

**Universidad de las Fuerzas Armadas ESPE**

**Laboratorio 4 WEB APP**

1. **Tema:**

Implementación de una interfaz web (Front End) responsiva, en el consumo de un servicio web RestFull (GET, PUT, POST, DELETE) con acceso a la base de datos en la nube usando React y otras librerías que permitan optimizar y mejorar la calidad del WebApp.

1. **Objetivo:**

Desarrollar una aplicación web front end que permita el consumo de un servicio RestFull con acceso a la base de datos en la nube computacional aplicando principios de usabilidad en el manejo de las interfaces para varios dispositivos.

* Usar librerías para mejorar la implementación de la interfaz de usuario.
* Conocer como crear un cliente RestFull con acceso a una Api previamente creada.

1. **Descripción:**

La aplicación web a desarrollar consumirá un servicio previamente desarrollado en temas anteriores (*Recurso 1*) con el fin de implementar un cliente RestFull que permita realizar las operaciones Get, Post, Put y Delete usando el Framework React, así también los datos obtenidos y cambios sobre ellos se encontraran reflejados y almacenados en una base de datos no sql en la nube en este caso MongoDB Atlas, dichos datos serán presentados en pantalla en una tabla con sus respectivos botones de acciones (editar, eliminar, crear). Se priorizara la creación de una interfaz con colores representativos a la acción a realizar en dicho campo asi también el uso de un diseño responsivo que se adapte a dispositivos con tamaños distintos de pantallas.

La aplicación web usara las siguientes tecnologías y conceptos:

* React: es una biblioteca Javascript diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo.
* Node.js: es un entorno en tiempo de ejecución multiplataforma para la capa de servidor orientado a eventos.
* Axios:es una biblioteca basada en promesas que se usan en navegadores y node.js
* URI: es una cadena de caracteres que identifica los recursos de una red de forma unívoca.
* Material UI:es una biblioteca que provee de material desing enfocado al diseño y visualización de páginas.
* Fortawesome: biblioteca que provee de iconos y fuentes basadas en CSS y LESS.

1. **Análisis**

Se desarrollará una aplicación web que permita obtener y modificar los datos de clientes de una determinada empresa.

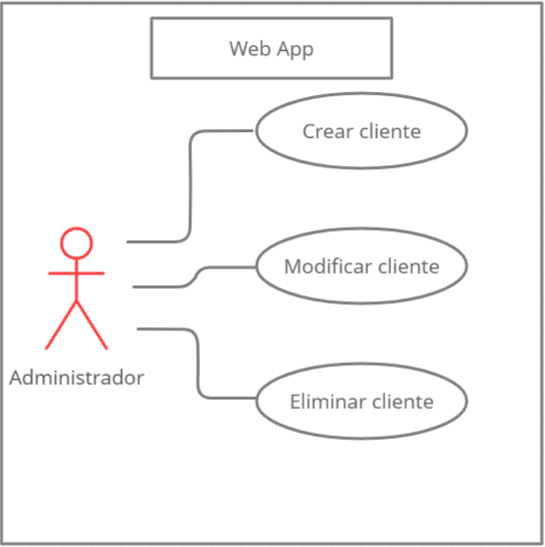
* 1. ***Requisitos Funcionales.***

1. La aplicación web deberá consumir un servicio rest que contenta todos los datos necesarios de los clientes.
2. La aplicación web mostrara todos los datos pertenecientes a los clientes en una tabla.
3. La aplicación web permitirá realizar las operaciones CRUD sobre los datos obtenidos y guardarlos en la base de datos después de ser usados.
   1. ***Requisitos No Funcionales.***
4. La aplicación web podrá adaptarse a la pantalla de cualquier dispositivo en un navegador de internet.
5. **Diseño**
   1. ***Diagrama de caso de uso***

En la *Figura 1* se muestra el diagrama de casos de uso que representa de la aplicación web a desarrollar donde tenemos:

* Un usuario, el cual lo nombraremos “Administrador” va a realizar las siguientes acciones:
* Crear un nuevo cliente
* Eliminar un cliente existente
* Modificar un cliente existente

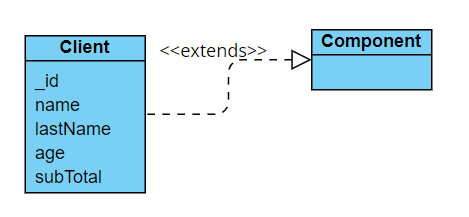
En este caso tenemos solo un tipo usuario por el tamaño la aplicación, pero se pueden agregar más con especificas acciones a realizar.



***Figura 1.*** *Diagrama casos de uso*

* 1. ***Diagrama de clases***

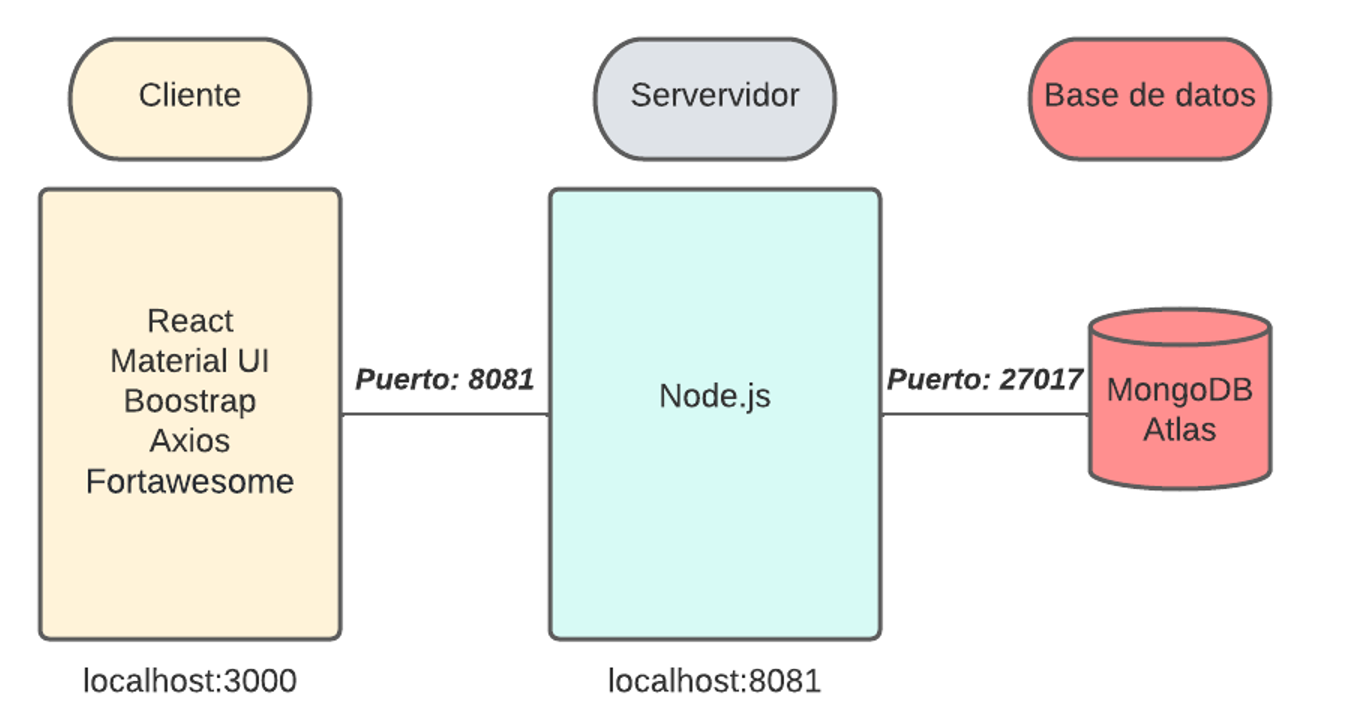
En la *Figura 2* se muestra el diagrama de clase Client que contara con los métodos get,put,post, delete, select y modalInsert que permitirán manejar los atributos \_id, name, lastName, age y subTotal consumidos del servicio rest.



***Figura 2.*** *Diagrama de clases*

* 1. ***Arquitectura web***

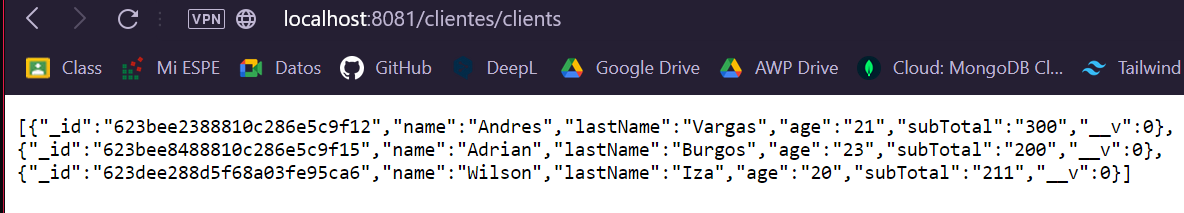
En la *Figura 3* se muestra la arquitectura que representa la aplicación web, contaremos con 3 capas; cliente, Servidor y BD *con* sus respectivos puertos y tecnologías usadas.



***Figura 3.*** *Arquitectura*

* 1. ***Diseño de la aplicación***

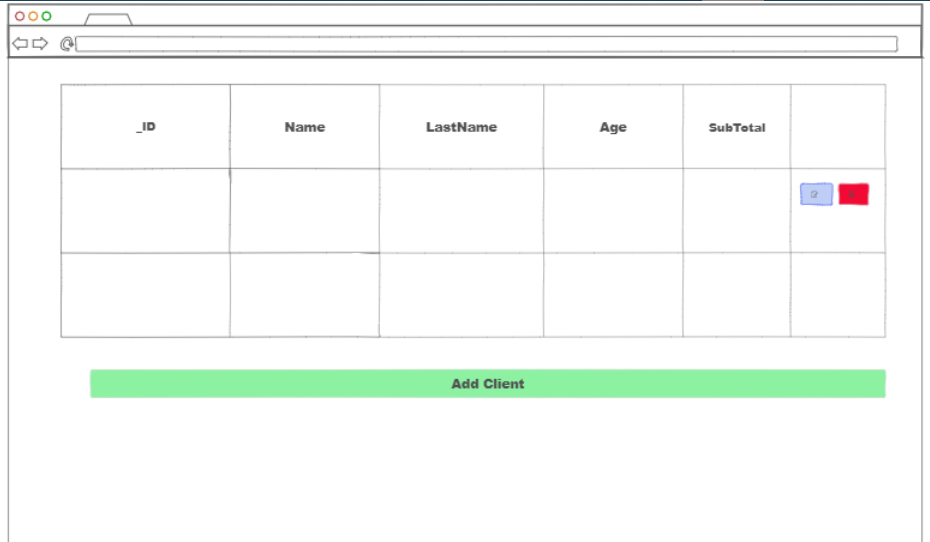
Como se menciono con anterioridad se usará el back end creado (*Recurso 1*) el mismo que con su URI (<http://localhost:8081/clientes/clients>) obtenemos lo siguiente:



***Figura 4.*** *Listado de clientes en formato JSON*

La información obtenida está en formato JSON el mismo que facilitara el manejo de dichos datos en el front end.

Al tener la información en un formato especifico y de que se compone realizamos un maquetado ubicando las claves json que son: \_id, name, lastName, age y subtotal en la cabecera de columna que le corresponde, y sus valores en los espacios designados, además debemos agregar una columna extra para los iconos de las acciones a realizar (modificar y eliminar) junto con un botón para crear un cliente en la parte inferior, el resultado se muestra se muestra en la *Figura 5*.



***Figura 5.*** *Maquetado tabla clientes*

1. ***Herramientas y entorno de desarrollo***

*NOTA*: Es posible que cuando usted realice este ejercicio, las versiones de las aplicaciones de software que instalen se hayan actualizado, sin embargo este hands-on lab considera que el software y sus versiones son compatibles y estarán estables por un período de tiempo razonable. Se le aconseja al lector, analizar las versiones más actuales. Además, el estudiante debe estar consciente que la instalación dependerá del Sistema Operativo de su elección, Windows, MacOS, Linux, etc.

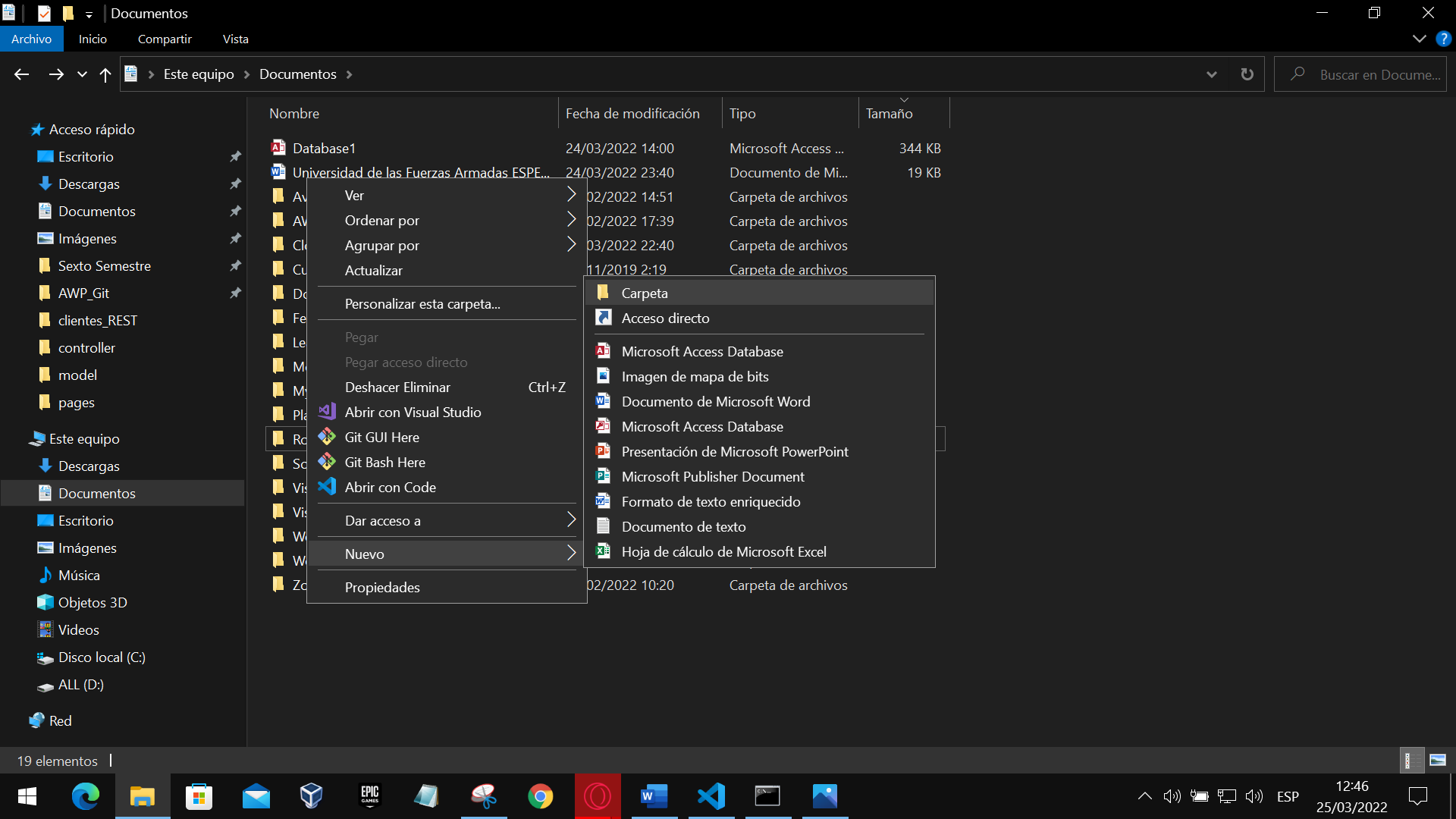
Recomendamos usar Visual Studio Code para el desarrollo de esta y demás practicas ya que este editor ofrece características necesarios y facilidades de uso.

Además de la previa instalación de node.js y react en nuestro equipo los cuales se repasaron en laboratorios anteriores.

* 1. *Visual Studio Code*

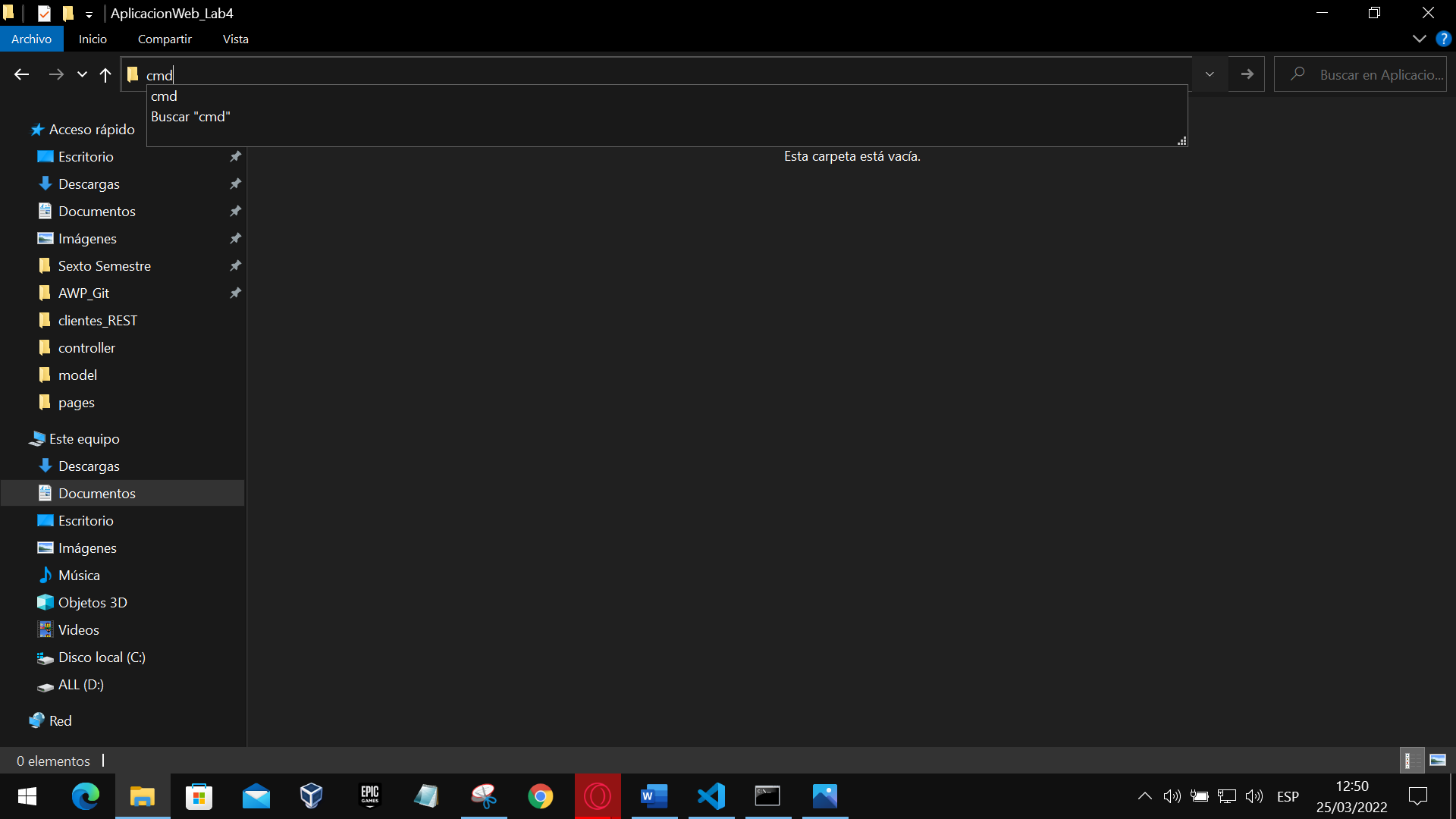
Link de descarga: <https://code.visualstudio.com/download>

* 1. *Creación del proyecto React* 
     1. Seleccionar el destino del proyecto o crearlo en una carpeta en el destino de preferencia personal.



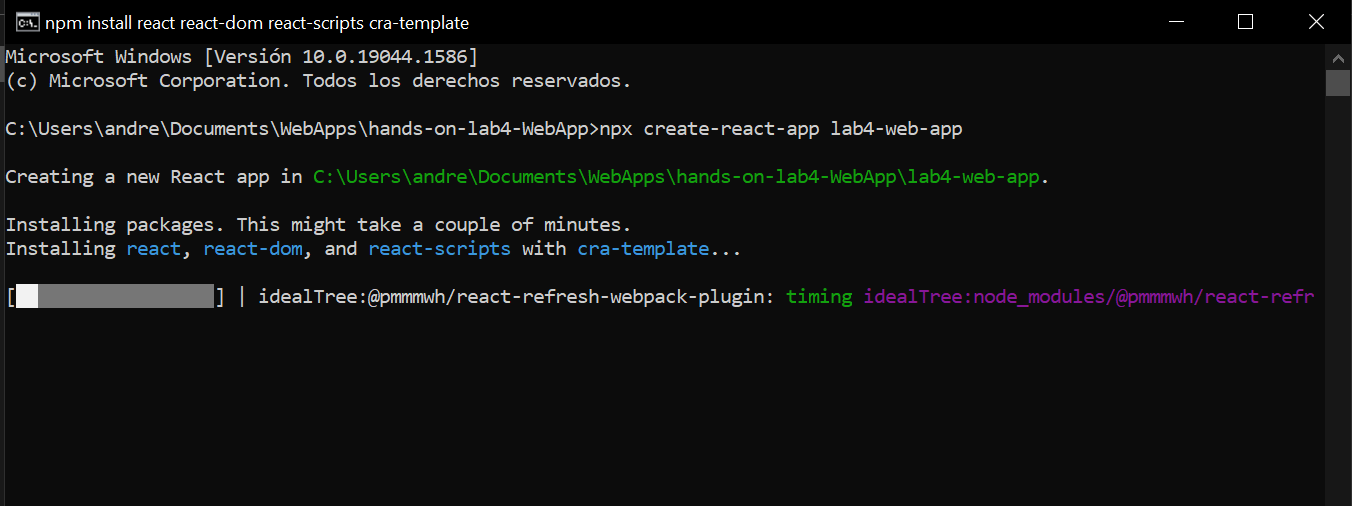
**Figura 6.** *Creación de la carpeta contenedora del proyecto React*

* + 1. Una ves dentro de la carpeta seleccionada o creada para el proyecto en la barra de búsqueda escribimos *cmd* y presionamosla tecla *intro.*

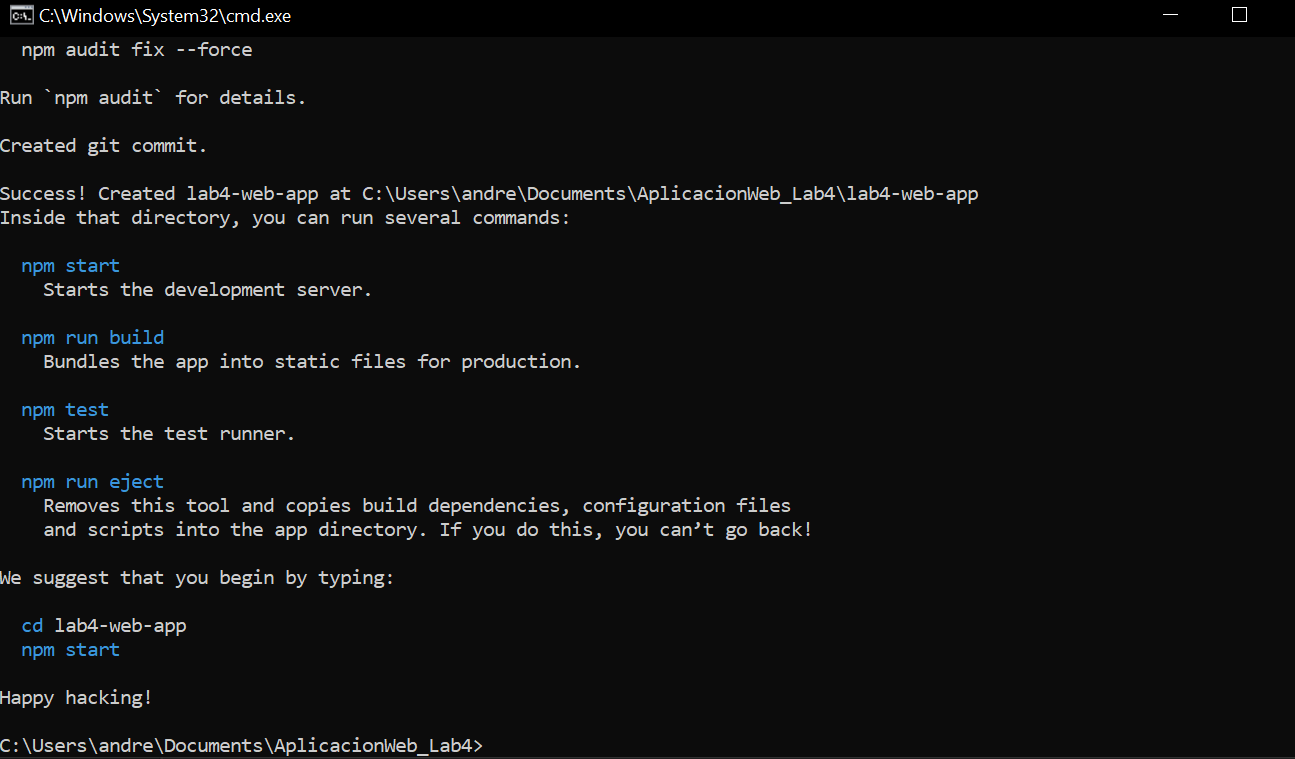


***Figura 7****. Abrir cmd en la ruta destino del proyecto*

* + 1. Automáticamente se abrirá una ventada cmd con la ruta destino del proyecto, aquí escribimos: **npx create-react-app lab4-web-app** iniciando así el proceso de creación del proyecto con el nombre *lab4-web-app*, (*puede ser nombrado a gusto personal*).

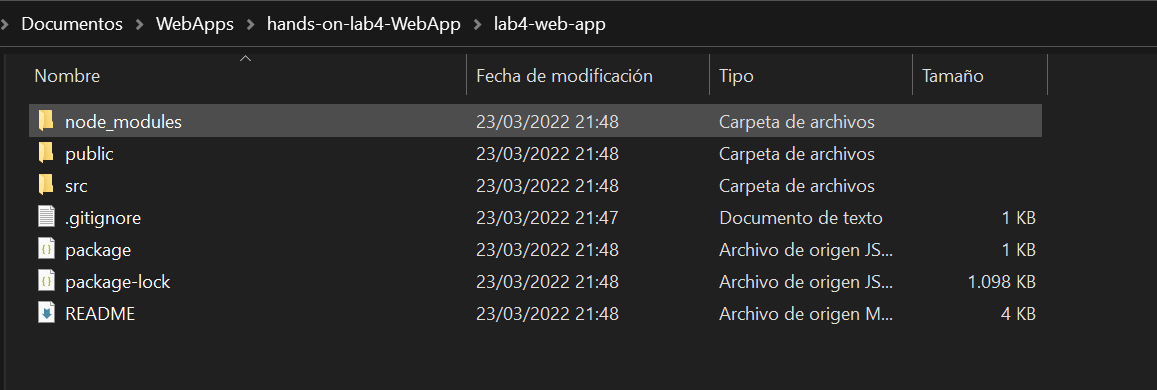


***Figura 8.*** *Creación del proyecta react****.***



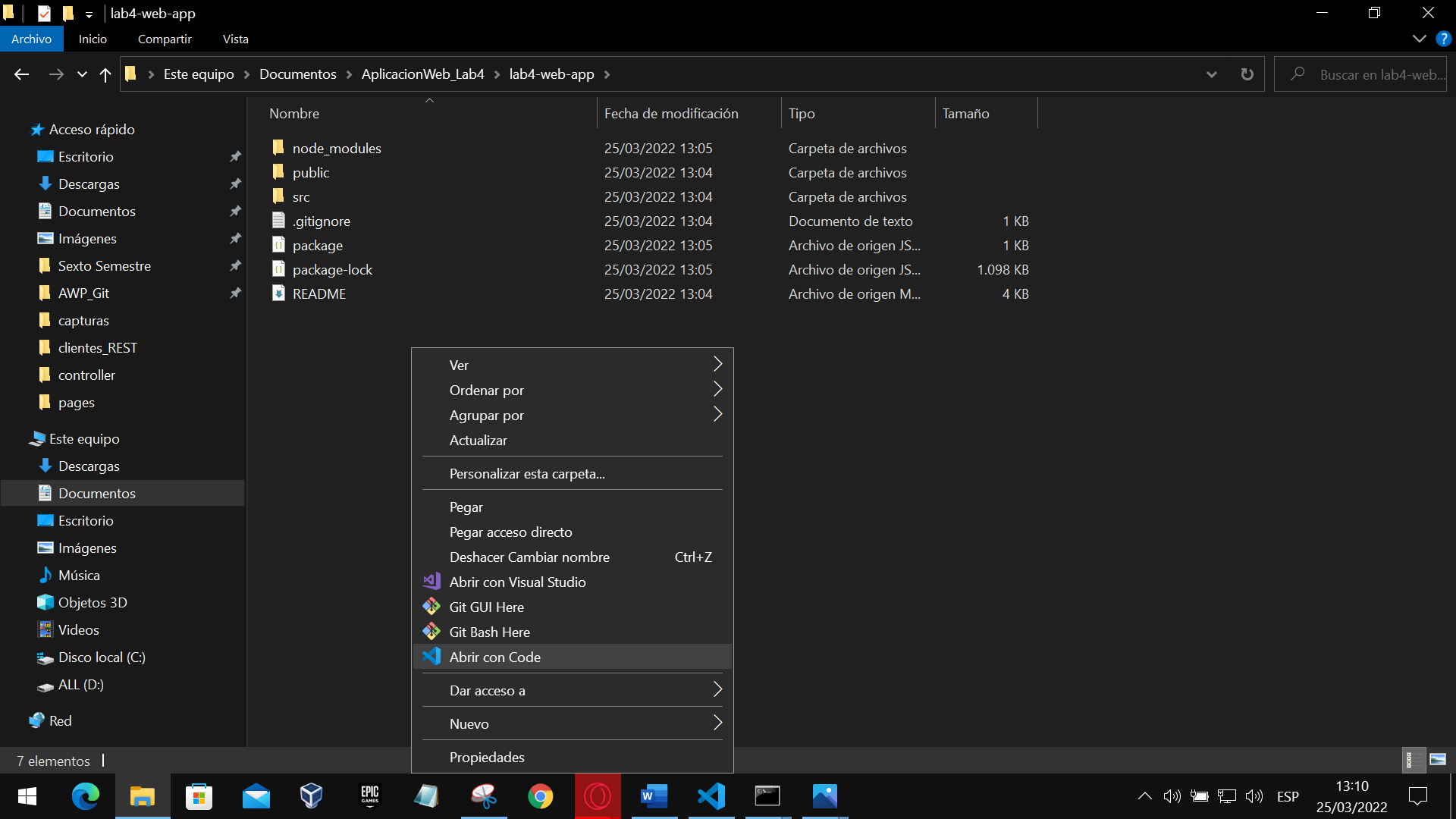
***Figura 9.*** *Finalización de la creación del proyecto de manera satisfactoria.*

* + 1. Finalizado el proceso anterior se obtendra la siguiente estructura de carpetas necesarias para el desarrollo del proyecto.



***Figura 10.*** *Estructura de carpeta creadas por cmd.*

* + 1. Abrir Visual Studio Code en la ruta del proyecto creado.

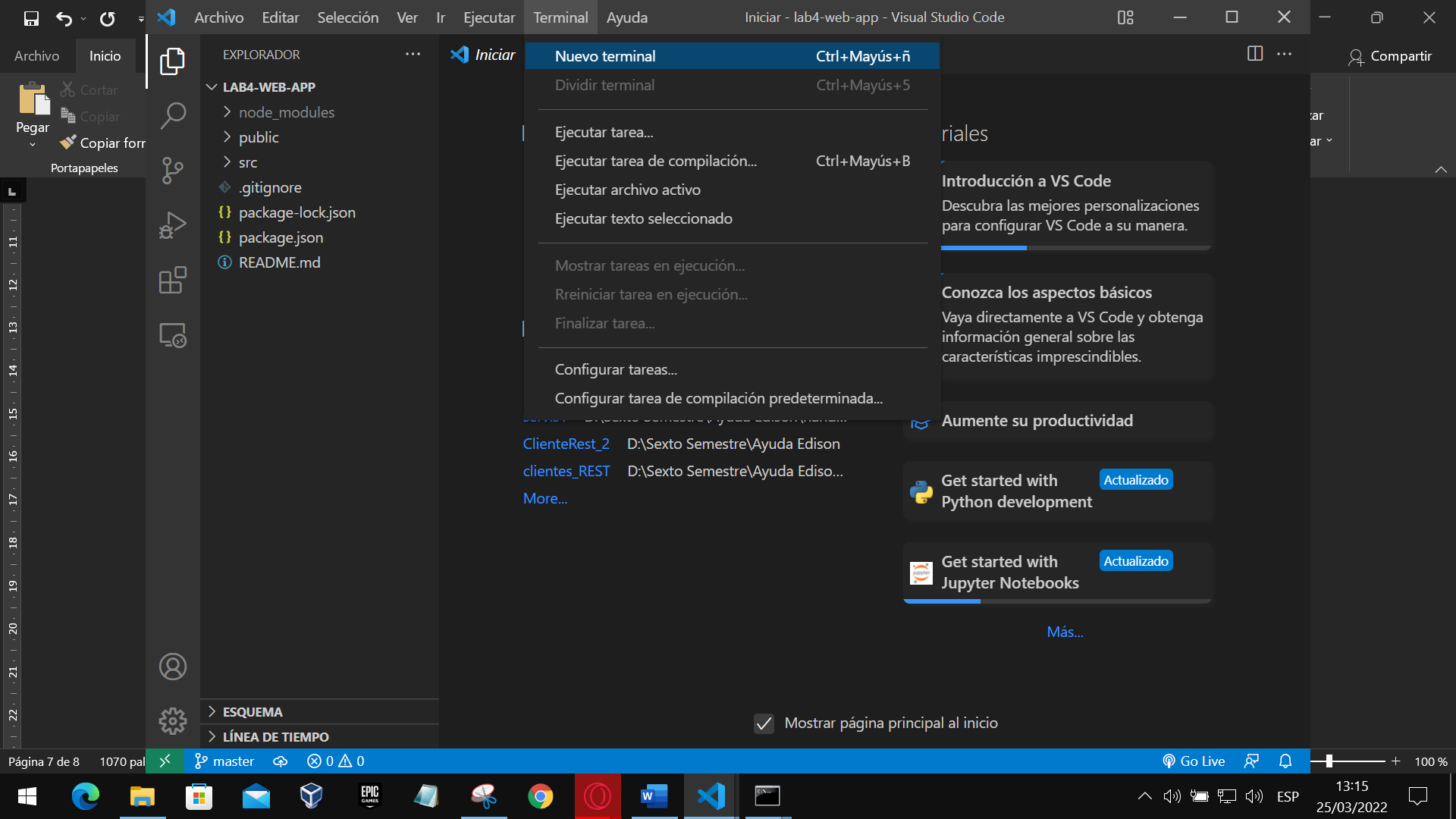




***Figura 11****. Abrir el proyecto con Visual Studio Code.*

Instalación de bibliotecas necesarias para el proyecto:

* + 1. Una vez dentro de Visual S. Code abrir una nueva terminal para la instalación de las bibliotecas necesarias para el desarrollo del proyecto.

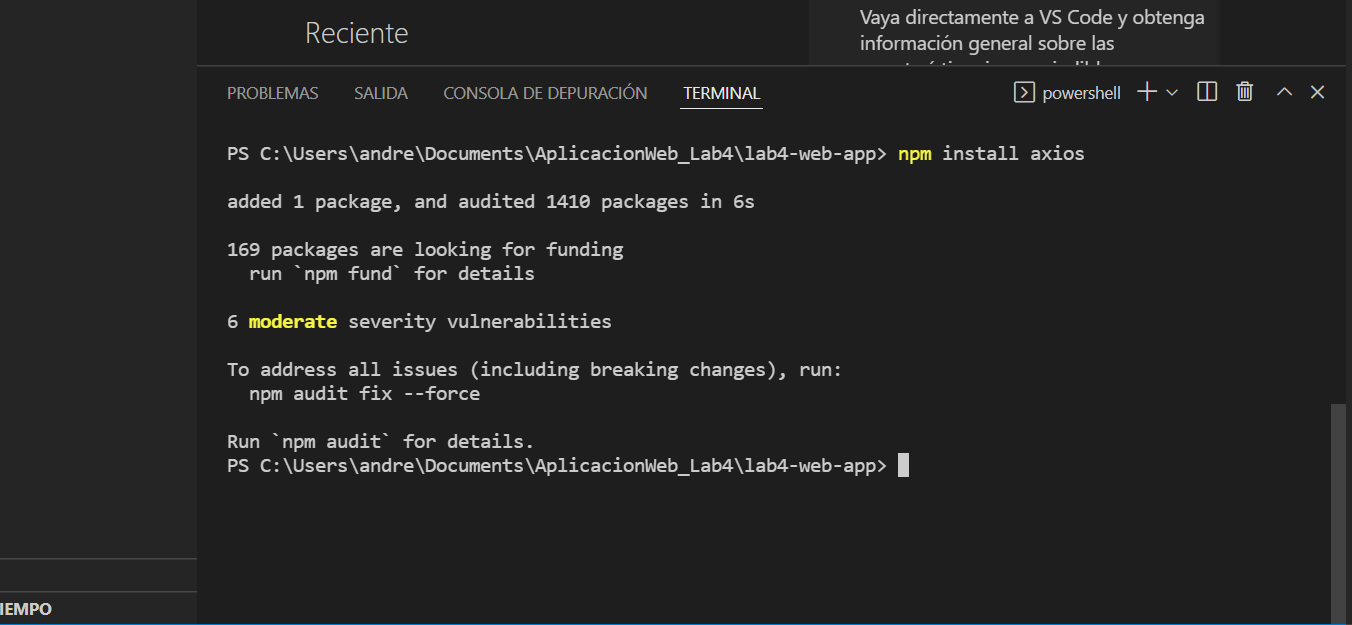




***Figura 12****. Abrir una nueva terminal en Visual S. Code*

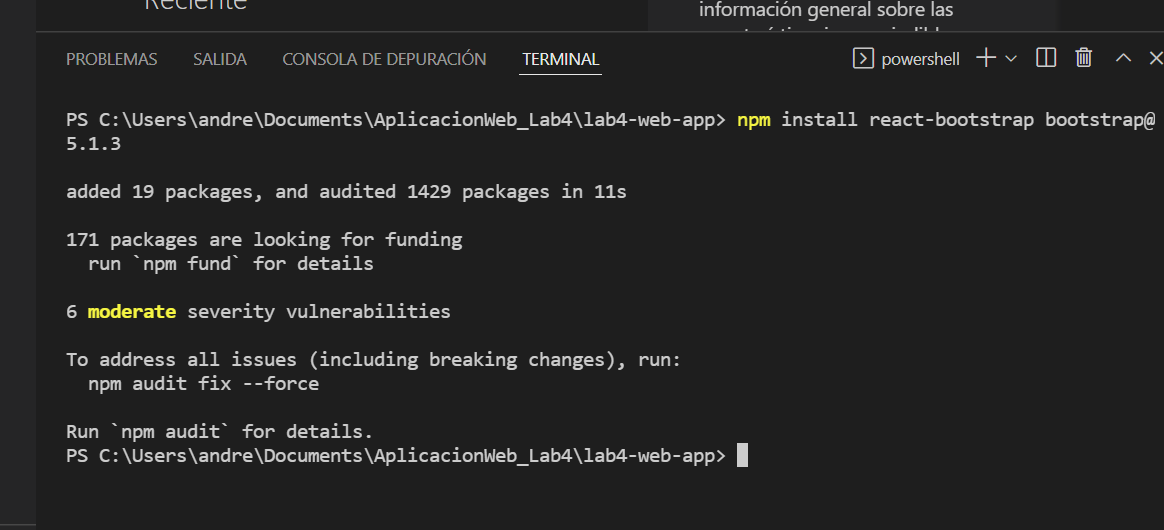
* + 1. Estando en la terminal escribe:

npm install axios



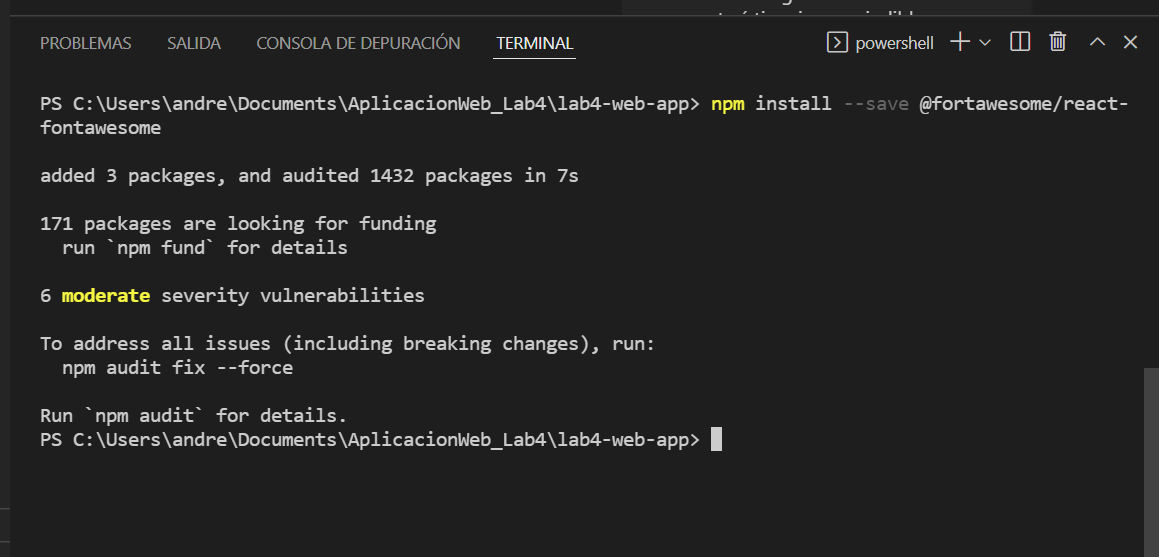
***Figura 13.*** *Instalación de axios.*

npm install react-bootstrap [bootstrap@5.1.3](mailto:bootstrap@5.1.3)



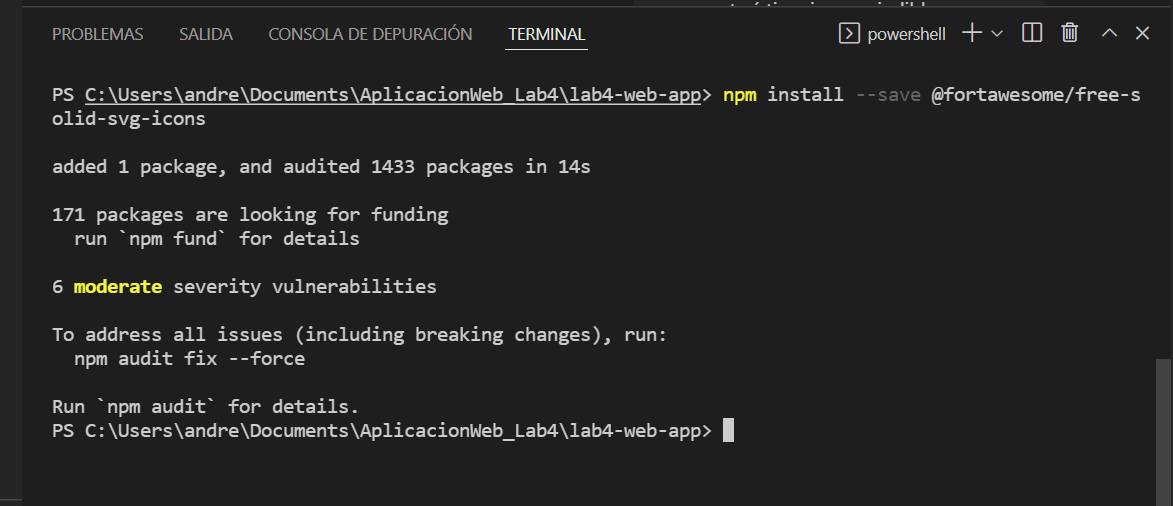
***Figura 14****. Instalación de Bootstrap*

npm install --save @fortawesome/react-fontawesome



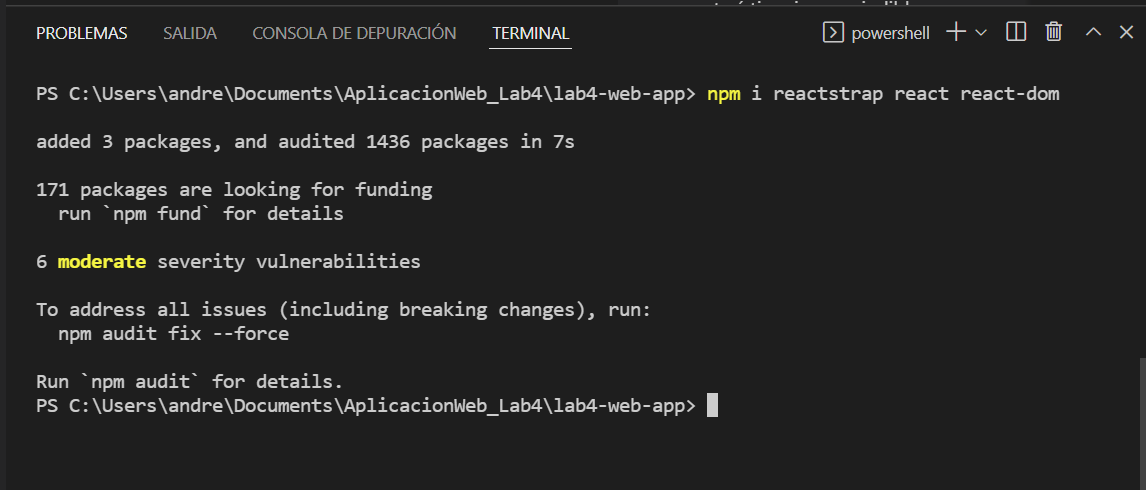
***Figura 15****. Instalación de @fortawesome/react-fontawesome*

npm install --save @fortawesome/free-solid-svg-icons



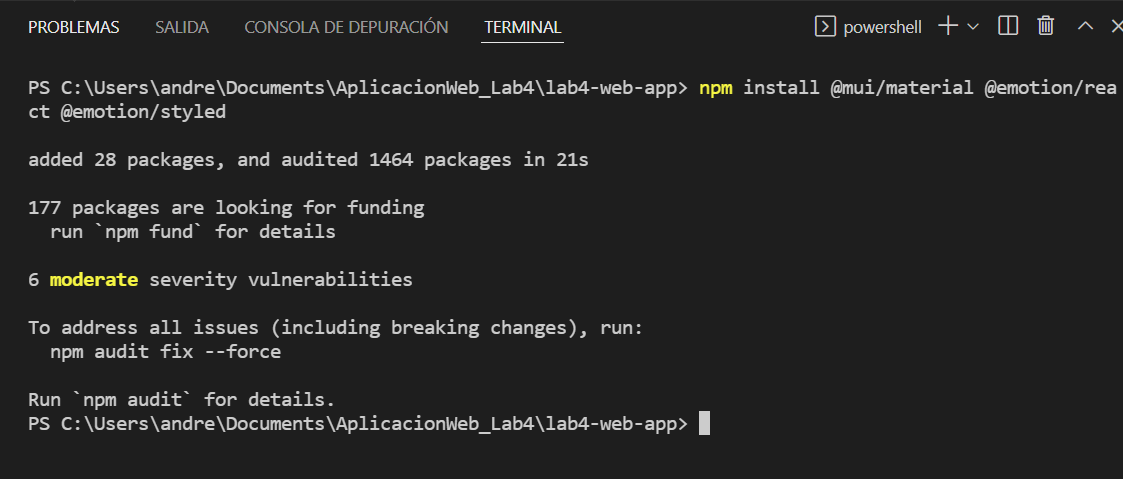
***Figura 16****. Instalación de* @fortawesome/free-solid-svg-icons

npm i reactstrap react react-dom



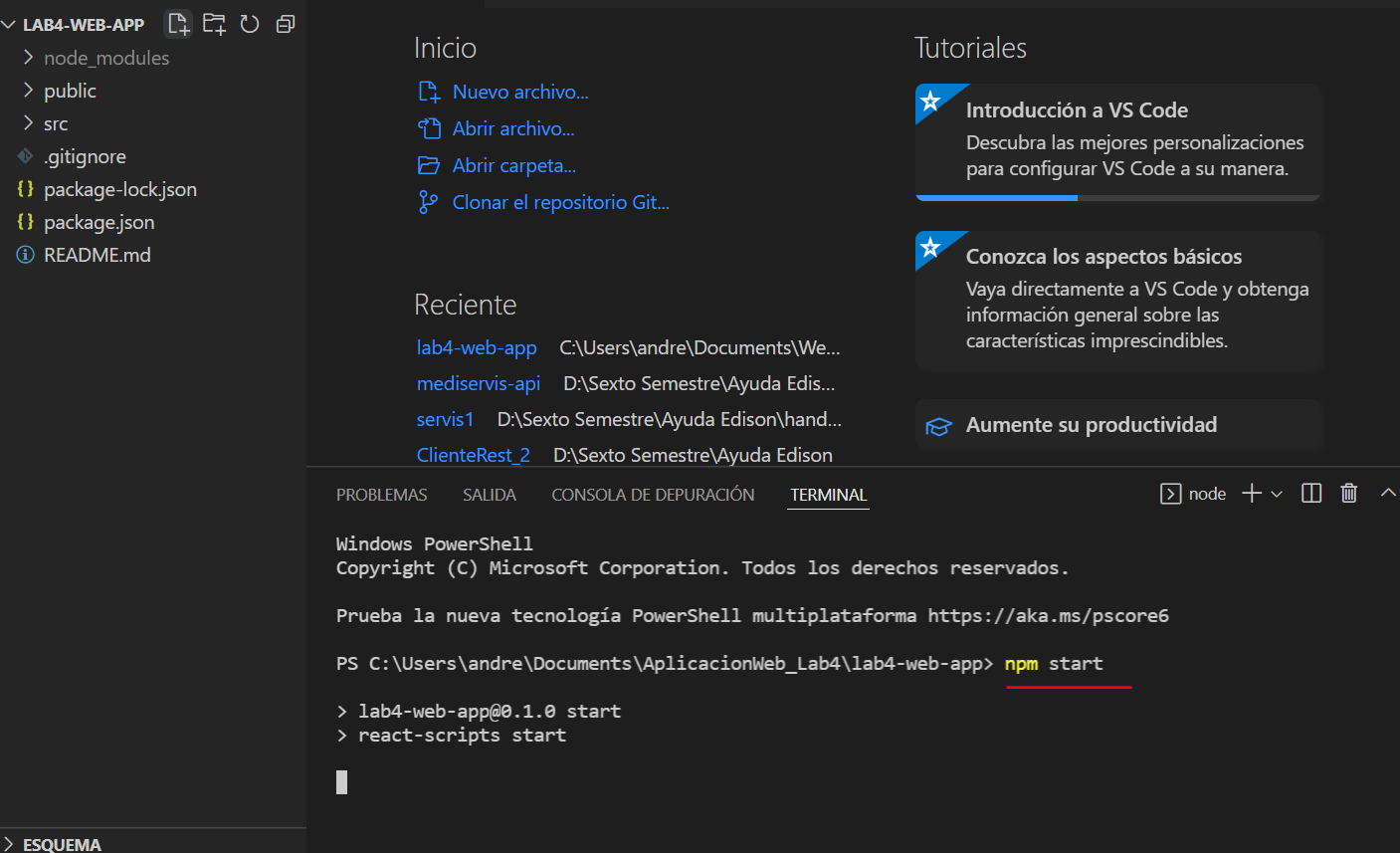
***Figura 17****. Instalación de reactstrap.*

npm install @mui/material @emotion/react @emotion/styled

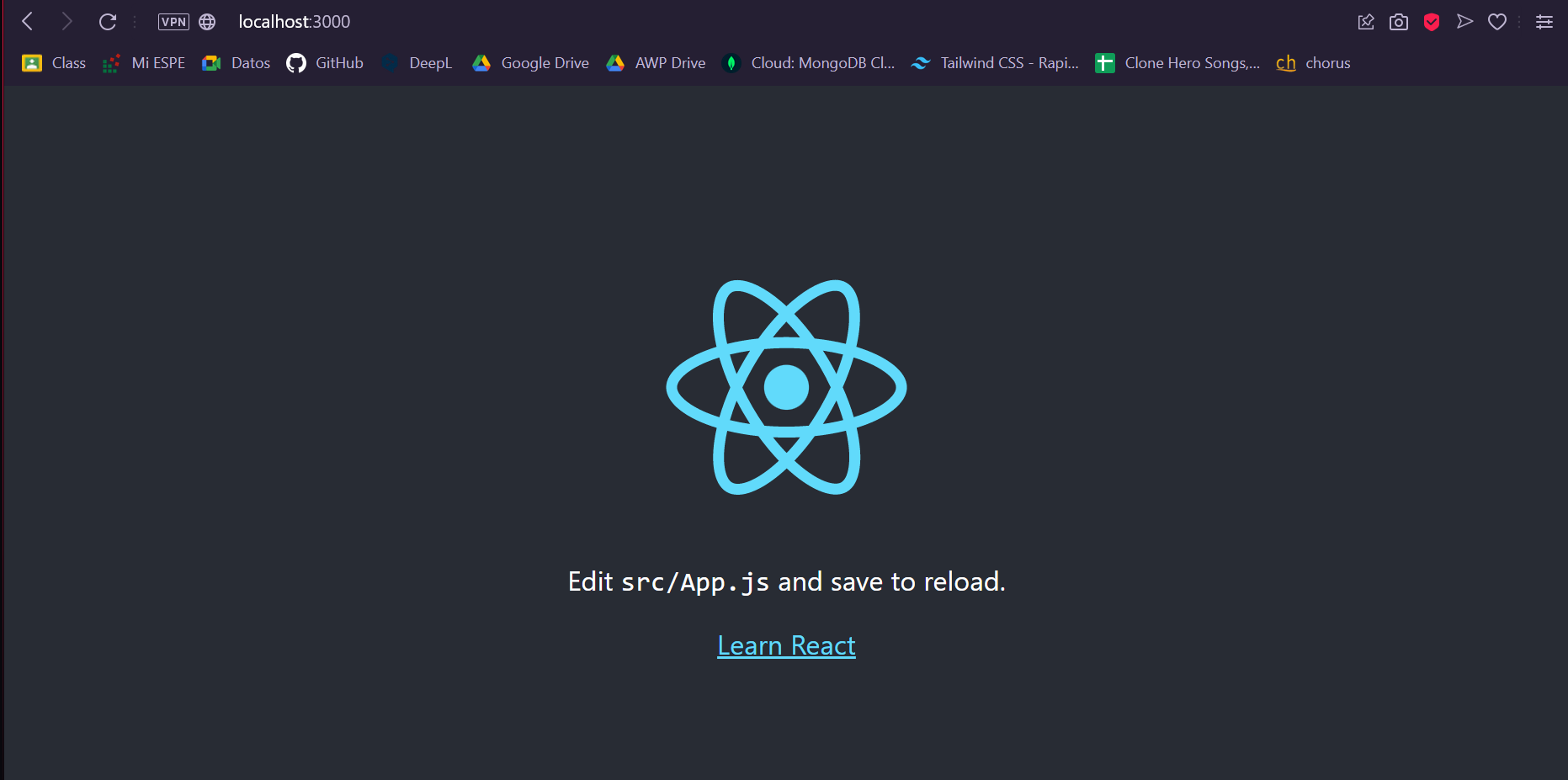


***Figura 18****. Instalación de mui/material.*

* + 1. Ahora escribe: npm start, para iniciar la ejecución del proyecto.



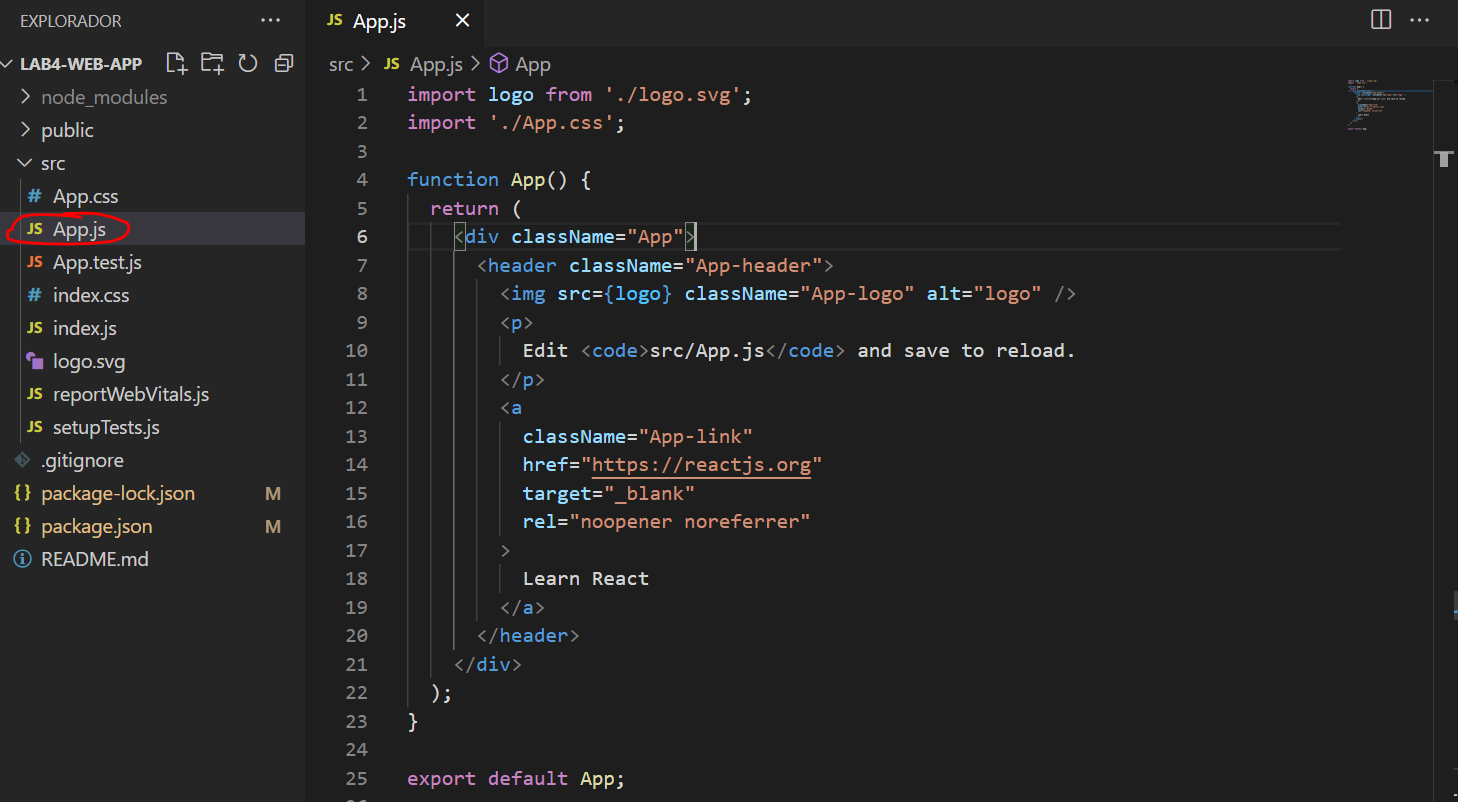
***Figura 19****. Comando npm start para ejecutar el proyecto.*



***Figura 20.*** *Proyecto react corriendo en el navegador en localhost:3000*

1. **Desarrollo de la aplicación**

En Visual S. Code abrimos el archivo App.js que se encuentra dentro de la carpeta /src de nuestra estructura de carpetas del proyecto.



***Figura 21.*** *Archivo App.js*

Borrar todas las líneas de código y seguido importar las bibliotecas previamente instaladas.

import React, { Component } from 'react';

import axios from "axios";

import "bootstrap/dist/css/bootstrap.min.css";

import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';

import { faEdit, faTrashAlt } from '@fortawesome/free-solid-svg-icons';

import { Modal, ModalBody, ModalFooter, ModalHeader } from 'reactstrap';

import Button from '@mui/material/Button';

Seguido crear un constante la cual tendrá la URI del servicio restFull al cual nos conectaremos, junto a esto crear una clase Client para permitir la creación de un estado donde se encontrara los datos y estados de ventanas flotantes a usar.

const URI="http://localhost:8081/clientes/clients/";

class Client extends Component {

  state={

    data:[],

    modalInsert: false,

    modalDelete: false,

    form:{

      \_id: '',

      name: '',

      lastName: '',

      age: '',

      subTotal: ''

    }

  }

}

export default Client;

Después crear los métodos GET, PUT, POST y DELETE necesarios para la manipulación de los datos, se nota que el método post es asincrónico (*async*) ya que se ejecutara en segundo plano cuando se termine de llenar los datos y cierre la ventana flotante.

callGet=()=>{

axios.get(URI).then(response=>{

  this.setState({data: response.data});

}).catch(error=>{

  console.log(error.message);

})

}

callPost=async()=>{

  delete this.state.form.\_id;

 await axios.post(URI,this.state.form).then(response=>{

    this.modalInsert();

    this.callGet();

  }).catch(error=>{

    console.log(error.message);

  })

}

callPut=()=>{

  axios.put(URI+this.state.form.\_id, this.state.form).then(response=>{

    this.modalInsert();

    this.callGet();

  })

}

callDelete=()=>{

  axios.delete(URI+this.state.form.\_id).then(response=>{

    this.setState({modalDelete: false});

    this.callGet();

  })

}

A continuación crear la ventana flotante de insertar (modalInsert) la misma que cambiara su estado *true* cuando se la invoque caso contrario tendrá un estado *false,* la función captureData ayudara a obtener los datos y mostrarlos al usuario en los campos input ejecutándose en segundo plano cuando este dese editarlos, esto mejora la usabilidad evitando problemas al momento que el usuario edite información.

modalInsert=()=>{

  this.setState({modalInsert: !this.state.modalInsert});

}

captureData=async data=>{

data.persist();

await this.setState({

  form:{

    ...this.state.form,

    [data.target.name]: data.target.value

  }

});

console.log(this.state.form);

}

Seguido crear una tabla con los datos correspondiente al proyecto, destacando la clase “table-responsive-md” la misma que permite hacer responsive una tabla de forma rápida

Nota: Es posible modificar la tabla para un tamaño particular “table-responsive{-sm|-md|-lg|-xl}”

Los botones son creados y asignados un evento en específico editar y eliminar representamos por los iconos faEdit y faTrashAlt respectivamente de la biblioteca FontAwesomeIcon.

render(){

    const {form}=this.state;

  return (

    <div className="App">

    <div class="table-responsive-md">

    <table className="table ">

      <thead>

        <tr>

          <th>ID</th>

          <th>Name</th>

          <th>LastName</th>

          <th>Age</th>

          <th>Sub Total</th>

        </tr>

      </thead>

      <tbody>

        {this.state.data.map(client=>{

          return(

            <tr>

          <td>{client.\_id}</td>

          <td>{client.name}</td>

          <td>{client.lastName}</td>

          <td>{client.age}</td>

          <td>{client.subTotal}</td>

          <td>

                <button className="btn btn-primary" onClick={()=>{this.selectClient(client); this.modalInsert()}}><FontAwesomeIcon icon={faEdit}/></button>

                {"   "}

                <button className="btn btn-danger" onClick={()=>{this.selectClient(client); this.setState({modalDelete: true})}}><FontAwesomeIcon icon={faTrashAlt}/></button>

                </td>

          </tr>

          )

        })}

      </tbody>

    </table>

    </div>

Consiguiente se crea las ventanas flotantes que permitirán el ingreso de datos ya sea cuando el usuario dese crea un nuevo cliente o cuando desee editar uno ya existente llamar la método captureData en cada input porque como se mencionó este permitirá la vista de los datos extraídos por \_id cuando se desee editar y cuando se cree uno nuevo no mostrara datos ya que no existe un \_id creado.

<Modal isOpen={this.state.modalInsert}>

                <ModalHeader style={{display: 'block'}}>

                  <span style={{float: 'right'}} onClick={()=>this.modalInsert()}>x</span>

                </ModalHeader>

                <ModalBody>

                  <div className="form-group">

                    <label htmlFor="\_id">ID</label>

                    <input className="form-control" type="text" name="id" id="id" readOnly onChange={this.captureData} value={form?form.\_id: this.state.data.length+1}/>

                    <br />

                    <label htmlFor="Name">Name</label>

                    <input className="form-control" type="text" name="name" id="name" onChange={this.captureData} value={form?form.name: ''}/>

                    <br />

                    <label htmlFor="LastName">LastName</label>

                    <input className="form-control" type="text" name="lastName" id="lastName" onChange={this.captureData } value={form?form.lastName: ''}/>

                    <br />

                    <label htmlFor="age">Age</label>

                    <input className="form-control" type="text" name="age" id="age" onChange={this.captureData } value={form?form.age:''}/>

                    <br />

                    <label htmlFor="subTotal">Sub Total</label>

                    <input className="form-control" type="text" name="subTotal" id="subTotal" onChange={this.captureData } value={form?form.subTotal:''}/>

                  </div>

                </ModalBody>

Crear los botones correspondientes a cada acción los mismos que tendrán un evento asignado llamando al método Post o Put cual sea el caso.

<ModalFooter>

                  {this.state.tipoModal=='insertar'?

                    <button className="btn btn-success" onClick={()=>this.callPost()}>

                    Insertar

                  </button>: <button className="btn btn-primary" onClick={()=>this.callPut()}>

                    Actualizar

                  </button>

  }

                    <button className="btn btn-danger" onClick={()=>this.modalInsert()}>Cancelar</button>

                </ModalFooter>

Para finalizar en App.js crear una alerta de para confirmar la eliminación de un registro y al final de la tabla la creación de un botón para crear un nuevo cliente con su respectivo evento.

 <ModalBody>

               Estás seguro que deseas eliminar el cliente: {form && form.name}

            </ModalBody>

            <ModalFooter>

              <button className="btn btn-danger" onClick={()=>this.callDelete()}>Sí</button>

              <button className="btn btn-secundary" onClick={()=>this.setState({modalDelete: false})}>No</button>

            </ModalFooter>

          </Modal>

          <Button

                className='login'

                fullWidth

                variant="contained"

                style={{ color: 'white', backgroundColor: "#5DC1B9" }}

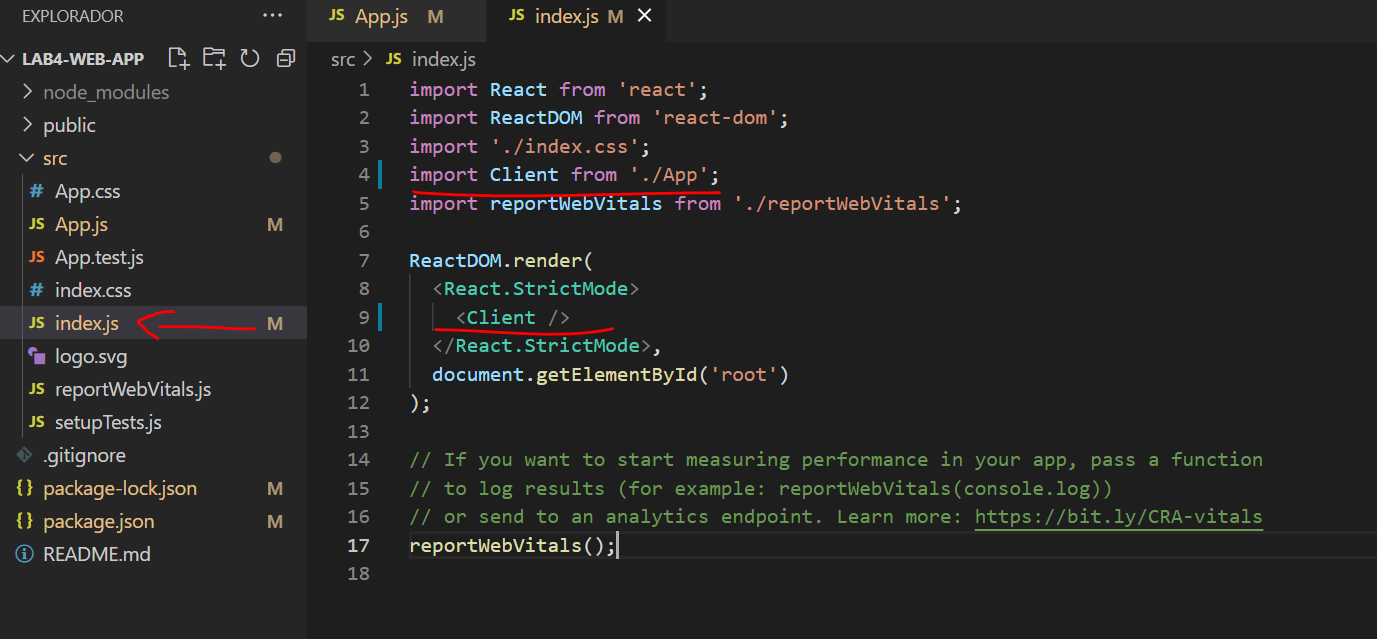
                onClick={()=>{this.setState({form: null, tipoModal: 'insertar'}); this.modalInsert()}}

              >

                <strong><p>Add client</p></strong>

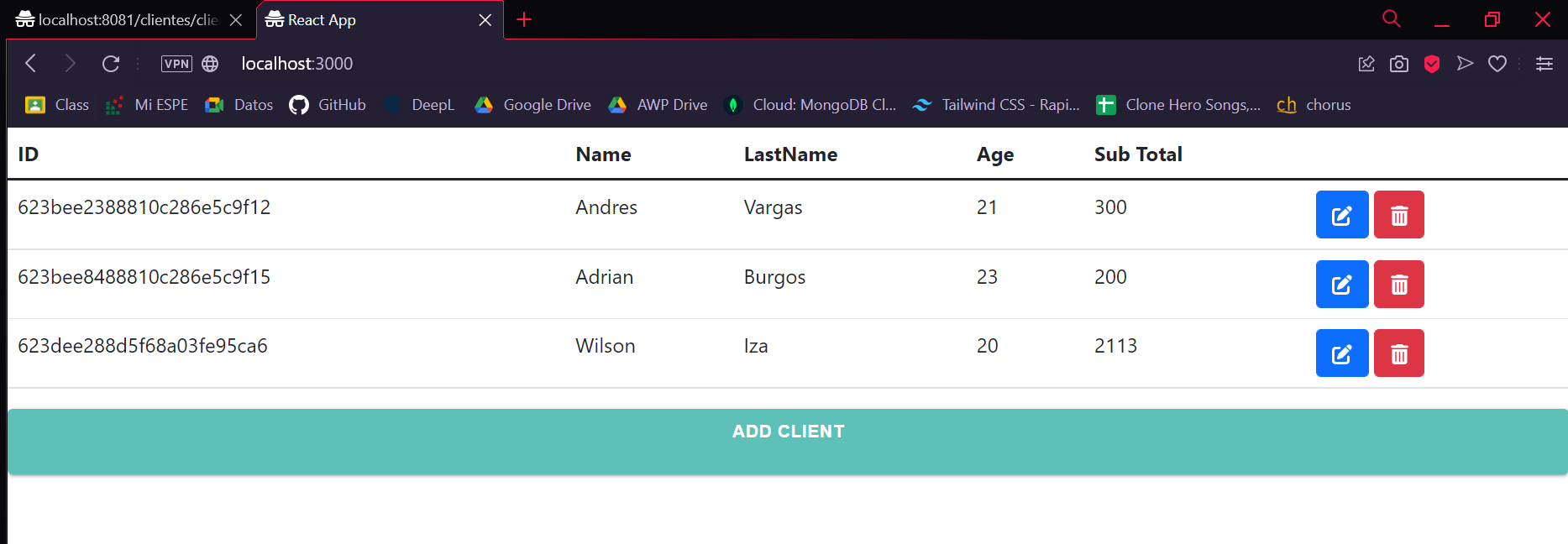
              </Button>

Para finalizar con el desarrollo abrir el archivo index.js que se encuentra en la carpeta raíz del proyecto, importar la clase creara Client y utilizarla para ser mostrada.

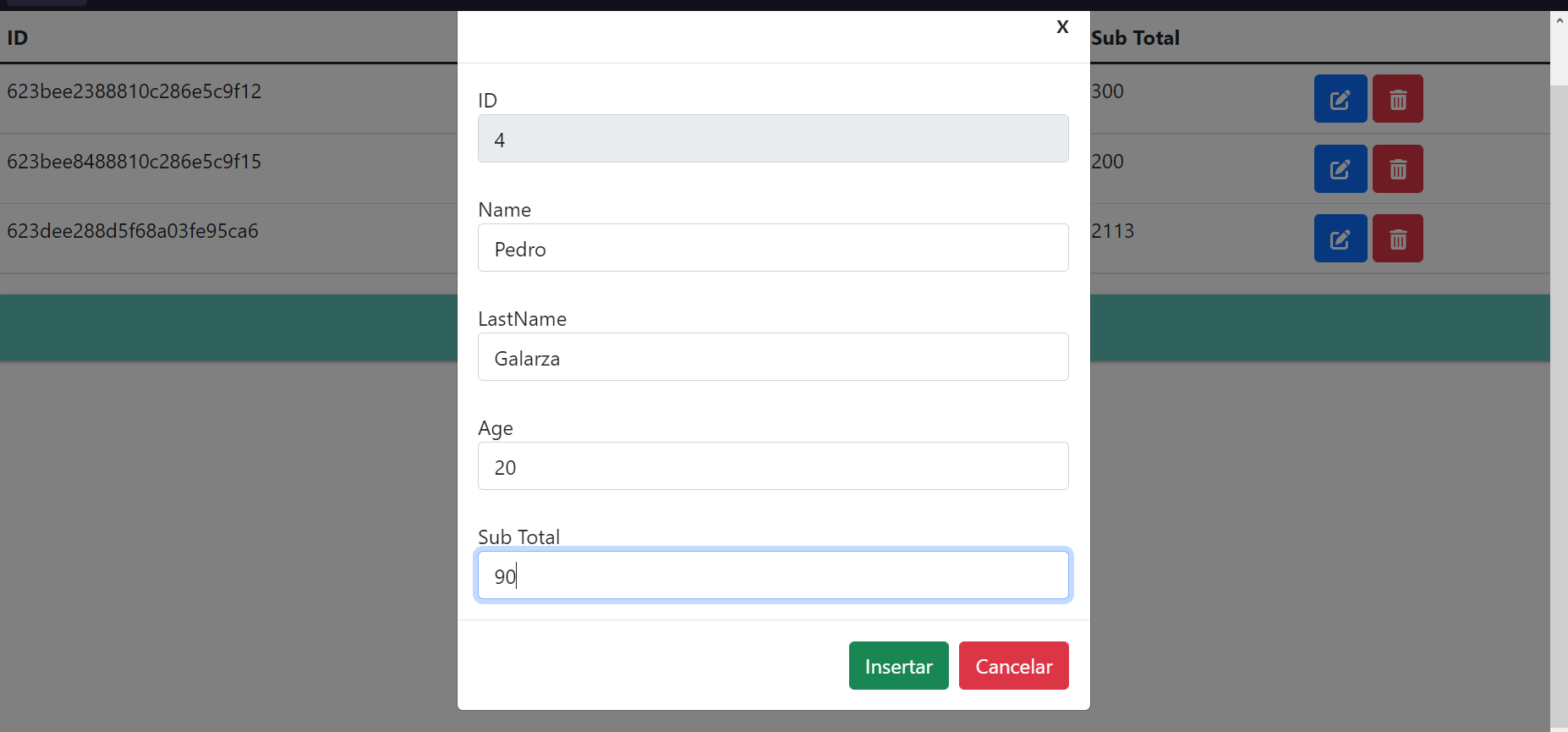


***Figura 22****. Archivo index.js e importación y uso de la clase Client*

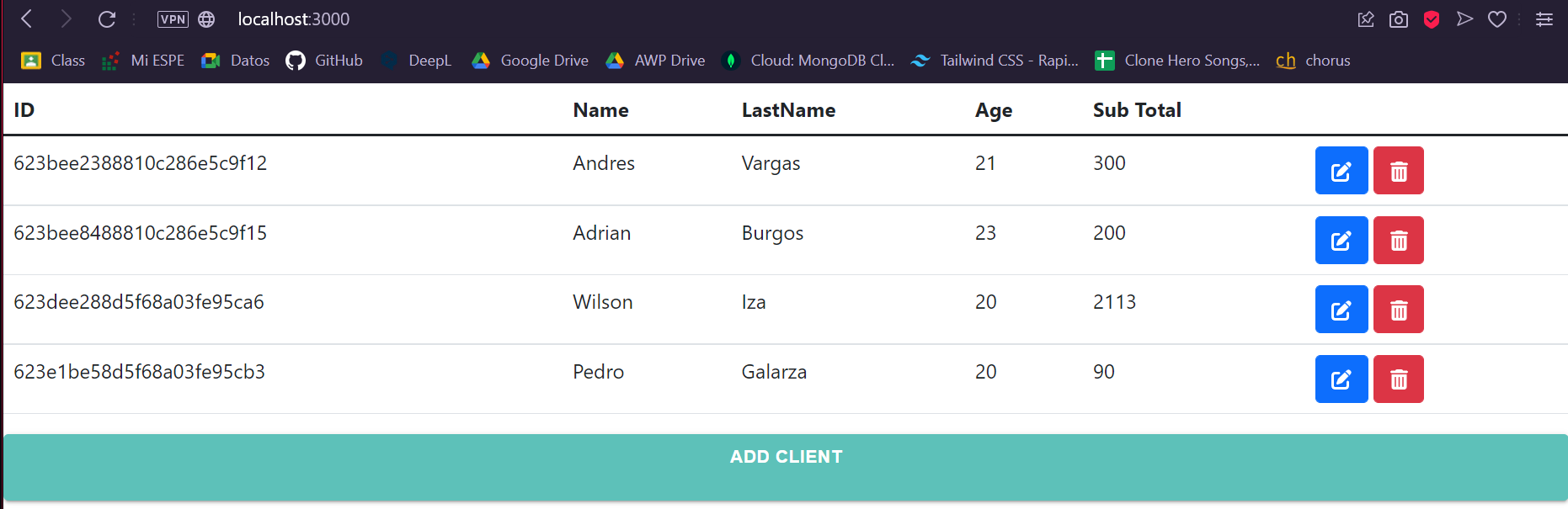
1. **Resultados**



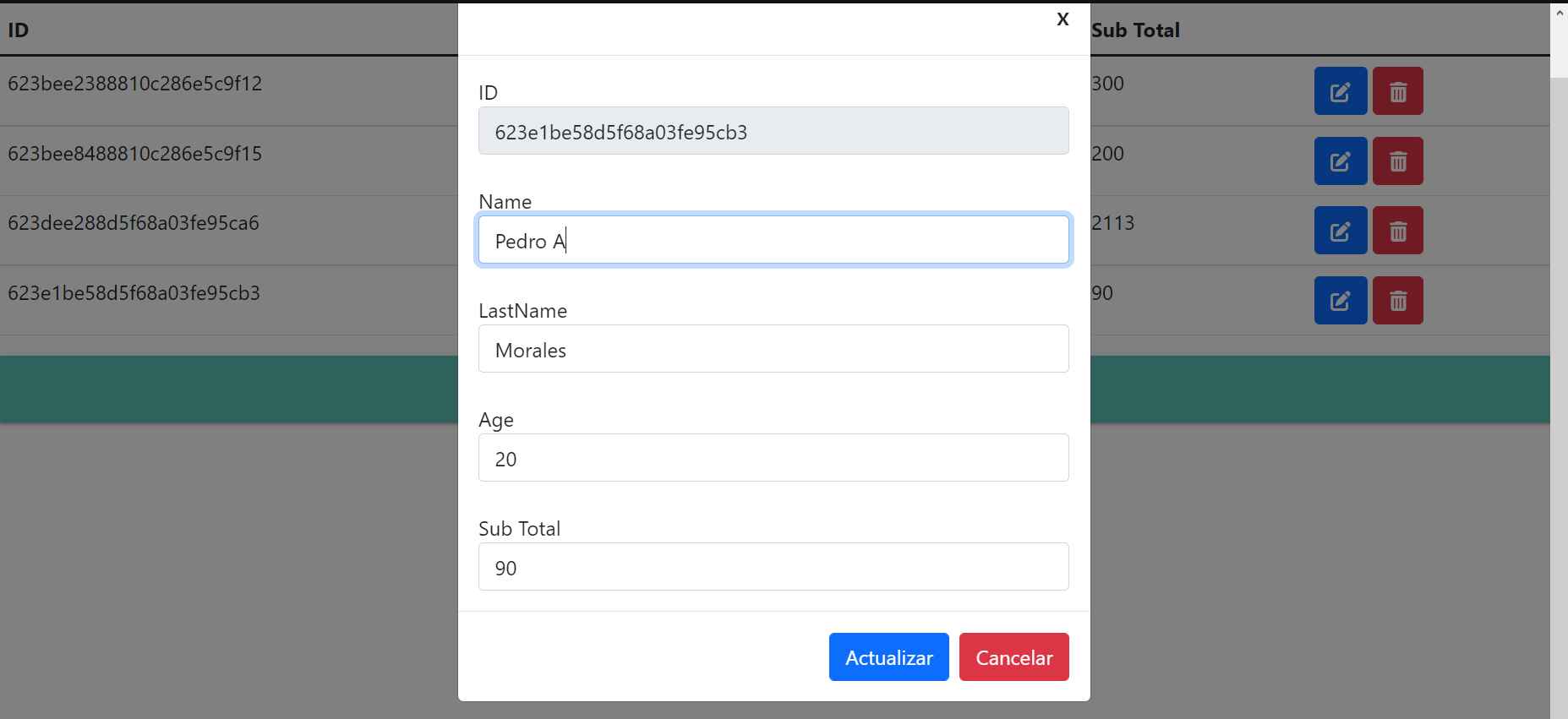
***Figura 23.*** *Obtención de datos (GET)*



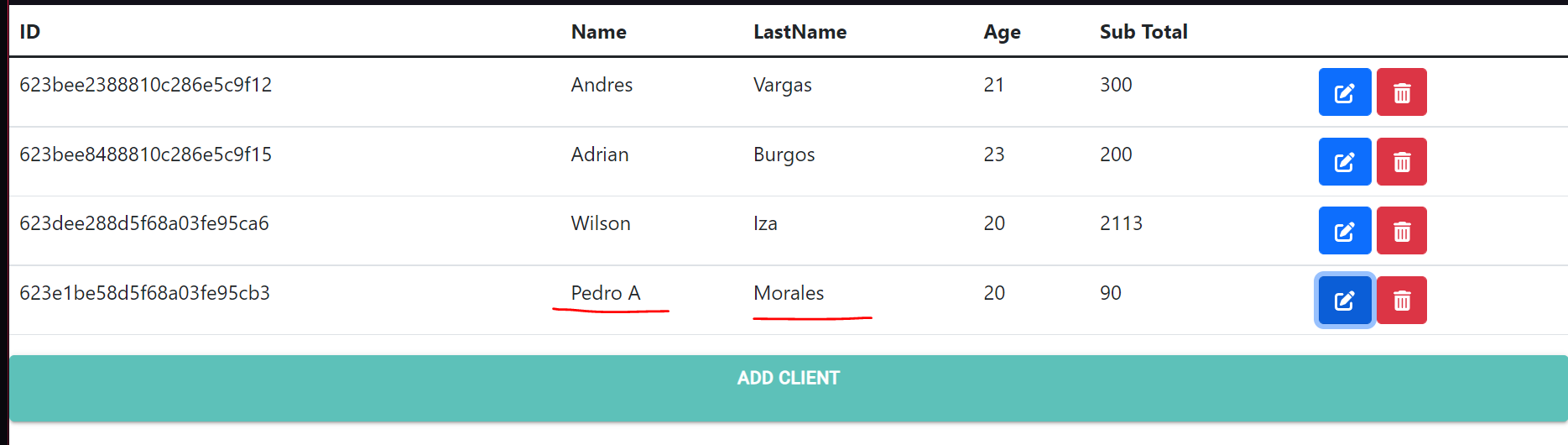
***Figura 24.*** *Agregar un nuevo cliente (método POST)*



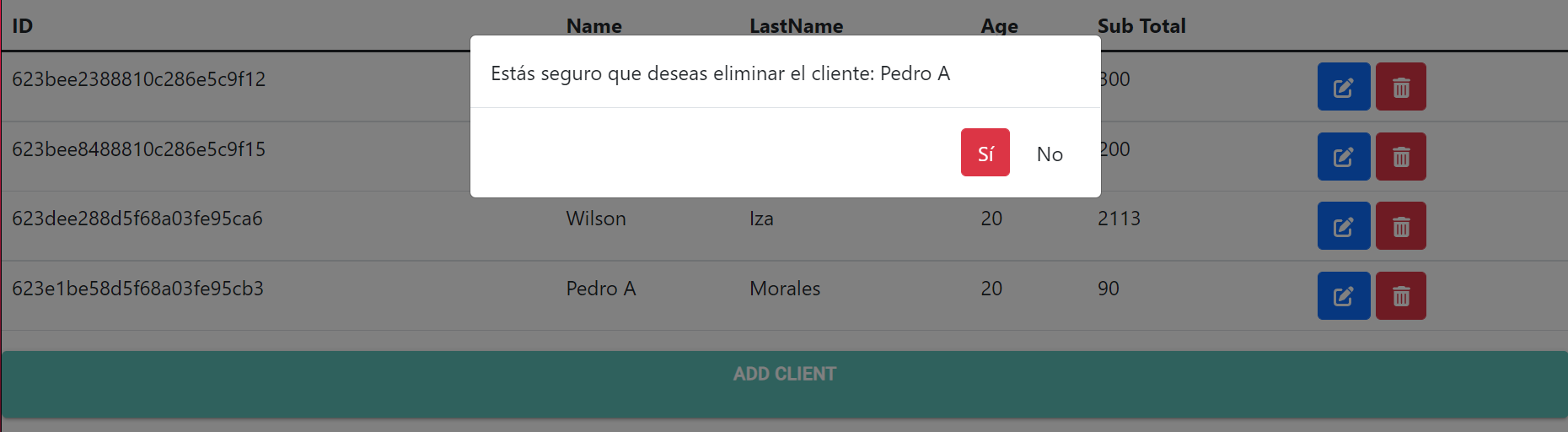
***Figura 25.*** *Comprobación del correcto funcionamiento del método POST*



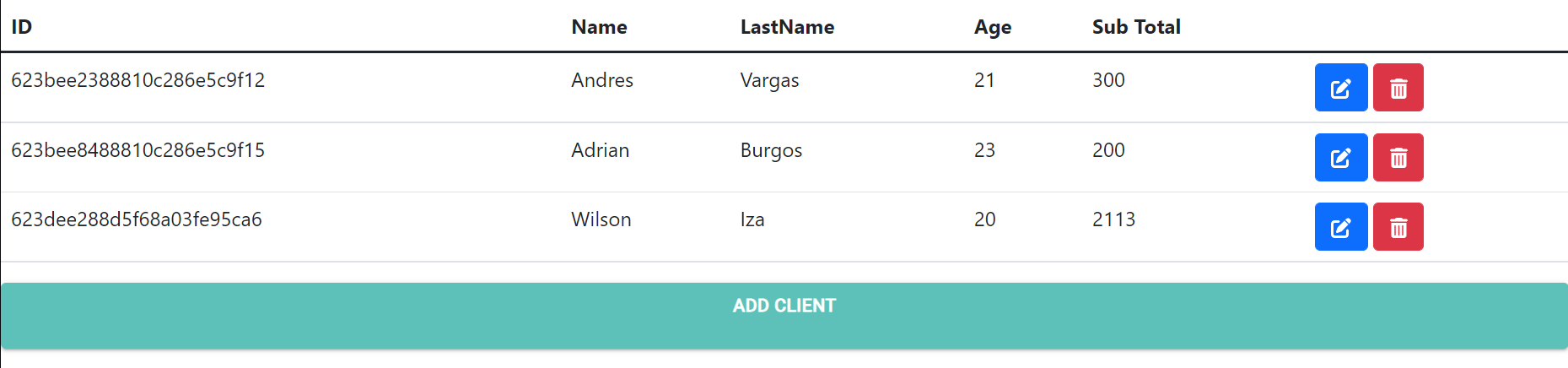
***Figura 26.*** *Modificar un cliente existente (método PUT)*



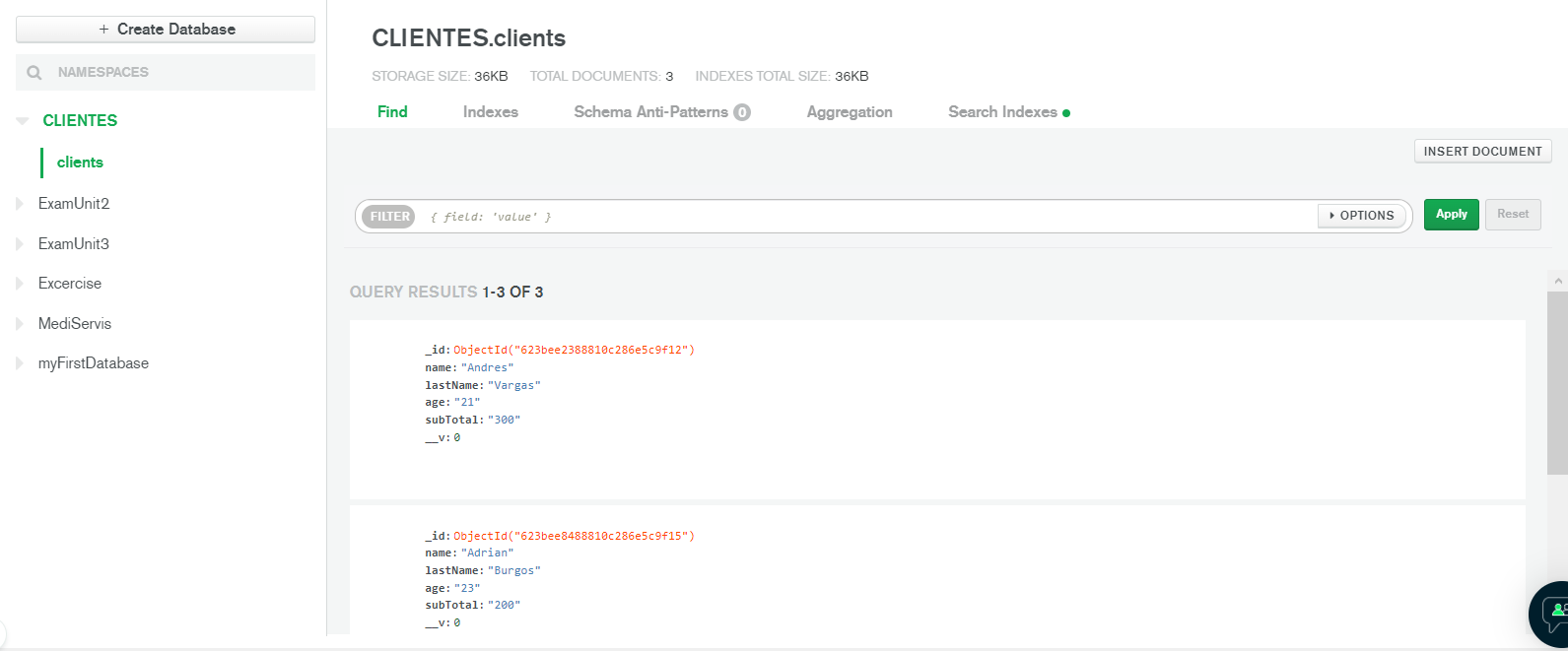
***Figura 27.*** *Comprobación del correcto funcionamiento del método PUT*



***Figura 28.*** *Eliminar un cliente (método Delete)*



***Figura 29****. Comprobación del correcto funcionamiento del método DELETE*



***Figura 30.*** *Persistencia de información en la base de datos en la nube (MongoDB Atlas)*

1. **Recursos**

*Recurso 1:* Servicio restFull utilizado en este proyecto: <https://youtu.be/CQN1ZQ5jVyE>

1. **Enlaces** 
   1. *Video tutorial del desarrollo de esta WebApp*

* <https://www.youtube.com/watch?v=DgJ_0jIDU3s>