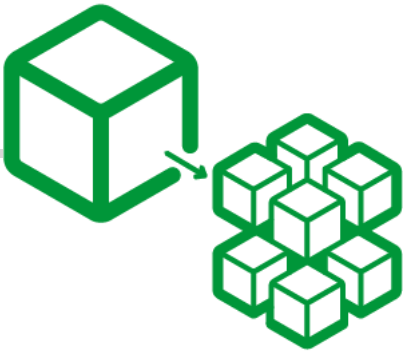
IT Arkitektur & Infrastruktur

Opgavesæt M7.03 - gRPC Service



Formål

I denne opgave skal I lave en simpel service som kan aflevere en besked med brug af gRPC. I skal bruge **Postman** igen til at teste endepunktet i jeres løsning.

> **Info**
>
> Denne opgave løses individuelt og skal ikke indgå i jeres organisations repos.

Opgave A - Opret et C# gRPC-service projekt

1. Open en terminal og find en ny folder til dit projekt.
2. Opret et nyt C# gRPC projekt i folderen (bemærk at i anvender **grpc** skabelonen):

```
$ dotnet new grpc -o MessageService
$ cd MessageService
```

3. Vi skal bruge en speciel nuget således at Postman kan udlede applikationens interface:

```
$ dotnet add package Grpc.AspNetCore.Server.Reflection
```

Opgave B - Inspektion af projektet fra template

Åben projektet med VS Code så du kan kigge dig omkring i filerne.

1. Start med at finde *greet.proto* under **Protos** folderen. Det er en **Protocol Buffer** for servicen. Den beskriver 2 beskedtyper: *HelloReply* & *HelloRequest*.

Der er også en service kaldet *Greeter*. *Greeter* indeholder en RPC-metode: **SayHello**, som tager imod et **HelloRequest** og afleverer et **HelloReply**.

> **Hvad betyder tallene i HelloRequest & HelloReply?**
>
> Hint: Assigning Field Numbers

2. Find filen *GreeterService.cs* i Services-folderen. Den indeholder den kode som skal køres når **SayHello** kaldes.
3. Åben filen *Program.cs*. På linjen med **app.MapGrpcService();** registreres ovennævnte service, internt i applikationen, til senere brug.
4. Bemærk at klasserne fra **greet.proto** bliver automatisk oprettet (men vi kan ikke se kildekoden!). I finder f.eks. *HelloReply* i **GreetService.cs**. Disse klasser blive oprettet når I bygger første gang og kan derefter findes i folderen **obj\Debug\net7.0\Protos**.

Opgave C - Tilføj reflection

For at **Postman** automatisk kan anvende **Protocol Bufferen** skal vi tilføje lidt ekstra funktionalitet til jeres app.

1. Installér gRPC reflektionsbibliotek til ASP-NET:

```
$ dotnet add package Grpc.AspNetCore.Server.Reflection
```

2. Åben *Program.cs* og tilføj følgende kode:

```
builder.Services.AddGrpc(); // <-- Tilføj koden efter denne linje!
builder.Services.AddGrpcReflection();
```

3. Tilføj også følgende kode:

```
IWebHostEnvironment env = app.Environment;
if (env.IsDevelopment())
{
 app.MapGrpcReflectionService();
}

app.Run(); // <-- Tilføj koden før denne linje!
```

Opgave D - Afprøv programmet

1. Kør nu dit C#-program med kommandoen **dotnet run**. Du finder adressen og porten som den lytter på i konsol-outputtet. Bemærk at du skal erstate **http://** med **grpc://** i punkt 3 herunder!
2. Åben nu **Postman** og opret et nyt gRPC **request**. Det gør du ved at trykke på "New"-knappen øverst til venstre. Vælg herefter **gRPC Request** fra dialog-vinduet.
3. Indtast adressen på din gRPC-service i *"Enter Server URL"* feltet og tast **tab** for at hoppe til det næste felt. Her vælger du **Use Server Reflection**, hvis det ikke allerede er valgt.
4. Vælg **Greeter / SayHello** fra dropdown menuen efter URL'en og tryk på **Invoke**-knappen.
5. Du har nu fået et svar i **Response** vinduet nederst i Postman!

> **Hvad med *name*?**
>
> I proto-filen beskrives at **HelloRequest**-metoden tager imod en **name**-parameter. Den skal selvfølgelig også sendes med fra Postman!
>
> Gentag opgave **D.4**, men i Postman **Message**-fanebladet, tilføj:
>
> ```
> {"name": "Dit navn"}
> ```
>
> inden du trykker **Invoke**.

++ Ekstra

Udvid protocol bufferen (*greet.proto*) således at **HelloRequest** også indeholder en property for **last_name**. I *GreeterService.cs* skal du sørge for at denne property medtages i svaret!

> **Postman Refresh**
>
> Når du laver ændringer til dit interface, skal Postman genindlæse interface-beskrivelsen. Dette gør du ved at trykke på refresh symbolet i dropdown-menuen!