**TENNIS ORGANIZER**

Piotr Wcisło & Kacper Sarnacki

**Korzystanie z aplikacji**

Logowanie

Po włączeniu aplikacji, mamy dostęp do okna logowania. Mamy dwie opcje:

1. Możemy zalogować się, jeśli posiadamy już konto
2. Możemy zarejestrować nowego użytkownika

W drugim przypadku musimy podać swoje dane.

Podczas rejestracji, musimy uzupełnić krótki formularz, w którym wpisujemy swoje dane osobowe oraz login i hasło. Określany samodzielnie poziom umiejętności oraz dane osobowe potrzebne są do lepszego dopasowania przeciwnika na korcie.

Dane powinny być poprawne – data urodzenia musi być zapisana w narzuconym formacie, podobnie email, hasło musi zawierać do najmniej trzy znaki oraz nie być polem pustym, podobnie jak login.

Aplikacja

Po zalogowaniu się, gracz otrzymuje dostęp do spersonalizowanej strony.

Aplikacja zawiera dwa górne panele. Na pierwszym z nich znajduje się logo oraz elementy umożliwiające zarządzanie bieżącą sesją: zmianę informacji o sobie, zmianę hasła, wylogowywanie się.

Drugi panel składa się z sekwencji przycisków umożliwiających główne funkcjonalności programu:

*PROFIL*

Tutaj gracz ma możliwość przeglądania statystyk o sobie – m.in. najwyższa osiągnięta pozycja w rankingu, ilość wygranych meczy. Ma też możliwość przeglądania zaplanowanych spotkań. Tutaj też wyświetlają się powiadomienia dotyczące spotkań:

* Informujące o wyzwaniu przez innego gracza – możemy je przyjąć lub odrzucić
* Informujące, czy gracz, którego wyzwaliśmy, odrzucił czy przyjął wyzwanie
* Proszące o wpisanie wyniku meczu – gra kończy się po zdobyciu trzech setów przez jednego z graczy. Wynik meczu wpisuje gracz wyzywający.

*WYZWIJ GRACZA*

W tej części gracz ma możliwość wyzwania innego gracza zarejestrowanego w aplikacji. Polami obowiązkowymi są data i godzina spotkania, reszta jest opcjonalna:

* Miasto – z jakiego miasta przeciwników oczekujemy
* Wiek od.. do.. – możemy określić wiek naszego przeciwnika
* Poziom od.. do.. – określamy poziom przeciwnika

Aplikacja dobierze nam przeciwników zgodnie z naszymi wymaganiami. Uwaga zwrócona została na określenie ram czasowych – nasz przeciwnik nie może już mieć zaplanowanego spotkania na ten dzień oraz my także nie możemy mieć w planach żadnego spotkania.

*ZNAJDŹ TRENERA*

Część analogiczna do wyzywania gracza, z tym że tu mamy do dyspozycji listę trenerów. Zdjęte jest z nich ograniczenie jednego meczu na dzień (trener, w przeciwieństwie do zawodnika, nie męczy się po spotkaniu i może po nim poświęcić czas innemu graczowi).

*RANKING*

W tej części można przeglądać ranking wszystkich graczy zarejestrowanych w aplikacji, klasyfikowanych według różnicy ilości meczy wygranych i przegranych.

**Architektura**

Kontrolery

*HomeController*

Ten kontroler zarządza widokami odpowiedzialnymi za proces logowania i rejestracji. Sprawdza poprawność loginu z hasłem, zarządza przekierowaniem do strony prywatnej w przypadku ich poprawności.

*MainController*

Odpowiedzialny jest za zarządzanie wszystkimi funkcjonalności programu. Dostęp do większości z widoków wymaga autoryzacji, stąd metody mają znacznik [Authorized].

*ErrorController*

Oczekuje momentów, gdy użytkownik wprowadzi adres podstrony, która nie istnieje. Wtedy uruchamia widok WrongPage() informujący o nieistnieniu strony.

Modele

*Account*

Odpowiada tabeli Konta w bazie danych. Zawiera informacje o loginie, haśle, odpowiadającym mu graczu. Umożliwia także wykonywanie podstawowych informacji, jak np. zmiana hasła – do tego służy metoda CheckPassword.

*Duel*

Odpowiada tabeli Pojedynki w bazie danych. Zawiera informacje o dacie pojedynku, o tym, czy drugi gracz go zaakceptował oraz o wyniku, jeśli się już odbył.

*Encrypter*

Odpowiada za hashowanie hasła. Używa do tego algorytmu SHA256.

*Mailer*

Po zakończonej powodzeniem rejestracji oraz o każdym meczu jesteśmy informowani mailowo (maile przychodzą na adres podany przez nas podczas rejestracji). Ten model zajmuje się wysyłaniem tych maili.

*Player*

Odpowiada tabeli Gracze w bazie danych. Zawiera podstawowe informacje określające gracza, takie jak jego data urodzenia, poziom umiejętności czy lista meczy, w których wziął udział. Ten model ma także możliwość wyszukiwania przeciwników po konkretnych kryteriach (przeciążona metoda GetOpponentsBy) oraz wyszukiwania konkretnego typu pojedynków (GetFinishedRatedDuels, GetNotFinishedDuels).

*TennisOrganizerContext*

Kontekst bazy danych. Zawiera trzy pola odpowiadające tabelom kont, graczy i pojedynków.

*TennisOrganizerInitializer*

Model dodaje przykładowe rekordy do bazy danych.

*Trainer*

Model dziedziczy po modelu Player. Tym samym ma te same funkcje co gracz. Dodatkowo posiada pole Description, w którym jest zapisana informacja, w której dziedzinie dany trener się specjalizuje.

ViewModele

Żaden z komponentów nie ma bezpośredniego dostępu do bazy danych. Komunikuje się z nią poprzez metody modeli.

*AccountEditorData*

Odpowiada tabeli Konta w bazie danych. Zawiera informacje o loginie, haśle, odpowiadającym mu

*ChallengeCriteria*

Służy do zbierania zestawu informacji, dzięki którym można wyszukać dopasowanych graczy (za pomocą metod GetOpponentsBy). Zawiera też listę tych graczy. Niektóre z pól tej klasy, jak np. wiek dopuszczają wartości nullowe – są to wartości, które nie są wymagane, do wyszukiwania graczy.

*PlayerDuels*

Ułatwia zarządzanie pojedynkami. Konstruktor tej klasy przyjmuje obiekt typu Player oraz Duel. Oferuje dostęp do daty spotkania, przeciwnika podanego w konstruktorze oraz wyniku, jeśli pojedynek się zakończył.

*PlayerStats*

Klasa daje dostęp do informacji o graczu. Jest wykorzystywana przy wyświetlaniu rankingu oraz statystyk.

*TrainingCriteria*

Klasa analogiczna do ChallengeCriteria, z tym że wykorzystywana jest przy wyszukiwaniu trenera.

Widoki (Views)

***HOME***

Wszystkie widoki z tej grupy mają wspólny layout, zawarty w pliku \_Layout.cshtml

*Index*

Strona startowa dla niezalogowanych użytkowników. Do niej także przekierowywany jest użytkownik, gdy bez autoryzacji spróbuje wejść na stronę jej wymagającą (oznaczenie [Authorized]).

*Register*

Strona zawierająca formularz, który następnie przesyła do serwera i po pozytywnej analizie w kontrolerze przekierowywana jest do RegisterSuccess.

*RegisterSuccess*

Jest wyświetlana gdy rejestracja przebiegnie prawidłowo.

***MAIN***

Wszystkie widoki z tej grupy mają wspólny layout, zawarty w pliku \_MainLayout.cshtml

*AccountEdition*

Strona umożliwiająca zmianę loginu i hasła. Zawiera formularz, który przesyła do serwera informacje poprzez klasę AccountEditorData.

*Challenge*

Posiada dwa formularze. Pierwszy Jest wysyłany do serwera po kliknięciu buttona „Szukaj gracza”. Następnie serwer (w kontrolerze MainController) przetwarza informacje z klasy ChallengeCriteria i zwraca listę graczy pasujących do zapytania. Po kliknięciu w „Wyzwij!”, klasa ta zawiera tym razem ID gracza, który został wybrany do pojedynku.

*ChallengeSuccess*

Gracz jest przekierowywany do tej strony, gdy pomyślnie umówi się z graczem

*Profile*

Zawiera informacje z meczami zaplanowanymi przez gracza oraz tymi, w których gracz uczestniczył (włącznie z wynikiem tego spotkania).

*ProfileEdition*

Umożliwia zmianę informacji personalnych o aktualnie zalogowanym graczu.

*Ranking*

Wyświetla ranking graczy. Informacje o nich pobiera przy pomocy klasy PlayerStats

*Statistics*

Strona ze statystykami zalogowanego gracza. Informacje przekazywane są przez klasę PlayerStats.

*Training*

Strona zachowuje się w sposób analogiczny do Challenge, tylko operuje na klasie Trainer zamiast Player.

*TrainingSuccess*

Gracz jest przekierowywany do tej strony, gdy pomyślnie umówi się z trenerem.