Herramientas de Programación 2

Tema Nº13:IMPLEMENTACIÓN DEL PROYECTO REACT II

Indicador de logro Nº13:Realiza formularios web utilizando las estructuras REACT para la implementación del proyecto de la UD.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

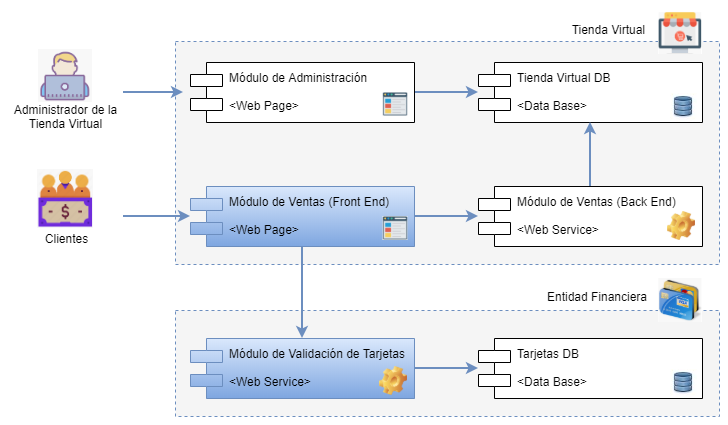
Descripción generada automáticamente

**TEMA Nº13:**

IMPLEMENTACIÓN DEL PROYECTO REACT II

**Subtema 13.1:**

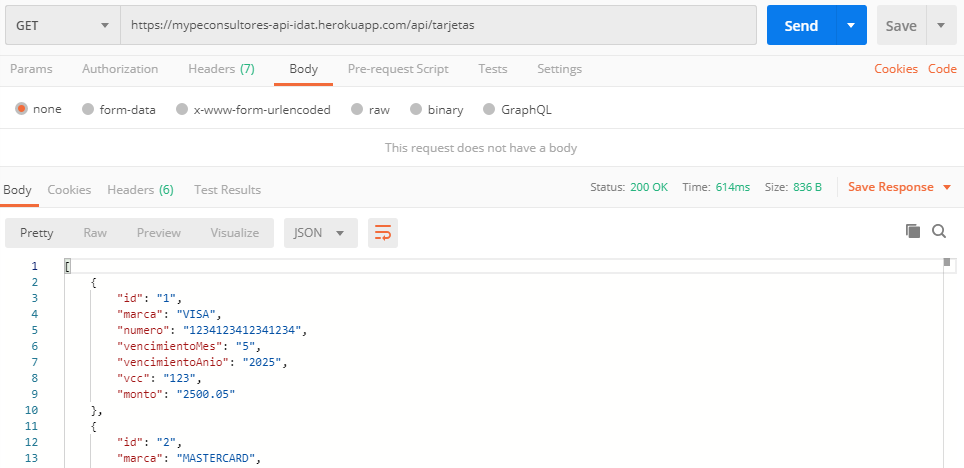
Módulo de Ventas



Paso 1. Verificar y probar el servicio externo que valida las tarjetas.

1. Abrir Postman para configurar el consumo del servicio GET.
2. Para obtener la lista de tarjetas que tiene registrada el servicio hacemos una llamada GET a la siguiente Url:

<https://mypeconsultores-api-idat.herokuapp.com/api/tarjetas>



1. Ahora abrimos una nueva pestana en Postman para configurar la llamada POST que realizará la validación de las tarjetas.
2. Primero configuramos una llamada POST a la siguiente Url:

https://mypeconsultores-api-idat.herokuapp.com/api/validarTarjeta

1. Configuramos el siguiente request (Json) en la sección del Body:

{

"id": 0,

"marca": "MASTERCARD",

"numero": "1111222233334444",

"vencimientoMes": "11",

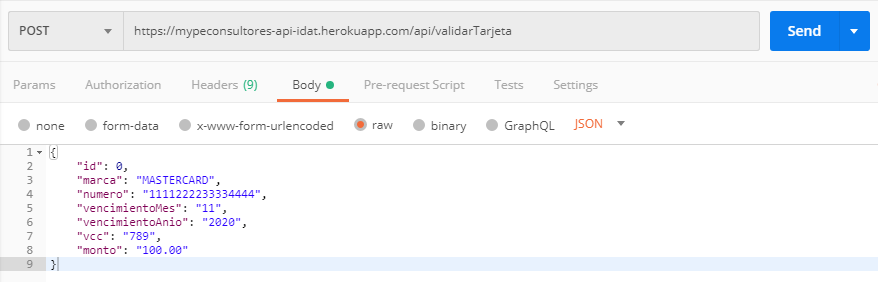
"vencimientoAnio": "2020",

"vcc": "789",

"monto": "100.00"

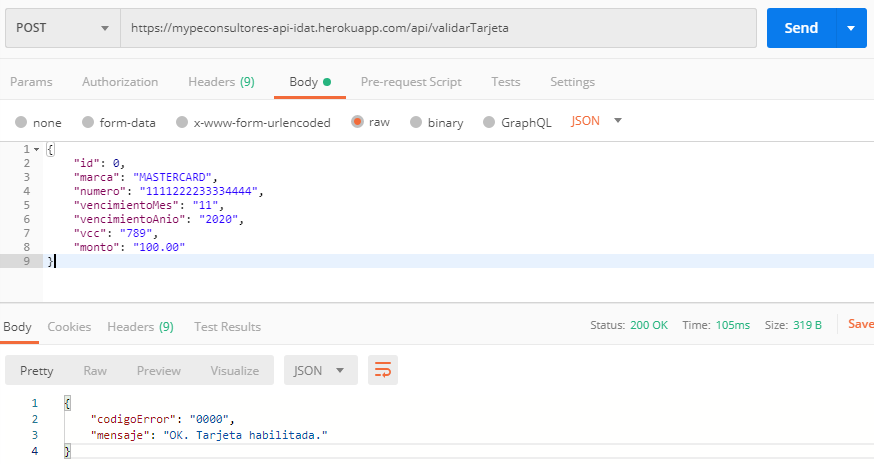
}

1. Seleccionamos el formato: JSON.
2. Para esta configuración el Postman debería estar de acuerdo a:

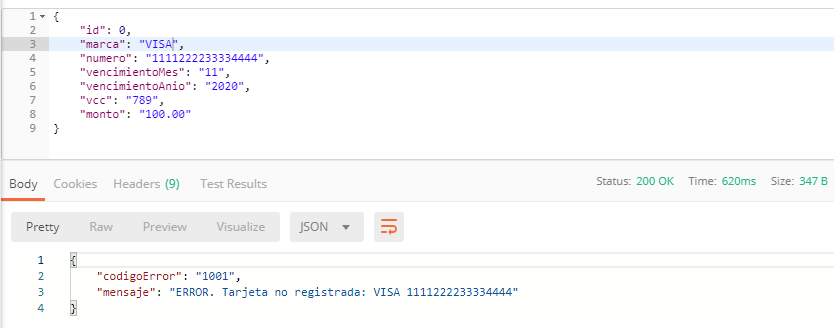


1. Presionamos el botón *Send*.

Obtenemos como resultado una respuesta satisfactoria, esto lo podemos comprobar con las tarjetas que obtuvimos en la llamada GET.



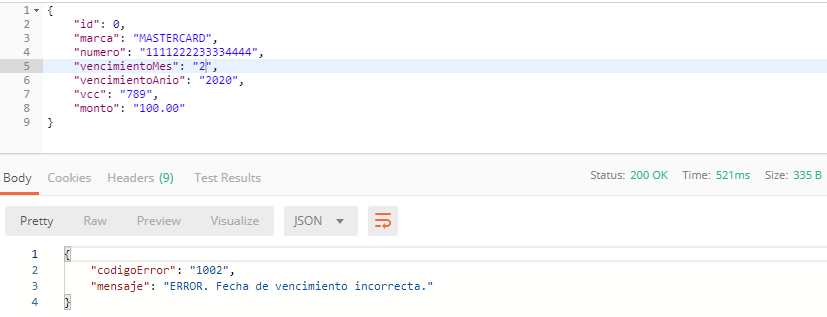
1. Si en el *request* modificamos el valor de algunos datos obtendremos diversos mensajes de error, por ejemplo si cambiamos la *marca* obtendremos:



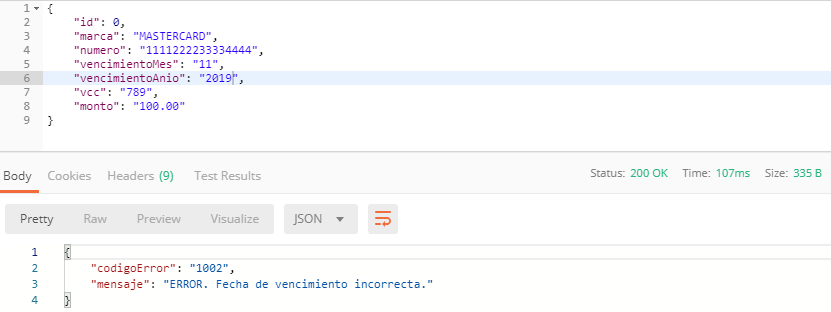
1. Si cambiamos el *numero*:



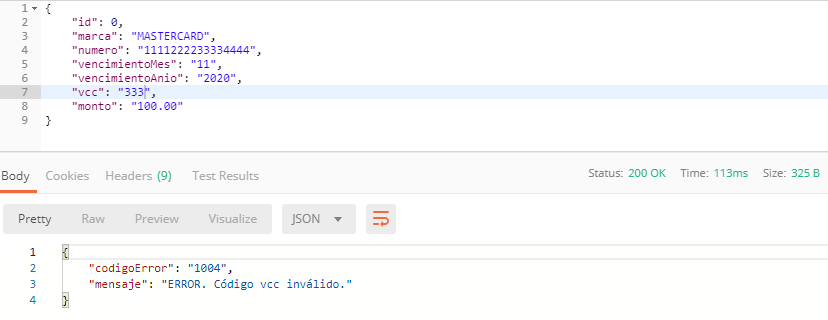
1. Si cambiamos el dato *vencimientoMes*:



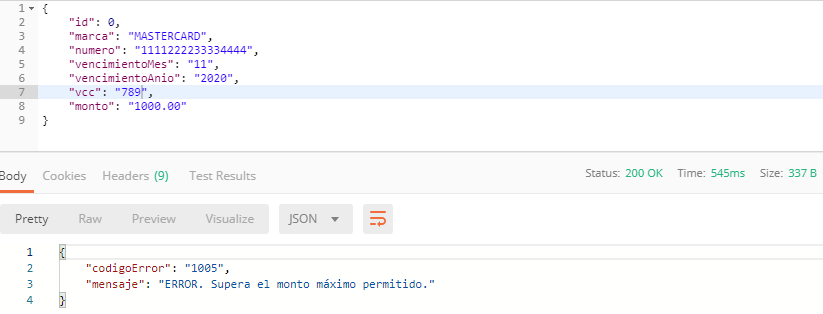
1. Si cambiamos el dato *vencimientoAnio*:



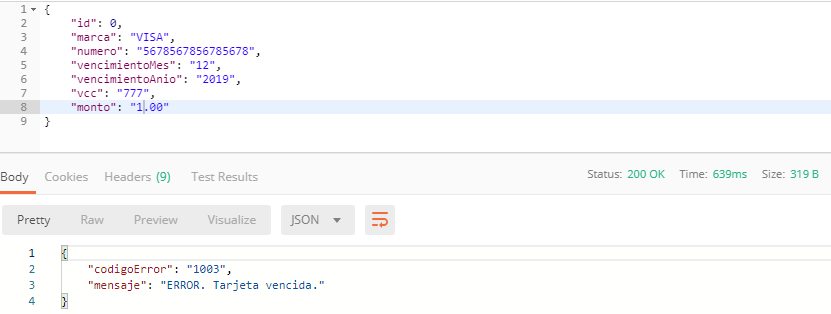
1. Si cambiamos el dato vcc (código de verificación de la tarjeta):



1. Si ponemos un monto que supera el monto de la tarjeta:



1. Si ponemos los datos de una tarjeta que ya venció:



Paso 2. Crear un formulario que permita ingresar los datos de la tarjeta y validar a través del consumo del servicio externo (POST).

1. Abrir el proyecto React de la semana anterior.
2. Agregamos el componente *CompFormAgregarTarjeta.js* y modificaos su código e acuerdo a:

import React, { Component } from 'react';

class CompFormAgregarTarjeta extends Component {

    state = {

        id: "0",

        marca: "",

        numero: "",

        vencimientoMes: "",

        vencimientoAnio: "",

        vcc: "",

        monto: "100.00",

        mensajeErrorNumero: "",

        mensajeErrorVcc: "",

        resultadoApiPost: {

            codigoError: "",

            mensaje: ""

        }

    }

    actualizarEstado = e => {

        if (e.target.name == "cmbMarca") {

            this.setState({

                marca: e.target.value

            })

        } else if (e.target.name == "txtNumero") {

            this.setState({

                numero: e.target.value.replace('-', '').replace(' ', '')

            })

        } else if (e.target.name == "cmbMes") {

            this.setState({

                vencimientoMes: e.target.value

            })

        } else if (e.target.name == "cmbAnio") {

            this.setState({

                vencimientoAnio: e.target.value

            })

        } else if (e.target.name == "txtVcc") {

            this.setState({

                vcc: e.target.value

            })

        }

    }

    validarDatosCliente = e => {

        this.setState({

            mensajeErrorNumero: "",

            mensajeErrorVcc: ""

        })

        if (this.state.numero == null || this.state.numero.length == 0 || /^\s+$/.test(this.state.numero)) {

            this.setState({

                mensajeErrorNumero: "Campo obligatorio"

            })

        } else if (this.state.numero.length !== 16) {

            this.setState({

                mensajeErrorNumero: "Ingrese 16 números"

            })

        } else if (this.state.vcc == null || this.state.vcc.length == 0 || /^\s+$/.test(this.state.vcc)) {

            this.setState({

                mensajeErrorVcc: "Campo obligatorio"

            })

        } else if (this.state.vcc.length !== 3) {

            this.setState({

                mensajeErrorVcc: "Ingrese 3 números"

            })

        } else {

            console.log(this.state);

            fetch("https://mypeconsultores-api-idat.herokuapp.com/api/validarTarjeta", {

                method: "POST",

                body: JSON.stringify(this.state),

                headers: {

                    "Content-type": "application/json"

                }

            })

                .then(respuestaPost => respuestaPost.json())

                .then(resultadoApiPost => this.setState({ resultadoApiPost }));

            console.log(this.state.resultadoApiPost);

        }

        e.preventDefault();

    }

    render() {

        return <form onSubmit={this.validarDatosCliente}>

            <p className="h5 text-secondary">Datos de la Tarjeta</p>

            <div className="form-row">

                <div className="form-group col-md-6">

                    <label for="cmbMarca">\*Marca:</label>

                    <select className="form-control" name="cmbMarca" id="cmbMarca"

                        value={this.state.marca} onChange={this.actualizarEstado}>

                        <option value="VISA">VISA</option>

                        <option value="MASTERCARD">MASTERCARD</option>

                    </select>

                </div>

                <div className="form-group col-md-3">

                    <label for="cmbMes">Vencimiento (mes):</label>

                    <select className="form-control" name="cmbMes" id="cmbMes"

                        value={this.state.vencimientoMes} onChange={this.actualizarEstado}>

                        <option value="1">1</option>

                        <option value="2">2</option>

                        <option value="3">3</option>

                        <option value="4">4</option>

                        <option value="5">5</option>

                        <option value="6">6</option>

                        <option value="7">7</option>

                        <option value="8">8</option>

                        <option value="9">9</option>

                        <option value="10">10</option>

                        <option value="11">11</option>

                        <option value="12">12</option>

                    </select>

                </div>

                <div className="form-group col-md-3">

                    <label for="cmbAnio">Vencimiento (año):</label>

                    <select className="form-control" name="cmbAnio" id="cmbAnio"

                        value={this.state.vencimientoAnio} onChange={this.actualizarEstado}>

                        <option value="2019">2019</option>

                        <option value="2020">2020</option>

                        <option value="2021">2021</option>

                        <option value="2022">2022</option>

                        <option value="2023">2023</option>

                        <option value="2024">2024</option>

                        <option value="2025">2025</option>

                    </select>

                </div>

            </div>

            <div className="form-row">

                <div className="form-group col-md-6">

                    <label for="txtNumero">\*Número:</label>

                    <input type="text" className="form-control" name="txtNumero" id="txtNumero"

                        value={this.state.numero} onChange={this.actualizarEstado} />

                    <div className="badge badge-danger">{this.state.mensajeErrorNumero}</div>

                </div>

                <div className="form-group col-md-3">

                    <label for="txtVcc">Vcc (código de verificación):</label>

                    <input type="text" className="form-control" name="txtVcc" id="txtVcc"

                        value={this.state.vcc} onChange={this.actualizarEstado} />

                    <div className="badge badge-danger">{this.state.mensajeErrorVcc}</div>

                </div>

                <div className="form-group col-md-3">

                    <label for="txtMonto">Monto del pedido:</label>

                    <input type="text" className="form-control" name="txtMonto" id="txtMonto"

                        value={this.state.monto} disabled />

                </div>

            </div>

            <div className="form-row">

                <div className="form-group col-md-6">

                    <span className="text-secondary">{this.state.resultadoApiPost.mensaje}</span>

                </div>

                <div className="form-group col-md-6 text-right">

                    <button type="submit" className="btn btn-primary">Agregar</button>

                </div>

            </div>

        </form>

    }

}

export default CompFormAgregarTarjeta;

1. Agregamos el componente *PagePedidoTarjeta.js* y modificaos su código e acuerdo a:

import React, {Component} from 'react';

import CompCabecera from './CompCabecera';

import CompFormAgregarTarjeta from './CompFormAgregarTarjeta';

import CompPiePagina from './CompPiePagina';

class PagePedidoCliente extends Component {

  render(){

    return  <div >

                <CompCabecera />

                <main role="main" className="container">

                    <CompFormAgregarTarjeta />

                </main>

                <CompPiePagina />

            </div>

  }

}

export default PagePedidoCliente;

1. Para no obtener errores modificamos el código del componente *CompCabecera.js* de acuerdo a:

import React, { Component } from 'react';

import { Link } from 'react-router-dom';

class CompCabecera extends Component {

    render() {

        return <nav className="navbar navbar-expand-md navbar-dark bg-dark mb-4">

            <div className="container">

                <span className="float-left">

                    <Link className="navbar-brand" to="/">

                        <img src="imagenes/logo.png" width="250" height="35" alt="" />

                    </Link>

                </span>

                <span className="float-right">

                    <div className="collapse navbar-collapse" id="navbarText">

                        <img src="imagenes/carrito.png" width="32" height="32" alt="" />

                        <Link className="text-secondary"

                            to={{

                                pathname: "/listaProductos",

                                search: "",

                                hash: "",

                                state: { idCategoria: 0 }

                            }} >

                            &nbsp;0 producto(s) - S/ 0.00

                        </Link>

                    </div>

                </span>

            </div>

        </nav>

    }

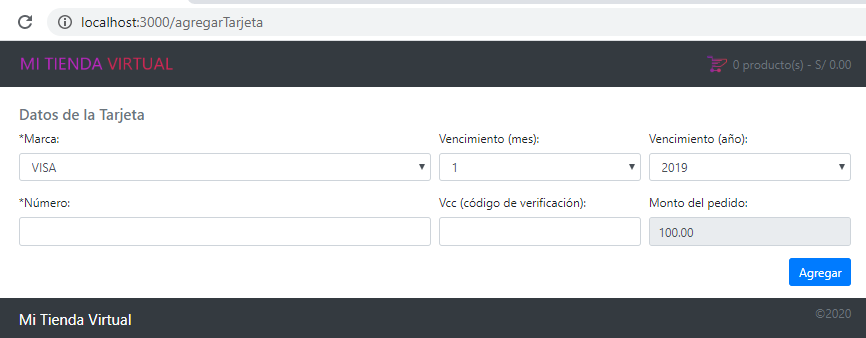
}

export default CompCabecera;

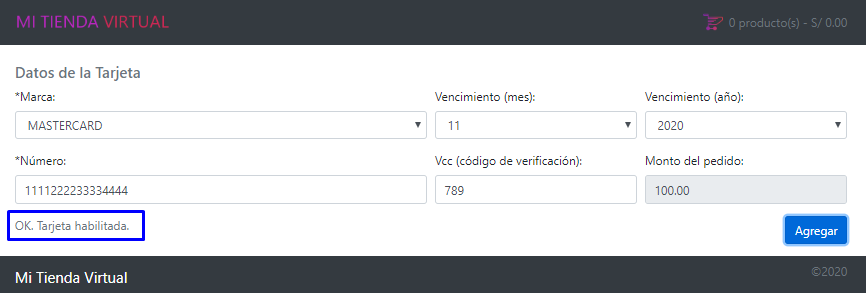
1. Modificamos el *Router* en el archivo *App.js* para que el nuevo componente *PagePedidoTarjeta.js* se muestre cuando se resuelva la Url:

<http://localhost:3000/agregarTarjeta>

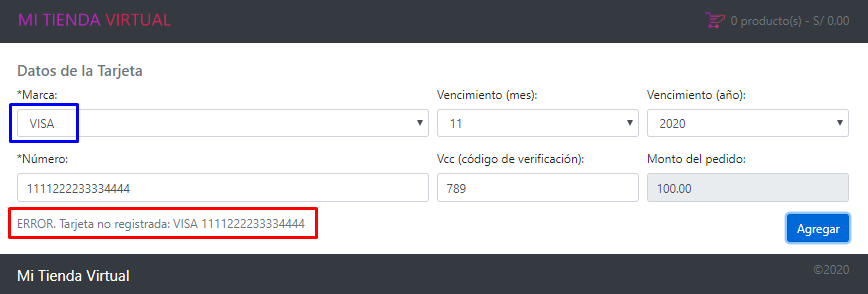
1. Ejecutamos la aplicación y resolvemos <http://localhost:3000/agregarTarjeta>



1. Ingresamos los datos de una tarjeta habilitada y presionamos en el boton Agregar, obtenemos:



1. El mensaje que obtenemos es el mensaje que nos devuelve el servicio POST que vimos en el ejemplo anterior, por lo que podemos hacer los mismos ejercicios para obtener los mensajes de error.



ACTIVIDAD VIRTUAL:

1. CUESTIONARIO TÉCNICO
   * 1. Crear los componentes necesarios de tal forma que si la tarjeta consultada es válida se muestre enseguida una página con todos los datos de la tarjeta consultada.
     2. En el proyecto Back End (Visual Studio) crear el controlador *ClienteController.cs* que permita, mediante una llamada POST, agregar los datos del Cliente al pedido. El *id* del Pedido debe enviarse como parámetro.
     3. Probar con Postman el servicio creado en el ejercicio anterior (ii).
     4. Modificar el componente React *CompFormAgregarCliente.js* de tal forma que consuma el servicio del segundo ejercicio (ii) y así se pueda agregar los datos del Cliente a pedido.
     5. Investigar que otro servicio externo se puede incluir en proyecto React (Front End).
2. CONCLUSIONES DE LA EXPERIENCIA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ACTIVIDAD** **VIRTUAL**:

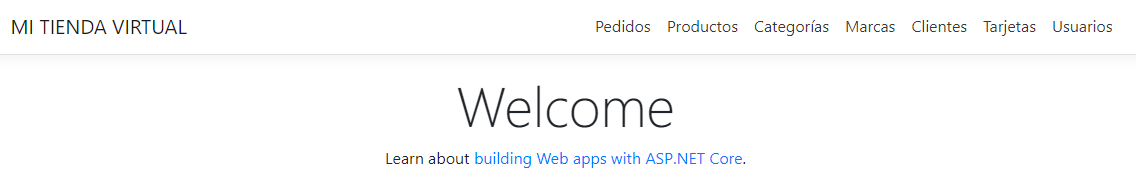
Ingresa a la plataforma virtual. Revisa y analiza el tema desarrollado en esta sesión, luego desarrolla las siguientes actividades propuestas:

1. CUESTIONARIO TÉCNICO
   * 1. Crear con Scaffold Controladores y vistas para el mantenimiento (CRUD) de las siguientes entidades (Tablas):

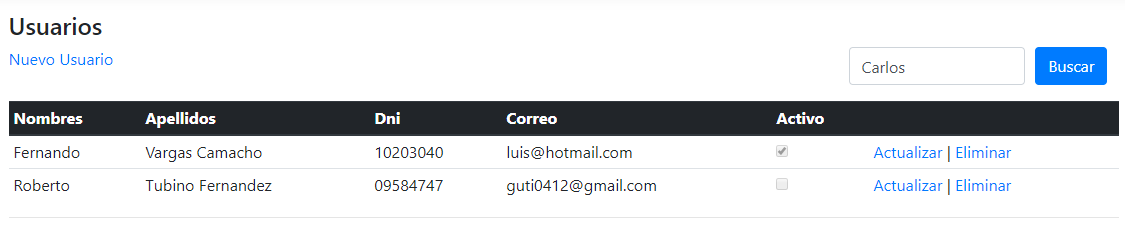
* Marca
* Categoría
* Producto
  + 1. Crear con Scaffold Controladores y vistas para el mantenimiento (CRUD) de las siguientes entidades (Tablas):
* Cliente
* Tarjeta

Estas dos entidades deben ser de solo lectura, esto es, no debe permitir crear, actualizar ni eliminar registros.

* + 1. Para los nuevos mantenimientos, desarrollar las personalizaciones de forma análoga al CRUD de Usuarios.
    2. Agregar en el menú superior links que permitan acceder a los nuevos mantenimientos. Además, alinear el menú a la derecha.



* + 1. Investigar la forma de agregar la funcionalidad de búsqueda en la lista de Usuarios, debe buscar por Nombres y Apellidos.



1. CONCLUSIONES DE LA EXPERIENCIA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Un motor paso a paso siempre debemos considerar su corriente de operación y los grados, como indica el ejemplo de la siguiente placa característica de la imagen mostrada:

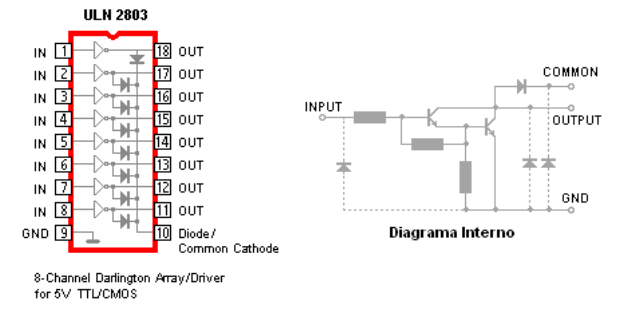


**Subtema 1.2:**

Transistores de potencia

Para operaciones con motores de bajo consumo tenemos circuitos integrados como el circuito ULN2803, el cual es un driver con transistores darlintong , con soporte de hasta 500 mA.

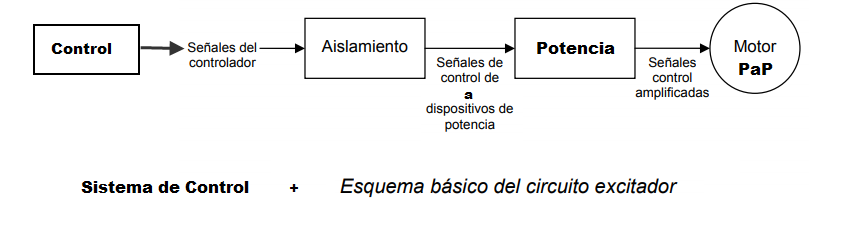
Por lo tanto si deseamos mayores corrientes debemos utilizar otros drivers o implementar uno concomponentes discretos , es decir con transistores de potencia.



**Subtema 1.3:**

Control del Motor

Las etapas para controlar un motor Paso a Paso



**Actividad:**

Los estudiantes desarrollarán ejercicios de Teoría de exponentes y radicales, propuestos por el docente. según lo aprendido en clase. a) CUESTIONARIO TÉCNICO

• ¿Qué diferencia hay entre motores paso a paso unipolares y bipolares?

• ¿Por qué es importante usar driver para motores paso a paso?

• ¿Qué características posee un motor paso a paso?

• ¿Para qué me sirve el puente H?

• Investigue sobre cómo controlar los motores paso a paso con PWM y LOGO! 8.