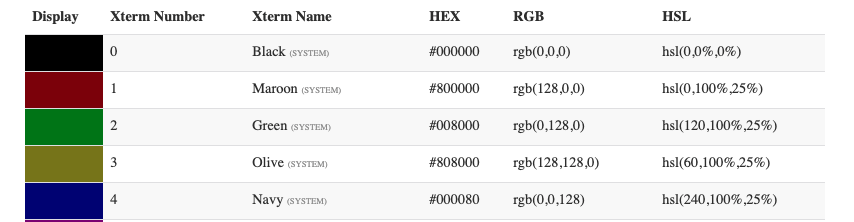
# **Laboratory 4 (Compulsory) – Express.js**

* This lab will be evaluated and averaged for calculating your final score for the lab part.
* This lab corresponds to 25% of the final score of the labs.
* This lab has to be submitted before Monday, 25th April at 23:00.
* Submissions after that day will be penalized with a 10% discount per late day. And no assignments will be accepted after Saturday, 30th April at 23:00.

This assignment is about using **Express.js** for managing an application that performs operations related to colours. The file with all the 256 colours can be downloaded from the Colours URL: <https://jonasjacek.github.io/colors/> The exact links for the JSON object is: <https://www.ditig.com/downloads/256-colors.json>



The file consists of an array of 256 elements where each element has the following fields:

* **colorId:** The number of colours on the list.
* **hexString:** The value in hexadecimal number
* **RGB:** The code of the colour in RGB format.
* **Hsl:** The code of the colour of HSL format.
* **Name:** The name of the colour in plain English.

For example, the black colour has: *{"colorId":0, "hexString":"#000000", "rgb": {"r":0,"g":0,"b":0}, "hsl": {"h":0,"s":0,"l":0}, "name":"Black"}.*

To accomplish the assignment, you will need to create a server in Express.js that will read from a file an object in JSON format which is an array with many fields about colours. To this end use the methods: GET, POST, PUT, and DELETE.

The assignment consists of the following points:

1.- **(1 pts)** Please, upload your project inside of a ZIP file with your name. For example, “LuisMiralles(T412…8).zip”. Also, add a 1-page max explaining which points have been accomplished and how to execute your code or if there is something that has to be considered. You can call it README.docx.

2.- **(3.5 pts)** Create a server using **Express.js** that loads the folder “data.json” and save it in an object called colours. The file can be downloaded manually from the website and stored in a folder from which Express.js will access to it. Then, you will have to implement the routing and middleware to properly attend to the following requests from external petitions. Create the code for creating an API able to respond to the following queries. Remember that the server has to update the changes in a file so it does not lose the changes.

The idea is that with the browser the user can type the address localhost:8080/index.html and download a page as shown in Figure 1 that will have an interface to interact and perform the main operations in an API.

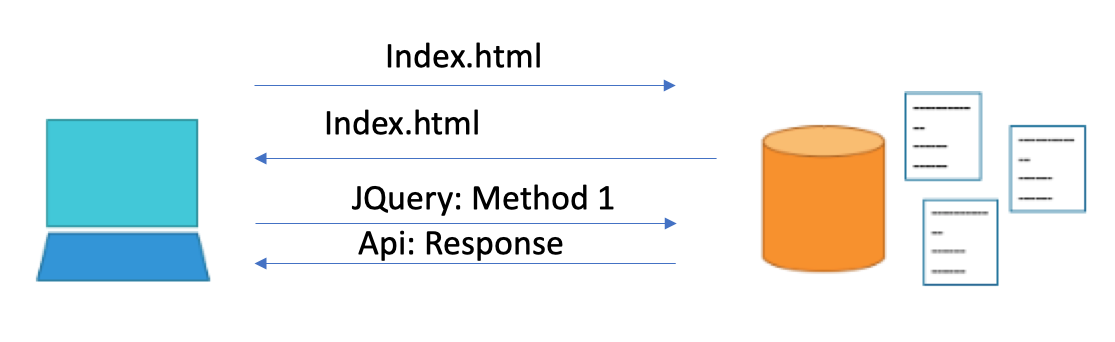


Fig 1: Communication between client and server.

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **URI** | **Details** | **Function** |
| GET | /colours | Safe, cacheable | Gets the list of all colours and their details |
| GET | /colours/123 | Safe, cacheable | Gets the details of colour id 123 |
| POST | /colours | N/A | Creates a new colour with the details provided. Response contains the URI for this newly created resource. |
| PUT | /colours/123 | Idempotent | Modifies colour id 123 (creates one if it doesn't already exist). Response contains the URI for this newly created resource. |
| DELETE | /colours/123 | Idempotent | Colour id 123 should be deleted, if it exists. Response should contain the status of the request. |
| DELETE or PUT | /colours | Invalid | Should be invalid. DELETE and PUT should specify which resource they are working on. |

3.- **(3.5 pts)** Create a website called Index.html with different methods that allow interacting with the server and test that it works correctly. The website can be implemented as in Figure 2. Please, note that the suggested interface is just an example. Each student can implement the interface in its own way. If the users type a wrong address, send him a message indicating that an error has been made and displaying a link to access the main page.

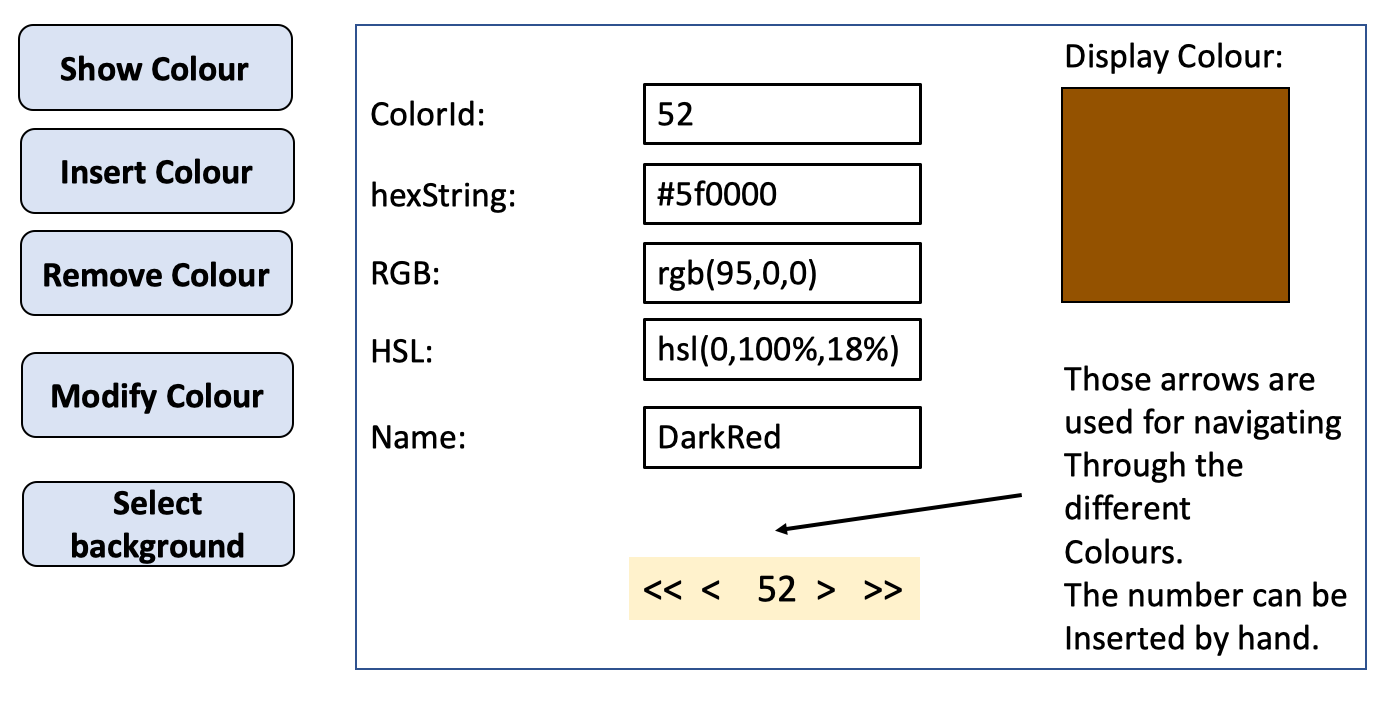


Figure 2: Example of a possible (not compulsory) interface.

4.- **(2 pts)** Cookies: You can implement a cookie functionality in which the selected background is remembered when the user refreshes the webpage by using cookies. For example, if the user selects for background colour “green”, if it closes the browser and accesses again, the colour green will come up. The same applies to the index in which the user was in the last session. Rather than starting in 1 when pressing “Show colour”, start with the last colour in which the user was. (See how to implement this in the references).

**References:**

https://www.tutorialspoint.com/expressjs/index.htm

https://www.tutorialspoint.com/expressjs/expressjs\_cookies.htm