

Web development

Command subsystem

Adam Bouchaala,
Teaching Fellow
Imperial College London
a.bouchaala@imperial.ac.uk
London, United Kingdom

Outline

- 01** Web app: definition
 - 02** Web app architecture
 - 03** Examples of Web Framework
 - 04** HTML/CSS/JavaScript
 - 05** React/Node/Express
-

Web applications: definition

Definition:

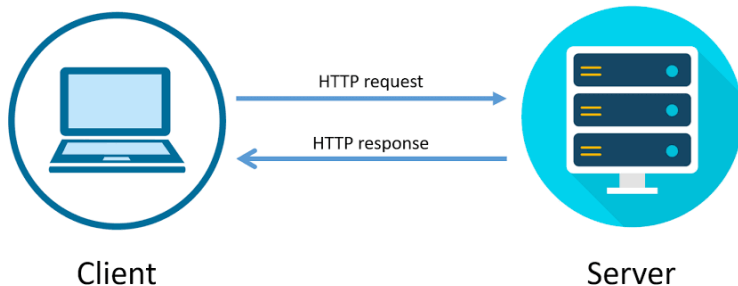
A web application is an application program stored on a remote server and delivered over the Internet through a browser interface. It refers to any program that can be accessed over a network connection using HTTP instead of existing in a device's memory. Web applications primarily run in a web browser.



HTTP

HTTP requests :

On the web, clients, like your browser, communicate with web servers using the HTTP protocol. This protocol controls how the client formulates its requests and how the server responds to them. The HTTP protocol knows different request methods.



Client:

Clients are anything that sends requests to the back-end. They are often browsers that display websites to the end user.

Server:

A server is simply a computer that listens for incoming requests. Although there are machines designed and optimized for this particular purpose, any computer connected to a network can act as a server. In fact, you will often use your own computer as a server when developing applications.

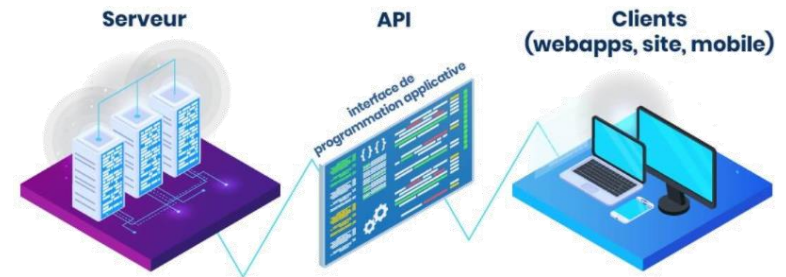
Architecture of a web application

Front-end:

In web development, the notion of "front end" refers to all the elements visible and accessible directly on a website (or even on a web application or a mobile web application).

Back-end:

The back-end is the set of technology needed to process the incoming request and generate and send the response to the client. It usually consists of three main parts: the server, the application, and the database.



What is an API ?

API:

An API is a communication tool that serves as an interface between two different applications so that they can communicate with each other.

REST API methods:

GET: Retrieve data from API

POST: Create a resource in the api

PUT: Update API resources

DELETE: Delete API data

Different types of APIs:

REST: the simplest and most used model in IOT (our case)

SOAP: is a standard protocol originally designed so that applications developed with different languages on different platforms can communicate.

REST API

REST API Model



Programming languages

Front-end

The most popular front-end programming languages are :

- **HTML**
- **CSS**
- **Javascript**
- **React**
- **Angular**
- **Vue**

Back-end

The most popular back-end programming languages are :

- **Node(Express/Nest)**
- **PHP(Laravel/Symphony)**
- **Java(JEE/Spring boot)**
- **.NET**
- **Python(Flask/Django)**

HTML

HTML:

- HTML **Hyper Text Markup Language** was born in 1989 under the impulse of Tim Berners Lee, "inventor" of the Web.
- It contains **commands**, implemented by **tags** to mark different types of text (titles, paragraph, lists...), to include images, forms, links...
- It is a **markup** language that describes the logical structure of a **hypertext** document. It was deliberately designed to be simple.
- All web browsers understand HTML

Tags:

- **HTML tags** are the building blocks of HTML coding. They are used to format a text, structure and prioritize the content of a page. Tags also tell the browser how to display the page in question. For the Internet user, they are invisible unless he displays the source code.

HTML

Some types of HTML tags:

`<html> – </html>`: main tag of all web pages.

`<head> – </head>` : page header

`<body> – </body>` : body of the page

`<h1> – <h6>` : title of different levels

`` : image using src (image address) and alt (alt text) attributes

`<p> – </p>`: paragraph

`<div> – </div>` : >: generic block type tag. The `<div>` tag defines a division or section in an HTML document. The `<div>` tag is used as a container for HTML elements - which are then styled with CSS or manipulated with JavaScript. The `<div>` tag is easily styled in using the class or id attribute.

HTML

HTML document declaration

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Title of the document</title>  
  </head>  
  
  <body>  
    The content of the document.....  
  </body>  
  
</html>
```

CSS

CSS definition:

CSS for Cascading Style Sheets, is a computer language used on the Internet for formatting HTML files and pages. It is translated into French by cascading style sheets. Appearing in the 1990s, **CSS** is presented as an alternative to formatting via tags, especially HTML.

CSS is a rules-based language — you define style rules for particular elements or groups of elements on the page. For example, "I want the main title of my page to appear in red in large print. The rule begins with a selector, the formatted HTML element.

CSS

```
body {  
  background-color: lightblue;  
}
```

```
h1 {  
  color: white;  
  text-align: center;  
}
```

```
p {  
  font-family: verdana;  
  font-size: 20px;  
}
```

```
<html>  
<head>  
<style>  
  .myDiv {  
    border: 5px outset red;  
    background-color: lightblue;  
    text-align: center;  
  }  
</style>  
</head>  
<body>  
  
  <div class="myDiv">  
    <h2>This is a heading in a div element</h2>  
    <p>This is some text in a div element.</p>  
  </div>  
  
</body>  
</html>
```

JavaScript definition

JavaScript is a programming language primarily used on the Internet, alongside HTML and CSS. It uses scripts to create dynamic content. It thus complements the two other basic languages of the Web and can store values, perform operations or even execute code according to certain events. It is a prototype-oriented language, that is to say similar to an object-oriented language, but without class.

Although mostly found on the client side, it can also be used on the server side. When it loads on the client side, it is a particularly fast and efficient language that does not require any querying to the server. It is appreciated for its simplicity, flexibility and power. JavaScript, not to be confused with Java, was created in 1995 by the Netscape Communication Corporation. First called LiveScript, it took the name JavaScript when the company partnered with SUN, the originators of Java. It has adapted perfectly to the changes of the Web over the years. JavaScript is a popular scripting language that improves web usability, making the experience more dynamic and enjoyable. It is also the only native programming language of the Web.

JavaScript

Variables declaration:

- We use **var**, **let** and **const** to declare variables.
- Variables defined with **let** cannot be redeclared.
- Variables defined with **const** cannot be redeclared and reassigned

Data types:

```
let length = 16; // Number
let lastName = "Johnson"; // String
let x = {firstName:"John", lastName:"Doe"}; // Object
const cars = ["Saab", "Volvo", "BMW"]; // Array
let x = false, x = true // Boolean
```

Function declaration

```
function myFunction(p1, p2) {  
  return p1 * p2; // The function returns the product of p1 and p2  
}
```

myFunction(5,2); returns 10

Arrow function

```
const myFunction = (p1,p2) => p1*p2
```


Table declaration

Arrays and arrays methods

```
const cars = ["Saab", "Volvo", "BMW"];
```

Let l = cars.length; returns 3

```
Cars[1] = "Volvo"
```

React JS

React definition:

- **React** is a frontend language that works within JS to enhance usability functionality. It is an open-source library which was initially released in 2013. It has been developed by Facebook since 2013 and is now used for major web applications such as Facebook, Instagram, WhatsApp, Yahoo! Etc.

React Components:

- Components allow you to break up the UI into independent, reusable elements, allowing you to consider each element in isolation.
- Conceptually, components are like JavaScript functions. They accept arbitrary input (called "props") and return React elements describing what should appear on the screen.

React JS

```
function App() {  
  return (  
    <div className="App">  
      <div  
className='container'>  
        <Sidebar/>  
        <Home />  
      </div>  
    </div>  
  );  
}  
  
export default App;
```

```
import React from 'react'  
import './home.css'  
import Grid from  
  '../components/grid/Grid'  
import Topbar from  
  '../components/topbar/Topbar'  
  
const Home = () => {  
  return (  
    <div className="home">  
      <Topbar/><  
      <Grid />  
    </div>  
  )  
}  
  
export default Home
```

Node JS definition

NodeJS is a runtime environment for using server-side JavaScript. Thanks to its non-blocking operation, it makes it possible to design high-performance network applications, such as a web server or an API. It was created in 2009 by Ryan Dahl. It is all about running and processing JS projects/applications on the server-side and not on the client (browser) side.

Node.js and NPM:

NPM is the package manager for Node.js and any JavaScript environment and includes over a million packages available for free. It's not the only package manager out there, but it's certainly the most popular thanks to its growing community of active developers.

NPM includes a command-line tool that allows, among other things, to install and uninstall packages, and manage module versions and project dependencies. It is from NPM that you can install, for example, the ExpressJS web framework, the Axios library for HTTP requests, the MongoDB Mongoose object modelling tool and many others.

Express JS

Express Js definition:

Express.js is a framework for building web applications based on Node.js.

Express is a minimalistic and flexible Node.js web application framework that provides a robust feature set for web and mobile applications.

Express Js Routing:

Routing refers to the definition of application endpoints (URIs) and how they respond to client requests.

CORS:

- When communicating over the Internet, **CORS** is the mechanism that allows browsers to access resources that they could not originally because the resource is of a different origin.

Express JS tutorial

Step 1:

Start your terminal/cmd, create a new folder named back-end

Step 2:

Navigate to the back-end folder, in the terminal type the command `npm init -y`, this command initializes a `package.json`

Step 3:

Install Express with the command `npm install express mysql cors nodemon`

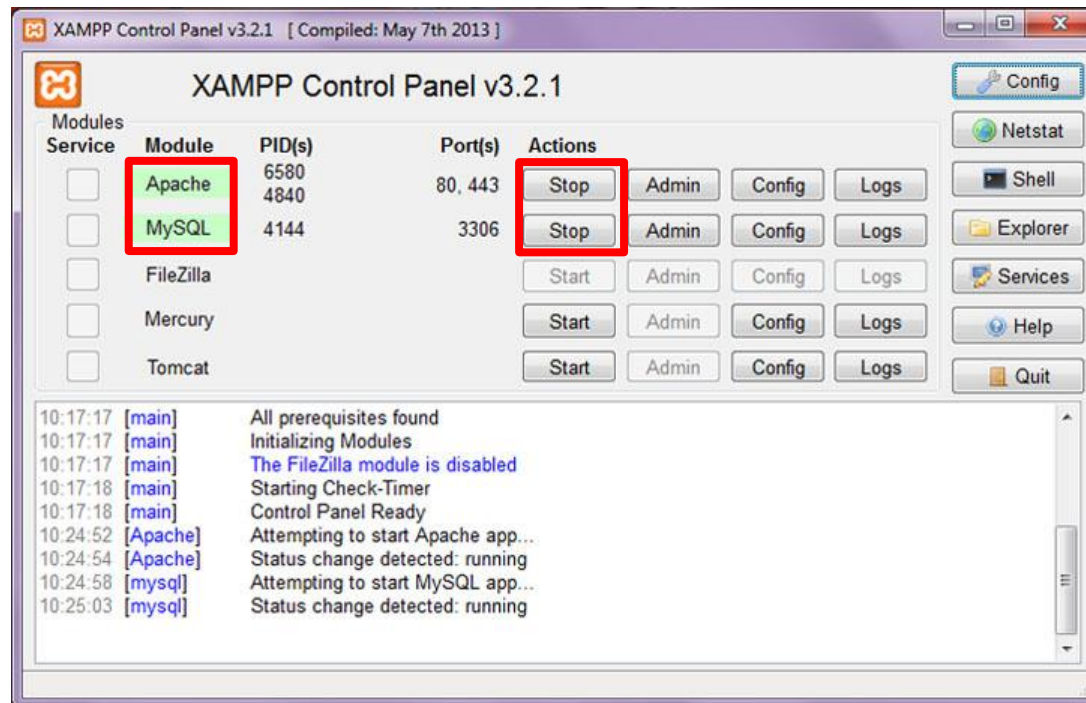
To run the `index.js` automatically in “scripts” “start”: “`nodemon index.js`”,

Step 4:

To launch `index.js` write: `npm start`

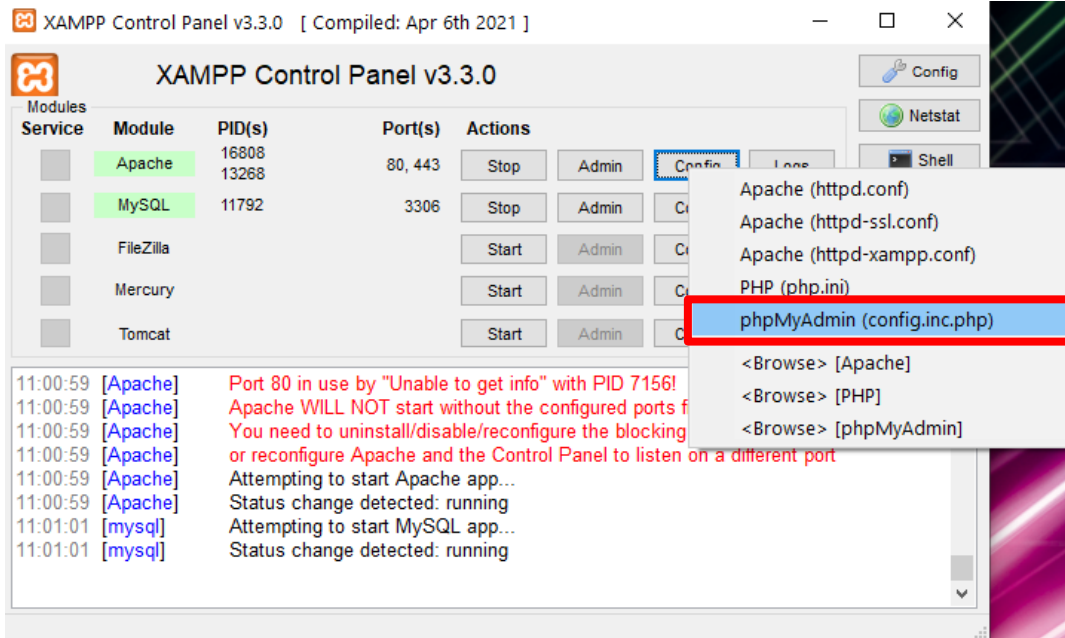
Xampp installation

Download link: <https://www.apachefriends.org/download.html>



Xampp installation

change password



```
/* Authentication type and info */  
$cfg['Servers'][$i]['auth_type'] = 'config';  
$cfg['Servers'][$i]['user'] = 'ESP32';  
$cfg['Servers'][$i]['password'] = 'esp32io.com';  
$cfg['Servers'][$i]['extension'] = 'mysqli';  
$cfg['Servers'][$i]['AllowNoPassword'] = true;  
$cfg['Lang'] = '';
```

```
/* Bind to the localhost ipv4 address and tcp */  
$cfg['Servers'][$i]['host'] = '127.0.0.1';  
$cfg['Servers'][$i]['connect_type'] = 'tcp';
```

```
/* User for advanced features */  
$cfg['Servers'][$i]['controluser'] = 'ESP32';  
$cfg['Servers'][$i]['controlpass'] = 'esp32io.com';
```

```
/* Advanced phpMyAdmin features */  
$cfg['Servers'][$i]['pmadb'] = 'phpmyadmin';  
$cfg['Servers'][$i]['bookmarktable'] = 'pma__bookmark';  
$cfg['Servers'][$i]['relation'] = 'pma__relation';
```


Database

Create a new data base

The screenshot shows the phpMyAdmin web interface in a browser. The browser's address bar displays the URL `localhost/phpmyadmin/index.php?route=/server/databases&server=1`. The phpMyAdmin logo is visible at the top left of the interface. On the left sidebar, under the 'Recent' tab, a tree view of databases is shown, with the 'New' button highlighted by a red number 1. The main content area is titled 'Databases' and contains a 'Create database' section. In this section, the text 'SensorData' is entered into the database name field, which is highlighted by a red number 2. The character set is set to 'utf8mb4_general_ci', and the 'Create' button is highlighted by a red number 4. A red number 3 is placed above the character set dropdown menu. Below the 'Create database' section, there is a 'Filters' section with a text input field labeled 'Containing the word:'. At the bottom of the interface, a table header is visible with columns 'Database', 'Collation', and 'Action'.

Table creation 1

Create Table Procedure 1 :

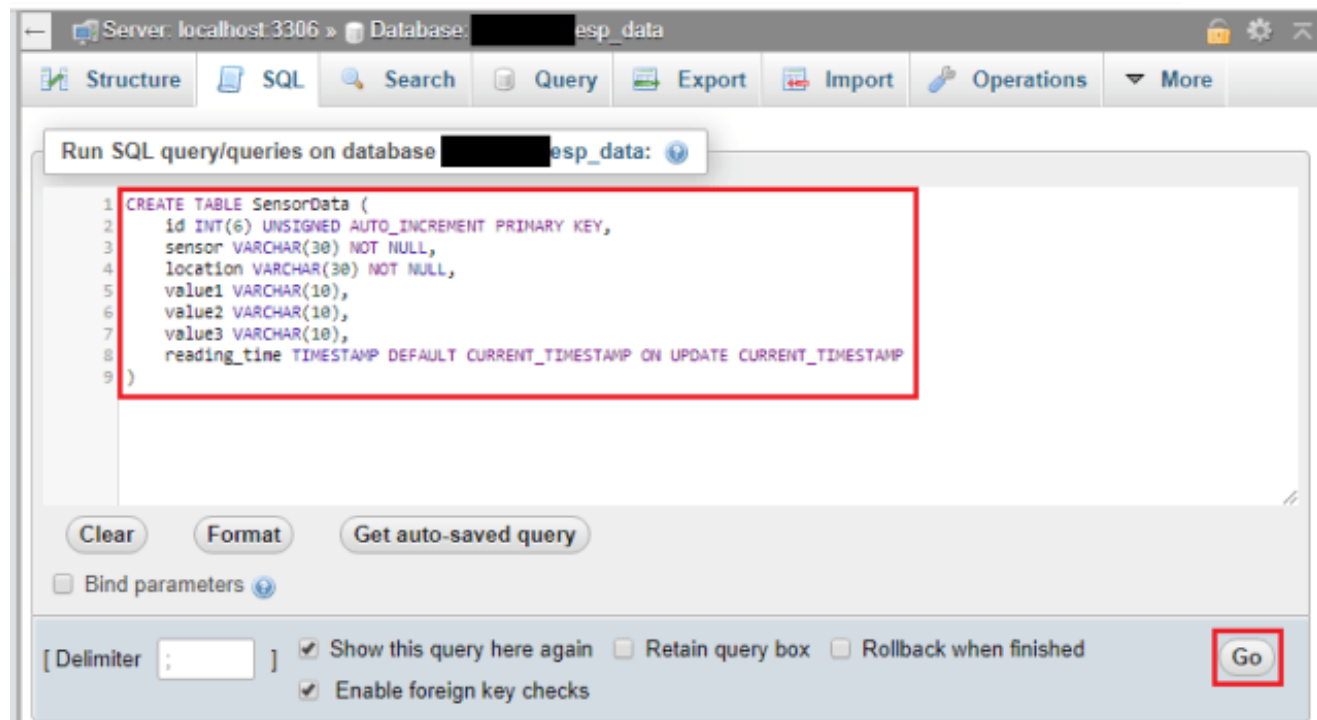
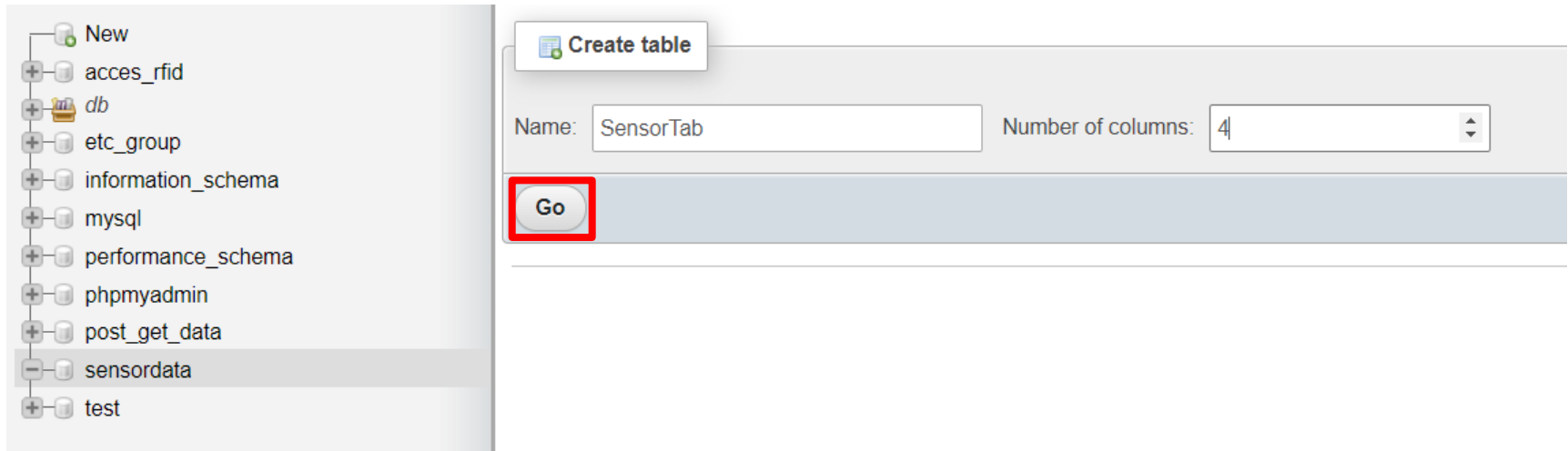


Table creation 2

Create Table Procedure 2 :



The screenshot displays a database management interface. On the left, a tree view shows a list of databases: New, acces_rfid, db, etc_group, information_schema, mysql, performance_schema, phpmyadmin, post_get_data, sensordata (highlighted), and test. On the right, a 'Create table' dialog box is open. It contains a 'Name' field with the text 'SensorTab' and a 'Number of columns' field with the value '4'. Below these fields, a 'Go' button is highlighted with a red square.

New

- + acces_rfid
- + db
- + etc_group
- + information_schema
- + mysql
- + performance_schema
- + phpmyadmin
- + post_get_data
- sensordata
- + test

Create table

Name: SensorTab Number of columns: 4

Go

Table creation 3

Server: 127.0.0.1 » Database: sensordata » Table: SensorTab

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Table name: SensorTab Add 1 column(s) Go

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index	A_	Comments
id	INT	6	None			<input type="checkbox"/>	PRIMARY	<input checked="" type="checkbox"/>	
<small>Pick from Central Columns</small>									
sensor	VARCHAR	30	None			<input type="checkbox"/>	---	<input type="checkbox"/>	
<small>Pick from Central Columns</small>									
value1	VARCHAR	30	None			<input type="checkbox"/>	---	<input type="checkbox"/>	
<small>Pick from Central Columns</small>									
access_time	TIMESTAMP	6	None			<input type="checkbox"/>	---	<input type="checkbox"/>	
<small>Pick from Central Columns</small>									

Structure

Table comments: Collation: Storage Engine: InnoDB

PARTITION definition:

Partition by: (Expression or column list)

Partitions:

Preview SQL Save

Table creation 4

Server: 127.0.0.1 » Database: sensordata » Table: sensortab

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	id	int(6)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/> 2	sensor	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/> 3	value1	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/> 4	access_time	timestamp(6)			No	current_timestamp(6)		ON UPDATE CURRENT_TIMESTAMP(6)	Change Drop More

```
SELECT * FROM `sensortab`
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

id sensor value1 access_time

Express JS tutorial

Step 4:

Create an **index.js** file and enter the following code:

```
// Installing Packages from Node Package Manager: we run the command "npm  
install express mysql cors"
```

```
// Importing these packages by writing the following code:
```

```
const express = require('express')
```

```
const mysql = require('mysql')
```

```
const cors = require('cors')
```

Express JS

`app = express()` // Invoking express function and calling it app

`app.use(cors())` // Applying use() method to set up "cors" middleware to our express function app

`port = 5000` // defining the port on which our app will listen

// Connecting to database by using specified host and name of the database and using createConnection method of mysql

```
const connection = mysql.createConnection({  
  host:"localhost",  
  database:"cps_ecc_22",  
  user:"root",  
  password:""  
})
```

Express JS

// Running our Express app

```
app.listen(port,(err)=>err?console.log(err):console.log(`Server Running on port ${port}`))
```

// Connecting to database

```
connection.connect(err=>{err?console.log(err):console.log("Connection to database OK")})
```

// Once connected to database, we create a ROUTE that specifies the REQUEST type or method

```
app.get('/data',(req,res)=>{  
  connection.query('SELECT * FROM iot',(err,rows)=>{  
    err?res.send(err):res.send(rows)  
  })  
})
```


React JS

To start your server, type the command `node index.js` in your terminal.

To test your application, open the browser and navigate to <http://localhost:5000>

Step 1: How to create a React app

In your terminal, type the command `npm create-react-app dashboard`

To create a new, React project, we can use the **npm tool**, provided we have a recent **npm** version.

Once you run this command, a folder named "dashboard" will be created where we specified it on your computer and all the packages it needs will be automatically installed.

Note: Creating a new React app usually takes 2-3 minutes, sometimes longer.

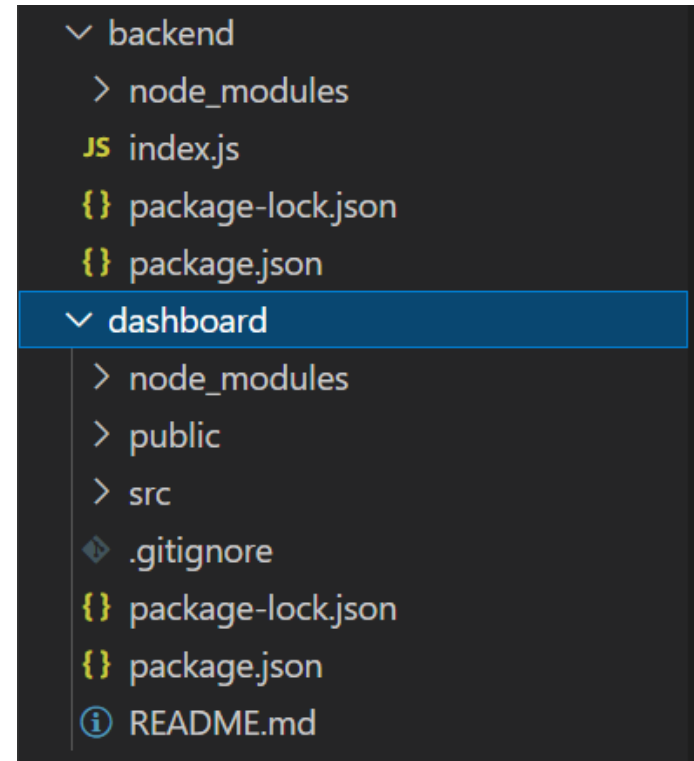
Step 2:

To start your React project, you can simply run: `npm start`

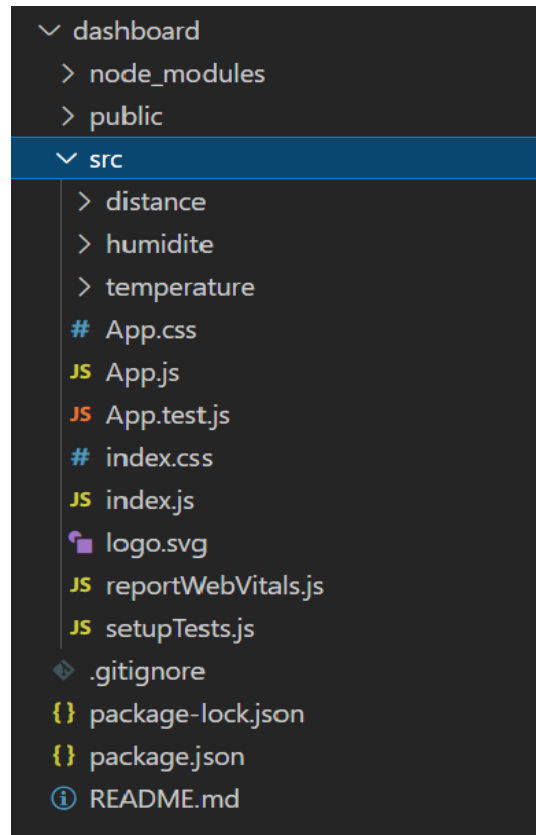
React JS

Step 3:

Once our react app is run, the structure of the back-end and front end folders is as indicated in the screenshot.



React JS



Step 4:

In the src folder, we will create a folder which called « components » . In « components » we will create three folders called « hunidity, temperature and distance ».

These three folders will present three components which will be called finally by the parent component **'App Js'**

React JS

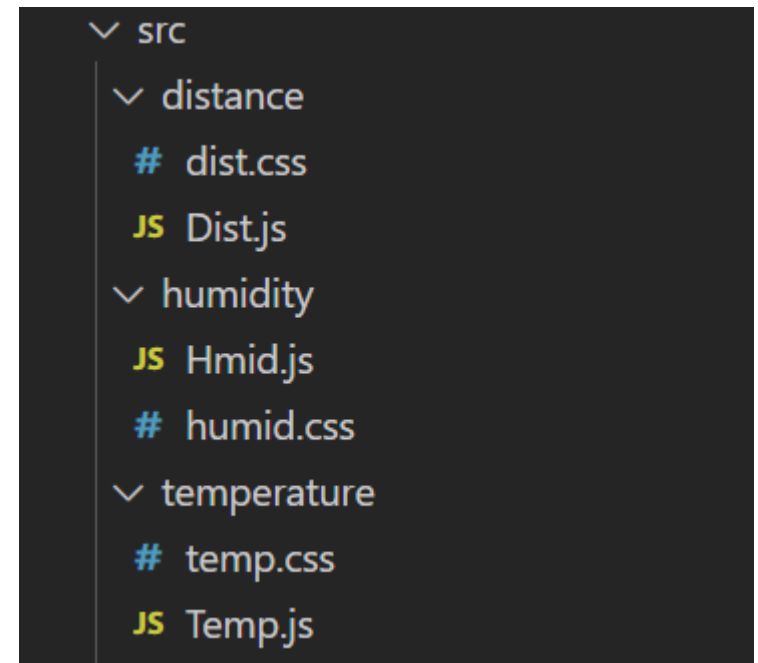
Step 5:

In distance, humidity and temperature folders, create JS and CSS files as indicated in the following screenshot.

The purpose is to develop and style every component separately then call them by App.js.


Now, let's code the distance indicator.

In the Dist.js, and dist.css write the following code indicated in the screenshot below



React JS

```
1  import React,{useState,useEffect} from 'react'
2  import axios from 'axios'
3  import './dist.css'
4
5  const Dist = () => {
6    const [info, setInfo] = useState([])
7    useEffect(() => {
8      const fetchData = async ()=>{
9        try {
10          const res = await axios.get('http://localhost:5000/data')
11          setInfo(res.data)
12          console.log(res.data)
13        } catch (error) {
14          console.log(error)
15        }
16      }
17      fetchData()
18    }, [info])
19
20    return (
21      <div className='dist'><h3>Distance</h3>{info[0]?.DISTANCE}</div>
22    )
23  }
24  export default Dist
```

```
1  .dist{
2    background-color:  bisque;
3    width: 250px;
4    height: 100px;
5    position: absolute;
6    top: 10%;
7    left: 38%;
8  }
```

React JS

```
1 import React,{useState,useEffect} from 'react'
2 import './temp.css'
3 import axios from 'axios'
4
5 const Temp = () => {
6   const [info, setInfo] = useState([])
7   useEffect(() => {
8     const fetchData = async ()=>{
9       try {
10         const res = await axios.get('http://localhost:5000/data')
11         setInfo(res.data)
12         console.log(res.data)
13       } catch (error) {
14         console.log(error)
15       }
16     }
17     fetchData()
18   }, [info])
19
20   return (
21     <div className='temp'><h3>Température</h3> {info[0]?.TEMPERATURE}</div>
22   )
23 }
24 export default Temp
```

```
1 .temp{
2   background-color: aquamarine;
3   width: 250px;
4   height: 100px;
5   position: absolute;
6   top: 10%;
7   left: 5%;
8 }
```

The same thing for Temp.js and Humid.js

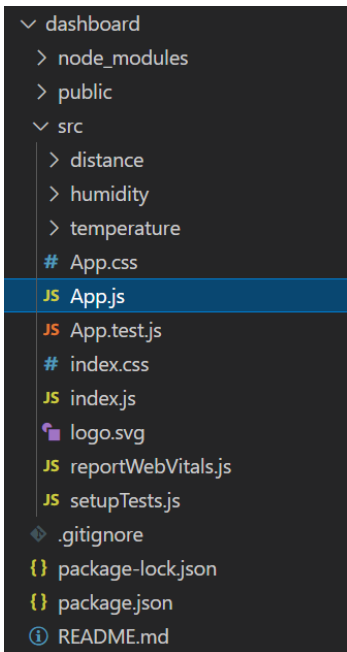
React JS

```
1 import React,{useState,useEffect} from 'react'
2 import axios from 'axios'
3 import './humid.css'
4
5 const Hmid = () => {
6   const [info, setInfo] = useState([])
7   useEffect(() => {
8     const fetchData = async ()=>{
9       try {
10         const res = await axios.get('http://localhost:5000/data')
11         setInfo(res.data)
12         console.log(res.data)
13       } catch (error) {
14         console.log(error)
15       }
16     }
17     fetchData()
18   }, [info])
19   return (
20     <div className='humid'><h3>Hmidité</h3>{info[0]?.HUMIDITE}</div>
21   )
22 }
23 export default Hmid
```

```
1 .humid{
2   background-color: aqua;
3   width: 250px;
4   height: 100px;
5   position: absolute;
6   top: 10%;
7   right: 5%;
8 }
```

The same thing for Temp.Js and Humid.Js

React JS



```
1  import './App.css';
2  import Temp from './temperature/Temp';
3  import Dist from './distance/Dist';
4  import Humid from './humidity/Hmid';
5
6  function App() {
7    return (
8      <div className="App">
9        <Temp/>
10       <Dist/>
11       <Humid/>
12     </div>
13   );
14 }
15
16 export default App;
17
```

Finally, we call all the components in the App.Js