



UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería de Sistemas

**Proyecto de implementación de un sistema de
ventas para grandes almacenes**

Curso: Calidad y Pruebas de Software

Docente: Ing. *Patrick Cuadros Quiroga*

Integrantes:

Edward Hernan Apaza Mamani

(2018060915)

Ronal Daniel Lupaca Mamani

(2020067146)

Carlos Andrés Escobar Rejas

(2021070016)

Aarón Pedro Paco Ramos

(2018000654)

**Tacna – Perú
2024**



Descripción

Descripción del Proyecto: " Implementación de un sistema de ventas para grandes almacenes "

Este proyecto se centra en la implementación de un sistema de ventas, diseñado específicamente para grandes almacenes o compañías como Plaza Veá. El sistema permite la recepción y procesamiento de ventas a través de múltiples cajas de atención, facilitando un control eficiente de productos, generación de recibos, gestión de usuarios para mayoristas y administración de personal. La duración del proyecto es de 90 días, comenzando el 1 de julio y finalizando el 28 de septiembre. El objetivo principal es mejorar la eficiencia en la gestión de ventas y la administración de productos en grandes almacenes.

Abstract

Project Description: " Implementation of a Sales System for Large Warehouses "

This project focuses on implementing a sales system specifically designed for large warehouses or companies like Plaza Veá. The system allows for the reception and processing of sales through multiple service counters, facilitating efficient product control, receipt generation, user management for wholesalers, and personnel administration. The project duration is 90 days, starting from July 1st to September 28th. The primary objective is to enhance efficiency in sales management and product administration in large warehouses.



Antecedentes o introducción

El presente proyecto se enfoca en la implementación de un sistema de ventas avanzado, especialmente diseñado para grandes almacenes y empresas de retail como Plaza Vea. La motivación principal detrás de este desarrollo es la necesidad de modernizar y optimizar los procesos de ventas y gestión de productos en estos entornos, donde la alta demanda y la variedad de productos requieren soluciones tecnológicas robustas y eficientes.

Actualmente, muchos grandes almacenes enfrentan desafíos significativos en la gestión de sus ventas y productos debido al uso de sistemas tradicionales que son propensos a errores y resultan ineficientes. La implementación de un sistema de ventas moderno permitirá mejorar estos procesos mediante la automatización y el uso de tecnologías avanzadas. Este sistema integrará múltiples funcionalidades clave, tales como la recepción y procesamiento de ventas en diversas cajas de atención, control de inventarios en tiempo real, generación automática de recibos, gestión de usuarios mayoristas, y administración del personal.

Al adoptar un enfoque integral y utilizando herramientas de análisis y pruebas de software como Snyk, SonarQube y SonarCloud, el proyecto garantiza que el sistema sea seguro, eficiente y fácil de usar. Esto no solo mejorará la experiencia de compra de los clientes, sino que también optimizará la operatividad interna de los almacenes, reduciendo errores humanos y aumentando la eficiencia operativa. En última instancia, este proyecto busca transformar la manera en que los grandes almacenes gestionan sus ventas, adaptándose a las demandas del mercado moderno y asegurando un servicio de alta calidad para sus clientes.

Título

Implementación de un sistema de ventas para grandes almacenes.

Autores

Edward Hernan Apaza Mamani 2018060915

Ronal Daniel Lupaca Mamani 2020067146

Carlos Andrés Escobar Rejas 2021070016

Aarón Pedro Paco Ramos 2018000654

Planteamiento del problema

Problema



La gestión de ventas en grandes almacenes presenta desafíos significativos debido a la gran cantidad de productos y transacciones diarias. El uso de sistemas tradicionales resulta en ineficiencias, errores humanos y tiempos de espera prolongados para los clientes.

Justificación

La implementación de un sistema de ventas avanzado mejorará la eficiencia operativa, reducirá errores y optimizará el tiempo de procesamiento de ventas. Esto se traducirá en una mejor experiencia para los clientes y en una administración más efectiva de los recursos del almacén.

Alcance

El proyecto abarca la implementación de un sistema de ventas en grandes almacenes, incluyendo la recepción de cajas, control de productos, generación de recibos, gestión de usuarios y administración de personal.

Objetivos

General

Implementar un sistema de ventas que permita gestionar eficientemente las ventas, control de productos, generación de recibos, y administración de usuarios y personal en grandes almacenes.

Específicos

- Facilitar la recepción y procesamiento de ventas a través de múltiples cajas de atención.
- Optimizar el control y la gestión de productos en el almacén.
- Generar recibos de venta de manera automatizada.
- Gestionar usuarios mayoristas y personal administrativo de manera eficiente.
- Mejorar la experiencia de compra para los clientes.

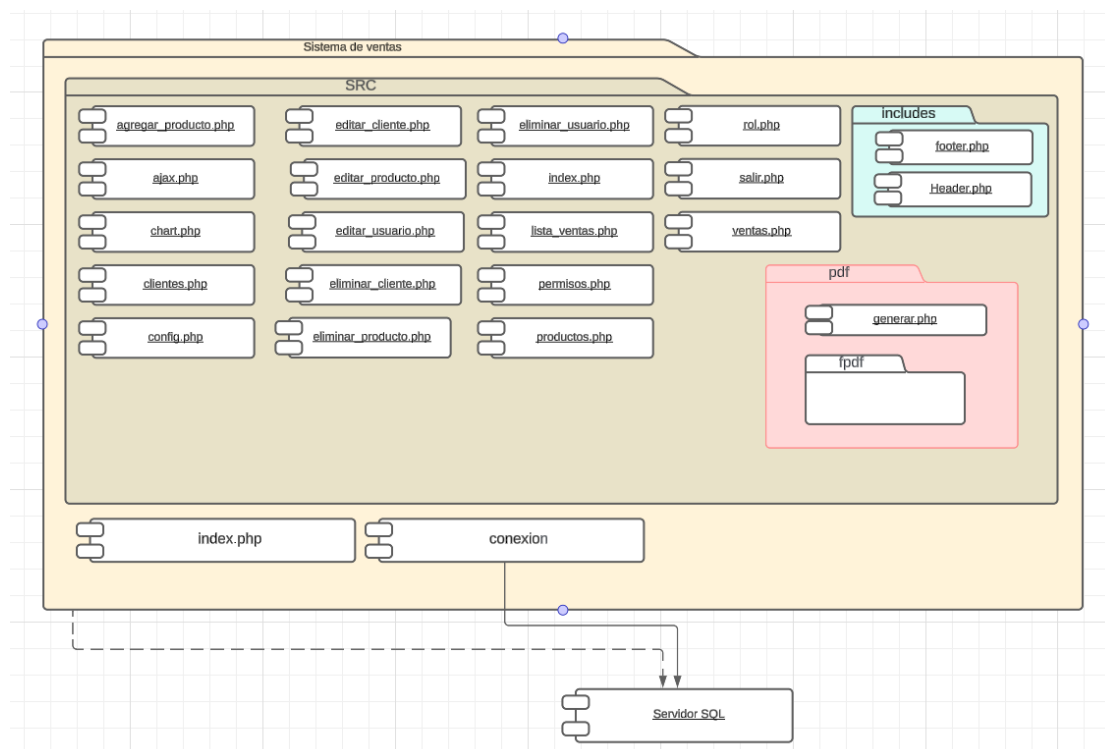


Diagrama de Casos de Uso

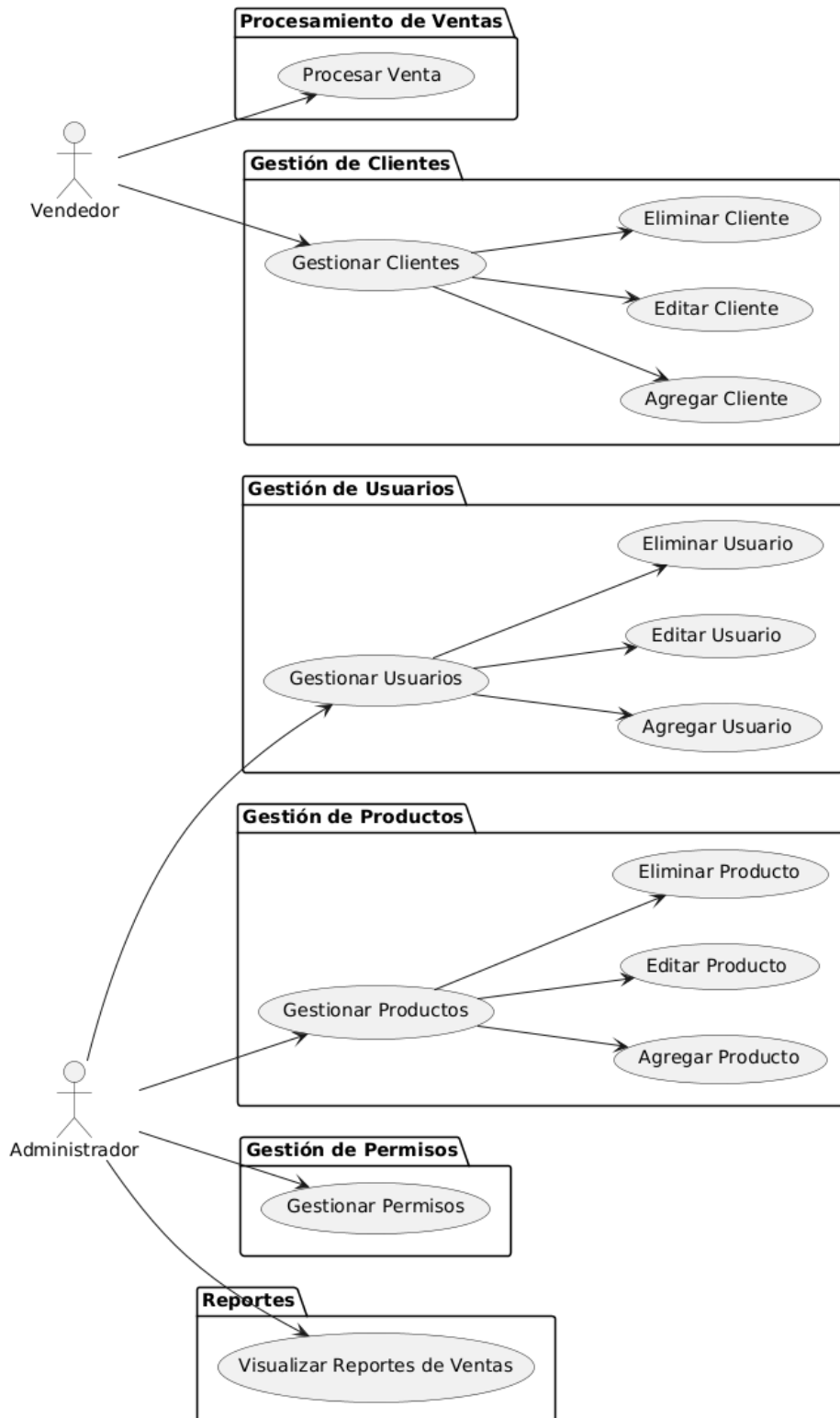


Diagrama de componentes

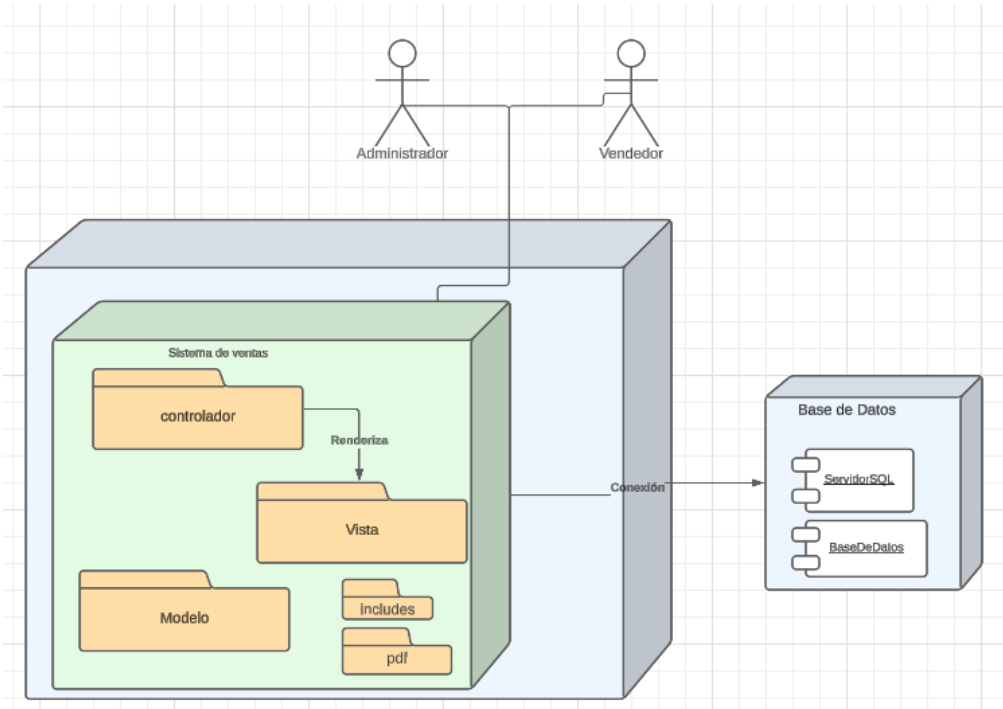
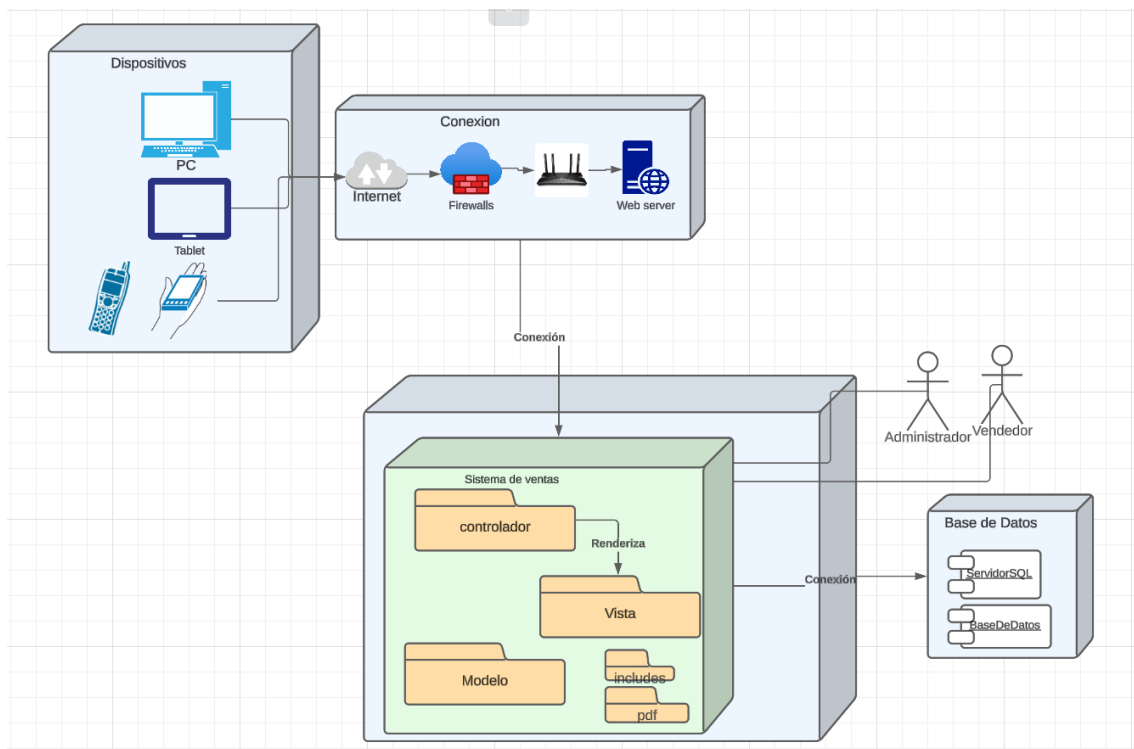


Diagrama de despliegue



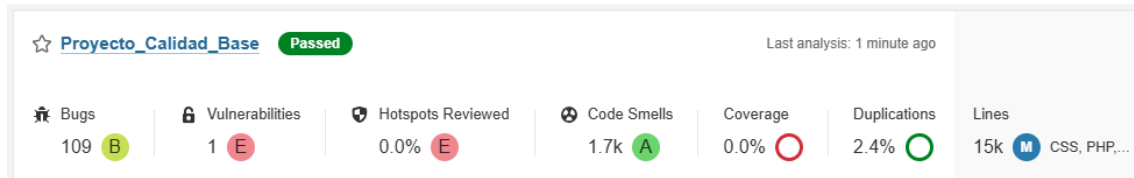


Para el desarrollo de la propuesta, se llevarán a cabo pruebas utilizando herramientas de análisis estático de código, tales como Snyk, SonarQube y SonarCloud. Estas herramientas nos permitirán identificar posibles vulnerabilidades, errores y problemas de calidad en el código, asegurando así un desarrollo más robusto y seguro. Además, facilitarán la implementación de buenas prácticas de programación y la mejora continua del software.

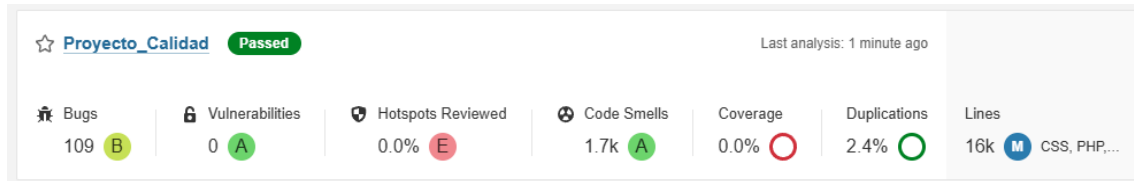


Pruebas realizadas en SonarQube

Antes:

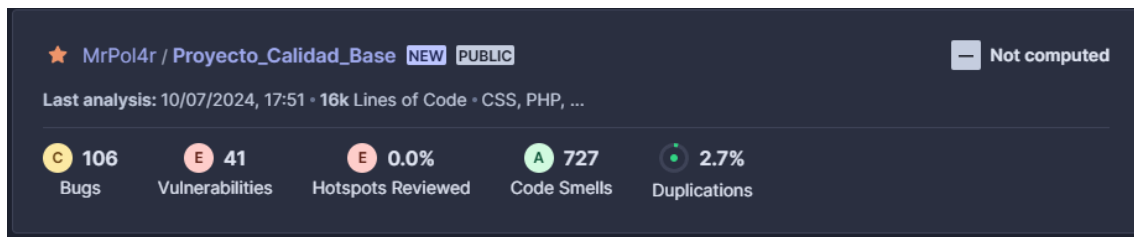


Después:

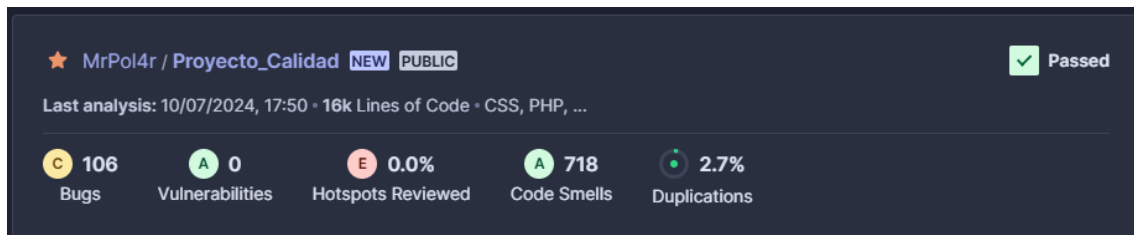


Pruebas realizadas en SonarCloud:

Antes:



Después:



Pruebas realizadas en Snyk:

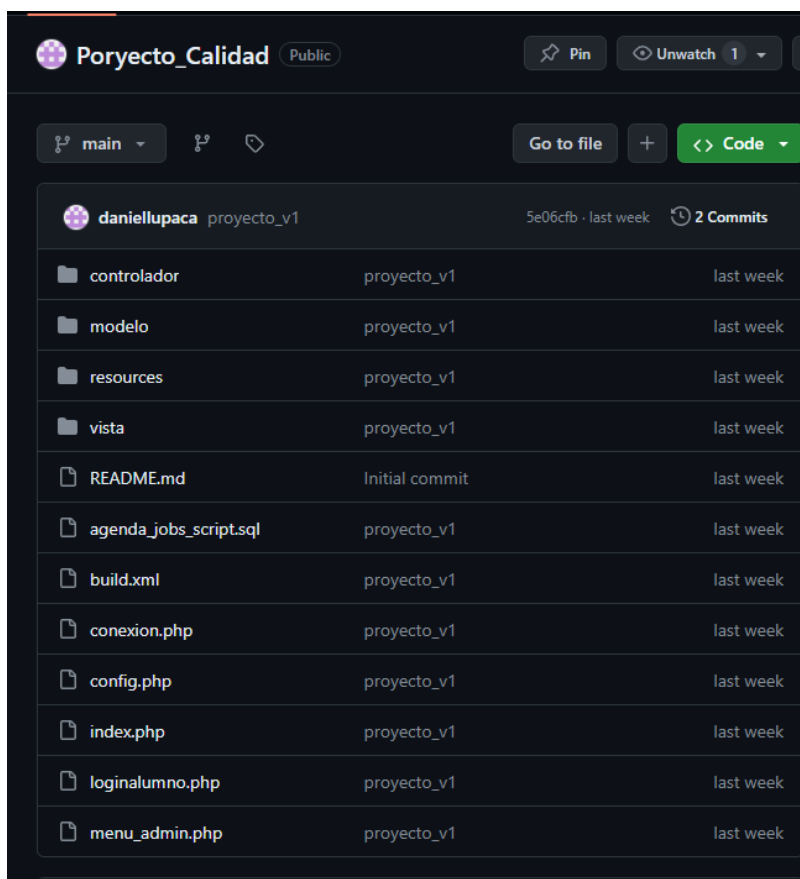
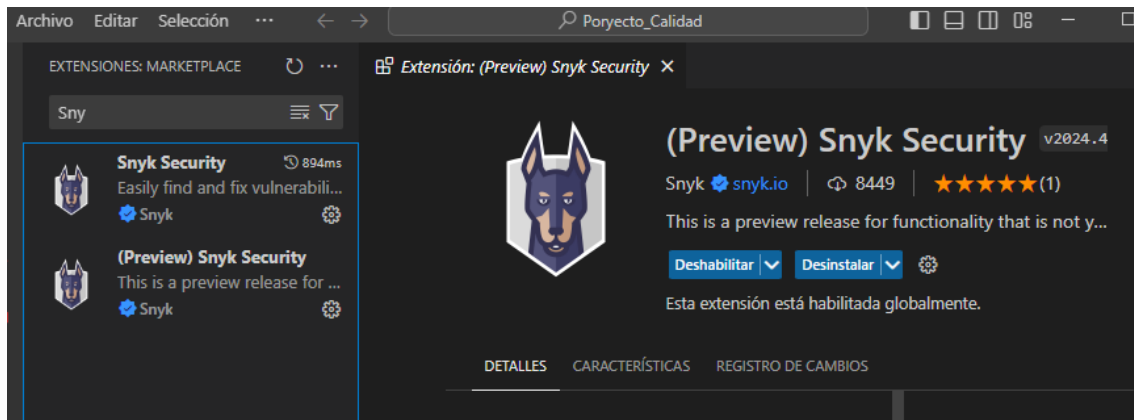
Antes:

Top vulnerable projects ?

Project	Tested	Issues	Actions
CrowProgrammer/sistemas-de-ventas	4 days ago	0 C 43 H 2 M 6 L	

Después:

▼ daniellupaca/Poryecto_Calidad	0 C 0 h 0 METRO 1 I ...
Proyecto	Importado Probado Asuntos ↓





Snyk Code Analysis interface showing a project named 'danielupaca/Proyecto_Calidat'.

Code Analysis

Overview History Settings

Created Sat 20th Apr 2024 | Snapshot for commit 5e06c7b taken by snyk.io 7 days ago | Retest now

IMPORTED BY: ronlupaca@upt.pe PROJECT OWNER: Add a project owner ENVIRONMENT: Add a value BUSINESS CRITICALITY: Add a value

LIFECYCLE: Add a value ANALYSIS SUMMARY: 161 analyzed files (75%) Repo breakdown

Issues: 39

SEVERITY: High (37), Medium (0), Low (2)

PRIORITY SCORE: Scored between 0 - 1000

STATUS: Open (39), Ignored (0)

Cross-site Scripting (XSS)

SNYK CODE CWE-79

SCORE: 854

Unsanitized input from an HTTP parameter flows into the echo statement, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

vista/docente/forma_modificacionalumno.php

Visual Studio Code editor showing the file 'forma_modificacionalumno.php' with a Snyk Code Vulnerability extension window open.

Snyk Code Vulnerability

Cross-site Scripting (XSS)

Vulnerability CWE-79 Position: line 54 Priority: score: 854

Learn about this vulnerability

FIX ANALYSIS VULNERABILITY OVERVIEW

Unsanitized input from an HTTP parameter [44: 54] flows into the echo statement [54], where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS).

COMMUNITY FIXES

This type of vulnerability was fixed in 73 open source projects. Here are 3 examples:

```
<title>/php echo $_SERVER["HTTP_HOST"] > :: Pimcore</title>
<title>/php echo htmlentities($_SERVER["HTTP_HOST"], ENT_QUOTES, 'UTF-8') > :: Pimcore</title>
```

Ignore on line 54

Ignore in this file

Snyk Language Server

PROBLEMAS 2024-04-26T18:15:15-05:00 INF Found 1 issues for path c:\Users\HP\Desktop\calidat\Proyecto_Calidat\vista\docente\forma_modificacionalumno.php and range 53:183-53:117 service:CodeActionsService

2024-04-26T18:15:15-05:00 INF Returning 1 code actions service:CodeActionsService

2024-04-26T18:15:15-05:00 INF CodeActionHandler - SENDING response [{"command": "snyk.openInBrowser", "title": "Learn more about Cross-site Scripting (XSS) (Snyk)", "diagnostic": {"code": "js/jsp/SSS", "codeDescription": {"href": "https://docs.snyk.io/learn/snyk-code/snyk-code-quality-rules", "message": "Cross-site Scripting (XSS): Unsanitized input from an HTTP parameter flows into the echo statement, ...", "range": {"start": {"character": 117, "line": 53}, "end": {"character": 183, "line": 53}}, "severity": "Error", "source": "Snyk Code"}}, {"kind": "quickfix", "title": "Learn more about Cross-site Scripting (XSS) (Snyk)"}]

CODE SECURITY

39 vulnerabilities found by Snyk

There are no vulnerabilities fixable ...

formA_modificacionalumno.php d...

Cross-site Scripting (XSS) for...

Cross-site Scripting (XSS) for...

Cross-site Scripting (XSS) for...

CODE QUALITY

Snyk found no issues! ✓

There are no vulnerabilities fixable ...

HELP & FEEDBACK



Durante la revisión del código del proyecto `sis_venta`, se identificaron varias vulnerabilidades de inyección SQL y otros problemas de seguridad. Utilizando la herramienta Snyk, se analizaron los archivos del proyecto y se implementaron las correcciones necesarias para mejorar la seguridad y robustez del sistema. A continuación, se presenta un resumen detallado de los errores encontrados y las soluciones aplicadas.

1. Inyección SQL

Archivo: `generar.php`

Problema: El código original permitía la inyección SQL debido al uso directo de parámetros de entrada (`$_GET['v']` y `$_GET['cl']`) en las consultas SQL sin ningún tipo de sanitización.

43 of 51 issues Group by none Sort by highest severity

H **SQL Injection** [🔗](#)

SNYK CODE | [CWE-89](#)

SCORE
865

```
9 $id = $_GET['v'];
10 $idcliente = $_GET['cl'];
11 $config = mysqli_query($conexion, "SELECT * FROM configuracion");
12 $datos = mysqli_fetch_assoc($config);
13 $clientes = mysqli_query($conexion, "SELECT * FROM cliente WHERE idcliente = $idcliente");
```

Unsanitized input from an HTTP parameter flows into `mysqli_query`, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

[src/pdf/generar.php](#) 5 steps in 1 file

[Learn about this type of vulnerability and how to fix it](#)

[Ignore](#) [Full details](#)

Código Original:

```
php
Copiar código
$id = $_GET['v'];
$idcliente = $_GET['cl'];
$config = mysqli_query($conexion, "SELECT * FROM configuracion");
$datos = mysqli_fetch_assoc($config);
$clientes = mysqli_query($conexion, "SELECT * FROM cliente WHERE
idcliente = $idcliente");
$datosC = mysqli_fetch_assoc($clientes);
$ventas = mysqli_query($conexion, "SELECT d.*, p.codproducto,
p.descripcion FROM detalle_venta d INNER JOIN producto p ON
d.id_producto = p.codproducto WHERE d.id_venta = $id");
```



Solución: Se utilizaron sentencias preparadas (prepared statements) para evitar la inyección SQL. Se asignaron los parámetros GET a variables seguras y se realizaron las consultas de manera segura.

Código Corregido:

```
php
Copiar código
$idSeguro = $_GET['v'];
$idclienteSeguro = $_GET['cl'];

// Consulta configuración
$configStmt = $conexion->prepare("SELECT * FROM configuracion");
$configStmt->execute();
$config = $configStmt->get_result();
$datos = $config->fetch_assoc();

// Consulta cliente
$clienteStmt = $conexion->prepare("SELECT * FROM cliente WHERE
idcliente = ?");
$clienteStmt->bind_param("i", $idclienteSeguro);
$clienteStmt->execute();
$clientes = $clienteStmt->get_result();
$datosC = $clientes->fetch_assoc();

// Consulta ventas
$ventasStmt = $conexion->prepare("SELECT d.*, p.codproducto,
p.descripcion FROM detalle_venta d INNER JOIN producto p ON
d.id_producto = p.codproducto WHERE d.id_venta = ?");
$ventasStmt->bind_param("i", $idSeguro);
$ventasStmt->execute();
$ventas = $ventasStmt->get_result();
```

2. Manejo Seguro de Variables de Entrada

Archivo: `editar_producto.php`

Problema: El uso directo de los parámetros de entrada (`$_GET['id']`, `$_POST['codigo']`, `$_POST['producto']`, `$_POST['precio']`) en las consultas SQL sin validación permitía la inyección SQL.



H **SQL Injection** [🔗](#)

SNYK CODE | [CWE-89](#) [🔗](#)

SCORE
865

```
40 $id_producto = $_REQUEST['id'];
41 if (!is_numeric($id_producto)) {
42     header("Location: productos.php");
43 }
44 $query_producto = mysqli_query($conexion, "SELECT * FROM producto WHERE codproducto = $id_producto");
```

Unsanitized input from an **HTTP parameter flows** into **mysqli_query**, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

[src/editor_producto.php](#) [🔗](#) 6 steps in 1 file

[📖 Learn about this type of vulnerability and how to fix it](#) [🔗](#)

[🗑️ Ignore](#) [📄 Full details](#)

Código Original:

```
php
Copiar código
$codproducto = $_GET['id'];
$codigo = $_POST['codigo'];
$producto = $_POST['producto'];
$precio = $_POST['precio'];
$query_update = mysqli_query($conexion, "UPDATE producto SET codigo = '$codigo', descripcion = '$producto', precio = $precio WHERE codproducto = $codproducto");
```

Solución: Se implementaron prepared statements para realizar consultas SQL seguras.

Código Corregido:

```
php
Copiar código
$codproductoSeguro = $_GET['id'];
$codigoSeguro = $_POST['codigo'];
$productoSeguro = $_POST['producto'];
$precioSeguro = $_POST['precio'];

$query_update_seguro = $conexion->prepare("UPDATE producto SET codigo = ?, descripcion = ?, precio = ? WHERE codproducto = ?");
$query_update_seguro->bind_param("ssdi", $codigoSeguro, $productoSeguro, $precioSeguro, $codproductoSeguro);
$query_update_seguro->execute();
```

3. Validación y Saneamiento de Entradas de Usuario

Archivo: `agregar_producto.php`



Problema: El código original permitía la inserción de datos no validados directamente en la base de datos, lo que exponía el sistema a ataques de inyección SQL.

Código Original:

```
php
Copiar código
$codigo = $_POST['codigo'];
$producto = $_POST['producto'];
$precio = $_POST['precio'];
$query = mysqli_query($conexion, "SELECT * FROM producto WHERE codigo = '$codigo'");
$result = mysqli_fetch_array($query);
$query_insert = mysqli_query($conexion, "INSERT INTO producto(codigo, descripcion, precio) values ('$codigo', '$producto', '$precio')");
```

Solución: Se realizaron consultas seguras utilizando prepared statements.

Código Corregido:

```
php
Copiar código
$codigoSeguro = $_POST['codigo'];
$productoSeguro = $_POST['producto'];
$precioSeguro = $_POST['precio'];
$cantidadSeguro = $_POST['cantidad'];
$usuario_idSeguro = $_SESSION['idUser'];

$query_insert_seguro = $conexion->prepare("INSERT INTO producto(codigo, descripcion, precio, existencia, usuario_id) VALUES (?, ?, ?, ?, ?)");
$query_insert_seguro->bind_param("ssdii", $codigoSeguro, $productoSeguro, $precioSeguro, $cantidadSeguro, $usuario_idSeguro);
$query_insert_seguro->execute();
```

4. Protección Contra Inyecciones SQL en Consultas de Selección

Archivo: `clientes.php`

Problema: El uso directo de parámetros GET y POST en las consultas SQL sin sanitización exponía el sistema a inyecciones SQL.

Código Original:

```
php
Copiar código
$id_cliente = $_GET['id'];
$cliente = mysqli_query($conexion, "SELECT * FROM cliente WHERE idcliente = $id_cliente");
```

Solución: Se utilizaron prepared statements para realizar consultas seguras.



Código Corregido:


```
php
Copiar código
$id_clienteSeguro = $_GET['id'];
$clienteSeguro = $conexion->prepare("SELECT * FROM cliente WHERE
idcliente = ?");
$clienteSeguro->bind_param("i", $id_clienteSeguro);
$clienteSeguro->execute();
$cliente = $clienteSeguro->get_result();
```

Conclusión

Se identificaron y corrigieron múltiples vulnerabilidades de inyección SQL en el proyecto `sis_venta`. La utilización de `prepared statements` y la sanitización de entradas de usuario fueron las principales medidas implementadas para asegurar el sistema. Estas correcciones no solo mejoran la seguridad del sistema, sino que también aumentan su robustez y confiabilidad.

Cómo vistos al inicio se detectó 51 vulnerabilidades , 43 de grado Alto y dos de medio y 6 de nivel bajo

Top vulnerable projects ?

Project	Tested	Issues	Actions
 CrowProgrammer/sistemas-de-ventas	4 days ago	<div><div>0</div><div>C</div><div>43</div><div>H</div><div>2</div><div>M</div><div>6</div><div>L</div></div>	

se trabajó en 4 tipos de vulnerabilidades :

1. Uso de Credenciales Codificadas (Hardcoded Credentials)
2. Cross-site Scripting (XSS)
3. Inyección SQL (SQL Injection)
4. Problemas de seguridad relacionados con la configuración del servidor (como la exposición del `HTTP_HOST`)

Se trabajó en la visual code para corregir y hacer el commit en el repositorio público para que Snyk pueda volver a escanear el proyecto. mostrando la eliminación de 38 vulnerabilidades.



▼ SEVERITY

<input type="checkbox"/> High	0
<input type="checkbox"/> Medium	0
<input type="checkbox"/> Low	1

Estas correcciones mejoran significativamente la seguridad de tu aplicación, protegiendo tanto los datos de los usuarios como la integridad de tu sistema.

Tecnología de información

Snyk: Utilizado para el escaneo de vulnerabilidades en las dependencias y para realizar análisis estático del código.

SonarQube: Implementado para la revisión continua del código, enfocado en la detección de bugs, vulnerabilidades y deuda técnica.

SonarCloud: Servicio en la nube que proporciona análisis automatizado para detectar y corregir fallas en el código.

Git: Sistema de control de versiones empleado para el seguimiento de cambios y colaboración en el desarrollo del proyecto.

GitHub: Plataforma de alojamiento para el repositorio del proyecto, integrada con Snyk para el análisis de seguridad tras cada push.

MySQL: Sistema de gestión de base de datos para almacenar, modificar y extraer la información utilizada en la aplicación.

PHP: Lenguaje de programación del lado del servidor para la implementación de la lógica de negocio de la aplicación.

Metodología, técnicas usadas

Desarrollo Ágil: Implementación iterativa e incremental para facilitar la flexibilidad y la entrega continua de valor.

Integración Continua/Despliegue Continuo (CI/CD): Prácticas que permiten la integración de cambios de forma automática y sistemática.



Revisión de Código: Práctica colaborativa en la que los cambios de código son revisados por pares antes de ser integrados al proyecto.

Análisis Estático de Código: Empleo de herramientas como Snyk y SonarQube para detectar problemas en el código sin ejecutarlo.

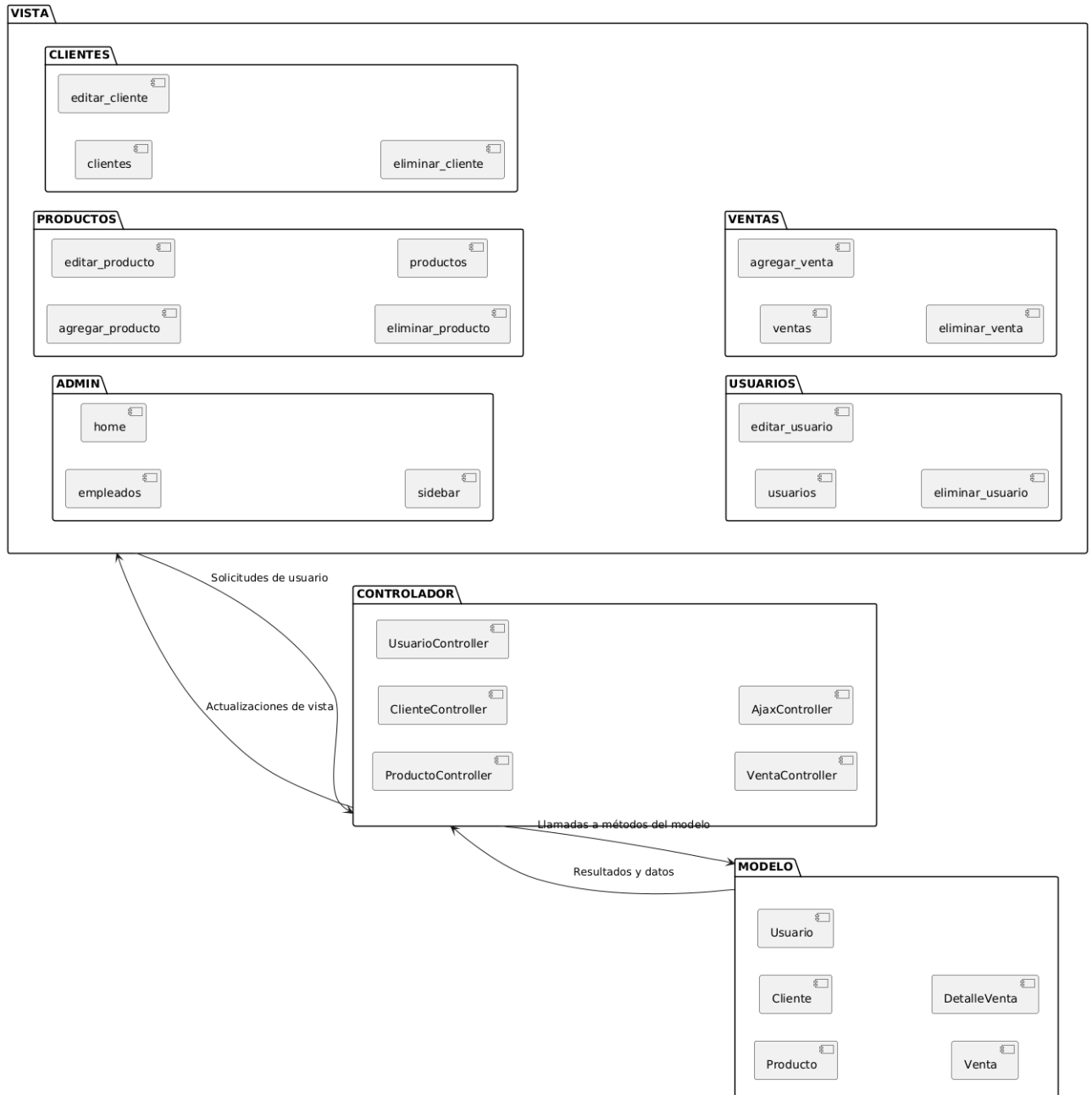
Seguridad de la Información: Aplicación de prácticas de codificación segura, tales como la sanitización de entradas y el uso de sentencias preparadas.

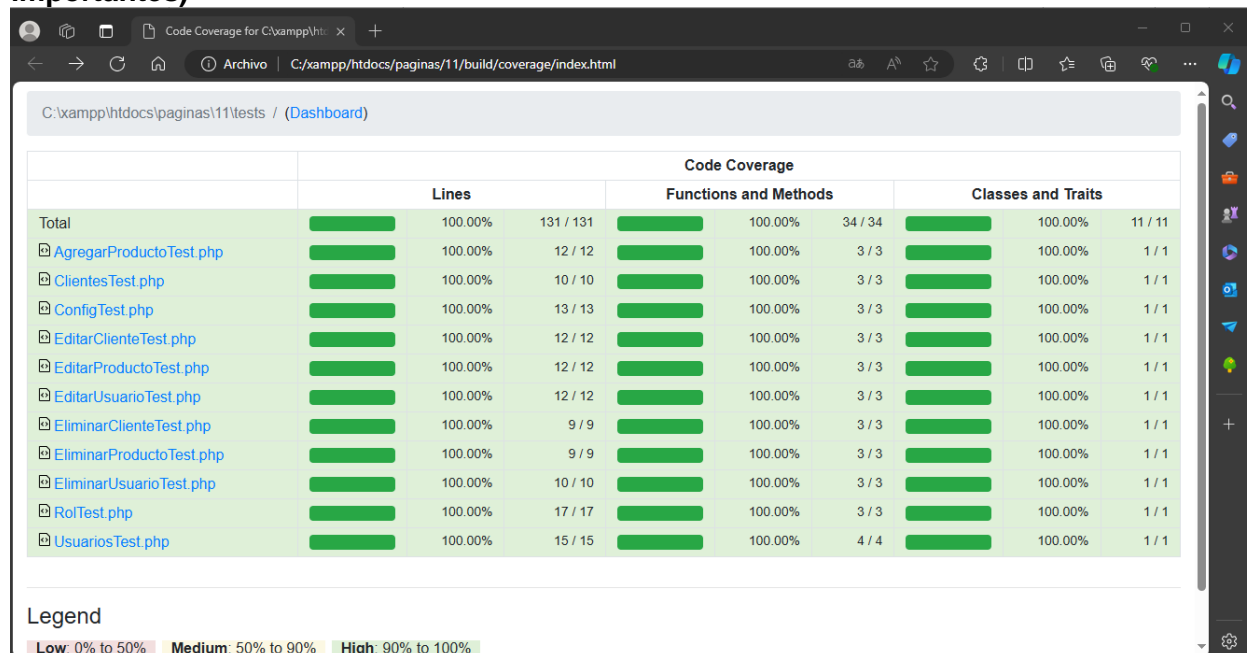
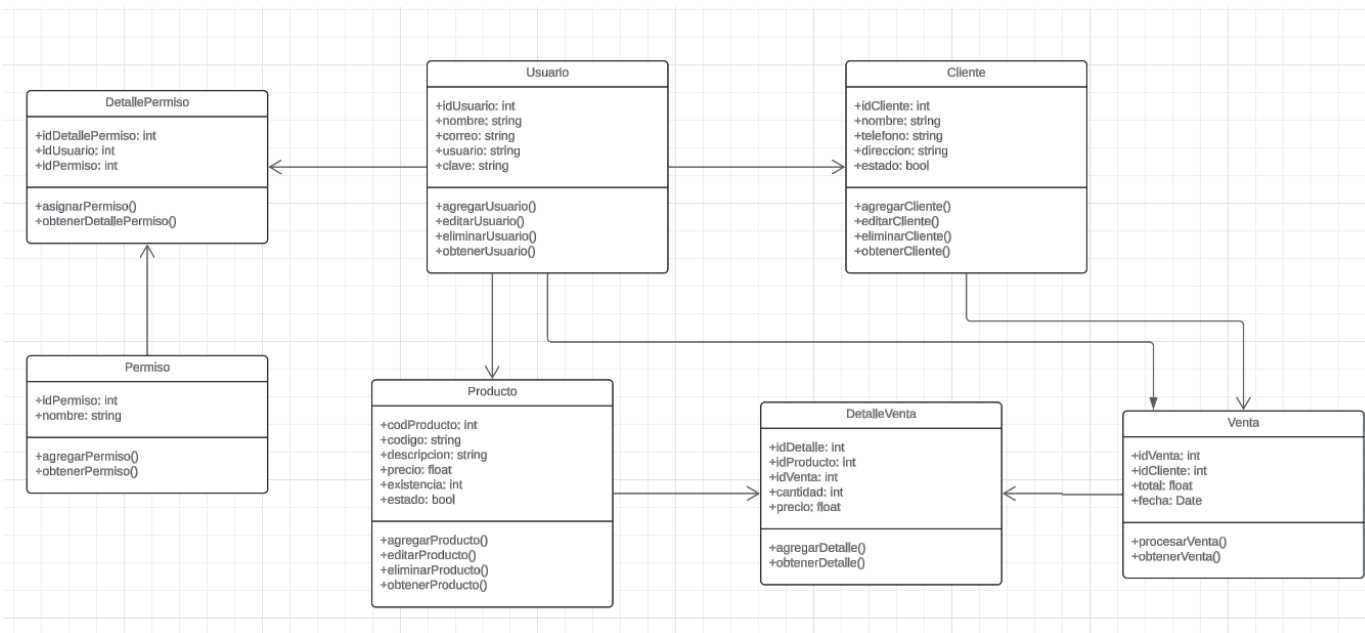
Cronograma (personas, tiempo, otros recursos) Basado en las observaciones que la herramienta SonarQube les informará sobre la aplicación, a fin de reducir la deuda técnica, vulnerabilidades, fallas, etc. a 0.

Actividades	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5
Vulnerabilidades Altas (Ronal Lupaca)	X	X			
Vulnerabilidades Medias (Ronal Lupaca)	X	X			
Vulnerabilidades Bajas (Ronal Lupaca y Carlos Escobar)	X	X	X		
Redundancia de Código (Paco Ramos, Apaza Mamani)			X		
Corrección de Bugs (Carlos Escobar, Ronal Lupaca)				X	
Pruebas Finales (Ronal Lupaca , Carlos Escobar, Paco Ramos, Apaza Mamani)					X

9. Desarrollo de Solución de Mejora

9.1 Diagrama de Arquitectura de la aplicación

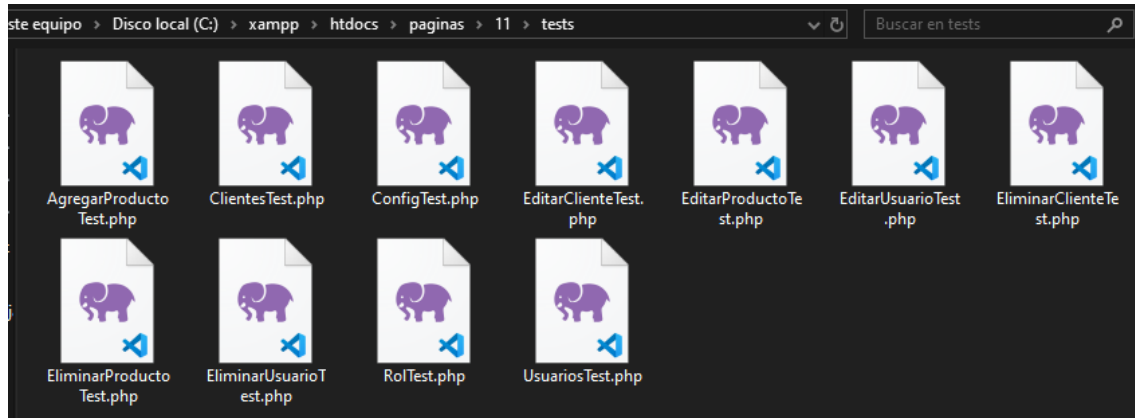




Se muestra nuestro reporte de cobertura de nuestra aplicación. En el reporte, cada archivo, como `AgregarProductoTest.php`, `ClientesTest.php`, `ConfigTest.php`, y otros, ha logrado una cobertura del 100% en tres categorías: líneas de código, funciones y métodos, y clases. Esto indica que las pruebas unitarias han ejecutado cada línea, función y clase en estos archivos, asegurando que todos los aspectos del código funcionan correctamente bajo las condiciones probadas.



TEST CREADOS:



AgregarProductoTest.php

	Code Coverage							
	Lines		Functions and Methods			Classes and Traits		
Total	<div></div>	100.00%	12 / 12	<div></div>	100.00%	3 / 3	<div></div>	100.00%
AgregarProductoTest	<div></div>	100.00%	12 / 12	<div></div>	100.00%	3 / 3	<div></div>	100.00%
setUp	<div></div>	100.00%	4 / 4	<div></div>	100.00%	1 / 1	<div></div>	100.00%
testActualizarProducto	<div></div>	100.00%	7 / 7	<div></div>	100.00%	1 / 1	<div></div>	100.00%
tearDown	<div></div>	100.00%	1 / 1	<div></div>	100.00%	1 / 1	<div></div>	100.00%

El objetivo principal del test `testActualizarProducto` en la clase `AgregarProductoTest` es verificar que la función `actualizarProducto` de la clase `ProductoModelTest` actualice correctamente un registro en la base de datos. Específicamente, este test busca asegurar que:

1. **Correcta Actualización de Datos:** Los campos `codigo`, `descripcion`, y `precio` del producto especificado deben ser actualizados correctamente en la base de datos. El test verifica que los valores en la base de datos coincidan con los nuevos valores proporcionados a la función `actualizarProducto`.
2. **Verificación de Resultados:** Además de actualizar la base de datos, el método puede retornar algún mensaje o valor que indique el éxito de la operación. El test verifica que la respuesta incluya una cadena específica ('Producto Modificado'), lo cual podría ser un mensaje de confirmación de que la actualización fue exitosa.



CientesTest.php

	Code Coverage							
	Lines			Functions and Methods			Classes and Traits	
Total	<div></div>	100.00%	10 / 10	<div></div>	100.00%	3 / 3	CRAP	<div></div> 100.00% 1 / 1
CientesTest	<div></div>	100.00%	10 / 10	<div></div>	100.00%	3 / 3	3	<div></div> 100.00% 1 / 1
setUp	<div></div>	100.00%	5 / 5	<div></div>	100.00%	1 / 1	1	
testRegistrarNewClient	<div></div>	100.00%	4 / 4	<div></div>	100.00%	1 / 1	1	
tearDown	<div></div>	100.00%	1 / 1	<div></div>	100.00%	1 / 1	1	

El test `testRegistrarNewClient` en la clase `CientesTest` está diseñado para validar la funcionalidad de registrar un nuevo cliente en la base de datos mediante la clase `ClienteModelTest`. Su objetivo principal es asegurarse de que el método `registrarCliente` funciona correctamente al añadir un nuevo registro en la tabla `cliente` y verifica dos aspectos clave:

1. **Inserción Correcta en la Base de Datos:** Tras llamar al método `registrarCliente`, el test verifica que realmente se ha insertado un nuevo registro en la tabla `cliente`. Esto se realiza mediante una consulta SQL que busca el cliente por nombre, asegurándose de que el número de filas retornadas sea exactamente uno, lo que indica que el cliente fue agregado correctamente.
2. **Respuesta Esperada:** Verifica que el método `registrarCliente` retorne la cadena 'Cliente registrado', que se espera sea el mensaje de éxito tras registrar un nuevo cliente. Esto puede ser parte de la lógica de la aplicación para proporcionar feedback directo al usuario o a otras partes del sistema sobre el resultado de la operación.

ConfigTest.php

	Code Coverage							
	Lines			Functions and Methods			Classes and Traits	
Total	<div></div>	100.00%	13 / 13	<div></div>	100.00%	3 / 3	CRAP	<div></div> 100.00% 1 / 1
ConfigTest	<div></div>	100.00%	13 / 13	<div></div>	100.00%	3 / 3	3	<div></div> 100.00% 1 / 1
setUp	<div></div>	100.00%	4 / 4	<div></div>	100.00%	1 / 1	1	
testActualizarConfiguracion	<div></div>	100.00%	8 / 8	<div></div>	100.00%	1 / 1	1	
tearDown	<div></div>	100.00%	1 / 1	<div></div>	100.00%	1 / 1	1	

3. El test `testActualizarConfiguracion` en la clase `ConfigTest` tiene como objetivo verificar la funcionalidad del método `actualizarConfiguracion` dentro de la clase `ConfiguracionModelTest`, que se encarga de modificar los datos de configuración de una empresa en la base de datos. Este test se centra en asegurar que los cambios se apliquen **correctamente** y que el método devuelva **un mensaje de confirmación adecuado**.



1. **Correcta Actualización en la Base de Datos:** Realiza consultas para asegurarse de que los datos en la base de datos correspondan a los nuevos valores suministrados, validando que cada campo (nombre, teléfono, email, dirección) se haya actualizado correctamente.
2. **Mensaje de Confirmación:** Comprueba que el resultado devuelto por el método contenga un mensaje específico (en este caso, algo como 'Datos modificados'), indicando el éxito de la operación.

EditarClienteTest.php

Code Coverage for C:\xampp\htdocs\paginas\11\build\coverage\EditarClienteTest.php.html

C:\xampp\htdocs\paginas\11\tests / EditarClienteTest.php

	Code Coverage									
	Lines			Functions and Methods			Classes and Traits			
Total	<div></div>	100.00%	12 / 12	<div></div>	100.00%	3 / 3	CRAP	<div></div>	100.00%	1 / 1
EditarClienteTest	<div></div>	100.00%	12 / 12	<div></div>	100.00%	3 / 3	3	<div></div>	100.00%	1 / 1
setUp	<div></div>	100.00%	4 / 4	<div></div>	100.00%	1 / 1	1			
testActualizarCliente	<div></div>	100.00%	7 / 7	<div></div>	100.00%	1 / 1	1			
tearDown	<div></div>	100.00%	1 / 1	<div></div>	100.00%	1 / 1	1			

El test `testActualizarCliente` dentro de la clase `EditarClienteTest` está diseñado para comprobar la funcionalidad de actualizar los detalles de un cliente existente en la base de datos a través del método `actualizarCliente` en la clase `ClienteModelTest`. Este test se enfoca en asegurar que los datos del cliente se actualicen correctamente y que el método retorne un mensaje indicativo del éxito de la operación.

1. **Comprobación de Datos en Base de Datos:** Realiza una consulta para verificar que los datos del cliente en la base de datos reflejen los nuevos valores proporcionados.
2. **Evaluación del Resultado del Método:** Verifica que el string retornado por el método contenga un mensaje específico (como 'Cliente Actualizado correctamente'), lo que indica que la actualización fue exitosa y que el método está funcionando como se espera.

EditarProductoTest.php

Code Coverage for C:\xampp\h...

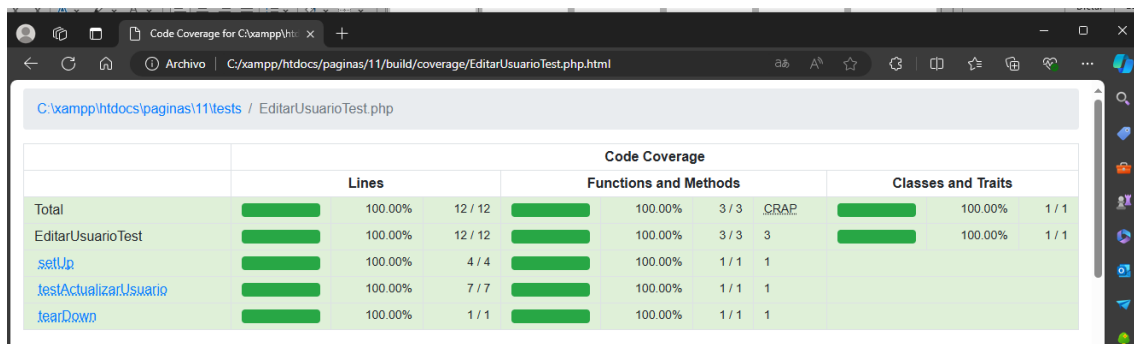
<



El test `testActualizarProducto` en la clase `EditarProductoTest` está diseñado para asegurar que la funcionalidad de actualizar los detalles de un producto específico en la base de datos funcione correctamente. Este se realiza a través de la clase `ProductoModelTest`, enfocándose en garantizar que los cambios se aplican correctamente y que el método devuelva una confirmación del éxito de la operación.

1. Comprobación de Datos en Base de Datos: Realiza una consulta para verificar que los datos del producto en la base de datos reflejen los nuevos valores proporcionados. Esto incluye comparar el código, descripción y precio.
2. Evaluación del Resultado del Método: Verifica que el string retornado por el método contenga un mensaje específico (como 'Producto Modificado'), lo que indica que la actualización fue exitosa y que el método está funcionando correctamente.

EditarUsuarioTest.php



El test `testActualizarUsuario` en la clase `EditarUsuarioTest` está diseñado para verificar la funcionalidad de actualizar la información de un usuario existente en la base de datos a través del método `actualizarUsuario` de la clase `UsuarioModelTest`. Este test se enfoca en asegurar que los cambios se realicen correctamente y que el método devuelva una confirmación apropiada del éxito de la operación.

1. Comprobación de Datos en la Base de Datos: Realiza una consulta para verificar que los datos del usuario en la base de datos reflejen los nuevos valores proporcionados, incluyendo nombre, correo y nombre de usuario.
2. Evaluación del Resultado del Método: Verifica que el string retornado por el método contenga un mensaje específico (como 'Usuario Actualizado'), lo que indica que la actualización fue exitosa y que el método está funcionando correctamente.

EliminarClienteTest.php

Code Coverage for C:\xampp\htdocs\paginas\11\build\coverage\EliminarClienteTest.php.html

C:\xampp\htdocs\paginas\11\tests / EliminarClienteTest.php

	Code Coverage							
	Lines			Functions and Methods			Classes and Traits	
Total	<div></div>	100.00%	9 / 9	<div></div>	100.00%	3 / 3	CRAP	<div></div> 100.00%1 / 1
EliminarClienteTest	<div></div>	100.00%	9 / 9	<div></div>	100.00%	3 / 3	3	<div></div> 100.00%1 / 1
setUp	<div></div>	100.00%	4 / 4	<div></div>	100.00%	1 / 1	1	
testEliminarCliente	<div></div>	100.00%	4 / 4	<div></div>	100.00%	1 / 1	1	
tearDown	<div></div>	100.00%	1 / 1	<div></div>	100.00%	1 / 1	1	

El test `testEliminarCliente` en la clase `EliminarClienteTest` tiene como objetivo verificar la correcta funcionalidad del método `eliminarCliente` de la clase `ClienteModelTest`, el cual está diseñado para cambiar el estado de un cliente en la base de datos, marcándolo como eliminado, en lugar de borrar completamente el registro. Esto es una práctica común en la gestión de datos para mantener la integridad del historial de datos.

1. Eliminación del Cliente: Invoca el método `eliminarCliente` para cambiar el estado del cliente especificado (en este caso, el cliente con `idcliente = 1`).
2. Verificación de la Eliminación:
 - Comprobación de Estado en la Base de Datos: Realiza una consulta para verificar que el estado del cliente en la base de datos refleje que ha sido marcado como eliminado (`estado = 0`), lo que indica que el cliente ya no está activo.

EliminarProductoTest.php

Code Coverage for C:\xampp\htdocs\...

Archivo | C:\xampp\htdocs\paginas\11\build\coverage\EliminarProductoTest.php.html

C:\xampp\htdocs\paginas\11\tests / EliminarProductoTest.php

	Code Coverage								
		Lines			Functions and Methods			Classes and Traits	
Total	<div></div>	100.00%	9 / 9	<div></div>	100.00%	3 / 3	CRAP	<div></div>	100.00% 1 / 1
EliminarProductoTest	<div></div>	100.00%	9 / 9	<div></div>	100.00%	3 / 3	3	<div></div>	100.00% 1 / 1
setUp	<div></div>	100.00%	4 / 4	<div></div>	100.00%	1 / 1	1		
testEliminarProducto	<div></div>	100.00%	4 / 4	<div></div>	100.00%	1 / 1	1		
tearDown	<div></div>	100.00%	1 / 1	<div></div>	100.00%	1 / 1	1		

El test `testEliminarProducto` dentro de la clase `EliminarProductoTest` está diseñado para asegurar que el método `eliminarProducto` de la clase `ProductoModelTest` funcione correctamente al cambiar el estado de un producto en la base de datos, marcándolo como eliminado sin eliminar físicamente el registro. Este enfoque ayuda a mantener la integridad de los datos y permite la restauración del estado anterior del producto si es necesario.

1. Eliminación del Producto: Utiliza el método `eliminarProducto` para cambiar el estado del producto especificado (en este caso, el producto con `codproducto = 1`).



2. Verificación de la Eliminación:

- **Comprobación de Estado en la Base de Datos:** Realiza una consulta para verificar que el estado del producto en la base de datos refleje que ha sido marcado como eliminado (estado = 0), indicando que el producto ya no está activo.

EliminarUsuarioTest.php

	Code Coverage					
	Lines		Functions and Methods		Classes and Traits	
Total	100.00%	10 / 10	100.00%	3 / 3	GRAP	100.00%
EliminarUsuarioTest	100.00%	10 / 10	100.00%	3 / 3	3	100.00%
setUp	100.00%	4 / 4	100.00%	1 / 1	1	
testEliminarUsuario	100.00%	5 / 5	100.00%	1 / 1	1	
tearDown	100.00%	1 / 1	100.00%	1 / 1	1	

El test `testEliminarUsuario` en la clase `EliminarUsuarioTest` está diseñado para comprobar la correcta funcionalidad del método `eliminarUsuario` de la clase `UsuarioModelTest`. Este método está destinado a cambiar el estado de un usuario en la base de datos, marcándolo como eliminado (usualmente cambiando el estado a '0') en lugar de eliminar completamente el registro. Esto permite mantener la integridad del historial de datos y facilita la restauración de estados anteriores si es necesario.

1. **Eliminación del Usuario:** Utiliza el método `eliminarUsuario` para cambiar el estado del usuario especificado (en este caso, el usuario con `idusuario = 1`).
2. **Verificación de la Eliminación:**
 - **Comprobación del Estado en la Base de Datos:** Realiza una consulta para verificar que el estado del usuario en la base de datos refleje que ha sido marcado como eliminado (estado = 0).
 - **Evaluación del Resultado del Método:** Verifica que el string retornado por el método contenga un mensaje específico (como 'Usuario eliminado'), lo que indica que la eliminación fue exitosa y que el método está funcionando correctamente.

RoITest.php

Code Coverage for C:\xampp\htdocs\...

Archivo | C:\xampp\htdocs\paginas\11\build\coverage\RoTest.php.html

C:\xampp\htdocs\paginas\11\tests / RoTest.php

	Code Coverage									
	Lines			Functions and Methods				Classes and Traits		
Total	<div></div>	100.00%	17 / 17	<div></div>	100.00%	3 / 3	CRAP	<div></div>	100.00%	1 / 1
RoTest	<div></div>	100.00%	17 / 17	<div></div>	100.00%	3 / 3	4	<div></div>	100.00%	1 / 1
setUp	<div></div>	100.00%	9 / 9	<div></div>	100.00%	1 / 1	1			
testActualizarPermisos	<div></div>	100.00%	7 / 7	<div></div>	100.00%	1 / 1	2			
tearDown	<div></div>	100.00%	1 / 1	<div></div>	100.00%	1 / 1	1			

El test `testActualizarPermisos` en la clase `RolTest` está diseñado para verificar la funcionalidad de la clase `RolModelTest`, específicamente el método `actualizarPermisos`, que se encarga de gestionar los permisos asociados a un usuario en la base de datos. Este test tiene como objetivo asegurar que los permisos se actualizan correctamente y que el sistema refleja estos cambios en la tabla `detalle_permisos`.

1. Actualización de Permisos: Utiliza el método `actualizarPermisos` para asignar un conjunto completo de permisos (todos los disponibles) al usuario con `idusuario = 1`.
2. Verificación de la Actualización:
 - Comprobación de Permisos en la Base de Datos: Realiza una consulta para verificar que los permisos en `detalle_permisos` para el usuario indicado reflejen los cambios esperados, es decir, que contengan todos los permisos especificados.
 - Evaluación del Resultado del Método: Verifica que el resultado devuelto por el método indique que los permisos fueron actualizados correctamente, esperando un mensaje específico como 'Permisos actualizados'.

UsuariosTest.php

Code Coverage for C:\xampp\htdo...

C:\xampp\htdocs\paginas/11/build/coverage/UsuariosTest.php.html

C:\xampp\htdocs\paginas/11/tests / UsuariosTest.php

	Code Coverage									
	Lines			Functions and Methods				Classes and Traits		
Total	<div><div></div></div>	100.00%	15 / 15	<div><div></div></div>	100.00%	4 / 4	CRAP	<div><div></div></div>	100.00%	1 / 1
UsuariosTest	<div><div></div></div>	100.00%	15 / 15	<div><div></div></div>	100.00%	4 / 4	4	<div><div></div></div>	100.00%	1 / 1
setUp	<div><div></div></div>	100.00%	4 / 4	<div><div></div></div>	100.00%	1 / 1	1			
testRegistrarUsuario	<div><div></div></div>	100.00%	8 / 8	<div><div></div></div>	100.00%	1 / 1	1			
testRegistrarUsuarioExistente	<div><div></div></div>	100.00%	2 / 2	<div><div></div></div>	100.00%	1 / 1	1			
tearDown	<div><div></div></div>	100.00%	1 / 1	<div><div></div></div>	100.00%	1 / 1	1			

La clase `UsuariosTest` contiene pruebas unitarias diseñadas para validar la funcionalidad del manejo de usuarios a través de la clase `UsuarioModelTest`. Hay dos pruebas clave en esta clase: `testRegistrarUsuario` y `testRegistrarUsuarioExistente`. Cada una está



destinada a asegurar que el registro de usuarios funcione de manera eficiente y correcta, tanto para nuevos usuarios como para casos donde el usuario podría estar duplicado.

- **Objetivo:** Asegurar que el sistema maneje correctamente los intentos de registro de un usuario que usa un correo electrónico ya registrado.

- **Proceso:**

- **Registro de Usuario Duplicado:** Se intenta registrar un nuevo usuario usando un correo electrónico que ya existe en la base de datos.
- **Verificación de Fallo de Registro:** Se comprueba que el sistema retorna un mensaje de error indicando que el correo ya existe.

Pruebas de aceptación basadas en Desarrollo Guiado por el Comportamiento una por cada caso de uso o historia de usuario.

BDD (Desarrollo Guiado por el Comportamiento, por sus siglas en inglés) es una metodología de desarrollo de software que se centra en la colaboración entre desarrolladores, testers y partes interesadas (stakeholders) no técnicas como analistas de negocio o usuarios finales. El objetivo principal de BDD es asegurar que el software se desarrolle desde la perspectiva del comportamiento del sistema, lo cual implica definir el comportamiento esperado antes de comenzar a implementar cualquier código.

Given (Dado): Describe el estado inicial o contexto antes de que ocurra la acción que estamos probando.

When (Cuando): Describe la acción o evento que queremos probar.

Then (Entonces): Describe el resultado esperado después de que ocurra la acción.

Estos escenarios son escritos en colaboración con los stakeholders y son utilizados para guiar tanto el desarrollo como las pruebas. Los pasos de BDD aseguran que el equipo entienda claramente lo que se está construyendo y que todos tengan una visión común del comportamiento esperado del sistema.

TEST PARA LOS ESCENARIOS (FEATURE)



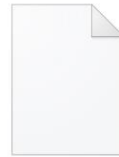
bootstrap



administrador_clientes.feature



administrador_permisos.feature



administrador_productos.feature



administrador_rolles.feature



administrador_usuarios.feature



administrador_ventas.feature



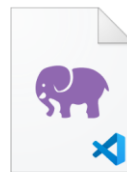
ClienteContext.php



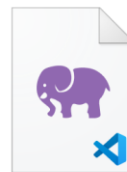
PermisosContext.php



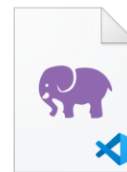
ProductosContext.php



RolesContext.php



UsuariosContext.php



VentasContext.php

Escenario para cotización (feature)

Feature: Administrar Clientes

Como Personal administrativo

Quiero administrar clientes en el sistema

Para poder crear, visualizar, actualizar y eliminar clientes

Scenario: Crear Cliente

Given que el Personal navega a la página de administración de clientes

And selecciona "Crear nuevo cliente"

And no ingresa datos enviando el formulario

And envía el formulario

Then el sistema muestra un mensaje de error "Los campos del formulario no pueden estar vacíos"

And completa el formulario con la información del cliente

And envía el formulario

Then el sistema guarda el nuevo cliente en la base de datos

And muestra un mensaje de confirmación

Scenario: Visualizar Clientes

Given que el Personal navega a la página de administración de clientes

Then el sistema muestra la lista de clientes disponibles

Scenario: Actualizar Cliente

Given que el Personal navega a la página de administración de clientes

And selecciona un cliente para actualizar



And modifica la información del cliente
And envía el formulario
Then el sistema actualiza el cliente en la base de datos
And muestra un mensaje de confirmación

Scenario: Eliminar Cliente

Given que el Personal navega a la página de administración de clientes
And selecciona un cliente para eliminar
And confirma la eliminación
Then el sistema elimina el cliente de la base de datos
And muestra un mensaje de confirmación

Resultado del Escenario

```
Feature: Administrar Clientes
  Como Personal administrativo
  Quiero administrar clientes en el sistema
  Para poder crear, visualizar, actualizar y eliminar clientes

  Scenario: Crear Cliente
    Given que el Personal navega a la página de administración de clientes
    And selecciona "Crear nuevo cliente"
    And no ingresa datos enviando el formulario
    And envía el formulario
    Then el sistema muestra un mensaje de error "los campos del formulario no pueden estar vacíos"
    And completa el formulario con la información del cliente
    And envía el formulario
    Then el sistema guarda el nuevo cliente en la base de datos
    And muestra un mensaje de confirmación
    | Cliente creado correctamente

    # features\administrar_clientes.feature:6
    # ClienteContext::queElPersonalNavegaALaPaginaDeAdministracionDeClientes()
    # ClienteContext::seleccionaCrearNuevoCliente()
    # ClienteContext::noIngresaDatosEnviandoElFormulario()
    # ClienteContext::enviaElFormulario()
    # ClienteContext::elSistemaMuestraUnMensajeDeError()
    # ClienteContext::completaElFormularioConLaInformacionDelCliente()
    # ClienteContext::enviaElFormulario()
    # ClienteContext::elSistemaGuardaElNuevoClienteEnLaBaseDeDatos()
    # ClienteContext::muestraUnMensajeDeConfirmacion()

  Scenario: Visualizar Clientes
    Given que el Personal navega a la página de administración de clientes
    Then el sistema muestra la lista de clientes disponibles

    # features\administrar_clientes.feature:17
    # ClienteContext::queElPersonalNavegaALaPaginaDeAdministracionDeClientes()
    # ClienteContext::elSistemaMuestraLaListaDeClientesDisponibles()

  Scenario: Actualizar Cliente
    Given que el Personal navega a la página de administración de clientes
    And selecciona un cliente para actualizar
    And modifica la información del cliente
    And envía el formulario
    Then el sistema actualiza el cliente en la base de datos
    And muestra un mensaje de confirmación
    | Cliente creado correctamente

    # features\administrar_clientes.feature:21
    # ClienteContext::queElPersonalNavegaALaPaginaDeAdministracionDeClientes()
    # ClienteContext::seleccionaUnClienteParaActualizar()
    # ClienteContext::modificaLaInformacionDelCliente()
    # ClienteContext::enviaElFormulario()
    # ClienteContext::elSistemaActualizaElClienteEnLaBaseDeDatos()
    # ClienteContext::muestraUnMensajeDeConfirmacion()

  Scenario: Eliminar Cliente
    Given que el Personal navega a la página de administración de clientes
    And selecciona un cliente para eliminar
    And confirma la eliminación
    Then el sistema elimina el cliente de la base de datos
    And muestra un mensaje de confirmación
    | Cliente creado correctamente

    # features\administrar_clientes.feature:29
    # ClienteContext::queElPersonalNavegaALaPaginaDeAdministracionDeClientes()
    # ClienteContext::seleccionaUnClienteParaEliminar()
    # ClienteContext::confirmaLaEliminacion()
    # ClienteContext::elSistemaEliminaElClienteDeLaBaseDeDatos()
    # ClienteContext::muestraUnMensajeDeConfirmacion()
```

Feature: Administrar Permisos

As a Personal administrativo

I want to administrar los permisos de los usuarios en el sistema

So that I can otorgar y revocar permisos

Scenario: Otorgar y Revocar Permisos

Given el Personal navega a la página de administración de permisos
And selecciona los permisos a otorgar o revocar
Then el sistema actualiza los permisos en la base de datos
And muestra un mensaje de confirmación para permisos

Resultado del Escenario



```
Feature: Administrar Permisos
As a Personal administrativo
I want to administrar los permisos de los usuarios en el sistema
So that I can otorgar y revocar permisos

Escenario: Otorgar y Revocar Permisos
  Given el Personal navega a la página de administración de permisos
  And selecciona los permisos a otorgar o revocar
  Then el sistema actualiza los permisos en la base de datos
  And muestra un mensaje de confirmación para permisos
  | Permisos actualizados correctamente

# features\administrar_permisos.feature:6
# PermisosContext::elPersonalNavegaALaPaginaDeAdministracionDePermisos()
# PermisosContext::seleccionaLosPermisosAOtorgarORevocar()
# PermisosContext::elSistemaActualizaLosPermisosEnLaBaseDeDatos()
# PermisosContext::muestraUnMensajeDeConfirmacionParaPermisos()
```

Feature: Administrar Productos

As a Personal administrativo

I want to administrar productos en el sistema

So that I can crear, visualizar, actualizar y eliminar productos

Scenario: Crear Producto

Given el Personal ha navegado a la página de administración de productos

When crea un nuevo producto con nombre "Nuevo Producto", precio "100" y descripción "Descripción del nuevo producto"

Then el sistema debería guardar el nuevo producto en la base de datos

And debería mostrar un mensaje de confirmación

Scenario: Visualizar Productos

Given el Personal ha navegado a la página de administración de productos

Then el sistema debería mostrar la lista de productos disponibles

Scenario: Actualizar Producto

Given el Personal ha navegado a la página de administración de productos

And hay un producto llamado "Producto a Actualizar"

When actualiza el producto con nombre "Producto a Actualizar" para que tenga precio "150" y descripción "Descripción actualizada del producto"

Then el sistema debería actualizar el producto en la base de datos

And debería mostrar un mensaje de confirmación

Scenario: Eliminar Producto

Given el Personal ha navegado a la página de administración de productos

And hay un producto llamado "Producto a Eliminar"

When elimina el producto con nombre "Producto a Eliminar"

Then el sistema debería eliminar el producto de la base de datos

And debería mostrar un mensaje de confirmación

Resultado del Escenario



```
Feature: Administrar Productos
As a Personal administrativo
I want to administrar productos en el sistema
So that I can crear, visualizar, actualizar y eliminar productos

Scenario: Crear Producto
  Given el Personal ha navegado a la página de administración de productos
  When crea un nuevo producto con nombre "Nuevo Producto", precio "100" y descripción "Descripción del nuevo producto"
  Then el sistema debería guardar el nuevo producto en la base de datos
  And debería mostrar un mensaje de confirmación
  | Producto creado correctamente

Scenario: Visualizar Productos
  Given el Personal ha navegado a la página de administración de productos
  Then el sistema debería mostrar la lista de productos disponibles
  |
  | Array
  | (
  |   [0] => Producto1
  |   [1] => Producto2
  | )

Scenario: Actualizar Producto
  Given el Personal ha navegado a la página de administración de productos
  And hay un producto llamado "Producto a Actualizar"
  When actualiza el producto con nombre "Producto a Actualizar" para que tenga precio "150" y descripción "Descripción actualizada del producto"
  Then el sistema debería actualizar el producto en la base de datos
  And debería mostrar un mensaje de confirmación
  | Producto creado correctamente

Scenario: Eliminar Producto
  Given el Personal ha navegado a la página de administración de productos
  And hay un producto llamado "Producto a Eliminar"
  When elimina el producto con nombre "Producto a Eliminar"
  Then el sistema debería eliminar el producto de la base de datos
  And debería mostrar un mensaje de confirmación
  | Producto creado correctamente

# features\administrar_productos.feature:6
# ProductosContext::elPersonalHaNavegadoALaPaginaDeAdministracionDeProductos()
# ProductosContext::crearNuevoProducto()
# ProductosContext::elSistemaDeberiaGuardarElNuevoProductoEnLaBaseDeDatos()
# ProductosContext::deberiaMostrarUnMensajeDeConfirmacion()

# features\administrar_productos.feature:12
# ProductosContext::elPersonalHaNavegadoALaPaginaDeAdministracionDeProductos()
# ProductosContext::elSistemaDeberiaMostrarLaListaDeProductosDisponibles()

# features\administrar_productos.feature:16
# ProductosContext::elPersonalHaNavegadoALaPaginaDeAdministracionDeProductos()
# ProductosContext::hayUnProductoLlamado()
# ProductosContext::actualizarElProducto()
# ProductosContext::elSistemaDeberiaActualizarElProductoEnLaBaseDeDatos()
# ProductosContext::deberiaMostrarUnMensajeDeConfirmacion()

# features\administrar_productos.feature:23
# ProductosContext::elPersonalHaNavegadoALaPaginaDeAdministracionDeProductos()
# ProductosContext::hayUnProductoLlamado()
# ProductosContext::eliminarElProducto()
# ProductosContext::elSistemaDeberiaEliminarElProductoDeLaBaseDeDatos()
# ProductosContext::deberiaMostrarUnMensajeDeConfirmacion()
```

Feature: Administrar Roles

As a Personal administrativo

I want to administrar los roles de los usuarios en el sistema

So that I can otorgar y modificar roles

Scenario: Otorgar y Modificar Roles

Given el Personal navega a la página de administración de roles

And selecciona el rol a otorgar o modificar

Then el sistema actualiza el rol en la base de datos

And muestra un mensaje de confirmación para roles

Resultado del Escenario

```
Feature: Administrar Roles
As a Personal administrativo
I want to administrar los roles de los usuarios en el sistema
So that I can otorgar y modificar roles

Scenario: Otorgar y Modificar Roles
  Given el Personal navega a la página de administración de roles
  And selecciona el rol a otorgar o modificar
  Then el sistema actualiza el rol en la base de datos
  And muestra un mensaje de confirmación para roles
  | Rol actualizado correctamente

# features\administrar_roles.feature:6
# RolesContext::elPersonalNavegaALaPaginaDeAdministracionDeRoles()
# RolesContext::seleccionaElRolAOtorgarOModificar()
# RolesContext::elSistemaActualizaElRolEnLaBaseDeDatos()
# RolesContext::muestraUnMensajeDeConfirmacionParaRoles()
```

Feature: Administrar Usuarios

As a Personal administrativo

I want to administrar usuarios en el sistema

So that I can crear, visualizar, actualizar y eliminar usuarios

Scenario: Crear Usuario

Given el Personal navega a la página de administración de usuarios

And selecciona "Crear nuevo usuario"

And no ingresa datos enviando el formulario de usuarios

And envía el formulario de usuarios



Then el sistema muestra un mensaje de error "Los campos del formulario de usuarios no pueden estar vacíos"

And completa el formulario con la información del usuario

And envía el formulario de usuarios

Then el sistema guarda el nuevo usuario en la base de datos

And muestra un mensaje de confirmación para usuarios

Scenario: Visualizar Usuarios

Given el Personal navega a la página de administración de usuarios

Then el sistema muestra la lista de usuarios disponibles

Scenario: Actualizar Usuario

Given el Personal navega a la página de administración de usuarios

And selecciona un usuario para actualizar

And modifica la información del usuario

And envía el formulario de usuarios

Then el sistema actualiza el usuario en la base de datos

And muestra un mensaje de confirmación para usuarios

Scenario: Eliminar Usuario

Given el Personal navega a la página de administración de usuarios

And selecciona un usuario para eliminar de la lista

And confirma la eliminación del usuario

Then el sistema elimina el usuario de la base de datos

And muestra un mensaje de confirmación para usuarios

Resultado del Escenario

```
Feature: Administrar Usuarios
  As a Personal administrativo
  I want to administrar usuarios en el sistema
  So that I can crear, visualizar, actualizar y eliminar usuarios

Scenario: Crear Usuario
  Given el Personal navega a la página de administración de usuarios
  And selecciona "Crear nuevo usuario"
  And no ingresa datos enviando el formulario de usuarios
  And envía el formulario de usuarios
  Then el sistema muestra un mensaje de error "Los campos del formulario de usuarios no pueden estar vacíos"
  And completa el formulario con la información del usuario
  And envía el formulario de usuarios
  Then el sistema guarda el nuevo usuario en la base de datos
  And muestra un mensaje de confirmación para usuarios
  | Usuario creado correctamente

Scenario: Visualizar Usuarios
  Given el Personal navega a la página de administración de usuarios
  Then el sistema muestra la lista de usuarios disponibles
  | Array
  | (
  |   [0] -> Usuario1
  |   [1] -> Usuario2
  | )

Scenario: Actualizar Usuario
  Given el Personal navega a la página de administración de usuarios
  And selecciona un usuario para actualizar
  And modifica la información del usuario
  And envía el formulario de usuarios
  Then el sistema actualiza el usuario en la base de datos
  And muestra un mensaje de confirmación para usuarios
  | Usuario creado correctamente

Scenario: Eliminar Usuario
  Given el Personal navega a la página de administración de usuarios
  And selecciona un usuario para eliminar de la lista
  And confirma la eliminación del usuario
  Then el sistema elimina el usuario de la base de datos
  And muestra un mensaje de confirmación para usuarios
  | Usuario creado correctamente

# features\administrar_usuarios.feature:6
# UsuariosContext::elPersonalNavegaALaPaginaDeAdministracionDeUsuarios()
# UsuariosContext::seleccionaCrearNuevoUsuario()
# UsuariosContext::noIngresaDatosEnviandoElFormularioDeUsuarios()
# UsuariosContext::enviaElFormularioDeUsuarios()
# UsuariosContext::elSistemaMuestraUnMensajeDeError()
# UsuariosContext::completaElFormularioConLaInformacionDelUsuario()
# UsuariosContext::enviaElFormularioDeUsuarios()
# UsuariosContext::elSistemaGuardaElNuevoUsuarioEnLaBaseDeDatos()
# UsuariosContext::muestraUnMensajeDeConfirmacionParaUsuarios()

# features\administrar_usuarios.feature:17
# UsuariosContext::elPersonalNavegaALaPaginaDeAdministracionDeUsuarios()
# UsuariosContext::elSistemaMuestraLaListaDeUsuariosDisponibles()

# features\administrar_usuarios.feature:21
# UsuariosContext::elPersonalNavegaALaPaginaDeAdministracionDeUsuarios()
# UsuariosContext::seleccionaUnUsuarioParaActualizar()
# UsuariosContext::modificaLaInformacionDelUsuario()
# UsuariosContext::enviaElFormularioDeUsuarios()
# UsuariosContext::elSistemaActualizaElUsuarioEnLaBaseDeDatos()
# UsuariosContext::muestraUnMensajeDeConfirmacionParaUsuarios()

# features\administrar_usuarios.feature:29
# UsuariosContext::elPersonalNavegaALaPaginaDeAdministracionDeUsuarios()
# UsuariosContext::seleccionaUnUsuarioParaEliminarDeLaLista()
# UsuariosContext::confirmaLaEliminacionDelUsuario()
# UsuariosContext::elSistemaEliminaElUsuarioDeLaBaseDeDatos()
# UsuariosContext::muestraUnMensajeDeConfirmacionParaUsuarios()
```



Feature: Administrar Ventas

As a Personal administrativo

I want to administrar ventas en el sistema

So that I can visualizar, actualizar y eliminar ventas

Scenario: Visualizar Ventas

Given el Personal navega a la página de administración de ventas

Then el sistema muestra la lista de ventas disponibles

Scenario: Actualizar Venta

Given el Personal navega a la página de administración de ventas

And selecciona una venta para actualizar

And modifica la información de la venta con datos incompletos

And envía el formulario de ventas

Then el sistema muestra un mensaje de error para ventas "Los campos del formulario no pueden estar vacíos"

And completa el formulario con la información correcta de la venta

And envía el formulario de ventas

Then el sistema actualiza la venta en la base de datos

And muestra un mensaje de confirmación para ventas actualizado

Scenario: Eliminar Venta

Given el Personal navega a la página de administración de ventas

And selecciona una venta para eliminar

And confirma la eliminación de la venta

Then el sistema elimina la venta de la base de datos

And muestra un mensaje de confirmación para ventas eliminado

Resultado del Escenario



```
Feature: Administrar Ventas
  As a Personal administrativo
  I want to administrar ventas en el sistema
  So that I can visualizar, actualizar y eliminar ventas

Escenario: Visualizar Ventas
  Given el Personal navega a la página de administración de ventas
  Then el sistema muestra la lista de ventas disponibles
  Array
  (
    [0] => Venta1
    [1] => Venta2
  )

Escenario: Actualizar Venta
  Given el Personal navega a la página de administración de ventas
  And selecciona una venta para actualizar
  And modifica la información de la venta con datos incompletos
  And envía el formulario de ventas
  | Los campos del formulario no pueden estar vacíos
  Then el sistema muestra un mensaje de error para ventas "Los campos del formulario no pueden estar vacíos"
  | Los campos del formulario no pueden estar vacíos
  And completa el formulario con la información correcta de la venta
  And envía el formulario de ventas
  | Los campos del formulario no pueden estar vacíos
  Then el sistema actualiza la venta en la base de datos
  Then el sistema actualiza la venta en la base de datos
  Then el sistema actualiza la venta en la base de datos
  Then el sistema actualiza la venta en la base de datos
  Then el sistema actualiza la venta en la base de datos
  Then el sistema actualiza la venta en la base de datos
  Then el sistema actualiza la venta en la base de datos
  Then el sistema actualiza la venta en la base de datos
  Then el sistema actualiza la venta en la base de datos
  | Venta actualizada correctamente
  And muestra un mensaje de confirmación para ventas actualizado
  | Venta actualizada correctamente

Escenario: Eliminar Venta
  Given el Personal navega a la página de administración de ventas
  And selecciona una venta para eliminar
  And confirma la eliminación de la venta
  Then el sistema elimina la venta de la base de datos
  | Venta eliminada correctamente
  And muestra un mensaje de confirmación para ventas eliminado
  | Venta eliminada correctamente

17 escenarios (17 passed)
84 steps (84 passed)
0m1.09s (10.15%)
```

Reporte general BDD

Automated test report - 11/07/2024 16:07:05

6 features (6 success)
17 escenarios (17 success)
84 steps (84 success)
0m1.09s - 12.63MB

Suite: default

Feature: Administrar Clientes |+

Como Personal administrativo Quiero administrar clientes en el sistema Para poder crear, visualizar, actualizar y eliminar clientes

Feature has passed

Feature: Administrar Permisos |+

As a Personal administrativo I want to administrar los permisos de los usuarios en el sistema So that I can otorgar y revocar permisos

Feature has passed

Feature: Administrar Productos |+

As a Personal administrativo I want to administrar productos en el sistema So that I can crear, visualizar, actualizar y eliminar productos

Feature has passed

Feature: Administrar Roles |+

As a Personal administrativo I want to administrar los roles de los usuarios en el sistema So that I can otorgar y modificar roles

Feature has passed

Feature: Administrar Usuarios |+

As a Personal administrativo I want to administrar usuarios en el sistema So that I can crear, visualizar, actualizar y eliminar usuarios

Feature has passed

Feature: Administrar Ventas |+

As a Personal administrativo I want to administrar ventas en el sistema So that I can visualizar, actualizar y eliminar ventas

Feature has passed



6 Features:

0 failed

17 Scenarios:

0 failed

0 undefined

0 skipped

84 Steps:

0 failed

0 undefined

0 skipped

All Passed Failed

Suite: default

Feature: Administrar Clientes |+

Como Personal administrativo
Quiero administrar clientes en el sistema
Para poder crear, visualizar, actualizar y eliminar clientes

Feature: Administrar Permisos |+

As a Personal administrativo
I want to administrar los permisos de los usuarios en el sistema
So that I can otorgar y revocar permisos

Feature: Administrar Productos |+

As a Personal administrativo
I want to administrar productos en el sistema
So that I can crear, visualizar, actualizar y eliminar productos

Feature: Administrar Roles |+

As a Personal administrativo
I want to administrar los roles de los usuarios en el sistema
So that I can otorgar y modificar roles

Feature: Administrar Usuarios |+

As a Personal administrativo
I want to administrar usuarios en el sistema
So that I can crear, visualizar, actualizar y eliminar usuarios

Feature: Administrar Ventas |+

As a Personal administrativo
I want to administrar ventas en el sistema
So that I can visualizar, actualizar y eliminar ventas