# VERSION 1

# Server.js 1

const express = require('express');

const cors = require('cors'); // Importar cors

const app = express();

app.use(cors()); // Habilitar CORS

app.use(express.json());

app.get('/', (req, res) => {

  res.send('Probando al Servidor - Prueba exitosa! :}');

});

const config = require('./config/config');

const server = app.listen(config.server.port, () => {

  console.log(`Servidor escuchando en el puerto ${config.server.port}`);

});

const sequelize = config.db;

sequelize.authenticate()

  .then(() => {

    console.log('Conectado a MariaDB usando Sequelize');

  })

  .catch(err => {

    console.error('No se pudo conectar a la base de datos:', err);

  });

const postRoute = require('./routes/post');

app.use('/servicios', postRoute);

function handleShutdown() {

  console.log('Cerrando servidor...');

  server.close(() => {

    console.log('Servidor cerrado.');

    process.exit(0);

  });

}

process.on('SIGINT', handleShutdown);

process.on('SIGTERM', handleShutdown);

# post.js 1

const sequelize = require('../config/config').db; // 1.1 Importa la conexión desde config

const express = require('express'); // 2.1 Importa Express

const router = express.Router(); // 2.2 Crea una instancia de Express Router

const mysql = require('mysql2/promise'); // 3.1 Importa el paquete mysql2/promise

const pool = mysql.createPool({ // 3.2 Configura el pool de conexiones a la base de datos

  host: process.env.DB\_HOST,

  user: process.env.DB\_USER,

  password: process.env.DB\_PASSWORD,

  database: process.env.DB\_NAME

});

// 4. Función para crear una entrada en cualquier tabla

async function createEntry(table, columns, values) {

  const columnsStr = columns.join(', '); // 4.1 Une los nombres de las columnas en una cadena

  const placeholders = columns.map(() => '?').join(', '); // 4.2 Crea una cadena de placeholders para valores

  const sql = `INSERT INTO ${table} (${columnsStr}) VALUES (${placeholders})`; // 4.3 Construye la consulta SQL de inserción

  const [result] = await pool.execute(sql, values); // 4.4 Ejecuta la consulta SQL

  return result; // 4.5 Devuelve el resultado de la inserción

}

// 5. Función para obtener todas las entradas de cualquier tabla

async function getEntries(table) {

  const sql = `SELECT \* FROM ${table}`; // 5.1 Construye la consulta SQL de selección

  const [rows] = await pool.execute(sql); // 5.2 Ejecuta la consulta SQL

  return rows; // 5.3 Devuelve todas las filas obtenidas

}

// 6. Función para obtener una entrada por ID de cualquier tabla

async function getEntryById(table, idColumn, id) {

  const sql = `SELECT \* FROM ${table} WHERE ${idColumn} = ?`; // 6.1 Construye la consulta SQL de selección por ID

  const [rows] = await pool.execute(sql, [id]); // 6.2 Ejecuta la consulta SQL con el ID especificado

  return rows[0]; // 6.3 Devuelve la primera fila obtenida (la entrada)

}

// 7. Función para actualizar una entrada por ID en cualquier tabla

async function updateEntryById(table, idColumn, id, columns, values) {

  const updates = columns.map(column => `${column} = ?`).join(', '); // 7.1 Une los nombres de las columnas con placeholders

  const sql = `UPDATE ${table} SET ${updates} WHERE ${idColumn} = ?`; // 7.2 Construye la consulta SQL de actualización

  const [result] = await pool.execute(sql, [...values, id]); // 7.3 Ejecuta la consulta SQL con los valores y el ID

  return result; // 7.4 Devuelve el resultado de la actualización

}

// 8. Función para eliminar una entrada por ID de cualquier tabla

async function deleteEntryById(table, idColumn, id) {

  const sql = `DELETE FROM ${table} WHERE ${idColumn} = ?`; // 8.1 Construye la consulta SQL de eliminación

  const [result] = await pool.execute(sql, [id]); // 8.2 Ejecuta la consulta SQL con el ID especificado

  return result; // 8.3 Devuelve el resultado de la eliminación

}

// 9. Definir las rutas utilizando Express Router

router.post('/create', async (req, res) => {

  try {

    const { table, columns, values } = req.body; // 9.1 Obtiene los datos de la solicitud

    const result = await createEntry(table, columns, values); // 9.2 Llama a la función de crear entrada

    res.status(201).json(result); // 9.3 Responde con el resultado

  } catch (error) {

    res.status(500).send('Error al crear la entrada'); // 9.4 Maneja los errores

  }

});

router.get('/entries/:table', async (req, res) => {

  try {

    const { table } = req.params; // 9.5 Obtiene el nombre de la tabla de los parámetros

    const entries = await getEntries(table); // 9.6 Llama a la función de obtener entradas

    res.json(entries); // 9.7 Responde con las entradas obtenidas

  } catch (error) {

    res.status(500).send('Error al obtener las entradas'); // 9.8 Maneja los errores

  }

});

router.get('/entry/:table/:idColumn/:id', async (req, res) => {

  try {

    const { table, idColumn, id } = req.params; // 9.9 Obtiene los datos de los parámetros

    const entry = await getEntryById(table, idColumn, id); // 9.10 Llama a la función de obtener entrada por ID

    if (entry) {

      res.json(entry); // 9.11 Responde con la entrada obtenida

    } else {

      res.status(404).send('Entrada no encontrada'); // 9.12 Maneja la entrada no encontrada

    }

  } catch (error) {

    res.status(500).send('Error al obtener la entrada'); // 9.13 Maneja los errores

  }

});

router.put('/update/:table/:idColumn/:id', async (req, res) => {

  try {

    const { table, idColumn, id } = req.params; // 9.14 Obtiene los datos de los parámetros

    const { columns, values } = req.body; // 9.15 Obtiene los datos del cuerpo de la solicitud

    const result = await updateEntryById(table, idColumn, id, columns, values); // 9.16 Llama a la función de actualizar entrada

    res.json(result); // 9.17 Responde con el resultado

  } catch (error) {

    res.status(500).send('Error al actualizar la entrada'); // 9.18 Maneja los errores

  }

});

router.delete('/delete/:table/:idColumn/:id', async (req, res) => {

  try {

    const { table, idColumn, id } = req.params; // 9.19 Obtiene los datos de los parámetros

    const result = await deleteEntryById(table, idColumn, id); // 9.20 Llama a la función de eliminar entrada

    res.json(result); // 9.21 Responde con el resultado

  } catch (error) {

    res.status(500).send('Error al eliminar la entrada'); // 9.22 Maneja los errores

  }

});

module.exports = router; // 10. Exporta el router para ser utilizado en app.js

# app.js 1

const express = require('express');

const cors = require('cors'); // Importar cors

const app = express();

app.use(cors()); // Habilitar CORS

app.use(express.json());

app.get('/', (req, res) => {

  res.send('Probando al Servidor - Prueba exitosa! :}');

});

const config = require('./config/config');

const server = app.listen(config.server.port, () => {

  console.log(`Servidor escuchando en el puerto ${config.server.port}`);

});

const sequelize = config.db;

sequelize.authenticate()

  .then(() => {

    console.log('Conectado a MariaDB usando Sequelize');

  })

  .catch(err => {

    console.error('No se pudo conectar a la base de datos:', err);

  });

const postRoute = require('./routes/post');

app.use('/servicios', postRoute);

function handleShutdown() {

  console.log('Cerrando servidor...');

  server.close(() => {

    console.log('Servidor cerrado.');

    process.exit(0);

  });

}

process.on('SIGINT', handleShutdown);

process.on('SIGTERM', handleShutdown);

# register.html 1

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta name="keywords" content="" />

    <meta name="description" content="" />

    <meta name="author" content="" />

    <link rel="shortcut icon" href="../images/favicon.png" type="image/x-icon">

    <title>Maria E. girls Fashion Confecciones.</title>

    <!-- slider stylesheet -->

    <link rel="stylesheet" type="text/css" href="https://cdnjs.cloudflare.com/ajax/libs/OwlCarousel2/2.3.4/assets/owl.carousel.min.css" />

    <!-- bootstrap core css -->

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

    <!-- Custom styles for this template -->

    <link href="../css/style.css" rel="stylesheet"/>

    <!-- responsive style -->

    <link href="../css/responsive.css" rel="stylesheet" />

    <style>

      body {

        background-color: #f8f9fa;

      }

      .container {

        max-width: 600px;

        margin-top: 50px;

        padding: 20px;

        background-color: #ffffff;

        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

        border-radius: 8px;

      }

      .form-control:focus {

        box-shadow: none;

      }

    </style>

</head>

<body>

  <!-- header section strats -->

  <header class="header\_section">

    <nav class="navbar navbar-expand-lg custom\_nav-container ">

      <a class="navbar-brand" href="index.html">

        <span>

          Maria E. girls Fashion Confecciones.<br>

          <h6>🌟 ! Donde el estilo conoce la comodidad ! 🌟</h6>

        </span>

      </a>

      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

        <span class=""></span>

      </button>

      <div class="collapse navbar-collapse" id="navbarSupportedContent">

        <ul class="navbar-nav  ">

          <li class="nav-item active">

            <a class="nav-link" href="index.html">Inicio <span class="sr-only">(current)</span></a>

          </li>

          <li class="nav-item">

            <a class="nav-link" href="shop.html">

              Tienda

            </a>

          </li>

          <li class="nav-item">

            <a class="nav-link" href="why.html">

              Nosotros

            </a>

          </li>

          <li class="nav-item">

            <a class="nav-link" href="testimonial.html">

              Testimonios

            </a>

          </li>

          <li class="nav-item">

            <a class="nav-link" href="contact.html">

              Contáctenos

            </a>

          </li>

        </ul>

        <div class="user\_option">

          <a href="">

            <i class="fa fa-user" aria-hidden="true"></i>

            <span>

              Ingresar

            </span>

          </a>

          <a href="">

            <i class="fa fa-shopping-bag" aria-hidden="true"></i>

          </a>

          <form class="form-inline ">

            <button class="btn nav\_search-btn" type="submit">

              <i class="fa fa-search" aria-hidden="true"></i>

            </button>

          </form>

        </div>

      </div>

    </nav>

  </header>

  <!-- end header section -->

  <div class="container">

    <h2 class="text-center">Registro de Usuario</h2>

    <!-- Para apuntar a la ruta del servidor -->

    <form action="http://localhost:10000/create" method="POST">

      <div class="form-group">

        <label for="id\_tipo\_documento">Tipo de Documento</label>

        <input type="text" class="form-control" id="id\_tipo\_documento" name="id\_tipo\_documento" placeholder="Ingresa el tipo de documento" required> <!-- Línea 1: Agregar campo tipo de documento -->

      </div>

      <div class="form-group">

        <label for="id\_cliente">Número de Documento</label>

        <input type="text" class="form-control" id="id\_cliente" name="id\_cliente" placeholder="Ingresa tu número de documento" required> <!-- Línea 2: Agregar campo número de documento -->

      </div>

      <div class="form-group">

        <label for="nombre\_cliente">Nombre Completo</label>

        <input type="text" class="form-control" id="nombre\_cliente" name="nombre\_cliente" placeholder="Ingresa tu nombre completo" required> <!-- Línea 3: Modificar campo nombre -->

      </div>

      <div class="form-group">

        <label for="direccion\_cliente">Dirección</label>

        <input type="text" class="form-control" id="direccion\_cliente" name="direccion\_cliente" placeholder="Ingresa tu dirección" required> <!-- Línea 4: Modificar campo dirección -->

      </div>

      <div class="form-group">

        <label for="celular\_cliente">Teléfono</label>

        <input type="tel" class="form-control" id="celular\_cliente" name="celular\_cliente" placeholder="Ingresa tu número de teléfono" required> <!-- Línea 5: Modificar campo teléfono -->

      </div>

      <div class="form-group">

        <label for="password\_cliente">Contraseña</label>

        <input type="password" class="form-control" id="password\_cliente" name="password\_cliente" placeholder="Ingresa tu contraseña" required> <!-- Línea 6: Modificar campo contraseña -->

      </div>

      <div class="form-group">

        <label for="confirmPassword">Confirmar Contraseña</label>

        <input type="password" class="form-control" id="confirmPassword" name="confirmPassword" placeholder="Confirma tu contraseña" required> <!-- Línea 7: Confirmar contraseña (para validación del lado del cliente) -->

      </div>

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

    </form>

  </div>

  <!-- jQuery -->

  <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>

  <!-- Popper.js (necesario para Bootstrap) -->

  <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"></script>

  <!-- Bootstrap JS -->

  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

  <!-- Owl Carousel -->

  <script src="https://cdnjs.cloudflare.com/ajax/libs/OwlCarousel2/2.3.4/owl.carousel.min.js"></script>

  <!-- Custom Scripts -->

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

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

</body>

</html>

# VERSION 2

# Server.js 2

const express = require('express'); // Importa el módulo express

const bodyParser = require('body-parser'); // Importa el módulo body-parser para analizar cuerpos de solicitudes

const cors = require('cors'); // Importa el módulo cors para habilitar CORS

const helmet = require('helmet'); // Importa el módulo helmet para mejorar la seguridad mediante encabezados HTTP

const sequelize = require('./db'); // Importa la configuración de la base de datos desde db.js

const app = express(); // Crea una instancia de una aplicación Express

const port = 10000; // Define el puerto en el cual el servidor escuchará

// Middleware

app.use(cors()); // Habilita CORS para permitir solicitudes entre dominios

app.use(helmet()); // Usa helmet para establecer varios encabezados de seguridad HTTP

app.use(bodyParser.urlencoded({ extended: true })); // Configura body-parser para analizar cuerpos de solicitudes URL codificadas

app.use(bodyParser.json()); // Configura body-parser para analizar cuerpos de solicitudes en formato JSON

// Establecer Content-Security-Policy

app.use((req, res, next) => {

  res.setHeader(

    "Content-Security-Policy",

    "default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; img-src 'self' data:; style-src 'self' 'unsafe-inline'; font-src 'self'; connect-src 'self'"

  ); // Configura la política de seguridad de contenidos (CSP) para permitir ciertos recursos

  next(); // Llama a la siguiente función de middleware en la cadena

});

// Ruta para crear cliente

app.post('/create', async (req, res) => {

  const { id\_tipo\_documento, id\_cliente, nombre\_cliente, direccion\_cliente, celular\_cliente, password\_cliente } = req.body; // Extrae datos del cuerpo de la solicitud

  try {

    // Ejecuta una consulta para insertar un nuevo cliente en la base de datos

    await sequelize.query(

      `INSERT INTO clientes (id\_tipo\_documento, id\_cliente, nombre\_cliente, direccion\_cliente, celular\_cliente, password\_cliente) VALUES (?, ?, ?, ?, ?, ?)`,

      {

        replacements: [id\_tipo\_documento, id\_cliente, nombre\_cliente, direccion\_cliente, celular\_cliente, password\_cliente], // Reemplaza los placeholders con los valores reales

        type: sequelize.QueryTypes.INSERT // Especifica que esta consulta es una inserción

      }

    );

    res.status(201).send('Registro exitoso'); // Envía una respuesta de éxito

  } catch (err) {

    console.error(err); // Registra cualquier error en la consola

    res.status(500).send('Error en el registro'); // Envía una respuesta de error

  }

});

// Iniciar el servidor

app.listen(port, () => {

  console.log(`Servidor ejecutándose en http://localhost:${port}`); // Mensaje de confirmación al iniciar el servidor

});

# post.js 2

const sequelize = require('../config/config').db; // 1.1 Importa la conexión desde config

const express = require('express'); // 2.1 Importa Express

const router = express.Router(); // 2.2 Crea una instancia de Express Router

const mysql = require('mysql2/promise'); // 3.1 Importa el paquete mysql2/promise

const pool = mysql.createPool({ // 3.2 Configura el pool de conexiones a la base de datos

  host: process.env.DB\_HOST,

  user: process.env.DB\_USER,

  password: process.env.DB\_PASSWORD,

  database: process.env.DB\_NAME

});

// 4. Función para crear una entrada en cualquier tabla

async function createEntry(table, columns, values) {

  const columnsStr = columns.join(', '); // 4.1 Une los nombres de las columnas en una cadena

  const placeholders = columns.map(() => '?').join(', '); // 4.2 Crea una cadena de placeholders para valores

  const sql = `INSERT INTO ${table} (${columnsStr}) VALUES (${placeholders})`; // 4.3 Construye la consulta SQL de inserción

  const [result] = await pool.execute(sql, values); // 4.4 Ejecuta la consulta SQL

  return result; // 4.5 Devuelve el resultado de la inserción

}

// 5. Función para obtener todas las entradas de cualquier tabla

async function getEntries(table) {

  const sql = `SELECT \* FROM ${table}`; // 5.1 Construye la consulta SQL de selección

  const [rows] = await pool.execute(sql); // 5.2 Ejecuta la consulta SQL

  return rows; // 5.3 Devuelve todas las filas obtenidas

}

// 6. Función para obtener una entrada por ID de cualquier tabla

async function getEntryById(table, idColumn, id) {

  const sql = `SELECT \* FROM ${table} WHERE ${idColumn} = ?`; // 6.1 Construye la consulta SQL de selección por ID

  const [rows] = await pool.execute(sql, [id]); // 6.2 Ejecuta la consulta SQL con el ID especificado

  return rows[0]; // 6.3 Devuelve la primera fila obtenida (la entrada)

}

// 7. Función para actualizar una entrada por ID en cualquier tabla

async function updateEntryById(table, idColumn, id, columns, values) {

  const updates = columns.map(column => `${column} = ?`).join(', '); // 7.1 Une los nombres de las columnas con placeholders

  const sql = `UPDATE ${table} SET ${updates} WHERE ${idColumn} = ?`; // 7.2 Construye la consulta SQL de actualización

  const [result] = await pool.execute(sql, [...values, id]); // 7.3 Ejecuta la consulta SQL con los valores y el ID

  return result; // 7.4 Devuelve el resultado de la actualización

}

// 8. Función para eliminar una entrada por ID de cualquier tabla

async function deleteEntryById(table, idColumn, id) {

  const sql = `DELETE FROM ${table} WHERE ${idColumn} = ?`; // 8.1 Construye la consulta SQL de eliminación

  const [result] = await pool.execute(sql, [id]); // 8.2 Ejecuta la consulta SQL con el ID especificado

  return result; // 8.3 Devuelve el resultado de la eliminación

}

// 9. Definir las rutas utilizando Express Router

router.post('/create', async (req, res) => {

  try {

    const { table, columns, values } = req.body; // 9.1 Obtiene los datos de la solicitud

    const result = await createEntry(table, columns, values); // 9.2 Llama a la función de crear entrada

    res.status(201).json(result); // 9.3 Responde con el resultado

  } catch (error) {

    res.status(500).send('Error al crear la entrada'); // 9.4 Maneja los errores

  }

});

router.get('/entries/:table', async (req, res) => {

  try {

    const { table } = req.params; // 9.5 Obtiene el nombre de la tabla de los parámetros

    const entries = await getEntries(table); // 9.6 Llama a la función de obtener entradas

    res.json(entries); // 9.7 Responde con las entradas obtenidas

  } catch (error) {

    res.status(500).send('Error al obtener las entradas'); // 9.8 Maneja los errores

  }

});

router.get('/entry/:table/:idColumn/:id', async (req, res) => {

  try {

    const { table, idColumn, id } = req.params; // 9.9 Obtiene los datos de los parámetros

    const entry = await getEntryById(table, idColumn, id); // 9.10 Llama a la función de obtener entrada por ID

    if (entry) {

      res.json(entry); // 9.11 Responde con la entrada obtenida

    } else {

      res.status(404).send('Entrada no encontrada'); // 9.12 Maneja la entrada no encontrada

    }

  } catch (error) {

    res.status(500).send('Error al obtener la entrada'); // 9.13 Maneja los errores

  }

});

router.put('/update/:table/:idColumn/:id', async (req, res) => {

  try {

    const { table, idColumn, id } = req.params; // 9.14 Obtiene los datos de los parámetros

    const { columns, values } = req.body; // 9.15 Obtiene los datos del cuerpo de la solicitud

    const result = await updateEntryById(table, idColumn, id, columns, values); // 9.16 Llama a la función de actualizar entrada

    res.json(result); // 9.17 Responde con el resultado

  } catch (error) {

    res.status(500).send('Error al actualizar la entrada'); // 9.18 Maneja los errores

  }

});

router.delete('/delete/:table/:idColumn/:id', async (req, res) => {

  try {

    const { table, idColumn, id } = req.params; // 9.19 Obtiene los datos de los parámetros

    const result = await deleteEntryById(table, idColumn, id); // 9.20 Llama a la función de eliminar entrada

    res.json(result); // 9.21 Responde con el resultado

  } catch (error) {

    res.status(500).send('Error al eliminar la entrada'); // 9.22 Maneja los errores

  }

});

module.exports = router; // 10. Exporta el router para ser utilizado en app.js

# app.js 2

const express = require('express');

const cors = require('cors'); // Importar cors

const app = express();

app.use(cors()); // Habilitar CORS

app.use(express.json());

app.get('/', (req, res) => {

  res.send('Probando al Servidor - Prueba exitosa! :}');

});

const config = require('./config/config');

const server = app.listen(config.server.port, () => {

  console.log(`Servidor escuchando en el puerto ${config.server.port}`);

});

const sequelize = config.db;

sequelize.authenticate()

  .then(() => {

    console.log('Conectado a MariaDB usando Sequelize');

  })

  .catch(err => {

    console.error('No se pudo conectar a la base de datos:', err);

  });

const postRoute = require('./routes/post');

app.use('/', postRoute); // Eliminando el prefijo '/servicios'

function handleShutdown() {

  console.log('Cerrando servidor...');

  server.close(() => {

    console.log('Servidor cerrado.');

    process.exit(0);

  });

}

process.on('SIGINT', handleShutdown);

process.on('SIGTERM', handleShutdown);

# register.html 2

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta name="keywords" content="" />

    <meta name="description" content="" />

    <meta name="author" content="" />

    <link rel="shortcut icon" href="images/favicon.png" type="image/x-icon">

    <title>Maria E. girls Fashion Confecciones.</title>

    <!-- slider stylesheet -->

    <link rel="stylesheet" type="text/css" href="https://cdnjs.cloudflare.com/ajax/libs/OwlCarousel2/2.3.4/assets/owl.carousel.min.css" />

    <!-- bootstrap core css -->

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

    <!-- Custom styles for this template -->

    <link href="../css/style.css" rel="stylesheet"/>

    <!-- responsive style -->

    <link href="../css/responsive.css" rel="stylesheet" />

    <style>

      body {

        background-color: #f8f9fa;

      }

      .container {

        max-width: 600px;

        margin-top: 50px;

        padding: 20px;

        background-color: #ffffff;

        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

        border-radius: 8px;

      }

      .form-control:focus {

        box-shadow: none;

      }

    </style>

</head>

<body>

  <!-- header section strats -->

  <header class="header\_section">

    <nav class="navbar navbar-expand-lg custom\_nav-container ">

      <a class="navbar-brand" href="index.html">

        <span>

          Maria E. girls Fashion Confecciones.<br>

          <h6>🌟 ! Donde el estilo conoce la comodidad ! 🌟</h6>

        </span>

      </a>

      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

        <span class=""></span>

      </button>

      <div class="collapse navbar-collapse" id="navbarSupportedContent">

        <ul class="navbar-nav  ">

          <li class="nav-item active">

            <a class="nav-link" href="index.html">Inicio <span class="sr-only">(current)</span></a>

          </li>

          <li class="nav-item">

            <a class="nav-link" href="shop.html">

              Tienda

            </a>

          </li>

          <li class="nav-item">

            <a class="nav-link" href="why.html">

              Nosotros

            </a>

          </li>

          <li class="nav-item">

            <a class="nav-link" href="testimonial.html">

              Testimonios

            </a>

          </li>

          <li class="nav-item">

            <a class="nav-link" href="contact.html">

              Contáctenos

            </a>

          </li>

        </ul>

        <div class="user\_option">

          <a href="">

            <i class="fa fa-user" aria-hidden="true"></i>

            <span>

              Ingresar

            </span>

          </a>

          <a href="">

            <i class="fa fa-shopping-bag" aria-hidden="true"></i>

          </a>

          <form class="form-inline ">

            <button class="btn nav\_search-btn" type="submit">

              <i class="fa fa-search" aria-hidden="true"></i>

            </button>

          </form>

        </div>

      </div>

    </nav>

  </header>

  <!-- end header section -->

  <div class="container" id="purchaseForm" style="display: none;">

    <h2 class="text-center">Registro de Usuario</h2>

    <!-- Para apuntar a la ruta del servidor -->

    <form action="http://localhost:10000/create" method="POST"> <div class="form-group"> <label for="id\_tipo\_documento">Tipo de Documento</label> <input type="text" class="form-control" id="id\_tipo\_documento" name="id\_tipo\_documento" placeholder="Ingresa el tipo de documento" required> </div> <div class="form-group"> <label for="id\_cliente">Número de Documento</label> <input type="text" class="form-control" id="id\_cliente" name="id\_cliente" placeholder="Ingresa tu número de documento" required> </div> <div class="form-group"> <label for="nombre\_cliente">Nombre Completo</label> <input type="text" class="form-control" id="nombre\_cliente" name="nombre\_cliente" placeholder="Ingresa tu nombre completo" required> </div> <div class="form-group"> <label for="direccion\_cliente">Dirección</label> <input type="text" class="form-control" id="direccion\_cliente" name="direccion\_cliente" placeholder="Ingresa tu dirección" required> </div> <div class="form-group"> <label for="celular\_cliente">Teléfono</label> <input type="tel" class="form-control" id="celular\_cliente" name="celular\_cliente" placeholder="Ingresa tu número de teléfono" required> </div> <div class="form-group"> <label for="password\_cliente">Contraseña</label> <input type="password" class="form-control" id="password\_cliente" name="password\_cliente" placeholder="Ingresa tu contraseña" required> </div> <button type="submit" class="btn btn-primary btn-block">Registrar</button> </form>

  <!-- jQuery -->

  <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>

  <!-- Popper.js (necesario para Bootstrap) -->

  <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"></script>

  <!-- Bootstrap JS -->

  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

  <!-- Owl Carousel -->

  <script src="https://cdnjs.cloudflare.com/ajax/libs/OwlCarousel2/2.3.4/owl.carousel.min.js"></script>

  <!-- Custom Scripts -->

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

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

</body>

</html>