BEVOR ES LOSGEHT . . .

NPM & Node.js installiert?

Code Editor mit Syntax-Highlighting?

Laptop mit aktuellem Browser?

Projekt herunterladen

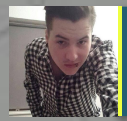
npm install

npm start

localhost:3000

localhost:3000/voyager

GraphQL Grundlagen



Dennis Dubbert
@ddubbert

Vimal Darius Seetohul
@vseetohu

Workshop 1

Workshop 2

01 GRAPHQL GRUNDLAGEN

GraphQL Grundpfeiler
Vergleich zu REST

02 GRAPHQL CLIENT

GraphQL Konzepte
Voyager
Playground
Queries

03 GRAPHQL SERVER

Serverelemente
GraphQL-Yoga
Typdefinitionen
Resolver
Subscriptions

04 FRAGEN

Fragen und Diskussion

01 02 03 04

GRAPHQL GRUNDLAGEN

GraphQL Grundpfeiler
Vergleich zu REST

GRAPHQL CLIENT

GraphQL Konzepte
Voyager
Playground
Queries

GRAPHQL SERVER

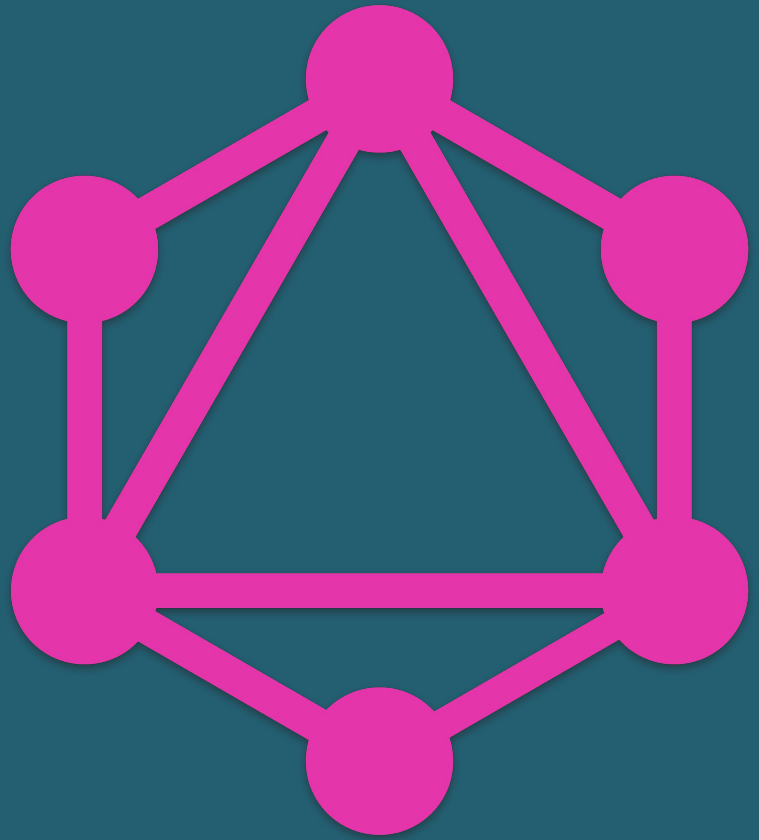
Serverelemente
GraphQL-Yoga
Typdefinitionen
Resolver

FRAGEN

Fragen und Diskussion

GraphQL

ist die Spezifikation einer **Abfragesprache** für APIs und einer serverseitigen **Laufzeitumgebung** für die Ausführung von Abfragen unter Verwendung eines **Typsystems**, welches für die abgefragten Daten definiert ist

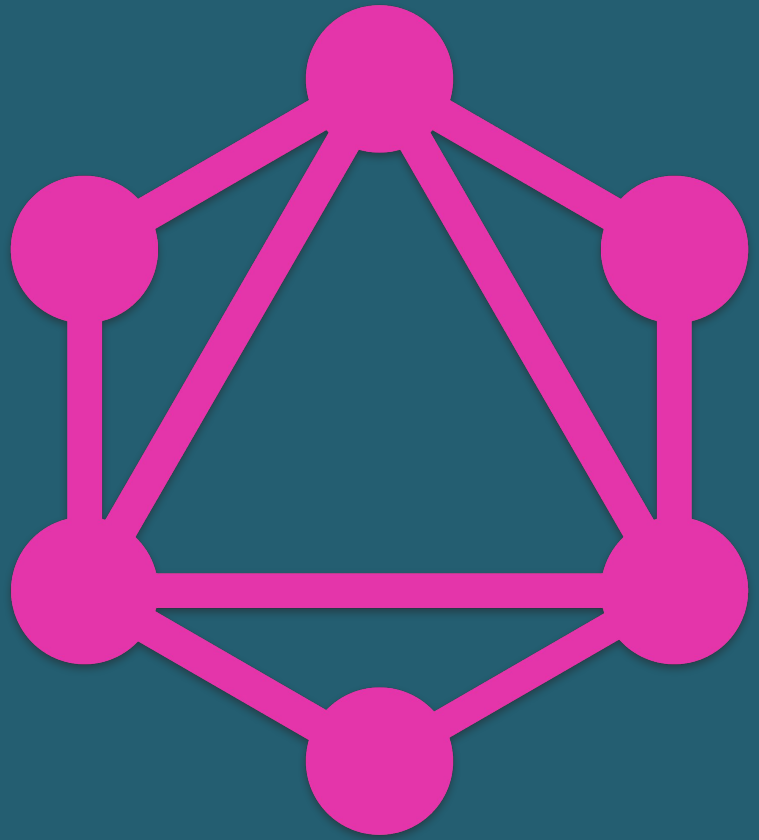


GraphQL

wurde von **Facebook** im Jahr 2012 aufgrund der Schwächen der bestehenden Architekturstile, wie REST und SOAP, und den derzeitigen Anforderungen entwickelt

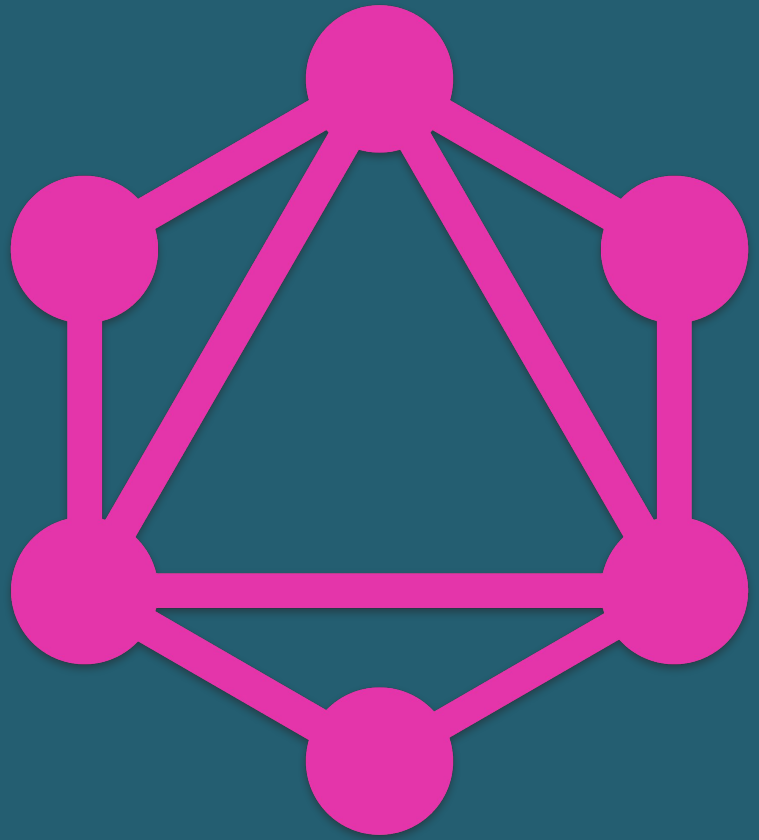
Open Source Spezifikation seit 2015

Gehostet von **Linux Foundation** 2018



GraphQL Grundpfeiler

- Stark typisierte Schnittstelle mit hierarchischem Aufbau
- Produktzentrierte Entwicklung (Anforderungen des Front-Ends zuerst / gegen Schnittstelle entwickeln)
- Client definiert bei Anfrage, welche Daten er haben möchte
- Introspektive Schnittstelle



REST GraphQL

REST	GraphQL
Endpunkte	Schema- und Typsystem (nur 1 Endpunkt)
CRUD-Operationen	Query, Mutation, Subscription
Server-driven	Client-driven
Langsam / Mehrere Anfragen	Schnell / eine Anfrage und spezifischer Aufruf
Ressourcenbasierte Anwendungen	Mehrere Microservices, mobile Anwendungen
Manuelle Dokumentation	Automatische Dokumentation


PayPal
sky
dailymotion
KLM

01

GRAPHQL GRUNDLAGEN

GraphQL Grundpfeiler
Vergleich zu REST

02

GRAPHQL CLIENT

GraphQL Konzepte
Voyager
Playground
Queries

03

GRAPHQL SERVER

Serverelemente
GraphQL-Yoga
Typdefinitionen
Resolver

04

FRAGEN

Fragen und Diskussion

Scalar Types

INT

Vorzeichenbehafteter 32-Bit Integer

FLOAT

Vorzeichenbehafteter Float mit doppelter Genauigkeit

STRING

UTF-8 Zeichenfolge

BOOLEAN

true oder false

ID

Eindeutige Kennung als String serialisiert

```
Address {
  street_name: String
  street_number: String
  city: String
  zip_code: Int
  country: String
}
```

Nullability

garantiert, dass non-nullable Attribute präsent (nicht null) sind bei einer GraphQL-Anfrage

```
name: String
name: String!

names: [String]
names: [String!]
names: [String]!
names: [String!]!
```

Object & Enum Types

OBJECT

Repräsentiert ein Objekt, dessen Attribute und ggf. dessen Beziehung zu anderen Objekten

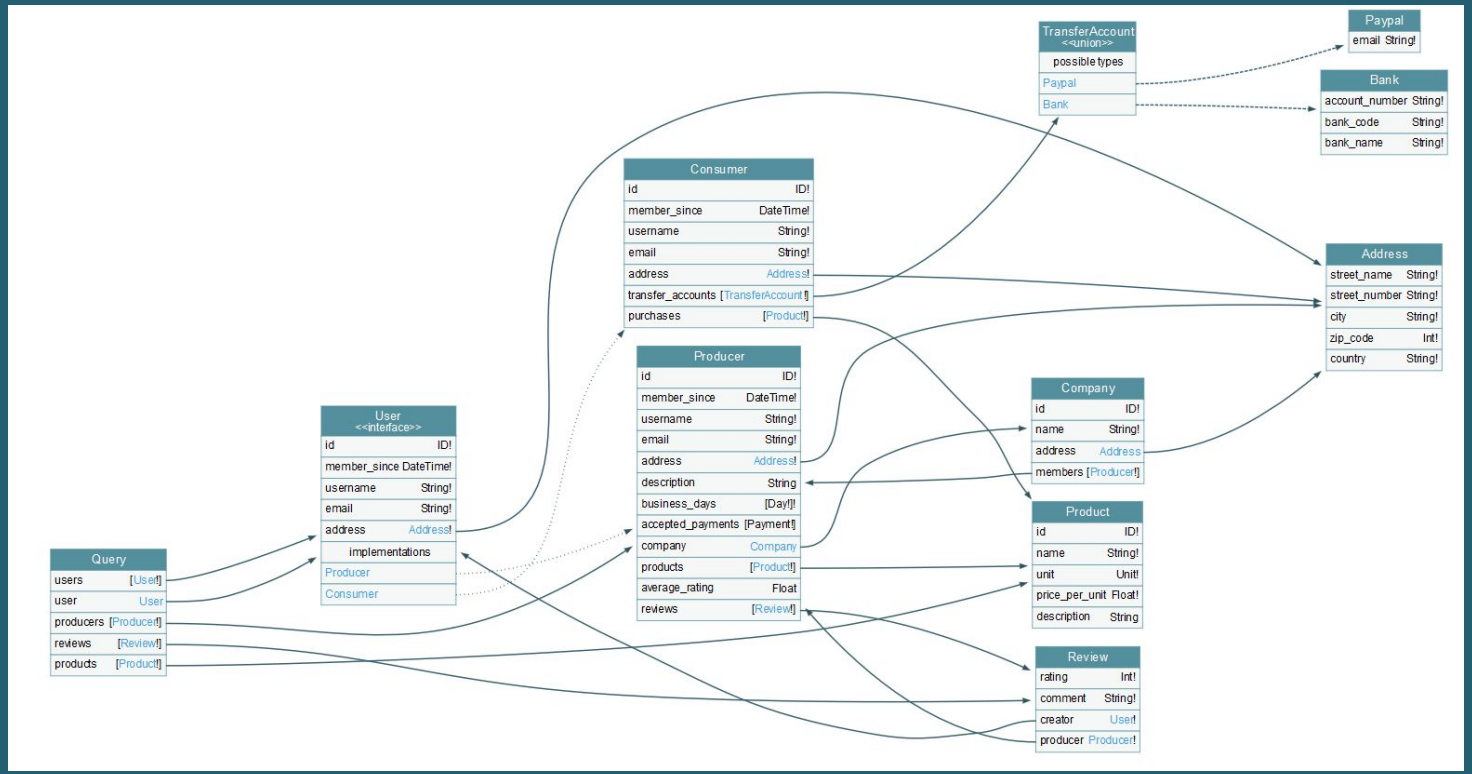
```
Company {
  id: ID!
  name: String!
  address: Address
  members: [Producer!]
}
```

Enum

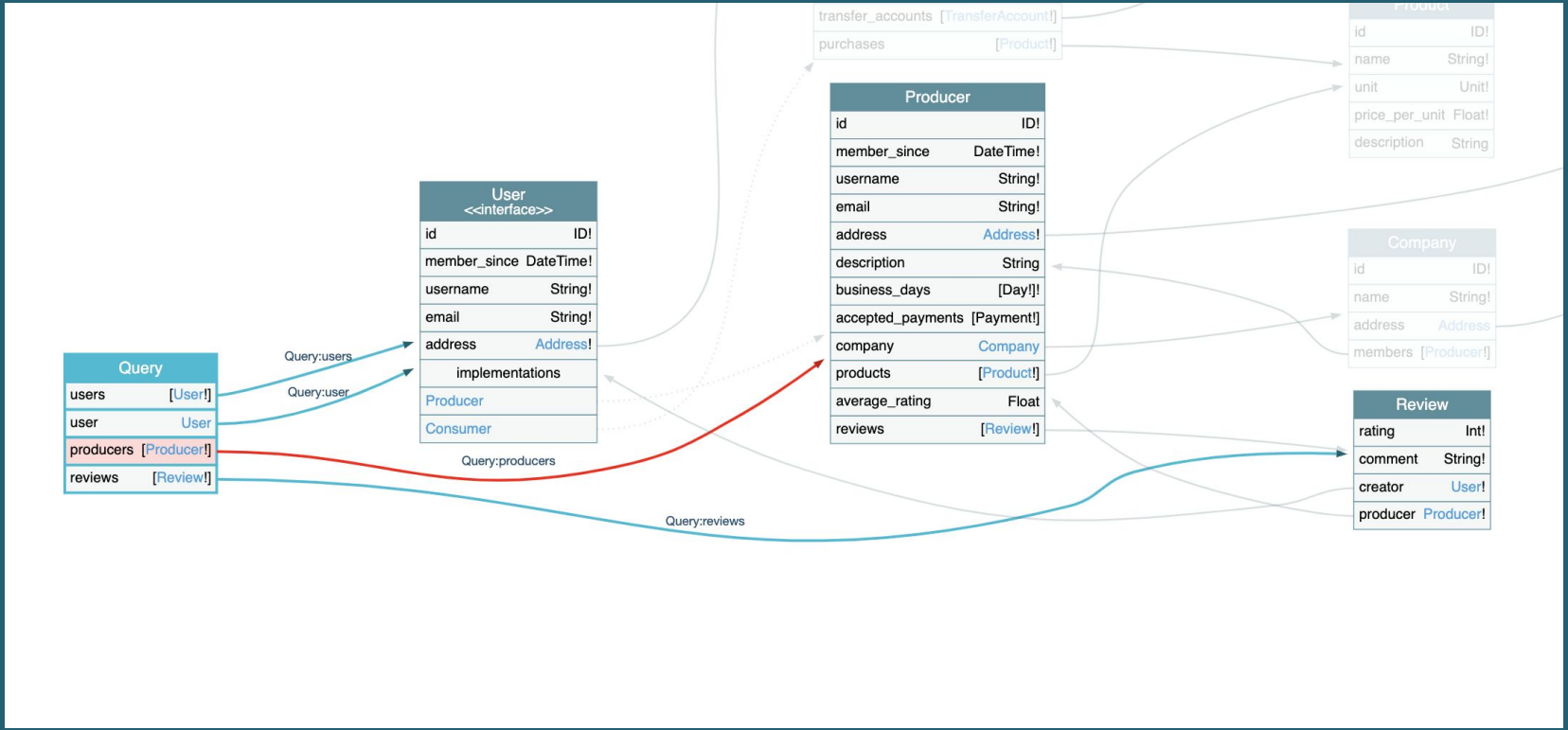
Aufzählungstyp mit endlichen und fest definierten Ausprägungen.

```
WorkingDay {
  MONDAY
  TUESDAY
  WEDNESDAY
  THURSDAY
  FRIDAY
}
```

GraphQL Voyager



GraphQL Query



GraphQL Query

Basic Query

```
{

  producers {
    id
    products {
      id
      ...
    }
    ...
  }

}
```

Named Query

```
query getProducers {

  producers {
    id
    products {
      id
      ...
    }
    ...
  }

}
```

Query mit Argumenten

```
query getProducers {

  producers(name:"Peter") {
    id
    products(name: "Apfel") {
      id
      ...
    }
    ...
  }

}
```

Playground Aufgaben 1+2

10 Minuten

- Ermitteln sie mithilfe der user-Query die ID des Nutzers mit dem username "klaus-dieter". Rufen sie anschließend alle von diesem Nutzer erstellten Reviews ab (reviews-Query). Fragen sie hier alle Attribute an (bei creator und producer reicht jedoch jeweils der username).

- Der Nutzer "klaus-dieter" hat einige Reviews erstellt, in welchen er dem Nutzer "peter-lustig" unbegründet schaden möchte. Nun ist Rache angesagt! Schreiben sie als "peter-lustig" ein Hate-Review gegen "klaus-dieter". Nutzen sie hierfür die createReview-Mutation. Benötigte ID's können mit der zuvor erstellten user-Query ermittelt werden.

Variables

VARIABLE

dient der Parametrisierung von Queries

```
mutation Review($producer:ID!, $creator:ID!){
  createReview(
    producer: $producer
    creator: $creator
  ){
    ...
  }
}
```

Query variables

```
{
    "producer": "da8ab4c0",
    "creator": "d467f50a"
}
```

Fragments

FRAGMENTS

ermöglichen es eine Menge von Attributen in verschiedenen Queries einzusetzen

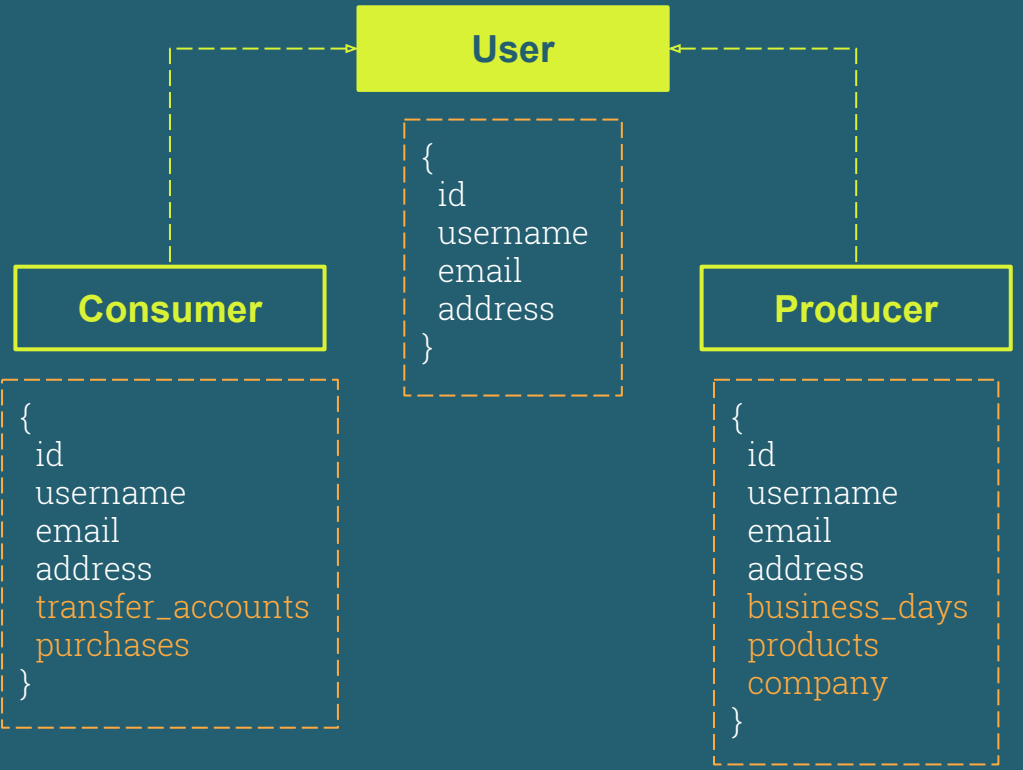
```
fragment UserFragment on User {
  id
  username
  address{
    street_name
    city
  }
}

query Users{
  users{
    ...UserFragment
  }
}
```

Interfaces

INTERFACE

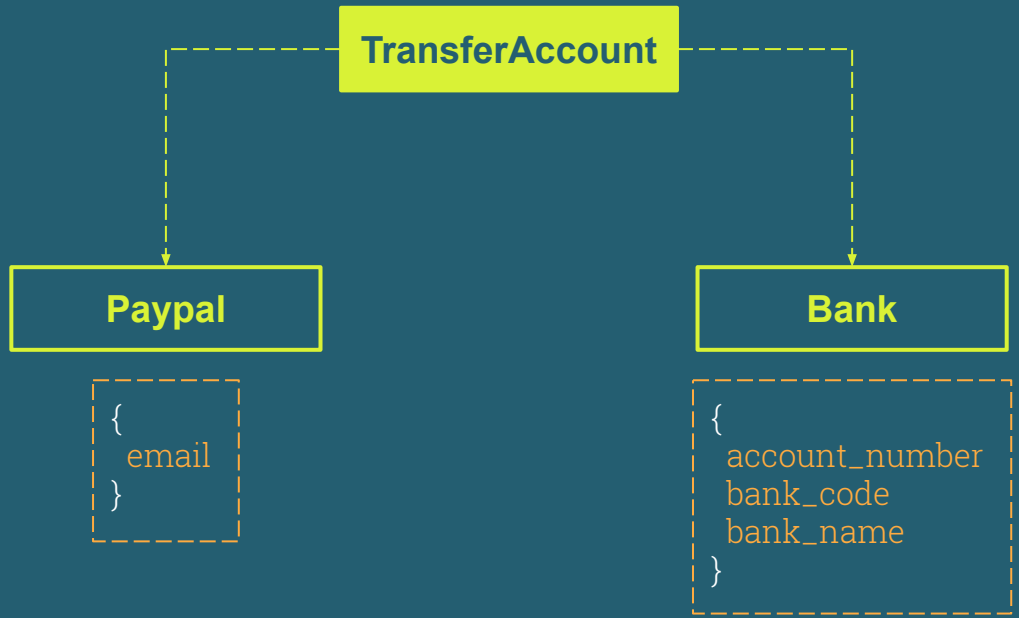
ist ein abstrakter Typ, welcher eine Kategorie von Typen darstellt. Definiert Attribute, die für alle Sub-Typen identisch sind.



Union Types

UNION TYPES

ähneln den Interfaces.
Sub-Typen besitzen jedoch keine gemeinsamen Attribute und werden lediglich für denselben Zweck verwendet.



Inline Fragments

Inline Fragments

Unterscheidung von Sub-Typen
und direkter Zugriff auf
Typ-spezifische Attribute.

Identisch für Interfaces und Union
Types.

```
query ... {
  transfer_accounts {
    ... on Paypal {
      email
    }

    ... on Bank {
      account_number
      bank_code
      bank_name
    }
  }
}
```

Playground Aufgaben 3+4

10 Minuten

- Erstellen sie Fragments für die Typen User, Producer und Consumer und nutzen sie diese in den zuvor erstellten Queries und Mutations. (Optional: Nutzen sie auch Variablen)
  a. Das User-Fragment sollte alle Attribute eines Users beinhalten.
  b. Beim Producer-Fragment reichen die Attribute business_days, accepted_payments und products.
  c. Beim Consumer-Fragment müssen die Attribute transfer_accounts und purchases angegeben werden. Achten sie darauf, dass das Attribut transfer_accounts ein Union-Type ist (Inline-Fragments).

- Nutzen sie nun die users-Query, um alle Nutzer mit den zu ihrer Rolle passenden Attributen auszugeben (Inline-Fragments). Welche Produkte bieten "peter-lustig" und "klaus-dieter" an? Welche Produkte wurden vom Herrn Paschulke bereits gekauft?

GraphQL Request-Struktur

```
POST /graphql HTTP/1.2
Host: localhost:3000/graphql
Content-Type: application/json
Body: {
   "query": "query User{ user(name:  "klaus-dieter"){ id username } }",
   "operationName": "User",
   "variables": {}
}
```

```
Response Body: {
   "data": {
      "user": {
         "id": "da8ab4c0",
         "username": "klaus-dieter"
      }
   }
}
```

Noch Fragen ?

Weiter mit der Server-Implementierung und GraphQL-Subscriptions im zweiten GraphQL-Workshop . . .

QUELLEN

- Bilder:
  - https://www.pexels.com/
  - https://www.graphql.com/
  - https://insights.stackoverflow.com/trends
  - https://www.getpostman.com/

- https://goodapi.co/blog/rest-vs-graphql
- https://www.apollographql.com/docs/apollo-server/