Ist GraphQL das bessere REST?

checkdomain

checkdomain

174 TSD	Aktive Domains
48 TSD	Aktive Kunden
22	Festangestellte Mitarbeiter

PHP, Symfony, NodeJS, React, MySQL, MongoDB, RabbitMQ,
DDD, Git, Gitlab CI, Ansible, Metrics

1

Einführung (5 min)
Zeitreise, Basics

2

REST unter der Lupe (10 min)
Probleme mit REST

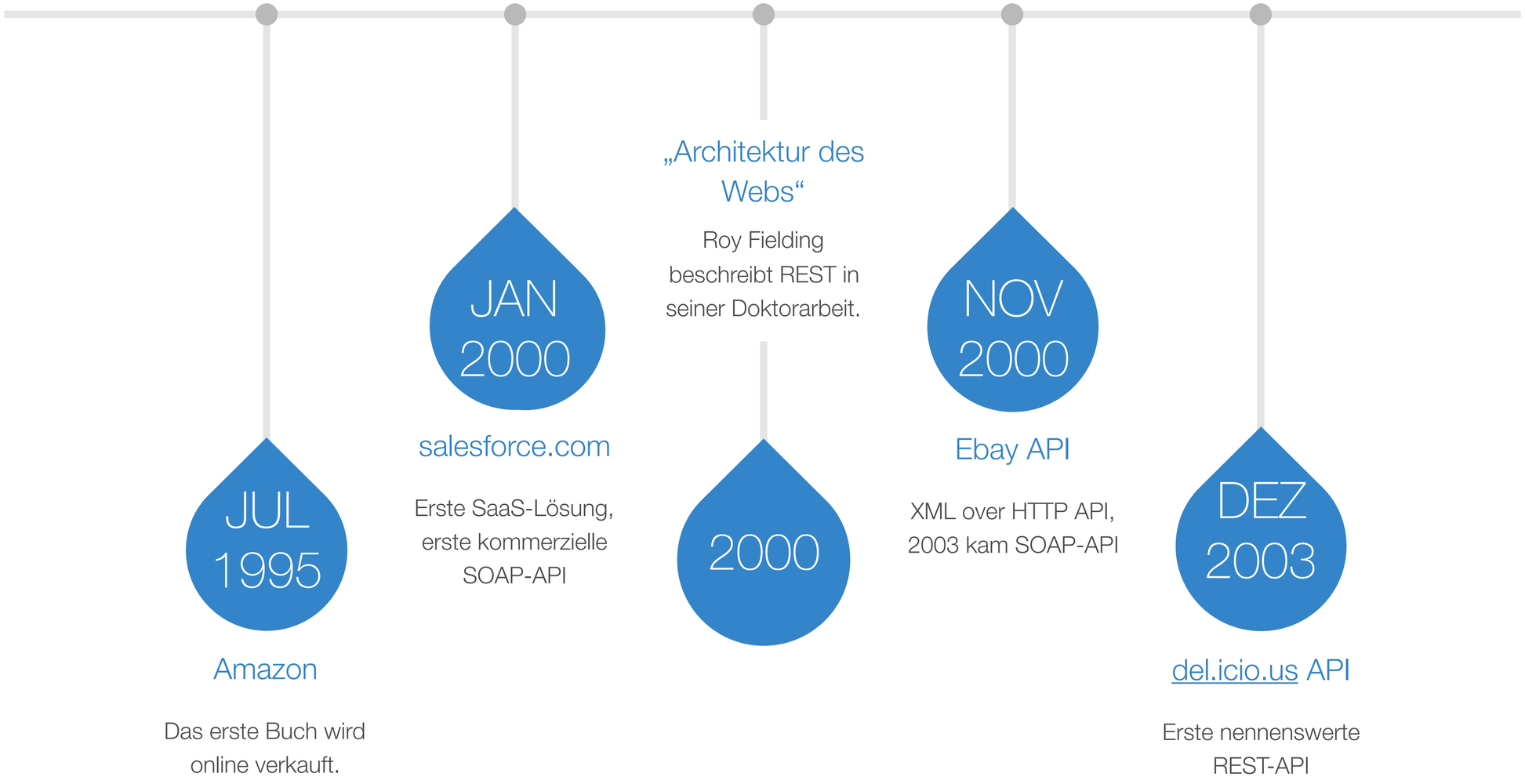
3

Alternative (15 min)
GraphQL

REST-Erfahrung

GraphQL-Erfahrung

Erste öffentliche API


JUL 1995
Amazon
Das erste Buch wird online verkauft.
JAN 2000
salesforce.com
Erste SaaS-Lösung, erste kommerzielle SOAP-API
„Architektur des Webs“
Roy Fielding beschreibt REST in seiner Doktorarbeit.
2000
NOV 2000
Ebay API
XML over HTTP API, 2003 kam SOAP-API
DEZ 2003
del.icio.us API
Erste nennenswerte REST-API

API-Konzepte (2000)

XML-RPC

SOAP

REST

XML-RPC

- Extensible Markup Language Remote Procedure Call
- 1998 von Dave Winer (Firma Userland)
- Simpel - 7 Seiten Spezifikation inkl. FAQ
- Transport: HTTP(S)
- Darstellung: XML
- http://xmlrpc.scripting.com/spec.html

```
POST /ws/api.dll HTTP/1.1
Host: ebay-kleinanzeigen.de
Content-Type: text/xml
Content-length: 180

<?xml version="1.0"?>
<methodCall>
  <methodName>auction.wasIsLetztePreis</methodName>
  <params>
    <param>
      <value><i4>41</i4></value>
    </param>
  </params>
</methodCall>

HTTP/1.1 200 OK
Connection: close
Content-Length: 146
Content-Type: text/xml

<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value><double>5.42</double></value>
    </param>
  </params>
</methodResponse>
```

SOAP

- Früher: Simple Object Access Protocol
- 1999 veröffentlicht (SOAP 0.9), Weiterentwicklung von XML-RPC
- XML-basiertes Nachrichtenprotokoll/-Standard, Konventionen für die Nachrichtenaufbau
- Beschreibt vielfältige Message-Pattern (z.B. RPC)
- Transport: agnostisch (TCP, HTTP, SMTP, JMS, etc.)
- Darstellung: XML
- Schnittstellenbeschreibung: WSDL —> Codegenerierung
- Enterprise: WS-Security (End-to-End-Verschlüsselung), ACID-Complicance/Retry-Logik etc.

Aufbau einer SOAP-Nachricht

```
<?xml version="1.0"?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
    <s:Header>
    </s:Header>
    <s:Body>
    </s:Body>
</s:Envelope>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
        xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <ns1:RequestHeader
          soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
          soapenv:mustUnderstand="0"
          xmlns:ns1="https://www.google.com/apis/ads/publisher/v201802">
      <ns1:networkCode>123456</ns1:networkCode>
      <ns1:applicationName>DfpApi-Java-2.1.0-dfp_test</ns1:applicationName>
    </ns1:RequestHeader>
  </soapenv:Header>
  <soapenv:Body>
    <getAdUnitsByStatement xmlns="https://www.google.com/apis/ads/publisher/v201802">
      <filterStatement>
        <query>WHERE parentId IS NULL LIMIT 500</query>
      </filterStatement>
    </getAdUnitsByStatement>
  </soapenv:Body>
</soapenv:Envelope>
```

DoubleClick Publisher API: getAdUnitsByStatement() looking for the root AdUnit
https://developers.google.com/doubleclick-publishers/docs/soap_xml

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <ResponseHeader xmlns="https://www.google.com/apis/ads/publisher/v201802">
      <requestId>xxxxxxxxxxxxxxxxxxxx</requestId>
      <responseTime>1063</responseTime>
    </ResponseHeader>
  </soap:Header>
  <soap:Body>
    <getAdUnitsByStatementResponse xmlns="https://www.google.com/apis/ads/publisher/v201802">
      <rval>
        <totalResultSetSize>1</totalResultSetSize>
        <startIndex>0</startIndex>
        <results>
          <id>2372</id>
          <name>RootAdUnit</name>
          [..]
          <inheritedAdSenseSettings>
            <value>
              <adSenseEnabled>true</adSenseEnabled>
              <borderColor>FFFFFF</borderColor>
              [..]
            </value>
          </inheritedAdSenseSettings>
        </results>
      </rval>
    </getAdUnitsByStatementResponse>
  </soap:Body>
</soap:Envelope>
```


DoubleClick Publisher API: getAdUnitsByStatement() looking for the root AdUnit
https://developers.google.com/doubleclick-publishers/docs/soap_xml

Representational State Transfer

- Architekturkonzept, kein Standard
- 2000 Roy Fielding
- Macht sich vorhandene www-Standards zu Nutze (HTTP/URI)
- Ressourcen-orientiert: Anwendungszustand und Funktionalität repräsentiert durch eindeutige URI
- Aktionen: HTTP-Verben (GET, POST, PUT, DELETE, HEAD, OPTIONS, CRUD)
- Transport: HTTP(S)
- Darstellung: JSON, XML, HTML, CSV, etc.

HTTP-Verben

Verb	Bedeutung	Idemp.	Cache
GET /users/1	Lesen einer Ressource	Ja	Ja
POST /users	Anlegen einer Ressource oder auslösen eines Datenverarbeitungsprozesses	Nein	Nein
PUT /users/1	Vollständiges Update einer Ressource	Ja	Nein
PATCH /users/1	Partielles Update einer Ressource	Nein	Nein
DELETE /users/1	Löschen einer Ressource	Ja	Nein
HEAD /users/1	Metadaten einer Ressource	Ja	Ja
OPTIONS /users/1	Mögliche Aktionen einer Ressource abfragen	Ja	Ja

HTTP-Status

Code	Bedeutung
200	OK
201	Ressource created
202	Accepted, processing not completed
204	No content
400	Bad request
401	Unauthorized
404	Not found
405	Method not allowed
500	Internal server error

```
PUT /spaces/yadj1kx9rmg0/environments/staging/entries/5KsDBWseXY6QegucYAoacS/published
Authorization: Bearer 0bee89b07a248e27c83fc3d5951213c1
X-Contentful-Version: 6

HTTP/1.1 200 OK
Connection: close
Content-Type: application/json

{
  "fields": {
    "title": {
      "en-US": "Hello, World!"
    },
    "body": {
      "en-US": "Bacon is healthy!"
    }
  },
  "sys": {
    "firstPublishedAt": "2015-05-15T13:38:11.311Z",
    "publishedCounter": 2,
    "publishedAt": "2015-05-15T13:38:11.311Z",
    "publishedBy": {
      "sys": {
        "type": "Link",
        "linkType": "User",
        "id": "4FLrUHftHW3v2BLi9fzfjU"
      }
    },
    "publishedVersion": 9
  }
}
```

Contentful API
https://www.contentful.com/developers/docs/references/content-management-api/#/reference/entries/entry-publishing/publish-an-entry/console

SPARKPOST

TRY FREE

cURL Nodejs Python PHP Java C# Go Elixir

```
curl -XPOST \
  https://api.sparkpost.com/api/v1/transmissions \
  -H "Authorization: <YOUR API KEY>" \
  -H "Content-Type: application/json" \
  -d '{"options": {"sandbox": true}, "content": {"from": "testing@sparkpostbox.com",
 "subject": "Oh hey", "text":"Testing SparkPost - the most awesomest email service in
 the world"}, "recipients": [{"address": "developers+curl@sparkpost.com"}]}'
```

https://www.sparkpost.com/

Vergleich (2000)

XML-RPC

+ Standard
+ Sehr Simpel

- Hohes Übertragungsvolumen durch XML
- Fester Transport (HTTP)

SOAP

+ Standard
+ Transport-agnostisch
+ Type checking/Contract (WSDL/XSD)
+ Enterprise-Features: ACID-Complicance, WS-Security

- Komplexe Architektur
- Keine Sensibilität für Remote Aufrufe (8 fallacies)
- Hohes Übertragungsvolumen

REST

+ Simpel
+ Viele Austauschformate (Menschen-lesbar)
+ Verwendung von HTTP-Tools (Caches, Analyse etc.)
+ Übertragungsvolumen um bis zu Faktor 10 geringer als bei SOAP
+ Lose Kopplung
+ Link-Integration

- folgt…

Google Trends | Erkunden

rest api
Suchbegriff

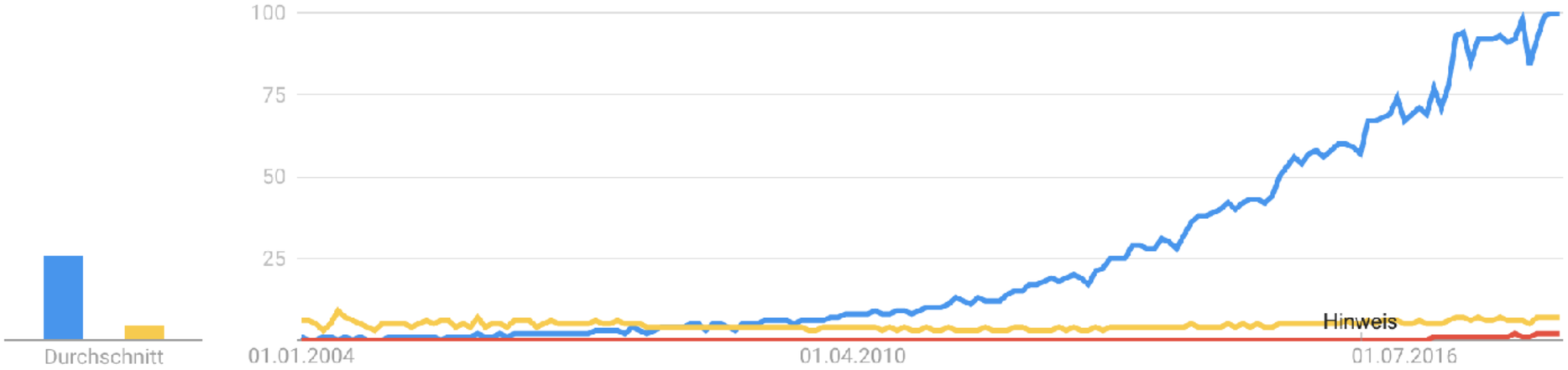
graphql api
Suchbegriff

soap api
Suchbegriff

+

Weltweit | 2004 - heute | Alle Kategorien | Websuche

Interesse im zeitlichen Verlauf



1

Einführung (5 min)
Kennenlernen, Zurück ins Jahr 2000, Basics

2

REST unter der Lupe (15 min)
Probleme mit REST

3

Alternativen (15 min)
GraphQL, JSON-RPC.

Mi Mi Mi

RESTish statt REST

- Richardson Maturity Model
  - Level 0: HTTP-Transport (XML-RPC/SOAP)
  - Level 1: Ressourcen
    Divide & conquer
  - Level 2: HTTP-Verben (RESTish)
    Voraussagbares Verhalten
  - Level 3: Hypermedia (REST)
    Discoverability, Selbstdokumentation
- Meiste vermeintliche REST-APIs nur Level 2

Expand groups | Collapse groups

Admins

Archive

Domains

/api/archive/getusedspace/[domain/<string>/][admin_id/<int>/]	[example]
/api/domain/add/domain/<string>/[destinations/<array>/][aliases/<array>/]	[example]
/api/domain/addlocalrecipient/domain/<string>/recipient/<string or array>/	[example]
/api/domain/archive/domain/<string>/action/<string>/	[example]
/api/domain/blacklistrecipient/domain/<string>/recipient/<string>/	[example]
/api/domain/blacklistsender/domain/<string>/sender/<string>/	[example]
/api/domain/edit/[domain/<string>/]destinations/<array>/	[example]
/api/domain/exists/[domain/<string>/]	[example]
/api/domain/filtering/domain/<string>/filtering/<string>/	[example]
/api/domain/getbandwidthusage/domain/<string>/since/<int>/until/<string>/	[example]
/api/domain/getfreelimit/destination_username/<string>/	[example]
/api/domain/gethardquota/domain/<string>/	[example]
/api/domain/getowner/domain/<string or array>/	[example]
/api/domain/getproducts/domain/<string>/	[example]
/api/domain/getroute/[domain/<string>/]	[example]
/api/domain/getsoftquota/domain/<string>/	[example]
/api/domain/getstorage/domain/<string>/	[example]
/api/domain/getvalidrecipientcount/domain/<string>/	[example]
/api/domain/move/domains/<string or array>/destination_username/<string>/[destination_password/<string>/]	[example]
/api/domain/recipientblacklist/domain/<string>/	[example]
/api/domain/recipientwhitelist/domain/<string>/	[example]
/api/domain/remove/domain/<string>/	[example]
/api/domain/removelocalrecipient/domain/<string>/recipient/<string or array>/	[example]
/api/domain/senderblacklist/domain/<string>/	[example]
/api/domain/senderwhitelist/domain/<string>/	[example]
/api/domain/setdays/domain/<string>/days/<int>/	[example]
/api/domain/sethardquota/domain/<string>/quota/<int or string>/	[example]

https://api.antispamcloud.com/api/help.php

Response-Codes

- 41 Response-Codes (15 Seiten) in HTTP/1.1-Spezifikation
  https://tools.ietf.org/html/rfc7231#page-62
- Individuelle Interpretationen durch API-Anbieter
- Update einer Ressource:
  `200 OK` oder `201 Created`? Kein `2XX Updated` vorhanden.
- In Browsern verschiedene Interpretationen von Response-Codes:
  - z.B. `307 Temporary redirect`)
  - In Browsern lediglich 200 und 500 konsistent interpretiert.

417 - Expectation failed?

RFC 7231 HTTP/1.1 Semantics and Content June 2014

6.5.14. 417 Expectation Failed

The 417 (Expectation Failed) status code indicates that the expectation given in the request's Expect header field (Section 5.1.1) could not be met by at least one of the inbound servers.

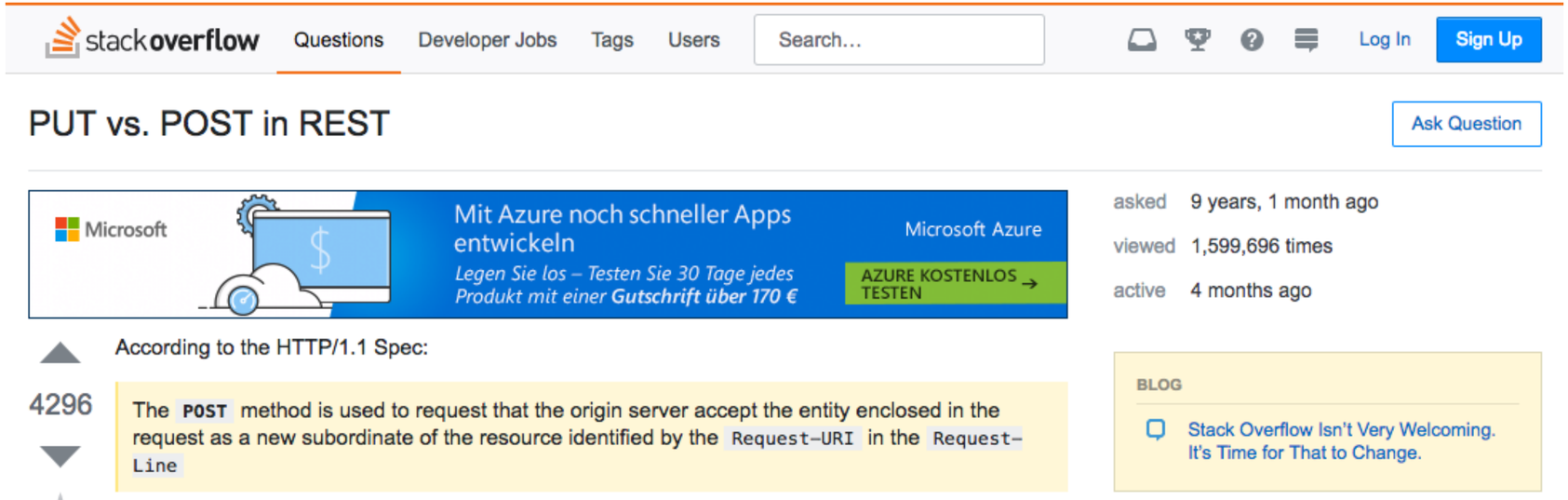
https://tools.ietf.org/html/rfc7231#page-62

Standards

Qual der Wahl:

- Microsoft API Guidelines
  https://github.com/Microsoft/api-guidelines/blob/vNext/Guidelines.md

- White House Web API Standards & weitere
  https://github.com/WhiteHouse/api-standards



IDL / DDL

- Interface Description Languages/ API Description Languages
  - swagger.io
  - apiblueprint.org
  - raml.org
  - avro.apache.org
- Data Description Languages
  - jsonapi.org
  - json-schema.org
  - github.com/kevinswiber/siren
  - jsonpatch.com (Payload für PATCH-Requests)
  - avro.apache.org

API-Design
There’s more than one way

Mehr Diskussionen über REST als über Problemdomäne

- Qual der Wahl
  - Welchen Standard
  - REST/HATEOAS oder Pragmatic REST
  - Description-Languages: Swagger, JSON Schema & Co
  - Kleinigkeiten: z.B. Query-String
- Pro Aktion
  - Status-Codes
  - Verben
  - Sub-Ressourcen

Beispiel

Wie lösche ich eine registrierte Domain zum Laufzeitende?

```
POST /users/1/domains
{
    name: register.me
}

HTTP/1.1 201 Created
{
    id: 1,
    name: register.me,
    status: active,
    autorenewal: true,
    created: 2018-05-03,
    contractPeriodEnd: 2019-05-02,
    user: /users/1
}
```

```
DELETE /users/1/domains/1
202 Accepted
{ id: 1, [..] }
```

```
DELETE /users/1/domains/1/autorenewal
204 No Content
```

```
PATCH /users/1/domains/1/
{ op: replace, path: /autorenewal, value: false }
200 OK
```

Wie storniere ich nun diese Löschung?

Fehlersuche an 7 Stellen

1. Request Method
2. Request URI/Header
   z.B. Query-Parameter, Content-Type, Trailing-Slash
3. Request Payload
4. Überschriebene Request Method
   Mangels Client-Support
5. Response Code
6. Überschriebener Status-Code
   Mangels Abdeckung über HTTP Status-Codes
7. Response Payload

n+1 Problem

Story: *Als Kunde benötige ich eine Artikelliste mit Preisen, damit [..]*

```
HTTP/1.1 200 OK
Content-Type: application/json
[
    {
        id: 1,
        name: '.de-Domain',
        link: '/articles/1'
    },
    {
        id: 2,
        name: '.com-Domain',
        link: '/articles/2'
    },
]
```

```
GET /articles/1 HTTP/1.1
Accept: application/json

HTTP/1.1 200 OK
Content-Type: application/json
{
    id: 1,
    name: '.de-Domain',
    description: '...',
    price: 9.99,
}
```

Für Abfrage der Daten sind n+1 Queries notwendig

2018 = 18 Jahre später

1

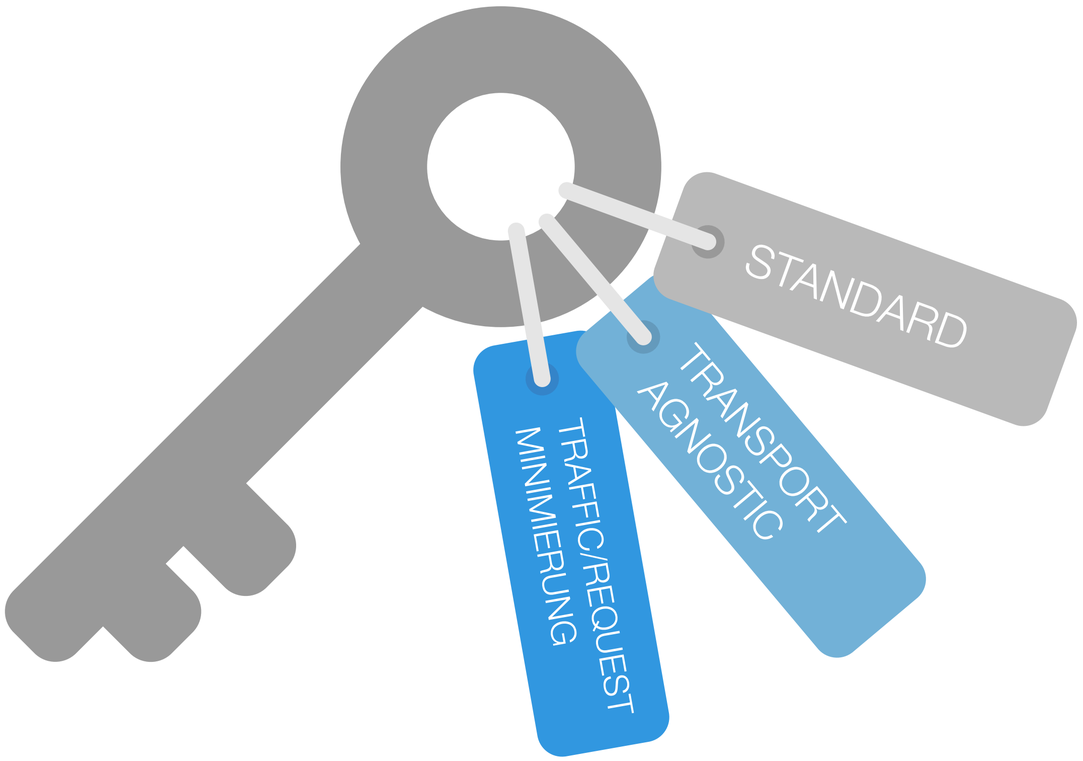
Einführung (5 min)
Kennenlernen, Geschichte, Basics

2

REST unter der Lupe (15 min)
Vor- und Nachteile von REST

3

Alternativen (15 min)
GraphQL, JSON-RPC…



01

Anzahl der nötigen Anfragen reduzieren, nur das liefern, was auch wirklich benötigt wird.

02

In Zeiten von Microservices und Queues sollte HTTP nicht das einzige Protokoll sein.

03

Viele Köcher verderben den Brei. Wir sind reif für einen neunen Standard.

2018!

- Anforderung ändern sich schneller
- Vielfältige Clients (Smartphone, Smart Home, Smart City, IoT, [..])
- Mobile Nutzung (2016 erstmals mehr Nutzer mit Mobilgeräten online als mit Desktop)
- Anforderungen Mobile/IoT/Microservice-Architekturen
  - Effiziente Bandbreitennutzung
  - Energieeffizienz

GraphQL

GraphQL was our opportunity to rethink mobile app data-fetching from the perspective of product designers and developers. It moved the focus of development to the client apps, where designers and developers spend their time and attention. GraphQL, a data query language.

Lee Byron

GraphQL

- Von Facebook in 2012 intern entwickelt, 2015 öffentlich
- Single Endpoint (vgl. XML-RPC/SOAP)
- Lesen: **Queries**
- Schreiben: **Mutations**
- Ein Request für alle Daten, die ein Client benötigt
- Validation by Design
- Integrierte SDL/Dokumentation

Queries

What you want is what you get

- Es werden nur die Daten ausgeliefert, die der Client anfragt
- Dadurch reduziertes Datenvolumen
- Reduzierung von Anfragen durch die Möglichkeit verknüpfte Ressourcen mit abzufragen
- Reduzierung von Anfragen durch Zusammenfassung von Anfragen/Daten aus mehreren Quellen
- Anfragen beginnen bei einen Start-Element
- Im Root Query-Object ist eine Liste aller Start-Elemente enthalten

Spezifische Felder

GitHub GraphQL API

Signed in as **mabrahamde**. You're ready to explore! Sign out

Heads up! GitHub's GraphQL Explorer makes use of your **real, live, production data.**

GraphiQL ▶ Prettify History

```
1 query {
2   viewer {
3     id
4     createdAt
5     login
6   }
7 }
8
```

QUERY VARIABLES

```
1 {}
```

```
{
  "data": {
    "viewer": {
      "id": "[redacted]",
      "createdAt": "2012-01-03T20:22:50Z",
      "login": "mabrahamde"
    }
  }
}
```

Schema **Query**

Lookup a repository owner (ie. either a User or an Organization) by login.

resource(url: URI!): UniformResourceLocatable
Lookup resource by a URL.

search(
  first: Int
  after: String
  last: Int
  before: String
  query: String!
  type: SearchType!
): SearchResultItemConnection!
Perform a search across resources.

topic(name: String!): Topic
Look up a topic by name.

user(login: String!): User
Lookup a user by login.

viewer: User!
The currently authenticated user.

https://developer.github.com/v4/explorer/

Verknüpfte Ressourcen

GitHub GraphQL API

Signed in as **mabrahamde**. You're ready to explore! Sign out

Heads up! GitHub's GraphQL Explorer makes use of your **real, live, production data.**

GraphiQL ▶ Prettify History

```
1  query {
2    viewer {
3      id
4      createdAt
5      login
6      following(first: 50) {
7        edges {
8          node {
9            login
10         }
11       }
12     }
13   }
14 }
15
```

QUERY VARIABLES

```
1  {}
```

```
{
  "data": {
    "viewer": {
      "id": "[redacted]",
      "createdAt": "2012-01-03T20:22:50Z",
      "login": "mabrahamde",
      "following": {
        "edges": [
          {
            "node": {
              "login": "ezzatron"
            }
          },
          {
            "node": {
              "login": "thebennos"
            }
          },
          {
            "node": {
              "login": "BenjaminPaap"
            }
          },
          {
            "node": {
              "login":
```

Schema **Query** ×

Lookup a repository owner (ie. either a User or an Organization) by login.

resource(url: URI!): UniformResourceLocatable
Lookup resource by a URL.

search(
  first: Int
  after: String
  last: Int
  before: String
  query: String!
  type: SearchType!
): SearchResultItemConnection!
Perform a search across resources.

topic(name: String!): Topic
Look up a topic by name.

user(login: String!): User
Lookup a user by login.

viewer: User!
The currently authenticated user.

https://developer.github.com/v4/explorer/

Argumente

GitHub GraphQL API

Signed in as **mabrahamde**. You're ready to explore! Sign out

Heads up! GitHub's GraphQL Explorer makes use of your **real, live, production data.**

GraphiQL ▶ Prettify History

```
1  query {
2    viewer {
3      id
4      createdAt
5      login
6      following(first: 50) {
7        edges {
8          node {
9            login
10         }
11       }
12     }
13   }
14 }
15
```

QUERY VARIABLES

```
1  {}
```

```
{
  "data": {
    "viewer": {
      "id": "[redacted]",
      "createdAt": "2012-01-03T20:22:50Z",
      "login": "mabrahamde",
      "following": {
        "edges": [
          {
            "node": {
              "login": "ezzatron"
            }
          },
          {
            "node": {
              "login": "thebennos"
            }
          },
          {
            "node": {
              "login": "BenjaminPaap"
            }
          },
          {
            "node": {
              "login":
```

Schema **Query**

Lookup a repository owner (ie. either a User or an Organization) by login.

resource(url: URI!): UniformResourceLocatable
Lookup resource by a URL.

search(
  first: Int
  after: String
  last: Int
  before: String
  query: String!
  type: SearchType!
): SearchResultItemConnection!
Perform a search across resources.

topic(name: String!): Topic
Look up a topic by name.

user(login: String!): User
Lookup a user by login.

viewer: User!
The currently authenticated user.

https://developer.github.com/v4/explorer/

Alias

GitHub GraphQL API

Signed in as **mabrahamde**. You're ready to explore! Sign out

Heads up! GitHub's GraphQL Explorer makes use of your **real, live, production data.**

GraphiQL ▶ Prettify History < Docs

```
1  query {
2      martin:
3        user(login: "mabrahamde") { starredRepositories(
4            first: 1,
5            orderBy: {field: STARRED_AT, direction: DESC}
6          ) {
7            edges { node {
8              name
9              url
10           }}
11         }
12       }
13     benny:
14       user(login: "benjaminpaap") { starredRepositories(
15           first: 1,
16           orderBy: {field: STARRED_AT, direction: DESC}
17         ) {
18           edges { node {
19             name
20             url
21           }}
22         }
23       }
24 }
25
```

QUERY VARIABLES

```
{
  "data": {
    "martin": {
      "starredRepositories": {
        "edges": [
          {
            "node": {
              "name": "tabler",
              "url": "https://github.com/tabler/tabler"
            }
          }
        ]
      }
    },
    "benny": {
      "starredRepositories": {
        "edges": [
          {
            "node": {
              "name": "avatars.dicebear.com",
              "url":
"https://github.com/DiceBear/avatars.dicebear.com"
            }
          }
        ]
      }
    }
  }
```

https://developer.github.com/v4/explorer/

Alias 2

GitHub GraphQL API

Signed in as **mabrahamde**. You're ready to explore! Sign out

Heads up! GitHub's GraphQL Explorer makes use of your **real, live, production data.**

GraphiQL ▶ Prettify History < Docs

```
1  query { user(login: "mabrahamde") {
2    login
3    following(first: 1) {
4      totalCount
5      edges {
6        node {
7          login
8          lastStarred: starredRepositories(
9            first: 1,
10           orderBy: {field: STARRED_AT, direction: DESC}
11         ) {
12           edges { starredAt node { name } }
13         }
14         firstStarred: starredRepositories(
15           first: 1,
16           orderBy: {field: STARRED_AT, direction: ASC}
17         ) {
18           edges { starredAt node { name } }
19         }
20 }}}}}
```

QUERY VARIABLES

```
{
  "data": {
    "user": {
      "login": "mabrahamde",
      "following": {
        "totalCount": 6,
        "edges": [
          {
            "node": {
              "login": "ezzatron",
              "lastStarred": {
                "edges": [
                  {
                    "starredAt": "2018-02-16T11:28:45Z",
                    "node": {
                      "name": "pixi.js"
                    }
                  }
                ]
              },
              "firstStarred": {
                "edges": [
                  {
                    "starredAt": "2009-06-30T00:11:11Z",
                    "node": {
                      "name": "swiftmailer"
                    }
                  }
```

https://developer.github.com/v4/explorer/

Fragmente

GitHub GraphQL API

Signed in as **mabrahamde**. You're ready to explore! Sign out

Heads up! GitHub's GraphQL Explorer makes use of your **real, live, production data.**

GraphiQL ▶ Prettify History < Docs

```
fragment repoStats on Repository {
  name description url
  stargazers { totalCount }
  forks { totalCount }
  watchers { totalCount }
}

query {
  user(login: "mabrahamde") {
    login
    following(first: 50) {
      edges { node {
        login
        starredRepositories(
          first: 10,
          orderBy: {field: STARRED_AT, direction: DESC}) {
            edges {
              starredAt node { ...repoStats }
            }
          }
        }
      }
    }
  }
}
```

QUERY VARIABLES


```
{
  "data": {
    "user": {
      "login": "mabrahamde",
      "following": {
        "edges": [
          {
            "node": {
              "login": "ezzatron",
              "starredRepositories": {
                "edges": [
                  {
                    "starredAt": "2018-02-16T11:28:45Z",
                    "node": {
                      "name": "pixi.js",
                      "description": "The HTML5 Creation
Engine: Create beautiful digital content with the fastest,
most flexible 2D WebGL renderer.",
                      "url":
"https://github.com/pixijs/pixi.js",
                      "stargazers": {
                        "totalCount": 18616
                      },
                      "forks": {
                        "totalCount": 2814
                      },
                      "watchers": {
                        "totalCount": 851
```


https://developer.github.com/v4/explorer/

[..]

- Aktionsnamen
- Variablen
- Default-Variablen
- Direktiven

https://developer.github.com/v4/explorer/

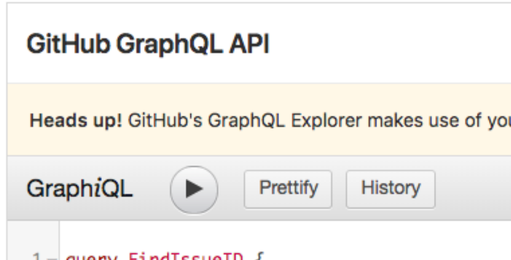
Mutations

Änderung

- Explizit unabhängig von Abfragen
- Schreiboperationen sollen nur über Mutations durchgeführt werden
- Aufbau
  - *mutationName*: Welche Änderung soll durchgeführt werden?
  - *input*: Für Änderung notwendige Daten
  - *payload:* Rückgabe vom Server, z.B. geändertes Objekt

```
mutation {
  mutationName(input: {MutationNameInput!}) {
    MutationNamePayload
}
```

https://developer.github.com/v4/guides/forming-calls/#about-mutations



Änderung: Beispiel

GitHub GraphQL API

Signed in as **mabrahamde**. You're ready to explore! Sign out

Heads up! GitHub's GraphQL Explorer makes use of your **real, live, production data.**

GraphiQL ▶ Prettify History

```
1  mutation {
2    addReaction(input:{
3      subjectId:"ABC",
4      content:HOORAY
5    }) {
6      reaction {
7        content
8      }
9      subject {
10       id
11     }
12   }
13 }
```

QUERY VARIABLES

Schema **addReaction** ×

Adds a reaction to a subject.

TYPE

AddReactionPayload

ARGUMENTS

input: AddReactionInput!

https://developer.github.com/v4/reference/mutation/addreaction/

Demo

https://developer.github.com/v4/explorer/

- + Reduzierung von Anfragen (kein n+1 Problem)
- + Kein Over-Fetching
- + Validation by Design
- + Integrierte SDL
- + Dokumentation
- + Anfragen kombiniert in einem Request
  - + Query Planning/ Optimierung
  - + Parallel Execution
- + Request Budgeting (z.B. Sekunden)
- + Error-Handling einfacher (aber per Default nicht maschinenlesbar
- + Für einfache Anwendung BaaS (z.B. graph.cool)

- - Error-Handling nur in Anwendungsschicht
- - Lesen und Schreiben unabhängig, Bezug fehlt (s. Ressource bei REST)
- - Kein Caching über HTTP möglich
- - Schlechte Predictability (z.B. wie lösche ich etwas)
- - Absicherung der Infrastruktur vor Risiken komplexer Anfragen
- - Rate Limits und Abrechnungsmodell muss ggf. neu
- - Select * komplex

Fragen

AND, OR, Filterung

- GraphQL ist näher an RPC als an einer Datenbank-Abfragesprache.
- GraphQL sollte die Problemdomäne und nicht die Persistenzschicht modellieren. (Outside-In)
- Für jeden Knoten kann über Argumente spezifische Filter und Abfrage-Funktionalität realisiert werden.

https://github.com/graphql/graphql-js/issues/585#issuecomment-262402544

Response-Struktur

- Struktur der Antworten durch Schema vorgegeben und nicht änderbar, außer Aliases

Microservices & GraphQL

- tbd

Danke

Martin Abraham

Web: https://www.checkdomain.de/
Email: m.abraham@checkdomain.de
Twitter: https://twitter.com/mabrahamde
Xing: https://www.xing.com/martin-abraham
Github: https://github.com/mabrahamde
Slideshare: https://www.slideshare.net/MartinAbraham9