Data: 20.05.2018

NTime
Aplikacja do przeliczania wyników
Web API

Dokumentacja Projektu

Autorzy:

Grzegorz Krzysiak
Tomasz Chudzik

Spis treści

Specyfikacja

Opis biznesowy

Aplikacja służy firmie przeprowadzającej pomiary czasu na zawodach sportowych. Dzięki tej aplikacji użytkownicy za pomocą przeglądarki mogą rejestrować się na zawody oraz przeglądać dane odnośnie wszystkich zawodów zorganizowanych przez firmę pomiarową. Ponadto organizator każdych zawodów może założyć sobie specjalne konto, dzięki któremu będzie w stanie edytować dane dla swoich zawodów i księgować wpłaty zawodników.

Wymagania funkcjonalne

1. Zawodnicy mają udostępniony interfejs webowy do przeglądania wszystkich zawodów organizowanych przez firmę pomiarową.
2. Dla każdych zawodów widoczna jest lista zawodników już zapisanych na dane zawody
3. Każdemu użytkownikowi poza podstawowymi danymi osobowymi wyświetla się informacja czy dany zawodnik został już opłacony
4. Jeżeli czas zapisu na zawody jeszcze nie upłynął każdy użytkownik strony może się zapisać na zawody poprzez formularz
5. Formularz zabezpieczony jest walidacją zarówno po stronie przeglądarki jak i po stronie serwera
6. Serwer pozwala na migracje między nowym i starym modelem danych
7. Formularz zabezpieczony jest również przeciwko robotom poprzez Google ReCAPTCHA.
8. Organizatorzy Mają udostępnione specjalne konta, które są przypisane do zawodów, które organizują
9. Podczas zakładania konta przy pomocy maila – adres email jest walidowany czy rzeczywiście istnieje, a konto potwierdzane poprzez link wysłany na adres email posiadacza konta
10. Zalogowanym organizatorom, dla ich zawodów, wyświetla się nowy widok zawodników, w którym mogą księgować wpłaty zawodników
11. API, z którego korzystają organizatorzy musi być zabezpieczone przeciwko niezautoryzowanym użytkownikom
12. Aplikacja przekierowywana jest na konkretną domenę zamawiającego
13. Dane szyfrowane są przy pomocy certyfikatu SSL
14. Strona jest responsywna i dostosowana do różnych rozmiarów ekranu
15. Kontrolki zastosowane w aplikacji mają spójny, ładny design - w tym wypadku oparty o Angular Material.
16. Codziennie tworzone są backupy bazy danych
17. Aplikacja została wdrożona praktycznego użytku i zostać udostępniona użytkownikom
18. Strona musi być w stanie obsłużyć minimum 200 zapisów w ciągu dnia.
19. Aplikacja została przetestowana przy pomocy ponad 80 testów jednostkowych

Wymagania niefunkcjonalne

Baza danych: Microsoft SQL Server Express

ORM: Entity Framework

Testy jednostkowe: NUnit, Jasme, Karma, Anuglar CLI

Interfejs graficzny: Angular, Angular Material

Technologia serwera REST API: .NET Web API

Technologia klienta REST API: Angular HttpClient

Zabezpieczenia autoryzacji i bezpieczeństwa: Oauth2, ASP.NET, Google ReCAPTCHA

Harmonogram projektu

20.12.2017 – Ustalenie wymagań aplikacji z zamawiającym

28.12.2017 – Zaplanowanie interfejsu i funkcjonalności aplikacji

6.12.2017 – Prezentacja pierwszego etapu

15.01.2018 – Wstępny interfejs graficzny aplikacji

20.01.2018 – Obsługa logowania przez użytkowników o różnym poziomie uprawnień

05.01.2018 – Wstępna budowa architektury w Angularze

10.01.2018 – Udostępnienie kontrolerów ASP.NET wraz z zabezpieczeniami autoryzacyjnymi

20.01.2018 – Stworzenie testowej wersji aplikacji

01.02.2018 – Stworzenie podstawowej bazy testów jednostkowych

01.03.2018 – Wdrożenie ReCAPTCHY

01.04.2018 – Wdrożenie podstawowej wersji na serwer testowej

10.04.2018 – Przekierowanie domeny na serwer i wprowadzenie certyfikatu SSL

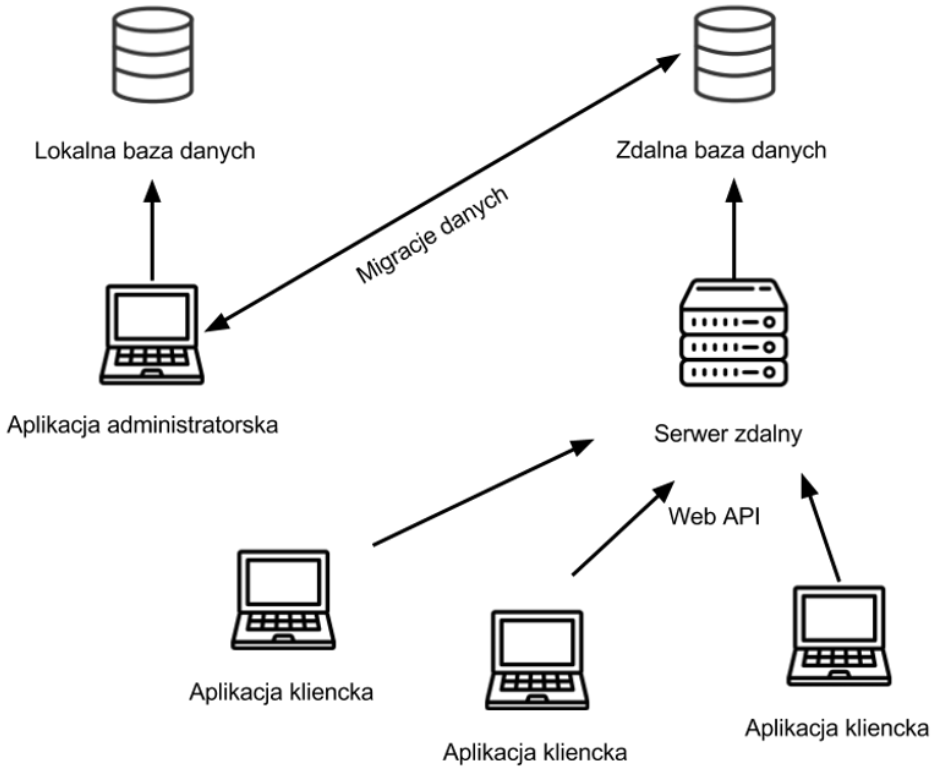
15.04.2018 – Wdrożenie aplikacji na serwer i udostępnienie jej użytkownikom

02.05.2018 – Dodanie edycji wielu użytkowników na raz

20.05.2018 – Oddanie całości aplikacji

Architektura aplikacji

Na poniższym rysunku przedstawiono architekturę platformy NTime z rozbiciem na poszczególne komponenty



Rysunek 1 Architektura NTime

Na przedstawionym rysunku wyróżniamy następujące elementy:

- **Lokalna baza danych** - Local DB, będąca w pogotowiu, by uruchomić jedną z lokalnych, binarnych kopii bazy danych na wypadek utraty połączenia z siecią
- **Zdalna baza danych** – Główna baza danych znajdująca się na serwerze, w które znajdują się wszystkie dane odnośnie przeprowadzonych zawodów oraz kont użytkowników
- **Aplikacja administratorska** – Aplikacja stworzona w pierwszym etapie projektu pozwalająca administratorowi na edycję wszystkich danych dotyczących zawodów oraz na konfigurację i obliczenie wyników w dniu imprezy
- **Serwer zdalny** – Serwer, na którym znajduje się REST API, z którego korzystają wszystkie aplikacje klienckie i który łączy się ze zdalną bazą danych
- **Aplikacja kliencka** – Aplikacja typu SPA wykonana w Angularze 5 z pomocą biblioteki do elementów wizualnych - Angular Material.

Dokumentacja końcowa (powykonawcza)

Wymagania systemowe:

Aplikacja użytkownika:

System Operacyjny Windows 7 lub wyższej,

Platforma .net Framework 4.7,

Dostostęp do internetu

Server:

System Opercayjny Windows Server 2012,

Baza danych MS SQL 2012,

IIS 8.0

Biblioteki wraz z określeniem licencji

nr	Komponent i wersja	Opis	Licencja
1	.Net Framework 4.7	Platforma programistyczna opracowana przez Microsoft, obejmująca środowisko uruchomieniowe (Common Language Runtime – CLR) oraz biblioteki klas dostarczające standardowej funkcjonalności dla aplikacji.	Freeware
2	Entity Framework 7	Narzędzie typu ORM (Object Relational Mapping), pozwalającym odwzorować relacyjną bazę danych za pomocą architektury obiektowej.	Apache License
3	NUnit 3	Platforma do testów jednostkowych	MIT License for 3.0, BSD-style (modified zlib license) for 2.x
4	Angular 5	Otwarty framework i platforma do tworzenia aplikacja SPA, napisany w języku TypeScript i wspierany oraz rozwijany przez Google	MIT
5	Jasmine	Otwarty framework służący do pisania testów jednostkowych dla aplikacji korzystających z JavaScript	MIT
6	Karma	Framework służący do uruchamiana w przeglądarce napisanych testów. Jest zintegrowany z Angular CLI i z korzystanie z Karmy jest niewidoczne dla prgramisty	MIT
7	ASP.NET	Zbiór technologii opartych na frameworku zaprojektowanym przez firmę Microsoft. Przeznaczony jest do budowy różnorodnych aplikacji internetowych, a także aplikacji typu XML Web Services.	Apache License

Wdrożenie projektu

Aplikacja użytkownika:

Produkt nie wymaga instalacji. Potrzebne jest jedynie spełnienie wymagań opisanych w sekcji Wymagania systemowe

Server:

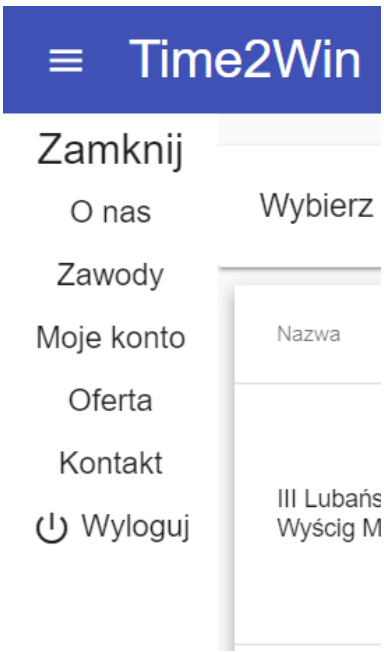
Wymagane jest wdrożenie aplikacji na serwer IIS. Następnie należy skonfigurować connection stringi o nazwach DefaultConnection i NTime. W folderze witryny należy umieścić wygenerowane pliki wdrożenia i folder bundles, który znajduje się w folderze Server projektu i jest generowany przy uruchomienie opcji *Publish.*

Uruchomienie projektu w Visual Studio

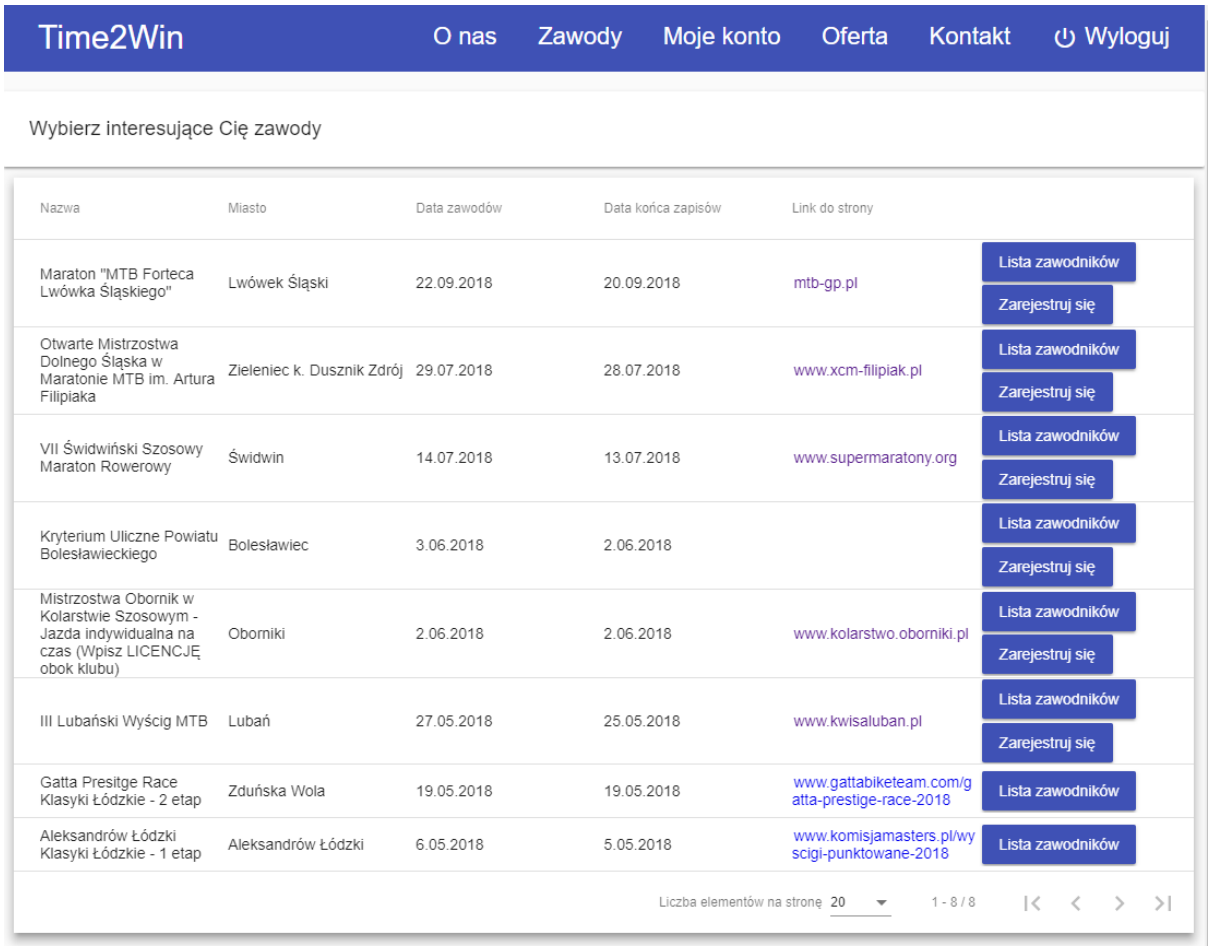
1. Najprościej proces instalacji przeprowadzić na komputerze z systemem Windows 10
2. Pozostałe wymagania: .Net framework 4.7, Visual Studio 2017.
3. Należy rozpakować plik ntime.zip do folderu *ntime*
4. Należy wejść do tego folderu i następnie wejść w następującą ścieżkę chudytom-ntime-bef80e073688\NTime\Server\ntimeAngular
5. W tym kroku konieczne będzie zainstalowanie wszystkich pakietów npm. W tym celu zakładamy, że npm został zainstalowany na komputerze. Można to łatwo sprawdzić włączając wiersz poleceń, np. Powershell i wpisać komendę *npm –version* Jeśli wyświetli się wersja npm, to znaczy, że jest on poprawnie zainstalowany. Jeśli nie, to należy zainstalować pełnego node.js ze strony pod linkiem: https://nodejs.org/en/download/ dla swojej platformy
6. Po upewnieniu się, że npm jest zainstalowany na naszym komputerzem, w folderze do którego weszliśmy w punkcie 4, wpisujemy w wierszu poleceń komendę *npm install.* Pobranych zostanie około 300 MB pakietów potrzebnych do uruchomienia aplikacji
7. W tym momencie można uruchomić aplikację korzystającą z przekierowania na serwer testowy wpisując w wierszu poleceń komendę *npm start*
8. Jeśli chcemy przejść do debugowania w Visual Studio wykonujemy kolejne kroki
9. Wpisujemy komendę *npm run-script build* Pozwoli to w odpowiedni sposób zbudować front aplikacji stworzony w Angularze
10. Następnie cofamy się do ścieżki ntime\chudytom-ntime-bef80e073688\NTime
11. Włączamy plik solucji *NTime.sln* przy pomocy Visual Studio 2017.
12. W zakładce *Solution Explorer* klikamy PPM na projekt *Server* i wybieramy opcję *Set as StartUp Project*
13. Przełączamy budowanie projektu w tryb *Release*
14. Należy upewnić się, że w Visualu jak wejdziemy w widok *SQL Server Object Explorer* w ścieżce SQL Server/(localdb)/MSSQLLocalDB(…)/Databases nie istnieje baza danych *aspnet-Server-20171216125843*
    Jeśli taka baza istnieje należy ją usunąć przed uruchomieniem programu. Ważne, żeby zaznaczyć obydwa checkboxy: *Delete backup (…)* i *Close existing connections.*
15. Przy pomocy klawiszy *ctrl + F5* uruchamiamy projekt

Instrukcja użycia

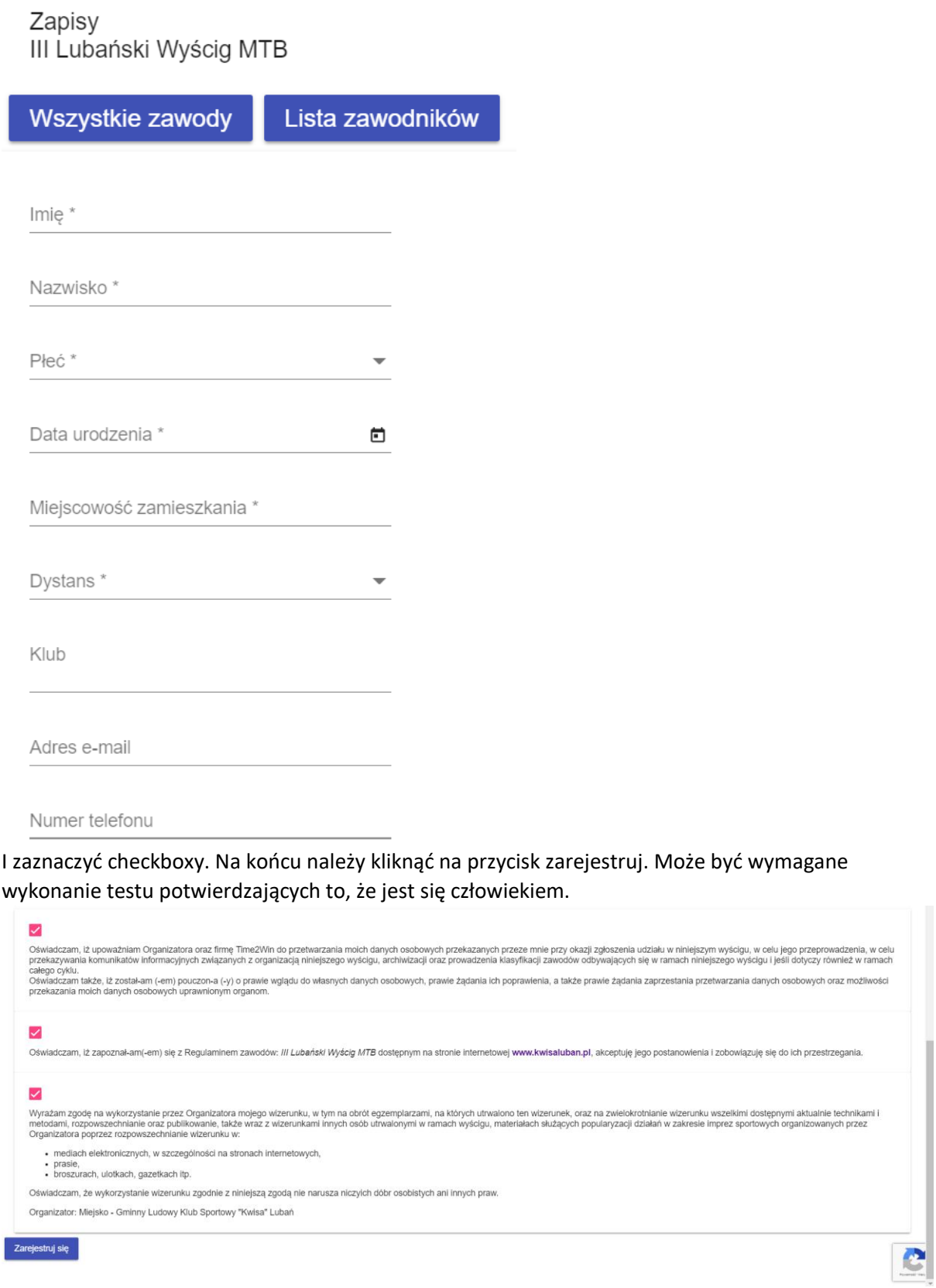
Strona składa się z 5 zakładek:



Po wyborze zakładki "Zawody", można obejrzeć listę dostępnych zawodów, sprawdzić listę zawodników i zarejestrować się na któreś.

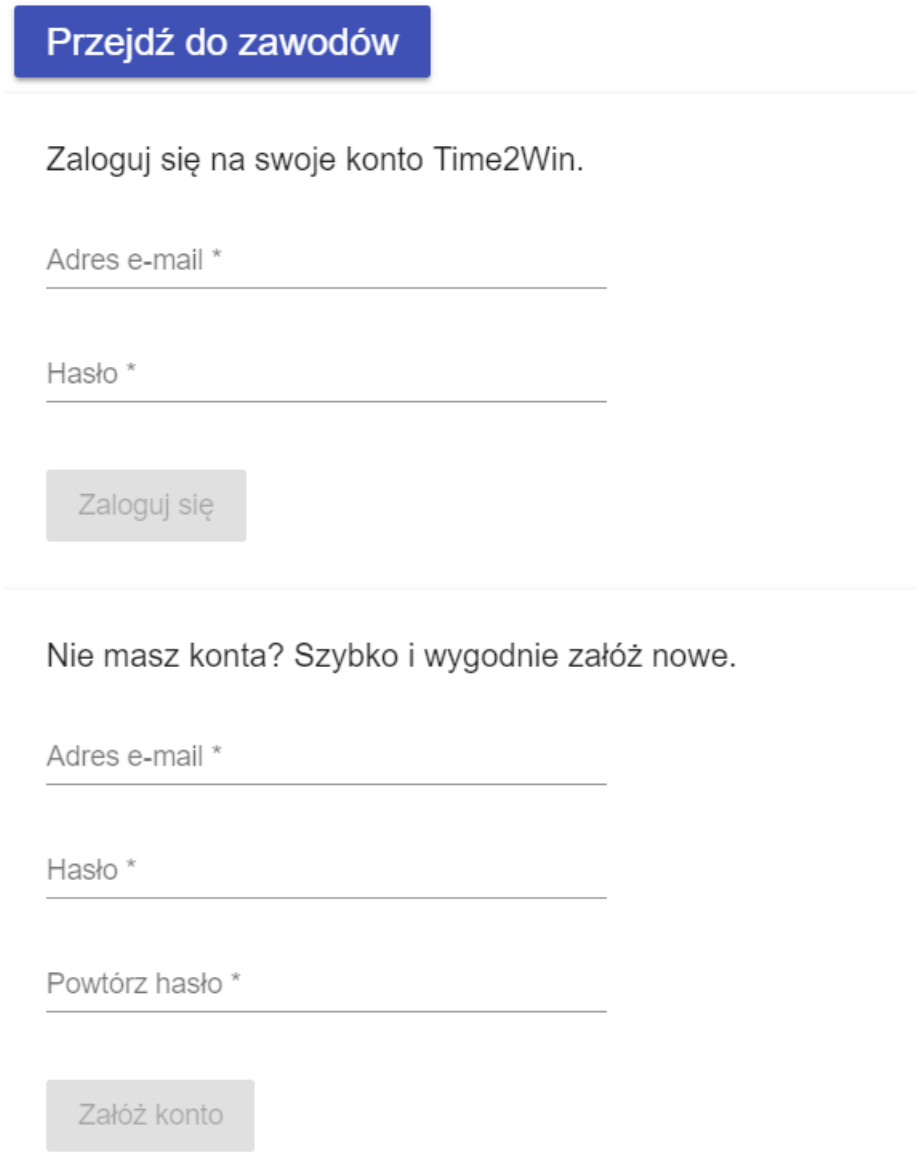


Żeby zarejestrować się na zawody należy uzupełnić następujące pola oznaczone gwiazdkami



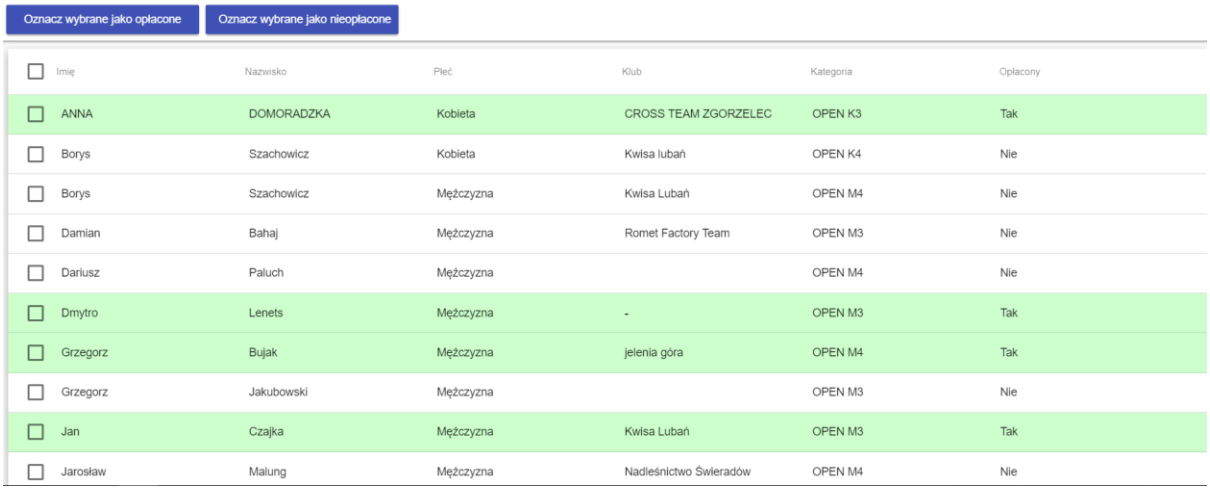
I zaznaczyć checkboxy. Na końcu należy kliknąć na przycisk zarejestruj. Może być wymagane wykonanie testu potwierdzających to, że jest się człowiekiem.

Można się zarejestrować jako organizator zawodów, w tym celu należy skorzystać z formularza znajdującego się na stronie https://www.t2w.pl/konto?role=organizer

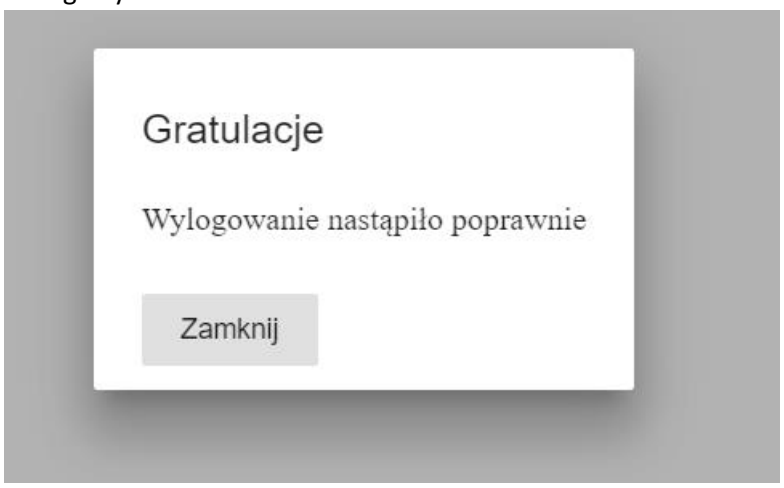


Po pomyślnym zarejestrowaniu można przypisać organizatora do konkretnych zawodów przy pomocy kontrolera POST api/OrganizerAccount/1/SetCompetition/1. Taka operacja może zostać wykonana jedynie przez administratora

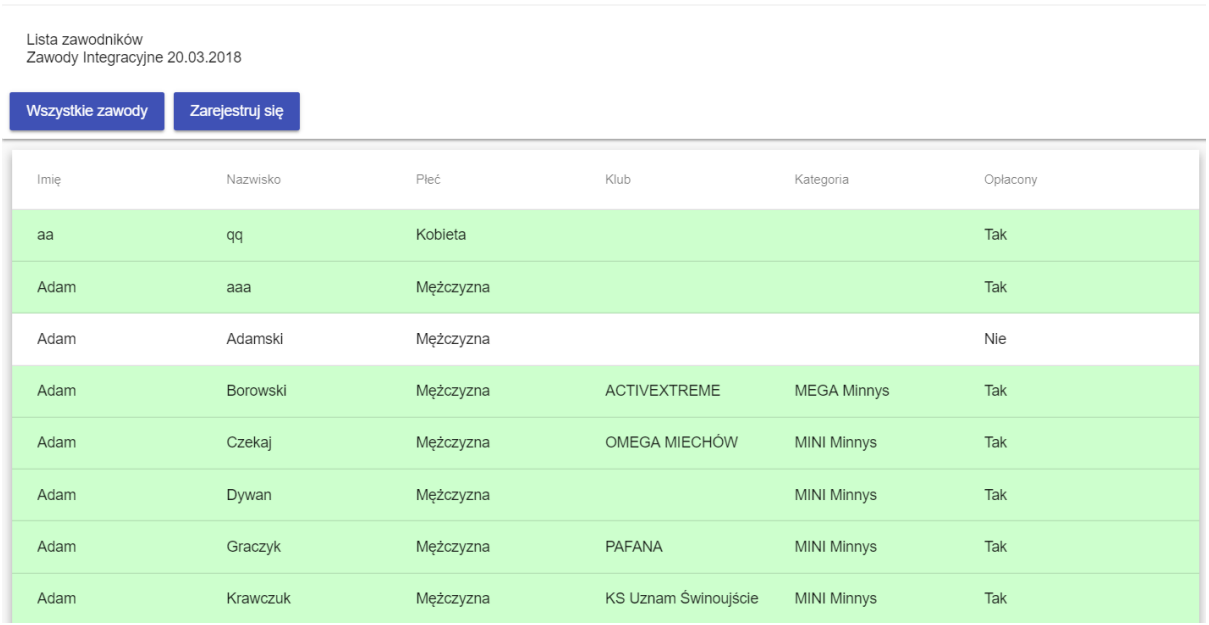
Po wybraniu przypisanych zawodów, organizator może zaznaczyć grupowo zawodników których chce mieć opłaconych. Wyświetli mu się wtedy taki widok:



Po zakończeniu pracy należy wybrać opcję wyloguj. Potwierdzenie pojawi się w oknie dialogowym



Jeśli użytkownik nie ma uprawnień administratora wyświetli mu się inny widok



Żeby móc samemu przetestować działanie strony należy można wejść na stronę www.t2w.pl Jest to jednak wersja produkcyjna i prosilibyśmy się na niej nie rejestrować. Ale listy zawodników są publiczne i jak najbardziej można je obejrzeć. Tak samo jak przetestować pozostałe funkcjonalności aplikacji.

Żeby móc przetestować pełną funkcjonalność aplikacji, możemy wejść na stronę wersji testowej pod adresem: www.testing.time2win.aspnet.pl (Inną możliwością jest wykonanie poleceń 1-7 z działu: *Uruchamianie projektu w Visual Studio,* kończących się *npm start)*. Tutaj znajdują się testowe zawody, dla których jak najbardziej można się zalogować. W dodatku, żeby przetestować możliwość operacji możliwych dla konta organizatora, należy zarejestrować się pod linkiem https://www.testing.time2win.aspnet.pl/konto?role=organizer Następnie przesłać adres e-mail, pod którym zostało założone konto do administratorów, po jeden z poniższych maili
tomasz.chudzik.pl@gmail.com
krzysiak.grzegorz@gmail.com
Podepniemy wtedy konto organizatora do zawodów testowych i po zalogowaniu będzie można przetestować dodatkowe funkcjonalności.

Instrukcja utrzymania

Dane zawodów są przechowywane w bazie danych MS SQL. Do archiwizacji danych można wykorzystać wbudowane mechanizmy tego systemu.

Raport odstępstw od specyfikacji wymagań

W trakcie realizacji projektu wprowadzonych zostało kilka zmian, które powodują, że część funkcjonalności odbiega od tych zawartych w specyfikacji wstępnej. Zmiany te powstawały podczas bieżących konsultacji z klientem. Często były to sugestie od zamawiającego, który zmieniał swoje wymagania. Niemniej jednak nie były to znaczące modyfikacje. Zdarzało się również, że na wniosek autorów projektu, niektóre zmiany funkcjonalności były zgłaszane klientowi i po jego akceptacji uwzględniane podczas wykonywania projektu. Odstępstwa od specyfikacji wymagań zostało zaprezentowane poniżej:

- Poza samymi controllerami do obsługi kont użytkowników oraz do zapisów powstały również controllery obsługujące konta administratora, organizatora oraz moderatora. Dzięki nim istnieje możliwość stworzenia aplikacji webowej, która pozwoli wykonać część takich samych czynności jak w aplikacji desktopowej stworzonej w pierwszym etapie. Funkcjonalności te nie będą raczej realizowane nawet w trakcie trzeciego etapu, niemniej jednak zostaną one wykorzystane podczas tworzenia ostatecznego produktu dla klienta
- W porównaniu ze specyfikacją wstępną zdecydowano, że podczas zapisów to zawodnik sam decyduje na jakim dystansie będzie odbywał zawody i która z dodatkowych informacji będzie dla niego odpowiednia. Możliwość dokonania tego wyboru wprowadzono podczas zapisów i podczas edycji danych dla zawodów, na które zawodnik już się zapisał
-

Dokumentacja utworzonego Web API

Strona: http://projektnet.mini.pw.edu.pl/NTime

Administrator:

Login/Email: admin@ntime.pl

Hasło: admin1

Założenia:

- Są 4 role:
    - Player - P
    - Organizer - O
    - Moderator - M
    - BustModerator – Moderator z odebranymi uprawnieniami
    - Administrator - A
- Organizator ma dostęp do jakiegoś zasobu związanego z zawodami tylko wtedy, gdy jest powiązany z danymi zawodami i może edytować je jak flaga w zawodach "ModeratorEditLock" jest na false
- Player może zapisywać się na zawody i modyfikowac owy wpis o ile operacje te nie są wykonywane po "SignUpEndDate" w obiekcie zawodów
- Player może mieć pełny wgląd tylko na własne wyniki
- Jeśli w ścieżce jest napisane "1" to oznacza, że trzeba tam podać id do danego zasobu (liczba dodatnia, całkowita), chyba że jest poprzedzony przez "/FromCompetition/", "/IntoCompetition/", "/ByCompetition/", "/SetCompetition/" lub "/UnsetCompetition/" wtedy należy podać id do zawodów, a jeśli "/FromPlayerAccount/" to podajemy id do konta zawodnika.
- W przypadku kontrolerów "AdministratorAccount", "ModeratorAccount" id to ciąg znaków
- Api obsługuje Json o ile Content-Type jest nie wskazuje inaczej
- Jeśli adres kończy się na "?ItemsOnPage=10&PageNumber=0" to oznacza, że wynik jest stronicowany, w odpowiedzi dostanie się całkowitą liczbę obiektów i tablicę przeznaczoną do wyświetlenia na wskazanej stronie (Uwaga: strony liczymy od zera). Zwracana jest struktura z polem TotalCount – int i Items – tablica.
- Hasła powinny zawierać co najmniej 8 znaków w tym co najmniej jedną cyfrę i mały znak
- Można ignorować pole id przy tworzeniu nowych zasobów, ale nie można go pominąć, może być dowolne a najlepiej –1
- Jeśli przy polu w wypisanej strukturze danych jest "*" to pole nie może przyjąć null-a, więc nie może być pominięte
- 404 jeśli id-ki się nie zgadzają
- Czas liczymy w sekundach jako zmienna typu decimal (oznaczone przy niektórych zmiennych)

Logowanie:

POST Token

Mogą: wszyscy

Content-Type: application/x-www-form-urlencoded

Body: grant_type=password&username=admin&password=admin1

Przykładowa odpowiedź:

{

"access_token": "BMxniHiUjhexqWm1RwexXkAK9Pq70n4-QZNdjRgle1gL15a1mrGLM3lHhjq0DOCaWsGh0T2Nr9KWqStfIQuPxD8iYDDYDKzRbU2Rv_sXPeEbBDPpvgAVsOnL9aiWZsG2ve811uGBiH0l0Nfko50uQDQcMAuHAbBQjXuwLChO-up5DTLgrHozIlc8han4U-t1nDWeiVd1vPqPcaJi6TvFm6BfNGjZhrm8G2xAoWF2C60894Qjs-VqWJlER9eSBLk6B6TfPFHrEpwHixf5olC2DQjpv1Jd-3PFNu2RQTnlbqMCl5pjJXasYGuzPGWW5n8J58D-tC7lxOYXqIKZTHfGKzAE2oYsN8kMxmC0QCc0tqgnBJvbGiJ-w0t8JktmicFKpJR0zUlC3YLQryPx-HvcvLnaIwXpenVATYosSDoCmvfT6kpy4Sg34qQYYw7dKf8RvJraxbnlqYmJqrZ6L9uD9oQ6CHHAj7Khsfah6KXSSexWZvL4S3vE20-TEIfMiC09UaSzANhDaYXIAMB7_gJo0g",

"token_type": "bearer",

"expires_in": 1209599,

"userName": "admin",

".issued": "Thu, 28 Dec 2017 20:42:00 GMT",

".expires": "Thu, 11 Jan 2018 20:42:00 GMT"

}

W każdym następnym żądaniu należy umieścić w nagłówku:

Authorization: Bearer BMxniHiUjhexqWm1Rwe....

Kontroler Account:

POST api/Account/Logout

Mogą: wszyscy zalogowani

POST api/Account/ChangePassword

Mogą: wszyscy zalogowani

Body:

- *OldPassword
- *NewPassword
- *ConfirmPassword

POST api/Account/Register

Mogą: wszyscy

Body:

- *Email
- *Password
- *ConfirmPassword

GET api/Account/Role

Mogą: wszyscy zalogowani

Zwraca id konta (string), email i przypisaną role

Kontroler Competition

Model danych

- *Id
- *Name
- *EventDate
- SignUpEndDate - jeśli null to nie ma deadline-u na zapisy
- Description
- Link
- Organizer - string
- City
- *OrganizerEditLock – nie da się zmodyfikować to poprzez PUT i POST (przy tworzeniu), po stworzeniu nowych zawodów ustawiony jest na false

GET /api/Competition?ItemsOnPage=10&PageNumber=0

Mogą: wszyscy

Pobiera stronicowaną listę zawodów sortowaną po czasie rozpoczęcia

GET /api/Competition/FromPlayerAccount/1?ItemsOnPage=10&PageNumber=0

Mogą: A, M, P

Pobiera stronicowaną listę zawodów, na które dany użytkownik jest zapisany.

GET /api/Competition/1

Mogą: wszyscy

Pobiera dane jednych zawodów

PUT /api/Competition/1

Mogą: A, O

Modyfikuje zawody

POST /api/Competition

Mogą: A

Tworzy nowe zawody

POST /api/Competition/1/OrganizerLock/true

POST /api/Competition/1/OrganizerLock/false

Mogą: A

Nic nie trzeba podawać nic w body

Ustawia flagę OrganizerEditLock

Kontrolery AgeCategory, Distance, ExtraPlayerInfo

W miejsce "<attr>" wstaw AgeCategory, Distance lub ExtraPlayerInfo

Model danych AgeCategory

- *Id
- *Name
- *YearFrom

- *YearTo

Model danych Distance

- *Id
- *Name
- *Length
- *DistanceTypeId – int: może przyjąć
    - 0 – na dystans
    - 1 – na okrążenia - wtedy LapsCunt jest istotne
    - 2 – na czas w pętli - wtedy TimeLimit jest istotne
- *LapsCount
- *TimeLimit - decimal

Model danych ExtraPlayerInfo

- *Id
- *Name
- *ShortName

GET api/<attr>/FromCompetition/1

Mogą: wszyscy

Wylistuje dane z danych zawodów

GET api/<attr>/1

Mogą: wszyscy

Pobiera obiekt o podanym id

PUT api/<attr>/1

Mogą: A, O

Modyfikuje obiekt o podanym id

POST api/<attr>/IntoCompetition/1

Mogą: A, O

Tworzy nowy obiekt w danych zawodach

DELETE api/<attr>/1

Usuwa obiekt o podanym id

Mogą: A

Kontroler Player

Dotyczy "wpisu" na konkretne zawody i ich ewentualne wyniki, a nie konta zawodników

Model danych filtru

Jeśli jest nullem to filtr nie jest brany pod uwagę

- *PlayerSort - int
    - 0 - imię
    - 1 – nazwisko
    - 2 – klub

    - 3 – numer startowy
    - 4 – czas startu
    - 5 – kategoria
    - 6 – data urodzenia
- *DescendingSort - bool
- Query – string, zaawansowane wyszukiwanie po napisie
    - "30 - 50" wybierze numery startowe z przedziału
    - "30, 40, 45, 50" wybierze podane numery startowe
    - "jan kow MINI" wybierze wpisy, w których zawierają się podane po spacji selektory np. Jan Kowalski z wystansu MINI i Jan Kowalik z wystansu MINI+
- Men – bool
- WithoutStartTime - bool
- Invalid - bool
- CompleatedCompetition - bool
- HasVoid - bool
- Distance – id
- AgeCategory – id
- ExtraPlayerInfo – id

Model danych do publicznej listy (tylko do wglądu)

- *Id
- *FirstName
- *LastName
- *IsMale - bool
- Team - string
- *StartNumber - int
- StartTime – decimal (liczone od godziny 00:00)
- FullCategory - string

Model danych do zapisów

- *Id
- *FirstName
- *LastName
- *BirthDate
- *IsMale
- Team
- *ExtraPlayerInfoId - id
- *DistanceId - id
- *CompetitionId -id

Model danych pełny

- *Id
- *FirstName
- *LastName
- *BirthDate
- *IsMale
- Team

- *StartNumber
- StartTime – decimal (liczone od godziny 00:00)
- *IsStartTimeFromReader - być może zostanie usunięte
- FullCategory – aktualizuje się automatycznie
- *LapsCount
- *Time - decimal
- *DistancePlaceNumber
- *CategoryPlaceNumber
- *CompetitionCompleted - bool
- ExtraPlayerInfoId - id
- DistanceId - id
- CompetitionId -i d
- PlayerAccountId - id

POST api/Player/TakeSimpleList/FromCompetition/1?ItemsOnPage=10&PageNumber=0

Mogą: wszyscy

W body podać obiekt filtru

Zwraca dane publicznej listy

POST api/Player/TakeFullList/FromCompetition/1?ItemsOnPage=10&PageNumber=0

Mogą: A, O, M

W body podać obiekt filtru

Zwraca pełne dane

GET api/Player/TakeFullList/FromPlayerAccount/1?ItemsOnPage=10&PageNumber=0

Mogą: A, M, P

Zwraca zapisy (pełne dane) z odbytych zawodów i zawodów na które się zapisało podanego zawodnika

GET api/Player/FromPlayerAccount/1/FromCompetition/1

Mogą: A, M, P

Zwraca pełne dane zarejestrowane użytkownika dla wybranych zawodów

GET api/Player/1

Mogą: A, O, M, P

Zwraca pełne dane o podanym id

PUT api/Player/1

Mogą: A, O, M

Modyfikuje pełne dane

DELETE api/Player/1

Mogą: A, O, M, P

Kasuje zapis

GET api/Player/Register/1

Mogą: A, O, M, P

Pobiera zapis w o podanym id

PUT api/Player/Register/1

Mogą: P

Modyfikuje zapis

POST api/Player/Register/IntoCompetition/1

Mogą: wszyscy

Zapisuje się na dane zawody

Kontroler PlayerAccount

Model Danych

- *Id
- *FirstName
- *LastName
- *BirthDate
- *IsMale
- Team
- PhoneNumber
- *EMail

GET api/PlayerAccount/Search/Filtr?ItemsOnPage=10&PageNumber=0

GET api/PlayerAccount/Search?ItemsOnPage=10&PageNumber=0

Może: A

Zwraca stronicowaną listę kont zawodników z lub bez filtru ("Filtr") który jest stringiem

GET api/PlayerAccount

Może: P

Zwraca dane aktualnie zalogowanego zawodnika

GET api/PlayerAccount/1

Może: A, P

Zwraca dane zawodnika o podanym id

GET api/PlayerAccount/FromCompetition/1?ItemsOnPage=10&PageNumber=0

Może: A

Zwraca listę kont zawodników z których zapisano się na konkretne zawody

PUT api/PlayerAccount/1

Może: A, P

Edycja

Kontroler OrganizerAccount

Wszystkie operacje może tylko administrator

Model Danych

- *Id
- *FirstName
- *LastName
- *PhoneNumber - string
- *EMail
- CompetitionDtos – tablica zawodów do których jest przypisanych (model danych zawodów), przy wylistowaniu null, pole pomijane przy dodawaniu i modyfikacjach

GET api/OrganizerAccount/Search/Filtr?ItemsOnPage=10&PageNumber=0

GET api/OrganizerAccount/Search?ItemsOnPage=10&PageNumber=0

Zwraca stronicowaną listę organizatorów z lub bez filtru ("Filtr") który jest stringiem

GET api/OrganizerAccount/ByCompetition/1

Zwraca organizatorów przypisanych do danych zawodów

GET api/OrganizerAccount/1

Pojedynczy organizator o podanym id, z polem CompetitionDtos

PUT api/OrganizerAccount/1

Edycja

POST api/OrganizerAccount

Stworzenie nowego organizatora z losowym hasłem. Hasło nie jest zwracane.

DELETE api/OrganizerAccount/1

Usunięcie organizatora

POST api/OrganizerAccount/1/PasswordReset

Reset hasła. Nowe, losowe hasło jest zwracane.

POST api/OrganizerAccount/1/SetCompetition/1

Ustawia organizatorowi o podanym id dostęp do zawodów o podanym id

POST api/OrganizerAccount/1/UnsetCompetition/1

Analogicznie

Kontroler ModeratorAccount

Wszystkie operacje może tylko administrator

Model danych taki jak w akcji "Role"

Id to string, (w prototypach oznaczony jako "aaa")

GET api/ModeratorAccount

Pobiera listę

POST api/ModeratorAccount

Tworzy nowe kont,

W body to samo jak przy rejestracji

DELETE api/ModeratorAccount/aaa

Usuwa

POST api/ModeratorAccount/aaa/PasswordReset

Zmienia hasło na losowe i je zwraca

POST api/ModeratorAccount/aaa/Bust

Zmienia role na "BustModerator"

POST api/ModeratorAccount/aaa/Unbust

Analogicznie

Kontroler AdministratorAccount

Wszystkie operacje może tylko administrator

Analogicznie jak w Moderatorze

GET api/AdministratorAccount

POST api/AdministratorAccount

DELETE api/AdministratorAccount/aaa

POST api/AdministratorAccount/aaa/PasswordReset

Dokumentacja końcowa (powykonawcza) - punkty wymagane przez prowadzącego zajęcia

Model danych