# Dokumentacja użytkowa:

## Ogólnie

### Założenia:

Projekt PingPoint ma na celu zautomatyzowanie gry w tenisa stołowego. Największym atutem jest samoistne liczenie punktów. Dzięki użyciu odpowiedniego sprzętu podczas rozrywki gracze mogą zobaczyć aktualny przebieg meczu. Dodatkowym uatrakcyjnieniem jest przegląd swoich meczy oraz możliwość zaplanowania turnieju.

### Jakie aplikacje są wykorzystywane?

1. **Aplikacja okienkowa** – interfejs użytkownika z którego najlepiej korzystać na komputerze z ekranem dotykowym. Aplikacja ta umożliwia przeprowadzenie meczu dla dwóch graczy posiadających konta w bazie. Gracze po zalogowaniu się w aplikacji wybierają typ meczu, który chcą odbyć. Do wyboru są mecze towarzyskie oraz, zaplanowane wcześniej, turniejowe. W przypadku tych pierwszych gracze mają wybór do ustalenia ilości punktów w secie i do ilu wygranych setów powinna toczyć się rozgrywka. Specjalna kamera nagrywa rozrywkę, a program przechwytuje z niej obraz i na tej podstawie ustala dla kogo padł punkt. Aplikacja okienkowa oferuje podgląd aktualnego wyniku i intuicyjny system dodawania i odejmowania punktu. Po zakończeniu seta gracze muszą go zaakceptować aby rozgrywka mogła zostać kontynuowana. Po zakończeniu meczu towarzyskiego istnieje możliwość rewanżu.
2. **Aplikacja c++** – program działający w tle połączony z kamerą i komunikujący się z aplikacją okienkową. Śledzi ruch piłeczki dzięki zbieraniu z kamery tylko odpowiednich wartości kolorów. Dzięki temu jest w stanie ocenić któremu graczu powinien być uznany punkt. Po ustaleniu wysyła tę informację do aplikacji okienkowej. W chwili rozpoczęcia rozgrywki użytkownik powinien ustawić w aplikacji suwaki odpowiedzialne za wybór dostrzegalnych przez program kolorów. Następnie należy kliknąć podwójnie, by program mógł odnaleźć w obrazie położenie stołu.
3. **Aplikacja webowa** – interfejs użytkownika z którego najlepiej korzystać na komputerze lub telefonie z ekranem dotykowym. Aplikacja ta umożliwia rejestrację oraz zalogowanie się na swoje konto. Użytkownik może obejrzeć swoje zaplanowane jak i ukończone mecze. W przypadku ukończonych aplikacja daje możliwość zobaczenia ich szczegółowego przebiegu. Istnieje też opcja zaplanowania turnieju. Wybierając te opcję, użytkownik staje się nadzorcą danego turnieju i jako jedyny ma możliwość ewentualnego anulowania tego turnieju. Nadzorca turnieju wybiera graczy, którzy mają uczestniczyć w turnieju (użytkownicy którzy również posiadają konta w bazie). Ustala on też do ilu punktów w secie i do ilu wygranych setów powinien trwać mecz. W ten sposób unika się sytuacji w której mecze w tym samym turnieju różniłyby się regułami gry. Do wyboru są dwa typy turniejów: ligowy i pucharowy.

## Aplikacja okienkowa

### Założenia:

Program w C# jest interfejsem użytkownika z możliwościami pokazywania naliczonych już punktów, możliwością manipulowania nimi podczas meczu i zapisywania rozegranych meczy do bazy danych. Dodatkowymi usprawnieniami są – system zliczania który zawodnik powinien serwować w danym momencie meczu, tworzenie i rozgrywanie rozgrywek towarzyskich (również zapisywane do bazy).

### Najlepsze środowisko:

Najprzyjemniejszym dla zawodników sposobem korzystania z programu jest komputer z ekranem dotykowym.

### Jak zacząć mecz?

Do rozpoczęcia rozgrywki należy:

1. Zalogować się jako jeden z graczy:   
   Zawodnik1 – osoba rozpoczynająca rozgrywkę po **lewej** stronie kamery.   
   Zawodnik2 – osoba rozpoczynająca rozgrywkę po **prawej** stronie kamery.
2. Wybrać tryb rozgrywki:

**Mecz towarzyski** – można stworzyć mecz na własnych zasadach.   
Tak zwana szybka gra.

**Turniejowy** – tryb gry podzielony na rozgrywki pucharowe i ligowe.  
Należy go wcześniej stworzyć (jedynie na stronie www)

Po tym można bez problemu zacząć grę.

### Wygrana seta:

Za każdym razem gdy zdobędzie się odpowiednią ilość (przewaga co najmniej 2) punktów do zdobycia seta, zawodnicy podejmują decyzję o zakończeniu go i zmieniają się stronami. Kiedy zawodnicy nie będą zgodni co do ostatniego punktu w secie mogą go poprawić.

### Wygrana meczu:

Po wygraniu ostatniego seta w grze (niebieski kwadrat przy numerze seta wskazuje liczbę wygranych) zostaje ogłoszony wynik wraz z możliwością natychmiastowego rewanżu na tych samych warunkach. Wyjątek od tego stanowią mecze turniejowe.

Gdy kończy się rozgrywkę i nie zażąda rewanżu, gracze zostają automatycznie wylogowani.

## Aplikacja c++

### Założenia:

Zadaniem programu OpenCVPingPoint jest przetwarzanie obrazu odtwarzanego w czasie rzeczywistym, oraz umożliwienie automatycznego zaliczania punktów w zależności od sytuacji na stole. Program jest bezpośrednio powiązany z aplikacją C#, porozumiewając się w celu wskazania która strona otrzymała punkt. Program może analizować albo obraz podpiętej kamery internetowej do komputera, albo podany plik wideo, z uprzednio nagraną rozgrywką.

Program zakłada że kolor piłki pingpongowej jest **pomarańczowy**.

### Wstęp:

Program został napisany w środowisku Visual Studio Community 2017, korzystając z biblioteki OpenCV w wersji 3.4.1. Program kompilowany jest w trybie **Release**, oraz jako program **64-bitowy**. Użytym językiem programowania jest **C++.**

Składa się tylko z jednego pliku: „OpenCVPingPoint.cpp” zawierający całość działania programu. Każda linia kodu w tym pliku jest szczegółowo skomentowana.

### Uruchomienie i kompilacja:

Aby uruchomić aplikację OpenCVPingPoint, należy najpierw uruchomić aplikację C#, inaczej otrzymamy komunikat o błędzie.

Aby poprawie skompilować dołączony projekt, należy umieścić pobraną bibliotekę OpenCV w wersji 3.4.1 w katalogu **„C:\Opencv”,** oraz dodać odpowiednią zmienną środowiskową za pomocą komendy:

setx /M PATH "%PATH%;C:\opencv\build\x64\vc15\bin"

W trakcie instalacji/modyfikacji Visual Studio Community 2017, należy zaznaczyć następujące komponenty:

„**Programowanie aplikacji klasycznych w języku C++** w sekcji **Pakiety robocze**”

„**Windows 10 SDK (10.0.16299.0)**  w sekcji **Opcjonalnie**”

Skompilowany za pomocą dołączonego projektu Visual Studio (znajduje się on w katalogu „**x64\Release**” wewnątrz projektu, należy uruchomić z linii poleceń, podając **dwa argumenty:**

**1 – „k” lub „v” –** mówi czy program ma działać w trybie kamery (k), lub trybu wczytania pliku wideo (v).

**2 – ID kamery lub ścieżka do pliku** – w trybie kamery należy podać ID kamery która jest podpięta do komputera, ID zaczyna się w systemie od 0, czyli 0 – pierwsza kamera, 1 – druga kamera itd. W trybie pliku, należy podać prawidłową ścieżkę do pliku. Obsługa plików zależy od biblioteki OpenCV.

Przykłady uruchomienia z linii poleceń:

C:\>OpenCVPingPoint.exe k 0

C:\>OpenCVPingPoint.exe v C:\Filmy\Film.mp4

### Kalibracja i elementy programu:

Po uruchomieniu programu powinny się pokazać **cztery** okna o poszczególnych nazwach:

**1 – Kalibracja stołu** – służy do ustalenia pozycji stołu, oraz jego poszczególnych stron

**2 – Kalibracja piłki –** służy do ustalenia wyglądu piłki oraz jej pozycji

**3 – Obraz na żywo z linią toru piłki –** jest to obraz podglądowy mający za zadanie sprawdzić poprawność działania programu podczas spotkania

4 – **Liczenie punktów** – zawiera przycisk służący do rozpoczynania, oraz zatrzymywania procesu liczenia punktów.

Wszystkie te okna z wyjątkiem ostatniego, można dowoli rozszerzać i zmniejszać.

Aby program działał prawidłowo, należy najpierw przygotować odpowiednie ujęcie do kalibracji, oto przykład którym należy się kierować:



Ważne elementy:

- wyraźny kontrast pomiędzy piłką a resztą obrazu

- jak najmniejsza ilość szumów

- brak elementów o podobnym kolorze do piłki

- wyraźnie widoczna piłka na stole

W trybie **video** można naciskać klawisz „**p”** aby zatrzymać/odtworzyć z powrotem plik video, w celu dokonania kalibracji.

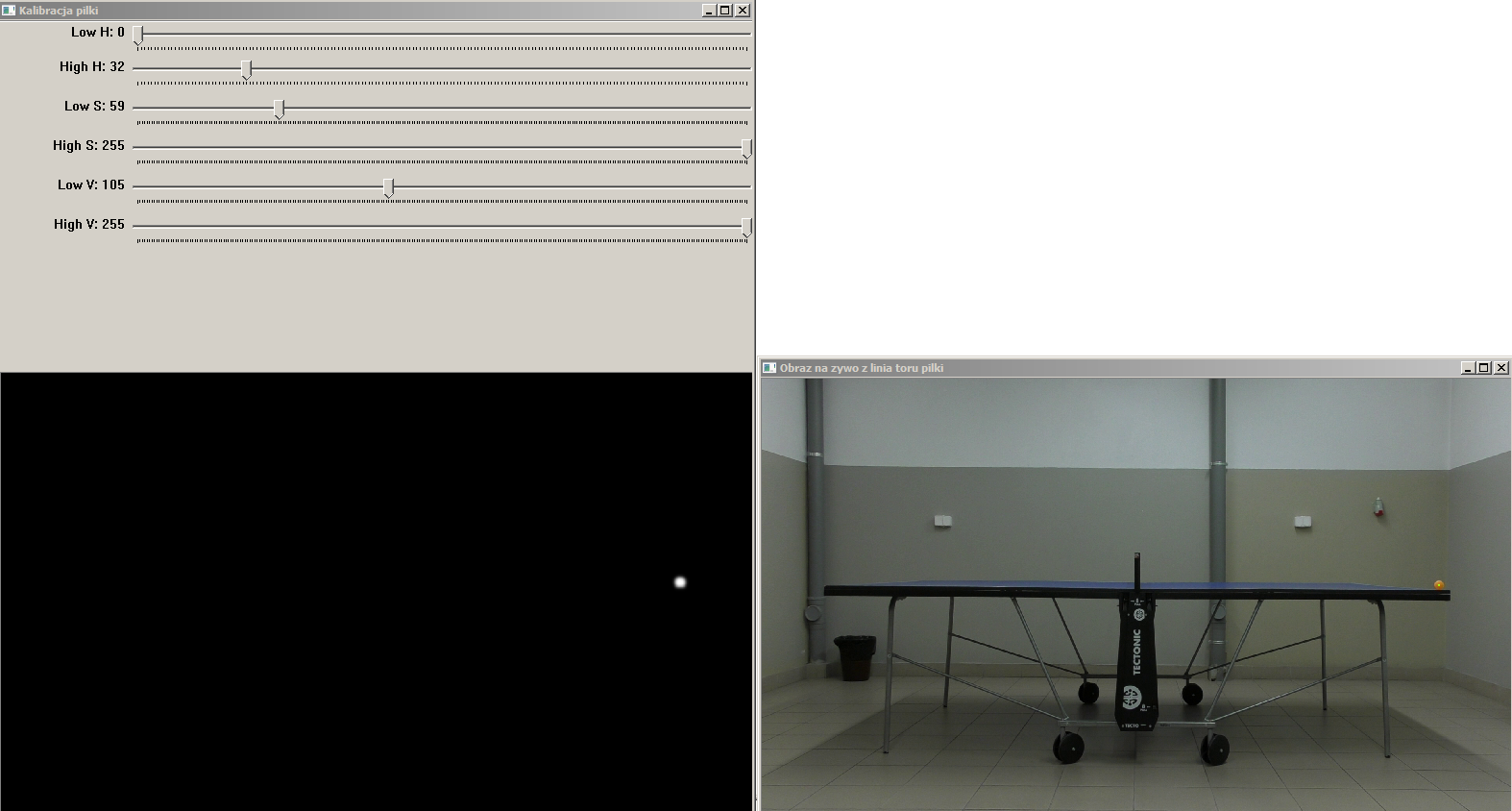
Należy zwrócić uwagę, że w zależności od rozdzielczości

Gdy już mamy odpowiednie ujęcie, możemy przejść do kalibracji:

1. W oknie **Kalibracja stołu** pojawiają się odpowiednie instrukcje, należy kliknąć **lewym** przyciskiem myszy na lewy wierzchołek stołu (który pojawi się jako czerwony punkt), a następnie na prawy (żółty punkt). Finalnie powinno to wyglądać tak:

Należy sprawdzić czy siatka pokrywa się z zielonym punktem symbolizującym środek stołu.

Jeżeli kalibracja jest niezadowalająca, możemy ją usunąć naciskając **prawy** przycisk myszy w tym oknie.

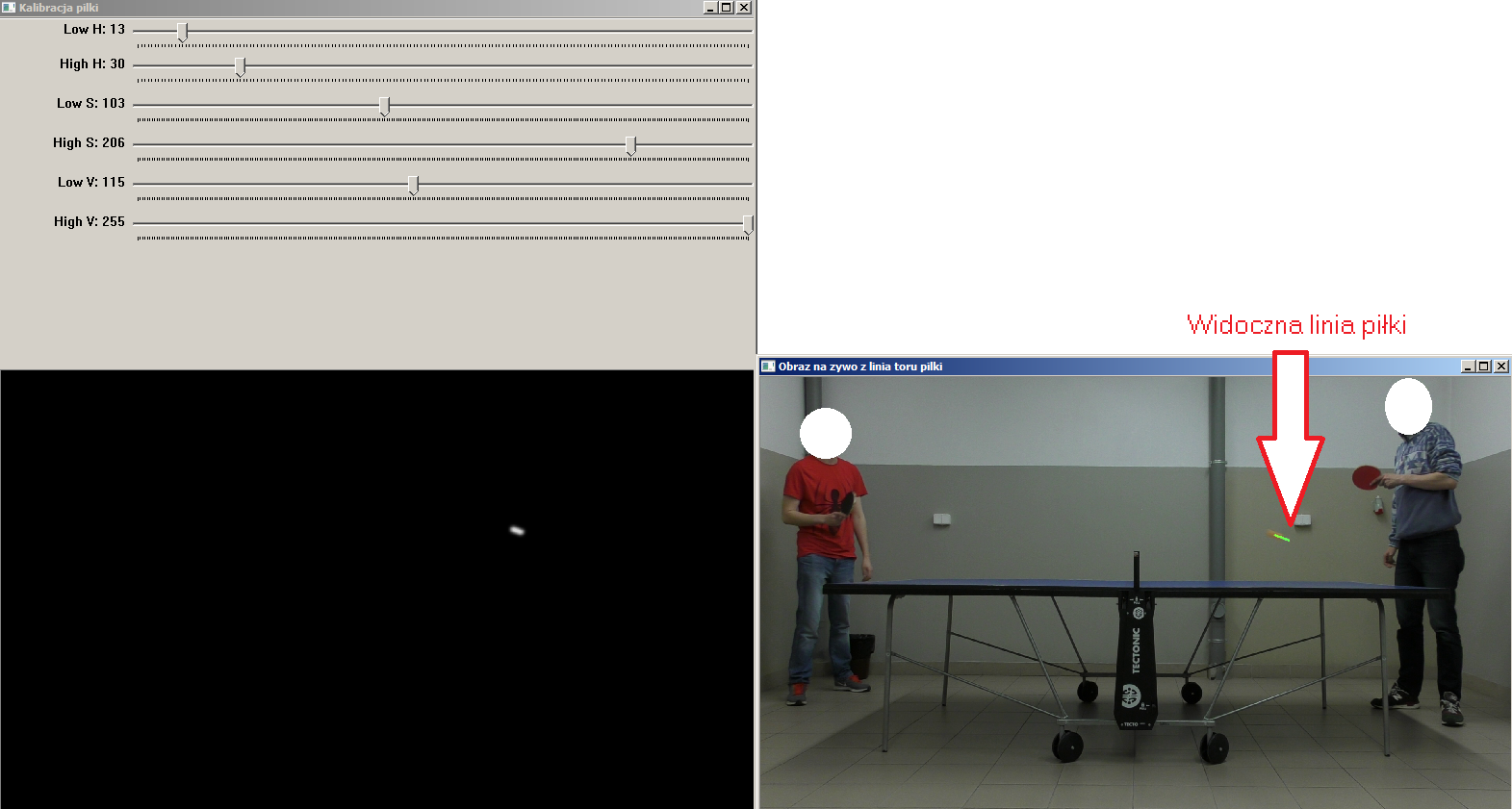
1. W oknie **Kalibracja piłki** należy przesuwać suwaki w taki sposób, aby widoczny pod suwakami obraz zawierał tylko piłkę jako biały okrąg. Należy zacząć od suwaka **Low H**, a następnie po kolei ustawiać coraz to niższe suwaki, aż piłka pozostanie jedynym obiektem. Ustawianie suwaków polega na tym, aby znaleźć jak najbardziej skrajną pozycję, w której piłka nie znika. Finalnie powinno to wyglądać następująco:

Można wtedy zauważyć że w oknie poglądowym, widzimy zieloną kropkę w środku piłki, jest to linia toru piłki którą będziemy obserwowali i jest ściśle powiązana z kalibracją stołu.

1. Należy kliknąć przycisk: **Zacznij liczenie** w ostatnim oknie.

Jeżeli wszystko zostało wykonane prawidłowo, punkty powinny być automatycznie liczone i uaktualniane w aplikacji C#. W celu unikania fałszywie rozpoznawanych punktów, warto zatrzymywać liczenie w razie potrzeby.

Przykład dobrej kalibracji:



### Zamknięcie:

Program zamykamy klikając klawisz **ESC** gdy jedno z czterech okien jest zaznaczone jako aktywne.

## Aplikacja webowa

### Założenia:

Aplikacja webowa jest interfejsem użytkownika z możliwościami stworzenia nowego turnieju, wyświetlania poszczególnych meczy (zarówno ligowych jak i pucharowych) oraz pokazywania naliczonych już punktów w danych setach. Dodatkowymi uprawnieniem jest możliwość usunięcia turnieju przez nadzorcę.

### Najlepsze środowisko:

Najprzyjemniejszym dla zawodników sposobem korzystania z programu jest komputer z ekranem dotykowym lub smartfon.

### Jak korzystać z aplikacji?

Po wejściu na stronę główna należy nacisnąć Utwórz Konto (prawy górny róg)  
Po naciśnięciu przenosimy się do strony z rejestracją gdzie należy podać :

a) login – nazwa użytkownika jaką będziemy się posługiwać w aplikacji  
b) email – skrzynkę pocztową użytkownika  
c) hasło – najlepiej 8 znaków z 1 dużą literą i znakiem specjalnym

Po zalogowaniu przejdziemy na stronę domową gdzie znajdują się statystyki naszego konta (np. wynik ostatnich meczy) oraz zaplanowane mecze turniejowe.

Jeżeli chcemy dowiedzieć się więcej na temat danego meczu wystarczy że naciśniemy przypadający mu przycisk ”relacja” który rozwinie nam szczegóły na temat zdobyczy punktowych w poszczególnych setach.

### Stworzenie Turnieju

Aby stworzyć turniej musimy być zalogowani. Po naciśnięciu „Utwórz Turniej” z górnego menu zostaniemy przekierowani do konfiguracji turnieju gdzie trzeba podać:

a) punkty – ilość punktów potrzebna do wygrania seta  
b) sety – ilość setów jaka ma być potrzebna do wygrania meczu  
c) typ turnieju – (ligowy - każdy z każdym, pucharowy - w formie drzewka)   
d) opis turnieju – ciekawy opis  
e) wybierz graczy – gracze, których chcemy dodać do turnieju

### Moje Turnieje

Aby zobaczyć listę wszystkich turniejów, w których bierze się udział, wystarczy nacisnąć na swój avatar, a następnie wybrać turnieje. Po naciśnięciu strzałki skierowanej w dół zobaczymy wszystkie mecze danego turnieju. Aby zobaczyć szczegóły meczu należy nacisnąć przycisk ”sety”. Jeżeli jesteśmy nadzorcą danego turnieju możemy go również usunąć klikając w czerwony przycisk z prawej strony.

### Moje mecze

Dostęp do moich meczy jest analogiczny do „Moje Turnieje”. Tu również zobaczymy listę meczy. Aby dowiedzieć się więcej o danym meczu naciskamy „relacja”. Jeżeli rozgrywany mecz był turniejowym możemy przejść bezpośrednio do danego turnieju klikając w „id turnieju : id”.

# Dokumentacja szczegółowa:

# Dokumentacja szczegółowa C#:

## Podstawowe informacje.

Aplikacja Ping Point C# (GUI) jest stworzona w Visual Studio 2015 dla .NET Framework 4.5.2(x86 and x64), jako Windows Forms Application, która zawiera 6 form:

* PingPoint\_main.cs – główne okienko aplikacji, które zarządza i wywołuje inne okienka.
* Login.cs – okienko wyskakujące podczas próby zalogowania się użytkownika.
* FriendlyMatchSettings.cs – okienko określające ustawienia meczu towarzyskiego.
* TournamentMatchSettings.cs – okienko w którym można wybrać turniej do rozegrania.
* Wait.cs – okienko służące do zatwierdzenia lub poprawienia seta przed zapisaniem w bazie.
* Endgame.cs – okienko informujące o wygraniu przez daną osobę meczu, z opcją rewanżu.

Dodatkową funkcjonalność – połączenia z bazą danych MySQL aplikacja realizuje za pomocą biblioteki MySql.Data.MySqlClient, a połączenie z aplikacją napisaną w c++ realizuje dzięki klasie NamedPipeServer.

## PingPoint\_main.

### Wykorzystywane zmienne i struktury

* struct point - Struktura, która przechowuje dane o graczu, oraz punkcie jaki zdobył.
* List<point> points – lista punktów.
* recivedPointSide – zmienna w której zapisywana jest wartość 0 lub 1 w zależności od strony zdobycia punktu (0 – lewa 1 – prawa) – wiadomość przychodzi z aplikacji C++.
* timer – jest to Timer służący pobieraniu wiadomości z aplikacji C++.
* ConnString – connection string określający parametry bazy danych.
* MySqlConnection conn – używające connection stringa połączenie do bazy MySQL.
* logged1/2 – zmienne boolowskie, informujące czy gracz się już zalogował, czy nie.
* choosed – zmienna boolowska, informująca czy został wybrany tryb rozgrywki.
* set\_max/point\_max – zmienne liczbowe określające ilość setów i punktów w grze.
* point\_margin – zmienna stanowiąca jaka musi być różnica punktów by zdobyć set.
* tournament\_id – zmienna przechowująca id turnieju. Pobierane z bazy danych.
* match\_id – zmienna przechowująca id meczu. Pobierane z bazy danych.
* tournament\_type – zmienna przechowująca typ turnieju. Pobierane z bazy danych.
* points\_id – zmienna określająca id punktu w danym meczu.
* point1/2, sets\_player1/2 – zmienne przetrzymujące obecny stan rozgrywki.
* rematch – zmienna boolowska określająca czy zawodnik zażądał rewanżu.
* accept\_set – zmienna boolowska określająca czy set został zaakceptowany.
* my\_login – zmienna przechowująca login z bazy danych.
* my\_password – zmienna przechowująca hasło z bazy danych.
* serve\_player1 – zmienna boolowska przechowująca który zawodnik serwuje.
* serve\_player\_start – zmienna określająca który gracz zaczynał rozgrywkę (1 set).
* serve\_number – zmienna określająca ile serwów ma każdy z graczy przed zmianą.
* serve\_points\_old – zmienna przechowująca poprzednią ilość sumy punktów zawodników w danym secie. (zmiana serwów)
* PServer1 – obiekt klasy NamedPipeServer który tworzy odbiera informacje z aplikacji w C++.
* free – zmienna boolowska ograniczająca (gdy false) dostęp do funkcji czytania.

### PingPoint\_main()

Funkcja inicjalizująca cały program.

### Obsługa połączenia z aplikacją w C++

**startConnectCpp()** - Funkcja tworząca połączenie z aplikacją w c++ przez utworzenie NamedPipe. W tym wypadku na stworzony obiekt (PServer1) działamy metodą start(), która uruchamia połączenie z aplikacją c++. Ponadto zostaje ustawiony timer który będzie wywoływał funkcję connectCpp za każdym razem wywołania eventu Tick.

**stopConnectCpp()** - Funkcja zatrzymująca połączenie z aplikacją w c++. Dzieje się to poprzez wywołanie metody StopServer() na obiekt PServer1.

**connectCpp(**object sender, EventArgs e**)** - Funkcja wywoływana za każdym razem gdy event Tick obiektu timer wystąpi. Funkcja zgodnie z otrzymanym sygnałem (0, 1) przyznaje punkt danemu graczu, uwzględniając zmiany stron po każdym secie.

### Obsługa połączenia z bazą danych MySQL

**OpenConnection() -** Funkcja tworząca połączenie z bazą danych. Po udanej próbie połączenia z bazą danych wartość zwracana będzie równa true, w przeciwnym wypadku false, oraz pokaże się komunikat o błędzie.

**CloseConnection()** - Funkcja zamykająca połączenie z bazą danych. Po udanym zamknięciu funkcja zwraca true, w przeciwnym wypadku false i zostaje pokazany komunikat o błędzie.

### Obsługa eventów

**PingPoint\_main\_Load(**object sender, EventArgs e**) -** Funkcja obsługująca event załadowania okienka PingPoint\_main.   
W tym wypadku wywołuje funkcję OpenConnection()

**button\_exit\_Click(**object sender, EventArgs e**)** - Funkcja obsługująca event kliknięcia na button\_exit.  
W tym wypadku wywołuje funkcję CloseConnection(), oraz zamyka aplikację.

**button\_start\_Click(**object sender, EventArgs e**)** - Funkcja obsługująca event kliknięcia na button\_start.   
W tym wypadku sprawdzamy czy osoby się zalogowały i wybrały rodzaj rozgrywki. W przeciwnym wypadku zgłaszamy komunikat o błędzie.  
Gdy wszystkie wymagania zostaną spełnione funkcja wywołuje cleanup(false) – czyszczenie bez wylogowania, oraz przygotowuje okienko do rozgrywki – wyświetla odpowiednie składowe, dodaje podstawowe wartości, oraz blokuje niektóre funkcje programu. Gdy jest grany mecz towarzyski – zostaje on dodany do bazy danych, w wypadku gdy jest to mecz turniejowy – odczytany zostaje jego id\_meczu z bazy danych.

**button\_login1\_Click(**object sender, EventArgs e**)** - Funkcja obsługująca zalogowanie się zawodnika1.   
Sprawdza poprawność wprowadzonych danych. W przypadku błędu zalogowania zostaje wyświetlony odpowiedni komunikat.

**button\_login2\_Click(**object sender, EventArgs e**)** - Funkcja obsługująca zalogowanie się zawodnika2.   
Sprawdza poprawność wprowadzonych danych. W przypadku błędu zalogowania zostaje wyświetlony odpowiedni komunikat.

**checkBox\_auto\_CheckedChanged(**object sender, EventArgs e**)** - Funkcja obsługująca opcję automatycznego zliczania punktów dzięki programowi w C++ z podłączoną kamerą.  
Gdy checkBox\_auto jest zaznaczony – punkty będą naliczane automatycznie. Jest to domyślna funkcjonalność.

**listBox\_rodzaj\_SelectedIndexChanged(**object sender, EventArgs e**)** - Funkcja obsługująca event wyboru trybu gry.  
Gdy użytkownik wybierze mecz turniejowy – zostaje uruchomione okienko z wyborem turnieju(TournamentMatchSettings), w którym zostaje wczytana lista turniejów dostępnych dla zalogowanych w tym czasie obu użytkowników – tak by mogli zagrać ten mecz.   
Gdy użytkownik wybierze mecz towarzyski – zostaje uruchomione okienko z możliwością ustawienia parametrów rozgrywki.

#### Interfejs zmiany punktów:

**label\_points\_up1\_Click(**object sender, EventArgs e**)** - Funkcja obsługująca dodanie punktu zawodnika1.  
Dodajemy do listy points kolejny punkt dla zawodnika1. Gdy punkty osiągną liczbę potrzebną do wygrania seta sprawdzamy czy przewaga punktowa jest większa niż 1. Jeżeli tak otwieramy okienko Wait.cs i oczekujemy na odpowiedź użytkownika (akceptacja seta / poprawienie).   
Każdorazowo funkcja powinna wywołać points\_update().

**label\_points\_down1\_Click(**object sender, EventArgs e**)** - Funkcja obsługująca odjęcie punktu zawodnika1.  
Usuwamy z listy points ostatni punkt zawodnika1. Gdy punkty osiągną liczbę potrzebną do wygrania seta sprawdzamy czy przewaga punktowa jest większa niż 1. Jeżeli tak otwieramy okienko Wait.cs i oczekujemy na odpowiedź użytkownika (akceptacja seta / poprawienie).   
Każdorazowo funkcja powinna wywołać points\_update().

**label\_points\_up2\_Click(**object sender, EventArgs e**)** - Funkcja obsługująca dodanie punktu zawodnika2.  
Funkcjonalność taka sama jak podana wyżej **label\_points\_up1\_Click(**object sender, EventArgs e**)**, jedynie dla zawodnika2.

**label\_points\_down2\_Click(**object sender, EventArgs e**)** - Funkcja obsługująca odjęcie punktu zawodnika2.  
Funkcjonalność taka sama jak podana wyżej **label\_points\_down1\_Click(**object sender, EventArgs e**)**, jedynie dla zawodnika2.

### last\_point(int player)

Funkcja wyszukuje ostatni punkt zawodnika o id 'player' w liście 'points'.

### points\_update()

Funkcja która wywołuje się za każdym razem gdy powinien zmienić się punkt. Odświeża ona interfejs użytkownika i zmienia zmienne odpowiedziane za punkty, zmienia zawodnika który powinien serwować – jeżeli jest taka potrzeba.

### sets\_update(int win)

Funkcja która wywołuje się za każdym razem gdy powinien zmienić się set. Przyjmuje ona jako parametr win – który zawodnik zdobył punkt. Funkcja zmienia co set osobę która zaczyna serw, dodaje wynik seta do tabeli punkty w bazie danych, oraz sprawdza czy jest to koniec meczu.  
Wywołuje funkcje endgame(zawodnik) – w przypadku końcu meczu  
Wywołuje funkcje czyszczenia listy punktów ‘points’, oraz points\_update()

### cleanup(bool all)

Funkcja czyszcząca ekran aplikacji. Gdy parametr 'all' == true wylogowuje zawodników. W środku funkcji odbywają się następujące czynności:

* Przywróć zmienne do domyślnych wartości.
* Przywróć atrybut Visible do domyślnych wartości.
* Przywróć atrybut Enabled do domyślnych wartości.
* Oraz ewentualne wylogowanie zawodników – zmienne logged, choosed itp.

### endgame(int win)

Funkcja wywoływana przy skończeniu meczu. Przyjmuje jako parametr win – zwycięzcę meczu, następnie zachowuje się następująco:

* 1. Gdy jest to mecz turniejowy
     1. Zmień datę zagranego meczu na dzisiejszą.
     2. Gdy typ turnieju jest pucharowy
        1. Gdy ilość nierozegranych meczy jest równa 0
           1. Liczymy ilość graczy w turnieju (N)
           2. Liczymy ilość meczy (M)
           3. Jeżeli puchar się nie skończył (M !=N)

Zauważmy że w gdy ilość meczy będzie równa liczbie zawodników – turniej można uznać za skończony.

Zapisujemy wygranych z ostatnich meczy

Gdy rozegrany został półfinał

Zapisujemy przegranych z ostatnich meczy

Dodaje do bazy danych mecze o 3 miejsce oraz mecz finałowy

W przeciwnym wypadku dodaj mecze do bazy danych osobom którzy byli zwycięzcami.

* 1. W przeciwnym wypadku – mecz NIE JEST turniejowy – dodaj opcję rewanżu.
  2. Wyświetl okienko Endgame.cs i oczekuj na odpowiedź użytkownika odnośnie dalszych postępowań (koniec meczu / rewanż).

## Login.

### button\_sign\_in\_Click(object sender, EventArgs e)

Funkcja wykonywana w momencie wystąpienia eventu kliknięcia na przycisk Zaloguj.  
Sprawdza czy wpisane wartości nie są puste, gdy nie - wysyła do PingPoint\_main.cs wpisane dane, a stamtąd szukane zostają w bazie danych. Po tych operacjach okienko zostaje zamknięte.

### textBox\_password\_KeyDown(object sender, KeyEventArgs e)

Funkcja wywoływana w momencie naciśnięcia klawisza w polu tekstowym na hasło.   
Sprawdza czy klawisz który został naciśnięty jest enterem – intuicyjne zatwierdzenie wprowadzonych danych.

## FriendlyMatchSettings.

### Ustawianie przykładowych wartości

**button\_var1\_Click(**object sender, EventArgs e**)** – funkcja która zostanie wykonana po wciśnięciu buton\_var1 przez użytkownika.  
Funkcja ustawia przykładowe wartości (krótki mecz) dla meczu towarzyskiego.

**button\_var2\_Click(**object sender, EventArgs e**)** – funkcja która zostanie wykonana po wciśnięciu buton\_var2 przez użytkownika.  
Funkcja ustawia przykładowe wartości (średni mecz) dla meczu towarzyskiego.

**button\_var3\_Click(**object sender, EventArgs e**)** – funkcja która zostanie wykonana po wciśnięciu buton\_var2 przez użytkownika.  
Funkcja ustawia przykładowe wartości (długi mecz) dla meczu towarzyskiego.

### button\_save\_Click(object sender, EventArgs e)

Funkcja jest wywoływana w momencie naciśnięcia przez użytkownika buton\_save.  
Ma ona za zadanie przesłać informacje do okienka macierzystego (PingPoint\_main) informacji dotyczących ustawienia meczu towarzyskiego. Po wykonaniu okienko zostaje zamknięte.

## TournamentMatchSettings.

### Wykorzystywane zmienne

* points – liczba punktów do której się gra. Wyciągnięte z bazy danych
* sets – liczba setów do której się gra. Wyciągnięte z bazy danych
* t\_id – id turnieju. Wyciągnięte z bazy danych
* t\_type – typ turnieju. Wyciągnięte z bazy danych

### TournamentMatchSettings(List<string> tablica, MySqlConnection conn)

Konstruktor klasy, parametry tablica i conn są odpowiednio – tabicą id wszystkich turniejów dostępnych dla zalogowanych graczy, conn – connection string potrzebny do połączenia się z bazą danych.  
W środku konstruktora znajduje się kod odpowiedzialny za odczytanie danych dla turniejów o id znajdujących się w tablicy. Dane zostają wczytane do DataTable data\_tournament\_info, lub w przypadku braku wyników – wysyłany zostaje komunikat o błędzie.

### dataGridView\_tournament\_SelectionChanged(object sender, EventArgs e)

Funkcja jest wywoływana gdy użytkownik zmieni zaznaczenie turnieju – wybierze jeden z podanych.  
Ma ona za zadanie zapisać do zmiennych points, sets, t\_id, t\_type dane z data\_tournament\_info utworzone w konstruktorze.

### button1\_Click(object sender, EventArgs e)

Funkcja wywoływana gdy buton1 zostanie wciśnięty.   
Jest odpowiedzialna za przesyłanie do okienka macierzystego (PingPoint\_main) informacji dotyczących turnieju takich jak np. jego id, ilość punktów, setów, czy typ. Po zakończeniu tej funkcji okienko zostaje zamknięte.

## Wait.

### Wait(string win)

Konstruktor klasy, przyjmujący jako parametr nazwę gracza który wygrał set.

### label\_change\_Click(object sender, EventArgs e)

Funkcja wywoływana gdy kliknie się na pole napisu informującego o zmianie seta.  
Wysyła informacje o poprawieniu seta do okienka macierzystego i zamyka bieżące.

### label\_next\_set\_Click(object sender, EventArgs e)

Funkcja wywoływana gdy kliknie się na pole napisu informującego o zaakceptowaniu seta.  
Wysyła informacje o zakończeniu seta do okienka macierzystego i zamyka bieżące.

## Endgame.

### Endgame(string win, bool turniej)

Konstruktor klasy, przyjmujący parametry win i turniej oznaczające odpowiednio – nazwę gracza który wygrał mecz, informację czy jest to mecz turniejowy (true) czy towarzyski (false).  
Gdy jest to mecz towarzyski funkcja przyzwala użytkownikowi na rozegranie rewanżu.

### button\_rematch\_Click(object sender, EventArgs e)

Funkcja możliwa do wywołania jedynie gdy jest to mecz towarzyski i został naciśnięty button\_rematch.  
Wysyła informacje o rozpoczęciu rewanżu do okienka macierzystego i zamyka bieżące.

### button\_endgame\_Click(object sender, EventArgs e)

Funkcja wywoływana poprzez naciśnięcie przycisku button\_endgame.  
Wywołanie funkcji skutkuje skończeniem meczu (poinformowaniu o tym okienko macierzyste – PingPoint\_main), oraz zamknięciem bieżącego okienka.

# Dokumentacja szczegółowa C++:

# Dokumentacja szczegółowa webowa:

## 1. Dodatkowe biblioteki

Aplikacja webowa Ping Point została napisana w języku Python wersji 3.6 z wykorzystaniem frameworka Flask z użyciem paczek:

* Flask
* Flask-MySQL
* Flask-MySQLdb
* Jinja2
* MarkupSafe
* PyMySQL
* Werkzeug
* click
* itsdangerous

## 2. app.py

Cały mechanizm działania aplikacji zawarty został w pliku app.py, w którym znajdują się następujące funkcje oraz metody:

* **is\_power2(n)** - funkcja sprawdzająca czy podana liczba jest potęgą dwójki. Używana przy tworzeniu nowego turnieju pucharowego w metodzie **newTournament()**
* **getMatchDetails(id\_meczu, login)** - funkcja, która pobiera id meczu oraz login zalogowanego użytkownika i zwraca słownik zawierający informacje o meczu w formie: {

'przeciwnik' - login przeciwnika

'data' - data rozegrania meczu

'punkty\_moje' - zdobyte przez zalogowanego użytkownika punkty w każdym secie

'punkty\_przeciwnika' - zdobyte przez przeciwnika punkty w każdym secie

'przebieg\_meczu' - zmienna zawierająca dane o zdobytych lub straconych punktach

'wynik\_meczu' - ostateczny wynik meczu 'id\_turnieju' - id turnieju, jeżeli mecz zawiera się w jakimś turnieju

}

* **getUserMatches(login)** - funkcja pobiera login użytkownika, zwraca listę rozegranych przez niego meczy zaczynając w kolejności od meczy rozegranych. Do pobierania danych o każdym z meczy używa funkcji **getMatchDetails(id\_meczu, login)**.
* **getTournamentMatches(id\_turnieju)** - funkcja, która pobiera id turnieju i zwraca listę meczy dla tego turnieju w kolejności zaczynając od meczy rozegranych, tak jak w przypadku **getUserMatches(login)**.
* **getTournamentDetails(id\_turnieju)** - funkcja pobiera id turnieju i zwraca słownik zawierający dane o turnieju: {

'id' - id turnieju

'nadzorca' - login nadzorcy czyli gracza, który utworzył turniej

'typ' - typ turnieju (ligowy lub pucharowy)

'opis' - opis turnieju

'mecze' - lista ze szczegółowymi danymi o meczach tego turnieju

'rundy' - poszczególne etapy turnieju pucharowego, zmienna potrzebna do zbudowania drzewa turnieju

'ranking' - ranking najlepszych uczestników turnieju

'tabela' - dane potrzebne do zbudowania tabeli wyników meczu ligowego }

* **main()** - metoda odpowiedzialna za wyświetlanie strony domowej zalogowanego użytkownika, zaś dla niezalogowanego, wyświetlona zostaje strona do logowania bądź rejestracji.
* **showSignUp()** - metoda wyświetla stronę z formularzem do zarejestrowania się. W przypadku zalogowanego użytkownika, zwróci jego stronę domową.
* **signUp()** - metoda odpowiedzialna za pobranie danych o rejestracji z formularza, walidację danych i wyświetlenie informacji o utworzeniu bądź nie utworzeniu konta. Korzysta z procedury w bazie danych: **sp\_createUser**.
* **showSignin()** - metoda odpowiada za wyświetlenie strony z formularzem do logowania, w przypadku gdy użytkownik jest już zalogowany wyświetla jego stronę domową.
* **validateLogin** - metoda wykonywana po wypełnieniu formularza logowania, przeprowadza walidację danych i w zależności od poprawności danych wyświetla stronę domową użytkownika lub informację o nieudanym zalogowaniu się. Zawiera wywołanie procedury w bazie danych: **sp\_validateLogin**.
* **userHome()** - metoda wyświetla stronę domową zalogowanego użytkownika,

Obliczone w niej zostają informacje w postaci słowników:

**statystyki**={

'rozegranych' - ilość rozegranych meczy

'wygranych' - ilość wygranych meczy

'nierozegranych' - ilość zaplanowanych meczy turniejowych }

**zdarzenia**={

'rozegrane\_mecze' - lista ostatnich trzech rozegranych meczy

'nierozegrane\_mecze' - lista wszystkich zaplanowanych meczy }

Wykorzystuje procedury w bazie danych: **sp\_getMecz**, **sp\_getWinners** oraz funkcję **getMatchDetails()**.

* **myMatches()** - metoda ta wyświetla wszystkie rozegrane mecze zalogowanego użytkownika. Wykorzystuje funkcję **getUserMatches()**.
* **showTournamentForm()** - metoda odpowiedzialna jest za zgromadzenie listy użytkowników zarejestrowanych w systemie i przekazanie tej listy do formularza tworzenia nowego turnieju. Wywołuje procedurę w bazie danych **sp\_getUsers**.
* **newTournament()** - metoda pobiera z formularza nowego turnieju dane o turnieju, tworzy turniej oraz jego mecze w bazie danych. Wywołuje w bazie danych procedury: **sp\_newTurniej**, **sp\_newMecz**.
* **showTournament(id)** - metoda pobiera id turnieju i wyświetla stronę z informacjami o tym turnieju. Wywołuje procedurę **getTournamentDetails()**.
* **myTournaments()** - metoda odpowiada za pobranie danych o wszystkich turniejach zalogowanego gracza i wyświetleniu ich. Korzysta z funkcji **getTournamentDetails()** oraz procedury w bazie danych **sp\_getTurniejeIdGracza**.
* **deleteTournament(id)** - metoda pobiera id turnieju i usuwa go z bazy danych wykorzystując procedurę w bazie danych **sp\_deleteTurniej**.
* **rank()** - metoda pobiera informacje o liczbie wygranych oraz przegranych meczy. Tworzy listę użytkowników wraz z tymi informacjami oraz sortuje według: ilość wygranych / ilość przegranych. Wywołuje procedury w bazie danych: **sp\_getWinners** i **sp\_getLoosers**
* **logout()** - metoda służąca do wylogowania się użytkownika.

# Dokumentacja szczegółowa bazy SQL:

## 1. Podstawowe informacje:

Baza danych została stworzona w języku mysql na serwerze phpadmin.

Składa się ona z 15 procedur i 4 tabel.

## 2. Tabele:

*Gracze* – zawiera ona informacje o użytkownikach aplikacji:

- *login* – login użytkownika za pomocą którego loguje się w aplikacjach, max. długość: 45 znaków, obowiązkowy, klucz główny tej tabeli,

- *email* – email kontaktowy danego użytkownika, max. długość: 45 znaków, obowiązkowy,

- *haslo* – hasło użytkownika za pomocą którego loguje się w aplikacjach, max. długość: 200 znaków, obowiązkowe.

|  |  |  |  |
| --- | --- | --- | --- |
| Kolumna | Typ | Null | Ustawienia domyślne |
| login | varchar(45) | Nie |  |
| email | varchar(45) | Nie |  |
| haslo | varchar(200) | Nie |  |

Indeksy:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Nazwa klucza | Typ | Jednoznaczny | Spakowany | Kolumna | Moc | Metoda porównywania napisów | Null |
| PRIMARY | BTREE | Tak | Nie | login | 24 | A | Nie |

*Mecze* – zawiera ona ogólne informacje na temat meczy:

- *mecze\_id* – klucz główny tabeli, zawiera unikalny numer meczu, będący liczbą naturalną, max. 11-cyfrową, autoinkrementowane,

- *turnieje\_id* – jeśli mecz jest meczem turniejowym, to kolumna ta zawiera klucz turnieju do którego należy, w przeciwnym wypadku wynosi null

- *gracz1\_id* – login pierwszego gracza, klucz obcy pochodzący z tabeli gracze, obowiązkowe,

- *gracz2\_id* – login drugiego gracza, klucz obcy pochodzący z tabeli gracze, obowiązkowe,

- *data* – data rozegrania meczu, w przypadku zaplanowanego meczu turniejowego, który jeszcze się nie odbył, wynosi ona null.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kolumna | Typ | Null | Ustawienia domyślne | Odsyłacze do |
| mecze\_id | int(11) | Nie |  |  |
| turnieje\_id | int(11) | Tak | NULL | turnieje -> turnieje\_id |
| gracz1\_id | varchar(45) | Nie |  | gracze -> login |
| gracz2\_id | varchar(45) | Nie |  | gracze -> login |
| data | date | Tak | NULL |  |

Indeksy:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Nazwa klucza | Typ | Jednoznaczny | Spakowany | Kolumna | Moc | Metoda porównywania napisów | Null |
| PRIMARY | BTREE | Tak | Nie | mecze\_id | 32 | A | Nie |
| gracz1\_id | BTREE | Nie | Nie | gracz1\_id | 32 | A | Nie |
| gracz2\_id | BTREE | Nie | Nie | gracz2\_id | 32 | A | Nie |
| turnieje\_id | BTREE | Nie | Nie | turnieje\_id | 16 | A | Tak |

*Punkty* – zawiera ona szczegółowe informacje na temat konkretnych meczy, każdy wiersz odpowiada jednemu punktowi w meczu:

- *numer\_setu* – numer setu, w którym padł dany punkt, max. dwucyfrowy, obowiązkowy

- *punkt* – login użytkownika, który zdobył punkt , klucz obcy z tabeli gracze

- *mecze\_id* – numer meczu do którego należy dany przebieg rozrywki, klucz obcy z tabeli mecze

- *punkty\_id* – unikalny numer, liczony od 1 dla każdego meczu, pokazuje w jakiej kolejności padały punkty, max. 11-cyfrowy.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kolumna | Typ | Null | Ustawienia domyślne | Odsyłacze do |
| numer\_setu | int(2) | Nie |  |  |
| punkt | varchar(45) | Nie |  | gracze -> login |
| mecze\_id | int(11) | Nie |  | mecze -> mecze\_id |
| punkty\_id | int(11) | Nie |  |  |

Indeksy:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Nazwa klucza | Typ | Jednoznaczny | Spakowany | Kolumna | Moc | Metoda porównywania napisów | Null |
| PRIMARY | BTREE | Tak | Nie | punkty\_id | 93 | A | Nie |
| mecze\_id | 559 | A | Nie |
| punkty\_mecze\_fk | BTREE | Nie | Nie | mecze\_id | 32 | A | Nie |
| punkty\_gracze\_fk | BTREE | Nie | Nie | punkt | 20 | A | Nie |

*Turnieje* – zawiera ogólne informacje na temat turniejów:

- *turnieje\_id* – unikalny numer turnieju, max. 11 cyfrowy, klucz główny tabeli, autoinkrementowane

- *do\_ilu\_punkty* – ilość punktów do której powinien toczyć się set, przy przewadze dwóch punktów, liczba max. dwucyfrowa,

- *do\_ilu\_sety* – ilość setów, która decyduje o wygranej w meczu, liczba jednocyfrowa

- *typ* – typ turnieju: pucharowy – pary, ligowy – każdy z każdym

- *opis* – opis turnieju,

- *nadzorca* – użytkownik odpowiedzialny za stworzenie turnieju, klucz obcy z tabeli gracze.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kolumna | Typ | Null | Ustawienia domyślne | Odsyłacze do |
| turnieje\_id | int(11) | Nie |  |  |
| do\_ilu\_punkty | int(2) | Nie | 11 |  |
| do\_ilu\_sety | int(1) | Nie | 3 |  |
| typ | varchar(9) | Nie |  |  |
| opis | varchar(200) | Tak | NULL |  |
| nadzorca | varchar(45) | Nie |  | gracze -> login |

Indeksy:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Nazwa klucza | Typ | Jednoznaczny | Spakowany | Kolumna | Moc | Metoda porównywania napisów | Null |
| PRIMARY | BTREE | Tak | Nie | turnieje\_id | 10 | A | Nie |
| nadzorca | BTREE | Nie | Nie | nadzorca | 5 | A | Nie |

## 3. Procedury:

- *sp\_createUser* – tworzy nowego użytkownika w tabeli *gracze* dla podanych: *login*, *email*, *haslo*,

- *sp\_deleteAll* – usuwa wszystkie dane bazy, ustawia autoinkrementację na 1, dodaje dwóch testowych użytkowników,

- *sp\_deleteTurniej* – usuwa dany turniej przy podaniu jego id oraz wszystkie jego mecze z tabeli *mecze* i ich przebieg z tabeli *punkty*,

- *sp\_getLoosers* – sporządza listę użytkowników z podaną ilością przegranych przez nich meczy, posortowaną nierosnąco,

- *sp\_getMecz* – pobiera wszystkie informacje o meczach z tabeli *mecze*, w których brał udział dany użytkownik o podanym loginie,

- *sp\_getMecz2* – pobiera wszystkie informacje o meczu z tabeli *mecze* o podanym id,

- *sp\_getMeczTurnieju* – pobiera wszystkie informacje o meczach turniejowych z tabeli *mecze*, które odbywają się w ramach turnieju o podanym id,

- *sp\_getNierozegraneMecze* – pobiera wszystkie informacje o nierozegranych meczach z tabeli *mecze*, w których brał udział dany użytkownik o podanym loginie,

- *sp\_getPunkty* – pobiera przebieg danego meczu o podanym id z tabeli *punkty*,

- *sp\_getRozegraneMecze* – pobiera wszystkie informacje o rozegranych meczach z tabeli *mecze*, w których brał udział dany użytkownik o podanym loginie,

- *sp\_getTurniej* – pobiera wszystkie informacje o danym turnieju o podanym id z tabeli *turnieje*,

- *sp\_getTurniejeIdGracza –* pobiera id tych turniejów z tabeli *turnieje*, w których bierze udział użytkownik o podanym loginie,

- *sp\_getUsers –* pobiera listę wszystkich loginów z tabeli *gracze*,

- *sp\_getWinners* – sporządza listę użytkowników z podaną ilością wygranych przez nich meczy, posortowaną nierosnąco,

- *sp\_newMecz* – tworzy nowy mecz w tabeli mecze dla podanych: *turnieje\_id, gracz1\_id, gracz2\_id,*

- *sp\_newTurniej –* tworzy nowy turniej w tabeli *turnieje* dla podanych: *do\_ilu\_punkty*, *do\_ilu\_sety*, *typ*, *opis*, *nadzorca*,

- *sp\_validateLogin –* pobiera informacje o danym użytkowniku z tabeli *gracze* o podanym loginie.