# Web Services and Cloud – Practical Exam – November 2015

## Task description

**Teleimot.bg web application description**

You are given the task to implement web services for real estate ads using ASP.NET Web API 2 and WCF. Parts of the endpoints can be accessed by anonymous users and other parts need authentication. **Users register with username, e-mail and password and then can request token for authentication and authorization.** Users can post ads about real estates they are selling or are giving for rent. Ads can be viewed or commented on. Additionally, each user can receive ratings from 1 to 5 from other people. See below for more details.

**Before submitting do not forget to delete the "bin", "obj" and "packages" folders.**

## Data tasks

### Data layer (10 points)

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

Validations:

* **Real estate title** is between 5 and 50 symbols, inclusive.
* **Real estate description** is between 10 and 1000 symbols, inclusive.
* **Real estate type** is one of the following – "Apartment", "House", "Office" or "Storehouse"
* **Real estate construction year** is minimum 1800.
* **Real estate title**, **description**, **address** and **contact** are **required**. Additionally the real estate can be published **only fo**r **renting**, **only for selling** or **for both**, but not without any of the two options.
* **Comments** have **required content** between 10 and 500 symbols, inclusive.
* **Ratings** are always between 1 and 5 (integers), inclusive.

### Repositories (5 points) + Services (8 points)

Create repositories using the *Repository pattern* to abstract the usage of the data layer. The usage of services is not obligatory but it will give you points. Instead of services, you can also use Unit of Work (again for full 8 points).

## Web API tasks

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

### Login and register services (7 points)

Create a Login/Register RESTful services following the format:

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Url endpoint** | **Description** |
| POST | /api/account/register | Registers a new user with **username**, e-mail and password |
| POST | /token | Logs in an existing user |

### Real estate services (20 points)

Implement functionality for **listing**/**creating**/**viewing** real estates.

**Non-authenticated users can see all listed real estates** that are created, but not their full details.

**Authenticated users** can:

* **See real estate** comments and contact information
* **Create new real estate** ad

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Url endpoint** | **Description** |
| GET | /api/RealEstates | **Does not require authentication.**  Gets all added real estates.  Return only the top 10 real estates, after sorting them by date and time of creation in **descending** order. |
| GET | /api/RealEstates*?skip=S&take=T* | **Does not require authentication.**  **The same as /api/RealEstates**, but skip S ads and take only T ads. Both parameters are not mandatory. Default skip is 0, default take is 10. Take cannot be more than 100. |
| GET | /api/RealEstates/*ID* | Returns information the about real estate ad with the provided ID. **Anonymous users cannot see the comments and the contact information of the ad. See the API description for more details.** |
| POST | /api/RealEstates | **Requires an authentication**.  Creates a new real estate ad, providing the required information to save one. **See the API description for more details.** |

### Comments services (15 points)

Implement functionality for adding and retrieving comments. All endpoints are available only for authenticated users. Follow the criteria:

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Url endpoint** | **Description** |
| GET | /api/Comments/*ID* | **Requires an authentication**.  Gets all comments by **real estate id**.  Return only the top 10 comments, after sorting them by date and time of creation in **ascending** order. |
| GET | /api/Comments/*ID?skip=S&take=T* | **Requires an authentication**.  **The same as /api/Comments/*ID***, but skip S comments and take only T comments. Both parameters are not mandatory. Default skip is 0, default take is 10. Take cannot be more than 100. |
| GET | /api/Comments/ByUser/*USERNAME* | **Requires an authentication**.  Gets all comments by **the user with the provided username**.  Return only the top 10 comments, after sorting them by date and time of creation in **ascending** order. |
| GET | /api/Comments/ByUser/*USERNAME?skip=S&take=T* | **Requires an authentication**.  **The same as /api/Comments/ByUser /*USERNAME***, but skip S comments and take only T comments. Both parameters are not mandatory. Default skip is 0, default take is 10. Take cannot be more than 100. |
| POST | /api/Comments | **Requires an authentication**.  Creates a new comment, providing the required information to save one. **See the API description for more details.** |

### Users services (10 points)

Implement functionality for getting user information and rating him/her.

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Url endpoint** | **Description** |
| GET | /api/Users/*USERNAME*  *No the best way done* | **Does not require authentication.**  Returns the users' username, total real estate ads, total comments and average total rating. |
| POST | /api/Users/Rate | **Requires an authentication**.  Adds new rating from 1 to 5 to a specified user. |

## WCF tasks

### Users info services (5 points)

Implement the following HTTP service using WCF (SOAP is not required):

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Url endpoint** | **Description** |
| GET | /users/top.svc/ | **Does not require an authentication**.  Returns the first 10 users, sorted by their average rating. |

## Testing tasks

### Unit testing, route testing and integration testing (10 points)

Write tests for the "api/Comments/ByUser/*USERNAME*" endpoint.

* **Unit tests** – test that the action behind this route has appropriate action filters and that with its default *skip* and *take* parameters it returns correct response with correct number of elements.
* **Integration tests** – test that the route requested without authenticated user returns unauthorized status code.
* **Route tests** – test that the route is resolved to the correct controller, action and route values for the following situations: with default skip and take, with provided custom skip, with provided custom take, with provided custom skip and custom take. Additionally, test that the route does not have valid model state, if you do not provide the username.

## 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 code that 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.