UNIVERSITARIO RUMIÑAHUI

Pregrado

Carrera: SISTEMAS Y GESTION DE DATA

Asignatura (UIC): Ejecución de proyectos en TIC

Trabajo de titulación previo a la obtención del

Título en:

Tecnólogo Superior en Sistemas y Gestión de Data

Tema: CREACIÓN DE UN APLICATIVO WEB QUE AUTOMATICE EL CUADRE DE CAJA PARA EL EMPRENDIMIENTO COFFEE BURGUER

Autor: GUALAVISI CAIZA JHON FERNANDO

Tutor general: PAEZ OSCULLO DANNY SANTIAGO

Tutor técnico: LLUMIQUINGA ESPINOZA FABRICIO

Fecha: septiembre 2025

Autor: GUALAVISI CAIZA JHON FERNANDO

Título a obtener: Tecnólogo Superior en Sistemas y Gestión de Data

Matriz: Sangolquí -Ecuador

Correo electrónico: jhon.gualavisi@ister.edu.ec

Dirigido por: LLUMIQUINGA ESPINOZA FABRICIO JOSE

Título: Mg. Gerencia de sistemas y tecnología empresaria

Matriz: Sangolquí -Ecuador

Correo electrónico: jose.llumiquinga@ister.edu.ec

Dirigido por: PAEZ OSCULLO DANNY SANTIAGO

Título: MG en lógica, computación e inteligencia artificia

Matriz: Sangolquí -Ecuador

Correo electrónico: danny.paez@ister.edu.ec

Gualavisi Caiza Jhon Fernando

"CREACIÓN DE UN APLICATIVO WEB QUE AUTOMATICE EL CUADRE DE CAJA PARA EL EMPRENDIMIENTO COFFEE BURGUER"

CARTA DE CESIÓN DE DERECHOS DEL TRABAJO DE TITULACIÓN

CT-ANX-2025-ISTER-2-2.3

Sangolquí, 01 de septiembre del 2025

MSc. Elizabeth Ordoñez
DIRECTORA DE DOCENCIA

MSc. Mónica Loachamín
COORDINADORA DE TITULACIÓN

INSTITUTO SUPERIOR TECNOLÓGICO RUMIÑAHUI CON CONDICIÓN DE UNIVERSITARIO

Presente

Por medio de la presente, yo, **Gualavisi Caiza Jhon Fernando** declaro y acepto en forma expresa lo siguiente: Ser autor del trabajo de titulación denominado CREACIÓN DE UN APLICATIVO WEB QUE AUTOMATICE EL CUADRE DE CAJA PARA EL EMPRENDIMIENTO COFFEE BURGUER, de la Tecnología Superior Universitaria en Sistemas y Gestión de Data; y a su vez manifiesto mi voluntad de ceder al Instituto Superior Tecnológico Rumiñahui con condición de Universitario los derechos de reproducción, distribución y publicación de dicho trabajo de titulación, en cualquier formato y medio, con fines académicos y de investigación.

Esta cesión se otorga de manera no exclusiva y por un periodo indeterminado. Sin embargo, conservo los derechos morales sobre mi obra.

En fe de lo cual, firmo la presente.

Atentamente,

Gualavisi Caiza Jhon Fernando
C.I.: 1721605572

FORMULARIO PARA ENTREGA DE PROYECTOS EN BIBLIOTECA INSTITUTO SUPERIOR TECNOLÓGICO RUMIÑAHUI CON CONDICIÓN DE UNIVERSITARIO

CT-ANX-2025-ISTER-3

CARRERA:

TECNOLO SUPERIOR EN SISTEMAS Y GESTION DE DATA

AUTOR /ES:

JHON FERNANDO GUALAVISI CAIZA

TUTOR ACADÉMICO: PAEZ OSCULLO DANNY SANTIAGO

TUTOR TECNICO: LLUMIQUINGA ESPINOZA FABRICIO JOSE

CONTACTO ESTUDIANTE:

0958934855

CORREO ELECTRÓNICO:

jhonfernandocuatro@gmail.com

TEMA:

CREACIÓN DE UN APLICATIVO WEB QUE AUTOMATICE EL CUADRE DE CAJA PARA EL EMPRENDIMIENTO COFFEE BURGUER

OPCIÓN DE TITULACIÓN:

UNIDAD DE INTEGRACION CURRICULAR

RESUMEN EN ESPAÑOL:

En este proyecto se diseñó e implemento un aplicativo web en PHP+MySQL para automatizar el cuadre de caja de Coffee Burguer, lo cual remplaza registros en papel y conciliaciones manuales. Lo cual también integra un inventario con recetas y registro de egresos con comprobantes digitales de ser el caso, permitiendo la trazabilidad de ingresos y egresos y tener un cuadre de caja exacto. Metodológicamente se considera Scrum siendo sprint quincenales y con validaciones.

Los objetivos principales son reducir el tiempo que requiere para cerrar caja de 2 a 3 horas a un tiempo estimado menor a 15 minutos, disminuyendo errores significativamente al menos del 1%

y asegura el registro de cada egreso. La arquitectura MVC desacopla presentación, lógica y datos; el modelo relacional gestiona usuarios con roles (administrador/cajero), ventas y pedidos, insumos con stock mínimo y recetas para descuento automático. Se incluyen reportes con gráficos y exportación a Excel. Cada resultado obtenido se considera como mejora significativa de eficiencia operativa, reducción de merma por control de inventario y ayuda para cualquier toma de decisión con información confiable, El trabajo aporta evidencia de beneficio en PYMES gastronómicas al digitalizar procesos críticos de caja y stock, y sienta bases para futuras mejoras.

PALABRAS CLAVE:

Cuadre de caja, Inventario, Arquitectura MVC, PHP+ MySQL

ABSTRACT:

This project designed and implemented a web application in PHP and MySQL to automate the cash reconciliation process of Coffee Burguer, replacing paper records and manual reconciliations. The system also integrates an inventory module based on recipes and an expense module with digital receipts, ensuring complete traceability of income and expenses and enabling an accurate cash closure.

Methodologically, the development followed the Scrum framework, organized into biweekly sprints with continuous validations. The main objectives are to reduce the cash closing time from 2–3 hours to less than 15 minutes, decrease reconciliation errors to below 1%, and ensure the proper registration of each expense. The MVC architecture separates presentation, logic, and data; while the

relational model manages users with roles (administrator/cashier), sales and orders, inputs with minimum stock, and recipes that allow automatic inventory deductions.

The system includes graphic reports and Excel export, providing significant improvements in operational efficiency, reducing losses through better inventory control, and supporting decision-making with reliable information. This work demonstrates the benefits of digitalizing critical processes of cash management and inventory in small gastronomic businesses, and establishes a foundation for future enhancements.

PALABRAS CLAVE:

Cash reconciliation, Inventory, MVC architecture, PHP + MySQL

_____
Firma del Estudiante (AUTOR)
Jhon Fernando Gualavisi Caiza
C.I.: 1721605572

SOLICITUD DE PUBLICACIÓN DEL TRABAJO DE TITULACIÓN

CT-ANX-2025-ISTER-4

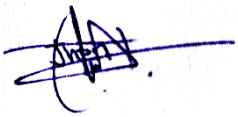
Sangolquí, 01 de septiembre del 2025

Sres.-
INSTITUTO SUPERIOR TECNOLÓGICO RUMIÑAHUI CON CONDICIÓN DE UNIVERSITARIO

Presente

Yo Jhon Fernando Gualavisi Caiza con C.I.: 1721605572 alumno de la Carrera de SISTEMAS Y GESTION DE DATA, cedo al INSTITUTO SUPERIOR TECNOLÓGICO RUMIÑAHUI CON CONDICIÓN DE UNIVERSITARIO, los derechos de publicaciones del presente trabajo de Titulación en el Repositorio Institucional para hacer uso de todos los contenidos con fines estrictamente académico o de investigación.

Atentamente,



_____
Firma del Estudiante
Jhon Fernando Gualavisi Caiza
C.I.: 1721605572

AGRADECIMIENTOS

Agradezco mucho a mis profesores, quienes con su dedicación, esfuerzo y conocimientos contribuyeron a mi formación académica, inculcándome valores de responsabilidad, disciplina y compromiso que fueron fundamentales a lo largo de mi carrera estudiantil.

En el proceso final muy agradecido con mis tutores, tutor general y tutor técnico quienes con su dedicatoria, esfuerzo y conocimientos contribuyeron a mi proyecto final, de tal manera que se obtuvieron valores de responsabilidad, disciplina y compromiso que fueron fundamentales en este proceso.

DEDICATORIA

Dedico este trabajo con todo mi cariño y gratitud a mis padres, en especial a mi padre, quien aún me acompaña y ha sido un ejemplo de esfuerzo y hombre trabajador junto con mi madre quien desde el cielo me protege y bendice cada paso que he dado siempre serán el motor de mi vida.

A mis hermanos cuñado y sobrinos, con quienes he compartido cada etapa de mi camino y han estado junto a mí en especial a mi hermano que partió antes de tiempo, cuya memoria sigue siendo una fuente de inspiración y fortaleza para mí.

A mi mujer, quien ha sido mi compañera por todos estos años y me dado ánimos en todo momento y lo cual hemos hecho de una casa un hogar de 4 personas, junto a nuestras mascotas hemos podido salir delante.

A todos ya antes nombrados dedico este logro, porque sin su presencia, su ejemplo y su amor, este sueño no habría sido posible.

ÍNDICE DE CONTENIDO

Índice de ilustraciones

Índice de tabla

1. CAPITULO I

1.1. RESUMEN

En este proyecto se diseñó e implemento un aplicativo web en PHP+MySQL para automatizar el cuadre de caja de Coffee Burguer, lo cual remplaza registros en papel y conciliaciones manuales. Lo cual también integra un inventario con recetas y registro de egresos con comprobantes digitales de ser el caso, permitiendo la trazabilidad de ingresos y egresos y tener un cuadre de caja exacto. Metodológicamente se considera Scrum siendo sprint quincenales y con validaciones.

Los objetivos principales son reducir el tiempo que requiere para cerrar caja de 2 a 3 horas a un tiempo estimado menor a 15 minutos, disminuyendo errores significativamente al menos del 1% y asegura el registro de cada egreso. La arquitectura MVC desacopla presentación, lógica y datos; el modelo relacional gestiona usuarios con roles (administrador/cajero), ventas y pedidos, insumos con stock mínimo y recetas para descuento automático. Se incluyen reportes con gráficos y exportación a Excel. Cada resultado obtenido se considera como mejora significativa de eficiencia operativa, reducción de merma por control de inventario y ayuda para cualquier toma de decisión con información confiable, El trabajo aporta evidencia de beneficio en PYMES gastronómicas al digitalizar procesos críticos de caja y stock, y sienta bases para futuras mejoras.

1.2. INTRODUCCION

La demandad de alimentación, sugiere otorgar a la gente una alimentación rápida en un lugar agradable esto se maneja desde el inicio de la amplificación de trabajos en jornada completa esto demanda a que un lugar tenga la acogida y calidez para ello se necesita que el negocio de comida rápida también cambie su forma de llevar sus productos y de llevar su contabilidad que es una figura determinante en su crecimiento y estabilidad

financiera. En un mundo cada vez más digitalizado, la simplificación de tareas es fundamental ya que mejora la eficiencia operativa, reduce errores de proveedores clientes y dueños y automatiza la gestión contable, negocios y franquicias de comida rápida requieren software adaptable que les del control financiero, sobre el negocio planteado, así como saber en dónde se puede mejorar e innovar dando la mejor atención a las personas que visitan su establecimiento.

El emprendimiento Coffee Burguer enfrenta desafíos en la gestión de su cuadre de caja debido a la ausencia de un sistema automatizado que facilite el registro de pedidos y administración de ingresos y egresos. Actualmente, los pedidos de los clientes se anotan manualmente en papel, se calculan los montos a pagar sin asistencia digital, y al finalizar la jornada se realiza una conciliación manual para determinar el estado financiero del negocio. Esta técnica habitual no lleva un control rigoroso lo que propensa a tener muchísimos y graves errores de ver el valor de ingresos netos; una iniciativa para solucionar este problema, es el desarrollo de un aplicativo web que digitalice el proceso de cuadre de caja, facilitando una interfaz gráfica responsiva e intuitiva, adaptada a las necesidades de Coffee Burguer. La aplicación permitirá a los usuarios registrar ventas, gestionar egresos, consultar datos históricos y administrar productos de manera eficiente, eliminando la necesidad de registros físicos en papel y reduciendo los errores en cálculos financieros.

Además, el sistema implementará una base de datos relacional, garantizando el almacenamiento seguro de la información y facilitando la consulta de reportes financieros estructurados. El proyecto se desarrollará utilizando el lenguaje de programación PHP, con un enfoque basado en metodologías ágiles, asegurando una implementación flexible y eficiente.

Con esta solución, Coffee Burguer podrá mejorar significativamente su gestión administrativa, optimizando su operatividad y fortaleciendo la toma de decisiones estratégicas basadas en datos precisos.

1.3. PLANTEAMIENTO DEL PROBLEMA

El proceso de cuadre de caja en *Coffee Burguer* se basa en registros físicos en papel y cálculos manuales, lo que implica una alta posibilidad de cometer errores y perder información en el momento de finalizar el día con el cierre de caja.

Actualmente la manera en la Coffee Burguer realiza sus operaciones administrativas desde que toma los pedidos hasta que cierra la caja es la siguiente: El mesero anota los pedidos en hojas de papel, sin estar notificado del inventario para tomar la orden, ya que, antes de tomar los pedidos debería saber si faltan ingredientes y proceder con la orden. Los precios de los productos son sumados manualmente por el mesero, si existe algún error en la suma puede generar falencias en los precios finales. Los gastos del negocio también son anotados manualmente, sin poder incluir pruebas documentales como imágenes de facturas. El cierre de caja se realiza manualmente al final del día mediante los registros físicos y se compara el dinero en caja con los ingresos anotados, el cierre de caja toma entre 2-3 horas extras del personal más tiempo del necesario al depender de registros físicos. Además, se debe tomar en cuenta que, no se tiene un sistema que registre los pagos del cliente en efectivo o transferencia, lo que dificulta la conciliación de ingresos, el pago se realiza con efectivo.

El proceso manual y desorganizado por parte de Coffee Burguer presenta consecuencias directas en el negocio por la falta de digitalización en la administración financiera:

- **Errores en cálculos financieros:** El método manual produce el descuadre en caja, afectando la precisión en el control económico del negocio. Sin conocer la realidad de ventas imposibilita la determinación de ganancias reales por productos (Costos de materiales utilizados vs. Precios de venta)
- **Pérdida por merma no registrada:** La falta del correcto control con productos caducados, genera pérdida de capital invertido en insumos.
- **Quiebres de stock:** La falta de insumos en horas pico impacta negativamente en las ventas y en la experiencia del cliente
- **Compras desorganizadas:** La falta de un historial de consumo puede generar sobrecompra (dinero inmovilizado) o subcompra (pérdida de ventas).
- **Costo real indefinido:** No se puede calcular el costo del plato si no se puede comparar el dinero empleado entre los ingredientes utilizados y lo que se vende.
- **Impedimento de consulta anterior:** Al no tener una base de datos digital impide que la planificación del negocio sea menos eficiente y no se pueda ver el mejor producto aceptado por el cliente y que tendrá mayor demanda de consumo.
- **Mala toma de acciones:** Sin información organizada y accesible, el administrador no puede evaluar patrones de consumo ni identificar oportunidades de mejora y muy alto impacto en la renta del negocio, perdida valiosa de tiempo. Sin esta información de los datos que ayude a optimizar el menú, los precios o compras; lo que no ayuda a identificar los productos con baja rentabilidad.

Dado los problemas descritos, es evidente que *Coffee Burguer* necesita una solución tecnológica que le permita optimizar el cuadre de caja, garantizando seguridad y accesibilidad en la gestión financiera lo que amenaza la sostenibilidad del negocio.

1.4. ANTECEDENTES

1.1.1. El adelanto contable en negocios de comida rápida

En Ecuador, la comida rápida es un estilo de vida, que tiene desafíos en sus ventas y como mejorara la atención al cliente; sin dejar a lado saber si se gana o se pierde, todo esto surge por la falta de tecnología adaptadas a sus necesidades, tanto personales como en los diferentes negocios. La gestión manual ha sido la práctica predominante, donde los registros financieros se llevaban en papel o en hojas de cálculo simples, lo que generaba problemas de precisión, pérdida de información y dificultad en la toma de decisiones estratégicas y el robo de los recursos financieros.

El avance tecnológico fortalece cada vez más emprendimientos pequeños grandes y medianos no importa el flujo de clientes sino de cómo se mejore sus servicios para saber si estos son factibles y rentables se han optado por la implementación de soluciones automatizadas para mejorar sus procesos administrativos. Un correcto control financiero como parte de la economía, permite a los comercios analizar, aclarar y entender el comportamiento de las inconstantes en su conjunto y la interacción que ellas mantienen dentro de la empresa para soportar la valoración, el costo de capital, el beneficio y la renta, así como, también el capital, su estructura, su valor de equilibrio, la información, la propiedad y las señales. Por su parte, las Pequeñas y Medianas Empresas (PYMES) se han vuelto un engranaje importante en la economía de los países como gestoras de producción y empleabilidad, es así que, el objetivo de la investigación es de dar un mejor servicio en comida rápida que la gente se sienta a gusto de comer en un lugar cómodo y su atención sea eficiente.

Los negocios gastronómicos han sido parte de esta evolución PYMES incorporando aplicaciones para la administración de pedidos, gestión de inventarios y seguimiento de ventas. Sin embargo, muchos de estos sistemas dejan de lado el proceso

fundamental del cuadre de caja, limitando la capacidad del negocio para visualizar sus ingresos reales y evaluar su rentabilidad. Con el uso de este aplicativo se controlará el valor total de las ventas obteniendo un real estado de caja.

1.1.2. El contexto de Coffee Burguer

¿Qué es *Coffee Burguer*? Mas que un sitio para degustar comida rápida también puedes degustar de un café o té; un sitio donde las personas pueden disfrutar de una experiencia completa que satisface sus necesidades y deseos, ya sea sociales, de trabajo o de familia; teniendo en cuenta que es Coffee Burguer sabemos que su administración la gestión financiera de cierre de caja se la sigue llevando en un cuaderno de ingresos y egresos, lo que ocasiona dificultades en la administración financiera del negocio. Porque no se sabe si se gana o se pierde durante un día común; en la actualidad, los pedidos de los clientes son registrados en papel, los cálculos del pago total se hacen sin asistencia digital, y al finalizar la jornada se recopilan todos los registros físicos para hacer una conciliación manual de ingresos y egresos y obtener la ganancia real esto confiando en la honorabilidad del personal de trabajo.

Como se ha mencionado realizar este método genera principalmente pérdida de tiempo por parte de todo el personal y también problemas de organización, afectando la capacidad del negocio para evaluar su rendimiento y tomar decisiones fundamentadas. Ante esta problemática, surge la necesidad de desarrollar un aplicativo web especializado, que permita automatizar el proceso de cuadre de caja, garantizando la seguridad de los datos y mejorando la eficiencia en la administración.

1.5. JUSTIFICACIÓN

La manera en la cual opera actualmente Coffee Burger resulta es ineficiente por tener una desconexión entre los flujos financieros y los materiales. La manera en la que se trabaja actualmente es mediante registros manuales en papel, lo que puede generar

errores no solo en el cuadre de caja, sino que también que impide que se integren y relacionen correctamente los datos de ventas, inventario y compras. Lo que puede generar quiebres de stock continuamente, y por ende tener pérdidas de venta y capital.

Este apartado desarrolla tres tipos de justificación: social/operativa, económica y técnica, mostrando la importancia del proyecto tanto en su impacto sobre los clientes y el negocio como en las herramientas tecnológicas empleadas para su implementación.

1.1.3. Justificción Social/Operativa

El desarrollo de la aplicación tiene beneficios tanto para las personas y clientes de *Coffee Burguer:*

a. Eliminará 2-3 horas de trabajo diario en la administración manual, reduciendo el estrés y jornadas extendidas.
b. Se minimizará errores al simplificar la toma de decisiones mediante la interfaz intuitiva
c. Facilita tomar decisiones operativas inmediatas ante alertas de stock y reposición
d. Mejora la experiencia de los clientes al evitar cancelar los pedidos por falta de insumos.

1.1.4. Justificación Económica

La automatización del cierre de caja en *Coffee Burguer* permitirá reducir errores administrativos, mejorar la organización de datos financieros y garantizar una gestión más confiable.

a. Reducción de merma: Mejor control de productos caducados y socnsumo real de insumos, evitando tener pérdidas en productos no utilizados.

b. Compras: Evita invertir capital en sobrecompras o subcompras.

c. Ahorro operativo: Elimina horas de trabajo extras para el cuadre de caja final.

d. Prevención de quiebres de stock: Genera una alerta si se tiene una reposición para maximizar ventas en horas pico.

1.1.5. Justificación Técnica

La implementación de un aplicativo web especializado permitirá resolver las dificultades actuales, brindando herramientas digitales que aseguren un registro preciso de ingresos y egresos, organizando la información en una base de datos relacional, permitiendo que también se integren los flujos financieros y de materiales.

El desarrollo web con dato relacionados en MySQL y backend en PHP ofrece:

a. Integridad de datos: Mejor relación entre inventario,ventas y finanzas.
b. Seguridad: Los datos financieros se encuentran encriptados y se puede generar una autenticidad de usuarios.
c. Sostenibilidad: El bajo costo del mantenimiento de genera sostenibilidad.

1.6. OBJETIVOS

1.1.6. Objetivo General

Desarrollar una aplicación web para la gestión integral del punto de venta y control de inventarios en el emprendimiento Coffee Burguer, con el fin de automatizar el cuadre de caja y optimizar la gestión de insumos para mejorar la eficiencia operativa y la rentabilidad del negocio.

1.1.7. Objetivos Específicos

1. Analizar los procesos actuales de venta y manejo de insumos para levantar requerimientos del sistema.
2. Diseñar la arquitectura del sistema, incluyendo la base de datos y el diseño de interfaz.
3. Desarrollar el módulo de ventas (registro de pedidos, cálculos, cuadre de caja).

4. Implementar el módulo de inventario, que permita:

- Registrar insumos.
- Definir recetas por producto.
- Disminuir stock automáticamente con cada venta.

5. Validar el sistema con pruebas y evaluar su impacto real en la operación del local.

1.7. ALCANCE

El aplicativo web de Coffee Burguer se enfocará en el proceso de ventas e inventarios necesario que ayude a mejorar el cuadre de caja y la correcta gestión de insumos.

1.1.8. Aspectos dentro del alcance del sistema

Módulo de Ventas. El sistema permitirá gestionar mejor el ciclo completo financiero desde que se registran los pedidos hasta que se realiza el cierre de caja. Mediante una interfaz intuitiva se relacionará cada venta del inventario en tiempo real. Lo que hace que el proceso sea más ágil y fácil de usar, lo que reduce los errores en el momento que se piden los pedidos. Los precios serán calculados automáticamente eliminando errores. Se incluirá un sistema de roles y permisos que tienen los empleados, asegurando que solo el administrador puede acceder a reportes y ajustar los precios y gestionar pedidos, mientras que el cajero solo podrá ingresar pedidos y egresos, sin opción de modificar registros históricos. Los egresos se registrarán con la facilidad de poder adjuntar comprobantes. Cada transacción será almacenada digitalmente y adjuntar imágenes de facturas, facilitando la consulta histórica de ventas, sustituyendo el papel y así tener una vinculación de cada venta con el inventario.

Módulo de Inventario. El inventario tendrá un catálogo digital que se actualice constantemente Se registrarán las entradas de stock por compras lo que permitirá definir con precisión la cantidad de insumos necesarios que se utilicen en cada venta. Las recetas de cada producto se podrán establecer tomando en cuenta los insumos y las cantidades que se utilizan para la preparación de platos. Si registramos una venta, podremos reducir automáticamente la cantidad de insumos en el inventario, lo que ayuda a mantener un preciso control del stock disponible. Los reportes de stock actual facilitarán la información para que los administradores se informen y tomen decisiones de reabastecimiento y gestionar mejor los insumos para planificar compras que se sustenten en el consumo real realizado, lo que evita los quiebres o excesos en el inventario.

1.1.9. Aspectos fuera del alcance del sistema (limitaciones)

Con la finalidad de mejorar la eficiencia operativa sin complicar el sistema funcional, se considera el pprocesamiento de pagos electrónicos. Nuestro aplicativo solo contemplara dos formas de pago, que es mediante dinero en efectivo y depósitos a la cuenta dada de Coffe Burger sin implementar un sistema de facturación electrónica, las transacciones no generarán automáticamente facturas digitales. No se incluye un sistema para controlar lotes o fechas de caducidad de los insumos con los que contamos. La gestión de relaciones con proveedores (CRM) no se implementará, dificultando una gestión y comunicación en conjunto con los proveedores.

2. Capítulo II: MARCO TEÓRICO

Al desarrollar el aplicativo web para la automatización del cuadre de caja en Coffee Burguer se fundamenta en los estudios para incorporar tecnología que ha demostrado su importancia en la digitalización en la gestión financiera de pequeños y grandes negocios, proporcionando la base conceptual para desarrollar el proyecto, valorando la conceptualización de administración financiera, digitalización de procesos y desarrollo de aplicaciones web.

2.1. Gestión Financiera en Pequeñas y Medianas Empresas (PYMES):

El control de cierre de caja y la planificación financiera son determinantes para que no desaparezca y mediante la sostenibilidad ver un crecimiento de las PYMES. Jiménez (2013) menciona que, el uso del capital se puede anticipar con el conocimiento y la información contable y las finanzas, calculando el dinero necesario para cubrir las necesidades actuales y si se necesitará en el futuro un financiamiento externo a corto plazo para cubrir las necesidades del capital utilizado, lo que ayuda a que se pueda verificar la concordancia entre el dinero que tenemos y los registros contables, identificando discrepancias y previniendo desviaciones financieras. Si automatizamos procesos, reducimos los errores humanos y mejoramos la trazabilidad podemos tener una reducción del 40% (García & López, 2021).

Fargo (2020) señala que para que una PYME siga constante en su crecimiento, debe tomar en cuenta la innovación en gestión financiera mediante la integración de herramientas tecnológicas y un plan que cuide los recursos financieros y a la vez impulse el crecimiento de la PYME, para ver un desarrollo en las PYMES se han destaca 5 pilares indispensables:

1. Planeación de flujos de efectivo: Proyecta objetivamente los ingresos que se pretende recibir luego de generar las ventas y los gastos operativos más

importantes que se han realizado, tomando mayor importancia los que se vinculan con los proveedores; una medida que ayuda a pagar el ciclo operativo y así evitar que el negocio se quede sin fondos monetarios.

2. Control financiero: Mediante el control financiero se programan actividades que ayudan a corregir errores cometidos que han sido detectados en primera instancia, las planificaciones se pueden generar a corto, mediano o largo plazo, dependiendo de las necesidades de consolidación de la empresa.

3. Monitoreo del entorno económico: Mediante un análisis no solo local o nacional, se analiza también globalmente el sector productivo en que está involucrado el negocio, con la finalidad de controlar los recursos financieros y de esta manera se mitigan riesgos directos que pueden afectar al negocio

4. Aprovechamiento de recursos tecnológicos: Se puede tomar decisiones precisas en el campo administrativo, financiero y de mercado, si en el área de tesorería se desarrollan procesos prácticos y de control que permitan unificar base de datos.

5. Desarrollo de estrategias comerciales: Se utilizan medios digitales para ampliar el mercado mediante la difusión de información para que adquieran productos o contraten servicios (Fargo, 2020).

2.2. Sistemas de Punto de Venta (POS):

Los sistemas punto de venta POS, son herramientas tecnológicas que se utiliza para que el cliente tenga facilidad de pagar en establecimientos comerciales, realizando transacciones de manera eficiente y precisa. Según Villamil (2021), menciona diferentes funcionalidades como; contar los artículos mediante el registro y la contabilidad de productos que el cliente desea adquirir, el pago del articulo o los artículos se puede

realizar mediante diferentes métodos de pago (efectivo, tarjetas de crédito y débito), actualizar el inventario conforme se realizan las compras, aplicar los descuentos o incrementos y entregar una factura al cliente

Los desarrollos de Punto de Venta son considerados como un tipo de software de aplicación que cumple una tarea específica (Villamil, 2021).

2.2.1. Componentes de un Sistema POS

Un sistema POS generalmente consta de varios componentes clave:

Hardware:

- Terminales con pantallas 27ardex27s
- Cajones de dinero para almacenar el efectivo.
- Impresoras para los comprobantes de compra.
- Escáneres de códigos de barras

Software:

- Módulo de gestión de ventas y programas que permiten registrar las transacciones, devoluciones, gestionar el inventario y generar reportes.
- Herramientas de análisis que reporten tendencias y la rentabilidad por producto.
- Interfaces que integren la contabilidad con el inventario para analizar los datos de venta

2.2.2. Beneficios de los Sistemas POS

Los sistemas POS permiten un procesamiento rápido de las transacciones, lo que mejora la experiencia del cliente y reduce las filas en el punto de venta.

- El sistema POS resulta conveniente para el cliente, ya que le permite pagar en cualquier punto de la tienda cuando visita un restaurante (Ramos, et al., 2017).
- El inventario con cada venta, se minimizan los errores y se facilita la gestión de stock.
- Proporcionan reportes detallados sobre ventas, tendencias y comportamiento del cliente, lo que ayuda a los gerentes a tomar decisiones informadas sobre promociones y gestión de productos.
- **Mejora en la atención al cliente** en el proceso de pago (Ramos, et al., 2017).

2.3. Sistemas de Gestión de Inventarios (IMS):

Los sistemas de gestión de inventario son herramientas tecnológicas y procesos que pueden ser manuales o automatizados, y suelen incluir software especializado que facilita la gestión y el análisis de datos relacionados con el inventario, que se utilizan para supervisar, controlar y optimizar la gestión del flujo de bienes, que pueden incluir productos terminados, productos en proceso y materia prima.

La correcta gestión de inventarios es esencial para garantizar que las operaciones de la empresa se realicen sin interrupciones y que se mantenga un nivel adecuado de stock mediante el registro y estado de los productos para satisfacer la demanda del cliente. Supervisar el movimiento de los inventarios, desde la recepción de materias primas hasta la venta de productos terminados. Prever las necesidades futuras de inventario basándose en patrones de consumo y tendencias de mercado. Mejorar la eficiencia operativa al reducir costos asociados con el exceso o la falta de inventario.

La gestión adecuada de inventarios es crucial para el éxito de cualquier organización, y su importancia se puede resumir en los siguientes puntos:

Según Múzquiz (2013), los inventarios permiten a las empresas mantener la continuidad en la producción y evitar retrasos en la entrega de productos. Esto es

fundamental para satisfacer la demanda del cliente y mantener la reputación de la empresa. Los inventarios actúan como reguladores o amortiguadores entre las diferentes fases del proceso productivo. Esto significa que pueden equilibrar las variaciones en la producción y la demanda, lo que ayuda a evitar cuellos de botella y a optimizar el flujo de trabajo (FAEDIS, 2016ª). La gestión de inventarios es una parte integral de la administración de recursos. Permite a las empresas planificar y controlar sus actividades de manera más efectiva, lo que contribuye a una rentabilidad adecuada (FAEDIS, 2016b). Un sistema de gestión de inventario eficiente ayuda a minimizar costos asociados con el almacenamiento, la obsolescencia y la merma de productos. Esto se traduce en una mejora en la rentabilidad de la empresa. La recopilación y análisis de datos sobre inventarios permite a los gerentes tomar decisiones informadas sobre compras, producción y ventas, lo que puede mejorar la competitividad de la empresa en el mercado.

Los sistemas de gestión de inventario son fundamentales para el funcionamiento eficiente de las organizaciones. No solo garantizan la disponibilidad de productos, sino que también contribuyen a la sostenibilidad y rentabilidad de la empresa al optimizar sus procesos y recursos.

La importancia del control de inventarios radica en contar con información suficiente y útil para minimizar costos de producción y aumentar liquidez, manteniendo un nivel de inventario óptimo y empezar a utilizar la tecnología con la disminución de gastos operativos (López-Prado, 2018). En definitiva, el control de inventario es una de las acciones y mecanismos más importante dentro de la administración y gestión organizacional. El control de inventarios puede realizarse por varios métodos, los cuales dependerán de la misión y visión de la organización. Dentro de todos los métodos o técnicas aplicados prevalece como una herramienta fundamental dentro del control de inventarios y es la toma física de los mismos. Según: Molina-Herrera (2015):

La toma física de los inventarios se entiende como la recopilación y determinación en forma debida de las existencias físicas, donde se efectuará a través de inventarios periódicos y reportes o auxiliares de bodega, estos deben cuadrar con los saldos contables (30ardex). Facilita el control y la capacitación oportuna en registros contables de los embarques, en su caso el registro de la factura. (p. 24).

Tabla 1

Objetivos de los inventarios

OBEJTIVOS	CARACTERÍSTICAS
Reducción del riesgo	Se desconoce con certeza de la demanda de productos terminado por tanto se debe mantener un stock de seguridad de productos terminados, para evitar un desabastecimiento de demanda ante un aumento y un stock de seguridad de materias primas, para evitar una detención del proceso de producción.
Abaratar las adquisiciones y la producción	La producción por lotes permite reducir costos, puesto que se distribuye el costo fijo de las maquinas. La adquisición de materias primas por lotes permite descuentos, reparto de costos de transporte, etc. En ambas se necesita un

	gran nivel de inventarios (de productos terminados y de materias primas).
Anticipar las variaciones previstas de la oferta y la demanda	Por ejemplo, la escasez de un producto debido a una huelga de sus productores, disminuye la oferta con lo que se debe acumular en los inventarios. Materias primas o productos terminados sometidos a variaciones estacionales aumenta la demanda, con lo que se acumulan en almacenes.
Facilidad al transporte y distribución del producto	Si los productos se consumen en un lugar distinto al que se producen, el transporte no puede ser utilizado de una forma continua, con lo que se realizara por lotes.
Especulación	Acumulación de productos ante futuras subidas de precio.

Fuente: Samamez-Pascual (2017).

2.3.1. Métodos de Valoración: PEPS (FIFO):

El método PEPS, conocido también como "Primero en Entrar, Primero en Salir" (FIFO por sus siglas en inglés: First In, First Out), es una técnica de gestión de inventarios ampliamente utilizada en empresas del sector alimentario. Este método consiste en que los primeros productos que ingresan al inventario son los primeros en salir o en ser utilizados para la venta o producción. De esta manera, se asegura que los productos más

antiguos se consuman antes que los más recientes, lo cual es fundamental para evitar la caducidad y el desperdicio, especialmente en alimentos perecederos.

La aplicación del método PEPS permite mantener la frescura de los productos ofrecidos al cliente y reduce el riesgo de pérdidas económicas por productos vencidos. Además, facilita el control de inventarios y la rotación de stock, lo que contribuye a una mejor planificación de compras y reposición de insumos. Según Krajewski et al. (2019), el método PEPS es el más recomendado para empresas que manejan productos con fecha de caducidad, ya que ayuda a cumplir con normativas sanitarias y de calidad.

En la práctica, la implementación del método PEPS implica organizar físicamente los almacenes de modo que los productos más antiguos sean los más accesibles para su uso o venta. Asimismo, los sistemas de gestión de inventario pueden automatizar este proceso, registrando las fechas de entrada de cada lote y sugiriendo la salida de los productos más antiguos.

En resumen, el método PEPS es esencial en la industria alimentaria para garantizar la calidad, seguridad y eficiencia en la gestión de inventarios, minimizando pérdidas y asegurando la satisfacción del cliente.

2.3.2. Concepto de Recetas (Bill of Materials – BOM)

En la gestión de inventarios y producción, especialmente en la industria alimentaria, el concepto de receta o “Bill of Materials” (BOM) es fundamental para el control eficiente de los insumos. Una BOM es una lista estructurada que detalla todos los ingredientes, materiales y cantidades necesarias para elaborar un producto compuesto, como puede ser una hamburguesa, una bebida o cualquier platillo que requiera la combinación de varios insumos.

La principal importancia de la BOM radica en que permite a las empresas conocer con exactitud la cantidad de cada insumo que se utiliza en la elaboración de un producto final. Esto facilita el control del inventario, ya que cada vez que se registra una venta, el sistema puede descontar automáticamente las cantidades correspondientes de cada ingrediente utilizado, evitando así el desabastecimiento o el exceso de stock. Además, la BOM ayuda a estandarizar los procesos de producción, asegurando que todos los productos mantengan la misma calidad y sabor, lo que es esencial para la satisfacción del cliente y la reputación del negocio.

Otra ventaja significativa de utilizar recetas o BOM es la posibilidad de calcular de manera precisa los costos de producción, lo que contribuye a una mejor fijación de precios y a la identificación de oportunidades para optimizar el uso de los recursos.

Asimismo, la BOM es una herramienta clave para la planificación de compras, ya que permite prever con mayor exactitud las necesidades de insumos en función de la demanda esperada.

La implementación de recetas o Bill of Materials en los sistemas de gestión de inventario es esencial para controlar el uso de insumos en productos compuestos, mejorar la eficiencia operativa, reducir desperdicios y garantizar la calidad constante de los productos ofrecidos.

2.3.3. Conceptos Clave de Inventario:

Aquí encontramos un conjunto de términos comunes usados en esta investigación:

- **Contabilidad de costos:** La principal en acumular y analizar la información relevante para que el gerente utilice de manera interna la información y planifique, controle y tome decisiones

- **Control de inventarios:** Herramienta que permite a las empresas conocer las cantidades existentes de productos disponibles para la venta.
- **Coste:** Valor monetario de los gastos de las materias primas, equipos, suministros, servicios, mano de obra, productos, etc., que se utilizan para la creación del producto o servicio.
- **Costeo directo:** Método del costo de producción, se forma con todas aquellas erogaciones de materia prima, mano de obra, y cargos indirectos que tengan un comportamiento variable con relación a los cambios en los volúmenes de producción.
- **Costo de ventas:** Aquel valor en que se incurre para producir, almacenar, transportar o comercializar un bien, o para prestar un servicio, o la suma de todos estos cuando se trate de materias primas que necesiten ser transformadas.
- **Indicador de ventas:** No es otra cosa diferente al valor monetario dado en ventas durante un periodo de medición.
- **Inventario de seguridad:** se conoce como una reserva de inventario o inventario adicional para la protección de la producción.
- **Inventario final:** Son las mercancías que se encuentran en existencia, disponibles para la venta.
- **Merma:** Cantidad de dinero que se ha perdido en productos mal logrados o mal manejados.
- **Método ABC:** Método de Costeo Basado en Actividades, el cual mide el desempeño de cada actividad, fundado en el uso clasificado de los recursos.

- **Método P.E.P.S:** Es un método de valuación de inventarios y consiste básicamente en darle salida del inventario a aquellos productos que se adquirieron primero, por lo que en los inventarios quedarán aquellos productos comprados más recientemente.

- **Stock:** Se refiere a la cantidad de bienes o productos que dispone una organización o un individuo en un determinado momento para el cumplimiento de sus operaciones comerciales.

- **Punto de reorden:** Nivel que genera una alerta para la reposición.

2.4. Tecnologías para el Desarrollo de la Aplicación:

El PHP es un lenguaje utilizado para poder desarrollar aplicaciones web dinámicas y que son de bajo costo, la base de datos es MySQL que garantiza integridad en la gestión de transacciones el inventario, permitiendo una consulta de datos (López & Ramírez, 2022). La interfaz se puede controlar desde dispositivos móviles y computadoras(Martinez et al., 2021). La combinación de estas herramientas permite que un desarrollo ágil mientras mantiene la sostenibilidad y protege los datos financieros.

3. CAPÍTULO III: MARCO METODOLÓGICO

Este capítulo detalla el marco metodológico empleado para el diseño, desarrollo y validación de la aplicación web para Coffee Burguer. Se describe el tipo de investigación, la metodología de desarrollo de software seleccionada, las fases del proyecto, las herramientas utilizadas y los procedimientos de validación.

3.1. Tipos y enfoques de investigación

La metodología usada durante la investigación, comprendiendo detalles sobre el diseño y desarrollo del aplicativo propuesto. Tipo de investigación con un enfoque cuasi-experimental. El sistema a desarrollar no tiene una alta complejidad, el código debe ser sencillo, tanto que pueda modificarse fácilmente, las iteraciones deben ser presentadas en corto tiempo y requiere ser hecho en el menor tiempo posible. Técnicamente, no se requiere abundante documentación a lo largo del proceso de desarrollo, ni la participación de un número considerable de desarrolladores para su construcción.

La metodología aplicada en la investigación tiene un diseño cuasi-experimental. Sólo se asigna la condición de diseño documental aquellas investigaciones en las cuales la fuente de los datos que se van a analizar para llegar a las conclusiones, son documentos, y estos datos están dirigidos a obtener conocimiento nuevo (Hurtado de Barrera, 2010). Del mismo modo, la investigación documental se concreta exclusivamente en la recopilación de información de diversas fuentes, con el objeto de organizarla describirla e interpretarla de acuerdo con ciertos procedimientos que garanticen confiabilidad y objetividad en la presentación de los resultados (Palella & Martins, 2010).

Una de las herramientas de este tipo de metodología es la revisión de literatura. Este instrumento consiste en detectar, consultar y obtener la bibliografía y otros materiales útiles para los propósitos del estudio, de los cuales se extrae y recopila información relevante y necesaria para el problema de investigación (Hernández-Sampieri et al., 2014). La información por extraer proviene de artículos científicos, trabajos de grado de diversas universidades, informes técnicos y documentos web (Vélez Vélez, 2022).

A continuación, se procede a hacer un análisis comparativo de la manera en la que se trabajaba en contraste con la implementación de Scrum.

La gestión de proyectos tradicional prioriza una planificación inicial exhaustiva, independientemente de si se comprenden plenamente los requisitos del proyecto. Esta planificación detallada busca establecer variables fijas como el tiempo, el coste y el alcance. Sin embargo, en el entorno actual, que evoluciona rápidamente, los requisitos del proyecto cambian con frecuencia. Por lo tanto, el considerable tiempo invertido en la planificación inicial puede desperdiciarse si posteriormente se producen cambios significativos en las especificaciones.

Fomenta la toma de decisiones iterativa y reduce el tiempo dedicado a variables desconocidas y propensas a cambios. Scrum se adapta al cambio como ningún otro. Se basa en el concepto de ofrecer el máximo valor al cliente en el menor tiempo posible, garantizando un producto potencialmente entregable al final de cada sprint (también conocido como iteración) (SCRUM, 2022).

En la **Tabla 2** se indica la comparativa de las características de la gestión de proyectos bajo la metodología tradicional, que es el registro y manejo de pedidos a mano

y e uso de papel, y la metodología Scrum. Se asignan calificaciones de 1 a 5 a cada característica, siendo 5 la máxima calificación y 1 la mínima. Estos valores han sido escogidos de acuerdo a la experiencia del autor del proyecto de titulación.

Tabla 2

Comparativa entre Metodología Scrum y Metodología Tradicional

Características	Scrum	Metodología Tradicional
Flexibilidad ante cambios	5	2
Colaboración del equipo	5	3
Entregas frecuentes	5	2
Documentación necesaria	3	5
Visibilidad del progreso	5	3
Satisfacción del cliente	5	3
Adaptación a requisitos del cliente	5	2
Suma Total	33	20

Fuente: Elaboración propia, 2025.

Flexibilidad ante cambios:

- **Scrum (5):** Permite adaptarse rápidamente a los cambios en los requisitos del cliente, ya que se fundamenta en iteraciones cortas (sprints) que permiten ajustes continuos.
- **Metodología Tradicional (2):** Generalmente sigue un enfoque rígido, donde los cambios son difíciles de implementar una vez que se ha definido el alcance del proyecto.

Colaboración del equipo:

- **Scrum (5):** Fomenta la colaboración constante entre los miembros del equipo a través de reuniones diarias y revisiones de sprint.
- **Metodología Tradicional (3):** La colaboración puede ser limitada, ya que los roles y responsabilidades están más definidos y separados.

Entregas frecuentes:

- **Scrum (5):** Se enfoca en entregar incrementos funcionales del producto al final de cada sprint, lo que permite obtener retroalimentación continua.
- **Metodología Tradicional (2):** Las entregas suelen ser al final del ciclo de vida del proyecto, lo que puede llevar a retrasos en la obtención de feedback.

Documentación necesaria:

- **Scrum (3):** Aunque se requiere cierta documentación, se prioriza la comunicación y la colaboración sobre la documentación exhaustiva.
- **Metodología Tradicional (5):** Requiere una documentación detallada en cada fase del proyecto, lo que puede ser un proceso laborioso.

Visibilidad del progreso:

- **Scrum (5):** Utiliza herramientas como tableros Kanban y gráficos de burndown para proporcionar visibilidad clara del progreso del proyecto.
- **Metodología Tradicional (3):** La visibilidad puede ser limitada, ya que el progreso se mide a través de hitos y entregas finales.

Satisfacción del cliente:

- **Scrum (5):** La participación activa del cliente en el proceso de desarrollo asegura que el producto final cumpla con sus expectativas.
- **Metodología Tradicional (3):** La satisfacción del cliente puede verse afectada si los requisitos no se comprenden completamente al inicio del proyecto.

Adaptación a requisitos del cliente:

- **Scrum (5):** Permite ajustes en los requisitos a lo largo del desarrollo, lo que resulta en un producto más alineado con las necesidades del cliente.
- **Metodología Tradicional (2):** Los cambios en los requisitos pueden ser costosos y difíciles de implementar una vez que el proyecto ha comenzado.

Como se observa en la tabla, la metodología Scrum obtuvo una calificación total de 33, en comparación con 20 de la metodología tradicional. Esto indica que Scrum es más eficaz en términos de flexibilidad, colaboración, entregas frecuentes y satisfacción del cliente.

3.2. Enfoque metodológico (Ágil Scrum)

Para el desarrollo de este aplicativo se ha escogido la metodología ágil Scrum, debido a que su marco de trabajo es ligero e iterativo; esto es ideal para proyectos de despliegue veloz y de alta iteración, lo que permite contar con un modelo MVP en un tiempo relativamente corto (Pyton, 2024).

Entre las principales características de esta metodología están: (Softeng, 2014)

- **Cumplimento de expectativas**: El cliente establece sus expectativas indicando el valor que le aporta cada requisito / historia del proyecto, el equipo los estima y con

esta información el Product Owner establece su prioridad, y comprueba que efectivamente los requisitos se han cumplido y transmite el feedback al equipo.

- **Flexibilidad a cambios:** Esta metodología tiene la capacidad de adaptarse a los cambios de requerimientos del cliente o evoluciones del mercado de manera rápida, enfocándose principalmente en proyectos complejos.
- **Reducción del Time to Market:** Característica permite al cliente utilizar las funcionalidades del proyecto sin necesidad de que este se haya finalizado.
- **Mayor calidad del software**: La metódica de trabajo y la necesidad de obtener una versión funcional después de cada iteración, ayuda a la obtención de un software de calidad superior.
- **Predicciones de tiempos:** Mediante esta metodología se conoce la velocidad media del equipo por Sprint (los llamados puntos historia), con lo que consecuentemente, es posible estimar fácilmente para cuando se dispondrá de una determinada funcionalidad que todavía está en el Backlog.

Esta metodología emplea ciclos cortos llamados sprints (al final de cada sprint se produce un incremento funcional) y cada uno de ellos presenta cuatro fases internas bien definidas: planificación, desarrollo, revisión y retroalimentación. El beneficio principal consiste en una entrega constante de resultados utilizables, brindando así la posibilidad de refinar el proyecto de forma continua

3.3.Justificación del enfoque escogido.

En un enfoque clásico que empieza desde el inicio, lo que dificulta llevar a cabo cambios posteriormente; en contraste, (Scrum) permite reinventar y modificar los

requisitos en cada sprint, presentar prototipos que ya están operativos y verificar constantemente con el cliente, lo que facilita la mejora rápida. (Capuñay, 2021). Otra razón importante para emplear Scrum es la colaboración dueño y programador.

La metodología promueve la participación activa del cliente durante todo el ciclo de desarrollo. Se ha demostrado que la comunicación constante entre los departamentos responsables del software y los usuarios finales reduce el riesgo de funcionalidades innecesarias, como se evidenció en un sistema de pagos para una fundación educativa, donde Scrum facilitó la interacción entre el ente educativo y su departamento financiero, asegurando que el producto final se ajustara al proceso real del usuario.

Esta metodología este diseño resulta ser una opción eficaz ya que no depende de una sociedad complicada. La información se la obtiene y se enfoca en historias de usuario y criterios de aceptación, y se da más importancia al software que funciona en lugar de a una documentación prolongada, lo que está en línea con el Manifiesto Ágil. (Cortés, 2023) También se consideró eXtreme Programming (XP) al elegir la metodología, valorando sus pruebas unitarias y su facilidad para proyectos más pequeños. De hecho, proyectos similares en el ámbito de la restauración han adoptado XP, dividiendo el trabajo en fases de planificación, diseño, desarrollo y pruebas. (Torres, 2020) Sin embargo, para Coffee Burguer se eligió Scrum debido a que proporciona una estructura de administración más integral deroles, eventos y artefactos claros y promueve una interacción constante con el cliente, balanceando de esta manera estructura y flexibilidad (Ciclick, 2024)

Para llevar a cabo un desarrollo ágil con Scrum, es necesario contar con ciertos roles fundamentales. Por ejemplo, el Product Owner que será el administrador deCoffee

Burguer, que actúa como representante del cliente se ocupa de organizar el backlog, que es la lista estructurada de tareas del proyecto; el Scrum Master se responsabiliza de facilitar el uso de la metodología y de remover obstáculos; y el equipo de desarrollo se dedica a crear los módulos y hacer ajustes conforme a las especificaciones establecidas (Pyton, 2024)

En proyectos pequeños, como el propuesto aquí, un mismo integrante puede asumir varios roles, mientras que el tutor actúa como Product Owner. Entre los artefactos esenciales figuran el Product Backlog, que prioriza necesidades del usuario, y el Sprint Backlog, que recoge las tareas de cada sprint. La dinámica es simple: tras cada ciclo se realiza una revisión y una retrospectiva que fomentan el trabajo colaborativo, la transparencia y la mejora continua. Para aplicar la metodología ágil en la aplicación de Coffee Burguer se contempla un Sprint 0 (planificación, levantamiento de requerimientos, diseño de arquitectura y setup del entorno), seguido de cuatro sprints de aproximadamente dos semanas cada uno para construir los módulos prioritarios.

Al concluir cada sprint, se llevará a cabo una presentación para el administrador con el fin de obtener una validación anticipada y comentarios, similar a lo que sucedió en un sistema web para una tienda de herramientas local que utilizó cuatro sprints con mejoras cuantificables. (Pyton, 2024) Este esquema de entrega incremental reduce riesgos y permite ajustes rápidos a las necesidades reales del negocio

3.4. Fases del desarrollo con objetivos específicos

Para fines de planificación es posible describir un proyecto de esta envergadura en fases o etapas análogas a los sprints iterativos propuestos en la metodología SCRUM.

A continuación, se detallan las fases principales en el desarrollo del software, incluyendo los objetivos específicos de cada fase. Estas fases se basan en la experiencia de proyectos similares recientes y se adecuan al contexto de Coffee Burguer:

Fase 1: En principio se construirá un análisis exhaustivo y se recopilarán los

Es esencial entender a fondo el proceso de encuadre de caja de Coffee Burguer.

Las tareas clave son:

- Entrevistas – conversación cara a cara con el administrador y el cajero, preguntando cuántas ventas manejan al día, qué hacen si se pierde una factura, cómo registran un gasto imprevisto, etc.;
- Observación de campo – un tercero mide cuántos minutos toma cada paso (contar efectivo, llenar una hoja de Excel o la libreta, firmar el reporte);
- Historias de usuario – frases simples como "Como cajero quiero subir la foto de la factura para no perderla", Tener estos requisitos claros y priorizados evita retrabajo y da evidencia de las variables que influirán en cada sprint por esto solo se contemplara pago en efectivo y trasferencias bancarias comprobadas en la banca virtual.

Fase 2: Diseño de la arquitectura y del modelo de datos.

Se adopta un patrón MVC con tres capas: presentación, lógica y datos. Esa división separa la pantalla del código de negocio y de la base de datos, haciendo el sistema más fácil de tocar a futuro (Chen, 2023). Capa de presentación: páginas PHP + Bootstrap para que el cajero vea formularios limpios y que en el móvil se vean bien. Capa de lógica:

controladores PHP que reciben la petición, calculan totales, validan datos y deciden qué vista mostrar. Capa de datos: MySQL con tablas sencillas (usuarios, roles, ventas, egresos, facturas). Se dibuja un diagrama ER donde ventas y egresos apuntan al usuario que registró la operación, y las facturas se ligan a los egresos.

- Flujos básicos: 1) Inicio de sesión → PHP revisa la tabla usuaria y, si la clave es correcta, manda al panel;
- 2) Carga de factura → se sube la imagen, el backend la guarda y registra la ruta en facturas;
- 3) Cuadre de caja → al final del día el sistema suma ventas menos egresos, muestra la diferencia y, si hay desajuste, avisa al administrador. Entregables: diagrama ER impreso, mockups de pantallas y diagramas de secuencia para esos tres flujos. Con esta división, si mañana se cambia MySQL por PostgreSQL o se retoca la interfaz, no hace falta reescribir todo el código (Raghu, 2025).

Fase 3 – Construcción con sprints iterativos (2 semanas cada uno)

Sprint 0 – Preparación del entorno

- Instalar XAMPP (Apache + PHP + MySQL).
- Crear la estructura MVC (carpetas, controllers, models, views).
- Prototipo de login con datos estáticos como base de prueba.

Sprint 1 – Autenticación y gestión de usuarios

- Formulario de login validado contra la tabla usuario.
- Dos roles mínimos: cajero y administrador.

- Interfaz CRUD para crear usuarios y asignar roles.
- Menús diferenciados según rol, reforzando la seguridad. Separar la autenticación en un sprint mejora la trazabilidad (Chen, 2023).

Sprint 2 – Registro de ventas y carga de facturas

- Formularios para fecha, monto y medio de pago.
- Subida de facturas en JPG/PNG/PDF con validación de tamaño y tipo.
- Almacenar archivos en el servidor y registrar metadatos en facturas.
- Listado de ventas/facturas con filtros por fecha o ID.

Objetivo: digitalizar evidencias y eliminar el papel; Banda (Leiva, 2019) reportó menos pérdidas de documentos con un módulo parecido.

Sprint 3 – Cuadre de caja y reportes

- Funciones PHP que calculan la diferencia ventas – egresos.
- Integración de Chart.js para:
- gráfico de barras (ingresos vs. Egresos diarios),
- gráfico de línea (tendencia mensual).
- Exportar reportes a Excel. En estudios análogos el tiempo de generación de reportes bajó de 12 min a ≈ 1 min. (Jiménez, 2024)

Sprint 4 – Pruebas, ajustes y despliegue en la nube

- Pruebas funcionales con cajeros y administrador.
- Pulir la interfaz (Bootstrap), mejorar mensajes de error.

- Configurar hosting PHP/MySQL en la nube (cPanel, Heroku o AWS), subir código, importar la base y activar HTTPS.

- Entregar manual de usuario y manual técnico. (Balarezo, 2021) describe un despliegue semejante para el proyecto “De Mar a Mar”, entregando un dominio a los empleados para facilitar la adopción del sistema.

Estas fases se resumen en la siguiente tabla, indicando el propósito principal de cada una:

Tabla 3

Fases.

Fase	**Descripción Actividades Principales**	**Objetivo Específico**
Fase 1: Análisis de Requerimientos	Recolección de necesidades con usuarios (entrevistas, historias de usuario); priorización de funcionalidades. Definición del alcance y criterios de éxito.	Entender el problema y qué debe resolver el sistema; obtener lista validada de requisitos.
Fase 2: Diseño de Arquitectura y DB	Diseño de la arquitectura en 3 capas (MVC); modelado de la base de datos; elaboración de diagramas UML (casos de uso, secuencia); planificación de componentes.	Establecer la solución técnica: cómo estará estructurado el sistema y cómo manejará los datos
Fase 3: Implementación (Desarrollo ágil)	Codificación iterativa en sprints de 2 semanas: construcción de módulos (autenticación, carga de facturas,	Construir el software de manera incremental, obteniendo retroalimentación continua y asegurando

	reportes). Reuniones SCRUM diarias (si aplica); revisión al final de cada sprint con stakeholders.	
Fase 4: Pruebas e Integración	Pruebas unitarias de cada módulo; pruebas integrales del sistema completo; pruebas con usuarios (aceptación). Ajuste de errores; optimización de rendimiento si es necesario.	Garantizar la calidad del aplicativo (funciona según lo esperado, sin fallos importantes) y que satisface las necesidades del usuario.
Fase 5: Despliegue y Cierre	Configuración de servidor en la nube; despliegue de la aplicación (carga de archivos PHP, importación de base de datos). Pruebas finales en entorno real; capacitación a usuarios finales; recopilación de feedback post-implementación.	Poner en marcha el sistema en el entorno productivo real de la empresa; asegurar la transición exitosa y la adopción por parte de los usuarios.

Cabe destacar que, aunque estas fases se presentan de forma secuencial para fines explicativos, en la práctica hubo solapamiento e iteración. Por ejemplo, el diseño podía refinarse en medio de la implementación, o algunas pruebas se realizaban continuamente durante el desarrollo (pruebas unitarias después de programar cada funcionalidad). Esto es consistente con el enfoque ágil adoptado, que no es estrictamente lineal sino adaptativo.
Cabe destacar que, aunque estas fases se presentan de forma secuencial para fines explicativos, en la práctica hubo solapamiento e iteración. Por ejemplo, el diseño podía refinarse en medio de la implementación, o algunas pruebas se realizaban continuamente durante el desarrollo (pruebas unitarias después de programar cada funcionalidad). Esto es consistente con el enfoque ágil adoptado, que no es estrictamente lineal sino adaptativo.

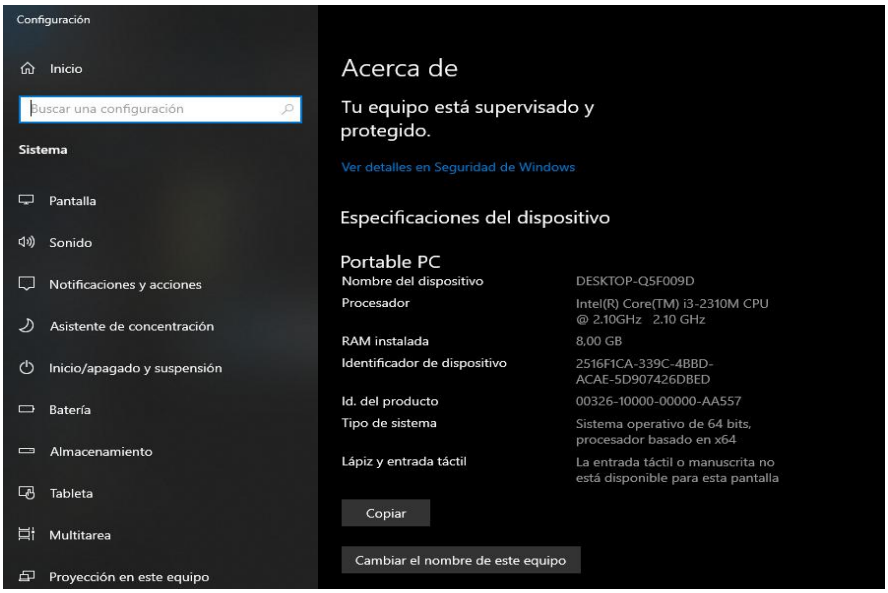
3.5. Herramientas y tecnologías utilizadas.

Para poder trabajar y utilizar el aplicativo es necesario tener con un equipo básico que servirá para poder utilizarlo a continuación detallaremos el equipo básico que el propietario debe de adquirir para el correcto funcionamiento.

Figura. 1 *características de un equipo para instalar el Software del aplicativo.*

Ilustración 1

Nota: Características básicas del equipo para el uso del aplicativo web



Sobre la instalación del aplicativo web, en nuestra portátil debe de tener las características básicas dadas para su correcta instalación y uso del aplicativo

- Intel(R) Core i3-2310M CPU @ 2.10GHz 2.10 GHz
- DISCO DURO DE 256 Gb.
- Memoria mínima de 8GB
- Sistema operativo de 64 bits, procesador basado en x64
- Windows 10 Home.

Una vez dadas las condiciones básicas para el funcionamiento del proyecto se trabajará en un conjunto de tecnologías accesibles para un desarrollador principiante,

cumpliendo con los requerimientos de costo (utilizando herramientas open-source) y facilidad de aprendizaje. En la siguiente tabla se listan las principales herramientas y tecnologías actuales empleadas, en el diseño de aplicativos web y bases de datos dejando claro que nuestro aplicativo será desarrollado con PHP. indicando su propósito dentro del desarrollo del aplicativo:

Tabla 2

Herramientas tecnológicas.

Tabla 4

Herramienta / Tecnología	**Descripción y Uso en el Proyecto**
PHP 8.x	Lenguaje de programación del lado del servidor utilizado para implementar la lógica del aplicativo web. Seleccionado por ser ampliamente difundido y adecuado para principiantes, con abundante documentación. Toda la funcionalidad de backend (procesamiento de formularios, autenticación, operaciones con base de datos) se codificará en PHP.
MySQL 8.x	Sistema gestor de bases de datos relacional donde se almacenará la información de la aplicación (usuarios, facturas, registros de cuadre, etc.). Se eligió MySQL por su fiabilidad y facilidad de uso; además se integra bien con PHP. El esquema de datos diseñado (ver fase 2) será implementado en MySQL.
Servidor Web Apache	XAMPP durante el desarrollo local y, posteriormente, en el servidor de producción (o un equivalente en la nube). Apache permite servir las páginas web de forma estable y es compatible con PHP.
Entorno de desarrollo XAMPP	Paquete de software libre que incluye Apache, MySQL, PHP y PHPMyAdmin. Se utilizó XAMPP en el entorno de desarrollo local para facilitar la instalación y configuración de todos los servicios necesarios. Esto permitió llevar a cabo

	pruebas del sistema en un PC personal antes del despliegue. Todos los servicios necesarios. Esto permitió efectuar pruebas del sistema en un PC personal antes del despliegue.
PHPMyAdmin	Herramienta web de administración de MySQL, provista junto con XAMPP. Empleada para gestionar la base de datos durante el desarrollo: creación de tablas, ejecución de scripts SQL, verificación de datos guardados, etc. Mejora la base de datos para programar una interfaz gráfica.
JavaScript (ES6) + AJAX	Lenguaje de programación del lado del cliente, usado para mejorar la interactividad de la aplicación. Se empleó JavaScript para validación de formularios en el cliente, actualizaciones dinámicas de contenido (p.ej., calcular subtotales en tiempo real al cargar facturas) y llamadas AJAX al servidor sin recargar la página (por ejemplo, para buscar facturas o actualizar datos de forma asíncrona). La biblioteca jQuery se utilizó en algunos casos para simplificar la manipulación del DOM y las peticiones AJAX.
Chart.js 3.x	Librería JavaScript de código abierto para generación de gráficos embebidos en páginas web. Se utilizó Chart.js para construir las visualizaciones de datos financieras en los reportes (gráficos de barras, líneas, pastel, etc.), aprovechando su fácil integración con datos provenientes de PHP (vía JSON) y su carácter interactivo. Esta librería ha sido recomendada para hacer reportes más digeribles e interactivos para los interesados.
Visual Studio Code (IDE)	Entorno de desarrollo integrado ligero, utilizado para escribir el código fuente PHP, HTML, CSS y JS. Se configuraron extensiones útiles (como soporte de PHP, depuración XDebug, etc.) para mejorar la productividad. Por su facilidad de uso y amplia comunidad, VSCode fue idóneo para un desarrollador principiante.

Control de versiones Git + GitHub	Sistema de control de versiones distribuido, utilizado de forma local para mantener un historial de cambios del código, y repositorio GitHub privado para respaldo remoto. Aunque el equipo es pequeño, usar Git ayudó a llevar registro de las evoluciones del proyecto y facilitó revertir cambios si era necesario.

Observamos que todas las tecnologías seleccionadas son libres o ampliamente disponibles, asegurando costo cero. La combinación PHP + MySQL es particularmente adecuada para este proyecto por ser una solución madura para desarrollo web, con abundantes ejemplos en la literatura técnica reciente. Adicionalmente, el uso de frameworks front-end (Bootstrap) y librerías de visualización (Chart.js) permite incorporar rápidamente funcionalidades avanzadas (diseño responsivo, gráficos) sin requerir un conocimiento experto en dichas áreas, cumpliendo con el criterio de accesibilidad para un desarrollador novel.

Es importante mencionar también que se siguieron buenas prácticas en la configuración del entorno: por ejemplo, en desarrollo se habilitó display_errors en PHP para facilitar el debut, mientras que en producción se deshabilitó y se configuró un log de errores; en MySQL se definieron usuarios con privilegios restringidos para la aplicación (principio de menor privilegio), etc. Estas prácticas aseguran una base tecnológica sólida para el proyecto.

3.6. Cronograma detallado de desarrollo.

Tabla 5

mes	julio				Agosto	
semana	Semana 1	Semana 2	Semana 3	Semana 4	Semana 5	Semana 5
Capitulo 1						
Capitulo 2						
Capitulo 3 y 4						
conclusiones						
recomendaciones						

En esta parte se muestra el calendario de actividades del proyecto, el cual va desde el comienzo de la fase de análisis hasta la implementación y finalización. Se empleó un gráfico de Gantt para organizar y representar las tareas a lo largo del tiempo, indicando las duraciones y el orden. La programación temporal se fundamentó en cerca de 2 meses de trabajo, lo que se traduce en un cronograma provisional para un proyecto de tesis de esta magnitud. En la tabla 2 se muestra el diagrama de Gantt del proyecto. En el eje vertical se listan las fases y entregables principales y en el eje horizontal se representan los meses y semanas del cronograma. Cada barra horizontal indica el período de ejecución de la actividad correspondiente. Las relaciones de precedencia se reflejan en la secuencia vertical de las barras (por ejemplo, el diseño del sistema comienza tras finalizar el análisis de requerimientos, la implementación se reparte en cuatro sprints sucesivos, etc.).

Como se observa en el calendario, el Estudio de Necesidades se programó para las primeras dos semanas (Julio 2025), seguido de dos semanas concentradas en el Diseño

del Sistema (segunda mitad de Julio). La Ejecución se segmentó en cuatro sprints quincenales durante agosto del 2025, donde cada sprint proporcionaba módulos operativos.(Business, 2021)

Luego, a inicios de septiembre, se destinaron aproximadamente dos semanas para Pruebas e Integración de todos los componentes desarrollados. Finalmente, la última quincena de septiembre 2025 se reservó para el Despliegue y Capacitación, es decir, el entrenamiento a los usuarios de Coffee Burguer en el uso de la nueva herramienta.

A continuación, se detalla brevemente el contenido de cada bloque temporal del cronograma:

- Mes 1 (mayo 2025): Comprende la Fase 1 y Fase 2. Semanas 1-2: Análisis de Requerimientos (recolección de datos, estudio de procesos de caja actuales, especificación de historias de usuario). Semanas 3-4: Diseño del Sistema (arquitectura, modelo de datos y prototipos de interfaz). Para el cierre de este mes, se esperaba tener claros todos los requisitos y un diseño aprobado, de forma que el siguiente mes iniciara la codificación con una visión bien definida.
- Mes 2 (junio 2025): Comprende Sprints 1 y 2 de la Fase 3 (Implementación). Semanas 5-6: Sprint 1 enfocado en autenticación y gestión de usuarios. Semanas 7-8: Sprint 2 enfocado en funcionalidad de carga de facturas y registro de ventas. Al finalizar este mes, el sistema ya contaría con las funcionalidades básicas de seguridad (login/roles) y entrada de datos (input de ventas/facturas), permitiendo a mitad del desarrollo tener un producto mínimo viable utilizable internamente para pruebas.
- Mes 3 (julio 2025): Comprende Sprints 3 y 4 de Implementación. Semanas 9-10: Sprint 3 dedicado al módulo de cuadre de caja y generación de reportes con gráficos.

Semanas 11-12: Sprint 4 dedicado a integraciones finales, refinamientos de la interfaz y preparación de pruebas. Al concluir septiembre, prácticamente todo el desarrollo funcional estaría completo. Esto concuerda con experiencias similares donde la construcción tomó alrededor de tres meses con metodologías ágiles.

- Mes 4 (agosto 2025): Últimas fases. Semanas 13-14: Pruebas e Integración del sistema completo. Se realizaron pruebas de usuario con datos reales simulados y se corrigieron defectos. Semanas 15-16: Despliegue en el servidor en la nube y capacitación al personal de Coffee Burguer. En la última semana se efectuó el go-live oficial del aplicativo, quedando operativo para el cierre de caja diario real en la empresa.

Este cronograma fue elaborado de manera realista considerando tiempos holgados para ajustes. Durante la ejecución, se hizo un seguimiento semanal de las tareas; algunas variaciones menores ocurrieron (por ejemplo, el Sprint 3 se extendió unos días más de lo previsto para afinar los gráficos de reportes). No obstante, en general se logró cumplir con los hitos establecidos. El uso de un diagrama de Gantt facilitó la gestión del tiempo y la comunicación del progreso al tutor de tesis, sirviendo como herramienta de control. De acuerdo a Atlassian, los cronogramas visuales tipo Gantt son útiles incluso en entornos ágiles para tener una vista macro de hitos y dependencias (aunque SCRUM no los exija formalmente, pueden complementar la planificación

3.7. Consideraciones éticas y de seguridad de la información.

En cualquier proyecto de tecnología que involucra datos financieros y operativos de una empresa, surgen importantes consideraciones *éticas* y responsabilidades en el manejo de la información. Coffee Burguer confiará datos sensibles al aplicativo (p. ej.,

montos de ventas diarias, archivos de facturas con información de clientes o proveedores, credenciales de usuarios), por lo cual se tomaron medidas para asegurar la confidencialidad, integridad y disponibilidad de dicha información.

Una primera medida ética-práctica fue implementar controles de acceso estrictos:

solo personal autorizado puede ingresar al sistema y cada usuario solo ve las funciones y datos según su rol (segregación de privilegios). Esto previene que información financiera delicada esté al alcance de quien no corresponda. Tal como recomienda la literatura de seguridad, es esencial controlar y auditar el acceso a los datos para garantizar su uso apropiado.

El sistema contara con una contraseña que permitirá detectar si alguien quiere ingresar sin permiso ya que dará solo dos opciones para intentar ingresar la contraseña esto dado por el dueño del local y puesto a consideración para su desarrollo Además, se reducirá los datos: el sistema guarda únicamente las ventas; no guarda o almacena en su base de datos información como usuarios anteriores ni recopila datos del personal innecesaria de clientes u otras personas, lo que disminuye el riesgo en caso de que se produzcan filtraciones. (Capuñay, 2021)

Las facturas cargadas, por ejemplo, pueden contener datos tributarios; estos archivos se almacenan en el servidor, pero no son expuestos públicamente, y se planificó una política de retención de datos consistente con las necesidades contables (por ejemplo, conservar comprobantes digitales al menos por el periodo fiscal requerido y luego archivarlos de forma segura fuera de línea).

La protección de la información se abordó también desde un enfoque técnico: el servidor fue ajustado para emplear conexiones seguras (HTTPS), impidiendo la

interceptación de datos mientras se mueven (particularmente importante al acceder desde redes ajenas). Además, las claves de acceso de los usuarios en la base de datos se guardan mediante algoritmos de hash confiables (Chen, 2023). De esta forma, aunque alguien lograra ingresar a la base de datos, no sería sencillo

para esa persona acceder a las claves en formato legible. Se implementaron copias de seguridad periódicas tanto de la base de datos como del repositorio de código, asegurando que se mantenga la disponibilidad y la posibilidad de restauración en caso de errores (esto también forma parte de la obligación ética: garantizar la continuidad del servicio en beneficio de los usuarios). Otro aspecto es la privacidad y cumplimiento legal. Si bien Coffee Burguer es una empresa local sin un gran volumen de datos personales, se consideraron los lineamientos de normativas como la Ley de Protección de Datos Personales (en el contexto local) y, por analogía, principios del GDPR europeo para el manejo de datos. Por ejemplo, se aseguró que solo los propietarios (la empresa) tienen acceso a los datos brutos de ventas; el desarrollador y terceros no conservarán copias de datos críticos tras concluido el proyecto. Cualquier acceso de mantenimiento al sistema en producción se hará con autorización de la empresa y respetando la confidencialidad de su información. Mantener la seguridad y privacidad de los datos financieros de los usuarios no solo es una medida ética, sino también un requisito legal en muchas jurisdicciones, por lo cual estas precauciones mitigan riesgos legales y reputacionales. En cuanto a la ética profesional, el desarrollo se llevó a cabo respetando los

estándares de propiedad intelectual de las herramientas usadas (todas son de licencia libre o comunitaria) y citando apropiadamente cualquier fragmento de código o diseño inspirado en fuentes externas. No se incurrió en uso indebido de activos de terceros sin permiso (por ejemplo, librerías como Bootstrap y Chart.js se usan bajo sus licencias open-

source adecuadas). Esto va en línea con actuar con honestidad y profesionalismo, aspectos cubiertos en códigos de ética en ingeniería de software.

Por último, se tomó en cuenta la moralidad al implementar el sistema entre los empleados: siempre hay un periodo de adaptación y posibles oposiciones al cambio cuando se mecaniza un procedimiento tradicional. Para solucionar esto, se llevó a cabo la capacitación de forma considerada, subrayando que la herramienta está destinada a facilitar su labor y no a supervisarla. Se creó un medio de comunicación para que los cajeros pudieran expresar inquietudes o recomendaciones durante la fase de adaptación inicial, promoviendo un clima de confianza. (Jiménez, 2024)

Se cuidó también que la implementación no violara la privacidad individual: los logs de auditoría de acceso registran usuarios por razones de seguridad, pero esos datos solo se revisarán ante incidencias, no para monitoreo invasivo.

En resumen, las consideraciones éticas del proyecto giraron en torno a proteger la información financiera de Coffee Burguer y respetar a las personas involucradas (usuarios finales, clientes cuyos datos pudieran figurar en facturas, etc.). El cumplimiento de buenas prácticas de seguridad informática (control de acceso, encriptación, integridad de datos) respaldó estos objetivos técnicos. Adicionalmente, se actuó con responsabilidad profesional y transparencia durante todo el desarrollo. De esta forma, el proyecto no solo logra sus metas funcionales, sino que lo hace de manera ética, protegiendo los intereses y la confianza de la empresa y sus stakeholders.

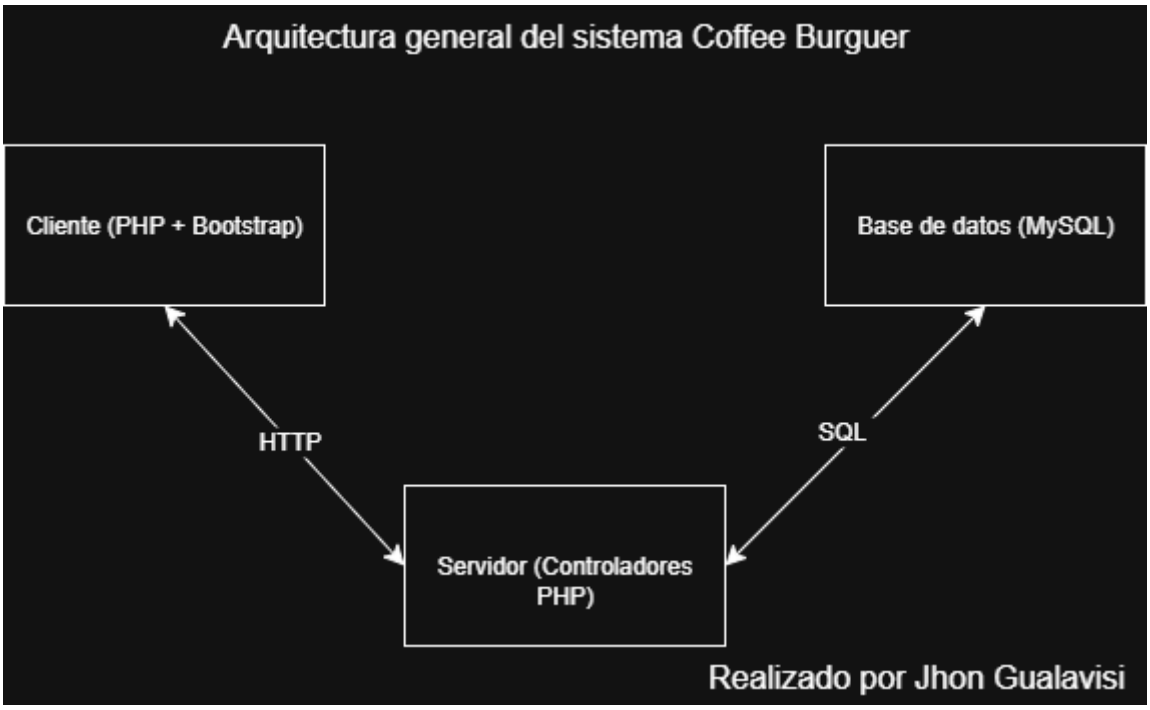
4. CAPITULO IV

4.1. Arquitectura del sistema

En esta sección se explicará cómo está estructurada la solución Coffee Burguer a nivel de componentes y capas indicada en la **ilustración 2**, tomando como base los objetivos y la metodología ya expuestos en los capítulos anteriores, sabiendo que cada estructura depende mucho de proyecto que se está realizando.

Ilustración 2

Diagrama de arquitectura general



Nota: diagrama hecho propio en 2025

Descripción de la arquitectura

Capa de presentación (cliente)

La capa de presentación se implementa íntegramente en PHP 8.x, apoyada por Bootstrap 5 para garantizar una experiencia responsiva y uniforme en distintos dispositivos. Todos los archivos de vista se alojan en la carpeta `/views/`, donde cada pantalla el login, registro de ventas, cuadre de caja, reportes y gestión de egresos corresponde a un script PHP que combina plantillas HTML con fragmentos de código de renderizado. Los usuarios interactúan mediante formularios y tablas dinámicas y, antes de enviar cualquier dato al servidor, se efectúan comprobaciones básicas en JavaScript para validar campos obligatorios (por ejemplo, verificar que el campo "monto" o "cantidad" no esté vacío). Además, los recursos estáticos (CSS y JS de Bootstrap, scripts propios) se organizan bajo `/public/`, mientras que Los archivos de factura subidos por el cajero se guardan en la carpeta `public/facturas/` de la raíz del proyecto, manteniendo sus nombres originales (ejemplo. `IMG_20250210_162628.jpg`) para facilitar su recuperación y visualización en los listados.

Capa de negocio (servidor)

La lógica central del sistema se encuentra en controladores PHP que dirigen el flujo de datos y aplican las reglas de negocio. Cada entidad principal dispone de su propio controlador por ejemplo, para usuarios, productos, ventas, egresos y cuadre de caja, que recibe las peticiones HTTP y delega en los modelos correspondientes. Estos modelos, ubicados en `/models/`, encierra el acceso a datos mediante PDO y consultas preparadas, garantizando la inyección segura de parámetros. En esta capa también se implementa la gestión de sesiones: tras un inicio de sesión exitoso, se almacena en `$_SESSION` el rol del usuario (cajero o administrador) y se aplica un filtro de autorización en cada recurso. La subida y validación de imágenes utiliza chequeos de tipo MIME y tamaño máximo,

guardando los archivos en un directorio seguro con nombres únicos para evitar colisiones y accesos no autorizados.

Capa de datos (MySQL)

El almacenamiento de información se realiza en un servidor de base de datos MySQL (gestionado mediante XAMPP), lo que permite transacciones atómicas y refuerza la integridad referencial mediante claves foráneas. La codificación **utf8mb4** asegura compatibilidad con caracteres especiales en nombres, descripciones y preguntas de seguridad. El esquema de base de datos incluye tablas fundamentales como `usuarios`, `productos`, `pedidos`, `pedido_detalle`, `egresos` y `cuadre_caja`. Cada tabla define su clave primaria auto incremental y establece relaciones de varios a uno, con las tablas correspondientes (por ejemplo, `pedido_detalle.producto_id` enlaza con `productos.id`). Además, se han añadido índices a campos de consulta frecuente (`email`, `fecha`, `usuario_id`) para optimizar el rendimiento de las consultas críticas.

Flujo de operación: registro de una venta

Cuando un cajero inicia una nueva venta, la interfaz web desarrollada en PHP y Bootstrap carga automáticamente un formulario con el catálogo de productos y solicita al servidor la información actualizada de recetas e insumos. Al completar las cantidades deseadas y enviar el formulario, el cliente emite una petición POST al VentaController, que a su vez trae el método verificarStockDisponible() del modelo Venta. Este método consulta las tablas recetas e insumos para confirmar que existe los insumos suficientes para cada producto mediante la receta ya definida. Si en algún momento detecta falta de stock, el servidor devuelve un mensaje de alerta al cajero indicándole que no puede agregar. En caso contrario, el controlador abre una transacción y procede a insertar el encabezado de la venta en la tabla pedidos, registra cada línea de producto en

pedido_detalle y actualiza los saldos de insumos restando las cantidades utilizadas según las recetas, para todo es importante que elija el tipo de pago ya sea efectivo o transferencia Al finalizar la transacción, el sistema confirma al cajero el éxito de la operación y muestra un mensaje de validación.

Al concluir la jornada laboral, el administrador accede al módulo de Cuadre de Caja, que agrupa las ventas diarias registradas en pedidos por método de pago (efectivo y transferencia) y calcula el total de egresos consultando los registros de la tabla egresos para la fecha correspondiente. A partir de esos valores, el sistema calcula en memoria el saldo final restando los egresos al total de ventas y presenta un informe detallado en pantalla. Además, ofrece la opción de exportar ese reporte a Excel, facilitando la auditoría y el mantenimiento del historial financiero sin necesidad de almacenar datos adicionales en la base de datos.

4.2. Diseño de la base de datos

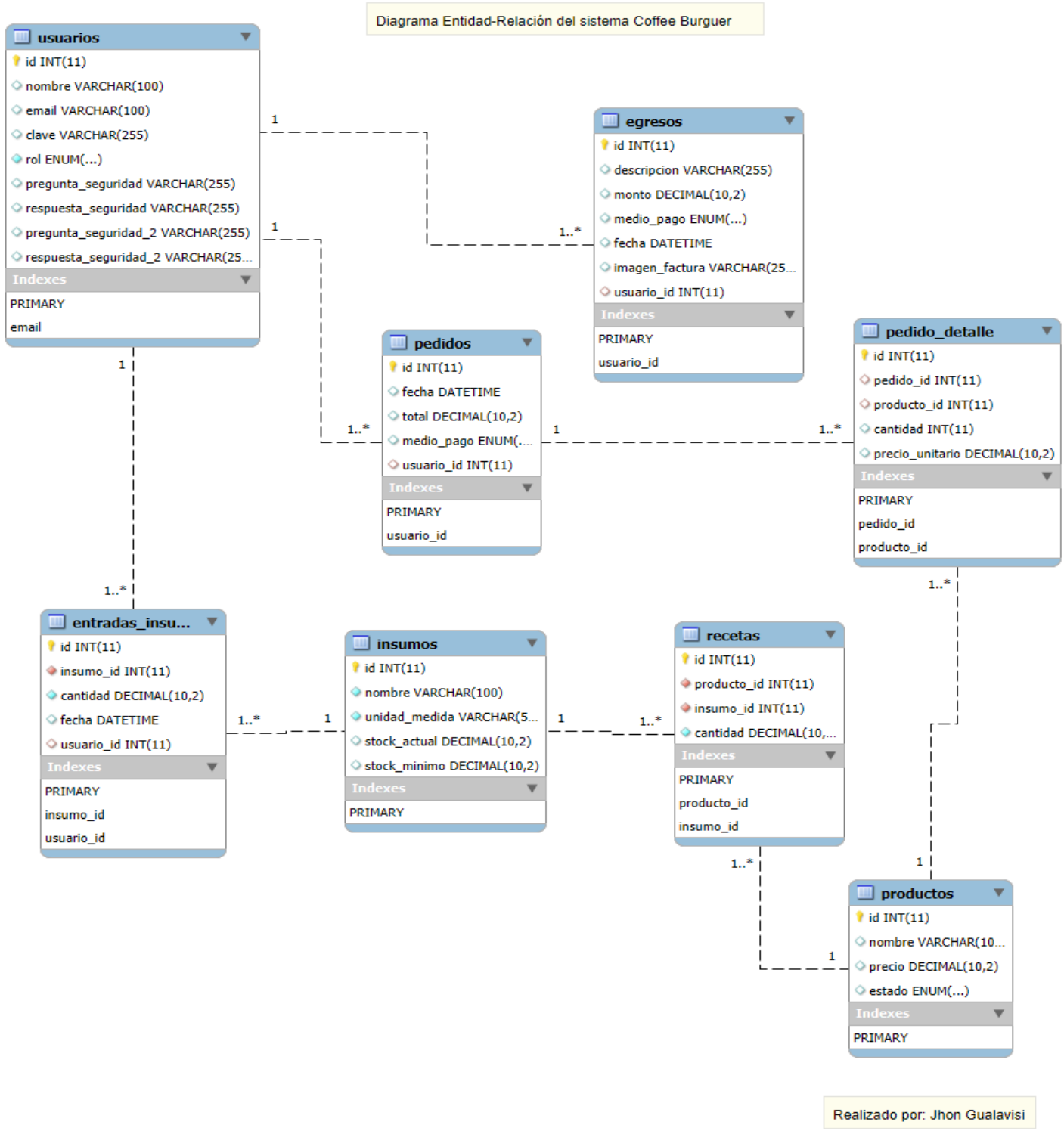
Para esta parte del proyecto detallamos cómo se tradujeron los requisitos funcionales y no funcionales de Coffee Burguer al esquema de base de datos que sustenta toda la aplicación. Partiendo de la necesidad de mantener la integridad de la información (por ejemplo, que cada venta se asocie siempre a un usuario válido, o que no se puedan eliminar insumos en uso), se optó por un modelo relacional normalizado en **MySQL**, lo que garantiza transacciones no incompletas, soporte nativo de claves foráneas y bloqueo a nivel de fila para maximizar la concurrencia. Asimismo, la codificación **utf8mb4**

asegura que todos los nombres, descripciones y preguntas de seguridad puedan incluir caracteres especiales (tildes, emojis, etc.) sin pérdida de datos.

Para reflejar visualmente esta estructura, a continuación, se presenta en la **ilustración 2 Diagrama Entidad-Relación (DER)** generado mediante la ingeniería inversa de nuestro esquema en MySQL Workbench. Este diagrama incluye las ocho entidades principales `usuarios`, `productos`, `insumos`, `recetas`, `pedidos`, `pedido_detalle`, `egresos` y `entradas_insumo` y sus interconexiones, reflejando de forma inequívoca las reglas de negocio: un usuario puede registrar múltiples ventas y egresos; cada pedido consta de varias líneas de detalle; cada producto se elabora a partir de uno o más insumos, y así sucesivamente.

Ilustración 3

Diagrama Entidad – relación



Nota: diagrama hecho propio en 2025 en Workbench.

En el DER, cada relación padre - hija muestra un **"1"** en la entidad que controla la asociación y un **"1..*"** (uno a muchos) en la entidad dependiente, reforzando visualmente que, por ejemplo, un solo registro de `usuarios` puede vincularse con varios registros de `pedidos` o de `egresos`. Las líneas sólidas subrayan la dirección y cardinalidad de cada vínculo, lo cual facilita tanto la comprensión de la lógica de datos como la detección de posibles anomalías en la fase de análisis.

4.3. Diccionario de datos

A continuación, se describe la forma detallada cada una de las ocho tablas principales, explicando el propósito de la entidad, el significado de cada campo, sus restricciones y por qué son necesarios en el modelo de datos de Coffee Burguer, que relevancia existe.

Tabla 6

Descripción de las tablas de la base de datos coffee_burguer

Tabla	Campo	Tipo	Restricciones	Descripción y justificación
usuarios	id	INT(11) PK AUTO_INCREMENT	PRIMARY KEY	Identificador único. Garantiza que cada usuario se distinga inequívocamente.
	nombre	VARCHAR(100)	NOT NULL	Nombre completo; se muestra en el navbar y en formularios de auditoría.
	email	VARCHAR(100) UNIQUE	NOT NULL, UNIQUE	Clave de login la restricción UNIQUE impide cuentas duplicadas.
	clave	VARCHAR(255)	NOT NULL	Almacena el hash de la contraseña (mediante password_hash()) para nunca guardar texto plano.
	rol	ENUM('admin','cajero')	NOT NULL, DEFAULT 'cajero'	Controla permisos: solo 'admin' puede eliminar o editar registros críticos y generar cuadre de caja.
	pregunta_seguridad	VARCHAR(255)	NOT NULL	Permite recuperación de contraseña; cada pregunta se guarda junto a un hash de la respuesta.

	respuesta_seguridad	VARCHAR(255)	NOT NULL	Hash de la respuesta a la pregunta de seguridad, para validar sin exponer texto real.
insumos	id	INT(11) PK AUTO_INCREMENT	PRIMARY KEY	Identificador único de materia prima.
	nombre	VARCHAR(100)	NOT NULL, INDEX	Denominación del insumo; índice para acelerar consultas en recetas y gestión de stock.
	unidad_medida	VARCHAR(50)	NOT NULL	Unidad de control (kg, L, unidades), necesaria para los cálculos de stock.
	stock_actual	DECIMAL(10,2)	NOT NULL, DEFAULT 0.00	Nivel de inventario disponible; se actualiza en cada venta y reposición.
	stock_minimo	DECIMAL(10,2)	NOT NULL, DEFAULT 0.00	Umbral que dispara alertas de reposición en la UI del administrador.
recetas	id	INT(11) PK AUTO_INCREMENT	PRIMARY KEY	Identificador interno.
	producto_id	INT(11) FK	FOREIGN KEY productos.id ON DELETE CASCADE	Relaciona cada receta con un producto; si un producto se elimina, sus recetas también.
	insumo_id	INT(11) FK	FOREIGN KEY → insumos.id ON DELETE RESTRICT	Vincula el insumo necesario; RESTRICT evita eliminar insumos en uso.
	cantidad	DECIMAL(10,2)	NOT NULL, CHECK (cantidad > 0)	Cantidad de insumo requerida para una unidad de producto; valida que sea positiva.
pedidos	id	INT(11) PK AUTO_INCREMENT	PRIMARY KEY	Identificador de la transacción de venta.
	fecha	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Marca temporal, crucial para filtros diarios en cuadre y reportes.
	total	DECIMAL(10,2)	NOT NULL, CHECK (total >= 0)	Suma de subtotales. El CHECK refuerza que no sea negativo.

	medio_pago	ENUM('efectivo','transferencia')	NOT NULL	Método de pago; utilizado en el módulo de cuadre para separar totales.
	usuario_id	INT(11) FK	FOREIGN KEY usuarios.id ON DELETE SET NULL	Registra qué cajero ejecutó la venta; SET NULL si el usuario se elimina (poco común).
Pedido_detalle	id	INT(11) PK AUTO_INCREMENT	PRIMARY KEY	Línea individual dentro de pedidos.
	pedido_id	INT(11) FK	FOREIGN KEY pedidos.id ON DELETE CASCADE	Asegura eliminación en cascada de detalles al borrar un pedido.
	producto_id	INT(11) FK	FOREIGN KEY productos.id ON DELETE RESTRICT	Vincula la línea al producto vendido; RESTRICT evita borrar productos históricos.
	cantidad	INT	NOT NULL, CHECK (cantidad > 0)	Unidades vendidas; CHECK garantiza al menos una unidad.
	precio_unitario	DECIMAL(10,2)	NOT NULL, CHECK	Precio aplicado; permite conservar históricos aunque el precio cambie luego.
egresos	id	INT(11) PK AUTO_INCREMENT	PRIMARY KEY	Identificador del gasto operativo.
	descripcion	VARCHAR(255)	NOT NULL	Motivo o detalle del egreso.
	monto	DECIMAL(10,2)	NOT NULL, CHECK (monto >= 0)	Valor económico del egreso. Avoid valores negativos.
	medio_pago	ENUM('efectivo','transferencia')	NOT NULL	Igual que en ventas, para comparar saldos.
	fecha	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Fecha de registro; usada en cuadre y filtros de reporte.
	imagen_factura	VARCHAR(255)	NULL	Ruta en public/facturas/; permite mostrar evidencia documental.
	usuario_id	INT(11) FK	FOREIGN KEY → usuarios.id ON DELETE SET NULL	Identifica quién registró el egreso; SET NULL si el usuario desaparece.

Entradas _insumo	id	INT(11) PK AUTO_INCREMENT	PRIMARY KEY	Identificador de reposición de insumo.
	insumo_id	INT(11) FK	FOREIGN KEY → insumos.id ON DELETE RESTRICT	Asocia la reposición al insumo; RESTRICT evita borrado de insumos con histórico.
	cantidad	DECIMAL(10,2)	NOT NULL, CHECK (cantidad > 0)	Cantidad agregada al inventario.
	fecha	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Fecha de la reposición; importante para auditoría de inventario.
	usuario_id	INT(11) FK	FOREIGN KEY → usuarios.id ON DELETE SET NULL	Usuario que validó o realizó la reposición; SET NULL si ya no existe
producto	id	INT(11)	NOT NULL	Identificador único del producto.
	nombre	VARCHAR(100)	NOT NULL	Nombre descriptivo del producto.
	precio	DECIMAL(10,2)	NOT NULL	Precio de venta por unidad; CHECK en la BD evita negativos.
	estado	ENUM('activo','inactivo')	NOT NULL	Control de disponibilidad: 'activo' o 'inactivo'.

4.4. diseño de la interfaz (UI/UX)

Como se indica en la **ilustración 4** en esta sección describimos con detalle cada una de las vistas principales de Coffee Burguer, explicando que propósito tiene cada una, interacción clave, validaciones y cómo encajan en el flujo de trabajo diario. Para cada pantalla se indica la ubicación del archivo PHP correspondiente y el nombre sugerido para la captura de pantalla.

Archivo: /views/login.php

Lo cual tiene como propósito Autenticar al usuario (cajero o administrador) antes de permitir el acceso al sistema. Y tiene como componente el formulario que está centrado con campos email y password. Y el botón “Iniciar sesión” en caso de alertas de Bootstrap (alert-danger) al no coincidir las indicara credenciales inválidas.

Validaciones y feedback con la validación con HTML5 (required, type="email") comprueba sintaxis mínima antes de enviar (formato correo). Si el servidor responde con error, se renderiza un div.alert mostrando el mensaje exacto, Si el login es exitoso, redirige a dashboard.php; si falla, se mantiene en login.php con alerta.

Ilustración 4

Pantalla principal del login en coffee burguer

Archivo: `/views/dashboard.php`

Al ingresar al sistema, el usuario es recibido por el dashboard, cuya plantilla

principal se encuentra en dashboard.php (complementada por header.php). En la parte superior, un **navbar fijo** facilita el acceso inmediato a los módulos esenciales: Ventas, Egresos, Cuadre de Caja, Reportes y la opción de Cerrar Sesión. Debajo, indica la ubicación actual dentro de la aplicación, reforzando la orientación del usuario al navegar por submenús. En el centro de la pantalla, una **tarjeta de bienvenida** muestra el nombre del usuario activo junto a la fecha y hora del servidor, ofreciendo un entorno personalizado y confiable. Gracias a este diseño, el menú permanece siempre visible, posibilitando transiciones ágiles entre tareas sin perder el contexto, mientras que el breadcrumb acompaña y explica el flujo de trabajo, de modo que al hacer clic en “Ventas” se carga inmediatamente la vista nueva_venta.php sin pasos intermedios innecesarios.

nota: la pantalla principal depende que usuario hizo login, por tema de permisos

Ilustración 5

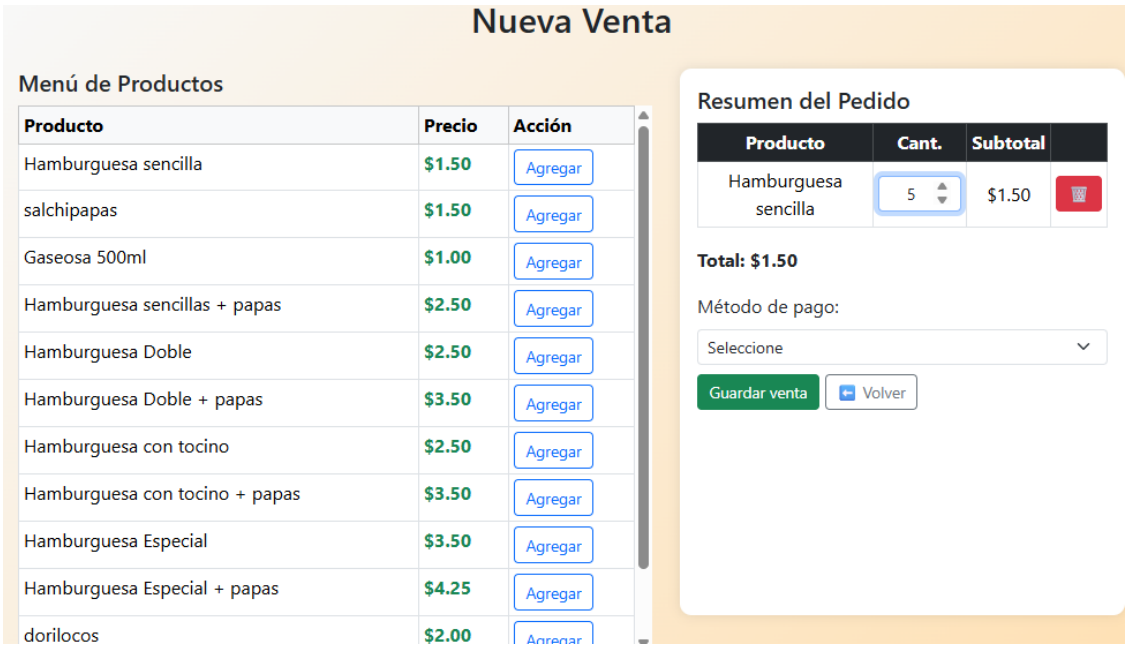
Pantalla de todos los módulos activos para el administrador

Archivo: `/views/nueva_venta.php`

el módulo de "Nueva Venta" como muestra la **ilustración 6** está concebida como el núcleo operativo para el cajero un desplegable de productos cargado por PHP muestra el catálogo actualizado, junto a un campo numérico donde se indica la cantidad. Al pulsar "Agregar", esa línea se añade dinámicamente a una **tabla de detalle**, que muestra columnas de Producto, Cantidad, Precio unitario y Subtotal, y recalcula en todo momento el **total acumulado** en el pie de la tabla. Para evitar errores de inventario, una validación instantánea comprueba el stock disponible; si la cantidad excede el límite, el campo recibe la clase `que es incalido` y despliega un mensaje para que el usuario corrija. El botón "Registrar Venta" permanece deshabilitado hasta que exista al menos una línea válida, garantizando que no se envíen ventas incompletas. Cuando el cajero confirma, el sistema invoca el controlador y, tras procesar la transacción, muestra un **toast** de éxito o error según corresponda.

Ilustración 6

Pantalla principal para realizar una venta

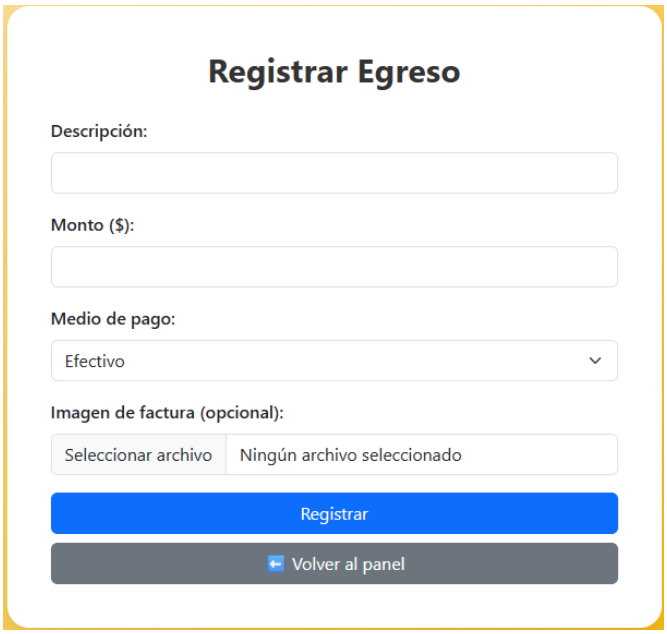


Archivo: `/views/egresos.php`

En la gestión de egresos (`egresos.php`) muestra en la **ilustración 7**, el formulario principal permite capturar la descripción del gasto, el monto, el método de pago y adjuntar una imagen de la factura mediante un `<input type="file">` estilizado. Los archivos se validan en cliente y servidor, rechazando formatos no permitidos o tamaños excesivos antes de la subida. Bajo el formulario, una **tabla responsiva** lista los egresos del día, mostrando miniaturas de las facturas junto a los datos relevantes. En cada fila, el administrador dispone de botones "Editar" y "Eliminar": al pulsar "Editar" se abre un **modal** repoblado con los valores actuales, y al elegir "Eliminar" el sistema solicita confirmación en otro diálogo, asegurando trazabilidad y control de cambios. Tras cualquier operación, la tabla se actualiza automáticamente, proporcionando al usuario una experiencia fluida y segura.

Ilustración 7

Pantalla principal del registro de egresos

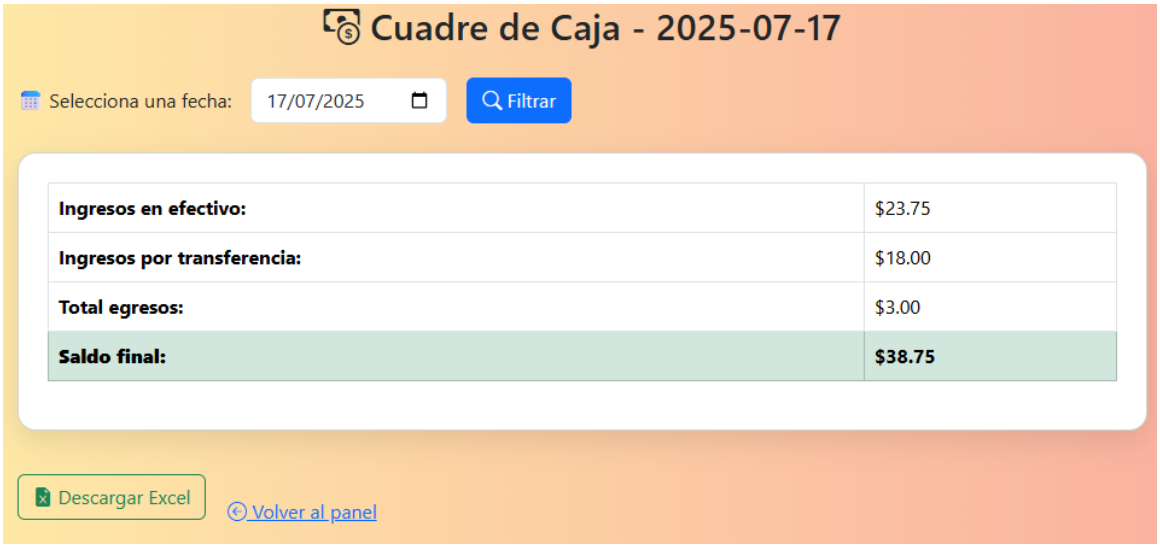


Archivo: `/views/cuadre_caja.php`

El módulo de Cuadre de Caja que se presenta en la **ilustración 8** y Reportes se organiza en dos pestañas dentro de `cuadre_caja.php`. La primera, **"Resumen"**, presenta tres tarjetas de Bootstrap con los totales de ventas en efectivo, ventas por transferencia y egresos del día, calculados por el controlador al cargar la vista. Debajo, un **gráfico de barras** creado con Chart.js compara visualmente las entradas y salidas, ofreciendo al administrador una visión inmediata del balance diario. La segunda pestaña, **"Exportar"**, incluye el botón "Descargar Excel", que al activarse solicita al servidor la generación de un archivo `.xlsx` via PhpSpreadsheet; durante este proceso, un spinner indica progreso y, al finalizar, aparece un toast de confirmación. Gracias a esta estructura, el administrador puede tanto consultar en pantalla el estado financiero como obtener un informe formal en un solo lugar, optimizando el cierre de jornada.

Ilustración 8

Pantalla que indica egresos tipo de pagos y saldo total



5. CAPITULO V

5.1.Procedimientos de validación del sistema.

La calidad y correcto funcionamiento del aplicativo web fueron verificados mediante una serie de pruebas y validaciones sistemáticas, apoyándose tanto en métricas objetivas como en retroalimentación de usuarios finales. Los procedimientos de validación abarcaron varios niveles

- Pruebas unitarias y de integración técnica: A medida que se fueron desarrollando las distintas funcionalidades, el desarrollador ejecutó pruebas unitarias informales sobre cada componente (por ejemplo, probar por separado la función de cálculo de saldo de caja con distintos conjuntos de datos de entrada, verificar que el módulo de autenticación rechaza correctamente credenciales inválidas, etc.). Posteriormente, en la fase 4 (Pruebas e Integración), se llevaron a cabo pruebas de integración que cubrieron los flujos completos del sistema. Por ejemplo, se simuló el proceso entero de un cierre de caja: inicio de sesión como cajero, registro de varias ventas con facturas, cierre de sesión, inicio de sesión como administrador, generación de reporte diario y mensual. Estas pruebas end-to-end permitieron verificar que todos los módulos interactúan correctamente (frontend ⇔ backend ⇔ base de datos) y que los datos fluyen sin inconsistencias. Se documentaron casos de prueba cubriendo escenarios normales y excepcionales, incluyendo pruebas de campos obligatorios, límites (e.g., subir un archivo de factura de tamaño mayor al permitido para ver el manejo del error), y pruebas de concurrencia básica (dos usuarios accediendo simultáneamente). Cualquier defecto detectado fue registrado y corregido antes del despliegue. Este enfoque se alinea con prácticas reportadas en proyectos similares donde “al finalizar la

programación se realizaron las pruebas necesarias para verificar el correcto funcionamiento" del sistema, asegurando que se cumplen todos los requisitos funcionales.

- Aceptación de los dueños y exámenes de validación: el propietario pidió exclusivamente que como duela solo el tener acceso a diseñar los menús y el cierre de caja un usuario final solo podrá ver los menús sin modificar y cerrar caja de Coffee Burguer. Se coordinó una sesión en la que un cajero y el administrador interactuaron con el sistema utilizando datos simulados (como registros de ventas de un día habitual), mientras el desarrollador supervisaba y anotaba cualquier problema de usabilidad o comportamiento no normal.

  Se preparó un checklist de criterios de aceptación derivados de las historias de usuario para comprobar uno a uno (p.ej., "El cajero puede cargar una factura y visualizarla luego en el listado de facturas del día" – Cumplido, "El sistema impide al cajero acceder al módulo de reportes de administrador" – Cumplido, etc.). Estas pruebas de aceptación confirmaron que el sistema satisface los requerimientos acordados. Adicionalmente, se aplicó una encuesta de satisfacción a los usuarios tras probar la aplicación, para evaluar su experiencia en cuanto a facilidad de uso, rendimiento y utilidad percibida. Los resultados fueron muy positivos: se obtuvo un 100% de respuestas de conformidad indicando que el personal estuvo totalmente de acuerdo en que la nueva aplicación mejora y agiliza el proceso de cuadre de caja. Este método de encuestar a usuarios es común en validaciones de sistemas, aunque idealmente debería enmarcarse en métodos formales de evaluación de usabilidad. En nuestro caso, sirvió para recoger

impresiones cualitativas valiosas y confirmar la aceptación del sistema en el ambiente operativo real.

- Métricas de desempeño y mejora: Más allá de comprobar que "funciona según lo esperado", se midieron algunos indicadores clave para cuantificar los beneficios del aplicativo en comparación con el proceso anterior manual. Por ejemplo, se cronometró el tiempo promedio que toma realizar el cuadre de caja al final del día con el sistema, y se comparó contra el tiempo que tomaba hacerlo manualmente. Los resultados mostraron una mejora significativa, en estudios similares se halló una reducción de más del 90% en el tiempo de elaboración del reporte de caja con la herramienta informática. En Coffee Burguer, inicialmente el cierre de caja podía tomar alrededor de 15-20 minutos; con el nuevo sistema, se comprobó que puede obtenerse el reporte en menos de 2 minutos, ya que el cálculo es automático y las facturas ya están registradas digitalmente. Otra métrica evaluada fue la reducción de errores: se comparó el número de inconsistencias o descuadres detectados en un mes antes y después del sistema. Gracias a las validaciones incorporadas (ej. el sistema no permite finalizar un cuadre si hay datos faltantes, o alerta de facturas duplicadas), se eliminaron ciertos errores humanos frecuentes de la era manual. Estas métricas, aunque medidas de manera sencilla, proveen evidencia objetiva del impacto positivo de la automatización, reforzando las conclusiones del trabajo.
- Pruebas de seguridad y roles: Dado que el sistema maneja información financiera sensible, se llevaron a cabo pruebas de seguridad básicas. Por un lado, se probó la robustez del mecanismo de autenticación: intentos de SQL injection en los campos de login (validados mediante consultas preparadas en PHP para prevenir

inyección), intentos de session hijacking simulados, etc. Por otro lado, se validó estrictamente el esquema de autorización por roles, intentando acceder a URL internas del sistema sin credenciales o con un usuario de rol inadecuado, confirmando que el sistema redirige o bloquea dichos accesos (por ejemplo, un cajero no puede forzar la entrada al módulo /admin/reportes, ya que el backend verifica su rol antes de servir la página). Estas pruebas de control de acceso coincidieron con casos de prueba enumerados en el diseño (ver Fase 3, Sprint 1) y todas fueron satisfactorias. De esta forma, nos aseguramos de que cada rol de usuario solo puede efectuar las acciones permitidas, fortaleciendo la seguridad global del sistema.

En conjunto, los procedimientos anteriores garantizan que el aplicativo web cumple con su propósito de forma correcta, eficiente y segura. Vale resaltar la importancia de la retroalimentación continua: al aplicar un enfoque ágil, la validación no se dejó solo para el final, sino que en cada sprint se obtenían comentarios y se hacían ajustes. Este ciclo de realimentación constante es una de las razones por las cuales la calidad del producto final tiende a ser más alta. Finalmente, antes de la entrega de la tesis, se efectuó un ciclo de pruebas finales en producción. Con todo ello, se puede afirmar que el sistema fue rigurosamente validado antes de su adopción operacional.

5.2. Tabla de casos de uso

Tabla 7

descripción de cada funcionalidad

ID Caso	**Descripción**	**Resultado Esperado**	**Resultado Obtenido**	**estado**
CP - 001	Iniciar sesión como administrador	Accede al panel principal si las credenciales son correctas	Se accede correctamente	aprobado
CP - 002	Iniciar sesión con datos incorrectos	Se muestra mensaje de error	"Credenciales incorrectas"	aprobado
CP - 003	Recuperar contraseña respondiendo pregunta	Permite definir nueva clave si respuestas son correctas	se cambia las contraseñas	aprobado
CP - 004	Registrar nueva venta	La venta se guarda correctamente en la base de datos	Venta registrada y se visualiza en el historial de ventas	aprobado
CP - 005	Registrar egreso con imagen de factura	Se guarda el egreso y se muestra miniatura de la imagen	Imagen cargada y egreso almacenado correctamente	aprobado
CP - 006	Registrar egreso sin imagen	El egreso se guarda igual, sin requerir imagen obligatoria	Se registra el egreso sin imagen	aprobado
CP - 007	Agregar nuevo producto	El producto se muestra como opción activa en la venta	Producto visible en la pantalla de nueva venta	aprobado
CP - 008	Editar un producto	El nombre y precio se actualizan en el sistema	Cambios visibles al volver a cargar pantalla de venta	aprobado
CP - 009	Desactivar producto	El producto ya no debe mostrarse en el módulo de ventas	Producto oculto en lista de venta	aprobado
CP - 010	Agregar insumos al inventario	El stock se incrementa correctamente	Stock actualizado automáticamente	aprobado
CP - 011	Registrar receta para un producto	Se guarda combinación de los insumos como receta	Receta mostrada en vista de detalle del producto	aprobado
CP - 012	Validar stock antes de vender	Solo permite cantidad máxima disponible según receta e insumos	Mensaje de alerta que no se puede vender por stock faltante	aprobado

CP - 013	Realizar cuadre de caja	Se genera total ingresos - egresos y muestra saldo	Se muestra resultado del cuadre con opción de exportar	aprobado
CP - 014	Exportar reporte de ventas a Excel	Descarga de archivo en formato .xlsx	Archivo generado con datos correctos	aprobado

6. CAPITULO VI

6.1.CONCLUSIONES.

El desarrollo de este aplicativo web para Coffee Burguer dio un complimiento al objetivo general plateado, ya que se logra implementar una herramienta el punto de venta con el control de inventario, con la automatización de caja y optimizando la gestión de insumos. Mejorando la eficiencia operativa y rentabilidad del negocio, eliminando en su totalidad el registro manual.

La revisión de los procesos actuales de los procedimientos de ventas, egresos e inventario permitió levantar requerimientos claros que sirvieron de base para el diseño del sistema. Esto garantizó que la solución respondiera de manera precisa a las necesidades reales del negocio.

El diseño de la arquitectura basada en el patrón MVC y base de datos relacionales en MySQL, acompañadas con un lenguaje de programación PHP y dando una interfaz más amigable con boostrap. Se obtuvo un sistema muy ordenado, fácil de manejar y con posibilidades de crecer en un futuro sin mayores complicaciones.

Al implementar un módulo de ventas funcional lo cual permite registrar pedidos, y a su vez se suma automáticamente todos los montos ingresos y egresos, se puede hacer un cuadre de caja en un tiempo muy reducido, logrando eliminar dependencias de registro y operaciones matemáticas manuales para evitar errores de cálculos.

Al integrar un inventario se pudo tener una organización de los insumos basado en las recetas de cada producto, permitiendo registrar entradas de stock, definir insumos por cada producto y que este se descuente de una manera automática en el inventario después de una venta.

Aporta al control del negocio para no quedarse sin insumos y permitiendo planificar las compras de acuerdo al consumo real.

Con las pruebas realizadas se comprobó que el sistema funciona correctamente y que realmente mejora el trabajo en Coffee Burguer. El cuadre de caja pasó de tardar 2 o 3 horas a solo unos 15 minutos, se redujeron los errores en los cálculos y ahora hay un mejor control de ingresos y egresos. Estos resultados demuestran que la solución fue la adecuada para el problema que se buscaba resolver.

6.2.RECOMENDACIONES.

Como recomendaciones se puede decir que hay que mejorar la gestión de egresos, como, haciendo que la imagen de la factura sea obligatoria siempre y cuando sea el egreso mayor a 5 dólares como ejemplo, con esto estamos mejorando un control financiero de una forma correcta.

Actualmente, el sistema funciona en un entorno local (localhost), por esto se recomienda la implementación del sistema en un servidor en la nube (por ejemplo, AWS, Google Cloud, Azure). De esta manera, la aplicación estaría disponible desde cualquier dispositivo con conexión a internet, incluyendo tablets y teléfonos móviles, sin depender de un entorno local de desarrollo.

Alerta de stock mínimo, Actualmente el módulo de inventario solo permite visualizar el estado de los insumos y muestra si un producto se encuentra por debajo del stock mínimo configurado. Sin embargo, no emite alertas automáticas que avisen oportunamente cuando un insumo está próximo a agotarse. Se recomienda implementar un sistema de notificaciones (por ejemplo, mensajes emergentes en la interfaz) que

informe al administrador y al cajero sobre el próximo quiebre de stock, así aseguramos que los dos actores estén al tanto de cada insumo.

Respaldar periódicamente la base de datos, se recomienda establecer un plan de respaldo periódico de la base de datos, con el objetivo de garantizar la continuidad operativa del sistema ante posibles fallas técnicas, caídas del servidor o pérdida de información. Dicho plan debe incluir la programación de copias automáticas diarias, almacenamiento seguro en dispositivos externos o en la nube.

Mejoramiento de roles de usuarios, la aplicación Actualmente cuenta únicamente con dos roles predefinidos (administrador y cajero), sin posibilidad de crear, editar o eliminar nuevos perfiles, lo cual se podría crear un módulo de gestión de roles que solo pueda hacer el administrador, y pueda crear nuevos perfiles con privilegios según la necesidad, puede ser para cajero de la mañana y cajero para la tarde.

Se recomienda desarrollar un módulo de facturación electrónica, en conformidad con la normativa tributaria vigente, que permita generar comprobantes válidos para el Servicio de Rentas Internas (SRI) y enviarlos automáticamente al correo electrónico del cliente. Asimismo, se sugiere integrar una pasarela de pagos que habilite el cobro con tarjetas de débito y crédito, complementando las formas de pago ya existentes (efectivo y transferencia).

REFERENCIAS

- Ángel Gutiérrez , José Luis López · Desarrollo y programación en entornos web 2017.
- AECA (2019). Gestión de tesorería en PYMES. Asociación Española de Contabilidad.
- Balarezo. (2021). Business, I. (2021). ¿Qué son y cuáles son más utilizadas? IEBS Blog. Metodologías ágiles.
- Bustamante. (2024). Cómo hacer arqueo de caja diario en restaurantes. Qamarero Blog. Obtenido de https://qamarero.com Capuñay. (2021).
- Carambia. (2021). Carambia, M. L., Gómez, G. M., & Impacto de la automatización en la industria contable: un análisis exploratorio. Carambia, M. L., Gómez, G. M., & Balbiani, J. L. (2019). Impacto de la automatización en la industria contable: un Revista Latinoamericana de Ciencias Sociales, Niñez y Juventud. doi:Carambia, M. L., Gómez, G. M., & Balbiani, J. L. (2019). Impacto de la automatización en la industria contable: un análisis exploratorio. Revistdoi.org/10.11600/rlcsnj.17.
- Corporation, O. (2006). “Understanding the Three-Tier Architecture. Oracle TopLink 10g Release 3 Developer Guide . Cortés, M. y. (2023).
- Docker. (2024). Overview of Docker containers. Docker docs. Obtenido de https://docs.docker.com ESEID. (2023).
- Estrada, C. (2019). La adopción de las TIC en restaurantes. Innovar. doi:Cruz Estrada, I., & Miranda Zavala, A. M. (2019). La adopción de las TIC en restaurantes de Pudoi.org/10.15446/innovar.v29n72.77932.

- Juan Martínez "Del Sueño a la Realidad: Curso Completo de Emprendimiento y Creación de Empresas" Editorial :Juan Martínez 202 Idioma Español
- Jiménez, J. I., Rojas, F. S., & Ospina, H. J. (2013). La importancia del ciclo de caja y cálculo del capital de trabajo en la gerencia PYME. Clio America, 13.
- Ferrer-Dávalos. (s.f.). . Adopción e impacto de las TIC en la gestión de microempresas. Revista Científica en Ciencias Sociales, 23(1). doi:10.53732
- Figueroa Cruz, E. E. (2021). Aplicativo web para el proceso de ventas de la empresa Fagum E.I.R.L . Repositorio Institucional Continen.
- Foundation, P. S. (2024). Creation of virtual environments. venv.
- García, M. & López, R. (2021). Digitalización financiera en hostelería. Universidad de Sevilla.
- GitHub. (2023). GitHub Actions documentation. Innovar. Obtenido de https://docs.github.com/en/actions
- Grinberg. (2023). Flask Web Development. O'Reilly Media.
- Guerrero. (2022). Digitalización empresarial para las PYMEs del sector gastronómico. PYME.
- Hat, R. (2022). Introducción a la metodología ágil: principios y beneficios. Red Hat DevOps Resources.
- Hat, R. (2022). introducción a la metodología ágil: principios y beneficios. Red Hat DevOps Resources.
- Jiménez. (2024). e Benefits of using the MVC pattern in agile environments. International Journal of Computer Scienc, 22(3).
- Krajewski, L., Ritzman, L., & Malhotra, M. (2019). Administración de operaciones: Procesos y cadenas de valor (12ª ed.). Pearson.

- Pasquel. (2024). La importancia del cuadre de caja diario en la salud financiera de tu negocio. Prikly, 1(2). Peréz. (2022).
- Pyton. (2024). Creation of virtual environments. Prikly Blog. Obtenido de venv — Creation of virtual environments. https://docs.python.org
- Ramos, Y., & Castro, A. O. (2017). Point-Of-Sales Systems in Food and Beverage Industry: Efficient Technology and Its User Acceptance. Journal of Information Sciences and Computing Technologies, 6(1), 582–591. Obtenido de: https://scitecresearch.com/journals/index.php/jisct/article/view/1024.
- Raghu. (2025). Optimizing microservices architecture for high-speed financial data processing. Electronic Journal. doi:doi.org/10.2139/ssrn.5238748 Ramiro. (2022).
- Rootstack. (2024). Seguridad en finanzas digitales: mejores prácticas. Rootstack Blog, 1(3).
- Rossum. (2022). SAP. Style Guide for Python Code., 234. Obtenido de https://peps.python.org/pep-0008
- Rossum, V. (2022). Style Guide for Python Code. PEP 8, 253. Obtenido de https://peps.python.org/pep-0008
- Salim, K. &. (2022). international Journal of Systems Research. architecture.
- Sánchez. (2021). características y usos. SQLite:. Obtenido de https://josejuansanchez.org
- Santos, G. (2919). Sistema web para la automatización del servicio en restaurantes. Tecnológico Nacional de México, 1(2).
- SAP. (2024). Ventajas de la digitalización de la contabilidad financiera en las empresas. Concur Argentina Blog. Torres. (2020).

- Softeng. (20 de Noviembre-2014) softeng.es. [Online]. http://www.softeng.es/es-es/empresa/metodologias-de-trabajo/metodologiascrum.html
- SCRUM. (2022, September 10). Gestión de proyectos ágil vs. tradicional: un análisis comparativo... https://www.scrumstudy.com/article/agile-vs-
- Touza. (2020). Producto mínimo viable: qué es y cómo hacerlo. ESERP Business & Tech Blog.
- Valley, T. (2024).
- Vargas. (2024). Resilient microservices architecture with embedded AI observability for financial systems. Proceedings of the IEEE International Conference on Cloud Computing , 124.
- Villamil, J. (2021). Diseño e Implementación de un Sistema POS, con Módulo de Gestión de Inventario de Productos para Clientes y Perfiles de Usuario, Aplicando Metodología RUP. https://repository.unad.edu.co/jspui/bitstream/10596/41806/1/jhvillamilr.pdf.
- W3C. (2023). Web Content Accessibility Guideline. Obtenido de https://www.w3.org/TR/WCAG22/

7. ANEXOS

INSTITUTO TECNOLÓGICO SUPERIOR “RUMIÑAHUI”

CARRERA DE SISTEMAS Y GESTION DE DATA

“MANUAL DE USUARIO DEL APLICATIVO WEB QUE AUTOMATICE EL CUADRE DE CAJA PARA EL EMPRENDIMIENTO COFFEE BURGUER”

AUTORES

JHON FERNANDO GUALAVISI CAIZA

DOCENTE:

ING. PAEZ OSCULLO DANNY.

SANGOLQUI, SEPTIEMBRE 2025

Contenido

Índice de ilustraciones

1. Introducción del manual

El presente manual está dirigido a los usuarios del sistema web Coffee Burguer. Este sistema permite gestionar ventas, productos, insumos, egresos y cuadre de caja de manera eficiente y centralizada. A través de este documento, los usuarios podrán comprender el funcionamiento de cada módulo y utilizarlo adecuadamente según su rol.

2.5. 2. Requisitos del sistema

- Servidor local: XAMPP (PHP 8.1+, MySQL)
- Navegador compatible: Google Chrome, Firefox, Edge
- Sistema operativo sugerido: Windows 10/11
- Resolución mínima: 1366x768

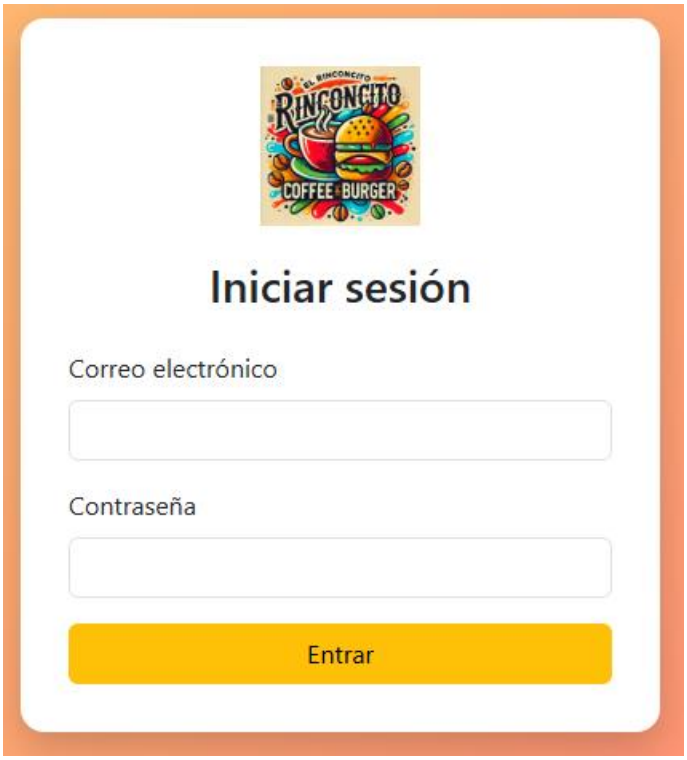
2.6. 3. Acceso al sistema

El sistema se accede desde el navegador mediante http://localhost/CoffeeBurguer/views/login.php

Lo cual solicita el correo y contraseña del usuario.

Ilustración 9

Pantalla principal del aplicativo web para el login.



2.7. 4. Roles de usuario

Administrador: es quien tiene el acceso total a todo el sistema sin depender de nadie y puede realizar acciones únicas en comparación al cajero.

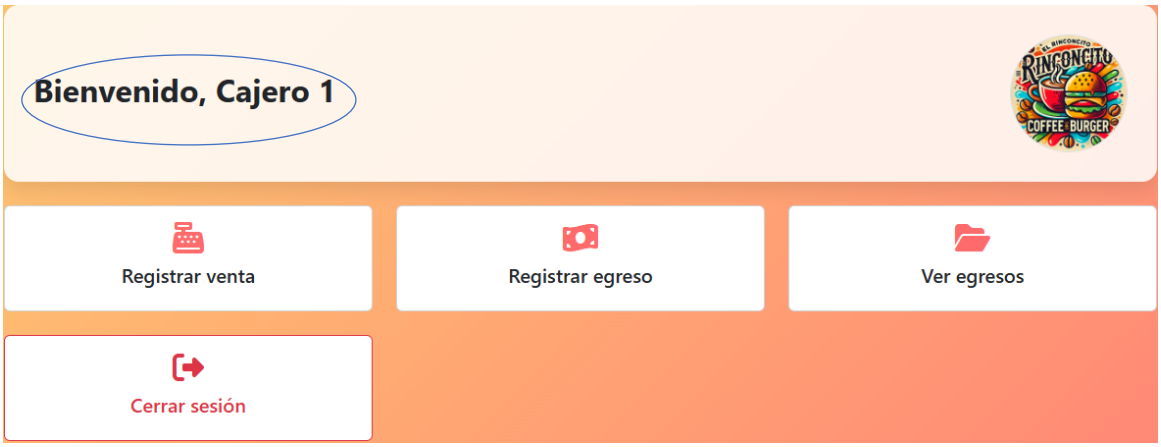
Ilustración 10

Pantalla de bienvenida al administrador

Cajero: es quien tiene su acceso limitado a funciones necesarias a su rol como registra ventas, añadir egresos, y ver egresos.

Ilustración 11

Pantalla de bienvenida a cajero 1

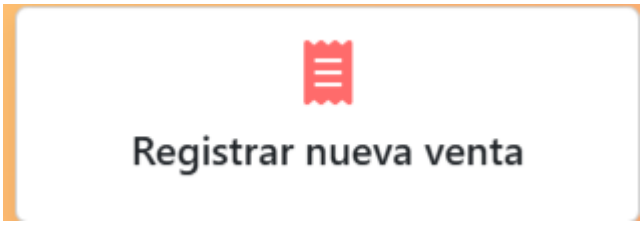


2.8. 5.1 Ventas

Registra venta es donde podemos acceder a la pantalla de nueva venta

Ilustración 12

Opción para registra nueva venta

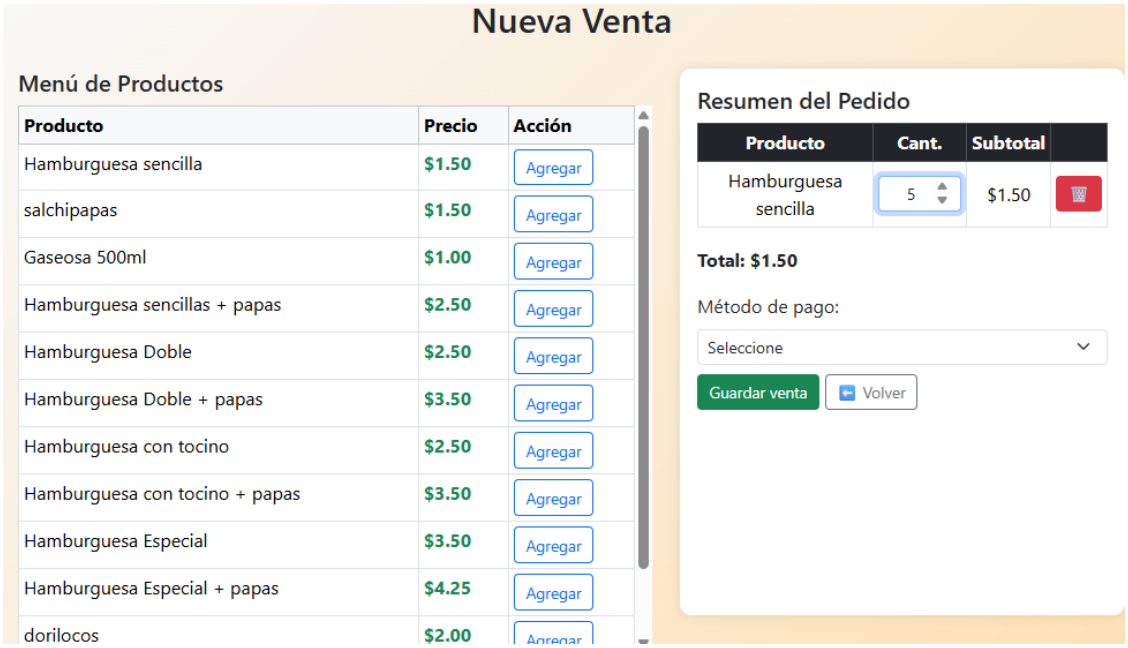


Una vez dentro de la pantalla nueva venta es donde tanto como **administrador** y **cajero** pueden registrar una venta nueva. Este proceso tiene que realizar de la siguiente manera:

- Se debe escoger el producto para vender dando clic en **agregar.**
- Este producto escogido pasara a la lista de resumen de pedido, ahí es donde se pude escoger la cantidad del producto o a su vez eliminar dando clic en icono rojo.
- Este proceso es para todos los productos que desee el cliente.
- Finalmente se observará la suma total del pedido.
- Lo cual se deberá escoger el método de pago del pedido, ya sea efectivo o transferencia.
- Por último, se debe dar clic en **Guardar Venta**. Después de eso, el sistema regresará automáticamente a la pantalla de inicio con la venta ya registrada.

Ilustración 13

Pantalla completa para realizar una venta



Nota: en este módulo el sistema evalúa primero la receta del producto y verifica que cada insumo esté disponible en stock para poder vender, caso contrario el sistema también indicara que no se puede vender por falta de insumos.

2.9. 5.2 Productos

Desde este módulo que tiene acceso solo **cajero** se pueden registrar nuevos productos, editarlos o desactivarlos. Solo los productos activos aparecen en el menú de venta.

Ilustración 14

Opción de la pantalla principal para gestionar un producto

En este módulo de productos disponibles muestra el precio actual y el estado se puede editar, lo cual se puede cambiar el nombre o el precio del producto y guardarlos, o cancelar si no se desea realizar ningún cambio.

Ilustración 15

Opciones de cada producto como añadir, editar o desactivar productos.

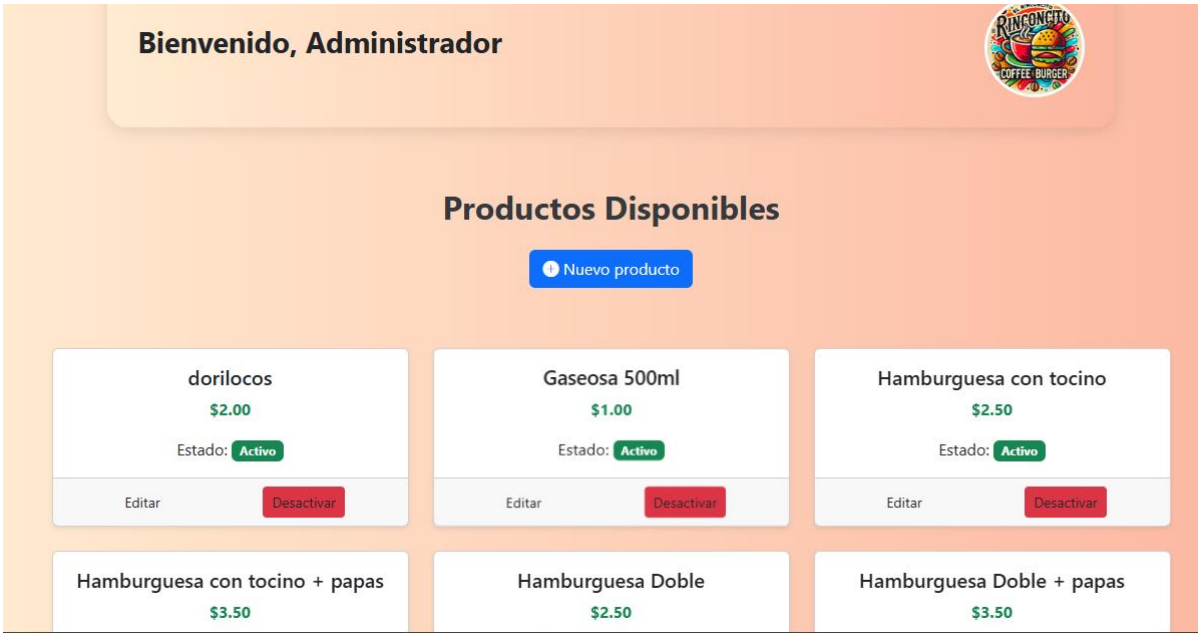


Ilustración 16

Editar producto

También se tiene la opción de desactivar el producto, lo cual nos permite activarlo o desactivarlo para que no se muestre en la pantalla de pedido, esto es cuando talvez existen productos que ya no se venda, mostrando el estado de cada producto.

Ilustración 17

Activar o desactivar algún producto

2.10. 5.3 Recetas

Permite definir al **cajero** la receta de cada producto: insumos, cantidades y unidades. Esto controla automáticamente el stock cuando se vende un producto.

Ilustración 18

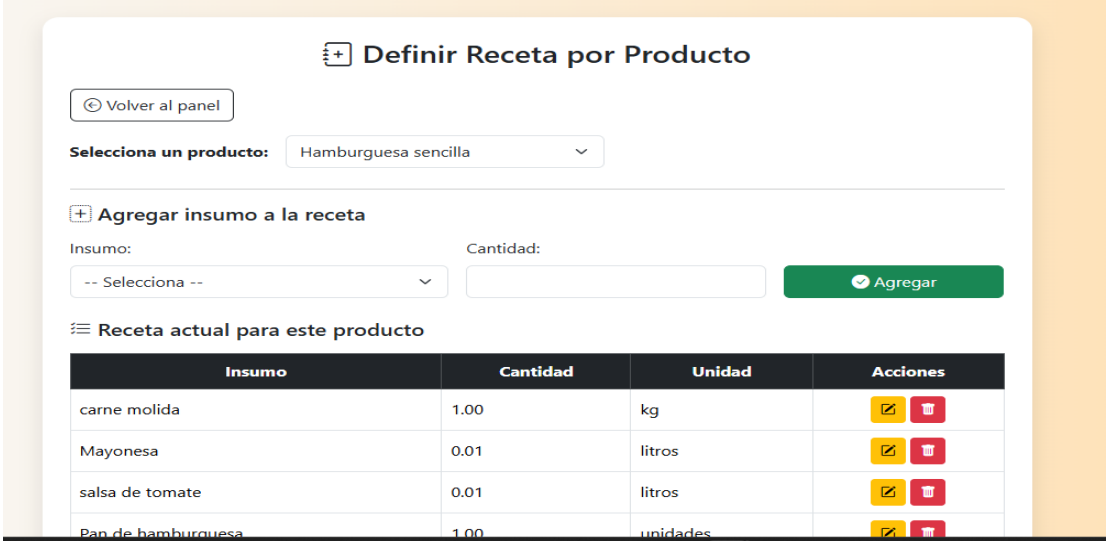
Opción en la pantalla principal para definir receta de algún producto

Se debe seleccionar un producto en específico como se puede ver en la imagen se seleccionó el producto hamburgués sencilla, al seleccionar el producto me indica directamente la receta actual del producto.

Nota: para definir alguna receta primero se debe estar agradado en el módulo de gestionar productos.

Ilustración 19

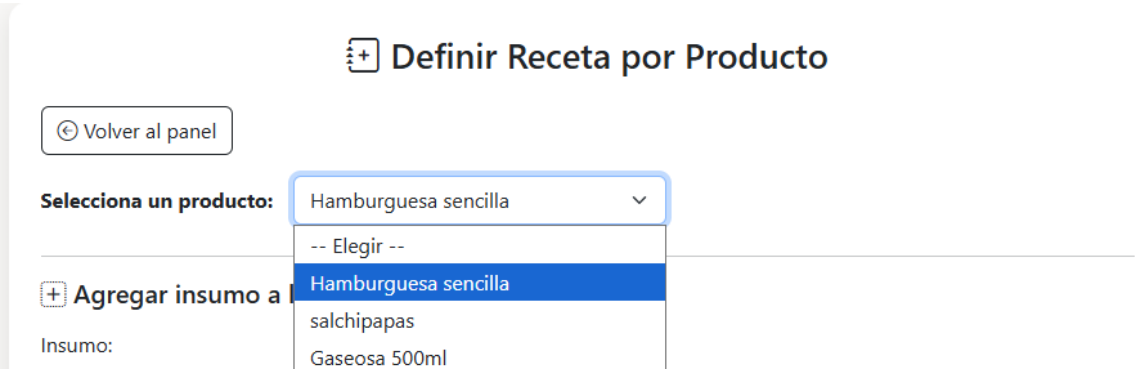
Pantalla principal para definir receta.



Aquí se selecciona el producto al que se va a definir una receta con diferentes insumos.

Ilustración 20

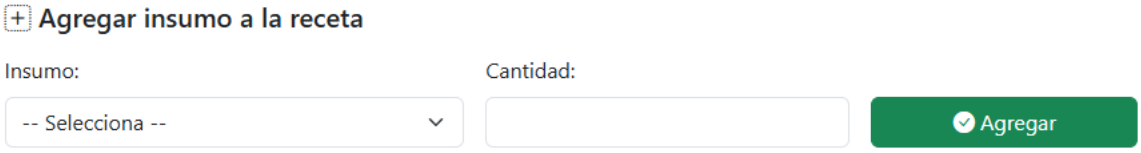
Selección de producto.



Se puede ingresar nuevos insumos a la receta seleccionando y la cantidad que se ocupa para la elaboración del producto y finalmente se debe agregar para que guarde el insumo en la receta.

Ilustración 21

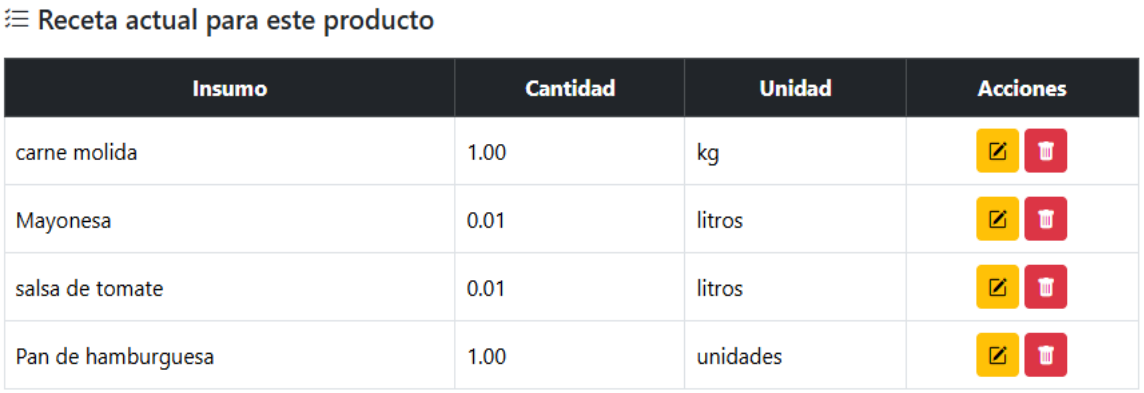
Selección de insumos para la receta.



Una vez seleccionado el producto aquí me indica la receta del producto que de igual manera se puede editar o eliminar el insumo o su vez eliminar el insumo de la receta actual.

Ilustración 22

Información de la receta con acciones de editar o eliminar insumo.

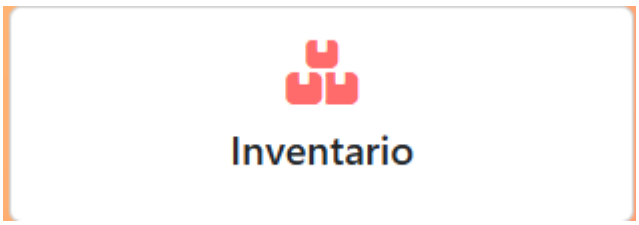


2.11. 5.4 Insumos

Aquí se puede ver solo el administrador el inventario actual, agregar nuevos insumos y modificar sus valores mínimos de stock.

Ilustración 23

Opción en la pantalla principal para ver el inventario



Se encuentra la información de todos los insumos del inventario como el nombre, unidad, stock actual, stock mínimo estado y acciones.

Ilustración 24

Vista principal del inventarió

Al escoger nuevo insumo es donde ingresamos al inventario un insumo nuevo al stock actual en el cual se define el nombre unidad de medida stock y stock mínimo del nuevo insumo, después de hacer este proceso se debe dar clic en guardar y se guardara automáticamente en nuestro sistema de inventario.

El stock mínimo nos define la alerta de la cantidad que se escoja para que aparezca el mensaje de estado ok y estado bajo.

Ilustración 25

Botón para agregar nuevo insumo

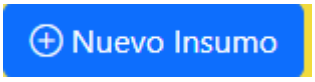


Ilustración 26

Campos requeridos para ingresar un nuevo insumo.

Al ingresar a registra compra/entrada es donde escogemos el insumo previamente declarado en el inventario de insumos es aquí donde ya no se declara nuevo insumo sino se registra la cantidad comprada del insumo en stock.

Ilustración 27

Botón de registro de compra.

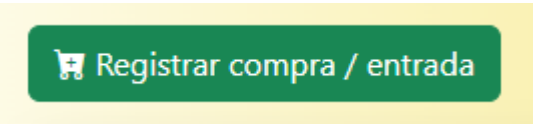
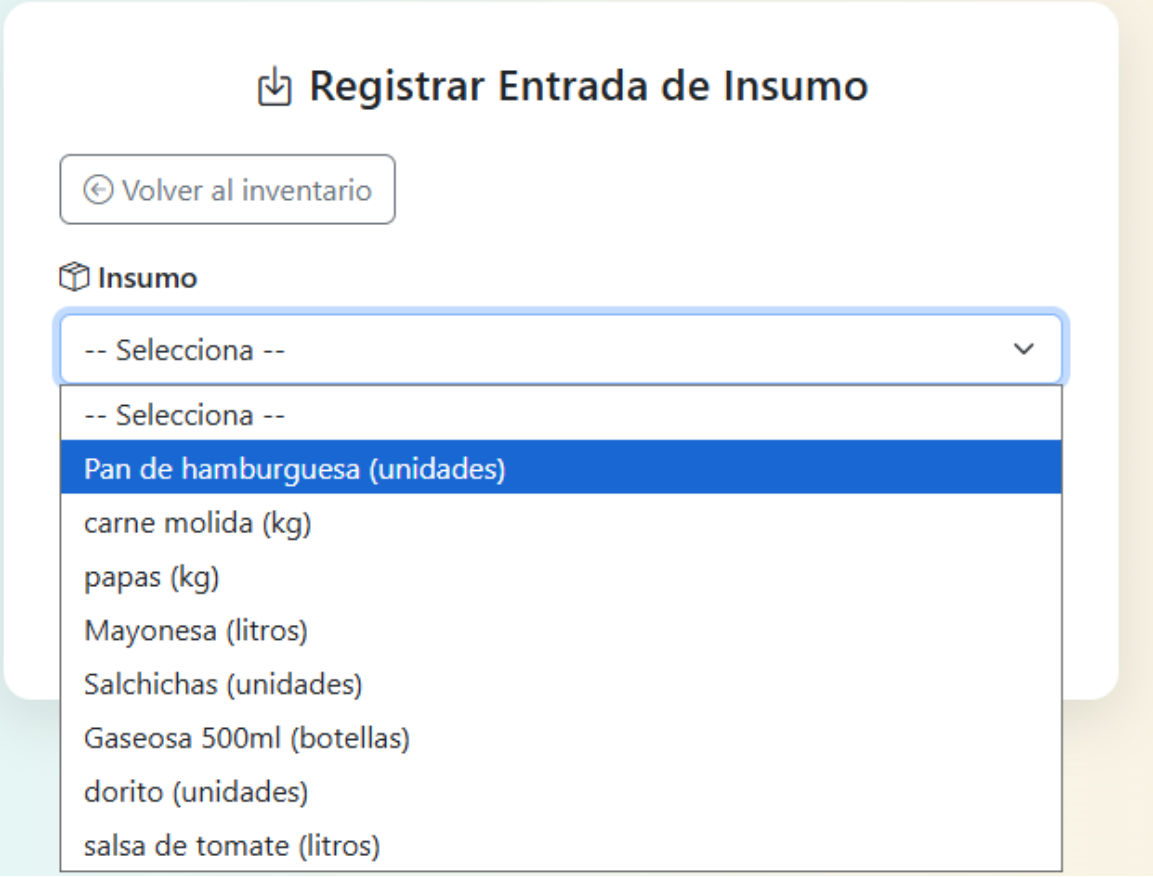


Ilustración 28

Selección que insumo se compro

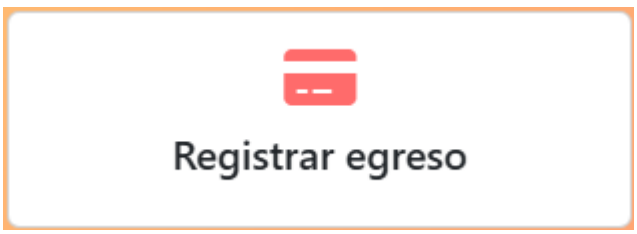


2.12. 5.5 Egresos

Se registran los gastos operativos el administrador como el cajero. El sistema permite adjuntar imágenes de facturas opcionalmente.

Ilustración 29

Opción en la pantalla principal para registrar egresos

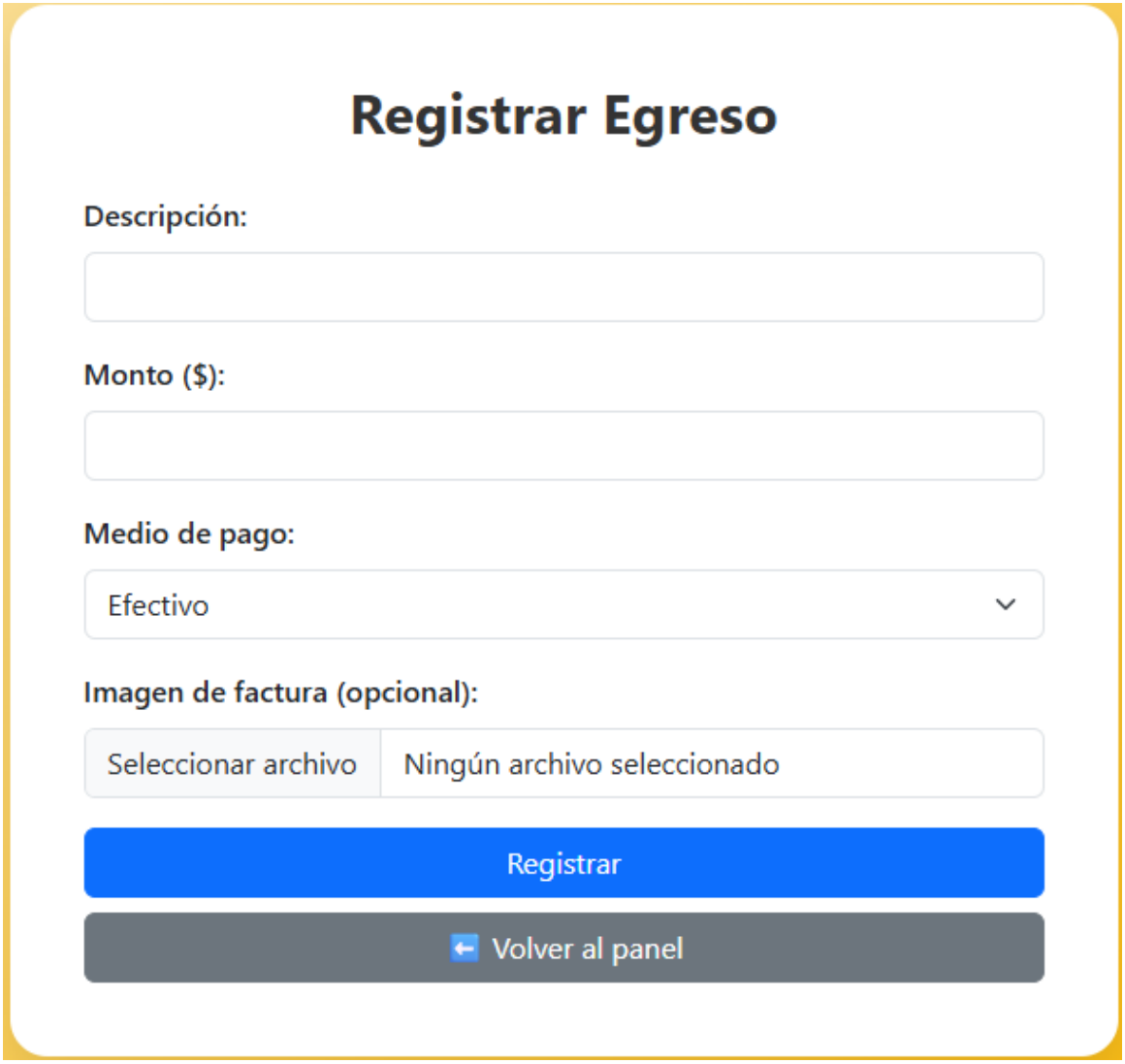


Se debe llenar los campos obligatoriamente como es la descripción del insumo o producto que se compró el monto que se pagó con la definición de medio de pago ya sea

transferencia o efectivo, y se subirá la foto de la factura esta última será una imagen opcional, dado todos estos pasos procederá a registra.

Ilustración 30

Campos necesarios de cada ingreso.

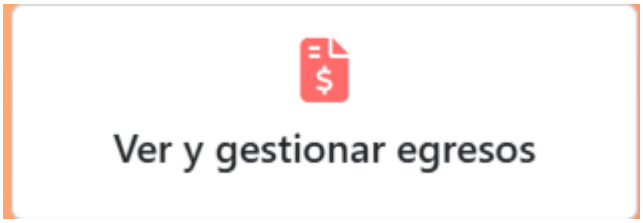


2.13. Listado de Egresos

Permite únicamente al administrador filtrar por fecha, visualizar todos los egresos realizados por el cajero o administrador, y ver las facturas adjuntas.

Ilustración 31

Opción en la pantalla principal para editar egresos.



Muestra una información detallada de cómo es el proceso de los egresos indicando desde la fecha y hora que se registró el egreso, como la descripción, el monto que se pagó, la factura en caso de haber subido una imagen se mostrara en tamaño pequeño los cual al dar clic se podrá ver en tamaño grande, registrado por los cual indica quien registro el egreso ya sea administrador o cajero.

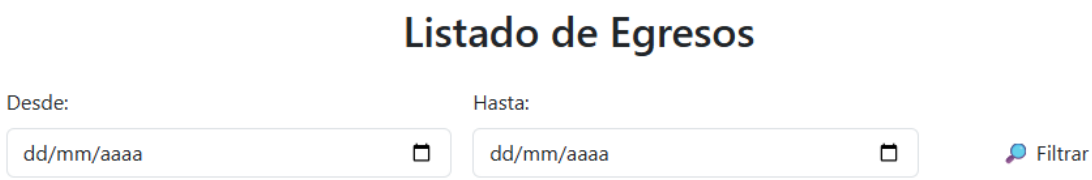
Ilustración 32

Información de los egresos actuales.

En este campo se puede filtrar la lista de egresos de fechas especificas colocando la fecha de inicio y la fecha hasta al pulsar el botón filtrar.

Ilustración 33

Filtrado de fechas de egresos



Siguiendo con las acciones la cual este permite al administrador a editar ya sea la descripción, monto, medio de pago o subir la imagen en caso de no haberla subido antes.

Ilustración 34

Botones para editar o eliminar un egreso.

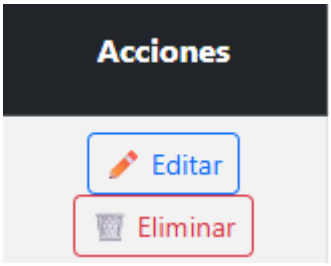


Ilustración 35

Campos de editar un egreso.

2.14. 5.6 Cuadre de caja

Solo el administrador puede ver el resumen diario de ingresos y egresos. El saldo final se muestra y se puede exportar.

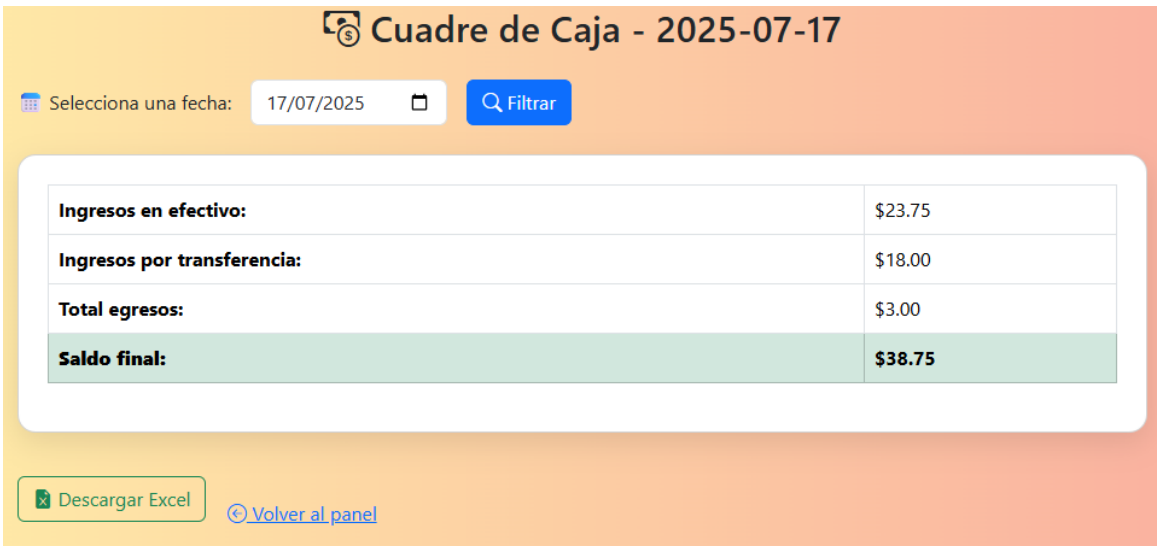
Ilustración 36

Opción en la pantalla principal para cuadrar caja.

Se selecciona al final del día cuando ya el local va a cerrar y al clic en filtrar indica los ingresos en efectivo, ingresos por transferencia, los egresos realizados. Dado estos parámetros nos indicara el saldo final que se debe tener.

Ilustración 37

Selecciona el día requerido.



Al seleccionar la opción de descargar Excel nos descargara en cuadre de caja de día en un archivo .xls, lo cual nos ayuda como información.

Ilustración 38

Botón para descarga de archivo

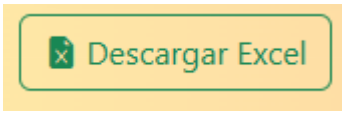
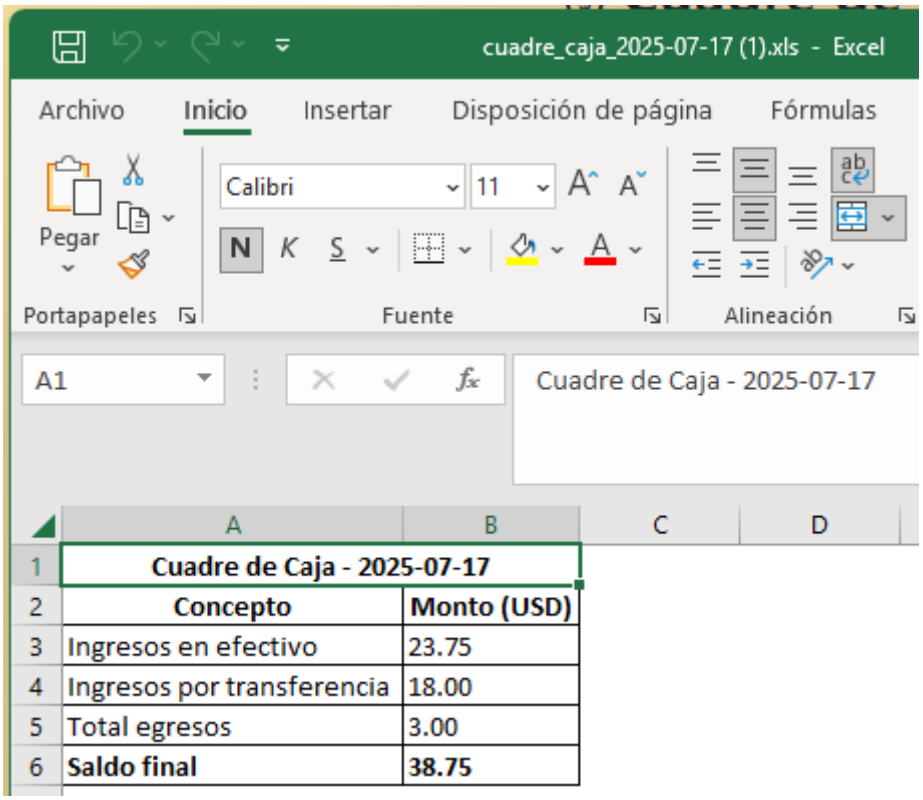


Ilustración 39

Archivo descargado.



5.7 Reportes

Permite consultar los productos más vendidos en un rango de fechas.

Ilustración 40

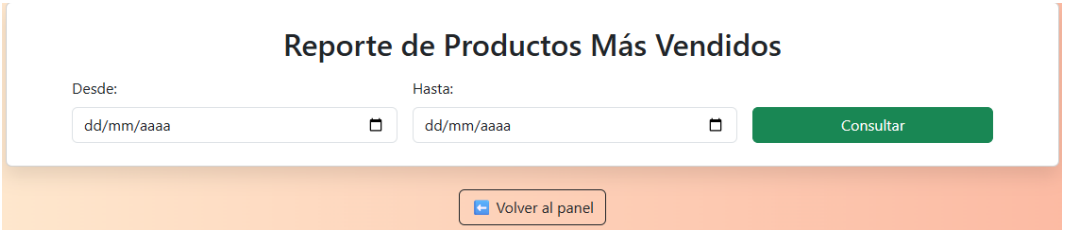
Opción en la pantalla principal de reporte en ventas.

Poner la fecha inicial para el reporte seguido poner la fecha hasta donde se requiere el reporte.

Con estos dos datos se puede consultar.

Ilustración 41

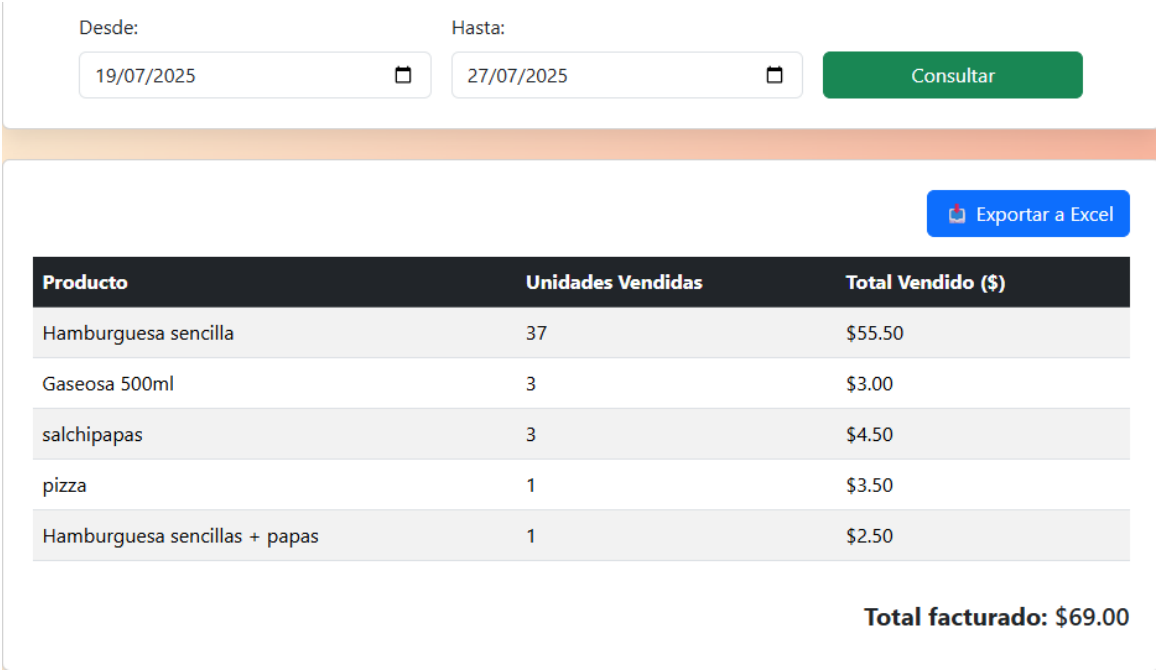
Filtrado por fechas de inicio y fin.



Una escogido las dos fechas requeridas indicara detalladamente el producto con las unidades que se a vendido en las fechas seleccionadas y el total de dinero que se ha vendido y con el total facturado en las fechas ya mencionadas.

Ilustración 42

Descripción del filtrado.



Dando la opción de exportar a Excel. recibiendo los mismos datos que se observó en la pantalla anterior.

Ilustración 43

Botón para exportar.

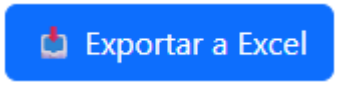
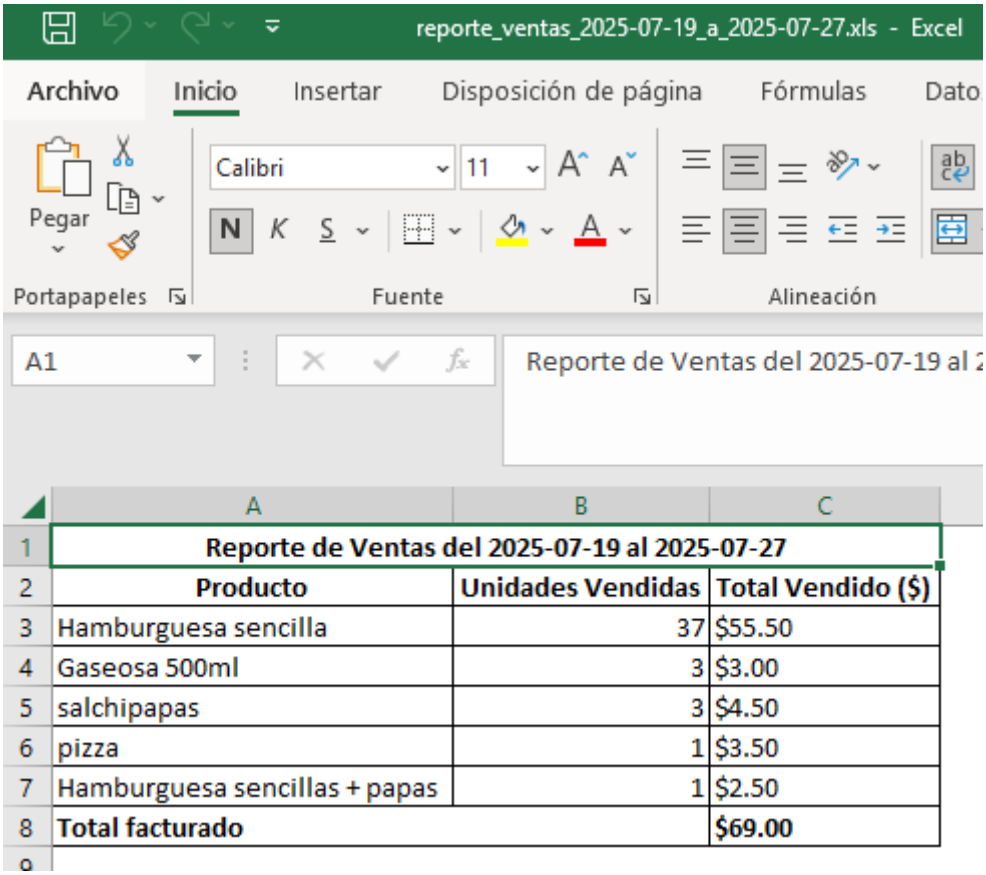


Ilustración 44

Archivo exportado



	A	B	C
1	**Reporte de Ventas del 2025-07-19 al 2025-07-27**		
2	**Producto**	**Unidades Vendidas**	**Total Vendido ($)**
3	Hamburguesa sencilla	37	$55.50
4	Gaseosa 500ml	3	$3.00
5	salchipapas	3	$4.50
6	pizza	1	$3.50
7	Hamburguesa sencillas + papas	1	$2.50
8	**Total facturado**		**$69.00**

2.15. 6. Recuperación de contraseña

Si se introduce mal la contraseña, el sistema permite recuperar el acceso respondiendo preguntas de seguridad.

En caso de no saber u olvidarse la contraseña existe un módulo de recuperación de contraseña. Este se obtiene después de una validación de usuario con preguntas especificas dependiendo el rol que deseamos recuperar.

Ilustración 45

Recuperación de contraseña.

2.16. 7. Recomendaciones y soporte

No compartir credenciales de acceso:

Cada usuario (administrador o cajero) debe contar con su propia cuenta personal. Compartir contraseñas puede comprometer la seguridad del sistema, alterar el flujo normal de trabajo y dificultar el seguimiento de acciones en el registro de operaciones.

Mantener actualizados los datos de seguridad:

Es importante que cada usuario recuerde y mantenga actualizadas sus preguntas y respuestas de recuperación de contraseña. Esto facilitará el proceso de restablecimiento en caso de olvido.

Realizar respaldos frecuentes de la base de datos:

Se recomienda generar copias de seguridad periódicas de la base de datos, especialmente después de días de alta actividad. Esto previene la pérdida de información relevante en caso de errores, apagones o daños en el sistema local.

Cerrar sesión correctamente:

Al finalizar el uso del sistema, se debe cerrar sesión utilizando el botón correspondiente. Esto es clave en entornos compartidos o públicos para evitar accesos no autorizados.

Utilizar navegadores actualizados:

Para asegurar la compatibilidad total y un rendimiento adecuado, se sugiere utilizar navegadores modernos como Google Chrome, Mozilla Firefox o Microsoft Edge en su versión más reciente.

Supervisión periódica por parte del administrador:

El administrador del sistema debe revisar regularmente los registros de ventas, egresos, insumos y cuadre de caja para garantizar la coherencia de los datos ingresados por los cajeros.

Capacitación básica al personal:

Se recomienda brindar una inducción inicial a los usuarios (especialmente cajeros) sobre el uso del sistema, para evitar errores comunes y mejorar la eficiencia operativa.

Soporte técnico y contacto:

Ante cualquier error, duda o incidencia en el funcionamiento del sistema, los usuarios pueden comunicarse con el desarrollador o personal encargado al siguiente correo electrónico:

jhon.gualavisi@ister.edu.ec

INSTITUTO TECNOLÓGICO SUPERIOR “RUMIÑAHUI”

CARRERA DE SISTEMAS Y GESTION DE DATA

“MANUAL TECNICO DEL APLICATIVO WEB QUE AUTOMATICE EL CUADRE DE CAJA PARA EL EMPRENDIMIENTO COFFEE BURGUER”

AUTORES

JHON FERNANDO GUALAVISI CAIZA

DOCENTE:

ING. PAEZ OSCULLO DANNY.

SANGOLQUI, SEPTIEMBRE 2025

Contenido

Índice de ilustraciones

Índice de tablas

Introducción del manual de programador

Para el siguiente manual está dirigido a desarrolladores y personal técnico que necesiten comprender como es la estructura, funcionamiento e instalación del sistema web Coffee Burguer. Este sistema fue desarrollado siguiendo el patrón de arquitectura MVC (Modelo-Vista-Controlador) en PHP, utilizando MySQL como motor de base de datos. Coffee Burguer automatiza el proceso de ventas, control de insumos, gestión de productos, egresos y cuadre de caja para un emprendimiento local.

Objetivos

Proporcionar a cualquier desarrollador una guía completa sobre el funcionamiento técnico del sistema **Coffee Burguer**. que incluye la descripción de cada módulo, la estructura del proyecto, detalles de la base de datos, instalación, mantenimiento, y recomendaciones para futuras modificaciones o mejoras.

Tecnologías utilizadas

Entorno de ejecución

- **XAMPP** (Apache 2.x + PHP 8.x + MySQL) como servidor local de desarrollo.
- **phpMyAdmin** para la gestión gráfica de la base de datos.

Backend

- **HTML5** y con **Bootstrap 5** (vía CDN) para una aplicación responsive.
- **JavaScript "vanilla"** para validaciones cliente (`is-valid`/`is-invalid`).
- **Chart.js** (vía CDN) para la generación dinámica de gráficos de barras en el reporte de ventas.

Base de datos

- **MySQL** con codificación **utf8mb4**:

- Claves foráneas para integridad referencial y bloqueo a nivel de fila.

Frontend

- **HTML5** con **Bootstrap 5** para una aplicación responsive.
- **Bootstrap Icons 1.10.5** (vía CDN) para iconografía uniforme.
- **JavaScript** para validaciones cliente (`is-valid`/`is-invalid`), timeouts de alertas y manejo de modales.

- **Chart.js** (vía CDN) para la generación dinámica de gráficos de barras en el reporte de ventas y cuadre de caja.

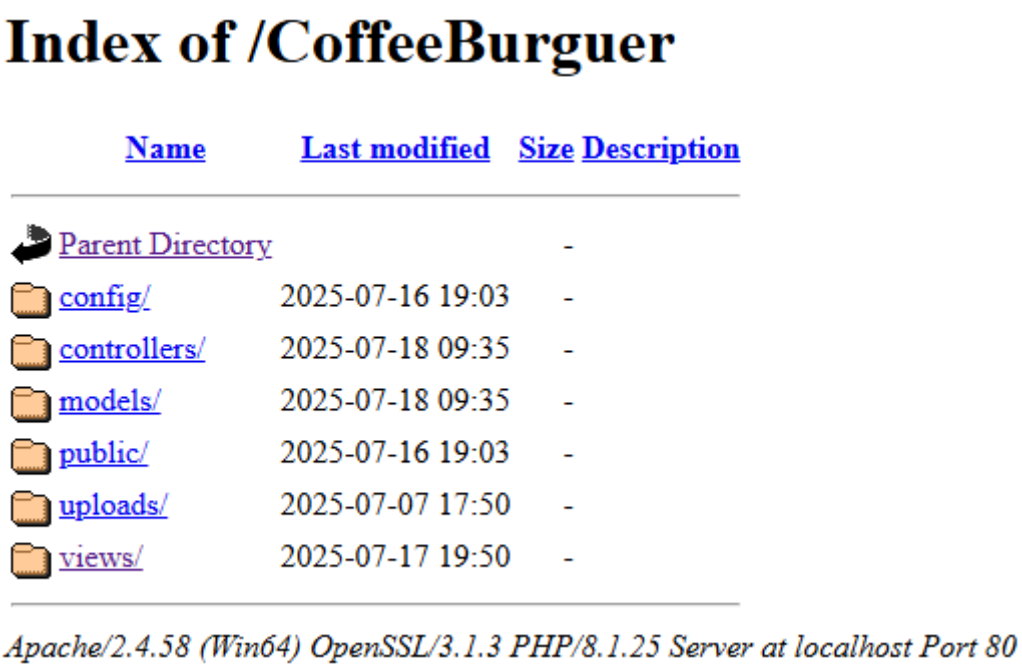
Imágenes y archivos

- Carpeta `public/facturas/` para almacenar las imágenes de facturas cargadas por el usuario, con validación de tipo MIME y tamaño máximo 2MB.
- Carpeta `public/` para assets estáticos como el logo.png

Otras herramientas

- **Visual Studio Code** ide principal para editar el código
- **Git** para control de versiones.

Ilustración 46 directorio principal del proyecto.



Requisitos técnicos

3. Hardware mínimo

- **CPU:** Procesador de al menos 2 GHz.
- **Memoria RAM:** 4 GB (8 GB recomendado).
- **Almacenamiento:** 10 GB libres en disco (para código, base de datos y archivos de facturas).
- **Red:** Conexión estable a internet.

4. Software

- Windows 10/11
- Servidor web (Apache 2.4.x)
- PHP 8.x

- **Base de datos (MySQL 8.x)**

Instalación de visual studio code

Principalmente se utilizará visual studio code ya que nos permite escribir código de una forma fácil ajustando al proyecto deseado, ya que puede funciona para muchos lenguajes de programación para la descarga se lo realiza en el siguiente link https://code.visualstudio.com/ . Se procede hacer la descarga y la instalación

Ilustración 47

descarga de visual studio code

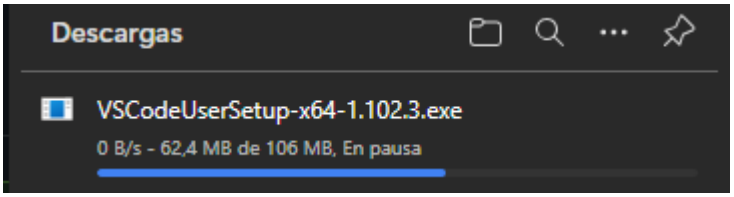


Ilustración 48 términos de licencia

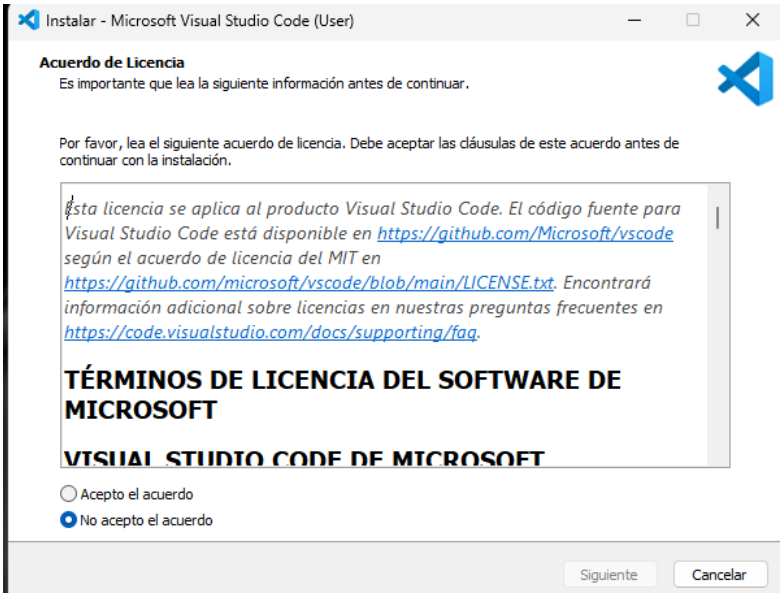


Ilustración 49 tareas adicionales

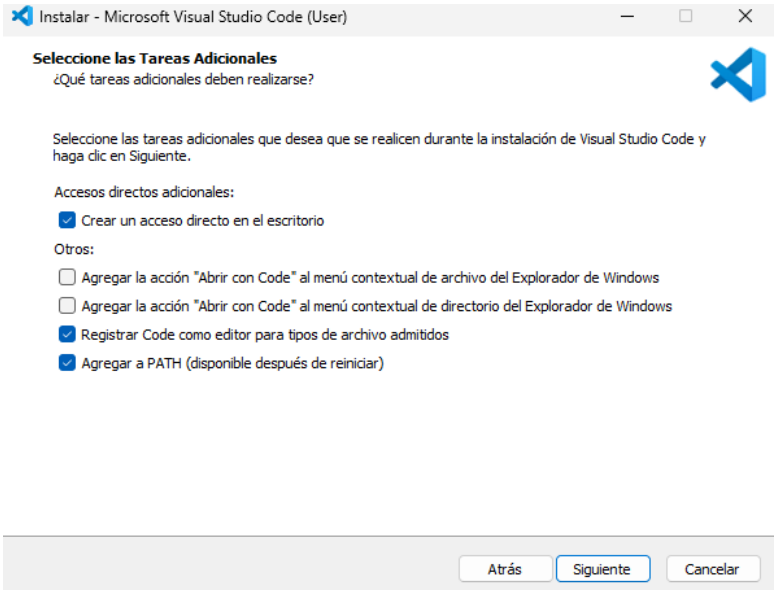


Ilustración 50 listo para iniciar la instalación

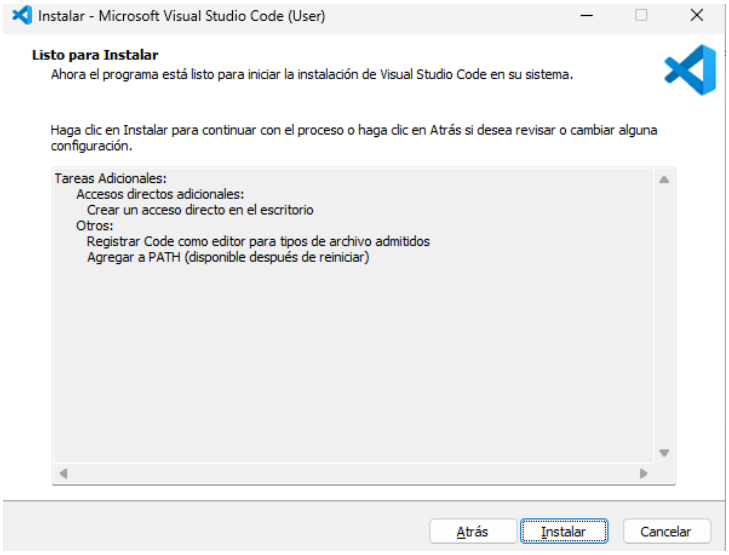


Ilustración 51 proceso de instalación

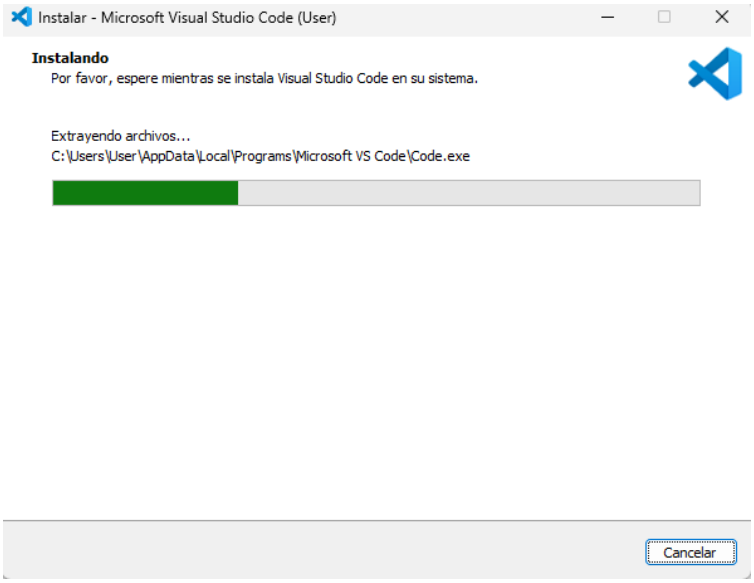
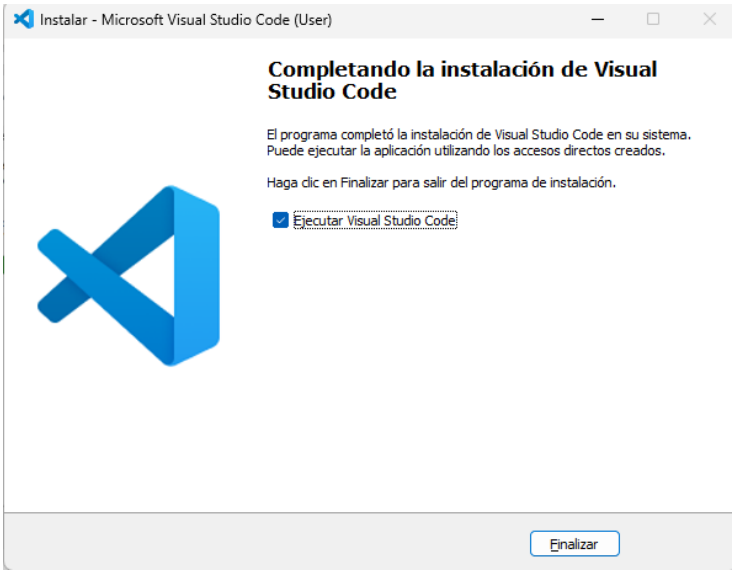


Ilustración 52 instalación completa

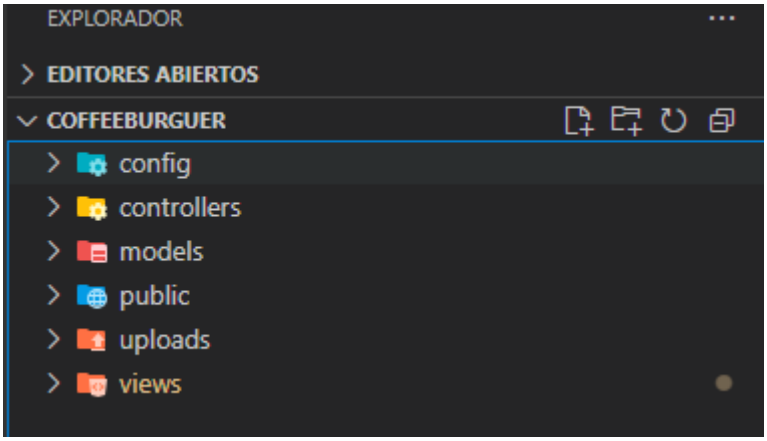


5. Creación del proyecto

Estructura del Proyecto

El proyecto Coffee Burguer está estructurado bajo el patrón de diseño Modelo-Vista-Controlador (MVC). A continuación, se describe la organización de carpetas y archivos principales. todas estas carpetas se encuentran dentro una carpeta llamada **CoffeeBurguer** como se muestra en la **ilustración 8**:

Ilustración 53 estructura del proyecto



controllers/: contiene los archivos encargados de la lógica de control, interacción con modelos y redirección de vistas.

models/: contiene las clases que gestionan la lógica de acceso a datos, especialmente conexión con la base de datos y consultas SQL.

views/: contiene las interfaces de usuario (archivos .php con HTML, formularios y tablas).

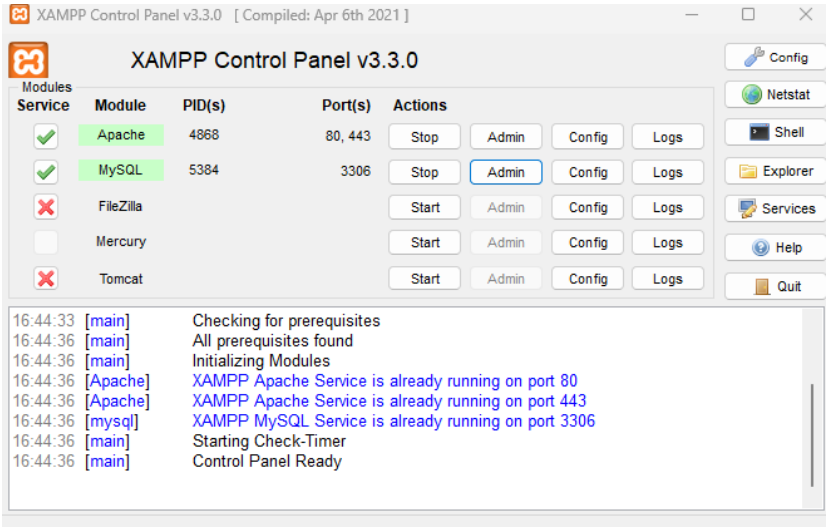
config/: contiene los archivos de configuración de conexión a la base de datos.

public/:contiene recursos estáticos como imágenes, hojas de estilo y scripts JS.

Creación de la base de datos

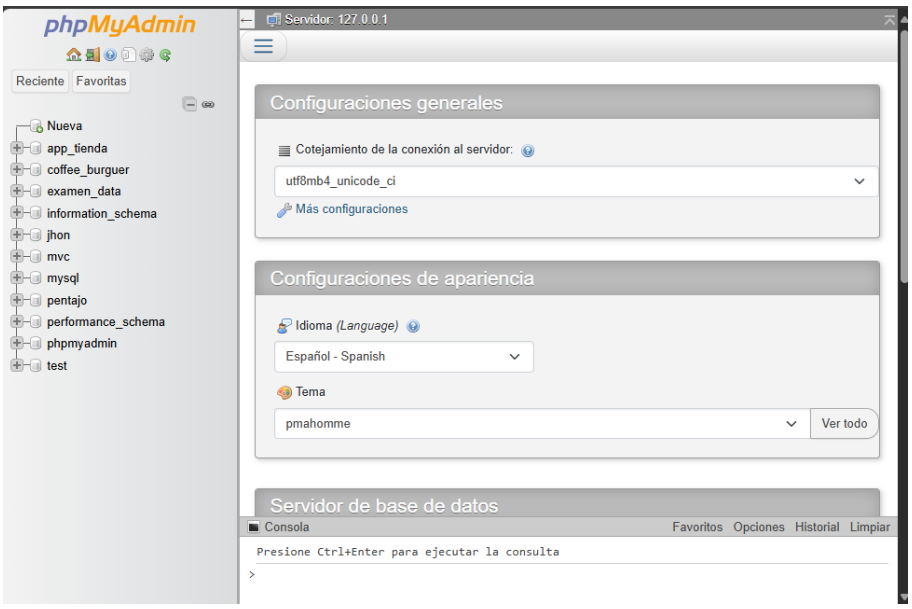
una vez abierto xampp se inicializa Apache y MySQL hasta ver los vistos verdes luego hacemos clic en admin de MySQL como se indica en la **ilustración 9**

Ilustración 54 pantalla principal de xampp



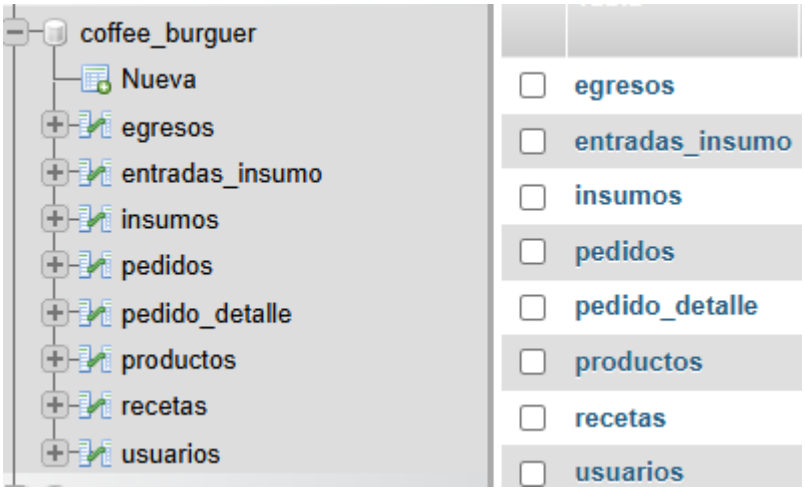
En pantalla principal de phpMyadmin se crea una nueva base de datos con el nombre de **coffee_burguer**

Ilustración 55 pantalla principal de phpMyAdmin



Una vez creada la base de datos se procede a crear las diferentes tablas que son necesarias para el proyecto como egresos, entrada insumos, insumos, pedidos, pedido detalle, productos, recetas, usuarios.

Ilustración 56 base de datos con sus tablas



A continuación, en la **tabla 1** se describe cada tabal con sus atributos y descripción breve.

Tabla de base de datos

Tabla 8 descripción de cada tabla con sus atributos

Tabla	Campo	Tipo	Restricciones	Descripción y justificación
usuarios	id	INT(11) PK AUTO_INCREMENT	PRIMARY KEY	Identificador único. Garantiza que cada usuario se distinga inequívocamente.
	nombre	VARCHAR(100)	NOT NULL	Nombre completo; se muestra en el navbar y en formularios de auditoría.
	email	VARCHAR(100) UNIQUE	NOT NULL, UNIQUE	Clave de login la restricción UNIQUE impide cuentas duplicadas.
	clave	VARCHAR(255)	NOT NULL	Almacena el hash de la contraseña (mediante password_hash()) para nunca guardar texto plano.
	rol	ENUM('admin','cajero')	NOT NULL, DEFAULT 'cajero'	Controla permisos: solo 'admin' puede eliminar o editar registros críticos y generar cuadre de caja.
	pregunta_seguridad	VARCHAR(255)	NOT NULL	Permite recuperación de contraseña; cada pregunta se guarda junto a un hash de la respuesta.
	respuesta_seguridad	VARCHAR(255)	NOT NULL	Hash de la respuesta a la pregunta de seguridad, para validar sin exponer texto real.

insumos	id	INT(11) PK AUTO_INCREMENT	PRIMARY KEY	Identificador único de materia prima.
	nombre	VARCHAR(100)	NOT NULL, INDEX	Denominación del insumo; índice para acelerar consultas en recetas y gestión de stock.
	unidad_medida	VARCHAR(50)	NOT NULL	Unidad de control (kg, L, unidades), necesaria para los cálculos de stock.
	stock_actual	DECIMAL(10,2)	NOT NULL, DEFAULT 0.00	Nivel de inventario disponible; se actualiza en cada venta y reposición.
	stock_minimo	DECIMAL(10,2)	NOT NULL, DEFAULT 0.00	Umbral que dispara alertas de reposición en la UI del administrador.
recetas	id	INT(11) PK AUTO_INCREMENT	PRIMARY KEY	Identificador interno.
	producto_id	INT(11) FK	FOREIGN KEY productos.id ON DELETE CASCADE	Relaciona cada receta con un producto; si un producto se elimina, sus recetas también.
	insumo_id	INT(11) FK	FOREIGN KEY → insumos.id ON DELETE RESTRICT	Vincula el insumo necesario; RESTRICT evita eliminar insumos en uso.
	cantidad	DECIMAL(10,2)	NOT NULL, CHECK (cantidad > 0)	Cantidad de insumo requerida para una unidad de producto; valida que sea positiva.
pedidos	id	INT(11) PK AUTO_INCREMENT	PRIMARY KEY	Identificador de la transacción de venta.
	fecha	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Marca temporal, crucial para filtros diarios en cuadre y reportes.
	total	DECIMAL(10,2)	NOT NULL, CHECK (total >= 0)	Suma de subtotales. El CHECK refuerza que no sea negativo.
	medio_pago	ENUM('efectivo','transferencia')	NOT NULL	Método de pago; utilizado en el módulo de cuadre para separar totales.
	usuario_id	INT(11) FK	FOREIGN KEY usuarios.id ON	Registra qué cajero ejecutó la venta; SET NULL si el usuario se elimina (poco común).

			DELETE SET NULL	
Pedido_detalle	id	INT(11) PK AUTO_INCREMENT	PRIMARY KEY	Línea individual dentro de pedidos.
	pedido_id	INT(11) FK	FOREIGN KEY pedidos.id ON DELETE CASCADE	Asegura eliminación en cascada de detalles al borrar un pedido.
	producto_id	INT(11) FK	FOREIGN KEY productos.id ON DELETE RESTRICT	Vincula la línea al producto vendido; RESTRICT evita borrar productos históricos.
	cantidad	INT	NOT NULL, CHECK (cantidad > 0)	Unidades vendidas; CHECK garantiza al menos una unidad.
	precio_unitario	DECIMAL(10,2)	NOT NULL, CHECK	Precio aplicado; permite conservar históricos aunque el precio cambie luego.
egresos	id	INT(11) PK AUTO_INCREMENT	PRIMARY KEY	Identificador del gasto operativo.
	descripcion	VARCHAR(255)	NOT NULL	Motivo o detalle del egreso.
	monto	DECIMAL(10,2)	NOT NULL, CHECK (monto >= 0)	Valor económico del egreso. Avoid valores negativos.
	medio_pago	ENUM('efectivo','transferencia')	NOT NULL	Igual que en ventas, para comparar saldos.
	fecha	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Fecha de registro; usada en cuadre y filtros de reporte.
	imagen_factura	VARCHAR(255)	NULL	Ruta en public/facturas/; permite mostrar evidencia documental.
	usuario_id	INT(11) FK	FOREIGN KEY → usuarios.id ON DELETE SET NULL	Identifica quién registró el egreso; SET NULL si el usuario desaparece.
Entradas_insumo	id	INT(11) PK AUTO_INCREMENT	PRIMARY KEY	Identificador de reposición de insumo.
	insumo_id	INT(11) FK	FOREIGN KEY → insumos.id ON DELETE RESTRICT	Asocia la reposición al insumo; RESTRICT evita borrado de insumos con histórico.

	cantidad	DECIMAL(10,2)	NOT NULL, CHECK (cantidad > 0)	Cantidad agregada al inventario.
	fecha	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Fecha de la reposición; importante para auditoría de inventario.
	usuario_id	INT(11) FK	FOREIGN KEY → usuarios.id ON DELETE SET NULL	Usuario que validó o realizó la reposición; SET NULL si ya no existe
producto	id	INT(11)	NOT NULL	Identificador único del producto.
	nombre	VARCHAR(100)	NOT NULL	Nombre descriptivo del producto.
	precio	DECIMAL(10,2)	NOT NULL	Precio de venta por unidad; CHECK en la BD evita negativos.
	estado	ENUM('activo','inactivo')	NOT NULL	Control de disponibilidad: 'activo' o 'inactivo'.

Conexión a base de datos

La conexión a la base de datos ya creada se la realizo mediante un archivo llamado **database.php** la cual se encapsula una conexión usando PDO, definiendo los parámetros.

Ilustración 57 archivo .php de conexión a la base de datos

Código fuente

```php
<?php
class Database {
    private $host = "localhost";
    private $db_name = "coffee_burguer";
    private $username = "root";
    private $password = "";
    public $conn;

    public function connect() {
        $this->conn = null;
        try {
            $this->conn = new PDO(
```

```
                    "mysql:host=" . $this->host . ";dbname=" . $this-
>db_name,
                    $this->username,
                    $this->password
                );
                $this->conn->exec("set names utf8");
            } catch(PDOException $exception) {
                echo "Error de conexión: " . $exception->getMessage();
            }
            return $this->conn;
        }
    }
```

Módulo de ventas

Permite registrar pedidos y calcular automáticamente el total de la venta. Incluye la selección del medio de pago (efectivo o transferencia). Cada venta descuenta los insumos definidos en la tabla recetas.

Archivos principales:

controllers/VentaController.php

```
<?php
session_start();
require_once __DIR__ . '/../models/Venta.php';

$venta = new Venta();

//  Verificación de stock desde JavaScript (AJAX GET)
if (isset($_GET['verificar_stock'])) {
    $producto_id = $_GET['verificar_stock'];

    $maximo = 1000; // Límite alto inicial
    $faltantes = [];

    for ($i = 1; $i <= 999; $i++) {
        $faltantesSim = $venta-
>verificarStockDisponible($producto_id, $i);
        if (!empty($faltantesSim)) {
            $maximo = $i - 1;
            $faltantes = array_map(function($f) {
```

```
                return "{$f['insumo']} (disponible: {$f['disponible']}, necesita: {$f['necesita']})";
            }, $faltantesSim);
            break;
        }
    }

    echo json_encode([
        'maximo_permitido' => $maximo,
        'faltantes' => $faltantes
    ]);
    exit;
}

// Registro de venta con verificación de stock del lado del servidor
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (!isset($_SESSION['usuario_id'])) {
        header("Location: ../views/login.php");
        exit();
    }

    $usuario_id = $_SESSION['usuario_id'];
    $medio_pago = $_POST['medio_pago'];
    $total = $_POST['total'];
    $productos = json_decode($_POST['productos'], true);

    // Verificar el stock de todos los productos antes de registrar
    foreach ($productos as $p) {
        $faltantes = $venta->verificarStockDisponible($p['id'], $p['cantidad']);
        if (!empty($faltantes)) {
            $mensaje = "Stock insuficiente para el producto '{$p['nombre']}'.";
            $_SESSION['error_venta'] = $mensaje;
            header("Location: ../views/nueva_venta.php?error=stock");
            exit();
        }
    }

    // Todo verificado, registrar
    $venta->registrarPedido($usuario_id, $medio_pago, $total, $productos);
    header("Location: ../views/dashboard.php?mensaje=venta_exitosa");
    exit();
}
```

models/Venta.php

```php
class Venta {
    private $conn;

    public function __construct() {
        $database = new Database();
        $this->conn = $database->connect();
    }

    //  Obtener productos para el menú
    public function obtenerProductos() {
        $stmt = $this->conn->prepare("SELECT * FROM productos WHERE estado = 'activo'");
        $stmt->execute();
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }

 // Registrar un nuevo pedido con productos y descontar inventario
public function registrarPedido($usuario_id, $medio_pago, $total, $productos) {
    $this->conn->beginTransaction();

    // 1. Registrar el pedido
    $stmt = $this->conn->prepare("INSERT INTO pedidos (usuario_id, medio_pago, total) VALUES (?, ?, ?)");
    $stmt->execute([$usuario_id, $medio_pago, $total]);
    $pedido_id = $this->conn->lastInsertId();

    // 2. Registrar detalles del pedido
    $stmtDetalle = $this->conn->prepare("INSERT INTO pedido_detalle (pedido_id, producto_id, cantidad, precio_unitario) VALUES (?, ?, ?, ?)");

    foreach ($productos as $producto) {
        $stmtDetalle->execute([
            $pedido_id,
            $producto['id'],
            $producto['cantidad'],
            $producto['precio']
        ]);
    }

    // 3. Descontar insumos del inventario según la receta de cada producto
    require_once __DIR__ . '/Receta.php';
```

```
        $recetaModel = new Receta();

        foreach ($productos as $producto) {
            $producto_id = $producto['id'];
            $cantidad_vendida = $producto['cantidad'];

            // Obtener receta del producto
            $receta = $recetaModel->obtenerPorProducto($producto_id);

            foreach ($receta as $componente) {
                $insumo_id = $componente['insumo_id'];
                $cantidad_usada = $componente['cantidad'] *
    $cantidad_vendida;

                // Descontar del inventario
                $stmtDescuento = $this->conn->prepare("UPDATE insumos
    SET stock_actual = stock_actual - :cantidad WHERE id = :id");
                $stmtDescuento->execute([
                    ':cantidad' => $cantidad_usada,
                    ':id' => $insumo_id
                ]);
            }
        }

        // Finalizar la transacción
        $this->conn->commit();
        return true;
    }
```

views/nueva_venta.php

```
            <!-- Resumen del pedido -->
            <div class="col-md-5">
                <div class="resumen-box">
                    <h5>Resumen del Pedido</h5>
                    <form action="../controllers/VentaController.php"
    method="POST" onsubmit="return validarEnvio()">
                        <div class="table-responsive">
                            <table class="table table-sm table-bordered
    text-center align-middle" id="tablaResumen">
                                <thead class="table-dark">
                                    <tr>
                                        <th>Producto</th>
                                        <th>Cant.</th>
                                        <th>Subtotal</th>
                                        <th></th>
                                    </tr>
```

```
                            </thead>
                            <tbody></tbody>
                        </table>
                    </div>

                    <p class="fs-6"><strong>Total: $<span
id="total">0.00</span></strong></p>

                    <div class="mb-2">
                        <label class="form-label">Método de
pago:</label>
                        <select name="medio_pago" class="form-
select form-select-sm" required>
                            <option value="">Seleccione</option>
                            <option
value="efectivo">Efectivo</option>
                            <option
value="transferencia">Transferencia</option>
                        </select>
                    </div>

                    <input type="hidden" name="productos"
id="productos">
                    <input type="hidden" name="total"
id="total_hidden">

                    <button type="submit" class="btn btn-success
btn-sm"> Guardar venta</button>
                    <a href="dashboard.php" class="btn btn-outline-
secondary btn-sm">⬅ Volver</a>
                </form>
            </div>
        </div>
    </div>
</div>

<script>
let productos = [];

function agregarProducto(id, nombre, precio) {
    fetch(`../controllers/VentaController.php?verificar_stock=${id}
`)
        .then(res => res.json())
        .then(data => {
            const maximo = data.maximo_permitido;
            const faltantes = data.faltantes;

            const existente = productos.find(p => p.id === id);

```

```
            if (existente) {
                if (existente.cantidad + 1 > maximo) {
                    alert(`Solo puedes agregar hasta ${maximo} unidades de ${nombre}. Faltan insumos:\n- ${faltantes.join("\n- ")}`);
                    return;
                }
                existente.cantidad++;
            } else {
                if (maximo < 1) {
                    alert(`No hay stock suficiente para vender '${nombre}'. Faltan insumos:\n- ${faltantes.join("\n- ")}`);
                    return;
                }
                productos.push({ id, nombre, precio, cantidad: 1 });
            }

            renderTabla();
        })
        .catch(err => {
            console.error("Error al verificar stock:", err);
            alert("No se pudo verificar el inventario. Intenta de nuevo.");
        });
}
```

Tablas relacionadas:

- pedidos: cabecera de cada venta.
- pedido_detalle: detalle de productos.
- productos, recetas, insumos: para control automático del stock.

Módulo de Egresos

Gestiona los gastos de la empresa. Permite registrar un egreso con descripción, monto y adjuntar imagen de la factura digital, vinculando el registro al usuario que lo generó para trazabilidad.

Arquitectura (MVC) y archivos principales

controllers/EgresoController.php

```
$egreso = new Egreso();
```

```
      // Registrar nuevo egreso
      if ($_SERVER["REQUEST_METHOD"] === "POST" &&
isset($_SESSION['usuario_id'])) {
          $accion = $_POST['accion'] ?? 'crear';

          if ($accion === 'crear') {
              $descripcion = $_POST['descripcion'];
              $monto = $_POST['monto'];
              $medio_pago = $_POST['medio_pago'];
              $usuario_id = $_SESSION['usuario_id'];

              $imagen = null;
              if (isset($_FILES['imagen']) && $_FILES['imagen']['error']
=== UPLOAD_ERR_OK) {
                  $nombreArchivo = time() . '_' .
basename($_FILES['imagen']['name']);
                  $rutaRelativa = "public/facturas/" . $nombreArchivo;
                  $rutaDestino = "../" . $rutaRelativa;

                  if (move_uploaded_file($_FILES['imagen']['tmp_name'],
$rutaDestino)) {
                      $imagen = $rutaRelativa;
                  }
              }

              if ($egreso->registrar($descripcion, $monto, $medio_pago,
$imagen, $usuario_id)) {
                  header("Location:
../views/egresos.php?mensaje=creado");
                  exit();
              } else {
                  header("Location: ../views/egresos.php?mensaje=error");
                  exit();
              }
          }
```

models/Egreso.php

```
      <?php
      require_once __DIR__ . '/../config/database.php';

      class Egreso {
          private $conn;
          private $table = 'egresos';

          public function __construct() {
              $database = new Database();
```

```
            $this->conn = $database->connect();
        }

        // Registrar un nuevo egreso
        public function registrar($descripcion, $monto, $medio_pago,
$imagen, $usuario_id) {
            $query = "INSERT INTO $this->table (descripcion, monto,
medio_pago, imagen_factura, usuario_id)
                      VALUES (:descripcion, :monto, :medio_pago,
:imagen_factura, :usuario_id)";
            $stmt = $this->conn->prepare($query);
            $stmt->bindParam(':descripcion', $descripcion);
            $stmt->bindParam(':monto', $monto);
            $stmt->bindParam(':medio_pago', $medio_pago);
            $stmt->bindParam(':imagen_factura', $imagen);
            $stmt->bindParam(':usuario_id', $usuario_id);

            return $stmt->execute();
        }
```

views/egresos.php

```
    <head>
        <meta charset="UTF-8">
        <title>Registrar Egreso - Coffee Burguer</title>
        <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min
.css" rel="stylesheet">
        <style>
            body {
                background: linear-gradient(135deg, #fceabb, #f8b500);
                min-height: 100vh;
            }

            .form-container {
                background-color: #fff;
                max-width: 600px;
                margin: 40px auto;
                padding: 40px;
                border-radius: 20px;
                box-shadow: 0 10px 25px rgba(0, 0, 0, 0.1);
            }

            .form-container h2 {
                margin-bottom: 25px;
                font-weight: bold;
                color: #333;
```

```
            }

            .btn-primary {
                background-color: #f0932b;
                border: none;
            }

            .btn-primary:hover {
                background-color: #eb7c00;
            }

            .form-label {
                font-weight: 500;
            }
        </style>
    </head>
    <body>

    <?php include 'header.php'; ?>

    <div class="container">
        <div class="form-container">
            <h2 class="text-center"> Registrar Egreso</h2>

            <form action="../controllers/EgresoController.php"
method="POST" enctype="multipart/form-data">
                <input type="hidden" name="accion" value="crear">

                <div class="mb-3">
                    <label class="form-label">Descripción:</label>
                    <input type="text" name="descripcion" class="form-
control" required>
                </div>

                <div class="mb-3">
                    <label class="form-label">Monto ($):</label>
                    <input type="number" step="0.01" name="monto"
class="form-control" required>
                </div>

                <div class="mb-3">
                    <label class="form-label">Medio de pago:</label>
                    <select name="medio_pago" class="form-select"
required>
                        <option value="efectivo">Efectivo</option>
                        <option
value="transferencia">Transferencia</option>
                    </select>
                </div>
```

```
                <div class="mb-3">
                    <label class="form-label">Imagen de factura
(opcional):</label>
                    <input type="file" name="imagen" class="form-
control">
                </div>

                <div class="d-grid gap-2">
                    <button type="submit" class="btn btn-
primary">Registrar</button>
                    <a href="dashboard.php" class="btn btn-
secondary">⬅ Volver al panel</a>
                </div>
            </form>
        </div>
    </div>
```

Tablas relacionadas

La tabla egresos tiene todos los siguientes atributos (id, usuario_id FK usuarios.id, descripcion, monto, ruta_comprobante, fecha).

La para las validaciones debe ser obligatorio el monto ingresado debe ser mayor a 0, la imagen es opcional y el egreso solo lo puede editar o eliminar el administrador.

Rutas

- GET /egresos: listado de egresos
- GET /egresos/nuevo: formulario de registro
- POST /egresos/crear: guardar egreso con o sin comprobante (imagen)
- POST /egresos/borrar: (solo Administrador) eliminar egreso por id

Módulo de Inventario con Recetas

Este módulo es quien controla la entrada y salida de insumos. Cada producto está asociado a una receta que define cuántos insumos se descuentan al realizar una venta. Permite registrar entradas, configurar stocks mínimos y mantener la consistencia del inventario.

Arquitectura (MVC) y archivos principales

controllers/InsumoController.php

```
<?php
require_once __DIR__ . '/../models/Insumo.php';
session_start();

$insumo = new Insumo();

// GUARDAR: Crear o actualizar insumo
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $nombre = $_POST['nombre'];
    $unidad = $_POST['unidad_medida'];
    $stock = $_POST['stock_actual'];
    $minimo = $_POST['stock_minimo'];

    if (isset($_POST['id'])) {
        // Actualizar
        $insumo->actualizar($_POST['id'], $nombre, $unidad, $stock,
$minimo);
    } else {
        // Crear nuevo
        $insumo->crear($nombre, $unidad, $stock, $minimo);
    }

    header('Location: ../views/insumos.php');
    exit();
}
```

controllers/ProductoController.php

```
<?php
require_once __DIR__ . '/../models/Producto.php';
session_start();

$producto = new Producto();

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
```

```
            $nombre = $_POST['nombre'];
            $precio = $_POST['precio'];

            if (isset($_POST['id'])) {
                $producto->actualizar($_POST['id'], $nombre, $precio);
            } else {
                $producto->crear($nombre, $precio);
            }

            header('Location: ../views/productos.php');
            exit();
        }

        if (isset($_GET['desactivar'])) {
            $producto->cambiarEstado($_GET['desactivar'], 'inactivo');
            header('Location: ../views/productos.php');
            exit();
        }

        if (isset($_GET['activar'])) {
            $producto->cambiarEstado($_GET['activar'], 'activo');
            header('Location: ../views/productos.php');
            exit();
        }
```

controllers/RecetaController.php

```
        <?php
        require_once __DIR__ . '/../models/Receta.php';
        session_start();

        $receta = new Receta();

        // Agregar insumo a receta
        if ($_SERVER['REQUEST_METHOD'] === 'POST' &&
isset($_POST['producto_id'], $_POST['insumo_id'], $_POST['cantidad'])) {
            $producto_id = $_POST['producto_id'];
            $insumo_id = $_POST['insumo_id'];
            $cantidad = $_POST['cantidad'];

            // Si viene desde editar_receta.php (lleva extra parámetro
'editar')
            if (isset($_POST['editar'])) {
                $receta->actualizarCantidad($producto_id, $insumo_id,
$cantidad);
                header("Location:
../views/definir_receta.php?producto_id=$producto_id&mensaje=actualizado"
);
                exit();
            }
```

```
        // Si es una nueva receta (desde definir_receta.php)
        $receta->agregar($producto_id, $insumo_id, $cantidad);
        header("Location:
../views/definir_receta.php?producto_id=$producto_id&mensaje=agregado");
        exit();
    }
```

models/Insumo.php

```
    class Insumo {
        private $conn;
        private $table = 'insumos';

        public function __construct() {
            $database = new Database();
            $this->conn = $database->connect();
        }

        public function obtenerTodos() {
            $query = "SELECT * FROM " . $this->table;
            return $this->conn->query($query)-
>fetchAll(PDO::FETCH_ASSOC);
        }
```

models/Producto.php

```
    class Producto {
        private $conn;

        public function __construct() {
            $db = new Database();
            $this->conn = $db->connect();
        }

        // Obtener todos los productos
        public function obtenerTodos() {
            $stmt = $this->conn->query("SELECT * FROM productos ORDER
BY nombre");
            return $stmt->fetchAll(PDO::FETCH_ASSOC);
        }
```

models/Receta.php

```
      class Receta {
          private $conn;
          private $table = 'recetas';

          public function __construct() {
              $db = new Database();
              $this->conn = $db->connect();
          }

          public function obtenerProductos() {
              return $this->conn->query("SELECT * FROM productos")-
>fetchAll(PDO::FETCH_ASSOC);
          }

          public function obtenerInsumos() {
              return $this->conn->query("SELECT * FROM insumos")-
>fetchAll(PDO::FETCH_ASSOC);
          }
```

views/insumos.php

```
      div class="container shadow p-4 bg-white rounded">
          <h2 class="mb-4"><?= $editando ? 'Editar Insumo' : 'Nuevo
Insumo'; ?></h2>

          <form action="../controllers/InsumoController.php"
method="POST">
              <?php if ($editando): ?>
                  <input type="hidden" name="id" value="<?= $data['id']
?>">
              <?php endif; ?>

              <div class="mb-3">
                  <label class="form-label">Nombre:</label>
                  <input type="text" class="form-control" name="nombre"
value="<?= htmlspecialchars($data['nombre']) ?>" required>
              </div>

              <div class="mb-3">
                  <label class="form-label">Unidad de medida:</label>
                  <input type="text" class="form-control"
name="unidad_medida" value="<?= htmlspecialchars($data['unidad_medida'])
?>" required>
              </div>

              <div class="mb-3">
                  <label class="form-label">Stock actual:</label>
```

```
                <input type="number" class="form-control"
name="stock_actual" step="0.01" value="<?= $data['stock_actual'] ?>"
required>
            </div>

            <div class="mb-3">
                <label class="form-label">Stock mínimo:</label>
                <input type="number" class="form-control"
name="stock_minimo" step="0.01" value="<?= $data['stock_minimo'] ?>"
required>
            </div>

            <div class="d-flex justify-content-between">
                <a href="insumos.php" class="btn btn-outline-dark">
                    <i class="fas fa-arrow-left"></i> Volver
                </a>
                <button type="submit" class="btn btn-primary">
                    <i class="fas fa-save"></i> Guardar
                </button>
            </div>
        </form>
    </div>
```

views/productos.php

```
    div class="container">
        <div class="form-container">
            <h2 class="text-center"><?= $titulo ?> 🛒</h2>

            <form action="../controllers/ProductoController.php"
method="POST">
                <input type="hidden" name="accion" value="<?= $modo
?>">
                <?php if ($modo === 'actualizar'): ?>
                    <input type="hidden" name="id" value="<?=
$datos['id'] ?>">
                <?php endif; ?>

                <div class="mb-3">
                    <label for="nombre" class="form-label">Nombre del
producto:</label>
                    <input type="text" name="nombre" id="nombre"
class="form-control" required value="<?=
htmlspecialchars($datos['nombre']) ?>">
                </div>

                <div class="mb-3">
```

```
                    <label for="precio" class="form-label">Precio
($):</label>
                    <input type="number" step="0.01" name="precio"
id="precio" class="form-control" required value="<?=
htmlspecialchars($datos['precio']) ?>">
                </div>

                <div class="d-grid gap-2">
                    <button type="submit" class="btn btn-primary">💾
Guardar</button>
                    <a href="productos.php" class="btn btn-
secondary">Cancelar</a>
                </div>
```

views/definir_recetas.php

```
    <div class="card-receta">
        <h3 class="text-center mb-4"><i class="bi bi-journal-plus"></i>
Definir Receta por Producto</h3>
        <div class="mb-3">
            <a href="dashboard.php" class="btn btn-outline-dark"><i
class="bi bi-arrow-left-circle"></i> Volver al panel</a>
        </div>

        <form method="GET" class="row g-3 align-items-center mb-4">
            <div class="col-auto">
                <label class="col-form-label fw-bold">Selecciona un
producto:</label>
            </div>
            <div class="col-auto">
                <select name="producto_id" class="form-select"
onchange="this.form.submit()">
                    <option value="">-- Elegir --</option>
                    <?php foreach ($productos as $p): ?>
                        <option value="<?= $p['id'] ?>" <?=
$producto_id == $p['id'] ? 'selected' : '' ?>>
                            <?= htmlspecialchars($p['nombre']) ?>
                        </option>
                    <?php endforeach; ?>
                </select>
            </div>
        </form>
```

Rutas / Endpoints

- GET /insumos, POST /insumos/crear → CRUD de insumos

- GET /productos, POST /productos/crear → CRUD de productos
- GET /recetas?producto={id} → ver/editar receta de un producto
- POST /recetas/guardar → persistir receta (con transacción)

Las tablas relacionadas son las siguientes:

- insumos (id, nombre, unidad, stock_actual, stock_minimo)
- entradas_insumo (id, insumo_id FK, cantidad, fecha)
- productos (id, nombre, precio, estado)
- recetas (producto_id FK, insumo_id FK, cantidad)

Es importante saber que las cantidades de los insumos deben ser ingresadas en números positivos, de igual manera las recetas o deben ser vacías.

Módulo de Cuadre de Caja

Una vez terminado la venta de todo el día, se procede a cerrar caja para todo esto se necesita tener todos los ingresos(pedidos) y todos los egresos ya sea realizados por el

Arquitectura (MVC) y archivos principales

views/cuadre_caja.php

```
<?php
require_once __DIR__ . '/../models/Venta.php';
require_once __DIR__ . '/../models/Egreso.php';
session_start();

if (!isset($_SESSION['rol']) || $_SESSION['rol'] !== 'admin') {
    echo "Acceso denegado";
    exit();
}

$fecha = $_GET['fecha'] ?? date('Y-m-d');

$ventaModel = new Venta();
$egresoModel = new Egreso();

$efectivo = $ventaModel->totalPorMedioPago('efectivo', $fecha);
$transferencia = $ventaModel->totalPorMedioPago('transferencia',
$fecha);
$egresos = $egresoModel->totalEgresosPorFecha($fecha);
$saldo = $efectivo + $transferencia - $egresos;
?>

<!DOCTYPE html>
```

```
      <html lang="es">
      <head>
          <meta charset="UTF-8">
          <title>Cuadre de Caja</title>
          <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min
.css" rel="stylesheet">
          <link href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.10.5/font/bootstrap-icons.css" rel="stylesheet">
          <style>
              body {
                  background: linear-gradient(to right, #ffeaa7,
#fab1a0);
                  min-height: 100vh;
                  padding: 30px;
              }
              .card {
                  border-radius: 15px;
                  box-shadow: 0 8px 25px rgba(0, 0, 0, 0.1);
              }
              .table th, .table td {
                  vertical-align: middle;
              }
          </style>
      </head>
      <body>

      <div class="container">
          <h2 class="text-center mb-4"><i class="bi bi-cash-coin"></i>
Cuadre de Caja - <?= $fecha ?></h2>

          <form method="GET" class="row g-3 align-items-center mb-4">
              <div class="col-auto">
                  <label class="col-form-label">🧮 Selecciona una
fecha:</label>
              </div>
              <div class="col-auto">
                  <input type="date" name="fecha" class="form-control"
value="<?= $fecha ?>">
              </div>
              <div class="col-auto">
                  <button type="submit" class="btn btn-primary"><i
class="bi bi-search"></i> Filtrar</button>
              </div>
          </form>

          <div class="card p-4 mb-4">
              <table class="table table-bordered table-hover">
```

```
                <tr><th>Ingresos en efectivo:</th><td>$<?=
number_format($efectivo, 2) ?></td></tr>
                <tr><th>Ingresos por transferencia:</th><td>$<?=
number_format($transferencia, 2) ?></td></tr>
                <tr><th>Total egresos:</th><td>$<?=
number_format($egresos, 2) ?></td></tr>
                <tr class="table-success">
                    <th><strong>Saldo final:</strong></th>
                    <td><strong>$<?= number_format($saldo, 2)
?></strong></td>
                </tr>
            </table>
        </div>

        <form method="GET" action="exportar_cuadre_excel.php" class="d-
inline">
            <input type="hidden" name="fecha" value="<?= $fecha ?>">
            <button type="submit" class="btn btn-outline-success"><i
class="bi bi-file-earmark-excel-fill"></i> Descargar Excel</button>
        </form>

        <a href="dashboard.php" class="btn btn-link mt-4"><i class="bi
bi-arrow-left-circle"></i> Volver al panel</a>
    </div>

    </body>
    </html>
```

Para que sea posible el cuadre de caja se utiliza el archivo:

views/cuadre_caja.php es quien control a la sesión y lee por fechas, consume los modelos y renderiza HTML.

Se utiliza los modelos:

- models/Venta.php (totales por medio de pago)
- models/Egreso.php (totales por fecha)
  para la exportacion a un archivo tipo Excel.

- views/exportar_cuadre_excel.php (descarga archivo Excel con el mismo cálculo).

```
    <?php
    require_once __DIR__ . '/../models/Venta.php';
```

```
        require_once __DIR__ . '/../models/Egreso.php';

        $fecha = $_GET['fecha'] ?? date('Y-m-d');

        $ventaModel = new Venta();
        $egresoModel = new Egreso();

        $efectivo = $ventaModel->totalPorMedioPago('efectivo',
$fecha);
        $transferencia = $ventaModel-
>totalPorMedioPago('transferencia', $fecha);
        $egresos = $egresoModel->totalEgresosPorFecha($fecha);
        $saldo = $efectivo + $transferencia - $egresos;

        // Cabeceras para descargar como Excel
        header("Content-Type: application/vnd.ms-excel");
        header("Content-Disposition: attachment;
filename=cuadre_caja_" . $fecha . ".xls");
        header("Pragma: no-cache");
        header("Expires: 0");

        // Contenido de la tabla
        echo "<table border='1'>";
        echo "<tr><th colspan='2'><strong>Cuadre de Caja -
{$fecha}</strong></th></tr>";
        echo "<tr><th>Concepto</th><th>Monto (USD)</th></tr>";
        echo "<tr><td>Ingresos en efectivo</td><td>" .
number_format($efectivo, 2) . "</td></tr>";
        echo "<tr><td>Ingresos por transferencia</td><td>" .
number_format($transferencia, 2) . "</td></tr>";
        echo "<tr><td>Total egresos</td><td>" .
number_format($egresos, 2) . "</td></tr>";
        echo "<tr><td><strong>Saldo final</strong></td><td><strong>"
. number_format($saldo, 2) . "</strong></td></tr>";
        echo "</table>";
```

Hay que tener en cuenta que el rol de administrador es quien puede realizar el cuadre de caja así mismo la exportación a un archivo Excel, ya que existe restricciones de cada usuario.

Conclusiones técnicas

El sistema Coffee Burguer es una aplicación web en PHP con MySQL que organiza el proceso completo: ventas, control de insumos por recetas, registro de egresos y cuadre diario. Con esto reducimos errores y dejamos un registro claro por usuario y por fecha. La versión actual

funciona estable de forma local; para producción bastaría fortalecer la seguridad, programar respaldos automáticos y desplegarlo en la nube. La base de datos tiene relaciones que cuidan la integridad de la información y las ventas se graban con transacciones para evitar inconsistencias. Como trabajo futuro considero mejorar los reportes, permitir roles configurables y añadir facturación electrónica.