Introduction to API

Webscraping & API

Sarra Ben Yahia | José Ángel García Sánchez

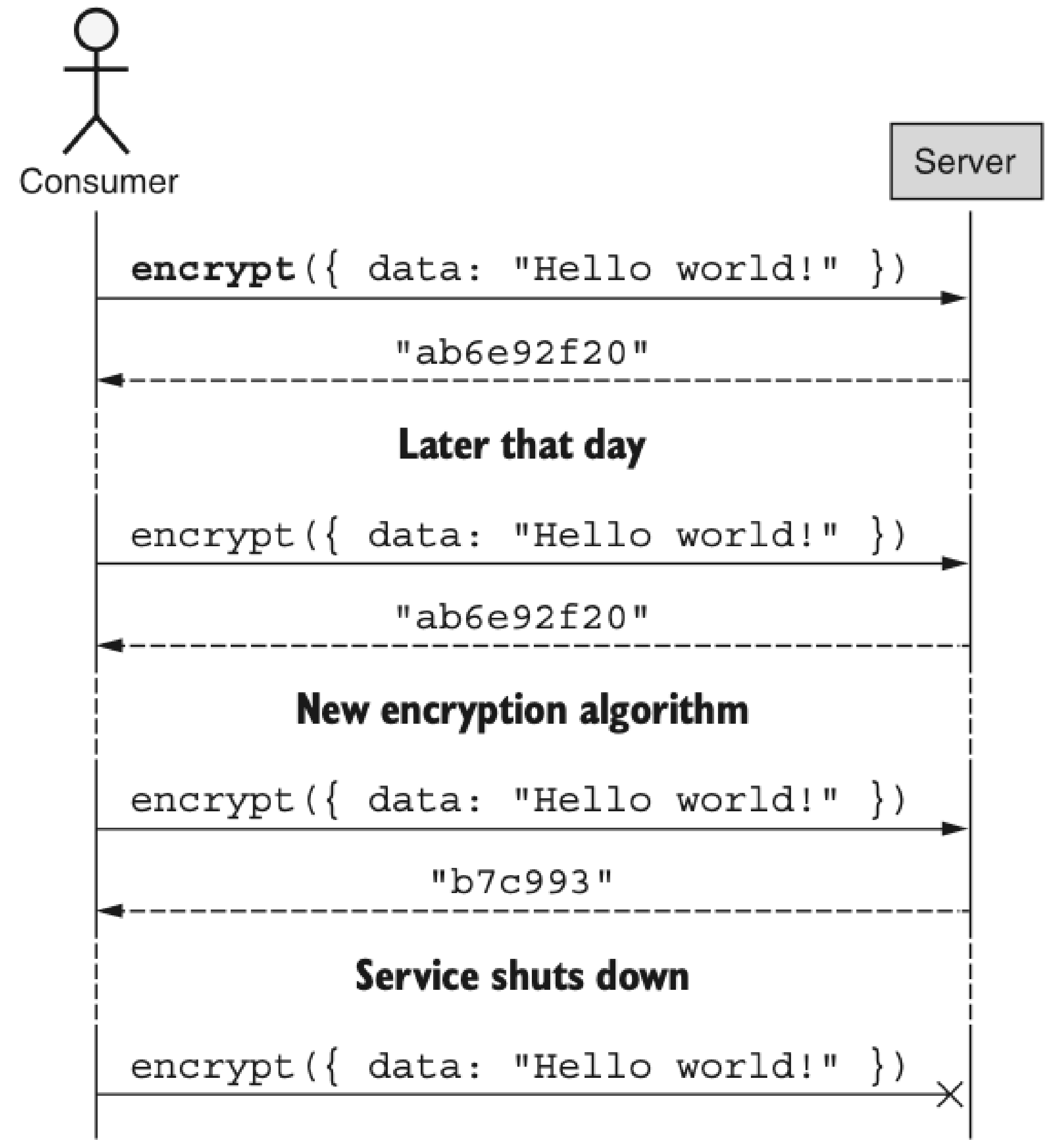
What’s an API

Definition

An API (Application Programming Interface) is a set of rules and protocols that allows different software applications to communicate with each other, enabling them to request and exchange data or functionality in a standardized manner.

Today, millions of APIs are available online, with websites like Reddit, X (formerly Twitter), and Facebook offering access to certain data through their APIs

Example


Consumer
Server
encrypt({ data: "Hello world!" })
"ab6e92f20"
Later that day
encrypt({ data: "Hello world!" })
"ab6e92f20"
New encryption algorithm
encrypt({ data: "Hello world!" })
"b7c993"
Service shuts down
encrypt({ data: "Hello world!" })

API in Python

Several Python frameworks to build API

FastAPI

Flask

Django

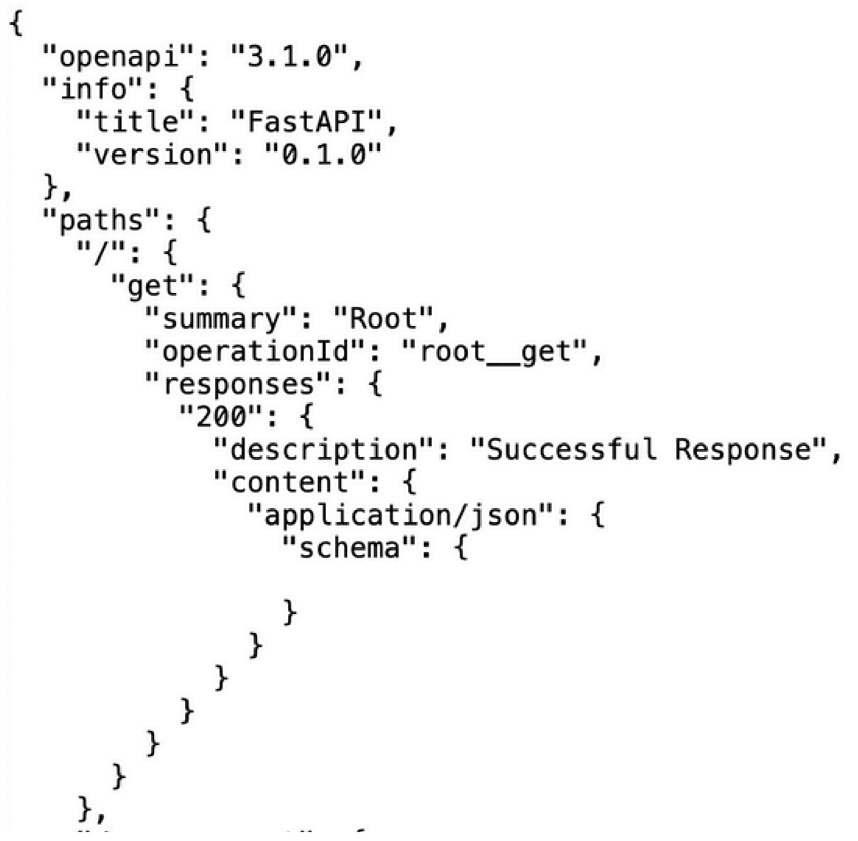
OpenAPI

OpenAPI document defines

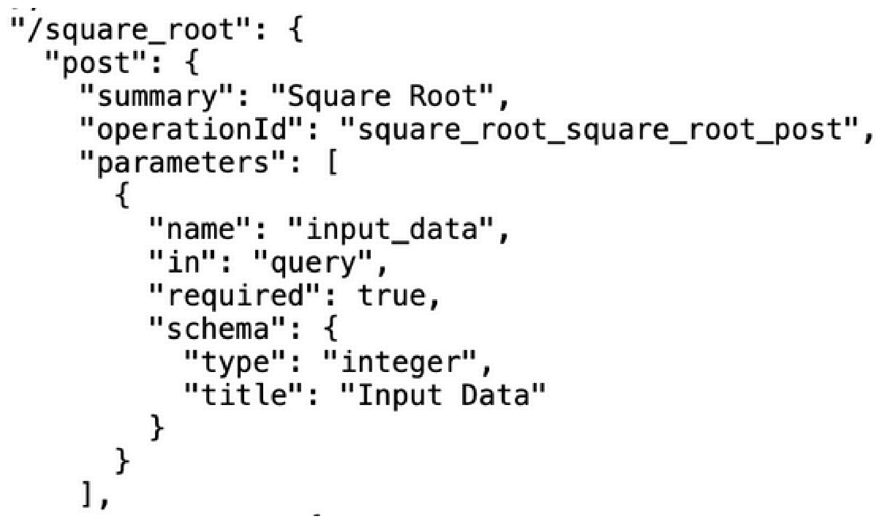
- Schema
- Data Format
- Data Type
- Path
- Object
- Others

Note: FastAPI generates the OpenAPI schema automatically helping to be RESTful compliant so individuals can use easily the API

OpenAPI



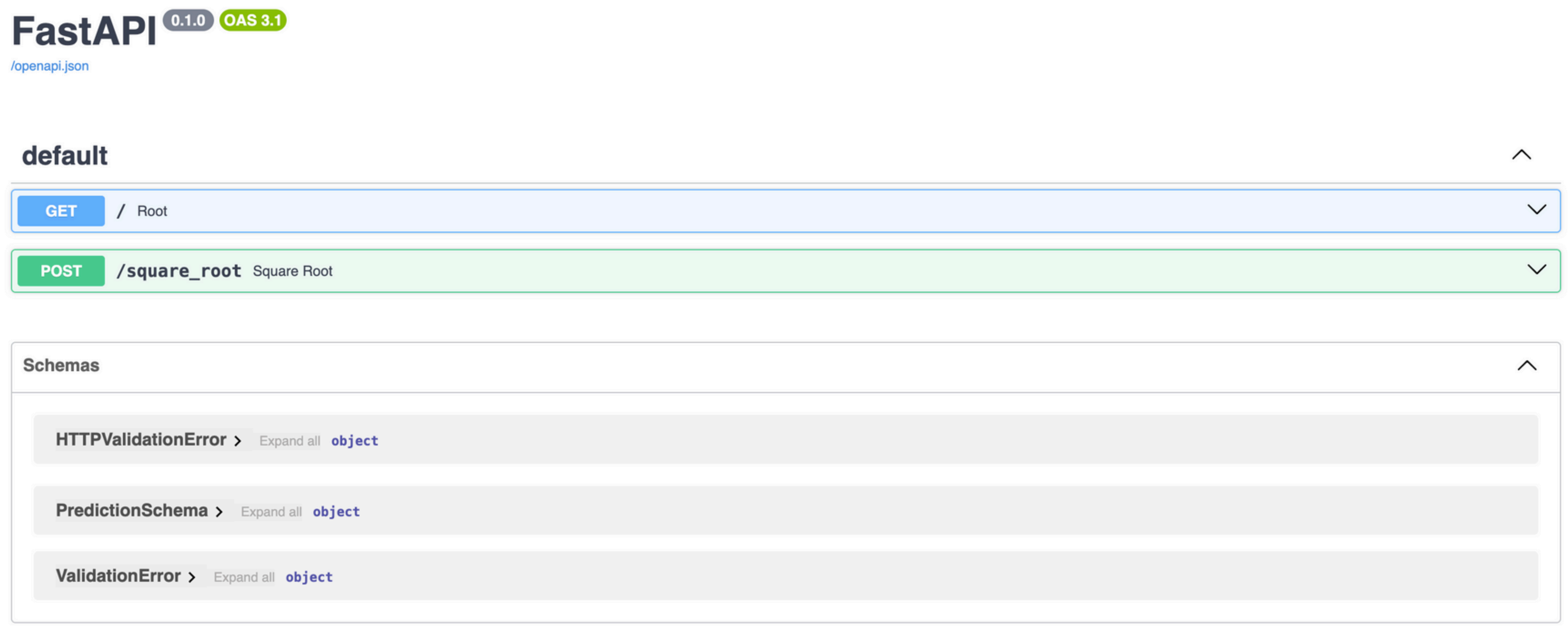
```
{
  "openapi": "3.1.0",
  "info": {
    "title": "FastAPI",
    "version": "0.1.0"
  },
  "paths": {
    "/": {
      "get": {
        "summary": "Root",
        "operationId": "root__get",
        "responses": {
          "200": {
            "description": "Successful Response",
            "content": {
              "application/json": {
                "schema": {

                }
              }
            }
          }
        }
      }
    },
```



```
"/square_root": {
  "post": {
    "summary": "Square Root",
    "operationId": "square_root_square_root_post",
    "parameters": [
      {
        "name": "input_data",
        "in": "query",
        "required": true,
        "schema": {
          "type": "integer",
          "title": "Input Data"
        }
      }
    ],
```

The OpenAPI schema is accessible from http://127.0.0.1:8000/openapi.json

Swagger UI


FastAPI 0.1.0 OAS 3.1
/openapi.json

default

GET / Root

POST /square_root Square Root

Schemas

HTTPValidationError > Expand all object

PredictionSchema > Expand all object

ValidationError > Expand all object

The swagger is accessible from http://127.0.0.1:8000/docs

FastAPI uses HTTP request methods

HTTP	CRUD
GET →	Read method that retrieves data
POST →	Creat method, to submit data
PUT →	Update the entire resource
PATCH →	Update part of the resource
DELETE →	Delete the resource

FastAPI uses HTTP request methods

HTTP		CRUD
TRACE	$\longrightarrow$	Perform a message loop-back
OPTIONS	$\longrightarrow$	Describe communication options
CONNECT	$\longrightarrow$	Create a tunnel to the server

Status code

FastAPI response status code

- An HTTP Status Code is used to help the Client (the user or system submitting data to the server) to understand what happened on the server side application.
- Status Codes are international standards on how a Client/Server should handle the result of a request.
- It allows everyone who sends a request to know if their submission was successful or not.

Status code

FastAPI response status code

$$\begin{tikzcd}
\text{1XX} \arrow[r] & \text{Informational Response} \\
\text{2XX} \arrow[r] & \text{Success} \\
\text{3XX} \arrow[r] & \text{Redirection} \\
\text{4XX} \arrow[r] & \text{Client errors} \\
\text{5XX} \arrow[r] & \text{Server errors}
\end{tikzcd}$$

Status code

2XX Successful status code

200: OK →	Standard Response for a Successful Request. Commonly used for successful Get requests when data is being returned.
201 →	The request has been successful, creating a new resource. Used when a POST creates an entity.
204: No →	The request has been successful, did not create an entity nor return anything. Commonly used with PUT requests.

Status code

4XX Client errors status code

400: BAD $\longrightarrow$	Cannot process request due to client error. Commonly used for invalid request methods.
401: Unauthorized $\longrightarrow$	Client does not have valid authentication for target resource
404: Not $\longrightarrow$	The clients requested resource can not be found
422: Unprocessable entity $\longrightarrow$	Semantic Errors in Client Request

Status code

5XX Server status code

500: Internal Server Error $\longrightarrow$ Generic Error Message, when an unexpected issue on the server happened.

Note: This issue is often caused by a Python error in the code or an unhandled exception

What is FastAPI

FastAPI Python web-framework

Fast for developement & performance.
Faster than Node.js & Go.
The key characteristics are:

- Few bugs
- Quick
- Easy
- Robust
- Standard
- RESTful

Official documentation

FastAPI users

API Endpoints

FastAPI Python web-framework

An API endpoint is a specific URL or URI (Uniform Resource Identifier) that serves as the destination for making a request to an API (Application Programming Interface). It acts as the "access point" for interacting with a specific resource or service that the API provides.

```
@app.get(“/annuaire”)
async def annuaire():
    class_list = [”Lea”, “Axel”]
    return {”annuaire”: class_list}
```

Endpoint example: https://mosef.com$\underbrace{\text{/annuaire}}$ $\longrightarrow$ GET method

API Parameters

FastAPI Python web-framework

There are different ways to pass information to an API:

- **Path**: Adding parameters directly in the URL path (e.g., /api/users/{id}).
- **Query**: Including parameters in the URL after a question mark (e.g., /api/users?id=123).
- **Header**: Sending information as part of the HTTP request header (e.g., Authorization: Bearer <token>).
- **Body**: Including data in the body of the request, typically for POST, PUT, or PATCH requests, in formats like JSON or XML."

API Parameters: Path

FastAPI Python web-framework

Path parameters are request parameters that have been attached to the URL. Usually defined as a way to find info **based on location**. Think of a computer file system, you can identify resources based on the file path:

/Users/Mosef/Documents/Github/API_course/lesson_1

https://mosef.com/annuraire/2023

```
@app.get("/annuaire/{year}")
async def annuaire_year():
    selected_year = annuaire[annuaire["year"]==year]
    return {"annuaire": selected_year}
```

API Parameters: Query

FastAPI Python web-framework

Query parameters are **key-value pairs** appended to the URL after a “**?**”. They are used to pass additional data to the server in a structured format, following the pattern **name=value**, and are typically separated by & for multiple parameters.

https://mosef.com/annuaire/?apprenticeship=health

```
@app.get(“/annuaire”)
async def annuaire(apprenticeship: str):
    selected_domain = annuaire[annuaire[”domain”]==”health”]
    return {”annuaire”: selected_domain}
```

API Parameters: Header

FastAPI Python web-framework

Headers are **key-value** pairs sent as part of the HTTP request, **separate from the URL and body.** They provide additional context or metadata about the request, such as **authentication tokens**, **content type, or caching details**. For example, the header might include the format the client expects in the response or information necessary for the server to process the request correctly. Each header is included in the HTTP request with a specific purpose.

header = {"Authorization": Bearer <token>, "Content-type": "application/json"}

API Parameters: Body

FastAPI Python web-framework

The **body** is the section of an HTTP request or response that contains the actual data being transmitted. Unlike headers or query parameters, the body is not visible in the URL and can carry more complex data, such as JSON, XML, or form data. This is where you send or receive the main information, especially when making **POST, PUT,** or **PATCH** requests, where data needs to be created or updated on the server.

API Parameters: Body

FastAPI Python web-framework

```
class StudentRecord(BaseModel):
    id: int
    last_validate_level: str = Field(min_length=2, max_length=2)
    last_average_mark: int
    dedicated_hours: int

@app.post(“/predict_passing_year”)
async def annuaire(student_info: StudentRecord):
    df_record = pd.DataFrame([StudentRecord])
    return {”score”: model.predict(df_record.drop(columns=”id”))}
```

Pydantics

FastAPI Python web-framework

Pydantics is a python library that is used for data modeling, data parsing and has efficient error handling. It is commonly used as a resource for data validation and how to handle data coming to our FastAPI application.

Create a different request model for data validation and field data validation on each variable:

```
class StudentRecord(BaseModel):
    id: int
    last_validate_level: str = Field(min_length=2, max_length=2)
    last_average_mark: int
    dedicated_hours: int
```

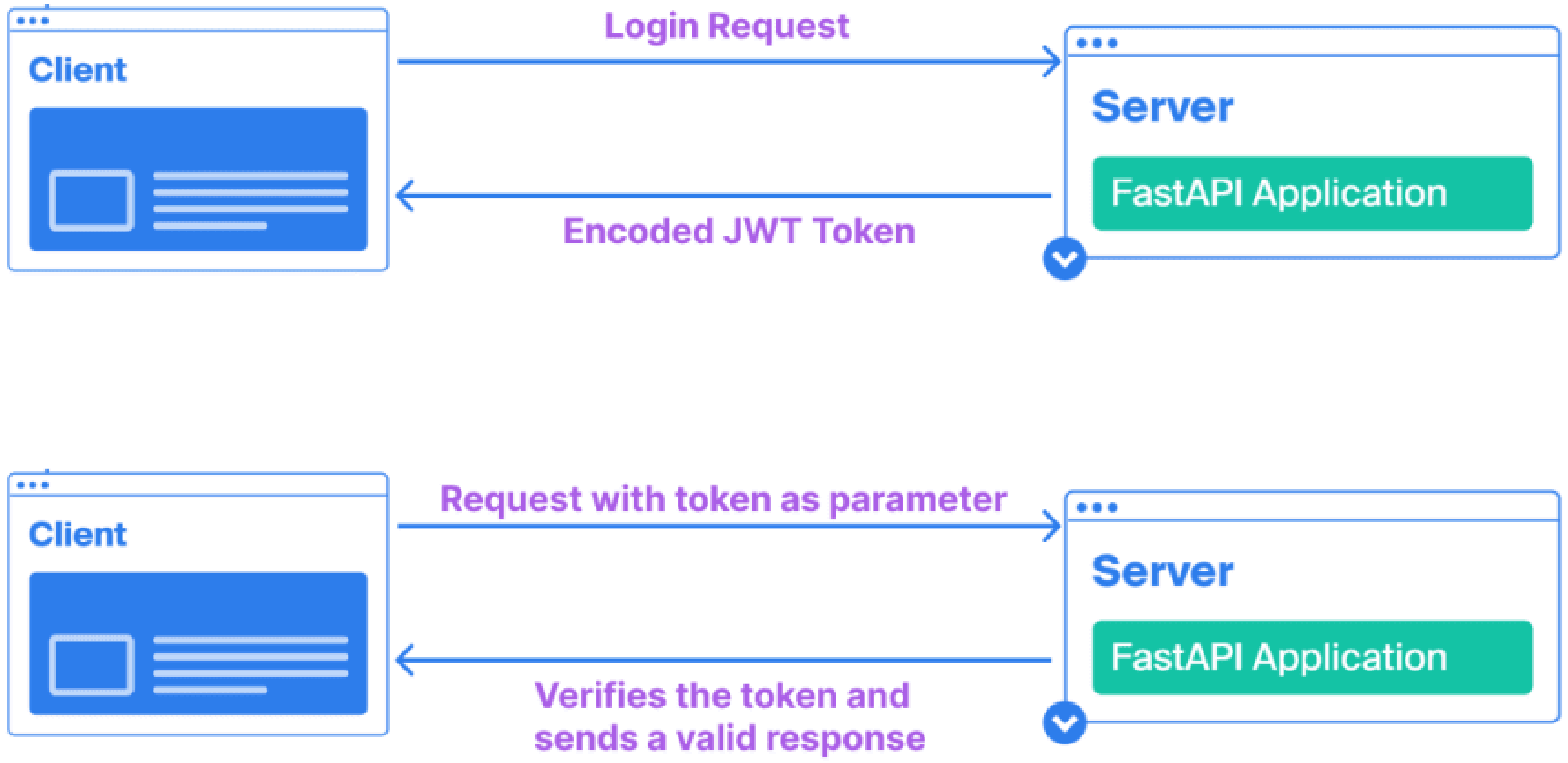
JSON Web Token (JWT)

FastAPI Python web-framework

- JSON Web Token is a self-contained way to securely transmit data and information between two parties using a JSON Object.
- JSON Web Tokens can be trusted because each JWT can be digitally signed, which in return allows the server to know if the JWT has been changed at all
- JWT should be used when dealing with authorization
- JWT is a great way for information to be exchanged between the server and client

JSON Web Token (JWT)

FastAPI Python web-framework



JSON Web Token (JWT)

FastAPI Python web-framework

A JSON Web Token is created of three separate parts separated by dots (.) which include:

- Header : (a)
- Payload : (b)
- Signature : (c)

JWT example: aaaaaaaa.bbbbbbbb.cccccccc

JSON Web Token (JWT)

FastAPI Python web-framework

A JWT **header** usually consist of two parts:

- (alg) The algorithm for signing
- “typ” The specific type of token

The JWT header is then encoded using Base64 to create the first part of the JWT (a)

JWT example: aaaaaaaa.bbbbbbbb.cccccccc

```
{
  ”alg”: “HS256”,
  “typ”: “JWT”
}
```

JSON Web Token (JWT)

FastAPI Python web-framework

A JWT **Payload** consists of the data. The Payloads data contains claims, and there are three different types of claims.

- Registered
- Public
- Private

The JWT Payload is then encoded using Base64 to create the second part of the JWT (b)

JWT example: aaaaaaaa.bbbbbbbb.cccccccc

```
{
  ”sub”: “123456”,
  “name”: “Albert Einstein”,
  “given_name”: “Albert”,
  “family_name”: “Einstein”,
  “admin”: true
}
```

JSON Web Token (JWT)

FastAPI Python web-framework

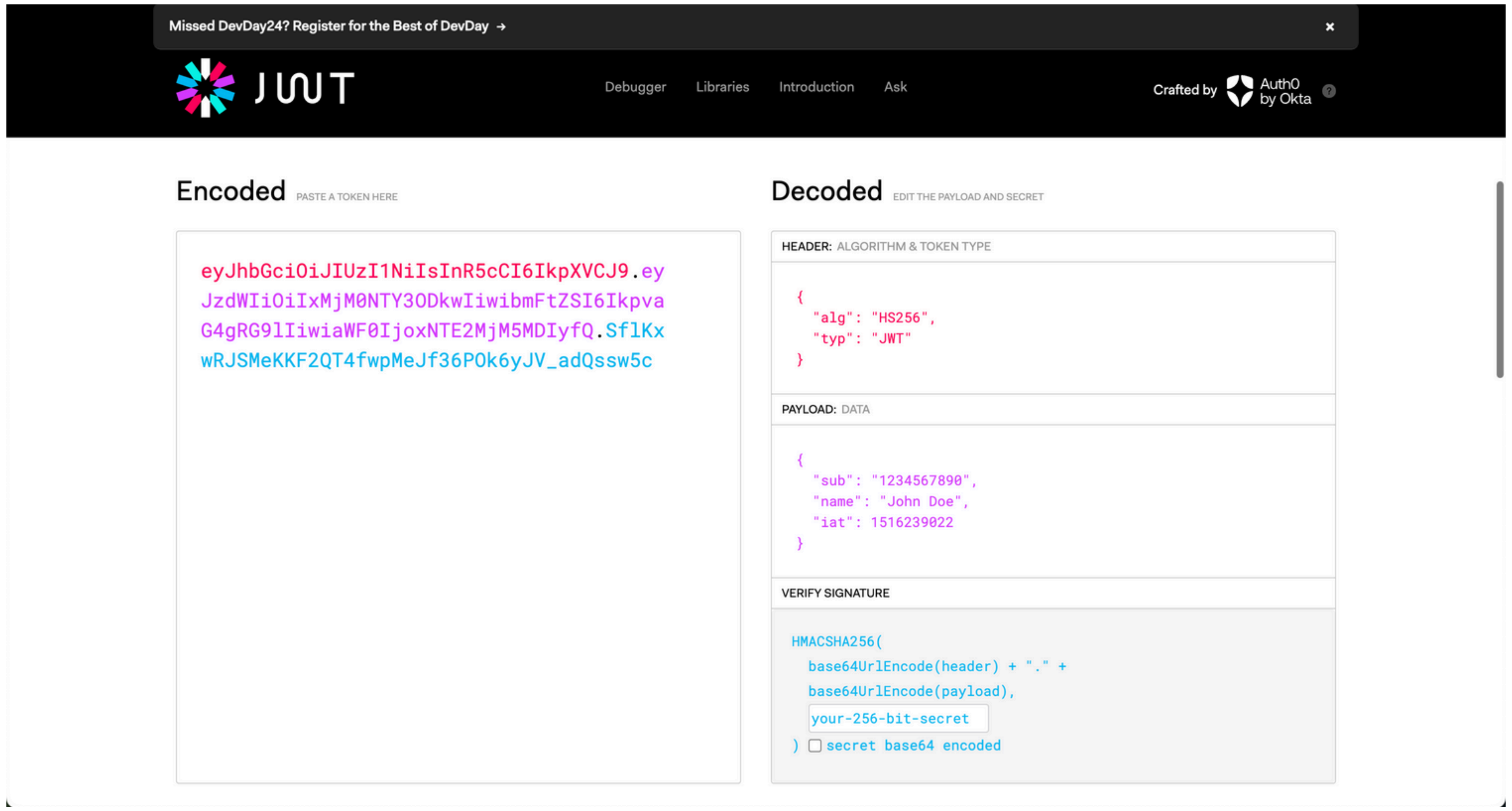
- A JWT Signature is created by using the algorithm in the header to hash out the encoded header, encoded payload with a secret.
- The secret can be anything, but is saved somewhere on the server that the client does not have access to
- The signature is the third and final part of a JWT (c)

```
HMACSHA256(
Base64UrlEncode(header)
+ "." +
Base64UrlEncode(payload)
)
```

JWT example: aaaaaaaa.bbbbbbbb.cccccccc

JSON Web Token (JWT)

FastAPI Python web-framework



Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKx
wRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

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

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

LINK

Coding time

Create a FastAPI API

The API should follow a **modular** and **maintainable structure** (push code to GitHub).

- **Post & Get**: You'll implement basic POST and GET endpoints that allow data submission and retrieval.
- **health_check endpoint**: This endpoint checks the health of the API, typically used for monitoring and determining whether the service is up and running.
- **Call to an external API**: You should add a call to an external API using requests or any other HTTP library.
- **JWT generator endpoint**: The API will have a route that generates a JWT token and a middleware function to validate tokens in secured routes.
- **Little model of machine learning** (even deterministic): For simplicity, implement a basic deterministic model, such as a function that predicts an output based on a mathematical formula (e.g., linear regression with fixed coefficients).