UNIVERSIDAD LUTERANA SALVADOREÑA

Facultad	Licenciatura en ciencias de la computación		
Asignatura	Proyectos de Aplicaciones Web		
Profesor	Licdo. Jorge Alberto Coto Zelaya		
Año	**2023**	**Ciclo**	**2**

Tema del Proyecto: cafeteria

Estudiantes

No.	Apellidos	Nombres	Carnet	Carrera
1	Mendoza Lopez	Cristofer Eduardo	ML01137152	Lic. en Ciencias de la Computación
2	Pérez Hernández	Diana Beatriz	PH01137391	Lic. en Ciencias de la Computación
3	Rauda Martínez	Juan Carlos	RM01137191	Lic. en Ciencias de la Computación
4	Mejia Jacinto	Noemy Marisol	MJ01137252	Lic. en Ciencias de la Computación

San Salvador, El Salvador, América Central

Contenido

Introducción

Crear y diseñar un sitio web nuestra cafetería, diseñada y desarrollada con el poderoso framework Laravel en PHP, se esfuerza por ofrecer una experiencia excepcional tanto en el mundo real como en el ciberespacio. En un mundo donde el café es mucho más que una bebida, en "cafetería" entendemos que cada taza es una historia esperando a ser escrita. Con Laravel, hemos construido una plataforma que permite a nuestros clientes navegar por un menú diverso, descubrir la historia detrás de cada grano de café y realizar pedidos con facilidad y que deseen disfrutar de nuestro café premium desde la comodidad de sus hogares.

Nuestra plataforma en línea, respaldada por la potencia de Laravel, garantiza una experiencia de usuario fluida y segura. Puedes explorar nuestro menú de café, consultar información detallada sobre cada variedad, realizar un pedido personalizado y realizar un seguimiento de tus compras desde cualquier ya sea Tablet o computadora el dispositivo con acceso a Internet.

En "cafetería", la calidad del café es primordial. Desde el cultivo y la selección de granos de café de origen hasta el proceso de tostado y preparación, cada etapa se lleva a cabo con dedicación y pasión. La eficiencia y confiabilidad que Laravel ofrece en la gestión de nuestro sitio web y sistemas internos nos permite centrarnos en lo que más importa: brindar el mejor café a nuestros clientes.

Acompáñanos en este viaje a través del sabor y la tecnología. En "cafetería", la tradición y la innovación se unen gracias a Laravel en PHP para ofrecer una experiencia única. Te invitamos a explorar nuestra carta y a unirte a nuestra comunidad de amantes del café haciendo pedidos desde la comodidad de tu casa.

Objetivo General

Implementar un sitio web integral de promoción y gestión de pedidos en línea para la "cafetería" utilizando el framework Laravel. Este enfoque permitirá mejorar sustancialmente la presencia en línea de la empresa y elevar la satisfacción del cliente al ofrecer una solución personalizada y altamente funcional.

Objetivos Específicos

Se diseñará una interfaz de usuario única que refleje la identidad y la estética de la cafetería. Los elementos visuales, colores y tipografía se seleccionarán cuidadosamente para crear una experiencia visualmente atractiva.

Se implementará un carrito de compras dinámico que permitirá a los clientes revisar y ajustar sus pedidos antes de finalizar la compra. El carrito mostrará los detalles de los productos, las cantidades y los precios totales de manera clara.

Diseñar y desarrollar un sistema de gestión de pedidos en línea utilizando el framework Laravel. Este sistema permitirá a los clientes realizar pedidos de manera eficiente y a la empresa gestionarlos de manera efectiva, optimizando todo el proceso.

optimizar el sitio web mediante el uso del framework Laravel para mejorar significativamente su visibilidad en los motores de búsqueda y, en última instancia, aumentar el tráfico orgánico hacia el sitio.

Justificación

La implementación de un sitio web de pedidos en línea para "cafetería “es crucial para la supervivencia y el crecimiento de la empresa en el mercado actual. Existen varias razones por las cuales la empresa necesita una presencia en línea sólida y un sistema de gestión de pedidos en línea. Mayor alcance y accesibilidad: El sitio web permitirá a "cafetería" llegar a una audiencia más amplia y hacer que sus productos sean más accesibles para los clientes. La presencia en línea también permitirá a la empresa atender a los clientes de manera más eficiente y en cualquier momento del día. Competitividad en el mercado: En un mercado altamente competitivo como el de las cafeterías, la presencia en línea es esencial para la competitividad de la empresa. Sin un sitio web de calidad y un sistema de gestión de pedidos en línea, "cafetería" puede perder oportunidades de negocio frente a la competencia. Mejora en la atención al cliente: La implementación de un sistema de gestión de pedidos en línea mejorará la atención al cliente y aumentará la satisfacción del cliente. Los clientes podrán realizar pedidos de manera más sencilla y eficiente, donde empresa podrá gestionar los pedidos de manera más efectiva. Promoción de la marca y fidelización de clientes: El sitio web permitirá a "cafetería" promocionar su marca y productos de manera más efectiva, lo que puede atraer nuevos clientes y fidelizar a los ya existentes. Además, el sitio web permitirá a la empresa recopilar información valiosa de los clientes y utilizarla para mejorar la experiencia del cliente. Experiencia de usuario optimizada: Laravel ofrece una arquitectura de desarrollo robusta que permite la creación de una plataforma en línea con una interfaz de usuario intuitiva y atractiva. Esto facilita a nuestros clientes la navegación por nuestro menú de café, la selección de productos y la realización de pedidos de manera sencilla. Seguridad de datos: La seguridad es una prioridad absoluta en la gestión de información sensible, como los datos de los clientes y los registros de transacciones. Laravel cuenta con características de seguridad integradas, lo que nos brinda tranquilidad al garantizar la protección de la información de nuestros clientes. Escalabilidad: "cafetería" tiene una visión a largo plazo de crecimiento y expansión. Laravel es altamente escalable, lo que nos permite gestionar un aumento en la demanda de nuestros servicios tanto en la cafetería física como en el comercio en línea, sin comprometer el rendimiento. Desarrollo rápido y eficiente: La eficiencia es esencial en el mundo competitivo de la industria del café. Laravel acelera el proceso de desarrollo, lo que nos permite lanzar nuevas características y actualizaciones de manera oportuna y eficaz. Gestión de contenido flexible: La versatilidad de Laravel en la gestión de contenido nos permite actualizar fácilmente nuestro sitio web y promociones, lo que es crucial para mantener a nuestros clientes informados sobre los nuevos productos, eventos y ofertas especiales. Integración de sistemas: Laravel admite la integración con una variedad de sistemas y aplicaciones de terceros, lo que nos permite ofrecer servicios adicionales, como métodos de pago seguros, sistemas de reserva en línea y programas de fidelización de clientes.

Marco teórico

Descripción del sitio web

Nuestra cafetería es una plataforma en línea diseñada para brindar a nuestros clientes una experiencia única e inolvidable. Nuestro objetivo principal es ofrecer un espacio donde los amantes del café puedan explorar y disfrutar de nuestra amplia variedad de productos y servicios. adquirir nuestros productos de alta calidad desde la comodidad de sus hogares. Al entrar a nuestro sitio web los usuarios pueden observar nuestros productos que son variados, también tenemos la opción de postres y café, nuestro sitio web también incluye una ventana para hacer tus pedidos, la cual eliges lo que deseas y lo compras.

Descripción de la empresa Hace 8 años, en 2015, surgió Coffee Light, una encantadora cafetería en San Salvador. Inspirados por su amor compartido por los postres y el café, Juan Rauda y Saraí Mejía dieron vida a su sueño. Tras minuciosos estudios de mercado y un plan de negocios sólido, obtuvieron fondos a través de inversionistas y ahorros personales. Con dedicación, transformaron el espacio en un rincón acogedor, adornado con detalles elegantes y cálidos. La gran inauguración atrajo a los lugareños con una selección de postres caseros irresistibles y café de calidad. La ubicación estratégica se encuentra en el centro histórico de San Salvador contribuyó al éxito inicial, respaldado por reseñas entusiastas. Con el tiempo, coffee light se convirtió en un refugio querido, donde las delicias dulces se mezclan armoniosamente con el aroma del café, creando memorias y momentos preciosos. En coffee light, hemos creado un espacio donde el placer se encuentra con la perfección en cada taza de café y cada bocado de postre. Somos mucho más que una simple cafetería, somos un destino para formar los mejores momentos. En coffee light, nos esforzamos por brindarte un ambiente acogedor donde puedes relajarte, conectarte con amigos y sumergirte en la delicia de los sabores. Ya sea que estés buscando un momento tranquilo de reflexión con una taza de café caliente o quieras celebrar la vida con un postre extravagante, estamos aquí para hacer que cada visita sea inolvidable. Sumérgete en un mundo de sabores tentadores con nuestros irresistibles postres. Cada bocado es una experiencia de dulzura y texturas exquisitas. ¡Descubre el placer en cada mordisco!

Misión y visión u historia, etc.

Misión

En nuestro sitio web de ventas nos esforzamos por brindar a las personas acceso conveniente y confiable a una amplia variedad de alimentos frescos y de calidad a través de nuestra plataforma de ventas en línea. No basta con crear una tienda online, nosotros queremos mejorarla, hacerla eficiente, lograr un diseño adecuado y personal, también, nos esforzamos por ser el destino en línea preferido para la búsqueda y adquisición de productos, brindando comodidad, confiabilidad y satisfacción en cada interacción.

Visión

Nos enfocamos en destacarnos en el comercio electrónico, redefiniendo constantemente la forma en que las personas compran en línea. Aspiramos a crear un mercado en línea que no solo ofrezca comodidad y accesibilidad, sino que también fomente hábitos alimenticios saludables y sostenibles, respaldados por tecnología de vanguardia y una plataforma segura. Nos esforzamos por ser reconocidos no solo por la calidad de nuestros productos y servicios, sino también por la innovación y el impacto positivo que generamos en la comunidad y en la experiencia de compra en línea en su conjunto.

Historia

Los sitios de ventas de alimentos en línea han experimentado un crecimiento significativo desde sus inicios a fines de la década de 1990. Con el tiempo, la comodidad de comprar alimentos en línea se convirtió en un factor importante para los consumidores, y la variedad de opciones disponibles aumentó considerablemente. La pandemia de COVID-19 aceleró aún más la adopción de compras de alimentos en línea, ya que las restricciones de movilidad llevaron a más personas a buscar formas seguras y convenientes de adquirir de alimentos. En resumen, la historia de los sitios de ventas de alimentos en línea refleja la evolución del comercio electrónico en general. Desde sus modestos comienzos hasta convertirse en un componente esencial del estilo de vida moderno, estos sitios continúan innovando para satisfacer las cambiantes demandas y expectativas de los consumidores en cuanto a comodidad, calidad y selección de alimento

Descripción del proyecto

Nuestro proyecto busca crear una plataforma digital que ofrezca a los clientes una experiencia única de adquirir una variedad de deliciosos postres y distintas variedades del café desde la comodidad de sus hogares. El enfoque principal será brindar productos de alta calidad, un proceso de compra fácil y seguro, y un servicio al cliente excepcional. A continuación, se presenta una descripción más detallada del proyecto:

1. Catálogo de Productos: El sitio contendrá una amplia gama de postres y opciones de café de buena calidad. Esto incluirá pasteles, tartas, galletas, chocolates y otros dulces, así como una selección de granos de café, café molido y productos relacionados, como tazas y cafeteras.
2. Experiencia de Usuario Intuitiva: El sitio se diseñará con una interfaz amigable y fácil de usar, permitiendo a los usuarios navegar por categorías, filtrar productos y acceder a información detallada de cada artículo. Se prestará especial atención a la presentación visual de los productos para estimular el apetito y el deseo de compra.
3. Personalización y Recomendaciones: Los usuarios tendrán la opción de crear perfiles con sus preferencias y gustos. Basándose en esta información, el sistema ofrecerá recomendaciones personalizadas para productos, lo que mejorará la experiencia del cliente y fomentará compras repetidas.
4. Seguridad en las Transacciones: Se implementarán medidas de seguridad robustas para garantizar que las transacciones en línea sean seguras y protegidas. Los métodos de pago serán diversos y confiables, incluyendo tarjetas de crédito, PayPal y otras opciones populares.
5. Entrega Eficiente: Se establecerá un sistema de entrega eficiente para garantizar que los productos lleguen frescos y en óptimas condiciones. Las opciones de entrega podrían incluir envío estándar, entrega exprés y recolección en tienda.
6. Atención al Cliente: Se brindará un servicio al cliente excepcional a través de múltiples canales, como correo electrónico y línea telefónica. Las consultas, problemas o comentarios de los clientes se resolverán de manera rápida y eficiente.
7. Responsabilidad Social y Sostenibilidad: Se considerarán prácticas sostenibles en el empaquetado y en la elección de proveedores. Se podría incluso explorar la posibilidad de donaciones a organizaciones benéficas locales.

Tecnologías a utilizar

Las tecnologías utilizadas para realizar nuestro proyecto primeramente utilizamos un editor de código llamado visual studio code, el cual nos facilitara muchas herramientas, facilitan la escritura y la administración del código también utilizamos laragon el cual nos ayuda a crear y gestionar aplicaciones web.

Visual Studio Code

Cuenta con soporte para depuración de código, y dispone de un sinnúmero de extensiones, que básicamente te da la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación. Inicialmente incluye un mínimo de componentes y funciones básicas de un editor con soporte nativo para JavaScript/TypeScript y Node.js, sin embargo, es personalizable con los cientos de plugins o extensiones disponibles para escribir código en diferentes lenguajes.

VS Studio Code incluye una terminal con todas las funciones, la cual se inicia fácilmente en el directorio de trabajo. La terminal integrada puede utilizar cualquier Shell instalado en el equipo, como PowerShell, Bash o cualquier otro. Contar con una terminal en el propio editor es de gran utilidad para ejecutar diferentes comandos necesarios cuando estamos desarrollando.

Características de Visual Studio Code

VS Code tiene una gran variedad de características útiles para agilizar el trabajo, que lo hacen el editor preferido por muchos (me incluyo) para trabajar los proyectos.

Multiplataforma: Es una característica importante en cualquier aplicación y más si trata de desarrollo. Visual Studio Code está disponible para Windows, GNU/Linux y macOS.

IntelliSense: Esta característica está relacionada con la edición de código, autocompletado y resaltado de sintaxis, lo que permite ser más ágil a la hora de escribir código. Como su nombre lo indica, proporciona sugerencias de código y terminaciones inteligentes en base a los tipos de variables.

Depuración: Visual Studio Code incluye la función de depuración que ayuda a detectar errores en el código. De esta manera, nos evitamos tener que revisar línea por línea a puro ojo humano para encontrar errores. VS Code también es capaz de detectar pequeños errores de forma automática antes de ejecutar el código o la depuración como tal.

Extensiones: Hasta ahora, he mencionado varias veces el término *extensiones* porque es uno de los puntos fuertes. Visual Studio Code es un editor potente y en gran parte por las extensiones. Las extensiones nos permiten personalizar y agregar funcionalidad adicional de forma modular y aislada. Por ejemplo, para programar en diferentes lenguajes

Laragon

En Laragon puedes crear fácilmente un entorno de desarrollo completo y funcional, sin preocuparte por la configuración del servidor.

características: es super rápido, fácil de usar, productivo y potente entorno de desarrollo para todos.

Laragon tiene un entorno aislado con sistema operativo y ofrece todo lo necesario para crear aplicaciones web modernas. Es portátil y muy flexible. Puedes mover la carpeta Laragon alrededor (a otros discos, a otros portátiles, sincronizar con Cloud, …) y la magia de Laragon estará contigo.

Trabajar Laragon es fácil y un placer, ya que tiene Apache + Nginx totalmente administrado.

Ventajas

1) Ligero

Si no tienes tanto espacio en tu máquina, no hay problema. Laragon portátil tiene solo 38 MB y, mientras se ejecuta, usa menos de 4 MB de RAM, pero si aun así no deseas que el entorno esté instalado en tu computadora

2) Facilidad

Es posible utilizar Laragon sin mayores dificultades, incluso sin leer toda la documentación, ya que su panel es minimalista e intuitivo. Además, es un entorno aislado, es decir, sin conflictos de herramientas.

3) Libertades

Entorno universal para PHP, Node.js, Python, Java, GO, Ruby, MariaDB. Laragon soporta casi todos los paquetes, sin tener la aplicación instalada en la máquina

4) Seguridad

En Laragon es posible generar certificados SSL automáticamente. Además, es posible cambiar las versiones de las herramientas con un solo clic

Diseño wireframe

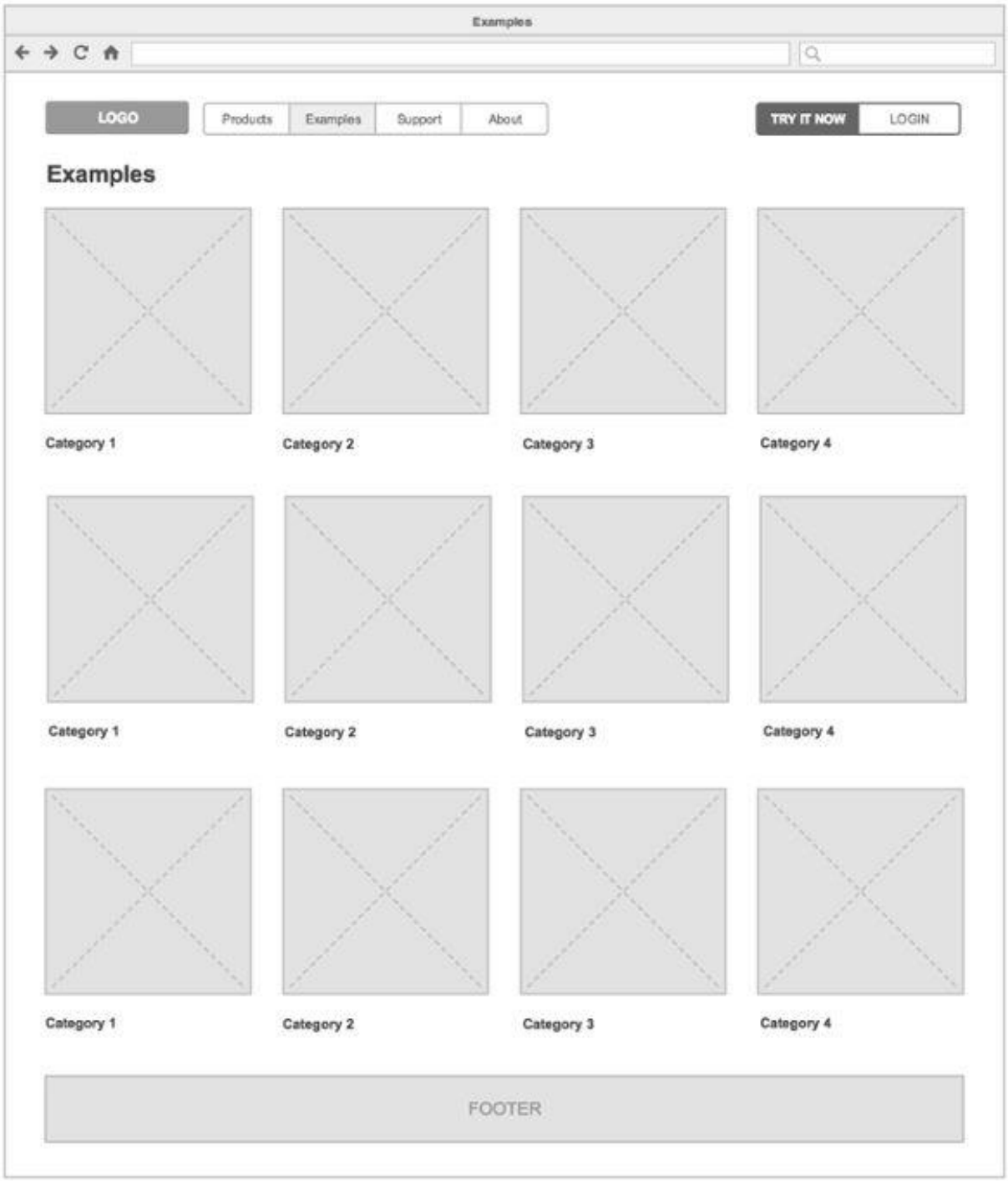
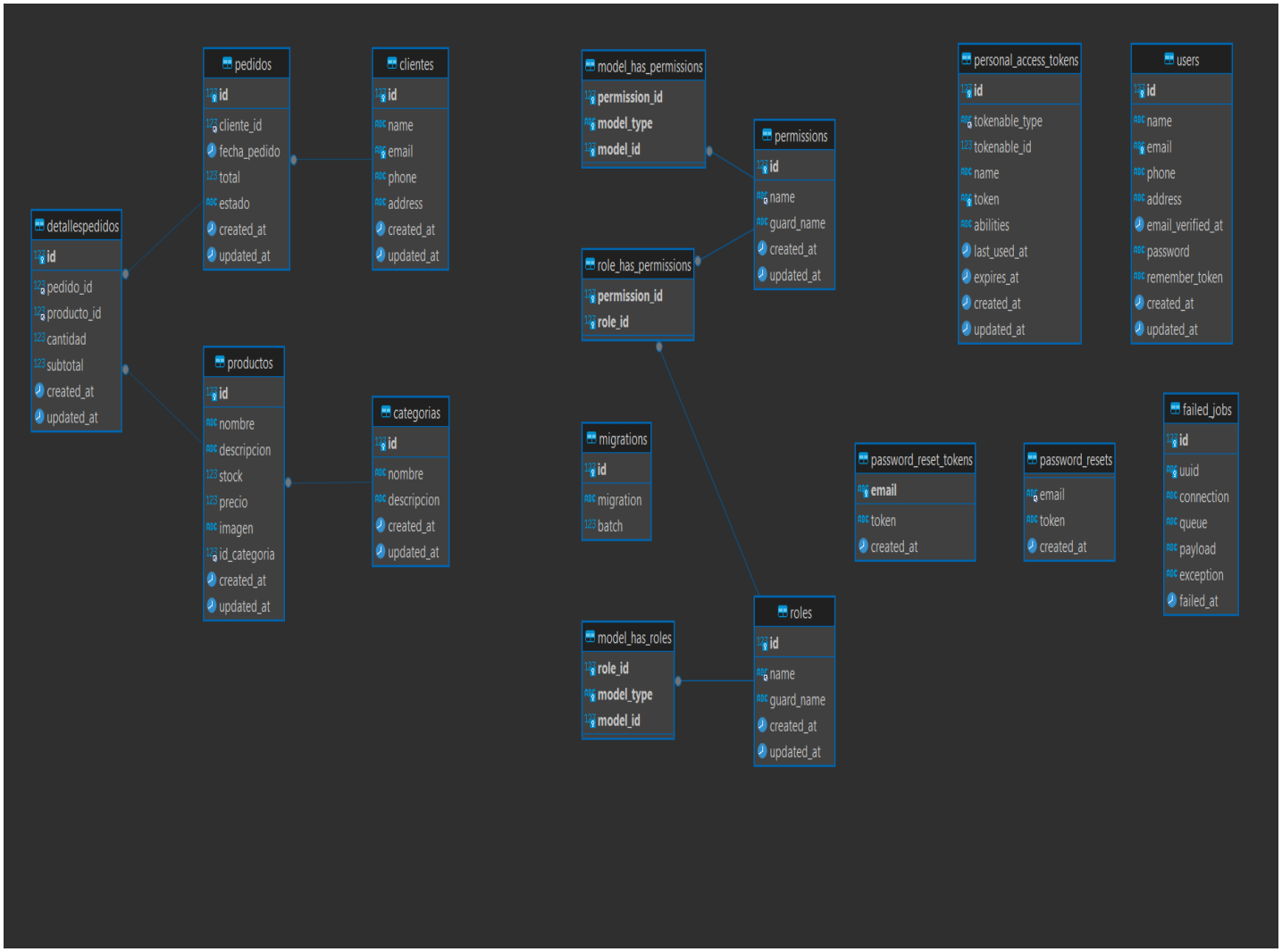


Diagrama del modelo físico de la base de datos



Alcances y limitaciones

Alcances

Los alcances del proyecto para la implementación del sitio web de promoción y gestión de pedidos en línea para "cafetería" incluyen:

Diseño y desarrollo del sitio web: Esto incluirá la selección del framework laravel, diseño y desarrollo de la interfaz de usuario, creación de secciones de contenido, integración de imágenes y otros elementos visuales, y configuración necesarios.

Implementación del sistema de gestión de pedidos: Esto incluirá la creación de una sección de pedidos en línea, configuración de los métodos de pago, la creación de una base de datos de clientes y pedidos, y la integración del sistema de gestión de pedidos con el sitio web.

Optimización del sitio web: Esto incluirá la optimización del contenido del sitio web y su estructura para mejorar su visibilidad en los motores de búsqueda, la 1. optimización de la velocidad del sitio web y la integración de herramientas de análisis y seguimiento del sitio web.

Pruebas y puesta en marcha: Una vez completados los pasos anteriores, se realizarán pruebas exhaustivas del sitio web y el sistema de gestión de pedidos para asegurar su funcionalidad y usabilidad. Luego, se procederá a la puesta en marcha del sitio web y se realizarán pruebas adicionales para asegurarse de que todo funcione correctamente.

Capacitación y soporte: Después de la puesta en marcha del sitio web y el sistema de gestión de pedidos, se brindará capacitación al personal de "cafetería" encargado de gestionar y actualizar el sitio web. Además, se proporcionará soporte técnico continuo para garantizar que el sitio web y el sistema de gestión de pedidos estén siempre en funcionamiento.

En resumen, los alcances del proyecto para la implementación del sitio web de promoción y gestión de pedidos en línea para "cafetería" incluyen el diseño y desarrollo del sitio web, implementación del sistema de gestión de pedidos, optimización del sitio web, pruebas y puesta en marcha, y capacitación y soporte técnico continuo.

Limitaciones

Las limitaciones que se deben tener en cuenta para la implementación del sitio web de promoción y gestión de pedidos en línea para "cafetería" son:

Recursos limitados: La empresa cuenta con recursos limitados, tanto financieros como de personal, lo que podría limitar la implementación del proyecto y la capacidad de respuesta para resolver problemas técnicos.

Requerimientos de los clientes: El proyecto debe tener en cuenta las necesidades y expectativas de los clientes de "cafetería" para asegurarse de que el sitio web y el sistema de gestión de pedidos sean lo suficientemente flexibles para satisfacer las necesidades de los clientes.

Disponibilidad de tiempo: El proyecto debe completarse dentro de un plazo establecido y puede haber limitaciones de tiempo para la capacitación del personal y la implementación del sitio web y el sistema de gestión de pedidos.

Experiencia técnica limitada: El personal de "cafetería" puede tener limitaciones en términos de experiencia técnica para el diseño y desarrollo de un sitio web y la implementación de un sistema de gestión de pedidos, lo que podría generar limitaciones en el alcance del proyecto.

Dependencia del proveedor de hosting: La implementación del sitio web de "cafetería" dependerá del proveedor de hosting seleccionado, lo que podría generar limitaciones en la configuración y personalización del sitio web y la integración del sistema de gestión de pedido.

Conclusión

En resumen, "cafetería" destaca por su compromiso con la calidad del café, la eficiencia operativa y la seguridad de datos gracias a la elección de Laravel en PHP como plataforma de desarrollo. El sitio web proporciona una experiencia de usuario optimizada, garantiza la protección de la información sensible y ofrece un enfoque versátil para la gestión de contenido.

Sin embargo, para el éxito continuo de "cafetería", es fundamental mantener el sitio web actualizado, seguir recopilando retroalimentación de los usuarios y adaptarse a las cambiantes necesidades de los clientes. La combinación de pasión por el café y excelencia tecnológica promete una experiencia única para los amantes del café y, con la continua atención a la calidad y la innovación, "cafetería" tiene el potencial de ser un destino destacado en la industria de la cafetería.

Recomendaciones

- Café de calidad: asegúrese de ofrecer café de alta calidad como base de su menú. Invierte en buenos granos y equipos de elaboración de cerveza.
- Menú variado: proporcione un menú diverso que satisfaga diferentes preferencias, incluidos varios tipos de café, tés y tal vez algunos bocadillos o pasteles.
- Ambiente acogedor: cree una atmósfera cómoda y acogedora con asientos acogedores, colores cálidos y tal vez algo de música de fondo suave
- Personal capacitado: capacite a su personal para que conozca las diferentes variedades de café y métodos de preparación. Un personal bien informado puede mejorar la experiencia del cliente
- Bebidas exclusivas: introduzca bebidas exclusivas o mezclas especiales que distingan a su cafetería y brinden a los clientes una razón para seguir regresando.
- WiFi y estaciones de carga: Ofrezca WiFi y estaciones de carga gratis para atraer clientes que buscan un lugar para trabajar o socializar.
- Especiales de temporada: mantenga las cosas frescas ofreciendo especiales de temporada o promociones por tiempo limitado para atraer a los clientes a probar algo nuevo.
- Presencia en las redes sociales: utilice las redes sociales para mostrar sus ofertas, interactuar con los clientes y promover cualquier evento o promoción.

Referencias

code, v. s. (n.d.). *https://code.visualstudio.com/Download*.

compras, c. h. (n.d.). *https://www.bing.com/videos/riverview/relatedvideo?q=como+hacer+una+pagina+web+con+un+carrito+en+laravel+5.1&mid=848F4A904F6DB1B11562848F4A904F6DB1B11562*.

descargar, c. (n.d.). *https://getcomposer.org/*.

jetstream, c. d. (n.d.). *https://www.bing.com/videos/riverview/relatedvideo?&q=c%c3%b3mo+instalar+laravel+jetstream&&mid=C7C13CD7E75E7C5FF9D6C7C13CD7E75E7C5FF9D6&&FORM=VRDGAR*.

laragon. (n.d.). *https://laragon.org/*.

node.js. (n.d.). *https://nodejs.org* .

ANEXOS

Manual de programador

Introducción

Laravel es un popular framework de desarrollo de aplicaciones web en PHP que sigue el paradigma de arquitectura MVC (Model-View-Controller). Ofrece una estructura organizada y elegante para construir aplicaciones web robustas y escalables. Por otro lado, Visual Studio Code (VS Code) es un editor de código ligero y potente desarrollado por Microsoft, que ha ganado popularidad entre los desarrolladores debido a su simplicidad, extensibilidad y soporte para una variedad de lenguajes de programación.

La combinación de Laravel y Visual Studio Code proporciona un entorno de desarrollo eficiente y productivo. Aquí hay algunas razones por las que muchos desarrolladores eligen esta combinación:

Integración sencilla: Visual Studio Code ofrece una integración fluida con Laravel, lo que facilita la escritura de código, la depuración y la gestión de proyectos. Además, la comunidad de desarrolladores ha creado extensiones específicas para Laravel, mejorando aún más la experiencia de desarrollo.

Soporte para PHP y Blade: Visual Studio Code tiene soporte nativo para PHP, el lenguaje en el que está escrito Laravel. También ofrece resaltado de sintaxis y autocompletado para Blade, el motor de plantillas de Laravel. Esto facilita la escritura y comprensión del código.

Extensiones y Complementos: La extensibilidad de Visual Studio Code permite la instalación de una amplia variedad de extensiones y complementos que mejoran la productividad del desarrollador. En el caso de Laravel, existen extensiones específicas que ofrecen funcionalidades como la generación rápida de código, la navegación por el proyecto y la administración de bases de datos.

Depuración y Testing: VS Code proporciona herramientas de depuración integradas que facilitan la identificación y corrección de errores en el código Laravel. Además, se pueden integrar fácilmente herramientas de prueba, lo que simplifica la escritura y ejecución de pruebas unitarias y funcionales.

Control de Versiones: Visual Studio Code incluye soporte integrado para sistemas de control de versiones como Git. Esto facilita la colaboración en equipo y la gestión de versiones en proyectos Laravel.

Terminal Integrada: VS Code cuenta con una terminal integrada que permite ejecutar comandos de Artisan de Laravel directamente desde el editor, simplificando tareas como la migración de la base de datos, la generación de controladores y la instalación de dependencias.

Instalación del entorno de desarrollo

Instalación de Visual Studio Code en Windows:

Descargar Visual Studio Code: Visita el sitio oficial de Visual Studio Code en https://code.visualstudio.com/.

Haz clic en el botón "Download" para descargar el instalador.

Ejecutar el Instalador: Una vez descargado, ejecuta el archivo de instalación (normalmente tiene un nombre como VSCodeSetup-version.exe).

Configurar Opciones de Instalación: Sigue las instrucciones del instalador.

Puedes elegir las opciones predeterminadas o personalizar la instalación según tus preferencias.

Iniciar Visual Studio Code: Después de completar la instalación, ejecuta Visual Studio Code desde el menú de inicio o el escritorio.

Configuración Adicional (Opcional):

Instalar Extensiones: Visual Studio Code es altamente personalizable a través de extensiones. Puedes instalar extensiones para admitir diferentes lenguajes de programación, temas y herramientas adicionales.

Abre Visual Studio Code, ve a la pestaña de extensiones (icono de cuadrados en la barra lateral izquierda) y busca las extensiones que desees.

Configurar Ajustes: Explora las configuraciones de Visual Studio Code según tus preferencias. Puedes acceder a ellas a través de File > Preferences en Windows.

Integración con el Terminal: Visual Studio Code tiene integración con terminales. Puedes configurar tu terminal preferido en las configuraciones.

Instalación de Visual Studio Code en Otros Sistemas Operativos:

Para macOS: Descarga el archivo .dmg desde el sitio web y sigue las instrucciones de instalación.

Para Linux: Descarga el archivo .deb o .rpm según tu distribución y sigue las instrucciones de instalación. También puedes instalarlo a través de la línea de comandos con comandos específicos para tu distribución.

Fundamentos del lenguaje de programación

1. PHP: Laravel utiliza PHP como lenguaje de programación principal. Familiarízate con la sintaxis de PHP, las variables, los arrays, las funciones y otros conceptos básicos. Aprende sobre la programación orientada a objetos (OOP) en PHP, ya que Laravel hace un uso extensivo de este paradigma.
2. Composer: Composer es una herramienta de administración de dependencias para PHP que Laravel utiliza para gestionar sus propias dependencias y bibliotecas.
1. Aprende a utilizar Composer para instalar y gestionar paquetes de PHP.
2. Eloquent ORM: Laravel utiliza Eloquent, un ORM (Mapeo Objeto-Relacional), para interactuar con la base de datos. Debes comprender los conceptos de modelado de datos, relaciones y consultas utilizando Eloquent
3. Blade Templating Engine: Laravel utiliza Blade como su motor de plantillas. Aprende las sintaxis y características de Blade para la creación de vistas en tu aplicación.
4. Routing: Entiende cómo funcionan las rutas en Laravel. Define rutas en el archivo web.php para establecer la relación entre las URL y los controladores.
5. Controladores: Los controladores en Laravel gestionan la lógica de la aplicación. Aprende a crear y utilizar controladores para organizar tu código de manera eficiente.
6. Middleware: Laravel utiliza middleware para filtrar las solicitudes HTTP antes de que lleguen a la aplicación. Comprende cómo se aplican y cómo puedes crear middleware personalizado.
7. Migraciones y Semillas: Laravel proporciona migraciones para gestionar esquemas de base de datos y semillas para poblar datos iniciales. Aprende a utilizar migraciones y semillas para gestionar la base de datos de tu aplicación.
8. Autenticación y Autorización: Laravel ofrece funciones integradas para la autenticación y autorización. Comprende cómo implementar la autenticación de usuarios y cómo controlar el acceso a las diferentes partes de tu aplicación.
9. Estructura del Proyecto: Familiarízate con la estructura de directorios de un proyecto Laravel. Comprende la función de cada directorio, como app, public, resources, y otros.
10. Testing: Laravel incluye herramientas de prueba integradas. Aprende a escribir pruebas para tus aplicaciones para garantizar su estabilidad y fiabilidad.

Código Fuente

Configuración de la base de datos:

Edita el archivo **.env** para configurar la conexión a tu base de datos. Laravel soporta varios sistemas de gestión de bases de datos, como MySQL, PostgreSQL, SQLite y SQL Server.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=proyecto
DB_USERNAME=root
DB_PASSWORD=
```

Migraciones y controladores:

Laravel utiliza migraciones para gestionar la estructura de la base de datos y seeders para poblarla con datos de prueba. Ejecuta los siguientes comandos:

php artisan make:migration NombreMigration

php artisan make:controller NombreController

- Esta migración se utiliza para crear la tabla de usuarios en la base de datos.

```
public function up(): void
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->string('phone')->nullable();
            $table->string('address')->nullable();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }
```

- Esta migración se utiliza para inicializar roles para usuarios en tu aplicación.

```
public function up(): void
    {
        $role1 = Role::create(['name' => 'admin']);
```

```
        $role2 = Role::create(['name' => 'cliente']);

        // Obtener los primeros cuatro usuarios registrados
        $users = User::limit(4)->get();

        // Asignar el rol de 'admin' a los usuarios
        foreach ($users as $user) {
            $user->assignRole($role1);
        }
    }
```

- Se utiliza para crear las tablas necesarias para implementar un sistema de control de acceso y roles utilizando el paquete **spatie/laravel-permission**. Este paquete permite gestionar permisos y roles de manera sencilla en una aplicación Laravel.

```
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        $teams = config('permission.teams');
        $tableNames = config('permission.table_names');
        $columnNames = config('permission.column_names');
        $pivotRole = $columnNames['role_pivot_key'] ?? 'role_id';
        $pivotPermission = $columnNames['permission_pivot_key'] ??
'permission_id';

        if (empty($tableNames)) {
            throw new \Exception('Error: config/permission.php not
loaded. Run [php artisan config:clear] and try again.');
        }
        if ($teams && empty($columnNames['team_foreign_key'] ?? null)) {
            throw new \Exception('Error: team_foreign_key on
config/permission.php not loaded. Run [php artisan config:clear] and try
again.');
        }

        Schema::create($tableNames['permissions'], function (Blueprint
$table) {
            $table->bigIncrements('id'); // permission id
            $table->string('name');       // For MySQL 8.0 use
string('name', 125);
```

```
            $table->string('guard_name'); // For MySQL 8.0 use
string('guard_name', 125);
            $table->timestamps();

            $table->unique(['name', 'guard_name']);
        });

        Schema::create($tableNames['roles'], function (Blueprint $table)
use ($teams, $columnNames) {
            $table->bigIncrements('id'); // role id
            if ($teams || config('permission.testing')) { //
permission.testing is a fix for sqlite testing
                $table-
>unsignedBigInteger($columnNames['team_foreign_key'])->nullable();
                $table->index($columnNames['team_foreign_key'],
'roles_team_foreign_key_index');
            }
            $table->string('name');       // For MySQL 8.0 use
string('name', 125);
            $table->string('guard_name'); // For MySQL 8.0 use
string('guard_name', 125);
            $table->timestamps();
            if ($teams || config('permission.testing')) {
                $table->unique([$columnNames['team_foreign_key'], 'name',
'guard_name']);
            } else {
                $table->unique(['name', 'guard_name']);
            }
        });

        Schema::create($tableNames['model_has_permissions'], function
(Blueprint $table) use ($tableNames, $columnNames, $pivotPermission,
$teams) {
            $table->unsignedBigInteger($pivotPermission);

            $table->string('model_type');
            $table->unsignedBigInteger($columnNames['model_morph_key']);
            $table->index([$columnNames['model_morph_key'],
'model_type'], 'model_has_permissions_model_id_model_type_index');

            $table->foreign($pivotPermission)
                ->references('id') // permission id
                ->on($tableNames['permissions'])
                ->onDelete('cascade');
            if ($teams) {
```

```
                $table-
>unsignedBigInteger($columnNames['team_foreign_key']);
                $table->index($columnNames['team_foreign_key'],
'model_has_permissions_team_foreign_key_index');

                $table->primary([$columnNames['team_foreign_key'],
$pivotPermission, $columnNames['model_morph_key'], 'model_type'],
                    'model_has_permissions_permission_model_type_primary'
);
            } else {
                $table->primary([$pivotPermission,
$columnNames['model_morph_key'], 'model_type'],
                    'model_has_permissions_permission_model_type_primary'
);
            }

        });

        Schema::create($tableNames['model_has_roles'], function
(Blueprint $table) use ($tableNames, $columnNames, $pivotRole, $teams) {
            $table->unsignedBigInteger($pivotRole);

            $table->string('model_type');
            $table->unsignedBigInteger($columnNames['model_morph_key']);
            $table->index([$columnNames['model_morph_key'],
'model_type'], 'model_has_roles_model_id_model_type_index');

            $table->foreign($pivotRole)
                ->references('id') // role id
                ->on($tableNames['roles'])
                ->onDelete('cascade');
            if ($teams) {
                $table-
>unsignedBigInteger($columnNames['team_foreign_key']);
                $table->index($columnNames['team_foreign_key'],
'model_has_roles_team_foreign_key_index');

                $table->primary([$columnNames['team_foreign_key'],
$pivotRole, $columnNames['model_morph_key'], 'model_type'],
                    'model_has_roles_role_model_type_primary');
            } else {
                $table->primary([$pivotRole,
$columnNames['model_morph_key'], 'model_type'],
                    'model_has_roles_role_model_type_primary');
            }
```

```
        });

        Schema::create($tableNames['role_has_permissions'], function
(Blueprint $table) use ($tableNames, $pivotRole, $pivotPermission) {
            $table->unsignedBigInteger($pivotPermission);
            $table->unsignedBigInteger($pivotRole);

            $table->foreign($pivotPermission)
                ->references('id') // permission id
                ->on($tableNames['permissions'])
                ->onDelete('cascade');

            $table->foreign($pivotRole)
                ->references('id') // role id
                ->on($tableNames['roles'])
                ->onDelete('cascade');

            $table->primary([$pivotPermission, $pivotRole],
'role_has_permissions_permission_id_role_id_primary');
        });

        app('cache')
            ->store(config('permission.cache.store') != 'default' ?
config('permission.cache.store') : null)
            ->forget(config('permission.cache.key'));
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        $tableNames = config('permission.table_names');

        if (empty($tableNames)) {
            throw new \Exception('Error: config/permission.php not found
and defaults could not be merged. Please publish the package
configuration before proceeding, or drop the tables manually.');
        }

        Schema::drop($tableNames['role_has_permissions']);
        Schema::drop($tableNames['model_has_roles']);
        Schema::drop($tableNames['model_has_permissions']);
        Schema::drop($tableNames['roles']);
        Schema::drop($tableNames['permissions']);
```

```
    }
};
```

- Esta migración se utiliza para crear la tabla clientes.

```
public function up(): void
    {
        Schema::create('clientes', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->string('phone')->nullable();
            $table->string('address')->nullable();
            $table->timestamps();
        }); }
```

- Esta migración se utiliza para crear la tabla pedidos.

```
public function up(): void
    {
        Schema::create('pedidos', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('cliente_id');
            $table->date('fecha_pedido');
            $table->decimal('total', 8, 2);
            $table->foreign('cliente_id')->references('id')->on('clientes')-
>onDelete('cascade');
            $table->string('estado');
            $table->timestamps();
        });
    }
```

- Esta migración se utiliza para crear la tabla categorias.

```
 public function up(): void
    {
        Schema::create('categorias', function (Blueprint $table) {
            $table->id();
            $table->string('nombre');
            $table->text('descripcion');
            $table->timestamps();
        });
    }
```

- Esta migración se utiliza para crear la tabla productos.

```
 public function up(): void
    {
        Schema::create('productos', function (Blueprint $table) {
            $table->id();
            $table->string('nombre');
            $table->text('descripcion');
            $table->integer('stock');
            $table->decimal('precio', 8, 2);
            $table->string('imagen');
            $table->unsignedBigInteger('id_categoria');
            $table->timestamps();
            $table->foreign('id_categoria')->references('id')-
>on('categorias');
        });
    }
```

- Esta migración se utiliza para crear la tabla detallespedidos.

```
public function up(): void
    {
        Schema::create('detallespedidos', function (Blueprint $table) {
            $table->id();
            $table->foreignId('pedido_id')->constrained('pedidos');
            $table->foreignId('producto_id')->constrained('productos');
            $table->integer('cantidad');
            $table->decimal('subtotal', 8, 2);
            $table->timestamps();
        });
    }
```

Controladores:

- Este controlador está diseñado para mostrar el carrito de compras, la página de pago (checkout), y procesar la información del pedido.

```
class CartController extends Controller
{
    public function mostrarCart()
    {
        $productos = Producto::all();
        return view('cart', compact('productos'));
    }
```

```
    public function mostrarCheckout()
    {
        return view('checkout');
    }

    public function finalizarPedido(Request $request)
    {
        $carritoJSON = $request->carrito;
        $carrito = json_decode($carritoJSON, true);
    }
}
```

- Este controlador es, específicamente para manejar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) de las categorías, con restricciones de acceso para roles de administrador.

```
class CategoriaController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $categorias = Categoria::paginate();

        return view('categoria.index', compact('categorias'))
            ->with('i', (request()->input('page', 1) - 1) * $categorias-
>perPage());
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
```

```
        $categoria = new Categoria();
        return view('categoria.create', compact('categoria'));
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        request()->validate(Categoria::$rules);

        $categoria = Categoria::create($request->all());

        return redirect()->route('categorias.index')
            ->with('success', 'Categoria created successfully.');
    }

    /**
     * Display the specified resource.
     *
     * @param  int $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $categoria = Categoria::find($id);

        return view('categoria.show', compact('categoria'));
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int $id
     * @return \Illuminate\Http\Response
     */
```

```php
    public function edit($id)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $categoria = Categoria::find($id);

        return view('categoria.edit', compact('categoria'));
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request $request
     * @param  Categoria $categoria
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, Categoria $categoria)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        request()->validate(Categoria::$rules);

        $categoria->update($request->all());

        return redirect()->route('categorias.index')
            ->with('success', 'Categoria updated successfully');
    }

    /**
     * @param int $id
     * @return \Illuminate\Http\RedirectResponse
     * @throws \Exception
     */
    public function destroy($id)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $categoria = Categoria::find($id)->delete();

        return redirect()->route('categorias.index')
            ->with('success', 'Categoria deleted successfully');
    }
```

```
    public function pdf()
    {
        $categoria = Categoria::paginate();

        $pdf = PDF::loadView('categoria.pdf',['categorias'=>$categoria]);
        return $pdf->stream();
    }
}
```

- Este controlador es, específicamente para manejar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) de cliente, con restricciones de acceso para roles de administrador.

```
class ClienteController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $clientes = Cliente::paginate();

        return view('cliente.index', compact('clientes'))
            ->with('i', (request()->input('page', 1) - 1) * $clientes-
>perPage());
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $cliente = new Cliente();
        return view('cliente.create', compact('cliente'));
```

```
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        request()->validate(Cliente::$rules);

        $cliente = Cliente::create($request->all());

        return redirect()->route('clientes.index')
            ->with('success', 'Cliente created successfully.');
    }

    /**
     * Display the specified resource.
     *
     * @param  int $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $cliente = Cliente::find($id);

        return view('cliente.show', compact('cliente'));
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
```

```
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $cliente = Cliente::find($id);

        return view('cliente.edit', compact('cliente'));
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request $request
     * @param  Cliente $cliente
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, Cliente $cliente)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        request()->validate(Cliente::$rules);

        $cliente->update($request->all());

        return redirect()->route('clientes.index')
            ->with('success', 'Cliente updated successfully');
    }

    /**
     * @param int $id
     * @return \Illuminate\Http\RedirectResponse
     * @throws \Exception
     */
    public function destroy($id)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $cliente = Cliente::find($id)->delete();

        return redirect()->route('clientes.index')
            ->with('success', 'Cliente deleted successfully');
    }
}
```

- Este controlador es, específicamente para manejar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) de Detallespedido, con restricciones de acceso para roles de administrador.

```
class DetallespedidoController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $detallespedidos = Detallespedido::paginate();

        return view('detallespedido.index', compact('detallespedidos'))
            ->with('i', (request()->input('page', 1) - 1) *
$detallespedidos->perPage());
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $detallespedido = new Detallespedido();
        return view('detallespedido.create', compact('detallespedido'));
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        if (!auth()->user()->hasRole('admin')) {
```

```
            abort(403, 'Acceso no autorizado');
        }
        request()->validate(Detallespedido::$rules);

        $detallespedido = Detallespedido::create($request->all());

        return redirect()->route('detallespedidos.index')
            ->with('success', 'Detallespedido created successfully.');
    }

    /**
     * Display the specified resource.
     *
     * @param  int $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $detallespedido = Detallespedido::find($id);

        return view('detallespedido.show', compact('detallespedido'));
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $detallespedido = Detallespedido::find($id);

        return view('detallespedido.edit', compact('detallespedido'));
    }

    /**
     * Update the specified resource in storage.
     *
```

```
     * @param  \Illuminate\Http\Request $request
     * @param  Detallespedido $detallespedido
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, Detallespedido $detallespedido)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        request()->validate(Detallespedido::$rules);

        $detallespedido->update($request->all());

        return redirect()->route('detallespedidos.index')
            ->with('success', 'Detallespedido updated successfully');
    }

    /**
     * @param int $id
     * @return \Illuminate\Http\RedirectResponse
     * @throws \Exception
     */
    public function destroy($id)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $detallespedido = Detallespedido::find($id)->delete();

        return redirect()->route('detallespedidos.index')
            ->with('success', 'Detallespedido deleted successfully');
    }
}
```

- Este controlador asegura que todas las acciones requieran autenticación y proporciona diferentes vistas de inicio según el rol del usuario autenticado. Además, incluye una acción específica (adminOnlyAction) que solo está permitida para usuarios con el rol 'admin', y muestra un mensaje de error 403 si un usuario no autorizado intenta acceder a esta acción.

```
class HomeController extends Controller
{
    /**
     * Create a new controller instance.
     *
     * @return void
```

```
     */
    public function __construct()
    {
        $this->middleware('auth');
    }

    /**
     * Show the application dashboard.
     *
     * @return \Illuminate\Contracts\Support\Renderable
     */
    public function index()
    {
        if (auth()->user()->hasRole('admin')) {
            return view('home_admin');
        } else {
            return view('home_user');
        }
    }

    public function adminOnlyAction()
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }

        // Resto del código para la acción
    }
}

```

- Este controlador es relacionado con la integración de PayPal para procesar pagos en una aplicación web.

```
lass PaypalController extends Controller
{
    public function createpaypal()
    {
        // return view('paypal_view');
        return view('checkout');
    }


    public function processPaypal(Request $request)
    {
```

```
        // Obtener el usuario autenticado
        $user = Auth::user();

        $provider = new PayPalClient;
        $provider->setApiCredentials(config('paypal'));
        $paypalToken = $provider->getAccessToken();

        $total = $request->input('total');
        // Almacenar el total en una variable de sesión
        $request->session()->put('total', $total);

        $response = $provider->createOrder([
            "intent" => "CAPTURE",
            "application_context" => [
                "return_url" => route('processSuccess'),
                "cancel_url" => route('processCancel'),
            ],
            "purchase_units" => [
                0 => [
                    "amount" => [
                        "currency_code" => "USD",
                        "value" => $total
                    ]
                ]
            ]
        ]);

        if (isset($response['id']) && $response['id'] != null) {

            // redirect to approve href
            foreach ($response['links'] as $links) {
                if ($links['rel'] == 'approve') {
                    return redirect()->away($links['href']);
                }
            }

            return redirect()
                ->route('createpaypal')
                ->with('error', 'Something went wrong.');
        } else {
            return redirect()
                ->route('createpaypal')
                ->with('error', $response['message'] ?? 'Something went
wrong.');
        }
```

```
    }


    public function processSuccess(Request $request)
    {
        $user = Auth::user();
        $userId = $user->id;

        $cliente_id = $userId;
        $fecha_pedido = now();

        // Obtener el total de la variable de sesión
        $total = $request->session()->get('total');
        $estado = 0;

        // Create a new Pedido record
        $pedido = Pedido::create([
            'cliente_id' => $cliente_id,
            'fecha_pedido' => $fecha_pedido,
            'total' => $total,
            'estado' => $estado,
        ]);

        // Obtener el ID del pedido recién creado
        $pedidoId = $pedido->id;

        // Obtener el carrito de la solicitud
        $carrito = json_decode($request->input('carrito'), true);

        // Verificar si $carrito es null antes de iterar sobre él
        if ($carrito) {
            // Iterar sobre los productos en el carrito
            foreach ($carrito as $producto) {
                $productoId = $producto['id'];
                $cantidad = $producto['cantidad'];
                $subtotal = $producto['precio'] * $cantidad;

                // Crear un nuevo registro en la tabla detallespedidos
                Detallespedido::create([
                    'pedido_id' => $pedidoId,
                    'producto_id' => $productoId,
                    'cantidad' => $cantidad,
                    'subtotal' => $subtotal,
                ]);
            }
```

```
        }

        $provider = new PayPalClient;
        $provider->setApiCredentials(config('paypal'));
        $provider->getAccessToken();
        $response = $provider->capturePaymentOrder($request['token']);

        if (isset($response['status']) && $response['status'] ==
'COMPLETED') {

            return redirect()
                ->route('createpaypal')
                ->with('success', 'Transaction complete.');
        } else {
            return redirect()
                ->route('createpaypal')
                ->with('error', $response['message'] ?? 'Something went
wrong.');
        }
    }


    public function processCancel(Request $request)
    {
        return redirect()
            ->route('createpaypal')
            ->with('error', $response['message'] ?? 'You have canceled the
transaction.');
    }
}
```

- Este controlador es, específicamente para manejar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) de pedidoController, con restricciones de acceso para roles de administrador.

```
class PedidoController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
```

```
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $pedidos = Pedido::paginate();

        return view('pedido.index', compact('pedidos'))
            ->with('i', (request()->input('page', 1) - 1) * $pedidos-
>perPage());
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $pedido = new Pedido();
        $clientes = Cliente::pluck('name', 'id');
        return view('pedido.create', compact('pedido', 'clientes'));
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        request()->validate(Pedido::$rules);

        $pedido = Pedido::create($request->all());

        return redirect()->route('pedidos.index')
            ->with('success', 'Pedido created successfully.');
    }

    /**
```

```
     * Display the specified resource.
     *
     * @param  int $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $pedido = Pedido::find($id);

        return view('pedido.show', compact('pedido'));
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $pedido = Pedido::find($id);
        $clientes = Cliente::pluck('name', 'id');
        return view('pedido.edit', compact('pedido', 'clientes'));
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request $request
     * @param  Pedido $pedido
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, Pedido $pedido)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        request()->validate(Pedido::$rules);
```

```
        $pedido->update($request->all());

        return redirect()->route('pedidos.index')
            ->with('success', 'Pedido updated successfully');
    }

    /**
     * @param int $id
     * @return \Illuminate\Http\RedirectResponse
     * @throws \Exception
     */
    public function destroy($id)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $pedido = Pedido::find($id)->delete();

        return redirect()->route('pedidos.index')
            ->with('success', 'Pedido deleted successfully');
    }
}
```

- Este controlador es, específicamente para manejar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) de productoController, con restricciones de acceso para roles de administrador.

```
class ProductoController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $productos = Producto::paginate();

        return view('producto.index', compact('productos'))
            ->with('i', (request()->input('page', 1) - 1) * $productos-
>perPage());
```

```
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $producto = new Producto();
        return view('producto.create', compact('producto'));
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        request()->validate(Producto::$rules);

        $producto = Producto::create($request->all());

        if ($request->hasFile('imagen')) {
            $imagen = $request->file('imagen');
            $nombreImagen = time() . '_' . $imagen->getClientOriginalName();
            $imagen->move(public_path('images'), $nombreImagen);
            $producto->imagen = 'images/' . $nombreImagen;
            $producto->save();
        }

        return redirect()->route('productos.index')
            ->with('success', 'Producto created successfully.');
    }

    /**
     * Display the specified resource.
```

```
     *
     * @param  int $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        // if (!auth()->user()->hasRole('admin')) {
        //     abort(403, 'Acceso no autorizado');
        // }
        $producto = Producto::find($id);

        return view('producto.show', compact('producto'));
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $producto = Producto::find($id);

        return view('producto.edit', compact('producto'));
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request $request
     * @param  Producto $producto
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, Producto $producto)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        request()->validate(Producto::$rules);

        $producto->update($request->all());
```

```
        if ($request->hasFile('imagen')) {
            $imagen = $request->file('imagen');
            $nombreImagen = time() . '_' . $imagen->getClientOriginalName();

            if ($producto->imagen) {
                $rutaImagen = public_path('images/' . $producto->imagen);

                if (file_exists($rutaImagen)) {
                    unlink($rutaImagen);
                }
            }

            $imagen->move(public_path('images'), $nombreImagen);
            $producto->imagen = 'images/' . $nombreImagen;
            $producto->save();
        }

        return redirect()->route('productos.index')
            ->with('success', 'Producto updated successfully');
    }

    /**
     * @param int $id
     * @return \Illuminate\Http\RedirectResponse
     * @throws \Exception
     */
    public function destroy($id)
    {
        if (!auth()->user()->hasRole('admin')) {
            abort(403, 'Acceso no autorizado');
        }
        $producto = Producto::find($id)->delete();

        return redirect()->route('productos.index')
            ->with('success', 'Producto deleted successfully');
    }

    public function pdf()
    {
        $producto = Producto::paginate();

        $pdf = PDF::loadView('producto.pdf', ['productos' => $producto]);
        return $pdf->stream();
    }
```

```
}
```

Rutas

Define las rutas para las diversas acciones y recursos en tu aplicación Laravel.

```
Auth::routes();

// Route::get('productos/pdf', [ProductoController::class, 'pdf'])-
>name('productos.pdf');
Route::get('/home', [App\Http\Controllers\HomeController::class, 'index'])-
>name('home');
Route::get('/home_admin', [App\Http\Controllers\HomeController::class,
'index'])->name('home_admin');
Route::get('/home_user', [App\Http\Controllers\HomeController::class,
'index'])->name('home_user');

Route::get('productos/pdf', [App\Http\Controllers\ProductoController::class,
'pdf'])->name('productos.pdf');
Route::get('categorias/pdf',
[App\Http\Controllers\CategoriaController::class, 'pdf'])-
>name('categorias.pdf');

Route::resource('clientes', \App\Http\Controllers\ClienteController::class);
Route::resource('pedidos', \App\Http\Controllers\PedidoController::class);
Route::resource('productos',
\App\Http\Controllers\ProductoController::class);
Route::resource('categorias',
\App\Http\Controllers\CategoriaController::class);
Route::resource('detallespedidos',
\App\Http\Controllers\DetallespedidoController::class);

Route::get('/cart', function () {
    return view('cart.index');
});

// Rutas del carro
Route::get('/cart', [CartController::class, 'mostrarCart'])->name('cart');
Route::get('/checkout', [CartController::class, 'mostrarCheckout'])-
>name('checkout');

// Rutas de paypal
Route::get('createpaypal', [PaypalController::class, 'createpaypal'])-
>name('createpaypal');
```

```
Route::get('processPaypal', [PaypalController::class, 'processPaypal'])-
>name('processPaypal');
Route::get('processSuccess', [PaypalController::class, 'processSuccess'])-
>name('processSuccess');
Route::get('processCancel', [PaypalController::class, 'processCancel'])-
>name('processCancel');
```

Views

Vista cart.blade.php

- Este código crea un botón que al hacer clic ejecutará la función verCarrito() y muestra un ícono de carrito con un contador en forma de círculo para indicar la cantidad de elementos en el carrito.

```
    <div class="container">
        <div class="row">
            <div class="col-md-12">
                <button class="btn btn-primary" onclick="verCarrito()">
                    Ver Carrito
                    <svg xmlns="http://www.w3.org/2000/svg" width="16"
height="16" fill="currentColor" class="bi bi-cart"
                        viewBox="0 0 16 16">
                        <path d="M0 1.5A.5.5 0 0 1 .5 1H2a.5.5 0 0 1
.485.379L2.89 3H14.5a.5.5 0 0 1 .491.592l-1.5 8A.5.5 0 0 1 13 12H4a.5.5 0 0
1-.491-.408L2.01 3.607 1.61 2H.5a.5.5 0 0 1-.5-.5zM3.102 4l1.313
7h8.17l1.313-7H3.102zM5 12a2 2 0 1 0 0 4 2 2 0
                                    0 0 0-4zm7 0a2 2 0 1 0 0 4 2 2 0 0 0 0-
4zm-7 1a1 1 0 1 1 0 2 1 1 0 0 1 0-2zm7 0a1 1 0 1 1 0 2 1 1 0 0 1
                                    0-2z">
                        </path>
                    </svg>
                    <span class="badge bg-danger text-white"
id="carritoCantidad">0</span>
                </button>
            </div>
        </div>
    </div>
```

- Este código crea un diseño responsivo utilizando Bootstrap para mostrar un catálogo de productos con información detallada y la capacidad de agregar productos al carrito.

```
<div class="container">
        <div class="row justify-content-center">
            <div class="col-md-12">
                <div class="welcome-message">
                    <h1 class="text-center">Catálogo de venta</h1>
                    <p class="text-center">Elige lo que deseas
comprar</p>
                </div>
            </div>
        </div>
    </div>
    <div class="container">
        <div class="row">
            @foreach ($productos as $producto)
                <div class="col-md-4">
                    <div class="card mb-4">
                        <div class="card-body">
                            <h5 class="card-title">Nombre: {{ $producto-
>nombre }}</h5>
                            <input type="text" id="nombre{{ $producto->id
}}" disabled
                                value="{{ $producto->nombre }}
                                    " hidden>
                            <img src="{{ asset($producto->imagen) }}"
alt="{{ $producto->nombre }}"
                                style="max-width: 100px;">
                            <p class="card-text">Disponible: {{
$producto->stock }}</p>
                            <input type="number" id="stock{{ $producto-
>id }}" disabled value="{{ $producto->stock }}"
                                hidden>
                            <p class="card-text">Precio: ${{ $producto-
>precio }}</p>
                            <input type="text" id="precio{{ $producto->id
}}" disabled
                                value="{{ $producto->precio }}
                                    " hidden>
                            <label for="cantidad">Cantidad:</label>
                            <input type="number" id="cantidad{{
$producto->id }}" value="1" min="1">
                            <br>
                            <div class="text-center">
```

```
                                {!! QrCode::size(75)->generate($producto-
>nombre) !!}
                            </div>
                            <br>
                            <div class="text-center">
                                <button class="btn btn-primary"
onclick="agregarAlCarrito({{ $producto->id }})">
                                    Agregar al carrito
                                </button>
                            </div>
                        </div>
                    </div>
                </div>
            @endforeach
        </div>
    </div>
```

- Este script proporciona la funcionalidad para agregar productos al carrito, gestionar la cantidad de productos y redirigir al usuario a la página de checkout. Además, actualiza dinámicamente el contador de la cantidad total del carrito.

```
<script>
        var productosSeleccionados =
JSON.parse(localStorage.getItem('carrito')) || [];

        function agregarAlCarrito(id) {
            var cantidadInput = document.getElementById('cantidad' + id);
            var stockInput = document.getElementById('stock' + id);
            var precioInput = document.getElementById('precio' + id);
            var nombreInput = document.getElementById('nombre' + id);
            var cantidad = parseInt(cantidadInput.value);
            var cantidadPedido = parseInt(stockInput.value);
            var precio = parseFloat(precioInput.value);
            var nombre = nombreInput.value;

            if (cantidad > 0 && cantidad <= cantidadPedido) {
                // Busca si el producto ya está en el carrito
                var productoEnCarrito =
productosSeleccionados.find(function(item) {
                    return item.id === id;
                });

                if (productoEnCarrito) {
```

```
                    // Si el producto ya está en el carrito, aumenta la
cantidad
                    productoEnCarrito.cantidad += cantidad;
                } else {
                    // Si el producto no está en el carrito, agrégalo
                    productosSeleccionados.push({
                        id: id,
                        nombre: nombre,
                        precio: precio,
                        cantidad: cantidad
                    });
                }

                // Guarda el carrito en el localStorage
                localStorage.setItem('carrito',
JSON.stringify(productosSeleccionados));

                // Desactiva el botón "Agregar al carrito"
                var agregarBtn =
document.querySelector(`[onclick="agregarAlCarrito(${id})"]`);
                agregarBtn.disabled = true;
                cantidadInput.disabled = true;

                // Actualiza la cantidad en el carrito
                actualizarCantidadCarrito();
            } else {
                alert('Puedes agregar desde 1 producto y tiene que ser
menor que la cantidad disponible');
            }
        }

        function actualizarCantidadCarrito() {
            var carrito = JSON.parse(localStorage.getItem('carrito')) ||
[];
            var totalCantidad = 0;
            carrito.forEach(function(item) {
                totalCantidad += item.cantidad;
            });
            document.getElementById('carritoCantidad').textContent =
totalCantidad;
        }

        function verCarrito() {
            window.location.href = '{{ route('checkout') }}';
        }
```

```
        actualizarCantidadCarrito();
    </script>
@endsection
```

Checkout.blade.php

En resumen, este código representa la estructura de la página de checkout de compra, donde se muestra un resumen del carrito, enlaces para navegar y un botón para finalizar la compra y pagar con PayPal.

```
@extends('layouts.app')

@section('content')
<link rel="stylesheet" href="{{ asset('css/checkout.css') }}"> <!-- Enlace
al archivo CSS externo -->

    <div class="container">
        <h1 class="text-center">Checkout de Compra</h1>
        <div class="row justify-content-center">
            <div class="col-md-8">
                <h2 class="text-center">Resumen del Carrito</h2>
                <ul id="carrito-container" class="list-group">
                    <!-- Aquí se mostrarán los productos del carrito -->
                </ul>
            </div>
            <div class="container">
                <div class="row justify-content-center">
                    <div class="col-md-8 text-center">
                        <p class="text-center">
                            <a href="{{ route('cart') }}" class="btn btn-
primary">Ir al Carrito</a>
                        </p>

                        <button onclick="finalizarCompra()" class="btn btn-
primary">Finalizar Compra y Pagar con PayPal</button>
                    </div>
                </div>
            </div>
```

- Este script de JavaScript está diseñado para trabajar con la página de checkout de compra y se integra con PayPal para procesar el pago.

```
<script>
        // Cargar el carrito del LocalStorage
        var carrito = JSON.parse(localStorage.getItem('carrito'))
|| [];

        // Obtener el elemento donde se mostrarán los productos del
carrito
        var carritoContainer = document.getElementById('carrito-
container');

        // Función para mostrar los productos del carrito
        function mostrarProductosDelCarrito() {
            if (carrito.length === 0) {
                carritoContainer.innerHTML = '<p>El carrito está
vacío</p>';
            } else {
                carritoContainer.innerHTML = ''; // Limpiar
contenido previo

                carrito.forEach(function(producto, index) {
                    var listItem = document.createElement('li');
                    listItem.className = 'list-group-item d-flex
justify-content-between align-items-center lh-sm';

                    var nombreProducto =
document.createElement('div');
                    nombreProducto.innerHTML = '<h4 class="my-0">'
+ producto.nombre +
                        '</h4><small class="text-body-
secondary"><h5 class="my-0">Cantidad: ' + producto.cantidad +
                        '</h5></small>';

                    var precioProducto =
document.createElement('span');
                    precioProducto.className = 'text-body-
secondary';
                    precioProducto.innerHTML = '<h5 class="my-0">$'
+ (producto.precio * producto.cantidad).toFixed(2) +
                        '</h5>';

                    var eliminarButton =
document.createElement('button');
                    eliminarButton.className = 'eliminar-button';
```

```
                    eliminarButton.innerHTML = 'Eliminar de la
lista';
                    eliminarButton.onclick = function() {
                        eliminarProducto(index);
                    };

                    listItem.appendChild(nombreProducto);
                    listItem.appendChild(precioProducto);
                    listItem.appendChild(eliminarButton);

                    carritoContainer.appendChild(listItem);
                });

                var totalVenta = carrito.reduce(function(total,
producto) {
                    return total + producto.precio *
producto.cantidad;
                }, 0);

                var totalVentaRedondeado = totalVenta.toFixed(2);

                var totalItem = document.createElement('li');
                totalItem.className = 'list-group-item d-flex
justify-content-between';
                totalItem.innerHTML = '<span><h4 class="my-0">Total
(USD)</h4></span><strong><h4 class="my-0">$' +
                    totalVentaRedondeado + '</h4></strong>';

                carritoContainer.appendChild(totalItem);
            }
        }

        function eliminarProducto(index) {
            carrito.splice(index, 1);
            localStorage.setItem('carrito',
JSON.stringify(carrito));
            mostrarProductosDelCarrito();
        }

        // Función para finalizar la compra y pagar con PayPal
        function finalizarCompra() {
            // Construir la URL de PayPal con los parámetros
necesarios
            var usuarioId = "{{ auth()->user()->id }}";
            var token = "{{ csrf_token() }}";
```

```
            var carritoJson = JSON.stringify(carrito);
            var total = calcularTotal().toFixed(2);;

            var paypalUrl = "{{ route('processPaypal') }}" +
                "?usuario=" + usuarioId +
                "&_token=" + token +
                "&carrito=" + encodeURIComponent(carritoJson) +
                "&total=" + total;

            // Limpiar el carrito en el LocalStorage
            localStorage.removeItem('carrito');

            // Redirigir a PayPal
            window.location.href = paypalUrl;
        }

        // Cargar y mostrar productos del carrito al cargar la
página
        mostrarProductosDelCarrito();

        // Función para calcular el total del carrito
        function calcularTotal() {
            var totalVenta = carrito.reduce(function(total,
producto) {
                return total + producto.precio * producto.cantidad;
            }, 0);

            return totalVenta;
        }
    </script>
@endsection
```

- Este código Blade se utiliza para la vista de inicio de sesión de un usuario en una cafetería en línea.

```
@extends('layouts.app')

@section('content')
    <link rel="stylesheet" href="{{ asset('css/homeuser.css') }}">
<!-- Enlace al archivo CSS externo -->
    <br><br>

   <center> <div class="card" style="width: 15rem;">
         <img src="{{ asset('imagenes/coffee-cup.png') }}" alt=""
height="200em" width="200em">
```

```
        <div class="card">
            <div class="row mb-1">
            </div>
        </div>
    </div></center>
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-md-12">
                <div class="welcome-message">
                    <h1 class="text-center">¡Bienvenido a nuestra
cafetria en linea!</h1>
                    <p class="text-center"><a href="{{
route('cart') }}" class="btn btn-primary" style="margin-left:
10px;">
                                Explora Nuestra Variedad
                        </a>
                    <p class="card-text">Nuestro café premium te
espera para una experiencia única.</p>
                    </p>
                    <div class="card-body">
                        <h5 class="card-title">¡Disfruta de un Café
Exquisito!</h5>


                    </div>

                </div>
            </div>


            <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">

            <a
href="https://wa.me/50370816666?text=Me%20gustaría%20saber%20el%20p
recio%20del%20coche" class="whatsapp"
                target="_blank"> <i class="fa fa-whatsapp whatsapp-
icon"></i></a>


        </div>
    </div>
@endsection

@push('styles')
```

```
@endpush
```

- Este código representa una página de inicio de sesión típica con un formulario para ingresar credenciales. La adición de estilos específicos mediante archivos CSS externos (login.css y style.css) permite personalizar la apariencia de la página.

```
@extends('layouts.app')

@section('content')
<link rel="stylesheet" href="{{ asset('css/login.css') }}"> <!--
Enlace al archivo CSS externo -->
<link rel="stylesheet" href="{{ asset('css/style.css') }}"> <!--
Enlace al archivo CSS externo -->

<br><br><br>
<div class="container">

    <div class="row justify-content-center">
        <div class="col-md-8">
            <div class="card">
                <div class="card-header">{{ __('Login') }}</div>

                <div class="card-body">
                    <form method="POST" action="{{ route('login')
}}">
                        @csrf
                        <div class="card">
                            <center> <img src="{{
asset('imagenes/coffee-cup.png') }}" alt="" height="200em"
width="200em"></center>
                            <div class="card-body">
                        <div class="row mb-3">
                            <label for="email" class="col-md-4 col-
form-label text-md-end">{{ __('Email Address') }}</label>

                            <div class="col-md-6">
                                <input id="email" type="email"
class="form-control @error('email') is-invalid @enderror"
name="email" value="{{ old('email') }}" required
autocomplete="email" autofocus>

                                @error('email')
                                    <span class="invalid-feedback"
role="alert">
```

```
                                    <strong>{{ $message
}}</strong>
                                </span>
                            @enderror
                        </div>
                    </div>

                    <div class="row mb-3">
                        <label for="password" class="col-md-4
col-form-label text-md-end">{{ __('Password') }}</label>

                        <div class="col-md-6">
                            <input id="password"
type="password" class="form-control @error('password') is-invalid
@enderror" name="password" required autocomplete="current-
password">

                            @error('password')
                                <span class="invalid-feedback"
role="alert">
                                    <strong>{{ $message
}}</strong>
                                </span>
                            @enderror
                        </div>
                    </div>



                    <div class="row mb-0">
                        <div class="col-md-8 offset-md-4">
                            <button type="submit" class="btn
btn-primary">
                                {{ __('Login') }}
                            </button>


                                </a>

                        </div>
                    </div>
                </form>
            </div>
        </div>
    </div>
```

```
    </div>
</div>
@endsection
```

- Este código representa una página de registro típica con un formulario para ingresar información del usuario. La adición de estilos específicos mediante archivos CSS.

```
@extends('layouts.app')

@section('content')
<link rel="stylesheet" href="{{ asset('css/login.css') }}"> <!--
Enlace al archivo CSS externo -->
    <link rel="stylesheet" href="{{ asset('css/style.css') }}"> <!-
- Enlace al archivo CSS externo -->
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <div class="card">
                <div class="card-header">{{ __('Register') }}</div>

                <div class="card-body">
                    <form method="POST" action="{{
route('register') }}">
                        @csrf
                        <div class="card">
                            <center> <img src="{{
asset('imagenes/coffee-cup.png') }}" alt="" height="200em"
width="200em"></center>
                             <div class="card-body">
                                <div class="row mb-3">
                        <div class="row mb-3">
                            <label for="name" class="col-md-4 col-
form-label text-md-end">{{ __('Name') }}</label>

                            <div class="col-md-6">
                                <input id="name" type="text"
class="form-control @error('name') is-invalid @enderror"
name="name" value="{{ old('name') }}" required autocomplete="name"
autofocus>

                                @error('name')
                                    <span class="invalid-feedback"
role="alert">
```

```
                                    <strong>{{ $message
}}</strong>
                                </span>
                            @enderror
                        </div>
                    </div>

                    <div class="row mb-3">
                        <label for="email" class="col-md-4 col-
form-label text-md-end">{{ __('Email Address') }}</label>

                        <div class="col-md-6">
                            <input id="email" type="email"
class="form-control @error('email') is-invalid @enderror"
name="email" value="{{ old('email') }}" required
autocomplete="email">

                            @error('email')
                                <span class="invalid-feedback"
role="alert">
                                    <strong>{{ $message
}}</strong>
                                </span>
                            @enderror
                        </div>
                    </div>
                    <div class="row mb-3">
                        <label for="phone" class="col-md-4 col-
form-label text-md-end">{{ __('Número de teléfono') }}</label>

                        <div class="col-md-6">
                            <input id="phone" type="text"
class="form-control @error('phone') is-invalid @enderror"
name="phone" value="{{ old('phone') }}" required
autocomplete="phone" autofocus>

                            @error('phone')
                                <span class="invalid-feedback"
role="alert">
                                    <strong>{{ $message
}}</strong>
                                </span>
                            @enderror
                        </div>
                    </div>
```

```
                        <div class="row mb-3">
                            <label for="address" class="col-md-4
col-form-label text-md-end">{{ __('Dirección') }}</label>

                            <div class="col-md-6">
                                <input id="address" type="text"
class="form-control @error('address') is-invalid @enderror"
name="address" value="{{ old('address') }}" required
autocomplete="address" autofocus>

                                @error('address')
                                    <span class="invalid-feedback"
role="alert">
                                        <strong>{{ $message
}}</strong>
                                    </span>
                                @enderror
                            </div>
                        </div>

                        <div class="row mb-3">
                            <label for="password" class="col-md-4
col-form-label text-md-end">{{ __('Password') }}</label>

                            <div class="col-md-6">
                                <input id="password"
type="password" class="form-control @error('password') is-invalid
@enderror" name="password" required autocomplete="new-password">

                                @error('password')
                                    <span class="invalid-feedback"
role="alert">
                                        <strong>{{ $message
}}</strong>
                                    </span>
                                @enderror
                            </div>
                        </div>

                        <div class="row mb-3">
                            <label for="password-confirm"
class="col-md-4 col-form-label text-md-end">{{ __('Confirm
Password') }}</label>
```

```
                            <div class="col-md-6">
                                <input id="password-confirm"
type="password" class="form-control" name="password_confirmation"
required autocomplete="new-password">
                            </div>
                        </div>

                        <div class="row mb-0">
                            <div class="col-md-6 offset-md-4">
                               <center> <button type="submit"
class="btn btn-primary">
                                    {{ __('Register') }}
                                </button></center>
                            </div>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
@endsection
```

Categoría: index.blade.php

Esta vista proporciona una interfaz para administrar categorías, con opciones para ver, editar, eliminar y crear nuevas categorías. El mensaje de éxito informa al usuario sobre el resultado de las acciones.

```
@extends('adminlte::page')

@section('title', 'Admin')

@section('content_header')
    <h1>Categorias</h1>
@stop

@section('content')
    <div class="container-fluid">
        <div class="row">
            <div class="col-sm-12">
                <div class="card">
                    <div class="card-header">
                        <div style="display: flex; justify-content: space-
between; align-items: center;">
```

```
                            <span id="card_title">
                                {{ __('Categoria') }}
                            </span>

                            <div class="float-right">
                                <a href="{{ route('categorias.pdf') }}"
class="btn btn-success btn-sm float-right" data-placement="right">
                                    {{ __('Generar PDF') }}
                                </a>
                            </div>

                            <div class="float-right">
                                <a href="{{ route('categorias.create') }}"
class="btn btn-primary btn-sm float-right"
                                    data-placement="left">
                                    {{ __('Create New') }}
                                </a>
                            </div>
                        </div>
                    </div>
                    @if ($message = Session::get('success'))
                        <div class="alert alert-success">
                            <p>{{ $message }}</p>
                        </div>
                    @endif
```

- Esta sección completa la estructura de la vista. La tabla muestra información sobre las categorías y proporciona botones para ver, editar y eliminar cada categoría. La paginación al final facilita la navegación entre las páginas de resultados si hay un gran número de categorías.

```
 <div class="card-body">
                        <div class="table-responsive">
                            <table class="table table-striped table-
hover">
                                <thead class="thead">
                                    <tr>
                                        <th>No</th>

                                        <th>Nombre</th>
                                        <th>Descripcion</th>

                                        <th></th>
                                    </tr>
                                </thead>
```

```
                                <tbody>
                                    @foreach ($categorias as $categoria)
                                        <tr>
                                            <td>{{ ++$i }}</td>

                                            <td>{{ $categoria->nombre
}}</td>
                                            <td>{{ $categoria-
>descripcion }}</td>

                                            <td>
                                                <form action="{{
route('categorias.destroy', $categoria->id) }}"
                                                    method="POST">
                                                    <a class="btn btn-sm
btn-primary "
                                                        href="{{
route('categorias.show', $categoria->id) }}"><i
                                                            class="fa fa-
fw fa-eye"></i> {{ __('Show') }}</a>
                                                    <a class="btn btn-sm
btn-success"
                                                        href="{{
route('categorias.edit', $categoria->id) }}"><i
                                                            class="fa fa-
fw fa-edit"></i> {{ __('Edit') }}</a>
                                                    @csrf
                                                    @method('DELETE')
                                                    <button type="submit"
class="btn btn-danger btn-sm"><i
                                                            class="fa fa-
fw fa-trash"></i> {{ __('Delete') }}</button>
                                                </form>
                                            </td>
                                        </tr>
                                    @endforeach
                                </tbody>
                            </table>
                        </div>
                    </div>
                </div>
                {!! $categorias->links() !!}
            </div>
        </div>
    </div>
```

```
@stop

@section('css')
    <link rel="stylesheet" href="/css/admin_custom.css">
@stop

@section('js')
    <script>
        console.log('Hi!');
    </script>
@stop
```

Cliente: index.blade.php

- Esta sección completa la estructura de la vista. La tabla muestra información sobre los clientes y proporciona botones para ver, editar y eliminar cada cliente.

```
@extends('adminlte::page')

@section('title', 'Admin')

@section('content_header')
    <h1>Clientes</h1>
@stop

@section('content')
<div class="container-fluid">
    <div class="row">
        <div class="col-sm-12">
            <div class="card">
                <div class="card-header">
                    <div style="display: flex; justify-content: space-
between; align-items: center;">

                        <span id="card_title">
                            {{ __('Cliente') }}
                        </span>

                         <div class="float-right">
                            <a href="{{ route('clientes.create') }}"
class="btn btn-primary btn-sm float-right"  data-placement="left">
                              {{ __('Create New') }}
                            </a>
                          </div>
```

```
                    </div>
                </div>
                @if ($message = Session::get('success'))
                    <div class="alert alert-success">
                        <p>{{ $message }}</p>
                    </div>
                @endif
```

- Esta vista proporciona una lista paginada de clientes con opciones para ver, editar y eliminar cada cliente.

```
 <div class="card-body">
                    <div class="table-responsive">
                        <table class="table table-striped table-hover">
                            <thead class="thead">
                                <tr>
                                    <th>No</th>

                                    <th>Nombre</th>
                                    <th>Email</th>
                                    <th>Telefono</th>
                                    <th>Direccion</th>

                                    <th></th>
                                </tr>
                            </thead>
                            <tbody>
                                @foreach ($clientes as $cliente)
                                    <tr>
                                        <td>{{ ++$i }}</td>

                                        <td>{{ $cliente->name }}</td>
                                        <td>{{ $cliente->email }}</td>
                                        <td>{{ $cliente->phone }}</td>
                                        <td>{{ $cliente->address }}</td>

                                        <td>
                                            <form action="{{
route('clientes.destroy',$cliente->id) }}" method="POST">
                                                <a class="btn btn-sm btn-
primary " href="{{ route('clientes.show',$cliente->id) }}"><i class="fa
fa-fw fa-eye"></i> {{ __('Show') }}</a>
                                                <a class="btn btn-sm btn-
success" href="{{ route('clientes.edit',$cliente->id) }}"><i class="fa
fa-fw fa-edit"></i> {{ __('Edit') }}</a>
```

```
                                                @csrf
                                                @method('DELETE')
                                                <button type="submit"
class="btn btn-danger btn-sm"><i class="fa fa-fw fa-trash"></i> {{
__('Delete') }}</button>
                                            </form>
                                        </td>
                                    </tr>
                                @endforeach
                            </tbody>
                        </table>
                    </div>
                </div>
            </div>
            {!! $clientes->links() !!}
        </div>
    </div>
</div>
@stop

@section('css')
    <link rel="stylesheet" href="/css/admin_custom.css">
@stop

@section('js')
    <script>
        console.log('Hi!');
    </script>
@stop
```

Detallespedido: detallespedido.blade.php

- Esta vista proporciona una interfaz para ver, editar y eliminar detalles de pedidos, utilizando el tema de administración de AdminLTE y una estructura de tabla para mostrar los datos de manera organizada.

```
@extends('adminlte::page')

@section('title', 'Admin')

@section('content_header')
    <h1>Cafeteria Admin</h1>
@stop
```

```
@section('content')
<div class="container-fluid">
    <div class="row">
        <div class="col-sm-12">
            <div class="card">
                <div class="card-header">
                    <div style="display: flex; justify-content: space-
between; align-items: center;">

                        <span id="card_title">
                            {{ __('Detallespedido') }}
                        </span>

                         <div class="float-right">
                            <a href="{{ route('detallespedidos.create') }}"
class="btn btn-primary btn-sm float-right"  data-placement="left">
                                {{ __('Create New') }}
                            </a>
                          </div>
                    </div>
                </div>
                @if ($message = Session::get('success'))
                    <div class="alert alert-success">
                        <p>{{ $message }}</p>
                    </div>
                @endif

                <div class="card-body">
                    <div class="table-responsive">
                        <table class="table table-striped table-hover">
                            <thead class="thead">
                                <tr>
                                    <th>No</th>

                                    <th>Pedido Id</th>
                                    <th>Producto Id</th>
                                    <th>Cantidad</th>
                                    <th>Subtotal</th>

                                    <th></th>
                                </tr>
                            </thead>
                            <tbody>
                                @foreach ($detallespedidos as
$detallespedido)
```

```
                                    <tr>
                                        <td>{{ ++$i }}</td>

                                        <td>{{ $detallespedido->pedido_id
}}</td>
                                        <td>{{ $detallespedido->producto_id
}}</td>
                                        <td>{{ $detallespedido->cantidad
}}</td>
                                        <td>{{ $detallespedido->subtotal
}}</td>


                                        <td>
                                            <form action="{{
route('detallespedidos.destroy',$detallespedido->id) }}" method="POST">
                                                <a class="btn btn-sm btn-
primary " href="{{ route('detallespedidos.show',$detallespedido->id) }}"><i
class="fa fa-fw fa-eye"></i> {{ __('Show') }}</a>
                                                <a class="btn btn-sm btn-
success" href="{{ route('detallespedidos.edit',$detallespedido->id) }}"><i
class="fa fa-fw fa-edit"></i> {{ __('Edit') }}</a>
                                                @csrf
                                                @method('DELETE')
                                                <button type="submit"
class="btn btn-danger btn-sm"><i class="fa fa-fw fa-trash"></i> {{
__('Delete') }}</button>
                                            </form>
                                        </td>
                                    </tr>
                                @endforeach
                            </tbody>
                        </table>
                    </div>
                </div>
            </div>
            {!! $detallespedidos->links() !!}
        </div>
    </div>
</div>
@stop

@section('css')
    <link rel="stylesheet" href="/css/admin_custom.css">
@stop
```

```
@section('js')
    <script>
        console.log('Hi!');
    </script>
@stop
```

Instalación Después de Descargar

Hacer migraciones:

php artisan migrate --path=/database/migrations/principales

Ingresar los seeders:

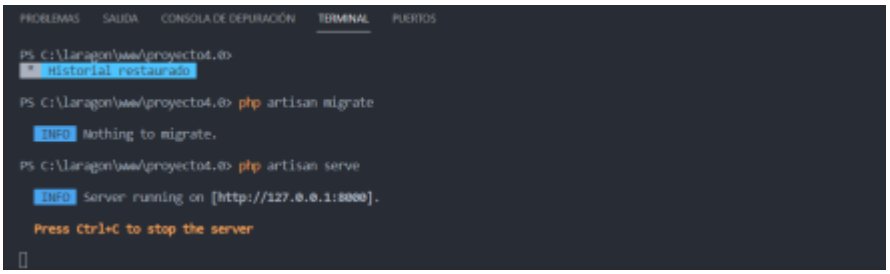
php artisan db:seed

Hacer una última migración:

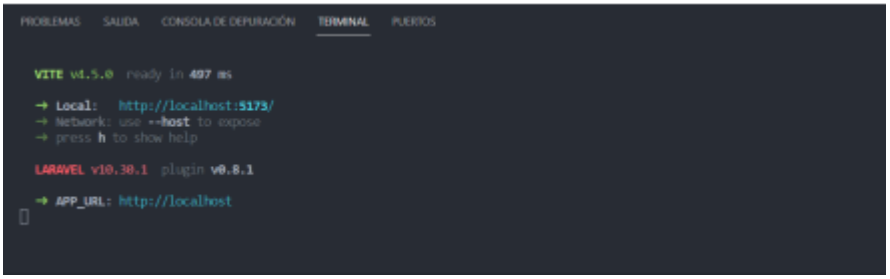
php artisan migrate

Activamos ambos comandos en distintas terminales:

php artisan serve



npm run dev



Y por último abrimos el proyecto.

Podemos iniciar sesión con sus correos de ejemplo (diana@gmail.com - cristofer@gmail.com - noemy@gmail.com)

La contraseña de todos es: 12345678

Si ingresan con los correos que les coloque ahí entraran al panel de administrador ahí pueden ver cómo va de momento el proyecto que espero arreglar estos días.

Si quieren ir a la parte de "cliente" solo hagan un nuevo usuario y ya con eso los mandara al área de clientes.

Integración de Spatie (Roles y permisos) en Laravel.

Primero accedemos a la pagina de Spatie y entramos a la sección de documentación mediante este link: https://spatie.be/docs/laravel-permission/v6/installation-laravel

La pagina nos indicará los siguientes pasos:

Instalar

1. Consulte la **Prerrequisitos** para consideraciones importantes con respecto a su **Usuario** ¡modelos!
2. Este paquete **Publica un archivo `config/permission.php`** . Si ya tiene un archivo con ese nombre, debe cambiarle el nombre o eliminarlo.
3. Puedes **Instale el paquete a través de Composer**:

```
composer require spatie/laravel-permission
```

4. Opcional: El proveedor de servicios se registrará automáticamente. O puede agregar manualmente el proveedor de servicios en su archivo: `config/app.php`

```
'providers' => [
    // ...
    Spatie\Permission\PermissionServiceProvider::class,
];
```

5. **Deberías publicar** La migración y el `config/permission.php` Archivo de configuración con:

```
php artisan vendor:publish --provider="Spatie\Permission\PermissionServiceProvider"
```

Y, por último, realizamos estos pasos:

7. **Borra la caché de configuración**. Este paquete requiere acceso a la configuración para ejecutar migraciones. Si ha estado almacenando configuraciones en caché localmente, borre la caché de configuración con cualquiera de estos comandos: `permission`

```
php artisan optimize:clear
# or
php artisan config:clear
```

8. **Ejecución de las migraciones**: Una vez que se hayan publicado y configurado la configuración y la migración, puede crear las tablas para este paquete ejecutando:

```
php artisan migrate
```

9. **Agregue el rasgo necesario a su modelo de usuario**:

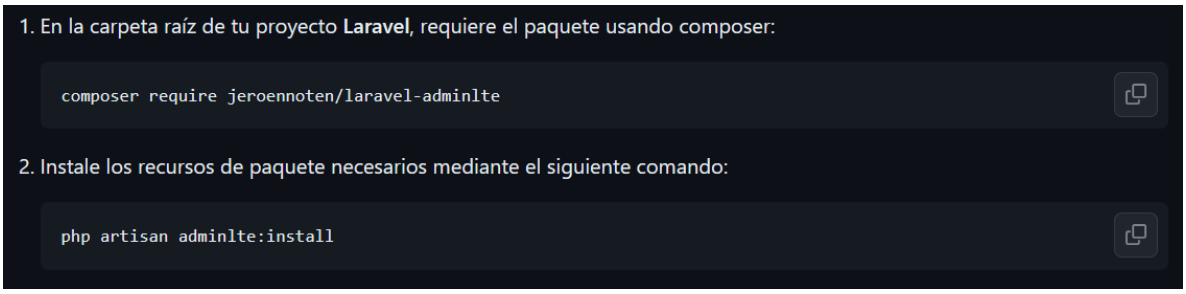
```
// The User model requires this trait
use HasRoles;
```

Con esto, ya podemos usar los roles y permisos en nuestro proyecto de Laravel, en la documentación de la pagina tienes mas información de como usar Spatie

Integración de panel administrativo (AdminLTE)

Primero accedemos al GitHub de AdminLTE: [Installation · jeroennoten/Laravel-AdminLTE Wiki · GitHub]

Luego procedemos con la instalación de este:

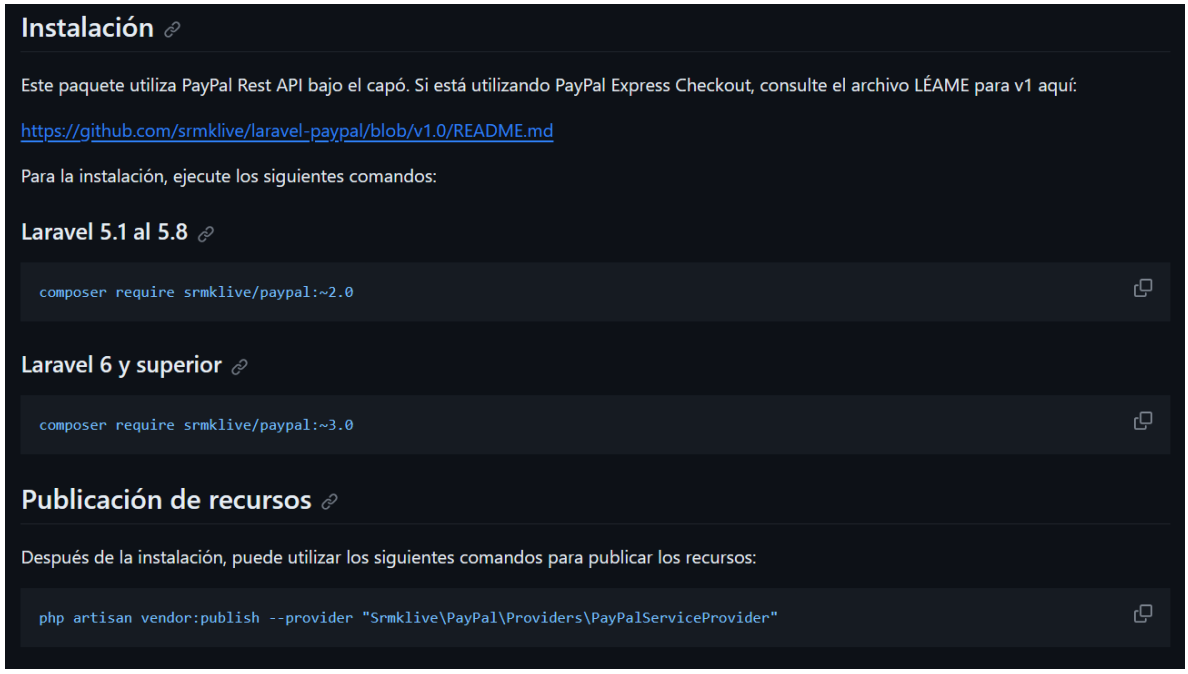


Con esto, ya tenemos instalado nuestro panel, para darle uso, en el GitHub tenemos más información sobre ello.

Integración de PayPal.

Primeramente, accedemos al GitHub de srmklive: [laravel-paypal/docs/source/installation.rst at documentation · srmklive/laravel-paypal · GitHub]

Posteriormente, procedemos con la instalación.



Archivo de configuración

El archivo de configuración **PayPal.php** se encuentra en la carpeta **config**. A continuación se detalla su contenido cuando se publicó:

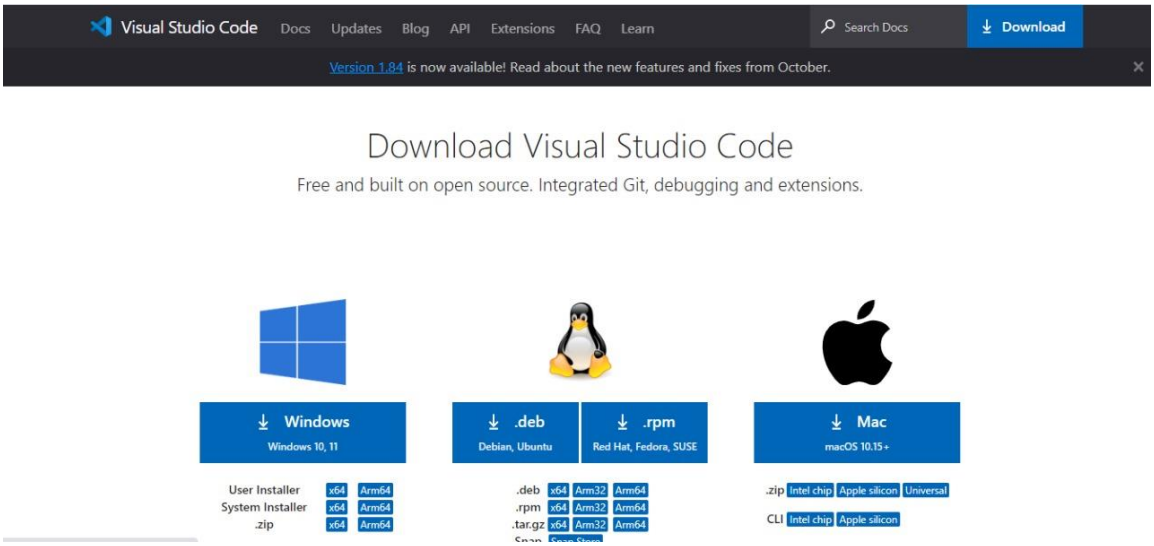
```
return [
    'mode'    => env('PAYPAL_MODE', 'sandbox'), // Can only be 'sandbox' Or 'live'. If empty or invalid, 'live' will be used.
    'sandbox' => [
        'client_id'         => env('PAYPAL_SANDBOX_CLIENT_ID', ''),
        'client_secret'     => env('PAYPAL_SANDBOX_CLIENT_SECRET', ''),
        'app_id'            => 'APP-80W284485P519543T',
    ],
    'live' => [
        'client_id'         => env('PAYPAL_LIVE_CLIENT_ID', ''),
        'client_secret'     => env('PAYPAL_LIVE_CLIENT_SECRET', ''),
        'app_id'            => '',
    ],

    'payment_action' => env('PAYPAL_PAYMENT_ACTION', 'Sale'), // Can only be 'Sale', 'Authorization' or 'Order'
    'currency'       => env('PAYPAL_CURRENCY', 'USD'),
    'notify_url'     => env('PAYPAL_NOTIFY_URL', ''), // Change this accordingly for your application.
    'locale'         => env('PAYPAL_LOCALE', 'en_US'), // force gateway language  i.e. it_IT, es_ES, en_US ... (for express che
    'validate_ssl'   => env('PAYPAL_VALIDATE_SSL', true), // Validate SSL when creating api client.
];
```

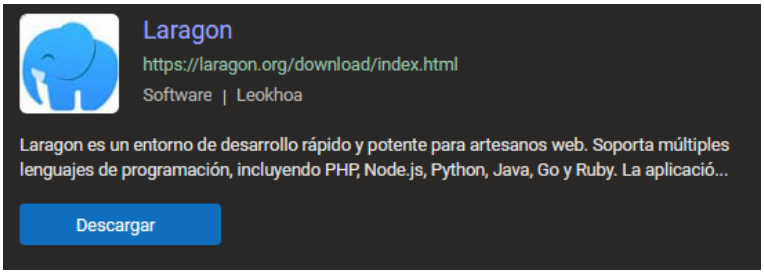
Eso sería lo básico, luego, para implementarlo, en la documentación de GitHub tenemos más información.

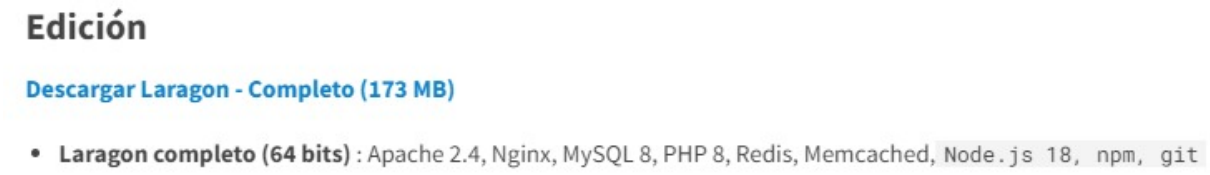
Manual de usuario

Lo primero que debemos de hacer tener instalado un editor de código en nuestro caso utilizaremos Visual Studio Code a continuación para poder instalarlo

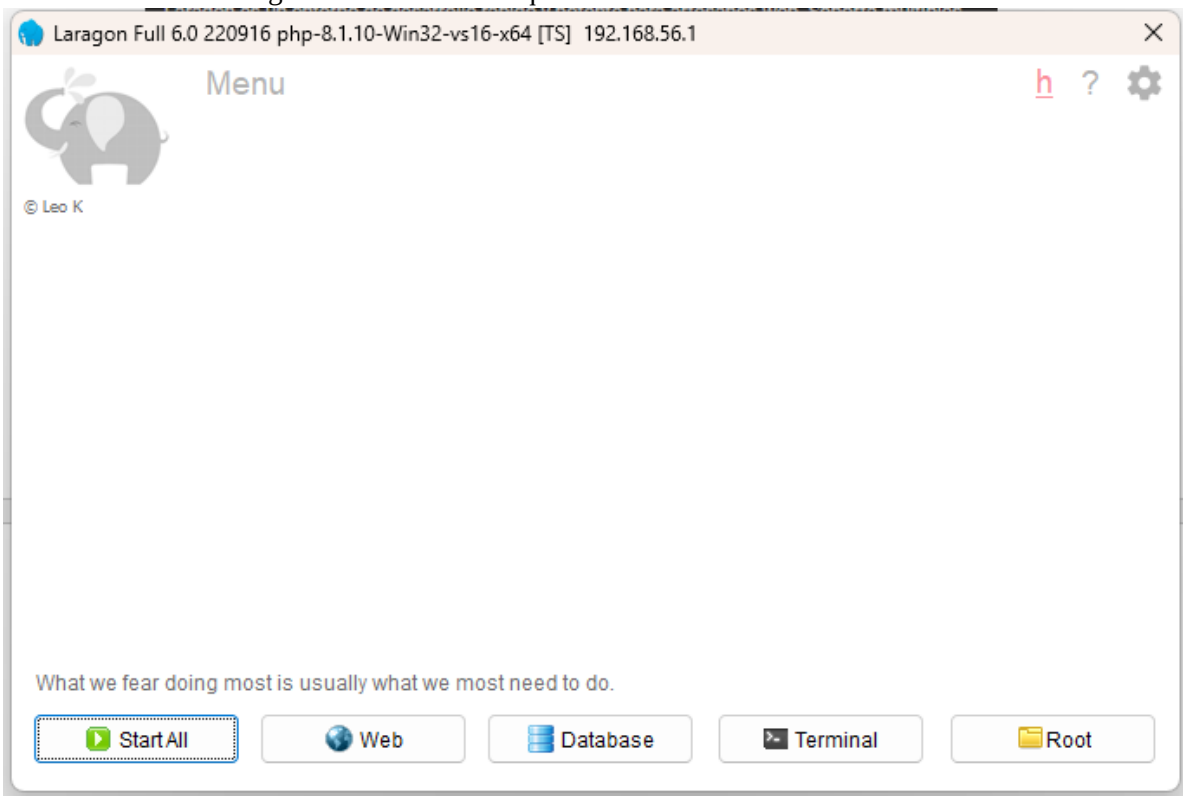


Después de haber instalado visual studio code el usuario tendrá que descargar he instalar laragon 10 ya que con esta versión esta desarrollado nuestro sitio web.



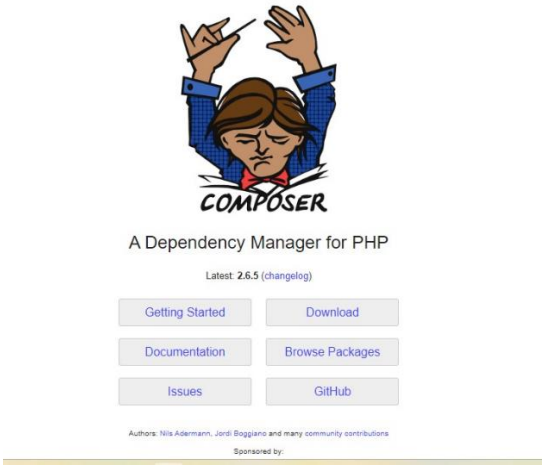


Como se inicia laragon le da en el botón que dice start all



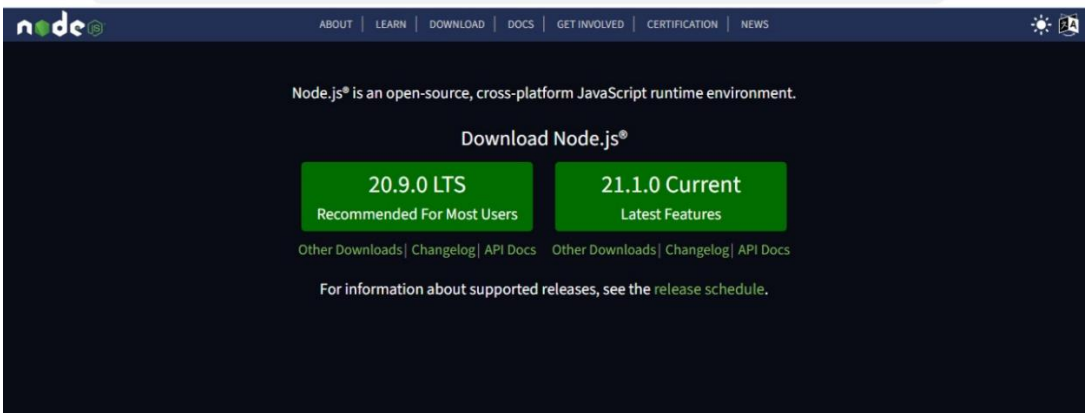
Después de haber descargado laragon tiene que descargar php en su última versión.

Composer es un sistema de gestión de paquetes para programar en PHP el cual provee los formatos estándar necesarios para manejar dependencias.

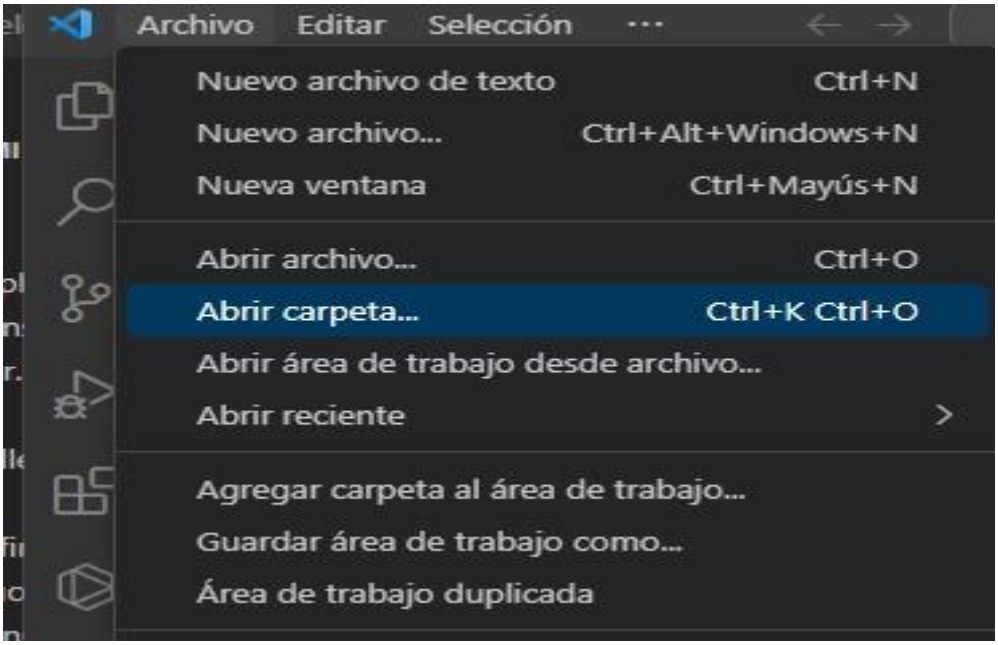


El usuario también deberá descargar e instalar node.js Lenguaje de programación

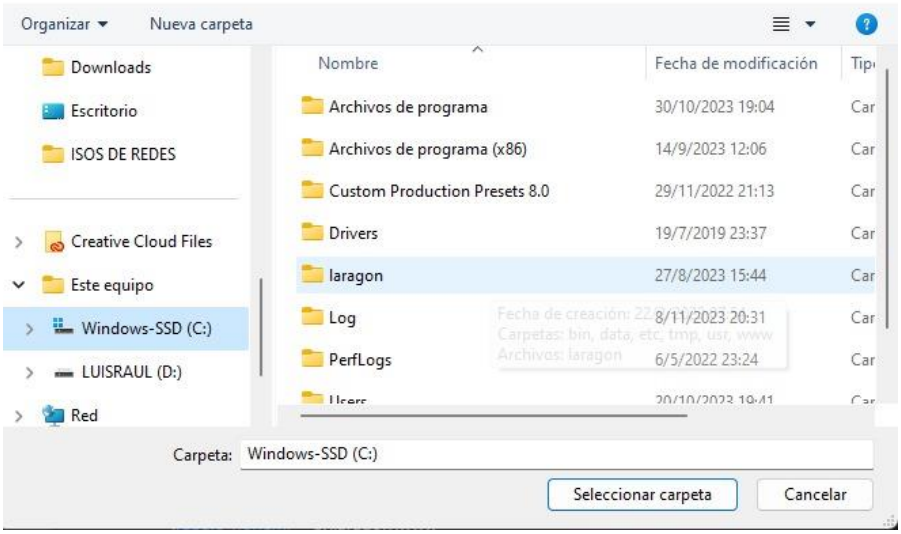
Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript.



Cuando tengamos nuestro editor de código ya instalado, lo abrimos y seleccionamos la carpeta de la cafetería

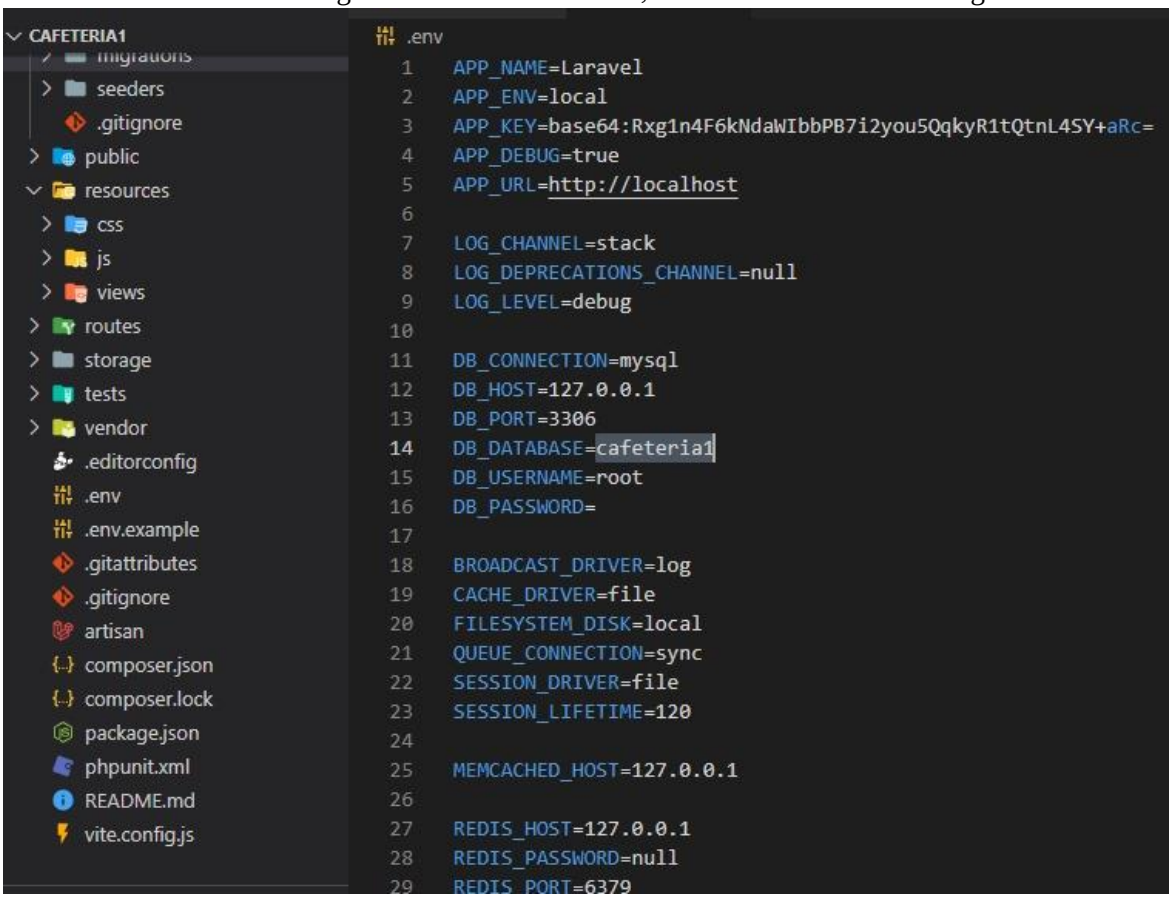


El proyecto se tendrá que guardar en la carpeta de laragon, dentro de www y ay estarán nuestro proyecto



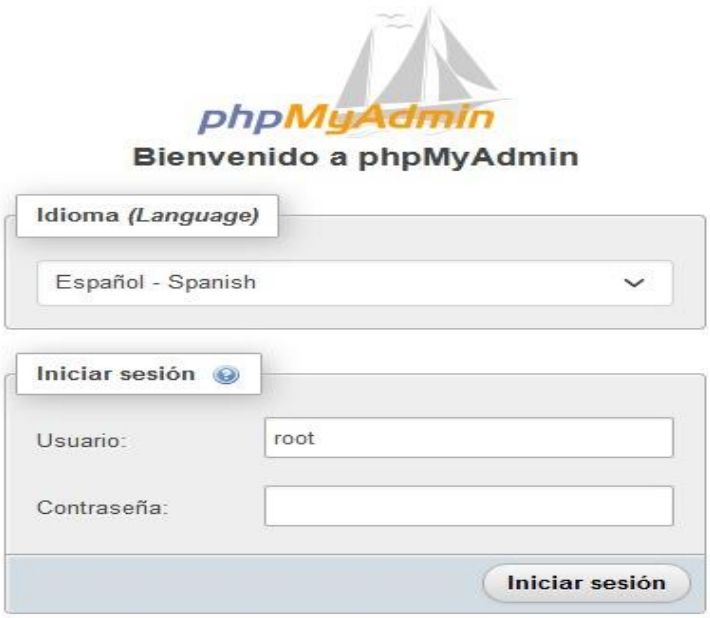
Seleccionamos la carpeta y la abrimos

En nuestro editor de código abrimos el archivo env, vemos donde aremos la migración

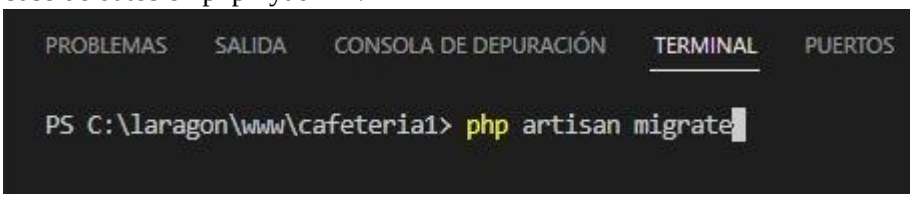


Para abrir nuestra base de datos nos vamos a database desde laragon

En nuestra base de datos ingresamos con usuario root solamente

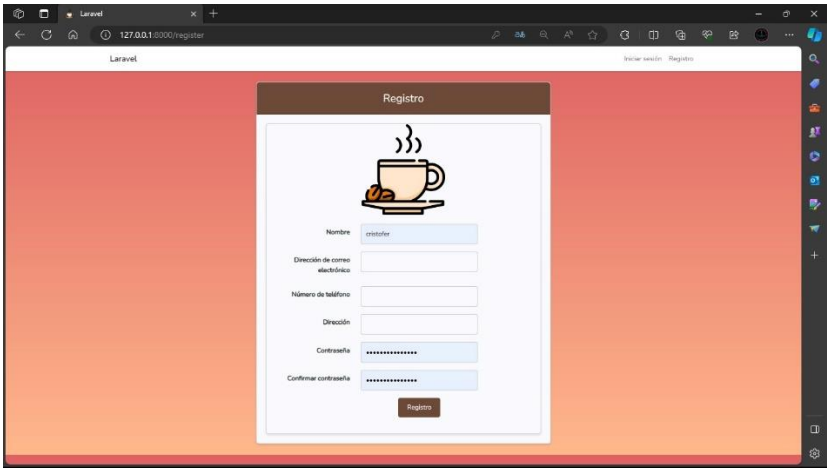


En la terminar de visual studio code, hacemos la migración a la base de datos ella sola crea una base de datos en phpmyadmin.

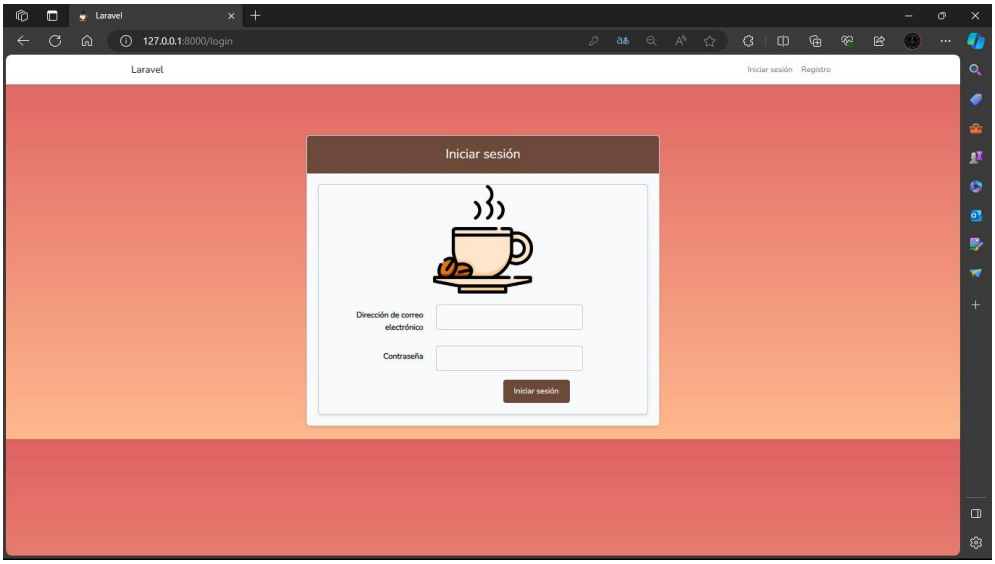


Con php artisan serve encendemos nuestro servidor y Podemos ingresar a nuestro proyecto

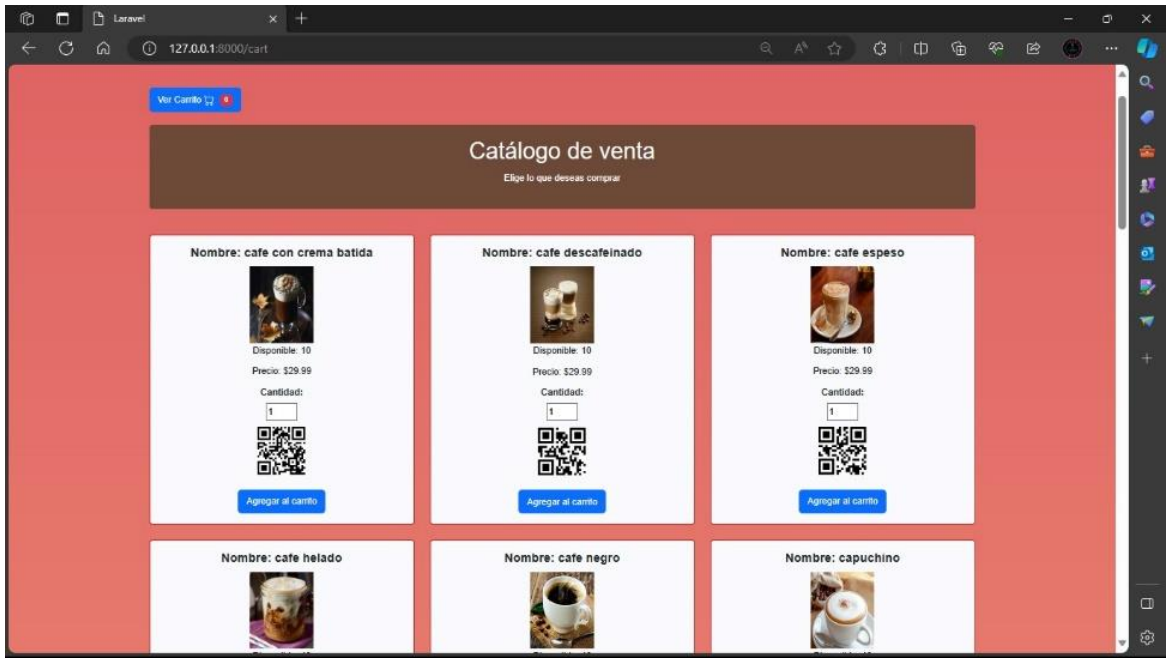
El usuario tendrá que registrarse para poder ingresar a la plataforma

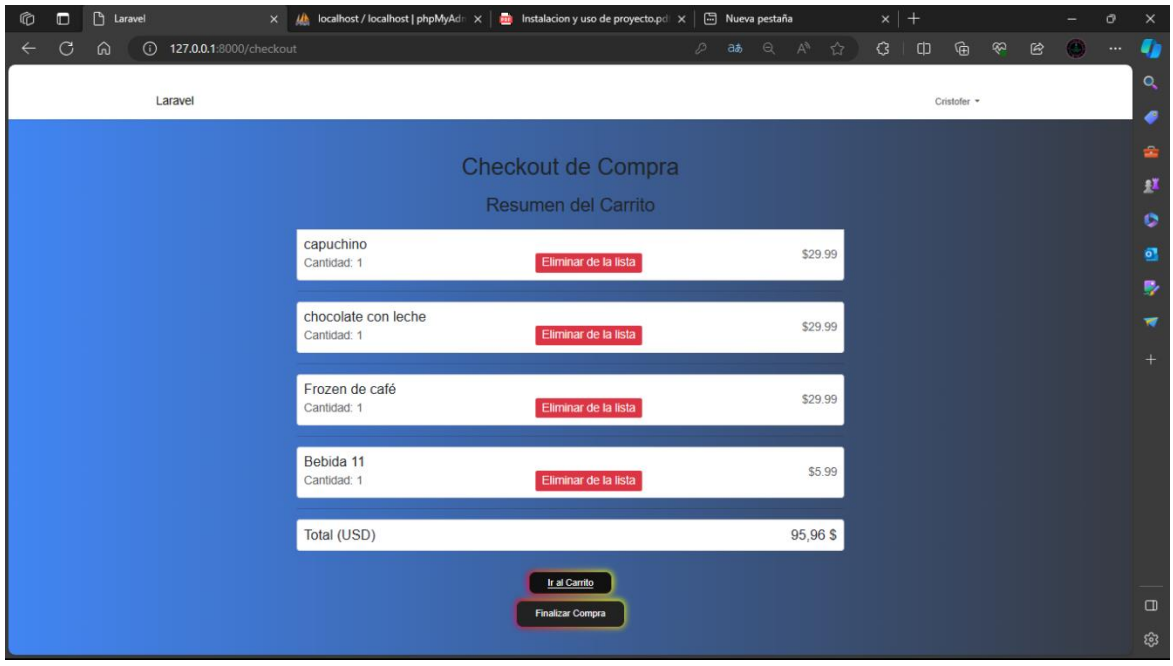


Después de todo este proceso el usuario tendrá que registrase con un correo electrónico y una contraseña para poder ingresar a la plataforma del sitio web de cafetería.

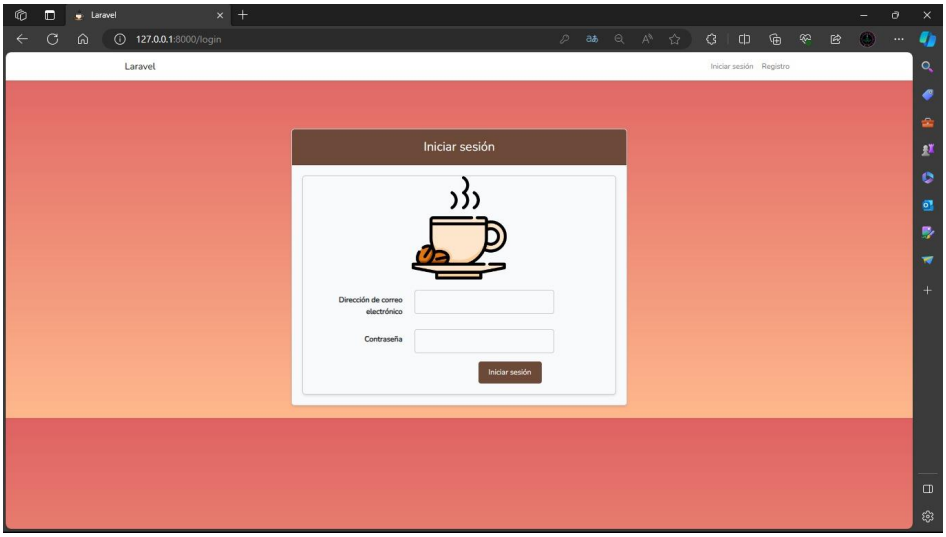


Después de haber visto nuestro menú el usuario podrá hacer su compra a nuestra cafetería por medio de un carrito de compras donde el usuario podrá seleccionar el postre o café de su preferencia y la cantidad que el desea que le llevemos hasta su casa.

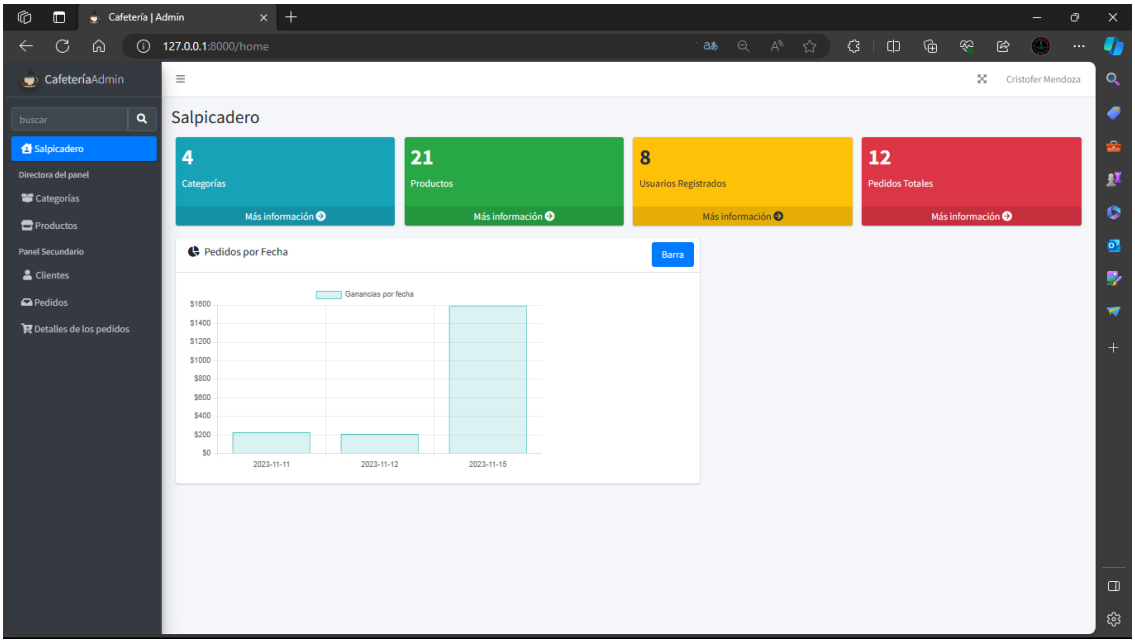


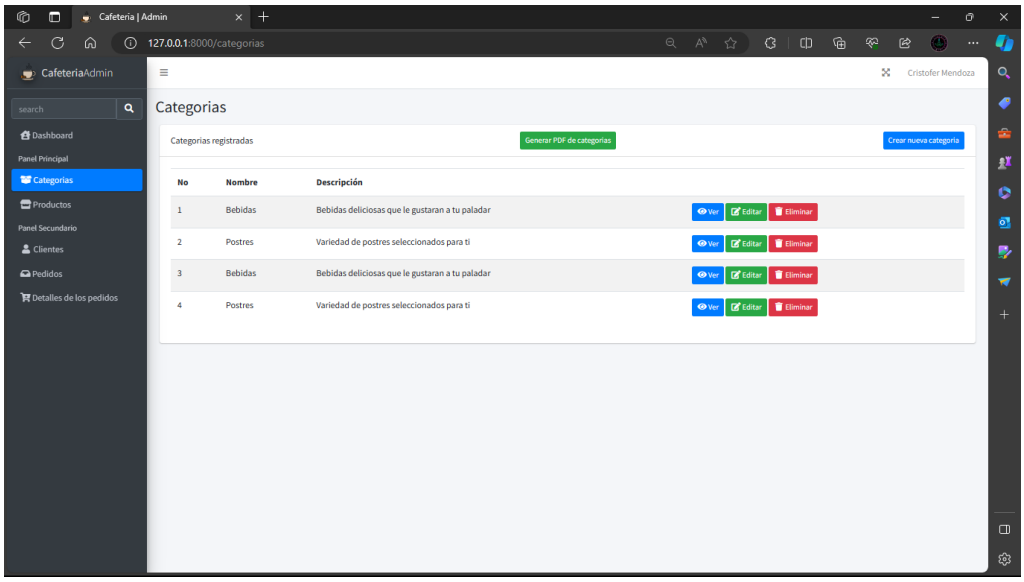


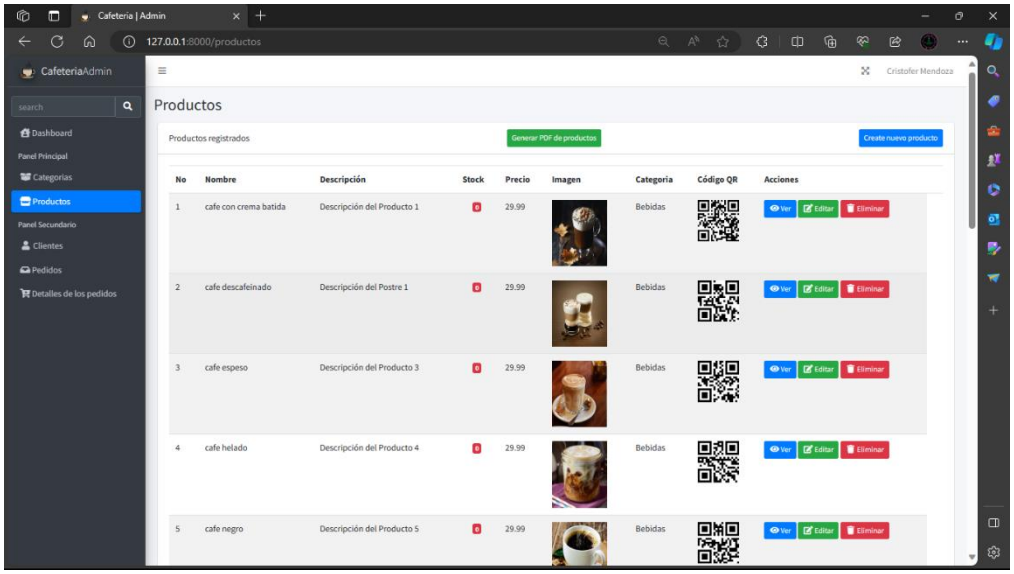
Si el usuario quiere volver hacer otro pedido el ya estará registrado en nuestra base de datos así que ya no será necesario que se vuelva registrar solo podrá ingresar con el correo y contraseña que ingreso la primera vez.

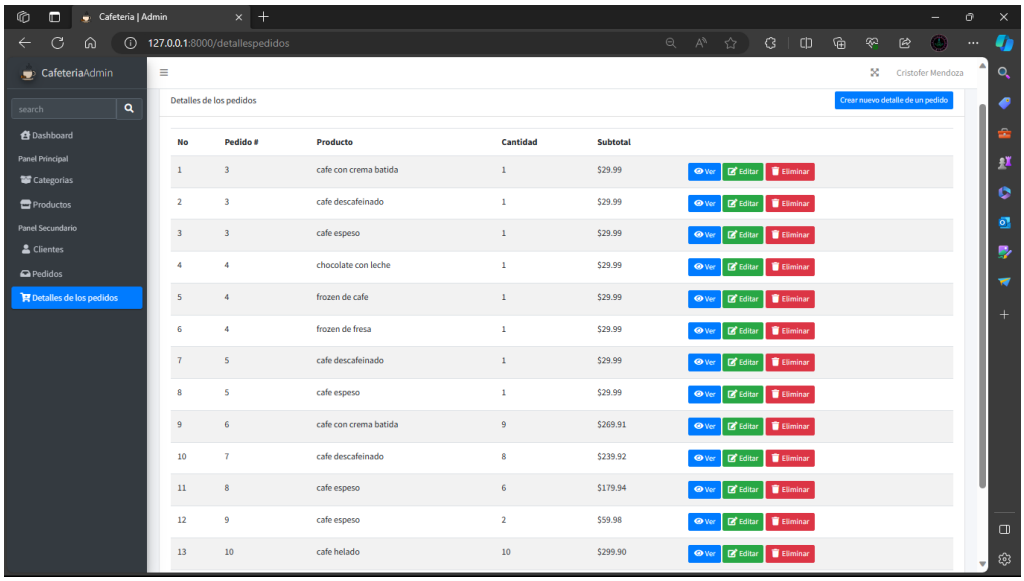


Panel de administrador




Cafeteria | Admin
127.0.0.1:8000/categorias
CafeteriaAdmin
Cristofer Mendoza
Categorias
Categorias registradas
Generar PDF de categorias
Crear nueva categoria
No
Nombre
Descripción
1 Bebidas Bebidas deliciosas que le gustaran a tu paladar
2 Postres Variedad de postres seleccionados para ti
3 Bebidas Bebidas deliciosas que le gustaran a tu paladar
4 Postres Variedad de postres seleccionados para ti
Ver
Editar
Eliminar


Cafeteria | Admin
127.0.0.1:8000/productos
CafeteriaAdmin
Cristofer Mendoza
Productos
Productos registrados
Generar PDF de productos
Create nuevo producto
No
Nombre
Descripción
Stock
Precio
Imagen
Categoria
Código QR
Acciones
1 cafe con crema batida Descripción del Producto 1 29.99 Bebidas
2 cafe descafeinado Descripción del Postre 1 29.99 Bebidas
3 cafe espeso Descripción del Producto 3 29.99 Bebidas
4 cafe helado Descripción del Producto 4 29.99 Bebidas
5 cafe negro Descripción del Producto 5 29.99 Bebidas
Ver
Editar
Eliminar


Cafeteria | Admin
127.0.0.1:8000/detallespedidos
CafeteriaAdmin
Cristofer Mendoza
Detalles de los pedidos
Crear nuevo detalle de un pedido
No
Pedido #
Producto
Cantidad
Subtotal
1 3 cafe con crema batida 1 $29.99
2 3 cafe descafeinado 1 $29.99
3 3 cafe espeso 1 $29.99
4 4 chocolate con leche 1 $29.99
5 4 frozen de cafe 1 $29.99
6 4 frozen de fresa 1 $29.99
7 5 cafe descafeinado 1 $29.99
8 5 cafe espeso 1 $29.99
9 6 cafe con crema batida 9 $269.91
10 7 cafe descafeinado 8 $239.92
11 8 cafe espeso 6 $179.94
12 9 cafe espeso 2 $59.98
13 10 cafe helado 10 $299.90
Ver
Editar
Eliminar

Enlaces de proyecto

Github

https://github.com/Chris-Men/proyecto

Drive

https://drive.google.com/drive/folders/1z9U06GkEUIlwkjbF1ruQtnNjbBeh7Nlt?usp=drive_link