A red text on a black background

Description automatically generated

Service Layer

Elite Wrestling Athletic Club

07/16/2024

Christopher Mullins

Overview

The backend service for my web application will be built using Express. This will be a REST API that will be called in order to call the data. This will be broken down into a few layers:

Route: This defines the paths that will be used to get the data

Controller: Routes will point to the controllers, then handles the incoming HTTP request and returns a response back.

Service: The Controller will call the service method and handle how the data being requested. This will actually deal with the database.

Routes:

The following list is the routes and endpoints for the service layers. I will use a placeholder for the link but will be using AWS when hosting.

Product Routes:

Get all products:

Method: GET

[URL: https://aws.elitewrestling.com/api/products](URL:%20https://aws.elitewrestling.com/api/products)

Purpose: When a user accesses the store the endpoint will be called and data returned to the products page.

Example request:

curl -X 'GET' \

' https://aws.elitewrestling.com/api/products’

Successful response:

A computer screen with white text

Description automatically generated

Error Response:

400 bad input parameter

Diagram:

Get a single product:

Method: GET

URL: <https://aws.elitewrestling.com/api/products/:id>

Purpose: When the user selects an item the endpoint will then redirect to the item detail page

Example request:

curl -X 'GET' \

‘https://aws.elitewrestling.com/api/products/d290f1ee-6c54-4b01-90e6-d701748f0851’

Example Respones:

A computer screen with white text

Description automatically generated

Error response:

400 bad input parameters

Diagram:

Search Product by Name:

Method: GET

URL: <https://aws.elitewrestling.com/api/products/:name>

Purpose: When the user enters a product name in the search box the endpoint will filter the results to products that match the search field.

Example request:

curl -X 'GET' \

'<https://aws.elitewrestling.com/api/products/Nike%20Shoe>’

Successful response:

A computer screen with white text

Description automatically generated

Error response:

400 bad input parameters

Diagram:

Create a new Product:

Method: POST

URL : <https://aws.elitewrestling.com/api/products/createProduct>

Purpose: When a staff member clicks on the add product button the endpoint will be called and post the information to the database.

Example request:

curl -X 'POST' \

<https://aws.elitewrestling.com/api/products/createProduct>’ \

-H 'accept: \*/\*' \

-H 'Content-Type: application/json' \

-d '{

"id": "d528",

"name": "Adidas Shoe",

"qty": 4,

"price": 59.99,

"availability": true,

"sizes": [

9,

10,

10.5,

12

]

}'

Example response:

A screenshot of a computer

Description automatically generated

Error response:

400 invalid input, object invalid

409 an existing item already exists

Update a product record:

Method: PUT

URL: <https://aws.elitewrestling.com/api/products/:id>

Purpose: When a staff member clicks on the edit button for a product the endpoint should call the product and put the affected changes in the database.

Example request:

curl -X 'PUT' \

<https://aws.elitewrestling.com/api/products/createProduct>’ \

-H 'accept: \*/\*' \

-H 'Content-Type: application/json' \

-d '{

"id": "d528",

"name": "Adidas Shoe",

"qty": 4,

"price": 59.99,

"availability": true,

"sizes": [

9,

10,

10.5,

12

]

}'

Example response:

A screenshot of a computer

Description automatically generated

Error response:

400 Invalid Id

404 product not found

405 validation exception

Delete a product:

Method: DELETE

URL: <https://aws.elitewrestling.com/api/products/:id>

Purpose: The staff member can click on the delete icon and the endpoint makes the call to delete the item from the database.

Example request:

curl -X 'DELETE' \

'https://aws.elitewrestling.com/api/products/5’

Example response:

A screenshot of a computer

Description automatically generated

Error resoonse:

400 Invalid ID supplied

404 Item not found

Diagram: