



UNIVERSIDAD
PRIVADA DEL NORTE

“SISTEMA WEB PARA LA OPTIMIZACIÓN EN LA ATENCIÓN AL CLIENTE Y CONTROL DE INVENTARIO DE LA PANADERÍA CREMA Y MILHOJAS LIMA, 2025”

INTEGRANTES:

- Arroyo Carrasco, Álvaro Iván (100%)
- Santiago Caraza, Luis Miguel (100%)
- Vargas Mallqui, Frank Maiccol (100%)
- Angulo Canchero, Merly Andrea (100%)
- Rivera Moreno, Gustavo Sebastián (100%)

DOCENTE:

- Mg. Calderon Aquiño, Jeanette Cinthia

CARRERA PROFESIONAL:

- Ingeniería de Sistemas Computacionales

CLASE: 10394

LIMA – 2025

INTRODUCCIÓN

En la actualidad, la digitalización de los procesos comerciales ha transformado la manera en que las empresas gestionan sus operaciones, permitiendo una mayor eficiencia y optimización de recursos. En el sector de las panaderías, la gestión tradicional de atención al cliente y control de inventario sigue siendo un desafío, ya que muchas de estas empresas operan de manera presencial, lo que genera demoras en la atención y errores en el manejo de productos. El 40% de los consumidores prefiere explorar y comprar productos alimenticios en línea, las panaderías que no adaptan su modelo de ventas pierden relevancia frente a la competencia.

CREMA & MILHOJAS S.A.C, una empresa dedicada a la elaboración y venta de productos de panadería y pastelería, busca dar un paso adelante en su transformación digital. Reconocida por ofrecer productos frescos, de alta calidad y un servicio confiable, la empresa se ha consolidado en su rubro como una opción preferida por los clientes locales que valoran el sabor tradicional y la atención personalizada. Este negocio opera en el sector alimentario, específicamente en el rubro de panadería y pastelería, brindando una variada oferta de productos que incluyen panes, pasteles y postres elaborados con ingredientes cuidadosamente seleccionados. Su propuesta de valor se basa en la frescura de sus productos, la innovación en sus recetas y el compromiso constante con la satisfacción del cliente.

La atención al cliente es un factor clave en la fidelización y satisfacción de los consumidores. Según Ocampo y Guevara (2020), la implementación de sistemas web en negocios locales ha demostrado mejorar la interacción con los clientes, reduciendo tiempos de esperar y facilitando la gestión de pedidos en línea. En el caso de la panadería Crema y Milhojas, la falta de un sistema automatizado ha generado dificultades en la comunicación con los clientes, afectando la eficiencia del servicio.

De igual modo, el control de inventario es fundamental para garantizar la disponibilidad de productos y evitar pérdidas económicas. Fajardo y Lorenzo (2017) destacan que la digitalización del inventario permite minimizar errores en el registro de productos, optimizar la planificación de abastecimiento y mejorar la rentabilidad del negocio. La panadería Crema y Milhojas enfrenta problemas en la gestión manual de su inventario, lo que provoca sobreventa de productos y falta de stock en momentos de alta demanda.

Ante este problema, la implementación de un sistema web representa una solución eficaz para mejorar la atención al cliente y optimizar el control de inventario. Ya que de acuerdo con Palpa Castro (2015) señala que los sistemas web permite automatizar procesos clave en negocios de venta de productos, facilitando la administración de pedidos y el monitoreo de inventario en tiempo real. Bajo este contexto, el presente proyecto se plantea desarrollar un sistema web para la panadería Crema y MilHojas, con el fin de mejorar la experiencia del cliente y garantizar una gestión eficiente de los productos disponibles. Además, esta iniciativa representa una oportunidad para aplicar conocimientos técnicos en un entorno real, aportando valor a una empresa local con gran potencial de expansión.

PRESENTACIÓN

En el presente proyecto se desarrollará un análisis completo sobre la situación actual de la empresa “Crema y mil hojas” abordando sus principales problemas, con el objetivo de proponer estrategias de mejora que impulsen su crecimiento y competitividad. A lo largo del trabajo se expondrán las funciones y objetivos del proyecto, el contexto empresarial, la problemática detectada, el análisis de datos y, finalmente, las conclusiones y recomendaciones.

Comenzando con el capítulo 1, en este se realiza una descripción general de la empresa “Crema y mil hojas”, detallando su historia, misión, visión, estructura organizativa y los servicios que ofrece. Además, se contextualiza su posición actual en el mercado y su entorno competitivo.

Capítulo 2, este capítulo se centra en identificar la problemática principal que afecta a la empresa. A partir del análisis situacional, se determinan las causas de dicha problemática. También se establecen los objetivos generales y específicos que guiarán el desarrollo del proyecto, así como la justificación de por qué es relevante abordar esta temática.

Siguiendo con el Capítulo 3, aquí se recopila información teórica relacionada con los conceptos clave del proyecto. El marco teórico sirve para sustentar el proyecto con conceptos, teorías y antecedentes que ayudan a entender mejor el problema y fundamentar las posibles soluciones.

Entrando en el capítulo 4, en este se describe el proceso de desarrollo del producto web utilizando la metodología ágil Scrum, con el objetivo de gestionar eficientemente el trabajo en equipo y adaptarse a los cambios de forma iterativa e incremental.

Finalmente, en el capítulo 5 se presentan las conclusiones obtenidas tras el desarrollo del proyecto, destacando los aprendizajes clave y el impacto esperado de la propuesta. Se ofrecen recomendaciones para asegurar la sostenibilidad de las mejoras planteadas.

ÍNDICE

I.	Generalidades	11
1.1.	Título del trabajo de campo	11
1.2.	Descripción de empresa	11
1.3.	Misión y visión de la empresa.....	11
1.3.1.	Misión	11
1.3.2.	Visión.....	12
1.4.	Organigrama.....	12
II.	Problemática.....	13
2.1.	Planteamiento del problema y descripción.....	13
2.1.1.	Problema General.....	13
2.1.2.	Problemas Específicos.....	14
2.2.	Objetivos	15
2.2.1.	Objetivo general.....	15
2.2.2.	Objetivos específicos	15
III.	MARCO TEÓRICO DE REFERENCIA.....	17
3.1.	Proceso de atención al cliente	17
3.2.	Proceso de control de inventario.....	17
3.3.	Ingeniería de Software.....	17
3.4.	Sistema web	18
3.5.	Dominio	18
3.6.	Servidor.....	18
3.7.	Visual Studio.....	19
3.8.	MySQL	19
3.9.	Tailwind CSS	19
3.10.	Javascript	20
3.11.	PHP	20
3.12.	XAMPP.....	20
3.13.	Modelo Vista Controlador (MVC)	21
3.14.	Metodología Scrum.....	21
3.15.	Artefactos de Scrum	21
3.15.1.	Product Backlog.....	22
3.15.2.	Sprint Backlog.....	22

3.15.3. Incremento (Increment).....	22
3.16. Eventos de Scrum	23
3.17. PhpMyAdmin	23
3.18. Pruebas de Software	23
3.19. Calidad de Software	24
IV. Desarrollo ágil del producto	25
4.1. Historias de usuario.....	25
4.2. Scrum team (Equipo Scrum).....	39
4.3. Matriz de Impacto	39
4.4. Matriz de Puntos Estimados	40
4.5. Product backlog	40
4.6. Product backlog priorizado	42
4.7. Entregables por sprint.....	43
4.8. Plan de trabajo	44
4.9. Diagrama de clases de diseño.....	45
4.10. Diagrama físico de la base de datos.....	46
4.11. Diagrama de paquetes de la arquitectura de software	47
4.12. Sprint 1.....	48
4.12.1. Sprint Backlog.....	48
4.12.2. Sprint Planning	48
4.12.3. Desarrollo	49
4.12.4. Burndown chart sprint 1	49
4.12.5. Review sprint 1	50
4.12.6. Matriz de casos de prueba	50
4.13. Sprint 2.....	52
4.13.1. Sprint Backlog.....	52
4.13.2. Sprint Planning	52
4.13.3. Desarrollo	53
4.13.4. Burndown chart sprint 2	53
4.13.5. Review sprint 2	54
4.13.6. Matriz de casos de prueba	54
4.14. Sprint 3.....	56
4.14.1. Sprint Backlog.....	56
4.14.2. Sprint Planning	56

4.14.3.	Desarrollo	57
4.14.4.	Burndown chart sprint 3	57
4.14.5.	Review sprint 3	58
4.14.6.	Matriz de casos de prueba	58
4.15.	Sprint 4	60
4.15.1.	Sprint Backlog.....	60
4.15.2.	Sprint Planning	60
4.15.3.	Desarrollo	61
4.15.4.	Burndown chart sprint 4	61
4.15.5.	Review sprint 4	62
4.15.6.	Matriz de casos de prueba	62
4.16.	Sprint 5.....	64
4.16.1.	Sprint Backlog.....	64
4.16.2.	Sprint Planning	64
4.16.3.	Desarrollo	65
4.16.4.	Burndown chart sprint 5	65
4.16.5.	Review sprint 5	66
4.16.6.	Matriz de casos de prueba	66
V.	Conclusiones y recomendaciones.....	68
5.1.	Conclusiones	68
5.2.	Recomendaciones.....	69
5.3.	Glosario de términos.....	70
VI.	Bibliografía.....	72
VII.	Anexos.....	75

ÍNDICE DE FIGURAS

Figura 1. Organigrama de crema & milhojas	12
Figura 2. Historia de Usuario N°1.....	25
Figura 3. Historia de Usuario N°2.....	26
Figura 4. Historia de Usuario N°3.....	27
Figura 5. Historia de Usuario N°4.....	28
Figura 6. Historia de Usuario N°5.....	29
Figura 7. Historia de Usuario N°6.....	30
Figura 8. Historia de Usuario N°7.....	31
Figura 9. Histora de Usuario N°8	32
Figura 10. Historia de Usuario N°9.....	33
Figura 11. Historia de Usuario N°10	34
Figura 12. Historia de Usuario N°11	35
Figura 13. Historia de Usuario N°12.....	36
Figura 14. Historia de Usuario N°13.....	37
Figura 15. Historia de Usuario N°14.....	38
Figura 16. Plan de trabajo	44
Figura 17. Diagrama de clases de diseño	45
Figura 18. Diagrama físico de la base de datos	46
Figura 19. Diagrama de paquetes de la arquitectura de software	47
Figura 20. Burndown chart sprint N°1	49
Figura 21. Burndown chart sprint N°2	53
Figura 22. Burndown chart sprint N°3	57
Figura 23. Burndown chart sprint N°4	61
Figura 24. Burndown chart sprint N°5	65

ÍNDICE DE TABLAS

Tabla 1. Matriz equipo scrum	39
Tabla 2. Matriz de prioridad.....	39
Tabla 3. Matriz de puntos estimados.....	40
Tabla 4. Product backlog.....	41
Tabla 5. Product backlog priorizado	42
Tabla 6. Entregables por sprint.....	43
Tabla 7. Matriz del sprint N°1	48
Tabla 8. Matriz de review del sprint N°1	50
Tabla 9. Matriz de casos de prueba del sprint N°1	50
Tabla 10. Matriz del sprint N°2	52
Tabla 11. Matriz de review del sprint N°2.....	54
Tabla 12. Matriz de casos de prueba del sprint N°2	54
Tabla 13. Matriz del sprint N°3	56
Tabla 14. Matriz de review del sprint N°3	58
Tabla 15. Matriz de casos de prueba del sprint N°3	58
Tabla 16. Matriz del sprint N°4	60
Tabla 17. Matriz de review del sprint N°4	62
Tabla 18. Matriz de casos de prueba del sprint N°4	62
Tabla 19. Matriz del sprint N°5	64
Tabla 20. Matriz de review del sprint N°5	66
Tabla 21. Matriz de casos de prueba del sprint N°5	66

ÍNDICE DE ANEXOS

Anexo 1. Formar el equipo de desarrollo y presentar el Formato de Proyecto de Software con las siguientes partes: título, integrantes y características técnicas del producto software a desarrollar.	75
Anexo 2. Implementación del software para el caso de uso Gestión de pedidos a un 50%	77
Anexo 3. Implementación del software para el caso de uso Gestión de pedidos a un 100%	81
Anexo 4. Implementación de las pruebas unitarias del software para el caso de uso principal	88
Anexo 5. Depuración del software para pasar con éxito las pruebas unitarias	101
Anexo 6. Diseño de casos de prueba para evaluar la funcionalidad del Software	104
Anexo 7. Ejecución de los casos de pruebas y depurar el software para pasar con éxito las pruebas funcionales	109
Anexo 8. Inspección del software	115
Anexo 9. Refactorización del software	135
Anexo 10. Acta de aprobación	146
Anexo 11. Actas de reuniones y planificaciones	147
Anexo 12. Video explicativo del software	161

CAPITULO I

I. Generalidades

1.1. Título del trabajo de campo

Sistema web para la optimización del proceso de ventas de la panadería

Crema y Milhojas lima, 2025

1.2. Descripción de empresa

CREMA & MILHOJAS S.A.C. es una empresa peruana con RUC 20608170805, establecida el 5 de julio de 2021. Desde sus inicios, se ha enfocado en el rubro gastronómico, desarrollando actividades relacionadas con la elaboración de productos de panadería, así como la operación de restaurantes y servicios de comida móvil. La empresa busca destacarse por su propuesta innovadora y su compromiso con la calidad, ofreciendo experiencias culinarias únicas a sus clientes.

1.3. Misión y visión de la empresa

1.3.1. Misión

Ofrecer al consumidor productos de panadería y pastelería artesanal de alta calidad y frescura, a precios accesibles, innovando continuamente y operando con eficiencia para generar valor a nuestros clientes, colaboradores, proveedores y a la comunidad.

1.3.2. Visión

Ser reconocidos a nivel nacional como una marca referente en panadería y pastelería artesanal, destacando por la excelencia de nuestros productos, la cercanía con el cliente y el compromiso con una operación responsable y sostenible.

1.4. Organigrama

Figura 1.

Organigrama de crema & milhojas



Nota: Elaboración propia en Canva que presenta el Organigrama de la panadería Crema & MilHojas

CAPITULO II

II. Problemática

2.1. Planteamiento del problema y descripción

2.1.1. Problema General

Las panaderías tradicionales suelen gestionar la atención al cliente de manera presencial, lo que puede generar demoras, dificultades en la comunicación y una experiencia poco eficiente, especialmente en horarios de alta demanda. La panadería Crema y Milhojas enfrenta estos mismos desafíos, ya que sus clientes experimentan dificultades al realizar pedidos de manera rápida y organizada, lo que afecta su satisfacción y fidelización.

Por otro lado, la gestión manual del inventario que posee el negocio provoca errores en el control de productos disponibles, lo que se traduce en problemas como la falta de stock, la sobreventa de productos y una administración ineficiente de los recursos. Al no contar con un sistema automatizado de control de inventario, el negocio no puede realizar un seguimiento preciso de la disponibilidad de productos, afectando la operatividad y la toma de decisiones estratégicas.

Según Espinal et. al (2020), la implementación de un sistema web permite agilizar los procesos de atención al cliente, facilitando la gestión de pedidos y mejorando la experiencia del usuario. Un sistema digitalizado optimiza la comunicación entre el negocio y sus clientes, reduciendo tiempos de espera y mejorando la eficiencia operativa.

También, de acuerdo con Fajardo y Lorenzo (2017) destacan que un sistema web de control de inventario minimiza pérdidas, mejora la planificación y evita errores

en la disponibilidad de productos, lo que impacta de manera directa en la rentabilidad del negocio.

Además, un estudio realizado en Lima sobre la implementación de un sistema web en empresas locales demostró que la digitalización del control de inventario incrementa la precisión en la gestión de productos en un 23.48%, mejorando la toma de decisiones estratégicas y reduciendo desperdicios (Chávez, 2023)

Con todo lo mencionado, la implementación de un sistema web para la atención al cliente y control de inventario permitiría agilizar la gestión de pedidos, mejorar la comunicación con los clientes y optimizar el registro y supervisión de productos. Esto no solo garantizaría una experiencia más eficiente para los consumidores, sino que también facilitaría el control de inventario, reduciendo errores y mejorando la administración del negocio.

2.1.2. Problemas Específicos

- Deficiencia en la atención al cliente debido a la falta de un sistema automatizado para gestionar pedidos, consultas y comunicación con los clientes, lo que provoca tiempos de espera prolongados y dificultades en la experiencia de compra.
- Errores en el control de inventario, que generan inconsistencias en la disponibilidad de productos, afectando la operatividad del negocio y la satisfacción del cliente por falta de stock.
- Dificultad en la toma de decisiones estratégicas, ya que la gestión manual impide un análisis preciso de datos sobre productos más vendidos, demanda en diferentes horarios y necesidades de reposición.

- Riesgo de sobreventa y pérdidas económicas por la ausencia de un sistema que actualice la cantidad de productos disponibles en tiempo real, lo que provoca problemas al momento de la entrega de pedidos.
- Limitaciones en la eficiencia operativa, dado que la falta de digitalización impide optimizar los procesos internos del negocio, aumentando el tiempo y esfuerzo requerido para la gestión manual de inventario y atención al cliente.

2.2. Objetivos

2.2.1. Objetivo general

Diseñar e implementar un sistema web para optimizar la atención al cliente y el control de inventario en la panadería Crema y Milhojas, mejorando la eficiencia en la gestión de pedidos, la comunicación con los clientes y la supervisión de productos disponibles, reduciendo errores y optimizando la toma de decisiones estratégicas.

2.2.2. Objetivos específicos

- Desarrollar un módulo de gestión de pedidos que permita a los clientes realizar solicitudes de productos de manera rápida y organizada, reduciendo tiempos de espera y mejorando la interacción con el negocio.
- Implementar un sistema automatizado de control de inventario que registre y actualice en tiempo real la disponibilidad de productos, evitando errores de stock y optimizando la administración de recursos.
- Diseñar una interfaz intuitiva y accesible que facilite la comunicación entre la panadería y sus clientes, proporcionando información clara sobre productos, disponibilidad y opciones de pedido.

- Desarrollar un sistema de generación de reportes que permita analizar datos de inventario y preferencias de los clientes, optimizando la toma de decisiones estratégicas en la administración del negocio.
- Optimizar los procesos internos de la panadería mediante la digitalización de la gestión de productos y atención al cliente, reduciendo la carga de trabajo manual y mejorando la eficiencia operativa.

CAPITULO III

III. MARCO TEÓRICO DE REFERENCIA

3.1. Proceso de atención al cliente

El proceso de atención al cliente constituye un elemento fundamental para garantizar una experiencia satisfactoria en cualquier entorno comercial. Como señala Quille (2023) "el proceso de atención al cliente es en donde se brinda la información detallada y se da seguimiento al servicio al cliente." Lo cual permite establecer una comunicación efectiva y continua con el usuario.

3.2. Proceso de control de inventario

Según Calle y Centeno (2023), el proceso de control de inventarios consiste en gestionar eficientemente el flujo de productos dentro y fuera de la empresa, asegurando que el stock disponible se mantenga en niveles óptimos. Este proceso implica varias etapas, comenzando con la planificación de la demanda, lo que permite prever las necesidades de productos en función de la variabilidad del mercado y la estacionalidad. A lo largo de este proceso, es fundamental registrar con precisión todas las entradas y salidas de mercancías para evitar errores que puedan generar desabastecimientos o excesos de inventario.

3.3. Ingeniería de Software

Tomando en cuenta la investigación de Rozo et. al. (2020) se puede afirmar que "Es indudable que las pruebas de software (PS) es uno de los aspectos que hacen que la ingeniería de software trascienda en los proyectos informáticos con calidad de producto."

3.4. Sistema web

En el contexto del desarrollo de tecnologías digitales, es fundamental comprender la estructura y funcionalidad de los sistemas web. En este sentido, De la Rosa y León (2024) explican que un sistema web se define como aquel que tiene servicios, tanto en contenido como en funcionalidad, que se ofrecerán mediante la web. El diseño de componentes se centra en los objetos de esta y en cómo se puede mostrar la interfaz al usuario final. Además, el diseño funcional del sistema web se enfoca en la manipulación de los objetos, realización de cálculos y consulta en base de datos.

3.5. Dominio

En el entorno digital actual, comprender el funcionamiento de los nombres de dominio es esencial para garantizar el acceso eficiente a los sitios web. En este marco, Chacalizaza y Yarin (2020) señalan que los nombres de dominio en Internet es la forma de poder resolver una dirección IP, todo servidor cuenta con un servidor DNS que es un sistema de nombres de dominio (DNS), la función principal es traducir las direcciones IP en un nombre fácil de recordar y de encontrar.

3.6. Servidor

El servidor web constituye un componente fundamental en la arquitectura de los sistemas en línea, ya que permite la entrega efectiva de los contenidos solicitados por los usuarios. En este sentido, Forra (2020) sostiene que un servidor web como su nombre lo indica, es un software instalado en el equipo con todas las condiciones necesarias para servir o entregar páginas web que le sean solicitadas por un navegador, asegurando que se muestren y representen todos los elementos necesarios para su correcto funcionamiento y visualización.

3.7. Visual Studio

En el ámbito del desarrollo de software, la elección del entorno de desarrollo adecuado es crucial para optimizar los procesos de programación. En este contexto, Palomo (2022) señala que Visual Studio Code es un editor de código fuente, es un IDE (Entorno de Desarrollo Integrado) ligero pero potente que se ejecuta en su escritorio y ofrece una nueva opción de herramienta para desarrolladores que combina la simplicidad y la experiencia optimizada de un IDE con lo necesario para su ciclo principal de compilación y depuración de código.

3.8. MySQL

En el desarrollo de aplicaciones modernas, el uso de sistemas de gestión de bases de datos desempeña un papel fundamental en la organización y acceso eficiente a la información. En esta línea, González (2021) "Es un servicio de base de datos completamente administrado, que, utilizando la base de datos de código abierto, permite a la mayoría de las organizaciones poder implementar aplicaciones nativas de la nube.", destacando así la importancia de las soluciones administradas y su integración con entornos en la nube.

3.9. Tailwind CSS

En el ámbito del desarrollo front-end, la eficiencia en la construcción de interfaces visuales ha llevado a la adopción de herramientas que optimizan tanto el diseño como la mantenibilidad del código. En este contexto, Coloma (2025) Tailwind CSS es un framework de diseño que facilita la creación de interfaces modernas y consistentes mediante el uso de clases de utilidad predefinidas. A diferencia del enfoque tradicional, donde los desarrolladores crean clases personalizadas en archivos CSS, Tailwind permite aplicar estilos directamente en los elementos HTML, reduciendo la necesidad de escribir

CSS adicional. Esto mejora la eficiencia y la mantenibilidad del código, permitiendo crear diseños complejos rápidamente.

3.10. Javascript

JavaScript es uno de los lenguajes más utilizados en el desarrollo web, especialmente en el lado del cliente, debido a su versatilidad y capacidad de adaptación a diversos paradigmas de programación. Como menciona Roca (2022) "JavaScript es un lenguaje de programación dinámico, de varios caracteres, de un solo subproceso y basado en prototipos que admite programación orientada a objetos, imperativa y declarativa."

3.11. PHP

En el desarrollo de aplicaciones web dinámicas, el lenguaje PHP continúa siendo una herramienta ampliamente utilizada por su flexibilidad y facilidad de integración. Según Romero & Cantos (2021) Se entiende que PHP es un lenguaje de programación Web de "código abierto" interpretado, ejecutado por un servidor. Asimismo, PHP es un lenguaje de programación que fue diseñado específicamente para el desarrollo y producción de páginas web que se ejecuta por medio de un servidor donde es presentado por medio de un navegador.

3.12. XAMPP

En el entorno del desarrollo web local, XAMPP se presenta como una herramienta fundamental para la configuración rápida y funcional de servidores. Rodríguez (2022) destaca que "Es una distribución de Apache completamente gratuita y fácil de instalar que contiene MariaDB, PHP y Perl. El paquete de instalación de XAMPP ha sido diseñado para ser increíblemente fácil de instalar y usar."

3.13. Modelo Vista Controlador (MVC)

El Modelo Vista Controlador (MVC) es uno de los patrones de arquitectura más ampliamente adoptados en el desarrollo de aplicaciones web modernas, debido a su capacidad para separar claramente las distintas responsabilidades del sistema. Según Nuñez (2024) "Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos", lo cual facilita la organización del código, mejora la mantenibilidad y permite una evolución más flexible del sistema.

3.14. Metodología Scrum

En el contexto de la gestión de proyectos ágiles, Scrum se ha consolidado como uno de los marcos de trabajo más adoptados para el desarrollo de productos complejos. Según Morejón y Romero (2023) Scrum es un proceso que propone la aplicación de un método de trabajo más colaborativo entre las áreas involucradas en el desarrollo de nuevos productos, trabajando de manera holística para lograr una mejor comunicación, una mejor integración y conocimiento de todos los roles para realizar el trabajo.

3.15. Artefactos de Scrum

Dentro del marco metodológico de Scrum, los artefactos juegan un papel esencial en la planificación y ejecución del trabajo. Según Leon (2025) "los artefactos de Scrum son herramientas fundamentales dentro de la metodología. En este contexto, un artefacto se considera todo aquello que se crea para resolver un problema.", lo cual enfatiza su función práctica en la resolución de necesidades concretas durante el desarrollo de productos.

3.15.1. Product Backlog

Dentro del marco metodológico de Scrum, el Product Backlog constituye una herramienta esencial para la planificación y el desarrollo incremental del producto. Según Pérez (2023) el Product Backlog es un listado de tareas que tiene que ser realizadas para que el producto final sea entregado a tiempo y funcional. Dicho listado es generado por el Product Owner quien asigna las tareas a concretar en cada Sprint con el objetivo de cumplir a cabalidad cada ítem.

3.15.2. Sprint Backlog

En el proceso de desarrollo ágil, el Sprint Backlog representa un componente operativo fundamental, ya que detalla las tareas específicas que el equipo debe completar durante cada iteración. Según Santillán (2024) el Sprint Backlog se enfoca en las tareas y funcionalidades clave identificadas durante la planificación del sprint. Para esta aplicación web, el Sprint Backlog se estructura en función de las secciones claves: inicio, catálogo, proceso de compra, nosotros, contacto y área de administración. Cada elemento en el Sprint Backlog tiene una descripción clara.

3.15.3. Incremento (Increment)

En la metodología Scrum, el Incremento representa el resultado tangible del trabajo realizado durante un Sprint, siendo una parte fundamental para evaluar el avance del producto. Según Santa Cruz y Zapata (2021) Un Incremento es un paso específico que lleva al objetivo del Producto, cada uno de los Increments es adicionado a los demás Increments con una revisión detallada, lo cual nos asegura que estos estén funcionando juntos de manera correcta; para que proporcionen valor estos deben ser utilizables.

3.16. Eventos de Scrum

Dentro del marco de trabajo Scrum, los eventos, también conocidos como ceremonias, son componentes esenciales que estructuran el flujo de trabajo del equipo. Como señalan Andy y Farías (2024) "los eventos de SCRUM o conocidas como ceremonias son actividades que el equipo de trabajo desarrolla para llevar con éxito el proyecto, desde la planificación hasta la retrospectiva."

3.17. PhpMyAdmin

Php MyAdmin es una herramienta ampliamente utilizada para la gestión de bases de datos, especialmente en entornos web. Para Caicedo y Ruiz (2024) Php MyAdmin es una herramienta que nos permite gestionar bases de datos MySQL, este gestor se utiliza principalmente cuando tenemos un alojamiento web, en especial si se requiere acceder a los datos, realizar alguna modificación en las tablas de la base de datos.

3.18. Pruebas de Software

En el proceso de desarrollo de sistemas, las pruebas de software desempeñan un papel crucial, ya que garantizan el correcto funcionamiento del sistema antes de su implementación definitiva. Como indican Cañizares y Perrazo (2024) "las pruebas de software son esenciales en el proceso del desarrollo del sistema, debido a que aseguran que el sistema funcione de manera adecuada y cumpla con los requisitos que fueron establecidos."

3.19. Calidad de Software

La calidad del software es un componente esencial en el desarrollo de sistemas, ya que permite garantizar el cumplimiento de los requisitos establecidos y la eficacia del producto final. Según Chambi (2021) con base en los requisitos funcionales y no funcionales determinados en la etapa de análisis del sistema, la calidad del software está relacionada con "el cumplimiento de los requisitos funcionales y de rendimiento claramente establecidos, así como los estándares de desarrollo completamente documentados, y todo el software desarrollado profesionalmente espera características ocultas".

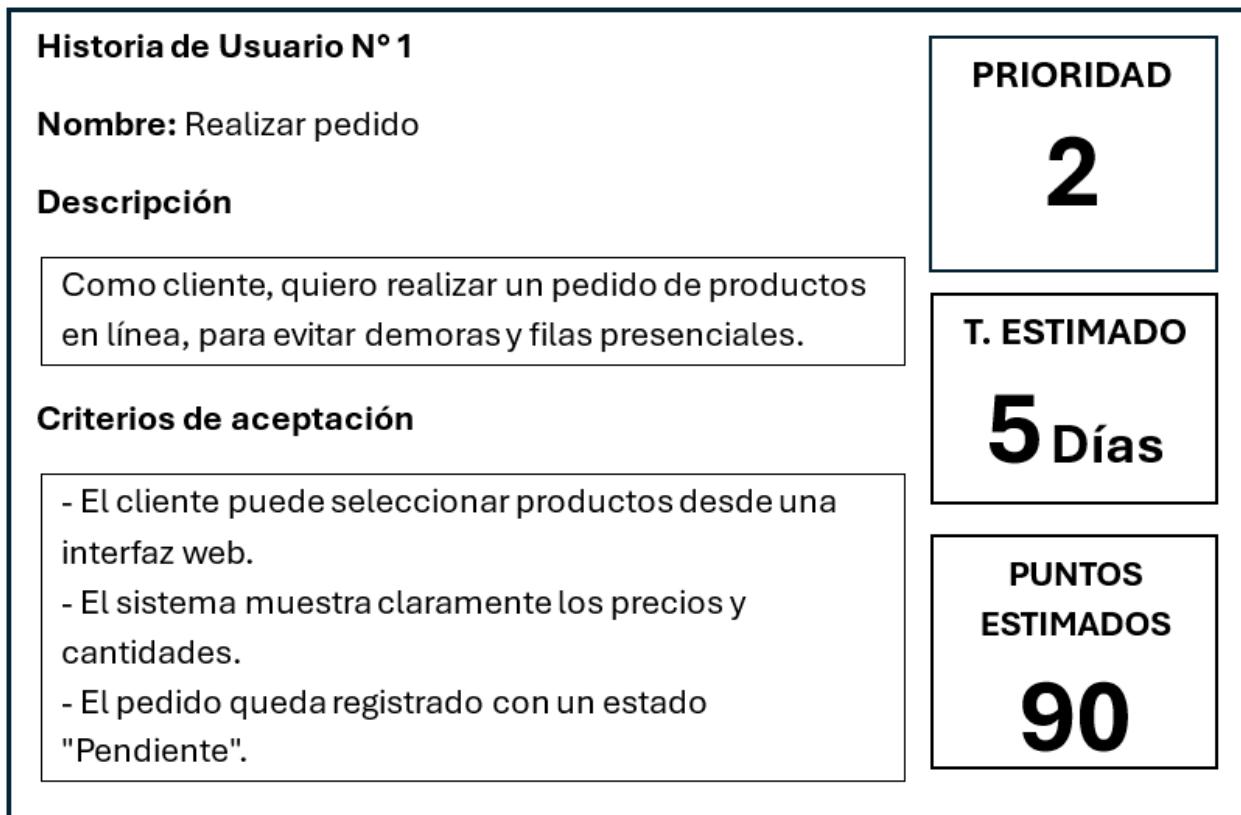
CAPITULO IV

IV. Desarrollo ágil del producto

4.1. Historias de usuario

Figura 2.

Historia de Usuario N°1



Nota: La imagen representa la Historia de Usuario N° 1 aplicada al sistema

Figura 3.

Historia de Usuario N°2

Historia de Usuario N° 2

Nombre: Ver disponibilidad

Descripción

Como cliente, quiero ver qué productos están disponibles antes de hacer un pedido, para evitar frustraciones por falta de stock.

Criterios de aceptación

- Solo se muestran productos con stock mayor a cero como disponibles.
- Se muestran los productos con stock igual a cero como agotados.
- No se puede seleccionar/elegir el producto agotado.
- La cantidad disponible se actualiza por compra hecha.

PRIORIDAD

2

T. ESTIMADO

3 Días

**PUNTOS
ESTIMADOS**

80

Nota: La imagen representa la Historia de Usuario N° 2 aplicada al sistema

Figura 4.

Historia de Usuario N°3

Nombre: Estado del pedido

Descripción

Como administrador, quiero visualizar y actualizar el estado de cada pedido, para controlar el proceso de entrega.

Criterios de aceptación

- Los pedidos pueden pasar de "pendiente" a "en proceso" y "entregado", hasta también en "cancelado".
- El cliente puede consultar el estado de su pedido.

PRIORIDAD

2

T. ESTIMADO

3 Días

PUNTOS
ESTIMADOS

40

Nota: La imagen representa la Historia de Usuario N° 3 aplicada al sistema

Figura 5.

Historia de Usuario N°4

Historia de Usuario N° 4

Nombre: Registrar productos

Descripción

Como administrador, quiero registrar, editar y eliminar productos en el inventario, para mantener actualizada la información del inventario.

Criterios de aceptación

- Se puede agregar, editar y eliminar productos.
- Los productos no se eliminan, solo cambia su estado.
- Cada producto tiene código, imagen, nombre, categoría, descripción, precio, stock, stock mínimo, disponibilidad y estado.

PRIORIDAD

1

T. ESTIMADO

3 Días

**PUNTOS
ESTIMADOS**

90

Nota: La imagen representa la Historia de Usuario N° 4 aplicada al sistema

Figura 6.

Historia de Usuario N°5

Historia de Usuario N° 5

Nombre: Actualización automática

Descripción

Como administrador, quiero que el inventario se actualice automáticamente cuando se hace un pedido, para reflejar el stock real.

Criterios de aceptación

- Se debe descontar la cantidad correspondiente del inventario, cuando se realiza una compra.
- Se debe visualizar el stock actual de los productos en la tabla productos del apartado administrador.

PRIORIDAD

1

T. ESTIMADO

3 Días

**PUNTOS
ESTIMADOS**

90

Nota: La imagen representa la Historia de Usuario N° 5 aplicada al sistema

Figura 7.

Historia de Usuario N°6

Historia de Usuario N° 6

Nombre: Alerta de stock

Descripción

Como administrador, quiero recibir una alerta cuando un producto tenga poco stock, para evitar quedarme sin productos clave.

Criterios de aceptación

- El sistema muestra una alerta visual cuando el stock es menor o igual al mínimo definido.
- Dicha alerta debe mostrarse cada vez que el administrador va hacia el apartado administrador de productos.

PRIORIDAD

1

T. ESTIMADO

2 Días

**PUNTOS
ESTIMADOS**

50

Nota: La imagen representa la Historia de Usuario N° 6 aplicada al sistema

Figura 8.

Historia de Usuario N°7

Historia de Usuario N°7

Nombre: Enviar consulta

Descripción

Como cliente, quiero enviar una consulta al negocio desde el sistema, para resolver dudas antes de hacer un pedido.

Criterios de aceptación

- Debe haber una ventana flotante de WhatsApp para las consultas.
- Se establece el número de celular para las consultas.
- El administrador puede ver las consultas mediante la cuenta de WhatsApp.

PRIORIDAD

3

T. ESTIMADO

1 Días

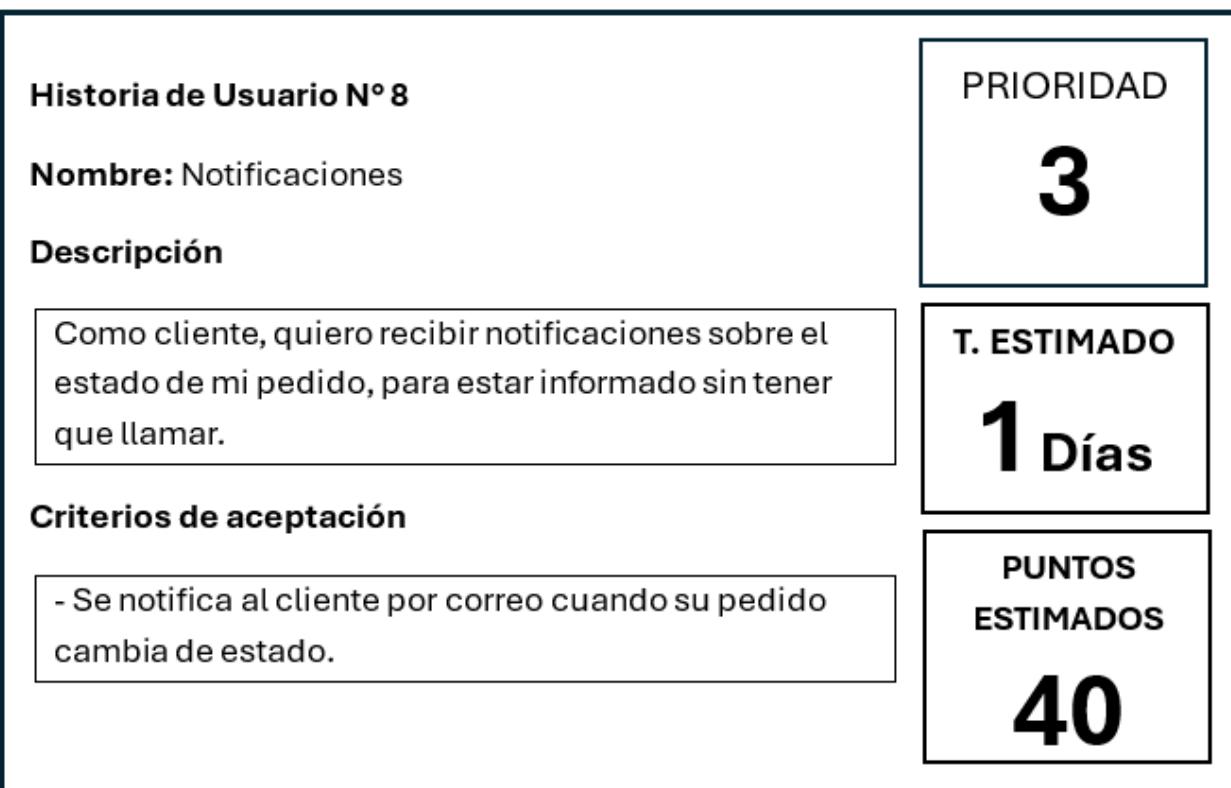
**PUNTOS
ESTIMADOS**

20

Nota: La imagen representa la Historia de Usuario N° 7 aplicada al sistema

Figura 9.

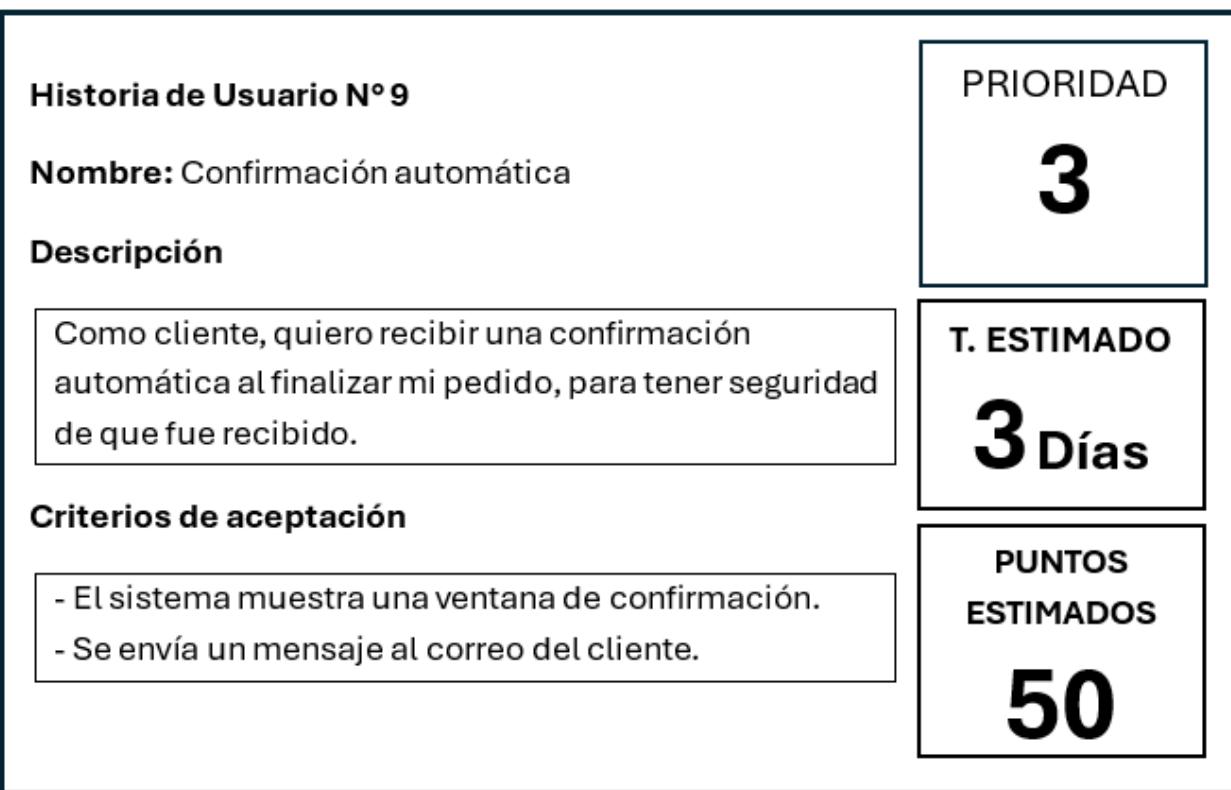
Historia de Usuario N°8



Nota: La imagen representa la Historia de Usuario N° 8 aplicada al sistema

Figura 10.

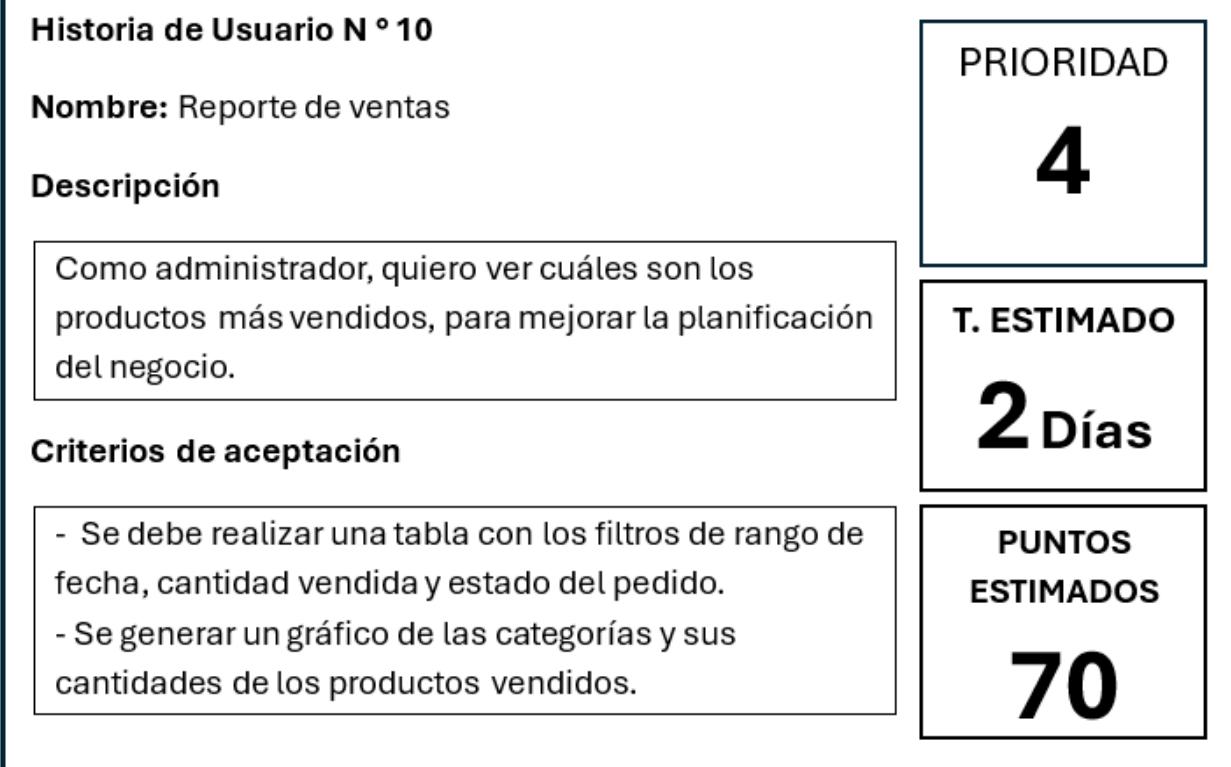
Historia de Usuario N°9



Nota: La imagen representa la Historia de Usuario N° 9 aplicada al sistema

Figura 11.

Historia de Usuario N°10



Nota: La imagen representa la Historia de Usuario N° 10 aplicada al sistema

Figura 12.

Historia de Usuario N°11

Historia de Usuario N°11

Nombre: Exportación

Descripción

Como administrador, quiero exportar los datos de los pedidos en Excel, para analizarlos o compartirlos fácilmente.

Criterios de aceptación

- Se debe tener un botón para exportar los reportes generados a través de una tabla.
- El archivo generado tiene formato legible.

PRIORIDAD

4

T. ESTIMADO

3 Días

**PUNTOS
ESTIMADOS**

40

Nota: La imagen representa la Historia de Usuario N° 11 aplicada al sistema

Figura 13.

Historia de Usuario N°12

Historia de Usuario N° 12

Nombre: Registro de cuenta cliente

Descripción

Como cliente, quiero poder crear una cuenta definiendo mis datos, para realizar mis pedidos.

Criterios de aceptación

- Se debe crear un apartado donde el cliente puede registrar sus datos como Nombre, Apellido, Teléfono, Correo, Contraseña, DNI, Dirección, Fecha de nacimiento y Distrito.
- Se debe aparecer una ventana confirmando la creación de su cuenta.

PRIORIDAD

5

T. ESTIMADO

2 Días

**PUNTOS
ESTIMADOS**

20

Nota: La imagen representa la Historia de Usuario N° 12 aplicada al sistema

Figura 14.

Historia de Usuario N°13

Historia de Usuario N° 13

Nombre: Inicio de sesión cliente

Descripción

Como cliente, quiero iniciar sesión con mis credenciales, para acceder de manera segura al sistema y realizar mis pedidos.

Criterios de aceptación

- Solo usuarios clientes autenticados pueden confirmar los pedidos.
- Se valida el correo y contraseña.

PRIORIDAD

5

T. ESTIMADO

2 Días

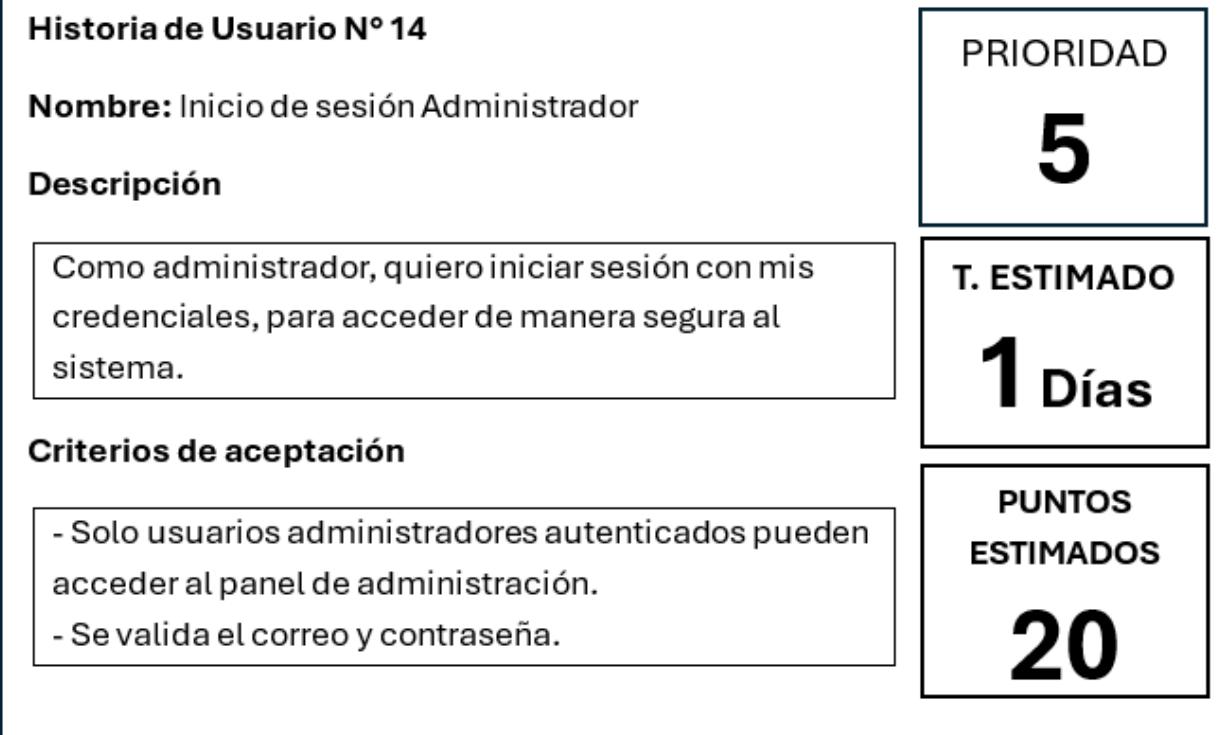
**PUNTOS
ESTIMADOS**

20

Nota: La imagen representa la Historia de Usuario N° 13 aplicada al sistema

Figura 15.

Historia de Usuario N°14



Nota: La imagen representa la Historia de Usuario N° 14 aplicada al sistema

4.2. Scrum team (Equipo Scrum)

El Equipo Scrum es un grupo multifuncional y autoorganizado responsable de entregar incrementos de producto valiosos al final de cada Sprint. Está compuesto por tres roles clave: el Product Owner (Dueño del Producto), el Scrum Master y el Equipo de Desarrollo. Cada miembro del equipo cumple funciones específicas que contribuyen al éxito del proyecto bajo el marco de trabajo Scrum.

Tabla 1.

Matriz equipo scrum

EQUIPO SCRUM	
INTEGRANTE	ROL
Álvaro Iván Arroyo Carrasco	Scrum Master
Zilva Ramirez Jose Luis	Product Owner
Frank Maiccol Vargas Mallqui	Developer
Gustavo Sebastian Rivera Moreno	Developer
Merly Andrea Angulo Canchero	Developer
Luis Miguel Santiago Caraza	Developer

Fuente: Elaboración propia

4.3. Matriz de Impacto

La matriz de impacto es una herramienta visual que ayuda a priorizar tareas según su prioridad.

Tabla 2.

Matriz de prioridad

PRIORIDAD	
Muy Alta	1
Alta	2
Media	3
Baja	4
Muy Baja	5

Fuente: Elaboración propia

4.4. Matriz de Puntos Estimados

La matriz de puntos estimados es una herramienta que nos ayuda a determinar los puntos estimados de recompensa que recibiríamos a cambio de lograr con ese requerimiento.

Tabla 3.

Matriz de puntos estimados

PUNTOS ESTIMADOS	
Demasiada alta	90
Muy alta	80
Alta	70
Media alta	60
Media	50
Media baja	40
Baja	30
Muy baja	20
Demasiada baja	10

Fuente: Elaboración propia

4.5. Product backlog

A continuación, se muestra el Product backlog donde se indica los requerimientos funcionales, así como el número de historia de usuario con su tiempo estimado, puntos estimados y prioridad.

Tabla 4.
Product backlog

REQUERIMIENTOS FUNCIONALES	HISTORIAS	T.E.	P.E.	PRIORIDAD
RF1: El sistema debe permitir a los clientes realizar pedidos en línea.	H1	5	90	2
RF2: El sistema debe mostrar la disponibilidad de productos.	H2	3	80	2
RF3: El sistema debe registrar los pedidos y asignarles un estado (pendiente, en proceso, entregado).	H3	3	40	2
RF4: El sistema debe registrar, editar y eliminar productos del inventario.	H4	3	90	1
RF5: El sistema debe actualizar el stock automáticamente cuando se realiza un pedido.	H5	3	90	1
RF6: El sistema debe generar alertas de stock bajo.	H6	2	50	1
RF7: El sistema debe permitir a los clientes enviar consultas o sugerencias.	H7	1	20	3
RF8: El sistema debe enviar notificaciones sobre el estado del pedido.	H8	1	40	3
RF9: El sistema debe mostrar mensajes automáticos de confirmación.	H9	3	50	3
RF10: El sistema debe generar reportes sobre productos más vendidos.	H10	2	70	4
RF11: El sistema debe exportar los datos a Excel o PDF.	H11	3	40	4
RF12: El sistema debe contar con apartado donde se puede registrar cuentas de tipo cliente.	H12	2	20	5
RF13: El sistema debe contar con inicio de sesión para clientes.	H13	2	20	5
RF14: El sistema debe contar con inicio de sesión para administrador.	H14	1	20	5

Fuente: Elaboración propia

4.6. Product backlog priorizado

A continuación, se muestra el Product Backlog correspondientemente ordenado de acuerdo con la prioridad establecida en la matriz de impacto, de igual forma incluye sus requerimientos funcionales como el número de historia de usuario, el tiempo y puntos estimados.

Tabla 5.

Product backlog priorizado

REQUERIMIENTOS FUNCIONALES	HISTORIAS	T.E.	P.E.	PRIORIDAD
RF4: El sistema debe registrar, editar y eliminar productos del inventario.	H4	3	90	1
RF5: El sistema debe actualizar el stock automáticamente cuando se realiza un pedido.	H5	3	90	1
RF6: El sistema debe generar alertas de stock bajo.	H6	2	50	1
RF1: El sistema debe permitir a los clientes realizar pedidos en línea.	H1	5	90	2
RF2: El sistema debe mostrar la disponibilidad de productos.	H2	3	80	2
RF3: El sistema debe registrar los pedidos y asignarles un estado (pendiente, en proceso, entregado).	H3	3	40	2
RF7: El sistema debe permitir a los clientes enviar consultas o sugerencias.	H7	1	20	3
RF8: El sistema debe enviar notificaciones sobre el estado del pedido.	H8	1	40	3
RF9: El sistema debe mostrar mensajes automáticos de confirmación.	H9	3	50	3
RF10: El sistema debe generar reportes sobre productos más vendidos.	H10	2	70	4
RF11: El sistema debe exportar los datos a Excel o PDF.	H11	3	40	4

RF12: El sistema debe contar con apartado donde se puede registrar cuentas de tipo cliente.	H12	2	20	5
RF13: El sistema debe contar con inicio de sesión para clientes.	H13	2	20	5
RF14: El sistema debe contar con inicio de sesión para administrador.	H14	1	20	5

Fuente: Elaboración propia

4.7. Entregables por sprint

En esta sección se especifica el número de Sprints, los requisitos funcionales del Product Backlog junto a sus prioridades y las estimaciones de tiempo correspondientes.

Tabla 6.

Entregables por sprint

MODULO	Nº SPRINT	REQUERIMIENTOS FUNCIONALES	H.U.	T.E.	P.E.	PRIORIDAD
Control de Inventario	SPRINT 1	RF4: El sistema debe registrar, editar y eliminar productos del inventario.	H4	3	90	1
		RF5: El sistema debe actualizar el stock automáticamente cuando se realiza un pedido.	H5	3	90	1
		RF6: El sistema debe generar alertas de stock bajo.	H6	2	50	1
Módulo de Gestión de Pedidos	SPRINT 2	RF1: El sistema debe permitir a los clientes realizar pedidos en línea.	H1	5	90	2
		RF2: El sistema debe mostrar la disponibilidad de productos.	H2	3	80	2
		RF3: El sistema debe registrar los pedidos y asignarles un estado (pendiente, en proceso, entregado, cancelado).	H3	3	40	2
Comunicación con Clientes	SPRINT 3	RF7: El sistema debe permitir a los clientes enviar consultas o sugerencias.	H7	1	20	3
		RF8: El sistema debe enviar notificaciones sobre el estado del pedido.	H8	1	40	3

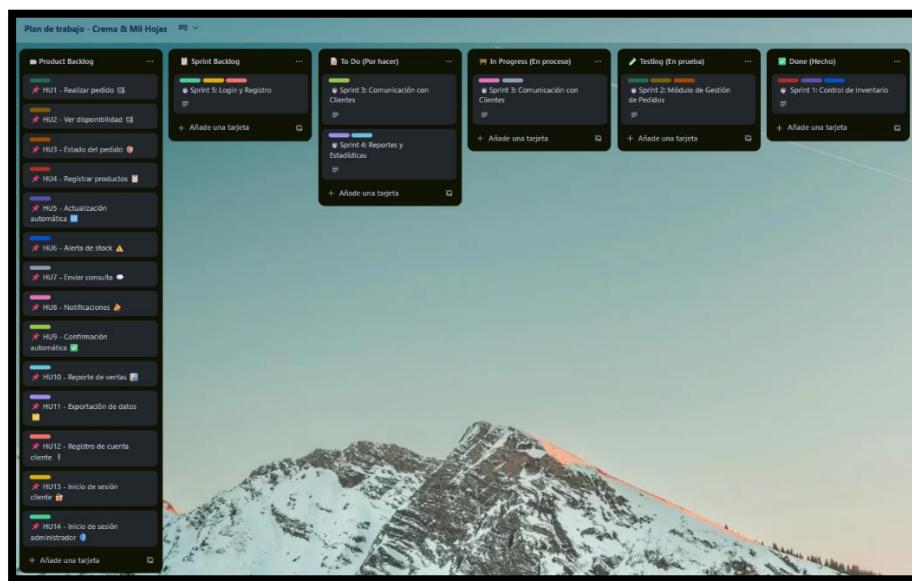
		RF9: El sistema debe mostrar mensajes automáticos de confirmación.	H9	3	50	3
Reportes y Estadísticas	SPRINT 4	RF10: El sistema debe generar reportes sobre productos más vendidos.	H10	2	70	4
		RF11: El sistema debe exportar los datos a Excel.	H11	3	40	4
Login y Registro	SPRINT 5	RF12: El sistema debe contar con apartado donde se puede registrar cuentas de tipo cliente.	H12	2	20	5
		RF13: El sistema debe contar con inicio de sesión para clientes.	H13	2	20	5
		RF14: El sistema debe contar con inicio de sesión para administrador.	H14	1	20	5

Fuente: Elaboración propia

4.8. Plan de trabajo

Figura 16.

Plan de trabajo



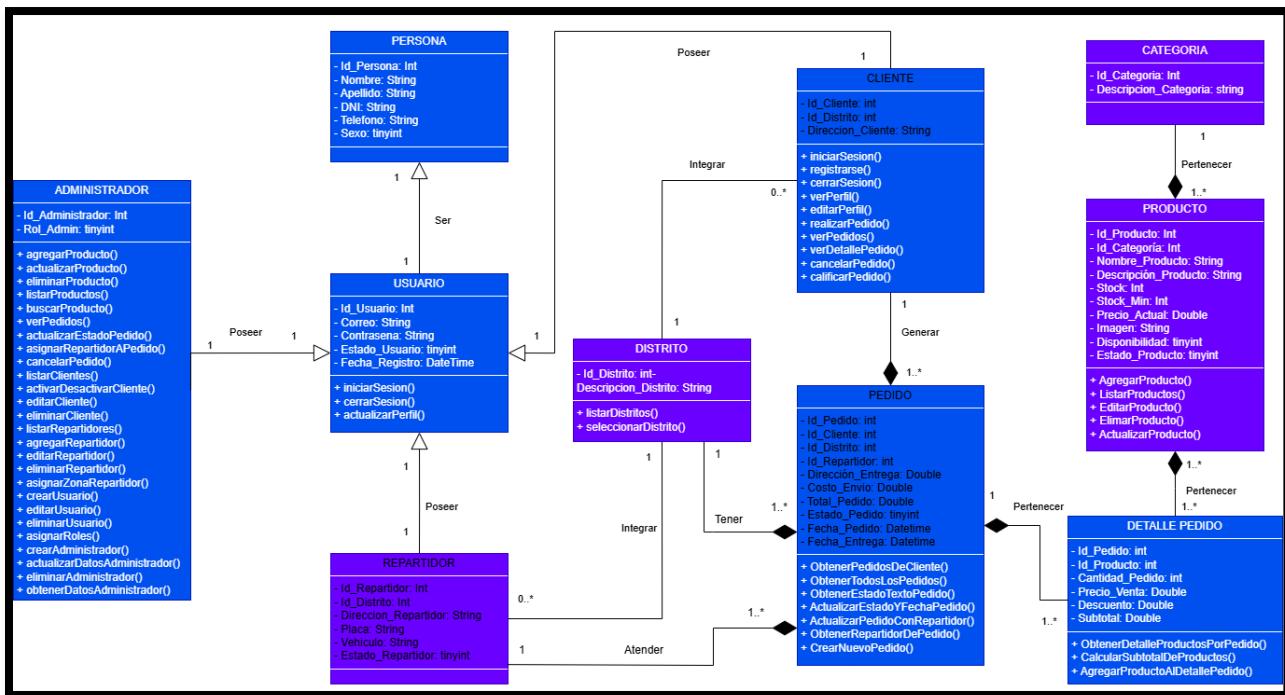
Nota: En esta imagen se visualiza el cronograma que se seguirá para el desarrollo del presente proyecto

4.9. Diagrama de clases de diseño

El siguiente diagrama muestra el diseño orientado a objetos y se utiliza para capturar la perspectiva del sistema para el desarrollo del software. A continuación, se observa las clases, atributos, relaciones y métodos(operaciones).

Figura 17.

Diagrama de clases de diseño



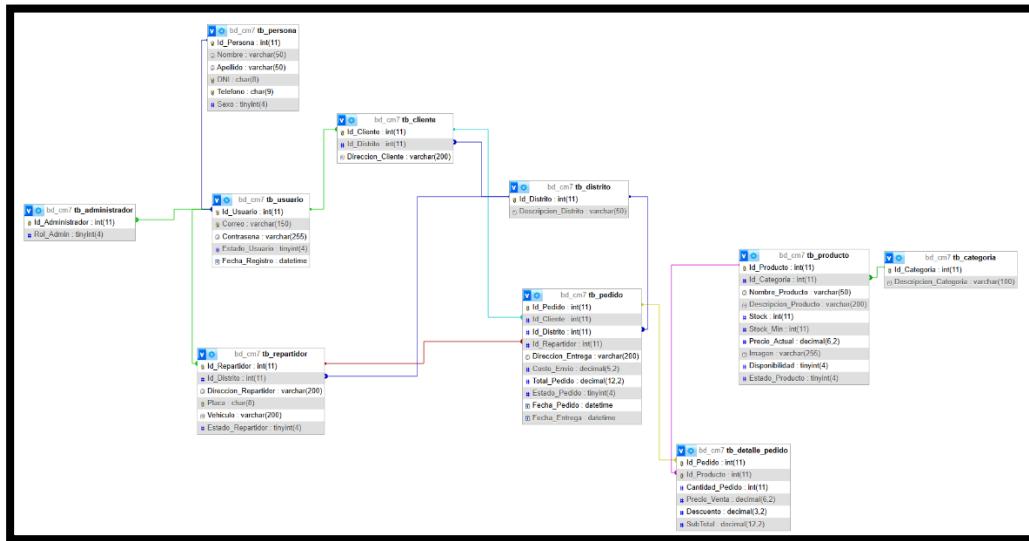
Nota: Elaboración propia

4.10. Diagrama físico de la base de datos

El siguiente diagrama muestra el diseño físico de la base de datos, donde se muestra todas las tablas relacionadas, en dónde se gestionará los datos del sistema web.

Figura 18.

Diagrama físico de la base de datos



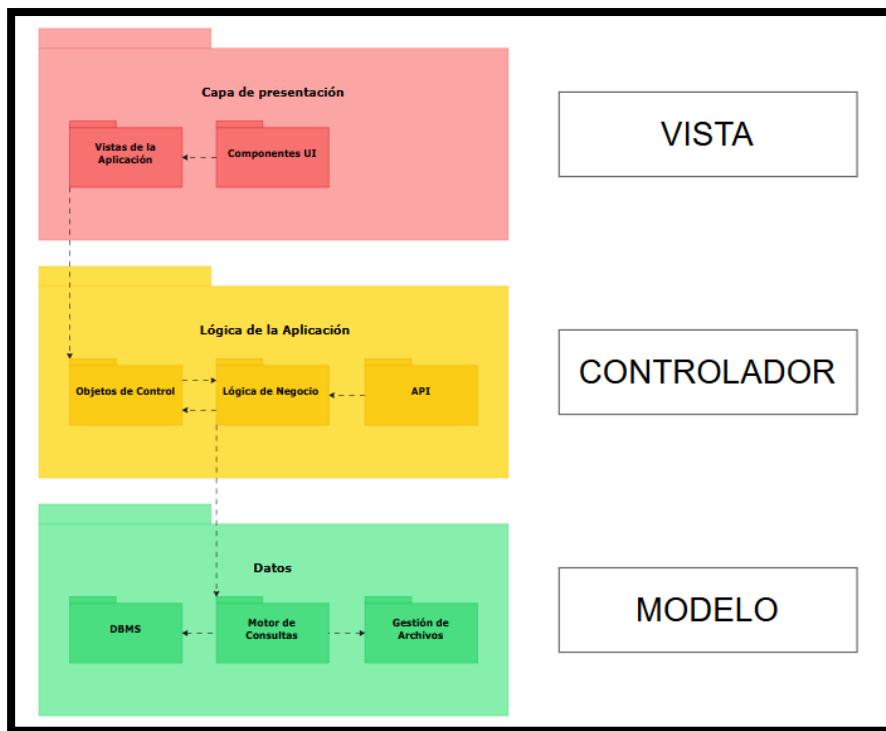
Nota: Elaboración propia

4.11. Diagrama de paquetes de la arquitectura de software

El siguiente diagrama representa el diseño del software de cómo es la estructura del sistema, por lo tanto, muestra los diferentes módulos como se agrupan y las tareas que se asignan.

Figura 19.

Diagrama de paquetes de la arquitectura de software



Nota: Elaboración propia

4.12. Sprint 1

4.12.1. Sprint Backlog

A continuación, se muestra el registro del Sprint Backlog número 1, el cual contiene los requisitos funcionales correspondientes.

Tabla 7.

Matriz del sprint N°1

Nº SPRINT	REQUERIMIENTOS FUNCIONALES	H.U.	T.E.	P.E.	PRIORIDAD
SPRINT 1	RF4: El sistema debe registrar, editar y eliminar productos del inventario.	H4	3	90	1
	RF5: El sistema debe actualizar el stock automáticamente cuando se realiza un pedido.	H5	3	90	1
	RF6: El sistema debe generar alertas de stock bajo.	H6	2	50	1

Fuente: Elaboración propia

4.12.2. Sprint Planning

En esta sección se detalla la manera en la cual se organizó el SPRINT N° 1 que se llevó a cabo el día 27/04/25 a las 8 a.m. con la presencia del Scrum Master, Product Owner y el equipo de desarrollo. Esta reunión tuvo una duración estimada de 3 horas, en donde el Scrum Master se encargó de organizar las tareas pertinentes a cada uno del equipo de desarrollo.

Tiempo estimado: 8 días.

4.12.3. Desarrollo

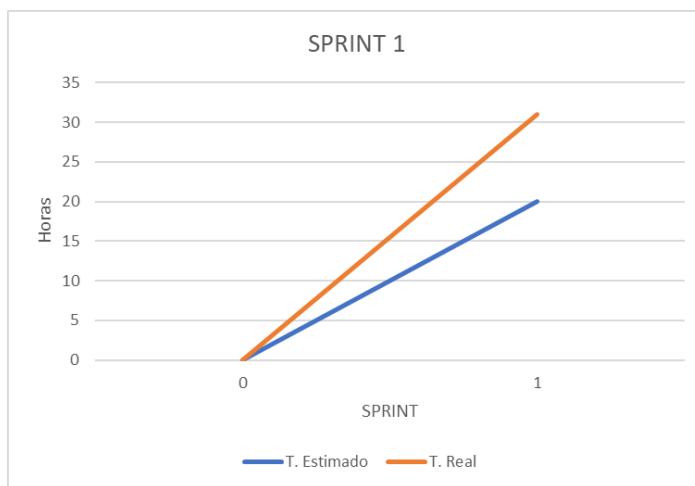
- **Requerimiento funcional N° 4:** El sistema debe registrar, editar y eliminar productos del inventario.
- **Requerimiento funcional N° 5:** El sistema debe actualizar el stock automáticamente cuando se realiza un pedido.
- **Requerimiento funcional N° 6:** El sistema debe generar alertas de stock bajo.

4.12.4. Burndown chart sprint 1

En siguiente figura se puede observar el Burndown chart del Sprint 1, el equipo estimo que esta actividad tomaría un tiempo estimado de 8 dias con una ganancia de 230 puntos estimados. En resumen, el desarrollo del Sprint 1 tuvo unos cambios y mejoras en el periodo de desarrollo, lo cual alargo el tiempo y llevo al incumplimiento del plazo estimado que se tenía.

Figura 20.

Burndown chart sprint N°1



Nota: La imagen representa el transcurso del tiempo (En horas) con respecto al Sprint N°1, siendo el tiempo real mayor que el estimado.

4.12.5. Review sprint 1

El equipo llevo a cabo una revisión del Sprint 1 el día 08 de Mayo del 2025 a las 4 p.m. con el objetivo de evaluar si se han cumplido con las actividades planificadas y si estas eran aprobadas por el Product Owner. Esta reunión involucró la participación del Scrum Master, el Product Owner y todos los miembros del equipo de desarrollo.

Tabla 8.

Matriz de review del sprint N°1

Nº	Requerimiento	Estado
1	Registrar productos	Cumplido
2	Actualización automática	Cumplido
3	Alerta de stock	Cumplido

Fuente: Elaboración propia

4.12.6. Matriz de casos de prueba

Tabla 9.

Matriz de casos de prueba del sprint N°1

Nº	Nº de historia de usuario relacionada	Nombre del caso de prueba	Pre - Condición	Pasos de ejecución	Resultados esperados	Estado (Pasó / Falló)
1	H4	Registrar productos	Como administrador, quiero registrar, editar y eliminar productos en el inventario, para mantener actualizada la información del inventario.	1. Clic en la interfaz Productos. 2. Clic en el botón Agregar Producto. 3. Rellenar los campos "nombre", "stock", "precio", "imagen", "categoria", "descripcion" y clic en el botón Guardar. 4. Para editar el producto; clic en el botón Editar.	El trabajador puede agregar, editar y eliminar productos.	Pasó

				5. Rellenar los campos "nombre", "stock", "precio", "imagen", "categoria", "descripcion". 6. Clic en el botón Guardar. 7. Para eliminar el producto; clic en el botón Eliminar.		
2	H5	Actualización automática	Como administrador, quiero que el inventario se actualice automáticamente cuando se hace un pedido, para reflejar el stock real.	1. Clic en la interfaz Productos. 2. Verificar si cada producto muestra su estado o disponibilidad (Activo/Inactivo, Disponible/No disponible). Cuando el cliente confirma su pedido, se descontará la cantidad correspondiente del stock del producto.	Al confirmar un pedido, el sistema descuenta la cantidad correspondiente del inventario.	Pasó
3	H6	Alerta de stock	Como administrador, quiero recibir una alerta cuando un producto tenga poco stock, para evitar quedarme sin productos clave.	1. Clic en la interfaz Productos. 2. El sistema muestra una alerta visual "Algunos productos están escasos o agotados. Verifica el inventario." cuando el stock es menor al mínimo definido.	El sistema muestra una alerta visual cuando el stock es menor al mínimo definido.	Pasó

Fuente: Elaboración propia

En la medida que se estuvo realizando el Sprint 1, se corroboró que los casos de prueba de la matriz de casos de prueba cumplían con los requerimientos de cada historia de usuario, pasando exitosamente cada uno de ellos con todas las pruebas realizadas.

4.13. Sprint 2

4.13.1. Sprint Backlog

A continuación, se muestra el registro del Sprint Backlog número 2, el cual contiene los requisitos funcionales correspondientes.

Tabla 10.

Matriz del sprint N°2

Nº SPRINT	REQUERIMIENTOS FUNCIONALES	H.U.	T.E.	P.E.	PRIORIDAD
SPRINT 2	RF1: El sistema debe permitir a los clientes realizar pedidos en línea.	H1	5	90	2
	RF2: El sistema debe mostrar la disponibilidad de productos.	H2	3	80	2
	RF3: El sistema debe registrar los pedidos y asignarles un estado (pendiente, en proceso, entregado).	H3	3	40	2

Fuente: Elaboración propia

4.13.2. Sprint Planning

En esta sección se detalla la manera en la cual se organizó el SPRINT N° 2 que se llevó a cabo el día 11/05/25 a las 8 a.m. con la presencia del Scrum Master, Product Owner y el equipo de desarrollo. Esta reunión tuvo una duración estimada de 4 horas, en donde el Scrum Master se encargó de organizar las tareas pertinentes a cada uno del equipo de desarrollo.

Tiempo estimado: 11 días.

4.13.3. Desarrollo

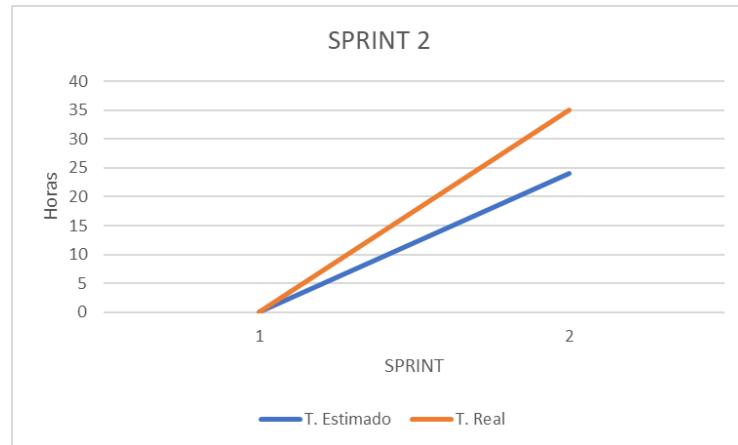
- **Requerimiento funcional N° 1:** El sistema debe permitir a los clientes realizar pedidos en línea.
- **Requerimiento funcional N° 2:** El sistema debe mostrar la disponibilidad de productos.
- **Requerimiento funcional N° 3:** El sistema debe registrar los pedidos y asignarles un estado (pendiente, en proceso, entregado).

4.13.4. Burndown chart sprint 2

En la siguiente figura se puede observar el Burndown chart del Sprint 2, el equipo estimo que esta actividad tomaría un tiempo estimado de 11 dias con una ganancia de 210 puntos estimados. En resumen, el desarrollo del Sprint 2 tuvo unos cambios y mejoras en el periodo de desarrollo, lo cual alargo el tiempo y llevo al incumplimiento del plazo estimado que se tenía.

Figura 21.

Burndown chart sprint N°2



Nota: La imagen representa el transcurso del tiempo (En horas) con respecto al Sprint N° 2, siendo el tiempo real mayor al estimado.

4.13.5. Review sprint 2

El equipo llevo a cabo una revisión del Sprint 2 el día 22 de Mayo del 2025 a las 4 p.m. con el objetivo de evaluar si se han cumplido con las actividades planificadas y si estas eran aprobadas por el Product Owner. Esta reunión involucró la participación del Scrum Master, el Product Owner y todos los miembros del equipo de desarrollo.

Tabla 11.**Matriz de review del sprint N°2**

Nº	Requerimiento	Estado
1	Realizar pedido	Cumplido
2	Ver disponibilidad	Cumplido
3	Estado del pedido	Cumplido

Fuente: Elaboración propia

4.13.6. Matriz de casos de prueba

Tabla 12.**Matriz de casos de prueba del sprint N°2**

Nº	Nº de historia de usuario relacionada	Nombre del caso de prueba	Pre - Condición	Pasos de ejecución	Resultados esperados	Estado (Pasó / Falló)
1	H1	Realizar pedido	Como cliente, quiero realizar un pedido de productos en línea, para evitar demoras y filas presenciales.	Clic en menú "Productos". Visualizará todos los productos. Seleccionas el producto y podrás ver el nombre y descripción del producto. Clic en el botón Agregar al carrito. Clic en el apartado "Carrito". Clic en el +/- para agregar o quitar cantidad del producto.	El administrador podrá realizar pedidos en línea satisfactoriamente	Pasó

				Clic en el botón "Seguir comprando" para seguir agregando productos al carrito.		
2	H2	Ver disponibilidad	Como cliente, quiero ver qué productos están disponibles antes de hacer un pedido, para evitar frustraciones por falta de stock.	Clic en menú "Productos". Podrá visualizar el catálogo de productos en stock. En cada producto aparece "Cantidad" el cual mostrará la disponibilidad del producto. No se dejará seleccionar el producto que esté agotado. Y cuando se seleccione el producto, la cantidad disponible se actualizará en tiempo real.	El cliente podrá visualizar si el producto posee stock o está agotado.	No Pasó
3	H3	Estado del pedido	Como administrador, quiero visualizar y actualizar el estado de cada pedido, para controlar el proceso de entrega.	Click en la interfaz Pedidos. Se visualizará la tabla de pedidos y se mostrarán sus estados. Para actualizar el estado de cada pedido; aplastar el botón "Editar". Editar el campo "Estado". Clic en el botón guardar.	El administrador podrá visualizar y actualizar el estado del pedido.	Pasó

Fuente: Elaboración propia

En la medida que se estuvo realizando el Sprint 2, se corroboró que los casos de prueba de la matriz de casos de prueba cumplieran con los requerimientos de cada historia de usuario, pasando exitosamente cada uno de ellos con todas las pruebas realizadas.

4.14. Sprint 3

4.14.1. Sprint Backlog

A continuación, se muestra el registro del Sprint Backlog número 3, el cual contiene los requisitos funcionales correspondientes.

Tabla 13.

Matriz del sprint N°3

Nº SPRINT	REQUERIMIENTOS FUNCIONALES	H.U.	T.E.	P.E.	PRIORIDAD
SPRINT 3	RF7: El sistema debe permitir a los clientes enviar consultas o sugerencias.	H7	1	20	3
	RF8: El sistema debe enviar notificaciones sobre el estado del pedido.	H8	1	40	3
	RF9: El sistema debe mostrar mensajes automáticos de confirmación.	H9	3	50	3

Fuente: Elaboración propia

4.14.2. Sprint Planning

En esta sección se detalla la manera en la cual se organizó el SPRINT N° 3 que se llevó a cabo el día 25/05/25 a las 8 a.m. con la presencia del Scrum Master, Product Owner y el equipo de desarrollo. Esta reunión tuvo una duración estimada de 4 horas, en donde el Scrum Master se encargó de organizar las tareas pertinentes a cada uno del equipo de desarrollo.

Tiempo estimado: 5 días.

4.14.3. Desarrollo

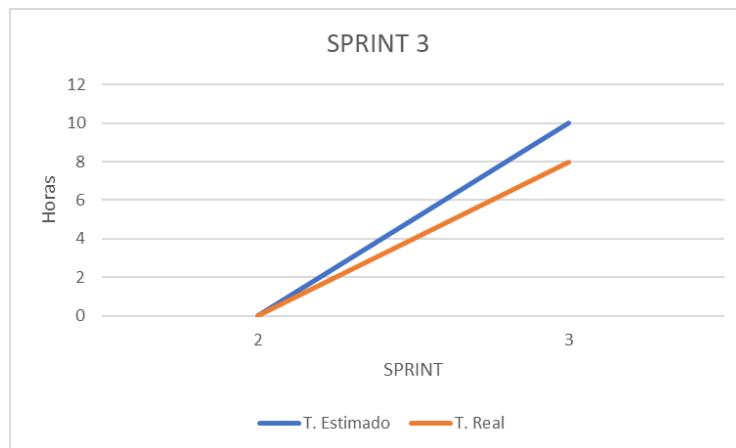
- **Requerimiento funcional N° 7:** El sistema debe permitir a los clientes enviar consultas o sugerencias.
- **Requerimiento funcional N° 8:** El sistema debe enviar notificaciones sobre el estado del pedido.
- **Requerimiento funcional N° 9:** El sistema debe mostrar mensajes automáticos de confirmación.

4.14.4. Burndown chart sprint 3

En la siguiente figura se puede observar el Burndown chart del Sprint 3, el equipo estimo que esta actividad tomaría un tiempo estimado de 5 dias con una ganancia de 110 puntos estimados. En resumen, el desarrollo del Sprint 3 en comparación con los anteriores sprint no tuvo inconvenientes, por tal motivo el tiempo de entrega acordado se llegó a cumplir con éxito y en el tiempo estimado.

Figura 22.

Burndown chart sprint N°3



Nota: La imagen representa el transcurso del tiempo (En horas) con respecto al Sprint N° 3, siendo el tiempo real menor al tiempo estimado.

4.14.5. Review sprint 3

El equipo llevo a cabo una revisión del Sprint 3 el día 29 de Mayo del 2025 a las 4 p.m. con el objetivo de evaluar si se han cumplido con las actividades planificadas y si estas eran aprobadas por el Product Owner. Esta reunión involucró la participación del Scrum Master, el Product Owner y todos los miembros del equipo de desarrollo.

Tabla 14.

Matriz de review del sprint N°3

Nº	Requerimiento	Estado
1	Enviar consulta	Cumplido
2	Notificaciones	Cumplido
3	Confirmación automática	Cumplido

Fuente: Elaboración propia

4.14.6. Matriz de casos de prueba

Tabla 15.

Matriz de casos de prueba del sprint N°3

Nº	Nº de historia de usuario relacionada	Nombre del caso de prueba	Pre - Condición	Pasos de ejecución	Resultados esperados	Estado (Pasó / Falló)
1	H7	Enviar consulta	Como cliente, quiero enviar una consulta al negocio desde el sistema, para resolver dudas antes de hacer un pedido.	1. Click en el logo de WhatsApp. 2. Redirige al usuario a una nueva pestaña en donde se abrirá WhatsApp con el numero de la empresa.	- Debe haber una ventana flotante de WhatsApp para las consultas. - Se establece el número de celular para las consultas. - El administrador puede ver las consultas mediante la cuenta de WhatsApp.	Pasó
2	H8	Notificaciones	Como cliente, quiero recibir notificaciones sobre el estado de mi pedido, para estar informado sin tener que llamar.	1. El encargado cambia el estado del pedido. 2. El sistema actualiza el estado del pedido. 3. El sistema notifica al cliente que el estado del	Se notifica al cliente por correo cuando su pedido cambia de estado	Pasó

				pedido a cambiado.		
3	H9	Confirmación automática	Como cliente, quiero recibir una confirmación automática al finalizar mi pedido, para tener seguridad de que fue recibido.	1. El sistema guarda la orden de pedido. 2. El sistema muestra un POP-UP mencionando que la orden se registró.	- El sistema muestra una ventana de confirmación. - Se envía un mensaje al correo del cliente.	Pasó

Fuente: Elaboración propia

En la medida que se estuvo realizando el Sprint 3, se corroboró que los casos de prueba de la matriz de casos de prueba cumplieran con los requerimientos de cada historia de usuario, pasando exitosamente cada uno de ellos con todas las pruebas realizadas.

4.15. Sprint 4

4.15.1. Sprint Backlog

A continuación, se muestra el registro del Sprint Backlog número 4, el cual contiene los requisitos funcionales correspondientes.

Tabla 16.

Matriz del sprint N°4

Nº SPRINT	REQUERIMIENTOS FUNCIONALES	H.U.	T.E.	P.E.	PRIORIDAD
SPRINT 4	RF10: El sistema debe generar reportes sobre productos más vendidos.	H10	2	70	4
	RF11: El sistema debe exportar los datos a Excel o PDF.	H11	3	40	4

Fuente: Elaboración propia

4.15.2. Sprint Planning

En esta sección se detalla la manera en la cual se organizó el SPRINT N° 4 que se llevó a cabo el día 01/06/25 a las 8 a.m. con la presencia del Scrum Master, Product Owner y el equipo de desarrollo. Esta reunión tuvo una duración estimada de 3 horas, en donde el Scrum Master se encargó de organizar las tareas pertinentes a cada uno del equipo de desarrollo.

Tiempo estimado: 5 días.

4.15.3. Desarrollo

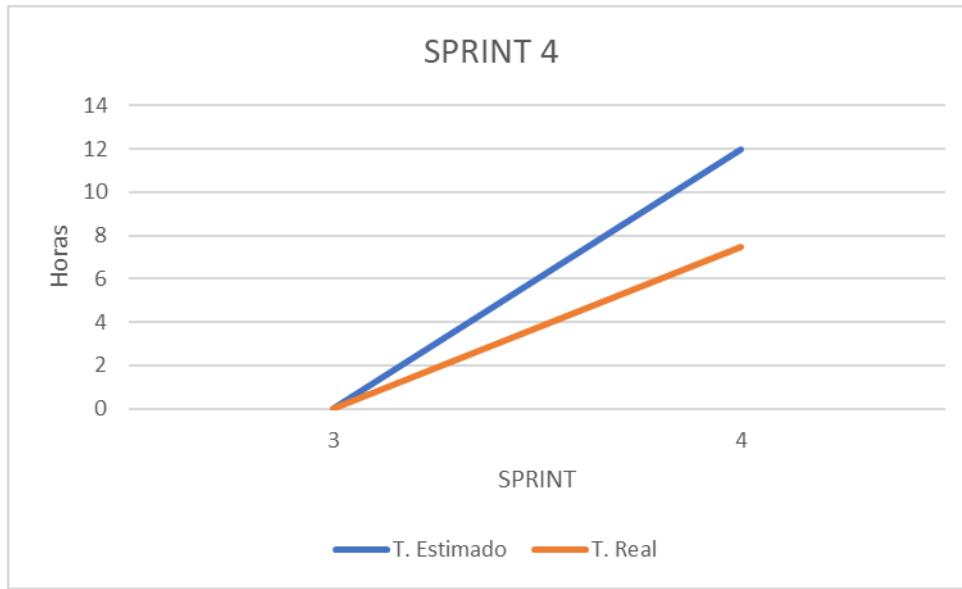
- **Requerimiento funcional N° 10:** El sistema debe generar reportes sobre productos más vendidos.
- **Requerimiento funcional N° 11:** El sistema debe exportar los datos a Excel.

4.15.4. Burndown chart sprint 4

En la siguiente figura puede observar el Burndown chart del Sprint 4, el equipo estimo que esta actividad tomaría un tiempo estimado de 5 dias con una ganancia de 110 puntos estimados. En resumen, el desarrollo del Sprint 4 en comparación con los anteriores sprint no tuvo inconvenientes, por tal motivo el tiempo de entrega acordado se llegó a cumplir con éxito y en el tiempo estimado.

Figura 23.

Burndown chart sprint N°4



Nota: La imagen representa el transcurso del tiempo (En horas) con respecto al Sprint N° 4, siendo el tiempo real menor que el estimado.

4.15.5. Review sprint 4

El equipo llevo a cabo una revisión del Sprint 4 el día 05 de Junio del 2025 a las 4 p.m. con el objetivo de evaluar si se han cumplido con las actividades planificadas y si estas eran aprobadas por el Product Owner. Esta reunión involucró la participación del Scrum Master, el Product Owner y todos los miembros del equipo de desarrollo.

Tabla 17.**Matriz de review del sprint N°4**

Nº	Requerimiento	Estado
1	Reporte de ventas	Cumplido
2	Exportación	Cumplido

Fuente: Elaboración propia

4.15.6. Matriz de casos de prueba

Tabla 18.**Matriz de casos de prueba del sprint N°4**

Nº	Nº de historia de usuario relacionada	Nombre del caso de prueba	Pre - Condición	Pasos de ejecución	Resultados esperados	Estado (Pasó / Falló)
1	H10	Reporte de ventas	Como administrador, quiero ver cuáles son los productos más vendidos, para mejorar la planificación del negocio.	1. Abrir la interfaz administrativa. 2. Click en la interfaz Dashboard. 3. El sistema muestra diferentes gráficos estadísticos de los productos.	Se genera un gráfico con productos y sus cantidades vendidas.	Pasó
2	H11	Exportación	Como administrador, quiero exportar los datos a Excel o PDF, para analizarlos o compartirlos fácilmente.	1. Abrir la interfaz de "Productos", "Clientes" o "Repartidores". 2. Seleccionar el botón "Exportar datos". 3. El sistema arroja una ventana para que el usuario seleccione donde guardar el archivo. 4. Click en guardar	El sistema exporta los datos en un archivo legible.	Pasó

				5. El sistema exporta todos los datos de la tabla.		
--	--	--	--	--	--	--

Fuente: Elaboración propia

En la medida que se estuvo realizando el Sprint 4, se corroboró que los casos de prueba de la matriz de casos de prueba cumplieran con los requerimientos de cada historia de usuario, pasando exitosamente cada uno de ellos con todas las pruebas realizadas.

4.16. Sprint 5

4.16.1. Sprint Backlog

A continuación, se muestra el registro del Sprint Backlog número 5, el cual contiene los requisitos funcionales correspondientes.

Tabla 19.

Matriz del sprint N°5

Nº SPRINT	REQUERIMIENTOS FUNCIONALES	H.U.	T.E.	P.E.	PRIORIDAD
SPRINT 5	RF12: El sistema debe contar con apartado donde se puede registrar cuentas de tipo cliente.	H12	2	20	5
	RF13: El sistema debe contar con inicio de sesión para clientes.	H13	2	20	5
	RF14: El sistema debe contar con inicio de sesión para administrador.	H14	1	20	5

Fuente: Elaboración propia

4.16.2. Sprint Planning

En esta sección se detalla la manera en la cual se organizó el SPRINT N° 5 que se llevó a cabo el día 08/06/25 a las 8 a.m. con la presencia del Scrum Master, Product Owner y el equipo de desarrollo. Esta reunión tuvo una duración estimada de 4 horas, en donde el Scrum Master se encargó de organizar las tareas pertinentes a cada uno del equipo de desarrollo.

Tiempo estimado: 5 días

4.16.3. Desarrollo

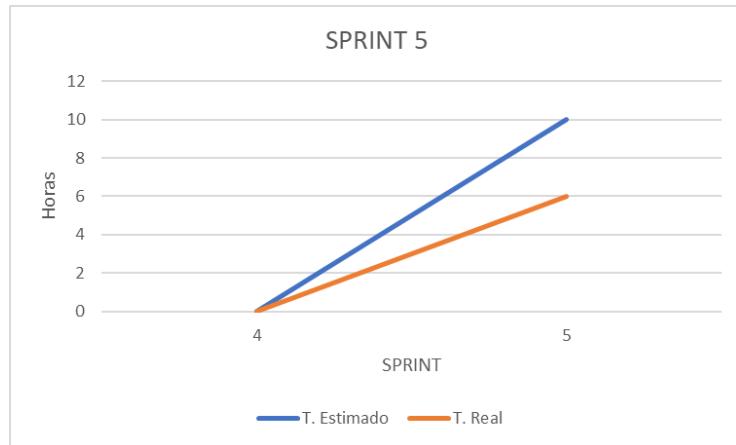
- **Requerimiento funcional N° 12:** El sistema debe contar con apartado donde se puede registrar cuentas de tipo cliente.
- **Requerimiento funcional N° 13:** El sistema debe contar con inicio de sesión para clientes.
- **Requerimiento funcional N° 14:** El sistema debe contar con inicio de sesión para administrador.

4.16.4. Burndown chart sprint 5

En la siguiente figura se puede observar el Burndown chart del Sprint 5, el equipo estimo que esta actividad tomaría un tiempo estimado de 5 días con una ganancia de 60 puntos estimados. En resumen, el desarrollo del Sprint 5 en comparación con los anteriores sprint no tuvo inconvenientes, por tal motivo el tiempo de entrega acordado se llegó a cumplir con éxito y en el tiempo estimado.

Figura 24.

Burndown chart sprint N°5



Nota: La imagen representa el transcurso del tiempo (En horas) con respecto al Sprint N° 5, siendo el tiempo real menor que el estimado.

4.16.5. Review sprint 5

El equipo llevo a cabo una revisión del Sprint 5 el día 12 de Junio del 2025 a las 4 p.m. con el objetivo de evaluar si se han cumplido con las actividades planificadas y si estas eran aprobadas por el Product Owner. Esta reunión involucró la participación del Scrum Master, el Product Owner y todos los miembros del equipo de desarrollo.

Tabla 20.

Matriz de review del sprint N°5

Nº	Requerimiento	Estado
1	Registro de cuenta cliente	Cumplido
2	Inicio de sesión cliente	Cumplido
3	Inicio de sesión Administrador	Cumplido

Fuente: Elaboración propia

4.16.6. Matriz de casos de prueba

Tabla 21.

Matriz de casos de prueba del sprint N°5

Nº	Nº de historia de usuario relacionada	Nombre del caso de prueba	Pre - Condición	Pasos de ejecución	Resultados esperados	Estado (Pasó / Falló)
1	H12	Registro de cuenta cliente	Como cliente, quiero poder crear una cuenta definiendo mis datos, para realizar mis pedidos.	1. Clic en el hipervínculo "Crear cuenta". 2. Completar todos los campos "nombre", "apellido", "teléfono", "correo electrónico", "contraseña", "dni", "dirección", "distrito". 3. Clic en el botón Registrar.	El usuario podrá registrarse en el sistema.	Pasó
2	H13	Inicio de sesión cliente	Como cliente, quiero iniciar sesión con mis credenciales, para acceder de manera segura al	1. Clic en el icono de usuario. 2. Completar el campo de "correo" y "contraseña". 3. Clic en el botón Ingresar.	El usuario podrá iniciar sesión en el sistema.	Pasó

			sistema y realizar mis pedidos.			
3	H14	Inicio de sesión Administrador	Como administrador, quiero iniciar sesión con mis credenciales, para acceder de manera segura al sistema.	<ol style="list-style-type: none"> 1. Clic en el ícono de usuario. 2. Completar el campo de "correo" y "contraseña" establecidas para el administrado. 3. Clic en el botón Ingresar. 	El administrador podrá ingresar en el sistema.	Pasó

Fuente: Elaboración propia

En la medida que se estuvo realizando el Sprint 5, se corroboró que los casos de prueba de la matriz de casos de prueba cumplieran con los requerimientos de cada historia de usuario, pasando exitosamente cada uno de ellos con todas las pruebas realizadas.

CAPITULO V

V. Conclusiones y recomendaciones

5.1. Conclusiones

- La implementación de un sistema web basado en los principios de calidad y prueba de software ha representado una transformación integral para la panadería Crema y Milhojas. La optimización de la atención al cliente y el control de inventario mediante la digitalización ha permitido reducir errores humanos, disminuir tiempos de espera y mejorar la satisfacción del cliente.
- La automatización del control de inventario ha sido un pilar clave en esta mejora, asegurando actualizaciones en tiempo real sobre la disponibilidad de productos, lo que previene errores de stock y optimiza la administración de insumos.
- La gestión eficiente de pedidos a través del módulo desarrollado ha facilitado una interacción más ágil con los clientes, mejorando su experiencia y permitiendo una adecuada planificación de la demanda diaria.
- El diseño de una interfaz intuitiva y accesible ha fortalecido la comunicación entre la panadería y sus clientes, proporcionando información clara y oportuna sobre productos y opciones de compra. Asimismo, la generación de reportes basados en datos ha optimizado la toma de decisiones estratégicas, permitiendo a la panadería ajustar su oferta de productos y anticipar las tendencias del mercado.
- Al integrar un apartado de reportes de las ventas, ha permitido contribuir en la toma de decisiones estratégicas en el negocio Crema y MilHojas.

- Finalmente, la digitalización de procesos internos ha reducido la carga de trabajo manual, optimizando la eficiencia operativa del negocio y permitiendo que el personal se enfoque en tareas de mayor valor agregado.

5.2. Recomendaciones

A continuación, se presentan una serie de recomendaciones que contribuirán a la optimización en la atención al cliente y control de inventario de la panadería crema & mil hojas. Estas recomendaciones están diseñadas para mejorar significativamente la eficiencia operativa, la calidad de la atención y la satisfacción general de los clientes.

- Se recomienda brindar capacitaciones periódicas al equipo de trabajo de la panadería para garantizar un uso correcto y eficiente del sistema implementado. Esto permitirá reducir errores operativos, aprovechar al máximo las funcionalidades del software y mejorar la experiencia del cliente.
- Es importante establecer un cronograma de mantenimiento preventivo y correctivo del sistema web, con el fin de asegurar su buen funcionamiento, prevenir fallos técnicos y mantener actualizadas las funcionalidades conforme a las necesidades del negocio.
- Para asegurar que el sistema sea amigable y útil para los usuarios, se recomienda aplicar encuestas breves o habilitar canales de sugerencias, a fin de identificar mejoras en la interfaz y adaptarla a las preferencias del cliente.
- A futuro, se puede evaluar la incorporación de opciones de pago en línea para facilitar el proceso de compra. Esto mejoraría la experiencia del usuario y permitiría ofrecer un servicio más completo y moderno.
- Se sugiere añadir funcionalidades como descuentos por compras frecuentes, mensajes de agradecimiento o promociones personalizadas para clientes

registrados. Estas acciones pueden incrementar la satisfacción y la lealtad del cliente.

- A medida que la panadería crezca, será necesario asegurar que el sistema web pueda adaptarse a una mayor cantidad de usuarios, pedidos y productos. Para ello, se recomienda revisar periódicamente la estructura del sistema y realizar las mejoras necesarias.
- Dado que el sistema maneja datos sensibles de clientes y transacciones, se recomienda fortalecer las medidas de seguridad, como el control de accesos, la validación de datos ingresados y la realización de copias de seguridad de manera periódica.

5.3. Glosario de términos

- **Sistema web:** Aplicación que se ejecuta en un servidor web y es accesible a través de un navegador. Permite la interacción entre usuarios y servicios mediante Internet.
- **Modelo Vista Controlador (MVC):** Patrón de arquitectura de software que separa la lógica de negocio (Modelo), la interfaz de usuario (Vista) y el control de flujo (Controlador) para mejorar la organización y mantenimiento del código.
- **Metodología Scrum:** Marco de trabajo ágil para la gestión de proyectos, especialmente en desarrollo de software, que promueve iteraciones cortas (sprints), trabajo en equipo y mejora continua.
- **Pruebas de software:** Conjunto de actividades destinadas a verificar que una aplicación funciona correctamente y cumple con los requisitos establecidos. Incluye pruebas funcionales y no funcionales.

- **Calidad de Software:** Medida de qué tan bien un software cumple con los requisitos funcionales y no funcionales, además de su confiabilidad, mantenibilidad y eficiencia.
- **Minimum Viable Product (MVP):** Producto con las características mínimas necesarias para funcionar y ser útil a los usuarios iniciales, con el fin de obtener retroalimentación rápida.
- **Depuración:** Proceso de identificar, analizar y corregir errores o fallos en el código fuente de un programa.
- **Refactorizar:** Mejorar la estructura interna del código sin alterar su comportamiento externo, con el objetivo de hacerlo más claro, eficiente o mantenable.
- **Front-End:** Parte del desarrollo web que se encarga de la interfaz visual con la que interactúa el usuario. Utiliza tecnologías como HTML, CSS y JavaScript.
- **Back-End:** Parte del desarrollo web encargada de la lógica de negocio, el manejo de datos y la comunicación con la base de datos y el servidor.
- **Testing:** Proceso de ejecución de un sistema con el objetivo de encontrar errores y asegurar que el software funcione correctamente bajo diferentes condiciones.
- **Desplegar:** Acción de publicar o poner en producción una aplicación o sistema, haciéndolo accesible a los usuarios finales.
- **Aceptación:** Etapa en la que el cliente o usuario final verifica y aprueba que el software cumple con los requisitos y está listo para ser utilizado.
- **Base de datos:** Conjunto de datos que se almacenan y gestionan electrónicamente, permitiendo su acceso, modificación y consulta eficiente.

VI. Bibliografía

- Andrade Vera, A. G. (7 de Junio de 2022). *Sistema web para la gestión de bodas y eventos*. Obtenido de Repositorio Universidad Estatal Península de Santa Elena: <https://repositorio.upse.edu.ec/handle/46000/7708>
- Andy Licuy, K. J., & Farías Pullaguari, L. A. (2024). *Desarrollo de un sistema web aplicando prácticas ágiles que permita gestionar la información de la piñatería "Camilita" ubicada en Latacunga*. Obtenido de Repositorio Universidad Técnica de Cotopaxi: <https://repositorio.utc.edu.ec/items/ea8a95a5-ead4-4800-9611-e52e8206afe8>
- Caicedo Macias, L. F., & Ruiz Montaño, P. A. (2024). *Desarrollo de una aplicación web para la recopilación de información de la fundación San Ezequiel Moreno*. Obtenido de Repositorio Institucional UniAri Uniagustiniana: <https://repositorio.uniagustiniana.edu.co/items/3f973d8f-ab1a-4a42-9e26-ee50645cc660>
- Calle Martinez, J. O., & Centeno Chaparro, G. (16 de Marzo de 2023). *Diseño de control de inventario de mercaderías y su incidencia en la rentabilidad de Ideas y Más Ideas S. A. C., Miraflores, 2020*. Obtenido de Repositorio Institucional UPN: <https://repositorio.upn.edu.pe/handle/11537/35333>
- Cañizares Guanoluisa, K. R., & Perrazo Cartagena, E. A. (2024). *Sistema web de agendamiento de citas médicas para el Centro de Estimulación Temprana Fonolandia*. Obtenido de Repositorio Institucional PUCE: <https://repositorio.puce.edu.ec/items/1f1a3d4d-3c3b-4d98-9367-9cc6e5a719ad>
- Chambi Bautista, L. F. (2021). *SISTEMA WEB DE CONTROL Y SEGUIMIENTO DE CONSULTA ODONTOLOGICA*. Obtenido de Repositorio Institucional UNIVERSIDAD PÚBLICA DE EL ALTO: <https://repositorio.upea.bo/jspui/handle/123456789/1291>
- Chávez, A. M. (2023). *Implementación de un sistema web y el control de inventario para una empresa de Lima, 2023*. Obtenido de Repositorio Institucional Digital de la Universidad Nacional del Callao: <https://repositorio.unac.edu.pe/handle/20.500.12952/9110>
- Coloma Fonseca, K. E. (6 de Mayo de 2025). *Desarrollo de un sistema web para la gestión de un centro de hospitalidad: módulo de gestión de habitaciones*. Obtenido de Repositorio Digital Institucional de la Escuela Politécnica Nacional: <https://bibdigital.epn.edu.ec/handle/15000/26359>
- de la Rosa Martín, T., & León González, J. L. (2024). *Sistema web para la creación de horarios de clases en la Universidad Metropolitana del Ecuador*. Obtenido de REMCA: <https://remca.umet.edu.ec/index.php/REMCA/article/view/698>
- Espinal Jarquín, I. L., Talavera Carranza, K. A., & Rizo Rodríguez, M. (2020). *Sistema web para la gestión de inventario y facturación*. Obtenido de Repositorio

- Institucional de la UNAN-Managua:
<https://repositorio.unan.edu.ni/id/eprint/13521/1/20082.pdf>
- Fajardo Chávez, J. A., & Lorenzo Alarcón, K. L. (2017). *Implementación de un sistema web para el control de inventario en la ferretería Christopher*. Obtenido de Repositorio Institucional UCH:
<https://repositorio.uch.edu.pe/handle/20.500.12872/111>
- Forra Layme, A. R. (2020). *SISTEMA WEB PARA LA VENTA DE LADRILLOS Y CONTROL DE PERSONAL*. Obtenido de Repositorio Institucional UNIVERSIDAD PÚBLICA DE EL ALTO:
<https://repositorio.upea.bo/jspui/handle/123456789/1220>
- González Rodríguez, J. J. (20 de Julio de 2021). *Desarrollo de un sistema web de gestión documental para la asociación de producción y comercialización de mayoristas de frutas y afines pacífico ASOPROCOMPA*. Obtenido de Repositorio Universidad Estatal Península de Santa Elena:
<https://repositorio.upse.edu.ec/handle/46000/5976>
- Hernández, U. (22 de Febrero de 2015). *MVC (Model, View, Controller) explicado*. Obtenido de CódigoFacilito: <https://codigofacilito.com/articulos/mvc-model-view-controller-explicado>
- Leon Cifuentes, I. M. (6 de Mayo de 2025). *Desarrollo de un sistema web para la gestión de un centro de hospitalidad: módulo de gestión de suministros*. Obtenido de Repositorio Digital Institucional de la Escuela Politécnica Nacional:
<https://bibdigital.epn.edu.ec/handle/15000/26360>
- Morejón Oña, D. A., & Romero Suntasig, A. M. (2023). *“Desarrollo de un sistema web para el control y gestión administrativa del gimnasio “Kevin Gym” del cantón La Maná”*. Obtenido de Repositorio Universidad Técnica de Cotopaxi:
<https://repositorio.utc.edu.ec/items/5fea8976-d7ad-45b9-960d-8217d2bd12aa>
- Nuñez Guerrero, J. D. (2024). *Sistema web basado en la arquitectura modelo vista controlador (MVC) para la gestión de fichas médicas de docentes y estudiantes de la unidad educativa González Suárez de la ciudad de Ambato*. Obtenido de Repositorio Universidad Técnica de Ambato:
<https://repositorio.uta.edu.ec/items/d8a7e315-e057-4f00-9d9d-4bbe67e661bc>
- Ocampo Fasabi, E., & Guevara Gonzales, C. F. (2020). *SISTEMA WEB DE ATENCIÓN Y ASISTENCIAS REMOTAS DE CLIENTES EN LA EMPRESA A & P INVERSIONES Y SERVICIOS S.A.C.* 2020. Obtenido de Repositorio Institucional Digital UNAP:
https://repositorio.unapiquitos.edu.pe/bitstream/handle/20.500.12737/8393/Edgar_d_Tesis_Titulo_2021.pdf
- Palomo Cajas, W. G. (2022). *Desarrollo de sistema web y aplicación móvil de venta de ropa de segunda mano : desarrollo de una aplicación móvil*. Obtenido de Repositorio Digital Institucional de la Escuela Politécnica Nacional:
<https://bibdigital.epn.edu.ec/handle/15000/23103>

- Palpa Castro, E. M. (2015). *Diseño e implementación del sistema web de atención al cliente en la empresa Super Tec S.A.C.* Obtenido de Repositorio Institucional UAP: <https://repositorio.uap.edu.pe/handle/20.500.12990/1963>
- Pérez Bautista, L. E. (Abril de 2023). *Desarrollo de un sistema web para la logística de una empresa de agua : Desarrollo del Frontend del sistema web para la logística de una empresa de agua.* Obtenido de Repositorio Digital Institucional de la Escuela Politécnica Nacional: <https://bibdigital.epn.edu.ec/handle/15000/23775>
- Quille Bernedo, G. A. (16 de Noviembre de 2023). *Implementación de un sistema web para la mejora del proceso de gestión de ventas en una empresa del sector de laboratorio de ensayos de alimentos y piensos.* Obtenido de Repositorio Académico UPC: <https://repositorioacademico.upc.edu.pe/handle/10757/671489>
- Rodríguez Del Pezo, R. A. (7 de Junio de 2022). *Desarrollo de un sistema web para la gestión de los procesos de entrega de víveres y texto, creación de horarios en la escuela de educación básica Nuevos Horizontes.* Obtenido de Repositorio Universidad Estatal Península de Santa Elena: <https://repositorio.upse.edu.ec/handle/46000/7710>
- Romero Castro, V., & Cantos Victores, A. L. (12 de Julio de 2021). *SISTEMA WEB DE GESTIÓN DE EVIDENCIA DE EVALUACIÓN PARA LA CARRERA DE INGENIERIA CIVIL.* Obtenido de Repositorio Digital UNESUM: <https://repositorio.unesum.edu.ec/handle/53000/3028>
- Rozo, M., Casanovas, I., & Pollo Cattaneo, M. F. (2020). *Transferencia de conocimiento en la gestión de calidad de la ingeniería de software.* Obtenido de Repositorio Institucional de la UNLP: <https://sedici.unlp.edu.ar/handle/10915/114345>
- Santa Cruz Álvarez, D. E., & Zapata Llanquiche, R. B. (2021). *Aplicación web para la clasificación de clientes de la empresa CFCGROUP - Arequipa 2021.* Obtenido de Repositorio Institucional Continental: <https://repositorio.continental.edu.pe/handle/20.500.12394/12289>
- Santillán Estrada, Y. G. (16 de Mayo de 2024). *Sistema web de gestión del proceso de ventas para la empresa "Konect Soluciones Tecnológicas CIA. LTDA. " : Desarrollo de un Sistema Web.* Obtenido de Repositorio Digital Institucional de la Escuela Politécnica Nacional: <https://bibdigital.epn.edu.ec/handle/15000/25520>
- Visual Paradigm Online. (2022). *Software de diagramas de paquetes en línea.* Obtenido de Visual Paradigm Online: <https://online.visual-paradigm.com/es/diagrams/features/package-diagram-software/>
- Yarin Achachaua, A. J., & Chacaliza Andia, P. J. (2020). *Implementación de un sistema web para optimizar la calidad del servicio administrativo en la parroquia nuestra señora de Guadalupe Ica, año 2019.* Obtenido de Repositorio Institucional Universidad Autónoma de Ica: <https://repositorio.autonomadeica.edu.pe/handle/20.500.14441/734>

VII. Anexos

Anexo 1. Formar el equipo de desarrollo y presentar el Formato de Proyecto de Software con las siguientes partes: título, integrantes y características técnicas del producto software a desarrollar.

❖ **Título**

Sistema web para la optimización del proceso de ventas de la panadería

Crema y Milhojas lima, 2025

❖ **Integrantes**

- Arroyo Carrasco, Álvaro Iván
- Angulo Canchero, Merly Andrea
- Rivero Moreno, Gustavo Sebastián
- Santiago Caraza, Luis Miguel
- Vargas Mallqui, Frank Maiccol

❖ **Características técnicas del producto software a desarrollar**

- Obligar iniciar sesión para realizar los pagos. • Poder registrarse como usuario
- Poseer un carrito de pedidos.
- Poder realizar los pagos.
- Debe tener un manejo intuitivo, su navegación no debe ser compleja.
- Debe poder reflejarse los datos de los productos de manera correcta junto con sus imágenes.

- Debe poder descartar los productos desde el carrito.
- Se debe poder mantener los productos del carrito cuando se haya finalizado la sesión.

Anexo 2. Implementación del software para el caso de uso Gestión de pedidos a un 50%

En esta sección se mostrará los avances del caso de uso gestión de pedidos, el cual se está abordando en el desarrollo del frontend para los apartados de la página principal, sección de productos, sección detalle del producto, la interfaz del carrito y con todos sus elementos con respondientes.

Para la creación de estas interfaces se ha utilizado diversas herramientas de desarrollo, con respecto al IDE ha sido Visual Studio Code, utilizando los frameworks para el apartado frontend como Tailwind CSS y Astro.

Además, se ha utilizado otras herramientas como Adobe illustrator para la realización y edición de las imágenes, Github para almacenar el proyecto, Git para el control de versiones y actualización del repositorio, también se usó Vercel para el hosting y con ello desplegar el proyecto.

❖ Página Principal

En este apartado del sistema web, se puede visualizar elementos principales como el encabezado que contiene el logo del negocio, las pestañas de opciones, el ícono de búsqueda (lupa) y el ícono de sesión. Además, se puede ver otras secciones como categorías de productos, productos destacados, anuncios, distribuidores, el pie de página (footer) con sus respectivos elementos y la ventana flotante llamada carrito en donde se establecerán los productos seleccionados.

The screenshot displays the website for 'Cremas y Millejón'. At the top, there is a navigation bar with links to 'Inicio', 'Productos', 'Sobre Nosotros', and 'Contacto'. A search bar and a user icon are also present. The main header features the text 'No es solo pan, es una experiencia para los sentidos' and 'Hacerlo mejor es lo que nos hace diferentes'. Below this, there are sections for 'Nuestros Productos', 'Delivery', 'Ingredientes y Calidad', and categories like 'Bollería y vienoiserie', 'Panes artesanales', 'Ediciones especiales', 'Dulces y pasteles', and 'Panadería saludable'. A 'Productos Destacados' section shows items such as 'Bollería + 1 un Milhojas', 'Repostería + 1 un Tarta de Manzana', 'Saludable + 5 un Pan Integral', 'Repostería + 1 un Pastel de Arándanos', 'Repostería + 1 un Pastel Red Velvet', and 'Exquisita + 1 un Empanada de Carne'. There are also promotional banners for 'Doble placer, doble tentación' and 'Un bocado, un susurro'. The footer contains links to 'Crema y Millejón', 'Enlaces Rápidos', 'Contáctanos', and 'Suscríbete', along with social media icons and a copyright notice.

❖ Sección de Productos

En este apartado del sistema web, se puede observar los productos que proporciona el negocio, dichos productos se pueden filtrar a través de las

secciones de filtros que se encuentra ubicado al lado izquierdo y además se puede visualizar los botones de las páginas debajo de los productos mostrados.

The screenshot displays the product listing page of the Crema y Milhojas website. On the left side, there are several filter sections:

- Categoría:** Includes checkboxes for "Bollería" (checked), "Dulces", "Especiales", "Pan artesanal" (checked), and "Pan saludable".
- Precios:** Includes checkboxes for price ranges: "Menos de S/ 5", "S/ 5 - S/ 10", "S/ 10 - S/ 25", and "Más de S/ 25".
- Tipo de Entrega:** Includes checkboxes for "Retiro en tienda" and "Entrega a domicilio".
- Preferencias:** Includes checkboxes for "Vegano", "Sin gluten", and "Sin lactosa".

The main content area shows a grid of nine bread products, each with an "AGREGAR" button:

Imagen	Nombre	Categoría	Precio	Opciones
	CROISSANT	BOLLERÍA	S/ 2.80	
	MIL HOJAS	BOLLERÍA	S/ 6.00	
	PAN CARIOLA	PAN ARTESANAL	S/ 2.50	
	PAN BAGUETTE	PAN ARTESANAL	S/ 2.20	
	PAN BRIOCHE	PAN ARTESANAL	S/ 2.90	
	PAN CACHITO	PAN ARTESANAL	S/ 2.80	Productos: 0
	PAN DE CENTENO	PAN ARTESANAL	S/ 5.90	
	CHOCOKARAMANDUKA	BOLLERÍA	S/ 4.30	
	PAN CIABATTA	PAN ARTESANAL	S/ 2.80	

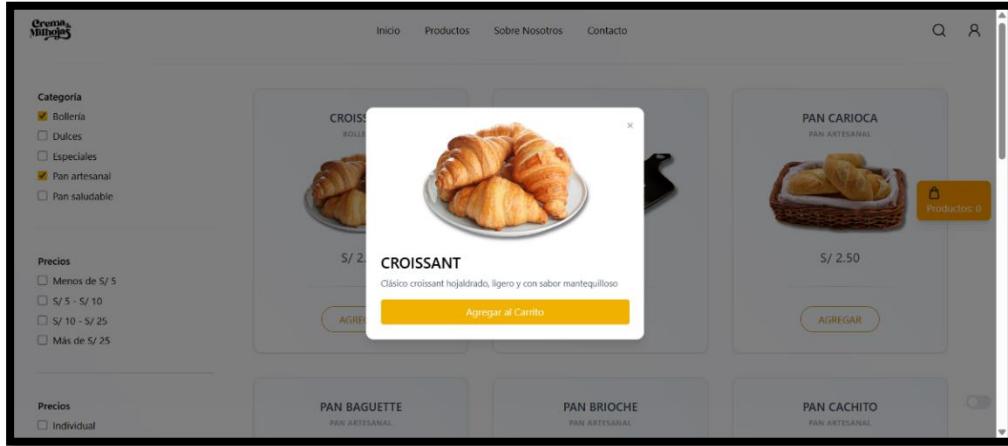
At the bottom of the page, there are two small buttons labeled "1" and "2". The footer contains the following information:

- Crema y Milhojas:** La mejor panadería para endulzar tus días. Nos especializamos en crear momentos inolvidables desde 1995. Social media links: Facebook, Instagram, Twitter, LinkedIn.
- Enlaces Rápidos:** Sobre Nosotros, Nuestro Menú, Pedidos Especiales, Servicios de Catering, Contactanos.
- Contactanos:** Av. Alfredo Mendiola 6062, +1 (555) 123-4567, contacto@cremamilhojas.com
- Suscríbete:** Recibe promociones y novedades exclusivas en tu correo. Ingrese tu email:

© 2024 Crema y Milhojas. Todos los derechos reservados.

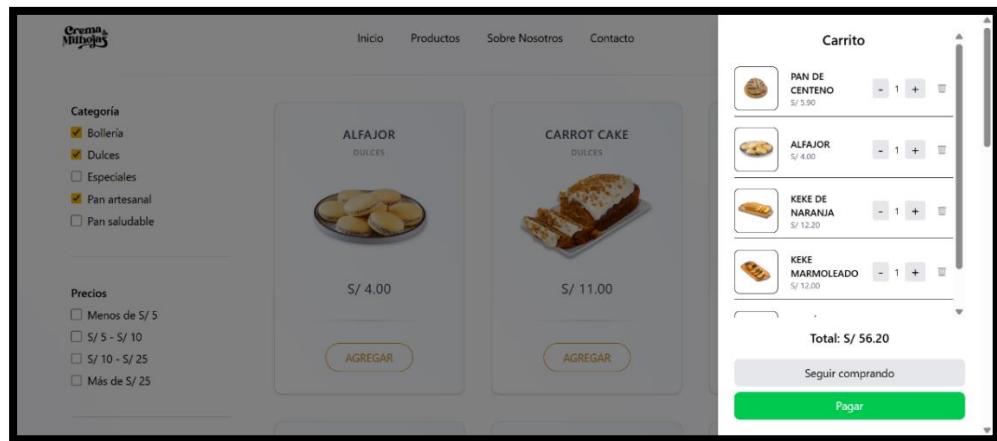
❖ Sección detalle del Producto

En esta sección de la página muestra la descripción de cada producto, mediante un pop-up del producto seleccionado, el cuál enfatiza ha dicho producto mostrándolo frente a la página actual y sombreándola.



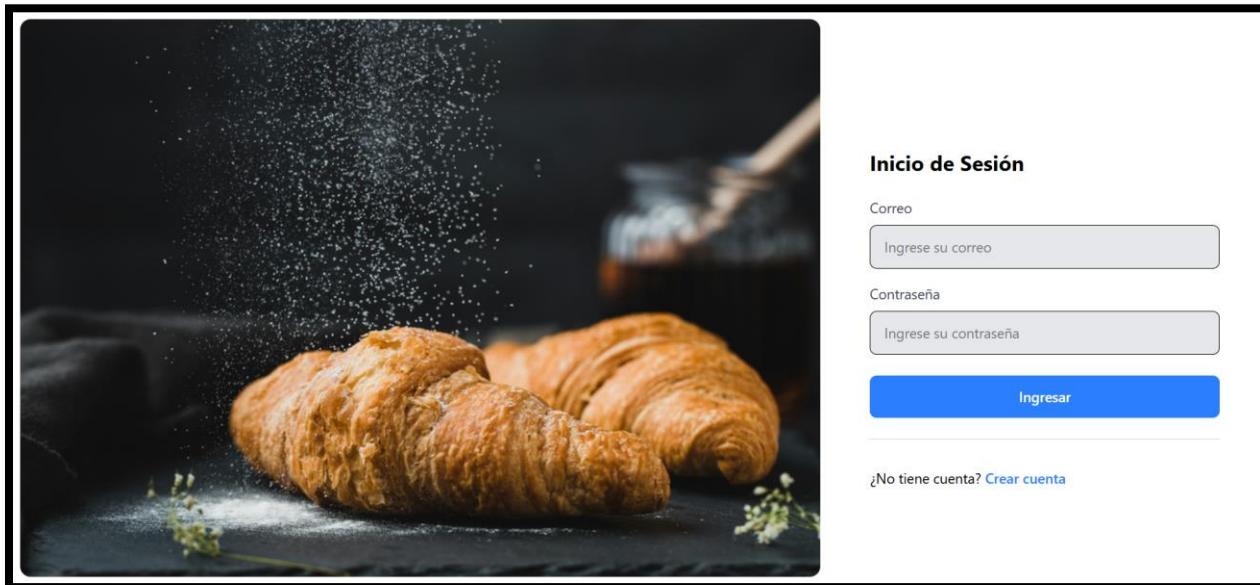
❖ Interfaz del Carrito

En esta sección del sistema, podemos observar todos los productos seleccionados, además podemos aumentar y disminuir la cantidad por producto, como también eliminar uno y ver el precio total a pagar. Por otro lado, las opciones (botones) de seguir comprando y pagar.



**Anexo 3. Implementación del software para el caso de uso Gestión de pedidos a un
100%**

❖ **Login**

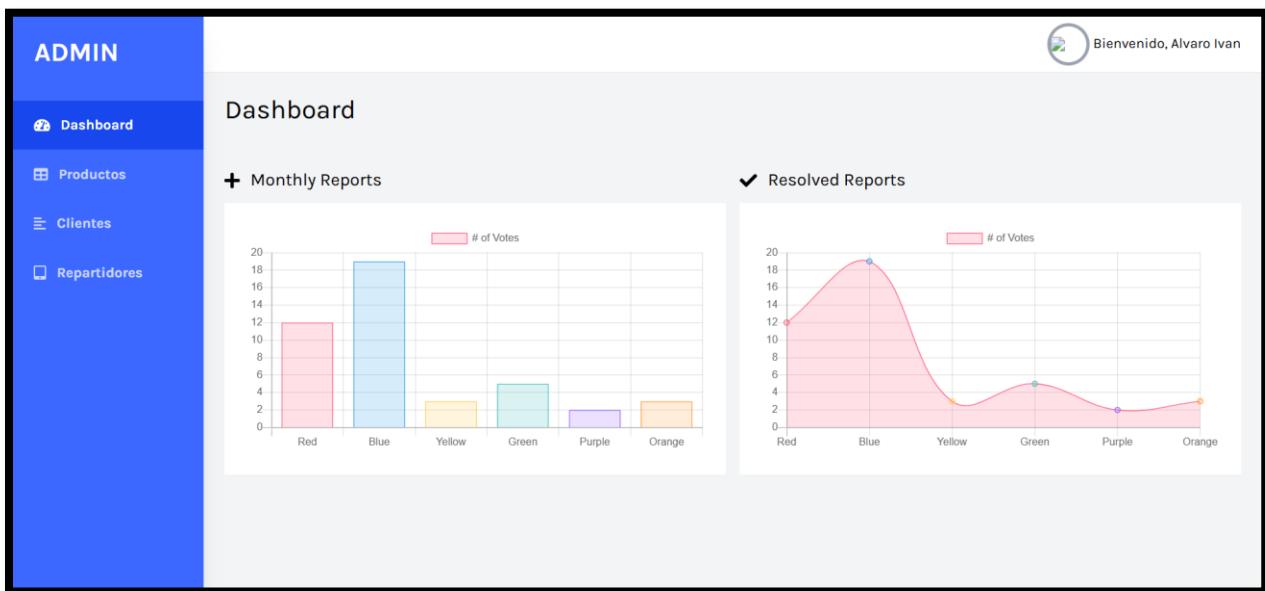


❖ Registro del usuario

The image shows a user registration form titled "Registro en la Panadería". The form consists of several input fields: Nombre (Name), Apellido (Last Name), Teléfono (Phone), Correo electrónico (Email), Contraseña (Password) with a visibility toggle, DNI, Dirección (Address), and a dropdown menu for "Selección un distrito" (Select a district). At the bottom is an orange "Registrar" (Register) button and a link "¿Ya tienes una cuenta? Iniciar sesión" (Already have an account? Log in).

❖ Apartados Administrativos

• Dashboard



• Productos

The screenshot shows the product management interface. On the left sidebar, under the 'Productos' section, there is a button labeled 'Agregar Producto'. The main area is titled 'Listado de Productos' and displays two items in a table:

CÓDIGO	IMAGEN	NOMBRE	DESCRIPCIÓN	STOCK	PRECIO	DISPONIBILIDAD	ESTADO
1		Karamandunga	dulce crocante	10	S/ 50.00	Agotado	0
2		Torta de Chocolate	Deliciosa torta de chocolate con crema	25	S/ 35.50	Disponible	1

• **Clientes**

The screenshot shows the client management interface. On the left sidebar, under the 'Clientes' section, there is a button labeled 'Agregar Cliente'. The main area is titled 'Listado de Clientes' and displays one client in a table:

DNI	NOMBRE	APELLIDO	TELÉFONO	GENERO	CORREO	DIRECCIÓN	ESTADO
75643215	Juan	Pérez	987654321	Macho	juan.perez@email.com	Av. Siempre Viva 123	Activo

- Repartidores

The screenshot shows the 'Listado de Clientes' (Client List) page within an admin interface. On the left, a sidebar titled 'ADMIN' lists 'Dashboard', 'Productos', 'Clientes', and 'Repartidores'. The main area has a search bar with fields for 'Buscar por DNI', 'Nombre', 'Apellido', 'Teléfono', and 'Correo'. Below the search bar are filters for 'Seleccionar Fecha:' (Select Date) and 'Género:' (Gender). A table displays client information with columns: DNI, NOMBRE, APELLIDO, TELÉFONO, GENERO, CORREO, DIRECCIÓN, PLACA, and VEHICULO. One row is visible: DNI 75643215, Nombre Juan, Apellido Pérez, Teléfono 987654321, Género Macho, Correo juan.perez@email.com, Dirección Av. Siempre Viva 123, Placa TGU-CHU, and Vehículo mustang shelby gt500.

- ❖ Apartados con el usuario registrado

- Inicio

The screenshot shows the homepage of 'Crema Milán'. At the top, there's a navigation bar with links to 'Inicio', 'Productos', 'Sobre Nosotros', and 'Contacto'. A user 'Bienvenido, Alvaro' is logged in. The main content features a large banner with the text 'No es solo pan, es una experiencia para los sentidos' (It's not just bread, it's an experience for the senses). Below the banner, there's a description: 'Cada capa es un susurro de sabor y cada mordida, un romance inolvidable' (Each layer is a whisper of flavor and each bite, an unforgettable romance). A call-to-action button says 'Pruébalo y siente la diferencia >'. To the right, there's a promotional section for 'NUEVO DESCAZ' (New Descaz) with images of bread and a price of '\$ 15.00'. A notification box indicates '2 Items'.

• **Productos**

The screenshot shows a web-based interface for managing inventory and customer service. At the top, there's a navigation bar with links for 'Inicio', 'Productos', 'Sobre Nosotros', and 'Contacto'. A user 'Bienvenido, Alvaro' is logged in, indicated by a profile icon and a welcome message. On the left, there are two sections: 'Categoría' and 'Precios'. Under 'Categoría', 'Dulces' is checked. Under 'Precios', 'Menos de S/ 5' is checked. Below these filters, there are three product cards: 'ALFAJOR' (DULCES, S/ 4.00), 'CARROT CAKE' (DULCES, S/ 11.00), and 'CHIFÓN DE NARANJA' (DULCES, S/ 22.10). Each card has an 'AGREGAR' button. A callout bubble shows '2 Items' and 'S/ 15.00'. At the bottom, there are four more product categories: 'CHOCO KEKE', 'KEKE GLASEADO', 'KEKE MARMOLEADO', and a toggle switch.

Categoría

Bollería

Dulces

Especiales

Pan artesanal

Pan saludable

Precios

Menos de S/ 5

S/ 5 - S/ 10

S/ 10 - S/ 25

Más de S/ 25

Porciones

ALFAJOR

DULCES

S/ 4.00

AGREGAR

CARROT CAKE

DULCES

S/ 11.00

AGREGAR

CHIFÓN DE NARANJA

DULCES

S/ 22.10

AGREGAR

2 Items

S/ 15.00

CHOCO KEKE

KEKE GLASEADO

KEKE MARMOLEADO

• Sobre nosotros

The screenshot displays the homepage of the Crema y Milhojas website. At the top, there's a navigation bar with links to Inicio, Productos, Sobre Nosotros, and Contacto, along with a search bar and a user profile icon. A large banner on the left features a croissant and a cup of coffee, with the text "HORNEAMOS CON ALMA, SERVIMOS CON PASIÓN." and "EL SABOR DE SIEMPRE, CON EL ENCANTO DE LO ETERNO." Below this, a timeline highlights key milestones:

- 2016**: Inicio del Sueño. Inauguración de la panadería.
- 2018**: Primer gran cambio. Menú renovado.
- 2020**: Adaptación. Pedidos a distancia.
- 2022**: Un compromiso con la calidad. Ingredientes naturales.
- 2024**: Crecemos juntos. Nueva imagen y mejoras.
- 2025**: No olvides sonreir.

The main content area features a large, bold headline "Lo mejor está por venir..." with three donuts floating above the text. At the bottom, there are footer sections for Crema y Milhojas, Enlaces Rápidos, Contáctanos, and a newsletter sign-up form. A copyright notice at the very bottom states: "© 2024 Crema y Milhojas. Todos los derechos reservados."

● Contacto

The screenshot shows the contact section of the Crema y Milhojas website. At the top, there's a navigation bar with links to Inicio, Productos, Sobre Nosotros, and Contacto. A search bar and a user greeting 'Bienvenido, Alvaro' are also present. The main content features a pink promotional message: 'Susúrranos tus antojos... — Y los haremos llegar a ti como un dulce pecado.' Below this is a smaller text: 'No hay espera eterna cuando el deseo es intenso... Un suspiro, un anhelo... y en un abrir y cerrar de ojos, nuestros más exquisitos manjares llegan hasta ti.' To the right is a large yellow graphic of a person carrying a red delivery bag. On the left, there's a yellow box containing a QR code and a form for messaging.

**Susúrranos tus antojos...
— Y los haremos llegar a ti
como un dulce pecado.**

No hay espera eterna cuando el deseo es intenso...
Un suspiro, un anhelo... y en un abrir y cerrar de ojos, nuestros más exquisitos manjares llegan hasta ti.

Escríbenos

Permitenos devolverte la dulzura de la vida, como un dulce abrazo que te envuelve en cada bocado.

Nombre: _____

Apellido: _____

Celular: _____

Correo: _____

Mensaje:

Enviar >

2 Items
S/ 15.00

Crema y Milhojas
La mejor panadería para endulzar tus días. Nos especializamos en crear momentos inolvidables desde 1995.
[Facebook](#) [Instagram](#) [Twitter](#) [YouTube](#)

Enlaces Rápidos
[Sobre Nosotros](#) [Nuestro Menú](#) [Pedidos Especiales](#) [Servicios de Catering](#) [Contáctanos](#)

Contáctanos
[Av. Alfredo Mendiola 6062](#) [+1 \(555\) 123-4567](#) contacto@cremamilhojas.com

Suscríbete
Recibe promociones y novedades exclusivas en tu correo.

© 2024 Crema y Milhojas. Todos los derechos reservados.

Anexo 4. Implementación de las pruebas unitarias del software para el caso de uso principal

Para la realización de las pruebas unitarias en el proyecto se ha usado el framework PHPUnit, el cual se obtuvo mediante gestor de dependencias composer. Se tuvo que crear una carpeta test desde la raíz del proyecto y dentro de ella se crearon los archivos de testeo para cada prueba unitaria.

❖ Pruebas unitarias del inicio de sesión

- Código de la clase ControlSesion

El siguiente código posee una función llamada autenticar() que permite autenticar los datos del correo y contraseña con los datos que tiene la base de datos para poder iniciar sesión.

```

1  <?php
2  class ControlSesion {
3      Tablone | Edit | Test | Explain | Document
4      public static function autenticar(PDO $conexion, string $correo, string $contrasena): array
5      {
6          $stmt_usuario = $conexion->prepare("SELECT Id_Usuario, Correo, Contrasena, Estado_Usuario FROM tb_usuario WHERE LOWER(Correo) = LOWER(:correo);");
7          $stmt_usuario->bindParam(":correo", $correo);
8          $stmt_usuario->execute();
9          $usuario = $stmt_usuario->fetch(PDO::FETCH_ASSOC);
10
11         if (!empty($usuario) && strcasecmp($usuario['Correo']) === strcasecmp($correo)) {
12             return ["success" => true, "message" => "Autenticación exitosa"];
13         }
14
15         $usuario_id = $usuario['Id_Usuario'];
16
17         $stmt_tipo = $conexion->prepare("
18             SELECT U.Id_Usuario, U.Correo, U.Contrasena, U.Estado_Usuario,
19             CASE
20                 WHEN C.Id_Cliente IS NOT NULL THEN 'Cliente'
21                 WHEN A.Id_Administrador IS NOT NULL THEN 'Administrador'
22                 WHEN R.Id_Repartidor IS NOT NULL THEN 'Repartidor'
23                 ELSE 'No especificado'
24             END AS Tipo_Usuario
25             FROM tb_usuario U
26             LEFT JOIN tb_cliente C ON U.Id_Usuario = C.Id_Cliente
27             LEFT JOIN tb_administrador A ON U.Id_Usuario = A.Id_Administrador
28             LEFT JOIN tb_repartidor R ON U.Id_Usuario = R.Id_Repartidor
29             WHERE U.Id_Usuario = :idusuario;
30         ");
31         $stmt_tipo->bindParam(':idusuario', $usuario_id);
32         $stmt_tipo->execute();
33         $usuario = $stmt_tipo->fetch(PDO::FETCH_ASSOC);
34     }
35 }
```

```

34
35     if (!$usuario) {
36         return ["success" => false, "message" => "Usuario no encontrado"];
37     }
38
39     if (!password_verify($contrasena, $usuario['Contraseña'])) {
40         return ["success" => false, "message" => "Contraseña Incorrecta"];
41     }
42
43     if ($usuario['Estado_Usuario'] == 0) {
44         return ["success" => false, "message" => "Cuenta inhabilitada"];
45     }
46
47     $redirect = match($usuario['Tipo_Usuario']) {
48         'Administrador' => 'Admin/admin-page.php',
49         'Cliente', 'Repartidor' => 'index.php',
50         default => null
51     };
52
53     if (!$redirect) {
54         return ["success" => false, "message" => "Tipo de usuario no válido"];
55     }
56
57     return [
58         "success" => true,
59         "message" => "Inicio de sesión como " . $usuario['Tipo_Usuario'],
60         "redirect" => $redirect
61     ];
62 }
63 ?>

```

- **Código de testeo**

Código que permite evaluar la lógica de la función autenticar() que posee la clase ControlSesion.

```

1  <?php
2  use PHPUnit\Framework\TestCase;
3
4  class ControlSesionTest extends TestCase
5  {
6      Tabnine [Edit | Test | Explain | Document
7      public function testInicioSessionExitoso()
8      {
9          $pdo = $this->createMock(PDO::class);
10         $stmtMock = $this->createMock(PDOStatement::class);
11
12         // Simula prepare() dos veces: usuario + tipo
13         $pdo->method("prepare")->willReturn($stmtMock);
14
15         // Simula execute() exitoso
16         $stmtMock->method("execute")->willReturn(true);
17
18         // Simula fetch() dos veces (usuario y luego tipo)
19         $stmtMock->method("fetch")->willReturnOnConsecutiveCalls(
20             [ // primer SELECT
21                 'Id_Usuario' => 1,
22                 'Correo' => 'test@correo.com',
23                 'Contraseña' => password_hash('1234', PASSWORD_DEFAULT),
24                 'Estado_Usuario' => 1,
25             ],
26             [ // segundo SELECT
27                 'Tipo_Usuario' => 'Cliente',
28                 'Contraseña' => password_hash('1234', PASSWORD_DEFAULT),
29                 'Estado_Usuario' => 1,
30                 'Correo' => 'test@correo.com',
31                 'Id_Usuario' => 1
32             ]
33         );
34
35         $resultado = ControlSesion::autenticar($pdo, 'test@correo.com', '1234');
36
37         $this->assertTrue($resultado['success']);
38         $this->assertEqual('Inicio de sesión como cliente', $resultado['message']);
39         $this->assertEquals('index.php', $resultado['redirect']);
40     }
41 }

```

```

41     public function testCorreoInvalido()
42     {
43         $pdo = $this->createMock(PDO::class);
44         $stmt = $this->createMock(PDOStatement::class);
45
46         $pdo->method('prepare')->willReturn($stmt);
47         $stmt->method('execute')->willReturn(true);
48         $stmt->method('fetch')->willReturn(false); // Simula que no encontró el correo
49
50         $resultado = ControlSesion::autenticar($pdo, 'noexiste@correo.com', 'cualquierClave');
51
52         $this->assertFalse($resultado['success']);
53         $this->assertSame('Correo inválido', $resultado['message']);
54     }
55
56     Tablero | Edit | Test | Explain | Document
57     public function testContrasenaIncorrecta()
58     {
59         $pdo = $this->createMock(PDO::class);
60         $stmt = $this->createMock(PDOStatement::class);
61
62         $pdo->method('prepare')->willReturn($stmt);
63
64         $stmt->method('execute')->willReturn(true);
65
66         // Simula fetch del primer query con usuario encontrado
67         $stmt->method('fetch')->willReturn($this->consecutivasCalls( // 'onConsecutiveCalls' is deprecated.
68             [
69                 'Id_Usuario' => 1,
70                 'Correo' => 'ejemplo@correo.com',
71                 'Contrasena' => password_hash('claveCorrecta', PASSWORD_DEFAULT),
72                 'Estado_Usuario' => 1
73             ],
74             [
75                 // Simula fetch del segundo query (tipo de usuario)
76                 'Tipo_Usuario' => 'Cliente',
77                 'Contrasena' => password_hash('claveCorrecta', PASSWORD_DEFAULT),
78                 'Estado_Usuario' => 1,
79                 'Correo' => 'ejemplo@correo.com',
80                 'Id_Usuario' => 1
81             ]
82         ));
83
84         $resultado = ControlSesion::autenticar($pdo, 'ejemplo@correo.com', 'claveIncorrecta');
85         $this->assertFalse($resultado['success']);
86         $this->assertSame('Contraseña incorrecta', $resultado['message']);
87     }
88
89 
```

- **Resultado del testeо**

```

C:\xampp\htdocs\Crema-y-Mil-Hojas-Front-End (0.794s)
./vendor/bin/phpunit test/ControlSesionTest.php --display-deprecations
PHPUnit 11.5.19 by Sebastian Bergmann and contributors.

Runtime:      PHP 8.2.12
..D
3 / 3 (100%)

Time: 00:00.427, Memory: 8.00 MB

OK, but there were issues!
Tests: 3, Assertions: 7, PHPUnit Deprecations: 1.

```

De acuerdo con el resultado obtenido se detalla lo siguiente:

Qué en el testeо para evaluar la lógica de inicio de sesión se hacen 3 procesos, los cuales son sesión exitosa, correo Invalido y contraseña Incorrecta. Esto se separan en 3 funciones diferentes para el test, dando como resultado una prueba exitosa. Los cuales fueron:

Tests: 3 => Las 3 funciones

Assertions: 7 => La cantidad de afirmaciones validadas

Deprecations: 1 => Se está usando algo que sigue funcionando ahora, pero en el futuro será eliminado o cambiará de comportamiento en PHPUnit.

❖ Pruebas unitarias de la creación de cuenta

- CODIFICACIÓN DE LOGICA

```

1 <?php
2 class Usuario
3 {
4     private $db;
5
6     public function __construct($db)
7     {
8         $this->db = $db;
9     }
10
11    Tabnine | Edit | Test | Explain | Document
12    public function agregarUsuario($nombre, $apellido, $telefono, $correo, $contrasena, $dni, $direccion, $distrito)
13    {
14        // Encriptar la contraseña
15        $contraseña_encriptada = password_hash($contrasena, PASSWORD_DEFAULT);
16
17        // Iniciar transacción
18        $this->db->beginTransaction();
19
20        try {
21            // Consultas para insertar los datos en las tablas correspondientes
22            $sql_persona = $this->db->prepare("INSERT INTO tb_persona (Nombre, Apellido, DNI, Telefono) VALUES (?, ?, ?, ?)");
23            $resultado_persona = $sql_persona->execute([$nombre, $apellido, $dni, $telefono]);
24
25            // Obtener el ID de la última persona insertada
26            $id_persona = $this->db->lastInsertId();
27
28            $sql_usuario = $this->db->prepare("INSERT INTO tb_usuario (Id_Usuario, Correo, Contrasena) VALUES (?, ?, ?)");
29            $resultado_usuario = $sql_usuario->execute([$correo, $contraseña_encriptada]);
30
31            $sql_cliente = $this->db->prepare("INSERT INTO tb_cliente (Id_Cliente, Id_Distrito, Direccion_Cliente) VALUES (?, ?, ?)");
32            $resultado_cliente = $sql_cliente->execute([$id_persona, $distrito, $direccion]);
33
34            // Verifica que todas las consultas se ejecutaron correctamente
35            if ($resultado_persona && $resultado_usuario && $resultado_cliente) {
36                $this->db->commit();
37                return ['status' => 'success', 'message' => 'Usuario agregado correctamente'];
38            } else {
39                $this->db->rollback();
40                return ['status' => 'error', 'message' => 'Error al agregar el usuario'];
41            }
42        } catch (Exception $e) {
43            $this->db->rollback();
44            return ['status' => 'error', 'message' => 'Error: ' . $e->getMessage()];
45        }
46    }
}

```

- Código de testeo

```

1 <?php
2 use PHPUnit\Framework\TestCase;
3
4 class CreateAccountTest extends TestCase
5 {
6     private $mockPdo;
7     private $usuario;
8
9     Tabnine | Edit | Test | Explain | Document
10    protected function setUp(): void
11    {
12        // Mock de PDO
13        $this->mockPdo = $this->getMockBuilder(PDO::class)
14            ->disableOriginalConstructor()
15            ->getMock();
16
17        // Crear la instancia de la clase Usuario
18        $this->usuario = new Usuario($this->mockPdo);
19    }
20
21    Tabnine | Edit | Test | Explain | Document
22    public function testAgregarUsuarioExitoso()
23    {
24        // Simulamos que las consultas a la base de datos se ejecutan correctamente
25        $stmt = $this->getMockBuilder(PreparedStatement::class)
26            ->expects($this->any())
27            ->method('execute')
28            ->willReturn(true);
29
30        // Simulamos que prepare() devuelve el PreparedStatement mockeado
31        $this->mockPdo->method('prepare')->willReturn($stmt);
32
33        // Llamar al método de agregar usuario
34        $resultado = $this->usuario->agregarUsuario('Juan', 'Perez', '123456789', 'juan@correo.com', 'contraseña123', '12345678', 'Calle Falsa 123', 'Distrito X');
35
36        // Verificar el resultado esperado
37        $this->assertEquals('success', $resultado['status']);
38        $this->assertEqual($resultado['resultado'], $resultado['message']);
39    }
}

```

```
39 public function testAgregarUsuarioError()
40 {
41     // Simulamos que alguna consulta falla
42     $stmt = $this->createMock(PDOStatement::class);
43     $stmt->expects($this->any())
44         ->method('execute')
45         ->willReturn(false);
46
47     // Simulamos que prepare() devuelve el PDOStatement mockeado
48     $this->mockPdo->method('prepare')->willReturn($stmt);
49
50     // Llamar al método de agregar usuario
51     $resultado = $this->usuario->agregarUsuario('Juan', 'Perez', '123456789', 'juan@correo.com', 'contraseña123', '12345678', 'Calle Falsa 123', 'Distrito X');
52
53     // Verificar el resultado esperado
54     $this->assertEquals('error', $resultado['status']);
55     $this->assertEquals('Error al agregar el usuario', $resultado['message']);
56 }
57 }
58 }
```

- **Resultado del testeо**

```
C:\xampp\htdocs\Crema-y-Mil-Hojas-Front-End (0.554s)
./vendor/bin/phpunit test/CreateAccountTest.php
PHPUnit 11.5.19 by Sebastian Bergmann and contributors.

Runtime:      PHP 8.2.12
.
.
.
Time: 00:00.184, Memory: 8.00 MB
OK (2 tests, 4 assertions)
```

De acuerdo con el resultado obtenido se detalla lo siguiente:

Qué en el testeо para evaluar la lógica de registro de usuarios se hacen 2 procesos, los cuales son registro exitoso y registro erróneo. Esto se separan en 2 funciones diferentes para el test, dando como resultado una prueba exitosa. Los cuales fueron:

Tests: 3 => Las 2 funciones

Assertions: 4 => La cantidad de afirmaciones validadas

❖ Pruebas unitarias de gestión de productos

Para estas pruebas unitarias se evaluaron las funciones de Agregar producto (agregarProducto()), Actualizar producto (actualizarProducto()) y Eliminar producto (eliminarProducto()). Dichas funciones pasaron las pruebas unitarias sin mayores complicaciones.

• Código de la lógica

```
<?php
class Producto
{
    Tabnine | Edit | Test | Explain | Document
    public static function agregarProducto(PDO $con, array $datos, array $archivo): array
    {
        $foto = '';
        $directorioDestino = "../Admin/images/";

        if (isset($archivo['imagenP']) && $archivo['imagenP']['error'] === UPLOAD_ERR_OK) {
            $fotoNombre = basename($archivo['imagenP'][name']);
            $rutaDestino = $directorioDestino . $fotoNombre;
            $rutaBD = "images/" . $fotoNombre;

            if (!is_dir($directorioDestino)) {
                mkdir($directorioDestino, 0777, true);
            }

            if (move_uploaded_file($archivo['imagenP']['tmp_name'], $rutaDestino)) {
                $foto = $rutaBD;
            }
        }

        $sql = $con->prepare("INSERT INTO tb_producto (Id_Categoría, Nombre_Producto, Descripción_Producto, Stock, Precio_Actual, Imagen)
                                VALUES (?, ?, ?, ?, ?, ?)");
        $sql->bindParam(1, $datos[categoría], PDO::PARAM_INT);
        $sql->bindParam(2, $datos[nombre], PDO::PARAM_STR);
        $sql->bindParam(3, $datos[descripción], PDO::PARAM_STR);
        $sql->bindParam(4, $datos[stock], PDO::PARAM_INT);
        $sql->bindParam(5, $datos[precio], PDO::PARAM_STR);
        $sql->bindParam(6, $foto, PDO::PARAM_STR);

        if ($sql->execute()) {
            return ["status" => "success", "message" => "Producto agregado correctamente"];
        } else {
            return ["status" => "error", "message" => "Error al agregar el producto"];
        }
    }
}
```

```

sistema | base | Help | Support | Documentos
public static function eliminarProducto(PDO $con, int $codigo): array
{
    $estado = 0;
    $sql = $con->prepare("UPDATE tb_producto SET Estado_Producto = ? WHERE Id_Producto = ?");
    $sql->bindParam(1, $estado, PDO::PARAM_INT);
    $sql->bindParam(2, $codigo, PDO::PARAM_INT);

    if ($sql->execute()) {
        return ["status" => "success", "message" => "Producto eliminado correctamente"];
    } else {
        return ["status" => "error", "message" => "Error al eliminar el producto"];
    }
}

Tabnine | Edit | Test | Explain | Document
public static function actualizarProducto(PDO $con, array $datos, array $archivos): array
{
    $directorioDestino = "../Admin/images/";
    $foto = '';
    $codigo = $datos['codigo'];
    $nombre = $datos['nombre'];
    $stock = $datos['stock'];
    $precio = $datos['precio'];
    $categoria = $datos['categoria'];
    $descripcion = $datos['descripcion'];
    $disponibilidad = $datos['disponibilidad'] === 'Disponible' ? 1 : 0;
    $estado = $datos['estado'] === 'Activo' ? 1 : 0;

    // Obtener imagen actual
    $sql_get_image = $con->prepare("SELECT Imagen FROM tb_producto WHERE Id_Producto = ?");
    $sql_get_image->execute([$codigo]);
    $producto = $sql_get_image->fetch(PDO::FETCH_ASSOC);
    $imagenActual = $producto['Imagen'];

    // Procesar nueva imagen
    if (!empty($archivos['imagen']['name']) && $archivos['imagen']['error'] === UPLOAD_ERR_OK) {
        $fotoNombre = basename($archivos['imagen']['name']);
        $rutaDestino = $directorioDestino . $fotoNombre;
        $rutaBD = 'Images/' . $fotoNombre;

        if (!is_dir($directorioDestino)) {
            mkdir($directorioDestino, 0777, true);
        }
    }
}

```

```

if (move_uploaded_file($archivos['imagen']['tmp_name'], $rutaDestino)) {
    if (!empty($imagenActual) && file_exists("../Admin/" . $imagenActual) && $imagenActual !== $rutaBD) {
        unlink("../Admin/" . $imagenActual);
    }
    $foto = $rutaBD;
} else {
    $foto = $imagenActual;
}
else {
    $foto = $imagenActual;
}

$sql_update = $con->prepare("UPDATE tb_producto
    SET Id_Categoría=?, Nombre_Producto=?, Descripción_Producto=?, Stock=?, Precio_Actual=?, Imagen=?, Disponibilidad=?, Estado_Producto=?
    WHERE Id_Producto=?");

$sql_update->bindParam(1, $categoria, PDO::PARAM_INT);
$sql_update->bindParam(2, $nombre, PDO::PARAM_STR);
$sql_update->bindParam(3, $descripcion, PDO::PARAM_STR);
$sql_update->bindParam(4, $stock, PDO::PARAM_INT);
$sql_update->bindParam(5, $precio, PDO::PARAM_STR);
$sql_update->bindParam(6, $foto, PDO::PARAM_STR);
$sql_update->bindParam(7, $disponibilidad, PDO::PARAM_INT);
$sql_update->bindParam(8, $estado, PDO::PARAM_INT);
$sql_update->bindParam(9, $codigo, PDO::PARAM_INT);

if ($sql_update->execute()) {
    return ["status" => "success", "message" => "Producto actualizado correctamente"];
} else {
    return ["status" => "error", "message" => "Error al actualizar el producto"];
}
}

```

- Código del testeo

```
<?php
use PHPUnit\Framework\TestCase;

require_once __DIR__ . '/../src/Producto.php';

class ProductoTest extends TestCase
{
    Tabnine | Edit | Test | Explain | Document
    public function testAgregarProductoExitosoSinImagen()
    {
        $stmtMock = $this->createMock(PDOStatement::class);
        $stmtMock->expects($this->once())
            ->method('execute')
            ->willReturn(true);

        $pdoMock = $this->createMock(PDO::class);
        $pdoMock->method('prepare')
            ->willReturn($stmtMock);

        $datos = [
            'categoria' => 1,
            'nombre' => 'Producto test',
            'descripcion' => 'Una descripción',
            'stock' => 20,
            'precio' => '50.00'
        ];
        $archivo = [];

        $result = Producto::agregarProducto($pdoMock, $datos, $archivo);

        $this->assertEquals('success', $result['status']);
        $this->assertEquals('Producto agregado correctamente', $result['message']);
    }
}
```

```
Tabnine | Edit | Test | Explain | Document
public function testAgregarProductoExitosoConImagen()
{
    $archivo = [
        'imagenP' => [
            'name' => 'producto.jpg',
            'tmp_name' => '/ruta/falsa/tmp/imagen.jpg',
            'error' => UPLOAD_ERR_OK
        ]
    ];

    $datos = [
        'categoria' => 2,
        'nombre' => 'Producto con imagen',
        'descripcion' => 'Descripción del producto con imagen',
        'stock' => 15,
        'precio' => '99.99'
    ];

    $stmtMock = $this->createMock(PDOStatement::class);
    $stmtMock->expects($this->once())
        ->method('execute')
        ->willReturn(true);

    $pdoMock = $this->createMock(PDO::class);
    $pdoMock->method('prepare')
        ->willReturn($stmtMock);

    $this->mockGlobalFunctions();

    $result = Producto::agregarProducto($pdoMock, $datos, $archivo);

    $this->assertEquals('success', $result['status']);
    $this->assertEquals('Producto agregado correctamente', $result['message']);
}
```

```
Tabnine | Edit | Test | Explain | Document
protected function mockGlobalFunctions()
{
    require_once __DIR__ . '/..src/Producto.php';

    if (!function_exists('is_dir')) {
        function is_dir($path) { return true; }
    }

    if (!function_exists('mkdir')) {
        function mkdir($path, $mode, $recursive) { return true; }
    }

    if (!function_exists('move_uploaded_file')) {
        function move_uploaded_file($from, $to) { return true; }
    }
}

Tabnine | Edit | Test | Explain | Document
public function testEliminarProductoExitoso()
{
    $stmtMock = $this->createMock(PDOStatement::class);
    $stmtMock->expects($this->once())
        ->method('execute')
        ->willReturn(true);

    $pdoMock = $this->createMock(PDO::class);
    $pdoMock->method('prepare')
        ->willReturn($stmtMock);

    $result = Producto::eliminarProducto($pdoMock, 123);

    $this->assertEquals('success', $result['status']);
    $this->assertEquals('Producto eliminado correctamente', $result['message']);
}
```

```
Tabnine | Edit | Test | Explain | Document
public function testActualizarProductoExitosoSinImagen()
{
    $datos = [
        'codigo' => 1,
        'nombre' => 'Producto Actualizado',
        'stock' => 10,
        'precio' => '29.99',
        'categoria' => 2,
        'descripcion' => 'Descripción actualizada',
        'disponibilidad' => 'Disponible',
        'estado' => 'Activo'
    ];

    $archivos = [ 'imagen' => [ 'name' => '', 'error' => UPLOAD_ERR_NO_FILE ] ];

    $stmtMockGet = $this->createMock(PDOStatement::class);
    $stmtMockGet->method('fetch')->willReturn(['Imagen' => 'images/actual.jpg']);

    $stmtMockUpdate = $this->createMock(PDOStatement::class);
    $stmtMockUpdate->expects($this->once())->method('execute')->willReturn(true);

    $pdoMock = $this->createMock(PDO::class);
    $pdoMock->method('prepare')->willReturnCallback(function($sql) use ($stmtMockGet, $stmtMockUpdate) {
        return str_contains($sql, 'SELECT') ? $stmtMockGet : $stmtMockUpdate;
    });

    $result = Producto::actualizarProducto($pdoMock, $datos, $archivos);

    $this->assertEquals('success', $result['status']);
    $this->assertEquals('Producto actualizado correctamente', $result['message']);
}
```

- **Resultado del testeo**

```
C:\xampp\htdocs\Cremas-y-Mil-Hojas-Front-End (1.211s)
./vendor/bin/phpunit test/productoTest.php
PHPUnit 11.5.20 by Sebastian Bergmann and contributors.

Runtime:      PHP 8.2.12
.
.
.
Time: 00:00.208, Memory: 8.00 MB
OK (4 tests, 12 assertions)
```

De acuerdo con los resultados, las pruebas unitarias realizadas a las funciones mencionadas fueron pasadas de manera satisfactoria.

- ❖ **Pruebas unitarias de carrito de compras**

En las pruebas unitarias para el carrito de compras, se tuvo que utilizar la librería Jest, ya que la lógica del carrito de compras está hecha en javascript, dentro del archivo cart.js se evaluaron funciones como Agregar producto al carrito (addToCart()), Eliminar producto del carrito (removeItem()) y Actualizar la cantidad del producto en el carrito (updateQuantity()). Para ello se tuvo que crear el archivo cart.test.js para evaluar dichas funciones.

- Código de la lógica

```

1 Tabnine | Edit | Test | Explain | Document
2 function removeItem(index) {
3   let cart = getCart();
4   if (index >= 0 && index < cart.length) {
5     cart.splice(index, 1);
6     saveCart(cart);
7   }
8 }
9 Tabnine | Edit | Test | Explain | Document
10 function updateQuantity(index, change) {
11   let cart = getCart();
12   if (index >= 0 && index < cart.length) {
13     const product = cart[index];
14     const newQuantity = Math.max(1, (product.quantity || 1) + change);
15     product.quantity = newQuantity;
16     saveCart(cart);
17   }
18 }
19 Tabnine | Edit | Test | Explain | Document
20 function addToCart(buttonDataset) {
21   const product = {
22     slug: buttonDataset.slug,
23     title: buttonDataset.title,
24     img: buttonDataset.img,
25     price: parseFloat(buttonDataset.price) || 0,
26     quantity: 1
27   };
28   if (product.slug.includes("(") || product.title.includes("(")) {
29     console.error("X Error: Los datos del producto no se están interpolando correctamente.");
30     return;
31   }
32   let cart = getCart();
33   cart.push(product);
34   saveCart(cart);
35 }
36 module.exports = [
37   removeItem,
38   updateQuantity,
39   addToCart
40 ];
41
42

```

- Código de testeo

```

1 beforeEach(() => {
2   global.localStorage = {
3     store: {},
4     getItem(key) {
5       return this.store[key] || null;
6     },
7     setItem(key, value) {
8       this.store[key] = value;
9     },
10    removeItem(key) {
11      delete this.store[key];
12    },
13    clear() {
14      this.store = {};
15    }
16  };
17 });
18
19 const {
20   addToCart,
21   removeItem,
22   updateQuantity,
23 } = require('../src/js/cart');
24 Tabnine | Edit | Test | Explain | Document
25 beforeEach(() => {
26   localStorage.clear();
27 });
28 Tabnine | Edit | Test | Explain | Document
29 test('addToCart añade un producto correctamente', () => {
30   const product = {
31     slug: 'test-product',
32     title: 'Test Product',
33     img: 'test.jpg',
34     price: '50'
35   };
36   addToCart(product);
37
38   const cart = getCart();
39   expect(cart.length).toBe(1);
40   expect(cart[0].slug).toBe('test-product');
41 });
42

```

```

43 Tabnine | Edit | Test | Explain | Document
44 test('removeItem elimina el producto por indice', () => {
45   const product = {
46     slug: 'item-1',
47     title: 'Item 1',
48     img: '1.jpg',
49     price: 10,
50     quantity: 1
51   };
52
53   localStorage.setItem('cart', JSON.stringify([product]));
54   removeItem(0);
55
56   const cart = getCart();
57   expect(cart.length).toBe(0);
58 });
59 Tabnine | Edit | Test | Explain | Document
60 test('updateQuantity aumenta y disminuye la cantidad correctamente', () => {
61   const product = {
62     slug: 'item-1',
63     title: 'Item 1',
64     img: '1.jpg',
65     price: 10,
66     quantity: 2
67   };
68
69   localStorage.setItem('cart', JSON.stringify([product]));
70
71   updateQuantity(0, 1);
72   let cart = getCart();
73   expect(cart[0].quantity).toBe(3);
74
75   updateQuantity(0, -2);
76   cart = getCart();
77   expect(cart[0].quantity).toBe(1);
78 });
79

```

- **Resultado del testeо**

```

npm test cart.test.js

> cremas-y-mil-hojas-front-end@1.0.0 test
> jest cart.test.js

(node:9560) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
FAIL test/cart.test.js
  ✕ addToCart añade un producto correctamente (2 ms)
  ✕ removeItem elimina el producto por índice (2 ms)
  ✕ updateQuantity aumenta y disminuye la cantidad correctamente (1 ms)

● addToCart añade un producto correctamente
getCart
  ReferenceError: getCart is not defined

    39 |
    40 |       }
    41 |       let cart = getCart();
    42 |       cart.push(product);
    43 |       saveCart(cart);
    44 |     }

  at getCart (srcjs/cart.js:41:16)
  at Object.addToCart (test/cart.test.js:37:5)

```

```
ReferenceError: getCart is not defined

  8 |
  9 |     function removeItem(index) {
> 10|       let cart = getCart();
   |
 11|       if (index >= 0 && index < cart.length) {
 12|         cart.splice(index, 1);
 13|         saveCart(cart);

at getCart (srcjs/cart.js:10:16)
at Object.removeItem (test/cart.test.js:54:5)

● updateQuantity aumenta y disminuye la cantidad correctamente

ReferenceError: getCart is not defined

 16 |
 17 |     function updateQuantity(index, change) {
> 18|       let cart = getCart();
   |
 19|       if (index >= 0 && index < cart.length) {
 20|         const product = cart[index];
 21|         const newQuantity = Math.max(1, (product.quantity || 1) + change);

at getCart (srcjs/cart.js:18:16)
at Object.updateQuantity (test/cart.test.js:71:5)

Test Suites: 1 failed, 1 total
Tests:       3 failed, 3 total
Snapshots:  0 total
Time:        3.261 s
```

Se ha detectado 3 errores en total, en las líneas 41 ,10 y 18. Los cuales son debido a un error de referencia, ya que la función getCart() no está definida.

Anexo 5. Depuración del software para pasar con éxito las pruebas unitarias

❖ Depuración código de carrito de compras

Para depuración se tuvo que identificar, analizar y corregir los errores en el código del archivo cart.js. Estos errores se dieron debido a que la función getCart() no estaba definida. Estos errores se arreglando, definiendo la función y a su vez definiendo la función saveCart() para evitar otros posibles errores al momento de realizar la prueba unitaria.

• Código depurado

```
Tabnine | Edit | Test | Explain | Document
1 function getCart() {
2   return JSON.parse(localStorage.getItem('cart')) || [];
3 }
4
5 Tabnine | Edit | Test | Explain | Document
6 function saveCart(cart) {
7   localStorage.setItem('cart', JSON.stringify(cart));
8 }
9
10 Tabnine | Edit | Test | Explain | Document
11 function removeItem(index) {
12   let cart = getCart();
13   if (index >= 0 && index < cart.length) {
14     cart.splice(index, 1);
15   }
16   saveCart(cart);
17 }
18
19 Tabnine | Edit | Test | Explain | Document
20 function updateQuantity(index, change) {
21   let cart = getCart();
22   if (index >= 0 && index < cart.length) {
23     const product = cart[index];
24     const newQuantity = Math.max(1, (product.quantity || 1) + change);
25     product.quantity = newQuantity;
26     saveCart(cart);
27   }
28 }
```

```
Tabnine | Edit | Test | Explain | Document
27 function addToCart(buttonDataset) {
28   const product = {
29     slug: buttonDataset.slug,
30     title: buttonDataset.title,
31     img: buttonDataset.img,
32     price: parseFloat(buttonDataset.price) || 0,
33     quantity: 1
34   };
35
36   if (product.slug.includes("{}") || product.title.includes("{}")) {
37     console.error(`✖ Error: Los datos del producto no se están interpolando correctamente.`);
38     return;
39   }
40
41   let cart = getCart();
42   cart.push(product);
43   saveCart(cart);
44 }
45
46 module.exports = {
47   getCart,
48   saveCart,
49   removeItem,
50   updateQuantity,
51   addToCart
52 };
```

- Código de testeo

```

Tabnine | Edit | Test | Explain | Document
1 beforeAll(() => {
2   global.localStorage = {
3     store: {},
4     getItem(key) {
5       return this.store[key] || null;
6     },
7    .setItem(key, value) {
8       this.store[key] = value;
9     },
10    removeItem(key) {
11      delete this.store[key];
12    },
13    clear() {
14      this.store = {};
15    }
16  };
17 });
18
19 const {
20   addToCart,
21   removeItem,
22   updateQuantity,
23   getCart
24 } = require('../srcjs/cart');
25 Tabnine | Edit | Test | Explain | Document
26 beforeEach(() => {
27   localStorage.clear();
28 });
29
30 test('`addToCart` añade un producto correctamente', () => {
31   const product = {
32     slug: 'test-product',
33     title: 'Test Product',
34     img: 'test.jpg',
35     price: '50'
36   };
37   addToCart(product);
38
39   const cart = getCart();
40   expect(cart.length).toBe(1);
41   expect(cart[0].slug).toBe('test-product');
42 });
43

```

```

Tabnine | Edit | Test | Explain | Document
44 test('`removeItem` elimina el producto por indice', () => {
45   const product = {
46     slug: 'item-1',
47     title: 'Item 1',
48     img: '1.jpg',
49     price: 10,
50     quantity: 1
51   };
52
53   localStorage.setItem('cart', JSON.stringify([product]));
54   removeItem(0);
55
56   const cart = getCart();
57   expect(cart.length).toBe(0);
58 });
59
60
61 Tabnine | Edit | Test | Explain | Document
62 test('`updateQuantity` aumenta y disminuye la cantidad correctamente', () => {
63   const product = {
64     slug: 'item-1',
65     title: 'Item 1',
66     img: '1.jpg',
67     price: 10,
68     quantity: 2
69   };
70
71   localStorage.setItem('cart', JSON.stringify([product]));
72
73   updateQuantity(0, 1); // debe pasar a 3
74   let cart = getCart();
75   expect(cart[0].quantity).toBe(3);
76
77   updateQuantity(0, -2); // debe bajar a 1 (mínimo permitido)
78   cart = getCart();
79   expect(cart[0].quantity).toBe(1);
80 });
81

```

- **Resultado del testeo (segunda prueba unitaria)**

```
C:\xampp\htdocs\Cremas-y-Mil-Hojas-Front-End (4.717s)
npm test cart.test.js

> cremas-y-mil-hojas-front-end@1.0.0 test
> jest cart.test.js

(node:18972) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please
use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
PASS  test/cart.test.js
  ✓ addToCart añade un producto correctamente (3 ms)
  ✓ removeItem elimina el producto por indice (1 ms)
  ✓ updateQuantity aumenta y disminuye la cantidad correctamente (1 ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:  0 total
Time:        2.555 s
Ran all test suites matching /cart.test.js/i.
```

Y con este último resultado se ve que las pruebas unitarias aplicadas a las funciones addToCart(), removeItem() y updateQuantity () pasaron de manera exitosa definiendo las funciones getCart() y saveCart().

Anexo 6. Diseño de casos de prueba para evaluar la funcionalidad del Software

Con el objetivo de garantizar la calidad y correcto funcionamiento del software desarrollado, se llevaron a cabo pruebas unitarias y funcionales para cada uno de los módulos implementados.

Para evaluar de forma más completa el avance del desarrollo, se elaboraron tablas de casos de prueba por cada Sprint, permitiendo una verificación sistemática de los requerimientos funcionales establecidos. Estas tablas contienen los criterios de prueba, los resultados esperados, los resultados obtenidos y el estado final de cada caso (aprobado o no aprobado).

❖ Matriz de casos de pruebas del control de inventario

Nº	Nº de Historia de Usuario relacionada	Nombre del Caso de Prueba	Pre-Condición	Pasos de Ejecución	Resultados Esperados	Estado (Pasó / Falló)
4	H4	Registrar productos	Como administrador, quiero registrar, editar y eliminar productos en el inventario, para mantener actualizada la información del inventario.	8. Clic en la interfaz Productos. 9. Clic en el botón Agregar Producto. 10. Rellenar los campos "nombre", "stock", "precio", "imagen", "categoria", "descripcion" y clic en el botón Guardar. 11. Para editar el producto; clic en el botón Editar. 12. Rellenar los campos "nombre", "stock", "precio", "imagen", "categoria", "descripcion". 13. Clic en el botón Guardar.	El administrador podrá registrar, eliminar y editar el producto	Pasó

				14. Para eliminar el producto; clic en el botón Eliminar.		
5	H5	Actualización automática	Como administrador, quiero que el inventario se actualice automáticamente cuando se hace un pedido, para reflejar el stock real.	3. Clic en la interfaz Productos. 4. Verificar si cada producto muestra su estado o disponibilidad (Activo/Inactivo, Disponible/No disponible). 5. Cuando el cliente confirma su pedido, se descontará la cantidad correspondiente del stock del producto.	El administrador podrá visualizar si el stock del producto disminuye automáticamente tras cada pedido	No Pasó
6	H6	Alerta de stock	Como administrador, quiero recibir una alerta cuando un producto tenga poco stock, para evitar quedarme sin productos clave.	3. Clic en la interfaz Productos. 4. El sistema muestra una alerta visual “Algunos productos están escasos o agotados. Verifica el inventario.” cuando el stock es menor al mínimo definido.	El administrador podrá visualizar la alerta que el sistema genera cuando un producto tiene stock bajo	Pasó

❖ Matriz de casos de pruebas de la gestión de pedidos

Nº	Nº de Historia de Usuario relacionada	Nombre del Caso de Prueba	Pre-Condición	Pasos de Ejecución	Resultados Esperados	Estado (Pasó / Falló)
1	H1	Realizar pedido	Como cliente, quiero realizar un pedido de productos en línea, para evitar demoras y filas presenciales.	<ol style="list-style-type: none"> 1. Clic en menú "Productos". 2. Visualizará todos los productos. 3. Seleccionas el producto y podrás ver el nombre y descripción del producto. 4. Clic en el botón Agregar al carrito. 5. Clic en el apartado "Carrito". 6. Clic en el +/- para agregar o quitar cantidad del producto. 7. Clic en el botón "Seguir comprando" para seguir agregando productos al carrito. 	El administrador podrá realizar pedidos en línea satisfactoriamente.	Pasó
2	H2	Ver disponibilidad	Como cliente, quiero ver qué productos están disponibles antes de hacer un pedido, para evitar frustraciones por falta de stock.	<ol style="list-style-type: none"> 1. Clic en menú "Productos". 2. Podrá visualizar el catálogo de productos en stock. 3. En cada producto aparece "Cantidad" el cual mostrará la disponibilidad del producto. 4. No se dejará seleccionar el producto que esté agotado. 5. Y cuando se seleccione el producto, la cantidad disponible se actualizará en tiempo real. 	El cliente podrá visualizar si el producto posee stock o está agotado.	No Pasó

3	H3	Estado del pedido	Como administrador, quiero visualizar y actualizar el estado de cada pedido, para controlar el proceso de entrega.	1. Click en la interfaz Pedidos. 2. Se visualizará la tabla de pedidos y se mostrarán sus estados. 3. Para actualizar el estado de cada pedido; aplastar el botón “Editar”. 4. Editar el campo “Estado”. 5. Clic en el botón guardar.	El administrador podrá visualizar y actualizar el estado del pedido.	Pasó
---	----	-------------------	--	---	--	------

❖ Matriz de casos de pruebas del login y registro

Nº	Nº de Historia de Usuario relacionada	Nombre del Caso de Prueba	Pre-Condición	Pasos de Ejecución	Resultados Esperados	Estado (Pasó / Falló)
1	H12	Registro de cuenta cliente	Como cliente, quiero poder crear una cuenta definiendo mis datos, para realizar mis pedidos.	4. Clic en el hipervínculo "Crear cuenta". 5. Completar todos los campos "nombre", "apellido", "teléfono", "correo electrónico", "contraseña", "dni", "dirección", "distrito". 6. Clic en el botón Registrar.	El usuario podrá registrarse en el sistema.	Pasó
2	H13	Inicio de sesión cliente	Como cliente, quiero iniciar sesión con mis credenciales, para acceder de manera segura al sistema y realizar mis pedidos.	4. Clic en el icono de usuario. 5. Completar el campo de "correo" y "contraseña". 6. Clic en el botón Ingresar.	El usuario podrá iniciar sesión en el sistema.	Pasó
3	H14	Inicio de sesión Administrador	Como administrador, quiero iniciar sesión con mis credenciales, para acceder de manera segura al sistema.	4. Clic en el icono de usuario. 5. Completar el campo de "correo" y "contraseña" establecidas para el administrador. 6. Clic en el botón Ingresar.	El administrador podrá ingresar en el sistema.	Pasó

Anexo 7. Ejecución de los casos de pruebas y depurar el software para pasar con éxito las pruebas funcionales

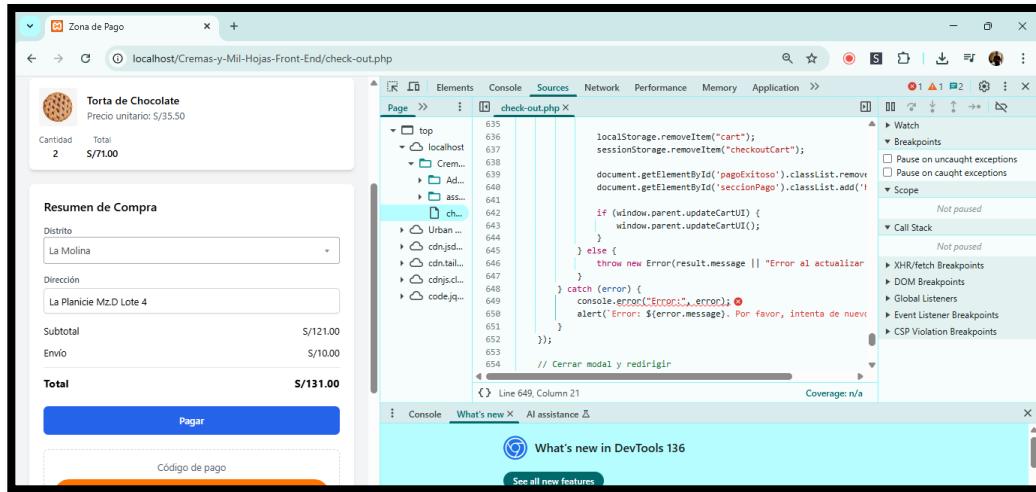
❖ Control de inventario (Actualización automática)

- **Caso de prueba:** El administrador asegura que el inventario se actualice automáticamente cuando se hace un pedido, para reflejar el stock real
- **Resultado esperado:** El sistema descuenta automáticamente el stock del producto al realizar el pago, reflejando el stock real en el panel del administrador.
- **Resultado real:** El sistema muestra un error porque intenta insertar un dato en la columna "distrito", la cual no existe en la tabla tb_pedido.
- **Acción:** Se depura el código y se corrige la columna "distrito" por "id_distrito". Luego de esto, al presionar "Pagado", el stock se actualiza correctamente en el panel del administrador.

Antes:

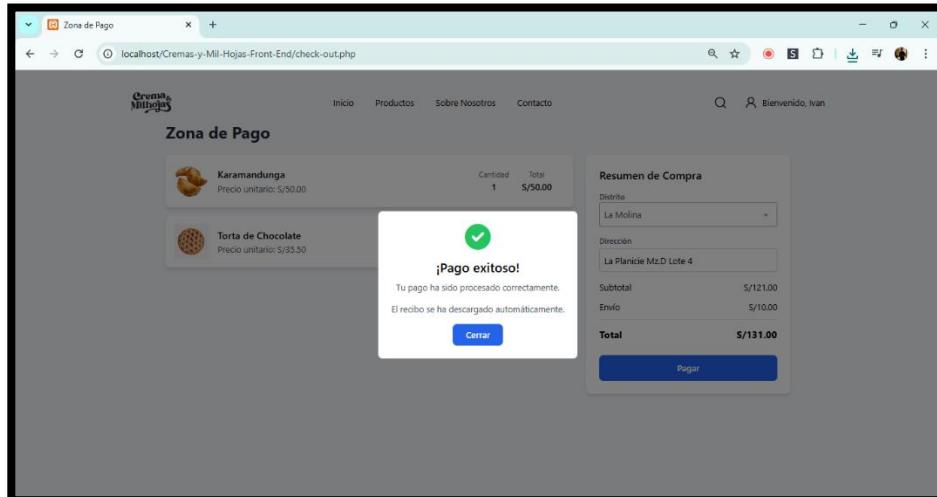
The screenshot shows a web browser window titled 'Admin-Productos' with the URL 'localhost/Cremas-y-Mil-Hojas-Front-End/Admin/productos-admin.php'. The page has a sidebar on the left with links: Dashboard, **Productos** (which is selected), Pedidos, Clientes, and Repetidores. The main content area is titled 'Listado de Productos' and contains a table of products. The table has columns: CÓDIGO, IMAGEN, CATEGORÍA, NOMBRE, DESCRIPCIÓN, STOCK, PRECIO, DISPONIBILIDAD, ESTADO, and ACCIONES. There are two rows of data:

CÓDIGO	IMAGEN	CATEGORÍA	NOMBRE	DESCRIPCIÓN	STOCK	PRECIO	DISPONIBILIDAD	ESTADO	ACCIONES
1		Bollería	Karamandunga	dulce crocante	3	S/ 50.00	Disponible	Activo	<button>Edit</button> <button>Delete</button>
2		Especiales	Torta de Chocolate	Deliciosa torta de chocolate con crema	3	S/ 35.50	Escasas	Activo	<button>Edit</button> <button>Delete</button>



```
17
18 $carrito = $data['carrito'];
19 $distrito = $data['distrito'];
20 $direccion = $data['direccion'];
21 //Total = $data['preciototal'];
22 $cliente_id = $_SESSION['id']; // Asegúrate que la sesión está activa y tiene este valor
23
24 try {
25     $con->beginTransaction();
26
27     // 1. Insertar pedido
28     $sql_pedido = "INSERT INTO tb_pedido (Id_Cliente, distrito, Direccion_Entrega)
29     | VALUES (?, ?, ?)";
30     $stmt_pedido = $con->prepare($sql_pedido);
31     $stmt_pedido->execute([$cliente_id, $distrito, $direccion]);
32
33     $pedido_id = $con->lastInsertId(); // Obtener el ID del pedido recién insertado
34
35     $productos_actualizados = [];
36 }
```

Después:



The screenshot shows a web browser window for 'localhost/Crema-y-Mil-Hojas-Front-End/catalogo.php'. The page has a header with the logo 'Crema & Milhojas' and navigation links for 'Inicio', 'Productos', 'Sobre Nosotros', and 'Contacto'. A user 'Bienvenido, Ivan' is logged in. On the left, there are three filter sections: 'Categoria' (Bollería, Dulces, Especiales, Pan artesanal, Pan saludable), 'Precios' (Menos de S/ 5, S/ 5 - S/ 10, S/ 10 - S/ 25, Más de S/ 25), and 'Porciones' (Individual, Mediano, Familiar). The main content area displays two products: 'Karamandunga' (Bollería) at S/ 50.00 with quantity 2, and 'Torta de Chocolate' (Especiales) at S/ 35.50 with quantity 1. Each product has an 'AGREGAR' button. A shopping cart icon in the top right corner shows 0 items and S/ 0.00.

The screenshot shows a web browser window for 'localhost/Crema-y-Mil-Hojas-Front-End/Admin/productos-admin.php'. The sidebar on the left includes 'Dashboard', 'Productos' (selected), 'Pedidos', 'Clientes', and 'Repartidores'. The main content is titled 'Listado de Productos' with a search bar for 'Buscar por Código' and 'Buscar por Nombre'. It features three filtering sections: 'Filtrar por Categoría', 'Filtrar por Disponibilidad', and 'Filtrar por Estado'. Below is a table with columns: CÓDIGO, IMAGEN, CATEGORÍA, NOMBRE, DESCRIPCIÓN, STOCK, PRECIO, DISPONIBILIDAD, ESTADO, and ACCIONES. Two products are listed: '1 Karamandunga' (Bollería) and '2 Torta de Chocolate' (Especiales). Each row has 'Editar' and 'Eliminar' buttons. A blue '+ Agregar Producto' button is located at the top right of the table.

```

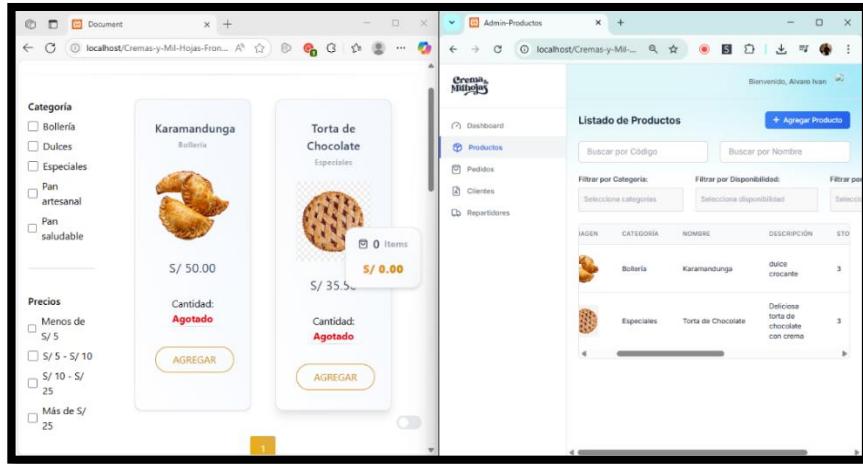
23
24 try {
25     $con->beginTransaction();
26
27 // 1. Insertar pedido
28 $sql_pedido = "INSERT INTO tb_pedido (Id_Cliente, Id_Distrito, Direccion_Entrega) |
29             VALUES (?, ?, ?)";
30 $stmt_pedido = $con->prepare($sql_pedido);
31 $stmt_pedido->execute([$cliente_id, $distrito, $direccion]);
32
33 $pedido_id = $con->lastInsertId(); // Obtener el ID del pedido recién insertado
34
35 $productos_actualizados = [];
36
37 $total_pedido2 = 0.00;
38

```

❖ Gestión de pedidos (Ver disponibilidad)

- **Caso de prueba:** El cliente visualiza los productos que están disponibles antes de hacer un pedido, para evitar frustraciones por falta de stock.
- **Resultado esperado:** Si hay stock disponible en la base de datos, el cliente ve el producto como "disponible" y puede agregarlo al carrito.
- **Resultado real:** El cliente visualiza el producto como "agotado" aunque hay stock disponible en el panel del administrador.
- **Acción:** Se depura el código y se corrige un error de sintaxis en la línea 56 del código, cambiando la condición de `product.stock < 0` a `product.stock > 0`. Con esto, ahora se refleja correctamente la disponibilidad del producto.

Antes:



Después:

Categoría

- Bollería
- Dulces
- Especiales
- Pan artesanal
- Pan saludable

Precios

- Menos de S/ 5
- S/ 5 - S/ 10
- S/ 10 - S/ 25
- Más de S/ 25



Karamandunga
Bollería

S/ 50.00

Cantidad: 3

[AGREGAR](#)



Torta de Chocolate
Especiales

S/ 35.50

Cantidad: 3

[AGREGAR](#)

Bienvenido, Alvaro Ivan

Dashboard

Productos

Pedidos

Clientes

Repartidores

Listado de Productos

+ Agregar Producto

Buscar por Código Buscar por Nombre

Filtrar por Categoría: Selecciona categorías

Filtrar por Disponibilidad: Selecciona disponibilidad

IMÁGEN	CATEGORÍA	NOMBRE	DESCRIPCIÓN	STO
	Bollería	Karamandunga	dulce crocante	3
	Especiales	Torta de Chocolate	Deliciosa torta de chocolate con crema	3

The screenshot displays two browser windows side-by-side.

Left Window (Product Catalog):

- Categoría:**
 - Bollería
 - Dulces
 - Especiales
 - Pan artesanal
 - Pan saludable
- Karamandunga** (Bollería)
Image: A golden-brown croissant.
Price: S/ 50.00
Stock: 0 Items (Red text)
Action: AGREGAR (Yellow button)
- Torta de Chocolate** (Especiales)
Image: A chocolate cake with a crisscross pattern.
Price: S/ 35.50
Stock: 3
Action: AGREGAR (Yellow button)

Right Window (Management Interface):

- Admin-Productos**
- Crema y Milanesas**
- Dashboard**
- Products** (Selected)
- Pedidos**
- Clientes**
- Repartidores**

Listado de Productos

Imagen	Categoría	Nombre	Descripción	Stock
	Bollería	Karamandunga	dulce crocante	0
	Especiales	Torta de Chocolate	Deliciosa torta de chocolate con crema	3

```
51 >   
52 </div>
53 <p class="text-gray-700 text-xl tracking-wide">$ ${product.Precio_Actual}</p>
54 <div class="py-8">
55   <span>Cantidad: ${product.Stock} > 0 ? product.Stock : <span class="text-red-600 font-bold">Agotado</span></span>
56   <div class="top-0 left-0 w-full h-[1px] bg-gradient-to-r from-transparent from-10% via-gray-300 to-transparent to-90%"></div>
57 </div>
58 <button
59   class="px-7 py-2 border rounded-full transition-all duration-300 ease-in-out ${
60     product.Stock > 0
61       ? 'border-yellow-600 text-yellow-600 hover:bg-transparent hover:border-yellow-700/80 hover:bg-gradient-to-r hover:from-yellow-600 hover:to-yellow-700/80 hover:bg-:
62       : 'border-gray-300 text-gray-400 cursor-not-allowed bg-gray-100"
63   "}>
64 </button>
65 </div>
```

Anexo 8. Inspección del software

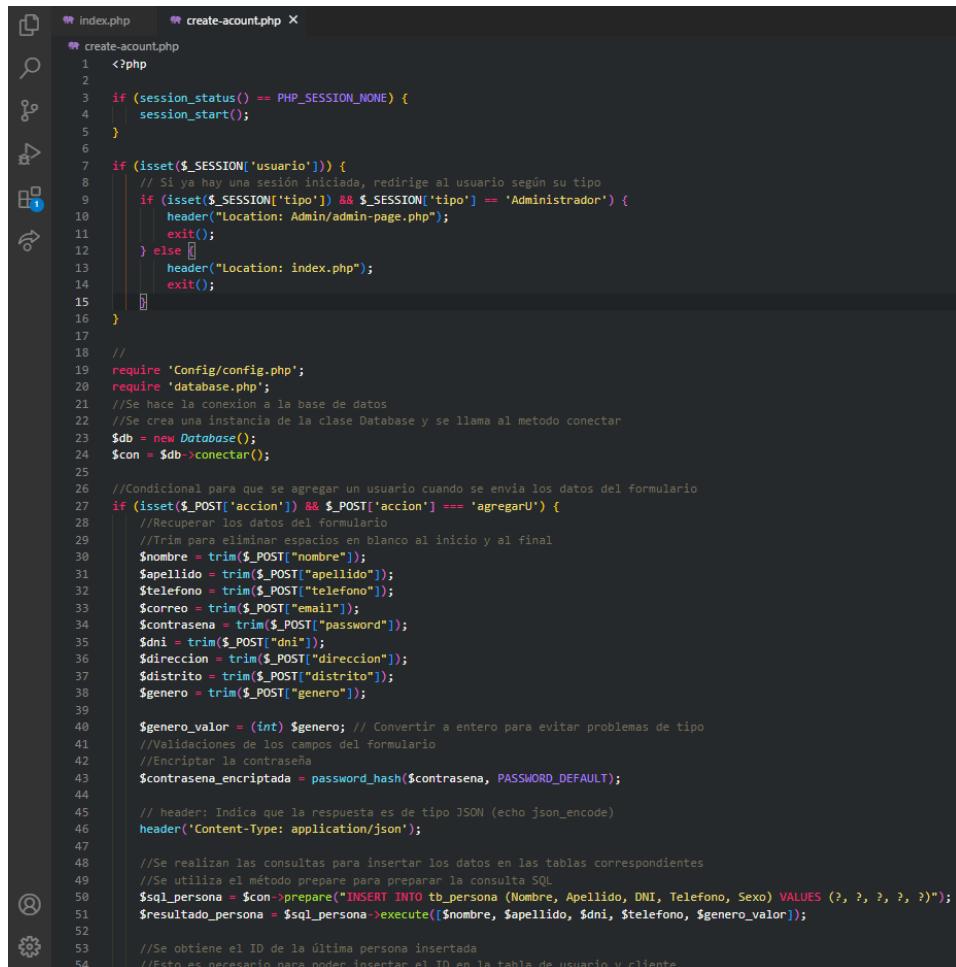
En este apartado se realizó la inspección total del software de nuestro proyecto.

A continuación, se mostrarán los códigos y las capturas de estos mismos ejecutados para demostrar que los requerimientos de los módulos del caso de uso principal se estén cumpliendo.

❖ Inspección del módulo Login y Registro

- R.F 12: El sistema debe contar con un apartado donde se pueda registrar cuentas de tipo cliente.

✓ Código:



The screenshot shows a code editor with a dark theme. The file being viewed is 'create-account.php'. The code is as follows:

```
<?php
if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

if (isset($_SESSION['usuario'])) {
    // Si ya hay una sesión iniciada, redirige al usuario según su tipo
    if ($_SESSION['tipo'] && $_SESSION['tipo'] == 'Administrador') {
        header("Location: Admin/admin-page.php");
        exit();
    } else {
        header("Location: index.php");
        exit();
    }
}

require 'Config/config.php';
require 'database.php';
//Se hace la conexión a la base de datos
//Se crea una instancia de la clase Database y se llama al método conectar
$db = new Database();
$conn = $db->conectar();

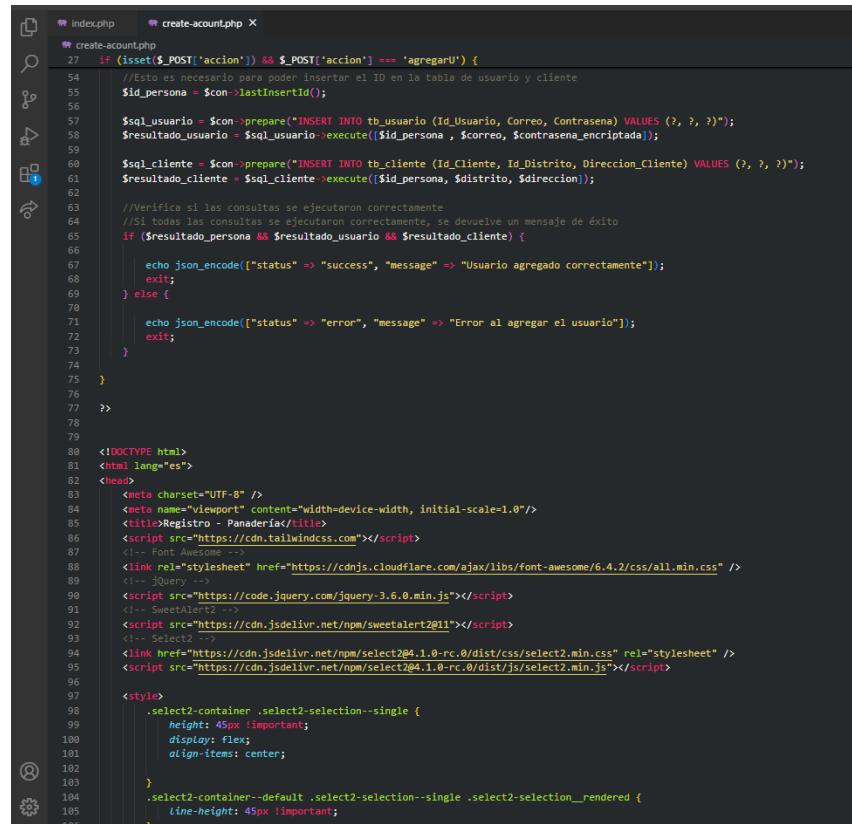
//Condicional para que se agregan un usuario cuando se envía los datos del formulario
if (isset($_POST['accion']) && $_POST['accion'] === 'agregarU') {
    //Recuperar los datos del formulario
    //Trim para eliminar espacios en blanco al inicio y al final
    $nombre = trim($_POST['nombre']);
    $apellido = trim($_POST['apellido']);
    $telefono = trim($_POST['telefono']);
    $correo = trim($_POST['email']);
    $contraseña = trim($_POST['password']);
    $dni = trim($_POST['dni']);
    $dirección = trim($_POST['direccion']);
    $distrito = trim($_POST['distrito']);
    $genero = trim($_POST['genero']);

    $genero_valor = (int) $genero; // Convertir a entero para evitar problemas de tipo
    //Validaciones de los campos del formulario
    //Encriptar la contraseña
    $contraseña_encriptada = password_hash($contraseña, PASSWORD_DEFAULT);

    // header: Indica que la respuesta es de tipo JSON (echo json_encode)
    header('Content-Type: application/json');

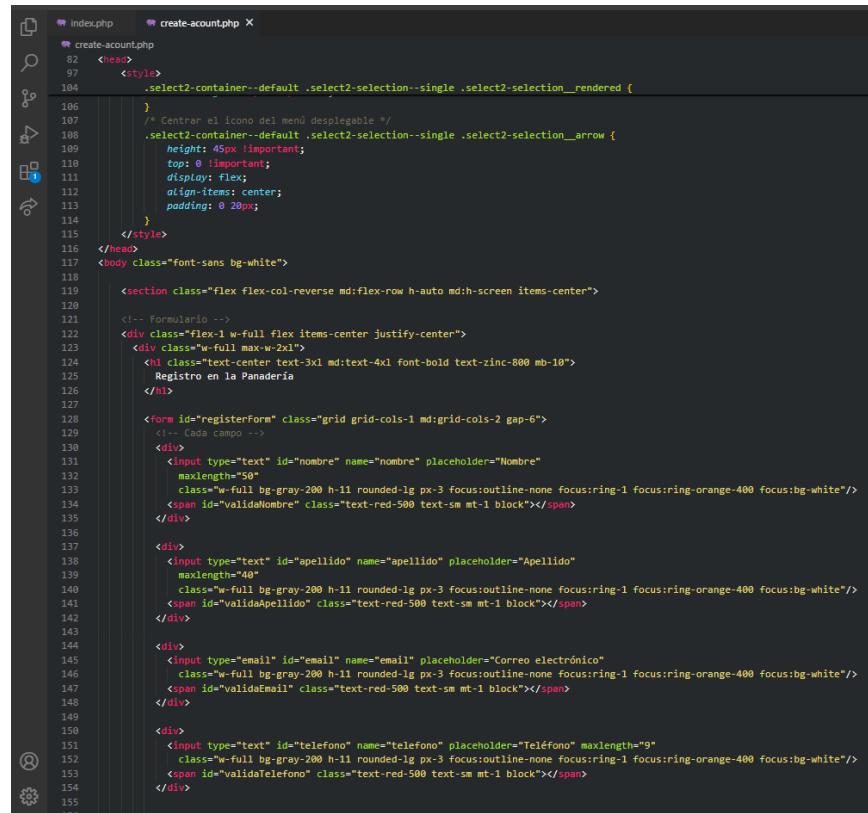
    //Se realizan las consultas para insertar los datos en las tablas correspondientes
    //Se utiliza el método prepare para preparar la consulta SQL
    $sql_persona = $conn->prepare("INSERT INTO tb_persona (Nombre, Apellido, DNI, Telefono, Sexo) VALUES (?, ?, ?, ?, ?)");
    $resultado_persona = $sql_persona->execute([$nombre, $apellido, $dni, $telefono, $genero_valor]);

    //Se obtiene el ID de la última persona insertada
    //Este es necesario para poder insertar el ID en la tabla de usuario v_cliente
}
```



```

27 if (isset($_POST['accion']) && $_POST['accion'] === 'agregarU') {
28     //Esto es necesario para poder insertar el ID en la tabla de usuario y cliente
29     $id_persona = $con->lastInsertId();
30
31     $sql_usuario = $con->prepare("INSERT INTO tb_usuario (Id_Usuario, Correo, Contrasena) VALUES (?, ?, ?)");
32     $resultado_usuario = $sql_usuario->execute([$id_persona, $correo, $contrasena_encriptada]);
33
34     $sql_cliente = $con->prepare("INSERT INTO tb_cliente (Id_Cliente, Id_Distrito, Direccion_Cliente) VALUES (?, ?, ?)");
35     $resultado_cliente = $sql_cliente->execute([$id_persona, $distrito, $direccion]);
36
37     //Verifica si las consultas se ejecutaron correctamente
38     //Si todas las consultas se ejecutaron correctamente, se devuelve un mensaje de éxito
39     if ($resultado_persona && $resultado_usuario && $resultado_cliente) {
40
41         echo json_encode(["status" => "success", "message" => "Usuario agregado correctamente"]);
42         exit;
43     } else {
44
45         echo json_encode(["status" => "error", "message" => "Error al agregar el usuario"]);
46         exit;
47     }
48 }
49
50 <!DOCTYPE html>
51 <html lang="es">
52 <head>
53     <meta charset="UTF-8" />
54     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
55     <title>Registro - Panadería</title>
56     <script src="https://cdn.tailwindcss.com"></script>
57     <!-- Font Awesome -->
58     <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.2/css/all.min.css" />
59     <!-- jQuery -->
60     <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
61     <!-- SweetAlert2 -->
62     <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
63     <!-- Select2 -->
64     <link href="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/css/select2.min.css" rel="stylesheet" />
65     <script src="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/js/select2.min.js"></script>
66
67     <style>
68         .select2-container .select2-selection--single {
69             height: 45px !important;
70             display: flex;
71             align-items: center;
72         }
73         .select2-container--default .select2-selection--single .select2-selection__rendered {
74             line-height: 45px !important;
75         }
76     </style>
77 
```



```

82 <head>
83     <style>
84         .select2-container--default .select2-selection--single .select2-selection__arrow {
85             /* Centrar el ícono del menú desplegable */
86             height: 45px !important;
87             display: flex;
88             top: 0 !important;
89             align-items: center;
90             padding: 0 20px;
91         }
92     </style>
93 </head>
94 <body class="font-sans bg-white">
95
96     <section class="flex flex-col-reverse md:flex-row h-auto md:h-screen items-center">
97
98         <!-- Formulario -->
99         <div class="flex-1 w-full flex items-center justify-center">
100             <div class="w-full max-w-2xl">
101                 <h1 class="text-center text-3xl md:text-4xl font-bold text-zinc-800 mb-10">
102                     Registro en la Panadería
103                 </h1>
104
105                 <form id="registerForm" class="grid grid-cols-1 md:grid-cols-2 gap-6">
106                     <!-- Cada campo -->
107                     <div>
108                         <input type="text" id="nombre" name="nombre" placeholder="Nombre"
109                         maxlength="50"
110                         class="w-full bg-gray-200 h-11 rounded-lg px-3 focus:outline-none focus:ring-1 focus:ring-orange-400 focus:bg-white">
111                         <span id="validaNombre" class="text-red-500 text-sm mt-1 block"></span>
112                     </div>
113
114                     <div>
115                         <input type="text" id="apellido" name="apellido" placeholder="Apellido"
116                         maxlength="40"
117                         class="w-full bg-gray-200 h-11 rounded-lg px-3 focus:outline-none focus:ring-1 focus:ring-orange-400 focus:bg-white">
118                         <span id="validaApellido" class="text-red-500 text-sm mt-1 block"></span>
119                     </div>
120
121                     <div>
122                         <input type="email" id="email" name="email" placeholder="Correo electrónico"
123                         class="w-full bg-gray-200 h-11 rounded-lg px-3 focus:outline-none focus:ring-1 focus:ring-orange-400 focus:bg-white">
124                         <span id="validaEmail" class="text-red-500 text-sm mt-1 block"></span>
125                     </div>
126
127                     <div>
128                         <input type="text" id="telefono" name="telefono" placeholder="Teléfono" maxlength="9"
129                         class="w-full bg-gray-200 h-11 rounded-lg px-3 focus:outline-none focus:ring-1 focus:ring-orange-400 focus:bg-white">
130                         <span id="validaTelefono" class="text-red-500 text-sm mt-1 block"></span>
131                     </div>
132
133                 </form>
134             </div>
135         </div>
136     </section>
137 
```

```
index.php create-account.php x
create-account.php
117 <body class="font-sans bg-white">
118   <div class="flex-1 w-full flex items-center justify-center">
119     <div class="w-full max-w-2xl">
120       <form id="registerForm" class="grid grid-cols-1 md:grid-cols-2 gap-6">
121
122       <!-- Dirección: ocupará toda la fila -->
123       <div class="md:col-span-2">
124         <input type="text" id="direccion" name="direccion" placeholder="Dirección"
125           class="w-full bg-gray-200 h-11 rounded-lg px-3 focus:outline-none focus:ring-1 focus:ring-orange-400 focus:bg-white"/>
126         <span id="validaDireccion" class="text-red-500 text-sm block mt-1"></span>
127       </div>
128
129       <!-- Género y Distrito en la misma fila -->
130       <div class="grid grid-cols-1 md:grid-cols-2 gap-6 md:col-span-2">
131         <div>
132           <select id="genero" name="genero"
133             class="w-full bg-gray-200 h-11 rounded-lg px-3 required">
134             <option value="">Selecciona género</option>
135             <option value="0">Masculino</option>
136             <option value="1">Femenino</option>
137             <option value="2">Otro</option>
138           </select>
139           <span id="validaGenero" class="text-red-500 text-sm block mt-1"></span>
140         </div>
141
142         <div>
143           <select id="distrito" name="distrito"
144             class="w-full bg-gray-200 h-11 rounded-lg px-3 required">
145             <option value="">Selecciona un distrito</option>
146             <option value="1">Anón</option>
147             <option value="2">Atac</option>
148             <option value="3">Barranco</option>
149             <option value="4">Breña</option>
150             <option value="5">Carabayllo</option>
151             <option value="6">Cercado de Lima</option>
152             <option value="7">Chalacayo</option>
153             <option value="8">Chorrillos</option>
154             <option value="9">Cieneguilla</option>
155             <option value="10">Comas</option>
156             <option value="11">El Agustino</option>
157             <option value="12">Independencia</option>
158             <option value="13">Jesús María</option>
159             <option value="14">La Molina</option>
160             <option value="15">La Victoria</option>
161             <option value="16">Lince</option>
162             <option value="17">Los Olivos</option>
163             <option value="18">Lurigancho</option>
164             <option value="19">Lurín</option>
165             <option value="20">Magdalena del Mar</option>
166             <option value="21">Miraflores</option>
167             <option value="22">Pachámac</option>
168             <option value="23">Pucusana</option>
169             <option value="24">Pueblo Libre</option>
170             <option value="25">Puente Piedra</option>
171           </select>
172         </div>
173       </div>
174     </div>
175   </form>
176 </div>
```

```
index.php create-account.php

create-account.php
117 <body class="font-sans bg-white">
118   <div class="flex-1 w-full flex items-center justify-center">
119     <div class="w-full max-w-2xl">
120       <form id="registerForm" class="grid grid-cols-1 md:grid-cols-2 gap-6">
121         <div class="grid grid-cols-1 md:grid-cols-2 gap-6">
122           <option value="23">Pucusana</option>
123           <option value="24">Pueblo Libre</option>
124           <option value="25">Puente Piedra</option>
125           <option value="26">Punta Hermosa</option>
126           <option value="27">Punta Negra</option>
127           <option value="28">Rimac</option>
128           <option value="29">San Bartolo</option>
129           <option value="30">San Borja</option>
130           <option value="31">San Isidro</option>
131           <option value="32">San Juan de Lurigancho</option>
132           <option value="33">San Juan de Miraflores</option>
133           <option value="34">San Luis</option>
134           <option value="35">San Martín de Porres</option>
135           <option value="36">San Miguel</option>
136           <option value="37">Santa Anita</option>
137           <option value="38">Santa María del Mar</option>
138           <option value="39">Santa Rosa</option>
139           <option value="40">Santiago de Surco</option>
140           <option value="41">Surquillo</option>
141           <option value="42">Villa El Salvador</option>
142           <option value="43">Villa María del Triunfo</option>
143         </select>
144         <span id="validoDistrito" class="text-red-500 text-sm block mt-1"></span>
145       </div>
146     </div>
147   </div>
148
149   <div type="text" id="dni" name="dni" placeholder="DNI" maxlength="8">
150     <input type="text" id="validoDni" class="text-red-500 text-sm block mt-1"></span>
151   </div>
152
153   <div class="relative">
154     <input type="password" id="password" name="password" placeholder="Contraseña"
155       class="w-full bg-gray-200 h-11 rounded-lg px-3 pr-10 focus:outline-none focus:ring-1 focus:ring-orange-400 focus:bg-white"/>
156     <span id="validoPassword" class="text-red-500 text-sm block mt-1"></span>
157   </div>
158
159   <!-- Botón de registro -->
160   <div class="col-span-1 md:col-span-2">
161     <button type="submit" class="w-full h-11 rounded-lg bg-orange-500 text-white font-medium py-3 px-4 focus:outline-none focus:ring-1 focus:ring-orange-400 focus:bg-white">
162       Registrarse
163     </button>
164   </div>
165
166   <div class="grid grid-cols-1 md:grid-cols-2 gap-6">
```

```
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
```

✓ Ejecución:

Registro en la Panadería

Nombre Apellido

Correo electrónico Teléfono

Dirección

Selección género Selección un distrito

DNI Contraseña

Registrar

¿Ya tienes una cuenta? [Iniciar sesión](#)



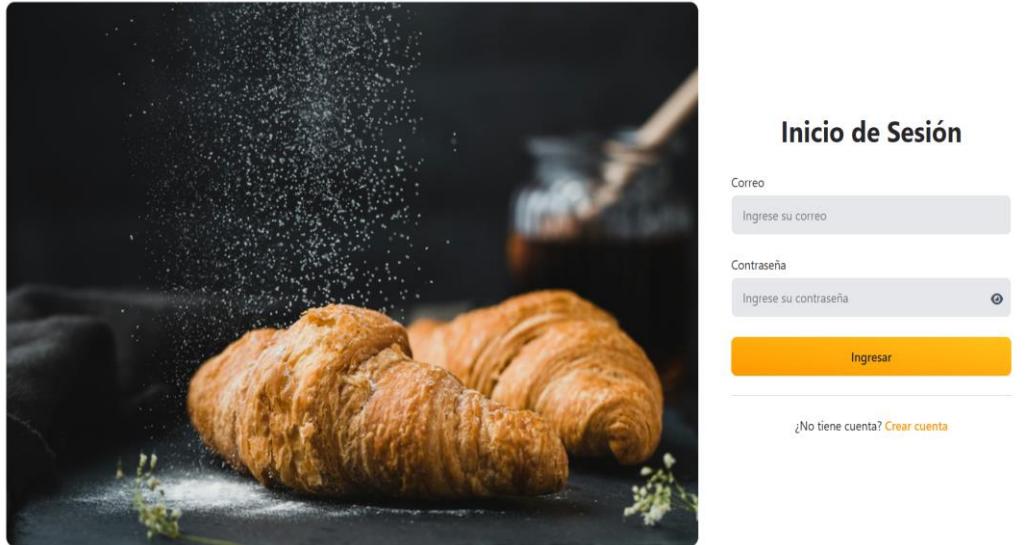
- R.F 13: El sistema debe contar con inicio de sesión para clientes.

✓ Código:

```
index.php create-account.php login.php x

login.php
1 <?php
2
3 if (session_status() == PHP_SESSION_NONE) {
4     session_start();
5 }
6
7
8 if (isset($_SESSION['usuario'])) {
9     if(isset($_SESSION['tipo']) && $_SESSION['tipo'] == 'Administrador'){
10         // Si ya hay una sesion iniciada, redirige al usuario segun su tipo
11         header("Location: Admin/admin-page.php");
12         exit();
13     }else if(isset($_SESSION['tipo']) && $_SESSION['tipo'] == 'Cliente'){
14         header("Location: index.php");
15         exit();
16     }
17 }
18
19
20 ?>
21 <!DOCTYPE html>
22 <html lang="en">
23 <head>
24     <meta charset="UTF-8" />
25     <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
26     <title>Login</title>
27     <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.2/css/all.min.css"/>
28     <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
29     <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
30     <script src="https://cdn.jsdelivr.net/npm/tailwindcss@4"></script>
31 </head>
32 <body class="bg-white">
33     <section class="flex flex-col md:flex-row h-screen p-3">
34
35         <!-- Imagen -->
36         <div class="hidden md:block md:w-1/2 lg:w-2/3 h-1/3 md:h-full">
37             
38         </div>
39
40         <!-- Formulario -->
41         <div class="w-full md:w-1/2 lg:w-1/3 h-full flex items-center justify-center px-4 sm:px-6 lg:px-12">
42             <div class="w-full max-w-md">
43                 <h1 class="text-3xl md:text-4xl font-bold text-center text-zinc-800 mb-8">Inicio de Sesión</h1>
```

✓ Ejecución:



- R.F 14: El sistema debe contar con inicio de sesión para administrador.

✓ Código:

```

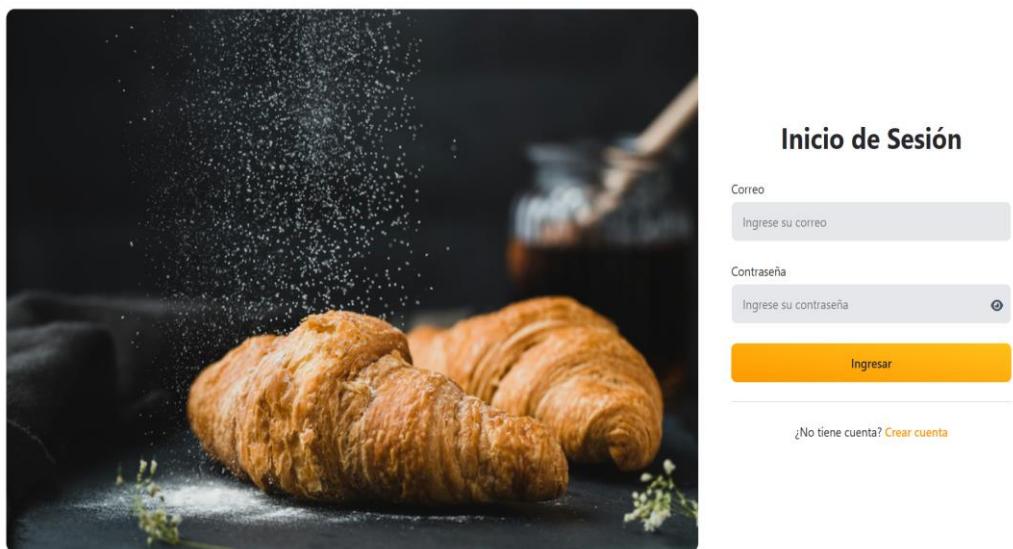
index.php          create-account.php      login.php      JS validarLogin.js ×
assets > js > validarLogin.js > addEventListener('click') callback > passwordField
1   document.addEventListener("DOMContentLoaded", function () {
2     console.log("Documento cargado");
3     $("#LoginForm").submit(function (e) {
4       e.preventDefault();
5
6       let correo = document.getElementById("emailLogin").value.trim();
7       let contrasena = document.getElementById("passwordLogin").value.trim();
8
9
10      if (correo === "" || contrasena === "") {
11        Swal.fire({ title: "Error", text: "Hay campos vacíos", icon: "error" });
12        return;
13      }
14      var formData = $(this).serialize();
15      $.ajax({
16        type: 'POST',
17        url: 'control_sesion.php',
18        data: formData,
19        dataType: 'json',
20        success: function (data) {
21          console.log("Respuesta del servidor:", data);
22          if (data.success) {
23            Swal.fire({
24              title: "Bienvenido@",
25              text: data.message,
26              icon: "success",
27              allowOutsideClick: false // Evita que se cierre al hacer clic fuera
28            }).then(() => {
29              window.location.href = data.redirect; // Redirige según el tipo de usuario
30            });
31          } else {
32            Swal.fire({ title: "Error", text: data.message, icon: "error", allowOutsideClick: false });
33          }
34        },
35        error: function (xhr, status, error) {
36          console.error("Error en AJAX:", xhr.responseText);
37          Swal.fire({
38            icon: "error",
39            title: "Error en la petición",
40            text: "Algo salió mal." + xhr.responseText,
41            confirmButtonColor: "#d33",
42            allowOutsideClick: false, // Evita que se cierre al hacer clic fuera
43          });
44        }
      });
    
```

```

47
48 document.getElementById('verContraseña').addEventListener('click', function () {
49     var passwordField = document.getElementById('passwordLogin');// Se captura el Id del input contraseña
50     var eyeIcon = document.getElementById('eyeIcon');// se captura el Id icono del ojo
51
52     if (passwordField.type === "password") {// Si es de tipo password el input de la contraseña
53         passwordField.type = "text";// Al darle clic se transforma en tipo text el cual permitira visualizar la contraseña
54         eyeIcon.classList.remove("fa-eye"); // se eliminara el icono del ojo tachado
55         eyeIcon.classList.add("fa-eye-slash"); // y se reemplaza por el icono ojo
56     } else {//si no es de tipo password el input de la contraseña
57         passwordField.type = "password"; //se establece de tipo password (para ocultar la contraseña nuevamente)
58         eyeIcon.classList.remove("fa-eye-slash"); // se remueve icono del ojo
59         eyeIcon.classList.add("fa-eye"); // se reemplaza por el icono ojo tachado
60     }
61 }
62 });
63

```

✓ Ejecución:



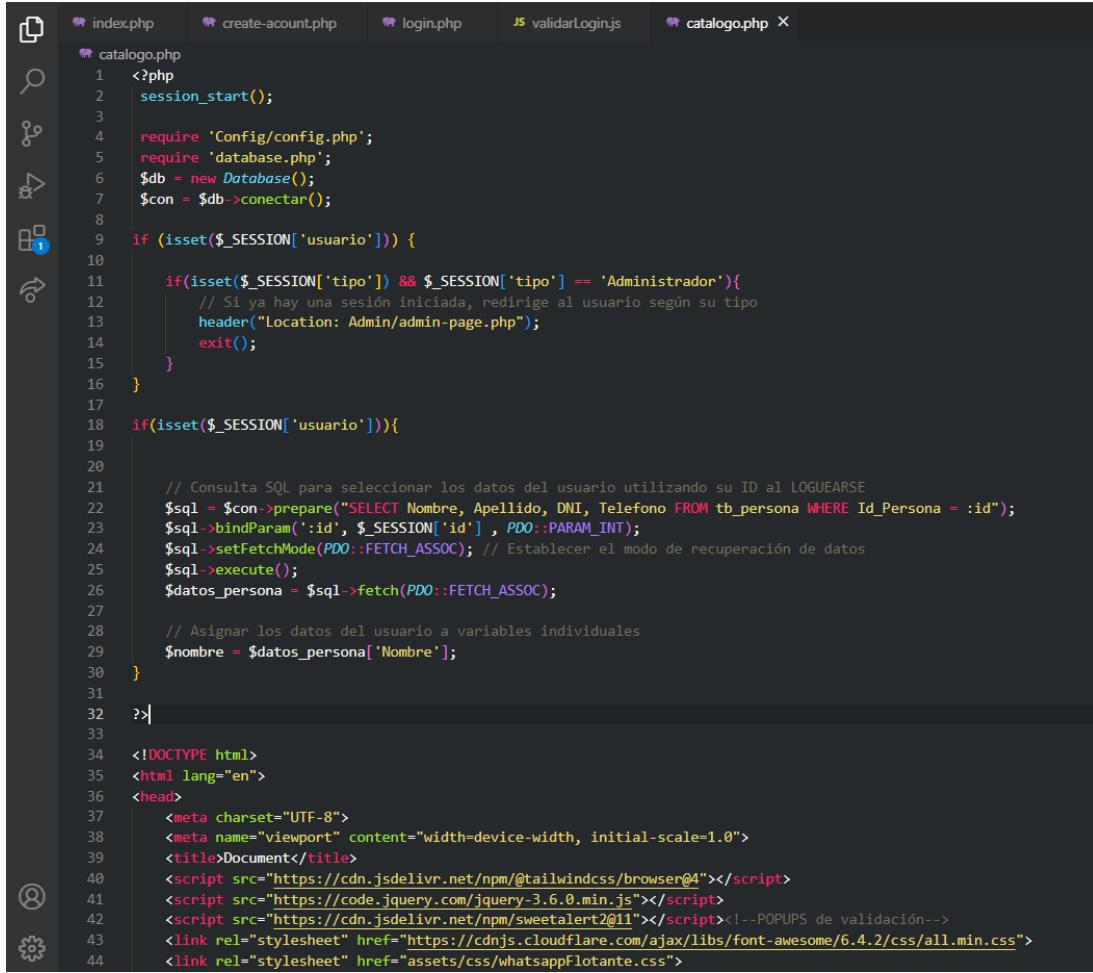
The screenshot displays the "Panel de Administración" (Admin Dashboard) for "Cremas Mithoas". The top navigation bar shows the brand logo and a welcome message: "Bienvenido, Alvaro Ivan". On the left, a sidebar menu lists "Dashboard", "Productos", "Pedidos", "Clientes", and "Repartidores". The main dashboard area features several key performance indicators (KPIs) and charts.

- Resumen General:**
 - Ventas Totales: \$24,780 (+12.5% vs mes anterior)
 - Clientes Nuevos: 1,245 (+8.2% vs mes anterior)
 - Órdenes Completadas: 856 (+5.3% vs mes anterior)
 - Productos Vendidos: 3,456 (+15.7% vs mes anterior)
- Ventas Mensuales:** A grouped bar chart comparing "Ventas Online" (dark blue) and "Ventas Físicas" (light blue) from January to August. Total sales show a steady increase over the period.
- Productos Tendencia:** A donut chart showing the distribution of product categories by sales volume. The legend includes: Electrónicos (purple), Ropa (blue), Hogar (teal), Alimentos (green), Juguete (orange), Libros (red), and Otros (grey).

❖ Inspección del módulo de Gestión de Pedidos

- R.F 1: El sistema debe permitir a los clientes realizar pedidos en línea.

✓ Código:



The screenshot shows a code editor interface with several tabs at the top: index.php, create-acont.php, login.php, validarLogin.js, and catalogo.php (which is the active tab). The catalogo.php file contains PHP code for handling user sessions and displaying user data. The code includes session starts, database connections, SQL queries, and HTML output. It checks if a user is logged in and if they are an administrator, then retrieves user data from the database and displays it in an HTML page.

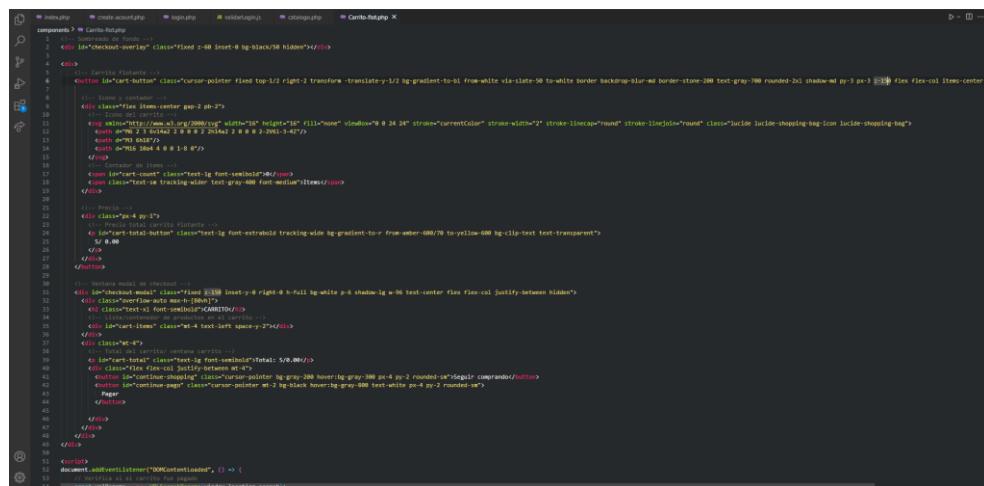
```
<?php
session_start();
require 'Config/config.php';
require 'database.php';
$db = new Database();
$con = $db->conectar();
if (isset($_SESSION['usuario'])) {
    if(isset($_SESSION['tipo']) && $_SESSION['tipo'] == 'Administrador'){
        // Si ya hay una sesión iniciada, redirige al usuario según su tipo
        header("Location: Admin/admin-page.php");
        exit();
    }
}
if(isset($_SESSION['usuario'])){
    // Consulta SQL para seleccionar los datos del usuario utilizando su ID al LOGUEARSE
    $sql = $con->prepare("SELECT Nombre, Apellido, DNI, Telefono FROM tb_persona WHERE Id_Persona = :id");
    $sql->bindParam(':id', $_SESSION['id'], PDO::PARAM_INT);
    $sql->setFetchMode(PDO::FETCH_ASSOC); // Establecer el modo de recuperación de datos
    $sql->execute();
    $datos_persona = $sql->fetch(PDO::FETCH_ASSOC);

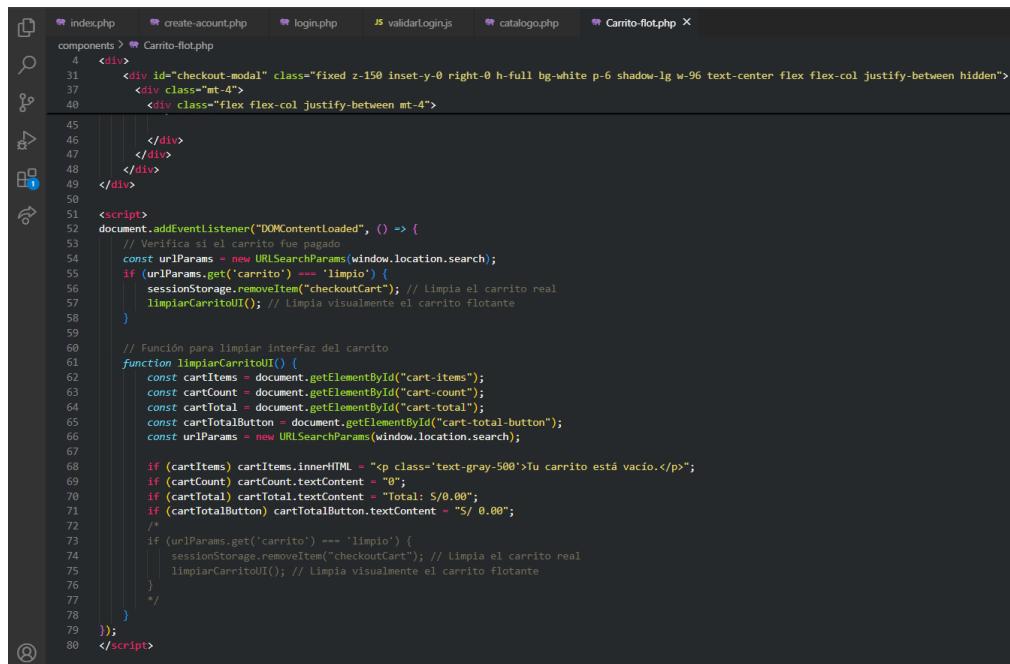
    // Asignar los datos del usuario a variables individuales
    $nombre = $datos_persona['Nombre'];
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script src="https://cdn.jsdelivr.net/npm/tailwindcss@4"></script>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script><!--POPUPS de validación-->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.2/css/all.min.css">
    <link rel="stylesheet" href="assets/css/whatsappFlotante.css">
```

```

45 </head>
46 <body>
47     <?php
48
49     // Verifica si el usuario está logeado (por ejemplo, si existe una variable de sesión "usuario")
50     if (isset($_SESSION['usuario'])) {
51         include 'components/Navbar_cliente.php'; // Navbar para usuarios logueados
52     } else {
53         include 'components/Navbar.php'; // Navbar para usuarios no logueados
54     }
55     include 'components/Productos.php';
56     include 'components/Toogle-flot.php';
57     include 'components/Carrito-flot.php';
58     include 'components/Footer.php';
59     include 'components/icono_flotante_whatsapp.php';
60
61     <!--<script src="assets/js/cerrar_sesion.js"></script>-->
62     <script src="assets/api/products.json"></script>
63     <script src="assets/js/category.js"></script>
64     <script src="assets/js/pagination.js"></script>
65     <script src="assets/cart.js" defer></script>
66 </body>
67 </html>

```



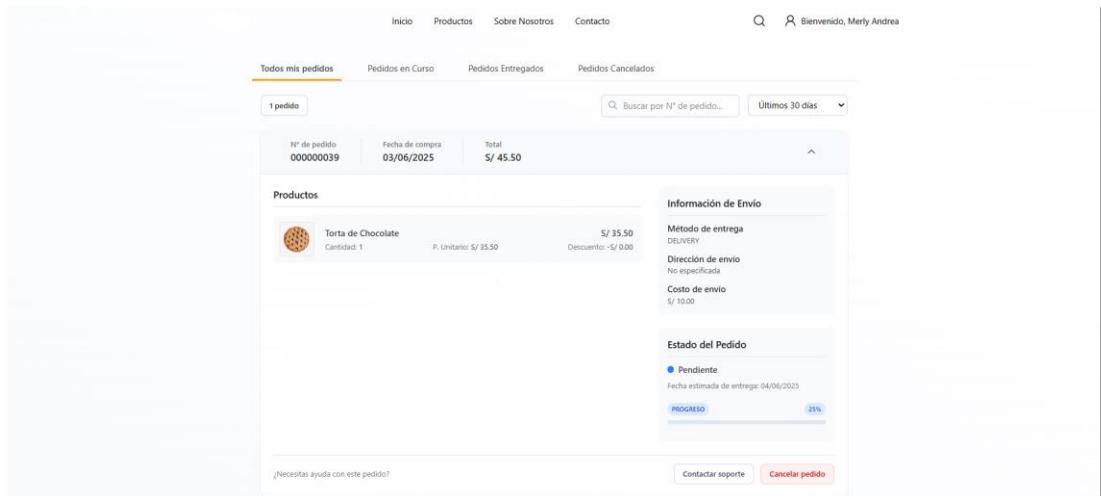


```

index.php          create-account.php    login.php      validarLogin.js    catalogo.php   Carrito-flot.php X
components > Carrito-flot.php
  4  <div>
  5      <div id="checkout-modal" class="fixed z-100 inset-y-0 right-0 h-full bg-white p-6 shadow-lg w-96 text-center flex flex-col justify-between hidden">
  6          <div class="mt-4">
  7              <div class="flex flex-col justify-between mt-4">
  8
  9      </div>
 10
 11      <script>
 12      document.addEventListener("DOMContentLoaded", () => {
 13          // Verifica si el carrito fue pagado
 14          const urlParams = new URLSearchParams(window.location.search);
 15          if (urlParams.get('carrito') === 'limpio') {
 16              sessionStorage.removeItem("checkoutCart"); // Limpia el carrito real
 17              limpiarCarritoUI(); // Limpia visualmente el carrito flotante
 18          }
 19
 20          // Función para limpiar interfaz del carrito
 21          function limpiarCarritoUI() {
 22              const cartItems = document.getElementById("cart-items");
 23              const cartCount = document.getElementById("cart-count");
 24              const cartTotal = document.getElementById("cart-total");
 25              const cartTotalButton = document.getElementById("cart-total-button");
 26              const urlParams = new URLSearchParams(window.location.search);
 27
 28              if (cartItems) cartItems.innerHTML = "<p class='text-gray-500'>Tu carrito está vacío.</p>";
 29              if (cartCount) cartCount.textContent = "0";
 30              if (cartTotal) cartTotal.textContent = "Total: S/0.00";
 31              if (cartTotalButton) cartTotalButton.textContent = "S/ 0.00";
 32
 33              /*
 34              if (urlParams.get('carrito') === 'limpio') {
 35                  sessionStorage.removeItem("checkoutCart"); // Limpia el carrito real
 36                  limpiarCarritoUI(); // Limpia visualmente el carrito flotante
 37              }
 38          }
 39      });
 40      </script>

```

✓ Ejecución:



The screenshot shows a user interface for managing orders. At the top, there's a navigation bar with links for Inicio, Productos, Sobre Nosotros, Contacto, and a search bar. Below that, a header displays 'Bienvenido, Merly Andrea'. The main content area shows a summary of a single order:

Todos mis pedidos		Pedidos En Curso	Pedidos Entregados	Pedidos Cancelados
1 pedido				
Nº de pedido	000000039	Fecha de compra	03/06/2025	Total
				S/ 45.50

Below the summary, there's a detailed view of the order items and shipping information.

Productos

	Torta de Chocolate	Cantidad: 1	P. Unitario: S/ 35.50	S/ 35.50	Descuento: -S/ 0.00
--	--------------------	-------------	-----------------------	----------	---------------------

Información de Envío

Método de entrega	DELIVERY
Dirección de envío	No especificada
Costo de envío	S/ 10.00

Estado del Pedido

Pendiente (radio button selected)

Fecha estimada de entrega: 04/06/2025

PROGRESO: 21%

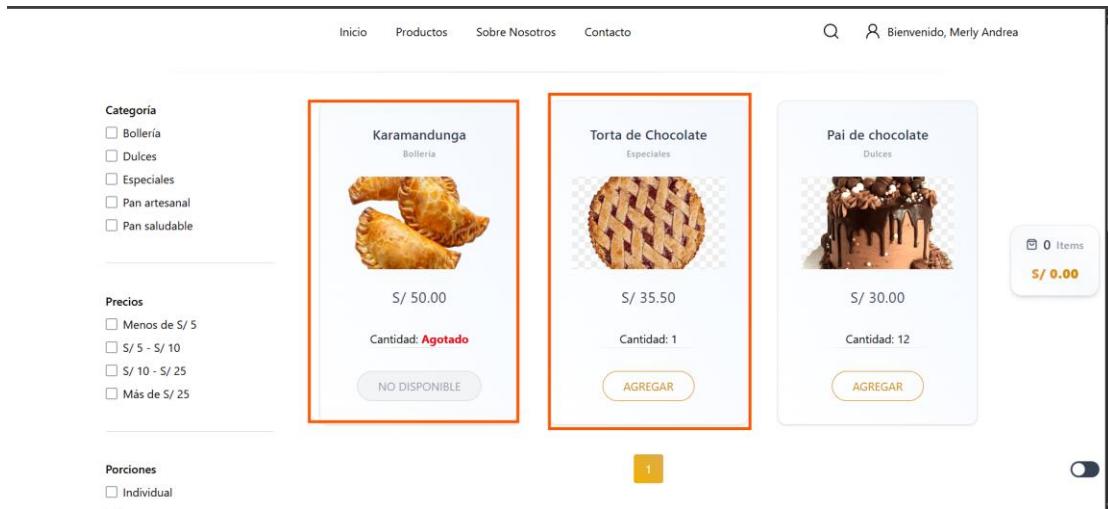
At the bottom, there are buttons for 'Contactar soporte' and 'Cancelar pedido'.

- R.F 2: El sistema debe mostrar en tiempo real la disponibilidad de productos.

✓ Código:

```
  assets > > pagination.js > document.addEventListener('DOMContentLoaded') callback > renderProducts > paginatedProductsForEach callback > productCard
  document.addEventListener('DOMContentLoaded', () => {
    const renderProducts = products, nro => => {
      const productCard = document.createElement('div');
      const productsHTML = products.map(product => {
        const start = nro * 8;
        const end = start + 8;
        const paginatedProducts = products.slice(start, end);
        const productCard = document.createElement('div');
        const productCardHTML = `
          <div class="row justify-content-between">
            <div class="col-12 col-md-6 col-lg-4">
              <div class="card mb-2">
                <div class="card-body">
                  <div class="row align-items-center justify-content-between">
                    <div>
                      
                      <div>
                        <strong>${product.Nombre_Producto}</strong>
                        <p>${product.Descripcion_Categoría}</p>
                      </div>
                    </div>
                    <div>
                      <strong>${product.Precio_Actual}</strong>
                      <small>${product.Precio_Descuento}</small>
                    </div>
                  </div>
                  <div>
                    ${product.Stock ? `<small>Stock: ${product.Stock}</small>` : `<small>AGOTADO</small>`}
                  </div>
                </div>
              </div>
            </div>
          </div>
        `;
        productCard.innerHTML = productCardHTML;
        return productCard;
      });
      productCard.innerHTML = productsHTML;
      productGrid.innerHTML += productCard;
    };
  });
  productGrid.innerHTML += productCard;
});
```

✓ Ejecución:

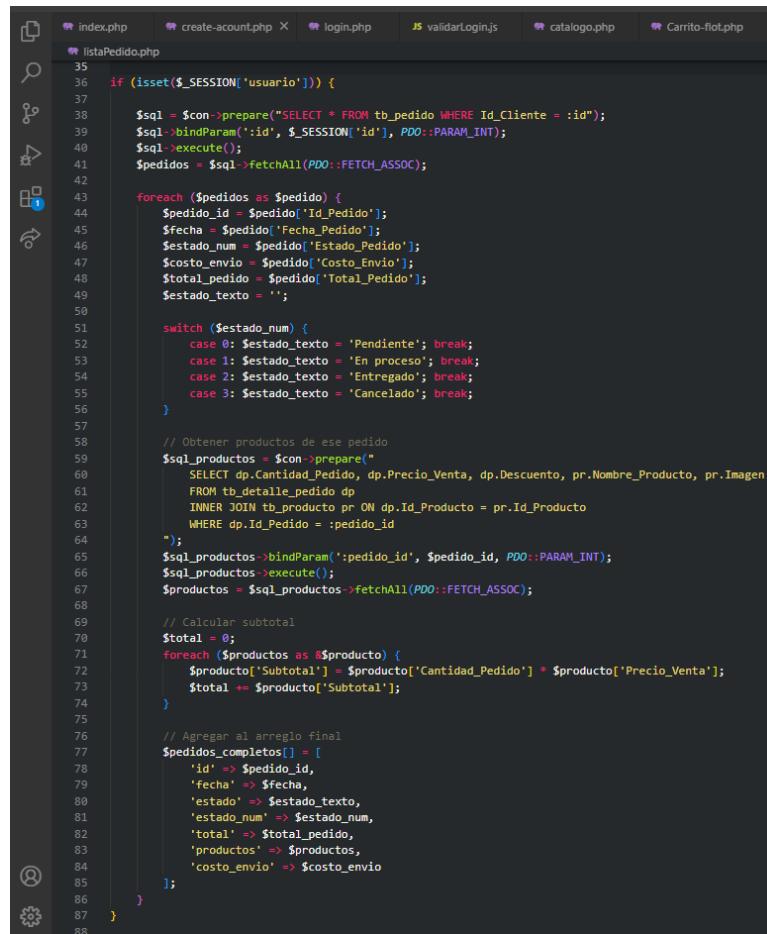


- R.F 3: El sistema debe registrar los pedidos y asignarles un estado.

✓ Código:

```
index.php          create-account.php    login.php      validarLogin.js    catalogo.php    Carrito-flo.php    pedidos

listaPedido.php
1 <?php
2
3 session_start();
4
5 require 'Config/config.php';
6 require 'database.php';
7 $db = new Database();
8 $con = $db->conectar();
9 //SI es que aún no has cerrado sesión te va enviar la página correspondiente, aunque cierres la página
10 //Para usar estas líneas de código se debe crear un apartado de cerrar sesión.
11
12 if(isset($_SESSION['usuario'])) {
13     // Si ya hay una sesión iniciada, redirige al usuario según su tipo
14     if(isset($_SESSION['tipo']) && $_SESSION['tipo'] == 'admin') {
15         header("Location: Admin/admin-page.php");
16         exit();
17     }
18 }
19
20 if(isset($_SESSION['usuario'])){
21
22
23     // Consulta SQL para seleccionar los datos del usuario utilizando su ID al LOGUEARSE
24     $sql = $con->prepare("SELECT Nombre, Apellido, DNI, Telefono FROM tb_persona WHERE Id_Persona = :id");
25     $sql->bindParam(':id', $_SESSION['id'], PDO::PARAM_INT);
26     $sql->execute(PDO::FETCH_ASSOC); // Establecer el modo de recuperación de datos
27     $sql->fetch(PDO::FETCH_ASSOC);
28     $datos_persona = $sql->fetch(PDO::FETCH_ASSOC);
29
30     // Asignar los datos del usuario a variables individuales
31     $Nombre = $datos_persona['Nombre'];
32 }
33
34 $pedidos_completos = [];
35
36 if(isset($_SESSION['usuario'])) {
37
38     $sql = $con->prepare("SELECT * FROM tb_pedido WHERE Id_Cliente = :id");
39     $sql->bindParam(':id', $_SESSION['id'], PDO::PARAM_INT);
40     $sql->execute();
41     $pedidos = $sql->fetchAll(PDO::FETCH_ASSOC);
42
43     foreach ($pedidos as $pedido) {
44         $pedido['id'] = $pedido['Id_Pedido'];
45     }
46 }
```

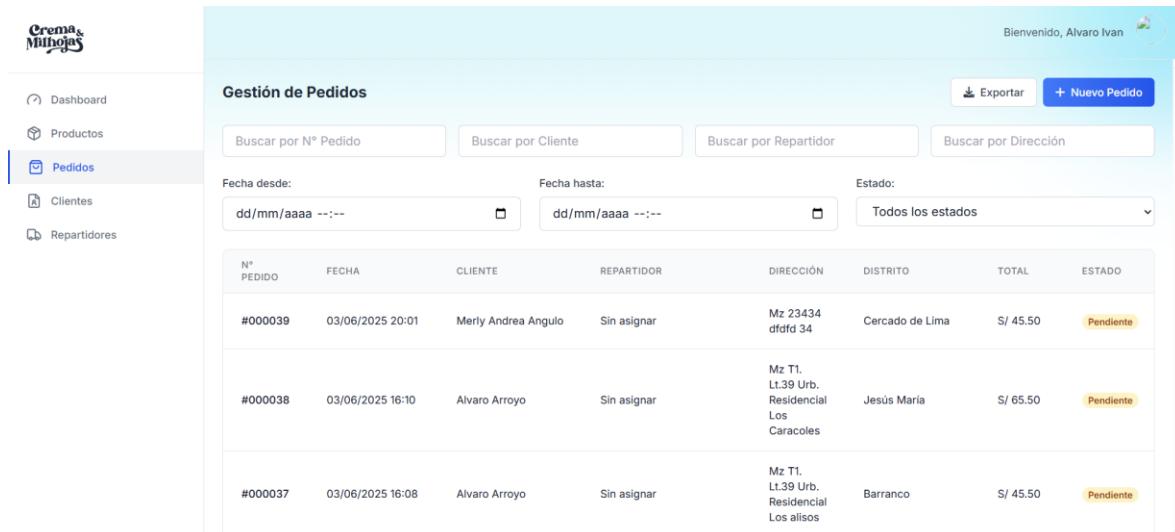


```

35     if (isset($_SESSION['usuario'])) {
36
37         $sql = $con->prepare("SELECT * FROM tb_pedido WHERE Id_Cliente = :id");
38         $sql->bindParam(':id', $_SESSION['id'], PDO::PARAM_INT);
39         $sql->execute();
40         $pedidos = $sql->fetchAll(PDO::FETCH_ASSOC);
41
42         foreach ($pedidos as $pedido) {
43             $pedido_id = $pedido['Id_Pedido'];
44             $fecha = $pedido['Fecha_Pedido'];
45             $estado_num = $pedido['Estado_Pedido'];
46             $costo_envio = $pedido['Costo_Envio'];
47             $total_pedido = $pedido['Total_Pedido'];
48             $estado_texto = '';
49
50             switch ($estado_num) {
51                 case 0: $estado_texto = 'Pendiente'; break;
52                 case 1: $estado_texto = 'En proceso'; break;
53                 case 2: $estado_texto = 'Entregado'; break;
54                 case 3: $estado_texto = 'Cancelado'; break;
55             }
56
57             // Obtener productos de ese pedido
58             $sql_productos = $con->prepare(
59                 "SELECT dp.Cantidad_Pedido, dp.Precio_Venta, dp.Descuento, pr.Nombre_Producto, pr.Imagen
60                 FROM tb_detalle_pedido dp
61                 INNER JOIN tb_producto pr ON dp.Id_Producto = pr.Id_Producto
62                 WHERE dp.Id_Pedido = :pedido_id
63             ");
64             $sql_productos->bindParam(':pedido_id', $pedido_id, PDO::PARAM_INT);
65             $sql_productos->execute();
66             $productos = $sql_productos->fetchAll(PDO::FETCH_ASSOC);
67
68             // Calcular subtotal
69             $total = 0;
70             foreach ($productos as $producto) {
71                 $producto['Subtotal'] = $producto['Cantidad_Pedido'] * $producto['Precio_Venta'];
72                 $total += $producto['Subtotal'];
73             }
74
75             // Agregar al arreglo final
76             $pedidos_completos[] = [
77                 'id' => $pedido_id,
78                 'fecha' => $fecha,
79                 'estado' => $estado_texto,
80                 'estado_num' => $estado_num,
81                 'total' => $total_pedido,
82                 'productos' => $productos,
83                 'costo_envio' => $costo_envio
84             ];
85         }
86     }
87 }

```

✓ Ejecución:



Nº PEDIDO	FECHA	CLIENTE	REPARTIDOR	DIRECCIÓN	DISTRITO	TOTAL	ESTADO
#000039	03/06/2025 20:01	Mery Andrea Angulo	Sin asignar	Mz 23434 dfdfd 34	Cercado de Lima	S/ 45.50	Pendiente
#000038	03/06/2025 16:10	Alvaro Arroyo	Sin asignar	Mz T1. Lt.39 Urb. Residencial Los Caracoles	Jesús María	S/ 65.50	Pendiente
#000037	03/06/2025 16:08	Alvaro Arroyo	Sin asignar	Mz T1. Lt.39 Urb. Residencial Los alisos	Barranco	S/ 45.50	Pendiente

❖ Inspección del módulo Control de Inventario

- R.F 4: El sistema debe registrar, editar y eliminar productos del inventario.

✓ Código:

```
index.php create-account.php login.php validarLogin.js catalogo.php Carrito-flot.php pedidos-admin

Admin > productos-admin.php
1  <?php
2  session_start();
3  require '../Config/config.php';
4  require '../database.php';
5
6  $db = new Database();
7  $con = $db->conectar();
8
9  // Redireccion según tipo de usuario
10 // Si ya hay una sesión iniciada, redirige al usuario según su tipo
11 if (isset($_SESSION['usuario'])) {
12     if (isset($_SESSION['tipo']) && $_SESSION['tipo'] == 'Cliente') [
13         header("Location: ../index.php");
14         exit();
15     ] else {
16         $sql = $con->prepare("SELECT Nombre, Apellido, DNI, Telefono FROM tb_persona WHERE Id_Persona = :id");
17         $sql->bindParam(':id', $_SESSION['id'], PDO::PARAM_INT);
18         $sql->setFetchMode(PDO::FETCH_ASSOC);
19         $sql->execute();
20         $datos_persona = $sql->fetch(PDO::FETCH_ASSOC);
21         $Nombre = $datos_persona['Nombre'];
22
23         if (isset($con)) {
24             $stmt = $con->query("SELECT * FROM tb_producto");
25             $productos = $stmt->fetchAll(PDO::FETCH_ASSOC);
26         } else {
27             echo "Error: La conexión no se ha establecido.";
28         }
29
30         // Lógica de la alerta de stock bajo
31         $alerta_stock_bajo = false;
32         foreach ($productos as $prod) {
33             if ($prod['Disponibilidad'] == 0 || $prod['Disponibilidad'] == 2) {
34                 $alerta_stock_bajo = true;
35                 break;
36             }
37         }
38     }
39 }
40
41 // Verificamos si se envió el formulario de agregar producto
42 if (isset($_POST['accion']) && $_POST['accion'] === 'agregar') {
43     // Ruta donde se guardarán las imágenes
44     $directorioDestino = "../Admin/images/";
45     $foto = '';//Inicializamos la variable foto como vacía
46
47     // Verificar si hay una imagen subida
48     // y si no hay errores en la carga
49     if (isset($_FILES['imagenP']) && $_FILES['imagenP']['error'] === UPLOAD_ERR_OK) {
50         $fotoNombre = basename($_FILES['imagenP']['name']); // nombre de la imagen de acuerdo al nombre del archivo
51         $rutaDestino = $directorioDestino . $fotoNombre; // ruta de la imagen
52         $rutaBD = "images/" . $fotoNombre; // ruta dentro del archivo Admin
53         // Asegurarse que la carpeta existe
54         if (!is_dir($directorioDestino)) {
```

```

index.php create-account.php login.php validarLogin.js catalogo.php Carrito-filt.php pedidos-admin.php Compras.php ListaPedido.php

Admin > ● productos-admin.php
42 if (isset($_POST['accion']) && $_POST['accion'] === 'agregar') {
43     if (is_uploaded_file($_FILES['ImagenP']) && $_FILES['ImagenP']['error'] === UPLOAD_ERR_OK) {
44         // Verificar que la carpeta destino existe
45         if (!is_dir($directorioDestino)) {
46             mkdir($directorioDestino, 0777, true);
47         }
48         // Mover el archivo a la carpeta destino
49         if (move_uploaded_file($_FILES['ImagenP']['tmp_name'], $rutaDestino)) {
50             $foto = $rutaBD; // Guardamos la ruta relativa en la BD
51         } else {
52             $foto = ''; // En caso de error, dejamos vacío
53         }
54     } else {
55         $foto = ''; // Si no hay imagen, se deja vacío
56     }
57 }
58
59 // Preparar la consulta SQL de inserción
60 $sql_insert = $con->prepare("INSERT INTO tb_producto (Id_Categoría, Nombre_Producto, Descripción_Producto, Stock, Stock_Min, Precio_Actual, Imagen)
VALUES (?, ?, ?, ?, ?, ?, ?)");
61
62 // Vincular parámetros y ejecutar la consulta
63 $categoria = $_POST['categoría'];
64 $nombre_producto = $_POST['nombreP'];
65 $descripcion = $_POST['descripciónP'];
66 $stock = $_POST['stockP'];
67 $stockMin = $_POST['stockMinP'];
68 $precio = $_POST['precioP'];
69
70 $sql_insert->bindParam(1, $categoria, PDO::PARAM_INT);
71 $sql_insert->bindParam(2, $nombre_producto, PDO::PARAM_STR);
72 $sql_insert->bindParam(3, $descripcion, PDO::PARAM_STR);
73 $sql_insert->bindParam(4, $stock, PDO::PARAM_INT);
74 $sql_insert->bindParam(5, $stockMin, PDO::PARAM_INT);
75 $sql_insert->bindParam(6, $precio, PDO::PARAM_STR);
76 $sql_insert->bindParam(7, $foto, PDO::PARAM_STR);
77
78
79 header('Content-Type: application/json');
80 if ($sql_insert->execute()) {
81     echo json_encode(["status" => "success", "message" => "Producto agregado correctamente"]);
82     exit;
83 } else {
84     echo json_encode(["status" => "error", "message" => "Error al agregar el producto"]);
85     exit;
86 }
87
88 // Verificar si se envió el formulario de actualización de producto
89 if (isset($_POST['accion']) && $_POST['accion'] === 'actualizar') {
90     // Ruta donde se guardarán las imágenes
91     $directorioDestino = '../Admin/images/';
92     $foto = ''; // Inicializamos la variable foto como vacía
93
94     // Obtener el código del producto
95
96 }
97
98
99 if (isset($_POST['accion']) && $_POST['accion'] === 'actualizar') {
100     // Obtener el código del producto
101     $codigo = $_POST['codigoEditarP'];
102     $nombre = $_POST['nombreEditarP'];
103     $stock = $_POST['stockEditarP'];
104     $stockMin = $_POST['stockMinEditarP'];
105     $precio = $_POST['precioEditarP'];
106     $categoria = $_POST['categoríaEditarP'];
107     $descripcion = $_POST['descripciónEditarP'];
108     $disponibilidad = $_POST['disponibilidadP'];
109     $estado = $_POST['estadoEditarP'];
110
111     $stock_valor = (float) $stock;
112     $stockMin_valor = (float) $stockMin;
113
114     if ($stock_valor == 0) {
115         $disponibilidad = 0; // Agotado
116     } else if ($stock_valor < $stockMin_valor) {
117         $disponibilidad = 2; // Escaso
118     } else {
119         $disponibilidad = 1; // Disponible
120     }
121
122     if ($estado == '1') {
123         $estado = 1; // Convertir a 1 si está activo
124     } else {
125         $estado = 0; // Convertir a 0 si no está activo
126     }
127
128     // Consulta para obtener la imagen actual del producto
129     $sql_get_image = $con->prepare("SELECT Imagen FROM tb_producto WHERE Id_Producto = ?");
130     $sql_get_image->execute([$codigo]);
131     $producto = $sql_get_image->fetch(PDO::FETCH_ASSOC);
132     $imagenActual = $producto['Imagen']; // Imagen actual en la BD
133
134     // Manejo de la nueva imagen
135     if (empty($_FILES['ImagenEditarP']['name']) && $_FILES['ImagenEditarP']['error'] === UPLOAD_ERR_OK) {
136         $fotoNombre = basename($_FILES['ImagenEditarP']['name']);
137         $rutaDestino = $directorioDestino . $fotoNombre;
138         $rutaBD = "images/" . $fotoNombre;
139
140         // Asegurar que la carpeta de destino existe
141         if (!is_dir($directorioDestino)) {
142             mkdir($directorioDestino, 0777, true);
143         }
144
145         if (move_uploaded_file($_FILES['ImagenEditarP']['tmp_name'], $rutaDestino)) {
146             // Verificar si se debe eliminar la imagen anterior
147             if (empty($imagenActual) && file_exists('../Admin/' . $imagenActual) && $imagenActual != $rutaBD) {
148                 unlink('../Admin/' . $imagenActual);
149             }
150         }
151         $foto = $rutaBD; // Guardamos la nueva ruta de imagen en la BD
152     }
153 }
154

```

```

index.php create-account.php login.php validarLogin.js catalogo.php Carrito-filt.php pedidos-admin.php Compras.php ListaPedido.php

Admin > ● productos-admin.php
99 if (isset($_POST['accion']) && $_POST['accion'] === 'actualizar') {
100
101     // Obtener el código del producto
102     $codigo = $_POST['codigoEditarP'];
103     $nombre = $_POST['nombreEditarP'];
104     $stock = $_POST['stockEditarP'];
105     $stockMin = $_POST['stockMinEditarP'];
106     $precio = $_POST['precioEditarP'];
107     $categoria = $_POST['categoríaEditarP'];
108     $descripcion = $_POST['descripciónEditarP'];
109     $disponibilidad = $_POST['disponibilidadP'];
110     $estado = $_POST['estadoEditarP'];
111
112     $stock_valor = (float) $stock;
113     $stockMin_valor = (float) $stockMin;
114
115     if ($stock_valor == 0) {
116         $disponibilidad = 0; // Agotado
117     } else if ($stock_valor < $stockMin_valor) {
118         $disponibilidad = 2; // Escaso
119     } else {
120         $disponibilidad = 1; // Disponible
121     }
122
123     if ($estado == '1') {
124         $estado = 1; // Convertir a 1 si está activo
125     } else {
126         $estado = 0; // Convertir a 0 si no está activo
127     }
128
129     // Consulta para obtener la imagen actual del producto
130     $sql_get_image = $con->prepare("SELECT Imagen FROM tb_producto WHERE Id_Producto = ?");
131     $sql_get_image->execute([$codigo]);
132     $producto = $sql_get_image->fetch(PDO::FETCH_ASSOC);
133     $imagenActual = $producto['Imagen']; // Imagen actual en la BD
134
135     // Manejo de la nueva imagen
136     if (empty($_FILES['ImagenEditarP']['name']) && $_FILES['ImagenEditarP']['error'] === UPLOAD_ERR_OK) {
137         $fotoNombre = basename($_FILES['ImagenEditarP']['name']);
138         $rutaDestino = $directorioDestino . $fotoNombre;
139         $rutaBD = "images/" . $fotoNombre;
140
141         // Asegurar que la carpeta de destino existe
142         if (!is_dir($directorioDestino)) {
143             mkdir($directorioDestino, 0777, true);
144         }
145
146         if (move_uploaded_file($_FILES['ImagenEditarP']['tmp_name'], $rutaDestino)) {
147             // Verificar si se debe eliminar la imagen anterior
148             if (empty($imagenActual) && file_exists('../Admin/' . $imagenActual) && $imagenActual != $rutaBD) {
149                 unlink('../Admin/' . $imagenActual);
150             }
151         }
152         $foto = $rutaBD; // Guardamos la nueva ruta de imagen en la BD
153     }
154

```

```
index.php    create-account.php    login.php    validLogin.php    catalogo.php    Cartito-Rut.php    pedidos-admin.php    Compras.php    listaPedidos.php    products-admin.php    x
Admin    products-admin.php
139 // Preparar la consulta SQL para insertar
140 $sql_update = $conn->prepare("UPDATE tb_producto SET Id_Categoría=?, Nombre_Producto=?, Descripción_Producto=?, Stock=?, Stock_Min=? ,Precio_Actual=?,Imagen=?, Disponibilidad=?, Estado_Producto=? WHERE Id_Producto=?");
141
142 // Vincular parámetros y ejecutar la consulta
143 $sql_update->bindParam(1, $catParam, PDO::PARAM_INT);
144 $sql_update->bindParam(2, $nameParam, PDO::PARAM_STR);
145 $sql_update->bindParam(3, $descParam, PDO::PARAM_STR);
146 $sql_update->bindParam(4, $stockParam, PDO::PARAM_INT);
147 $sql_update->bindParam(5, $stockMinParam, PDO::PARAM_INT);
148 $sql_update->bindParam(6, $priceParam, PDO::PARAM_FLOAT);
149 $sql_update->bindParam(7, $imageParam, PDO::PARAM_STR);
150 $sql_update->bindParam(8, $availabilityParam, PDO::PARAM_INT);
151 $sql_update->bindParam(9, $statusParam, PDO::PARAM_INT);
152 $sql_update->bindParam(10, $codigoParam, PDO::PARAM_STR);
153
154 header('Content-type: application/json');
155 if ($sql_update->execute()) {
156     echo json_encode(["status" => "success", "message" => "Producto actualizado correctamente"]);
157 } else {
158     echo json_encode(["status" => "error", "message" => "Error al actualizar el producto"]);
159 }
160
161 // Verificar si se envió la solicitud para eliminar un producto
162 if (isset($_POST['eliminar'])) $sql_update->execute();
163
164 // Preparar la consulta SQL de eliminación
165 $sql_delete = $conn->prepare("DELETE tb_producto SET Estado_Producto = ? WHERE Id_Producto = ?");
166
167 // Vincular el código del producto y ejecutar la consulta de eliminación
168 $codigo = $_POST['codigo'];
169 $estado = 0;
170
171 $sql_delete->bindParam(1, $estado, PDO::PARAM_INT);
172 $sql_delete->bindParam(2, $codigo, PDO::PARAM_INT);
173
174 header('Content-type: application/json');
175 if ($sql_delete->execute()) {
176     echo json_encode(["status" => "success", "message" => "Producto eliminado correctamente"]);
177 } else {
178     echo json_encode(["status" => "error", "message" => "Error al eliminar el producto"]);
179 }
```

✓ Ejecución:

Bienvenido, Alvaro Ivan

Listado de Productos

+ Agregar Producto

Buscar por Código

Buscar por Nombre

Filtrar por Categoría:

Selecciona categorías

Filtrar por Disponibilidad:

Selecciona disponibilidad

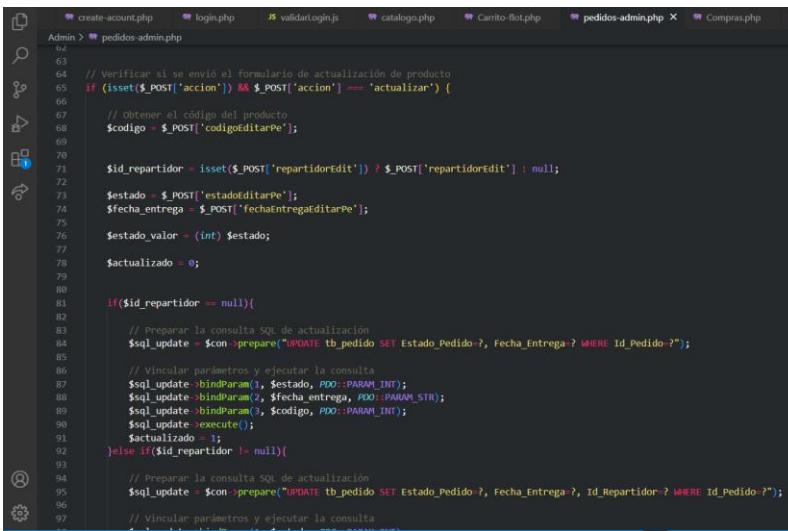
Filtrar por Estado:

Selecciona estado

GO	IMAGEN	CATEGORÍA	NOMBRE	DESCRIPCIÓN	STOCK	PRECIO	DISPONIBILIDAD	ESTADO	ACCIONES
		Bolleria	Karamandunga	dulce crocante	0	S/ 50.00	Escaso	Activo	<button>Editar</button> <button>Eliminar</button>
		Especiales	Torta de Chocolate	Deliciosa torta de chocolate con crema	1	S/ 35.50	Escaso	Activo	<button>Editar</button> <button>Eliminar</button>
		Especiales	Tutifrutti	GAAA	5	S/ 20.00	Escaso	Inactivo	<button>Editar</button> <button>Eliminar</button>

- R.F 5: El sistema debe actualizar el stock automáticamente cuando se realiza un pedido.

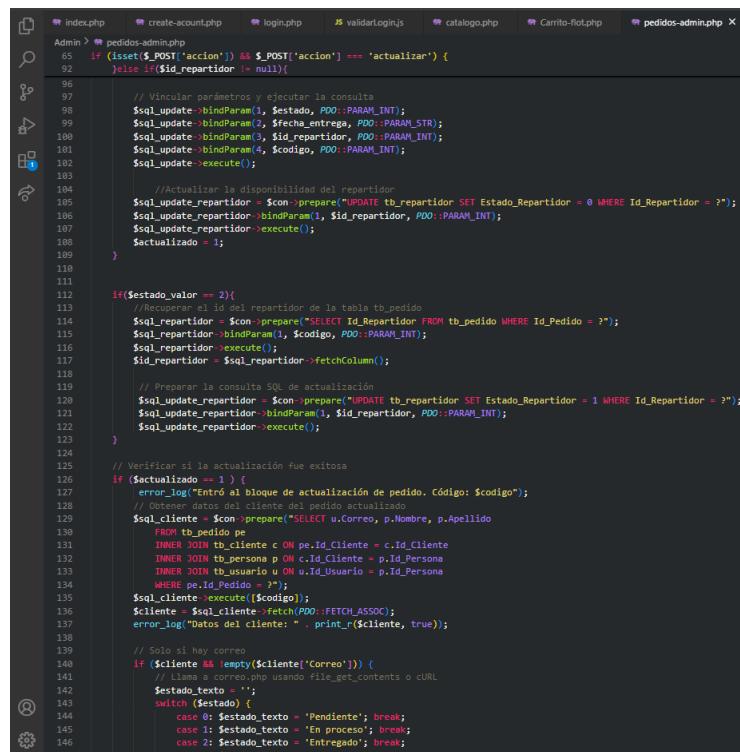
✓ Código:



```

Admin > pedidos-admin.php
62
63 // Verificar si se envió al formulario de actualización de producto
64 if (isset($_POST['accion']) && $_POST['accion'] == 'actualizar') {
65
66     // Obtener el código del producto
67     $codigo = $_POST['codigoEditarPe'];
68
69
70     $id_repartidor = isset($_POST['repartidorEdit']) ? $_POST['repartidorEdit'] : null;
71
72     $estado = $_POST['estadoditarPe'];
73     $fecha_entrega = $_POST['fechaEntregaEditarPe'];
74
75     $estado_valor = (int) $estado;
76
77     $actualizado = 0;
78
79
80     if($id_repartidor == null){
81
82         // Preparar la consulta SQL de actualización
83         $sql_update = $con->prepare("UPDATE tb_pedido SET Estado_Pedido=? , Fecha_Entrega=? WHERE Id_Pedido=?");
84
85         // Vincular parámetros y ejecutar la consulta
86         $sql_update->bindParam(1, $estado, PDO::PARAM_INT);
87         $sql_update->bindParam(2, $fecha_entrega, PDO::PARAM_STR);
88         $sql_update->bindParam(3, $codigo, PDO::PARAM_INT);
89         $sql_update->execute();
90         $actualizado = 1;
91
92     }else if($id_repartidor != null){
93
94         // Preparar la consulta SQL de actualización
95         $sql_update = $con->prepare("UPDATE tb_pedido SET Estado_Pedido=? , Fecha_Entrega=?, Id_Repartidor=? WHERE Id_Pedido=?");
96
97         // Vincular parámetros y ejecutar la consulta

```



```

Admin > pedidos-admin.php
65 if (isset($_POST['accion']) && $_POST['accion'] == 'actualizar') {
66     $id_repartidor = $_POST['repartidor'];
67
68     // Vincular parámetros y ejecutar la consulta
69     $sql_update->bindParam(1, $estado, PDO::PARAM_INT);
70     $sql_update->bindParam(2, $fecha_entrega, PDO::PARAM_STR);
71     $sql_update->bindParam(3, $id_repartidor, PDO::PARAM_INT);
72     $sql_update->bindParam(4, $codigo, PDO::PARAM_INT);
73     $sql_update->execute();
74
75     //Actualizar la disponibilidad del repartidor
76     $sql_update_repartidor = $con->prepare("UPDATE tb_repartidor SET Estado_Repartidor = 0 WHERE Id_Repartidor = ?");
77     $sql_update_repartidor->bindParam(1, $id_repartidor, PDO::PARAM_INT);
78     $sql_update_repartidor->execute();
79     $actualizado = 1;
80
81
82     if($estado_valor == 2){
83         //Recuperar el id del repartidor de la tabla tb_pedido
84         $sql_repartidor = $con->prepare("SELECT Id_Repartidor FROM tb_pedido WHERE Id_Pedido = ?");
85         $sql_repartidor->bindParam(1, $id_repartidor, PDO::PARAM_INT);
86         $sql_repartidor->execute();
87         $id_repartidor = $sql_repartidor->fetchColumn();
88
89         // Preparar la consulta SQL de actualización
90         $sql_update_repartidor = $con->prepare("UPDATE tb_repartidor SET Estado_Repartidor = 1 WHERE Id_Repartidor = ?");
91         $sql_update_repartidor->bindParam(1, $id_repartidor, PDO::PARAM_INT);
92         $sql_update_repartidor->execute();
93
94
95     // Verificar si la actualización fue exitosa
96     if ($actualizado == 1) {
97         error_log("Entró al bloque de actualización de pedido. Código: $codigo");
98         // Obtener datos del cliente del pedido actualizado
99         $sql_cliente = $con->prepare("SELECT u.Correo, p.Nombre, p.Apellido
100             FROM tb_pedido pe
101             INNER JOIN tb_cliente c ON pe.Id_Cliente = c.Id_Cliente
102             INNER JOIN tb_persona p ON c.Id_Cliente = p.Id_Persona
103             INNER JOIN tb_usuario u ON u.Id_Usuario = p.Id_Persona
104             WHERE pe.Id_Pedido = ?");
105         $sql_cliente->execute($codigo);
106         $cliente = $sql_cliente->fetch(PDO::FETCH_ASSOC);
107         error_log("Datos del cliente: " . print_r($cliente, true));
108
109         // Solo si hay correo
110         if ($cliente && !empty($cliente['Correo'])) {
111             // Llama a correo.php usando file_get_contents o curl
112             $estado_texto = '';
113             switch ($estado) {
114                 case 0: $estado_texto = 'Pendiente'; break;
115                 case 1: $estado_texto = 'En proceso'; break;
116                 case 2: $estado_texto = 'Entregado'; break;
117                 case 3: $estado_texto = 'Cancelado'; break;
118             }
119             $correo_texto = "Tu pedido con código $codigo ha sido $estado_texto";
120             file_get_contents("correo.php");
121         }
122     }
123 }

```

```

index.php          create-account.php    login.php      JS validarLogin.js   catalogo.php   Carrito-float.php   ped
Admin > pedidos-admin.php
65 if (isset($_POST['accion']) && $_POST['accion'] === 'actualizar') {
66     if ($actualizado == 1) {
67         if ($cliente && !empty($cliente['Correo'])) {
68             switch ($estado) {
69                 case 3: $estado_texto = 'Cancelado'; break;
70                 default: $estado_texto = 'Actualizado'; break;
71             }
72             $mensaje = "El estado de su pedido ha sido actualizado a: $estado_texto.";
73             // Usando file_get_contents (requiere allow_url_fopen = On)
74             $postdata = http_build_query([
75                 'id_pedido' => $codigo, // o el nombre de tu variable de ID de pedido
76                 'nombre' => $cliente['Nombre'],
77                 'apellido' => $cliente['Apellido'],
78                 'email' => $cliente['Correo'],
79                 // 'telefono' => '', // Si tienes el teléfono, agrégalo aquí
80                 'tipo' => 'estado',
81                 'estado' => $estado_texto // Por ejemplo: "En proceso"
82             ]);
83             $opts = [
84                 'http' => [
85                     'method' => 'POST',
86                     'header' => 'Content-type: application/x-www-form-urlencoded',
87                     'content' => $postdata
88                 ]
89             ];
90             // Construye la URL completa a correo.php
91             $url = 'http://localhost/Cremas-y-Mil-Hojas-Front-End/correo.php';
92             $ch = curl_init($url);
93             curl_setopt($ch, CURLOPT_POST, 1);
94             curl_setopt($ch, CURLOPT_POSTFIELDS, $postdata);
95             curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
96             $response = curl_exec($ch);
97             if ($response === false) {
98                 error_log("Error al llamar a correo.php con cURL: " . curl_error($ch));
99             } else {
100                 error_log("Respuesta de correo.php (cURL): " . $response);
101             }
102             curl_close($ch);
103         }
104     }
105     header('Content-Type: application/json');
106     if ($actualizado == 1) {
107         echo json_encode(["status" => "success", "message" => "Pedido actualizado correctamente"]);
108         exit;
109     } else {
110         echo json_encode(["status" => "error", "message" => "Error al actualizar el pedido"]);
111         exit;
112     }
113 }

```

✓ Ejecución:

The screenshot shows a web application for managing inventory. The left sidebar has navigation links: Dashboard, Productos (selected), Pedidos, Clientes, and Repartidores. The main content area displays a table of products with columns: GO, IMAGEN, CATEGORÍA, NOMBRE, DESCRIPCIÓN, STOCK, PRECIO, DISPONIBILIDAD, ESTADO, and ACCIONES. The table contains four rows of data:

GO	IMAGEN	CATEGORÍA	NOMBRE	DESCRIPCIÓN	STOCK	PRECIO	DISPONIBILIDAD	ESTADO	ACCIONES
		Bolleria	Karamandunga	dulce crocante	0	S/ 50.00	Escaso	Activo	<button>Editar</button> <button>Eliminar</button>
		Especiales	Torta de Chocolate	Deliciosa torta de chocolate con crema	1	S/ 35.50	Escaso	Activo	<button>Editar</button> <button>Eliminar</button>
		Especiales	Tutifrutti	GAAA	5	S/ 20.00	Escaso	Inactivo	<button>Editar</button> <button>Eliminar</button>
		Dulces	Pai de chocolate	FFF	12	S/ 30.00	Escaso	Activo	<button>Editar</button> <button>Eliminar</button>

At the top, there are three filter sections: Filtrar por Categoría, Filtrar por Disponibilidad, and Filtrar por Estado. The status bar at the top right says "Bienvenido, Alvaro Ivan".

Dashboard de Productos									
Filtrar por Categoría: <input type="text" value="Buscar por categoría"/>									
Filtrar por Disponibilidad: <input type="text" value="Buscar por disponibilidad"/>									
Selecciona categorías					Selecciona disponibilidad			Selecciona estado	
CÓDIGO IMAGEN CATEGORÍA NOMBRE DESCRIPCIÓN STOCK PRECIO DISPONIBILIDAD ESTADO ACCIONES									
1		Bollería	Karamandunga	dulce crocante	0	S/ 50.00	Escaso	Activo	<button>Editar</button> <button>Eliminar</button>
2		Especiales	Torta de Chocolate	Deliciosa torta de chocolate con crema	1	S/ 35.50	Escaso	Activo	<button>Editar</button> <button>Eliminar</button>
3		Especiales	Tutifrutti	GAAA	5	S/ 20.00	Escaso	Inactivo	<button>Editar</button> <button>Eliminar</button>
4		Dulces	Pai de chocolate	FFF	11	S/ 30.00	Escaso	Activo	<button>Editar</button> <button>Eliminar</button>

- R.F 6: El sistema debe generar alertas de stock bajo.

✓ Código:

```
29
30     //Lógica de la alerta de stock bajo
31     $alerta_stock_bajo = false;
32     foreach ($productos as $prod) {
33         if ($prod['Disponibilidad'] == 0 || $prod['Disponibilidad'] == 2) {
34             $alerta_stock_bajo = true;
35             break;
36         }
37     }
38 }
39 }
```

```
880
881 <?php if ($alerta_stock_bajo): ?>
882
883 <script>
884     document.addEventListener('DOMContentLoaded', function () {
885         Swal.fire({
886             icon: 'warning',
887             title: 'Atención',
888
889             text: 'Algunos productos están escasos o agotados. Verifica el inventario.',
890
891             confirmButtonText: 'Entendido'
892         });
893     });
894
895     </script>
896 <?php endif; ?>
```

✓ Ejecución:

Bienvenido, Alvaro Ivan

✓ Ejecución:

Listado de Productos

Buscar por Código

Filtrar por Categoría:

Selecciona categorías

CÓDIGO IMAGEN CATEGORÍA PRECIO DISPONIBILIDAD ESTADO ACCIONES

CÓDIGO	IMAGEN	CATEGORÍA	PRECIO	DISPONIBILIDAD	ESTADO	ACCIONES			
1		B	S/ 50.00	Escaso	Activo	<button>Editar</button> <button>Eliminar</button>			
2		E	S/ 35.50	Escaso	Activo	<button>Editar</button> <button>Eliminar</button>			
3		Especiales	Tutifrutti	GAAA	5	S/ 20.00	Escaso	Inactivo	<button>Editar</button> <button>Eliminar</button>

Atención

Algunos productos están escasos o agotados.
Verifica el inventario.

Entendido

Anexo 9. Refactorización del software

En este apartado se realizó la refactorización total del software de nuestro proyecto. A continuación, se mostrarán los códigos y las capturas de un antes y un después de estos mismos para demostrar la diferencia que hubo.

❖ Refactorización del módulo Login y Registro

- Registro y Login

✓ Antes:

El código original primero asegura que la sesión esté iniciada. Si detecta un usuario soqueado, usa una estructura if-else anidada para redirigirlo: a Admin/admin-page.php si es 'Administrador', o a index.php en cualquier otro caso, terminando la ejecución tras la redirección.

```
create-account.php
1  <?php
2
3  if (session_status() == PHP_SESSION_NONE) {
4      session_start();
5  }
6
7  if (isset($_SESSION['usuario'])) {
8      // Si ya hay una sesión iniciada, redirige al usuario según su tipo
9      if (isset($_SESSION['tipo']) && $_SESSION['tipo'] == 'Administrador') {
10          header("Location: Admin/admin-page.php");
11          exit();
12      } else {
13          header("Location: index.php");
14          exit();
15      }
16  }
17
```

✓ Después:

La versión mejorada también inicia la sesión. Si hay un usuario logueado, simplifica la redirección: usa el operador ?? para obtener el tipo de usuario y un operador ternario para definir la URL (admin o index), redirigiendo de forma más compacta. Adicionalmente, incorpora la

inclusión de archivos de configuración y base de datos, mejorando la estructura del proyecto.

```
create-account.php
1  <?php
2
3  if (session_status() == PHP_SESSION_NONE) {
4  | | session_start();
5  }
6
7  if (isset($_SESSION['usuario'])) {
8  $tipo = $_SESSION['tipo'] ?? '';
9  $url = ($tipo === 'Administrador') ? "Admin/admin-page.php" : "index.php";
10 header("Location: $url");
11 exit();
12 }
13
14 //
15 require 'Config/config.php';
16 require 'database.php';
17 //Se hace la conexión a la base de datos
```

- Registro encriptación de contraseña

✓ **Antes:**

El código recupera todos los datos enviados por el formulario, incluyendo la contraseña, y los guarda en variables individuales después de eliminar espacios en blanco con trim(). Específicamente, la contraseña se almacena en la variable \$contraseña (línea 29). Solo después de que todos los datos han sido recuperados y asignados a variables, se procede a encriptar la contraseña. La encriptación se realiza en la línea 39, utilizando password_hash(\$contraseña, PASSWORD_DEFAULT), y el resultado se guarda en \$contraseña_encriptada. Esto significa que la contraseña en texto plano existe en una variable (\$contraseña) antes de ser encriptada.

```
21
22 //Condicional para que se agregar un usuario cuando se envia los datos del formulario
23 if (isset($_POST['accion']) && $_POST['accion'] === 'agregarU') [
24     //Recuperar los datos del formulario
25     //Trim para eliminar espacios en blanco al inicio y al final
26     $nombre = trim($_POST["nombre"]);
27     $apellido = trim($_POST["apellido"]);
28     $telefono = trim($_POST["telefono"]);
29     $correo = trim($_POST["email"]);
30     $contraseña = trim($_POST["password"]);
31     $dni = trim($_POST["dni"]);
32     $direccion = trim($_POST["direccion"]);
33     $distrito = trim($_POST["distrito"]);
34     $genero = trim($_POST["genero"]);
35
36     $genero_valor = (int) $genero; // Convertir a entero para evitar problemas de tipo
37     //Validaciones de los campos del formulario
38     //Encriptar la contraseña
39     $contraseña_encriptada = password_hash($contraseña, PASSWORD_DEFAULT);
40 ]
```

✓ Después:

Se optimiza el proceso de encriptación y la gestión de la contraseña. En lugar de almacenar la contraseña en texto plano en una variable temporal (\$contraseña) y luego encriptarla, la encriptación se realiza directamente sobre el valor recibido del formulario. La línea 30 ahora lee \$_POST["password"] y lo encripta inmediatamente usando password_hash(trim(\$_POST["password"]), PASSWORD_DEFAULT), asignando el resultado directamente a \$contraseña_encriptada. Esto elimina la necesidad de la variable \$contraseña en texto plano, reduciendo la exposición de datos sensibles.

```
22 //Condicional para que se agregar un usuario cuando se envia los datos del formulario
23 if (isset($_POST['accion']) && $_POST['accion'] === 'agregarU') [
24     //Recuperar los datos del formulario
25     //Trim para eliminar espacios en blanco al inicio y al final
26     $nombre = trim($_POST["nombre"]);
27     $apellido = trim($_POST["apellido"]);
28     $telefono = trim($_POST["telefono"]);
29     $correo = trim($_POST["email"]);
30     $contrasena_encriptada = password_hash(trim($_POST["password"]), PASSWORD_DEFAULT);
31     $dni = trim($_POST["dni"]);
32     $direccion = trim($_POST["direccion"]);
33     $distrito = trim($_POST["distrito"]);
34     $genero = trim($_POST["genero"]);
35
36     $genero_valor = (int) $genero; // Convertir a entero para evitar problemas de tipo
37     //Validaciones de los campos del formulario
38 
```

❖ Refactorización del módulo gestión de pedidos

- Catálogo

✓ Antes:

El código original inicia la sesión y luego presenta dos bloques separados para verificar si el usuario está logueado (isset(\$_SESSION['usuario'])). El primer bloque maneja la redirección de administradores, enviándolos a una página específica. Si el usuario no es un administrador o no fue redirigido previamente, un segundo bloque idéntico vuelve a verificar la sesión para luego consultar la base de datos y obtener los datos del usuario, lo que resulta en una verificación redundante y un flujo de código menos eficiente.

```
catalogo.php
1 </php
2 session_start();
3
4 require 'Config/config.php';
5 require 'database.php';
6 $db = new Database();
7 $con = $db->conectar();
8
9 if (isset($_SESSION['usuario'])) {
10
11     if(isset($_SESSION['tipo']) && $_SESSION['tipo'] == 'Administrador'){
12         // Si ya hay una sesión iniciada, redirige al usuario según su tipo
13         header("Location: Admin/admin-page.php");
14         exit();
15     }
16 }
17 if(isset($_SESSION['usuario'])){
18     // Consulta SQL para seleccionar los datos del usuario utilizando su ID al LOGUEARSE
19     $sql = $con->prepare("SELECT Nombre, Apellido, DNI, Telefono FROM tb_persona WHERE Id_Persona = :id");
20     $sql->bindParam(':id', $_SESSION['id'] , PDO::PARAM_INT);
21     $sql->setFetchMode(PDO::FETCH_ASSOC); // Establecer el modo de recuperación de datos
22     $sql->execute();
23     $datos_persona = $sql->fetch(PDO::FETCH_ASSOC);
24
25     // Asignar los datos del usuario a variables individuales
26     $nombre = $datos_persona['Nombre'];
27 }
28 ?>
```

✓ Despues:

La versión refactorizada mejora el flujo de la aplicación iniciando la sesión, cargando la configuración y estableciendo la conexión a la base de datos al inicio. La verificación de la sesión (isset(\$_SESSION['usuario'])) ahora se realiza en un único bloque. Dentro de este, primero se redirige a los administradores. Si el usuario está logueado pero no es administrador, el código continúa para ejecutar la consulta a la base de datos y cargar los datos del usuario, eliminando la verificación de sesión redundante y haciendo el proceso más directo y organizado.

```
catalogo.php
1 <?php
2 session_start();
3
4 require 'Config/config.php';
5 require 'database.php';
6
7 $db = new Database();
8 $con = $db->conectar();
9
10 if (isset($_SESSION['usuario'])) {
11     if (isset($_SESSION['tipo']) && $_SESSION['tipo'] === 'Administrador') {
12         header("Location: Admin/admin-page.php");
13         exit();
14     }
15
16     // Si no es administrador, obtener sus datos
17     $sql = $con->prepare("SELECT Nombre, Apellido, DNI, Telefono FROM tb_persona WHERE Id_Persona = :id");
18     $sql->bindParam(':id', $_SESSION['id'], PDO::PARAM_INT);
19     $sql->setFetchMode(PDO::FETCH_ASSOC);
20     $sql->execute();
21     $datos_persona = $sql->fetch();
22
23     // Asignar los datos del usuario a variables
24     $nombre = $datos_persona['Nombre'];
25 }
26 ?>
27
```

- Lista de pedidos

✓ **Antes:**

Presenta una ineficiencia notable al verificar el estado de la sesión del usuario (isset(\$_SESSION['usuario'])) hasta tres veces en puntos distintos del script. Esta redundancia implica que la lógica para redirigir a administradores, cargar datos del usuario y manejar operaciones del carrito se ejecuta de forma dispersa y repetitiva, haciendo el código menos legible y más complejo de mantener, ya que el estado de la sesión se evalúa innecesariamente en múltiples ocasiones.

```

listadoPedidos.php
1 <?php
2 session_start();
3 require 'Config/config.php';
4 require 'database.php';
5 $db = new Database();
6 $con = $db->conectar();
7 //Si es que aún no has cerrado sesión te va enviar la página correspondiente, aunque cierres la página
8 if (!isset($_SESSION['usuario'])) {
9     // Si ya hay una sesión iniciada, redirige al usuario según su tipo
10    if (!isset($_SESSION['tipo']) || $_SESSION['tipo'] == 'admin') {
11        header("Location: Admin/admin-page.php");
12        exit();
13    }
14 }
15 if(isset($_SESSION['usuario'])){
16     // Consulta SQL para seleccionar los datos del usuario utilizando su ID al LOGUEARSE
17     $sql = $con->prepare("SELECT Nombre, Apellido, DNI, Telefono FROM tb_persona WHERE Id_Persona = :id");
18     $sql->bindParam(':id', $_SESSION['id'], PDO::PARAM_INT);
19     $sql->setFetchMode(PDO::FETCH_ASSOC); // Establecer el modo de recuperación de datos
20     $sql->execute();
21     $datos_persona = $sql->fetch(PDO::FETCH_ASSOC);
22     // Asignar los datos del usuario a variables individuales
23     $Nombre = $datos_persona['Nombre'];
24 }
25 $pedidos_completos = [];
26 if (!isset($_SESSION['usuario'])) {
27
28     $sql = $con->prepare("SELECT * FROM tb_pedido WHERE Id_Cliente = :id");
29     $sql->bindParam(':id', $_SESSION['id'], PDO::PARAM_INT);
30     $sql->execute();
31     $pedidos = $sql->fetchAll(PDO::FETCH_ASSOC);
32
33     foreach ($pedidos as $pedido) {
34         $pedido_id = $pedido['Id_Pedido'];
35         $fecha = $pedido['Fecha_Pedido'];
36         $estado_num = $pedido['Estado_Pedido'];
37         $costo_envio = $pedido['Costo_Envio'];
38         $total_pedido = $pedido['Total_Pedido'];
39         $estado_texto = '';
40
41         switch ($estado_num) {
42             case 0: $estado_texto = 'Pendiente'; break;
43             case 1: $estado_texto = 'En proceso'; break;
44             case 2: $estado_texto = 'Entregado'; break;
45             case 3: $estado_texto = 'Cancelado'; break;
46         }
47
48         // Obtener productos de ese pedido
49         $sql_productos = $con->prepare("
50             SELECT dp.Cantidad_Pedido, dp.Precio_Venta, dp.Descuento, pr.Nombre_Producto, pr.Imagen
51             FROM tb_detalle_pedido dp
52             INNER JOIN tb_producto pr ON dp.Id_Producto = pr.Id_Producto
53             WHERE dp.Id_pedido = :pedido_id
54         ");
55         $sql_productos->bindParam(':pedido_id', $pedido_id, PDO::PARAM_INT);
56         $sql_productos->execute();
57         $productos = $sql_productos->fetchAll(PDO::FETCH_ASSOC);
58
59         // Calcular subtotal
60         $total = 0;
61         foreach ($productos as $producto) {
62             $producto['Subtotal'] = $producto['Cantidad_Pedido'] * $producto['Precio_Venta'];
63             $total += $producto['Subtotal'];
64         }
65
66         // Agregar al arreglo final
67         $pedidos_completos[] = [
68             'id' => $pedido_id,
69             'fecha' => $fecha,
70             'estado' => $estado_texto,
71             'estado_num' => $estado_num,
72             'total' => $total_pedido,
73             'productos' => $productos,
74             'costo_envio' => $costo_envio
75         ];
76     }
77 }

```

✓ Después:

La versión optimizada de "LISTA DE PEDIDOS" centraliza y simplifica el manejo de la sesión y la lógica asociada. Después de iniciar la sesión e incluir las configuraciones esenciales, se realiza una única verificación `if (isset($_SESSION['usuario']))`. Dentro de este bloque consolidado, se maneja primero la redirección para administradores y

luego se procede a cargar los datos del usuario y a gestionar todas las operaciones del carrito de compras. Este enfoque unificado elimina la redundancia, mejora significativamente la eficiencia y la claridad del código, facilitando su comprensión y mantenimiento.

```

1  DataPedido.php
2  {
3      $db = new Database();
4      $con = $db->conectar();
5      $pedidos_completos = [];
6
7      if (isset($_SESSION['usuario'])) {
8          if (isset($_SESSION['tipo']) && $_SESSION['tipo'] === 'admin') {
9              header("Location: Admin/admin-page.php");
10             exit();
11         }
12
13         $sql = $con->prepare("SELECT Nombre, Apellido, DNI, Telefono FROM tb_persona WHERE Id_Persona = :id");
14         $sql->bindParam(":id", $_SESSION['id'], PDO::PARAM_INT);
15         $sql->execute();
16         $datos_persona = $sql->fetch(PDO::FETCH_ASSOC);
17         $Nombre = $datos_persona['Nombre'];
18
19         $sql = $con->prepare("SELECT * FROM tb_pedido WHERE Id_Cliente = :id");
20         $sql->bindParam(":id", $_SESSION['id'], PDO::PARAM_INT);
21         $sql->execute();
22         $pedidos = $sql->fetchAll(PDO::FETCH_ASSOC);
23
24         foreach ($pedidos as $pedido) {
25             $pedido_id = $pedido['Id_Pedido'];
26             $estado_texto = match ((int) $pedido['Estado_Pedido']) {
27                 1 => 'En proceso',
28                 2 => 'Entregado',
29                 3 => 'Cancelado',
30                 default => 'Desconocido'
31             };
32
33             $sql_productos = $con->prepare(`
34                 SELECT dp.Cantidad_Pedido, dp.Precio_Venta, dp.Descuento, pr.Nombre_Producto, pr.Imagen
35                 FROM tb_detalle_pedido dp
36                 INNER JOIN tb_producto pr ON dp.Id_Producto = pr.Id_Producto
37                 WHERE dp.Id_Pedido = $pedido_id
38             `);
39             $sql_productos->bindParam(':pedido_id', $pedido_id, PDO::PARAM_INT);
40             $sql_productos->execute();
41             $productos = $sql_productos->fetchAll(PDO::FETCH_ASSOC);
42
43             $total = 0;
44             foreach ($productos as $producto) {
45                 $producto['Subtotal'] = $producto['Cantidad_Pedido'] * $producto['Precio_Venta'];
46                 $total += $producto['Subtotal'];
47             }
48
49             $pedidos_completos[] = [
50                 'id' => $pedido_id,
51                 'fecha' => $pedido['Fecha_Pedido'],
52                 'estado' => $estado_texto,
53                 'total' => $pedido['Total_Pedido'],
54                 'productos' => $productos,
55                 'costo_envio' => $pedido['Costo_Envio']
56             ];
57         }
58     }
59 }
60
61
62 }
63

```

❖ Refactorización del módulo control de inventario

- Producto - Admin

✓ Antes:

El código antes mostraba la lógica para determinar la disponibilidad de un producto directamente incrustada en el flujo principal, utilizando una serie de condicionales if-else if-else para asignar un estado de 'agotado', 'escaso' o 'disponible' basado en el stock actual y mínimo. Esta implementación, aunque funcional, dejaba la lógica de decisión repetitiva y no modular, lo que dificultaría su reutilización o actualización en otros puntos del sistema.

```
109     $precio = $_POST['precioEditarP'];
110     $categoria = $_POST['categoriaEditarP'];
111     $descripcion = $_POST['descripcionEditarP'];
112     $disponibilidad = $_POST['disponibilidadP'];
113     $estado = $_POST['estadoEditarP'];
114
115     $stock_valor = (float) $stock;
116     $stockMin_valor = (float) $stockMin;
117
118     if ($stock_valor == 0) {
119         $disponibilidad = 0; //Agotado
120     } else if ($stock_valor <= $stockMin_valor && $stockMin_valor != 0) {
121         $disponibilidad = 2; //Escaso
122     } else {
123         $disponibilidad = 1; //Disponible
124     }
125
126     if ($estado == '1') {
127         $estado = 1; // Convertir a 1 si está activo
128     } else {
129         $estado = 0; // Convertir a 0 si no está activo
130     }
131
132     // Consulta para obtener la imagen actual del producto
133     $sql_get_image = $con->prepare("SELECT Imagen FROM tb_producto WHERE Id_Producto = ?");
134     $sql_get_image->execute([$codigo]);
```

✓ Despues:

En contraste, el código después mejora significativamente este aspecto al encapsular toda la lógica de determinación de disponibilidad dentro de una función dedicada llamada calcularDisponibilidad. Esta función, que recibe el stock y el stock mínimo como argumentos, devuelve el estado de disponibilidad, permitiendo que el código principal simplemente llame a esta función. Esto no solo hace el código más limpio y legible, sino que también promueve la reutilización, facilita el mantenimiento (si las reglas de disponibilidad cambian, solo se modifica la función) y reduce la posibilidad de errores al evitar la duplicación de código.

```
115     $stock_min_valor = (float) $stockMin;
116     $stockMin_valor = (float) $stockMin;
117
118     function calcularDisponibilidad($stock, $stockMin) {
119         if ($stock == 0) return 0;
120         if ($stock <= $stockMin && $stockMin != 0) return 2;
121         return 1;
122     }
123
124     $disponibilidad = calcularDisponibilidad((float)$stock, (float)$stockMin);
125
126
127     if ($estado == '1') {
128         $estado = 1; // Convertir a 1 si está activo
```

- Pedidos - Admin

✓ Antes:

El código original para la administración de pedidos (PEDIDOS-ADMIN) presenta una redundancia en el manejo de la actualización de pedidos basada en el \$id_repartidor. Contiene dos bloques if casi idénticos, diferenciados únicamente por la inclusión o exclusión del campo Id_Repartidor en la consulta SQL y en los parámetros a vincular. Si \$id_repartidor es null, se usa una consulta; si no es null, se usa otra consulta que incluye el Id_Repartidor. Esta duplicación de código hace que el script sea menos eficiente, más largo y más propenso a errores si se necesitan hacer cambios en la lógica de actualización, ya que se debería modificar en dos lugares diferentes.

```
75
76
77     if($id_repartidor == null){
78
79         // Preparar la consulta SQL de actualización
80         $sql_update = $con->prepare("UPDATE tb_pedido SET Estado_Pedido=? , Fecha_Entrega=? WHERE Id_Pedido=?");
81
82         // Vincular parámetros y ejecutar la consulta
83         $sql_update->bindParam(1, $estado, PDO::PARAM_INT);
84         $sql_update->bindParam(2, $fecha_entrega, PDO::PARAM_STR);
85         $sql_update->bindParam(3, $codigo, PDO::PARAM_INT);
86         $sql_update->execute();
87         $actualizado = 1;
88     }else if($id_repartidor != null){
89
90         // Preparar la consulta SQL de actualización
91         $sql_update = $con->prepare("UPDATE tb_pedido SET Estado_Pedido=? , Fecha_Entrega=? , Id_Repartidor=? WHERE Id_Pedido=?");
92
93         // Vincular parámetros y ejecutar la consulta
94         $sql_update->bindParam(1, $estado, PDO::PARAM_INT);
95         $sql_update->bindParam(2, $fecha_entrega, PDO::PARAM_STR);
96         $sql_update->bindParam(3, $id_repartidor, PDO::PARAM_INT);
97         $sql_update->bindParam(4, $codigo, PDO::PARAM_INT);
98         $sql_update->execute();
99
100        //Actualizar la disponibilidad del repartidor
101        $sql_update_repartidor = $con->prepare("UPDATE tb_repartidor SET Estado_Repartidor = 0 WHERE Id_Repartidor = ?");
102        $sql_update_repartidor->bindParam(1, $id_repartidor, PDO::PARAM_INT);
103        $sql_update_repartidor->execute();
104        $actualizado = 1;
105    }
106}
```

✓ Después:

La versión optimizada del código elimina la redundancia al consolidar la lógica de actualización en un solo bloque. Utiliza la función empty(\$id_repartidor) para determinar si el repartidor debe ser incluido en la actualización. La clave de la mejora es que la sentencia SQL para la actualización del pedido (UPDATE tb_pedido SET Estado_Pedido=?, Fecha_Entrega=?) ya incluye el Id_Repartidor en el WHERE Id_Pedido=?, y el valor de \$id_repartidor se pasa como un parámetro en la vinculación. Esto significa que la misma consulta y secuencia de parámetros pueden manejar ambos casos (con o sin repartidor asignado) de manera más eficiente, evitando la duplicación del código de consulta y vinculación de parámetros.

```
75
76     if(empty($id_repartidor)){
77         // Preparan la consulta SQL de actualización
78         $sql_update = $con->prepare("UPDATE tb_pedido SET Estado_Pedido=? , Fecha_Entrega=? WHERE Id_Pedido=?");
79
80         // Vincular parámetros y ejecutar la consulta
81         $sql_update->bindParam(1, $estado, PDO::PARAM_INT);
82         $sql_update->bindParam(2, $fecha_entrega, PDO::PARAM_STR);
83         $sql_update->bindParam(3, $codigo, PDO::PARAM_INT);
84         $sql_update->execute();
85         $actualizado = 1;
86     }else{
87         $sql_update = $con->prepare("UPDATE tb_pedido SET Estado_Pedido=? , Fecha_Entrega=? , Id_Repartidor=? WHERE Id_Pedido=?");
88         // Vincular parámetros y ejecutar la consulta
89         $sql_update->bindParam(1, $estado, PDO::PARAM_INT);
90         $sql_update->bindParam(2, $fecha_entrega, PDO::PARAM_STR);
91         $sql_update->bindParam(3, $id_repartidor, PDO::PARAM_INT);
92         $sql_update->bindParam(4, $codigo, PDO::PARAM_INT);
93         $sql_update->execute();
94         $sql_update->execute();
95         //Actualizar la disponibilidad del repartidor
96         $sql_update_repartidor = $con->prepare("UPDATE tb_repartidor SET Estado_Repartidor = 0 WHERE Id_Repartidor = ?");
97         $sql_update_repartidor->bindParam(1, $id_repartidor, PDO::PARAM_INT);
98         $sql_update_repartidor->execute();
99         $actualizado = 1;
100    }
```

Anexo 10. Acta de aprobación



Carta de Aprobación

Yo **ZILVA RAMIREZ JOSE LUIS**, gerente general de la empresa **Crema y Milhojas S.A.C.** que suscribe:

Hacer constar.

El estudiante **ARROYO CARRASCO ÁLVARO IVÁN** de la escuela de ingeniería de sistemas computacionales de la Universidad Privada del Norte realizo un **Sistema web para la optimización en la atención al cliente y control de inventario**, teniendo así una mejor rapidez en la atención y control de los productos.

Se expide la presente constancia para los fines pertinentes.

Lima, 19 de mayo del 2025

Atentamente.

A handwritten signature in black ink, appearing to read "Zilva Malca Jose Luis".

ZILVA MALCA JOSE LUIS

Gerente General

Anexo 11. Actas de reuniones y planificaciones

ACTA DE REUNIÓN PLANIFICACIÓN DE SPRINT N° 1

Fecha : 27/04/25

Scrum Master : Arroyo Carrasco Álvaro Iván

Product Owner : Zilva Ramirez Jose Luis

Mediante la presente acta se valida y se da conformidad de que el equipo Scrum determinó las historias de usuario para el Sprint 1 para el desarrollo del proyecto **Sistema Web Para La Optimización En La Atención Al Cliente Y Control De Inventario** en la empresa **Crema y Milhojas S.A.C.** Acordando satisfactoriamente los objetivos del Sprint 1, como también los elementos de la Pila de Producto (Historias) que contiene el sprint mencionado.

Dentro del Sprint 1 se determinó lo siguiente:

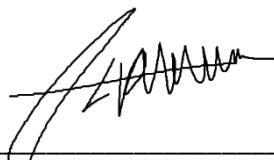
SPRINT	OBJETIVO	HISTORIAS
1	El sistema debe registrar, editar y eliminar productos del inventario.	(H4) Registrar productos
	El sistema debe actualizar el stock automáticamente cuando se realiza un pedido.	(H5) Actualización automática
	El sistema debe generar alertas de stock bajo.	(H6) Alerta de Stock

Firma en señal de la conformidad



Zilva Ramirez Jose Luis

Gerente General



Arroyo Carrasco Álvaro Iván

ACTA DE ENTREGA DEL SPRINT N° 1

Fecha : 01/05/25

Scrum Master : Arroyo Carrasco Álvaro Iván

Product Owner : Zilva Ramirez Jose Luis

Mediante la presente acta se valida y se da conformidad de que la entrega del sprint 1 presenta la **(H4) Registrar producto** del sistema con las funcionalidades ya predeterminadas por el Product Owner en la acta de reunión de planificación del sprint 1 donde se detalla las historias de usuario y objetivos, pero también se dieron a conocer nuevos cambios, mejoras y agregar detalles a los objetivos **(H5) Actualización automática y (H6) Alerta de Stock**; elaboradas las especificaciones por el equipo Scrum y Scrum master se da la aprobación del Sprint 1, donde se decide de manera unánime aprobar el sprint mencionado donde se presentaron los requerimientos para el proyecto **Sistema Web Para La Optimización En La Atención Al Cliente Y Control De Inventario.**

Firma en señal de conformidad



Zilva Ramirez Jose Luis

Gerente General



Arroyo Carrasco Álvaro Iván

ACTA DE REUNIÓN PLANIFICACIÓN DE SPRINT N° 1

Fecha : 04/05/25

Scrum Master : Arroyo Carrasco Álvaro Iván

Product Owner : Zilva Ramirez Jose Luis

Mediante la presente acta se valida y se da conformidad de que el equipo Scrum determinó los nuevos cambios de las historias de usuario 5 y 6 para el Sprint 1 para el desarrollo del proyecto **Sistema Web Para La Optimización En La Atención Al Cliente Y Control De Inventario** en la empresa **Crema y Milhojas S.A.C.** Acordando satisfactoriamente los objetivos del Sprint 1, como también los elementos de la Pila de Producto (Historias) que contiene el sprint mencionado.

Dentro del Sprint 1 se determinó lo siguiente:

SPRINT	OBJETIVO	HISTORIAS
1	El sistema debe registrar, editar y eliminar productos del inventario.	(H4) Registrar productos
	El sistema debe actualizar el stock automáticamente cuando se realiza un pedido.	(H5) Actualización automática
	El sistema debe generar alertas de stock bajo.	(H6) Alerta de Stock

Firma en señal de la conformidad

Zilva Ramirez Jose Luis

Gerente General

Arroyo Carrasco Álvaro Iván

ACTA DE ENTREGA DEL SPRINT N° 1

Fecha : 08/05/25

Scrum Master : Arroyo Carrasco Álvaro Iván

Product Owner : Zilva Ramirez Jose Luis

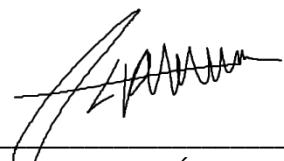
Mediante la presente acta se valida y se da conformidad de que la entrega del sprint 1 presenta la **(H4) Registrar productos , (H5) Actualización automática y (H6) Alerta de Stock** del objetivo del sistema con las funcionalidades ya predeterminadas por el Product Owner en la acta de reunión de planificación del sprint 1 donde se detalla las historias de usuario y objetivos; elaboradas las especificaciones por el equipo Scrum y Scrum master se da la aprobación del Sprint 1, donde se decide de manera unánime aprobar el sprint mencionado donde se presentaron los requerimientos para el proyecto **Sistema Web Para La Optimización En La Atención Al Cliente Y Control De Inventario.**

Firma en señal de conformidad



Zilva Ramirez Jose Luis

Gerente General



Arroyo Carrasco Álvaro Iván

ACTA DE REUNIÓN PLANIFICACIÓN DE SPRINT N° 2

Fecha : 11/05/25

Scrum Master : Arroyo Carrasco Álvaro Iván

Product Owner : Zilva Ramirez Jose Luis

Mediante la presente acta se valida y se da conformidad de que el equipo Scrum determinó las historias de usuario para el Sprint 2 para el desarrollo del proyecto **Sistema Web Para La Optimización En La Atención Al Cliente Y Control De Inventario** en la empresa **Crema y Milhojas S.A.C.** Acordando satisfactoriamente los objetivos del Sprint 2, como también los elementos de la Pila de Producto (Historias) que contiene el sprint mencionado.

Dentro del Sprint 2 se determinó lo siguiente:

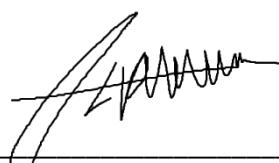
SPRINT	OBJETIVO	HISTORIAS
2	El sistema debe permitir a los clientes realizar pedidos en línea.	(H1) Realizar pedido
	El sistema debe mostrar en tiempo real la disponibilidad de productos.	(H2) Ver disponibilidad
	El sistema debe registrar los pedidos y asignarles un estado (pendiente, en proceso, entregado).	(H3) Estado del pedido

Firma en señal de la conformidad



Zilva Ramirez Jose Luis

Gerente General



Arroyo Carrasco Álvaro Iván

ACTA DE ENTREGA DEL SPRINT N° 2

Fecha : 15/05/25

Scrum Master : Arroyo Carrasco Álvaro Iván

Product Owner : Zilva Ramirez Jose Luis

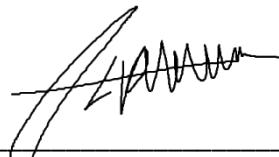
Mediante la presente acta se valida y se da conformidad de que la entrega del sprint 2 presenta el **(H1) Realizar pedido** del sistema con las funcionalidades ya predeterminadas por el Product Owner en la acta de reunión de planificación del sprint 2 donde se detalla las historias de usuario y objetivos, pero también se dieron a conocer nuevos cambios, mejoras y agregar detalles a los objetivos **(H2) Ver disponibilidad y (H3) Estado del pedido**; elaboradas las especificaciones por el equipo Scrum y Scrum master se da la aprobación del Sprint 2, donde se decide de manera unánime aprobar el sprint mencionado donde se presentaron los requerimientos para el proyecto **Sistema Web Para La Optimización En La Atención Al Cliente Y Control De Inventario**.

Firma en señal de conformidad



Zilva Ramirez Jose Luis

Gerente General



Arroyo Carrasco Álvaro Iván

ACTA DE REUNIÓN PLANIFICACIÓN DE SPRINT N° 2

Fecha : 18/05/25

Scrum Master : Arroyo Carrasco Álvaro Iván

Product Owner : Zilva Ramirez Jose Luis

Mediante la presente acta se valida y se da conformidad de que el equipo Scrum determinó los nuevos cambios de las historias de usuario 2 y 3 para el Sprint 2 para el desarrollo del proyecto **Sistema Web Para La Optimización En La Atención Al Cliente Y Control De Inventario** en la empresa **Crema y Milhojas S.A.C.** Acordando satisfactoriamente los objetivos del Sprint 2, como también los elementos de la Pila de Producto (Historias) que contiene el sprint mencionado.

Dentro del Sprint 2 se determinó lo siguiente:

SPRINT	OBJETIVO	HISTORIAS
2	El sistema debe permitir a los clientes realizar pedidos en línea.	(H1) Realizar pedido
	El sistema debe mostrar en tiempo real la disponibilidad de productos.	(H2) Ver disponibilidad
	El sistema debe registrar los pedidos y asignarles un estado (pendiente, en proceso, entregado).	(H3) Estado del pedido

Firma en señal de la conformidad



Zilva Ramirez Jose Luis

Gerente General



Arroyo Carrasco Álvaro Iván

ACTA DE ENTREGA DEL SPRINT N° 2

Fecha : 22/05/25

Scrum Master : Arroyo Carrasco Álvaro Iván

Product Owner : Zilva Ramirez Jose Luis

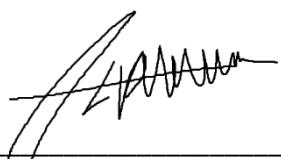
Mediante la presente acta se valida y se da conformidad de que la entrega del sprint 2 presenta el **(H1) Realizar pedido, (H2) Ver disponibilidad y (H3) Estado del pedido** del objetivo del sistema con las funcionalidades ya predeterminadas por el Product Owner en la acta de reunión de planificación del sprint 2 donde se detalla las historias de usuario y objetivos; elaboradas las especificaciones por el equipo Scrum y Scrum master se da la aprobación del Sprint 2, donde se decide de manera unánime aprobar el sprint mencionado donde se presentaron los requerimientos para el proyecto **Sistema Web Para La Optimización En La Atención Al Cliente Y Control De Inventario.**

Firma en señal de conformidad



Zilva Ramirez Jose Luis

Gerente General



Arroyo Carrasco Álvaro Iván

ACTA DE REUNIÓN PLANIFICACIÓN DE SPRINT N° 3

Fecha : 25/05/25

Scrum Master : Arroyo Carrasco Álvaro Iván

Product Owner : Zilva Ramirez Jose Luis

Mediante la presente acta se valida y se da conformidad de que el equipo Scrum determinó las historias de usuario para el Sprint 3 para el desarrollo del proyecto **Sistema Web Para La Optimización En La Atención Al Cliente Y Control De Inventario** en la empresa **Crema y Milhojas S.A.C.** Acordando satisfactoriamente los objetivos del Sprint 3, como también los elementos de la Pila de Producto (Historias) que contiene el sprint mencionado.

Dentro del Sprint 3 se determinó lo siguiente:

SPRINT	OBJETIVO	HISTORIAS
3	El sistema debe permitir a los clientes enviar consultas o sugerencias.	(H7) Enviar consulta
	El sistema debe enviar notificaciones sobre el estado del pedido.	(H8) Notificaciones
	El sistema debe mostrar mensajes automáticos de confirmación.	(H9) Confirmación automática

Firma en señal de la conformidad



Zilva Ramirez Jose Luis

Gerente General



Arroyo Carrasco Álvaro Iván

ACTA DE ENTREGA DEL SPRINT N° 3

Fecha : 29/05/25

Scrum Master : Arroyo Carrasco Álvaro Iván

Product Owner : Zilva Ramirez Jose Luis

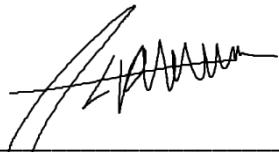
Mediante la presente acta se valida y se da conformidad de que la entrega del sprint 3 presenta el **(H7) Enviar consulta, (H8) Notificaciones y (H9) Confirmación automática** del sistema con las funcionalidades ya predeterminadas por el Product Owner en la acta de reunión de planificación del sprint 3 donde se detalla las historias de usuario y objetivos; elaboradas las especificaciones por el equipo Scrum y Scrum master se da la aprobación del Sprint 3, donde se decide de manera unánime aprobar el sprint mencionado donde se presentaron los requerimientos para el proyecto **Sistema Web Para La Optimización En La Atención Al Cliente Y Control De Inventario.**

Firma en señal de conformidad



Zilva Ramirez Jose Luis

Gerente General



Arroyo Carrasco Álvaro Iván

ACTA DE REUNIÓN PLANIFICACIÓN DE SPRINT N° 4

Fecha : 01/06/25

Scrum Master : Arroyo Carrasco Álvaro Iván

Product Owner : Zilva Ramirez Jose Luis

Mediante la presente acta se valida y se da conformidad de que el equipo Scrum determinó las historias de usuario para el Sprint 4 para el desarrollo del proyecto **Sistema Web Para La Optimización En La Atención Al Cliente Y Control De Inventario** en la empresa **Crema y Milhojas S.A.C.** Acordando satisfactoriamente los objetivos del Sprint 4, como también los elementos de la Pila de Producto (Historias) que contiene el sprint mencionado.

Dentro del Sprint 4 se determinó lo siguiente:

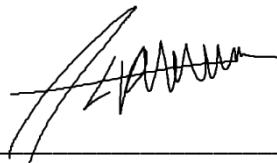
SPRINT	OBJETIVO	HISTORIAS
4	El sistema debe generar reportes sobre productos más vendidos.	(H10) Reporte de ventas
	El sistema debe exportar los datos a Excel o PDF.	(H11) Exportación

Firma en señal de la conformidad



Zilva Ramirez Jose Luis

Gerente General



Arroyo Carrasco Álvaro Iván

ACTA DE ENTREGA DEL SPRINT N° 4

Fecha : 05/06/25

Scrum Master : Arroyo Carrasco Álvaro Iván

Product Owner : Zilva Ramirez Jose Luis

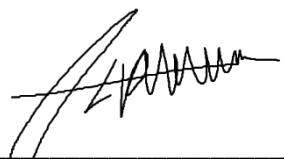
Mediante la presente acta se valida y se da conformidad de que la entrega del sprint 4 presenta el **(H10) Reporte de ventas y (H11) Exportación** del sistema con las funcionalidades ya predeterminadas por el Product Owner en la acta de reunión de planificación del sprint 4 donde se detalla las historias de usuario y objetivos; elaboradas las especificaciones por el equipo Scrum y Scrum master se da la aprobación del Sprint 4, donde se decide de manera unánime aprobar el sprint mencionado donde se presentaron los requerimientos para el proyecto **Sistema Web Para La Optimización En La Atención Al Cliente Y Control De Inventario**.

Firma en señal de conformidad



Zilva Ramirez Jose Luis

Gerente General



Arroyo Carrasco Álvaro Iván

ACTA DE REUNIÓN PLANIFICACIÓN DE SPRINT N° 5

Fecha : 08/06/25

Scrum Master : Arroyo Carrasco Álvaro Iván

Product Owner : Zilva Ramirez Jose Luis

Mediante la presente acta se valida y se da conformidad de que el equipo Scrum determinó las historias de usuario para el Sprint 5 para el desarrollo del proyecto **Sistema Web Para La Optimización En La Atención Al Cliente Y Control De Inventario** en la empresa **Crema y Milhojas S.A.C.** Acordando satisfactoriamente los objetivos del Sprint 5, como también los elementos de la Pila de Producto (Historias) que contiene el sprint mencionado.

Dentro del Sprint 5 se determinó lo siguiente:

SPRINT	OBJETIVO	HISTORIAS
5	El sistema debe contar con apartado donde se puede registrar cuentas de tipo cliente.	(H12) Registro de cuenta cliente
	El sistema debe contar con inicio de sesión para clientes.	(H13) Inicio de sesión cliente
	El sistema debe contar con inicio de sesión para administrador.	(H14) Inicio de sesión Administrador

Firma en señal de la conformidad

Zilva Ramirez Jose Luis

Gerente General

Arroyo Carrasco Álvaro Iván

ACTA DE ENTREGA DEL SPRINT N° 5

Fecha : 12/06/25

Scrum Master : Arroyo Carrasco Álvaro Iván

Product Owner : Zilva Ramirez Jose Luis

Mediante la presente acta se valida y se da conformidad de que la entrega del sprint 5 presenta el **(H12)** **Registro de cuenta cliente, (H13) Inicio de sesión cliente e (H14) Inicio de sesión Administrador** del sistema con las funcionalidades ya predeterminadas por el Product Owner en la acta de reunión de planificación del sprint 5 donde se detalla las historias de usuario y objetivos; elaboradas las especificaciones por el equipo Scrum y Scrum master se da la aprobación del Sprint 5, donde se decide de manera unánime aprobar el sprint mencionado donde se presentaron los requerimientos para el proyecto **Sistema Web Para La Optimización En La Atención Al Cliente Y Control De Inventario.**

Firma en señal de conformidad



Zilva Ramirez Jose Luis

Gerente General



Arroyo Carrasco Álvaro Iván

Anexo 12. Video explicativo del software

Link del video:

https://youtu.be/SQzJNcD_p70?si=Nu7FZlydmv7sfjx