Swagger Workshop

Jan Illetschko

https://github.com/Swagger-Training/swagger-training

Targets

- Read/Write JSON and YAML
- Understand JSON Schema
- Efficient REST API Design with Swagger/OpenAPI

Agenda

- Introduction
  - Review RESTful Web Service
  - Contract Driven Development
- JSON
  - JSON Syntax
  - YAML
  - JSON Schema
- Swagger/OpenAPI
  - OpenAPI Schema
  - Base Properties
  - Resources and Operations
  - Reuse Definitions
  - CRUD
  - Object Relationships
  - Security
  - Best Practices

RESTful Web Services

RESTful Web Service

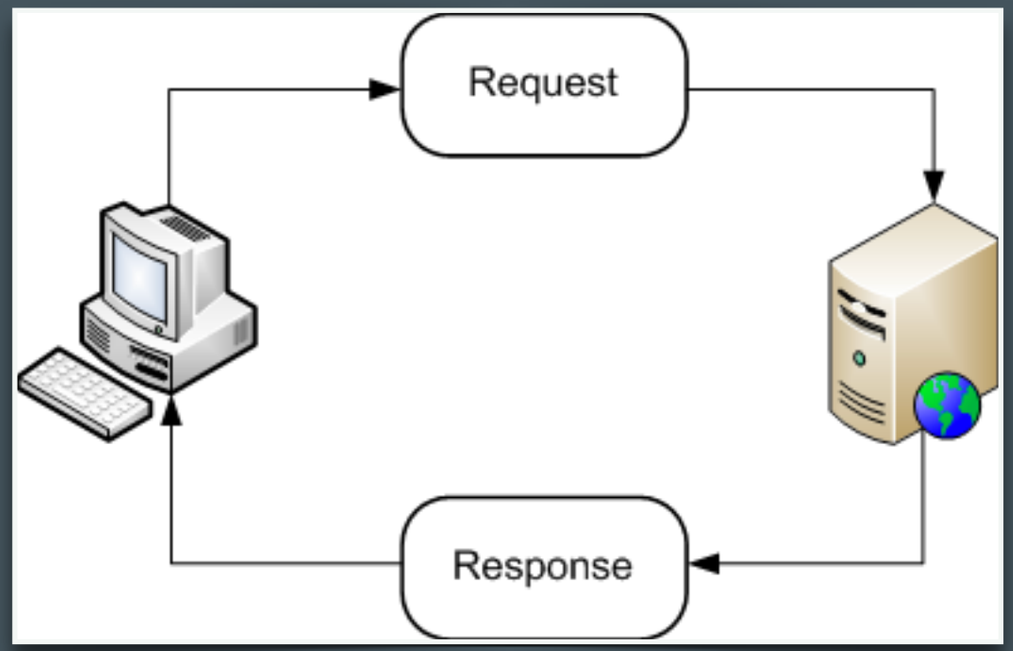
- REST … Representational State Transfer
- “REST-compliant web services allow the requesting systems to access and manipulate textual representations of web resources by using a uniform and predefined set of stateless operations”

Eigenschaften RESTful Web Services

- Zustandslos
- Transport Protokol:
  - HTTP
- Operationen:
  - GET, POST, PUT, …
- Web Resourcen:
  - URL´s/URI´s
- Datenprotokol:
  - JSON, XML, …

Eigenschaften HTTP

- Stateless
- Request-Response
- Metadata:
  - Request/Response Headers
- Implemented by:
  - Web Browser (User Agent)
  - Web Server
  - Kommandozeilen Tools
  - Development Libraries



RESTful Resource URL's

- URL's identifizieren Objekte und Funktionen
  - Collections (Listen):
    - http://api.service.com/resources?param=val
      - zB: http://api.service.com/v1/currencies?byRegion=EU
  - Eindeutiges Objekt:
    - http://service.com/.../resources/{id}
      - zB: http://service.com/v1/currencies/{currencyCode}
  - Relationen:
    - http://service.com/.../resources/{id}/relations
    - http://service.com/.../resources/{id}/relations/{relationsId}
      - zB: http://service.com/policies/privacy/translations/de

RESTful Operationen

URL	GET	PUT	PATCH	POST	DELETE
Collection: https://api.example.com/resources	Auflistung der Elemente	Ersetzen der kompletten Liste von Elementen	Typischerweise nicht benutzt	Anlegen eines neuen Elements in der Liste. Liefert üblicherweise die Id für das neue Element	Löscht die komplette Liste
Element: https://api.service.com/resources/item10	Holen des Elements	Ersetzen des Elements oder neu Anlegen falls es nicht existiert	Ersetzen von speziischen Attributen eines Elements	Typischerweise nicht benutzt.	Löscht ein Element aus der Liste

Demo - Simple REST Service

REST API Contract

REST Contract

- Beschreibt die Schnittstelle
- mittels Beschreibungssprache:
  - RESTful API Description Langauge (analog zu WSDL)
  - swagger.io (Open API Spec)
  - RAML

Contract Driven Development

- Interface-First Development
    - Die Schnittstelle existiert vor dem Code und nicht umgekehrt
- Schnittstelle wir mit Beschreibungssprache umgesetzt und nicht in Code
- Code Generieren für
    - Server
    - Client
- Mocking
- Dokumentation
- Clients (Konsumenten) und Server (Implementierungen) können getrennt von einander umgesetzt werden, sobald die Schnittstelle existiert
- Typischerweise von oder gemeinsam mit der Fachabteilung umgesetzt

REST API mit Open API Spec

```yaml
swagger: '2.0'

info:
  title: Redbull Policy Center Internal API
  description: The Redbull Policy Center Internal API provides restful
    operations for SPA.
  version: 1.3.0

basePath: /internal/api/policies

produces:
  - application/json

consumes:
  - application/json

tags:
  - name: policies_internal
    description: Policy Center Internal API

paths:
  /projects:
    get:
      summary: List projects
      tags:
        - policies_internal
      parameters:
        - in: query
          name: unpublished
          description: if true, returns only projects with unpublished
            policy translations.
          type: boolean
          required: false
          default: false
      x-timeout: 60000
      responses:
        200:
          description: a list of projects
          schema:
            type: array
            description: A list of project objects.
            items:
              $ref: '#/definitions/Project'
```

Redbull Policy Center Internal API

1.3.0

[Base URL: /internal/api/policies]

The Redbull Policy Center Internal API provides restful operations for SPA.

policies_internal Policy Center Internal API

GET	/projects	List projects
POST	/projects	Create project.
GET	/projects/{projectId}	Retrieve project by id.
PUT	/projects/{projectId}	Create or update project.
DELETE	/projects/{projectId}	Deletes a project.
GET	/projects/{projectId}/policies	Retrieve all policies assigned to a project
GET	/projects/{projectId}/policies/{policyId}/publishedUrls	Retrieve published urls for a policy and project

Code Generierung Server/Client

Generate Server ▾ **Generate Client ▾**

ada-server	jaxrs-resteasy	rust-server
aspnetcore	jaxrs-resteasy-eap	scala-lagom-server
erlang-server	jaxrs-spec	scalatra
finch	lumen	sinatra
go-server	msf4j	slim
haskell	nancyfx	spring
inflector	nodejs-server	undertow
java-pkmst	php-silex	ze-ph
java-play-framework	php-symfony	
java-vertx	pistache-server	
jaxrs	python-flask	
jaxrs-cxf	rails5	
jaxrs-cxf-cdi	restbed	

Generate Client ▾

ada	elixir	jmeter	scalaz
akka-scala	elm	kotlin	swagger
android	erlang-client	lua	swagger-yaml
apex	flash	objc	swift
bash	go	perl	swift3
clojure	groovy	php	swift4
cpprest	haskell-http-client	powershell	tizen
csharp	html	python	typescript-angular
csharp-dotnet2	html2	qt5cpp	typescript-angularjs
cwiki	java	r	typescript-aurelia
dart	javascript	ruby	typescript-fetch
dynamic-html	javascript-closure-angular	rust	typescript-jquery
eiffel	jaxrs-cxf-client	scala	typescript-node

Demo - Simple Rest Service mit Swagger

JSON

JSON

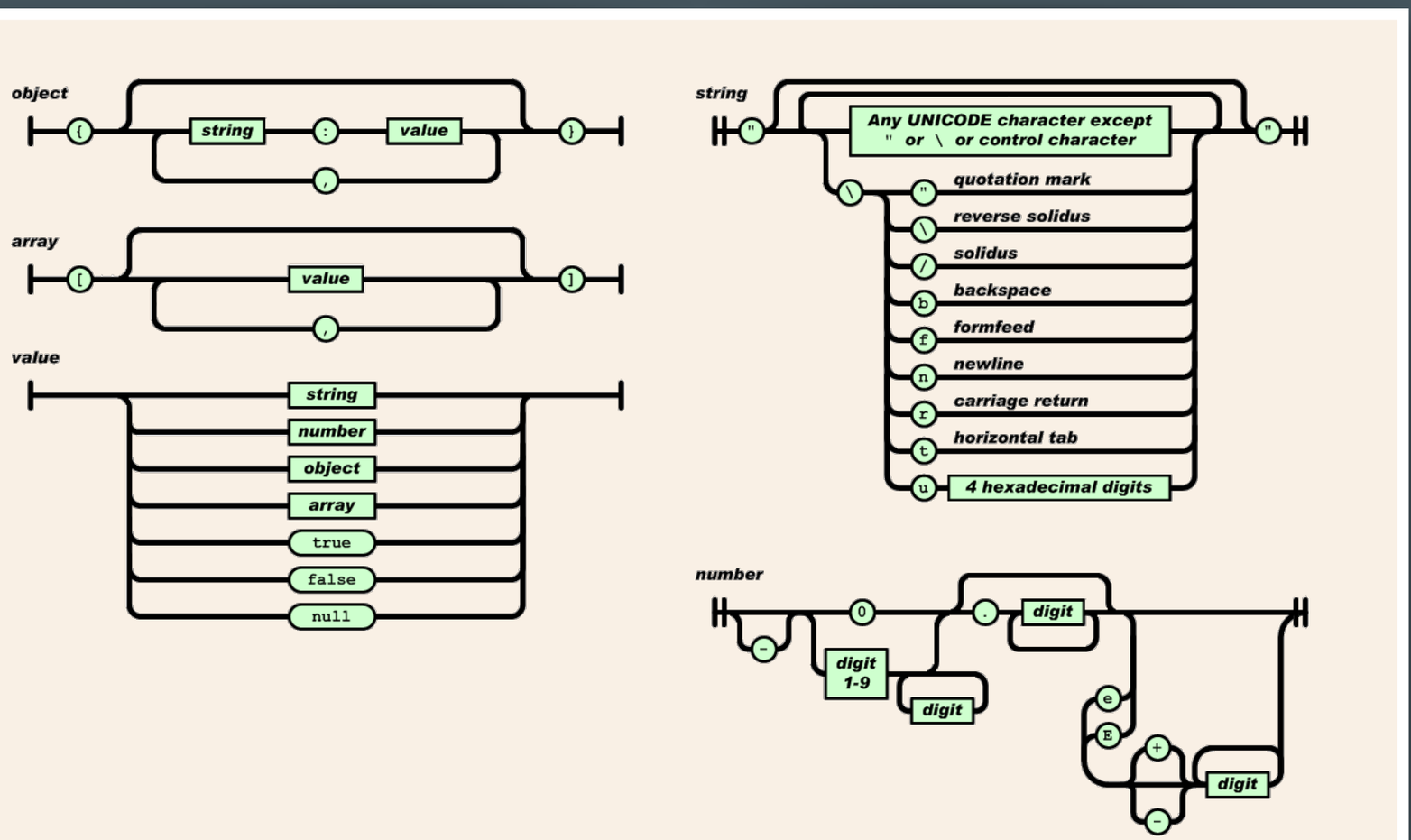
- “Javascript Object Notation”
- Hierarchical Structure
- Enables efficient parsing
- Javascript nativ (subset)
- Programming language agnostic
- Excellent tooling
- Spec: http://json.org

```json
{
    "stringValue": "value",
    "numberValue": 0,
    "booleanValue": true,
    "nullValue": null,

    "object": {
        "stringProperty": "value",
        "intProperty": 0,
        "boolProperty": true
    },

    "intArray": [0, 1, 2, 3, 5],
    "stringArray": ["a", "b", "c"],

    "objectArray": [{
            "stringProperty": "value",
            "intProperty": 0,
            "boolProperty": true
        },
        {
            "stringProperty": "xxx",
            "intProperty": 1,
            "boolProperty": false
        }
    ]
}
```

JSON vs. XML

- XML is a document markup language. JSON is not.
- XML has a schema. JSON does not.
- XML includes namespaces. JSON does not.
- JSON maps directly to object models. XML does not.
- JSON is less verbose than XML.
- JSON parsing is faster.

JSON Syntax


object
{
string
:
value
}
,
array
[
value
]
,
value
string
number
object
array
true
false
null
string
"
Any UNICODE character except
" or \ or control character
quotation mark
reverse solidus
solidus
backspace
formfeed
newline
carriage return
horizontal tab
4 hexadecimal digits
number
-
0
digit
1-9
digit
.
digit
e
E
+
-
digit

YAML

- YAML Ain't Markup Langage
- YAML is a human friendly data serialization standard for all programming languages
- http://yaml.org/

```yaml
swagger: '2.0'

info:
  title: Hello API
  version: 1.0.0

paths:
  /hello/{username}:
    get:
      produces:
        - application/json
      parameters:
        - name: username
          in: path
          required: true
          type: string
      responses:
        200:
          description: Success
          schema:
            type: string
          examples:
            application/json:
              jan
```

YAML vs. JSON

```yaml
swagger: '2.0'

info:
  title: Hello API
  version: 1.0.0

paths:
  /hello/{username}:
    get:
      produces:
        - application/json
      parameters:
        - name: username
          in: path
          required: true
          type: string
      responses:
        200:
          description: Success
          schema:
            type: string
          examples:
            application/json:
              jan
```

```json
{
  "swagger": "2.0",
  "info": {
    "title": "Hello API",
    "version": "1.0.0"
  },
  "paths": {
    "/hello/{username}": {
      "get": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "username",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "Success",
            "schema": {
              "type": "string"
            },
            "examples": {
              "application/json": "jan"
            }
          }
        }
      }
    }
  }
}
```

http://sangsoonam.github.io/2017/03/13/yaml-vs-json.html

Demo - JSON<->YAML

JSON Schema

- Describes existing data format(s).
- Provides clear human- and machine- readable documentation.
- Validates data which is useful for:
  - Automated testing.
  - Ensuring quality of client submitted data.
- https://json-schema.org/understanding-json-schema/

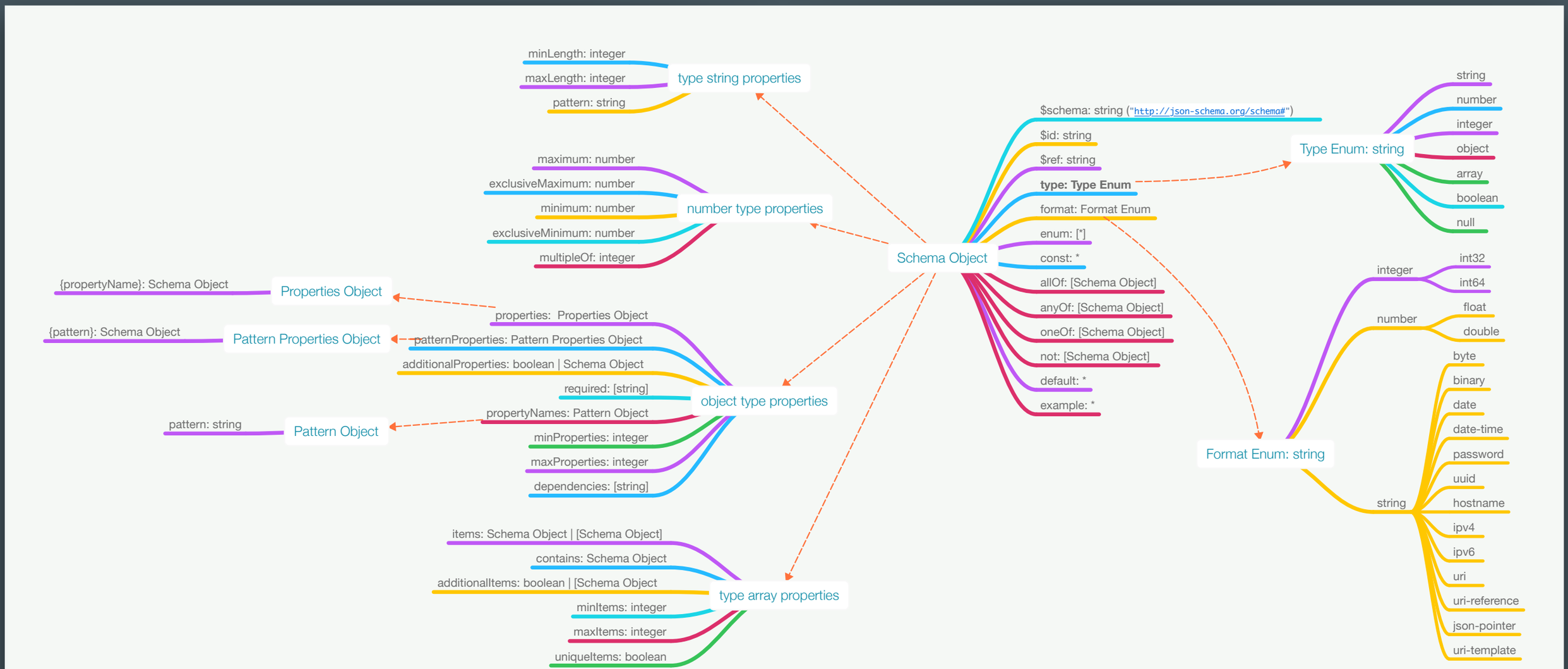
JSON Schema Beispiel

```
{
  "name": "George Washington",
  "birthday": "February 22, 1732",
  "address": "Mount Vernon, Virginia, United States"
}
```

```
{
  "first_name": "George",
  "last_name": "Washington",
  "birthday": "1732-02-22",
  "address": {
    "street_address": "3200 Mount Vernon Memorial Highway",
    "city": "Mount Vernon",
    "state": "Virginia",
    "country": "United States"
  }
}
```

```
{
  "type": "object",
  "properties": {
    "first_name": { "type": "string" },
    "last_name": { "type": "string" },
    "birthday": { "type": "string", "format": "date-time" },
    "address": {
      "type": "object",
      "properties": {
        "street_address": { "type": "string" },
        "city": { "type": "string" },
        "state": { "type": "string" },
        "country": { "type" : "string" }
      }
    }
  }
}
```

JSON Schema Object



JSON Schema Tools

- Command Line
  - *ajv*: Another JSON Schema Validator
    - https://ajv.js.org/
- Online
  - JSON Schema Tool
    - https://www.jsonschema.net/
  - JSON Editor Online
    - https://jsoneditoronline.org/
  - JSON Schema Validator
    - https://www.jsonschemavalidator.net/

JSON Schema Demo

Lab 01 - Create a JSON Schema

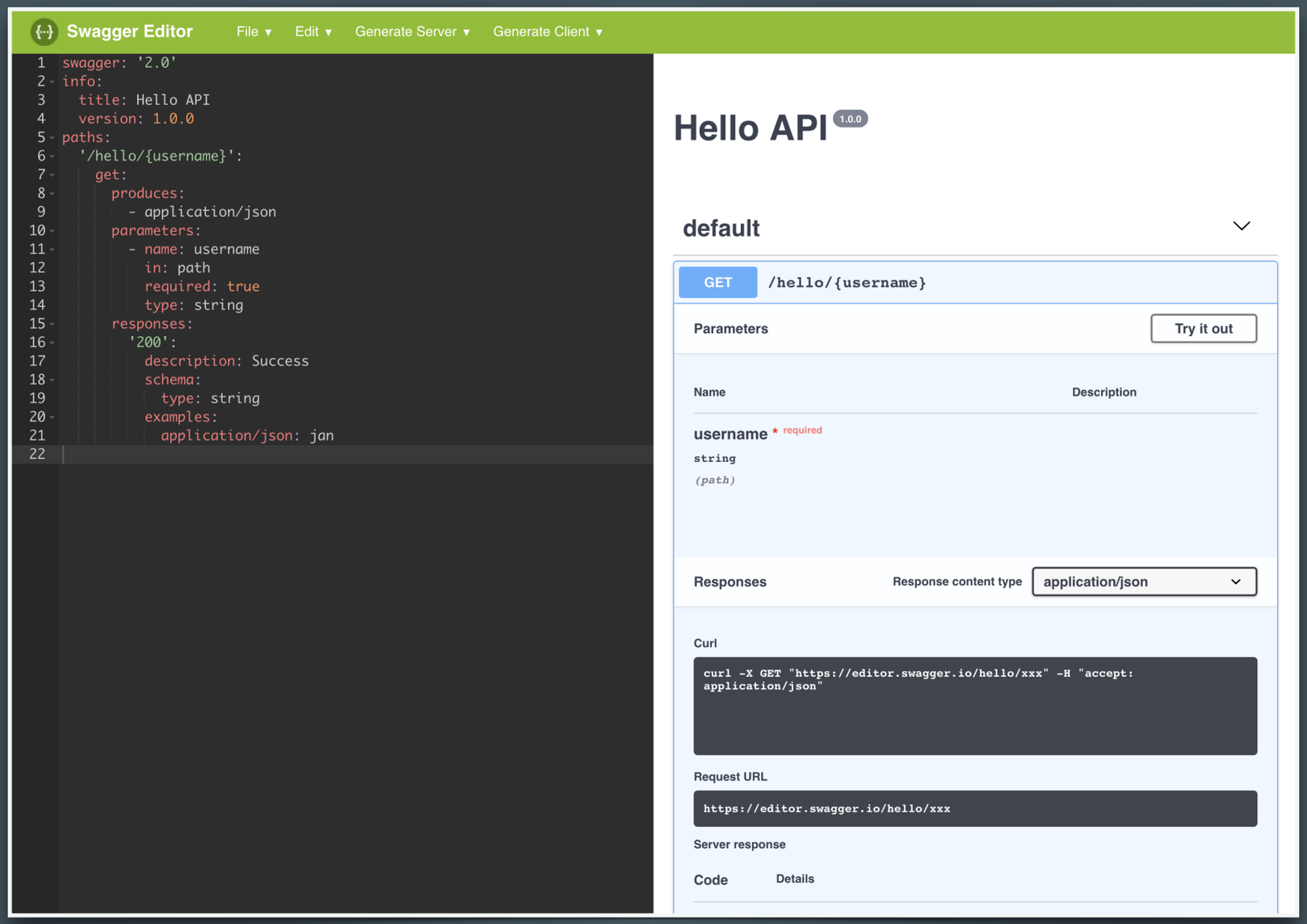
https://github.com/Swagger-Training/swagger-training/tree/master/labs/01_json_schema

Swagger Tools

Swagger Tools

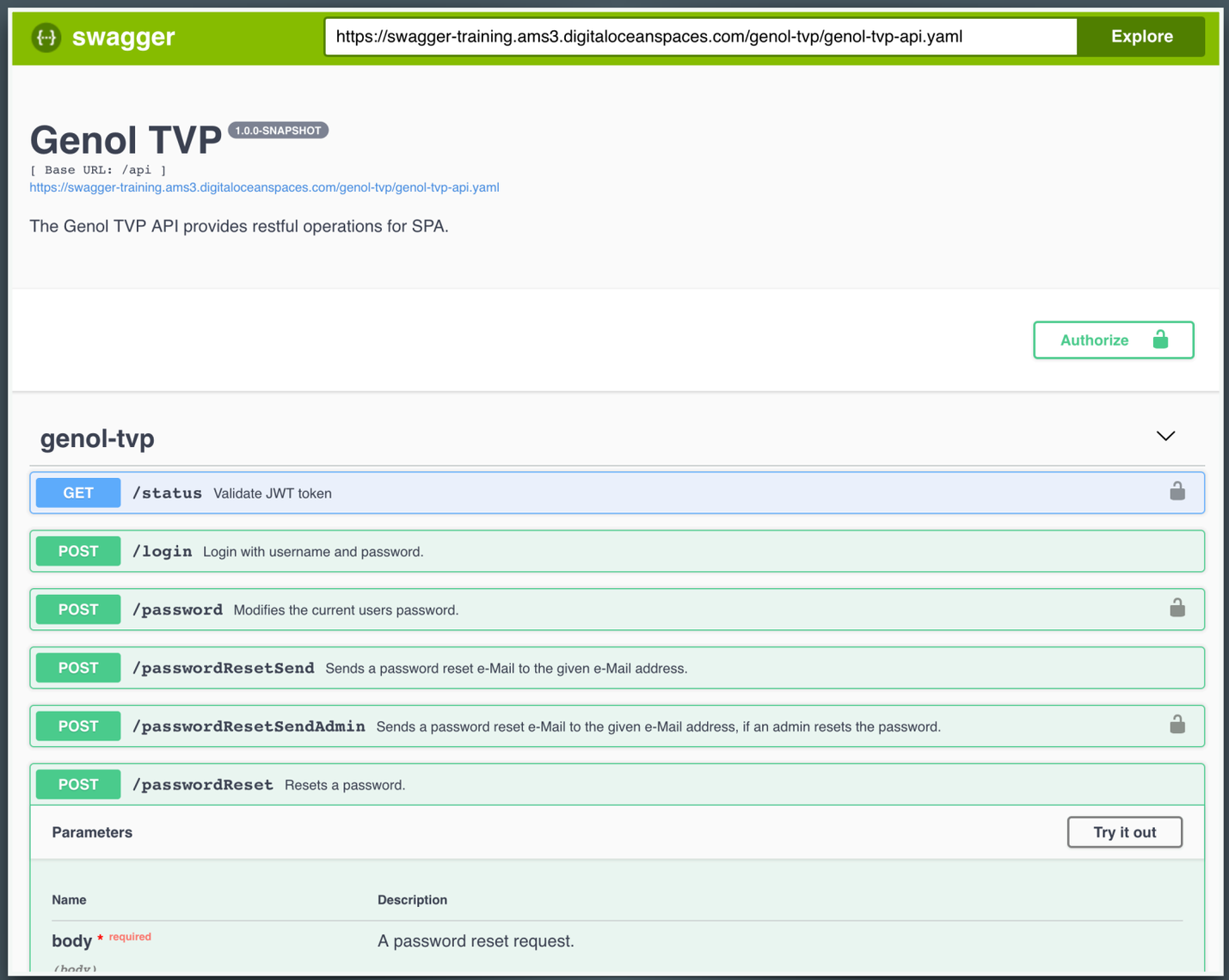
- Swagger Editor
- Swagger UI
- Swagger Codegen
- Swagger Hub

Swagger Editor



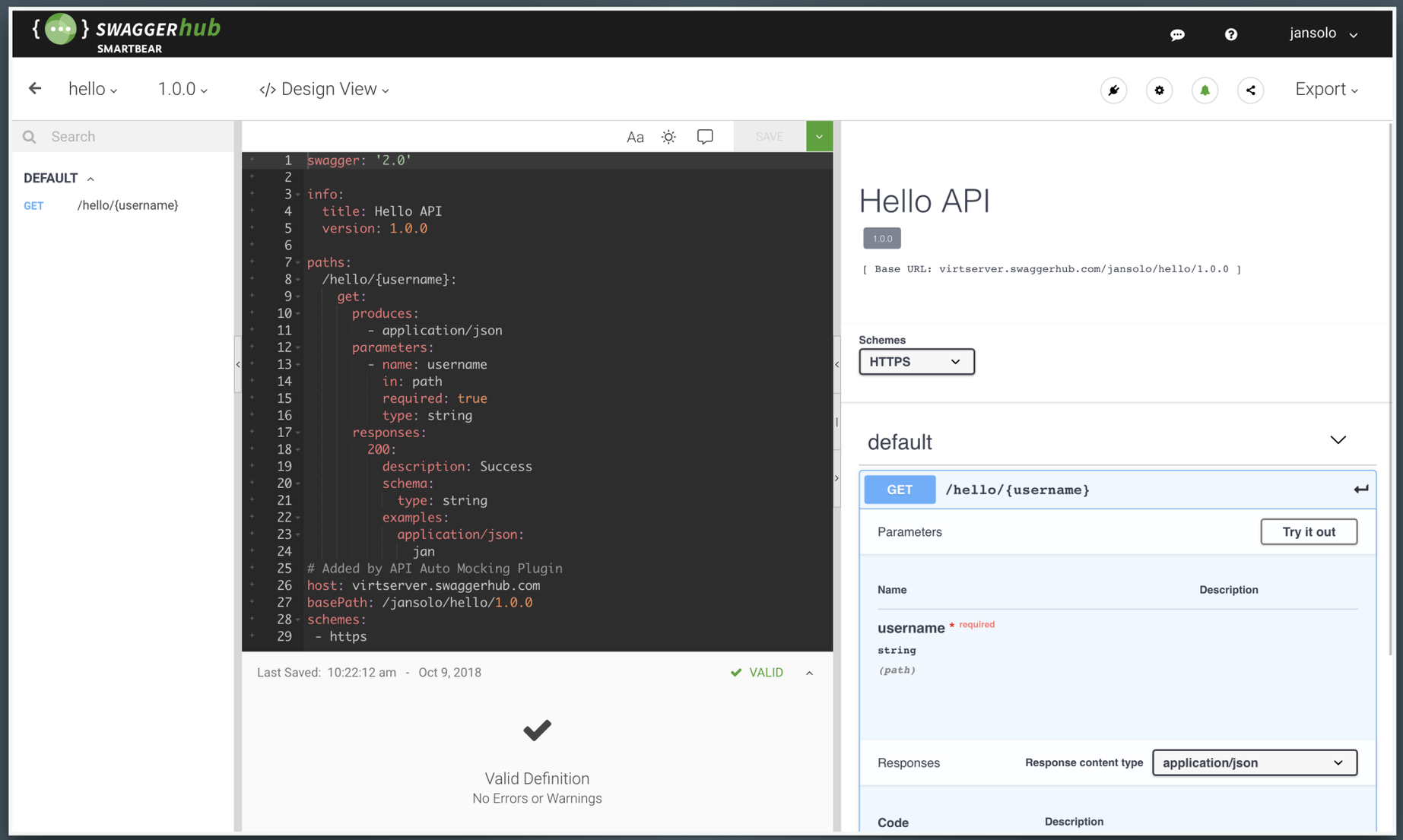
https://editor.swagger.io/

Swagger UI



https://editor.swagger.io/

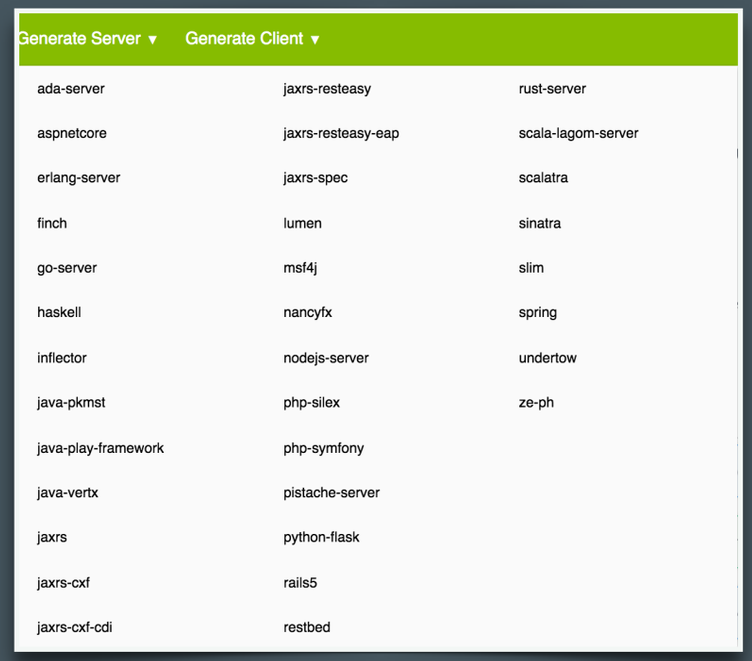
Swagger-Hub

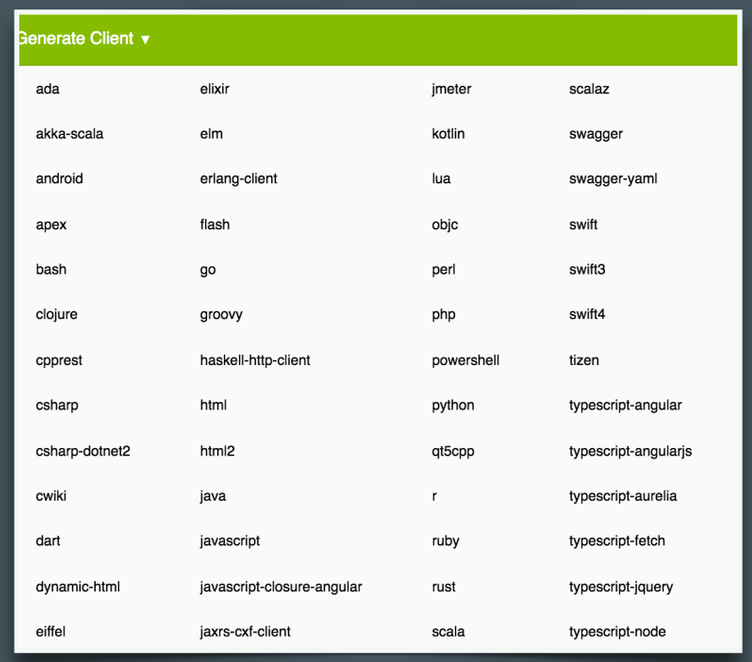


https://app.swaggerhub.com

Swagger Codegen

- Code generation from Swagger/Open API Spec
  - Server
  - Client
- Template-driven
  - extensible through own Templates
- https://github.com/swagger-api/swagger-codegen





Demo Swagger Tools

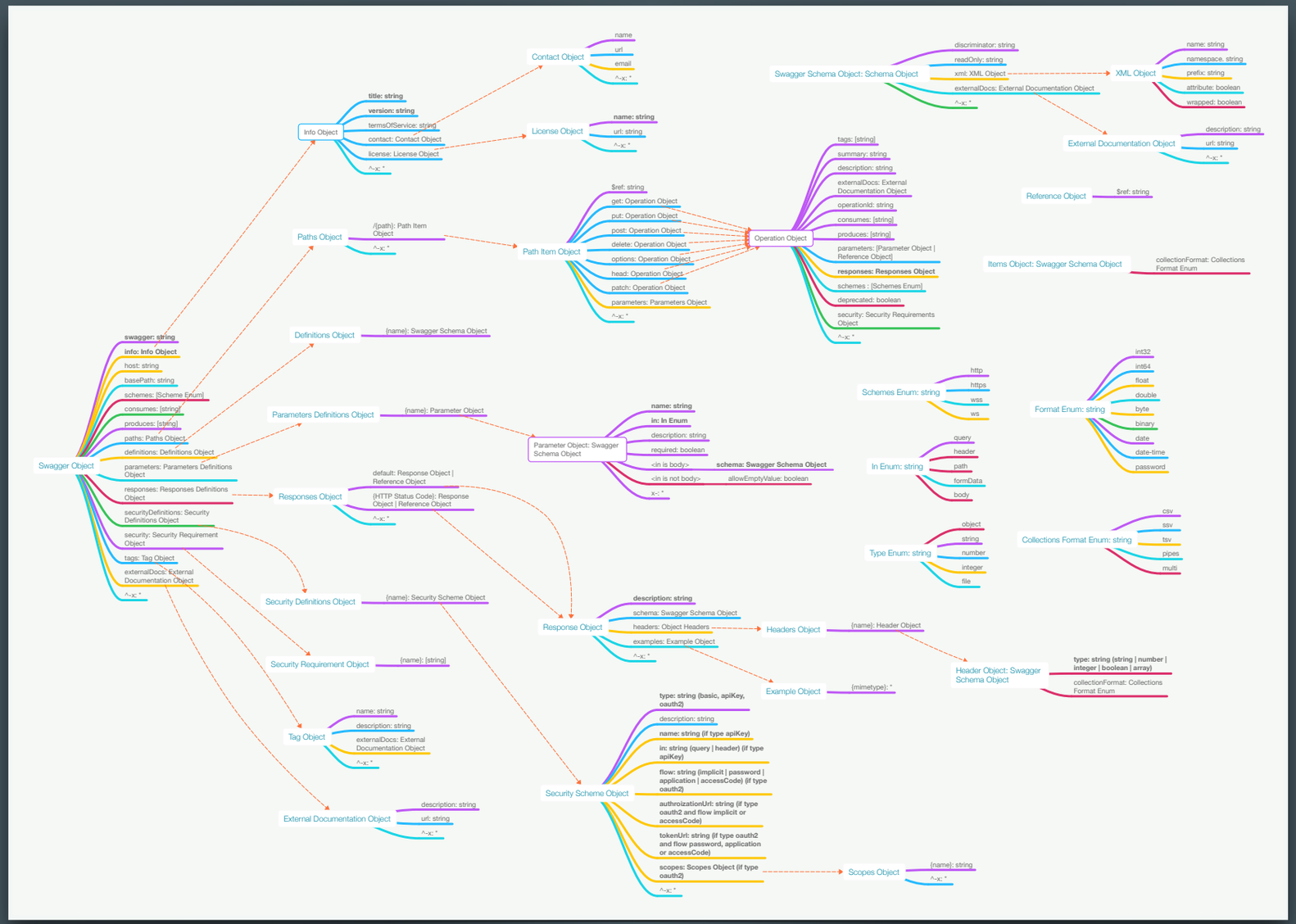
Swagger / OpenAPI Spec

Swagger/OpenAPI Spec

- “Swagger is a project used to describe and document RESTful APIs.”
- Versions:
  - OpenApi Specification 2.0
    - https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md
  - OpenAPI Specification 3.0
    - https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.2.md
- Are written in JSON or YAML

```yaml
swagger: '2.0'

info:
  title: Policies API
  version: '1.0.0'

basePath: /policies-api/1

consumes:
    - application/json
produces:
    - application/json
    - text/plain

paths:
    /policies:
        get:
            summary: Gets all policies
            description: Gets policies by given parameters
            parameters:
                -   name: audience
                    in: query
                    type: string
                    description: Query policies by audience
                -   name: policyType
                    in: query
                    type: string
                    description: Query policies by policy type
                -   name: mandatory
                    in: query
                    type: boolean
                    description: Query mandatory policies
            responses:
                200:
                    description: Successfully retreived policies
                    schema:
                        type: array
                        items:
                            $ref: '#/definitions/policy'
                    examples:
                        application/json:
                            -   title: Privacy Policy International
                                policyType: privacy
                                audience: International
                                defaultLanguage: en
                                mandatory: true
                                translations:
                                    -   title: سياسة الخصوصية الخاصة بريد بول
                                        releaseDate: 2017-04-21T11:50:37.358Z
                                        language: ar
```

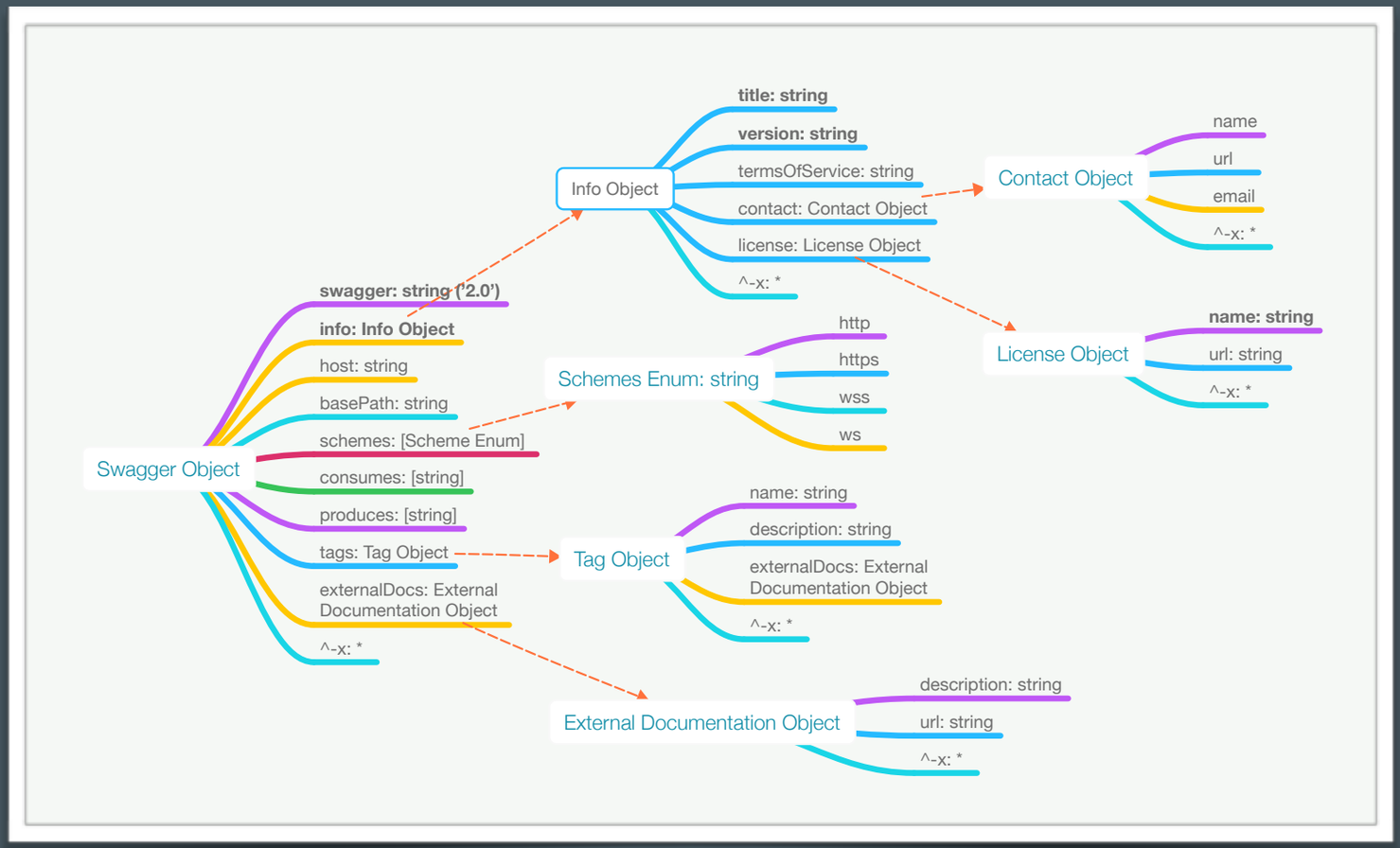
Swagger Schema



http://openapi-map.apihandyman.io/?version=2.0

Base Properties

- *swagger: '2.0'* (swagger spec version)
- *info*
  - *title*
  - *version*
  - *contact*
  - *license*
- *basePath*
- *host*
- *schemes*
  - *http, https, ws, wss*
- *produces*
- *consumes*
- *tags*
- *externalDocs*
- *x-* (vendor extensions)



Mime types

- Mime types can be configured for request and response body
  - Request -> *consumes* property
    - Request “Content Type” header must be set to a mime type defined in ***consumes*** property
  - Response -> ***produces*** property
- Multi mime types allowed for ***consumes*** and ***produces***
- Can be set on global API or resource operation level

```yaml
swagger: '2.0'

info:
  title: Policies API
  version: '1.0.0'

basePath: /policies-api/1

consumes:
    - application/json
produces:
    - application/json
    - text/plain
```

Lab 02 - Swagger Base Properties

https://github.com/Swagger-Training/swagger-training/tree/master/labs/02_swagger_base

Resources and Operations

- Define methods of an API
- Resources are defined as path *{/pathname}* properties in the path object
- Path may contain variable parameters *{name}*
  - Path parameters must be defined in in ***parameters*** property with ***required: true***
- Operations are defined in the operations object
  - Each operation is a (optional) property
  - Supported operations: *GET*, *PUT*, *POST*, *DELETE*, *OPTIONS*, *HEAD*, *PATCH*
- Each operation defines the details of an API method:
  - Parameters, responses, mime types, schemes, security constraints

```yaml
paths:
    /policies:
        get:
            summary: Gets all policies
            description: Gets policies by given parameters
            parameters:⟺
            responses:⟺

        post:
            summary: New policy
            description: Creates a new policy
            parameters:⟺
            security:⟺
            responses:⟺

    /policies/{policyId}:
        get:
            summary: Gets a policy
            description: Gets policy by id
            parameters:⟺
            responses:⟺

        put:
            summary: Creates or updates a policy
            description: Creates or updates a policy by id
            parameters:⟺
            security:⟺
            responses:⟺

        delete:
            summary: Deletes a policy
            description: Deletes a policy by id
            parameters:⟺
            security:⟺
            responses:⟺
```

Parameters

- Define the expected request parameters
- Supports ***body***, ***query***, ***path***, ***formData*** or ***header*** as ***in*** types
- Parameters can be defined globally or on operation level

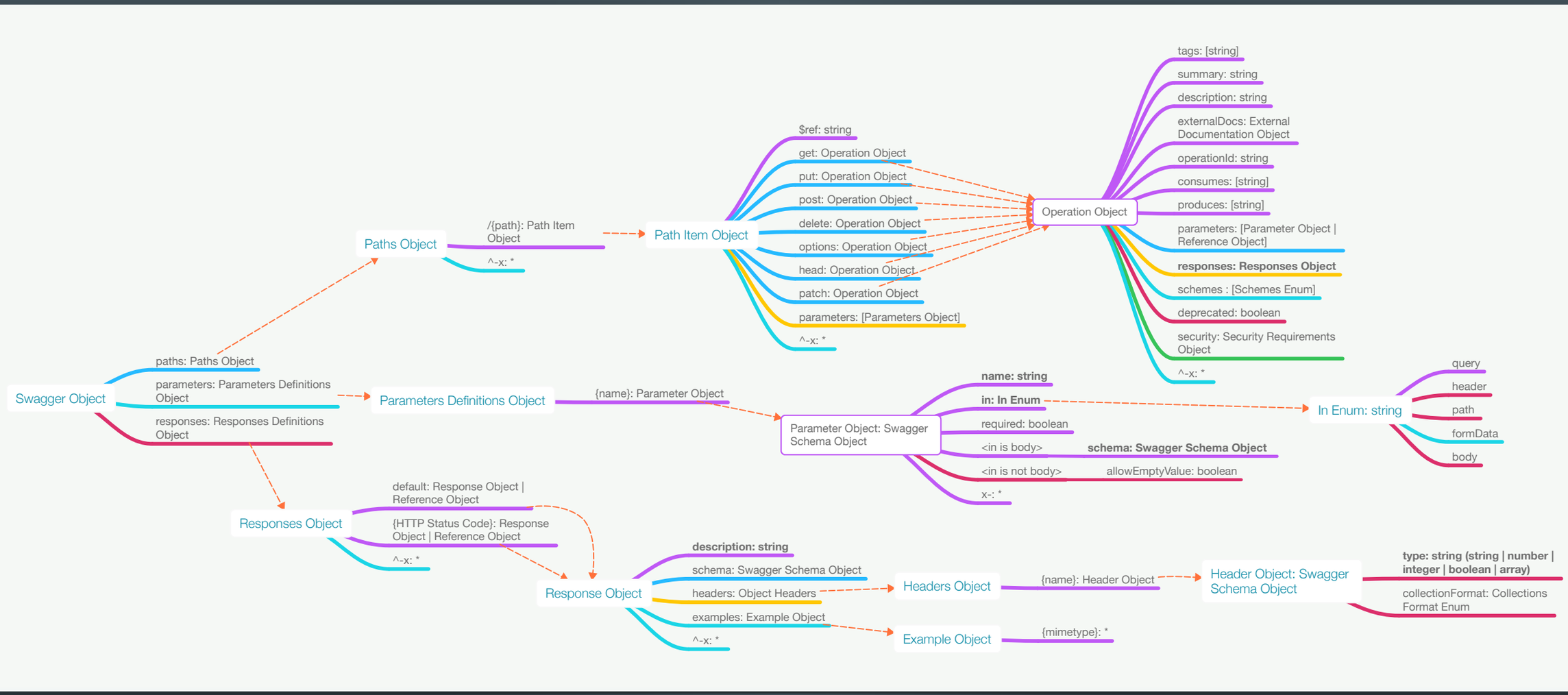
```yaml
parameters:
    -   name: audience
        in: query
        type: string
        description: Query policies by audience
    -   name: policyType
        in: query
        type: string
        description: Query policies by policy type
    -   name: mandatory
        in: query
        type: boolean
        description: Query mandatory policies
```

Responses

- Are defined in the ***responses*** object of the operations object
- Properties of the responses object are HTTP status codes (*200 OK*)
- Each status property contains the schema definition of the response type
- Responses can be defined globally or on operation level

```yaml
responses:
    200:
        description: Successfully retreived policies
        schema:
            type: array
            items:
                type: object
                description: A policy
                properties:
                    id:
                        type: string
                        description: An unique id
                    title:
                        type: string
                        description: The title of the policy
```

Operations Schema



Lab 03 - Creating a Resource Operation

https://github.com/Swagger-Training/swagger-training/tree/master/labs/03_swagger_get

Definitions and References

- Definitions contain reusable schemas.
- Are defined in the *definitions* root property
- Can be referenced from *schema* properties
    - body parameters
    - responses
- Can be externalized into separate schema files
- Are referenced with reference objects: *$ref*
    - Format: *<URI>#<Anchor>*
        - internal references:
            - *$ref: '#/definitions/policy'*
        - external references:
            - *$ref: './model.json#/definitions/policy'*

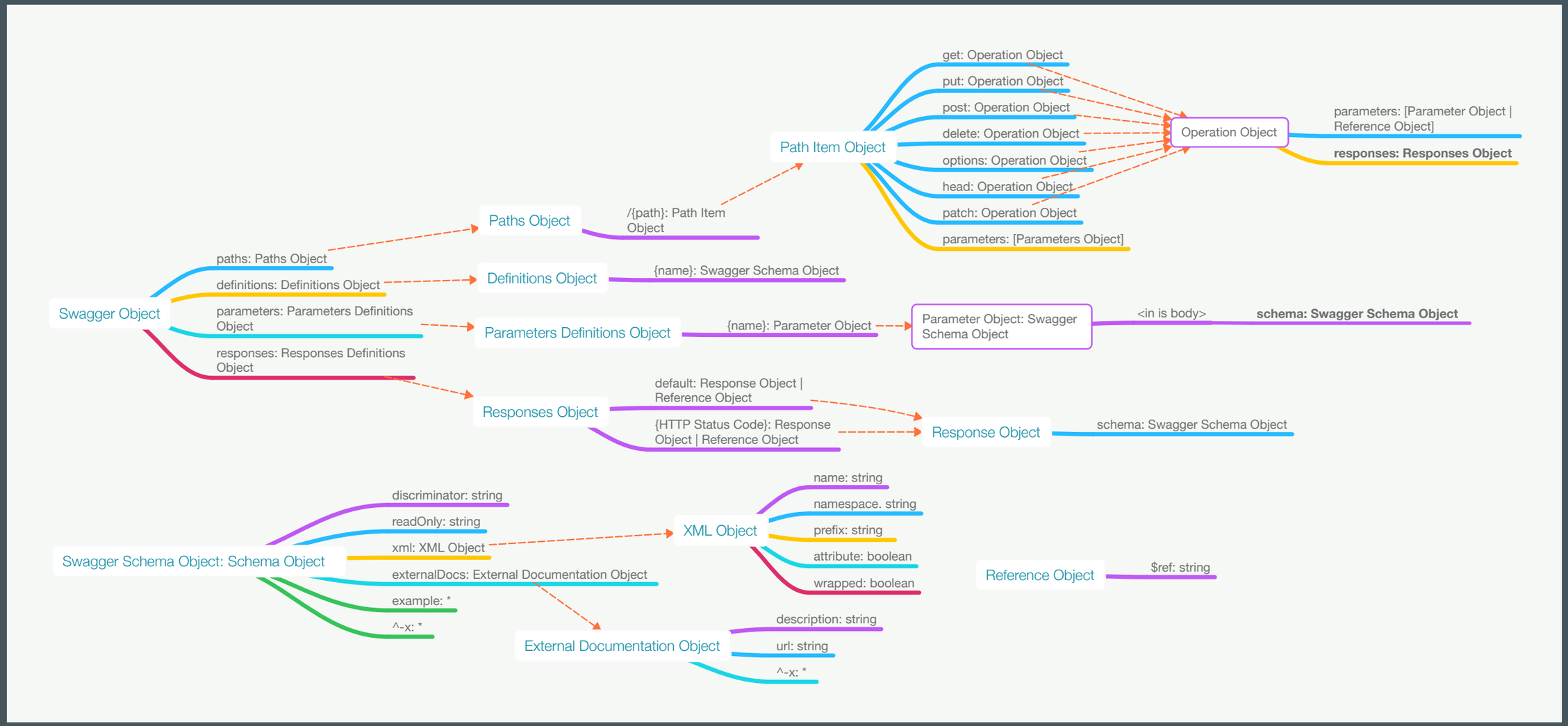
```yaml
definitions:
    policy:

    translation:
        type: object
        description: A translation for policy
        properties:
            title:
                type: string
                description: The localized title of the policyType
            releaseDate:
                type: string
                format: date-time
                description: The release date of the translation revision
            language:
                $ref: '#/definitions/language'
            version:
                type: integer
                description: the major version of the policy translations
        required:
            - title
            - releaseDate
            - language
            - version

    policyType:

    audience:

    language:
        type: string
        description: A policy language
        minLength: 2
        maxLength: 2
        default: en
        example: de
```

Parameter & Response Definitions

- Reusable Parameter Definitions can be defined on …
  - … global specification level in the property ***parameters***
  - … path item level for all operations of the resource in the property ***parameters***
- Reusable Response Definitions in the root level ***responses*** property

```yaml
responses:
  success:
    description: Object not found.
    schema:
      $ref: '#/definitions/Success'
    examples:
      application/json:
        message: The operation succeeded.
```

Definitions, Parameter Definitions & Response Definitions



Lab 04 - Reuse

https://github.com/Swagger-Training/swagger-training/tree/master/labs/04_swagger_reuse

CRUD Operations

- Create, Update, Delete
- Are represented by HTTP methods, not separate resources
  - GET
  - POST
  - PUT
  - DELETE
  - PATCH
  - HEAD
  - OPTIONS

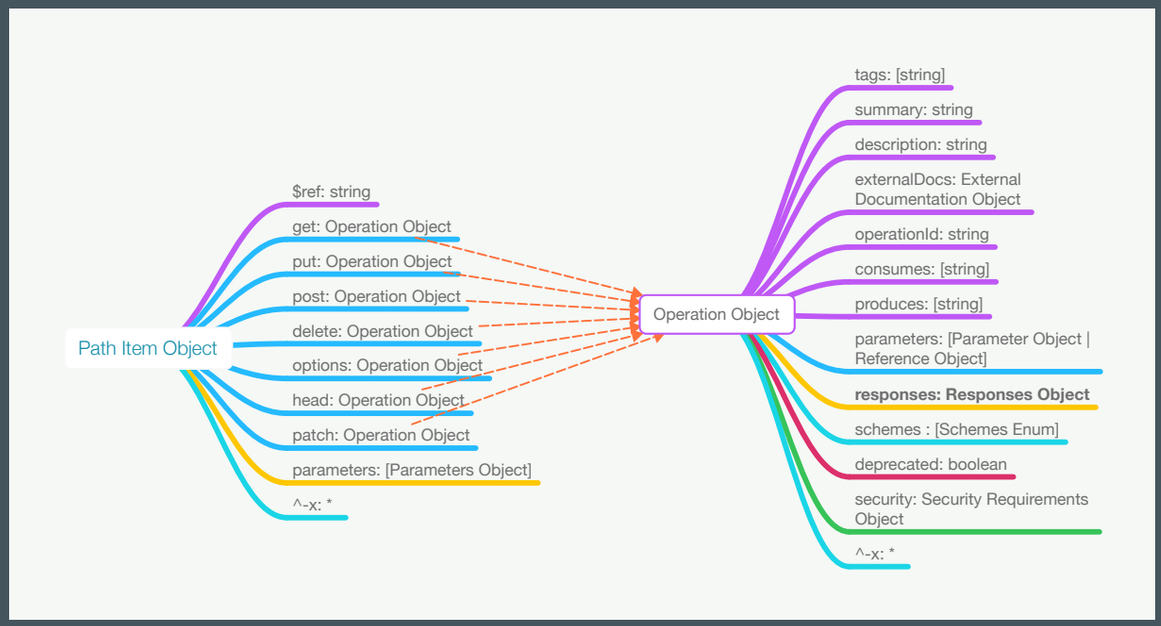
```
paths:
  /policies:
    get:⟺

    post:⟺

  /policies/{policyId}:
    get:⟺

    put:⟺

    delete:⟺
```

CRUD Operations

URL	GET	PUT	PATCH	POST	DELETE
collection: https://api.example.com/resources	lists elements	replaces the whole list	typically not implemented	creates a new element, typically returns the newly created id or element	deletes the whole list
element: https://api.service.com/resources/item10	fetches a single element (by id)	replaces an element, or creates a new one, if it does not exist	replaces a specific property of an element	typically not implemented	deletes an element from the list

CRUD Response Codes

GET	PUT	PATCH	POST	DELETE
200 OK	200 OK 201 Created	200 OK	201 Created	203 No Content




```yaml
paths:
    /policies:
        get:
            summary: Gets all policies
            description: Gets policies by given parameters
            parameters:
            responses:
                200:

        post:
            summary: New policy
            description: Creates a new policy
            parameters:
            security:
            responses:
                201:

    /policies/{policyId}:
        get:
            summary: Gets a policy
            description: Gets policy by id
            parameters:
            responses:
                200:

        put:
            summary: Creates or updates a policy
            description: Creates or updates a policy by id
            parameters:
            security:
            responses:
                200:

        delete:
            summary: Deletes a policy
            description: Deletes a policy by id
            parameters:
            security:
            responses:
                204:
                404:
```

Lab 05 - CRUD Operations

https://github.com/Swagger-Training/swagger-training/tree/master/labs/05_swagger_crud

Error Handling

- Use HTTP Status Codes to transmit error conditions when calling a resource
  - 2xx (Success category)
    - 200 Ok, 201 Created, 204 No Content
  - 3xx (Redirection Category)
    - 304 Not Modified
  - 4xx (Client Error Category)
    - 400 Bad Request, 401 Unauthorized, 403 Forbidden, 404 Not Found, 410 Gone
  - 5xx (Server Error Category)
    - 500 Internal Server Error, 503 Service Unavailable
- Create and use reusable error responses

```yaml
paths:
    /policies:⟺

    /policies/{policyId}:
        get:⟺

        put:⟺
        delete:
            summary: Deletes a policy
            description: Deletes a policy by id
            parameters:⟺
            responses:
                204:⟺
                404:
                    $ref: '#/responses/notFound'

responses:
  notFound:
    description: Not found
    schema:
        $ref: '#/definitions/error'
    examples:
        text/plain:
            element not found

definitions:
    error:
        type: string
        description: An error message
```

Lab 06 - Error Handling

https://github.com/Swagger-Training/swagger-training/tree/master/labs/06_swagger_errors

Model Relationships

Schema references between model objects

policy {

	description:	*A policy*
	id*	string($uuid)
	title*	string *maxLength: 50* The title of the policy
	policyType*	policyType string A valid policy type Enum:
	audience	▸ Array [3] audience string *default: international* The target audience Enum:
	defaultLanguage	▸ Array [2] language string *minLength: 2* *maxLength: 2* *default: en* *example: de* A policy language
	mandatory	boolean *default: false*
	translations	▾ [**translation** ▾ {
		description: *A translation for policy*
		title* string The localized title of the policyType
		releaseDate* string($date-time) The release date of the translation revision
		language* language string *minLength: 2* *maxLength: 2* *default: en* *example: de* A policy language
		version* integer the major version of the policy translations
		}]

```yaml
definitions:
    policy:
        type: object
        description: A policy
        properties:
            id:⟺
            title:⟺
            policyType:⟺
            audience:⟺
            defaultLanguage:⟺
            mandatory:⟺
            translations:
                type: array
                items:
                    $ref: '#/definitions/translation'
        required:⟺
        example:⟺

    translation:
        type: object
        description: A translation for policy
        properties:
            title:⟺
            releaseDate:⟺
            language:⟺
            version:⟺
        required:⟺
```

Resource Relationships

Relationships between objects are represented in the resource url

GET	`/policies/{policyId}/translations`	Gets policy translations
GET	`/policies/{policyId}/translations/{language}`	Gets a specific translation
DELETE	`/policies/{policyId}/translations/{language}`	Deletes a specific translation
PUT	`/policies/{policyId}/translations/{language}`	Creates or updates a translation

```
paths:
    /policies:

    /policies/{policyId}:

    /policies/{policyId}/translations:

    /policies/{policyId}/translations/{language}:
```

API Relationships Best Practices

- Reduce resource calls
    - Avoid 1*N calls to resolve relationships
- Optimize transferred data volume
    - Only transmit what is required by the API client
- Model object relationships as compositions for objects that do not exist on their own
- Model the resources with the consumers in mind
    - Do not mirror the relational database scheme
    - What does the API client need to perform its operation?

Lab 07 - Object Relationships

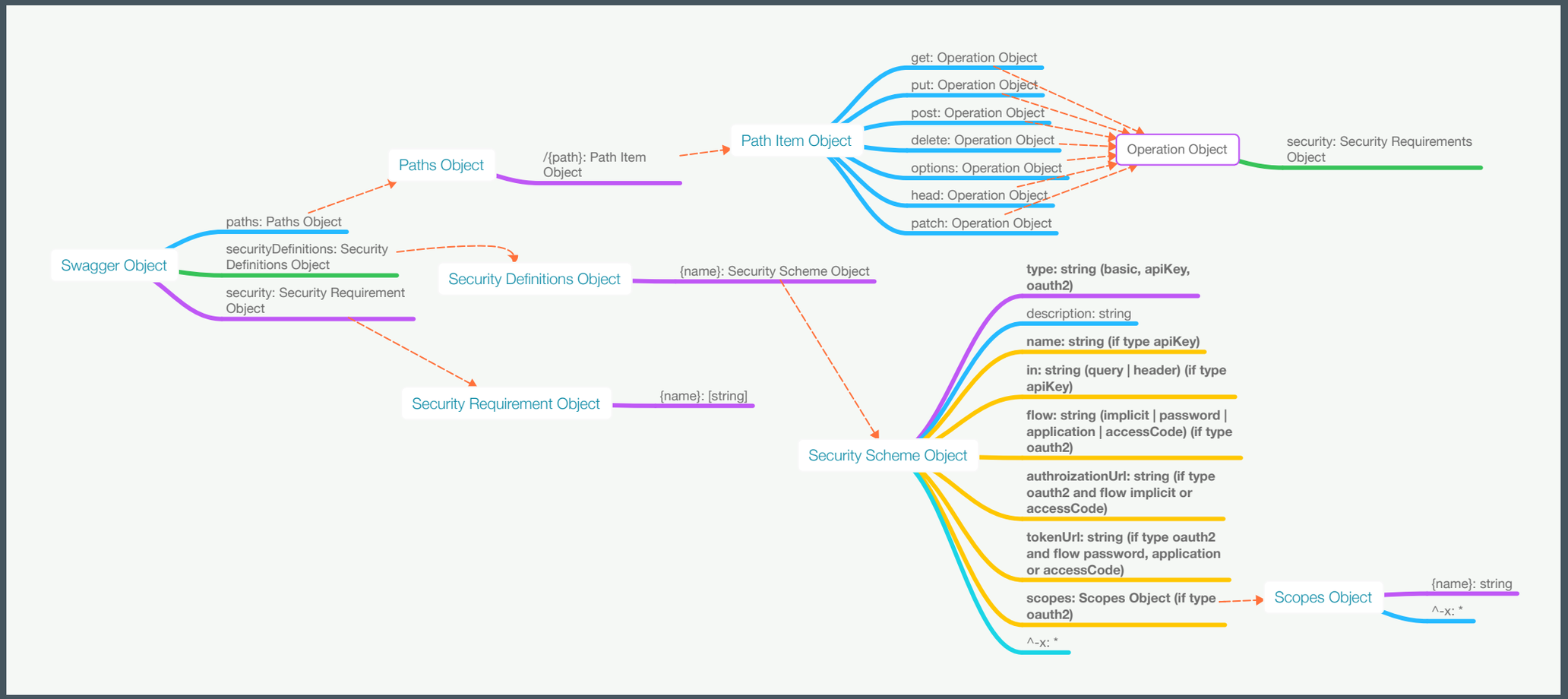
https://github.com/Swagger-Training/swagger-training/tree/master/labs/07_swagger_relationships

Swagger Security

- Security definitions
  - Configures security options for an API
  - Supports OAUTH, Basic Auth and API Key authentication
- Security requirements
  - Assigns security definitions to resource operations
  - Can be set globally for the API or operation level
  - Multiple security definitions can be used on one resource (AND, OR, override)

```
securityDefinitions:
    external:
        type: apiKey
        name: policy-api-key
        in: header
    admin:
        type: basic

security:
    -   external: []
```

Swagger Security Scheme



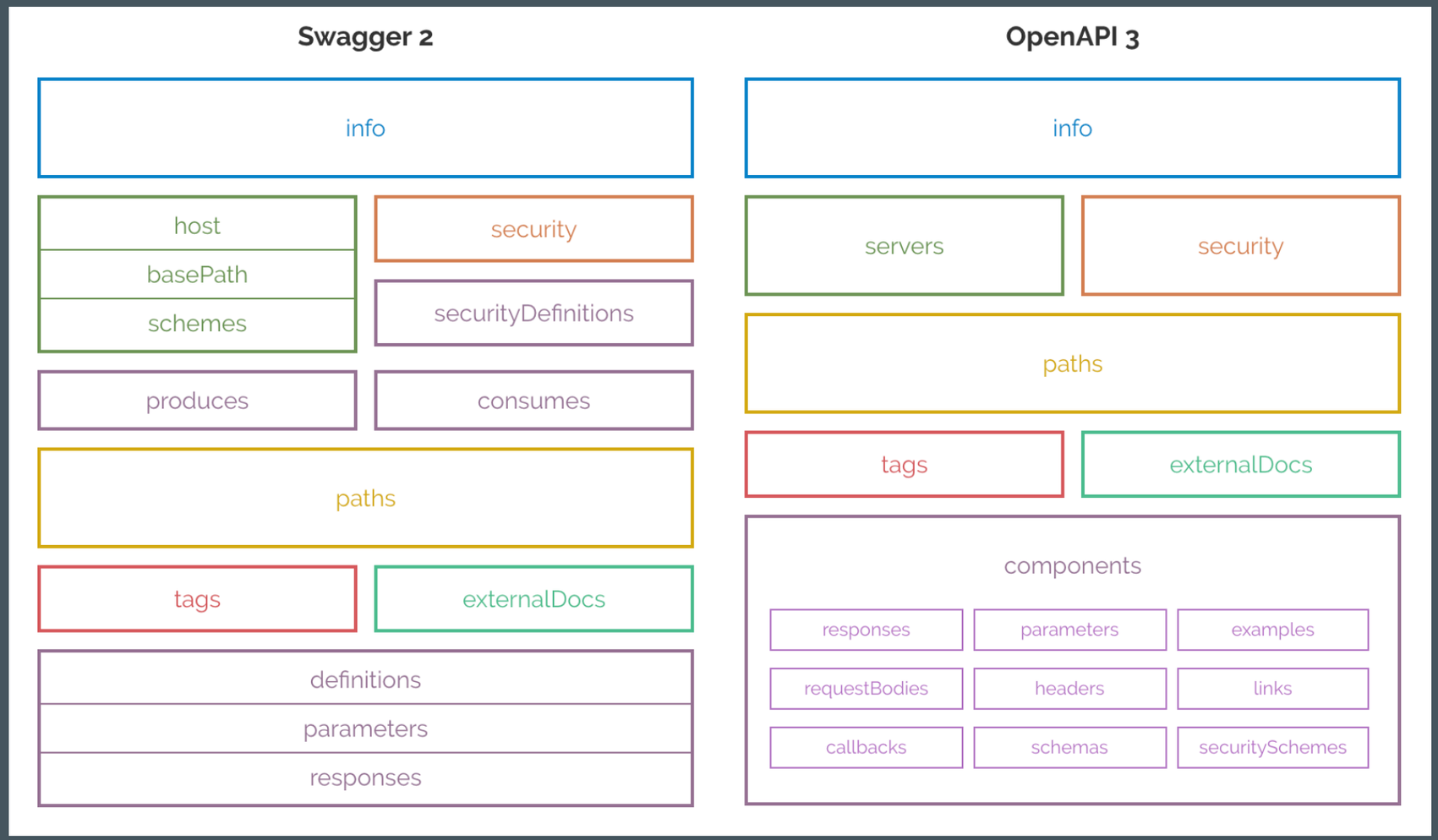
Lab 08 - Securing the API

https://github.com/Swagger-Training/swagger-training/tree/master/labs/08_swagger_security

REST API Design Best Practices

- The paths should contain the plural form of resources and the HTTP method should define the kind of action to be performed on the resource.
- The URL is a sentence, where resources are nouns and HTTP methods are verbs.
- Use HTTP response status codes.
- Searching, sorting, filtering and pagination -> No new API's methods, instead append query params to existing GET methods.
- Versioning -> Use version numbers in the API base url. Change version number on the url with every breaking API change, but not bug fixes and extensions.

OpenAPI Spec 3.0



https://blog.readme.io/an-example-filled-guide-to-swagger-3-2/