WOMEN WHO CODE
/london

RESTful API 101

Madhura Chaganty

Lead Software Engineer

e-mobility solutions @Paythru

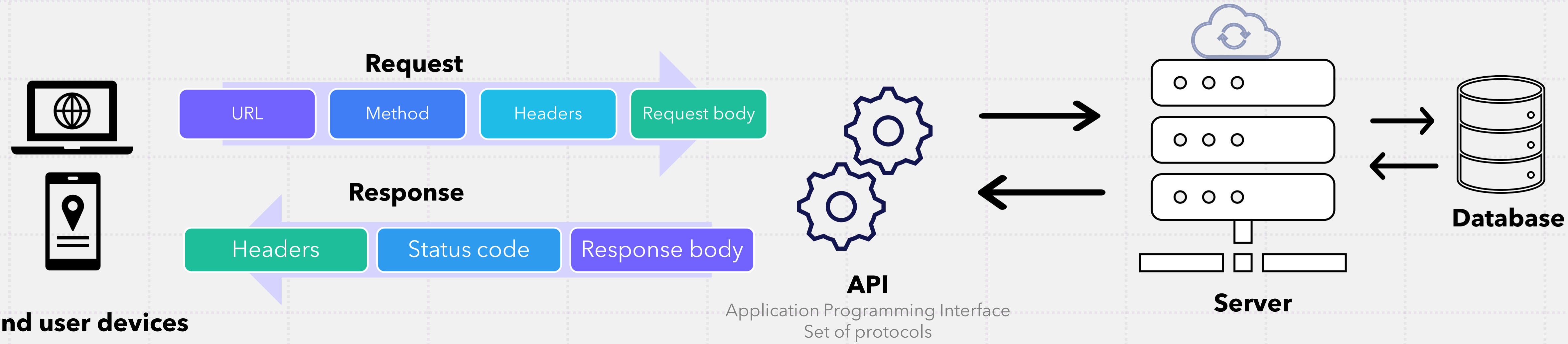
Key topics

In terms of
Mentorship REST **API**

Target audience
Individuals with minimal to no prior understanding of REST concepts

- Fundamentals of API
- Anatomy of RESTful API
- Maturity model
- Authentication and authorization
- API versioning
- API documentation
- API testing
- Logging and monitoring
- Key contributors for design and development
- Total cost of ownership
- Q&A

API – Application Programming Interface



End user devices

Request

URL | Method | Headers | Request body

Response

Headers | Status code | Response body

API

Application Programming Interface
Set of protocols

Server

Database

Request URL

https://api.example.com/users?type=adhoc&offset=0&limit=100

Base URL: https://api.example.com/

Path parameter: users
Part of the URL
Mandatory

Query parameters: ?type=adhoc&offset=0&limit=100
Filtering, pagination
Mandatory or Optional

SOAP
XML based, strict,
favored in enterprises

REST
Scalable using HTTP
methods

Covered

GraphQL
Query language for API

gRPC
High performance
framework

WebSockets
Bi-directional
Real-time

Webhook
Event driven,
HTTP callbacks

MQTT
Lightweight messaging
protocol

AMQP
Open messaging
protocol

What is not covered

- Language specific implementations
- Other API Architectural styles
- Authentication protocols like Oauth, JWT in depth
- And of course, Microservices!

REST - REpresentational State Transfer

REST - Architectural constraints

Entity : Data object in the application

Resource : Abstracted version of entity

Representation : Encoded resource in JSON, XML

Constraints

Uniform HTTP interface : HTTP CRUD operations on resources

Client-Server architecture : Separation of concerns

Stateless : No session information on the server

Cacheable : Ability to cache responses at client side

Layered system : Loose coupling and independence of requests

Code on demand : Optional

Entity
mentors
(data object)

Resource
mentors
(abstracted)

Representation
mentors
(json representation)

RESTful APIs conform to REST architectural style

URI constraints

Resource identification in requests

Resource manipulation through representations

Self-descriptive messages

HATEOAS (Hypermedia as the Engine of Application State)
inclusion of hypermedia links in API responses

Request GET /mentors/M123
Response

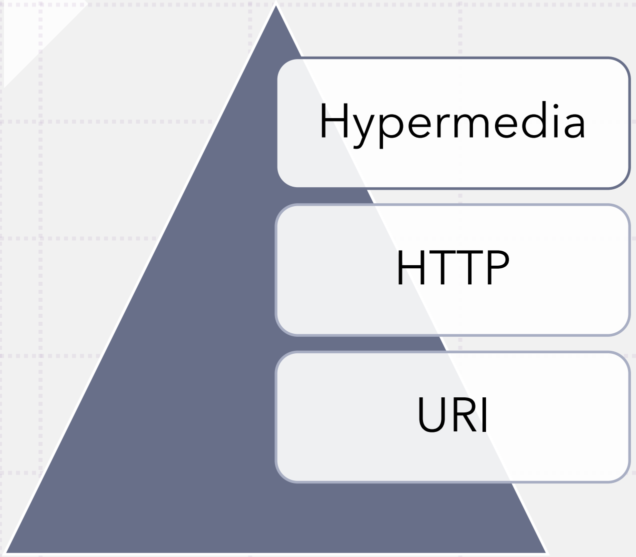
```
{
  "id": M123,
  "name": "John Doe",
  "email": "john.doe@example.com",
  “type": “long_term_mentor”
  "links": [
      { "rel": "self", "href": "/mentors/M123" },
      { "rel": “prevPage", "href":
"/mentors/M123?offset=0&limit=100"},
      { "rel": “nextPage", "href":
"/mentors/M123?offset=201&limit=100"}
  ] }
```

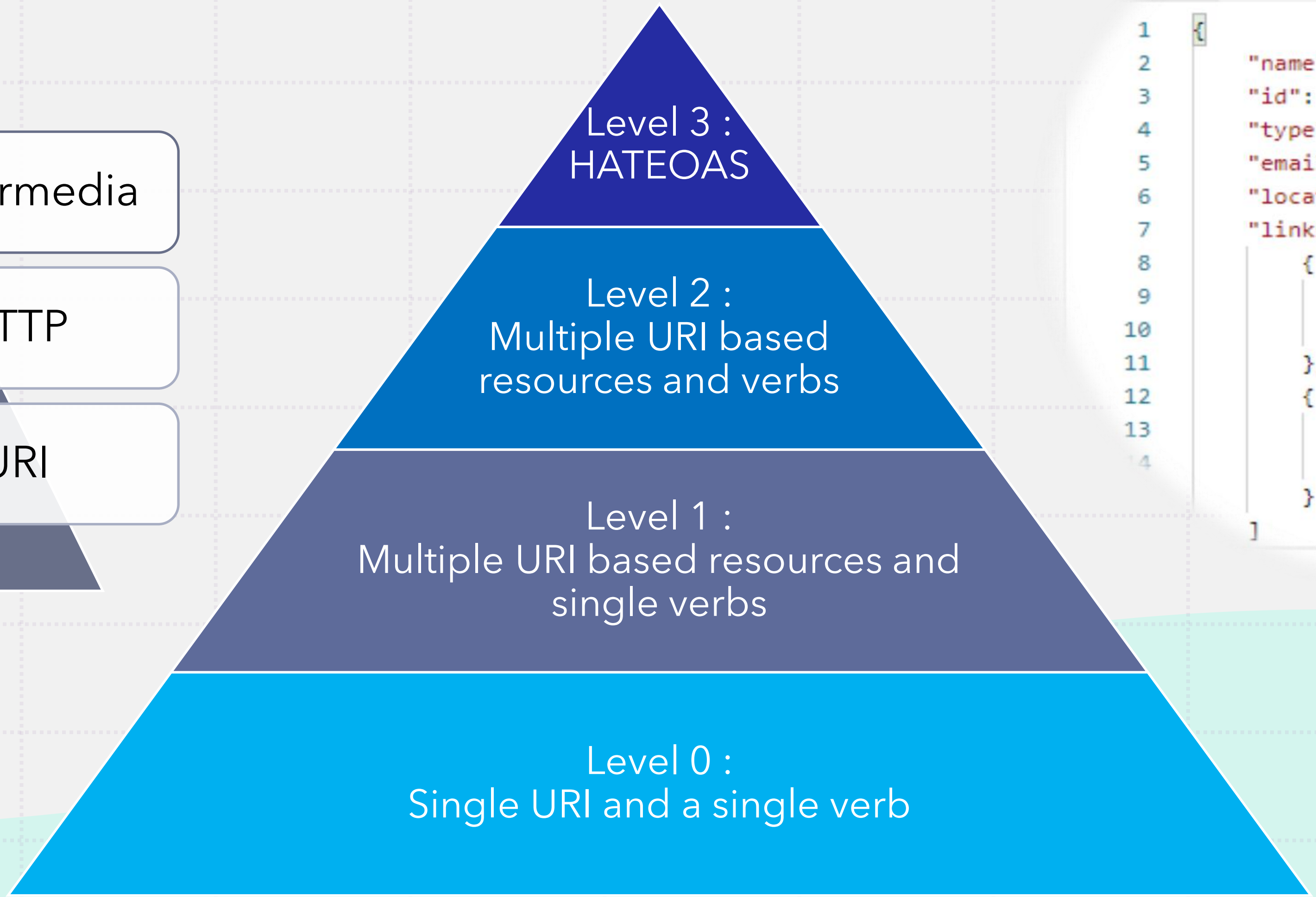
https://api.example.com/mentors?type=adhoc_mentor&offset=0&limit=100

Resources	URI	Response
mentors	GET /mentors	Collection
mentees	GET /mentee/[id]	Single
mentorship_programs	GET /mentorship_programs/[id]/schedules	Sub-collection within single resource
mentorship_sessions	GET /mentorship_sessions/[id]/topics/[tid]	Single resource within sub-collection
mentorship_resources	GET /mentorship_resources?type=articles	Filter mentorship resources by type
feedbacks	GET /feedbacks?offset=0&limit=100	Pagination using offset and limit

Richardson Maturity Model

Determines how much the web services are REST compliant



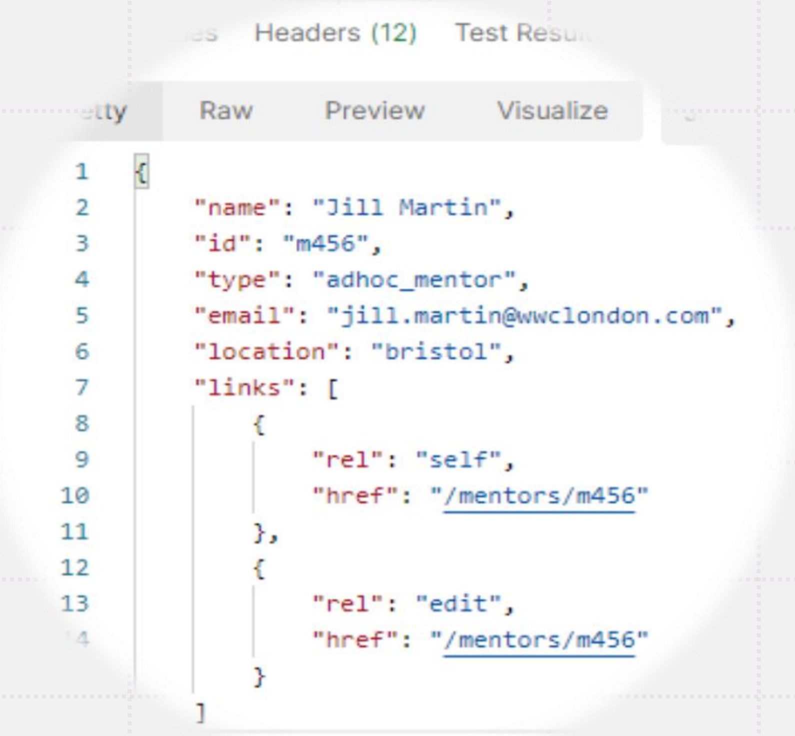


Level 3 :
HATEOAS

Level 2 :
Multiple URI based resources and verbs

Level 1 :
Multiple URI based resources and single verbs

Level 0 :
Single URI and a single verb



```
{
    "name": "Jill Martin",
    "id": "m456",
    "type": "adhoc_mentor",
    "email": "jill.martin@wwclondon.com",
    "location": "bristol",
    "links": [
        {
            "rel": "self",
            "href": "/mentors/m456"
        },
        {
            "rel": "edit",
            "href": "/mentors/m456"
        }
    ]
```

Worth reading - https://martinfowler.com/articles/richardsonMaturityModel.html

HTTP methods and status codes

Safe - GET, HEAD, OPTIONS, TRACE

Operations that do not modify resources

Idempotent - GET, HEAD, OPTIONS, TRACE, PUT, PATCH, DELETE

Operations that produce the same results if executed once or multiple times

POST	GET	PUT	PATCH	DELETE
Create	Read	Update/replace	Partial update/modify	Delete
201 (Created)	200 OK 204 No content 404 Not found	200 OK 204 No content 404 Not found 405 Method not allowed	200 OK 204 No content 404 Not found 405 Method not allowed	200 OK 404 Not found 405 Method not allowed
	☑ Safe			
	☑ Idempotent	☑ Idempotent	☑ Idempotent	☑ Idempotent

HTTP Status Codes

1xx : Informational
- 100 Continue
- 101 Switching protocol
- 102 Processing

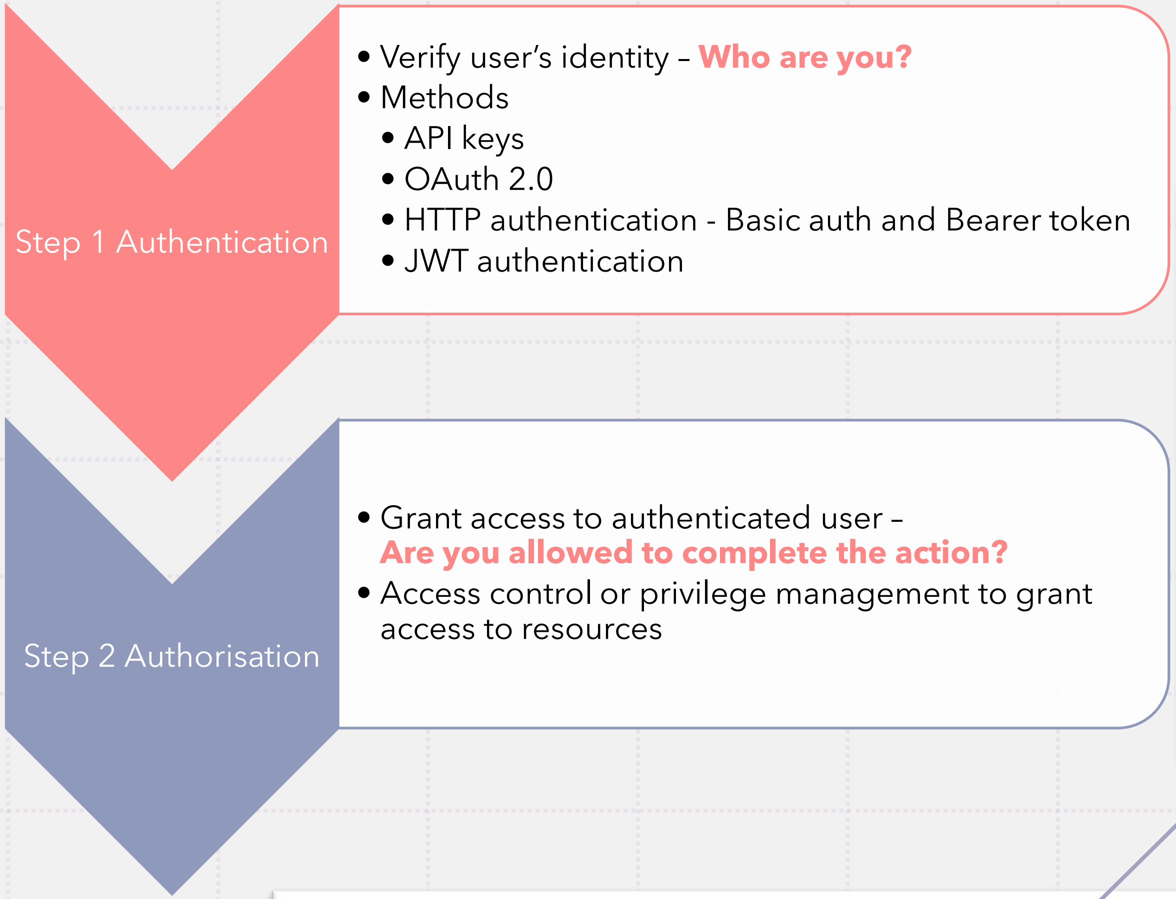
2xx : Success
- 200 OK
- 202 Accepted

3xx : Redirection
- 301 Moved permanently
- 302 Found (New location)

4xx: Client Error
- 400 Bad request
- 401 Unauthorized
- 403 Forbidden
- 404 Not found
- 405 Method not allowed
- 409 Conflict

5xx: Server Error
- 500 Internal server error
- 501 Not implemented

Authentication & Authorisation

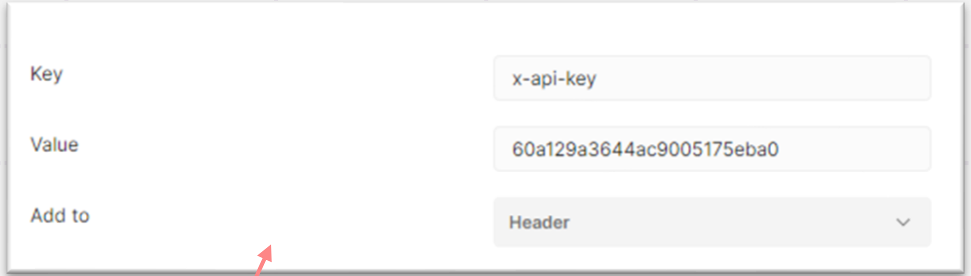


Step 1 Authentication

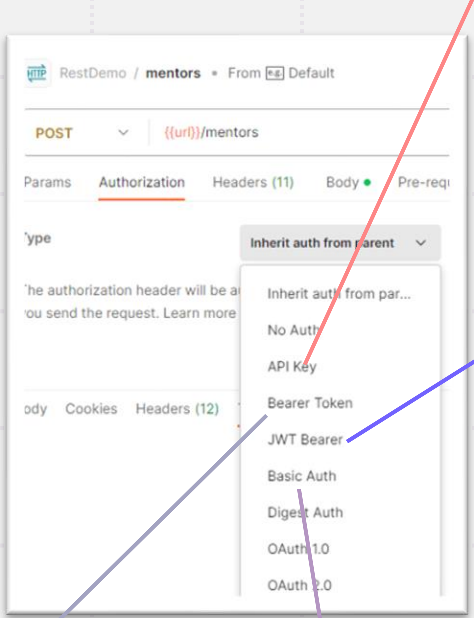
- Verify user's identity – **Who are you?**
- Methods
  - API keys
  - OAuth 2.0
  - HTTP authentication - Basic auth and Bearer token
  - JWT authentication

Step 2 Authorisation

- Grant access to authenticated user – **Are you allowed to complete the action?**
- Access control or privilege management to grant access to resources

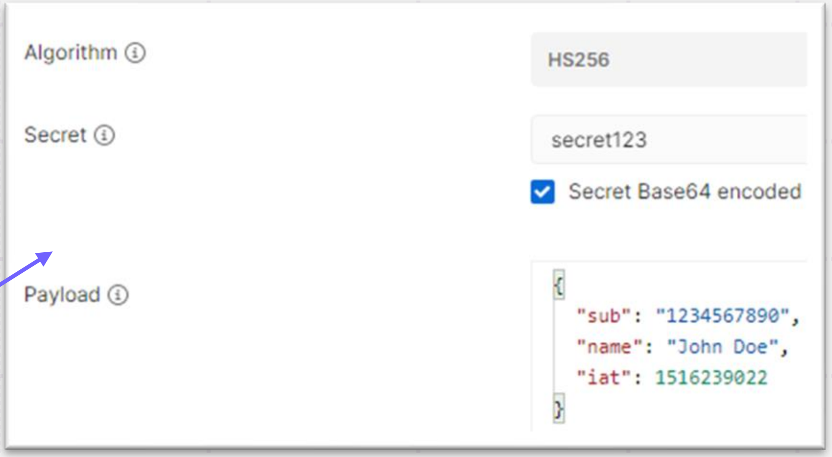


Key	x-api-key
Value	60a129a3644ac9005175eba0
Add to	Header



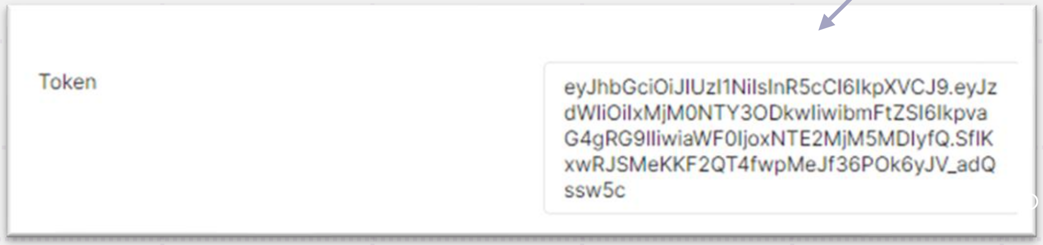
Type: Inherit auth from parent

'he authorization header will be a... 'ou send the request. Learn more



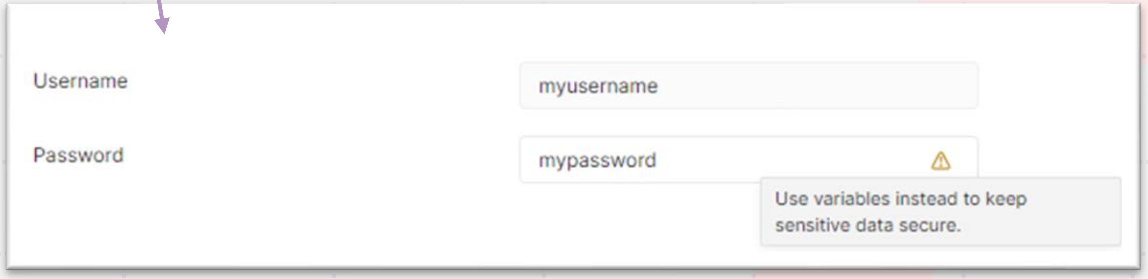
Algorithm ⓘ	HS256
Secret ⓘ	secret123
	☑ Secret Base64 encoded
Payload ⓘ	

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```



Token

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJz
dWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflK
xwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQ
ssw5c
```

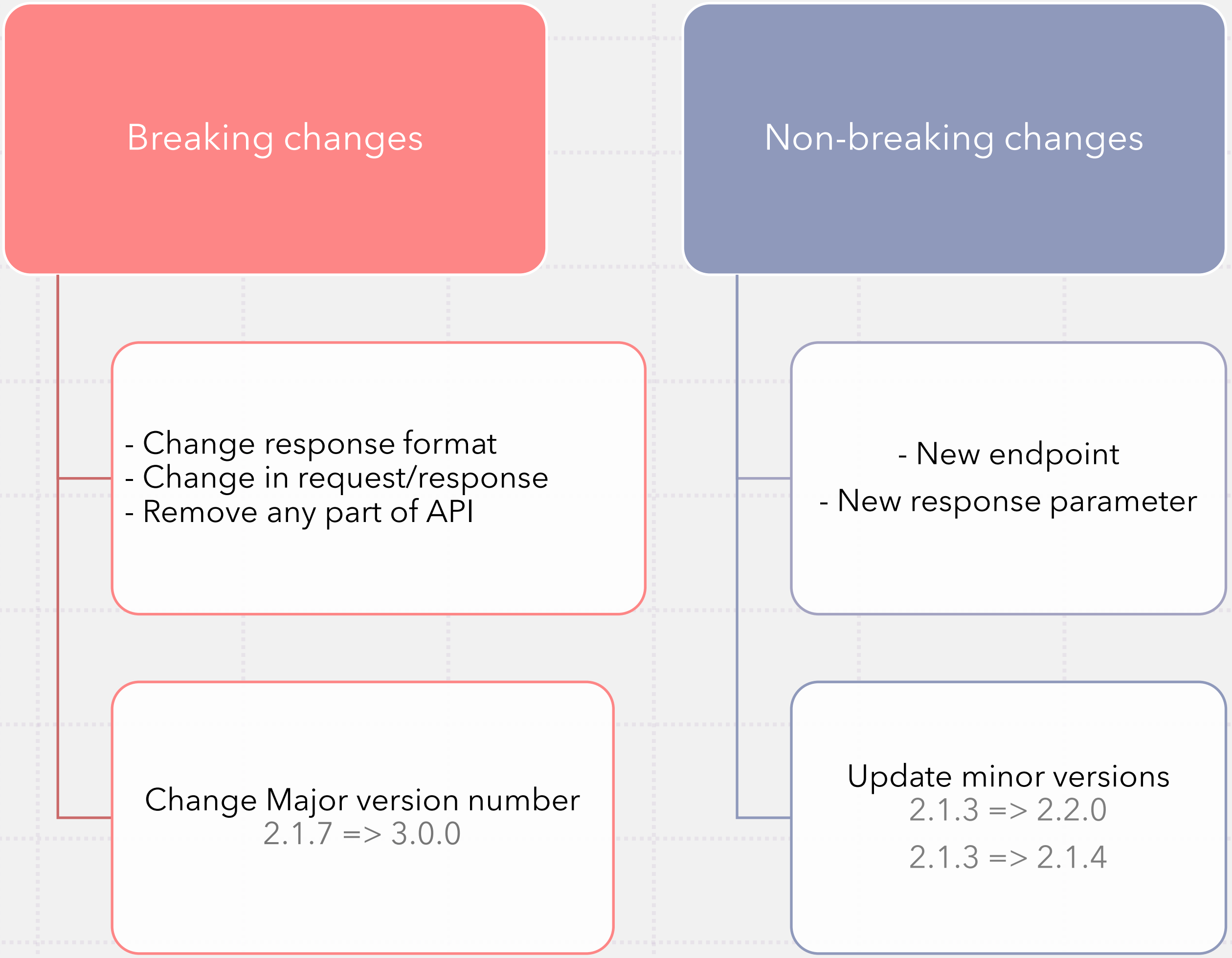


Username	myusername
Password	mypassword ⚠

Use variables instead to keep sensitive data secure.

API Versioning

MAJOR.MINOR.PATCH - v2.1.3



Breaking changes

- Change response format
- Change in request/response
- Remove any part of API

Change Major version number
2.1.7 => 3.0.0

Non-breaking changes

- New endpoint
- New response parameter

Update minor versions
2.1.3 => 2.2.0
2.1.3 => 2.1.4

URI Versioning
https://api.example.com/v1/resource

- Can be handled by routing

Query parameter
https://www.test.com/api/resource?version=1

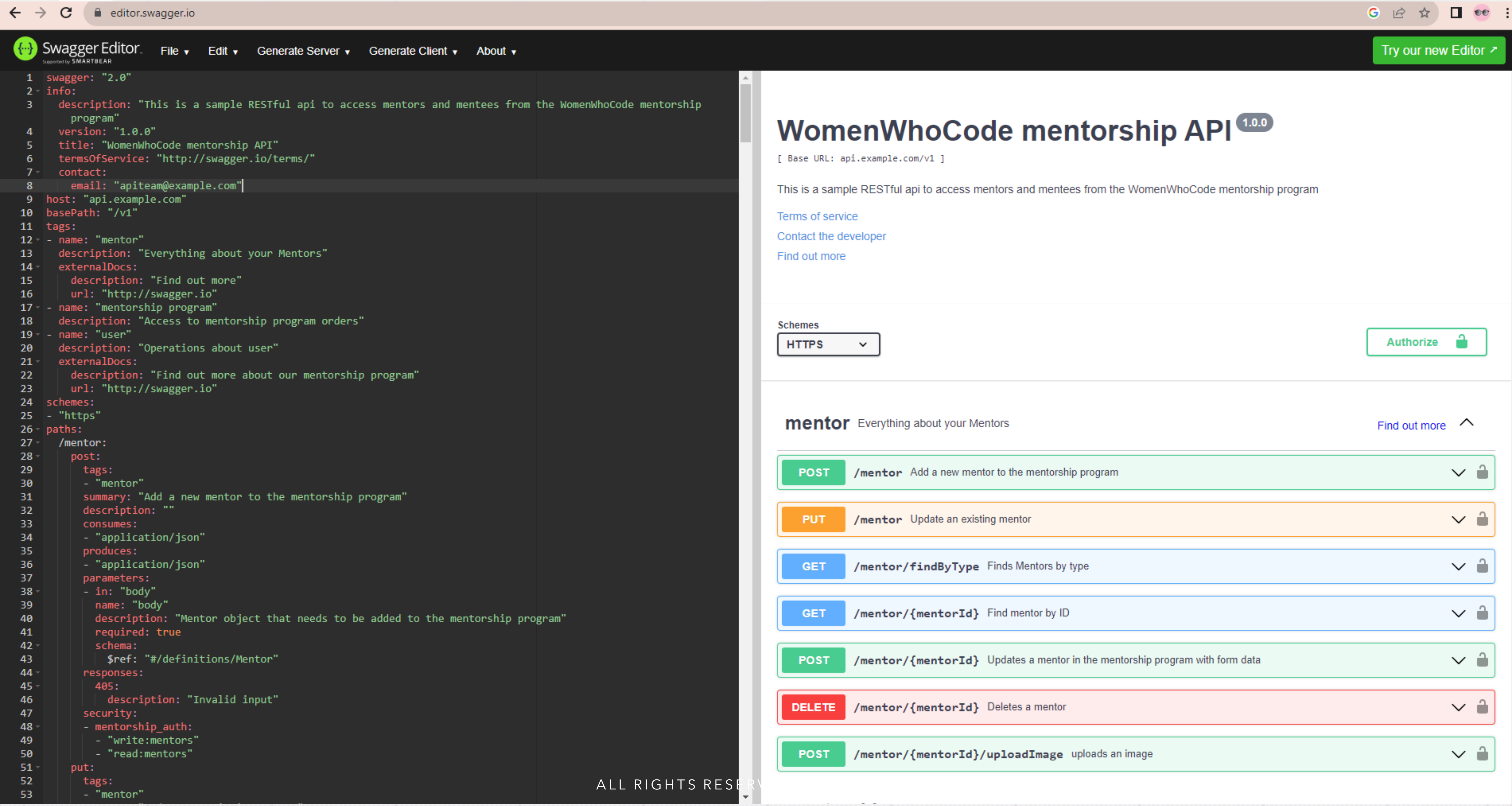
- Easy to switch to newer version

Custom Request Headers
Accept-version: v1

- API controller responsible for version control

Accept header
Accept: application/vnd.example+json:version=1.0

- API controller responsible for version control



```yaml
swagger: "2.0"
info:
  description: "This is a sample RESTful api to access mentors and mentees from the WomenWhoCode mentorship program"
  version: "1.0.0"
  title: "WomenWhoCode mentorship API"
  termsOfService: "http://swagger.io/terms/"
  contact:
    email: "apiteam@example.com"
host: "api.example.com"
basePath: "/v1"
tags:
- name: "mentor"
  description: "Everything about your Mentors"
  externalDocs:
    description: "Find out more"
    url: "http://swagger.io"
- name: "mentorship program"
  description: "Access to mentorship program orders"
- name: "user"
  description: "Operations about user"
  externalDocs:
    description: "Find out more about our mentorship program"
    url: "http://swagger.io"
schemes:
- "https"
paths:
  /mentor:
    post:
      tags:
      - "mentor"
      summary: "Add a new mentor to the mentorship program"
      description: ""
      consumes:
      - "application/json"
      produces:
      - "application/json"
      parameters:
      - in: "body"
        name: "body"
        description: "Mentor object that needs to be added to the mentorship program"
        required: true
        schema:
          $ref: "#/definitions/Mentor"
      responses:
        405:
          description: "Invalid input"
      security:
      - mentorship_auth:
        - "write:mentors"
        - "read:mentors"
    put:
      tags:
      - "mentor"
```

WomenWhoCode mentorship API 1.0.0

[Base URL: api.example.com/v1]

This is a sample RESTful api to access mentors and mentees from the WomenWhoCode mentorship program

Terms of service

Contact the developer

Find out more

Schemes

HTTPS

Authorize

mentor Everything about your Mentors — Find out more

POST /mentor Add a new mentor to the mentorship program

PUT /mentor Update an existing mentor

GET /mentor/findByType Finds Mentors by type

GET /mentor/{mentorId} Find mentor by ID

POST /mentor/{mentorId} Updates a mentor in the mentorship program with form data

DELETE /mentor/{mentorId} Deletes a mentor

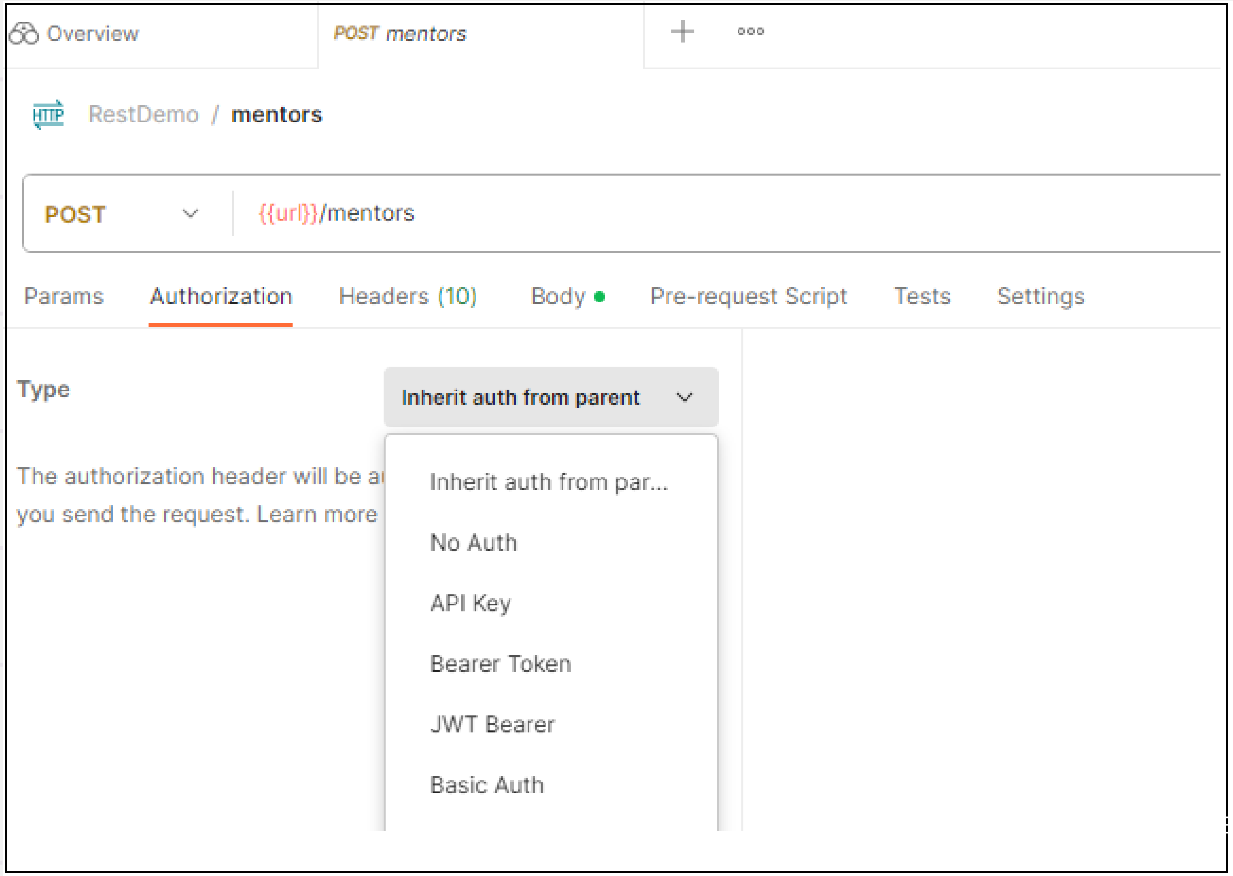
POST /mentor/{mentorId}/uploadImage uploads an image

Testing APIs

Key areas to test
Authentication and Authorization
Data validation
Security vulnerabilities
Error handling and responses
HTTP methods and status codes

Tools
Postman
Swagger (OpenAPI)
Insomnia *
Rest Assured * Java based

WOMEN WHO CODE /london
Postman Study Group
• Presentation covering basic topics
• Hands-On Activities and Exercises
• Study Materials, Resources
• Regular Virtual Catch-Ups
Online
Starts in September
Sign up now

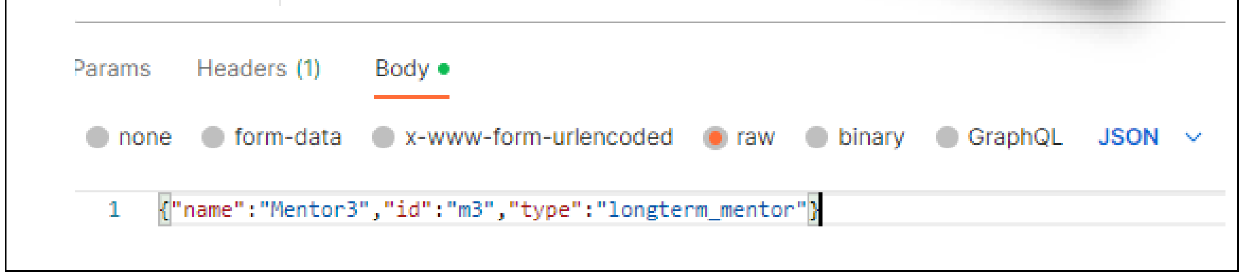


POST {{url}}/mentors

Type: Inherit auth from parent

The authorization header will be a
you send the request. Learn more

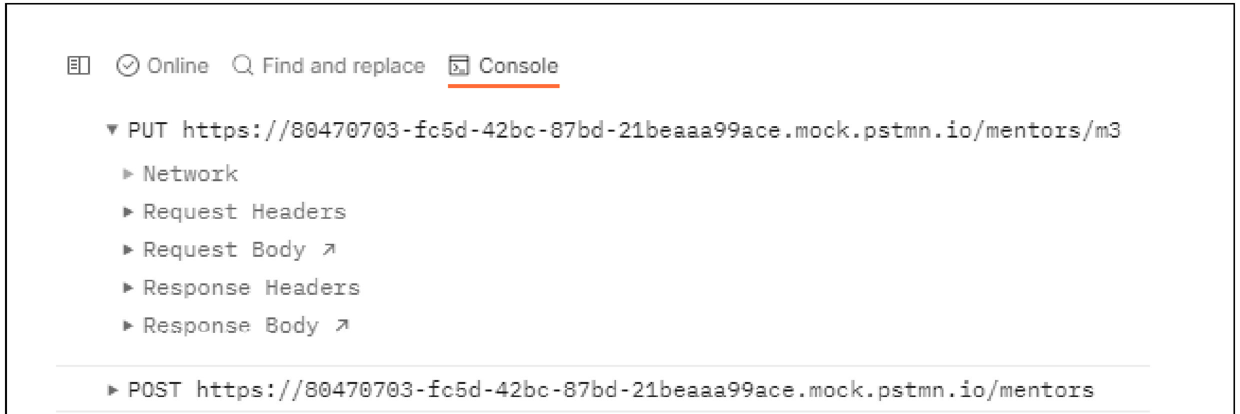
- Inherit auth from par...
- No Auth
- API Key
- Bearer Token
- JWT Bearer
- Basic Auth



POST {{url}}/mentors

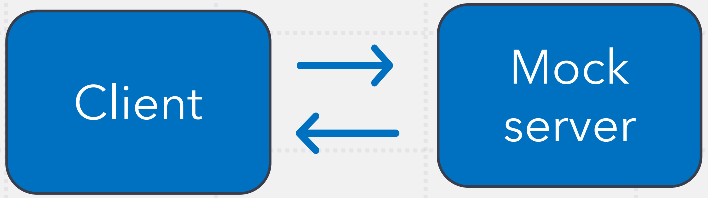
○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{"name":"Mentor3","id":"m3","type":"longterm_mentor"}
```



```
▾ PUT https://80470703-fc5d-42bc-87bd-21beaaa99ace.mock.pstmn.io/mentors/m3
  ▸ Network
  ▸ Request Headers
  ▸ Request Body ↗
  ▸ Response Headers
  ▸ Response Body ↗
▸ POST https://80470703-fc5d-42bc-87bd-21beaaa99ace.mock.pstmn.io/mentors
```

Mock APIs

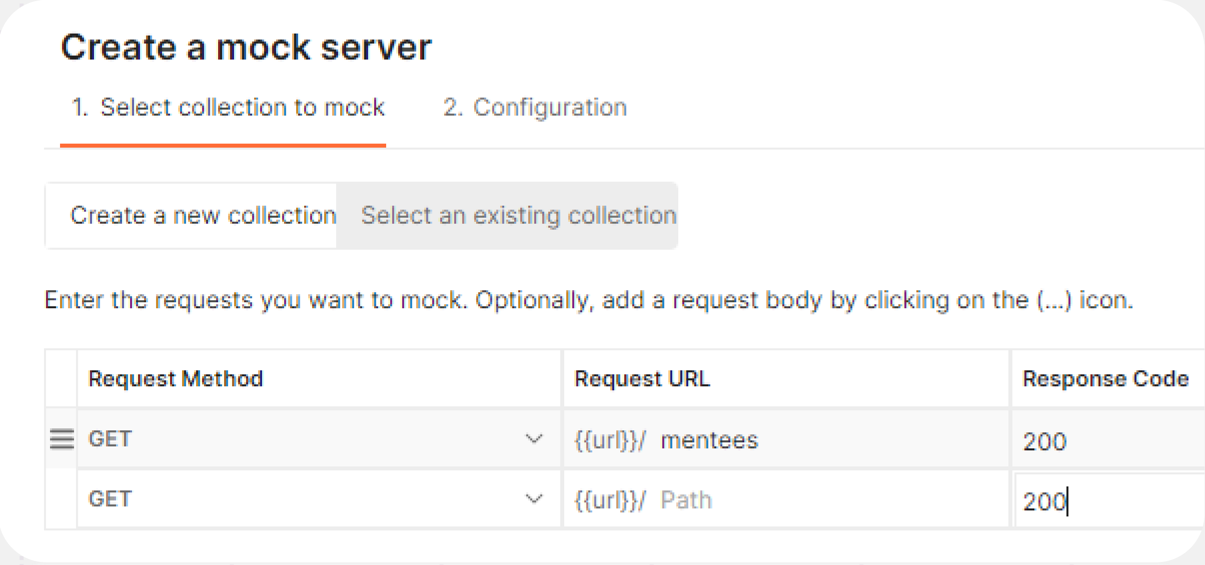


Advantages

- ✓ Parallel development of frontend and backend
- ✓ Isolation from external dependencies
- ✓ Cost efficiency as actual calls are not made to the API endpoints
- ✓ Easy to simulate test scenarios
- ✓ Early and easier onboarding of the consumers

Available options

- ➤ Postman mock server
- ➤ Mockaroo
  https://www.mockaroo.com/apis
- ➤ SwaggerHub
  https://support.smartbear.com/swaggerhub/docs/index.html



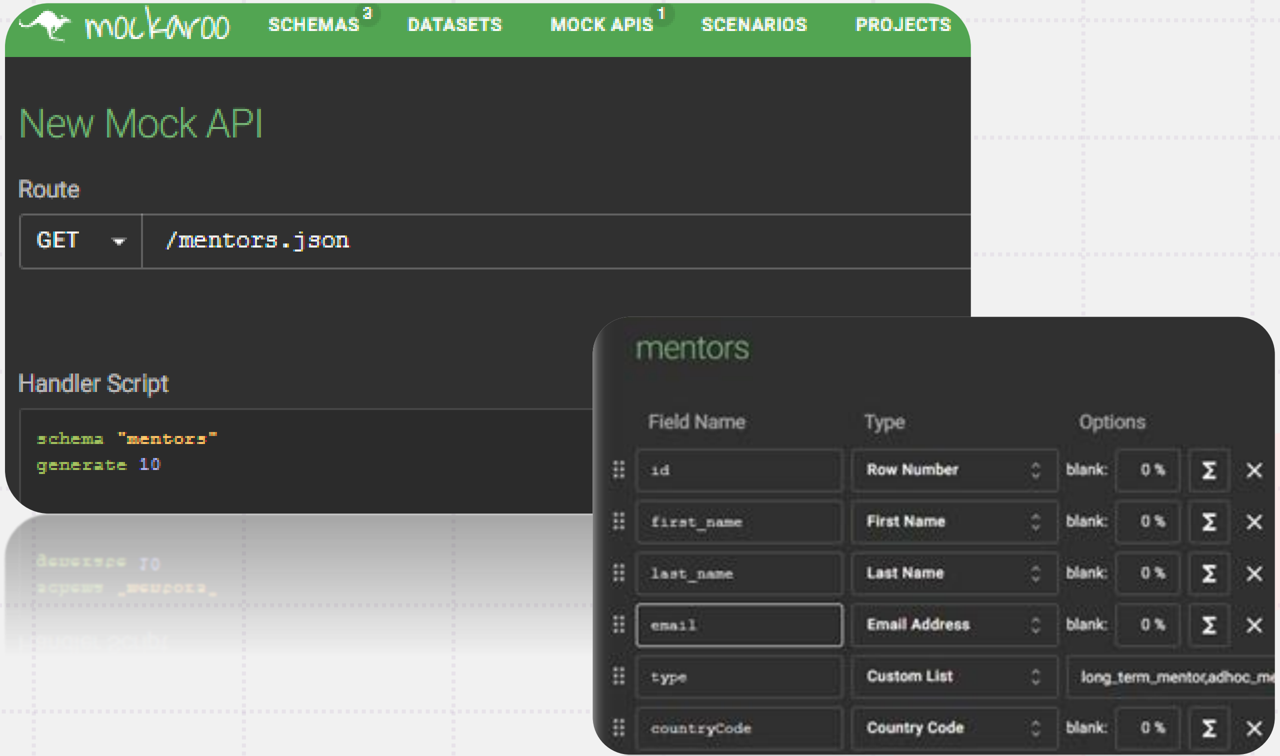
Create a mock server

1. Select collection to mock   2. Configuration

Create a new collection | Select an existing collection

Enter the requests you want to mock. Optionally, add a request body by clicking on the (...) icon.

Request Method	Request URL	Response Code
GET	{{url}}/ mentees	200
GET	{{url}}/ Path	200



New Mock API

Route

GET /mentors.json

Handler Script

```
schema "mentors"
generate 10
```

mentors

Field Name	Type	Options
id	Row Number	blank: 0 %
first_name	First Name	blank: 0 %
last_name	Last Name	blank: 0 %
email	Email Address	blank: 0 %
type	Custom List	long_term_mentor,adhoc_m
countryCode	Country Code	blank: 0 %

** the registered trademarks and logos of third parties are the property of their respective owners

Logging and monitoring

Logging

Centralised logging

Built in logging and monitoring feature
Cloud monitoring services
AWS CloudWatch, Google Cloud Monitoring and Azure monitor

Log libraries

Capture relevant logs

Implement log levels

INFO, DEBUG, WARNING, ERROR

Structured logging

JSON or key-value pairs

Contextual information

Relevant Context - timestamps, request urls

Monitoring

Performance Monitoring Tools

Utilize tools like New Relic such as response times, error rates, and resource usage

Alerting

Set up alerts based on predefined thresholds.

Dashboards

Create customized dashboards using tools like Grafana to visualize important metrics

Application Insights

Platforms like AWS, Azure Application Insights or Google Cloud Monitoring offer built-in monitoring capabilities, simplifying integration with cloud-based APIs

Error tracking tools

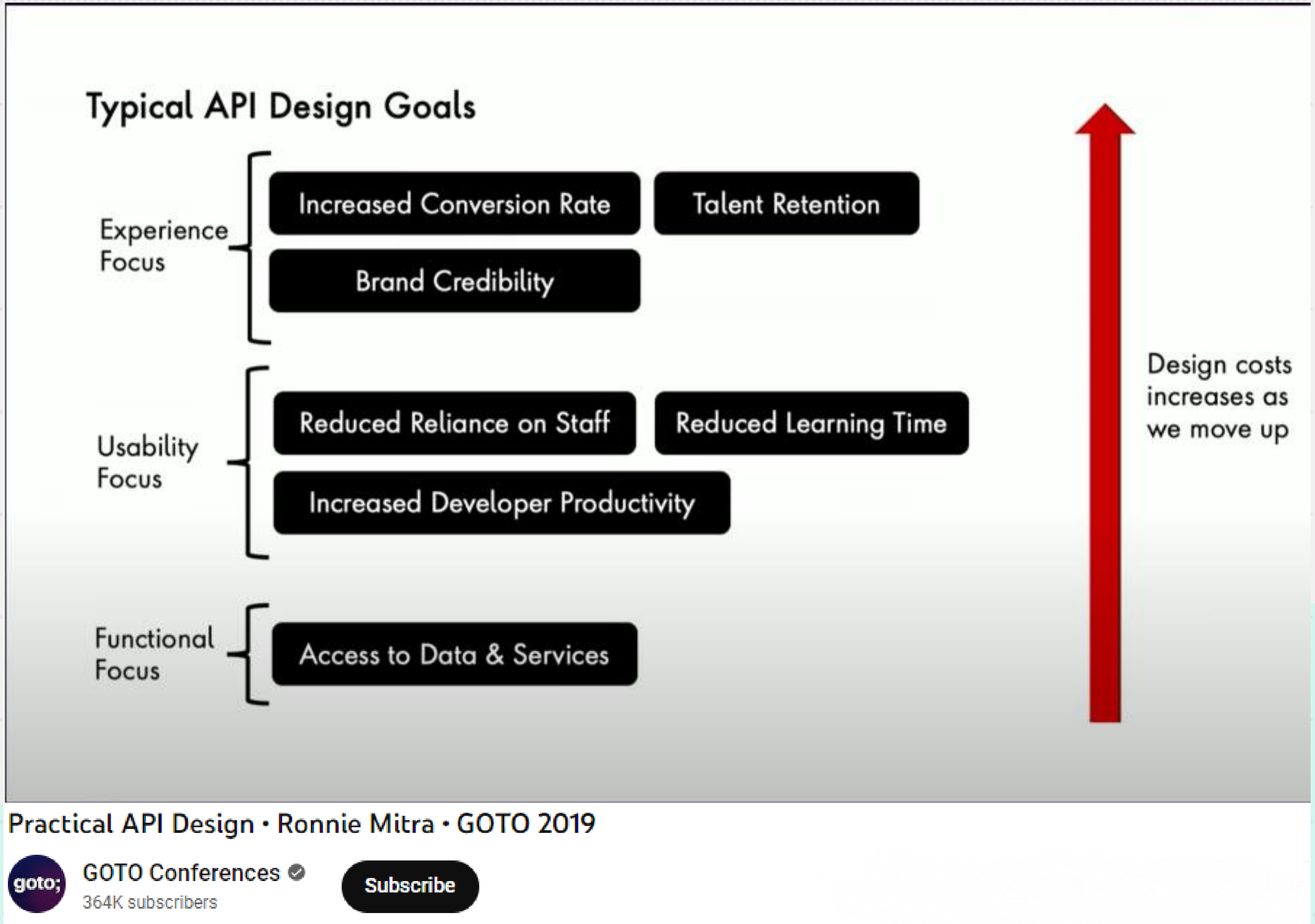
Monitor and capture API errors with tools like Sentry

Key contributors to API design and development

Requirements	Design	Development	Deployment	Documentation	Utilisation
Product managers Business analysts End users	API Architect API Product owner	API Developers	DevOps Team	API Developers Documentation Team	End users and consumers

Total cost of ownership

Initial development costs + ongoing expenses for operating, maintaining, and evolving the API over its lifecycle.



Typical API Design Goals

Experience Focus
- Increased Conversion Rate
- Talent Retention
- Brand Credibility

Usability Focus
- Reduced Reliance on Staff
- Reduced Learning Time
- Increased Developer Productivity

Functional Focus
- Access to Data & Services

Design costs increases as we move up

Practical API Design • Ronnie Mitra • GOTO 2019

Worth watching - Practical API Design • Ronnie Mitra • GOTO 2019
https://youtu.be/272ZZ53HS_4

Thank you for listening!