

Logotipo

El contenido generado por IA puede ser incorrecto.



**Documentación de Pruebas del Módulo de Productos con PHPUnit**

**1. Introducción**

Este documento describe el proceso de integración de **PHPUnit** en el proyecto PATY SPORT desarrollado en PHP puro, así como las pruebas realizadas para el módulo de gestión de productos, usuarios, ticket de salida y ticket de ingreso. Se detalla cómo se configuró PHPUnit, cómo se ejecutaron las pruebas y una explicación de cada prueba realizada para los métodos del módulo productos.

**2. ¿Qué es PHPUnit?**

PHPUnit es un framework para realizar pruebas unitarias en PHP. Permite automatizar la ejecución de pruebas, asegurando que las funciones y métodos del código funcionen correctamente y se mantengan estables frente a cambios futuros.

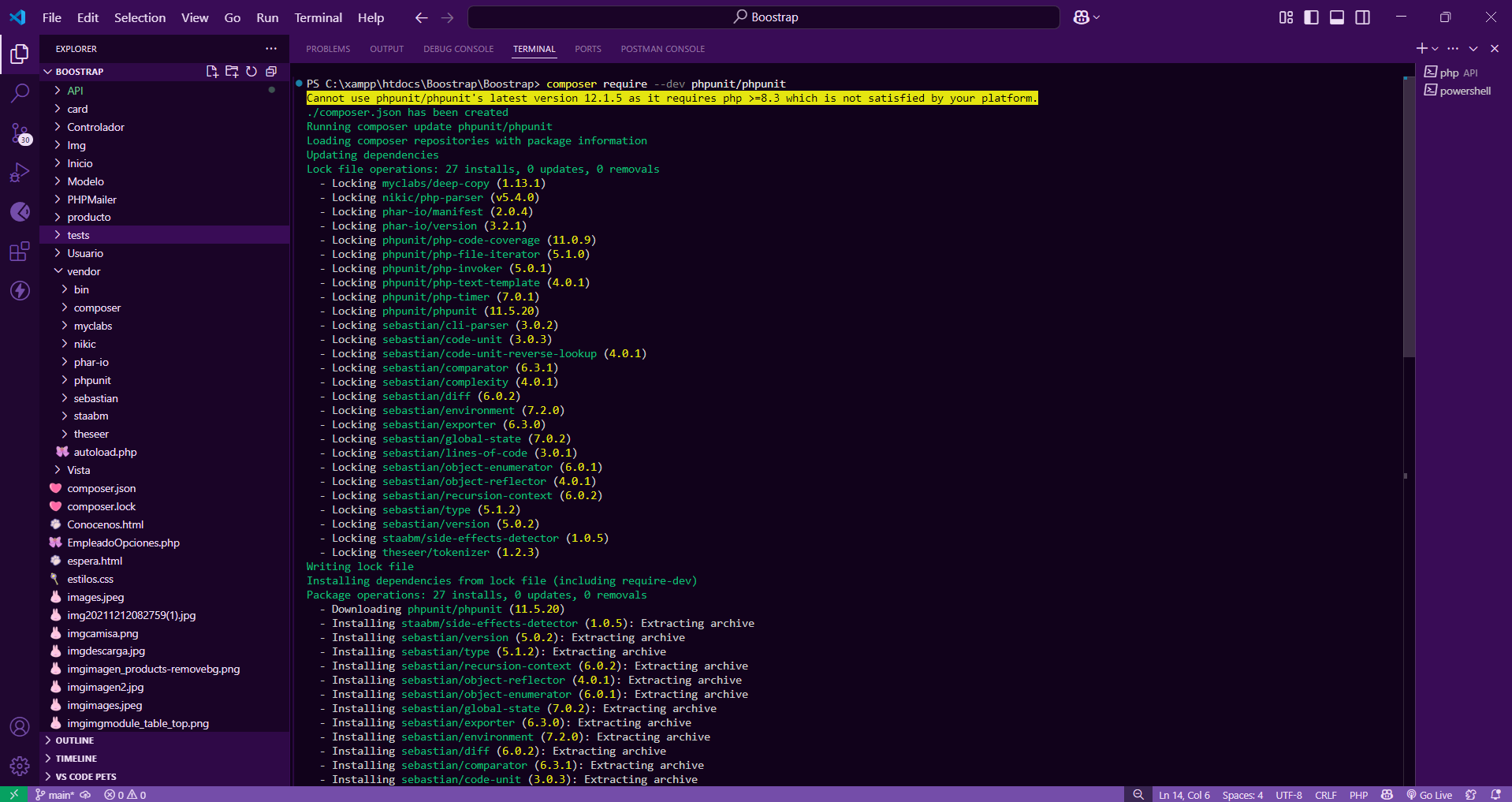
**3. Integración de PHPUnit en un Proyecto PHP Puro**

**3.1. Instalación de PHPUnit**

Para integrar PHPUnit en un proyecto PHP sin frameworks, se siguieron los siguientes pasos:

1. **Agregar PHPUnit como dependencia de desarrollo**  
   En la raíz del proyecto, ejecutar:

composer require --dev phpunit/phpunit



1. **Se crea automáticamente la carpeta VENDOR**   
   Se crea en el momento de que termine la ejecución del comando dado, la carpeta vendor que es donde se realizan las pruebas.

Texto

El contenido generado por IA puede ser incorrecto.

1. **Crear la carpeta de pruebas**  
   Se creó una carpeta llamada tests/ en la raíz del proyecto principal para almacenar los archivos de prueba.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

**3.2. Estructura del Proyecto con PHPUnit**Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

**4. Explicación de las Pruebas del Módulo Productos (Método por Método)**

**4.1. testAgregarProducto()**

* **Qué prueba:**  
  Verifica que el método agregarProducto() pueda insertar un producto sin lanzar excepciones.
* **Cómo:**  
  Se llama a AgregarProducto() con datos de prueba válidos.  
  No se espera un valor de retorno, por eso la prueba es que no lance excepción.

Texto

El contenido generado por IA puede ser incorrecto.Una captura de pantalla de una computadora

El contenido generado por IA puede ser incorrecto.

* **Validación:**  
  Si no se lanza ninguna excepción, la prueba pasa.

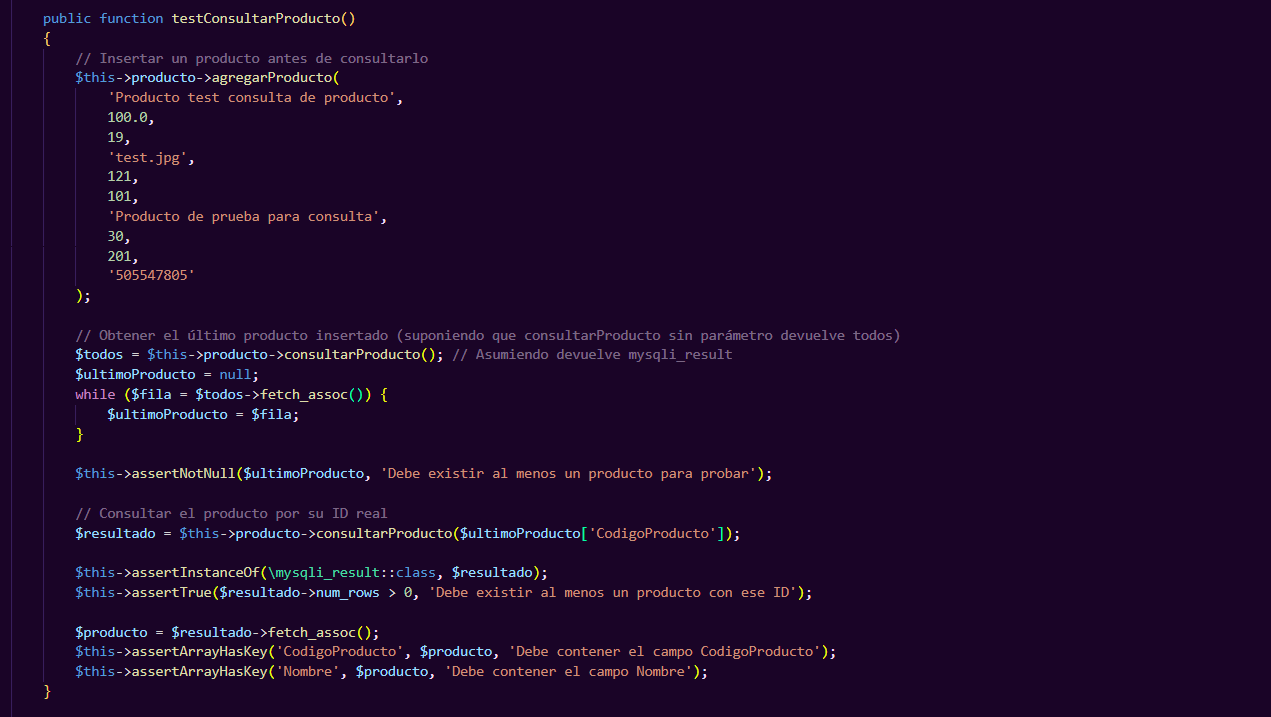


Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

**4.2. testConsultarProducto()**

* **Qué prueba:**  
  Comprueba que el método consultarProducto() retorne los datos correctos de un producto específico.
* **Cómo:**  
  Primero inserta un producto con agregarProducto().  
  Luego obtiene todos los productos y selecciona el último insertado.  
  Finalmente, consulta el producto por su ID para validar que los datos coinciden.



Una captura de pantalla de una computadora

El contenido generado por IA puede ser incorrecto.



* **Validación:**  
  El resultado debe ser un objeto mysqli\_result con al menos una fila que contenga las claves esperadas (por ejemplo, CodigoProducto y Nombre).

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

**4.3. testActualizarProducto()**

* **Qué prueba:**  
  Verifica que el método actualizarProducto() modifique correctamente los datos de un producto existente.
* **Cómo:**  
  Se llama a actualizarProducto() con un ID de producto existente y nuevos datos.

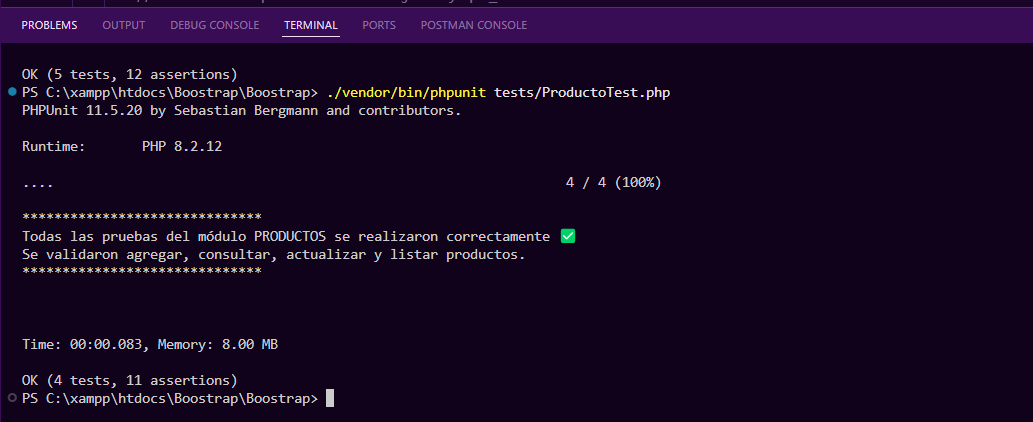
Texto

El contenido generado por IA puede ser incorrecto.Una captura de pantalla de una computadora

El contenido generado por IA puede ser incorrecto.



* **Validación:**  
  El método debe retornar true, indicando que la actualización fue exitosa.



**5. Consideraciones Adicionales**

* **Dependencias de Datos**:  
  Para que las pruebas funcionen correctamente, los IDs usados en las pruebas (id\_categoria, id\_estado, id\_marca, id\_talla, y documento\_usuario) deben existir en sus respectivas tablas de la base de datos.
* **Configuración de Base de Datos**:  
  Se recomienda tener una base de datos de pruebas para no afectar los datos reales del proyecto.
* **Método tearDownAfterClass()**:  
  Al finalizar todas las pruebas, imprime un mensaje en consola indicando que todas las pruebas del módulo productos se realizaron con éxito.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.



**6. Ejecución y Resultado de las Pruebas**

Al ejecutar vendor/bin/phpunit en la raíz del proyecto, se corren automáticamente todas las pruebas definidas en ProductoTest.php. Si todas las pruebas pasan, el resultado será algo similar a:

**PS C:\xampp\htdocs\Boostrap\Boostrap> ./vendor/bin/phpunit tests/ProductoTest.php**

**PHPUnit 11.5.20 by Sebastian Bergmann and contributors.**

**Runtime: PHP 8.2.12**

**PHPUnit 11.5.20 by Sebastian Bergmann and contributors.**

**Runtime: PHP 8.2.12**

**.... 4 / 4 (100%)**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Todas las pruebas del módulo PRODUCTOS se realizaron correctamente ✅**

**Se validaron agregar, consultar, actualizar y listar productos.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Time: 00:00.083, Memory: 8.00 MB**

**OK (4 tests, 11 assertions)**

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

**7. Ejecución de las Pruebas**

Para ejecutar las pruebas, en la terminal se debe correr el comando:

**vendor/bin/phpunit tests/ProductoTest.php**

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Este comando ejecuta todas las pruebas dentro del directorio /tests y muestra un reporte con resultados.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

**Documentación de Pruebas del Módulo de Usuarios con PHPUnit**

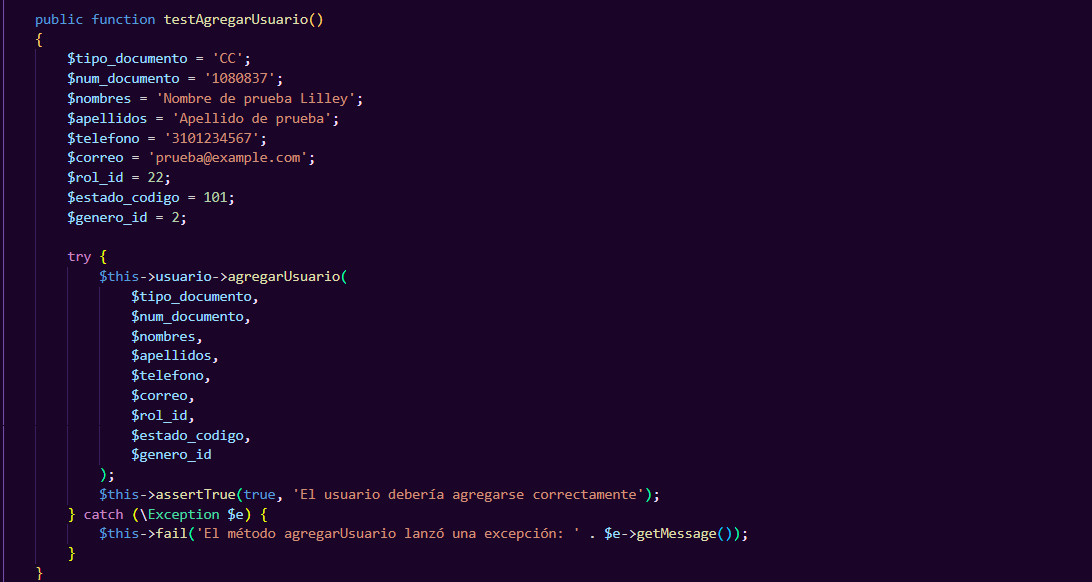
**1. Introducción**

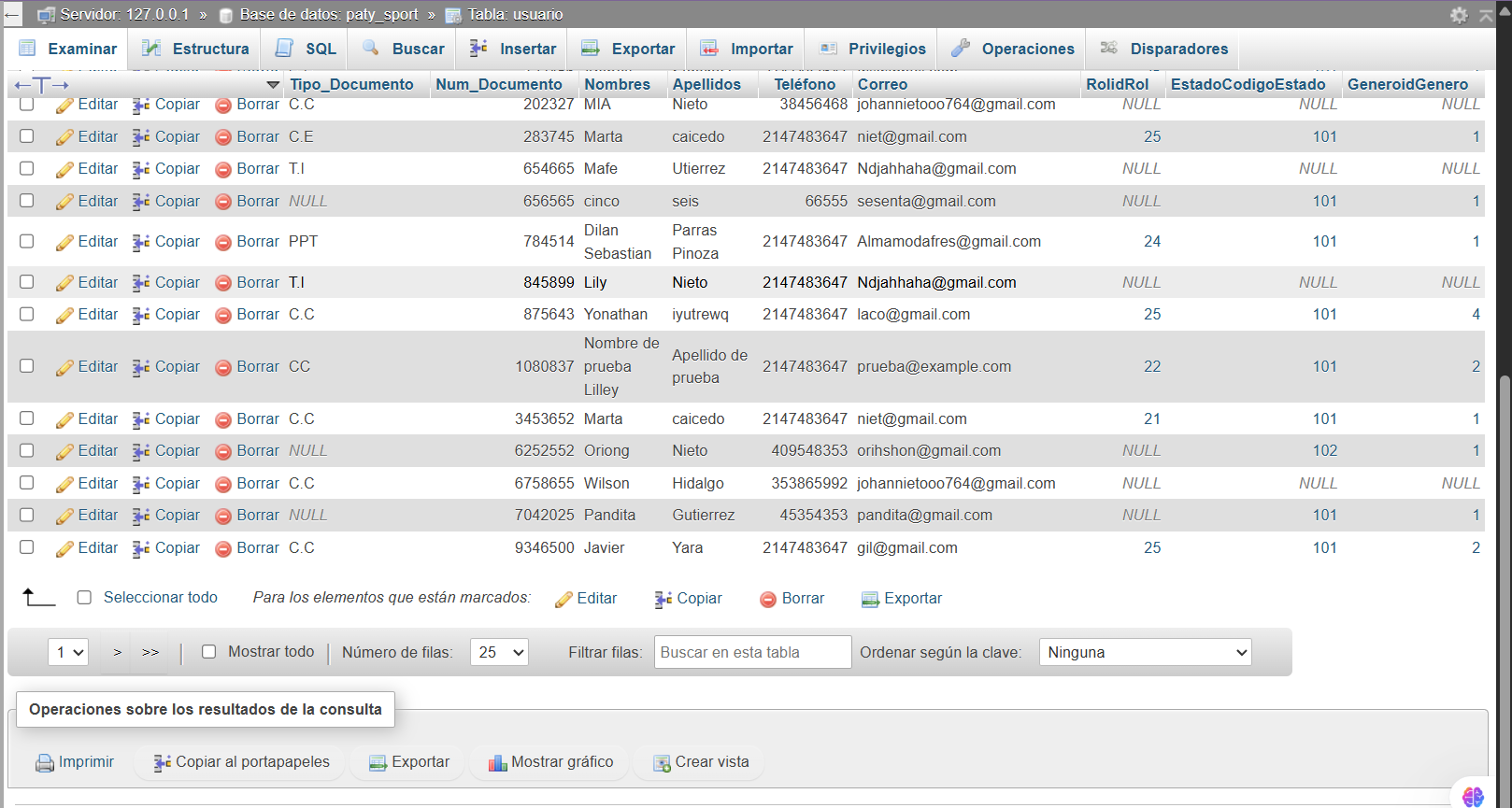
Este documento describe el proceso de integración de **PHPUnit** en el proyecto PATY SPORT desarrollado en PHP puro, así como las pruebas realizadas para el módulo de gestión de usuarios. Se detalla cómo se configuró PHPUnit, cómo se ejecutaron las pruebas y una explicación de cada prueba realizada para los métodos del módulo productos.

**2. Explicación de las Pruebas del Módulo Productos (Método por Método)**

**3.1. testAgregarUsuario()**

* **Qué prueba:**  
  Verifica que el método agregarUsuario() pueda insertar un usuario sin lanzar excepciones.
* **Cómo:**  
  Se llama a AgregarUsuario() con datos de prueba válidos.  
  No se espera un valor de retorno, por eso la prueba es que no lance excepción.





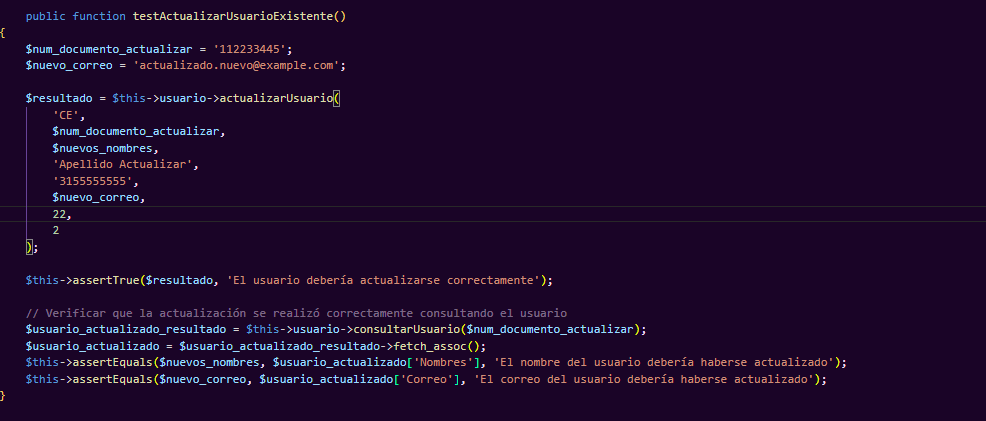


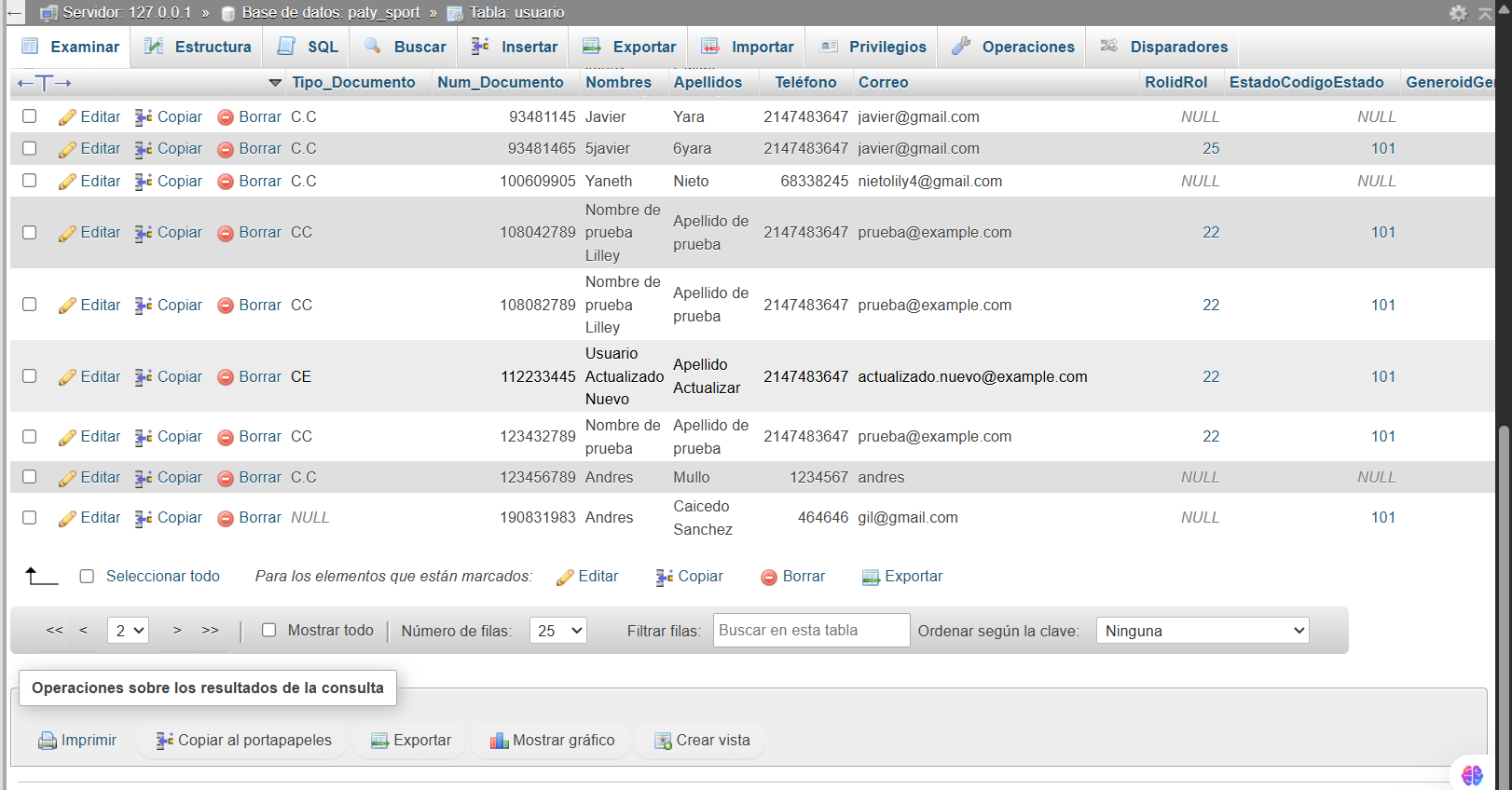
* **Validación:**  
  Si no se lanza ninguna excepción, la prueba pasa.

Texto

El contenido generado por IA puede ser incorrecto.

**3.2. testActualizarUsuario()**

* **Qué prueba:**  
  Verifica que el método actualizarUsuario() modifique correctamente los datos de un usuario existente.
* **Cómo:**  
  Se llama a actualizarUsuario() con un número de documento del usuario existente y nuevos datos.





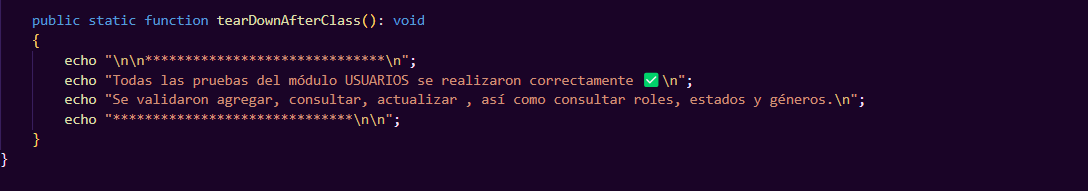
* **Validación:**  
  El método debe retornar true, indicando que la actualización fue exitosa.

Texto

El contenido generado por IA puede ser incorrecto.

**4. Consideraciones Adicionales**

* **Dependencias de Datos**:  
  Para que las pruebas funcionen correctamente, los Números de documentos usados en las pruebas (id\_rol, id\_estado y id\_genero) deben existir en sus respectivas tablas de la base de datos.
* **Configuración de Base de Datos**:  
  Se recomienda tener una base de datos de pruebas para no afectar los datos reales del proyecto.
* **Método tearDownAfterClass()**:  
  Al finalizar todas las pruebas, imprime un mensaje en consola indicando que todas las pruebas del módulo usuario se realizaron con éxito.



Texto

El contenido generado por IA puede ser incorrecto.



**5. Ejecución y Resultado de las Pruebas**

Al ejecutar vendor/bin/phpunit en la raíz del proyecto, se corren automáticamente todas las pruebas definidas en UsuariosTest.php. Si todas las pruebas pasan, el resultado será algo similar a:

**PS C:\xampp\htdocs\Boostrap\Boostrap> ./vendor/bin/phpunit tests/UsuariosTest.php**

**PHPUnit 11.5.20 by Sebastian Bergmann and contributors.**

**PHPUnit 11.5.20 by Sebastian Bergmann and contributors.**

**Runtime: PHP 8.2.12**

**..... 5 / 5 (100%)**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

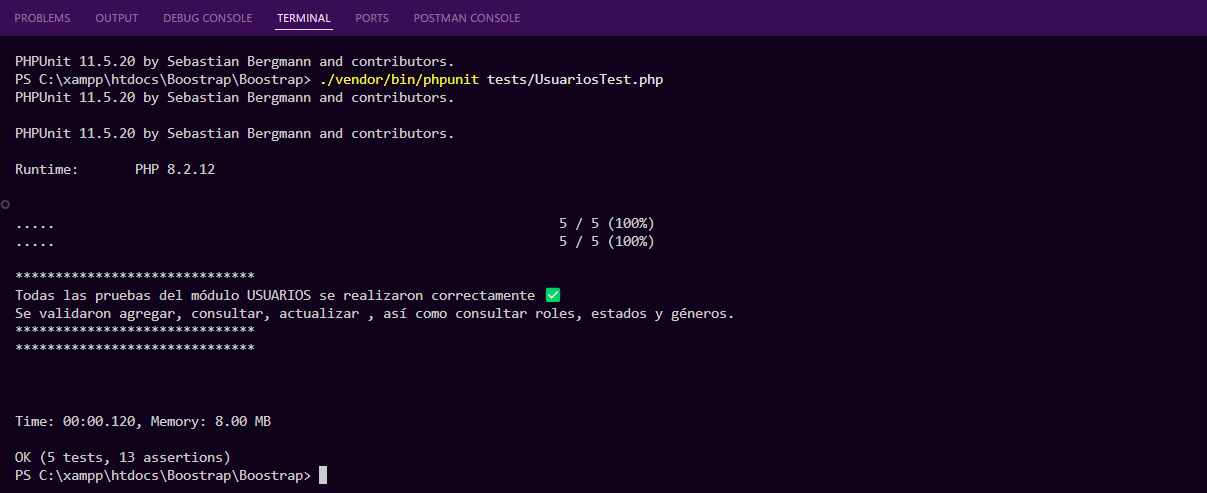
**Todas las pruebas del módulo USUARIOS se realizaron correctamente ✅**

**Se validaron agregar, consultar, actualizar , así como consultar roles, estados y géneros.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Time: 00:00.120, Memory: 8.00 MB**

**OK (5 tests, 13 assertions)**



**7. Ejecución de las Pruebas**

Para ejecutar las pruebas, en la terminal se debe correr el comando:

**vendor/bin/phpunit tests/UsuariosTest.php**

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Este comando ejecuta todas las pruebas dentro del directorio /tests y muestra un reporte con resultados.

Captura de pantalla de computadora

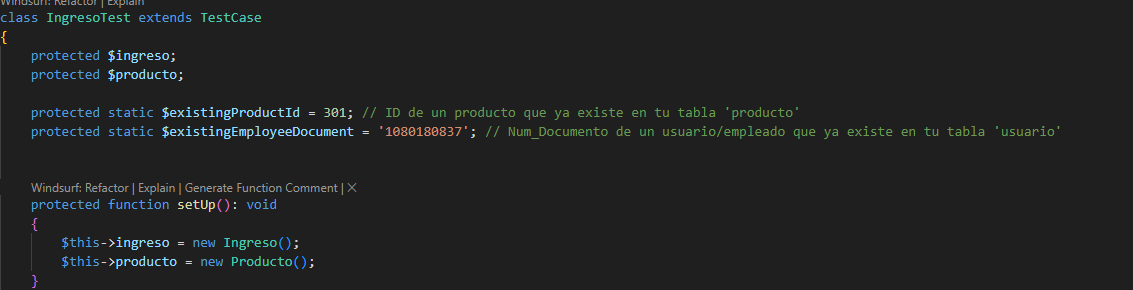
El contenido generado por IA puede ser incorrecto.

**Documentación de Pruebas del Módulo de Ticket de Ingreso con PHPUnit**

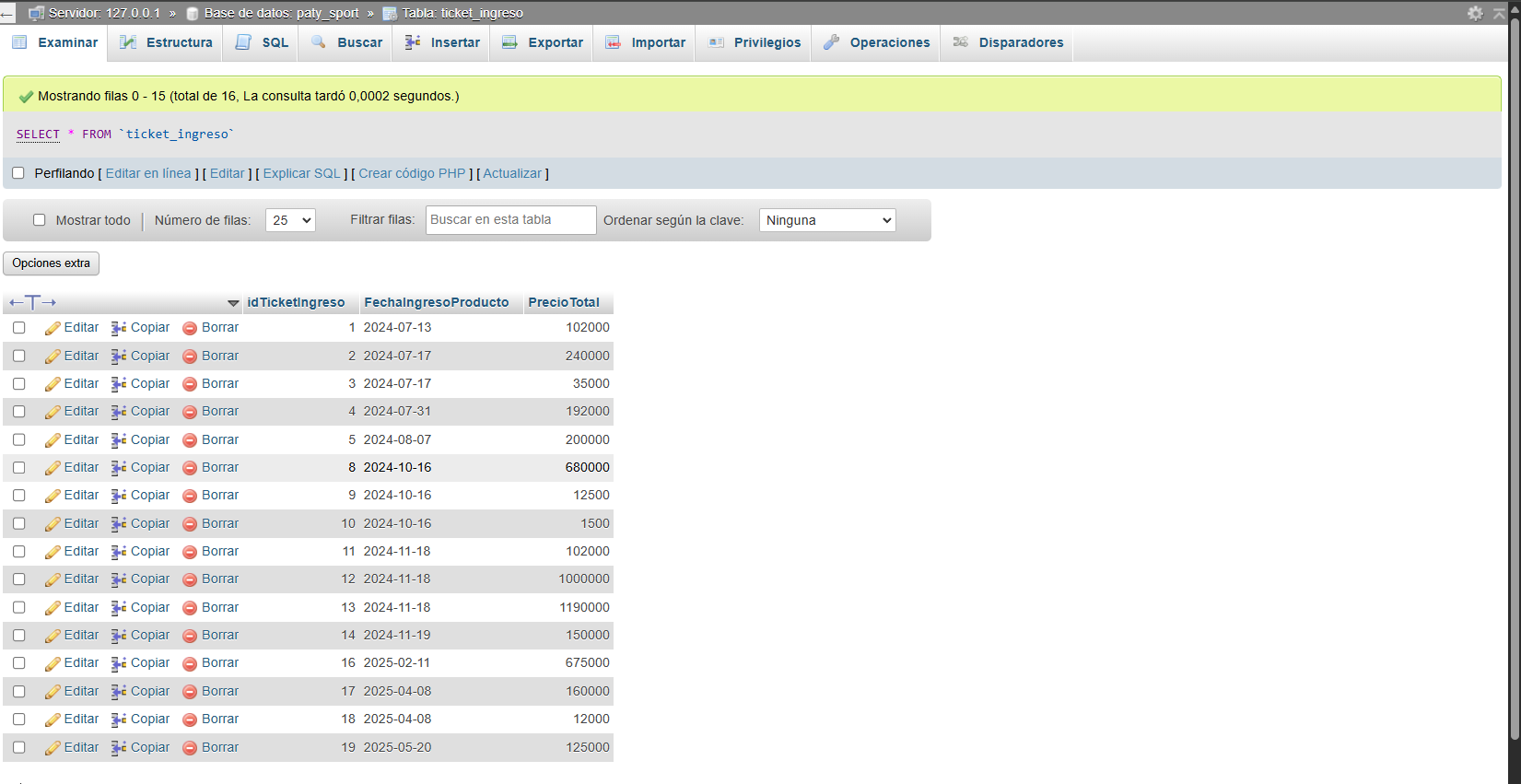
**1. Explicación de las Pruebas del Módulo Ticket de Ingreso (Método por Método)**

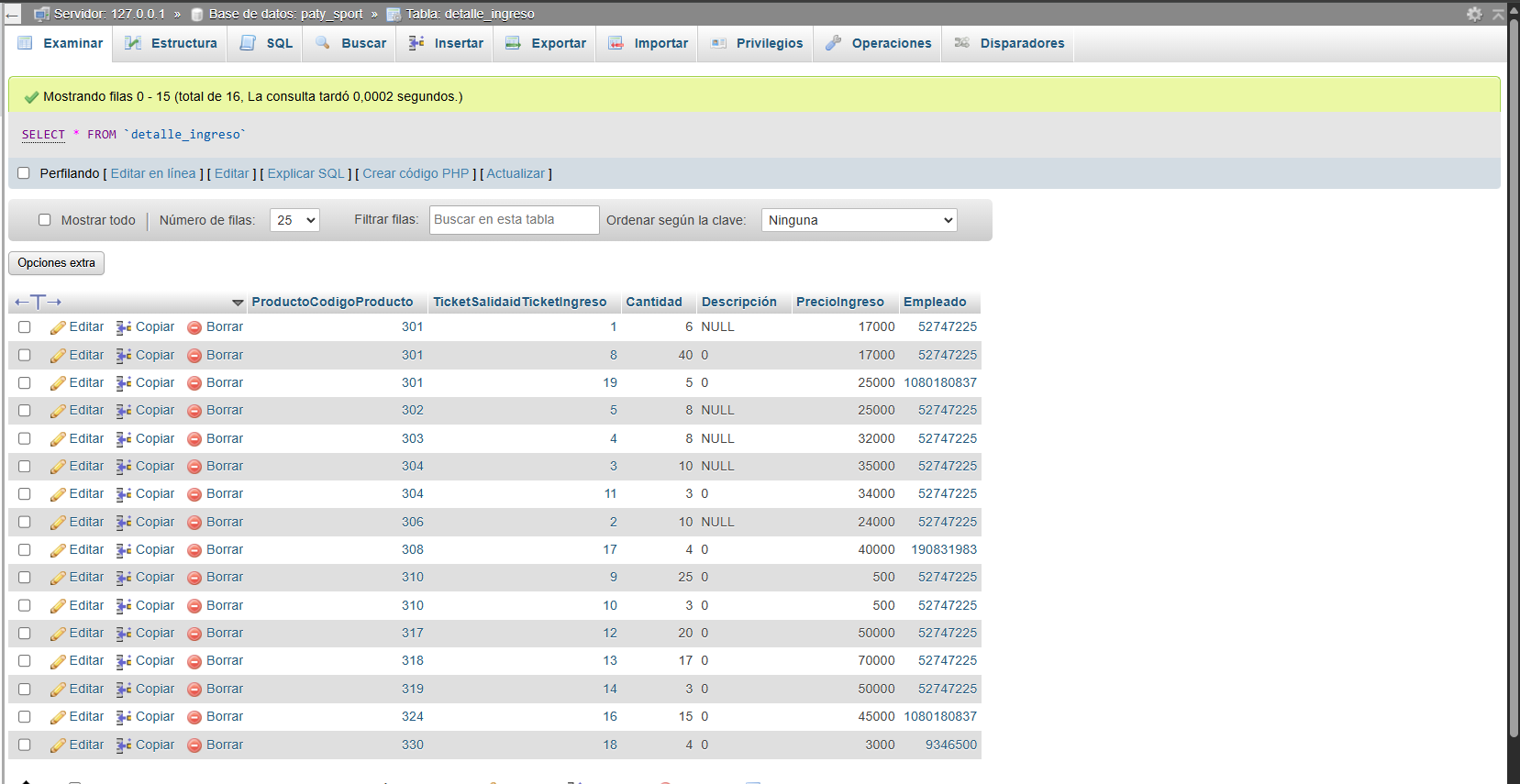
**2.1. testAgregarIngreso()**

* **Qué prueba:**  
  Verifica que el método AgregarIngreso() pueda insertar un ingreso de varios productos sin lanzar excepciones.
* **Cómo:**  
  Se llama a AgregarIngreso() con datos de prueba válidos.  
  No se espera un valor de retorno, por eso la prueba es que no lance excepción.

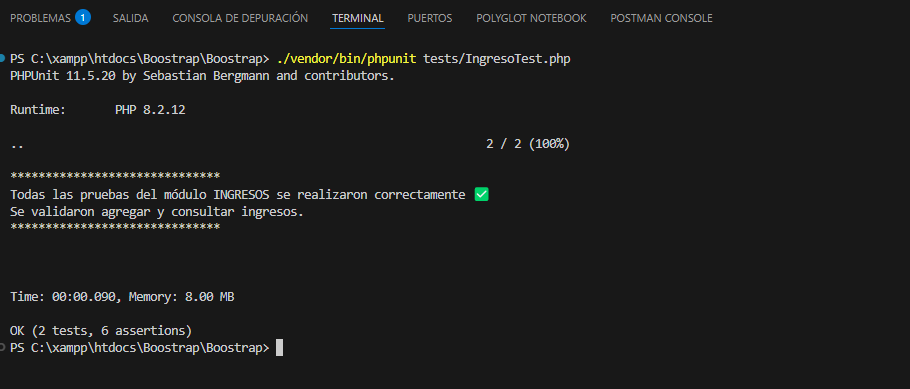






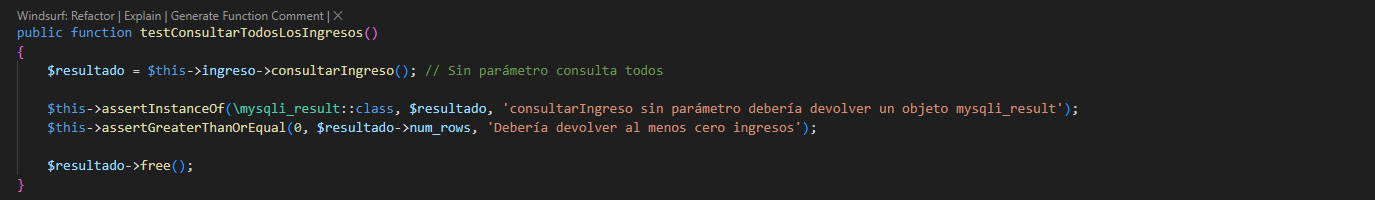


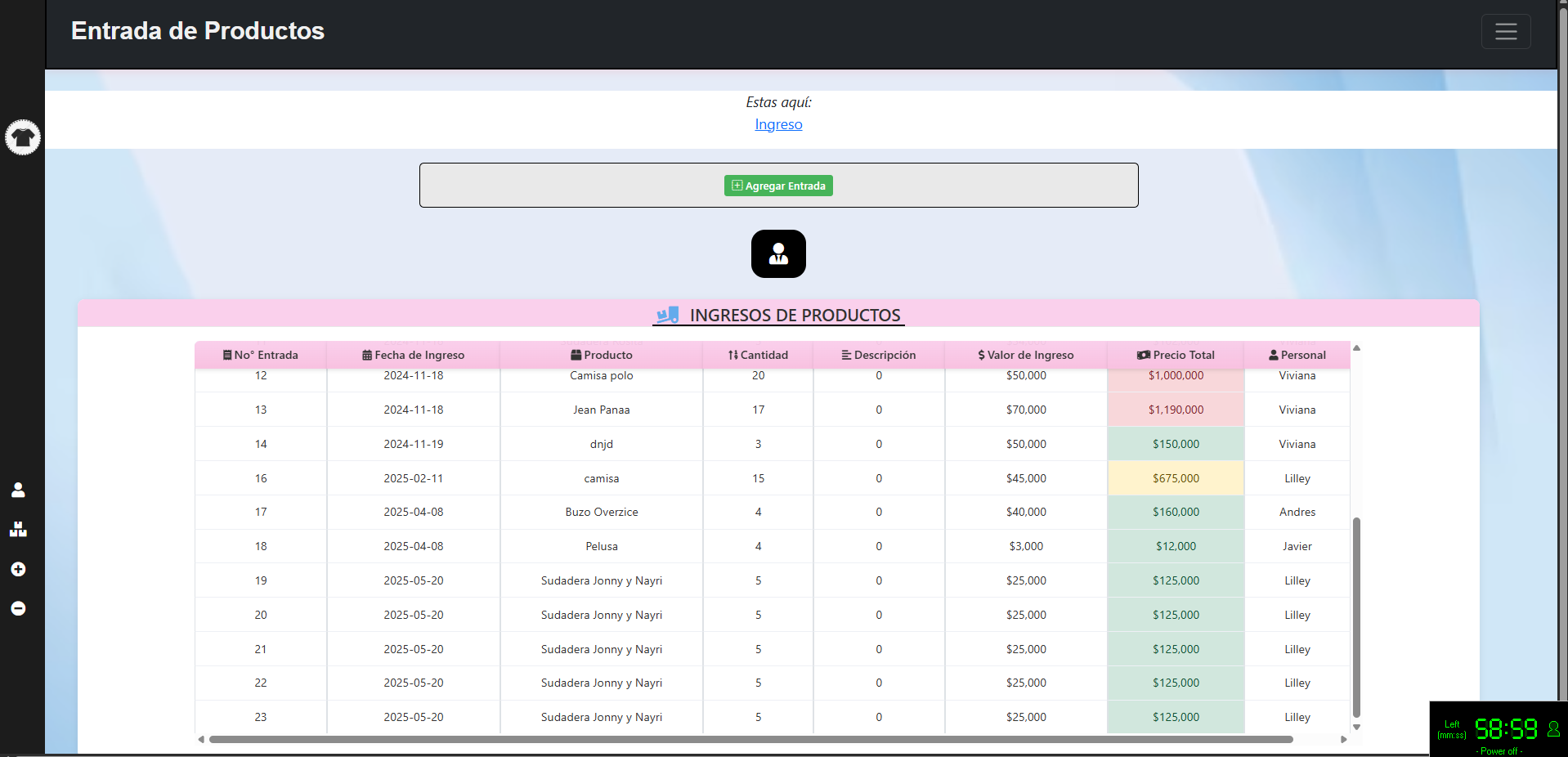
* **Validación:**  
  Si no se lanza ninguna excepción, la prueba pasa.



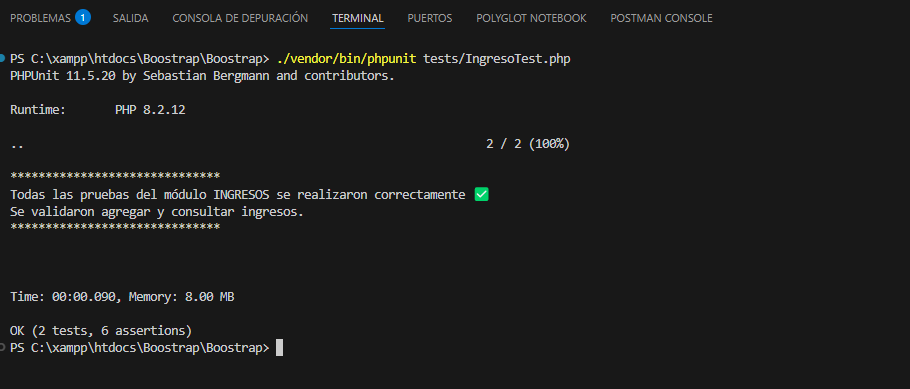
**2.2. testConsultarTodosLosIngresos()**

* **Qué prueba:**  
  Verifica que el método ConsultarTodosLosIngresos() me consulte correctamente los datos de todos los ticket de ingreso existentes.
* **Cómo:**  
  Se llama a ConsultarTodosLosIngresos() porque obtiene todos los ingresos que hacen de los productos y los muestra.





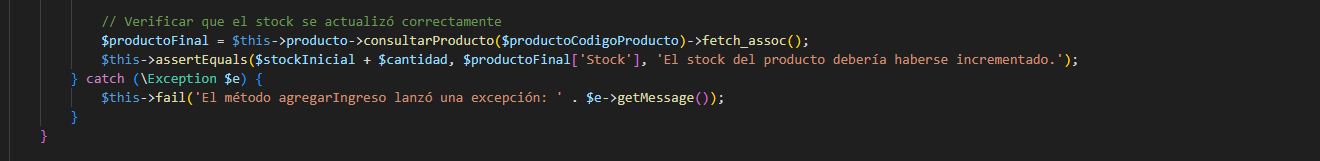
* **Validación:**  
  El método debe retornar true, indicando que la actualización fue exitosa.



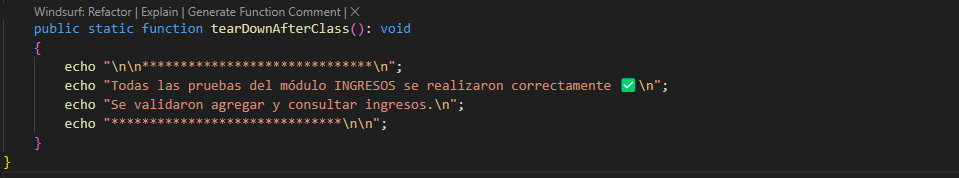
**3. Consideraciones Adicionales**

* **Dependencias de Datos**:  
  Para que las pruebas funcionen correctamente, los productos a darle ingreso deben ser existentes en sus respectivas tablas de la base de datos.
* **Configuración de Base de Datos**:  
  Se recomienda tener una base de datos de pruebas para no afectar los datos reales del proyecto.
* **Actualización de Stock en la Tabla Productos:**

Se hace la debida inserción del stock en agregar ingreso para que ella lleve esa cantidad a la tabla de productos.



* **Método tearDownAfterClass()**:  
  Al finalizar todas las pruebas, imprime un mensaje en consola indicando que todas las pruebas del módulo de ticket de ingreso se realizaron con éxito.



**4. Ejecución y Resultado de las Pruebas**

Al ejecutar vendor/bin/phpunit en la raíz del proyecto, se corren automáticamente todas las pruebas definidas en IngresoTest.php. Si todas las pruebas pasan, el resultado será algo similar a:

**PS C:\xampp\htdocs\Boostrap\Boostrap> ./vendor/bin/phpunit tests/IngresoTest.php**

**PHPUnit 11.5.20 by Sebastian Bergmann and contributors.**

**Runtime: PHP 8.2.12**

**.. 2 / 2 (100%)**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

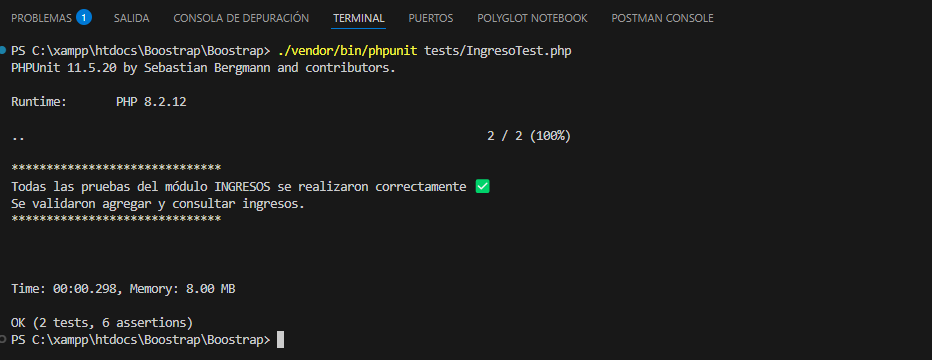
**Todas las pruebas del módulo INGRESOS se realizaron correctamente ✅**

**Se validaron agregar y consultar ingresos.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Time: 00:00.298, Memory: 8.00 MB**

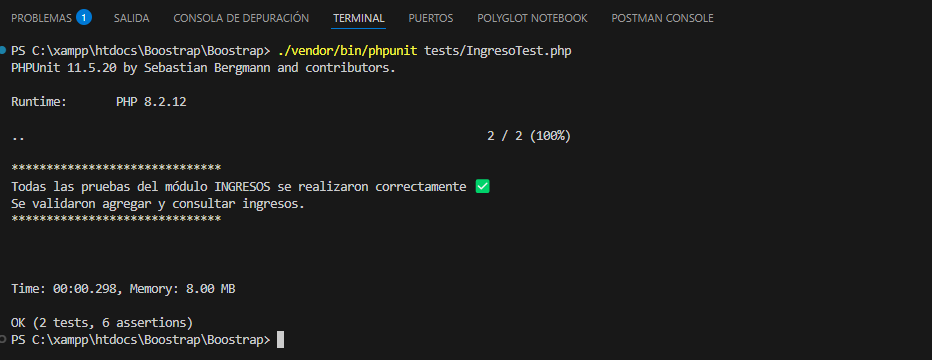
**OK (2 tests, 6 assertions)**



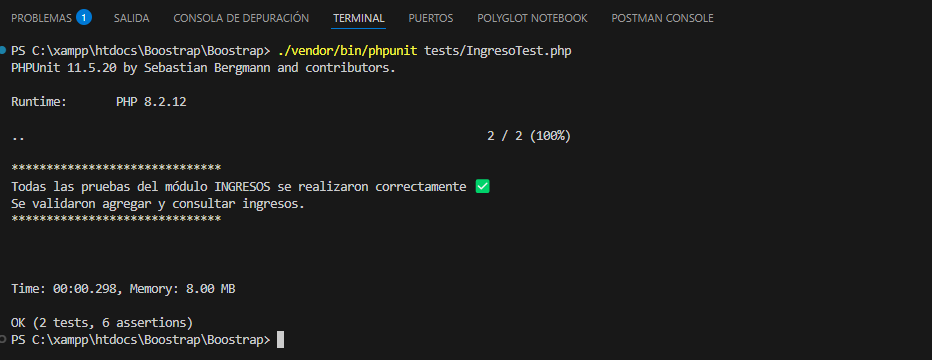
**5. Ejecución de las Pruebas**

Para ejecutar las pruebas, en la terminal se debe correr el comando:

**vendor/bin/phpunit tests/IngresoTest.php**



Este comando ejecuta todas las pruebas dentro del directorio /tests y muestra un reporte con resultados.

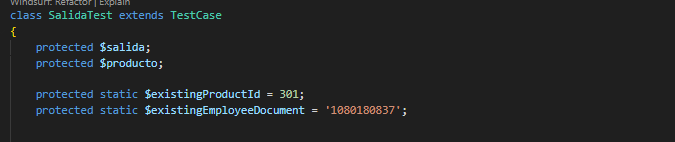


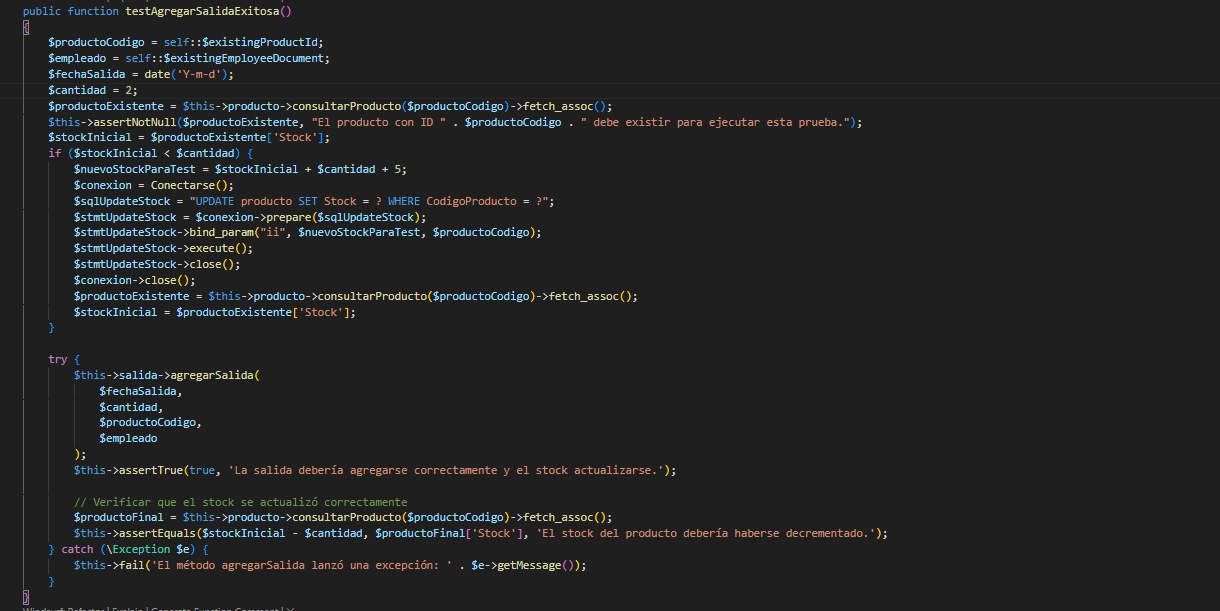
**Documentación de Pruebas del Módulo de Ticket de Salida con PHPUnit**

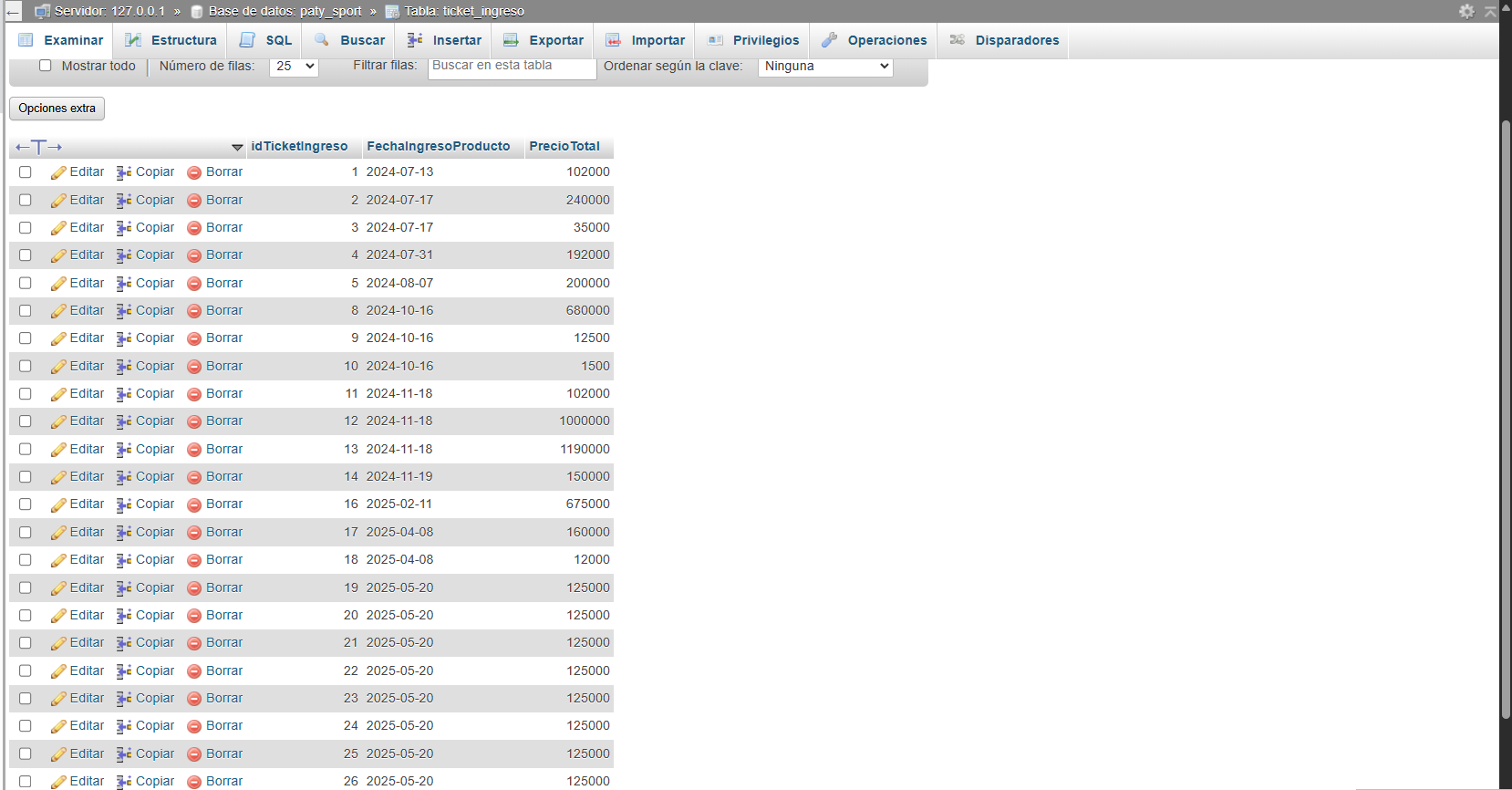
**1. Explicación de las Pruebas del Módulo Ticket de Salida (Método por Método)**

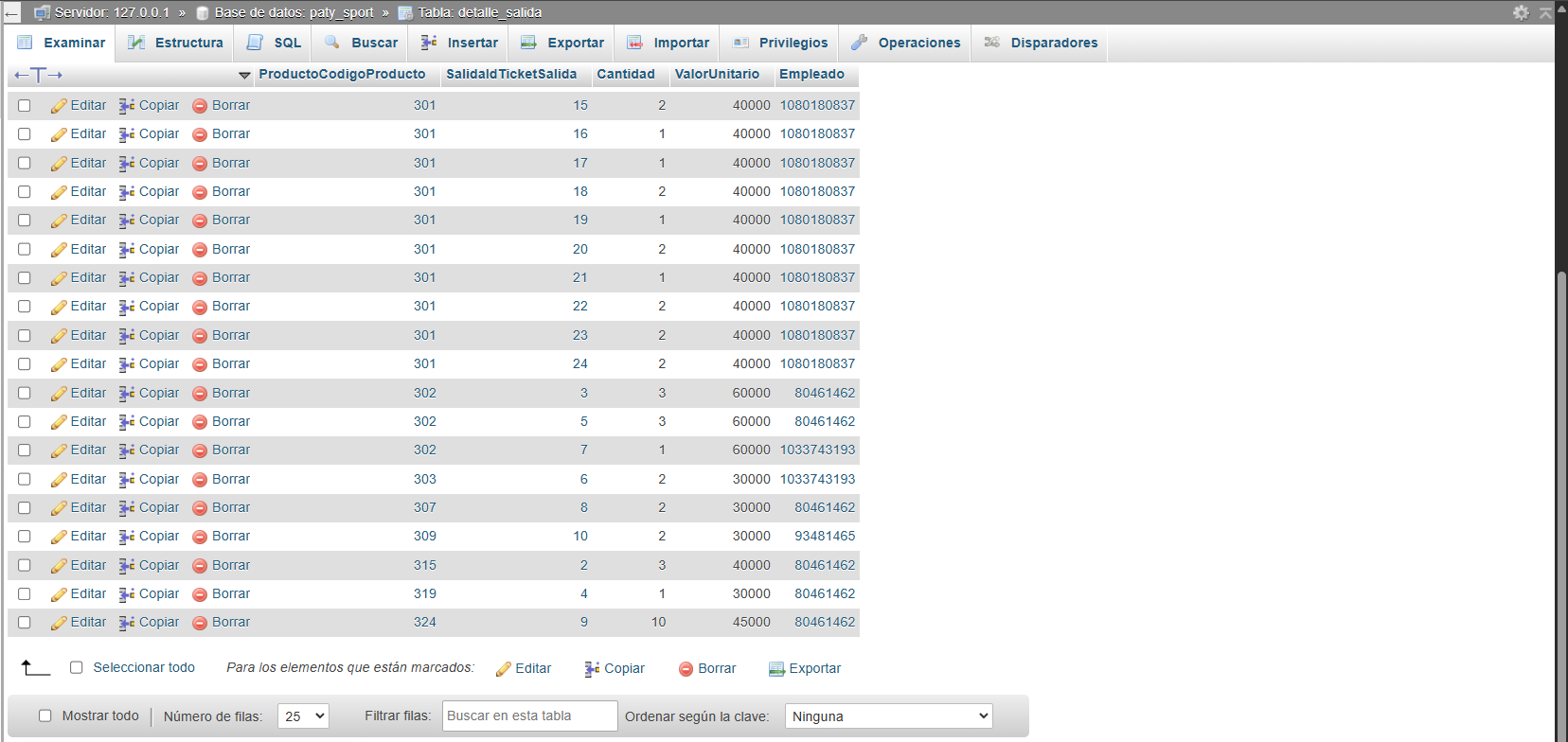
**2.1. testAgregarSalidaExitosa()**

* **Qué prueba:**  
  Verifica que el método AgregarSalidaExitoda() puede insertar una salida de varios productos sin lanzar excepciones.
* **Cómo:**  
  Se llama a AgregarSalidaExitosa() con datos de prueba válidos.  
  No se espera un valor de retorno, por eso la prueba es que no lance excepción.

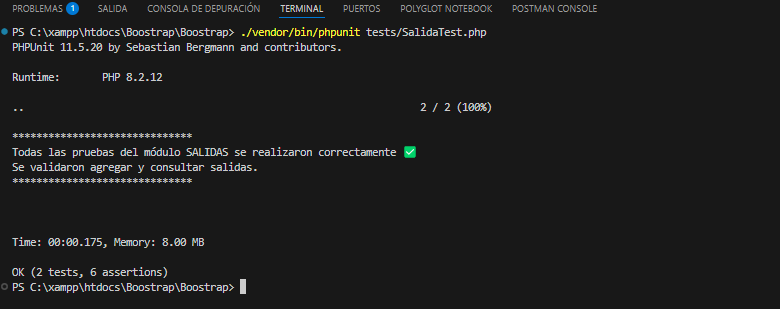






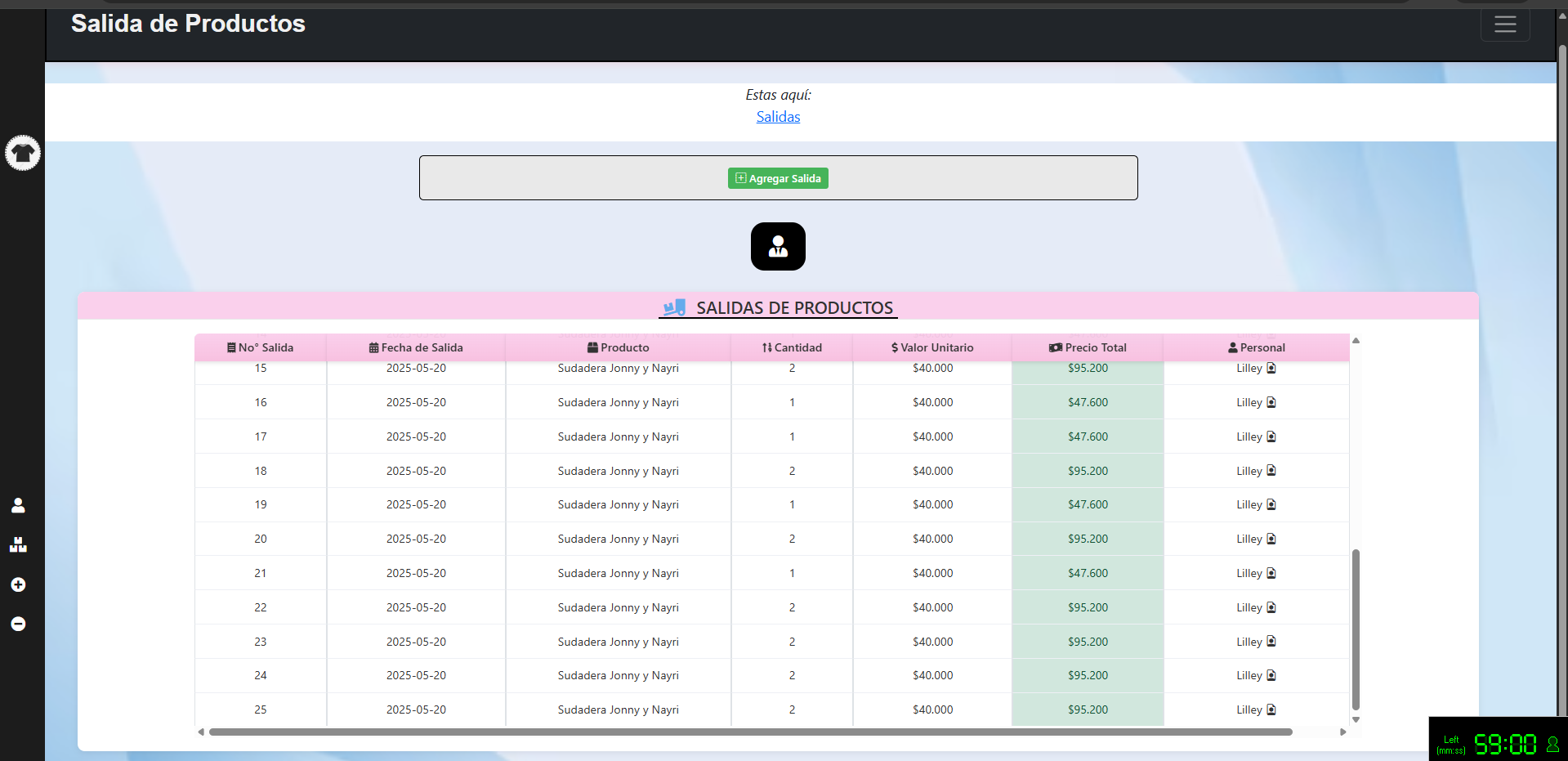


* **Validación:**  
  Si no se lanza ninguna excepción, la prueba pasa.

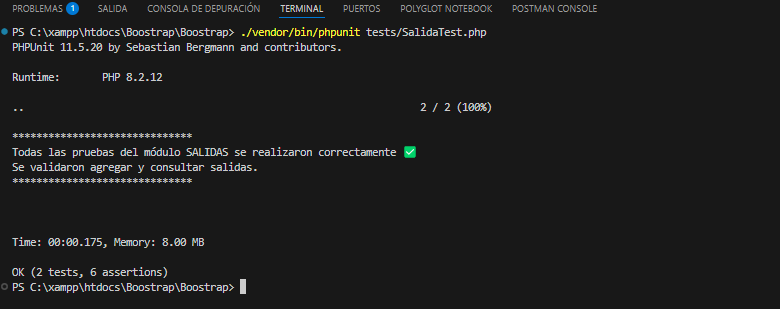


**2.2. test Consultar Todas Las Salidas()**

* **Qué prueba:**  
  Verifica que el método testConsultarTodasLasSalidas() me consulte correctamente los datos de todos los ticket de salida existentes.
* **Cómo:**  
  Se llama a ConsultarTodosLasSalidas() porque obtiene todos las salidas que hacen de los productos y los muestra.



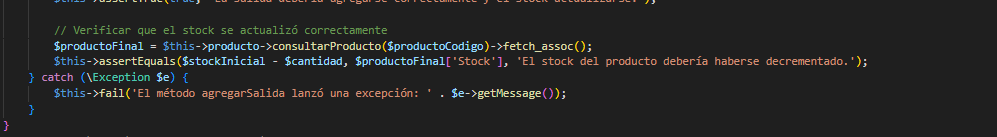
* **Validación:**  
  El método debe retornar true, indicando que la actualización fue exitosa.



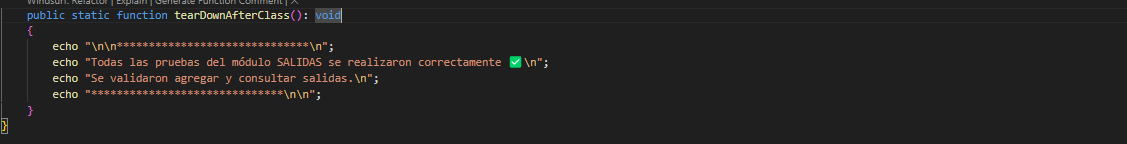
**3. Consideraciones Adicionales**

* **Dependencias de Datos**:  
  Para que las pruebas funcionen correctamente, los productos a darle salida deben ser existentes en sus respectivas tablas de la base de datos.
* **Configuración de Base de Datos**:  
  Se recomienda tener una base de datos de pruebas para no afectar los datos reales del proyecto.
* **Actualización de Stock en la Tabla Productos:**

Se hace la debida inserción del stock en agregar una salida para que ella lleve esa cantidad a la tabla de productos.



* **Método tearDownAfterClass()**:  
  Al finalizar todas las pruebas, imprime un mensaje en consola indicando que todas las pruebas del módulo de ticket de salida se realizaron con éxito.



**4. Ejecución y Resultado de las Pruebas**

Al ejecutar vendor/bin/phpunit en la raíz del proyecto, se corren automáticamente todas las pruebas definidas en SalidaTest.php. Si todas las pruebas pasan, el resultado será algo similar a:

**PS C:\xampp\htdocs\Boostrap\Boostrap> ./vendor/bin/phpunit tests/SalidaTest.php**

**PHPUnit 11.5.20 by Sebastian Bergmann and contributors.**

**Runtime: PHP 8.2.12**

**Runtime: PHP 8.2.12**

**.. 2 / 2 (100%)**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

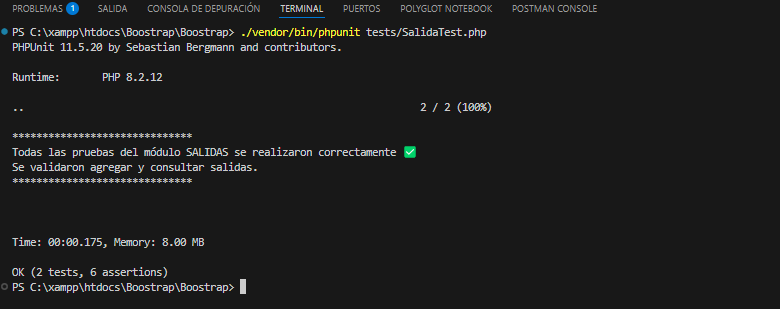
**Todas las pruebas del módulo SALIDAS se realizaron correctamente ✅**

**Se validaron agregar y consultar salidas.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Time: 00:00.175, Memory: 8.00 MB**

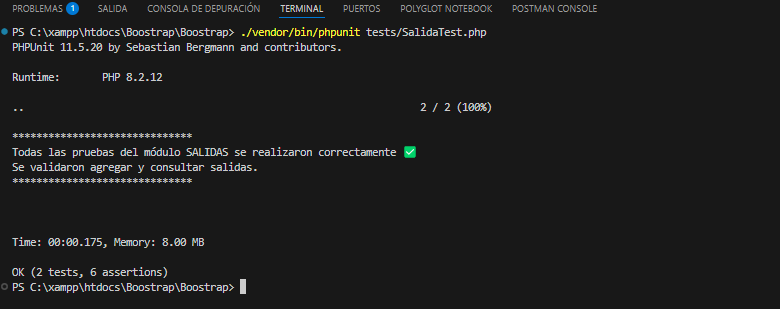
**OK (2 tests, 6 assertions)**



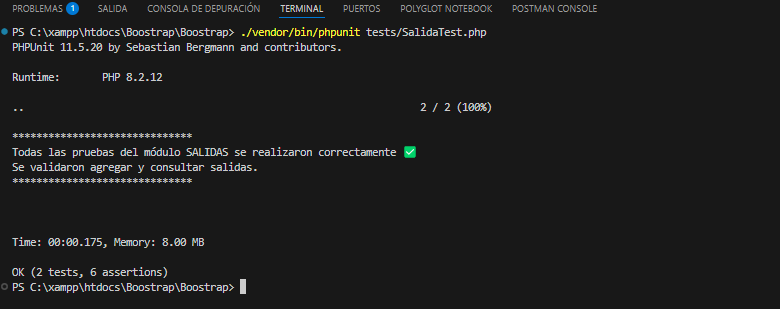
**5. Ejecución de las Pruebas**

Para ejecutar las pruebas, en la terminal se debe correr el comando:

**vendor/bin/phpunit tests/IngresoTest.php**



Este comando ejecuta todas las pruebas dentro del directorio /tests y muestra un reporte con resultados.



**Conclusión**

La integración de PHPUnit en el proyecto PHP puro permitió asegurar la calidad y estabilidad del módulo de productos, usuarios, ticket de salida y ticket de ingreso. Las pruebas unitarias desarrolladas cubren las operaciones CRUD principales y las consultas a tablas relacionadas, facilitando la detección temprana de errores y regresiones en el código.