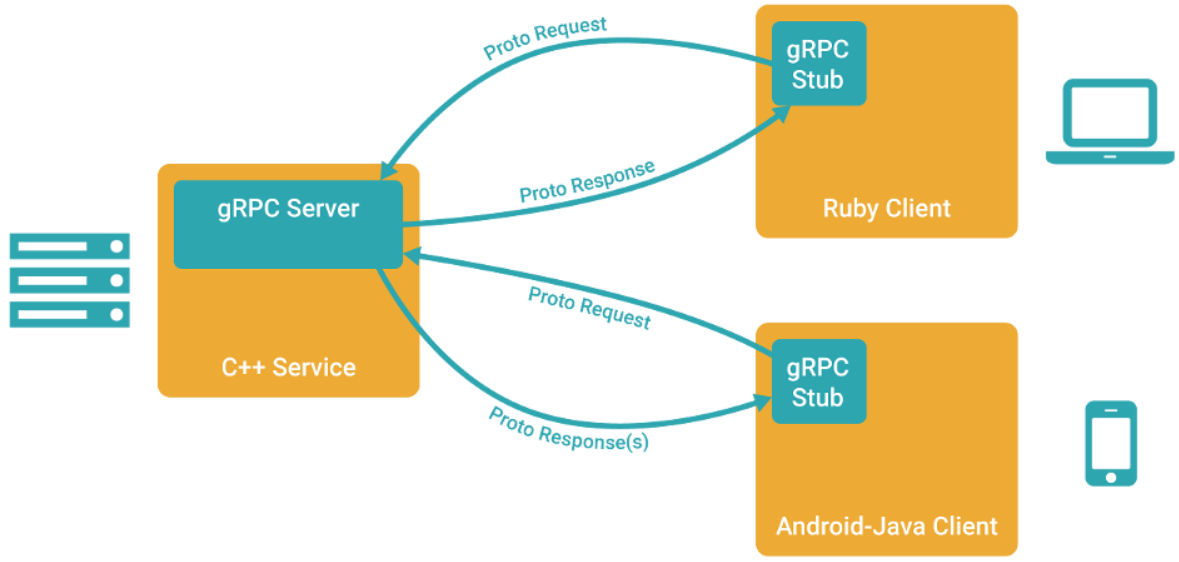
gRPC ile Redis Bağlantısı

NUR SEDA GÜZAY

1.1. RPC (REMOTE PROCEDURE CALL/UZAKTAN YORDAM ÇAĞRISI) NEDİR?

Server ve client arasındaki iletişim için tasarlanmıştır. Kullanılan herhangi bir program sunucu-istemci iletişimine ihtiyaç duyuyorsa prosedürü sağlayan bilgisayar programına izin veren süreçler arası iletişim teknolojisidir. Kısaca bir sunucunun bir istemcide kod çalıştırmasına yarayan programdır.

1.2.gRPC NEDİR?



gRPC; transport, iletişim, veri iletimi için HTTP/2 protokolünü kullanmaktadır. Dolayısıyla gRPC'nin faydalarını anlayabilmek için önce HTTP /2'nin HTTP /1'e nazaran getirilerini anlamak gerekmektedir.

HTTP /1; her bir statik dosya için (.css, .js, .png vs) ayrı istek göndermektedir. Bu durum ise yük ve maliyeti arttıracağından dolayı ekstradan bekleme süresinin artmasına sebep olmaktadır.

HTTP /2'de ise dosyalar için tüm istekler tek seferde toplu olarak yapılabilmektedir. Böylece açılış hızı artmakta, süresi düşürülmektedir. Bu duruma "multiplexing" denmektedir. Teknik olarak, tek bir TCP bağlantısı üzerinde birden çok ve paralel request ve response yeteneği olarak yorumlanabilir.

HTTP /1, metin tabanlı (text based) bir protokoldür. Her requestte sıkıştırılmamış vaziyette header gönderilir. Bir requeste bir response döner.

HTTP /2'de client ile server arasındaki iletişim binary formattaki küçük framelere ayrılmaktadır. Bu duruma "Binary Protocol" denmektedir. Her requestte headerler HPACK ile sıkıştırılarak gönderilmektedir. Bu duruma "Header Compression" denmektedir. Bir requeste karşılık birden fazla response alınabilir (server push).

gRPC; CPU, bellek ve bant genişliği kullanımının önemli ve kritik olduğu uygulamalarda JSON yerine binary formatta çalıştığı için tercih edilebilecek bir teknolojidir.

Transfer edilecek dataları Google tarafından geliştirilmiş, binary serialization protokolü olan “Protocol Buffers (Protobuf)” kullanarak iletmekte ve seralize ve deseralize etmektedir. Yani veriyi JSON yahut XML gibi text yerine binary formata çevirmekte ve diğer formatlara göre hatrı sayılır miktarda hız ve performans elde edebilmektedir.

Binary format her ne kadar JSON gibi insan seviyesinde okunabilir bir format olmasa da JSON’a nazaran daha hızlı işlenebildiğinden ve ayrıca herhangi bir string parsing işleminin olmamasından dolayı tercih edilmektedir. Tabi ihtiyaç doğrultusunda JSON gibi text tabanlı formatları da desteklemektedir.

1.2.1.PROTOCOL BUFFERS (PROTOBUF) NEDİR?

Google’ın geliştirdiği ve hala gelişmekte olan bir binary serialization protokolüdür. Özünde bir Arayüz Tanımlama Dili/Interface Definition Language (IDL)’dir. Kullanılan platform ve programlama dili farkını gözetmeksizin, client ve server arasında haberleşmeyi sağlayabilmek için IDL compiler sayesinde her iki tarafa da (client ve server) “stub” ismi verilen gerekli arayüzlerin oluşturulmasını sağlayan bir dildir. İçerik olarak gRPC servis tanımlarını ve iletişim sürecinde kullanılacak olan mesaj tanımlarını tutmaktadır.

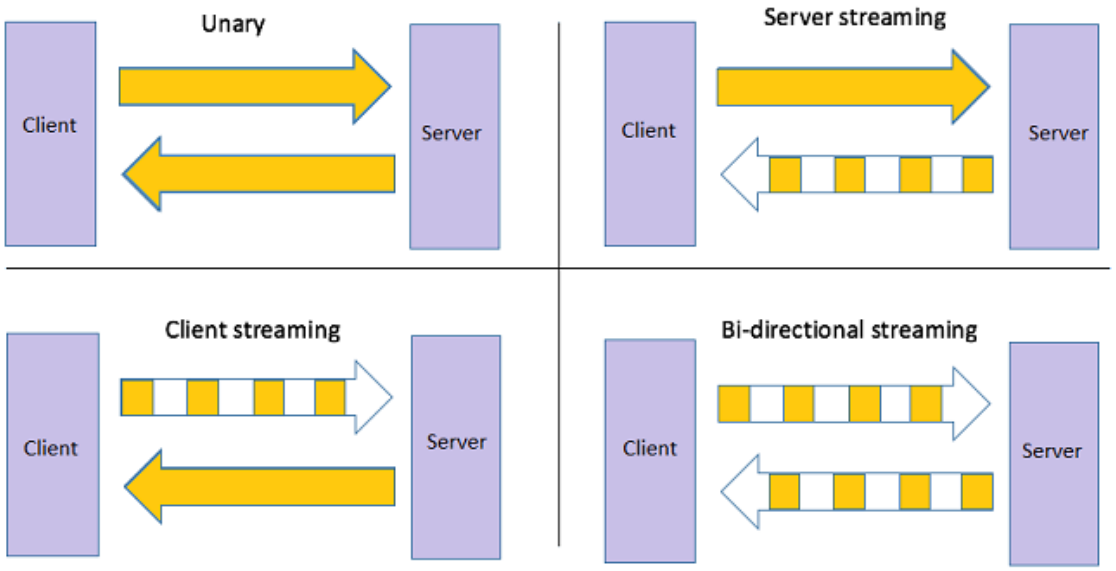
1.2.2.gRPC vs RESTful

Rest servislerde yapılan request neticesinde response’un alınabilmesi için gönderilen tüm dataların topluca işlenmesi gerekmektedir. Halbuki gRPC’de ise yapılan request neticesinde response, tüm verilerin işlenmesini beklemeksizin alınabilmekte ve veriler parça parça işlendikçe bütünden bağımsız bir şekilde response edilmektedir. Bu durum gRPC’de “Data Stream” olarak nitelendirilmektedir.

gRPC isteklerinde encoding ve decoding işlemleri istemcide gerçekleştirilmektedir. Böylece bu işlemlerin yükü serverdan arındırılmış olmaktadır.

gRPC’de server ve client arasındaki haberleşme için iletilecek mesajın türünü ve iletin yöntemini bildirecek bir servis sözleşmesi gerekmektedir. Bu sözleşme her iki uygulamada da (client ve server) “proto” dosyası olarak ayarlanmaktadır.

1.2.3.gRPC İLETİŞİM TÜRLERİ

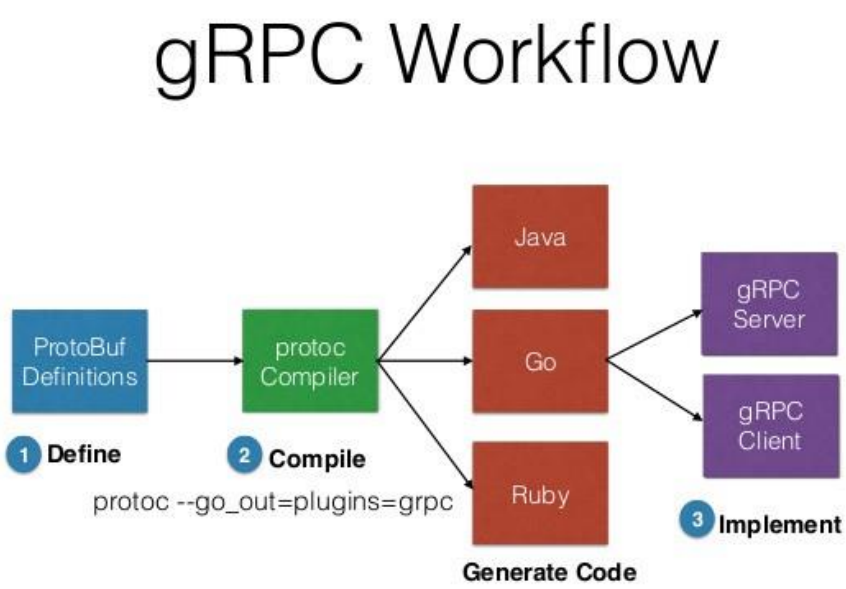


Unary: Clientın servera tek bir istek gönderdiği ve normal bir işlev çağrısı gibi tek bir yanıt aldığı RPC türüdür.

Server Streaming: Clientın servera tek bir istek gönderdiği ama serverın stream dönmeye başladığı RPC türüdür.

Client Streaming: Clientın servera stream olarak istek gönderdiği ama serverdan tek bir yanıt aldığı RPC türüdür.

Bi-directional Streaming: Client ve serverın stream döndüğü RPC türüdür.



2.1. PROJE TANIMI

gRPC ile Remote Redis bağlantısı kurularak istenilen JSON tipindeki veriyi çekme (NODE.JS).

2.2. KULLANILAN KÜTÜPHANELER

- @grpc/grpc-js
- @grpc/proto-loader
- redis
- redis-json

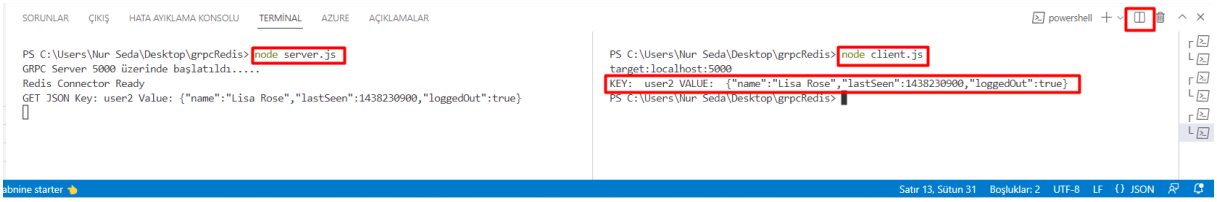
package.json dosyası;

{} package.json > {} dependencies

```
{
  "name": "grpcredis",
  "version": "1.0.0",
  "description": "gRPC İLE REDIS BAĞLANTISI",
  "main": "server.js",
  ▷ Hata Ayıkla
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "Nur Seda Güzay",
  "license": "ISC",
  "dependencies": {
    "@grpc/grpc-js": "^1.7.0",
    "redis": "^4.3.1",
    "redis-json": "^6.0.3"
  }
}
```

2.3. PROJEYİ ÇALIŞTIRMAK

Projeyi çalıştırmak için gösterilen yerden terminal ikiye bölünür. Görselde sol kısım server için, sağ kısım client için kullanılmıştır. “node server.js” ve “node client.js” komutlarıyla ilk önce server, daha sonra client tarafı çalıştırılır. Programın hata vermeden çalışması için Redis bağlantısı yapılacak olan bilgisayarda Redis server açık olmalıdır. Diğer türlü Redis bağlantı hatası verecektir.



3.1. NOTLAR

- Proje yapılırken IDE olarak VSCODE kullanıldı.
- Proje Node.js ile yapıldı.
- Node.js versiyonu v18.7.0
- RedisConnector.js ve config.js import edilerek kullanıldı.
- Uygulamanın çalıştırılacağı bilgisayarda Redis yüklü olmak zorunda değil.
- gRPC bağlantısı localde oluşturuldu.

```
server.bindAsync("127.0.0.1:5000", grpc.ServerCredentials.createInsecure(), (err) => {
  server.start();
  console.log("GRPC Server 5000 üzerinde başlatıldı.....");
});
```

```
var client = new grpc_proto.grpcServ('localhost:5000', grpc.credentials.createInsecure());
```

- Redis bağlantısının yapılacağı bilgisayar config.js dosyasında belirtildi;

```
8     "redisSettings": {
9         "redisAddress": "redis://192.168.1.32:6379",
10        "username" : "default",
11        "password" : ""
12    },
```