

Création de l'API

Etape n°1 – Création BDD

Aller sur Phpmyadmin et Créer nouvelle BDD (table, propriété, etc)



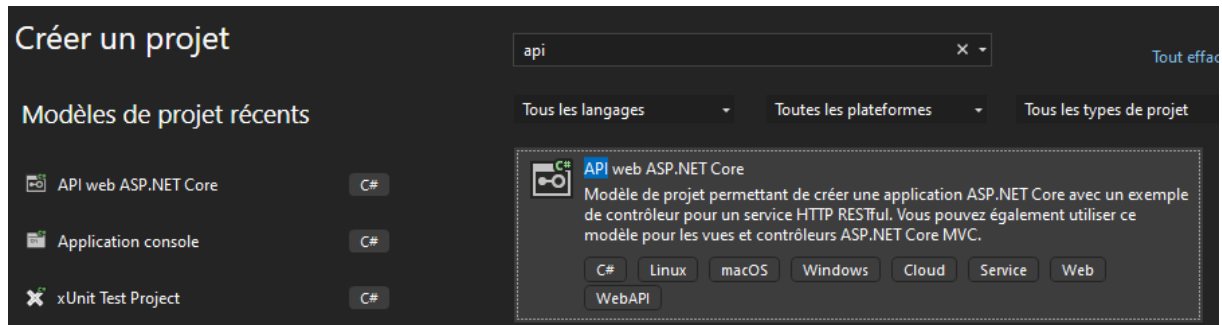
The screenshot shows the phpMyAdmin interface. On the left, there's a sidebar with the 'statit' database selected. The main area displays the structure of the 'joueur' table. The table has four columns: 'id_joueur' (int, primary key, AUTO_INCREMENT), 'pseudo' (varchar(50), utf8mb4_0900_ai_ci), 'rank' (varchar(50), utf8mb4_0900_ai_ci), and 'mmr' (int). Each column has a checkbox, a pencil icon for 'Modifier', a trash icon for 'Supprimer', and a plus icon for 'Plus'.

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
☐	1 id_joueur	int			Non	Aucun(e)		AUTO_INCREMENT	 Modifier  Supprimer  Plus
☐	2 pseudo	varchar(50)	utf8mb4_0900_ai_ci		Non	Aucun(e)			 Modifier  Supprimer  Plus
☐	3 rank	varchar(50)	utf8mb4_0900_ai_ci		Non	Aucun(e)			 Modifier  Supprimer  Plus
☐	4 mmr	int			Non	Aucun(e)			 Modifier  Supprimer  Plus

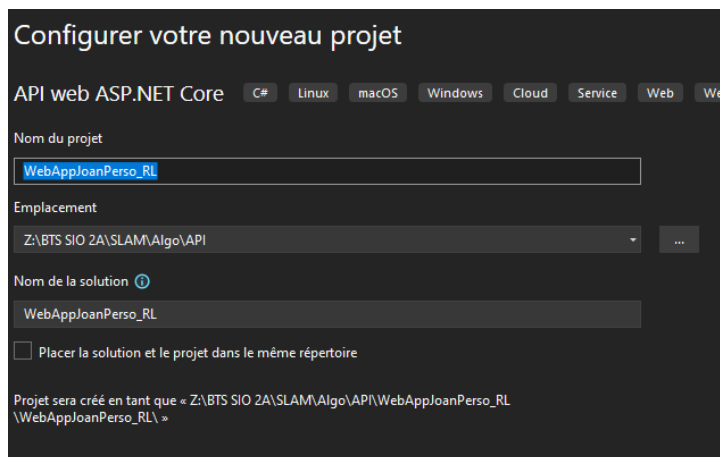
Etape n°2 – Création Projet sur VS

- Démarrer Visual studio 2022 et non pas le Visual studio 2022 PREVIEW

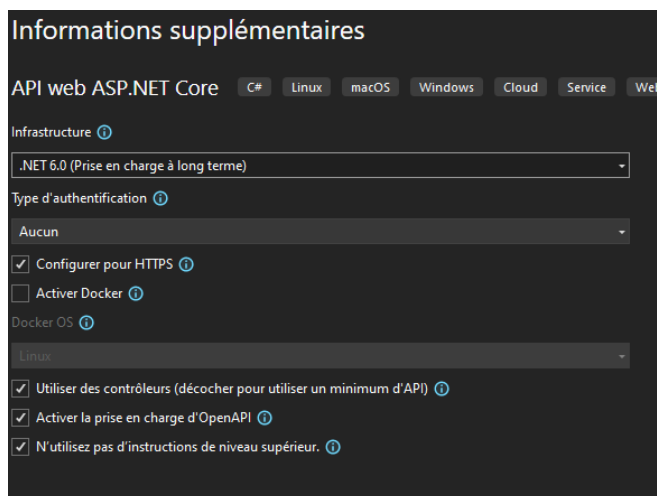
- Créé un projet :



- Configuration :



Info supp :

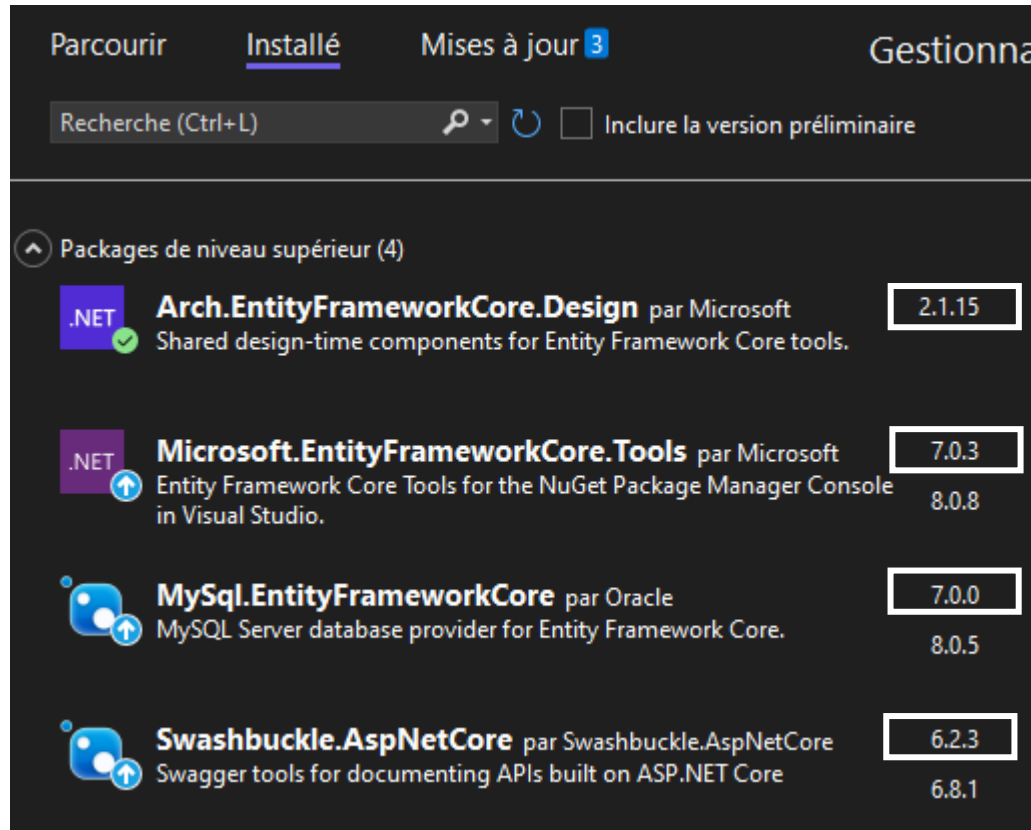


Puis, Créer.

Etape n°3 – Construction API

Ajout des Package Nuget :

- Aller dans « Projet », puis « Gérer les package Nuget »
- Installer ces packages (avec leur version respective) :



Exécution de commandes :

- Aller dans « Affichage », puis « Autre fenêtres » et « Console du Gestionnaire de package »
- Taper :

Scaffold-DbContext

"server=localhost;port=3306;user=root;password=;database=VotreBASE"

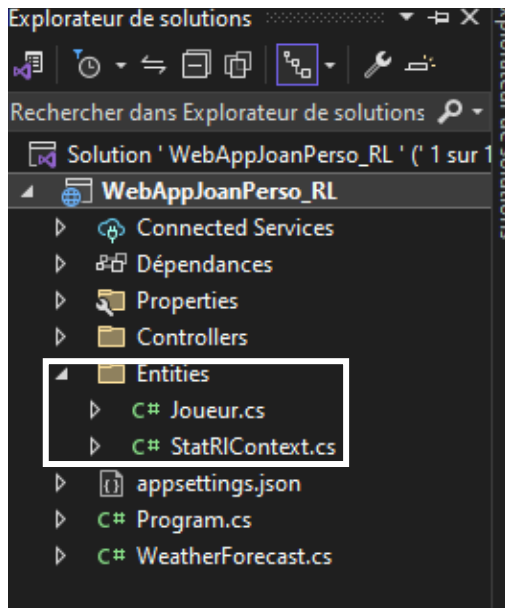
MySql.EntityFrameworkCore -OutputDir Entities -f

- Ceci doit s'afficher :

```
PM> Scaffold-DbContext "server=localhost;port=3306;user=root;password=;database=StatRL" MySql.EntityFrameworkCore -OutputDir
Entities -f
Build started...
Build succeeded.
To protect potentially sensitive information in your connection string, you should move it out of source code. You can avoid
scaffolding the connection string by using the Name= syntax to read it from configuration - see https://go.microsoft.com/fwlink/?
linkid=2131148. For more guidance on storing connection strings, see http://go.microsoft.com/fwlink/?LinkId=723263.
PM> |
```

TUTORIEL API Joan

- Une fois la commande exécutée, les templates sont créés automatiquement ainsi que le « VotreBaseContext ». Voir dans l'explorateur de solutions.



Ce n'est pas le plus approprié que la chaîne de connexion à la base de données soit spécifiée dans la méthode **OnConfiguring**. Pour faire mieux, nous utiliserons le fichier `appsettings.json`, dans lequel nous pouvons définir cette configuration.

Ce fichier est fait pour ça , et nous allons donc l'utiliser.

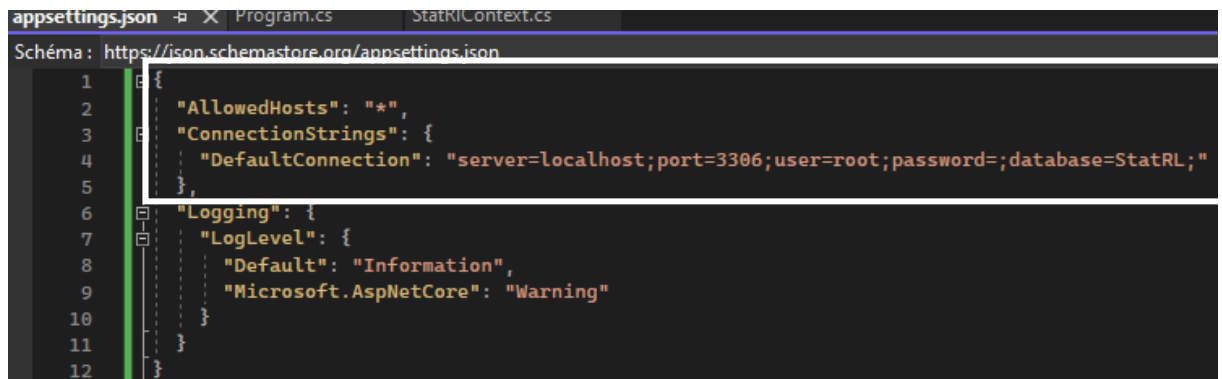
- Effacer le contenu de la méthode `OnConfiguring` :

```
0 références
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) { }
```

- Ajouter le **DB context** dans `program.c` (et le **using**):

```
1 using MySql.EntityFrameworkCore.Extensions;
2 using WebAppJoanPerso_RL.Entities;
3 using Microsoft.EntityFrameworkCore;
4
5
6 namespace WebAppJoanPerso_RL
7 {
8     0 références
9     public class Program
10     {
11         0 références
12         public static void Main(string[] args)
13         {
14             var builder = WebApplication.CreateBuilder(args);
15
16             // Add services to the container.
17             builder.Services.AddEntityFrameworkMySQL().AddDbContext<StatRLContext>(options => {
18                 options.UseMySQL(builder.Configuration.GetConnectionString("DefaultConnection"));
19             });
20 }
```

- Ajouter la connexionstring retiré précédemment dans appsettings.json :



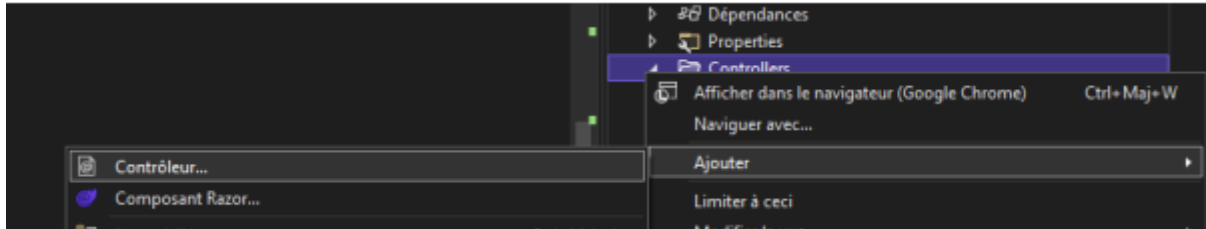
```
appsettings.json  Program.cs  StatRLContext.cs
Schéma : https://json.schemastore.org/appsettings.json
1  {
2    "AllowedHosts": "*",
3    "ConnectionStrings": {
4      "DefaultConnection": "server=localhost;port=3306;user=root;password=;database=StatRL;"
5    },
6    "Logging": {
7      "LogLevel": {
8        "Default": "Information",
9        "Microsoft.AspNetCore": "Warning"
10     }
11   }
12 }
```

Etape n°4 – Contrôleurs de l'API Web

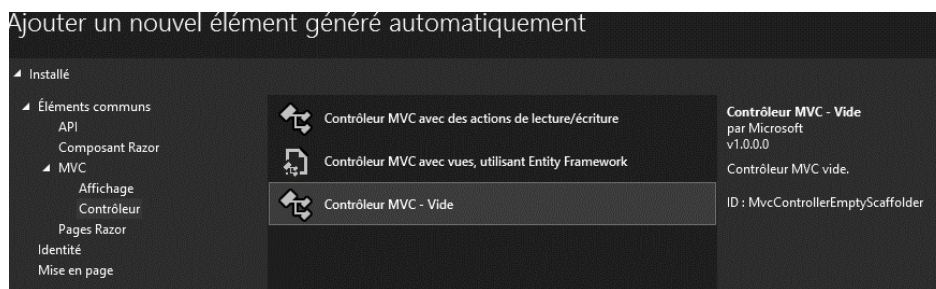
Maintenant, nous allons ajouter des contrôleurs.

Ils permettront d'établir des méthodes pour effectuer des opérations CRUD sur les tables de la BDD et de les exposer via l'API Web.

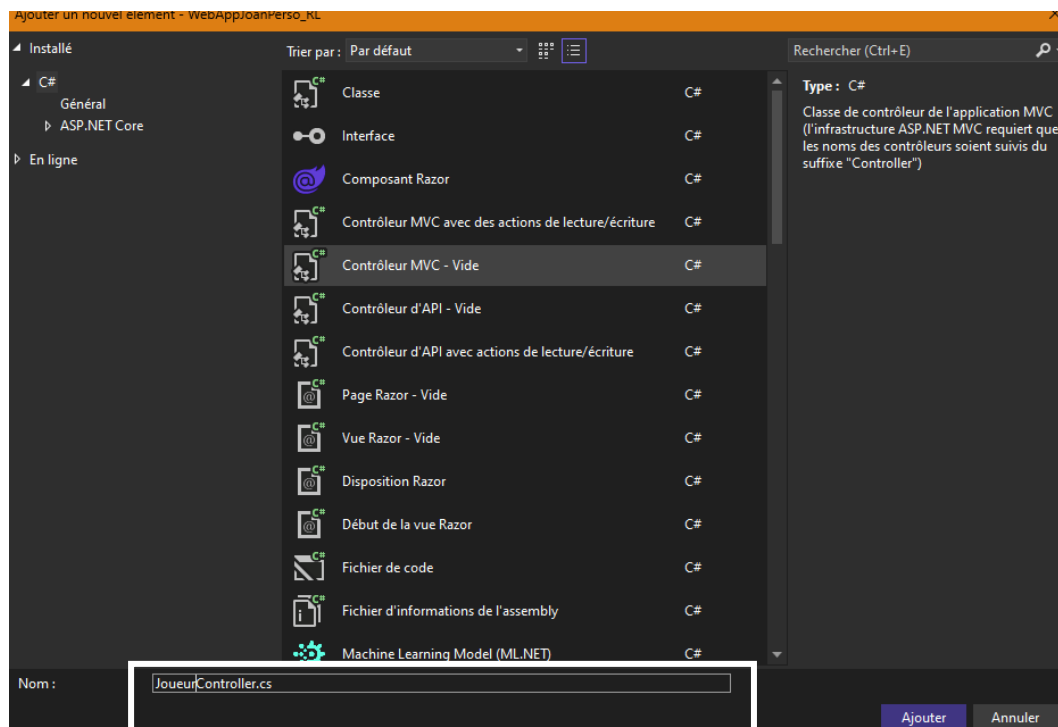
Depuis le dossier Controllers, nous ajouterons un contrôleur appelé UserController:



Puis, on choisit « Contrôleur vide ».



Et, on renomme selon votre entité.



Le code suivant permet d'avoir un CRUD sur une entité « User ». Ajouter ce code dans votre projet et adapter à votre entité (moi par exemple le nom de ma table « Joueur ») :

```
using WebAppJoanPerso_RL.Entities;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Net;
using System.Threading.Tasks;
using System.Collections.Generic;

namespace WebAppJoanPerso_RL.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class JoueurController : ControllerBase
    {
        private readonly StatRlContext _statRlContext; // Note l'underscore ici

        public JoueurController(StatRlContext statRlContext) // Paramètre modifié
        {
            _statRlContext = statRlContext; // Assignment correcte à la variable d'instance
        }

        /// <summary>
        /// Récupère tous les joueurs
        /// </summary>
        /// <response code="200">Liste des joueurs récupérée avec succès</response>
        /// <response code="404">Aucun joueur trouvé</response>
        /// <response code="500">Erreur interne du serveur</response>
        [HttpGet("GetJoueurs")]
        public async Task<ActionResult<List<Joueur>>> GetJoueurs()
```

```

{
    var joueurList = await _stateProviderContext.Joueurs.Select(
        s => new Joueur
        {
            IdJoueur = s.IdJoueur,
            Pseudo = s.Pseudo,
            Rank = s.Rank,
            Mmr = s.Mmr
        }
    ).ToListAsync();

    if (joueurList == null || joueurList.Count == 0)
    {
        return NotFound();
    }

    return joueurList;
}

/// <summary>
/// Récupère un joueur par son ID
/// </summary>
/// <param name="id">ID du joueur</param>
/// <response code="200">Joueur trouvé</response>
/// <response code="404">Joueur non trouvé</response>
[HttpGet("GetJoueurById/{id}")]
public async Task<ActionResult<Joueur>> GetJoueurById(int id)
{
    var joueur = await _stateProviderContext.Joueurs.Select(
        s => new Joueur
        {

```

```

        IdJoueur = s.IdJoueur,
        Pseudo = s.Pseudo,
        Rank = s.Rank,
        Mmr = s.Mmr
    }
).FirstOrDefaultAsync(s => s.IdJoueur == id);

if (joueur == null)
{
    return NotFound();
}

return joueur;
}

/// <summary>
/// Insère un nouveau joueur
/// </summary>
/// <response code="201">Joueur créé</response>
[HttpPost("InsertJoueur")]
public async Task<HttpStatusCode> InsertJoueur(Joueur joueur)
{
    var entity = new Joueur()
    {
        IdJoueur = joueur.IdJoueur,
        Pseudo = joueur.Pseudo,
        Rank = joueur.Rank,
        Mmr = joueur.Mmr
    };

    _statRlContext.Joueurs.Add(entity);

```

```
await _stateProviderContext.SaveChangesAsync();  
return HttpStatusCode.Created;  
}
```

```
/// <summary>
```

```
/// Met à jour un joueur existant
```

```
/// </summary>
```

```
/// <response code="200">Joueur mis à jour</response>
```

```
[HttpPut("UpdateJoueur")]
```

```
public async Task<HttpStatusCode> UpdateJoueur(Joueur joueur)
```

```
{
```

```
    var entity = await _stateProviderContext.Joueurs.FirstOrDefaultAsync(s => s.IdJoueur ==  
joueur.IdJoueur);
```

```
    if (entity == null)
```

```
    {
```

```
        return HttpStatusCode.NotFound;
```

```
    }
```

```
    entity.Pseudo = joueur.Pseudo;
```

```
    entity.Rank = joueur.Rank;
```

```
    entity.Mmr = joueur.Mmr;
```

```
    await _stateProviderContext.SaveChangesAsync();
```

```
    return HttpStatusCode.OK;
```

```
}
```

```
/// <summary>
```

```
/// Supprime un joueur par son ID
```

```
/// </summary>
```

```
/// <response code="200">Joueur supprimé</response>
```

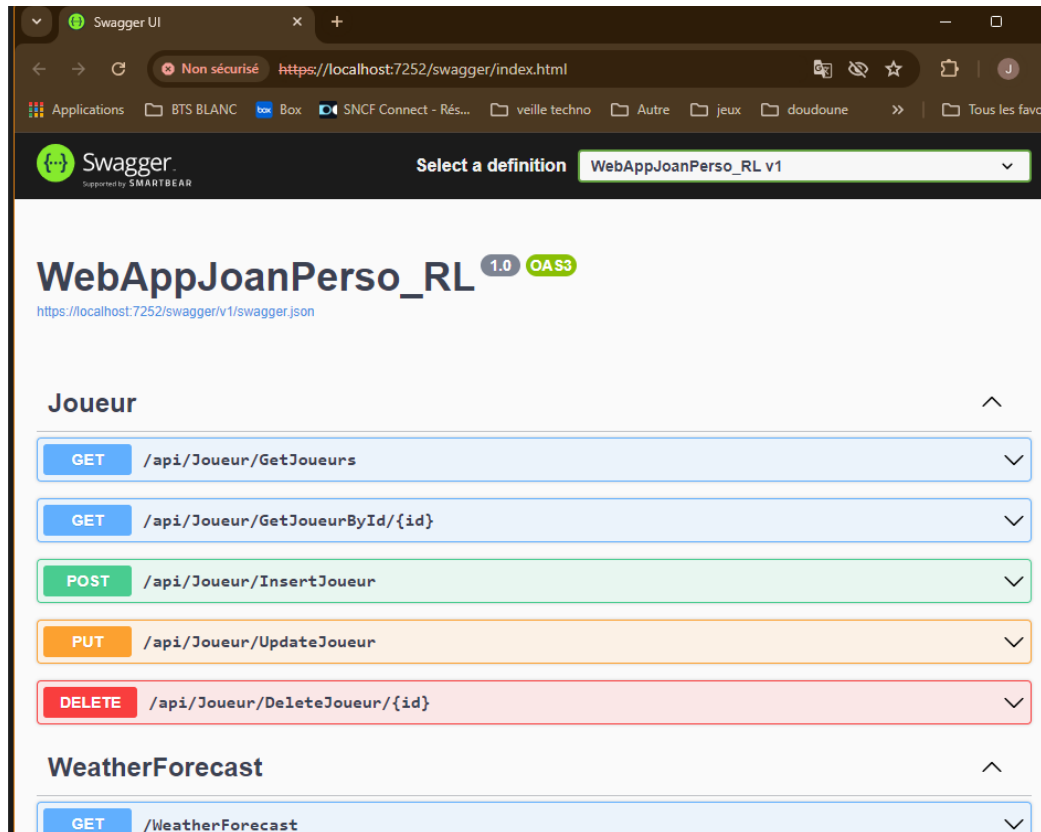
```
[HttpDelete("DeleteJoueur/{id}")]
public async Task<HttpStatusCode> DeleteJoueur(int id)
{
    var entity = await _statRlContext.Joueurs.FirstOrDefaultAsync(s => s.IdJoueur == id);

    if (entity == null)
    {
        return HttpStatusCode.NotFound;
    }

    _statRlContext.Joueurs.Remove(entity);
    await _statRlContext.SaveChangesAsync();
    return HttpStatusCode.OK;
}
}
```

Puis exécuter pour tester :

TUTORIEL API Joan



Etape n°5 – Ajout d'un swagger

Ajouter cette ligne dans program.cs (le swagger et le using):

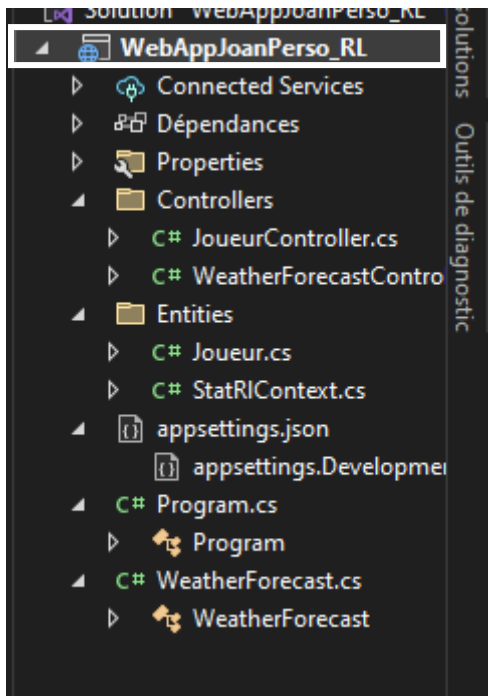
```
1 using WebApplicationSIOJOAN.Entities;
2 using Microsoft.EntityFrameworkCore;
3 using MySql.EntityFrameworkCore.Extensions;
4 using Microsoft.OpenApi.Models;
5
6
7 namespace WebApplicationSIOJOAN
8 {
9     0 références
10     public class Program
11     {
12         0 références
13         public static void Main(string[] args)
14         {
15             var builder = WebApplication.CreateBuilder(args);
16
17             // Add services to the container.
18             builder.Services.AddEntityFrameworkMySQL().AddDbContext<ProjetApiSio2aContext>(options => {
19                 options.UseMySQL(builder.Configuration.GetConnectionString("DefaultConnection"));
20             });
21
22             builder.Services.AddControllers();
23
24             // Swagger configuration with details
25             builder.Services.AddSwaggerGen(options =>
26             {
27                 options.SwaggerDoc("v1", new OpenApiInfo
28                 {
29                     Version = "v1",
30                     Title = "User API",
31                     Description = "An ASP.NET Core Web API for managing users",
32                     TermsOfService = new Uri("https://example.com/terms"),
33                     Contact = new OpenApiContact
34                     {
35                         Name = "Example Contact",
36                         Url = new Uri("https://example.com/contact")
37                     },
38                     License = new OpenApiLicense
39                     {
40                         Name = "Example License",
41                         Url = new Uri("https://example.com/license")
42                     }
43                 });
44             });
45
46             var app = builder.Build();
47
48             // Configure the HTTP request pipeline.
49             if (app.Environment.IsDevelopment())
50             {
51                 app.UseSwagger();
52                 app.UseSwaggerUI();
53             }
54
55             app.UseHttpsRedirection();
56
57             app.UseAuthorization();
58
59             app.MapControllers();
60
61             app.Run();
62 }
```

```
// Swagger configuration with details

builder.Services.AddSwaggerGen(options =>
{
    options.SwaggerDoc("v1", new OpenApiInfo
    {
        Version = "v1",
        Title = "Joueur API",
        Description = "An ASP.NET Core Web API for managing users",
        TermsOfService = new Uri("https://example.com/terms"),
        Contact = new OpenApiContact
        {
            Name = "Example Contact",
            Url = new Uri("https://example.com/contact")
        },
        License = new OpenApiLicense
        {
            Name = "Example License",
            Url = new Uri("https://example.com/license")
        }
    });
});
```

Changer le User, etc...

Enfin, Ouvrir le fichier csproj :



Ajouter cette ligne dans le fichier Csproj :

```
<GenerateDocumentationFile>true</GenerateDocumentationFile>
```

```
<NoWarn>$(NoWarn);1591</NoWarn>
```

Ici :

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <GenerateDocumentationFile>true</GenerateDocumentationFile>
    <NoWarn>$(NoWarn);1591</NoWarn>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Arch.EntityFrameworkCore.Design" Version="2.1.15" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="7.0.3">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
    </PackageReference>
    <PackageReference Include="MySQL.EntityFrameworkCore" Version="7.0.0" />
    <PackageReference Include="Swashbuckle.AspNetCore" Version="6.2.3" />
  </ItemGroup>

</Project>
```