

|  |
| --- |
| Task #1 |
| EPM-OGST |

# TASK DESCRIPTION

Create skeleton for the future solution which contains business objects, services and interfaces. For this task, UI doesn’t required. In order to test services, use any well-known tool for testing API (for ex. Fiddler, Postman).

# MODELS

Create the following models:

**Game:**

|  |  |  |
| --- | --- | --- |
| Property name | Type | Comment |
| Key | String | unique; (alias for link crating) |
| Name | string |  |
| Description | String |  |

**Comment:**

|  |  |  |
| --- | --- | --- |
| Property name | Type | Comment |
| Name | String |  |
| Body | String |  |

Game could have comments. User can comment Game or reply on a comment of any user.

**Additional details:**

* “Comment” has “1 – to – Many” relationship with Game. (Game has 0 or many comments; comment always associated with 1 game).
* “Comment” could have replies. Reply should have the following format: “*[Author], Text of Reply*“, where
  + “Author” – Name of the author of parent comment.
  + “[Author]” – Link to the parent comment;

**Genre:**

|  |  |  |
| --- | --- | --- |
| Property name | Type | Comment |
| Name | String | Unique |

Solution should contain predefined genres:

* Strategy
  + RTS
  + TBS
* RPG
* Sports
* Races
  + Rally
  + Arcade
  + Formula
  + Off-road
* Action
  + FPS
  + TPS
  + Misc.
* Adventure
* Puzzle & Skill
* Misc.

**Additional details:**

* Genre could be nested. (Pay attention on predefined genres: Strategy, Races, Action)
* “Game” has “many – to – many” relationship with “Game”.

**PlatformType:**

|  |  |  |
| --- | --- | --- |
| Property name | Type | Comment |
| Type | String | unique |

**Additional details:**

* Store has several types of platform: mobile, browser, desktop, console.
* “Game” has “many–to–many” relation with platform type.

# SERVICES

Service(s) should support the following methods:

* create new game
* edit game
* delete game
* get game by key
* get all games
* add comment to game
* get all comments by game key
* get games by genre
* get games by platformTypes.

# DATA ACCESS LAYER

Repository pattern should be used. In order to work with repositories, Unit of Work should be implemented.

**Additional details:**

* Use the latest stable version of Microsoft Entity framework.
* Use MS SQL Server Express (don’t use mssql compact).

# ADMIN PANEL

The following user actions should be implemented:

* Create game (POST URL: /games/new).
* Edit game (POST URL: /games/update).
* Get game details by key (GET URL: /game/{key}).
* Get all games (GET URL: /games).
* Delete game (POST URL: /games/remove).
* Leave comment for game (POST URL: /game/{gamekey}/newcomment).
* Leave comment for another comment (POST URL: /game/{gamekey}/newcomment)
* Get all comments by game key (POST URL: /game/{gamekey}/comments).
* Download game (jut return any binary file as response) (GET URL: game/{gamekey}/download)

**Additional details:**

- ASP.NET MVC should be used.

# GENERAL REQUIRMENTS

* Use the latest stable version of ASP.NET MVC (empty template).
* Implement error and events logging.
* Solution should contain separated projects for each layer.
* Use principles of object-oriented programming – SOLID
* Use JSON for communication with server.

# OPTIONAL

* Use Output Cache filter to cache get post and get all post response for 1 minute.
* Use global filter to log IP of requests in txt file.
* Use filters for logging performance of services working.

| REVISION HISTORY | | | | | |
| --- | --- | --- | --- | --- | --- |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| 1.0 |  |  | 21-Jul-2018 |  | 21-Jul-2018 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |