Provided Requirements

Write a new micro-service that will allow users to share feedback on their last game session and allow visibility to a live operations team.

Users can rate their session from 1 to 5 and leave a comment. Session id is provided in the url path and the user id is in the header named Ubi-UserId.

Players within the same gaming session rate via the same session id, however a player can only leave one feedback per session.

Following RESTful principles please write the following:

* HTTP endpoint for players to post a new feedback for a session.
* HTTP endpoint to get the last 15 feedbacks left by players and allow filtering by rating.

Explanation

Although the requirements are asking for a new microservice, I have decomposed it into 3 microservices based on differing contexts:

* Users
* Sessions
* Ratings

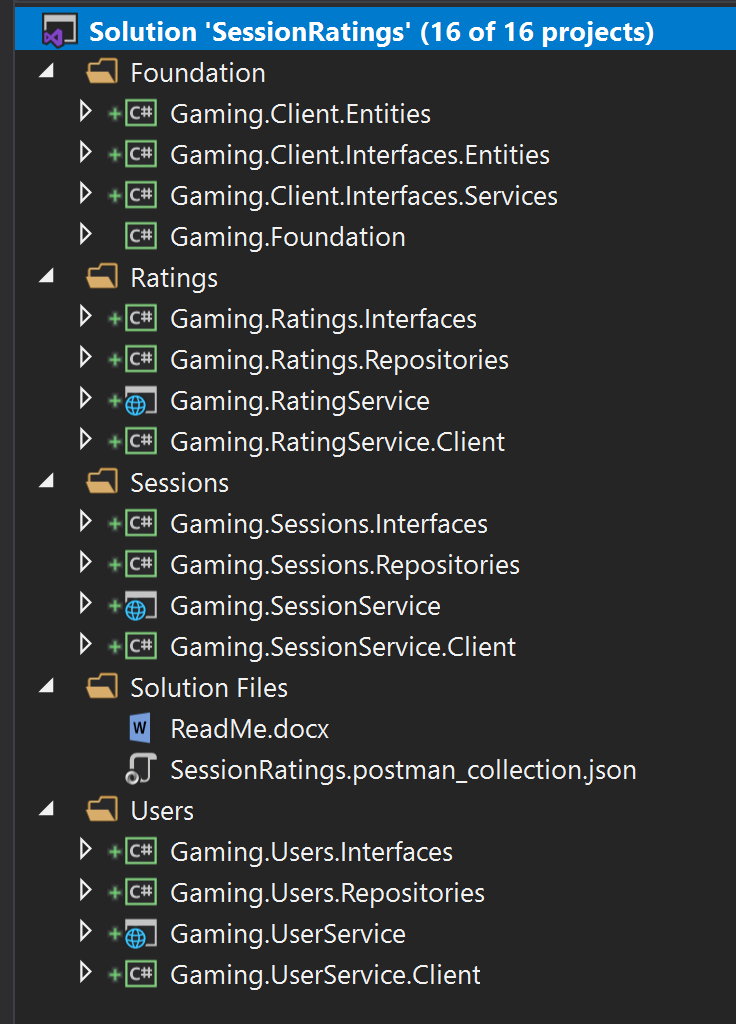
Required software

**Visual Studio 2019 Professional** or **Enterprise** updated to **16.5.4**

**Azure Storage Emulator** : <https://go.microsoft.com/fwlink/?linkid=717179&clcid=0x409>

**Azure Storage Explorer**: <https://azure.microsoft.com/en-us/features/storage-explorer/>

**Postman (Optional)**: <https://www.postman.com/product/api-client>

Solution Breakdown

The Solution consists of 16 Projects:

Foundation contains support projects.

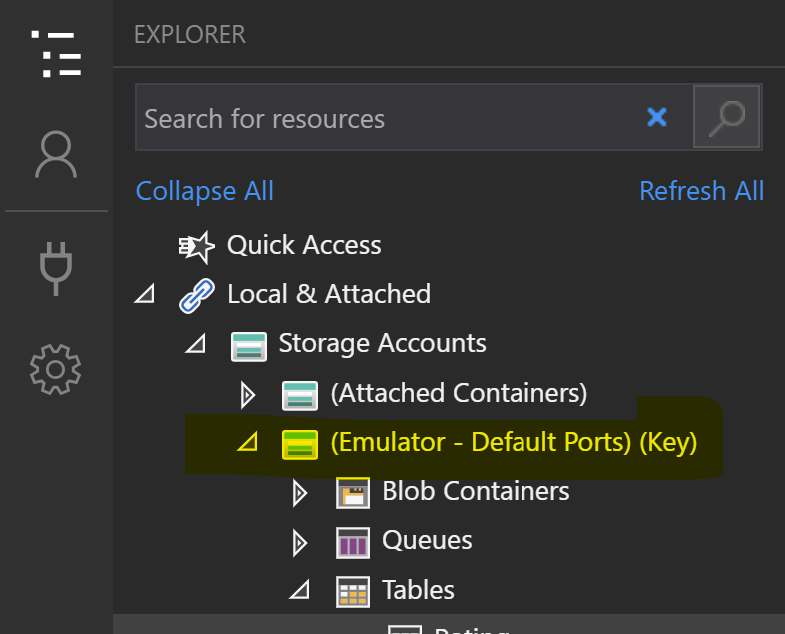
Gaming.Client.Entities – contains client entities. For the sake of time, I kept a single project to contain all of the client entities across all projects.

Gaming.XXXXService.Client – I created 3 client projects in order to provide a client side API to any consumers of the service.

Each solution folder contains 4 projects

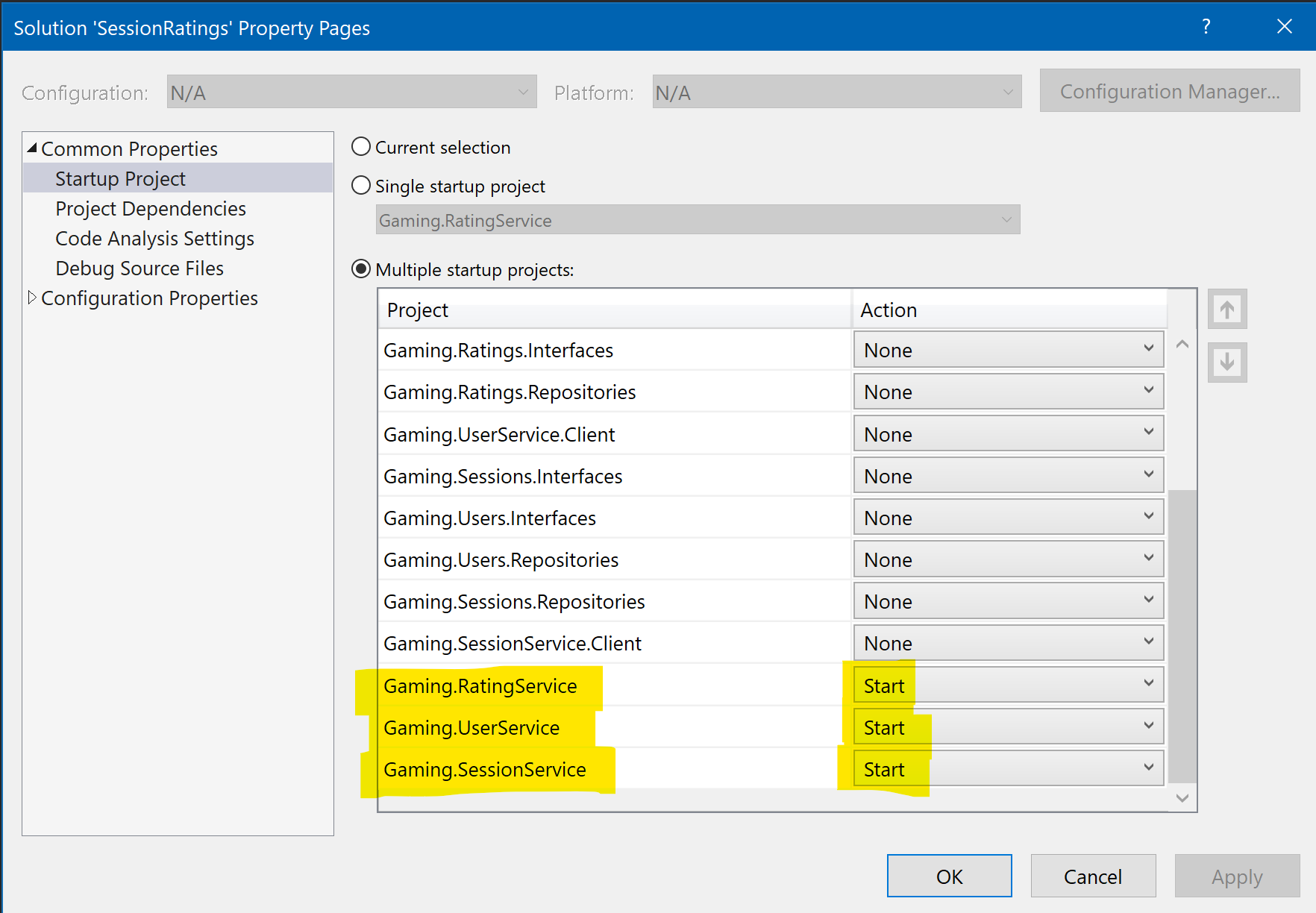
Execution

1. Start Azure Storage Emulator.
2. Open Azure Storage Explorer to verify it has started correctly:



1. In Visual Studio, ensure that the startup project at the solution level is set to ‘Multiple Startup Projects’ with the following set to Start:

* Gaming.RatingService
* Gaming.UserService
* Gaming.SessionService



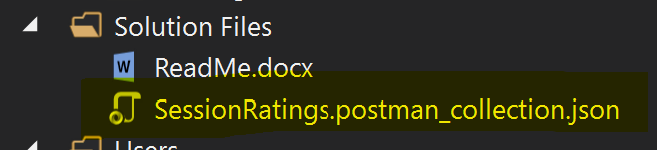
These 3 projects are set to open a browser when they start.

1. In Visual Studio, press Ctrl-F5 to start the solution.
2. Open Postman (or your preferred web client) to test services.

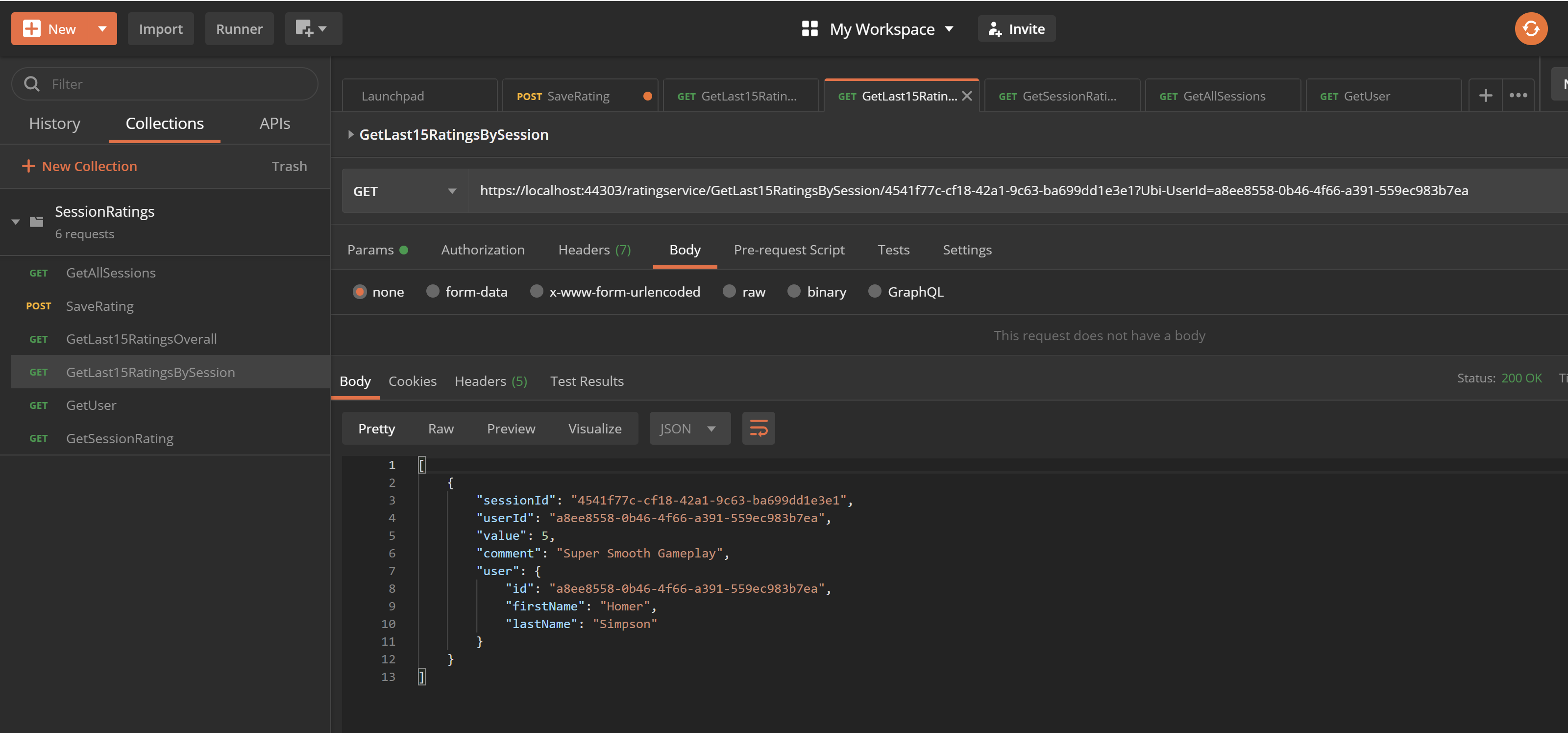
Testing

Using Postman, in the Solution Files folder of the solution you will find a json file that includes all of the client requests for Postman.

Click on File > Import, and choose the file from your local Filesystem.



Choose the service to test and execute:



More services are included in this collection than the requested services in the requirements, and there are more services in the solution than are in this collection as well.

Required Services

**SaveRating Url** - [https://localhost:44303/ratingservice/SaveRating/{userId}](https://localhost:44303/ratingservice/SaveRating/%7buserId%7d)

**Message Header**: Ubi-UserId – {userId} GUID

**Json Payload:**

{

"value": 4,

"comment": "Super Smooth Gameplay"

}

Value – int 1 to 5

Comment, any string.

**GetLast15RatingsOverall Url** - <https://localhost:44303/ratingservice/GetLast15RatingsOverall>

Returns last 15 ratings across all sessions.

Sample Output:

[

    {

        "sessionId": "4541f77c-cf18-42a1-9c63-ba699dd1e3e1",

        "userId": "a8ee8558-0b46-4f66-a391-559ec983b7ea",

        "value": 4,

        "comment": "Super Smooth Gameplay",

        "user": {

            "id": "a8ee8558-0b46-4f66-a391-559ec983b7ea",

            "firstName": "Homer",

            "lastName": "Simpson"

        }

    }

]

**GetLast15RatingsBySession Url** - <https://localhost:44303/ratingservice/GetLast15RatingsBySession/4541f77c-cf18-42a1-9c63-ba699dd1e3e1>

Returns last 15 ratings for a single session.

Same json structure as above.

Criticisms

Several pieces of this solution are incomplete. Undone includes the following:

**Security** – Ubi-UserId is a user account id, however I would use a security token that contains security claims in order to identify which services and data the user should have access to. Security is not implemented. Also service to service communication does not have it’s own token.

**Validation** – Validation should occur at the entity level when possible. This is not implemented.

**Testing** – No Unit or System Tests. Should have a set of system tests to execute once deployed to the test environment. To support testing, there would usually be a set of Mock repositories – this also does not exist.