BitBusters

¡BitBusters, bit a bit,
conquistamos el mundo de las
bases de datos un 1 o 0 a la vez!

UNIVERSIDAD AUTÓNOMA METROPOLITANA

UNIDAD AZCAPOTZALCO

Bases de datos

Proyecto final - 04

PROFESOR:

Hugo Pablo Leyva

INTEGRANTES DEL EQUIPO:

Diego Flores Chávez - 2203031129

Carlos Adrián Delgado Frías - 2192000090

GRUPO:

CSI81

Página web

Índice

Introducción

Descripción del proyecto

El proyecto consiste en el desarrollo de un panel de control web destinado a la gestión eficiente de la base de datos de Exmicror, una empresa especializada en la

comercialización y producción de desinfectantes naturales con propiedades curativas. Esta plataforma proporciona funcionalidades para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en cada una de las tablas que conforman la base de datos alojada en PostgreSQL. Su propósito fundamental es optimizar la administración de datos, permitiendo una manipulación ágil y segura de la información relacionada con los productos, clientes, inventario y demás aspectos relevantes para la operación de Exmicror.

Objetivos

Objetivo general

Desarrollar un panel de control web para la gestión integral de la base de datos de Exmicror, con el fin de optimizar la administración de información relacionada con la venta y fabricación de desinfectantes naturales y curativos.

Objetivos específicos

- Implementar funcionalidades de Crear, Leer, Actualizar y Eliminar (CRUD) en las tablas principales de la base de datos, permitiendo la gestión eficiente de productos, clientes, inventario y pedidos.
- Diseñar una interfaz intuitiva y amigable que facilite la navegación y el acceso a la información relevante, garantizando una experiencia de usuario satisfactoria para los empleados encargados de la administración de la base de datos.

Arquitectura del sistema

Tecnologías utilizadas

Laravel

Laravel es un framework de desarrollo de aplicaciones web con una sintaxis elegante y expresiva, basado en el lenguaje de programación PHP. Su estructura modular y su amplio ecosistema de herramientas hacen de Laravel una elección popular para el desarrollo de proyectos web robustos y escalables. La motivación para utilizar Laravel radica en su capacidad para agilizar el proceso de desarrollo mediante la automatización de tareas comunes, como la gestión de rutas, la autenticación de usuarios y la interacción con bases de datos. Entre las ventajas destacadas de Laravel se encuentran su arquitectura MVC (Modelo-Vista-Controlador) que fomenta la separación de preocupaciones y facilita el mantenimiento del código, así como su activa comunidad de desarrolladores que contribuyen con extensiones y recursos útiles. La interoperabilidad de Laravel con otras tecnologías, a través de la integración de paquetes y APIs, permite una integración fluida con otras herramientas y servicios.

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto conocido por su robustez, confiabilidad y capacidad para manejar grandes volúmenes de datos. La elección de PostgreSQL como motor de base de datos se fundamenta en su cumplimiento con los estándares SQL, su soporte para características avanzadas como la replicación, la alta disponibilidad y la extensibilidad a través de la creación de funciones y tipos de datos personalizados. Además, PostgreSQL ofrece una amplia gama de herramientas y extensiones para el rendimiento, la seguridad y la administración de bases de datos. Su interoperabilidad con otros sistemas y lenguajes de programación, mediante controladores y adaptadores específicos, facilita la integración con el resto de la aplicación.

Docker

Docker es una plataforma de contenedores que simplifica el desarrollo, la implementación y la administración de aplicaciones mediante la encapsulación de software y sus dependencias en contenedores ligeros y portátiles. La adopción de Docker como herramienta de virtualización se justifica por su capacidad para crear entornos de desarrollo y producción consistentes y reproducibles, independientemente del sistema operativo subyacente. La utilización de contenedores Docker facilita la gestión de las dependencias del proyecto, la escalabilidad horizontal y la implementación en entornos de producción, al tiempo que garantiza la portabilidad y la consistencia entre los diferentes entornos de desarrollo. Además, Docker ofrece una integración fluida con herramientas de orquestación como Docker Compose y Kubernetes, lo que facilita la gestión y la escalabilidad de aplicaciones distribuidas.

Descripción general de la arquitectura

La arquitectura de la aplicación sigue los principios estándar del framework Laravel, basados en el patrón de diseño Modelo-Vista-Controlador (MVC), proporcionando una estructura organizada y modular para el desarrollo de aplicaciones web robustas y escalables.

Modelo-Vista-Controlador (MVC):

- **Modelo:** En Laravel, los modelos representan la estructura y lógica de la base de datos. Cada tabla de la base de datos tiene su propio modelo asociado, el cual define las interacciones con los datos.

  “En el contexto de este proyecto, la capa de "Modelo" no se refiere a clases de modelos en el sentido tradicional de un framework MVC, ya que la base de datos ya estaba establecida antes del inicio del proyecto Laravel. En su lugar, la lógica de acceso a los datos se gestiona a través de procedimientos almacenados y gatillos almacenados en la base de datos PostgreSQL. Estos procedimientos almacenados y gatillos actúan como el componente lógico del "Modelo", definiendo las operaciones y la lógica de negocio directamente en la base de datos. Cada controlador interactúa con la base de datos invocando estos procedimientos almacenados para realizar operaciones CRUD en los datos. Esta arquitectura proporciona un enfoque centralizado y eficiente para gestionar la interacción con la base de datos, manteniendo la consistencia e integridad de los datos en todo momento.”

- **Vista:** Las vistas son las interfaces de usuario que se presentan al usuario final. En este proyecto, cada tabla de la base de datos tiene su propia carpeta de vistas en la carpeta "views", lo que facilita la organización y mantenimiento del código.

- **Controlador:** Los controladores actúan como intermediarios entre el modelo y la vista, gestionando las solicitudes del usuario y coordinando la lógica de la aplicación. En este proyecto, se utilizan controladores dedicados para cada entidad de la base de datos, como "PackagingDetailsController", "BarcodeLabelsController", entre otros.

Rutas

Las rutas en Laravel, definidas en el archivo "web.php", establecen la correspondencia entre las URLs del sitio web y las acciones que deben ejecutarse en respuesta a esas URLs. Cada ruta especifica un URI y un controlador junto con el método del controlador que debe ejecutarse cuando se accede a esa URI. Por ejemplo:

Bloque de código 1. Ruta de la página de Exmicror.

```
Route::get('/packaging', [PackagingDetailsController::class, 'index']);
```

Esta ruta mapea la URL "/packaging" a la acción "index" del controlador "PackagingDetailsController", lo que significa que al acceder a "/packaging", se mostrará la lista de detalles de empaque.

Ventajas y motivación

- **Organización y mantenimiento:** La arquitectura MVC de Laravel proporciona una estructura clara y organizada que facilita el mantenimiento y la escalabilidad del proyecto.
- **Separación de preocupaciones:** La separación de las responsabilidades entre el modelo, la vista y el controlador permite un desarrollo más modular y una mejor reutilización del código.
- **Productividad:** El uso de Laravel agiliza el proceso de desarrollo gracias a su amplio conjunto de características integradas y su comunidad activa.
- **Interoperabilidad:** Laravel es compatible con una amplia gama de tecnologías y servicios, lo que facilita la integración con otras herramientas y sistemas externos.
- **Facilidad de despliegue:** La estructura estandarizada de Laravel y su compatibilidad con herramientas como Docker simplifican el despliegue y la administración de la aplicación en entornos de producción.

Consulta

Para facilitar la referencia futura en la documentación y mejorar la accesibilidad al código fuente del proyecto, se proporciona un repositorio público en GitHub que contiene todos los archivos relacionados con el desarrollo de la aplicación. Este repositorio incluye tanto el código fuente de la aplicación desarrollada con Laravel, como el script de SQL utilizado para la creación y mantenimiento de la base de datos PostgreSQL. La disponibilidad del código fuente en un repositorio centralizado permite a los desarrolladores acceder fácilmente a todas las partes del proyecto, revisar el código, realizar contribuciones y colaborar en futuras mejoras:

https://github.com/DieeegoFC/Exmicror-DB

Desarrollo

Detalles sobre la implementación del proyecto

Configuración de controladores

Los controladores en este proyecto siguen una estructura coherente y modular para gestionar las diferentes acciones y operaciones relacionadas con cada entidad de la base de datos. Cada controlador está diseñado para interactuar con la base de datos PostgreSQL a través de procedimientos almacenados, facilitando así la manipulación eficiente de los datos.

- **Método index():** Este método se encarga de recuperar todos los registros de la tabla correspondiente en la base de datos y enviarlos a la vista asociada para su visualización. Utiliza la función DB::select() para ejecutar una consulta SQL y obtener los datos necesarios.
- **Método create():** El método create() simplemente devuelve la vista de creación correspondiente, que contiene un formulario para ingresar nuevos datos.
- **Método store(Request $request):** Cuando se envía el formulario de creación, el método store() recibe la solicitud HTTP y extrae los datos ingresados por el usuario. Luego, utiliza un procedimiento almacenado para insertar estos datos en la base de datos y redirige al usuario a la página de inicio correspondiente.
- **Método edit($id):** Este método recibe el ID del registro que se desea editar y recupera los datos asociados a través de una consulta SQL. Luego, estos datos se pasan a la vista de edición para que el usuario pueda realizar los cambios necesarios.
- **Método update(Request $request, $id):** Después de que el usuario envía el formulario de edición, el método update() recibe la solicitud HTTP y extrae los datos modificados. Utiliza un procedimiento almacenado para actualizar el registro correspondiente en la base de datos y redirige al usuario a la página de inicio.
- **Método destroy($id):** Finalmente, el método destroy() elimina el registro específico de la base de datos utilizando un procedimiento almacenado y redirige al usuario a la página de inicio.

Esta estructura de controladores asegura una separación clara de las responsabilidades y promueve la reutilización del código al utilizar procedimientos almacenados para la lógica de acceso a datos[1].

Bloque de código 2. Ejemplo de controlador: BarcodeLabels.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
```

[1] Se recomienda encarecidamente consultar el repositorio de GitHub provisto en la sección de Consulta de esta documentación. Ahí se pueden revisar los detalles de la implementación con profundidad.

```php
use Illuminate\Support\Facades\DB;

class BarcodeLabelsController extends Controller
{
    public function index()
    {
        $barcodeLabels = DB::select('SELECT * FROM BarcodeLabels');
        return view('barcodeLabels.index', compact('barcodeLabels'));
    }

    public function create()
    {
        return view('barcodeLabels.create');
    }

    public function store(Request $request)
    {
        $labelInformation = $request->input('label_information');
        $barcode = $request->input('barcode');

        $id = DB::selectOne('SELECT insert_barcode_label(?, ?) AS id', [$labelInformation,
$barcode])->id;

        return redirect('/barcodeLabels');
    }

    public function edit($id)
    {
        $barcodeLabel = DB::selectOne('SELECT * FROM BarcodeLabels WHERE id_label = ?', [$id]);
        return view('barcodeLabels.edit', compact('barcodeLabel'));
    }

    public function update(Request $request, $id)
    {
        $labelInformation = $request->input('label_information');
        $barcode = $request->input('barcode');

        DB::selectOne('SELECT update_barcode_label(?, ?, ?)', [$id, $labelInformation, $barcode]);

        return redirect('/barcodeLabels');
    }

    public function destroy($id)
    {
        DB::selectOne('SELECT delete_barcode_label(?)', [$id]);

        return redirect('/barcodeLabels');
    }
}
```

Configuración de contenedores y levantamiento de servicios

El archivo docker-compose.yml proporciona una configuración detallada para la gestión de contenedores y el levantamiento de servicios necesarios para el entorno de desarrollo de la aplicación Exmicror. Este archivo define dos servicios principales: postgres para el motor de base de datos PostgreSQL y laravel para el entorno de desarrollo de Laravel.

Servicio PostgreSQL

- **Imagen y contenedor:** Este servicio utiliza la imagen oficial de PostgreSQL, con la última versión disponible. El contenedor se nombra como exmicror-db para facilitar la identificación.
- **Variables de entorno:** Se establecen variables de entorno para configurar la base de datos, incluyendo el nombre de la base de datos (DB_NAME), el usuario (DB_USER) y la contraseña (DB_PASSWORD).
- **Puertos:** El servicio expone el puerto 5432 para permitir la conexión a la base de datos desde el host.
- **Volúmenes:** Se montan volúmenes para persistir los datos de la base de datos (./data:/var/lib/postgresql/data), así como para cargar archivos SQL de inicialización (./sql:/docker-entrypoint-initdb.d) y archivos de volcado de datos (./dump:/csv_data).

Servicio Laravel

- **Imagen y contenedor:** Para el entorno de desarrollo de Laravel, se utiliza la imagen Ubuntu como base. El contenedor se nombra como exmicror-app.
- **Puertos:** El servicio expone el puerto 8000 para acceder a la aplicación desde el host.
- **Volúmenes:** Se monta un volumen para vincular el directorio de la aplicación Laravel (./exmicror:/app), lo que permite realizar cambios en el código fuente de manera persistente.
- **Comando:** Se ejecuta un comando Bash para realizar la configuración necesaria dentro del contenedor, incluyendo la instalación de PHP y sus extensiones, la configuración de la zona horaria y el inicio del servidor de desarrollo de Laravel utilizando php artisan serve.
- **Dependencias:** Este servicio depende del servicio postgres, lo que garantiza que la base de datos esté disponible antes de que se inicie el servidor de Laravel.

Esta configuración de Docker Compose proporciona un entorno de desarrollo coherente y escalable para el proyecto Exmicror, permitiendo una fácil gestión de los servicios y una rápida puesta en marcha del entorno de desarrollo.

Bloque de código 3. Configuración de los contenedores.

```
version: '3.8'

services:
  postgres:
    image: postgres:latest
    container_name: exmicror-db
    environment:
      POSTGRES_DB: ${DB_NAME}
      POSTGRES_USER: ${DB_USER}
      POSTGRES_PASSWORD: ${DB_PASSWORD}
    ports:
      - "5432:5432"
    volumes:
      - ./data:/var/lib/postgresql/data
      - ./sql:/docker-entrypoint-initdb.d
      - ./dump:/csv_data

  laravel:
```

```
    image: ubuntu:latest
    container_name: exmicror-app
    ports:
      - "8000:8000"
    volumes:
      - ./exmicror:/app
    command: >
      bash -c "
        export DEBIAN_FRONTEND=noninteractive &&
        apt-get update &&
        apt-get install -y php php-cli php-mbstring php-xml php-pgsql tzdata &&
        ln -fs /usr/share/zoneinfo/Your/Timezone /etc/localtime &&
        dpkg-reconfigure --frontend noninteractive tzdata &&
        cd /app &&
        php artisan serve --host=0.0.0.0 --port=8000"
    depends_on:
      - postgres
```

Gracias a esta configuración, basta con ejecutar el comando “docker compose up -d” para iniciar el proyecto y usarlo[2].

Pruebas de uso

En esta sección, se presentarán casos de ejemplo seleccionados para demostrar el funcionamiento esencial de la aplicación Exmicror bajo diferentes escenarios de uso. A través de capturas de pantalla y descripciones detalladas, se ilustra la interacción del usuario con las funcionalidades clave de la aplicación, proporcionando una visión clara y concisa de su operatividad y usabilidad. Estas pruebas de uso servirán como referencia práctica para los usuarios y contribuirán a validar el rendimiento y la eficacia de la aplicación en situaciones del mundo real.

[2] Se recomienda revisar la sección de instrucciones para el despliegue en esta documentación.

Preparación del servidor

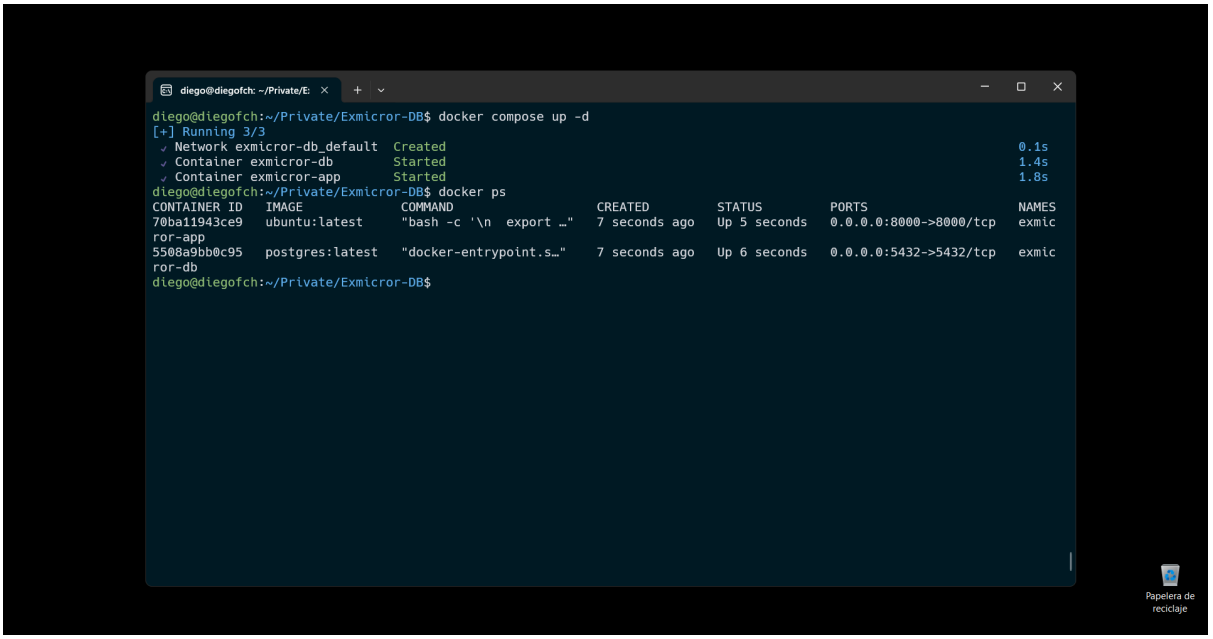

Figura 1. Levantamiento exitoso de la aplicación.

Prueba de control de detalles de embalaje

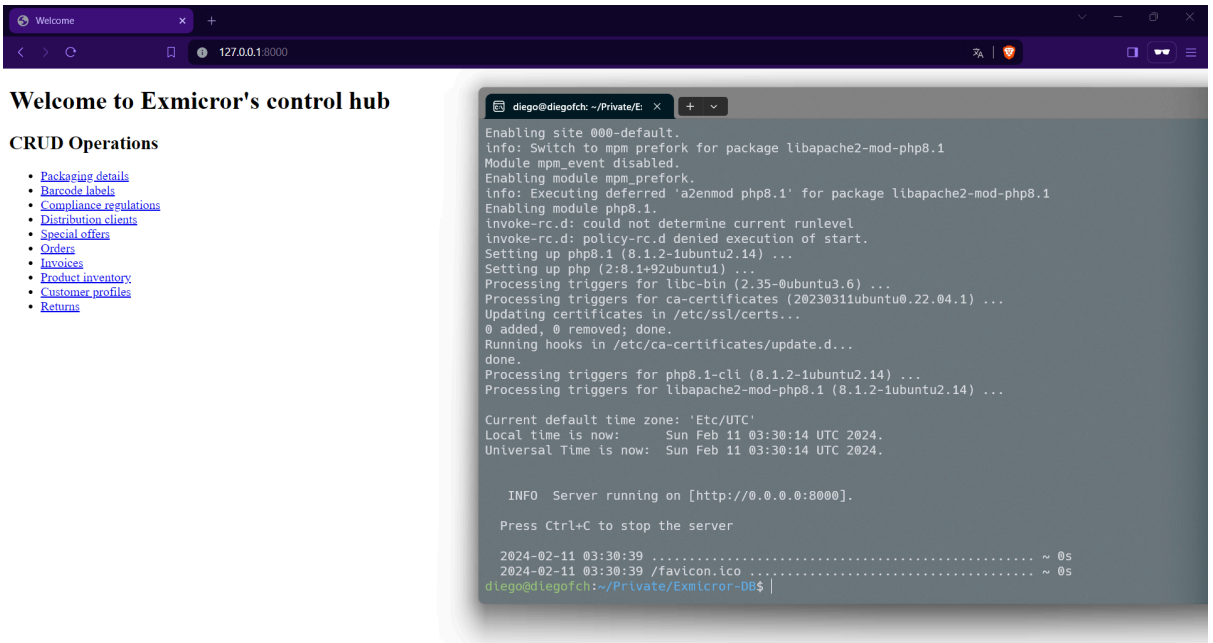

Figura 2. Acceso verificado a la página web.

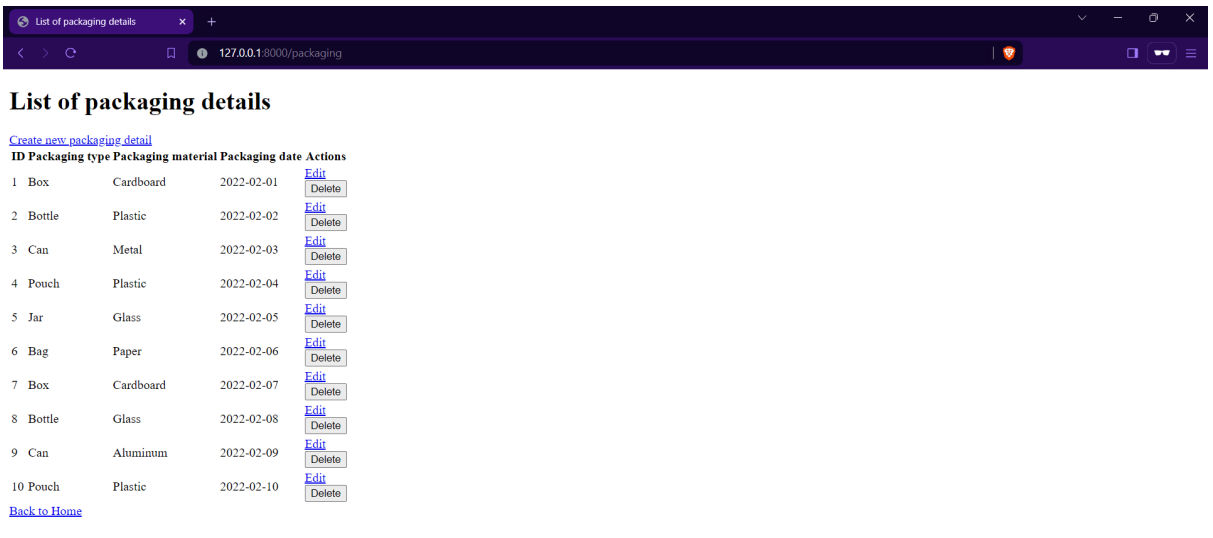

Figura 3. Vista de detalles de embalaje.

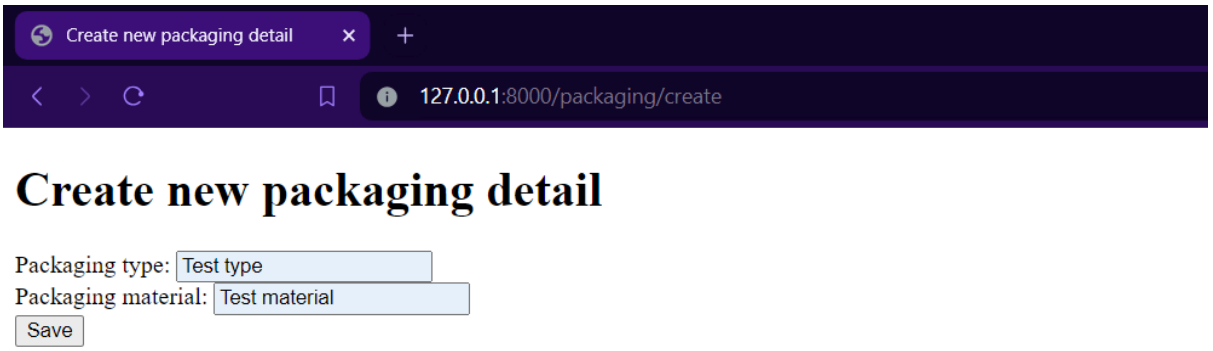

Figura 4. Vista de creación de un detalle de embalaje.

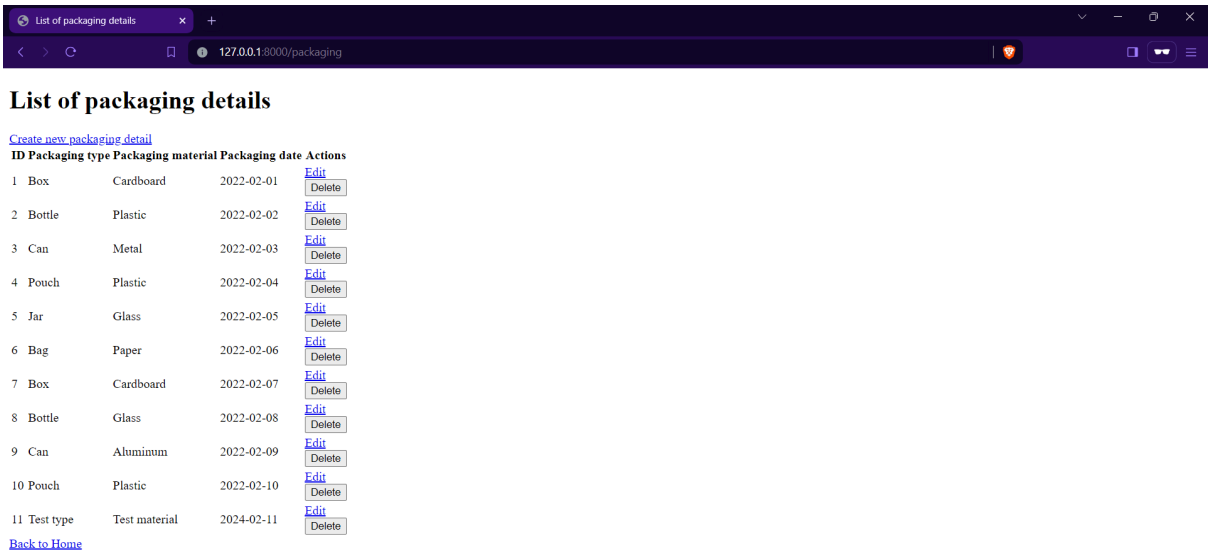

Figura 5. Creación exitosa de un detalle de embalaje.

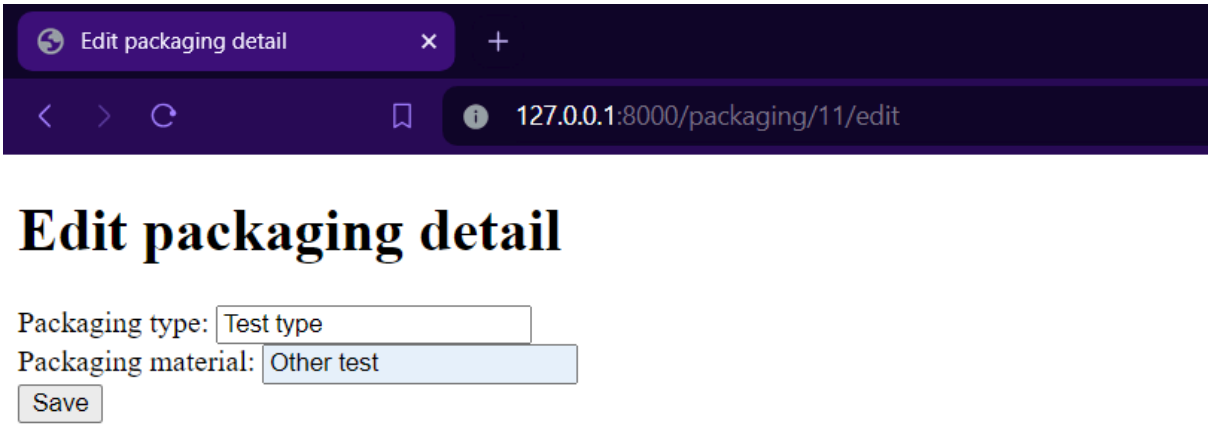

Figura 6. Vista de edición de un detalle de embalaje.

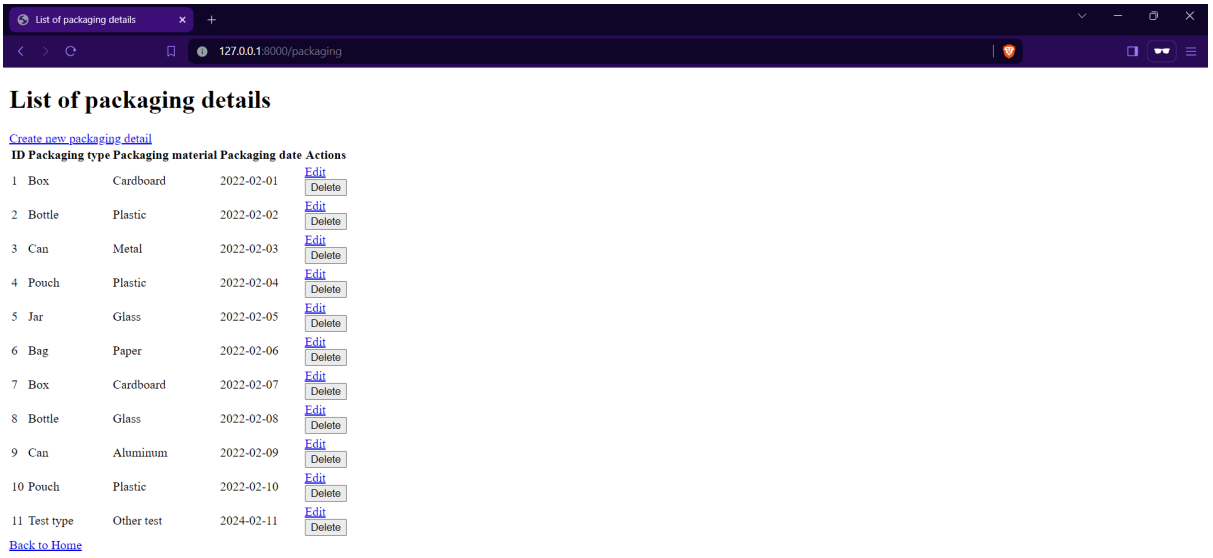

Figura 7. Edición exitosa de un detalle de embalaje.

Figura 7. Eliminación exitosa de un detalle de embalaje.

Prueba de control de inventario de productos

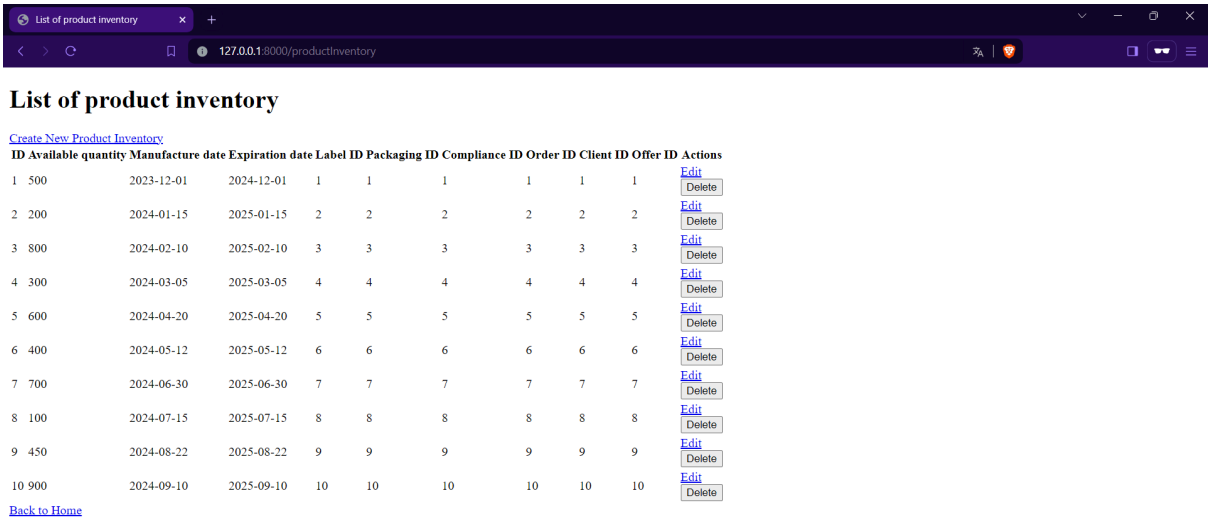

Figura 8 Vista de inventario de productos.

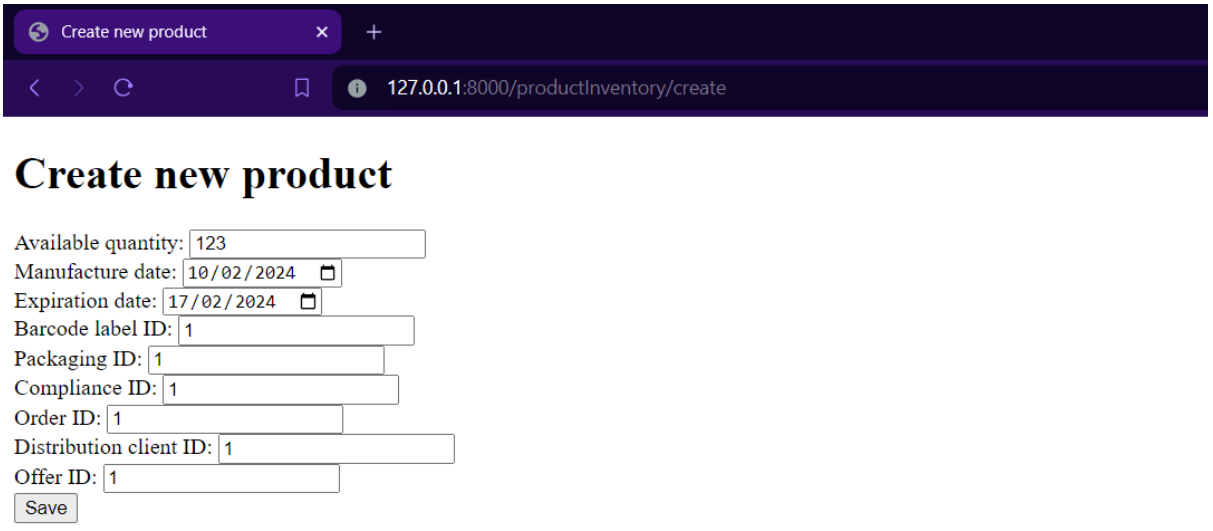

Figura 9. Vista de creación de un producto.

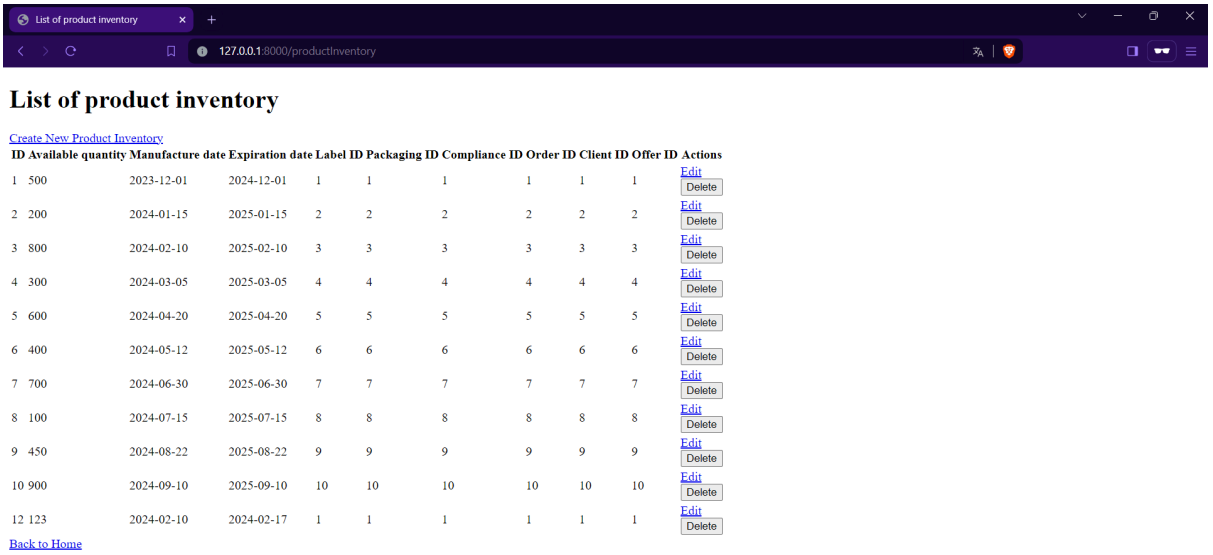

Figura 10. Creación exitosa de un producto

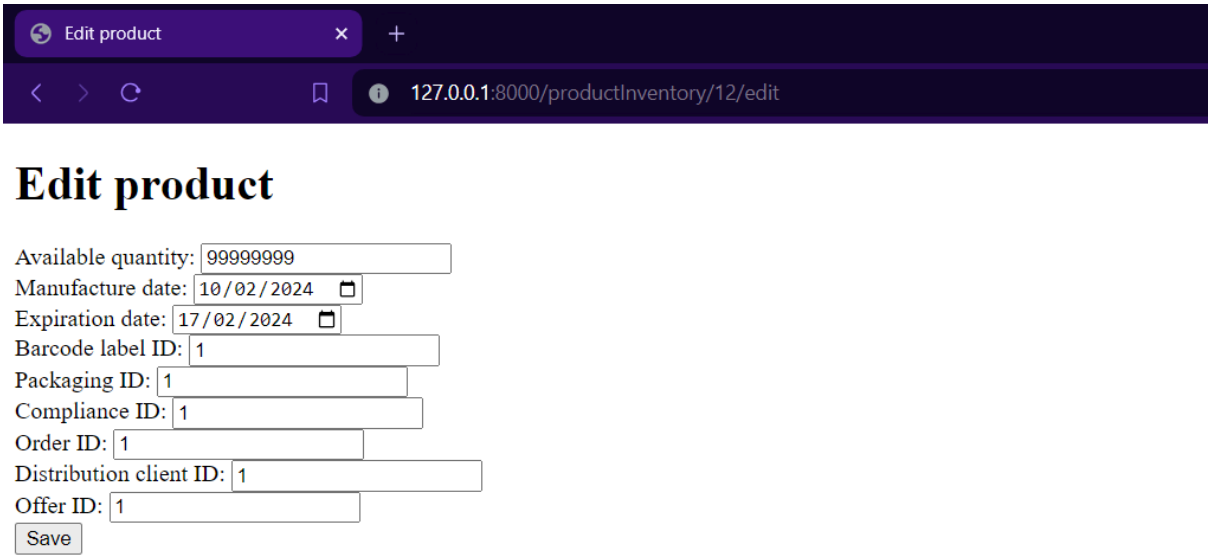

Figura 11. Vista de edición de un producto.

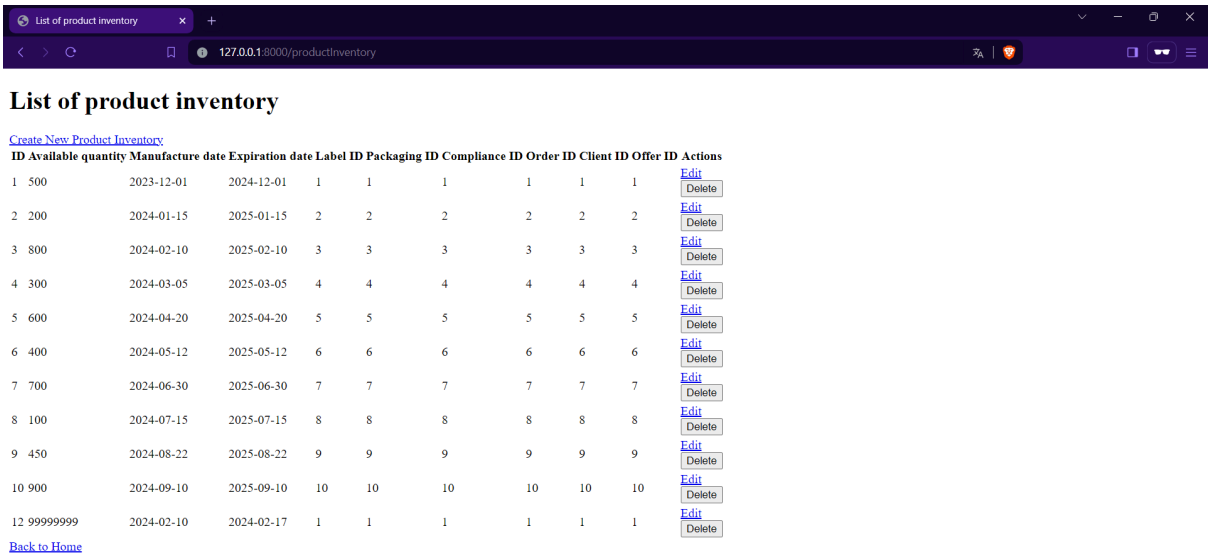

Figura 12. Edición exitosa de un producto.

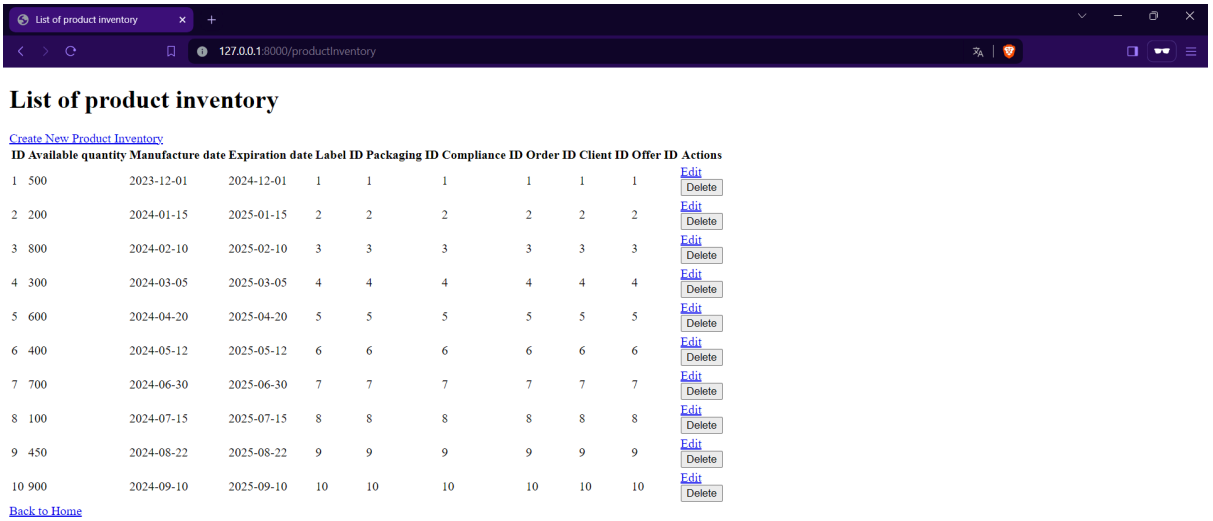

Figura 13. Eliminación exitosa de un producto.

Resultados

Los resultados presentados en esta subsección confirman el correcto funcionamiento de la página web de Exmicror, evidenciado a través de las capturas de pantalla que muestran la ejecución exitosa de operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre dos elementos fundamentales en la base de datos.

Las capturas proporcionadas han sido seleccionadas estratégicamente para demostrar la funcionalidad esencial de la aplicación y su capacidad para gestionar datos de manera efectiva. Se ha optado por mostrar operaciones CRUD sobre elementos representativos, lo que permite validar la funcionalidad general de la plataforma sin necesidad de incluir capturas de absolutamente todas las operaciones posibles.

Es importante destacar que todas las operaciones CRUD en la aplicación Exmicror siguen exactamente el mismo formato y estructura, lo que hace que la demostración de un conjunto representativo de casos sea suficiente para validar la consistencia y el funcionamiento adecuado de todas las funcionalidades. Incluir capturas de todas las operaciones sería redundante y aumentaría innecesariamente la extensión de esta sección.

Instrucciones para el despliegue de la aplicación

Para desplegar la aplicación Exmicror en un entorno de producción, se pueden seguir los siguientes pasos:

- **Configuración del entorno:** Asegúrese de tener un servidor adecuado disponible para alojar la aplicación. Este servidor debe tener Docker y Docker Compose instalados y configurados correctamente.

- **Preparación del archivo docker-compose.yml:** Descargue el archivo docker-compose.yml proporcionado y asegúrate de que esté correctamente configurado para tu entorno de producción. Ajuste las variables de entorno según sea necesario para la configuración de la base de datos PostgreSQL (DB_NAME, DB_USER, DB_PASSWORD).
- **Despliegue de la aplicación:** Ejecute el siguiente comando en el directorio donde se encuentra el archivo docker-compose.yml para levantar el servicio de la base de datos PostgreSQL y la página de Laravel en conjunto:

Bloque de código 4. Comando para el levantamiento de los servicios.

```
docker-compose up -d
```

- **Verificación del despliegue**: Una vez que los contenedores estén en funcionamiento, verifique que la aplicación esté disponible accediendo a la dirección IP o nombre de dominio de su servidor seguido del puerto 8000 en tu navegador web. Por ejemplo: http://su_direccion_ip:8000.
- **Configuración adicional (opcional):** Si es necesario, configure un servidor web como Nginx o Apache para que actúe como proxy inverso y redirija el tráfico al contenedor de la aplicación Laravel en el puerto 8000. Esto puede mejorar el rendimiento y la seguridad de la aplicación en producción.

Configuración de las variables de entorno

El archivo .env es un componente crítico en la configuración de Laravel, ya que contiene información sensible y parámetros de conexión necesarios para interactuar con la base de datos y otros servicios. En el caso de la configuración de la base de datos, las variables de entorno juegan un papel fundamental en la especificación de los detalles de conexión.

A continuación se detallan las variables de entorno relacionadas con la configuración de la base de datos en el archivo .env:

- DB_CONNECTION: Este parámetro especifica el tipo de conexión que se utilizará para interactuar con la base de datos. Puede tener valores como mysql, pgsql, sqlite, etc. Dependiendo del motor de base de datos que estés utilizando.
- DB_HOST: Aquí se debe especificar la dirección IP del servidor de base de datos al que se conectará la aplicación Laravel. En el caso de despliegues locales o en un entorno Dockerizado, generalmente se utiliza 127.0.0.1 o localhost. Sin embargo, para la configuración en producción, se debe usar la dirección IP de la misma máquina.
- DB_PORT: Este parámetro define el puerto en el que el servidor de base de datos está escuchando las conexiones entrantes. El valor predeterminado para PostgreSQL es 5432.
- DB_DATABASE: Indica el nombre de la base de datos a la que se conectará la aplicación Laravel.
- DB_USERNAME: Especifica el nombre de usuario que se utilizará para autenticarse en la base de datos.

- DB_PASSWORD: Este parámetro contiene la contraseña asociada al usuario de la base de datos.

Es importante destacar que, al configurar la variable DB_HOST en el archivo .env, se debe utilizar la dirección IP de la misma máquina en la que se encuentra alojada la base de datos PostgreSQL. Esto es especialmente relevante en entornos de producción, donde la base de datos puede residir en un servidor diferente al de la aplicación web.

Posibles mejoras o características futuras

La entrega actual de la aplicación Exmicror se ha enfocado principalmente en el desarrollo y la funcionalidad del backend, dejando el diseño estético del frontend en un estado básico debido a limitaciones de tiempo y priorización de la funcionalidad principal. Sin embargo, existen diversas áreas de mejora y características adicionales que podrían implementarse en futuras iteraciones para mejorar la experiencia del usuario y fortalecer la seguridad del sistema.

- ➔ **Mejoras en el diseño estético del frontend:**
  - ◆ *Introducción de frameworks de diseño:* Se pueden incorporar librerías como Bootstrap, Tailwind CSS o Chakra UI para mejorar el aspecto visual y la usabilidad del frontend. Estos frameworks proporcionan componentes predefinidos y estilos CSS que facilitan la creación de interfaces de usuario atractivas y responsivas.
  - ◆ *Refinamiento de la experiencia del usuario:* Se pueden realizar ajustes en la disposición de los elementos, la paleta de colores y los estilos visuales para crear una experiencia de usuario más agradable y coherente en toda la aplicación.
- ➔ **Implementación de un sistema de autenticación:**
  - ◆ *Incorporación de Laravel Passport:* Laravel Passport es una biblioteca que facilita la implementación de autenticación OAuth2 y la gestión de tokens de acceso para proteger las rutas y recursos de la aplicación. Esto permitirá a los usuarios registrarse, iniciar sesión de forma segura y acceder a áreas protegidas de la aplicación.
  - ◆ *Configuración de roles y permisos:* Se pueden establecer roles de usuario (administrador, empleado, cliente, etc.) y definir permisos específicos para cada rol, lo que garantiza un acceso controlado a las funcionalidades y datos de la aplicación.
- ➔ Refuerzo de la seguridad:
  - ◆ *Implementación de medidas de seguridad adicionales:* Se pueden añadir capas adicionales de seguridad, como la protección contra ataques de inyección SQL, la validación de datos de entrada y la autenticación de dos factores, para prevenir vulnerabilidades de seguridad y proteger la integridad de los datos.
  - ◆ *Auditoría de seguridad:* Realizar auditorías regulares de seguridad para identificar posibles puntos de vulnerabilidad y aplicar parches de seguridad según sea necesario para mantener la aplicación protegida contra posibles amenazas externas.

- ➔ **Optimización del rendimiento:**
  - ◆ *Mejora de la velocidad de carga:* Realizar optimizaciones en el código y la estructura de la aplicación para reducir los tiempos de carga y mejorar la respuesta del servidor, lo que proporcionará una experiencia de usuario más rápida y fluida.
  - ◆ *Implementación de caché:* Utilizar técnicas de almacenamiento en caché para almacenar datos temporales y reducir la carga en la base de datos, lo que mejorará el rendimiento general de la aplicación.

En conclusión, la incorporación de estas mejoras y características futuras no solo mejorará la estética y la funcionalidad del frontend, sino que también fortalecerá la seguridad y el rendimiento de la aplicación Exmicror, garantizando una experiencia de usuario óptima y segura para todos los usuarios.

Conclusión

El desarrollo de la aplicación Exmicror ha sido un proceso enriquecedor que ha involucrado diversos desafíos y aprendizajes significativos. Uno de los aspectos más destacados fue la configuración de los contenedores Docker y el montado de rutas, que representan áreas de mayor dificultad durante el proceso de desarrollo. Sin embargo, a través de la investigación, la experimentación y la colaboración con el equipo, logramos superar estos obstáculos y establecer una infraestructura sólida para la aplicación.

Uno de los principales aprendizajes adquiridos durante la creación de esta aplicación fue la importancia de la planificación y la modularidad en el desarrollo de software. La implementación de operaciones CRUD se alineó perfectamente con la creación de procedimientos almacenados y gatillos en entregas anteriores, lo que nos permitió mantener un código limpio, organizado y fácilmente mantenible.

El trabajo en equipo, la resolución de problemas y la adaptabilidad fueron fundamentales para el éxito de este proyecto. A medida que avanzamos hacia el futuro, esperamos seguir aplicando estos aprendizajes y principios para mejorar continuamente la aplicación Exmicror y ofrecer una experiencia excepcional a nuestros usuarios.