# Web Services and Cloud – Practical Exam – September 2014

## Task description

### Bulls and Cows game description

Implement the server logic of the Bulls and Cows game. The rules of the game are:

1. Both players think up a **4 digits**, consisting of the digits 0-9, with no repeating digits
2. We will name one of the players "**red player**" and the other one "**blue player**"
3. The game is **played in turns**, a player is **randomly selected** to have the first turn
4. The player who is in turn to play **tries to guess the opponent's number**:
   * He receives information about his guess in the forms of "count of bulls" and "count of cows"
   * A "bull" is a **correct digit** at the **correct position**
   * A "cow" is a **correct digit** at the **wrong position**
   * E.g. if the opponent's number is 1234 and the current player guesses "4631", then he has 1 bull (3) and 2 cows (1 and 4)
5. It becomes the other player's turn.
6. Steps 4 and 5 are repeated until someone guesses the opponents number (i.e. has 4 "bulls")

The game always ends with one player winning

### Game logic

The game logic is performed as follows:

1. A **user registers** an account, providing credentials (username and password)
2. The **user can login**into their account using their credentials (username and password)
   * S/he receives an access token that is used to authenticated theirself in the system
3. An authenticated user **creates a game**
   * The game is created and is marked as **available for joining**
   * The creator user is **marked as red player**
   * S/he **provides a four non-repeating digits**that aretheir number(called *user-number*)
4. Another authenticated user **joins the game**
   * S/he **provides a four non-repeating digits**, that aretheir number (called *user-number*)
   * He is **marked as blue player**
5. When a blue player joins a game:
   * The server **randomly decides** which player starts first
     + For explanation purposes, let's assume the **red**  is chosen
   * The **creator of the game receives a notification** that their game now has a blue player
   * The **player in turn receives a notification** that it is their turn in the game
   * The game can be played
6. The player in turn (in our case - the red player)**can make a guess against the number** of their opponent (in our case - the blue player)
   * They **send four non-repeating digits** (called *guess-number*)
   * The server compares the *guess-number*against their opponents *user-number* and returns the corresponding bulls and cows
   * If the *guess-number* is the same as the opponent's *user-number* (i.e. it has 4 bulls), the game is**finished**, and **scores are applied**
     + The winner gets a score "**won"**
     + The loser gets a score "**lost"**
   * If the *guess-number* is not the same as the user-number, the server marks that the other player is in turn (in our case the blue)
   * The **other player receives a notification** that their opponent have finished their turn
7. **Repeat step 6**while one of the players guesses right *the user-number* of their opponent
8. **When a player guesses the number** of their opponent:
   * The **winner** is given **a score "won"** and **receives a notification** that s/he has won the game
   * The **loser** is given **a score "lost"** and **receives a notification** that s/he has lost the game

Every user can play many games simultaneously.

## Data tasks

### Data layer(8 points)

Create a database that can be used with the Bulls and Cows game. You must use MS SQL server and Entity framework with either code-first or database-first approaches.

### Repositories (5 points)

Create repositories using the *Repository pattern* to abstract the usage of the data layer. The usage of the *unit-of-work pattern* is not obligatory.

## Web API tasks

**All services have a full description in the file "Web-Services-and-Cloud-Services-Description.docx"**

### Login and register services (5 points)

Create a Login/Register RESTful services following the format:

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Url endpoint** | **Description** |
| POST | /api/account/register | Registers a new user |
| POST | /token | Logs in an existing user |

### Games services (20 points)

Implement functionality for **listing**/**creating**/**joining** games.

**Not authenticated users can see all games** that are created, but still don't have two players (i.e. a blue player has not joined yet)

**Authenticated players** can:

* **See all games** that are created, but still don't have two players (i.e. a blue player has not joined yet)
* **See all games** that s/he is part of and are still not finished (i.e. they are still played)
* **Create a new game**, proving a name for the game and a *user-number* for the blue player to guess
* **Join a created game**, that is still available (i.e. a blue player has not joined yet), providing a user-number for the red player to guess
* **View the details for a game**, that s/he is in

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Url endpoint** | **Description** |
| GET | /api/games | **Does not require authenticatin**  Gets games that are in state available for joining.  Return only the top 10 games, after sorting:   * By game state * Then by the name of the game * Then by the date of creation * Then by the name of the red player |
| GET | /api/games?*page=P* | **Does not require authentication**  **The same as /*api/games***, but returns only the games on the page P. Each page has a size of 10 games |
| GET | /api/games | **Requires an authentication**  Get games that either:   * The authenticated user is part of and are not finished yet * Are available for joining, meaning that a blue player has yet to join the game   Return only the top 10 games, after sorting:   * By game state * Then by the name of the game * Then by the date of creation * The by the name of the red player |
| GET | /api/games?*page=P* | **Requires an authentication**.  **The same as *api/games***, but returns only the games on the page P. Each page has a size of 10 games |
| GET | /api/games/*ID* | **Requires an authentication**.  Can be called only from a player that is either red or blue player in this game.  Gets the details for a started game:   * Names of red and blue players * Red player's guesses and results * Blue player's guesses and results * User's user-number in the corresponding game |
| POST | /api/games | **Requires an authentication**.  Creates a new game, proving a *name of the game* and *user-number* |
| PUT | /api/games | **Requires an authentication**.  Joins a game that is available for joining, proving a *user-number* |

### Guess services (7 points)

Implement functionality for playing a game. Follow the criteria:

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Url endpoint** | **Description** |
| POST | /api/games/*GAME\_ID*/guess | **Requires an authentication**.  The player in turn makes a guess for a game with ID = GAME\_ID |

### Notifications services (15 points)

Implement functionality for notifications.

Notification is created when:

* A game is joined by a blue player, the red player receives a notification
* A player plays their turn, the other player receives a notification that it is their turn
* A player guesses the number of their opponent. Both players receive notifications

**All returned UNREAD notifications are marked as READ**

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Url endpoint** | **Description** |
| GET | /api/notifications/ | **Requires an authentication**.  Returns the notifications of the authenticated user. The notifications are ordered by date and only the most recent 10 are returned |
| GET | /api/nofications?page=P | **Requires an authentication**.  **The same as the above**, but returns the notifications from p\*10 to p\*11-1, i.e. if P=6, then returns notifications from 60 to 69 |
| GET | /api/notifications/next | **Requires an authentication**.  Returns a single notification - the oldest unread notification of the authenticated user |

### Scores services (5 points)

Implement web services for the high score board of the application

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Url endpoint** | **Description** |
| GET | /scores | **Does not require an authentication**.  Returns the top 10 players with highest ranks. The rank is calculated using the following formula:  Rank = winsCount \* 100 + lossesCount\*15 |

## WCF tasks

### Usersinfoservices (10 points)

Implement the following services using WCF:

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Url endpoint** | **Description** |
| GET | /users | **Does not require an authentication**.  Returns the first 10 users, sorted by username |
| GET | /users*?page=P* | **Does not require an authentication**.  **The same as with /users**, but returns the P-th page with users. i.e. if P=6, then the returned users must be from 60 to 69, when sorted alphabetically by username |
| GET | /users/*ID* | **Does not require an authentication**.  Returns a detailed information about a user with the provided ID.  The information contains the user's id, username, number of won games, number of lost games and rank |

## Testing tasks

### Unit testing (7 points)

Write unit tests for the following functionality that uses the endpoints:

* /scores – getsthe high score board
* /api/games – gets the available for joining games (the one without authentication)

### Integration testing (8 points)

Write integration tests for the endpoints:

* /scores – get the high score board
* /api/games – get the available for joining games (the one without authentication)

## High quality code and validation tasks

### High-quality code (10 points)

* Provide validation on all needed services
* In case of error return the appropriate HTTP Status codes
* Write high-quality, abstract codethat is easy to maintain and extend

## Evaluation Criteria

The evaluation criteria are as follows:

* Correct and complete fulfillment of the requirements.
* Good technical design and appropriate use of technologies.
* High-quality programming code – correctness, readability, maintainability.
* Performance – highly-efficient code.

## Other Terms

During the exam you are allowed to use any teaching materials, lectures, books, existing source code, and other paper or Internet resources.

The Telerik Academy Anti-cheat client should be turned on during the entire exam.

Direct or indirect communication with anybody in class or outside is forbidden. This includes, but does not limit to, technical conversations with other students, using mobile phones, chat software (Skype, ICQ, etc.), email communication, posting in forums, folder synchronization software (like Dropbox), etc.

## Exam Duration

Students are allowed to work up to 8 hours.

# Services Explanations

## Login and Register Services

### Register user

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request** | | | | |
| **HTTP Method:** | POST | **URL:** | [http://localhost:XXXXX/**api/account/register**](http://localhost:XXXXX/api/account/register) | |
| **Headers** | Content-Type: *application/x-www-form-urlencoded* | | | |
| **Request Body:** | Email=doncho@minkov.it&Password=123456q&ConfirmPassword=123456q | | | |
| **Response** | | | | |
| **Status Code:** | 200 Ok | **Body:** | | empty |

### Login user

A user provides a username, password and grant\_type and if correct receives an**access\_token**. The access\_token is used to authenticate the user against the server and use the private services (those that require aunthentication)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request** | | | | |
| **HTTP Method:** | POST | **URL:** | [http://localhost:XXXXX/**token**](http://localhost:XXXXX/token) | |
| **Headers** | Content-Type: *application/x-www-form-urlencoded* | | | |
| **Request Body:** | Username=doncho%40minkov.it&Password=123456q&grant\_type=password | | | |
| **Response** | | | | |
| **Status Code:** | 200 Ok | **Body:** | | {  **"access\_token": "LONG\_STRING",**  "token\_type": "bearer",  "expires\_in": 1209599,  "userName": "doncho@minkov.it",  ".issued": "Mon, 22 Sep 2014 11:25:34 GMT",  ".expires": "Mon, 06 Oct 2014 11:25:34 GMT"  } |

## Game services

### Get public games

Does not require authentication

Returns games, that can be joined (i.e. do not have a blue player). The returned games are always 10 or less, depending on the number of games on the server and the page.

The games are sorted:

* By game state
* Then by the name of the game
* Then by the date of creation
* Then by the name of the red player

The *?page* parameter is optional. If it is present, return the games at the given page.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request** | | | | |
| **HTTP Method:** | GET | **URL:** | [http://localhost:XXXXX/**api/games?page=P**](http://localhost:XXXXX/api/games?page=P) | |
| **Headers** | Content-Type: *application/json* | | | |
| **Request Body:** | empty | | | |
| **Response** | | | | |
| **Status Code:** | 200 Ok | **Body:** | | [  {  "Id": 5,  "Name": "Battle of the titans",  **"Blue": "No blue player yet"**,  "Red": "doncho@minkov.it",  **"GameState": "WaitingForOpponent"**,  "DateCreated": "2014-09-22T14:31:51.067"  },  { /\* another game \*/},  { /\* another game \*/}  ] |

### Get available for join games and authenticated user games

Requires an authentication

Returns the games that are available for joining and the games, that the authenticated user is part of. The returned games are always 10 or less, depending on the number of games on the server and the page.

The games are sorted:

* By game state
* Then by the name of the game
* Then by the date of creation
* Then by the name of the red player

The *?page* parameter is optional. If it is present, return the games at the given page.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request** | | | | |
| **HTTP Method:** | GET | **URL:** | [http://localhost:XXXXX/**api/games?page=P**](http://localhost:XXXXX/api/games?page=P) | |
| **Headers** | Content-Type: *application/json*  Authorization: *Bearer ACCESS\_TOKEN (received after login)* | | | |
| **Request Body:** | empty | | | |
| **Response** | | | | |
| **Status Code:** | 200 Ok | **Body:** | | [  {  "Id": 5,  "Name": "Battle of the titans",  **"Blue": "No blue player yet"**,  "Red": "doncho@minkov.it",  **"GameState": "WaitingForOpponent"**,  "DateCreated": "2014-09-22T14:31:51.067"  },  {  "Id": 1,  "Name": "New game by doncho@minkov.it",  **"Blue": "minkov@doncho.it"**,  "Red": "doncho@minkov.it",  **"GameState": "RedInTurn"**,  "DateCreated": "2014-09-22T10:39:37.087"  }  ] |

### Get game details

Requires an authentication

The authenticated user must be either blue or red player in the game

Returns the game details about a game that is currently played (i.e. is not available for joining and is not finished)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request** | | | | |
| **HTTP Method:** | GET | **URL:** | [http://localhost:XXXXX/**api/games/{ID}**](http://localhost:XXXXX/api/games/%7bID%7d) | |
| **Headers** | Content-Type: *application/json*  Authorization: *Bearer ACCESS\_TOKEN (received after login)* | | | |
| **Request Body:** | empty | | | |
| **Response** | | | | |
| **Status Code:** | 200 Ok | **Body:** | | {  "Id": 1,  "Name": "Battle of the titans",  "DateCreated": "2014-09-22T10:39:37.087",  "Red": "doncho@minkov.it",  "Blue": "minkov@doncho.it",  "YourNumber": 1234,  "YourGuesses": [  {  "Id": 8,  "UserId": "7e1aaf37-d7c3-42e3-8781-e49bce747206",  "Username": "doncho@minkov.it",  "GameId": 1,  "Number": "1234",  "DateMade": "2014-09-22T14:48:01.16",  "CowsCount": 4,  "BullsCount": 0  },  {  "Id": 10,  "UserId": "7e1aaf37-d7c3-42e3-8781-e49bce747206",  "Username": "doncho@minkov.it",  "GameId": 1,  "Number": "4576",  "DateMade": "2014-09-22T14:48:19.617",  "CowsCount": 0,  "BullsCount": 1  },  {  "Id": 12,  "UserId": "7e1aaf37-d7c3-42e3-8781-e49bce747206",  "Username": "doncho@minkov.it",  "GameId": 1,  "Number": "1209",  "DateMade": "2014-09-22T14:48:27.32",  "CowsCount": 2,  "BullsCount": 0  }  ],  "OpponentGuesses": [  {  "Id": 9,  "UserId": "12d10b41-fdd4-4d61-8ad5-980af83263d8",  "Username": "dodo@minkov.it",  "GameId": 1,  "Number": "5432",  "DateMade": "2014-09-22T14:48:14.753",  "CowsCount": 2,  "BullsCount": 1  },  {  "Id": 11,  "UserId": "12d10b41-fdd4-4d61-8ad5-980af83263d8",  "Username": "dodo@minkov.it",  "GameId": 1,  "Number": "8523",  "DateMade": "2014-09-22T14:48:24.003",  "CowsCount": 2,  "BullsCount": 0  },  {  "Id": 13,  "UserId": "12d10b41-fdd4-4d61-8ad5-980af83263d8",  "Username": "dodo@minkov.it",  "GameId": 1,  "Number": "4562",  "DateMade": "2014-09-22T14:48:31.12",  "CowsCount": 2,  "BullsCount": 0  }  ],  "YourColor": "red",  "GameState": "RedInTurn"  } |

### Create a new game

Requires authentication

Creates a new game, providing a game name and a user-number.The authenticated user is automatically marked as red player

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request** | | | | |
| **HTTP Method:** | POST | **URL:** | [http://localhost:XXXXX/**api/games**](http://localhost:XXXXX/api/games) | |
| **Headers** | Content-Type: *application/json*  Authorization: *Bearer ACCESS\_TOKEN (received after login)* | | | |
| **Request Body:** | {"name": "The Empire strikes back!",  "number": "5976"} | | | |
| **Response** | | | | |
| **Status Code:** | 201Created | **Body:** | {  "Id": 6,  "Name": "The Empire strikes back!",  "Blue": "No blue player yet",  "Red": "dodo@minkov.it",  "GameState": "WaitingForOpponent",  "DateCreated": "2014-09-23T06:41:51.5816277+03:00"  } |

### Join an available for join game

Requires an authentication

The authenticated user joins a created game. A game, created by a user, cannot be joined by the same user.

The red player (the creator) receives a message that a blue player has joined his game.

The first player in turn is decided randomly at the server. The selected player in turn receives a notification that their turn has come up

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request** | | | | |
| **HTTP Method:** | PUT | **URL:** | [http://localhost:XXXXX/**api/games/{ID}**](http://localhost:XXXXX/api/games/%7bID%7d) | |
| **Headers** | Content-Type: *application/json*  Authorization: *Bearer ACCESS\_TOKEN (received after login)* | | | |
| **Request Body:** | {"number": "9148"} | | | |
| **Response** | | | | |
| **Status Code:** | 200 Ok | **Body:** | | {"result": "You joined game \"The Empire strikes back!\""} |

## Guess services

### Make a guess for a game

Requires an authentication

Makes a guess for a game with the provided ID

The game must be in playing mode (i.e. not finished or available for joining)

The authenticated user must be either blue or red player in the game and should be their turn

If the users guesses right their opponent's user-number, then the game is finished, both players are applied the score of the game, and they both receive a notification:

* The winner receives a score "won" and awinning notification
* The loser receives a score "lost" and a losing notification

If the player-in-turn does not guess their opponent's user-number, then the turn is switched to the other player and s/he receives a notification

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request** | | | | |
| **HTTP Method:** | POST | **URL:** | [http://localhost:XXXXX/**api/games/{ID}/guess**](http://localhost:XXXXX/api/games/%7bID%7d/guess) | |
| **Headers** | Content-Type: *application/json*  Authorization: *Bearer ACCESS\_TOKEN (received after login)* | | | |
| **Request Body:** | {"number": "1234"} | | | |
| **Response** | | | | |
| **Status Code:** | 200 Ok | **Body:** | | {  "Id": 15,  "UserId": "12d10b41-fdd4-4d61-8ad5-980af83263d8",  "Username": "dodo@minkov.it",  "GameId": 6,  "Number": "1234",  "DateMade": "2014-09-23T06:52:47.038633+03:00",  "CowsCount": 2,  "BullsCount": 0  } |

## Score services

### Get high score

Publicly available – does not require authentication

Returns the high score board for the Bulls and Cows games, the results are sorted by rank, then by username

The high score board contains the top 10 users with greatestranks.

A user rank is calculated by the formula:

**USER\_RANK = 100 \* USER\_WINS\_COUNT + 15 \* USER\_LOSSES\_COUNT** (i.e. if a user has **3 wins** and **1 loss**, their rank is **100 \* 3 + 15 \* 1 = 315)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request** | | | | |
| **HTTP Method:** | GET | **URL:** | [http://localhost:XXXXX/**api/scores**](http://localhost:XXXXX/api/scores) | |
| **Headers** | Content-Type: *application/json* | | | |
| **Request Body:** | *empty* | | | |
| **Response** | | | | |
| **Status Code:** | 200 Ok | **Body:** | | [{"Username": "doncho@minkov.it",  "Rank": 315},  { "Username": "doncho@minkov.com",  "Rank": 115},  { "Username": "dodo@minkov.it",  "Rank": 30 } ] |

## Notifications services

### Get notifications page

Requires an authentication

Returns the notifications for the authenticated user

The notifications are sorted by date, and only the most recent are returned

The *?page* parameter is optional. If it is present, return the notifications at the given page.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request** | | | | |
| **HTTP Method:** | GET | **URL:** | [http://localhost:XXXXX/**api/notifications?page=0**](http://localhost:XXXXX/api/notifications?page=0) | |
| **Headers** | Content-Type: *application/json*  Authorization: *Bearer ACCESS\_TOKEN (received after login)* | | | |
| **Request Body:** | *empty* | | | |
| **Response** | | | | |
| **Status Code:** | 200 Ok | **Body:** | | [  {  "Id": 19,  "Message": "It is your turn in game \"The Empire strikes back!\"",  "DateCreated": "2014-09-23T06:52:47.057",  "Type": "YourTurn",  "State": "Unread",  "GameId": 6  },  {  "Id": 18,  "Message": "It is your turn in game \"New game by doncho@minkov.it\"",  "DateCreated": "2014-09-23T06:51:45.327",  "Type": "YourTurn",  "State": "Unread",  "GameId": 2  },  {  "Id": 17,  "Message": "You beat darth@vader.sith in game \"New hope\""  "DateCreated": "2014-09-22T14:48:31.123",  "Type": "GameWon",  "State": "Unread",  "GameId": 111  },  {  "Id": 15,  "Message": "darth@vader.sith beat you in game \"The Empire strikes back\"",  "DateCreated": "2014-09-22T14:48:24.01",  "Type": "GameLost",  "State": "Unread",  "GameId": 112  },  {  "Id": 13,  "Message": "It is your turn in game \"New game by doncho@minkov.it\"",  "DateCreated": "2014-09-22T14:48:14.757",  "Type": "YourTurn",  "State": "Read",  "GameId": 1  },  {  "Id": 11,  "Message": "It is your turn in game \"New game by doncho@minkov.it\"",  "DateCreated": "2014-09-22T14:47:52.153",  "Type": "YourTurn",  "State": "Read",  "GameId": 4  },  {  "Id": 9,  "Message": "It is your turn in game \"New game by doncho@minkov.it\"",  "DateCreated": "2014-09-22T14:47:48.323",  "Type": "YourTurn",  "State": "Read",  "GameId": 4  },  {  "Id": 7,  "Message": "dodo@minkov.it joined your game \"New game by doncho@minkov.it\"",  "DateCreated": "2014-09-22T12:32:46.297",  "Type": "GameJoined",  "State": "Read",  "GameId": 4  },  {  "Id": 4,  "Message": "It is your turn in game \"New game by doncho@minkov.it\"",  "DateCreated": "2014-09-22T12:32:29.713",  "Type": "YourTurn",  "State": "Read",  "GameId": 1  }  ] |

### Get next notification

Returns the **oldest unread notification**

If no unread notifications, return empty response body and status code 304 (Not modified)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request** | | | | |
| **HTTP Method:** | GET | **URL:** | [http://localhost:XXXXX/**api/notifications/next**](http://localhost:XXXXX/api/notifications/next) | |
| **Headers** | Content-Type: *application/json*  Authorization: *Bearer ACCESS\_TOKEN (received after login)* | | | |
| **Request Body:** | *Empty* | | | |
| **Response** | | | | |
| **Status Code:** | 200 Ok | **Body:** | | {  "Id": 1,  "Message": "dodo@minkov.it joined your game \"New game by doncho@minkov.it\"",  "DateCreated": "2014-09-22T12:27:44.77",  "Type": "GameJoined",  "State": "Unread",  "GameId": 3  } |

## User services

**Implemented with WCF**

### Get users

Does not require authentication

Returns a collection containing the registered users, sorted by username, in pages of 10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request** | | | | |
| **HTTP Method:** | GET | **URL:** | [http://localhost:YYYYY/**services/users.svc?page=0**](http://localhost:YYYYY/services/users.svc?page=0) | |
| **Headers** | Content-Type: *application/json* | | | |
| **Request Body:** | *empty* | | | |
| **Response** | | | | |
| **Status Code:** | 200 Ok | **Body:** | | [{"Id": "12d10b41-fdd4-4d61-8ad5-980af83263d8",  "Username": "doncho@minkov.it"},  {"Id": "2d3d901e-2f6b-49b3-9224-1cd30840846a",  "Username": "darth@vader.sith" },  {"Id": "7e1aaf37-d7c3-42e3-8781-e49bce747206",  "Username": "luke@skywalker.jedi" }  ] |

### Get user details

Does not require authentication

Returns the details of a user by ID

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request** | | | | |
| **HTTP Method:** | GET | **URL:** | [http://localhost:YYYYY/**services/users.svc/{ID}**](http://localhost:YYYYY/services/users.svc/%7bID%7d) | |
| **Headers** | Content-Type: *application/json* | | | |
| **Request Body:** | *empty* | | | |
| **Response** | | | | |
| **Status Code:** | 200 Ok | **Body:** | | {  "Id": "7e1aaf37-d7c3-42e3-8781-e49bce747206",  "Losses": 7,  "Rank": 705,  "Username": "darth@vader.sith",  "Wins": 6  } |