



UNIVERSIDAD DE GUADALAJARA
CENTRO UNIVERSITARIO DE LA COSTA SUR
DEPARTAMENTO DE INGENIERÍAS
INGENIERÍA EN TELEINFORMÁTICA

REPORTE PARA OBTENER EL TÍTULO DE INGENIERO EN
TELEINFORMÁTICA BAJO LA MODALIDAD DE TITULACIÓN
TESIS, TESINA O INFORMES
OPCIÓN

TESIS

CON EL TÍTULO

**“Desarrollo de un sistema de facilitación de
creación y gestión del corte de caja en la tienda
Santana Elia en Autlán de Navarro, Jalisco”**

PRESENTAN
César Adrian Gabriel Cruz

DIRECTOR
M. S. C. Agustín Jaime Núñez Rodríguez

ASESOR
Dr. Luis Isidro Aguirre Salas

Autlán de Navarro, Jalisco, Mayo 2025

INTEGRANTES



César Adrian Gabriel Cruz. Originario de Autlán de Navarro, Jalisco, es un estudiante destacado de la carrera de Ingeniería en Teleinformática, generación 2021-2025. Egreso del bachillerato en la Escuela Preparatoria Regional de Autlán en la generación 2015-2018. Las áreas de interés son la ciberseguridad, informática forense y redes.

ÍNDICE

CAPÍTULO 1: GENERALIDADES.....	5
Introducción.....	5
Objetivo general.....	16
Objetivos particulares.....	17
CAPÍTULO 2: MARCO TEÓRICO.....	18
2.1 Metodologías de Desarrollo Ágiles.....	18
2.1.1 SCRUM.....	19
2.2 Bases de datos.....	21
2.2.1 Tipos de Bases de Datos.....	22
2.2.2 Gestores de bases de datos.....	23
2.3 Lenguajes de programación.....	24
2.3.1 JavaScript.....	25
2.3.2 PHP.....	26
2.4 Herramientas de desarrollo.....	26
2.4.1 Visual Studio Code.....	26
2.4.2 (IDEs, Frameworks, controladores de versiones, etc).....	27
CAPÍTULO 3: DESARROLLO.....	32
3.1 Registrar gerente e iniciar sesión (login).....	32
3.1.1 Descripción del caso de uso.....	32
3.1.2 Modelo de datos.....	38
3.1.3 Diseño de la interfaz.....	39
3.1.4 Fragmentos de código.....	45
3.2 Registrar, inhabilitar y/o eliminar empleados.....	52
3.2.1 Descripción del caso de uso.....	52
3.2.2 Modelo de datos.....	55
3.2.3 Diseño de la interfaz de registro de empleados.....	56
3.2.4 Fragmentos de código.....	61
3.3 Actualizar datos de usuario.....	68
3.3.1 Descripción del caso de uso.....	68
3.3.2 Modelo de datos.....	69
3.3.3 Diseño de la interfaz de registro de empleados.....	70
3.3.4 Fragmentos de código.....	75
3.4 Registrar corte, gasto y/o pago con tarjeta.....	81
3.4.1 Descripción del caso de uso.....	81

3.4.2 Modelo de datos.....	83
3.4.3 Diseño de la interfaz.....	84
3.4.4 Fragmentos de código.....	98
3.5 Consultar historial de cortes realizados.....	105
3.5.1 Descripción del caso de uso.....	105
3.5.2 Modelo de datos.....	108
3.5.3 Diseño de la interfaz.....	109
3.5.4 Fragmentos de código.....	111
3.6 Visualizar gráficos.....	115
3.6.1 Descripción del caso de uso.....	115
3.6.2 Modelo de datos.....	116
3.6.3 Diseño de la interfaz.....	117
3.6.4 Fragmentos de código.....	119
3.7 Registrar, visualizar y eliminar importes.....	124
3.7.1 Descripción del caso de uso.....	124
3.7.2 Modelo de datos.....	132
3.7.3 Diseño de la interfaz.....	133
3.7.4 Fragmentos de código.....	137
3.8 Registrar, visualizar y eliminar productos faltantes.....	149
3.8.1 Descripción del caso de uso.....	149
3.8.2 Modelo de datos.....	150
3.8.3 Diseño de la interfaz.....	151
3.8.4 Fragmentos de código.....	156
3.9 Cerrar sesión.....	162
3.9.1 Descripción del caso de uso.....	162
3.9.2 Modelo de datos.....	162
3.9.3 Diseño de la interfaz de registro de empleados.....	163
3.9.4 Fragmentos de código.....	163
CAPÍTULO 4: RESULTADOS.....	165
4.1 Resultados de las pruebas.....	165
4.2 Conclusiones.....	171
4.3 Trabajo a futuro.....	173
BIBLIOGRAFÍA.....	178
APÉNDICE A. GLOSARIO TÉCNICO.....	180

CAPÍTULO 1: GENERALIDADES

Introducción

El proceso de corte de caja en la tienda Santana Elia ubicada en Brizuela #99-C, colonia Centro, C. P. 48900, en Autlán de Navarro, Jalisco, se realiza manualmente en una libreta exclusiva para la obtención y registro de la información, en esta libreta se registran datos necesarios para el corte de caja, donde los campos a llenar son:

- Total de caja que el empleado en turno dejará.
- Total de la venta del turno.
- Los gastos realizados en el turno.
- El tiempo aire que aún hay en el sistema.
- El dinero recolectado del tiempo aire hasta ese momento.
- El dinero total que corresponde al área de tragamonedas.

Una vez tengamos todos los datos registrados en la libreta de manera correcta, el siguiente paso es tomar una fotografía utilizando dispositivo móvil personal, paso seguido, mediante el uso de la aplicación WhatsApp, se envía la imagen al encargado, quien es el responsable de llevar el control de todos esos registros.

Este método anticuado presenta la dependencia de los dispositivos móviles del personal, quienes de manera manual se debe enviar información financiera de la tienda haciendo uso de aplicaciones externas, además, el uso obsoleto de libretas de la que se requiere la compra continua y cuidado físico de la misma, aunque, el factor más determinante es el entendimiento de los datos (véase Figura 1.1.1 y Figura 1.1.2).

Las actuales limitaciones nos ha impulsado a explorar una solución más eficiente como segura e innovadora dada la actualidad en cuestiones financieras y/o administrativas, para ello, una implementación de un sistema digitalizado que nos permita realizar el corte de caja de forma confiable y

rápida, excluyendo el uso de dispositivos móviles personales y mejorando la gestión de la información financiera en el Santana Elia.

Como ejemplo, anteriormente se implementó un sistema que permitía registrar los importes de manera digital, este fue realizado por César Adrian su servidor, el sistema muestra una fotografía de una libreta de apuntes de recargas con fecha 25 al 29 de febrero de 2024 (véase Figura 1.1.3), como una muestra de la antigua metodología, debido a que de la misma manera se registran los importes, a continuación, se presenta la sustitución de esta libreta mediante un sistema digital (véase Figura 1.1.4), el cual logró satisfactoriamente reemplazar el uso de la libreta física, este sistema funcionaba mediante el uso de Java y MySQL a través de XAMPP, proporcionando una alternativa más eficiente y precisa para la gestión de la información financiera.

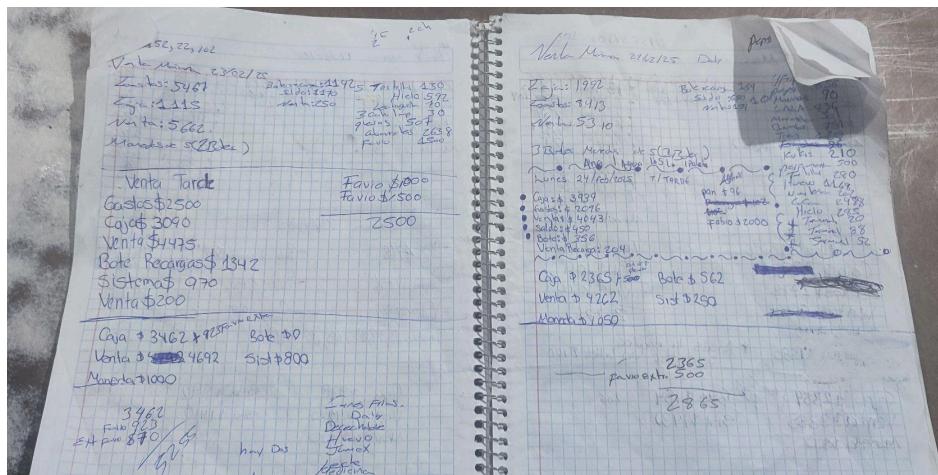


Figura 1.1.1 Libreta actual en el que se registra el corte y posterior envío al gerente mediante WhatsApp.

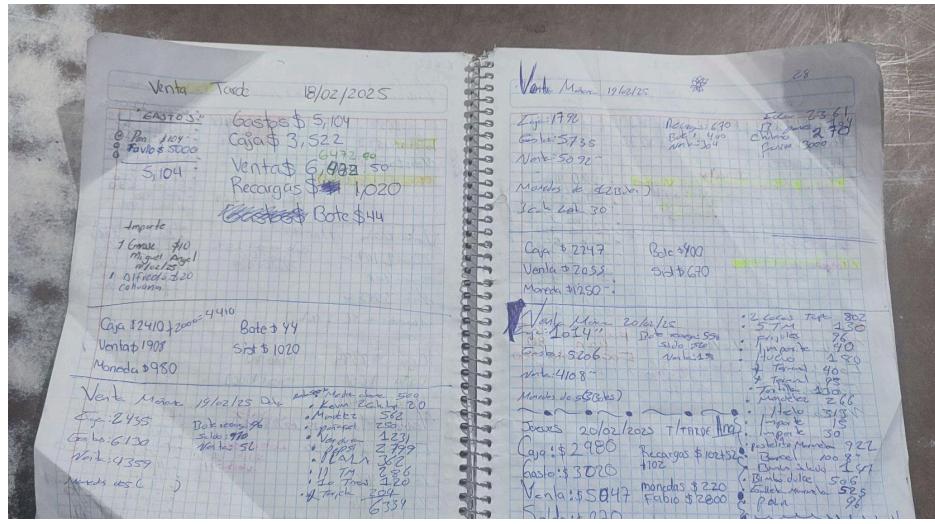


Figura 1.1.2 Otro ejemplo de la libreta actual en el que se registra el corte y posterior envío al gerente mediante WhatsApp.

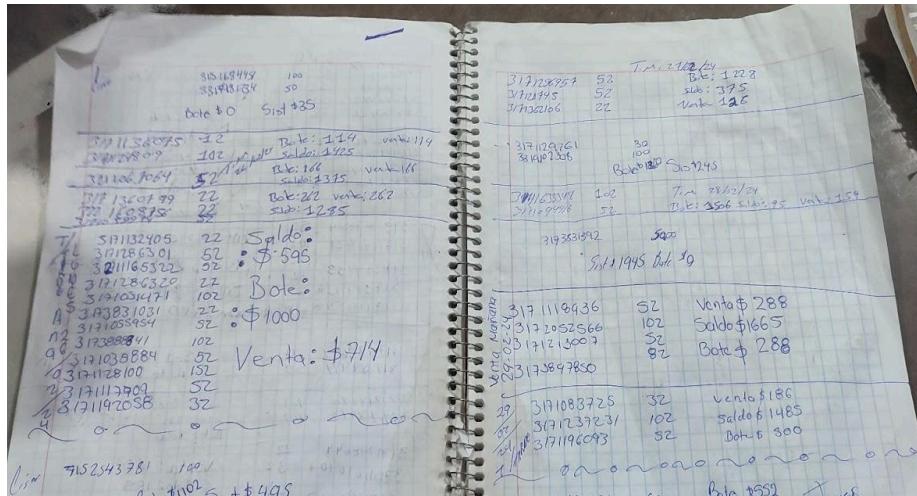


Figura 1.1.3 Libreta de apuntes de recargas utilizado anteriormente.
Actualmente ya no se usa este método.

FECHA	HORA	NOMBRE	ENVASE	CANTIDAD	IMPORTE
2024-02-29	23:16:18	aNDRES	CAGUAMA	3	\$30
2024-02-29	17:15:28	FERNANDO	CAGUAMA	2	\$20
2024-02-28	23:53:16	ADAN	CAGUAMA	2	\$20
2024-02-28	02:04:52	Moy	CAGUAMA	1	\$10
2024-02-27	23:03:30	Cristian	COCA 500ml	1	\$10
2024-02-26	23:49:24	Cristian Cobar...	COCA 1L	1	\$15

Figura 1.1.4 Software que reemplazo la libreta de importes, actualmente en funcionamiento.

(Planteamiento del problema) El actual proceso de corte de caja en la tienda Santana Elia ubicada en Brizuela #99-C, colonia Centro en Autlán de Navarro, C.P. 48900, ha demostrado tener limitaciones que impacta mucho en la entrega de información, se utiliza una libreta en la cual se registran los datos necesarios para cumplir el proceso, después, mediante una fotografía tomada por el dispositivo del personal se envía mediante WhatsApp al encargado de recolectar dichos datos.

Este proceso cuenta con tres limitaciones principales, uno de ellos, es la dependencia del dispositivo móvil del personal en turno, esto produce un riesgo de divulgación de la información financiera y es visto de informalidad de la tienda Santana Elia, otra limitación es el uso anticuado de libretas para la realización del proceso corte de caja, actualmente, todo está migrando a un proceso digital, la última y más apremiante limitación va encaminada con la primera, la seguridad de los datos hoy en día es fundamental no solo para las

empresas, también para cada individuo, corriendo el riesgo de que un tercero tenga acceso a los datos financieros de la tienda Santana Elia.

La implementación de un sistema digitalizado para el corte de caja crea una oportunidad para superar las limitaciones actuales, proporcionando rapidez, eficacia, seguridad, gestión y control de información financiera en el Santana Elia.

(Justificación) El proceso actual de corte de caja en la tienda Santana Elia, implica el uso de libretas, lapiceras y dispositivos móviles personales para el registro y envío de información financiera, la cual presenta limitaciones que afectan a la eficiencia en la gestión financiera.

Alguno de los problemas identificados son:

- La dependencia de dispositivos móviles personales: El uso de dispositivos personales para el registro y envío de información financiera, introduciendo riesgos de seguridad y falta de formalidad en el registro contable en el Santana Elia
- Uso obsoleto de libretas: El uso de libretas es poco práctica y propensa a sufrir daños físicos o pérdida de información, lo que afecta la integridad y confidencialidad de los datos.
- Gran posibilidad de errores humanos: Al hacer uso de libretas para la captura de información financiera está propensa a una gran posibilidad a errores humanos, lo que puede provocar datos financieros y lecturas erróneas para la elaboración de corte de caja.

La implementación de este proyecto al Santana Elia, no solo trae consigo beneficios en la mejora y sistematización en el proceso de corte de caja, también contribuye a la toma de decisiones en inversiones futuras con mayor información y proponiendo un entorno comercial más amigable al dueño como a los mismos empleados del Santana Elia.

Anteriormente, este sistema se empezó a realizar con anterioridad, sin embargo, de una manera robusta o poco eficiente, donde lo poco funcional

que este tenía, era observar los cortes realizados, a continuación, se mostraran algunas capturas de pantalla del antiguo sistema CorteCaja (Nombre anterior del sistema):

- El apartado principal donde seleccionamos iniciar sesión como gerente o empleado, además el apartado de registro (véase Figura 1.1.5)
- El inicio de sesión de empleados, viéndose de una manera muy simple, este login es similar al del gerente, ya que ambos tenían controladores de inicio de sesión por separado (véase Figura 1.1.6).
- El apartado principal del usuario logueado, muestra mediante un CardView, los cortes realizados con anterioridad, mostrando el más reciente de izquierda a derecha, además, podemos observar que las opciones disponibles para el empleado, es únicamente el registro del corte de caja (véase Figura 1.1.7).
- La interfaz para registrar un corte de caja, sigue siendo muy robusta y manual, donde el empleado tiene que realizar los cálculos necesarios y anotarlos en su campo correspondiente, siendo aún susceptible a errores de cálculo (véase Figura 1.1.8).
- Por último, para que un gerente se registre, aunque el formulario es lo requerido, mantiene la misma vista que los anteriores formularios, viéndose muy poco atractivo (véase Figura 1.1.9).

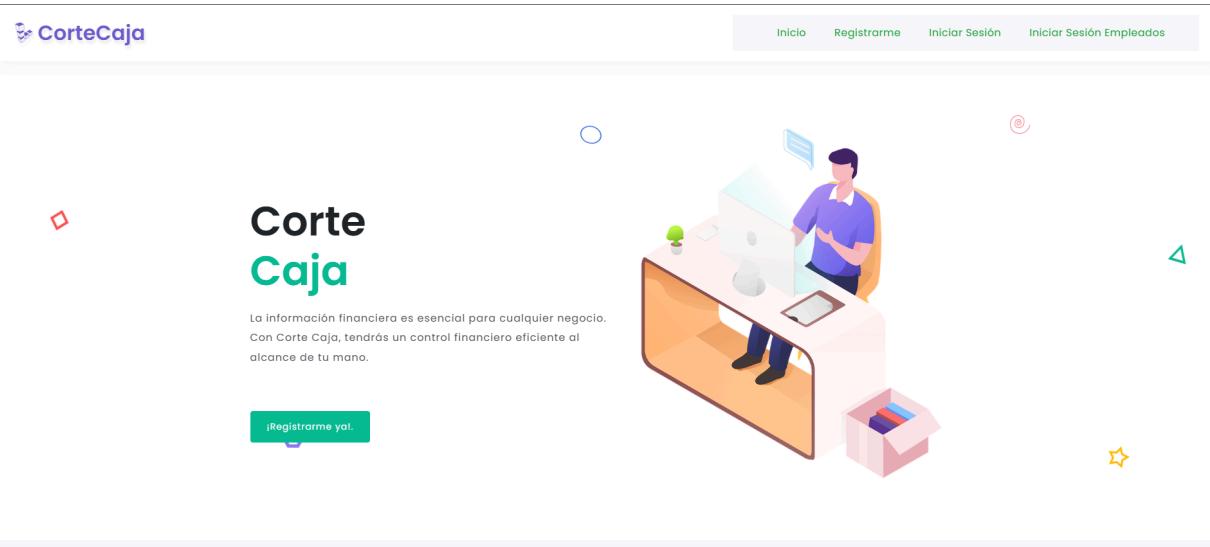


Figura 1.1.5 Inicio del sistema CorteCaja.

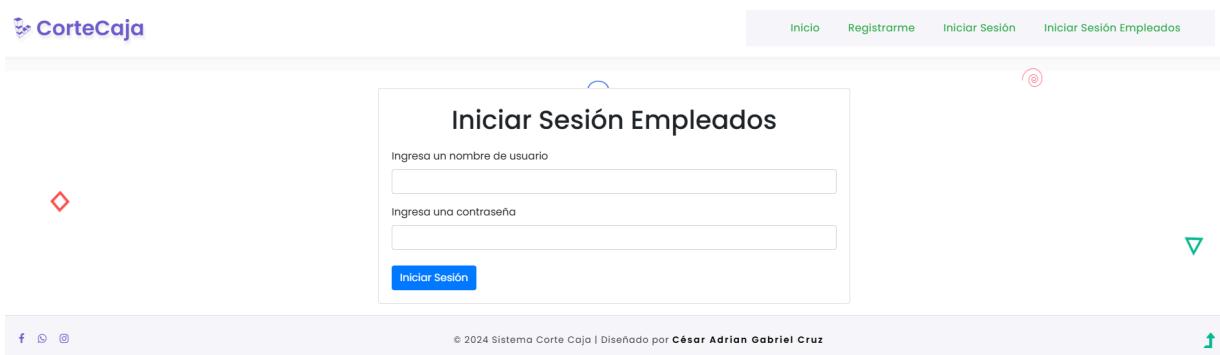


Figura 1.1.6 Logeo para los empleados.

Detalles del corte	Detalles del corte	Detalles del corte
<p>Empleado: César Gabriel Cruz</p> <p>Caja: 2022</p> <p>Venta: 443</p> <p>Gasto Total: 0</p> <p>Gasto Descripción: 0</p> <p>Sistema: 705</p> <p>Bote de Recargas: 680</p> <p>Caja Anterior: 1609</p> <p>Monedas: 1000</p> <p>Fecha: 2024-08-01 01:53:55</p>	<p>Empleado: Mia Keiry Aguilar Chávez</p> <p>Caja: 424</p> <p>Venta: 2918</p> <p>Gasto Total: 5901</p> <p>Gasto Descripción: medina 500 TIRAS 55,4TM 104,2FRUJOLES 44PEPSI 961,5TM 120,PEÑAFIEL 415,COCA 2062,MONDELEZ 528,LALA 472,HIELO 320,RECARGA 310.</p> <p>Sistema: 825</p> <p>Bote de Recargas: 510</p> <p>Caja Anterior: 3407</p> <p>Monedas: 2</p> <p>Fecha: 2024-07-31 14:03:23</p>	<p>Empleado: César Gabriel Cruz</p> <p>Caja: 2396</p> <p>Venta: 1093</p> <p>Gasto Total: 800</p> <p>Gasto Descripción: Pipis 700, pipis 100</p> <p>Sistema: 200</p> <p>Bote de Recargas: 1315</p> <p>Caja Anterior: 2103</p> <p>Monedas: 600</p> <p>Fecha: 2024-07-30 01:51:50</p>

Figura 1.1.7 Interfaz al iniciar sesión.

Crear Corte

Ingresá la cantidad de tu caja
Ejemplo: 4000

Ingresá la cantidad de tu venta
Ejemplo: 3000

Ingresá la cantidad de caja anterior
Ejemplo: 2000

Ingresá la suma de todos los gastos
Ejemplo: 1000

Ingresá todos los gastos y su cantidad (Descripción)
Ejemplo: Pan-\$100, Fabio-\$2000, Pipis-\$2000, Coca Cola-\$2000

Ingresá el saldo restante en sistema de recargas
Ejemplo: \$1000

Ingresá el dinero total en bote de recargas
Ejemplo: \$1000

Ingresá el total de monedas
Ejemplo: 1000

Agregar

Figura 1.1.8 Apartado para el registro del corte de caja.

Figura 1.1.9 Registro del administrador en el sistema.

En el mercado existen diversos sistemas diseñados para facilitar la realización del corte de caja, especialmente los sistemas de puntos de venta, que ofrecen una solución eficiente y precisa para la gestión de los datos, además, muchos de estos sistemas incluyen funciones complementarias que enriquecen su utilidad, permitiendo una mayor exactitud en los registros y una mejor toma de decisiones. A continuación, se describe un ejemplo de cómo funcionan estos sistemas.

El ejemplo más puntual dentro del espacio geográfico de la tienda Santana Elia, es [Abarrotes Punto de Venta #1 de México | Regístrate | SICAR ®](#), donde las funcionalidades que este software incluye son las siguientes:

- Control sobre robo hormiga: SICAR Abarrotes Punto de Venta cuenta con tecnología que te entrega cuentas claras y así podrás tener un mayor control de movimientos en tu negocio.
- Control de inventario: SICAR Abarrotes Punto de Venta te da la solución controlando el robo hormiga e inventario, reduciendo las pérdidas semana a semana.

- Importar/ exportar desde excel: SICAR Abarrotes Punto de Venta sabe lo importante que es ahorrar tiempo y esfuerzo, por tal motivo, cuenta con un importador de artículos desde Excel, fácil y seguro.
- Monedero electrónico: SICAR Abarrotes Punto de Venta te ayuda a crear clientes fieles a tu negocio que regresan una y otra vez a acumular y gastar los puntos que obtienen.
- Recargas y pago de servicios: SICAR Abarrotes Punto de Venta te ayuda a incrementar tus ventas ofreciendo servicios extra para tus clientes creando una mejor experiencia para tus clientes.
- Conexión con básculas: SICAR Abarrotes Punto de Venta obtén el precio según el peso del producto y cobra mucho más rápido.
- Lotes y caducidades: SICAR Abarrotes Punto de Venta te ayuda a controlar el lote y caducidad de todos tus productos para que así tus clientes no experimenten esta situación tan desagradable.
- Cortes de caja: SICAR Abarrotes Punto de Venta con poco esfuerzo y rápidamente obtendrás el corte de caja del día.
- Gráficas y comparativas: SICAR Abarrotes Punto de Venta cuenta con gráficas inteligentes que te mostrarán el crecimiento que has tenido usando comparativas entre fechas y así podrás tomar decisiones para el futuro de tu negocio.
- Movimientos de caja: SICAR Abarrotes Punto de Venta te muestra un estado de cuenta con todos los movimientos realizados para que a simple vista identifiques el problema.
- Reportes: SICAR Abarrotes Punto de Venta te muestra los artículos específicos que realmente necesita tu negocio.
- Facturación electrónica: SICAR Abarrotes Punto de Venta cuenta con el timbrado más rápido de América Latina, y con un 0% de fallas al momento de realizar facturas.

Las funcionalidades que ofrece el sistema SICAR Abarrotes Punto de Venta son complejas, efectivas y eficientes, este sistema ofrece herramientas claves para la gestión administrativa de un negocio, abarcando diversas áreas como el inventario, gestión financiera, robos hormiga, también ofrece la actualización de importar datos de Excel, servicios básicos, además, permite la conexión con básculas para agilizar cobros, una generación rápida de corte de caja y reportes entre muchas funciones más, un ejemplo de su visualización la encontramos a continuación (véase Figura 1.1.10).

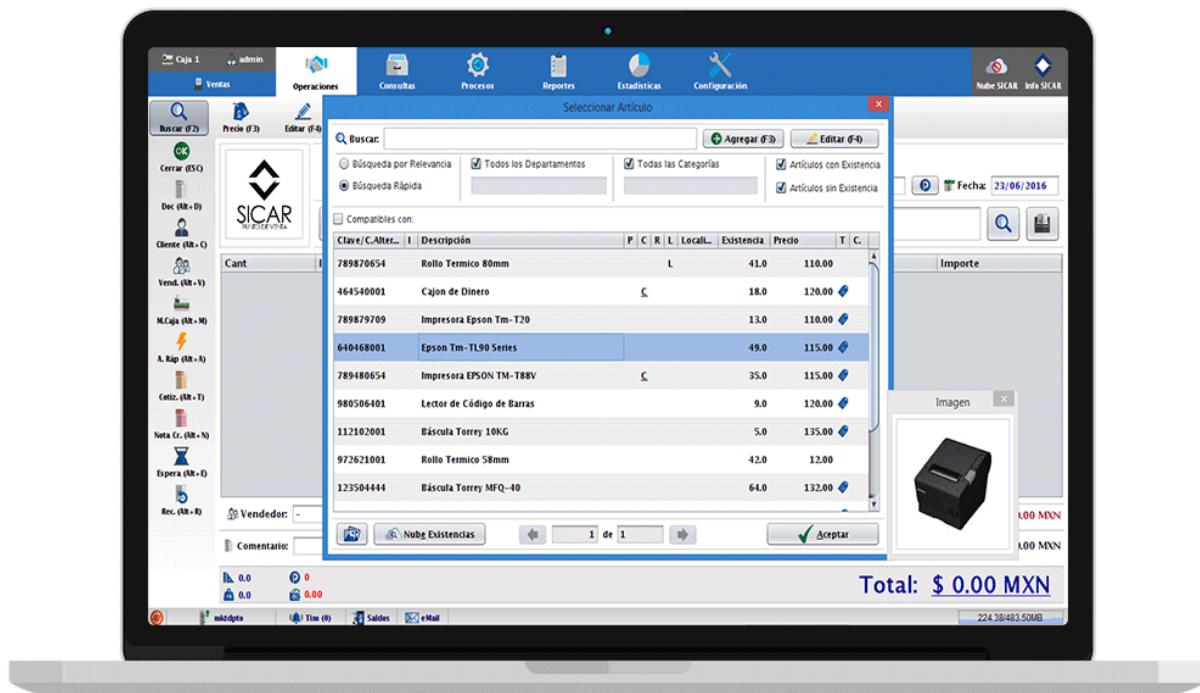


Figura 1.1.10 Representación visual del sistema SICAR Abarrotes Punto de Venta.

Aunque este sistema puede ser utilizado en Santana Elia, las principales desventajas de este software es el costo y su uso, a continuación se explicara dichas desventajas:

- Costo: Debido a lo completo que este software de la empresa SICAR está realizado, se deben realizar pagos para su uso, siendo en esto un precio alto en comparación de otras opciones similares, pero con un precio más accesible para las tiendas de abarrotes pequeñas, estos

software accesibles aunque no son completos como el que ofrece SICAR, si cumplen con los requisitos indispensables para el funcionamiento óptimo en una tienda de abarrotes.

- Uso del software: El software que presenta SICAR es demasiado excelente, que a su vez, lo vuelve complejo a personas que tienen un conocimiento de uso de computadora muy básico, en esto podemos integrar a la mayoría de personas mayores de 40 años que sean gerentes o empleadas de una tienda de abarrotes, donde mismos usuarios de tiendas de abarrotes que usan el software de SICAR, han comentado que al ser tan complejo el software, prefieren seguir con el uso anticuado de libretas, siendo este una posibilidad para expandirse el software SIGMA (Sistema Integral de Gestión para Microempresas de Abarrotes) en otros establecimientos ,

Con un poco mencionado los puntos positivos y negativos de algunos puntos de venta dentro de la región a la que pertenece Autlán de Navarro, la aplicación SIGMA (Sistema Integral de Gestión para Microempresas de Abarrotes) no busca ser un punto de venta, si no, un apoyo a la realización de un corte de caja como su mismo nombre lo indica, buscando ser fácil y sencillo su uso para el empleado o empleada donde requiera lo mínimo de conocimiento básico de uso de computadora, además, siendo intuitiva para el gerente la obtención específica de los datos que se necesitan.

Objetivo general

Desarrollar e implementar un sistema digitalizado para la elaboración de el corte de caja en la tienda Santana Elia, con el objetivo de mejorar el control de la información financiera de la tienda Santana Elia, adaptando este sistema digitalizado con la finalidad de una solución sistematizada y removiendo los mé todos actuales.

Objetivos particulares

1. Investigación sobre tecnologías y herramientas apropiadas al proyecto: Realizar una investigación amplia sobre tecnologías y herramientas disponibles para el desarrollo de nuestra aplicación corte de caja, proporcionando la mejor solución para el corte de caja.
2. Diseño general de la aplicación web: Diseñar la arquitectura del sistema para el corte de caja, tomando en cuenta la estructura de datos, interfaz de usuario, operaciones necesarias para el corte de caja y la separación de gerente y empleados.
3. Desarrollo de la aplicación web: Desarrollar la aplicación web, con los lenguajes de programación necesarios para su correcto funcionamiento.
4. Implementación del sistema en el Santana Elia: Implementar la aplicación web en la tienda Santana Elia con la realización de pruebas para garantizar el correcto funcionamiento de la aplicación web, también, en la búsqueda de bugs que pudieran surgir así como su solución.
5. Capacitación al personal: Proporcionar capacitación al personal de la tienda Santana Elia, en el uso correcto de la aplicación web para el corte de caja, resolviendo todas las dudas generadas por los mismos empleados.
6. Evaluación de la aplicación web: Evaluar la eficiencia de la aplicación web para el corte de caja, recopilando las ventajas y desventajas de un antes y después de la implementación de nuestra aplicación web en la tienda Santana Elia.

CAPÍTULO 2: MARCO TEÓRICO

2.1 Metodologías de Desarrollo Ágiles

Las metodologías ágiles permiten adaptar la forma de trabajo a las condiciones del proyecto, para conseguir una flexibilidad e inmediatez en la repuestos para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno, muchas empresas optan por ello para conseguir gestionar sus proyecto de forma flexible, autónoma y eficaz, logrando reducir los costes e incrementar la productividad. [1]

A continuación se explicaran algunos métodos:

- Extreme Programming XP: Es útil en entornos para startups o empresas que están en proceso de consolidación, siendo su principal objetivo ayudar en las relaciones entre los empleados y clientes.
- Scrum: Conocida también por ser la “metodología del caos” basada en una estructura de desarrollo incremental,donde el cielo de desarrollo del producto y/o servicio se desgrana en pequeños proyectos.
- Kanban: También conocida como “tarjeta visual” es muy útil para los responsables de proyectos, consiste en la elaboración de un cuadro o diagrama en el que se reflejan tres columnas de tareas, pendientes, en proceso o terminadas, con la finalidad de evitar la repetición de tareas o el olvido de alguna de ellas.
- Agile Inception: Orientada de los objetivos generales de las empresas, su meta es clarificar cuestiones como el tipo de cliente objetivos, las propuestas de valor añadido y las formas de venta, su objetivo es mediante pequeñas reuniones entre los socios y el equipo de trabajo en lo que las intervenciones no pueden superar los 5 minutos.

2.1.1 SCRUM

La metodología SCRUM es un proceso para llevar a cabo un conjunto de tareas de forma regular con el objetivo principal de trabajar de manera colaborativa, además de ofrecer un marco ágil que promueve la colaboración, la flexibilidad y la entrega rápida de productos de alta calidad, este método se van realizando entregas regulares y parciales del trabajo final, de manera prioritaria y en función del beneficio que aportan dichas entregas a los receptores del proyecto. [2]

¿Cuándo aplicar la metodología SCRUM? Se aplica en proyectos donde la obtención de resultados a corto plazo es necesaria y en aquellos en los que existen situaciones de incertidumbre y tareas poco definidas, también se resuelven aquellas partes de un proyecto que no se le está entregando a tiempo a un determinado cliente, cuando dichas tareas se dilatan demasiado en el tiempo o la calidad es baja.

Las fases de la metodología SCRUM son las siguientes (véase Figura 2.1.1.1):

1. Planificación: Product Backlog es la fase en la que se establecen las tareas prioritarias y donde se obtiene información breve y detallada sobre el proyecto que se va a desarrollar, es necesario para poder arrancar con el primer sprint, tiene permitido cambiar y crecer tantas veces como sea necesario en función del aprendizaje adquirido en el desarrollo del producto. Además de establecer tareas prioritarias, es fundamental que el Product Owner trabaje estrechamente con los stakeholders y el equipo de desarrollo para asegurar que el Product Backlog refleje las necesidades del negocio y las expectativas del cliente.
2. Ejecución: Sprint es el corazón, un intervalo de tiempo que como máximo tiene una duración de un mes y en donde se produce el desarrollo de un producto que es entregable potencialmente, también se

puede definir el Sprint como un mini proyecto en donde el equipo de trabajo se focaliza en el desarrollo de tareas para alcanzar el objetivo que se ha definido previamente en el Sprint planning, el Sprint culmina con una revisión del Sprint (Sprint Review), donde se presenta el trabajo completado a los stakeholders, y una retrospectiva del Sprint (Sprint Retrospective), donde el equipo reflexiona sobre el proceso y busca maneras de mejorar para el próximo Sprint.

3. Control y monitorización: Daily Scrum es una reunión diaria corta donde el equipo sincroniza actividades y reporta progresos y obstáculos. El Burn Down Chart es una herramienta visual para rastrear la cantidad de trabajo que queda versus el tiempo. Integrar el uso de herramientas de seguimiento de proyectos en tiempo real puede complementar el Burn Down Chart, proporcionando una visión más amplia del progreso y facilitando la comunicación entre el equipo.
4. Revisión y Adaptación: Sprint Review y Retrospective, al final de cada Sprint, el equipo realiza dos reuniones clave: la Sprint Review, para evaluar el trabajo completado y ajustar el Product Backlog si es necesario; y la Sprint Retrospective, donde el equipo reflexiona sobre su desempeño y busca formas de mejorar en el próximo Sprint. Utilizar las retrospectivas para establecer un plan de acción claro que aborde los problemas identificados puede mejorar significativamente la eficiencia y la moral del equipo.

La metodología SCRUM se centra en la mejora continua, la flexibilidad y la entrega de valor de manera eficiente.

SCRUM PROCESS

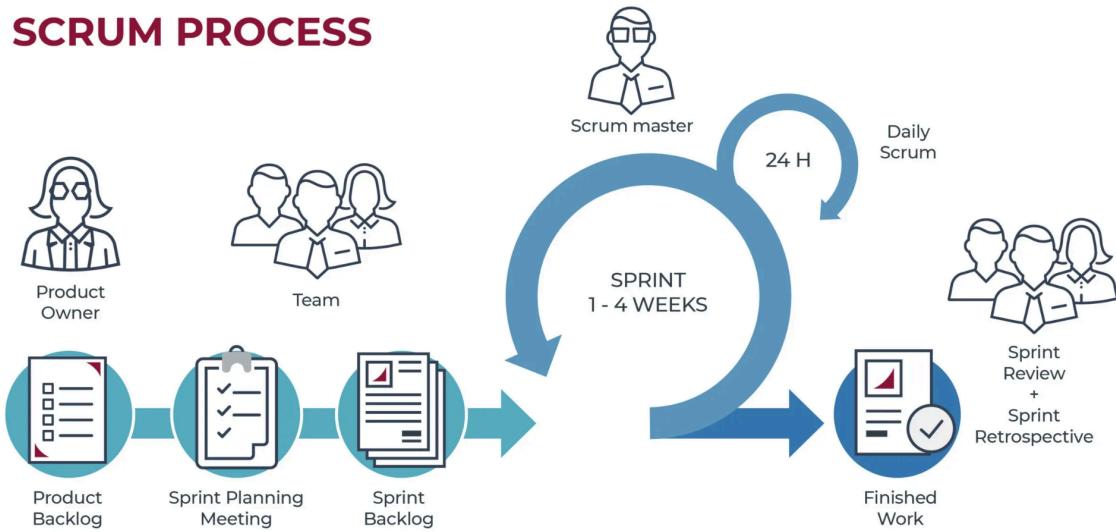


Figura 2.1.1.1 Imagen ilustrativa a la metodología SCRUM.

2.2 Bases de datos

Una base de datos es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático. Una base de datos está controlada por un sistema de gestión de bases de datos (DBMS), en conjunto, los datos y el DBMS, junto con las aplicaciones asociadas a ellos, reciben el nombre de sistema de bases de datos, abreviado normalmente a simplemente base de datos. Los datos de los tipos más comunes de bases de datos en funcionamiento actualmente se suelen utilizar como estructuras de filas y columnas (véase Figura 2.2.0.1) en una serie de tablas para aumentar la eficacia del procesamiento y la consulta de datos, así, se puede acceder, gestionar, modificar, actualizar, controlar y organizar fácilmente los datos. La mayoría de las bases de datos utilizan un lenguaje de consulta estructurada (SQL) para escribir y consultar datos. [3]

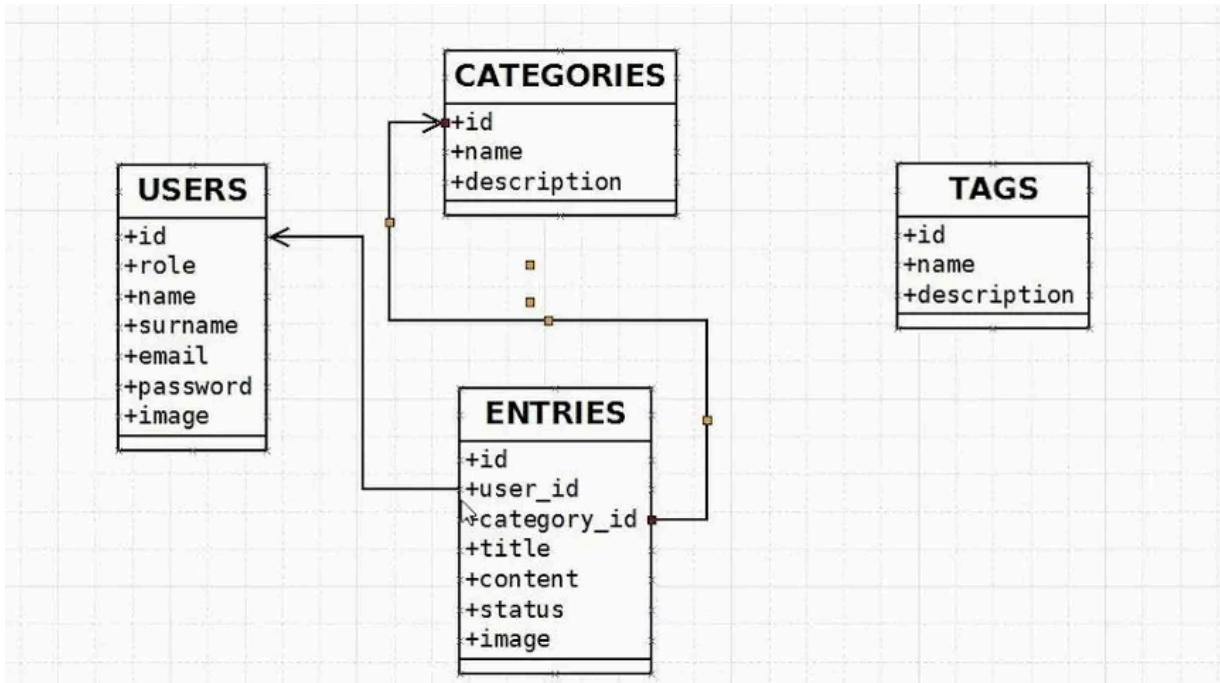


Figura 2.2.0.1 Imagen ilustrativa a un diagrama de base de datos..

2.2.1 Tipos de Bases de Datos

Existen distintos tipos de bases de datos, todo tiene relación para el funcionamiento que se le dé, para ello, se explicarán a continuación:

- **Bases de datos relacionales:** Los elementos de una base de datos relacional se organizan como un conjunto de tablas con columnas y filas, la tecnología de bases de datos relacionales proporciona la forma más eficiente y flexible de acceder a información estructurada.
- **Bases de datos orientadas a objetos:** La información de una base de datos orientada a objetos se representa en forma de objetos, como en la programación orientada a objetos.
- **Bases de datos distribuidas:** Consta de dos o más archivos que se encuentran en sitios diferentes, la base de datos puede almacenarse en varios ordenadores, ubicarse en la misma ubicación física o repartirse en diferentes redes. Una base de datos distribuida consta de dos o más archivos que se encuentran en sitios diferentes.

- Almacenes de datos: Un repositorio central de datos, un data warehouse es un tipo de base de datos diseñado específicamente para consultas y análisis rápidos.
- Bases de datos NoSQL: Permite almacenar y manipular datos no estructurados y semiestructurados (a diferencia de una base de datos relacional, que define cómo se deben componer todos los datos insertados en la base de datos), las bases de datos NoSQL se hicieron populares a medida que las aplicaciones web se volvían más comunes y complejas.
- Bases de datos de documentos/JSON: Diseñadas para almacenar, recuperar y gestionar información orientada a los documentos, las bases de datos de documentos son una forma moderna de almacenar los datos en formato JSON en lugar de hacerlo en filas y columnas.

2.2.2 Gestores de bases de datos

Una base de datos requiere un programa de software de bases de datos completo, conocido como sistema de gestión de bases de datos (DBMS), un DBMS sirve como interfaz entre la base de datos y sus programas o usuarios finales, lo que permite a los usuarios recuperar, actualizar y gestionar cómo se organiza y se optimiza la información, un DBMS también facilita la supervisión y el control de las bases de datos, lo que permite una variedad de operaciones administrativas como la supervisión del rendimiento, el ajuste, la copia de seguridad y la recuperación.

Algunos ejemplos de software de bases de datos o DBMS populares incluyen MySQL, Microsoft Access, Microsoft SQL Server, FileMaker Pro, Oracle Database y dBASE.

2.2.3 Lenguajes de programación para Bases de Datos

Algunos de los lenguajes más usados por los administradores de base de datos son los siguientes: [4]

- Python: Este lenguaje tiene desde gestión de sistemas hasta las herramientas web, las funcionalidades de Python son como realizar análisis, visualización y almacenamiento de datos.
- SQL: El lenguaje de consulta estructurado (SQL) es el lenguaje de programación estándar de facto utilizado por muchos de los servidores de bases de datos, SQL es robusto, escalable, mezcla de expresiones, consultas y declaraciones para producir una gran cantidad de métodos para acceder y extraer conjunto de datos pequeños y grandes.
- C#: La suite C de lenguajes de programación proporciona flexibilidad y eficiencia al desarrollar aplicaciones que se ejecutan de forma nativa en el hardware del sistema.
- R: Es un lenguaje cuyas fortalezas se basan en la computación estadística y se usa ampliamente en minería de datos, software estadístico y análisis de datos.
- PHP: PHP también tiene fuertes vínculos con las bases de datos a través del uso de scripts del lado del servidor para impulsar los sitios web que utilizan la conectividad de la base de datos para almacenar, escribir y recuperar registros de manera rápida y eficiente

2.3 Lenguajes de programación

En informática, se conoce como lenguaje de programación a un programa destinado a la construcción de otros programas informáticos, su nombre se debe a que comprende un lenguaje formal que está diseñado para organizar algoritmos y procesos lógicos que serán luego llevados a cabo por un ordenador o sistema informático, permitiendo controlar así su comportamiento físico, lógico y su comunicación con el usuario humano. Dicho lenguaje está

compuesto por símbolos y reglas sintácticas y semánticas, expresadas en forma de instrucciones y relaciones lógicas, mediante las cuales se construye el código fuente de una aplicación o pieza de software determinada. [5]

2.3.1 JavaScript

JavaScript es un lenguaje de programación que los desarrolladores utilizan para hacer páginas web interactivas, desde actualizar fuentes de redes sociales a mostrar animaciones y mapas interactivos, las funciones de JavaScript pueden mejorar la experiencia del usuario de un sitio web, como lenguaje de scripting del lado del servidor, se trata de una de las principales tecnologías de la World Wide Web. JavaScript funciona mediante la traducción de sintaxis similar a la del inglés a código de máquina, que posteriormente el sistema operativo se encarga de ejecutar. JavaScript se clasifica principalmente como un lenguaje de scripting o interpretado. El código JavaScript es interpretado, es decir, directamente traducido a código de lenguaje de máquina subyacente mediante un motor de JavaScript, en el caso de otros lenguajes de programación, un compilador se encarga de compilar todo el código en código de máquina en un paso diferente, en consecuencia, todos los lenguajes de scripts son lenguajes de programación, pero no todos los lenguajes de programación son lenguajes de scripts. [6]

Los beneficios que ofrece JavaScript son:

- Facilidad de aprender y utilizar.
- Obtener independencia de plataformas.
- Reducir la carga del servidor.
- Mejorar la interfaz de usuario.
- Admitir simultaneidad.

2.3.2 PHP

PHP es un lenguaje de programación destinado a desarrollar aplicaciones para la web y crear páginas web, favoreciendo la conexión entre los servidores y la interfaz de usuario, generalmente es definido como un lenguaje del lado del servidor, esto significa que se aplica en la programación que tiene lugar en el servidor web responsable de ejecutar la aplicación o, más a menudo, en un sitio web, este trabajo previo permite cargar los elementos de una página antes de mostrarlos al usuario que accede a un sitio web. [7]

Algunas de las ventajas de usar PHP son las siguientes:

- Aprendizaje intuitivo simplificado.
- Código abierto.
- Admite una gran cantidad de datos.
- Compatibilidad con las principales bases de datos.

2.4 Herramientas de desarrollo

Las herramientas de desarrollo de software son aplicaciones que se emplean en el diseño de programas nuevos y la optimización de programas existentes. Estos instrumentos optimizan el software a través de la edición, la administración, el soporte y la depuración. Debido al gran número de herramientas que existen y lo útil de sus funciones, es importante que quienes trabajan en desarrollo de software conozcan las más importantes para facilitar su trabajo. [8]

2.4.1 Visual Studio Code

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft, diseñado para ser ligero, rápido y altamente personalizable, a diferencia de otros editores de código, como los IDE completos (Entornos de Desarrollo Integrados), VS Code se enfoca en ofrecer una experiencia ágil para la escritura y edición de código, sin sacrificar características avanzadas,

tiene la capacidad de ser extendido con una gran cantidad de extensiones y herramientas, adaptándose a las necesidades específicas de cada desarrollador. Entre sus características clave, se incluyen la autocompletado inteligente mediante IntelliSense, la depuración integrada, y el control de versiones con Git, lo que convierte a VS Code en una opción versátil tanto para desarrolladores individuales como para equipos. [9]

2.4.2 (IDEs, Frameworks, controladores de versiones, etc)

En el desarrollo del sistema de Sistema Integral de Gestión para Microempresas de Abarrotes, se emplearon diversos métodos y enfoques para la creación del software, con el fin de garantizar su funcionalidad, eficiencia y facilidad de uso.

Framework: Bootstrap es un framework CSS utilizado en aplicaciones front-end (véase Figura 2.4.2.1), es decir, en la pantalla de interfaz con el usuario, para desarrollar aplicaciones que se adaptan a cualquier dispositivo, el framework combina CSS (véase Figura 2.4.2.2) y JavaScript (véase Figura 2.4.2.3) para estilizar los elementos de una página HTML (véase Figura 2.4.2.4), permite mucho más que, simplemente, cambiar el color de los botones y los enlaces, esta es una herramienta que proporciona interactividad en la página, por lo que ofrece una serie de componentes que facilitan la comunicación con el usuario, como menús de navegación, controles de página, barras de progreso y más, además de todas las características que ofrece el framework, su principal objetivo es permitir la construcción de sitios web responsive para dispositivos móviles, esto significa que las páginas están diseñadas para funcionar en desktop, tablets y smartphones, de una manera muy simple y organizada. [10]

Bootstrap está constituido por una serie de archivos CSS y JavaScript responsables de asignar características específicas a los elementos de la página, hay un archivo principal llamado bootstrap.css, que contiene una

definición para todos los estilos utilizados. Básicamente, la estructura del framework se compone de dos directorios:

- css: contiene los archivos necesarios para la estilización de los elementos y una alternativa al tema original.
- js: contiene la parte posterior del archivo bootstrap.js (original y minificado), responsable de la ejecución de aplicaciones de estilo que requieren manipulación interactiva.

Para asignarle una característica a un elemento, simplemente debemos informar la clase correspondiente en la propiedad “class” del elemento que será estilizado.

Para la gestión de bases de datos tenemos los siguientes:

- phpMyAdmin: PhpMyAdmin (véase Figura 2.4.2.5) es una aplicación web que sirve para administrar bases de datos MySQL de forma sencilla y con una interfaz amistosa, se trata de un software muy popular basado en PHP, la ventaja de usar una aplicación web es que nos permite conectarnos con servidores remotos, a los cuales no siempre se puede acceder usando programas de interfaz gráfica. [11]
- MySQL Workbench: MySQL WorkBench (véase Figura 2.4.2.6) es una herramienta visual ideal para modelar, diseñar y administrar bases de datos MySQL y también para el uso de código MySQL, se trata de una herramienta gráfica que al igual que el sistema de gestión de bases de datos MySQL fue creado por la compañía Oracle. [12]
- XAMPP: XAMPP (véase Figura 2.4.2.7) es el servidor más utilizado y popular entre los desarrolladores web y programadores, porque permite instalar y configurar el entorno de un servidor local de manera sencilla, esto es útil para el desarrollo, pruebas y depuración de aplicaciones web antes de

desplegarlas en un servidor de producción en línea. XAMPP (Apache + MariaDB + PHP + Perl) es un acrónimo que representa los componentes principales del software. [13]

- X: El sistema operativo (Windows, Linux o macOS).
- A: Apache, que es un servidor web de desarrollo local.
- M: MySQL o MariaDB, que es un sistema de gestión de bases de datos relacional.
- P: PHP, que es un lenguaje de programación para el desarrollo web.
- P: Perl o Python, que es un lenguaje de programación de alto nivel.



Figura 2.4.2.1 Bootstrap.



Figura 2.4.2.2 CSS 3.



Figura 2.4.2.3 JavaScript.



Figura 2.4.2.4 HTML 5.



Figura 2.4.2.5 phpMyAdmin.



Figura 2.4.2.6 MySQL Workbench.



Figura 2.4.2.7 XAMPP.

CAPÍTULO 3: DESARROLLO

3.1 Registrar gerente e iniciar sesión (login)

3.1.1 Descripción del caso de uso.

La descripción del caso de uso es el acceso a la aplicación web, este caso se tiene mucha amplitud debido a que se tomarán los subtemas desde el registro del gerente, donde el gerente puede registrar a sus empleados y después los empleados pueden realizar su inicio de sesión:

- Registro del gerente: El gerente, al principio de la aplicación tendrá en el header el acceso al registro en el botón de “Registrarme” o “¡Registrarme ya!” debajo del nombre SIGMA (véase Figura 3.1.1.1), al presionar uno de estos botones, se redirigirá a un formulario para el registro del gerente (véase Figura 3.1.1.2), lo datos a ingresar son:
 - Nombre del gerente.
 - Nombre del establecimiento.
 - Número de celular del gerente.
 - Usuario para el gerente.
 - Contraseña para su inicio de sesión.
 - Repetir su contraseña para verificar que coincidan.

Una vez concluya con esto, el gerente ya estará registrado en la plataforma y podrá iniciar sesión e incluso, al finalizar el formulario, se le redirige al inicio de sesión.

- Registro de empleados: Una vez el gerente tenga acceso a la plataforma con su inicio de sesión, tendrá variedad de opciones para realizar, uno de ellos se encuentra en el menú desplegable “Mis Empleados”, esta contiene dos más, uno de ellos es “Aregar” (véase Figura 3.1.1.3), en referencia a agregar empleados, una vez acceda a este submenú, se le presentará un

formulario para el registro del empleado (véase Figura 3.1.1.4), los campos a llenar son:

- Nombre del empleado.
 - Número de celular del empleado.
 - Seleccionar su turno (Matutino, Vespertino o Nocturno).
 - Rol:
 - Encargado: Este rol tendrá privilegios, los cuales serán los siguientes:
 - Podrá agregar tipos de importes así como modificar los precios correspondientes.
 - Podrá agregar frutas y verduras así como modificar los precios de frutas y verduras correspondientes.
 - Podrá agregar conceptos nuevos de gastos.
 - Cajera: Este rol es más limitado, dado que no tiene los privilegios que un encargado.
 - Usuario para el empleado.
 - Contraseña para su inicio de sesión.
 - Repetir su contraseña para verificar que coincidan.
- Inicio de sesión para ambos:

Con lo anterior mencionado, el gerente y el empleado podrán iniciar sesión con su usuario y contraseña correspondiente, simplemente se accede a la página mediante el botón “¡Iniciar sesión!” y se ingresan los datos correspondientes (véase Figura 3.1.1.5).



Figura 3.1.1.1 Muestra los botones para que un gerente pueda registrarse.

A screenshot of the "Registro de Gerente" (Manager Registration) form. The form is titled "Registro de Gerente" in bold black text at the top center. It contains six input fields, each with a small icon to its left: "Nombre" (Name) with a person icon, "Establecimiento" (Establishment) with a storefront icon, "Celular" (Cellular) with a phone icon, "Usuario" (User) with an envelope icon, "Contraseña" (Password) with a lock icon, and "Repetir Contraseña" (Repeat Password) with a lock icon. Below these fields is a green button labeled "Registrar" (Register) with a small icon. The background of the form has a light gray gradient.

Figura 3.1.1.2 Formulario de registro para gerentes.

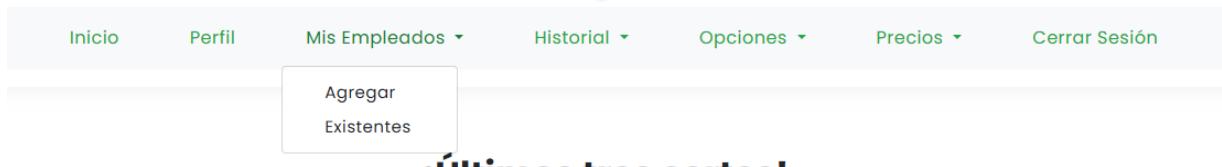


Figura 3.1.1.3 Menú desplegable para el gerente.

A screenshot of an employee registration form titled 'Registro de Empleados'. The form consists of seven input fields: 'Nombre del empleado' (Employee Name) with a person icon, 'Número de celular' (Cell Phone Number) with a phone icon, 'Turno' (Shift) with a clock icon and placeholder 'Selecciona un turno', 'Rol' (Role) with a person icon and placeholder 'Selecciona un rol', 'Usuario' (User) with a user icon, 'Contraseña' (Password) with a lock icon, and 'Repetir contraseña' (Repeat Password) with a lock icon. Below the fields is a green 'Registrar' (Register) button with a checkmark icon.

Figura 3.1.1.4 Formulario para registrar empleados.

[Inicio](#) [Registrarme](#) [Iniciar Sesión](#)

 **Iniciar Sesión**

	Nombre de usuario	
	Contraseña	
 Ingresar		

Figura 3.1.1.5 Formulario para iniciar sesión.

SIGMA



The image shows a mobile application interface titled "SIGMA". A central modal window is displayed with the title "Registro de Gerente". Inside the modal, there are six input fields: "Nombre" (Name), "Establecimiento" (Establishment), "Celular" (Cellular), "Usuario" (User), "Contraseña" (Password), and "Repetir Contraseña" (Repeat Password). Below these fields is a green "Registrar" (Register) button.

Figura 3.1.1.6 Presentación de registro de gerente en dispositivo móvil..

SIGMA



The image shows a mobile application interface titled "SIGMA". A central modal window is displayed with the title "Iniciar Sesión" (Login). It contains two input fields: "Nombre de usuario" (User name) and "Contraseña" (Password). Below the password field is a "Copiar" (Copy) button. At the bottom is a green "Ingresar" (Enter) button. The background of the main screen shows three small icons: a magnifying glass, a person, and a gear. At the bottom, there is a copyright notice: "© 2025 Sistema Integral de Gestión para Microempresas de Abarrotes | Diseñado por César Adrián Gabriel Cruz".

Figura 3.1.1.7 Presentación de inicio de sesión en dispositivo móvil..

3.1.2 Modelo de datos

El modelo de datos para este caso de uso, son las necesarias para su funcionamiento, las cuales son únicamente dos tablas (véase Figura 3.1.2.1):

- admin: Es la tabla donde se hace el registro del gerente con sus campos correspondientes.
- empleados: Es la tabla donde se hace el registro del empleado.

En estas tablas se hace la verificación del usuario y su relación con la contraseña ingresada, algo importante es que los nombres de usuario no se pueden repetir, por lo tanto son únicos en combinación de ambas tablas, es decir, si juntamos el usuario de las tablas admin y empleados, los nombres de usuario no se repiten.

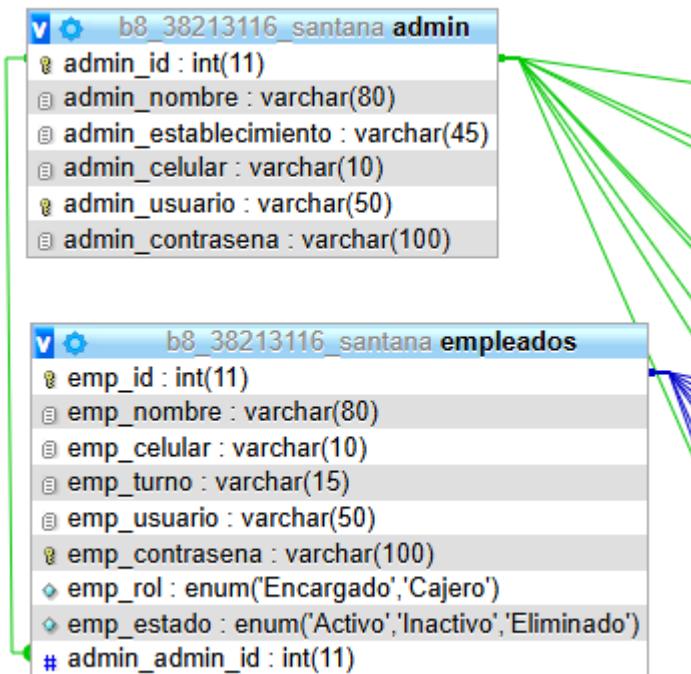


Figura 3.1.2.1 Modelo de datos.

3.1.3 Diseño de la interfaz

El diseño de la interfaz busco ser lo más simple posible, para que esto sea fácil de entender y poder llenar los campos correctamente, hablaremos de ambos casos, para el registro del gerente y inicio de sesión:

- Registro del gerente (véase Figura 3.1.3.1): El formulario de registro de gerente tiene las siguientes características:
 - Tenemos un formulario resaltado por sombras alrededor de este.
 - Cuenta con los campos necesarios, mínimos y básicos para la información de dicho registro.
 - Al lado de los campos existe un ícono ilustrativo a su llenado, esto para apoyo visual a quien esté llenando la información.
 - El placeholder funge como guía para lo que hay que llenar en cada campo, este placeholder tiene la propiedad de no desaparecer mientras el usuario está escribiendo (véase Figura 3.1.3.2).
 - Sí al momento de enviar el formulario, los campos de contraseña no coinciden, este recibirá una alerta y no se hará dicho registro (véase Figura 3.1.3.3).
 - Sí al momento de enviar el formulario, el nombre de usuario ya ha sido registrado, este recibirá una alerta y no se hará dicho registro (véase Figura 3.1.3.4).
- Inicio de sesión (véase Figura 3.1.3.5): El formulario de inicio de sesión tiene las siguientes características:
 - Al igual que el formulario de registro del gerente, el placeholder no desaparece mientras el usuario escribe (véase Figura 3.1.3.6).

- El campo de contraseña, tiene la propiedad de ver su contraseña, para en caso de duda si es correcta o no, pueda verificarlo (véase Figura 3.1.3.7).
- En caso que el usuario haya ingresado datos erróneos, aparecerá una pequeña alerta debajo del formulario para su información (véase Figura 3.1.3.8).

Registro de Gerente

	Nombre
	Establecimiento
	Celular
	Usuario
	Contraseña
	Repetir Contraseña
 Registrar	

Figura 3.1.3.1 Formulario de registro del gerente.

Registro de Gerente

	Nombre César Adrian Gabriel Cruz
	Establecimiento Santana Elia
	Celular 3173893408
	Usuario Cesar
	Contraseña
	Repetir Contraseña
<input type="button" value="Registrar"/>	

Figura 3.1.3.2 El placeholder no desaparece cuando el usuario a llenado el campo.

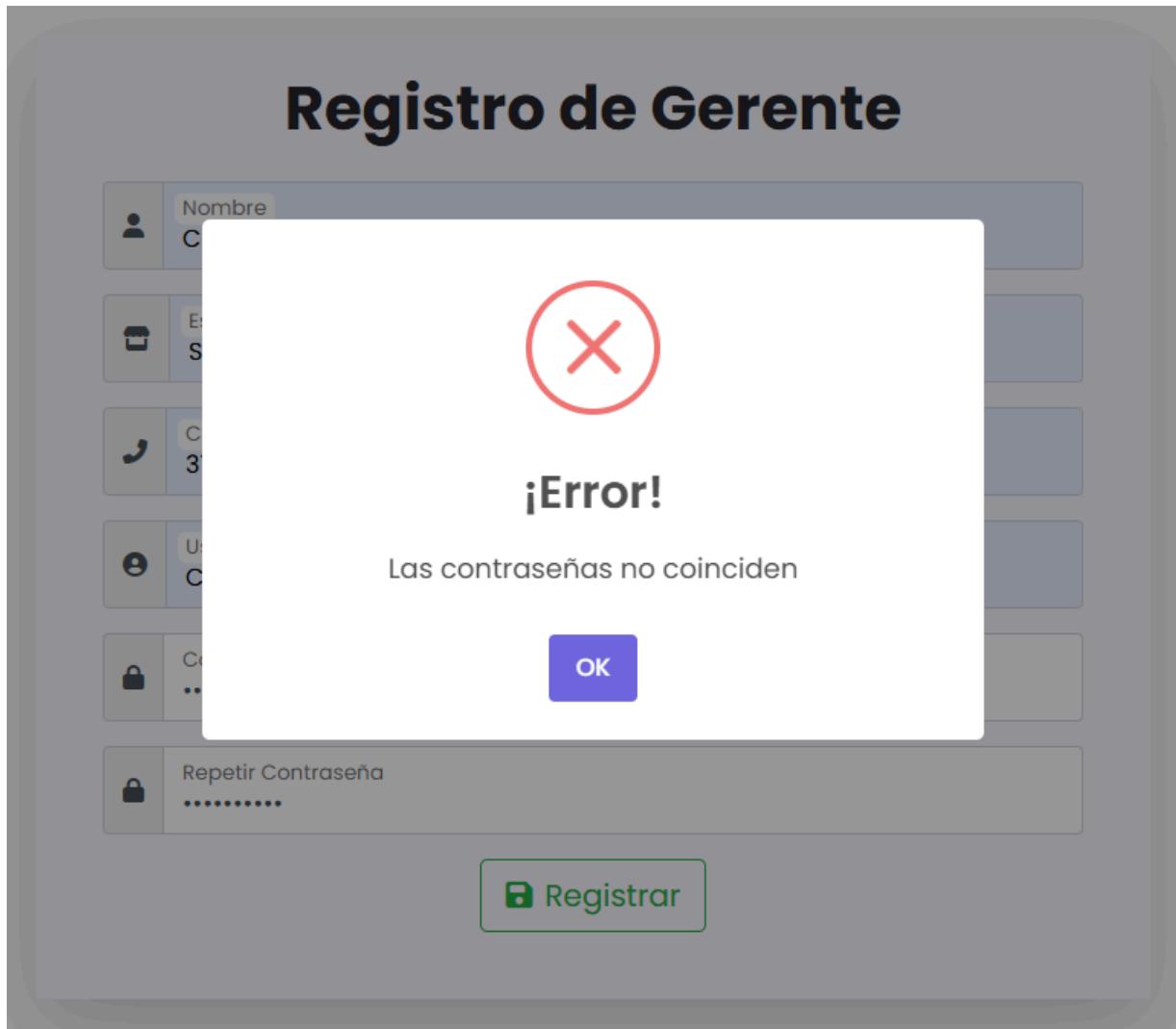


Figura 3.1.3.3 Alerta de contraseñas no coinciden.

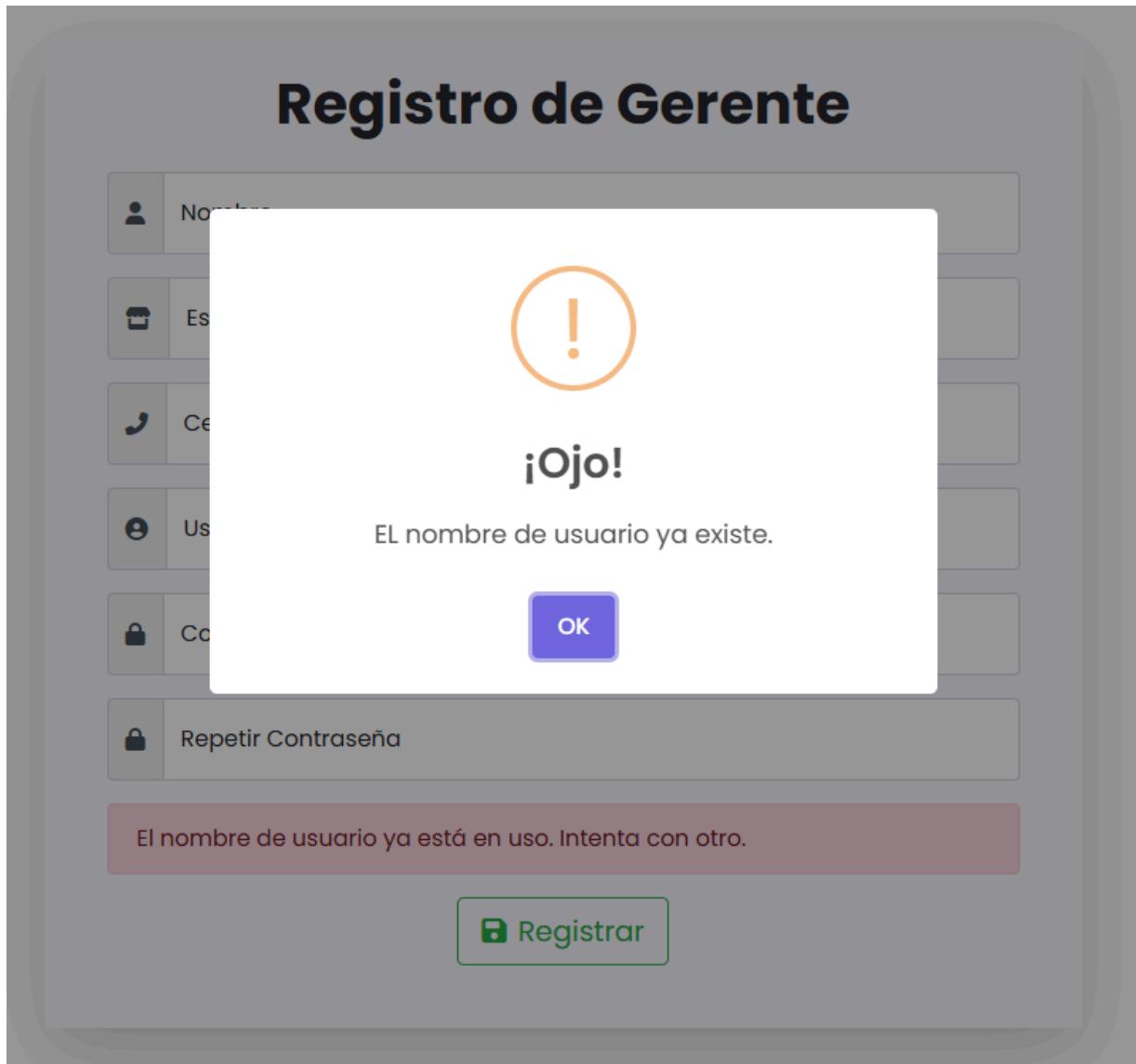


Figura 3.1.3.4 Alerta de usuario no disponible.

Iniciar Sesión

 Nombre de usuario

 Contraseña 

 Ingresar

Figura 3.1.3.5 Formulario para inicio de sesión.

Iniciar Sesión

 Nombre de usuario
Cesar

 Contraseña
..... 

 Ingresar

Figura 3.1.3.6 El placeholder no desaparece.



Figura 3.1.3.7 El usuario puede ver su contraseña ingresada.

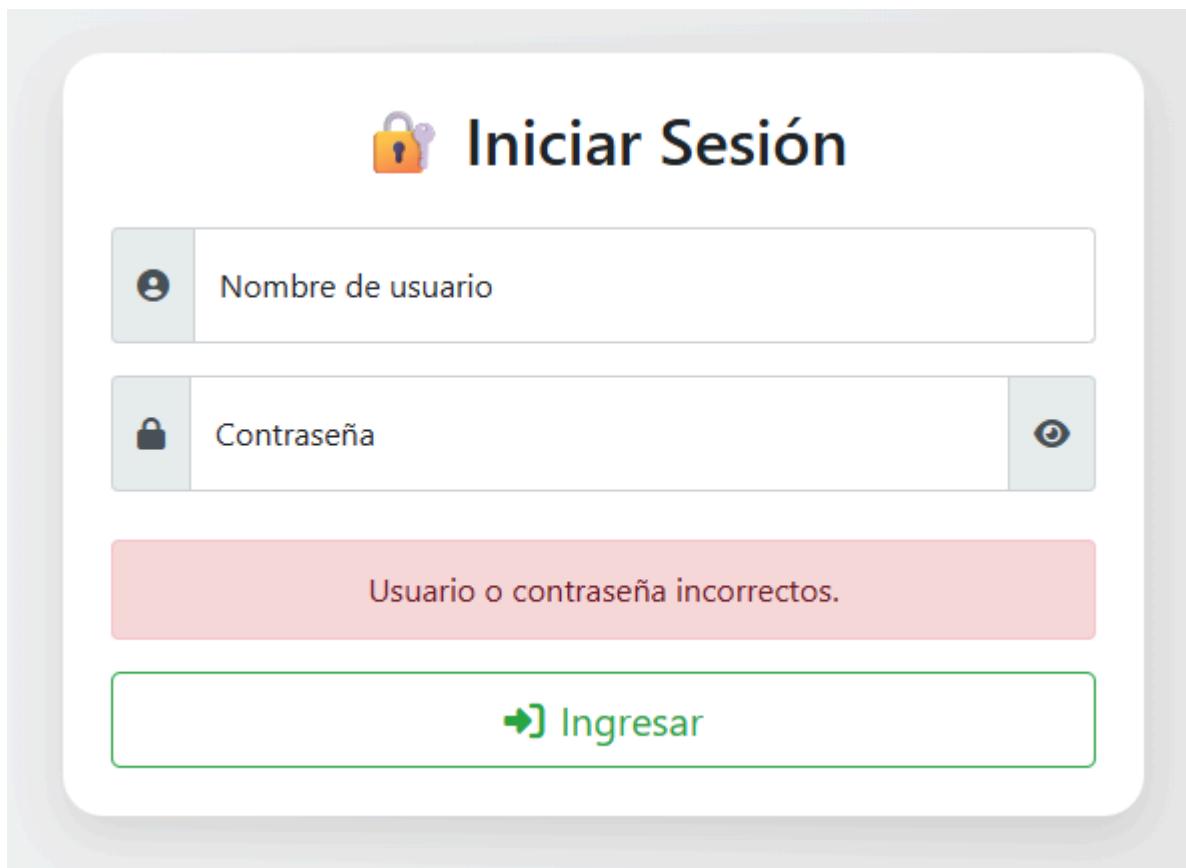


Figura 3.1.3.8 Alerta de usuario o contraseña incorrecta.

3.1.4 Fragmentos de código

Para este apartado, de la misma manera que las anteriores, dividiremos en dos, una con explicaciones sobre el registro del gerente y otra parte del inicio de sesión, se abarcaran partes de la conexión con el backend y conexión con la base de datos.

- Registro de gerente: Para el registro del gerente, como ya se vió en el apartado del diseño de la interfaz, el formulario conecta con el backend para poder registrarse en la base de datos.
 - Diseño de la interfaz: Tenemos el archivo llamado `admin_register.php` (véase Figura 3.1.4.1), en la interfaz tenemos contenedor que junta todo el formulario, dentro de este están las propiedades de las sombras, el placeholder, los campos son obligatorios con la propiedad `required`, además, de contener las funciones como la validación de contraseñas y el mostrar las alertas.
 - Backend: Tenemos el archivo llamado `admin_register_submit.php` (véase Figura 3.1.4.2), en el backend tenemos código que hace la función del registro del gerente en la base de datos, primeramente obtenemos mediante método `post`, la información, la cual se guarda en variables, después de esto tenemos dos pasos de verificación:
 - Verificar que estas variables tengan información, en caso de que alguna no tenga información se redirige a la interfaz gráfica y avisar al usuario que faltan campos por competir, pero, si todas las variables tienen información, continua en la siguiente verificación.
 - Se verifica si ya existe un usuario registrado con el mismo nombre de usuario que ingresó el registrante, en caso de que exista, se redirige una alerta a la interfaz gráfica de usuario ya registrado al registrarte, en caso que el usuario aún no ha sido registrado, se continúa el proceso de registro.

Una vez finalizado el proceso de verificación se cifra la contraseña y se procede a guardar la información del registrante en la base de datos, dando por concluido el registro.

- Inicio de sesión: Para el inicio de sesión, de igual manera tenemos dos archivos, uno que maneja la interfaz gráfica y otro maneja la conexión con la base de datos, o el backend, a continuación serán descritos:
 - Diseño de la interfaz: Tenemos el archivo llamado login.php (véase Figura 3.1.4.3) en el cual tenemos un formulario simple con los campos de usuario y contraseña, estos se envían mediante método post al submit del mismo, además, mantenemos la función de togglePasswordVisibility() que permite ver las contraseñas ingresadas en el campo de contraseña. Mucha información del mismo se describió en el apartado de diseño de la interfaz.
 - Backend: Tenemos el archivo llamado login_submit.php (véase Figura 3.1.4.4) en el cual primero recibimos la información post y las guardamos en variables correspondientes, luego, pasamos a verificar que las variables contengan información, en caso de no tener información en alguna de las variables, se redirige al diseño de la interfaz con el aviso de llenar todos los campos, pero si todas las variables tiene información sucederán varios aspectos:
 - Primero, hacemos una solicitud primeramente a la tabla admin, obteniendo todos los datos y comparando el usuario recibido con alguno que exista en la base de datos de la tabla admin.

- En caso de encontrar alguno en la tabla admin, ahora se compara la contraseña ingresada con la contraseña descifrada de la base de datos, de ser la misma, los campos obtenidos de la consulta a la tabla admin, pasan a ser variables de sesión y este redirige al menú principal del gerente, y finaliza el proceso.
- En caso de no encontrar ningún usuario o contraseña igual en la tabla admin, seguimos en la siguiente consulta.
- Ahora buscamos en la tabla de empleados, obtenemos todos los datos donde coinciden el usuario ingresado con el usuario de empleados.
- En caso de encontrar algún registro idéntico en la tabla de empleados, se procede a comparar las contraseñas, la ingresada por el usuario y la descifrada obtenida de la consulta, de ser la misma contraseña, los campos de la consulta pasan a ser variables de sesión, y se redirigen al menú principal del empleado y finaliza el proceso.
- En caso de no encontrar ningún usuario o contraseña igual en la tabla empleados, se envía una advertencia a la interfaz de usuario del login mencionando que usuario o contraseña son incorrectos y finaliza el proceso.

```
admin_register.php
1 <?php
2 // admn_register.php >
3 <div> container </div> <div> justify-content-center </div> <div> col-md-8 </div> <div> card shadow-lg rounded-4 border-0 </div> <div> card-body p-4 bg-light </div> <div> card-body </div> <form> <div> input-group mb-3 </div>
4 <!-- Formulario de registro de administrador -->
5 <div> class="row justify-content-center" </div>
6 <div> class="col-12" </div>
7 <div> class="card shadow-lg rounded-4 border-0" </div>
8 <div> class="card-body p-4 bg-light" </div>
9 <div> class="text-center fw-bold mb-1" > Registro de Gerente </div>
10 <div> class="form-control" </div>
11 <form action="admin_register_submit.php" method="post" onsubmit="return validarContrasenas();">
12
13 <!-- Campo Nombre -->
14 <div> class="input-group mb-3" </div>
15 <span> class="input-group-text" <div> class="fas fa-user" </div> </span>
16 <div> class="form-control" id="nombre" name="nombre" placeholder="Ingresa tu nombre" value="php echo isset($nombre) ? $nombre : '' ; ?" required>
17 <input type="text" class="form-control" id="nombre" name="nombre" placeholder="Ingresa tu nombre" value="php echo isset($nombre) ? $nombre : '' ; ?" required>
18 <label for="nombre">Nombre</label>
19 <br>
20 </div>
21
22 <!-- Campo Establecimiento -->
23 <div> class="input-group mb-3" </div>
24 <span> class="input-group-text" <div> class="fas fa-store" </div> </span>
25 <div> class="form-control" id="establecimiento" name="establecimiento" placeholder="Ingresa el nombre del establecimiento" value="php echo isset($establecimiento) ? $establecimiento : '' ; ?" required>
26 <input type="text" class="form-control" id="establecimiento" name="establecimiento" placeholder="Ingresa el nombre del establecimiento" value="php echo isset($establecimiento) ? $establecimiento : '' ; ?" required>
27 <label for="establecimiento">Establecimiento</label>
28 <br>
29 </div>
30
31 <!-- Campo Celular -->
32 <div> class="input-group mb-3" </div>
33 <span> class="input-group-text" <div> class="fas fa-phone-alt" </div> </span>
34 <div> class="form-control" id="celular" name="celular" placeholder="Ingresa tu numero de celular" value="php echo isset($celular) ? $celular : '' ; ?" required>
35 <input type="tel" class="form-control" id="celular" name="celular" placeholder="Ingresa tu numero de celular" value="php echo isset($celular) ? $celular : '' ; ?" required>
36 <label for="celular">Celular</label>
37 <br>
38 </div>
39
40 <!-- Campo Usuario -->
41 <div> class="input-group mb-3" </div>
42 <span> class="input-group-text" <div> class="fas fa-user-circle" </div> </span>
43 <div> class="form-control" id="usuario" name="usuario" placeholder="Ingresa un nombre de usuario" value="php echo isset($usuario) ? $usuario : '' ; ?" required>
44 <input type="text" class="form-control" id="usuario" name="usuario" placeholder="Ingresa un nombre de usuario" value="php echo isset($usuario) ? $usuario : '' ; ?" required>
45 <label for="usuario">Usuario</label>
46 <br>
47 </div>
48
49 <!-- Campo Contraseña -->
50 <div> class="input-group mb-3" </div>
51 <span> class="input-group-text" <div> class="fas fa-lock" </div> </span>
52 <div> class="form-control" id="contraseña" name="contraseña" placeholder="Contraseña" required>
53 <input type="password" class="form-control" id="contraseña" name="contraseña" placeholder="Contraseña" required>
54 <label for="contraseña">Contraseña</label>
55 <br>
56 </div>
57
58 <!-- Campo Repetir Contraseña -->
59 <div> class="input-group mb-3" </div>
60 <span> class="input-group-text" <div> class="fas fa-lock" </div> </span>
61 <div> class="form-control" id="repetir_contraseña" name="repetir_contraseña" placeholder="Repetir Contraseña" required>
62 <input type="password" class="form-control" id="repetir_contraseña" name="repetir_contraseña" placeholder="Repetir Contraseña" required>
63 <label for="repetir_contraseña">Repetir Contraseña</label>
64 <br>
```

Figura 3.1.4.1 Código del diseño de la interfaz del registro del gerente.

```

  admin_register_submit.php ×
CorteCaja > admin_register_submit.php > ...
1  <?php
2  // Incluir la conexión a la base de datos
3  include 'includes/conexion_db.php';
4
5  if ($_SERVER["REQUEST_METHOD"] == "POST") {
6
7      $nombre = $_POST['nombre'];
8      $establecimiento = $_POST['establecimiento'];
9      $celular = $_POST['celular'];
10     $usuario = $_POST['usuario'];
11     $contrasena = $_POST['contrasena'];
12
13     // Validar campos vacíos
14     if (empty($nombre) || empty($establecimiento) || empty($celular) || empty($usuario) || empty($contrasena)) {
15         header("Location: admin_register.php?error=empty_fields");
16         exit();
17     }
18
19     // Verificar si el usuario ya existe
20     $sql_check = "SELECT COUNT(*) FROM (
21         SELECT admin_usuario AS usuario FROM admin WHERE admin_usuario = :usuario
22         UNION ALL
23         SELECT emp_usuario FROM empleados WHERE emp_usuario = :usuario
24     ) AS total";
25
26     $stmt_check = $pdo->prepare($sql_check);
27     $stmt_check->execute([':usuario' => $usuario]);
28     $usuario_existente = $stmt_check->fetchColumn();
29
30     if ($usuario_existente > 0) {
31         header("Location: admin_register.php?error=usuario_existente");
32         exit();
33     }
34
35     // Hash de la contraseña
36     $contrasena_hash = password_hash($contrasena, PASSWORD_DEFAULT);
37
38     // Insertar nuevo administrador
39     $sql_insert = "INSERT INTO admin (admin_id, admin_nombre, admin_establecimiento, admin_celular, admin_usuario, admin_contrasena)
40     VALUES (null, :nombre, :establecimiento, :celular, :usuario, :contrasena)";
41     $stmt_insert = $pdo->prepare($sql_insert);
42
43     if ($stmt_insert->execute([
44         'nombre' => $nombre,
45         'establecimiento' => $establecimiento,
46         'celular' => $celular,
47         'usuario' => $usuario,
48         'contrasena' => $contrasena_hash
49     ])) {
50         header("Location: login.php");
51         exit();
52     } else {
53         echo "Error al registrar el administrador: " . $stmt_insert->errorInfo()[2];
54     }
55
56     // Cerrar conexiones
57     $stmt_check = null;
58     $stmt_insert = null;
59     $pdo = null;
60 }
61

```

Figura 3.1.4.2 Código del backend para registrar un gerente.

```

login.php
1 <?php
2 session_start();
3 $error_message = isset($_SESSION['error_message']) ? $_SESSION['error_message'] : "";
4 unset($_SESSION['error_message']);
5 ?
6
7 <?php include 'includes/header.php'; ?
8
9 > <style>...
10 </style>...
11
12 <div class="container mt-2">
13   <div class="row justify-content-center align-items-center py-5">
14     <div class="col-md-6">
15       <div class="card login-card p-4">
16         <h2 class="text-center mb-4"> Iniciar Sesión</h2>
17
18         <form action="login_submit.php" method="post" novalidate>
19
20           <!-- Usuario -->
21           <div class="input-group mb-3">
22             <span class="input-group-text"><i class="fas fa-user-circle"></i></span>
23             <div class="form-floating flex-grow-1">
24               <input type="text" class="form-control" id="usuario" name="usuario" placeholder="Usuario" autocomplete="username" required>
25               <label for="usuario">Nombre de usuario:</label>
26             </div>
27           </div>
28
29           <!-- Contraseña -->
30           <div class="input-group mb-4">
31             <span class="input-group-text"><i class="fas fa-lock"></i></span>
32             <div class="form-floating flex-grow-1">
33               <input type="password" class="form-control" id="contrasena" name="contrasena" placeholder="Contraseña" autocomplete="current-password" required>
34               <label for="contrasena">Contraseña:</label>
35             </div>
36             <span class="input-group-text toggle-password" onclick="togglePasswordVisibility()">
37               <i class="fas fa-eye" id="eyeIcon"></i>
38             </span>
39           </div>
40
41           <?php if (!empty($error_message)) : ?>
42             <div class="alert alert-danger text-center">
43               <?php echo $error_message; ?>
44             </div>
45           <?php endif; ?>
46
47           <div class="d-grid">
48             <button type="submit" class="btn btn-outline-success btn-lg">
49               <i class="fas fa-sign-in-alt me-2"></i>Ingresar
50             </button>
51           </div>
52         </form>
53       </div>
54     </div>
55   </div>
56 </div>
57
58 </script>
59
60   function togglePasswordVisibility() {
61     const passwordInput = document.getElementById('contrasena');
62     const eyeIcon = document.getElementById('eyeIcon');
63
64     if (passwordInput.type === 'password') {
65       passwordInput.type = 'text';
66       eyeIcon.classList.remove('fa-eye');
67       eyeIcon.classList.add('fa-eye-slash');
68     } else {
69       passwordInput.type = 'password';
70       eyeIcon.classList.remove('fa-eye-slash');
71       eyeIcon.classList.add('fa-eye');
72     }
73   }

```

Figura 3.1.4.3 Código del diseño de la interfaz del login.

```

login_submit.php
ConeCaja > login_submit.php > ...
1  <?php
2  session_start();
3  include 'includes/conexion_db.php';
4
5  $error_message = "";
6
7  if ($_SERVER["REQUEST_METHOD"] == "POST") {
8      $usuario = $_POST['usuario'];
9      $contrasena = $_POST['contrasena'];
10
11     if (empty($usuario) || empty($contrasena)) {
12         $_SESSION['error_message'] = "Todos los campos son requeridos.";
13         header("Location: login.php");
14         exit();
15     }
16
17    try {
18        // 1. Buscar en la tabla de administradores
19        $stmt = $pdo->prepare("SELECT admin_id, admin_nombre, admin_establecimiento, admin_celular, admin_usuario, admin_contrasena FROM admin WHERE admin_usuario = :usuario");
20        $stmt->execute(['usuario' => $usuario]);
21        $admin = $stmt->fetch(PDO::FETCH_ASSOC);
22
23        if ($admin && password_verify($contrasena, $admin['admin_contrasena'])) {
24            $_SESSION['admin_id'] = $admin['admin_id'];
25            $_SESSION['admin_nombre'] = $admin['admin_nombre'];
26            $_SESSION['admin_establecimiento'] = $admin['admin_establecimiento'];
27            $_SESSION['admin_celular'] = $admin['admin_celular'];
28            $_SESSION['admin_usuario'] = $admin['admin_usuario'];
29            header('Location: admin_index.php');
30            exit();
31        }
32
33        // 2. Buscar en la tabla de empleados
34        $stmt = $pdo->prepare("SELECT emp_id, emp_nombre, emp_turno, emp_celular, emp_usuario, emp_contraseña, emp_rol, emp_estado, admin_admin_id FROM empleados WHERE emp_usuario = :usuario");
35        $stmt->execute(['usuario' => $usuario]);
36        $empleado = $stmt->fetch(PDO::FETCH_ASSOC);
37
38        if ($empleado) {
39            if ($empleado['emp_estado'] != 'Activo') {
40                $_SESSION['error_message'] = "Usuario inactivo o eliminado.";
41                header("Location: login.php");
42                exit();
43            }
44
45            if (password_verify($contrasena, $empleado['emp_contraseña'])) {
46                $_SESSION['emp_id'] = $empleado['emp_id'];
47                $_SESSION['emp_nombre'] = $empleado['emp_nombre'];
48                $_SESSION['emp_turno'] = $empleado['emp_turno'];
49                $_SESSION['emp_celular'] = $empleado['emp_celular'];
50                $_SESSION['emp_usuario'] = $empleado['emp_usuario'];
51                $_SESSION['emp_rol'] = $empleado['emp_rol'];
52                $_SESSION['emp_estado'] = $empleado['emp_estado'];
53                $_SESSION['admin_admin_id'] = $empleado['admin_admin_id'];
54                header('Location: employee_index.php');
55                exit();
56            }
57        }
58
59        // Si llega aqui, usuario no encontrado o contraseña incorrecta
60        $_SESSION['error_message'] = "Usuario o contraseña incorrectos.";
61        header("Location: login.php");
62        exit();
63
64    } catch (Exception $e) {
65        $_SESSION['error_message'] = "Error del sistema: " . $e->getMessage();
66        header("Location: login.php");
67        exit();
68    }
69 }

```

Figura 3.1.4.4 Código del backend para logeo de un usuario.

3.2 Registrar, inhabilitar y/o eliminar empleados.

3.2.1 Descripción del caso de uso

La descripción del caso de uso podemos decir que son tres acciones, las cuales se explican a continuación:

- Registrar empleados: Una vez el gerente haya accedido al sistema con su usuario, necesitará registrar a sus empleados al sistema, y esto se realiza dentro del apartado del gerente

únicamente, los datos solicitados para el registro de un empleado son (véase Figura 3.2.1.1):

- Nombre del empleado.
- Número de celular.
- Selección de turno (Matutino, Vespertino o Nocturno).
- Selección de rol (Cajero o Encargado).
- Usuario.
- Contraseña.
- Repetición de contraseña.

Los empleados son parte fundamental del sistema ya que ellos son quienes registran los gastos, pagos con tarjeta, importes, consultas de precios, productos faltantes y realización del corte en cada turno, Además, existen dos tipos de roles para empleados los cuales son cajero y encargado, la única diferencia es que el encargado tiene privilegios de editar precios de frutas y verduras así como de importes y agregar nuevos tipos de gastos e importes, el rol de encargado funge como representación al gerente, quien como su rol dice, está encargado de la asignación de los privilegios para llevar una administración adecuada y correcta en la venta y registros de los productos anteriormente mencionados.

- Inhabilitar empleados: Esta función está para cuando un empleado está de vacaciones, ha hecho uso indebido del sistema o de las instalaciones del gerente (como una advertencia de despido por acciones que infrinjan el reglamento del gerente), lo que realiza la función es inhabilitar el acceso al usuario temporalmente sin la necesidad de eliminarlo (véase Figura 3.2.1.2).

- Eliminar empleados: Al momento de despedir a un empleado, se deberá eliminar su usuario para evitar el acceso al sistema, se puede también habilitar, pero la función de eliminar, no borra el registro del usuario dado que este es un registro que está en otras tablas de la base de datos, lo que realiza al eliminar un usuario es dejar de estar disponible para el gerente, el gerente no podrá consultarlos directamente, pero el usuario si aparecerá en el historial de corte realizados así como de gastos o pagos con terminal (véase Figura 3.1.1.2)

The screenshot shows a user interface for employee registration. At the top, there is a navigation bar with links: Inicio, Perfil, Mis Empleados (highlighted in green), Historial, Opciones, Precios, and Cerrar Sesión. Below the navigation bar is a search bar with a magnifying glass icon. The main content area has a title 'Registro de Empleados'. The form consists of seven input fields, each with a small icon to its left:

- Nombre del empleado (User icon)
- Número de celular (Phone icon)
- Turno (Clock icon): Seleccióna un turno
- Rol (User icon): Seleccióna un rol
- Usuario (User icon)
- Contraseña (Lock icon)
- Repetir contraseña (Lock icon)

At the bottom of the form is a green button labeled 'Registrar' with a small icon.

Figura 3.2.1.1 Formulario para registrar empleados.

Opciones para Cesar



Figura 3.2.1.2 Botones para eliminar o inhabilitar un usuario.

A screenshot of a mobile device screen displaying a registration form titled 'Registro de Empleados'. The form consists of several input fields with accompanying icons: 'Nombre del empleado' (person icon), 'Número de celular' (phone icon), 'Turno' (clock icon) with the sub-instruction 'Selecciona un turno', 'Rol' (person icon) with the sub-instruction 'Selecciona un rol', 'Usuario' (user icon), 'Contraseña' (lock icon), and 'Repetir contraseña' (lock icon). At the bottom is a green 'Registrar' button with a small icon.

Figura 3.2.1.3 Registro de empleados vista desde dispositivo móvil.

3.2.2 Modelo de datos

El modelo de datos tenemos dos tablas involucradas, la de admin y empleados (véase Figura 3.2.2.1), en estas dos se trabajara este caso de uso, a continuación se explicará:

- Admin: La tabla admin es donde se encuentra el gerente, quien realiza el registro en la tabla de empleados, además, el gerente es clave foránea en la tabla de empleados.
- Empleados: En la mayor parte de este caso de uso, se enfoca en esta tabla, dado que se habla de los empleados.

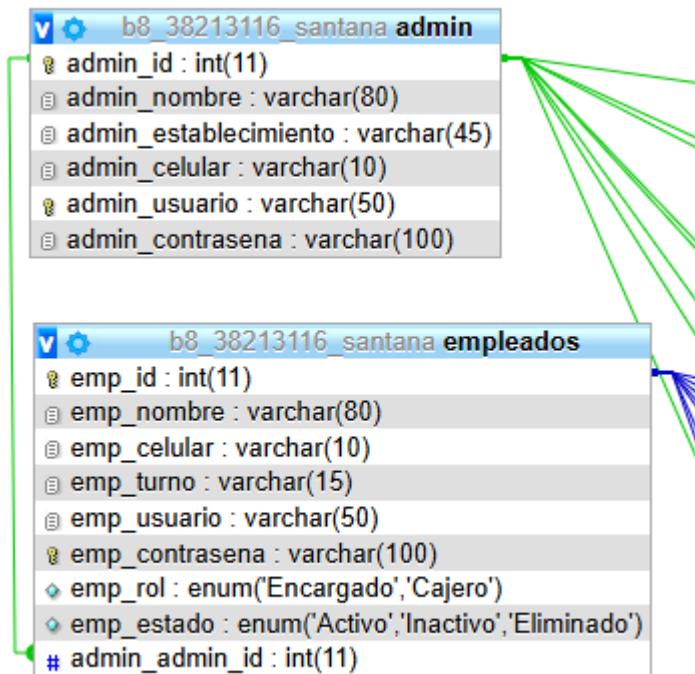


Figura 3.2.2.1 Modelo de datos.

3.2.3 Diseño de la interfaz de registro de empleados

El diseño de la interfaz busca ser simple de usar simple e intuitivo, a continuación se explicaran ambas:

- Registro de empleados: El registro de empleados se contempla mediante un formulario (véase Figura 3.2.3.1), esta ventana puede ser encontrada en el menú desplegable “Mis Empleados”, y en el submenú “Agregar”, este formulario cuenta con los campos necesarios para la información de un empleado, el formulario

mantiene un esquema limpio y sencillo de entender, el formulario cuenta con:

- Nombre del empleado.
- Número de celular.
- Selección de turno (Matutino, Vespertino o Nocturno).
- Selección de rol (Cajero o Encargado).
- Usuario.
- Contraseña.
- Repetición de contraseña.

Al final contamos con un botón que envía los datos al controlador correspondiente, para su registro, sin embargo, debe pasar las siguientes limitaciones:

- Todos los campos deben ser contestados.
- Los apartados de la contraseña deben coincidir.
- El nombre de usuario debe ser único.

Cuando se realice el registro, podemos observar en los empleados registrados, lo cual accedemos mediante en el menú desplegable podemos encontrar en “Mis Empleados”, después en “Existentes” (véase Figura 3.2.3.2), y tendremos una tabla con los campos:

- Usuario: Se eligió este dato que el nombre completo suele ser más amplio.
- Turno: Como campo informativo de su horario laboral.
- Acciones: Tenemos iconos representativos a acciones, en la cual se describen:
 - WhatsApp: Sirve para que el gerente al presionar este ícono, lo mande directo a la aplicación de mensajería de su número.

- Llamada: Sirve para que el gerente al presionar este icono, lo mande directo a la aplicación de llamada directa a su número.
- Editar: Sirve para varias cosas, en este icono nos despliega una ventana en la cual podemos inhabilitar, eliminar o cambiar la contraseña al empleado.
- Inhabilitar o eliminar empleados: En este diseño, se encuentran ambas opciones, dado que son acciones que generalmente se encuentran juntas, para acceder a estas opciones, en el menú desplegable podemos encontrar en “Mis Empleados”, después en “Existentes” (véase Figura 3.2.3.3), y seleccionando la fila del empleado, presionamos el ícono editar en color amarillo, este nos despliega una ventana flotante, de manera organizada contamos con los botones de Inhabilitar, eliminar y un formulario para actualizar la contraseña.

Registro de Empleados

	Nombre del empleado
	Número de celular
	Turno Selecciona un turno
	Rol Selecciona un rol
	Usuario
	Contraseña
	Repetir contraseña

Registrar

Figura 3.2.3.1 Formulario para registrar empleados.

Empleados

Usuario	Turno	Acciones	Estado	Rol
Cesar	Nocturno		Activo	Encargado

Figura 3.2.3.2 Tabla de empleados registrados.



Figura 3.2.3.3 Botones para eliminar o inhabilitar un usuario.

Usuario	Turno	Acciones	Estado
Cesar	Nocturno		Activo
Dalila	Matutino		Activo
Trini	Vespertino		Activo
Ramon	Vespertino		Activo
Ana	Vespertino		Activo
Lesly	Vespertino		Activo

Figura 3.2.3.4 Tabla de empleados vista de un dispositivo móvil.

3.2.4 Fragmentos de código

En este apartado, mostraremos los fragmentos de código de todas las partes involucradas:

- Registro de empleados interfaz: Tenemos el archivo `admin_employee_register.php` (véase Figura 3.2.4.1) en el cual tenemos un contenedor para almacenar los distintos campos de textos, tenemos la podriedad de redondeo de esquinas y sombra al contenedor, cada campo de texto tiene un placeholder y required, además, un script que ayuda a validar las contraseñas en el aspecto que sean las mismas. Si el formulario cumple con las indicaciones que se mencionaron en los apartados anteriores, los datos se envían al controlador submit.
- Registro de empleados backend: Tenemos el archivo `admin_employee_register_submit.php` (véase Figura 3.2.4.2). el cual se encarga de agregar al empleado a la base de datos, a continuación se explica:
 - Comenzamos recibiendo la información del post enviado, esa información la guardamos en variables.
 - Verificamos que cada variable tenga información, de no ser así, se redirige a la interfaz gráfica con un mensaje donde todos los campos deben ser completados.
 - Si están completos, se verifica si el usuario recibido mediante método post ya existe en las tablas admin y empleados, si no existe, se redirige a la interfaz con el mensaje de que el usuario ya existe.
 - Si no existe el usuario, se registra el empleado en la base de datos y después se redirige al usuario a visualizar los empleados ya registrados.

- Mostrar empleados: Tenemos el archivo admin_employee_view.php (véase Figura 3.2.4.3) en el cual se buscó ser un poco más seguros respecto a la información, dicho método al ser un tanto complicado y confuso para el programador, se decidió no repetir dichas acciones, este archivo, crea un contenedor donde sus características de redondeo y sombreado se hacen presentes, después se manda a llamar el archivo admin_employee_view_submit.php donde se maneja la información del empleado en una tabla.
- Obtener información de empleados: En el archivo admin_employee_view_submit.php (véase Figura 3.2.4.4) se maneja la consulta a la base de datos y la organización en cómo estos datos se muestran en una tabla, dicha tabla es responsive con texto al centro, además, de incluir los botones de acciones, como enviar mensaje a WhatsApp, llamada directa y acciones.
- Inhabilitar, habilitar y eliminar empleados: En el archivo admin_employee_view_submit.php (véase Figura 3.2.4.5) tenemos un modal, que al ser presionado en la columna de acciones, se abrirá este modal, proporcionandonos el botón de inhabilitar o habilitar según su estatus, además el de eliminar, junto con dos campos de texto y un botón para el cambio de contraseña.
- Cambio de estado de empleados: En el archivo admin_employee_view_status.php (véase Figura 3.2.4.6) en caso de cambiar un estado como inhabilitar, habilitar o eliminar llega la información a este archivo, primeramente obtenemos los datos necesarios al guardarlos en variables, se verifica que estas variables tenga información, en caso de que no, se redirige un mensaje de datos faltantes a la interfaz gráfica, en caso que se

verifiquen correctamente, se actualiza la información en la base de datos.

- Cambio de contraseña de empleados: En el archivo admin_employee_view_password.php (véase Figura 3.2.4.7) se obtiene la información recibida y es guardada en variables, después, se realiza las modificaciones en la base de datos

```
admin_employee_register.php X
CorteCaja > admin_employee_register.php ...
1 <?php session_start();
2 include 'includes/header.php';
3 if (!isset($_SESSION['admin_id'])) {
4 header("Location: login.php");
5 exit();
6 }
7 ?>
8
9 <!-- Formulario de registro de empleados -->
10 <div class="container my-5">
11 <div class="row justify-content-center">
12 <div class="col-md-8">
13 <div class="card shadow-lg rounded-4 border-0">
14 <div class="card-body p-4 bg-light">
15 <h1 class="text-center fw-bold mb-1">Registro de Empleados</h1>
16 <div class="card-body">
17 <form action="admin_employee_register_submit.php" method="post" onsubmit="return validarContrasenas()">
18 <!-- Campo Nombre -->
19 <div class="input-group mb-3">
20 <span class="input-group-text"><i class="fas fa-user"></i></span>
21 <div class="form-floating flex-grow-1">
22 <input type="text" class="form-control" id="nombre" name="nombre" placeholder="Nombre del empleado" value="php echo isset($nombre) ? $nombre : ''; ?" required>
23 <label for="nombre">Nombre del empleado</label>
24 </div>
25 </div>
26
27 <!-- Campo Celular -->
28 <div class="input-group mb-3">
29 <span class="input-group-text"><i class="fas fa-phone-alt"></i></span>
30 <div class="form-floating flex-grow-1">
31 <input type="tel" class="form-control" id="celular" name="celular" placeholder="Número de celular del empleado" value="php echo isset($celular) ? $celular : ''; ?" required>
32 <label for="celular">Número de celular</label>
33 </div>
34 </div>
35
36 <!-- Campo Turno -->
37 <div class="input-group mb-3">
38 <span class="input-group-text"><i class="fas fa-clock"></i></span>
39 <div class="form-floating flex-grow-1">
40 <select class="form-control" id="turno" name="turno" required>
41 <option value="">Selecciona un turno</option>
42 <option value="Matutino">Matutino</option>
43 <option value="Vespertino">Vespertino</option>
44 <option value="Nocturno">Nocturno</option>
45 </select>
46 <label for="turno">Turno</label>
47 </div>
48 </div>
49
50 <!-- Campo Rol -->
51 <div class="input-group mb-3">
52 <span class="input-group-text"><i class="fas fa-user-tag"></i></span>
53 <div class="form-floating flex-grow-1">
54 <select class="form-control" id="rol" name="rol" required>
55 <option value="">Selecciona un rol</option>
56 <option value="Cajero">Cajero</option>
```

Figura 3.2.4.1 Código del formulario para registrar un empleado.

```

admin_employee_register_submit.php x
CorteCaja > admin_employee_register_submit.php > ...
1  <?php
2  session_start(); // Iniciar la sesión
3
4  include 'includes/conexion_db.php';
5
6  if ($_SERVER['REQUEST_METHOD'] == "POST") {
7
8      $nombre = $_POST['nombre'];
9      $celular = $_POST['celular'];
10     $turno = $_POST['turno'];
11     $rol = $_POST['rol'];
12     $usuario = $_POST['usuario'];
13     $contrasena = $_POST['contrasena'];
14     $admin_admin_id = $_SESSION['admin_id']; // Asumiendo que ya tienes la sesión del administrador
15
16     // Validar que los campos no estén vacíos
17     if (empty($nombre) || empty($celular) || empty($turno) || empty($usuario) || empty($contrasena) || empty($admin_admin_id) || empty($rol)) {
18         // Redirigir al formulario con un mensaje de error
19         header("Location: admin_employee_register.php?error=empty_fields");
20         exit();
21     }
22
23     // Verificar si el usuario ya existe
24     $sql_check = "SELECT COUNT(*) FROM (
25         SELECT admin_usuario AS usuario FROM admin WHERE admin_usuario = :usuario
26         UNION ALL
27         SELECT emp_usuario FROM empleados WHERE emp_usuario = :usuario
28     ) AS total";
29
30     $stmt_check = $pdo->prepare($sql_check);
31     $stmt_check->execute([':usuario' => $usuario]);
32     $usuario_existente = $stmt_check->fetchColumn();
33
34     if ($usuario_existente > 0) {
35         header("Location: admin_employee_register.php?error=usuario_existente");
36         exit();
37     }
38
39     // Hash de la contraseña
40     $contrasena_hash = password_hash($contrasena, PASSWORD_DEFAULT);
41
42     // Preparar la consulta SQL
43     $sql = "INSERT INTO empleados (emp_id, emp_nombre, emp_celular, emp_turno, emp_usuario, emp_contraseña, emp_rol, emp_estado, admin_admin_id)
44     VALUES (null, :nombre, :celular, :turno, :usuario, :contraseña, :rol, 'Activo', :admin_admin_id)";
45
46     // Preparar la declaración
47     $stmt = $pdo->prepare($sql);
48
49     // Ejecuta la consulta con los valores user y password obtenidos del formulario
50     if ($stmt->execute([':nombre' => $nombre, ':celular' => $celular, ':turno' => $turno,
51                         ':usuario' => $usuario, ':contraseña' => $contrasena_hash, ':rol' => $rol,
52                         ':admin_admin_id' => $admin_admin_id])) {
53         header("Location: admin_employee_view.php");
54     } else {
55         echo "Error al registrar el empleado: " . $stmt->errorInfo()[2];
56     }

```

Figura 3.2.4.2 Código del backend para agregar un empleado a la base de datos.

```
admin_employee_view.php X
CorteCaja > admin_employee_view.php > div.container.my-5 > div.row.justify-content-center > div.col-md-8
1  <?php
2  session_start();
3  include 'includes/header.php'; ?>
4
5  <div class="container my-5">
6      <div class="row justify-content-center">
7          <div class="col-md-8">
8              <div class="card shadow-lg rounded-4 border-0">
9                  <div class="card-body p-4 bg-light">
10                     <h1 class="text-center fw-bold mb-1">Empleados</h1>
11                     <div class="card mt-4">
12                         <?php include 'admin_employee_view_submit.php'; ?>
13                     </div>
14                 </div>
15             </div>
16         </div>|
17     </div>
18 </div>
19
20 <?php include 'includes/footer.php'; ?>
```

Figura 3.2.4.3 Código de la interfaz de usuario para mostrar empleados.

```

admin_employee_view_submit.php X
CorteCaja > admin_employee_view_submit.php > ...
1  <?php
2  if (!isset($_SESSION['admin_id'])) {
3      echo "La sesión 'admin_id' no está configurada correctamente.";
4      exit;
5  }
6
7  include 'includes/conexion_db.php';
8  $admin_admin_id = $_SESSION['admin_id'];
9
10 // Obtener empleados del administrador
11 $sql = "SELECT * FROM empleados
12 WHERE admin_admin_id = :admin_admin_id AND emp_estado IN ('Activo', 'Inactivo')";
13 $stmt = $pdo->prepare($sql);
14 $stmt->execute(['admin_admin_id' => $admin_admin_id]);
15
16 // Generar tabla
17 if ($stmt->rowCount() > 0) {
18     echo "<div class='table-responsive text-center'>";
19     echo "<table class='table'>";
20     echo "<thead class='table-success'>";
21     echo "<tr><th>Usuario</th><th>Turno</th><th>Acciones</th><th>Estado</th><th>Rol</th></tr>";
22     echo "</thead>";
23     echo "<tbody>";
24     while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
25         $usuario = htmlspecialchars($row['emp_usuario']);
26         $turno = htmlspecialchars($row['emp_turno']);
27         $celular = htmlspecialchars($row['emp_celular']);
28         $emp_id = (int)$row['emp_id'];
29         $rol = htmlspecialchars($row['emp_rol']);
30         $estatus = htmlspecialchars($row['emp_estado']);
31
32         echo "<tr>";
33         echo "<td>$usuario</td>";
34         echo "<td>$turno</td>";
35         echo "<td>";
36             echo "<a href='https://wa.me/$celular' class='btn btn-success btn-sm' target='_blank'>";
37             echo "<i class='fa fa-whatsapp'></i>";
38             echo "</a>";
39             echo "<a href='tel:$celular' class='btn btn-primary btn-sm'>";
40             echo "<i class='fa fa-phone'></i>";
41             echo "</a>";
42             echo "<button class='btn btn-warning btn-sm' data-toggle='modal' data-target='#modalEmpleado$emp_id'>";
43             echo "<i class='fa fa-edit'></i>";
44             echo "</button>";
45         echo "</td>";
46         echo "<td>$estatus</td>";
47         echo "<td>$rol</td>";
48         echo "</tr>";
49
50     // Modal para cada empleado
51     echo "<div class='modal fade' id='modalEmpleado$emp_id' tabindex='-1' aria-labelledby='modalLabel$emp_id' aria-hidden='true'>";
52         echo "<div class='modal-dialog'>";
53             echo "<div class='modal-content'>";
54                 echo "<div class='modal-header'>";
55                     echo "<h5 class='modal-title'>Opciones para $usuario</h5>";
56                     echo "<button type='button' class='close' data-dismiss='modal'>&times;</button>";
57                 echo "</div>";
58                 echo "<div class='modal-body'>";

```

Figura 3.2.4.4 Código del backend para el manejo de información del empleado.

```

// Modal para cada empleado
echo "<div class='modal fade' id='modalEmpleado$emp_id' tabindex='-1' aria-labelledby='modalLabel$emp_id' aria-hidden='true'>
    <div class='modal-dialog'>
        <div class='modal-content'>
            <div class='modal-header'>
                <h5 class='modal-title'>Opciones para $usuario</h5>
                <button type='button' class='close' data-dismiss='modal'>&times;</button>
            </div>
            <div class='modal-body'>

                <button class='btn btn-warning btn-block mb-2' onclick=\"cambiarEstatus($emp_id, \".$estatus === 'Activo' ? 'Inactivo' : 'Activo').\"\">
                    .($estatus === 'Activo' ? 'Inhabilitar' : 'Habilitar').
                </button>

                <button class='btn btn-danger btn-block mb-2' onclick=\"cambiarEstatus($emp_id, 'Eliminado')\">
                    Eliminar
                </button>

                <form id='formCambiarPass$emp_id' onsubmit='return cambiarPassword(event, $emp_id)'>
                    <div class='form-group'>
                        <label>Nueva Contraseña</label>
                        <input type='password' class='form-control' id='pass1_$emp_id' required>
                    </div>
                    <div class='form-group'>
                        <label>Confirmar Contraseña</label>
                        <input type='password' class='form-control' id='pass2_$emp_id' required>
                    </div>
                    <button type='submit' class='btn btn-info btn-block'>Restaurar Contraseña</button>
                </form>
            </div>
        </div>
    </div>";

```

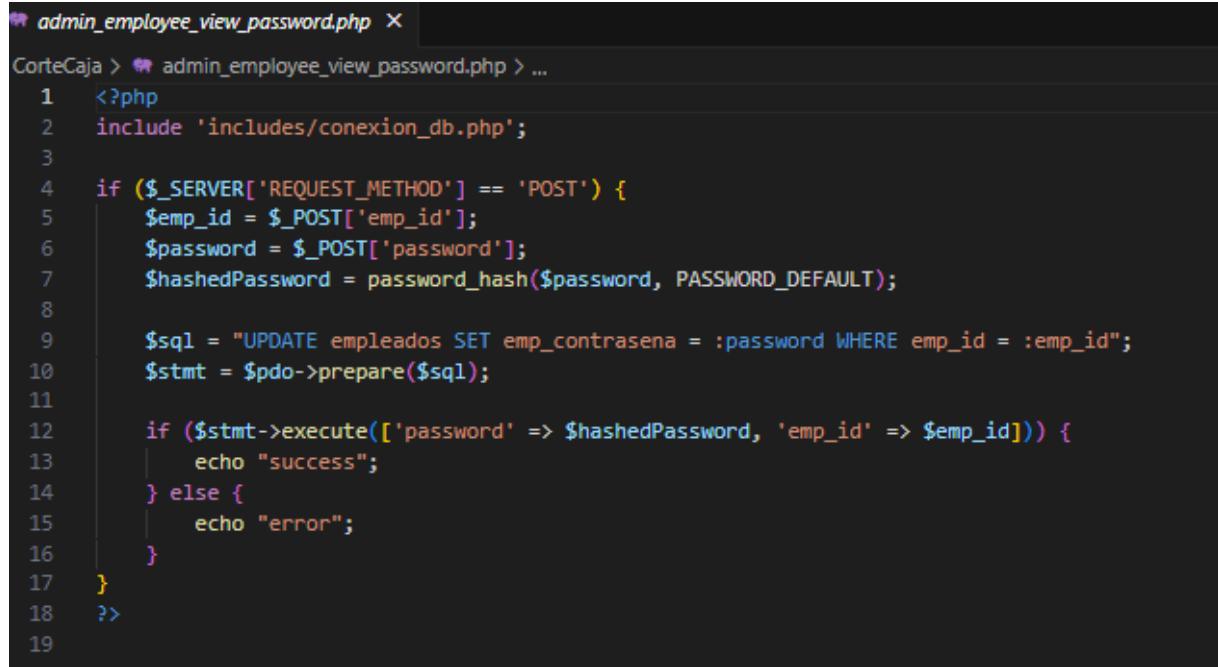
Figura 3.2.4.5 Código del backend habilitar, inhabilitar o cambiar contraseña de empleados.

```

admin_employee_view_status.php X
CorteCaja > admin_employee_view_status.php > ...
1  <?php
2  include 'includes/conexion_db.php';
3
4  if ($_SERVER['REQUEST_METHOD'] == 'POST') {
5      $emp_id = $_POST['emp_id'] ?? null;
6      $estatus = $_POST['estatus'] ?? null;
7
8      // Verificar que los datos llegan correctamente
9      file_put_contents('debug_log.txt', "ID: $emp_id, Estado: $estatus\n", FILE_APPEND);
10
11     if (!$emp_id || !$estatus) {
12         echo "error: datos faltantes";
13         exit;
14     }
15
16     $sql = "UPDATE empleados SET emp_estado = :estatus WHERE emp_id = :emp_id";
17     $stmt = $pdo->prepare($sql);
18
19     if ($stmt->execute(['estatus' => $estatus, 'emp_id' => $emp_id])) {
20         echo $estatus;
21     } else {
22         echo "error";
23     }
24 }
25
26 ?>
27

```

Figura 3.2.4.6 Código del backend para cambiar el estado de empleados.



```
admin_employee_view_password.php X
CorteCaja > admin_employee_view_password.php > ...
1 <?php
2 include 'includes/conexion_db.php';
3
4 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
5     $emp_id = $_POST['emp_id'];
6     $password = $_POST['password'];
7     $hashedPassword = password_hash($password, PASSWORD_DEFAULT);
8
9     $sql = "UPDATE empleados SET emp_contrasena = :password WHERE emp_id = :emp_id";
10    $stmt = $pdo->prepare($sql);
11
12    if ($stmt->execute(['password' => $hashedPassword, 'emp_id' => $emp_id])) {
13        echo "success";
14    } else {
15        echo "error";
16    }
17 }
18 ?>
19
```

Figura 3.2.4.7 Código del backend para cambiar la contraseña de empleados.

3.3 Actualizar datos de usuario.

3.3.1 Descripción del caso de uso

La descripción de este caso de uso es la actualización de los datos de un usuario, empleado o gerente, estos usuarios pueden editar:

- Nombre.
- Establecimiento (caso en gerente).
- Número de celular.
- Usuario.
- Contraseña.

En ocasiones los datos pueden ser erróneos o se han modificado, por ello, tener la información actualizada siempre es importante. Esta opción se encuentra en el menú desplegable “Perfil”, nos aparecerá un botón “Actualizar

mis datos” (véase Figura 3.3.1.1), el cual nos enviará al formulario para la actualización de los datos (véase Figura 3.3.1.2).



Figura 3.3.1.1 Botón para la actualización de datos.

A screenshot of a "Actualizar Empleado" (Update Employee) form. The title is at the top center in a large, bold, dark font. Below it is a sub-instruction: "No es necesario llenar todos los campos." (It is not necessary to fill all fields.) The form consists of five input fields, each with a small icon in a grey box to its left: "Nombre" (Name) with a person icon, "Número de celular" (Cell phone number) with a telephone icon, "Usuario" (User) with an eye icon, "Contraseña" (Password) with a lock icon, and "Repetir contraseña" (Repeat password) with a lock icon. At the bottom is a green button with a white border containing a small icon of a clipboard with a checkmark followed by the text "Actualizar Perfil" (Update Profile).

Figura 3.3.1.2 Formulario para la actualización de datos.

3.3.2 Modelo de datos

El modelo de datos para este caso de uso son únicamente dos tablas (véase Figura 3.3.2.1), las cuales según el usuario será la tabla en la que sea

afectada, por ejemplo, si un gerente decide actualizar sus datos, la tabla admin será la que sea modificada, mientras que un empleado será la tabla empleados.

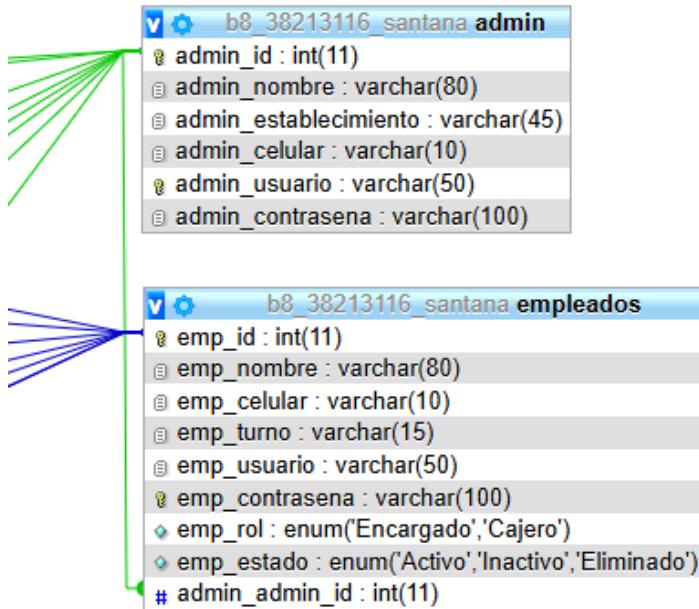


Figura 3.3.2.1 Tablas involucradas en el caso de uso.

3.3.3 Diseño de la interfaz de registro de empleados

El diseño de la interfaz gráfica al igual que las anteriores, busca ser simple de entender y ejecutar, la ubicación de está se encuentra en la sección de “Perfil”, donde nos muestra la información del usuario logueado (véase Figura 3.3.3.1 y Figura 3.3.3.2), la información que nos muestra del usuario es la siguiente:

- ID del usuario.
- Nombre completo.
- Nombre de usuario.
- Número de celular.
- Turno del empleado (Solo para empleados).
- Nombre del establecimiento (Solo para gerentes).

Una vez tenemos esta interfaz, podremos ver un botón “Actualizar mis datos”, para ambos usuarios (gerentes y empleados) el formulario es similar, la única diferencia es el campo para el nombre del establecimiento en el usuario de gerentes (véase Figura 3.3.3.3 y Figura 3.3.3.4). El formulario tiene la leyenda “No es necesario llenar todos los campos”, sólo se llenarán los campos que el usuario requiera, siendo un formulario más amigable, fácil de comprender y limpio para el usuario.

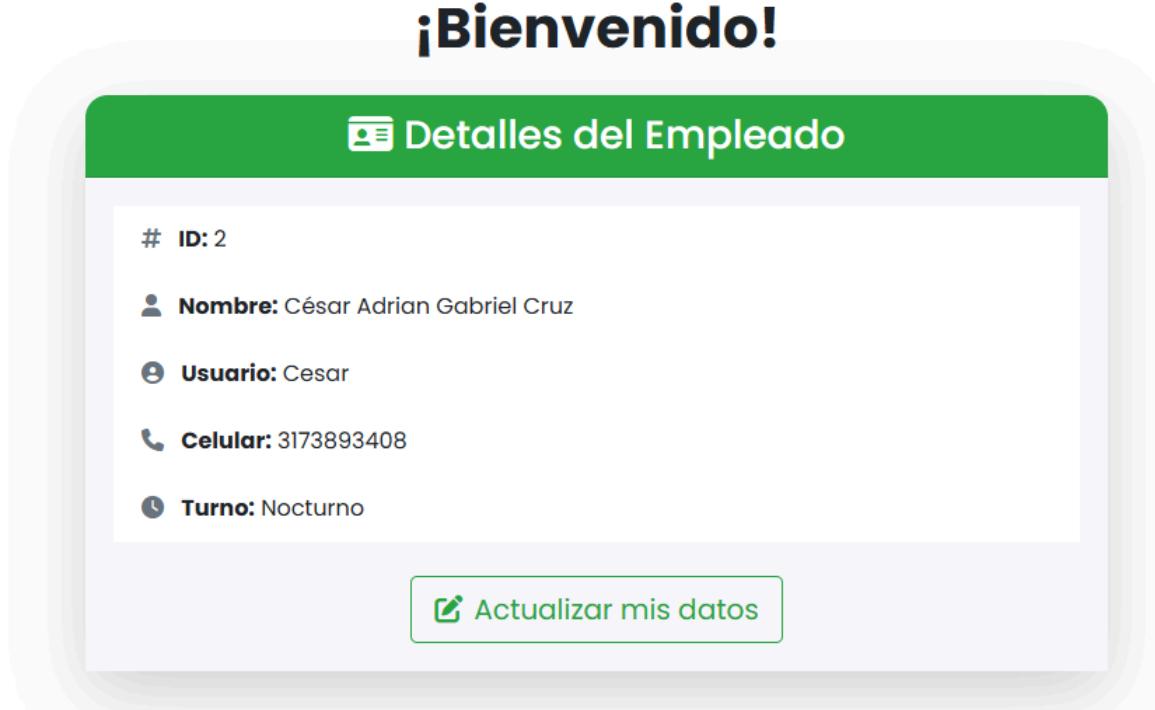


Figura 3.3.3.1 Información de un empleado.

¡Bienvenido!

Detalles del administrador

ID: 2

👤 Nombre: Cipriano Gonzalez

👤 Usuario: Pipis

📞 Celular: 3173879348

📍 Establecimiento: Santana Elia

[Actualizar mis datos](#)

Figura 3.3.3.2 Información de un gerente.

Actualización de Administrador

No es necesario llenar todos los campos.

	Nombre
	Establecimiento
	Número de celular
	Usuario
	Contraseña
	Repetir contraseña
 Actualizar Perfil	

Figura 3.3.3.3 Formulario de actualización de un gerente.

Actualización de Empleado

No es necesario llenar todos los campos.

	Nombre
	Número de celular
	Usuario
	Contraseña
	Repetir contraseña

 Actualizar Perfil

Figura 3.3.3.4 Formulario de actualización de un empleado.

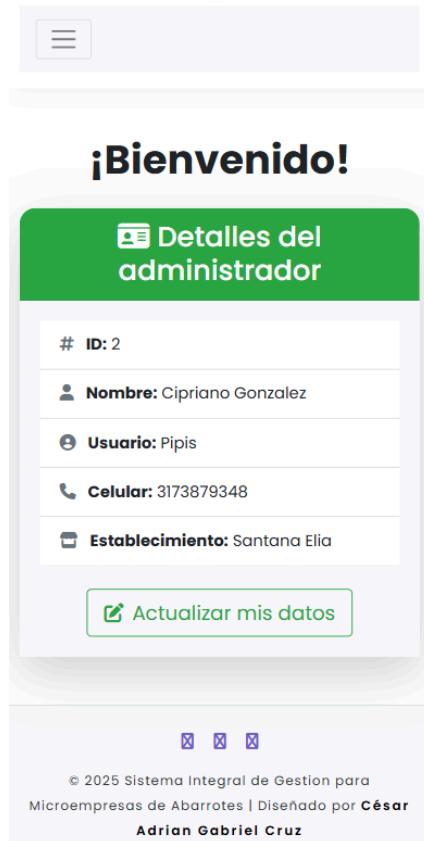


Figura 3.3.3.5 Datos del gerente vista de un dispositivo móvil.

3.3.4 Fragmentos de código

Los elementos del código tenemos las partes de la vista de información y la actualización de los datos del empleado y gerente, a continuación se describe cada una de ellas:

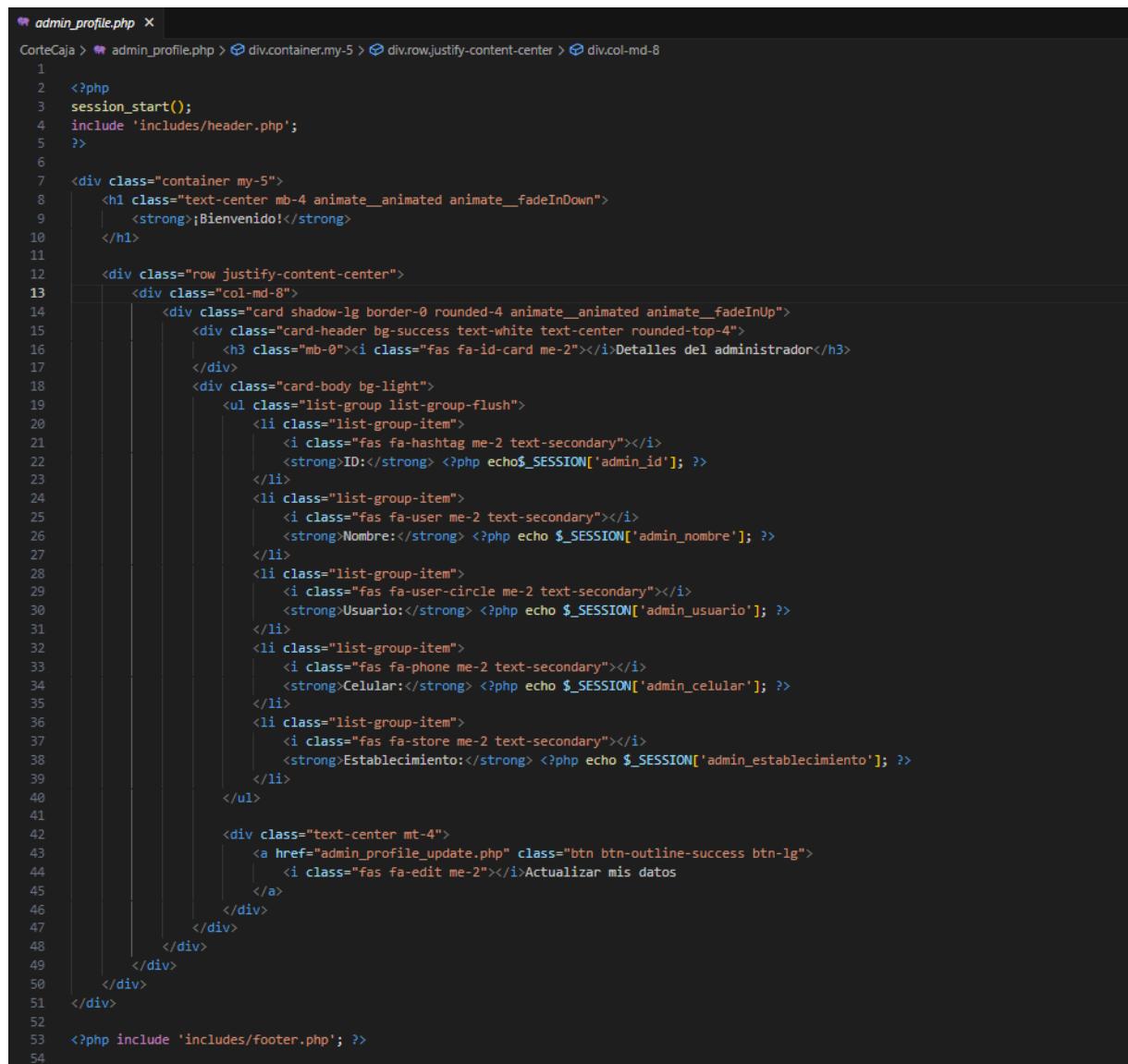
- Vista de información: Tenemos el archivo `admin_profile.php` (véase Figura 3.3.4.1) en el cual tenemos un contenedor para la información, este se ve constituido por iconos con su respectiva descripción, para mostrar la información se hace uso de las variables de sesión que son creadas al realizar el logueo a la aplicación web, tales como `id`, `nombre`, `usuario`, `celular` y `establecimiento`. Al final tenemos un botón que nos redirige al formulario para actualizar los datos del gerente. Para el empleado

tenemos el archivo employee_profile.php (véase Figura 3.3.4.2) en el cual es similar al card del gerente, diferenciándose con el turno en lugar del establecimiento.

- Formulario de actualización: Tenemos el archivo admin_profile_update.php (véase Figura 3.3.4.3) donde tenemos distintos inputs para el ingreso de información, también una función para validar las contraseñas sean estas idénticas, el formulario se envía a su respectivo submit mediante post. Para el empleado tenemos el archivo employee_profile_update.php (véase Figura 3.3.4.4) donde sigue manteniendo la misma diferencia.
- Actualización de datos: Tenemos el archivo admin_profile_update_submit.php (véase Figura 3.3.4.5) para gerentes y employee_profile_update_submit.php (véase Figura 3.3.4.6) para empleados en cual fungue como conexión con la base de datos para la actualización, sin embargo se explica a continuación:
 - Primeramente recibimos los datos del post y las guardamos en variables, los cuales como algunos datos post serán nulos, de no recibir información ese dato será nulo dicha variable, pero si tiene información, se le asignará dicha información a la variable,
 - Después continuamos con una consulta para obtener los datos del usuario y estos datos se le asignan únicamente a las variables con valor nulo.
 - Luego se hace una comparación, en la cual si el usuario recibido mediante post (si es nulo al principio, el usuario será el que tome de la base de datos al consultar la información) es diferente al obtenido en la consulta en la

base de datos, se verificará si este usuario existe en las tablas de admin y empleado para evitar una duplicación, si existe un usuario ya registrado, se dará aviso al usuario que no se puede cambiar dicha información, en caso de que el usuario sea igual al de la base de datos o diferente.

- Se actualizarán todas las variables en la base de datos.
- También se actualizarán las variables de sesión.



```

1  <?php
2  session_start();
3  include 'includes/header.php';
4  ?>
5
6
7  <div class="container my-5">
8      <h1 class="text-center mb-4 animate__animated animate__fadeInDown">
9          <strong>¡Bienvenido!</strong>
10     </h1>
11
12    <div class="row justify-content-center">
13        <div class="col-md-8">
14            <div class="card shadow-lg border-0 rounded-4 animate__animated animate__fadeInUp">
15                <div class="card-header bg-success text-white text-center rounded-top-4">
16                    <h3 class="mb-0"><i class="fas fa-id-card me-2"></i>Detalles del administrador</h3>
17                </div>
18                <div class="card-body bg-light">
19                    <ul class="list-group list-group-flush">
20                        <li class="list-group-item">
21                            <i class="fas fa-hashtag me-2 text-secondary"></i>
22                            <strong>ID:</strong> <?php echo $_SESSION['admin_id']; ?>
23                        </li>
24                        <li class="list-group-item">
25                            <i class="fas fa-user me-2 text-secondary"></i>
26                            <strong>Nombre:</strong> <?php echo $_SESSION['admin_nombre']; ?>
27                        </li>
28                        <li class="list-group-item">
29                            <i class="fas fa-user-circle me-2 text-secondary"></i>
30                            <strong>Usuario:</strong> <?php echo $_SESSION['admin_usuario']; ?>
31                        </li>
32                        <li class="list-group-item">
33                            <i class="fas fa-phone me-2 text-secondary"></i>
34                            <strong>Celular:</strong> <?php echo $_SESSION['admin_celular']; ?>
35                        </li>
36                        <li class="list-group-item">
37                            <i class="fas fa-store me-2 text-secondary"></i>
38                            <strong>Establecimiento:</strong> <?php echo $_SESSION['admin_establecimiento']; ?>
39                        </li>
40                    </ul>
41
42                    <div class="text-center mt-4">
43                        <a href="admin_profile_update.php" class="btn btn-outline-success btn-lg">
44                            <i class="fas fa-edit me-2"></i>Actualizar mis datos
45                        </a>
46                    </div>
47                </div>
48            </div>
49        </div>
50    </div>
51 <?php include 'includes/footer.php'; ?>
52
53
54

```

Figura 3.3.4.1 Card para mostrar información del gerente.

```
employee_profile.php ×
CorteCaja > employee_profile.php > ...
1  <?php
2  session_start();
3  include 'includes/header.php';
4  ?>
5
6  <div class="container my-5">
7      <h1 class="text-center mb-4 animate__animated animate__fadeInDown">
8          <strong>Bienvenido!</strong>
9      </h1>
10
11     <div class="row justify-content-center">
12         <div class="col-md-8">
13             <div class="card shadow-lg border-0 rounded-4 animate__animated animate__fadeInUp">
14                 <div class="card-header bg-success text-white text-center rounded-top-4">
15                     <h3 class="mb-0"><i class="fas fa-id-card me-2"></i>Detalles del Empleado</h3>
16                 </div>
17                 <div class="card-body bg-light">
18                     <ul class="list-group list-group-flush">
19                         <li class="list-group-item">
20                             <i class="fas fa-hashtag me-2 text-secondary"></i>
21                             <strong>ID:</strong> <?php echo $_SESSION['emp_id']; ?>
22                         </li>
23                         <li class="list-group-item">
24                             <i class="fas fa-user me-2 text-secondary"></i>
25                             <strong>Nombre:</strong> <?php echo $_SESSION['emp_nombre']; ?>
26                         </li>
27                         <li class="list-group-item">
28                             <i class="fas fa-user-circle me-2 text-secondary"></i>
29                             <strong>Usuario:</strong> <?php echo $_SESSION['emp_usuario']; ?>
30                         </li>
31                         <li class="list-group-item">
32                             <i class="fas fa-phone me-2 text-secondary"></i>
33                             <strong>Celular:</strong> <?php echo $_SESSION['emp_celular']; ?>
34                         </li>
35                         <li class="list-group-item">
36                             <i class="fas fa-clock me-2 text-secondary"></i>
37                             <strong>Turno:</strong> <?php echo $_SESSION['emp_turno']; ?>
38                         </li>
39                     </ul>
40
41                     <div class="text-center mt-4">
42                         <a href="employee_profile_update.php" class="btn btn-outline-success btn-lg">
43                             <i class="fas fa-edit me-2"></i>Actualizar mis datos
44                         </a>
45                     </div>
46                 </div>
47             </div>
48         </div>
49     </div>
50
51     <?php include 'includes/footer.php'; ?>
52
53
```

Figura 3.3.4.2 Card para mostrar información del empleado.

```

  admin_profile_update.php
CorteCaja > admin_profile_update.php > ...
1 <?php
2 session_start();
3 include 'includes/header.php';
4 >>
5
6 <div class="container my-5">
7   <div class="row justify-content-center">
8     <div class="col-md-8">
9       <div class="card shadow-lg rounded-4 border-0">
10         <div class="card-body p-4 bg-light">
11           <h1 class="text-center fw-bold mb-1">Actualización de Administrador</h1>
12           <p class="text-center text-muted mb-4">No es necesario llenar todos los campos.</p>
13
14           <form action="admin_profile_update_submit.php" method="post" onsubmit="return validarContrasenas();">
15
16             <input type="hidden" name="admin_id" value="php echo $_SESSION['admin_id']; ?">
17
18             <!-- Nombre -->
19             <div class="input-group mb-4">
20               <span class="input-group-text"><i class="fas fa-user"></i></span>
21               <div class="form-floating flex-grow-1">
22                 <input type="text" class="form-control" id="nombre" name="nombre" placeholder="Actualiza tu nombre" value="php echo isset($nombre) ? $nombre : '' ; ?&gt;"&gt;
23                 &lt;label for="nombre"&gt;Nombre&lt;/label&gt;
24               &lt;/div&gt;
25             &lt;/div&gt;
26
27             &lt;!-- Establecimiento --&gt;
28             &lt;div class="input-group mb-4"&gt;
29               &lt;span class="input-group-text"&gt;&lt;i class="fas fa-store"&gt;&lt;/i&gt;&lt;/span&gt;
30               &lt;div class="form-floating flex-grow-1"&gt;
31                 &lt;input type="text" class="form-control" id="establecimiento" name="establecimiento" placeholder="Nombre del establecimiento" value="<?php echo isset($establecimiento) ? $establecimiento : '' ; ?&gt;"&gt;
32                 &lt;label for="establecimiento"&gt;Establecimiento&lt;/label&gt;
33               &lt;/div&gt;
34             &lt;/div&gt;
35
36             &lt;!-- Celular --&gt;
37             &lt;div class="input-group mb-4"&gt;
38               &lt;span class="input-group-text"&gt;&lt;i class="fas fa-phone-alt"&gt;&lt;/i&gt;&lt;/span&gt;
39               &lt;div class="form-floating flex-grow-1"&gt;
40                 &lt;input type="tel" class="form-control" id="celular" name="celular" placeholder="Celular" value="<?php echo isset($celular) ? $celular : '' ; ?&gt;"&gt;
41                 &lt;label for="celular"&gt;Número de celular&lt;/label&gt;
42               &lt;/div&gt;
43             &lt;/div&gt;
44
45             &lt;!-- Usuario --&gt;
46             &lt;div class="input-group mb-4"&gt;
47               &lt;span class="input-group-text"&gt;&lt;i class="fas fa-user-circle"&gt;&lt;/i&gt;&lt;/span&gt;
48               &lt;div class="form-floating flex-grow-1"&gt;
49                 &lt;input type="text" class="form-control" id="usuario" name="usuario" placeholder="Usuario" value="<?php echo isset($usuario) ? $usuario : '' ; ?&gt;"&gt;
50                 &lt;label for="usuario"&gt;Usuario&lt;/label&gt;
51               &lt;/div&gt;
52             &lt;/div&gt;
53
54             &lt;!-- Contraseña --&gt;
55             &lt;div class="input-group mb-4"&gt;
56               &lt;span class="input-group-text"&gt;&lt;i class="fas fa-lock"&gt;&lt;/i&gt;&lt;/span&gt;
57               &lt;div class="form-floating flex-grow-1"&gt;
58                 &lt;input type="password" class="form-control" id="contrasena" name="contrasena" placeholder="Contraseña"&gt;
59               &lt;/div&gt;
60             &lt;/div&gt;
61
62           &lt;/form&gt;
63
64         &lt;/div&gt;
65       &lt;/div&gt;
66     &lt;/div&gt;
67   &lt;/div&gt;
68 &lt;/div&gt;
</pre

```

Figura 3.3.4.3 Formulario para actualizar información del gerente.

```

  employee_profile_update.php
CorteCaja > employee_profile_update.php > ...
1 <?php
2 session_start();
3 include 'includes/header.php';
4 >>
5
6 <div class="container my-5">
7   <div class="row justify-content-center">
8     <div class="col-md-8">
9       <div class="card shadow-lg rounded-4 border-0">
10         <div class="card-body p-4 bg-light">
11           <h1 class="text-center fw-bold mb-1">Actualización de Empleado</h1>
12           <p class="text-center text-muted mb-4">No es necesario llenar todos los campos.</p>
13
14           <form The input element represents a typed data field, usually with a form control to allow the user to edit the data. onsubmit="return validarContrasenas();">
15             MDN Reference
16             <input type="hidden" name="emp_id" value="php echo $_SESSION['emp_id']; ?">
17
18             <!-- Nombre -->
19             <div class="input-group mb-3">
20               <span class="input-group-text"><i class="fas fa-user"></i></span>
21               <div class="form-floating flex-grow-1">
22                 <input type="text" class="form-control" id="nombre" name="nombre" placeholder="Nombre" value="php echo isset($nombre) ? $nombre : '' ; ?&gt;"&gt;
23                 &lt;label for="nombre"&gt;Nombre&lt;/label&gt;
24               &lt;/div&gt;
25             &lt;/div&gt;
26
27             &lt;!-- Celular --&gt;
28             &lt;div class="input-group mb-3"&gt;
29               &lt;span class="input-group-text"&gt;&lt;i class="fas fa-phone-alt"&gt;&lt;/i&gt;&lt;/span&gt;
30               &lt;div class="form-floating flex-grow-1"&gt;
31                 &lt;input type="tel" class="form-control" id="celular" name="celular" placeholder="Número de celular" value="<?php echo isset($celular) ? $celular : '' ; ?&gt;"&gt;
32                 &lt;label for="celular"&gt;Número de celular&lt;/label&gt;
33               &lt;/div&gt;
34             &lt;/div&gt;
35
36             &lt;!-- Usuario --&gt;
37             &lt;div class="input-group mb-3"&gt;
38               &lt;span class="input-group-text"&gt;&lt;i class="fas fa-user-circle"&gt;&lt;/i&gt;&lt;/span&gt;
39               &lt;div class="form-floating flex-grow-1"&gt;
40                 &lt;input type="text" class="form-control" id="usuario" name="usuario" placeholder="Usuario" value="<?php echo isset($usuario) ? $usuario : '' ; ?&gt;"&gt;
41                 &lt;label for="usuario"&gt;Usuario&lt;/label&gt;
42               &lt;/div&gt;
43             &lt;/div&gt;
44
45             &lt;!-- Contraseña --&gt;
46             &lt;div class="input-group mb-3"&gt;
47               &lt;span class="input-group-text"&gt;&lt;i class="fas fa-lock"&gt;&lt;/i&gt;&lt;/span&gt;
48               &lt;div class="form-floating flex-grow-1"&gt;
49                 &lt;input type="password" class="form-control" id="contrasena" name="contrasena" placeholder="Contraseña"&gt;
50               &lt;/div&gt;
51             &lt;/div&gt;
52
53           &lt;/form&gt;
54
55         &lt;/div&gt;
56       &lt;/div&gt;
57     &lt;/div&gt;
58   &lt;/div&gt;
59 &lt;/div&gt;
</pre

```

Figura 3.3.4.4 Formulario para actualizar información del empleado.

```

admin_profile_update_submit.php ×
CorteCaja > admin_profile_update_submit.php > ...
1  <?php
2  // Incluir la conexión a la base de datos
3  include 'includes/conexion_db.php';
4  session_start(); // Iniciar la sesión
5
6  // Verificar si se ha enviado el formulario
7  if ($_SERVER["REQUEST_METHOD"] == "POST") {
8
9      // Obtener el ID del administrador desde la sesión
10     $admin_id = $_SESSION['admin_id'];
11
12     // Validar que el campo admin_id tenga dato
13     if (empty($admin_id)) {
14         // Redirigir al formulario con un mensaje de error
15         header("Location: admin_profile_update.php");
16         exit();
17     }
18
19     // Recuperar los datos ingresados por el usuario
20     $nombre = !empty($_POST['nombre']) ? $_POST['nombre'] : null;
21     $establecimiento = !empty($_POST['establecimiento']) ? $_POST['establecimiento'] : null;
22     $celular = !empty($_POST['celular']) ? $_POST['celular'] : null;
23     $usuario = !empty($_POST['usuario']) ? $_POST['usuario'] : null;
24     $contrasena = !empty($_POST['contrasena']) ? $_POST['contrasena'] : null;
25
26     // Obtener los valores actuales del administrador si un campo está vacío
27     $sqlSelect = "SELECT admin_nombre, admin_establecimiento, admin_celular, admin_usuario FROM admin WHERE admin_id = :admin_id";
28     $stmtSelect = $pdo->prepare($sqlSelect);
29     $stmtSelect->execute(['admin_id' => $admin_id]);
30     $adminActual = $stmtSelect->fetch(PDO::FETCH_ASSOC);
31
32     // Si el campo está vacío, mantener el valor actual de la base de datos
33     $nombre = $nombre ?? $adminActual['admin_nombre'];
34     $establecimiento = $establecimiento ?? $adminActual['admin_establecimiento'];
35     $celular = $celular ?? $adminActual['admin_celular'];
36     $usuario = $usuario ?? $adminActual['admin_usuario'];
37
38     // Verificar si el nuevo usuario ya está en uso por otro admin

```

Figura 3.3.4.5 Backend para actualizar información del gerente.

```

employee_profile_update_submit.php ×
CorteCaja > employee_profile_update_submit.php > ...
1  <?php
2  // Incluir la conexión a la base de datos
3  include 'includes/conexion_db.php';
4  session_start(); // Iniciar la sesión
5
6  // Verificar si se ha enviado el formulario
7  if ($_SERVER["REQUEST_METHOD"] == "POST") {
8
9      // Obtener el ID del empleado desde la sesión
10     $emp_id = $_SESSION['emp_id'];
11
12     // Validar que los campo emp_id tenga dato
13     if (empty($emp_id)) {
14         // Redirigir al formulario con un mensaje de error
15         header("Location: employee_profile_update.php");
16         exit();
17     }
18
19     // Recuperar los datos ingresados por el usuario
20     $nombre = !empty($_POST['nombre']) ? $_POST['nombre'] : null;
21     $celular = !empty($_POST['celular']) ? $_POST['celular'] : null;
22     $usuario = !empty($_POST['usuario']) ? $_POST['usuario'] : null;
23     $contrasena = !empty($_POST['contrasena']) ? $_POST['contrasena'] : null;
24
25     // Obtener los valores actuales del empleado si un campo está vacío
26     $sqlSelect = "SELECT emp_nombre, emp_celular, emp_usuario FROM empleados WHERE emp_id = :emp_id";
27     $stmtSelect = $pdo->prepare($sqlSelect);
28     $stmtSelect->execute(['emp_id' => $emp_id]);
29     $empleadoActual = $stmtSelect->fetch(PDO::FETCH_ASSOC);
30
31     // Si el campo está vacío, mantener el valor actual de la base de datos
32     $nombre = $nombre ?? $empleadoActual['emp_nombre'];
33     $celular = $celular ?? $empleadoActual['emp_celular'];
34     $usuario = $usuario ?? $empleadoActual['emp_usuario'];
35
36     // Verificar si el nuevo usuario ya está en uso por otro admin

```

Figura 3.3.4.6 Backend para actualizar información del empleado.

3.4 Registrar corte, gasto y/o pago con tarjeta.

3.4.1 Descripción del caso de uso

La descripción de este caso de uso es de la más importantes dado que es una herramienta primordial dentro de la aplicación web, este caso abarca tres funciones las cuales se explicará a continuación:

- Registrar pago con tarjeta: Actualmente los negocios incorporan los pagos con tarjeta para que los clientes puedan pagar sus productos, dado que es una herramienta que permite al cliente no cargar dinero en efectivo o pagos a meses. Los pagos con tarjeta forman parte de la realización de corte de caja, debido a que son ingresos al negocio y forman parte de la venta.
- Registrar gasto: Los gastos son las salidas de dinero de caja, entonces es para adquirir productos para su posterior venta, siendo esto, los gastos se deben registrar debido a que forman parte del corte de caja, además, son salidas de caja de dinero de grandes cantidades.
- Registrar corte: Está siendo la función primordial para la obtención de la venta y registro del corte de caja, la cual depende del registro de pagos con tarjeta y gastos para su elaboración.

Con dicha información, durante un turno se deben registrar constantemente los pagos con tarjeta y gastos conforme se van realizando, dado que este también se guarda la hora y fecha de su realización para posibles situaciones futuras, ya que la realización del corte de caja, solo contempla los registros, si hace falta registrar alguno, este por obvias razones no serán tomadas en cuenta, por ello su importancia de registrar cada pago con tarjeta y gasto.

Dichas funciones se encuentran en el usuario de empleado, en el menú desplegable en “Opciones” (véase Figura 3.4.1.1), aquí podemos encontrar varias funciones entre ellas las de este caso de uso, cada opción llevará a su respectivo registro y/o ventana.

El proceso es durante el turno del empleado, este realizará un registro en las ventanas de Gastos y Pagos Terminal, al finalizar el turno, el empleado accede a la función de Corte, llena los campos correspondientes y agrega dicha información, esta información aparecerá en el index del empleado y gerente (véase Figura 3.4.1.2), para su posterior visualización.

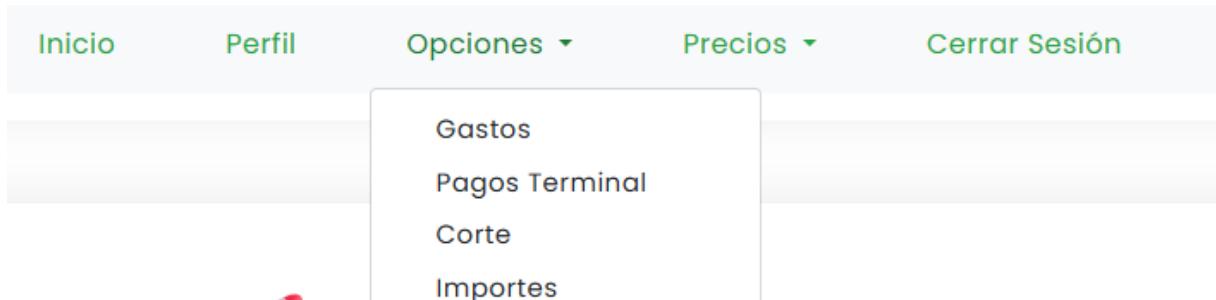


Figura 3.4.1.1 Funciones para el empleado.

Detalle del Corte	Detalle del Corte	Detalle del Corte
Empleado: Dalila Gomez Farias Caja: \$340 Venta: \$2873 Gasto Total: \$6673 Descripción: TORTILLA - \$140 (04:13 PM), TIRAS - \$120 (04:13 PM), PEPSI - \$1494 (04:13 PM), MEDINA PERFUMERIA - \$1000 (04:14 PM), ABARROTE - \$120 (04:15 PM), COYOTE - \$516 (04:15 PM), NORTEÑA - \$108 (04:16 PM), COCA COLA - \$1935 (04:17 PM), LALA - \$475 (04:17 PM), MARINELA - \$765 (04:19 PM) Pago Terminal: \$201 Comentario: Saldo en sistema: \$590 - Bote de recargas: \$510 - Bote de monedas: \$705 Caja Anterior: \$4341 Fecha: 05/05 04:22 PM	Empleado: César Adrian Gabriel Cruz Caja: \$2811 Venta: \$1725 Gasto Total: \$1300 Descripción: PIPIS - \$1300 (02:41 AM) Pago Terminal: \$162 Comentario: Saldo en sistema: \$1090 - Bote de recargas: \$0 - Bote de monedas: \$900 Caja Anterior: \$2548 Fecha: 05/05 03:07 AM	Empleado: Trinidad Caja: \$2548 Venta: \$4993 Gasto Total: \$5000 Descripción: FAVIO - \$5000 (10:36 PM) Pago Terminal: \$1065 Comentario: Saldo en sistema: \$1190 - Bote de recargas: \$550 - Bote de monedas: \$1 Caja Anterior: \$3620 Fecha: 04/05 10:58 PM

Figura 3.4.1.2 Página principal despues de loguearse.

3.4.2 Modelo de datos

El modelo de datos para este caso de uso es de los más extensos dado que abarca muchas tablas (véase Figura 3.4.2.1), las cuales se explicaran a continuación:

- Admin: Esta es la tabla la cual forma parte como clave foránea en todas las demás tablas involucradas, esto debido a la separación de datos por gerente.
- Empleados: Dicha tabla forma parte como clave foránea en las tablas como gasto, tarjeta y corte, para rescatar quien fue quien realizó dicho registro.
- Categorias_gasto: Esta tabla siendo la más simple, registra la descripción de los gastos, esto para que sea más simple de seleccionar dicha descripción sin tener la necesidad de escribirla, esto lleva un orden y fácil de identificación dentro de los gastos.
- Gasto: Esta tabla se registra cada gasto, junto con la cantidad y descripción, aquí tenemos una columna llamada procesado, esto es para el momento de registrar el corte de caja, solo se tomen los gastos que no han sido procesados, al momento de registrar el corte estos gastos pasan a tener un si procesados.
- Tarjeta: Similar a la tabla de gasto, sin embargo, no cuenta con la descripción debido a que no hay necesidad de, por ello también tenemos la columna de procesado para la misma funcionalidad de la tabla gasto.
- Corte: Siendo la tabla principal donde llega todo la información, en esta se realiza el registro de los gastos, su descripción, pagos con tarjeta, comentarios, venta entre más campos, esta tabla se puede definir como la final del proceso de este caso de uso, donde posteriormente se especificará más a detalle el uso de esta.

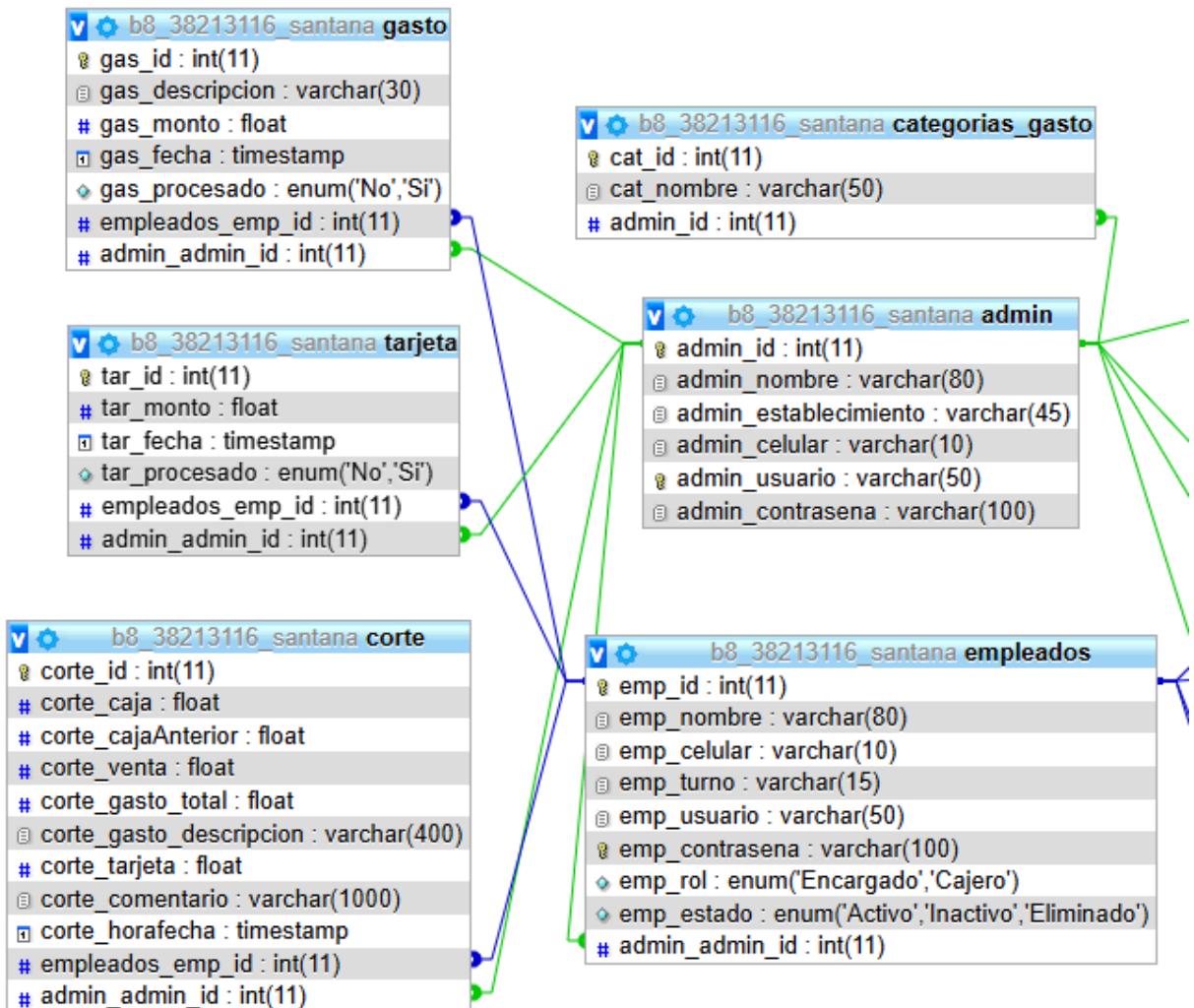


Figura 3.4.2.1 Modelo de datos para la realización del corte de caja.

3.4.3 Diseño de la interfaz

El diseño de la interfaz tenemos que separarlos para poder explicarlos a profundidad, la interfaz gráfica busca ser simple para que el usuario no tenga que memorizar procesos amplios o complejos, teniendo formulario y tablas de consulta rápida y entendible, cada una se explica a continuación:

- Pagos con tarjeta: Dentro de la interfaz gráfica tenemos dos card view (véase Figura 3.4.3.1), uno para registrar el pago con

terminal, y otro para visualizar los que ya fueron registrados antes de realizar el corte de caja.

- Dentro del formulario únicamente hay que ingresar la cantidad del pago con terminal y presionar el botón “Aregar”, lo cual, el card view inferior se actualiza mostrando una tabla con los pagos con terminal registrados (véase Figura 3.4.3.2).
- En la tabla de los pagos con terminal registrados, nos mostrará el monto, la hora en la que se realizó y acciones, las cuales son dos:
 - Actualizar: Botón de color verde, en caso de una equivocación en la cantidad, se podrá cambiar la cantidad en la columna de monto, despues de cambiar dicha cantidad se presiona el botón verde para actualizar la cantidad (véase Figura 3.4.3.3).
 - Borrar: Botón de color rojo tiene la funcionalidad de eliminar el registro, en caso que no se permita actualizar, o incluso que el pago con tarjeta haya sido rechazado o alguna situación en la que dicho ingreso no debe ser contemplado. Solamente nos situamos en la fila del registro a eliminar, presionamos el botón de color rojo, y aceptamos su eliminación (véase Figura 3.4.3.4).
- Gastos: Dentro de la interfaz gráfica (véase Figura 3.4.3.5) podemos observar que es similar a la ventana de los pagos con terminal, esto como se mencionó anteriormente, busca ser similar para que el usuario memorice menos pasos.
 - Tenemos una diferencia en la cual tenemos un selector para el motivo del gasto, esto usualmente para los proveedores

cuando dejan el producto, el empleado paga dicho producto, por ende se registra al ser una salida de dinero.

- El selector de motivos, cuando el empleado tiene permisos de rol encargado, aparece en el selector “AAGREGAR NUEVO GASTO” (véase Figura 3.4.3.6), el cual se habilita un campo de texto extra, para agregar el nuevo nombre del motivo del gasto.
- Tenemos un selector con los diferentes motivos de gastos (véase Figura 3.4.3.7), donde el empleado con rol de encargado ya registró, además, por default la aplicación web ya te entrega algunos por defecto, pero los nuevos que son registrado por el encargado no aparecerán en otros negocios, dado que estos nuevos registros se relación al id del gerente que está relacionado el empleado.
- Una vez ingresada la cantidad del gasto y el motivo, al presionar el botón “Agregar”, se actualiza la tabla de gastos registrados.
- Similar al de pago con terminal (véase Figura 3.4.3.8), tenemos las columnas cantidad, hora y acciones, agregando también la columna de motivo o descripción del gasto.
 - Actualizar: Botón de color verde, en caso de una equivocación en la cantidad o el motivo, se podrá cambiar la cantidad en la columna de monto y el motivo en la columna de descripción, después de cambiar dicha cantidad y/o descripción se presiona el botón verde para actualizar el registro (véase Figura 3.4.3.9).

- Borrar: Botón de color rojo tiene la funcionalidad de eliminar el registro, en caso que no se permita actualizar, o incluso que el gasto se haya rechazado o alguna situación en la que dicho egreso no debe ser contemplado. Solamente nos situamos en la fila del registro a eliminar, presionamos el botón de color rojo, y aceptamos su eliminación (véase Figura 3.4.3.10).
- Corte de caja: Está herramienta realiza un conjunto de las dos anteriores, buscando ser lo más intuitivo y fácil posible de usar, esta herramienta la encontramos en el menú desplegable en “Opciones” y seleccionamos “Corte”, dicho es de uso al finalizar el turno.
 - Tenemos campos de texto junto con iconos representativos el cual nos ayudará a realizar y registrar el corte de caja (véase Figura 3.4.3.11), los datos a llenar son los siguientes:
 - Caja actual: En este campo está diseñado para ingresar la cantidad total que el empleado ha contado en su caja.
 - Total de venta: Este campo se actualiza automáticamente, usando la siguiente fórmula la cual es: caja actual + pagos con tarjeta + total de gasto - caja del turno anterior, además, usa la propiedad readonly que solo es para lectura.
 - Caja del turno anterior: Este campo también se autocompleta de manera automática, evitando que el usuario tenga que buscar dicha información, sin embargo, puede ser editado en caso que al contar la

caja anterior al recibir el turno, este ha sido diferente, en este campo simplemente se actualiza de forma manual escribiendo el número correcto.

- Total de gasto: Este campo muestra la suma total de todos los gastos registrados en el turno del empleado, además, usa la propiedad readonly que solo es para lectura..
- Descripción del gasto: Este campo realiza junta todos los motivos junto con su hora de registro y los muestra, además, usa la propiedad readonly que solo es para lectura..
- Pago con tarjeta: Este campo muestra la suma total de todos los pagos con tarjeta registrados en el turno del empleado, además, usa la propiedad readonly que solo es para lectura..
- Comentarios: Esta diseñado para agregar algún comentario por parte del empleado, para el Santana Elia, este está configurado para mostrar dicho comentario, dado que se tiene el servicio de tiempo aire y uso de tragamonedas, esta información es propio del establecimiento pero también se agrega en el corte de caja.
 - Al finalizar el registro, únicamente presionamos el botón de “Aregar”, y este nos redirige a la ventana principal del empleado, teniendo el corte más reciente a la izquierda.
 - Realizamos un ejemplo de prueba (véase Figura 3.4.3.12) y lo agregamos al sistema (véase Figura 3.4.3.13).

Registrar Pago Terminal

Cantidad del pago

Agregar

Pagos Con Terminal Registrados

No hay gastos registrados.

Figura 3.4.3.1 Interfaz gráfica para el registro de un pago con terminal.

Pagos Con Terminal Registrados

Monto	Hora	Acciones
100	07:30	Editar Borrar

The table data is as follows:

Monto	Hora	Acciones
100	07:30	Editar Borrar

Figura 3.4.3.2 Visualización gráfica al registro un pago con terminal.

Pagos Con Terminal Registrados

Monto	Hora	Acciones
200	07:30	 

Pago con terminal actualizado correctamente.

Figura 3.4.3.3 Actualización del registro de un pago con terminal.

Pagos Con Terminal Registrados

No hay gastos registrados.

Pago con terminal eliminado correctamente.

Figura 3.4.3.4 Eliminación del registro de un pago con terminal.

Registrar Gasto

Cantidad del Gasto

Motivo del Gasto
Seleccione motivo del gasto

Agregar

Gastos Registrados

No hay gastos registrados.

Figura 3.4.3.5 Interfaz gráfica para el registro de un gasto.

Registrar Gasto

Cantidad del Gasto

Motivo del Gasto
AAGREGAR NUEVO GASTO

Nuevo gasto

Agregar

This screenshot shows a graphical user interface for registering a payment. At the top is a large title 'Registrar Gasto'. Below it are three input fields: 'Cantidad del Gasto' (Amount), 'Motivo del Gasto' (Reason), and 'Nuevo gasto' (New expense). The 'Motivo del Gasto' field has a dropdown menu open, showing the option 'AAGREGAR NUEVO GASTO'. A green button labeled 'Agregar' (Add) is centered below the input fields.

Figura 3.4.3.6 Interfaz gráfica para el registro de un pago con terminal.

Registrar Gasto

Cantidad del Gasto
100

Motivo del Gasto
Seleccione motivo del gasto

Seleccione motivo del gasto

AAGREGAR NUEVO GASTO
ABARROTE
AMPER
ATLANTIDA
BARCEL
BASURA
BIMBO

This screenshot shows the same graphical interface as Figure 3.4.3.6, but with a different state. The 'Motivo del Gasto' dropdown is now closed, and its previous value 'AAGREGAR NUEVO GASTO' is displayed. A new dropdown menu titled 'Seleccione motivo del gasto' is open, listing several options: 'AAGREGAR NUEVO GASTO', 'ABARROTE', 'AMPER', 'ATLANTIDA', 'BARCEL', 'BASURA', and 'BIMBO'. The 'AAGREGAR NUEVO GASTO' option is highlighted in blue, indicating it is selected.

Figura 3.4.3.7 Selección de algún concepto para el gasto.

Gastos Registrados			
Monto	Descripción	Hora	Acciones
100	BARCEL	07:43	 

Figura 3.4.3.8 Visualización gráfica al registro de un gasto.

Gastos Registrados			
Monto	Descripción	Hora	Acciones
100	ESCOBAS	07:43	 
Gasto actualizado correctamente.			

Figura 3.4.3.9 Actualización del registro de un gasto.

Gastos Registrados

No hay gastos registrados.

Gasto eliminado correctamente.

Figura 3.4.3.10 Eliminación del registro de un gasto.

Crear Corte

La venta se calcula automáticamente
Se puede dejar caja en numeros negativos

	Caja actual
	Total de venta
	Caja del turno anterior 3000
	Total de Gasto
	Descripción del Gasto
	Pago con Tarjeta
	Comentarios Saldo en sistema: \$ - Bote de recargas: \$ - Bote de monedas: \$

Agregar

Figura 3.4.3.11 Interfaz gráfica para el registro de un corte de caja.

Crear Corte

La venta se calcula automáticamente
Se puede dejar caja en numeros negativos

	Caja actual 3500
	Total de venta 1700.00
	Caja del turno anterior 3000
	Total de Gasto 900
	Descripción del Gasto COCA COLA - \$500 (01:16 AM), BIMBO - \$400 (01:16 AM)
	Pago con Tarjeta 300
	Comentarios Saldo en sistema: \$ - Bote de recargas: \$ - Bote de monedas: \$

Agregar

Figura 3.4.3.12 Ingreso de datos como ejemplo para el registro.

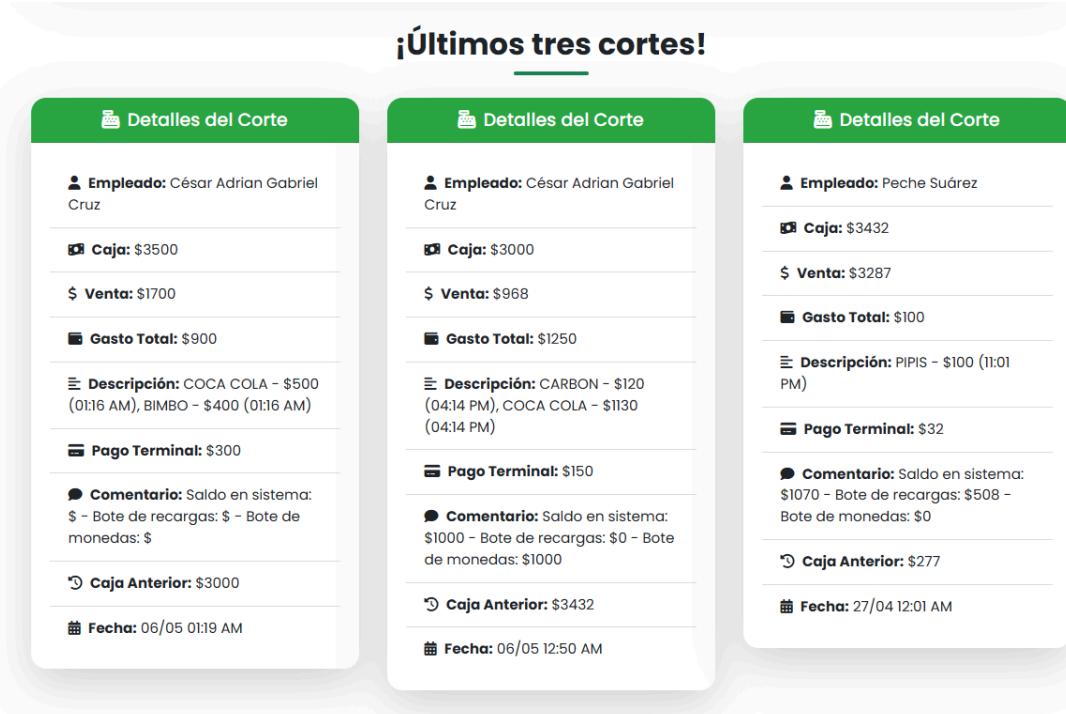


Figura 3.4.3.13 Menú principal donde se muestra el registro de los datos de ejemplo.



Figura 3.4.3.14 Vista del corte realizado en un dispositivo móvil.

3.4.4 Fragmentos de código

Para la explicación del código también se dividirá en tres partes para una mejor comprensión y visualización del mismo:

- Controlador para las herramientas de gastos y pagos con terminal: Este controlador llamado `employee_generate_editDelete.php` (véase Figura 3.4.4.1), realiza las siguientes funciones:
 - `obtenerGastosEmpleados`: Esta función obtiene todos los gastos del empleado que aun no han sido procesados, obteniendo el id, monto, descripción y fecha, estos se organizan por fecha de manera descendente.
 - `obtenerTarjetaEmpleado`: Esta función obtiene todos los pagos con tarjeta del empleado que aun no han sido procesados, obteniendo el id, monto y fecha, estos se organizan por fecha de manera descendente.
 - `obtenerTodasCategoriasGastos`: Esta función obtiene todas las categorías (motivos) de los gastos, obtienen aquellos que están relacionados al id del gerente o aquellos nulos, estos se organizando por nombre de manera ascendente.
 - Métodos post: Al momento de recibir datos mediante método post, para aceptar solicitudes de actualización y eliminación de registros
- Interfaz gráfica pagos con tarjeta: Este archivo llamado `employee_generate_card.php` (véase Figura 3.4.4.2), tiene dos card view, utilizando el controlador para la obtención de de datos, además, implementa una tabla para mostrar los pagos con tarjeta registrados.
- Registro de pagos con tarjeta: Este archivo llamado `employee_generate_card_submit.php` (véase Figura 3.4.4.3), en

el cual se registran los pagos con tarjeta en el cual se obtienen los datos recibidos, guardarlos en variables, verificar que estas variables tengan información, en caso de que no, se regresa un aviso a la interfaz gráfica, si las variables tienen información, se hace el registro en la base de datos.

- Interfaz gráfica gastos: Este archivo llamado employee_generate_purchases.php (véase Figura 3.4.4.4), tiene dos card view, utilizando el controlador para la obtención de datos, además, implementa una tabla para mostrar los pagos con tarjeta registrados.
- Registro de gastos: Este archivo llamado employee_generate_purchases_submit .php (véase Figura 3.4.4.5), en el cual se registran los gastos, primeramente se obtienen los datos recibidos, guardarlos en variables, verificar que estas variables tengan información, en caso de que no, se regresa un aviso a la interfaz gráfica, si las variables tienen información, se hace el registro en la base de datos, además, en caso de recibir el concepto de “AAGREGAR NUEVO GASTO”, antes del registro de dicho gasto, se busca en la base de datos que el nombre nuevo del concepto no exista ya, en caso de existir se redirige a la interfaz gráfica con un mensaje, pero si no existe, este primero se registra en la tabla de categorias_gasto.
- Controlador para la herramienta de corte: Este controlador llamado index_query.php (véase Figura 3.4.4.6), realiza las siguientes funciones:
 - obtenerUltimoCorte: Esta función obtiene los cortes realizados donde el id del gerente esté relacionado con su empleado, sin embargo, esta consulta se limita a un solo resultado por fecha descendente.

- obtenerGastosTotales: Esta función obtiene la suma de todos los gastos y además concatena la descripción del gasto junto con su monto con fecha y hora, donde estos gastos aún no son procesados del empleado en turno.
 - obtenerTerminalTotal: Esta función obtiene la suma total de los pagos con tarjeta de los registros no procesados del empleado en turno
- Interfaz gráfica corte: Este archivo llamado employee_generate_expenses.php (véase Figura 3.4.4.7), obtiene datos de su controlador para ser implementados en ciertos campos de texto, además, tenemos un script que actualiza la venta automáticamente mediante un evento cada que el campo de caja actual se actualiza, para generar la fórmula y tener el resultado de la venta.
- Registro de corte: Este archivo llamado employee_generate_expenses_submit.php (véase Figura 3.4.4.8), en el cual se registran los cortes de caja, primeramente se obtienen los datos recibidos, guardarlos en variables, verificar que estas variables tengan información, en caso de que no, se regresa un aviso a la interfaz gráfica, si las variables tienen información, se hace el registro en la base de datos, después, se actualizan todos los registros de gastos y pagos con tarjeta cambiando su procesado de no a sí, para que en un futuro estos ya no sean contemplados.

```

employee_generate_editDelete.php <-- employee_generate_editDelete.php > ...
CorteCaja > employee_generate_editDelete.php > ...
1  <?php
2
3  include 'includes/conexion_db.php';
4
5  function obtenerGastosEmpleado($pdo)
6  > { ...
7
27 }
28
29  function obtenerTarjetaEmpleado($pdo)
30 > { ...
51 }
52
53  function obtenerTodasCategoriasGastos($pdo)
54 > { ...
78 }
79
80 // Necesario para los mensajes de correctamente
81 if (session_status() === PHP_SESSION_NONE) {
82     session_start();
83 }
84
85 > if ($_SERVER['REQUEST_METHOD'] == 'POST') { ...
166 }
167

```

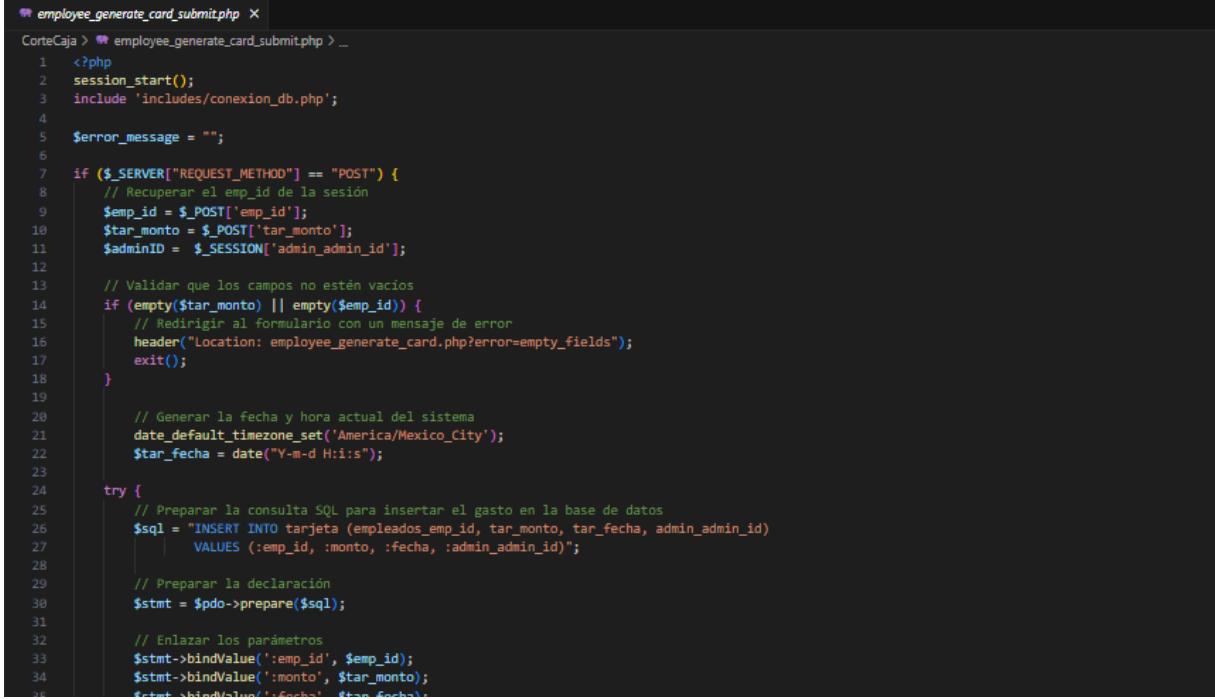
Figura 3.4.4.1 Controlador para las herramientas de gastos y pagos con terminal.

```

employee_generate_card.php <-- employee_generate_card.php > ...
CorteCaja > employee_generate_card.php > ...
1  <?php
2  session_start();
3  $error_message = isset($_SESSION['error_message']) ? $_SESSION['error_message'] : "";
4  unset($_SESSION['error_message']);
5
6  include 'includes/header.php';
7  include 'employee_generate_editDelete.php';
8  include 'includes/conexion_db.php';
9
10 $tarjetaEmp = obtenerTarjetaEmpleado($pdo);
11 >
12
13 <-- Formulario para generar pago tarjeta -->
14 <div class="container my-5">
15     <div class="row justify-content-center">
16         <div class="col-md-8">
17             <div class="card shadow-lg rounded-4 border-0">
18                 <div class="card-body p-4 bg-light">
19                     <h1 class="text-center fw-bold mb-3">Registrar Pago Terminal</h1>
20                     <div class="card-body">
21                         <form action="employee_generate_card_submit.php" method="post">
22                             <!-- Campo Monto de Tarjeta -->
23                             <div class="form-group mb-3">
24                                 <div class="input-group">
25                                     <span class="input-group-text"><i class="fas fa-money-bill-alt"></i></span>
26                                     <div class="form-floating flex-grow-1">
27                                         <input type="number" class="form-control" id="tar_monto" name="tar_monto" required placeholder="Ingresa la cantidad del pago" min="1">
28                                         <label for="tar_monto">Cantidad del pago</label>
29                                     </div>
30                                 </div>
31                         </div>
32
33                         <!-- Alerta Campos Vacios -->
34                         <?php if (isset($_GET['error']) && $_GET['error'] == 'empty_fields'): ?>
35                             <div class="alert alert-danger mt-3" role="alert">
36                                 El campo debe completarse.
37                             </div>
38                         </?php endif; ?>
39
40                         <input type="hidden" name="emp_id" value="<?php echo $_SESSION['emp_id']; ?>">
41                         <!-- Botón -->
42                         <div class="text-center">
43                             <button type="submit" class="btn btn-outline-success btn-lg">
44                                 <i class="fas fa-save me-2"></i> Agregar
45                             </button>
46                         </div>

```

Figura 3.4.4.2 Interfaz gráfica de pagos con tarjeta.

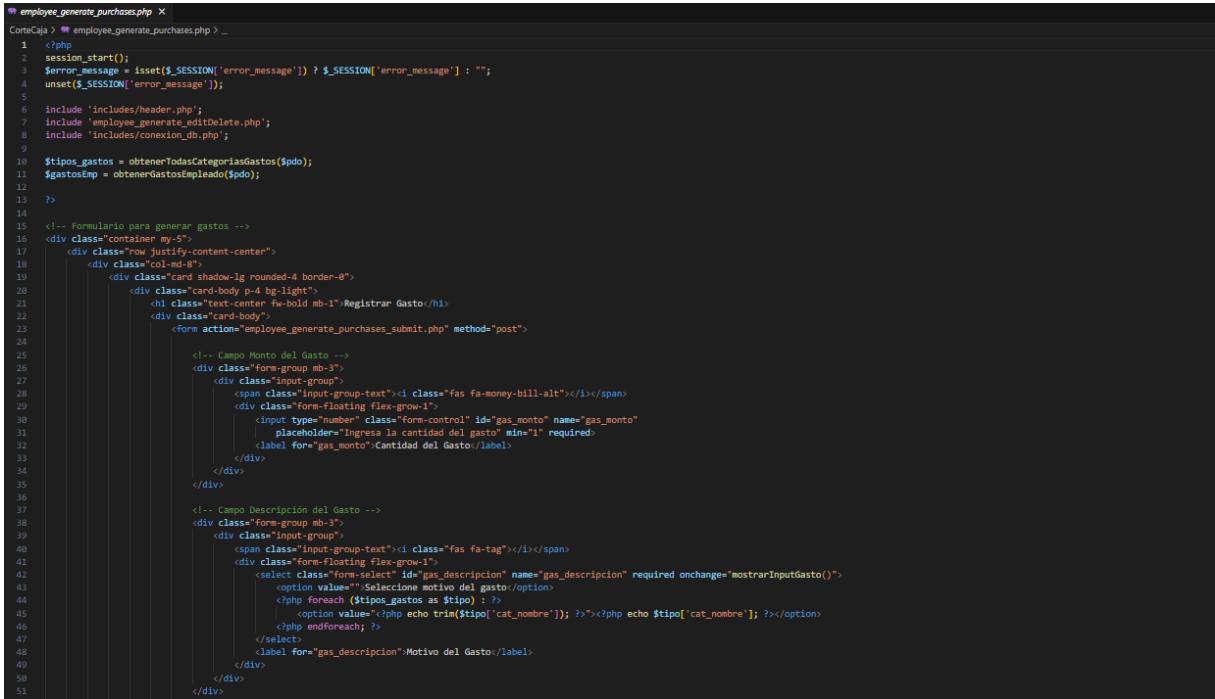


```

employee_generate_card_submit.php
CorteCaja > employee_generate_card_submit.php ...
1  <?php
2  session_start();
3  include 'includes/conexion_db.php';
4
5  $error_message = "";
6
7  if ($_SERVER["REQUEST_METHOD"] == "POST") {
8      // Recuperar el emp_id de la sesión
9      $emp_id = $_POST['emp_id'];
10     $tar_monto = $_POST['tar_monto'];
11     $adminID = $_SESSION['admin_admin_id'];
12
13     // Validar que los campos no estén vacíos
14     if (empty($tar_monto) || empty($emp_id)) {
15         // Redirigir al formulario con un mensaje de error
16         header("Location: employee_generate_card.php?error=empty_fields");
17         exit();
18     }
19
20     // Generar la fecha y hora actual del sistema
21     date_default_timezone_set('America/Mexico_City');
22     $tar_fecha = date("Y-m-d H:i:s");
23
24     try {
25         // Preparar la consulta SQL para insertar el gasto en la base de datos
26         $sql = "INSERT INTO tarjeta (empleados_emp_id, tar_monto, tar_fecha, admin_admin_id)
27               VALUES (:emp_id, :monto, :fecha, :admin_admin_id)";
28
29         // Preparar la declaración
30         $stmt = $pdo->prepare($sql);
31
32         // Enlazar los parámetros
33         $stmt->bindValue(':emp_id', $emp_id);
34         $stmt->bindValue(':monto', $tar_monto);
35         $stmt->bindValue(':fecha', $tar_fecha);
36     }
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51

```

Figura 3.4.4.3 Backend de registro de pagos con tarjeta.



```

employee_generate_purchases.php
CorteCaja > employee_generate_purchases.php ...
1  <?php
2  session_start();
3  $error_message = isset($_SESSION['error_message']) ? $_SESSION['error_message'] : "";
4  unset($_SESSION['error_message']);
5
6  include 'includes/header.php';
7  include 'employee_generate_editDelete.php';
8  include 'includes/conexion_db.php';
9
10  $tipos_gastos = obtenerTodasCategoriasGastos($pdo);
11  $gastosEmp = obtenerGastosEmpleado($pdo);
12
13  >
14
15  <!-- Formulario para generar gastos -->
16  <div class="container my-5">
17      <div class="row justify-content-center">
18          <div class="col-md-8">
19              <div class="card shadow-lg rounded-4 border-0">
20                  <div class="card-body p-4 bg-light">
21                      <h1 class="text-center fw-bold mb-1">Registrar Gasto</h1>
22                      <div class="card-body">
23                          <form action="employee_generate_purchases_submit.php" method="post">
24
25                              <!-- Campo Monto del Gasto -->
26                              <div class="form-group mb-3">
27                                  <div class="input-group">
28                                      <span class="input-group-text" id="gas_monto"><i class="fas fa-money-bill-alt"></i></span>
29                                      <div class="form-control" id="gas_monto" name="gas_monto" type="text" placeholder="Inresa la cantidad del gasto" min="1" required></div>
30                                      <label for="gas_monto">Cantidad del Gasto</label>
31                                  </div>
32                              </div>
33
34
35
36
37
38
39
39
40
41
42
43
44
45
46
47
48
49
50
51

```

Figura 3.4.4.4 Interfaz gráfica de gastos.

```

employee_generate_purchases_submit.php X
CorteCaja > employee_generate_purchases_submit.php > ...
1  <?php
2  session_start();
3  include 'includes/conexion_db.php';
4
5  $error_message = "";
6
7  if ($_SERVER["REQUEST_METHOD"] == "POST") {
8      // Recuperar el emp_id de la sesión
9      $emp_id = $_POST['emp_id'];
10
11     // Recuperar los datos del formulario
12     $gas_monto = $_POST['gas_monto'];
13     $gas_descripcion = $_POST['gas_descripcion'];
14     $gas_nuevo = $_POST['nuevo_gasto'];
15     $gas_nuevo = strtoupper($gas_nuevo); // Convertir a mayúsculas antes de verificar en la BD
16
17     // Validar que los campos no estén vacíos
18     if (empty($gas_monto) || empty($gas_descripcion) || empty($emp_id)) {
19         // Redirigir al formulario con un mensaje de error
20         header("Location: employee_generate_purchases.php?error=empty_fields");
21         exit();
22     }
23
24     $adminID = $_SESSION['admin_admin_id'];
25
26     // Si el usuario seleccionó "AAGREGAR NUEVO GASTO" y escribió un nuevo gasto, lo guardamos en la BD
27     if ($gas_descripcion === "AAGREGAR NUEVO GASTO" && !empty($gas_nuevo)) {
28         try {
29
30             // Verificar si el gasto ya existe
31             $sql = "SELECT COUNT(*) FROM categorias_gasto WHERE cat_nombre = :nuevo_gasto AND admin_id = :admin_id";
32             $stmt = $pdo->prepare($sql);
33             $stmt->bindValue(':nuevo_gasto', $gas_nuevo);
34             $stmt->bindValue(':admin_id', $adminID);
35             $stmt->execute();
36             $existeGasto = $stmt->fetchColumn();
37
38             if ($existeGasto == 0) {
39
40                 // Insertar el nuevo gasto en la tabla categorias_gasto
41                 $sql = "INSERT INTO categorias_gasto (cat_nombre, admin_id) VALUES (:nuevo_gasto, :admin_id)";
42                 $stmt = $pdo->prepare($sql);
43                 $stmt->bindValue(':nuevo_gasto', $gas_nuevo);
44                 $stmt->execute();
45             }
46         } catch (Exception $e) {
47             echo "Error: " . $e->getMessage();
48         }
49     }
50
51     // Redirigir al formulario con un mensaje de éxito
52     header("Location: employee_generate_purchases.php?success=nuevo_gasto");
53     exit();
54 }

```

Figura 3.4.4.5 Backend de registro de gastos.

```

index_query.php X
CorteCaja > index_query.php > ...
104
105     // Función para obtener el último corte
106     function obtenerUltimoCorte($pdo)
107     > { ...
131     }
132
133     function obtenerGastosTotales($pdo, $emp_id)
134     > { ...
152     }
153
154     function obtenerTerminalTotal($pdo, $emp_id)
155     > { ...
172     }
173

```

Figura 3.4.4.6 Controlador para la herramienta de corte.

```

employee_generate_expenses.php
CorteCaja > employee_generate_expenses.php > div.container.my-5 > div.row.justify-content-center > div.col-md-8 > div.card.shadow-lg.rounded-4.border-0 > div.card-body.p-4.bg-light > h6.text-center.text-muted.mb-1
1 <?php
2 session_start();
3 $error_message = isset($_SESSION['error_message']) ? $_SESSION['error_message'] : "";
4 unset($_SESSION['error_message']);
5
6 include 'index_query.php';
7 include 'includes/conexion_db.php';
8 include 'includes/header.php';
9
10 // Obtener el ultimo corte
11 $ultimoCorte = obtenerUltimoCorte($pdo);
12
13 // Obtener la suma de gastos y descripciones
14 $emp_id = $_SESSION['emp_id']; // Obtener el emp_id de la sesión
15 $gastosTotales = obtenerGastosTotales($pdo, $emp_id);
16 $terminalTotal = obtenerTerminalTotal($pdo, $emp_id);
17
18 ?>
19
20 <!-- Formulario para generar corte -->
21 <div class="container my-5">
22 <div class="row justify-content-center">
23 <div class="col-md-8">
24 <div class="card shadow-lg rounded-4 border-0">
25 <div class="card-body p-4 bg-light">
26 <h3 class="text-center mb-2">Crear Corte</h3>
27 <div class="text-center text-muted mb-1">La venta se calcula automáticamente</h6>
28 <div class="text-center text-muted mb-1">Se puede dejar caja en números negativos</h6>
29 <div class="card-body">
30 <form action="employee_generate_expenses_submit.php" method="post">
31
32 <!-- Campo Caja -->
33 <div class="form-group mb-3">
34 <div class="input-group">
35 <span class="input-group-text"><i class="fas fa-cash-register"></i></span>
36 <div class="form-floating flex-grow-1">
37 <input type="number" class="form-control" id="cortecaja" name="cortecaja" placeholder="Caja actual">
38 <label for="cortecaja">Caja actual</label>
39 </div>
40 </div>
41 </div>
42
43 <!-- Campo Venta -->
44 <div class="form-group mb-3">
45 <div class="input-group">
46 <span class="input-group-text"><i class="fas fa-shopping-cart"></i></span>
47 <div class="form-floating flex-grow-1">
48 <input type="number" class="form-control" id="cordeventa" name="cordeventa" placeholder="Total de venta" required min="1" readonly>
49 <label for="cordeventa">Total de venta</label>
50 </div>
51 </div>
52
53 </div>
54
55 // Recuperar el emp_id de la sesión
56 $emp_id = $_POST['emp_id'];
57
58 // Recuperar los datos del formulario
59 $cortecaja = $_POST['cortecaja'];
60 $cordeventa = $_POST['cordeventa'];
61 $cortecajaanterior = $_POST['cortecajaanterior'];
62 $cortegastototal = !empty($_POST['cortegastototal']) ? $_POST['cortegastototal'] : 0;
63 $cortegastodescripcion = !empty($_POST['cortegastodescripcion']) ? $_POST['cortegastodescripcion'] : "Sin descripción";
64 $cortecomentario = !empty($_POST['cortecomentario']) ? $_POST['cortecomentario'] : "Sin comentario";
65 $cortetarjeta = !empty($_POST['cortetarjeta']) ? $_POST['cortetarjeta'] : 0;
66 $adminID = $_SESSION['admin_admin_id'];
67
68 // Validar que los campos no estén vacíos
69 if (empty($cortecaja) || empty($cordeventa) || empty($cortecajaanterior) || empty($emp_id)) {
70 // Redirigir al formulario con un mensaje de error
71 header("location: employee_generate_expenses.php?error=empty_fields");
72 exit();
73 }
74
75 try {
76 // Generar la fecha y hora actual del sistema
77 date_default_timezone_set('America/Mexico_City');
78 $corte_horafecha = date("Y-m-d H:i:s");
79
80 // Preparar la consulta SQL para insertar el corte en la base de datos
81 $sql = "INSERT INTO corte (corte_caja, corte_venta, corte_cajamterior, corte_gasto_total, corte_gasto_descripcion, corte_tarjeta, corte_comentario, corte_horafecha, empleados_emp_id, admin_admin_id)
82 VALUES (:caja, :venta, :cajamterior, :gasto_total, :gasto_descripcion, :tarjeta, :comentario, :horafecha, :emp_id, :admin_id)";
83
84 // Preparar la declaración
85 $stmt = $pdo->prepare($sql);
86
87 // Enlazar los parámetros
88 $stmt->bindValue(':caja', $cortecaja);
89 $stmt->bindValue(':venta', $cordeventa);

```

Figura 3.4.4.7 Intefaz gráfica de corte.

```

employee_generate_expenses_submit.php
CorteCaja > employee_generate_expenses_submit.php > ...
1 <?php
2 session_start();
3 include 'includes/conexion_db.php';
4
5 $error_message = "";
6
7 if ($_SERVER["REQUEST_METHOD"] == "POST") {
8
9 // Recuperar el emp_id de la sesión
10 $emp_id = $_POST['emp_id'];
11
12 // Recuperar los datos del formulario
13 $cortecaja = $_POST['cortecaja'];
14 $cordeventa = $_POST['cordeventa'];
15 $cortecajaanterior = $_POST['cortecajaanterior'];
16 $cortegastototal = !empty($_POST['cortegastototal']) ? $_POST['cortegastototal'] : 0;
17 $cortegastodescripcion = !empty($_POST['cortegastodescripcion']) ? $_POST['cortegastodescripcion'] : "Sin descripción";
18 $cortecomentario = !empty($_POST['cortecomentario']) ? $_POST['cortecomentario'] : "Sin comentario";
19 $cortetarjeta = !empty($_POST['cortetarjeta']) ? $_POST['cortetarjeta'] : 0;
20 $adminID = $_SESSION['admin_admin_id'];
21
22 // Validar que los campos no estén vacíos
23 if (empty($cortecaja) || empty($cordeventa) || empty($cortecajaanterior) || empty($emp_id)) {
24 // Redirigir al formulario con un mensaje de error
25 header("location: employee_generate_expenses.php?error=empty_fields");
26 exit();
27 }
28
29 try {
30 // Generar la fecha y hora actual del sistema
31 date_default_timezone_set('America/Mexico_City');
32 $corte_horafecha = date("Y-m-d H:i:s");
33
34 // Preparar la consulta SQL para insertar el corte en la base de datos
35 $sql = "INSERT INTO corte (corte_caja, corte_venta, corte_cajamterior, corte_gasto_total, corte_gasto_descripcion, corte_tarjeta, corte_comentario, corte_horafecha, empleados_emp_id, admin_admin_id)
36 VALUES (:caja, :venta, :cajamterior, :gasto_total, :gasto_descripcion, :tarjeta, :comentario, :horafecha, :emp_id, :admin_id)";
37
38 // Preparar la declaración
39 $stmt = $pdo->prepare($sql);
40
41 // Enlazar los parámetros
42 $stmt->bindValue(':caja', $cortecaja);
43 $stmt->bindValue(':venta', $cordeventa);
44
45 // Ejecutar la consulta
46 $stmt->execute();
47
48 // Obtener el ID del corte recién insertado
49 $corte_id = $stmt->lastInsertId();
50
51 // Redirigir al formulario con el ID del corte
52 header("location: employee_generate_expenses.php?corte_id=$corte_id");
53 exit();
54 }
55
56 // Cerrar la conexión
57 $pdo->close();
58
59 // Mostrar el mensaje de éxito
60 echo "Corte registrado con éxito!";
61
62 // Salir del script
63 exit();
64
65 // Cerrar la sesión
66 session_destroy();
67
68 // Salir del script
69 exit();
70 }
71
72 // Cerrar la conexión
73 $pdo->close();
74
75 // Salir del script
76 exit();
77
78 // Cerrar la sesión
79 session_destroy();
80
81 // Salir del script
82 exit();
83
84 // Cerrar la conexión
85 $pdo->close();
86
87 // Salir del script
88 exit();
89
90 // Cerrar la sesión
91 session_destroy();
92
93 // Salir del script
94 exit();
95
96 // Cerrar la conexión
97 $pdo->close();
98
99 // Salir del script
100 exit();
101
102 // Cerrar la sesión
103 session_destroy();
104
105 // Salir del script
106 exit();
107
108 // Cerrar la conexión
109 $pdo->close();
110
111 // Salir del script
112 exit();
113
114 // Cerrar la sesión
115 session_destroy();
116
117 // Salir del script
118 exit();
119
120 // Cerrar la conexión
121 $pdo->close();
122
123 // Salir del script
124 exit();
125
126 // Cerrar la sesión
127 session_destroy();
128
129 // Salir del script
130 exit();
131
132 // Cerrar la conexión
133 $pdo->close();
134
135 // Salir del script
136 exit();
137
138 // Cerrar la sesión
139 session_destroy();
140
141 // Salir del script
142 exit();
143
144 // Cerrar la conexión
145 $pdo->close();
146
147 // Salir del script
148 exit();
149
150 // Cerrar la sesión
151 session_destroy();
152
153 // Salir del script
154 exit();
155
156 // Cerrar la conexión
157 $pdo->close();
158
159 // Salir del script
160 exit();
161
162 // Cerrar la sesión
163 session_destroy();
164
165 // Salir del script
166 exit();
167
168 // Cerrar la conexión
169 $pdo->close();
170
171 // Salir del script
172 exit();
173
174 // Cerrar la sesión
175 session_destroy();
176
177 // Salir del script
178 exit();
179
180 // Cerrar la conexión
181 $pdo->close();
182
183 // Salir del script
184 exit();
185
186 // Cerrar la sesión
187 session_destroy();
188
189 // Salir del script
190 exit();
191
192 // Cerrar la conexión
193 $pdo->close();
194
195 // Salir del script
196 exit();
197
198 // Cerrar la sesión
199 session_destroy();
200
201 // Salir del script
202 exit();
203
204 // Cerrar la conexión
205 $pdo->close();
206
207 // Salir del script
208 exit();
209
210 // Cerrar la sesión
211 session_destroy();
212
213 // Salir del script
214 exit();
215
216 // Cerrar la conexión
217 $pdo->close();
218
219 // Salir del script
220 exit();
221
222 // Cerrar la sesión
223 session_destroy();
224
225 // Salir del script
226 exit();
227
228 // Cerrar la conexión
229 $pdo->close();
230
231 // Salir del script
232 exit();
233
234 // Cerrar la sesión
235 session_destroy();
236
237 // Salir del script
238 exit();
239
240 // Cerrar la conexión
241 $pdo->close();
242
243 // Salir del script
244 exit();
245
246 // Cerrar la sesión
247 session_destroy();
248
249 // Salir del script
250 exit();
251
252 // Cerrar la conexión
253 $pdo->close();
254
255 // Salir del script
256 exit();
257
258 // Cerrar la sesión
259 session_destroy();
260
261 // Salir del script
262 exit();
263
264 // Cerrar la conexión
265 $pdo->close();
266
267 // Salir del script
268 exit();
269
270 // Cerrar la sesión
271 session_destroy();
272
273 // Salir del script
274 exit();
275
276 // Cerrar la conexión
277 $pdo->close();
278
279 // Salir del script
280 exit();
281
282 // Cerrar la sesión
283 session_destroy();
284
285 // Salir del script
286 exit();
287
288 // Cerrar la conexión
289 $pdo->close();
290
291 // Salir del script
292 exit();
293
294 // Cerrar la sesión
295 session_destroy();
296
297 // Salir del script
298 exit();
299
300 // Cerrar la conexión
301 $pdo->close();
302
303 // Salir del script
304 exit();
305
306 // Cerrar la sesión
307 session_destroy();
308
309 // Salir del script
310 exit();
311
312 // Cerrar la conexión
313 $pdo->close();
314
315 // Salir del script
316 exit();
317
318 // Cerrar la sesión
319 session_destroy();
320
321 // Salir del script
322 exit();
323
324 // Cerrar la conexión
325 $pdo->close();
326
327 // Salir del script
328 exit();
329
330 // Cerrar la sesión
331 session_destroy();
332
333 // Salir del script
334 exit();
335
336 // Cerrar la conexión
337 $pdo->close();
338
339 // Salir del script
340 exit();
341
342 // Cerrar la sesión
343 session_destroy();
344
345 // Salir del script
346 exit();
347
348 // Cerrar la conexión
349 $pdo->close();
350
351 // Salir del script
352 exit();
353
354 // Cerrar la sesión
355 session_destroy();
356
357 // Salir del script
358 exit();
359
360 // Cerrar la conexión
361 $pdo->close();
362
363 // Salir del script
364 exit();
365
366 // Cerrar la sesión
367 session_destroy();
368
369 // Salir del script
370 exit();
371
372 // Cerrar la conexión
373 $pdo->close();
374
375 // Salir del script
376 exit();
377
378 // Cerrar la sesión
379 session_destroy();
380
381 // Salir del script
382 exit();
383
384 // Cerrar la conexión
385 $pdo->close();
386
387 // Salir del script
388 exit();
389
390 // Cerrar la sesión
391 session_destroy();
392
393 // Salir del script
394 exit();
395
396 // Cerrar la conexión
397 $pdo->close();
398
399 // Salir del script
400 exit();
401
402 // Cerrar la sesión
403 session_destroy();
404
405 // Salir del script
406 exit();
407
408 // Cerrar la conexión
409 $pdo->close();
410
411 // Salir del script
412 exit();
413
414 // Cerrar la sesión
415 session_destroy();
416
417 // Salir del script
418 exit();
419
420 // Cerrar la conexión
421 $pdo->close();
422
423 // Salir del script
424 exit();
425
426 // Cerrar la sesión
427 session_destroy();
428
429 // Salir del script
430 exit();
431
432 // Cerrar la conexión
433 $pdo->close();
434
435 // Salir del script
436 exit();
437
438 // Cerrar la sesión
439 session_destroy();
440
441 // Salir del script
442 exit();
443
444 // Cerrar la conexión
445 $pdo->close();
446
447 // Salir del script
448 exit();
449
450 // Cerrar la sesión
451 session_destroy();
452
453 // Salir del script
454 exit();
455
456 // Cerrar la conexión
457 $pdo->close();
458
459 // Salir del script
460 exit();
461
462 // Cerrar la sesión
463 session_destroy();
464
465 // Salir del script
466 exit();
467
468 // Cerrar la conexión
469 $pdo->close();
470
471 // Salir del script
472 exit();
473
474 // Cerrar la sesión
475 session_destroy();
476
477 // Salir del script
478 exit();
479
480 // Cerrar la conexión
481 $pdo->close();
482
483 // Salir del script
484 exit();
485
486 // Cerrar la sesión
487 session_destroy();
488
489 // Salir del script
490 exit();
491
492 // Cerrar la conexión
493 $pdo->close();
494
495 // Salir del script
496 exit();
497
498 // Cerrar la sesión
499 session_destroy();
500
501 // Salir del script
502 exit();
503
504 // Cerrar la conexión
505 $pdo->close();
506
507 // Salir del script
508 exit();
509
510 // Cerrar la sesión
511 session_destroy();
512
513 // Salir del script
514 exit();
515
516 // Cerrar la conexión
517 $pdo->close();
518
519 // Salir del script
520 exit();
521
522 // Cerrar la sesión
523 session_destroy();
524
525 // Salir del script
526 exit();
527
528 // Cerrar la conexión
529 $pdo->close();
530
531 // Salir del script
532 exit();
533
534 // Cerrar la sesión
535 session_destroy();
536
537 // Salir del script
538 exit();
539
540 // Cerrar la conexión
541 $pdo->close();
542
543 // Salir del script
544 exit();
545
546 // Cerrar la sesión
547 session_destroy();
548
549 // Salir del script
550 exit();
551
552 // Cerrar la conexión
553 $pdo->close();
554
555 // Salir del script
556 exit();
557
558 // Cerrar la sesión
559 session_destroy();
560
561 // Salir del script
562 exit();
563
564 // Cerrar la conexión
565 $pdo->close();
566
567 // Salir del script
568 exit();
569
570 // Cerrar la sesión
571 session_destroy();
572
573 // Salir del script
574 exit();
575
576 // Cerrar la conexión
577 $pdo->close();
578
579 // Salir del script
580 exit();
581
582 // Cerrar la sesión
583 session_destroy();
584
585 // Salir del script
586 exit();
587
588 // Cerrar la conexión
589 $pdo->close();
590
591 // Salir del script
592 exit();
593
594 // Cerrar la sesión
595 session_destroy();
596
597 // Salir del script
598 exit();
599
600 // Cerrar la conexión
601 $pdo->close();
602
603 // Salir del script
604 exit();
605
606 // Cerrar la sesión
607 session_destroy();
608
609 // Salir del script
610 exit();
611
612 // Cerrar la conexión
613 $pdo->close();
614
615 // Salir del script
616 exit();
617
618 // Cerrar la sesión
619 session_destroy();
620
621 // Salir del script
622 exit();
623
624 // Cerrar la conexión
625 $pdo->close();
626
627 // Salir del script
628 exit();
629
630 // Cerrar la sesión
631 session_destroy();
632
633 // Salir del script
634 exit();
635
636 // Cerrar la conexión
637 $pdo->close();
638
639 // Salir del script
640 exit();
641
642 // Cerrar la sesión
643 session_destroy();
644
645 // Salir del script
646 exit();
647
648 // Cerrar la conexión
649 $pdo->close();
650
651 // Salir del script
652 exit();
653
654 // Cerrar la sesión
655 session_destroy();
656
657 // Salir del script
658 exit();
659
660 // Cerrar la conexión
661 $pdo->close();
662
663 // Salir del script
664 exit();
665
666 // Cerrar la sesión
667 session_destroy();
668
669 // Salir del script
670 exit();
671
672 // Cerrar la conexión
673 $pdo->close();
674
675 // Salir del script
676 exit();
677
678 // Cerrar la sesión
679 session_destroy();
680
681 // Salir del script
682 exit();
683
684 // Cerrar la conexión
685 $pdo->close();
686
687 // Salir del script
688 exit();
689
690 // Cerrar la sesión
691 session_destroy();
692
693 // Salir del script
694 exit();
695
696 // Cerrar la conexión
697 $pdo->close();
698
699 // Salir del script
700 exit();
701
702 // Cerrar la sesión
703 session_destroy();
704
705 // Salir del script
706 exit();
707
708 // Cerrar la conexión
709 $pdo->close();
710
711 // Salir del script
712 exit();
713
714 // Cerrar la sesión
715 session_destroy();
716
717 // Salir del script
718 exit();
719
720 // Cerrar la conexión
721 $pdo->close();
722
723 // Salir del script
724 exit();
725
726 // Cerrar la sesión
727 session_destroy();
728
729 // Salir del script
730 exit();
731
732 // Cerrar la conexión
733 $pdo->close();
734
735 // Salir del script
736 exit();
737
738 // Cerrar la sesión
739 session_destroy();
740
741 // Salir del script
742 exit();
743
744 // Cerrar la conexión
745 $pdo->close();
746
747 // Salir del script
748 exit();
749
750 // Cerrar la sesión
751 session_destroy();
752
753 // Salir del script
754 exit();
755
756 // Cerrar la conexión
757 $pdo->close();
758
759 // Salir del script
760 exit();
761
762 // Cerrar la sesión
763 session_destroy();
764
765 // Salir del script
766 exit();
767
768 // Cerrar la conexión
769 $pdo->close();
770
771 // Salir del script
772 exit();
773
774 // Cerrar la sesión
775 session_destroy();
776
777 // Salir del script
778 exit();
779
780 // Cerrar la conexión
781 $pdo->close();
782
783 // Salir del script
784 exit();
785
786 // Cerrar la sesión
787 session_destroy();
788
789 // Salir del script
790 exit();
791
792 // Cerrar la conexión
793 $pdo->close();
794
795 // Salir del script
796 exit();
797
798 // Cerrar la sesión
799 session_destroy();
800
801 // Salir del script
802 exit();
803
804 // Cerrar la conexión
805 $pdo->close();
806
807 // Salir del script
808 exit();
809
810 // Cerrar la sesión
811 session_destroy();
812
813 // Salir del script
814 exit();
815
816 // Cerrar la conexión
817 $pdo->close();
818
819 // Salir del script
820 exit();
821
822 // Cerrar la sesión
823 session_destroy();
824
825 // Salir del script
826 exit();
827
828 // Cerrar la conexión
829 $pdo->close();
830
831 // Salir del script
832 exit();
833
834 // Cerrar la sesión
835 session_destroy();
836
837 // Salir del script
838 exit();
839
840 // Cerrar la conexión
841 $pdo->close();
842
843 // Salir del script
844 exit();
845
846 // Cerrar la sesión
847 session_destroy();
848
849 // Salir del script
850 exit();
851
852 // Cerrar la conexión
853 $pdo->close();
854
855 // Salir del script
856 exit();
857
858 // Cerrar la sesión
859 session_destroy();
860
861 // Salir del script
862 exit();
863
864 // Cerrar la conexión
865 $pdo->close();
866
867 // Salir del script
868 exit();
869
870 // Cerrar la sesión
871 session_destroy();
872
873 // Salir del script
874 exit();
875
876 // Cerrar la conexión
877 $pdo->close();
878
879 // Salir del script
880 exit();
881
882 // Cerrar la sesión
883 session_destroy();
884
885 // Salir del script
886 exit();
887
888 // Cerrar la conexión
889 $pdo->close();
890
891 // Salir del script
892 exit();
893
894 // Cerrar la sesión
895 session_destroy();
896
897 // Salir del script
898 exit();
899
900 // Cerrar la conexión
901 $pdo->close();
902
903 // Salir del script
904 exit();
905
906 // Cerrar la sesión
907 session_destroy();
908
909 // Salir del script
910 exit();
911
912 // Cerrar la conexión
913 $pdo->close();
914
915 // Salir del script
916 exit();
917
918 // Cerrar la sesión
919 session_destroy();
920
921 // Salir del script
922 exit();
923
924 // Cerrar la conexión
925 $pdo->close();
926
927 // Salir del script
928 exit();
929
930 // Cerrar la sesión
931 session_destroy();
932
933 // Salir del script
934 exit();
935
936 // Cerrar la conexión
937 $pdo->close();
938
939 // Salir del script
940 exit();
941
942 // Cerrar la sesión
943 session_destroy();
944
945 // Salir del script
946 exit();
947
948 // Cerrar la conexión
949 $pdo->close();
950
951 // Salir del script
952 exit();
953
954 // Cerrar la sesión
955 session_destroy();
956
957 // Salir del script
958 exit();
959
960 // Cerrar la conexión
961 $pdo->close();
962
963 // Salir del script
964 exit();
965
966 // Cerrar la sesión
967 session_destroy();
968
969 // Salir del script
970 exit();
971
972 // Cerrar la conexión
973 $pdo->close();
974
975 // Salir del script
976 exit();
977
978 // Cerrar la sesión
979 session_destroy();
980
981 // Salir del script
982 exit();
983
984 // Cerrar la conexión
985 $pdo->close();
986
987 // Salir del script
988 exit();
989
990 // Cerrar la sesión
991 session_destroy();
992
993 // Salir del script
994 exit();
995
996 // Cerrar la conexión
997 $pdo->close();
998
999 // Salir del script
1000 exit();
1001
1002 // Cerrar la sesión
1003 session_destroy();
1004
1005 // Salir del script
1006 exit();
1007
1008 // Cerrar la conexión
1009 $pdo->close();
1010
1011 // Salir del script
1012 exit();
1013
1014 // Cerrar la sesión
1015 session_destroy();
1016
1017 // Salir del script
1018 exit();
1019
1020 // Cerrar la conexión
1021 $pdo->close();
1022
1023 // Salir del script
1024 exit();
1025
1026 // Cerrar la sesión
1027 session_destroy();
1028
1029 // Salir del script
1030 exit();
1031
1032 // Cerrar la conexión
1033 $pdo->close();
1034
1035 // Salir del script
1036 exit();
1037
1038 // Cerrar la sesión
1039 session_destroy();
1040
1041 // Salir del script
1042 exit();
1043
1044 // Cerrar la conexión
1045 $pdo->close();
1046
1047 // Salir del script
1048 exit();
1049
1050 // Cerrar la sesión
1051 session_destroy();
1052
1053 // Salir del script
1054 exit();
1055
1056 // Cerrar la conexión
1057 $pdo->close();
1058
1059 // Salir del script
1060 exit();
1061
1062 // Cerrar la sesión
1063 session_destroy();
1064
1065 // Salir del script
1066 exit();
1067
1068 // Cerrar la conexión
1069 $pdo->close();
1070
1071 // Salir del script
1072 exit();
1073
1074 // Cerrar la sesión
1075 session_destroy();
1076
1077 // Salir del script
1078 exit();
1079
1080 // Cerrar la conexión
1081 $pdo->close();
1082
1083 // Salir del script
1084 exit();
1085
1086 // Cerrar la sesión
1087 session_destroy();
1088
1089 // Salir del script
1090 exit();
1091
1092 // Cerrar la conexión
1093 $pdo->close();
1094
1095 // Salir del script
1096 exit();
1097
1098 // Cerrar la sesión
1099 session_destroy();
1100
1101 // Salir del script
1102 exit();
1103
1104 // Cerrar la conexión
1105 $pdo->close();
1106
1107 // Salir del script
1108 exit();
1109
1110 // Cerrar la sesión
1111 session_destroy();
1112
1113 // Salir del script
1114 exit();
1115
1116 // Cerrar la conexión
1117 $pdo->close();
1118
1119 // Salir del script
1120 exit();
1121
1122 // Cerrar la sesión
1123 session_destroy();
1124
1125 // Salir del script
1126 exit();
1127
1128 // Cerrar la conexión
1129 $pdo->close();
1130
1131 // Salir del script
1132 exit();
1133
1134 // Cerrar la sesión
1135 session_destroy();
1136
1137 // Salir del script
1138 exit();
1139
1140 // Cerrar la conexión
1141 $pdo->close();
1142
1143 // Salir del script
1144 exit();
1145
1146 // Cerrar la sesión
1147 session_destroy();
1148
1149 // Salir del script
1150 exit();
1151
1152 // Cerrar la conexión
1153 $pdo->close();
1154
1155 // Salir del script
1156 exit();
1157
1158 // Cerrar la sesión
1159 session_destroy();
1160
1161 // Salir del script
1162 exit();
1163
1164 // Cerrar la conexión
1165 $pdo->close();
1166
1167 // Salir del script
1168 exit();
1169
1170 // Cerrar la sesión
1171 session_destroy();
1172
1173 // Salir del script
1174 exit();
1175
1176 // Cerrar la conexión
1177 $pdo->close();
1178
1179 // Salir del script
1180 exit();
1181
1182 // Cerrar la sesión
1183 session_destroy();
1184
1185 // Salir del script
1186 exit();
1187
1188 // Cerrar la conexión
1189 $pdo->close();
1190
1191 // Salir del script
1192 exit();
1193
1194 // Cerrar la sesión
1195 session_destroy();
1196
1197 // Salir del script
1198 exit();
1199
1200 // Cerrar la conexión
1201 $pdo->close();
1202
1203 // Salir del script
1204 exit();
1205
1206 // Cerrar la sesión
1207 session_destroy();
1208
1209 // Salir del script
1210 exit();
1211
1212 // Cerrar la conexión
1213 $pdo->close();
1214
1215 // Salir del script
1216 exit();
1217
1218 // Cerrar la sesión
1219 session_destroy();
1220
1221 // Salir del script
1222 exit();
1223
1224 // Cerrar la conexión
1225 $pdo->close();
1226
1227 // Salir del script
1228 exit();
1229
1230 // Cerrar la sesión
1231 session_destroy();
1232
1233 // Salir del script
1234 exit();
1235
1236 // Cerrar la conexión
1237 $pdo->close();
1238
1239 // Salir del script
1240 exit();
1241
1242 // Cerrar la sesión
1243 session_destroy();
1244
1245 // Salir del script
1246 exit();
1247
1248 // Cerrar la conexión
1249 $pdo->close();
1250
1251 // Salir del script
1252 exit();
1253
1254 // Cerrar la sesión
1255 session_destroy();
1256
1257 // Salir del script
1258 exit();
1259
1260 // Cerrar la conexión
1261 $pdo->close();
1262
1263 // Salir del script
1264 exit();
1265
1266 // Cerrar la sesión
1267 session_destroy();
1268
1269 // Salir del script
1270 exit();
1271
1272 // Cerrar la conexión
1273 $pdo->close();
1274
1275 // Salir del script
1276 exit();
1277
1278 // Cerrar la sesión
1279 session_destroy();
1280
1281 // Salir del script
1282 exit();
1283
1284 // Cerrar la conexión
1285 $pdo->close();
1286
1287 // Salir del script
1288 exit();
1289
1290 // Cerrar la sesión
1291 session_destroy();
1292
1293 // Salir del script
1294 exit();
1295
1296 // Cerrar la conexión
1297 $pdo->close();
1298
1299 // Salir del script
1300 exit();
1301
1302 // Cerrar la sesión
1303 session_destroy();
1304
1305 // Salir del script
1306 exit();
1307
1308 // Cerrar la conexión
1309 $pdo->close();
1310
1311 // Salir del script
1312 exit();
1313
1314 // Cerrar la sesión
1315 session_destroy();
1316
1317 // Salir del script
1318 exit();
1319
1320 // Cerrar la conexión
1321 $pdo->close();
1322
1323 // Salir del script
1324 exit();
1325
1326 // Cerrar la sesión
1327 session_destroy();
1328
1329 // Salir del script
1330 exit();
1331
1332 // Cerrar la conexión
1333 $pdo->close();
1334
1335 // Salir del script
1336 exit();
1337
1338 // Cerrar la sesión
1339 session_destroy();
1340
1341 // Salir del script
1342 exit();
1343
1344 // Cerrar la conexión
1345 $pdo->close();
1346
1347 // Salir del script
1348 exit();
1349
1350 // Cerrar la sesión
1351 session_destroy();
1352
1353 // Salir del script
1354 exit();
1355
1356 // Cerrar la conexión
1357 $pdo->close();
1358
1359 // Salir del script
1360 exit();
1361
1362 // Cerrar la sesión
1363 session_destroy();
1364
1365 // Salir del script
1366 exit();
1367
1368 // Cerrar la conexión
1369 $pdo->close();
1370
1371 // Salir del script
1372 exit();
1373
1374 // Cerrar la sesión
1375 session_destroy();
1376
1377 // Salir del script
1378 exit();
1379
1380 // Cerrar la conexión
1381 $pdo->close();
1382
1383 // Salir del script
1384 exit();
1385
1386 // Cerrar la sesión
1387 session_destroy();
1388
1389 // Salir del script
1390 exit();
1391
1392 // Cerrar la conexión
1393 $pdo->close();
1394
1395 // Sal
```

3.5 Consultar historial de cortes realizados

3.5.1 Descripción del caso de uso

La descripción del caso de uso es consultar historial de cortes realizados, al logear el usuario puede ver los tres cortes de caja realizados más recientes, sin embargo, el caso de uso tiene la finalidad en caso de requerir información con mayor antigüedad, el usuario pueda observar, pero este se divide para los dos actores, lo cual se explica a continuación:

- Para ambos usuarios, empleados y gerentes, podrán observar los últimos tres cortes realizados más recientes alloguearse (véase la Figura 3.5.1.1) .
- Empleados: Los empleados podrán contar con la opción “Historial” (véase la Figura 3.5.1.2) para consultar los 8 cortes de caja realizados más recientes.
- Gerentes: Los gerentes podrán contar con la opción “Historial” (véase la Figura 3.5.1.3), a diferencia del empleado, el gerente tendrá dos filtros (véase la Figura 3.5.1.4) el cual podrá seleccionar ver los 8, 20 o 50 cortes de caja realizados más recientes o seleccionar una fecha en específica, este filtro es mostrado únicamente al gerente debido a que el empleado no requiere esta información, sin embargo, para el gerente puede ser necesario, en especial el filtro de los 8 cortes realizados más recientes. Para el uso de este, solo es cuestión de presionar los botones del filtro correspondiente, para el caso de los número ya por defecto solo es presionar, pero en el tema del uso de fecha específica, al presionarlo se abrirá el calendario seleccionando por defecto el día actual de la consulta, seleccionamos el dia a consultar, presionamos el botón “Filtrar por fecha” y este mostrará los cortes realizado en dicha fecha (véase la Figura 3.5.1.5).

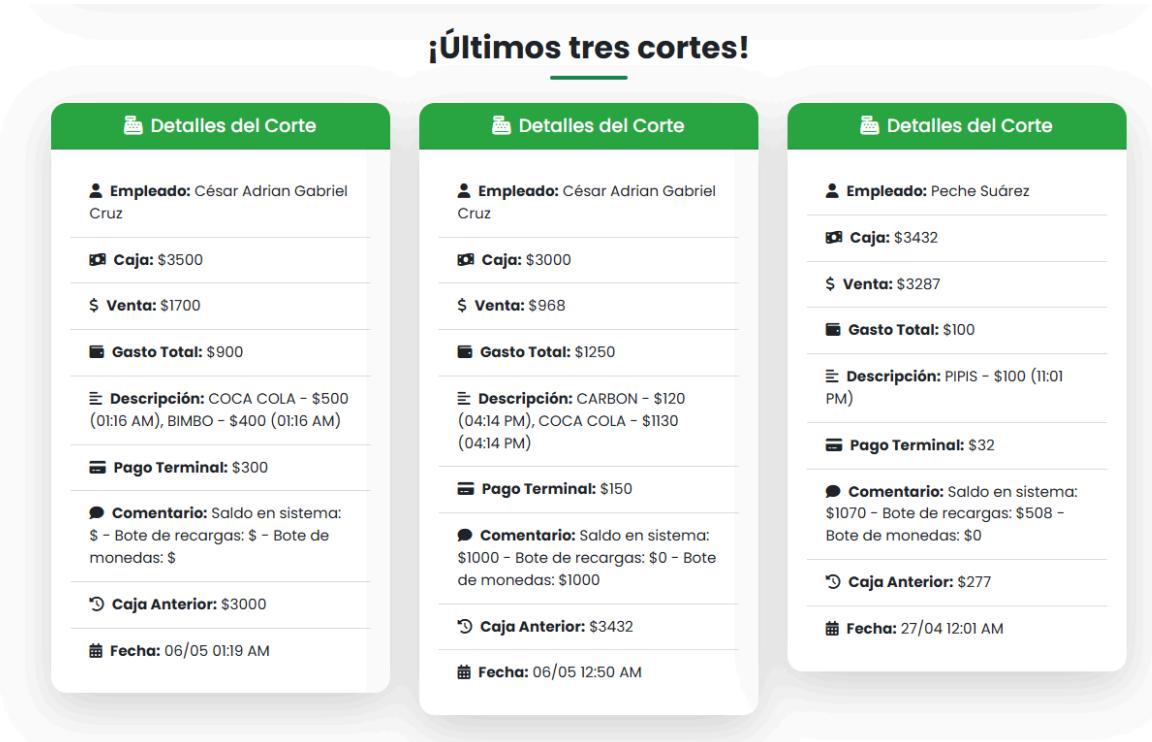


Figura 3.5.1.1 Menú principal donde se muestra el registro de los tres últimos cortes realizados.



Figura 3.5.1.2 Muestra la opción Historial en la barra de herramientas del empleado.

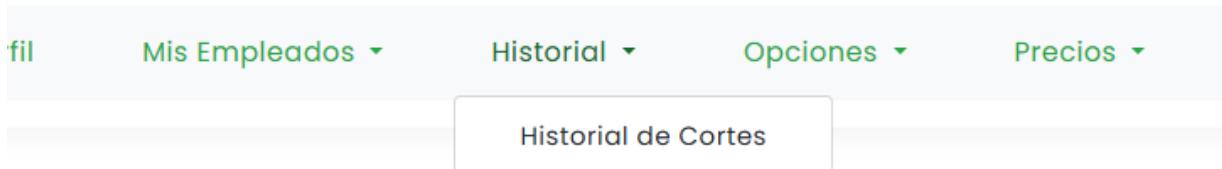


Figura 3.5.1.3 Muestra la opción Historial en la barra de herramientas del gerente.

¡Últimos 8 cortes!

El corte #1 es el más reciente, el #8 es el más antiguo.

Últimos 8

Últimos 20

Últimos 50

dd/mm/aaaa



Filtrar por fecha

Figura 3.5.1.4 Muestra las opciones para filtrar el total de cortes realizados más recientes o una fecha en específico, en la imagen está seleccionado el número 8.

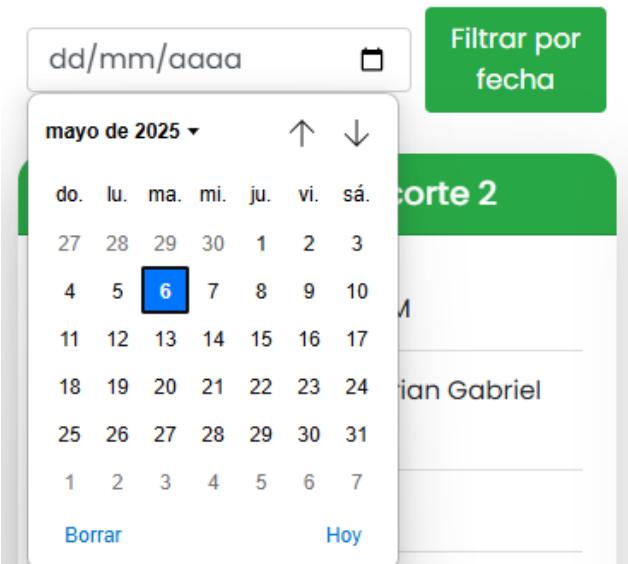


Figura 3.5.1.5 Selector de fecha específica.

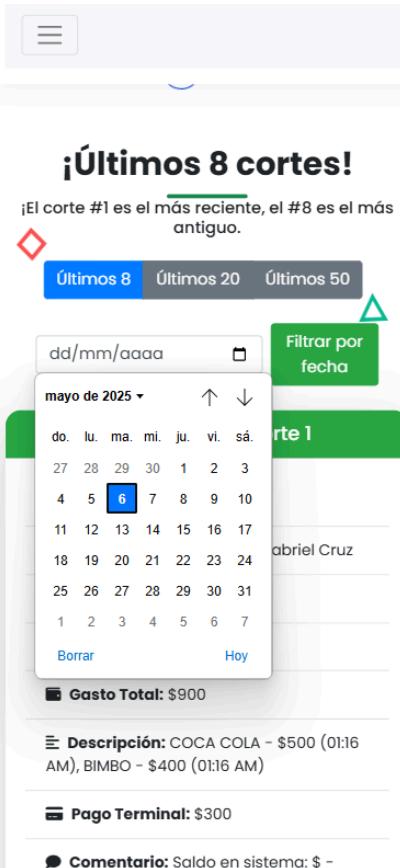


Figura 3.5.1.6 Vista de filtrados de un dispositivo móvil.

3.5.2 Modelo de datos

Para lograr obtener el historial de los cortes realizados, son necesarias tres tablas, las cuales son admin, empleados y corte (véase Figura 3.5.2.1), las cuales se explican a continuación:

- Admin: Esta tabla como se ha visto esta como clave foránea en todas las tablas, debido a la separación de futuros nuevos gerentes.
- Empleados: El empleado es el responsable de realizar el registro de su corte, esto se apoya con el uso de las tablas gastos y terminal para la realización del corte de caja, donde al realizarlo los datos se guardan en la tabla de corte.

- Corte: Es la tabla donde se guardarán los datos del corte de caja, usando los datos necesarios para un guardado de información correcta y completa.

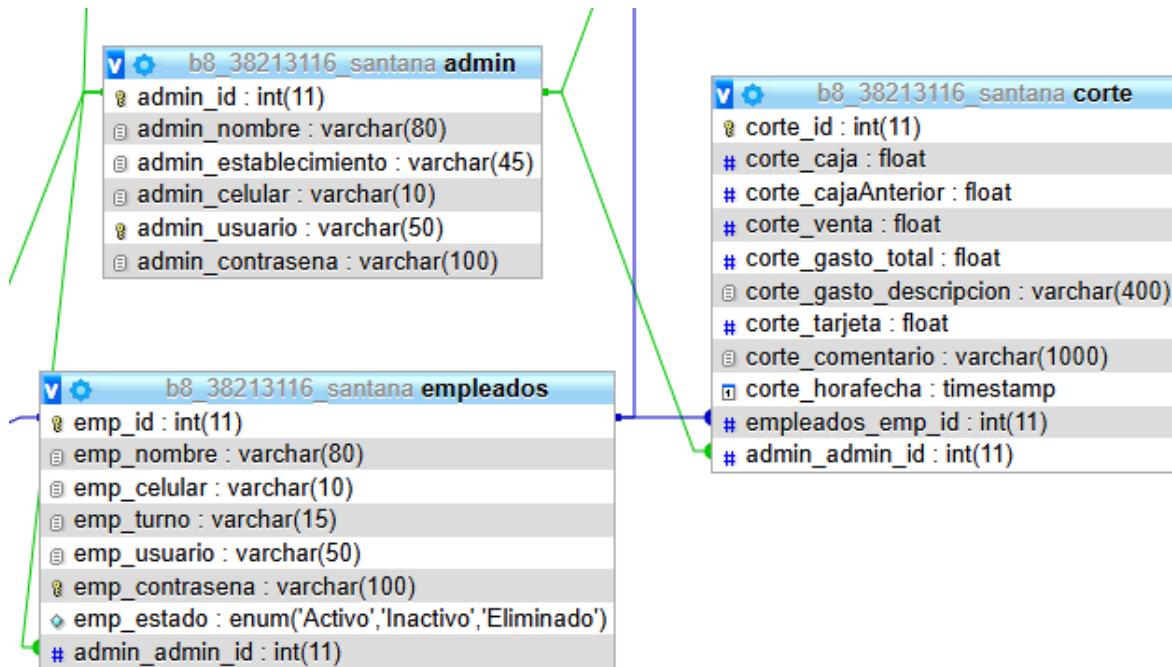


Figura 3.5.2.1 Las tablas admin, empleados y corte, nos ayuda a la elaboración de un historial.

3.5.3 Diseño de la interfaz

Para observar un historial reciente de cortes realizados, nos basaremos en el apartado del gerente, el cual mantiene un filtro para la consulta en número de historial, estas cantidades son en 8, 20 y 50, además de contar con un selector de fecha específica, para identificar los cortes en reciente o antiguo, se menciona que el corte número 1 es el más reciente, mientras que conforme el número va aumentando, el corte será más antiguo, además, así como la lectura que va de izquierda a derecha y de arriba hacia abajo, se puede interpretar los cortes de la misma manera, el más reciente comienza en la parte superior izquierda y conforme avanza a la derecha y hacia abajo, va teniendo mayor antigüedad.

En Santana Elia, es normal consultar tres cortes realizados con antigüedad, sin embargo, casi nunca se revisa más allá del quinto corte con antigüedad, pero por estética, se configuró 8 como opción predeterminada tanto por la antigüedad que supera los 5, y por la estética visual al ser 3 cortes realizados por fila (véase Figura 3.5.3.1). Además, de ser necesario el uso del filtro de fecha específica cumple con su función (véase Figura 3.5.3.2).

¡Últimos 8 cortes!

El corte #1 es el más reciente, el #8 es el más antiguo.

Últimos 8 Últimos 20 Últimos 50

dd/mm/aaaa Filtrar por fecha

Detalles del corte 1	Detalles del corte 2	Detalles del corte 3
Fecha: 06/05 01:19 AM Empleado: César Adrian Gabriel Cruz Caja: \$3500 Venta: \$1700 Gasto Total: \$900 Descripción: COCA COLA - \$500 (01:16 AM), BIMBO - \$400 (01:16 AM) Pago Terminal: \$300 Comentario: Saldo en sistema: \$ - Bote de recargas: \$ - Bote de monedas: \$ Caja Anterior: \$3000	Fecha: 06/05 12:50 AM Empleado: César Adrian Gabriel Cruz Caja: \$3000 Venta: \$968 Gasto Total: \$1250 Descripción: CARBON - \$120 (04:14 PM), COCA COLA - \$1130 (04:14 PM) Pago Terminal: \$150 Comentario: Saldo en sistema: \$1000 - Bote de recargas: \$0 - Bote de monedas: \$1000 Caja Anterior: \$3432	Fecha: 27/04 12:01 AM Empleado: Peche Suárez Caja: \$3432 Venta: \$3287 Gasto Total: \$100 Descripción: PIPIS - \$100 (11:01 PM) Pago Terminal: \$32 Comentario: Saldo en sistema: \$1070 - Bote de recargas: \$508 - Bote de monedas: \$0 Caja Anterior: \$277

Figura 3.5.3.1 Se observa un historial de cortes realizados.

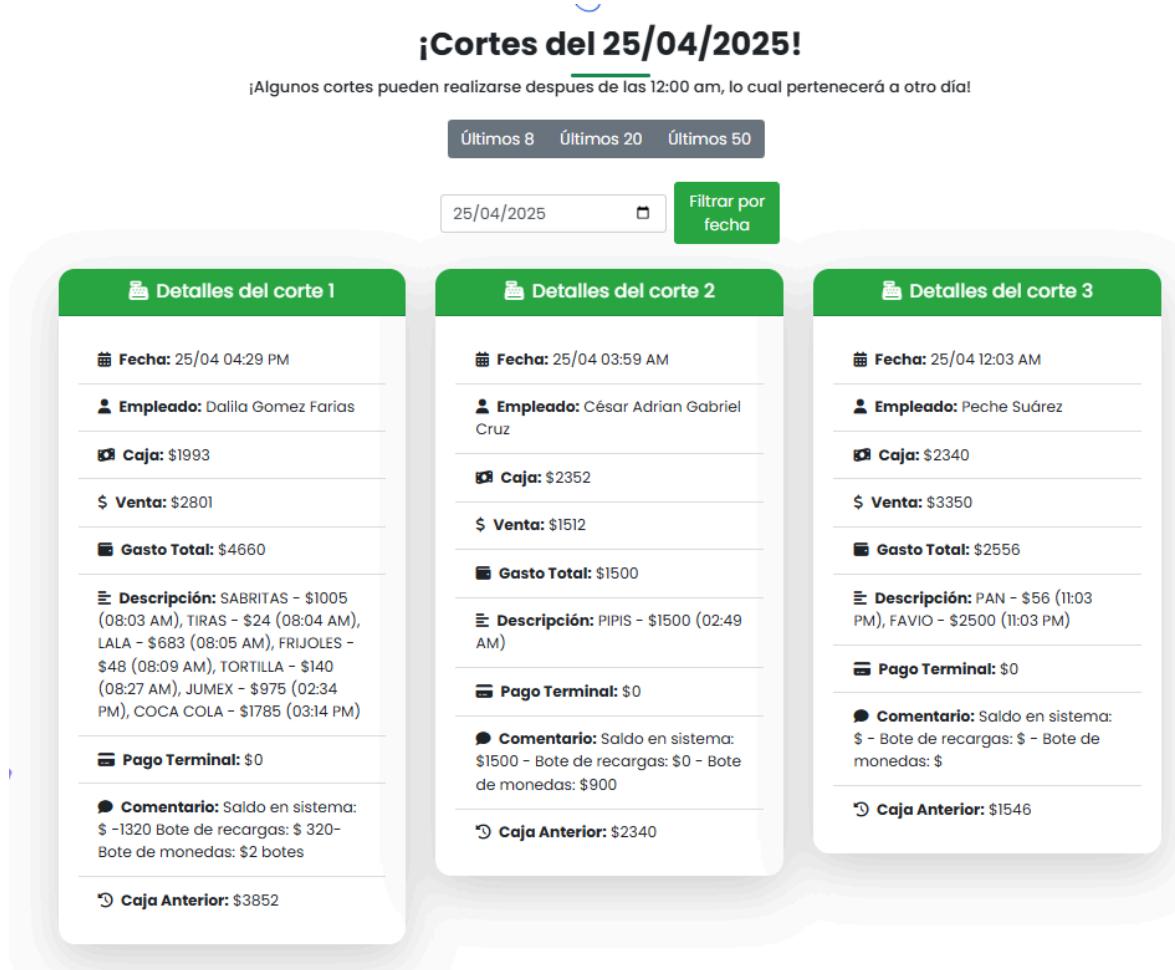


Figura 3.5.3.2 Uso del filtrado por fecha específica.

3.5.4 Fragmentos de código

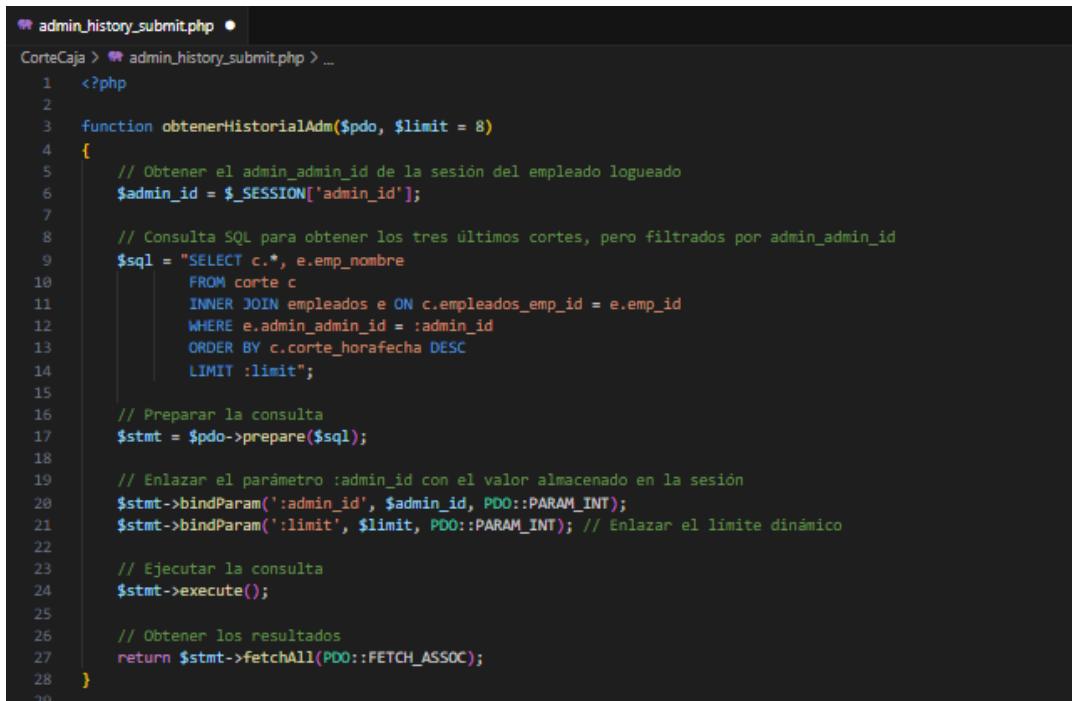
A continuación se explicara parte del código, el cual de la misma manera los explicaremos individualmente:

- En el backend, tenemos el archivo `admin_history_submit.php`, el cual funcionara para almacenar las consultas próximas mediante functions:
 - Comenzamos con la function `obtenerHistorialAdmin()` (véase Figura 3.5.4.1), el cual recibe dos parámetros, la conexión a la base de datos y el límite de corte que son 8, 20 o 50, como se observa en el uso de las tablas

involucradas, la consulta obtiene todos los datos de la tabla de corte y el nombre del empleado de la tabla empleado, se involucran ambas tablas, con la condición que en ambas tablas aparezca el mismo id del gerente, el resultado se ordenará por fecha descendente, donde solo se muestran los primeros 8 como predeterminado, el cual después mediante el filtro, se podrá cambiar el límite.

- A diferencia del historial mostrado al empleado, se basa en lo mismo, solo que el empleado no tiene la característica de consultar como límite 20 y 50 cortes realizados recientemente, para el empleado se mantiene únicamente el límite predeterminado que son 8 (véase Figura 3.5.2.2).
- También otro filtrado para el gerente es la función obtenerHistorialPorFecha (véase Figura 3.5.2.5), ésta como su nombre lo dice busca específicamente por la fecha indicada.
- En la interfaz, tenemos el archivo admin_history.php (véase Figura 3.5.4.4), en el cual, se crearon cards para que la visualización fuese atractiva y simple de entender para el gerente, mostrando los datos más relevantes dentro de un corte, para esto, en el código comenzamos con la asignación de mostrar un límite de 8 cortes realizados recientemente, este número se envía a un function dentro del controlador en el backend el cual se explicará después, una vez se realice la conexión con el controlador y se obtengan los datos, se mostrará el título y subtítulo, debajo del filtro para seleccionar el número total de cortes realizados para ser mostrados, finalizamos con mostrar los datos obtenidos de la function en un card, el cual cada corte realizado será mostrado dentro de uno, ordenado de la manera explicada anteriormente. A

su vez, el selector de fecha específica también envía mediante post al submit con la finalidad de realizar su función correspondiente.

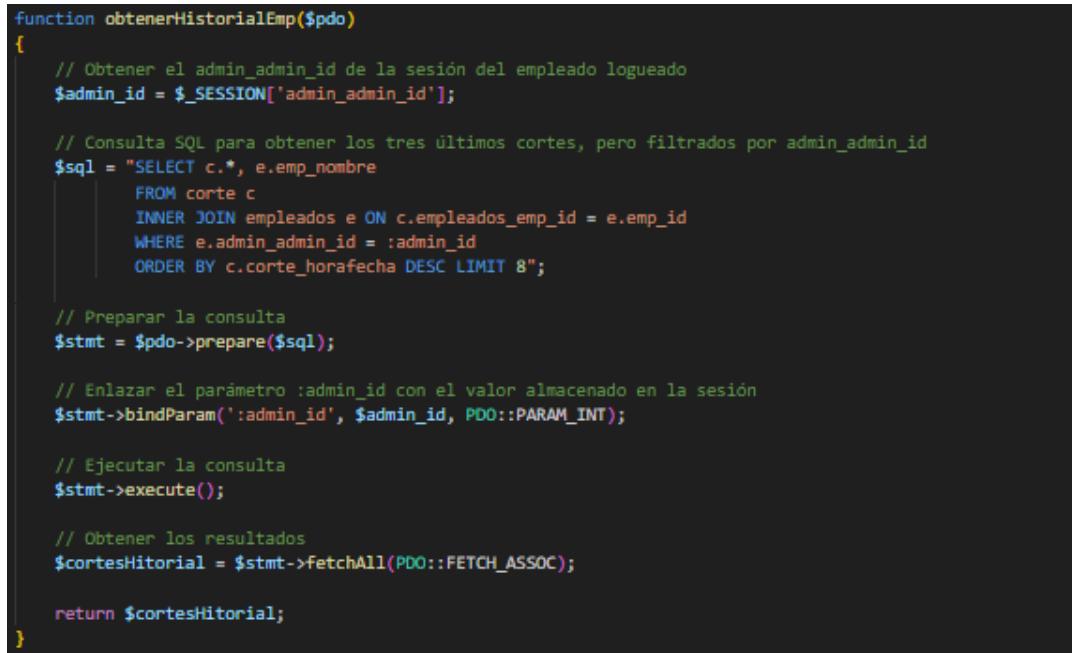


```

1 <?php
2
3 function obtenerHistorialAdm($pdo, $limit = 8)
4 {
5     // Obtener el admin_admin_id de la sesión del empleado logueado
6     $admin_id = $_SESSION['admin_id'];
7
8     // Consulta SQL para obtener los tres últimos cortes, pero filtrados por admin_admin_id
9     $sql = "SELECT c.*, e.emp_nombre
10           FROM corte c
11             INNER JOIN empleados e ON c.empleados_emp_id = e.emp_id
12             WHERE e.admin_admin_id = :admin_id
13             ORDER BY c.corte_horafecha DESC
14             LIMIT :limit";
15
16     // Preparar la consulta
17     $stmt = $pdo->prepare($sql);
18
19     // Enlazar el parámetro :admin_id con el valor almacenado en la sesión
20     $stmt->bindParam(':admin_id', $admin_id, PDO::PARAM_INT);
21     $stmt->bindParam(':limit', $limit, PDO::PARAM_INT); // Enlazar el límite dinámico
22
23     // Ejecutar la consulta
24     $stmt->execute();
25
26     // Obtener los resultados
27     return $stmt->fetchAll(PDO::FETCH_ASSOC);
28 }
29

```

Figura 3.5.4.1 Se muestra la función obtenerHistorialAdmin().



```

function obtenerHistorialEmp($pdo)
{
    // Obtener el admin_admin_id de la sesión del empleado logueado
    $admin_id = $_SESSION['admin_admin_id'];

    // Consulta SQL para obtener los tres últimos cortes, pero filtrados por admin_admin_id
    $sql = "SELECT c.*, e.emp_nombre
           FROM corte c
             INNER JOIN empleados e ON c.empleados_emp_id = e.emp_id
             WHERE e.admin_admin_id = :admin_id
             ORDER BY c.corte_horafecha DESC LIMIT 8";

    // Preparar la consulta
    $stmt = $pdo->prepare($sql);

    // Enlazar el parámetro :admin_id con el valor almacenado en la sesión
    $stmt->bindParam(':admin_id', $admin_id, PDO::PARAM_INT);

    // Ejecutar la consulta
    $stmt->execute();

    // Obtener los resultados
    $cortesHistorial = $stmt->fetchAll(PDO::FETCH_ASSOC);

    return $cortesHistorial;
}

```

Figura 3.5.4.2 Se muestra la function obtenerHistorialAdmin().

```
function obtenerHistorialPorFecha($pdo, $fecha)
{
    $adminID = $_SESSION['admin_id'];

    $sql = "SELECT c.*, e.emp_nombre
           FROM corte c
           INNER JOIN empleados e ON c.empleados_emp_id = e.emp_id
           WHERE c.admin_admin_id = :adminID
           AND DATE(c.corte_horafecha) = :fecha
           ORDER BY c.corte_horafecha DESC";

    $stmt = $pdo->prepare($sql);
    $stmt->bindParam(':adminID', $adminID, PDO::PARAM_INT);
    $stmt->bindParam(':fecha', $fecha, PDO::PARAM_STR);
    $stmt->execute();

    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}
```

Figura 3.5.4.3 Se muestra la function obtenerHistorialPorFecha().

```
admin_history.php
CorteCaja > admin_history.php > div.container.mt-5
164
165 <div class="container mt-5">
166     <h2 class="section-title">
167         </?php
168         if ($modoFiltro == "fecha") {
169             echo "|Cortes del " . date("d/m/Y", strtotime($fechaSeleccionada)) . "|";
170         } else {
171             echo "|Últimos $limite cortes|";
172         }
173     </h2>
174
175     <?php if ($modoFiltro == "cantidad") : ?>
176         <h6 class="mb-4 text-center">El corte #1 es el más reciente, el #<?php echo $limite ?> es el más antiguo.</h6>
177     <?php elseif ($modoFiltro == "fecha") : ?>
178         <h6 class="mb-4 text-center">Algunos cortes pueden realizarse después de las 12:00 am, lo cual pertenecerá a otro día!</h6>
179     <?php endif; ?>
180
181     <!-- Botones de filtro -->
182     <div class="d-flex justify-content-center mb-4">
183         <div class="btn-group me-2">
184             <a href="admin_history.php?limite=8"
185                 class="btn <?php echo (isset($limite) && $limite == 8) ? 'btn-primary' : 'btn-secondary'; ?>">Últimos 8</a>
186
187             <a href="admin_history.php?limite=20"
188                 class="btn <?php echo (isset($limite) && $limite == 20) ? 'btn-primary' : 'btn-secondary'; ?>">Últimos 20</a>
189
190             <a href="admin_history.php?limite=50"
191                 class="btn <?php echo (isset($limite) && $limite == 50) ? 'btn-primary' : 'btn-secondary'; ?>">Últimos 50</a>
192         </div>
193     </div>
194
195     <!-- Selector de fecha -->
196     <div class="d-flex justify-content-center mb-4">
197         <form method="GET" action="admin_history.php" class="d-flex align-items-center">
198             <input type="date" name="fecha" class="form-control me-2" required value=<?php echo isset($_GET['fecha']) ? $_GET['fecha'] : ''; ?>>
199             <button type="submit" class="btn btn-success">Filtrar por fecha</button>
200         </form>
201     </div>
202
203     <div class="row justify-content-center">
204         <div class="col-md-12">
205             <div class="row">
```

Figura 3.5.4.4 Interfaz gráfica de la ventana historial.

3.6 Visualizar gráficos

3.6.1 Descripción del caso de uso

La descripción del caso de uso es visualizar de manera gráfica los datos financieros más importantes para la tienda Santana Elia, cabe destacar que esta función está únicamente para el gerente, en la cual requiere observar con prioridad gastos y ventas, esto es importante debido a dicha información poder formular estrategias de venta, así como la preparación a gastos a realizar. Los gráficos se dividen en tres secciones, las cuales se mencionan a continuación:

- Las ventas y gastos de los últimos 3 días: Esto a petición del gerente de la tienda Santana Elia, se puede observar de manera inmediata los gastos y ventas de dichos 3 días, lo que facilita al gerente visualizar estas estadísticas que necesita.
- Gráfico de pastel sobre las ventas y gastos del mes actual: Además de tener una visualización de los últimos 3 días, es requerido observar la evaluación de gastos-ventas del mes, lo cual se puede obtener el rendimiento financiero de la empresa.
- Día con mas venta y gasto del mes: Algo importante con lo anterior, es observar que día fue en el que mas se vendió y se gastó, estos días son independientes entre sí, con la finalidad de observar que día fue el que mas se vendió e indagar en el porque, así mismo, el día con mayor gasto, poder tener una preparación para días similares.

Esta función se encuentra en la barra de herramientas, en la opción “Opciones” y después en el submenú “Gráficos” (véase Figura 3.6.1.1).



Figura 3.6.1.1 Barra de herramientas del gerente.

3.6.2 Modelo de datos

Para lograr hacer las comparaciones de gastos versus ventas, son necesarias 3 tablas (véase Figura 3.6.2.1), las cuales en conjunto, se realizan las consultas necesarias para los gráficos, las cuales son las siguientes:

- admin: De esta tabla se obtiene el id del gerente logueado, debido que los cortes y los gastos están vinculados a este id.
- corte: Esta tabla tiene la información de las ventas de cada turno.
- gasto: Al igual que la de corte, la tabla gasto tiene la información de los gastos, debido a que en la tabla de corte se registra el total de los gastos, sin embargo, existen dos gastos que no deben ser contemplados, por ello, nos basamos en esta tabla.

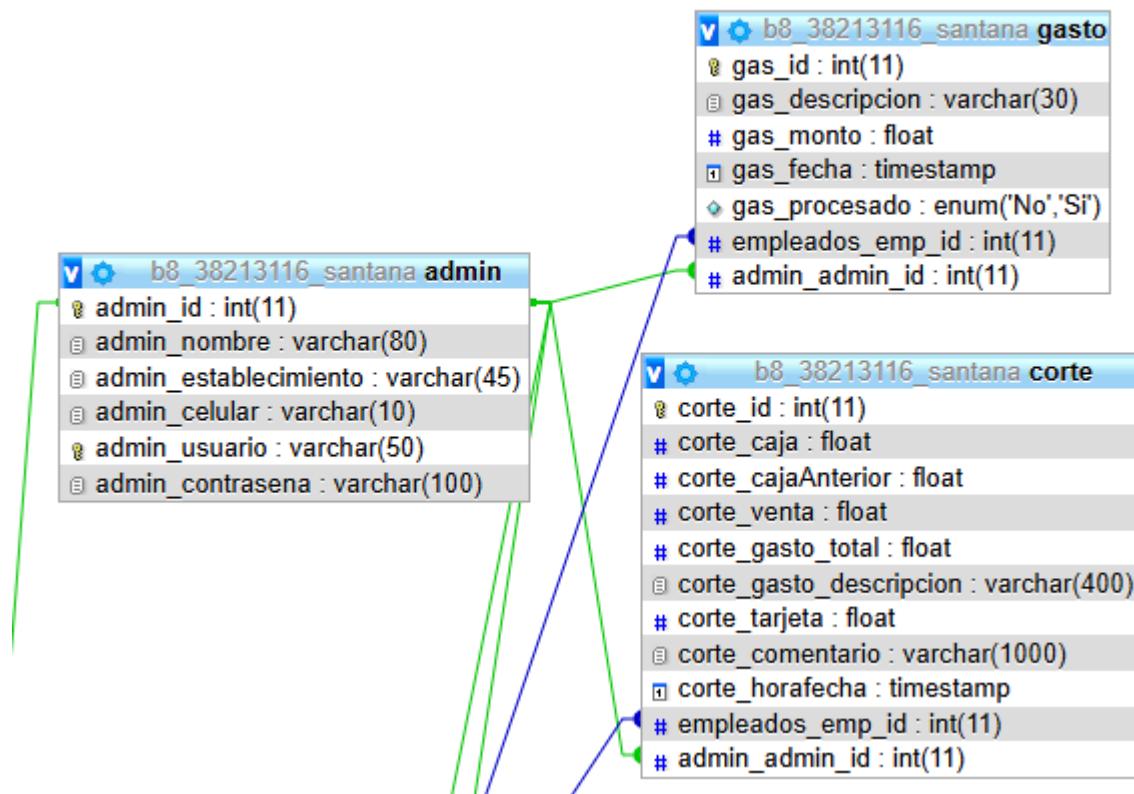


Figura 3.6.2.1 Tablas involucradas en el caso de uso.

3.6.3 Diseño de la interfaz

El diseño se enfocó en tres cosas muy importantes, las cuales son muy significativas que permitirán al gerente deducir su rendimiento lo más rápido y eficaz posible los datos financieros (véase Figura 3.6.3.1), se explica a continuación:

- Prioridad: Una de las cosas prioritarias, es observar la relación de gastos versus ventas de días anteriores, aunque no sea un listado completo, por ello se enfocó en los últimos tres días recientes, siguiente la personalización de inicio, donde se muestran los últimos tres cortes.
- Visualización: La visualización para su interpretación es importante, por ello, se usaron dos tipos de gráficos, de barras y de pastel, con colores azul y rojo, representando el color azul como positivo en el caso de ventas y el color rojo como negativo en el caso de gastos.
- Orden: La manera en el que se organizaron las distintas gráficas, es directo y de rápido acceso, algo que va de la mano con la prioridad y visualización.



Figura 3.6.3.1 Diseño de las graficas implementadas en gastos versus ventas.



Figura 3.6.3.2 Visualización de gráficas desde un dispositivo móvil.

3.6.4 Fragmentos de código

En el backend y frontend, seguimos manejando la misma dinámica, en este caso tenemos el archivo `admin_graphic.php` y `admin_graphic_submit.php`, estos dos archivos hacen posible este caso de uso, lo cual se explicará a continuación:

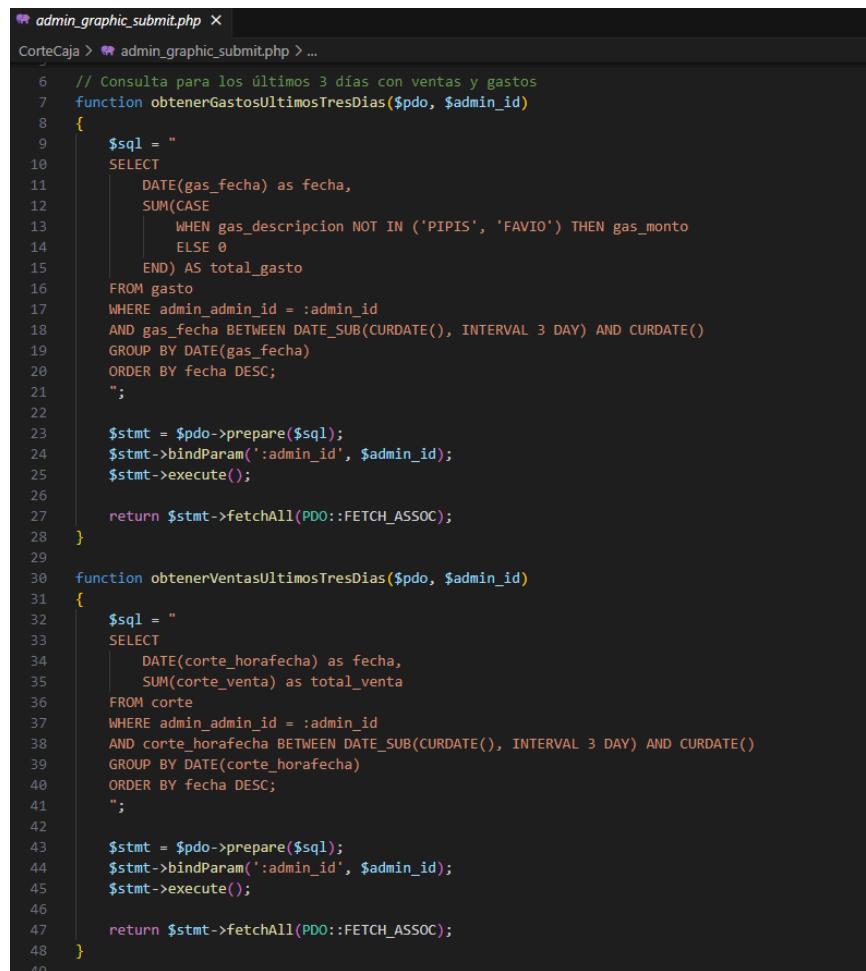
- El archivo `admin_graphic_submit.php` (véase Figura 3.6.4.4) contiene las consultas necesarias para los datos que se insertan en las gráficas, en un total de 6 consultas, se dividen en dos por gráfica, una para obtener las ventas y otra para obtener la consulta, ¿Por qué?, esto se debe al juntar la obtención de los datos en una sola consulta, generaba datos erróneos, por ello se

optó hacerlo de esta manera, una solución simple y eficaz que da continuidad al proyecto.

- Para la primera gráfica, la cual es la que obtiene los gastos y ventas de los tres días más recientes, para ello, como mencionamos anteriormente se dividen en dos consultas, la primera function tiene la consulta para obtener los gastos, llamada obtenerGastosUltimosTresDias(), en esta le compartimos la conexión a la base de datos y el id del administrador, con ello, obtenemos la fecha, sumamos los gastos evitando aquellos que su descripción sea “PIPIS” o “FAVIO”, debido a que son los gerentes que recogen el dinero, esto de la tabla gasto, usando la clave foránea del id del admin, y además que la fecha tenga un intervalo entre la fecha actual y un intervalo de 3 días atrás, se agrupa por fecha y se ordena de manera descendente (véase Figura 3.6.4.1). Algo similar sucede con la toma de ventas, donde tenemos la function obtenerVentasUltimosTresDias(), compartimos la conexión a la base de datos y el id del admin, se obtienen los datos de la fecha y el total de venta, de la tabla corte, donde el admin sea igual al que compartimos y un rango de fecha de 3 días, agrupado por fecha y se ordena de manera descendente (véase Figura 3.6.4.1). Ambos resultados de dichas consultas anteriores, se envían a admin_graphic.php donde se usan para dicha gráfica.
- Para la segunda gráfica, la cual muestra los datos de gastos y ventas del mes, nuevamente se dividen en dos consultas, las cuales las funtions son obtenerVentasMesActual() y obtenerGastosMesActual(), siguiendo el mismo concepto de la primer gráfica, con la diferencia que en lugar de ser un intervalo de 3 días, se hace referencia a todo el mes actual, con esto se

usa MONTH y YEAR de la propiedad CURDATE() (véase Figura 3.6.4.2). También, estos datos se envían a admin_graphics.php para la montar los datos en la gráfica.

- Para la tercera y última gráfica ambas buscando el dia donde hubo más gastos y ventas del mes actual, para ello se usa las propiedades de CURDATE y SUM de las consultas anteriores, sin embargo, en estas consultas se limitan a un resultado, la cual nos otorgará la cantidad mayor (véase Figura 3.6.4.3), de la misma manera que las anteriores, esta información se envía a admin_graphic.php .



```

admin_graphic_submit.php ×
CorteCaja > admin_graphic_submit.php > ...
6 // Consulta para los últimos 3 días con ventas y gastos
7 function obtenerGastosUltimosTresDias($pdo, $admin_id)
8 {
9     $sql = "
10    SELECT
11        DATE(gas_fecha) as fecha,
12        SUM(CASE
13            WHEN gas_descripcion NOT IN ('PIPIS', 'FAVIO') THEN gas_monto
14            ELSE 0
15        END) AS total_gasto
16    FROM gasto
17    WHERE admin_admin_id = :admin_id
18    AND gas_fecha BETWEEN DATE_SUB(CURDATE(), INTERVAL 3 DAY) AND CURDATE()
19    GROUP BY DATE(gas_fecha)
20    ORDER BY fecha DESC;
21    ";
22
23    $stmt = $pdo->prepare($sql);
24    $stmt->bindParam(':admin_id', $admin_id);
25    $stmt->execute();
26
27    return $stmt->fetchAll(PDO::FETCH_ASSOC);
28 }
29
30 function obtenerVentasUltimosTresDias($pdo, $admin_id)
31 {
32     $sql = "
33    SELECT
34        DATE(corte_horafecha) as fecha,
35        SUM(corte_venta) as total_venta
36    FROM corte
37    WHERE admin_admin_id = :admin_id
38    AND corte_horafecha BETWEEN DATE_SUB(CURDATE(), INTERVAL 3 DAY) AND CURDATE()
39    GROUP BY DATE(corte_horafecha)
40    ORDER BY fecha DESC;
41    ";
42
43    $stmt = $pdo->prepare($sql);
44    $stmt->bindParam(':admin_id', $admin_id);
45    $stmt->execute();
46
47    return $stmt->fetchAll(PDO::FETCH_ASSOC);
48 }
49

```

Figura 3.6.4.1 Consultas para obtener gastos y ventas de últimos tres días.

```
admin_graphic_submit.php X
CorteCaja > admin_graphic_submit.php > ...
49
50 // Consulta para el mes actual con ventas y gastos
51 function obtenerVentasMesActual($pdo, $admin_id)
52 {
53     $sql = "
54     SELECT
55         SUM(corte_venta) as total_venta
56     FROM corte
57     WHERE corte.admin_admin_id = :admin_id
58     AND MONTH(corte.corte_horafecha) = MONTH(CURDATE())
59     AND YEAR(corte.corte_horafecha) = YEAR(CURDATE());
60     ";
61
62     $stmt = $pdo->prepare($sql);
63     $stmt->bindParam(':admin_id', $admin_id);
64     $stmt->execute();
65
66     return $stmt->fetch(PDO::FETCH_ASSOC);
67 }
68
69 function obtenerGastosMesActual($pdo, $admin_id)
70 {
71     $sql = "
72     SELECT
73         SUM(CASE
74             WHEN gasto.gas_descripcion NOT IN ('PIPIS', 'FAVIO') THEN gasto.gas_monto
75             ELSE 0
76         END) AS total_gasto
77     FROM gasto
78     WHERE gasto.admin_admin_id = :admin_id
79     AND MONTH(gasto.gas_fecha) = MONTH(CURDATE())
80     AND YEAR(gasto.gas_fecha) = YEAR(CURDATE());
81     ";
82
83     $stmt = $pdo->prepare($sql);
84     $stmt->bindParam(':admin_id', $admin_id);
85     $stmt->execute();
86
87     return $stmt->fetch(PDO::FETCH_ASSOC);
88 }
```

Figura 3.6.4.2 Consultas para obtener gastos y ventas del mes actual.

```
admin_graphic_submit.php X
CorteCaja > admin_graphic_submit.php > ...
89
90 // Consulta Dia mayor venta y gasto
91 function obtenerDiaMasVendido($pdo, $admin_id) {
92     $sql = "
93         SELECT
94             DATE(corte_horafecha) as fecha,
95             SUM(corte_venta) as total_venta
96         FROM corte
97         WHERE admin_admin_id = :admin_id
98         AND MONTH(corte_horafecha) = MONTH(CURDATE())
99         AND YEAR(corte_horafecha) = YEAR(CURDATE())
100        GROUP BY DATE(corte_horafecha)
101        ORDER BY total_venta DESC
102        LIMIT 1;
103    ";
104
105    $stmt = $pdo->prepare($sql);
106    $stmt->bindParam(':admin_id', $admin_id);
107    $stmt->execute();
108
109    return $stmt->fetch(PDO::FETCH_ASSOC);
110 }
111 function obtenerDiaMasGasto($pdo, $admin_id) {
112     $sql = "
113         SELECT
114             DATE(gas_fecha) as fecha,
115             SUM(CASE
116                 WHEN gasto.gas_descripcion NOT IN ('PIPIS', 'FAVIO') THEN gasto.gas_monto
117                 ELSE 0
118             END) AS total_gasto
119         FROM gasto
120         WHERE admin_admin_id = :admin_id
121         AND MONTH(gas_fecha) = MONTH(CURDATE())
122         AND YEAR(gas_fecha) = YEAR(CURDATE())
123         GROUP BY DATE(gas_fecha)
124         ORDER BY total_gasto DESC
125         LIMIT 1;
126    ";
127
128    $stmt = $pdo->prepare($sql);
129    $stmt->bindParam(':admin_id', $admin_id);
130    $stmt->execute();
131
132    return $stmt->fetch(PDO::FETCH_ASSOC);
133 }
134
```

Figura 3.6.4.3 Consultas para obtener gastos y ventas del mes actual.

```

admin_graphic.php
ConeCaja > admin_graphic.php > ...
1 <?php
2 session_start();
3 include 'includes/header.php';
4 include 'admin_graphic_submit.php'; // Incluir el archivo con las funciones de consultas
5
6 // Obtener los datos de ventas y gastos de los últimos 3 días
7 $gastosUltimosTresDias = obtenerGastosUltimosTresDias($pdo, $_SESSION['admin_id']);
8 $ventasUltimosTresDias = obtenerVentasUltimosTresDias($pdo, $_SESSION['admin_id']);
9
10 // Función para transformar las fechas a "Ayer", "Domingo", "Sábado", etc. en español
11 function obtenerDiaDeLaSemana($fecha)
12 {
13     $ hoy = date('Y-m-d');
14     $ ayer = date('Y-m-d', strtotime('-1 day'));
15
16     // Array con los días de la semana en español
17     $diasSemana = [
18         "Sunday" => "Domingo",
19         "Monday" => "Lunes",
20         "Tuesday" => "Martes",
21         "Wednesday" => "Miércoles",
22         "Thursday" => "Jueves",
23         "Friday" => "Viernes",
24         "Saturday" => "Sábado"
25     ];
26
27     // Si la fecha es hoy, devolver "Hoy"
28     if ($fecha == $hoy) {
29         return 'Hoy';
30     }
31     // Si la fecha es ayer, devolver "Ayer"
32     elseif ($fecha == $ayer) {
33         return 'Ayer';
34     }
35     // Si no es ni hoy ni ayer, devolver el nombre del día en español
36     else {
37         return $diasSemana[date('l', strtotime($fecha))]; // Usamos date('l') para obtener el nombre del día en inglés y lo mapeamos a español
38     }
39 }
40
41 // Combinar los resultados de ventas y gastos
42 $datosCombinados = [];
43 foreach ($ventasUltimosTresDias as $venta) {
44     $fecha = $venta['fecha'];
45     $total_venta = $venta['total_venta'];
46
47     // Buscar el gasto correspondiente a la misma fecha
48     $total_gasto = 0;
49     foreach ($gastosUltimosTresDias as $gasto) {
50         if ($gasto['fecha'] == $fecha) {
51             $total_gasto += $gasto['total_gasto'];
52             break;
53         }
54     }
55 }

```

Figura 3.6.4.4 Código de la interfaz gráfica de la ventana Gráficas.

3.7 Registrar, visualizar y eliminar importes.

3.7.1 Descripción del caso de uso

La descripción del caso de uso es registrar, visualizar y eliminar los importes generados en un lapso de tiempo, usualmente en las tiendas de abarrotes se maneja el uso de importes para envases retornables, aquellos que requieren del intercambio del mismo envase por el producto a comprar, es decir, si el cliente quiere adquirir una Coca Cola de 2.5 litros, este deberá llevar el envase de dicho producto y pagar el precio del producto, sin embargo, si no lleva el envase requerido se le cobrará un importe, este importe ayuda a la tienda a amortiguar el pago del envase faltante al momento de adquirir producto con los proveedores.

El sistema SIGMA (Sistema Integral de Gestión para Microempresas de Abarrotes) tiene la herramienta de hacer registro del mismo (véase Figura 3.7.1.1), el cual los campos a llenar son el nombre del cliente, cantidad de envases y seleccionar un tipo de envase (véase Figura 3.7.1.2), una vez llenado estos campos, se guarda en la base de datos, la cual, se muestra mediante una tabla (véase Figura 3.7.1.3) con el nombre del cliente, la descripción del importe el cual se conforma de la cantidad de envases y el tipo de envase, también se tiene una columna con el importe total, la fecha en que se realizó y que empleado hizo el registro, como utilidad se tienen dos botones, que se le denomina como acciones, a continuación:

- Acciones con el registro de importe:
 - Marcar como pagado: Esta representado con un botón de color verde con el signo de pesos, su funcionalidad son dos, primeramente el registro como realizado para mostrarse el registro en la tabla de importes entregados (véase Figura 3.7.1.4) y se agrega la cantidad total del importe a la tabla de gastos del usuario logueado (véase Figura 3.7.1.5), esto por que se considera un gasto el regresar un importe, de esta manera, evitamos que el empleado tenga la necesidad de registrar dicho gasto.
 - Eliminar importe: Esta representado con un botón de color rojo con el icono de basura, su funcionalidad es simplemente eliminar el registro, en caso de equivocación.

Los importes tienen un respaldo de 15 días, es decir, si pasa de este tiempo limite, el registro se elimina, por ello, no se regresaría el importe al no existir un registro, esto se hace para evitar acumulacion de datos innecesarios y a su vez el interés del cliente por el importe, se ha observado de quien devuelve el envase, normalmente se hace en un lapso de horas hasta 5 días después.

Además, el gerente podrá visualizar los importes realizados, esta opción se encuentra en el menú desplegable al seleccionar “Opciones” y en el submenú “Importes” (véase Figura 3.7.1.6), sin oportunidad de eliminar o marcar como pagado, esto debido a que el gerente de manera cotidiana no se encuentra en el establecimiento, por ello, se evita que los datos tengan una alteración o sean eliminados erróneamente, para ello se limita solo en su visualización (véase Figura 3.7.1.7).

También, el gerente tanto como el usuario con rol de encargado, podrán ajustar los precios de los importes, agregar nuevos tipos de importes y eliminar tipos de importe (véase Figura 3.7.1.9). esto con la finalidad de cuando el establecimiento adquiera nuevos productos o deje de adquirir los que ya tiene que requieran envase. La ubicación de esta ventana para el empleado con rol de encargado y gerente está en el menú desplegable “Precios” y en el submenú “Importes” (véase Figura 3.7.1.10 y Figura 3.7.1.11)

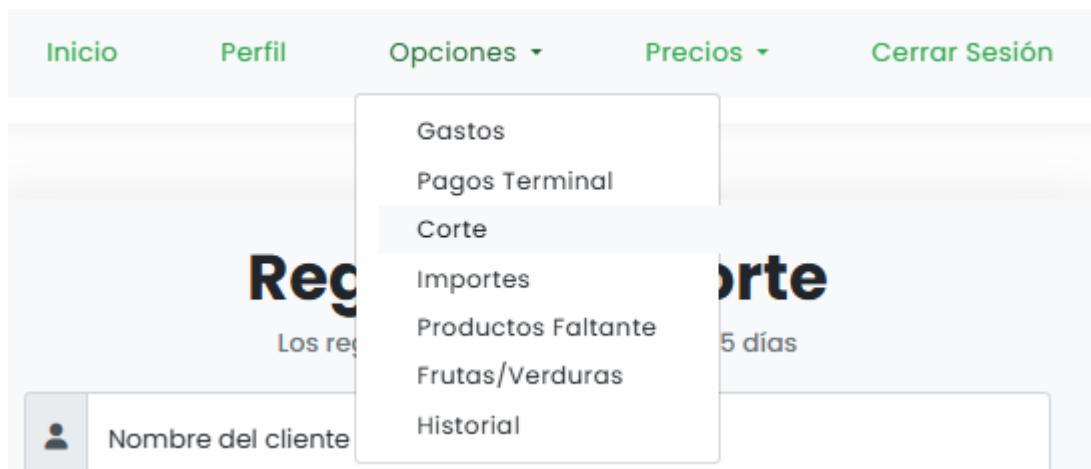


Figura 3.7.1.1 Herramienta de registrar importe.

Registrar Importe

Los registros solo se guardan por 15 días

Selecciona un tipo de importe
▼

Agregar

Figura 3.7.1.2 Formulario para registrar importe.

Importes Registrados

Cliente	Envase(s)	Total	Fecha	Empleado	Acciones
MELECIO	1 COCA 1L	\$15.00	25/04 14:33	Dalila	\$ Delete
GERMAN	1 CAGUAMA	\$10.00	16/04 02:17	Cesar	\$ Delete
EFRAIN ORTEGA	1 CAGUAMA	\$10.00	15/04 23:55	Cesar	\$ Delete
LUIS	1 COCA 1.5L	\$15.00	15/04 22:35	Ramon	\$ Delete
JOSE JUARES	2 MODELO	\$10.00	13/04 12:23	Sulma	\$ Delete
GUAYABA	1 CAGUAMA	\$10.00	13/04 01:36	Cesar	\$ Delete

Figura 3.7.1.3 Tabla para mostrar importes.

<h2>Importes Entregados</h2>				
Cliente	Envase(s)	Cantidad	Entregado Por	Fecha Devolución
RAFAEL	1 CAGUAMA	\$10	Cesar	24/04 00:03

Figura 3.7.1.4 Tabla para mostrar importes entregados.

<h2>Gastos Registrados</h2>			
Monto	Descripción	Hora	Acciones
20	IMPORTE	02:02	

Figura 3.7.1.5 Se agrega como gasto al marcar un importe como entregado.

The screenshot shows a user interface with a navigation bar at the top containing links like 'Historial', 'Opciones', 'Precios', and 'Cerrar Sesión'. Below this, there is a section titled 'Últimos tres c' (Last three) with a dropdown menu open over it. The dropdown menu contains three items: 'Productos Faltantes', 'Graficos', and 'Importes'.

Figura 3.7.1.6 Herramienta de importes para el gerente.

Importes Entregados

Cliente	Envase(s)	Cantidad	Entregado Por	Fecha Devolución
JOEL	2 CAGUAMA	\$20	Cesar	06/05 14:02

Importes Registrados

Nombre	Envase(s)	Total	Fecha	Empleado
JOEL	2 CAGUAMA	\$20.00	04/05 01:27	Cesar
ALEX	1 CAGUAMA	\$10.00	04/05 01:27	Cesar
GERMAN	1 CAGUAMA	\$10.00	04/05 01:22	Cesar
JORGE	2 MODELO	\$10.00	03/05 23:57	Cesar

Figura 3.7.1.7 Importes vista para el gerente.

Importes Entregados			
	Envase(s)	Cantidad	Entregado Por
	2 CAGUAMA	\$20	Cesar

Importes Registrados				
Nombre	Envase(s)	Total	Fe	
JOEL	2 CAGUAMA	\$20.00	04 0	
ALEX	1 CAGUAMA	\$10.00	04 0	
GERMAN	1 CAGUAMA	\$10.00	04 0	

Figura 3.7.1.8 Visualización Importes vista del gerente en dispositivo móvil, la tabla tiene desplazamiento horizontal.

The screenshot shows a mobile application interface. At the top, a header bar contains the text "Registrar Tipo Importe". Below this, there are two input fields: "Nombre del tipo de importe" (Import Type Name) and "Precio unitario del importe" (Unitary Price). A green "Agregar" (Add) button is located below the second field. Below the input fields, the title "Tipos Importes Registrados" (Registered Import Types) is displayed, followed by a table listing four entries:

Nombre	Precio	Acciones
CAGUAMA	10	
CARTON 1/2	100	
CARTON 1/4	100	
COCOA		

Figura 3.7.1.9 Ventana para agregar, eliminar o actualizar tipo de importe.

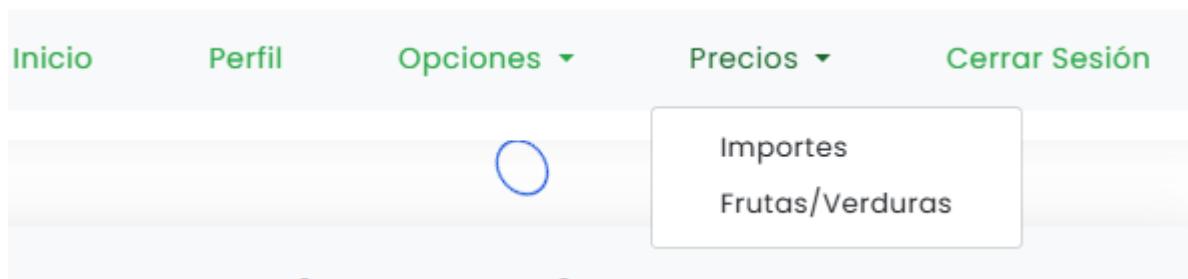
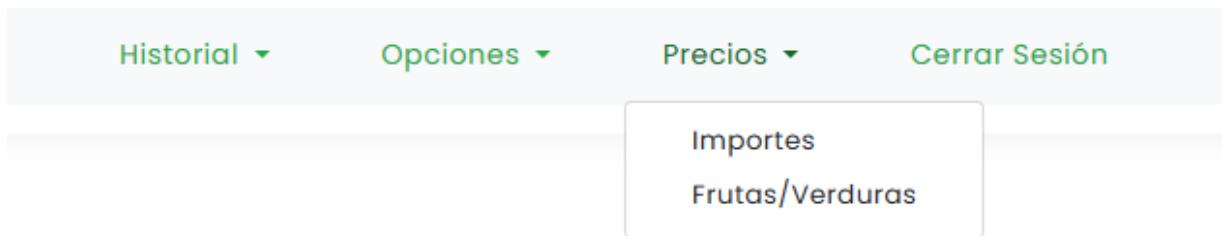


Figura 3.7.1.10 Ajustes en tipos de importe submenú del empleado.



¡timos tres cortes!

Figura 3.7.1.11 Ajustes en tipos de importe submenú del gerente.

3.7.2 Modelo de datos

Para la realización del registro, visualización y eliminación de los importes, se involucran 5 tablas (véase Figura 3.7.2.1), las cuales son las siguientes:

- admin: Esta tabla se involucra como clave foránea en las tablas de importes y categorias_importes, esto tiene como función separar un registro de importe según el gerente del negocio, además, como en gastos, también tenemos categorías, donde el id del gerente funciona como identificador de qué categoría corresponde a un establecimiento.
- importes: Es la tabla principal para el registro, en este podemos encontrar un id autoincremental, el nombre del cliente, la descripción que es un conjunto con cantidad de envases y tipo de envase, el total del importe, la fecha y hora en que se realiza el registro, el empleado que hizo el registro, y el id del administrador, estos datos son los mínimos necesarios que se requieren para recibir y pagar el importe.
- empleados: Esta tabla funciona también como clave foránea cuando se realiza un registro en la tabla importes, menciona que empleado hizo dicho registro.
- gasto: Al seleccionar el botón de pagar un importe, la cantidad total del registro se suma a la tabla de gasto, esto con la finalidad de que el usuario empleado no tenga que ir a registrar dicho gasto efectuado.
- categorias_importes: La tabla funciona como para clasificar los tipos de envases en los cuales se cobran importe, esto, para asignarle un nombre y un precio, los cuales en el formulario se mandan a llamar y efectuar la operación entre cantidad de

envases y precio del importe. Además, de agregar, eliminar o actualizar tipos de importe que se requieran.

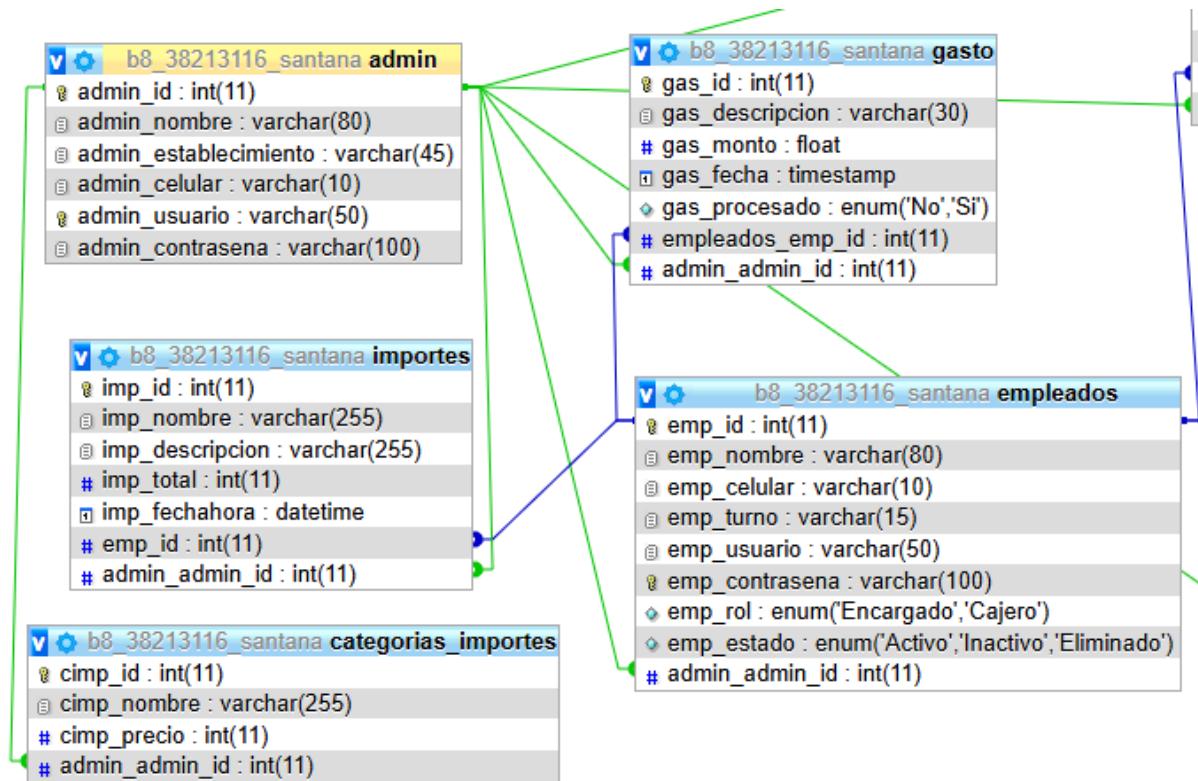


Figura 3.7.2.1 Tablas involucradas para registrar, visualizar y eliminar un importe.

3.7.3 Diseño de la interfaz

El diseño de la interfaz para el empleado mantiene el mismo orden y objetividad (véase Figura 3.7.3.1), dado el sistema busca ser lo más simple posible para los usuarios, por ello, se tomó las siguientes decisiones:

- **Formulario similar:** Este formulario se diseñó en base a los anteriores, tratando de ser lo más familiarizado posible, para que el usuario no tenga la necesidad de realizar acciones distintas para realizar un registro.
- **Tabla para mostrar registros:** De la misma manera, el diseño trata de ser amigable y familiar, aunque, debido a mucha información esta tabla

aumenta de longitud, y debido a la cantidad de importes registrados en la tienda Santana Elia, no existe un filtrado, ya que también está pensado para almacenar los importes a solo 10 días después del registro, con ello, disminuyen los registros totales.

- Acceso al registro: Al igual que registrar algún gasto, pago con terminal o corte, la opción para registrar un importe se encuentra en el menú de opciones.

De la misma manera que lo anterior visto, para el gerente tenemos únicamente la visualización de los importes registrados (véase Figura 3.7.3.2), esto para evitar pérdidas de datos, el gerente podrá acceder a ello desde el menú de herramientas en la opción “Importes”, como se mencionó anteriormente:

- Marcar como pagado: El gerente no tiene posibilidad de generar gastos, por ello, es posible que marque un error.
- Eliminar registró: El gerente de manera cotidiana no se encuentra en el establecimiento, por ello, no mantiene una atención al registro de importes.

Tipos de importes (véase Figura 3.7.3.3) para usuarios empleados con rol de encargado y gerentes:

- Agregar: Tenemos un formulario para registrar tipos de importes en caso que el establecimiento adquiera nuevos productos que requieran envases y estos no estén registrados.
- Eliminar: A su vez, en caso que el establecimiento deje de consumir productos con envase, los usuarios mencionados podrán eliminar el tipo de importe para evitar acumular datos innecesarios.
- Actualizar: Al actualizar podremos cambiar tanto el nombre como el monto del tipo de importe, esto por actualizaciones de precio futuro o en caso de alguna equivocación.

Registrar Importe

Los registros solo se guardan por 15 días

	Nombre del cliente
	Cantidad de envases
	Tipo de importe Selecciona un tipo de importe

Agregar

Importes Entregados

Cliente	Envase(s)	Cantidad	Entregado Por	Fecha Devolución
JOEL	2 CAGUAMA	\$20	Cesar	06/05 14:02

Importes Registrados

Cliente	Envase(s)	Total	Fecha	Empleado	Acciones
ALEX	1 CAGUAMA	\$10.00	04/05 01:27	Cesar	
GERMAN	1 CAGUAMA	\$10.00	04/05 01:22	Cesar	
JORGE	2 MODELO	\$10.00	03/05	Cesar	

Figura 3.7.3.1 Diseño de la interfaz para registrar, visualizar y eliminar importes.

The screenshot shows a software application window with a light gray header bar containing navigation links: Inicio, Perfil, Mis Empleados (with a dropdown arrow), Historial (with a dropdown arrow), Opciones (with a dropdown arrow), Precios (with a dropdown arrow), and Cerrar Sesión. Below the header is a search bar with a magnifying glass icon and a progress bar.

Importes Entregados

Cliente	Envase(s)	Cantidad	Entregado Por	Fecha Devolución
JOEL	2 CAGUAMA	\$20	Cesar	06/05 14:02

Importes Registrados

Nombre	Envase(s)	Total	Fecha	Empleado
JOEL	2 CAGUAMA	\$20.00	04/05 01:27	Cesar
ALEX	1 CAGUAMA	\$10.00	04/05 01:27	Cesar
GERMAN	1 CAGUAMA	\$10.00	04/05 01:22	Cesar
JORGE	2 MODELO	\$10.00	03/05 23:57	Cesar
-----	-----	-----	-----	-

Figura 3.7.3.2 Diseño de la interfaz visualizar importes.

Nombre	Precio	Acciones
CAGUAMA	10	
CARTON 1/2	100	

Figura 3.7.3.3 Diseño de la interfaz añadir, eliminar o actualizar tipos de importes.

3.7.4 Fragmentos de código

En el fragmento del código, se dividieron en varios archivos PHP, esto debido a la seguridad y separación de funciones, las cuales se describirán a continuación:

- Interfaz gráfica empleado: Este archivo se nombró como `employee_generate_import.php` (véase Figura 3.7.4.1), este código almacena todo lo gráfico, se tiene acceso a varios archivos, como lo son en header, footer y consultas, primero se nos presenta el formulario para realizar un registro, tenemos los campos en mayúsculas para ingresar el nombre del cliente, la cantidad de envases y el tipo de importe, estos datos se envían a la consulta que los agregara a la base de datos, además,

tenemos, un AAGREGAR NUEVO IMPORTE, esto con la finalidad de poder registrar un nuevo importe, esto habilita dos campos extras para agregar el nombre del importe y el precio del mismo. También, más abajo de este formulario, se encuentra la tabla en el cual se muestra la información de los importes registrados.

- Funciones para empleado: Este archivo se nombró como employee_generate_import_impquery.php (véase Figura 3.7.4.2), su funcionalidad principal es almacenar funciones para la interfaz grafica y funcionamiento de los botones de color verde y rojo, encontramos funciones como:
 - Eliminar importes antiguos: El cual cada vez que se accede a los importes, ejecuta la función para eliminar importes con almacenamiento mayor a 30 días.
 - Obtener los importes registrados: Se obtienen todos los importes registrados ligados al id del administrador, y estos se ordenan por fecha, los más recientes al principio.
 - Obtener las categorías de importes: Esto se obtiene el listado de los diferentes importes existentes para el id del administrador, este listado se muestra en el formulario a través de un menú desplegable donde el empleado selecciona uno.
 - Eliminar importe: Mediante la solicitud de eliminar importe, esa información llega a esta función, en el cual como su nombre lo dice, eliminar el importe con el id recibido.
 - Marcar como pagado: Primeramente se implementa el agregar el total del importe seleccionado, a la tabla gastos, esto con la finalidad de que el empleado no tenga que registrarla, y después de esto, el importe se elimina.

- Agregar importe para empleado: Este archivo se nombró como employee_generate_import_submit.php (véase Figura 3.7.4.3), en esta ocasión se realiza el registro del importe y el registro del nuevo importe, para ello, primeramente recibidos los datos, se verifica que los campos mínimos necesarios están completos con alguna información, por consiguiente, si el empleado registrara un nuevo importe, se verifica que los campos necesarios están disponibles, de ser así, se agrega la nueva categoría, y continua con el registro del importe.
- Interfaz gráfica gerente: Este archivo se nombró como admin_import.php (véase Figura 3.7.4.4), en este, mantiene la misma estructura que la interfaz del empleado, sin embargo, se limita únicamente a la visualización de los importes ya registrados.
- Funciones para el gerente: Este archivo se nombró como admin_import_submit.php (véase Figura 3.7.4.5), la funcionalidad de este, es solo obtener los registros de la tabla importes, siendo una funcionalidad similar al de empleado.
- Header empleados: Para empleados con rol de encargado, se agregó la condición en el header para obtener el menú “Precios” (véase Figura 3.7.4.6), mientras que el gerente tiene dicho menú sin ninguna condición.
- Interfaz gráfica gerente y empleado con rol encargado para tipos de importe: Tenemos los archivos admin_price_import.php (véase Figura 3.7.4.7) y employee_price_import.php (véase Figura 3.7.4.8) los cuales tienen muy similar debido a que en este aspecto el empleado con rol encargado tiene exactamente los mismo privilegios que el encargado, por ello su similitud.
- Controlador para tipo de importe: Tenemos los archivos admin_price_import_query.php (véase Figura 3.7.4.10) y

`employee_price_import_query.php` (véase Figura 3.7.4.9) en los cuales tenemos mucha similitud con las funciones:

- `tiposImportesRegistrados`: La finalidad es obtener el tipo de los importes registrados, como los son sus nombres y precios, que corresponden al id del gerente, todos ordenados por nombre de manera ascendente.
- `eliminarTipolImporte`: La función como su nombre lo dice, eliminar un tipo de importe, mediante método post se recibe el id del tipo de importe y el id del gerente, para su posterior eliminación.
- `actualizarTipolImporte`: Actualiza el importe, mediante el post recibimos id, precio y nombre del tipo de importe, y además el id del gerente para su actualización.
- Submit para el registro de tipos de importe: A su vez, siendo estos archivos muy parecidos, tenemos primeramente la recepción de datos, las cuales se guardan en variables, después se verifica que estas tengan información, si no, el registro no se realiza, si tienen información primero se verifica que el nombre no se repita dentro del establecimiento, en caso de ser repetido el registro no se realiza, en caso de ser nombre único se procede con su registro en la base de datos.

```

employee_generate_import.php
CorteCaja > employee_generate_import.php > div.container.mt-5 > div.row.justify-content-center > div.col-md-8 > div.card > div.card-body > form > div.alert.alert-success.mt-3
1  <?php
2  session_start();
3  $error_message = isset($_SESSION['error_message']) ? $_SESSION['error_message'] : "";
4  unset($_SESSION['error_message']);
5  $success_message = isset($_SESSION['success_message']) ? $_SESSION['success_message'] : "";
6  unset($_SESSION['success_message']);
7
8
9  include 'includes/header.php';
10 include 'includes/conexion_db.php';
11 include 'employee_generate_imquery.php';
12
13 $tipos_importes = obtenerTodasCategoriasImportes($pdo);
14 $importes = obtenerImportesRegistrados($pdo);
15 $eliminar = eliminarImportesAntiguedad($pdo);
16 ?>
17
18 <!-- Formulario para generar importes -->
19 <div class="container mt-5">
20   <div class="row justify-content-center">
21     <div class="col-md-8">
22       <div class="card">
23         <h1 class="text-center mt-4">Registrar Importe</h1>
24         <h6 class="text-center mt-1">Los registros solo se guardan por 30 días</h6>
25         <div class="card-body">
26           <form action="employee_generate_import_submit.php" method="post">
27
28             <div class="form-group input-group">
29               <div class="input-group-prepend">
30                 | <span class="input-group-text"><i class="fas fa-user"></i></span>
31               </div>
32               <input type="text" class="form-control" id="imp_nombre" name="imp_nombre" required placeholder="Ingresa el nombre del cliente">
33
34             <div class="form-group input-group">
35               <div class="input-group-prepend">
36                 | <span class="input-group-text"><i class="fas fa-cube"></i></span>
37               </div>
38               <input type="number" class="form-control" id="imp_cantidad" name="imp_cantidad" required placeholder="Cantidad de envases" min="1">
39
40             <div class="form-group input-group">
41               <div class="input-group-prepend">
42                 | <span class="input-group-text"><i class="fas fa-tag"></i></span>
43               </div>
44               <select class="form-control" id="imp_descripcion" name="imp_descripcion" required onchange="mostrarInputImporte()">
45                 <option value="">Selecciona un tipo de importe</option>
46                 <php foreach ($tipos_importes as $tipo) : ?>
47                   <option value=<?php echo trim($tipo['imp_nombre']); ?>>
48                     <?php echo $tipo['imp_nombre']; ?>
49                   </option>
50                 <php endforeach; ?>
51               </select>
52
53             </div>
54
55             <!-- Campo para ingresar un nuevo importe (oculto por defecto) -->
56             <div class="form-group" id="nuevoImporteDiv" style="display: none;">
57               <div class="input-group">
58                 <div class="input-group-prepend">
59                   | <span class="input-group-text"><i class="fas fa-pen"></i></span>
60                 </div>
61                 <input type="text" class="form-control" id="nuevoImporteInput" name="nuevo_importe" placeholder="Escriba el nombre del nuevo importe">
62
63               <div class="input-group mt-3">
64                 <div class="input-group-prepend">
65                   | <span class="input-group-text"><i class="fas fa-dollar-sign"></i></span>
66                 </div>
67                 <input type="number" class="form-control" id="nuevoImporteCosto" name="nuevo_importe_costo" placeholder="Ingrese el costo unitario" step="1" min="1">
68
69             </div>
70
71           </div>

```

Figura 3.7.4.1 Código de la interfaz gráfica del empleado.

The screenshot shows a code editor window with the following details:

- Title Bar:** employee_generate_impquery.php X
- Project Path:** CorteCaja > employee_generate_impquery.php > eliminarImporte
- Code Content:** The code is written in PHP and defines several functions:
 - eliminarImportesAntiguedad(\$pdo):** Deletes imports older than 30 days.
 - obtenerImportesRegistrados(\$pdo):** Selects imported items registered by a specific admin.
 - obtenerTodasCategoriasImportes(\$pdo):** Selects all categories of imports.
 - eliminarImporte(\$pdo, \$imp_id):** Deletes a specific import by its ID.
- Editor Features:** The interface includes tabs for 'File', 'Edit', 'View', 'Search', 'Tools', and 'Help'. It also shows a sidebar with file navigation and a status bar at the bottom.

Figura 3.7.4.2 Código de las funciones.

```
employee_generate_import_submit.php X
CoreCaja > employee_generate_import_submit.php > ...
1  <?php
2  session_start();
3  include 'includes/conexion_db.php';
4
5  // Verificar si el usuario está autenticado
6  if (!isset($_SESSION['emp_id']) || !isset($_SESSION['admin_admin_id'])) {
7      $_SESSION['error_message'] = "Error de autenticación.";
8      header("Location: employee_generate_import.php");
9      exit();
10 }
11
12 // Obtener datos del formulario
13 $imp_nombre = trim($_POST['imp_nombre']); // Nombre del cliente
14 // Convertir a mayúsculas para mantener uniformidad
15 $imp_nombre = strtoupper($imp_nombre);
16 $imp_descripcion = $_POST['imp_descripcion']; // Nombre o nombre del tipo de importe
17 $imp_cantidad = (int) $_POST['imp_cantidad']; // Cantidad de envases
18 $emp_id = $_SESSION['emp_id']; // ID del empleado
19 $admin_admin_id = $_SESSION['admin_admin_id']; // ID del administrador
20 $nuevo_importe = trim($_POST['nuevo_importe'] ?? ""); // Nombre del nuevo importe (si aplica)
21 $nuevo_importe_costo = $_POST['nuevo_importe_costo'] ?? null; // Costo del nuevo importe (si aplica)
22
23 // Validar que los campos básicos no estén vacíos
24 if (empty($imp_nombre) || empty($imp_cantidad)) {
25     $_SESSION['error_message'] = "El nombre del cliente y la cantidad de envases son obligatorios.";
26     header("Location: employee_generate_import.php");
27     exit();
28 }
29
30 // Si el usuario seleccionó "AAGREGAR NUEVO IMPORTE", se debe agregar primero el nuevo importe
31 if ($imp_descripcion === "AAGREGAR NUEVO IMPORTE") {
32     // Validar que el nuevo importe y su costo estén llenos
33     if (empty($nuevo_importe) || empty($nuevo_importe_costo)) {
34         $_SESSION['error_message'] = "Debe ingresar el nombre y el costo del nuevo importe.";
35         header("Location: employee_generate_import.php");
36         exit();
37     }
38
39 // Convertir a mayúsculas para mantener uniformidad
40 $nuevo_importe = strtoupper($nuevo_importe);
41
42 try {
43     // Insertar el nuevo tipo de importe en la base de datos
44     $stmt = $pdo->prepare("INSERT INTO categorias_importes (cimp_nombre, cimp_precio, admin_admin_id) VALUES (:nombre, :precio, :admin_admin_id)");
45     $stmt->execute([
46         ':nombre' => $nuevo_importe,
47         ':precio' => $nuevo_importe_costo,
48         ':admin_admin_id' => $admin_admin_id
49     ]);
50
51     $imp_descripcion = $nuevo_importe;
52
53 } catch (PDOException $e) {
54     $_SESSION['error_message'] = "Error al registrar el nuevo importe.";
55     header("Location: employee_generate_import.php");
56     exit();
57 }
58 }
```

Figura 3.7.4.3 Código del registro de importes.

```
admin_import.php X
CorteCaja > admin_import.php > div.icon-effects-w3l > img.img-fluid.shape-wthree.shape-w3-three
1  <?php
2  session_start();
3
4  include 'includes/header.php';
5  include 'includes/conexion_db.php';
6  include 'admin_import_submit.php';
7
8
9  $importes = obtenerImportesRegistrados($pdo);
10 ?>
11
12 <!-- Formulario para mostrar importes -->
13 <div class="container mt-5">
14   <div class="card">
15     <h2 class="text-center mt-4">Importes Registrados</h2>
16     <div class="card-body">
17       <div class="table-responsive">
18         <table class="table table-bordered table-hover">
19           <thead class="table-success">
20             <tr>
21               <th>Nombre</th>
22               <th>Descripción</th>
23               <th>Total</th>
24               <th>Fecha</th>
25               <th>Empleado</th>
26             </tr>
27           </thead>
28           <tbody>
29             <?php foreach ($importes as $importe) : ?>
30               <tr>
31                 <td><?php echo htmlspecialchars($importe['imp_nombre']); ?></td>
32                 <td><?php echo htmlspecialchars($importe['imp_descripcion']); ?></td>
33                 <td><?php echo number_format($importe['imp_total'], 2); ?></td>
34                 <td><?php echo date('m-d H:i', strtotime($importe['imp_fechahora'])); ?></td>
35                 <td><?php echo htmlspecialchars($importe['emp_usuario']); ?></td>
36               </tr>
37             <?php endforeach; ?>
38           </tbody>
39         </table>
40       </div>
41     </div>
42   </div>
43 </div>
44
45
46 <!-- Animaciones -->
47 <div class="icon-effects-w3l">
48   <img alt="Icono de efectos de animación" data-bbox="144 104 258 118" />
```

Figura 3.7.4.4 Código de la interfaz gráfica del gerente.

admin_import_submit.php X

CorteCaja > admin_import_submit.php > obtenerImportesRegistrados

```

1  <?php
2
3  function obtenerImportesRegistrados($pdo)
4  {
5      $adminID = $_SESSION['admin_id'];
6
7      $sql = "SELECT i.*, e.emp_usuario
8          FROM importes i
9          JOIN empleados e ON i.emp_id = e.emp_id
10         WHERE i.admin_admin_id = :adminID
11         ORDER BY i.imp_fechahora DESC";
12
13     $stmt = $pdo->prepare($sql);
14     $stmt->bindParam(':adminID', $adminID, PDO::PARAM_INT);
15     $stmt->execute();
16
17     return $stmt->fetchAll(PDO::FETCH_ASSOC);
18 }
19
20 ?>

```

Figura 3.7.4.5 Código para obtener los importes registrados para el gerente.

header.php X

CorteCaja > includes > header.php ...

```

1  <html lang="es">
2  <body>
3    <div classe="main-top" id="home">
4      <header>
5        <div classe="container-fluid">
6            </div>
7            <div classe="dropdown-menu text-success" aria-labelledby="navbarDropdown">
8                <a classe="dropdown-item" href="employee_generate_purchases.php">Gastos</a>
9                <a classe="dropdown-item" href="employee_generate_card.php">Pagos Terminal</a>
10               <a classe="dropdown-item" href="employee_generate_expenses.php">Corte</a>
11               <a classe="dropdown-item" href="employee_generate_import.php">Importes</a>
12               <a classe="dropdown-item" href="employee_generate_product.php">Productos Faltante</a>
13               <a classe="dropdown-item" href="employee_see_vegetables.php">Frutas/Verduras</a>
14               <a classe="dropdown-item" href="employee_history.php">Historial</a>
15            </div>
16        </div>
17        <?php IF (isset($_SESSION['emp_rol'])) && $_SESSION['emp_rol'] === 'Encargado' : ?>
18        <li classe="nav-item dropdown">
19            <a classe="nav-link dropdown-toggle text-success" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
20                Precios
21            </a>
22            <div classe="dropdown-menu text-success" aria-labelledby="navbarDropdown">
23                <a classe="dropdown-item" href="employee_price_import.php">Importes</a>
24                <a classe="dropdown-item" href="employee_price_vegetables.php">Frutas/Verduras</a>
25            </div>
26        </li>
27        <?php endif; ?>
28        <li classe="nav-item">
29            <a classe="nav-link text-success" href="logout.php">Cerrar Sesión</a>
30        </li>
31        <?php else : ?>
32            <!-- Mostrar menú para usuarios no logeados -->
33            <li classe="nav-item active">
34                <a classe="nav-link text-success" href="index.php">Inicio</a>
35            </li>
36        <?php endif; ?>
37    </div>
38    </header>
39    <div classe="main-content">
40        <div classe="container">
41            <div classe="row">
42                <div classe="col-12">
43                    <h1>CorteCaja</h1>
44                </div>
45            </div>
46        </div>
47    </div>
48
```

Figura 3.7.4.6 Código header el cual muestra el menú “Precios” al empleado con rol encargado.

```

admin_price_import.php X
CoreCaja > admin_price_import.php > ...
1  <?php
2  session_start();
3  $error_message = isset($_SESSION['error_message']) ? $_SESSION['error_message'] : "";
4  unset($_SESSION['error_message']);
5
6  include 'includes/header.php';
7  include 'includes/conexion_db.php';
8  include 'admin_price_import_query.php';
9
10 $AdminID = $_SESSION['admin_id'];
11 $tiposImp = tiposImportesRegistrados($pdo, $AdminID);
12
13 ?>
14
15 <!-- Formulario para generar gastos -->
16 <div class="container my-5">
17   <div class="row justify-content-center">
18     <div class="col-md-8">
19       <div class="card shadow-lg rounded-4 border-0">
20         <div class="card-body p-4 bg-light">
21           <h1 class="text-center fw-bold mb-1">Registrar Tipo Importe</h1>
22           <div class="card-body">
23             <form action="admin_price_import_submit.php" method="post">
24
25               <!-- Nombre del importe -->
26               <div class="input-group mb-3">
27                 <span class="input-group-text"><i class="fas fa-tag"></i></span>
28                 <div class="form-floating flex-grow-1">
29                   <input type="text" class="form-control" id="nombre_importe" name="nombre_importe" required placeholder="Nombre del importe">
30                   <label for="nombre_importe">Nombre del importe</label>
31                 </div>
32               </div>
33
34               <script>
35                 // Convierte el texto ingresado en mayúsculas mientras el usuario escribe
36                 document.getElementById("nombre_importe").addEventListener("input", function() {
37                   this.value = this.value.toUpperCase();
38                 });
39               </script>
40
41             </div>
42           </div>
43         </div>
44       </div>
45     </div>
46   </div>
47 </div>

```

Figura 3.7.4.7 Código de interfaz tipo importe del gerente.

```

employee_price_import.php X
CoreCaja > employee_price_import.php > ...
1  <?php
2  session_start();
3  $error_message = isset($_SESSION['error_message']) ? $_SESSION['error_message'] : "";
4  unset($_SESSION['error_message']);
5
6  include 'includes/header.php';
7  include 'includes/conexion_db.php';
8  include 'employee_price_import_query.php';
9
10 $adminID = $_SESSION['admin_admin_id'];
11 $tiposImp = tiposImportesRegistrados($pdo, $adminID);
12
13 ?>
14
15 <!-- Formulario para generar gastos -->
16 <div class="container my-5">
17   <div class="row justify-content-center">
18     <div class="col-md-8">
19       <div class="card shadow-lg rounded-4 border-0">
20         <div class="card-body p-4 bg-light">
21           <h1 class="text-center fw-bold mb-1">Registrar Tipo Importe</h1>
22           <div class="card-body">
23             <form action="employee_price_import_submit.php" method="post">
24
25               <!-- Nombre del tipo de importe -->
26               <div class="input-group mb-3">
27                 <span class="input-group-text"><i class="fas fa-tag"></i></span>
28                 <div class="form-floating flex-grow-1">
29                   <input type="text" class="form-control" id="nombre_importe" name="nombre_importe" required placeholder="Ingresa el nombre del tipo de importe">
30                   <label for="nombre_importe">Nombre del tipo de importe</label>
31                 </div>
32               </div>
33
34               <!-- Precio unitario del tipo de importe -->
35               <div class="input-group mb-3">
36                 <span class="input-group-text"><i class="fas fa-money-bill-alt"></i></span>
37                 <div class="form-floating flex-grow-1">
38                   <input type="number" class="form-control" id="precio_importe" name="precio_importe" required placeholder="Ingresa la cantidad unitaria del tipo de importe" min="1">
39                   <label for="precio_importe">Precio unitario del importe</label>
40               </div>
41             </div>
42           </div>
43         </div>
44       </div>
45     </div>
46   </div>
47 </div>

```

Figura 3.7.4.8 Código de interfaz tipo importe del empleado.

```
employee_price_import_query.php X
CorteCaja > employee_price_import_query.php > ...
1  <?php
2
3 > function tiposImportesRegistrados($pdo, $adminID) { ...
14  }
15
16  include 'includes/conexion_db.php';
17
18 // PROCESAR SOLICITUDES
19 > if ($_SERVER["REQUEST_METHOD"] === "POST") { ...
52  }
53
54 // FUNCION PARA ELIMINAR TIPO IMPORTE
55 function eliminarTipoImporte($pdo, $cimp_id, $adminID)
56 > { ...
63  }
64
65 // FUNCION PARA ACTUALIZAR TIPO IMPORTE
66 function actualizarTipoImporte($pdo, $cimp_id, $adminID, $cimp_precio, $cimp_nombre)
67 > { ...
86  }
87
88 ?>
```

Figura 3.7.4.9 Controlador tipo importe del empleado.

```
admin_price_import_query.php X
CorteCaja > admin_price_import_query.php > ...
1  <?php
2
3 > function tiposImportesRegistrados($pdo, $adminID) { ...
14  }
15
16  include 'includes/conexion_db.php';
17
18 // PROCESAR SOLICITUDES
19 > if ($_SERVER["REQUEST_METHOD"] === "POST") { ...
52  }
53
54 // FUNCION PARA ELIMINAR TIPO IMPORTE
55 function eliminarTipoImporte($pdo, $cimp_id, $adminID)
56 > { ...
63  }
64
65 // FUNCION PARA ACTUALIZAR TIPO IMPORTE
66 function actualizarTipoImporte($pdo, $cimp_id, $adminID, $cimp_precio, $cimp_nombre)
67 > { ...
86  }
87
88 ?>
```

Figura 3.7.4.10 Controlador tipo importe del gerente.

```
employee_price_import_submit.php X
CorteCaja > employee_price_import_submit.php > ...
1  <?php
2
3  session_start();
4
5  include 'includes/conexion_db.php';
6
7  $error_message = "";
8
9  if ($_SERVER["REQUEST_METHOD"] == "POST") {
10    // Recuperar el emp_id de la sesión
11    $adminID = $_POST['admin_id'];
12    $cimp_nombre = $_POST['nombre_importe'];
13    $cimp_precio = (int)$_POST['precio_importe'];
14    $cimp_nombre = strtoupper($cimp_nombre); // Convertir a mayúsculas antes de verificar en la BD
15
16    // Validar que los campos no estén vacíos
17    if (empty($adminID) || empty($cimp_nombre) || empty($cimp_precio)) {
18      // Redirigir al formulario con un mensaje de error
19      header("Location: employee_price_import.php?error=empty_fields");
20      exit();
21    }
22    try {
23
24      // Verificar si el tipo importe ya existe
25      $sql = "SELECT COUNT(*) FROM categorias_importes WHERE cimp_nombre = :cimp_nombre AND admin_admin_id = :admin_id";
26      $stmt = $pdo->prepare($sql);
27      $stmt->bindValue(':cimp_nombre', $cimp_nombre);
28      $stmt->bindValue(':admin_id', $adminID);
29      $stmt->execute();
30      $existeImporte = $stmt->fetchColumn();
31
32      if ($existeImporte == 0) {
33
34        // Insertar el nuevo tipo importe en la tabla categorias_importes
35        $insertSql = "INSERT INTO categorias_importes (cimp_nombre, admin_admin_id) VALUES (:cimp_nombre, :admin_id)";
36        $insertStmt = $pdo->prepare($insertSql);
37        $insertStmt->bindValue(':cimp_nombre', $cimp_nombre);
38        $insertStmt->bindValue(':admin_id', $adminID);
39        $insertStmt->execute();
40
41        // Redirigir al formulario con un mensaje de éxito
42        header("Location: employee_price_import.php?success=imported");
43        exit();
44      }
45    } catch (Exception $e) {
46      // Manejar errores
47      $error_message = "Ocurrió un error al intentar importar el tipo de importe. Por favor, inténtalo de nuevo más tarde." . $e->getMessage();
48    }
49  }
50
51  // Redirigir al formulario con un mensaje de error
52  header("Location: employee_price_import.php?error=empty_fields");
53  exit();
54 }
```

Figura 3.7.4.11 Backend para el registro tipo importe del empleado.

admin_price_import_submit.php

```
CorteCaja > admin_price_import_submit.php > ...
1 <?php
2
3 session_start();
4
5 include 'includes/conexion_db.php';
6
7 $error_message = "";
8
9
10 if ($_SERVER["REQUEST_METHOD"] == "POST") {
11     // Recuperar el emp_id de la sesión
12     $adminID = $_POST['admin_id'];
13     $cimp_nombre = $_POST['nombre_importe'];
14     $cimp_precio = (int)$_POST['precio_importe'];
15     $cimp_nombre = strtoupper($cimp_nombre); // Convertir a mayúsculas antes de verificar en la BD
16
17     // Validar que los campos no estén vacíos
18     if (empty($adminID) || empty($cimp_nombre) || empty($cimp_precio)) {
19         // Redirigir al formulario con un mensaje de error
20         header("Location: admin_price_import.php?error=empty_fields");
21         exit();
22     }
23     try {
24
25         // Verificar si el tipo importe ya existe
26         $sql = "SELECT COUNT(*) FROM categorias_importes WHERE cimp_nombre = :cimp_nombre AND admin_admin_id =";
27         $stmt = $pdo->prepare($sql);
```

Figura 3.7.4.11 Backend para el registro tipo importe del gerente.

3.8 Registrar, visualizar y eliminar productos faltantes.

3.8.1 Descripción del caso de uso

La descripción del caso de uso es registrar, visualizar y eliminar los productos faltantes, este caso surge para resolver la problemática en otros establecimientos, no directamente en la tienda Santana Elia, pero la implementación de esta herramienta, conforme se conozca su funcionalidad, será usada con frecuencia, dicha problemática surge al momento que el empleado percibe que un producto está por finalizar, el empleado manda un mensaje a un grupo de WhatsApp exclusivo para dar esta información, sin embargo, se depende de este grupo para la organización de solicitar el producto y marcar como pendiente finalizado, lo cual puede ser abrumador la consulta en este grupo, dado la cantidad de mensajes puede provocar la confusión de las mismas solicitudes.

El sistema SIGMA (Sistema Integral de Gestión para Microempresas de Abarrotes) tiene la herramienta similar pero resultando tener una mayor eficiencia y organización, enfocándose en lo siguiente:

- Se agrega un nombre del producto con pocas unidades junto con la cantidad aproximada o exacta de dicho producto faltante.
- El registro le da prioridad para mostrar al principio los productos con menos unidades.
- Se puede marcar como completado, y este se mostrará en un pequeño historial de 3 últimos registros, logrando observar quien marcó como completado.
- En caso de equivocarse, el registro se puede eliminar.

Sin embargo, el gerente no tiene todo el acceso a estas herramientas, específicamente en el registro de productos faltantes, esto por la razón que los empleados son quienes están en observación de los productos, conocen aquellos que están por finalizar su existencia, mientras que el gerente es quien genera el pedido a los proveedores.

Dicha opción se muestra en el menú de opciones, la cual se nombró como “Producto Faltante” (véase Figura 3.8.1.1 y Figura 3.8.1.2).

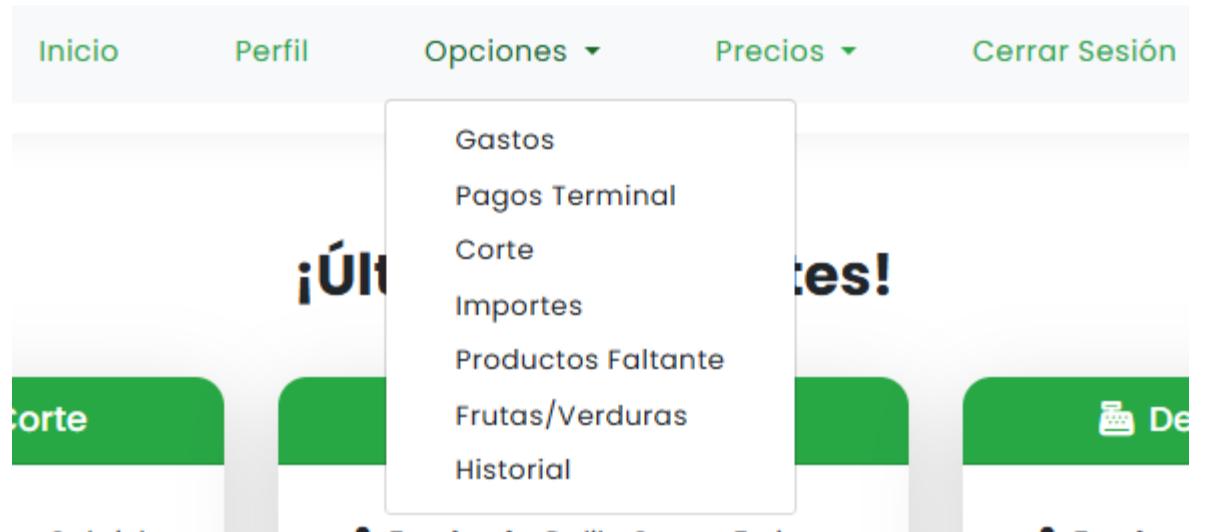


Figura 3.8.1.1 Opción “Productos Faltantes” en el menú de empleados.



Figura 3.8.1.2 Opción “Productos Faltantes” en el menú de gerentes.

3.8.2 Modelo de datos

Para la realización del registro, visualización y eliminación de productos faltante, se involucran 3 tablas (véase Figura 3.8.2.1), las cuales son las siguientes:

- admin: Esta tabla se involucra como clave foránea, para la separación entre futuras otras tiendas o negocios, además, en la opción marcar como completado, también se registrará si el gerente fue quien lo marco.

- empleados: Esta tabla se involucra como clave foránea, al momento de registrar un producto faltante y en la opción marcar como completado.
- productos_faltantes: Siendo la tabla donde se maneja en su mayoría el manejo de datos en esta tabla, donde se insertan los datos necesarios para su manejo, además, se creó de tal manera que pudiera manejar un historial.

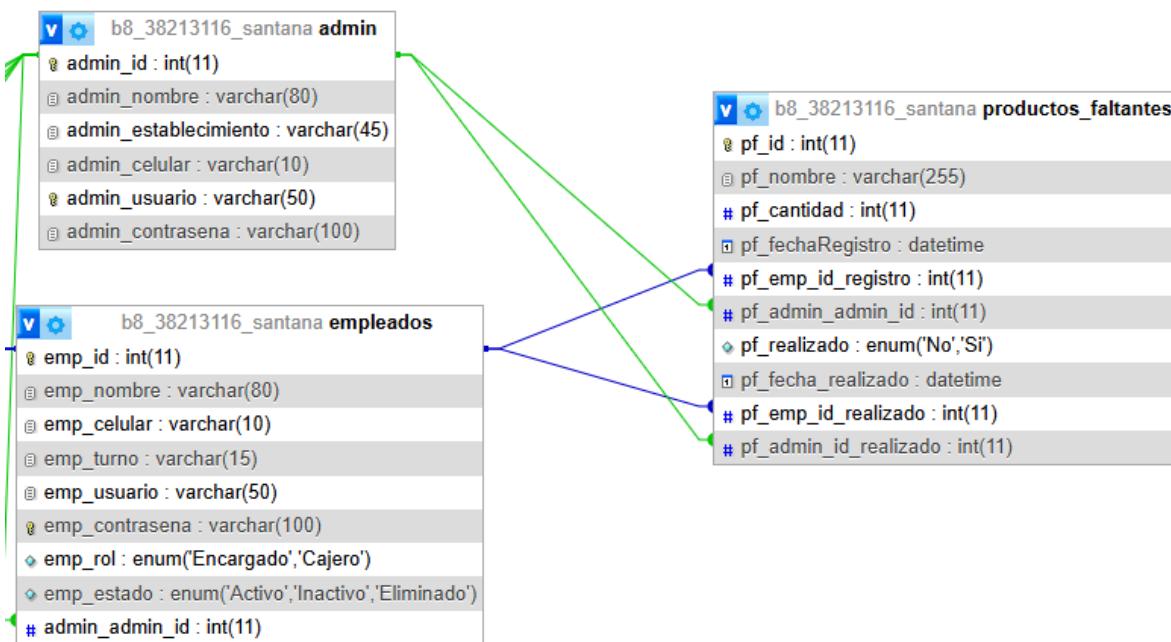


Figura 3.8.2.1 Tablas involucradas para registrar, visualizar y eliminar un producto faltante.

3.8.3 Diseño de la interfaz

El diseño de la interfaz para el empleado mantiene el orden y objetividad (véase Figura 3.8.3.1), buscando ser similar a los anteriores, por ello, se tomó las siguientes decisiones:

- Formulario similar: Este formulario se diseñó en base a los anteriores, tratando de ser lo más familiarizado posible, para que el usuario no

tenga la necesidad de realizar acciones distintas para realizar un registro.

- Tabla para mostrar registros: El diseño es familiar para el empleado, lo cual resulta ser sencillo de interpretar, se ordenan entre menos piezas existentes, se mostrarán al principio.
- Tabla para mostrar historial: En esta tabla es sencillo para su interpretación, mostrando solo máximo 3 resultados recientes, también siendo lo menos invasivo en toda la información.
- Card en la página principal: Se muestran tarjetas en el index del empleado, con la finalidad de que sea informativo (véase Figura 3.8.3.2).
- El empleado podrá marcar como completado o eliminar algún registro.

De la misma manera que lo anterior visto, para el gerente tenemos únicamente la visualización de los productos faltantes (véase Figura 3.8.3.3), esto para evitar registros duplicados:

- Marcar como realizado: El gerente quien realiza el pedido con los proveedores, podrá marcar como completado el pedido
- Eliminar registró: El gerente también podrá eliminar algún registro.
- Como anteriormente se comentó, se muestra un historial de productos pendientes resueltos, además, de visualizar los productos faltantes aún pendientes.
- En el index del gerente, también podrá observar los productos faltantes (véase Figura 3.8.3.4).

The diagram illustrates the user interface for managing missing products through three overlapping windows:

- Top Window:** A modal titled "Registrar Producto Faltante" (Register Missing Product) contains fields for "Nombre del producto faltante" (Missing product name) and "Cantidad existente" (Existing quantity), both with placeholder icons. A green "Agregar" (Add) button is at the bottom.
- Middle Window:** A list titled "Productos Faltantes Solucionados" (Solved Missing Products) shows one entry: "DELAWER" marked as solved by "Cesar" on "05/06 05:38 PM".
- Bottom Window:** A list titled "Productos Faltantes Registrados" (Registered Missing Products) shows one entry: "NEW MIX CANTARITO" with a quantity of "2" registered by "Cesar" on "05/06 05:38 PM". It includes a green checkmark and a red delete icon in the actions column.

Figura 3.8.3.1 Diseño de la interfaz “Productos Faltantes” del empleado.



Figura 3.8.3.2 Diseño de la interfaz “Index” del empleado.

Productos Faltantes Solucionados

Producto	Marcado Por	Fecha Atendido
DELAWER	Cesar	05/06 05:38 PM

Productos Faltantes Registrados

Producto Faltante	Cantidad Restante	Registrado Por	Fecha de Registro	Acciones
NEW MIX CANTARITO	2	Cesar	05/06 05:38 PM	
COCA COLA 2.5 L	6	Cesar	05/06 05:37 PM	
AZUCAR BOLSAS 1 KG	7	Cesar	05/06 05:38 PM	

Figura 3.8.3.3 Diseño de la interfaz “Productos Faltantes” del gerente.

Productos Faltantes

NEW MIX CANTARITO 2 piezas	COCA COLA 2.5 L 6 piezas	AZUCAR BOLSAS 1 KG 7 piezas	Más información Presiona aquí
-------------------------------	-----------------------------	--------------------------------	--

Figura 3.8.3.4 Diseño de la interfaz “Index” del gerente.

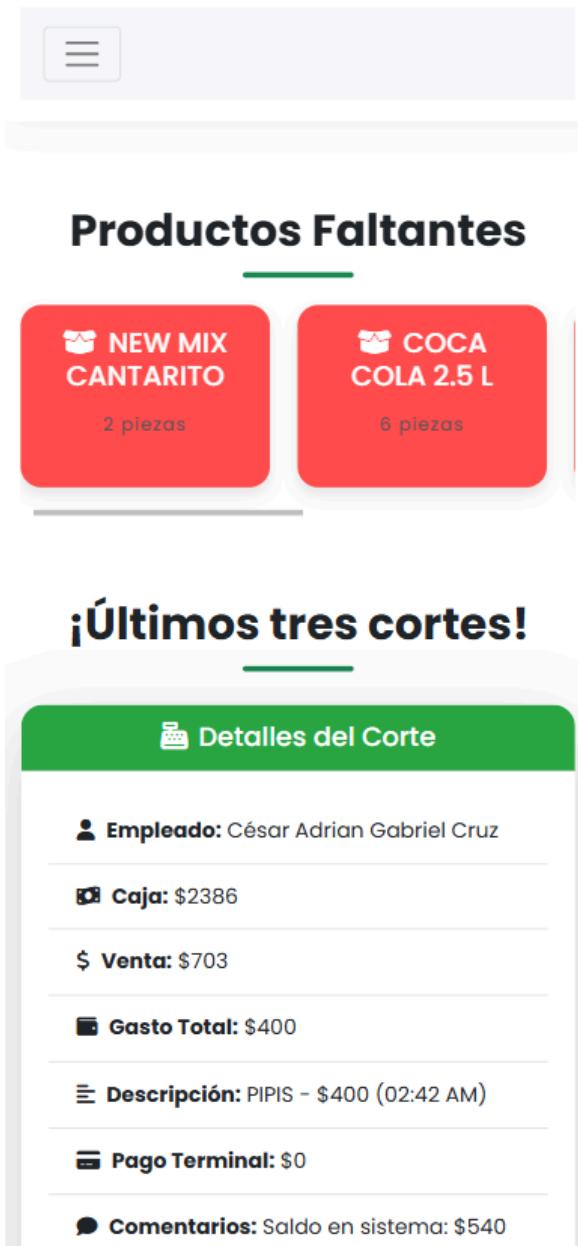


Figura 3.8.3.5 Interfaz “Index” del gerente desde un dispositivo móvil.

Productos Faltantes Solucionados

Producto	Marcado Por	Fecha Atendido
DELAWER	Cesar	05/06 05:38 PM

Productos Faltantes Registrados

Producto Faltante	Cantidad Restante	Registrado Por
NEW MIX CANTARITO	2	Cesar

Figura 3.8.3.6 Interfaz “Productos Faltantes” del gerente desde un dispositivo móvil.

3.8.4 Fragmentos de código

En el fragmento del código, se dividieron en varios archivos PHP, esto debido a la seguridad y separación de funciones, además, también se dividieron por actor, las cuales se describirán a continuación:

- **Interfaz gráfica empleado:** Este archivo se nombró como employee_generate_product.php (véase Figura 3.8.4.1), el código mantiene la interfaz gráfica, como en otras interfaces, dejamos al

principio un formulario para registrar el producto faltante y las cantidades existentes, esto nos funciona para identificar rápidamente el producto y según las cantidad, darle dicha prioridad. Tenemos un historial para mostrar los productos faltantes ya resueltos, pero únicamente los tres más recientes, para que este campo no abarque mucho espacio y sea abrumador para el empleado o gerente, esto con la finalidad de tener conocimiento de aquello que ya se soluciono y quien dio por finalizado dicho pendiente. También tenemos una tabla para mostrar todos los pendientes aún sin solucionar, esto con los campo necesarios para entender el registro, como lo es el nombre del producto faltante, la cantidad, que empleado realizó el registro en caso de posibles dudas, la fecha del registro y las accione para eliminar o marcar como realizado el registro.

- Funciones para empleado: Este archivo se nombró como employee_generate_product_pfquery.php (véase Figura 3.8.4.2), su funcionalidad es almacenar funciones que se requieren para la interfaz gráfica y la base de datos, las cuales se describen a continuación:
 - Obtener productos registrados: Esta función llamada obtenerProductosRegistrados() como su nombre indica, obtiene todos los productos registrados que aún no se han marcado como realizado y los correspondientes a dicho gerente, esto para evitar datos de otras microempresas de abarrotes, se obtiene el id del producto faltante, nombre, la cantidad, fecha del registro, el empleado que realizó el registro.
 - Obtener los productos registrados marcados como realizados: Esta función llamada

obtenerProductosRegistradosAtendidos() tiene como objetivo mandar a llamar el historial, con los datos del nombre del producto faltante, así como la fecha en que se marcó como realizado y el usuario que lo marcó como realizado, esto contemplando que el registro ya este como realizado, además, se ordena por fecha siendo el más reciente primero y se limita a obtener sólo tres registros.

- Eliminar producto faltantes: Esta función llamada eliminarProductoRealizado(), cumple su función de eliminar los registros que se marcaron como realizado, con un periodo mayor a tres días, esto para no acumular datos innecesarios, esta función se manda a llamar cada ocasión que el empleado o el gerente entra a la interfaz gráfica de los productos faltantes.
- Marcar como realizado: Se nombró marcarComoRealizado(), esta función se ejecuta cuando el empleado o gerente marca como realizado algún registro de la tabla de productos faltantes, lo cual actualiza el registro en la base de datos.
- Eliminar producto faltante: eliminarProductoFaltante(), es la función más fácil de todas, esta se manda a llamar cuando el empleado o gerente presiona el botón de eliminar registro y este manda el id del producto faltante a la función de delete.
- Agregar registro para empleado: Este archivo se nombró como employee_generate_product_submit.php (véase Figura 3.8.4.3), como lo hemos mencionado anteriormente, el empleado es el único que puede hacer registros, a su vez, una vez completado en formulario, mediante el uso de post, se envía al archivo con

finalización submit, y este obtiene los datos así como verificando que realmente existan datos, en caso de que sí, se hace el insert en la tabla con los datos o campos correspondientes.

- Interfaz gráfica gerente: Este archivo se nombró como admin_generate_product.php (véase Figura 3.8.4.4), en este, mantiene la misma estructura que la interfaz del empleado, sin embargo, se eliminó el campo del formulario para agregar registro.
 - Funciones para el gerente: Este archivo se nombró como admin_generate_pfquery.php (véase Figura 3.8.4.5), la funcionalidad de este, son las mismas que las del empleado, sin embargo, dado las variables de sesión, las funciones o consultas realizan un cambio aquí. Además, al momento de marcar como realizado, la consulta también tiene su diferencia al registrar al gerente, dado que es una columna distinta a como si lo hiciera un empleado

```
employee_generate_product.php
CortoCaso > employee_generate_product.php > div.container.mt-5 > div.row.justify-content-center > div.col-md-8 > div.card > div.card-body > form > div.form-group.input-group > div.input-group-prepend
  1 $productos = obtenerProductosRegistrados($pdo);
  2 $productosResultados = obtenerProductosRegistradosAtendidos($pdo);
  3 $eliminados = eliminarProductoFaltanteAtendido($pdo);
  4
  5
  6 <!-- Formulario para generar productos -->
  7 <div class="container mt-5">
  8   <div class="row justify-content-center">
  9     <div class="col-md-8">
 10       <div class="card">
 11         <h1 class="text-center mt-4">Registrar Producto Faltante</h1>
 12         <div class="card-body">
 13           <form action="employee_generate_product_submit.php" method="post">
 14
 15             <div class="form-group input-group">
 16               <div class="input-group-prepend">
 17                 <span class="input-group-text"><i class="fas fa-shopping-cart"></i></span>
 18               </div>
 19               <input type="text" class="form-control" id="nombre" name="nombre" required placeholder="Ingresa el nombre del producto faltante">
 20             </div>
 21
 22             <div class="form-group input-group">
 23               <div class="input-group-prepend">
 24                 <span class="input-group-text"><i class="fas fa-cubes"></i></span>
 25               </div>
 26               <input type="number" class="form-control" id="cantidad" name="cantidad" required placeholder="Ingresa la cantidad del producto existente" min="0">
 27             </div>
 28
 29             <!-- Alerta Campos Vacíos -->
 30             <php if (isset($_GET['error']) && $_GET['error'] == 'empty_fields') : >>
 31               <div class="alert alert-danger mt-3" role="alert">
 32                 Los campos deben completarse.
 33               </div>
 34             <php endif; ?>
 35
 36             <button type="submit" class="btn btn-primary"><i class="fas fa-plus"></i> Agregar</button>
 37           </form>
 38         </div>
 39       </div>
 40     </div>
 41   </div>
 42 </div>
 43
 44 <!-- Tabla de Productos Atendido -->
 45 <div class="row justify-content-center">
 46   <div class="col-md-8">
 47     <div class="card">
 48       <h1 class="text-center mt-4">Productos Faltantes Solucionados</h1>
 49       <div class="card-body">
 50         <php if (empty($productosResultados)) : >>
 51           <p>No se han encontrado resultados para esta búsqueda.
 52         </php>
 53       </div>
 54     </div>
 55   </div>
 56 </div>
```

Figura 3.8.4.1 Código de la interfaz gráfica del empleado.

```

employee_generate_pfquery.php

1 <?php
2
3 include 'includes/conexion_db.php';
4
5 function obtenerProductosRegistrados($pdo)
6 {
7     $adminID = $_SESSION['admin_admin_id'];
8
9     $sql = "SELECT pf.id_pf, pf.nombre, pf.cantidad, pf.fechaRegistro, pf.emp_id_registro, e.emp_usuario
10    FROM productos_faltantes pf
11    LEFT JOIN empleados e ON pf.emp_id_registro = e.emp_id
12   WHERE pf_pf.admin_id = :adminID AND pf_pf.realizado = 'No'
13   ORDER BY pf.cantidad ASC";
14
15 $stmt = $pdo->prepare($sql);
16 $stmt->bindParam(':adminID', $adminID, PDO::PARAM_INT);
17 $stmt->execute();
18
19 return $stmt->fetchAll(PDO::FETCH_ASSOC);
20 }
21
22 function obtenerProductosRegistradosAtendidos($pdo)
23 {
24     $adminID = $_SESSION['admin_admin_id'];
25
26     $sql = "SELECT
27         pf.p_nombre,
28         pf_pf.fecha_realizado,
29         COALESCE(e.emp_usuario, a.admin_usuario) AS usuario_realizado
30     FROM productos_faltantes pf
31     LEFT JOIN empleados e ON pf.emp_id_registro = e.emp_id
32     LEFT JOIN administradores a ON pf_pf.admin_id = a.admin_id
33   WHERE pf_pf.admin_id = :adminID
34   AND pf_pf.realizado = 'Si'
35   ORDER BY pf_pf.fecha_realizado DESC
36   LIMIT 3";
37
38 $stmt = $pdo->prepare($sql);
39 $stmt->bindParam(':adminID', $adminID, PDO::PARAM_INT);
40 $stmt->execute();
41
42 return $stmt->fetchAll(PDO::FETCH_ASSOC);
43 }
44
45 function eliminarProductoFaltanteAtendido($pdo)
46 {
47     $adminID = $_SESSION['admin_admin_id'];
48
49     $sql = "DELETE FROM productos_faltantes
50   WHERE pf_realizado = 'Si' AND pf_fecha_realizado < NOW() - INTERVAL 3 DAY
51   AND pf_admin_admin_id = :adminID";
52
53 $stmt = $pdo->prepare($sql);
54 $stmt->bindParam(':adminID', $adminID, PDO::PARAM_INT);
55
56 return $stmt->execute();
57 }
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
306
307
307
308
309
309
310
311
312
313
314
315
315
316
317
317
318
319
319
320
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1
```

```
 9 <?php admin_generate_product.php ?>
10
11 CorreCapa > admin_generate_product.php > div.container.mt-5 > div.row.justify-content-center > div.col-md-8 > div.card > div.card-body > div.table-responsive > table.table-bordered.table-striped > tbody > tr > td > form
12
13 $productos = obtenerProductosRegistrados($pdo);
14 $productosResueltos = obtenerProductosRegistradosAtendidos($pdo);
15 $eliminados = eliminarProductoFaltanteAtendido($pdo);
16
17 >>>
18 <!-- Tabla de Productos Atendido -->
19 <div class="container mt-5">
20   <div class="row justify-content-center">
21     <div class="col-md-8">
22       <div class="card">
23         <div class="text-center mt-4" >Productos Faltantes Solucionados</h1>
24         <div class="card-body">
25           <?php if (empty($productosResueltos)) : ?>
26             <div class="alert alert-info text-center" role="alert">
27               No hay productos faltantes solucionados recientemente.
28             </div>
29           <?php else : ?>
30             <div class="table-responsive">
31               <table class="table table-bordered table-striped">
32                 <thead class="table-success">
33                   <tr>
34                     <th>Producto</th>
35                     <th>Marcado Por</th>
36                     <th>Fecha Atendido</th>
37                   </tr>
38                 </thead>
39                 <tbody>
40                   <?php foreach ($productosResueltos as $productoResuelto) : ?>
41                     <tr>
42                       <td>?php echo $productoResuelto['pf_nombre']; ?</td>
43                       <td>?php echo $productoResuelto['usuario_realizado']; ?</td>
44                       <td>?php echo date("m/d h:i A", strtotime($productoResuelto['pf_fecha_realizado'])); ?</td>
45                     </tr>
46                   <?php endforeach; ?>
47                 </tbody>
48               </table>
49             </div>
50           </div>
51         </div>
52       </div>
53     </div>
54   </div>
55
56 <!-- Tabla de Productos Registrados -->
57 <div class="container mt-5">
58   <div class="row justify-content-center">
59     <div class="col-md-8">
60       <div class="card">
61         <div class="text-center mt-4" >Productos Faltantes Registrados</h1>
62         <div class="card-body">
63           <?php if (empty($productos)) : ?>
```

Figura 3.8.4.4 Código de la interfaz gráfica del gerente.

```
admin_generate_pfqesq.php <

ConteCaja > ▾ admin_generate_pfqesq.php > ⌂ marcarComoRealizado
  ↳php
  ↳
  ↳include 'includes/conexion_db.php';
  ↳
  ↳function obtenerProductosRegistrados($pdo)
  ↳{
  ↳    $adminID = $_SESSION['admin_id'];
  ↳
  ↳    $sql = "SELECT pf_id, pf_nombre, pf_cantidad, pf_fechaRegistro, pf_pf_id_registro, e.emp_usuario
  ↳          FROM productos_faltantes pf
  ↳          JOIN empleados e ON pf_pf_id_registro = e.emp_id
  ↳          WHERE pf_pf_admin_admin_id = :adminID AND pf_pf_realizado = 'No'
  ↳          ORDER BY pf_pf_cantidad ASC";
  ↳
  ↳    $stmt = $pdo->prepare($sql);
  ↳    $stmt->bindParam(':adminID', $adminID, PDO::PARAM_INT);
  ↳    $stmt->execute();
  ↳
  ↳    return $stmt->fetchAll(PDO::FETCH_ASSOC);
  ↳}
  ↳
  ↳function obtenerProductosRegistradosAtendidos($pdo)
  ↳{
  ↳    $adminID = $_SESSION['admin_id'];
  ↳
  ↳    $sql = "SELECT
  ↳            pf_pf_nombre,
  ↳            pf_pf_fechaRealizado,
  ↳            e.emp_usuario,
  ↳            e.emp_id,
  ↳            a.admin_usuario AS usuario_realizado
  ↳        FROM productos_faltantes pf
  ↳        LEFT JOIN empleados e ON pf_pf_id_registro = e.emp_id
  ↳        LEFT JOIN admin a ON pf_pf_admin_id_realizado = a.admin_id
  ↳        WHERE pf_pf_admin_admin_id = :adminID
  ↳        AND pf_pf_realizado = 'Si'
  ↳        ORDER BY pf_pf_fechaRealizado DESC
  ↳        LIMIT 3";
  ↳
  ↳    $stmt = $pdo->prepare($sql);
  ↳    $stmt->bindParam(':adminID', $adminID, PDO::PARAM_INT);
  ↳    $stmt->execute();
  ↳
  ↳    return $stmt->fetchAll(PDO::FETCH_ASSOC);
  ↳}
  ↳
  ↳function eliminarProductoFaltanteAtendido($pdo)
  ↳{
  ↳    $adminID = $_SESSION['admin_id'];
  ↳
  ↳    $sql = "DELETE FROM productos_faltantes
  ↳          WHERE pf_realizado = 'Si' AND pf_fecha_realizado < NOW() - INTERVAL 3 DAY
  ↳          AND pf_pf_admin_admin_id = :adminID";
  ↳
  ↳    $stmt = $pdo->prepare($sql);
  ↳    $stmt->bindParam(':adminID', $adminID, PDO::PARAM_INT);
  ↳    $stmt->execute();
  ↳}
```

Figura 3.8.4.5 Funciones para la operabilidad del gerente.

3.9 Cerrar sesión

3.9.1 Descripción del caso de uso

La descripción del caso de uso el cierre de una sesión, siendo una de las funciones más básicas, la cual es terminar con la sesión, esta opción se encuentra en el menú desplegable al final o último “Cerrar Sesión” (véase Figura 3.9.1.1 y Figura 3.9.1.2), en cual destruye las variables de sesión y redirige al usuario al index general.

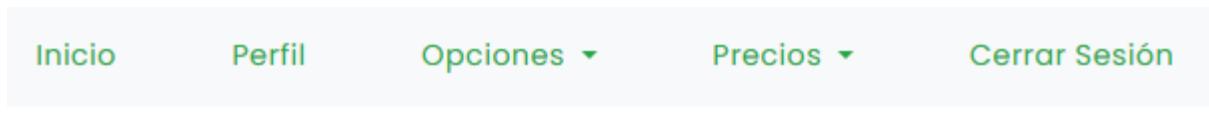


Figura 3.9.1.1 Opción “Cerrar Sesión” del menú del empleado.

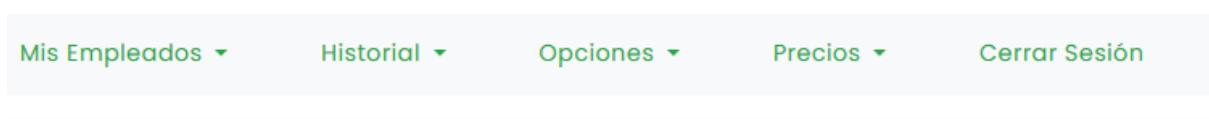


Figura 3.9.1.2 Opción “Cerrar Sesión” del menú del gerente.

3.9.2 Modelo de datos

Las tablas usadas para este caso de uso, se consideran ser las de empleados y admin (véase Figura 3.9.2.1), debido a que parte de estas son variables de sesión, la cual al finalizar o cerrar sesión estas simplemente son destruidas.

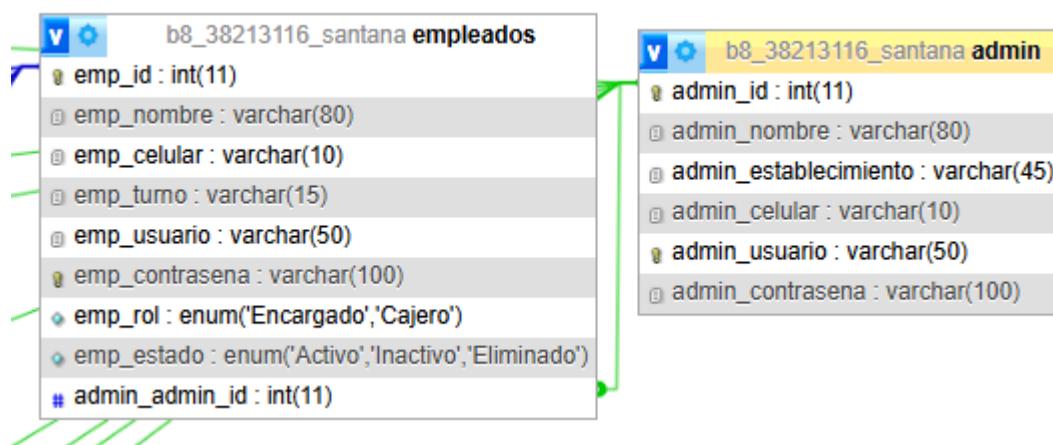


Figura 3.9.2.1 Modelo de datos.

3.9.3 Diseño de la interfaz de registro de empleados

El diseño de la interfaz se compone únicamente del botón “Cerrar Sesión” (véase Figura 3.9.3.1) en el header o menú desplegable.



Cerrar Sesión

Figura 3.9.2.1 Modelo de datos.

3.9.4 Fragmentos de código

El diseño de la interfaz se compone únicamente del botón “Cerrar Sesión” en el header o menú desplegable y la destrucción de las variables.

- Tenemos el archivo header.php (véase Figura 3.9.4.1) donde se encuentra el menú desplegable, en este para ambos usuarios tenemos el botón de cerrar sesión, lo cual redirige la sesión al logout.php.
- Tenemos el archivo logout.php (véase Figura 3.9.4.2) el cual finaliza la sesión destruyendo las variables de sesión y reenviando al usuario al index general.

```


1  <?php
2  session_start(); // Iniciar la sesión
3
4  // Destruir todas las variables de sesión
5  $_SESSION = array();
6
7  if (ini_get("session.use_cookies")) {
8      $params = session_get_cookie_params();
9      setcookie(
10          session_name(),
11          '',
12          time() - 42000,
13          $params["path"],
14          $params["domain"],
15          $params["secure"],
16          $params["httponly"]
17      );
18  }
19
20 // Destruir la sesión
21 session_destroy();
22
23 header("Location: index.php");
24 exit;
25
26


```

Figura 3.9.4.1 Código del header donde se encuentra el botón.

```


1  <?php
2  session_start(); // Iniciar la sesión
3
4  // Destruir todas las variables de sesión
5  $_SESSION = array();
6
7  if (ini_get("session.use_cookies")) {
8      $params = session_get_cookie_params();
9      setcookie(
10          session_name(),
11          '',
12          time() - 42000,
13          $params["path"],
14          $params["domain"],
15          $params["secure"],
16          $params["httponly"]
17      );
18  }
19
20 // Destruir la sesión
21 session_destroy();
22
23 header("Location: index.php");
24 exit;
25
26


```

Figura 3.9.4.2 Backend para finalizar la sesión.

CAPÍTULO 4: RESULTADOS

4.1 Resultados de las pruebas

La aplicación web comenzó en uso el día 10 de marzo 2025, esta presentó varias situaciones no cruciales para el funcionamiento de la misma, además, de mejoras dadas por los usuarios empleados de la tienda Santana Elia, las cuales lograron enriquecer dicha experiencia de uso, a continuación se describirán las situaciones y su respectiva corrección.

- Correcciones:
 - Registro de usuarios en admin_register_submit.php y admin_employee_register_submit.php: Los usuarios que se registraban tenían una separación entre tablas de empleados y admin, es decir, al momento de registrar un gerente, este sólo se verificaba que no se repitiera el usuario en la tabla admin, a su vez, al registrar un empleado, se verificaba únicamente que el usuario no se repitiera en la tabla empleados. Su solución fue al momento de registrar un usuario siendo este gerente o empleado, tener un filtro el cual verifique en ambas tablas la existencia de dicho usuario, esto se logra mediante una consulta unida con otra consulta (véase Figura 4.1.1), como se menciona busca el usuario en ambas tablas, en caso de encontrar similitud, no realiza el registro y muestra la interfaz gráfica un mensaje de usuario existente.
 - Login en login_submit.php: Antes de la corrección anterior, el usuario al loguearse debería seleccionar en un selector si el usuario era gerente o empleado, para al momento de realizar la consulta de si el usuario si existe, pudiese realizarla en la tabla correcta, y dar o negar el acceso. Esto

logró corregirse mediante la realización de verificar primero si el usuario existe en tabla admin, si se encuentra, se crean las variables de sesión y se da acceso al sistema, en caso de no ser así, se busca en la tabla empleados, en caso de encontrarlo, verifica que el empleado tenga un estado “Activo” y si lo es crea las variables de sesión y da acceso al sistema, en caso de que el usuario no fue encontrado en ninguna base de datos (véase Figura 4.1.2).

- Registro de gastos nuevos en employee_generate_purchases_submit.php: El sistema ya trae consigo gastos predeterminados, siendo estos los más comunes en una tienda de abarrotes, sin embargo, en la tabla categorias_gastos (véase Figura 4.1.3) dentro de la base de datos tenemos una columna admin_id, la cual liga esa categoría a cierto establecimiento, sin embargo, al registrar este nuevo gasto la consulta anterior solo verificaba si esta categoría ya existía en relación al id del gerente, por lo cual como las categorías predeterminadas no tienen un id definido, se creaban registros duplicados. Esto se solucionó con una modificación en dicha consulta, al momento del where aparte de verificar el nombre, agregaremos un AND y entre paréntesis comprobando que el id del gerente sea el id al que está ligado el empleado o este sea nulo (véase Figura 4.1.4), es decir, si este nuevo gasto ya existe en el establecimiento o es de los predeterminados, el registro no será ejecutado.
- Problemas de horario: De manera general en el proyecto, al realizar un registro en la mayoría de tablas tienen una columna de fecha y hora, aquí surge el problema o

inconveniente, al momento de las pruebas la fecha y hora eran correctas al realizar la inserción con la consulta se usaba la propiedad NOW() en su respectiva columna, sin embargo, al montar la aplicación web en el servidor ByetHost este mantenía una zona horaria distinta, la cual era menos dos horas, por ejemplo, si en el establecimiento Santana Elia realizaba una inserción a las 11 pm, en la aplicación web aparecía con la hora 9 pm, se usó una solución básica donde a la inserción de la fecha y hora se usaba NOW() + INTERVAL 2 HOURS, lo que lograba era sumarle 2 horas a las 9 pm tomando el ejemplo anterior, lo cual la hora sería la correcta, sin embargo en el cambio de horario, ahora los registros se realizaban con una hora más a la que debía ser, es decir, en Santana Elia se realizaba el registro a las 11 pm, y en el sistema marcaba a las 12 am. La solución definitiva fue usar date_default_timezone_set (véase Figura 4.1.5) lo cual toma la zona horaria de la Ciudad de México y después le da el formato para la base de datos. Esto fue actualizado a todos los archivos que generan una inserción de fecha y hora en la base de datos.

- Verificar que el envío de datos al submit: en la mayoría de archivos submit, las cuales registran los datos recibidos en la base de datos, sin embargo, al tener campos de textos con la propiedad required, obliga al usuario a no dejar vacío el campo de texto, sin embargo, al presionar click derecho y seleccionamos la opción “Inspeccionar”, podemos modificar el código, logrado quitar esta propiedad required, si esto sucede, al momento de llegar al submit en caso de no enviar la información correspondiente, esto genera un error,

debido a que no se envían datos para ser registrados. La solución como se mencionó en muchas ocasiones en los casos de uso, es guardar la información recibida mediante método post en variables, después se declara una condicional en la cual si una variable está vacía, el registro no se realiza y se envía un mensaje a la interfaz gráfica, se adjunta un ejemplo del archivo employee_generate_expenses_submit.php (véase Figura 4.1.6).

- Mejoras:

- Caja actual en números negativos: Principalmente en el corte de caja, tenemos un campo de texto para añadir la caja actual del empleado, esto es la cantidad que hay en caja al finalizar el turno, sin embargo, en el establecimiento Santana Elia, en ocasiones la caja tiene números negativos, esto es posible gracias a dos botes donde se almacenan dinero, pero estos no pertenecen a caja, un bote es dedicado para las máquinas tragamonedas, otro es para la venta de tiempo aire, sin embargo cuando los gastos son amplios, se toma dinero “prestado” de estos botes para cumplir con los pagos, dejando una “deuda”, de esta situación se menciona que la caja queda en números negativos. Su solución fue muy simple, anteriormente el campo de texto tenía un cantidad mínima aceptable, la cual era 1, se eliminó esto (véase Figura 4.1.7).

```

// Verificar si el usuario ya existe
$sql_check = "SELECT COUNT(*) FROM (
    SELECT admin_usuario AS usuario FROM admin WHERE admin_usuario = :usuario
    UNION ALL
    SELECT emp_usuario FROM empleados WHERE emp_usuario = :usuario
) AS total";

$stmt_check = $pdo->prepare($sql_check);
$stmt_check->execute(['usuario' => $usuario]);
$usuario_existente = $stmt_check->fetchColumn();

if ($usuario_existente > 0) {
    header("Location: admin_employee_register.php?error=usuario_existente");
    exit();
}

```

Figura 4.1.1 Consulta que verifica si el usuario existe en las tablas admin y empleados.

```

login_submit.php ×
CorteCaja > login_submit.php > ...
16
17 try {
18     // 1. Buscar en la tabla de administradores
19     $stmt = $pdo->prepare("SELECT admin_id, admin_nombre, admin_establecimiento, admin_celular, admin_usuario, admin_contrasena FROM admin WHERE admin_usuario = :usuario");
20     $stmt->execute(['usuario' => $usuario]);
21     $admin = $stmt->fetch(PDO::FETCH_ASSOC);
22
23     if ($admin && password_verify($contrasena, $admin['admin_contrasena'])) {
24         $_SESSION['admin_id'] = $admin['admin_id'];
25         $_SESSION['admin_nombre'] = $admin['admin_nombre'];
26         $_SESSION['admin_establecimiento'] = $admin['admin_establecimiento'];
27         $_SESSION['admin_celular'] = $admin['admin_celular'];
28         $_SESSION['admin_usuario'] = $admin['admin_usuario'];
29         header("Location: admin_index.php");
30         exit();
31     }
32
33     // 2. Buscar en la tabla de empleados
34     $stmt = $pdo->prepare("SELECT emp_id, emp_nombre, emp_turno, emp_celular, emp_usuario, emp_contrasena, emp_rol, emp_estado, admin_admin_id FROM empleados WHERE emp_usuario = :usuario");
35     $stmt->execute(['usuario' => $usuario]);
36     $empleado = $stmt->fetch(PDO::FETCH_ASSOC);
37
38     if ($empleado) {
39         if ($empleado['emp_estado'] != 'Activo') {
40             $_SESSION['error_message'] = "Usuario inactivo o eliminado.";
41             header("Location: login.php");
42             exit();
43         }
44
45         if (password_verify($contrasena, $empleado['emp_contrasena'])) {
46             $_SESSION['emp_id'] = $empleado['emp_id'];
47             $_SESSION['emp_nombre'] = $empleado['emp_nombre'];
48             $_SESSION['emp_turno'] = $empleado['emp_turno'];
49             $_SESSION['emp_celular'] = $empleado['emp_celular'];
50             $_SESSION['emp_usuario'] = $empleado['emp_usuario'];
51             $_SESSION['emp_rol'] = $empleado['emp_rol'];
52             $_SESSION['emp_estado'] = $empleado['emp_estado'];
53             $_SESSION['admin_admin_id'] = $empleado['admin_admin_id'];
54             header("Location: employee_index.php");
55             exit();
56         }
57     }

```

Figura 4.1.2 Consulta del login para dar acceso al sistema.

cat_id	cat_nombre	admin_id
1	BIMBO	NULL
2	HIELO	NULL
3	VERDURA	NULL
4	FRIJOLES	NULL
5	MARINELA	NULL
6	COCA COLA	NULL
7	LALA	NULL
8	TIRAS	NULL
9	TORTILLA	NULL

Figura 4.1.3 Tabla categorias_gasto.

```
// Verificar si el gasto ya existe
$sql = "SELECT COUNT(*) FROM categorias_gasto WHERE cat_nombre = :nuevo_gasto AND (admin_id = :admin_id OR admin_id IS NULL)";
$stmt = $pdo->prepare($sql);
$stmt->bindValue(':nuevo_gasto', $gas_nuevo);
$stmt->bindValue(':admin_id', $adminID);
$stmt->execute();
$existeGasto = $stmt->fetchColumn();

if ($existeGasto == 0) {
    // Insertar el nuevo gasto en la tabla categorias_gasto
    $sql = "INSERT INTO categorias_gasto (cat_nombre, admin_id) VALUES (:nuevo_gasto, :admin_id)";
    $stmt = $pdo->prepare($sql);
    $stmt->bindValue(':nuevo_gasto', $gas_nuevo);
    $stmt->bindValue(':admin_id', $adminID);
    $stmt->execute();
}
// Asignar el nuevo gasto como la descripción del gasto a registrar
$gas_descripcion = $gas_nuevo; // Aquí asignamos el nuevo gasto a la variable de descripción
```

Figura 4.1.4 Consulta para el registro de nuevos gastos.

```
// Generar la fecha y hora actual del sistema
date_default_timezone_set('America/Mexico_City');
$gas_fecha = date("Y-m-d H:i:s");
```

Figura 4.1.5 Obtiene la hora y fecha del dispositivo donde se realiza el registro .

```
// Validar que los campos no estén vacíos
if (empty($cortecaja) || empty($corteventa) || empty($cortecajaanterior) || empty($emp_id)) {
    // Redirigir al formulario con un mensaje de error
    header("Location: employee_generate_expenses.php?error=empty_fields");
    exit();
}
```

Figura 4.1.6 Validación de repetición de datos.

```
<!-- Campo Caja -->
<div class="form-group mb-3">
  <div class="input-group">
    <span class="input-group-text"><i class="fas fa-cash-register"></i></span>
    <div class="form-floating flex-grow-1">
      <input type="number" class="form-control" id="cortecaja" name="cortecaja" placeholder="Caja actual">
      <label for="cortecaja">Caja actual</label>
    </div>
  </div>
</div>
```

Figura 4.1.7 Campo de texto para digitar la caja actual.

4.2 Conclusiones

Las conclusiones de este proyecto son amplias, debido a esto, podemos categorizarlas en varios aspectos, los cuales se explicarán a continuación:

- Eliminación a dependencia de dispositivos móviles personales: La implementación de la aplicación web ha logrado remover la necesidad de utilizar los dispositivos móviles personales para el envío del corte de caja al gerente, ahora, este corte es consultado por el gerente desde la aplicación web en su dispositivo móvil personal, mejorando la seguridad y centralizando la información.
- Sustitución del uso obsoleto de libretas: Antes esto era algo común de uso para el registro del corte de caja, sin embargo, la aplicación web también ha logrado disminuir el uso de la misma, siendo cuestión de adaptación por parte del gerente para remover por completo el uso de libretas, actualmente se usa para agregar los siguientes parámetros:
 - Caja actual: Cantidad de dinero en caja al finalizar el turno del empleado.
 - Venta: Cantidad vendida en el turno del empleado.
 - Bote de monedas: La cantidad total en bote de monedas para las máquinas tragamonedas.

- Bote de tiempo aire: La cantidad total en bote de tiempo aire.
- Tiempo aire restante: La cantidad total en el sistema de venta de tiempo aire.

Con lo anterior, han disminuido notablemente los campos originales.

- Disminución de errores humanos en cálculos: Anteriormente los cálculos se realizaban de uno en uno, es decir, si existían 20 gastos, estos se sumaban con una calculadora de uno en uno hasta sumar los 20, sin embargo, al ser muchos números que sumar, podría existir un número erróneo dentro de la suma, eso sin contar las ventas realizadas con la terminal, y la operación matemática para obtener la venta. Ahora con el uso de la aplicación web, el empleado debe registrar cada gasto y pago con terminal, al finalizar el turno, y seleccionar la herramienta “Corte”, la aplicación web suma todos los gastos y pago con terminal, además, el empleado solo debe llenar el campo de “Caja Actual”, al ingresar algún número, el campo “Venta” se actualizará según el número ingresado en “Caja Actual”, eliminando la posibilidad de errores humanos en las operaciones matemáticas.
- De manera general puedo concluir que el sistema o aplicación web ha logrado resolver las problemáticas principales que motivaron a su desarrollo, además las herramientas incluidas lograran abarcar una mayor amplitud de problemas futuros.
- La creación de este software supero mis expectativas, amplio mis conocimientos en los lenguajes y tecnologías como:
 - PHP.
 - JavaScript.
 - CSS.

- BootStrap.
- HTML.
- MySQL.

Lo cual hicieron posible la aplicación web, se realizó multitud de investigaciones de dichos lenguajes, también al principio del proyecto el entusiasmo por esta aplicación web era grande pero al notar el potencial de la misma, me siento orgulloso del futuro que prepara a la complejidad que tendrá este sistema no solo para las micro empresas de abarrotes si no una posibilidad de ampliar este sistema a nuevos micro negocios, teniendo una adaptabilidad y crecimiento futuro.

4.3 Trabajo a futuro

El trabajo a futuro que se puede implementar a este proyecto son muchas cosas, las cuales se explicaran a continuación:

- Implementación de animaciones: Aunque esto ya empezó a implementarse, en la mayoría del proyecto al guardar, eliminar o modificar algún registro se muestra únicamente un mensaje, lo ideal es mostrar una ventana emergente para una mejor visualización, además, la implementación de una gran animación al registrar un corte.
- Implementación de herramientas: Actualmente se tienen 3 herramientas importantes para su desarrollo, las cuales son:
 - Precios de frutas/verduras: Esta herramienta es para la consulta de precios de las frutas o verduras que se vendan en el establecimiento, debido a que son precios que cambian con frecuencia, el empleado con rol encargado o el gerente podrán agregar, editar y eliminar frutas y verduras para su posterior consulta. Actualmente está en desarrollo,

lo cual muy pronto pasará a una entrega dentro de la aplicación web.

- Productos a consumo del personal: Actualmente en Santana Elia, cuando algún empleado toma producto para consumo personal, este debe registrar el producto que tomó en una libreta, como se argumentó al principio del proyecto, estas sufren daños y pérdidas de información, además, este registro no lleva un orden, se registran usualmente en la última hoja de la libreta y cuando el empleado paga los productos que consumió simplemente se raya los productos escritos y escribe “Pagado a...”, debido a la implementación de la tecnología, se busca primordialmente dejar de usar dichas libretas, siendo una herramienta excelente para la aplicación web, donde tendríamos dos vistas:
 - Empleados con rol cajero: Podrán tener un formulario para agregar el nombre del producto que consumirá, el precio a venta al público, estos datos se registran en una tabla que muestre los productos, costo, fecha y hora del registro, al final de esta tabla tener un subtotal de todos los productos tomados, además, aquellos que ya se hayan pagado, también se mostrarán dicho conjunto de productos que se pagaron, fecha y hora del pago, el monto del pago y a quien se realizó el pago(solo los gerente y empleados con rol de encargado podrán tomar acción de pago).
 - Empleados con rol de encargado y gerentes: Podrán visualizar una tabla de todos los empleados que han tomado algún producto de consumo personal en estado aún no pagado, observarán las mismas

- columnas, un subtotal a precio de tienda, un subtotal con descuento de empleado y añadiendo la opción de “Pagar”, esto es al momento que el empleado pague los productos de consumo personal aún pendientes por pagar, sea cambiado a estado “Pagado a…”, esta información se añade a la tabla de historial sobre productos de consumo personal pagados.
- Información de turno al gerente: Tener en el menú principal del gerente información general sobre los gastos realizados, el total de estos, el total de pagos con terminal y venta al momento, con la finalidad que el gerente tenga mayor conocimiento de lo que está sucediendo en el establecimiento.
 - Notificaciones de gastos y corte realizado: Similar a la herramienta “Información de turno al gerente” leída con anterioridad, se busca que se muestren notificaciones en el dispositivo personal del gerente sobre los gastos y cortes que se van registrando en la base de datos, esto con la finalidad que el gerente tenga conocimiento de lo que está sucediendo el establecimiento, estas notificaciones solo mostrarán lo siguiente:
 - Gastos: Al momento de registrar un gasto el empleado en turno, el gerente recibiría una notificación donde se incluye la descripción del gasto, el monto del gasto, la hora de su registro y el empleado que realizó el registró, un ejemplo sería el siguiente: “Se ha agregado el gasto Coca Cola de un total de \$1250.00 a las 5:38 pm por César Adrian Gabriel Cruz”.
 - Cortes: Similar al anterior, pero sin mostrar información, únicamente informar al gerente que se ha realizado un corte

de caja y el empleado que lo realizó, por ejemplo: “Ya puedes consultar el corte de César Adrian Gabriel Cruz”.

- PWA: Una aplicación web progresiva son páginas o aplicaciones web, pero con el uso de Service Workers y otras tecnologías se comportan más como aplicaciones normales que como aplicaciones web, pero, sin la necesidad de instalar dicha aplicación. Ejemplo de esto son X y Facebook.
- Mejora de seguridad en contraseñas: Actualmente las contraseñas no tienen parámetros mínimos, por lo que cualquier texto puede ser una contraseña, éste no es la mejor práctica, por ello, se espera que mínimo tenga como requisito una letra mayúscula, un número y longitud de 7 caracteres.
- Multinegocios: Siendo esta una de las mejores herramientas a futuro, la intención es que un mismo gerente pueda tener la información de sus establecimientos con un solo perfil, actualmente el gerente de Santana Elia, tiene varios establecimientos de abarrotes más, sin embargo, el sistema solo te permite consultar un establecimiento por gerente, lo cual al tener varios establecimientos de estos mismos, no es práctico, al tener que loguearse y desloguearse para consultar otro establecimiento.
- Optimización de código: Al ser la primera aplicación extensa o amplia que he creado, es claro que existen malas prácticas al escribir código, dentro del mismo proyecto puede notar que se puede reducir la cantidad de líneas de código escrita, además, de ser más fluido al estar mejor estructurada así como mejorar la seguridad del mismo proyecto.
- Corrección de posibles errores: Debido a la implementación de nuevas tecnologías y herramientas, se tendrán que realizar pruebas que permitan verificar el buen funcionamiento del mismo.

- Diseño de interfaz: Este trabajo a futuro se basa únicamente en la interfaz gráfica, buscar formas en que la aplicación tenga una presentación visualmente más atractiva y llamativa para los usuarios.
- Expansión a micronegocios: Este software podría buscar ampliar a más micronegocios que tengan problemáticas similares a la de Santana Elia, llegando a crear un negocio con cobro mensual, bimestral o semestral, e incluso con la ampliación de negocios, añadir mejores herramientas que enriquezcan la aplicación web.
- Recuperación de cuentas: Cuando un empleado olvidó su contraseña, el gerente puede generar una nueva contraseña para dicho usuario, sin embargo, el gerente en caso de olvidar su contraseña, no tiene como recuperarla, por ello, se buscaría implementar una herramienta que te ayude a facilitar el acceso a la cuenta en caso que la contraseña sea olvidada.
- FAQ: Tener un apartado de preguntas y respuestas rápidas o más comunes, además, de videos intuitivos, cortos y fáciles de entender para los usuarios, aunque, la aplicación web es simple de usar, en ocasiones se puede dificultar acciones específicas, además, este ayuda audiovisual puede mejorar el uso de la aplicación web.

BIBLIOGRAFÍA

- [1] IEBS Business School, "¿Qué son las metodologías ágiles? Agile y Scrum," *IEBSchool*, 20-sep-2023. [Online]. Available: <https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/>. [Accessed: 17-Feb-2025].
- [2] APD España, "Cómo aplicar la metodología Scrum y qué es el método Scrum," APD España, 09-Apr-2024. [Online]. Available: <https://www.apd.es/metodologia-scrum-que-es/>. [Accessed: 17-Feb-2025].
- [3] "¿Qué es una base de datos?," Oracle.com, 24-Nov-2020. [Online]. Available: <https://www.oracle.com/mx/database/what-is-database/> [Accessed: 17-Feb-2025].
- [4] IT User, "5 lenguajes de programación que los administradores de bases de datos deben aprender," Discover the New, 19-Jan-2021. [Online]. Available: <https://discoverthenew.ituser.es/devops/2021/01/5-lenguajes-de-programacion-que-los-administradores-de-bases-de-datos-deben-aprender>. [Accessed: 17-Feb-2025].
- [5] Concepto.de, "Lenguaje de programación," Concepto.de, [Online]. Available: <https://concepto.de/lenguaje-de-programacion/>. [Accessed: 17-Feb-2025].
- [6] Amazon Web Services, "¿Qué es JavaScript?," AWS, [Online]. Available: <https://aws.amazon.com/es/what-is/javascript/>. [Accessed: 17-Feb-2025].
- [7] Rock Content, "PHP: qué es, para qué sirve y diferencias con otros lenguajes," Rock Content, [Online]. Available: <https://rockcontent.com/es/blog/php/>. [Accessed: 17-Feb-2025].
- [8] Indeed, "Herramientas de desarrollo de software," Indeed, [Online]. Available:

<https://mx.indeed.com/orientacion-profesional/desarrollo-profesional/herramientas-desarrollo-software>. [Accessed: 17-Feb-2025].

[9] OpenWebinars, "¿Qué es Visual Studio Code y qué ventajas ofrece?", OpenWebinars, [Online]. Available: <https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/>. [Accessed: 17-Feb-2025].

[10] Rock Content, "Bootstrap: guía para principiantes de qué es, por qué y cómo usarlo," Rock Content, 12-Apr-2020. [Online]. Available: <https://rockcontent.com/es/blog/bootstrap/>. [Accessed: 17-Feb-2025].

[11] Arsys, "phpMyAdmin," Arsys, [Online]. Available: <https://www.arsys.es/blog/phpmyadmin>. [Accessed: 17-Feb-2025].

[12] Innovación Digital 360, "Guía completa de MySQL Workbench: descarga, instalación y uso," Innovación Digital 360, [Online]. Available: <https://www.innovaciondigital360.com/big-data/guia-completa-de-mysql-workbench-descarga-instalacion-y-uso/>. [Accessed: 17-Feb-2025].

[13] Norvic Software, "¿Qué es XAMPP?," Norvic Software, [Online]. Available: <https://norvicsoftware.com/que-es-xampp/>. [Accessed: 17-Feb-2025].

APÉNDICE A. GLOSARIO TÉCNICO

- Corte de caja de Santana Elia: Este es un corte simple, el cual tenemos campos específicos para este establecimiento, los cuales se componen de:
 - Caja actual.
 - Caja anterior.
 - Venta del turno.
 - Gasto total del turno.
 - Descripción de los gastos del turno.
 - Total en ventas por terminal.
 - Dinero total en bote para las máquinas tragamonedas.
 - Dinero total en bote de tiempo aire.
 - Total restante de tiempo aire disponible en sistema.

La fórmula para obtener la venta en Santana Elia es: caja actual + pagos con tarjeta + total de gasto - caja del turno anterior.

- XAMPP: Es una aplicación que hace funcionar una PC o Laptop como un mini servidor, esto es útil para el funcionamiento de una página o aplicación web, usualmente es usado para efectos de prueba de alguna aplicación web, lo cual xampp funcionó en la elaboración de este proyecto.
- CardView - Cards: Son tarjetas que se usan en aplicaciones o páginas web para mostrar información de manera ordenada y visualmente atractiva.
- Aplicación web: Es como una página web, sin embargo, a diferencia de, en una aplicación web es interactiva, es decir, escribir, enviar, subir archivos, presionar botones entre otras acciones, mientras que una página web solo es para lectura de información, además, en una

aplicación web no requieres instalar nada, únicamente accedes con un navegador.

- Interfaz gráfica: Es todo lo que se visualiza en la pantalla al usar una aplicación o página web, es la parte visual con la que se interactúa para usar el sistema sin tener que escribir comandos.
- Header: También conocido como encabezado, es la parte de arriba de una página web o aplicación, donde se encuentra el logo, menú de navegación, botones como “Inicio”, “Contacto” o “Cerrar Sesión”.
- Container - Contenedor: Es una caja invisible que se usa para agrupar y ordenar los elementos dentro de una página o aplicación web, estructura los componentes en un bloque.
- Placeholder: Es un texto por lo general en color gris claro dentro de un campo de texto, como un formulario, sirve como ejemplo o guía para saber qué es lo que se debe ingresar, por ejemplo “Escribe tu correo”.
- Readonly: Es cuando en un campo de texto, este solo se puede leer, no se puede modificar, esto es útil cuando se requiere mostrar información sin que el usuario pueda editar para evitar algún error.
- Backend: Es parte de la aplicación o página web que no se ve, pero hace que funcione, por ejemplo, guardar datos, procesar información y comunicarse con bases de datos, es como el motor de un auto, no lo ves mientras se maneja, pero sin él, el auto no funciona.
- Frontend: La interfaz gráfica está dentro del frontend, además, también influye el comportamiento e interacción con las distintas páginas.
- Método POST: Es una de las formas en las que se envía datos de un navegador a un servidor, pero de una manera segura y oculta, este POST se usa al presionar botón “Enviar”.
- Archivos submit: Dentro del proyecto, los archivos que terminan con el nombre de “submit”, envían datos de registro al servidor, por ejemplo, registrar un empleado, un gasto o un corte.

- Clave primaria: Es un valor único que se le asigna a cada registro o fila de una tabla en una base de datos para su posterior identificación, por ejemplo, en una universidad con varios salones, estos deberían tener un identificador único para la ubicación del mismo.
- Clave foránea: Es una clave primaria en otra tabla, esto se usa para conectar dos tablas diferentes en una base de datos.
- ID: En este proyecto el ID es la clave primaria.
- Required: Es una indicación que se usa en formularios o campos de texto para ser de este obligatorio llenar dicho campo, normalmente representado por un asterisco en color rojo al final de este campo.
- Script: Del lenguaje de JavaScript es un conjunto de instrucciones que dice que realizar a la página o aplicación web cuando se interactúa con esta, normalmente se refleja en acciones interactivas.
- Variables de sesión: Es información almacenada mientras dura la sesión activa en una aplicación o página web, en estas se almacena normalmente el nombre de usuario.
- Controlador: En este proyecto el controlador funge como el intermediario entre la visualización de datos en la interfaz gráfica y la base de datos, el controlador obtiene la información requerida para posteriormente pueda ser mostrada.
- Function - Función: Es una instrucción de código que realiza algo en específico, es un bloque que puede ser reutilizado varias ocasiones y puede ser integrada en otros espacios del código solo con “llamarla”, evitando código repetitivo.
- CURDATE(): Es una función que se usa en bases de datos para obtener la fecha actual del sistema.
- SUM(): Suma todos los valores de una columna numérica y obtenga un total de esta.

- ID autoincrementable: Es un tipo de campo en una base de datos, comúnmente el id es la clave primaria de una tabla, incrementable es que con cada registro se aumenta en uno al último registro, es decir 1, 2, 3, 4 y así sucesivamente, si el último fue 4, al registrar nuevamente este será 5.
- NOW() + INTERVAL 2 HOURS: Se usa comúnmente en SQL para obtener una fecha y hora actual sumándole un intervalo de 2 horas.
- ByetHost: Es un servicio de alojamiento web que ofrece planes gratuitos y de pago, en este proyecto, se usó un alojamiento gratuito con limitaciones de espacio en disco y ancho de banda, sin embargo, nos ofrece un subdominio o url para acceder a la aplicación web.