DOKUMENTACJA PROJEKTU – NAUKA SZYBKIEGO PISANIA NA KLAWIATURZE

DOMINIK WIŚNIEWSKI | INFORMATYKA STOSOWANA V SEMESTR | INDEKS 113131

MICHAEL SZYCHULSKI | INFORMATYKA STOSOWANA V SEMESTR | INDEKS 113115

## Główne założenia realizacji projektu z wykorzystaniem języka C#:

Zamysłem projektowym aplikacji było utworzenie w pełni działającej aplikacji webowej za wykorzystaniem platformy ASP.NET. Dzięki temu rozwiązaniu, wytworzony został szkielet aplikacji backendowej, obsługiwanej w pełni przez język C#, i aplikacja kliencka wykonana w Javascript z pomocą frameworku React.  
Aplikacja startowo jest na porcie 44306 z protokołem https i 50400 z protokołem http.

Nauka szybkiego pisania na klawiaturze odbywa się poprzez zjawisko „grywalizacji”. Nasz projekt zakłada, że chęć uzyskiwania coraz lepszych wyników w tabeli rankingowych, będzie motywatorem do dalszego pisania tekstów i tym samym nauki szybkiego pisania.

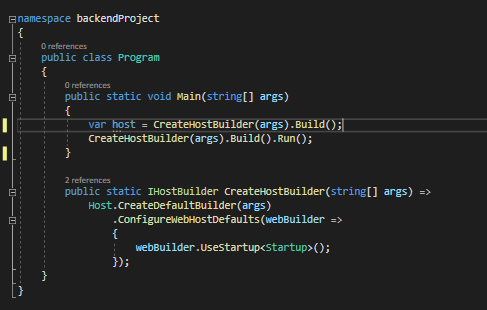
Na podstawie projektu sprawdziliśmy działanie współpracy backendu C# poprzez ASP.NET oraz Entity Framework do zarządzania połączeniem z warstwą frontendu i wysyłaniem danych pomiędzy nimi i przekazywaniem ich do bazy danych.

## Opis dostępnego API obsługiwanego przez backend w C#:

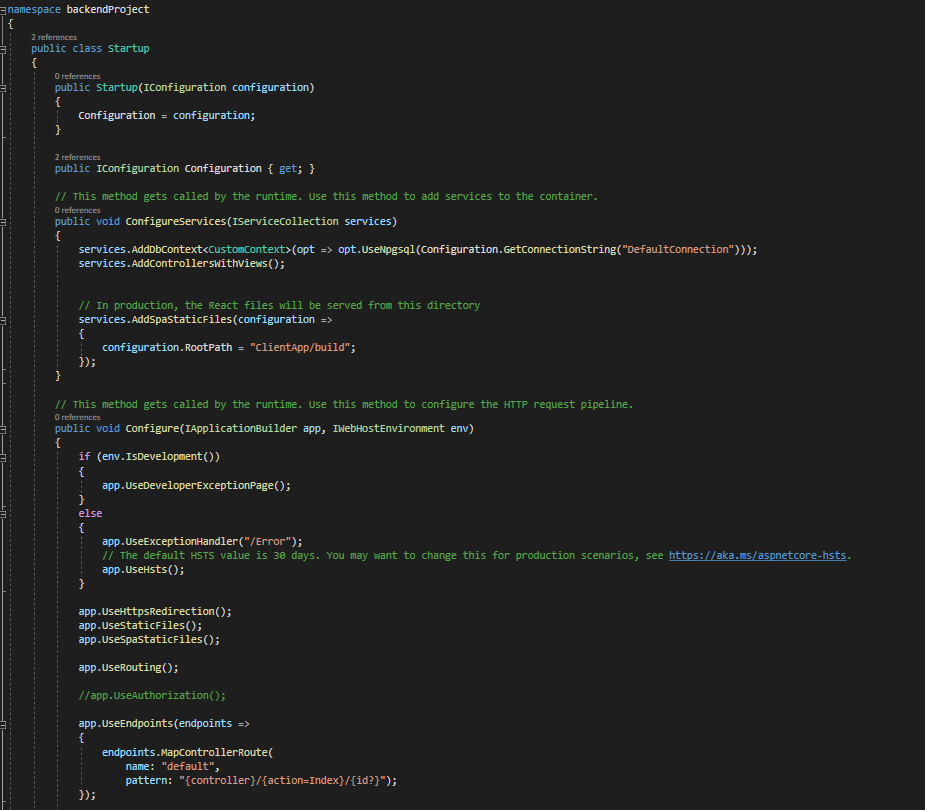
|  |  |  |
| --- | --- | --- |
| Składania zapytania: http://{localhost}:{port}/{model}/{x} Preambuła : http://{localhost}:{port}/ | Opis | Typ wyniku |
| Typ metody: GET  /Accounts | Pobranie wszystkich dostępnych kont | {{  "id": 1,  "username": "testowy",  "password": "test",  "last\_logged": "2021-01-12T18:30:10.47843",  "created": "2021-01-12T18:30:10.480785"  },{  "id": 2,  "username": "pierwszy",  "password": "TEST",  "last\_logged": "2021-01-12T18:58:11.251023",  "created": "2021-01-12T18:58:11.2531"  }} |
| Typ metody: GET  / Accounts /find?id={id} | Pobranie konta po jego numerze id | {  "id": 1,  "username": "testowy",  "password": "test",  "last\_logged": "2021-01-12T18:30:10.47843",  "created": "2021-01-12T18:30:10.480785"  } |
| Typ metody: POST  / Accounts /login  Request body:  {“username”:{username},  “password”:{password}} | Sprawdzenie czy dany użytkownik istnieje i czy jego hasło jest poprawne | Response 200 OK  Response 404 Not Found |
| Typ metody: POST  / Accounts /add  Request body:  {“username”:{username},  “password”:{password}} | Dodanie nowe konta użytkownika do serwisu | Response 201 Created  Response 404 Not Found |
| Typ metody: DELETE  / Accounts /delete?id={id} | Usuwa użytkownika o podanym ID | Response 200 OK  Response 404 Not Found |
| Typ metody: PUT  / Accounts /update?id={id}  Request body:  {“username”:{username},  “password”:{password}} | Zmienia Username i Password użytkownika o podanym ID | Response 200 OK  Response 404 Not Found |
| Typ metody: PUT  /Accounts/lastlogged | Aktualizuje czas ostatniego zalogowania użytkownika o podanym id na obecny | Response 200 OK  Response 404 Not Found |
| Typ metody: GET  /results/player?accId={accId} | Pobranie wszystkich wyników dla gracza o podanym id | {  "id": 2,  "game": null,  "wordSpeed": 10,  "account": {  "id": 3,  "username": "drugi",  "password": "test",  "last\_logged": "2021-01-12T18:59:22.412457",  "created": "2021-01-12T18:59:22.412477"  },  "finish\_date": "2021-01-16T17:23:37.327277",  "isPassed": false,  "mistakes": 1  },  {  "id": 3,  "game": null,  "wordSpeed": 107.33,  "account": {  "id": 3,  "username": "drugi",  "password": "test",  "last\_logged": "2021-01-12T18:59:22.412457",  "created": "2021-01-12T18:59:22.412477"  },  "finish\_date": "2021-01-16T18:04:18.098837",  "isPassed": true,  "mistakes": 0  }, |
| Typ metody: GET  /games/find?id={id} | Pobranie gry o danym id | {  "id": 2,  "gameName": "tryme",  "textToWrite": {  "id": 2,  "text": "to jest tekst ktory musimy wrzucic do bazy danych ze spacjami.",  "source": "moja glowa",  "wordCount": 11,  "topSpeed": 0,  "averageSpeed": 0  },  "wordCount": 0,  "minWordspeed": 20,  "maxMistakes": 1,  "difficulty": 0  } |
| Typ metody: POST  /results/add  Request body:  {“gameID”:{ gameID },  “username”:{username},  “wordSpeed”:{ wordSpeed },  “mistakes”:{ mistakes}} | Utworzenie wyniku użytkownika o podanym username, dla danego ID gry, wysyłane są statystyki, czyli prędkość wpisywanych słów oraz liczba pomyłek | Response 201 Created  Response 404 Not Found |
| Typ metody: GET  /results | Pobranie wszystkich wyników gier dla wszystkich użytkowników. | {  "id": 10,  "game": {  "id": 2,  "gameName": "tryme",  "textToWrite": null,  "wordCount": 0,  "minWordspeed": 20,  "maxMistakes": 1,  "difficulty": 0  },  "wordSpeed": 87.93,  "account": {  "id": 3,  "username": "drugi",  "password": "test",  "last\_logged": "2021-01-12T18:59:22.412457",  "created": "2021-01-12T18:59:22.412477"  },  "finish\_date": "2021-01-17T01:13:20.739607",  "isPassed": true,  "mistakes": 0  },  {  "id": 11,  "game": {  "id": 8,  "gameName": "Lorem welcome",  "textToWrite": null,  "wordCount": 0,  "minWordspeed": 20,  "maxMistakes": 6,  "difficulty": 0  },  "wordSpeed": 44.58,  "account": {  "id": 3,  "username": "drugi",  "password": "test",  "last\_logged": "2021-01-12T18:59:22.412457",  "created": "2021-01-12T18:59:22.412477"  },  "finish\_date": "2021-01-17T01:38:25.282066",  "isPassed": true,  "mistakes": 0  } |
| Typ metody: POST  results/player/criteria  {“accountID”:{ accountID },  “isPassed”:{ isPassed },  “startDate”:{ startDate },  “endDate”:{ endDate }} | Pobranie wszystkich wyników spełniających kryteria podane w request body | {JSON} |
| Typ metody: DELETE  /results/delete?id={id} | Usuń wynik o podanym id | Response 200 OK  Response 404 Not Found |
| Typ metody: GET  /results/scoretable | Pobranie wszystkich wyników w formacie dostosowanym jako wiersze tabeli wyników | {  "id": 12,  "game": "Lorem welcome",  "wordSpeed": 54.64,  "account": "drugi",  "finish\_date": "2021-01-17T13:03:31.156853",  "isPassed": true,  "mistakes": 0  },  {  "id": 11,  "game": "Lorem welcome",  "wordSpeed": 44.58,  "account": "drugi",  "finish\_date": "2021-01-17T01:38:25.282066",  "isPassed": true,  "mistakes": 0  },  Response 404 Not Found |
| Typ metody: GET  /results/scoretable/player?accId={accId} | Pobranie wszystkich wyników dla danego użytkownika w formacie dostosowanym jako wiersze tabeli wyników | {{  "id": 1,  "game": "testGame",  "wordSpeed": 12,  "account": "pierwszy",  "finish\_date": "2021-01-12T23:48:39.281193",  "isPassed": false,  "mistakes": 0  }  Response 404 Not Found |
| Typ metody: GET  /results/scoretable/game?gameId={gameId} | Pobranie wszystkich wyników dla danej gry w formacie dostosowanym jako wiersze tabeli wyników | {  "id": 10,  "game": "tryme",  "wordSpeed": 87.93,  "account": "drugi",  "finish\_date": "2021-01-17T01:13:20.739607",  "isPassed": true,  "mistakes": 0  },  {  "id": 9,  "game": "tryme",  "wordSpeed": 53.86,  "account": "drugi",  "finish\_date": "2021-01-17T01:13:08.444502",  "isPassed": true,  "mistakes": 0  },  Response 404 Not Found |
| Typ metody: GET  /games | Pobranie listy wszystkich dostępnych obiektów gier wraz z powiązanymi obiektami teksów, który zawiera dodatkowe informacje | {  "id": 1,  "gameName": "testGame",  "textToWrite": {  "id": 1,  "text": "testowy\_tekst\_do\_gry\_bo\_tak\_trzeba",  "source": "ja",  "wordCount": 1,  "topSpeed": 0,  "averageSpeed": 0  },  "wordCount": 0,  "minWordspeed": 20,  "maxMistakes": 0,  "difficulty": 0  },  {  "id": 7,  "gameName": "tryme7",  "textToWrite": {  "id": 2,  "text": "to jest tekst ktory musimy wrzucic do bazy danych ze spacjami.",  "source": "moja glowa",  "wordCount": 11,  "topSpeed": 0,  "averageSpeed": 0  },  "wordCount": 11,  "minWordspeed": 60,  "maxMistakes": 0,  "difficulty": 2  }  } |
| Typ metody: POST  /games/add  Request body:  {“gameName”:{ gameName },  “textToWrite”:{ textToWrite },  “difficulty”:{ difficulty }} | Dodanie gry o podanych parametrach do bazy | Response 200 OK  Response 404 Not Found |
| Typ metody: GET  /games/find?id={id} | Pobranie gry o podanym id | {  "id": 2,  "gameName": "tryme",  "textToWrite": {  "id": 2,  "text": "to jest tekst ktory musimy wrzucic do bazy danych ze spacjami.",  "source": "moja glowa",  "wordCount": 11,  "topSpeed": 0,  "averageSpeed": 0  },  "wordCount": 0,  "minWordspeed": 20,  "maxMistakes": 1,  "difficulty": 0  } |
| Typ metody: DELETE  /games/delete?id={id} | Usunięcie gry o podanym id | Response 200 OK  Response 404 Not Found |
| Typ metody: PUT  /games/update?id={id}  Request body:  {“gameName”:{ gameName },  “textToWrite”:{ textToWrite },  “difficulty”:{ difficulty }} | Zaktualizowanie danych gry o podanym id według danych podanych w request body | Response 200 OK  Response 404 Not Found |
| Typ metody: GET  /writingtexts | Pobranie wszystkich zapisanych tekstów | {  "id": 2,  "text": "to jest tekst ktory musimy wrzucic do bazy danych ze spacjami.",  "source": "moja glowa",  "wordCount": 11,  "topSpeed": 0,  "averageSpeed": 0  },  {  "id": 3,  "text": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.  "source": "lorem ipsum",  "wordCount": 69,  "topSpeed": 0,  "averageSpeed": 0  } |
| Typ metody: POST  /writingtexts/add  Request body:  {“text”:{ text },  “source”:{source}} | Dodanie tekstu z podanymi parametrami | Response 200 OK  Response 404 Not Found |
| Typ metody: GET  /writingtexts/find?id={id} | Pobranie tekstu o podanym id | {  "id": 2,  "text": "to jest tekst ktory musimy wrzucic do bazy danych ze spacjami.",  "source": "moja glowa",  "wordCount": 11,  "topSpeed": 0,  "averageSpeed": 0  } |
| Typ metody: DELETE  /writingtexts/delete?id={d} | Usunięcie tekstu o podanym id | Response 200 OK  Response 404 Not Found |
| Typ metody: PUT  / writingtexts /update?id={id}  Request body:  {“text”:{ text },  “source”:{source}} | Zaktualizowanie tekstu o podanym id według podanych parametrów | Response 200 OK  Response 404 Not Found |
| Typ metody: PUT  / writingtexts /updateSpeeds?id={id} | Zaktualizowanie TopSpeed I AvarageSpeed dla tekstu o podanym id | Response 200 OK  Response 404 Not Found |
| Typ metody: GET  /custom/get/{words} | Pobranie {words} losowych słów, gdzie {words} to liczba słów które chcemy otrzymać | [  "determiner",  "descendible",  "brusk",  "mournings",  "calderas",  "claystones",  "macks",  "ganglion",  "ruffled",  "swearword"  ] |

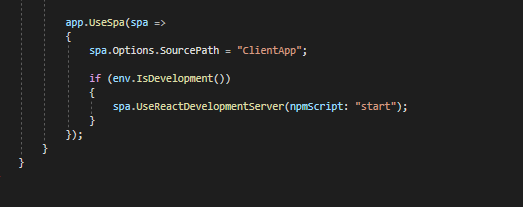
## Opis kodu całej aplikacji ASP.NET:

Klasa Program i metoda Main. Inicjalizuje ona cały rozruch aplikacji. Uruchamiana jest od razu aplikacja po stronie backendu i aplikacja klienta, dostępne pod localhostem dla protokołów http/https o określonych portach na samym początku dokumentacji.



Klasa Startup. Odpowiada ona za całą konfigurację backendu aplikacji i ustanawia połączenie z bazą danych.

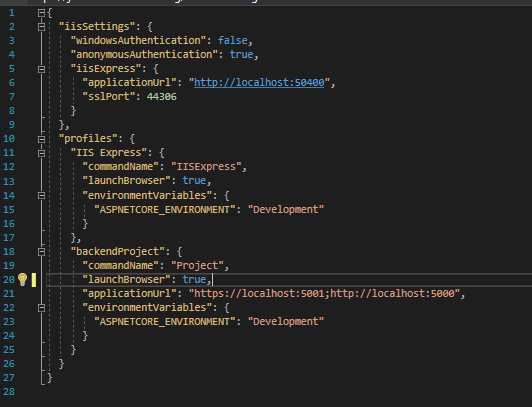




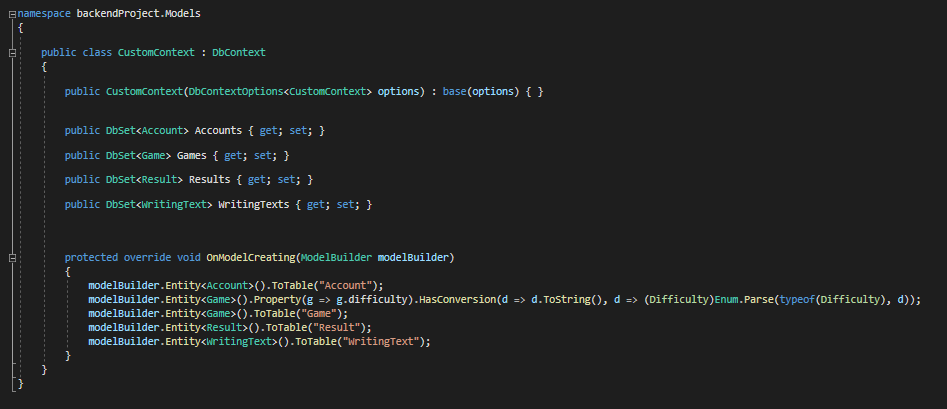
Tworzony jest Context, który odpowiada za połączenie z bazą danych dzięki metodzie:

services.AddDbContext<CustomContext>(opt => opt.UseNpgsql(Configuration.GetConnectionString("DefaultConnection")));

Poprzez pakiety NuGet – EntityFrameworkCore i Npsql dla PostgreSQL – Context utworzony dla połączenia podanego w ustawieniach „DefaultConnection”:



Służy on do zarządzania zmianami w bazie danych ze strony serwisu. Tworzone są obiekty DbSet<T>, który każdy służy do reprezentowania stanu powiązanej tabeli w bazie danych. Poniżej klasa CustomContext wykorzystana w projekcie. Każda tabela ma swój odpowiednik DbSet. Metoda OnModelCreating zapewnia utworzenie tabel oraz ich ustawień.

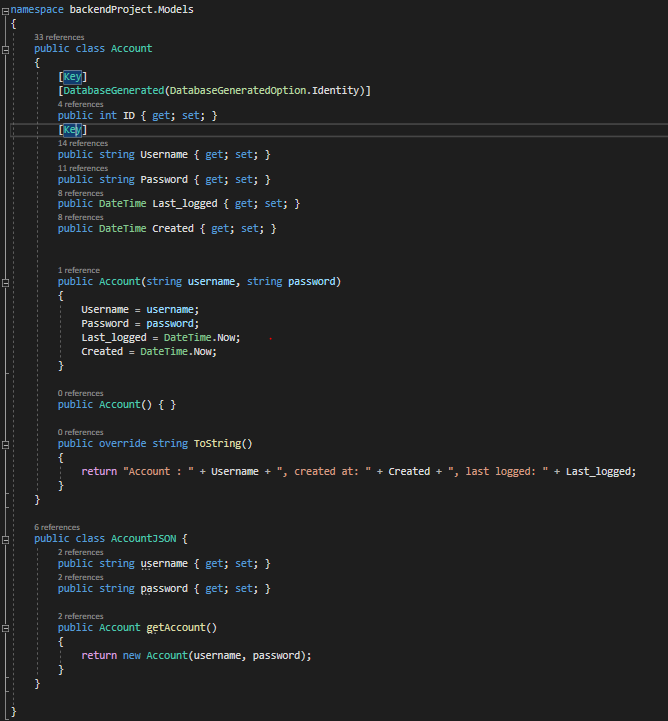


## Opis wykorzystywanych modeli i ich metod:

Klasa modelu Account. Jej pola to:

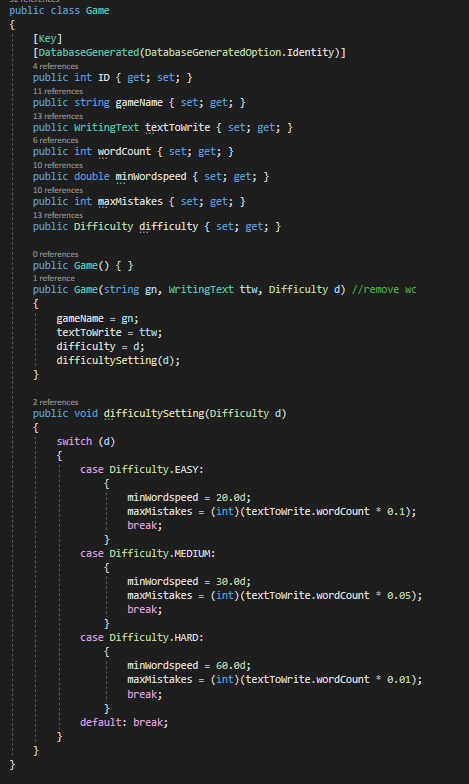
* ID – Unikalny klucz główny modelu, automatycznie generowany dla każdego kolejnego zapisanego w bazie obiektu
* Username – Unikalna nazwa użytkownika
* Password – Hasło za pomocą którego użytkownik loguje się do aplikacji
* Last\_logged – Data ostatniego zalogowania się użytkownika
* Created – Data utworzenia konta użytkownika

Posiada ona podklasę AccountJSON utworzoną w celu łatwiejszego serializowania i deserializowania danych z i do frontu aplikacji.



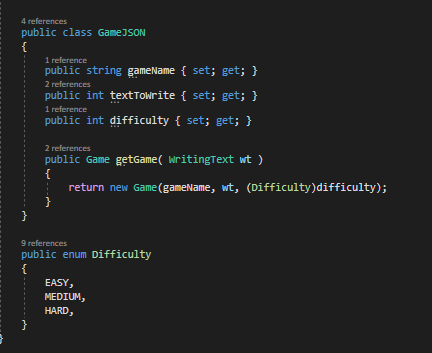
Klasa modelu Game. Posiada ona pola:

* ID - Unikalny klucz główny modelu, automatycznie generowany dla każdego kolejnego zapisanego w bazie obiektu
* gameName – Nazwa gry
* textToWrite – powiązanie Tekstu do danej gry. W bazie danych jest to klucz obcy do tabeli WritingText
* wordCount – liczba słów w tekscie
* minWordSpeed – minimalna wartość słów na minute potrzebna do ukończenia gry
* maxMistakes – maksymalna ilość błędów popełnionych w trakcie gry
* difficulty – stopnień trudności gry. Na jego podstawie ustalane są wartości minWordSpeed i maxMistakes dzięki metodzie difficultySetting()



Posiada ona podklasę GameJSON utworzoną w celu łatwiejszego serializowania i deserializowania danych z i do frontu aplikacji.

Dodatkowo enum Difficulty ze stopniami trudności.

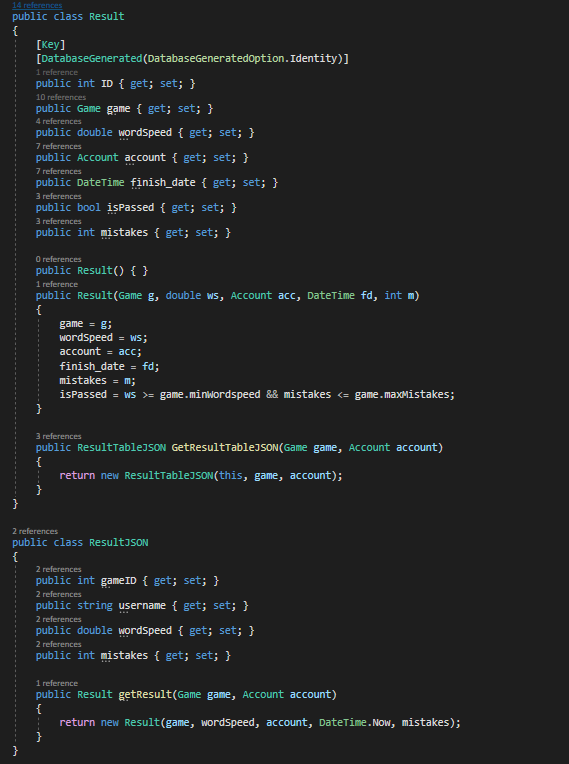


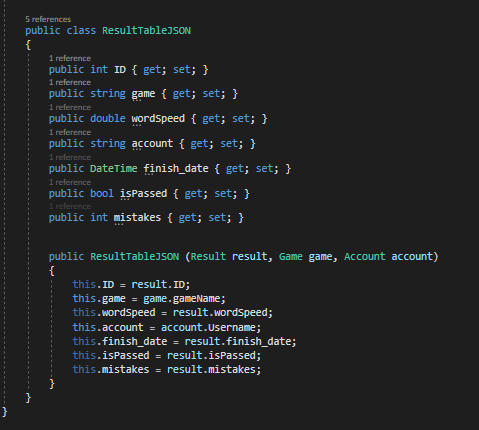
Klasa modelu Result. Posiada ona pola:

* ID – Unikalny klucz główny modelu, automatycznie generowany dla każdego kolejnego zapisanego w bazie obiektu
* Game – Klucz obcy do tabeli Game
* wordSpeed – uzyskana szybkość słów na minute na końcu gry
* account – klucz obcy do tabeli Account
* finish\_date – data ukończenia gry
* isPassed – osądzenie czy wordSpeed i mistakes były dostateczne do ukończenia gry z wynikiem pozytywnym
* mistakes – liczba błędów

Posiada również 3 podklasy:

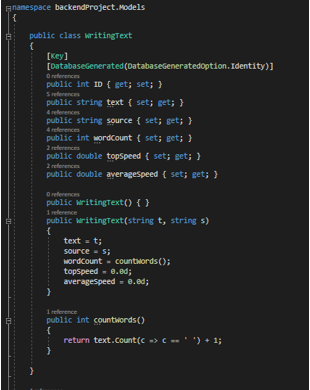
* ResultJSON – klasa do łatwej serializacji i deserializacji całego obiektu Result
* ResultSearchJSON – klasa do obsługi parametrów przekazywanych do filtrowania wyników
* ResultTableJSON – klasa służąca do mapowania obiektów Result na formę gotową do zapełnienia tabeli wyników.



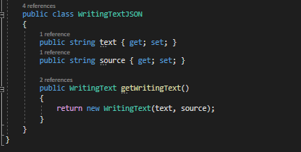


Klasa modelu WritingText. Posiada ona pola:

* ID – Unikalny klucz główny modelu, automatycznie generowany dla każdego kolejnego zapisanego w bazie obiektu
* Text – Tekst który użytkownik będzie pisać
* Source – źródło tesktu
* wordCount – liczba słów w tekście
* topSpeed – najwyższa uzyskana szybkość pisania
* averageSpeed – średnia uzyskana szybkość pisania

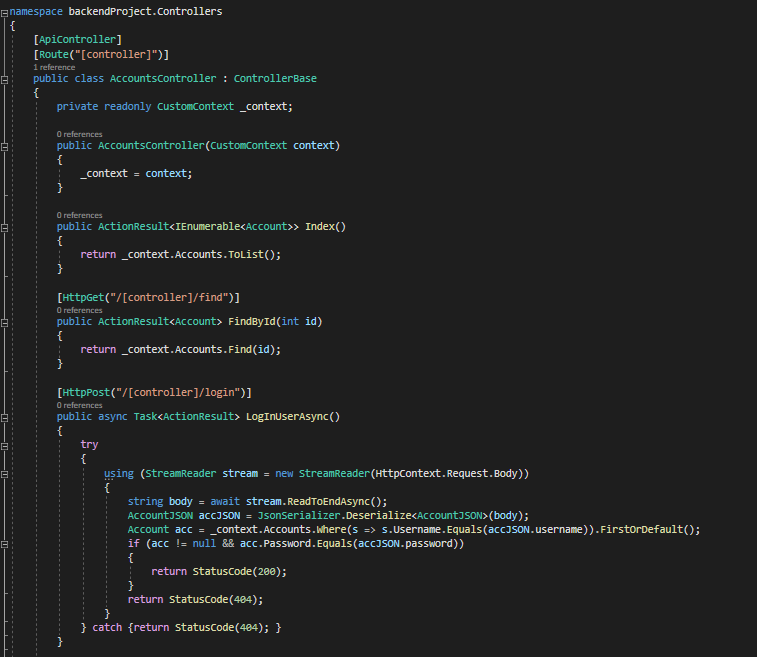


Posiada ona również podklasę WritingTextJSON do łatwej serializacji i deserializacji obiektów WritingText.

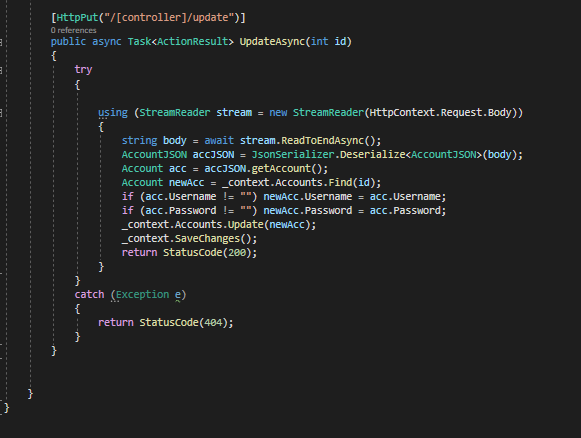


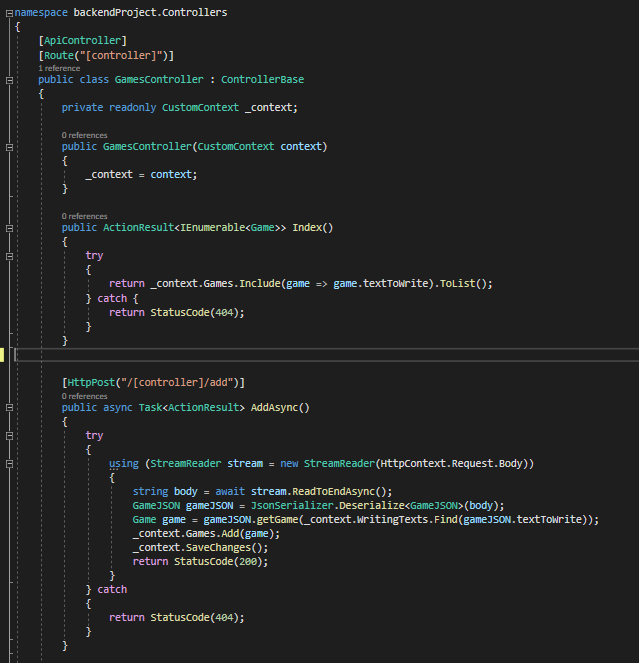
## Opis działania kontrolerów serwisu do komunikacji:

Do komunikacji z serwisem wykorzystujemy Controllery. Służą one do obsługi zapytań poprzez end-pointy. Lista dostępnych end-pointów i ich opis wraz z przykładowymi wynikami znajduję się w tabeli API. Jak na aplikację webową API jest wystawione przez serwer i do tych adresów wysyłane są zapytania, które obsługują wspomniane wcześniej kontrolery.

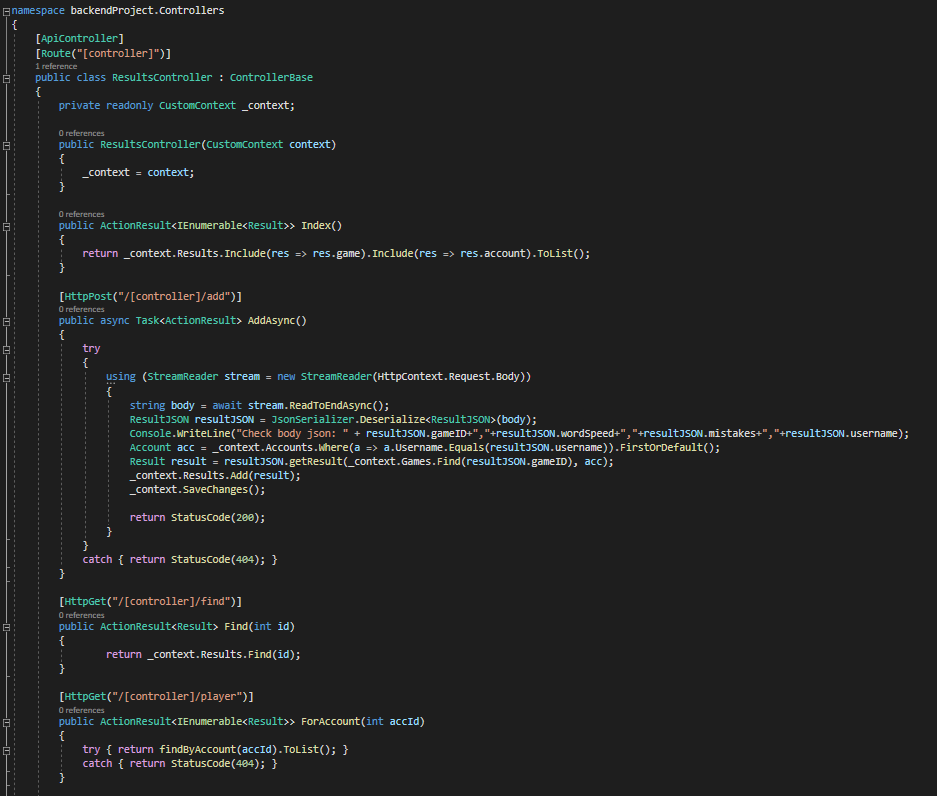


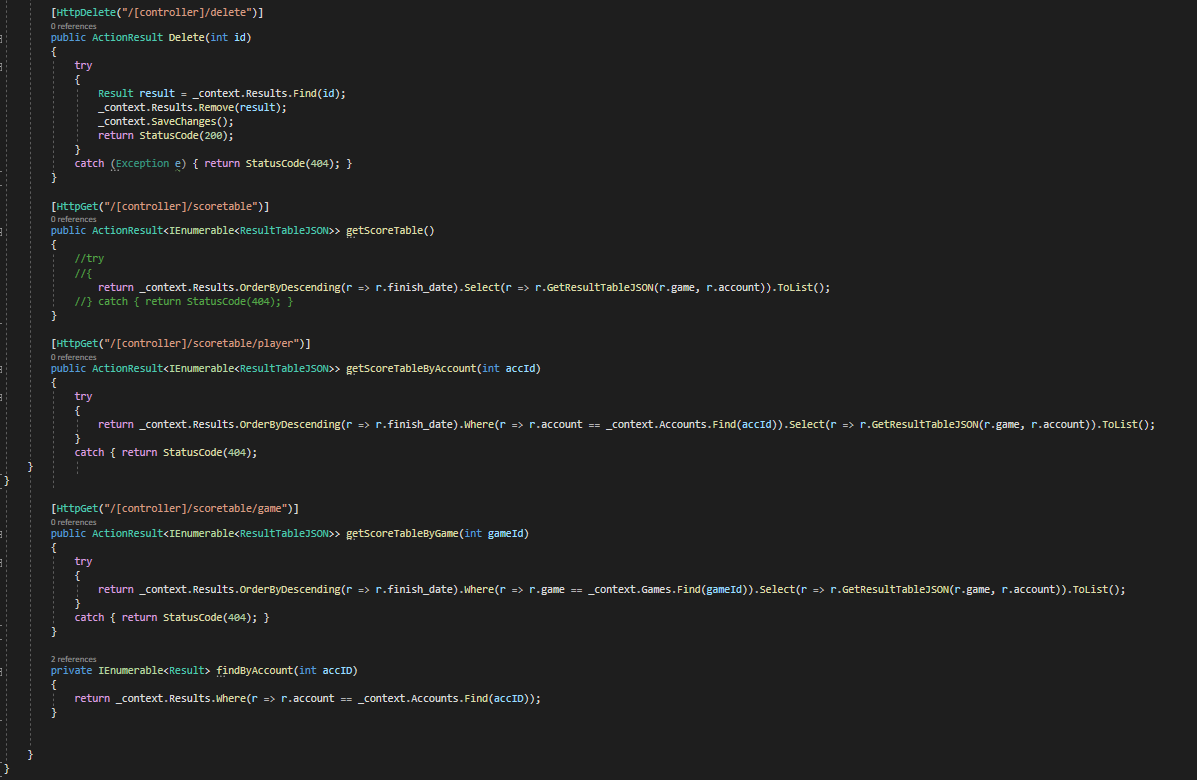


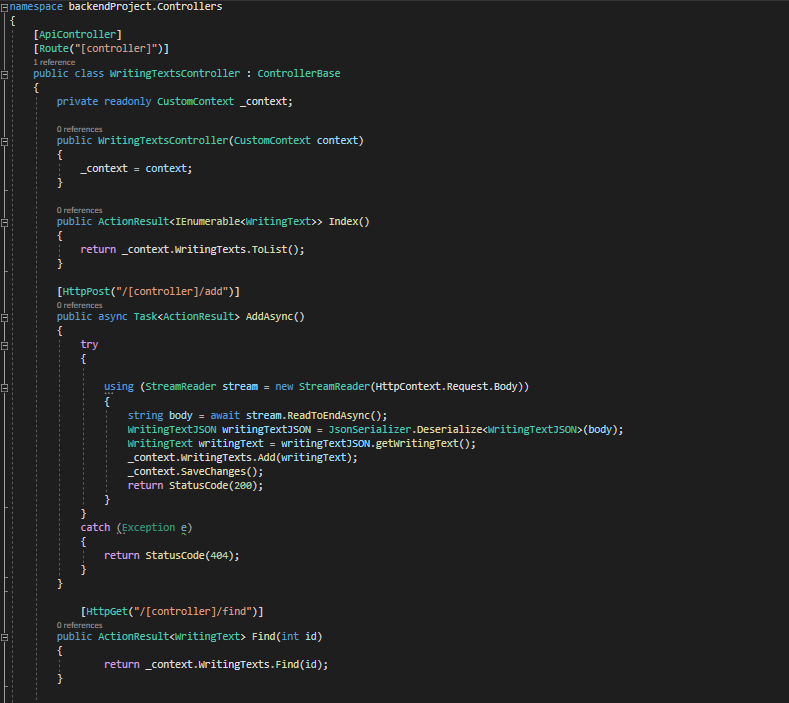






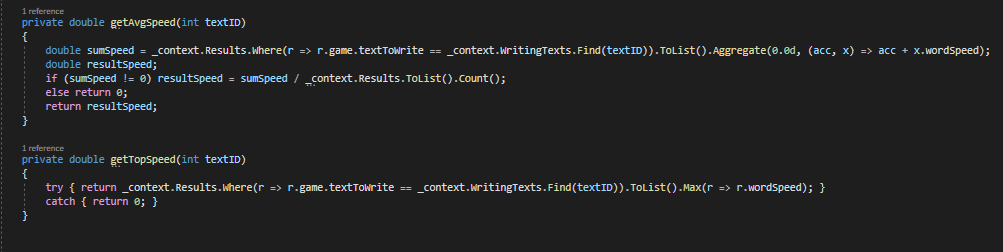




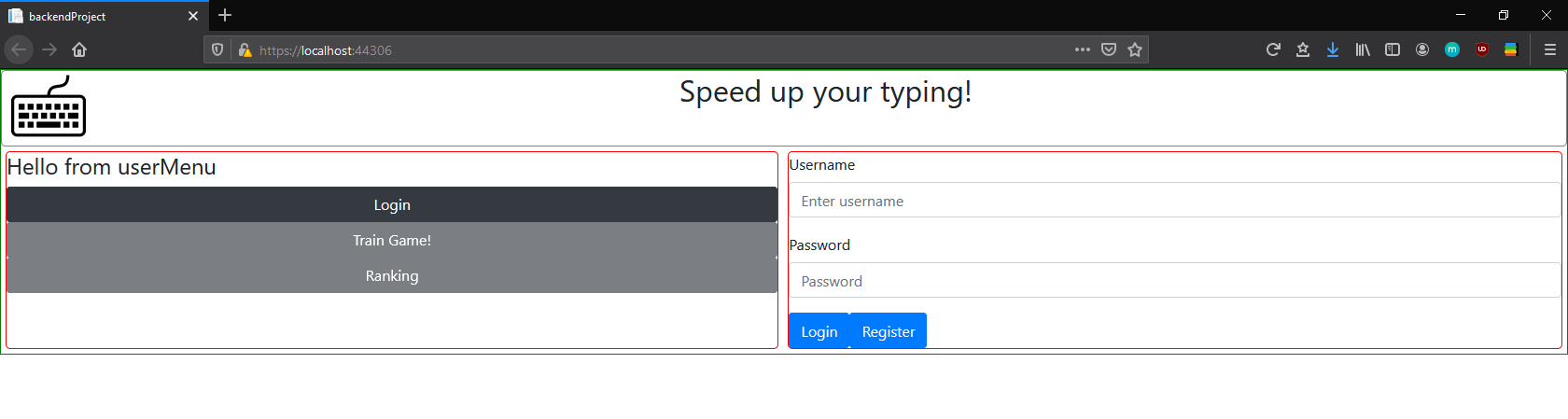


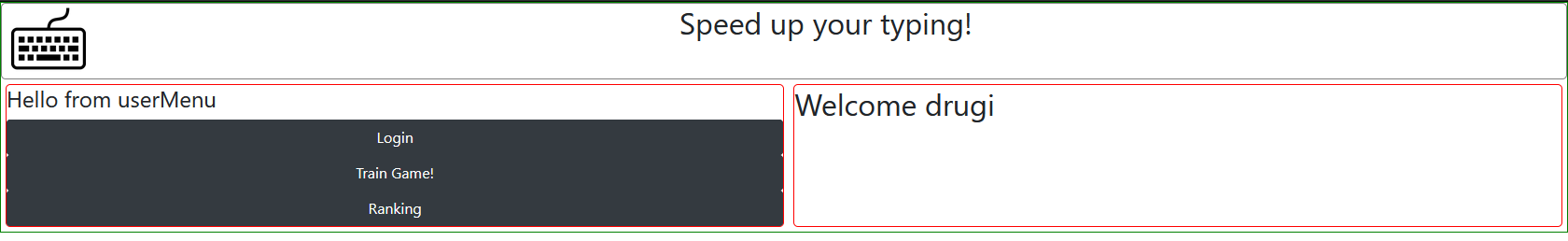


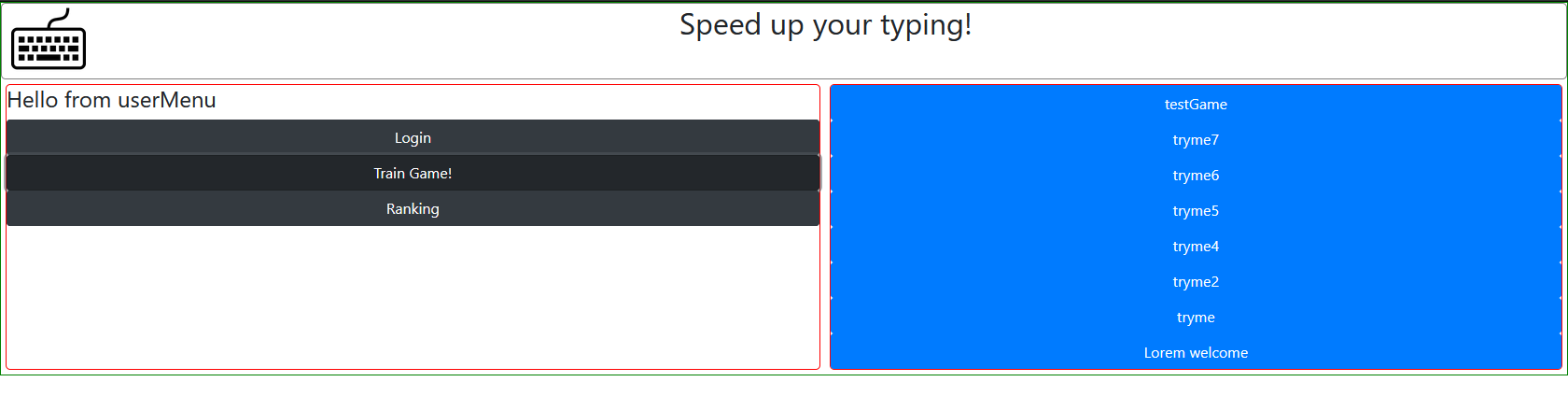
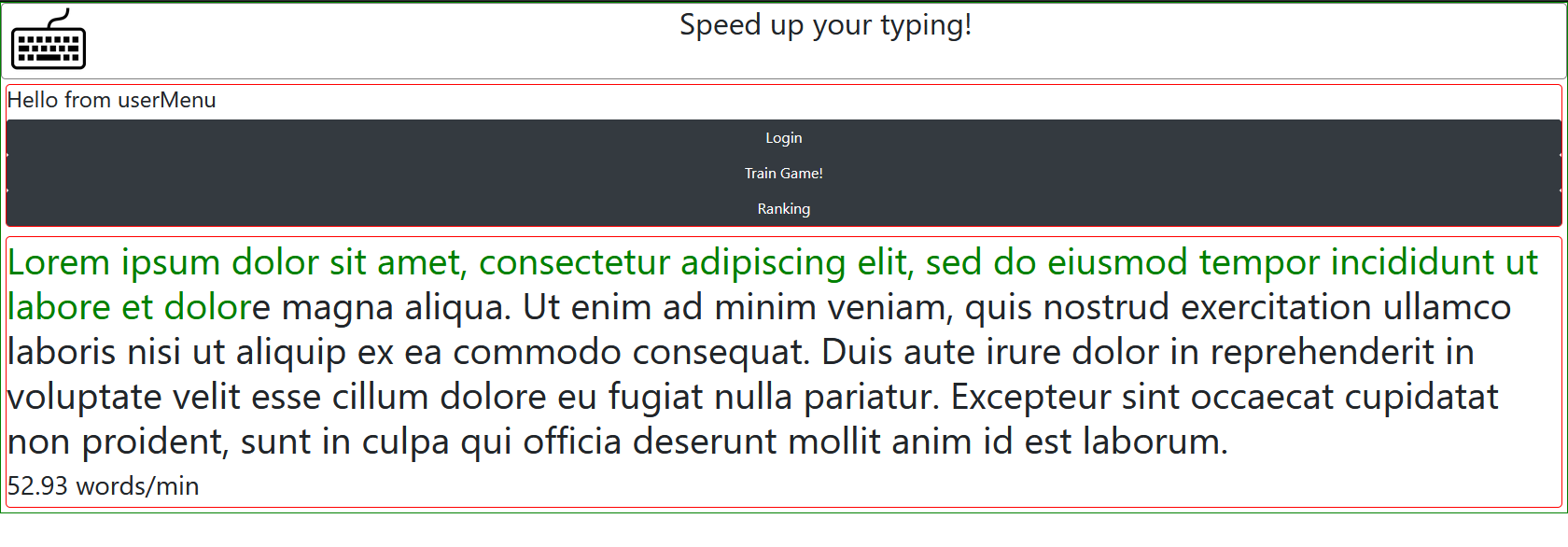
Metody dla tekstu, do bieżącego aktualizowania najlepszego wyniku i średniej prędkości:



## Wygląd poszczególnych okien ze strony aplikacji klienckiej:

Menu główne po uruchomieniu aplikacji:  


Po udanym zalogowaniu:  


Wybór gry po kliknięciu w przycisk Train Game!:  
  
Główna funkcjonalność, czyli wpisywanie tekstu na czas:  


Po zakończeniu wpisywania, dane o grze wysyłane są do serwisu i zapisywane w bazie danych. Można je potem sprawdzić przechodząc do okna Ranking:  
