

GRPC

REALTIME MICROSERVICES

GRPC MIT .NET

ÜBERSICHT

- ▶ Über mich
- ▶ HTTP/2 in einem Chart
- ▶ Protobuf Grundlagen
- ▶ GRPC = HTTP/2 plus Protobuf
- ▶ Nuget Pakete und Code Generierung
- ▶ Live-Coding Beispiel
- ▶ Ressourcen

ÜBER MICH

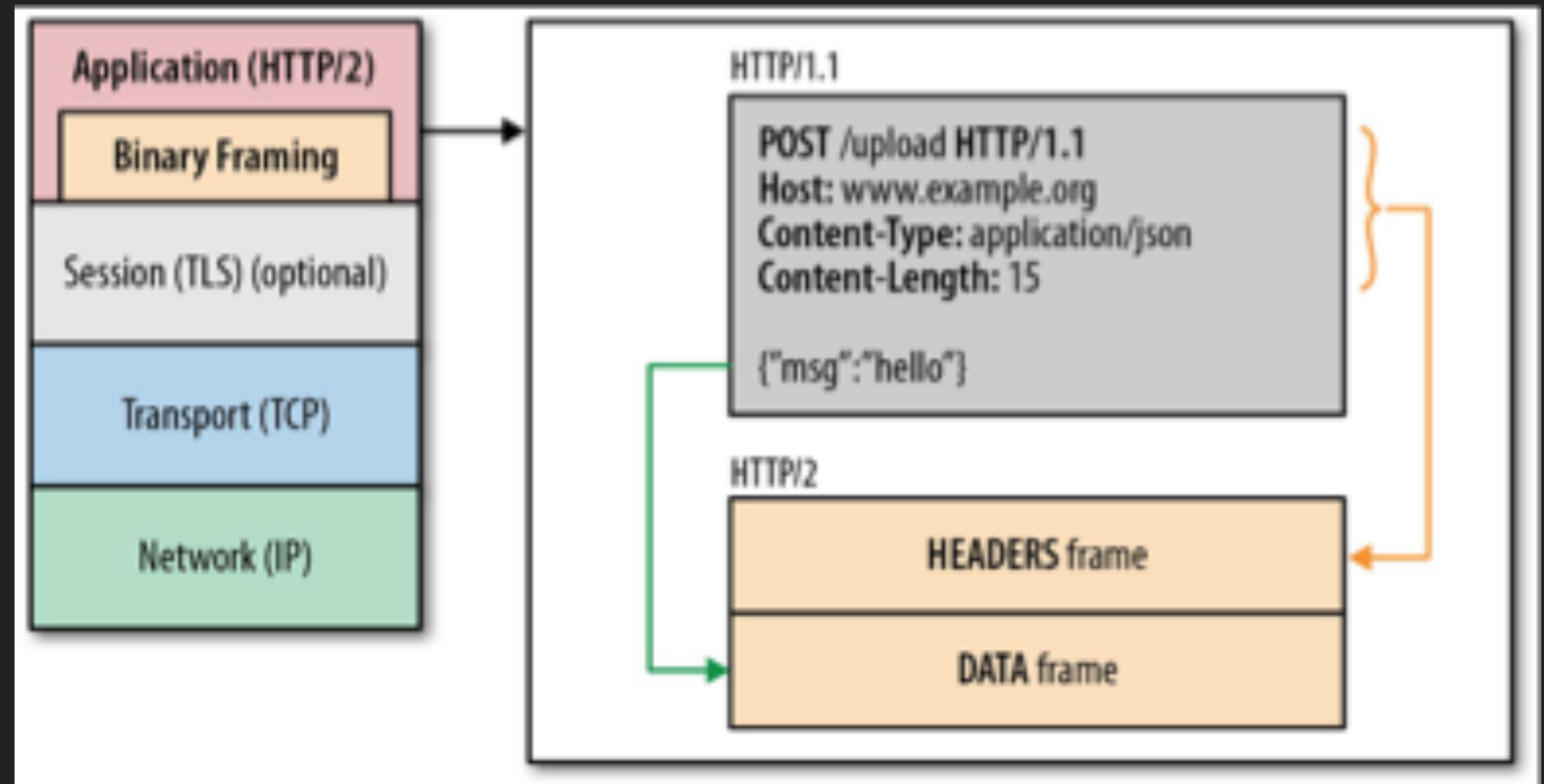
- ▶ Studium Wirtschaftsinformatik
- ▶ 5 Jahre Accenture
- ▶ 5 Jahre ISV für Börsenhandel
- ▶ 10 Jahre CTO bei einer Börsenhandelsgesellschaft
- ▶ Täglich 250 - 400 Mio. Nachrichten werden umgewandelt in Orders und Trades

GRPC ÜBERBLICK

- ▶ Öffentlicher Nachfolger von Stubby bei Google für Microservices
- ▶ GRPC = HTTP/2 + Protocol Buffer

HTTP/2 IN EINER FOLIE

- ▶ Eine TCP Verbindung
- ▶ Request -> Stream
 - ▶ Streams mit Multiplex
 - ▶ Streams mit Priorität
- ▶ Binäre Rahmen Schicht
 - ▶ Periodisierung
 - ▶ Flow Control
 - ▶ Server Push
- ▶ Header Kompression



PROTOBUF

- ▶ Strukturierte Darstellung von Daten
- ▶ Googles allgemeine Beschreibungssprache für Daten
 - ▶ 48.000 Nachrichtentypen
 - ▶ 12000 Proto files
- ▶ Evolutionäre Entwicklung (zurzeit Version 3)

WARUM PROTOBUF

- ▶ Effiziente Darstellung von Daten
- ▶ Klare Kompatibilitätsregeln
- ▶ Idiomatisch - eine Darstellung für alle Sprachen (C++, C#, Java, Java Script, Objective-C, Ruby, Python, Go)
- ▶ Stark typisiert für Performance und Sicherheit

NACHRICHTENFORMAT (PROTO 3)

- ▶ Eindeutige nummerierte Felder
 - ▶ Typisiert
 - ▶ Hierarchisch
 - ▶ Arrays
 - ▶ Erweiterbar durch zusätzliche Felder
- ```
syntax = „proto3“

message Person {

 string Name = 1;

 int32 Id = 2;

}
```



## PROTOBUF SKALARE WERTETYPEN

- ▶ double
- ▶ float
- ▶ int32/int64
- ▶ bool
- ▶ string
- ▶ bytes

## PROTOBUF ENUMS

- ▶ Enums ähnlich wie C#

```
enum Corpus {

 UNIVERSAL = 0;

 WEB = 1;

 IMAGES = 2;

}

Corpus corpus = 4;
```

## PROTOBUF KOMPOSITION

- ▶ Strukturen können andere Strukturen enthalten
- ▶ .Proto Dateien können einander referenzieren

```
message SearchResponse {
 repeated Result results = 1;
}
```

```
message Result {
 string url = 1;
 string title = 2;
 repeated string snippets = 3;
}
```

```
import "myproject/other_protos.proto";
```

# PROTOCOL BUFFER COMPILER

<https://github.com/google/protobuf>

# GRPC

- ▶ Service Definition mittels .proto Datei
- ▶ Code Generierung in 10 Sprachen - in .NET via Nuget Pakete und mit einem C-Kern
- ▶ Nutzt Http/2
- ▶ Punkt zu Punkt Verbindung
- ▶ Streaming / Uni und Bidirektional
- ▶ Authentifizierung via SSL/TLS und OAUTH2

## GRPC VS SIGNALR

- ▶ SignalR funktioniert über Websockets
- ▶ SignalR ist eine reine .NET Architektur
- ▶ GRPC hat Vorteile, wenn man sprach-unabhängig sein will
- ▶ GRPC wird zurzeit nicht direkt in Webseiten unterstützt

## PROJEKTSTRUKTUR GRPC

- ▶ ServiceDefinition Projekt
  - ▶ .Proto mit Definitionen
  - ▶ Generierter Code
- ▶ ServiceClient Projekt
- ▶ ServiceServer Projekt

```
syntax = "proto3";

package helloworld;

service Greeter {

 rpc SayHello (HelloRequest)
 returns (HelloReply) {}

 message HelloRequest { string
 name = 1; }

 message HelloReply { string
 message = 1; }
```

# LIVE CODING STRUKTUR

- ▶ 3 Projekte aufsetzen
- ▶ Nuget Pakete referenzierten
- ▶ .proto Datei erstellen
- ▶ Code generieren
- ▶ Generierten Code im ServiceDefinition Projekt hinzufügen und Kompilieren
- ▶ Im Client und Server Projekt das Service Definition hinzufügen
- ▶ Schnittstellen implementieren
- ▶ Vorführung anhand der Kommandozeile



# ERFAHRUNGEN

- ▶ Server API ist ein modernes async/await .NET API
- ▶ Einfache und schnelle Implementierung
- ▶ Keine decimals
- ▶ Keine nullübles

## RESSOURCEN

- ▶ <http://www.grpc.io/>
- ▶ <https://github.com/grpc/grpc>
  - ▶ Wichtig: Universelles Projekt für alle Sprachen
  - ▶ Allgemeine und .NET spezifische Verzeichnis