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!");

});

}

C’est tout. Nous venons de faire le minimun pour intergrer IdentityServer à notre projet. Nous pouvons desormais l’utiliser comme service de gestion sécuriser des accès à nos applications.

Mais avant, nous allons faire quelques modifications pour nous assurer que les clients pointeront toujours la bonne application.

1-3 Configuration du client

Tout client qui fait appel à notre serveur de gestion d’identité doit être un client de confiance. C’est pourquoi, ce dernier doit être être reférencé dans l’application IdentityServer.

Par ailleurs, toute ressources (API par exemple), donc l’accès est securrisé, doit être repertorie dans l’application IdentityServer.

Nous allons

using IdentityServer4.Models;

using System.Collections.Generic;

namespace AspNetCoreIdentityServer

{

public class Config

{

public static IEnumerable<Client> GetClients()

{

return new List<Client>

{

};

}

public static IEnumerable<ApiResource> GetApiResources()

{

return new List<ApiResource>

{

};

}

}

}

services.AddIdentityServer()

.AddDeveloperSigningCredential()

.AddInMemoryApiResources(Config.GetApiResources())

.AddInMemoryClients(Config.GetClients());

1-4 Modification de l’hote

Nous devons nous assurer que notre application IdentityServer sera toujours accessible via la même adresse lorsqu’elle est en execution. Par ailleurs, il environement de developpement et lorsqu’on est en mode apprentissage, il est interressant de voir en temps réel les logs de notre application dans la console.

Pour cela, nous allons accéder à l’onglet « Déboguer » dans les propriétés de notre projet. Nous allons dérouler la zone « Profil » et selectionner le nom de application (AspNetCoreIdentityServer).

Les URL suivantes doivent être définies dans le champ URL de l’application, si ce n’est pas le cas :

<https://localhost:5001;http://localhost:5000>



Générez et exécutez votre application.

NB : Les projets ASP.NET Core 2.1 sont configurés pour utiliser par defaut SSL. Pour eviter les avertissement SSL dans le navigateur, vous devez accepter le certificat auto-signé généré par ASP.NET Core. Une notification s’affichera à cet effet à la première exécution de votre application.