Hätt‘ ich das früher gewusst

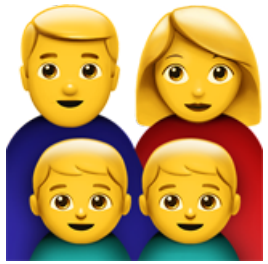
Good Practices bei API-Konzeption & -Entwicklung

Sven Hesse | JFS | 07.07.2022

DB Systel
Digital bewegen.
Gemeinsam.

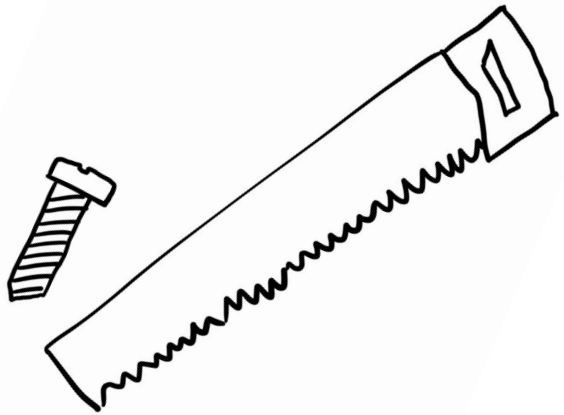
Hallo Welt,
ich bin Sven.





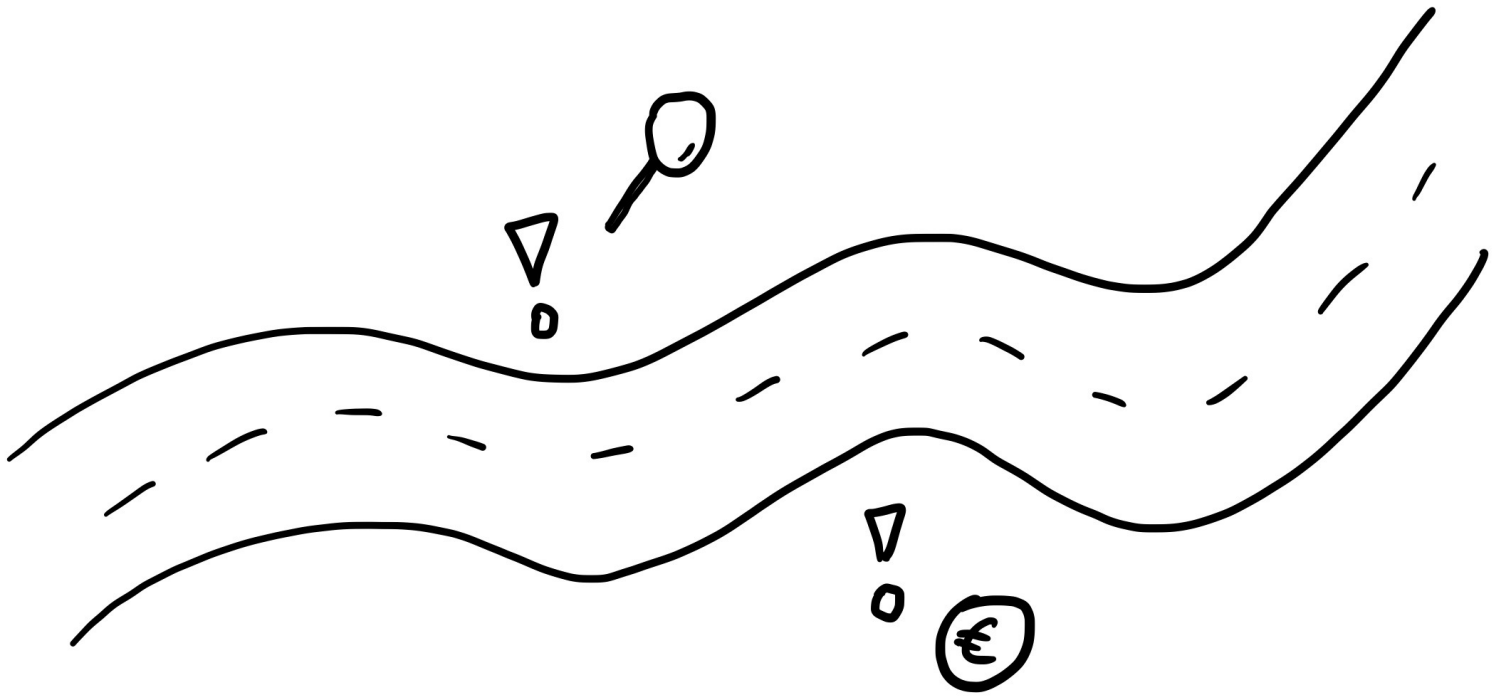

sven.sv.hesse@deutschebahn.com / kontakt@svenhesse.de
@dersvenhesse

API ist mehr
als Design und Technik.



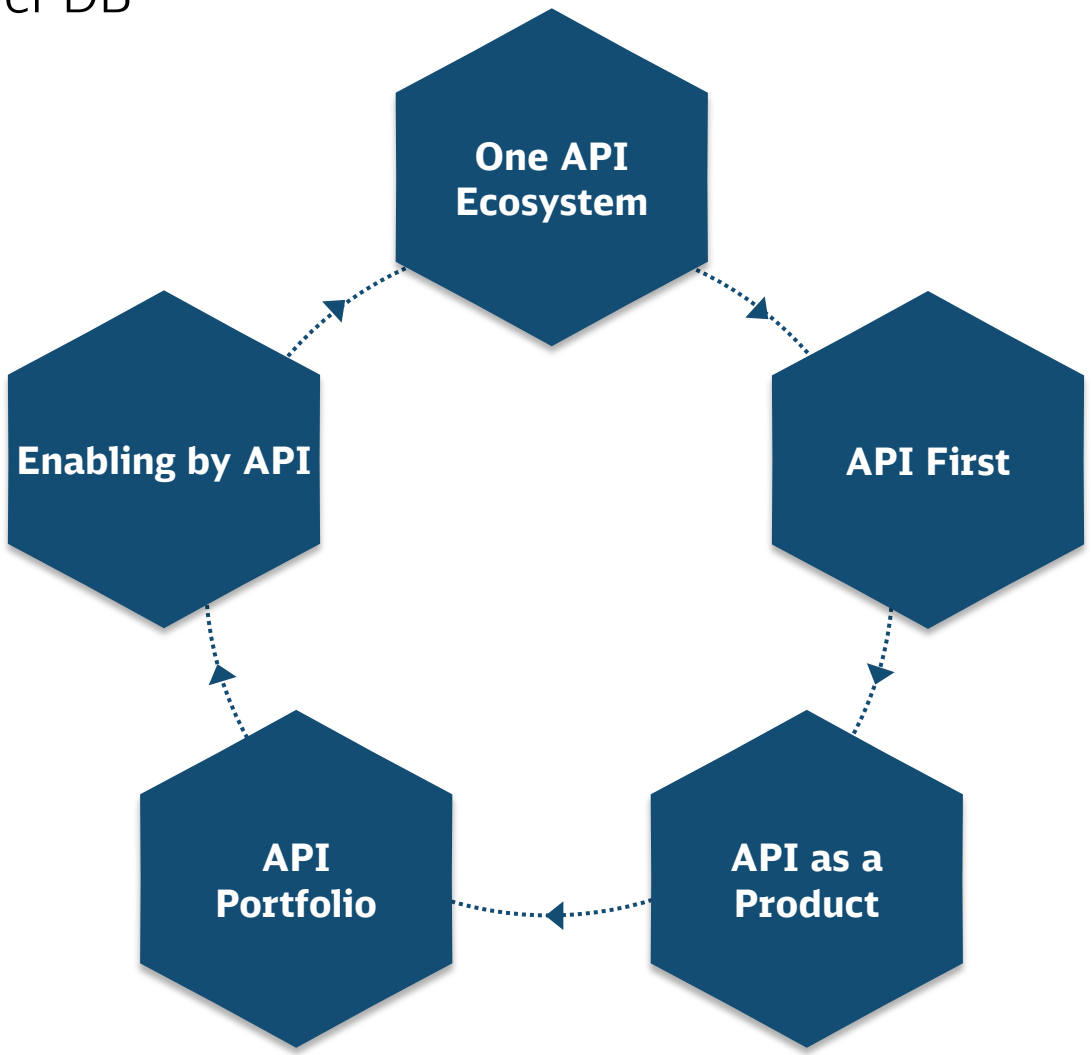
API ist mehr als Design und Technik.

Auffindbarkeit, Bestellbarkeit, Abrechenbarkeit beachten



API ist mehr als Design und Technik.

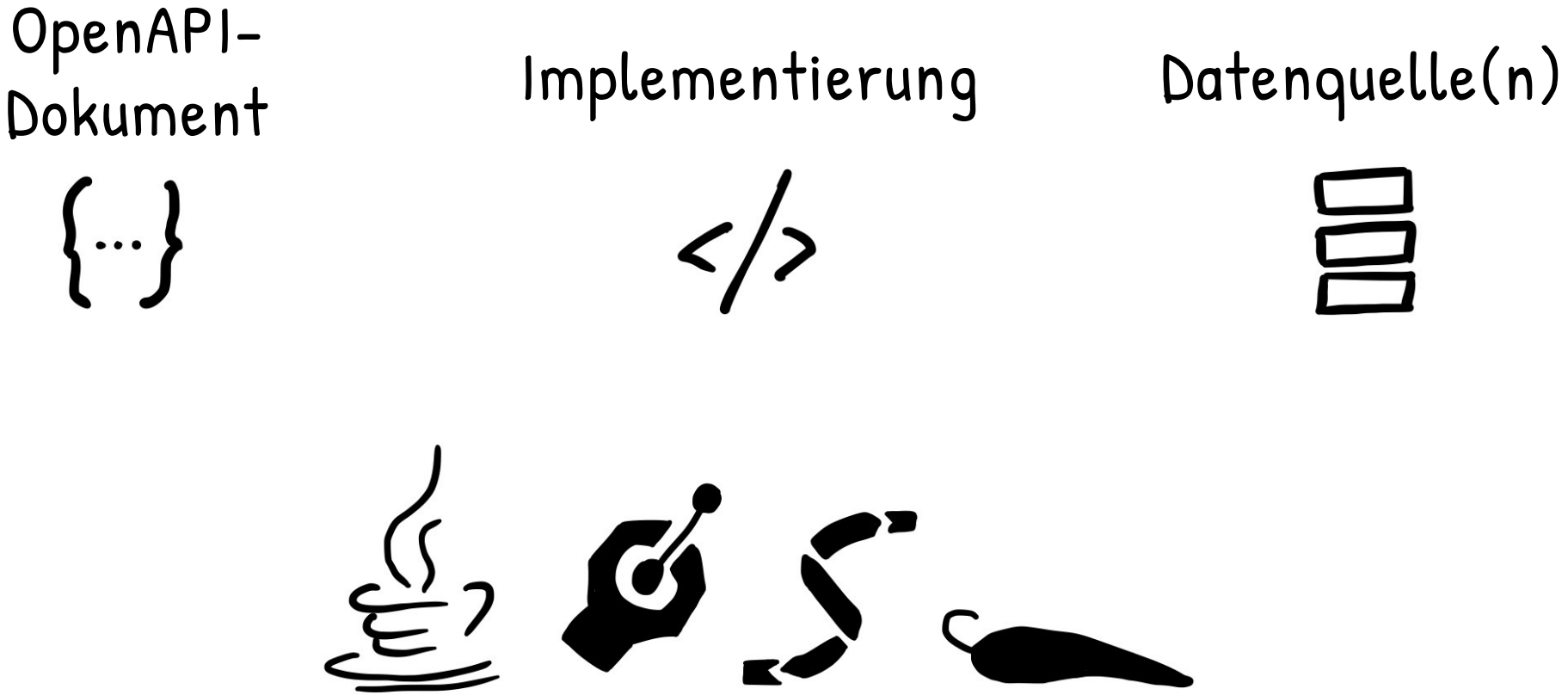
Eckpfeiler der API-Strategie der DB



API-first,
aber auch Datenquelle-first.

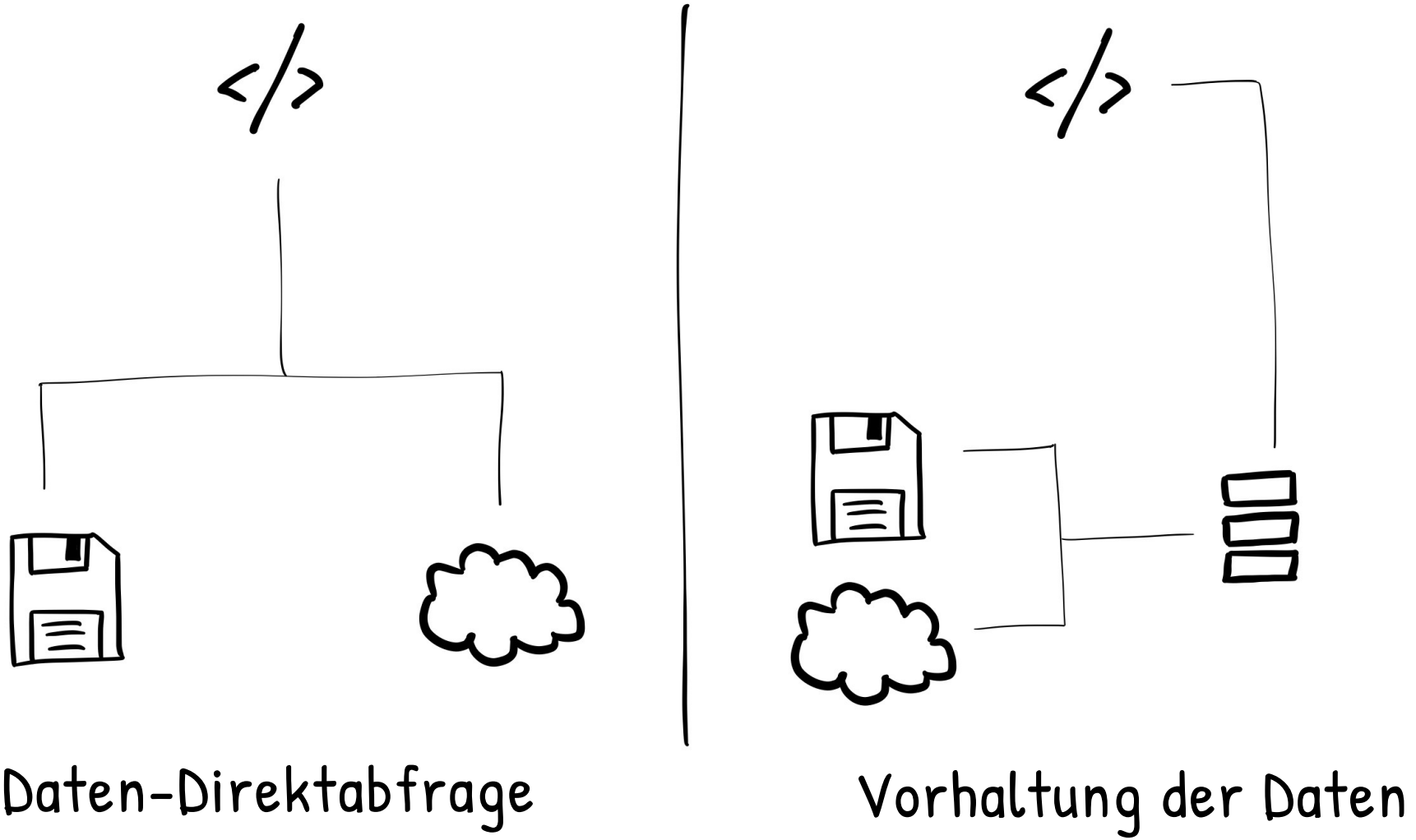
API-first,
aber auch Datenquelle-first.

Datenquelle(n) haben Einfluss auf Implementierung und API-Design

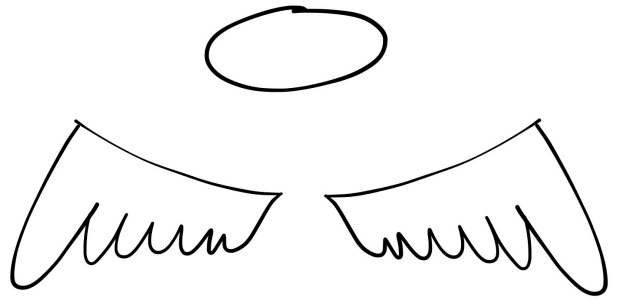


API-first,
aber auch Datenquelle-first.

Zusammenführung von Daten verschiedener Quellen

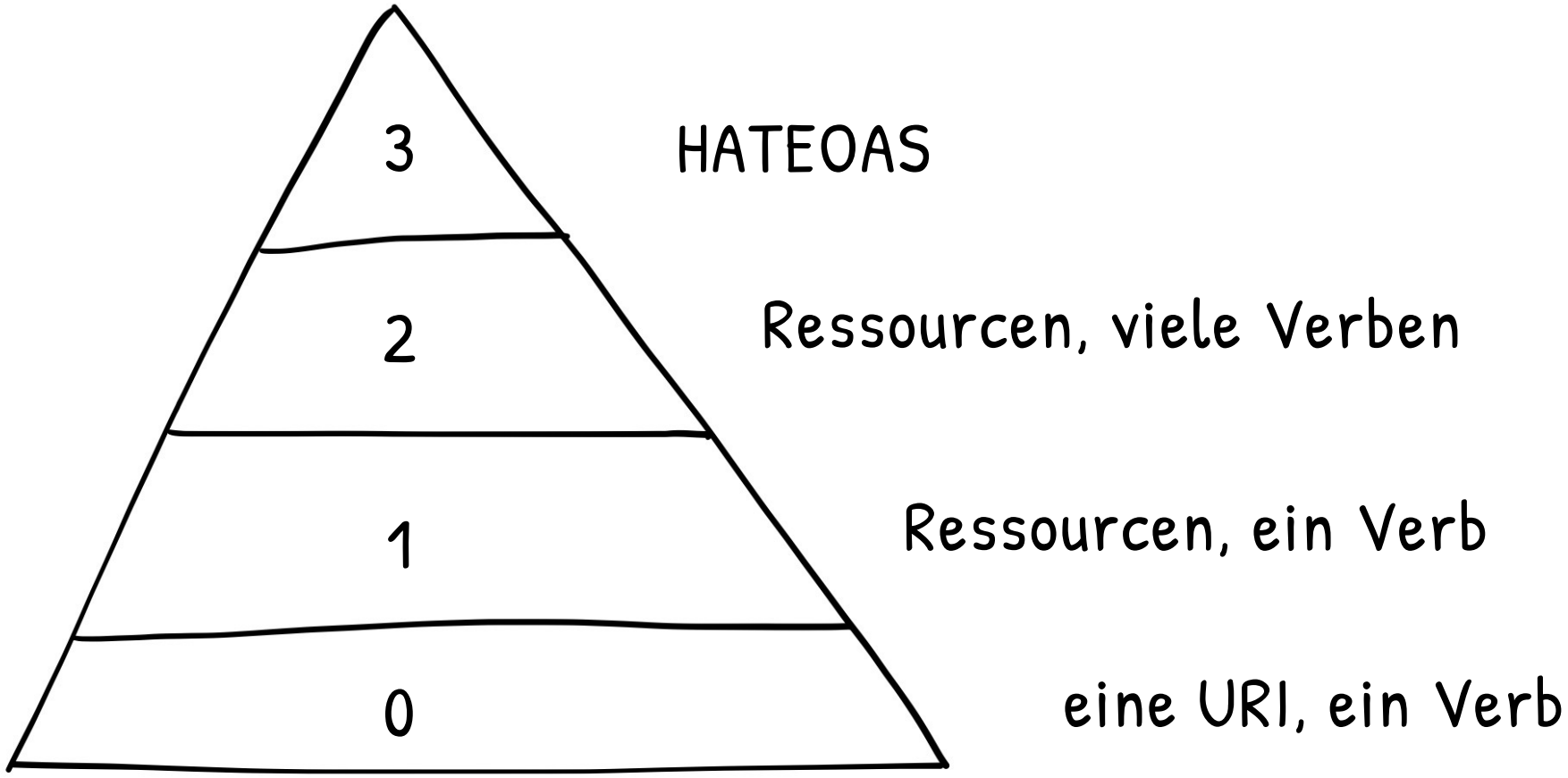


REST ist nicht heilig.

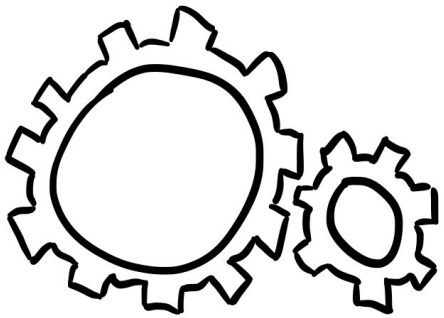


REST ist nicht heilig.

Richardson Maturity Model



Codegenerierung: jetzt.



Codegenerierung: jetzt.

Spezifikation

```yaml
openapi: 3.0.3
info:
    title: JFS-API
    version: '0.1'
servers:
    -   url: 'http://localhost:8080/jfs/v1'
paths:
    /speakers:
        get:
            tags:
                - Speaker
            responses:
                200:
                    description: Returns all persons speaking now.
                    content:
                        application/json:
                            schema:
                                type: array
                                items:
                                    $ref: '#/components/schemas/Person'
components:
    schemas:
        Person:
            required:
                - name
            type: object
            properties:
                name:
                    type: string
                    description: The full name of a person.
                    example: Erika Mustermann
```

Codegenerierung: jetzt.

Generator-Konfiguration

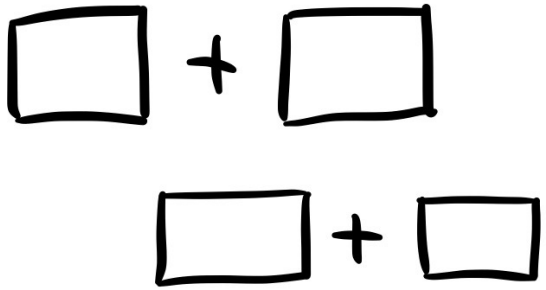
```xml
<plugin>
    <groupId>org.openapitools</groupId>
    <artifactId>openapi-generator-maven-plugin</artifactId>
    <version>${openapi-generator-maven-plugin.version}</version>
    <executions>
        <execution>
            <goals>
                <goal>generate</goal>
            </goals>
            <configuration>
                <inputSpec>${project.basedir}/docs/openapi.yaml</inputSpec>
                <generatorName>spring</generatorName
                <apiPackage>${project.groupId}.${project.name}.controller</apiPackage>
                <generateApis>true</generateApis
                <modelPackage>${project.groupId}.${project.name}.domain</modelPackage>
                <generateModels>true</generateModels>
                <strictSpec>true</strictSpec>
                <configOptions>
                    <dateLibrary>java8</dateLibrary>
                    <disableHtmlEscaping>true</disableHtmlEscaping>
                    <hateoas>false</hateoas>
                    <hideGenerationTimestamp>true</hideGenerationTimestamp>
                    <interfaceOnly>true</interfaceOnly>
                    <sourceFolder>models</sourceFolder>
                    <useTags>true</useTags>
                </configOptions>
            </configuration>
        </execution>
    </executions>
</plugin>
```

Codegenerierung: jetzt.

Implementierung mit generierten Klassen

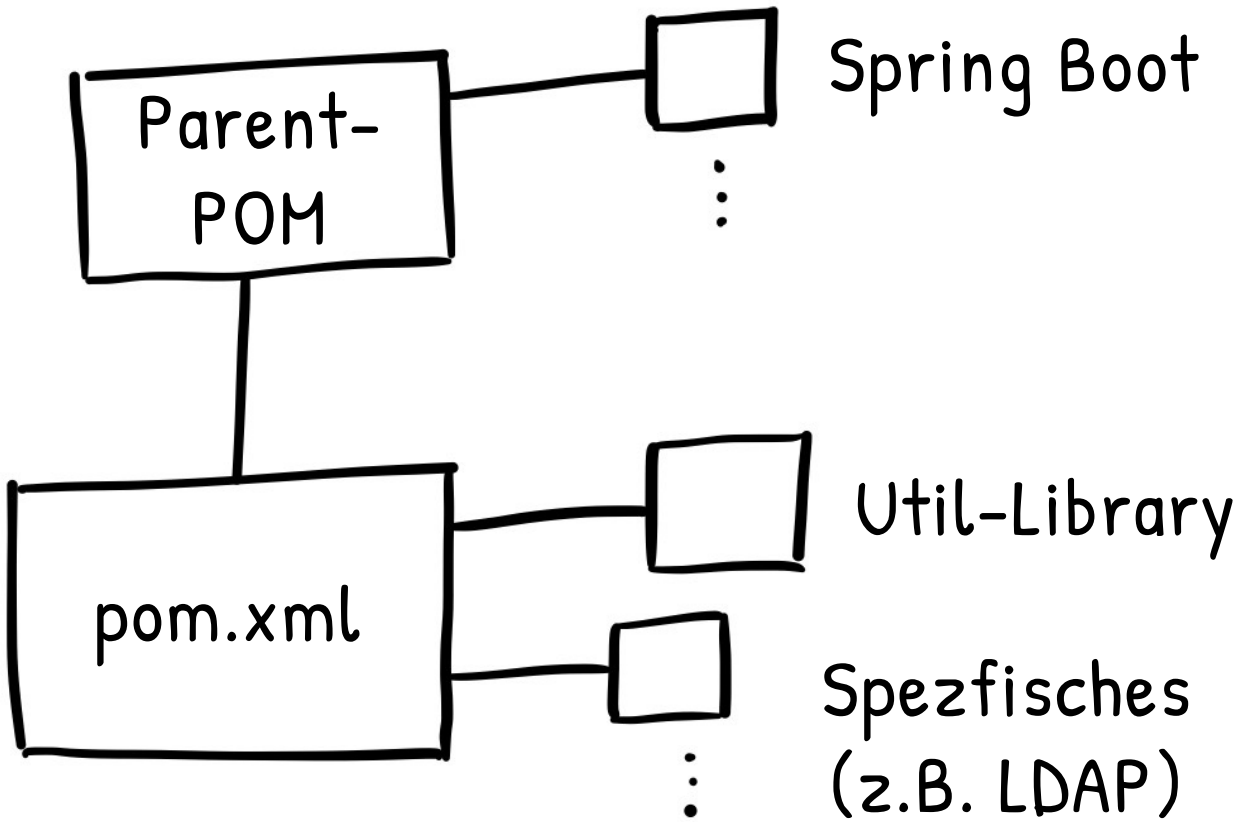
```java
package de.svenhesse.talk.jfs.controller;

import de.svenhesse.talk.jfs.domain.Person;
import java.util.List;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping(value = "/v1")
public class SpeakerController implements SpeakerApi {

    @Override
    public ResponseEntity<List<Person>> speakersGet() {
        return ResponseEntity.ok(
                List.of(
                        new Person().name("Stefan"),
                        new Person().name("Jörg"),
                        new Person().name("Bernd"),
                        new Person().name("Frank"),
                        new Person().name("Matúš"),
                        new Person().name("Hendrik"),
                        new Person().name("Sven")
                )
        );
    }

}
```

Bounded Context vs Copy & Paste

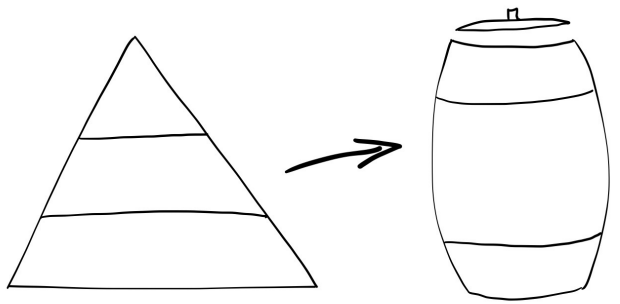


Bounded Context vs Copy & Paste

Auslagerung von Artefakten

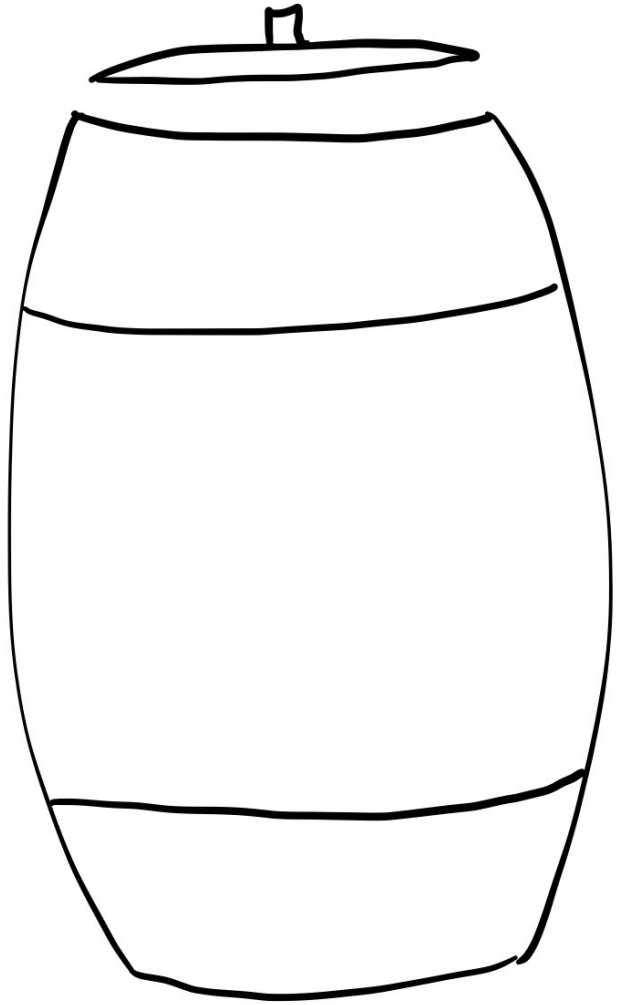


Tests sind für die Tonne.



Tests sind für die Tonne.

Fokus auf Integrationstests

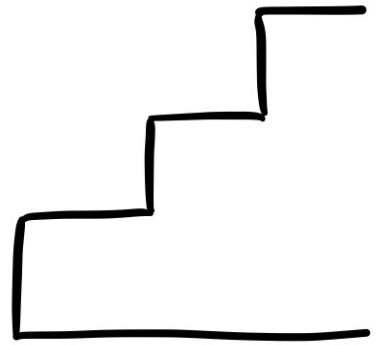


End-2-End-Tests mit Postman/Newman

Integrationstests mit Karate, WireMock

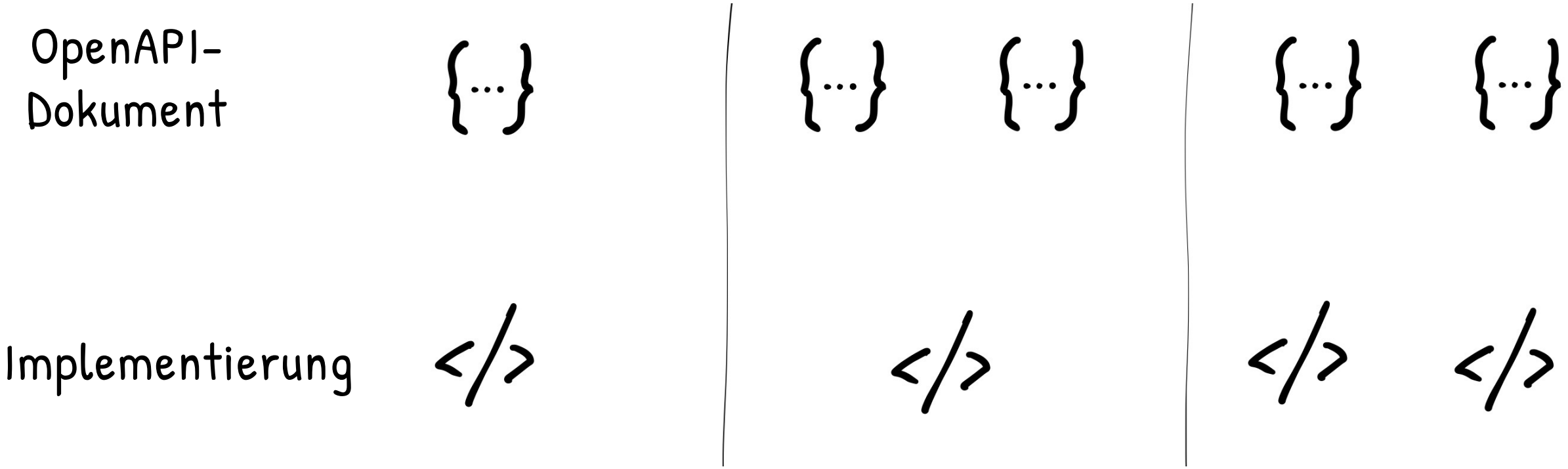
Unit-Tests mit JUnit

Keine Angst vor v2.



Keine Angst vor v2.

Umsetzungsmöglichkeiten



Sänk ju for listening.

sven.sv.hesse@deutschebahn.com / kontakt@svenhesse.de
@dersvenhesse

DB Systel
Digital bewegen.
Gemeinsam.