Sending APIs

1. For learning and api testing - open postman

Agenda:

Sending Apis to a lot of routes with crud functionality and exploring main ways how to send information to server.

The company has made market research as it aims to beat Starbucks and found nieche that has not been filled - selling used coffee. They have heard that girls on OnlyFans are making a lot of money selling used stuff online.
every user who sold coffee would get 95% of price.
every client would pay extra 10% in fees for purchased coffee.
But business did not have enough funds to make proper market place(website) so they have created only apis that clients can use to sell and buy used coffee.

params

/users/:id/product/:id

queries

/users/:id/product/:id?sold=false?order=dsc|asc?search=black

headers

for private routes will use Headers and I will try to mix headers to get more training possibilities. Some will need only Authorization, some username + api-key, banana etc.
auth database:

provide auth database with userId so every time someone tries to login again, every value is changed.

id userId username token api-key banana X-DIRECTIVE-IMPORTANT please

please - in message (okay, only because you said please, if not provided will respond what is the magical word)
Authorization: "Bearer token"
Api-key: api-key
banana:
username: "john dow"

body with JSON

All POST and PUT/PATCH routes
E.g registering
{"email": "test@test.com", "password": "secret", "username": "john dow"}

users

```
CREATE TABLE users (
    id INT PRIMARY KEY,
    email VARCHAR(255),
    password VARCHAR(50),
    username VARCHAR(255),
);


CREATE TABLE auth_users (
    id INT PRIMARY KEY,
    userId INT,
    username VARCHAR(255),
    token VARCHAR(255),
    api_key VARCHAR(255),
    banana VARCHAR(10) DEFAULT 'banana',
    please VARCHAR(10) DEFAULT 'please',
    X_DIRECTIVE_STUFF VARCHAR(255),
    expires DATE,
    FOREIGN KEY (userId) REFERENCES users(id)
);
```

used_coffee

```
CREATE TABLE used_coffee (
    id INT PRIMARY KEY,
    userId INT,
    name VARCHAR(255),
    brand VARCHAR(50) DEFAULT 'NO BRAND',
    description TEXT,
    image_link VARCHAR(255),
    price DECIMAL(10, 2),
    pct_left INT,
    purchase_date VARCHAR(100),
    sold BOOLEAN DEFAULT FALSE,
    FOREIGN KEY (userId) REFERENCES users(id),
);
```

orders

```
CREATE TABLE orders (
    id INT PRIMARY KEY,
    user_id INT,
    used_coffee_id INT,
    address VARCHAR(255),
    order_date DATE,
    status VARCHAR(255),
    FOREIGN KEY (userId) REFERENCES users(id),
    FOREIGN KEY (product_id) REFERENCES used_coffee(id),
);
```

rating

```
CREATE TABLE rating (
    id INT PRIMARY KEY,
    user_id INT,
    used_coffee_id INT,
    rating INT,
    date DATE,
    status VARCHAR(255),
    FOREIGN KEY (userId) REFERENCES users(id),
    FOREIGN KEY (product_id) REFERENCES used_coffee(id),
);
```

User Routes

@desc	@route	@access
Register User	POST /api/v1/auth/register	Public
Login User	POST /api/v1/auth/login	Public

@desc	@route	@access
Get Single User	GET /api/v1/users/:id	Admin
Get All Users	GET /api/v1/users/	Admin

Register

Register Instruction

Send POST request to correct route with body that would contain JSON format data
Correct Example: {"email": "test@test.com", "password": "secret", "username": "john dow"}

Register Error

send body to correct api route:
need unique email, password must be at least 6char long, username need at least 3 characters.

Register OK

send back message that user has been registered
{message: "user has been registered successfully"}

Login

Login Instruction

Send POST request to correct route with body that would contain JSON format data
Correct Example: {"email": "test@test.com", "password": "secret"}

Login Error

email and password need to match with values in server, if not send error

Login OK

{message: "logged in successfully", data: {username: your-username, token: "random id generated by some package=", api-key: "random id", banana: "banana", X-DIRECTIVE-STUFF: "random id", please: "please", }}

Group Work (5 minutes) 3-4 people

Check each others Postman setup and ask for advice if something did not work.

Get Single User

Get Single User Instruction

send GET request to correct route with id
Request needs to have header:

key	value
who-is-the-boss	IAMTHEBOSS

Get Single User Error

wrong id - {message: "User Id does not exist"}
Not authorized - {message: "You are not my boss!"}

Get Single User OK

{message: "here you go boss", data: {id: 1, username: "marisklava", email:"test@test.com"}}
Password is encrypted in database and excluded for security reasons

Get All Users

Get All Users Instruction

send GET request to correct route
Request needs to have headers:

key	value
who-is-the-boss	IAMTHEBOSS
banana	banana

Use Query Params:
?username=maris | mar | ar | without anything

Get All Users Error

Not authorized - {message: "You are not my boss!"}
no banana {message: "no banana, no data"}

Get All Users OK

{message: "here you go boss, thanks for banana", data: [{id: 1, username: "marisklava", email:"test@test.com"}, {id: 2, username: "john dow", email:"test@test.com"}]}
Password is encrypted in database and excluded for security reasons

Group Work (5 minutes) 3-4 people

Check each others Postman setup and ask for advice if something did not work.

Used Coffee Routes

@desc	@route	@access
Create used coffee post	POST /api/v1/used-coffee	Private
Get Users coffee posts	GET /api/v1/used-coffee/my	Private / Post Owner
Get all used coffee posts	GET /api/v1/used-coffee/	Public
Get Single used coffee post	GET /api/v1/used-coffee/:id	Public
Update used coffee post	PUT /api/v1/used-coffee/:id	Private/ Post Owner
Buy used coffee	PATCH /api/v1/used-coffee/:id/buy	Private
Delete used coffee post	DELETE /api/v1/used-coffee/:id	Private/ Post Owner

Create Used Coffee Post

Create Used Coffee Post Instruction

Send POST request to correct route with body that would contain JSON format data
Optional fields - default if not provided
brand
image_link

Request needs to have headers:

key	value
token	Bearer token(Token received when logged in)
please	please

Correct Example: { name: "Black Coffee", brand: "STARBUCKS", "description": "It was Good Coffee", "pct_left": "50", price: "1.00" purchase_date: "31.12.2023"}

Create Used Coffee Post Error

wrong token - {message: "No user with token provided"}
no please - {message: "What is the magic word?"}

no name - {message: "Please provide name"}
no description - {message: "Please provide description"}
no price - {message: "Please provide price"}
no pct_left - {message: "Please provide pct_left"}
no purchase_date - {message: "Please provide purchase_date"}
cannot change sold to TRUE through body! - {message: "sold field cannot be modified"}

Create Used Coffee Post OK

{message: "Your used coffee could be someone else's treasure! Thank you for your post! ", data: new_coffee_post}

Get Users coffee posts

Get Users coffee posts Instruction

Send GET request to correct route

Request needs to have headers:

key	value
username	your-username
api-key	api-key received when logged in
banana	banana

Get Users coffee posts Error

wrong username or api-key - {message: "Username of Api key is incorrect"}
no banana - {message: "no banana, no data"}

Get Users coffee posts OK

{message: "Here you go! ", data: users_used_coffee_posts}

Get all used coffee posts posts

Get all used coffee posts posts Instruction

Send GET request to correct route

Use Query Params:
?name=turk
?brand=rih
?price=>1 | 1 | <1 (greater than 1 euro | 1 euro | less than 1 euro)

Get all used coffee posts posts Error

wrong route

Get all used coffee posts posts OK

{message: "British scientists say that used coffee is good for environment. ", data: all_used_coffee_posts}

Get Single used coffee post

Get Single used coffee post Instruction

Send GET request to correct route

Get Single used coffee post Error

wrong id - {message: "Used Coffee Id does not exist"}

Get Single used coffee post OK

{message: "Good Choice! Place Order now and regret later! ", data: used_coffee}

Update used coffee post

Update used coffee post Instruction

Send PUT request to correct route with body that would contain JSON format data
Optional fields - default if not provided
brand
image_link

Request needs to have headers:

key	value
X-DIRECTIVE-STUFF	X-DIRECTIVE-STUFF (received when logged in)
please	please

Correct Example: { name: "Latte", brand: "RihardinaCoffee", "description": "It was the worst Coffee", "pct_left": "99", price: "0.05" purchase_date: "yesterday"}

Update used coffee post Error

wrong id - {message: "Used Coffee Id does not exist"}
unauthorized - {message: "X-DIRECTIVE-STUFF is incorrect"}
please - {message: "what is magical word?"}

no name - {message: "Please provide name"}
no description - {message: "Please provide description"}
no price - {message: "Please provide price"}
no pct_left - {message: "Please provide pct_left"}
no purchase_date - {message: "Please provide purchase_date"}
cannot change sold to TRUE through body! - {message: "sold field cannot be modified"}

Update used coffee post OK

{message: "GREAT SUCCESS! ", data: updated_coffee}

Buy used coffee (update)

Buy used coffee (update) Instruction

Send PATCH request to correct route with body that would contain JSON format data
Optional fields - default if not provided
brand
image_link

Request needs to have headers:

key	value
username	your-username
api-key	api-key received when logged in
X-DIRECTIVE-STUFF	X-DIRECTIVE-STUFF (received when logged in)

Correct Example: { "sold": "TRUE"}

Buy used coffee (update) Error

wrong username or api-key - {message: "Username of Api key is incorrect"}
unauthorized - {message: "X-DIRECTIVE-STUFF is incorrect"}

cannot by users own coffee - {message: "cannot by your own used coffee. C'mon, I had more expectations from people who buy used coffee!"}
coffee has already been sold - {message: "cannot buy this coffee, someone else already got the best deal of his life!"}

Buy used coffee (update) OK

{message: "Sit, relax and enjoy your used coffee! ", data: bought_used_coffee}

Delete used coffee post

Delete used coffee post Instruction

Send GET request to correct route

Request needs to have headers:

key	value
token	token
banana	banana

Delete used coffee post Error

unauthorized - {message: "it is not your used coffee, hands off!"}
no banana - {message: "no banana, no delete"}

Delete used coffee post OK

{message: "Noooooo! What have you done! Used Coffee post deleted. I am sad. ", data: deleted_coffee_post}