

# JWT Schemas



Full Stack Web Development

# Authenticate Endpoint

- Without these annotations:
  - tags
  - description
  - note
- Then endpoint is invisible to the Swagger plugin

```
authenticate: {
  auth: false,
  handler: async function(request, h) {
    try {
      const user = await db.userStore.getUserByEmail(request.payload.email);
      if (!user) {
        return Boom.unauthorized("User not found");
      } else if (user.password !== request.payload.password) {
        return Boom.unauthorized("Invalid password");
      } else {
        const token = createToken(user);
        return h.response({ success: true, token: token }).code(201);
      }
    } catch (err) {
      return Boom.serverUnavailable("Database Error");
    }
  },
}
```

```
authenticate: {
  auth: false,
  handler: async function(request, h) {
    //...
  },
  tags: ["api"],
  description: "Authenticate a User",
  notes: "If user has valid email/password, create and return a JWT token",
}
```

# Authenticate Annotations

**POST** **/api/users/authenticate** Authenticate a User

If user has valid email/password, create and return a JWT token

**Parameters** Try it out

No parameters

**Responses** Response content type application/json

Code	Description
default	Successful
	<b>Example Value</b>   Model
	"string"

```
authenticate: {
  auth: false,
  handler: async function(request, h) {
    try {
      const user = await db.userStore.getUserByEmail(request.payload.email);
      if (!user) {
        return Boom.unauthorized("User not found");
      } else if (user.password !== request.payload.password) {
        return Boom.unauthorized("Invalid password");
      } else {
        const token = createToken(user);
        return h.response({ success: true, token: token }).code(201);
      }
    } catch (err) {
      return Boom.serverUnavailable("Database Error");
    }
  },
}
```

## Authenticate Response Schema

```
export const JwtAuth = Joi.object()
  .keys({
    success: Joi.boolean().example("true").required(),
    token: Joi.string().example("eyJhbGciOiJIUzI1NiIsInR5cCI6IkpzZW50b3R5IiwiaWF0IjoiMTUxMjM0NTY3OTk5In0").required(),
  })
  .label("JwtAuth");
```

```
export const JwtAuth = Joi.object()
  .keys({
    success: Joi.boolean().example("true").required(),
    token: Joi.string().example("eyJhbGciOiJIJND.g5YmJisIjoiaGYwNTNjA0hE.gCWGmY5-YigQw0DCBo").required(),
  })
  .label("JwtAuth");
```

```
import { UserSpec, UserSpecPlus, IdSpec, UserArray, JwtAuth } from "../models/joi-schemas.js";
...
authenticate: {
  //...
  notes: "If user has valid email/password, create and return a JWT token",
  response: { schema: JwtAuth, failAction: validationError }
}
```

# Authenticate Response Documentation

**POST** `/api/users/authenticate` Authenticate a User

If user has valid email/password, create and return a JWT token

**Parameters** Try it out

No parameters

**Responses** Response content type `application/json`

Code	Description
200	Successful

**Example Value** | **Model**

```
{
  "success": true,
  "token": "eyJhbGciOiJIJND.g5YmJisIjoiaGYwNTNjA0hE.gCWGmY5-YigQw0DCBo"
}
```



# Authenticate Parameter

- Credentials Spec already defined

```
export const UserCredentialsSpec = {  
  email: Joi.string().email().required(),  
  password: Joi.string().required()  
};
```

```
import { UserCredentialsSpec, UserSpec, UserSpecPlus, IdSpec, UserArray, JwtAuth } from "../models/joi-schemas.js";  
  
authenticate: {  
  //...  
  notes: "If user has valid email/password, create and return a JWT token",  
  validate: { payload: UserCredentialsSpec, failAction: validationError },  
  response: { schema: JwtAuth, failAction: validationError }  
}
```

**POST** `/api/users/authenticate` Authenticate a User

If user has valid email/password, create and return a JWT token

Parameters Try it out

Name	Description
body object (body)	<div>Example Value   Model</div> <pre>{   "email": "homer@simpson.com",   "password": "secret" }</pre> <div>Parameter content type application/json</div>

# Test Failures

- Many tests now failing
- This is because authenticate is sending full user details:
  - first name
  - lastname
  - email
  - password
- User Credentials should be:
  - email
  - password

```
...  
"firstName" is not allowed  
...
```

```
await playtimeService.authenticate(maggie);
```

```
export const maggie = {  
  firstName: "Maggie",  
  lastName: "Simpson",  
  email: "maggie@simpson.com",  
  password: "secret"  
};
```

## Test Credentials

- Additional fixture for authenticate

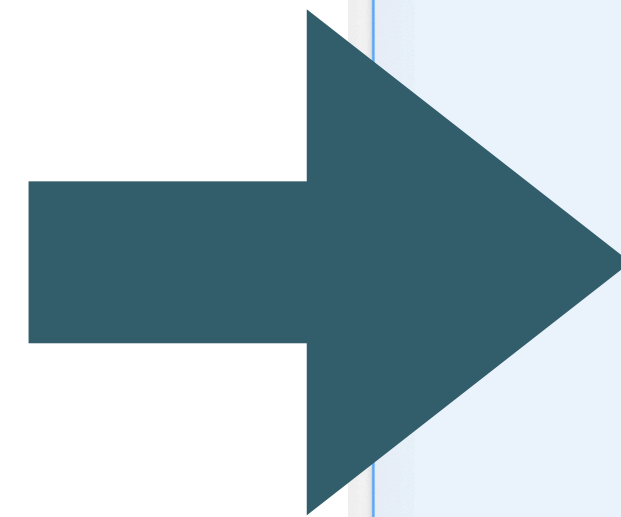
```
export const maggieCredentials = {  
  email: "maggie@simpson.com",  
  password: "secret"  
};
```

```
... await playtimeService.authenticate(maggieCredentials);
```



# Swagger Endpoint Access

- With JWT enabled, the endpoint are no longer accessible from the documentation



api

GET /api/playlists Get all playlists

Returns all playlists

Parameters

No parameters

Execute Clear

Responses

Response content type application/json

Curl

```
curl -X 'GET' \
  'http://localhost:3000/api/playlists' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:3000/api/playlists
```

Server response

Code	Details
401	Error: Unauthorized

Response body

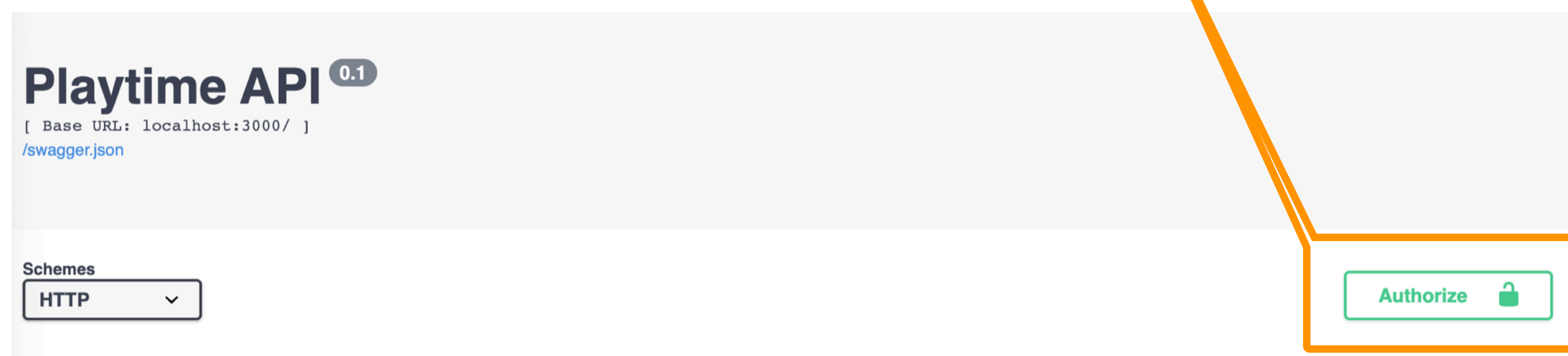
```
{
  "statusCode": 401,
  "error": "Unauthorized",
  "message": "Missing authentication"
}
```

Response headers

```
cache-control: no-cache
connection: keep-alive
content-length: 76
content-type: application/json; charset=utf-8
date: Wed, 02 Mar 2022 08:06:12 GMT
keep-alive: timeout=5
www-authenticate: Token error="token is null"
```

# Swagger Security Definition

- Extend swaggerOptions to include additional security parameters
- This make an Authorize button appear



```
const swaggerOptions = {  
  info: {  
    title: "Playtime API",  
    version: "0.1"  
  },  
  securityDefinitions: {  
    jwt: {  
      type: "apiKey",  
      name: "Authorization",  
      in: "header"  
    }  
  },  
  security: [{ jwt: [] }]  
};
```

# Authorizations Value



**Available authorizations** ×

**jwt (apiKey)**  
Name: Authorization  
In: header  
Value:

Authorize

Close

# Create & Authorize User

Execute

Clear

Responses

Response content type

application/json

Curl

```
curl -X 'POST' \
  'http://localhost:3000/api/users/authenticate' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "email": "homer@simpson.com",
    "password": "secret"
  }'
```

Request URL

http://localhost:3000/api/users/authenticate

Server response

Code

Details

201

Undocumented

Response body

```
{
  "success": true,
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYyMWYyNzk3YWU2ZDVlMWZhNDU1ZjcxYyIsImVtYWlsIjoiaG9tZXJAc2ltcHNvbi5jb20iLCJpYXQiOi0jE2NDYyMDg5MjksImV4cCI6IjYyMTY0NjIxMjUyOX0.ew0TzEQ6BGsi3pPgNCMY0f4WVzVFZPjD0dr-JYbzVK8"
}
```

Response headers

```
cache-control: no-cache
connection: keep-alive
content-length: 235
content-type: application/json; charset=utf-8
date: Wed, 02 Mar 2022 08:15:29 GMT
keep-alive: timeout=5
```

- Copy Token from successful authenticate response

Available authorizations

✕

jwt (apiKey)

Name: Authorization

In: header

Value:

'0f4WVzVFZPjD0dr-JYbzVK8'

Authorize

Close

- All secured routes now become available

## Authorise Using Token

Execute		Clear
Response content type <span style="border: 1px solid black; padding: 2px 5px;">application/json ▼</span>		
<p><b>Curl</b></p> <pre>curl -X 'GET' \ 'http://localhost:3000/api/playlists' \ -H 'accept: application/json' \ -H 'Authorization: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZICIjYyMYWYNzk3YWU2ZDVlMWZhNDU1ZjcxYyIsImVtYWlsIjoiaG9tZXJAc2ltcHNvbi5jb20iLCJpYXQiOjE2NDYyMDg5Mjk5ImV4cCI6MTY0NjIxMjUyOXB0.ev'</pre>		
Request URL		
<b>http://localhost:3000/api/playlists</b>		
Server response		
Code	Details	
200	<p>Response body</p> <pre>[ {   "_id": "621f22e75fc33e9a3acb530c",   "title": "Mozart Favourites",   "userid": "621f22e75fc33e9a3acb5307",   "__v": 0 } ]</pre> <p style="text-align: right;">Download</p> <p>Response headers</p> <pre>accept-ranges: bytes cache-control: no-cache connection: keep-alive content-length: 108 content-type: application/json; charset=utf-8 date: Wed, 02 Mar 2022 08:19:14 GMT keep-alive: timeout=5</pre>	
Responses		
200	<p>Description</p> <p>Successful</p> <p>Example Value   Model</p> <pre>[ {   "title": "Beethoven Sonatas",   "userid": "string",   "tracks": [     {       "title": "Piano Sonata No. 7",       "artist": "Beethoven",       "duration": 12,       "playlistid": "string",       "_id": "string",       "__v": 0     }   ],   "_id": "string",   "__v": 0 }</pre>	



# Curl

- Command line tool for low level network exploration
- Supports a large range of protocols and standards



**command line tool and library**  
for transferring data with URLs  
(since 1998)

## Supports...

DICT, FILE, FTP, FTPS, GOPHER, GOPHERS, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, MQTT, POP3, POP3S, RTMP, RTMPS, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET and TFTP. curl supports SSL certificates, HTTP POST, HTTP PUT, FTP uploading, HTTP form based upload, proxies, HTTP/2, HTTP/3, cookies, user+password authentication (Basic, Plain, Digest, CRAM-MD5, SCRAM-SHA, NTLM, Negotiate and Kerberos), file transfer resume, proxy tunneling and more.

## What's curl used for?

curl is used in command lines or scripts to transfer data. curl is also used in cars, television sets, routers, printers, audio equipment, mobile phones, tablets, settop boxes, media players and is the Internet transfer engine for thousands of software applications in over *ten billion installations*.

curl is used daily by virtually every Internet-using human on the globe.

## Who makes curl?

curl is free and [open source](#) software and exists thanks to [thousands of contributors](#) and our awesome [sponsors](#). The curl project [follows well established open source best practices](#). You too can [help us](#) improve!

## What's the latest curl?

The most recent stable version is **7.81.0**, released on 5th of January 2022. Currently, 79 of the listed [downloads](#) are of the latest version.

## Where's the code?

Check out the latest [source code from GitHub](#).

## Top Sponsors

**haxx**

**wolfSSL**

**fastly**

**TeamViewer**

Time to [donate](#)  
to the curl  
project?

[Commercial](#)  
[support](#)  
available!

[Everything curl](#) is a  
detailed and totally  
free book that  
explains basically  
everything there is to  
know about curl.



# Curl Command

## Curl

```
curl -X 'GET' \
'http://localhost:3000/api/playlists' \
-H 'accept: application/json' \
-H 'Authorization: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYyMWYyNzk3YWU2ZDVlMWZhNDU1ZjcxYyIsImVtYWlsIjoiaG9tZXJAc2ltcHNvbi5jb20iLCJpYXQiOjE2NDYyMDg5MjksImV4cCI6MTY0NjIxMjUyOX0.ev
```


- Token visible from Curl command



```
Eamonns-Mac-mini:~ edeleastar$ curl -X 'GET' \
> 'http://localhost:3000/api/playlists' \
> -H 'accept: application/json' \
> -H 'Authorization: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYyMjFiZWMyZmE2Y2I4N2VmMDQ5MTIyOStIsImVtYWlsIjoiaG9tZXJAc2ltcHNvbi5ib20iLCJpYX0iOiE2NDYzNza3NDAsImV4cCI6MTY0Ni40MjM0MH0.nnwbOYaRnLBwLcaNXfdDkIGVaWmIOcCvNkd9VVnz5kA'
[{"_id":"6221bec2fa6cb87ef049122f","title":"Mozart Favourites","userid":"6221bec2fa6cb87ef049122d","__v":0}]Eamonns-Mac-mini:~ edeleastar$
```

## Request

- Response

- 



## Debugger

Libraries

## Introduction

As

Crafted by  auth0 

Encoded

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV\_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

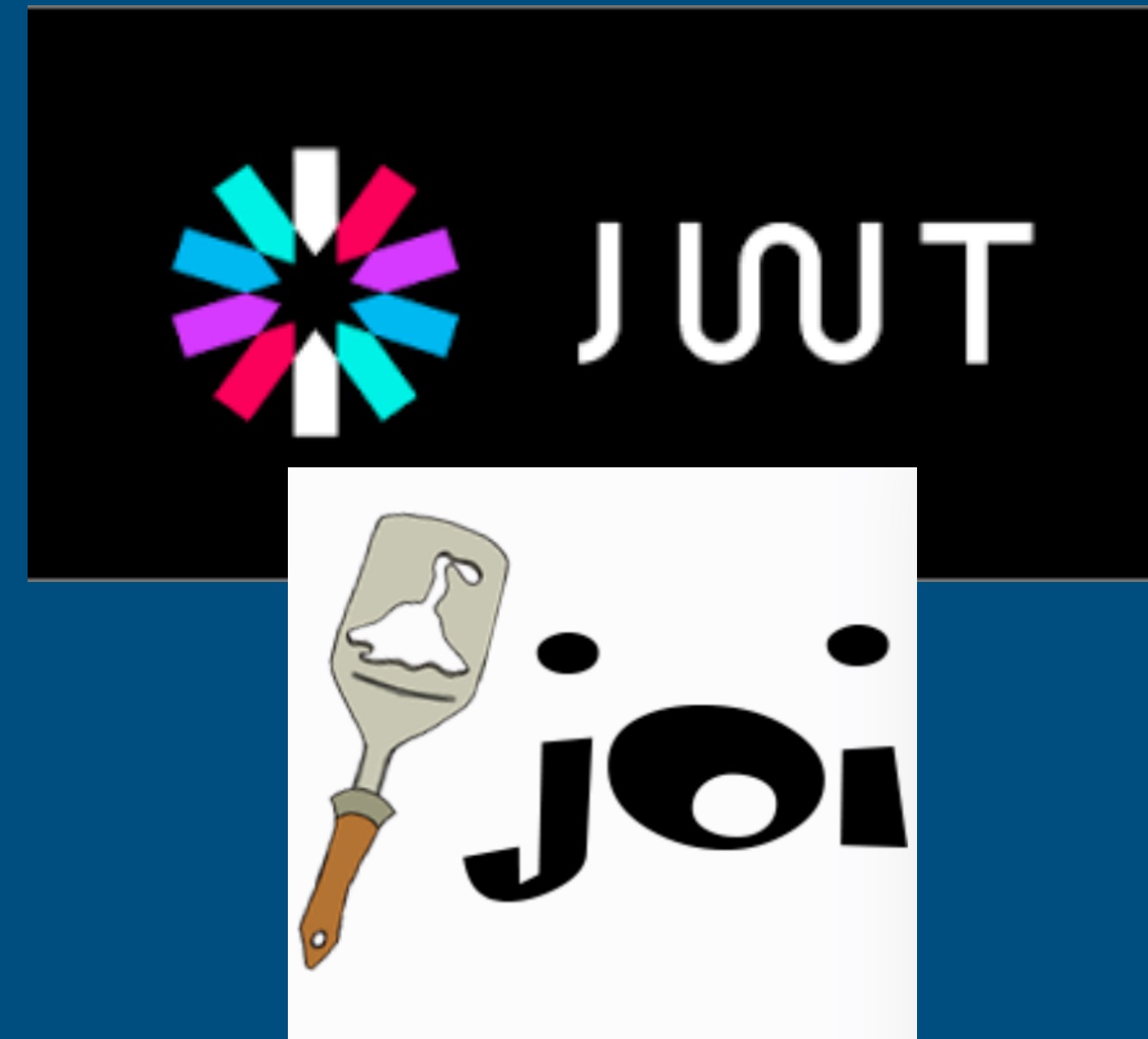
```

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

```

jwt.io

# JWT Schemas



Full Stack Web Development