Codeigniter

Especialidad PHP

Felipe Izquierdo Romero

Índice

1. Descripción del problema

Crear una aplicación de gestión de Stock donde se vea una lista con todo el stock disponible con su nombre y cantidad.

La aplicación deberá poder añadir más productos y modificar los ya existentes.

2. Desarrollo

Comenzando por el archivo “Routes.php” que contiene las rutas definidas y las tres vistas que conforman la aplicación.

```
10  $routes->get('/stock', 'StockController::index');
11  $routes->get('/stock/view/(:segment)', 'StockController::view/$1');
12  $routes->get('/stock/create', 'StockController::create');
13  $routes->post('/stock/create/post', 'StockController::create_post');
14  $routes->post('/stock/remove/(:segment)', 'StockController::remove/$1');
15  $routes->get('/stock/edit/(:segment)', 'StockController::edit/$1');
16  $routes->post('/stock/update/(:segment)', 'StockController::update/$1');
```

Ilustración 1- Rutas

La primera vista sería “Stock_Index.php” que contiene una lista con todo el stock, botones para añadir producto, modificar y borrar, dos mensajes de alerta que aparecen cuando se completa con éxito o sin él una operación y un script para hacer desaparecer los errores tras 2 segundos.

```
25      <ul class="list-group">
26          <?php foreach($products as $product): ?>
27              <li class="list-group-item" style="display:inline-flex; gap: 30px;"><a class="h3"
28              href="<?=base_url('/stock/view/'.$product['id'])?>"><?= $product['name'] ?></a>
29              <p class="h3">Cantidad: <?= $product['quantity'] ?></p>
30                  <form action="<?=base_url('/stock/remove/'.$product['id'])?>" method="POST">
31                      <button type="submit" class="btn btn-danger">Eliminar producto</button>
32                  </form>
33                  <form action="<?=base_url('/stock/edit/'.$product['id'])?>" method="GET">
34                      <button type="submit" class="btn btn-warning">Modificar producto</button>
35                  </form>
36              </li>
37          <?php endforeach;?>
38      </ul>
39      <br>
40      <form action="/stock/create" method="GET">
41          <button type="submit" class="btn btn-primary">Añadir Producto</button>
42      </form>
```

Ilustración 2- Lista de stock

```
13      <?php if(session()->getFlashdata('success')): ?>
14          <div id="div-success" class="alert alert-success" style="transition: opacity 0.3s ease;">
15              <?= session()->getFlashdata('success') ?>
16          </div>
17      <?php elseif(session()->getFlashdata('error')): ?>
18          <div id="div-error" class="alert alert-danger" style="transition: opacity 0.3s ease;">
19              <?= session()->getFlashdata('error') ?>
20          </div>
21      <?php else: ?>
22          <br><br>
23      <?php endif; ?>
```

Ilustración 3- Alertas

```
<script>
    var succesAlert = document.getElementById('div-success');
    var errorAlert = document.getElementById('div-error');
    setTimeout(function() {
        succesAlert.style.opacity = 0;
    }, 2000);
    setTimeout(function() {
        errorAlert.style.opacity = 0;
    }, 2000);
</script>
```

Ilustración 4- Desaparecer alertas

La vista “Stock_View.php” contiene una lista con los datos del producto y un botón para volver al index.

```
<div>
    <h1 class="display-1"><?= $product['name'] ?></h1>
    <ul class="list-group">
        <li class="list-group-item">Cantidad: <?= $product['quantity'] ?></li>
        <li class="list-group-item">Precio: <?= $product['price'] ?></li>
    </ul>
    <br>
    <form action="<?= base_url('/stock')?>" method='GET'>
        <button class="btn btn-primary" type="submit" >Atras</button>
    </form>
</div>
```

Ilustración 5- Datos de producto

Por último “Stock_Create.php” contiene un formulario para crear un producto y si quieres modificar un producto te enviará a este formulario cargando los datos del producto que se desea modificar y un botón para realizar cualquiera de las dos opciones que variará su color y texto según su respectiva operación. Además de un botón para volver al index.

Cada etiqueta “input” de datos está dividido en un “div” y debajo otro “div” que mostrara un mensaje a la hora de creación de un producto cuando se intenta realizar la operación con un dato que no es válido.

Clic para ampliar la foto

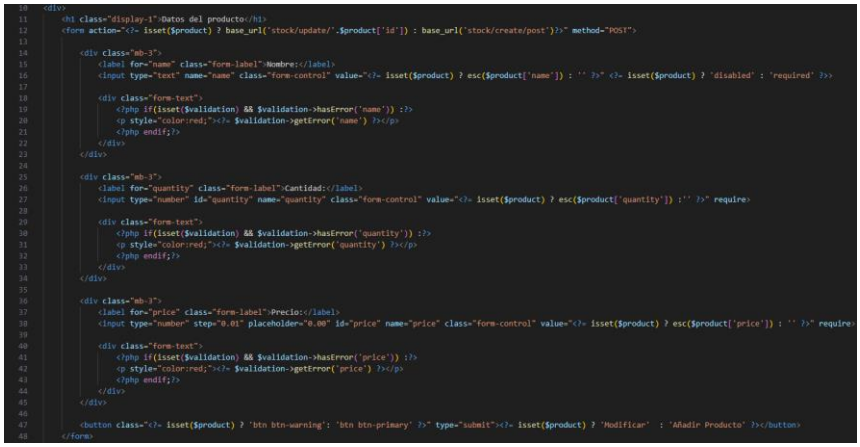

Ilustración 6- Formulario creación o modificación

El controlador “StockController.php” contiene 6 funciones para realizar las acciones de listar productos, crear, borrar modificar y visualizar los datos de un producto. Todas estas funciones crean un objeto del modelo stock que se explicará más abajo.

- Index: recoge todos los productos y retorna a la vista del index con los datos.

```
8       public function index()
9       {
10          $stock = new Stock();
11          //$data['products'] = $stock->findAll();
12          $data['products'] = $stock->getAllStock();
13          return view('/stock_index',$data);
14      }
```

Ilustración 7- Index

- View: busca un producto por identificador y retorna a la vista view con los datos del producto.

```
16      public function view($id)
17      {
18          $stock = new Stock();
19          //$data['product'] = $stock->where('id',$id)->first();
20          $data['product'] = $stock->getOneStock($id);
21          return view('stock_view',$data);
22      }
```

Ilustración 8- View

- Create: lleva a la vista de crear un producto.
- Créate_post: recoge y valida los datos para crear un producto, si no son validos retornará un array con los mensajes de error cometidos al introducir los valores. Si son validos se guardará y retornara mensaje de éxito.

```
24      {
25          return view('stock_create');
26      }
27
28      public function create_post()
29      {
30          $stock = new Stock();
31          $data = $this->request->getPost(['name', 'quantity', 'price']);
32
33          $data['price'] = floatval($data['price']);
34
35          if(!$this->validate($stock->getValidationRules(),$stock->getValidationMessages()))
36          {
37              return view('stock_create',['validation'=>$this->validator]);
38          }
39          else
40          {
41              $stock->save($data);
42              return redirect()->to('/stock')->with('success', 'Producto añadido correctamente');
43          }
44
45          return view('stock_create');
46      }
```

Ilustración 9- create, create_post

- Remove: eliminará un producto por su identificador.

```php
public function remove($id)
{
    $stock = new Stock();
    //$stock->delete($id);
    if($stock->removeOne($id))
    {
        return redirect()->to('/stock')->with('success', 'Producto eliminado');
    }
    else
    {
        return redirect()->to('/stock')->with('error', 'Error al eliminar el producto');
    }

}
```

Ilustración 10- remove

- Edit: busca el producto a modificar por su identificador y lo carga en la vista de crear.
- Update: actualizará los datos si son validos retornando un mensaje de éxito y redirigiendo al index. De lo contrarió redigirá al index con un mensaje de error.

```php
public function edit ($id)
{
    $stock = new Stock();
    //$data['product'] = $stock->where('id',$id)->first();
    $data['product'] = $stock->getOneStock($id);
    return view('stock_create',$data);
}

public function update($id) {
    $stock = new Stock();
    $data = $this->request->getPost(['quantity', 'price']);

    if ($stock->update($id, $data)) {
        return redirect()->to('/stock')->with('success', 'Producto modificado correctamente');
    } else {
        return redirect()->to('/stock')->with('error', 'Error de modificacion');
    }
}
```

Ilustración 11- edit, update

La base de datos se gestiono con una migración que creó la tabla de stock.

```php
public function up()
{
    $this->forge->addField([
        'id'=>[
            'type'=>'INT',
            'auto_increment'=>TRUE,
            'constraint'=>5, // Esto significa solo numeros de 5 digitos
            'null'=>FALSE
        ],
        'name'=>[
            'type'=>'VARCHAR',
            'constraint'=>'250',
            'null'=>FALSE
        ],
        'quantity'=>[
            'type'=>'INT',
            'null'=>FALSE
        ],
        'price'=>[
            'type'=>'DECIMAL',
            'constraint' => '10,2',
            'null'=>FALSE
        ]
    ]);
    $this->forge->addKey('id', true);
    $this->forge->createTable('stock');
}

public function down()
{
    $this->forge->dropTable('stock');
}
```

Ilustración 12- Migración tabla de stock

Para finalizar el documento se mostrará las partes mas importantes del modelo Stock.

Los atributos del modelo.

```php
    protected $table            = 'stock';
    protected $primaryKey       = 'id';
    protected $useAutoIncrement = true;
    protected $returnType       = 'array';
    protected $useSoftDeletes   = false;
    protected $protectFields    = true;
    protected $allowedFields    = [
        'name',
        'quantity',
        'price'
    ];
```

Ilustración 13- Atributos Stock

Las validaciones que deben de cumplir los datos del modelo Stock.

```
30      protected $validationRules      = [
31          'name'=>'required|regex_match[/^[a-zA-Z0-9\s]+$/]|is_unique[stock.name]',
32          'quantity'=>'required|integer|greater_than[0]',
33          'price'=>'required|decimal|greater_than[0]',
34      ];
```

Ilustración 14- Validaciones

Mensajes definidos para cada validación si fuera errónea.

```
35      protected $validationMessages    = [
36          'name'=>[
37              'required'=>'El nombre del producto es obligatorio',
38              'is_unique'=>'El nombre del producto esta repetido',
39              'regex_match' => 'El nombre no acepta caracteres especiales'
40          ],
41          'quantity'=>[
42              'required'=>'La cantidad de stock es obligatoria',
43              'greater_than'=>'La cantidad debe ser mayor que 0',
44              'integer'=>'La cantidad debe de ser un numero entero'
45          ],
46          'price'=>[
47              'required'=>'El precio es un campo obligatorio',
48              'greater_than'=>'El precio debe de ser mayor que 0.0',
49              'decimal'=>'El precio debe de ser un numero decimal'
50          ]
51      ];
```

Ilustración 15- Mensajes de error

Funciones del modelo stock para trabajar con la base de datos.

```
66      public function getAllStock()
67      {
68          return $this->findAll();
69      }
70
71      public function getOneStock($id)
72      {
73          return $this->where('id',$id)->first();
74      }
75      public function removeOne($id)
76      {
77          return $this->delete($id);
78      }
79      public function updateOne($id,$data)
80      {
81          return $this->set($id,$data);
82      }
```

Ilustración 16- Funciones Stock