INSTITUTO EDUCACIÓN SUPERIOR TECNOLOGICO PÚBLICO

"PEDRO P. DÍAZ"

Programa de Estudios

DESARROLLO DE SISTEMAS DE INFORMACIÓN

Desarrollo de aplicaciones Backend

Docente:

Ing: Dulio Chavez

Semestre:

III

Integrantes:

- Cacya Cáceres José Luis (Prueba y Documentación)
- Ramos Mollocondo Camila (Programador)
- Quispe Nuñonca Alexandro (Jefe de proyecto y Documentación)

AREQUIPA – PERÚ

27/07/2025

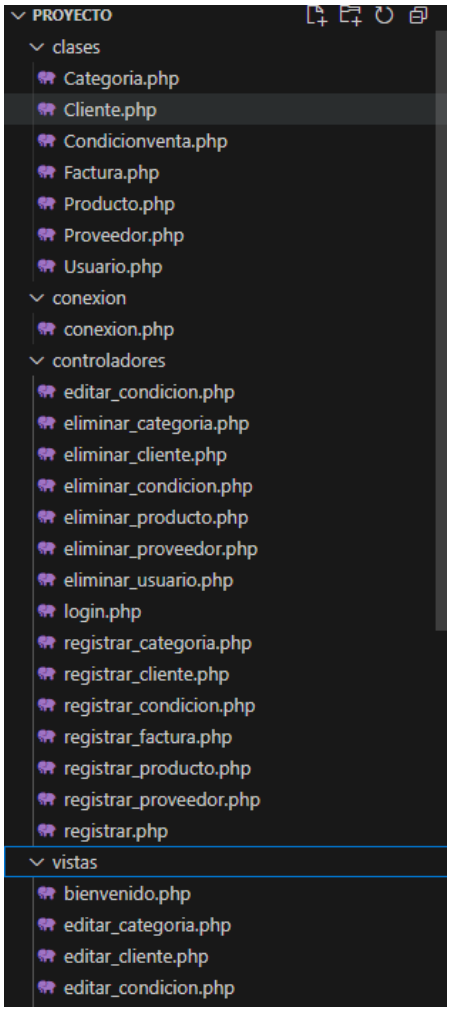
Índice del – Proyecto Backend

PROYECTO FINAL

Este documento describe la estructura de un proyecto de aplicación web, detallando las clases, archivos de conexión, controladores, vistas y pruebas para gestionar usuarios, clientes, proveedores, productos, categorías, condiciones de venta y facturas.

Estructura

Organiza el proyecto en componentes clave: clases para modelos de datos, conexión para acceso a la base de datos, controladores para la lógica de negocio, vistas para la interfaz de usuario y pruebas para validar funcionalidades.



ESPECIFICACIÓN

Lo que estamos apreciando es un sistema web desarrollado en PHP que sigue el patrón arquitectónico MVC (Modelo-Vista-Controlador), específicamente diseñado para la gestión integral de un negocio.

En la estructura podemos observar:

Las clases representan el modelo de datos de tu aplicación - tienes entidades como Categoria, Cliente, Factura, Producto, Proveedor y Usuario, que son los elementos centrales que tu sistema va a manejar.

Los controladores muestran una implementación completa de operaciones CRUD (Create, Read, Update, Delete) para cada entidad. Tienes controladores para registrar, editar y eliminar cada tipo de objeto, además de un sistema de autenticación con login.

Las vistas contienen la interfaz de usuario, incluyendo una página de bienvenida y formularios para la edición de diferentes entidades.

La conexión maneja la comunicación con la base de datos.

Lo que estamos viendo es esencialmente un ERP (Enterprise Resource Planning) básico que permitirá gestionar inventario, clientes, proveedores, facturación y usuarios del sistema. Es una aplicación de gestión empresarial bastante completa para pequeñas y medianas empresas

Clases

Contiene archivos de clases PHP que definen los modelos de datos y la lógica para las entidades del sistema.

- **Usuario.php**: Define la clase Usuario, que gestiona datos y operaciones relacionadas con usuarios, como autenticación y administración de perfiles.

Usuario.php | listar_usuarios.php | editar_usuario.php | eliminar_usuario.php | login.php

clases > Usuario.php

```php
<?php
require_once __DIR__ . '/../conexion/conexion.php';
class Usuario extends Conexion {
    public function __construct() {
        parent::__construct();
    }
    public function registrar($nomusuario, $password, $apellidos, $nombres, $email) {
        $hash = password_hash($password, PASSWORD_DEFAULT);
        $sql = "INSERT INTO usuarios (nomusuario, password, apellidos, nombres, email)
                VALUES (?, ?, ?, ?, ?)";
        $stmt = $this->conn->prepare($sql);
        $stmt->bind_param("sssss", $nomusuario, $hash, $apellidos, $nombres, $email);
        return $stmt->execute();
    }
    public function iniciarSesion($nomusuario, $password) {
        $sql = "SELECT * FROM usuarios WHERE nomusuario = ? AND estado = 'activo'";
        $stmt = $this->conn->prepare($sql);
        $stmt->bind_param("s", $nomusuario);
        $stmt->execute();
        $resultado = $stmt->get_result();

        if ($resultado->num_rows === 1) {
            $user = $resultado->fetch_assoc();
            if (password_verify($password, $user['password'])) {
                return $user;
            }
        }
        return false;
    }
```

ESPECIFICACIÓN

Lo que estamos apreciando es la clase Usuario.php, que representa el modelo de datos y la lógica de negocio para el manejo de usuarios en tu sistema.

Puntos clave del código:

Línea 3: La clase extiende de Conexion, heredando las funcionalidades de conexión a la base de datos.

Líneas 8-14: El método registrar() implementa la inserción de nuevos usuarios. Usa una consulta preparada con INSERT INTO usuarios y establece por defecto el estado como 'activo' y el hash de la contraseña.

Líneas 16-27: El método iniciarSesion() maneja la autenticación. Ejecuta un SELECT buscando al usuario por nombre y verificando que esté activo (línea 16).

Líneas 22-26: La validación crítica del login - primero verifica si existe el usuario ($resultado->num_rows === 1) y luego valida la contraseña usando password_verify() para comparar la contraseña ingresada con el hash almacenado.

Línea 25: Retorna el objeto usuario si la autenticación es exitosa.

Línea 29: Retorna false si falla cualquier validación.

Lo que vemos es una implementación robusta de autenticación con contraseñas hasheadas y consultas preparadas para prevenir inyección SQL, siguiendo buenas prácticas de seguridad.

- **Cliente.php**: Define la clase Cliente, que maneja información de clientes, como detalles de contacto e historial de transacciones.

```php
<?php
require_once __DIR__ . '/../conexion/conexion.php';

class Cliente extends Conexion {
    public function __construct() {
        parent::__construct();
    }

    // registrar nuevo cliente
    public function registrar($id, $nombre, $ruc, $direccion, $telefono, $email) {
        $sql = "INSERT INTO clientes (idcliente, nomcliente, ruccliente, dircliente, telcliente, emailcliente)
                VALUES (?, ?, ?, ?, ?, ?)";
        $stmt = $this->conn->prepare($sql);
        $stmt->bind_param("ssssss", $id, $nombre, $ruc, $direccion, $telefono, $email);
        return $stmt->execute();
    }

    // obtener todos los clientes
    public function obtenerTodos() {
        $sql = "SELECT * FROM clientes";
        $res = $this->conn->query($sql);
        return $res->fetch_all(MYSQLI_ASSOC);
    }

    // obtener cliente por id
    public function obtenerPorId($id) {
        $sql = "SELECT * FROM clientes WHERE idcliente=?";
        $stmt = $this->conn->prepare($sql);
        $stmt->bind_param("s", $id);
        $stmt->execute();
        return $stmt->get_result()->fetch_assoc();
    }
```

ESPECIFICACIÓN

Lo que estamos apreciando es la clase Cliente.php, que maneja toda la lógica de negocio relacionada con los clientes del sistema.

Puntos clave del código:
Líneas 6-12: El método registrar() inserta nuevos clientes en la base de datos. Maneja múltiples campos como ID, nombre, RUC, dirección, teléfono y email, usando consultas preparadas para seguridad.

Líneas 15-20: El método obtenerTodos() recupera todos los clientes de la base de datos mediante un SELECT * FROM clientes, ejecuta la consulta y retorna todos los resultados usando fetch_all(MYSQL_ASSOC).

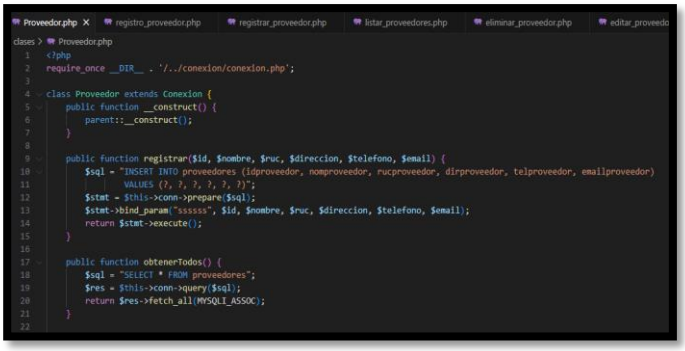
Líneas 23-29: El método obtenerPorId() permite buscar un cliente específico por su ID. Usa una consulta preparada con WHERE idcliente=? y retorna un único registro con fetch_assoc().

Aspectos importantes:

- Línea 2: Extiende de la clase Conexion para heredar funcionalidades de base de datos
- Líneas 8 y 25: Implementa consultas preparadas (prepare()) para prevenir inyección SQL
- Líneas 9 y 26: Usa bind_param() para vincular parámetros de forma segura

Lo que vemos es una implementación completa del patrón Repository para la entidad Cliente,

- **Proveedor.php**: Define la clase Proveedor, que gestiona datos de proveedores, incluyendo información de contacto y detalles de suministro.



ESPECIFICACIÓN

Lo que estamos apreciando es la clase Proveedor.php, que gestiona toda la funcionalidad relacionada con los proveedores en tu sistema de gestión empresarial.

Puntos clave del código:

Líneas 9-15: El método registrar() inserta nuevos proveedores con una amplia gama de campos. La consulta preparada maneja ID, nombre, RUC, dirección, teléfono y email, usando placeholders (?, ?, ?, ?, ?, ?) para máxima seguridad.
Línea 12: Implementa bind_param() con el tipo "ssssss" indicando que todos los parámetros son strings, vinculando de forma segura: $id, $nombre, $ruc, $direccion, $telefono, $email.
Líneas 18-23: El método obtenerTodos() recupera la lista completa de proveedores mediante SELECT * FROM proveedores, ejecuta la consulta y retorna todos los registros usando fetch_all(MYSQL_ASSOC) para obtener un array asociativo.

Aspectos destacados:

- Línea 4: Hereda de la clase Conexion para acceso a la base de datos
- Línea 10: Usa consultas preparadas para prevenir inyección SQL
- Línea 22: Retorna datos estructurados listos para usar en las vistas

Lo que vemos es una implementación sólida del modelo Proveedor que sigue el mismo patrón que Cliente, proporcionando operaciones fundamentales de registro y consulta con las mejores prácticas de seguridad en PHP.

- **Categoria.php**: Define la clase Categoría, que organiza las categorías de productos para filtrado y clasificación.

ESPECIFICACIÓN

Lo que estamos apreciando es la clase Categoria.php, que maneja la gestión de categorías de productos en tu sistema, siendo una de las clases más simples pero fundamentales.

Puntos clave del código:

Líneas 7-12: El método registrar() inserta nuevas categorías con una estructura muy sencilla. Usa INSERT INTO categorias (nombrecategoria) VALUES (?) - solo maneja el nombre de la categoría, lo cual es típico ya que el ID probablemente sea auto-incremental.

Línea 10: Implementa bind_param("s", $nombre) con un solo parámetro string, mostrando la simplicidad de esta entidad comparada con Cliente o Proveedor.

Líneas 15-19: El método obtenerTodos() recupera todas las categorías mediante SELECT * FROM categorias, retornando un array asociativo completo con fetch_all(MYSQL_ASSOC).

Líneas 22-27: El método obtenerPorId() busca una categoría específica usando WHERE idcategoria=?, retornando un único registro con fetch_assoc().

Aspectos destacados:

- Línea 4: Hereda de Conexion siguiendo el patrón establecido
- Estructura simple: Solo maneja nombre, a diferencia de las entidades más complejas
- Patrón consistente: Mantiene los mismos métodos (registrar, obtenerTodos, obtenerPorId) que las otras clases

Lo que vemos es una implementación limpia y minimalista para el manejo de categorías, esencial para clasificar productos en tu sistema de inventario.

- **Producto.php**: Define la clase Producto, que maneja detalles de productos como nombre, precio y existencias.

```php
<?php
require_once __DIR__.'/../conexion/conexion.php';

class Producto extends Conexion {
    public function __construct(){ parent::__construct(); }

    public function registrar($nom, $unidad, $stock, $costo, $precio, $idcat, $idprov) {
        $sql = "INSERT INTO productos (nomproducto, unimed, stock, cosuni, preuni, idcategoria, idproveedor)
                VALUES (?, ?, ?, ?, ?, ?, ?)";
        $stmt = $this->conn->prepare($sql);
        $stmt->bind_param("ssiddii", $nom, $unidad, $stock, $costo, $precio, $idcat, $idprov);
        return $stmt->execute();
    }

    public function obtenerTodos() {
        $sql = "SELECT p.*, c.nomcategoria, pr.nomproveedor
                FROM productos p
                LEFT JOIN categorias c ON p.idcategoria = c.idcategoria
                LEFT JOIN proveedores pr ON p.idproveedor = pr.idproveedor";
        $res = $this->conn->query($sql);
        return $res->fetch_all(MYSQLI_ASSOC);
    }
```

ESPECIFICACIÓN

Lo que estamos apreciando es la clase Producto.php, que representa la entidad más compleja y central de tu sistema de inventario, manejando toda la información de productos con sus relaciones.

Puntos clave del código:

Líneas 8-15: El método registrar() inserta productos con múltiples atributos empresariales críticos. Maneja nombre, unidad, stock, costo, precio, y crucialmente las claves foráneas idcat (categoría) e idprov (proveedor), estableciendo las relaciones entre entidades.

Línea 11: La consulta preparada usa 7 placeholders (?, ?, ?, ?, ?, ?, ?) para los campos: nombre, unidad, stock, costo, precio, idcategoría e idproveedor.

Líneas 18-25: El método obtenerTodos() implementa JOINS complejos para traer datos relacionados. Hace LEFT JOIN con las tablas categorias y proveedores para mostrar nombres legibles en lugar de solo IDs.

Líneas 20-22: Las uniones SQL permiten obtener c.nombrecategoria y pr.nomproveedor junto con los datos del producto, creando una vista completa y rica en información.

Aspectos destacados:

- Relaciones: Conecta productos con categorías y proveedores
- Datos financieros: Maneja costo, precio y stock para control de inventario
- Consultas avanzadas: Usa JOINs para presentar datos normalizados de forma legible

Lo que vemos es el corazón del sistema de inventario, donde convergen todas las relaciones y se gestiona la información más crítica del negocio.

- **Factura.php**: Define la clase Factura, que gestiona datos de facturas, incluyendo detalles de facturación y transacciones.

```php
<?php
require_once __DIR__ . '/../conexion/conexion.php';

class Factura extends Conexion {
    public function __construct() {
        parent::__construct();
    }

    public function registrar($fecha, $idcliente, $idusuario, $idcondicion, $valorventa, $igv, $detalles) {
        $this->conn->begin_transaction();

        $stmt = $this->conn->prepare("
            INSERT INTO facturas (fecha, idcliente, idusuario, idcondicion, valorventa, igv)
            VALUES (?, ?, ?, ?, ?, ?)
        ");
        $stmt->bind_param("siiidd", $fecha, $idcliente, $idusuario, $idcondicion, $valorventa, $igv);
        $stmt->execute();

        $idfactura = $this->conn->insert_id;
        $stmtDet = $this->conn->prepare("
            INSERT INTO detallefactura (idfactura, idproducto, cant, cosuni, preuni)
            VALUES (?, ?, ?, ?, ?)
        ");

        foreach ($detalles as $d) {
            $stmtDet->bind_param("iiidd", $idfactura, $d['idproducto'], $d['cant'], $d['cosuni'], $d['preuni']);
            $stmtDet->execute();
        }

        return $this->conn->commit();
    }
```

ESPECIFICACIÓN

Lo que estamos apreciando es la clase Factura.php, que implementa el módulo de facturación con transacciones de base de datos para garantizar integridad de datos.

Puntos clave del código:

Línea 12: Inicia una transacción con begin_transaction() para asegurar que toda la operación se complete correctamente o se revierta completamente.

Líneas 14-18: Registra la cabecera de la factura con datos como fecha, cliente, usuario, condición, valor de venta e IGV usando consultas preparadas.

Líneas 20-21: Obtiene el ID de la factura recién insertada con insert_id() para usarlo en los detalles.

Líneas 23-29: Inserta cada detalle de factura mediante un foreach que recorre el array $detalles, registrando producto, cantidad, costo y precio unitario para cada ítem.

Línea 32: Confirma toda la transacción con commit(), asegurando que tanto la factura como todos sus detalles se guarden correctamente.

Lo que vemos es una implementación robusta de facturación que maneja la complejidad de registros maestro-detalle con transacciones, típico de sistemas contables profesionales.

- **Condicionventa.php**: Define la clase Condición de Venta, que maneja los términos y condiciones de las transacciones de venta.

```php
<?php
require_once __DIR__ . '/../conexion/conexion.php';

class Condicionventa extends Conexion {
    public function __construct() {
        parent::__construct();
    }

    // Obtener todas las condiciones de venta
    public function obtenerTodas() {
        $sql = "SELECT * FROM condicionventa ORDER BY idcondicion";
        $res = $this->conn->query($sql);
        return $res->fetch_all(MYSQLI_ASSOC);
    }

    // Registrar nueva condición
    public function registrar($nomcondicion) {
        $stmt = $this->conn->prepare("INSERT INTO condicionventa (nomcondicion) VALUES (?)");
        $stmt->bind_param("s", $nomcondicion);
        return $stmt->execute();
    }
```

ESPECIFICACIÓN

Lo que estamos apreciando es la clase Condicionventa.php, que maneja los términos y condiciones comerciales para las transacciones de venta en tu sistema.

Puntos clave del código:

Líneas 12-17: El método obtenerTodas() recupera todas las condiciones de venta con ORDER BY idcondicion, proporcionando un listado ordenado de opciones como "Contado", "Crédito 30 días", etc.

Líneas 20-25: El método registrar() permite crear nuevas condiciones de venta usando una consulta preparada simple que solo maneja el nombre de la condición (nomcondicion).

Línea 16: Usa fetch_all(MYSQL_ASSOC) para retornar un array asociativo completo, facilitando su uso en formularios select o dropdowns.

Aspectos destacados:

- Funcionalidad empresarial: Esencial para definir si una venta es al contado, a crédito, o con plazos específicos
- Integración: Se relaciona con la tabla facturas para establecer términos de pago
- Simplicidad: Estructura minimalista pero fundamental para el flujo comercial

Lo que vemos es una clase de soporte que, aunque simple, es crucial para el módulo de facturación y define cómo se realizarán los cobros en tu sistema comercial.

Conexion

Administra la conectividad con la base de datos de la aplicación.

- **Conexión.php**: Establece y gestiona la conexión a la base de datos, permitiendo el acceso a los datos almacenados.

```php
<?php
class Conexion {
    private $host = "localhost";
    private $usuario = "root";
    private $contrasena = "";
    private $bd = "proyectofinal";
    protected $conn;

    public function __construct() {
        $this->conn = new mysqli($this->host, $this->usuario, $this->contrasena, $this->bd);
        if ($this->conn->connect_error) {
            die("Conexión fallida: " . $this->conn->connect_error);
        }
    }

    public function obtenerConexion() {
        return $this->conn;
    }
}
?>
```

ESPECIFICACIÓN

Lo que estamos apreciando es la clase Conexion.php, que representa la base fundamental de todo tu sistema, manejando la conectividad con la base de datos MySQL.

Puntos clave del código:

Líneas 3-6: Define las credenciales de conexión como propiedades privadas: host (localhost), usuario (root), contraseña vacía, y base de datos (proyectofinal).

Líneas 9-15: El constructor establece la conexión automáticamente usando mysqli() con los parámetros definidos, e implementa manejo de errores con connect_error para detectar fallos de conexión.

Línea 14: Si hay error de conexión, termina la ejecución con die() mostrando el mensaje de error específico.

Líneas 18-20: El método obtenerConexion() actúa como getter para retornar el objeto de conexión, permitiendo que las clases hijas accedan a $this->conn.

Aspectos destacados:

- Patrón Singleton implícito: Una conexión por instancia
- Herencia: Todas las clases del sistema extienden de esta clase base
- Configuración local: Credenciales típicas de desarrollo (root sin password)

Lo que vemos es el pilar arquitectónico que gestiona todo el sistema, proporcionando conectividad

Controladores

Contiene archivos PHP que manejan la lógica de negocio y procesan solicitudes para acciones como editar, eliminar y registrar entidades.

- **editar_condicion.php**: Controlador para editar detalles de una condición de venta.

```php
<?php
require_once '../clases/Condicionventa.php';

if ($_POST) {
    $obj = new Condicionventa();
    $obj->actualizar($_POST['idcondicion'], $_POST['nomcondicion']);
    header("Location: ../vistas/listar_condicion.php");
}
```

ESPECIFICACIÓN

Lo que estamos apreciando es un controlador de edición para condiciones de venta. Línea 4: Valida petición POST del formulario. Línea 5: Instancia objeto Condicionventa(). Línea 6: Ejecuta actualizar() con ID y nombre desde $_POST. Línea 7: Redirige a vista de listado tras completar la operación. Patrón MVC clásico: recibe datos, procesa con modelo, redirige a vista.

- **eliminar_categoria.php**: Controlador para eliminar una categoría de producto del sistema.

controladores > eliminar_categoria.php

```php
<?php
require '../clases/Categoria.php';
if(isset($_GET['id'])){
  (new Categoria())->eliminar($_GET['id']);
}
header("Location: ../vistas/listar_categorias.php");

```

ESPECIFICACIÓN

Lo que estamos apreciando es un controlador de eliminación para categorías. Línea 2: Incluye la clase Categoria.php. Línea 3: Valida que existe el parámetro id en GET. Línea 4: Instancia objeto Categoria() y ejecuta eliminar() con el ID recibido por URL. Línea 6: Redirige a vista de listado tras la eliminación. Controlador simple para operaciones DELETE del CRUD.

- **eliminar_cliente.php**: Controlador para eliminar un cliente de la base de datos.

controladores > eliminar_cliente.php

```php
<?php
require_once '../clases/Cliente.php';

if (isset($_GET['id'])) {
    $cliente = new Cliente();
    $cliente->eliminar($_GET['id']);
}

header("Location: ../vistas/listar_clientes.php");

```

ESPECIFICACIÓN

Lo que estamos apreciando es un controlador de eliminación para clientes. Línea 2: Incluye la clase Cliente.php. Línea 4: Valida que existe el parámetro id en GET. Líneas 5-6: Instancia objeto Cliente() y ejecuta eliminar() con el ID recibido por URL. Línea 9: Redirige a vista de listado de clientes tras la eliminación. Controlador DELETE estándar del patrón CRUD.

- **eliminar_condicion.php**: Controlador para borrar una condición de venta.

controladores > eliminar_condicion.php

```php
<?php
require_once '../clases/Condicionventa.php';

if (isset($_GET['id'])) {
    $obj = new Condicionventa();
    $obj->eliminar($_GET['id']);
    header("Location: ../vistas/listar_condicion.php");
}

```

ESPECIFICACIÓN

Lo que estamos apreciando es un controlador de eliminación para condiciones de venta. Línea 2: Incluye la clase Condicionventa.php. Línea 4: Valida que existe el parámetro id en GET. Líneas 5-6: Instancia objeto Condicionventa() y ejecuta eliminar() con el ID recibido por URL. Línea 7: Redirige a vista de listado de condiciones tras la eliminación. Controlador DELETE del CRUD para términos comerciales.

- **eliminar_producto.php**: Controlador para eliminar un producto del sistema.

controladores > eliminar_producto.php

```php
<?php
require_once '../clases/Producto.php';
if(isset($_GET['id'])){
  (new Producto())->eliminar($_GET['id']);
}
header("Location: ../vistas/listar_productos.php");

```

ESPECIFICACIÓN

Lo que estamos apreciando es un controlador de eliminación para productos. Línea 2: Incluye la clase Producto.php. Línea 3: Valida que existe el parámetro id en GET. Línea 4: Instancia objeto Producto() y ejecuta eliminar() con el ID recibido por URL en una sola línea. Línea 6: Redirige a vista de listado de productos tras la eliminación. Controlador DELETE compacto para gestión de inventario.

- **eliminar_proveedor.php**: Controlador para eliminar un proveedor de la base de datos.

controladores > eliminar_proveedor.php

```php
<?php
require_once '../clases/Proveedor.php';

if (isset($_GET['id'])) {
    $p = new Proveedor();
    $p->eliminar($_GET['id']);
}

header("Location: ../vistas/listar_proveedores.php");

```

ESPECIFICACIÓN

Lo que estamos apreciando es un controlador de eliminación para proveedores. Línea 2: Incluye la clase Proveedor.php. Línea 4: Valida que existe el parámetro id en GET. Líneas 5-6: Instancia objeto Proveedor() (variable $p) y ejecuta eliminar() con el ID recibido por URL. Línea 9: Redirige a vista de listado de proveedores tras la eliminación. Controlador DELETE para gestión de proveedores comerciales.

- **Controladores/eliminar_usuario.php**: Controlador para eliminar un usuario del sistema.

```php
<?php
require_once '../clases/Usuario.php';

if (isset($_GET['id'])) {
    $usuarioObj = new Usuario();
    $usuarioObj->eliminar($_GET['id']);
}

header("Location: ../vistas/listar_usuarios.php");
```

ESPECIFICACIÓN

Lo que estamos apreciando es un controlador de eliminación para usuarios. Línea 2: Incluye la clase Usuario.php. Línea 4: Valida que existe el parámetro id en GET. Líneas 5-6: Instancia objeto Usuario() (variable $usuarioobj) y ejecuta eliminar() con el ID recibido por URL. Línea 9: Redirige a vista de listado de usuarios tras la eliminación. Controlador DELETE para gestión de usuarios del sistema.

- **Controladores/login.php**: Controlador para manejar la autenticación de inicio de sesión de usuarios.

```php
<?php
session_start();
require_once '../clases/Usuario.php';

$nomusuario = $_POST['nomusuario'];
$password = $_POST['password'];

$usuarioObj = new Usuario();
$datos = $usuarioObj->iniciarSesion($nomusuario, $password);

if ($datos) {
    $_SESSION['idusuario'] = $datos['idusuario'];
    $_SESSION['nomusuario'] = $datos['nomusuario'];
    $_SESSION['nombres'] = $datos['nombres'];
    header("Location: ../vistas/bienvenido.php");
} else {
    echo "Credenciales inválidas o usuario inactivo. <a href='../vistas/login.html'>Volver</a>";
}
?>
```

ESPECIFICACIÓN

Lo que estamos apreciando es el controlador de autenticación (login). Línea 1: Inicia sesión PHP. Líneas 4-5: Captura credenciales del formulario POST. Líneas 7-8: Instancia Usuario() y ejecuta iniciarSesion() con las credenciales. Líneas 10-15: Si la autenticación es exitosa, guarda datos del usuario en $_SESSION y redirige al dashboard. Línea 17: Si falla, muestra mensaje de error y enlace de retorno. Controlador completo de autenticación con manejo de sesiones.

- **Controladores/registrar.php**: Controlador para registrar nuevos usuarios en el sistema.

```php
<?php
require_once '../clases/Usuario.php';

$nomusuario = $_POST['nomusuario'];
$password = $_POST['password'];
$apellidos = $_POST['apellidos'];
$nombres = $_POST['nombres'];
$email = $_POST['email'];

$usuarioObj = new Usuario();
if ($usuarioObj->registrar($nomusuario, $password, $apellidos, $nombres, $email)) {
    echo "Usuario registrado correctamente. <a href='../vistas/login.html'>Iniciar sesión</a>";
} else {
    echo "Error al registrar.";
}
?>
```

ESPECIFICACIÓN

Lo que estamos apreciando es el controlador de registro de usuarios. Líneas 3-7: Captura todos los datos del formulario POST (usuario, password, apellidos, nombres, email). Línea 9: Instancia objeto Usuario(). Línea 10: Ejecuta registrar() con todos los parámetros capturados. Líneas 10-11: Si el registro es exitoso, muestra mensaje de confirmación con enlace al login. Líneas 12-14: Si falla, muestra mensaje de error. Controlador de registro completo con validación y retroalimentación al usuario.

-
- **registrar_categoria.php**: Controlador para agregar una nueva categoría de producto.

controladores > registrar_categoria.php

```php
<?php
require '../clases/Categoria.php';
if ($_POST) {
    $cat = new Categoria();
    $cat->registrar($_POST['nomcategoria']);
    header("Location: ../vistas/listar_categorias.php");
}

```

ESPECIFICACIÓN

Lo que estamos apreciando es el controlador de registro para categorías. Línea 2: Incluye la clase Categoria.php. Línea 3: Valida petición POST del formulario. Líneas 4-5: Instancia objeto Categoria() y ejecuta registrar() con el nombre capturado desde $_POST['nomcategoria']. Línea 6: Redirige a vista de listado tras completar el registro. Controlador CREATE simple del CRUD para clasificación de productos.

- **registrar_cliente.php**: Controlador para registrar un nuevo cliente.

```php
<?php
require_once '../clases/Cliente.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $cliente = new Cliente();
    $cliente->registrar(
        $_POST['idcliente'],
        $_POST['nomcliente'],
        $_POST['ruccliente'],
        $_POST['dircliente'],
        $_POST['telcliente'],
        $_POST['emailcliente']
    );
    header("Location: ../vistas/listar_clientes.php");
}
```

ESPECIFICACIÓN

Lo que estamos apreciando es el controlador de registro para clientes. Línea 2: Incluye la clase Cliente.php. Línea 4: Valida método POST usando $_SERVER['REQUEST_METHOD']. Líneas 5-13: Instancia objeto Cliente() y ejecuta registrar() con todos los datos del formulario (ID, nombre, RUC, dirección, teléfono, email). Línea 14: Redirige a vista de listado tras completar el registro. Controlador CREATE completo para gestión de clientes empresariales.

- **registrar_condicion.php**: Controlador para agregar una nueva condición de venta.

controladores > registrar_condicion.php

```php
<?php
require_once '../clases/Condicionventa.php';

if ($_POST) {
    $obj = new Condicionventa();
    $obj->registrar($_POST['nomcondicion']);
    header("Location: ../vistas/listar_condicion.php");
}

```

ESPECIFICACIÓN

Lo que estamos apreciando es el controlador de registro para condiciones de venta. Línea 2: Incluye la clase Condicionventa.php. Línea 4: Valida petición POST del formulario. Líneas 5-6: Instancia objeto Condicionventa() y ejecuta registrar() con el nombre capturado desde $_POST['nomcondicion']. Línea 7: Redirige a vista de listado tras completar el registro. Controlador CREATE para términos comerciales del sistema de facturación.

- **registrar_factura.php**: Controlador para crear una nueva factura.

controladores > registrar_factura.php

```php
<?php
require '../clases/Factura.php';

if ($_POST) {
    $det = array_values($_POST['detalles']);
    (new Factura())->registrar(
        $_POST['fecha'],
        $_POST['idcliente'],
        $_SESSION['idusuario'], // o bien un select si admin genera
        $_POST['idcondicion'],
        $_POST['valorventa'] ?? 0,
        $_POST['igv'] ?? 0,
        $det
    );
    header("Location: ../vistas/listar_facturas.php");
}

```

ESPECIFICACIÓN

Lo que estamos apreciando es el controlador de registro para facturas. Línea 2: Incluye la clase Factura.php. Línea 5: Extrae array de detalles con array_values() desde $_POST['detalles']. Líneas 6-14: Instancia Factura() y ejecuta registrar() con datos de cabecera (fecha, cliente, usuario de sesión, condición, valores) y el array de detalles de productos. Líneas 11-12: Usa operador ternario ?? 0 para valores por defecto. Línea 15: Redirige a listado tras registro. Controlador CREATE complejo para transacciones maestro-detalle.

- **registrar_producto.php**: Controlador para agregar un nuevo producto.

controladores > registrar_producto.php

```php
<?php
require_once '../clases/Producto.php';
if($_POST){
  (new Producto())->registrar(
    $_POST['nomproducto'],$_POST['unimed'],$_POST['stock'],
    $_POST['cosuni'],$_POST['preuni'],$_POST['idcategoria'],$_POST['idproveedor']
  );
}
header("Location: ../vistas/listar_productos.php");

```

ESPECIFICACIÓN

Lo que estamos apreciando es el controlador de registro para productos. Línea 2: Incluye la clase Producto.php. Línea 3: Valida petición POST del formulario. Líneas 4-7: Instancia Producto() y ejecuta registrar() con todos los datos del inventario (nombre, unidad, stock, costo, precio, categoría, proveedor). Línea 9: Redirige a vista de listado tras completar el registro. Controlador CREATE para la entidad más compleja del sistema de inventario.

- **registrar_proveedor.php**: Controlador para registrar un nuevo proveedor.

controladores > registrar_proveedor.php

```php
<?php
require_once '../clases/Proveedor.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $p = new Proveedor();
    $p->registrar($_POST['id'], $_POST['nombre'], $_POST['ruc'], $_POST['direccion'], $_POST['telefono'], $_POST['email']);
    header("Location: ../vistas/listar_proveedores.php");
}

```

ESPECIFICACIÓN

Lo que estamos apreciando es el controlador de registro para proveedores. Línea 2: Incluye la clase Proveedor.php. Línea 4: Valida método POST usando $_SERVER['REQUEST_METHOD']. Líneas 5-6: Instancia Proveedor() y ejecuta registrar() con todos los datos empresariales (ID, nombre, RUC, dirección, teléfono, email). Línea 7: Redirige a vista de listado tras completar el registro. Controlador CREATE para gestión de proveedores comerciales.

Vistas

Contiene archivos PHP y HTML que definen la interfaz de usuario para mostrar e interactuar con los datos.

- **Vistas/bienvenido.php**: Muestra una página de bienvenida para usuarios que han iniciado sesión.

```php
<?php
session_start();
if (!isset($_SESSION['nomusuario'])) {
    header("Location: login.html");
    exit();
}
?>

<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Bienvenido</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light">
  <div class="container py-5">
    <div class="alert alert-success text-center">
      <h3>Bienvenido, <?php echo $_SESSION['nombres']; ?> 👋</h3>
      <p>Usuario: <strong><?php echo $_SESSION['nomusuario']; ?></strong></p>
      <a href="../logout.php" class="btn btn-danger">Cerrar sesión</a>
        <a href="listar_usuarios.php" class="btn btn-warning">Ver Usuarios</a>
    </div>
  </div>
</body>
</html>
```

ESPECIFICACIÓN

Es una página protegida que solo pueden ver usuarios autenticados. Si alguien intenta acceder sin estar logueado, es redirigido al formulario de login. Si está logueado, ve un panel con opciones para cerrar sesión o ver una lista de usuarios.

- **editar_categoria.php**: Interfaz para editar detalles de una categoría de producto.

```php
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Editar Categoría</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light">

<div class="container mt-5">
  <h3 class="mb-4">Editar Categoría</h3>
  <form method="post" class="shadow p-4 bg-white rounded">
    <input type="hidden" name="idcategoria" value="<?= $cat['idcategoria'] ?>">
    <div class="mb-3">
      <label class="form-label">Nombre de la Categoría</label>
      <input type="text" class="form-control" name="nomcategoria" value="<?= $cat['nomcategoria'] ?>" required>
    </div>
    <button type="submit" class="btn btn-primary">Guardar Cambios</button>
    <a href="listar_categorias.php" class="btn btn-secondary">Cancelar</a>
  </form>
</div>
```

ESPECIFICACIÓN

Es un formulario de edición de categorías que permite modificar datos existentes. El campo oculto idcategoria identifica qué registro se está editando, mientras que el input nomcategoria viene prellenado con el valor actual desde la variable PHP $cat['nomcategoria'].

Al hacer submit, los datos se procesan vía POST para actualizar la categoría en la base de datos. Si el usuario cancela, regresa al listado de categorías mediante listar_categorias.php.

El formulario usa Bootstrap para estilos responsive y validación HTML5 con el atributo required para garantizar que el nombre de categoría no esté vacío antes del envío.

- **editar_cliente.php**: Interfaz para actualizar información de un cliente.

```php
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Editar Cliente</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light p-4">
  <div class="container">
    <h2 class="text-center mb-4">Editar Cliente</h2>
    <form method="post">
      <input type="hidden" name="idcliente" value="<?= $data['idcliente'] ?>">
      <div class="mb-3"><label>Nombre</label><input type="text" name="nomcliente" class="form-control" value="<?= $data['nomcliente'] ?
      <div class="mb-3"><label>RUC</label><input type="text" name="ruccliente" class="form-control" value="<?= $data['ruccliente'] ?>"
      <div class="mb-3"><label>Dirección</label><input type="text" name="dircliente" class="form-control" value="<?= $data['dircliente'
      <div class="mb-3"><label>Teléfono</label><input type="text" name="telcliente" class="form-control" value="<?= $data['telcliente'
      <div class="mb-3"><label>Email</label><input type="email" name="emailcliente" class="form-control" value="<?= $data['emailcliente
      <button type="submit" class="btn btn-primary">Actualizar</button>
      <a href="listar_clientes.php" class="btn btn-secondary">Cancelar</a>
    </form>
  </div>
</body>
</html>
```

ESPECIFICACIÓN

Es un formulario UPDATE para clientes que permite modificar todos los datos empresariales. Los campos vienen precargados desde el array PHP $data y al enviar se actualizan en la base de datos. Incluye validación visual con Bootstrap y navegación de retorno al listado.

- **editar_condicion.php**: Interfaz para modificar detalles de una condición de venta.

```php
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Editar Condición</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light p-4">
<div class="container">
  <h3 class="mb-4">Editar Condición de Venta</h3>
  <form action="../controladores/editar_condicion.php" method="post" class="bg-white p-4 rounded shadow">
    <input type="hidden" name="idcondicion" value="<?= $condicion['idcondicion'] ?>">
    <div class="mb-3">
      <label for="nomcondicion" class="form-label">Nombre</label>
      <input type="text" name="nomcondicion" class="form-control" value="<?= $condicion['nomcondicion'] ?>" required>
    </div>
    <button type="submit" class="btn btn-primary">Actualizar</button>
    <a href="listar_condicion.php" class="btn btn-secondary">Cancelar</a>
  </form>
</div>
</body>
</html>
```

ESPECIFICACIÓN

Es un formulario UPDATE para condiciones de venta que permite modificar el nombre de condiciones existentes (como "Contado", "Crédito 30 días", etc.). El campo viene precargado desde el array PHP $condicion y al enviar se procesa la actualización en el controlador correspondiente.

- **editar_productos.php**: Interfaz para actualizar información de productos.

```php
<div class="container mt-5">
  <h3 class="mb-4">Editar Producto</h3>
  <form method="post" class="p-4 bg-white shadow rounded">
    <input type="hidden" name="idproducto" value="<?= $prod['idproducto'] ?>">
    <div class="mb-3"><label>Nombre</label><input class="form-control" name="nomproducto" value="<?= $prod['nomproducto'] ?>" required
    <div class="mb-3"><label>Unidad</label><input class="form-control" name="unimed" value="<?= $prod['unimed'] ?>"></div>
    <div class="mb-3"><label>Stock</label><input type="number" class="form-control" name="stock" value="<?= $prod['stock'] ?>"></div>
    <div class="mb-3"><label>Costo unitario</label><input type="number" step="0.0001" class="form-control" name="cosuni" value="<?= $p
    <div class="mb-3"><label>Precio unitario</label><input type="number" step="0.0001" class="form-control" name="preuni" value="<?= $
    <div class="mb-3"><label>Categoría</label>
      <select name="idcategoria" class="form-select">
        <?php foreach($cats as $c): ?>
        <option value="<?= $c['idcategoria'] ?>" <?= $prod['idcategoria']==$c['idcategoria']?'selected':'' ?>><?= $c['nomcategoria'] ?
        <?php endforeach ?>
      </select>
    </div>
    <div class="mb-3"><label>Proveedor</label>
      <select name="idproveedor" class="form-select">
        <?php foreach($provs as $p): ?>
        <option value="<?= $p['idproveedor'] ?>" <?= $prod['idproveedor']==$p['idproveedor']?'selected':'' ?>><?= $p['nomproveedor'] ?
```

ESPECIFICACIÓN

Es un formulario UPDATE completo para productos que permite editar datos básicos (nombre, unidad, stock, costos, precios) y relaciones con categorías y proveedores. Los selects vienen precargados con las opciones correctas seleccionadas según los datos actuales del producto.

- **editar_proveedor.php**: Interfaz para editar detalles de un proveedor.

```php
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Editar Proveedor</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light p-4">
  <div class="container">
    <h2 class="text-center mb-4">Editar Proveedor</h2>
    <form method="post">
      <input type="hidden" name="id" value="<?= $prov['idproveedor'] ?>">
      <div class="mb-3"><label>Nombre</label><input type="text" name="nombre" class="form-control" value="<?= $prov['nomproveedor'] ?>
      <div class="mb-3"><label>RUC</label><input type="text" name="ruc" class="form-control" value="<?= $prov['rucproveedor'] ?>"></div
      <div class="mb-3"><label>Dirección</label><input type="text" name="direccion" class="form-control" value="<?= $prov['dirproveedor
      <div class="mb-3"><label>Teléfono</label><input type="text" name="telefono" class="form-control" value="<?= $prov['telproveedor'
      <div class="mb-3"><label>Email</label><input type="email" name="email" class="form-control" value="<?= $prov['emailproveedor'] ?>
      <button type="submit" class="btn btn-primary">Actualizar</button>
      <a href="listar_proveedores.php" class="btn btn-secondary">Cancelar</a>
    </form>
  </div>
</body>
</html>
```

ESPECIFICACIÓN

Es un formulario UPDATE para proveedores que permite modificar todos los datos comerciales de empresas proveedoras. Los campos vienen precargados desde el array PHP $prov y al enviar se actualizan en la base de datos. Incluye validación de email y navegación de retorno al listado de proveedores.

- **listar_categorias.php**: Muestra una lista de todas las categorías de productos.

```php
<div class="container mt-5">
  <div class="d-flex justify-content-between align-items-center mb-3">
    <h3>Categorías Registradas</h3>
    <a href="registro_categoria.php" class="btn btn-primary">Nueva Categoría</a>
  </div>
  <table class="table table-striped table-bordered">
    <thead class="table-dark">
      <tr>
        <th>ID</th>
        <th>Nombre de Categoría</th>
        <th>Acciones</th>
      </tr>
    </thead>
    <tbody>
      <?php foreach ($cats as $c): ?>
      <tr>
        <td><?= $c['idcategoria'] ?></td>
        <td><?= $c['nomcategoria'] ?></td>
        <td>
          <a href="editar_categoria.php?id=<?= $c['idcategoria'] ?>" class="btn btn-sm btn-warning">Editar</a>
          <a href="../controladores/eliminar_categoria.php?id=<?= $c['idcategoria'] ?>" class="btn btn-sm btn-danger" onclick="return
        </td>
      </tr>
      <?php endforeach ?>
    </tbody>
  </table>
</div>
```

ESPECIFICACIÓN

Es una vista INDEX para gestión de categorías que permite visualizar todas las categorías registradas en formato tabla, con opciones para crear nuevas categorías, editar existentes y eliminar con confirmación JavaScript. Utiliza Bootstrap para diseño responsive y estilos de botones.

- **listar_clientes.php**: Muestra una lista de todos los clientes registrados.

```php
<body class="bg-light p-4">
  <div class="container">
    <div class="d-flex justify-content-between align-items-center mb-3">
      <h2>Listado de Clientes</h2>
      <a href="registro_cliente.php" class="btn btn-success">Nuevo Cliente</a>
    </div>
    <table class="table table-bordered table-hover">
      <thead class="table-dark">
        <tr>
          <th>ID</th><th>Nombre</th><th>RUC</th><th>Dirección</th><th>Teléfono</th><th>Email</th><th>Acciones</th>
        </tr>
      </thead>
      <tbody>
        <?php foreach ($clientes as $c): ?>
        <tr>
          <td><?= $c['idcliente'] ?></td>
          <td><?= $c['nomcliente'] ?></td>
          <td><?= $c['ruccliente'] ?></td>
          <td><?= $c['dircliente'] ?></td>
          <td><?= $c['telcliente'] ?></td>
          <td><?= $c['emailcliente'] ?></td>
          <td>
            <a href="editar_cliente.php?id=<?= $c['idcliente'] ?>" class="btn btn-warning btn-sm">Editar</a>
            <a href="../controladores/eliminar_cliente.php?id=<?= $c['idcliente'] ?>" class="btn btn-danger btn-sm" onclick="return co
          </td>
        </tr>
        <?php endforeach ?>
      </tbody>
    </table>
  </div>
</body>
</html>
```

ESPECIFICACIÓN

Es una vista INDEX completa para gestión de clientes que muestra todos los datos empresariales en formato tabla responsive, con opciones para crear nuevos clientes, editar información existente y eliminar registros con confirmación JavaScript.

- **listar_condicion.php**: Muestra una lista de todas las condiciones de venta.

```
  <title>Condiciones de Venta</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light p-4">
<div class="container">
  <h3 class="mb-4">Lista de Condiciones de Venta</h3>
  <a href="registro_condicion.php" class="btn btn-success mb-3">Registrar Nueva Condición</a>
  <table class="table table-bordered table-striped">
    <thead class="table-dark">
      <tr>
        <th>ID</th>
        <th>Nombre de Condición</th>
        <th>Acciones</th>
      </tr>
    </thead>
    <tbody>
      <?php foreach ($condiciones as $c): ?>
      <tr>
        <td><?= $c['idcondicion'] ?></td>
        <td><?= $c['nomcondicion'] ?></td>
        <td>
          <a href="editar_condicion.php?id=<?= $c['idcondicion'] ?>" class="btn btn-sm btn-warning">Editar</a>
          <a href="../controladores/eliminar_condicion.php?id=<?= $c['idcondicion'] ?>" class="btn btn-sm btn-danger" onclick="return
        </td>
      </tr>
      <?php endforeach ?>
    </tbody>
  </table>
</div>
</body>
</html>
```

ESPECIFICACIÓN

Es una vista INDEX para gestión de condiciones de venta que permite visualizar, crear, editar y eliminar condiciones comerciales (términos de pago) con confirmación JavaScript para eliminación y diseño responsive con Bootstrap.

- **listar_facturas.php**: Muestra una lista de todas las facturas.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Facturas</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light p-4">
<div class="container">
  <div class="d-flex justify-content-between mb-3">
    <h3>Facturas</h3>
    <a href="registro_factura.php" class="btn btn-primary">Nueva Factura</a>
  </div>
  <table class="table table-striped table-bordered">
    <thead class="table-dark"><tr>
      <th>ID</th><th>Fecha</th><th>Cliente</th><th>Usuario</th><th>Condición</th><th>Valor</th><th>IGV</th><th>Acciones</th>
    </tr></thead>
    <tbody>
      <?php foreach ($fac as $f): ?>
      <tr>
        <td><?= $f['idfactura'] ?></td>
        <td><?= $f['fecha'] ?></td>
        <td><?= $f['nomcliente'] ?></td>
        <td><?= $f['nomusuario'] ?></td>
        <td><?= $f['nomcondicion'] ?></td>
        <td><?= $f['valorventa'] ?></td>
        <td><?= $f['igv'] ?></td>
        <td>
          <a href="ver_factura.php?id=<?= $f['idfactura'] ?>" class="btn btn-sm btn-info">Ver</a>
        </td>
      </tr>
```

ESPECIFICACIÓN

Es una vista INDEX para gestión de facturas que muestra el registro completo de ventas con información del cliente, usuario, condiciones comerciales, valores monetarios e IGV. Permite crear nuevas facturas y ver detalles de facturas existentes con diseño responsive.

- **listar_productos.php**: Muestra una lista de todos los productos.

```
<div class="container mt-5">
  <div class="d-flex justify-content-between align-items-center mb-3">
    <h3>Productos</h3>
    <a href="registro_producto.php" class="btn btn-primary">Nuevo Producto</a>
  </div>
  <table class="table table-striped table-bordered">
    <thead class="table-dark"><tr>
      <th>ID</th><th>Nombre</th><th>Unidad</th><th>Stock</th><th>Costo</th><th>Precio</th>
      <th>Categoría</th><th>Proveedor</th><th>Estado</th><th>Acciones</th>
    </tr></thead>
    <tbody>
      <?php foreach($prods as $p): ?>
      <tr>
        <td><?= $p['idproducto'] ?></td>
        <td><?= $p['nomproducto'] ?></td>
        <td><?= $p['unimed'] ?></td>
        <td><?= $p['stock'] ?></td>
        <td><?= $p['cosuni'] ?></td>
        <td><?= $p['preuni'] ?></td>
        <td><?= $p['nomcategoria'] ?></td>
        <td><?= $p['nomproveedor'] ?></td>
        <td><?= $p['estado'] ?></td>
        <td>
          <a href="editar_productos.php?id=<?= $p['idproducto'] ?>" class="btn btn-sm btn-warning">Editar</a>
          <a href="../controladores/eliminar_producto.php?id=<?= $p['idproducto'] ?>" class="btn btn-sm btn-danger" onclick="r
        </td>
      </tr>
      <?php endforeach ?>
    </tbody>
  </table>
</div>
</body>
</html>
```

ESPECIFICACIÓN

Es una vista INDEX completa para gestión de inventario que muestra todos los productos con información comercial, stock, precios, relaciones con categorías y proveedores. Permite crear, editar y eliminar productos con confirmación JavaScript y diseño responsive.

- **listar_proveedores.php**: Muestra una lista de todos los proveedores.

```
<body class="bg-light p-4">
  <div class="container">
    <div class="d-flex justify-content-between mb-3">
      <h2>Proveedores Registrados</h2>
      <a href="registro_proveedor.php" class="btn btn-success">Nuevo Proveedor</a>
    </div>
    <table class="table table-bordered table-hover">
      <thead class="table-dark">
        <tr>
          <th>ID</th><th>Nombre</th><th>RUC</th><th>Dirección</th><th>Teléfono</th><th>Email</th><th>Acciones</th>
        </tr>
      </thead>
      <tbody>
        <?php foreach ($proveedores as $prov): ?>
        <tr>
          <td><?= $prov['idproveedor'] ?></td>
          <td><?= $prov['nomproveedor'] ?></td>
          <td><?= $prov['rucproveedor'] ?></td>
          <td><?= $prov['dirproveedor'] ?></td>
          <td><?= $prov['telproveedor'] ?></td>
          <td><?= $prov['emailproveedor'] ?></td>
          <td>
            <a href="editar_proveedor.php?id=<?= $prov['idproveedor'] ?>" class="btn btn-sm btn-warning">Editar</a>
            <a href="../controladores/eliminar_proveedor.php?id=<?= $prov['idproveedor'] ?>" class="btn btn-sm btn-danger" onclick="r
          </td>
        </tr>
        <?php endforeach ?>
      </tbody>
    </table>
  </div>
</body>
</html>
```

ESPECIFICACIÓN

Es una vista INDEX para gestión de proveedores que muestra todos los datos empresariales de empresas proveedoras en formato tabla responsive, con opciones para crear nuevos proveedores, editar información existente y eliminar registros con confirmación JavaScript.

- **Vistas/editar_usuario.php**: Interfaz para actualizar información de un usuario.

```php
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Editar Usuario</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light p-4">
  <div class="container">
    <h2 class="text-center mb-4">Editar Usuario</h2>
    <form method="post">
      <input type="hidden" name="idusuario" value="<?= $datos['idusuario'] ?>">
      <div class="mb-3">
        <label class="form-label">Usuario</label>
        <input type="text" name="nomusuario" class="form-control" value="<?= $datos['nomusuario'] ?>" required>
      </div>
      <div class="mb-3">
        <label class="form-label">Apellidos</label>
        <input type="text" name="apellidos" class="form-control" value="<?= $datos['apellidos'] ?>" required>
      </div>
      <div class="mb-3">
        <label class="form-label">Nombres</label>
        <input type="text" name="nombres" class="form-control" value="<?= $datos['nombres'] ?>" required>
      </div>
      <div class="mb-3">
        <label class="form-label">Email</label>
        <input type="email" name="email" class="form-control" value="<?= $datos['email'] ?>">
      </div>
      <div class="mb-3">
        <label class="form-label">Estado</label>
        <select name="estado" class="form-select">
          <option value="activo" <?= $datos['estado'] === 'activo' ? 'selected' : '' ?>>Activo</option>
          <option value="inactivo" <?= $datos['estado'] === 'inactivo' ? 'selected' : '' ?>>Inactivo</option>
        </select>
      </div>
      <button type="submit" class="btn btn-success">Guardar Cambios</button>
      <a href="listar_usuarios.php" class="btn btn-secondary">Cancelar</a>
    </form>
```

ESPECIFICACIÓN

Es un formulario UPDATE para usuarios del sistema que permite modificar datos personales (usuario, nombres, apellidos, email) y cambiar el estado (activo/inactivo). Los campos vienen precargados desde el array PHP $datos y incluye validación HTML5 con campos requeridos.

- **Vistas/listar_usuario.php**: Muestra una lista de todos los usuarios registrados.

```php
    <table class="table table-bordered table-hover table-striped">
      <thead class="table-dark">
        <tr>
          <th>ID</th>
          <th>Usuario</th>
          <th>Apellidos</th>
          <th>Nombres</th>
          <th>Email</th>
          <th>Estado</th>
          <th>Acciones</th>
        </tr>
      </thead>
      <tbody>
        <?php foreach ($usuarios as $u): ?>
        <tr>
          <td><?= $u['idusuario'] ?></td>
          <td><?= $u['nomusuario'] ?></td>
          <td><?= $u['apellidos'] ?></td>
          <td><?= $u['nombres'] ?></td>
          <td><?= $u['email'] ?></td>
          <td><?= $u['estado'] ?></td>
          <td>
            <a href="editar_usuario.php?id=<?= $u['idusuario'] ?>" class="btn btn-sm btn-warning">Editar</
            <a href="../controladores/eliminar_usuario.php?id=<?= $u['idusuario'] ?>" class="btn btn-sm b
          </td>
        </tr>
        <?php endforeach ?>
      </tbody>
    </table>
  </div>
</body>
</html>
```

ESPECIFICACIÓN

Líneas 1-2: Tabla Bootstrap con clases (table table-bordered table-hover table-striped) y thead con clase table-dark para fondo oscuro.

Líneas 3-10: Encabezados de la tabla:

- "ID" - Identificador del usuario
- "Usuario" - Nombre de usuario para login
- "Apellidos" - Apellidos del usuario
- "Nombres" - Nombres del usuario
- "Email" - Correo electrónico
- "Estado" - Estado del usuario (activo/inactivo)
- "Acciones" - Botones de operaciones CRUD

- **Login.html**: Página HTML para la interfaz de inicio de sesión de usuarios.

```html
<div class="login-container">
  <h2>Ventana de Inicio de Sesión</h2>
  <form id="loginForm" action="../controladores/login.php" method="post">
    <div class="form-group">
      <label for="nomusuario">Usuario</label>
      <input type="text" id="nomusuario" name="nomusuario" placeholder="Ingrese su usuario" required>
    </div>
    <div class="form-group">
      <label for="password">Contraseña</label>
      <input type="password" id="password" name="password" placeholder="Ingrese su contraseña" required>
    </div>
    <div class="form-actions">
      <button type="submit" id="submitBtn" disabled>Confirmar</button>
    </div>
  </form>

  <!-- enlace para registrarse -->
  <a href="registro.html" class="registro-link">¿No tienes una cuenta? Regístrate aquí</a>
</div>

<script>
  const usuario = document.getElementById('nomusuario');
  const contrasena = document.getElementById('password');
  const boton = document.getElementById('submitBtn');
```

ESPECIFICACIÓN

Es un formulario de autenticación completo que envía credenciales al controlador de login, incluye validación HTML5 con campos requeridos, control JavaScript del botón submit y enlace para registro de nuevos usuarios. El botón está inicialmente deshabilitado hasta que se completen los campos.

- **Registro.html**: Página HTML para el registro de usuarios.

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Registro de Usuario</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light d-flex justify-content-center align-items-center" style="height: 100vh;">

  <div class="card shadow p-4" style="width: 450px;">
    <h4 class="text-center text-primary mb-4">Registrar Usuario</h4>
    <form action="../controladores/registrar.php" method="post">
      <div class="mb-3">
        <label for="nomusuario" class="form-label">Nombre de Usuario</label>
        <input type="text" class="form-control" id="nomusuario" name="nomusuario" required>
      </div>
      <div class="mb-3">
        <label for="password" class="form-label">Contraseña</label>
        <input type="password" class="form-control" id="password" name="password" required>
      </div>
      <div class="mb-3">
        <label for="apellidos" class="form-label">Apellidos</label>
        <input type="text" class="form-control" id="apellidos" name="apellidos" required>
      </div>
      <div class="mb-3">
        <label for="nombres" class="form-label">Nombres</label>
        <input type="text" class="form-control" id="nombres" name="nombres" required>
      </div>
      <div class="mb-3">
        <label for="email" class="form-label">Correo Electrónico</label>
        <input type="email" class="form-control" id="email" name="email">
      </div>
      <button type="submit" class="btn btn-success w-100">Registrar</button>
    </form>
    <div class="text-center mt-3">
      <a href="login.html">¿Ya tienes cuenta? Inicia sesión</a>
    </div>
```

ESPECIFICACIÓN

Es un formulario de registro completo que recopila datos del usuario (nombre de usuario, contraseña, apellidos, nombres, email) con validación HTML5, diseño responsive centrado con Bootstrap y navegación hacia el login. Los datos se envían al controlador para crear nuevos usuarios en el sistema.

- **registro_categoria.php**: Interfaz para agregar una nueva categoría de producto.

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Registrar Categoría</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light">

<div class="container mt-5">
  <h3 class="mb-4">Registrar Nueva Categoría</h3>
  <form action="../controladores/registrar_categoria.php" method="post" class="shadow p-4 bg-white rounded">
    <div class="mb-3">
      <label for="nomcategoria" class="form-label">Nombre de la Categoría</label>
      <input type="text" class="form-control" name="nomcategoria" required>
    </div>
    <button type="submit" class="btn btn-success">Registrar</button>
    <a href="listar_categorias.php" class="btn btn-secondary">Volver</a>
  </form>
</div>

</body>
</html>
```

ESPECIFICACIÓN

Líneas 1-7: Configuración básica HTML5 con título "Registrar Categoría" y Bootstrap CSS.}

Líneas 8-12: Contenedor Bootstrap con título "Registrar Nueva Categoría" y formulario con:

- Action hacia ../controladores/registrar_categoria.php
- Método POST
- Estilos Bootstrap (shadow p-4 bg-white rounded)

Formulario CREATE simple para agregar nuevas categorías al sistema. Captura solo el nombre de la categoría, incluye validación HTML5 y navegación de retorno al listado.

- **registro_cliente.php**: Interfaz para registrar un nuevo cliente.

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Registrar Cliente</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light p-4">
  <div class="container">
    <h2 class="text-center mb-4">Registrar Cliente</h2>
    <form action="../controladores/registrar_cliente.php" method="post">

      <div class="mb-3"><label class="form-label">Nombre</label><input type="text" name="nomcliente" class="form-control" requi
      <div class="mb-3"><label class="form-label">RUC</label><input type="text" name="ruccliente" class="form-control"></div>
      <div class="mb-3"><label class="form-label">Dirección</label><input type="text" name="dircliente" class="form-control"></
      <div class="mb-3"><label class="form-label">Teléfono</label><input type="text" name="telcliente" class="form-control"></d
      <div class="mb-3"><label class="form-label">Email</label><input type="email" name="emailcliente" class="form-control"></d
      <button type="submit" class="btn btn-success">Registrar</button>
      <a href="listar_clientes.php" class="btn btn-secondary">Volver</a>
    </form>
  </div>
</body>
</html>
```

ESPECIFICACIÓN

Líneas 1-8: Configuración básica HTML5 con título "Registrar Cliente" y Bootstrap CSS.

Línea 9: Body con clases bg-light p-4 para fondo claro y padding.

Líneas 10-12: Contenedor Bootstrap con título "Registrar Cliente" y formulario con:

- Action hacia ../controladores/registrar_cliente.php
- Método POST

- **registro_condicion.php**: Interfaz para agregar una nueva condición de venta.

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Registrar Condición de Venta</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light p-4">
<div class="container">
  <h3 class="mb-4">Registrar Condición de Venta</h3>
  <form action="../controladores/registrar_condicion.php" method="post" class="bg-white p-4 rounded shadow">
    <div class="mb-3">
      <label for="nomcondicion" class="form-label">Nombre de la Condición</label>
      <input type="text" class="form-control" id="nomcondicion" name="nomcondicion" required>
    </div>
    <button type="submit" class="btn btn-primary">Guardar</button>
    <a href="listar_condicion.php" class="btn btn-secondary">Volver</a>
  </form>
</div>
</body>
</html>
```

ESPECIFICACIÓN

Formulario HTML simple para registrar condiciones de venta. Tiene un campo de texto obligatorio para el nombre, botón "Guardar" que envía los datos a PHP, y botón "Volver". Usa Bootstrap para el diseño.

- **registro_factura.php**: Interfaz para crear una nueva factura.

```php
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Registrar Factura</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light p-4">
<div class="container">
  <h3>Registrar Nueva Factura</h3>
  <form action="../controladores/registrar_factura.php" method="post" class="bg-white p-4 shadow rounded">
    <!-- campos principales -->
    <div class="row">
      <div class="col-md-4 mb-3">
        <label class="form-label">Fecha</label><input type="date" name="fecha" class="form-control" required>
      </div>
      <div class="col-md-4 mb-3">
        <label>Cliente</label>
        <select name="idcliente" class="form-select" required>
          <?php foreach ($clientes as $c): ?>
            <option value="<?= $c['idcliente'] ?>"><?= $c['nomcliente'] ?></option>
          <?php endforeach ?>
        </select>
      </div>
      <div class="col-md-4 mb-3">
        <label>Condición</label>
        <select name="idcondicion" class="form-select" required>
          <?php foreach ($conditions as $co): ?>
            <option value="<?= $co['idcondicion'] ?>"><?= $co['nomcondicion'] ?></option>
          <?php endforeach ?>
        </select>
      </div>
    </div>
```

ESPECIFICACIÓN

Formulario web para crear facturas con tres campos obligatorios: fecha (selector de calendario), cliente (dropdown con lista de clientes) y condición de venta (dropdown con condiciones). Usa PHP para cargar dinámicamente las opciones de clientes y condiciones desde la base de datos. Diseñado con Bootstrap en layout de columnas responsivas.

- **registro_producto.php**: Interfaz para agregar un nuevo producto.

```php
</head>
<body class="bg-light">
<div class="container mt-5">
  <h3 class="mb-4">Registrar Producto</h3>
  <form action="../controladores/registrar_producto.php" method="post" class="p-4 bg-white shadow rounded">
    <!-- campos -->
    <div class="mb-3"><label class="form-label">Nombre</label><input class="form-control" name="nomproducto" required></div>
    <div class="mb-3"><label>Unidad</label><input class="form-control" name="unimed"></div>
    <div class="mb-3"><label>Stock</label><input type="number" class="form-control" name="stock" value="0"></div>
    <div class="mb-3"><label>Costo unitario</label><input type="number" class="form-control" name="cosuni" step="0.0001"></div>
    <div class="mb-3"><label>Precio unitario</label><input type="number" class="form-control" name="preuni" step="0.0001"></div>
    <div class="mb-3"><label>Categoría</label>
      <select class="form-select" name="idcategoria">
        <?php foreach($cats as $c): ?>
          <option value="<?= $c['idcategoria'] ?>"><?= $c['nomcategoria'] ?></option>
        <?php endforeach ?>
      </select>
    </div>
    <div class="mb-3"><label>Proveedor</label>
      <select class="form-select" name="idproveedor">
        <?php foreach($provs as $p): ?>
          <option value="<?= $p['idproveedor'] ?>"><?= $p['nomproveedor'] ?></option>
        <?php endforeach ?>
      </select>
    </div>
    <button class="btn btn-success">Registrar</button>
    <a href="listar_productos.php" class="btn btn-secondary">Volver</a>
  </form>
</div>
</body>
</html>
```

ESPECIFICACIÓN

Formulario completo para registrar productos con 6 campos: nombre, stock, costo unitario, precio unitario, categoría y proveedor. Los campos numéricos permiten decimales, y las listas de categorías y proveedores se cargan dinámicamente desde PHP. Incluye validación obligatoria en todos los campos y diseño Bootstrap responsivo

- **registro_proveedor.php**: Interfaz para registrar un nuevo proveedor.

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Registrar Proveedor</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light p-4">
  <div class="container">
    <h2 class="mb-4 text-center">Registrar Proveedor</h2>
    <form action="../controladores/registrar_proveedor.php" method="post">

      <div class="mb-3"><label>Nombre</label><input type="text" name="nombre" class="form-control" required></div>
      <div class="mb-3"><label>RUC</label><input type="text" name="ruc" class="form-control"></div>
      <div class="mb-3"><label>Dirección</label><input type="text" name="direccion" class="form-control"></div>
      <div class="mb-3"><label>Teléfono</label><input type="text" name="telefono" class="form-control"></div>
      <div class="mb-3"><label>Email</label><input type="email" name="email" class="form-control"></div>
      <button type="submit" class="btn btn-success">Registrar</button>
      <a href="listar_proveedores.php" class="btn btn-secondary">Volver</a>
    </form>
  </div>
</body>
</html>
```

ESPECIFICACIÓN

Formulario completo para registrar proveedores con 5 campos obligatorios: nombre, RUC, dirección, teléfono y email. Incluye validación automática del email, diseño Bootstrap responsivo y navegación de retorno al listado de proveedores.

- **Logout.php**: Maneja el cierre de sesión de usuarios y la terminación de la sesión.

```php
<?php
session_start();
session_destroy();
header("Location: vistas/login.html");
```

ESPECIFICACIÓN

Este script se ejecuta cuando un usuario quiere cerrar sesión en una aplicación web. La secuencia es:

1. Se inicia la sesión (para poder acceder a ella)
2. Se destruye completamente la sesión (logout)
3. Se redirige al usuario de vuelta a la página de login

Es un patrón común en sistemas de autenticación web para garantizar que el usuario quede completamente desconectado del sistema.

Pruebas

Describe la estructura de pruebas para verificar el funcionamiento de los diferentes módulos de la aplicación.

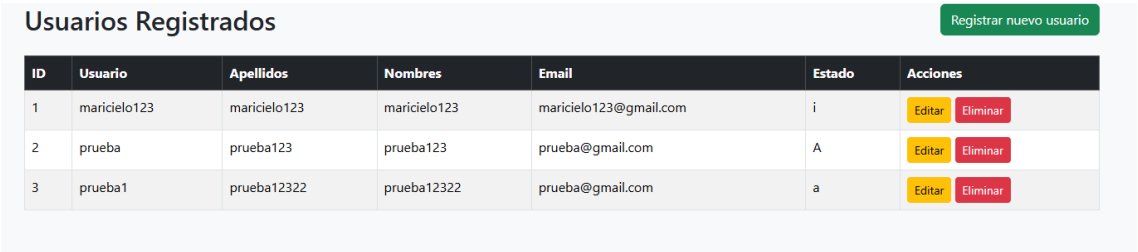
- **LOGIN**: Prueba la funcionalidad de inicio de sesión para garantizar una autenticación segura.

- **Bienvenido.php**: Prueba la visualización de la página de bienvenida para usuarios autenticados.

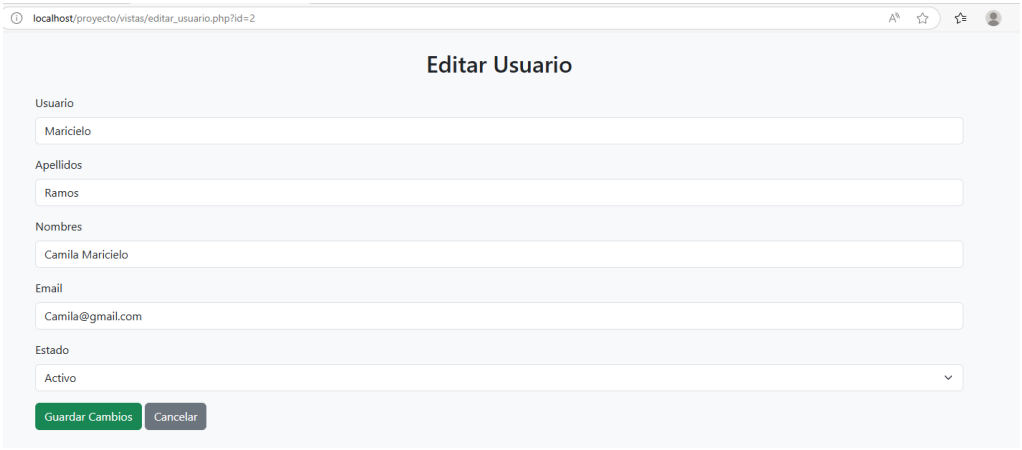
- **Usuarios**:

Prueba las funcionalidades relacionadas con usuarios, como creación, edición y eliminación.

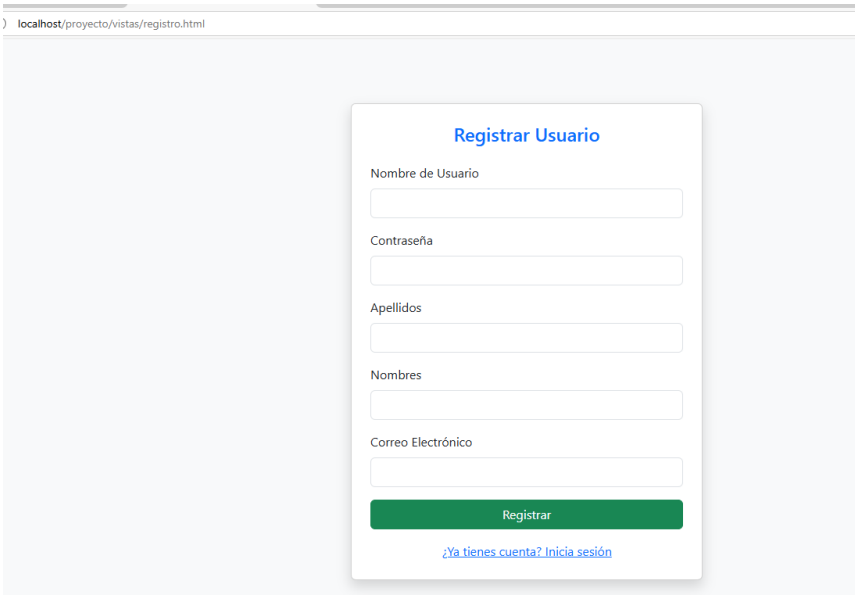
  - **Lista De Usuarios**: Prueba la visualización de la lista de usuarios.



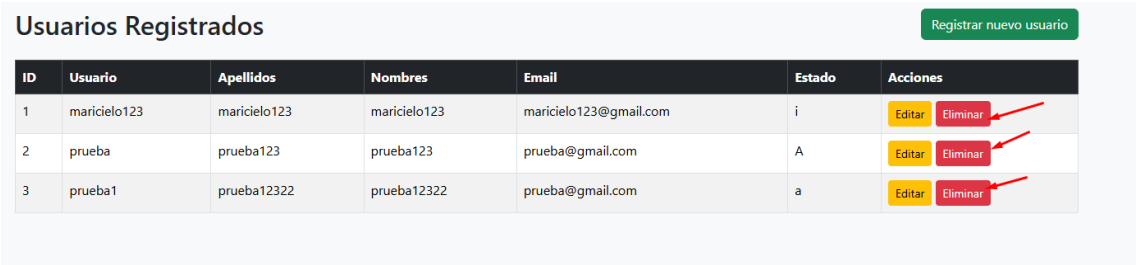
  - **Editar Usuario**: Prueba la funcionalidad de edición de usuarios.



  - **Registrar Usuario**: Prueba el proceso de registro de usuarios.



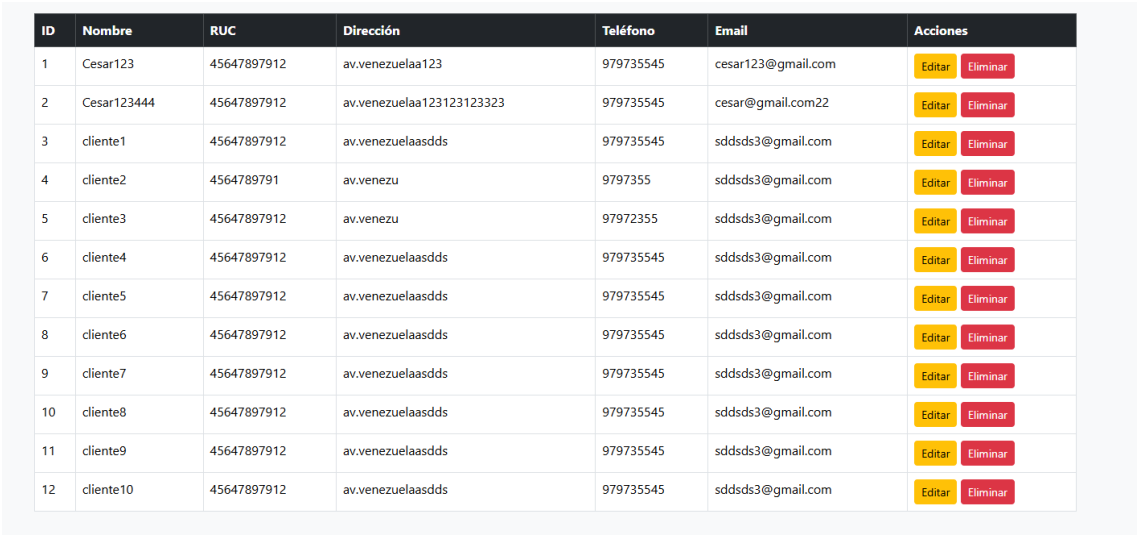
- **Eliminar Usuario**: Prueba el proceso de eliminación de usuarios.



- **Clientes**

  Prueba las funcionalidades relacionadas con clientes.

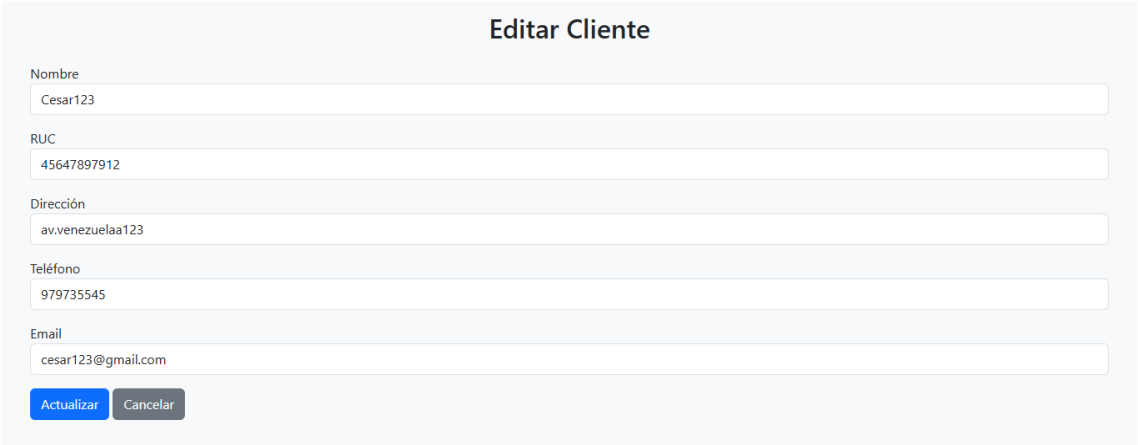
  - **Lista de clientes**: Prueba la visualización de la lista de clientes.



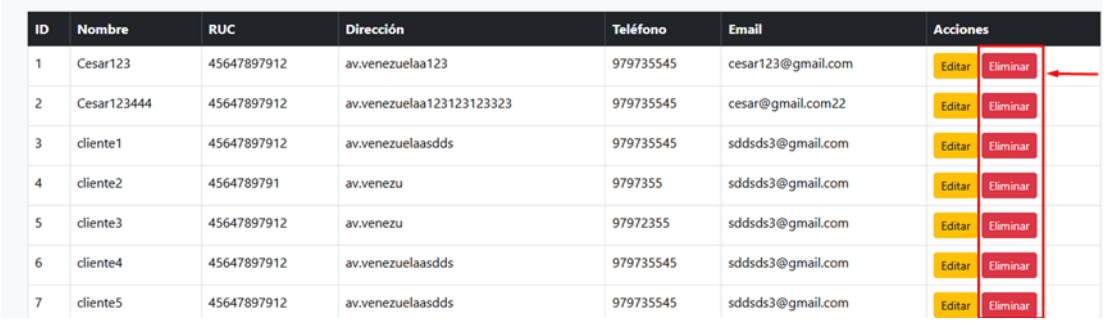
ID	Nombre	RUC	Dirección	Teléfono	Email	Acciones
1	Cesar123	45647897912	av.venezuelaa123	979735545	cesar123@gmail.com	Editar Eliminar
2	Cesar123444	45647897912	av.venezuelaa123123123323	979735545	cesar@gmail.com22	Editar Eliminar
3	cliente1	45647897912	av.venezuelaasdds	979735545	sddsds3@gmail.com	Editar Eliminar
4	cliente2	4564789791	av.venezu	9797355	sddsds3@gmail.com	Editar Eliminar
5	cliente3	45647897912	av.venezu	97972355	sddsds3@gmail.com	Editar Eliminar
6	cliente4	45647897912	av.venezuelaasdds	979735545	sddsds3@gmail.com	Editar Eliminar
7	cliente5	45647897912	av.venezuelaasdds	979735545	sddsds3@gmail.com	Editar Eliminar
8	cliente6	45647897912	av.venezuelaasdds	979735545	sddsds3@gmail.com	Editar Eliminar
9	cliente7	45647897912	av.venezuelaasdds	979735545	sddsds3@gmail.com	Editar Eliminar
10	cliente8	45647897912	av.venezuelaasdds	979735545	sddsds3@gmail.com	Editar Eliminar
11	cliente9	45647897912	av.venezuelaasdds	979735545	sddsds3@gmail.com	Editar Eliminar
12	cliente10	45647897912	av.venezuelaasdds	979735545	sddsds3@gmail.com	Editar Eliminar

  - **Registro de clientes**: Prueba el proceso de registro de clientes.

- **Editar cliente**: Prueba la funcionalidad de edición de clientes.



- **Eliminar clientes**: Prueba el proceso de eliminación de clientes.

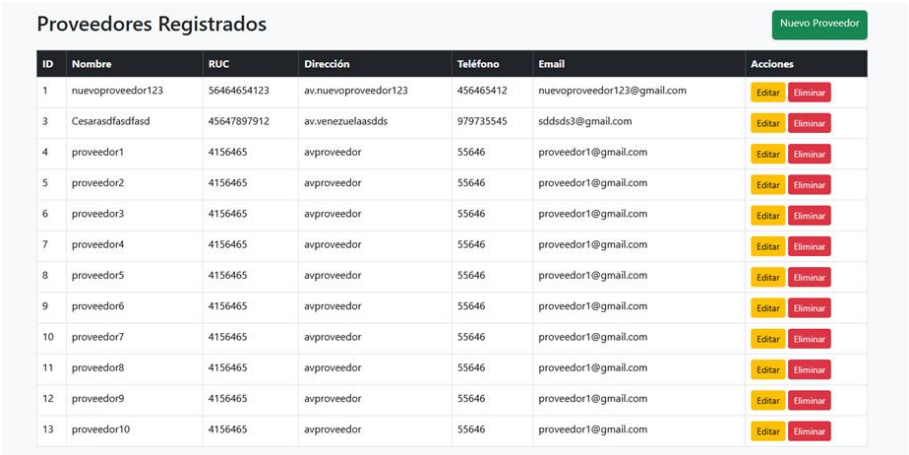


ID	Nombre	RUC	Dirección	Teléfono	Email	Acciones
1	Cesar123	45647897912	av.venezuelaa123	979735545	cesar123@gmail.com	Editar Eliminar
2	Cesar123444	45647897912	av.venezuelaa123123123323	979735545	cesar@gmail.com22	Editar Eliminar
3	cliente1	45647897912	av.venezuelaasdds	979735545	sddsds3@gmail.com	Editar Eliminar
4	cliente2	4564789791	av.venezu	9797355	sddsds3@gmail.com	Editar Eliminar
5	cliente3	45647897912	av.venezu	97972355	sddsds3@gmail.com	Editar Eliminar
6	cliente4	45647897912	av.venezuelaasdds	979735545	sddsds3@gmail.com	Editar Eliminar
7	cliente5	45647897912	av.venezuelaasdds	979735545	sddsds3@gmail.com	Editar Eliminar

- **Proveedores**

  Prueba las funcionalidades relacionadas con proveedores.

  - **Lista Proveedores**: Prueba la visualización de la lista de proveedores.



Proveedores Registrados — Nuevo Proveedor

ID	Nombre	RUC	Dirección	Teléfono	Email	Acciones
1	nuevoproveedor123	56464654123	av.nuevoproveedor123	456465412	nuevoproveedor123@gmail.com	Editar Eliminar
3	Cesarasdfasdfasd	45647897912	av.venezuelaasdds	979735545	sddsds3@gmail.com	Editar Eliminar
4	proveedor1	4156465	avproveedor	55646	proveedor1@gmail.com	Editar Eliminar
5	proveedor2	4156465	avproveedor	55646	proveedor1@gmail.com	Editar Eliminar
6	proveedor3	4156465	avproveedor	55646	proveedor1@gmail.com	Editar Eliminar
7	proveedor4	4156465	avproveedor	55646	proveedor1@gmail.com	Editar Eliminar
8	proveedor5	4156465	avproveedor	55646	proveedor1@gmail.com	Editar Eliminar
9	proveedor6	4156465	avproveedor	55646	proveedor1@gmail.com	Editar Eliminar
10	proveedor7	4156465	avproveedor	55646	proveedor1@gmail.com	Editar Eliminar
11	proveedor8	4156465	avproveedor	55646	proveedor1@gmail.com	Editar Eliminar
12	proveedor9	4156465	avproveedor	55646	proveedor1@gmail.com	Editar Eliminar
13	proveedor10	4156465	avproveedor	55646	proveedor1@gmail.com	Editar Eliminar

- **Registro Proveedores**: Prueba el proceso de registro de proveedores.

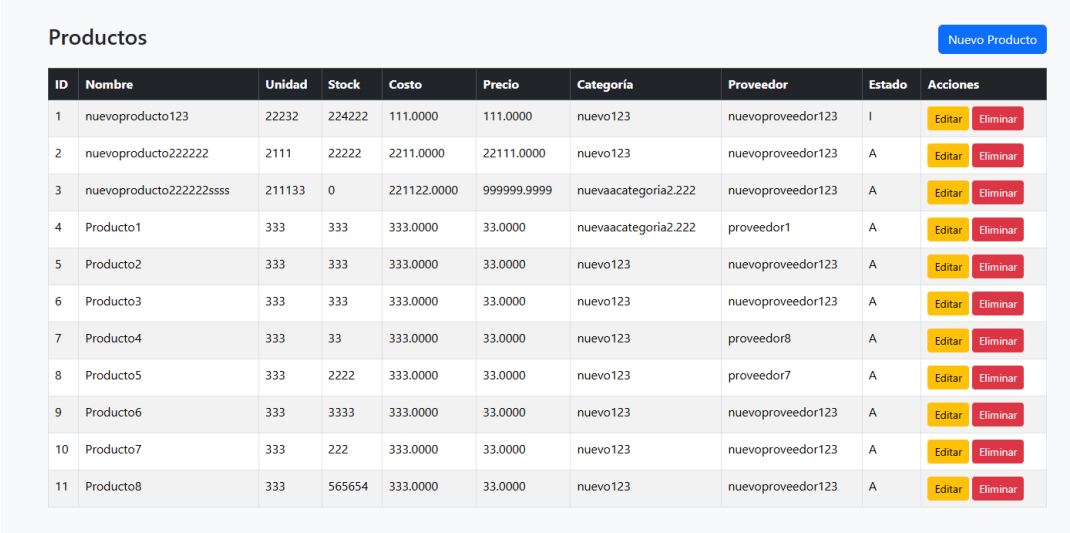
- **Editar Proveedor**: Prueba la funcionalidad de edición de proveedores.

- **Eliminar Proveedor**: Prueba el proceso de eliminación de proveedores.

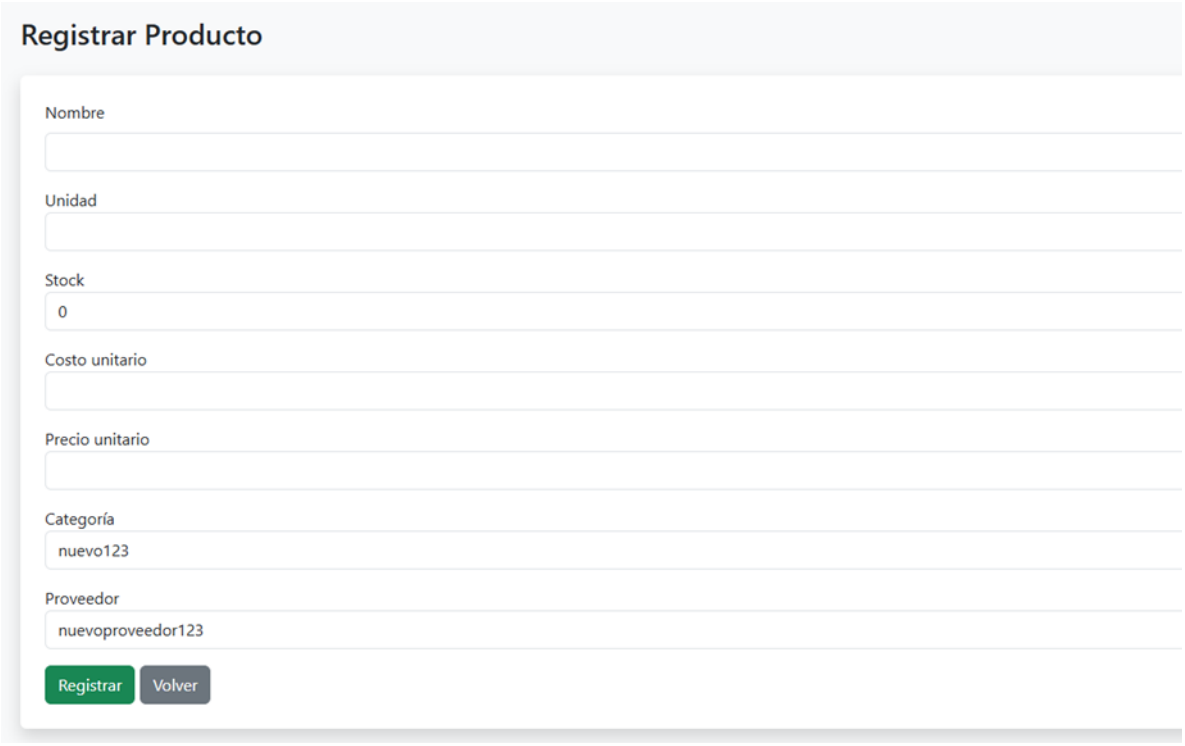
- **Productos**

  Prueba las funcionalidades relacionadas con productos.

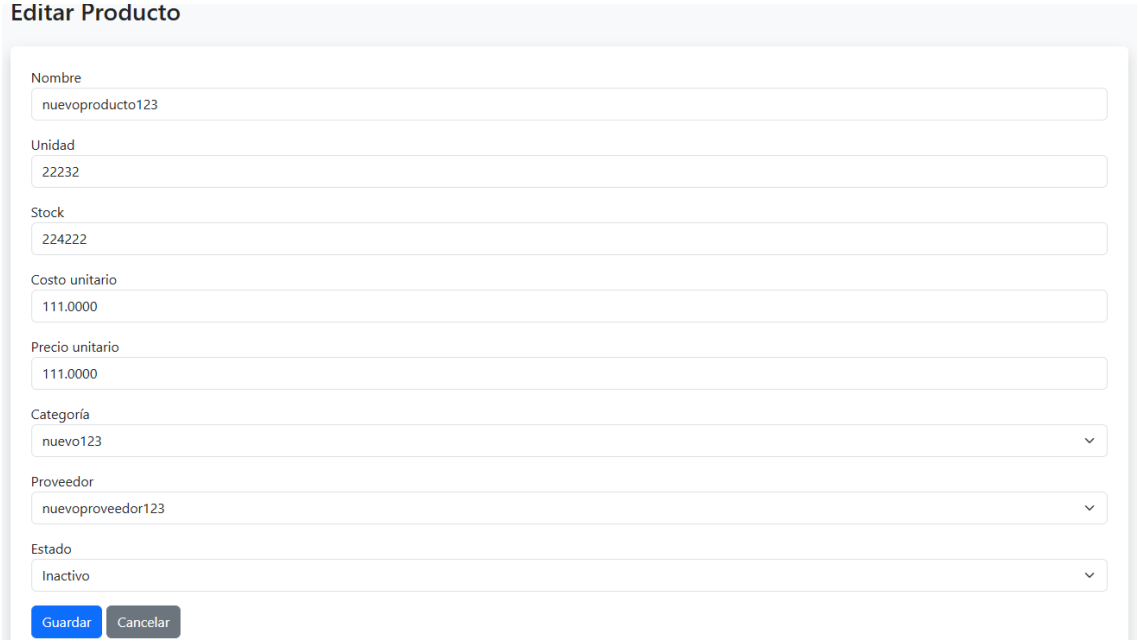
  - **Lista de Productos**: Prueba la visualización de la lista de productos.



  - **Registro de Productos**: Prueba el proceso de registro de productos.



- **Editar Productos**: Prueba la funcionalidad de edición de productos.



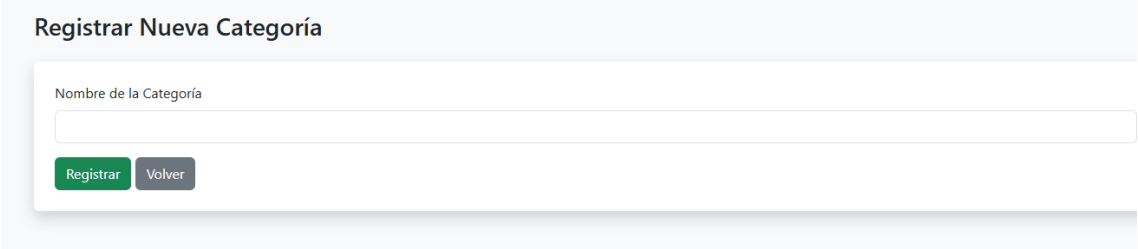
- **Eliminar Productos**: Prueba el proceso de eliminación de productos.

- **Categorias**

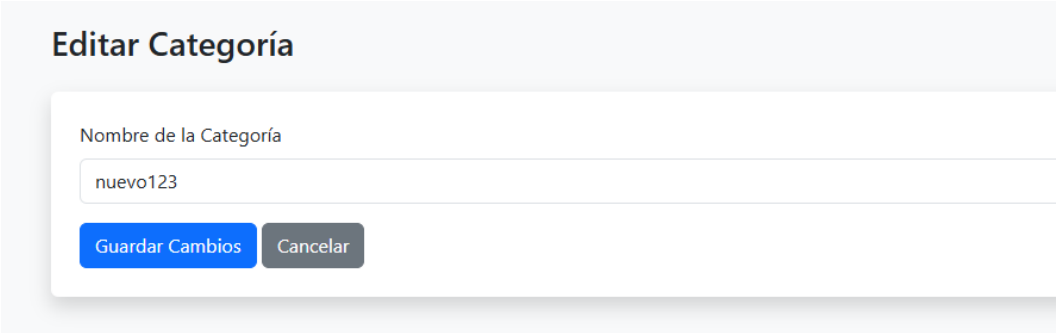
  Prueba las funcionalidades relacionadas con categorías.

  - **Lista de Categorias**: Prueba la visualización de la lista de categorías.

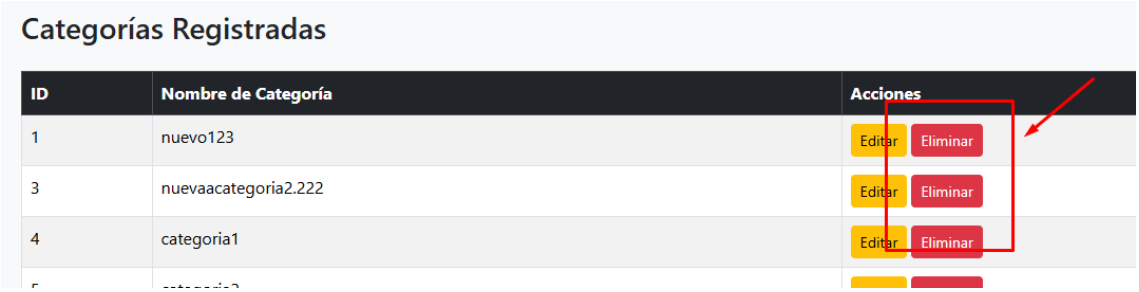
- **Registrar Categoria**: Prueba el proceso de registro de categorías.



- **Editar Categoria**: Prueba la funcionalidad de edición de categorías.



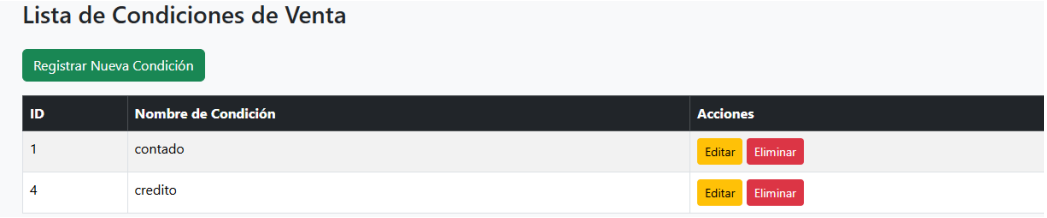
- **Eliminar Categoria**: Prueba el proceso de eliminación de categorías.



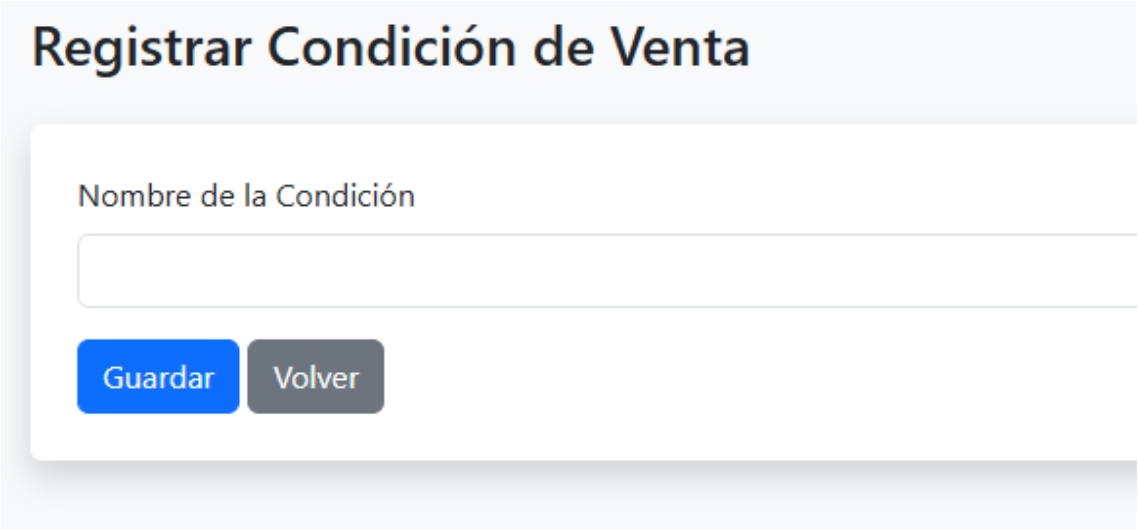
- **Condiciones de Venta**

  Prueba las funcionalidades relacionadas con condiciones de venta.

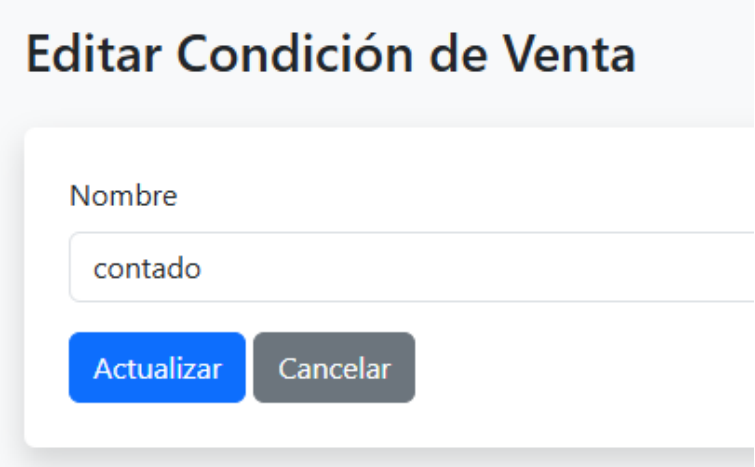
  - **Lista de Condiciones de Venta**: Prueba la visualización de la lista de condiciones de venta.



- **Registro de Condicion de Venta**: Prueba el proceso de registro de condiciones de venta.



- **Editar Condicion de Venta**: Prueba la funcionalidad de edición de condiciones de venta.



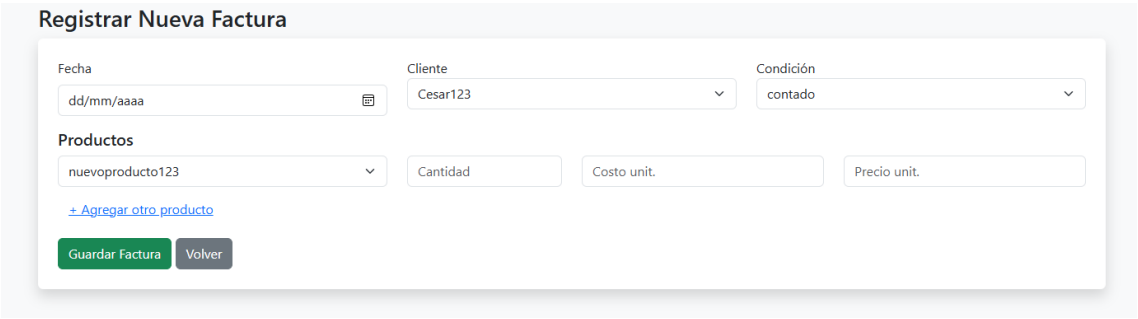
- **Borrar Condicion de Venta**: Prueba el proceso de eliminación de condiciones de venta.

- **Factura**

  Prueba las funcionalidades relacionadas con facturas.

  - **Lista de Facturas**: Prueba la visualización de la lista de facturas.

  - **Registro de Facturas**: Prueba el proceso de creación de facturas.



  - **Detalle de Factura**: Prueba la visualización de los detalles de una factura.