



UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERIA

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**

CONTROL DE VERSIONES					
Versión	Hecha por	Revisada por	Aprobada por	Fecha	Motivo
1.0	Ronal Lupaca Mamani			11/07/2024	Creación del documentó sad con observaciones de coberturas pruebas BDD y Test

**Sistema *Proyecto de implementación de un sistema de
ventas para grandes almacenes***
Documento de Arquitectura de Software

Versión 1.5

CONTROL DE VERSIONES					
Versión	Hecha por	Revisada por	Aprobada por	Fecha	Motivo
1.0	MPV	ELV	ARV	11/07/2024	Versión Original

INDICE GENERAL

Contenido

1.	INTRODUCCIÓN.....	4
1.1.	Propósito (Diagrama 4+1).....	4
1.2.	Alcance.....	4
1.3.	Definición, siglas y abreviaturas	5
2.	OBJETIVOS Y RESTRICCIONES ARQUITECTONICAS.....	6
2.1.1.	Requerimientos Funcionales	6
2.1.2.	Requerimientos No Funcionales – Atributos de Calidad.....	6
3.	REPRESENTACIÓN DE LA ARQUITECTURA DEL SISTEMA.....	8
3.1.	Vista de Caso de uso	8
3.1.1.	Diagramas de Casos de uso	8
3.2.	Vista Lógica.....	9
3.2.1.	Diagrama de Subsistemas (paquetes)	9
3.2.2.	Diagrama de Secuencia (vista de diseño)	9
3.2.3.	Diagrama de Objetos.....	12
3.2.4.	Diagrama de Clases	12
3.2.5.	Diagrama de Base de datos (relacional o no relacional)	13
3.3.	Vista de Implementación (vista de desarrollo)	14
3.3.1.	Diagrama de arquitectura software (paquetes).....	14
3.3.2.	Diagrama de arquitectura del sistema (Diagrama de componentes)	14
4.	ATRIBUTOS DE CALIDAD DEL SOFTWARE	15
4.1.	Informe de cobertura	15
4.2.	Informe de ejecución de Pruebas.....	26

1. INTRODUCCIÓN

1.1. Propósito (Diagrama 4+1)

El propósito del sistema de ventas para grandes almacenes es modernizar y optimizar los procesos de ventas y gestión de productos en grandes almacenes, como Plaza Vea. Para lograr esto, utilizamos el marco de trabajo de vistas arquitectónicas 4+1, que se enfoca en diferentes perspectivas del sistema para garantizar una solución completa y robusta.

La Vista Lógica se centra en la funcionalidad del sistema desde el punto de vista de los desarrolladores. Aquí, se incluyen diagramas de clases, paquetes y casos de uso que detallan cómo interactúan los componentes del sistema entre sí. Esta vista permite a los desarrolladores comprender mejor cómo se estructura el sistema y cómo deben trabajar juntos los distintos módulos.

La Vista de Desarrollo muestra cómo se organizan los componentes del sistema en módulos o subsistemas. Incluye diagramas que explican la estructura y agrupación de los componentes, facilitando la gestión del desarrollo y mantenimiento del sistema.

La Vista de Procesos describe el comportamiento dinámico del sistema. Esta perspectiva incluye diagramas de secuencia y de actividad que muestran cómo se gestionan las interacciones y el flujo de datos durante la ejecución del sistema. Es crucial para entender cómo se comporta el sistema en tiempo real y cómo responde a diferentes eventos y acciones de los usuarios.

La Vista Física representa la distribución del hardware en el que se despliega el sistema. Los diagramas de despliegue ilustran cómo se distribuyen los componentes del sistema en diferentes nodos de hardware, garantizando que el sistema funcione de manera eficiente y segura en su entorno operativo.

Finalmente, los Casos de Uso proporcionan descripciones detalladas de los escenarios de uso del sistema desde la perspectiva del usuario final. Esta vista asegura que se comprendan claramente los requisitos funcionales del sistema y que se diseñen soluciones que cumplan con las expectativas y necesidades de los usuarios.

1.2. Alcance

El proyecto de implementación de un sistema de ventas para grandes almacenes abarca varias áreas clave para garantizar una gestión eficiente y moderna de las ventas y productos. El sistema facilitará la recepción y procesamiento de ventas a través de múltiples cajas de atención, lo que permitirá un flujo eficiente y rápido de transacciones, reduciendo los tiempos de espera y mejorando la experiencia del cliente. Además, implementará un control de inventarios en tiempo real, permitiendo la actualización automática del stock de productos al momento de la venta, asegurando así que siempre se tenga información precisa y actualizada sobre la disponibilidad de productos, evitando problemas de sobreventa o falta de stock.

La generación de recibos será automatizada, lo que mejorará la eficiencia del proceso y reducirá los errores humanos, facilitando el trabajo del personal y proporcionando a los clientes recibos precisos y profesionales. La gestión de usuarios permitirá administrar usuarios mayoristas y personal administrativo de manera eficiente, asignando roles y permisos específicos según las necesidades, garantizando que solo las personas autorizadas tengan acceso a determinadas funciones y datos, mejorando la seguridad y la gestión interna.

La administración de personal será optimizada mediante herramientas que faciliten la gestión de turnos y tareas del personal que trabaja en el almacén, mejorando la organización y la productividad del equipo. Finalmente, se utilizarán herramientas de análisis estático de código como Snyk, SonarQube y SonarCloud para identificar y

corregir vulnerabilidades, asegurando que el sistema sea robusto y seguro, protegiendo tanto los datos del almacén como la información de los clientes.

1.3. Definición, siglas y abreviaturas

- a) MPV (Modelo de Procesos y Vistas): Es una metodología para la representación arquitectónica de sistemas que combina modelos de procesos y vistas estructurales.
- b) ELV (Elemento Lógico de Vista): Componente que define los aspectos funcionales del sistema desde una perspectiva lógica.
- c) ARV (Arquitectura de Referencia de Vistas): Conjunto de vistas arquitectónicas que proporcionan una estructura para la documentación y diseño del sistema.
- d) Snyk: Herramienta de seguridad para el análisis de vulnerabilidades en las dependencias y código de proyectos.
- e) SonarQube: Plataforma para la revisión continua del código, detectando bugs, vulnerabilidades y deuda técnica.
- f) SonarCloud: Servicio en la nube para análisis automatizado del código, enfocado en la detección y corrección de fallas.
- g) CI/CD (Integración Continua/Despliegue Continuo): Prácticas que permiten la integración y despliegue automático y sistemático de cambios en el código.
- h) MVC (Modelo-Vista-Controlador): Patrón de diseño utilizado para implementar interfaces de usuario dividiendo la lógica de la aplicación en tres componentes interconectados.

2. OBJETIVOS Y RESTRICCIONES ARQUITECTONICAS

[Establezca las prioridades de los requerimientos y las restricciones del proyecto]

2.1. Priorización de requerimientos

2.1.1. Requerimientos Funcionales

Nº	Requerimiento	Descripción	Prioridad
RF-01	Inicio de Sesión	Los usuarios deben poder iniciar sesión en el sistema proporcionando su nombre de usuario y contraseña.	Alta
RF-02	Crear Usuario	El sistema debe permitir a los usuarios registrarse proporcionando información básica como nombre, correo electrónico y contraseña.	Alta
RF-03	Gestión de Productos	El sistema debe permitir a los usuarios agregar, editar y eliminar productos, incluyendo detalles como código, descripción, precio y existencia.	Alta
RF-04	Gestión de Clientes	El sistema debe permitir a los usuarios crear y actualizar información de clientes, como nombre, teléfono, dirección y estado.	Alta
RF-05	Procesar Ventas	Los usuarios deben poder realizar ventas seleccionando productos, asociando clientes y calculando el total de la venta.	Alta
RF-06	Generación de Recibos	El sistema debe generar recibos de ventas que puedan ser impresos o enviados por correo electrónico a los clientes.	Media
RF-07	Gestión de Permisos	El sistema debe permitir asignar diferentes permisos a los usuarios para acceder a distintas funcionalidades del sistema.	Media
RF-08	Informes y Estadísticas	El sistema debe proporcionar informes y estadísticas sobre las ventas realizadas, productos más vendidos, etc.	Media
RF-09	Registro de Detalles de Ventas	El sistema debe registrar los detalles de cada venta, incluyendo productos vendidos, cantidad, precio y total de la venta.	Alta
RF-10	Actualización de Inventario	El sistema debe actualizar automáticamente el inventario de productos después de cada venta.	Alta
RF-11	Búsqueda de Productos	El sistema debe permitir a los usuarios buscar productos por código o descripción.	Media
RF-12	Envío de Notificaciones	El sistema debe enviar notificaciones a los usuarios sobre eventos importantes, como bajas existencias de productos.	Baja
RF-13	Gestión de Configuraciones	El sistema debe permitir la configuración de parámetros generales del sistema, como nombre, teléfono y dirección de la tienda.	Baja

2.1.2. Requerimientos No Funcionales – Atributos de Calidad

Nº	Requerimiento	Descripción	Prioridad
RNF-01	Seguridad	El sistema debe implementar medidas de seguridad para proteger contra ataques como SQL Injection y Cross-Site Scripting (XSS).	Alta
RNF-02	Rendimiento	El sistema debe ser capaz de responder rápidamente a las solicitudes de los usuarios, con tiempos de carga mínimos.	Alta

RNF-03	Escalabilidad	El sistema debe ser escalable para manejar un incremento en el número de usuarios y datos sin degradar el rendimiento.	Media
RNF-04	Usabilidad	El sistema debe tener una interfaz intuitiva y fácil de usar para los usuarios.	Media
RNF-05	Mantenibilidad	El sistema debe ser fácil de mantener y actualizar, con código bien estructurado y documentado.	Alta
RNF-06	Disponibilidad	El sistema debe estar disponible la mayor parte del tiempo, minimizando los tiempos de inactividad.	Alta
RNF-07	Portabilidad	El sistema debe ser capaz de funcionar en diferentes plataformas sin requerir modificaciones significativas.	Baja
RNF-08	Compatibilidad	El sistema debe poder integrarse con otros sistemas externos sin problemas.	Media
RNF-09	Documentación	El sistema debe contar con documentación completa y detallada para facilitar su uso y mantenimiento.	Media

2.2. Restricciones

Tecnología: El sistema debe ser desarrollado utilizando PHP para la lógica del servidor y MySQL como sistema de gestión de base de datos.

Debe ser compatible con las últimas versiones de los navegadores web más utilizados, como Google Chrome, Mozilla Firefox y Microsoft Edge.

Plazos: El proyecto debe completarse en un plazo de 86 días, comenzando el 5 de abril y terminando el 30 de junio de 2024.

Seguridad: El sistema debe cumplir con las normativas de seguridad y privacidad de datos vigentes, protegiendo la información personal y transaccional de los usuarios.

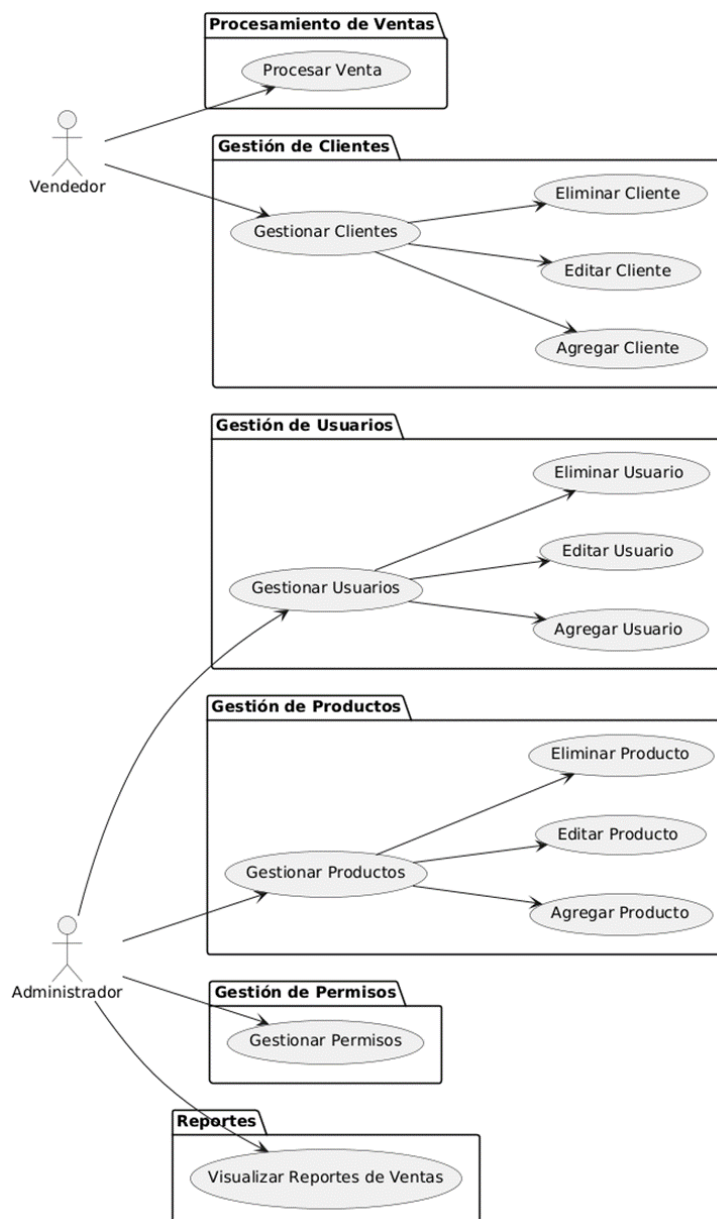
3. REPRESENTACIÓN DE LA ARQUITECTURA DEL SISTEMA

3.1. Vista de Caso de uso

En esta sección se describen los casos de uso del sistema de ventas, abarcando todas las funcionalidades del sistema, mostrando los actores que interactúan en el sistema y las funcionalidades asociadas. Se listan los casos de uso o escenarios del modelo de casos de uso que representan funcionalidades centrales del sistema, que requieren una gran cobertura arquitectónica o aquellos que implican algún punto especialmente delicado de la arquitectura.

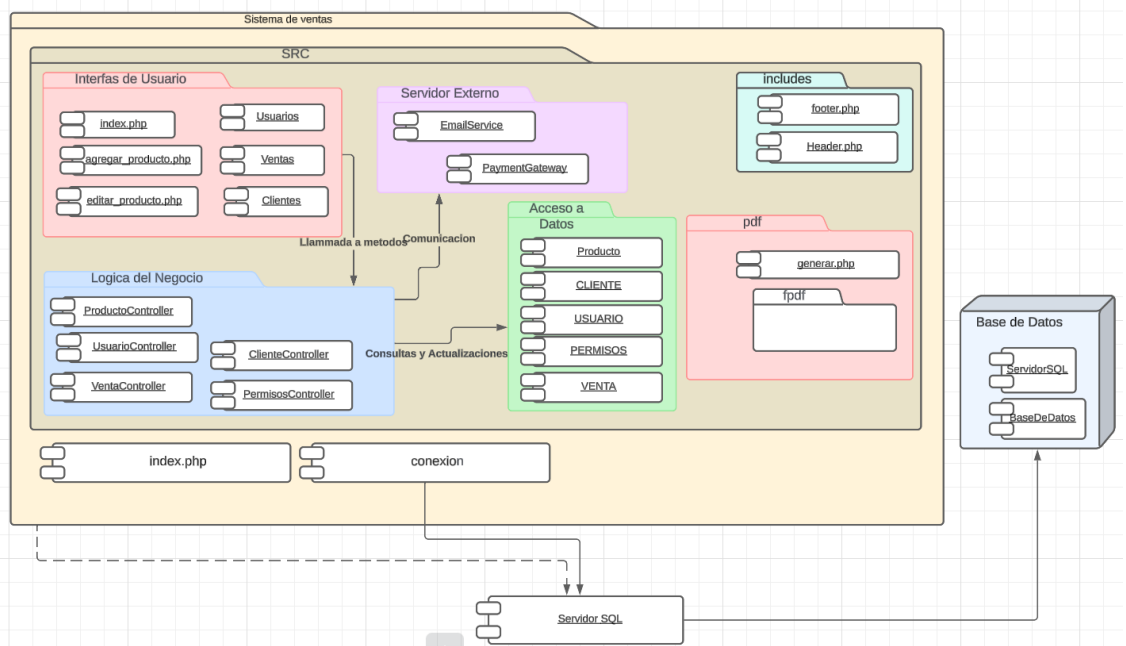
3.1.1. Diagramas de Casos de uso

El diagrama de casos de uso muestra cómo los diferentes actores (Administrador y Vendedor) interactúan con las funcionalidades principales del sistema, tales como gestionar usuarios, gestionar productos, gestionar clientes, procesar ventas, y generar informes.



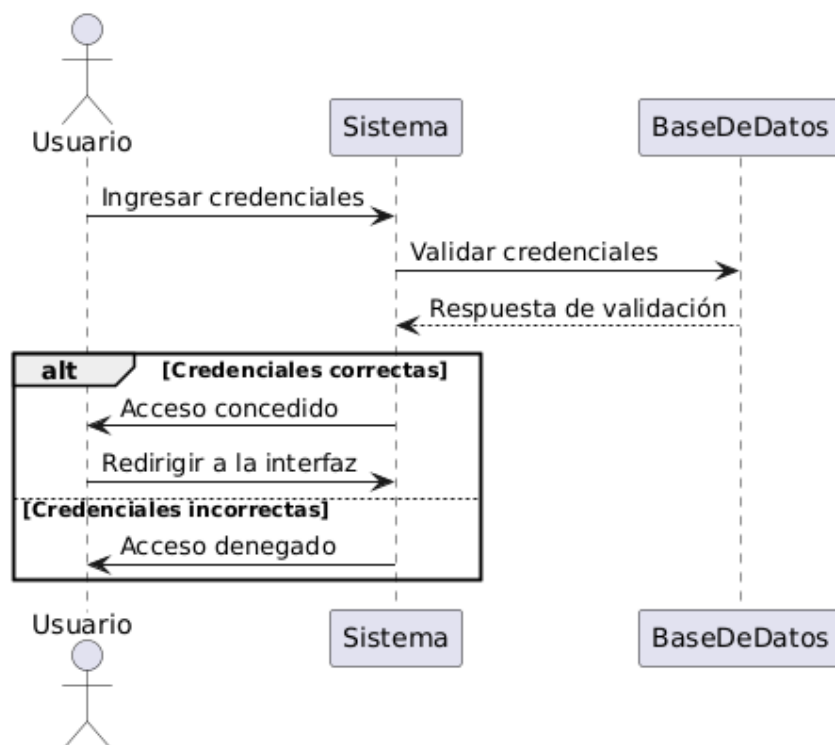
3.2. Vista Lógica

3.2.1. Diagrama de Subsistemas (paquetes)



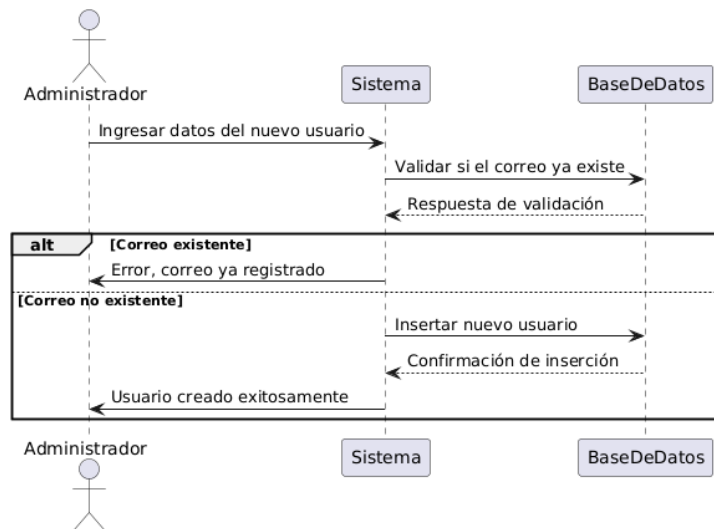
3.2.2. Diagrama de Secuencia (vista de diseño)

Diagrama de Secuencia para el Caso de Uso: Inicio de Sesión



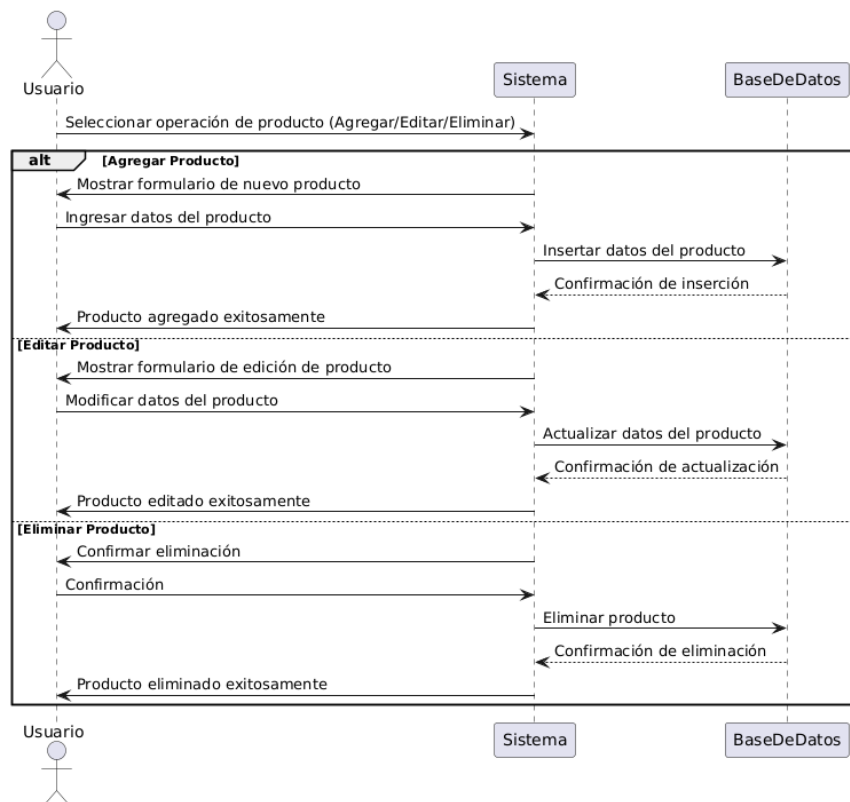
Este diagrama muestra el proceso de autenticación de un usuario en el sistema. Se detalla cómo el usuario ingresa sus credenciales, cómo el sistema valida esas credenciales contra la base de datos, y la respuesta correspondiente dependiendo de la validez de las credenciales.

Diagrama de Secuencia para el Caso de Uso: Crear Usuario

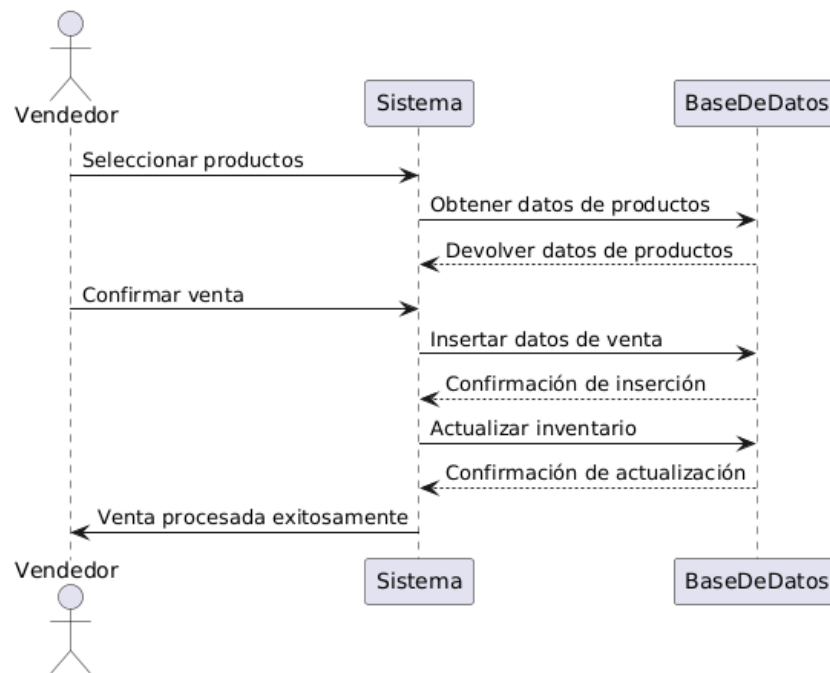


Este diagrama describe el proceso de creación de un nuevo usuario por parte del administrador. Incluye la verificación de que el correo del nuevo usuario no esté ya registrado y la posterior inserción del nuevo usuario en la base de datos.

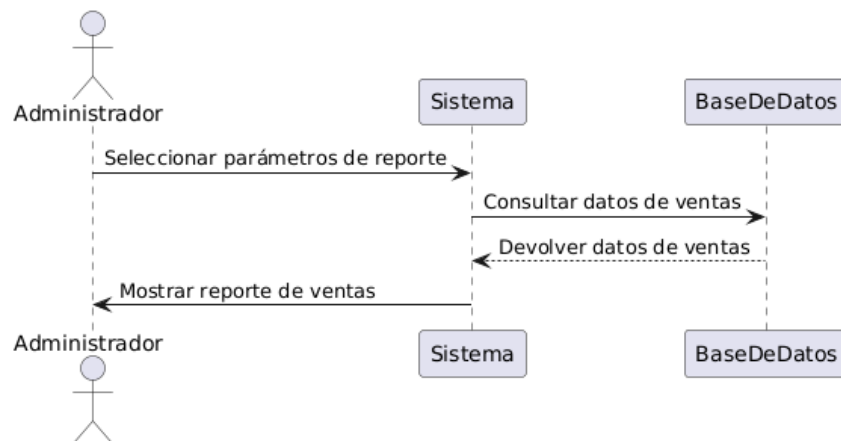
Diagrama de Secuencia para el Caso de Uso: Gestionar Productos



Este diagrama muestra cómo un usuario puede agregar, editar o eliminar productos en el sistema. Incluye la presentación de formularios, la entrada de datos por parte del usuario y la interacción con la base de datos para realizar las operaciones solicitadas.

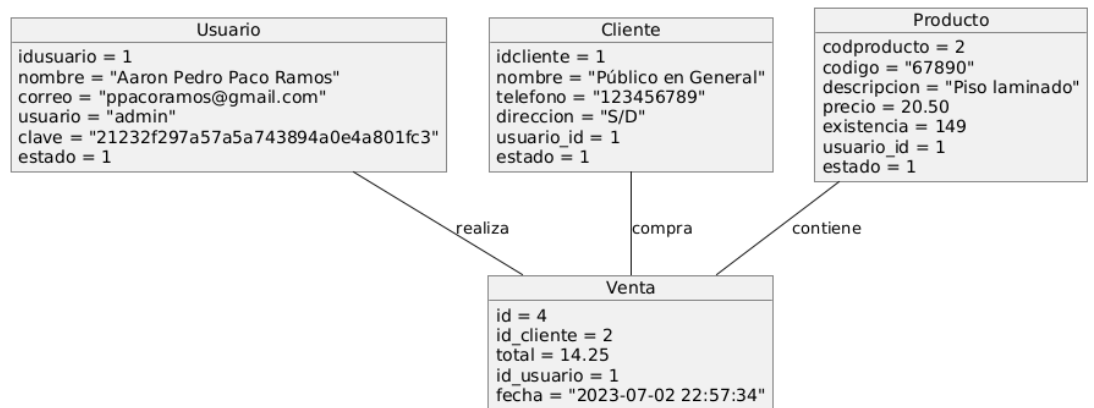
Diagrama de Secuencia para el Caso de Uso: Procesar Venta

Este diagrama describe el proceso de una venta, desde la selección de productos por parte del vendedor hasta la actualización del inventario y la inserción de los datos de la venta en la base de datos.

Diagrama de Secuencia para el Caso de Uso: Generar Reportes de Ventas

Este diagrama ilustra cómo un administrador puede generar reportes de ventas. Muestra la interacción del administrador con el sistema para seleccionar parámetros del reporte y cómo el sistema consulta la base de datos para obtener los datos necesarios.

3.2.3. Diagrama de Objetos



Usuario: Representa una instancia de la clase Usuario con atributos como idusuario, nombre, correo, usuario, clave y estado.

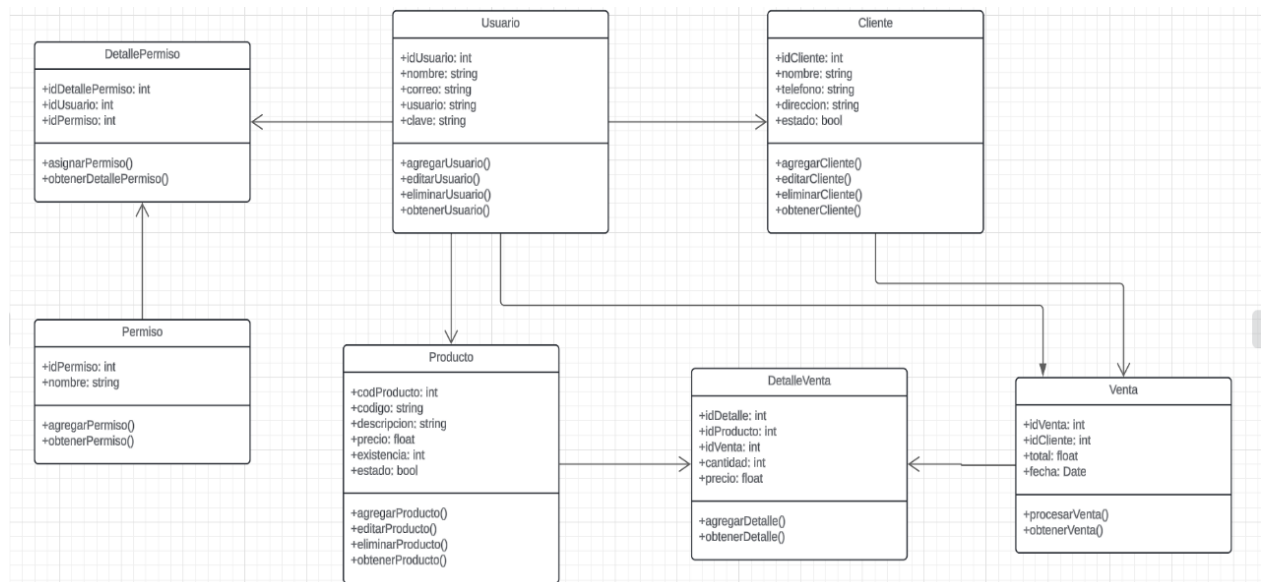
Cliente: Representa una instancia de la clase Cliente con atributos como idcliente, nombre, telefono, direccion, usuario_id y estado.

Producto: Representa una instancia de la clase Producto con atributos como codproducto, codigo, descripcion, precio, existencia, usuario_id y estado.

Venta: Representa una instancia de la clase Venta con atributos como id, id_cliente, total, id_usuario y fecha.

Las relaciones entre estos objetos muestran cómo un Usuario realiza una Venta, un Cliente realiza una compra y un Producto está contenido en una Venta.

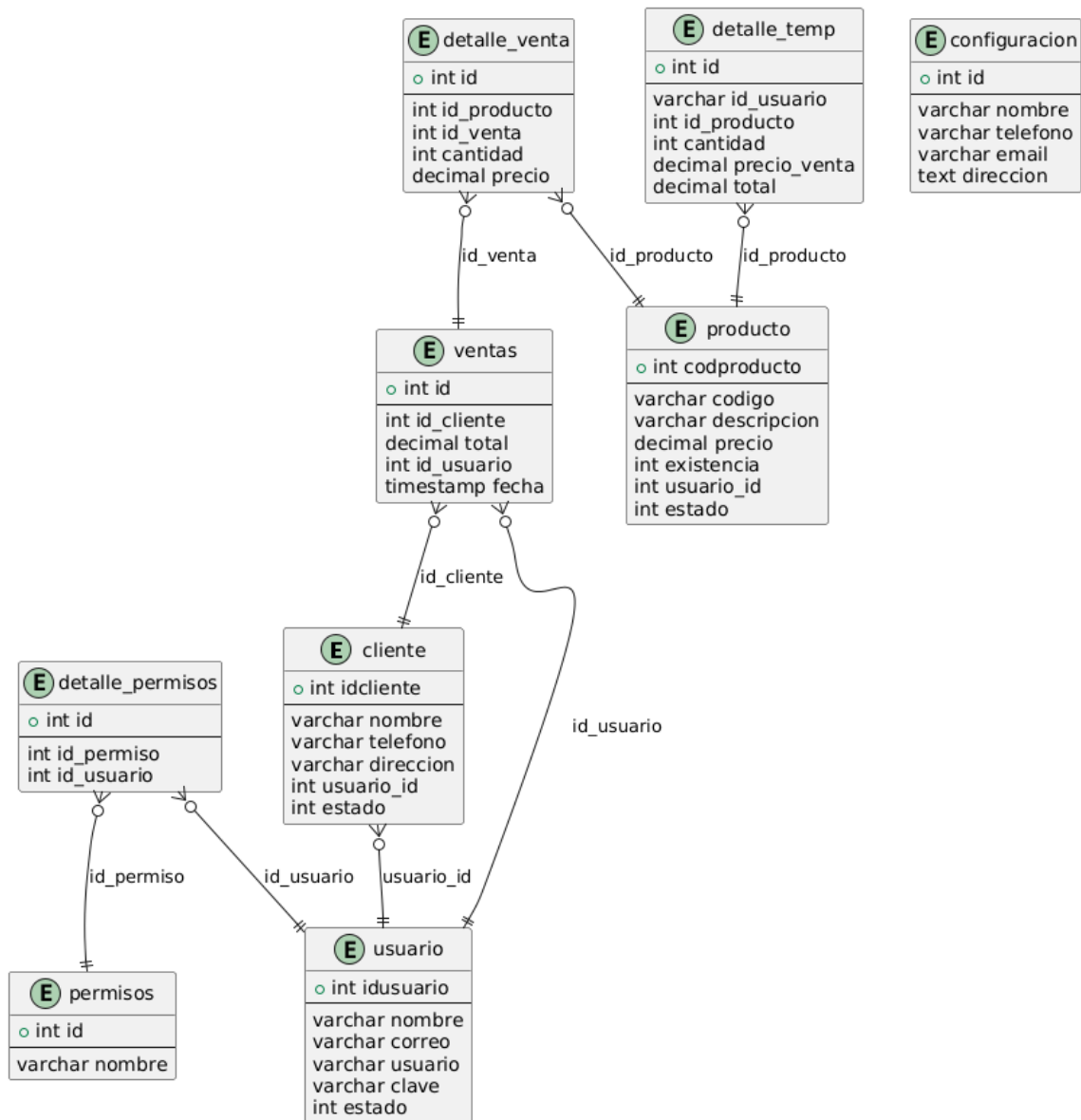
3.2.4. Diagrama de Clases



3.2.5. Diagrama de Base de datos (relacional o no relacional)

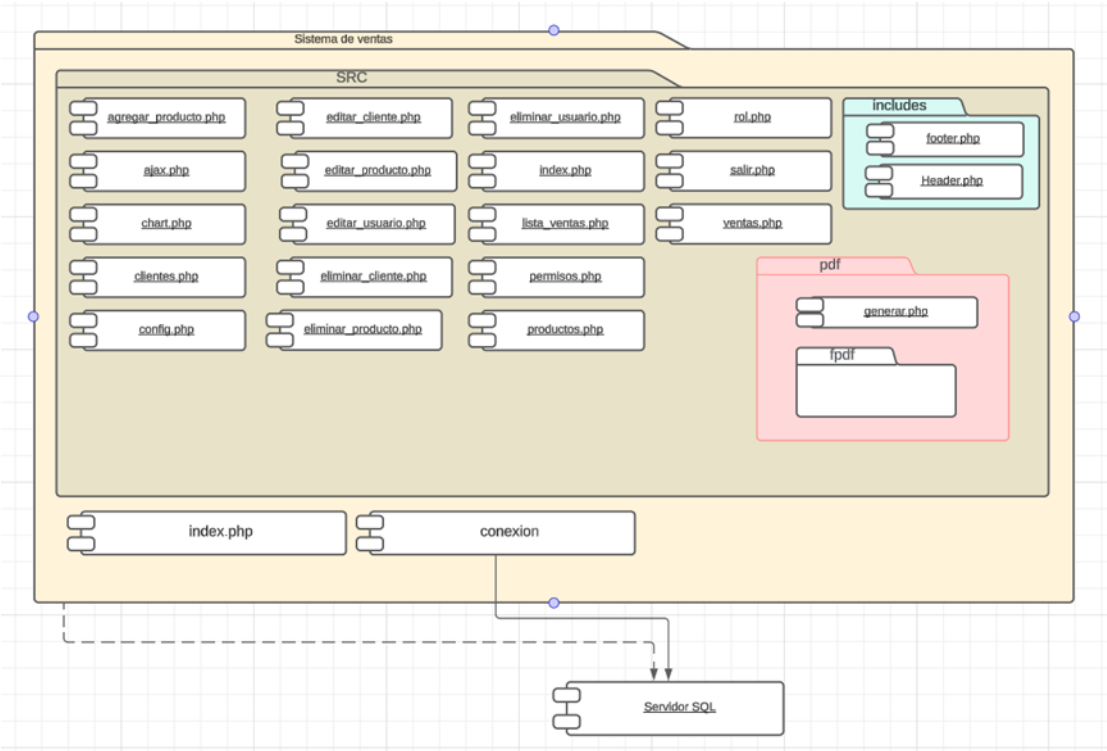
El diagrama de base de datos relacional muestra las tablas de la base de datos junto con sus relaciones, claves primarias y claves foráneas. Este diagrama ayuda a entender cómo se estructuran los datos y cómo están interrelacionados.

A continuación, se presenta el diagrama de la base de datos sis_venta:

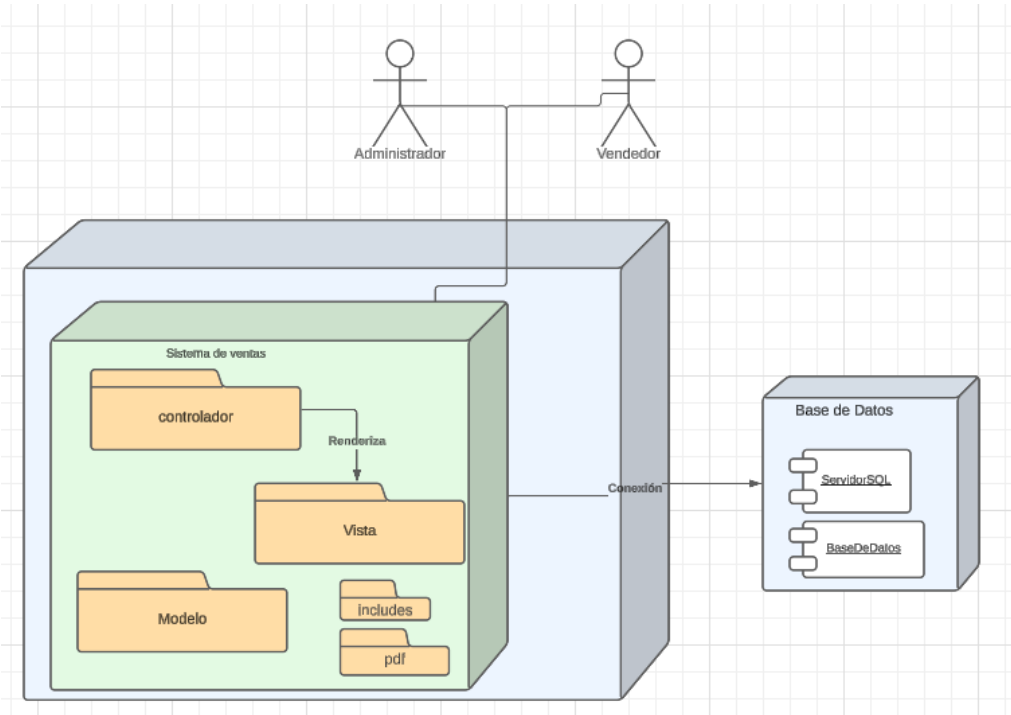


3.3. Vista de Implementación (vista de desarrollo)

3.3.1. Diagrama de arquitectura software (paquetes)



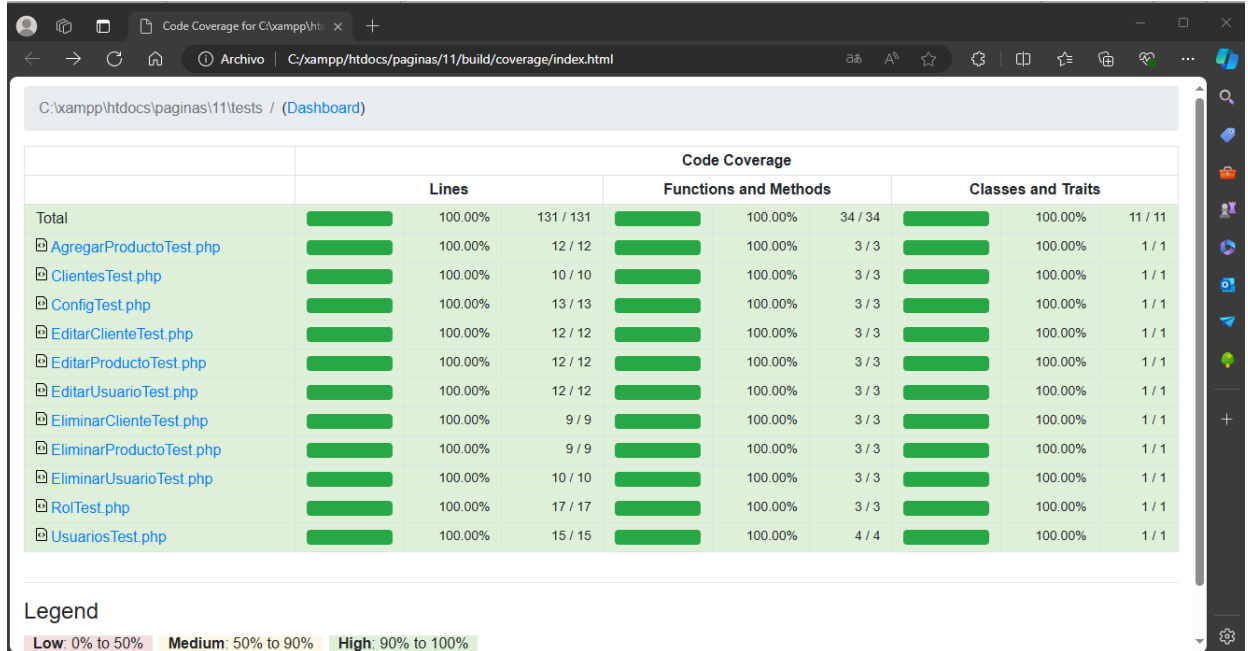
3.3.2. Diagrama de arquitectura del sistema (Diagrama de componentes)



4. ATRIBUTOS DE CALIDAD DEL SOFTWARE

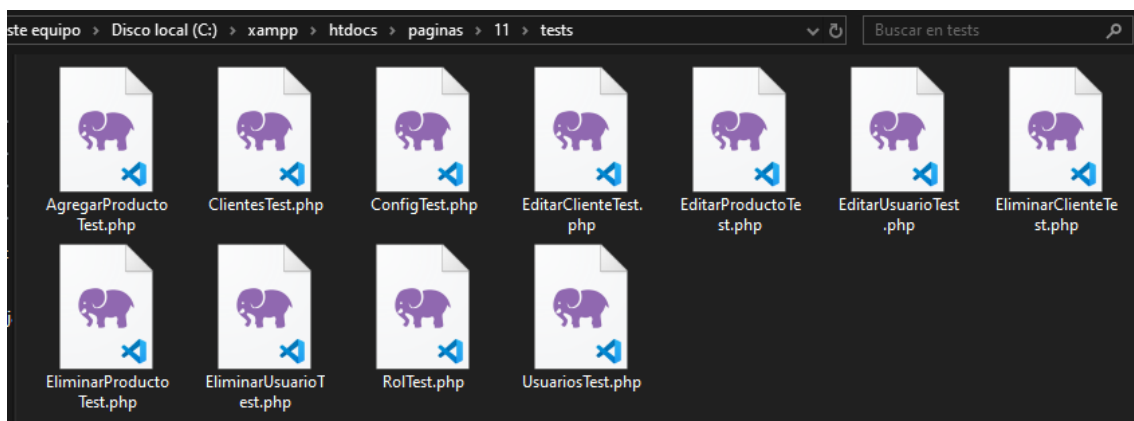
4.1. Informe de cobertura

Utilizando cualquier herramienta que pueda visualizar la cobertura, Sonarqube, coverlet, cobertura, etc.



Se muestra nuestro reporte de cobertura de nuestra aplicación. En el reporte, cada archivo, como `AgregarProductoTest.php`, `CientesTest.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 for C:\xampp\ht...

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

C:\xampp\htdocs\paginas\11\tests / AgregarProductoTest.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	
AgregarProductoTest	<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			
testActualizarProducto	<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 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			
testRegisterNewClient	<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 `testRegisterNewClient` 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	

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.

Propósito del Test:

- **Integridad de Datos:** Asegurar que los datos críticos de la configuración de la empresa se puedan actualizar correctamente y reflejen los cambios esperados.
- **Confianza en el Método:** Probar que el método `actualizarConfiguracion` es fiable y realiza sus funciones como se espera, incluyendo la devolución de un mensaje apropiado que puede ser utilizado para feedback al usuario.
- **Automatización y Regresión:** Facilitar la ejecución automática de pruebas para detectar rápidamente cualquier regresión o error introducido en futuras modificaciones del código.

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	

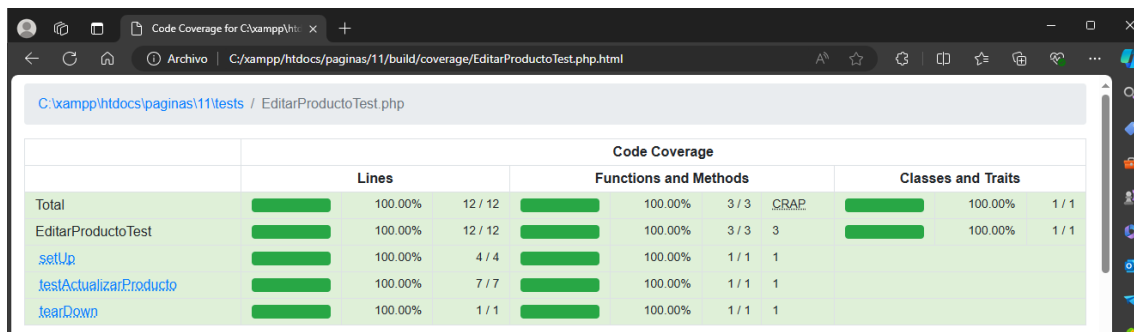
El test `testActualizarCliente` dentro de la clase `EditarClienteTest` está diseñado para comprobar la funcionalidad de actualizar los detalles de un cliente existente en la base de datos a través del método `actualizarCliente` en la clase `ClienteModelTest`. Este test se enfoca en asegurar que los datos 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.

Objetivos del Test:

- **Integridad de Datos:** Confirma que los datos del cliente se pueden actualizar correctamente en la base de datos, lo que es crucial para mantener la precisión y relevancia de la información del cliente.
- **Confirmación del Proceso:** Asegura que el método proporciona feedback adecuado sobre el éxito de la operación, lo cual es esencial para la interacción del usuario final o para la lógica de control en aplicaciones más grandes.
- **Automatización de Regresiones:** Facilita la identificación rápida de regresiones o errores introducidos en el código relacionado con la actualización de clientes, permitiendo correcciones antes de que el código se despliegue en un entorno de producción.

EditarProductoTest.php



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.

Objetivos del Test:

- **Integridad de Datos:** Confirma que los datos del producto se pueden actualizar correctamente en la base de datos, lo que es crucial para mantener la precisión y relevancia de la información del producto.
- **Confirmación del Proceso:** Asegura que el método proporciona feedback adecuado sobre el éxito de la operación, lo cual es esencial para la interacción del usuario final o para la lógica de control en aplicaciones más grandes.
- **Automatización de Regresiones:** Facilita la identificación rápida de regresiones o errores introducidos en el código relacionado con la actualización de productos, permitiendo correcciones antes de que el código se despliegue en un entorno de producción.

EditarUsuarioTest.php

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

<

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.

Objetivos del Test:

- **Integridad de Datos:** Confirma que los datos del usuario se pueden actualizar correctamente en la base de datos, lo que es crucial para mantener la precisión y relevancia de la información del usuario.
- **Confirmación del Proceso:** Asegura que el método proporciona feedback adecuado sobre el éxito de la operación, lo cual es esencial para la interacción del usuario final o para la lógica de control en aplicaciones más grandes.
- **Automatización de Regresiones:** Facilita la identificación rápida de regresiones o errores introducidos en el código relacionado con la actualización de usuarios, permitiendo correcciones antes de que el código se despliegue en un entorno de producción.

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			

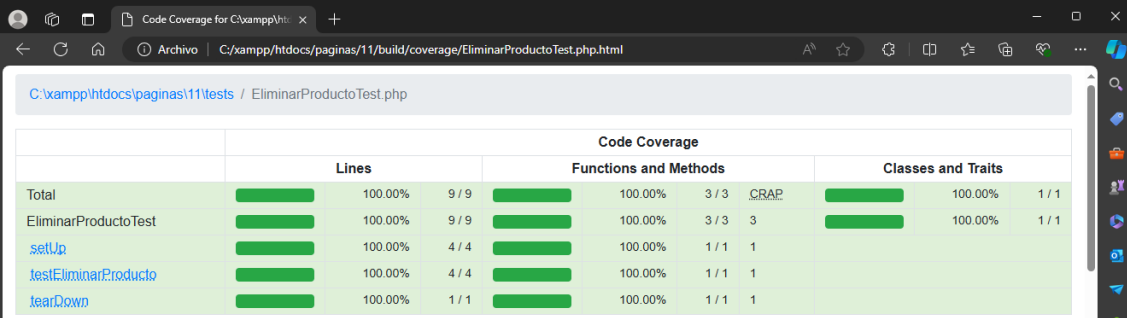
El test `testEliminarCliente` en la clase `EliminarClienteTest` tiene como objetivo verificar la correcta funcionalidad del método `eliminarCliente` de la clase `ClienteModelTest`, el cual está diseñado para cambiar el estado de un cliente en la base de datos, marcándolo como eliminado, en lugar de borrar 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.

Objetivos del Test:

- **Correcta Funcionalidad del Método:** Confirmar que el método `eliminarCliente` realiza correctamente la transición del estado del cliente a 'inactivo' o 'eliminado', lo cual es vital para aplicaciones que requieren un control de estados para sus registros.
- **Integridad de Datos:** Asegurar que el método maneja adecuadamente los cambios de estado sin eliminar completamente los datos, permitiendo una recuperación fácil del estado anterior si es necesario.
- **Automatización de Pruebas:** Facilitar la detección temprana de errores en la implementación de la funcionalidad de eliminación de clientes, ayudando a mantener la calidad y la fiabilidad del software.

EliminarProductoTest.php



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

Objetivos del Test:

- **Correcta Funcionalidad del Método:** Confirmar que el método `eliminarProducto` realiza correctamente el cambio de estado del producto a 'inactivo' o 'eliminado', lo cual es esencial para aplicaciones que necesitan mantener un historial de cambios o revertir estados previos.
- **Integridad de Datos:** Asegurar que el método gestiona adecuadamente los cambios de estado sin eliminar los datos físicamente, permitiendo una fácil restauración del estado previo si es necesario.
- **Automatización de Pruebas:** Facilitar la detección temprana de errores en la implementación de la funcionalidad de eliminación de productos, ayudando a mantener la calidad y la fiabilidad del software.

EliminarUsuarioTest.php

	Code Coverage					
	Lines		Functions and Methods		Classes and Traits	
Total	100.00%	10 / 10	100.00%	3 / 3	CRAP	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.

Objetivos del Test:

- **Correcta Funcionalidad del Método:** Confirmar que el método `eliminarUsuario` realiza correctamente el cambio de estado del usuario a 'inactivo' o 'eliminado', lo cual es vital para aplicaciones que requieren mantener un control de estados para sus registros.
- **Integridad de Datos:** Asegurar que el método maneja adecuadamente los cambios de estado sin eliminar los datos físicamente, permitiendo una fácil restauración del estado previo si es necesario.
- **Automatización de Pruebas:** Facilitar la detección temprana de errores en la implementación de la funcionalidad de eliminación de usuarios, ayudando a mantener la calidad y la fiabilidad del software.

RolTest.php

Code Coverage for 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			

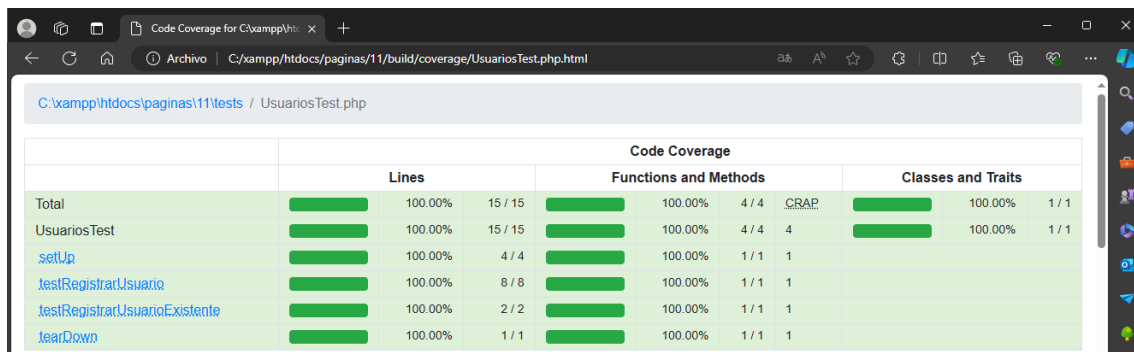
El test `testActualizarPermisos` en la clase `RolTest` está diseñado para verificar la funcionalidad de la clase `RolModelTest`, específicamente el método `actualizarPermisos`, que se encarga de gestionar los permisos asociados a un usuario en la base de datos. Este test tiene como objetivo asegurar 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'.

Objetivos del Test:

- **Funcionalidad Correcta del Método:** Confirmar que el método `actualizarPermisos` realiza correctamente la asignación de permisos, cambiando los permisos del usuario de acuerdo a lo especificado.
- **Integridad de Datos:** Asegurar que los cambios en la asignación de permisos se reflejen adecuadamente en la base de datos, lo que es crucial para el control de acceso dentro de la aplicación.
- **Automatización de Pruebas:** Facilitar la detección temprana de cualquier regresión o error en el manejo de permisos, contribuyendo a mantener la calidad y seguridad del sistema.

UsuariosTest.php



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.

1. **Objetivo:** Asegurar que el sistema maneje correctamente los intentos de registro de un usuario que usa un correo electrónico ya registrado.
2. **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.

Objetivos del Test:

- **Integridad de Datos:** Asegurar que los datos ingresados durante el registro de usuarios se almacenen correctamente y que no haya duplicados no deseados.
- **Manejo de Errores:** Verificar que el sistema identifique y maneje adecuadamente los casos donde pueda intentarse registrar un usuario con datos que violarían las reglas de unicidad, como el correo electrónico.
- **Automatización de Pruebas:** Facilitar la identificación rápida de problemas en el registro de usuarios, permitiendo que se detecten y corrijan antes de que el código se implemente en producción.

4.2. Informe de ejecución de Pruebas

Utilizando cualquier herramienta de visualización de resultados de ejecución, Azure Devops, Github actions, Aws Code Build, Specflow LivinDoc.

Para realizar el informe se usó la dependencia `emuse/BehatHTMLFormatter` es un formateador de informes para Behat que genera informes detallados en formato HTML.

Para instalar esta dependencia se utiliza el siguiente comando →

- `composer require --dev emuse/behat-html-formatter`

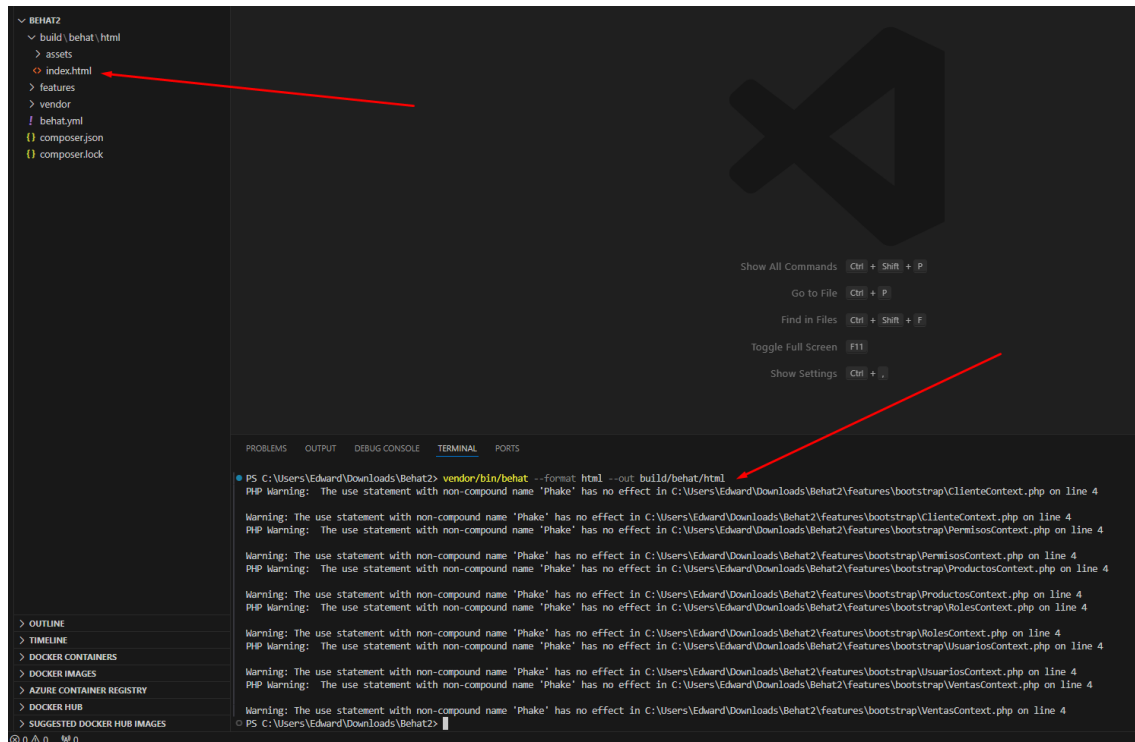
```
PS D:\CALIDAD\Behat2> composer require --dev emuse/behat-html-formatter
./composer.json has been updated
Running composer update emuse/behat-html-formatter
Loading composer repositories with package information
Updating dependencies
Lock file operations: 0 installs, 1 update, 0 removals
  - Upgrading emuse/behat-html-formatter (v0.2.0 => v1.0.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 0 installs, 1 update, 0 removals
  - Downloading emuse/behat-html-formatter (v1.0.0)
  - Upgrading emuse/behat-html-formatter (v0.2.0 => v1.0.0): Extracting archive
Generating autoload files
19 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
No security vulnerability advisories found.
Using version ^1.0 for emuse/behat-html-formatter
PS D:\CALIDAD\Behat2>
```

Luego modificamos el archivo .yml →

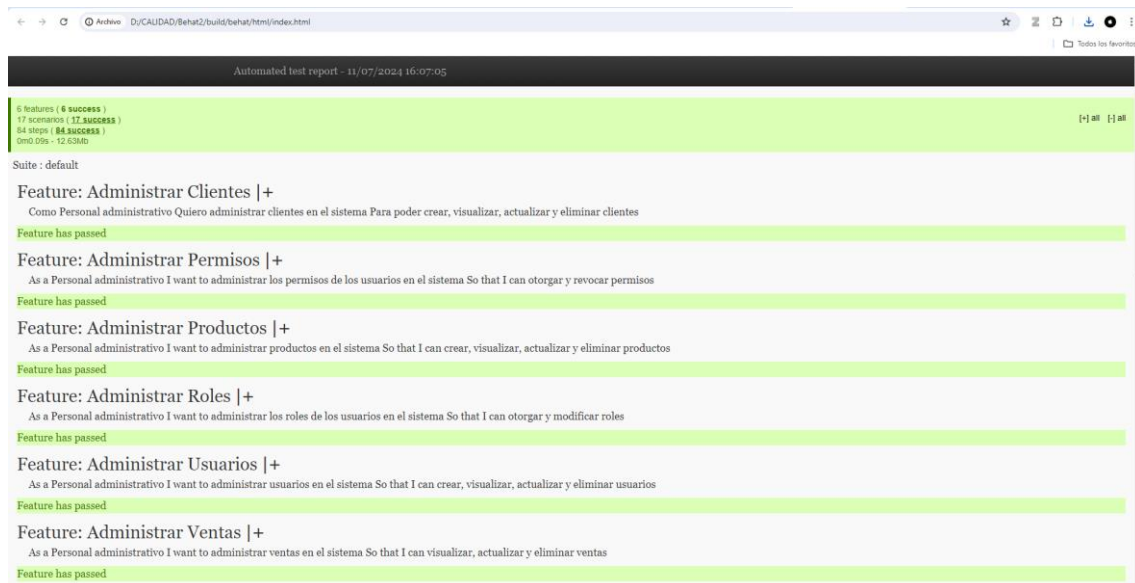
```
! behat.yml x
! behat.yml
1  default:
2  suites:
3  default:
4  paths:
5    - "%paths.base%/features"
6  contexts:
7    - ClienteContext
8    - PermisosContext
9    - ProductosContext
10   - RolesContext
11   - UsuariosContext
12   - VentasContext
13  # formatters:
14  # junit:
15    # output_path: "build/junit"
16  extensions:
17    emuse\BehatHTMLFormatter\BehatHTMLFormatterExtension:
18      name: html
19      renderer: "Twig, Behat2"
20      file_name: index
21      print_args: true
22      print_outp: true
23      loop_break: true
24
```

Por ultimo ejecutamos el siguiente comando →

`vendor/bin/behat --format html --out build/behat/html`



Resultado del Escenario





Desplegar las pruebas automatizadas en una solución de gestión de pruebas, pudiendo utilizar como Azure Devops, AWS Codepipeline, Google Code Build, Github actions, etc.

Creamos la parte .github

Este equipo > Disco local (D:) > CALIDAD > Behat2

Nombre	Fecha de modificación	Tipo	Tamaño
.github	10/07/2024 22:50	Carpeta de archivos	
build	11/07/2024 9:06	Carpeta de archivos	
features	08/07/2024 23:44	Carpeta de archivos	
vendor	11/07/2024 14:03	Carpeta de archivos	
behat.yml	10/07/2024 10:29	Archivo de origen ...	1 KB
composer.json	11/07/2024 14:03	Archivo de origen ...	1 KB
composer.lock	11/07/2024 14:03	Archivo LOCK	78 KB

dentro otra carpeta workflows y ahi el archivo ci.yml →

Este equipo > Disco local (D:) > CALIDAD > Behat2 > .github > workflows

Nombre	Fecha de modificación	Tipo	Tamaño
ci.yml	10/07/2024 23:03	Archivo de origen ...	2 KB

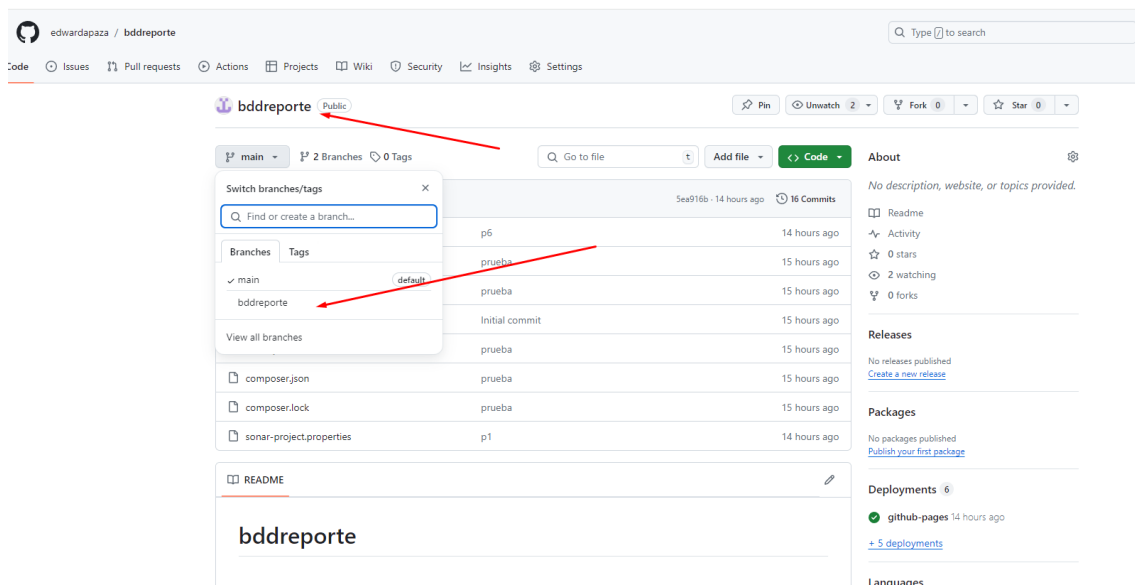
Dentro del archivo ci.yml →

```

github > workflows > ci.yml
1 name: Tarea Automatizada de ejecución de pruebas
2
3 env:
4   SONAR_ORG: 'edwardapaza' # Nombre de la organización en SonarCloud
5   SONAR_PROJECT: 'edwardapaza_bddreporte' # Key ID del proyecto en SonarCloud
6
7 on:
8   push:
9     branches: [ "main" ]
10  workflow_dispatch:
11
12 jobs:
13   build:
14     name: Build and Analyze
15     runs-on: ubuntu-latest
16
17     steps:
18     - uses: actions/checkout@v3
19
20     - name: Set up PHP
21       uses: shivammathur/setup-php@v2
22       with:
23         php-version: '8.2' # Ajusta la versión de PHP según tus necesidades
24         extensions: mbstring, intl, pdo_mysql
25
26     - name: Install Composer dependencies
27       run: composer install
28
29     - name: Create reports directory
30       run: mkdir -p features/bootstrap/reports
31
32     - name: Give execute permission to Behat
33       run: chmod +x vendor/bin/behat
34
35     - name: Run Behat tests
36       run: vendor/bin/behat --format-html --out=features/bootstrap/reports/index.html
37
38     - name: SonarCloud Scan
39       uses: SonarSource/sonarcloud-github-action@master
40       env:
41         GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN } # Needed to get PR information, if any
42         SONAR_TOKEN: ${ secrets.SONAR_TOKEN }
43
44     - name: Upload Behat HTML report
45       if: always()
46       uses: actions/upload-artifact@v2
47       with:
48         name: behat-report
49         path: features/bootstrap/reports/index.html
50
51     - name: Deploy Behat HTML report to GitHub Pages
52       if: always()
53       uses: peaceiris/actions-gh-pages@v3
54       with:
55         personal_token: ${ secrets.PAT_TOKEN }
56         publish_branch: bddreporte
57         publish_dir: features/bootstrap/reports
58

```

Creamos un repositorio con el nombre bddreporte y la rama bddreporte →



Nos dirigimos a settings → Pages → cambiamos la rama a bddreporte y se generara el web page site →

The screenshot shows the GitHub Pages settings for the repository 'edwardapaza / bddreporte'. The left sidebar contains navigation links: General, Access, Code and automation, Pages (selected), Security, Integrations, and Email notifications. The main content area is titled 'GitHub Pages' and includes the following sections:

- General:** States 'Your site is live at <https://edwardapaza.github.io/bddreporte/>'. A red box highlights this information.
- Build and deployment:** Shows the source as 'Deploy from a branch'. The branch is set to 'bddreporte' and the directory is '/ (root)'. A red box highlights these settings, with a red arrow pointing to the 'Save' button.
- Custom domain:** Includes a text input field and 'Save' and 'Remove' buttons.
- Enforce HTTPS:** A checkbox that is checked, with a note: 'Required for your site because you are using the default domain (edwardapaza.github.io)'.

Luego generamos el PAT_TOKEN → nos dirigimos a esta url → <https://github.com/settings/tokens/new>

The screenshot shows the 'New personal access token (classic)' page. The left sidebar has links for GitHub Apps, OAuth Apps, Personal access tokens (selected), and Tokens (classic). The main content area includes:

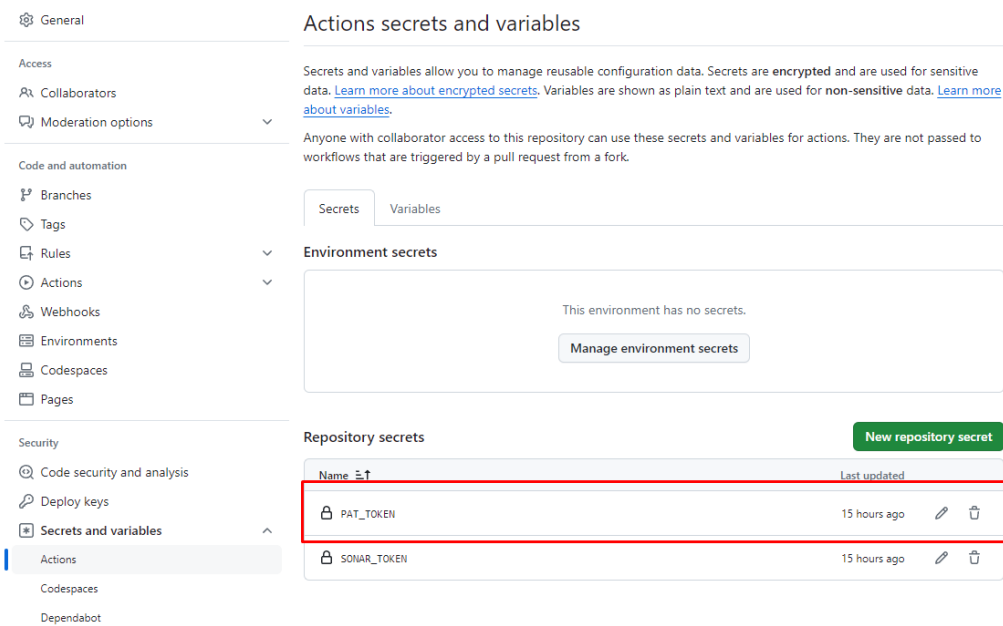
- Note:** A text input field for a note.
- Expiration:** A dropdown menu set to '30 days'.
- Select scopes:** A list of scopes with checkboxes. The 'repo' scope is checked and highlighted with a red arrow. Other scopes include 'repo:status', 'repo_deployment', 'public_repo', 'repo:invite', and 'security_events'.

Le damos en click en →

Generate token

Cancel

Luego nos dirigimos a settings → settings and variables → actions → creamos New repository secret → PAT_TOKEN y agregamos la key que nos genero anteriormente



The screenshot shows the GitHub Actions 'Secrets and variables' page for a repository. The left sidebar contains navigation links: General, Access, Collaborators, Moderation options, Code and automation (Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, Pages), Security (Code security and analysis, Deploy keys, Secrets and variables, Actions, Codespaces, Dependabot), and Actions. The main content area is titled 'Actions secrets and variables' and includes a description of secrets and variables. Below this, there are tabs for 'Secrets' and 'Variables'. The 'Environment secrets' section shows a message 'This environment has no secrets.' with a 'Manage environment secrets' button. The 'Repository secrets' section has a 'New repository secret' button and a table of secrets. The table has columns for 'Name', 'Last updated', and actions (edit, delete). Two secrets are listed: 'PAT_TOKEN' and 'SONAR_TOKEN', both updated 15 hours ago. The 'PAT_TOKEN' row is highlighted with a red box.

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Codespaces

Pages

Security

Code security and analysis

Deploy keys

Secrets and variables

Actions

Codespaces

Dependabot

Actions secrets and variables

Secrets and variables allow you to manage reusable configuration data. Secrets are **encrypted** and are used for sensitive data. [Learn more about encrypted secrets](#). Variables are shown as plain text and are used for **non-sensitive** data. [Learn more about variables](#).

Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.

Secrets Variables

Environment secrets

This environment has no secrets.

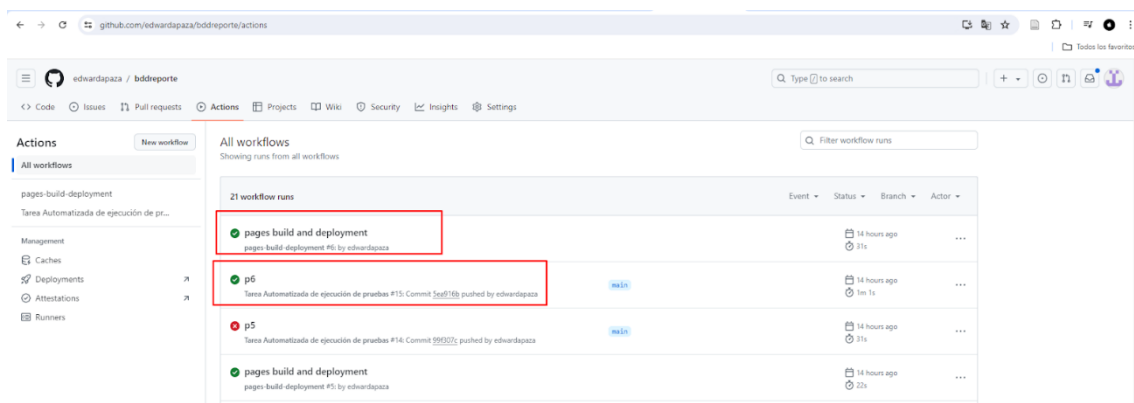
Manage environment secrets

Repository secrets

New repository secret

Name	Last updated
PAT_TOKEN	15 hours ago
SONAR_TOKEN	15 hours ago

Luego haremos el su respectivo git add . → git commit → git push →



The screenshot shows the GitHub Actions 'All workflows' page for the repository 'edwardapaza / bddreporte'. The left sidebar contains navigation links: Actions, All workflows, pages-build-deployment, Tarea Automatizada de ejecución de pr..., Management (Caches, Deployments, Attestations, Runners), and Runners. The main content area is titled 'All workflows' and shows a list of workflow runs. The first run is 'pages build and deployment' (pages build deployment #6 by edwardapaza) with a green status icon. The second run is 'p6' (Tarea Automatizada de ejecución de pruebas #15: Commit 5e7f1b3 pushed by edwardapaza) with a green status icon. The third run is 'p5' (Tarea Automatizada de ejecución de pruebas #14: Commit 999027c pushed by edwardapaza) with a red status icon. The fourth run is 'pages build and deployment' (pages build deployment #5 by edwardapaza) with a green status icon. The 'pages build and deployment' run is highlighted with a red box.

edwardapaza / bddreporte

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Actions

New workflow

All workflows

pages-build-deployment

Tarea Automatizada de ejecución de pr...

Management

Caches

Deployments

Attestations

Runners

All workflows

Showing runs from all workflows

Filter workflow runs

21 workflow runs

Event	Status	Branch	Actor
pages build and deployment	Success	pages build deployment #6	edwardapaza
p6	Success	Tarea Automatizada de ejecución de pruebas #15: Commit 5e7f1b3 pushed by edwardapaza	edwardapaza
p5	Failure	Tarea Automatizada de ejecución de pruebas #14: Commit 999027c pushed by edwardapaza	edwardapaza
pages build and deployment	Success	pages build deployment #5	edwardapaza

Nos mostrara el diagrama → pages-build-deployment →

github.com/edwardapaza/bddreporte/actions/runs/9885878116

edwardapaza / bddreporte

CodeIssuesPull requestsActionsProjectsWikiSecurityInsightsSettings

pages build and deployment #6

Summary

Jobs

- build
- report-build-status
- deploy

Run details

- Usage

Triggered via dynamic 14 hours ago

edwardapaza 4e20a14

Status

Success

Total duration

31s

Artifacts

1

pages-build-deployment

on: dynamic

build7s

report-build-status3s

deploy9s

https://edwardapaza.github.io/bddreporte/

Artifacts

Produced during runtime

Name	Size
github-pages	11.2 KB

Hacemos click →

Triggered via dynamic 14 hours ago

edwardapaza 4e20a14

Status

Success

Total duration

31s

Artifacts

1

pages-build-deployment

on: dynamic

build7s

report-build-status3s

deploy9s

https://edwardapaza.github.io/bddreporte/

Artifacts

Produced during runtime

Name	Size
github-pages	11.2 KB

Nos mostrará el reporte bdd →

