Eric Ferrer

CS 496 HW3pt.B

10/23/2016

An explanation of the URL structure used to access resources

To access the welcome page of the site you go to http://162.243.137.191:3000/

There are three different table so to access each table you would enter the following:

http://162.243.137.191:3000/api/recipes

http://162.243.137.191:3000/api/logs

http://162.243.137.191:3000/api/users

I used postman for this assignment and to get, put, patch, and delete a particular id

you would have the url such as that the following:

http://162.243.137.191:3000/api/logs/580d47619ed5ce80135ef492

and you could use postman to delete the field, edit the whole field or just a particular one and delete it.

A description of which RESTful constraints you met and which you did not (you do not need to make a RESTful app, but you do need to know why it does not meet all the constraints of a RESTful API)

Client-Server: This constraint has been made as there is a clearn separation of client-server architecture. When a user of the API makes a request then the server either rejects or allows the request and sends a corresponding response back to the client. This is made clear when you enter the following url: http://162.243.137.191:3000/api/recipes.

Stateless: The stateless constraint has also been made as no information from the user is being stored when the user is sending requesting or when a response is being sent back to the user.

Cacheable: I did not add anything to my codebase that would have made this api cachable so I would say it is not cachable.

Layered System: While this API doesn't have many layers to go to many other servers because the scope of this project is so small, it still fits the constraint of a layered system as the user still doesn't know how many layers there are to get the actual response.

Uniform Interface: Yes this meets the constraint. This clearly meets the constraint as there resources that are identified and representations that are returned in JSON format.

A discussion of any changes you needed to make to the schema from last week

I ended up making no changes to my schema design from the previous week. The only change was that I used mongoose instead of plain ole MongoDb.

Things Done Differently

I wish I chose a different database other than MongoDb, it's hard to tell if everything in the many relationship was deleted correctly. When I deleted an item log then the id would stay in the user and other times it would stick with an empty array. I can delete recipes using CURL just fine but in postman I've been getting a syntax error that I just can't figure out which is unexpected token "n" so I am submitting everything as is. I would say 95% of the code works except for deleting the many portion of the logs/user but I should get it fixed shortly after the deadline passes.

Testing

Bash Script

```
#!/bin/bash

CURL='/usr/bin/curl'

GRECIPE="162.243.137.191:3000/api/recipes/"

GLOGS="162.243.137.191:3000/api/logs/"

GUSERS="162.243.137.191:3000/api/users/"

URL="580c0aa3b58e52704ede057c"

#CURLARGS="-f -s -S -k"




# or you can redirect it into a file:

$CURL  $GRECIPE >> output.txt

echo ""

$CURL  $LOGS >> output.txt

echo "" >> output.txt

$CURL  $GUSERS >> output.txt

echo "" >> output.txt

echo "" >> output.txt

echo "TESTING POST ON RECIPES" >> output.txt

$CURL -H "Content-Type: application/json" -X POST -d '{"name": "Chicken Fry", "ingredients":"chicken, fry", "url": "budgetbytes.com"}' $GRECIPE

#curl 162.243.137.191:3000/api/recipes
```

#URL -H "Content-type: application/json" -X PUT -d '{"name": "Chicken Fry", "ingredients":"chicken, fry", "url": "budgetbytes.com"}' $GRECIPE

echo "GET PARTICULAR ID" >> output.txt

$CURL 162.243.137.191:3000/api/recipes/580c0aa3b58e52704ede057c >> output.txt

echo "EDIT THE FIELDS OF THAT ID" >> output.txt

$CURL -X PUT -H "Content-type: application/json" -d '{"name": "daaaaaaaaaaank mexican tacos", "ingredients":"so much gooooooodness", "url": "budgetbytes.com/dddanktacos"}' 162.243.137.191:3000/api/recipes/580c0aa3b58e52704ede057c >> output.txt

$CURL $GRECIPE >> output.txt

curl -X DELETE 162.243.137.191:3000/api/recipes/580d9d8842cc3219d19f4a22 >> output.txt

Result of bash script:

[{"_id":"5809647d5b9d664d3ec13383","name":"Dragon Noodles","ingredients":"Noodles","url":"http://www.budgetbytes.com/2012/08/spicy-noodles/"},{"_id":"5809647d5b9d664d3ec13384","name":"Tikka masala","ingredients":"Chicken, sauce","url":"www.budgetbytes.com/2015/12/slow-cooker-chicken-tikka-masala"},{"_id":"580a8712d75e8f4ede7e41cb","name":"meows","ingredients":"pollo","url":"budgetbytes","__v":0},{"_id":"580d66b77d3a60151fc30e85","ingredients":"pollo","__v":0},{"_id":"580d916842cc3219d19f4a1a","name":"Chicken Fry","ingredients":"chicken, fry","url":"budgetbytes.com","__v":0},{"_id":"580d958142cc3219d19f4a1b","name":"Chicken Fry","ingredients":"chicken, fry","url":"budgetbytes.com","__v":0},{"_id":"580d966642cc3219d19f4a1c","name":"Chicken Fry","ingredients":"chicken, fry","url":"budgetbytes.com","__v":0},{"_id":"580d97c642cc3219d19f4a1e","name":"Chicken Fry","ingredients":"chicken, fry","url":"budgetbytes.com","__v":0},{"_id":"580d97b442cc3219d19f4a1d","name":"Chicken Fritters","ingredients":"chicken, moe chick","url":"budgetbytes.com/ds","__v":0},{"_id":"580d9a7742cc3219d19f4a1f","name":"Chicken Fry","ingredients":"chicken, fry","url":"budgetbytes.com","__v":0},{"_id":"580c0aa3b58e52704ede057c","name":"daaaaaaaaaaank mexican tacos","ingredients":"so much gooooooodness","url":"budgetbytes.com/dddanktacos","__v":0},{"_id":"580d9bba42cc3219d19f4a20","n

ame":"Chicken Fry","ingredients":"chicken,
fry","url":"budgetbytes.com","__v":0},{"_id":"580d9ca242cc3219d19f4a21","name":"Chicken
Fry","ingredients":"chicken,
fry","url":"budgetbytes.com","__v":0},{"_id":"580d9d8b42cc3219d19f4a23","name":"Chicken
Fry","ingredients":"chicken,
fry","url":"budgetbytes.com","__v":0},{"_id":"580d9e0242cc3219d19f4a24","name":"Chicken
Fry","ingredients":"chicken,
fry","url":"budgetbytes.com","__v":0},{"_id":"580d9eb242cc3219d19f4a25","name":"Chicken
Fry","ingredients":"chicken,
fry","url":"budgetbytes.com","__v":0},{"_id":"580da0da42cc3219d19f4a26","name":"Chicken
Fry","ingredients":"chicken,
fry","url":"budgetbytes.com","__v":0},{"_id":"580da0f842cc3219d19f4a27","name":"Chicken
Fry","ingredients":"chicken,
fry","url":"budgetbytes.com","__v":0},{"_id":"580da17742cc3219d19f4a28","name":"Chicken
Fry","ingredients":"chicken,
fry","url":"budgetbytes.com","__v":0},{"_id":"580da21042cc3219d19f4a29","name":"Chicken
Fry","ingredients":"chicken,
fry","url":"budgetbytes.com","__v":0},{"_id":"580da23842cc3219d19f4a2a","name":"Chicken
Fry","ingredients":"chicken, fry","url":"budgetbytes.com","__v":0}]

[]

TESTING POST ON RECIPES

GET PARTICULAR ID

{"_id":"580c0aa3b58e52704ede057c","name":"daaaaaaaaaaank mexican tacos","ingredients":"so much
goooooodness","url":"budgetbytes.com/dddanktacos","__v":0}EDIT THE FIELDS OF THAT ID

I didn’t have enough time to finish the bash script but I did a lot of testing on postman as well and just took some curl scripts from there as well. This is what happened when I deleted a log that was part of the many relationship to the user. You can the 204 response of no content means the item was

successfully deleted.