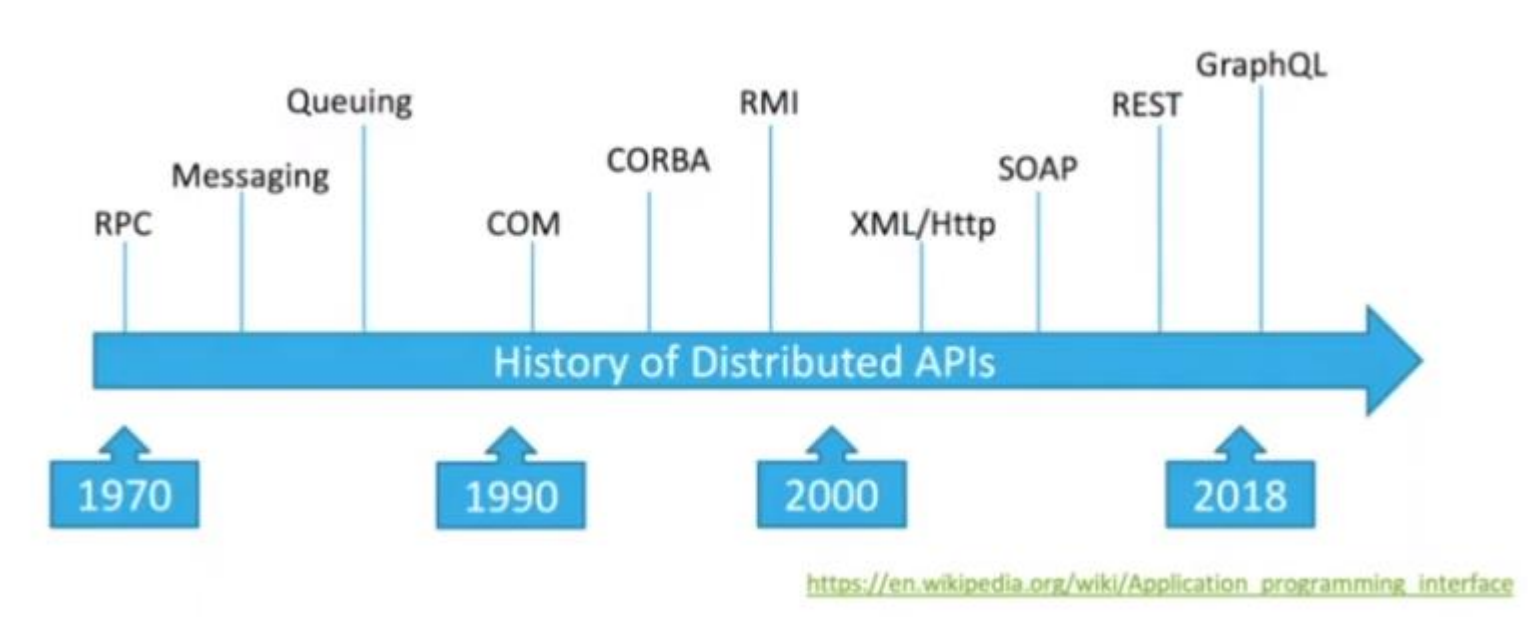
gRPC i .NET

gRPC

- Brz, cross platform, multi-language okvir za high-performance komunikaciju
- Dizajniran da adresira probleme u arhitekturi mikroservisa i distribuiranih sistema
- Podržava bidirekcioni striming
- Lako se integriše sa ASP.NET Core
- Naslednik WCF-a?

Istorija


RPC
Messaging
Queuing
COM
CORBA
RMI
XML/Http
SOAP
REST
GraphQL
History of Distributed APIs
1970
1990
2000
2018
https://en.wikipedia.org/wiki/Application_programming_interface

gRPC - prednosti

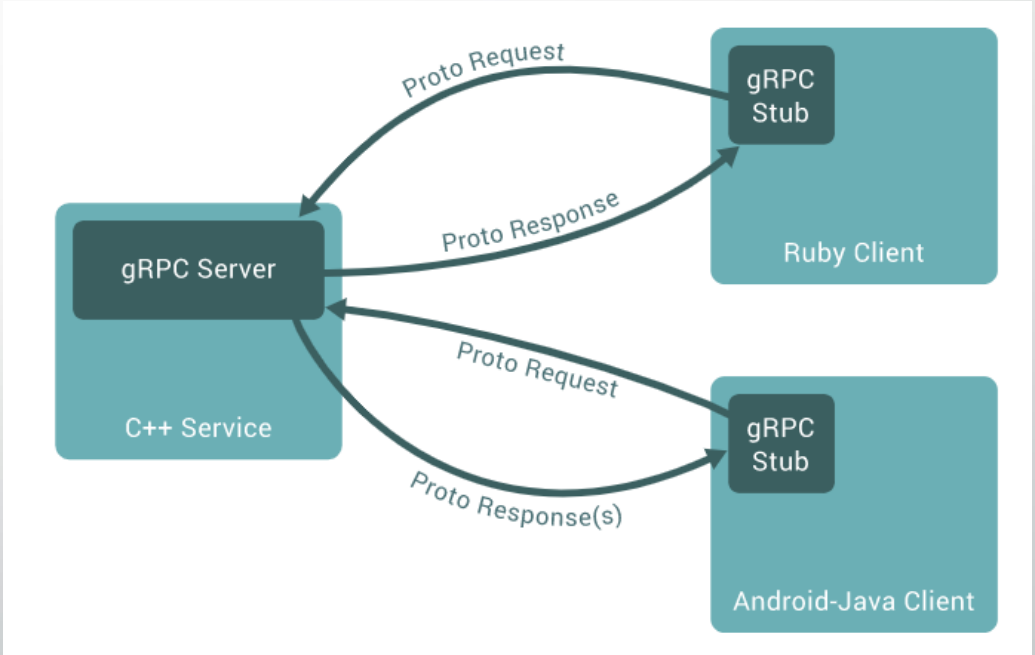
- Perfomanse
- Interoperabilnost
- Podrška za striming
- Timeouts i cancellation
- Sigurnost

Remote Procedure Call

- 2001. Google Stubby
- 2015 open-sourceovan kao gRPC
- 2019 gRPC kao deo .NET okvira

gRPC

- gRPC omogućava klijentskoj aplikaciji da pozove metode na serveru koje se nalaze na drugoj mašini kao da se radi o lokalnom objektu.
- Bazira se na ideji definicije servisa, specifikaciji metoda koje mogu biti pozvane remotely sa svim parametrima i povratnim tipovima.
  - Server implementira interfejs i pokreće gRPC server koji će biti zadužen za pozive klijenta
  - Klijent ima stub koji daje iste metode kao i server.



Protobuf

- Po definiciji, gRPC koristi Protocol Buffers, Google-ov open-source mehanizam za serijalizaciju struktuiranih podataka
  - Interface definition language
  - Language & platform neutral
  - Extensible
  - Serializable
  - Nije vezan za gRPC

Protobuf

- Prvi korak u radu sa protobaferima jeste definisanje strukture podataka koji se serijalizuju u proto fajlu
  - Proto fajl je običan tekstualni fajl sa .proto ekstenzijom
- Protobafer podaci su organizovani u poruke (messages) gde je svaka poruka mala logička celina informacija koja sadrži niz *ime-vrednost* parova (fields)

```
message Person
{
        string name = 1;
        int32 id = 2;
        bool has_ponycopter = 3;
}
```

Protobuf

- Nakon specifikacije strukture podataka, koristi se protobuf kompajler **protoc** da bi se generisale klase za pristup podacima u željenom jeziku
  - Ove klase sadrže jednostavne akcesore za svako polje poput name() i setName(), kao i metode za serijalizaciju i parsiranje cele strukture u i iz niza bajtova

```
message Person
{
        string name = 1;
        int32 id = 2;
        bool has_ponycopter = 3;
}
```

Protobuf poruke

WCF

```
namespace TraderSys
{
    [DataContract]
    public class Stock
    {
        [DataMember]
        public int Id { get; set;}
        [DataMember]
        public string Symbol { get; set;}
        [DataMember]
        public string DisplayName { get; set;}
        [DataMember]
        public int MarketId { get; set; }
    }
}
```

ProtoBuf

```
syntax = "proto3";

option csharp_namespace = "TraderSys";

message Stock {

    int32 id = 1;
    string symbol = 2;
    string display_name = 3;
    int32 market_id = 4;

}
```

Protobuf skalarni tipovi podataka

Osnovni tipovi podataka	
Protobuf tip	**C# tip**
double	double
float	float
int32	int
int64	long
uint32	uint
uint64	ulong
sint32	int
sint64	long
fixed32	uint
fixed64	ulong
sfixed32	int
sfixed64	long
bool	bool
string	string
bytes	ByteString

Drugi .NET tipovi	
C# type	**Protobuf well-known type**
DateTimeOffset	google.protobuf.Timestamp
DateTime	google.protobuf.Timestamp
TimeSpan	google.protobuf.Duration

C# type	Well Known Type wrapper
double?	google.protobuf.DoubleValue
float?	google.protobuf.FloatValue
int?	google.protobuf.Int32Value
long?	google.protobuf.Int64Value
uint?	google.protobuf.UInt32Value
ulong?	google.protobuf.UInt64Value

Protobuf tipovi podataka

- Ugnježđeni tipovi
- Repeated
- Reserved
- Any i Oneof
- Enumeracije
- Protobuf map

```
message Outer {
    message Inner {
        string text = 1;
    }
    Inner inner = 1;
}
```

```
var inner = new Outer.Types.Inner { Text = "Hello" };
```

gRPC servisi

- gRPC servisi se definišu u običnim proto fajlovima sa RPC parametrizovanim fajlovima gde su povratni tipovi specificirani kao protobuffer poruke
- gRPC koristi *protoc* sa specijalnim gRPC plugin-om za generisanje koda iz proto fajla
  - Generiše se gRPC klijent i server kod, kao i regularni protobuf kod za populisanje, serijalizaciju i dobavljanje različitih tipova poruka

- Literatura: https://developers.google.com/protocol-buffers/docs/overview

Definicija servisa

- Kao i većina drugih RPC sistema, gRPC je baziran na ideji definicije servisa koja uključuje specifikaciju metoda koje mogu biti pozvani remotely sa određenim parametrima.
- Iako koristi protocol buffers kao IDL, moguće je korišćenje i drugih alternativa
- Synchronous vs. asynchronous
- Moguće je definisati četiri tipa servisnih metoda:
  - Unarni RPC – klijnt šalje jedan zahtev serveru i dobija jedan odgovor
  - Server streaming RPC – klijent šalje zahtev serveru i dobija tok odakle može da čita sekvencu poruka
  - Client streaming RPC – klijent šalje sekvencu poruka serveru preko toka, server vraća jedan odgovor
  - Bidirectional streaming RPC – obe strane šalju sekvence poruka preko read-write toka

Korišćenje API-ja

- Počevši od definicije servisa u .proto fajlu, gRPC obezbeđuje pluginove za kompajliranje protobufera koji generišu klijentski i serverski kod.
- Korisnici često zovu ove API-je na klijentskoj strani i implementiraju odgovarajuće API-je na serverskoj strani

RPC life cycle – Unary RPC

- Kada klijent pozove stub metodu, server dobija notifikaciju da je pozvan RPC sa metapodacima klijenta
- Server može da vrati svoje metapodatke odmah (pre bilo kakvog odgovora) ili da čeka klijentski zahtev. Od aplikacije zavisi i implementacija.
- Jednom kada server primi klijentski zahtev, obrađuje ga i popunjava odgovor. Ako je uspešno, odgovor se vraća klijentu zajedno sa statusnim detaljima (status kod i opciona poruka) i opcionim metapodacima.
- Ako je status odgovora OK, klijent dobija odgovor čime se kompletira poziv na klijentskoj strani.

RPC life cycle – Server streaming RPC

- Slično kao kod unary RPC
- Server sada vraća niz poruka kao odgovor na klijentov zahtev
- Nakon što pošalje sve poruke, status servera i opcioni metapodaci se šalju klijentu. Ovim se kompletira procesiranje na serverskoj strani
- Klijent kompletira metodu kada primi sve poruke koje je server poslao

RPC life cycle – Client streaming RPC

- Slično kao kod unary RPC
- Klijent sada šalje niz poruka kao zahtev serveru
- Server odgovara jednom porukom zajedno sa statusom i opcionim metapodacima
- Server uglavnom šalje odgovor nakon što primi sve poruke klijenta

RPC life cycle – Bidirectional streaming RPC

- Komunikacija se inicira kada klijent pozove metodu a server primi metapodatke klijenta, ime metode i rok.
- Server može odgovoriti slanjem inicijalnih metapodataka ili čekati da klijent započne slanje.
- Procesiranje podataka i sa klijentske i sa serverske strane je specifično za aplikacije.
- Dva toka podataka su nezavisna, pa klijent i server mogu čitati i pisati poruke u bilo kom redosledu.

Deadlines/Timeouts, Cancelling

- gRPC dozvoljava klijentima da specificiraju koliko će čekati na odgovor. Server ima mogućnosti da proveri da li je neki RPC istekao
- I klijent i server imaju mogućnost da prekinu RPC u bilo koje vreme

C# podrška za .proto fajlove

- .NET tipovi za serise, klijente i poruke se automatski generišu ubacivanjem *.proto fajlova u projekat
- ASP.NET Core može hostovati gRPC servise
  - gRPC servisi imaju pun pristup opcijama poput logovanja, DI, autentifikacije i autorizacije.

Poziv gRPC servisa iz .NET klijenata

- gRPC klijenti su konkretni tipovi generisani pomoću .proto fajlova
  - Imaju metode koje prevode gRPC servise u .proto fajlove
- Kreiraju se korišćenjem kanala koji predstavljaju dugoročnu konekciju sa gRPC servisom.

Literatura

- Introduction to gRPC on .NET
- ASP.NET Core gRPC for WCF Developers
- Tutorial: Create a gRPC client and server in ASP.NET Core