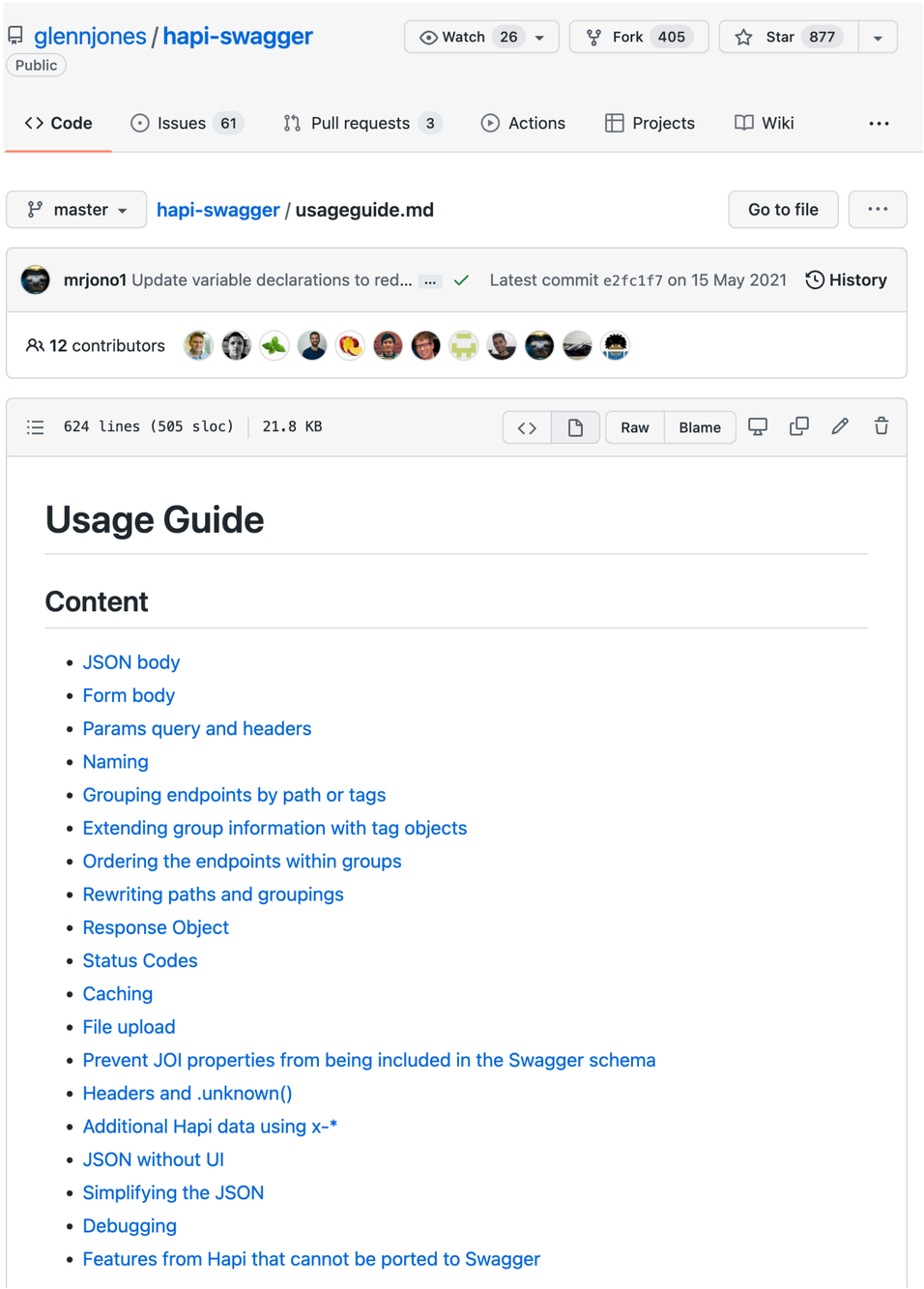
Playtime Swagger

Full Stack Web Development

hapi-swagger

- A plugin for HAPI
- Leverages Hapi Routes + Joi Schemas to automatically generate Swagger / Open API 2.0 Specification



Install Plugin

- Install
- Configure
- Initialise

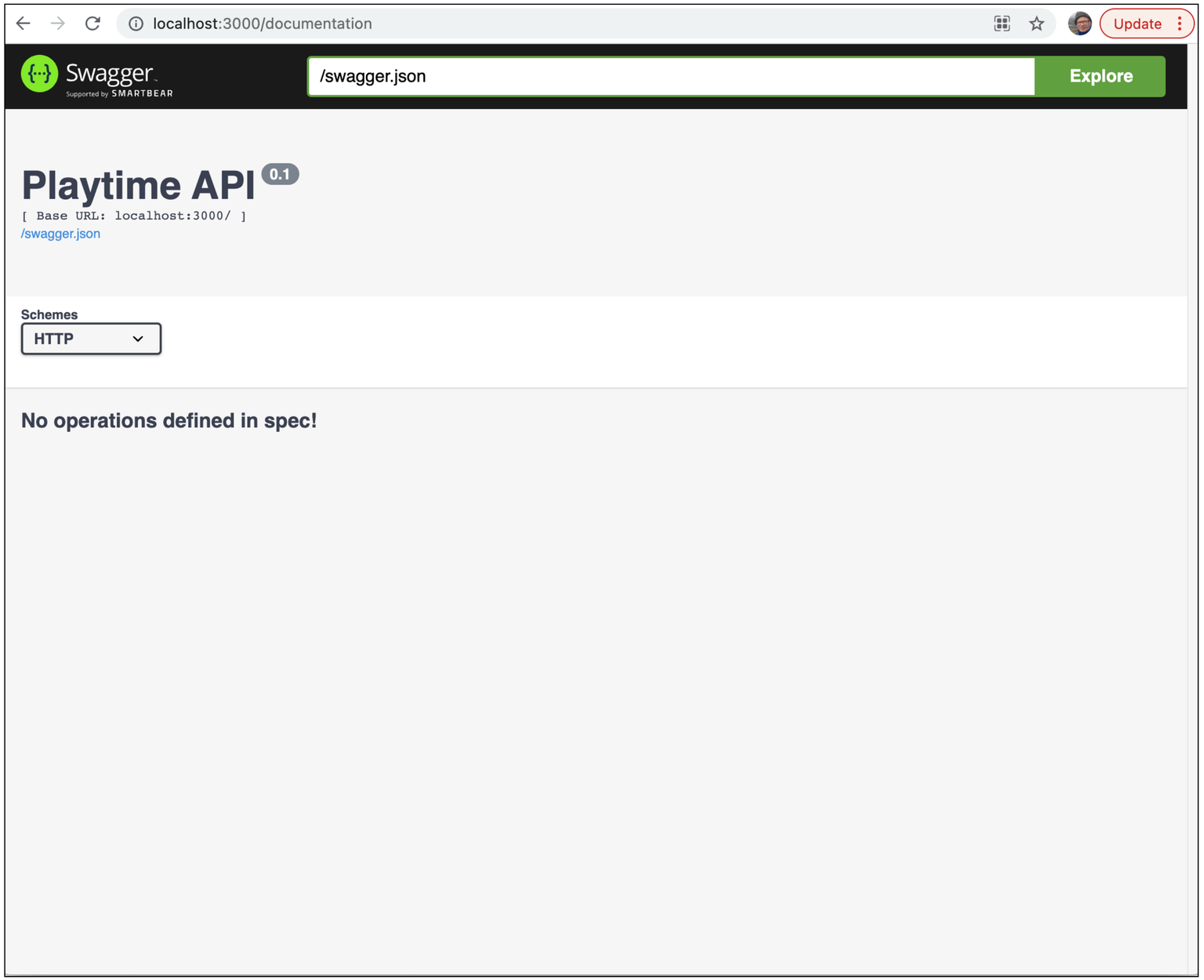
```
npm install hapi-swagger
```

```
import HapiSwagger from "hapi-swagger";

const swaggerOptions = {
  info: {
    title: "Playtime API",
    version: "0.1",
  },
};
```

```
await server.register([
  Inert,
  Vision,
  {
    plugin: HapiSwagger,
    options: swaggerOptions,
  },
]);
```

API Hub Site

- Plugin generates a Web Site Compatible with the SwaggerHub
- Web site hosts
  - Generated Spec
  - Forms for exploring the API
  - API Invocation workbench



API Controller Annotations

- Directly in the api controllers, specify:
  - tag
  - Description
  - Notes

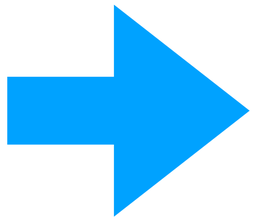
user-api.js

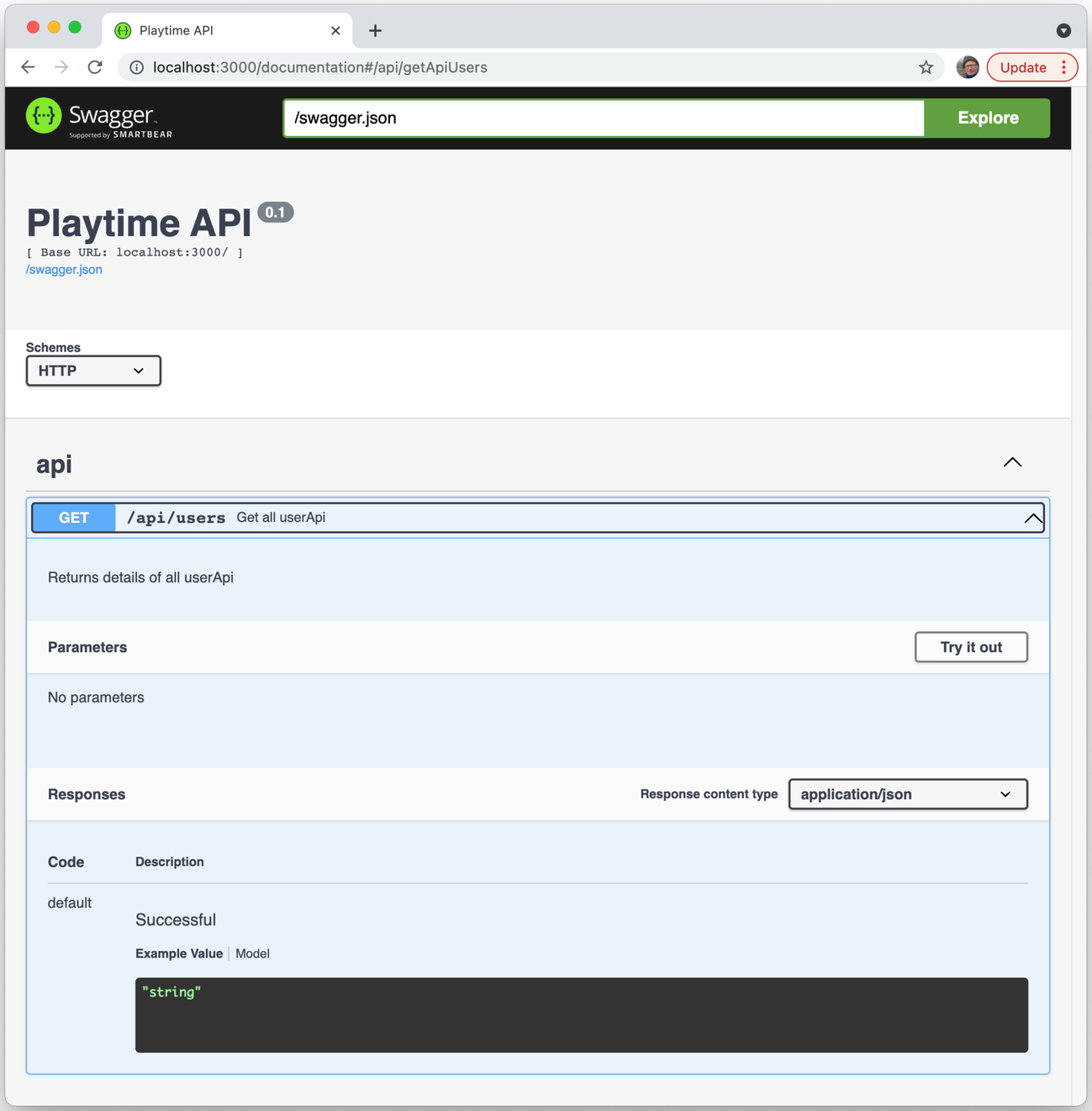
```
export const userApi = {
  find: {
    auth: false,
    handler: async function(request, h) {
      try {
        const users = await db.userStore.getAllUsers();
        return users;
      } catch (err) {
        return Boom.serverUnavailable("Database Error");
      }
    },
    tags: ["api"],
    description: "Get all userApi",
    notes: "Returns details of all userApi",
  },
```

user-api.js

```
export const userApi = {
  find: {
    auth: false,
    handler: async function(request, h) {
      try {
        const users = await db.userStore.getAllUsers();
        return users;
      } catch (err) {
        return Boom.serverUnavailable("Database Error");
      }
    },
    tags: ["api"],
    description: "Get all userApi",
    notes: "Returns details of all userApi",
  },
```

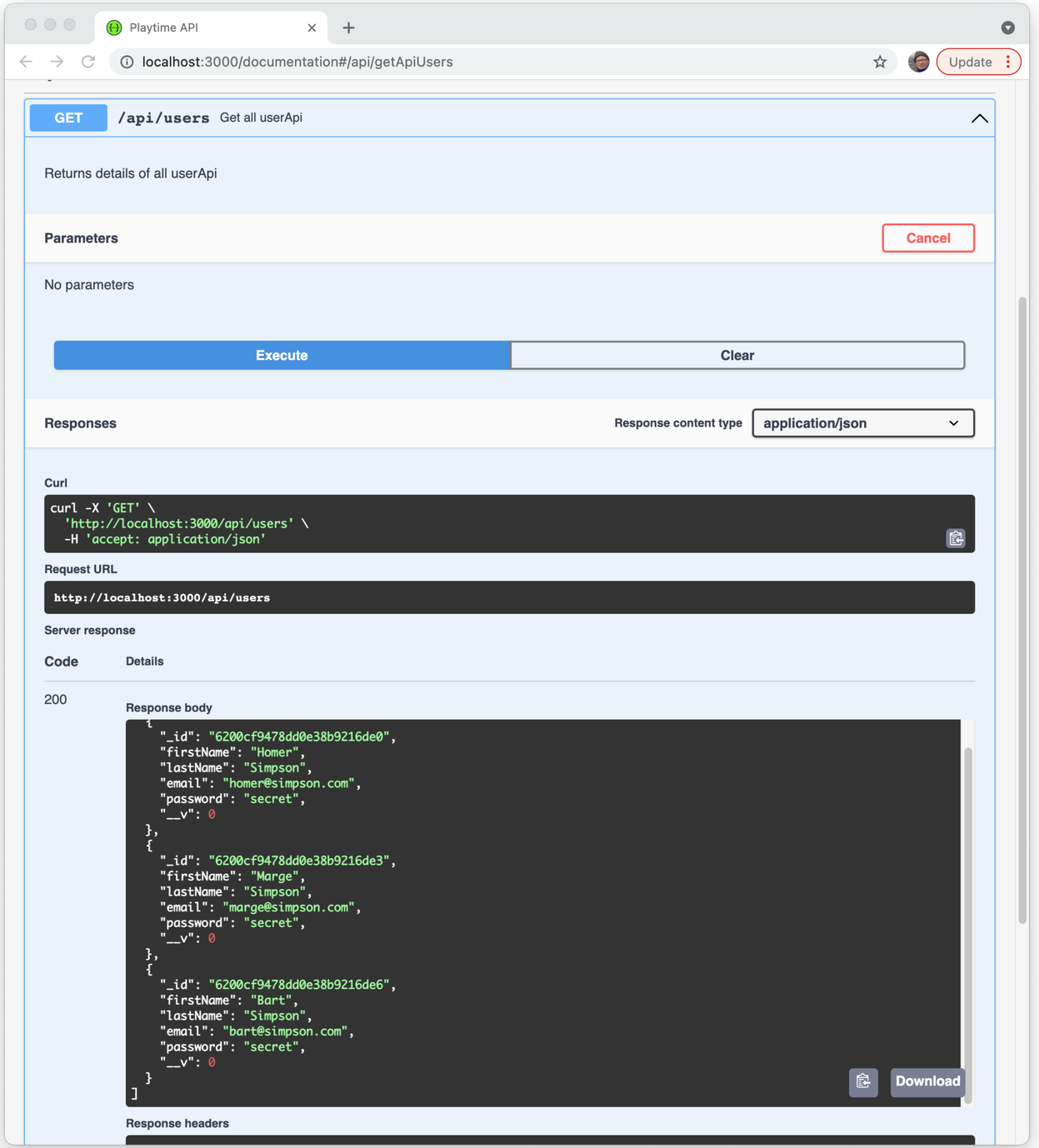
- Plug in generates API specification

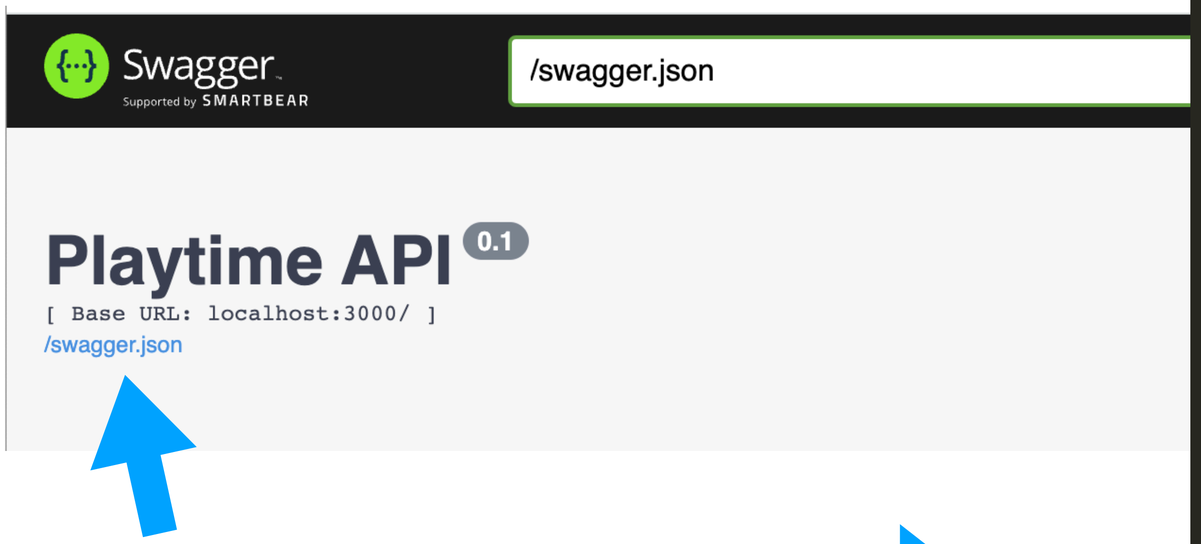


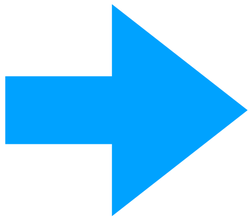


Execute

- Response from GET /api/users



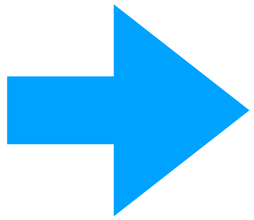





```
{
  "swagger": "2.0",
  "host": "localhost:3000",
  "basePath": "/",
  "schemes": [
    "http"
  ],
  "info": {
    "title": "Playtime API",
    "version": "0.1"
  },
  "tags": [

  ],
  "paths": {
    "/api/users": {
      "get": {
        "summary": "Get all userApi",
        "operationId": "getApiUsers",
        "description": "Returns details of all userApi",
        "tags": [
          "api"
        ],
        "responses": {
          "default": {
            "schema": {
              "type": "string"
            },
            "description": "Successful"
          }
        }
      }
    }
  },
  "definitions": {

  }
}
```


- swagger.json available from the generated swagger site

```json
{
  "swagger": "2.0",
  "host": "localhost:3000",
  "basePath": "/",
  "schemes": [
    "http"
  ],
  "info": {
    "title": "Playtime API",
    "version": "0.1"
  },
  "tags": [

  ],
  "paths": {
    "/api/users": {
      "get": {
        "summary": "Get all userApi",
        "operationId": "getApiUsers",
        "description": "Returns details of all userApi",
        "tags": [
          "api"
        ],
        "responses": {
          "default": {
            "schema": {
              "type": "string"
            },
            "description": "Successful"
          }
        }
      }
    }
  },
  "definitions": {

  }
}
```



```yaml
swagger: '2.0'
host: localhost:3000
basePath: "/"
schemes:
- http
info:
  title: Playtime API
  version: '0.1'
tags: []
paths:
  "/api/users":
    get:
      summary: Get all userApi
      operationId: getApiUsers
      description: Returns details of all userApi
      tags:
      - api
      responses:
        default:
          schema:
            type: string
          description: Successful
definitions: {}
```

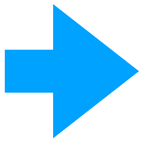
Yaml version via

https://www.json2yaml.com/

Joi/Swagger

- Current UserSpec Joi Schema employed for form validation.
- Joi has additional attributes, including **example** & **label**

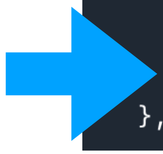
```
export const UserSpec = {
  firstName: Joi.string().required(),
  lastName: Joi.string().required(),
  email: Joi.string().email().required(),
  password: Joi.string().required(),
};
```



```
export const UserSpec = Joi.object().keys({
  firstName: Joi.string().example("Homer").required(),
  lastName: Joi.string().example("Simpson").required(),
  email: Joi.string().email().example("homer@simpson.com").required(),
  password: Joi.string().example("secret").required(),
}).label("User Details");

export const UserArray = Joi.array().items(UserSpec).label("User Array");
```

Response Validation

```
export const UserSpec = Joi.object().keys({
  firstName: Joi.string().example("Homer").required(),
  lastName: Joi.string().example("Simpson").required(),
  email: Joi.string().email().example("homer@simpson.com").required(),
  password: Joi.string().example("secret").required(),
}).label("User Details");

export const UserArray = Joi.array().items(UserSpec).label("User Array");
```

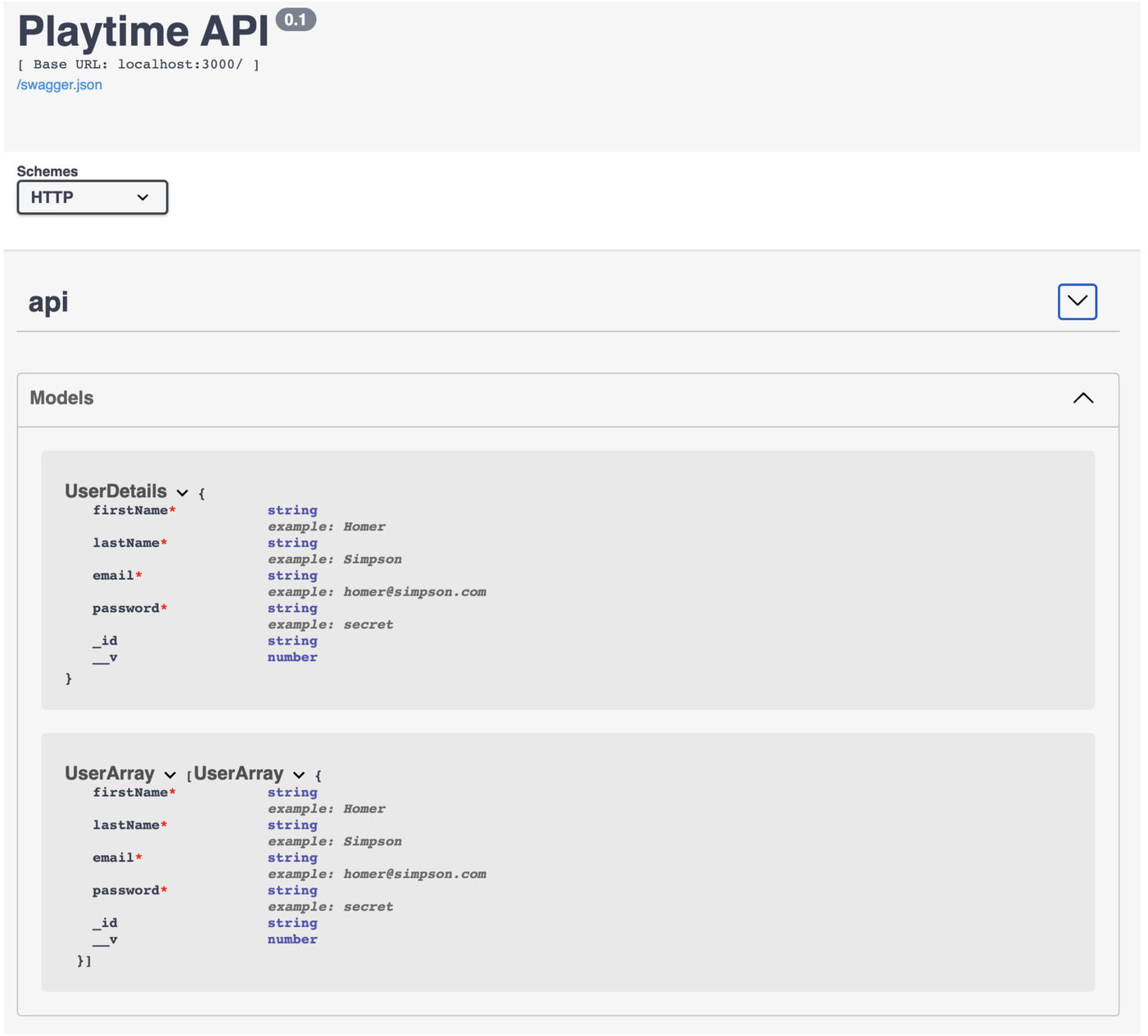


```
export const userApi = {
  find: {
    auth: false,
    handler: async function(request, h) {
      try {
        const users = await db.userStore.getAllUsers();
        return users;
      } catch (err) {
        return Boom.serverUnavailable("Database Error");
      }
    },
    tags: ["api"],
    description: "Get all userApi",
    notes: "Returns details of all userApi",
    response: { schema: UserArray }
  },
```

- **Response:** Joi validation will be engaged to validate the response

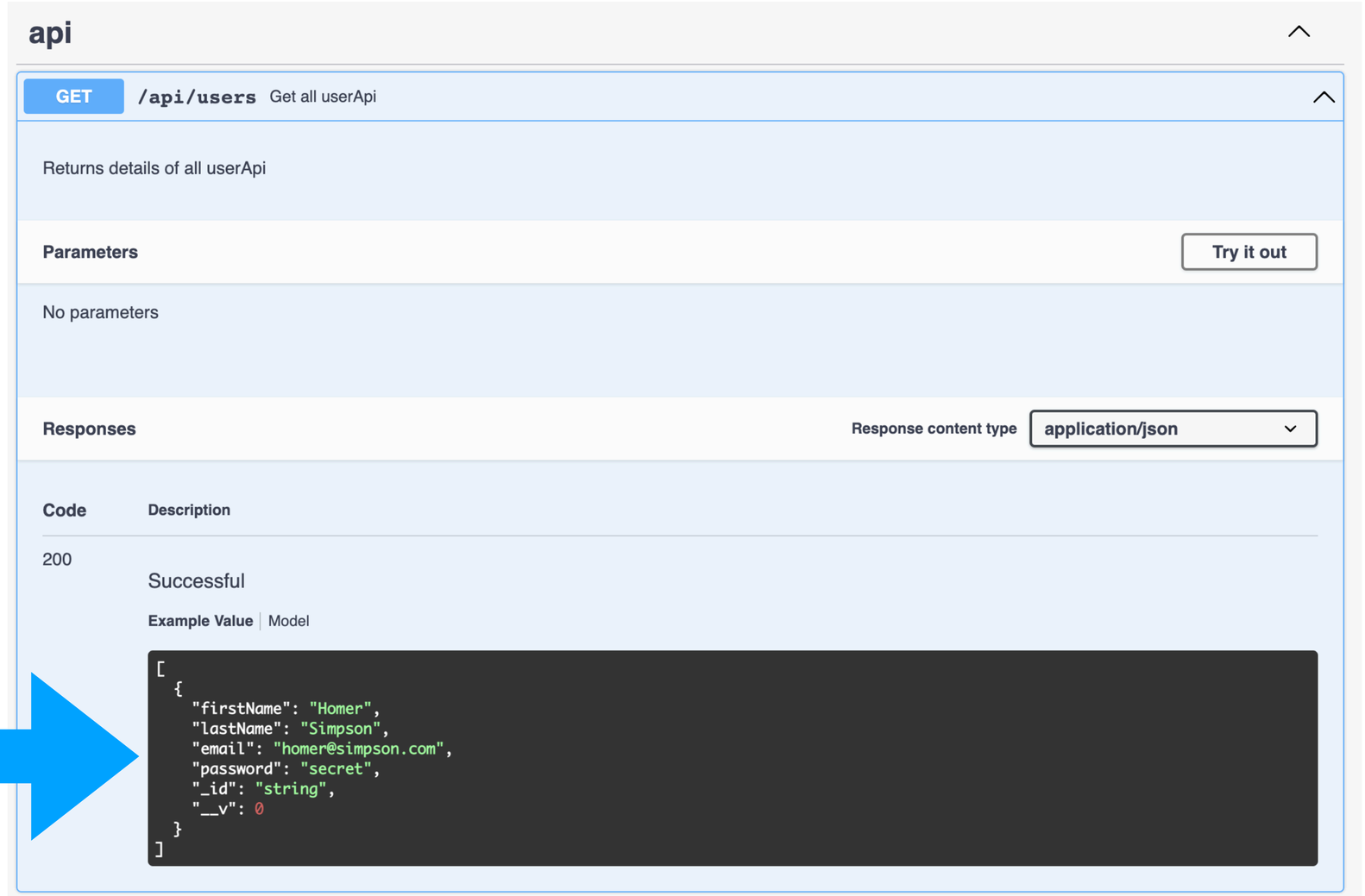
Model Documentation

- Models section generated in documentation



Response Documentation

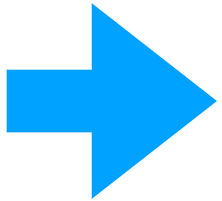
- Example successful response generated from Joi schema



Swagger/OpenAPI Specification

```
export const UserSpec = Joi.object().keys({
  firstName: Joi.string().example("Homer").required(),
  lastName: Joi.string().example("Simpson").required(),
  email: Joi.string().email().example("homer@simpson.com").required(),
  password: Joi.string().example("secret").required(),
}).label("User Details");

export const UserArray = Joi.array().items(UserSpec).label("User Array");
```

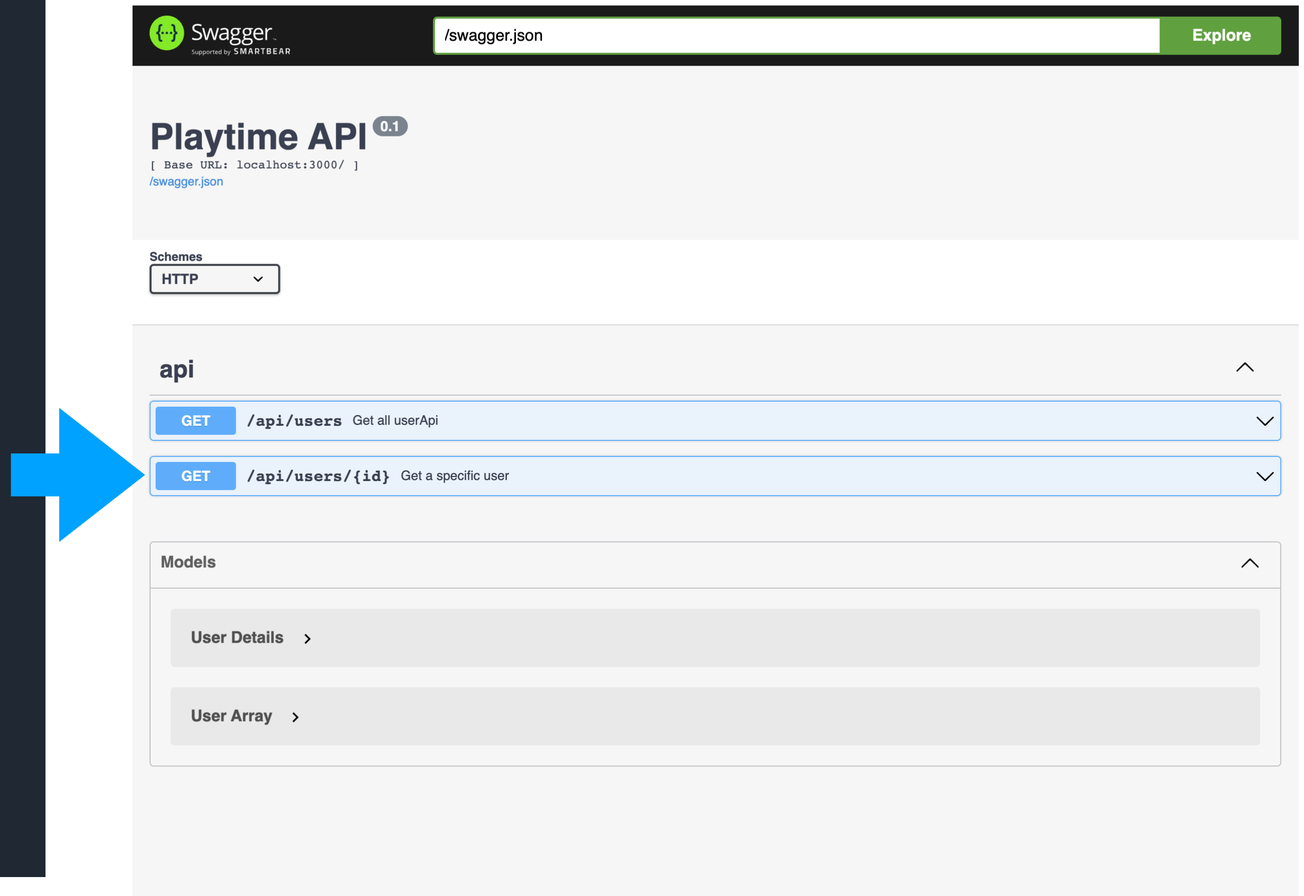
```
export const userApi = {
  find: {
    auth: false,
    handler: async function(request, h) {
      try {
        const users = await db.userStore.getAllUsers();
        return users;
      } catch (err) {
        return Boom.serverUnavailable("Database Error");
      }
    },
    tags: ["api"],
    description: "Get all userApi",
    notes: "Returns details of all userApi",
    response: { schema: UserArray }
  },
```



```
swagger: '2.0'
host: localhost:3000
basePath: "/"
schemes:
- http
info:
  title: Playtime API
  version: '0.1'
tags: []
paths:
  "/api/users":
    get:
      summary: Get all userApi
      operationId: getApiUsers
      description: Returns details of all userApi
      tags:
      - api
      responses:
        '200':
          schema:
            "$ref": "#/definitions/UserArray"
          description: Successful
definitions:
  UserDetails:
    type: object
    properties:
      firstName:
        type: string
        example: Homer
      lastName:
        type: string
        example: Simpson
      email:
        type: string
        example: homer@simpson.com
        x-format:
          email: true
      password:
        type: string
        example: secret
      _id:
        type: string
        x-alternatives:
        - type: string
        - "$ref": "#/x-alt-definitions/_id"
      __v:
        type: number
    required:
    - firstName
    - lastName
    - email
    - password
  UserArray:
    type: array
    items:
      "$ref": "#/definitions/UserDetails"
x-alt-definitions:
  _id:
    type: object
```

Validate Response

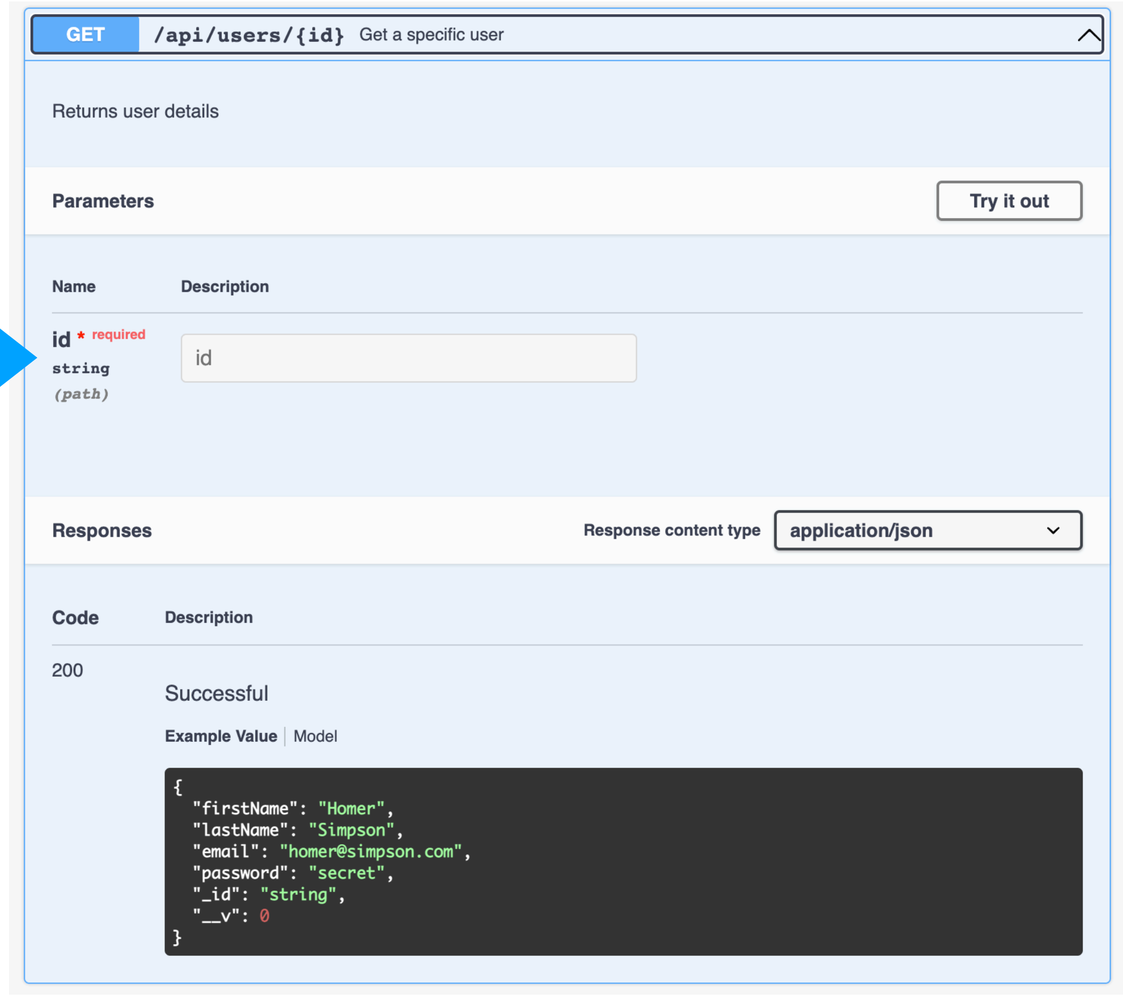
```
import { UserSpec, UserArray } from "../models/joi-schemas.js";
...
  findOne: {
    auth: false,
    handler: async function (request, h) {
      try {
        const user = await db.userStore.getUserById(request.params.id);
        if (!user) {
          return Boom.notFound("No User with this id");
        }
        return user;
      } catch (err) {
        return Boom.serverUnavailable("No User with this id");
      }
    },
    tags: ["api"],
    description: "Get a specific user",
    notes: "Returns user details",
    response: { schema: UserSpec, failAction: validationError },
  },
```




```
export function validationError(request, h, error) {
  console.log(error.message);
}
```

Validate Request

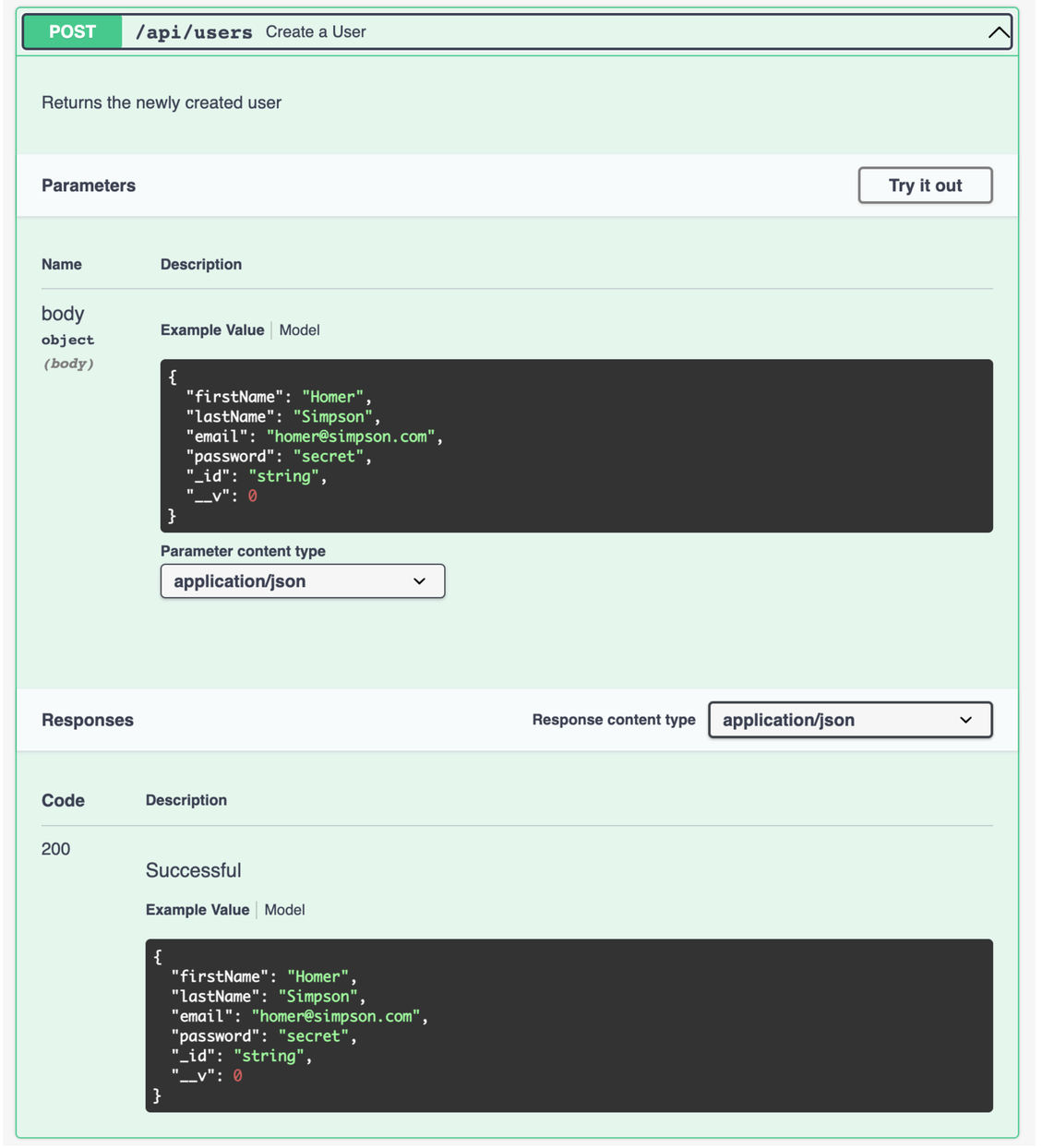
```
findOne: {
  auth: false,
  handler: async function(request, h) {
    try {
      const user = await db.userStore.getUserById(request.params.id);
      if (!user) {
        return Boom.notFound( message: "No User with this id");
      }
      return user;
    } catch (err) {
      return Boom.serverUnavailable( message: "No User with this id");
    }
  },
  tags: ["api"],
  description: "Get a specific user",
  notes: "Returns user details",
  validate: { params: { id: IdSpec }, failAction: validationError },
  response: { schema: UserSpec, failAction: validationError },
},
```



- Validate input paramater

Validate Request/Response

```
create: {
  auth: false,
  handler: async function(request, h) {
    try {
      const user = await db.userStore.addUser(request.payload);
      if (user) {
        return h.response(user).code(201);
      }
      return Boom.badImplementation("error creating user");
    } catch (err) {
      return Boom.serverUnavailable("Database Error");
    }
  },
  tags: ["api"],
  description: "Create a User",
  notes: "Returns the newly created user",
  validate: { payload: UserSpec, failAction: validationError },
  response: { schema: UserSpec, failAction: validationError },
},
```



Playtime API 0.1

[Base URL: localhost:3000/]

/swagger.json

Schemes

HTTP

api

GET /api/users Get all userApi

POST /api/users Create a User

DELETE /api/users Delete all userApi

GET /api/users/{id} Get a specific user

Models

UserDetails

UserArray

```yaml
swagger: '2.0'
host: localhost:3000
basePath: "/"
schemes:
- http
info:
  title: Playtime API
  version: '0.1'
tags: []
paths:
  "/api/users":
    get: ...
    post: ...
    delete: ...
  "/api/users/{id}":
    get: ...
definitions:
  UserDetails:
    type: object
    properties:
      firstName:
        type: string
        example: Homer
      lastName:
        type: string
        example: Simpson
      email:
        type: string
        example: homer@simpson.com
        x-format:
          email: true
      password:
        type: string
        example: secret
      _id:
        type: string
        x-alternatives:
        - type: string
        - "$ref": "#/x-alt-definitions/_id"
      __v:
        type: number
    required:
    - firstName
    - lastName
    - email
    - password
  UserArray:
    type: array
    items:
      "$ref": "#/definitions/UserDetails"
x-alt-definitions:
  _id:
    type: object
```

Playtime Swagger

Full Stack Web Development