CROWDCODE

JAVAFORUM stuttgart

Vortrag von Ingo Düppe

OpenAPI Codegenerierung:

The Good, The Bad & The Ugly

WIR TRANFORMIEREN BROWNFIELD IN DIE ZUKUNFT!

SOFTWARE **WARTUNG**	SOFTWARE **MODERNISIERUNG**	SOFTWARE **ENTWICKLUNG**	**DIGITAL PACE** BERATUNG
ORACLE JAVA EXIT	**KI-** ENTWICKLUNG	DEVELOPMENT **ACCELERATION**	WISSENS- **TRANSFER**

AGENDA

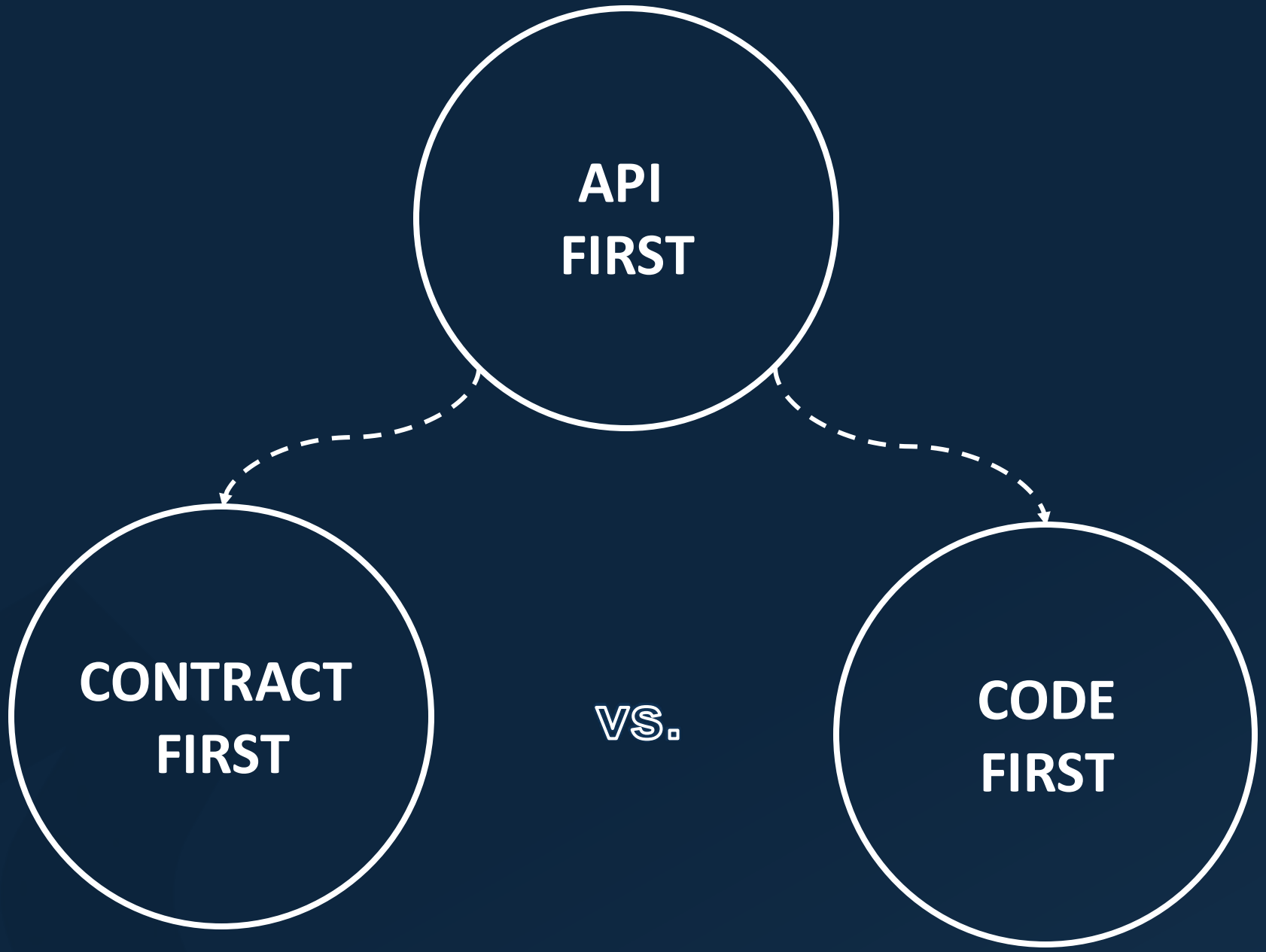
- Warum OpenAPI?
- Was läuft gut?
- Was läuft nicht rund?
- Was steht im Weg?

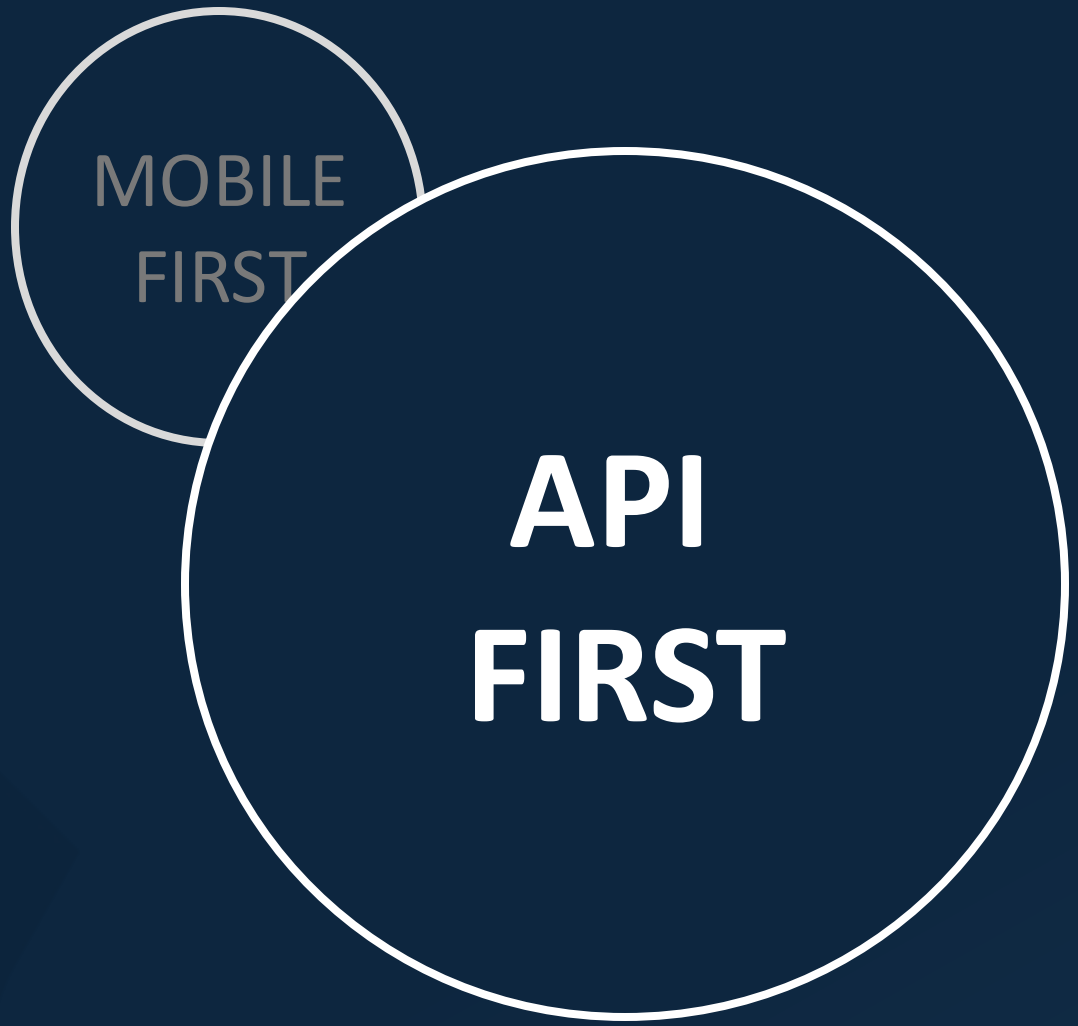
APIs are eating the World!

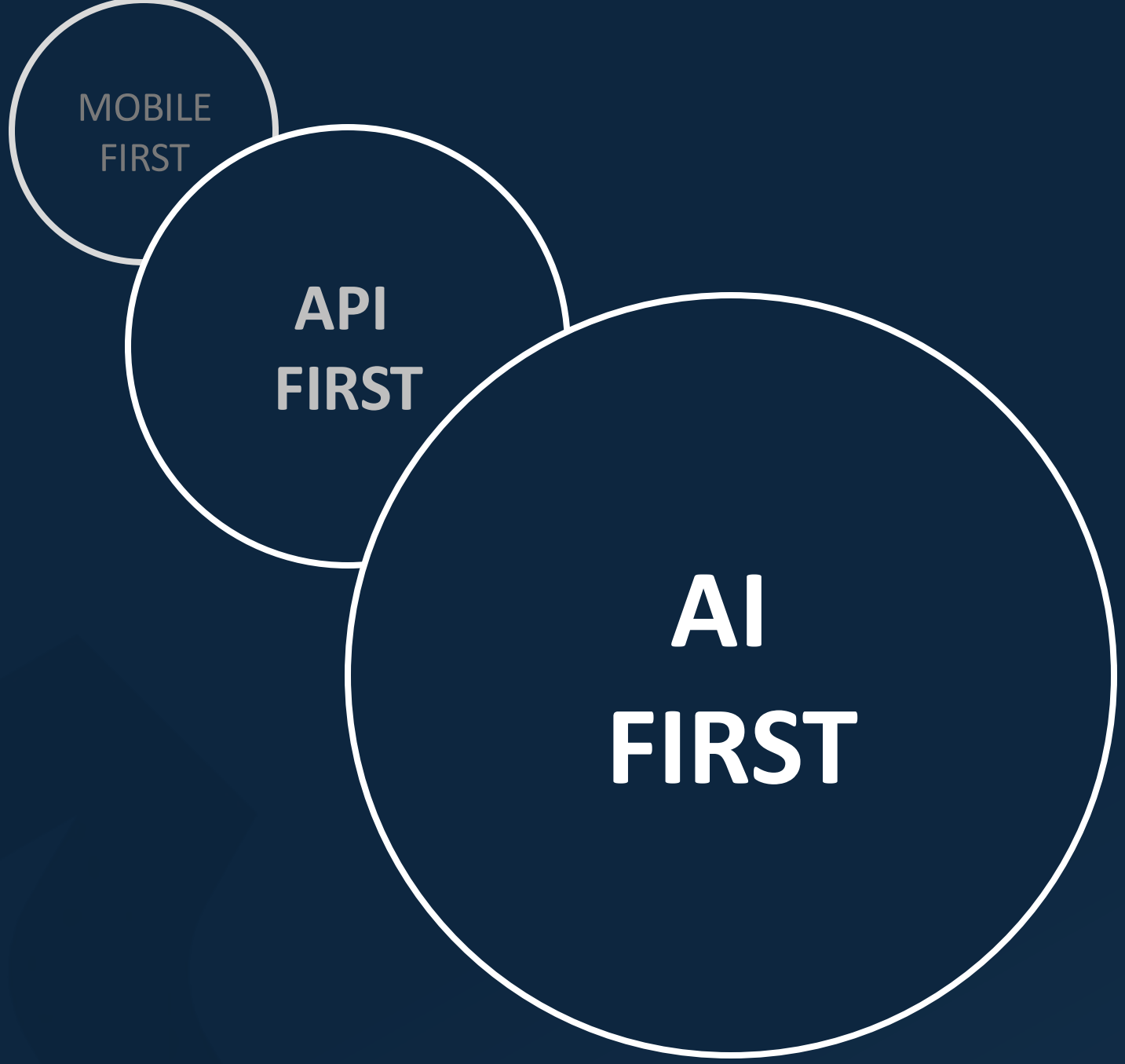
APIs SIND DER DIGITALE KOMMUNIKATIONS-KANAL VON UNTERNEHMEN IN DIGITALISIERTEN MÄRKTEN!

DAHER

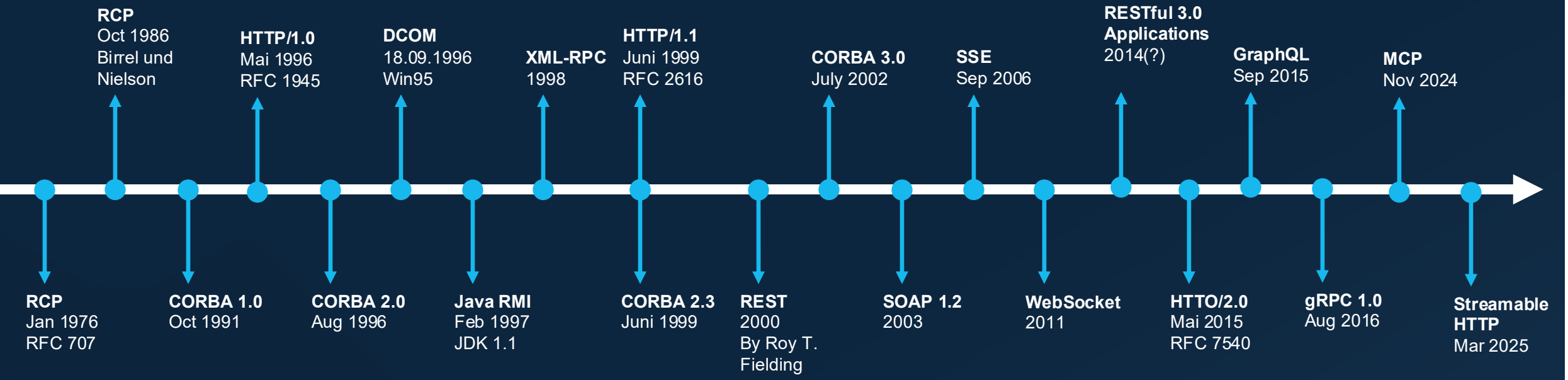
API-DESIGN MUSS IN DEN ENTWICKLUNGS-PROZESSEN STRATEGISCH INTEGRIERT SEIN


API
FIRST
CONTRACT
FIRST
vs.
CODE
FIRST

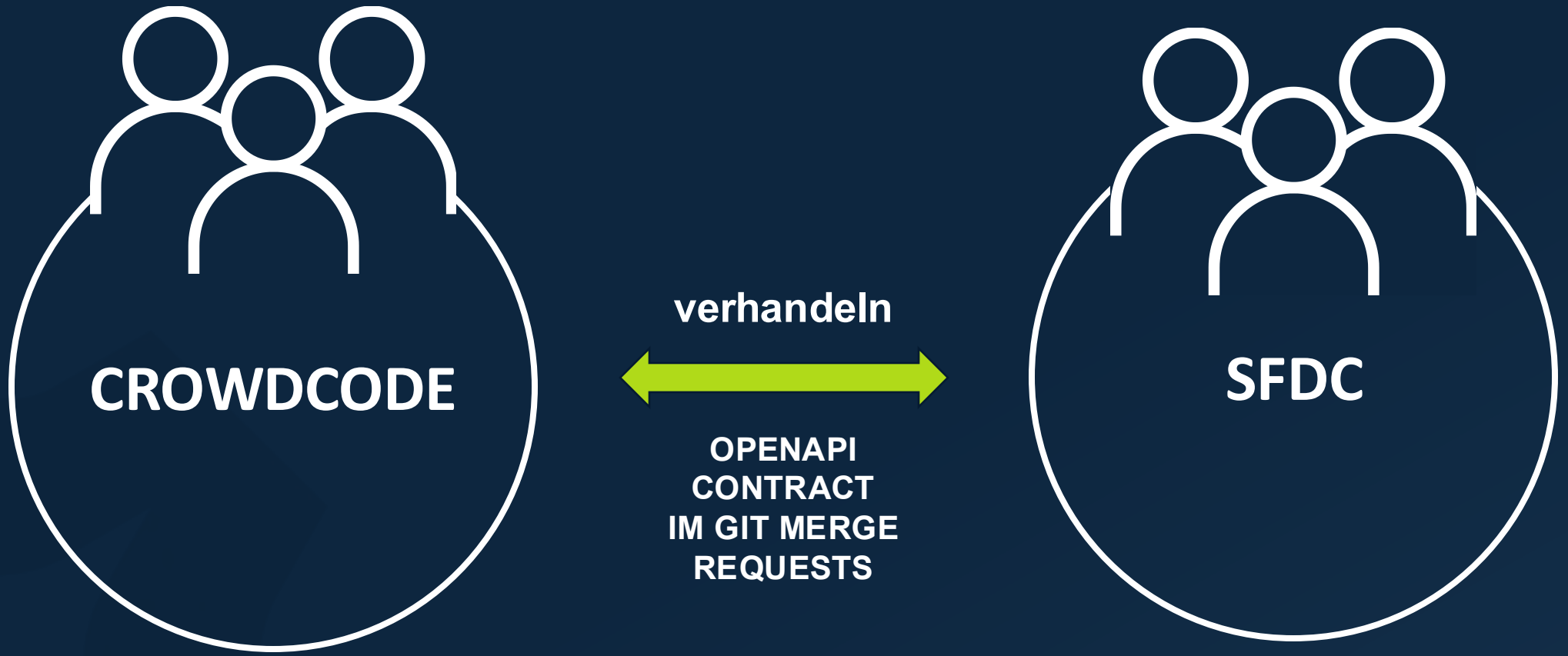

MOBILE
FIRST
API
FIRST


MOBILE
FIRST
API
FIRST
AI
FIRST

Historie der Inter-Process-Communication



CONTRACT FIRST



API STYLE GUIDELINES SIND ESSENZIELL UND SOLLTEN IN DEN TEAMS ETABLIERT SEIN!

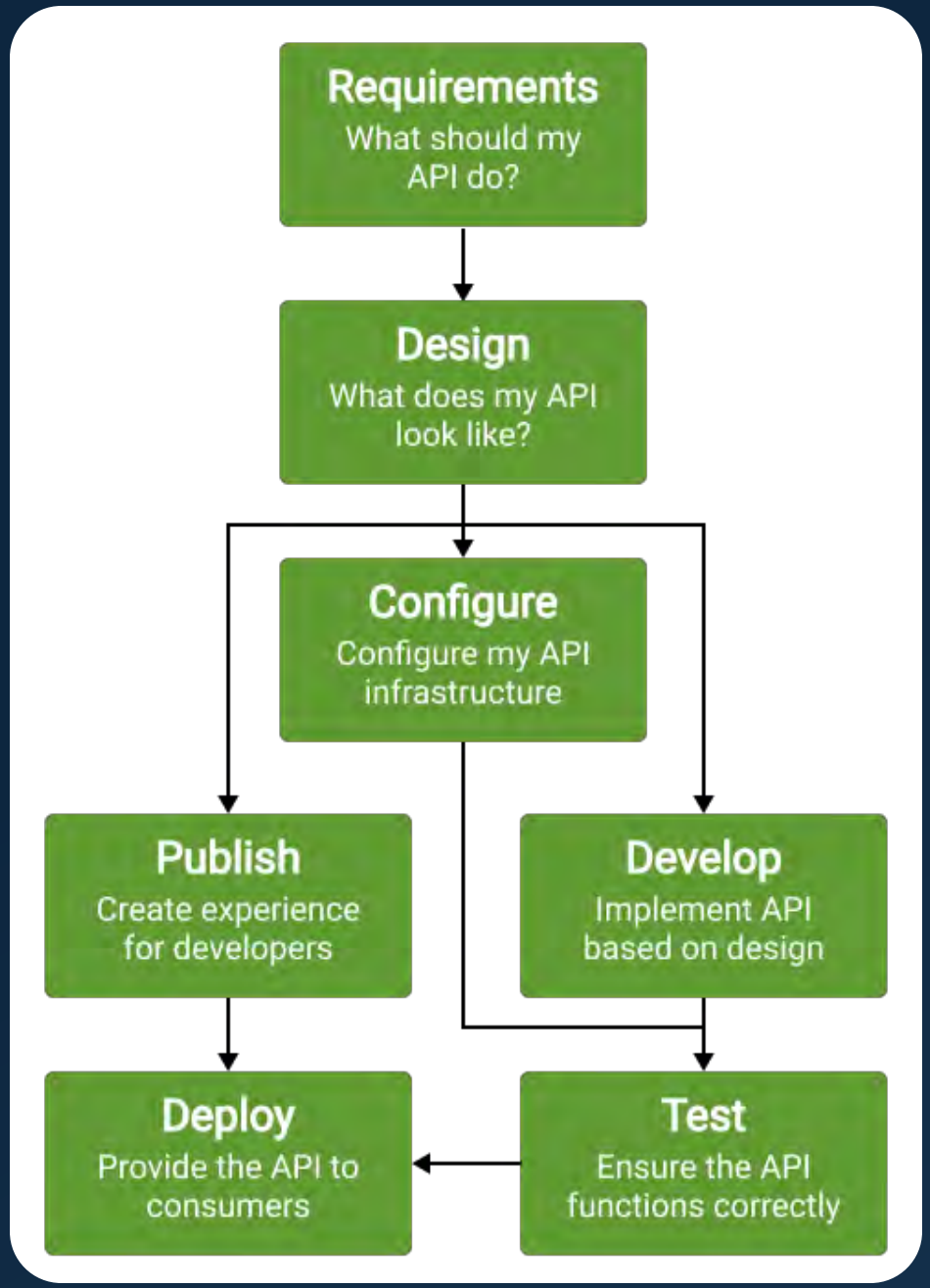
ERIC WILDE HAT 28 API-GUIDELINES ZUSAMMENGETRAGEN!

https://dret.github.io/guidelines/

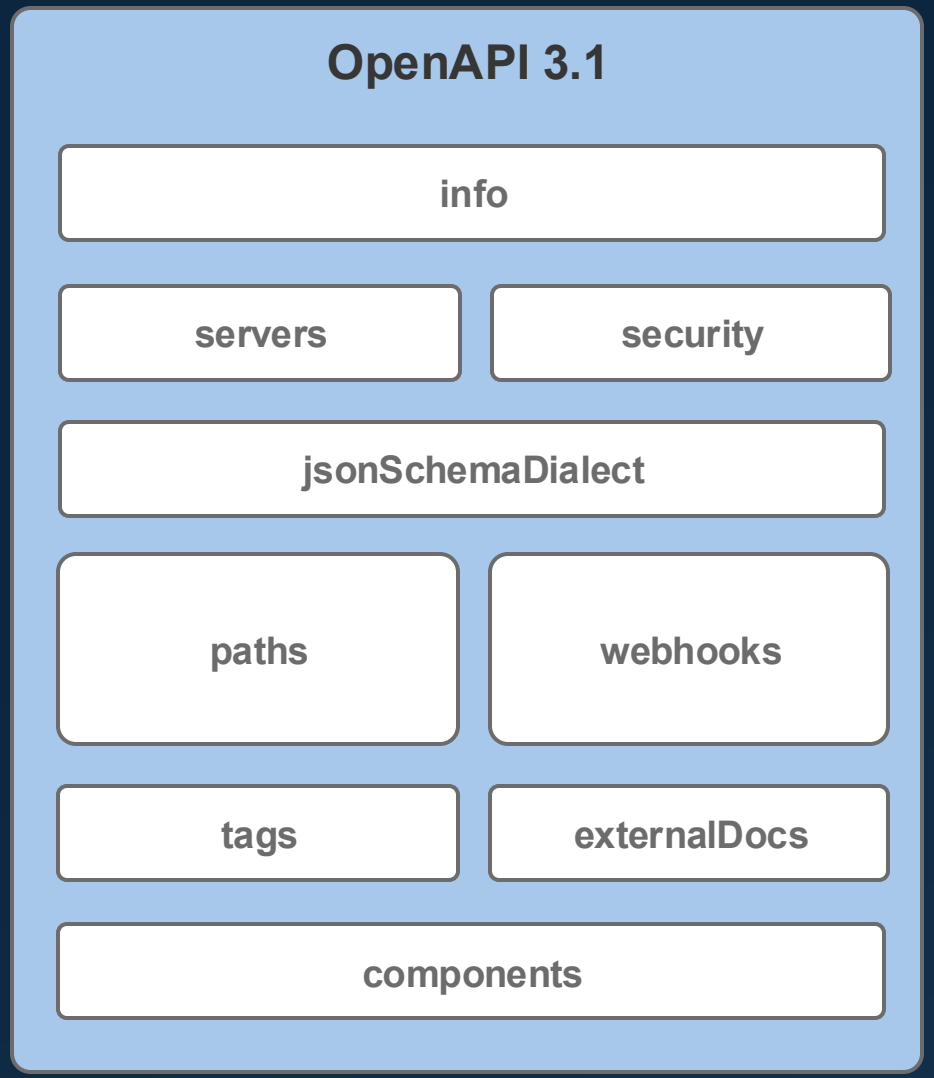
CONTRACT FIRST
BEGINNT, WENN DER CONTRACT ALS
CODE-ARTEFACT
IM BUILD-PROZESS VOLLSTÄNDIG INTEGRIERT IST!


OPENAPI
INITIATIVE

DIE OPENAPI-SPEZIFIKATIONEN BIETEN EINEN FORMALEN STANDARD FÜR DIE BESCHREIBUNG VON HTTP-APIS.



DIE AKTUELLE SPEZIFIKATION IST OPENAPI V3.1.1



DIE INTEGRATION IN DEN BUILDPROZESS ERFOLGT ÜBER OPENAPI GENERATOR MIT 162 GENERATOREN FÜR 83 PROGRAMMIER-SPRACHEN

CONTRACT FIRST MUSS IN DER BUILD UND DEVELOPMENT PIPELINE / PROZESSEN FRIKTIONSFREI INTEGRIERT SEIN.

DRY

DON‘T REPEAT YOURSELF

SOC

SEPARATION OF CONCERNS

COMPONENT BASED DEVELOPMENT

DISCLAIMER

THE GOOD

Wunderbar, danke dass wir es nutzen dürfen!

THE BAD

Wunderbar, wir können Open Source mit etwas Zutun nutzen!

THE UGLY

Wunderbar, die Community braucht unsere Hilfe!

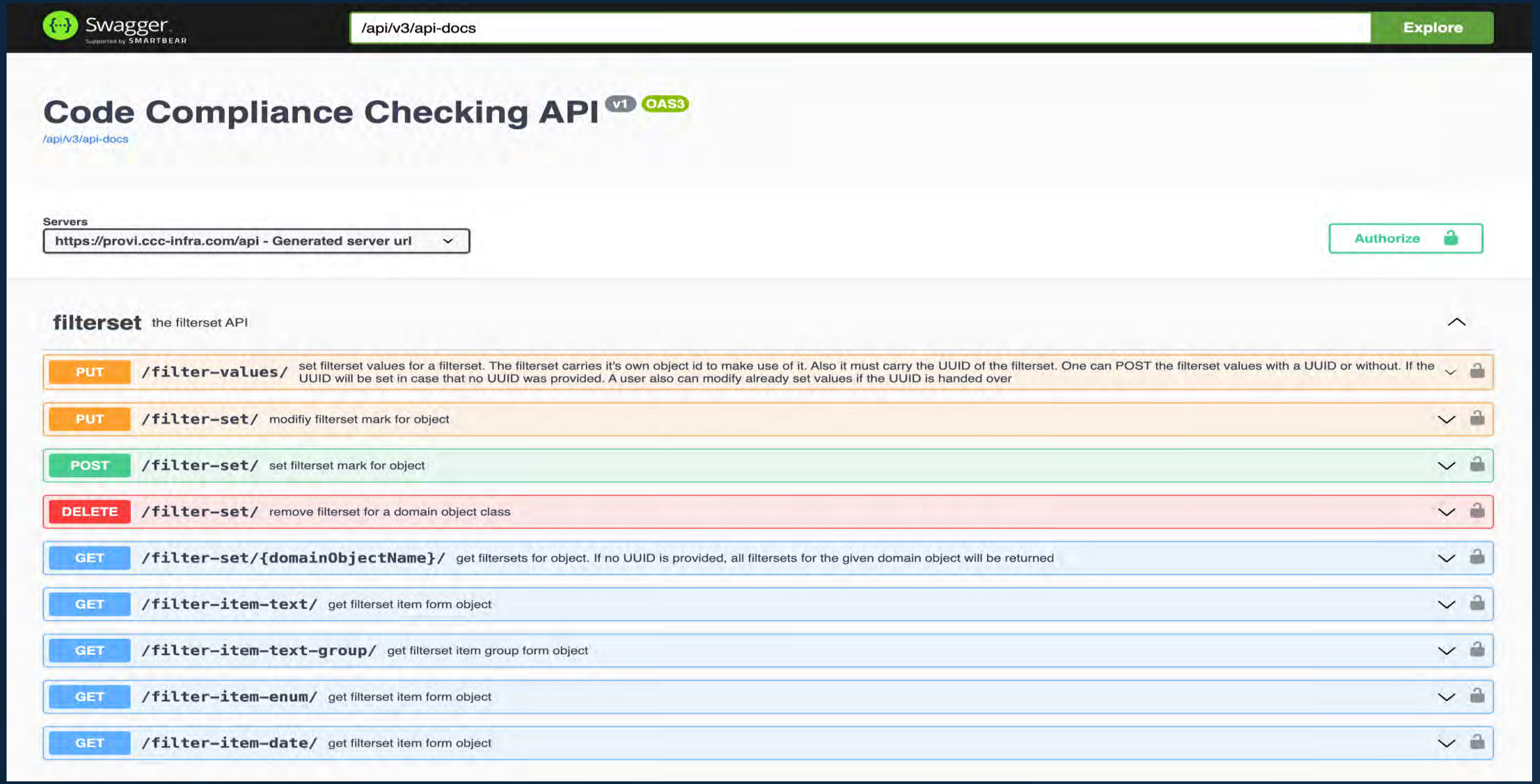
THE GOOD

FREIHEIT IM FORMAT

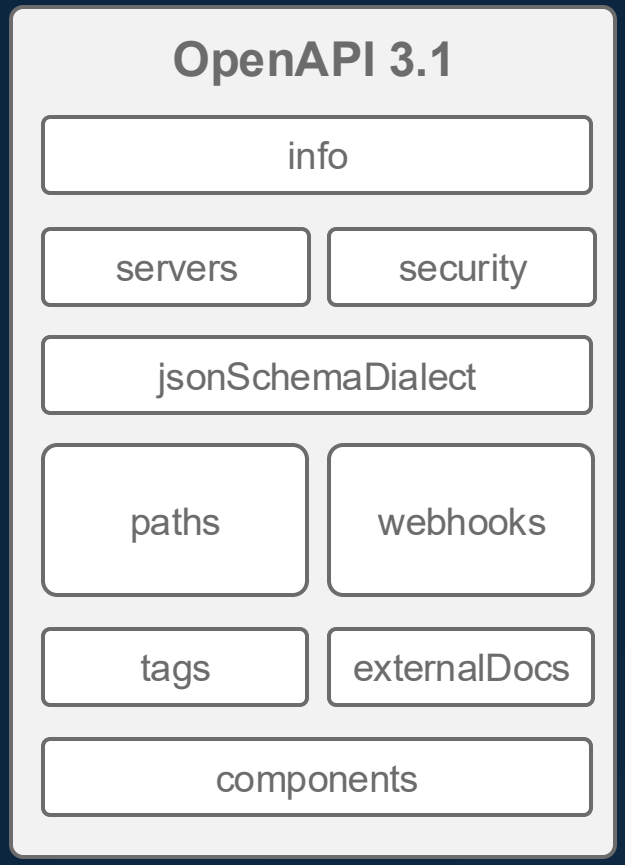

```yaml
servers:
- url: https://development.gigantic-server.com/v1
  description: Development server
- url: https://staging.gigantic-server.com/v1
  description: Staging server
- url: https://api.gigantic-server.com/v1
  description: Production server
```

```json
{ "servers":
 [
   {
     "url": "https://development.gigantic-server.com/v1",
     "description": "Development server" },
   { "url": "https://staging.gigantic-server.com/v1",
     "description": "Staging server" },
   { "url": "https://api.gigantic-server.com/v1",
     "description": "Production server" }
 ]
}
```

UI-Testing & -Dokumentation

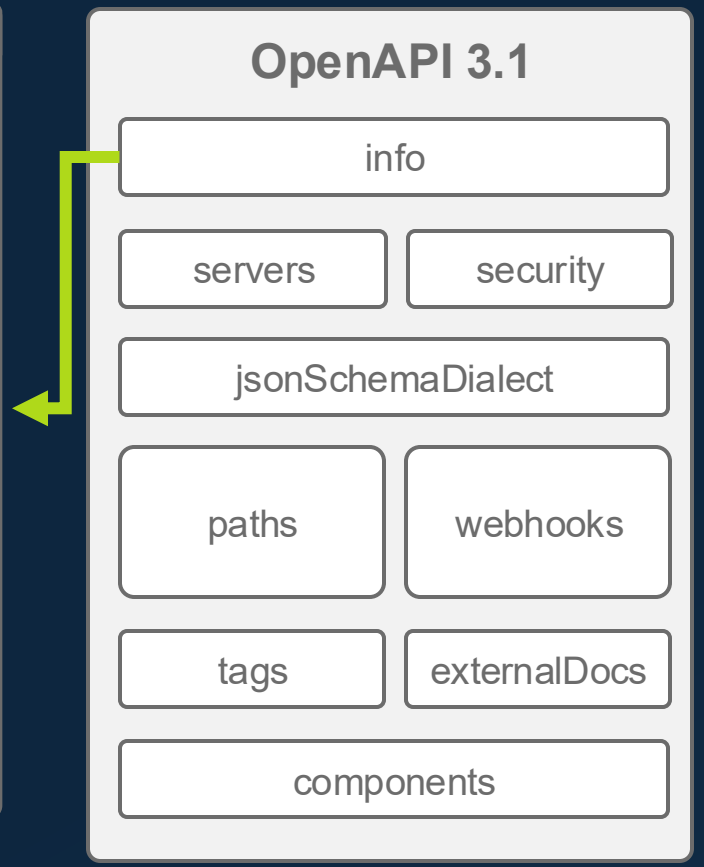


OPENAPI 3.1 Struktur



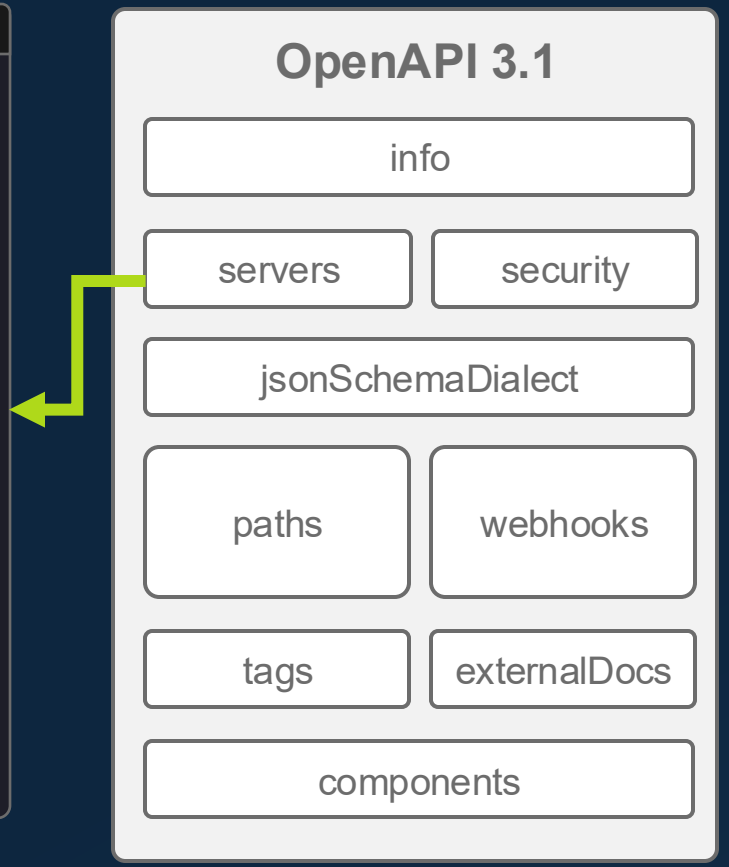
OPENAPI 3.1 Struktur

```yaml
openapi: 3.1.1
info:
  version: 1.0.0
  title: "Code Compliance Checker Service Api"
  description: |
    The Code Compliance Checker API designed to
    help users manage various infrastructure
    architecture planning...
  contact:
    name: CROWDCODE GmbH & Co. KG
    url: "https://www.crowdcode.io/"
    email: support@crowdcode.io
  x-audience: external-partner
  x-api-id: e28a9281-7ffe-43d1-ba17-4386e745dc5d
```

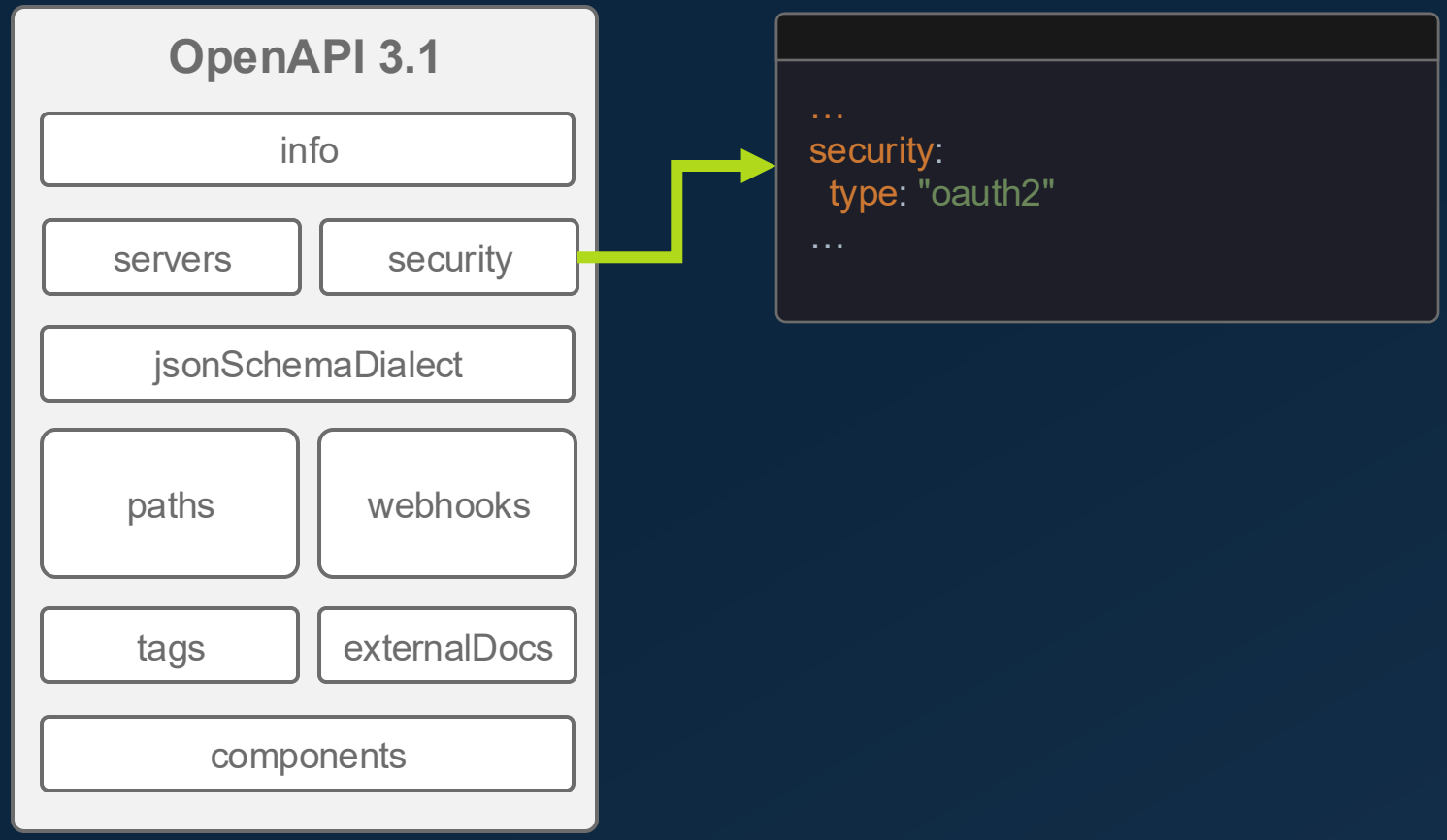


OPENAPI 3.1 Struktur

```yaml
…
servers:
  - url: https://{customer}-{stage}.crowdcode.io/api
    description: Generated server url
    variables:
      customer:
        default: demo
        descriptions: Shortcut of the customer
      stage:
        default: dev
        enum:
          - 'dev'
          - 'test'
          - 'acc'
…
```



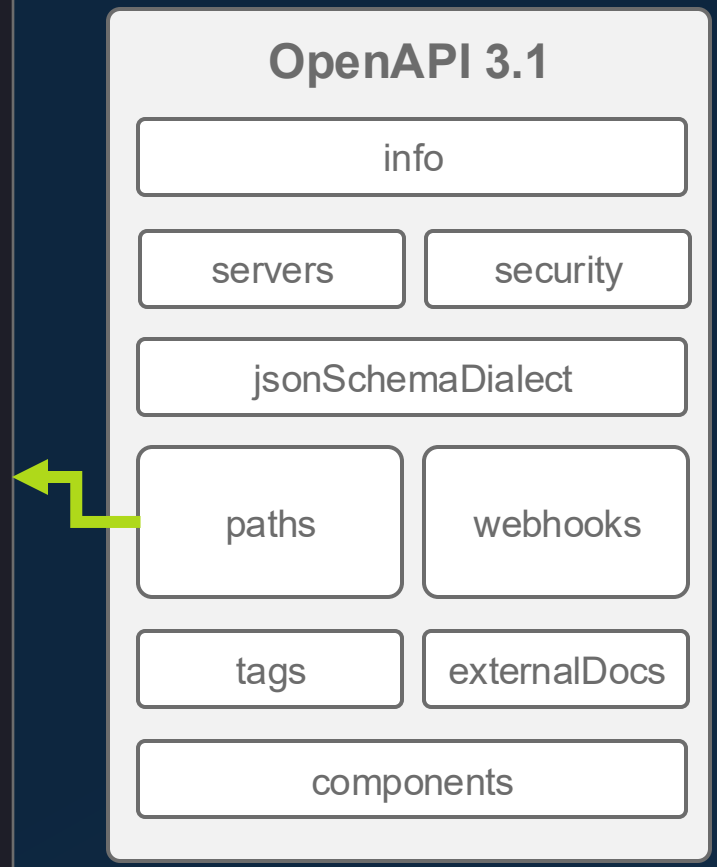
OPENAPI 3.1 Struktur



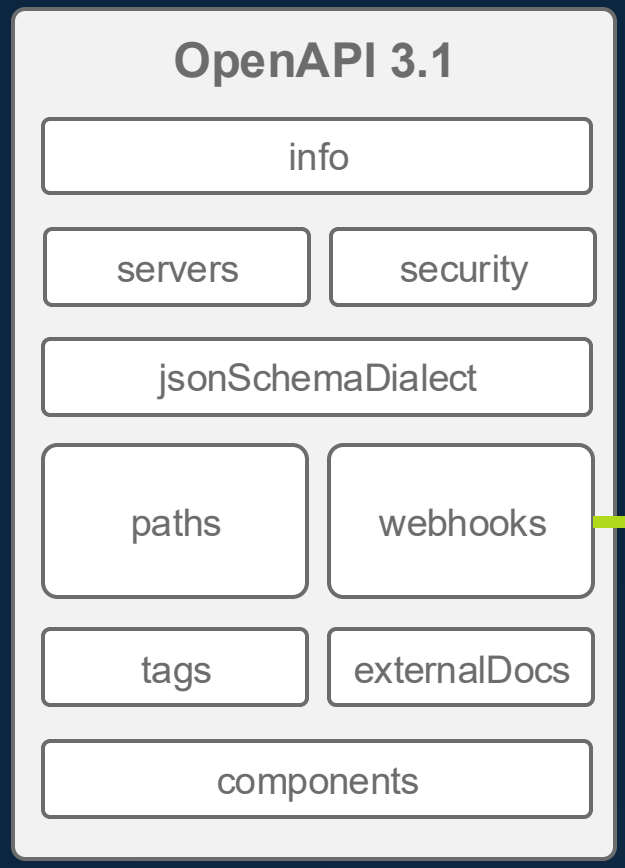
OPENAPI 3.1 Struktur

OPENAPI 3.1 Struktur

```yaml
/projects/{uuid}:
  get:a
    tags:
      - Project
    summary: Get project by uuid
    description: Get project by uuid
    operationId: getProject
    parameters:
      - name: uuid
        in: path
        description: Uuid of model
        required: true
        schema:
          type: string
          format: uuid
    responses:
      '200':
        description: Project response
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Project'
      default:
        description: Unexpected error
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ProblemDetail'
```



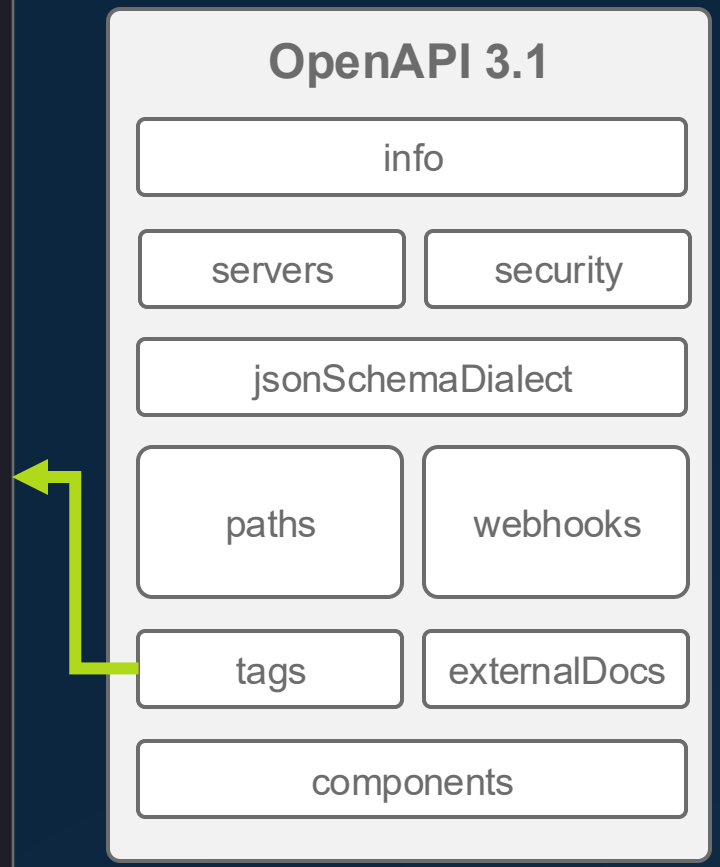
OPENAPI 3.1 Struktur



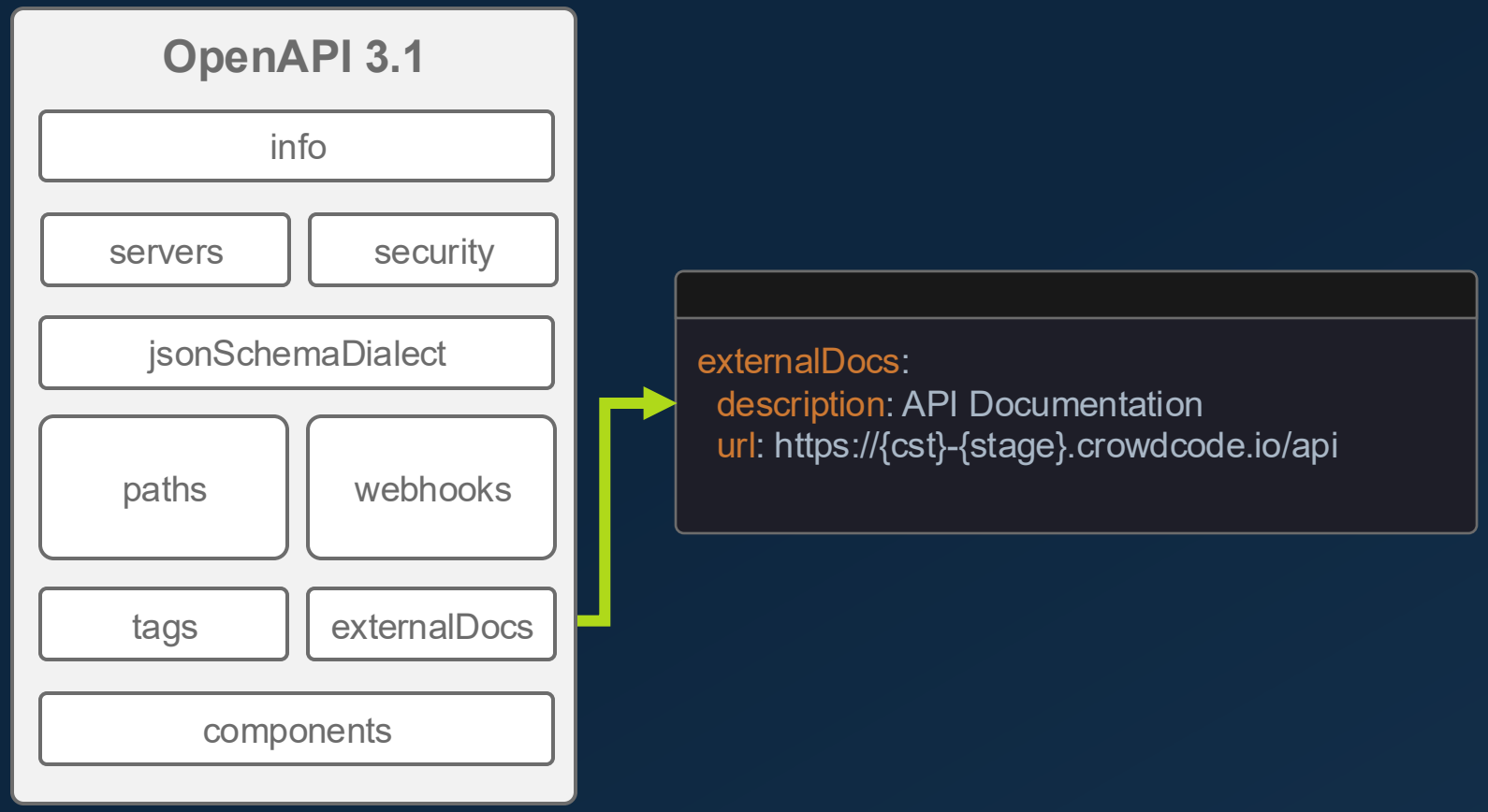

```yaml
webhooks:
  projectChanged:
    post:
      requestBody:
        description: |
          Information of the current project state.
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/Project"
      responses:
        "202":
          description: |
            Return a 202 status to indicate that
            the data was received successfully
```

OPENAPI 3.1 Struktur

```
...
tags:
  - name: filterset
    description: the filterset API
  - name: RuleConfig
    description: the RuleConfig API
  - name: CheckResult
    description: the CheckResult API
  - name: Bcf
    description: the Bcf API
  - name: TestRun
    description: the TestRun API
  - name: RuleSet
    description: the RuleSet API
  - name: Favorite
    description: the Favorite API
...
```

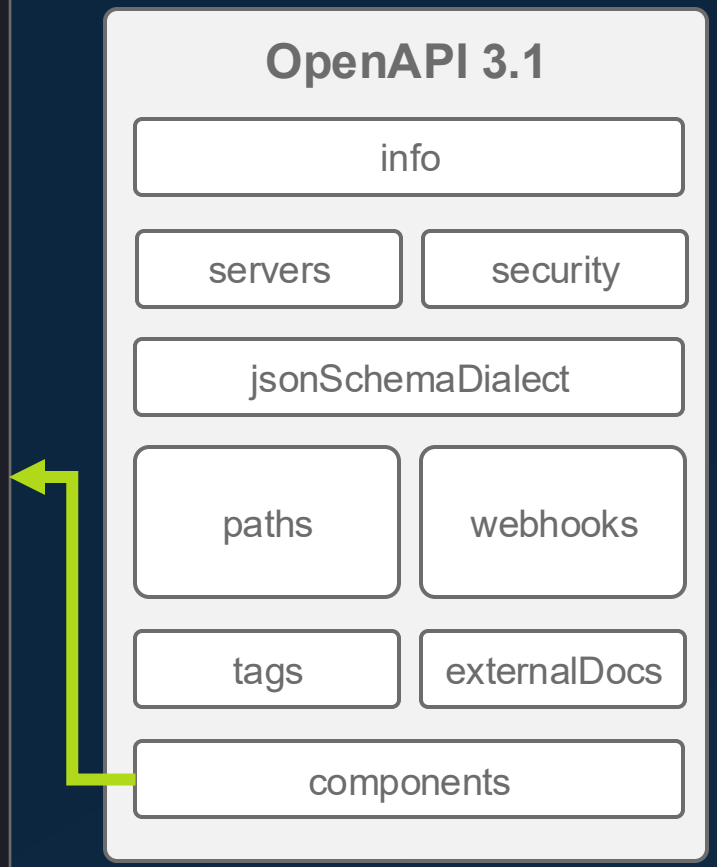


OPENAPI 3.1 Struktur

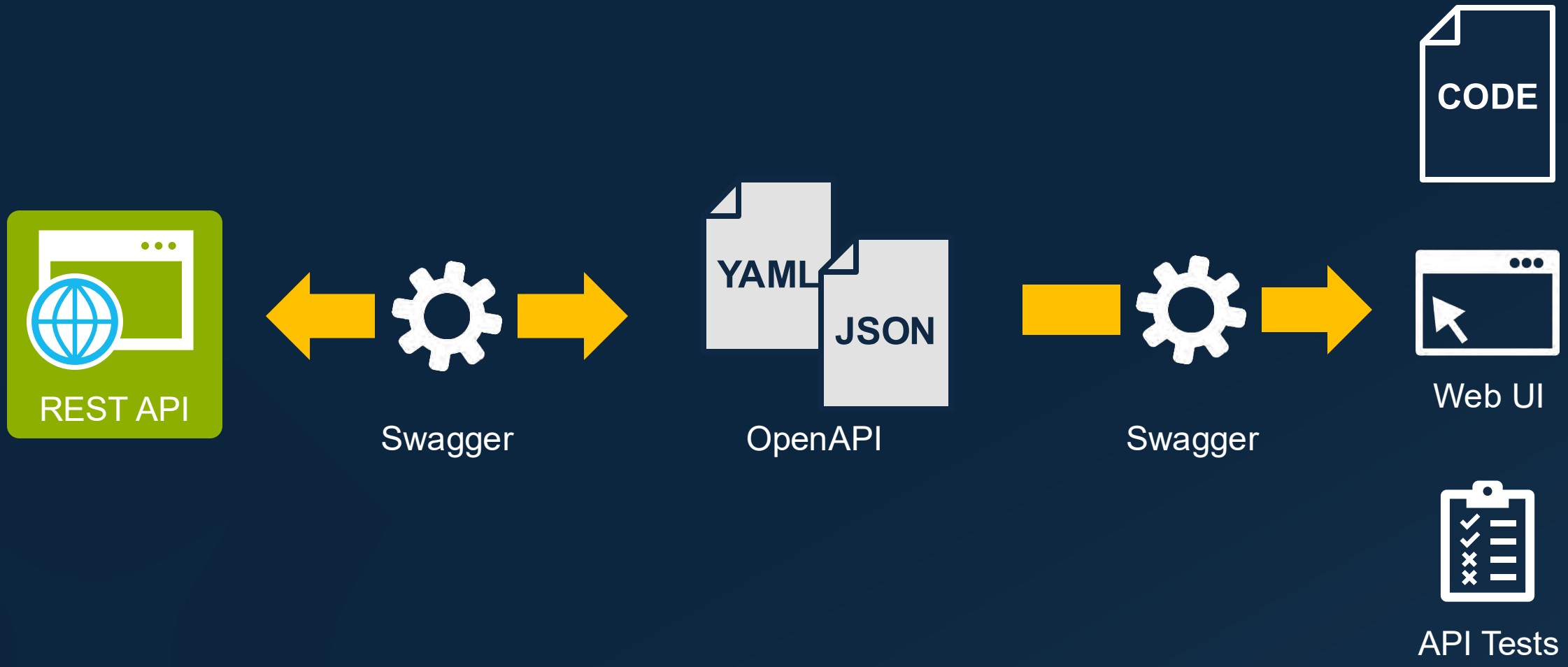


OPENAPI 3.1 Struktur

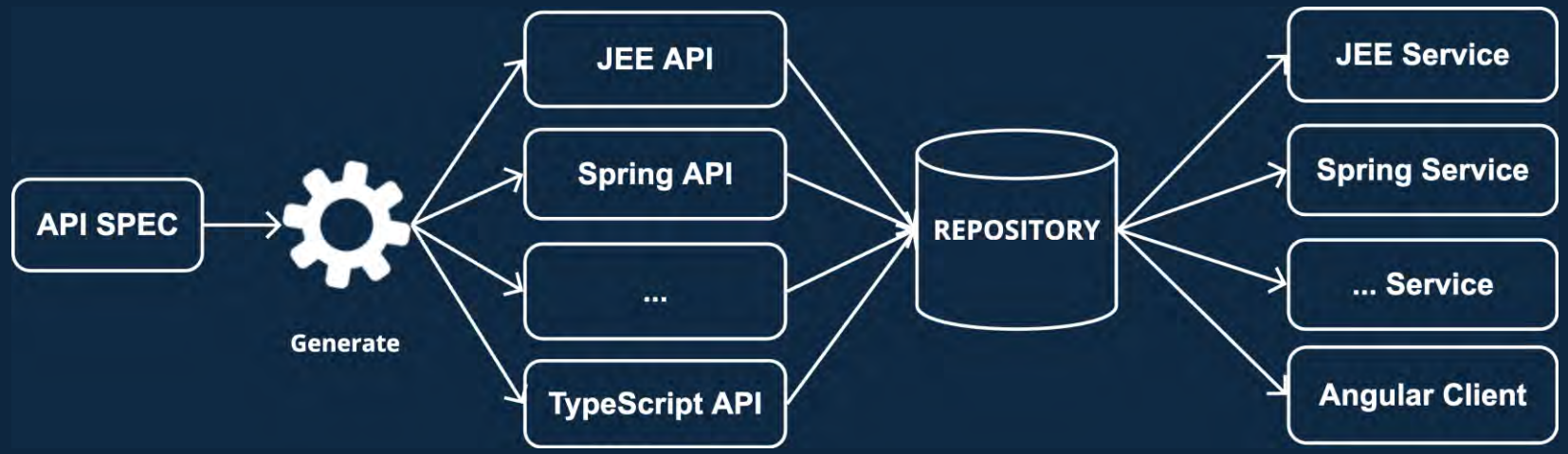
```yaml
…
components:
  schemas:
    FilterConfig:
      type: object
      properties:
        attributeGroup:
          type: array
          items:
            $ref: "#/components/schemas/AttributeGroup"
    AttributeGroup:
      type: object
      properties:
        key:
          type: string
          description: "Key of the attribute group"
        label:
          type: string
          description: "Label of the attribute group"
        selectType:
          type: string
          enum:
            - SINGLE
            - MULTIPLE
        attributes:
          type: array
          items:
            $ref: "#/components/schemas/AttributeValue“
```



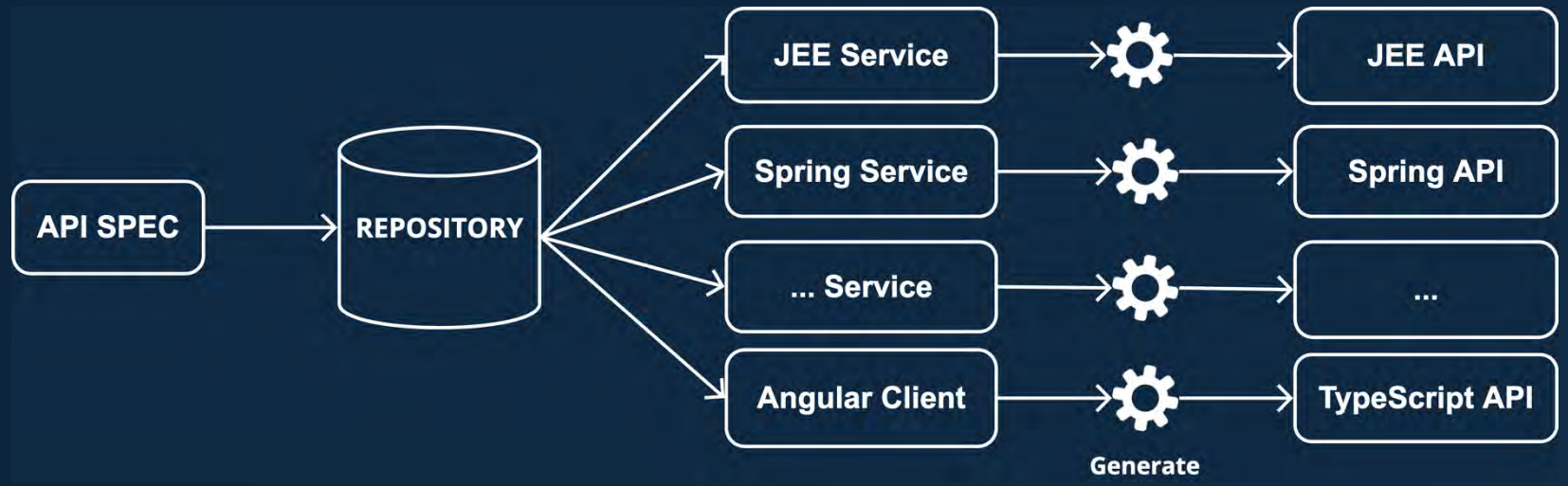
OpenAPI Code-Generierung



Wann wird was generiert?



VS.



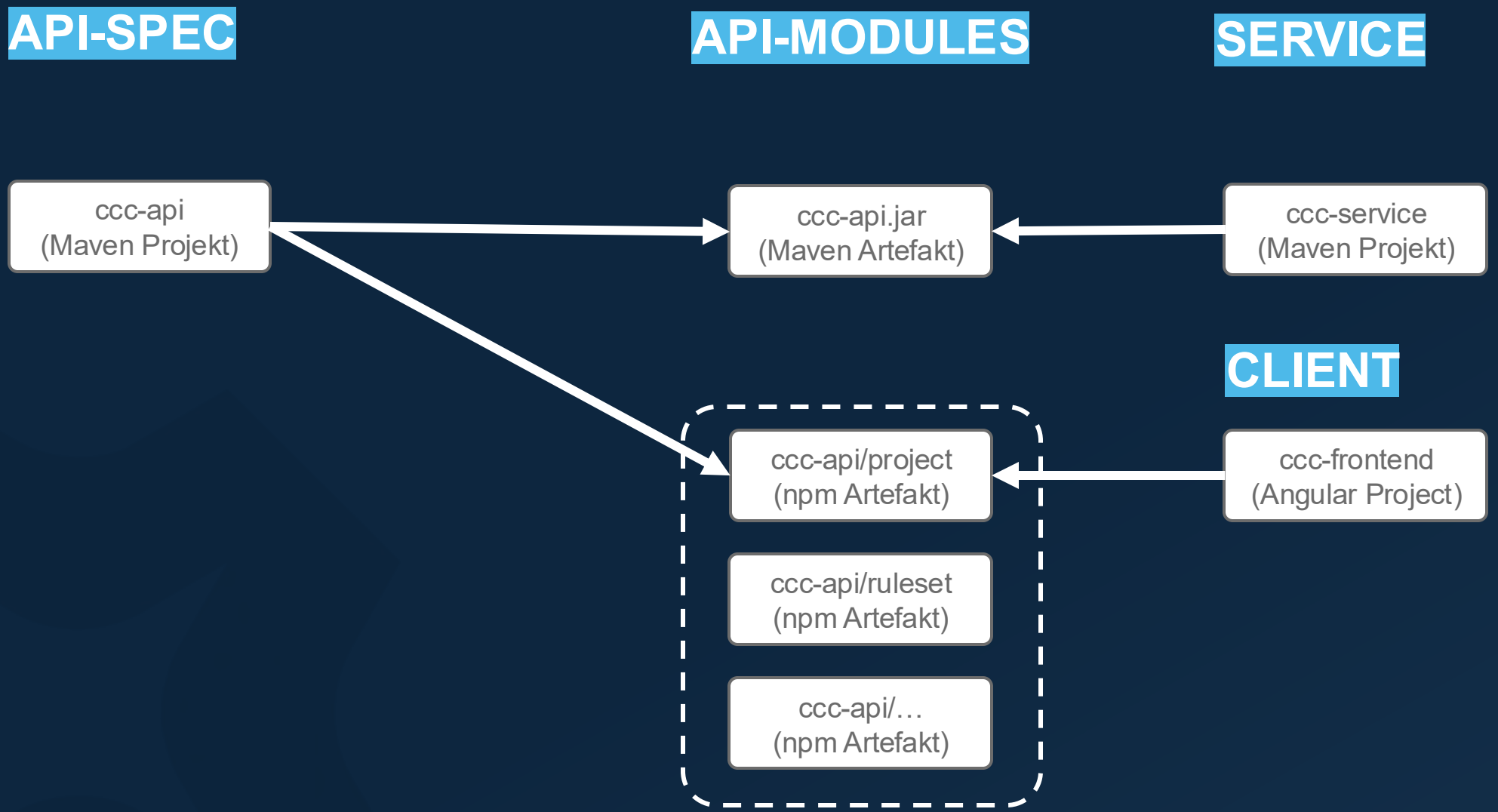
API-MODUL-ANSATZ

API-Projekt generiert die Client-Stubs und Server-Skeletons, welche die Server- & Client-Projekte importieren.

API-ONLY-ANSATZ

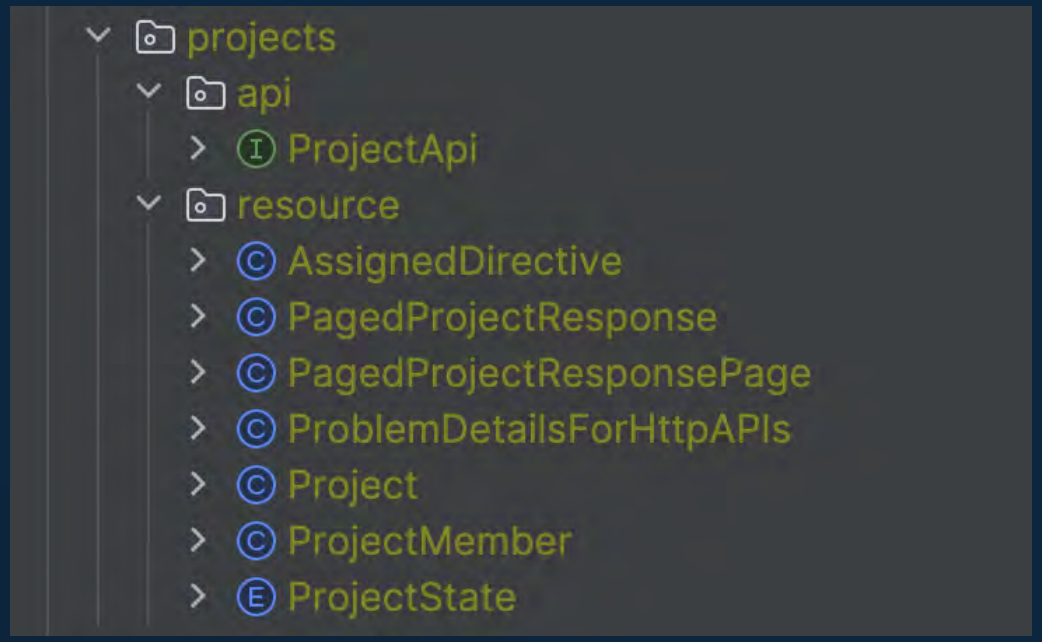
API-Projekt enthält nur die OpenAPI-Spezifikation. Das Client- und Server-Projekte importieren die Spezifikation und generieren daraus ihren Stub bzw. Skeleton.

OpenAPI Generator

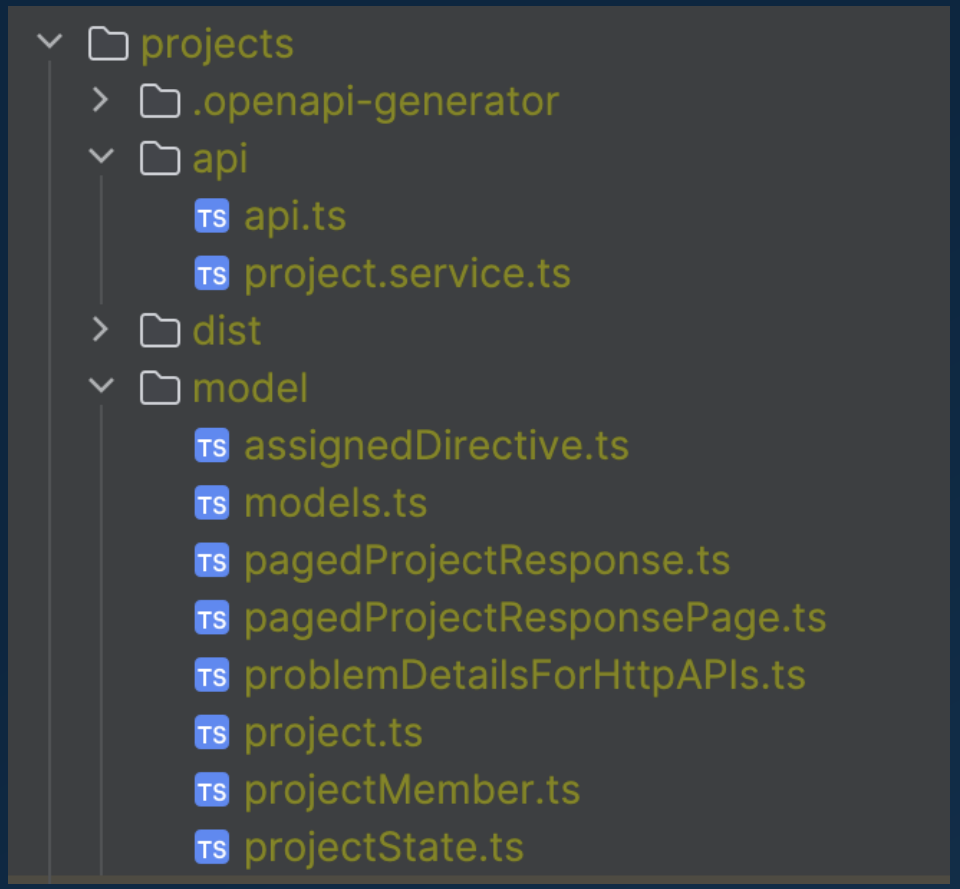


OpenAPI Generator

Java Spring Server Skeleton



TypeScript-Angular Client Stub



Extra Eigenschaften definieren

```
…
components:
  schemas:
    PagedProjectResponse:
      description: "Page of projects response"
      type: object
      properties:
        page:
          $ref: "./common/schemas/page-metadata.yml#/components/schemas/PageMetadataSorted"
        projects:
          type: array
          x-field-extra-annotation: "@lombok.Builder.Default"
          items:
            $ref: "#/components/schemas/Project“
…
```

API-ENTWICKLUNGSZYKLUS

- Feature Driven Development mit BGAV
  - FDD-Branch erzeugt im API-Modul eine neue BGAV-SNAPTSHOT Version
  - Service-App & Client-App werden manuell auf die API-BGAV-Version umgestellt
  - Damit sind API-Artefakte sowohl Lokal als auch innerhalb der CI/CD isoliert
  - Feature-Entwicklung erfolgt und abgeschlossen
  - API wird gemerged und eine neue Release Nummer erstellt
  - Server- und Client-Projekte stellen auf neue Release-Nummer zurück

https://bit.ly/3MYCbXi

REFACTORING

Von Code-First zu Contract-First in 4 Schritten

Laufende Anwendung unter **/api/v3/api-docs** die OpenAPI-Definition als JSON abrufen

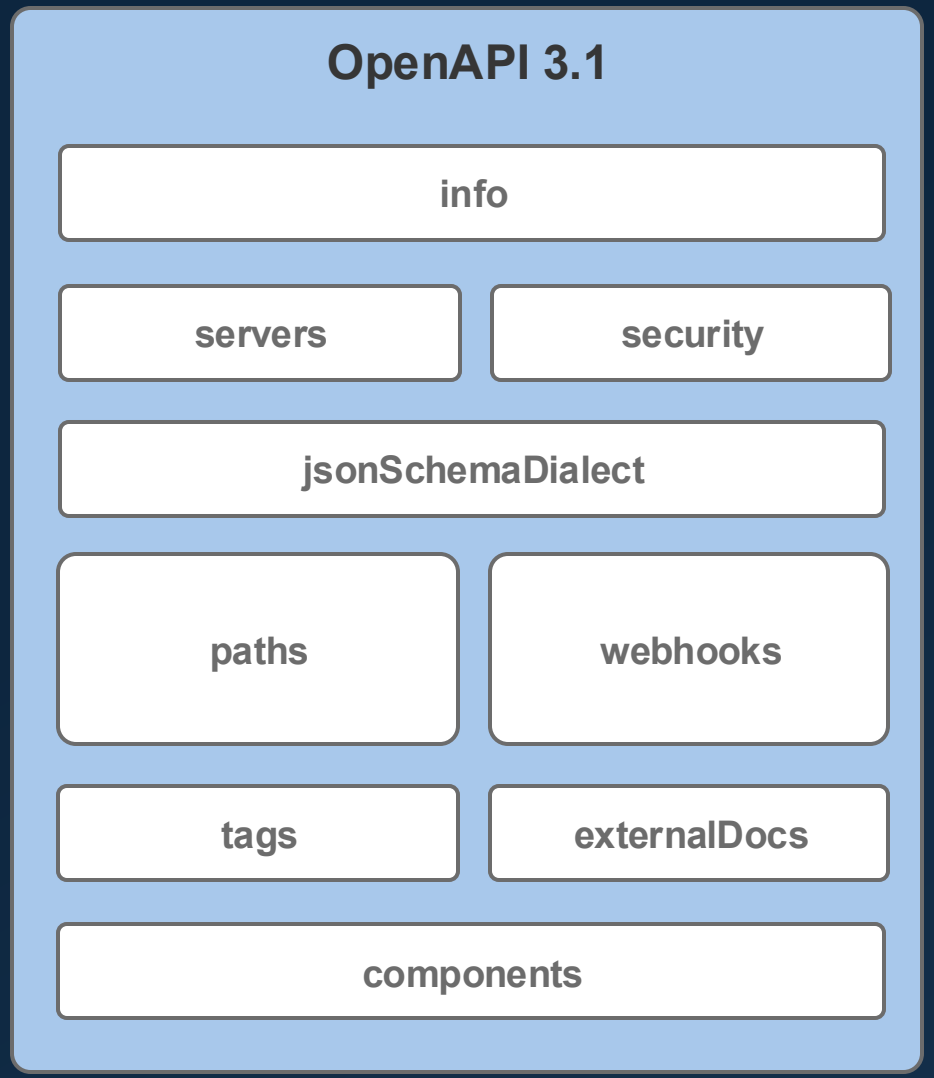
JSON mit dem Swagger-Editor von JSON zu YAML transformieren

OpenAPI API-Projekt anlegen und mit OpenAPI Generators die Stubs- und Skeletons generieren

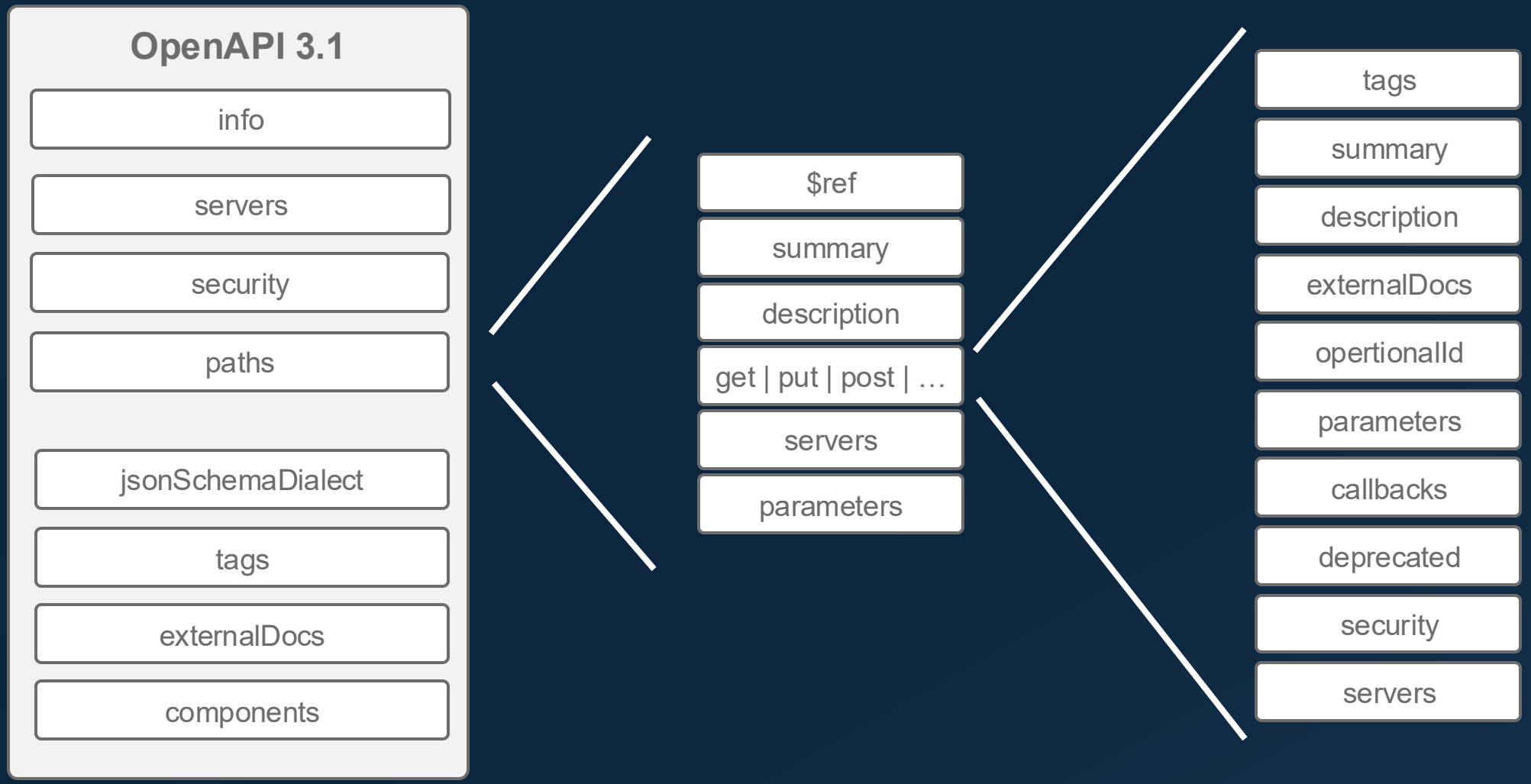
Generierte Klassen & Interfaces im Projekt einbinden und redundante Klassen & Annotationen entfernen

THE BAD

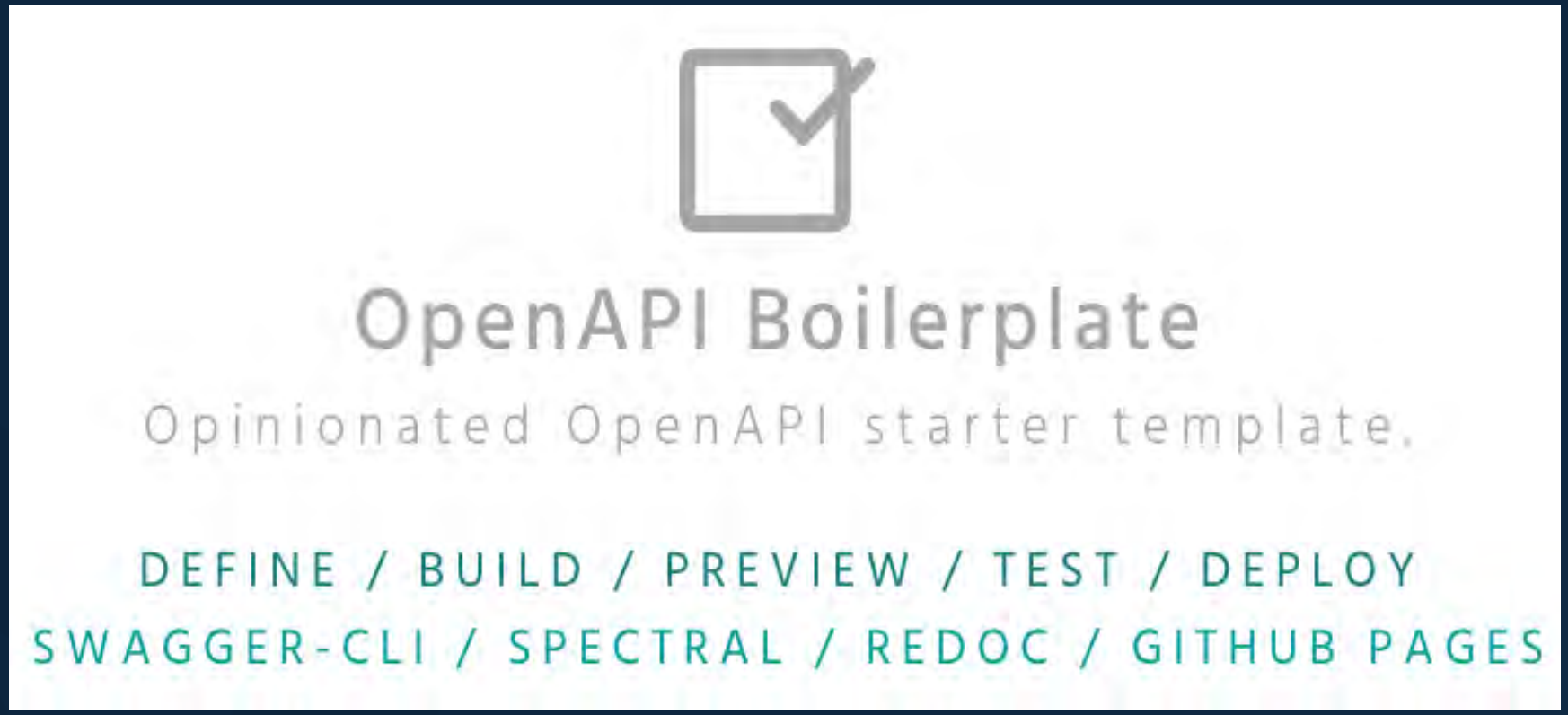
WIE WIRD EINE EINE OPENAPI DEFINITION IN SINNVOLLE WIEDERVERWENDBARE FRAGMENTE ZERLEGT?



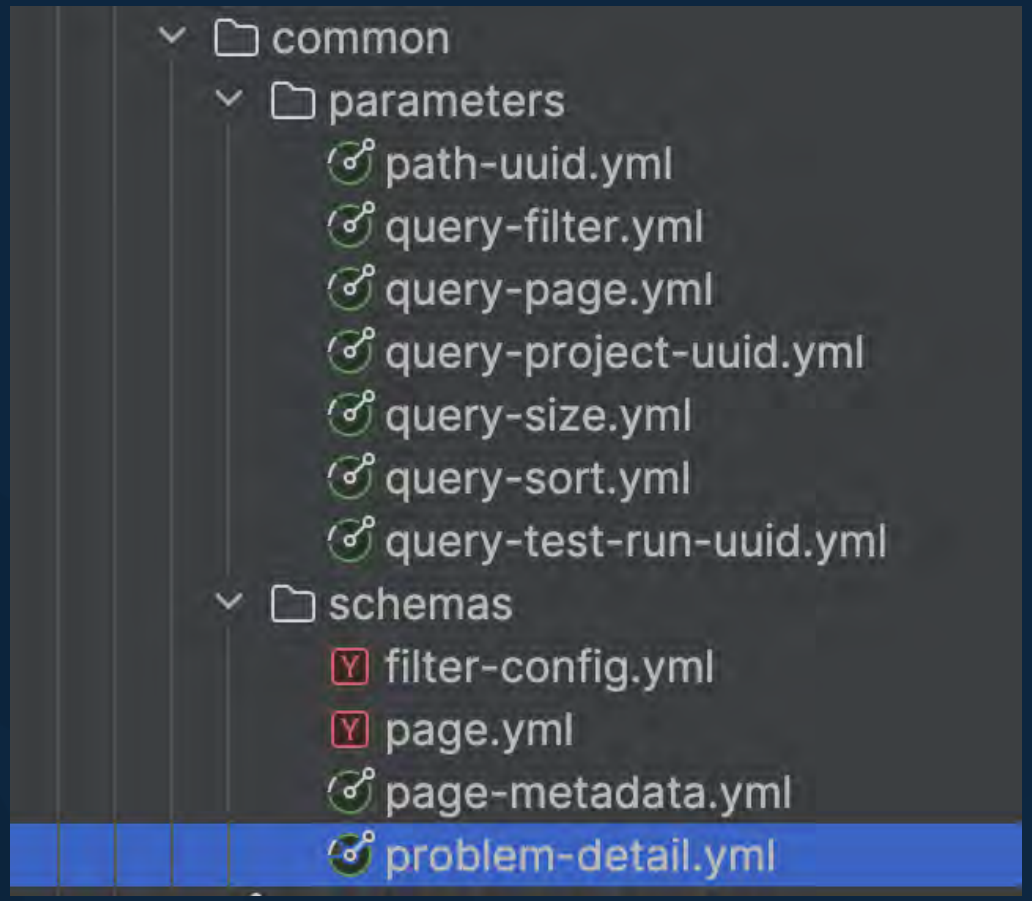
DEKOMPOSITION DER OPENAPI.YAML



HOW TO SPLIT AN OPENAPI FILE?



HOW TO SPLIT AN OPENAPI FILE?




```yaml
# Project Endpoints
/projects:
  get:
    summary: "Page of all available projects"
    operationId: "getProjectPage"
    description: "Page of all available projects"
    tags:
      - "Project"
    parameters:
      - $ref: './common/parameters/query-page.yml'
      - $ref: './common/parameters/query-size.yml'
      - $ref: './common/parameters/query-sort.yml'
      - $ref: './common/parameters/query-filter.yml'
    responses:
      '200':
        description: "Page of projects response"
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/PagedProjectResponse'
      default:
        description: "Unexpected error"
        content:
          application/json:
            schema:
              $ref: './common/schemas/problem-detail.yml#/components/schemas/ProblemDetail'
```

$ref ist leider nur an wenigen Stellen zulässig, und verlangt logisch zusammenhängende Teile zur trennen ☹

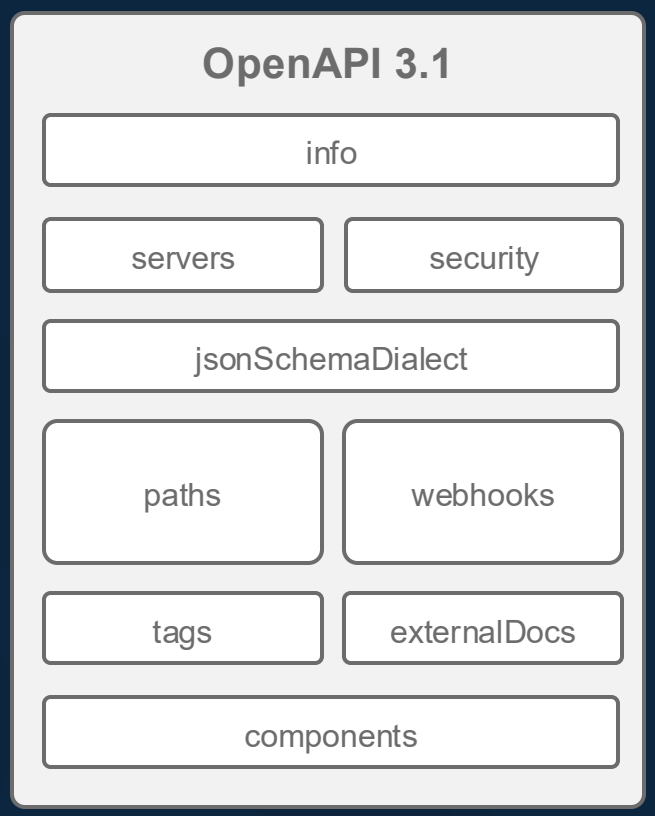
SPEC-STRICT-REFS

- Schema
- Response
- Parameter
- Example
- RequestBody
- Header
- SecurityScheme
- Link
- Callback
- PathItem

MODULE GENERIEREN

Ziel unterschiedliche Module / Kontexte generieren

EINE API-DEFINITION

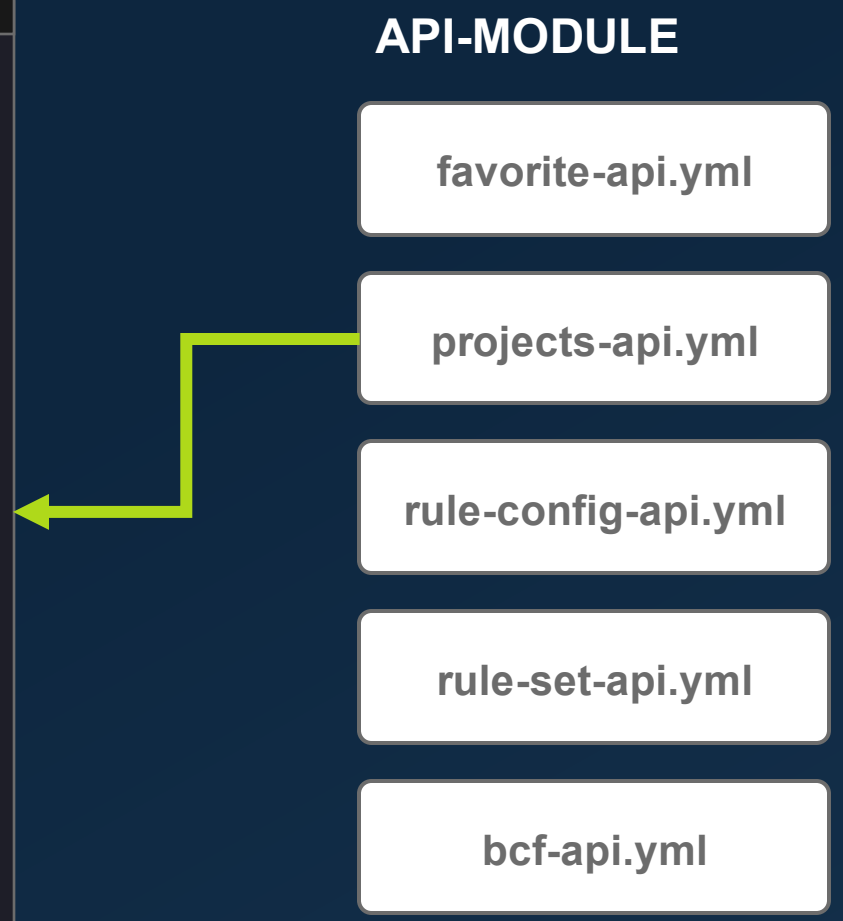


MEHRERE MODULE

- favorite
- project
- rule-config
- rule-set
- bcf

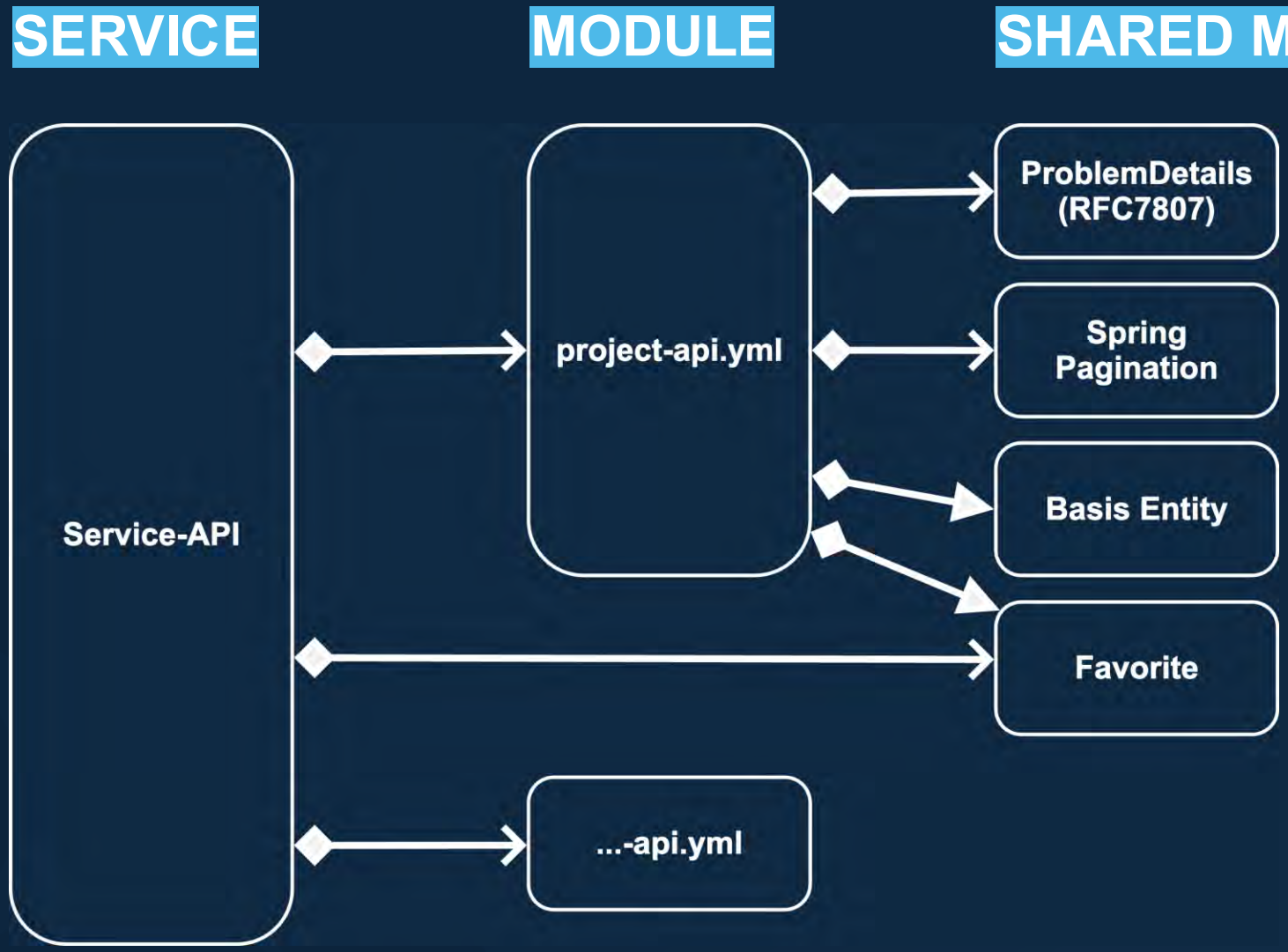
WORKAROUND
MODULE GENERIEREN

```
...
<execution>
    <id>projects-api</id>
    <phase>process-resources</phase>
    <goals>
        <goal>generate</goal>
    </goals>
    <configuration>
        <generatorName>spring</generatorName>
        <inputSpec>${project.build.directory}/schema/projects-api.yml</inputSpec>
        <apiPackage>${api.base.package}.project.api</apiPackage>
        <modelPackage>${api.base.package}.project.resource</modelPackage>
        <invokerPackage>${api.base.package}.project.client</invokerPackage>
        <generateApis>true</generateApis>
        <generateApiDocumentation>true</generateApiDocumentation>
        <generateSupportingFiles>false</generateSupportingFiles>
        <addCompileSourceRoot>true</addCompileSourceRoot>
    </configuration>
</execution>
...
```



THE UGLY

MODULE WIEDERVERWENDEN



Wiederverwendung von Standard Antworten im Fehlerfall

Wiederverwendung von Standardstrukturen und Parametern

Vererbung von allgemeinen Eigenschaften (Polymorphy)

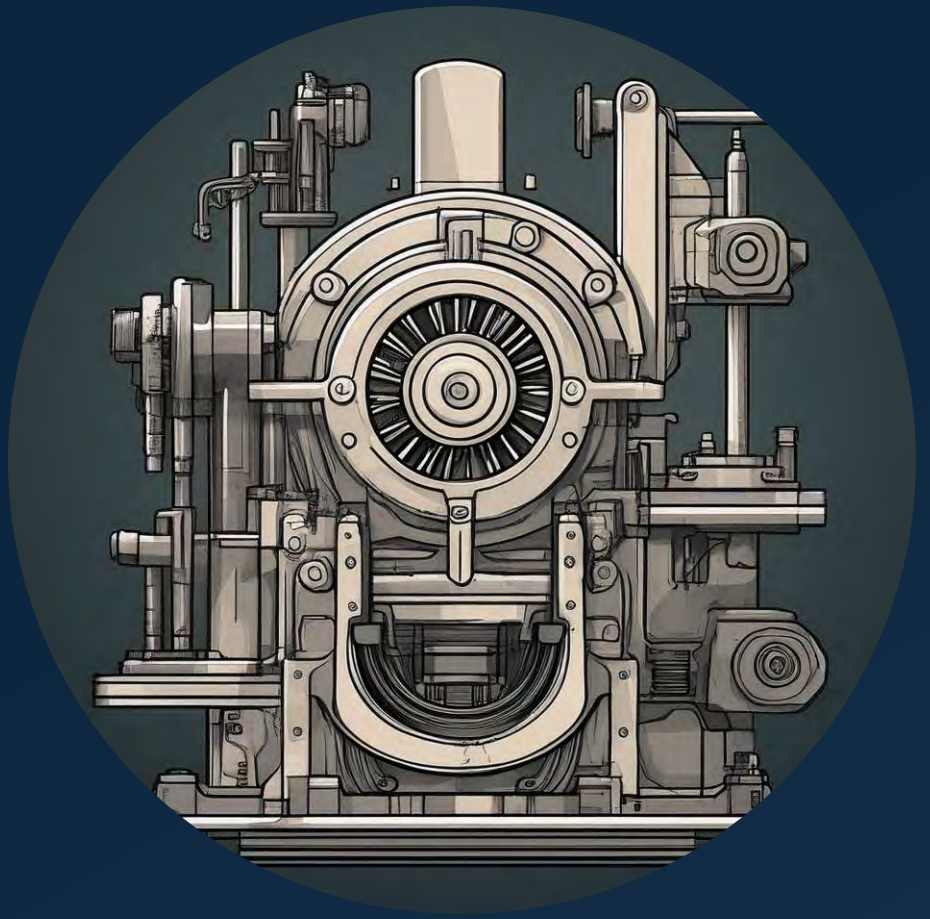
Einbinden von Common Libraries mit API-Features

WORKAROUND
KLASSEN WIEDERVERWENDEN

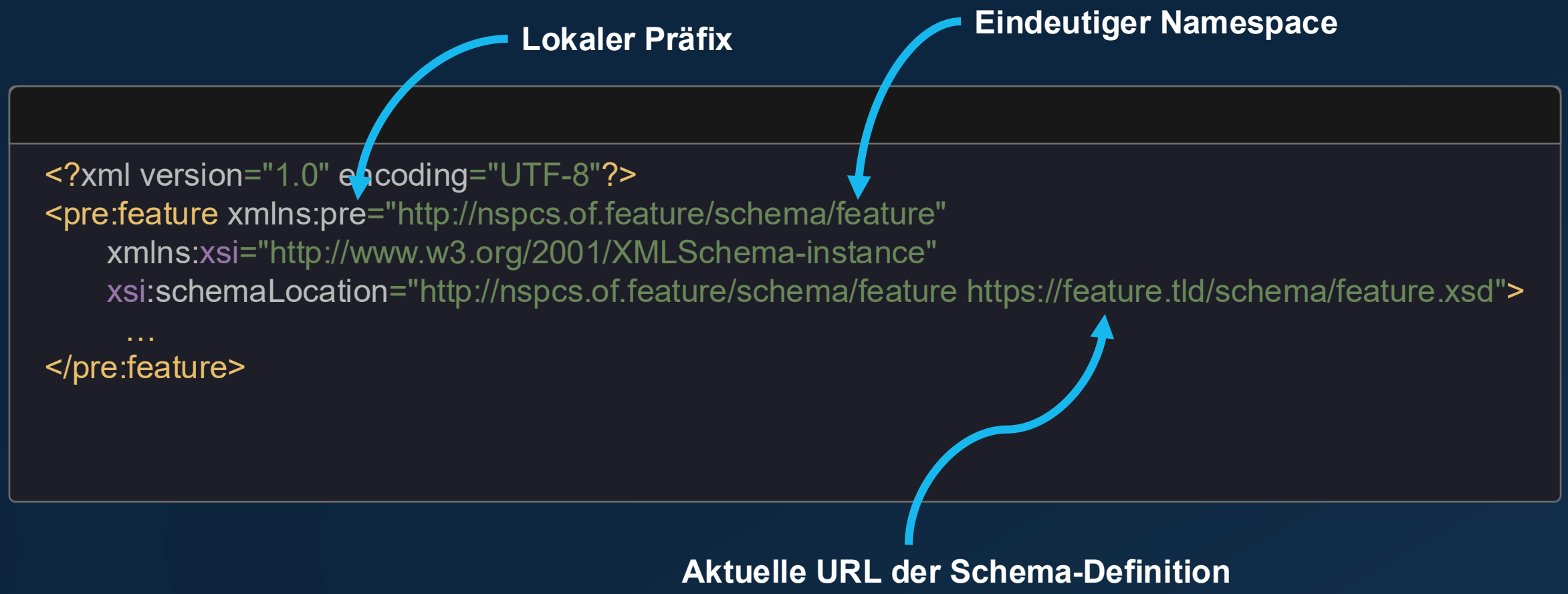
```
…
<schemaMappings>
    <schemaMapping>
        PageMetadataSorted=io.crowdcode.ncx.filters.PageMetadataSorted
    </schemaMapping>
    <schemaMapping>
        ProblemDetail=org.springframework.http.ProblemDetail
    </schemaMapping>
    <schemaMapping>
        AttributeGroup=io.crowdcode.ncx.filters.AttributeGroup
    </schemaMapping>
    <schemaMapping>
        AttributeValue=io.crowdcode.ncx.filters.AttributeValue
    </schemaMapping>
    <schemaMapping>
        FilterConfig=io.crowdcode.ncx.filters.FilterConfig
    </schemaMapping>
    <schemaMapping>
        Filter=io.crowdcode.ncx.filters.Filter
    </schemaMapping>
</schemaMappings>
…
```

```
…
<importMappings>
    <importMapping>
        PageMetadataSorted=io.crowdcode.ncx.filters.PageMetadataSorted
    </importMapping>
    <importMapping>
        Sort=org.springframework.data.domain.Sort
    </importMapping>
    <importMapping>
        Order=org.springframework.data.domain.Sort.Order
    </importMapping>
</importMappings>
…
```

Für mich sieht die Verwendung von $ref nach BROKEN BY DESIGN aus, da sie den Entwicklungszyklus von APIs ausblendet und keine Entkopplung von Modulen ermöglicht!



WAS WAR FALSCH AN NAMESPACES UND XSD:IMPORT?

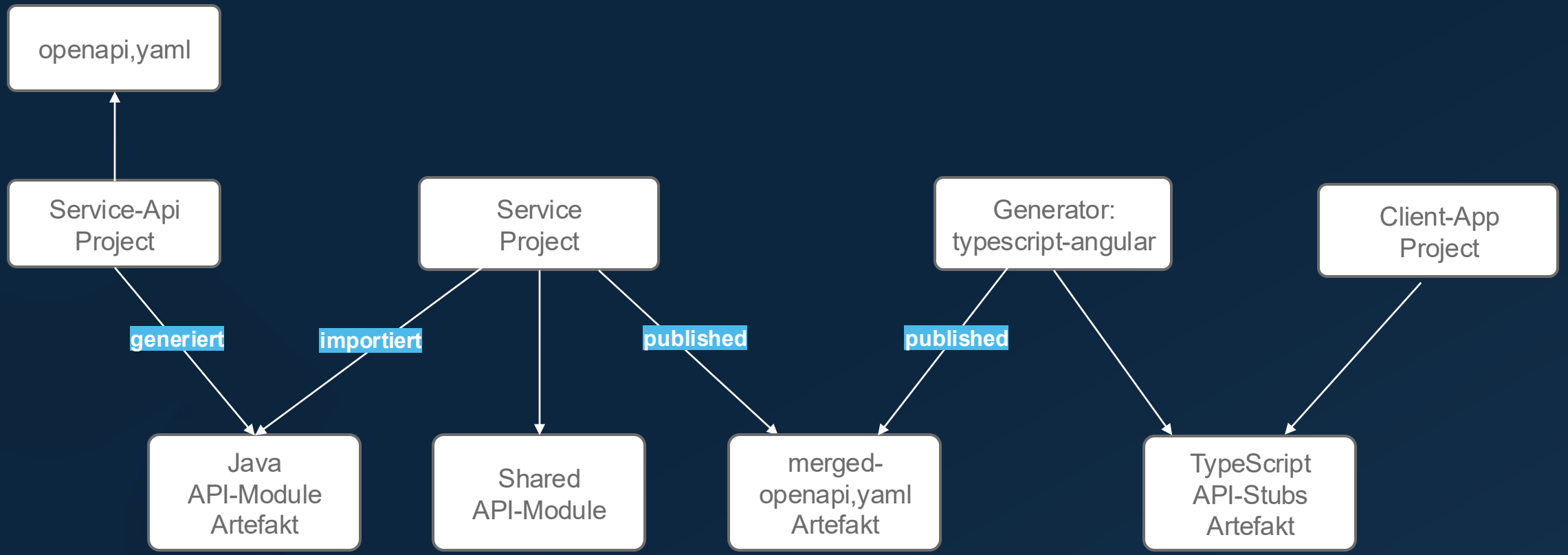


UGLY WORKAROUND

EINE BLENDED CODE GENERIERUNG MIT CONTRACT FIRST UND CODE SECOND STUFE!

»UGLY« WORKAROUND

Mehrstufige OpenAPI Generator Prozess



»Es ist eine Idiotie, das Shared-Nothing-Dogma auch auf das API-Design und die API-Entwicklung anzuwenden.«

Ingo, hier und heute

LEARNINGS

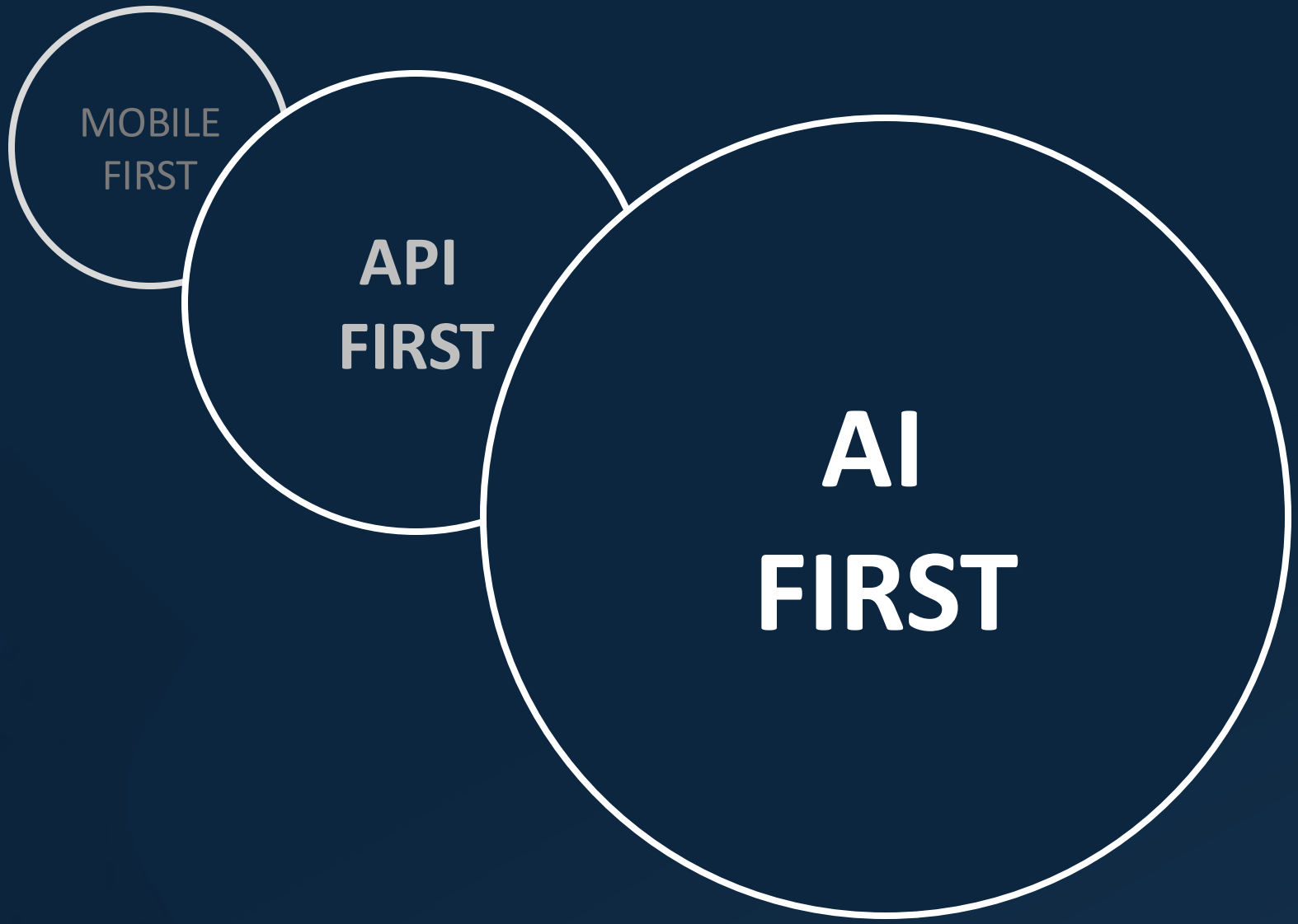
- Contract-First bei Service-API
- Flexible Contract- oder Code-First bei Closed Shared Modules
- Splitten in Segmenten
- Don‘t try inheritance over shared module
- Kompromisse beim DRY-Prinzip leben (notgedrungen)

FAZIT

Mit Kompromissen lässt sich sehr viel mit OpenAPI erreichen.

Um eine echte **modulare Komposition** zu ermöglichen, muss die Architektur von OpenAPI Generator umgebaut werden!

AUSBLICK



Ihr Ansprechpartner

Ingo Düppe

M +49 172 2325299

ingo.dueppe@crowdcode.io

CROWDCODE

CROWDCODE

DANKE!

CROWDCODE GmbH & Co. KG
Am Mittelhafen 16
48155 Münster

CROWDCODE GmbH & Co. KG
Robert-Bosch-Str. 10/3
56410 Montabaur

Kontakt
+49 2602 83900 30
kontakt@crowdcode.io
www.crowdcode.io

Geschäftsführende Gesellschafter
Ingo Düppe und Marcus Nörder-Tuitje
Amtsgericht Montabaur HRA 21597