



Instituto
de Ciencias
Tecnológicas

De la Universidad
San Sebastián

Unidad 1: Evaluación de la Unidad

Asignatura: Framework y Programación Web

Docente: Víctor Cofré Farías

Alumnos: Eliú Martínez Pincheira

Sección y Grupo: Sección 50

Fecha de Entrega: 13/11/2023

Caso a Resolver

Instrucciones:

Diseñar un software gestión de stock.

- Registrar un producto con los siguientes datos, id auto incrementable, código único del producto, nombre del producto, categoría, sucursal en la que se encuentra, y descripción, cantidad, y precio venta.
- Asignar productos a sucursal. (Defina 3 sucursales.)
- Consultar productos por código, nombre, y opcionalmente la sucursal.
- Dar de baja un producto. (Con la opción de eliminar el producto).
- Actualizar nombre, precio y descripción del producto.
- Crear un login para administrador del sistema.

Se solicita:

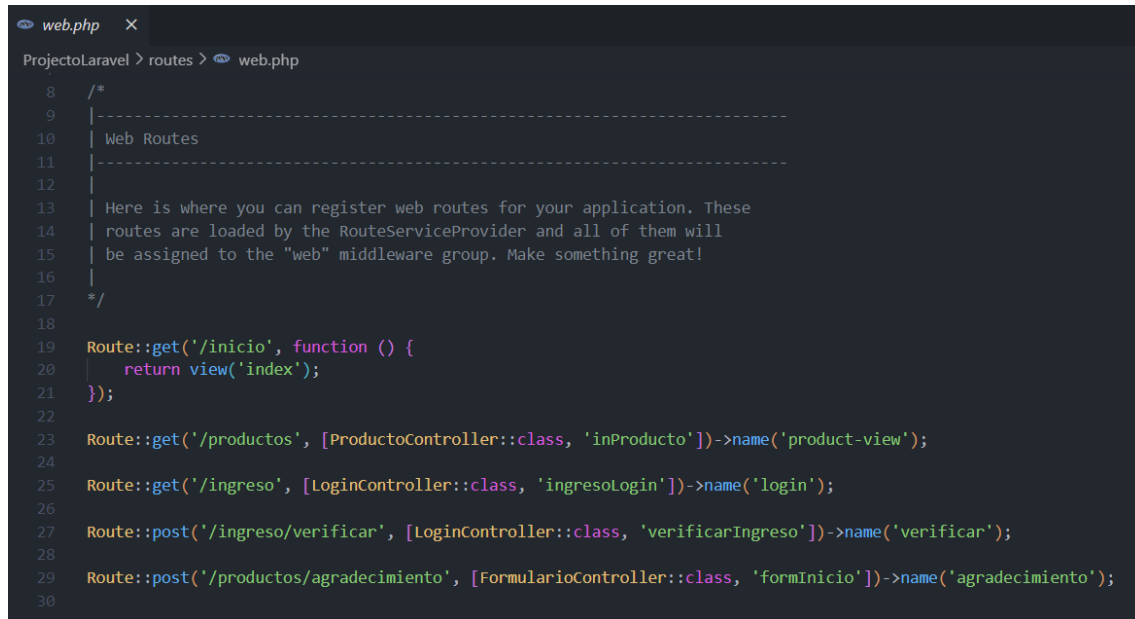
- Diseñe un estilo visual o utilice una plantilla Bootstrap que se ajuste a los requerimientos.
- Valide los campos obligatorios de las vistas y envíelos a una clase PHP la cual mostrará los datos por pantalla.
-

Formato de entrega :

- Adjunte en la plataforma virtual el proyecto e informe en una sola carpeta.
- Portada: título del trabajo, datos identificación (asignatura, logo de CIISA, nombre del profesor y de los alumnos).
- Desarrollo de los ítemes temáticos o instrucciones de la tarea.
- Archivo formato Word o PDF para evitar desconfiguración de tablas, imágenes o mapas.
- Número de páginas: Máximo 5.
- Tipo de letra: Arial, tamaño 12.
- Interlineado: 1,5 líneas
- Alineación: justificada.
- Páginas numeradas.
- El nombre a dar al archivo es el siguiente EV_Unidad1_Nombre_Apellido.zip

I. Continuación de Proyecto

Hemos revisado las vistas que, nos permitirán desarrollar el proyecto, ahora veremos las rutas que hemos creado y además los controladores que, nos permitirán dirigirnos a las vistas de manera más segura y recibiendo parámetros de formulario, para verificar que la información entregada, sea correcta.



```
web.php X
ProyectoLaravel > routes > web.php
8  /*
9  |-----
10 | Web Routes
11 |-----
12 |
13 | Here is where you can register web routes for your application. These
14 | routes are loaded by the RouteServiceProvider and all of them will
15 | be assigned to the "web" middleware group. Make something great!
16 |
17 */
18
19 Route::get('/inicio', function () {
20 |     return view('index');
21 });
22
23 Route::get('/productos', [ProductoController::class, 'inProducto'])->name('product-view');
24
25 Route::get('/ingreso', [LoginController::class, 'ingresoLogin'])->name('login');
26
27 Route::post('/ingreso/verificar', [LoginController::class, 'verificarIngreso'])->name('verificar');
28
29 Route::post('/productos/agradecimiento', [FormularioController::class, 'formInicio'])->name('agradecimiento');
30
```

Aquí podemos apreciar, las rutas creadas que se comunican con nuestras vistas, solo tenemos una ruta, que nos dirige directamente a una vista, las otras nos dirigen a **controladores**, y estos serán los que determinarán la llegada de la ruta.

II. Rutas & Controladores

```
Route::get('/productos', [ProductoController::class, 'inProducto'])->name('product-view');
```

```
class ProductoController extends Controller
{
    public function inProducto(){
        return view('productos');
    }
}
```

Ruta **/producto**, ésta lo único que realiza, es que la envía al controlador **ProductoController**, y luego, el método **inProducto()**, nos dirige a la página de productos.

```
Route::post('/productos/agradecimiento', [FormularioController::class, 'formInicio'])->name('agradecimiento');
```

```
class FormularioController extends Controller
{
    public function formInicio(Request $request) {

        $this->validate($request, [
            'nombre' => 'required|min:3',
            'edad' => 'required|integer|min:18|max:90',
            'mensaje' => 'required|min:10|max:50',
            'categoria' => 'required'
        ]);

        $nombre = $request->input('nombre');

        return view('agradecimiento')
            ->with('nombre', $nombre);
    }
}
```

Ahora veremos la ruta **/producto/agradecimiento**, esta se dirige al controlador **FormularioController** a su método **formInicio()**, como podemos ver, toma datos de un formulario, verifica por medio del **request** los valores, en caso de que sean correcto toma el dato del **input('nombre')** y lo devuelve a la vista **agradecimiento**.

```
Route::get('/ingreso', [LoginController::class, 'ingresoLogin'])->name('login');

Route::post('/ingreso/verificar', [LoginController::class, 'verificarIngreso'])->name('verificar');
```

```
class LoginController extends Controller
{
    public function ingresoLogin(){
        return view('ingreso');
    }

    public function verificarIngreso(Request $request){

        $userDB = "Rolando";
        $passwordDB = "12345";

        $this->validate($request, [
            "u-nombre" => "required",
            "u-contrasena" => "required"
        ]);

        $user = $request->input('u-nombre');
        $password = $request->input('u-contrasena');

        if($userDB == $user && $passwordDB == $password){
            return view('administrador')->with('user', $user);
        } else{
            return redirect()->route('login')->withErrors(['error' => 'Usuario o contraseña incorrecta']);
        }
    }
}
```

Estas rutas la veremos en más detalle, ya que realiza una verificación simulando los datos de un **BD**.

La ruta **/ingreso**, se dirige a la vista ingreso, en esta se encuentra el formulario del **login de administrador**, al enviar este formulario, se dirige a la ruta **/ingreso/verificar**, y a la vez esta se dirige al controlador de un método específico.

En este método, creamos las variables **\$userDB** y **\$passwordDB**, estas simulan credenciales de una base de datos.

Primero se valida que los campos no estén vacíos, si todo esta bien se guardan los campos ingresados y se evalúan que coincida con las credenciales de la base de datos.

En caso de que las credenciales sean correctas, se redirige a la vista **administrador**, en caso contrario, se redirige a la ruta del login del administrador, agregando un error y explicando que el usuario o la contraseña son incorrectos.

III. Conclusión

Se logra simular un inicio de sesión, también se validan campos entregados desde formularios a rutas, que a su vez las rutas se redireccionan a los controladores, para realizar todas las operaciones lógicas y de funcionamiento delicado, se adjunta proyecto para revisión en detalle y enlace de Git.

Enlace Git:

<https://github.com/gasdar/LaravelMyLogo>