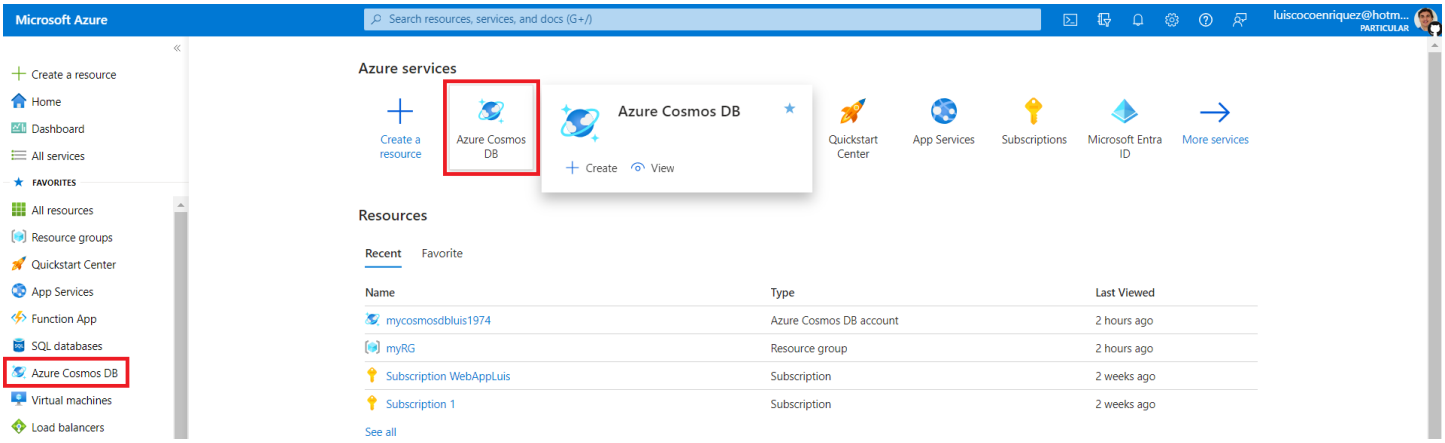
How to create a .NET8 WebAPI CRUD Azure CosmosDB Microservice

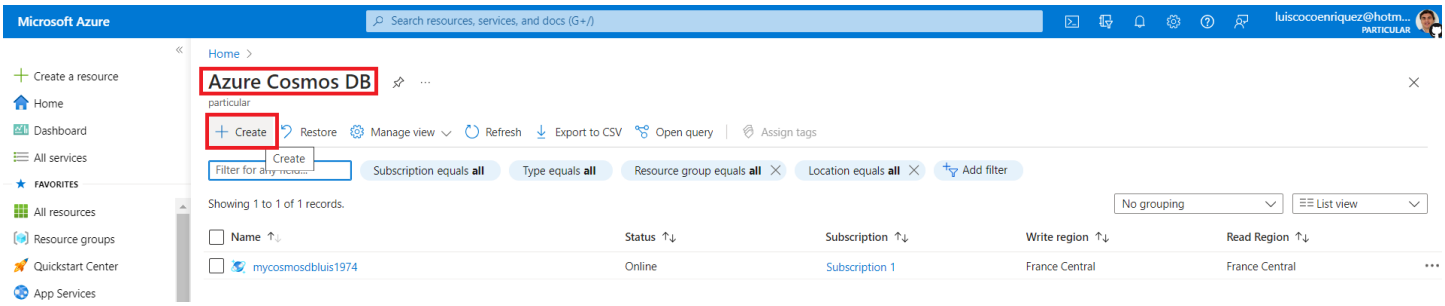
The code for this example is available in this github repo:
https://github.com/luiscoco/MicroServices_dotNET8_CRUD_WebAPI-AzureCosmosDB

1. Create Azure CosmosDB

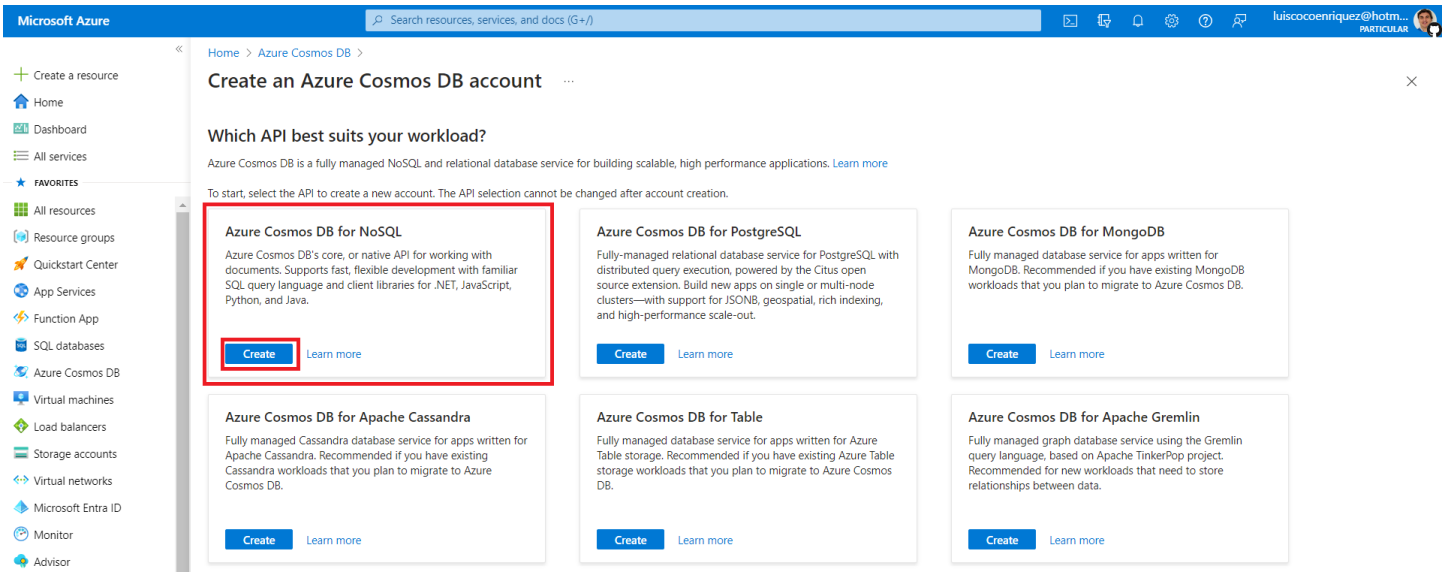
We navigate to **Azure CosmosDB** service and we click in this option to create a new account:



We click on **Azure CosmosDB account** button



Now we select the option **Azure Cosmos DB for NoSQL**, and we press the **create** button



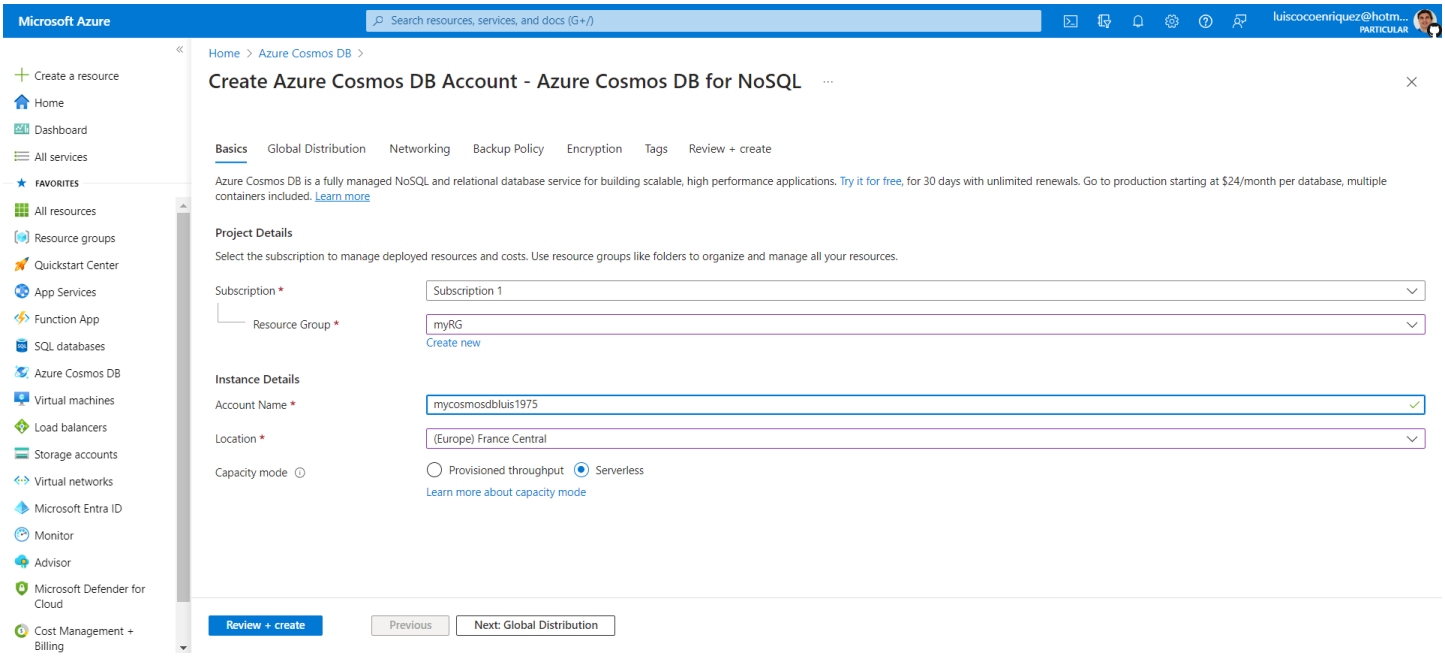
In following screen we input the required data for creating the service

We create a new **ResourceGroup name**: myRG

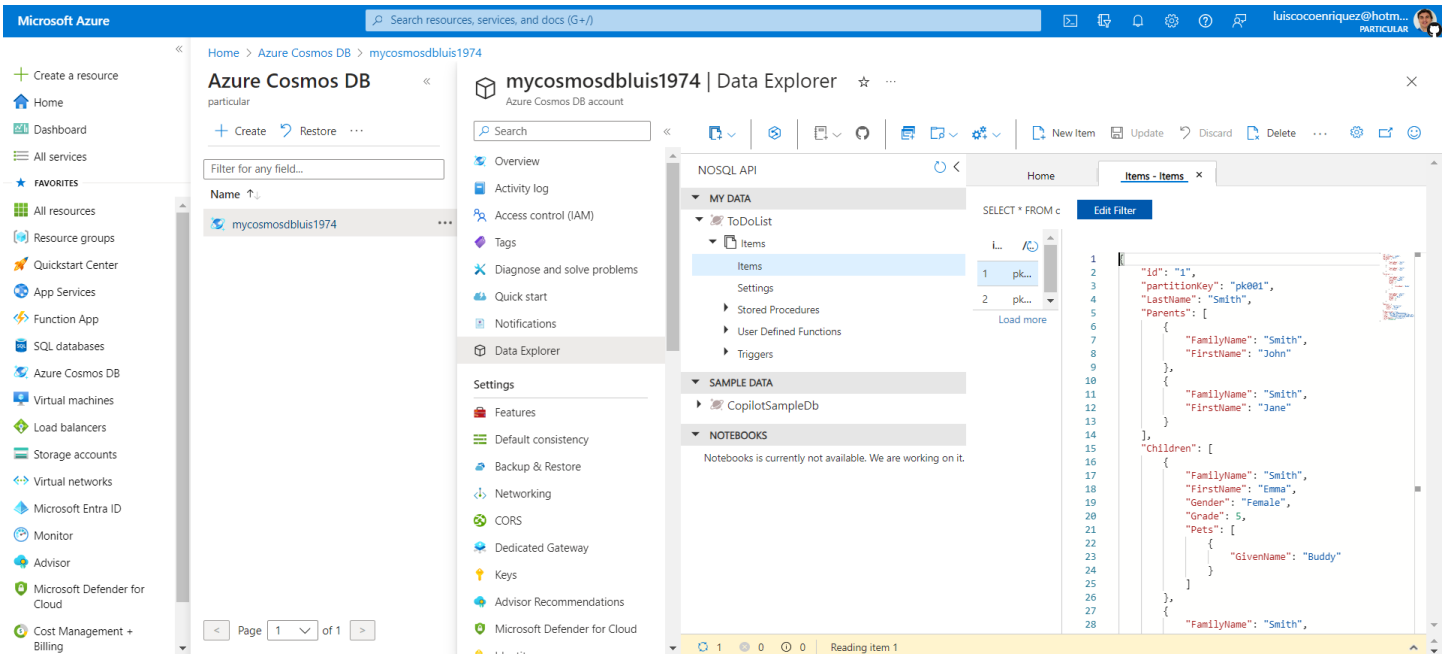
We set the **account name**: mycosmosdbluis1974

We choose the service **location**: France Central

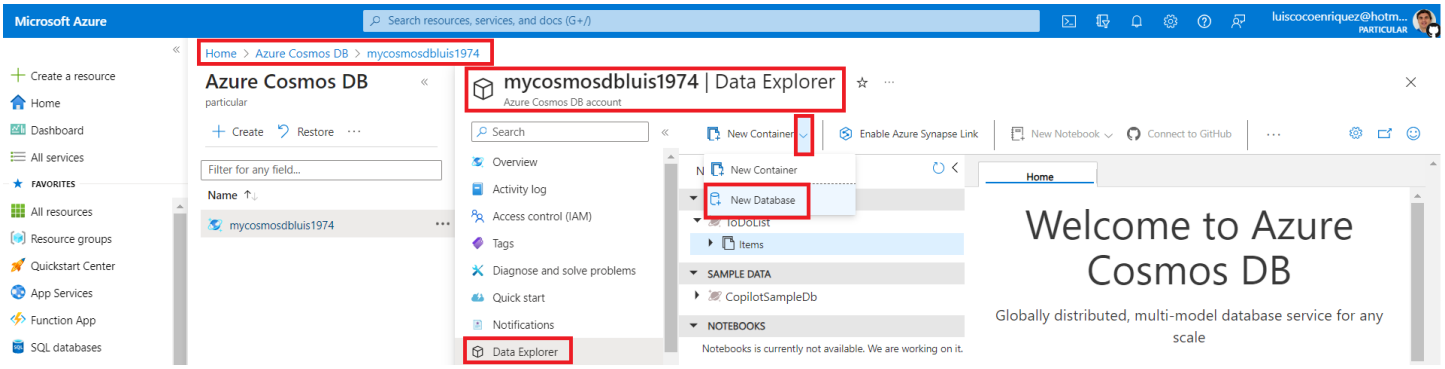
Capacity mode: **serverless**



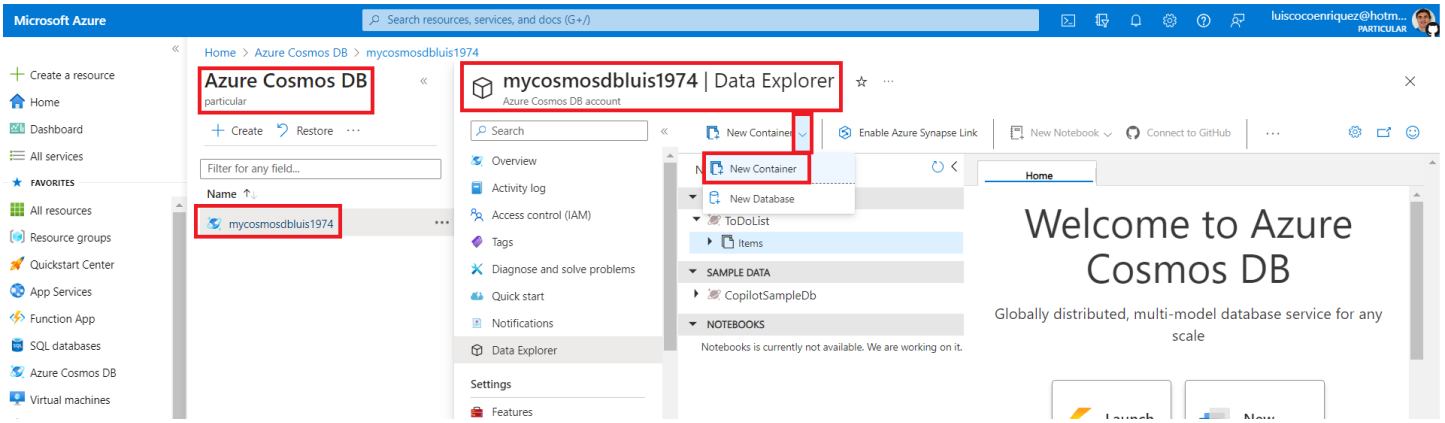
We navigate to the **Data Explorer** page and we create a **New Database** and a **New Container**



We first create a **New Database**. We input the **DatabaseId**



We also create a **New Container**



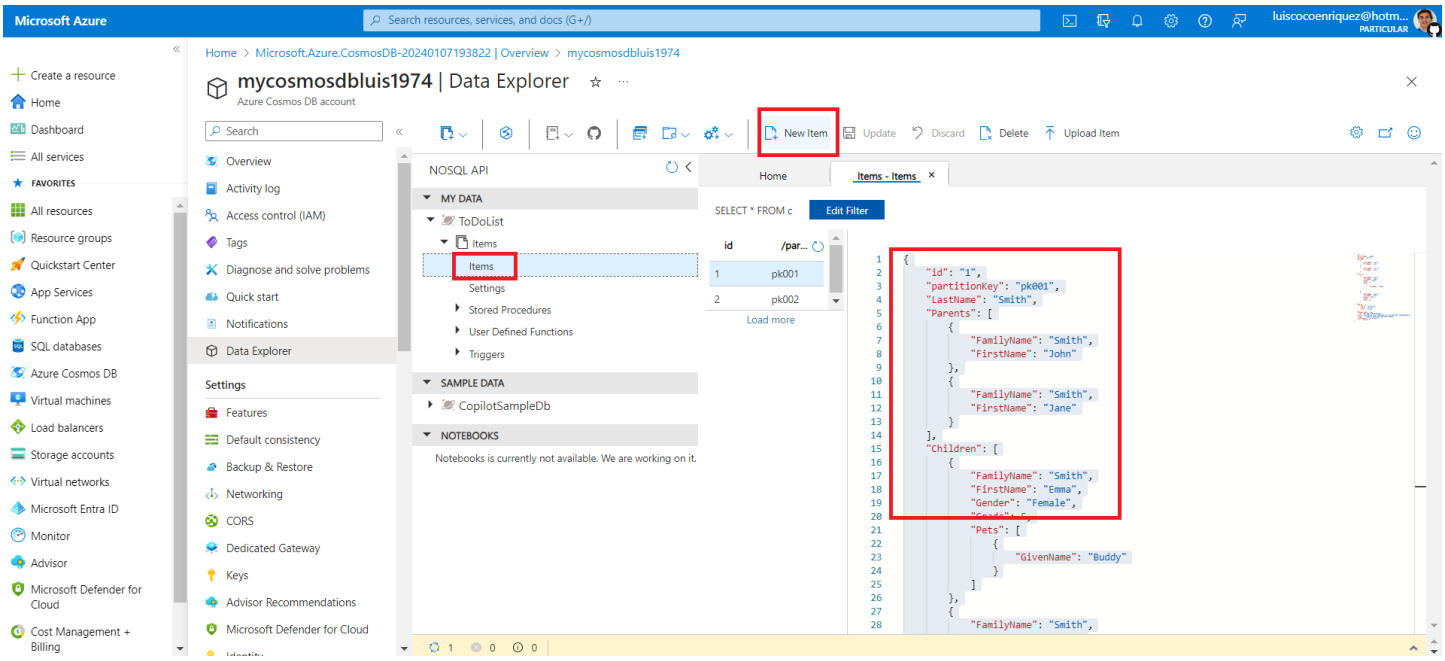
2. Insert the new items in the Azure CosmosDB

This is the **new item** json file:

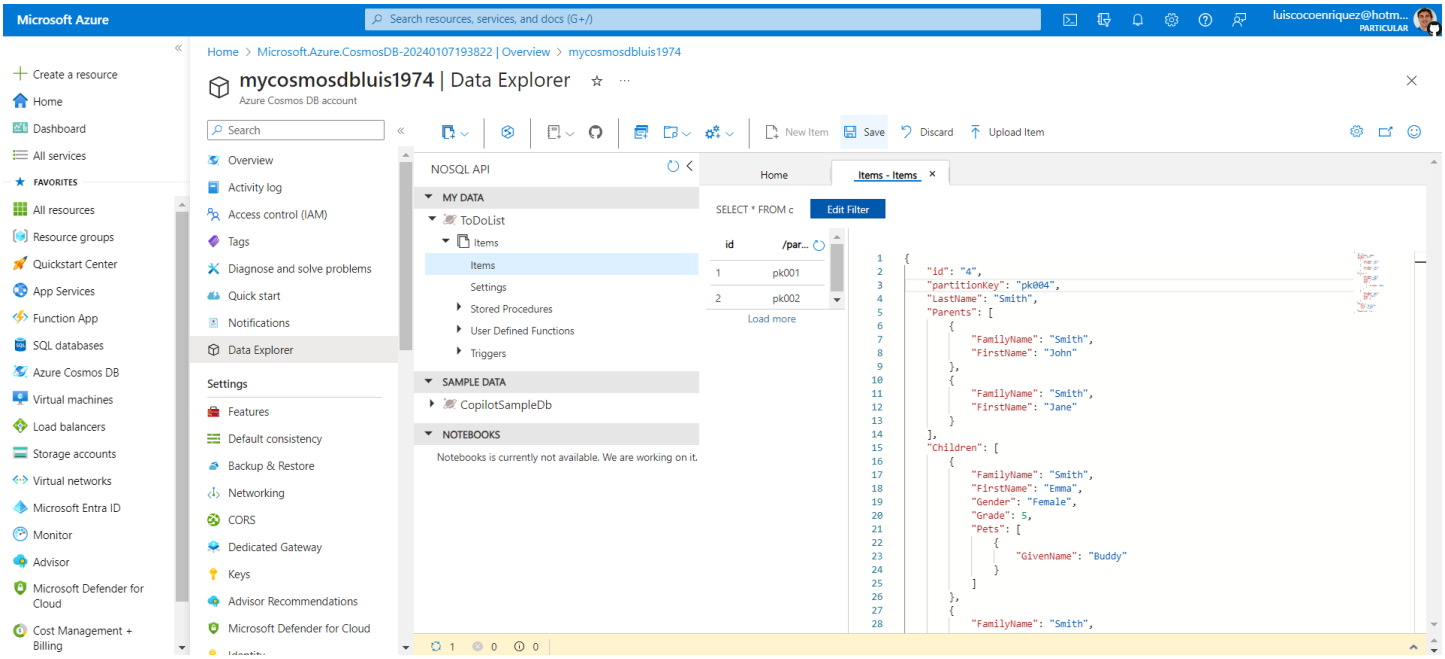
```
{
    "id": "1",
    "partitionKey": "pk001",
```

```
    "LastName": "Smith",
    "Parents": [
        {
            "FamilyName": "Smith",
            "FirstName": "John"
        },
        {
            "FamilyName": "Smith",
            "FirstName": "Jane"
        }
    ],
    "Children": [
        {
            "FamilyName": "Smith",
            "FirstName": "Emma",
            "Gender": "Female",
            "Grade": 5,
            "Pets": [
                {
                    "GivenName": "Buddy"
                }
            ]
        },
        {
            "FamilyName": "Smith",
            "FirstName": "Mike",
            "Gender": "Male",
            "Grade": 8,
            "Pets": []
        }
    ],
    "Address": {
        "State": "California",
        "County": "Orange",
        "City": "Irvine"
    },
    "IsRegistered": true
}
```

We click in **Items** and then **New Item**



Then we copy and paste the json content and press **Save** button

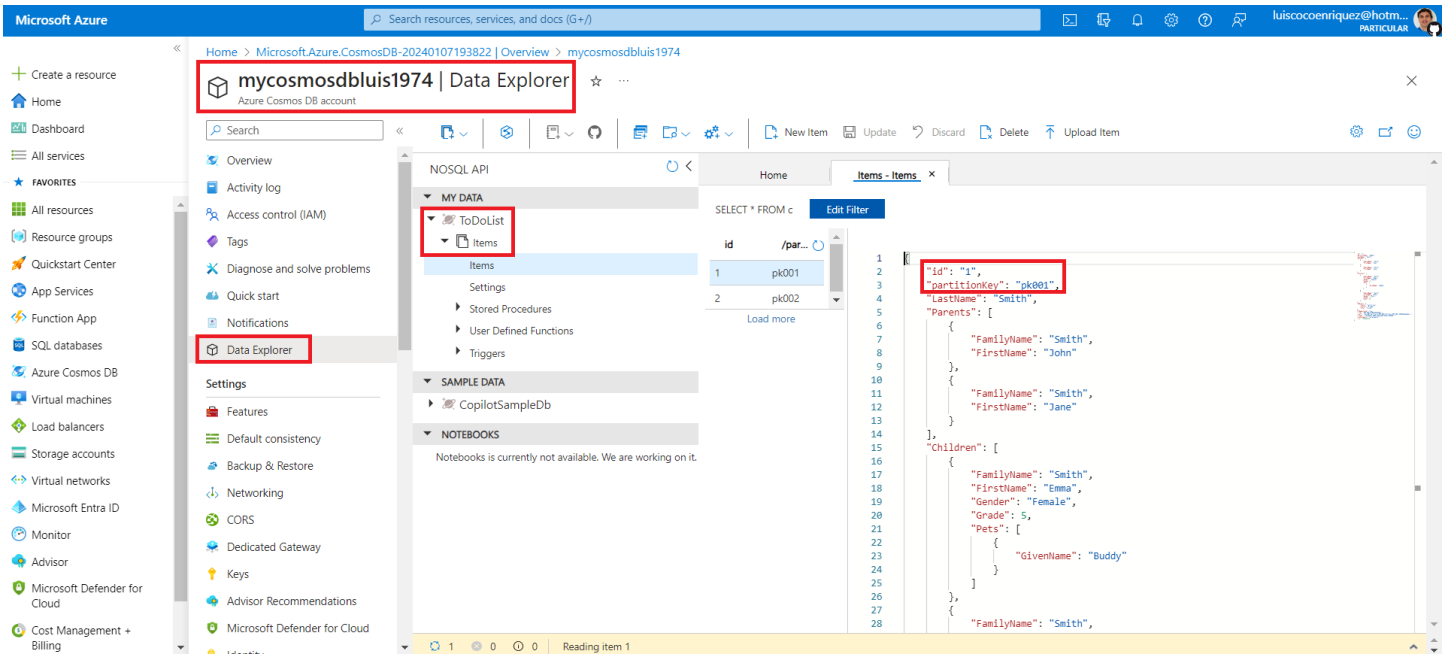


3. appsettings.json

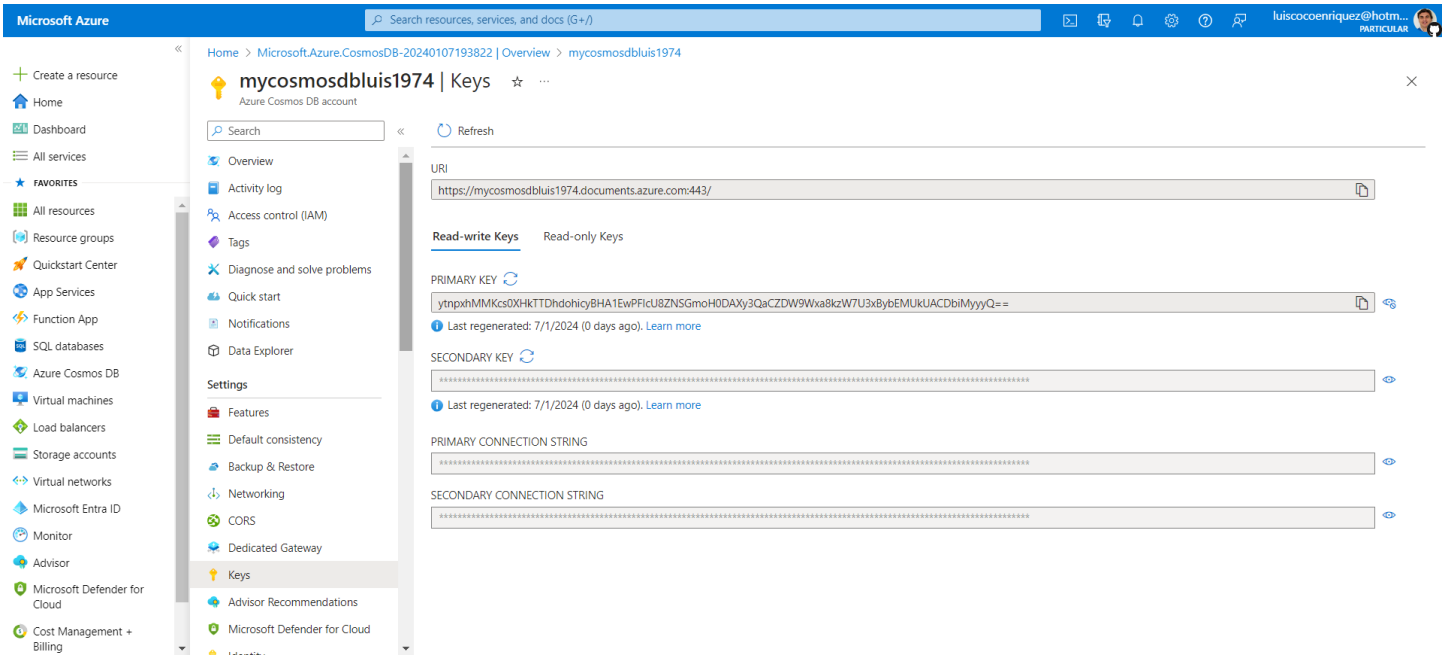
We copy the **DatabaseId** and **ContainerId**

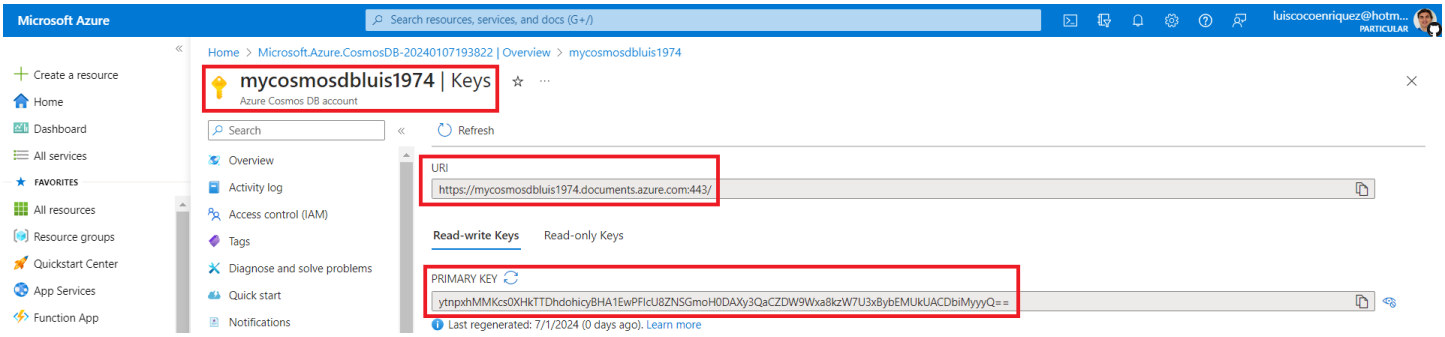
DatabaseId: ToDoList

ContainerId: Items

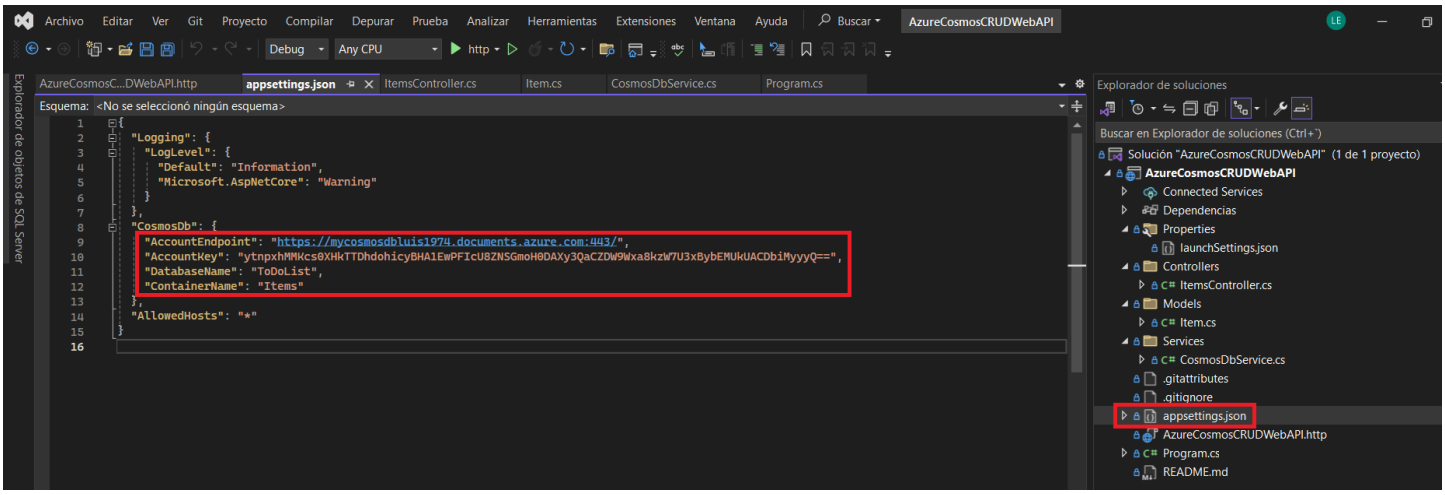


We copy the **URI** and **Primary Key**





This is the **appsettings.json** file




```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "CosmosDb": {
    "AccountEndpoint": "https://mycosmosdbluis1974.documents.azure.com:443/",
    "AccountKey": "ytnpxhMMKcs0XHkTTDhdohicyBHA1EwPFIcU8ZNSGmoH0DAXy3QaCZDW9Wxa8kzW7U3xBybEMUk
    "DatabaseName": "ToDoList",
    "ContainerName": "Items"
  },
  "AllowedHosts": "*"
}
```

4. Program.cs

```
using Microsoft.Azure.Cosmos;
using AzureCosmosCRUDWebAPI.Services;
using Microsoft.Extensions.Configuration;
using Microsoft.AspNetCore.Diagnostics;
using Newtonsoft.Json;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllers();

// Cosmos DB Configuration
var cosmosDbConfig = builder.Configuration.GetSection("CosmosDb");
builder.Services.AddSingleton<CosmosClient>(s =>
    new CosmosClient(cosmosDbConfig["AccountEndpoint"], cosmosDbConfig["AccountKey"]));
builder.Services.AddSingleton<CosmosDbService>(s =>
    new CosmosDbService(s.GetRequiredService<CosmosClient>(), cosmosDbConfig["DatabaseName"],
```

```
// Add other necessary services like Swagger if needed
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseDeveloperExceptionPage();
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseExceptionHandler(a => a.Run(async context =>
{
    var exceptionHandlerPathFeature = context.Features.Get<IExceptionHandlerPathFeature>();
    var exception = exceptionHandlerPathFeature?.Error;

    var result = JsonConvert.SerializeObject(new { error = exception?.Message });
    context.Response.ContentType = "application/json";
    await context.Response.WriteAsync(result);
}));

app.UseHttpsRedirection();
app.UseAuthorization();
app.MapControllers();

app.Run();
```

5. Models

Item.cs

```
using Newtonsoft.Json;

namespace AzureCosmosCRUDWebAPI.Models
{
    public class Family
    {
        [JsonProperty(PropertyName = "id")]
        public string Id { get; set; }
        [JsonProperty(PropertyName = "partitionKey")]
        public string PartitionKey { get; set; }
        public string LastName { get; set; }
        public Parent[] Parents { get; set; }
        public Child[] Children { get; set; }
        public Address Address { get; set; }
        public bool IsRegistered { get; set; }
```

```
        public override string ToString()
        {
            return JsonConvert.SerializeObject(this);
        }
    }

    public class Parent
    {
        public string FamilyName { get; set; }
        public string FirstName { get; set; }
    }

    public class Child
    {
        public string FamilyName { get; set; }
        public string FirstName { get; set; }
        public string Gender { get; set; }
        public int Grade { get; set; }
        public Pet[] Pets { get; set; }
    }

    public class Pet
    {
        public string GivenName { get; set; }
    }

    public class Address
    {
        public string State { get; set; }
        public string County { get; set; }
        public string City { get; set; }
    }

}
```

6. CosmosDbService.cs

Pay attention we set the **PartitionKey** "/Id"

```
using Microsoft.Azure.Cosmos;
using AzureCosmosCRUDWebAPI.Models;
using System.Collections.Generic;
using System.Threading.Tasks;
using System.Linq;
using Newtonsoft.Json;

namespace AzureCosmosCRUDWebAPI.Services
{
    public class CosmosDbService
    {
```

```
    private Container _container;

    public CosmosDbService(CosmosClient dbClient, string databaseName, string containerNam
    {
        this._container = dbClient.GetContainer(databaseName, containerName);
    }

    public async Task AddItemAsync(Family item)
    {
        try
        {
            await this._container.CreateItemAsync(item, new PartitionKey(item.PartitionKey
        }
        catch (CosmosException ex)
        {
            Console.WriteLine($"Cosmos DB error in AddItemAsync. Status code: {ex.StatusCo
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error in AddItemAsync: {ex.Message}, StackTrace: {ex.Stack
            throw;
        }
    }

    // Read an item by id
    public async Task<Family> GetItemAsync(string id, string partitionKeyValue)
    {
        try
        {
            ItemResponse<Family> response = await this._container.ReadItemAsync<Family>(id
            return response.Resource;
        }
        catch (CosmosException ex) when (ex.StatusCode == System.Net.HttpStatusCode.NotFou
        {
            return null;
        }
    }

    // Update an existing item
    public async Task UpdateItemAsync(string id, Family item)
    {
        await this._container.UpsertItemAsync(item, new PartitionKey(id));
    }

    // Delete an item
    public async Task DeleteItemAsync(string id, string partitionKeyValue)
    {
        await this._container.DeleteItemAsync<Family>(id, new PartitionKey(partitionKeyVal
    }

    // List all items
    public async Task<IEnumerable<Family>> GetItemsAsync(string queryString)
```

```
        {
            var query = this._container.GetItemQueryIterator<Family>(new QueryDefinition(query
            List<Family> results = new List<Family>();
            while (query.HasMoreResults)
            {
                var response = await query.ReadNextAsync();
                results.AddRange(response.ToList());
            }
            return results;
        }
    }
}
```

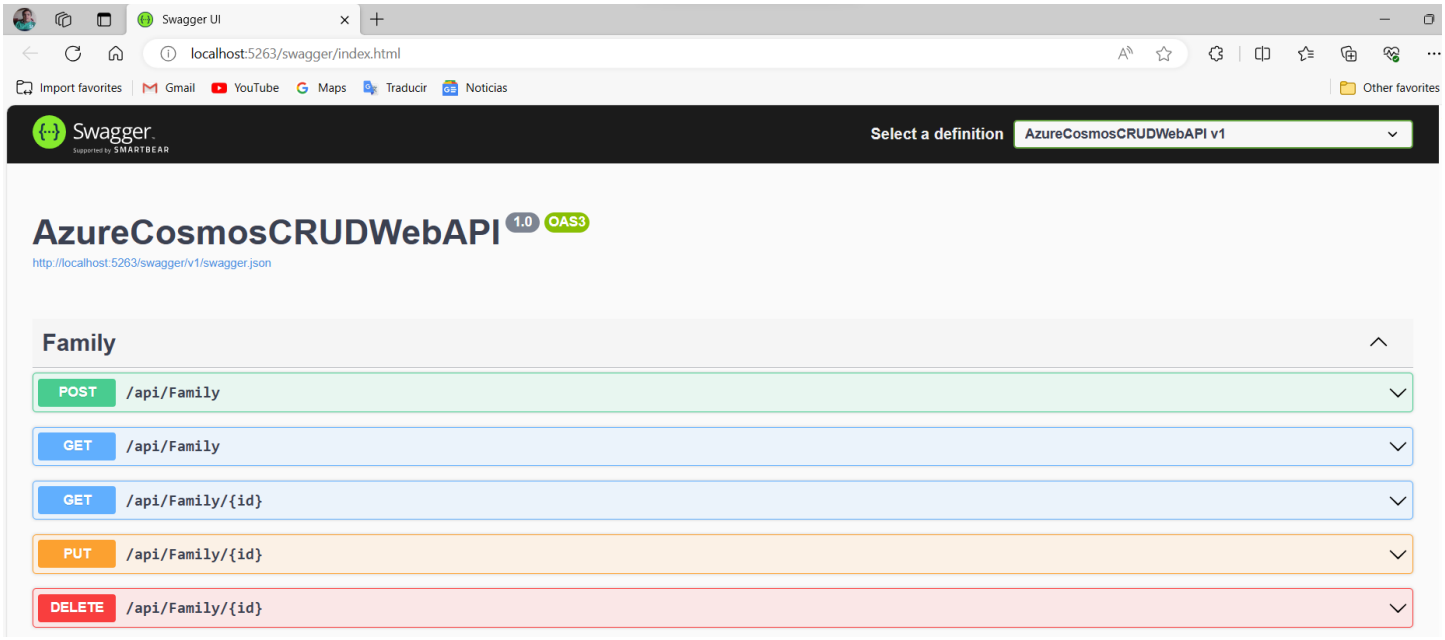
7. Controller

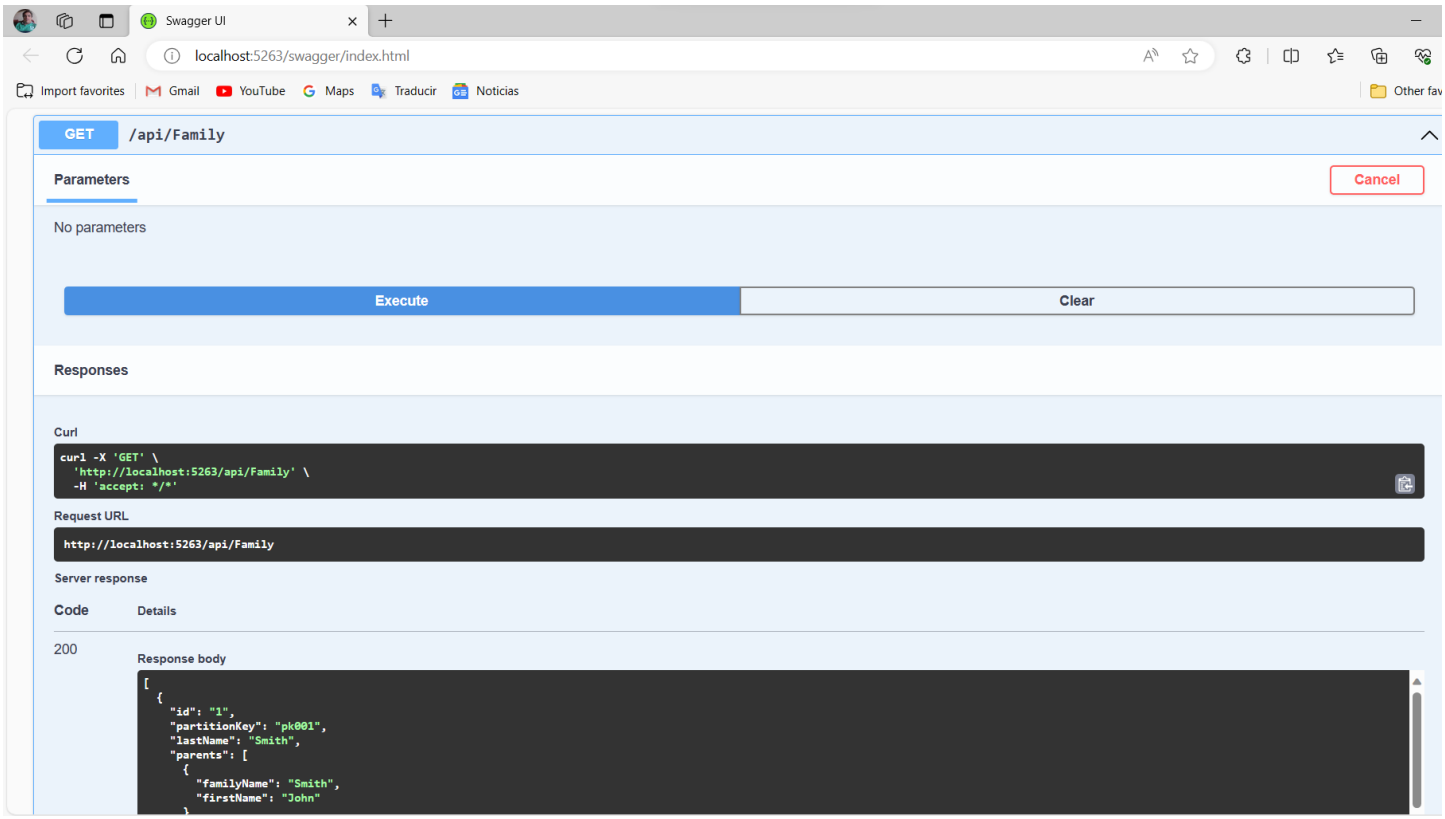
ItemsController.cs

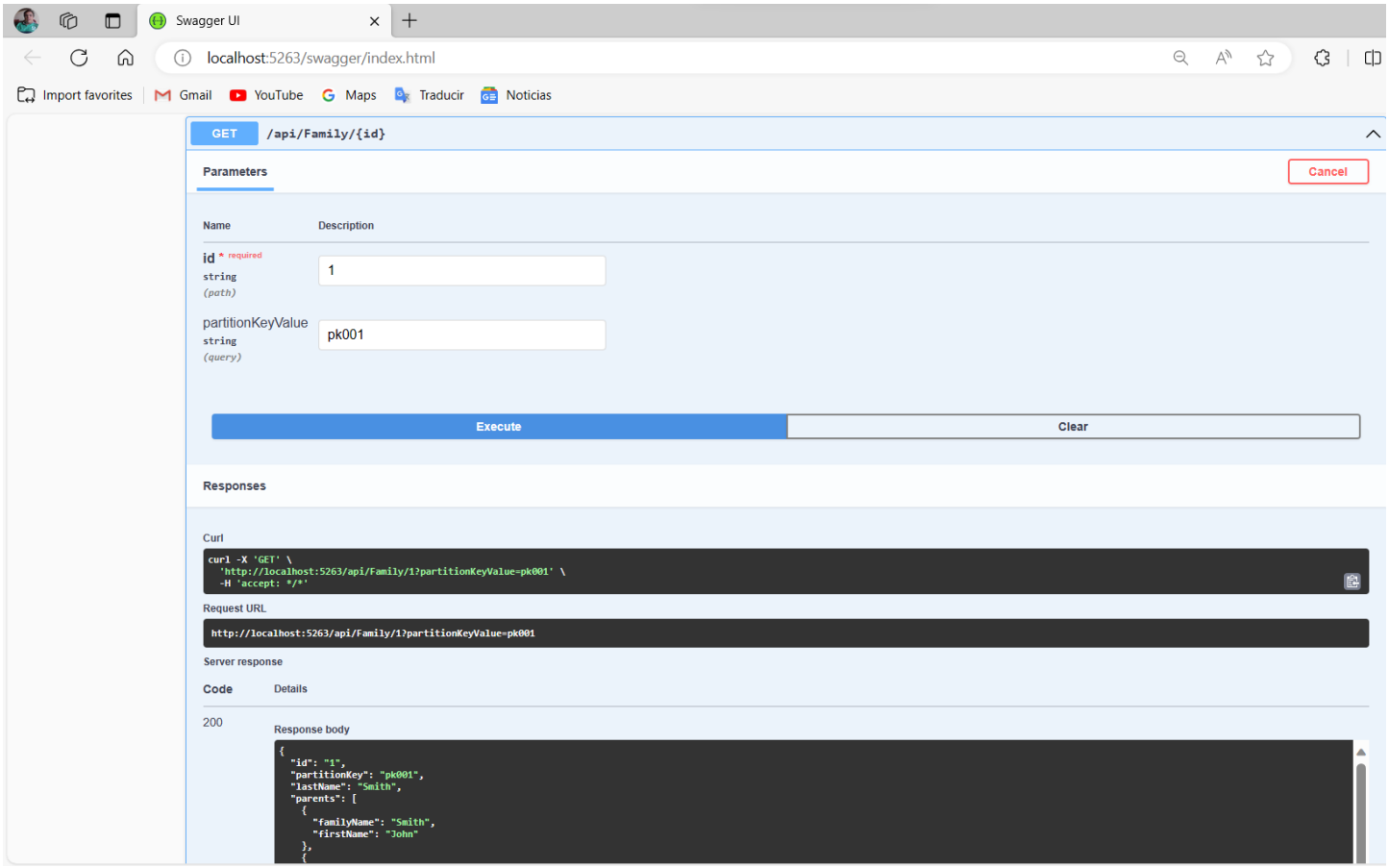
```
using Microsoft.AspNetCore.Mvc;
using AzureCosmosCRUDWebAPI.Models;
using AzureCosmosCRUDWebAPI.Services;
using System.Threading.Tasks;

namespace AzureCosmosCRUDWebAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class FamilyController : ControllerBase
    {
        private readonly CosmosDbService _cosmosDbService;

        public FamilyController(CosmosDbService cosmosDbService)
        {
            _cosmosDbService = cosmosDbService;
        }

        // POST api/family
        [HttpPost]
        public async Task<IActionResult> Create([FromBody] Family family)
        {
            if (family == null)
            {
                return BadRequest("Family cannot be null");
            }

            await _cosmosDbService.AddItemAsync(family);
            return CreatedAtAction(nameof(Get), new { id = family.Id }, family);
        }

        // GET api/family/{id}
        [HttpGet("{id}")]
```

```
        public async Task<IActionResult> Get(string id, string partitionKeyValue)
        {
            var family = await _cosmosDbService.GetItemAsync(id, partitionKeyValue);
            if (family == null)
            {
                return NotFound();
            }

            return Ok(family);
        }

        // PUT api/family/{id}
        [HttpPut("{id}")]
        public async Task<IActionResult> Update(string id, [FromBody] Family family)
        {
            if (family == null || family.Id != id)
            {
                return BadRequest();
            }

            await _cosmosDbService.UpdateItemAsync(id, family);
            return NoContent();
        }

        // DELETE api/family/{id}
        [HttpDelete("{id}")]
        public async Task<IActionResult> Delete(string id, string partitionKeyValue)
        {
            await _cosmosDbService.DeleteItemAsync(id, partitionKeyValue);
            return NoContent();
        }

        // GET api/family
        [HttpGet]
        public async Task<IActionResult> GetAll()
        {
            var families = await _cosmosDbService.GetItemsAsync("SELECT * FROM c");
            return Ok(families);
        }
    }
}
```

8. Application verification


Swagger UI
localhost:5263/swagger/index.html
Import favorites
Gmail
YouTube
Maps
Traducir
Noticias
Other favorites
Swagger
Select a definition
AzureCosmosCRUDWebAPI v1
AzureCosmosCRUDWebAPI
1.0
OAS3
http://localhost:5263/swagger/v1/swagger.json
Family
POST /api/Family
GET /api/Family
GET /api/Family/{id}
PUT /api/Family/{id}
DELETE /api/Family/{id}


Swagger UI
localhost:5263/swagger/index.html
Import favorites
Gmail
YouTube
Maps
Traducir
Noticias
GET /api/Family
Parameters
Cancel
No parameters
Execute
Clear
Responses
Curl
curl -X 'GET' \
'http://localhost:5263/api/Family' \
-H 'accept: */*'
Request URL
http://localhost:5263/api/Family
Server response
Code
Details
200
Response body
[
{
"id": "1",
"partitionKey": "pk001",
"lastName": "Smith",
"parents": [
{
"familyName": "Smith",
"firstName": "John"


Swagger UI
localhost:5263/swagger/index.html
Import favorites
Gmail
YouTube
Maps
Traducir
Noticias
GET
/api/Family/{id}
Parameters
Cancel
Name
Description
id * required
string
(path)
1
partitionKeyValue
string
(query)
pk001
Execute
Clear
Responses
Curl
curl -X 'GET' \
'http://localhost:5263/api/Family/1?partitionKeyValue=pk001' \
-H 'accept: */*'
Request URL
http://localhost:5263/api/Family/1?partitionKeyValue=pk001
Server response
Code
Details
200
Response body
{
"id": "1",
"partitionKey": "pk001",
"lastName": "Smith",
"parents": [
{
"familyName": "Smith",
"firstName": "John"
},
{