Presented by
Zdeněk Tomis

WHAT IS OPENAPI?

An open standard for defining RESTful APIs

Enables machine-readable descriptions of API structure and behavior

Originated in Swagger, became a standalone initiative in 2015

WHAT IS SWAGGER?

A suite of tools for designing, documenting, and consuming APIs

Initially started as a framework that combined API description with tools

Evolved as OpenAPI became a standard

RELATIONSHIP BETWEEN OPENAPI AND SWAGGER

OpenAPI is the specification format

Swagger is the toolset that supports OpenAPI

OpenAPI is the “blueprint,” Swagger is the “toolbox”

WHY USE OPENAPI?

Ensures consistent API design across projects

Promotes collaboration across teams

Enables automated testing and CI/CD integration

Enhances developer experience

IMPORTANCE OF API DOCUMENTATION

Improves usability and ease of adoption

Increases productivity and reduces support requests

Enhances onboarding for new developers

CORE COMPONENTS OF OPENAPI SPECIFICATION

Paths - Represent API endpoints

Operations - Actions on paths (GET, POST, etc.)

Parameters - Inputs for API (query, path, body)

Responses - Expected outputs with status codes

Security - Authentication and authorization

STRUCTURE OF AN OPENAPI SPECIFICATION

JSON or YAML format

High-level sections: info, servers, paths, components

Organized hierarchy for detailed API description

```yaml
openapi: 3.1.0
info:
  title: Train Travel API
  description: |
    API for finding and booking train trips across Europe.

  version: 1.0.0
  contact:
    name: Train Support
    url: https://example.com/support
    email: support@example.com
  license:
    name: Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International
    identifier: CC-BY-NC-SA-4.0
servers:
  - url: https://api.example.com
    description: Production
    x-internal: false

  - url: https://mocks.example.com/rest
    description: Mock Server
    x-internal: false
```

PATHS IN OPENAPI

Define API endpoints, such as `/users` or `/products/{id}`

Organized by resource (e.g., users, products)

Example: `GET /users` retrieves a list of users

```yaml
paths:
  /stations:
    get:
      summary: Get a list of train stations
      description: Returns a paginated and searchable list of all train stations.
      operationId: get-stations
      tags:
        - Stations
      parameters:
        - $ref: '#/components/parameters/page'
        - $ref: '#/components/parameters/limit'
        - name: coordinates
          in: query
          description: >
            The latitude and longitude of the user's location, to narrow down
            the search results to sites within a proximity of this location.
          required: false
          schema:
            type: string
          example: 52.5200,13.4050
        - name: search
          in: query
```

https://github.com/bump-sh-examples/train-travel-api/blob/main/openapi.yaml

OPERATIONS IN OPENAPI

HTTP methods (GET, POST, PUT, DELETE) for actions on paths

Defined under each path for specific actions

Example: `POST /users` creates a new user

```yaml
paths:
  /stations:
    get:
      summary: Get a list of train stations
      description: Returns a paginated and searchable list of all train stations.
      operationId: get-stations
      tags:
        - Stations
      parameters:
        - $ref: '#/components/parameters/page'
        - $ref: '#/components/parameters/limit'
        - name: coordinates
          in: query
          description: >
            The latitude and longitude of the user's location, to narrow down
            the search results to sites within a proximity of this location.
          required: false
          schema:
            type: string
          example: 52.5200,13.4050
        - name: search
          in: query
```

https://github.com/bump-sh-examples/train-travel-api/blob/main/openapi.yaml

PARAMETERS IN OPENAPI

Specify inputs for API operations

Types: path, query, header, and body

Example: `limit` query parameter for results pagination

```yaml
paths:
  /stations:
    get:
      summary: Get a list of train stations
      description: Returns a paginated and searchable list of all train stations.
      operationId: get-stations
      tags:
        - Stations
      parameters:
        - $ref: '#/components/parameters/page'
        - $ref: '#/components/parameters/limit'
        - name: coordinates
          in: query
          description: >
            The latitude and longitude of the user's location, to narrow down
            the search results to sites within a proximity of this location.
          required: false
          schema:
            type: string
          example: 52.5200,13.4050
        - name: search
          in: query
```

https://github.com/bump-sh-examples/train-travel-api/blob/main/openapi.yaml

RESPONSES IN OPENAPI

Define expected outputs for each operation

Includes status codes (e.g., 200 OK, 404 Not Found)

Example: 200 response for `GET /users` returns a list of users

```yaml
      responses:
        '201':
          description: Booking successful
          content:
            application/json:
              schema:
                allOf:
                  - $ref: '#/components/schemas/Booking'
                  - properties:
                      links:
                        $ref: '#/components/schemas/Links-Self'

              example:
                id: efdbb9d1-02c2-4bc3-afb7-6788d8782b1e
                trip_id: efdbb9d1-02c2-4bc3-afb7-6788d8782b1e
                passenger_name: John Doe
                has_bicycle: true
                has_dog: true
                links:
                  self: https://api.example.com/bookings/efdbb9d1-02c2-4bc3-afb7-6788d8782b1e
            application/xml:
              schema:
                allOf:
                  - $ref: '#/components/schemas/Booking'
                  - properties:
                      links:
                        $ref: '#/components/schemas/Links-Self'
        '400':
          $ref: '#/components/responses/BadRequest'
        '401':
          $ref: '#/components/responses/Unauthorized'
        '404':
          $ref: '#/components/responses/NotFound'
        '409':
          $ref: '#/components/responses/Conflict'
        '429':
          $ref: '#/components/responses/TooManyRequests'
        '500':
          $ref: '#/components/responses/InternalServerError'
```

https://github.com/bump-sh-examples/train-travel-api/blob/main/openapi.yaml

SECURITY DEFINITIONS IN OPENAPI

Define API authentication and authorization methods

Options include OAuth2, API Key, and more

Example: API key required for `/admin/` endpoints

```yaml
components:
  securitySchemes:
    oauth2Profiles:
      type: oauth2
      flows:
        clientCredentials:
          tokenUrl: https://learn.openapis.org/oauth/2.0/token
          scopes:
            board:read: Read the board
            board:write: Write to the board
        authorizationCode:
          authorizationUrl: https://learn.openapis.org/oauth/2.0/auth
          tokenUrl: https://learn.openapis.org/oauth/2.0/token
          scopes:
            board:read: Read the board
            board:write: Write to the board

```

```yaml
openapi: 3.1.0
info:
  title: Tic Tac Toe
  description: |
    This API allows writing down marks on a Tic Tac Toe board
    and requesting the state of the board or of individual squares.
  version: 1.0.0
security:
  - oauth2Profiles:
    - board:read
    - board:write
paths:
  /board:
    get:
      security:
        - oauth2Profiles: []

```

https://learn.openapis.org/specification/security.html Describing API Security

ADDITIONAL OPENAPI FEATURES

Authentication and rate limiting

Error handling and versioning

OVERVIEW OF SWAGGER TOOLS

Swagger UI - Interactive API documentation

Swagger Editor - Writing and validating specs

Swagger Codegen - Generates client/server code

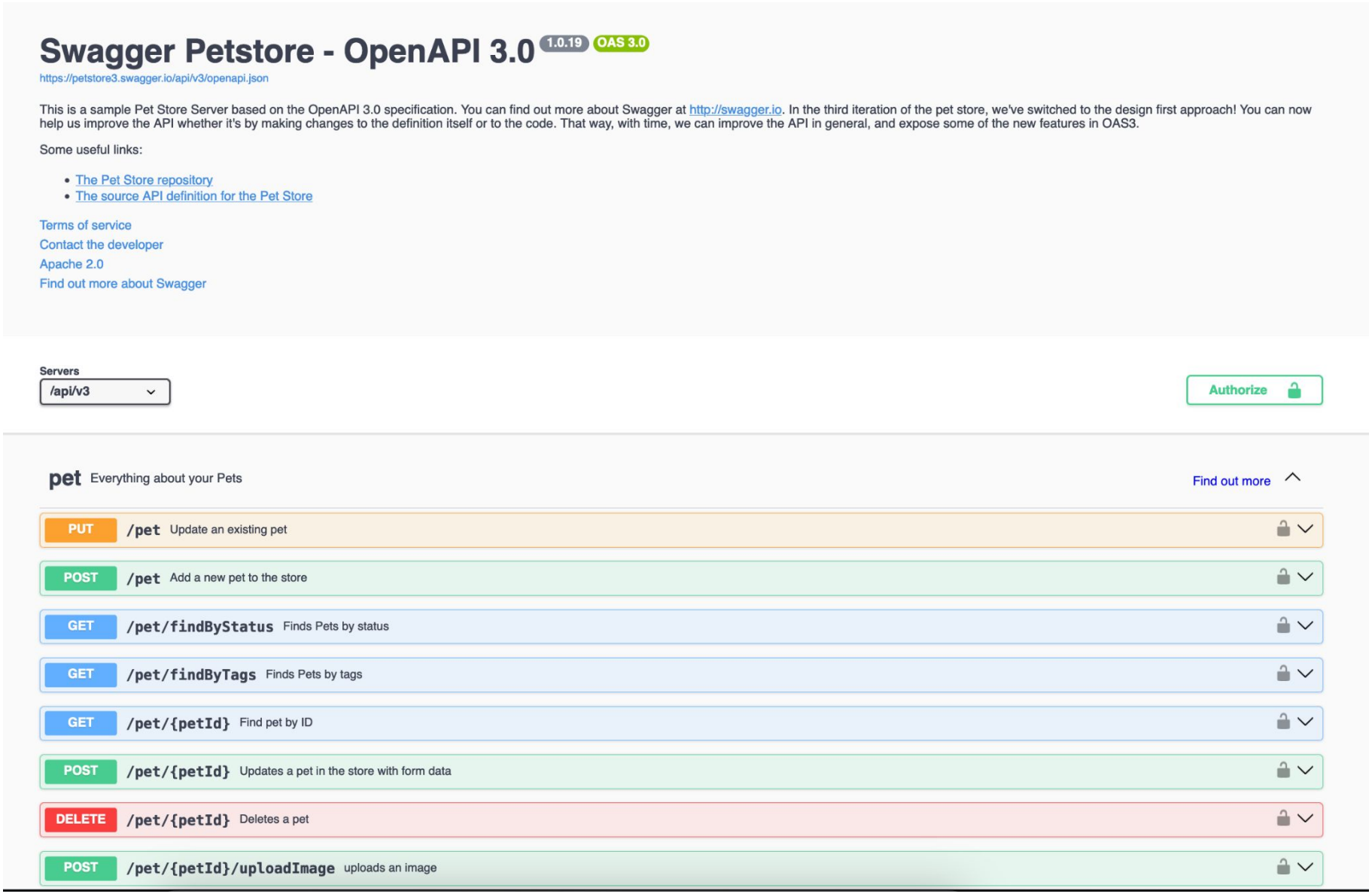
SwaggerHub - Team collaboration platform

SWAGGER UI

Web interface for viewing and testing APIs

Interactive documentation with try-it-out feature

Example: Test `GET /users` endpoint directly



https://petstore.swagger.io/

Demo time!

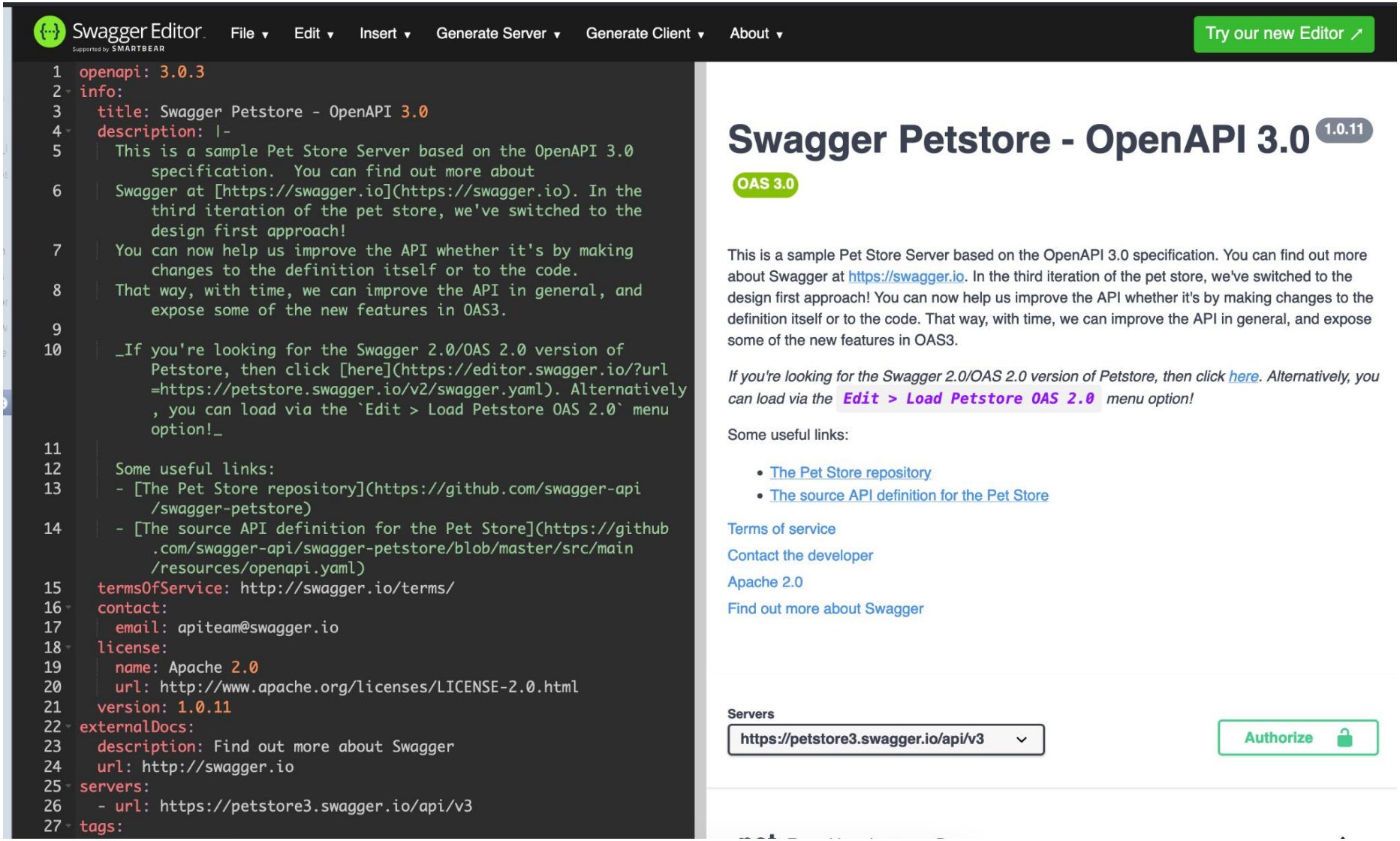
petstore.swagger.io

SWAGGER EDITOR

IDE-like environment for creating OpenAPI specs

Real-time validation and syntax highlighting

Preview structured documentation


Swagger Editor
Supported by SMARTBEAR
File
Edit
Insert
Generate Server
Generate Client
About
Try our new Editor ↗
openapi: 3.0.3
info:
title: Swagger Petstore - OpenAPI 3.0
description: |-
This is a sample Pet Store Server based on the OpenAPI 3.0 specification. You can find out more about
Swagger at https://swagger.io. In the third iteration of the pet store, we've switched to the design first approach!
You can now help us improve the API whether it's by making changes to the definition itself or to the code.
That way, with time, we can improve the API in general, and expose some of the new features in OAS3.
If you're looking for the Swagger 2.0/OAS 2.0 version of Petstore, then click [here](https://editor.swagger.io/?url=https://petstore.swagger.io/v2/swagger.yaml). Alternatively, you can load via the `Edit > Load Petstore OAS 2.0` menu option!
Some useful links:
- [The Pet Store repository](https://github.com/swagger-api/swagger-petstore)
- [The source API definition for the Pet Store](https://github.com/swagger-api/swagger-petstore/blob/master/src/main/resources/openapi.yaml)
termsOfService: http://swagger.io/terms/
contact:
email: apiteam@swagger.io
license:
name: Apache 2.0
url: http://www.apache.org/licenses/LICENSE-2.0.html
version: 1.0.11
externalDocs:
description: Find out more about Swagger
url: http://swagger.io
servers:
- url: https://petstore3.swagger.io/api/v3
tags:
Swagger Petstore - OpenAPI 3.0
1.0.11
OAS 3.0
This is a sample Pet Store Server based on the OpenAPI 3.0 specification. You can find out more about Swagger at https://swagger.io. In the third iteration of the pet store, we've switched to the design first approach! You can now help us improve the API whether it's by making changes to the definition itself or to the code. That way, with time, we can improve the API in general, and expose some of the new features in OAS3.
If you're looking for the Swagger 2.0/OAS 2.0 version of Petstore, then click here. Alternatively, you can load via the Edit > Load Petstore OAS 2.0 menu option!
Some useful links:
The Pet Store repository
The source API definition for the Pet Store
Terms of service
Contact the developer
Apache 2.0
Find out more about Swagger
Servers
https://petstore3.swagger.io/api/v3
Authorize

SWAGGER CODEGEN

Generates client SDKs, server stubs, and API documentation

Supports multiple languages (Java, Python, JavaScript, etc.)

Example: Generate a Python SDK for API interaction

SWAGGERHUB

Collaborative platform for API design and documentation

Version control, role-based access, CI/CD integration

Example: Sync with GitHub for automated updates

CREATING AN API SPEC WITH OPENAPI

Example: Todo API - basic info, server, and paths

Define operations for CRUD actions

YAML example for starting structure

```yaml

openapi: 3.0.3
info:
  title: Todo API
  description: A simple API to manage todo items.
  version: "1.0.0"
servers:
  - url: https://api.example.com/v1
    description: Production server
  - url: https://staging-api.example.com/v1
    description: Staging server
```

ADDING PARAMETERS TO THE API SPEC

Define query and path parameters

Example: Add `status` query parameter to filter todos

Set data type and default values

```yaml
/todos/{id}:
  get:
    summary: Get a specific todo
    description: Retrieve a specific todo by its ID.
    security:
      - ApiKeyAuth: []
    parameters:
      - name: id
        in: path
        required: true
        schema:
          type: string
    responses:
      '200':
        description: Todo found
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Todo'
      '404':
        description: Todo not found
  put:
    summary: Update a specific todo
    description: Update the details of an existing todo.
```

ADDING RESPONSES TO THE API SPEC

Define possible responses for each operation

Status codes and example data structure

Example: 200 and 404 responses for `GET /todos`

```yaml
post:
      summary: Create a new todo
      description: Add a new item to the todo list.
      security:
        - ApiKeyAuth: []
      requestBody:
        description: Todo object to add
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Todo'
      responses:
        '201':
          description: Todo created successfully
        '400':
          description: Invalid input
```

ADDING SECURITY TO THE API SPEC

Define security schemes like API key or OAuth

Apply globally or to specific paths

Example: API key required for `/admin` routes

```yaml
paths:
  /todos:
    get:
      summary: Get all todos
      description: Retrieve a list of all todo items.
      security:
        - ApiKeyAuth: []
      responses:
        '200':
          description: A list of todos
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/Todo'
        '500':
          description: Server error
```

BEST PRACTICES IN OPENAPI AND SWAGGER USAGE

Detailed documentation for parameters and responses

Keep specs updated to reflect changes

COMMON PITFALLS TO AVOID

Incomplete or outdated specs

Poor naming conventions

Lack of error handling and security details

PERSONAL EXPERIENCE

EXTRAS

Integrate Postman with OpenAPI

You can import your existing OpenAPI 3.0 and 3.1 definitions (OpenAPI Specification) into Postman. Postman supports both YAML and JSON formats.

New features in OpenAPI and Swagger tools

Trends: adoption growth

Q&A

Thank you for your time, question, and attention.