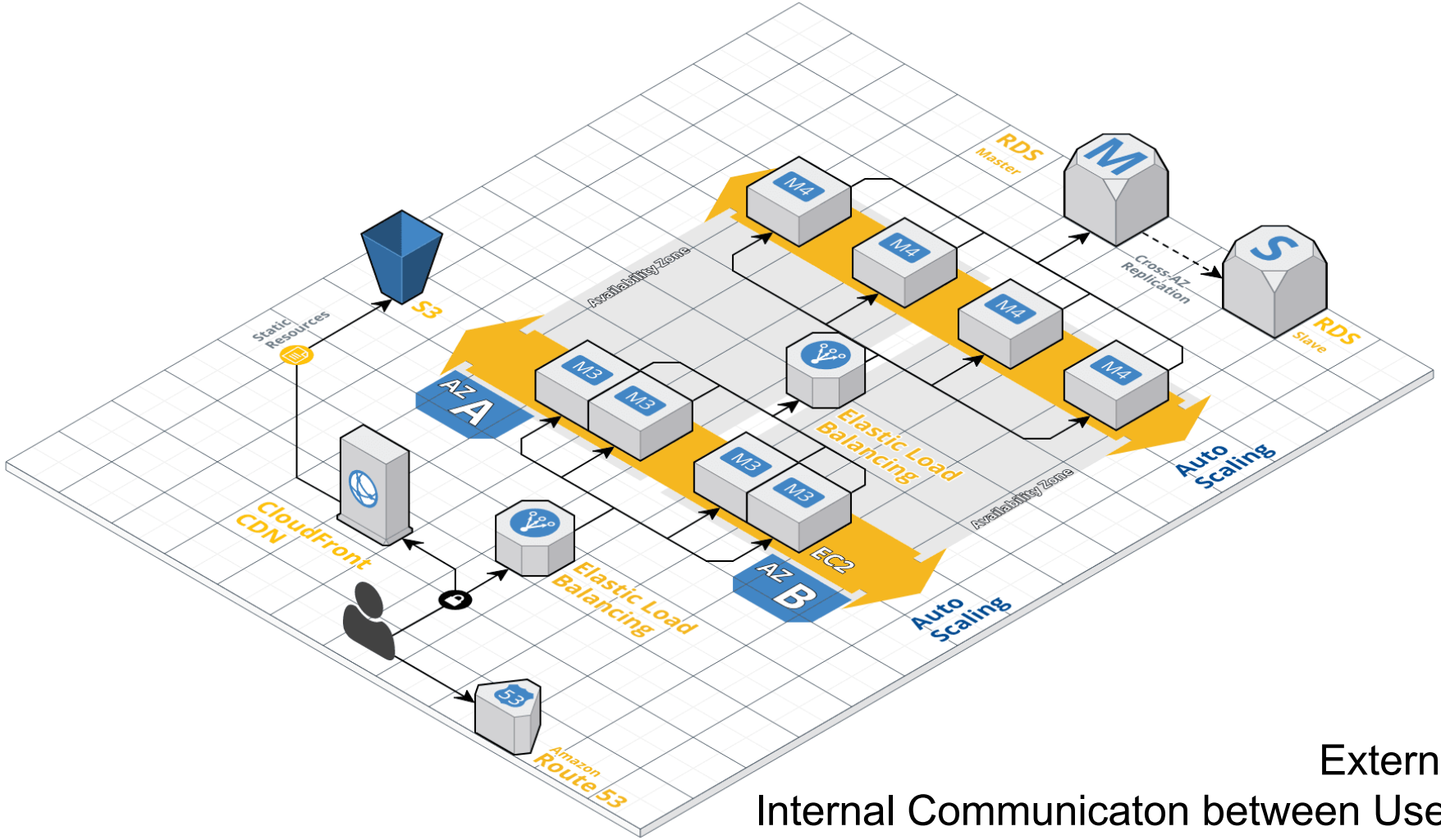
Communication

MICROSERVICES

Content

- Overview
- REST
- gRPC
- MQTT (in an IoT World)
- Backend 4 Frontend Pattern (REST)
- graphQL
- Messaging

Overview



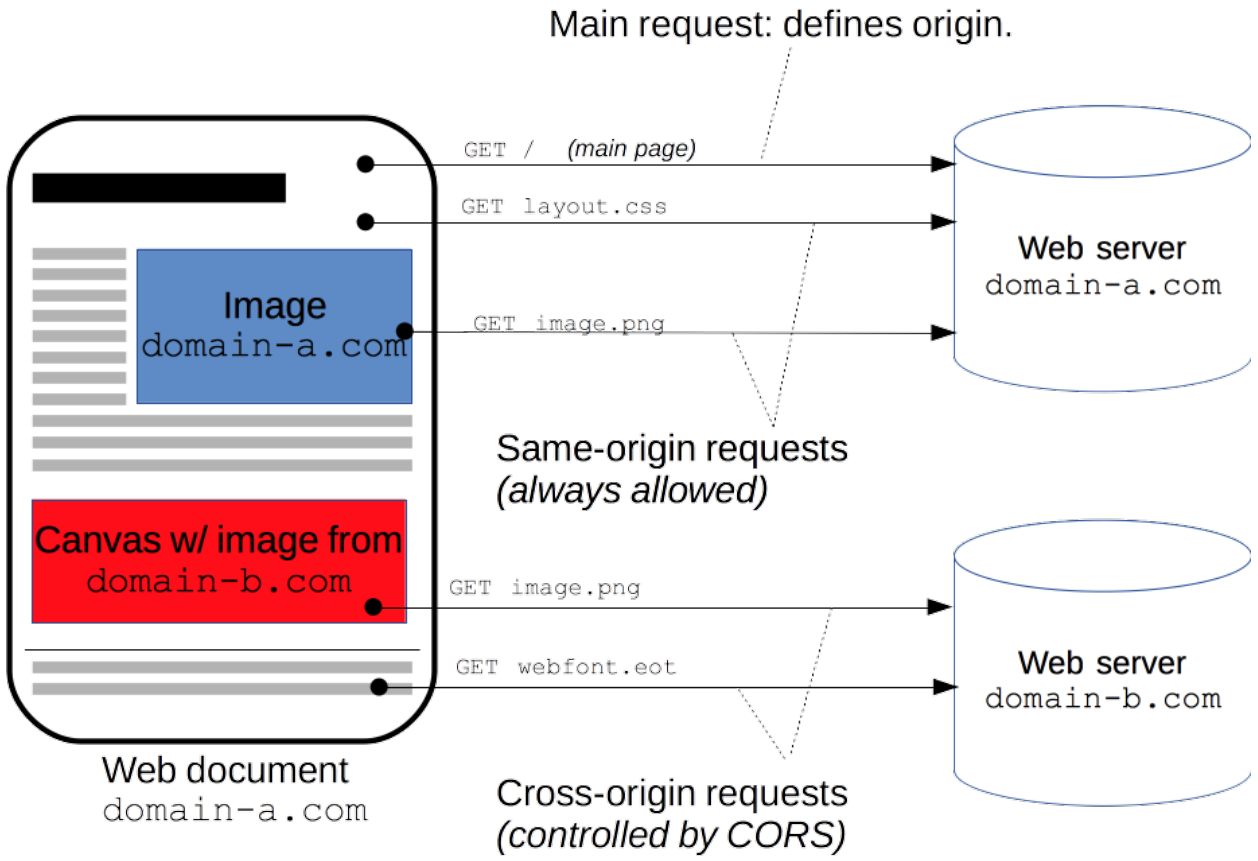
External and
Internal Communicaton between User and
different microservices via graphQL, REST over HTTP, gRPC, ...

REST – Fundamentals & Best Practises

- ➔ VV Semester 4
- Resources
  - Nouns, not Verbs
  - Coarse Grained, not Fine Grained
  - Architectural style for use-case scalability
- Keep It Simple
  - Collection Resource /users/
  - Instance Resource /users/1
- Behavior
  - GET, PUT, POST, DELETE („CRUD“), Head (Headers, no Body)
- Use HTTP Reponse Codes
- Date/Time/Timestamp
  - ISO 8601 is the standard
  - Use UTC!
- Use query params for offset and limit
- ....

REST – Accept CORS

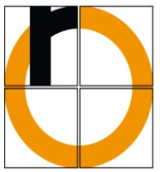
- Cross-Origin Resource Sharing (CORS) is a mechanism that uses additional HTTP headers to let a user agent gain permission to access selected resources from a server on a different origin (domain) than the site currently in use.



https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

REST – API Anti Patterns!

- Poor error handling
- REST APIs that ignore HTTP rules
- Exposing your raw underlying data model
- Security complexity
- Unexpected & undocumented releases
- Poor developer expereience
- Poor documentation



REST

RESTful API

GET /user/1

```
{
„username“: „jona7o“,
„email“: „tobias.jonas@fh-
rosenheim.de“,
...
}
```

Non-RESTful API

GET /last_active_users?page=3

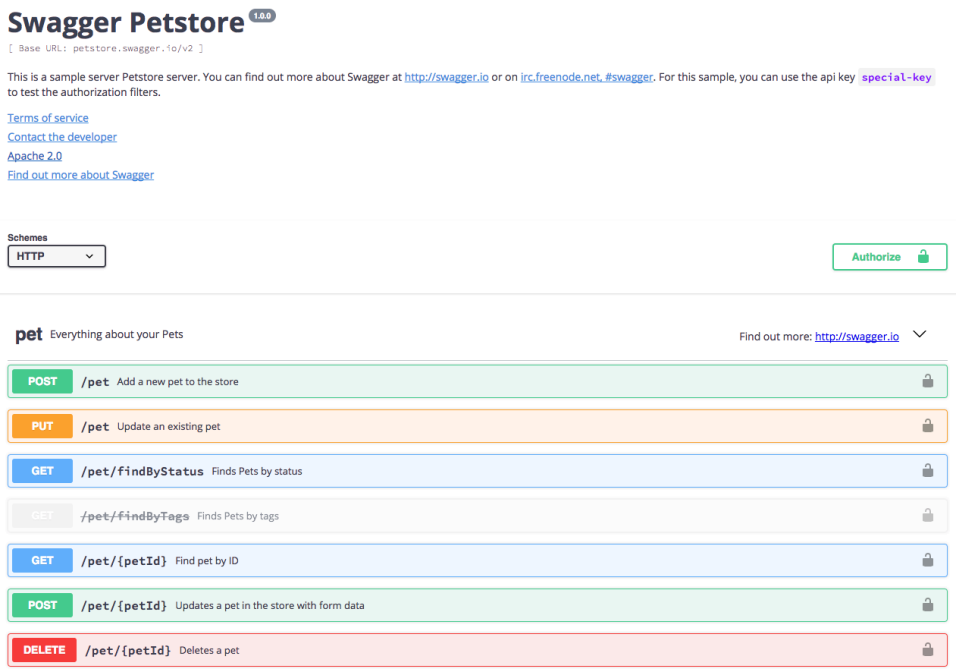
```
{
„users“: [...]
...
}
/// more shit
/getAccount
/getAllAccounts
/verifyAccountEmailAddress
/searchGroupsByName
```

➔ Smells like bad RPC. DON‘T DO THIS!! Then look at gRPC!!!
➔ I am getting frustrated by the number of people calling any HTTP-based interface a REST API – Roy Fielding

REST – OpenAPI as API documentation

- Framework for describing APIs for the discovery of APIs by humans and machines

- You can generate your swagger documentation from code
- Or you can generate code for your API Endpoints
- YAML or JSON file



- https://www.openapis.org/
- Swagger is the most popular API Tool
  - https://swagger.io/

gRPC

- gRPC stands for gRPC (Remote Procedure Calls)
- A high performance, general purpose, feature-rich RPC framework (RPC ➔ 4. Semester VV Prof. Beneken)
- Part of Cloud Native Computing Foundation cncf.io
- HTTP/2 and mobile first
- Open sourced version of Stubby RPC used in Google
- gRPC is not RMI
- Abstractions and best practices on how to design RPCs

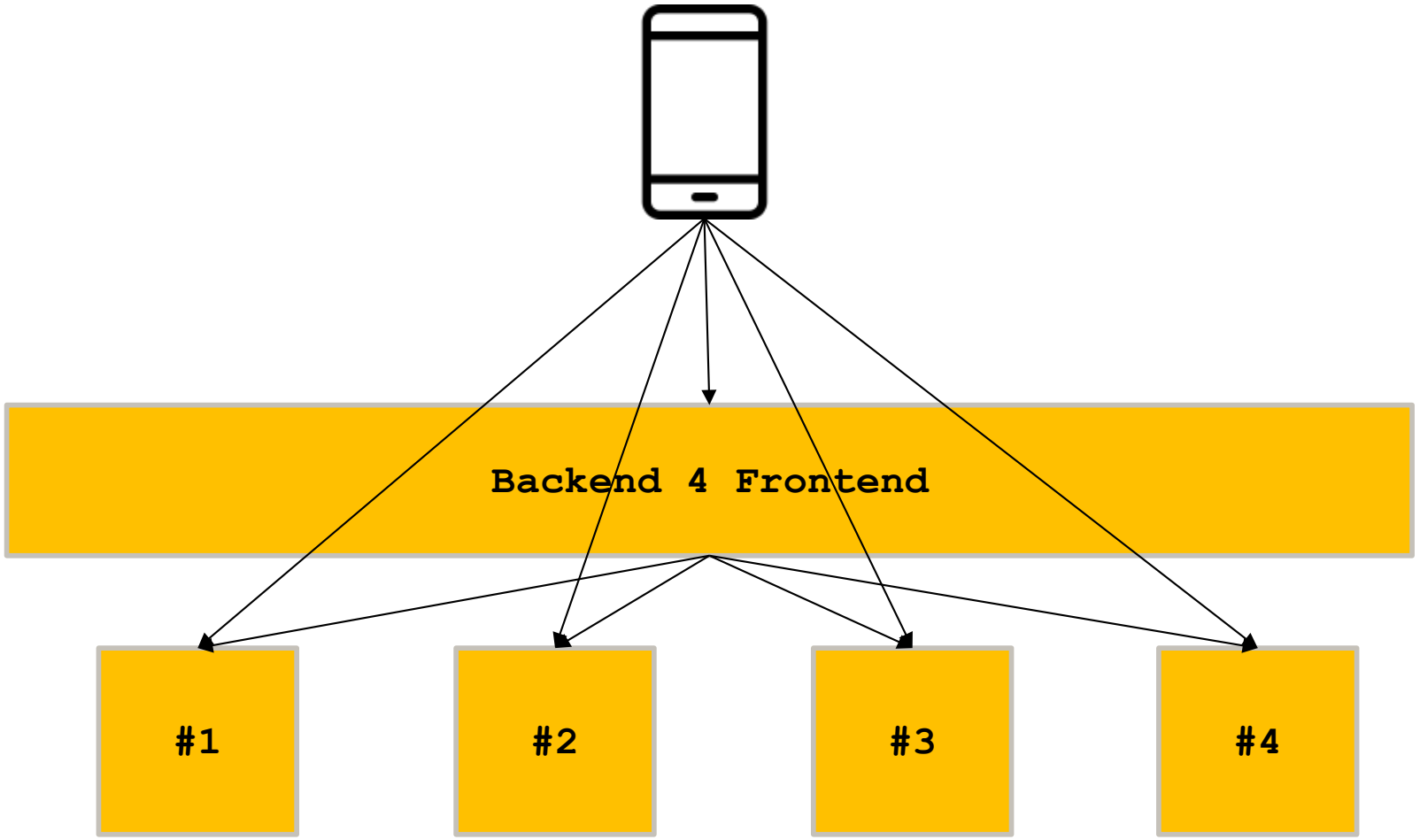
- more gRPC in presentation from 23.11.2017

MQTT

- MQTT: Message Queue TelemetryTransport
- Very lightweight messaging protocol – used in IoT World
- Publish/Subscribe Architecture
- 3 QOS Levels higher
  - higher level → more server resources needed
  - higher level → more bandwidth needed
- A lot of clients on a single server

- more MQTT in presentation from 23.11.2017

Backend 4 Frontend



What is graphQL

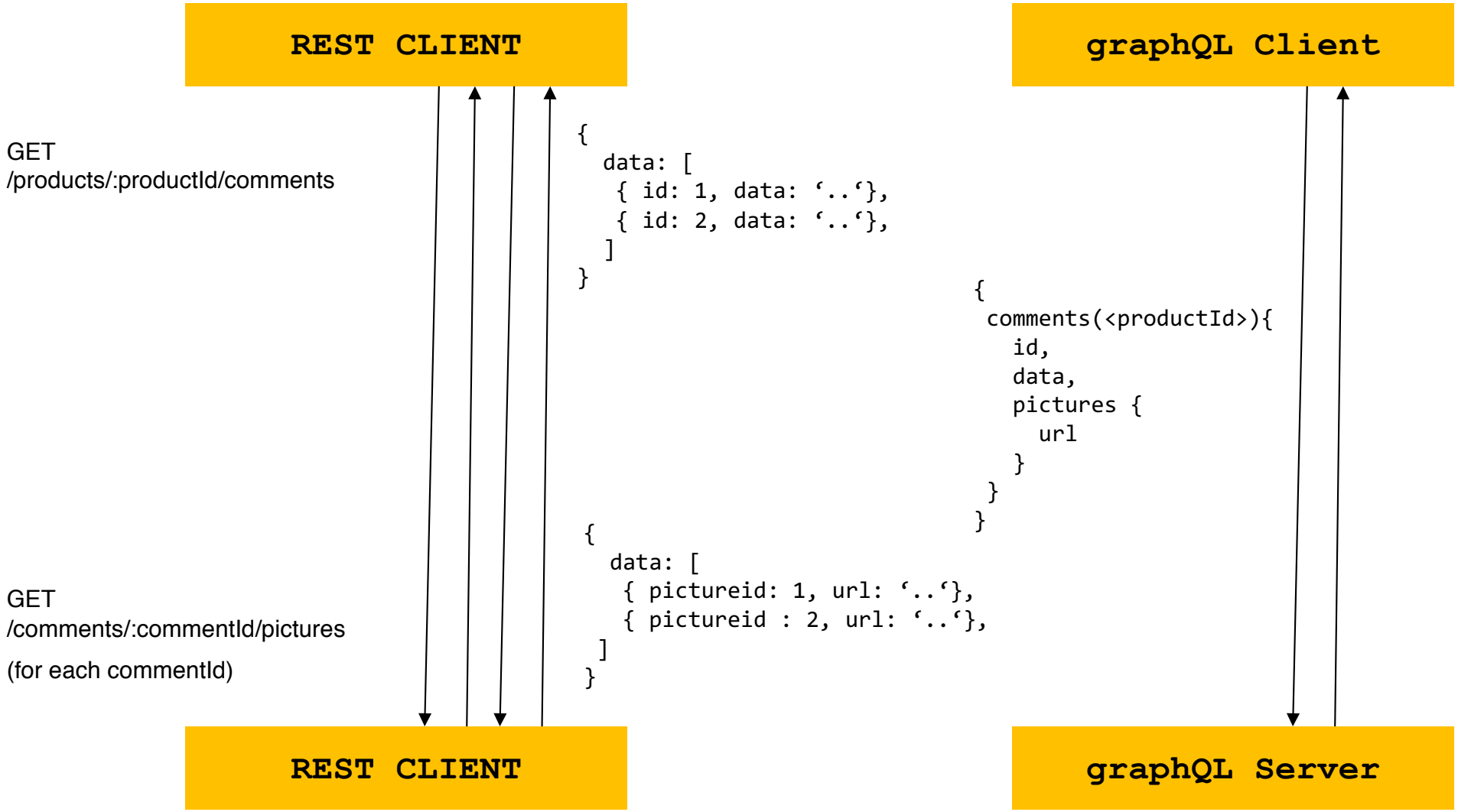
- A query language for your API
- Created by Facebook in 2012
- Key conecpts of the graphQL query language are
  - hierachrical
  - product-centric
  - strong-typed
  - Introspective

- Why graphQL?
  - Ask for what you need, get exactly that
  - Get many resources in a single request
  - Describe what‘s possible with a type system
  - Move faster with powerful developer toools
  - Evolve your API without versions
  - Bring your own data and code

https://www.slideshare.net/Codemotion/tomer-elmalem-graphql-apis-rest-in-peace-codemotion-milan-2017

https://apis.guru/graphql-voyager/

REST vs. graphQL



Messaging

- Services must handle requests from the application's clients. Furthermore, services must sometimes collaborate to handle those requests. They must use an inter-process communication protocol.
- Solution
  - Use asynchronous messaging for inter-service communication. Services communicating by exchanging messages over messaging channels.
- There are numerous examples of asynchronous messaging technologies
  - Apache Kafka
  - RabbitMQ
- This pattern has the following benefits:
  - Loose coupling since it decouples client from services
  - Improved availability since the message broker buffers messages until the consumer is able to process them
  - Supports a variety of communication patterns including request/reply, notifications, request/async response, publish/subscribe, publish/async response etc

http://microservices.io/patterns/communication-style/messaging.html

Kafka – What is Kafka?

- Distributed, partitioned, replicated commit log service
- Pub/Sub messaging functionality
- Created by LinkedIn, now an Apache open-source project
- Fast
  - But helps with Back-Pressure (Fast Producer, Slow Consumer Problem)
- Resilient
  - Brokers persist data to disk
  - Broker Partitions are replicated to other nodes
  - Consumers start where they left off
  - Producers can retry at-least-once messaging
- Scalable
  - Capacity can be added at runtime wihout downtime
  - Topics can be larger than any single node could hold
  - Additional partitions can be added to add more parallelism
- Perhaps we will deep dive into Kafka in data persistence lesson (event sourcing in a journal with kafka)

https://kafka.apache.org/