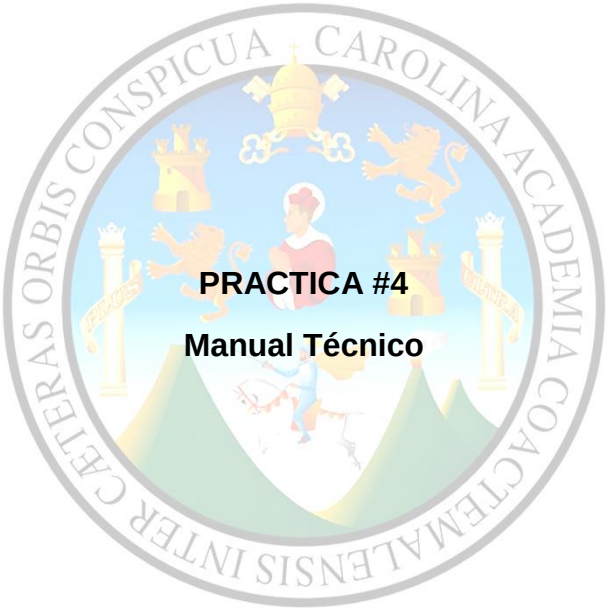
Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Análisis y Diseño de Sistemas 1, Sección “N”

Ing. José Ricardo Morales Prado



PRACTICA #4

Manual Técnico

Grupo no. 9

Carne	Nombre
200711904	William Antonio López Morales
201503936	Elmer Orlando Real Ixcayau
201504200	Denilson Eduardo Argueta Higueros
201513700	Christian Adolfo Real Ixcayau
201503666	Miguel Ángel Omar Ruano Roca
201503733	Daniel Eduardo García Paiz

Guatemala, 3 de noviembre de 2,018

Contenido

PRIORIDAD	NOMBRE TAREA	ENCARGADO
1	Pruebas TDD a Funciones	Elmer Real
2	Pruebas a las Vistas	William López
3	CodeCoverage	Elmer Real/William López
4	Autenticación de usuarios: Los usuarios deberán de identificarse en el sistema para acceder de acuerdo a su nivel de accesibilidad en el mismo, si es un cliente, ingresa al sistema de catálogos de productos.	Christian Real
5	Registrar usuarios: El sistema permitirá al usuario registrarse al sistema, solicitando distintos datos como nombre, correo, contraseña y dirección.	Christian Real
6	Modificar usuarios: El sistema permite al administrador modificar los datos de todos los usuarios.	Christian Real
7	Eliminar usuarios: El administrador de la página podrá dar de baja a cualquier usuario.	Christian Real
8	Consultar productos: El sistema ofrecerá al usuario información general de los productos que puede comprar, las cantidades de existencia y el precio del producto que desea tomar.	Daniel García
9	Registrar productos: Permite al usuario administrador, registrar un nuevo producto al sistema, solicitando la información más relevante de un nuevo producto	Daniel García
10	Modificar productos: El administrador podrá modificar los productos en caso se quiera aumentar el stock o hacer un cambio en el precio.	Daniel García
11	Eliminar productos: El administrador podrá eliminar los productos en caso ya no se desean comercializar	Daniel García
12	Finalizar Estado carrito: El cliente antes de realizar la compra deberá de finalizar la compra en el carrito de compras.	Miguel Ruano
13	Agregar a carrito: Permite al usuario o cliente, agregar al carrito de compra el producto que desee. Al momento de agregar el producto, se muestra el subtotal de su compra.	Miguel Ruano
14	Modificar carrito: El usuario podrá añadir productos al carrito de compras	Miguel Ruano

15	Eliminar ítem de carrito: El usuario podrá eliminar productos que no se desean adquirir del carrito de compras.	Miguel Ruano
16	Registrar orden de envió: El administrador podrá agregar a la cola de envíos nuevos pedidos	Miguel Ruano
17	modificar estado de envió: El administrador podrá cambiar el estado de pendiente a entregado de los envíos.	Denilson Argueta
18	cancelar envió: El usuario podrá cancelar el envió del pedido en caso ya no lo desee	Denilson Argueta
19	Realizar pago: Permite al usuario realizar el pago de los productos que desea comprar, los cuales fueron agregados al carrito de compra.	Denilson Argueta
20	ver factura: El usuario podrá tener una copia de la factura sobre la compra realizada	Miguel Ruano
21	Listar factura: El administrador podrá ver todas las facturas generadas	Denilson Argueta
22	Cambiar estado factura: El administrador podrá cambiar el estado de las facturas al ser estas canceladas	Denilson Argueta
23	(TDD) Pruebas de los controladores que devuelven las vistas	William López
24	(TDD) Pruebas del método de que calcula el Valor a devolver al Cliente cuando pague una cantidad mayor a su saldo.	Christian Real
25	Creación de la lógica, controladores, rutas y organizador de las vistas de la funcionalidad de Pago en Efectivo.	Denilson Argueta
26	Creación de la vista del Pago en efectivo del cliente.	Miguel Ruano
27	Método para Calcular el Valor a Devolver al cliente y cálculo de SubTotal y Total a pagar.	Elmer Real
28	Creación de la Vista de Valor a Devoler al cliente	Daniel García

Link de Repositorio: https://github.com/AYD1G5/practica3_grupo5/

REQUERIMIENTOS

REQUERIMIENTOS FUNCIONALES

Requisito funcional 1:

• Autenticación de usuarios: Los usuarios deberán de identificarse en el sistema para acceder de acuerdo a su nivel de accesibilidad en el mismo, si es un cliente, ingresa al sistema de catálogos de productos.

Requisito funcional 2:

• Consultar productos: El sistema ofrecerá al usuario información general de los productos que puede comprar, las cantidades de existencia y el precio del producto que desea tomar.

Requisito funcional 3:

• Registro de usuarios: El sistema permitirá al usuario registrarse al sistema, solicitando distintos datos como nombre, correo, contraseña y dirección.

Requisito funcional 4:

• Modificar: El sistema permite al administrador modificar los datos de todos los usuarios.

Requisito funcional 5:

• Agregar a carrito: Permite al usuario o cliente, agregar al carrito de compra el producto que desee. Al momento de agregar el producto, se muestra el subtotal de su compra.

Requisito funcional 6:

• Realizar pago: Permite al usuario realizar el pago de los productos que desea comprar, los cuales fueron agregados al carrito de compra.

Requisito funcional 7:

• Registro de productos: Permite al usuario administrador, registrar un nuevo producto al sistema, solicitando la información más relevante de un nuevo producto.

Requisito funcional 8:

• Pagos en Efectivo: esta opción permite registrar el pago de las facturas que tiene generado un cliente, se muestra el total por pagar, y tiene la opción de ingresar un montón el cual rebajara ese monto por pagar, y si el cliente paga una cantidad mayor a su saldo, se calcula y muestra el valor por devolver (Vuelto).

REQUERIMIENTOS NO FUNCIONALES

1. Requerimiento de rendimiento:

a. Garantizar que el diseño de las consultas u otro proceso no afecte el desempeño de la base de datos, ni el sistema.

2. Seguridad:

a. Garantizar la seguridad de la información proporcionada por los diferentes usuarios. Los registros podrán ser consultados y actualizados permanente y simultáneamente.

3. Fiabilidad:

a. El sistema debe de proveer una interfaz de uso sencilla e intuitiva. Debe de ajustarse a las características de la empresa que solicite el servicio.

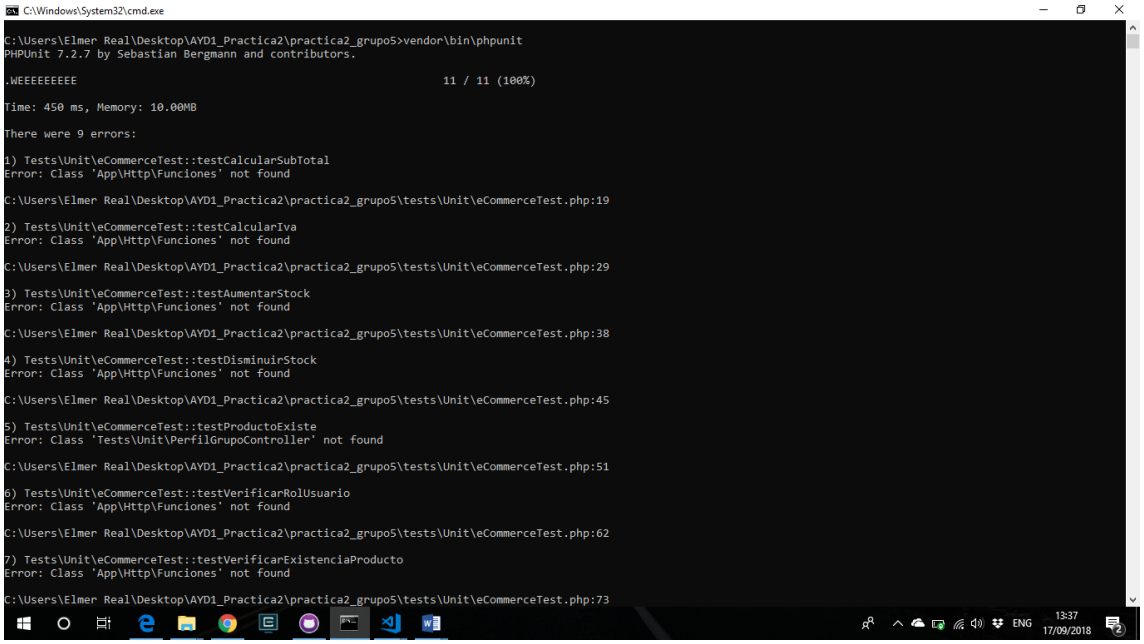
4. Disponibilidad:

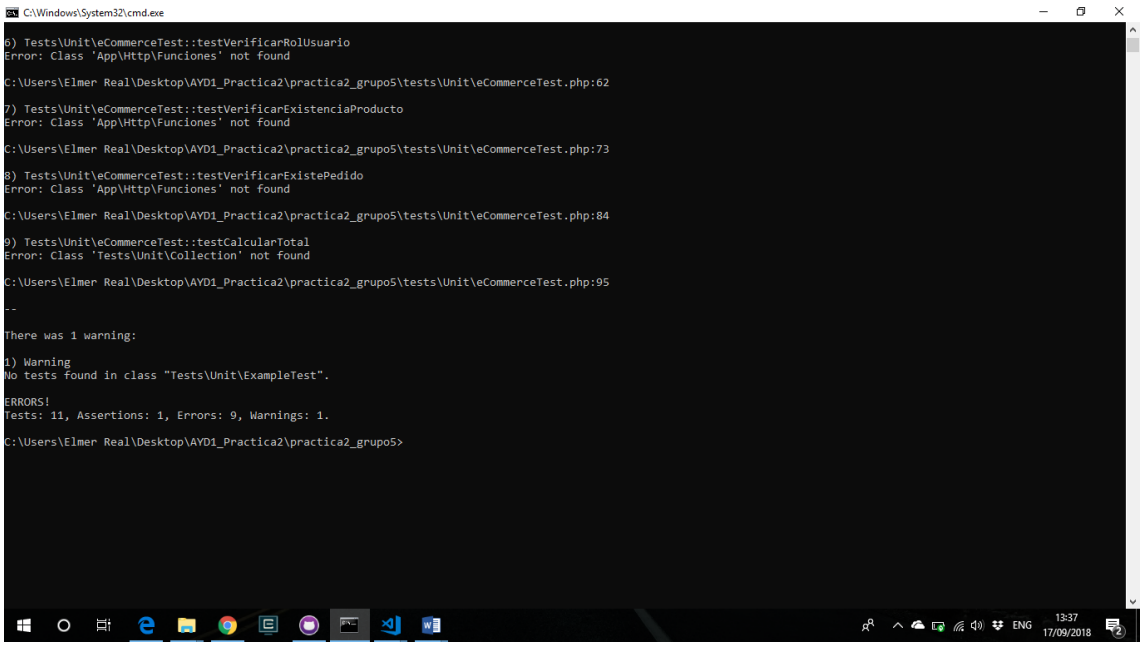
a. La disponibilidad del sistema debe de ser continua, funcionando para que el usuario realice su compra en cualquier momento.

5. Portabilidad:

a. El sistema al ser una plataforma web debe de ser compatible con cualquier sistema operativo

PRUEBAS UNITARIAS (TDD)





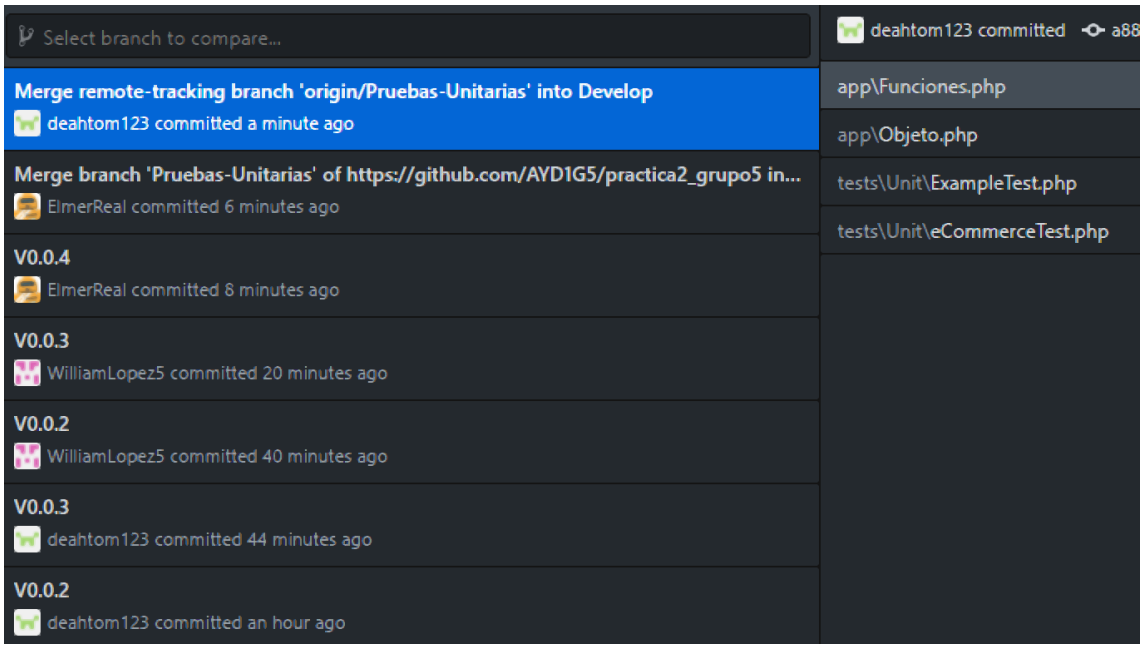
C:\Windows\System32\cmd.exe

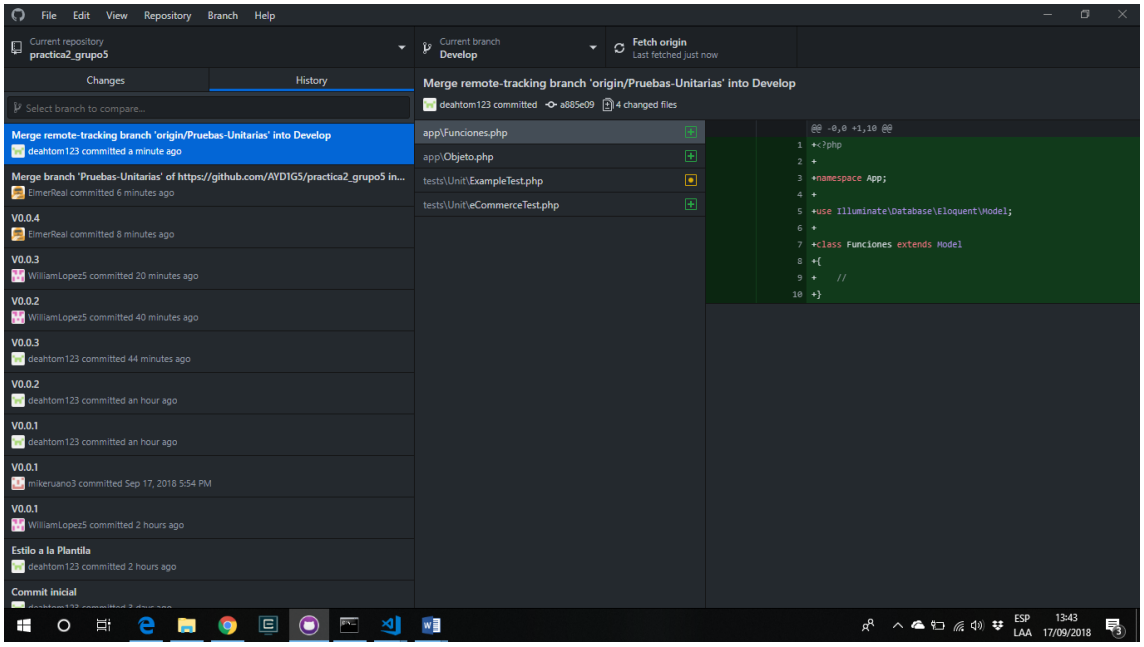
```
C:\Users\Elmer Real\Desktop\AYD1_Practica2\practica2_grupo5>vendor\bin\phpunit
PHPUnit 7.2.7 by Sebastian Bergmann and contributors.

.WEEEEEEEEE                                                       11 / 11 (100%)

Time: 450 ms, Memory: 10.00MB

There were 9 errors:

1) Tests\Unit\eCommerceTest::testCalcularSubTotal
Error: Class 'App\Http\Funciones' not found

C:\Users\Elmer Real\Desktop\AYD1_Practica2\practica2_grupo5\tests\Unit\eCommerceTest.

2) Tests\Unit\eCommerceTest::testCalcularIva
Error: Class 'App\Http\Funciones' not found

C:\Users\Elmer Real\Desktop\AYD1_Practica2\practica2_grupo5\tests\Unit\eCommerceTest.

3) Tests\Unit\eCommerceTest::testAumentarStock
Error: Class 'App\Http\Funciones' not found
```

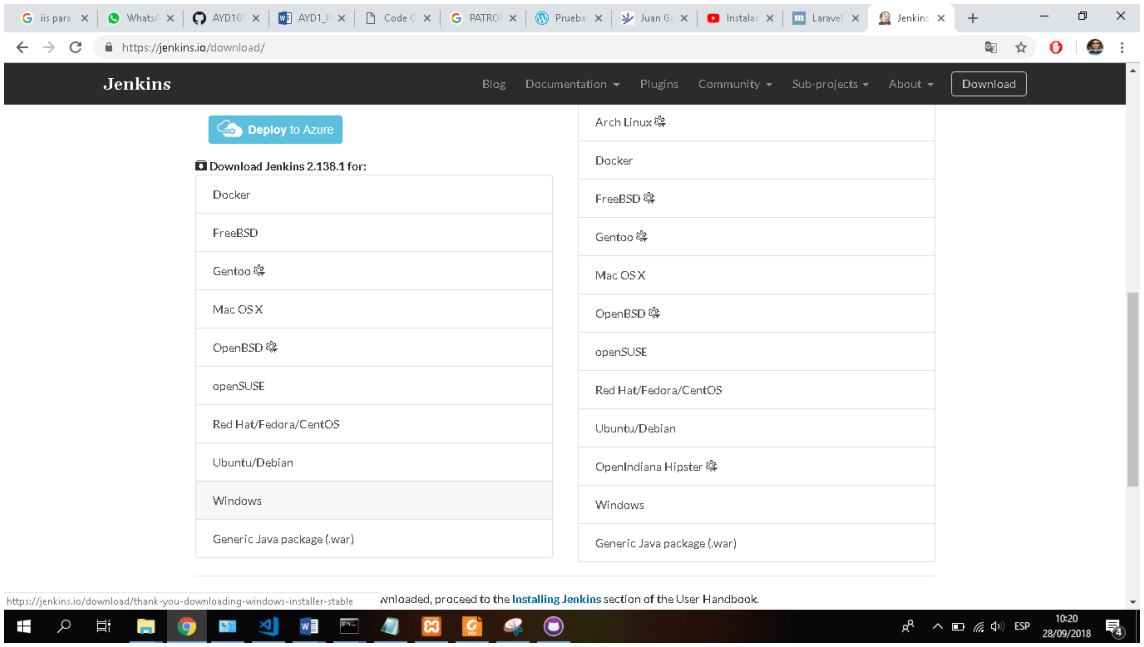
```
C:\Users\Elmer Real\Desktop\AYD1_Practica2\practica2_grupo5\tests\Unit\eCommerceTest.

9) Tests\Unit\eCommerceTest::testCalcularTotal
Error: Class 'Tests\Unit\Collection' not found

C:\Users\Elmer Real\Desktop\AYD1_Practica2\practica2_grupo5\tests\Unit\eCommerceTest.

--

There was 1 warning:

1) Warning
No tests found in class "Tests\Unit\ExampleTest".

ERRORS!
Tests: 11, Assertions: 1, Errors: 9, Warnings: 1.

C:\Users\Elmer Real\Desktop\AYD1_Practica2\practica2_grupo5>
```

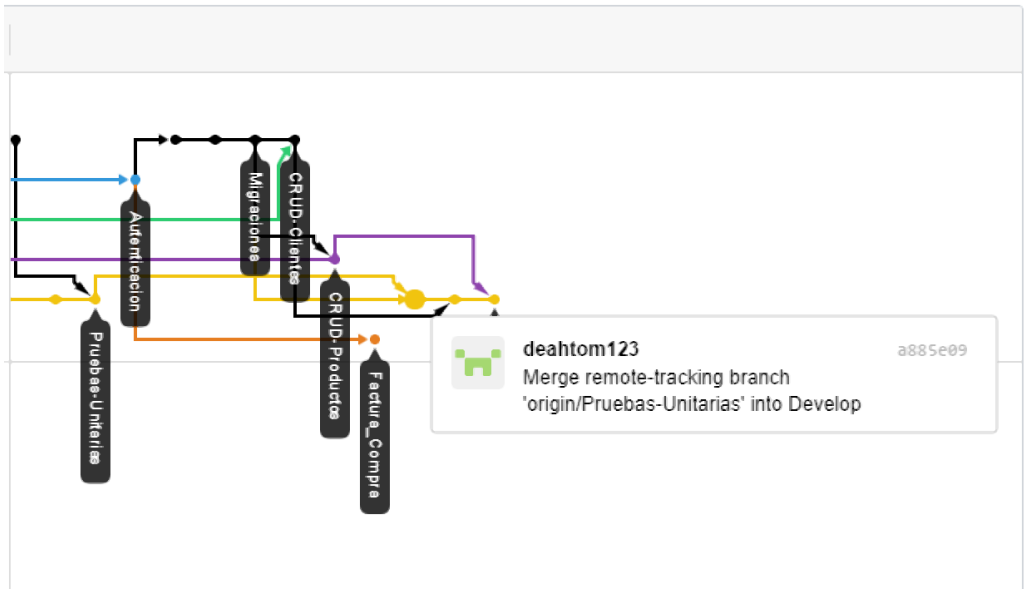
```
C:\xampp\htdocs\practica2_grupo5>vendor\bin\phpunit --coverage-html CodeCoverage
PHPUnit 7.3.5 by Sebastian Bergmann and contributors.

.....................................................             53 / 53 (100%)

Time: 1.15 minutes, Memory: 22.00MB

OK (53 tests, 71 assertions)

Generating code coverage report in HTML format ... done

C:\xampp\htdocs\practica2_grupo5>
```

```
C:\xampp\htdocs\practica3_grupo5>vendor\bin\phpunit --coverage-html CodeCoverage
PHPUnit 7.3.5 by Sebastian Bergmann and contributors.

.....................................................           58 / 58 (100%)

Time: 1.77 minutes, Memory: 24.00MB

OK (58 tests, 75 assertions)

Generating code coverage report in HTML format ... done

C:\xampp\htdocs\practica3_grupo5>
```






Jenkins
Blog
Documentation
Plugins
Community
Sub-projects
About
Download
Deploy to Azure
Download Jenkins 2.138.1 for:
Docker
FreeBSD
Gentoo
Mac OS X
OpenBSD
openSUSE
Red Hat/Fedora/CentOS
Ubuntu/Debian
Windows
Generic Java package (.war)
Arch Linux
Docker
FreeBSD
Gentoo
Mac OS X
OpenBSD
openSUSE
Red Hat/Fedora/CentOS
Ubuntu/Debian
OpenIndiana Hipster
Windows
Generic Java package (.war)
https://jenkins.io/download/thank-you-downloading-windows-installer-stable
Installing Jenkins
ESP
10:20
28/09/2018


Autenticacion
Migraciones
CRUD-Clientes
CRUD-Productos
Pruebas-Unitarias
Feature_Compra
deahtom123
a885e09
Merge remote-tracking branch
'origin/Pruebas-Unitarias' into Develop

CODE COVERAGE

C:\xampp\htdocs\practica2_grupo5\app / (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		92.33%	325 / 352		88.73%	63 / 71		71.43%	15 / 21
Console		100.00%	4 / 4		100.00%	2 / 2		100.00%	1 / 1
Exceptions		100.00%	3 / 3		100.00%	2 / 2		100.00%	1 / 1
Http		91.11%	246 / 270		84.78%	39 / 46		61.54%	8 / 13
Providers		87.50%	21 / 24		88.89%	8 / 9		80.00%	4 / 5
Carrito.php		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0
Carrito_Producto.php		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0
Detalle_Compra.php		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0
Factura.php		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0
Factura_Producto.php		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0
Funciones.php		100.00%	51 / 51		100.00%	12 / 12		100.00%	1 / 1
Objeto.php		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0
Producto.php		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0
User.php		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0

```
C:\xampp\htdocs\practica3_grupo5>vendor\bin\phpunit --coverage-html CodeCoverage
PHPUnit 7.3.5 by Sebastian Bergmann and contributors.

.........................................................        58 / 58 (100%)

Time: 1.77 minutes, Memory: 24.00MB

OK (58 tests, 75 assertions)

Generating code coverage report in HTML format ... done

C:\xampp\htdocs\practica3_grupo5>
```

Código de Pruebas:

```
    /**Evaluar la respuesta del metodo del controlador que
    * devulve la vista Carrito/FinalizarCompra
    */
    public function testCarritoFinalizarCompra(){
        //Arrange (Preparar)
            //crear un usuario
        $usuario = new User();
        $usuario->name = 'NuevoNF';
        $usuario->apellido = 'ApellidoNF';
        $usuario->nit = '333-3NF';
        $usuario->email = '1NF@gmail.com';
        $usuario->password = Hash::make('PasswordNF');
        $usuario->rol = '1';
        $usuario->save();
        $carrito = new Carrito();
        $carrito->id_user = $usuario->id;
        $carrito->save();
                //autenticarse
                $response = $this->call('POST', '/login', [
                    'email' =>  $usuario->email,
                    'password' =>'PasswordNF',
                    '_token' => csrf_token()
                ]);
                //Establecer respuesta correcta

        $RespuestaCorrecta=200; //Codigo HTTP de respuesta correcta

        //Act (Actuar)
        $llamaVista=$this->get('Carrito/FinalizarCompra');
        $respuestaFuncion=$llamaVista->getStatusCode();
        $carrito->delete();
        $usuario->delete();

        //Assert (Afirmar)
        $this->assertEquals($RespuestaCorrecta,$respuestaFuncion);
    }

    /**Evaluar la respuesta del metodo del controlador que
    * devulve la vista Carrito/FinalizarCompra
    */
    public function testCarritoFinalizarCompraTexto(){
        //Arrange (Preparar)
            //crear un usuario
        $usuario = new User();
        $usuario->name = 'NuevoNF';
        $usuario->apellido = 'ApellidoNF';
        $usuario->nit = '333-3NF';
```

```
        $usuario->email = '1NF@gmail.com';
        $usuario->password = Hash::make('PasswordNF');
        $usuario->rol = '1';
        $usuario->save();
        $carrito = new Carrito();
        $carrito->id_user = $usuario->id;
        $carrito->save();
                //autenticarse
                $response = $this->call('POST', '/login', [
                    'email' =>  $usuario->email,
                    'password' =>'PasswordNF',
                    '_token' => csrf_token()
                ]);
                //Establecer respuesta correcta

        $RespuestaCorrecta='Pago en Efectivo'; //Codigo HTTP de respuesta
correcta

        //Act (Actuar)
        $llamaVista=$this->get('Carrito/FinalizarCompra');

        $carrito->delete();
        $usuario->delete();

        //Assert (Afirmar)
        $llamaVista->assertSeeText($RespuestaCorrecta);
    }

    /**
    * Prueba para la funcion  vuelto
    */
    public function testCalcularVuelto()
    {
        //Arrange (Preparar)
        $newFuncionController=new NewFuncionController();
        $total=50;
        $pagar=100;

        //Act (Actuar)
        $respuestaFuncion=$newFuncionController->vuelto($total,$pagar);
        $RespuestaCorrecta=$pagar-$total;

        //Assert (Afirmar)
        $this->assertEquals($RespuestaCorrecta,$respuestaFuncion);

    }
```

```php
class eCommerceTest extends TestCase
{
    /**
    * Prueba para la funcion calcular SubTotal de Compra
    */
    public function testCalcularSubTotal()
    {
        //Arrange (Preparar)
        $funciones=new Funciones();
        $prod = new Producto();
        $prod->nombre = 'ProductoX';
        $prod->descripcion='Producto de Prueba';
        $prod->cantidad_disponible='100';
        $prod->ruta_imagen='Db.jpg';
        $prod->precio = '150';
        $prod->save();

        //Act (Actuar)
        $respuestaFuncion=$funciones->calcularSubTotal($prod-
>cantidad_disponible,$prod->precio);
        $RescpuestaCorrecta=$prod->cantidad_disponible*$prod->precio;
        $prod->delete();

        //Assert (Afirmar)
        $this->assertEquals($RescpuestaCorrecta,$respuestaFuncion);

    }

    /**
    * Prueba para la funcion calcular IVA
    */
    public function testCalcularIva()
    {
        //Arrange (Preparar)
        $funciones=new Funciones();
        $prod = new Producto();
        $prod->nombre = 'ProductoX';
        $prod->descripcion='Producto de Prueba';
        $prod->cantidad_disponible='100';
        $prod->ruta_imagen='Db.jpg';
        $prod->precio = '150';
        $prod->save();
        $razonIVA=1.12;
        $IVA=0.12;

        //Act (Actuar)
        $costo=$prod->precio/$razonIVA;
        $RescpuestaCorrecta=$costo*$IVA;
        $respuestaFuncion=$funciones->calcularIva($prod->precio);
```

```
        $prod->delete();

        //Assert (Afirmar)
        $this->assertEquals($RescpuestaCorrecta,$respuestaFuncion);

    }

    /**
    * Prueba para la funcion Aumentar Stock
    */
    public function testAumentarStock()
    {
        //Arrange (Preparar)
        $funciones=new Funciones();
        $prod = new Producto();
        $prod->nombre = 'ProductoX';
        $prod->descripcion='Producto de Prueba';
        $prod->cantidad_disponible='100';
        $prod->ruta_imagen='Db.jpg';
        $prod->precio = '100';
        $prod->save();

        //Act (Actuar)
        $aumento=50;
        $respuestaFuncion=$funciones->aumentarStock($prod-
>id_producto,$aumento);
        $RescpuestaCorrecta=$prod->cantidad_disponible+$aumento;
        $prod->delete();

        //Assert (Afirmar)
        $this->assertEquals($RescpuestaCorrecta,$respuestaFuncion);
    }
    /**
    * Prueba para la funcion calcular IVA
    */
    public function testDisminuirStock()
    {
        //Arrange (Preparar)
        $funciones=new Funciones();
        $prod = new Producto();
        $prod->nombre = 'ProductoX';
        $prod->descripcion='Producto de Prueba';
        $prod->cantidad_disponible='100';
        $prod->ruta_imagen='Db.jpg';
        $prod->precio = '100';
        $prod->save();

        //Act (Actuar)
        $disminucion=50;
```

```
        $respuestaFuncion=$funciones->disminuirStock($prod-
>id_producto,$disminucion);
        $RescpuestaCorrecta=$prod->cantidad_disponible-$disminucion;
        $prod->delete();

        //Assert (Afirmar)
        $this->assertEquals($RescpuestaCorrecta,$respuestaFuncion);
    }

    public function testProductoExiste()
    {
        //Arrange (Preparar)
        $funciones=new Funciones();
        $prod = new Producto();
        $prod->nombre = 'ProductoX';
        $prod->descripcion='Producto de Prueba';
        $prod->cantidad_disponible='100';
        $prod->ruta_imagen='Db.jpg';
        $prod->precio = '100';
        $prod->save();

        //Act (Actuar)
        $respuestaFuncion=$funciones->productoExiste($prod->id_producto);
        $prod->delete();

        //Assert (Afirmar)
        $this->assertTrue($respuestaFuncion);
    }

    /**
     * Prueba para averiguar el rol del usuario especifico.
     *
     */
    public function testVerificarRolUsuario()
    {
        //Arrange (Preparar)
        $funciones=new Funciones();
        $usuario = new User();
        $usuario->name = 'NameX';
        $usuario->apellido = 'ApellidoX';
        $usuario->nit = '333-3';
        $usuario->email = 'name@gmail.com';
        $usuario->password = 'PasswordX';
        $usuario->rol = '1';
        $usuario->save();

        //Act (Actuar)
        $respuestaFuncion=$funciones->verificarRolUsuario($usuario->id);
        $RescpuestaCorrecta=$usuario->rol;
```

```
        $usuario->delete();

        //Assert (Afirmar)

        $this->assertEquals($RescpuestaCorrecta,$respuestaFuncion);
    }

    /**
     * Prueba para averiguar si de un producto especifico se puede
comprar cierta cantidad de elementos.
     * Respuesta =True
     */
    public function testVerificarExistenciaProducto()
    {
        //Arrange (Preparar)
        $funciones=new Funciones();
        $prod = new Producto();
        $prod->nombre = 'ProductoX';
        $prod->descripcion='Producto de Prueba';
        $prod->cantidad_disponible='100';
        $prod->ruta_imagen='Db.jpg';
        $prod->precio = '100';
        $prod->save();

        //Act (Actuar)
        $cantidaRetirar=50;
        $respuestaFuncion=$funciones->verificarExistenciaProducto($prod-
>id_producto,$cantidaRetirar);
        $prod->delete();

        //Assert (Afirmar)
        $this->assertTrue($respuestaFuncion);
    }

        /**
     * Prueba para averiguar si de un producto especifico se puede
comprar cierta cantidad de elementos.
     * Respuesta =false
     */
    public function testVerificarExistenciaProducto2()
    {
        //Arrange (Preparar)
        $funciones=new Funciones();
        $prod = new Producto();
        $prod->nombre = 'ProductoX';
        $prod->descripcion='Producto de Prueba';
        $prod->cantidad_disponible='100';
        $prod->ruta_imagen='Db.jpg';
        $prod->precio = '100';
```

```
        $prod->save();

        //Act (Actuar)
        $cantidaRetirar=150;
        $respuestaFuncion=$funciones->verificarExistenciaProducto($prod->id_producto,$cantidaRetirar);
        $prod->delete();

        //Assert (Afirmar)
        $this->assertTrue(!$respuestaFuncion);
    }

    /**
     * Prueba para sumar el subtotal de una coleccion.
     *
     */
    public function testCalcularTotal()
    {
        //Arrange (Preparar)
        $funciones=new Funciones();
        $subtotalesCollection = new Collection();
        $objeto1 = new Objeto();
        $objeto1->subtotal = 50;
        $subtotalesCollection->push($objeto1);
        $objeto2 = new Objeto();
        $objeto2->subtotal = 120;
        $subtotalesCollection->push($objeto2);
        $objeto3 = new Objeto();
        $objeto3->subtotal = 70;
        $subtotalesCollection->push($objeto3);
        $objeto4 = new Objeto();
        $objeto4->subtotal = 250;
        $subtotalesCollection->push($objeto4);

        //Act (Actuar)
        $respuestaFuncion=$funciones->calcularTotal($subtotalesCollection);
        $RescpuestaCorrecta=490;

        //Assert (Afirmar)
        $this->assertEquals($RescpuestaCorrecta,$respuestaFuncion);
    }

        /**
     * Prueba para averiguar el estado del pedido realizado por un cliente.
     *
     */
    public function testVerificarExistePedido()
```

```
    {
        //Arrange (Preparar)
        $funciones=new Funciones();
        $factura=new Factura();
        $factura->id_user = '1';
        $factura->fecha = '21-09-18, 12:31:49 AM';
        $factura->estado = '0';
        $factura->total = '1000';
        $factura->save();

        //Act (Actuar)
        $respuestaCorrecta=1;
        $respuestaFuncion=$funciones->verificarExistePedido($factura-
>id_factura);
        $factura->delete();

        //Assert (Afirmar)
        $this->assertEquals($respuestaCorrecta,$respuestaFuncion);
    }

     /**
     * Prueba para averiguar si el stock de productos puede satisfacer la
compra a realizar.
     * Respuesta: True
     */

    public function testVerificarStock()
    {
        //Arrange (Preparar)
        $funciones=new Funciones();
        $prod = new Producto();
        $prod->nombre = 'ProductoX';
        $prod->descripcion='Producto de Prueba';
        $prod->cantidad_disponible='100';
        $prod->ruta_imagen='Db.jpg';
        $prod->precio = '150';
        $prod->save();

        //Act (Actuar)
        $disminucion=50;
        $respuestaFuncion=$funciones->verificarStock($prod-
>id_producto,$disminucion);
        $RescpuestaCorrecta=true;
        $prod->delete();

        //Assert (Afirmar)
        $this->assertEquals($RescpuestaCorrecta,$respuestaFuncion);

    }
```

```php
    /**
     * Prueba para averiguar si el stock de productos puede satisfacer la
compra a realizar.
     * Respuesta: False
     */
    public function testVerificarStock2()
    {
        //Arrange (Preparar)
        $funciones=new Funciones();
        $prod = new Producto();
        $prod->nombre = 'ProductoX';
        $prod->descripcion='Producto de Prueba';
        $prod->cantidad_disponible='100';
        $prod->ruta_imagen='Db.jpg';
        $prod->precio = '150';
        $prod->save();

        //Act (Actuar)
        $disminucion=150;
        $respuestaFuncion=$funciones->verificarStock($prod-
>id_producto,$disminucion);
        $RescpuestaCorrecta=false;
        $prod->delete();

        //Assert (Afirmar)
        $this->assertEquals($RescpuestaCorrecta,$respuestaFuncion);

    }

   /**
     * Prueba para verificiar el metodo que realiza aumentos en el
carrito de
     * compras de cierto usuario.
     *
     */
    public function testAumentarCarro()
    {
        //Arrange (Preparar)
        $funciones=new Funciones();
        $usuario = new User();
        $usuario->name = 'NameX';
        $usuario->apellido = 'ApellidoX';
        $usuario->nit = '333-3';
        $usuario->email = 'name@gmail.com';
        $usuario->password = 'PasswordX';
        $usuario->rol = '1';
        $usuario->no_items = '7';
        $usuario->save();
```

```
        $aumento=5;

        //Act (Actuar)
        $respuestaFuncion=$funciones->aumentarCarro($usuario-
>id,$aumento);
        $usuario2=User::where('id',$usuario->id)->first();
        $RescpuestaCorrecta=$usuario2->no_items;
        $usuario->delete();

        //Assert (Afirmar)

        $this->assertEquals($RescpuestaCorrecta,$respuestaFuncion);
    }

   /**
     * Prueba para verificiar el metodo que realiza aumentos en el
carrito de
     * compras de cierto usuario.
     *
     */
    public function testDisminuirCarro()
    {
        //Arrange (Preparar)
        $funciones=new Funciones();
        $usuario = new User();
        $usuario->name = 'NameX';
        $usuario->apellido = 'ApellidoX';
        $usuario->nit = '333-3';
        $usuario->email = 'name@gmail.com';
        $usuario->password = 'PasswordX';
        $usuario->rol = '1';
        $usuario->no_items = '7';
        $usuario->save();
        $aumento=5;

        //Act (Actuar)
        $respuestaFuncion=$funciones->disminuirCarro($usuario-
>id,$aumento);
        $usuario2=User::where('id',$usuario->id)->first();
        $RescpuestaCorrecta=$usuario2->no_items;
        $usuario->delete();

        //Assert (Afirmar)

        $this->assertEquals($RescpuestaCorrecta,$respuestaFuncion);
    }
```