Fiche de procédure C# (création API)

1. Pré-requis :

- Rider (JetBrains)

- .NET SDK (version 8 ou plus récente)

- Bibliothèque Entity Framework Core

- Swagger (pour tester l'API)

2. Création d’un projet Web API :

I - Créer un projet Web API avec des contrôleurs :

dotnet new webapi --use-controllers -o TodoApi

Ouvre Rider et dans le terminal, exécute les commandes suivantes :

pi --use-controlledotnet new webars -o TodoApi
cd TodoApi

Cela génère un projet Web API dans le dossier `TodoApi`.

II - Ajouter EntityFramework Core InMemory :

dotnet add package Microsoft.EntityFrameworkCore.InMemory

3. Ajouter le modèle et le contexte :

I - Créer un modèle :

- Ajouter un dossier `Models` dans ton projet.

- Créer une classe `TodoItem.cs` dans ce dossier avec le contenu suivant :

```
namespace TodoApi.Models
{
    public class TodoItem
    {
        public long Id { get; set; }
        public string? Name { get; set; }
        public bool IsComplete { get; set; }
        public string? Secret { get; set; }
    }
}
```

II - Créer le contexte de la base de données :

- Ajoute une classe `TodoContext.cs` dans le dossier `Models` :

```
using Microsoft.EntityFrameworkCore;
using TodoApi.Models;

namespace TodoApi.Models;

public class TodoContext : DbContext
{
    public TodoContext(DbContextOptions<TodoContext> options)
        : base(options)
    {
    }

    public DbSet<TodoItem> TodoItems { get; set; } = null!;
}
```

4. Configurer le projet :

I - Modifier `Program.cs` pour enregistrer le contexte de la base de données :

```
using Microsoft.EntityFrameworkCore;
using TodoApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();
```

```
builder.Services.AddDbContext<TodoContext>(opt =>
   opt.UseInMemoryDatabase("TodoList"));
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

if (app.Environment.IsDevelopment())
{
   app.UseSwagger();
   app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

5. Générer un contrôleur pour l'API :

I - Ajouter les dépendances pour le scaffolding :

Exécuter ces commandes dans le terminal de Rider :

```
dotnet add package Microsoft.VisualStudio.Web.CodeGeneration.Design
dotnet add package Microsoft.EntityFrameworkCore.Design
dotnet add package Microsoft.EntityFrameworkCore.SqlServer
dotnet add package Microsoft.EntityFrameworkCore.Tools
dotnet tool install -g dotnet-aspnet-codegenerator
```

II - Créer un contrôleur TodoItems :

Exécute la commande suivante pour générer un contrôleur CRUD pour les éléments `TodoItem` :

```
dotnet aspnet-codegenerator controller -name TodoItemsController -async -api -m TodoItem -dc TodoContext -outDir Controllers
```

6. Tester l'API avec Swagger :

I - Lancer l'application :

- Exécute cette commande pour démarrer l'application :

    dotnet run --launch-profile https

    ou just contrôle f5

II - Accéder à Swagger :

- Ouvre ton navigateur et accède à l'URL suivante : `https://localhost:<port>/swagger`
- Utilise Swagger pour tester les différentes méthodes de ton API :
    - GET `/api/todoitems` : Pour récupérer tous les éléments.
    - POST `/api/todoitems` : Pour ajouter un nouvel élément.
    - GET `/api/todoitems/{id}` : Pour récupérer un élément spécifique.

type.parse(variable) pour changer le type d’un variable