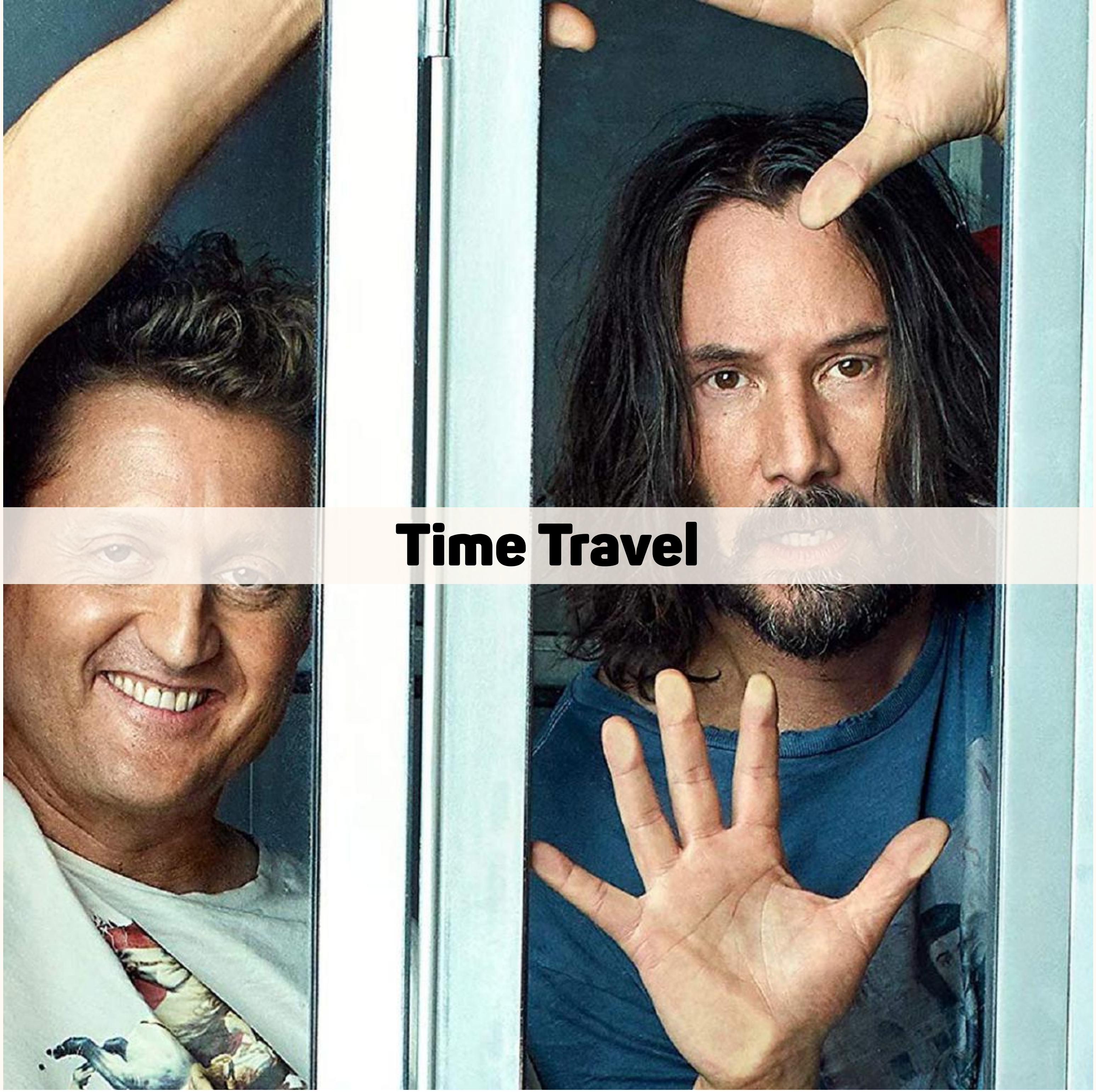
API Experience – Good design for better and successful APIs that engage with your customers

Daniel Kocot, Senior Solution Architect / Head of API Experience & Operations

Name:	Daniel Kocot
Role:	Senior Solution Architect / Head of API Experience & Operations
Email:	daniel.kocot@codecentric.de
Twitter:	@dk_1977
LinkedIn:	https://www.linkedin.com/in/danielkocot/

Customers?

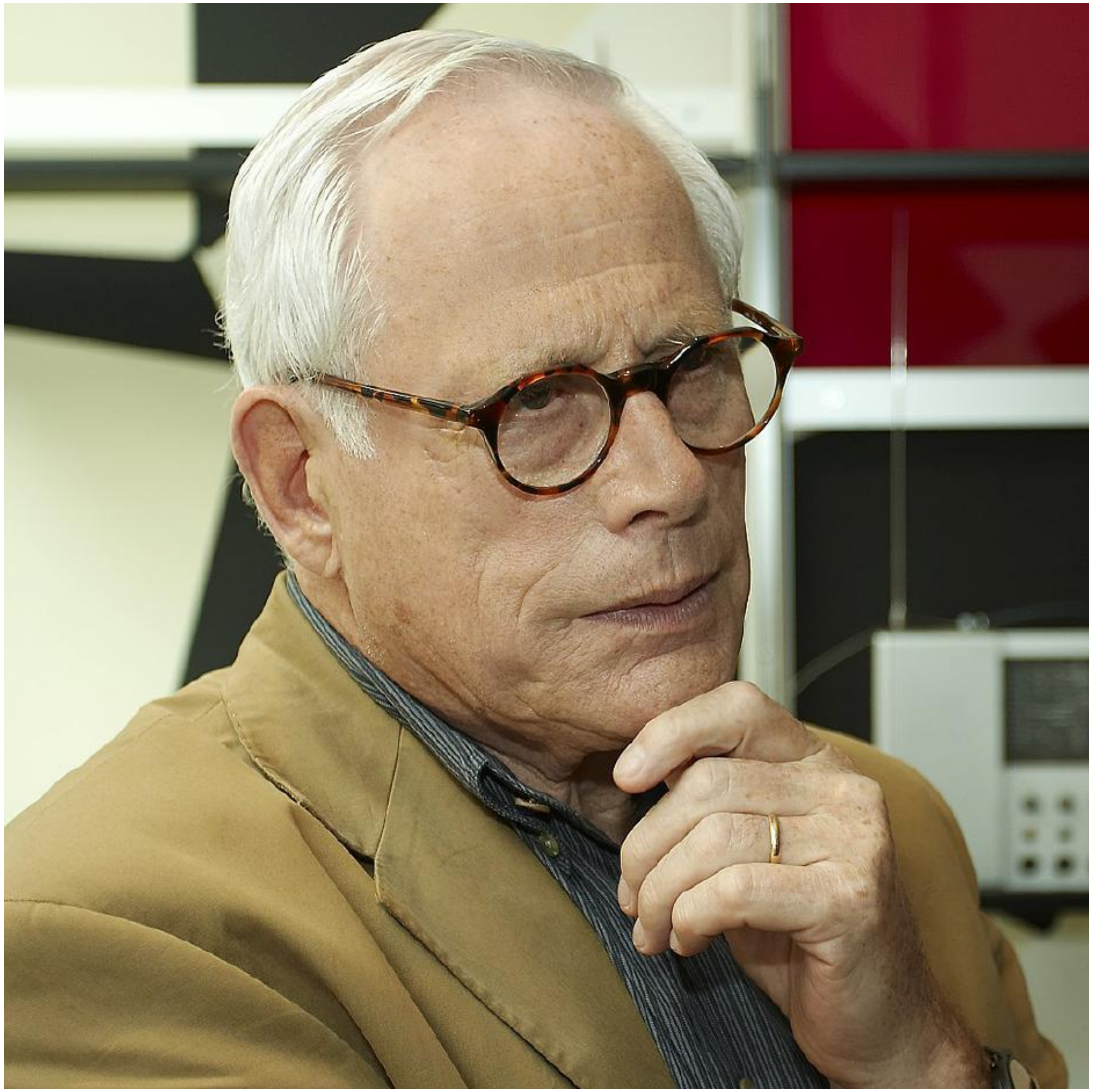
Good Design?



Time Travel

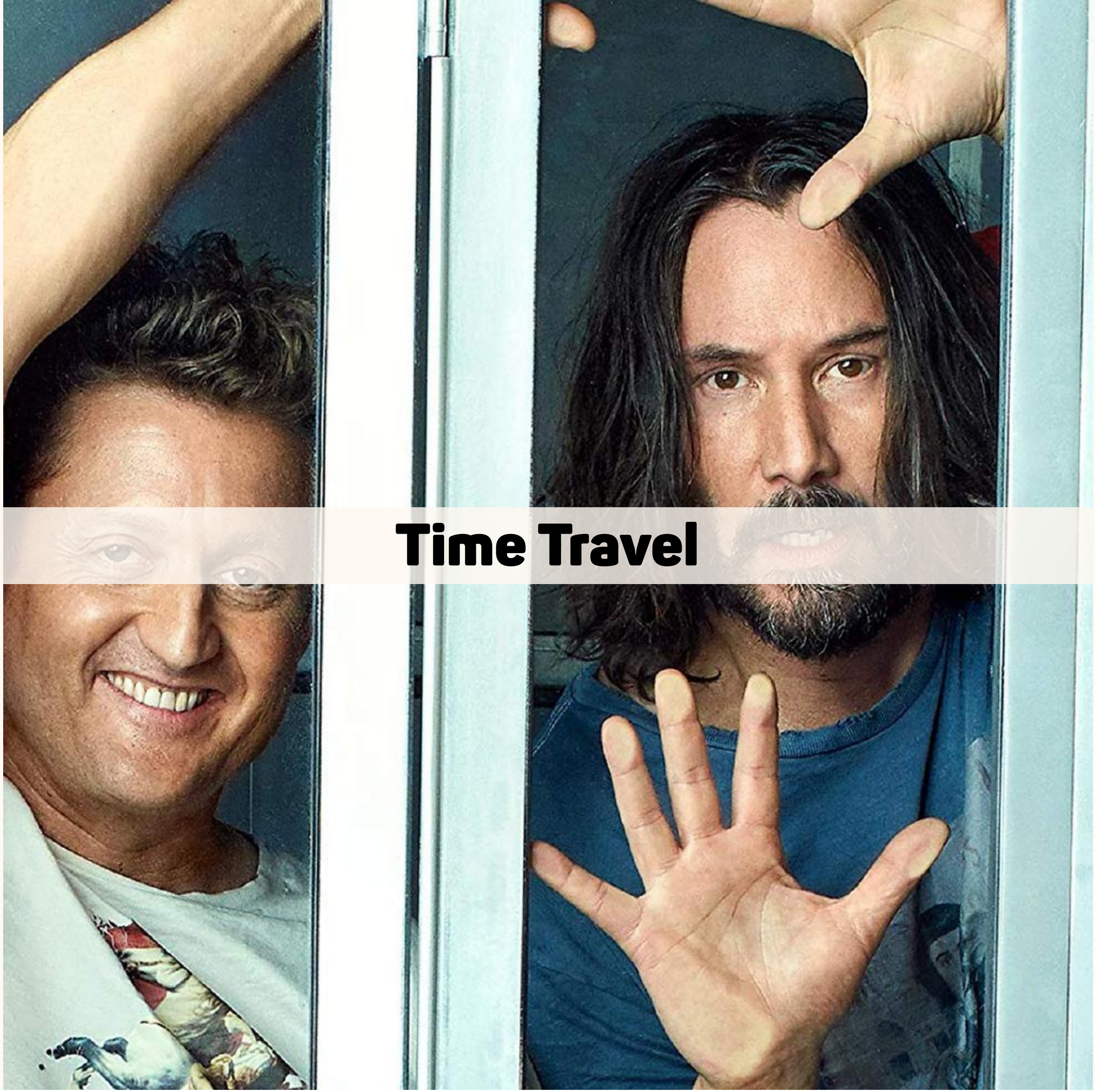
1970s

Dieter Rams



10 Principles of Good Design

Good design is constantly evolving

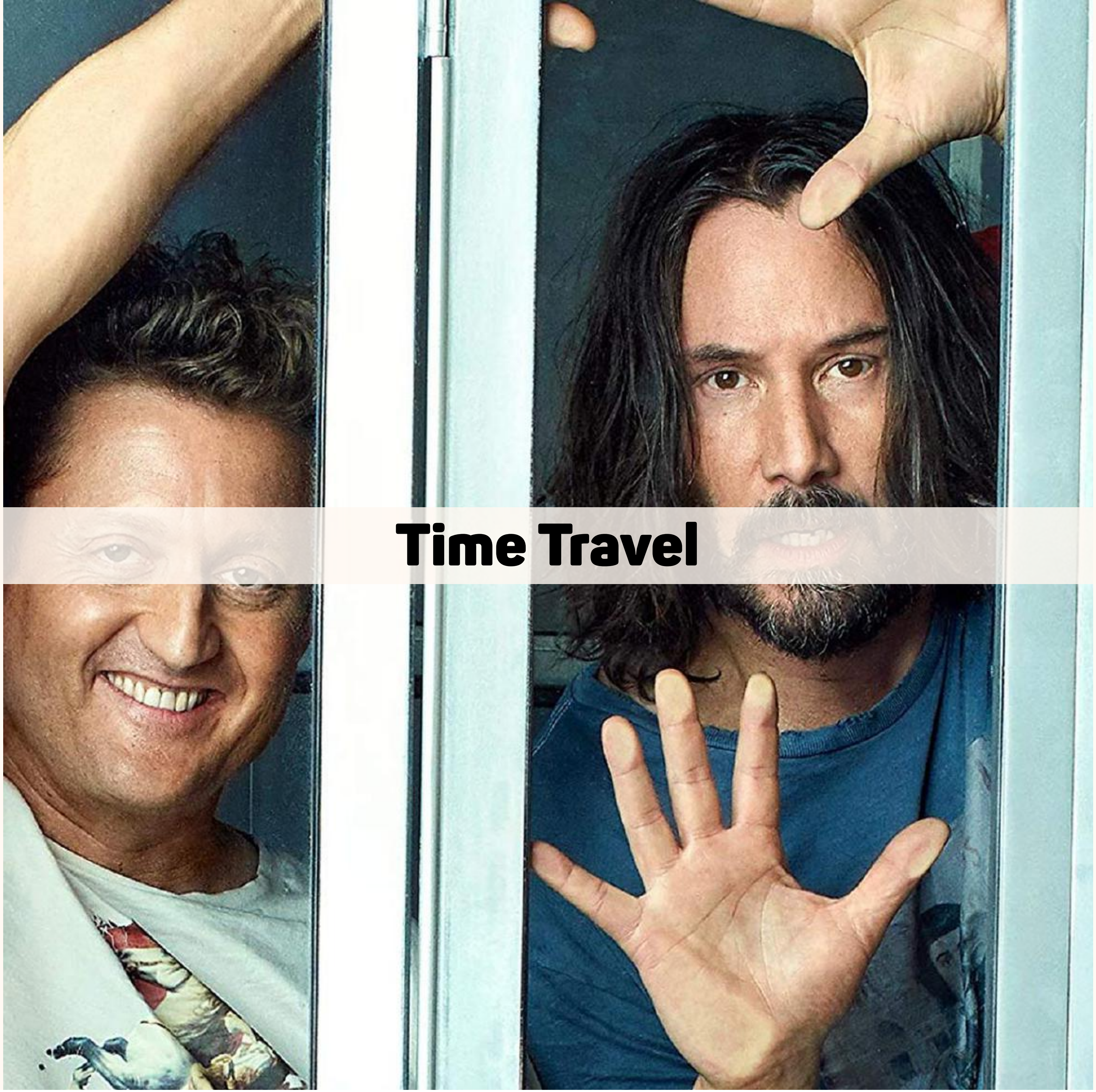


Time Travel

1995

Jakob Nielsen and Rolf Molich

10 Usability Heuristics for User Interface Design



Time Travel

2017

Ronnie Mitra

7 Usability Heuristics for API Design

#1 Visibility of system status

- *Is it difficult to learn when something has gone wrong in the system?*
- *Does the interface tell us the result of invocations and requests?*
- *Should the system describe any relevant side-effects that may have occurred?*

#2 Match between the system and the real world

- *Do the message formats, libraries and message patterns match the user's world?*
- *Is the vocabulary of the API a good match for the user?*
- *Does the API act like the APIs that users are used to using?*

#3 Consistency and standards

- *Is the API consistent in its signature (e.g. URI format, query controls)?*
- *Is the vocabulary of the API consistent? Do words have the same meaning everywhere?*
- *Is the documentation and support tooling consistent across all parts of the API? Does the runtime behavior match the documented behavior?*

#4 Error prevention

- *Are documented examples incorrect or misleading?*
- *Is the API designed in a way that makes it “brittle” — where changes to the interface can easily break the application?*
- *Is the design overly complex? Are there opportunities to simplify the cognitive workload of the user?*

#5 Flexiblity and efficiency of use

- *How suitable is the interface for the first-time user?*
- *Does the API provide controls and shortcuts for more advanced and experienced users? Are defaults used for special controls?*
- *Are there opportunities to optimize any repetitive or unnecessary steps?*

#6 Help user recognize, diagnose and recover from errors

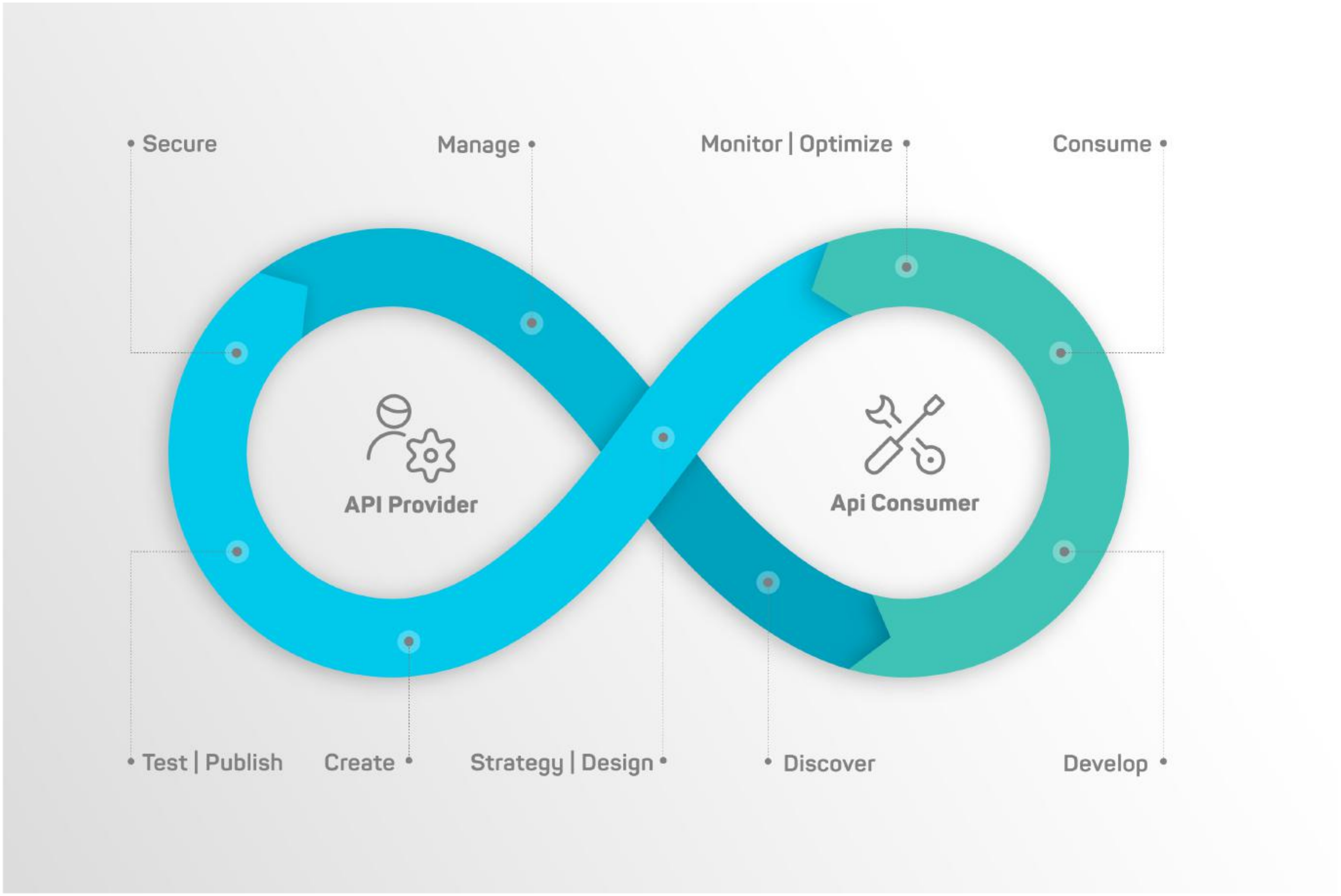
- *Is error information correct?*
- *Is machine readable information provided?*
- *Does it describe the error in a way that the human use can understand it?*
- *Is enough information provided to correct the error?*

#7 Help and documentation

- *Does the documentation address the needs of different learning stages (beginner, intermediate and expert?)*
- *How much documentation needs to be read before a call can be made? Are examples provided in the docs?*
- *How well does the documentation structure map to the problems that a user will try to solve?*

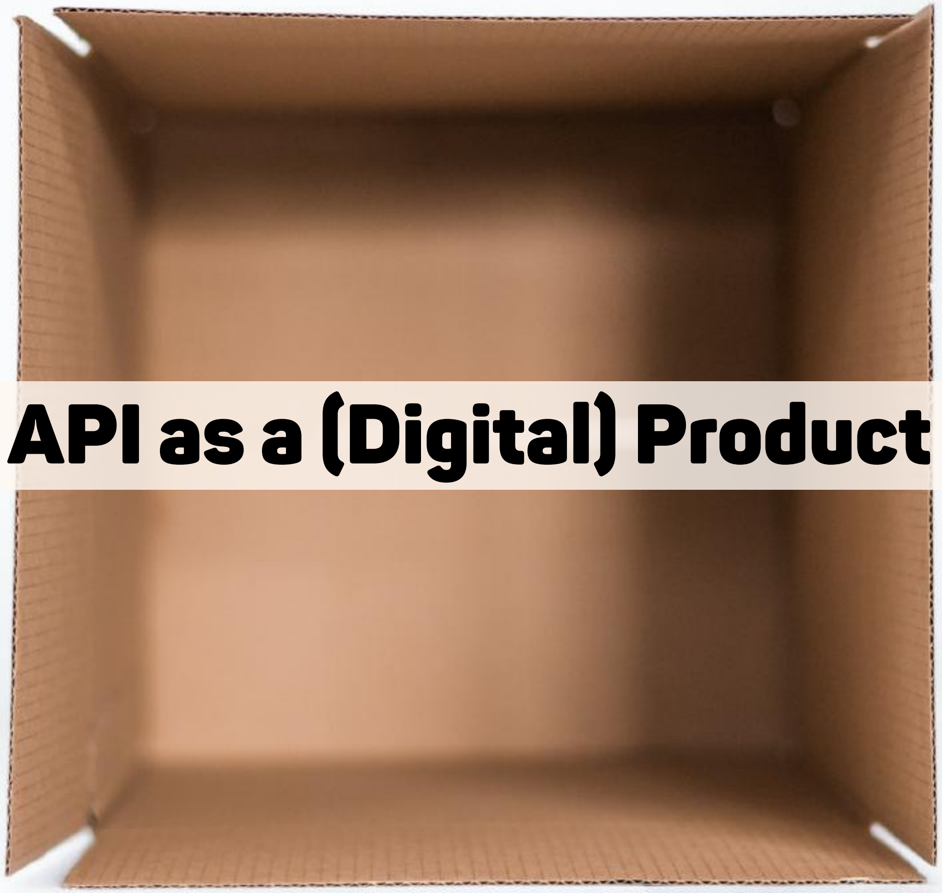
Process

API Management

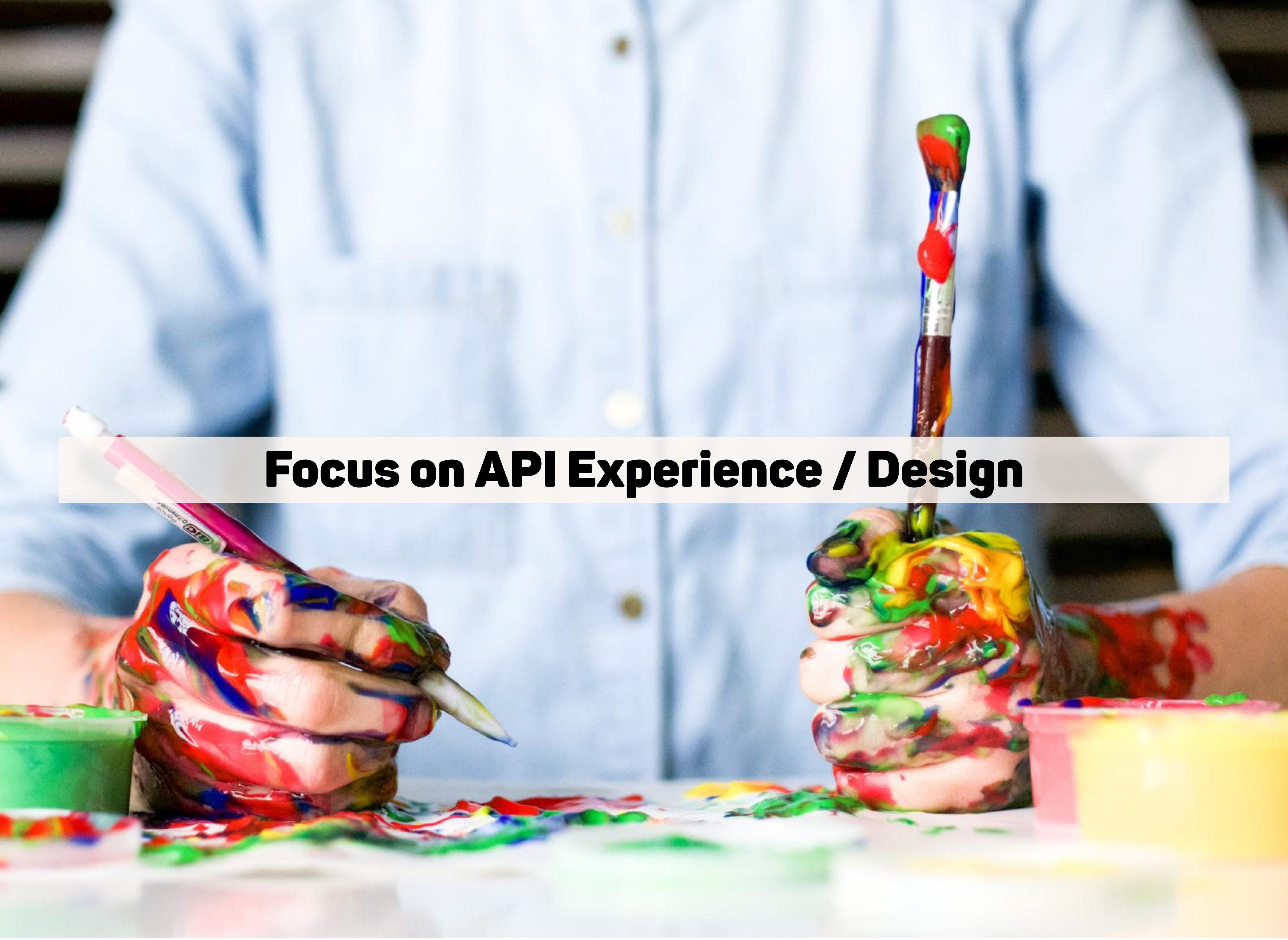


API first

- An API is the first (and often only) interface to users of an application
- An API comes first — before the implementation
- An API is described (documented) or self-descriptive



API as a (Digital) Product

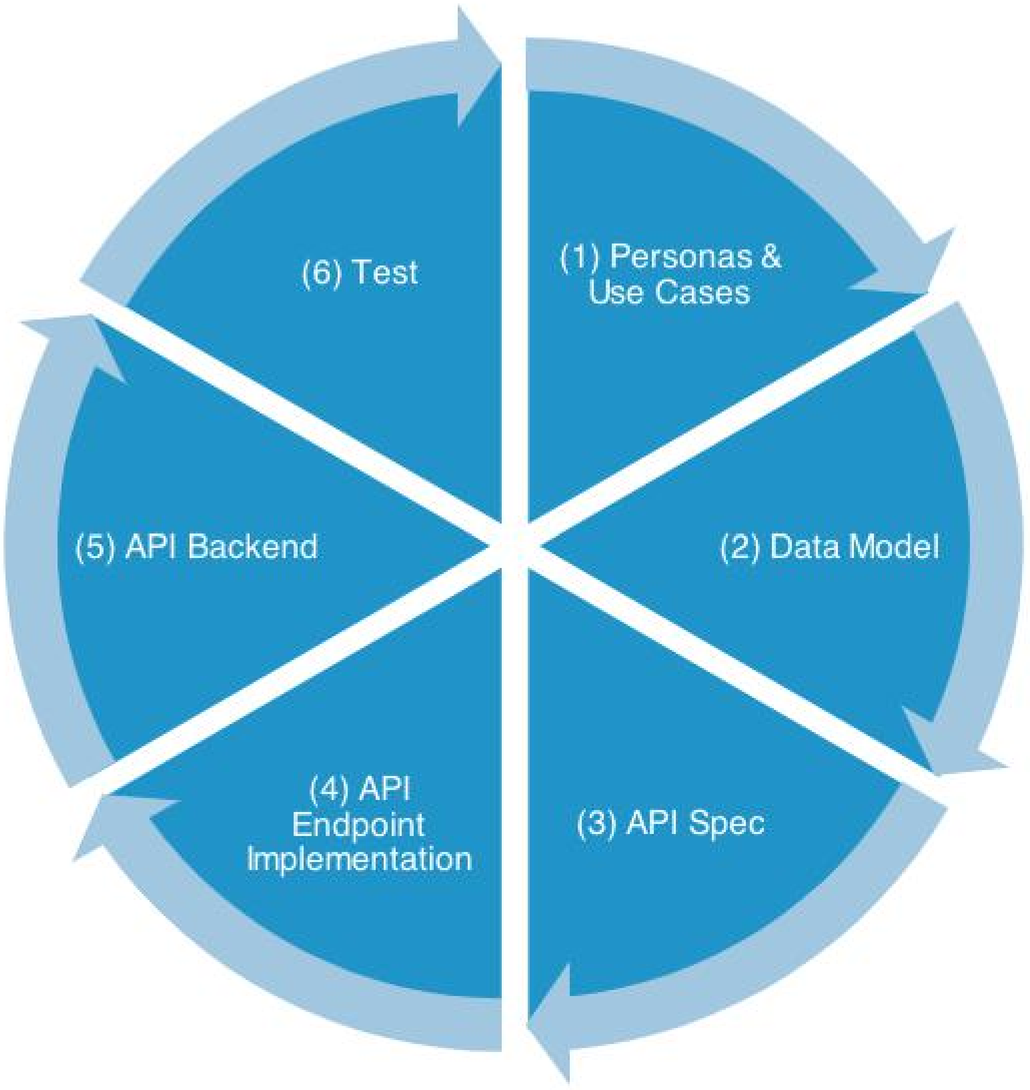


Focus on API Experience / Design

User Experience

'READ THIS BOOK AND DO WHAT IT SAYS IF YOU WANT TO BUILD BETTER PRODUCTS FASTER.'
EV WILLIAMS, FOUNDER OF MEDIUM AND TWITTER
SPRINT
how to
SOLVE BIG PROBLEMS
AND TEST NEW IDEAS
IN JUST FIVE DAYS
JAKE KNAPP
WITH JOHN ZERATSKY & BRADEN KOWITZ
from GOOGLE VENTURES
WONDER TAPE
Original KD

Design Sprint

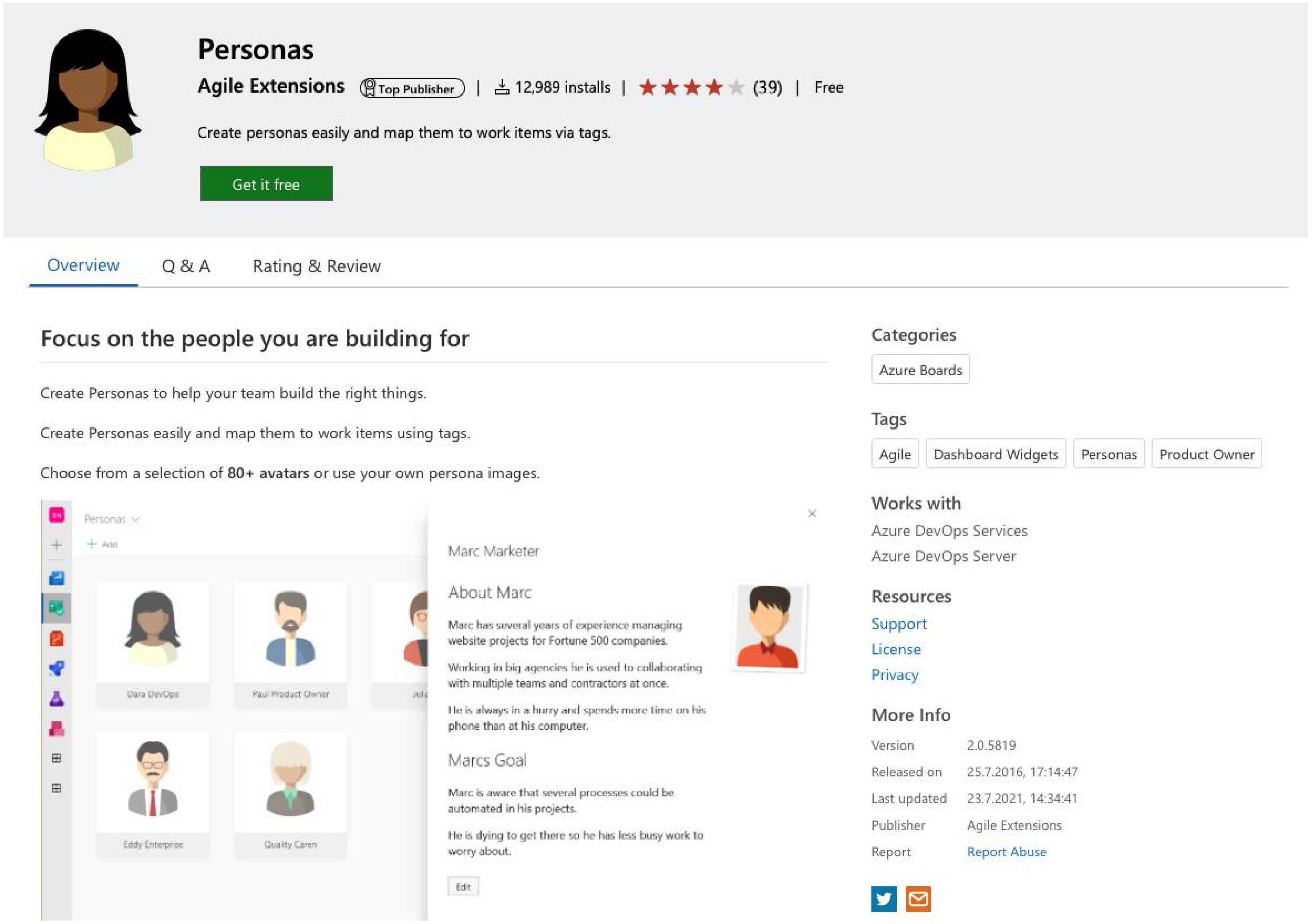

(1) Personas & Use Cases
(2) Data Model
(3) API Spec
(4) API Endpoint Implementation
(5) API Backend
(6) Test

Personas & Use Cases

Personas

Who is going to use the API?

Introduction of characters



Personas

Agile Extensions Top Publisher | 12,989 installs | ★★★★☆ (39) | Free

Create personas easily and map them to work items via tags.

Get it free

Overview Q & A Rating & Review

Focus on the people you are building for

Create Personas to help your team build the right things.

Create Personas easily and map them to work items using tags.

Choose from a selection of **80+ avatars** or use your own persona images.

Marc Marketer

About Marc

Marc has several years of experience managing website projects for Fortune 500 companies.

Working in big agencies he is used to collaborating with multiple teams and contractors at once.

He is always in a hurry and spends more time on his phone than at his computer.

Marcs Goal

Marc is aware that several processes could be automated in his projects.

He is dying to get there so he has less busy work to worry about.

Categories

Azure Boards

Tags

Agile | Dashboard Widgets | Personas | Product Owner

Works with

Azure DevOps Services
Azure DevOps Server

Resources

Support
License
Privacy

More Info

Version	2.0.5819
Released on	25.7.2016, 17:14:47
Last updated	23.7.2021, 14:34:41
Publisher	Agile Extensions
Report	Report Abuse

Security discussion in reference to a Persona

Building an Authentication / Authorization Flow

- Technical user
- Functional user

Use Cases

Describing why the API is needed and what systems are involved.

BUT...

You are not the consumer!

API by use case first

API Styles

- Tunnel
- Resource
- Query
- Event-Based

Richardson Maturity Model for Web APIs

- Level 0: API uses RPC style
- Level 1: API exposes Resources
- Level 2: API uses HTTP methods and uses HTTP efficiently
- Level 3: API uses HATEOAS. The API is self-documenting and flexible

Dissertation by Roy Thomas Fielding (2000)

Architectural Styles and the Design of Network-based Software Architectures

REST emphasizes scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems. I describe the software engineering principles guiding REST and the interaction constraints chosen to retain those principles, contrasting them to the constraints of other architectural styles. Finally, I describe the lessons learned from applying REST to the design of the Hypertext Transfer Protocol and Uniform Resource Identifier standards, and from their subsequent deployment in Web client and server software.

Hypermedia as the Engine of Application State (HATEAOS)

Data Model

imation
2HD IBM
formatted formatee
1.44MB
HD
imation
12/11/03

Data first

Internal vs. External data model

- Does the existent data model support the required use cases?
- Is a middleware for the aggregations and/or transformations needed?

Master Data Management

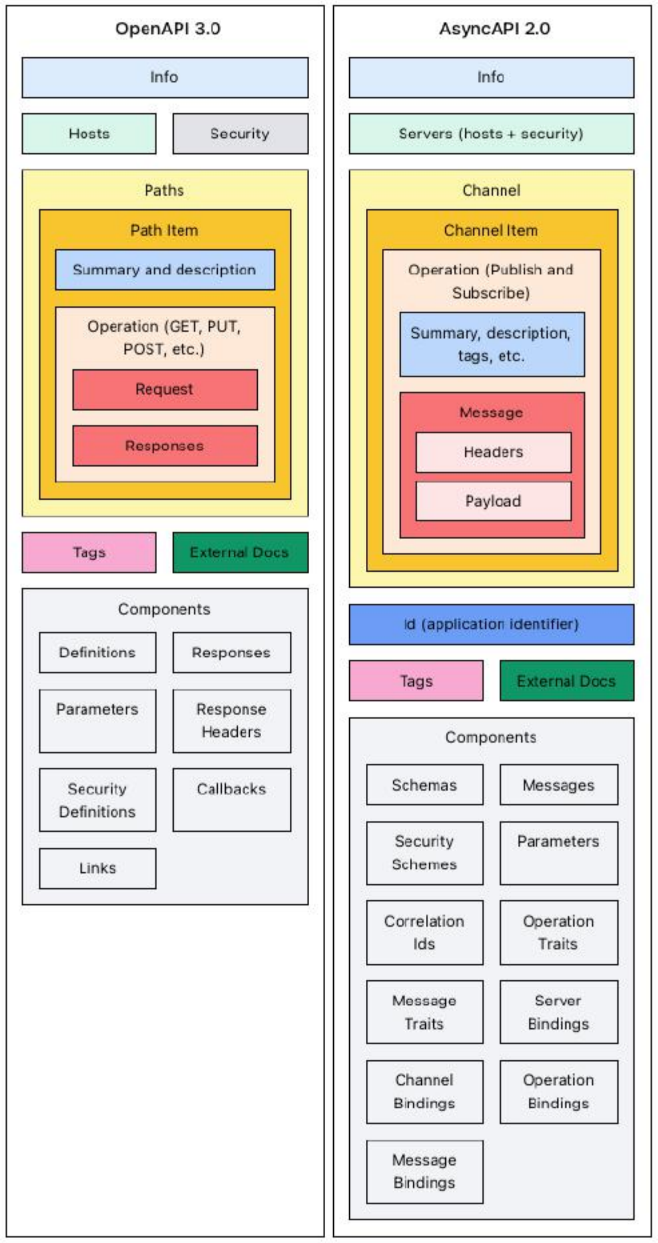
Adopting the idea of a Data Catalog

Data Representation

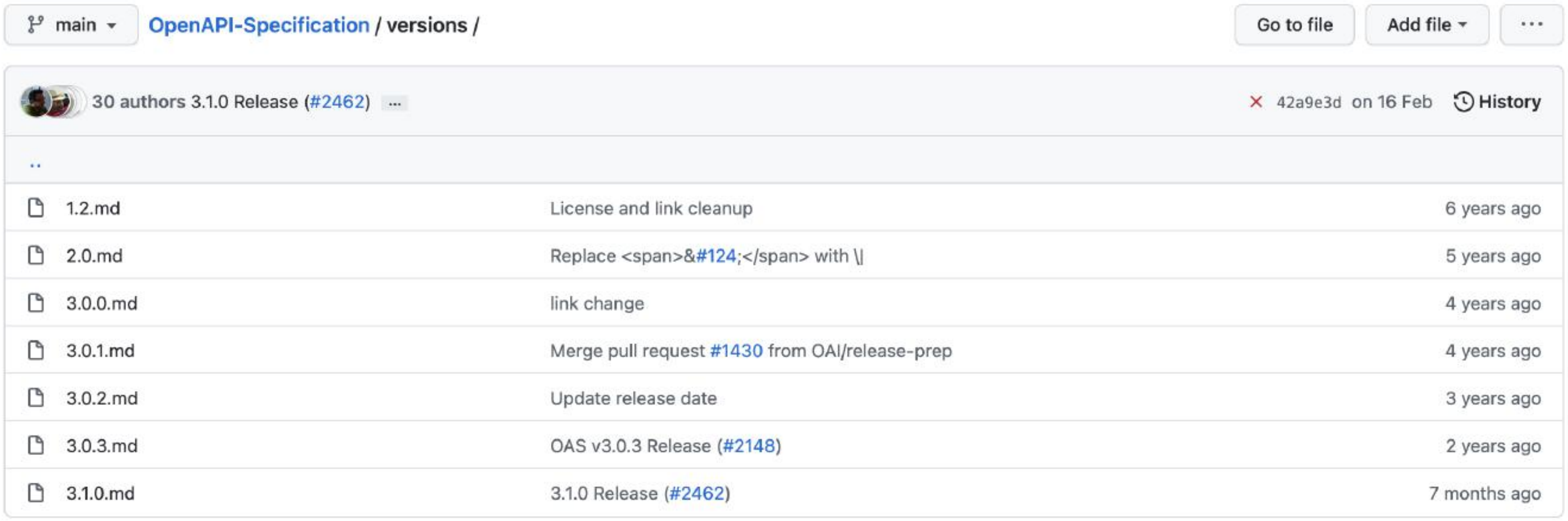
- XML
- JSON Schema
- JSON for Linking Data (JSON-LD)
- RDF (Resource Description Framework)
- CBOR (Concise Binary Object Representation)

API Specification

OpenAPI / AsyncAPI



Specification Version



main OpenAPI-Specification / versions /

Go to file Add file ···

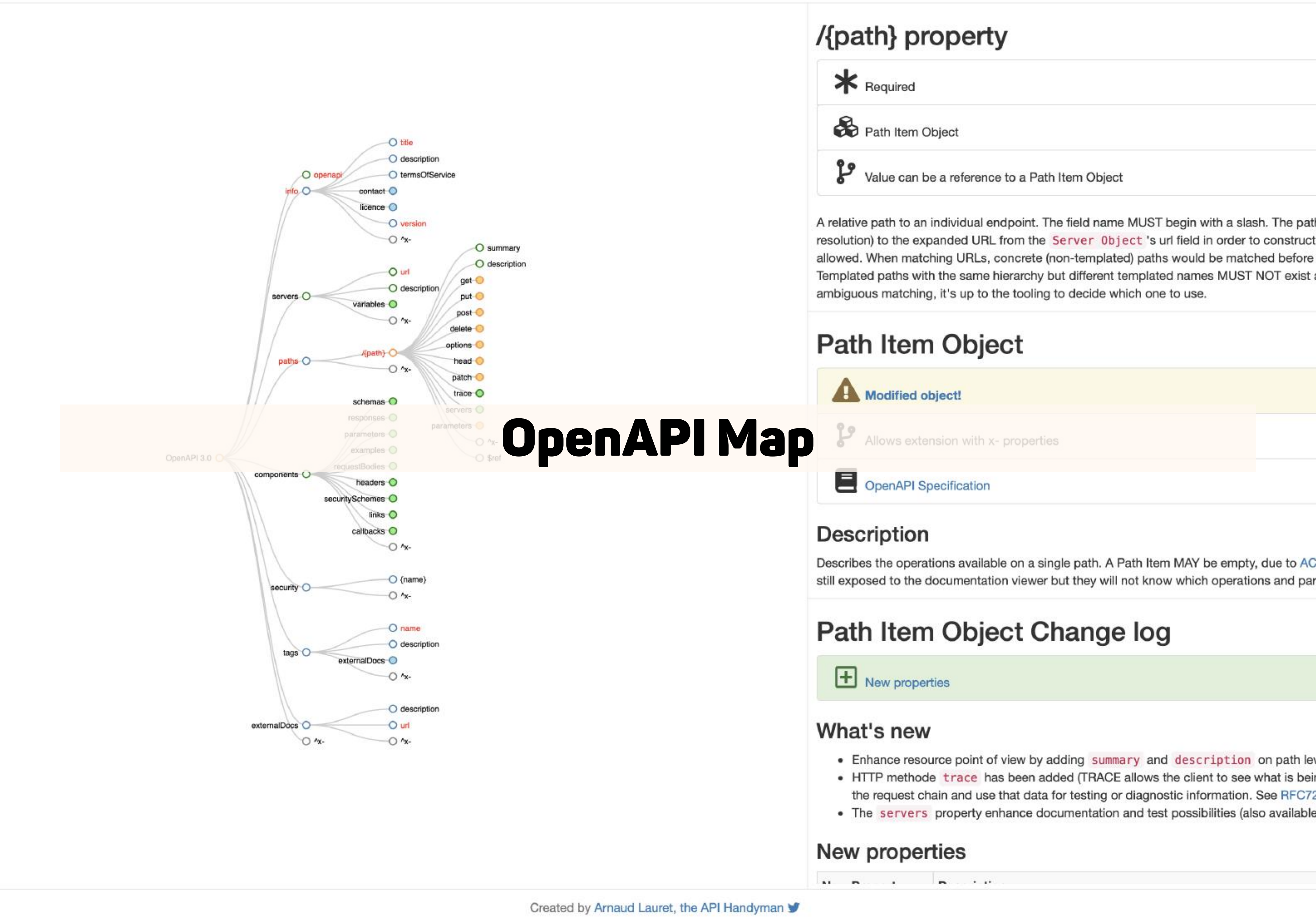
30 authors 3.1.0 Release (#2462) ··· 42a9e3d on 16 Feb History

..

1.2.md	License and link cleanup	6 years ago
2.0.md	Replace <span>|</span> with |	5 years ago
3.0.0.md	link change	4 years ago
3.0.1.md	Merge pull request #1430 from OAI/release-prep	4 years ago
3.0.2.md	Update release date	3 years ago
3.0.3.md	OAS v3.0.3 Release (#2148)	2 years ago
3.1.0.md	3.1.0 Release (#2462)	7 months ago

Focus Rest(ful) APIs

OpenAPI Map



/{path} property

Required

Path Item Object

Value can be a reference to a Path Item Object

A relative path to an individual endpoint. The field name MUST begin with a slash. The path
resolution) to the expanded URL from the `Server Object` 's url field in order to construct
allowed. When matching URLs, concrete (non-templated) paths would be matched before
Templated paths with the same hierarchy but different templated names MUST NOT exist a
ambiguous matching, it's up to the tooling to decide which one to use.

Path Item Object

Modified object!

Allows extension with x- properties

OpenAPI Specification

Description

Describes the operations available on a single path. A Path Item MAY be empty, due to AC
still exposed to the documentation viewer but they will not know which operations and par

Path Item Object Change log

New properties

What's new

- Enhance resource point of view by adding `summary` and `description` on path lev
- HTTP methode `trace` has been added (TRACE allows the client to see what is bein the request chain and use that data for testing or diagnostic information. See RFC72
- The `servers` property enhance documentation and test possibilities (also available

New properties

Created by Arnaud Lauret, the API Handyman

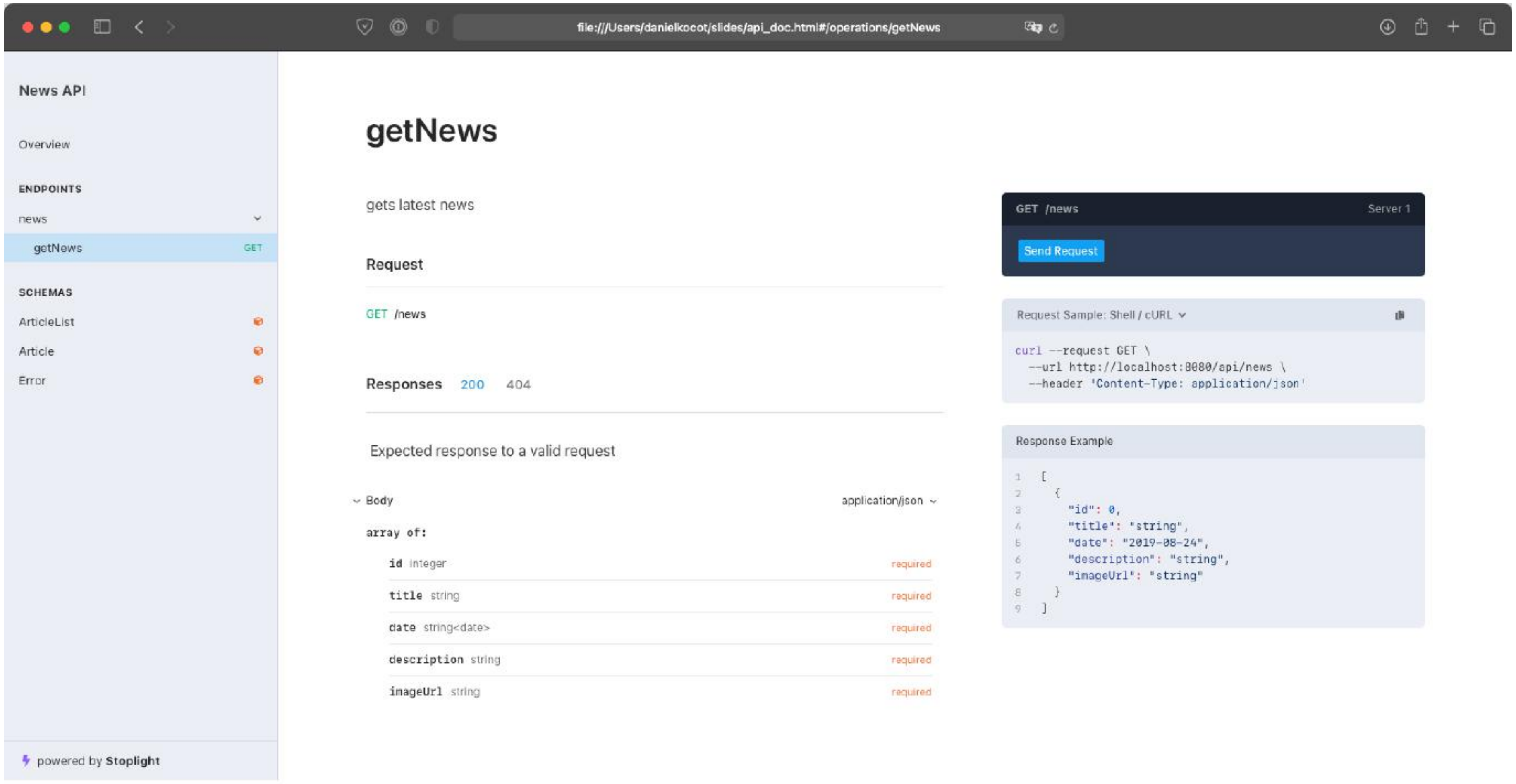
Principles Rest(ful) HTTP

Object oriented interface	Rest(ful) HTTP
getEmployees()	GET /employees
updateEmployee(id)	PUT /employees/{id}
addEmployee()	POST /employees
deleteEmployee(id)	DELETE /employees/{id}
getEmployeeRoles(id)	GET /employees/{id}/roles

HTTP Methods / Verbs

HTTP Methods	Safe	Idempotent
GET	X	X
HEAD	X	X
PUT	-	X
POST	-	-
DELETE	-	X
OPTIONS	X	X
PATCH	-	-

OpenAPI Spec Example



getNews

gets latest news

Request

GET /news

Responses 200 404

Expected response to a valid request

Body application/json

```
array of:
  id integer                        required
  title string                      required
  date string<date>                 required
  description string                required
  imageUrl string                   required
```

```
curl --request GET \
  --url http://localhost:8080/api/news \
  --header 'Content-Type: application/json'
```

Response Example

```
[
  {
    "id": 0,
    "title": "string",
    "date": "2019-08-24",
    "description": "string",
    "imageUrl": "string"
  }
]
```

OpenAPI Description Example

```yaml
openapi: 3.0.3
servers:
  - url: 'http://localhost:8080'
info:
  version: 1.0.0
  title: News API
  contact:
    name: Daniel Kocot
    url: 'http://www.codecentric.de'
    email: daniel.kocot@codecentric.de
  license:
    name: MIT
    url: 'https://www.tldrelgal.com/mit'
  description: An API to provide news
tags:
  - name: news
paths:
  /news:
    get:
      description: gets latest news
      operationId: getNews
      tags:
        - news
      responses:
```

Errorhandling

- 1xx serves information purposes
- 2xx is used for successful request
- 3xx shows redirects
- 4xx is used for client-side errors
- 5xx is used for errors

Problem Details for HTTP APIs (RFC7807)

```
HTTP/1.1 403 Forbidden
  Content-Type: application/problem+json
  Content-Language: en

  {
   "type": "https://example.com/probs/out-of-credit",
   "title": "You do not have enough credit.",
   "detail": "Your current balance is 30, but that costs 50.",
   "instance": "/account/12345/msgs/abc",
   "balance": 30,
   "accounts": ["/account/12345",
                "/account/67890"]
  }
```

Examples in OpenAPI Descriptions

Media Type Object

```
responses:
  '200':
    description: response
    content:
      application/vnd.github.v3.object:
        schema:
          "$ref": "#/components/schemas/content-tree"
      application/json:
        schema:
          oneOf:
          - "$ref": "#/components/schemas/content-directory"
          - "$ref": "#/components/schemas/content-file"
          - "$ref": "#/components/schemas/content-symlink"
          - "$ref": "#/components/schemas/content-submodule"
        examples:
          response-if-content-is-a-file:
            "$ref": "#/components/examples/content-file-response-if-content-is-a-file"
          response-if-content-is-a-directory:
            "$ref": "#/components/examples/content-file-response-if-content-is-a-directory"
          response-if-content-is-a-symlink:
            "$ref": "#/components/examples/content-file-response-if-content-is-a-symlink"
          response-if-content-is-a-submodule:
            "$ref": "#/components/examples/content-file-response-if-content-is-a-submodule"
```

Schema Object Examples

```yaml
components:
  schemas:
    ArticleList:
      title: ArticleList
      type: array
      items:
        $ref: '#/components/schemas/Article'
    Article:
      title: Article
      description: A article is a part of a news.
      type: object
      properties:
        id:
          type: integer
        title:
          type: string
          example: First Article
        date:
          type: string
          pattern: '^\d{4}(0[1-9]|1[012])(0[1-9]|[12][0-9]|3[01])$'
          example: "20210525"
        description:
          type: string
          example: A description
```

Examples for API Design Patterns

- Long Running Operations
- Paging / Filtering
- Large Payloads

Goal: Establishing a design library

Architecture Discussion

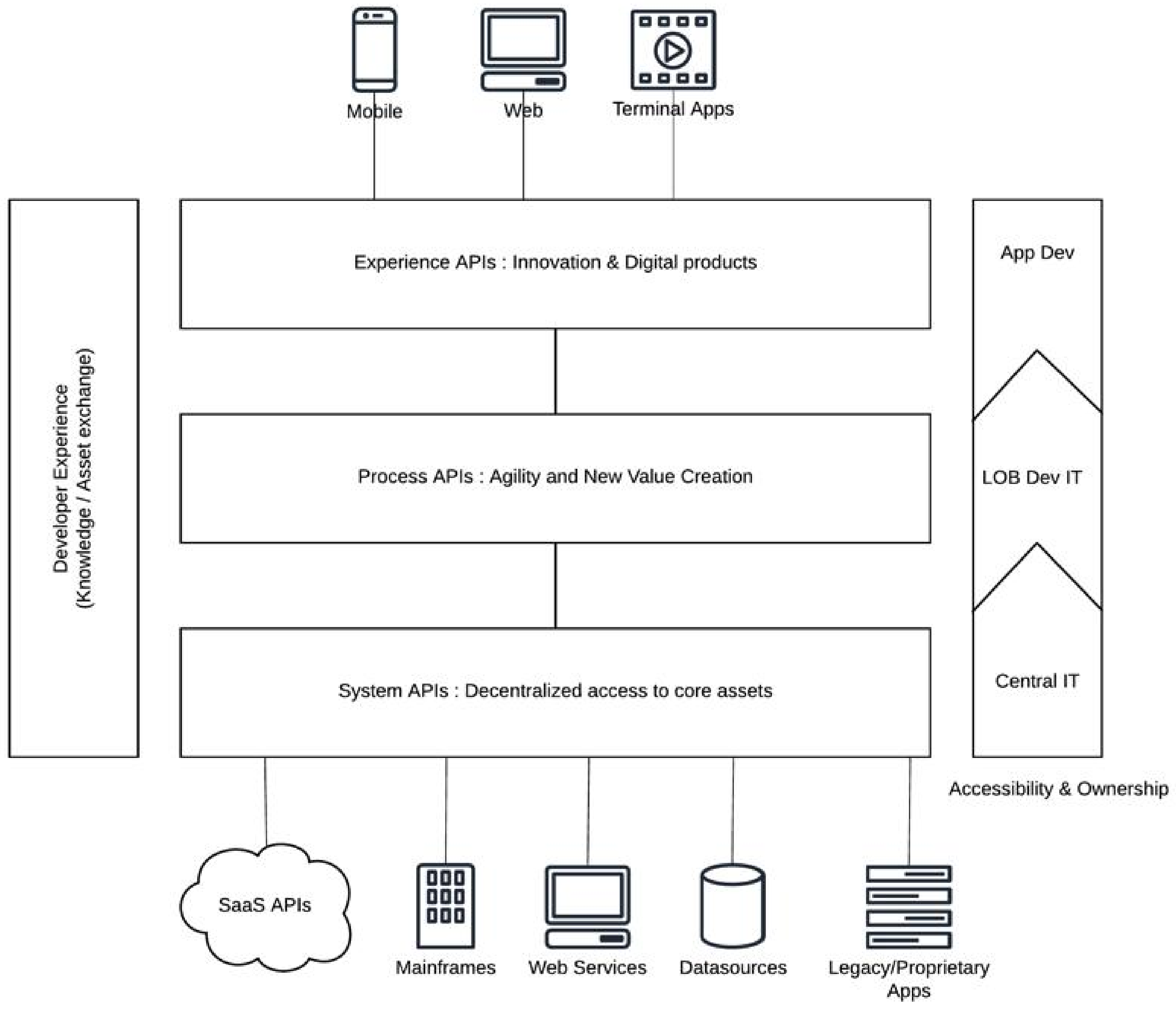
Solutions Architecture Pattern

Using patterns which are well established in the industry...

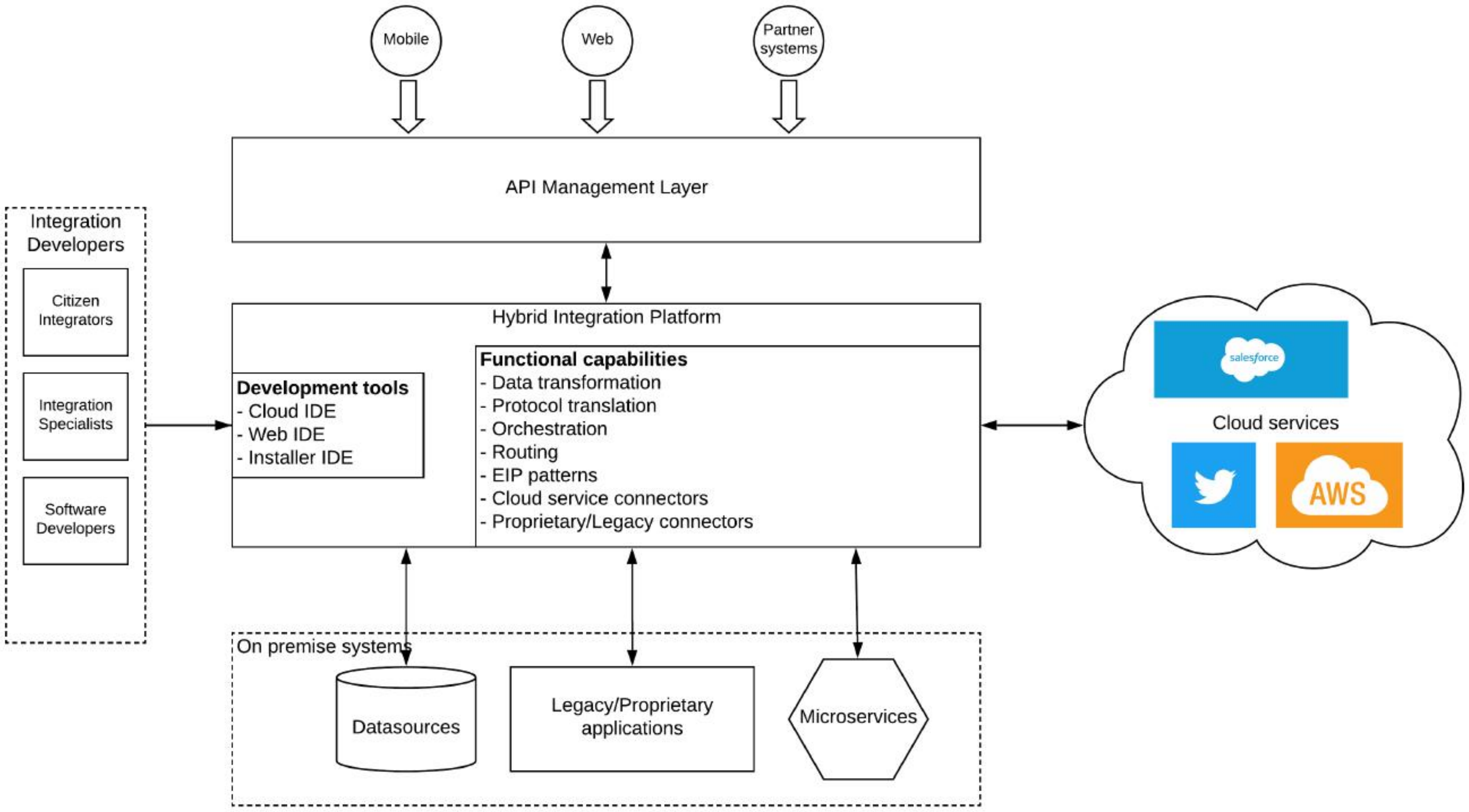
But also still evolving

But please only adopt them

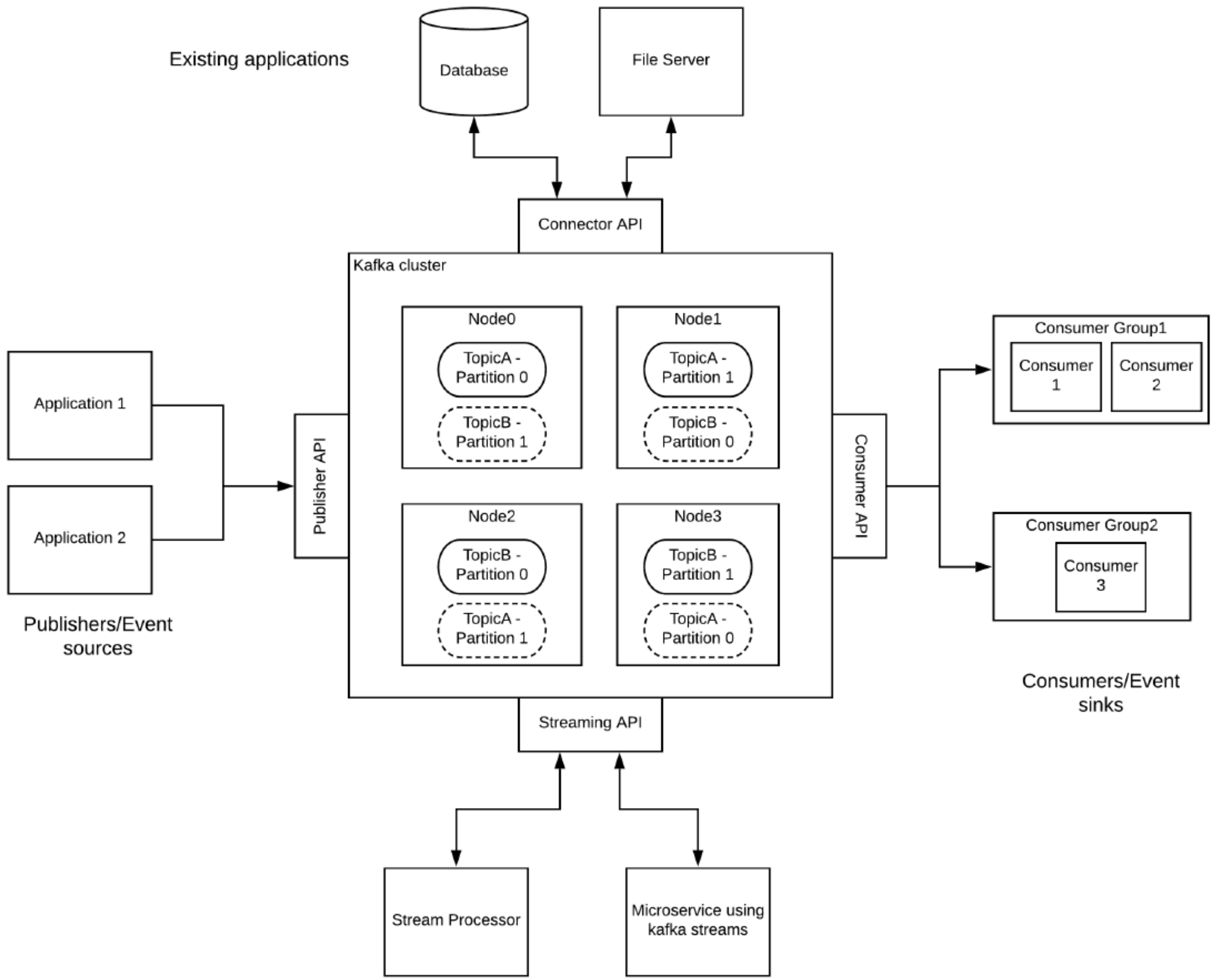
API-led connectivity


Mobile
Web
Terminal Apps
Developer Experience (Knowledge / Asset exchange)
Experience APIs : Innovation & Digital products
Process APIs : Agility and New Value Creation
System APIs : Decentralized access to core assets
App Dev
LOB Dev IT
Central IT
Accessibility & Ownership
SaaS APIs
Mainframes
Web Services
Datasources
Legacy/Proprietary Apps

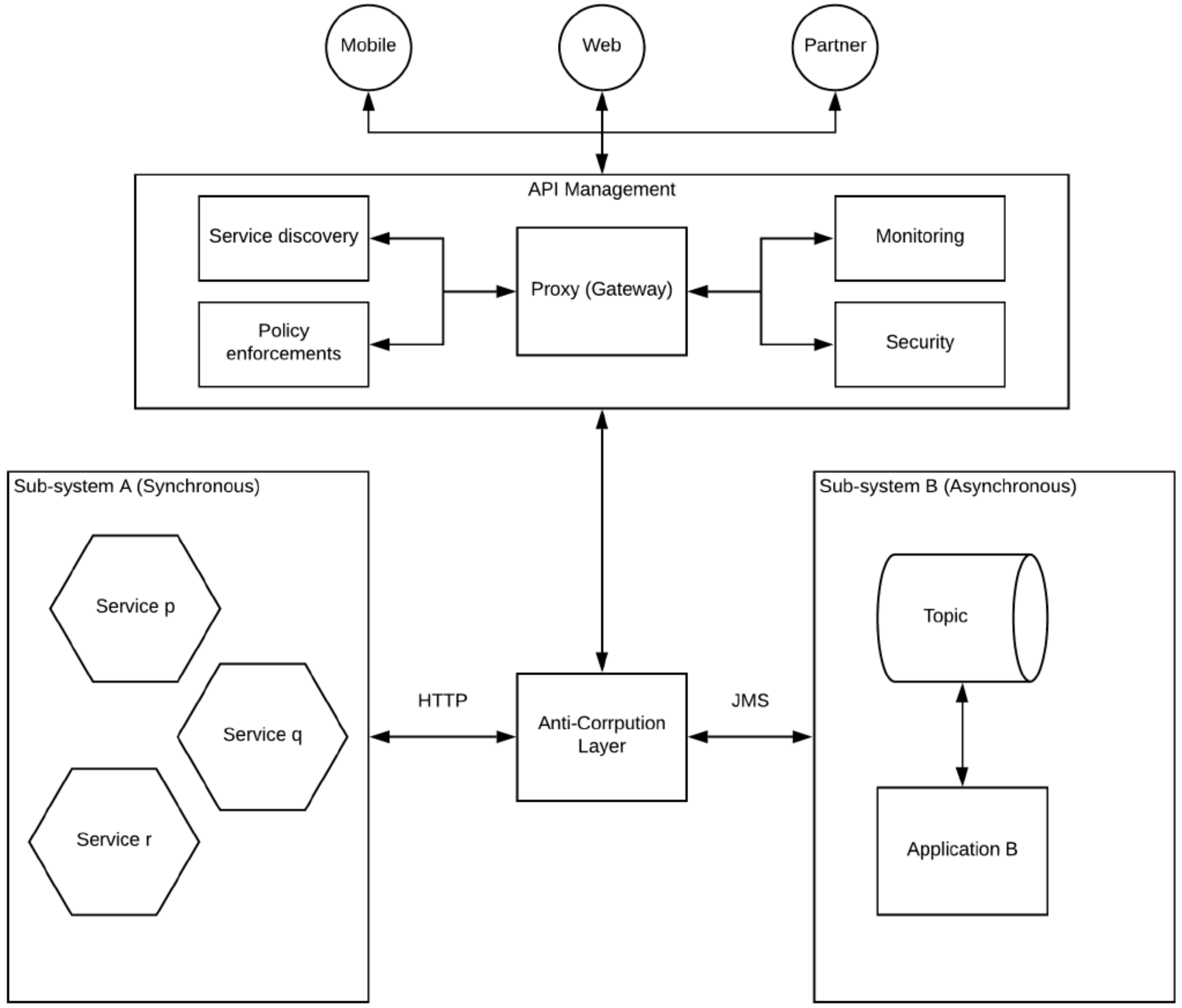
Hybrid integration



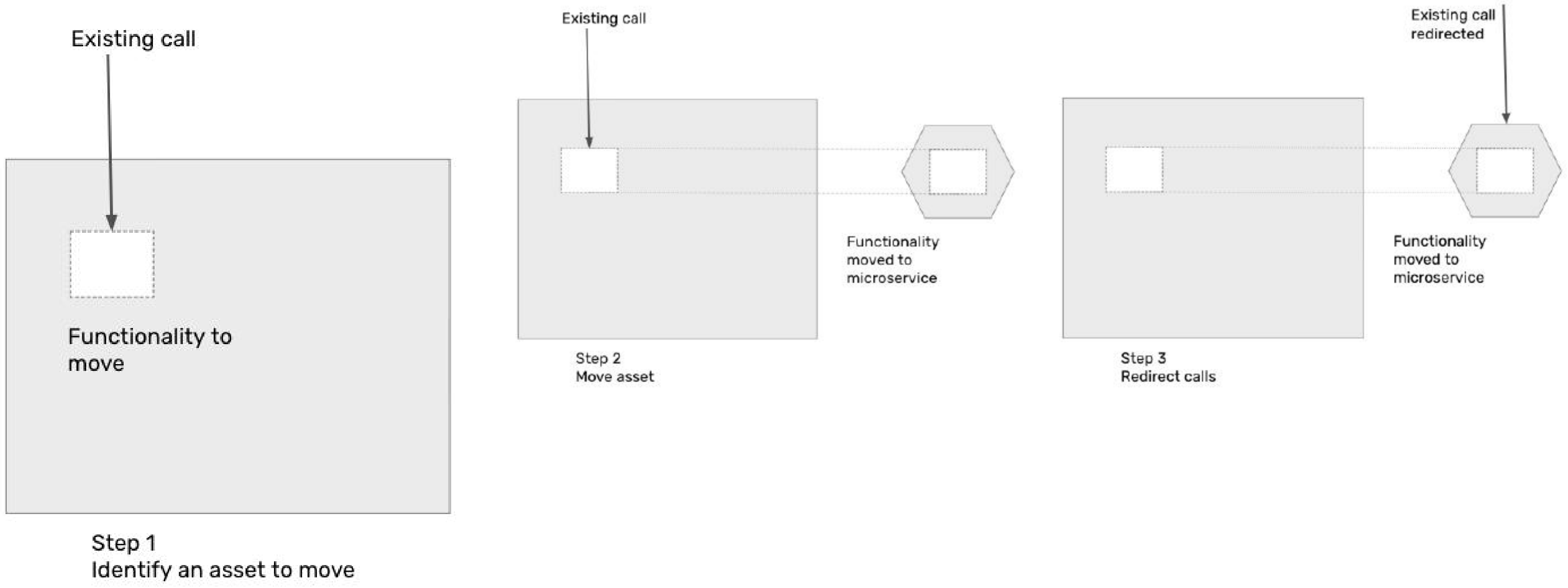
Event-Driven architecture

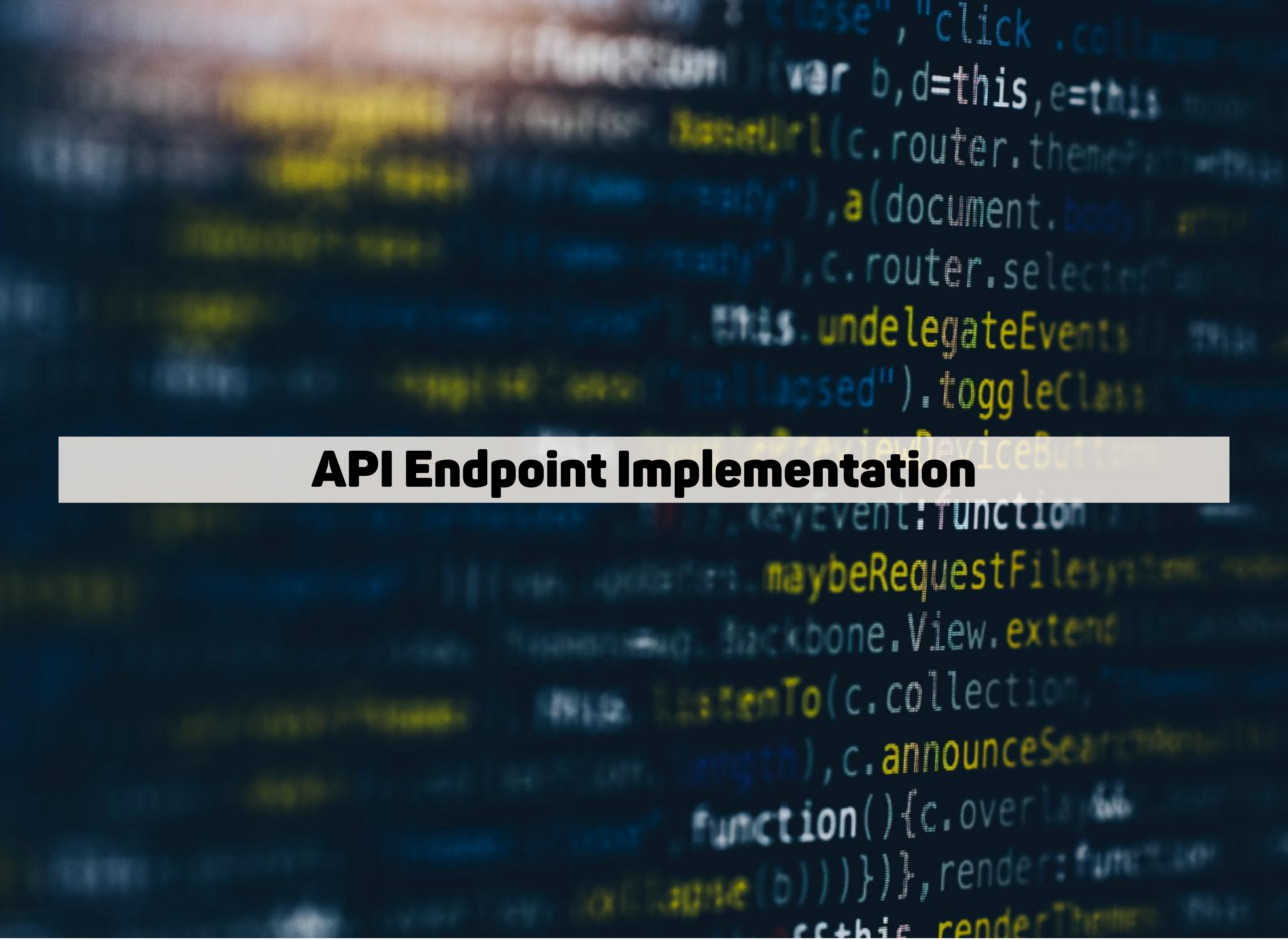


Anti-Corruption Layer



Strangler Pattern





API Endpoint Implementation

API Mediation

- Architectural layer to manage, protect and enrich an API
- Intercepting API traffic
- Concept of "outer" APIs
- No business logic should be handled within this layers

Generating model classes for clients

API Backend

Services for Backend Systems

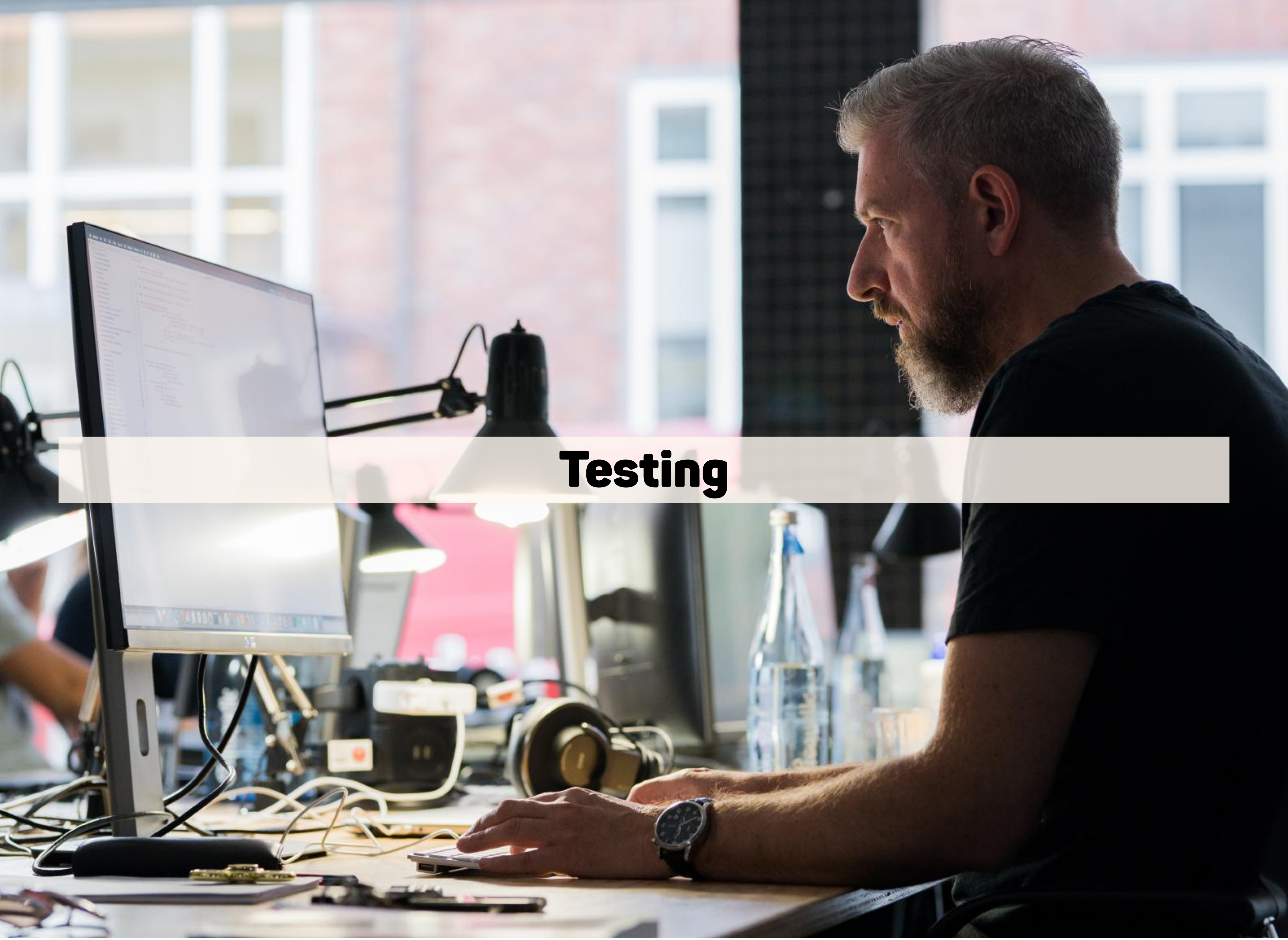
- Use a framework the development team is proficient with
- To create a first representation of the data
- Transformation is maybe needed

Transformations

- Use Enterprise Integration Patterns
- Apache Camel, Spring Integration, Apache Nifi, SaaS Service (e.g. Make), ...

Services for Aggregations

- Use again a framework the development team is proficient with
- To create aggregated oder composed representation of data from Backend APIs
- These APIs help to create a better experience for the user



Testing

Based on the description

Description becomes a contract

Provide a Postman Collection of the API product

Portman

Load testing

- Smoke
- Load
- Stress
- Soak

Wrap Up

Posts on codecentric blog:

https://blog.codecentric.de/en/author/daniel-kocot/

Posts on my blog:

https://danielkocot.github.io

Posts on Medium:

https://medium.com/@daniel.kocot

Q&A

Thank you

🙇🎉

References

- Photo by Blake Wisz on Unsplash
- By docsearls - Flickr, CC BY-SA 2.0, https://commons.wikimedia.org/w/index.php?curid=1328081
- Photo by Jean-Philippe Delberghe on Unsplash
- Photo by Kelli McClintock on Unsplash
- Photo by Faizur Rehman on Unsplash
- Photo by Erik Mclean on Unsplash
- Photo by Markus Spiske on Unsplash
- Photo by John Salvino on Unsplash
- Photo by Gautam Lakum on Unsplash
- Photo by Fredy Jacob on Unsplash
- Photo by Emil Widlund on Unsplash
- Photo by Dan Dennis on Unsplash

codecentric