HOCHSCHULE FULDA
UNIVERSITY OF APPLIED SCIENCES

WEB-APPLIKATIONEN

REST

AI
Manuel Richardt, Dominik Rupprecht
FB **ANGEWANDTE INFORMATIK**

REST ALS ARCHITEKTUR

1. REST-ARCHITEKTUR WURDE VON ROY THOMAS FIELDING ALS ARCHITEKTURSTIL FÜR VERTEILTE HYPERMEDIA SYSTEME FORMULIERT.
2. REST (REPRESENTATIONAL STATE TRANSFER)
3. DIE REST-ARCHITEKTUR NUTZT DAS HTTP-PROTOKOLL VOLLSTÄNDIG (IM GEGENSATZ ZU ANDEREN WEBSERVICES)

AI
Manuel Richardt, Dominik Rupprecht
FB **ANGEWANDTE INFORMATIK**

REST

1. METHODE FÜR WEBDIENSTE ZUM SENDEN UND EMPFANGEN VON DATEN.
2. DIE RESTFUL WEBSERVICES SIND ZUSTANDSLOS (STATELESS)
3. LIEFERN DIE DATEN IN UNTERSCHIEDLICHEN REPRÄSENTATIONEN (FORMATE) AN DEN AUFRUFER (MEISTENS IN JSON).
4. DATEN WERDEN MIT DER URI EINDEUTIG IDENTIFIZIERT UND ALS RESSOURCE BEZEICHNET.
5. DURCH DIE ZUSTANDSLOSE KOMMUNIKATION KÖNNEN ALLE CACHING MECHANISMEN VON HTTP-SERVERN VERWENDET WERDEN.

REST

1. VERWENDET HTTP-METHODEN
   - C – Create a new Entry (POST)
   - R – Read a Entity (GET)
   - U – Update a Entity (PUT)
   - D – Delete a Entity (DELETE)
2. PATCH, HEAD, OPTIONS
3. BROWSER KÖNNEN NUR GET UND POST

AI
Manuel Richardt, Dominik Rupprecht
FB **ANGEWANDTE INFORMATIK**

RESSOURCEN

1. RESSOURCE IST EINE SEMANTISCHE ENTITÄT UND WIRD ÜBER EINE LANGLEBIGE URI (UNIFORM RESOURCE IDENTIFIER) IDENTIFIZIERT BZW. ADRESSIERT.
2. SIE KANN EINE ODER MEHRERE REPRÄSENTATIONEN (TEXT, XML, JSON,…) BESITZEN.
3. BEISPIELE:
   - https://api.frank-rahn.de/customers/4711
   - https://api.frank-rahn.de/drivers/1234
   - https://api.frank-rahn.de/products/1
   - Oder auch: http://localhost/blog/post/12

SERVICE UND CONTROLLER

› RESTful Service als Routen

```
var express = require('express');
var router = express.Router();
var blogController = require('../controllers/apiBlogController');

/* http://localhost/api/blog */
router.route('/blog/')
  .get(blogController.getAllPosts)
  .post(blogController.createPost);

/* http://localhost/api/blog/newPost */
router.route('/blog/:postID')
  .get(blogController.readPost)
  .put(blogController.updatePost)
  .delete(blogController.deletePost)
```

› Mit entsprechendem Controller

```
var fs = require('fs');
var data = fs.readFileSync('models/blog.json');

let blog;
if (data.length > 0) {
    blog = JSON.parse(data);
} else {
    blog = [];
}

const getAllPosts = (req, res) => {
    res.status(200).json({ "success": true, "data": blog });
}

const createPost = (req, res) => {
    let post = {
        id: blog.length ? blog[blog.length - 1].id + 1 : 0,
        title: req.body.title,
        username: req.body.username,
        date: req.body.date,
        text: req.body.text,
        file: filename
    }
    blog.push(post);
    fs.writeFileSync('models/blog.json', JSON.stringify(blog, null, 2));
    res.status(201).json({ "success": true, "data": post });
}
```

AI
Manuel Richardt, Dominik Rupprecht
FB **ANGEWANDTE INFORMATIK**

HOCHSCHULE FULDA
UNIVERSITY OF APPLIED SCIENCES

WEB-APPLIKATIONEN

HTTP-METHODEN UND STATUSCODES

AI
Manuel Richardt, Dominik Rupprecht
FB **ANGEWANDTE INFORMATIK**

HTTP-METHODEN

1. GET
   - Anforderung einer Ressource
   - Webservice liefert die Repräsentation einer Ressource
   - Ist safe und idempotent
2. POST
   - Erzeugt eine oder mehrere Ressourcen deren URIs noch nicht bekannt sind.
   - Bietet die meiste Flexibilität, da sie weder safe noch idempotent ist

HTTP-METHODEN

3. PUT
   - Aktualisierung einer Ressource
   - Sollte die Ressource nicht existieren, wird sie unter der URI angelegt
   - Die Ressource wird vom Consumer in einer gültigen Repräsentation gesendet und ersetzt die bestehende Ressource
   - Ist idempotent

4. PATCH
   - Modifiziert nur einen Teil der angegebenen Ressource
   - Gegenüber der HTTP Methode PUT wird die Ressource nicht komplett überschrieben
   - Ist weder safe noch idempotent

HTTP-METHODEN

5. DELETE

- Löschen einer Ressource
- Falls die Ressource nicht existiert, wird kein Fehler ausgelöst, da das gewünschte Ergebnis schon erreicht ist.
- Die Ressource darf nicht mehr unter der ursprünglichen URI geliefert werden.
- Das mehrfache Löschen darf keinen Fehler liefern.
- Ist idempotent

6. HEAD

- Fragt den HTTP-Header zu einer identifizierten Ressource ab.
- Liefert den gleichen Header, wie die HTTP Methode GET, nur ohne Daten.
- Ist safe und idempotent

HTTP-METHODEN

7. OPTIONS

- Liefert die HTTP Methoden, welche auf der identifizierten Ressource zur Verfügung stehen.
- Ist idempotent

8. TRACE

- Liefert die Anfrage so zurück, wie der Server sie empfangen hat. So kann überprüft werden, ob und wie die Anfrage auf dem Weg zum Server verändert worden ist.
- Liefert den gleichen Header, wie die HTTP Methode GET, nur ohne Daten.
- Ist idempotent

HTTP STATUS CODES

1. DIE ANTWORT EINES RESTFUL WEBSERVICES BEINHALTET EINEN HTTP-STATUSCODE UND GGF. DIE ANGEFORDERTE RESSOURCE.
2. WERDEN IN FÜNF KLASSEN EINGETEILT
   - 100er Gruppe – Informative Antworten
   - 200er Gruppe – Erfolgreiche Antworten
   - 300er Gruppe – Umleitungen
   - 400er Gruppe – Client-Fehler
   - 500er Gruppe – Server-Fehler

HTTP STATUS CODES - AUSZUG

› 200er Gruppe - Erfolg
- 200 OK
- 201 Created
- 204 No Content

› 300er Gruppe - Weiterleitungen
- 301 Moved Permanently
- 303 See other
- 304 Not Modified

› 400er Gruppe - Fehlermeldungen
- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found

› 500er Gruppe – Fehler im Service
- 500 Internal Server Error
- 502 Bad Gateway
- 503 Service Unavailable
- 504 Gateway Timeout

AI
Manuel Richardt, Dominik Rupprecht
FB **ANGEWANDTE INFORMATIK**

ZUSAMMENFASSUNG

1. DIE REST-ARCHITEKTUR NUTZT DAS HTTP-PROTOKOLL VOLLSTÄNDIG.
2. HTTP-METHODEN:
   - GET
   - POST
   - PUT
   - DELETE
3. HTTP-STATUSCODES WERDEN IN FÜNF KLASSEN EINGETEILT UND ZEIGEN DEN ZUSTAND DER ANFRAGE AN.