Herramientas de Programación 2

Tema Nº12:FORMULARIOS WEB (PARTE 2)

Indicador de logro Nº12:Diseña formularios web para la validación y registro de información en la BD siguiendo las instrucciones en REACT.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

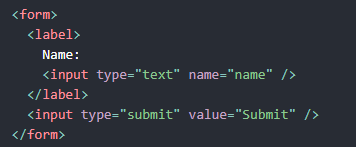
**TEMA Nº12:**

FORMULARIOS WEB (PARTE 2)

**Subtema 1.1:**

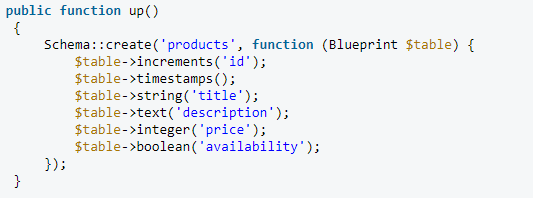
React Forms

Los elementos de formularios en HTML funcionan un poco diferente a otros elementos del DOM en React, debido a que los elementos de formularios conservan naturalmente algún estado interno. Por ejemplo, este formulario solamente en HTML, acepta un solo nombre.

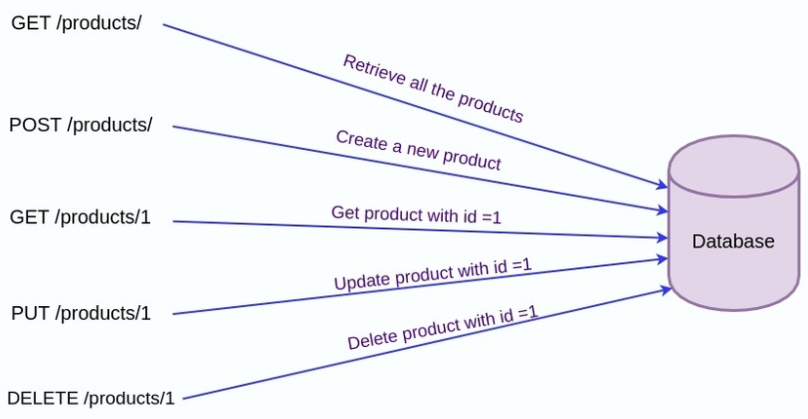


**Passing Functions**

Puede definir un Function passing solo a nivel del componente.



**Delete and Update**



React con servicios RESTful.

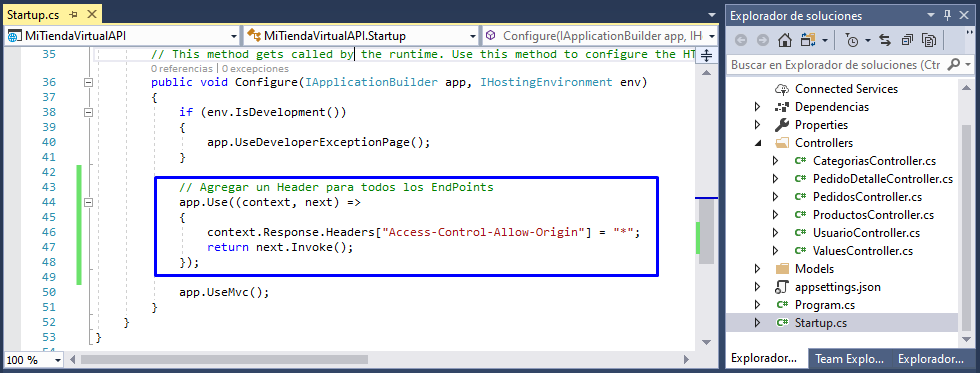
En los anteriores artículos vimos como empezar con React, mostrando una serie de datos en una aplicación web (categorías, productos y pedidos). Estos objetos estaban en archivos dentro del código de la aplicación web. Lo real es que estos datos van a llegar en formato JSON desde un servicio REST, ya sea nuestro o de un tercero.

En los ejemplo de hoy vamos a llamar a la API RESTful de nuestro proyecto del Back End que nos devuelve la información de las categorías y productos en formato JSON, y en nuestra aplicación React vamos a consumirlos usando nuevas funciones presentes en ECMAScript6



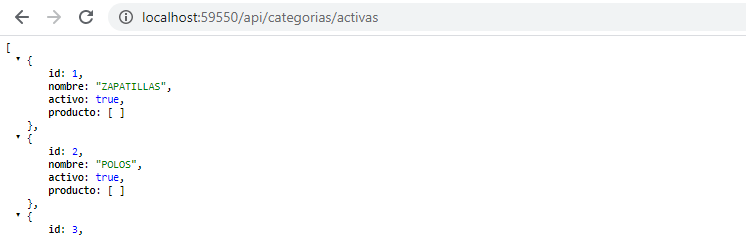
Paso 1. Preparar los servicios API del Back End.

1. Para que nuestro servicio RESTful del Back End no tenga inconvenientes al momento de ser consumido por clientes externos debemos hacer la siguiente configuración.
2. Abrimos el proyecto *MiTiendaVirtualAPI* en Visual Studio.
3. Abrimos el archivo *Startup.cs*.
4. Dentro del método *Configure()* agregamos el llamado a la función *app.Use()* de acuerdo a:



1. Ejecutamos la aplicación y resolvemos en el navegador el siguiente End Point:

<http://localhost:59550/api/categorias/activas>



Paso 2. Consumir servicios GET: Lista de Categorías.

1. Abrimos en Visual Studio Code el proyecto React del taller de la semana anterior.
2. Ejecutamos la aplicación para verificar que no existan errores.
3. Vamos a obtener la lista de categorías activas desde el Back End, para ello vamos a consumir el End Point */api/categorias/activas*.
4. Modificamos el código del componente *CompListaCategorias.js* de acuerdo a:

import React, { Component } from 'react';

class CompListaCategorias extends Component {

  state = {

    dataJsonListaCategorias: []

  }

  async componentDidMount() {

    const respuestaGet = await fetch('http://localhost:59550/api/categorias/activas');

    const respuestaJson = await respuestaGet.json();

    this.setState({

      dataJsonListaCategorias: respuestaJson

    })

  }

  render() {

    return (

      <div className="col-3">

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

        <div className="card">

          <ul className="list-group list-group-flush">

            <li className="list-group-item"><a href="productos/destacados" className="text-secondary">DESTACADOS</a></li>

            {

              this.state.dataJsonListaCategorias.map((categoria, indice) => {

                var url = "productos/porcategoria/" + categoria.id;

                return <li className="list-group-item" key={categoria.id}><a href={url} className="text-secondary">{categoria.nombre}</a></li>

              })

            }

          </ul>

        </div>

      </div>

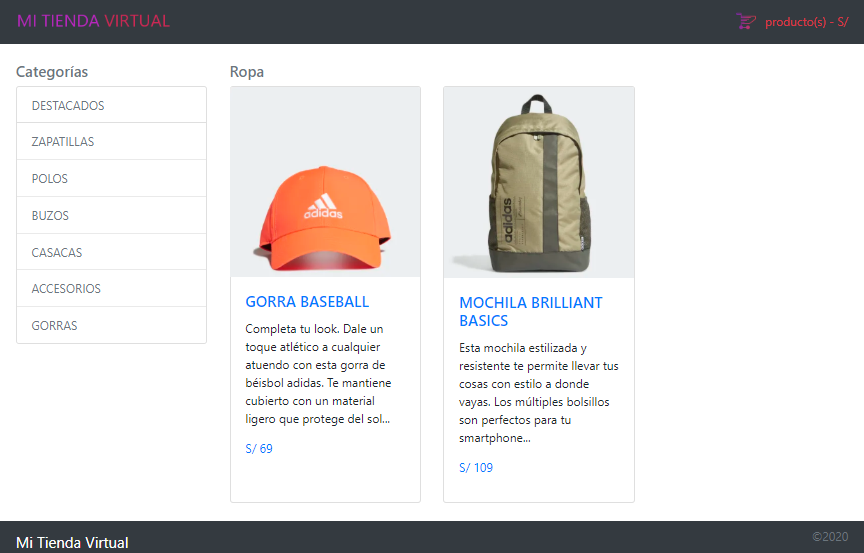
    )

  }

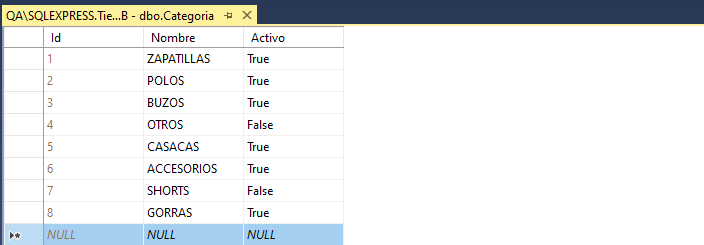
}

export default CompListaCategorias;

1. Configuramos la aplicación para ver inicialmente el componente *PageListaProductos*.
2. Ejecutamos la aplicación React para revisar los cambios.



1. Verificamos que la lista de categorías que se muestran son os que están activos en la BD.



Paso 3. Consumir servicios GET: Lista de Productos.

1. Vamos a obtener la lista de productos destacados desde el Back End, para ello vamos a consumir el End Point */api/productos/destacados*.
2. Modificamos el código del componente *PageListaProductos.js* de acuerdo a:

import React, {Component} from 'react';

import CompCabecera from './CompCabecera';

import CompListaCategorias from './CompListaCategorias';

import CompPiePagina from './CompPiePagina';

import CompListaProductos from './CompListaProductos';

class PageListaProductos extends Component {

  state = {

    dataJsonListaProductos: []

  }

  async componentDidMount() {

    const respuestaGet = await fetch('http://localhost:59550/api/productos/destacados');

    const respuestaJson = await respuestaGet.json();

    this.setState({

      dataJsonListaProductos: respuestaJson

    })

  }

  render(){

    return  <div >

                <CompCabecera />

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

                    <div className="row">

                        <CompListaCategorias />

                        <CompListaProductos pListaDeProductos={this.state.dataJsonListaProductos}/>

                    </div>

                </main>

                <CompPiePagina />

            </div>

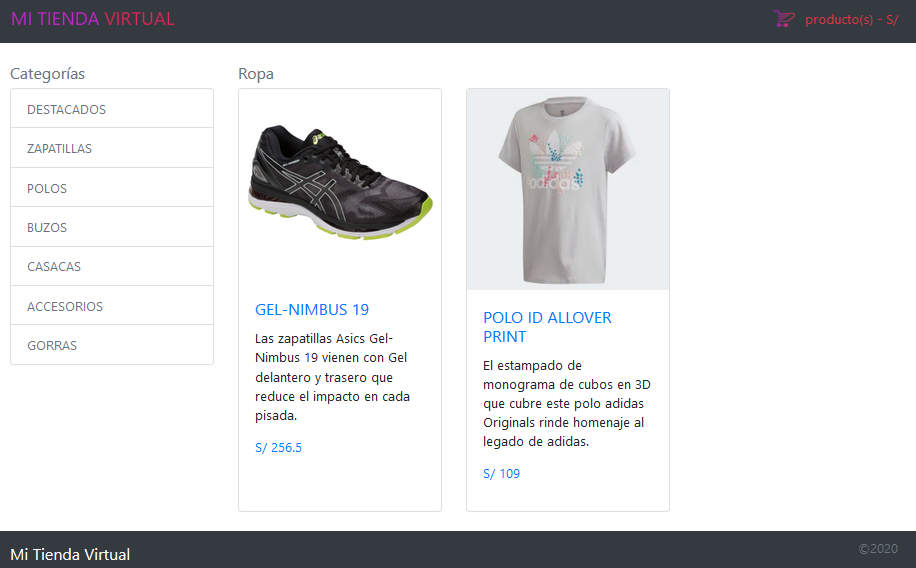
  }

}

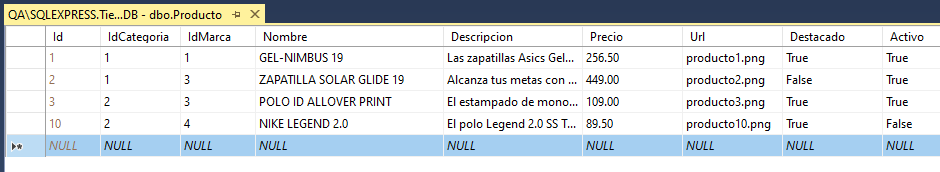
export default PageListaProductos;

1. Observamos en el explorador web los cambios.

Recordar que dentro de la carpeta *imagenes* se deben agregar las imágenes de los productos y los nombres de los archivos deben figurar en la tabla *Producto* de la BD.



1. Verificamos que la lista de productos destacados que se muestran son los que están en la BD como destacados y activos.



Paso 3. Instalar react-router-dom.

1. Antes de usar ruteadores en React vamos a instalar la librería correspondiente.
2. Dejamos de ejecutar la aplicación, para ello en la consola de VS Code presionamos las teclas *Ctrl+c*.



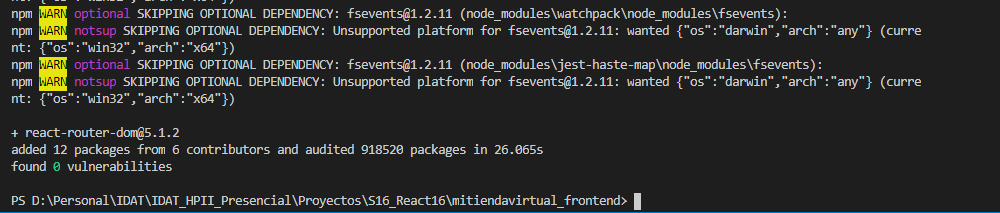
1. Presionamos la tecla “S” y luego la tecla *Enter*.



1. Ahora ejecutamos el comando: *npm install react-router-dom*



1. Finalizado la descarga y la instalación obtenemos:



1. Volvemos a ejecutar la aplicación React para ver que no existan errores.

Paso 4. Mapear las rutas de las páginas.

1. El mapeo de las rutas las vamos hacer en el componente App.js.
2. Abrimos el archivo App.js y modificamos su código de acuerdo a:

import React, { Component } from 'react';

import { BrowserRouter, Route } from 'react-router-dom';

import './App.css';

import PageListaProductos from './Componentes/PageListaProductos';

import PageProducto from './Componentes/PageProducto';

import PagePedidoDetalle from './Componentes/PagePedidoDetalle';

import PagePedidoCliente from './Componentes/PagePedidoCliente';

class App extends Component {

  render() {

    return <div>

      <BrowserRouter>

        <React.Fragment>

          <Route exact path="/" component={PageListaProductos} />

          <Route exact path="/detalleProducto" component={PageProducto} />

          <Route exact path="/miPedido" component={PagePedidoDetalle} />

          <Route exact path="/miPedidoCliente" component={PagePedidoCliente} />

        </React.Fragment>

      </BrowserRouter>

    </div>

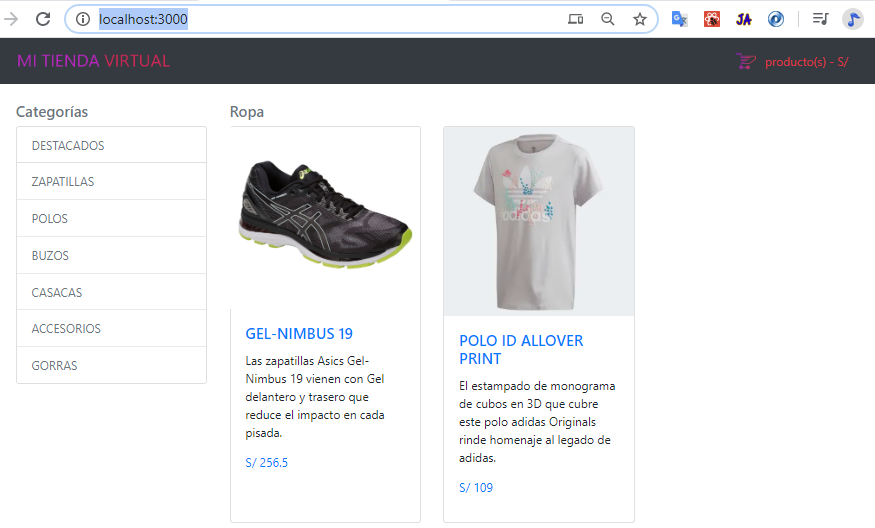
  }

}

export default App;

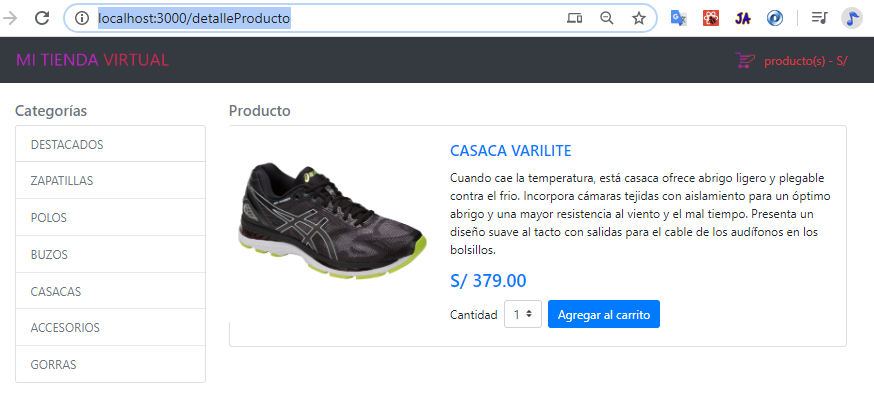
1. En el explorador web resolvemos la siguiente Url:

<http://localhost:3000/>



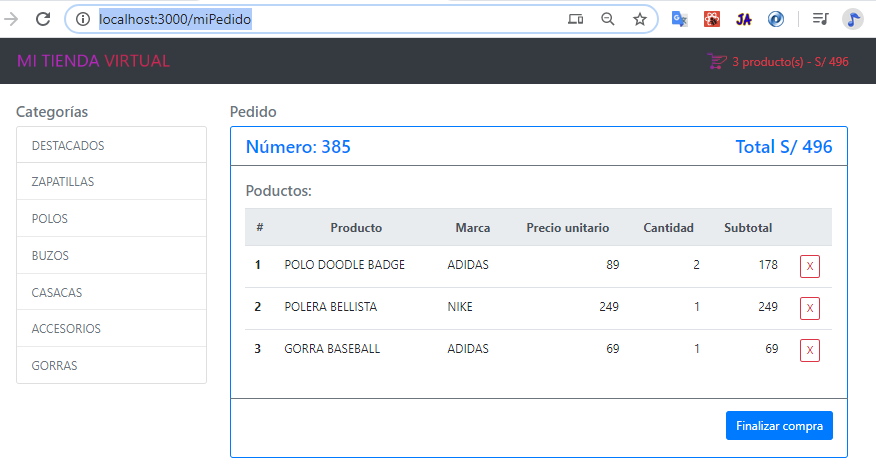
1. En el explorador web resolvemos la siguiente Url:

<http://localhost:3000/detalleProducto>



1. En el explorador web resolvemos la siguiente Url:

<http://localhost:3000/miPedido>



1. En el explorador web resolvemos la siguiente Url:

<http://localhost:3000/miPedidoCliente>



Paso 5. Agregar enlaces (Links).

1. Vamos a configurar los dos links que están en la cabecera:

* Logo
* Pedido

1. Abrimos el archivo *CompCabecera.js* y actualizamos su código 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-danger" to="/miPedido">

                                &nbsp;{this.props.pCantidadProductos} producto(s) - S/ {this.props.pTotal}

                            </Link>

                        </div>

                    </span>

                </div>

            </nav>

  }

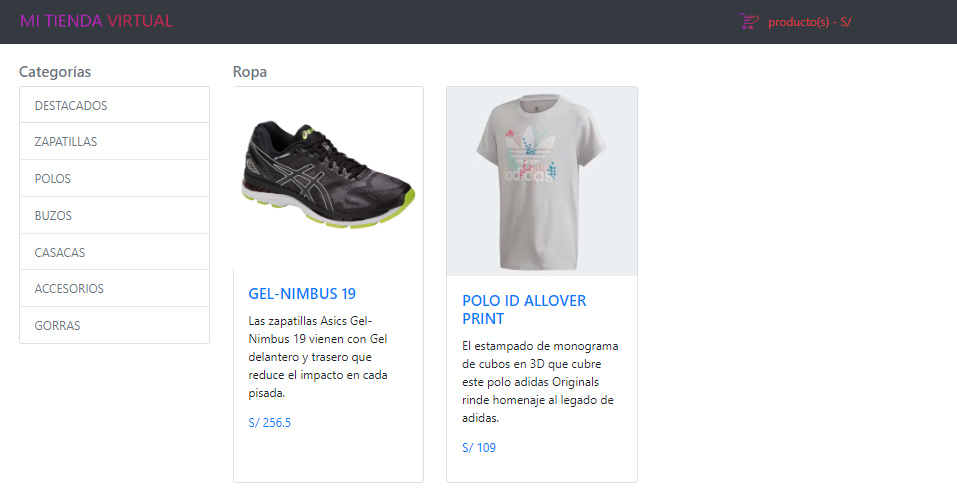
}

export default CompCabecera;

1. En el explorador web hacemos click en el enlace superior derecho de la cabecera, obtenemos:



1. Hacemos click en el logo (que es un enlace).



ACTIVIDAD VIRTUAL:

1. CUESTIONARIO TÉCNICO
   * 1. En el componente *CompProductoResumen.js* configurar los enlaces que se encuentran en la imagen del producto y su nombre de tal forma que direccionen al componente *PageProducto.js*.
     2. En la parte central del pie de página agregar cuatro enlaces (links) cada uno de ellos debe direccionar a alguno de los siguientes componentes.

* *PageListaProductos.js*
* *PageProducto.js*
* *PagePedidoDetalle.js*
* *PagePedidoCliente.js*
  + 1. En el componente *CompListaCategorias.js* configurar los enlaces de la lista de categorías de tal forma que direccionen al componente *PageListaProductos.js*.
    2. En el componente *CompPedidoDetalle.js* configurar el botón *Finalizar compra* de tal forma que direccione al componente *PagePedidoCliente.js*.
    3. Investigar la forma de consumir servicios web RESTful de tipo POST desde aplicaciones React.

1. CONCLUSIONES DE LA EXPERIENCIA

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

**ACTIVIDAD** **VIRTUAL**:

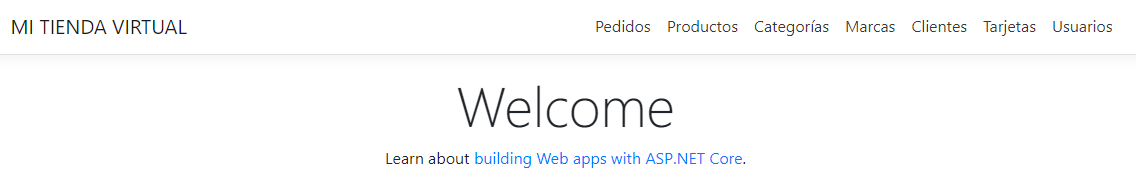
Ingresa a la plataforma virtual. Revisa y analiza el tema desarrollado en esta sesión, luego desarrolla las siguientes actividades propuestas:

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

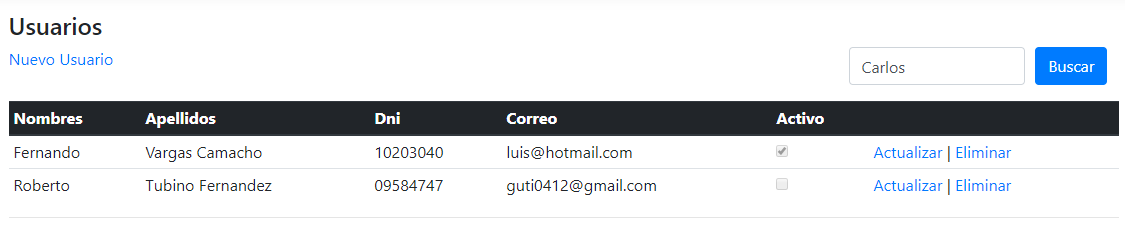
* Marca
* Categoría
* Producto
  + 1. Crear con Scaffold Controladores y vistas para el mantenimiento (CRUD) de las siguientes entidades (Tablas):
* Cliente
* Tarjeta

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:

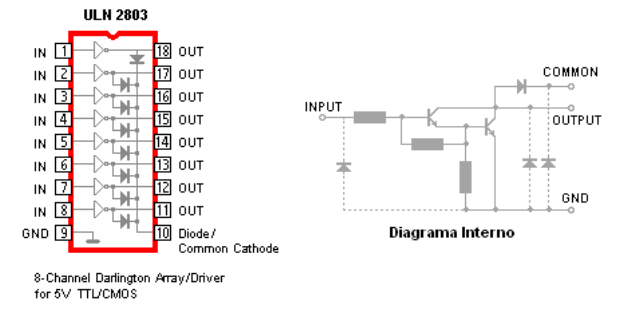


**Subtema 1.2:**

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.

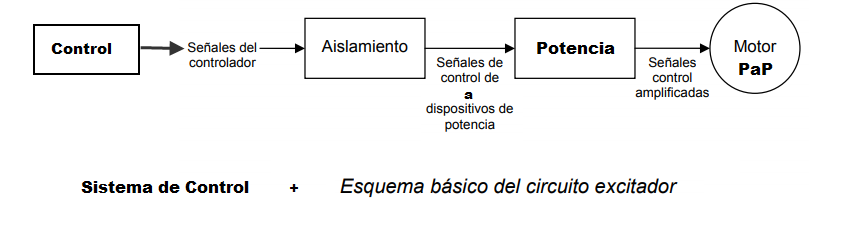
Por lo tanto si deseamos mayores corrientes debemos utilizar otros drivers o implementar uno concomponentes discretos , es decir con transistores de potencia.



**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.