Les archictectures microservices gagenent en popularité dans le developpement d’applications modernes.

1 – Creation du projet et configuration de IdentityServer

1-1 Creation du projet

Nous allons commencer à partir de zéro en creant une nouvelle application ASP.NET Core qui sera notre IdentityServer. Elle doit être basée sur le modele « Vide » et n’avoir aucune authentification :



Une fois le nouveau projet créé, vous devez ajouter une référence au package IdentityServer4, en utilisant le gestionnaire de packages NuGet :



Si vous utilisez Visual Studio Code, vous pouvez utiliser la commande :

Dotnet add package IdentityServer4

1-2 Configuration d’IdentityServer

Vous aurez besoin d’enregisterer IdentityServer dans le conteneur d’injection de dépendances de ASP.NET Core et ajouter le middleware de ce dernier dans le pipeline HHTP du framework.

Pour enregistrer IdentityServer, vous devez éditer le fichier Startup.cs et modifier la méthode ConfiguresServices comme suit :

public void ConfigureServices(IServiceCollection services)

{

services.AddIdentityServer()

.AddDeveloperSigningCredential();

}

AddIdentityServer est une méthode d’extension qui permet d’enregister IdentityServer dans le conteneur d’IoC.

La dépendance minimale que nous avons besoin pour l’instant est AddDeveloperSigningCredential(). Cette extension permet de créer une clée temporaire et le necessaire piour signer les jetons (Tokens). C’est pratique pour demarrer en enviroenment de develloppement. Mais, vous ne devez pas le laisser trainer la en production et vous devez fournir le nécessaire pour gerer cela.

Pour ajouter Ie middleware IdentityServer dans le pipeline HTTP de ASP.NET Core, vous devez modifier la méthode Configure() du fichier Startup.cs et ajouter la ligne de code :

app.UseIdentityServer();

Le code complet de cette méthode est le suivant :

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseIdentityServer();

app.Run(async (context) =>

{

await context.Response.WriteAsync("Hello World!");

});

}