# Pruebas unitarias Wishlist:

## Creación de Fixtures:

Los datos insertados en la Base de Datos, para hacer los tests, son los siguientes, de acuerdo al modelo :

```php
class WishFixture extends CakeTestFixture {

    public $fields = array(
        'id' => array('type' => 'integer', 'key' => 'primary'),
        'product_id' => array('type' => 'integer', 'null' => false),
        'user_id' => array('type' => 'integer', 'null' => false),
        'created' => 'datetime'
    );

    public $records = array(

        array(
            'id' => 1,
            'product_id'=>4,
            'user_id'=>1,
            'created' => '2014-10-16 15:56:23'
            ),
        array(
            'id' => 2,
            'product_id'=>5,
            'user_id'=>2,
            'created' => '2014-10-16 15:56:23'
            ),
    );
}
```

1Fixture Modelo Wish

## Pruebas del modelo

Para probar el modelo y así comprobar que la comunicación con la base de datos está bien se hicieron dos test en el modelo.

Estos fueron, el primero, sirve para probar que al pedir todos los productos de la base de datos efectivamente se retornen todos los registros que deberían estar en la base de datos.

Para lograr probarlo se creó la función **getAllWishes()** en el modelo, dicha función es la siguiente:

```php
public function getAllWishes(){
    return $this->find('all',array(
        'fields' => array('id','user_id','product_id')));
}
```

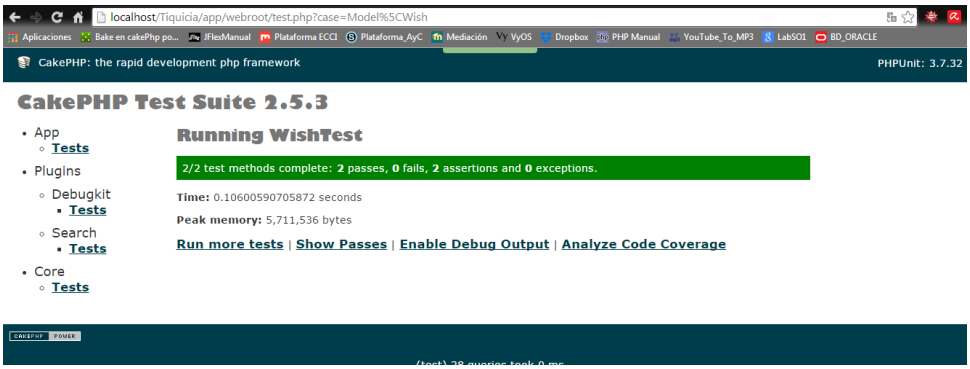
2Función getAllWishes

El segundo sirve para probar si efectivamente las condiciones de consulta se hacen correctamente desde el modelo, para esto se creó la función **getWishesFiltrados($id)**, el cual por medio del **id**, llave primaria, como criterio de consulta, se restringe la devolución de registros. La función **getWishesFiltrados($id)** es la siguiente:

```php
public function getWishesFiltrados($id = null) {
    return $this->find('first', array(
        'conditions' => array('Wish.id' => $id),
        'fields' => array('id','user_id','product_id')
    ));
}
```

3Función getWishesFiltrados

El resultado de ejecutar las pruebas del modelo **Wish** se pueden observar en la figura 4.




4Resultado Test al Modelo Wish

Ahora si se observa la impresión de las consultas de la base de datos de la figura 5 se notan los dos **select (resaltado)** que se hacen el primero que pide todos los registros y el segundo que está restringido con respecto al id = 1. Que corresponden a los llamados de las funciones **getAllWishes** y **getWishesFiltrados** :

| | | | | |
|---|---|---|---|---|
| 12 | INSERT INTO `test`.`wishes` (`id`, `product_id`, `user_id`, `created`) VALUES (?,?,?,?) , params[ 2, 5, 2, 2014-10-16 15:56:23 ] | 0 | 0 | 0 |
| 13 | COMMIT | 0 | 0 | 0 |
| 14 | SELECT `Wish`.`id`, `Wish`.`user_id`, `Wish`.`product_id` FROM `test`.`wishes` AS `Wish` LEFT JOIN `test`.`users` AS `User` ON (`Wish`.`user_id` = `User`.`id`) LEFT JOIN `test`.`products` AS `Product` ON (`Wish`.`product_id` = `Product`.`id`) WHERE 1 = 1 | 2 | 2 | 0 |
| 15 | BEGIN | 2 | 2 | 0 |
| 16 | INSERT INTO `test`.`products` (`id`, `name`, `category`, `subcategory`, `subsubcategory`, `type`, `price`, `weight`, `unit`, `filename`, `dir`, `keywords`, `volumen`, `created`, `updated`, `stock`, `state`, `category_id`, `subcategory_id`, `subsubcategory_id`) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?) , params[ 4, Balon Nike, Futbol, Balones, Nike, El mejor balon de todos, 450, 3, Kg, bolaNike.jpg, img\uploads\product\filename, nike Balon, 50, 2014-10-16 15:56:23, 2014-10-17 15:56:23, 15, 1, 3, 4, 1 ] | 2 | 2 | 0 |
| 17 | INSERT INTO `test`.`products` (`id`, `name`, `category`, `subcategory`, `subsubcategory`, `type`, `price`, `weight`, `unit`, `filename`, `dir`, `keywords`, `volumen`, `created`, `updated`, `stock`, `state`, `category_id`, `subcategory_id`, `subsubcategory_id`) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?) , params[ 5, Raquetas, Tenis, Msdera, Pintadas a mano, La mejor raqueta de todas, 450, 3, Kg, raquetas.jpg, img\uploads\product\filename, tenis raquetas, 50, 2014-10-16 15:56:23, 2014-10-17 15:56:23, 15, 1, 3, 4, 1 ] | 2 | 2 | 0 |
| 18 | COMMIT | 2 | 2 | 0 |
| 19 | BEGIN | 2 | 2 | 0 |
| 20 | INSERT INTO `test`.`users` (`id`, `first_name`, `middle_name`, `last_name`, `email`, `identification`, `birth_date`, `username`, `role`) VALUES (?,?,?,?,?,?,?,?,?) , params[ 1, Daniel, Mora, Madrigal, dan182mora@gmail.com, 123456789, 1992-12-22, danielm123, admin ] | 2 | 2 | 0 |
| 21 | INSERT INTO `test`.`users` (`id`, `first_name`, `middle_name`, `last_name`, `email`, `identification`, `birth_date`, `username`, `role`) VALUES (?,?,?,?,?,?,?,?,?) , params[ 2, Alejandro, Cordoba, Soto, coba@gmail.com, 123756789, 1992-10-26, coba2, admin ] | 2 | 2 | 0 |
| 22 | INSERT INTO `test`.`users` (`id`, `first_name`, `middle_name`, `last_name`, `email`, `identification`, `birth_date`, `username`, `role`) VALUES (?,?,?,?,?,?,?,?,?) , params[ 3, Jose, Mora, Madrigal, Jmora@gmail.com, 190456789, 2003-11-14, Jm123, admin ] | 2 | 2 | 0 |
| 23 | COMMIT | 2 | 2 | 0 |
| 24 | BEGIN | 2 | 2 | 0 |
| 25 | INSERT INTO `test`.`wishes` (`id`, `product_id`, `user_id`, `created`) VALUES (?,?,?,?) , params[ 1, 4, 1, 2014-10-16 15:56:23 ] | 2 | 2 | 0 |
| 26 | INSERT INTO `test`.`wishes` (`id`, `product_id`, `user_id`, `created`) VALUES (?,?,?,?) , params[ 2, 5, 2, 2014-10-16 15:56:23 ] | 2 | 2 | 0 |
| 27 | COMMIT | 2 | 2 | 0 |
| 28 | SELECT `Wish`.`id`, `Wish`.`user_id`, `Wish`.`product_id` FROM `test`.`wishes` AS `Wish` LEFT JOIN `test`.`users` AS `User` ON (`Wish`.`user_id` = `User`.`id`) LEFT JOIN `test`.`products` AS `Product` ON (`Wish`.`product_id` = `Product`.`id`) WHERE `Wish`.`id` = 1 LIMIT 1 | 1 | 1 | 0 |

5Consultas realizadas a la base de datos por las pruebas

## Pruebas del Controlador

Para probar el controlador de la clase Wish se hicieron 7 tests, de los cuales todos se aprobaron. Dichos tests son los siguientes:

1. **testPrueba:** Se creó la función prueba en el controlador, para simular el comportamiento del index, y no modificar la función del index original, en la figura6 se puede observar como filtra los resultados devueltos por user_id = 1 y devuelve solo los campos id,user_id y product_id que corresponden a campos del modelo Wish.
   Es por esta razón que en el test de la función prueba (ver Figura 5) se compara el registro devuleto con uno que tiene como user_id = 1, y como se puede observar en el fixture de Wish(Figura X) este registro esperado es el único con user_id = 1.

```
function prueba(){
    $wiii1 = $this->Wish->find('all', array('conditions'=>array('Wish.user_id'=> 1),'fields' => array('id','user_id','product_id')));
    $this->set('Wishes',$wiii1);
}
```

6Función que simula el comportamiento de Index pero con user_id fijo.

```php
//Se usa para probar el index
public function testPrueba() {

    $result1 = $this->testAction('/wishes/prueba/');

    $expected =
        array( 'Wish' =>
            array(
                'id' => 1,
                'product_id'=>4,
                'user_id'=>1
    ));
    $result = Hash::extract($this->vars['Wishes'], '{n}');
    $this->assertEquals($expected, $result['0']);

}
```

7Función para probar el index

2. **testDelete:** Se el modo en que se prueba está función del controlador es así: se utiliza el index para filtrar la devolución de registros a solo los wish que poseen el user_id = 1, luego de esto se llama a la función Delete, para que elimine ese registro, una vez hecho esto se pide que devuelva ese registro, pero como ya no está, se utiliza la función assertNotEquals para asegurar que son diferentes ya que el registro devuelto es nulo.

```php
public function testDelete(){
    $result1 = $this->testAction('/wishes/prueba/');
    $expected1 =
        array( 'Wish' =>
            array(
                'id' => 1,
                'product_id'=>4,
                'user_id'=>1
            ));
    $result = Hash::extract($this->vars['Wishes'], '{n}');
    $this->assertEquals($expected1, $result['0']);

    $this->testAction('/wishes/delete/1');
    $result1 = $this->testAction('/wishes/prueba/');
    $result = Hash::extract($this->vars['Wishes'], '{n}');
    $this->assertNotEquals($expected1, $result);
}
```

8Test Delete

3. **testAdd:** Para probar esta función se procedió a hacer lo siguiente: se inserta un registro a la base de datos, utilizando la función **Add** , una vez hecho esto se llama a la función **getWishesFiltrados($id_Wish)** creada en el modelo **Wish**, para que devuelva específicamente ese registro de la base de datos , el cual se compara si son iguales con el anteriormente añadido y da como resultado una prueba exitosa, por lo cual es seguro que la función **Add** funciona correctamente.

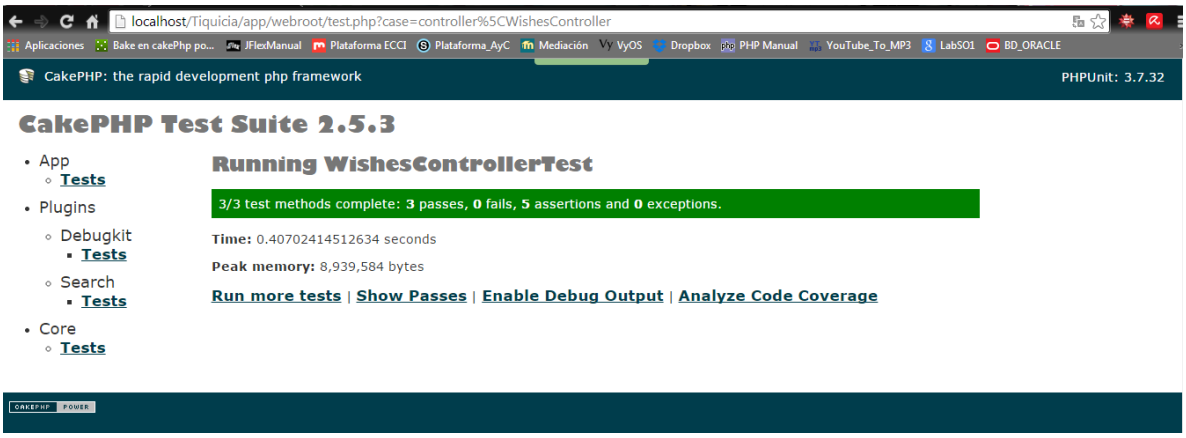
```php
public function testAdd() {
    $insercion =  array( 'Wish' =>
        array(
            'id' => 3,
            'product_id'=>4,
            'user_id'=>1
    ));

    $expected =  array( 'Wish' =>
        array(
            'id' => 3,
            'product_id'=>4,
            'user_id'=>1
        ));

    $this->testAction('/wishes/add', array('data' => $insercion, 'method' => 'post'));
    $result = $this->Wish->getWishesFiltrados(3);
    $this->assertEquals($insercion, $expected);
    $this->assertEquals($expected, $result);
}
```

9TestAdd

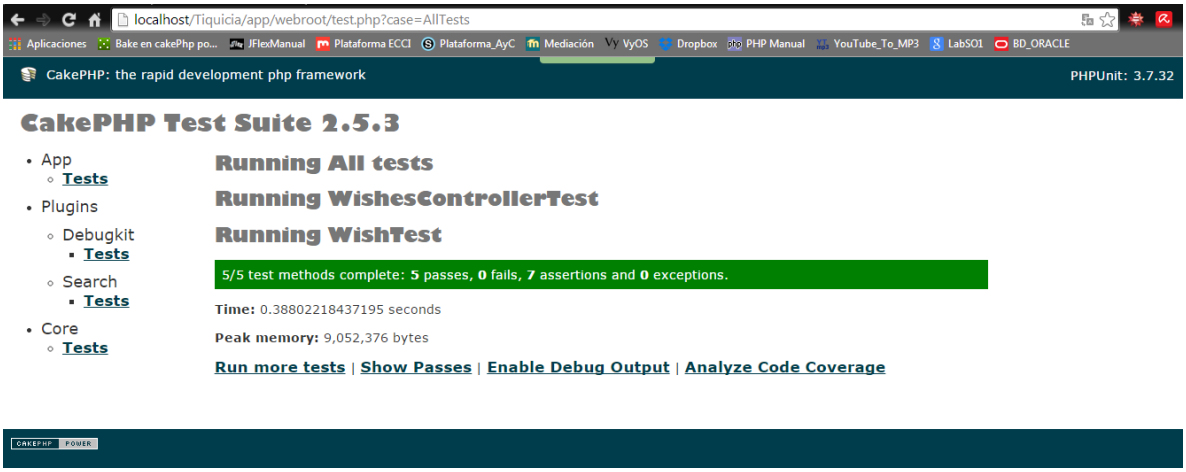
El resultado de efectuar estas pruebas es el siguiente:




10Resultado de ejecutar las pruebas unitarias del controlador

## Pruebas de todas las Pruebas:

Al correr la opción de Test.php AllTest , se podrán observar el resultado de todas las pruebas realizadas en el proyecto. Tal resultado se puede ver en la figura 11.




11Test All Tests