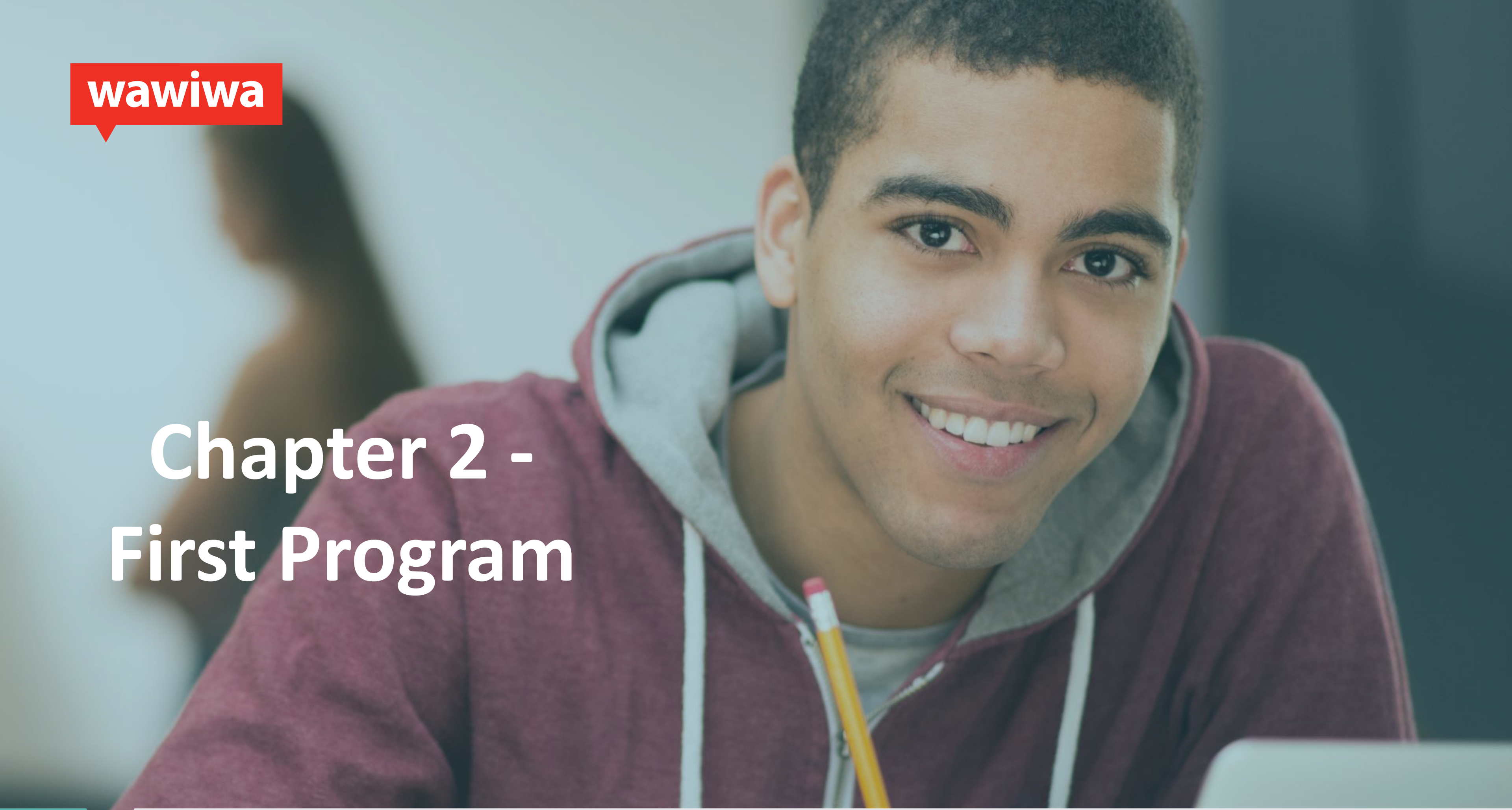
wawiwa

NODEJS Practice Booklet



wawiwa

Chapter 2 - First Program

do it yourself 1

Export a Name Module. It has 2 functions:

1. A function that receives 2 numbers num1 and num2 and returns num1*num2

2. A function that receives 3 numbers , num1, num2, num3 and return the average of them.

Create a program and use it.

wawiwa

Chapter 3 - First Server

wawiwa

Do it yourself 1

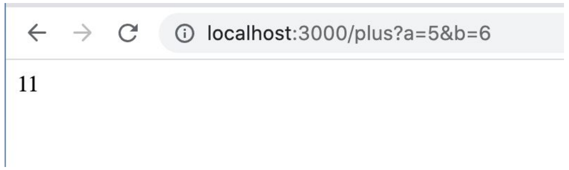
Create a server that responds to 127.0.0.1:3000 with your city name and your country name.

Do it yourself 2

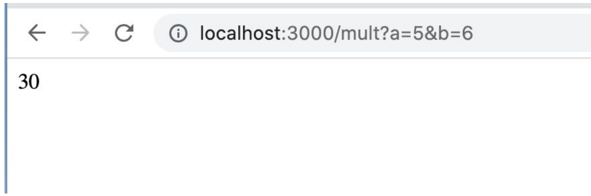
Build a server on port 3000 that responds to the user in the following way:

1. For the url ‘/city - return “Paris”
2. For the url ‘/country return “Romania”
3. For every other request responds with “I don’t know how to respond to that”

Do it yourself 3


localhost:3000/plus?a=5&b=6

```
11
```

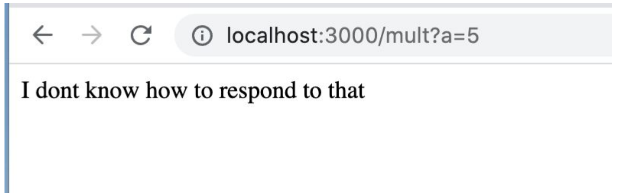

localhost:3000/mult?a=5&b=6

```
30
```

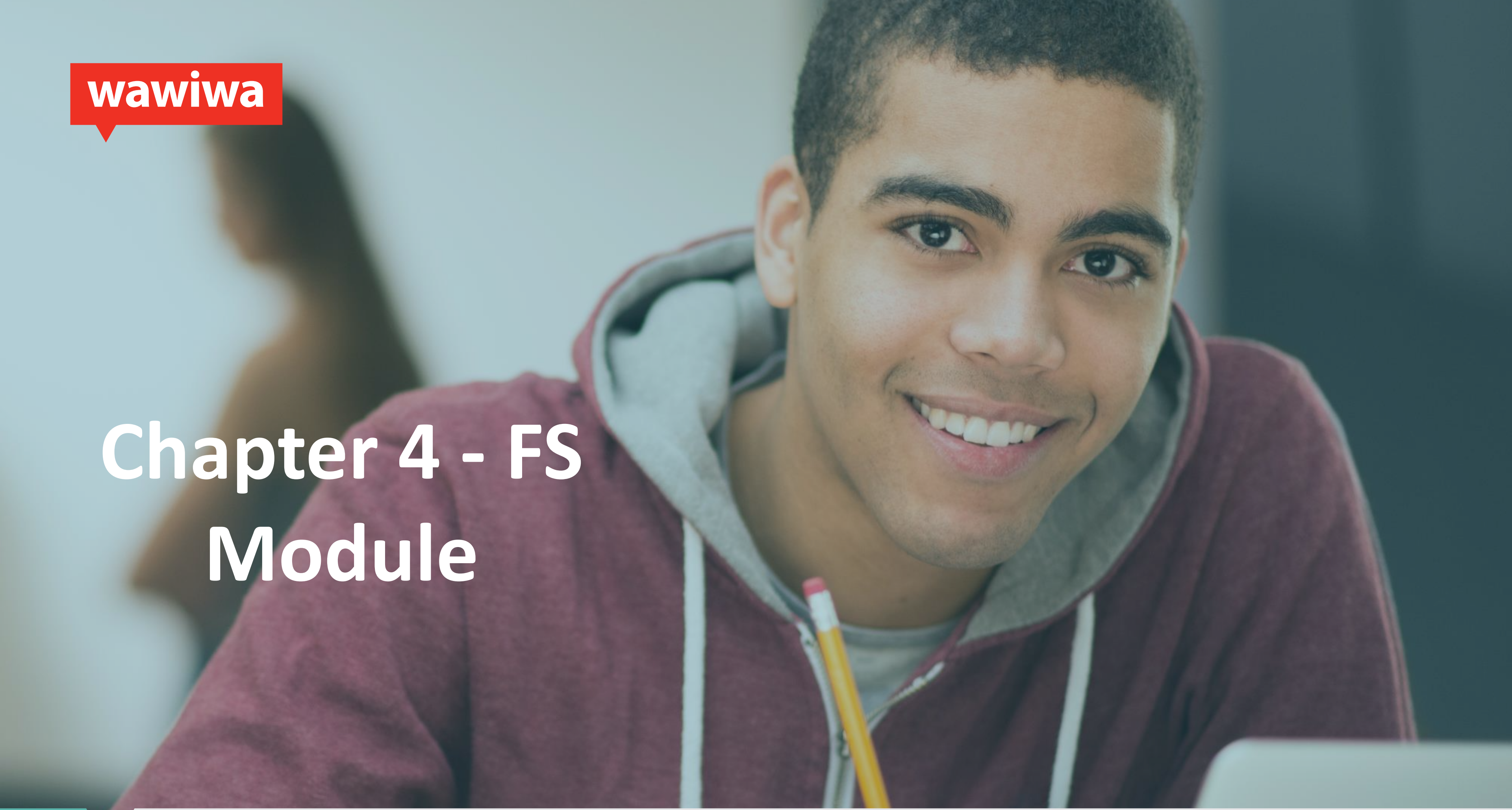
Build a server on port 3000 that responds to the user in the following way:

The server will receive two parameters a and b and

1. For the url /plus?a=5&b=6 - return sum of parameters
2. For the url /mult?a=5&b=6 - return multiplied parameter
3. For every other request responds with "I don't know how to respond to that"


localhost:3000/mult?a=5

```
I dont know how to respond to that
```



Chapter 4 - FS Module

wawiwa

Do it yourself 1

Create a program in node that writes **synchronously** to a file named test.txt with your city and country name

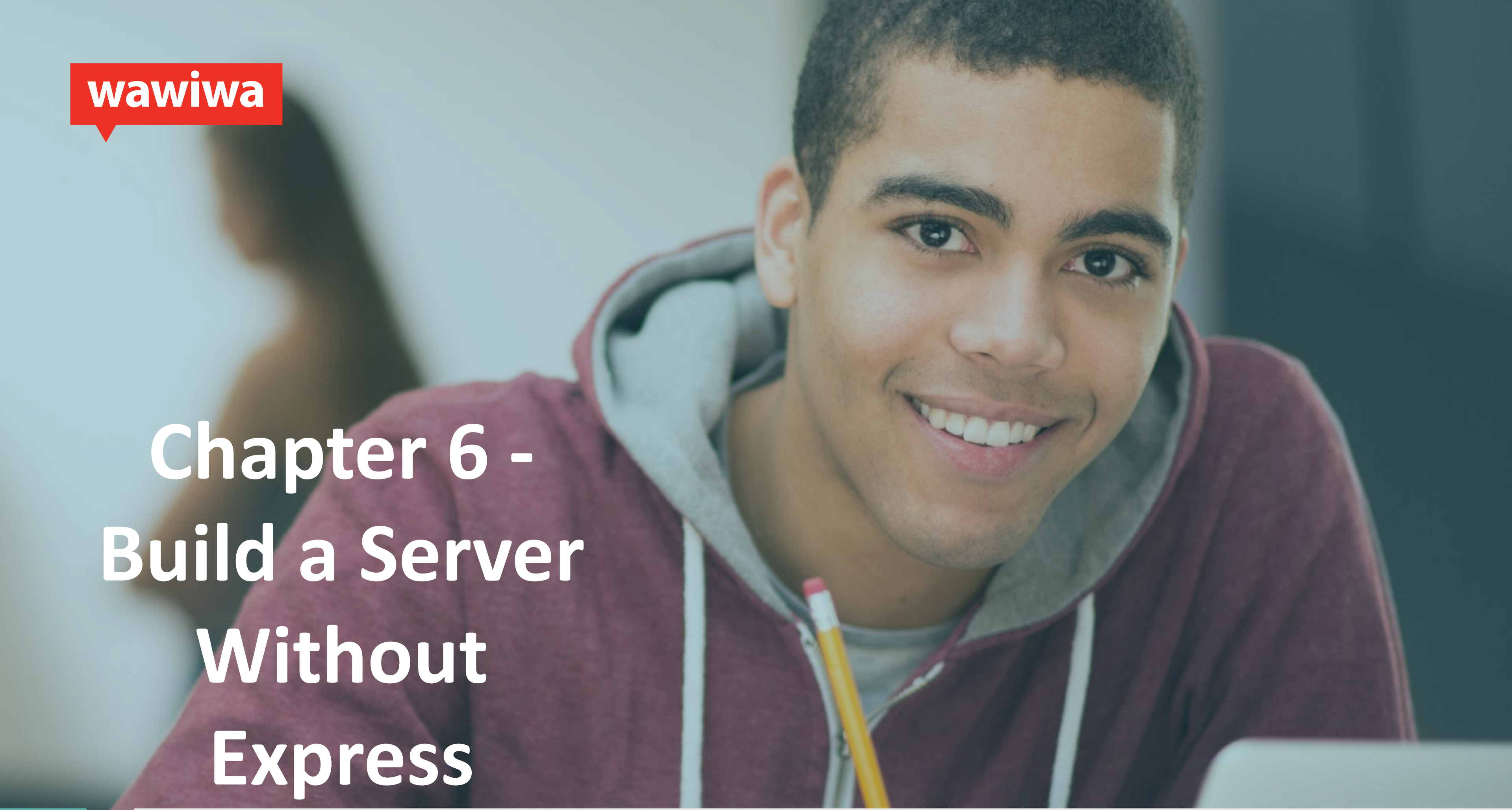
Do it yourself 2

Create a program in node that writes **asynchronously** to a file named test.txt with your city and country .

wawiwa

Do it yourself 3

Write a program that reads **synchronous** from a file named test.txt and prints the text to the console

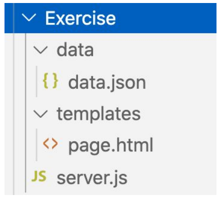


wawiwa

Chapter 6 - Build a Server Without Express

wawiwa

Add this file to data folder



Do it yourself 1

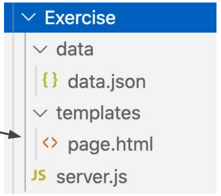
1. Create a folder with 2 directories
   a. data that contain data.json

```
[
    {
      "product": "Heart",
      "Description":"Pink Heart",
      "price": "50",
      "image": "💗"
    }
]
```

   b. templates - contain page.html - next slides content of page.html
1. Add to the folder file named server.js

wawiwa

Add this file to templates folder


Exercise
data
data.json
templates
page.html
server.js

page.html

```
<!DOCTYPE html>
<html lang="en">
 <head>
   <meta charset="UTF-8" />
   <meta name="viewport" content="width=device
   <meta http-equiv="X-UA-Compatible" content=
   <title>Products</title>
   <style>
     *
     {
       margin: 0;
       padding: 0;
       box-sizing: inherit;
     }

     html {
       box-sizing: border-box;
     }
```

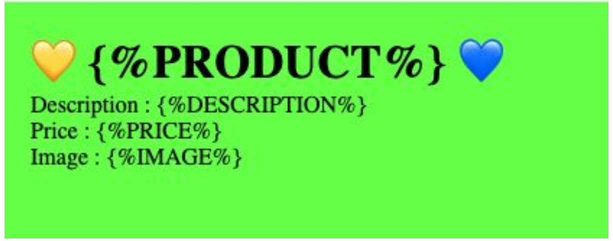
```
body {
  background:hsl(110, 100%, 64%);
}
.container{
  padding:20px;
}
.cards{ial-scale=1.0"  />
  padding: 5px;
}
.card{
  border: 5px solid royalblue;
  width: 25%;
  border-radius: 10px;
  padding: 5px;
  font-size: 20px;
  margin:10px 10px;
  float: left;
}
    </style>
  </head>
```

wawiwa

page.html continue

You can use any sign. (It is not mandatory to use the% sign)

```
 <body>
   <div class="container">
     <h1>💛 {%PRODUCT%}  💙</h1>

     <div>

       <p>Description : {%DESCRIPTION%} </p>

       <p>Price : {%PRICE%} </p>

       <p>Image : {%IMAGE%} </p>

     </div>

   </div>
 </body>
</html>
```

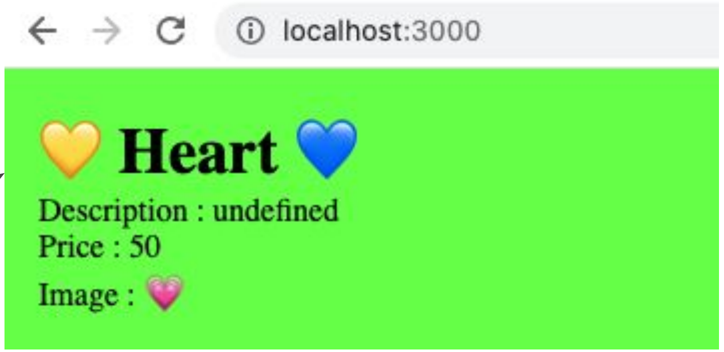


💛 {%PRODUCT%} 💙

Description : {%DESCRIPTION%}
Price : {%PRICE%}
Image : {%IMAGE%}

Exercise continue

Create a server that listens to port 3000 and responds with the relevant data.json.


localhost:3000

💛 Heart 💙

Description : undefined
Price : 50
Image : 💗

response

wawiwa

Chapter 9 - Build a Server Using Express

Do it yourself 1 - Build A Server Using Express

Create a server that responds to the following requests:

Request	Response	Method
localhost:3000/api/name	Your full name	get
localhost:3000/students/number	Random Number between 0 to 100	get
localhost:3000/courses/n1ton2	Random Number between 1000 to 2000	Post

wawiwa

do it yourself 2 - data/player.json

```
[
    {
        "id":0,
        "firstname":"Andre",
        "lastname":"Iguodala",
        "age":37,
        "Team":"Warriors"
    },
    {
         "id":1,
         "firstname":"Carmelo",
         "lastname":"Anthony",
         "age":37,
         "Team":"Lakers"
     }
]
```

Do it yourself 2 - Build A Server Using Express

Create an express server that responds for the following requests:

Request	Response	Method	Comment
`/api/v1/players`	All players	get	Get all players from players.json
`/api/v1/players`	The created player	Post	Create new Player
`/api/v1/players/:id`	Player by id	Get	Get player by id
`/api/v1/players/:id`	Update player	Patch	Update player by id
`/api/v1/players/:id`	delete player	Delete	Delete player by id

wawiwa

Do it yourself 3- refactor exercise 2

Create an express server that responds to the following requests:

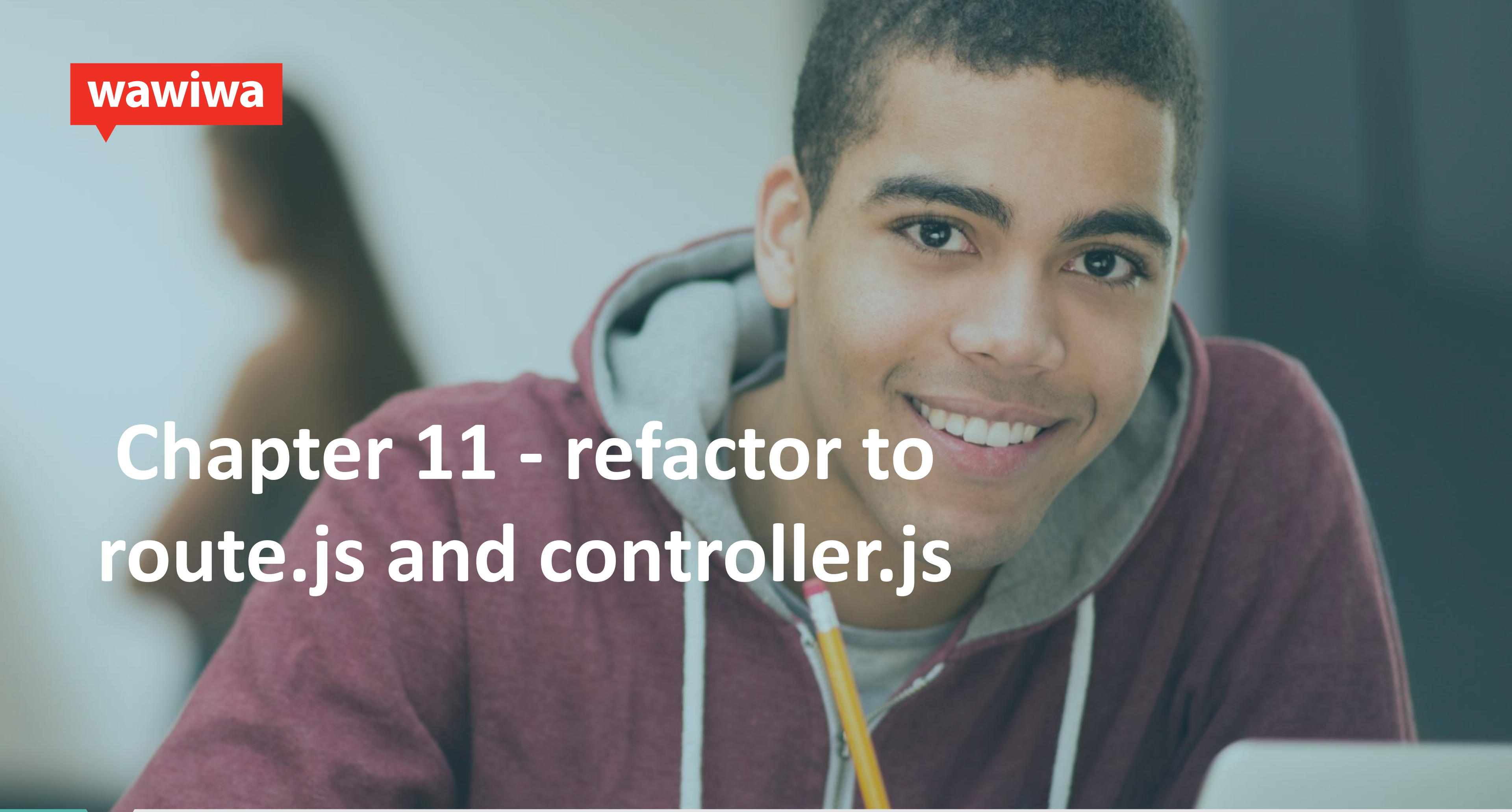
Request	Response	Method	Comment
`/api/v1/players`	All players	get	Get all players from players.json
`/api/v1/players`	The created player	Post	Create new Player
`/api/v1/players/:id`	Player by id	Get	Get player by id
`/api/v1/players/:id`	Update player	Patch	Update player by id
`/api/v1/players/:id`	delete player	Delete	Delete player by id

Chapter 10 - middleware

Do it yourself 1- Add middleware

Add middleware with console.log()

Request	Response	Method	Comment
`/api/v1/players`	All players	get	Get all players from players.json
`/api/v1/players`	The created player	Post	Create new Player
`/api/v1/players/:id`	Player by id	Get	Get player by id
`/api/v1/players/:id`	Update player	Patch	Update player by id
`/api/v1/players/:id`	delete player	Delete	Delete player by id

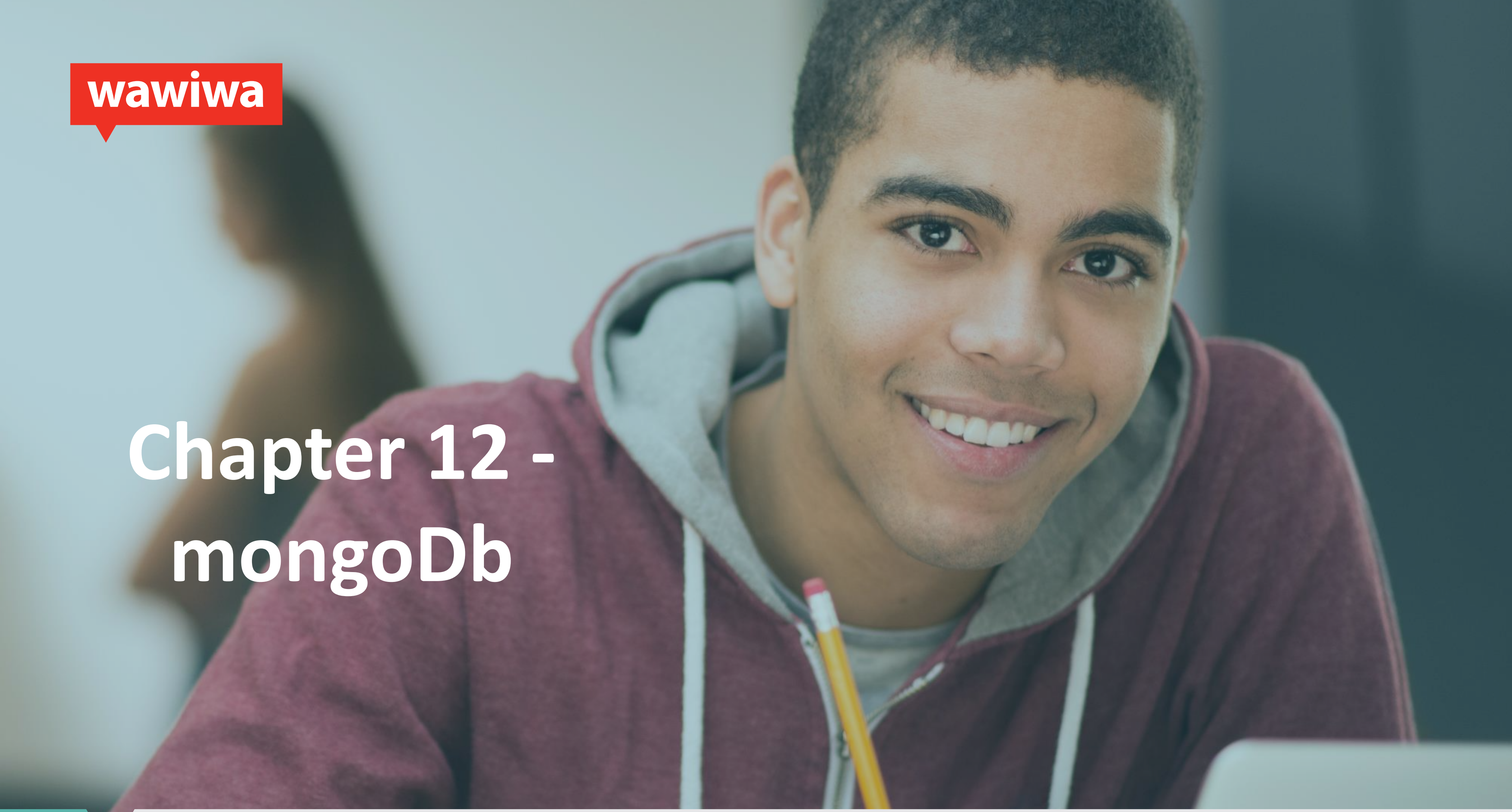


Chapter 11 - refactor to route.js and controller.js

wawiwa

Refactor exercise 3 chapter 3 - to router.js and controller.js

Request	Response	Method	Comment
`/api/v1/players`	All players	get	Get all players from players.json
`/api/v1/players`	The created player	Post	Create new Player
`/api/v1/players/:id`	Player by id	Get	Get player by id
`/api/v1/players/:id`	Update player	Patch	Update player by id
`/api/v1/players/:id`	delete player	Delete	Delete player by id



Chapter 12 - mongoDb

wawiwa

Do it yourself 1

The sequence available in the next slides (till the end of the chater) will only work if we perform all the steps described in the previous slides - connection to DB,try, Catch etc

1. Open an account in mongodb
2. After registering, make sure you have a user in the account's user list who has access to the account
3. Make sure your ip is in the ip list if not then add it
4. Create a js file named b1.js
5. Click connect and then connect your application and select nodejs version 4 Copy the code and paste it in b1.js Run the file by node b1

Do it yourself 2

```
Create a product object with the following attributes in the b1.js file:
   let product = {
       "title":  "ball" ,
       "description": "Big blue ball" ,
       "tags": [ "circle", "toy", "kids" ],
       "age": 12,
       "price": 20
   }
Add the correct commands to put this product into mongodb using the insert command
Run the b1 file and make sure a product logs in
```

wawiwa

Do it yourself 3

`Create 3 additional objects that describe different products in the b1.js file`

```
Add the correct commands to insert into mongodb using the insert command
Run the b1 file and make sure a product logs in
```

Do it yourself 4

Open a file named b4.js and write a suitable code that will print to the console all the records in mongodb
Help with the command:

```
 find().toArray();
```

Do it yourself 5

Open a file named b5.js and write a suitable code that will print to the console all the records that are in mongodb that cost 20.
Help with the command:

```
 find().toArray();
```

Do it yourself 6

Open a file named b6.js and write a suitable code that will print to the console all the records that are in mongodb that cost more than 20.
Help with the command:

```
 find().toArray();
```

wawiwa

Do it yourself 7

Open a file named b7.js and write a suitable code that will print to the console all the records that are in mongodb that cost less than 20.
Help with the command:

```
 find().toArray();
```

Do it yourself 8

Open a file named b8.js and write a suitable code that will print to the console all the records that are in mongodb that cost between 20 and 40.
Help with the command:

```
 find().toArray();
```

wawiwa

Do it yourself 9

Open a file called b9.js and write a suitable code that will print to the console all the records that are in mongodb that are priced larger than 20 and are also suitable for ages 12
Help with the command:

```
 find().toArray();
```

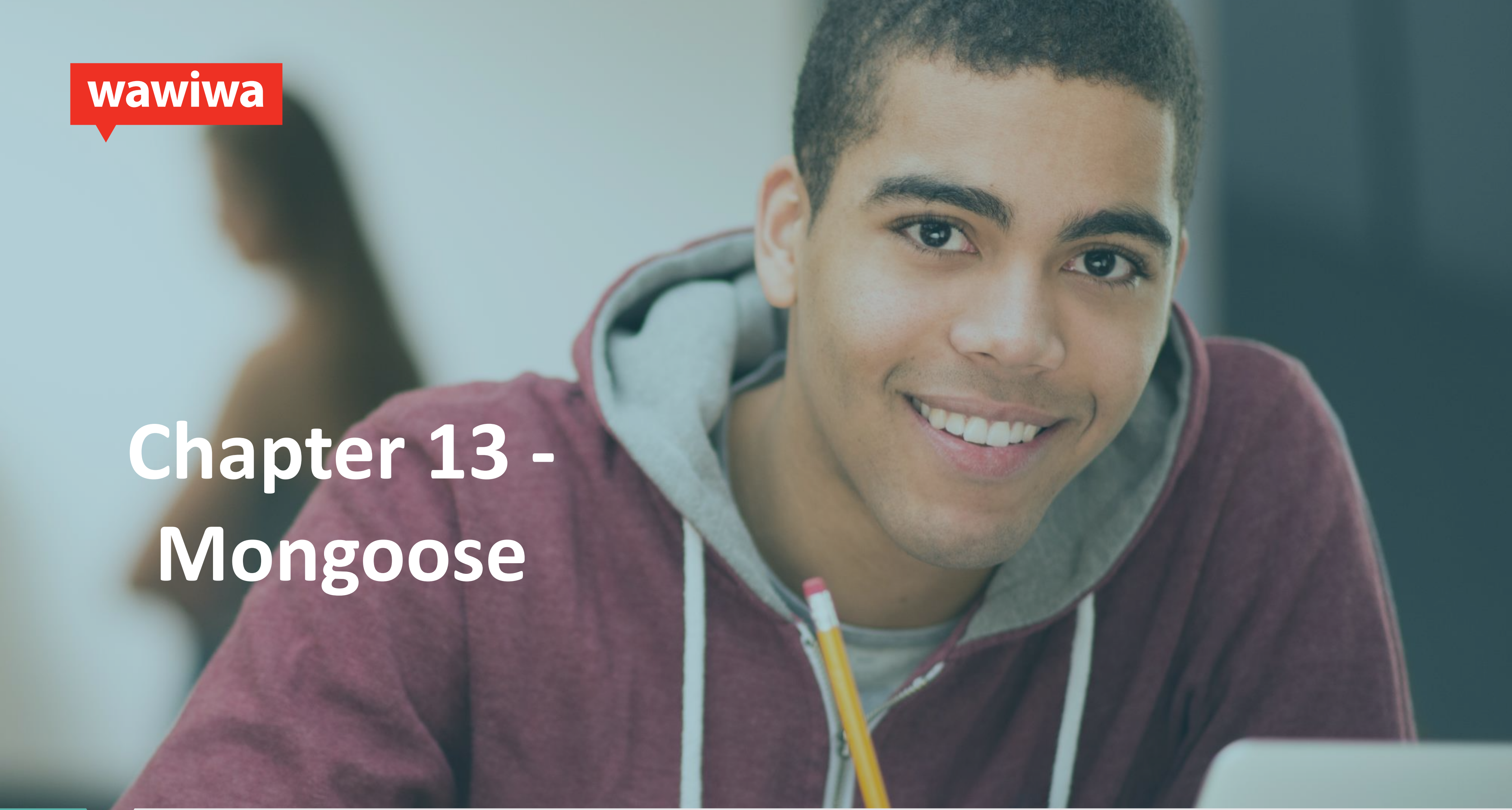
Do it yourself 10

Open a file named b10.js and write a suitable code that will print to the console all the records, limit the number of records to 2.
Help with the command:

```
find().toArray();
```

Do it yourself 11

Open a file named b11.js and write an appropriate code that will update the record that cost 20 to 30 Help with updateOne:



wawiwa

Chapter 13 - Mongoose

Do it yourself 1

1. Create a folder that contains an app.js file
2. Add to the folder a model named PersonModel.js that contains the following fields: first name, family, city, country, salary
3. Add a new function that insert person to db
4. Run the application and insert 3 persons to db
5. Check in mongodb that the entries have been entered

Do it yourself 2

1. Add a new function to app.js to retrieve the data from the db
2. Rerun the application and navigate to localhost:3000/api/v1/products

wawiwa

Do it yourself 3

1. Add a new function to app.js to retrieve the person by _id from the db
2. Rerun the application and navigate to localhost:3000/api/v1/products/_ _ _ _ _ _ _ _ _ _ _ (select an id)

wawiwa

Do it yourself 4

1. Add a new function to app.js to update the person by _id
2. Rerun the application and navigate to localhost:3000/api/v1/products/_ _ _ _ _ _ _ _ _ _ _ (select an id and run from postman patch request and update the salary to 10000 for id selected)

Do it yourself 5

1. Add a new function to app.js to delete person by _id
2. Rerun the application and navigate to localhost:3000/api/v1/products/_ _ _ _ _ _ _ _ _ _ _ (select an id and run from postman delete request and delete the selected person)

Do it yourself 6

1. Update the function from exercise 2 and to filter data from db.
2. Rerun the application and navigate to localhost:3000/api/v1/persons/ and retrieve all person with salary 1000
3. Rerun the application and navigate to localhost:3000/api/v1/persons/ and retrieve all person with salary firstname Mike

wawiwa

Do it yourself 7

1. Update the function from exercise 6 and to advance filter data from db.
2. Rerun the application and navigate to localhost:3000/api/v1/persons/ and retrieve all persons with salary above 1000
3. Rerun the application and navigate to localhost:3000/api/v1/persons/ and add parameters to the query that retrieve all persons with salary below 1000
4. Rerun the application and navigate to localhost:3000/api/v1/persons/ and add parameters to the query that retrieve all persons with salary between 1000 to 2000

wawiwa

Do it yourself 8

1. Update the function from exercise 6 and to and add the option to sort.
2. Rerun the application and navigate to localhost:3000/api/v1/persons/ and retrieve all persons sorting according to salary

Do it yourself 9

1. Add a function that displays statistics about people. The function will return the average salary, the minimum salary and the maximum salary
2. Rerun the application and navigate to localhost:3000/api/v1/get/statistic/ and retrieve the average, max salary and max salary