Tommy Las Z23517623 Tommy-Las

March 13th 2022 [Github Repository](https://github.com/COT4930-Spring-2022-Fullstack-Web/hw-4-Tommy-Las)

Full Stack Web Development COT4930

Homework #4

Part 1 :

For the first part, I had to copy the studentserver.js program from chapter 05 example from the textbook. Then I had to do a GET request using a record id, a GET request using first & last name, a POST request to add a student, a PUT request to update a student’s information giving a record\_id, and a DELETE request to remove a student from the system based on the record\_id. This requests I did them with the Postman software, at first I couldn’t make the POST request work, but after trying a couple of times I was able to do it. First, I need to run the server.



Then I went to postman. Postman is very useful because it lets you do request in a very simple way. I set the request to POST, added the URL, and in the body raw I added the data as a JSON object. It shows on the bottom the record\_id and that it was succesfuly created. The student was created as a JSON file in the students folder in the server:

Graphical user interface, text, application, email

Description automatically generated

I created another student to test: Graphical user interface, text, application, email

Description automatically generated

I modified the POST request code in order to prevent the user to create a student that is already in the system. I created a function called searchStudent that searches the files to check for duplicates, if it finds an existing user with the same first and last name from input, then the user already exists in the system. The function returns an object with the student information if it is found, if it is not found it returns an empty object. I’m going to use this function later for my GET request: Text

Description automatically generated

I do a POST request using a first name and a last name that is already in the system. It displays the error message.

Graphical user interface, text, application, email

Description automatically generated

This is not required, but I show it because this GET request has no parameters, so it displays ALL students in the system:

Graphical user interface, text, application, email

Description automatically generated

This next GET request is also for /students, but since it is given parameters, then it is going to display the student based on the given first & last name. I had to modify the GET function for this. What I did was to check whether there were given any parameters by checking if the data object was empty or not. If it is not empty, then it is going to search through all the files and find the student with the given first and last name

Text

Description automatically generated Graphical user interface, text, application, email

Description automatically generated

This is a GET request using a record id. The record id needs to be in the URL after students/ . This student was added later so this is why is not included in previous screenshots: Graphical user interface, text, application, email

Description automatically generated

GET (student not found):

Graphical user interface, application

Description automatically generated

For the DELETE request you need to include the record id you want to be removed in the URL:

Graphical user interface, text, application, email

Description automatically generated

DELETED (GET all students):

Graphical user interface, text, application, email

Description automatically generated

This PUT requests replaces the information of a student. As in the DELETE request, you need to add the record id of the student you want to update in the URL:

Graphical user interface, text, application, email

Description automatically generated

GET all students (show it was replaced):

Graphical user interface, text, application, email

Description automatically generated

Part 2:

For this part, I had to program a front end that interacts with the backend. First, I created a main HTML with all the requests that you can make. You input your data in the request you want to make, you hit submit and then on the bottom right it will be displayed the information of a student, an error message or a success message:

Graphical user interface

Description automatically generated

I used AJAX in order to make each of the requests. For example, if you want to delete a student, you add the record id in the input box, and you hit the submit button. The button is going to run a function called deleteStudent, located in the script.js file, and it will send a DELETE request to the server using ajax.The examples of the book helped me and guided me on how to make this whole interaction work. The html file, css file and the script file MUST be in the public folder in the server, if not this would not work. In the script file, I created functions for each request.

Text

Description automatically generated

For example, this function adds a student to the server, which means that it sends a POST request to http://localhost:5678/students. First I need to grab the data that the user inputs in the form. I did that using jQuery. Then, in the AJAX method, I added the URL where the POST request is sent, the type of request which is POST, and the data for the server to handle. The success function has the server response as parameter and displays a success message. The response in this case contains the record id, but in other functions it contains the information from a student. The error function displays an error message if an error exists. All other functions were made similarly to this function.

This assignment was very helpful because it taught me to create each of the requests (POST, GET, DELETE, POST) and how to handle the data. It also taught me how as a user send requests to the server, in this case I used Postman. I also learned how to create a HTML that can interact with the server, front end (client side) interacting with the backend (server side). Also, I utilized for the first time JSDoc, a way to describe javascript functions in a program