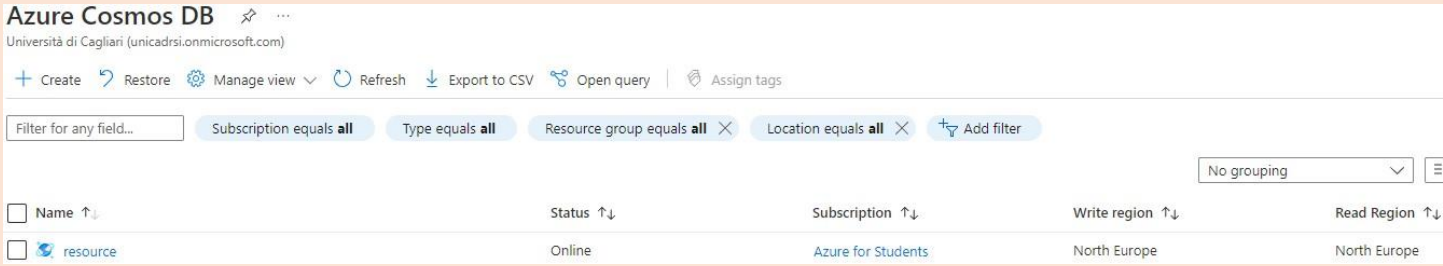
Achievement System

This project is a useful base for developers interested in implementing azure cosmos DB functionality in their Unity Scenes!



Introduction

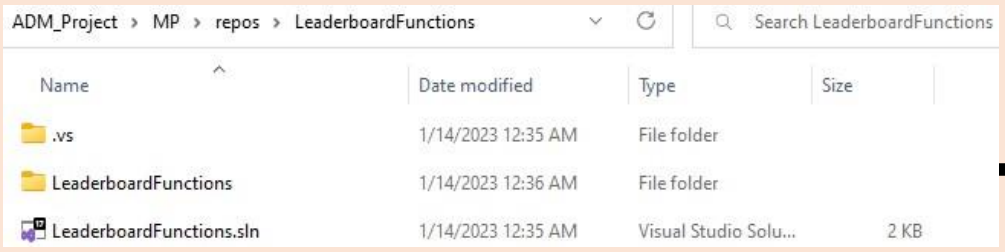
We are going to build an achievement system in Unity. We are doing this with the help of Microsoft Azure Cloud that lets us create azure functions and azure storages for read and write data. We can consider Unity as our running editor application, where we can set our data like we want and then upload that on cloud and show the response in the Unity Scene.

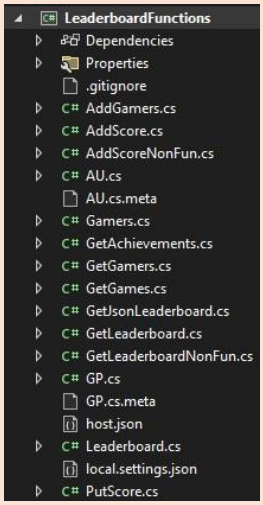
An Azure subscription is mandatory when we are using Azure resources.

From **Azure** we create a **cosmos DB** resource group, function app, storage account.

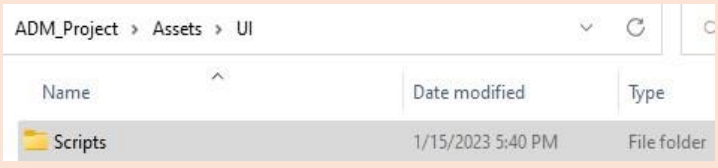
From **Unity** we create an empty 3D project, SLN structure file, file folder("scripts") dedicated to Unity scripts.

LeaderboardFunction.sln

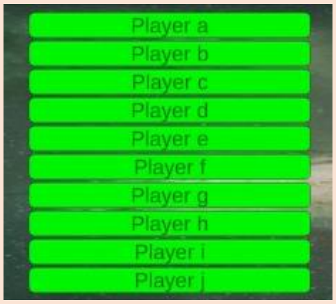




Local Scripts



Scripts and Code



Inside this project "AchievementSystem\Assets\" we have 2 scenes: Menu and **SceneLeaderboard**, we can consider Menu as an example for understanding a local login access between a choice of 10 players, we choose our player , once we click the button then the scene change preserving the player that we chose and this can be easily extended to more complex login methods by coding inside the single "Menu.cs" script !

SceneLeaderboard is our main achievement system scene where we can debug and manage data like leaderboards, games, and achievements for each player. Multiple scripts allow it to work, it can be considered as a real case scenario working multiplatform system, can be extended for real platforms like Xbox, steam, PlayStation, google play etc... by using their own SDK!

Inside "AchivementSystem\Assets\UI\Scripts" we can explore the pseudo-CRUD system that is composed of three fundamental classes: *Gamers*, *GP*, *AU*.

Gamers has a list where every single gamer ordered by GamerSingle class owns the following attributes: name, TotAU(Total Achievement Unlocked), TotScoreD(Sum of all Score Difficulty), numberGP(Number of Games Played), TotScoreR(total ratio from all games).

GP (Game Played) has a list where every single Game played by a player has: nameGame, gameScoreD(Sum of all the score difficulty of all the achievement), gameScoreR(gameRateo*gameScoreD), gameRateo(Sum of all ratios for each achievement), AOPTPTG(Amount of players that played that game).

AU (Achievements Unlocked) has a list for each game, containing id, timestamp, scoreD(Difficulty), scoreR((achRateo) *scoreD), achRateo (single Rateo of an achievement =achievementRarity/AOPTUTA), flag, AOPTUTA(Amount Of Players That Unlocked That Achievement).

So, every Player has "tot" games with "tot" achievements, all this data is managed by script, inside "AchivementSystem\Assets\UI\Scripts", we have our three classes: "Gamers.cs", "GP.cs", "AU.cs" then "Leaderboard.cs" that has a list of single leaderboards for each player with the following attributes: name and score.


```
{"gamerSingleList":[
{"name":"player a","TotAU":100,"TotScoreD":10000,"numberGP":2,"TotScoreR":20000,"Rateo":2.0 },
{"name":"player b","TotAU":100,"TotScoreD":10000,"numberGP":2,"TotScoreR":20000,"Rateo":2.0}
                    ]
}
```

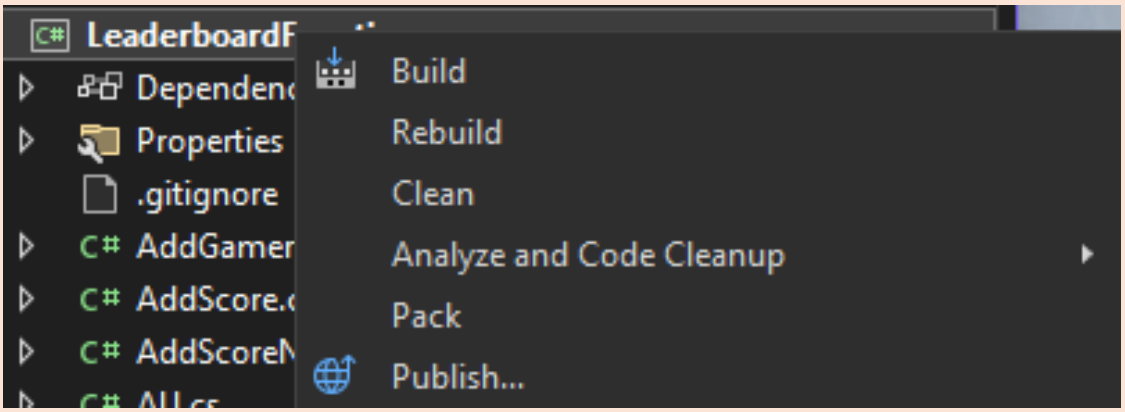

Figure 1 Gamers as a JSON

Scripts and Code

“LeaderboardUI.cs” is a sample for showing the Leaderboard data with a UI! It is attached to UI game objects in the scene.

“**WebRequest.cs**” is an important script from where we can create our CRUD functions! Actually we have some complex one for json handling, get, post and put, all good working examples.

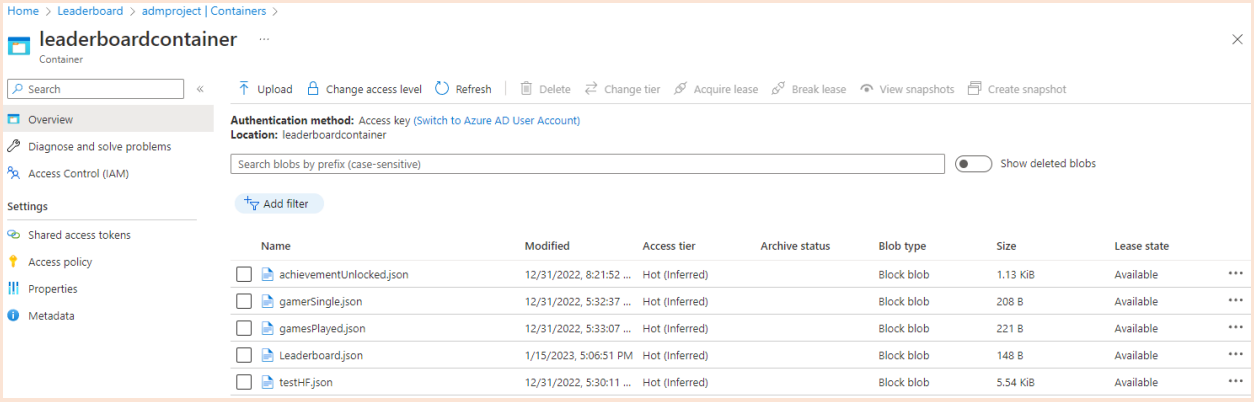
“**TestL.cs**” is the core script, it is attached to an empty object in the scene, used for testing, it’s a bit tricky to explain, pay a bit of attention here. We call functions from **WebRequest** scripts, they take the **URL** of the functions **published** from **SLN** to the Azure **Function App** as input! So, we manage all the data from here based on the SLN functions that we published! Therefore, we can send, receive and edit data asynchronously from requests and responses, it means that we have our data saved in Azure Cosmos DB.

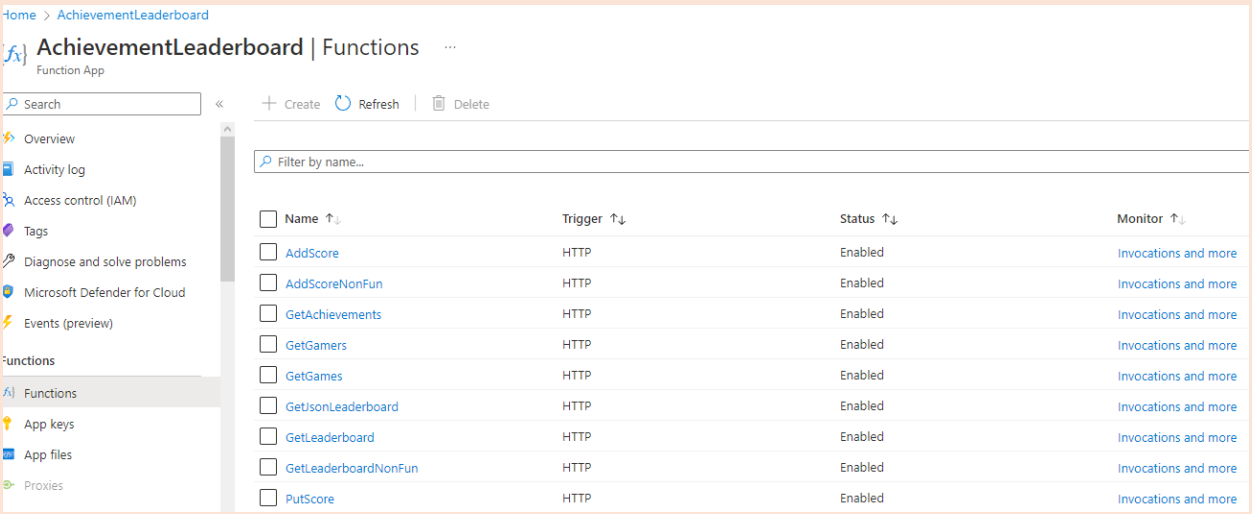


Scripts and Code

In the SLN we have our local functions, and the same classes that we want to manage, so we have the same three classes as before “Gamers.cs”, “GP.cs”, “AU.cs”, “Leaderboard.cs” (even here, they are the same but not duplicate, because are inside the namespace of the function app).

The classes declared before having their values studied to be compatible with JSON file in the blob blocks inside the Containers of Azure.





In SLN we have some working examples of get functions that parse the JSON into code so we can debug it as we want

“GetGamers.cs”, “GetGP.cs”, “GetAU.cs”, “GetLeaderboard.cs”→all using get or getJson function from WebRequest

“PutScore.cs” →put from WebRequest

“AddScore.cs” →post from WebRequest

The game scene: SceneLeaderboard

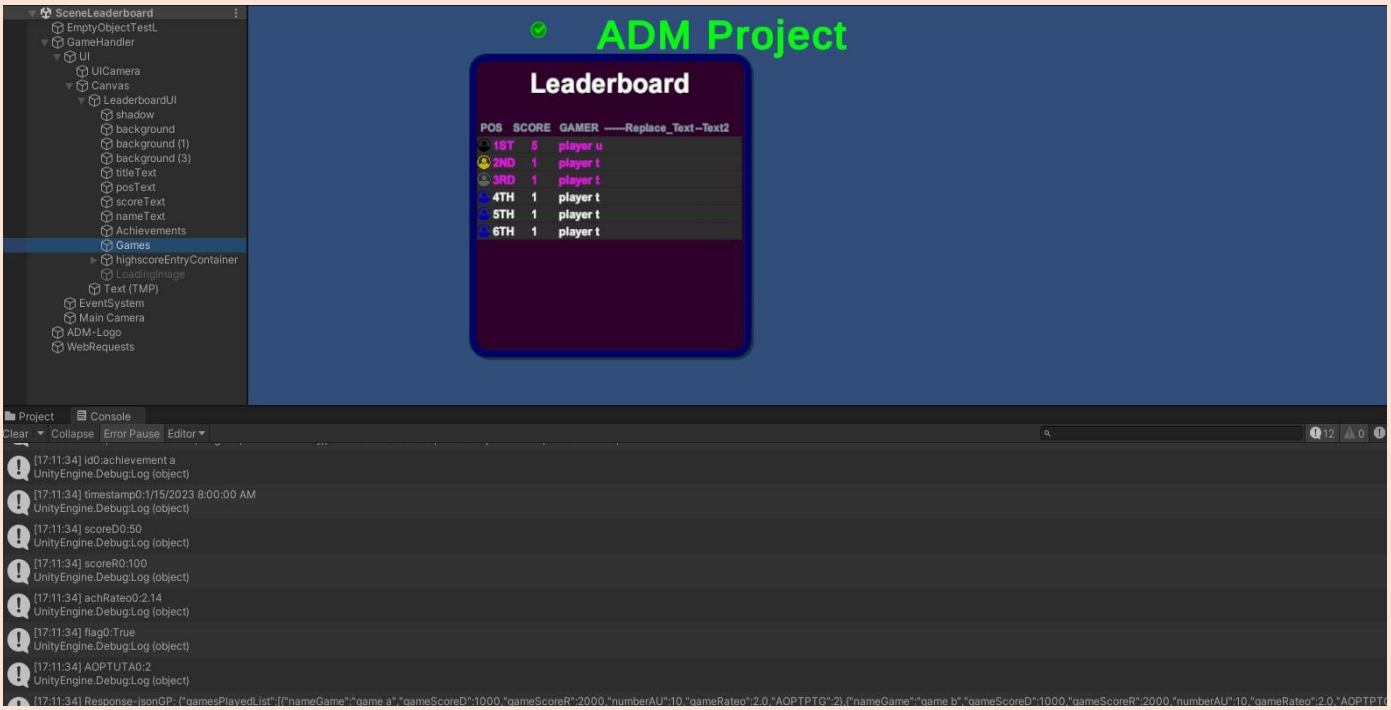
Testing the Scene

Pressing ‘U’ will **retrieve** and print **all the data** for every single player-games-achievements as JSON and the parsed code of each field.

An example of **get** from the leaderboard can be tested by pressing ‘T’, it is also ordered by leaderboardUI functions, players are ordered by score in the example, it’s the most interesting final part and can be customized as we want!

And an example of **post** can be tested by pressing ‘Y’, it adds players and score.

Pressing ‘R’ for a **put** on the first field of the leaderboard.



A better explanation of how it works will be discussed in the oral discussion of the project.