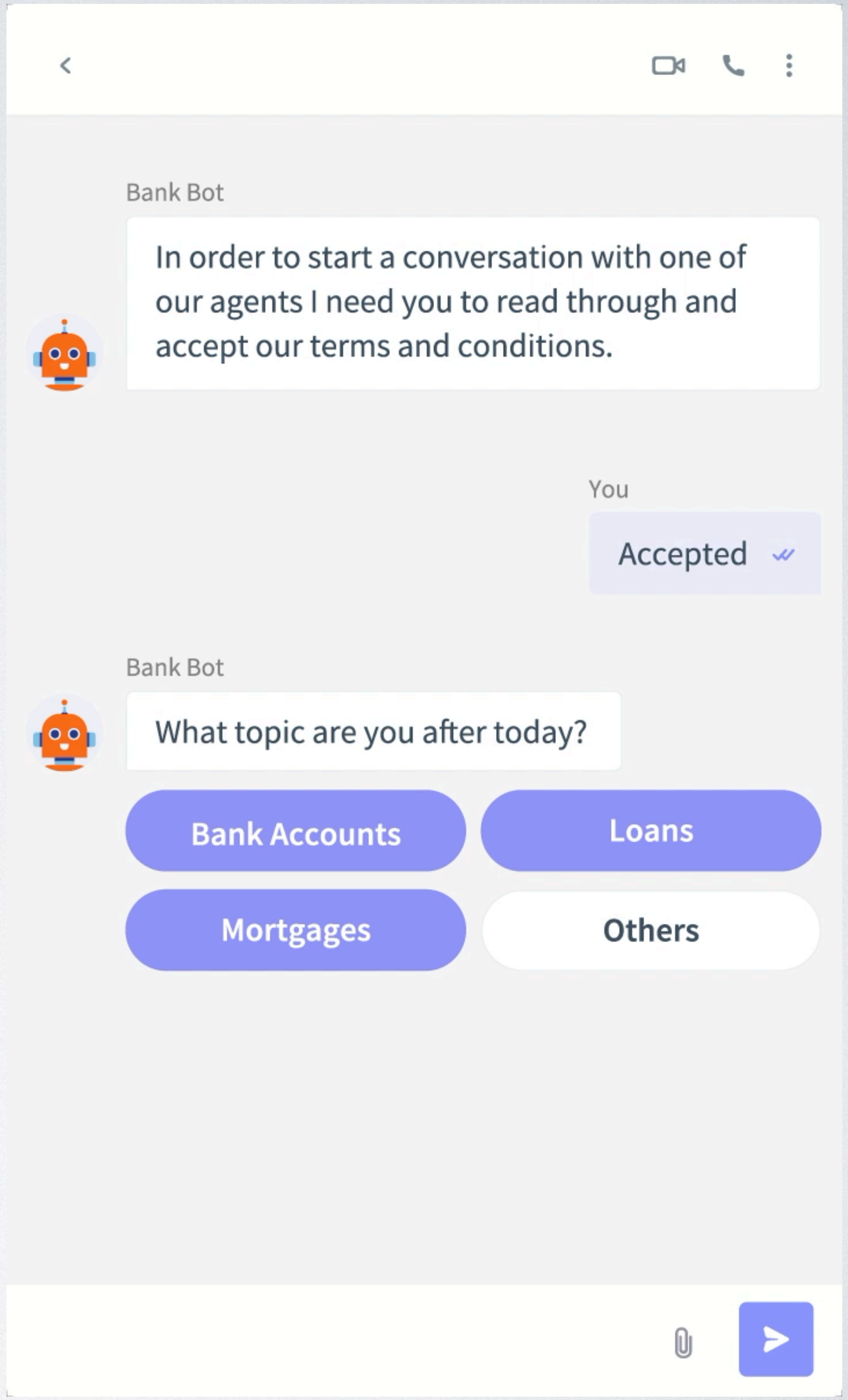
Eclipse MicroProfile OpenAPI

Eclipse DemoCamp Zurich 2019 - Jérémie Bresson unblu

Jérémie Bresson

@j2r2b

jmini



Bank Bot

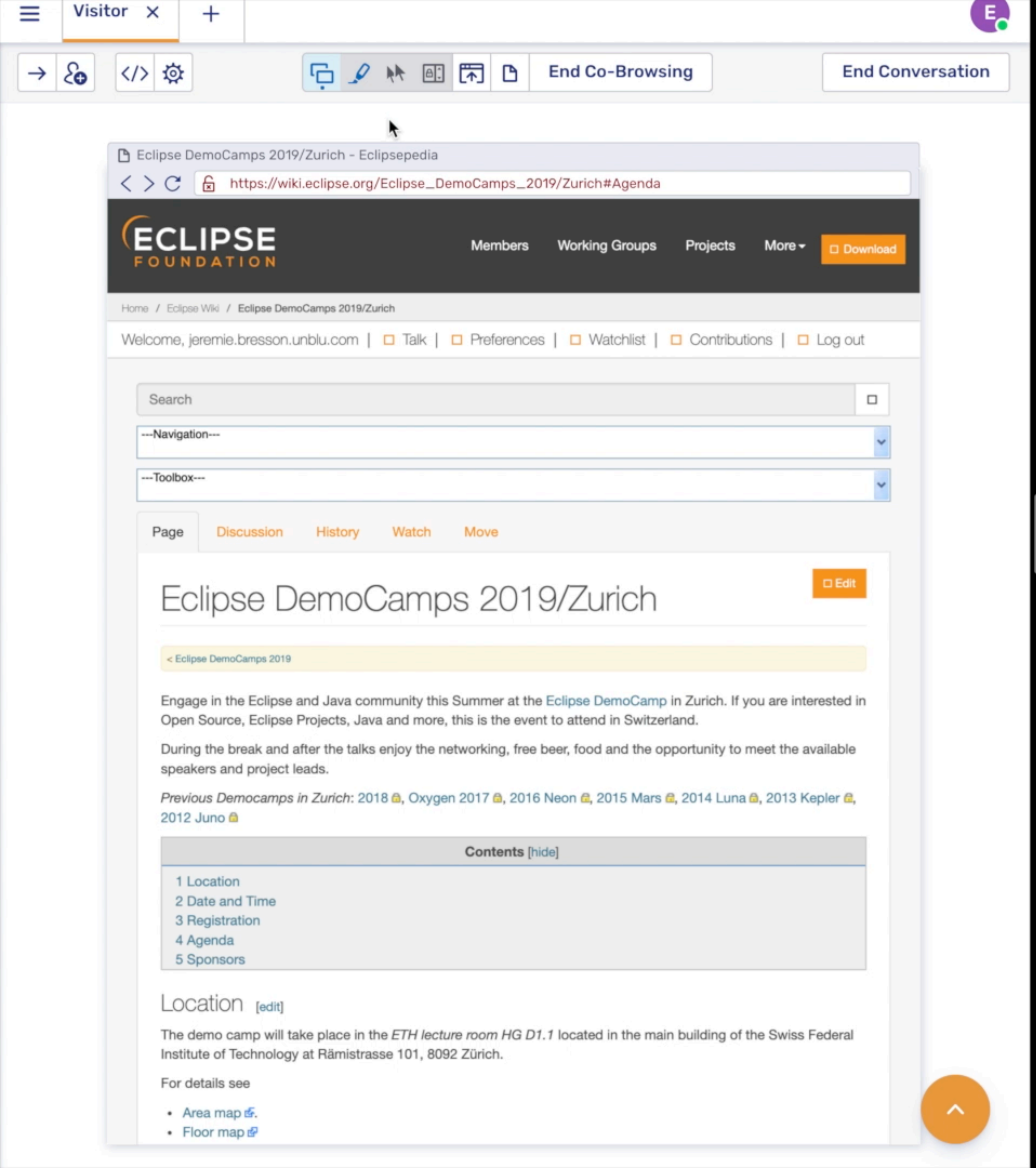
In order to start a conversation with one of our agents I need you to read through and accept our terms and conditions.

You

Accepted

Bank Bot

What topic are you after today?



Eclipse DemoCamps 2019/Zurich - Eclipsepedia

https://wiki.eclipse.org/Eclipse_DemoCamps_2019/Zurich#Agenda

Home / Eclipse Wiki / Eclipse DemoCamps 2019/Zurich

Welcome, jeremie.bresson.unblu.com | Talk | Preferences | Watchlist | Contributions | Log out

Eclipse DemoCamps 2019/Zurich

< Eclipse DemoCamps 2019

Engage in the Eclipse and Java community this Summer at the Eclipse DemoCamp in Zurich. If you are interested in Open Source, Eclipse Projects, Java and more, this is the event to attend in Switzerland.

During the break and after the talks enjoy the networking, free beer, food and the opportunity to meet the available speakers and project leads.

Previous Democamps in Zurich: 2018, Oxygen 2017, 2016 Neon, 2015 Mars, 2014 Luna, 2013 Kepler, 2012 Juno

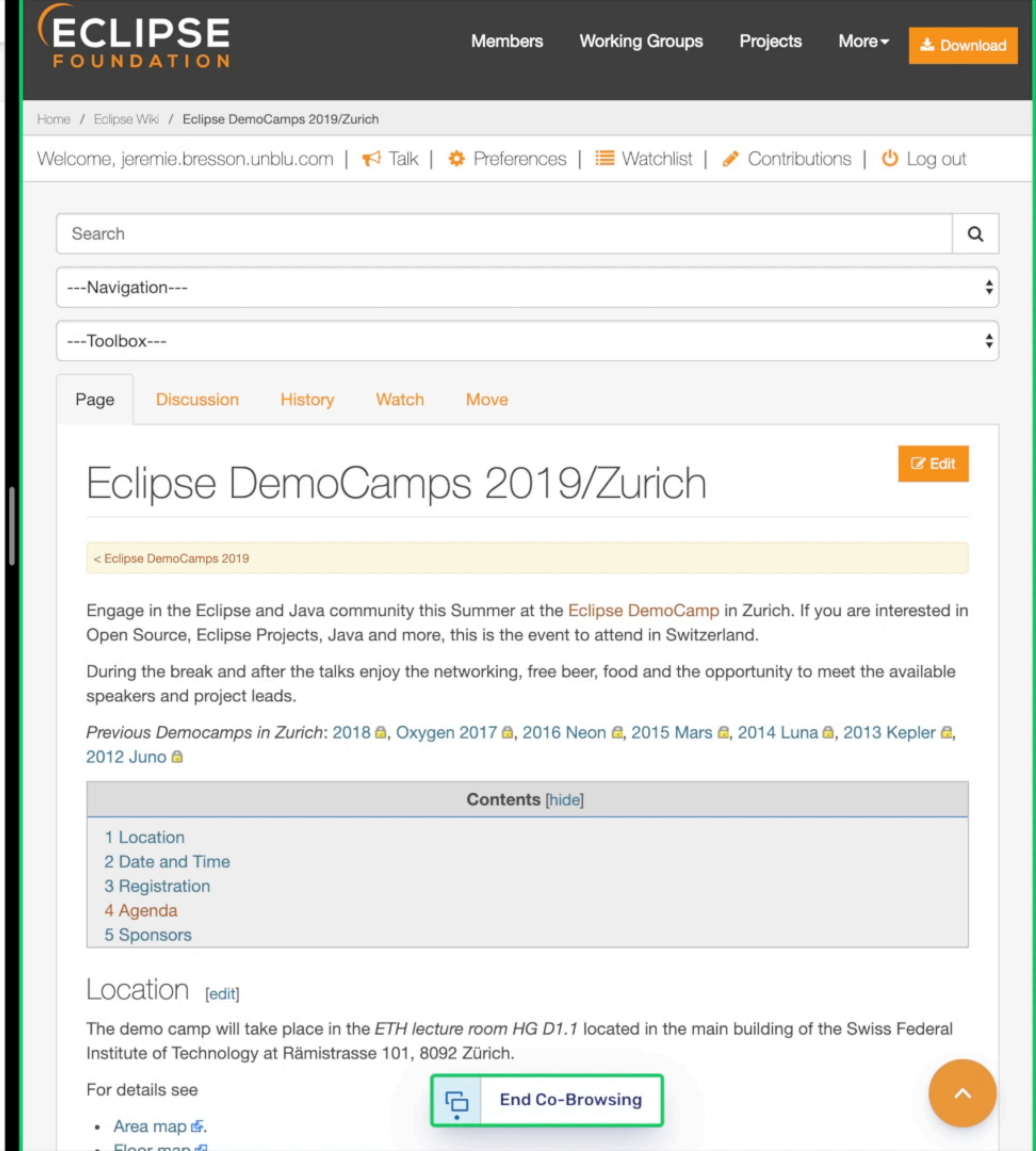
Contents [hide]

Location [edit]

The demo camp will take place in the *ETH lecture room HG D1.1* located in the main building of the Swiss Federal Institute of Technology at Rämistrasse 101, 8092 Zürich.

For details see

- Area map.
- Floor map



Home / Eclipse Wiki / Eclipse DemoCamps 2019/Zurich

Welcome, jeremie.bresson.unblu.com | Talk | Preferences | Watchlist | Contributions | Log out

Eclipse DemoCamps 2019/Zurich

< Eclipse DemoCamps 2019

Engage in the Eclipse and Java community this Summer at the Eclipse DemoCamp in Zurich. If you are interested in Open Source, Eclipse Projects, Java and more, this is the event to attend in Switzerland.

During the break and after the talks enjoy the networking, free beer, food and the opportunity to meet the available speakers and project leads.

Previous Democamps in Zurich: 2018, Oxygen 2017, 2016 Neon, 2015 Mars, 2014 Luna, 2013 Kepler, 2012 Juno

Contents [hide]

1 Location
2 Date and Time
3 Registration
4 Agenda
5 Sponsors

Location [edit]

The demo camp will take place in the *ETH lecture room HG D1.1* located in the main building of the Swiss Federal Institute of Technology at Rämistrasse 101, 8092 Zürich.

For details see

- Area map.

Eclipse Foundation

ECLIPSE®
FOUNDATION

Eclipse Foundation

- Vendor neutral player for open-source
- Intellectual Property Management
- Development Process
- Infrastructure
- Ecosystem Development

Eclipse Foundation

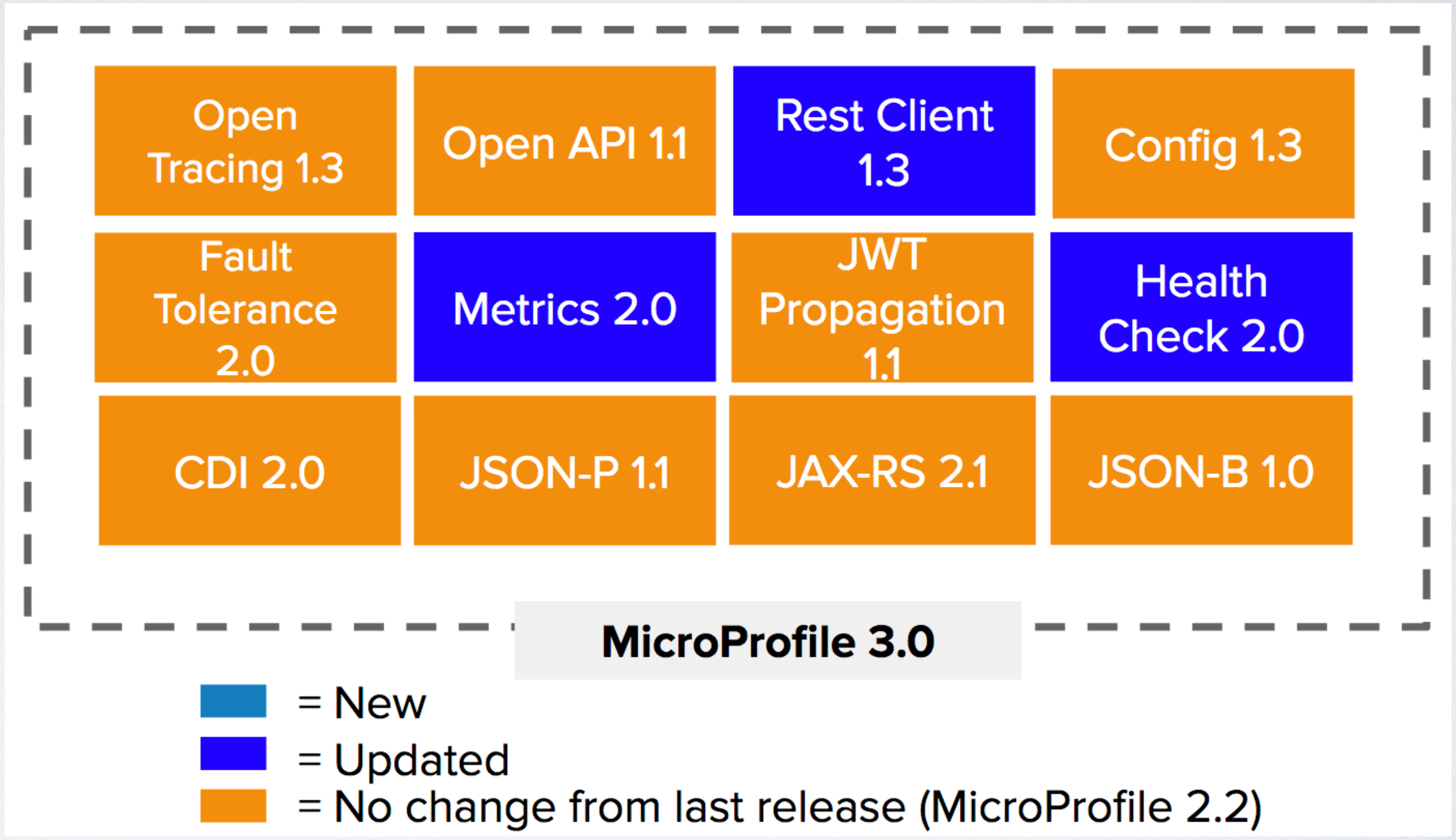
- Working groups: https://www.eclipse.org/org/workinggroups/explore.php
- Projects: https://projects.eclipse.org/

MicroProfile

MICROPROFILE™

OPTIMIZING ENTERPRISE JAVA

MicroProfile



Open Tracing 1.3	Open API 1.1	Rest Client 1.3	Config 1.3
Fault Tolerance 2.0	Metrics 2.0	JWT Propagation 1.1	Health Check 2.0
CDI 2.0	JSON-P 1.1	JAX-RS 2.1	JSON-B 1.0

MicroProfile 3.0

🟦 = New

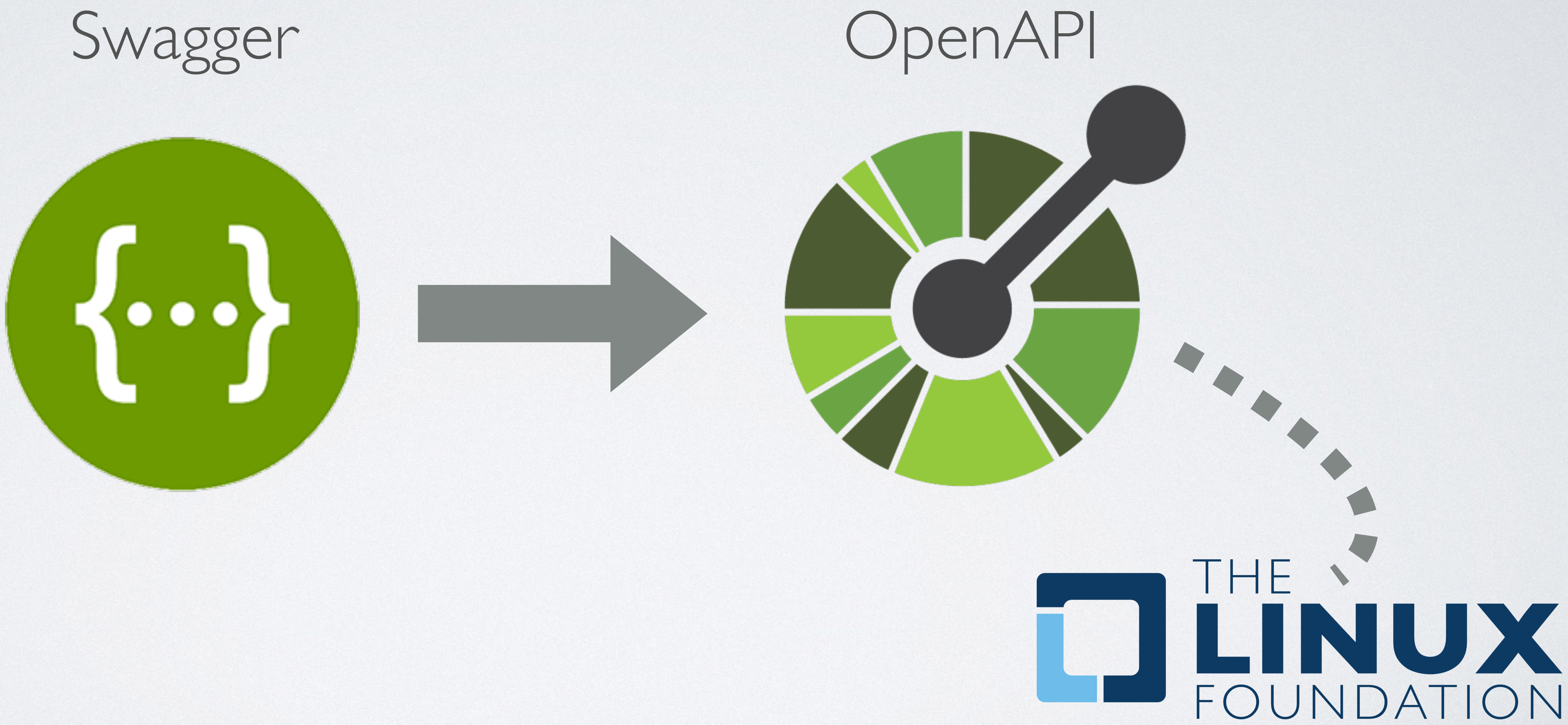
🟪 = Updated

🟧 = No change from last release (MicroProfile 2.2)

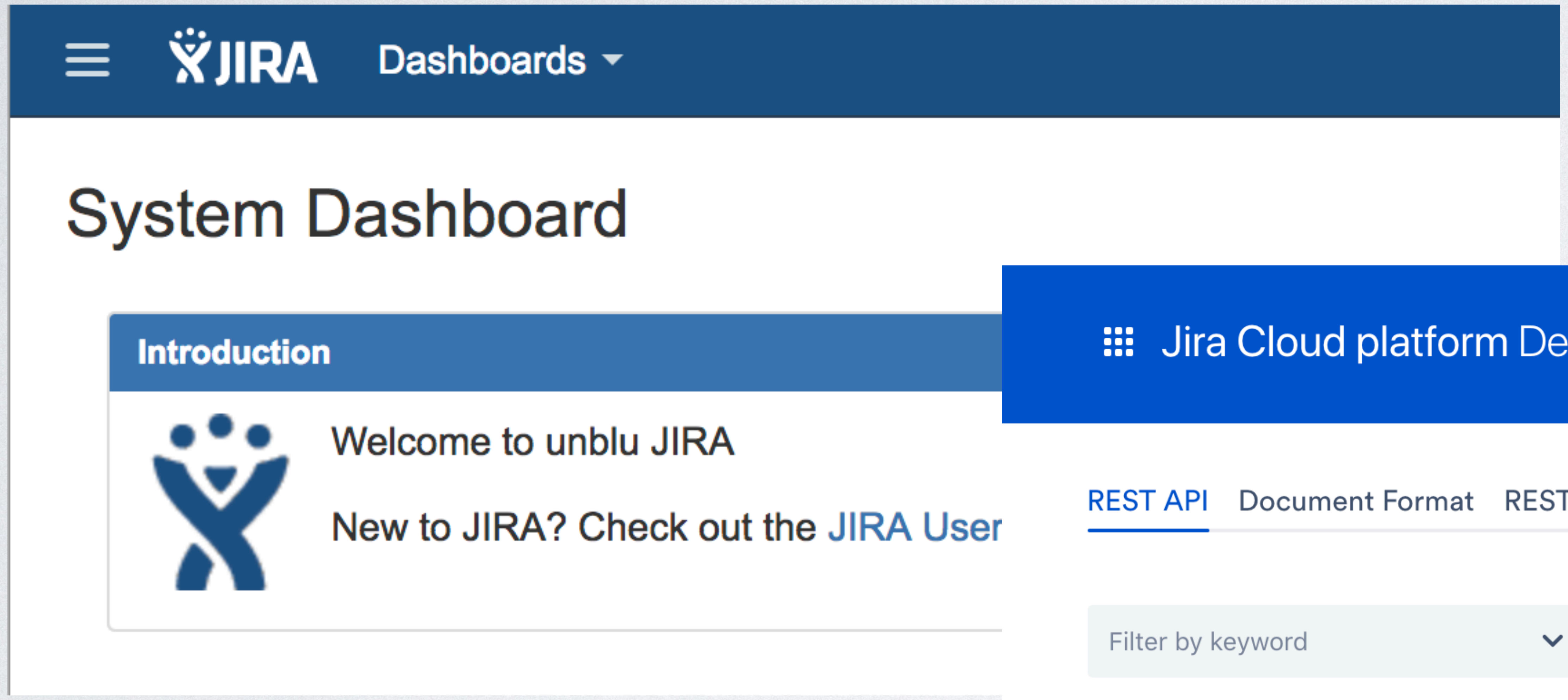
JAKARTA EE

OpenAPI

All the same



OpenAPIs are everywhere

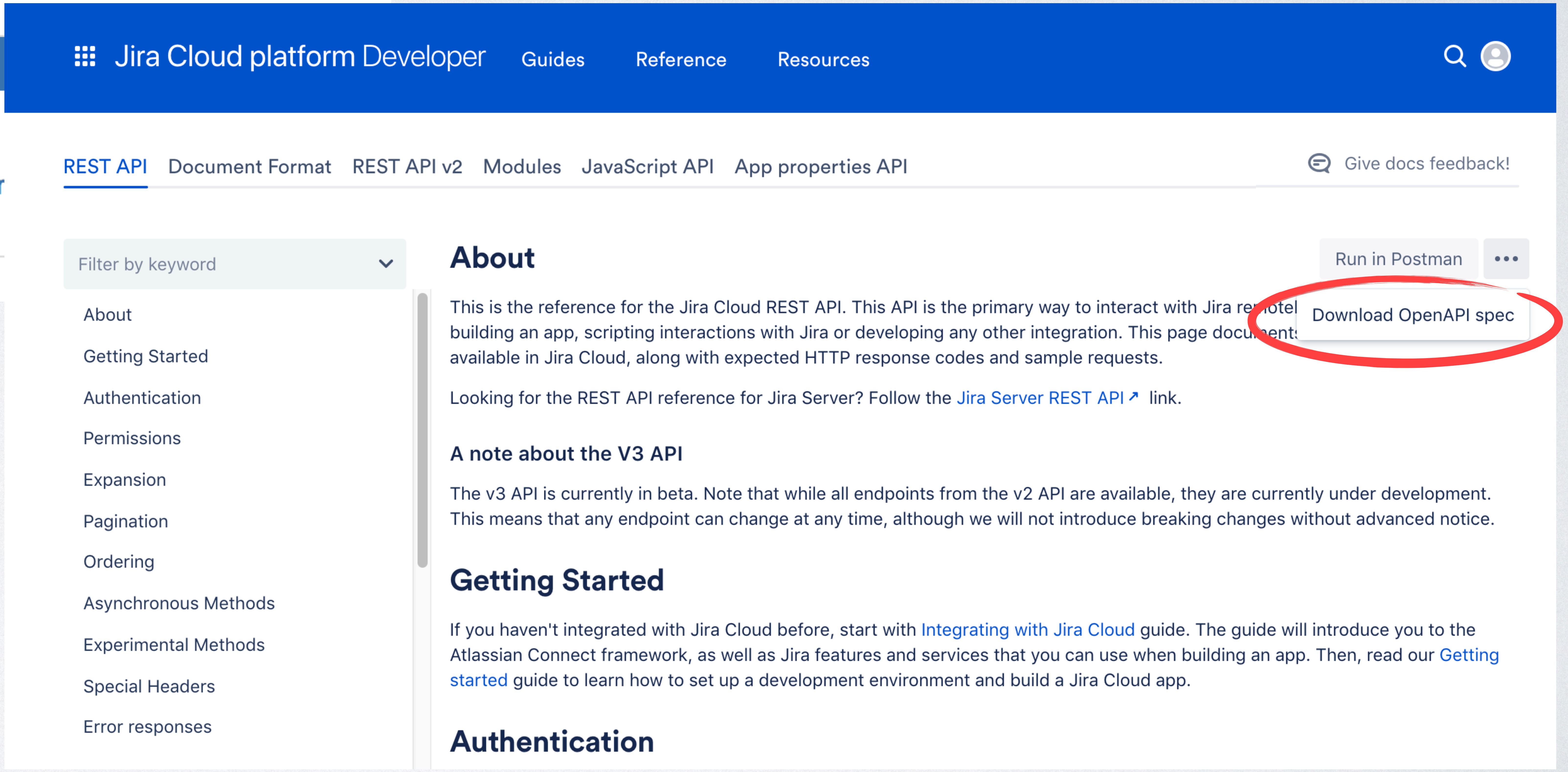


System Dashboard

Introduction

Welcome to unblu JIRA

New to JIRA? Check out the JIRA User



About

This is the reference for the Jira Cloud REST API. This API is the primary way to interact with Jira remotel building an app, scripting interactions with Jira or developing any other integration. This page documents available in Jira Cloud, along with expected HTTP response codes and sample requests.

Looking for the REST API reference for Jira Server? Follow the Jira Server REST API ↗ link.

A note about the V3 API

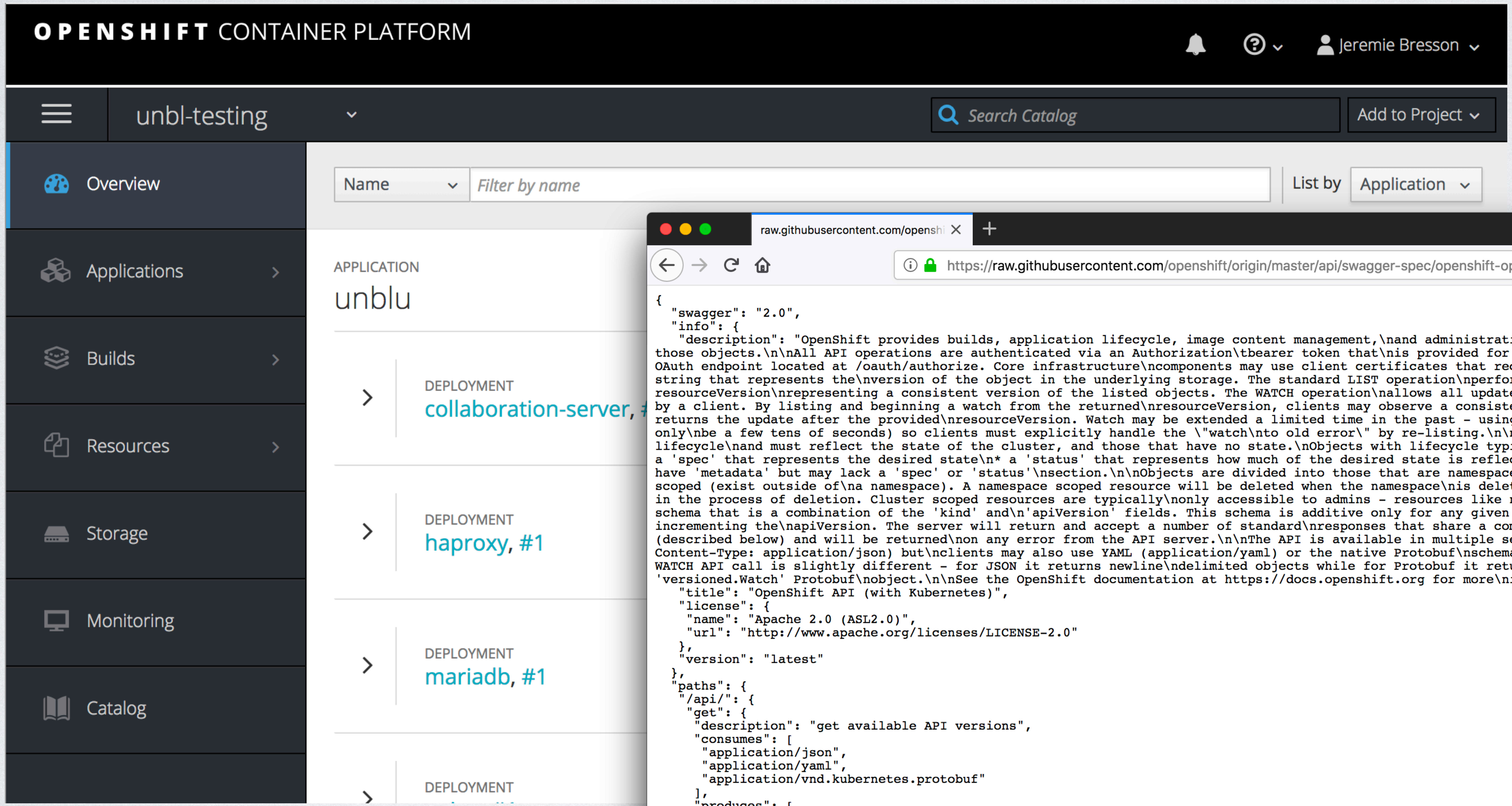
The v3 API is currently in beta. Note that while all endpoints from the v2 API are available, they are currently under development. This means that any endpoint can change at any time, although we will not introduce breaking changes without advanced notice.

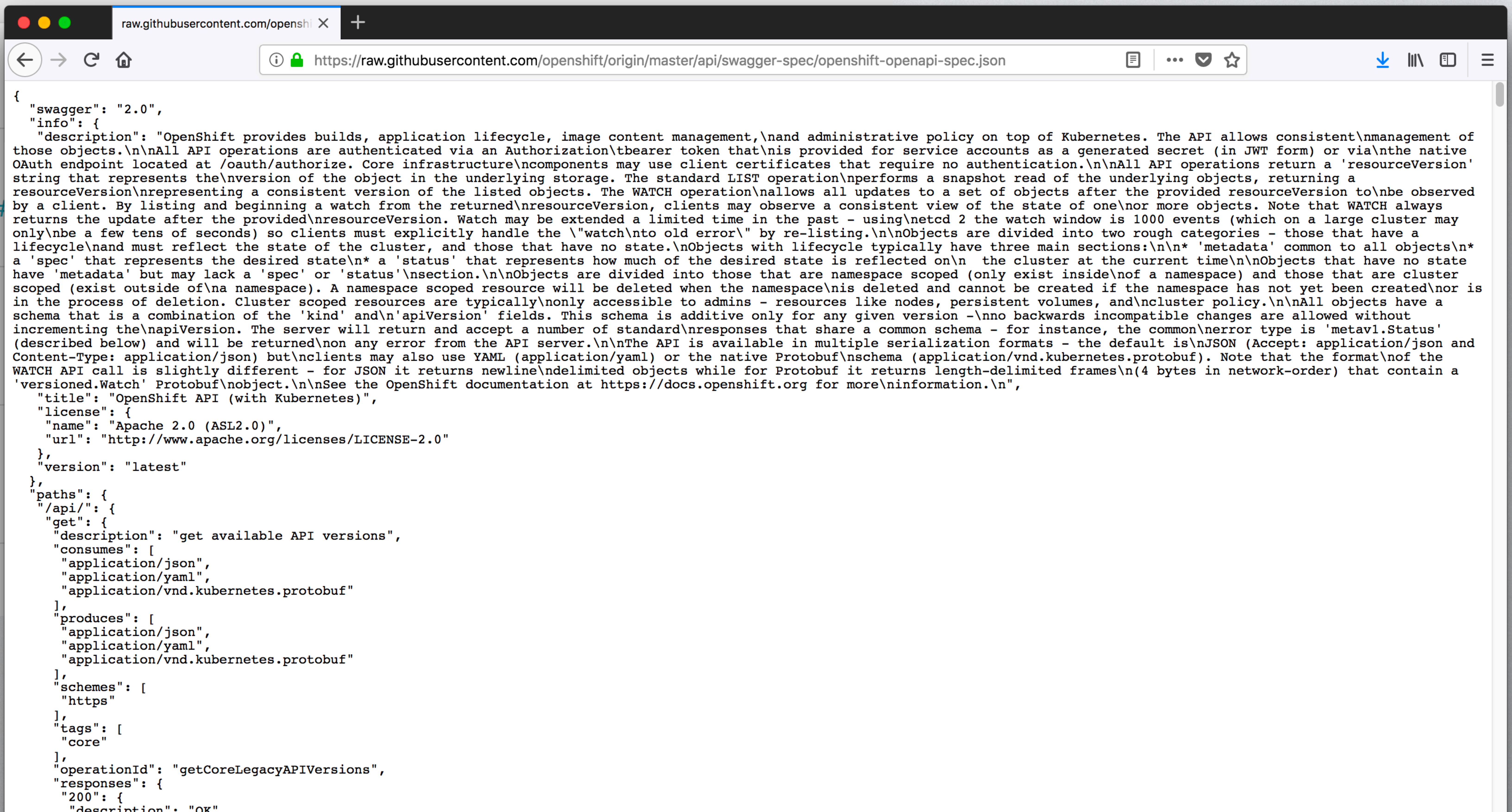
Getting Started

If you haven't integrated with Jira Cloud before, start with Integrating with Jira Cloud guide. The guide will introduce you to the Atlassian Connect framework, as well as Jira features and services that you can use when building an app. Then, read our Getting started guide to learn how to set up a development environment and build a Jira Cloud app.

Authentication

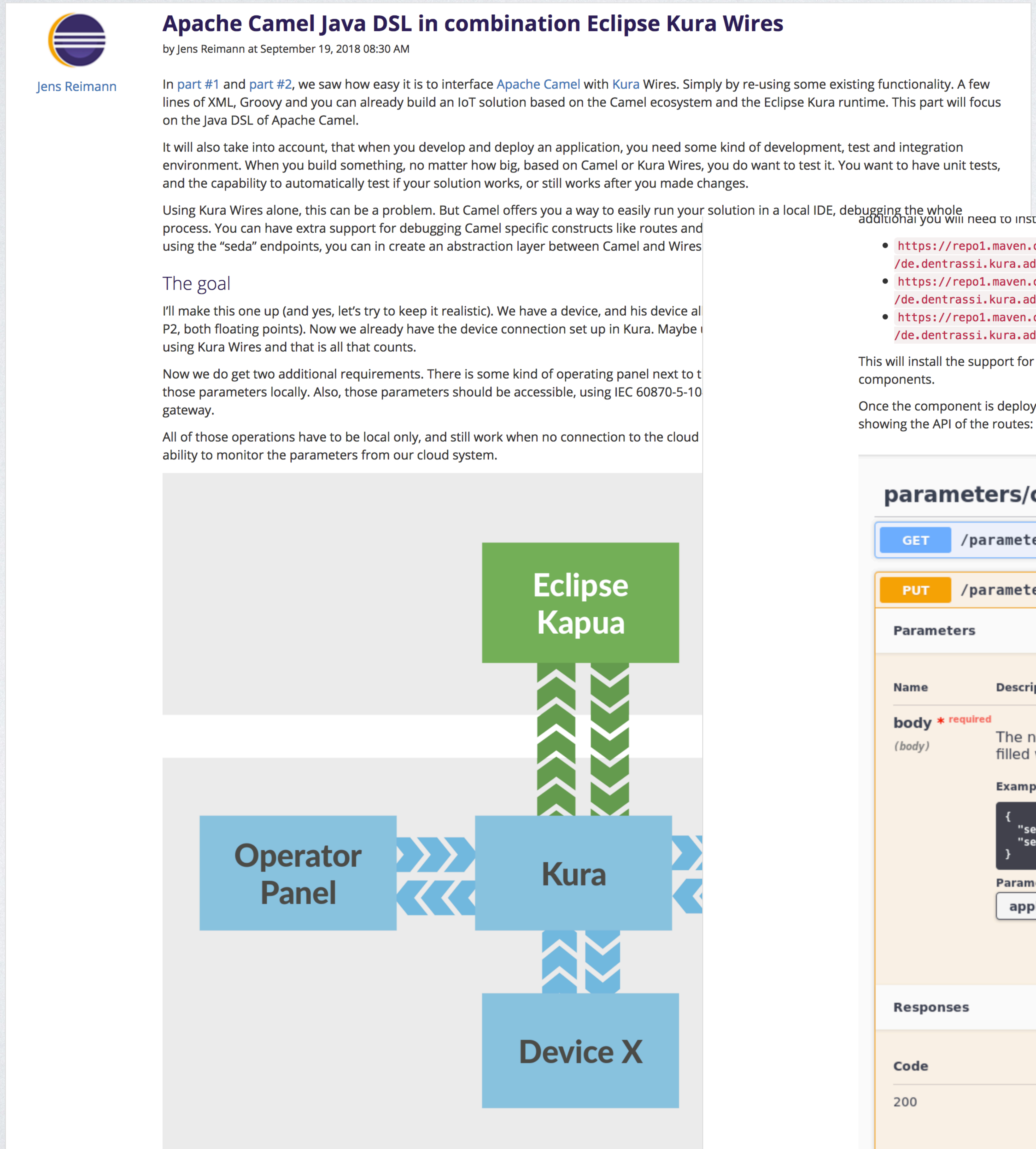
OpenAPIs are everywhere




raw.githubusercontent.com/opensh
https://raw.githubusercontent.com/openshift/origin/master/api/swagger-spec/openshift-openapi-spec.json

```
{
  "swagger": "2.0",
  "info": {
   "description": "OpenShift provides builds, application lifecycle, image content management,\nand administrative policy on top of Kubernetes. The API allows consistent\nmanagement of
those objects.\n\nAll API operations are authenticated via an Authorization\tbearer token that\nis provided for service accounts as a generated secret (in JWT form) or via\nthe native
OAuth endpoint located at /oauth/authorize. Core infrastructure\ncomponents may use client certificates that require no authentication.\n\nAll API operations return a 'resourceVersion'
string that represents the\nversion of the object in the underlying storage. The standard LIST operation\nperforms a snapshot read of the underlying objects, returning a
resourceVersion\nrepresenting a consistent version of the listed objects. The WATCH operation\nallows all updates to a set of objects after the provided resourceVersion to\nbe observed
by a client. By listing and beginning a watch from the returned\nresourceVersion, clients may observe a consistent view of the state of one\nor more objects. Note that WATCH always
returns the update after the provided\nresourceVersion. Watch may be extended a limited time in the past - using\netcd 2 the watch window is 1000 events (which on a large cluster may
only\nbe a few tens of seconds) so clients must explicitly handle the \"watch\nto old error\" by re-listing.\n\nObjects are divided into two rough categories - those that have a
lifecycle\nand must reflect the state of the cluster, and those that have no state.\nObjects with lifecycle typically have three main sections:\n\n* 'metadata' common to all objects\n*
a 'spec' that represents the desired state\n* a 'status' that represents how much of the desired state is reflected on\n  the cluster at the current time\n\nObjects that have no state
have 'metadata' but may lack a 'spec' or 'status'\nsection.\n\nObjects are divided into those that are namespace scoped (only exist inside\nof a namespace) and those that are cluster
scoped (exist outside of\na namespace). A namespace scoped resource will be deleted when the namespace\nis deleted and cannot be created if the namespace has not yet been created\nor is
in the process of deletion. Cluster scoped resources are typically\nonly accessible to admins - resources like nodes, persistent volumes, and\ncluster policy.\n\nAll objects have a
schema that is a combination of the 'kind' and\n'apiVersion' fields. This schema is additive only for any given version -\nno backwards incompatible changes are allowed without
incrementing the\napiVersion. The server will return and accept a number of standard\nresponses that share a common schema - for instance, the common\nerror type is 'metav1.Status'
(described below) and will be returned\non any error from the API server.\n\nThe API is available in multiple serialization formats - the default is\nJSON (Accept: application/json and
Content-Type: application/json) but\nclients may also use YAML (application/yaml) or the native Protobuf\nschema (application/vnd.kubernetes.protobuf). Note that the format\nof the
WATCH API call is slightly different - for JSON it returns newline\ndelimited objects while for Protobuf it returns length-delimited frames\n(4 bytes in network-order) that contain a
'versioned.Watch' Protobuf\nobject.\n\nSee the OpenShift documentation at https://docs.openshift.org for more\ninformation.\n",
   "title": "OpenShift API (with Kubernetes)",
   "license": {
    "name": "Apache 2.0 (ASL2.0)",
    "url": "http://www.apache.org/licenses/LICENSE-2.0"
   },
   "version": "latest"
  },
  "paths": {
   "/api/": {
    "get": {
     "description": "get available API versions",
     "consumes": [
      "application/json",
      "application/yaml",
      "application/vnd.kubernetes.protobuf"
     ],
     "produces": [
      "application/json",
      "application/yaml",
      "application/vnd.kubernetes.protobuf"
     ],
     "schemes": [
      "https"
     ],
     "tags": [
      "core"
     ],
     "operationId": "getCoreLegacyAPIVersions",
     "responses": {
      "200": {
       "description": "OK",
```

OpenAPIs are everywhere



Jens Reimann

Apache Camel Java DSL in combination Eclipse Kura Wires

by Jens Reimann at September 19, 2018 08:30 AM

In part #1 and part #2, we saw how easy it is to interface Apache Camel with Kura Wires. Simply by re-using some existing functionality. A few lines of XML, Groovy and you can already build an IoT solution based on the Camel ecosystem and the Eclipse Kura runtime. This part will focus on the Java DSL of Apache Camel.

It will also take into account, that when you develop and deploy an application, you need some kind of development, test and integration environment. When you build something, no matter how big, based on Camel or Kura Wires, you do want to test it. You want to have unit tests, and the capability to automatically test if your solution works, or still works after you made changes.

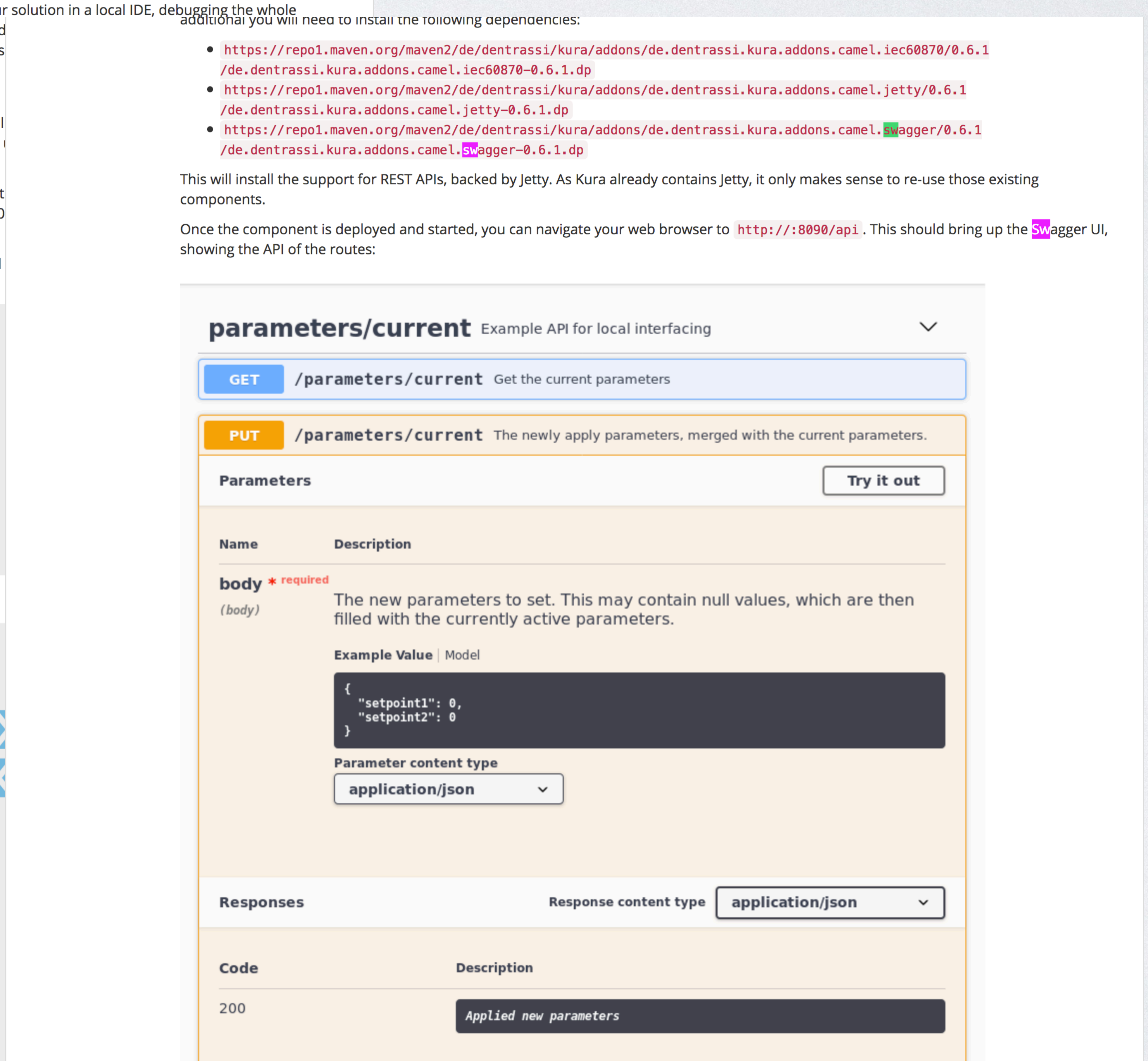
Using Kura Wires alone, this can be a problem. But Camel offers you a way to easily run your solution in a local IDE, debugging the whole process. You can have extra support for debugging Camel specific constructs like routes and
using the "seda" endpoints, you can in create an abstraction layer between Camel and Wires

The goal

I'll make this one up (and yes, let's try to keep it realistic). We have a device, and his device al
P2, both floating points). Now we already have the device connection set up in Kura. Maybe
using Kura Wires and that is all that counts.

Now we do get two additional requirements. There is some kind of operating panel next to t
those parameters locally. Also, those parameters should be accessible, using IEC 60870-5-10
gateway.

All of those operations have to be local only, and still work when no connection to the cloud
ability to monitor the parameters from our cloud system.



additional you will need to install the following dependencies:

- `https://repo1.maven.org/maven2/de/dentrassi/kura/addons/de.dentrassi.kura.addons.camel.iec60870/0.6.1/de.dentrassi.kura.addons.camel.iec60870-0.6.1.dp`
- `https://repo1.maven.org/maven2/de/dentrassi/kura/addons/de.dentrassi.kura.addons.camel.jetty/0.6.1/de.dentrassi.kura.addons.camel.jetty-0.6.1.dp`
- `https://repo1.maven.org/maven2/de/dentrassi/kura/addons/de.dentrassi.kura.addons.camel.swagger/0.6.1/de.dentrassi.kura.addons.camel.swagger-0.6.1.dp`

This will install the support for REST APIs, backed by Jetty. As Kura already contains Jetty, it only makes sense to re-use those existing components.

Once the component is deployed and started, you can navigate your web browser to `http://:8090/api`. This should bring up the Swagger UI, showing the API of the routes:

parameters/current Example API for local interfacing

GET `/parameters/current` Get the current parameters

PUT `/parameters/current` The newly apply parameters, merged with the current parameters.

Parameters Try it out

Name	Description
body * required (body)	The new parameters to set. This may contain null values, which are then filled with the currently active parameters.

Example Value | Model

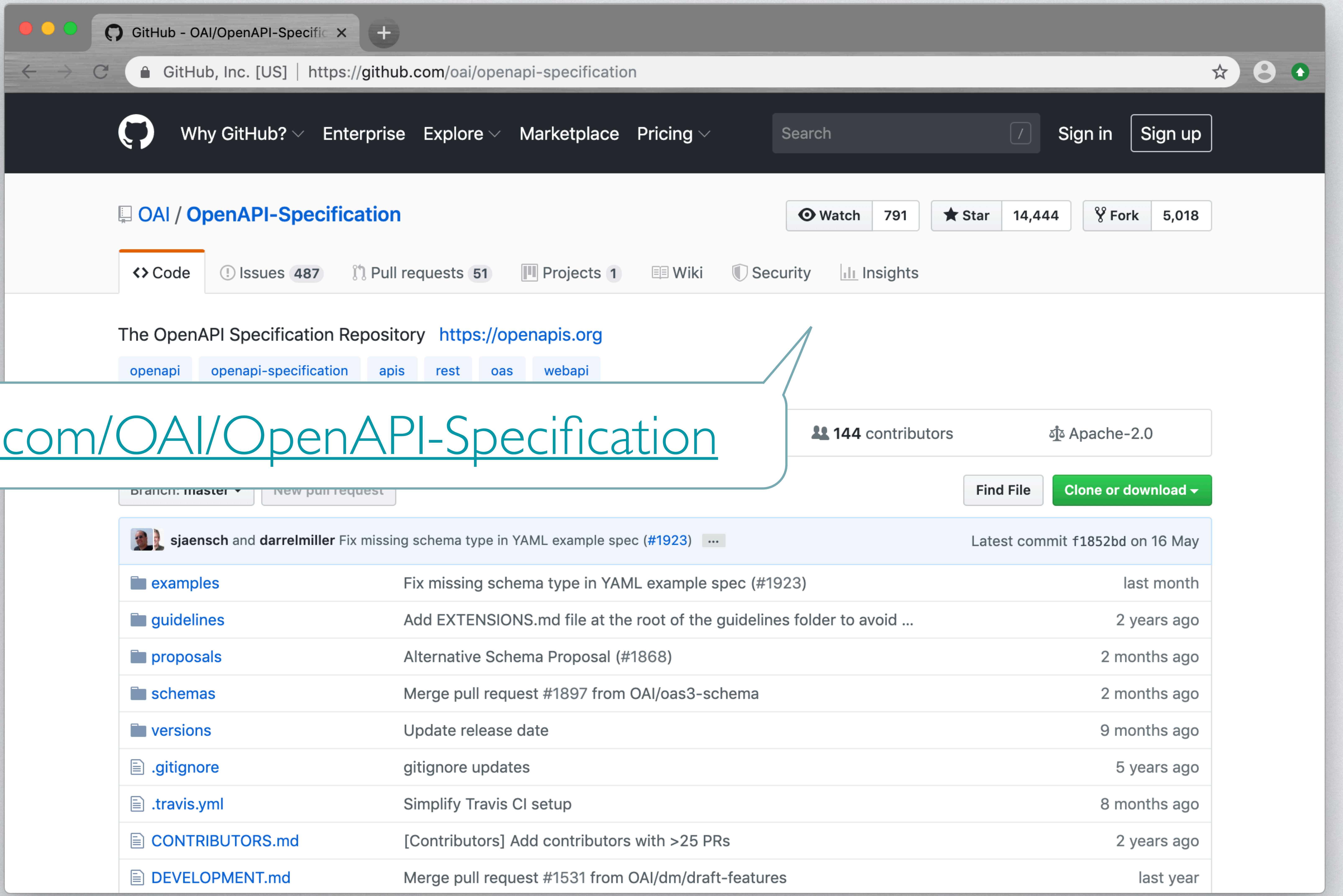
```
{
  "setpoint1": 0,
  "setpoint2": 0
}
```

Parameter content type: application/json

Responses Response content type: application/json

Code	Description
200	*Applied new parameters*

Specification GitHub Project



GitHub - OAI/OpenAPI-Specific

GitHub, Inc. [US] | https://github.com/oai/openapi-specification

Why GitHub? Enterprise Explore Marketplace Pricing Search Sign in Sign up

OAI / OpenAPI-Specification

Watch 791 Star 14,444 Fork 5,018

Code Issues 487 Pull requests 51 Projects 1 Wiki Security Insights

The OpenAPI Specification Repository https://openapis.org

openapi openapi-specification apis rest oas webapi

144 contributors Apache-2.0

Branch: master New pull request Find File Clone or download

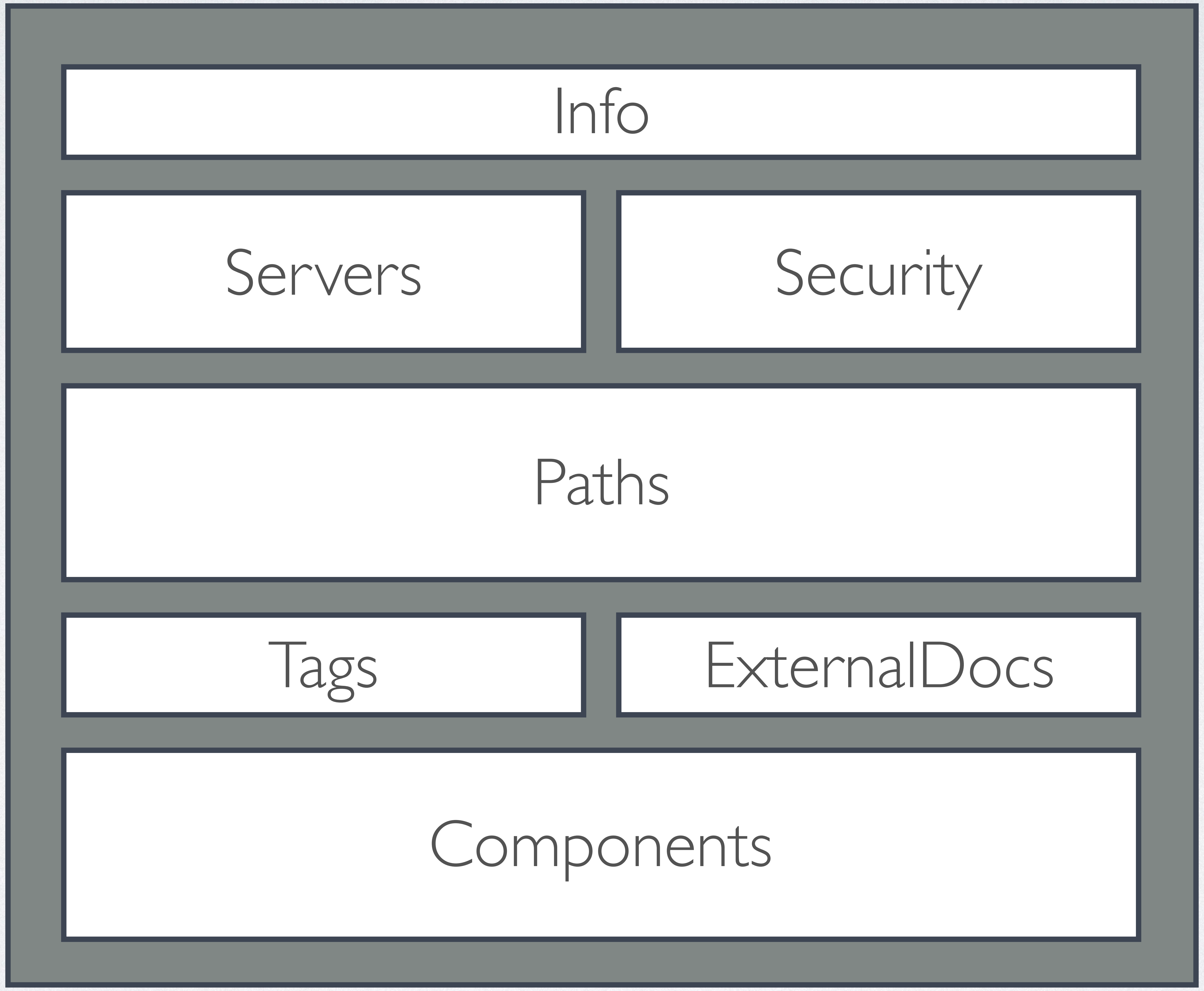
sjaensch and darrelmiller Fix missing schema type in YAML example spec (#1923) ... Latest commit f1852bd on 16 May

examples	Fix missing schema type in YAML example spec (#1923)	last month
guidelines	Add EXTENSIONS.md file at the root of the guidelines folder to avoid ...	2 years ago
proposals	Alternative Schema Proposal (#1868)	2 months ago
schemas	Merge pull request #1897 from OAI/oas3-schema	2 months ago
versions	Update release date	9 months ago
.gitignore	gitignore updates	5 years ago
.travis.yml	Simplify Travis CI setup	8 months ago
CONTRIBUTORS.md	[Contributors] Add contributors with >25 PRs	2 years ago
DEVELOPMENT.md	Merge pull request #1531 from OAI/dm/draft-features	last year

https://github.com/OAI/OpenAPI-Specification

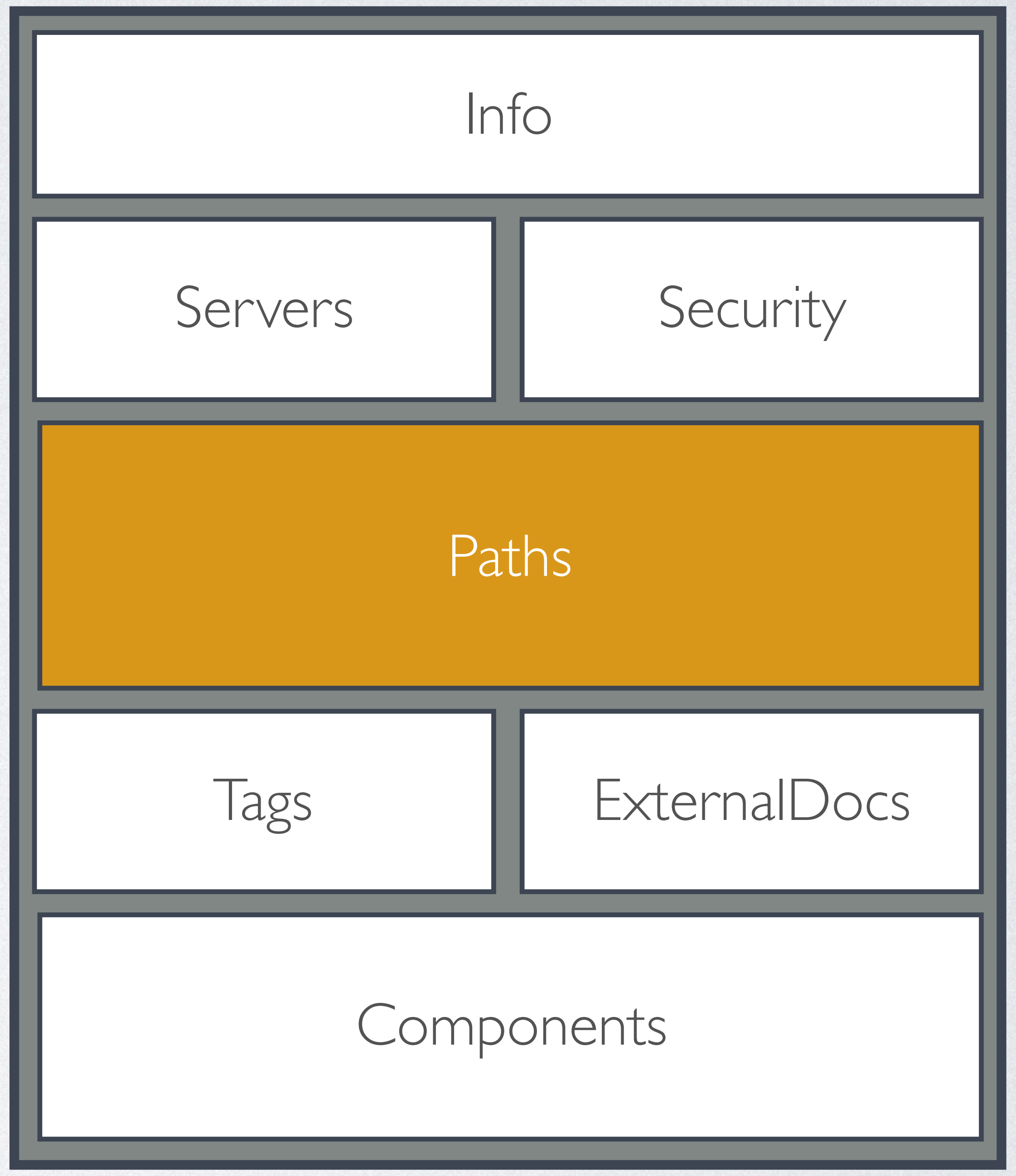
An OpenAPI Specification

OpenAPI v3

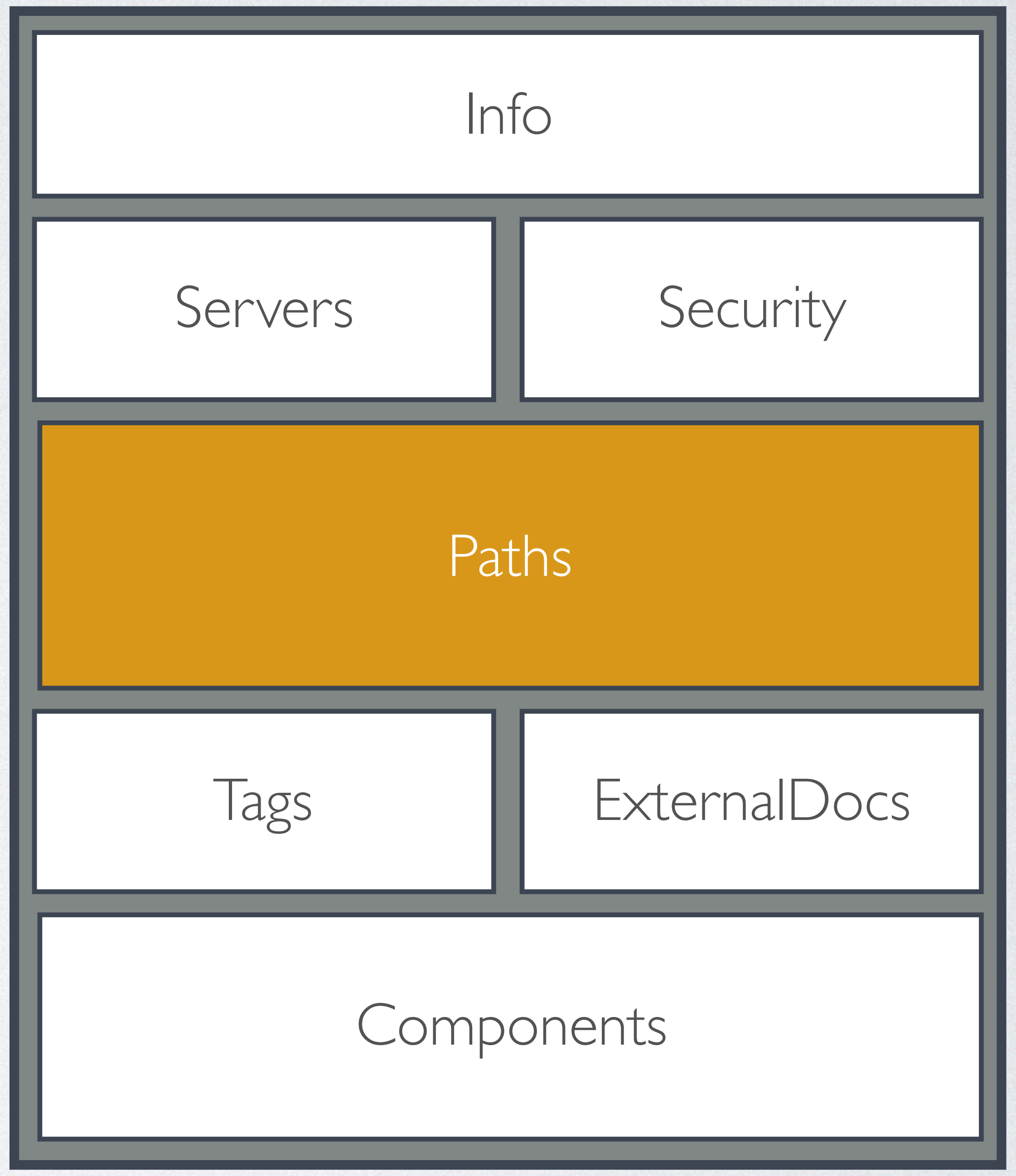


JSON
or
YAML

```yaml
openapi: 3.0.1
info:
  title: Todo Backend
  version: "1.0"
paths:
  /api/{id}:
    get:
      summary: Get the one todo
      operationId: todoGetOne
      parameters:
      - name: id
        in: path
        description: The id of the todo
        required: true
        schema:
          format: int64
          type: integer
        example: "42"
      responses:
        200:
          description: The requested Todo
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Todo'
```

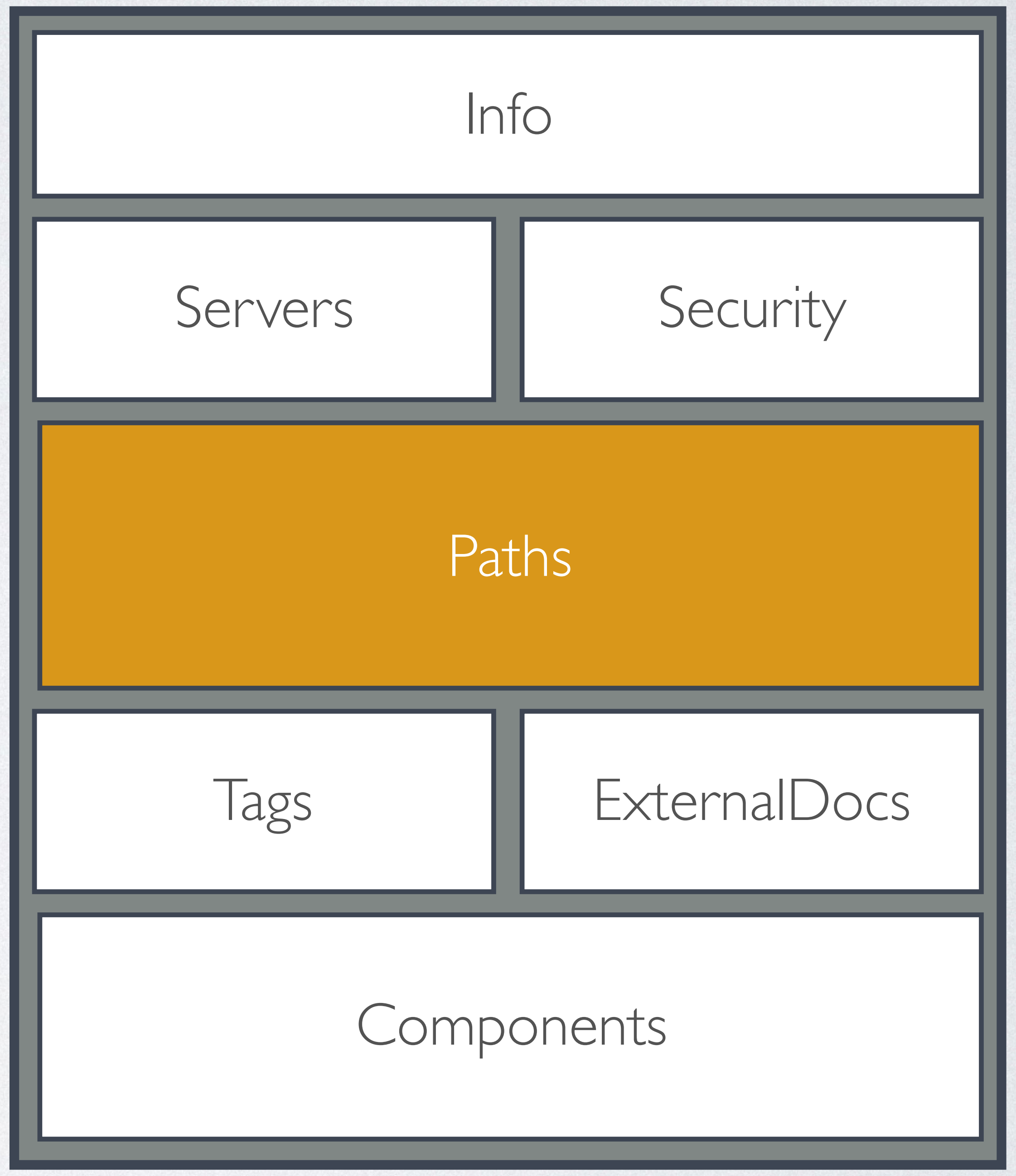


```yaml
openapi: 3.0.1
info:
  title: Todo Backend
  version: "1.0"
paths:
  /api/{id}:
    get:
      summary: Get the one todo
      operationId: todoGetOne
      parameters:
      - name: id
        in: path
        description: The id of the todo
        required: true
        schema:
          format: int64
          type: integer
        example: "42"
      responses:
        200:
          description: The requested Todo
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Todo'
```



```yaml
openapi: 3.0.1
info:
  title: Todo Backend
  version
paths:
  /api/{id}:
    get:
      summary: Get the one todo
      operationId: todoGetOne
      parameters:
      - name: id
        in: path
        description: The id of the todo
        required: true
        schema:
          format: int64
          type: integer
        example: "42"
      responses:
        200:
          description: The requested Todo
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Todo'
```

get, post, patch, delete, put, options, head

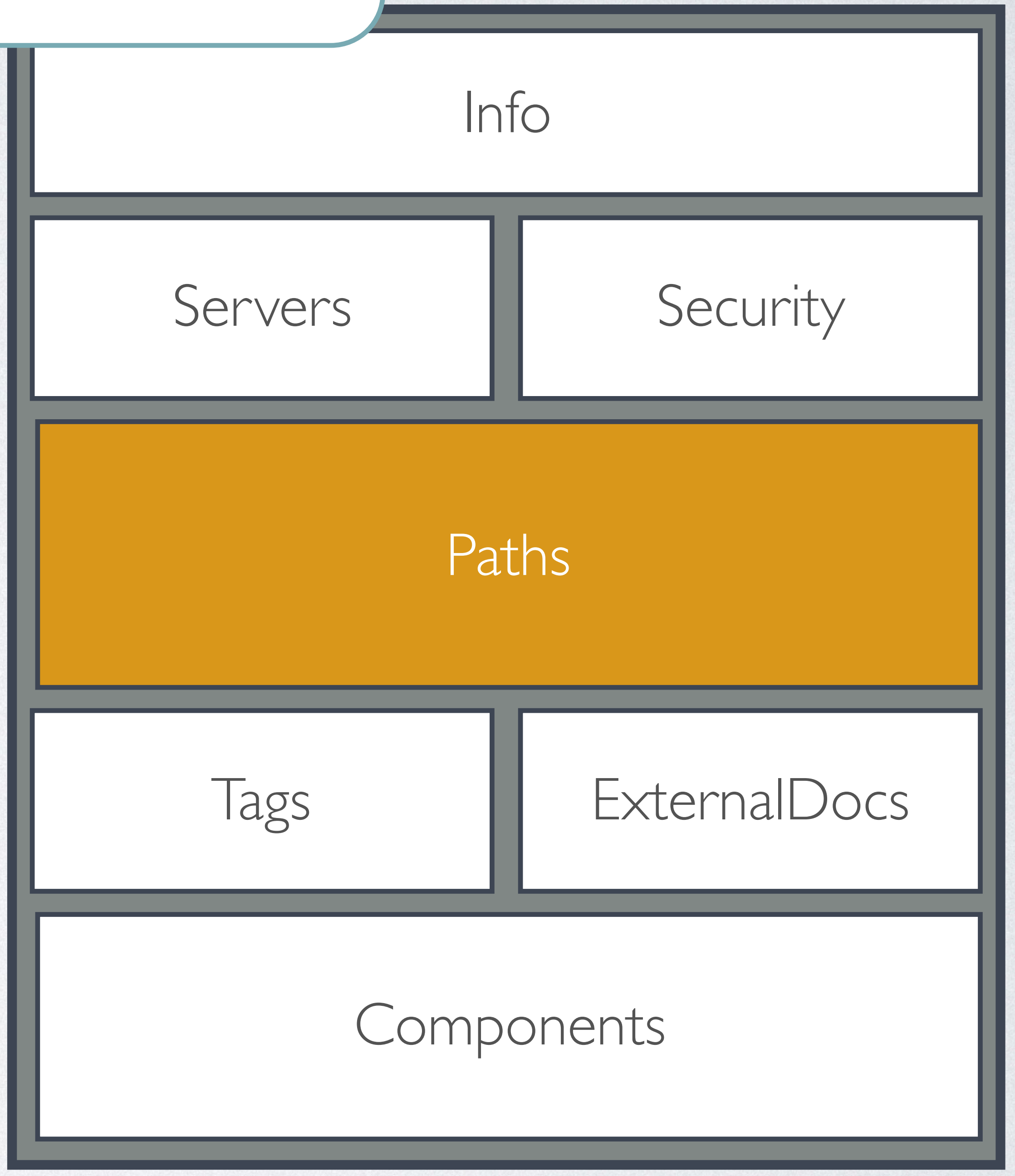


```yaml
openapi: 3.0.1
info:
  title: Todo Backend
  version: "1.0"
paths:
  /api/{id}:
    get:
      summary: Get the one todo
      operationId: todoGetOne
      parameters:
      - name: id
        in: path
        description: The id of the todo
        required: true
        schema:
          format: int64
          type: integer
        example: "42"
      responses:
        200:
          description: The requested Todo
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Todo'
```

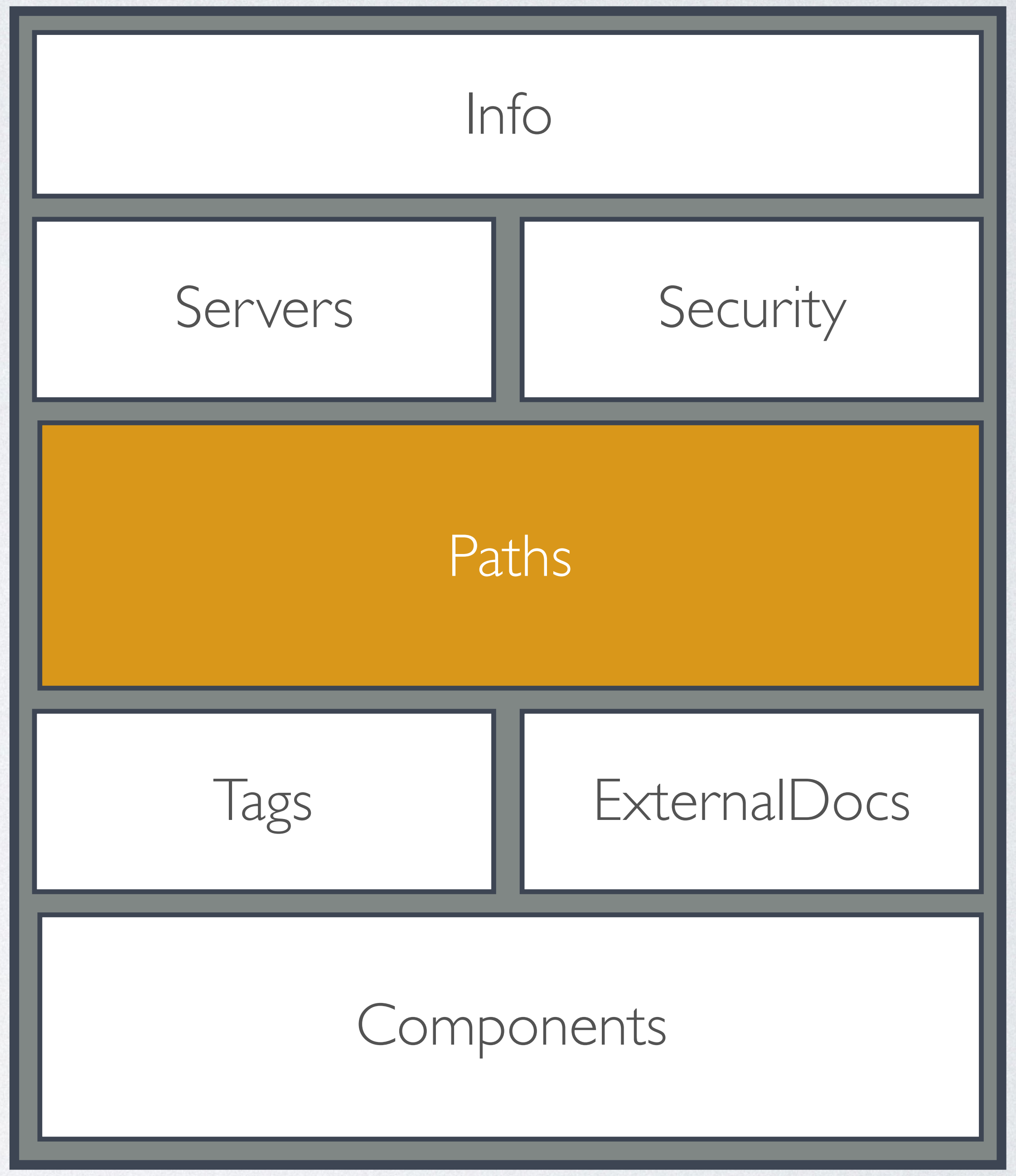
parameters (query, path…)

request body

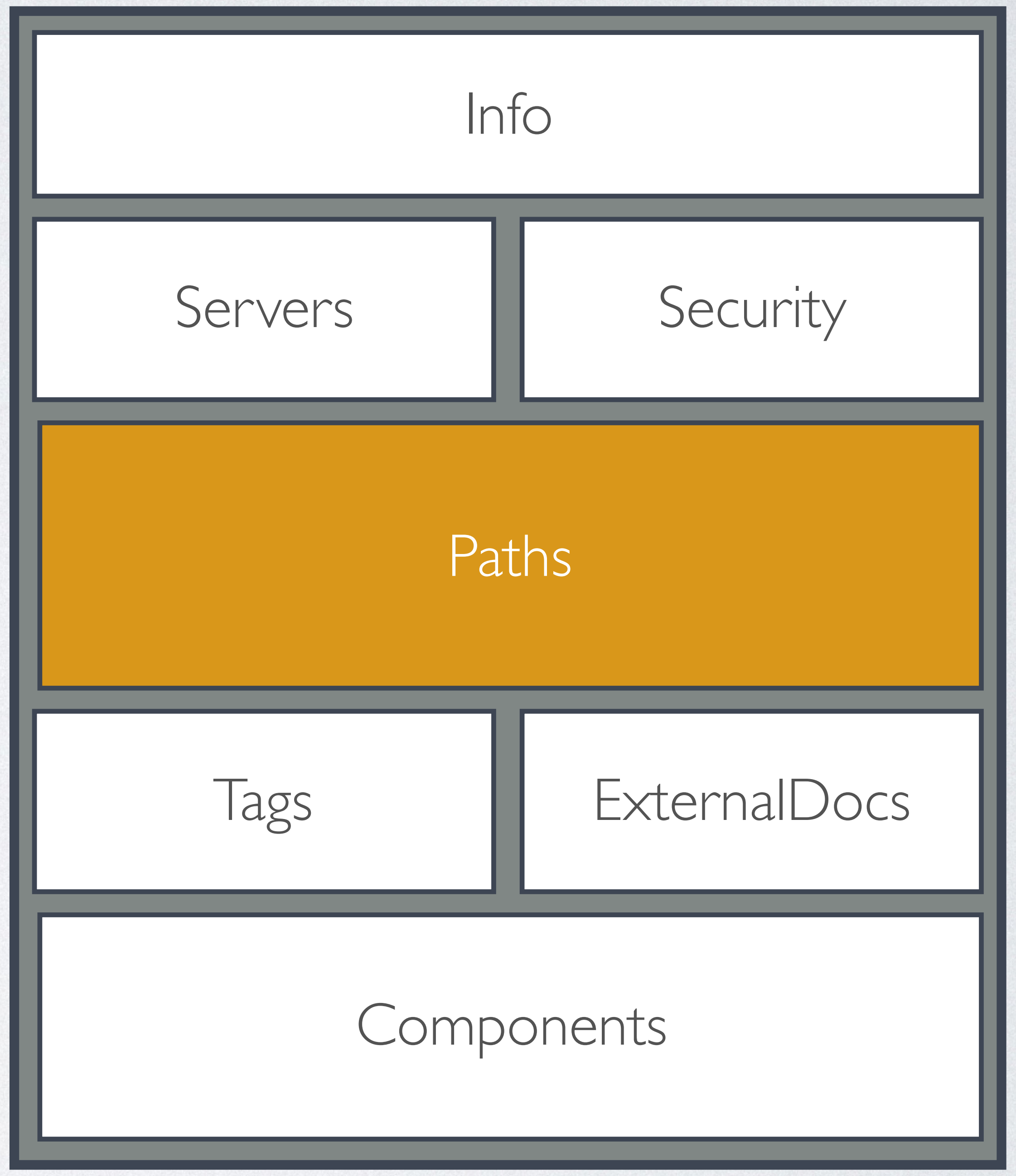
responses



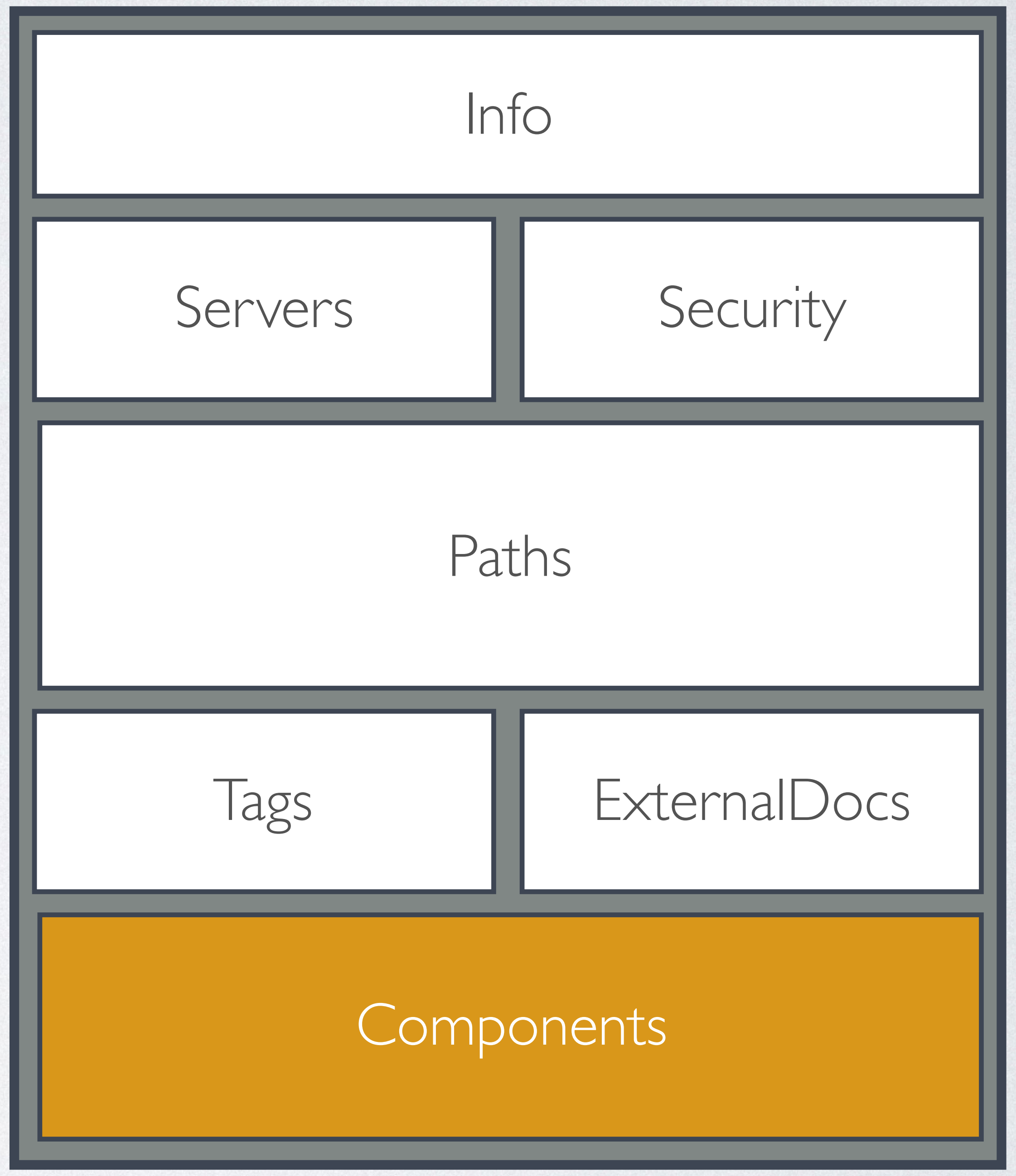
```yaml
openapi: 3.0.1
info:
  title: Todo Backend
  version: "1.0"
paths:
  /api/{id}:
    get:
      summary: Get the one todo
      operationId: todoGetOne
      parameters:
      - name: id
        in: path
        description: The id of the todo
        required: true
        schema:
          format: int64
          type: integer
        example: "42"
      responses:
        200:
          description: The requested Todo
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Todo'
```



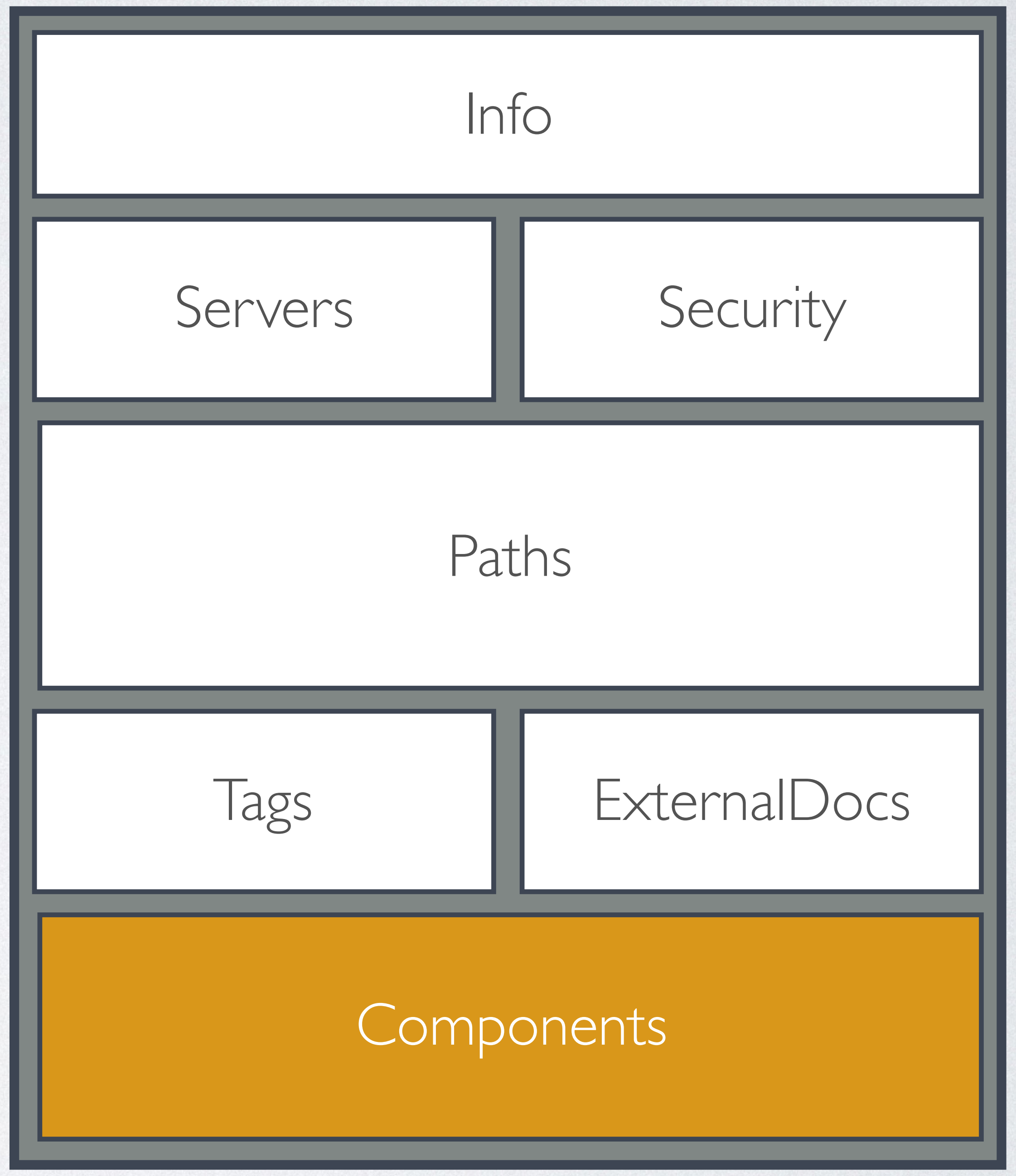
```yaml
openapi: 3.0.1
info:
  title: Todo Backend
  version: "1.0"
paths:
  /api/{id}:
    get:
      summary: Get the one todo
      operationId: todoGetOne
      parameters:
      - name: id
        in: path
        description: The id of the todo
        required: true
        schema:
          format: int64
          type: integer
        example: "42"
      responses:
        200:
          description: The requested Todo
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Todo'
```



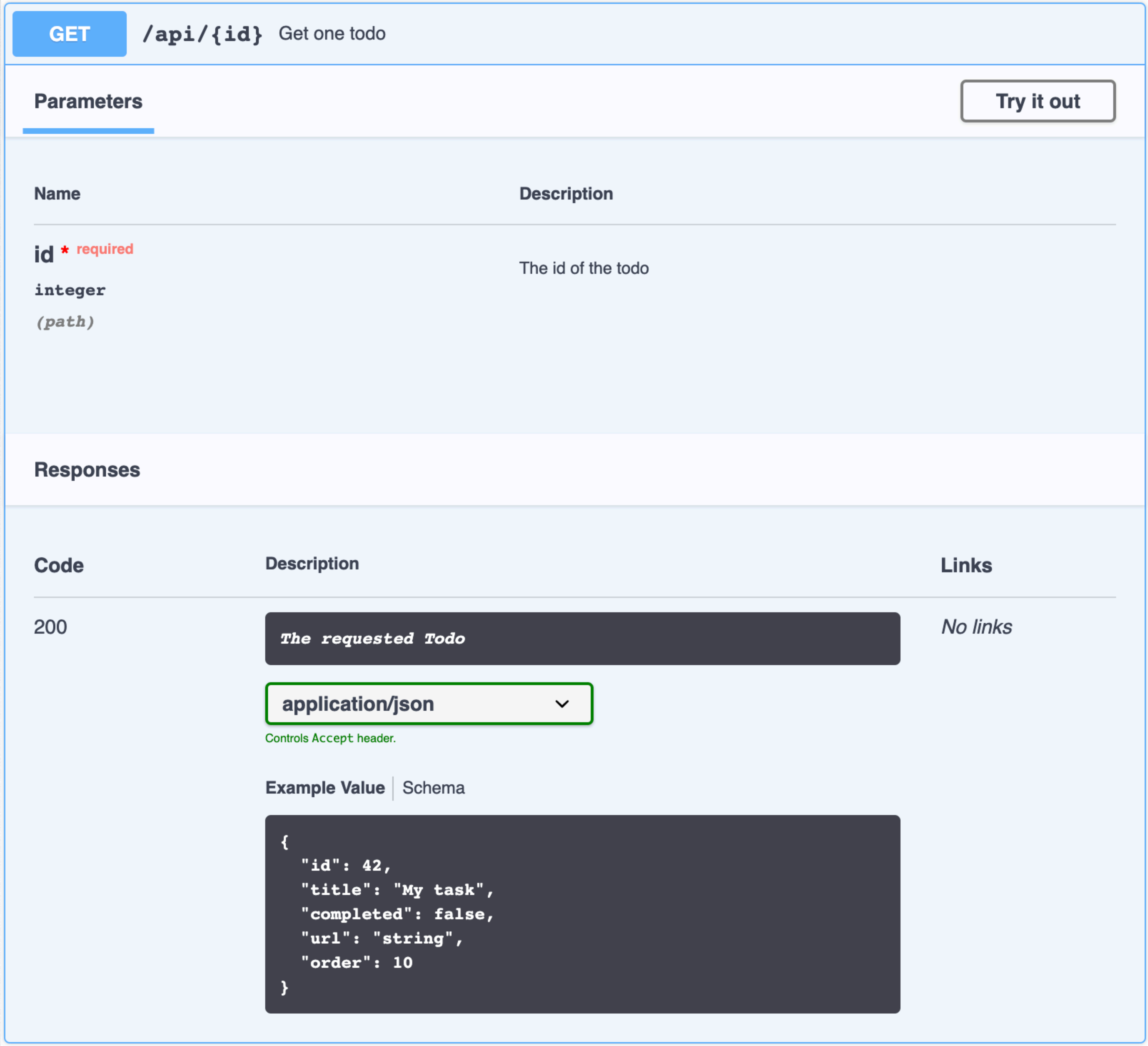
```yaml
components:
  schemas:
    Todo:
      description: Object representing a Todo
      type: object
      properties:
        id:
          description: id of the entity
          format: int64
          type: integer
          example: "42"
        title:
          description: title of the todo
          type: string
          example: My task
        completed:
          description: whether the todo is completed or not
          type: boolean
          example: "false"
        url:
          description: url associated with the todo
          type: string
        order:
          format: int32
          description: order in the priority list
          type: integer
          example: "10"
```



```yaml
components:
  schemas:
    Todo:
      description: Object representing a Todo
      type: object
      properties:
        id:
          description: id of the entity
          format: int64
          type: integer
          example: "42"
        title:
          description: title of the todo
          type: string
          example: My task
        completed:
          description: whether the todo is completed or not
          type: boolean
          example: "false"
        url:
          description: url associated with the todo
          type: string
        order:
          format: int32
          description: order in the priority list
          type: integer
          example: "10"
```



Swagger UI



`/api/{id}` Get one todo

Parameters

Name	Description
id * required integer *(path)*	The id of the todo

Responses

Code	Description	Links
200	*The requested Todo*	*No links*

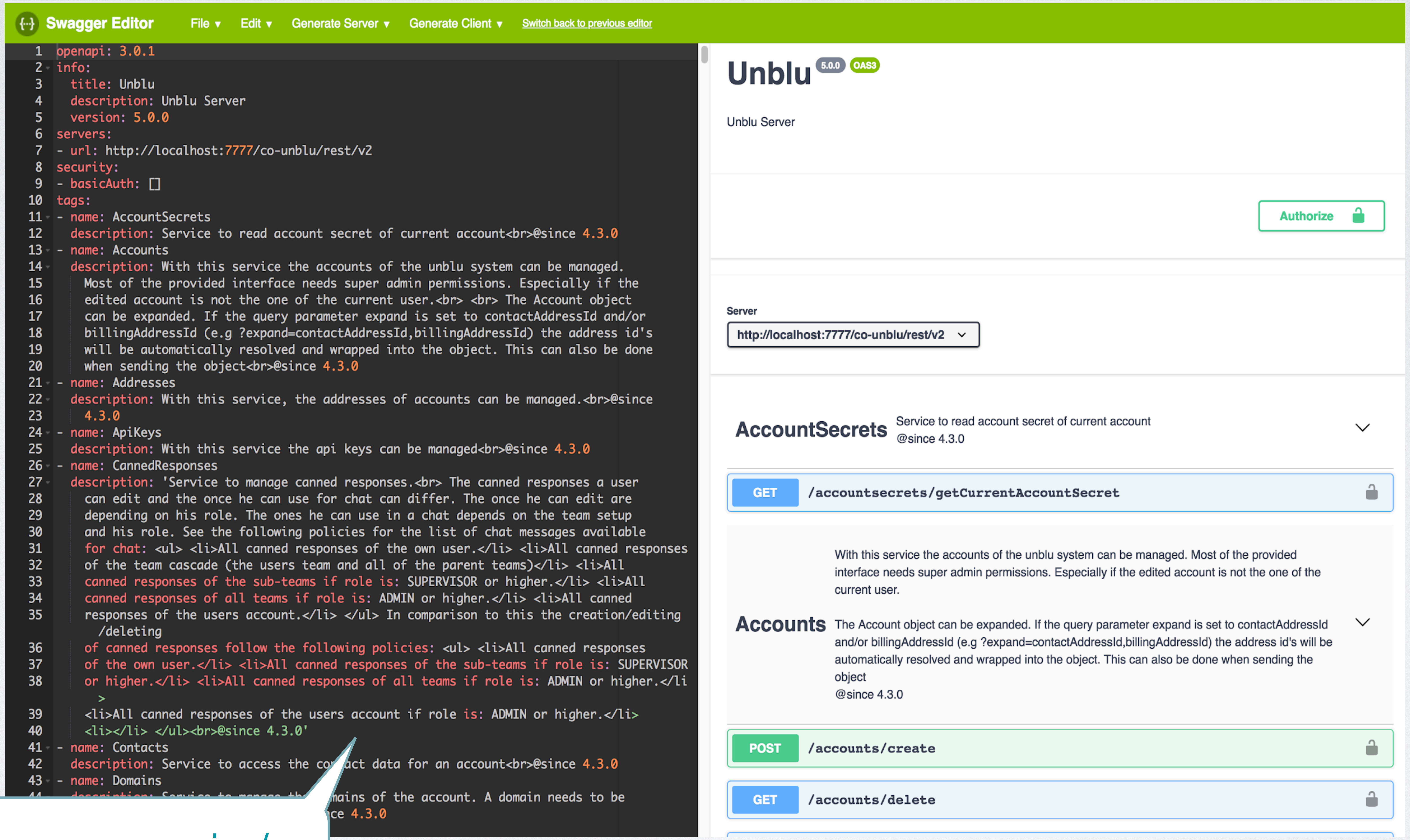
application/json

Controls Accept header.

Example Value | Schema

```
{
  "id": 42,
  "title": "My task",
  "completed": false,
  "url": "string",
  "order": 10
}
```

Swagger Online Editor



```yaml
openapi: 3.0.1
info:
  title: Unblu
  description: Unblu Server
  version: 5.0.0
servers:
- url: http://localhost:7777/co-unblu/rest/v2
security:
- basicAuth: []
tags:
- name: AccountSecrets
  description: Service to read account secret of current account<br>@since 4.3.0
- name: Accounts
  description: With this service the accounts of the unblu system can be managed.
    Most of the provided interface needs super admin permissions. Especially if the
    edited account is not the one of the current user.<br> <br> The Account object
    can be expanded. If the query parameter expand is set to contactAddressId and/or
    billingAddressId (e.g ?expand=contactAddressId,billingAddressId) the address id's
    will be automatically resolved and wrapped into the object. This can also be done
    when sending the object<br>@since 4.3.0
- name: Addresses
  description: With this service, the addresses of accounts can be managed.<br>@since
    4.3.0
- name: ApiKeys
  description: With this service the api keys can be managed<br>@since 4.3.0
- name: CannedResponses
  description: 'Service to manage canned responses.<br> The canned responses a user
    can edit and the once he can use for chat can differ. The once he can edit are
    depending on his role. The ones he can use in a chat depends on the team setup
    and his role. See the following policies for the list of chat messages available
    for chat: <ul> <li>All canned responses of the own user.</li> <li>All canned responses
    of the team cascade (the users team and all of the parent teams)</li> <li>All
    canned responses of the sub-teams if role is: SUPERVISOR or higher.</li> <li>All
    canned responses of all teams if role is: ADMIN or higher.</li> <li>All canned
    responses of the users account.</li> </ul> In comparison to this the creation/editing
      /deleting
    of canned responses follow the following policies: <ul> <li>All canned responses
    of the own user.</li> <li>All canned responses of the sub-teams if role is: SUPERVISOR
    or higher.</li> <li>All canned responses of all teams if role is: ADMIN or higher.</li>
    <li>All canned responses of the users account if role is: ADMIN or higher.</li>
    <li></li> </ul><br>@since 4.3.0'
- name: Contacts
  description: Service to access the co    act data for an account<br>@since 4.3.0
- name: Domains
  description: Service to manage th     mains of the account. A domain needs to be
                                     ce 4.3.0
```

Unblu 5.0.0 OAS3

Unblu Server

Authorize

Server

http://localhost:7777/co-unblu/rest/v2

AccountSecrets Service to read account secret of current account
@since 4.3.0

GET /accountsecrets/getCurrentAccountSecret

Accounts With this service the accounts of the unblu system can be managed. Most of the provided interface needs super admin permissions. Especially if the edited account is not the one of the current user.

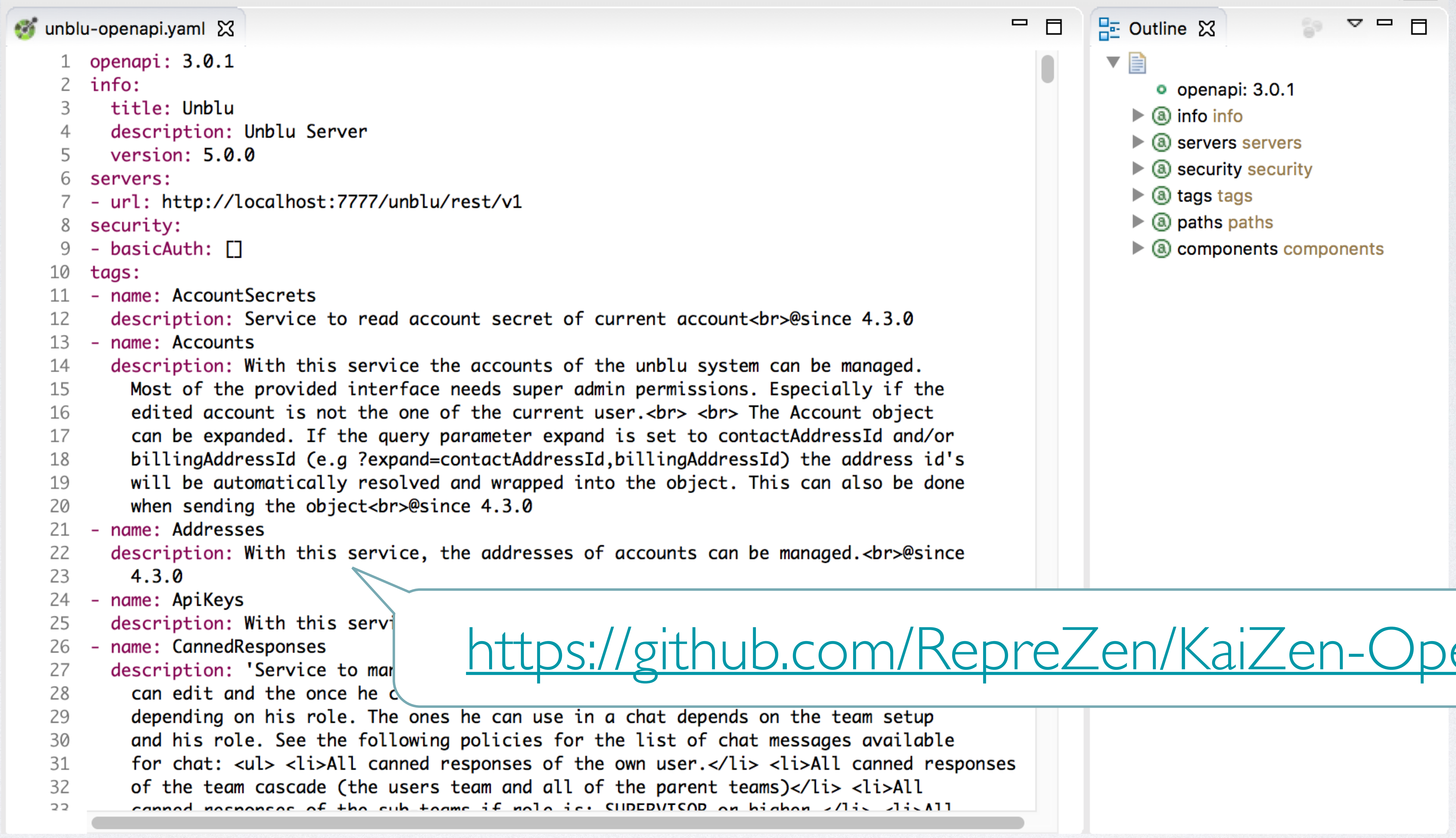
The Account object can be expanded. If the query parameter expand is set to contactAddressId and/or billingAddressId (e.g ?expand=contactAddressId,billingAddressId) the address id's will be automatically resolved and wrapped into the object. This can also be done when sending the object
@since 4.3.0

POST /accounts/create

GET /accounts/delete

https://editor.swagger.io/

Eclipse IDE plugin: KaiZen-OpenAPI-Editor

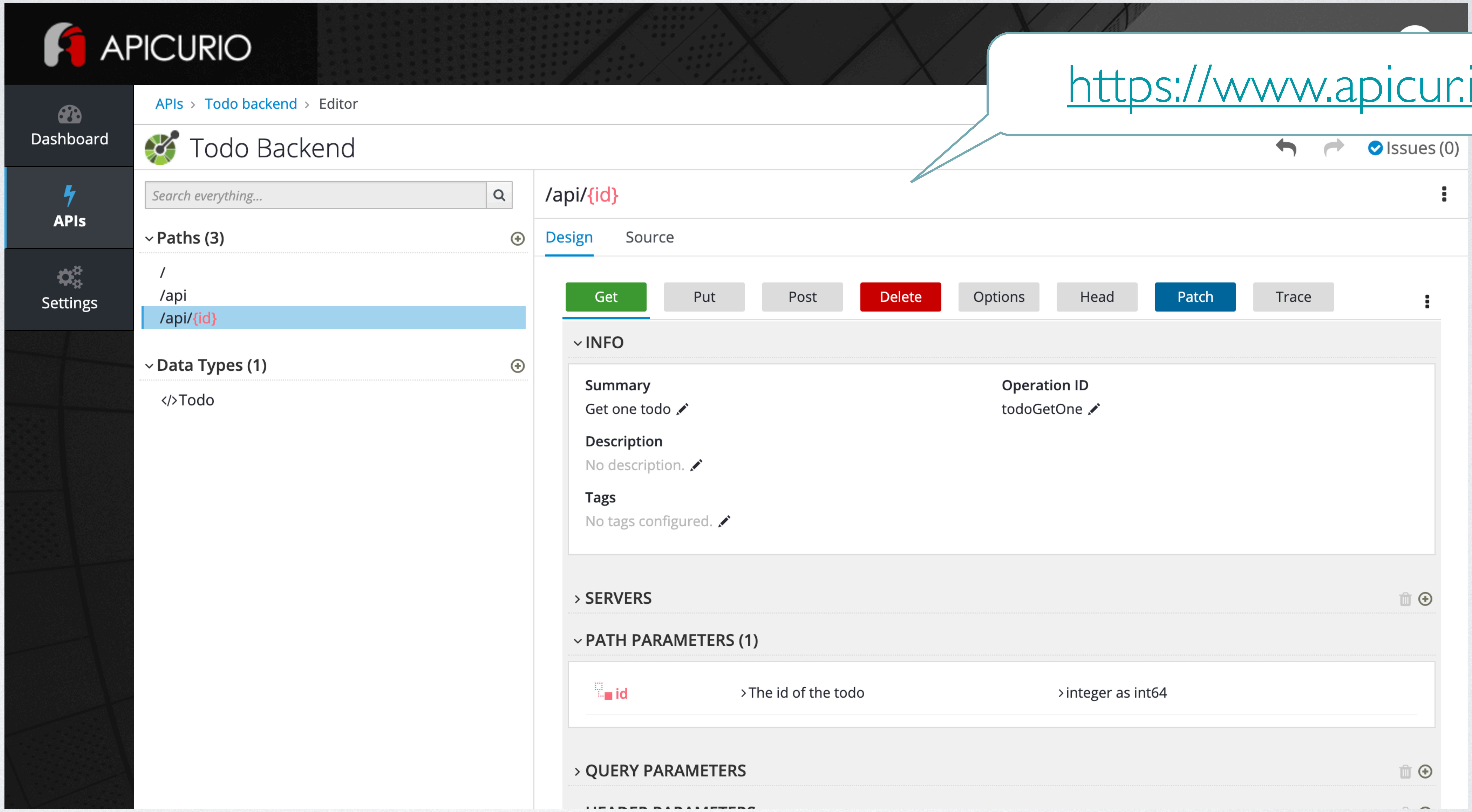


```
openapi: 3.0.1
info:
  title: Unblu
  description: Unblu Server
  version: 5.0.0
servers:
- url: http://localhost:7777/unblu/rest/v1
security:
- basicAuth: []
tags:
- name: AccountSecrets
  description: Service to read account secret of current account<br>@since 4.3.0
- name: Accounts
  description: With this service the accounts of the unblu system can be managed.
    Most of the provided interface needs super admin permissions. Especially if the
    edited account is not the one of the current user.<br> <br> The Account object
    can be expanded. If the query parameter expand is set to contactAddressId and/or
    billingAddressId (e.g ?expand=contactAddressId,billingAddressId) the address id's
    will be automatically resolved and wrapped into the object. This can also be done
    when sending the object<br>@since 4.3.0
- name: Addresses
  description: With this service, the addresses of accounts can be managed.<br>@since
    4.3.0
- name: ApiKeys
  description: With this servi
- name: CannedResponses
  description: 'Service to mar
    can edit and the once he c
    depending on his role. The ones he can use in a chat depends on the team setup
    and his role. See the following policies for the list of chat messages available
    for chat: <ul> <li>All canned responses of the own user.</li> <li>All canned responses
    of the team cascade (the users team and all of the parent teams)</li> <li>All
    canned responses of the sub teams if role is: SUPERVISOR or higher.</li> <li>All
```

```
openapi: 3.0.1
info info
servers servers
security security
tags tags
paths paths
components components
```

https://github.com/RepreZen/KaiZen-OpenAPI-Editor

APICURIO



APIs › Todo backend › Editor

Todo Backend

Paths (3)
- /
- /api
- /api/{id}

Data Types (1)
- </>Todo

/api/{id}

INFO

Summary: Get one todo

Operation ID: todoGetOne

Description: No description.

Tags: No tags configured.

SERVERS

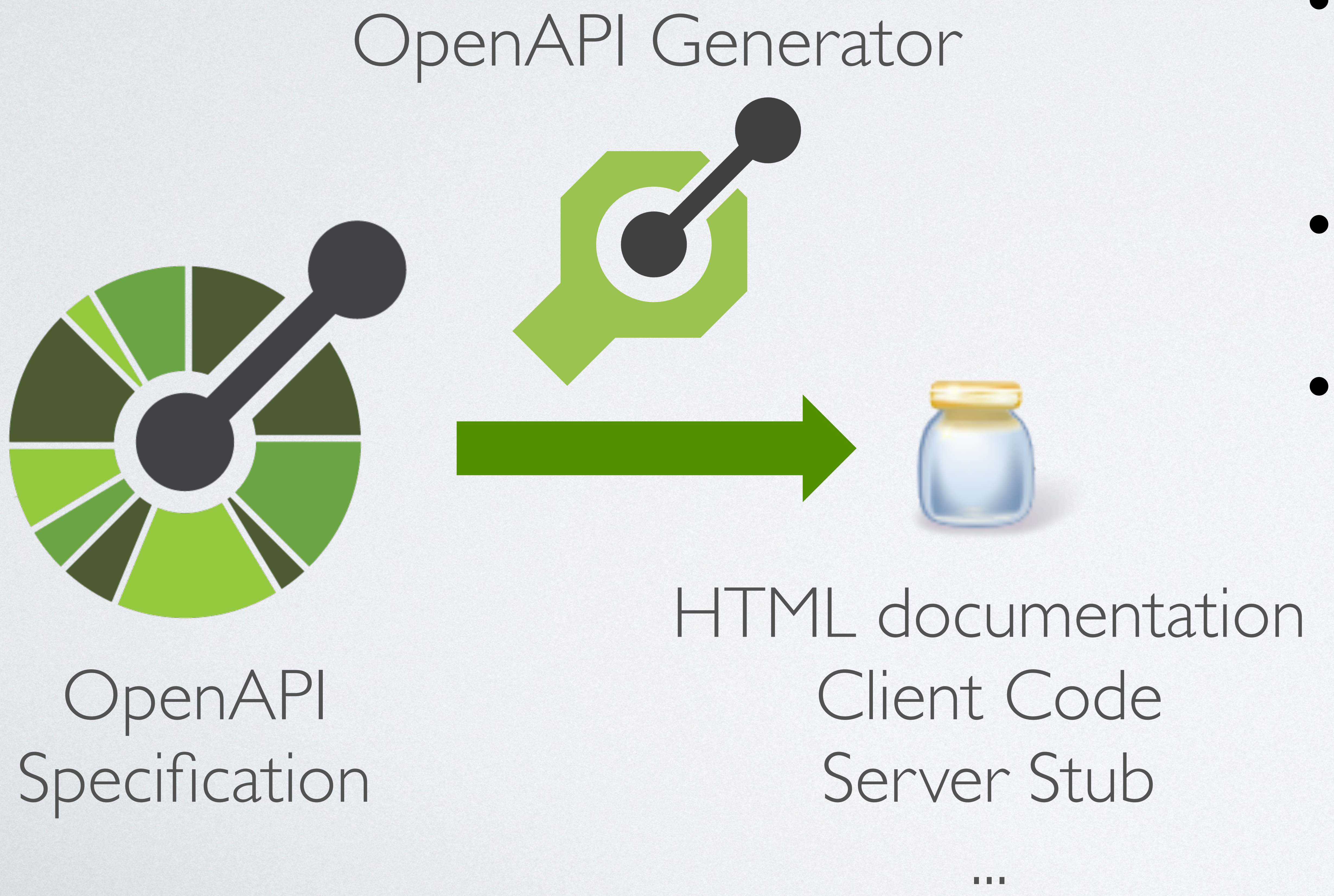
PATH PARAMETERS (1)

id	› The id of the todo	› integer as int64

QUERY PARAMETERS

https://www.apicur.io/

Code generator: OpenAPI-Generator



- **Open Source** (Apache 2.0 License)
- Hosted on **GitHub**: https://github.com/OpenAPITools/openapi-generator
- Java code & mustache templates
- Fork of Swagger-Codegen

MicroProfile OpenAPI

MicroProfile OpenAPI

- specification documentation
- code: annotations & models & programming interfaces (implementation is required)
- to be used on top of JAX-RS

Annotations

```
@PATCH
@Path("/{id}")
@Operation(
        operationId = "todoUpdate",
        summary = "Update an exsiting todo")
@APIResponses(
        value = @APIResponse(
                        responseCode = "200",
                        description = "The updated Todo",
                        content = @Content(
                                schema = @Schema(
                                        implementation = Todo.class))))
public Response update(@RequestBody(
        description = "The todo to update",
        content = @Content(
            schema = @Schema(
                implementation = Todo.class))) Todo todo,
    @PathParam("id") @Parameter(
        description = "The id of the todo",
        name = "id",
        example = "42",
        required = true,
        schema = @Schema(type = SchemaType.INTEGER, format = "int64")) Long id)
```

Annotations

JAX-RS

```
@PATCH
@Path("/{id}")
@Operation(
        operationId = "todoUpdate",
        summary = "Update an exsiting todo")
@APIResponses(
        value = @APIResponse(
                        responseCode = "200",
                        description = "The updated Todo",
                        content = @Content(
                                schema = @Schema(
                                        implementation = Todo.class))))
public Response update(@RequestBody(
        description = "The todo to update",
        content = @Content(
            schema = @Schema(
                implementation = Todo.class))) Todo todo,
    @PathParam("id") @Parameter(
        description = "The id of the todo",
        name = "id",
        example = "42",
        required = true,
        schema = @Schema(type = SchemaType.INTEGER, format = "int64")) Long id)
```

Annotations

MicroProfile OpenAPI

```
@PATCH
@Path("/{id}")
@Operation(
        operationId = "todoUpdate",
        summary = "Update an existing todo")
@APIResponses(
        value = @APIResponse(
                        responseCode = "200",
                        description = "The updated Todo",
                        content = @Content(
                                schema = @Schema(
                                        implementation = Todo.class))))
public Response update(@RequestBody(
        description = "The todo to update",
        content = @Content(
            schema = @Schema(
                implementation = Todo.class))) Todo todo,
    @PathParam("id") @Parameter(
        description = "The id of the todo",
        name = "id",
        example = "42",
        required = true,
        schema = @Schema(type = SchemaType.INTEGER, format = "int64")) Long id)
```

Models

- Interfaces to represent an OpenAPI specification
- Builder pattern
- Typed, instead of looking at a JSON/YAML tree
- `package org.eclipse.microprofile.openapi`

Models

```
createOpenAPI()
    .paths(
        createPaths()
            .addPathItem("/api/{id}", createPathItem()
                .GET(
                    createOperation()
                        .operationId("todoGetOne")
                        .summary("Get the one todo")
                        .addParameter(createParameter()
                            .name("id")
                            .in(In.PATH)
                            .description("The id of the todo")
                            .required(true)
                            .schema(createSchema()
                                .type(SchemaType.INTEGER)
                                .format("int64"))
                            .example(42))
                        .responses(
                            createAPIResponses()
                                .addAPIResponse(
                                    "200", createAPIResponse()
                                        .description("The requested Todo")
                                        .content(createContent()
                                            .addMediaType("application/json", createMediaType()
                                            .schema(createSchema()
                                                .ref("#/components/schemas/Todo"))))))))));
```

Serving the OpenAPI Spec

- `GET http://<host>:<port>/openapi`
- Format (`JSON` or `YAML`) can be specified with the Accept header
- Document is generated based on:
  - Result returned by `OASModelReader.buildModel()`
  - Static OpenAPI file
  - Process annotations
  - Filter model via `OASFilter`

Serving the OpenAPI Spec

Annotation Based

- JAX-RS and MP-OpenAPI annotations are leading the document
- Code is verbose

Static file

- Spec is easier to write
- A mechanism to keep JAX-RS annotations in sync with the OpenAPI Specification is necessary

Wrapping other parsers

- Some java tools already exist to parse and manipulate OpenAPI specifications:

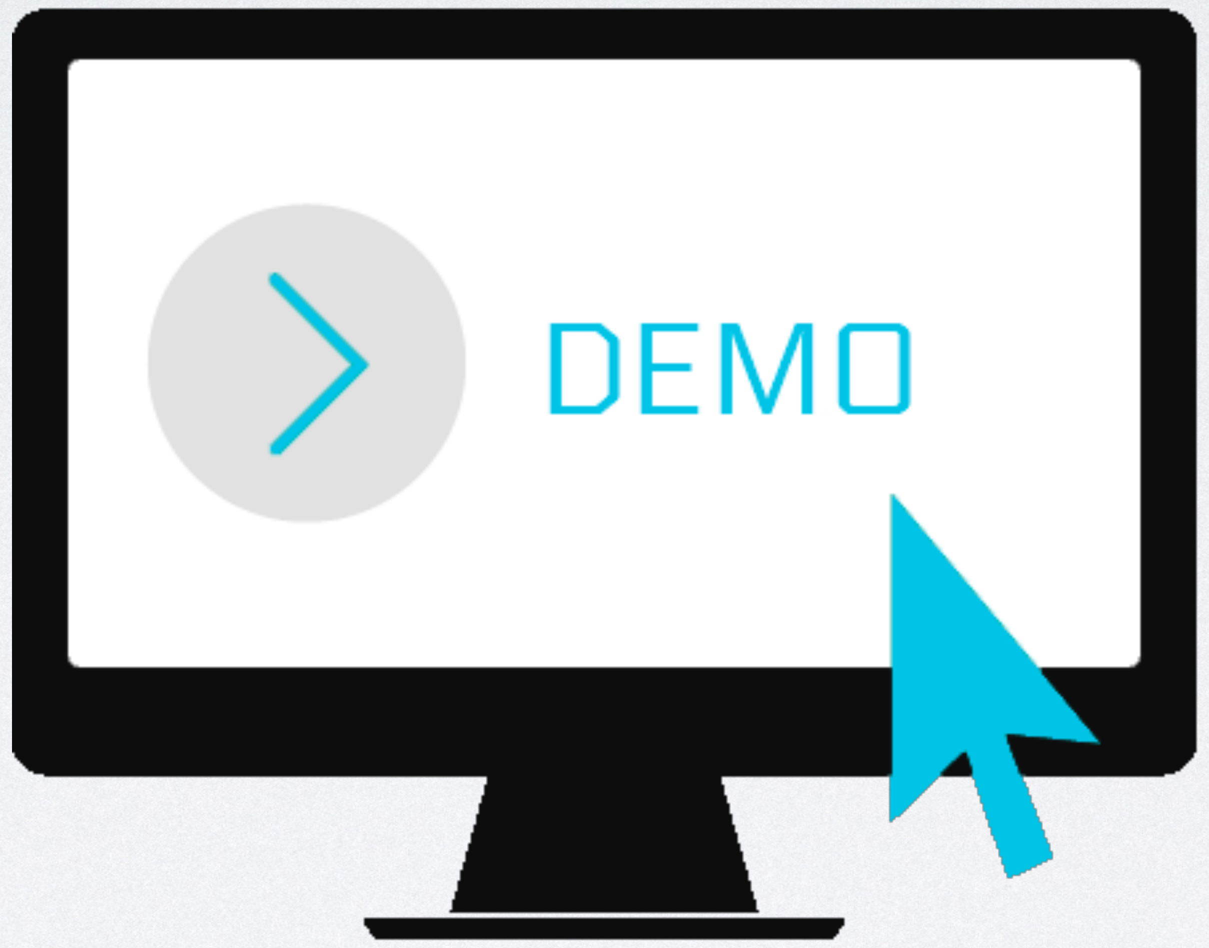
Swagger-Parser

KaiZen OpenAPI Parser

- When you write a tool, you would like program against an API to be able to exchange the underlying implementation.

- Model interfaces of MicroProfile OpenAPI can be this layer.

- See https://github.com/OpenAPITools/empoa


DEMO

2018 demo

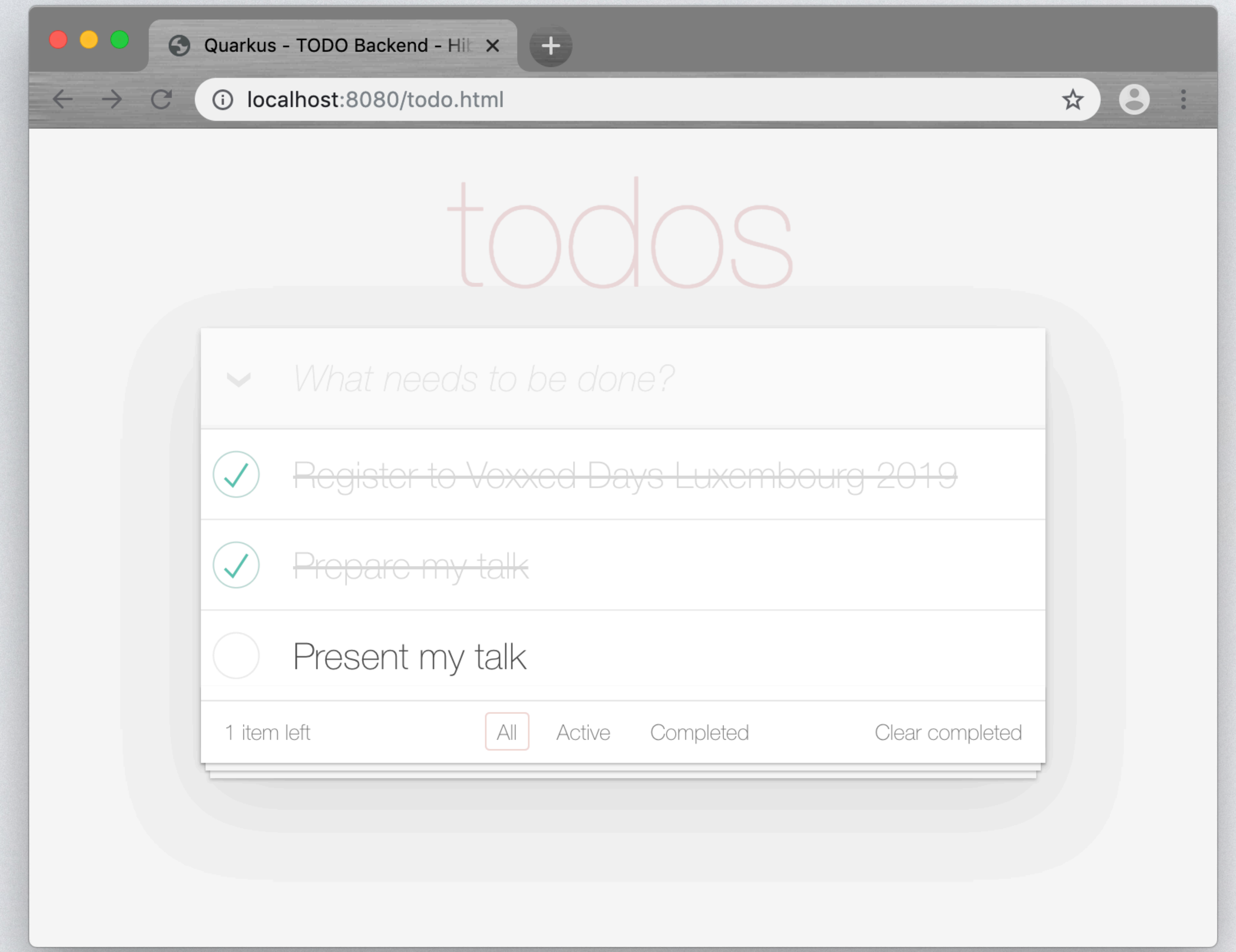
THORNTAIL

(a.k.a WildFly Swarm)
https://thorntail.io/

Open Liberty

https://openliberty.io/

Demo: todo-backend


Quarkus - TODO Backend - Hib

localhost:8080/todo.html

todos

What needs to be done?

~~Register to Voxxed Days Luxembourg 2019~~

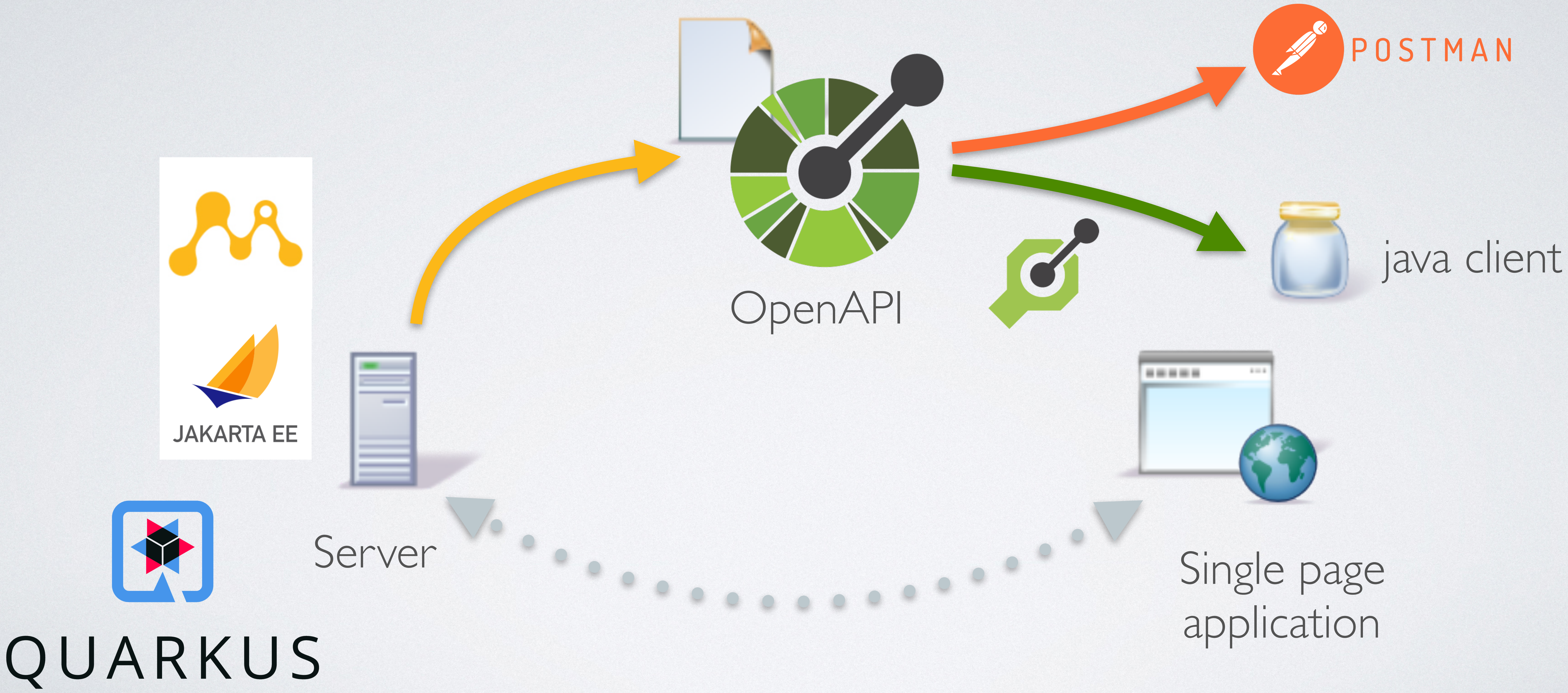
~~Prepare my talk~~

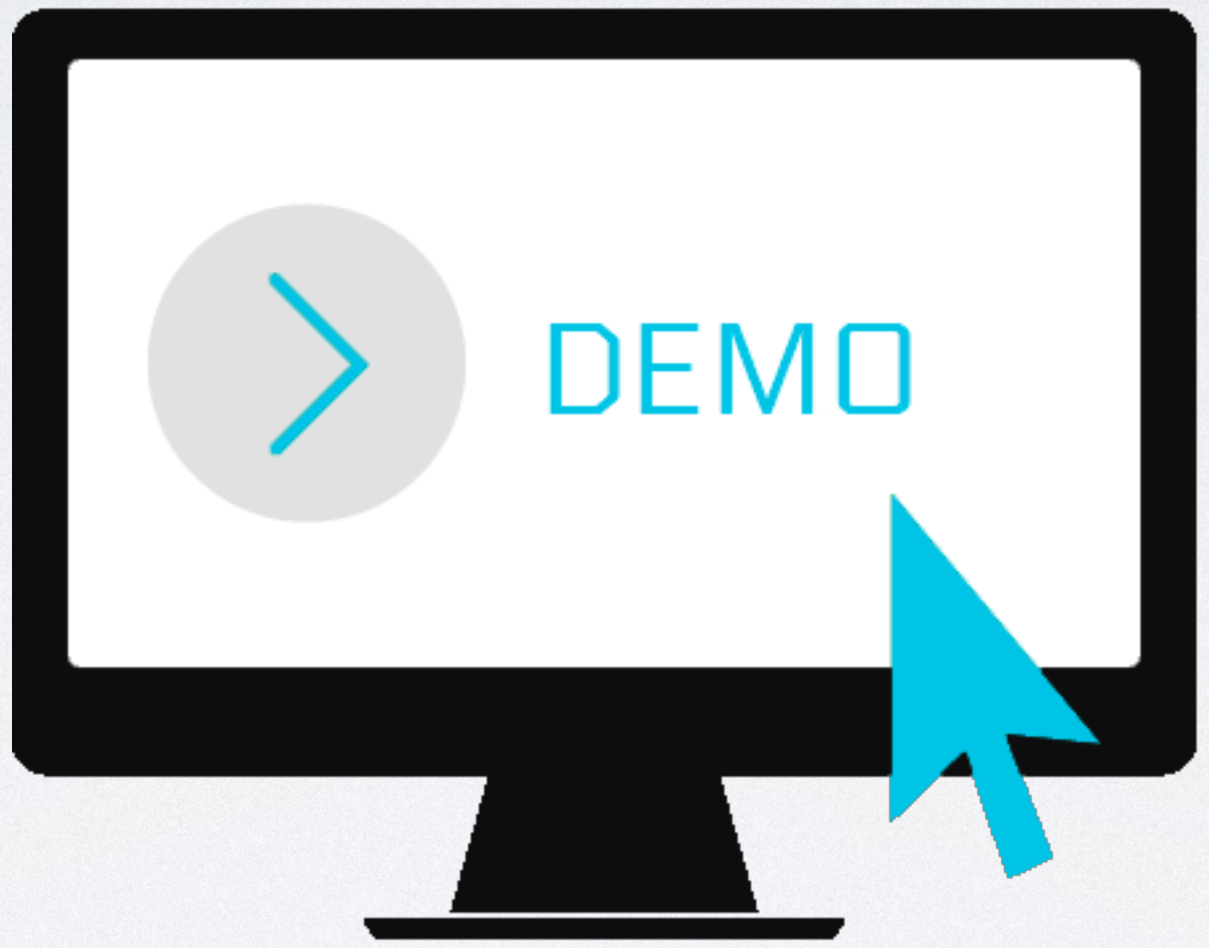
Present my talk

1 item left All Active Completed Clear completed

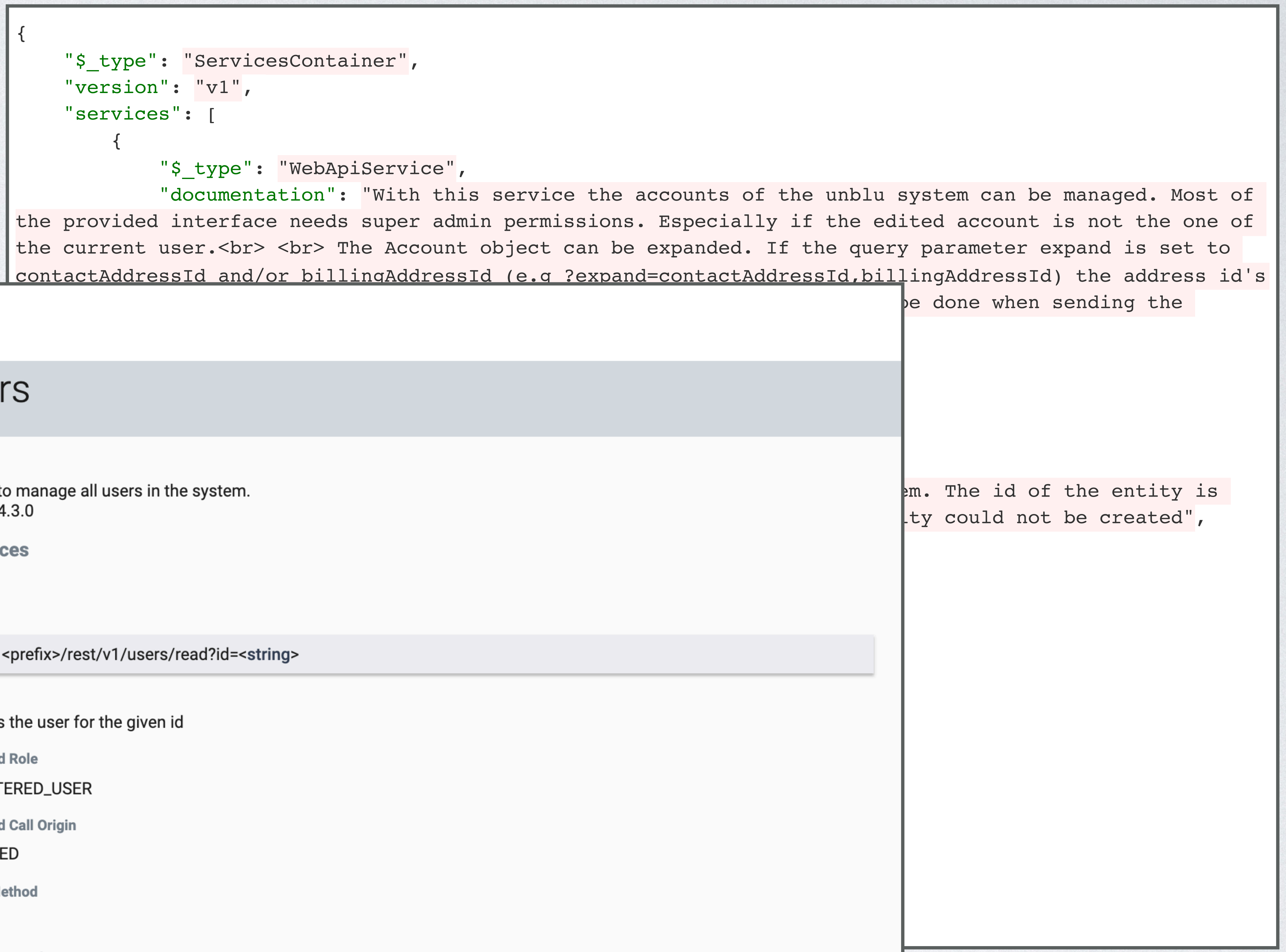
QUARKUS

Code first approach




DEMO

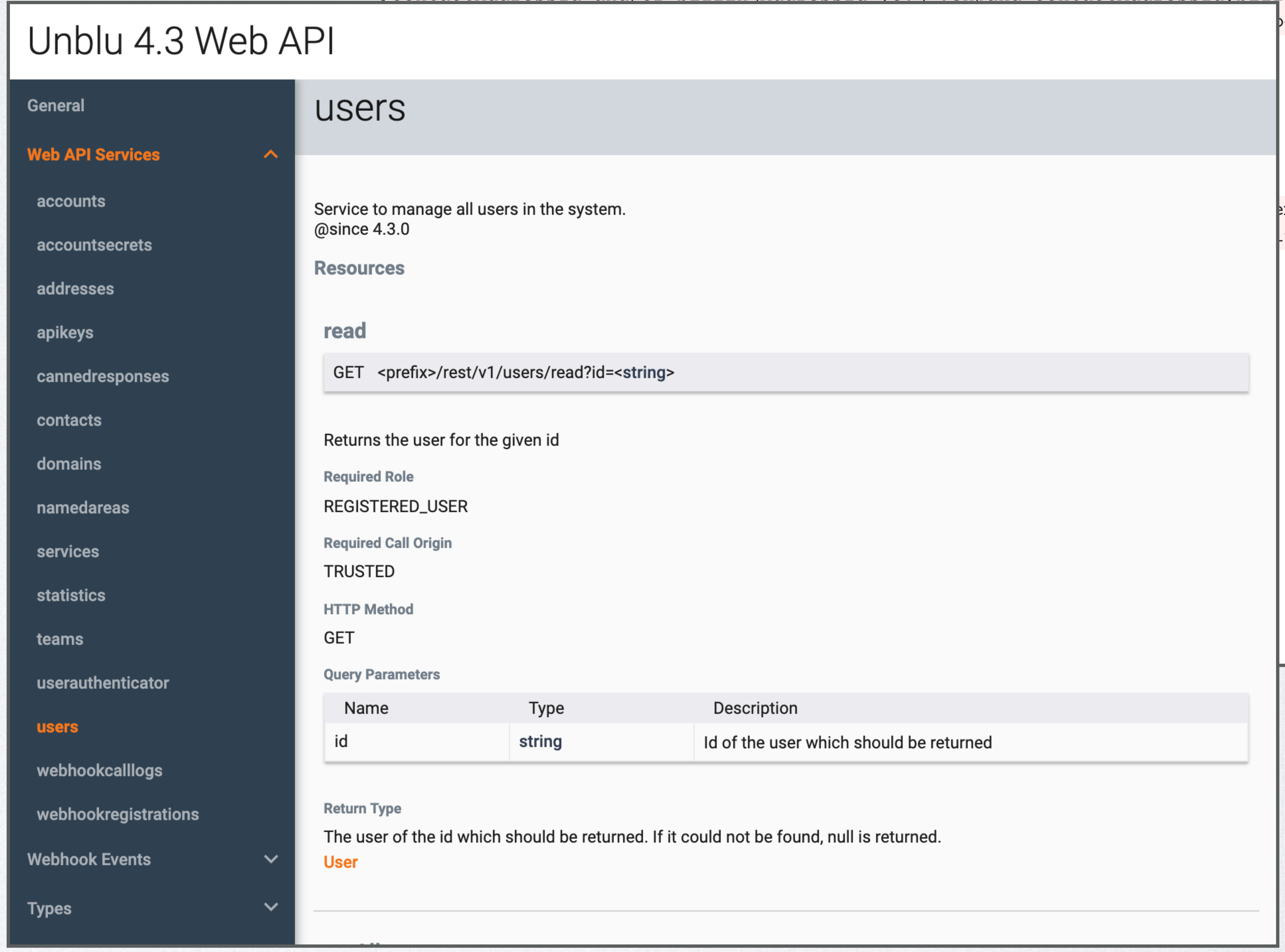
At the beginning…



```
{
    "$_type": "ServicesContainer",
    "version": "v1",
    "services": [
        {
            "$_type": "WebApiService",
            "documentation": "With this service the accounts of the unblu system can be managed. Most of
the provided interface needs super admin permissions. Especially if the edited account is not the one of
the current user.<br> <br> The Account object can be expanded. If the query parameter expand is set to
contactAddressId and/or billingAddressId (e.g ?expand=contactAddressId,billingAddressId) the address id's
                                                                              e done when sending the

                                                                              m. The id of the entity is
                                                                              ty could not be created",
```

Custom JSON



Unblu 4.3 Web API

users

Service to manage all users in the system.
@since 4.3.0

Resources

read

GET <prefix>/rest/v1/users/read?id=<string>

Returns the user for the given id

Required Role
REGISTERED_USER

Required Call Origin
TRUSTED

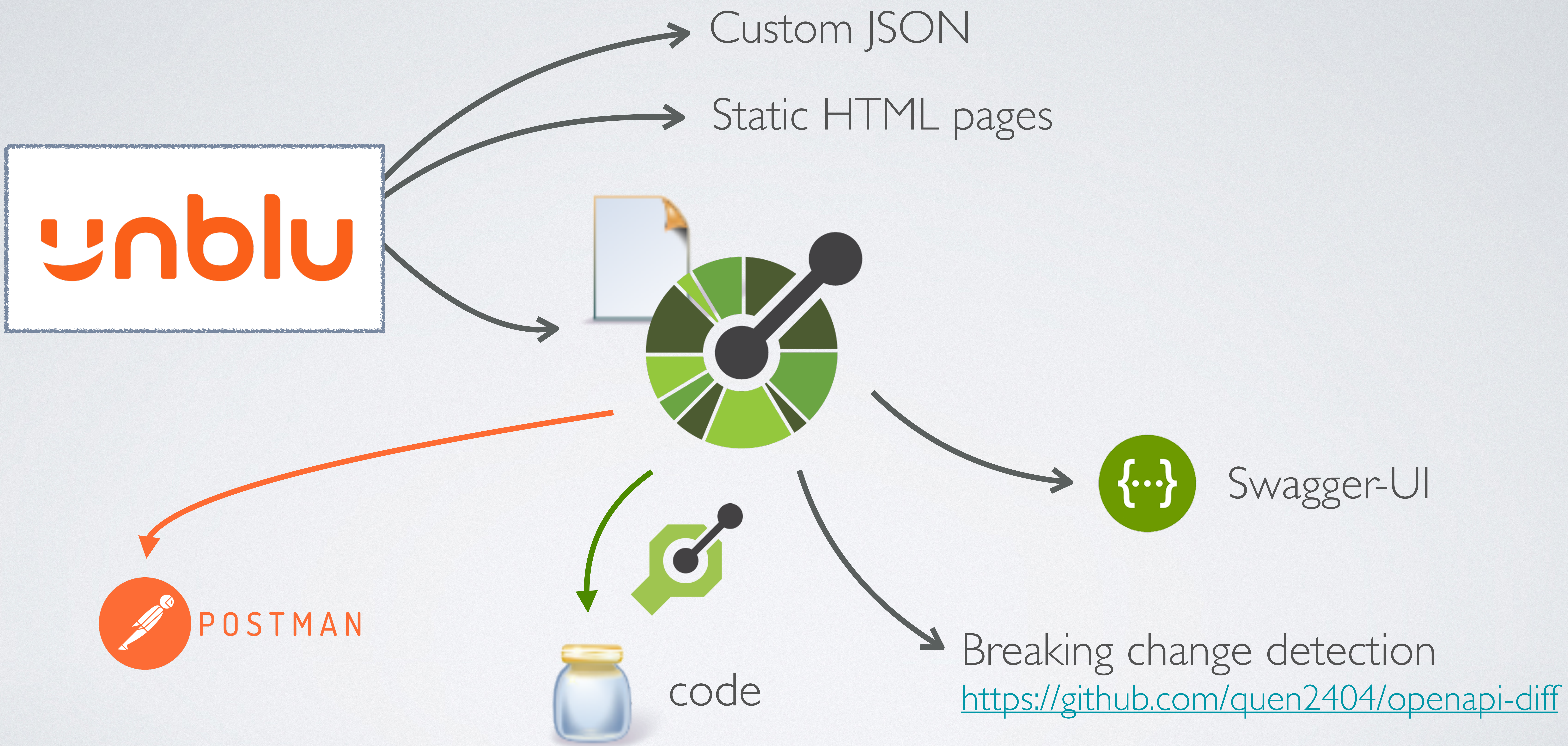
HTTP Method
GET

Query Parameters

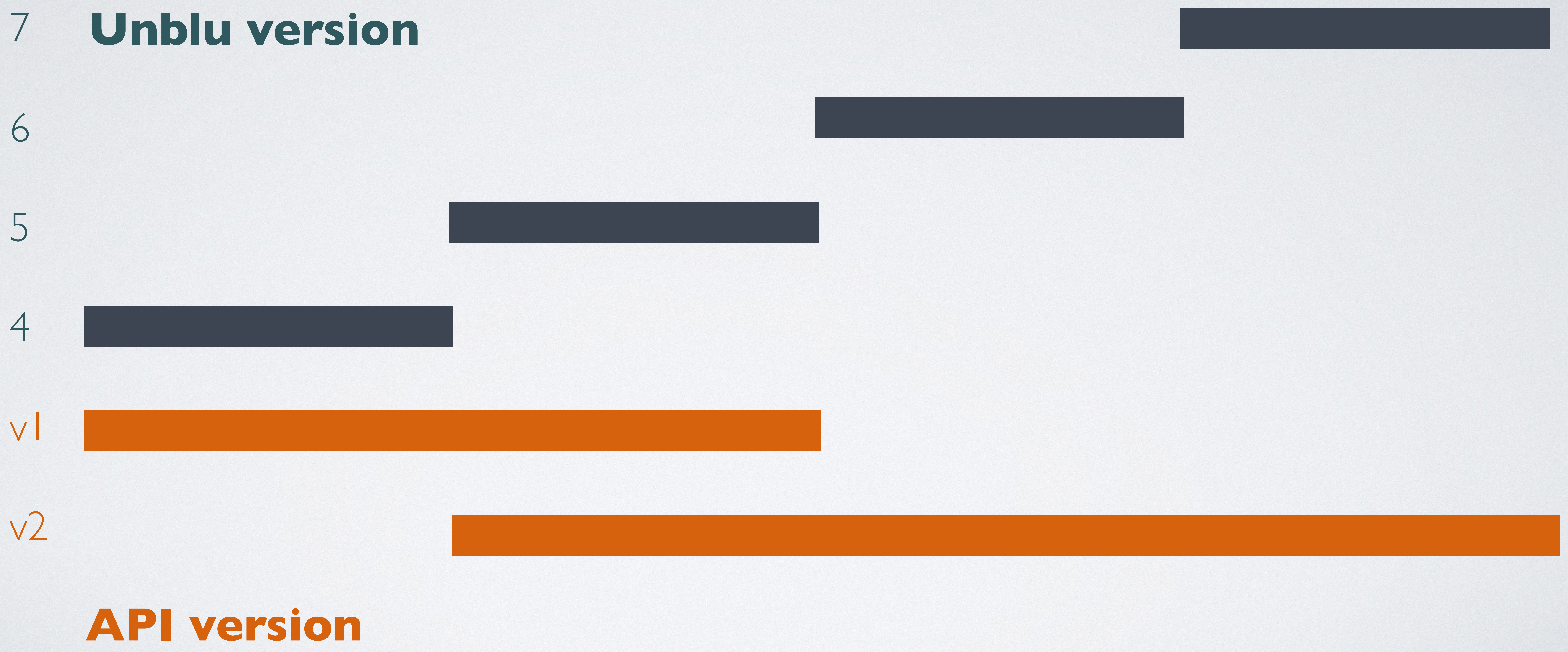
Name	Type	Description
id	string	Id of the user which should be returned

Return Type
The user of the id which should be returned. If it could not be found, null is returned.
User

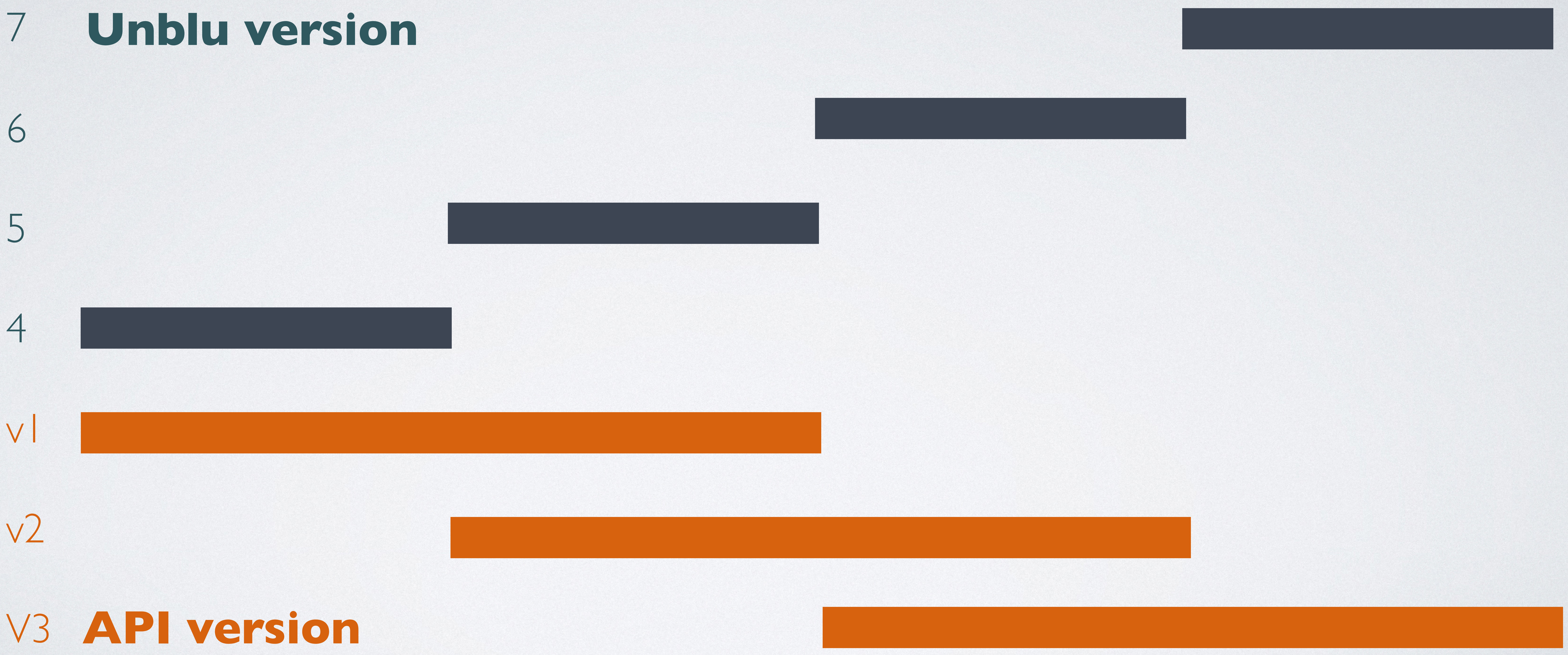
Static HTML pages

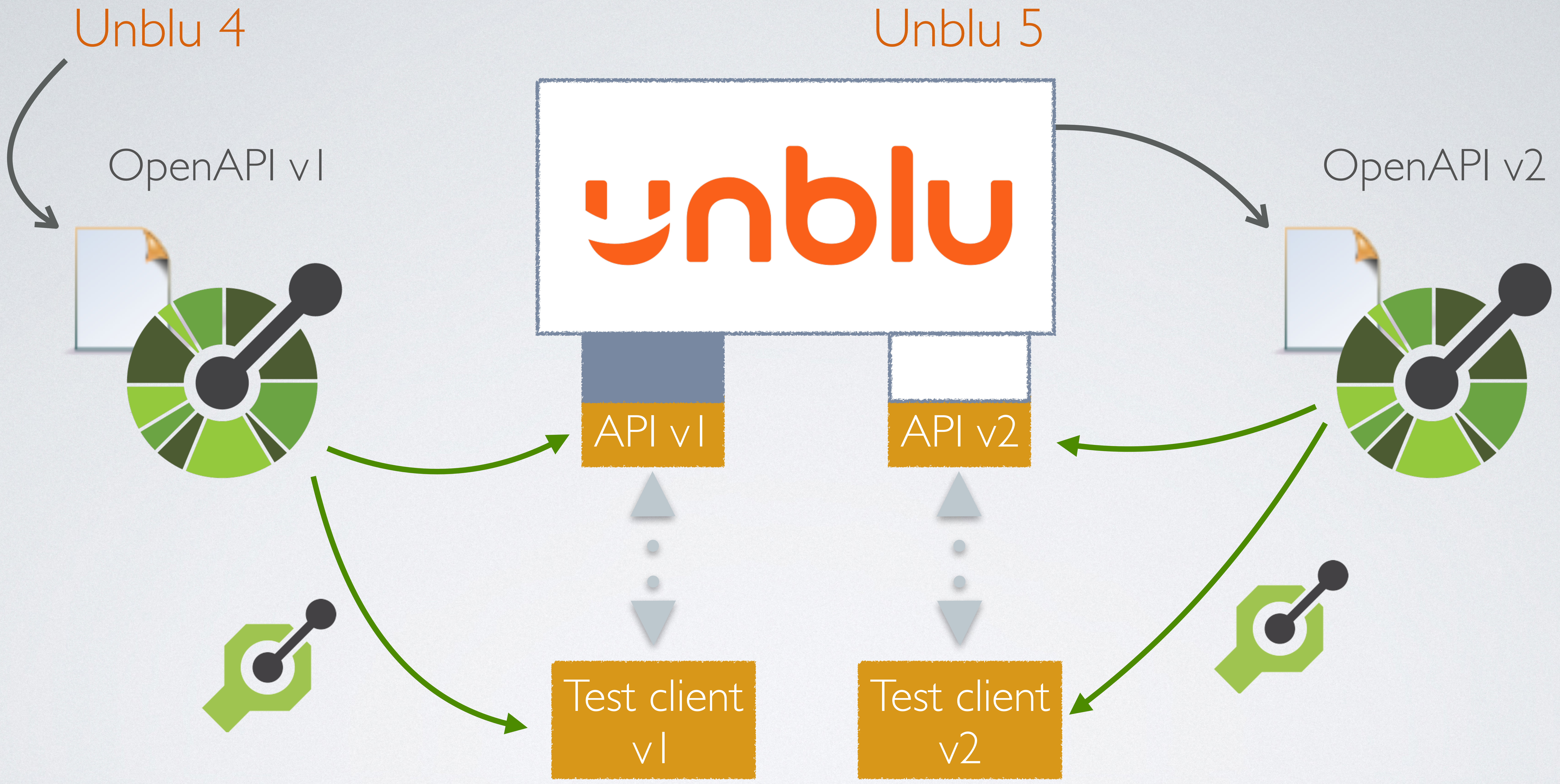

unblu
Custom JSON
Static HTML pages
Swagger-UI
POSTMAN
code
Breaking change detection
https://github.com/quen2404/openapi-diff

API versioning



API versioning




Unblu 4
Unblu 5
OpenAPI v1
OpenAPI v2
unblu
API v1
API v2
Test client v1
Test client v2

Thank you!

@j2r2b

jmini

Code Examples: https://github.com/jmini/openapi-talk