się w postać. Często są to magowie, wojownicy, czy postacie świata fantasy. Oprawa graficzna jest bogata, a gracz ma swobodę poruszania się po świecie, co przekłada się na zwiększenie poziomu realności wykreowanego świata. Przykładem RPG jest seria polskiej produkcji Wiedźmin, gdzie akcja gry toczy się w uniwersum zwanym światem wiedźmina, wykreowanym przez Andrzeja Sapkowskiego. Następnym rodzajem gier są gry edukacyjne. Odznaczają się rozwijaniem, uczeniem, kształtowaniem postaw graczy. W środowiskach dydaktycznych gry edukacyjne są doceniane, ponieważ łączą elementy zabawy z nauką, pozwalają rozwijać umiejętności, wyobraźnię. Ostatnim omawianym rodzajem gier są gry sportowe, które skupione są na jednej bądź wielu dyscyplinach sportowych. Rozwijają pasję sportową oraz zdolności manualne. Popularne są

wyścigi, w których gracz wybiera środek transportu – często samochód lub motocykl– a gracze prześcigają przeciwników, okrążają trasy, zdobywają miejsca na podium. Kultowym przykładem jest seria Need for Speed. Gry sportowe są podobne do gier zręcznościowych z uwagi na wymóg szybkości i refleksu. Gatunek MMORPG (ang. Massively Multiplayer Online Role Playing Game) to odmiana gier wieloosobowych sieciowych. Umożliwiają rozgrywkę wielu osobom połączonych ze sobą przy pomocy sieci Internet. Podobnie jak w przypadku gier fabularnych gracz wciela się w postać i widzi zaprojektowany świat oczami swojego bohatera. Standardowo zawierają system rozwijania postaci uzależniony od zdobytego doświadczenia, system klanów dla organizacji graczy w późniejsze społeczności, administracji serwera/ GM (ang. Game Master), którzy nadzorują i kontrolują zachowania graczy, postacie NPC (ang. non-player character), czyli bohaterów niezależnych.

2.2 Przegląd gier sieciowych

Tematyka gier sieciowych jest szeroka i rozwija się coraz prężniej. Zwykłe gry desktopowe przeznaczone dla jednego użytkownika odchodzą nieco na dalszy plan z uwagi na przywiązanie ludzi do Internetu oraz szeroki i łatwy dostęp do tego medium w dzisiejszych czasach.

Co miesiąc Newzoo wraz z Overwolf publikują rankingi najpopularniejszych gier PC w Stanach Zjednoczonych i Europie [6]. Ranking przedstawiony w tabeli 2.1 jest oparty na liczbie niepowtarzalnych sesji w miesiącu kalendarzowym wśród milionów entuzjastów gier, którzy korzystają z Overwolfa, czyli oprogramowania, które integruje narzędzia mediów społecznościowych takich jak Facebook, Twitter, Skype i MSN w celu dzielenia się doświadczeniem z gry i zapewnienia kontaktu ze znajomymi w trakcie rozgrywki. Biorąc pod uwagę temat pracy magisterskiej bliżej zostanie omówiony Heartstone, ponieważ jest podobny do wykonanego projektu – gry Dabble – ze względu na prowadzenie rozgrywki online poprzez pojedynek z innymi graczami za pomocą kart, co przekłada się na pozycję w rankingu. Jednak w przypadku gry ze świata Warcrafta mechanika jest zupełnie inna. Polega na tworzeniu własnej talii kart z setek dostępnych kart. Etapy gry obejmują: wybór bohatera należącego do danej klasy, wybór określonej liczby kart z talii, wymianę do trzech kart, wystawienie na stół karty, zakończenie tury i dobranie karty. Rozgrywka kończy się, gdy któryś z graczy będzie miał zero albo mniej punktów życia. Za wygraną grę rankingową gracz otrzymuje gwiazdki, a za przegrany zostają one odjęte, co odzwierciedla się na pozycję w rankingu. Premiera odbyła się 11 marca 2014 roku, a później wydano także wersję mobilną na Androida, Windows 8 i iOSa. Przykładowy zrzut ekranu widnieje na rys. 2.1 [7].

Tabl. 2.1. Najpopularniejsze gry PC

Logo	Pozycja	Nazwa	Wydawca
	1	League of Legends	Riot
	2	Hearthstone	Blizzard Entertainment
	3	Minecraft	Mojang
	4	Counter-Strike: Global Offensive	Valve
	5	World of Warcraft	Blizzard Entertainment
	6	Overwatch	Blizzard Entertainment
	7	World Of Tanks	Wargaming
	8	Tom Clancy's Rainbow Six: Siege	Ubisoft
	9	Rocket League	Psyonix
	10	Grand Theft Auto: San Andreas Multiplayer	Rockstar Games



Rys. 2.1. Screenshot z gry Heartstone

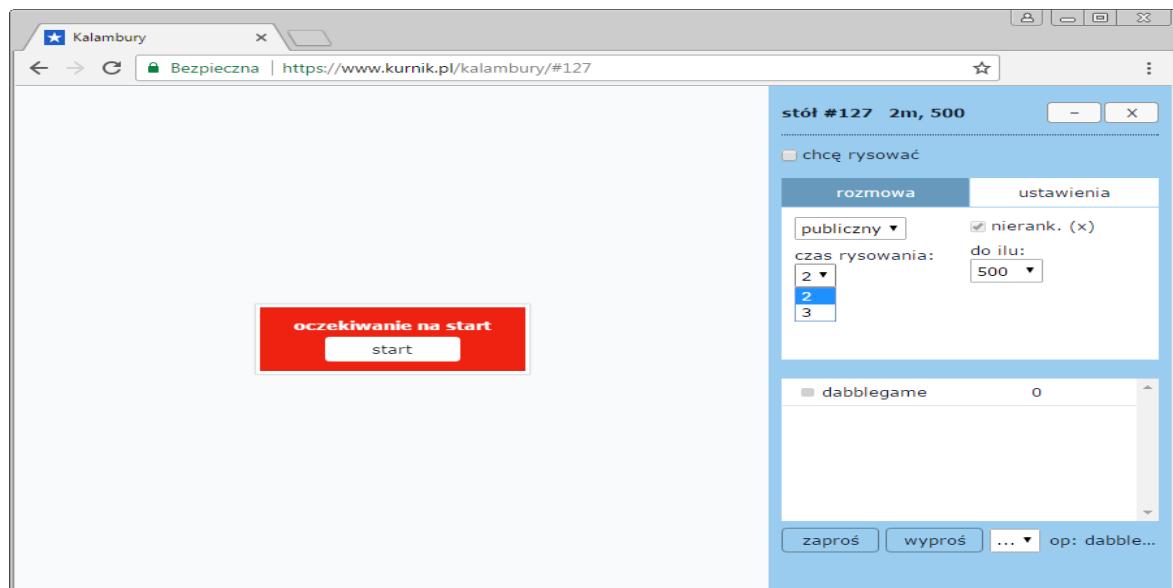
Inną grą karcianą jest Gwint: Wiedźmińska gra karciana wydana przez CD Project, która premierę Open Beta miała 24 maja 2017 roku. Rozgrywka imituje starcie pomiędzy dwoma armiami. Plansza widoczna na rys. 2.2 podzielona jest na dwie połowy, a każda z nich na trzy części [8]. Licząc od środka są to kolejno rzędy przeznaczone dla: jednostek walczących wręcz, formacji strzelających, machin oblężniczych. Talia kart ma określone ilości, a do każdej karty jest przypisana linia, gdzie można taką kartę umieścić. Dodatkowo każda frakcja posiada dowódcę. Podczas tury gracz może wykonać jeden z trzech ruchów: wyłożyć kartę na pole bitwy, użyć mocy dowódcy albo spasować. Pas oznacza poddanie bitwy. Jedna runda składa się z maksymalnie trzech bitew, bo żeby wygrać rundę należy zyskać dwa zwycięstwa.

Warta uwagi jest również strona www.kurnik.pl. Oferuje kilka rodzajów gier. Z gier planszowych znajdują się tam m.in. szachy, warcaby, kółko i krzyżyk, monopol, młynek. Z gier karcianych np. 3-5-8, brydż, dureń, kierki, makao, pan, piki, tysiąc, a z różnych gier domino, kalambury, kości, mahjong. Po wybraniu gry następuje logowanie bądź gra jako użytkownik niezarejestrowany – gość. Następuje wówczas przekierowanie do tworzenia nowego stołu o określonych parametrach, co jest widoczne na rys. 2.3. Parametry dotyczą czasu trwania tury, trybu gry – rankingowego, albo normalnego, liczby punktów potrzebnych do zwycięstwa, włączenie lub wyłączenie zakazu cofania. Oprócz założenia własnego stołu gracz może przeglądać i dołączyć do wybranego stołu. Do prywatnych stołów gracz może dołączyć jedynie wtedy, gdy zna wymagane hasło. Stoły prywatne są przeznaczone dla graczy, którzy się znają i nie chcą prowadzić rozgrywki z

przypadkowymi osobami. Zdobyte punkty przekładają się na indywidualny ranking dla konkretnego rodzaju gry. Oferowane gry zostały napisane przy użyciu HTML5. Serwis został utworzony 15.06.2001, a wraz z czasem zostały dodawane kolejne rodzaje gier, kupowane nowe serwery, dodawane nowe opcje ustawień w grach, organizowane turnieje.



Rys. 2.2 Screenshot z gry Gwent



Rys. 2.3. Screenshot z serwisu kurnik.pl

W podsumowaniu przeglądu gier można zauważać, że każdy tytuł oferuje inny rodzaj rozgrywki i został inaczej zaprojektowany. Aplikacje mogą być webowe, desktopowe albo mobilne. Do istniejącej już aplikacji można napisać jej mobilną wersję. Przedstawione dwie gry karciane Heartstone oraz Gwent to gry desktopowe. Należy mieć je zainstalowane na komputerze, żeby móc z nich korzystać. Projekt magisterski również jest grą tego typu. Pomysł systemu tworzenia pokoi,

który jest typowy dla aplikacji webowych (wymagających uruchomienia w przeglądarce internetowej), został wykorzystany w zaprojektowanej aplikacji desktopowej. Więcej o założeniach projektowych zawiera rozdział 4.1.

3. NARZĘDZIA WYKORZYSTANE DO ZAPROJEKTOWANIA GRY DABBLE

3.1 Visual Studio

Aplikacja Visual Studio udostępnia środowisko IDE (ang. Integrated Development Environment – zintegrowane środowisko programistyczne) dla systemu Windows i komputerów Mac. Udostępnia komfortowe edytowanie kodu źródłowego, kompilowanie go, tworzenie zasobów z rodzaju formatek WinForms, komunikowanie się z bazą danych.

3.1.1 Instalacja Visual Studio 2015

Dostępne są różne wersje Visual Studio, które zostały umieszczone w tabeli 3.1 [9]. Różnią się głównie co do wymaganego miejsca na dysku twardym.

Tabl. 3.1. Wymagania systemowe wybranych wersji Visual Studio 2015

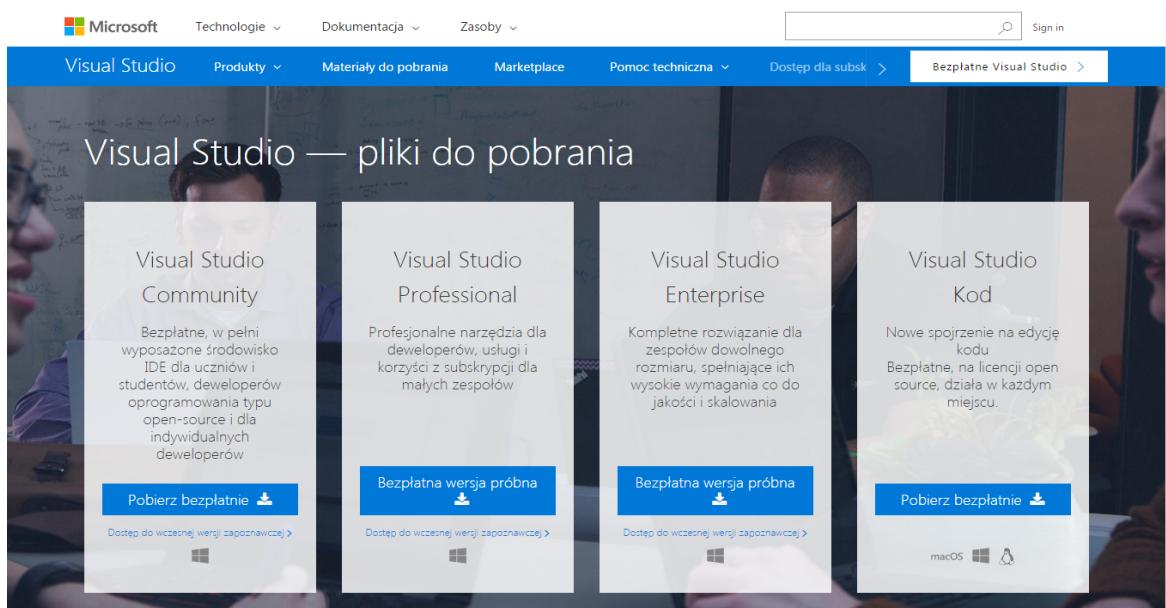
Wersja/ Wymagania sprzętowe	Visual Studio Community 2015	Visual Studio Language Pack 2015	Visual Studio Professional 2015
Procesor	1,6 GHz lub szybszy	1,6 GHz lub szybszy	1,6 GHz lub szybszy
Pamięć RAM	1GB (1,5 GB w przypadku maszyny wirtualnej)	1GB (1,5 GB w przypadku maszyny wirtualnej)	1GB (1,5 GB w przypadku maszyny wirtualnej)
Dostępne miejsce na dysku twardym o szybkości 5400 obr./min	4 GB	600 GB	10 GB
Karta wideo	obsługująca technologię DirectX 9	obsługująca technologię DirectX 9	obsługująca technologię DirectX 9
Systemy operacyjne	Windows 10 Windows 8.1 Windows 8 Windows 7 z dodatkiem SP1 Windows Server 2012 z dodatkiem R2 Windows Server 2012 Windows Server 2008 R2 SP1	Windows 10 Windows 8.1 Windows 8 Windows 7 z dodatkiem SP1 Windows Server 2012 z dodatkiem R2 Windows Server 2012 Windows Server 2008 R2 SP1	Windows 10 Windows 8.1 Windows 8 Windows 7 z dodatkiem SP1 Windows Server 2012 z dodatkiem R2 Windows Server 2012 Windows Server 2008 R2 SP1

Różnice w możliwościach edycji zamieszczone są w tabeli 3.2. Edycja Express jest najuboższa, za to Professional oferuje najwięcej. Jest jeszcze Team System editions, które zawiera dodatkowe rozszerzenia takie jak Code Analysis, czy tworzenie programów dla procesorów Itanium.

Tabl. 3.2 Porównanie możliwości różnych edycji Visual Studio

Edycja/ cecha	Visual Studio Express	Visual Studio Standard	Visual Studio Professional
Rozszerzenia	minimalne	tak	tak
Zewnętrzne narzędzia	minimalne	tak	tak
Refaktoring	ograniczony	tak	tak
Debugowanie	ograniczone	tak	tak
Tworzenie programów dla platform 64 bitowych	nie	tak	tak
Tworzenie programów dla Windows Phone	Wymaga instalacji Windows Phone Framework	tak	tak
Visual Studio Tools For Office	nie	nie	tak

Aby pobrać instalator Visual Studio należy wykonać kilka prostych kroków. Pierwszym krokiem jest wejście na stronę internetową z plikami instalacyjnymi. Link do nich został zamieszczony w odnośniku [9]. Następnie należy wybrać z menu „Materiały do pobrania”. Widok okna po wybranym tej opcji pokazuje rys. 3.1.

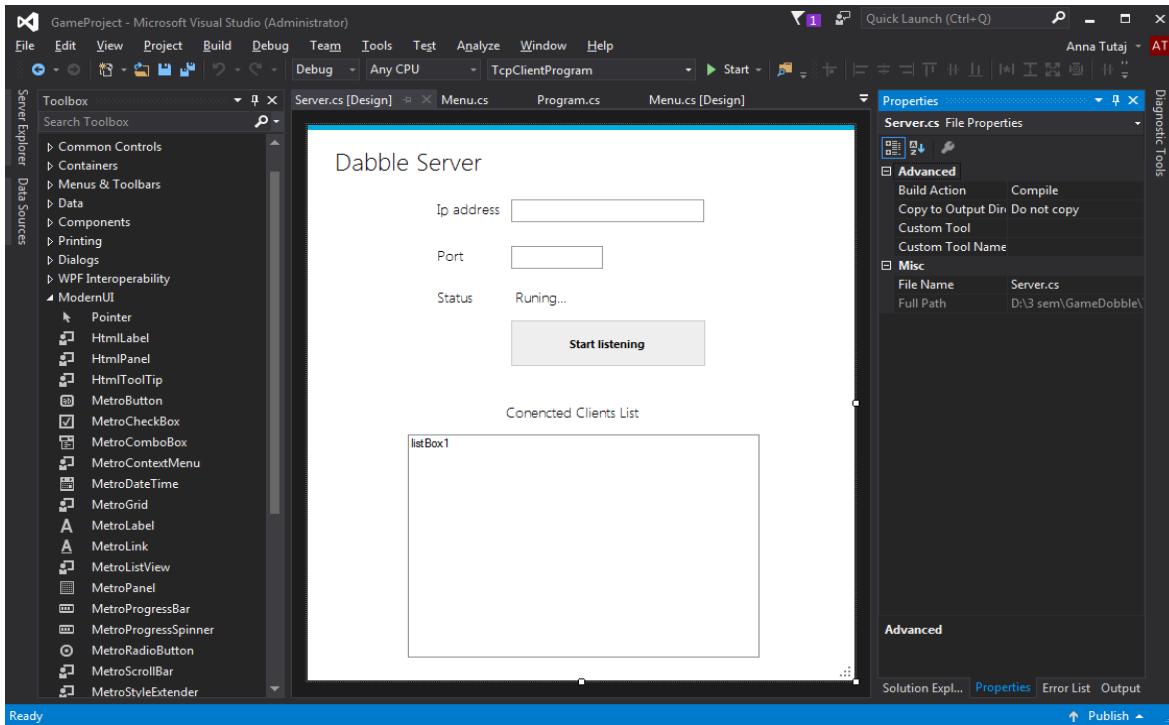


Rys. 3.1. Materiały do pobrania

Wersja Community jest bezpłatna i zalecana, żeby zapoznać się ze środowiskiem. Po kliknięciu na guzik „Pobierz bezpłatnie” plik instalacyjny .exe zostanie ściągnięty i należy przejść przez standardowe kroki instalatora.

3.1.2. Interfejs Visual Studio 2015

Interfejs Visual Studio składa się z wielu elementów, co jest pokazane na rys. 3.3. Każdy może według uznania poustawiąć okna, a najczęściej używane zostaną omówione.



Rys. 3.3 Interfejs Visual Studio 2015

Main Menu – znajduje się na samej górze okna. Dzięki niemu można tworzyć, otwierać, dodawać projekty, zarządzać widokami, projektem, budować, debugować projekt. W narzędziach (Tools) znajdują się opcje odnośnie połączenia z bazą, ustawieniem edytora,

- **Toolbar** – znajduje się poniżej Main Menu. Zawiera ikony z najczęściej używanymi opcjami takimi jak nowy projekt, debudowanie, szukanie,
- **Toolbox** – umiejscowiony jest z lewej strony. Zawiera wszystkie dostępne kontrolki, których się używa przy projektowaniu wyglądu aplikacji. Znaleźć tu można m.in. label, button, checkBox, grid, textBox, dateTime,
- **Designer** – zajmuje środkowy obszar. Na rys. 3.3 widać w nim otwartą WinForm. Edytuje się w nim otwarty plik, może to być też kod źródłowy,
- **Solution Explorer** – otwarta zakładka po prawej stronie. W niej widać zawartość i hierarchię całej solucji,
- **Properties** – kolejna zakładka. Służy do zarządzania właściwościami pliku,
- **Error List** – zawiera błędy i ich opisy,

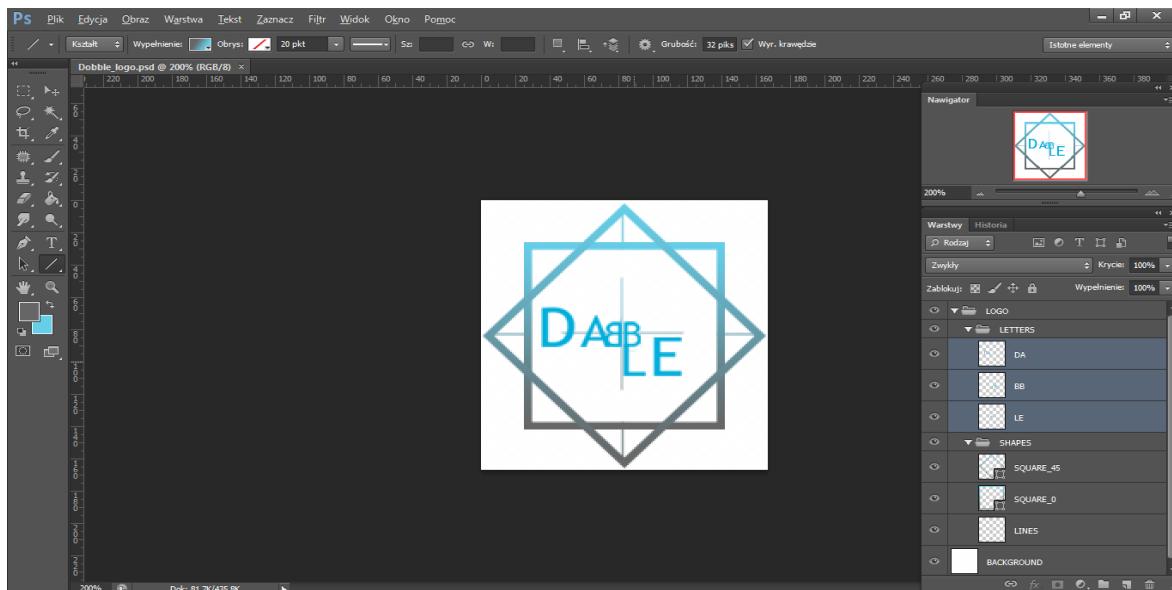
- **Output** – wypisywane jest tu wyjście.

3.2 Programy grafiki rastrowej

Grafika rastrowa prezentuje obraz za pomocą siatki pikseli odpowiednio pokolorowanych na urządzeniu wyjściowym takim jak monitor komputera, ekran telefonu, drukarka. Obok grafiki rastrowej istnieje grafika wektorowa, która charakteryzuje się tym, że obraz nie jest opisywany jako poszczególne punkty, a jest zdefiniowany matematycznie przy użyciu np. krzywych, punktów, czy wielokątów. Grafika wektorowa dlatego może być swobodnie skalowana bez utraty jakości obrazu.

3.2.1 Adobe Photoshop CS6

Program Adobe Photoshop CS6 oferuje najnowocześniejsze mechanizmy obróbki i tworzenia grafiki 2D. Pozwala na retusz zdjęć, tworzenie własnych grafik, a także wspiera produkcję filmów. Rysunek 3.4 przedstawia interfejs programu.



Rys. 3.4. Interfejs Adobe Photoshop CS6

Okna można ustawać według własnego uznania. Widok obszaru roboczego można swobodnie powiększać, chowając niepotrzebne okna w trakcie pracy. Interfejs jest przejrzysty i czytelny, a do jego najważniejszych elementów należą:

- **Menu główne** – zawiera szereg przydatnych opcji m.in:
 - **Plik** – standardowe opcje dotyczące tworzenia, zapisywania, wczytywania pliku, a także eksportowania np. klatek wideo do warstw i importowania do formatu np. ścieżek do Illustratora,
 - **Edycja** – cofanie, wycinanie, kopiowanie, wklejanie, a także przekształcania swobodne, ustawiania skrótów klawiszowych,

- **Obraz** – oprócz ustawiania rozmiaru obrazu i kadrowania posiada pod zakładką dopasowania szereg przydatnych funkcji do obróbki koloru obrazu takie jak: jasność/kontrast, poziomy, krzywe, ekspozycja, jaskrawość, barwa/nasyście, balans kolorów, mieszanie kanałów, mapa gradientu, kolor selektywny, cienie/podświetlenie, wariacje itd.
- **Filtr** – zawiera wbudowane filtry modyfikujące obraz, a także pikselowanie, rozmycie, szum, wyostrzanie, czy znieksztalcenie przydatne przy korekcji zdjęć,
- **Narzędzia** – zawiera ikony z narzędziami takimi jak: zaznaczenie prostokątne, lasso, róźdżka, zakraplacz, pędzel, łatka, gumka, wypełnianie kolorem, tekst, lupa,
- **Opcje** – prezentuje dostępne modyfikowalne ustawienia wybranego narzędzia,
- **Przestrzeń robocza** – stanowi obszar w którym obrabia się grafikę,
- **Historia** – udostępnia łatwy dostęp do wcześniej wykonanych działań,
- **Nawigator** – dzięki niemu można przybliżać i oddalać grafikę, a także poruszać się po niej.
- **Warstwy** – podgląd warstw wraz z ich trybem, stopniem krycia i wypełnienia

Produkty Adobe można pobierać ze strony umieszczonej w odnośniku [10]. Są to darmowe triale. Podczas zakupu Photoshopa CC obecnie (stan na 31.05.2017 rok) koszt wynosi 19,99 € /miesiąc + 23% VAT czyli razem 24,59 € /miesiąc .

3.2.2 Paint Tool SAI

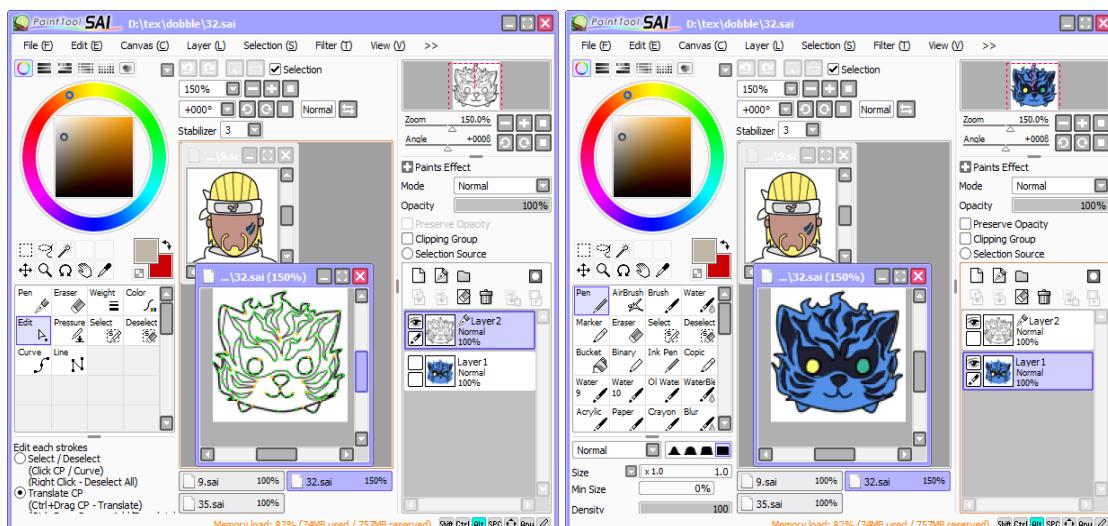
Paint Tool SAI to produkt japoński i stosowany jest głównie do rysowania. Pobrać plik instalacyjny wersji trial można z umieszczonej w odnośniku [11] strony. Instalacja przebiega standardowo. Jest to wersja próbna na 31 dni. Licencja na dzień dzisiejszy (maj 2017 r.) kosztuje 5400JPY (czyli ok. 180 PLN).

Wskazana jest praca z wykorzystaniem tabletu graficznego i piórka (ang. stylus). Spowodowanie jest to właściwością nacisku piórka, która wpływa się na grubość linii. Rysując myszką pojedynczą linię otrzymuje się linię o jednolitej grubości na całej długości. Natomiast używając tabletu linia może być grubsza na początku, a cieńsza na końcu.

Oprócz klasycznego używania pędzla umożliwia narzędzie służące do wektorowego określania ścieżek, co widnieje na porównaniu rys. 3.5. Z lewej strony widoczne jest mniej ikon narzędzi – to one służą do umieszczania punktów, które są podstawą tworzenia grafiki wektorowej. Są to: Pen, Easer, Weight, Color, Edit, który umożliwia m.in. translację punktu bądź całej kreski,

deformacją punktu kontrolnego (ang. anchor – punkt o określonych współrzędnych będący fragmentem krzywej Baziera – ścieżki wektorowej). Interfejs składa się z części:

- **Główne menu** – oprócz standardowych opcji udostępnia pod zakładkami:
 - **Canvas** - zmianę wielkości obszaru roboczego albo rozdzielczości obrazu,
 - **Filter** - ustawianie parametrów barwy/ nasycenia, jasności/ kontrastu.
- **Lewy panel** – metoda wybierania koloru (między innymi: koło barw, suwaki RGB/HSV, mikser kolorów, przyciski z zapisanymi wcześniej ulubionym barwami), poniżej opcje zaznaczania i nawigacji, pod nimi ikony narzędzi rysowania, a ostatnią część zajmują aktualnie wybranego narzędzia. Dla pędzla są to: kształt krawędzi, rozmiar, minimalny rozmiar, kształt, tekstura, rozmycie, rozcieńczenie, trwałość.
- **Prawy panel** – stanowi okno nawigacyjne oraz warstwy wraz z ich trybami krycia.
- **Obszar roboczy** – w nim tworzy się grafikę.

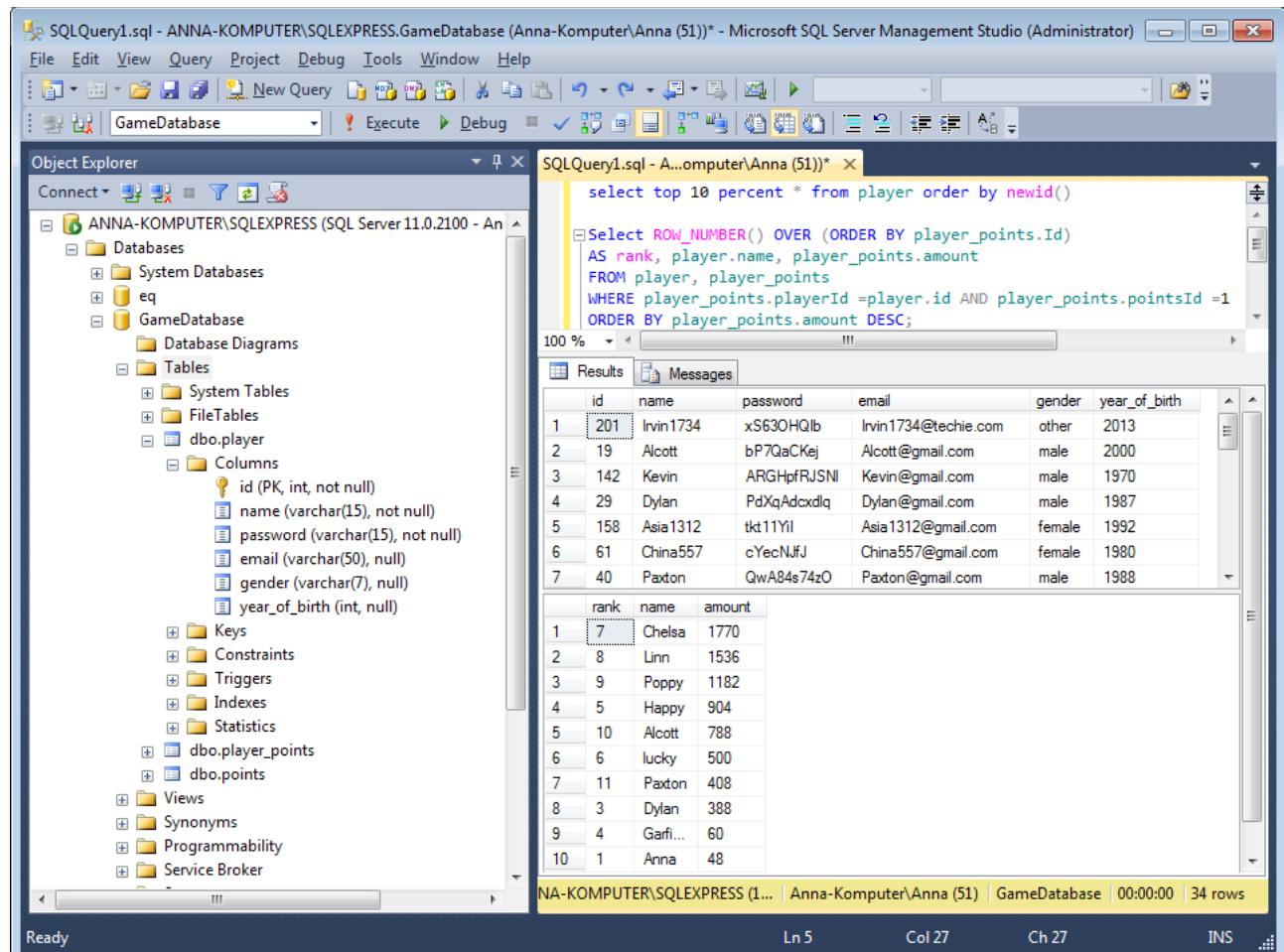


Rys. 3.5. Wektorowy lineart (lewa strona) i wypełnienie kolorem (prawa strona)

3.3 Microsoft SQL Server 2012

Microsoft SQL Server (MS SQL) to system zarządzania bazą danych wpierwany przez korporację Microsoft. Obok komercyjnych wersji Microsoft udostępnia darmowe edycje (Express), które można pobrać ze strony zamieszczonej w odnośniku [13] i użyć nawet w komercyjnych celach. SQL Server Management Studio to zintegrowane środowisko służące do zarządzania komponentami takimi jak baza danych, usługi analityczne, raportowe wchodzące w skład Microsoft SQL Server. Zawiera narzędzia konfigurujące i monitorujące, administrujące instancje SQL Server. Pozwala na budowę zapytań (ang. query) i skryptów.

Interfejs widoczny na rys 3.6 jest intuicyjny. Na samej górze znajduje się pasek głównego menu ze standardowymi opcjami. Zakładka Query pozwala na projektowanie zapytań i zarządzanie ich opcjami. Z lewej strony widoczny jest Object Explorer, który pozwala przeglądać, wybierać i wykonywać operacje na elementach serwera, takie jak dodanie nowej bazy danych, dodanie nowej tabeli, dodanie nowej kolumny. Z prawej strony znajduje się obszar roboczy. Jego górną część stanowi edytor, gdzie można pisać zapytania. Wyniki i wiadomości są wyświetlane poniżej.



Rys. 3.6. Interfejs SQL Server Management Studio

4. ZAŁOŻENIA, STRUKTURA I REALIZACJA GRY

4.1 Założenia projektowe

Opracowana gra sieciowa Dabble została zainspirowana najpopularniejszą grą planszową Dobble. Wyniki rankingu najpopularniejszych gier ułożonych na podstawie raportu Ceneo.pl za I-III kw. 2014 roku zamieszczone są w tabeli 4.1. W oryginalnej rozgrywce gracze otrzymują okrągłe karty, na których widnieją piktogramy. Wygląd gry, jak i kart zaprezentowano na rys 4.1 [12]. Są pełne kolorowych obrazków roślin, postaci, rzeczy, a każda z nich, co jest kluczowe w logice gry, jest unikalna i łączy się z innym kartami dokładnie jednym symbolem. Główna karta leży na stole i wszyscy ją widzą. Każdy z graczy widzi oprócz tej głównej karty pierwszą kartę z własnego stosu. Zadaniem gracza jest odszukanie wspólnego piktogramu i zastąpienie głównej karty własną. Wygrywa ten, kto jako pierwszy pozbędzie się wszystkich kart.

Zaletami tej gry są szybka, dynamiczna runda, interakcja z innymi graczami, proste zasady, które umożliwiają grę nie tylko nastolatkom, czy dorosłym, a także dzieciom, czy osobom starszym.

Tabl. 4.1 Ranking najpopularniejszych gier planszowych

Lp.	Tytuł	Średnia cena [zł]
1	Dobble	50,84
2	Talisman: Magia i Miecz	162,42
3	Dixit	100,73
4	Osadnicy z Catanu	103,67
5	Monopoly Polska	103,22



Rys. 4.1. Gra planszowa Dobble

Zaprojektowana na potrzebę pracy magisterskiej gra Dabble jest podobna do gry Dobble pod względem zasad. Do serwera zgłaszą się klienci – gracze. Serwer generuje odpowiednie talie kart i przydziela je graczom. Gracz widzi wyświetlaną aktualną główną kartę oraz pierwszą kartę ze swojego stosu. Zadaniem gracza jest wskazanie poprzez kliknięcie takiego samego symbolu na obydwóch kartach. Jeśli wskazanie było poprawne, to wskazana karta zastępuje główną, a gracz odsłania kolejną ze swojego stosu. Jeśli wskazanie nie było poprawne, to użytkownik otrzymuje karę czasową, zapobiegającą klikaniu bez przemyślenia.

Solucja posiada dwa projekty, a są to aplikacje serwera oraz klienta. Serwer zawiera całą logikę gry. Klient jedynie wysyła żądania poprzez interfejs aplikacji klienckiej, a serwer mu odpowiada na żądania, przez co widok okna gracza jest odpowiednio aktualizowany.

Do założeń projektowych wlicza się możliwości:

- rejestracji i logowania do konta przez klienta,
- tworzenia nowego pokoju do gry o określonych opcjach, a są to:
 - liczba graczy od 1 do 8,
 - w przypadku gry solo nie są dodawane punkty za wygraną,
 - zestaw pictogramów np. anime, kształty, zwierzęta, jedzenie,
 - tryb gry – liczba obrazków przypadająca na jedną kartę (6 albo 8).
- gry obejmującej:
 - podgląd statusu graczy przebywających w pokoju (nick, liczba kart w talii) z wyszczególnieniem własnych statystyk,
 - czat,
 - informację, kto ostatni położył kartę,
 - widok na główną kartę oraz swoją wraz z możliwością wyboru dowolnego pictogramu,
 - system banowania (kary czasowej) za złe decyzje,
- zbierania punktów i podglądu rankingów.

Obydwa projekty zostały napisane w języku C# w środowisku .NET z wykorzystaniem Windows Forms. Dane użytkowników oraz stan ich punktów są zapisane w bazie danych MS SQL. Grafika została zaprojektowana w programach Adobe Photoshop oraz Paint Tool SAI.

4.2 Architektura aplikacji

Na architekturę aplikacji składa się wiele czynników. Zostaną omówione najważniejsze czyli architektura klient – serwer, użyty wzorzec MVC oraz protokół TCP.

4.2.1 Architektura klient – serwer [2, 3]

Wyróżnia się dwa rodzaje architektury, albo sieci opartej na serwerach, albo opartej na sieci równorzędnej. Podział architektury typu klient-serwer może obejmować tzw. „grubego” i „cienkiego” klienta.

Architekturę równorzędną (ang. peer-to-peer) charakteryzuje posiadania możliwości przez każdego użytkownika do jednoczesnego korzystania z zasobów innych komputerów oraz udostępniania własnych zasobów. Urządzenia w sieci mają taki sam status, zatem nie ma żadnego koordynatora, ani przyporządkowania. Użytkownik sam zarządza własnym urządzeniem i udostępnia swoje zasoby innym użytkownikom. Takie rozwiązanie jest stosowane z reguły w małych sieciach, czyli liczących do dziesięciu komputerów. Wszystkie informacje odnośnie udostępnionych zasobów i użytkownikach uprawnionych do ich pobrania zapisane są na udostępniającym dany zasób urządzeniu. Jeśli korzysta się z kilku serwerów, na których zapisana jest lokalnie informacja dotycząca zasobów, to na każdym z nich z osobna należy uzyskać dostęp poprzez wpisywanie hasła na każdym serwerze. Sieć peer-to-peer jest tania w budowie, ale trudna w utrzymaniu i zarządzaniu.

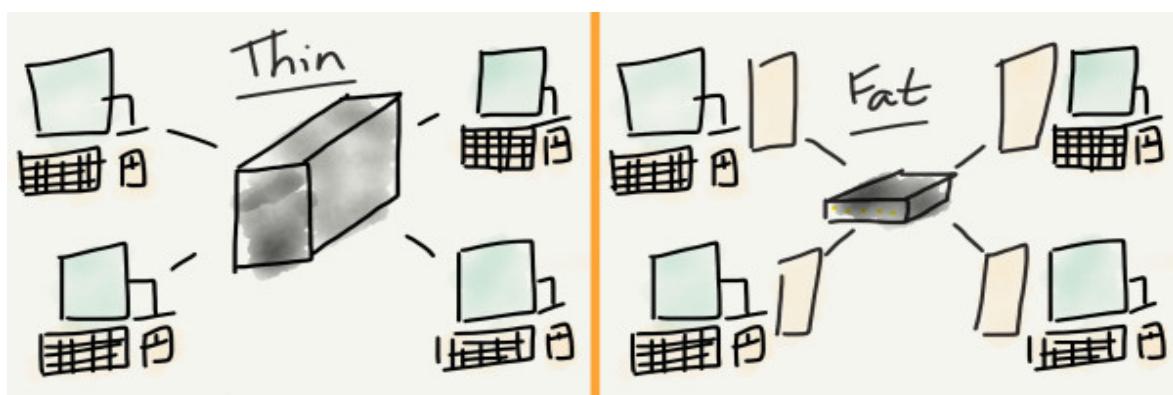
Natomiast architektura klient-serwer (ang. client-server) posiada przynajmniej jeden komputer pełniący rolę serwera. Na nim zainstalowany jest sieciowy system operacyjny, umożliwiający realizację zadań serwera. Serwer udostępnia oraz przechowuje zasoby, takie jak pliki, zarządzanie współdzieleniem drukarek, przechowuje wspólnie wykorzystywaną bazę danych o zasobach sieci, jej użytkownikach oraz uprawnieniach użytkowników. Klient poprzez aplikację kliencką komunikuje się z serwerem.

„Gruby klient” (ang. rich client, fat client) to typ architektury klient-serwer. Według jego założeń po stronie klienta znajduje się warstwa prezentacji oraz przetwarzania danych. Po stronie serwera znajduje się jedynie baza danych i serwer plików. Koszt przetwarzania w takim przypadku jest bardzo wysoki, ponieważ obejmuje kosztowne komputery typu fat client – komputer spełniający rolę klienta, będący wyposażony w klasyczny system operacyjny i wiele aplikacji oraz koszty administracyjne i konserwacji oprogramowania i sprzętu.

„Cienki klient” (ang. thin client) jest związany z technologią WWW. Przeglądarki internetowe mogą działać nawet na słabo wyposażonych komputerach oraz spada wówczas koszt

administracji i utrzymania aplikacji. Wszystkie prace administracyjne i konserwujące sprowadzają się do przestrzeni serwera aplikacji, nie zaś klienta. Eliminuje to całkowicie konieczność wprowadzania zmian w każdym komputerze klienta. Nawet słabe komputery klasy PC mogą obsłużyć rozbudowane aplikacji dzięki technologii internetowej. Istnieje również możliwość uruchamiania aplikacji na różnych platformach oraz dostępu do danych z różnych miejsc sieci.

Porównując „grubego” klienta z „cienkim” ten pierwszy wymaga okresowego połączenia z serwerem, ale często charakteryzuje się zdolnością do wykonywania wielu funkcji bez połączenia z serwerem. Przeciwnie do tego, cienki klient sam przetwarza bardzo mało danych, tylko tyle, ile jest niezbędne. „Cienki” klient polega na pozyskiwaniu dostępu do serwera za każdym razem, gdy dane wejściowe muszą być przetworzone lub poddane walidacji. Różnicę dobrze w sposób uproszczony obrazuje grafika na rys. 4.2, gdzie widać znakomite znaczenie jednostek centralnych komputerów klienckich przy „cienkim” kliencie oraz ich dużą wagę przy połączeniu w przypadku grubego klienta. [13].



Rys 4.2. Symbolicznie wyjaśniona różnica pomiędzy "cienkim", a "grubym" klientem

Podsumowując, gra Dabble jest grą o architekturze klient-serwer, ponieważ wyróżnia się jeden wyspecjalizowany serwer i klientów, którzy żądają od niego zasobów. Typ tej architektury to klient „cienki” ze względu na przetwarzanie danych, które odbywa się po stronie serwera. Aplikacja kliencka zawiera warstwę prezentacji niezbędną do komunikowania się z serwerem oraz wyświetlania stanu gry, ale nie przetwarza danych, wysyła jedynie zapytania i otrzymuje odpowiedzi od serwera.

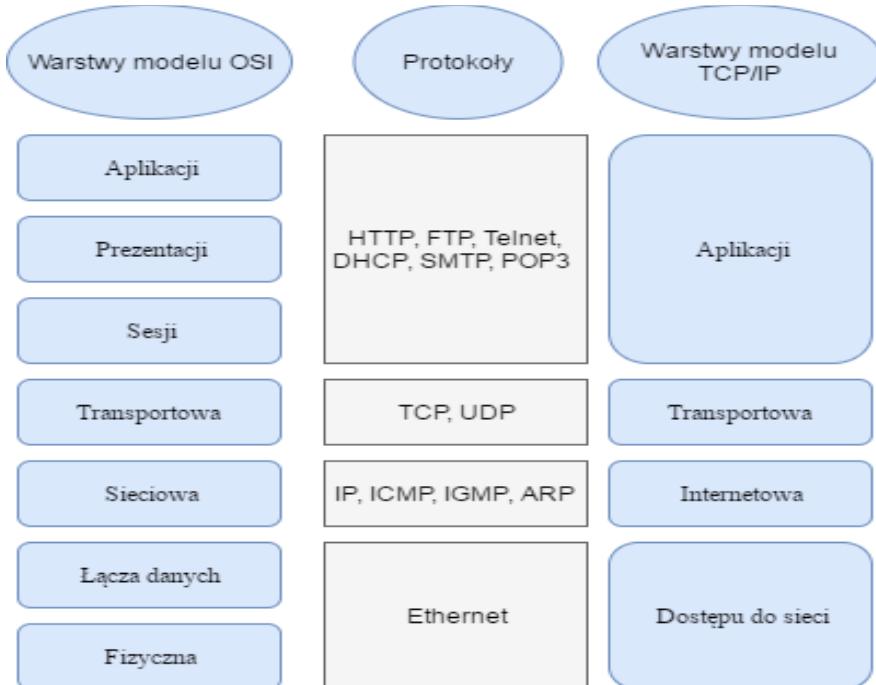
4.2.2 Transmission Control Protocol (TCP) [2]

Urządzenia w sieci komputerowej komunikują się ze sobą, a żeby wymiana danych mogła być możliwa musiał zostać określony sposób umożliwiający wymianę komunikatów. Wszystkie urządzenia komunikujące się z innymi przestrzegają ustalonych reguł. Protokół komunikacyjny to właśnie zbiór norm i zasad.

Wyróżnia się tryb połączeniowy (ang. connection oriented) oraz tryb bezpołączeniowy (ang. connectionless oriented). Tryb połączeniowy, jak sama nazwa wskazuje, polega na ustaleniu logicznego połączenia między komunikującymi się urządzeniami. W celu rozpoczęcia komunikacji należy nawiązać połączenie. Ten tryb jest przydatny, gdy wymiana komunikatów odbywa się w obu kierunkach. Natomiast tryb bezpołączeniowy charakteryzuje się niezależnym przekazywaniem komunikatów np. wysłanie wiadomości e-mail.

Model TCP/IP jest powszechnie stosowany, m.in. w sieci Internet. Model TCP/IP składa się z czterech warstw, chociaż niektóre źródła podają pięć. Wynika to z tego, że warstwa dostępu do sieci jest podzielona na interfejs sieciowy oraz warstwę fizyczną, przez co uznaje się pięć warstw. Warstwa aplikacji spełnia funkcje trzech najwyższych warstw modelu OSI (czyli aplikacji, prezentacji i sesji). W warstwie tej działa np. http, telnet, ssh, smtp, pop3 itp. Głównym zadaniem warstwy transportowej jest zapewnienie komunikacji między programami użytkownika, a także może także zarządzać przepływem informacji oraz dostarczać niezawodności poprzez porządkowanie segmentów danych i retransmisję uszkodzonych lub zagubionych segmentów. Porty umożliwiają działanie wielu aplikacji wymieniających dane w sieci przez jeden komputer. Dzięki temu rozwiązaniu porty przypisane są do określonego połączenia i nie następuje wymieszanie się przesyłanych przez nie danych. Strumień danych jest dzielony na segmenty, a w nagłówku umieszczany jest numer portu identyfikujący aplikację wysyłającą lub odbierającą dane. W warstwie transportowej działa protokół połączeniowy TCP oraz bezpołączeniowy UDP. Następną warstwą jest warstwa internetowa przyjmująca segmenty z warstwy transportowej razem z informacjami identyfikującymi odbiorcę. Ma ona za zadanie wysyłanie pakietów i dostarczanie ich do miejsca przeznaczenia, niezależnie od trasy. Protokół IP zarządza tą warstwą, a podczas podziału danych na pakiety i dodawaniu nagłówka jest umieszczany adres IP odbiorcy i nadawcy. Decyzja, czy wysłać pakiet wprost do sieci lokalnej, czy skierować go do routera jest podejmowana ze względu na adres IP odbiorcy. Kolejny etap to tzw. trasowanie lub routing, polegający na wyznaczaniu najlepszych tras pakietu od nadawcy do odbiorcy. Warstwa ta odpowiada również za obsługę przychodzących pakietów, sprawdza ich poprawność i stwierdza, czy przesłać je dalej, czy przetwarzać na miejscu, a także wysyła komunikaty kontrolne i komunikaty dotyczące błędów oraz obsługuje komunikaty przychodzące. Ostatnią warstwą jest warstwa dostępu do sieci odbierająca pakiety i przesyłającą je przez daną sieć. Zapewnia interfejs z siecią fizyczną i zajmuje się przekazywaniem danych przez fizyczne połączenia (najczęściej modemy i karty sieciowe) między urządzeniami. Formatuje dane do transmisji przez nośnik oraz adresuje dane do podsieci, opierając

się na adresach fizycznych. Zapewnia sprawdzanie błędów przesyłu danych za pomocą sumy kontrolnej ramki.



Rys. 4.3. Porównanie modelu OSI i TCP/IP

Protokół TCP (ang. Transmission Control Protocol) działa w trybie połączeniowym w warstwie transportowej, co widać na rys. 4.3. Ustawienie połączenia zapewnia dostarczenie danych do odbiorcy. Połączenia TCP rozpoznawane są po adresach i portach urządzeń nadawczych i odbiorczych. Ich cechami są możliwość sterowania przepływem, potwierdzeniem odbioru, kontrola błędów i retransmisja danych, zachowanie kolejności danych.

Na komputerze posiadającym jeden adres IP może być uruchomionych jednocześnie kilka aplikacji. Aby móc je identyfikować korzysta się z portów, które reprezentowane są przez liczby naturalne z zakresu 0-65535. Przykładowo dla usług takich jak WWW jest zarezerwowany port 80, telnet korzysta z 23. Gniazdo (ang. socket) to kombinacja adresu IP i numeru portu. Służy do komunikacji się pomiędzy aplikacjami, ponieważ jednoznacznie określa proces w sieci lub zakończenie logicznego łącza komunikacyjnego pomiędzy aplikacjami. Gdy aplikacje działają na dwóch różnych urządzeniach, to para odpowiadających im gniazd definiuje owe połączenie. Serwer otwiera gniazdo i oczekuje na połączenie. Aby klient mógł połączyć się z otwartym gniazdem musi znać adres sieciowy komputera oraz numer portu. Kolejnymi numerami sekwencyjnymi są oznaczane przesyłane segmenty danych. Nim nastąpi rozpoczęcie transmisji danych nadawca i odbiorca wymieniają się swoimi numerami sekwencyjnymi. Odbiorca sprawdzając numer sekwencyjny wie, czy wszystkie segmenty dotarły już do miejsca przeznaczenia. Wysłanie przez

odbiorcę numeru następnego segmentu, który powinien być przesłany stanowi potwierdzenie odebrania segmentu. W polu okno jest określona liczba danych po odebraniu których następuje wysłanie potwierdzenia. Jest to przydatne w przypadku zakłóceń, dużej liczby błędów i potrzeby częstszej retransmisji, ponieważ wielkość okna może być zmniejszona, a częstotliwość wysyłania potwierdzeń się zwiększy. W przypadku mniejszej liczby błędów rozmiar okna jest większy, co wpływa na wzrost przepustowości sieci.

W zaprojektowanej grze Dabble komunikacja odbywa się za pomocą protokołu TCP. Wykorzystano przestrzeń nazw System.Net, SystSystem.IO które dostarcza klas o konstruktorach odpowiedzialnych za:

- TcpClient – inicjalizuje nową instancję klasy TcpClient, która zapewnia połączenie TCP po stronie klienta,
- TcpListener – inicjalizuje nową instancję klasy TcpListener, która nasłuchuje połączeń nadchodzących na określonych w parametrach adresie IP i numerze portu,
- StreamWriter – inicjalizuje nową instancję klasy StreamWriter dla określonego strumienia przy użyciu kodowania UTF-8 i domyślnego rozmiaru bufora,
- IPAdress - inicjalizuje nową instancję klasy IPAdress z adresem określonym jako tablica bajtów.

4.2.3 Wzorzec Model View Controller (MVC) [1]

Wzorzec MVC to wzorzec architektoniczny, który służy do organizowania struktury aplikacji posiadających GUI (ang. graphical user interface – graficzny interfejs użytkownika) poprzez oddzielenie modelu, logiki od prezentacji. Odseparowanie logiki od widoku przynosi wiele korzyści zarówno krótko, jak i długoterminowych, a są to m.in.:

- rozbudowa (ang. development) - poszczególne komponenty nie są bezpośrednio od siebie zależne, co oznacza, że mogą być łatwiej rozwijane samodzielnie. Komponenty mogą być również łatwo zastępowane albo podstawiane, zapobiegając powikłaniom przez komponent, który by wpływał na inne komponenty z którymi współdziała,
- testowalność (ang. testability) – luźne powiązanie komponentów pozwala na implementacji testów zastępować „przetwarzane” komponenty. Ułatwia to pomijanie nawiązywania połączenia z bazą danych poprzez zastąpienie komponentu, który tworzy połączenie z bazą takim, który po prostu zwraca statyczne dane. Możliwość korzystania z udawanych danych

na wejściu przyjmowanych jako parametry znacznie ułatwia proces testowania i może drastycznie zwiększyć niezawodność systemu w czasie.

- konserwacja (ang. maintenace) – odizolowana logika komponentów oznacza, że zmiany są zwykle dotyczą małej liczby komponentów – często tylko jednego. Ponieważ ryzyko zmiany jest zwykle skorelowane ze zmianą zakresu, modyfikowanie mniejszej liczby komponentów jest zalecane

Wzorzec MVC dzieli aplikację na trzy warstwy: modelu, widoku, prezentacji, co jest zaprezentowane na rys. 4.4. Każda z warstw ma określone zadanie i nie jest informowana, w jaki sposób inne warstwy wykonują swoje zadania.



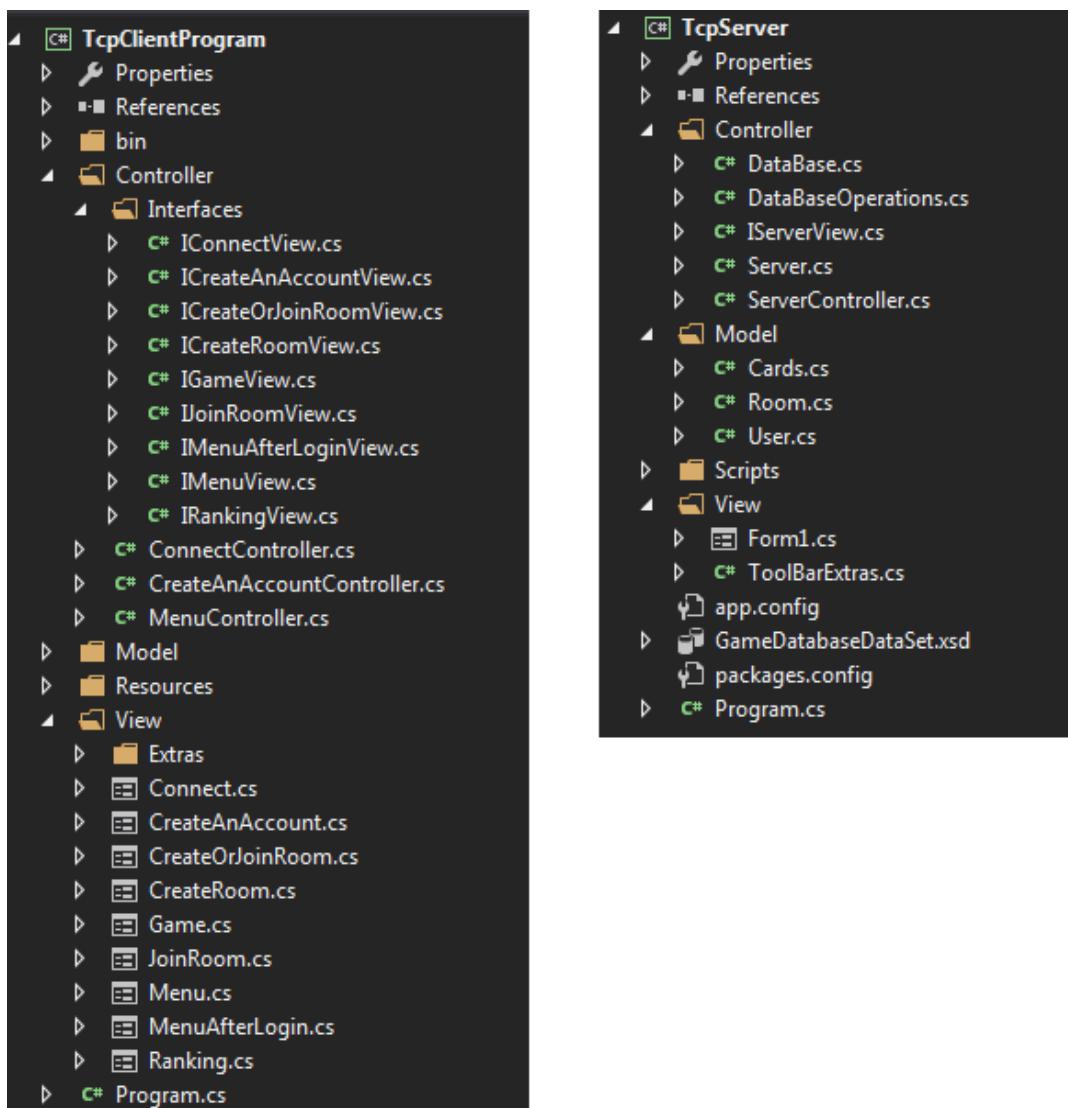
Rys. 4.4. Koncepcja wzorca MVC

Model reprezentuje głównie dane, które użytkownicy przeglądają lub modyfikują oraz jego zachowania, czyli metody. W grze Dabble jako przykład można podać klasę Cards reprezentującą karty, która zawiera zmienne takie jak `numberOfCards`, `numberOfSymbols`, `setOfCards` oraz metody `createCards`, `generateCardsForPlayer`, które generują odpowiednie talie kart. Oprócz modelu kart są jeszcze modele pokoju (`Room`) oraz użytkownika (`User`).

Widok jest odpowiedzialny za sposób wyświetlania obiektów modelu w interfejsie użytkownika. W aplikacjach generowany jest zwykle kod HTMLa, który jest renderowany przez przeglądarkę internetową. W zaprojektowanej grze użyto jednak Windows Forms. Raz zaprojektowany model może być zwizualizowany korzystając z różnych reprezentacji np. HTML, PDF, XML i innych. Zgodnie z zamysłem oddzielenia logiki biznesowej od widoku, widok zajmuje się jedynie wyświetlaniem danych i nie powinien zawierać żadnej logiki biznesowej, ponieważ kontroler i model dostarcza widokowi wszystkiego, czego on potrzebuje.

Kontroler obsługuje przychodzące żądania, wykonuje operacje na modelu oraz zwraca wynik do wyświetlania. Jak sama nazwa wskazuje kontroluje logikę aplikacji i działa jako pośrednik pomiędzy widokiem, a modelem. Manipuluje danymi z modelu, ale nie powinien ich nadpisywać czy zmieniać. Czasem przekazuje sterowanie do innego kontrolera.

Kwestią sporną często jest miejsce, gdzie należy wykonywać obliczenia, czy w modelu czy kontrolerze. Po przeczytaniu książek, publikacji, zdobywaniu doświadczenia w pisaniu kodu opartego o wzorzec MVC można stwierdzić, że umieszczenie kodu dotyczącego obliczeń, to kwestia indywidualna i nie ma sztywnej reguły. Gdy obliczenia są relatywnie małe, to można je wykonać w kontrolerze. Natomiast, gdy te same obliczenia są lub mogą być w przyszłości potrzebne, to należy je umieścić w modelu, ponieważ zwraca on gotowe do przekazania do widoku dane. Kontroler w takim wypadku jedynie pośredniczy pomiędzy modelem, a widokiem, co stanowi rekomendowane działanie i tzw. dobrą praktykę. Ze względu na to rekomendowane działanie w zaprojektowanej grze Dabble generowaniem talii kart na potrzeby pojedynczej rozgrywki zajmuje się model. Kontroler odpowiada za pobieranie odpowiednich danych dotyczących talii gracza oraz głównej karty, które są wyświetlane dzięki widokowi w aplikacji Windows Forms. Solucja zawiera dwa projekty widoczny jest na rys. 4.5



Rys. 4.5. Widok solucji opartej na wzorcu MVC

Implementacja wzorca omówiona zostanie na przykładzie kliknięcia przez użytkownika w guzik o nazwie button2 odpowiadający za tworzenie konta. Wywoływana jest wówczas metoda klasy danej WindowsForm, w tym przypadku jest to public partial class Menu : MetroForm, której struktura wygląda następująco: private void button1_Click_2 (object sender, EventArgs e) {} . Chcąc wykonać kod odpowiadający za tworzenie konta można go umieścić bezpośrednio w ciele tej funkcji, jednak przecież to separacji logiki prezentacji od widoku. W tym celu został utworzony interfejs oraz kontroler. Zatem klasa implementuje interfejs IMenuView poprzez dodanie jego nazwy po przecinku public partial class Menu : MetroForm, IMenuView. Interfejs ten zawiera jedynie sygnatury metod bez ich ciał. Każdy z tworzonych interfejsów musi posiadać metodę odpowiedzialną za przypisanie kontrolera: public interface IMenuView {void SetController(MenuController controller); }

Każdy kontroler jest pisany w oparciu o szablon wyglądający następująco:

```
public class MenuController{

    IMenuView _view;

    public MenuController(IMenuView view) {
        _view = view;
        view.SetController(this);
    }

    ...

    public void CreateAnAccount() {
        CreateAnAccount caa = new CreateAnAccount();
        caa.Visible = false;

        CreateAnAccountController controller =
            new CreateAnAccountController(caa);

        controller.LoadViewCreateAnAccount();

        caa.ShowDialog();

        _view.Hide();
    }
}
```

Kontroler stanowi osobną klasę i w nim znajduje się implementacja wymaganych funkcjonalności.

Do klasy Menu zostaje dopisany następujący kod: MenuController _controller;

```
public void SetController(MenuController controller) {
    _controller = controller;
}

private void button2_Click_1(object sender, EventArgs e) {
    this.Hide();
    this._controller.CreateAnAccount();
}
```

Funkcja SetController w klasie widoku Menu pozwala na powiadomienie widoku, do której instancji kontrolera musi ona przekazywać zdarzenia oraz wszystkie procedury obsługi zdarzenia wywołującą odpowiednią metodę danego zdarzenia w kontrolerze. Omówiony został prosty przykład reagowania na żądanie wyboru poprzez kliknięcie w przycisk bez przekazywania dodatkowych danych. Bardziej złożony jest np. interfejs odpowiadający za tworzenie nowego konta, a wygląda następująco:

```
public interface ICreateAnAccountView
{
    void SetController(CreateAnAccountController controller);
    void Hide();
    void Close();
    void DisposeForm();
    void AddItemGenderComboBox(string s);
    void AddItemYearComboBox(int i);
    void CreateImage();
    void ShowInformation(string s1, string s2);
    string ServeripTextbox { get; set; }
    string UserNameTextbox { get; set; }
    string PortTextbox { get; set; }
    string PasswordTextbox { get; set; }
}
```

Podczas tworzenia widoku formularza rejestracji wypełnianie elementów kontrolki comboBox odbywa się nie w widoku, a w kontrolerze. Są to dane dotyczące możliwości wyborów z listy i umieszczone w metodach, takie jak `AddItemGenderComboBox`, czyli metoda odpowiadająca za dodanie możliwych do wybrania płci, czy `AddItemYearComboBox` – metoda odpowiadająca za dodanie możliwych do wyboru lat urodzeń. Ładowanie widoku odbywa się przy użyciu metody:

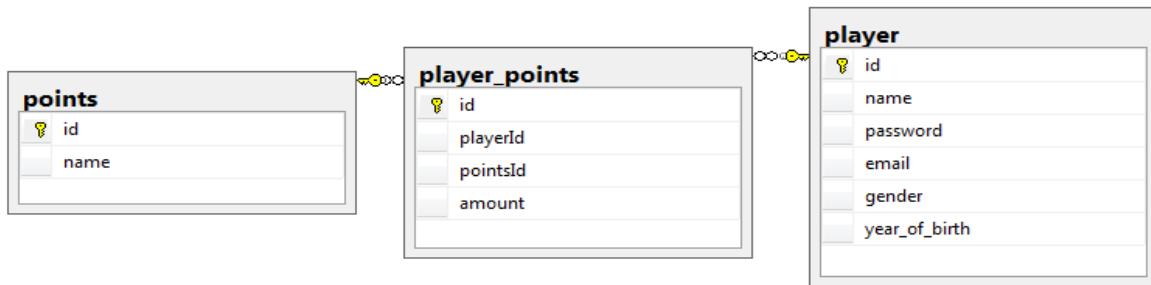
```
public void LoadViewCreateAnAccount() {  
    LoadSettings();  
    int currentYear = 2017;  
    for (int i = currentYear; i >= currentYear - 120; i--)  
    {  
        _view.AddItemYearComboBox(i);  
    }  
    _view.AddItemGenderComboBox("female");  
    _view.AddItemGenderComboBox("male");  
    _view.AddItemGenderComboBox("other");  
    _view.CreateImage();  
}
```

Poprzez wywołaniu w kontrolerze np. `_view.AddItemYearComboBox` następuje wywołanie w klasie widoku metody `AddItemYearComboBox`, która dodaje liczbę przekazaną jako parametr do `comboBoxa`:

```
public void AddItemYearComboBox(int i) {  
    yearComboBox.Items.Add(i);  
}
```

4.3 Baza danych

Została wykorzystana baza danych Microsoft SQL Server 21012 (MS SQL). Biorąc pod uwagę wymagania projektowe w bazie zaprezentowanej na rys. 4.6 znajdują się jedynie dane użytkownika, które podał podczas rejestracji konta oraz zdobyte punkty. Alternatywnym rozwiązaniem byłoby umieszczenie pola dotyczącego zgromadzonych punktów w tabeli gracza, ale takie rozwiązanie nie zapewnia elastyczności i skalowalności.



Rys. 4.6. Baza danych

Tabela player zawiera id jako klucz główny (ang. primary key, PK). Oprócz tego zawiera atrybuty dotyczące nicku, hasła, adresu e-mail, płci, roku urodzenia. Losowe 10 rekordów zostało pokazanych na rys. 4.7. Dane zostały wygenerowane poprzez napisanie własnej metody, która zwraca odpowiednie dane, a część jej kodu wygląda następująco:

```

Random rnd = new Random();

string[] girlName = { "Addison", "Amanda", "Andrea", "Asia", "Callie",
"Camelia", "Chelsa", "China", "dabria", "ester", "Eraaa", "Hunter",
"Linn", "Pink", "Poppy", "Rosa" };

string[] nameBoy = { "Andey", "Alcott", "alger", "almond", "ANSON",
"Arden", "Ashby", "Baker", "ballard", "Cade", "Colden", "Dylan", "Dee",
"Dudu", "Garfield", "Happy", "lucky", "Star", "Irvin", "Kevin", "Linton",
"Newt", "Paxton", "ProPlayer", "Skipper", "Teddy", "True", "Tayler" };

string[] emailAdress = { "@onet.pl", "@gmail.com", "@outlook.com",
"@hotmail.com", "@hushmail.me", "@techie.com", "@gmx.com",
"@facebook.com" };

int newNumber = rnd.Next(1, 2017);

int number;

for (int i = 0; i < girlName.Length; i++) {

    number = newNumber;

    nick = girlName[i];

    password = GetRandomAlphanumericString(rnd.Next(8, 12));

    email = girlName[i] + emailAdress[1];

    age = rnd.Next(1970, 2011).ToString();

    gender = "female";

    string[] sqlInsert = { "a", nick, password, email, age, gender };
}

```

```

createNewAccount(sqlInsert);

newNumber = rnd.Next(1, 2017);

}

```

Aby zapisać dane do bazy została napisana metoda `createNewAccount` w klasie `DatabaseOperations`, która przyjmuje jako parametr łańcuch znaków. Jest uniwersalna, ponieważ zarówno wygenerowane dane, jak i dane przekazane za pomocą formularza umieszcza w bazie danych.

	id	name	password	email	gender	year_of_birth
1	135	Dee	OUZvwnAKmc	Dee@gmail.com	male	1987
2	98	Linton1240	BStRDhPXm	Linton1240@techie.com	other	1979
3	196	Dudu1542	CrcJPaClh	Dudu1542@techie.com	other	1978
4	4	Andrea	4uJyMqKB	Andrea@gmail.com	female	1986
5	202	Kevin78	cxnJaaplhs	Kevin78@techie.com	other	1994
6	116	ester	AqR2NbXJ	ester@gmail.com	female	1991
7	124	Alcott	czEoidLxh0H	Alcott@gmail.com	male	2000
8	115	dabria	mTK6DhX1H	dabria@gmail.com	female	1990
9	37	Kevin	AldcbNBRJp	Kevin@gmail.com	male	1970
10	164	Chelsa1759	LYffblm7g	Chelsa1759@gmail.com	female	2000
11	121	Poppy	qf8qz8FTp	Poppy@gmail.com	female	1997
12	172	Eraaa733	t6R1eSLq6q	Eraaa733@gmail.com	female	2012
13	187	ANSON1525	I3lb7ywF	ANSON1525@techie.com	other	2004

Rys. 4.7. Dziesięć losowych rekordów tabeli player

```

void createNewAccount(string[] response) {

DataTable dt = db.CreateDataTable("select id, name,password,email,
gender, year_of_birth from player;");

try {

    int id = nextId(dt) + 1;

    int age = Convert.ToInt32(response[4]);

    string sql = "insert into GameDatabase.dbo.player
values ('{0}', '{1}', '{2}', '{3}', '{4}', '{5}') ;";

    sql = "insert into GameDatabase.dbo.player values('" + id
+ "', '" + response[1] + "','" + response[2] + "','" + response[3] + "','" +
response[5] + "','" + age + "') ;";

    db.RunQuery(sql);
}

```

```

        }

        catch {

            extra.ShowInformation("Error", "Error with save data");

        }

    }

```

Tabela points reprezentuje dany tryb gry. Aktualnie są dwa do wyboru – ośmioelementowy (trudniejszy) oraz sześciosymbolowy (łatwiejszy). W przyszłości można rozbudować grę Dabble i swobodnie dodawać inne rodzaje gier, nie będące nawet związanymi z opracowaną już talią. Mogą to być szachy, statki, kalambury, czy inne. Dzięki tabeli pośredniczącej player_points i umieszczeniu w niej dwóch kluczy obcych (ang. foreign key, FK) – playerId oraz pointsId liczba zgromadzonych punktów jest przetrzymywana w osobnej tabeli w składowej krotki o nazwie amount. Dane zapisane w obydwóch tabelach są umieszczone na rys. 4.8.

The screenshot shows two tables side-by-side. The left table, titled 'player_points', has columns id, playerId, pointsId, and amount. It contains 9 rows of data. The right table, titled 'points', has columns id and name, containing 2 rows of data.

	id	playerId	pointsId	amount
1	1	1	1	40
2	2	209	2	115
3	3	209	1	23
4	4	29	1	40
5	5	31	2	40
6	6	32	1	60
7	7	33	1	120
8	8	34	1	500
9	9	2	2	12

	id	name
1	1	Osmiosymbolowy
2	2	Szesciosymbolowy

Rys. 4.8. Widok danych z tabeli player_points oraz points

Aktualizacja punktów gracza odbywa się za pomocą metody updatePlayerPoints:

```

void updatePlayerPoints(int playerId, int pointsId, int finallyPoints) {
    DataTable dtPlayerPoints = db.CreateDataTable("select id, playerId,
    pointsId, amount from player_points;");

    try {

        string sql = "update GameDatabase.dbo.player_points set amount
        ='{0}' where GameDatabase.dbo.player_points.playerId = '{1}' and
        GameDatabase.dbo.player_points.pointsId ='{2}';";

        sql = "update GameDatabase.dbo.player_points set amount =" +
        finallyPoints + "where GameDatabase.dbo.player_points.playerId = " +
        playerId + "and GameDatabase.dbo.player_points.pointsId =" + pointsId;
    }
}

```

```
        db.RunQuery(sql);  
  
    catch{ extra.ShowInformation("Error", "Error with save data"); }
```

Projekt Server zawiera dostęp do bazy danych. Aplikacja kliencka nie łączy się z bazą danych, a jedynie wysyła zapytania do serwera o potrzebne dane. Aplikacja serwera pobiera je i wysyła dane klientowi.

Do obsługi bazy danych zostały napisane dwie klasy: Database oraz DatabaseOperations, z których pierwsza korzysta z przestrzeni nazw System.Data.SqlClient posiadających klasy takie jak:

- SqlDataAdapter – reprezentująca zestaw poleceń danych i połączenie z bazą danych, które służą do wypełnienia DataSet i aktualizowania bazy danych SQL,
- SqlConnection – posiada właściwość odpowiadającą za otworzenie połączenia,
- SqlCommand - reprezentująca instrukcję języka Transact-SQL lub procedury składowanej do wykonania na bazie danych SQL.

Klasa Database zapewnia połączenie z bazą danych. Tworzy połączenie z bazą oraz posiada zaimplementowaną metodę adaptera, który wysyła zapytania do bazy. Tworzy obiekty DataTable oraz DataRow, które odpowiadają tabeli oraz wierszowi z bazy danych. Klasa DatabaseOperations skupia się na operacjach wykonywanych na bazie danych. Wykorzystuje metody zimplementowane w klasie Database do połączenia się z bazą, tworzenia konta, a także walidacji. Metoda bool CheckLogin (string nick, string password) sprawdza, czy podczas logowania nick znajduje się w bazie, a jeśli tak, to czy podane jako parametr hasło odpowiada hasłu przypisanemu do konta w bazie danych i zwraca wartość boolowską (logiczną - true, gdy się zgadza, false, gdy są różne).

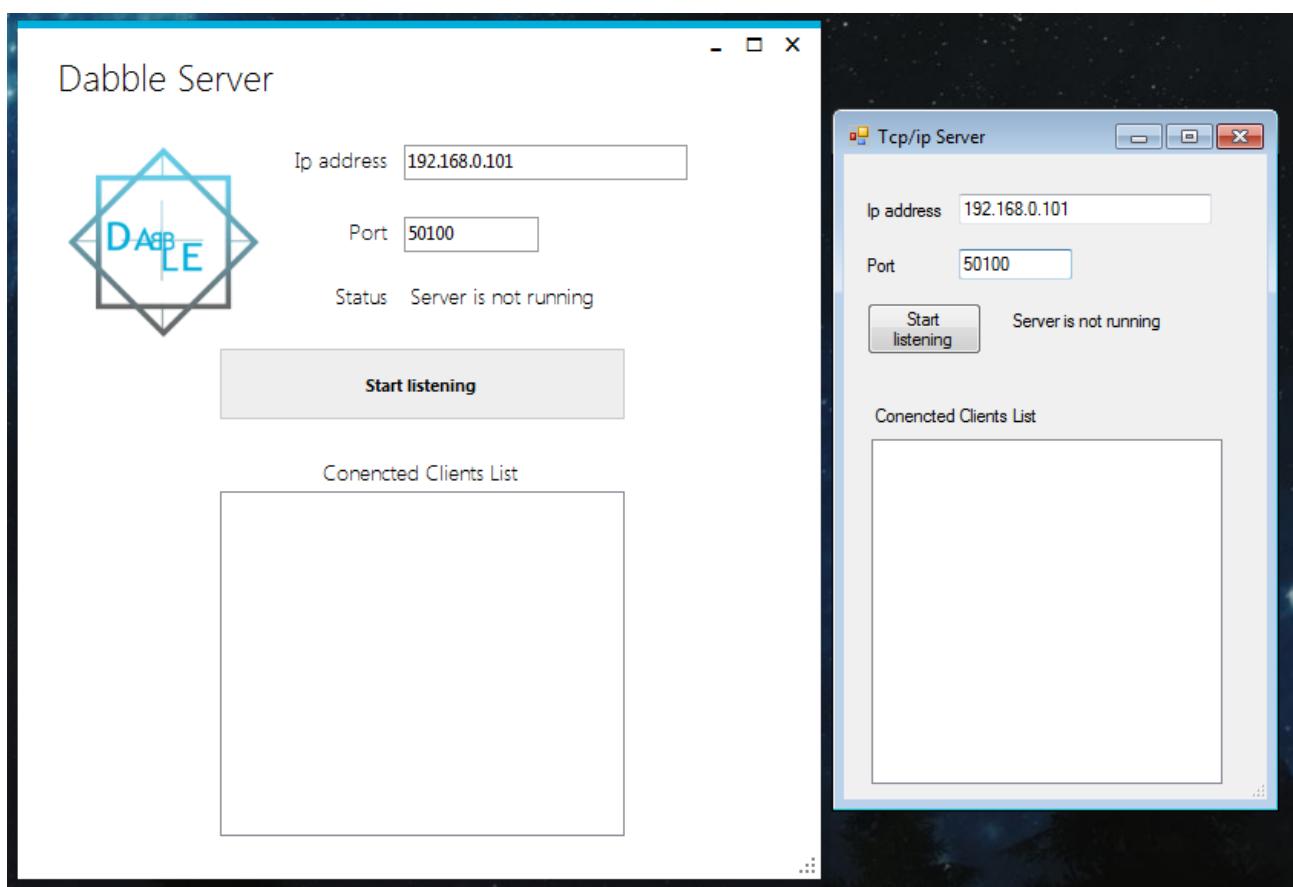
Zawiera metody odpowiadające za sprawdzenie, czy podczas rejestracji użytkownik nie wpisał już zajętego nicku, tworzenia nowej krotki, gdy gracz po raz pierwszy gra w dany tryb i aktualizacji jego punktów, gdy już wcześniej w niego grał. Przypisuje też do zmiennych typu list<string> dane dotyczące poszczególnych rankingów, przy użyciu zapytania: Select ROW_NUMBER()
OVER(ORDER BY player_points.amount DESC) as rank, player.name as name,
player_points.amount as amount from player, player_points where
player_points.playerId = player.id AND player_points.pointsId = 2 ORDER
BY player_points.amount DESC;". W przypadku gry trudniejszej (ośmiosymbolowej)
wartość player_points.pointsId wynosi 1.

4.4 Serwer

Serwer to aplikacja, która ma za zadanie obsługiwać żądania klientów. Za pomocą kliknięcia w przycisk „Start listening” rozpoczyna się nasłuch na wybranym adresie IP oraz porcie, które stanowią gniazdo. Domyślnie pola tekstowe są uzupełniane lokalnym adresem IP komputera oraz numerem portu ustawionym na 50100.

4.4.1 Prezentacja widoku

Do zaprojektowania UI (ang. user interface) wykorzystano MetroFramework, ponieważ wygląd UI wygląda świeżej, nowocześniej, niż w przypadku standardowych form, co widać na porównaniu z rys. 4.9. Z lewej strony znajduje się widok okna zaprojektowanego przy pomocy MetroFramework, a z prawej zwykłych Form. Początkowo aplikacja była projektowana w oparciu o wbudowane komponenty Windows Forms, jednak podmienienie tych elementów nie stanowiło dużej trudności z uwagi na zastosowany wzorzec MVC, który był opisany szerzej w podrozdziale 4.3.2. W późniejszym etapie gra została nazwana Dabble i wykonano logo w programie Adobe Photoshop zgodnie z obowiązującymi w roku 2017 trendami, jakimi są minimalizm, gradient, geometria.



Rys. 4.9. Porównanie widoku MetroForm ze zwykłym wyglądem Form

4.4.2 Algorytm generowania talii i system rozdawania kart

Podstawowym problemem implementacyjnym było zaprojektowanie algorytmu odpowiadającego za generowanie talii kart i późniejsza prezentacja otrzymanych liczb w postaci obrazków na kartach.

Algorytm znajduje się w modelu Cards. Podczas działania algorytmu najpierw wybierana jest główna liczba n . Zbiór możliwych do wygenerowania kart ma wówczas zakres $n^2 + n + 1$, zawierający pary liczb $\{0, \dots, n-1\}$ oraz $n+1$ singletonów z zakresu $\{0, 1, \dots, n-1, \infty\}$ ("punkty w nieskończoności").

Dla każdego $0 \leq a \leq n-1$ i $0 \leq b \leq n-1$ otrzymuje się kartę z zakresu $n+1$ zawierającą pary $\{(x, ax + b \bmod n)\}$ oraz singleton a .

Są generowane również specjalne karty, dla każdego $0 \leq c \leq n-1$, zawierające pary $\{(c, x)\}$ oraz singleton ∞ . Jedna "super" karta zwiera wszystkie $n+1$ singletonów.

Oczywiście, dwie karty z tym samym a przecinają się tylko w singletonie. Dwie karty z różnym a przecinają się w unikalnej solucji $a_1x + b_1 = a_2x + b_2 \pmod{n}$. Dwie specjalne karty przecinają się się tylko w singletonach, natomiast zwykła i specjalna karta przecina się w $(c, ac + b)$. Super karta przecina resztę w singletonach.

Wykorzystując opisaną teorię w praktyce, gdy podstawi się za $n = 7$ otrzymuje się:

- $n^2 + n + 1$, czyli $50 + 7 = 57$ kart,
- $n+1$, czyli $7+1 = 8$ symboli,
- spełniony warunek, że dwie dowolne karty zawierają dokładnie jeden symbol wspólny.

Aktualnie w grze są zaimplementowane rozgrywki wykorzystujące 8 bądź 6 symboli, jednak zmieniając parametr p można uzyskać inne talie. Niektóre z nich zostały zaprezentowane w tabelach 4.2-4.8. Zamiast z $n^2 + n + 1$ korzystano z $n^2 + n$, by mieć parzystą liczbę kart w talii.

Tabl. 4.2. Parametr $p=2$

Parametr $p = 2$			
Id	Nr symboli		
#0	0	1	5
#1	2	3	5
#2	0	2	6
#3	1	3	6
#4	0	3	7
#5	5	6	7

Tabl. 4.3. Parametr p=3

Parametr p =3				
Id	Nr symboli			
#0	0	1	2	10
#1	3	4	5	10
#2	6	7	8	10
#3	0	3	6	11
#4	1	4	7	11
#5	2	5	8	11
#6	0	4	8	12
#7	1	5	6	12
#8	2	3	7	12
#9	0	5	7	13
#10	1	3	8	13
#11	10	11	12	13

Tabl. 4.4. Parametr p=4

Parametr p =4					
Id	Nr symboli				
#0	0	1	2	3	17
#1	4	5	6	7	17
#2	8	9	10	11	17
#3	12	13	14	15	17
#4	0	4	8	12	18
#5	1	5	9	13	18
#6	2	6	10	14	18
#7	3	7	11	15	18
#8	0	5	10	15	19
#9	1	6	11	12	19
#10	2	7	8	13	19
#11	3	4	9	14	19
#12	0	6	8	14	20
#13	1	7	9	15	20
#14	2	4	10	12	20
#15	3	5	11	13	20
#16	0	7	10	13	21
#17	1	4	11	14	21
#18	2	5	8	15	21
#19	17	18	19	20	21

Tabl. 4.5. Parametr p=5

Parametr p =5						
Id	Nr symboli					
#0	0	1	2	3	4	26
#1	5	6	7	8	9	26
#2	10	11	12	13	14	26
#3	15	16	17	18	19	26
#4	20	21	22	23	24	26
#5	0	5	10	15	20	27
#6	1	6	11	16	21	27
#7	2	7	12	17	22	27
#8	3	8	13	18	23	27
#9	4	9	14	19	24	27
#10	0	6	12	18	24	28
#11	1	7	13	19	20	28
#12	2	8	14	15	21	28
#13	3	9	10	16	22	28
#14	4	5	11	17	23	28
#15	0	7	14	16	23	29
#16	1	8	10	17	24	29
#17	2	9	11	18	20	29
#18	3	5	12	19	21	29
#19	4	6	13	15	22	29
#20	0	8	11	19	22	30
#21	1	9	12	15	23	30
#22	2	5	13	16	24	30
#23	3	6	14	17	20	30
#24	4	7	10	18	21	30
#25	0	9	13	17	21	31
#26	1	5	14	18	22	31
#27	2	6	10	19	23	31
#28	3	7	11	15	24	31
#29	26	27	28	29	30	31

Tabl.4.6. Parametr p=6

Parametr p =6							
Id	Nr symboli						
#0	0	1	2	3	4	5	37
#1	6	7	8	9	10	11	37
#2	12	13	14	15	16	17	37
#3	18	19	20	21	22	23	37
#4	24	25	26	27	28	29	37
#5	30	31	32	33	34	35	37
#6	0	6	12	18	24	30	38
#7	1	7	13	19	25	31	38
#8	2	8	14	20	26	32	38
#9	3	9	15	21	27	33	38
#10	4	10	16	22	28	34	38
#11	5	11	17	23	29	35	38
#12	0	7	14	21	28	35	39
#13	1	8	15	22	29	30	39
#14	2	9	16	23	24	31	39
#15	3	10	17	18	25	32	39
#16	4	11	12	19	26	33	39
#17	5	6	13	20	27	34	39
#18	0	8	16	18	26	34	40
#19	1	9	17	19	27	35	40
#20	2	10	12	20	28	30	40
#21	3	11	13	21	29	31	40
#22	4	6	14	22	24	32	40
#23	5	7	15	23	25	33	40
#24	0	9	12	21	24	33	41
#25	1	10	13	22	25	34	41
#26	2	11	14	23	26	35	41
#27	3	6	15	18	27	30	41
#28	4	7	16	19	28	31	41
#29	5	8	17	20	29	32	41
#30	0	10	14	18	28	32	42
#31	1	11	15	19	29	33	42
#32	2	6	16	20	24	34	42
#33	3	7	17	21	25	35	42
#34	4	8	12	22	26	30	42
#35	5	9	13	23	27	31	42
#36	0	11	16	21	26	31	43
#37	1	6	17	22	27	32	43
#38	2	7	12	23	28	33	43
#39	3	8	13	18	29	34	43
#40	4	9	14	19	24	35	43
#41	37	38	39	40	41	42	43

Tabl. 4.7. Parametr p=7

Id	Parametr p =7								
	Nr symboli								
#0	0	1	2	3	4	5	6	50	
#1	7	8	9	10	11	12	13	50	
#2	14	15	16	17	18	19	20	50	
#3	21	22	23	24	25	26	27	50	
#4	28	29	30	31	32	33	34	50	
#5	35	36	37	38	39	40	41	50	
#6	42	43	44	45	46	47	48	50	
#7	0	7	14	21	28	35	42	51	
#8	1	8	15	22	29	36	43	51	
#9	2	9	16	23	30	37	44	51	
#10	3	10	17	24	31	38	45	51	
#11	4	11	18	25	32	39	46	51	
#12	5	12	19	26	33	40	47	51	
#13	6	13	20	27	34	41	48	51	
#14	0	8	16	24	32	40	48	52	
#15	1	9	17	25	33	41	42	52	
#16	2	10	18	26	34	35	43	52	
#17	3	11	19	27	28	36	44	52	
#18	4	12	20	21	29	37	45	52	
#19	5	13	14	22	30	38	46	52	
#20	6	7	15	23	31	39	47	52	
#21	0	9	18	27	29	38	47	53	
#22	1	10	19	21	30	39	48	53	
#23	2	11	20	22	31	40	42	53	
#24	3	12	14	23	32	41	43	53	
#25	4	13	15	24	33	35	44	53	
#26	5	7	16	25	34	36	45	53	
#27	6	8	17	26	28	37	46	53	
#28	0	10	20	23	33	36	46	54	
#29	1	11	14	24	34	37	47	54	
#30	2	12	15	25	28	38	48	54	
#31	3	13	16	26	29	39	42	54	
#32	4	7	17	27	30	40	43	54	
#33	5	8	18	21	31	41	44	54	
#34	6	9	19	22	32	35	45	54	
#35	0	11	15	26	30	41	45	55	
#36	1	12	16	27	31	35	46	55	
#37	2	13	17	21	32	36	47	55	
#38	3	7	18	22	33	37	48	55	
#39	4	8	19	23	34	38	42	55	
#40	5	9	20	24	28	39	43	55	
#41	6	10	14	25	29	40	44	55	
#42	0	12	17	22	34	39	44	56	
#43	1	13	18	23	28	40	45	56	
#44	2	7	19	24	29	41	46	56	
#45	3	8	20	25	30	35	47	56	
#46	4	9	14	26	31	36	48	56	
#47	5	10	15	27	32	37	42	56	
#48	6	11	16	21	33	38	43	56	
#49	0	13	19	25	31	37	43	57	
#50	1	7	20	26	32	38	44	57	
#51	2	8	14	27	33	39	45	57	
#52	3	9	15	21	34	40	46	57	
#53	4	10	16	22	28	41	47	57	
#54	5	11	17	23	29	35	48	57	
#55	50	51	52	53	54	55	56	57	

Tabl. 4.8. Parametr p=8

Id	Parametr p =8								
	Nr symboli								
#0	0	1	2	3	4	5	6	7	65
#1	8	9	10	11	12	13	14	15	65
#2	16	17	18	19	20	21	22	23	65
#3	24	25	26	27	28	29	30	31	65
#4	32	33	34	35	36	37	38	39	65
#5	40	41	42	43	44	45	46	47	65
#6	48	49	50	51	52	53	54	55	65
#7	56	57	58	59	60	61	62	63	65
#8	0	8	16	24	32	40	48	56	66
#9	1	9	17	25	33	41	49	57	66
#10	2	10	18	26	34	42	50	58	66
#11	3	11	19	27	35	43	51	59	66
#12	4	12	20	28	36	44	52	60	66
#13	5	13	21	29	37	45	53	61	66
#14	6	14	22	30	38	46	54	62	66
#15	7	15	23	31	39	47	55	63	66
#16	0	9	18	27	36	45	54	63	67
#17	1	10	19	28	37	46	55	66	
#18	2	11	20	29	38	47	48	57	67
#19	3	12	21	30	39	40	49	58	67
#20	4	13	22	31	32	41	50	59	67
#21	5	14	23	24	33	42	51	60	67
#22	6	15	16	25	34	43	52	61	67
#23	7	8	17	26	35	44	53	62	67
#24	0	10	20	30	32	42	52	62	68
#25	1	11	21	31	33	43	53	63	68
#26	2	12	22	24	34	44	54	66	
#27	3	13	23	25	35	45	55	67	
#28	4	14	16	26	36	46	48	58	68
#29	5	15	17	27	37	47	49	59	68
#30	6	8	18	28	38	40	50	60	68
#31	7	9	19	29	39	41	51	61	68
#32	0	11	22	25	36	47	50	61	69
#33	1	12	23	26	37	40	51	62	69
#34	2	13	16	27	38	41	52	63	69
#35	3	14	17	28	39	42	53	66	
#36	4	15	18	29	32	43	54	57	69
#37	5	8	19	30	33	44	55	58	69
#38	6	9	20	31	34	45	48	59	69
#39	7	10	21	24	35	46	49	60	69
#40	0	12	16	28	32	44	48	60	70
#41	1	13	17	29	33	45	49	61	70
#42	2	14	18	30	34	46	50	62	70
#43	3	15	19	31	35	47	51	63	70
#44	4	8	20	24	36	40	52	56	70
#45	5	9	21	25	37	41	53	57	70
#46	6	10	22	26	38	42	54	58	70
#47	7	11	23	27	39	43	55	59	70
#48	0	13	18	31	36	41	54	59	71
#49	1	14	19	24	37	42	55	60	71
#50	2	15	20	25	38	43	48	61	71
#51	3	8	21	26	39	44	49	62	71
#52	4	9	22	27	32	45	50	63	71
#53	5	10	23	28	33	46	51	56	71
#54	6	11	16	29	34	47	52	57	71
#55	7	12	17	30	35	40	53	58	71
#56	0	14	20	26	32	46	52	58	72
#57	1	15	21	27	33	47	53	59	72
#58	2	8	22	28	34	40	54	60	72
#59	3	9	23	29	35	41	55	61	72
#60	4	10	16	30	36	42	48	62	72
#61	5	11	17	31	37	43	49	63	72
#62	6	12	18	24	38	44	50	56	72
#63	7	13	19	25	39	45	51	57	72
#64	0	15	22	29	36	43	50	57	73
#65	1	8	23	30	37	44	51	58	73
#66	2	9	16	31	38	45	52	59	73
#67	3	10	17	24	39	46	53	60	73
#68	4	11	18	25	32	47	54	61	73
#69	5	12	19	26	33	40	55	62	73
#70	6	13	20	27	34	41	48	63	73
#71	65	66	67	68	69	70	71	72	73

Tabela 4.9 przedstawia zależność pomiędzy liczbą symboli występujących na kartach, a liczbą możliwych kart w talii do wygenerowania. Aktualnie zaimplementowane są rozgrywki korzystające z 6 i 8 symboli, czyli odpowiednio 30 i 56 kart.

Tabl. 4.9. Zależność liczby symboli w stosunku do liczby kart

Liczba symboli	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Liczba możliwych do utworzenia kart	6	12	20	30	42	56	72	90	110	132	156	182	210	240	272	306	342	380

Wygenerowane karty są zapisane w odpowiednich tablicach i listach. Jako parametry przyjmowane są liczba symboli i liczba graczy. Cała talia wygenerowanych kart jest zapisywana w macierzy `int[,] tab = new int[numberOfCards, numberOfSymbols]`. W komórce [0,0] znajduje się pierwszy symbol pierwszej karty, w komórce [0,1] znajduje się drugi symbol pierwszej karty, w komórce [0, numberOfSymbols] znajduje się ostatni symbol pierwszej karty.

Kolejnym etapem jest przypisanie graczom ich kart. W tym celu najpierw jest określana liczba kart przypadających na gracza za pomocą zmiennej `double div = (numberOfCards - countOfMainCard) / numberOfPlayers`, która jest zaokrąglana w dół.

Spośród wszystkich dostępnych id kart `int numberOfCards` losowana jest jedna liczba, która będzie id głównej karty. Reszta id kart trafia do listy `List<int> availableIdOfPlayerCards`. Tablica zawierająca główną kartę jest tworzona poprzez przypisanie jej wartości z macierzy `tab`. Potem jej wartości są mieszane (ang. shuffle) przy pomocy algorytmu „the Fisher-Yates shuffle”, aby później otrzymać różne rozmieszczenie symboli. Lista `availableIdOfPlayerCards` jest również mieszana, aby pierwszy gracz nie otrzymywał zawsze początkowych id kart, a ostatni – ostatnich. W pętli tworzona jest lista dla każdego gracza `List<int> set` oraz ta lista jest dodawana do listy, która zawiera wszystkie sety graczy `List<List<int>> listOffsets`. Następną listą jest `List<int[,]> setsOfCards`, która zawiera np. dla dwóch graczy dwie tablice `int[,]`, a są w nich `setForPlayer` – tablice z numerami kart. Do listy `set` są dodawane w pętli kolejne elementy z listy `availableIdOfPlayerCards`, a kolejność symboli na danej karcie jest mieszana. Model User posiada tabelę `int[] currentCard`, w której aktualizowane są dane dotyczące aktualnie trzymanej karty za pomocą metody, która jest wywoływana, gdy gracz otrzymuje pierwszą kartę na początku rozgrywki oraz gdy gracz położy swoją kartę i wyciąga kolejną ze swojej talii.

```
void currentCardElements(int currentCardId, int numberMode) {
    for (int a = 0; a < numberMode; a++) {
```

```
    currentCard[a] = setForPlayer[currentCardId, a]; }}
```

Sprowadzając omawiane teorie i kod do przykładu dla gry dla dwóch graczy otrzymuje się:

- wygenerowanie talii 56 kart, wartości widoczne w macierzy z tabl. 4.6.
- liczbę kart dla gracza: 27 (z funkcji `cardsPerSetFunc`)
- id karty głównej: 37 (losowe id z zakresu dostępnych kart),
- dodanie wszystkich kart (55 szt) poza tą wylosowaną o id 37 do nowej listy:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,
28, 29, 30, 31, 32, 33, 34, 35, 36, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52,
53, 54, 55.

- wymieszanie id dostępnych kart:

53, 24, 0, 41, 23, 35, 55, 15, 21, 17, 40, 31, 34, 51, 19, 36, 43, 52, 6, 20, 30, 22, 10, 39, 9,
42, 11, 47, 4, 3, 14, 49, 46, 8, 13, 29, 27, 54, 33, 44, 48, 2, 7, 25, 38, 32, 50, 26, 12, 28, 18,
5, 1, 45, 16.

- utworzenie talii dla graczy:

- dla pierwszego kolejno od 0 do 27:

53, 24, 0, 41, 23, 35, 55, 15, 21, 17, 40, 31, 34, 51, 19, 36, 43, 52, 6, 20, 30, 22, 10, 39,
9, 42, 11.

- dla drugiego od 28 do 53:

47, 4, 3, 14, 49, 46, 8, 13, 29, 27, 54, 33, 44, 48, 2, 7, 25, 38, 32, 50, 26, 12, 28, 18, 5, 1,
45.

- karta o id 16 zostaje nierozdana, ponieważ każdy z graczy posiada taką samą liczbę kart wynoszącą 27, a jest jeszcze 1 karta główna, co daje w sumie 55 kart. Talia zawiera 56 kart, zatem, aby każdy gracz miał taką samą liczbę kart jedna musi pozostać nieprzydzielona.

Drugą ze składowych modelu jest klasa `User`, reprezentująca użytkownika i zawierająca następujące pola udostępnianie poprzez getery i setery:

```
private TcpClient tcpClient;  
  
private string userName;  
  
private int idNumber;
```

```

private int idRoom;

static List<int> set = new List<int>();

static int numberOfSymbols = 8;

int[,] setForPlayer = new int[set.Count, numberOfSymbols];

private int currentCardId = 0;

private int[] currentCard = new int[numberOfSymbols];

```

Posiada dwie metody, pierwsza odpowiada za zwrócenie liczby kart znajdującej się aktualnie w secie kart, a druga odpowiada za wypełnienie aktualnej karty wartościami znajdującymi się w kolejnej karcie, co zostało już omówione przy omawianiu tablicy `currentCard`.

Trzecią klasą modelu jest `Room`. Dziedziczy ona z klasy `Cards`. Jest to umotywowane tym, że klasa `Room` (klasa pochodna) korzysta w swojej definicji z klasy `Cards` (klasy bazowej). Obiekty klasy `Room` zawierają w sobie elementy klasy `Cards`. Klasa `Room` korzysta z funkcjonalności klasy `Cards` i rozszerza jej możliwości. Reprezentuje pokój, więc zawiera zmienne, które są ustawiane raz poprzez konstruktor, a są to: nazwa pokoju, id użytkownika, id pokoju, maksymalna liczba graczy w pokoju, rodzaj tryby gry, nazwa symboli, parametr `p`, lista graczy.

```

private string roomName;

private int idNumberUser;

private int idNumberRoom;

private int numberOfPlayers;

private int numberMode;

private string nameOfSymbols;

private int p;

private List<int> players = new List<int>();

```

Zawartość listy `List <int> players` zmienia się wraz z dołączaniem kolejnych graczy.

Klasa `Server` posiada listę `List<Room> room`, do której dodaje wszystkie utworzone pokoje. Dane odnośnie parametrów pokoju są czytane z aplikacji użytkownika, następnie kodowane i wysypane do serwera, który je odkodowuje, przetwarza i korzystając z metody `createNewRoom(TcpClient this_client, string[] response)` tworzy nowy pokój. Metoda tworzy obiekt `Room` wypełniając wcześniej omawiane pola konstruktora, dodaje pokój do

listy List<Room> room, wysyła odpowiedź klientowi, generuje talię kart do gry poprzez room.createCardsForGame(room.RoomNumberofPlayers, room.P). Metoda createCardsForGame ustawia w obiekcie klasy Cards wcześniej omawiane parametry. Są to numberofSymbols, który wynosi $p+1$, numberofCards, czyli $p^2 + p$, inicjuje tablicę zawierającą symbole głównej karty, tworzy talię kart, przydziela karty graczom.

4.4.3 Opracowanie komunikatów

Podczas pracy serwera odpowiada on na żądania: połączenia, wysłania do kogoś wiadomości, utworzenia nowej gry, dołączenia do istniejącej gry, wybrania jednego z symboli znajdujących się na karcie – wykonania ruchu, odświeżenia listy dostępnych pokoi, wybrania pokoju, utworzenia konta, wyświetlenia informacji dotyczących rankingu. Wszystkie one są zakodowane i po otrzymaniu jakiejkolwiek z nich serwer wywołuje odpowiednią metodę, która wykonuje docelowe żądanie. W przejrzysty sposób obrazuje to metoda:

```
private void connectionHandler(TcpClient client) {  
    TcpClient this_client = client;  
  
    Stream clientStream = this_client.GetStream();  
  
    StreamWriter strWriter = new StreamWriter(clientStream);  
  
    StreamReader strReader = new StreamReader(clientStream);  
  
    string clientMessage;  
  
    while (this.isServerRunning == true) {  
  
        clientMessage = strReader.ReadLine();  
  
        string[] response = clientMessage.Split(';');  
  
        if (response[0] == "CONNECT_REQUEST")  
  
        {  
  
            connectRequest(client, response);  
  
        }  
  
        if (response[0] == "SEND_MSG")  
  
        {  
  
            chatWithOtherUserTest(this_client, response);  
  
        }  
  
        if (response[0] == "NEW_GAME")  
    }
```

```

    {

        createNewRoom(this_client, response);

    }

    if (response[0] == "JOIN")

    {

        join(this_client, response);

    }

    if (response[0] == "CHOSEN_PIC")

    {

        chosenPic(this_client, response);

    }

    if (response[0] == "REFRESH_LIST")

    {

        refreshList(this_client, response);

    }

    if (response[0] == "CREATE_ACCOUNT")

    {

        createAccount(this_client, response);

    }

    if (response[0] == "RANKING")

    {

        createRanking(this_client, response);

    }

}

}

```

Po uruchomieniu serwera i rozpoczęcia nasłuchu następuje wywołanie metody startListing(), która odpowiada za rozpoczęcie nasłuchu i utworzeniu wątku w tle odpowiadającego za podtrzymywanie nasłuchu. Dla każdego użytkownika, który się podłączy tworzy osobny wątek.

Nicki wszystkich aktualnie podłączonych użytkowników serwer trzyma zapisane w liście
List<User> tcpClients = new List<User>(). Klient wysyła komunikaty za pomocą zakodowanej w łańcuchu znaków wiadomości. Dane są oddzielane za pomocą średników. Pierwsza dana dotyczy rodzaju komunikatu (CONNECT_REQUEST, SEND_MSG, NEW_GAME itd.), a następne są opcjonalne. Po wysłaniu:

```
string connectionEstablish = "CONNECT_REQUEST;" + userNameTextbox + ";" +  
passwordTextBox;  
  
strWriter.WriteLine(connectionEstablish);
```

Serwer sprawdzając warunek if (response[0] == "CONNECT_REQUEST") wykona metodę connectRequest(client, response). W response[1] w tym przypadku znajduje się nick, a w response[2] hasło.

Wysłanie komunikatu CREATE_ACCOUNT stanowi żądanie utworzenia nowego konta. Następuje walidacja danych podanych przez użytkownika. Sprawdzane jest, czy nick jest już zajęty, zawiera białą spację, jest dłuższy niż 20 znaków, czy hasło jest krótsze niż 6 znaków, czy adres e-mail zawiera „@”. Jeśli którekolwiek ze sprawdzeń okaże się być twierdzące, to serwer wysyła wiadomość o błędnie wpisanych danych. W przeciwnym razie zakłada konto i zapisuje dane w bazie danych.

Przetworzenie komunikatu CONNECT_REQUEST polega na sprawdzeniu, czy użytkownik o podanym loginie i haśle figuruje w bazie danych. Jeśli tak, to łączy się z serwerem.

SEND_MSG to wysłanie wiadomości do innego gracza. W parametrach przekazywane są nick odbiorcy oraz wiadomość. Serwer odszukuje wśród zalogowanych użytkowników odbiorcy i wysyła mu wiadomość wraz z powiadomieniem, kto jest nadawcą.

NEW_GAME odpowiada za utworzenie nowego pokoju i gry. Później następuje oczekiwanie na pozostałych graczy. Gdy liczba graczy jest równa liczbie graczy docelowej pokoju następuje wysłanie komunikatu o rozpoczęciu gry do wszystkich graczy danego pokoju. Odkrywana jest pierwsza główna karta oraz każdy z graczy widzi pierwszą własną kartę. Aktualizowany jest status gry.

CHOSSEN_PIC po otrzymaniu tego komunikatu następuje ustalenie, jaki numer symbolu wybrał gracz i czy powtarza się on na głównej karcie. Gracz wysyła tag symbolu, który odpowiada jego pozycji. W grze na 8 symboli tagi są ponumerowane od 1 do 8 i nie zmieniają pozycji – są przypisane do miejsc i pictureBoxów. Gracz klikając w środkowy symbol wysyła nr 1 na serwer,

co odpowiada pierwszej pozycji w jego secie kart. Więcej o tym mechanizmie znajduje się w podrozdziale 4.5.2. Następuje odczytanie z tablicy mainCardTab wartości znajdującej się pod id o jeden mniejszym (ponieważ tablice numerowane są od 0, a nie 1) i przekazanie jej do sprawdzenia, czy znajduje się ten symbol na karcie gracza. Jeśli tablica gracza currentCard zawiera dany symbol, to klientowi pokazywana jest jego kolejna karta ze stosu, a jego stara karta jest umieszczana w miejscu karty głównej i status gry jest odświeżany. Jeśli natomiast symbole nie pokrywają się, to gracz otrzymuje karę czasową (ang. ban) na 3 sekundy podczas których inni gracze mogą dokonywać wyborów, a ukarany gracz nie może wykonywać żadnego ruchu.

JOIN to żądanie dołączenia do pokoju. Jeśli w pokoju przebywa już maksymalna liczba graczy, to nowy gracz nie może dołączyć. W przeciwnym razie jest dodawany do pokoju i przechodzi do oczekiwania na resztę graczy. Jeśli zapełnił ostatnie miejsce, to gra się zaczyna.

REFRESH_LIST jest przydatne, przy odświeżaniu widoku pokoi.

RANKING służy do pobrania danych z bazy danych i pokazania ich w postaci najlepszych dziesięciu graczy. Generowanie rankingu i komunikacja z bazą danych była opisana w podrozdziale 4.3.

4.5 Klient

Aplikacja kliencka wysyła żądania do serwera, a serwer je obsługuje. Użytkownik musi znać adres IP i numer portu serwera, aby się z nim połączyć. Raz wprowadzone dane zostają zapisane w UserAppDataRegistry, aby korzystanie z aplikacji było bardziej komfortowe i nie wymagało podawania numerów ręcznie za każdym razem. Oczywiście dane gniazda można edytować przed próbą połączenia.

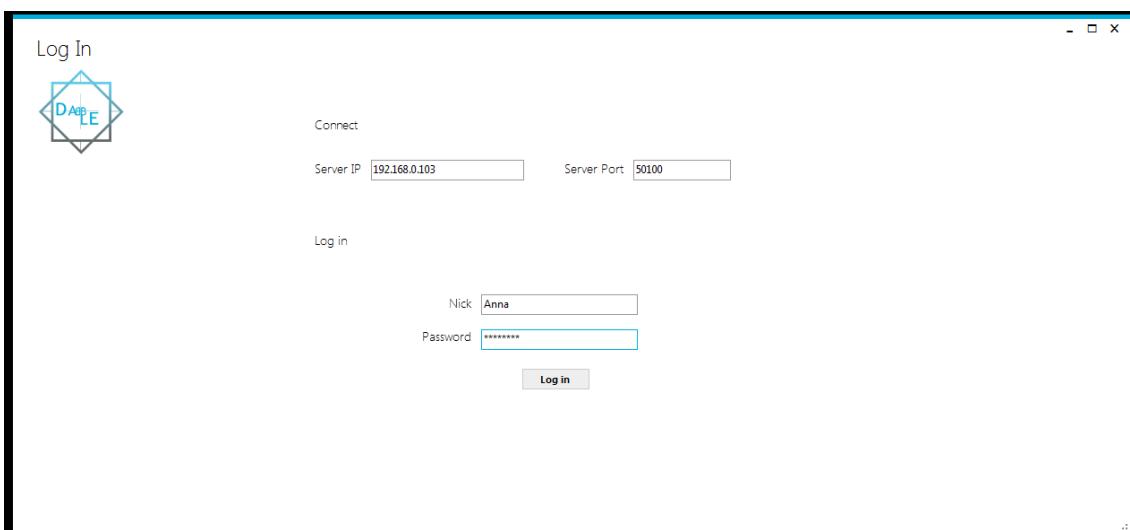
Projekt klienta opierał się w dużej mierze na zaprojektowaniu GUI, które spełnia potrzeby użytkowników oraz implementacji. Składa się z ośmiu form, pomiędzy którymi nawigacja odbywa się za pomocą kontrolera.

4.5.1 Prezentacja widoków

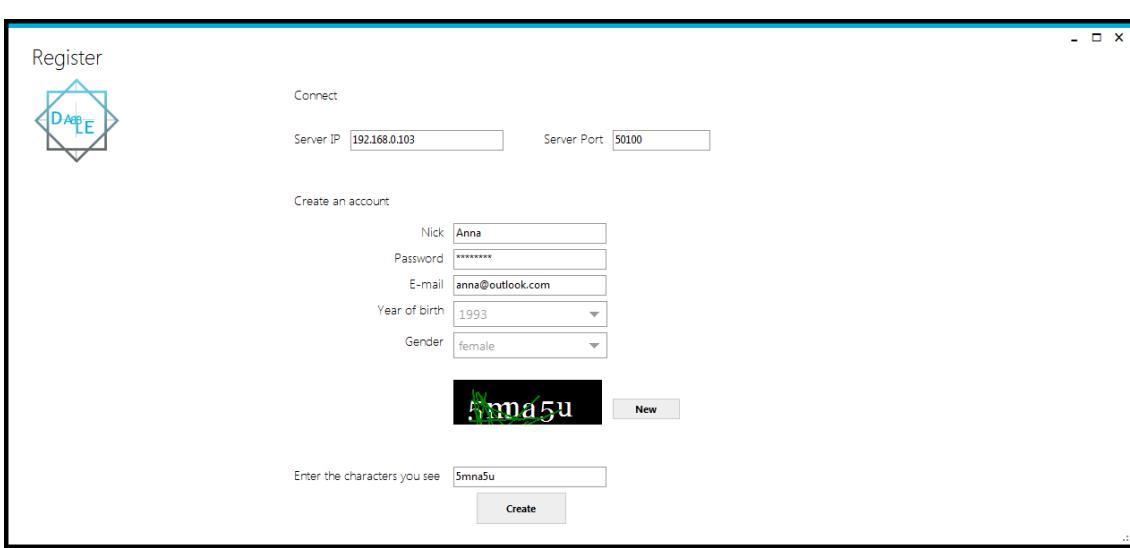
Na rys. 4.10 – 4.17 zostały zaprezentowane widoki: startu, logowania, rejestracji, tworzenia pokoju, gry, dołączenia do pokoju, menu, rankingu, z czego widok gry jest oknem, w którym odbywa się cała rozgrywka i będzie najszerzej omówiony. Do zaprojektowania okien, tak samo jak w przypadku serwera, wykorzystano bibliotekę MetroForms. Tła form są białe, tak samo jak tło kartki, więc okna aplikacji zostały obramowane na czarno, by lepiej je pokazać.



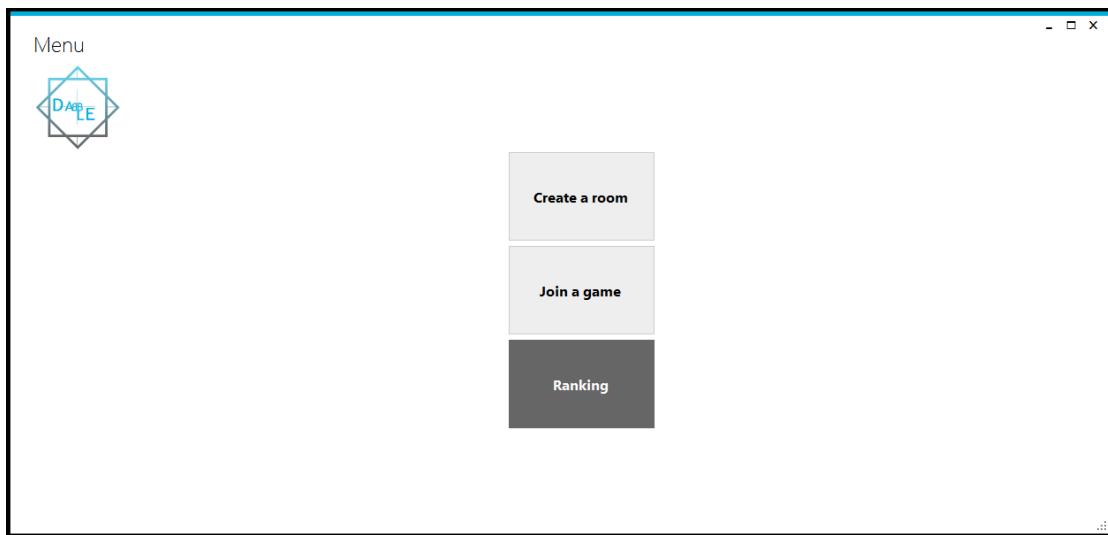
Rys. 4.10. Menu



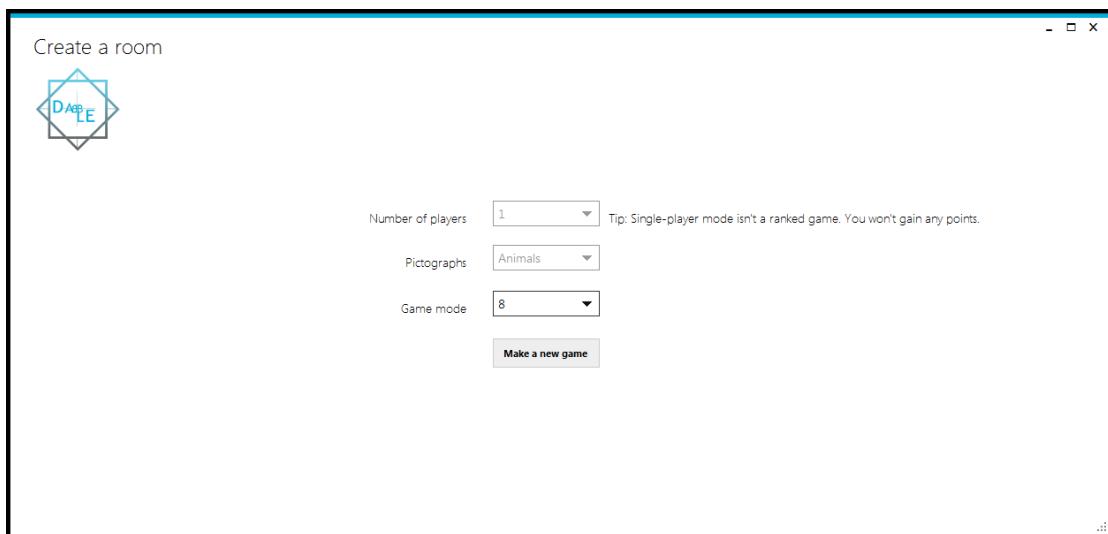
Rys. 4.11. Logowanie



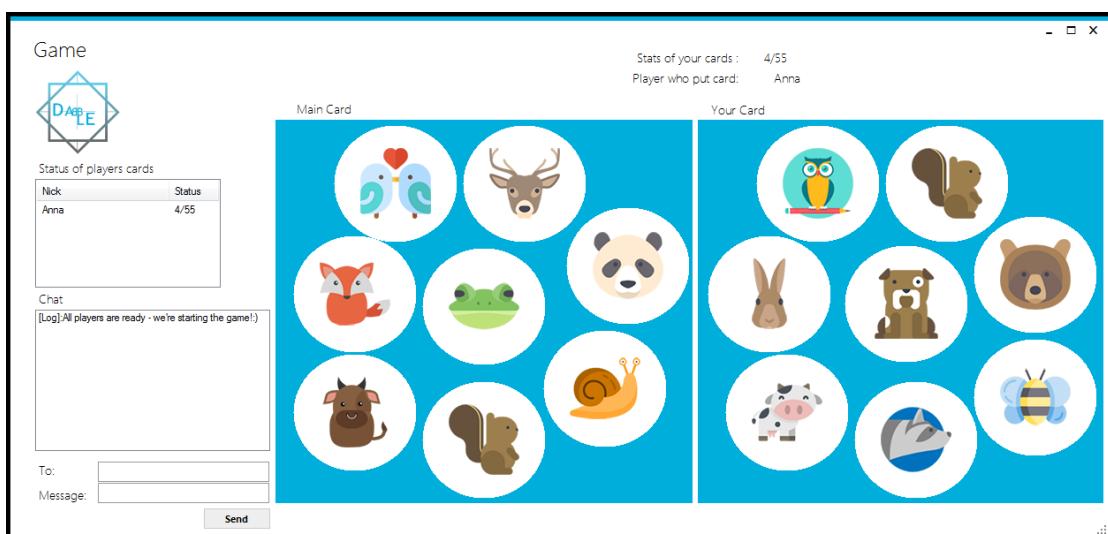
Rys. 4.12. Rejestracja



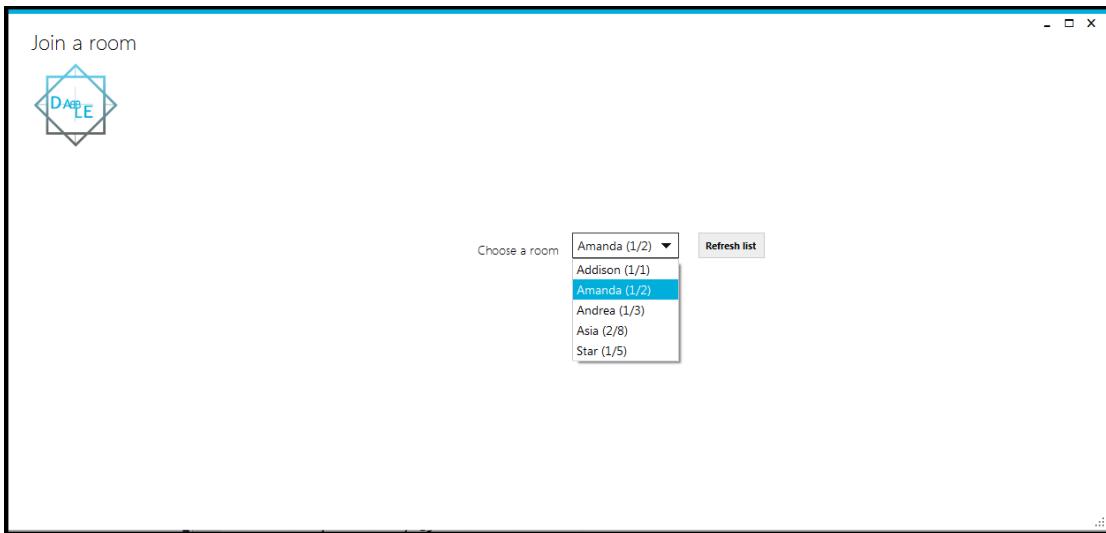
Rys. 4.13. Menu



Rys. 4.14. Create a room



Rys. 4.15. Game



Rys. 4.16. Join a room

Rank	Nick	Score
1	Chelsa	1000
2	Linn	920
3	Poppy	720
4	Happy	596
5	lucky	500
6	Alcott	480
7	Paxton	254
8	Dylan	80
9	Garfield	60
10	Anna	40
11	True570	23

Rank	Nick	Score
1	ProPlayer	2512
2	Tayler	1998
3	dabria1478	1752
4	Linn127	1750
5	Dee344	1532
6	lucky1733	1484
7	Xayah	1284
8	True570	115
9	Dudu	40
10	Addison	12
11	Anna	6

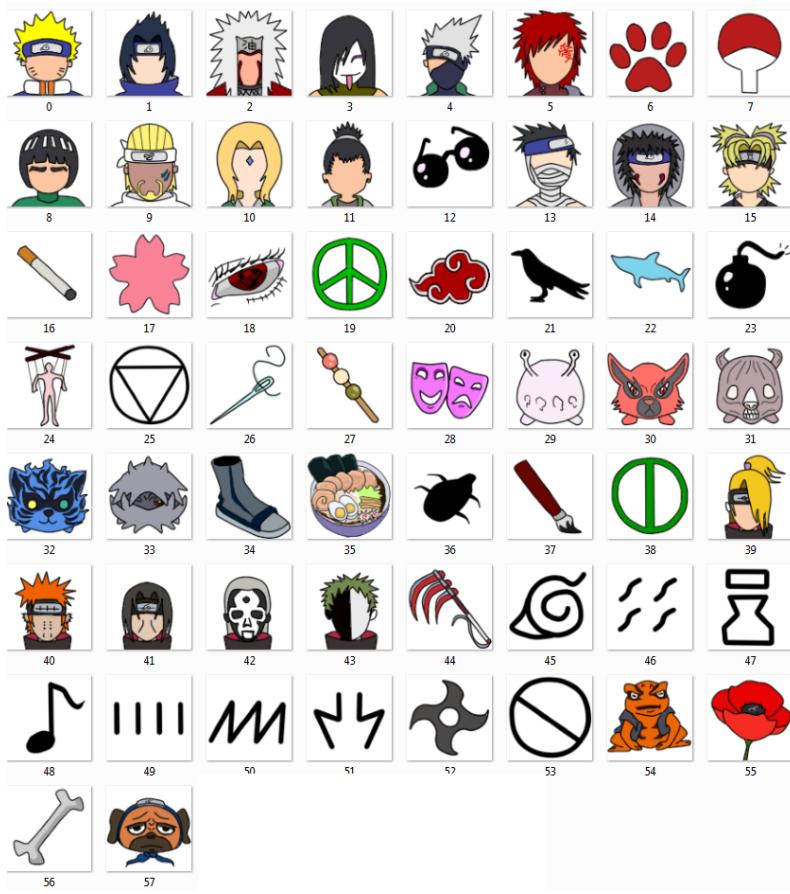
Rys. 4.17. Ranking

Po uruchomieniu aplikacji klienckiej wyświetla się okno Start przedstawione na rys 4.10, gdzie możliwe do wyboru są dwa przyciski - „Sign in” oraz „Create an account” odpowiadające za logowanie oraz zakładanie nowego konta. Widok okna logowania pokazany jest na rys 4.11 – należy podać adres IP i numer portu serwera oraz swój nick wraz z hasłem. W przypadku niepowodzenia zostaje wyświetlony odpowiedni komunikat informujący o niepoprawnie podanych danych albo o nieudanym niepołączeniu do serwera. Rys 4.12 przedstawia formularz rejestracyjny. Należy podać swój nick, hasło, adres e-mail, rok urodzenia, płeć oraz CAPTCHA (ang. Completely Automated Public Turing test to tell Computers and Humans Apart). CAPTCHA jest techniką zabezpieczeń stosowaną, aby dopuścić do przesłania danych tylko te, które zostały wypełnione przez człowieka, a nie maszynę. Formularze są chronione przed spamem poprzez konieczność odczytania i wpisania znaków (liter oraz cyfr), które znajdują się na obrazku. Odczytanie zawartości obrazka przez maszynę jest z założenia trudne. W projekcie został wykorzystany

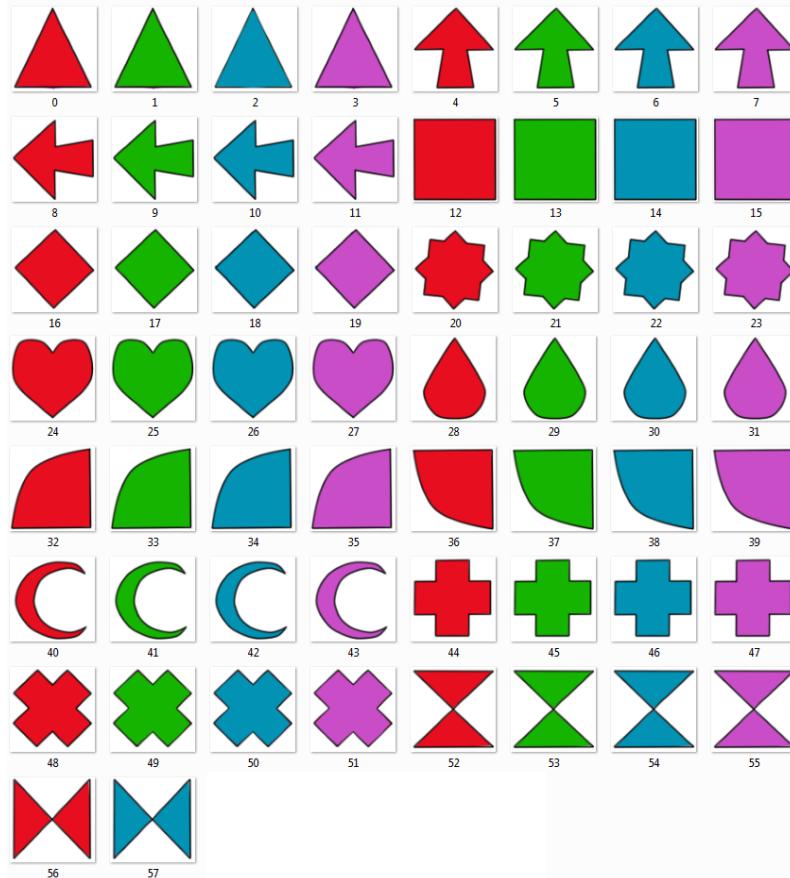
algorytm generowania CAPTCHA na licencji Apache, załączony w odnośniku [14]. Główne Menu prezentuje rys. 4.13. Możliwe są do wyboru przyciski „Create a room”, „Join a room”, „Ranking”. Okno tworzenia pokoju przedstawia rys 4.15. Do wyboru są następujące opcje „Number of players” od 1 do 8, „Pictographs” - wybór pictogramów zawierających obrazki anime, kształty, jedzenie, zwierzęta, „Game mode” - 8 (tryb trudniejszy), 6 (tryb łatwiejszy). Po wybraniu parametrów należy nacisnąć przycisk „Make a new game”. Dołączenie do pokoju pokazuje rys 4.16. Zawiera listę rozwijaną z dostępnymi pokojami wraz z ich nazwą, liczą aktualnie przebywających graczy, liczbę wymaganej liczby graczy. Rys 4.17 prezentuje ranking, gdzie z lewej strony znajduje się tryb trudniejszy, z prawej – łatwiejszy. Pokazywane są informacje dotyczące pozycji w rankingu, nicku, zgromadzonych punktów. Widok gry przedstawia rys 4.15. Pod logo z lewej strony znajduje się lista graczy oraz ich status kart, a pod nim czat. Centralną część okna zajmują karty. Z lewej strony znajduje się główna karta, w której okrągłe obrazki można kliknąć, a z prawej karta gracza. Nad nimi jest umieszczona informacja dotycząca stanu kart gracza oraz nick gracza, który jako ostatni położył kartę. Obrazki (rys.4.18. - rys. 4.21.) zostały wykonane własnoręcznie dla pictogramów „anime” oraz „shapes”, natomiast „animals” oraz „food” zostały pobrane ze strony umieszczonej odnośniku [15].



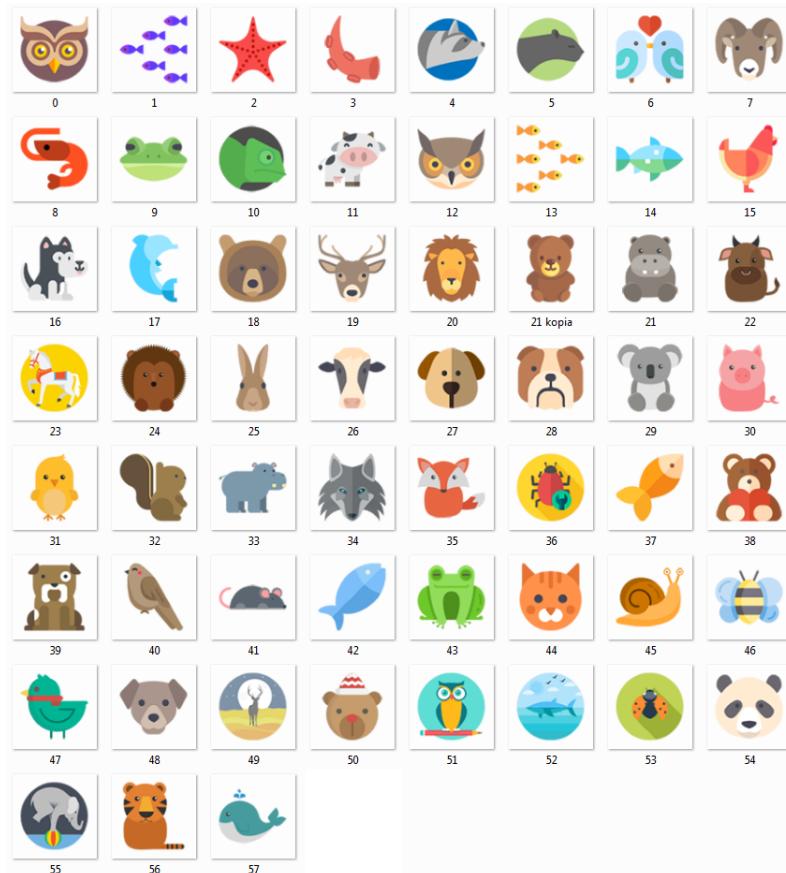
Rys. 4.19. Piktogramy „food”



Rys. 4.20. Piktogramy "anime"



Rys. 4.21. Piktogramy "shapes"



Rys. 4.22. Piktogramy "animals"

4.5.2. Omówienie funkcji kontrolera

Metoda wysyłania komunikatów odbywa się analogicznie, jak w przypadku serwera. Po wysłaniu żądania klient otrzymuje odpowiedź, a mogą to być: nick jest dostępny, niepowodzenie logowania, niepoprawny nick, nadchodząca wiadomość, liczba graczy, nazwa symboli, pokój nieznaleziony, pokój nieznaleziony, pokój pełen, rozpoczęcie gry, czekanie na pozostałych graczy, symbole głównej karty, symbole karty gracza, liczba kart w talii, numer aktualnej karty, nick ostatniego gracza, który położył kartę, ban, unban, wygrana, lista pokoi, czyszczenie statusu, czyszczenie listy pokoi, status graczy, ranking trudniejszego trybu, ranking łatwiejszego trybu.

Rozkładanie kart stanowi kluczowy element gry. Po otrzymaniu komunikatu `MAIN_CARD` klient odczytuje z przesłanej tablicy stringów liczby i przypisuje je do zmiennych typu `string`. Następnie konwertuje te dane w postać liczbową i przypisuje kolejno do tablicy `int cards.MainCardTab[]`. Ostatnim krokiem jest wyświetlenie danych poprzez `_viewGame.SetMainCard(path, cards.MainCardTab)`, gdzie `path` jest ścieżką do obrazków. Obrazki pogrupowane w odpowiednie foldery tematyczne są zapisane w folderze

znajdującym się po stronie klienta w folderze bin/Debug/Images. Metoda odpowiedzialna za wyświetlanie obrazków dla głównej karty z ośmioma symbolami wygląda następująco:

```
public void SetMainCard(string path, int[] tab)
{
    string dir =
Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);

    ovalPictureBox1.Image = Image.FromFile(Path.Combine(dir, path
+ tab[0] + ".png"));

    ovalPictureBox2.Image = Image.FromFile(Path.Combine(dir, path
+ tab[1] + ".png"));

    ovalPictureBox3.Image = Image.FromFile(Path.Combine(dir, path
+ tab[2] + ".png"));

    ovalPictureBox4.Image = Image.FromFile(Path.Combine(dir, path
+ tab[3] + ".png"));

    ovalPictureBox5.Image = Image.FromFile(Path.Combine(dir, path
+ tab[4] + ".png"));

    ovalPictureBox6.Image = Image.FromFile(Path.Combine(dir, path
+ tab[5] + ".png"));

    ovalPictureBox7.Image = Image.FromFile(Path.Combine(dir, path
+ tab[6] + ".png"));

    ovalPictureBox8.Image = Image.FromFile(Path.Combine(dir, path
+ tab[7] + ".png"));

}
```

Wczytywanie obrazów karty gracza odbywa się analogicznie dla komponentów ovalPictureBox9 – ovalPictureBox16 z wykorzystaniem tablicy int cards.PlayerCardTab[] i komunikatu PLAYER_CARD.

Każdemu komponentowi ovalPictureBox głównej karty został nadany tag od 1 do 8. Zatem ovalPictureBox1 ma przypisany tag 1, ovalPictureBox2 ma tag 2, ..., ovalPictureBox8 ma tag 8. Wartości tagów odpowiadają indeksom z tablicy int MainCardTab[], ale są powiększone o jeden. Po wybraniu danego obrazka następuje wywołanie metody, która wysyła do kontrolera tag w parametrze metody chosenPic(object tag). Ona z

kolei wysyła tag na serwer. Serwer przetwarza otrzymaną wiadomość i otrzymuje numer wybranego symbolu poprzez odnalezienie w tablicy `MainCardTab[]` wartości znajdującej się pod id o jeden mniejszym niż tag.

5. TESTY GRY SIECIOWEJ

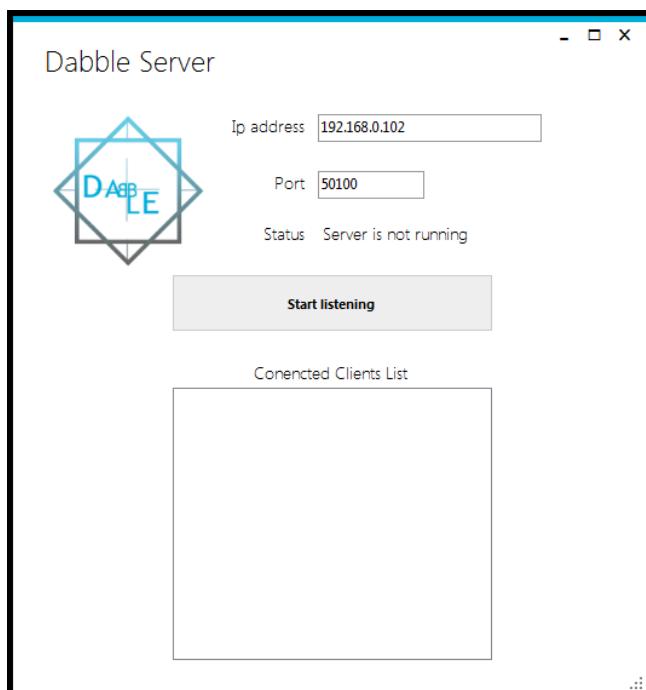
5.1 Warunki techniczne przeprowadzonych testów

Gra testowana była pod systemem Windows 7 oraz Windows 8. Testy zostały przeprowadzone na siedmiu komputerach. Pierwszym był laptop Lenovo Y580 z procesorem Intel Core i7-3630QM @ 2.40 GHz wyposażonym w 8.0GB pamięci RAM i dwie karty graficzne: NVIDIA GeForce GTX 660M i zintegrowaną Intel HD Graphics 4000. Drugi to PC z procesorem AMD Phenom II X4 955 3.2 GHz, pamięcią RAM 8.0GB i kartą graficzną NVIDIA GeForce GTX 950. Pozostałe komputery miały podobne parametry do przedstawionego PC, ale miały zainstalowany system Windows 8.

W każdym przypadku należało wyłączyć zaporę systemu Windows – Firewall, aby połączenie sieciowe mogło być zrealizowane. Wszystkie podłączone komputery musiały znajdować się w jednej sieci lokalnej (LAN albo Wi-Fi), aby mogły połączyć się z serwerem, który korzystał z wewnętrznego, a nie zewnętrznego adresu IP.

5.2 Przeprowadzane testy

Testowanie gry zaczyna się od uruchomienia programu .exe aplikacji serwera. Pokazuje się okno widoczne na rys. 5.1. Po naciśnięciu na przycisk „Start listening” następuje rozpoczęcie nasłuchu na nadchodzące połączenia (rys. 5.2.). Do elementów listy „Connected Client List” dodawane są kolejne logi pokazujące nick gracza i jego id przydzielone mu przez serwer.



Rys.5.1.Włączenie aplikacji serwera

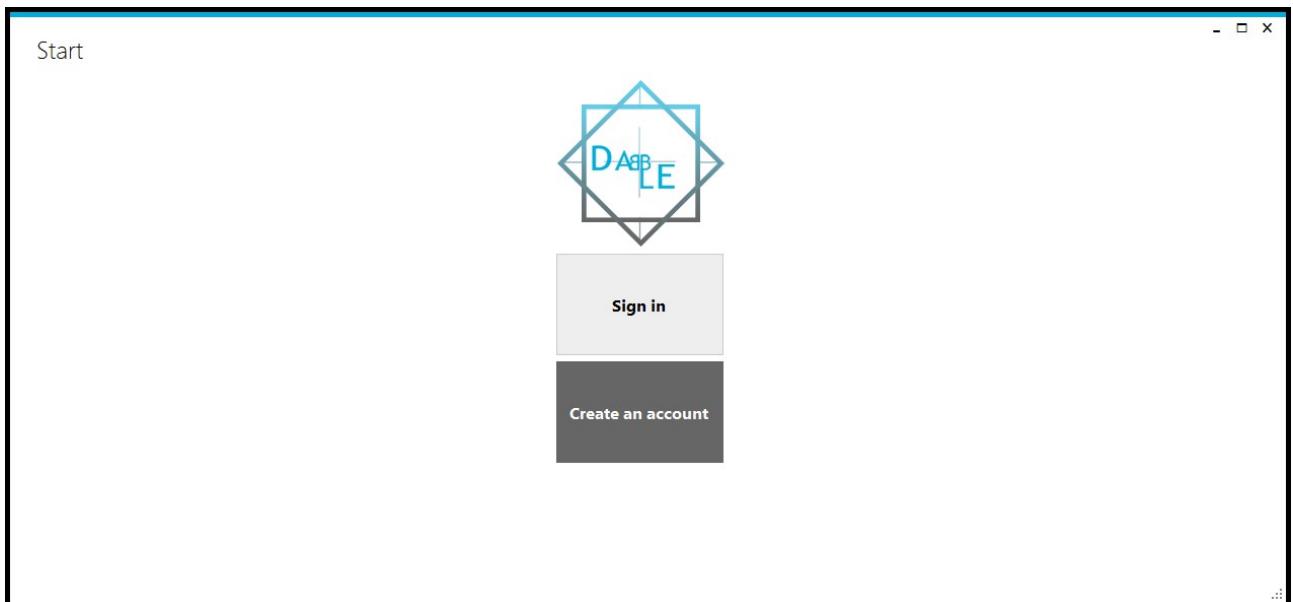


Rys. 5.2. Włączenie serwera



Rys. 5.3. Serwer w trakcie pracy

Klient włącza grę poprzez uruchomienie programu .exe. Wyświetla się wówczas ekran startowy z dwoma możliwościami wyboru. Zakładanie nowego konta odbywa się poprzez wybór opcji „Create an account” (rys. 5.4.).

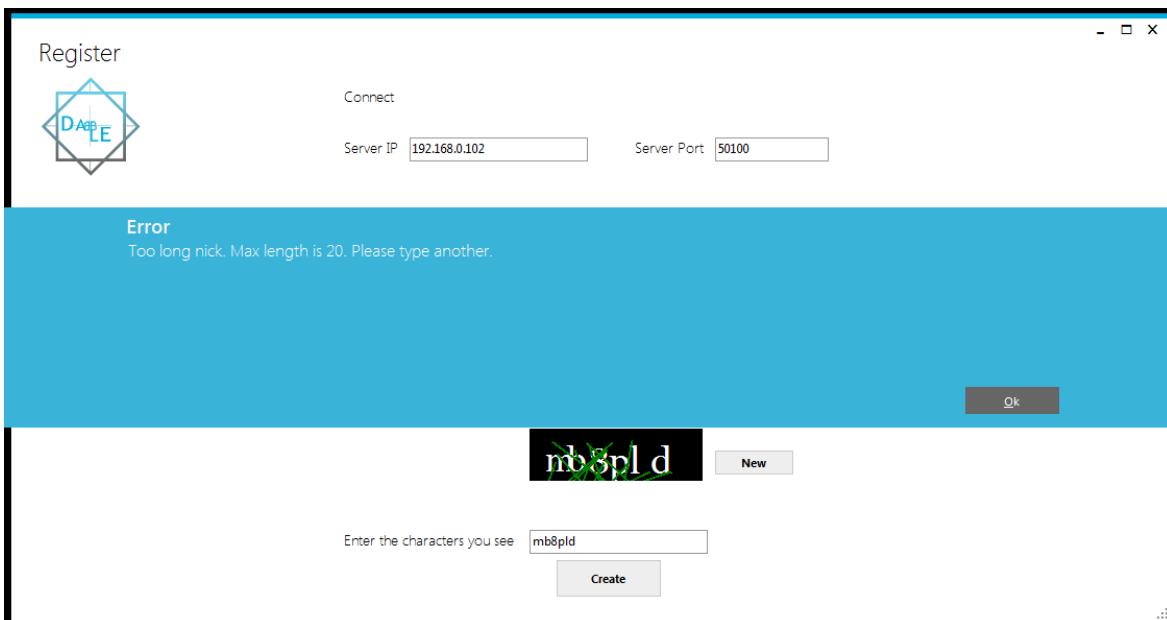


Rys. 5.4. Ekran startowy – wybór zakładania konta

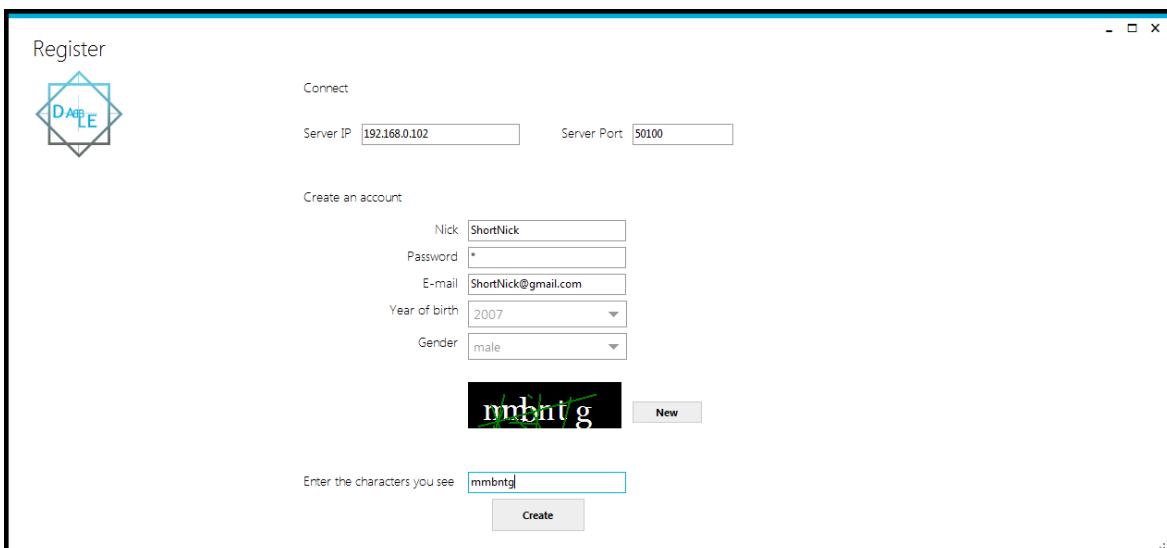
Walidacja formularza obejmuje pola, które użytkownik sam wypełnia, a są to: nick, hasło, e-mail. W przypadku podania niedopuszczalnych wartości serwer wysyła komunikat o błędzie (rys.5.5 – rys. 5.9.). Captcha jest sprawdzana po stronie klienta, aby nie obciążać serwera (rys. 5.10-5.12).

A screenshot of a Windows application window titled "Register". It features a logo at the top left and a "Connect" button with fields for "Server IP" (192.168.0.102) and "Server Port" (50100). Below this is a "Create an account" section with fields for Nick (containing the invalid value "ITryToTypeTooLongNick"), Password (*****), E-mail (sample@gmail.com), Year of birth (1987), and Gender (other). A CAPTCHA field shows the text "mb8pld" with some green scribbles over it, and a "New" button is next to it. At the bottom, there is a field labeled "Enter the characters you see" with the value "mb8pld" and a "Create" button.

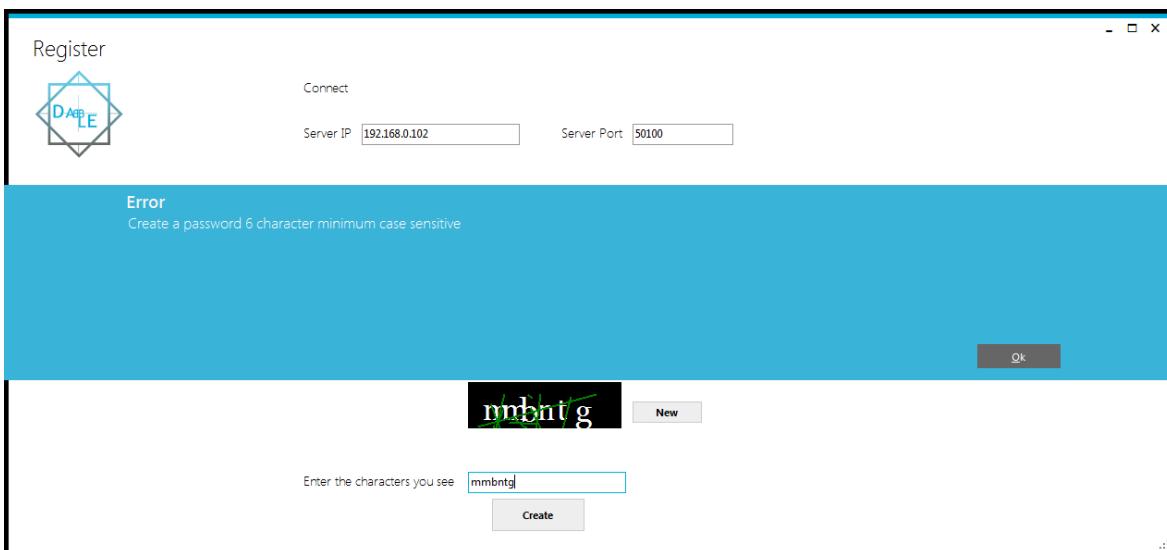
Rys. 5.5. Rejestracja - wpisanie za długiego nicku



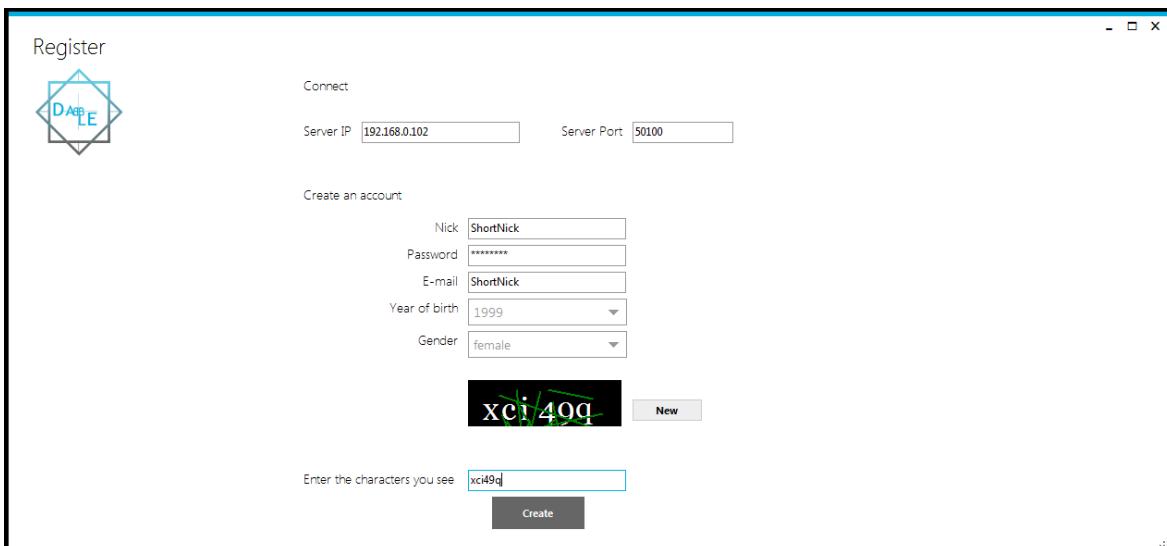
Rys. 5.6. Rejestracja - komunikat odnośnie za długiego nicku



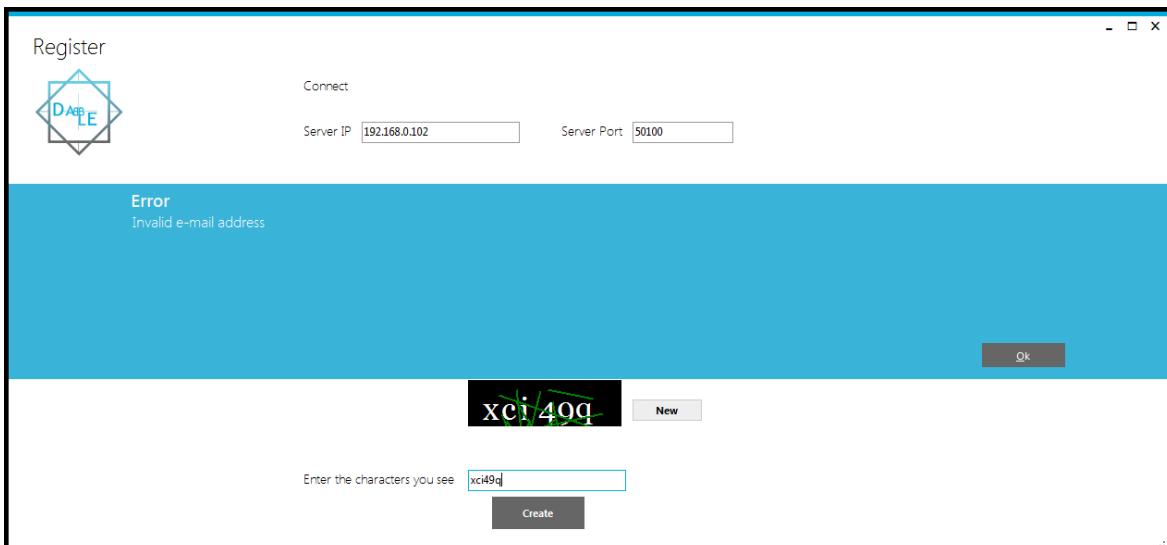
Rys. 5.7. Rejestracja - wpisanie niepoprawnego hasła



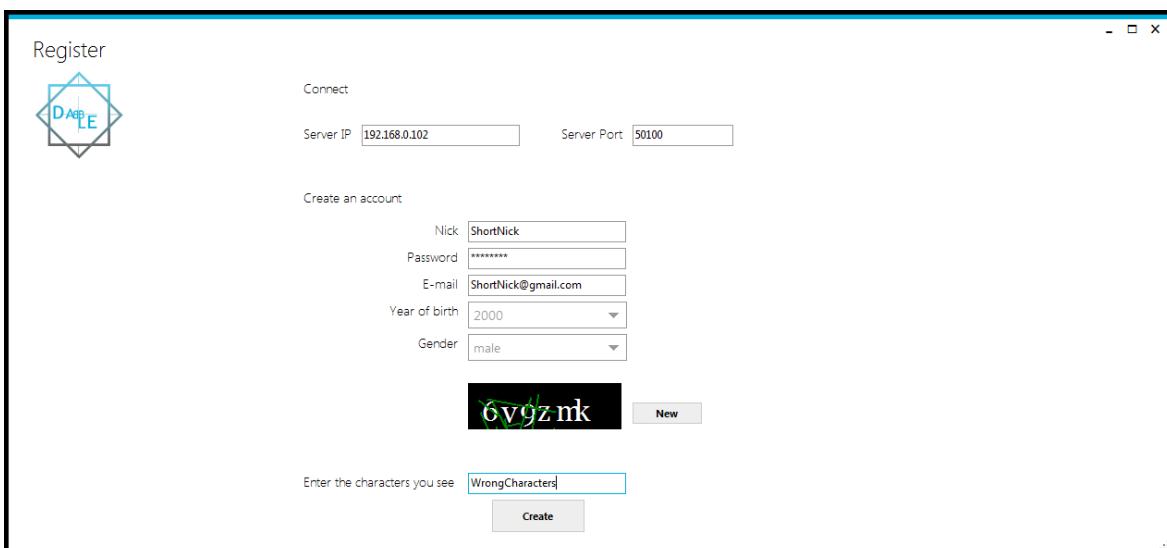
Rys. 5.8. Rejestracja - komunikat odnośnie niepoprawnego hasła



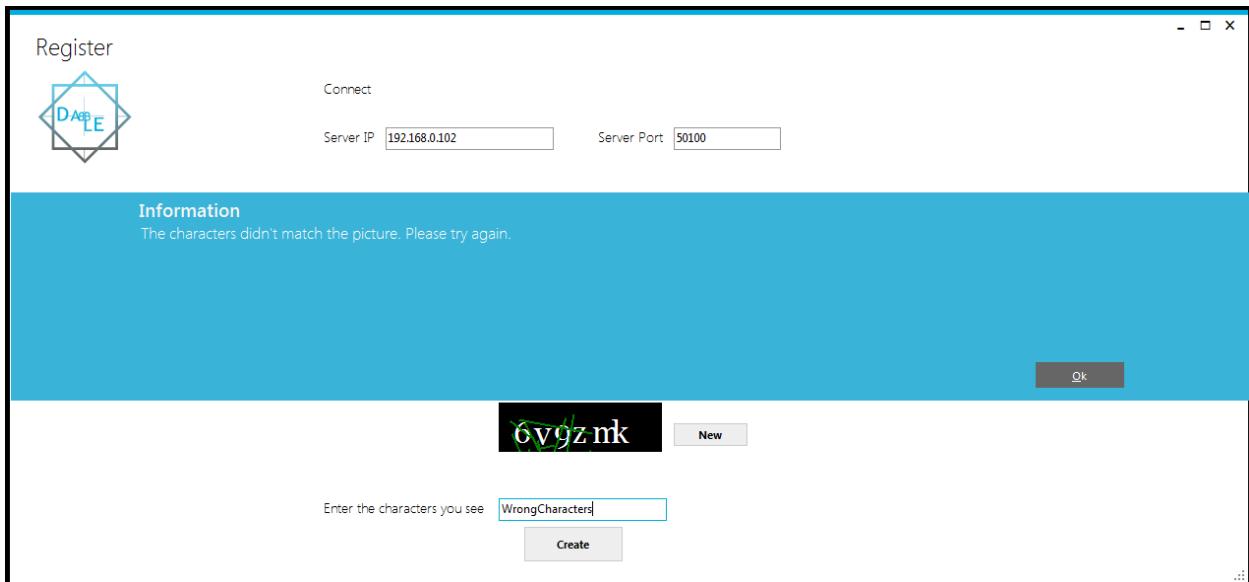
Rys. 5.9. Rejestracja - wpisanie niepoprawnego adresu e-mail (bez „@”)



Rys. 5.10. Rejestracja - Komunikat odnośnie niepoprawnego adresu e-mail

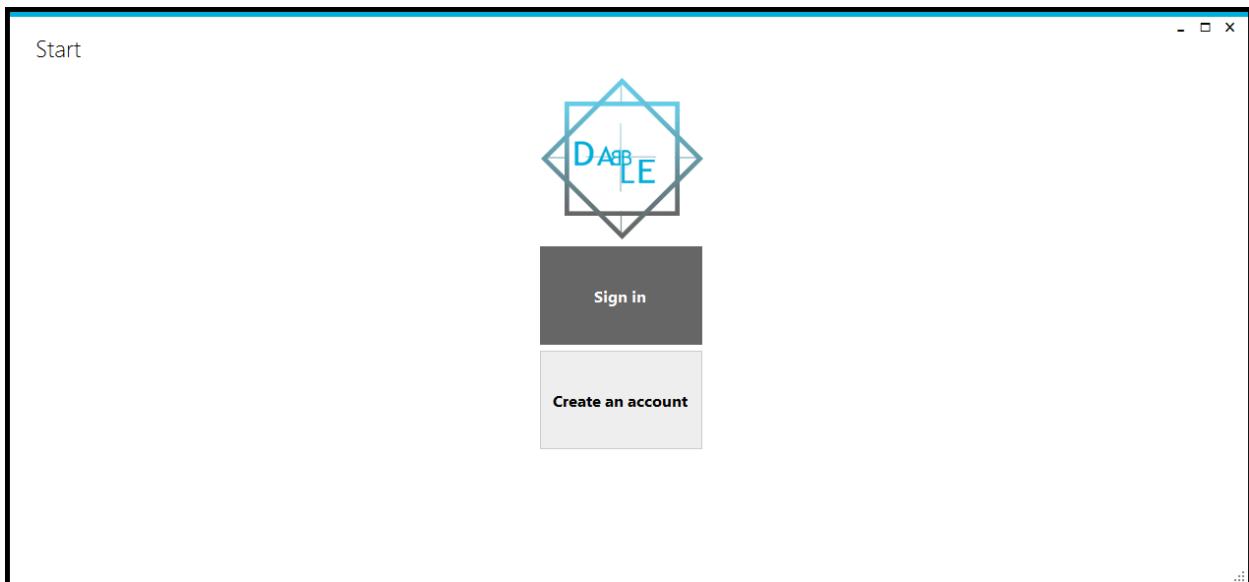


Rys. 5.11. Rejestracja - źle wpisana Captcha

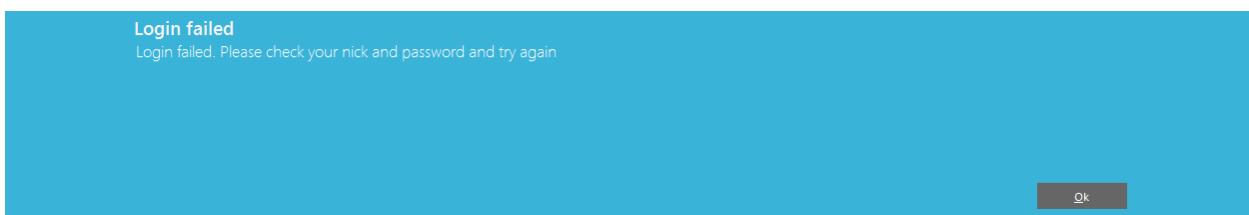


Rys. 5.12. Rejestracja - Komunikat odnośnie niepoprawnie przepisanych znaków

Testowanie gry w praktyce opierało się na prowadzeniu rozgrywki. W tym celu gracz najpierw się loguje poprzez wybór opcji „Sign in” (rys. 5.13.), a później uzupełnia formularz logowania (rys. 4.11.). Jeśli poda nick lub hasło, które nie jest zapisane w bazie danych, to serwer powiadomi go o nieudanej próbie logowania komunikatem z rys. 5.14. W przeciwnym wypadku następuje przekierowanie do Menu (rys. 4.13.). Aby założyć pokój należy wybrać opcję „Create a room”.



Rys. 5.13. Ekran startowy – wybór logowania



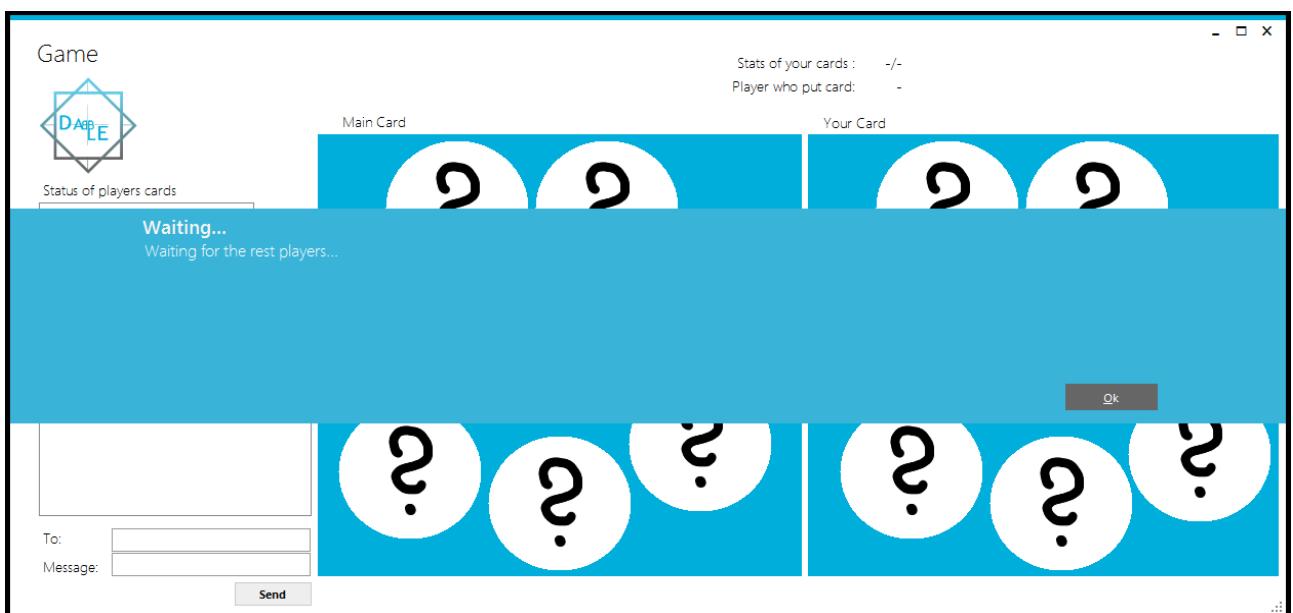
Rys. 5.14. Komunikat nieudanego logowania

Dostępne parametry zakładanego pokoju pokazuje rys. 5.15. Są to: liczba graczy od 1 do 8, rodzaj pictogramów: anime, kształty, jedzenie, zwierzęta, tryb gry: 8 (trudniejszy), 6 (łatwiejszy).

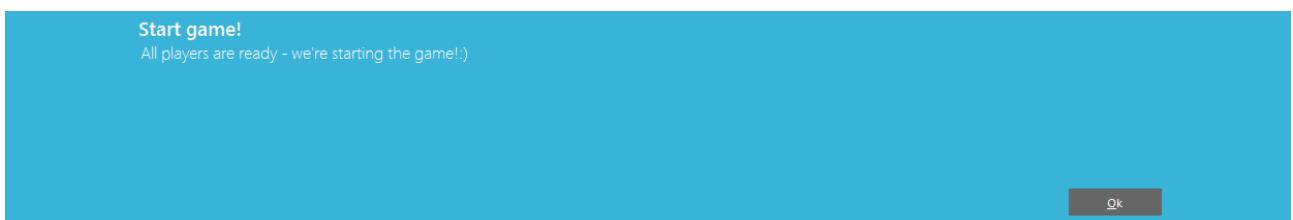


Rys. 5.15. Parametry zakładanego pokoju

Jeśli po dołączenia do pokoju brakuje graczy, to wyświetla się komunikat informujący o czekaniu (rys. 5.16.). Karty są odwrócone koszulkami do góry i pictogramy są niewidoczne. Gdy dołączy ostatni wymagany gracz, każdy z graczy znajdujących się w pokoju otrzymuje komunikat o rozpoczęciu gry, widoczny na rys. 5.17.



Rys. 5.16. Komunikat czekania



Rys. 5.17. Komunikat rozpoczęcia gry

Istnieją jeszcze inne komunikaty takie jak powiadomienie o dołączeniu do pokoju (rys. 5.18), o wygranej (rys. 5.19), o karze czasowej (rys. 5.20), o ukończeniu kary czasowej (5.21).



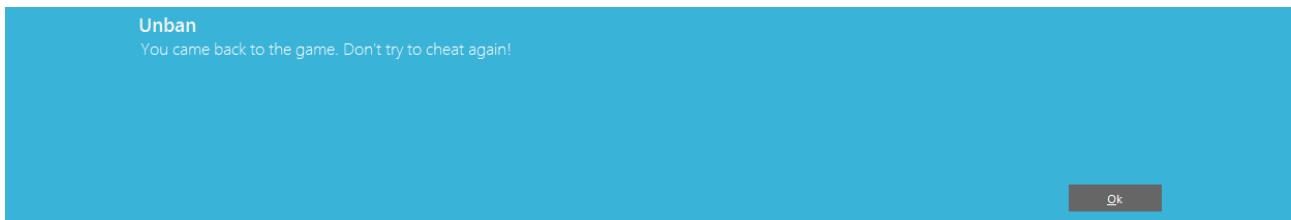
Rys. 5.18. Komunikat o dołączeniu do pokoju



Rys. 5.19. Komunikat o wygranej

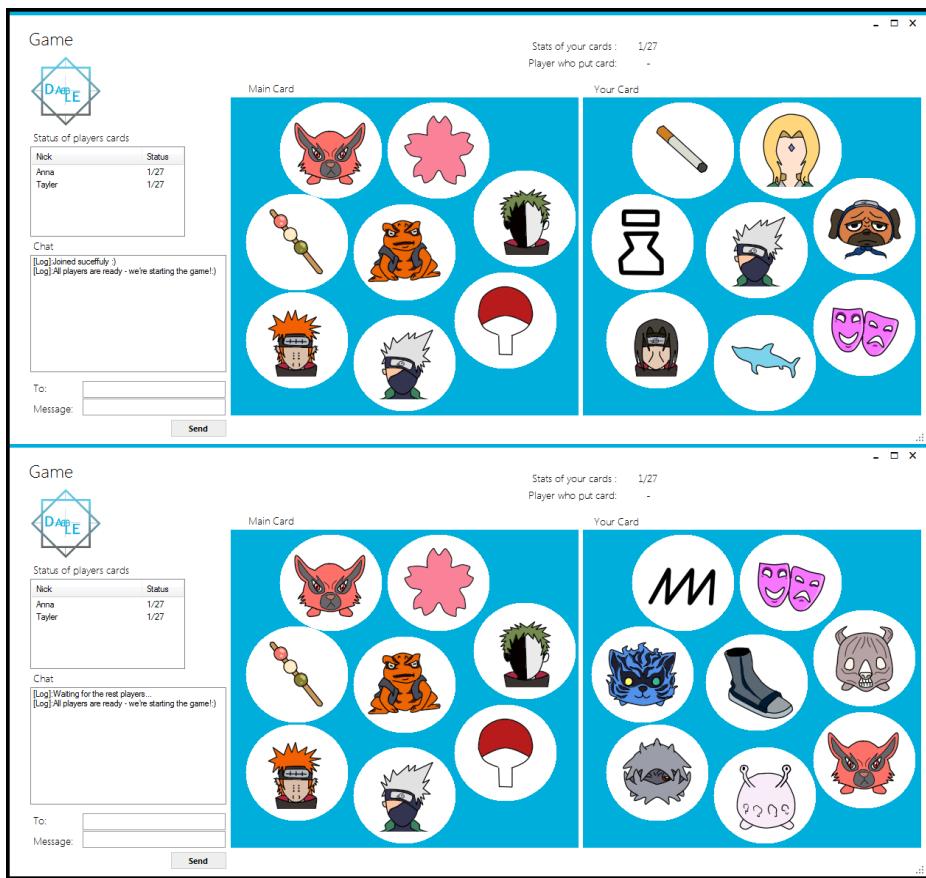


Rys. 5.20. Komunikat o otrzymaniu kary czasowej

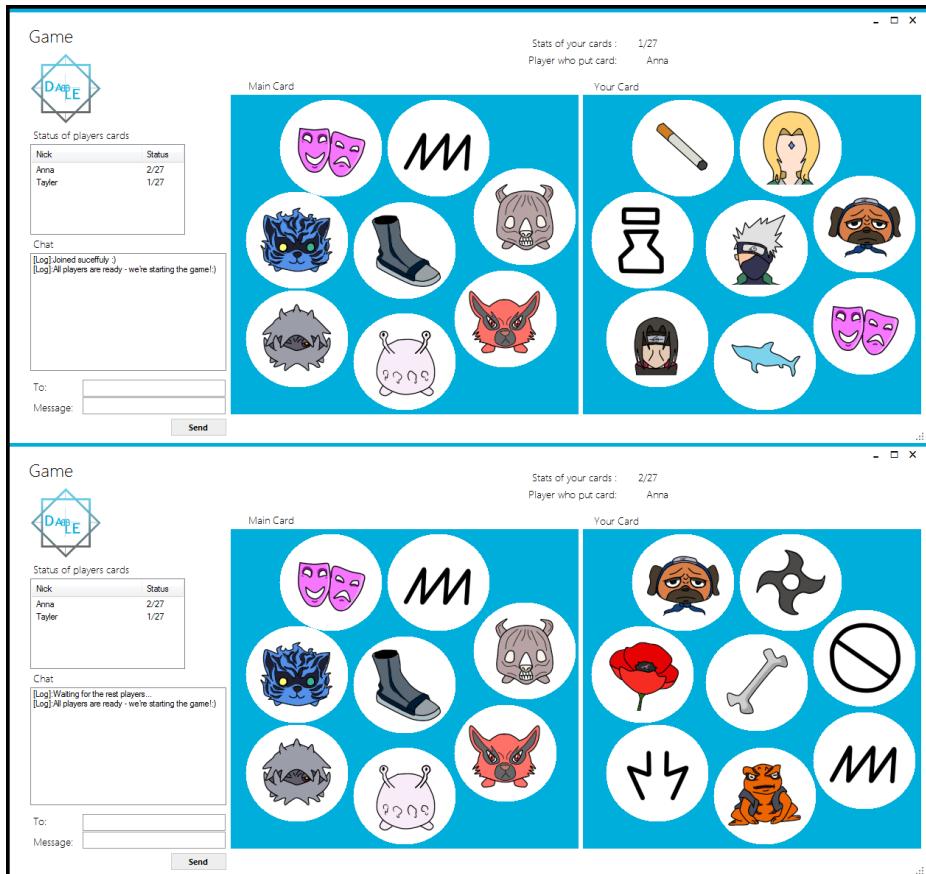


Rys. 5.21. Komunikat o minięciu kary czasowej

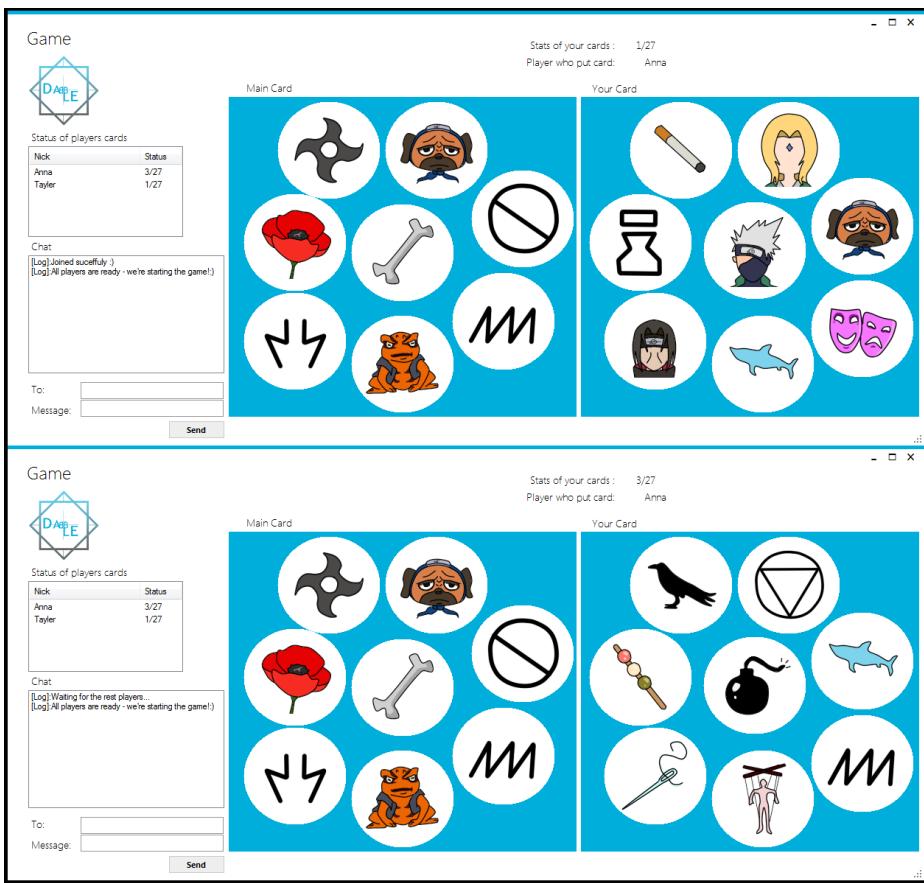
W trakcie projektowania i testowania gry były sprawdzane wszystkie możliwe kombinacje ustawień, których jest 64. Część przebiegu przykładowej rozgrywki prezentują rys. 5.22 – 5.26. Na każdym rysunku znajdują się dwa zrzuty ekranów – na górze pierwszego gracza o nicku Tayler, a na dole drugiego gracza o nicku Anna. Rys 5.22 prezentuje początkowe karty obu graczy. Gracz o nicku Anna wskazał na „Main Card” poprzez kliknięcie lewym przyciskiem myszy obrazek znajdujący się w lewym górnym rogu. Symbol ten znajduje się też na karcie gracza Anna, więc „Main Card” została zamieniona na aktualną kartę gracza Anna, a ten gracz odkrył swoją kolejną kartę. Status wyświetlanych informacji dotyczący liczb kart graczy został zaktualizowany, co pokazuje rys. 5.23. Gracz Anna ponownie zaznaczył poprawnie symbol i karty zostały zaktualizowane (rys. 5.24). Na rys 5.25 to gracz pierwszy (Tayler) wyłożył swoją pierwszą kartę. Status kart i widoki zostały odpowiednio zaktualizowane. Rys 5.26 przedstawia zrzuty ekranu z trwającej rozgrywki, gdzie każdy z graczy wyłożył po kilka kart. Pojawiły się również logi odnośnie banowania widniejące w oknie czatu.



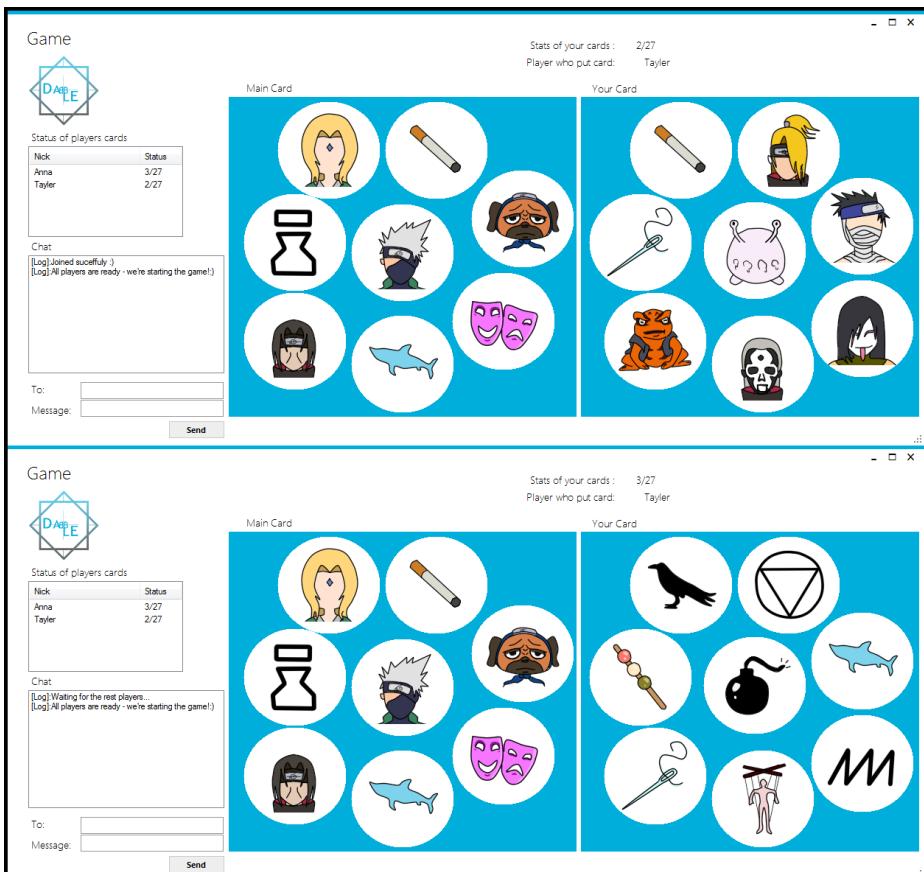
Rys. 5.22. Początek gry



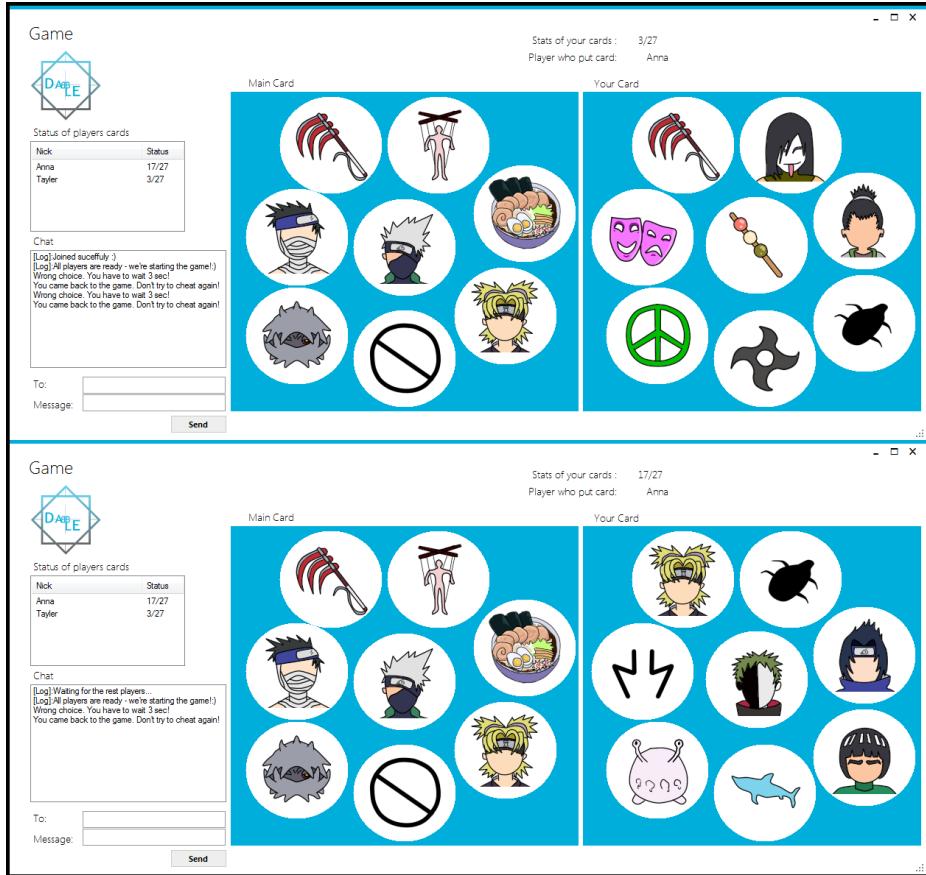
Rys. 5.23 Drugi gracz (Anna) położył kartę



Rys. 5.24 Ponownie drugi gracz (Anna) położył kartę

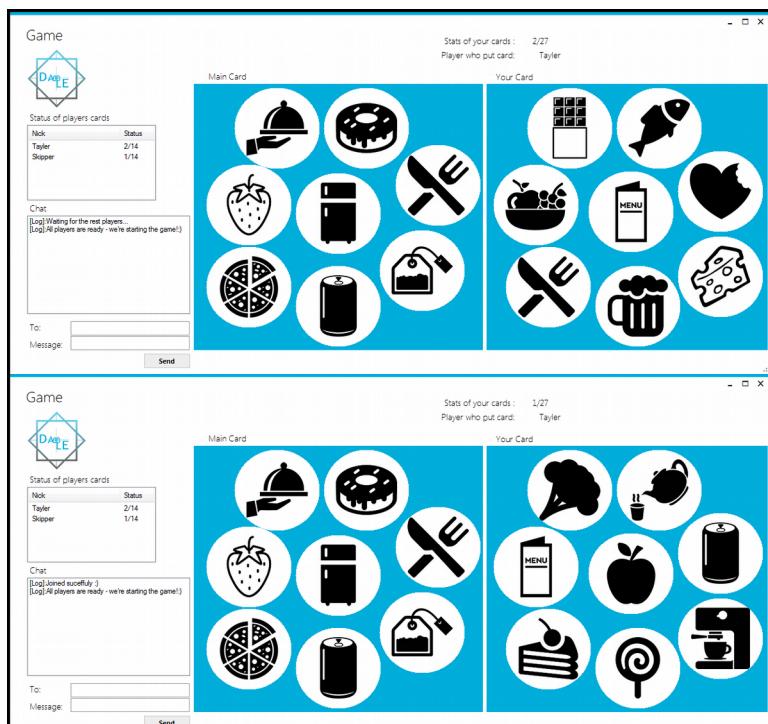


Rys. 5.25. Pierwszy gracz (Tayler) położył kartę

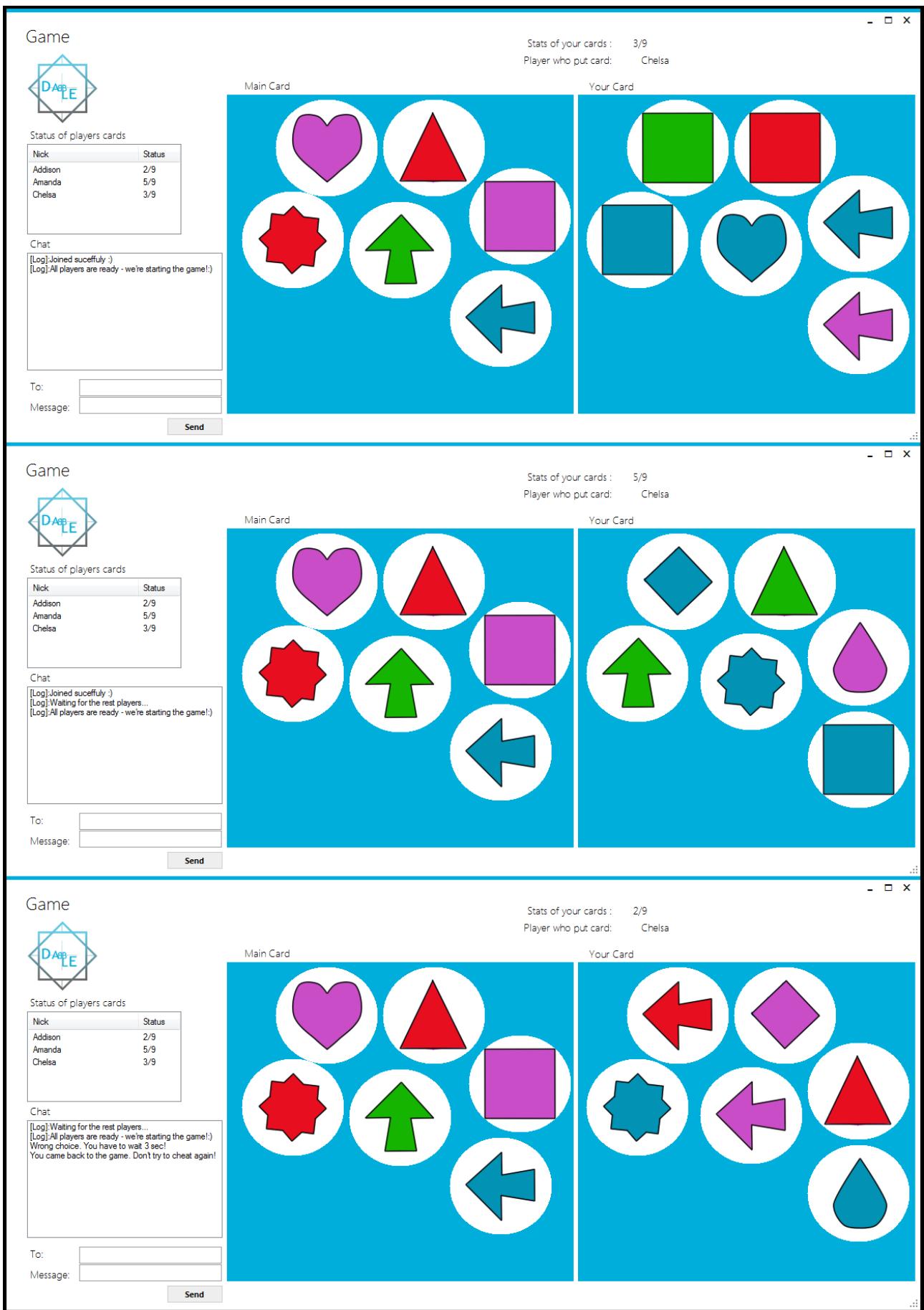


Rys. 5.26. Dalsza część rozgrywki

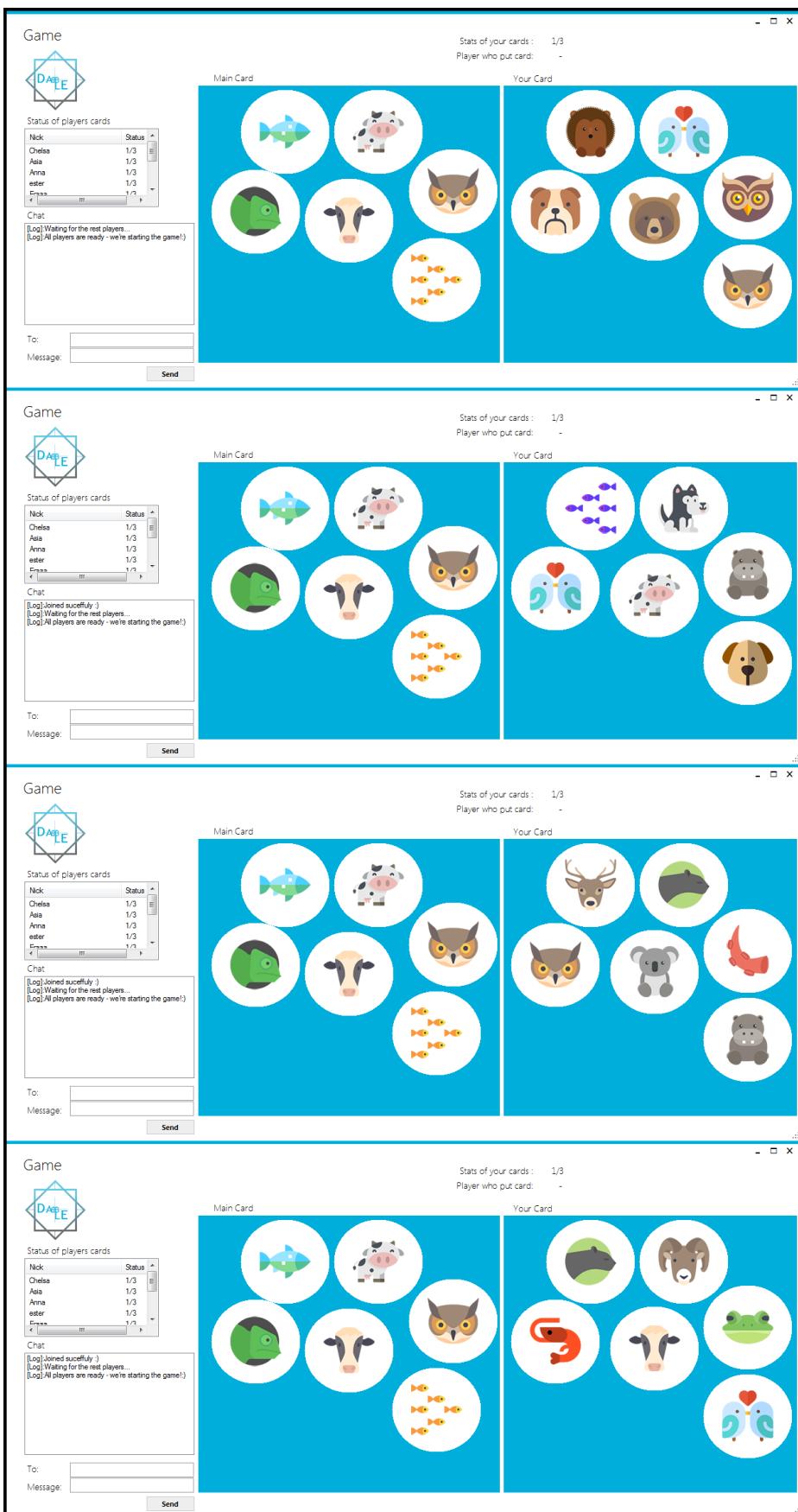
Przykładowe pojedyncze screeny z jednego momentu z innych rozgrywek prezentują rys. 5.27-5.30. Rys 5.31 prezentuje możliwość czatu (gracze nie muszą znajdować się w jednym pokoju).



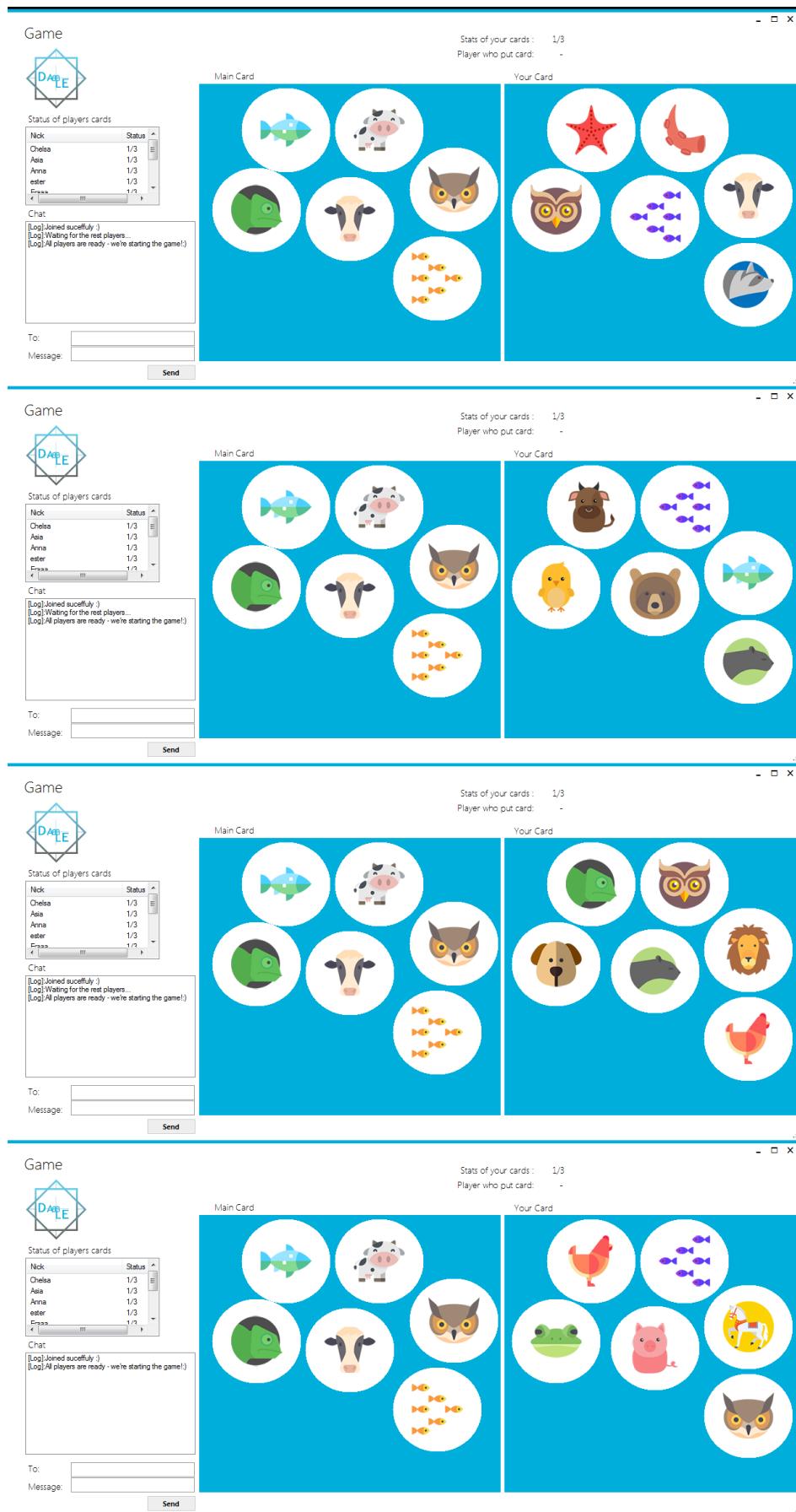
Rys. 5.27. Gra trudniejsza dla 2 graczy z symbolami "food"



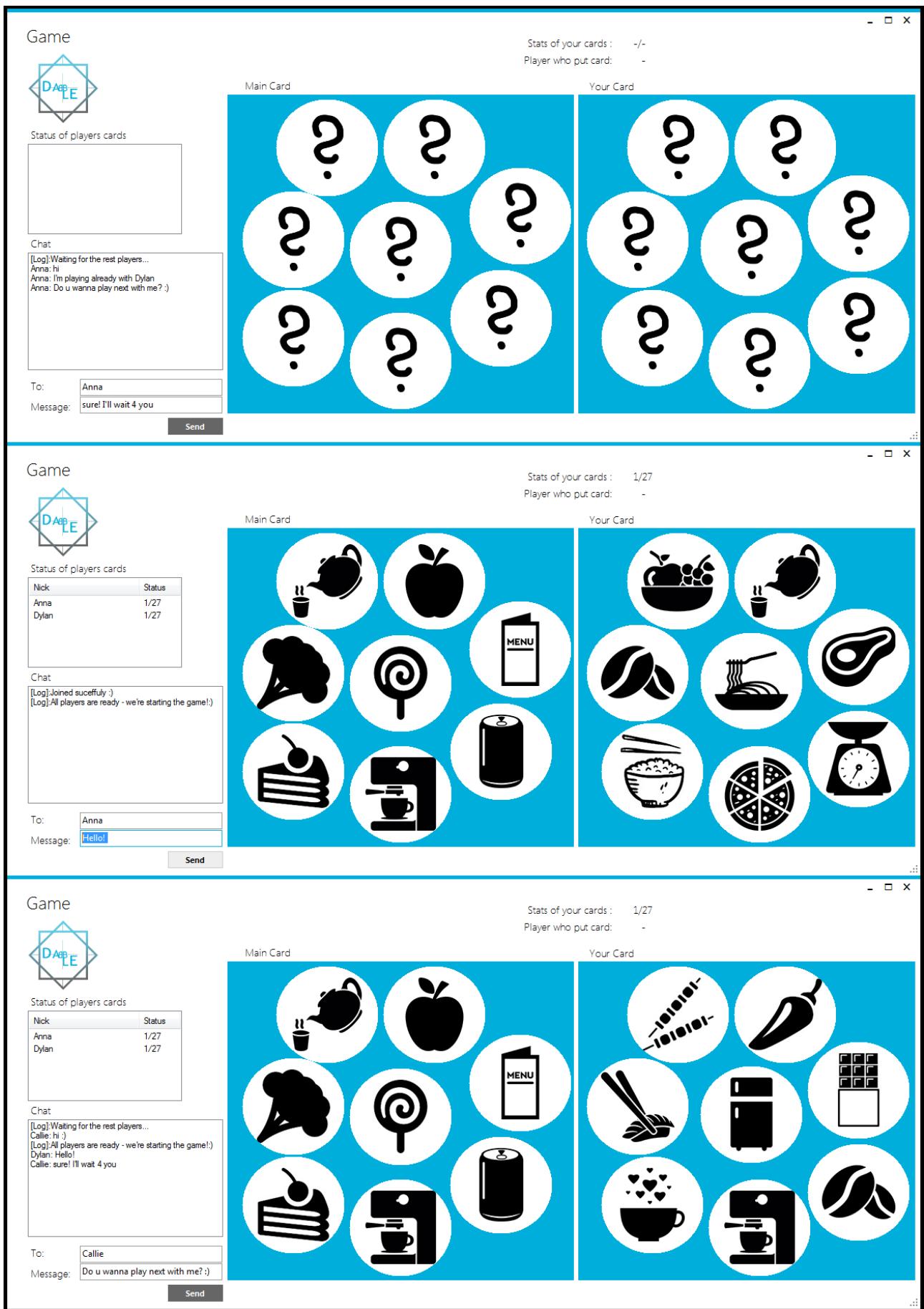
Rys. 5.28. Gra w łatwym trybie dla 3 graczy z symbolami „shapes”



Rys. 5.29. Gra łatwiejsza dla 8 graczy z symbolami "animalas" (cz. 1)



Rys. 5.30. Gra łatwiejsza dla 8 graczy z symbolami "animalas" (cz. 2)



Rys. 5.31. Czat

Przedstawione zrzuty ekranów prezentowały testowaną grę pod różnymi oferowanymi opcjami w różnych konfiguracjach. Rozgrywka na pięciu komputerach przebiegała dla pięciu graczy bezproblemowo. Nie widać było opóźnień, serwer popranie wykonywał swoje zadania. Nie zauważono błędów w synchronizacji. W trakcie testów zauważane błędy były notowane i później poprawiane. Zaliczało się do tego np. możliwość zalogowania się dwukrotnie na jedno konto.

Testy wykazały poprawność zastosowanych algorytmów i mechanizmów. Podczas projektowania korzystano z testów modułowych (np. testowanie wartości brzegowych) i integracyjnych (funkcjonalne, wydajnościowe).

Testy obciążeniowe wykonano przy pomocy dwóch komputerów – na jednym uruchomiony został serwer oraz klient, a na drugim 100 klientów. Z uwagi na brak zewnętrznego IP gra mogła być testowana jedynie w sieci lokalnej, co utrudniło symulację rzeczywistego obciążenia i opóźnień związanych z drogą, jaką pokonują przesyłane pakiety pomiędzy sieciami.

Aplikacja serwera wykorzystuje bardzo małą ilość zasobów systemowych. W zależności od liczby obsługiwanych klientów wynosiła ok. 10 MB pamięci (rys. 5.32).

The screenshot shows the Windows Task Manager window titled 'Menedżer zadań Windows'. The 'Procesy' tab is selected. A table lists processes with their names, owners, memory usage, and descriptions. The 'Pamięć (pry...)' column is highlighted. The TcpServer.exe process is selected, showing 10 528 K of memory usage and the description 'TcpServer'. Other processes listed include explorer.exe, mspaint.exe, csrss.exe, taskmgr.exe, WTouchUser..., CCleaner64.exe, nvxsync.exe, conhost.exe, TabTip.exe, wisptis.exe, Pen_Tablet.exe, nvvsvc.exe, utility.exe, taskhost.exe, Energy Manag..., taskhost.exe, wisptis.exe, PSUAMain.exe, winlogon.exe, dwm.exe, TabTip32.exe..., and InputPersonal... The bottom status bar shows 'Procesy: 81', 'Procesor CPU: 0%', and 'Pamięć fizyczna: 25%'. There is also a checkbox for 'Pokaż procesy wszystkich użytkowników' and a 'Zakończ proces' button.

Nazwa obrazu	Nazwa ...	P...	Pamięć (pry...)	Opis
explorer.exe	Anna	00	49 488 K	Eksplorator Windows
mspaint.exe	Anna	00	28 368 K	Paint
TcpServer.exe...	Anna	00	10 528 K	TcpServer
csrss.exe	SYSTEM	00	7 016 K	Proces wykonawczy klienta/serwera
taskmgr.exe	Anna	00	5 128 K	Menedżer zadań Windows
WTouchUser....	Anna	00	4 068 K	Touch User Mode Driver
CCleaner64.exe	Anna	00	3 308 K	CCleaner
nvxsync.exe	SYSTEM	00	3 124 K	NVIDIA User Experience Driver Compon...
conhost.exe	Anna	00	2 756 K	Host okna konsoli
TabTip.exe	Anna	00	2 628 K	Tablet PC Input Panel Accessory
wisptis.exe	Anna	00	2 492 K	Składnik wprowadzania dotykowego i pr...
Pen_Tablet.exe	SYSTEM	00	2 332 K	Tablet Service for consumer driver
nvvsvc.exe	SYSTEM	00	1 804 K	NVIDIA Driver Helper Service, Version 2...
utility.exe	Anna	00	1 768 K	Lenovo Battery Management Software ...
taskhost.exe	Anna	00	1 708 K	Proces hosta dla zadań systemu Windows
Energy Manag...	Anna	00	1 544 K	Lenovo Energy Management Software 7.0
taskhost.exe	Anna	00	1 528 K	Proces hosta dla zadań systemu Windows
wisptis.exe	SYSTEM	00	1 252 K	Składnik wprowadzania dotykowego i pr...
PSUAMain.exe...	Anna	00	1 200 K	AV Console
winlogon.exe	SYSTEM	00	1 116 K	Aplikacja logowania systemu Windows
dwm.exe	Anna	00	552 K	Menedżer okien pulpitu
TabTip32.exe...	Anna	00	400 K	Tablet PC Input Panel Helper
InputPersonal...	Anna	00	336 K	Serwer personalizacji wprowadzania

Rys. 5.31. Obciążenie

6. ZAKOŃCZENIE

Celem pracy dyplomowej magisterskiej był projekt i implementacja gry sieciowej z wykorzystaniem .NET Framework. Niniejsza praca stanowi opis zagadnień zarówno teoretycznych, jak i praktycznych dotyczących realizacji gry sieciowej. Proces projektowania gry obejmował wiele aspektów. Począwszy od wybrania pomysłu na grę poprzez opracowanie jej zasad, zaprojektowanie GUI, wykonanie potrzebnych grafik, skończywszy na programowaniu i testowaniu, co zajęło najwięcej czasu z uwagi na charakter pracy magisterskiej.

Zastosowany model architektoniczny MVC ułatwia skalowanie i rozbudowę projektu, co jest dużą zaletą. Napisane metody mogą być używane wiele razy przyjmując różne parametry, co pozwala na szybkie wprowadzenie kolejnych analogicznych trybów gry. Opracowany algorytm generowania talii, gdzie znajduje się N kart z K symbolami, z czego każda karta łączy się dokładnie jednym symbolem z pozostałymi kartami, stanowi mocną stronę projektu, ponieważ na nim opiera się cały zamysł rozgrywki. System komunikacji serwera z klientem oparty na protokole TCP również zasługuje na uwagę. Ponadto zostały opracowane pomniejsze systemy odpowiadające za rejestrację i logowanie użytkownika, tworzenie pokoju do gry o wybranych parametrach, komunikację z innymi graczami, przeprowadzanie rozgrywki. Kolejnym ciekawym aspektem gry jest baza danych, która składa się oprócz danych podawanych przy logowaniu – punkty, które gracz otrzymuje po wygraniu rozgrywki.

Wadą jest brak menu „Opcji”, gdzie użytkownik by dostosował rozmiar okna do rozdzielczości własnego monitora, ustawił dźwięk, wybrał preferowany język. Następną słabą stroną jest mała liczba trybów gry (albo 8 symboli, albo 6). Można jednak łatwo dodać kolejne korzystając z już napisanych metod. Aktualnie do wyboru są 4 zestawy obrazków, które również można powiększyć. Źle prezentuje się też moment, w którym gracz łączy się z serwerem (okno wpisywania adresu IP i portu). Powinien być umieszczony w osobnym oknie, a nie łączony z logowaniem/rejestracją.

Patrząc szerzej na projekt widać potencjał w grze, która już teraz jest grywalna i oferuje przeprowadzenie pełnej rozgrywki. W planowanej kontynuacji pierwszym z pomysłów byłoby wprowadzenie dodatkowych opcji, żeby ją urozmaicić grę. Mogą to być koła ratunkowe w postaci usunięcia dwóch niepasujących symboli, czy zablokowanie przeciwnika na 5 sekund. Przeniesienie gry na platformę Android stanowi drugi kierunek rozwoju projektu.

LITERATURA

- [1] Chadwick Jess, Panda Hrusikesh, Synder Todd,: „Programming ASP.NET 4 MVC 4”, O'Reilly, 2012
- [2] Osetek Sylwia, Pytel Krzysztof,: „Systemy operacyjne i sieci komputerowe Część I”, WSiP
- [3] Węgrzyn Agnieszka, Węgrzym Marek: „Aplikacje typu klient-serwer w sieci Internet”, ZielMAN, 1999
- [4] Urbańska-Galanciak Dominika, Zajączkowski Borys,: „Co o współczesnych grach wiedzieć powinniśmy”, SPiDOR, 2009
- [5] Ceneo.pl wraz z partnerami - e-Commerce Polska Izba Gospodarki Elektronicznej oraz TNS: „Raport eZakupy 2015” [online]. <http://dlasklepow.ceneo.pl/wp-content/uploads/2016/10/E-zakupy-2015-raport-Ceneo-i-TNS-Polska-Ranking-Zaufanych-Opinii-2015.pdf>
- [6] Newzoo - TOP 20 CORE PC GAMES | US & EU - <https://newzoo.com/insights/rankings/top-20-core-pc-games/>
- [7] <http://eu.battle.net/hearthstone/pl/>
- [8] <https://www.playgwent.com/pl>
- [9] <https://www.visualstudio.com/>
- [10] <http://www.adobe.com/pl/downloads.html>
- [11] <https://www.systemax.jp/en/sai/>
- [12] <https://www.rebel.pl/product.php/1,1203/20640/Dobble.html>
- [13] <https://stratechery.com>
- [14] <https://code.msdn.microsoft.com/windowsapps/Captcha-for-Windows-c5ea873c>
- [15] <http://www.flaticon.com>
- [16] <https://www.microsoft.com/en-us/download/details.aspx?id=29062>