# Web Service & Cloud Retake Exam (December 2015) – Criminal Activities

Design and implement **RESTful Web Services** based on ASP.NET Web API, Entity Framework Code First and MS SQL Server for a global **criminal tracking system**. The **criminal tracking system** should hold **criminals**, **cartels**, **activities, locations** and **cities**.   
**Anonymous visitors** can:

* View all cartels
* View a single criminal by name and alias (optional)
* View all activities

**Registered users** can:

* Register criminal
* Add location and activity to criminal
* Edit/Delete activity
* Register cartel

### Compile and Run the Project

You are given a Visual Studio project "CriminalActivities" 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 **Activity**, **ActivityType**, Cartel, City, Criminal and Location, as well as a ready **DbContext**.
* There is already some basic data seeded into the Database.

Internet connection is required to download the referenced NuGet packages.

### Get All Cartels and Related Criminals

Write a REST service to list all Cartels and Related Criminals

* Returns all cartels, ordered by number of criminals in descending order, and then by name alphabetically. Returns name and criminals (name, alias, date of birth and status).

|  |  |  |
| --- | --- | --- |
| Request | GET /api/cartels | |
| Response | 200 OK  [{"name":"Guadalajara", "criminals":[{"name":"Brian Hicks","alias":"The Gun", "birthDate": "1954-06-04T00:00:00","status":0},…] | Returns the list of all cartels (name + criminals in them) as JSON array. |

5 score

### Get Criminal by Name and Alias

Write a REST service to get a criminal by **name** and **alias** (optional). If only the **name** is specified returns the found criminal. If both parameters are specified, the criminal should **match both name and alias**.

|  |  |  |
| --- | --- | --- |
| Request | | Example |
| GET /api/criminals?Name={Name}&Alias={Alias} | | GET /api/criminals?Name=Brian+Hicks&Alias=The+Gun |
| Response | 200 OK  [{"name":"Brian Hicks","alias":"The Gun","birthDate":"1954-06-04T00:00:00","status":0}] | On success, the service returns **200 OK**  Returns: name, alias, date of birth and status |
| Error Response | 400 Bad Request  {"Message":"The request is invalid.","ModelState":{"Name":["The Name field is required."]}} | Returned in case of **missing name** |
| Error Response | 404 Not Found | Returned in case there is no criminal with the specified **name**, or if an **alias** is provided, there is no criminal that matches both |

10 score

### Register Criminal

Logged in users can register a new criminal. **Name** is required, **alias** is optional. Newly registered criminals have **Active status** and a **creator** (the logged user).

|  |  |  |
| --- | --- | --- |
| Request | | Example |
| POST /api/criminals  Content-Type: application/x-www-form-urlencoded  Name={name}  Alias={alias} | | POST /api/criminals  Name=Gosho The Criminal  Alias=Jorjito |
| Response | 201 Created  Location: http://localhost:{port}/api/criminals?Name=Gosho%20The%20Criminal  Body:  {"name":"Gosho The Criminal","alias":"Jorjito","status":0,"registeredBy":"goshko"} | On success, the service returns 201 Created  + the route of the newly created criminal (**only name** as parameter)  + name, alias, status and creator as JSON array |
| Error  Response | 400 Bad Request | Returned in case of a **missing criminal name** |
| Error Response | 401 Unauthorized | Returned in case the user is not logged in. |
| Error Response | 409 Conflict | Returned if a criminal with the same name already exists. |

5 score

### Add Location to Criminal

Write a REST service to **add a location** to an **existing criminal**.

* Sends a request containing the date the individual was last seen (**required**) and the name of the city **(required)**.
* If there is no city with the **specified name**, a new city should be created.

|  |  |  |
| --- | --- | --- |
| Request | POST /api/criminals/{id}/addLocation  Content-Type: application/x-www-form-urlencoded  Date={Date}  CityName={CityName} | Example |
| POST /api/criminals/3/addLocation  Date=2012/2/15  CityName=Petrich |
| Response | 200 OK  {"lastSeen":"2012-02-15T00:00:00","city":"Petrich","criminal":"George Graham"} | Returns 200 OK if request was successful  + a JSON array, holding the **date**, **city name** and **criminal's name** |
| Error Response | 404 Not Found | Returned when there is no criminal with the specified ID |
| Error Response | 400 Bad Request | Returned in case of invalid model state |
| Error Response | 401 Unauthorized | Returned in case the user is not logged in |

10 score

### Add Activity to Criminal

Write a REST service to add an **activity** to an **existing criminal**.

* Sends a request containing the **description**, **type**, **active from** (date of activity start) and **end date** (date of activity end, **optional**).
* The **activity type** and **criminal** should already exist.

|  |  |  |
| --- | --- | --- |
| Request | POST /api/criminals/{id}/addActivity  Content-Type: application/x-www-form-urlencoded  Description={Description}  ActiveFrom={ActiveFrom}  ActiveTo={ActiveTo}  Type={Type} | Example |
| POST /api/criminals/3/addActivity  Description=Sold weapons to the mafia  ActiveFrom=2009/2/16  Type=WeaponsDealing |
| Response | 200 OK  {"description":"Sold weapons to the mafia","activeFrom":"2009-02-16T00:00:00","activeTo":null,"type":"WeaponsDealing","criminal":"George Graham"} | Returns 200 OK if request was successful  + a JSON array, holding the description, active from and active to date, activity type, and criminal name |
| Error Response | 404 Not Found | Returned when there is no criminal with the specified ID or no such activity type |
| Error Response | 400 Bad Request | Returned in case of invalid model state |
| Error Response | 401 Unauthorized | Returned in case the user is not logged in |

10 score

### Edit Activity

Write a REST service to edit an existing activity.

* Sends a request containing the **description**, **type**, **activeFrom** date and **activeTo** date(optional)
* The activity and activity type should exist

|  |  |  |
| --- | --- | --- |
| Request | | Example |
| PUT /api/activities/{id}  Content-Type: application/x-www-form-urlencoded  Description={Description}  ActiveFrom={ActiveFrom}  ActiveTo={ActiveTo}  Type={Type} | | PUT /api/activities/1  Description=Generic douchebag behaviour  ActiveFrom=2010/5/16  ActiveTo=2016/2/22  Type=Racketeering |
| Response | 200 OK  {"description":"Generic douchebag behaviour","activeFrom":"2010-05-16T00:00:00","activeTo":"2016-02-22T00:00:00","type":"Racketeering","criminal":"George Graham"} | On success, the service returns **200 OK** + the data as JSON array. |
| Error Response | 400 Bad Request | Returned in of invalid model state. |
| Error Response | 401 Unauthorized | Returned in there is no logged in user |
| Error Response | 404 Not Found | Returned when the activity or the new activity type do not exist |

5 score

### Delete Activity by ID

Write a REST service to delete an activity by ID.

|  |  |  |  |
| --- | --- | --- | --- |
| Request | DELETE /api/activities/*{id}* | Example | DELETE /api/activities/1 |
| Response | 200 OK  {"message:":"Activity #1 deleted" } | On success, the service returns 200 (OK) + optional human-readable message, explaining that the activity was deleted. | |
| Error Response | 401 Unauthorized | Returned in case there is user logged in. | |
| Error Response | 404 Not Found | Returned when the requested activity does not exist | |

5 score

### Register New Cartel

Write a REST service to create a new cartel. The **name** is required.

* The request body **may** contain a collection of criminals holding their **Id**s. In such case, they should be added as members of the newly created cartel.

|  |  |  |
| --- | --- | --- |
| Request | | Example |
| POST /api/cartels  Content-Type: application/x-www-form-urlencoded  Name={Name}  CriminalIDs={id}  CriminalIDs={id} | | POST /api/cartels  Name=Koza Nostra  CriminalIDs=2  CriminalIDs=3 |
| Response | 200 OK | On success, the service returns **200** **OK** |
| Error Response | 400 Bad Request | Returned in case of missing or invalid cartel data |
| Error Response | 401 Unauthorized | Returned in case the user is not logged in. |
| Error Response | 404 Not Found  {"message":"Criminal with id #2 does not exist"} | Returned in case a criminal with the specified ID does not exist |
| Error Response | 409 Conflict  {"message":"A cartel with the same name already exists"} | Returned in case there is already a cartel with the same name. |
| Error Response | 409 Conflict  {"message":" Criminal with id #3 already has a cartel"} | If one of the requested criminals already has a cartel |

10 score

### View Activities

Write a REST service that **filters activities** by **type** and by **criminal name**.

* Parameters are not required, so if both are missing, return all activities

|  |  |  |
| --- | --- | --- |
| Request | | Example |
| GET api/activities?ActivityType={ActivityType}&CriminalName={CriminalName} | | GET api/activities?ActivityType=WeaponsDealing |
| Response | 200 OK  [{"description":"Sold guns to serbian football fans","activeFrom":"2009-02-16T00:00:00","activeTo":null,"type":"WeaponsDealing","criminal":"George Graham"}, …] | On success, the service returns **200** **OK**  **+** a JSON array, containing the matched activities [description, active from, active to, type, criminal] |

10 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* | | POST /api/account/register  username=maria&password=pAss123 |
| 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

You are given a REST service to **login** existing user by **username** and **password**. Do not touch it, just play with it to learn how it works:

|  |  |  |
| --- | --- | --- |
| Request | | Example |
| POST /api/account/login Content-Type: application/x-www-form-urlencoded  username=*some\_username*&pasword=*some\_password* | | POST /api/account/login  username=maria&password=pAss123 |
| Response | 200 OK  {"access\_token":"VccMrKjEWki…", "token\_type":"bearer",  "userName":"maria", … } | On success, the service returns 200 (OK) + the logged-in 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 invalid username or password). |

### Write Integration Tests for "Register Criminal" Service

Write **integration tests** for the "Register Criminal" REST service. Ensure you cover all interesting cases. Put your tests in a new class called "CriminalsIntegrationTests".

10 score

### Repository and Unit of Work

Before you modify 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 Criminal by Name and Alias" Service

Write **unit tests** with **mocking** for the "**Get Criminal by Name and Alias**" 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 "**Criminal**ControllerTests".

15 score

### Write Unit Tests for "Delete Activity by ID" Service

Write **unit tests** with **mocking** for the "**Delete Activity by ID**" 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**.