

PROGRAMOWANIE OBIEKTOWE
Projekt

TEMAT:

System obstawiania zakładów – klient serwer

Spis treści

APLIKACJA W JĘZYKU C#	2
1. ZAŁOŻENIA I OPIS FUNKCJONALNY PROGRAMU	2
1.1. OGÓLNY ZARYS FUNKCJONALNOŚCI PROGRAMU	2
1.2. PROCES LOGOWANIA	2
1.3. ARCHITEKTURA KLIENT-SERWER	2
2. DIAGRAMY UML	3
3. OPIS DOKUMENTACJI PROJEKTOWEJ ZAWARTEJ W DOŁĄCZONYM PLIKU	3
4. OPIS UŻYTKOWY PROGRAMU	3
4.1. PROCES INSTALACJI	3
4.2. PORUSZANIE SIĘ PO PROGRAMIE	4
4.2.1 MENU	4
4.2.2 ZASADY GRY	4
4.2.3 OKNO LOGOWANIA	4
4.2.3.1 PROCES REJESTRACJI UŻYTKOWNIKA	5
4.2.3.2 PROCES LOGOWANIA UŻYTKOWNIKA	6
4.2.4 OKNO GRY ADMINISTRATORA	9
4.2.5 OKNO GRY KLIENTA	11
5. SCHEMAT BLOKOWY	12
APLIKACJA W JĘZYKU C++	12
1. ZAŁOŻENIA I OPIS FUNKCJONALNY PROGRAMU	12
1.1 OGÓLNY ZARYS FUNKCJONALNOŚCI PROGRAMU	12
1.2 PROCES DODAWANIA I USUWANIA UŻYTKOWNIKÓW	12
2. DIAGRAMY UML	13
3. OPIS DOKUMENTACJI PROJEKTOWEJ ZAWARTEJ W DOŁĄCZONYM PLIKU	13
4. OPIS UŻYTKOWY PROGRAMU	13
4.1. PROCES INSTALACJI	13
4.2. PORUSZANIE SIĘ PO PROGRAMIE	14
4.2.1 MENU	14
4.2.2 ZASADY GRY	14
4.2.3 OKNO GRY	15
4.2.4 PROCES DODAWANIA I USUWANIA UŻYTKOWNIKÓW	15
4.2.4.1 DODAWANIE UŻYTKOWNIKÓW DO LISTY GRACZY	15
4.2.4.2 USUWANIE UŻYTKOWNIKÓW Z LISTY GRACZY	18
5. SCHEMAT BLOKOWY	20
6. WNIOSKI	21

Aplikacja w języku C#

1. Założenia i opis funkcjonalny programu

1.1. Ogólny zarys funkcjonalności programu

Ogólnym założeniem projektu jest stworzenie aplikacji symulującej zakład bukmacherski. Jednym z celów aplikacji jest zapoznanie użytkownika z podstawowymi procesami i operacjami, które zachodzą podczas obstawiania zakładów. W aplikacji zostały stworzone dwa rodzaje kont – administratora oraz zwykłego gracza. Konto administratora jest rozszerzeniem zwykłego konta i pozwala na rozpoczynanie wyścigu oraz wgląd do historii połączeń wszystkich klientów. Po rejestracji każdy z użytkowników może zalogować się na swoje konto, dodać pieniądze oraz postawić dwa rodzaje zakładów. Dodatkiem jest chat dzięki któremu każdy z klientów może się komunikować z wszystkimi innymi graczami. W przypadku wygranej każdy z graczy, który dobrze wytypował zwycięzcę dostaje adekwatną sumę pieniędzy. Istotnym elementem w programie jest losowość, ponieważ każdy z obiektów reprezentowanych jako psy biorące udział w wyścigu mają taką samą szansę na wygraną, a kolejne wyścigi są od siebie niezależne. Aplikacja umożliwia ciągłą interakcję z użytkownikiem, poprzez specyficzny zakład stawiany w trakcie gry którego specyfikacja opisana jest w zasadach gry do których gracze mają ciągły dostęp za pomocą menu.

1.2. Proces logowania

W aplikacji zaimplementowany został pełny system logowania wraz z uwierzytelnianiem danych. Dane użytkowników trzymane są w bazie Microsoft SQL Server. Przy starcie programu każdy z użytkowników może stworzyć nowe konto pod warunkiem, że dana nazwa użytkownika nie jest już zajęta lub zalogować się na już istniejące konto. W bazie przetrzymywane są: nazwa użytkownika, hasło oraz jego obecny stan konta. Login i hasło pozwalają na szybko walidację poprawności danych podczas logowania lub rejestracji. Dzięki takiemu rozwiązaniu dane nie są tracone po zamknięciu aplikacji i administrator ma ciągły dostęp i możliwość kontroli danych każdego z użytkowników za pomocą programu Microsoft SQL Server Management studio.

1.3. Architektura klient-serwer

Jednym z celów projektu jest zaimplementowane architektury klient-serwer, która pozwoli na asynchroniczny kontakt klientów z serwerem. Dzięki takiej architekturze aplikacja w której zalogowane na swoje konto jest administrator służy jako serwer na którym wykonywane są wszystkie operacje związane z zarządzaniem klientami. Dzięki takiemu rozwiązaniu istnieje wiele aplikacji, które podłączone są do jednego serwera który nieustannie komunikuje się z pozostałymi klientami w celu wymiany danych. Dzięki asynchronicznej komunikacji wiele klientów może stawiać zakłady jednocześnie oraz komunikować się między sobą za pomocą zaimplementowanego chatu. Całość została wykonana przy użyciu framework'a SignalR, który jest częścią platformy ASP.NET stworzonej przez Microsoft. Oprogramowanie pozwala na prosty proces zarządzania protokołami HTTP oraz pozwala na łatwą interakcję pomiędzy serwerem a klientem za pomocą Hubów – specjalnych klas zdefiniowanych w SignalR, które w znacznym stopniu uproszczają proces komunikacji za pomocą websocketów. W Hubach zdefiniowane zostały metody które odpowiedzialne są za wymienianie danych

między serwerem a klientami. Hub umożliwia również bezpośrednie wysyłanie danych z serwera do wszystkich podłączonych klientów. Po zalogowaniu się na konto administratora, zostaje uruchomiony serwer, do którego zostaje podłączona każda kolejna aplikacja kliencka. Tylko konto administratora ma dostęp do rozpoczęcia wysięgu i przekazuje tę informację do pozostałych klientów. Wtedy rozpoczyna się proces przesyłania danych do klientów w celu aktualizacji ich form. Administrator posiada również specjalne okno w którym wyświetlane są wszystkie informacje dotyczące obecnych połączeń klientów.

2. Diagramy UML

Diagram przypadków użycia, który w uproszczeniu obrazuje główne operacje, które dostępne są dla poszczególnych użytkowników, załączony jest w osobnym pliku o nazwie: Use case diagram-BettingParlorC#.

Diagram klas również załączony jest w osobnym pliku o nazwie: ClassDiagram-BettingParlorC#, dołączonym do dokumentacji. Obrazuje on podstawowe zależności między klasami oraz ich strukturę. Część metod i klas została pominięta aby uprościć cały diagram i skupić się na przedstawieniu logiki kryjącej się za całą aplikacją.

3. Opis dokumentacji projektowej zawartej w dołączonym pliku

Wszystkie klasy wraz z ich metodami i polami są bezpośrednio udokumentowane w plikach źródłowych za pomocą komentarzy wykorzystujących język XML. Z wszystkich komentarzy, które bezpośrednio odwołują się do kodu źródłowego wygenerowano dokumentację w postaci strony, która znacząco ułatwia pracę z dokumentacją. Całość została wykonana przy użyciu programu o nazwie Sand castle help file builder. Każda klasa zawiera krótki opis zadania które realizuje. Każde z pól także posiada krótki komentarz przechowywanych wartości oraz sposobu ich wykorzystania. Wszystkie metody są udokumentowane komentarzami opisującymi realizowane zadania oraz listę parametrów. Plik zawierający stronę jest dołączony w osobnym pliku o nazwie: Help. Aby całość poprawnie działała plik Help musi mieć odnośnik do folderu zawierającego wszystkie wygenerowane strony i skrypty przez Sand castle. Znajdują się one w folderze Documentation. Użycie komentarzy z wykorzystaniem języka XML pozwala na wykorzystanie potężnego narzędzia zaimplementowanego w Visual Studio jakim jest IntelliSense. W całym projekcie za każdym razem gdy odwołamy się do danej klasy lub metody od razu otrzymamy krótki opis zadania jakie realizuje dana klasa lub metoda. IntelliSense pozwala także na szybsze pisanie kodu z wykorzystaniem poprzednio udokumentowanych klas, ponieważ podpowiada nam krótkie opisy funkcji oraz listę parametrów, które mogą być automatycznie uzupełnianie.

4. Opis użytkowy programu

4.1. Proces instalacji

Aplikacja została wykonana przy użyciu technologii WinForms stworzoną przez Microsoft. Do aplikacji został opracowany instalator, który gwarantuje zainstalowanie wszystkich potrzebnych komponentów do sprawnego działania aplikacji. Docelowym środowiskiem w którym ma działać aplikacja jest jeden komputer na którym loguje się administrator który zarządza serwerem hostowanym na VPS wraz z bazą SQL natomiast reszta klientów podłącza się do serwera.

4.2. Poruszanie się po programie

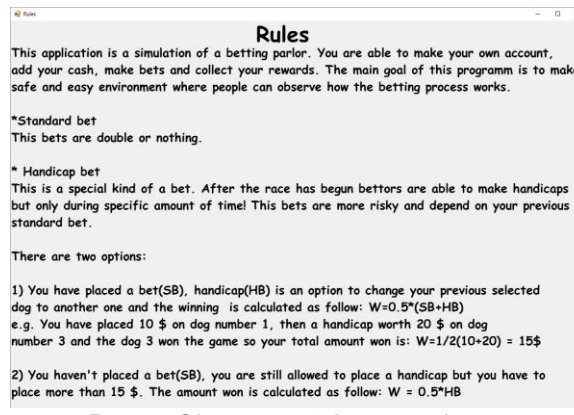
4.2.1 Menu



Rys. 1 - Wygląd menu.

Pierwsze okno które pokazuje się użytkownikowi po starcie aplikacji. Użytkownik ma do wyboru 3 przyciski: Rules, Play, Exit. Przycisk odpowiedzialny za zasady gry ukrywa obecne okno i wyświetla okno z zasadami gry. Przycisk Exit zamyka całą aplikację, natomiast ostatni z przycisków nawiguje do następnego okna w którym odbywa się proces logowania lub rejestracji użytkownika.

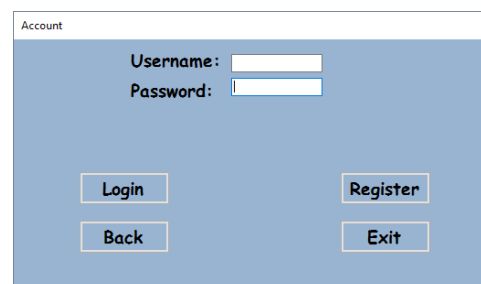
4.2.2 Zasady gry



Rys. 2 - Okno prezentujące zasady gry.

Okno zawiera jedno duże pole tekstowe w którym umieszczone zostały zasady obowiązujące w rozgrywce. Po zamknięciu okna ponownie pokazywane jest okno zawierające menu.

4.2.3 Okno logowania



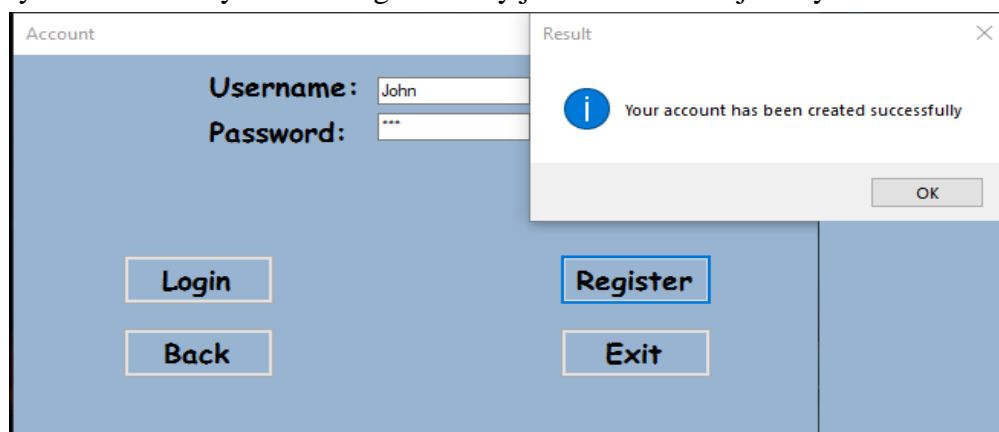
Rys. 3 - Okno służące do logowania i rejestracji użytkownika.

W tym oknie odbywa się proces pobrania danych od użytkownika, które następnie zostaną przekazane do obiektu klasy Login przetrzymywanego w tej formie gdzie nastąpi proces walidacji danych. Przycisk Exit zamyka całą aplikację natomiast przycisk Back zamyka obecne okno i ponownie pokazuje okno zawierające menu. Pozostałe przyciski są wykorzystane do odpowiednio logowania i rejestrowania użytkowników. W zależności od podanych danych, obecnego statusu serwera lub stanu gry próba logowania może zostać odrzucona. Poniżej opisane zostały wszystkie możliwe przypadki w których proces logowania lub rejestracji zakończył się sukcesem lub porażką.

4.2.3.1 Proces rejestracji użytkownika

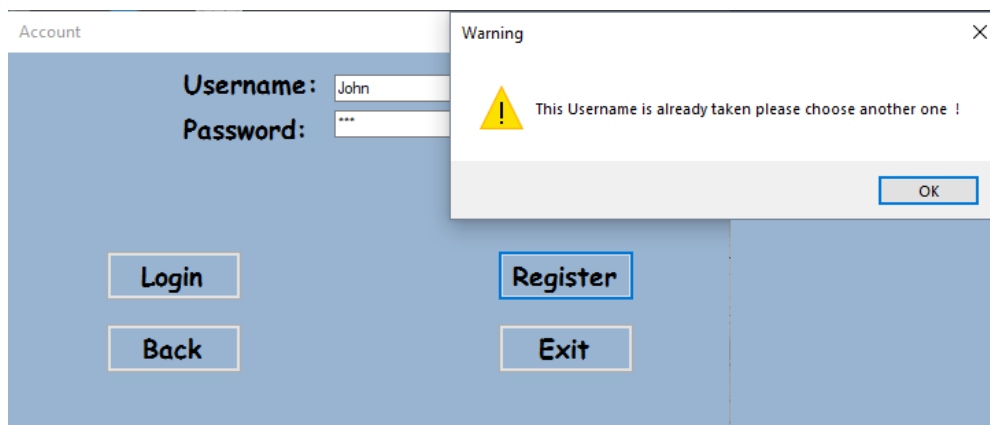
Każdy login jest unikalny i nie może się powtarzać, ponieważ jest wykorzystywany do identyfikacji danego gracza. Cały proces walidacji sprawdzający czy dane podane przez użytkownika są unikalne odbywa się w obiekcie klasy Login za pomocą metody RegisterNewPlayer. Proces pobierania danych z bazy i tworzenia nowych rekordów odbywa się za pomocą technologii LINQ TO SQL. Na początku baza jest przeszukiwana w celu odnalezienia gracza o danym loginie, jeśli taki rekord nie istnieje tworzony jest nowy wpis z loginem obecnie rejestrującego się gracza. Natomiast jeśli użytkownik o danym loginie zostaje odnaleziony, proces jest przerywany i gracz otrzymuje informację, że dany login jest już zajęty.

- a) Użytkownik tworzy konto o loginie który jeszcze nie istnieje w systemie.



Rys. 4 - Proces rejestracji zakończony sukcesem.

- b) Użytkownik próbuje stworzyć konto o loginie który już istnieje.

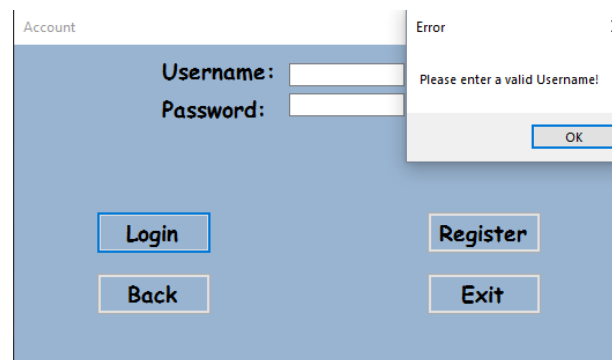


Rys. 5 - Proces rejestracji zakończony niepowodzeniem.

4.2.3.2 Proces logowania użytkownika.

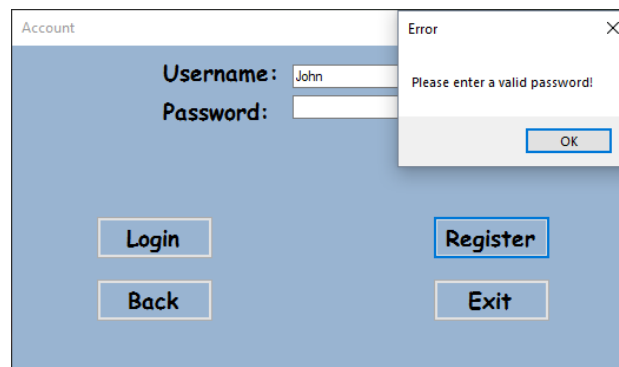
Proces logowania użytkownika jest bardziej skomplikowany od rejestracji, ponieważ należy uwzględnić więcej przypadków. Żaden z klientów nie może zalogować się na swoje konto gdy serwer nie jest hostowany przez administratora oraz podczas wyścigu, ponieważ gracze którzy zalogują się po rozpoczęciu gry mogliby stawiać zakłady, krótko przed zakończeniem co spowodowałoby stu procentowe szanse na wygraną. Klient również nie jest w stanie zalogować się na swoje konto gdy jest już zalogowany na innej aplikacji. Proces logowania administratora może zostać przerwany tylko gdy próbuje on uruchomić serwer na adresie na którym hostowany jest już inny serwer. Wszystkie te możliwości zostały przewidziane i reakcje na daną sytuację zostały odpowiednio zaimplementowane. Proces sprawdzenia poprawności danych z rekordami trzymanymi w bazie został zaimplementowany w klasie Login natomiast sprawdzenie stanu wyścigu oraz statusu serwera odbywa się w klasie Client i Server.

a) Użytkownik próbuje zalogować się bez podania danych.



Rys. 6 - Próba logowania bez wprowadzenia danych.

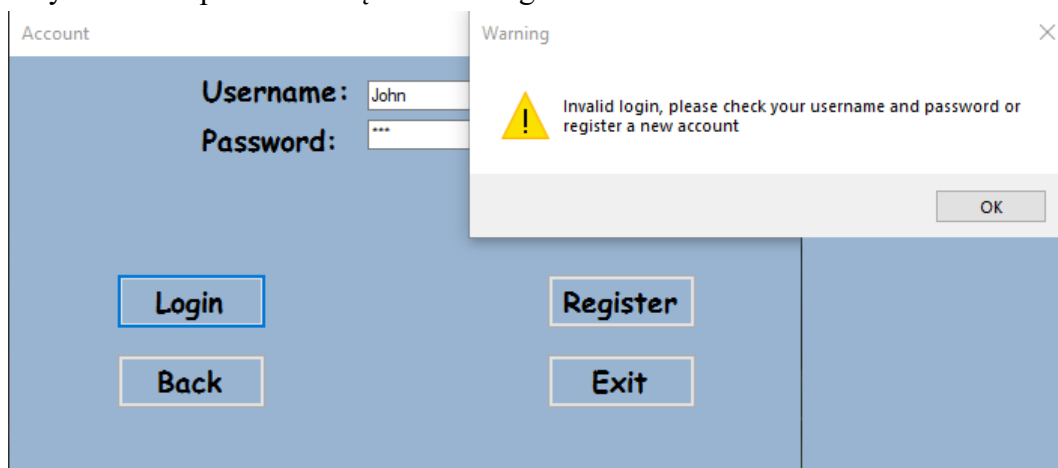
b) Użytkownik próbuje zalogować się bez podania hasła.



Rys. 7 - Próba logowania bez wprowadzenia hasła.

W sytuacjach przytoczonych w podpunktach a i b walidacja odbywa się po stronie okna dialogowego. Taka implementacja pozwala na wykluczenie sytuacji w której sprawdzamy poprawność danych w bazie z pustymi napisami. Dzięki walidacji w oknie odciażamy liczbę połączeń wykonywanych do bazy danych.

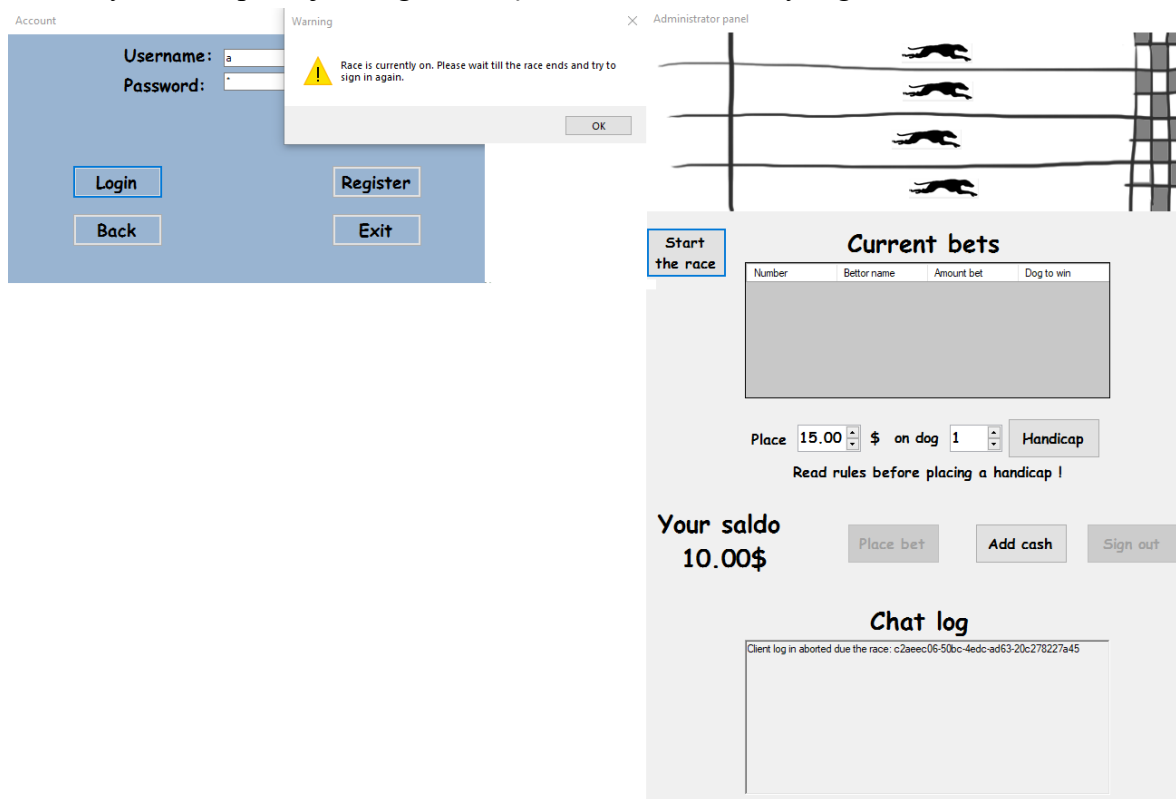
c) Użytkownik wprowadza błędne dane logowania.



Rys. 8 - Proces logowania przerwany z powodu wprowadzenia błędnych danych.

Po wprowadzeniu błędnych danych lub próbie logowania bez wcześniejszego zarejestrowania wyświetlany jest powyższy komunikat.

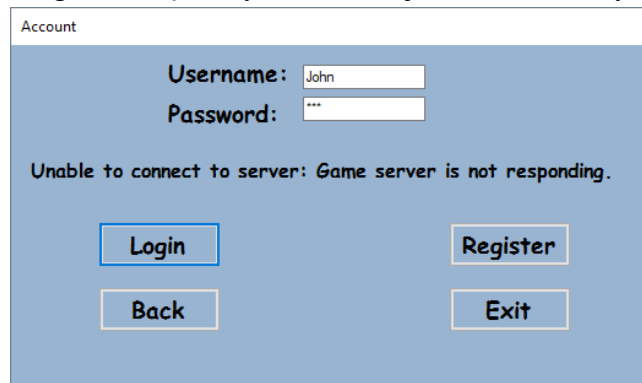
d) Użytkownik próbuje zalogować się w trakcie trwania wyścigu.



Rys. 9 - Proces logowania klienta zostaje przerwany z powodu obecnie trwającego wyścigu.

Po wprowadzeniu poprawnych danych następuje sprawdzenie statusu serwera oraz stanu wyścigu. W tym przypadku jak widzimy na załączonym rysunku wyścig już się rozpoczął więc klient nie może zalogować się na swoje konto o czym informuje go powyższy komunikat. Administrator otrzymuje informację o zablockowanej próbie logowania w swoim oknie.

- e) Klient próbuje zalogować się kiedy serwer nie jest uruchomiony.

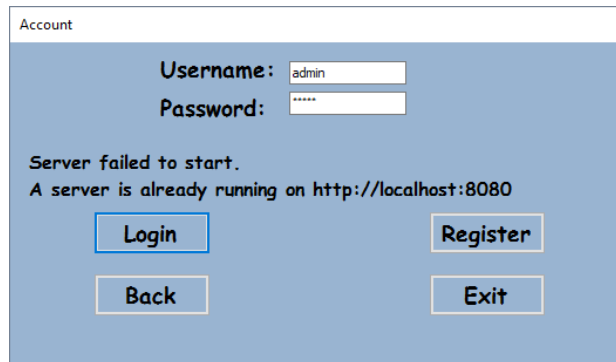


The screenshot shows a window titled "Account" with a blue background. It contains two input fields: "Username:" with the text "John" and "Password:" with three asterisks. Below the fields is a message: "Unable to connect to server: Game server is not responding." At the bottom, there are four buttons: "Login", "Register", "Back", and "Exit".

Rys. 10 - Proces logowania zostaje przerwany z powodu braku odpowiedzi serwera.

Po wprowadzeniu poprawnych danych następuje sprawdzenie statusu serwera. Jeżeli administrator nie zalogował się na swoje konto, serwer nie został uruchomiony więc klienci nie mogą się z nim połączyć.

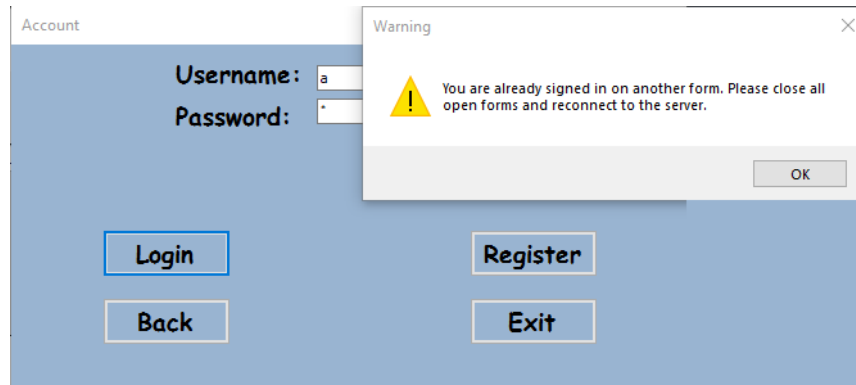
- f) Administrator próbuje zalogować się na swoje konto i tym samym uruchomić serwer na adresie na którym hostowany jest już inny serwer.



The screenshot shows a window titled "Account" with a blue background. It contains two input fields: "Username:" with the text "admin" and "Password:" with six asterisks. Below the fields is a message: "Server failed to start. A server is already running on http://localhost:8080". At the bottom, there are four buttons: "Login", "Register", "Back", and "Exit".

Rys. 11 - Proces logowania zostaje przerwany z powodu próby uruchomienia serwera na zajętych adresie.

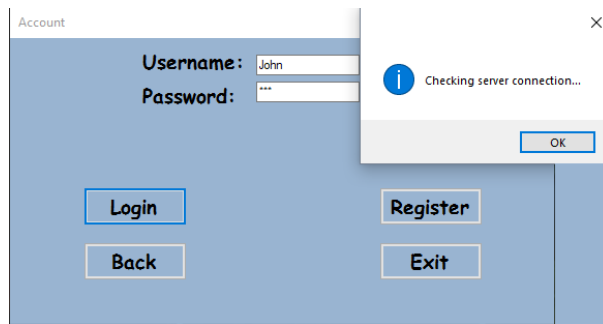
- g) Klient próbuje zalogować się na swoje konto będą zalogowanym już na innej aplikacji.



The screenshot shows the "Account" login window with a blue background. A "Warning" dialog box is overlaid on top of it. The dialog box has a yellow warning icon and the text: "You are already signed in on another form. Please close all open forms and reconnect to the server." with an "OK" button. The "Account" window behind it shows the "Username:" field with the letter "a" and the "Password:" field with one asterisk. The "Login", "Register", "Back", and "Exit" buttons are visible at the bottom of the "Account" window.

Rys. 12 - Proces logowania przerwany z powodu zalogowania na innej aplikacji.

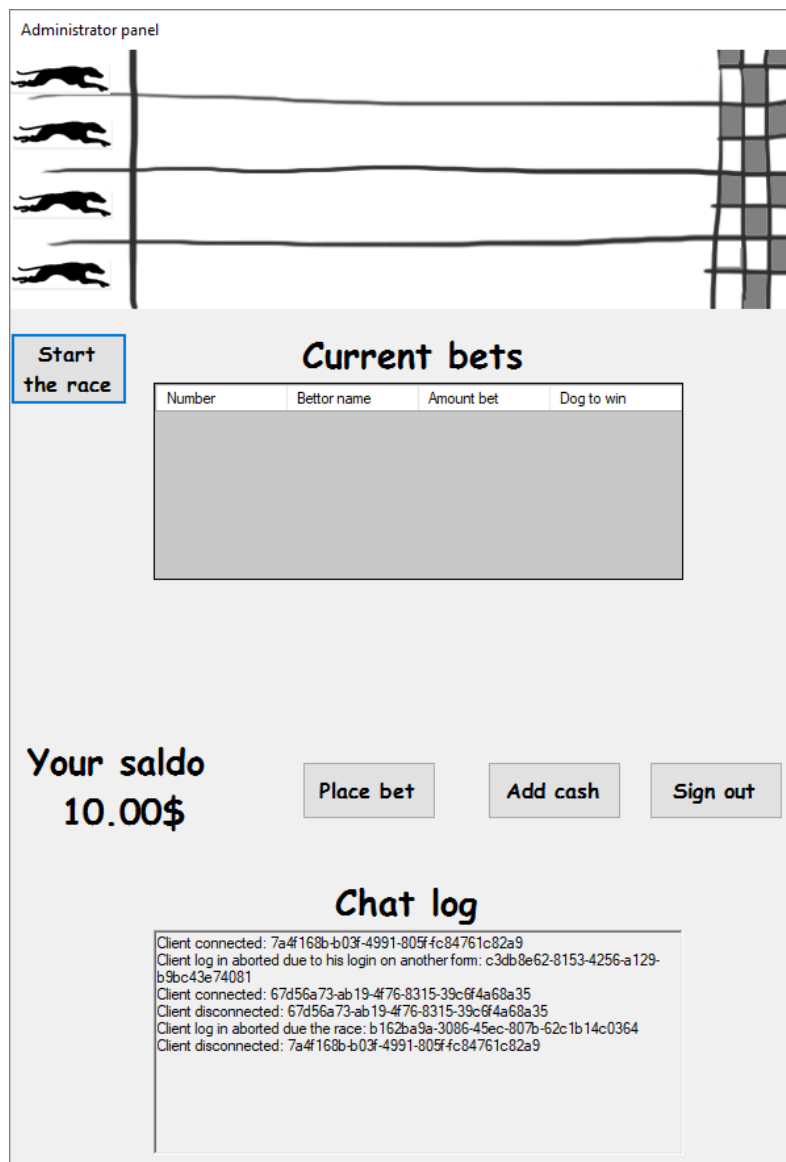
a) Użytkownik jest w stanie połączyć się z serwerem.



Rys. 13 - Użytkownik wprowadził poprawne dane.

Po wprowadzeniu poprawnych danych następuje sprawdzenie obecnego statusu serwera i stanu gry. Po tej walidacji zostaje otwarte okno gdzie klient może stawiać zakłady i brać udział w grze.

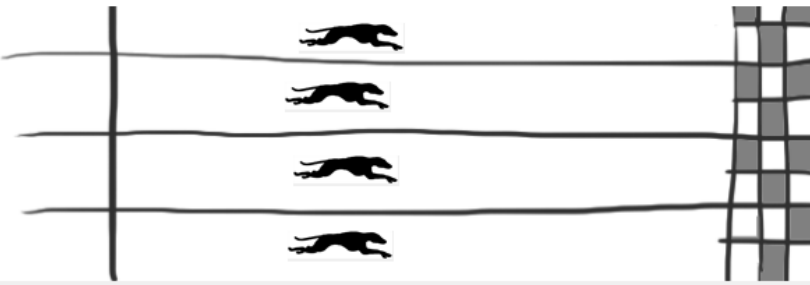
4.2.4 Okno gry administratora



Rys. 14 - Panel administratora przed wyścigiem.

Administrator może rozpoczynać wyścig, przelewać pieniądze na swoje konto, stawiać dwa rodzaje zakładów oraz przeglądać historie połączeń klientów. W konsoli zaprezentowano wszystkie możliwe komunikaty.

Administrator panel



Start the race

Current bets

Number	Bettor name	Amount bet	Dog to win
1	b	10.0000	1
2	d	10.0000	2
3	c	10.0000	3
4	a	10.0000	4
5	admin	10.0000	4

Place \$ on dog **Handicap**

Read rules before placing a handicap !

Your saldo
0.00\$

Place bet **Add cash** **Sign out**

Chat log

```
Client connected: e6ad0097-48c5-479e-9f55-195362cc8f92
Client connected: 38668de0-ad75-4ff1-a8ab-e05fa5238460
Client connected: 7b4d5ea0-af7a-4807-97c3-293b87e75450
Client connected: cb469e68-93c1-40ad-a396-170ac2bd76c5
```

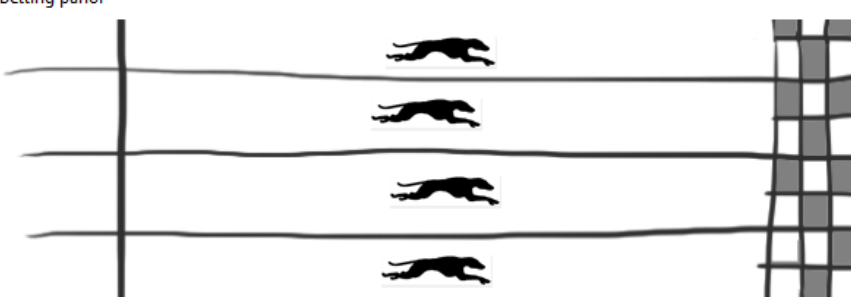
Rys. 15 - Panel administratora podczas gry.

Administrator po rozpoczęciu wyścigu sam może brać udział w grze poprzez obstawianie wybranego psa. Dodatkowo może sprawdzać stan połączeń klientów, śledząc ich łącza. Proces poruszania psów w oknie administratora jest losowy. Każdy z wyścigów jest od siebie niezależny co gwarantuje sportową rywalizację. Administrator musi być świadomy faktu, że jego wylogowanie oznacza zamknięcie uruchomionego serwera. Gdy wylosowuje się z swojego konta musi mieć pewność, że żaden z klientów nie jest obecnie połączony.

4.2.5 Okno gry klienta

Klienci mogą przelewać pieniądze na swoje konto oraz stawiać dwa rodzaje zakładów – standardowe i handicap. Opisy tych zakładów zawarte są w zasadach gry. Każdy z podłączonych klientów ma również dostęp do czatu, za pomocą którego może komunikować się z wszystkimi innymi graczami. Istotny jest tutaj proces poruszania psów. W przeciwieństwie do serwera nie jest on losowy, ponieważ na każdej z form musi wygrać ten sam pies. Z każdym posunięciem psa w oknie administratora, lokalizację psów są przesyłane do klientów i tam odpowiednio aktualizowane.

Betting parlor



Current bets

Number	Bettor name	Amount bet	Dog to win
1	admin	10.0000	1
2	a	10.0000	2
3	c	10.0000	3
4	b	10.0000	2

Place \$ on dog **Handicap**

Read rules before placing a handicap !

Your saldo
30.00\$

Please wait till the race ends to sign out.

Chat

a: hey !
c: What's up ?
b: Doing fine !
a: My dog will win fo sure !
c: No way i will win it all !
b: Winning !

Rys. 16 - Okno klienta w trakcie wyścigu.

Przyciski Place bet został zablokowany, ponieważ klient postawił już jeden zakład standardowy przed rozpoczęciem gry. Przycisk Sign out jest zablokowany aby zapobiec zniszczeniu obiektu klienta na serwerze razem z jego obecnym zakładem. Wylogowanie jest równoznaczne z odłączeniem klienta od serwera i tym samym zniszczenia jego obiektu zawierającego dane użytkownika. Oczywiście przed zniszczeniem dane są zapisywane

do bazy danych lecz informacja o obecnym zakładzie jest tracona i po zakończeniu administrator nie jest w stanie wypłacić klientowi który wylogowałby się w trakcie wyścigu jego należności. Panel służący do stawiania specjalnych zakładów podczas gry – handicapów jest dezaktywowany krótko przed końcem wyścigu aby nie dopuścić do sytuacji w której każdy z graczy czeka do momentu w którym psy są tuż przed metą i w tedy stawia swój zakład. Zwiększyłyby to znacznie szanse na wygraną wszystkich graczy i nie byłoby to zgodne z zasadami.

5. Schemat blokowy

Schemat prezentuje proces obstawiania danego zakładu przez użytkownika. Na diagramie dobrze uwidoczona jest logika aplikacji. Przed analizą diagramu warto zapoznać się z zasadami gry, gdzie widnieje zasada świadcząca o tym, że dozwolony jest tylko jeden zakład standardowy natomiast bet typu handicap zawsze zależy od zakładu standardowego co łatwo zauważyć na diagramie zaprezentowanym poniżej. W szczególnym przypadku gdy handicap jest jedynym zakładem wartość przekazywana do fabryki równa jest zero. Ze względu na duży rozmiar schemat znajduje się w załączonym pliku o nazwie BettingProcess-Code schema.pdf.

Aplikacja w języku C++

1. Założenia i opis funkcjonalny programu

1.1 Ogólny zarys funkcjonalności programu

Program jest uproszczoną wersją aplikacji wykonanej w języku C#. Ogólnym założeniem projektu jest stworzenie aplikacji symulującej zakład bukmacherski. Jednym z celów aplikacji jest zapoznanie użytkownika z podstawowymi procesami i operacjami, które zachodzą podczas obstawiania zakładów. W wyścigu może brać udział jednocześnie tylko czterech graczy. Za pomocą panelu umieszczonego w głównym oknie gracz może dodać siebie do listy bądź usunąć się z listy użytkowników. W odróżnieniu od aplikacji wykonanej w C# dane nie są zapisywane w osobnej bazie przez co są tracone po zamknięciu aplikacji. W programie obowiązują takie same zasady jak w aplikacji wykonanej w języku C#. Docelowym środowiskiem w którym działa aplikacja jest jeden komputer na którym na zmianę czterech graczy obstawia swoje zakłady i gdy każdy z nich potwierdzi gotowość uruchamiają oni wyścig. Nie zostało tutaj zaimplementowane konto administratora, który ma większe możliwości niż zwykli gracze. Każdy z graczy jest równoważny i wspólnie podejmują oni decyzję o starcie wyścigu.

1.2 Proces dodawania i usuwania użytkowników

Wszystkie dane użytkowników są przetrzymywane bezpośrednio w aplikacji. W rozgrywce może brać udział maksymalnie czterech graczy. Każdy z nich po dodaniu się do listy użytkowników ma swoją kontrolkę za pomocą której jest w stanie dodać pieniądze do swojego konta oraz postawić dwa rodzaje zakładów. W przerwie pomiędzy wyścigami każdy z graczy może usunąć się z listy tym samym zwalniając miejsce dla innego gracza.

2. Diagramy UML

Diagram przypadków użycia, który w uproszczeniu obrazuje główne operacje, które dostępne są dla użytkowników, załączony jest w osobnym pliku o nazwie: Use case diagram-BettingParlorC++.

Diagram klas został opracowany przy użyciu narzędzia Visual Paradigm i znajduje się w pliku o nazwie: ClassDiagram-BettingParlorC++. Obrazuje on uproszczony schemat komunikacji pomiędzy klasami. Część metod i klas odpowiedzialnych za szatę graficzną aplikacji została pominięta aby uprościć schemat diagramu i skupić się na logice aplikacji.

3. Opis dokumentacji projektowej zawartej w dołączonym pliku

Wszystkie klasy wraz z plikami nagłówkowymi zostały skomentowane bezpośrednio w kodzie źródłowym aplikacji. Przy pomocy generatora dokumentacji – Doxygen wszystkie komentarze zostały wyeksportowane i przedstawione w postaci strony, która ułatwia pracę z dokumentacją. Doxygen pozwala na łatwe i szybkie zarządzanie dokumentacją a cały proces generowania dokumentacji jest zautomatyzowany. Każda klasa zawiera krótki opis zadania które realizuje. Każde z pól także posiada krótki komentarz przechowywanych wartości oraz sposobu ich wykorzystania. Wszystkie metody są udokumentowane komentarzami opisującymi realizowane zadania oraz listę parametrów. Plik zawierający stronę jest dołączony w osobnym pliku o nazwie: annotated. Wygenerowana dokumentacji pozwala na szybką i przyjemną pracę z kodem źródłowym, ponieważ przy każdym komentarzu do danej funkcji lub pola mamy bezpośrednio odnośnik do linii kodu gdzie dana funkcja została zaimplementowana.

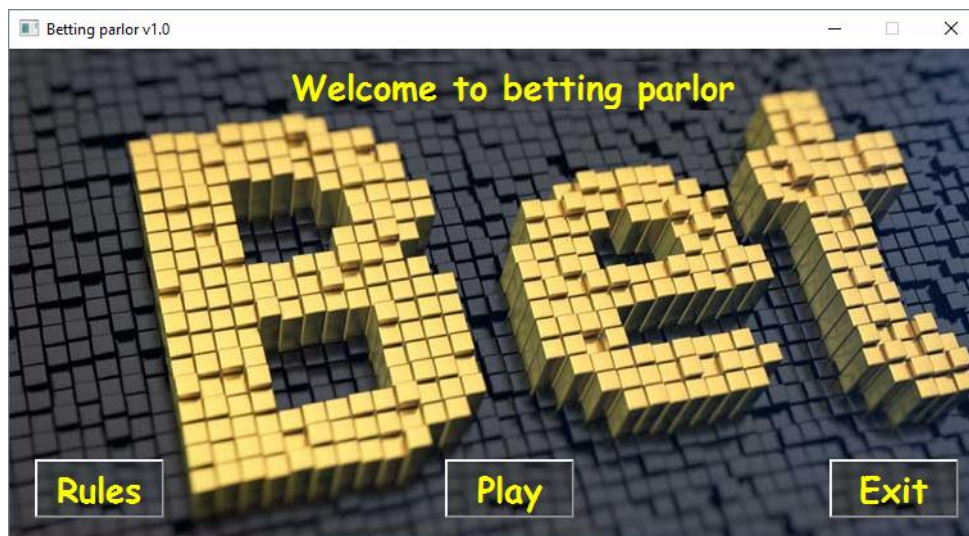
4. Opis użytkowy programu

4.1. Proces instalacji

Aplikacja została wykonana za pomocą framework'a Qt. Qt umożliwiło stworzenie aplikacji cross – platformowej dzięki czemu oprogramowanie działa sprawnie zarówno na systemie linux jak i windows. Systemy na których przetestowano sprawność aplikacji to Windows 10 (64 bit) oraz Ubuntu 16.04 LTS. Do aplikacji nie opracowano instalatora, ponieważ program nie wymaga żadnych zależności które muszą być zainstalowane zanim aplikacja będzie sprawna do działania na danym systemie. Jedynym wymogiem jest obecność bibliotek wykorzystywanych przez Qt framework w miejscu gdzie znajduje się plik .exe uruchamiający program. Plik o rozszerzeniu .exe znajdujący się w folderze aplikacji działa poprawnie tylko na systemie windows. Natomiast po przeniesieniu projektu na drugi system i kompilacji program działa równie sprawnie.

4.2. Poruszanie się po programie

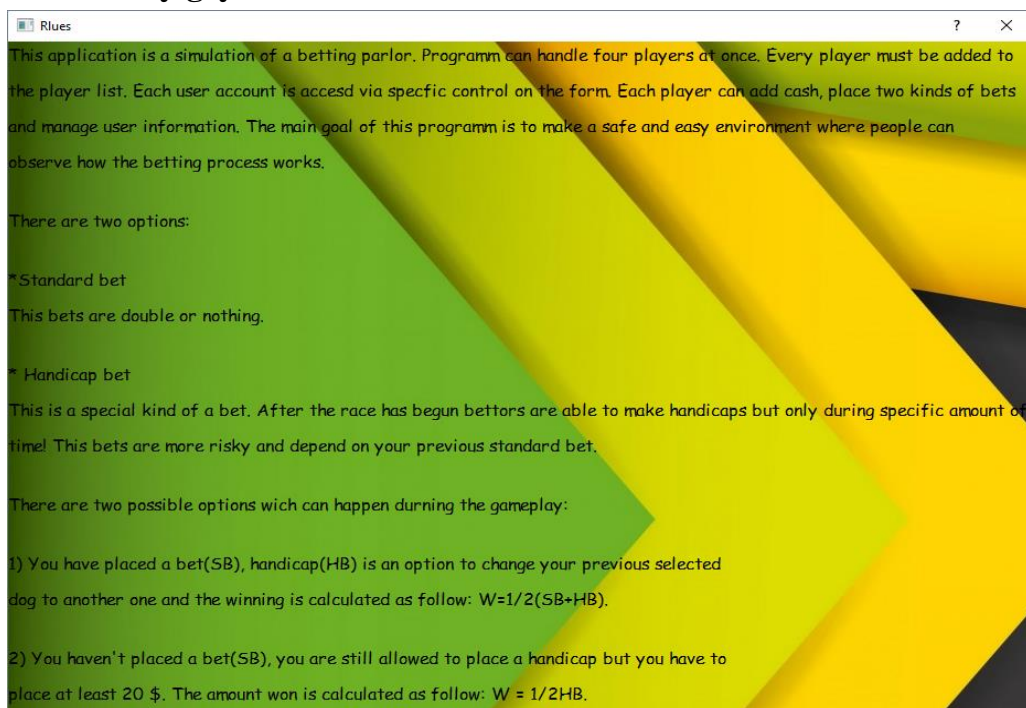
4.2.1 Menu



Rys. 17 - Wygląd menu.

Okno jest głównym punktem odniesienia dla całego programu. Użytkownik ma możliwość zobaczenia zasad, bezpośredniego zamknięcia aplikacji oraz rozpoczęcia nowej gry.

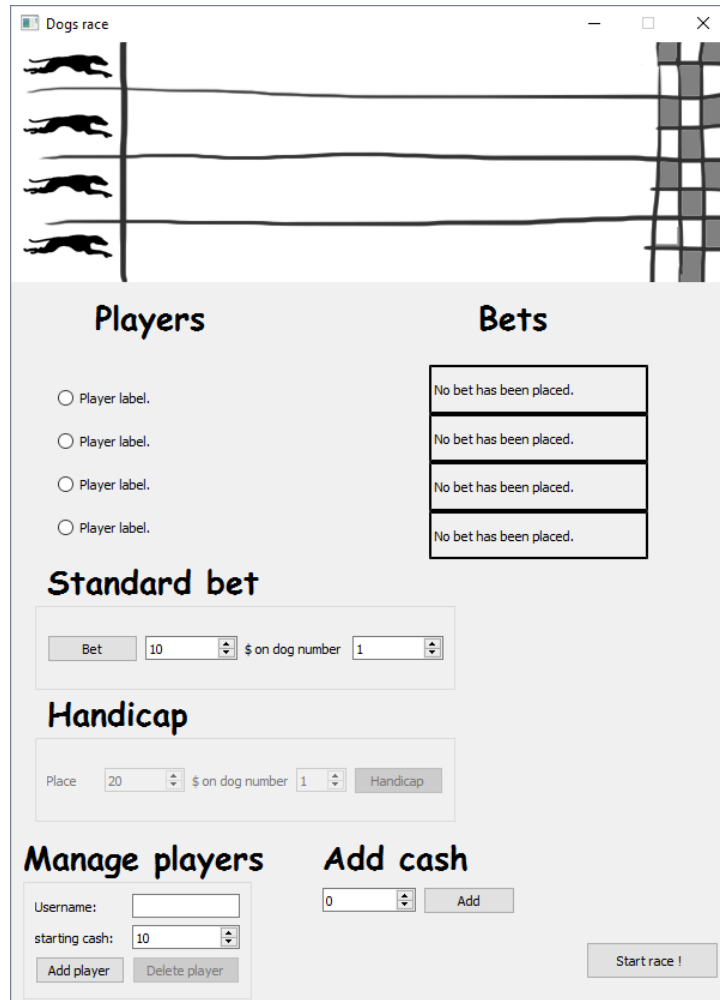
4.2.2 Zasady gry



Rys. 18 - Zasady gry.

System zakładów jest analogiczny do tego zaimplementowanego w aplikacji w języku C#. Cała rozgrywka odbywa się na tej samej zasadzie.

4.2.3 Okno gry



Rys. 19 - Okno gry przed startem wyścigu.

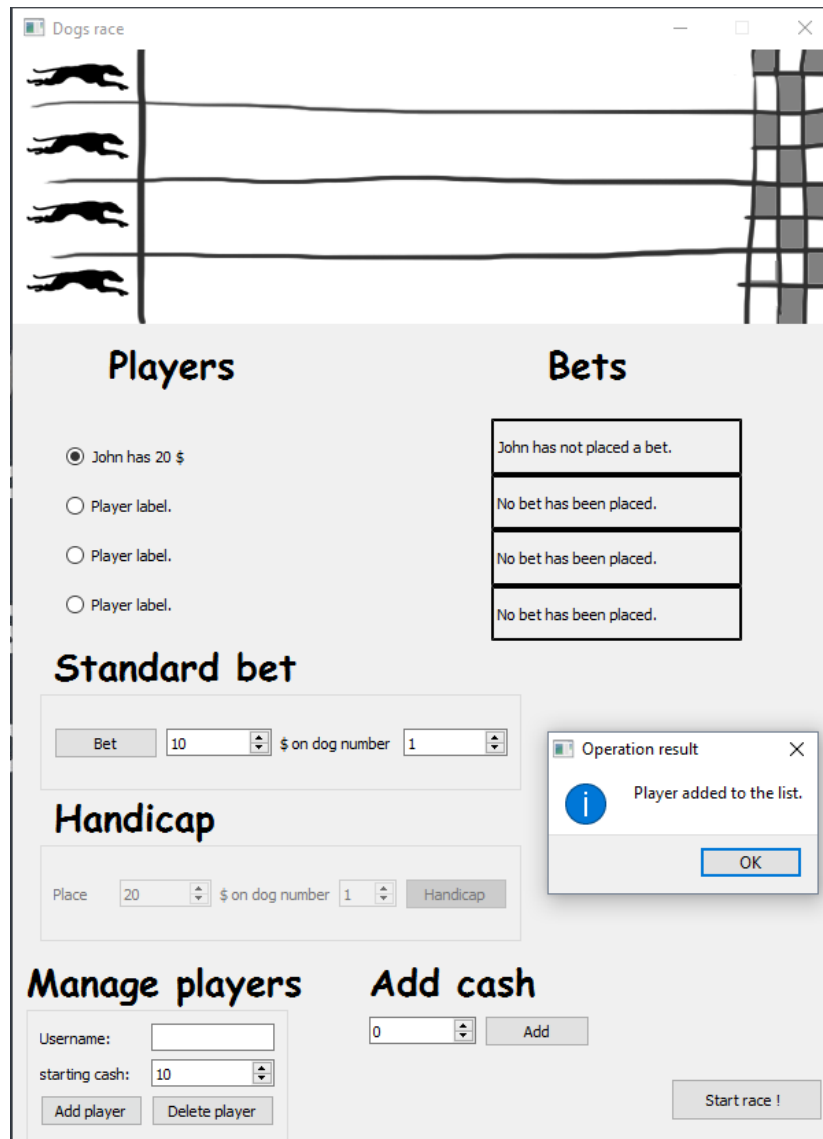
Cztery kontrole znajdujące się po lewej stronie (radio buttons) odpowiadają za każdego z graczy. W ten sposób aplikacja wie który z graczy obecnie chce postawić swój zakład. Panel na dole aplikacji (Manage players) umożliwia dodawanie oraz usuwanie graczy z listy obecnych użytkowników. Panele Standard bet i Handicap bet odpowiedzialne są za stawianie zakładów przez wybranego gracza. Natomiast w tabelce znajdującej się po prawej stronie na górze znajduje się lista obecnych zakładów. Każdy wiersz tabelki odpowiada jednemu z graczy.

4.2.4 Proces dodawania i usuwania użytkowników

4.2.4.1 Dodawanie użytkowników do listy graczy

W wyścigu może brać udział maksymalnie czterech graczy. Wszyscy gracze reprezentowani są przez obiekty klasy Bettor. Wszystkie te obiekty przetrzymywane są w wektorze wskaźników na te klasy – playerList. Za pomocą panelu Manage players obiekty są dodawane lub usuwane z listy w zależności od wykonywanej operacji.

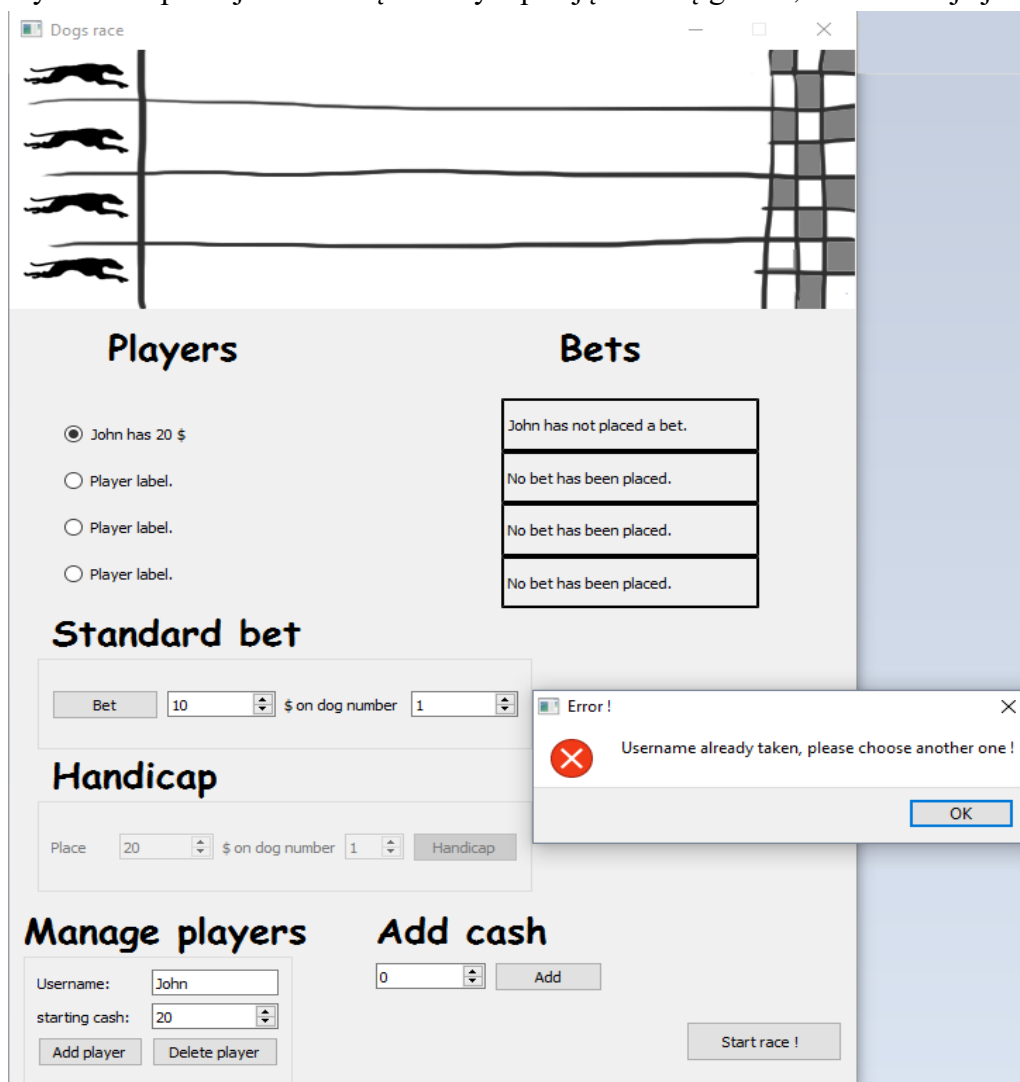
- a) Użytkownik dodaje swoje konto do listy obecnych graczy.



Rys. 20 - Operacja dodania użytkownika zakończona sukcesem.

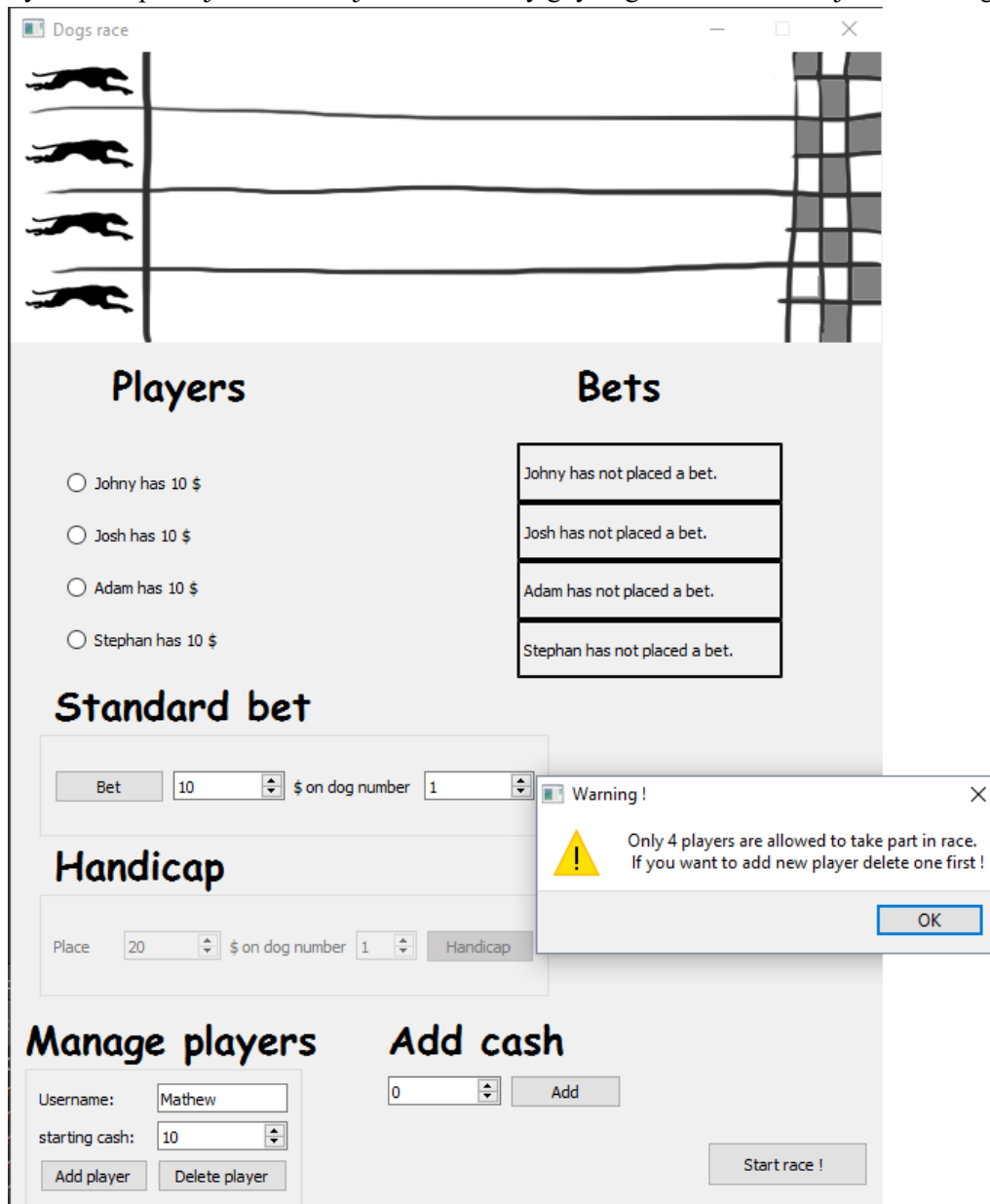
Na załączonym obrazku widzimy, komunikat świadczący o pozytywnym zakończeniu operacji oraz zmianę kontrolki w panelu Players, która pokazuje dodanego gracza.

b) Użytkownik próbuje dodać się do listy wpisując nazwę gracza, która istnieje już na liście.



Rys. 21 - Operacja dodania użytkownika przerwana z powodu zajętej nazwy gracza.

- c) Użytkownik próbuje dodać swoje konto do listy gdy w grze bierze udział już czterech graczy



Rys. 22 - Proces dodawania użytkownika przerwany z powodu zbyt dużej liczby graczy.

Gdy w grze bierze udział już czterech graczy i następny użytkownik chce dołączyć do wyścigu bez usunięcia jednego z obecnych graczy wyświetlany jest komunikat pokazany na powyższym obrazku.

4.2.4.2 Usuwanie użytkowników z listy graczy

Po osiągnięciu limitu wynoszącego czterech graczy, uruchomiona zostaje opcja usuwania graczy. Po wpisaniu poprawnego imienia gracza w polu Username możemy skasować wybranego użytkownika. Usunięcie obiektu gracza odbywa się za pomocą wywołania destruktoru klasy Bettor, która zapewnia poprawne zwolnienie zaalokowanej pamięci. Następnie wskaźnik do danego obiektu usuwany jest z wektora playerList i zwalniane jest miejsce dla następnego użytkownika. Poniżej zaprezentowano wszystkie możliwe przypadki w których proces usuwania gracza może zakończyć się pomyślnie lub niepowodzeniem.

a) Użytkownik został poprawnie usunięty z listy

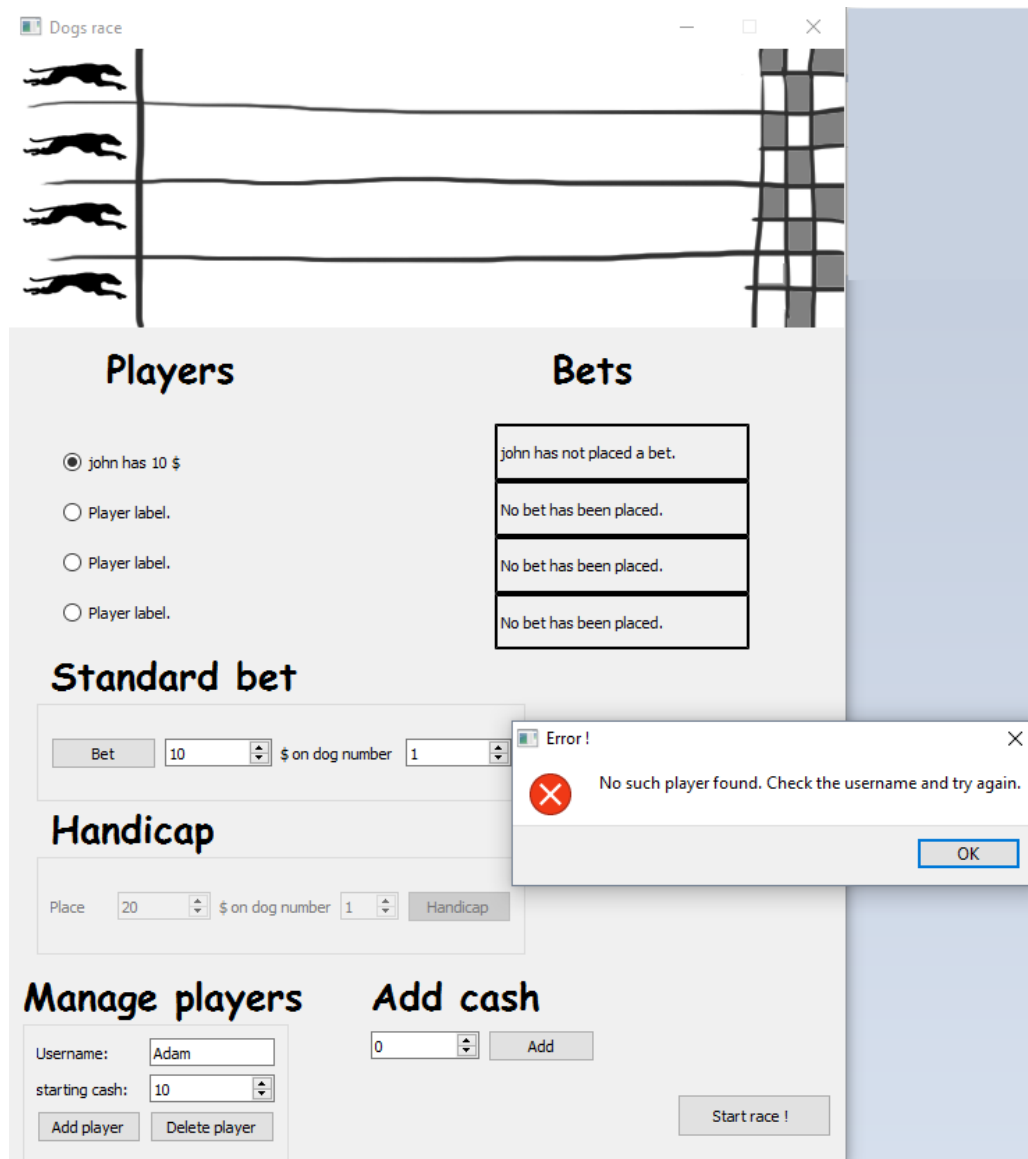
The screenshot displays a software application titled "Dogs race". At the top, there is a graphic of a race track with four dogs running from left to right towards a checkered finish line. Below the track, the interface is divided into several sections:

- Players:** Contains four radio buttons, each followed by the text "Player label."
- Bets:** A vertical stack of four rectangular boxes, each containing the text "No bet has been placed."
- Standard bet:** Includes a "Bet" button, a numeric input field with the value "10", the text "\$ on dog number", and another numeric input field with the value "1".
- Handicap:** Includes a "Place" label, a numeric input field with the value "20", the text "\$ on dog number", a numeric input field with the value "1", and a "Handicap" button.
- Manage players:** Features a "Username:" label, an empty text input field, a "starting cash:" label, a numeric input field with the value "10", and two buttons: "Add player" and "Delete player".
- Add cash:** Features a numeric input field with the value "0" and an "Add" button.
- Start race !:** A button located at the bottom right of the main interface.

An "Operation result" dialog box is overlaid on the right side of the application. It has a title bar with a close button (X). The dialog contains an information icon (i) and the text "Player removed from the list." Below this text is an "OK" button.

Rys. 23 - Użytkownik został poprawnie usunięty z listy.

- b) Użytkownik próbuje usunąć gracza o nazwie użytkownika która nie występuje w programie.



Rys. 24 - Proces usuwania użytkownika został przerwany, ponieważ gracz o danej nazwie nie został odnaleziony.

5. Schemat blokowy

Schemat blokowy ilustruje proces dodawania użytkownika do listy graczy. Diagram w prosty sposób obrazuje algorytm, który po kolei sprawdza warunki i w zależności od spełnienia danego kryterium kontynuuje lub od razu przerywa cały proces i wyświetla odpowiedni komunikat. Schemat ze względu na swoje rozmiaru i lepszą czytelność został załączony w oddzielnym pliku o nazwie: AddingUserToListProcess-Code schema.

6. Wnioski

Wszystkie główne cele aplikacji zostały zrealizowane zgodnie z założeniami projektowymi, które zostały określone przed rozpoczęciem prac nad projektem. Technologie w których zostały wykonane obie aplikacje dobrze pokazują różnice pomiędzy dwoma językami w których zostały napisane. Aplikacje znacząco różnią się zastosowaną architekturą. Program napisany w języku C++ z wykorzystaniem framework'a Qt jest aplikacją która pozwala na symulację wyścigu psów wraz z udziałem czterech graczy. Gracze nie mogą jednocześnie zarządzać swoimi kontami, ponieważ cały proces odbywa się na jednej maszynie. Natomiast w aplikacji stworzonej w technologii Winforms zaimplementowano architekturę klient – serwer przy użyciu framework'a SignalR, który znacznie ułatwił proces komunikacji pomiędzy klientem a serwerem. Aplikacja pozwala na połączenie wielu klientów z jednym serwerem, dzięki czemu liczba graczy zależy tylko wyłącznie od specyfikacji serwera. Każdy klient z zainstalowaną aplikacją i dostępem do Internetu może korzystać z aplikacji na własnej maszynie. Struktura kodu w języku C++ znacząco różni się od tej w C#, ponieważ widzimy tam jednoznaczny podział kodu na pliki nagłówkowe z rozszerzeniem .hpp oraz .cpp. Praca z frameworkiem Qt jest bardzo intuicyjna a dokumentacja znacząco ułatwia oprogramowywanie poszczególnych funkcji. W obu aplikacjach zastosowane zostały główne zasady programowania obiektowego między innymi: dziedziczenie, polimorfizm, abstrakcja, hermetyzacja oraz niektóre z zasad SOLID. Kod był tworzony zgodnie z zasadami clean-code. Wszystkie zmienne i funkcje są intuicyjnie nazwane dzięki czemu można łatwo wywnioskować realizowane przez nie zadania. W aplikacji napisanej w języku C# zaimplementowano również wzorzec fabryki, który odseparował proces tworzenia obiektu od reszty kodu. Jedną z głównych różnic, która uwydatniła się podczas tworzenia aplikacji w języku C++ był brak garbage collector'a, który odpowiedzialny jest za automatyczne zwalnianie pamięci zaalokowanej przez programistę. Z tego powodu należało zaimplementować wirtualne destruktory oraz z każdą alokacją pamięci należało pamiętać o poprawnym usunięciu danego bloku. Aplikacja napisana w języku C# pokazuje jak bardzo przydatne są frameworki w pracy programisty. SignalR znacznie uprościł cały system komunikacji pomiędzy serwerem a klientami. Za pomocą zdefiniowanych funkcji można w łatwy sposób połączyć z sobą wiele niezależnych aplikacji. W aplikacji napisanej w WinForms zastosowano również technologię LINQ TO SQL, która znacznie przyspieszyła proces pisania kodu odpowiedzialnego za połączenia z bazą danych w której przetrzymywane są dane użytkowników. Formuły Language Integrated Query (LINQ) zostały zaimplementowane przez Microsoft w środowisku .NET. Pozwalają na łatwą pracę z obiektami, dokumentami oraz plikami. LINQ TO SQL automatycznie mapuje wybrane tabele zawarte w bazie danych i tworzy z nich klasy, które mogą być później wykorzystane do pobierania i wysyłania danych. Aplikacja posiada również interfejsy, które pozwalają na większy poziom abstrakcji. Wykorzystano również obsługę wyjątków, która ma miejsce podczas połączeń klientów z serwerem. Zaimplementowano również wątki, które umożliwiły wykonywanie operacji w tle dzięki czemu okna nie blokowały się podczas wykonywania funkcji. Projekt w języku C# tworzony był zgodnie z konwencją TDD – test driven development. Najpierw tworzono ogólny zarys klasy po czym pisano testy jednostkowe do niektórych metod gdzie kod odpowiadający za logikę był dość skomplikowany a następnie tworzono już kod źródłowy który przechodził przez dane testy pomyślnie. Całość została wykonana przy pomocy podstawowego pakietu służącego do pisania testów jednostkowych, który zaimplementowany został w Visual Studio.

Reasumując cały projekt, można zauważyć, że aplikacja napisana w języku C# oferuje więcej możliwości i wykorzystuje większy potencjał tego języka. Moim zdaniem język ten oferuje większe możliwości ale pozbawia programistę pełnej kontroli nad pisanym oprogramowaniem. Język C# oferuje także szerszy wachlarz rozwiązań i technologii, które znacznie ułatwiają pracę nad daną aplikacją. Uważam, że odseparowanie definicji funkcji oraz

zmiennych od kod kodu źródłowego jest złym rozwiązaniem, ponieważ programista musi podczas pracy nieustannie zmieniać pliki. Największą wadą i jednocześnie zaletą języka C++ jest brak garbage collector. Programista sam musi zatroszczyć się o zaalokowaną pamięć. Dzięki temu osoba tworząca dane oprogramowanie posiada pełną kontrolę nad tym co dzieje się w programie ale jednocześnie musi pamiętać o każdym zaalokowanym bloku. Własna alokacja i zwalnianie pamięci przekłada się również na szybkość działania aplikacji. O ile w tak małych projektach jak te realizowane na zajęciach z programowania obiektowego nie ma to większego znaczenia, tak przy dużych aplikacjach gdzie szybkość przetwarzania danych jest istotna może mieć to kluczowy wpływ na płynność i szybkość działania programu. Rozbudowane technologie takie jak SignalR czy LINQ znacznie przyspieszają niektóre aspekty programowania wysokopoziomowego ale programista wykorzystujący dany framework traci pełną kontrolę nad implementowanym zagadnieniem.