Herramientas de Programación 2

Tema Nº11:

IMPLEMENTACIÓN DEL PROYECTO REACT I (PARTE 1)

Indicador de logro Nº11:Aplica los conocimientos adquiridos con REACT en la implementación del proyecto de la UD.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº11:**

IMPLEMENTACIÓN DEL PROYECTO REACT I (PARTE 1)

**Subtema 11.1:**

STATE

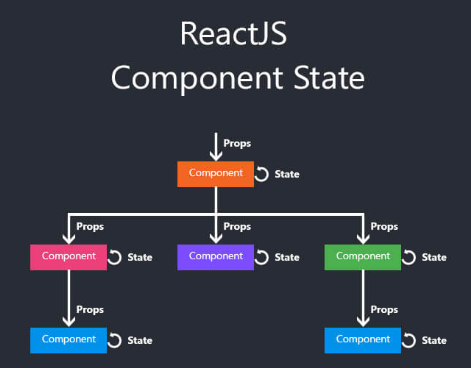
Además de las *props*, los componentes en React pueden mantener un estado (state).

Lo característico del estado es que si éste cambia, el componente se renderiza automáticamente.

Algo que podemos hacer con los estados es mantener información de cada componente y manejar esa información es decir cambiando de estado esa información cuando existe un comportamiento del usuario es decir un click en un botón o todo lo que llame a la acción del usuario.

¿Qué hace setState?

setState() programa una actualización al objeto estado de un componente. Cuando el estado cambia, el componente responde volviendo a renderizar.



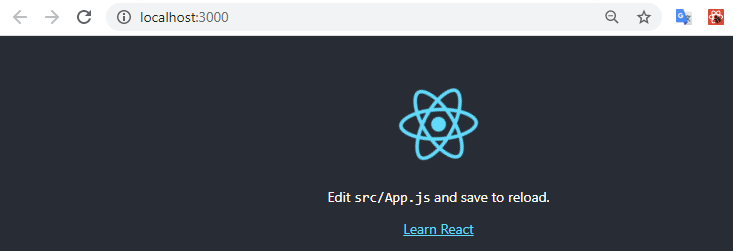
Paso 1. Declaración y actualización del estado (state) de un componente.

1. Creamos en Visua Studio Code el proyecto React *ejemplos\_state\_react*.

(*npx create-react-app ejemplos\_state\_react*)

1. Ejecutamos la aplicación.

(*npm start*)



1. Abrios el nuevo proyecto en Visual Studio Code.
2. Dentro de la carpeta *src* creamos la carpeta *Componentes*.
3. Dentro de la carpeta *Componentes* creamos el componente *ejemplo1.js*
4. Modificamos el código (jsx) del componente *ejemplo1.js* de acuerdo a:

import React, {Component} from 'react';

class Ejemplo1 extends Component {

    render(){

        return <div>

        <h1>Bienvenido</h1>

    </div>

    }

  }

  export default Ejemplo1;

1. Modificamos el código (jsx) del componente *App.js* de acuerdo a:

import React, {Component} from 'react';

import Ejemplo1 from './Componentes/ejemplo1'

class App extends Component {

  render(){

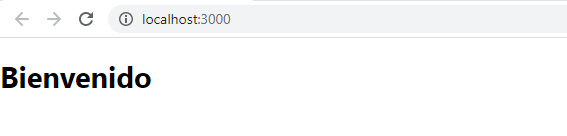
      return <Ejemplo1 />

    }

}

export default App;

1. Ejecutamos la aplicación.



1. Agregaremos código para analizar la utilidad de state.
2. Modificamos el código (jsx) del componente *ejemplo1.js* de acuerdo a:

import React, { Component } from 'react';

class Ejemplo1 extends Component {

    //declaracion simple de state:

    /\*

    state = {

        mensaje: "Bienvenido"

    }

    \*/

    constructor() {

        super()

        this.state = {

            mensaje: "Bienvenido",

            contador: 0

        }

    }

    cambiarMensaje() {

        this.setState({

            mensaje: "Hasta pronto",

            contador: this.state.contador + 1

        })

    }

    render() {

        return (

            <div>

                <h1>{this.state.mensaje}</h1>

                <h3>{this.state.contador}</h3>

                <button onClick={() => this.cambiarMensaje()}>Cambiar mensaje</button>

            </div>

        )

    }

}

export default Ejemplo1;

1. Vemos los cambios en el explorador web.





Cerramos el explorador web y el proyecto en Visual Studio Code.

Paso 2. Crear datos en formato Json: Pedido y Detalle de un Pedido.

1. Abrimos en Visual Studio Code el proyecto React del taller de la semana anterior (*mitiendavirtual\_frontend*).
2. Ejecutamos la aplicación para verificar que no existan errores.
3. Dentro de la carpeta DataJson creamos el archivo *dataJsonPedido.json*. Dentro de este archivo agregamos datos en formato Json de acuerdo a:

{

    "id": 385,

    "idCliente": 3,

    "idTarjeta": 12,

    "fechaHora": "",

    "estado": "EN PROCESO",

    "total": 496.00

}

1. Dentro de la carpeta DataJson creamos el archivo *dataJsonListaDetallePedido.json*. Dentro de este archivo agregamos datos en formato Json de acuerdo a:

[

    {

        "id": 1001,

        "idPedido": 385,

        "idProducto": 23,

        "cantidad": 2.00,

        "precioUnitario": 89.00,

        "subTotal": 178.00,

        "producto": {

            "id": 23,

            "idCategoria": 2,

            "idMarca": 1,

            "nombre": "POLO DOODLE BADGE",

            "precio": 89.00,

            "categoria": {

                "id": 2,

                "nombre": "POLOS"

            },

            "marca": {

                "id": 2,

                "nombre": "ADIDAS"

            }

        }

    },

    {

        "id": 1002,

        "idPedido": 385,

        "idProducto": 23,

        "cantidad": 1.00,

        "precioUnitario": 249.00,

        "subTotal": 249.00,

        "producto": {

            "id": 31,

            "idCategoria": 2,

            "idMarca": 1,

            "nombre": "POLERA BELLISTA",

            "precio": 179.00,

            "categoria": {

                "id": 2,

                "nombre": "ROPA"

            },

            "marca": {

                "id": 2,

                "nombre": "NIKE"

            }

        }

    },

    {

        "id": 1003,

        "idPedido": 385,

        "idProducto": 23,

        "cantidad": 1.00,

        "precioUnitario": 69.00,

        "subTotal": 69.00,

        "producto": {

            "id": 23,

            "idCategoria": 2,

            "idMarca": 1,

            "nombre": "GORRA BASEBALL",

            "precio": 179.00,

            "categoria": {

                "id": 2,

                "nombre": "ACCESORIOS"

            },

            "marca": {

                "id": 2,

                "nombre": "ADIDAS"

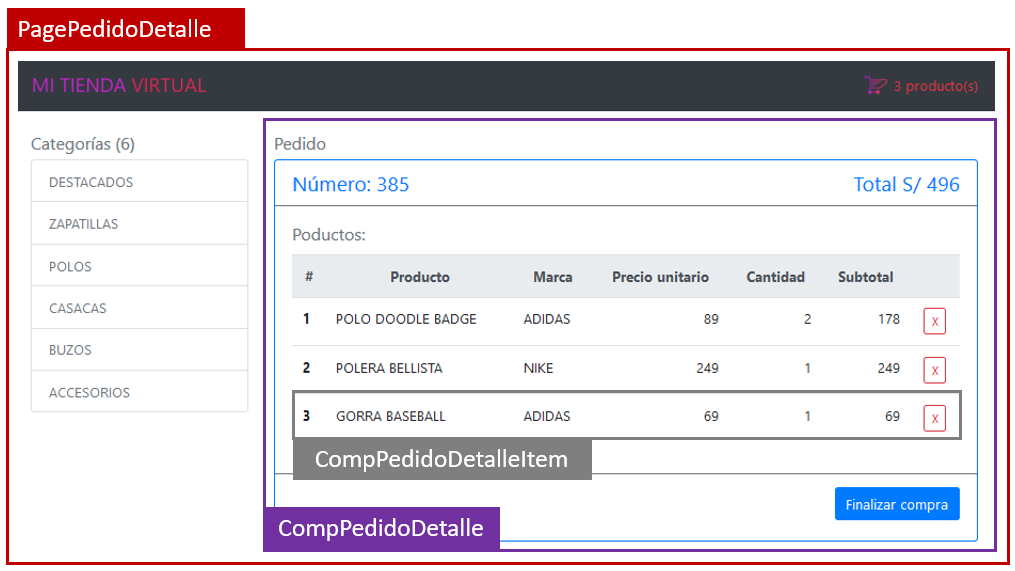
            }

        }

    }

]

Paso 3. Crear los componentes necesarios para implementar la página donde se muestra el detalle del pedido (carrito de compras). Agregar los datos (del pedido y su detalle) en el *state* del componente *PagePedidoDetalle* y pasarlo a los componentes *CompPedidoDetalle* y *CompPedidoDetalleItem*.



1. Dentro de la carpeta *Componentes* creamos el archivo *PagePedidoDetalle.js*, modificamos su código de acuerdo a:

import React, { Component } from 'react';

import CompCabecera from './CompCabecera';

import CompListaCategorias from './CompListaCategorias';

import CompPedidoDetalle from './CompPedidoDetalle';

import CompPiePagina from './CompPiePagina';

import dataJsonPedido from '../DataJson/dataJsonPedido.json';

import dataJsonListaDetallePedido from '../DataJson/dataJsonListaDetallePedido.json';

class PagePedidoDetalle extends Component {

    state = {

        miPedido: dataJsonPedido,

        miPedidoDetalle: dataJsonListaDetallePedido

    }

    render() {

        return <div >

            <CompCabecera />

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

                <div className="row">

                    <CompListaCategorias />

                    <CompPedidoDetalle pPedido={this.state.miPedido} pPedidoDetalle={this.state.miPedidoDetalle} />

                </div>

            </main>

            <CompPiePagina />

        </div>

    }

}

export default PagePedidoDetalle;

1. Dentro de la carpeta *Componentes* creamos el archivo *CompPedidoDetalle.js*, modificamos su código de acuerdo a:

import React, { Component } from 'react';

import CompPedidoDetalleItem from './CompPedidoDetalleItem';

class CompPedidoDetalle extends Component {

    render() {

        return <div className="col-9">

            <p className="h5 text-secondary">Pedido</p>

            <div className="card border-primary">

                <div className="card-header bg-transparent border-secondary h4 text-primary">

                    <span style={{ float: "left" }}>Número: {this.props.pPedido.id}</span>

                    <span style={{ float: "right" }}>Total S/ {this.props.pPedido.total}</span>

                </div>

                <div className="card-body text-secondary">

                    <h5 className="card-title">Poductos:</h5>

                    <table className="table">

                        <thead className="thead-light">

                            <tr>

                                <th scope="col" className="text-center">#</th>

                                <th scope="col" className="text-center">Producto</th>

                                <th scope="col" className="text-center">Marca</th>

                                <th scope="col" className="text-center">Precio unitario</th>

                                <th scope="col" className="text-center">Cantidad</th>

                                <th scope="col" className="text-center">SubtotalLL</th>

                                <th scope="col" className="text-center"></th>

                            </tr>

                        </thead>

                        {

                            this.props.pPedidoDetalle.map((item, indice) => {

                                return <CompPedidoDetalleItem pItem={item} pIndice={indice} key={indice}/>

                            })

                        }

                    </table>

                </div>

                <div className="card-footer bg-transparent border-secondary h4 text-secondary text-right">

                    <button type="submit" className="btn btn-primary my-1">Finalizar compra</button>

                </div>

            </div>

        </div>

    }

}

export default CompPedidoDetalle;

1. Dentro de la carpeta *Componentes* creamos el archivo *CompPedidoDetalleItem.js*, modificamos su código de acuerdo a:

import React, { Component } from 'react';

class CompPedidoDetalleItem extends Component {

    render() {

        return <tbody>

            <tr>

                <th scope="row">{this.props.pIndice + 1}</th>

                <td>{this.props.pItem.producto.nombre}</td>

                <td>{this.props.pItem.producto.marca.nombre}</td>

                <td className="text-right">{this.props.pItem.precioUnitario}</td>

                <td className="text-right">{this.props.pItem.cantidad}</td>

                <td className="text-right">{this.props.pItem.subTotal}</td>

                <td className="text-center">

                    <button type="button" className="btn btn-outline-danger btn-sm">X</button>

                </td>

            </tr>

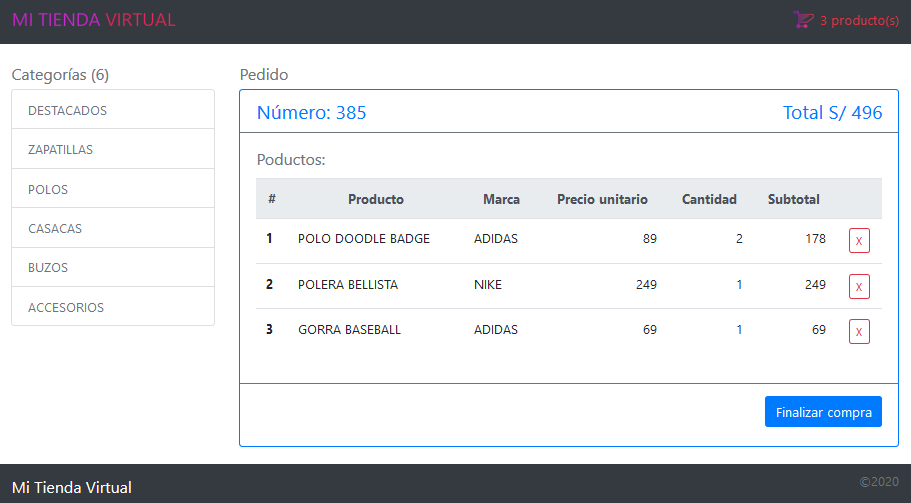
        </tbody>

    }

}

export default CompPedidoDetalleItem;

1. Configuraos el archivo *App.js* para que pueda visualizarse la pagina *PagePedidoDetalle.*
2. Ejecutamos la aplicación.



Paso 4. Agregar la funcionalidad de eliminar un producto del pedido actualizando el *state* del componente *PagePedidoDetalle*.

1. Vamos agregar en el archivo *PagePedidoDetalle.js* la función que se encargará de eliminar un producto del pedido. Esta función le pasaremos como una propieda al componente *CompPedidoDetalle.js* y ésta luego lo pasará al componente *CompPedidoDetalleItem.js* para que lo ejecute.
2. Modificamos el código del archivo *PagePedidoDetalle.js* de acuerdo a:

import React, { Component } from 'react';

import CompCabecera from './CompCabecera';

import CompListaCategorias from './CompListaCategorias';

import CompPedidoDetalle from './CompPedidoDetalle';

import CompPiePagina from './CompPiePagina';

import dataJsonPedido from '../DataJson/dataJsonPedido.json';

import dataJsonListaDetallePedido from '../DataJson/dataJsonListaDetallePedido.json';

class PagePedidoDetalle extends Component {

    state = {

        miPedido: dataJsonPedido,

        miPedidoDetalle: dataJsonListaDetallePedido

    }

    eliminarProductoDelPedido = (id) => {

        let nuevoPedidoDetalle = this.state.miPedidoDetalle.filter(itemDetalle => itemDetalle.id !== id);

        this.setState({

            miPedidoDetalle: nuevoPedidoDetalle

        });

        //recalcular el total del pedido

        let total = 0;

        let pedido = this.state.miPedido;

        nuevoPedidoDetalle.map(itemDetalle => total = total + itemDetalle.subTotal);

        pedido.total = total;

        this.setState({

            miPedido: pedido

        })

    }

    render() {

        return <div >

            <CompCabecera />

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

                <div className="row">

                    <CompListaCategorias />

                    <CompPedidoDetalle pPedido={this.state.miPedido}

                        pPedidoDetalle={this.state.miPedidoDetalle}

                        pFuncionEliminarProducto={this.eliminarProductoDelPedido} />

                </div>

            </main>

            <CompPiePagina />

        </div>

    }

}

export default PagePedidoDetalle;

1. Modificamos el código del archivo *CompPedidoDetalle.js* de acuerdo a:

import React, { Component } from "react";

import CompPedidoDetalleItem from "./CompPedidoDetalleItem";

class CompPedidoDetalle extends Component {

  render() {

    return (

      <div className="col-9">

        <p className="h5 text-secondary">Pedido</p>

        <div className="card border-primary">

          <div className="card-header bg-transparent border-secondary h4 text-primary">

            <span style={{ float: "left" }}>

              Número: {this.props.pPedido.id}

            </span>

            <span style={{ float: "right" }}>

              Total S/ {this.props.pPedido.total}

            </span>

          </div>

          <div className="card-body text-secondary">

            <h5 className="card-title">Poductos:</h5>

            <table className="table">

              <thead className="thead-light">

                <tr>

                  <th scope="col" className="text-center">

                    #

                  </th>

                  <th scope="col" className="text-center">

                    Producto

                  </th>

                  <th scope="col" className="text-center">

                    Marca

                  </th>

                  <th scope="col" className="text-center">

                    Precio unitario

                  </th>

                  <th scope="col" className="text-center">

                    Cantidad

                  </th>

                  <th scope="col" className="text-center">

                    SubtotalLL

                  </th>

                  <th scope="col" className="text-center"></th>

                </tr>

              </thead>

              {this.props.pPedidoDetalle.map((item, indice) => {

                return (

                  <CompPedidoDetalleItem

                    pItem={item}

                    pIndice={indice}

                    key={indice}

                    pFuncEliminarProducto={this.props.pFuncionEliminarProducto}

                  />

                );

              })}

            </table>

          </div>

          <div className="card-footer bg-transparent border-secondary h4 text-secondary text-right">

            <button type="submit" className="btn btn-primary my-1">

              Finalizar compra

            </button>

          </div>

        </div>

      </div>

    );

  }

}

export default CompPedidoDetalle;

1. Modificamos el código del archivo *CompPedidoDetalleItem.js* de acuerdo a:

import React, { Component } from 'react';

class CompPedidoDetalleItem extends Component {

    render() {

        return <tbody>

            <tr>

                <th scope="row">{this.props.pIndice + 1}</th>

                <td>{this.props.pItem.producto.nombre}</td>

                <td>{this.props.pItem.producto.marca.nombre}</td>

                <td className="text-right">{this.props.pItem.precioUnitario}</td>

                <td className="text-right">{this.props.pItem.cantidad}</td>

                <td className="text-right">{this.props.pItem.subTotal}</td>

                <td className="text-center">

                    <button type="button"

                    className="btn btn-outline-danger btn-sm"

                    onClick={this.props.pFuncEliminarProducto.bind(this, this.props.pItem.id)}>X</button>

                </td>

            </tr>

        </tbody>

    }

}

export default CompPedidoDetalleItem;

1. Para actualizar la cantidad de productos del pedido en la Cabecera actualizamos el código del componente *CompCabecera.js* de acuerdo a:

import React, {Component} from 'react';

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">

                        <a className="navbar-brand" href="#">

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

                        </a>

                    </span>

                    <span className="float-right">

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

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

                            <a className="text-danger" href="#">&nbsp;{this.props.pCantidadProductos} producto(s)</a>

                        </div>

                    </span>

                </div>

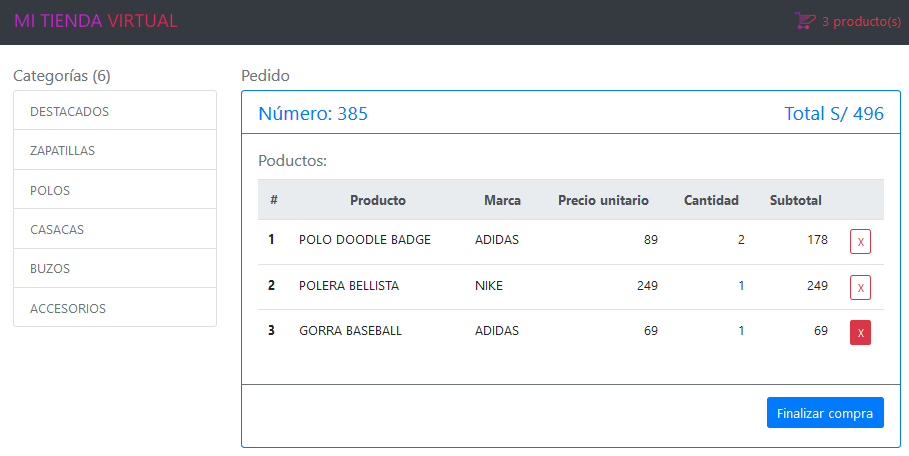
            </nav>

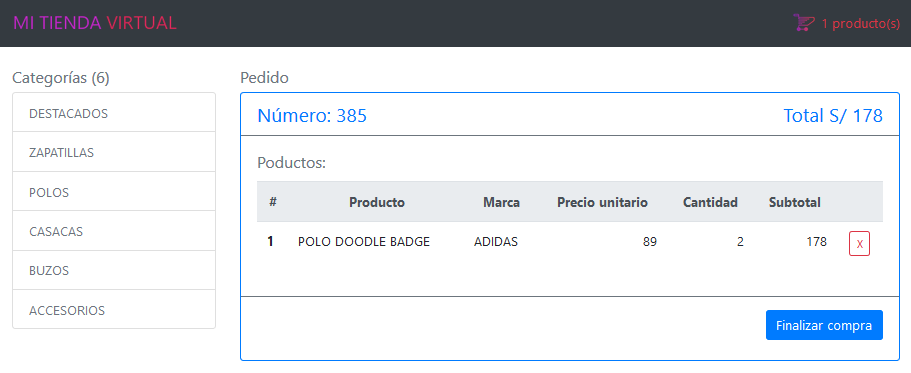
  }

}

export default CompCabecera;

1. Observamos los resultados en el explorador web, eliminamos los productos del pedido y observamos como se actualiza el total del mismo.





ACTIVIDAD VIRTUAL:

Ingresa a la plataforma virtual. Luego desarrolla la siguiente actividad propuesta:

1. CUESTIONARIO TÉCNICO
   * 1. En el archivo *dataJsonListaDetallePedido.json* agregar 5 productos (detalle del pedido), además en el archivo *dataJsonPedido.json* actualizar el total de acuerdo a los productos agregados en el detalle del pedido.
     2. Agregar la columna *Categoría* después de la columna *Marca* en el detalle del pedido.
     3. Agregar en la cabecera de la página el monto total del pedido, este monto debe actualizarse cada vez que se elimina un producto del pedido.



* + 1. Agregar el botón *Limpiar pedido* ubicado al costado izquierdo del botón *Finalizar compra*, al presionar el botón debe eliminar todos los productos del pedido, además de actualizar los totales.
    2. Investigar la forma de validar datos de un formulario con Javascript.

1. CONCLUSIONES DE LA EXPERIENCIA

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

**ACTIVIDAD** **VIRTUAL**:

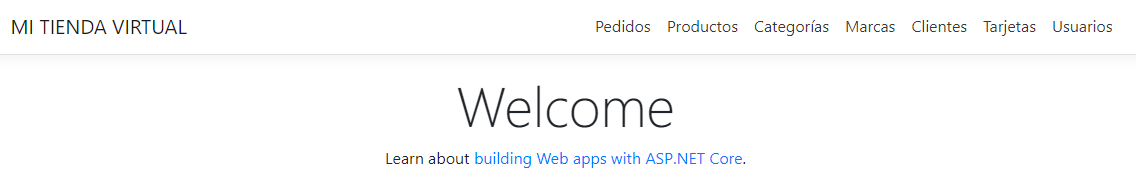
Ingresa a la plataforma virtual. Revisa y analiza el tema desarrollado en esta sesión, luego desarrolla las siguientes actividades propuestas:

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

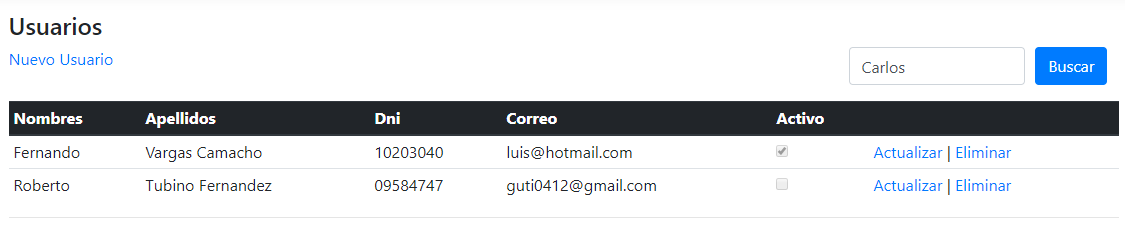
* Marca
* Categoría
* Producto
  + 1. Crear con Scaffold Controladores y vistas para el mantenimiento (CRUD) de las siguientes entidades (Tablas):
* Cliente
* Tarjeta

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:

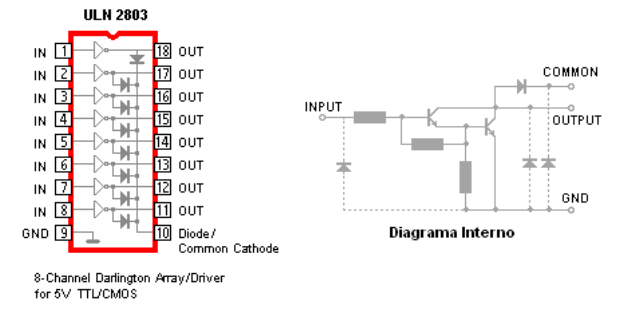


**Subtema 1.2:**

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.

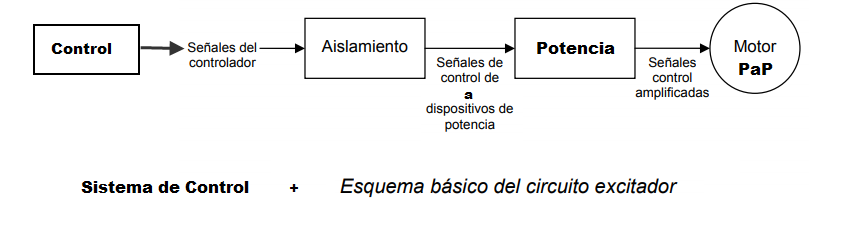
Por lo tanto si deseamos mayores corrientes debemos utilizar otros drivers o implementar uno concomponentes discretos , es decir con transistores de potencia.



**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.