# Web Services and Cloud Exam (November 2015) – Videos

Design and implement **RESTful Web Services** based on ASP.NET Web API, Entity Framework Code First and MS SQL Server for a **video system**. The video system holds **videos, locations, tags, playlists** and **users**.

**Anonymous visitors** can view **videos**.  
**Registered users** can:

* Create videos.
* Create, edit and delete tags and add tags to videos.
* Create Playlists and add videos and tags to playlists.

### Compile and Run the Project

You are given a Visual Studio project "Videos" holding a **data layer** (holding EF data models and EF data context) and a **Web API application** (RESTful Web services). You need to **compile and run** the Web API application.

The project **already holds the user login and user register** functionality. You will need to write the rest.

You are given the data model classes holding:

* **Video** - has **title**, **video status**, **location** and a **set of tags**
* **Tag** - holds **name** and **hasAdultContent** flag
* **Playlist -** holds **name**, a **set of videos** and a **set of tags**
* **Location** - holds **country** and **city**
* **VideoStatus** - holds **name**

You are given a **Postman collection** of HTTP requests to test your REST service. Play with it.

Internet connection is required to download the referenced NuGet packages.

### Get All Videos

Write a REST service to list all Videos by given location.

* Returns all videos **with status Published** by **location id**, ordered by **username alphabetically**, then by **video title alphabetically**. Returns **id**, **title**, **status, country, city** and **owner** (owner’s name).

|  |  |  |
| --- | --- | --- |
| Request | GET /api/videos?locationId={locationId} | |
| Response | 200 OK  [{"id":6,"title":"Example","country":"Canada","city":"Ottawa","owner":"owner.mail@gmail.com"}..] | Returns the list of all published videos in the specified location, ordered by owner username, than by title as JSON array. |
| Error Response | 400 Bad Request | Returned in case of missing or invalid location data (e.g. not provided or does not have locatoinId). |

5 score

### Create New Video

Write a REST service to create a new video.

|  |  |  |
| --- | --- | --- |
| Request | | Example |
| POST /api/videos  Content-Type: application/json  {"title":"{title}","locationId":"{id}"} | | POST /api/videos  title=War+and+War&locationId=3756 |
| Response | 201 Created  Location: http://localhost:9016/api/videos/7  {"id":7,"Title":"War and war","status": "Pending", "country":"Bulgaria", "City":"Sofia", "Owner": "iownyou@mail.me"} | On success, the service returns **201** (**Created**) + a header "Location" holding the URL of the created video + the video data as JSON object in the response body. |
| Error Response | 400 Bad Request | Returned in case of missing or invalid video data (e.g. empty video title). |
| Error Response | 401 Unauthorized | Returned in case the user is not logged in. |

5 score

### Add Tags to a Video

Write a REST service to add tags to existing video. Video tags should be unique.

|  |  |  |
| --- | --- | --- |
| Request | | Example |
| POST /api/videos/{id}/addTag  Content-Type: application/x-www-form-urlencoded  tagId=*{tagId}* | | POST /api/videos/2/addTag  tagId=5 |
| Response | 200 OK | On success, the service returns 200 OK. |
| Error  Response | 400 Bad Request | Returned in case the tag already exits |
| Error  Response | 400 Bad Request | Returned in case of **invalid tag id** |
| Error Response | 401 Unauthorized | Returned in case the user is not logged in. |
| Error Response | 404 Not Found | Returned in case of **invalid video id**. |

10 score

### Create Tag

Write a REST service to create Tags in the system.

|  |  |  |
| --- | --- | --- |
| Request | POST /api/Tags  Content-Type: application/x-www-form-urlencoded  name={tagName}&isAdultContent ={isAdultContent} | Example |
| POST /api/Tags |
| Response | 201 Created  Location: http://localhost:9016/api/Tags/4  {"id":4,"name":"Tom&Jerry for adults","isAdultContent":true} | Returns the created tag. |
| Error Response | 400 Bad Request | Returned in case of tag already existing. |
| Error Response | 400 Bad Request | Returned in case of no data provided. |

5 score

### Delete Tag by Id

Write a REST service to delete Tag by id. If the tag is associated with a video or a playlist it should not be deleted.

|  |  |  |  |
| --- | --- | --- | --- |
| Request | DELETE /api/Tags/*{id}* | Example | DELETE /api/Tags/3 |
| Response | 200 OK  {"message":"Tag #3 deleted."} | On success, the service returns 200 (OK) + optional human-readable message, explaining that the tag was deleted. | |
| Error Response | 401 Unauthorized | Returned in case the user is not the tag owner. | |
| Error Response | 404 Not Found | Returned when the requested tag does not exists (invalid tag id). | |
| Error Response | 409 Bad Request | If the tag is associated with a video or a playlist | |

5 score

### Edit Tag

Create REST service to edit tags data.

|  |  |  |  |
| --- | --- | --- | --- |
| Request | PUT /api/Tags/*{id}*  Content-Type: application/x-www-form-urlencoded  name={tagName}&isAdultContent={isAdultContent} | Example | PUT /api/Tags/3  Name=EditTag&isAdultContent= false |
| Response | 200 OK  {"id":3,"name":"EditTAg","isAdultContent":false} | On success, the service returns 200 (OK) + the tag data. | |
| Error Response | 401 Unauthorized | Returned in case the user is not logged. | |
| Error Response | 404 Not Found | Returned when the requested tag does not exists (invalid tag id). | |
| Error Response | 409 Bad Request | If there is no tag data provided. | |

5 score

### Create Playlist

Write a REST service to create empty playlist.

|  |  |  |
| --- | --- | --- |
| Request | | Example |
| POST /api/Playlists  Content-Type: application/x-www-form-urlencoded  name={playlistName} | | POST /api/Playlists  name=Music |
| Response | 201 Created  Location: http://localhost:9016/api/videos/7  {"id":7,"Name":"Music", "Owner": iownyou@mail.me, "tags":[], "videos":[]} | On success, the service returns **201** (**Created**) + a header "Location" holding the URL of the created playlist + the playlist data as JSON object in the response body. |
| Error Response | 400 Bad Request | Returned in case of missing or invalid playlist data (e.g. empty video title). |
| Error Response | 401 Unauthorized | Returned in case the user is not logged in. |

5 score

### Add Video to Playlist

Add existing video to existing playlist.

|  |  |  |
| --- | --- | --- |
| Request | | Example |
| POST /api/playlists/{id}/addVideo  Content-Type: application/x-www-form-urlencoded  videoId=*{tagId}* | | POST /api/playlists/2/addVideo  videoId=5 |
| Response | 200 OK | On success, the service returns 200 OK. |
| Error  Response | 400 Bad Request | Returned in case of **invalid video id** |
| Error Response | 401 Unauthorized | Returned in case the user is not logged in. |
| Error Response | 404 Not Found | Returned in case of **invalid playlist id**. |

5 score

### Add Tag to Playlist

Write a REST service to add existing tags to playlist. Tags should be unique.

|  |  |  |
| --- | --- | --- |
| Request | | Example |
| POST /api/playlists/{id}/addTag  Content-Type: application/x-www-form-urlencoded  tagId=*{tagId}* | | POST /api/playlists/2/addTag  tagId=5 |
| Response | 200 OK | On success, the service returns 200 OK. |
| Error  Response | 400 Bad Request | Returned in case the tag already exits |
| Error  Response | 400 Bad Request | Returned in case of **invalid tag id** |
| Error Response | 401 Unauthorized | Returned in case the user is not logged in. |
| Error Response | 404 Not Found | Returned in case of **invalid playlist id**. |

10 score

### List Playlists by Location and by Adult Content

Write a REST service to get the **playlists** of the **current user** with paging.

* If a **locationId** is provided filter the videos from playlists from that location. (**For each playlist return only the videos from the given location**)
* If **adultContentAllowed** is provided:
  + If true – select all videos
  + If false – you should **not** return videos or playlists that have **Tag** with **isAdultContent** flag equals **true.**
* If **adultContentAllowed** is not provided assume it is **false.**
* **Paging**
  + **Example**: If **startPage = 2** and **limit = 4** then the **first 8 playlists should be skipped** (page 0 and 1) and **playlists 9, 10, 11 and 12** should be taken

|  |  |  |
| --- | --- | --- |
| Request | | Example |
| GET /api/Playlists?startPage={start-page}&limit={page-size}& locationId={locationId}&adultContentAllowed={adultContentAllowed} | | GET /api/ Playlists?startPage=0&limit=2& locationId=7&adultContentAxllowed=false |
| Response | 200 OK  [{"id":7,"Name":"Music", "Owner": iownyou@mail.me, "tags":[{"id":4,"name":"Tom&Jerry for adults","isAdultContent": true},…], "videos":[{"id":6,"title":"Example","country":"Canada","city":"Ottawa","owner":"owner.mail@gmail.com"},…] | On success, the service returns **200** **OK** |
| Error Response | 401 Unauthorized | Returned in case the user is not logged in. |

15 score

### Register User

You are given a REST service to **register** a user account by **username** (unique) and **password**. Do not touch it, just play with it to learn how it works:

|  |  |  |
| --- | --- | --- |
| Request | | Example |
| POST /api/account/register Content-Type: application/x-www-form-urlencoded  username=*some\_username*&pasword=*some\_password&confirmPassword=some\_password&email=myMail@gmail.com* | | POST /api/user/register  username=maria&password=pAss123&  confirmPassword=pAss123&email= myMail@gmail.com |
| Response | 200 OK  {"access\_token":"VccMrKjEWki…", "token\_type":"bearer",  "userName":"maria", … } | On success, the service returns 200 (OK) + the registered username + the access\_token for bearer authorization. |
| Error Response | 400 Bad Request | Returned in case of **missing** or **invalid** user account data (e.g. empty password) or **duplicated** username. |

### User Login

Once you have registered you can get your Bearer token by calling:

|  |  |  |
| --- | --- | --- |
| Request | | Example |
| POST /Token Content-Type: application/x-www-form-urlencoded  grant\_type=password&username=*some\_username*&pasword=*some\_password* | | POST /Token  grant\_type=password&username=maria&password=pAss123 |
| Response | 200 OK  {"access\_token":"VccMrKjEWki…", "token\_type":"bearer", … } | On success, the service returns 200 (OK) and the access\_token for bearer authorization. |
| Error Response | 400 Bad Request | Returned in case of **missing** or **invalid** user account data (e.g. empty password or invalid username or password). |

### Write Integration Tests for "Tags" Service

Write **integration tests** for the "**Edit** **Tags**" REST service. Ensure you cover all interesting cases. Put your tests in a new class called "TagsIntegrationTests".

10 score

### Repository and Unit of Work

Before you modifying your project first **backup your work**.

Implement the classical **Repository** and **Unit of Work patterns** to separate the EF data layer from the Web API controllers through interfaces and simplify the eventual unit testing of the Web API controllers. Use your new data layer in the Web API controllers instead of directly calling Entity Framework.

5 score

### Write Unit Tests for "Get All Videos" Service

Write **unit tests** with **mocking** for the "**Get all videos by location**" REST service. Use a **fake or mocked repository** and unit of work implementations. Test the work of the Web API controller only. Your unit test should not access the database. Ensure you cover all interesting cases. Put your tests in a new class called "**Videos**ControllerTests".

15 score

### Write Unit Tests for "Create Playlist" Service

Write **unit tests** with **mocking** for the "**Create Playlist**" REST service.

Bonus 10 score

## Exam Information

You are allowed to use any resources you have, e.g. Internet, software, existing code.

You are not allowed to get help from other people. Skype, ICQ, FB, email, talks, phone calls, etc. are forbidden.

Exam time: **6 hours**.