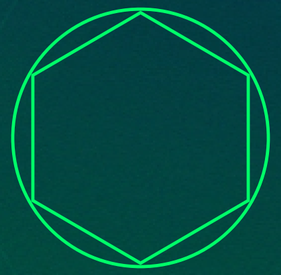
SUPERSCALING EXPERT KNOWLEDGE



ERIUM

MASTER THE COMPLEXITY

ERIUM

TABLE OF CONTENTS

ERIUM

RESTFUL APIS

REpresentational **S**tate **T**ransfer

WAS BEDEUTET RESTFUL ÜBERHAUPT?

REST ist ein architektonischer Stil und Sammlung von Best Practices für die Entwicklung von (HTTP) APIs.

HTTP APIs

Schnittstellen, die es Programmen (= Clients), ermöglicht über das Internet Informationen auszutauschen.

PRINZIPIEN

1. Client - Server
2. Stateless
3. Cachable
4. Uniform Interface
5. Layered System
6. Code on demand (optional)

APIs die alle 6 Prinzipien erfüllt = RESTful API

ERIUM

WARUM HTTP APIS?

BASIEREND AUF HTTP

URLs, Server, Clients, Requests, Responses, Headers, JSON, Caching, Versioning, Content Types, Sessions, etc.

HTTP API ≠ REST API

Nicht jede HTTP API ist automatisch eine REST API
Nicht jede REST API ist automatisch eine HTTP API

ZUGANG ZU DATEN

Nutzer können zeitnahe und ohne Einschränkungen* auf Daten zugreifen:

- Erstellen (**C**reate) HTTP POST
- Lesen (**R**ead) HTTP GET
- Aktualisieren (**U**pdate) HTTP PUT
- Löschen (**D**elete) HTTP DELETE

GUIDING PRINCIPLES

1. CLIENT SERVER
Client und Server dürfen keine Abhängigkeiten zueinander haben. Einfache Skalierbarkeit dank SoC.

2. STATELESS
Der Server kennt den Zustand der Clients nicht. Jede HTTP Anfrage wird als neu betrachtet und muss alle benötigten Informationen beinhalten.

3. CACHABLE
Caching sollte genutzt werden, wenn möglich. *Eine Anfrage die nicht gestellt werden muss ist die schnellste Anfrage.*

4. UNIFORM INTERFACE
Jede Resource hat eine eindeutige "Adresse" (= URI), sollte selbsterklärend sein und sollte mit Hilfe von Standardmethoden manipulierbar sein. Sollte/muss **Hypermedia** unterstützen!

5. LAYERED SYSTEM
Dem Client ist nur eine Schnittstelle bekannt, dahinter-liegende Ebenen/Resourcen bleiben verborgen.

6. CODE ON DEMAND (optional)
Der Server übertragt dem Client im Bedarfsfall Code zur lokalen Ausführung.

ERIUM

= RESTFUL API

" All the above constraints help you build a truly RESTful API, and you should follow them. Still, at times, you may find yourself violating one or two constraints. Do not worry; you are still making a RESTful API - but not “truly RESTful. ”

CLIENT

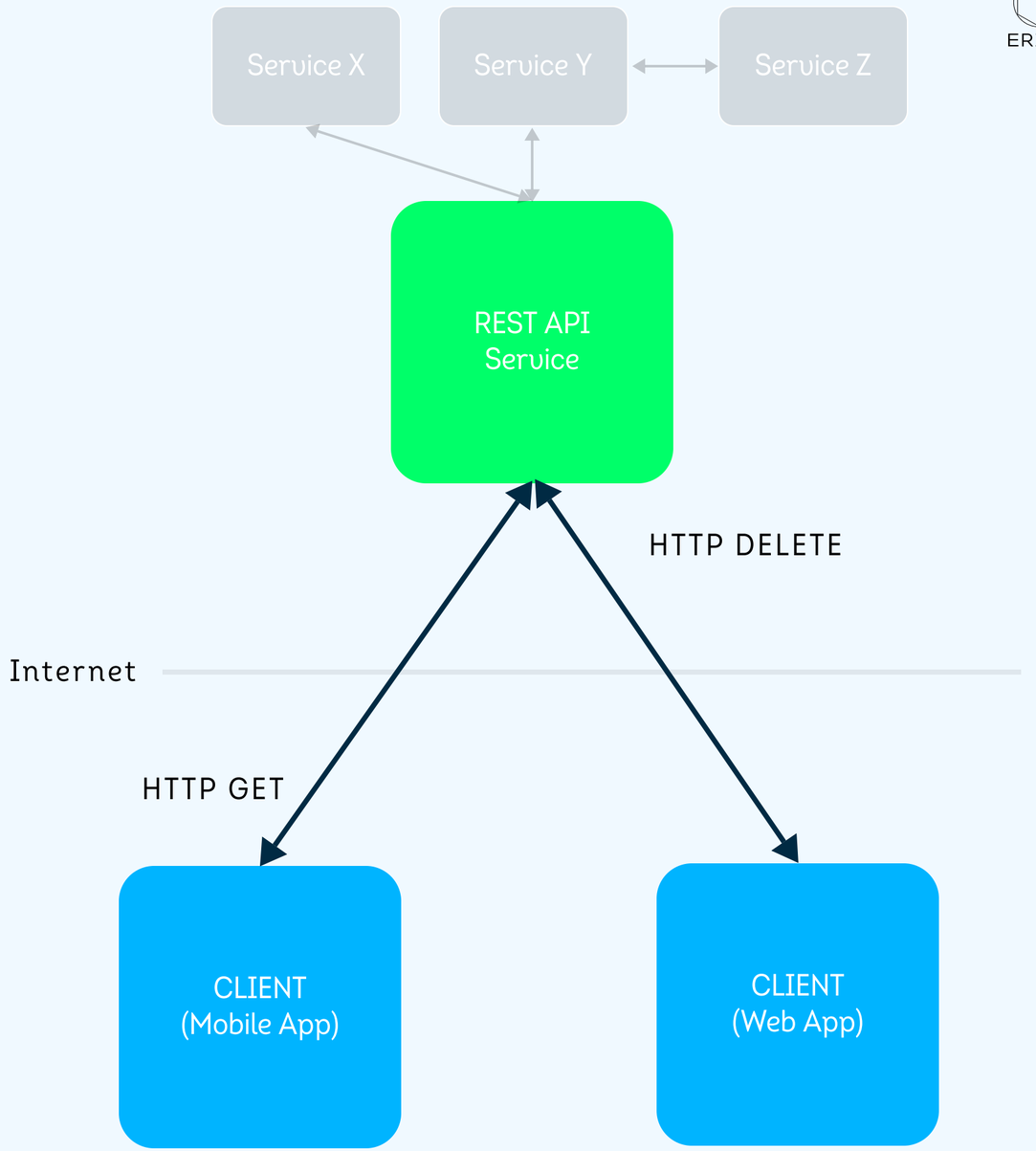
Jeder HTTP-fähiger Client bzw. Programmiersprachen und Tools. Z.B. Browser, Mobile Apps, M2M Software, etc.

SERVER

Web bzw. Application-Server die HTTP Anfragen entgegennehmen und beantworten.

Server sind **stateless** und behalten den Zustand zwischen mehreren Anfrage nicht.

Vertikale und Horizontale Skalierbarkeit dank Segregation of Concerns.



AUFBAU VON
REST APIS 1/2

ERIUM

GET http://myapp.com/api/persons/123

RESOURCES

Jede Information die benannt werden kann = Resource
Dokument, Objekte (z.B. Person), Bilder, etc.

RESOURCE IDENTIFIER

Eine eindeutige Zuordnung von Resources.
URI = Unique Resource Identifier

RESOURCE METHODS

Methoden um eine Resource zu verändern.

In dem meisten Fällen Resource Method = HTTP Method

- Create Resource = HTTP POST
- Read Resource = HTTP GET
- Update Resource = HTTP PUT
- Delete Resource = HTTP DELETE

Welche Methode verwendet wird ist egal, solange das **4. Uniform Interface** Prinzip eingehalten wird.

AUFBAU VON REST APIS 2/2

ERIUM

GET http://myapp.com/api/persons/123

RESOURCE REPRESENTATION

Besteht aus den Daten selbst, Metadaten und **Hypermedia** Links. Das Format der Daten wird **Media Type** genannt.

Daten

```
public class Person {
  int id;
  string firstName;
  string lastName;
  int age;
}
```

JSON

```
{
  "id": 123,
  "firstName": "Elon",
  "lastName": "Musk",
  "age": 49
}
```

XML

```
<Person>
  <id>123</id>
  <firstName>Elon</firstName>
  <lastName>Musk</lastName>
  <age>49</age
</Person>
```

YAML

```
id: 123
firstName: Elon
lastName: Musk
age: 49
```

```yaml
openapi: 3.0.0
info:
  title: My API
  version: 1.0.0
servers:
  - url: http://myapp.com/api

paths:
  /persons/{id}:
    get:
      summary: Returns a person by ID.
      parameters:
        - name: id
          in: path
          required: true
          description: The unique identifier of the person
          schema:
            type : integer
            minimum: 1
      responses:
        '200':
          description: A user object
          content:
            application/json:
              schema:
                type: object
                properties:
                  id:
                    type: integer
                    example: 4
                  firstName:
                    type: string
                    example: Elon
                  lastName:
                    type: string
                    example: Musk
                  age:
                    type: integer
                    example: 49
```

ERIUM

DOKUMENTATION VON **REST APIS?**

OpenAPI Specification (Swagger)

Industriestandard um APIs zu dokumentieren. Unabhängig von gewählter Programmiersprache bzw. Stack. **https://swagger.io**

DEMO

ERIUM

IMMER PLURAL

GET /**persons**/123 **statt** GET /**person**/123

SUB RESOURCES FÜR BEZIEHUNGEN

GET /persons/123/pets/3
GET /pets/3

MEDIA TYPES ÜBER HTTP HEADER

Content-Type: application/json; charset UTF-8;
Accept: application/json, application/xhtml+xml;

FILTERING, SORTING, PAGING & FIELDS

GET /persons?age<=50
GET /persons?sort=+age
GET /persons?fields=firstName
GET /persons?offset=20&limit=10

HTTP ERROR CODES

- 200 OK
- 201 RESOURCE CREATED
- 204 RESOURCE DELETED
- 401 UNAUTHORIZED
- 404 RESOURCE NOT FOUND

B(R)EST PRACTICES?

ERIUM

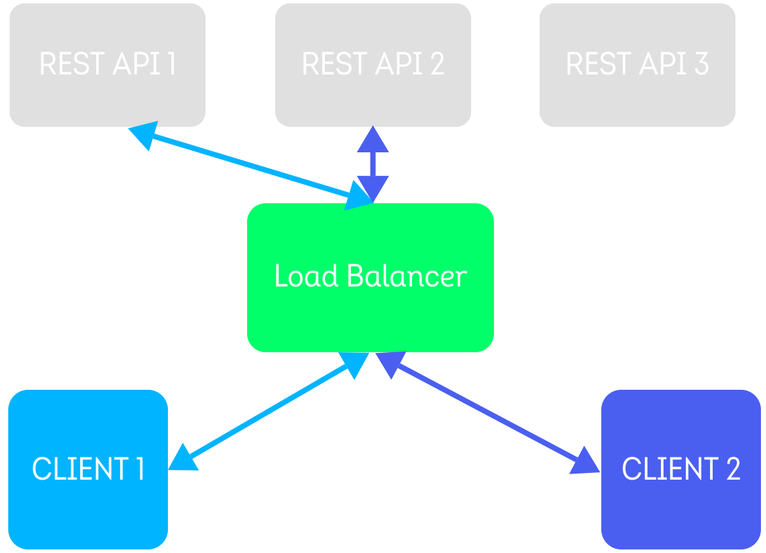
SKALERIUNG VON **REST APIS**

VERTIKALE SKALIERUNG (scale up)

"Aufrüsten" von bestehender Infrastruktur
- mehr CPU/RAM
- neuere CPU Modelle

HORIZONTALE SKALERIUNG (scale out)

Hinzufügen neuer Infrastruktur:
- mehrere Server
- Load Balancing
  - Round Robin
  - Least connections
  - Source IP Hash



ERIUM

AUTHENTIFIZIERUNG VON **REST APIS?**

HTTP BASIC AUTH

Username + Password in Base64 encodiert
Authorization: Basic bG932....

BEARER (TOKEN) AUTH

Ein eindeutiger Token wird von der API generiert.
Authorization: Bearer <Token>

ANDERE AUTH MÖGLICHKEITEN

- Form Based
- Json Web Token (JWT)
- OAuth
- etc.

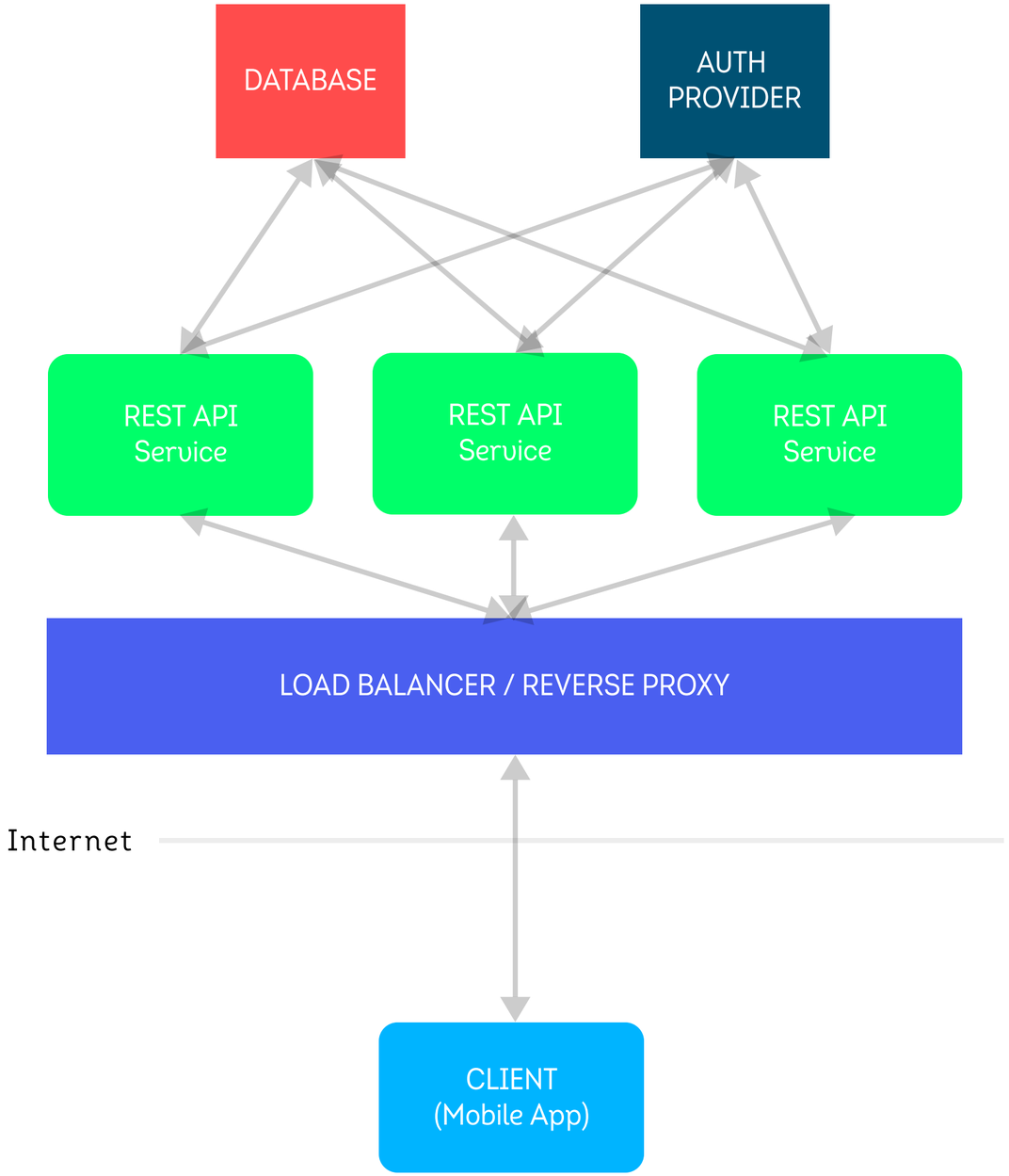
ERIUM

REST UND HYPERMEDIA

HATEOAS

Hypermedia As The Engine Of Application State

```
{
    "account": {
        "account_number": 12345,
        "balance": {
            "currency": "usd",
            "value": 100.00
        },
        "links": {
            "deposit": "/accounts/12345/deposit",
            "withdraw": "/accounts/12345/withdraw",
            "transfer": "/accounts/12345/transfer",
            "close": "/accounts/12345/close"
        }
    }
}
```



ERIUM

ADVANCED API SETUP