gRPC, GraphQL, REST APIs which way to go?

Phil Wilkins

Phil.Wilkins@capgemini.com

uk.linkedin.com/in/philWilkins

@PhilAtCapgemini /

@MP3Monster

Oracle-integration.cloud /

APIPlatform.cloud /

Blog.mp3monster.org

Today ....

1. Introduction
2. Evolution of Communication Strategies
3. An Overview of:
   - REST
   - gRPC
   - GraphQL
4. Side by Side Pros & Cons
5. Recommendation

Capgemini is One of the World's Largest Consulting, Technology, and Outsourcing Firms & a global "full service" business transformation provider

Revenue

12,8

Billion EUR (2017)

Net Profit

€1,18B

5 Businesses

Application Services
Infrastructure Services
Business Process Outsourcing
Consulting (Capgemini Consulting)
Local Professional

4 Elements to successful collaboration

- Targeting Value
- Mitigating Risk
- Optimising Capabilities
- Aligning the Organisation

7 values

Honesty, Boldness, Trust
Freedom, Team Spirit,
Modesty, Fun

6 Strategic Partners

EMC² – HP – IBM – MICROSOFT
ORACLE – SAP

EXPERTISE 6 in Major Sectors

Telecom, Media & Entertainment	Manufacturing, Engineering & Construction	Consumer Products, Retail & Distribution	Financial Services	Public Sector	Energy, Utilities & Chemicals

Group Workforce: 200,000+ Globally

Across 40+ countries, 100 nationalities

North America
UK & Ireland

Central Europe
Morocco

Nordics
Benelux

Asia Pacific
Latin America

"It is the quality of our people, and their capacity to deliver fitting solutions, with you and for you, that drive real business results."

Capgemini & Oracle

Alliance and Strategic Partnership

- Cloud Premier Partner
- Oracle Diamond Partner
- Oracle Cloud Managed Service Provider (*New!) partner – only a handful of SI's
- Only Global SI to be accredited as Oracle Authorized Education Center
- Part of Beta programmes for:
  - Cotainer Native & Microservices
  - Inteligent Chatbot
  - API platform
  - Integration cloud
  - Process cloud
  - Oracle Self-Service Automation
  - Oracle IoT Cloud
  - Oracle Mobile Cloud

Awards & Recognitions

- 2018 –Global Excellence Award for Extend and Connect
- 2018 –Silver Awards for Managed Services, Middleware & infrastructure Services - UKOUG Partner of the Year
- 2018 – PaaS & API Community Awards
- 2017 – Gold & Silver UKOUG Partner of the Year Awards
- 2017 – API 2017 – Global Excellence Award for Extend and Connect
- & PaaS Community Award
- 2017 – Chatbot PaaS Community Award
- 2016 – Oracle Specialized Partner of the Year: Industry
- 2016 – Oracle University Partner of the Year
- 2016 – BPM and Cloud community awards
- 2015 – Oracle Customer Support Services Partner of the Year
- 2011 – Global Partner of the Year Award for Oracle Applications
- 2012 – Fusion Middleware partner of the year
- 2010 – Partner of the year for Oracle Fusion Middleware
- 2010 – 2010 EMEA Industry Partner of the Year
- 2010 – Oracle Customer Services Partner of The Year
- 2009 – Oracle Customer Services Partner of The Year
- 2008 – Oracle Customer Services Partner of The Year

Thought Leadership

- Continuous investments in cloud accelerators
- 5 Oracle Aces: 2 Directors, 3 Aces
- Real experts and thought leaders including several books:
  - 2013: Oracle SOA Governance Implementation
  - 2015: Oracle API Management Implementation
  - 2016: Oracle Case Management Solutions
  - 2017: Implementing Cloud service
  - Oracle API Platform CS Implementation
  - Enterprise API Management
- Several publications in OTN, Oracle Magazine, Oracle Scene & Other

About: Phil Wilkins

- Technical Enterprise Architect specializing in Integration, APIs and PaaS.
- Started out as a developer working on Radar systems
- Moved into integration space – using Open Source Tech e.g. JBoss App Server & Fuse, Apache Camel etc.
- Worked with Oracle middleware & PaaS >9yrs
- Worked in end user companies, ISVs & consultancy.
- Oracle Ace
- Co-authored books on API Platform CS & Oracle Integration Cloud
- contributing to development of more than a dozen other titles ranging from Apache Camel to Cloud Computing Design
- Articles published in a range of journals on Cloud Strategy, PaaS, Integration & APIs.

Phil Wilkins

Phil.Wilkins@capgemini.com

uk.linkedin.com/in/philWilkins

@PhilAtCapgemini / @MP3Monster

Oracle-Integration.Cloud / APIPlatform.cloud / Blog.mp3monster.org

Co-Author 1st Oracle iPaaS Book Implementing Oracle Integration Cloud - Jan, 2017

Co-Author of Implementing Oracle API Platform - Mar, 2018

Peer technical review on a variety of books published by Thomas Erl (Prentice Hall), Packt etc

Articles published in a range of Journals

THE Open GROUP

TOGAF® 9 Certified

TOGAF 9 Certified
2013

LONDON

ORACLE®

{ developer } Meetup

#OracleDeveloperMeetup

https://www.meetup.com/Oracle-Developer-Meetup-London/

http://bit.ly/OracleDevMeetup

Phil Wilkins

Phil.Wilkins@capgemini.com

uk.linkedin.com/in/philWilkins

@PhilAtCapgemini /

@MP3Monster

Oracle-integration.cloud /

APIPlatform.cloud /

Blog.mp3monster.org

The Book …

https://apiplatform.cloud

https://www.packtpub.com/virtualization-and-cloud/oracle-api-platform-cloud-service

API and its Evolution

What is an API?

In computer programming, an **application programming interface** (**API**) is a set of subroutine definitions, communication protocols, and tools for building software. In general terms, it is a set of clearly defined methods of communication among various components. A good API makes it easier to develop a computer program by providing all the building blocks, which are then put together by the programmer.

Wikipedia

What is an API?

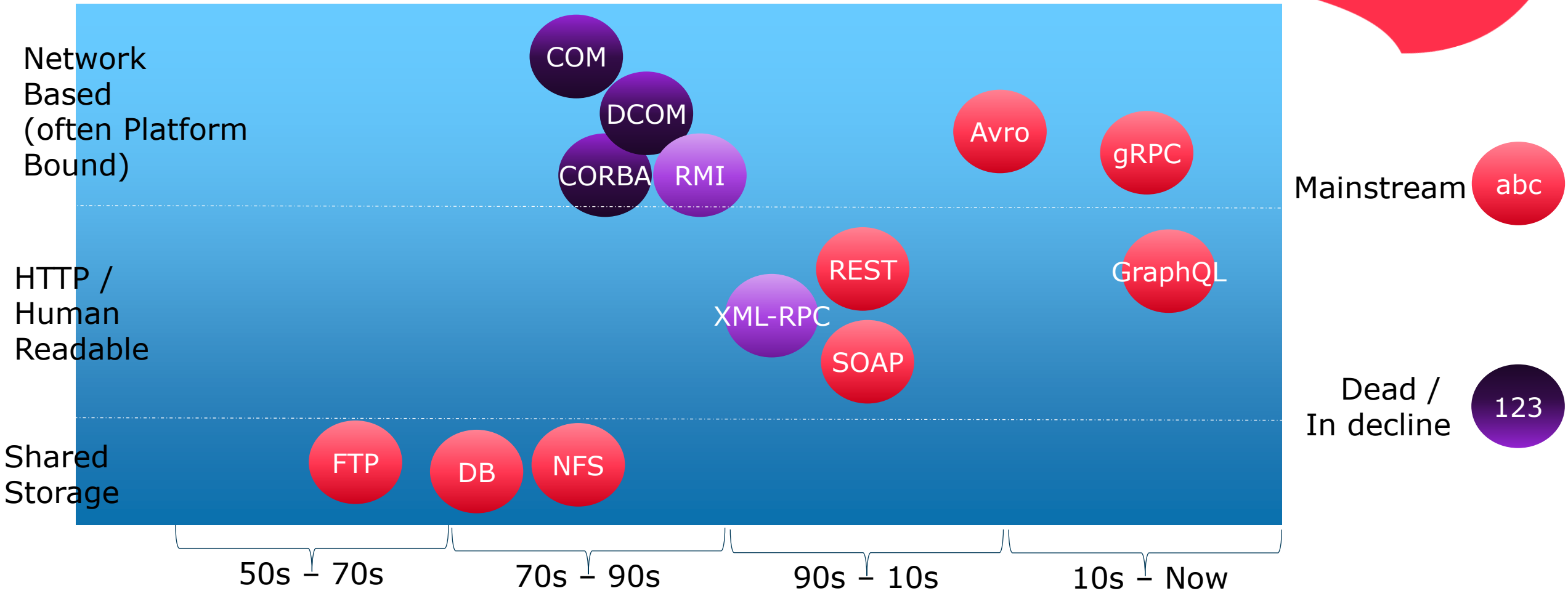
In computer programming, an **application programming interface** (**API**) is a set of subroutine definitions, **communication protocols**, and **tools for building software**. In general terms, it is a set of clearly defined methods of **communication among various components**. A good API **makes it easier to develop** a computer program by providing all the building blocks, which are then put together by the programmer.

Wikipedia

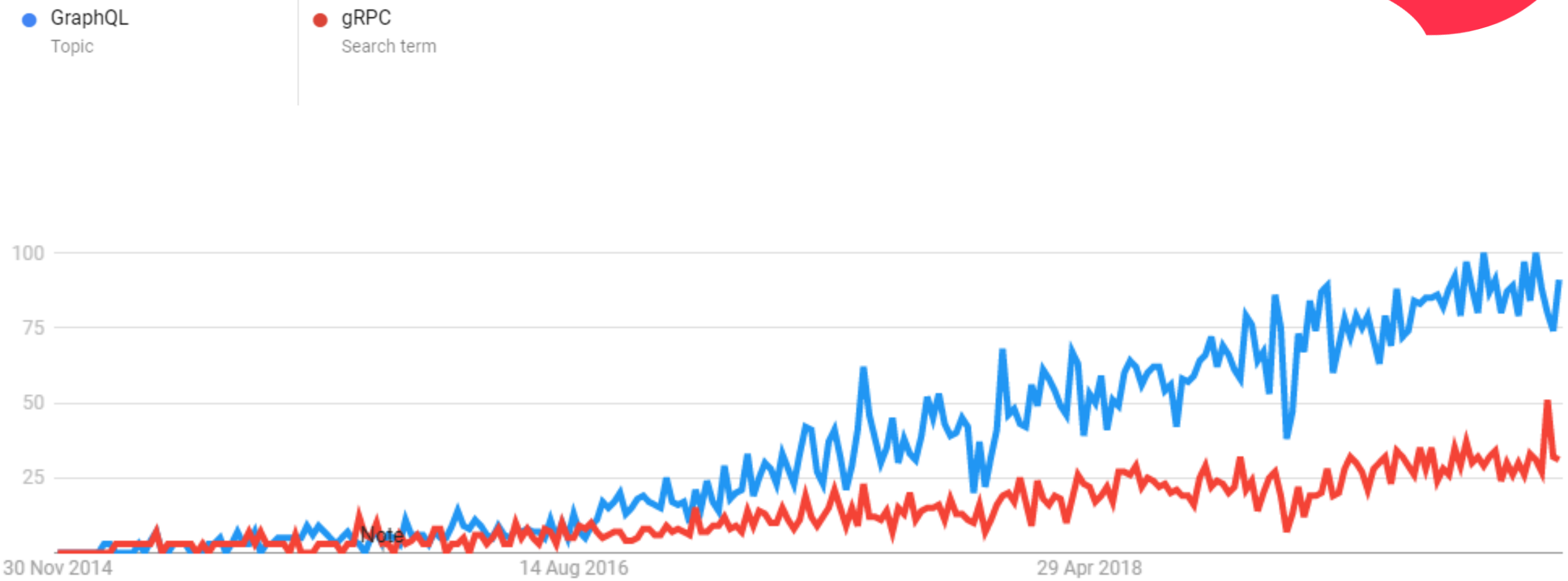
A **Web API** is an **application programming interface** for either a **web server** or a **web browser**. It is a web development concept, usually limited to a web application's client-side (including any web frameworks being used), and thus usually does not include web server or browser implementation details such as SAPIs or APIs unless publicly accessible by a remote web application.

Wikipedia

Evolution of Technologies Used for Data Sharing / Inter-Process Collaboration



Impact / Adoption



Data taken from Google Trends on 24-11-19

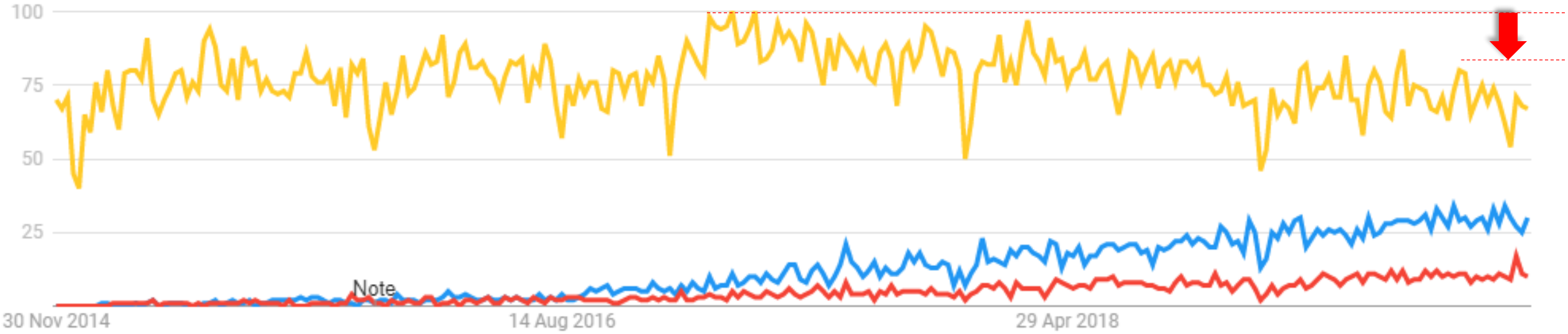
Impact / Adoption

GraphQL
Topic

gRPC
Search term

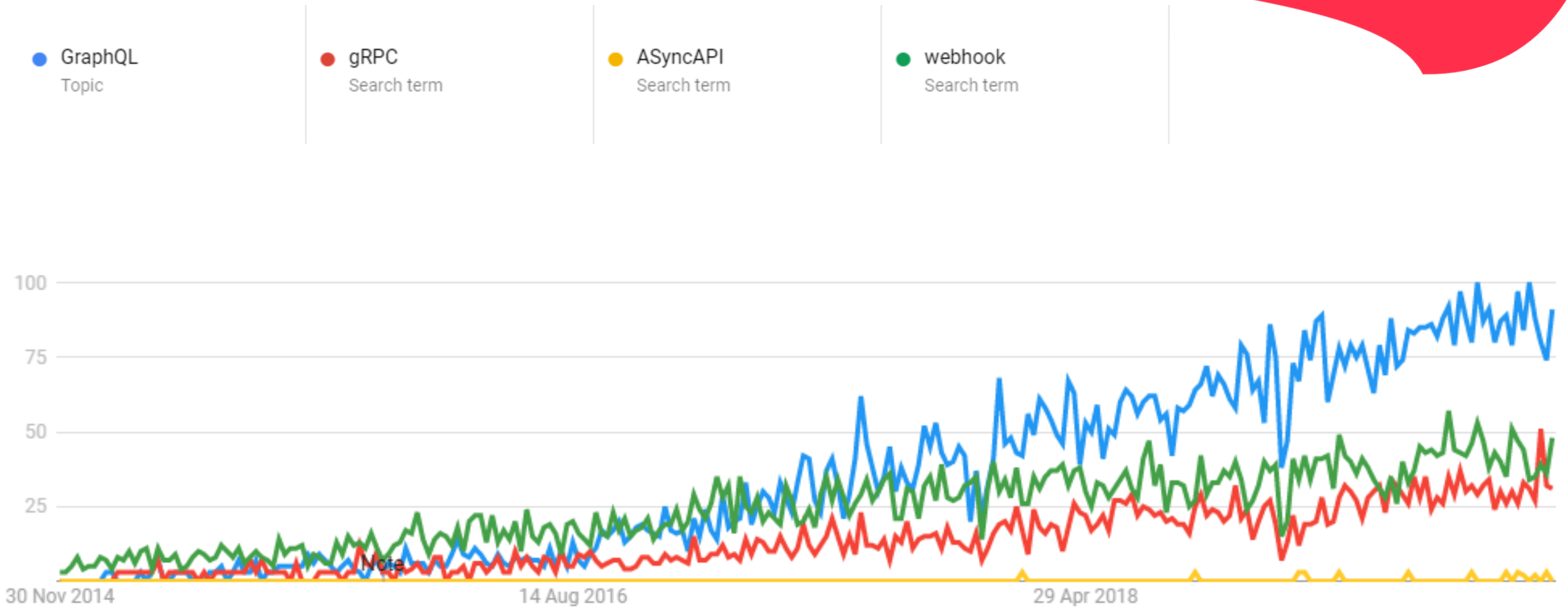
RESTful
Search term

Any assertions that REST is dead – are mistaken all the alternates have at most impacted by 15-20%

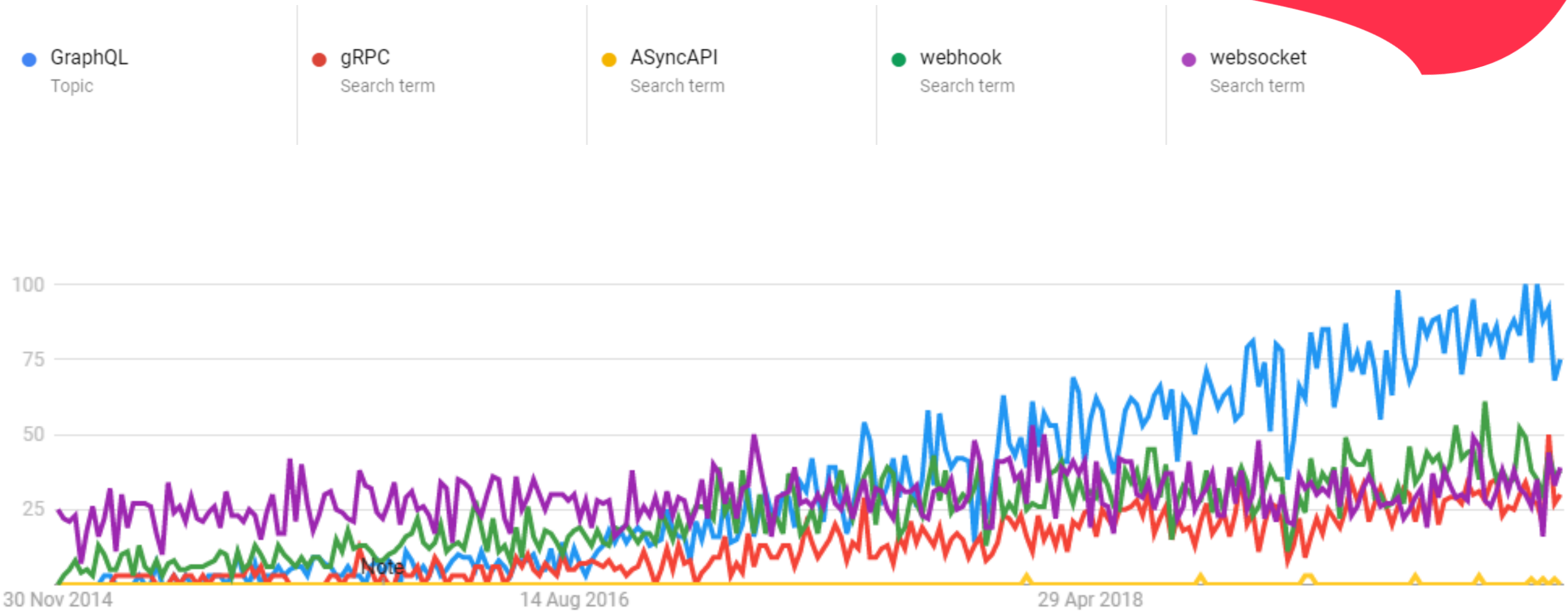


Data taken from Google Trends on 24-11-19

Impact / Adoption



Data taken from Google Trends on 24-11-19



Data taken from Google Trends on 24-11-19

RESTful Introduction

Useful Resources:

- restfulapi.net/
- docs.oracle.com/javaee/6/tutorial/doc/gijqy.html
- roy.gbiv.com/talks/webarch_9805/index.htm

What is a REST API?

Representational State Transfer (REST) - First defined by Roy Fielding in 2000 as part of his PhD
http://bit.ly/RESTPaper

01

Uniform Interface
- Resource Based
- Self describing messages
- HATEOAS

02

Resource manipulation through representations

03

Client/Server

04

Stateless
- Everything needed is provided as part of the request

05

Code on Demand (Optional)
- Ability to pass code for execution e.g. JavaScript to be executed

06

Layered
- Client does not know (or need to) whether is connecting to the source or anything between

07

Cacheable
- Client can cache the responses
- Tell the client whether it is safe to cache

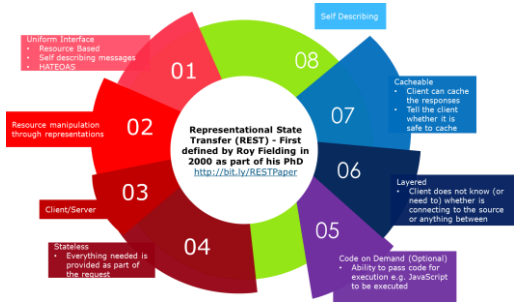
08

Self Describing

REST API Overview

Verb	Description	Idem-potent	Correct HTTP Response Codes
Get	Used for retrieving information, but we can make it conditional with headers for example If-Modified-Since.	Yes	• 200 • 404 (Not Found), if ID not found or invalid.
Post	Typically used for delivering updates, as the URL is known. But can be used for a Create	No	• 200 success with a response payload • 201 if the result is a new entity
Put			
Delete			

Header	Description	Examples
Accept	A family of header values describing the payload types that can be handled, to character sets, encoding, language	• Accept-Encoding: gzip, deflate • Accept-Language: en-US
Accept-Control-Request-Method	For helping to manage Cross Origin use cases	• Access-Control-Request-Method: GET
Content-Length, Content-MD5	Describing the content, can be used to help detect tampering for example	• Content-MD5: dfghkojdfgbfgb==
If-Unmodified-Since	Only send me the data if it has changed since	• If-Unmodified-Since: Sat, 29 Oct 1994 19:43:31 GMT
If-Match	Only perform the action if the client supplied entity matches the same entity on the server. This can be used as an aid to locking	• If-Match: "v123345"
Warning	Allows warning information about the payload.	• Warning: 199 Miscellaneous warning
Expect	The client can ask the server to behave in particular ways	• Expect: 100-continue

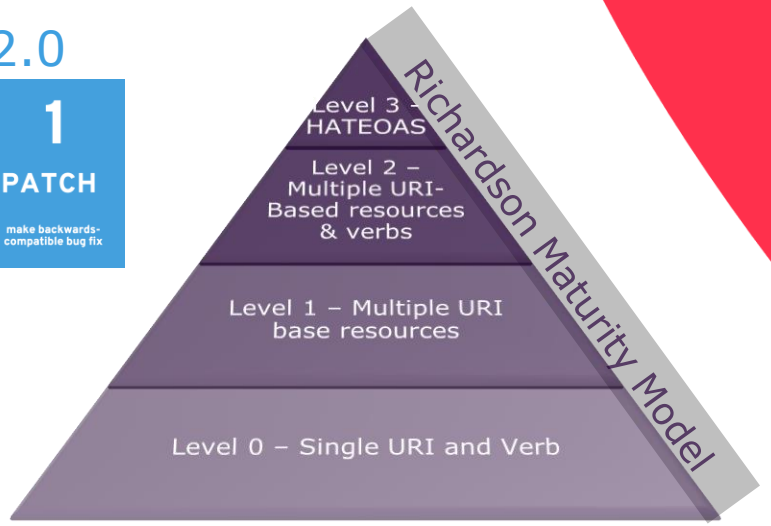


SemVer 2.0

1 . 3 . 1

MAJOR . MINOR . PATCH

incompatible API changes | add backwards-compatible functionality | make backwards-compatible bug fix



{Coding} Standards

http://example.com/myEntity?x=y&xEval=eq&a=b&aEval=gt

http://example.com/myEntity?filter="x equals y AND a > b"

REST APIs – Leverage HTTP/1

IETF®

ISO International Organization for Standardization

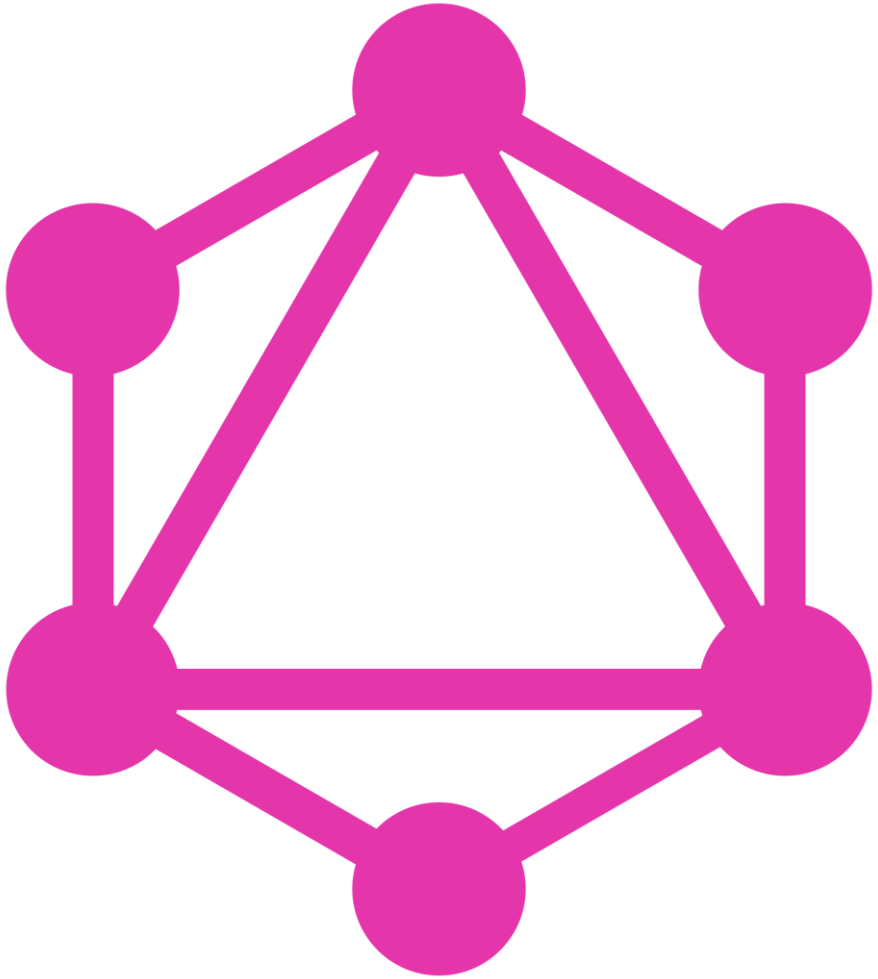
REST – Pros

- ✓ Established, well understood, well supported
- ✓ Doesn't have any requirements of the application server (e.g. HTTP/1 rather than HTTP/2)
- ✓ Use of URIs makes it easy to cache and ease workloads across different levels in the network (browser, CDN, reverse proxy)
- ✓ Self describing…
  - ✓ Makes easy for payload to be used by middleware
  - ✓ Easy Query content – so tolerant of content make up

REST – Cons

- Flexibility can lead to suboptimal APIs such as…
  - Tolerant of SQL in URIs
  - Poor documentation – making hard to understand
  - Strategies for evolution/versioning, not explicit
- Can be too terse in the data provided (multiple API calls needed) or too verbose (excessive information provided as standard)
- Overheads in the communication – conversion to/from String, no compression (other than HTTP prompts)

GraphQL



Useful Resources:

- Graphql.org
- www.graphqlbin.com
- www.apollographql.com/

GraphQL - Introduction

- Originally developed by Facebook to solve the problem of user experience and performance with their Mobile app
  - Facebook published the standard 2015
  - Since 2018 the standard is managed by the GraphQL Foundation as a subsidiary organization of the Linux Foundation
  - Includes open source reference implementations of tooling
- In addition to Facebook, strong adoption including big names such as:
  - GitHub
  - Yelp
  - Twitter
  - Instagram
- Leverages common standards including
  - HTTP
  - JSON
- The standard incorporates ideas around handling
  - Versioning
  - Pagination
- *Not* related to Graph Databases

GraphQL – Basic Query

```
interface Character
{ id: ID!
  name: String!
  friends: [Character]
  appearsIn: [String]!
}
```

```
Droid implements Character
{ id: ID!
  name: String!
  friends: [Character]
  appearsIn: [Episode]!
  primaryFunction: String
}
```

```
type Query {droid(id: ID!): Droid }
type Mutation {deleteDroid(id: ID!)
type Mutation (addDroid(newDroid: Droid!}
```

```
Query
{
  droid(id: "2000")
  {
    name,
    primaryFunction
  }
}
```

```
{
  "data":
  {
    "droid":
    {
      "name": "C-3PO“
      “primaryFunction”:
        “Interpreter”
    }
  }
}
```

- Schemas with strong typing
- Schemas can define multiple entities
- Schemas support the idea of abstraction through interfaces
- Different entities can be linked via common attributes

- Schemas can define different types of operations
  - Query → get
  - Mutations → insert / update / delete

- Operations can then be used
- Operations can define the attributes to use/retrieve

Based on: https://github.com/graphql/swapi-graphql

GraphQL – Query with Sub-Selections

```
type Episode
{ id: ID!
  title: String!
  openCrawl: String
  director: String!
}
```

```
Droid implements Character
{ id: ID!
  name: String!
  friends: [Character]
  appearsIn: [Episode]!
  primaryFunction: String
  hasFriends: Boolean =>
    @computed(value:friends != null)
}
```

- Types can be referenced by other types
- Define arrays of values,
- null & not null
- Default values and operations

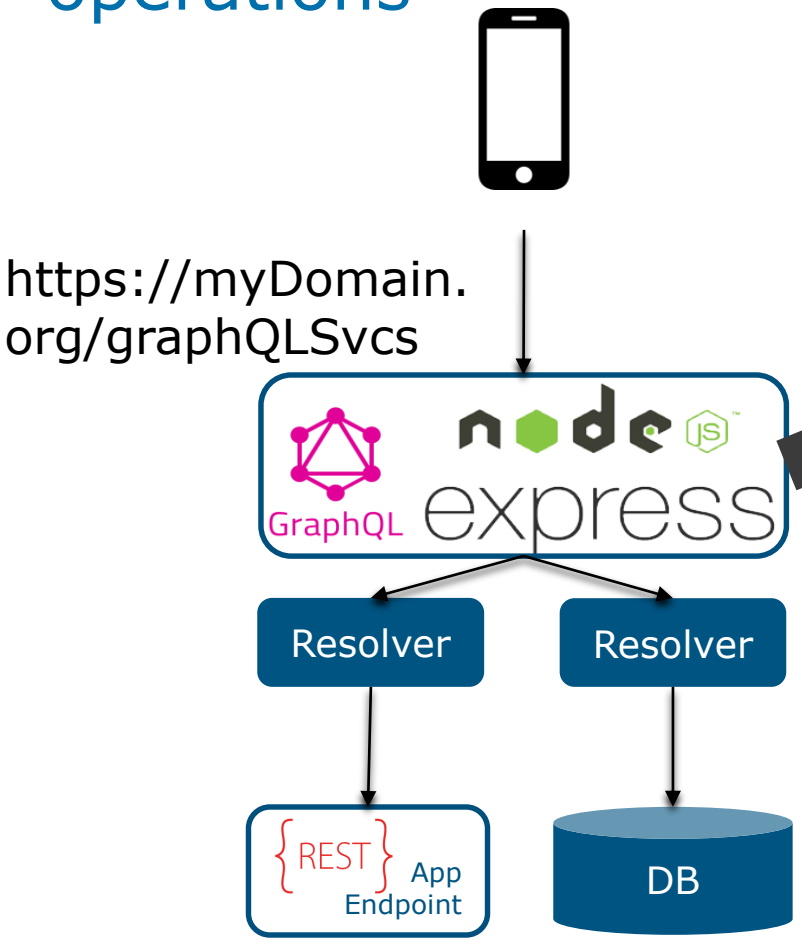
```
Query
{
  droid(id: "2000")
  {
    name,
    appearsIn
    {
      title
    }
  }
}
```

```
{
  "data":
  {
    "droid":
    {
      "name": "C-3PO“
      “appearsIn”:
       [
         {“title”:”A New Hope”},
         {“title”:”Revenge of the
Sith”},
    }
  }
}
```

- Can go further …
  - Expressions can define ‘join’ style operations
  - Arguments for each subtype, so could ask for droids
  - Typically use a client framework to make the query and traverse the response

Based on:https://github.com/graphql/swapi-graphql

Backend takes request can decompose into multiple operations




```
var express = require('express');
var graphqlHTTP = require('express-graphql');
var { buildSchema } = require('graphql');

// Construct a schema, using GraphQL schema language
var schema = buildSchema(/* load gql file */);

// The root provides a resolver function for each API endpoint
var root = {
  droid: (id) => {
    return {/* logic to handle droid requerst by Id */};
  },
  deleteDroid: (id) => {
    return {/* logic to handle droid deletion */};
  },
  addDroid: (droid) => {
    return {/* logic to handle droid addition*/};
  },

};

var app = express();
app.use('/graphql', graphqlHTTP({
  schema: schema,
  rootValue: root,
  graphiql: true,
}));
app.listen(4000);
console.log('Running a GraphQL API server at http://localhost:4000/graphql');
```

- the backend for handling a GraphQL operation is implemented using a server such as Apollo, Express with GraphQL extensions
- The backend executes the necessary operations using ‘resolvers’ to perform the necessary work

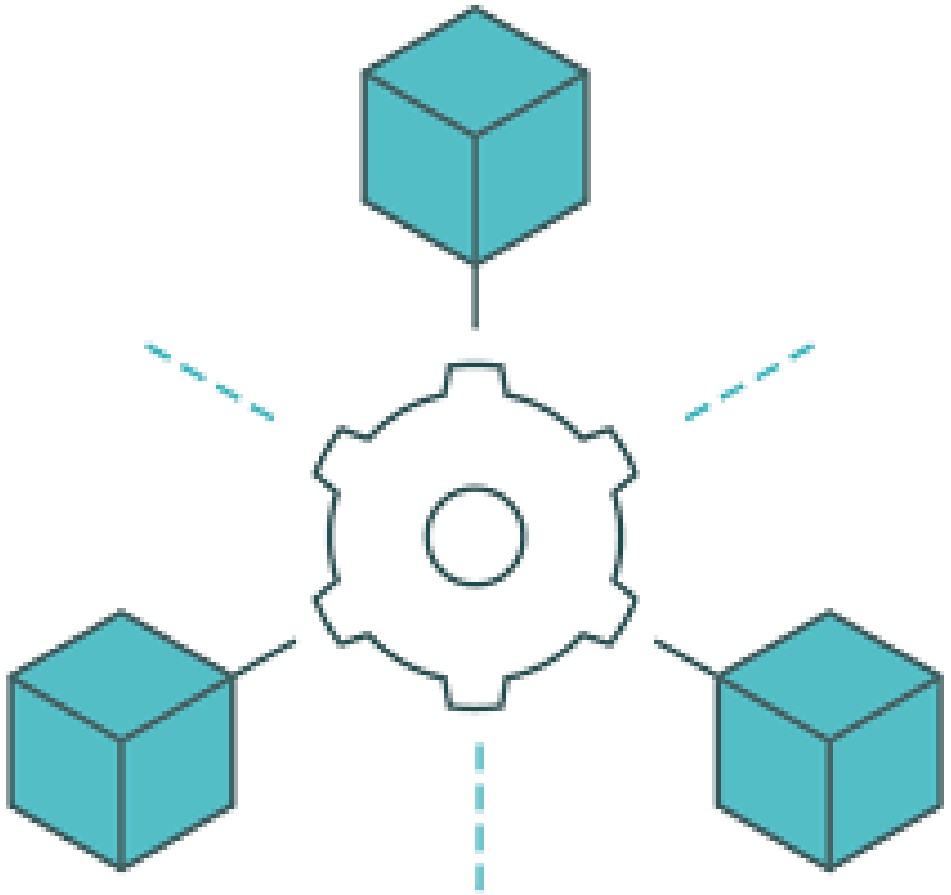
Based on:https://github.com/graphql/swapi-graphql

GraphQL – Pros

- ✓ Can define the data wanted – making API calls very efficient
- ✓ Ability to put workload into the backend blending data together
- ✓ Reduced round tripping of successive API calls
- ✓ Strongly structured & typed
- ✓ JSON based meaning…
  - ✓ Easy to work with
  - ✓ Human readable
  - ✓ Easy to formulate queries and responses (although libraries further simplify)
- ✓ Version managed greatly simplified (API enhancement unlikely to be disruptive)

GraphQL – Cons

- ✗ Doesn’t leverage HTTP to its maximum benefit
- ✗ Can’t exploit web based caching to reduce backend workload (URLs are always the same – differences are in the body)
- ✗ Simple requests do require more effort
- ✗ As notation and mechanics are closer to relational and object persistence, risk of reflecting underlying data models & being vulnerable to change

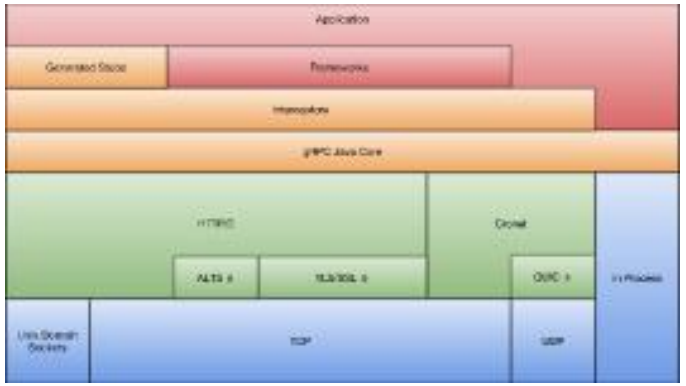


Useful Resources:

- grpc.io/
- developers.google.com/protocol-buffers/docs/javatutorial
- github.com/grpc

gRPC - Introduction

- Developed by Google (the original meaning of ‘g’ in gRPC (Google have played with the meaning of the g in different releases)
- Is open-source
- In addition to Google, big adopters include…
  - Netflix
  - Cisco
- Is technical RPC style with concepts including…
  - Services & messages rather than objects & references
  - Simplicity & lightweight – work with small devices
  - Interoperability over internet infrastructure
  - Layering to allow evolution
  - Payload & implementation agnostic
  - Standardized error codes



- 10/11 Programming Languages officially supported (C/C++/C#, Objective C, Dart, Go, Node, PHP, Java, Ruby, Dart, Python)
- Leverages …
  - HTTP/2 (based on Google’s SPEDY) – not HTTP. Common verb semantics but differences at the wire level. Impact on server support
- Supports ideas of …
  - Versioning & compatibility
- Some common concepts to Apache Avro

gRPC

```
syntax = "proto3";

message Id {
  uint32 id = 1;}

message Character {
   Id id =1;
  string name =2;
  repeated string
     appearsIn =3;
  repeated Character
     friends =4;
}

message Droid
{
  Id id = 1;
  string primaryFunction =2;
  Character character = 3;
}

message Ids {
  repeated uint32 id = 1;}
```

```
service GetService
{
  rpc getDroidById (Id) returns (Droid) {}
  rpc getCharacterById (Id) returns (Character) {}
  rpc getCharactersById (Ids) returns
    (stream Character) {}
}
```

- Uses a DDL called Protobuf to describe the messages
- 2 Versions of Protobuf (v2 & v3) – which aren't compatible
- Each part of the message has a position in the message structure
- Strongly typed – but typing maps to byte sizing
- Specific sizing & types enables techniques compress the representation
- Omitted values are replaced with default values in the payload

- Define a service (aka class)
- With operations – no explicit indication of immutability
- Run to protoc (protobuf compiler) on these message definitions to create language specific code

Based on: https://github.com/graphql/swapi-graphql

Generated code

```
option java_package = "AltSWAPI";
option go_package = "AltSWAPI";
option java_generic_services = true;
option java_multiple_files = true;
```

Add some metadata to the protobuf file or CLI

```
protoc --java_out=.
sample.proto
```

```
 * Protobuf service {@code MySWAPI.GetCharactersService}
 */
public  abstract class GetCharactersService
    implements com.google.protobuf.Service {
  protected GetCharactersService() {}

  public interface Interface {…

  public static com.google.protobuf.Service newReflectiveService(…

  public static com.google.protobuf.BlockingService
      newReflectiveBlockingService(final BlockingInterface impl) {…

  /**…
  public abstract void getDroidById(…

  /**…
  public abstract void getCharacterById(…

  /**…
  public abstract void getCharactersById(…

  public static final
      com.google.protobuf.Descriptors.ServiceDescriptor
      getDescriptor() {…
  public final com.google.protobuf.Descriptors.ServiceDescriptor
      getDescriptorForType() {…

  public final void callMethod(…

  public final com.google.protobuf.Message
      getRequestPrototype(…

  public final com.google.protobuf.Message
      getResponsePrototype(…

  public static Stub newStub(…

  public static final class Stub extends AltSWAPI.GetCharactersService implements Interface {…

  public static BlockingInterface newBlockingStub(…

  public interface BlockingInterface {…

  private static final class BlockingStub implements BlockingInterface {…

  // @@protoc_insertion_point(class_scope:MySWAPI.GetCharactersService)
}
```

```
447   @java.lang.Override
448   public Builder toBuilder() {…
```

Based on: https://github.com/graphql/swapi-graphql

gRPC – Pros

- ✓ Support for forward & backward compatibility, and notation actively encourages such consideration
- ✓ Very efficient use of network bandwidth (payload is compressed etc)
- ✓ Code generation means no need for SDKs as part of the process of adoption creates an SDK
- ✓ Encourages a design 1st methodology – rather than building messages from existing classes
- ✓ Binary compression – adds more effort for a hacker
- ✓ Multiple languages

gRPC – Cons

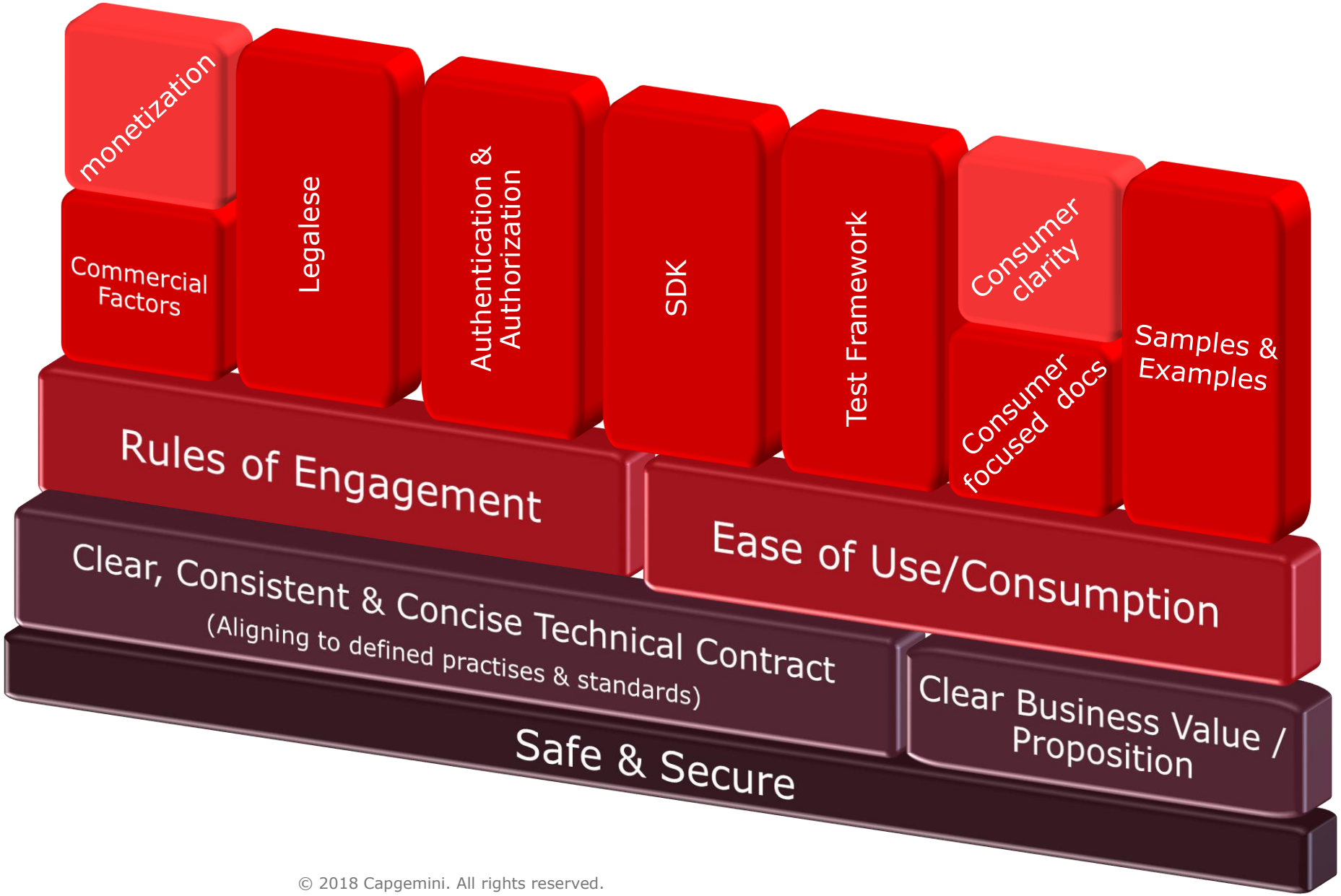
- ✗ Binary payload means it isn’t human readable – increasing effort for any intervention steps
- ✗ Can’t query the payload to find values – need to know in advance all the elements in the definition
- ✗ Mainstream languages supported, but special cases could be an issue without implementing your own Protobuf Compiler.
- ✗ Limits how service meshes can OOTB help
- ✗ Requires HTTP/2 conversant server and client
- ✗ Two versions – although it is possible to reference Proto2 definitions in a Proto3 message

Recommendations

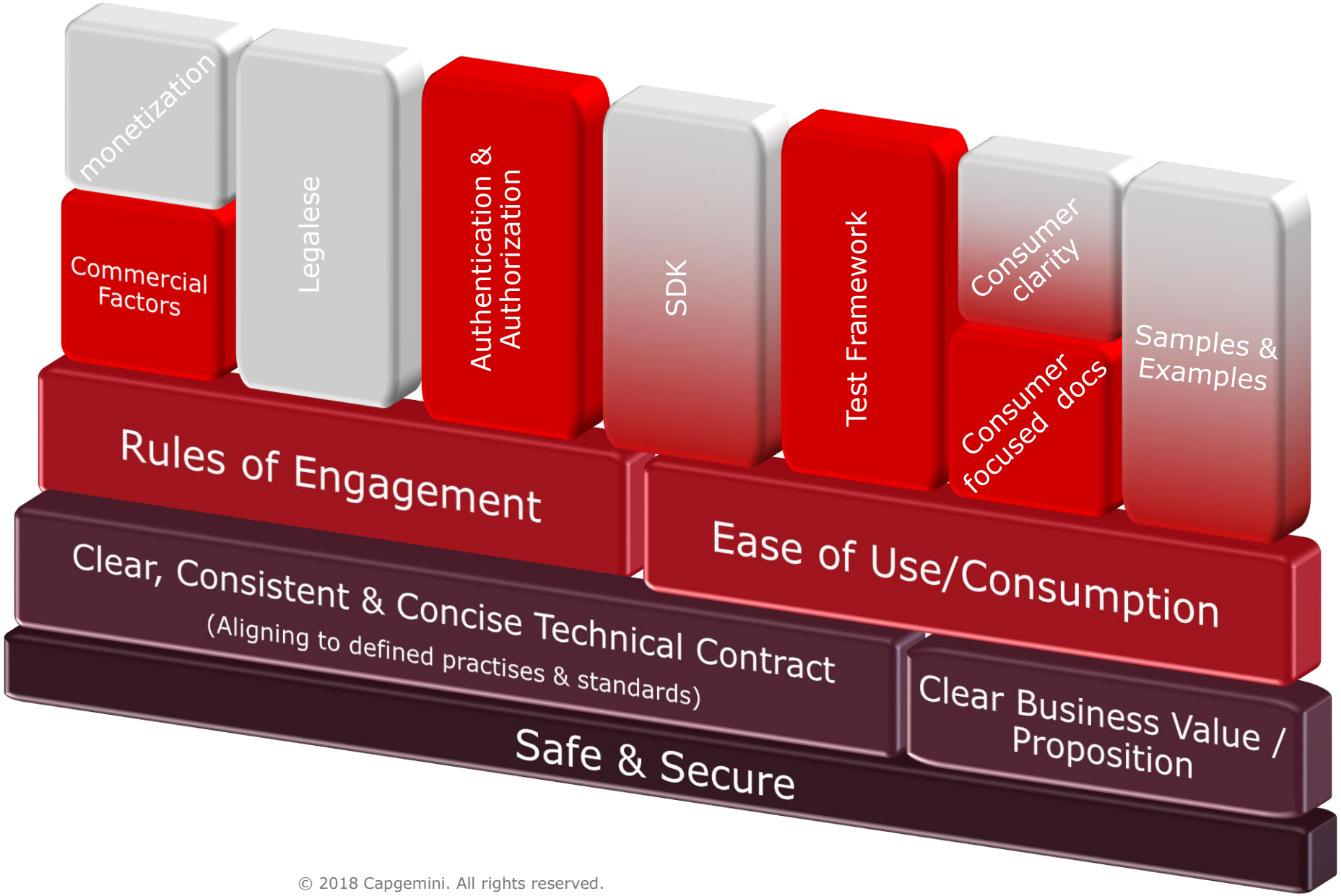
More important than the technicalities of an API …

- API addresses my needs
- API gives value to the consumer not the provider
- A Well Designed Understandable API – will just about always **WIN**
  - As a consumer – if an API is clear, with all the assets of a Good API – then most technicalities can be overcome

The Make-Up of a Good API – **Public Facing** Use Case



The Make-Up of a Good API – **Internal** Use Case



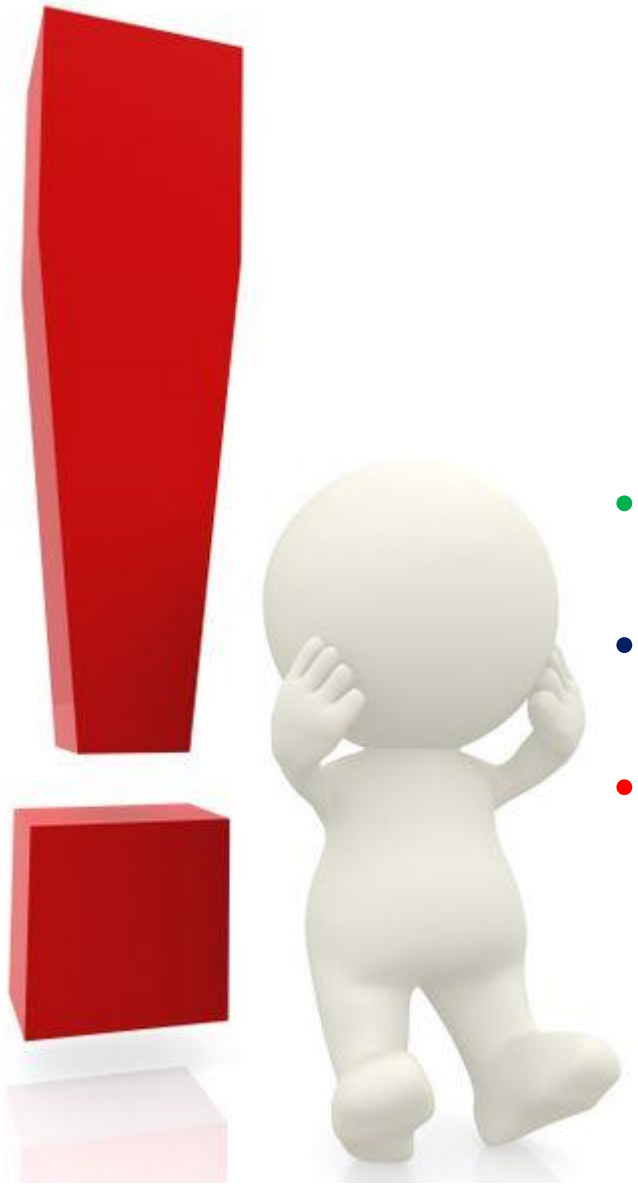
One way to consider it – to use Fielding's goals

Criteria	Rest	GraphQL	gRPC
Client / Server	Yes	Yes	Yes
Stateless	Yes	Yes	Yes
Cacheable	Yes	Possible	No
Layered	Yes	Possible	No
Code on Demand	Yes	No	No
Uniform Interface			
- Resource Identifiers	Yes	No	No
- Resource Representation	Yes	No	No
- Self Describing	Yes	Yes	No
- Hyptertext as an Engine of State	Yes	Possible	No

Recommendation

Use RESTful for …	Use GraphQL for …	Use gRPC for …
Public APIs – unknown target or system to system • represents lowest common denominator – so easiest to adopt • Consider strategies that allow consumers to choose data elements	Public APIs / APIs supporting mobile device or platforms with low bandwidth/high latency Make life easier providing SDKs	System to … • system flows where intermediaries don't need to understand the payload • e.g inter-microservice calls • client only when you have control of both client & server and don't need intermediaries
APIs where webhooks are the best way to deliver asynchronous data	Streaming API use cases	Bandwidth sensitive environments (exploit binary compression)
Internal APIs – system to system where processes may need to query payloads – e.g. ESB managed traffic	Internal APIs – system to system where processes may need to query payloads – e.g. ESB managed traffic	Streaming or bi-directional data flows are involved.
	Exposing data warehouses/lakes without encouraging bulk data movement	Small footprint clients – where clients have small resource footprints e.g. IoT devices (executable footprint is compact)

Conclusion



- ***Right tool for the right job***
- ***There is no single silver bullet***
- ***Have a kit bag, not 1 hammer***

People matter, results count.

About Capgemini

With more than 190,000 people, Capgemini is present in over 40 countries and celebrates its 50th Anniversary year in 2018. A global leader in consulting, technology and outsourcing services, the Group reported 2016 global revenues of EUR 12.5 billion. Together with its clients, Capgemini creates and delivers business, technology and digital solutions that fit their needs, enabling them to achieve innovation and competitiveness. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

Learn more about us at

www.capgemini.com