# UNIVERSIDAD BOLIVIANA DE INFORMÁTICA

## CARRERA DE INGENIERÍA DE SISTEMAS



# Proyecto de Sistema Web CRUD Comercial

ESTUDIANTE:  
Marco Rodolfo Cortez

ASIGNATURA:  
GES-703 Taller de Sistemas III

SEMESTRE:  
Séptimo – Turno Noche

DOCENTE:  
Ing. José Efraín Ovando Moscoso

LA PAZ – BOLIVIA  
2025

## 1. Introducción

El auge de las pequeñas y medianas empresas ha generado la necesidad de contar con herramientas tecnológicas que automaticen y mejoren la administración de sus procesos comerciales. Este trabajo se enfoca en el desarrollo de una aplicación web funcional utilizando el framework Laravel junto con la plantilla AdminLTE. La aplicación permite realizar operaciones CRUD sobre productos, clientes, categorías y compras.  
  
Un elemento clave de este sistema es la gestión de relaciones muchos a muchos entre compras y productos, con el uso de una tabla intermedia que almacena datos adicionales como la cantidad adquirida y el subtotal correspondiente. Este desafío técnico fue superado mediante el aprovechamiento de funcionalidades avanzadas proporcionadas por Laravel.

## 2. Objetivo General

Crear una solución web robusta que facilite la administración de la información comercial de una empresa, incluyendo productos, clientes, categorías y compras. Esto se logra mediante el uso de Laravel como framework de desarrollo y AdminLTE para una interfaz gráfica moderna y adaptable.

## 3. Justificación

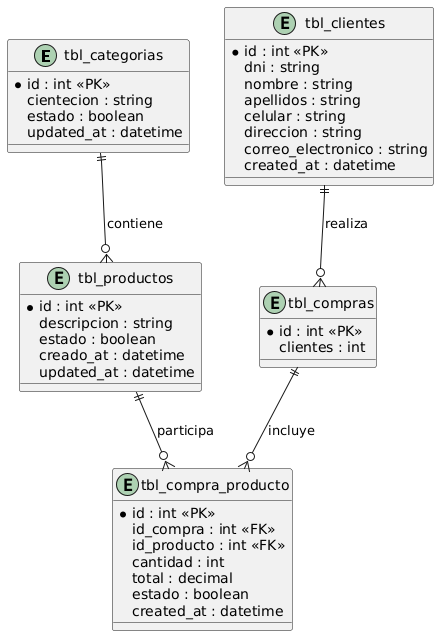
La transformación digital permite una mayor eficiencia en los procesos empresariales, minimizando errores humanos. En muchos casos, se continúan utilizando métodos rudimentarios como hojas de cálculo para manejar información comercial. Este sistema propone una alternativa profesional, confiable y escalable, ofreciendo una gestión precisa y sencilla a través de tecnologías actuales como Laravel y AdminLTE.  
  
La inclusión del CRUD para la tabla pivote permite simular escenarios reales de compra, almacenando datos como cantidad y precio de los productos comprados.

## 4. Análisis de la Base de Datos

La base de datos consta de cinco tablas principales:  
- tbl\_categorias: contiene las categorías para organizar los productos.  
- tbl\_productos: almacena información de productos vinculados a categorías.  
- tbl\_clientes: guarda los datos de los clientes.  
- tbl\_compras: registra las compras realizadas por los clientes.  
- tbl\_compra\_producto: actúa como tabla intermedia que registra los productos en cada compra, junto con su cantidad y total parcial.  
  
Relaciones principales:  
- Una categoría puede tener varios productos (1:N).  
- Un cliente puede realizar múltiples compras (1:N).  
- Las compras y productos tienen una relación de muchos a muchos (N:M) a través de la tabla intermedia.

## 5. Diseño de la Base de Datos

Se elaboró un diagrama entidad-relación que representa gráficamente la estructura de la base de datos, el cual fue implementado en Laravel usando migraciones, respetando las relaciones, claves foráneas y restricciones necesarias.



## 6. Tecnologías Utilizadas

- Laravel (v12): Framework PHP para desarrollar aplicaciones web seguras y escalables.  
- AdminLTE (v3): Plantilla basada en Bootstrap que proporciona una interfaz moderna.  
- PostgreSQL: Motor de base de datos relacional.  
- Blade: Motor de plantillas de Laravel.  
- Composer y Artisan: Herramientas CLI para gestión del proyecto.  
- XAMPP/Laragon: Servidores locales para pruebas.

## 7. Implementación de los Módulos CRUD

### 7.1 Categorías (tbl\_categorias)

Este módulo permite gestionar las categorías de los productos. Se pueden registrar, editar y eliminar las categorías, asegurando que el nombre sea único y obligatorio. Existe una relación de uno a muchos con los productos, lo que permite clasificar cada producto dentro de una categoría.

### 7.2 Productos (tbl\_productos)

Se habilita la creación, lectura, actualización y eliminación de productos. El sistema realiza validaciones sobre el nombre, el precio, el stock y la asignación a una categoría específica. Además, se controla el stock inicial desde el momento del registro del producto.

### 7.3 Clientes (tbl\_clientes)

Permite el registro de datos básicos del cliente como nombre, correo electrónico y teléfono. Se implementan funciones para listar, buscar, modificar o eliminar los datos de los clientes. Cada cliente puede estar vinculado a múltiples compras.

### 7.4 Compras (tbl\_compras)

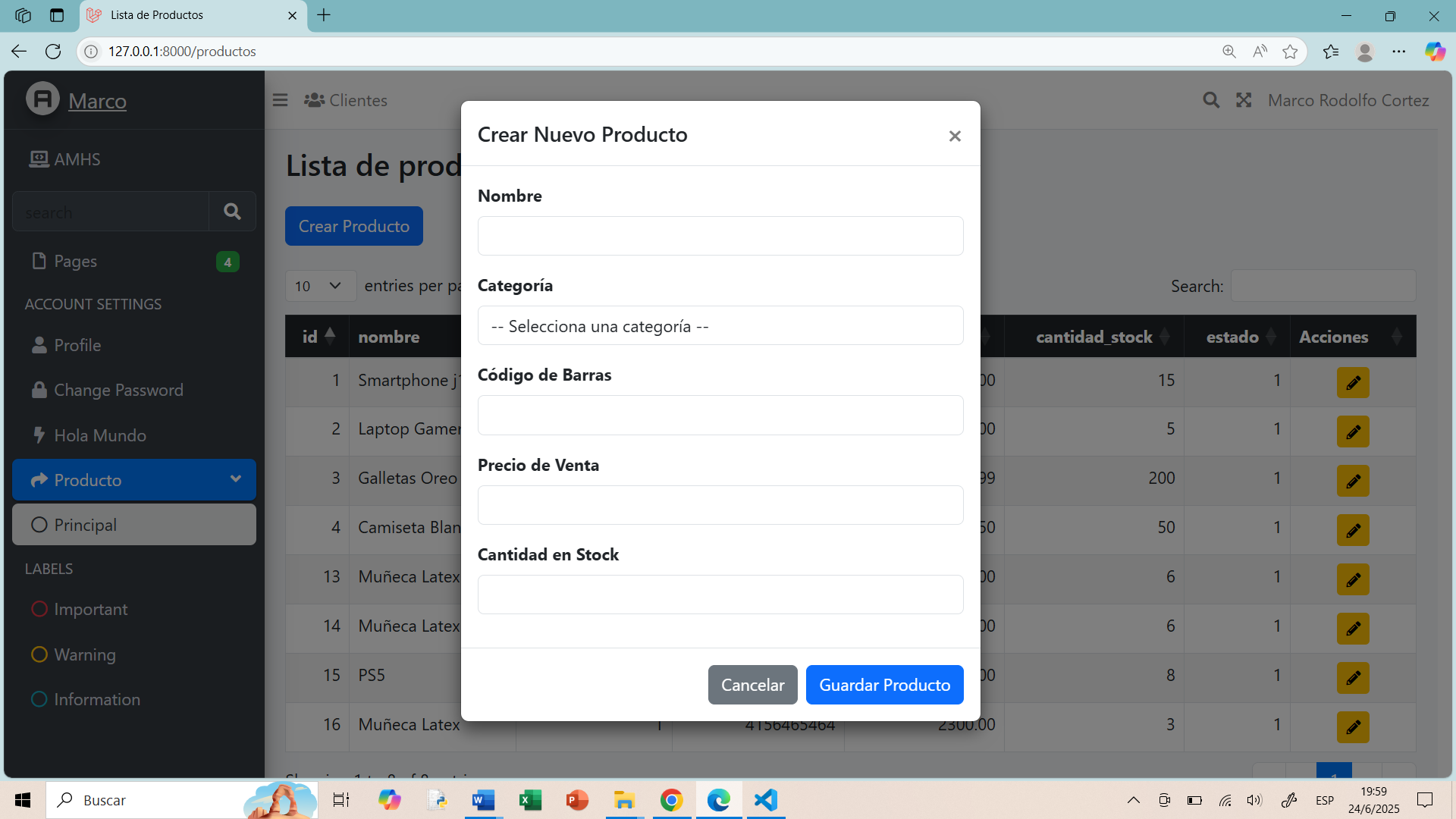
Este módulo permite registrar nuevas compras vinculadas a un cliente determinado. Se puede consultar un historial de compras por cliente. A través de la tabla intermedia, es posible asignar varios productos a una compra.

## 8. CRUD de la Tabla Intermedia tbl\_compra\_producto

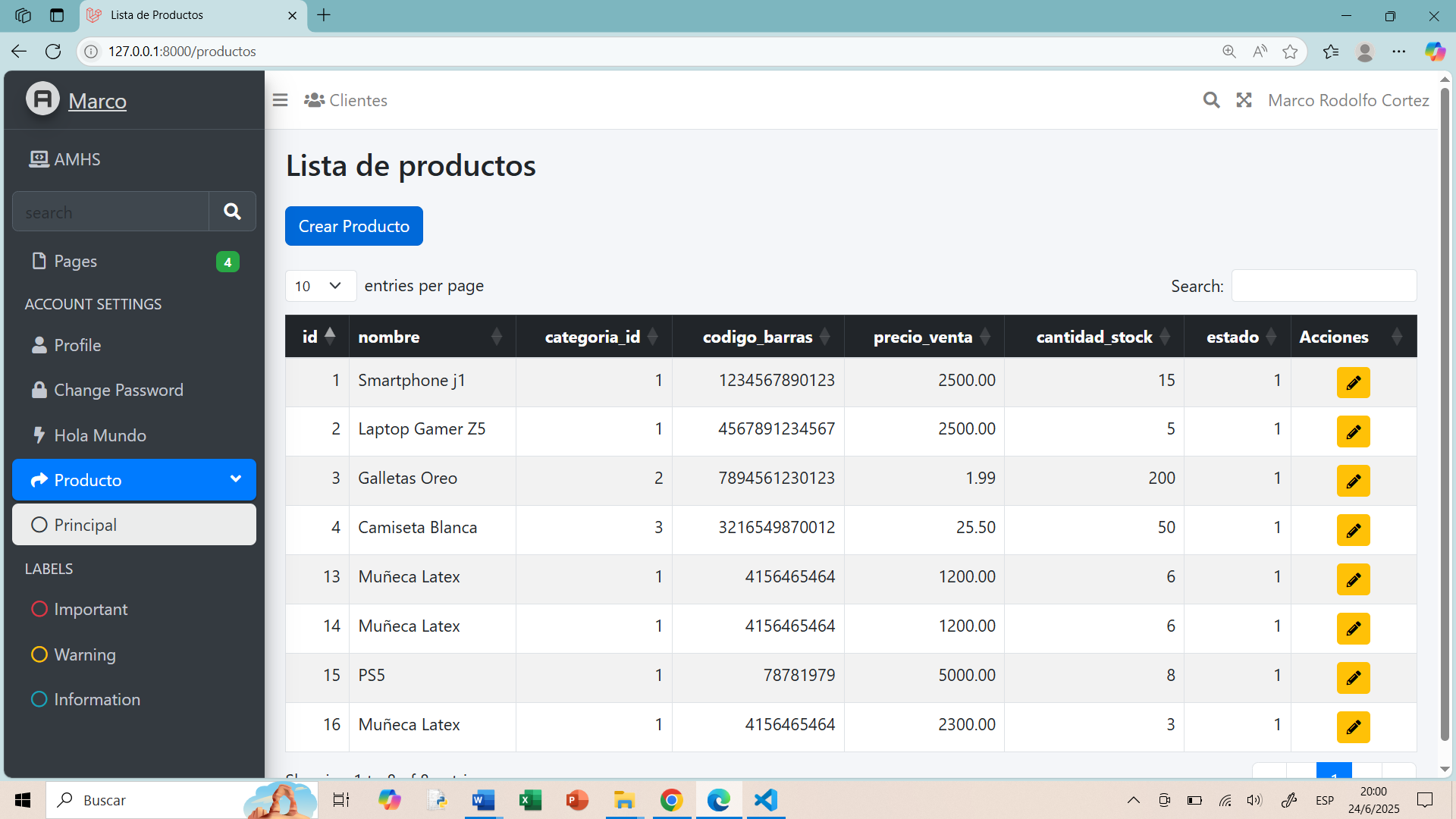
La tabla tbl\_compra\_producto es esencial ya que permite registrar qué productos se han incluido en cada compra, así como la cantidad adquirida y el total correspondiente por producto.  
  
Entre las funcionalidades implementadas destacan:  
- Selección de una compra existente.  
- Agregado dinámico de productos mediante formularios interactivos.  
- Registro de cantidad de unidades por producto.  
- Cálculo automático del total por ítem.  
- Validación en tiempo real del stock disponible.  
- Posibilidad de modificar productos dentro de la compra.  
- Eliminación específica de productos sin afectar la compra completa.  
  
Se utilizaron los métodos attach(), updateExistingPivot() y detach() de Laravel para la gestión adecuada de la relación muchos a muchos.

# 9. Capturas de Pantalla

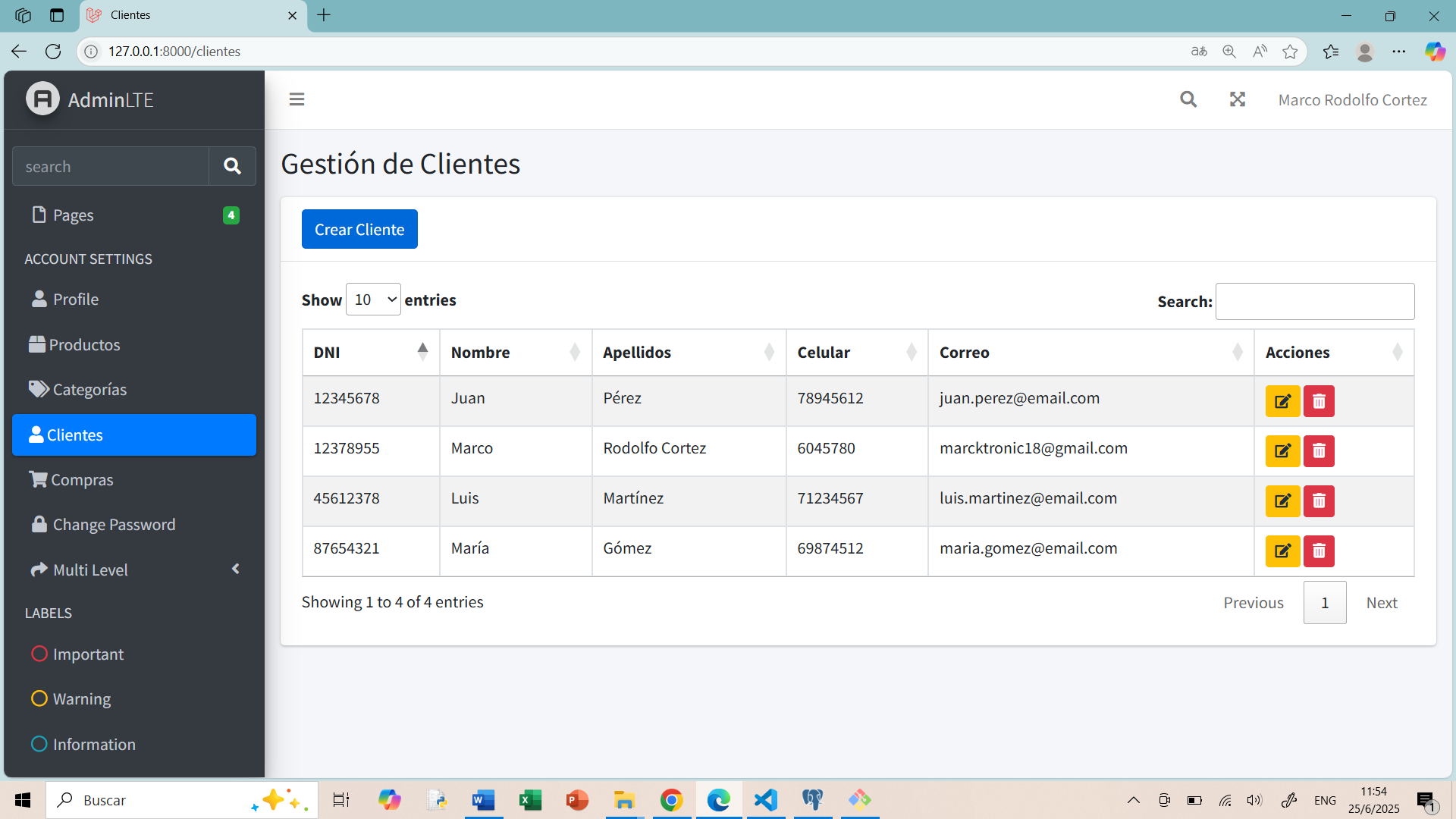
* Formulario de registro de producto.



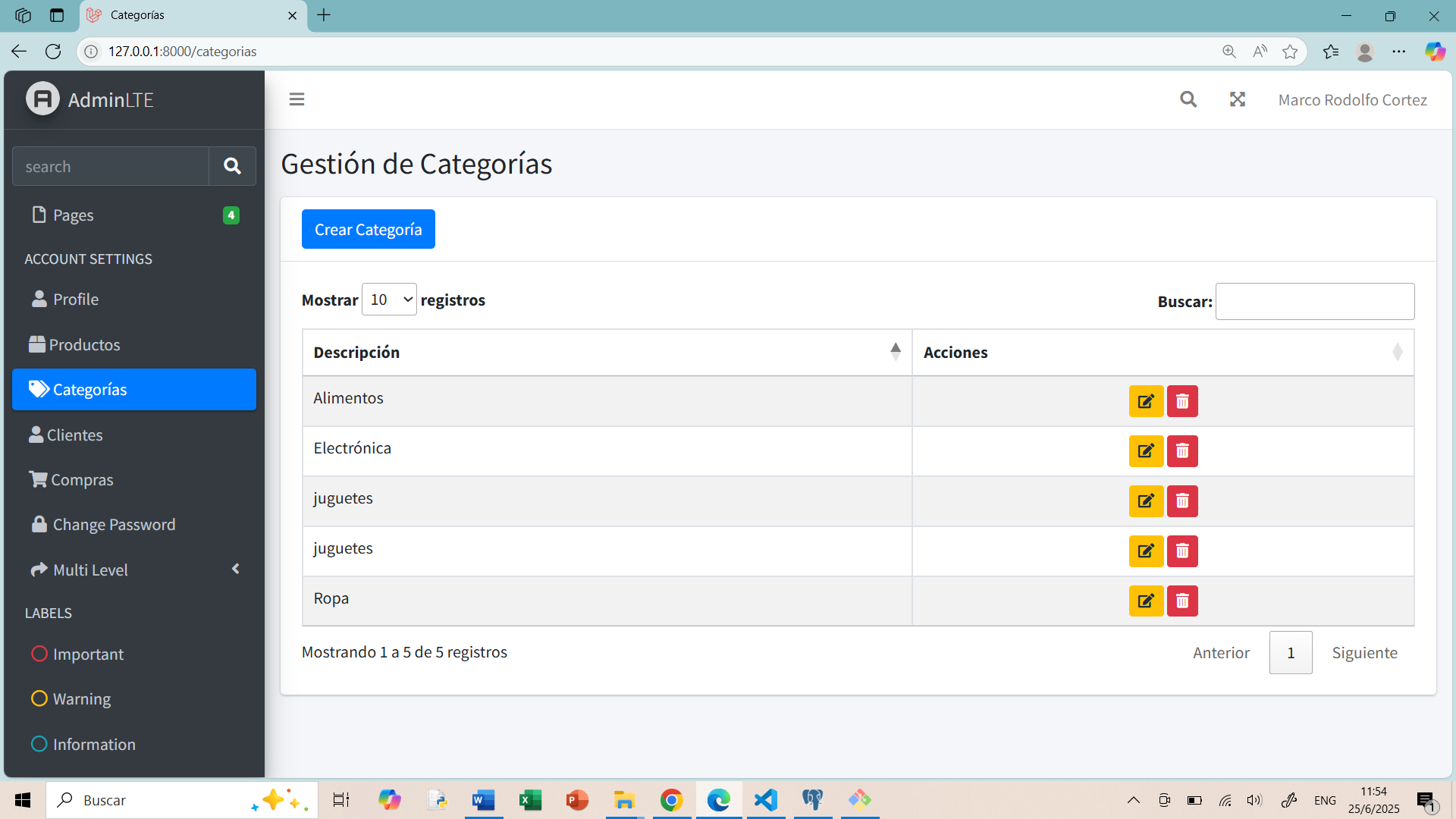
Productos:



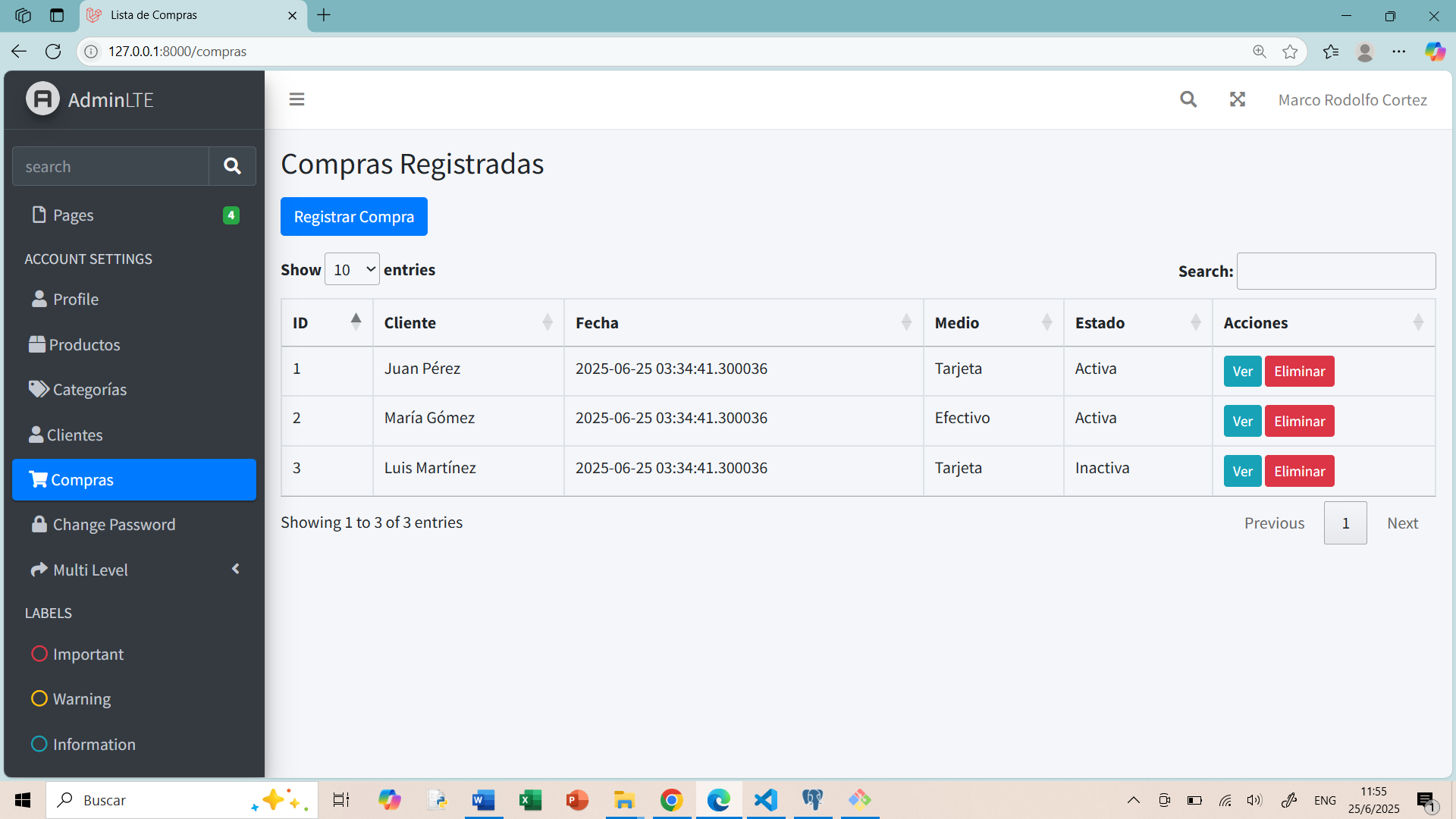
Clientes:



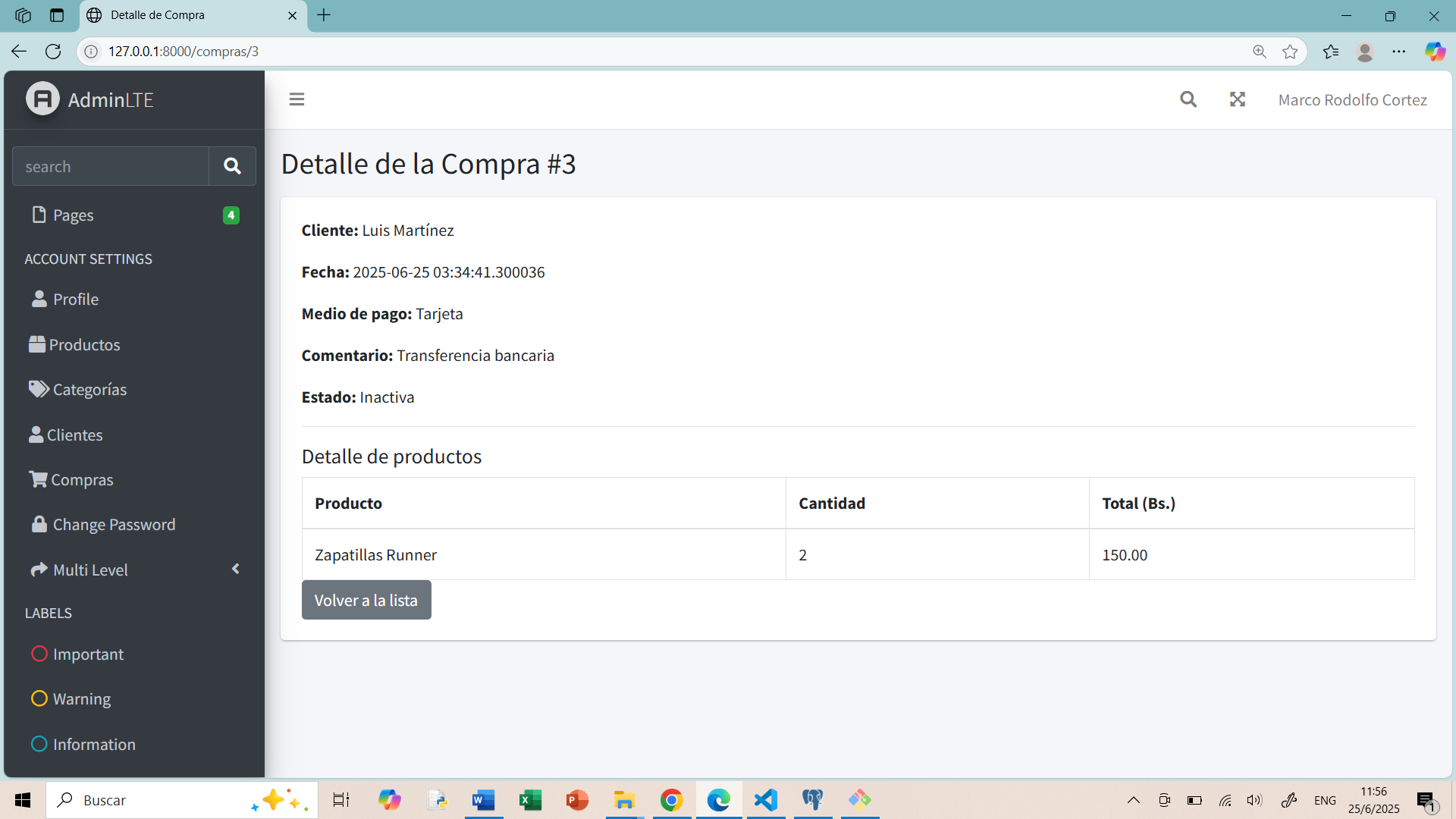
Categorías



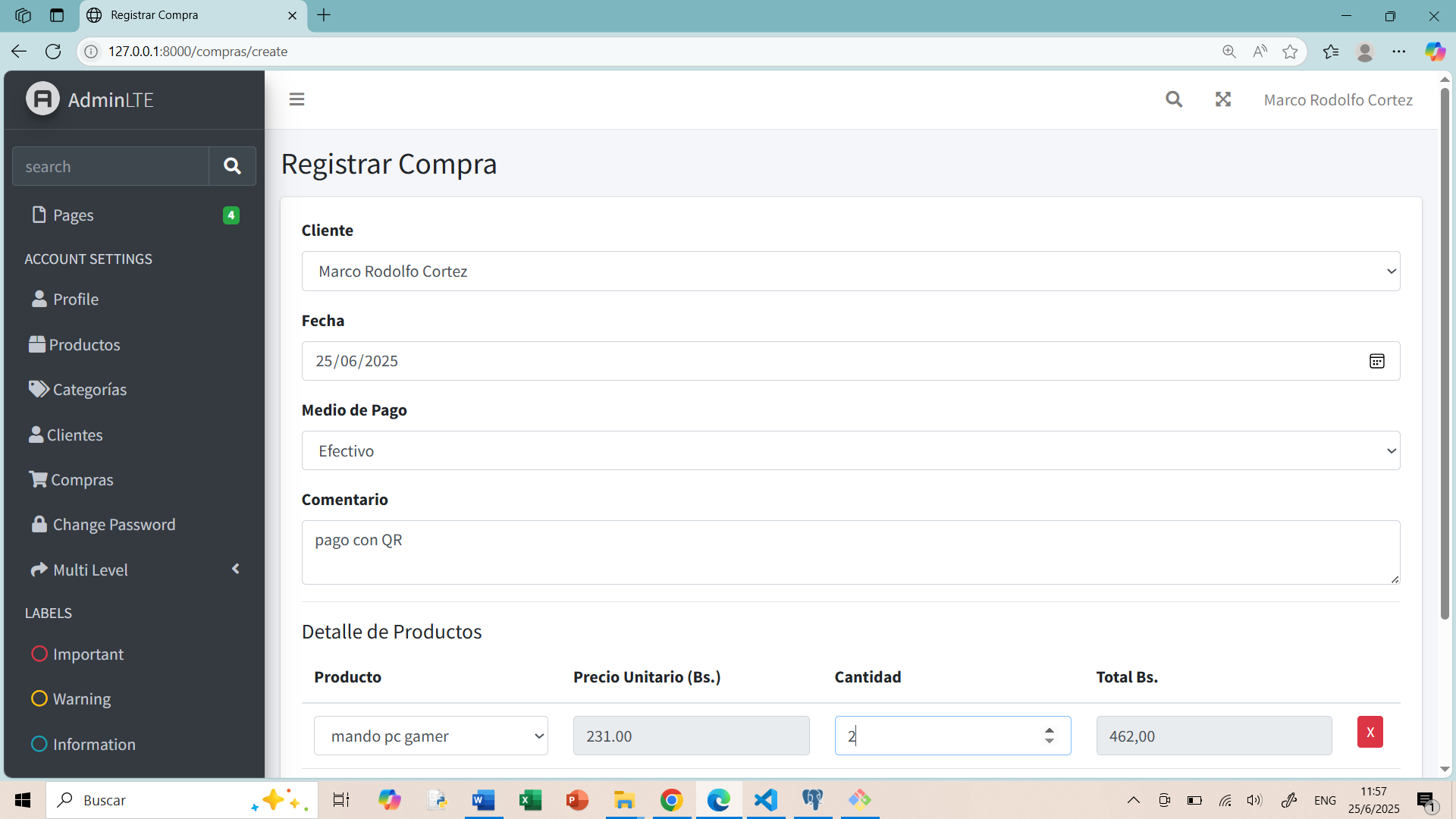
Compras:



Compras reazadas:



Crear compras:



## 10. Problemas Encontrados y Soluciones

1. Problema: Eliminación Lógica de Productos en lugar de Física.

\* Descripción: El método destroy en ProductoController solo cambiaba el estado de un producto a false (eliminación lógica), no lo eliminaba de la base de datos.

\* Solución Aplicada: Se modificó el método destroy en ProductoController para realizar una eliminación física del producto y, crucialmente, también eliminar los registros asociados en tbl\_compra\_producto utilizando DB::transaction para asegurar la integridad.

2. Problema: Error "Swal is not defined" y problemas con SweetAlert2 en el frontend.

\* Descripción: Al intentar usar SweetAlert2 para mensajes de éxito o error, aparecía el error Swal is not defined en la consola del navegador. Esto indicaba que la librería SweetAlert2 no estaba cargada o no era accesible globalmente. También se observaron errores CORS con CDN de DataTables y problemas de carga de archivos min.js.

\* Solución Aplicada:

\* Se verificó la inclusión correcta de SweetAlert2 en las vistas del frontend.

\* Se recomendaron pasos para asegurar que todos los assets de JavaScript (incluyendo SweetAlert2) se compilaran y cargaran correctamente con npm install y npm run dev / npm run build.

\* Se verificó que los scripts de SweetAlert2 estuvieran cargados antes de ser llamados.

3. Problema: Los productos eliminados lógicamente (estado = false) seguían apareciendo en la vista productos.index.

\* Descripción: Aunque se implementó la eliminación lógica, la vista productos.index mostraba todos los productos sin filtrar por su estado.

\* Solución Aplicada: En el método index de ProductoController, se aseguró que al seleccionar los productos se incluyera el campo estado, permitiendo que la vista los muestre como "Activo" o "Inactivo" según corresponda, y eliminando cualquier filtro where("estado", true) para mostrar todos los productos.

4. Problema: El proyecto Laravel no se subía a GitHub (repositorio remoto vacío o solo con README.md).

\* Descripción: A pesar de varios intentos, el repositorio de GitHub (FinalLaravelCrud) permanecía vacío o solo mostraba el README.md y un Initial commit. El historial de commits en GitHub no reflejaba el commit grande del proyecto Laravel completo.

\* Solución Aplicada (a través de varios intentos):

\* Inicialización de Git: Se confirmó que Git estuviera inicializado correctamente en la carpeta raíz del proyecto Laravel (git init).

\* Staging y Commit: Se verificó que todos los archivos estuvieran añadidos al área de staging (git add .) y que se realizara un commit exitoso (git commit -m "..."). Se identificó que el commit no se había realizado inicialmente, lo que causaba el error src refspec main does not match any.

\* Configuración de Identidad de Git: Se resolvió el error \*\*\* Please tell me who you are. al configurar el user.email y user.name globalmente (git config --global).

\* Conexión con el Remoto: Se aseguró que el repositorio local estuviera correctamente vinculado al repositorio remoto de GitHub (git remote add origin ...).

\* Autenticación de GitHub: Se resolvió la ventana de autenticación de GitHub en VS Code utilizando la opción de "Sign in with your browser" para autorizar la conexión.

\* Historiales No Relacionados (fatal: refusing to merge unrelated histories): Este fue un punto crítico. Git se negó a fusionar porque los historiales local y remoto no tenían una base común.

\* Solución Final: Se instruyó a usar git push -u origin main --force para forzar la subida del historial local (que contenía el proyecto completo) al repositorio remoto, sobrescribiendo el historial vacío o mínimo en GitHub. Esta fue la solución definitiva para que el proyecto apareciera en GitHub.

## 11. Lecciones Aprendidas

- Comprensión profunda del diseño de bases de datos con relaciones muchos a muchos y campos adicionales.  
- Dominio del uso de Laravel para construir módulos CRUD completos.  
- Implementación de interfaces modernas usando AdminLTE.  
- Aplicación de validaciones del lado del cliente y servidor.  
- Importancia de seguir la arquitectura MVC para mantener un código limpio y escalable.

## 12. Conclusiones

El sistema desarrollado cumple de forma efectiva los objetivos planteados, demostrando que es posible implementar una solución funcional de gestión comercial utilizando Laravel. El adecuado manejo de relaciones, validaciones y estructuras de datos refuerza la lógica del sistema y garantiza una experiencia de usuario satisfactoria. Se logró integrar los módulos correctamente, optimizar la base de datos y diseñar una interfaz accesible y eficiente.

## 13. Recomendaciones

- Incorporar autenticación y roles de usuario.  
- Añadir funciones de exportación de reportes (PDF, Excel).  
- Implementar filtros avanzados en los listados.  
- Mejorar el control de stock ante ventas de productos agotados.  
- Considerar el despliegue del sistema en la nube para su uso real.