

# BACKEND

NodeJS & API-Technologien

# AGENDA

Softwaredemo

NodeJS

API-Technologien

- gRPC
- REST
- GraphQL
- Vergleich

# NODEJS — ALLGEMEIN

Serverseitiges Ausführen von JavaScript

Google Chrome's JavaScript-Laufzeitumgebung V8

Skalierbare Netzwerkanwendungen

NPM

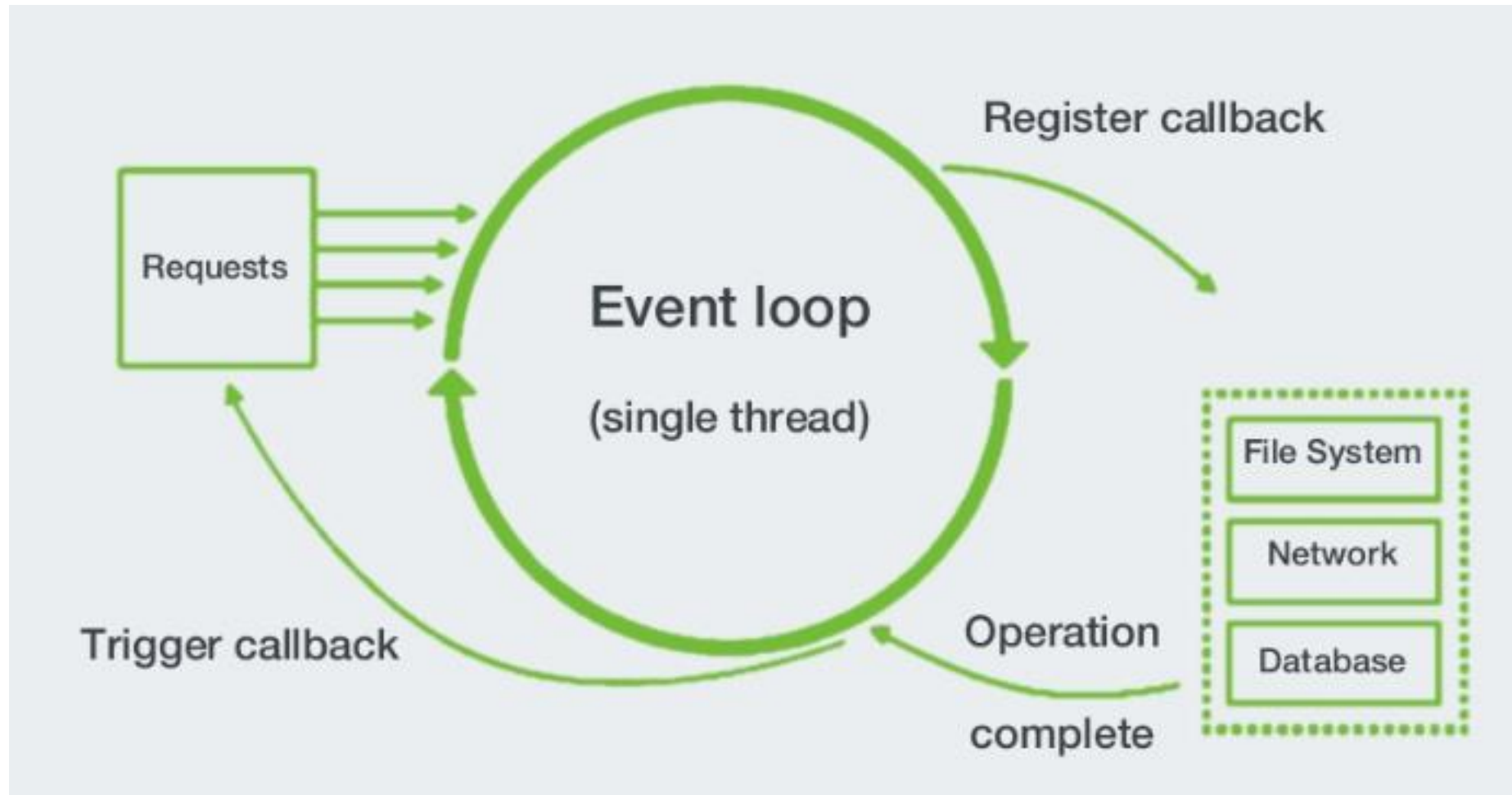
# NODEJS - NPM

Weltgrößte Software-Register

Module installieren und veröffentlichen

Dependency-Management

# NODEJS — EVENTLOOP



# NODEJS — JAVASCRIPT END-TO-END

V8 ermöglicht performante serverseitige Skriptsprache

Bekannte Skriptsprache für Web-Entwickler

Kein Kontextwechsel

Code-Sharing

# NODEJS — NACHTEILE

Experimentelles Multi-Threading

Kein Multi-Threading in LTS Version

# API-TECHNOLOGIEN

gRPC

REST

GraphQL



# GRPC - ALLGEMEIN

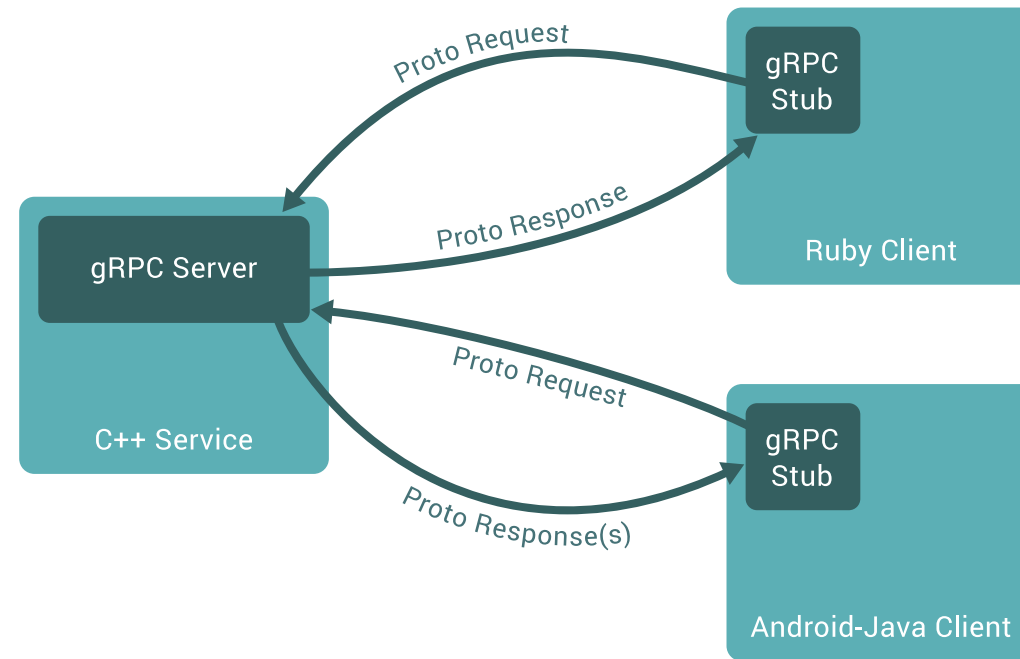
Modernes RPC Framework

Server-Methodenaufrufe vom Client als wäre es ein lokales Objekt

HTTP/2

Protocol Buffers

# gRPC — METHODENAUFRUF



# GRPC — HTTP/2

ServerPush

Kommunikation auf einem Kanal

Stream Priorisierung

Kompression des Headers

→ Höhere Geschwindigkeiten und weniger Overhead

# GRPC — PROTOCOL BUFFERS

Serialisierung strukturierter Daten

Ähnlich zu JSON/XML

Generiert Klassen mit Getter-/Setter-Methoden

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

# GRPC — VORTEILE

Performance

Geringe Datengröße

# GRPC — NACHTEILE

## Browserinkompatibilität

- Aktuelle gRPC-Bibliotheken durch technische Limitierungen beschränkt
- Proxydienst

# REST - ALLGEMEIN

Bewährte API-Technologie

Seit 2000 von Roy Fielding

Kommunikation über HTTP-Anfragen

- GET
- POST
- PUT
- DELETE

# REST — SECHS PRINZIPIEN

Client-Server

Stateless

Cache

Uniform-Interface

Layered System

Code-On-Demand



# REST — VORTEILE

Ausgereifte Lösung

Methoden-Leitfaden

# REST — NACHTEILE

Fehlerhafte REST-Implementierungen

Overhead durch die Prinzipien

# GRAPHQL — ALLGEMEIN

Von Facebook entwickelt

Query-Anfragen

Viele Client-Arten

GraphiQL

# GRAPHQL — VORTEILE

Optimale Datenrückgabe

# GRAPHQL — NACHTEILE

## Standardmäßig kein Caching

- Nur ermöglicht durch Client-Bibliotheken

## Kein Datei-Upload

- Benötigt zusätzliche Bibliothek oder REST-API

# VERGLEICH

gRPC	REST	GraphQL
+ Schnelle Datenübertragung	+ Bewährte Technologie	+ Effiziente Rückgabedaten
+ Weniger Anfragen durch HTTP/2	+ Discoverability durch HATEOAS	+ Weniger Anfragen durch Query-Dynamik
- Browsersupport	- Overhead durch Prinzipien	+ Dynamisch für viele verschiedene Clientarten
	- Große Payloads oder viele spezielle Methoden	- Kein Fileupload
	- Viele Aufrufe	- Kein Caching

**Mikroservices**

**Komplexe Methoden**

**Komplexe Datenstrukturen  
oder viele Client-Arten**

# NODEJS & API-TECHNOLOGIEN

Fragen?