gRPC

Remote Procedure Calls

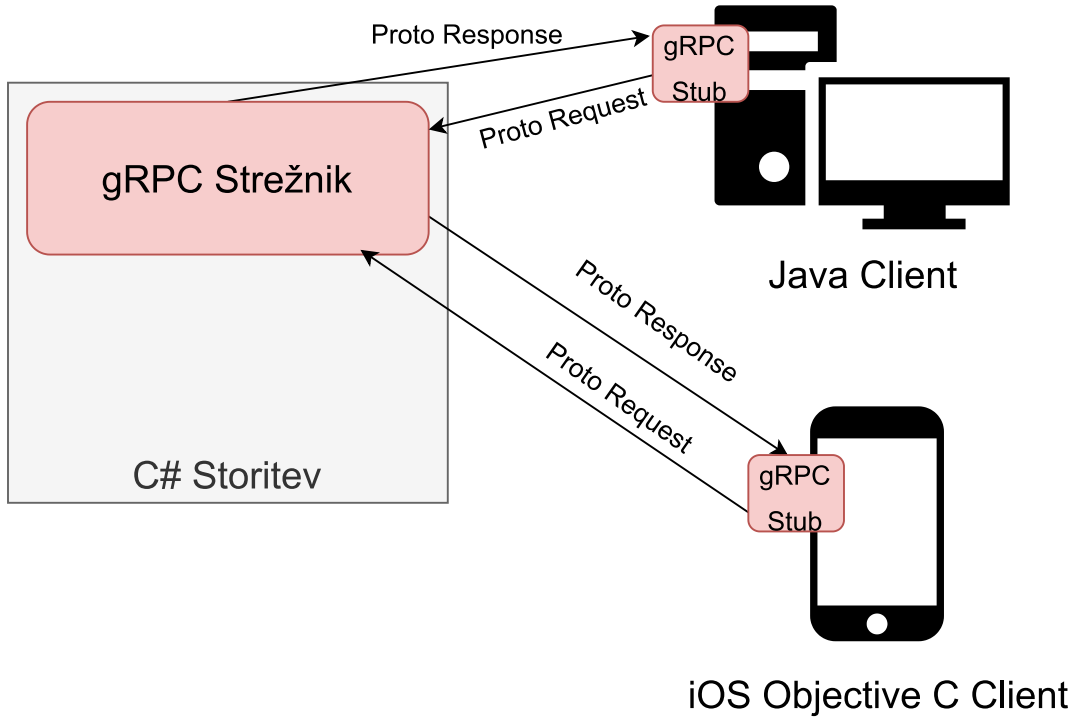
gRPC

- odprtokodno ogrodje RPC, ki se uporablja za gradnjo skalabilnih in hitrih API-jev, alternativa RESTu in SOAPu
- Hitrejši od RESTa, manjši “overhead” pri sporočanju v primerjavi z REST
- gRPC vs REST:

	gRPC	**REST API**
HTTP protokol	*HTTP 2*	*HTTP 1.1*
Sporočilni format	*Binarno kodirana sporočila (Protobuf)*	*JSON ali XML in drugi, po navaid v tekstu*
Code Generation	*npr. Protoc Compiler*	*npr. Swagger*
Communication	*Odjemalec zahteva->Odgovor ali obojesmerna*	*Odjemalec Zahteva->Odgovor*

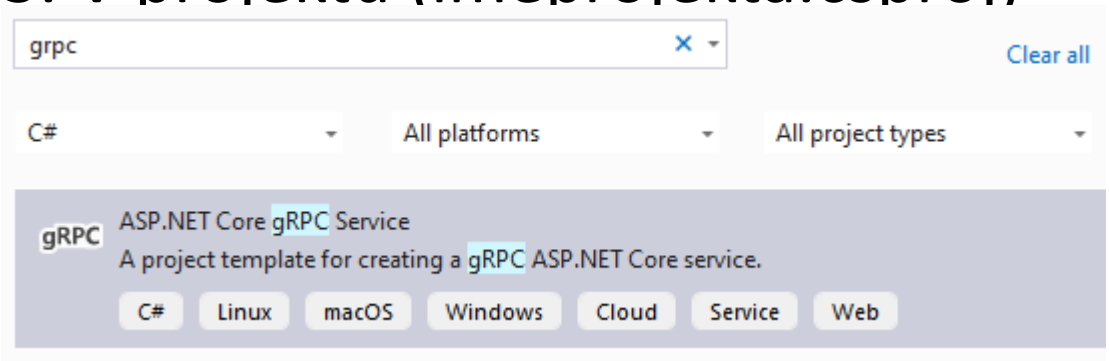
gRPC

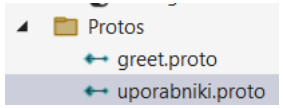
- Komunikacija ni odvisna od uporabljenega programskega jezika.
- Storitve po pristopu “Contract-First”
- *Pogodbo* definiramo v datotekah .proto, kjer se nahajajo definicije **storitev** in **sporočil**



Ustvarjanje storitve 1/

1. Ustvarimo ASP.NET Core gRPC service in namestimo protobuf-net
2. V direktoriju Protos ustvarimo storitev.proto
3. V projektu (imeprojekta.csproj)






```
<ItemGroup>
  <Protobuf Include="Protos\greet.proto" GrpcServices="Server" />
  <Protobuf Include="Protos\uporabniki.proto" GrpcServices="Server" />
</ItemGroup>
```

Protos/osebe.proto

Ustvarjanje storitve (storitev.proto) <- kot interface 2/

```
syntax = "proto3";
option csharp_namespace = "Odjemalec";
package osebe;

service Osebe
rpc GetOsebe (GetOsebeRequest) returns (GetOsebeReply);

rpc DodajOsebo (DodajOseboRequest) returns (DodajOseboReply);
}

message GetOsebeRequest {
}
message GetOsebeReply {
  repeated OsebaMessage osebe = 1;
}
```

Vhodni parametri

Return type

Metoda

Sporočilo! To bo pretvorjeno v ProtoBuf!

```
message DodajOseboRequest {
OsebaMessage oseba = 1;
}

message DodajOseboReply{
string sporocilo = 1;
}

message OsebaMessage{
string name = 1;
string priimek = 2;
int32 letoRojstva = 3;
}
```

https://learn.microsoft.com/en-us/aspnet/core/grpc/protobuf?view=aspnetcore-7.0#scalar-value-types ← pretvorbe tipov, npr. Protobuf -> int32, C# -> int

Ustvarjanje storitve implementacija

- C# / C# EntityFramework, … bomo dodali metodo za serializacijo / deserializacijo v ProtoBuf

osebe.proto

```
message OsebaMessage{
string name = 1;
string priimek = 2;
int32 letoRojstva = 3;
}
```

Oseba.cs

```
public class Oseba
{
    0 references
    public  Oseba(OsebaMessage OsebaMessage){
        this.OsebaMessage = OsebaMessage;
    }
    3 references
    public Oseba(string ime, string priimek, int letoRojstva)
    {
        Ime = ime;
        Priimek = priimek;
        LetoRojstva = letoRojstva;
    }

    0 references
    public Oseba() { }

    3 references
    public string Ime { get; set; }
    3 references
    public string Priimek  { get; set; }
    3 references
    public int LetoRojstva { get; set; }

    public OsebaMessage OsebaMessage {

        get {
            return new OsebaMessage {
                Name = this.Ime,
                Priimek = this.Priimek,
                LetoRojstva = this.LetoRojstva };
        }
        set {
            this.Ime = value.Name;
            this.Priimek = value.Priimek;
            this.LetoRojstva = value.LetoRojstva;
        }
    }
}
```

Services/OsebaService.cs

Ustvarjanje storitve implementacija

Implementira storitev(.proto):StoritevBase

```
public class OsebeService : Osebe.OsebeBase
    {
        public override Task<GetOsebeReply> GetOsebe(GetOsebeRequest request, ServerCallContext context)
        {
            // Tukaj bi dostopali do realnih podatkov ...
            List<Oseba> seznamOseb = new List<Oseba>();
            seznamOseb.Add(new Oseba("Janez", "Novak", 1990));
            seznamOseb.Add(new Oseba("Marko", "Novak", 1980));
            seznamOseb.Add(new Oseba("Sandi", "Novak", 1960));

            var reply = new GetOsebeReply();
            seznamOseb.ForEach(x => reply.Osebe.Add(x.OsebaMessage));



            return Task.FromResult(reply);
        }
    }
```

Dodamo v main() (storitev

- `app.MapGrpcService<OsebeService>();`

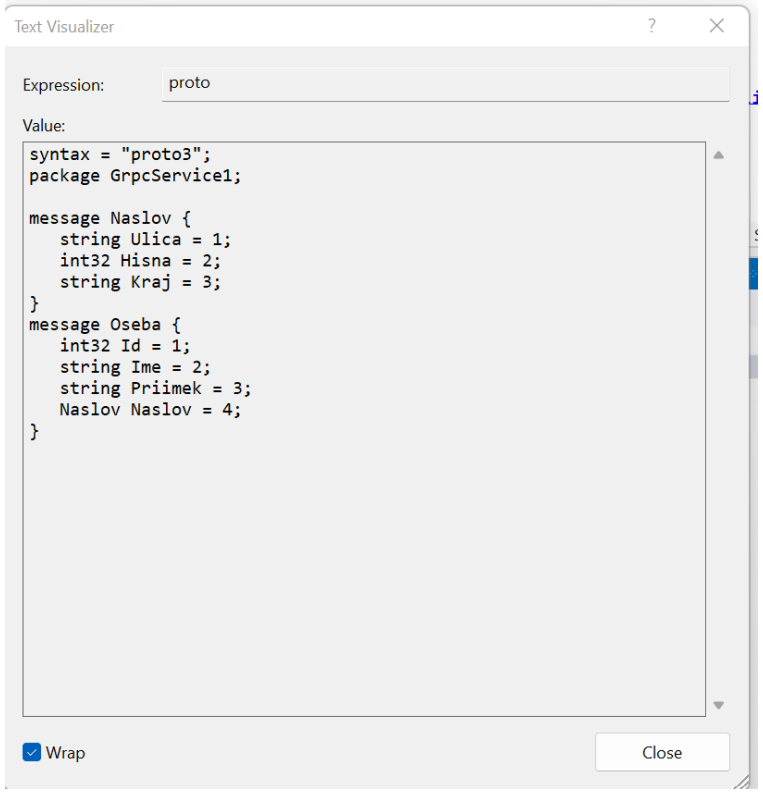
alternativni način generiranja proto datotek

Message (“razredi”) z uporabo anotacij

```
// …… Razred.cs
using ProtoBuf;

namespace GrpcService1
{
    [ProtoContract]
    class Oseba
    {
        [ProtoMember(1)]
        public int Id { get; set; }
        [ProtoMember(2)]
        public string Ime { get; set; }

        [ProtoMember(3)]
        public string Priimek { get; set; }

        [ProtoMember(4)]
        public Naslov Naslov { get; set; }
    }
    [ProtoContract]
    class Naslov
    {
        [ProtoMember(1)]
        public string Ulica { get; set; }
        [ProtoMember(2)]
        public int hisna { get; set; }

        [ProtoMember(3)]
        public string kraj { get; set; }
    }
}
```

```
string proto = Serializer.GetProto<Oseba>();
```

Inspect element prikaže sledeče… ☺

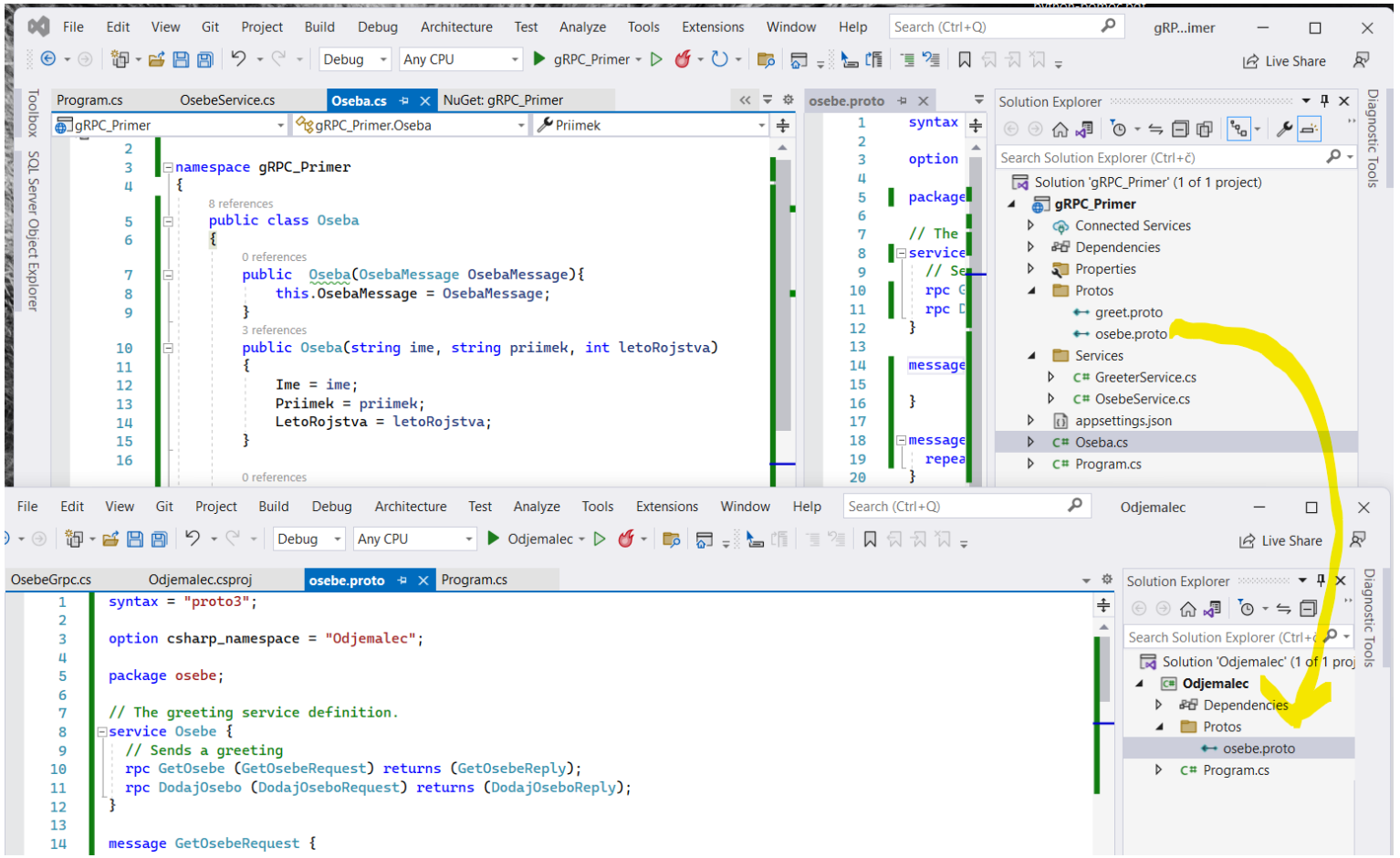


Odjemalec gRPC 1/3

- Namestimo odvisnosti:
  - Grpc.Net.Client
  - Google.Protobuf

Odjemalec gRPC 2/3

1. proto datoteke dodamo v project odjemalca



Odjemalec gRPC 3/3

```
using System.Net;
using System.Threading.Tasks;
using Grpc.Net.Client;
using Odjemalec;
using ProtoBuf;
```

```
// The port number must match the port of the gRPC server.
using var channel = GrpcChannel.ForAddress("https://localhost:7176");
var client = new Osebe.OsebeClient(channel);
var reply = client.GetOsebe(new GetOsebeRequest());
Console.WriteLine("Odgovor: " + reply);
Console.WriteLine("Press any key to exit...");
Console.ReadKey();
```

Dodatni viri

- ProtoBuf:
  - https://developers.google.com/protocol-buffers
- Protokol gRPC:
  - https://grpc.io/
- Strežnik:
  - https://learn.microsoft.com/en-us/aspnet/core/grpc/basics?view=aspnetcore-7.0
  - https://learn.microsoft.com/en-us/aspnet/core/grpc/services?view=aspnetcore-7.0
  - https://learn.microsoft.com/en-us/aspnet/core/grpc/protobuf?view=aspnetcore-7.0
- Odjemalec:
  - https://learn.microsoft.com/en-us/aspnet/core/grpc/client?view=aspnetcore-7.0
- Splošni vodiči:
  - https://www.yogihosting.com/grpc-aspnet-core/
  - https://www.infoq.com/articles/getting-started-grpc-dotnet/
  - https://sahansera.dev/building-grpc-server-dotnet/