Crear Front End

1. Crear archivo html (cualquier nombre para el ejemplo index.html)
2. Importar librerias de bootstrap y alertify

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-KyZXEAg3QhqLMpG8r+8fhAXLRk2vvoC2f3B09zVXn8CA5QIVfZOJ3BCsw2P0p/We" crossorigin="anonymous">

    <link rel="stylesheet" href="css/alertify.min.css" />

La librería es el CDN de bootstrap directamente de la web

La librería de alertify esta en la carpeta css

1. En el cuerpo de la página crear los encabezados de la tabla junto con el boton crear (Se puede ajustar para otro CRUD

 <div class="container mt-4 shadow-lg p-3 mb-5  rounded ">

        <button id="btnCrear" type="button" class="btn btn-primary">Crear Proveedor</button>

        <table id="provedores" class="table mt-2 table-bordered table-striped">

          <thead>

            <tr class="text-center">

              <th>ID</th>

              <th>NOMBRE</th>

              <th>EMAIL</th>

              <th>ACCIONES</th>

            </tr>

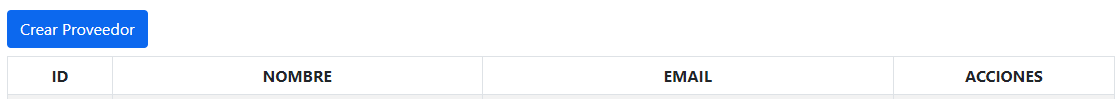
          </thead>

          <tbody>

          </tbody>

        </table>

      </div>



1. Crear el formulario modal que servirá para crear un nuevo registro o para modificar un registro existente. El formulario incluye una dupla de etiqueta y de caja de texto(Se puede ajustar para otro CRUD)

<div class="modal fade" id="modalProveedor" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">

        <div class="modal-dialog">

          <div class="modal-content">

            <div class="modal-header bg-primary text-white">

              <h5 class="modal-title" id="exampleModalLabel">Provvedores</h5>

              <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

            </div>

            <div class="modal-body">

              <form>

                <div class="mb-3">

                  <label for="id" class="col-form-label">Id Proveedor :</label>

                  <input id="id" type="number" placeholder="Digite el identificador" class="form-control" id="recipient-name"

                    required autocomplete="off" oninvalid="setCustomValidity('Digite una Identificación')"

                    oninput="setCustomValidity('')">

                </div>

                <div class="mb-3">

                  <label for="nombre" class="col-form-label">Nombre Proveedor :</label>

                  <input id="nombre" placeholder="Digite el nombre" type="text" class="form-control" id="recipient-name"

                    required autocomplete="off" oninvalid="setCustomValidity('Digite un Nombre')"

                    oninput="setCustomValidity('')">

                </div>

                <div class="mb-3">

                  <label for="email" class="col-form-label">Email Proveedor :</label>

                  <input id="email" placeholder="Digite un email" type="email" class="form-control"

                    id="recipient-name" required autocomplete="off" oninvalid="setCustomValidity('Digite un Email')"

                    oninput="setCustomValidity('')">

                </div>

                <div class="modal-footer">

                  <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Cerrar</button>

                  <button type="submit" class="btn btn-primary">Guardar</button>

                </div>

              </form>

            </div>

          </div>

        </div>

      </div>

Parametros de validación de las cajas de texto

* Required **(Campo requerido)**
* autocomplete="off" **(Deshabilitar autocompletado)**
* oninvalid="setCustomValidity('Digite un Código')" oninput="setCustomValidity('') **(Campo requerido)**

1. incluir librerias de javascript tanto de bootstrap como de alertify y el archivo proveedores.js que hará el crud

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"

      integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"

      crossorigin="anonymous"></script>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.bundle.min.js"

      integrity="sha384-ho+j7jyWK8fNQe+A12Hb8AhRq26LrZ/JpcUGGOn+Y7RsweNrtN/tE3MoK7ZeZDyx"

      crossorigin="anonymous"></script>

    <script src="js/alertify.min.js"></script>

    <script src="js/proveedores.js"></script>

1. Crear archivo proveedores.js
2. Definir el path del API

const url = "http://localhost:8080/proveedor/"

const url1 = "http://localhost:8080/proveedor/list"

1. Seleccionar de la tabla creada en el index.html el <tbody> para enviar la información que se traiga de la base de datos

const contenedor = document.querySelector('tbody')

1. Definir una variable de trabajo almacenar la información que se trae de la base de datos

let resultados = ''

1. Definir variables de trabajo por medio del DOM de la página, la primera modalclientes identifica la venta modal, la segunda formClientes identifica el formulario creado, las siuientes variables indenfican cada uno de los campos del formulario que servirá para insertar y/o modificar. (Se puede ajustar para otro CRUD)

const modalProveedores = new bootstrap.Modal(document.getElementById('modalProveedor'))

const formProvedores = document.querySelector('form')

const tipoCliente = document.getElementById('tipdoc')

const idProveedor = document.getElementById('id')

const nombreProveedor = document.getElementById('nombre')

const emailProveedor = document.getElementById('email')

1. Definir variable que identificará la acción Guardar o Actualizar

let opcion = ''

1. Asingar la acción al boton crear, en este procedimiento se limpia la información de cada una de las cajas de texto (por si se habia utilizado antes en modificar), se habilita la caja de texto del idCliente, se coloca visible la venta modal y se define la opcion a ‘crear’. (Se puede ajustar para otro CRUD)

btnCrear.addEventListener('click', () => {

    idProveedor.value = ''

    nombreProveedor.value = ''

    emailProveedor.value = ''

    idProveedor.disabled = false

    modalProveedores.show()

    opcion = 'crear'

})

1. Se crea la función para mostrar los registro. (Se puede ajustar para otro CRUD)

const mostrar = (Proveedores) => {

    Proveedores.forEach(Proveedor => {

        resultados += `<tr>

                        <td >${Proveedor.idProveedor}</td>

                        <td >${Proveedor.nombreProveedor}</td>

                        <td >${Proveedor.emailProveedor}</td>

                        <td class="text-center" width="20%"><a class="btnEditar btn btn-primary">Editar</a><a class="btnBorrar btn btn-danger">Borrar</a></td>

                    </tr>`

    })

    contenedor.innerHTML = resultados

}

1. Se crea procedimiento para mostrar los registros

fetch(url1)

    .then(response => response.json())

    .then(data => mostrar(data))

    .catch(error => console.log(error))

const on = (element, event, selector, handler) => {

    element.addEventListener(event, e => {

        if (e.target.closest(selector))

            handler(e)

    })

}

1. Se agrega el evento click al resto de botones (btnBorrar y btnEditar) que son los botones creados al recorrer la información que devuelve el API y se almacena en la tabla del archivo html. (Botón Borrar)(Se puede ajustar para otro CRUD)

on(document, 'click', '.btnBorrar', e => {

    const fila = e.target.parentNode.parentNode

    const id = fila.firstElementChild.innerHTML

    console.log(id)

    alertify.confirm("Desea eliminar el Proveedor "+id,

        function () {

            fetch(url + id, {

                method: 'DELETE'

            })

                .then(() => location.reload())

        },

        function () {

            alertify.error('Cancel')

        });

})

Si el boton presionado es el de eliminar se lanza una ventana emergente que pregunta si realmente se desea eliminar el registro, si se responde que sí se lanza el API que llama el DELETE, si se presiona que no, no se realiza nada.

Si el boton presionado es editar se envia la información asociada en la fila a las cajas de texto ,se coloca visible la ventana modal , se deshabilita la caja de texto idProveedor y se define la opcion como “editar”

let idForm = 0

on(document, 'click', '.btnEditar', e => {

    const fila = e.target.parentNode.parentNode

    idForm = fila.children[0].innerHTML

    const nombre = fila.children[1].innerHTML

    const email = fila.children[2].innerHTML

    idProveedor.value = idForm

    idProveedor.disabled = true

    nombreProveedor.value = nombre

    emailProveedor.value = email

    opcion = 'editar'

    modalProveedores.show()

})

Esta instrucción selecciona la fila asociada al boton presionado (Explicación)

const fila = e.target.parentNode.parentNode

luego se acceder a cada uno de los elementos que serían los hijos de la fila ejem: (Explicación)

idForm.value = fila.children[0].innerHTML

1. Se programa la accion del boton guardar que de acuerdo con la opción “guardar” o “Editar” llama el API correspondiente PUT o POST. (Se puede ajustar para otro CRUD)

formProvedores.addEventListener('submit', (e) => {

    e.preventDefault()

        if (opcion == 'crear') {

            fetch(url, {

                method: 'POST',

                headers: {

                    'Content-Type': 'application/json'

                },

                body: JSON.stringify({

                    idProveedor: idProveedor.value,

                    nombreProveedor: nombreProveedor.value,

                    emailProveedor: emailProveedor.value

                })

            })

                .then(response => response.json())

                .then(data => {

                    const nuevoProveedor = []

                    nuevoProveedor.push(data)

                    mostrar(nuevoProveedor)

                })

        }

        if (opcion == 'editar') {

            fetch(url, {

                method: 'PUT',

                headers: {

                    'Content-Type': 'application/json'

                },

                body: JSON.stringify({

                    idProveedor: idForm,

                    nombreProveedor: nombreProveedor.value,

                    emailProveedor: emailProveedor.value

                })

            })

                .then(response => location.reload())

        }

        modalProveedores.hide()

})

Para cualquiera de las dos opciones se envía la información que se encuentra almacenada en las cajas de texto y se envian a través de la variable data (la información viaja en formato JSON).