

Tasca M14

INDICE:

Tasca M14	1
INDICE:	1
Problema:	1
Descripció	1
Nivell 1	1
Nivell 2	2
Solucion:	2
1.- Rutas del proyecto:	2
2.- Formularios:	3
Formularios de seguridad:	4
Shops/Galerias	5
Pictures/Cuadros	5
3 .- Laravel Test	6
Home:	7
Register:	7
Login:	7
Listar Shops:	7
Añadir shops:	8
Edit shop:	8
Borrar shop:	8
Listar cuadros:	8
LogOut:	8
Correciones:	8

Problema:

Descripció

En aquesta pràctica aprendras a crear una API completa. Hauras de crear una aplicació, per a una botiga de quadres, aplicant el patró de disseny de software MVC(Model-Vista-Controlador).

Nivell 1

- Exercici 1

Tenim una franquicia de botiga de quadres il·legals que fa veure que ven collarets blancs, per això es diu “white collar”.

Aquesta franquicia té varies botigues, cadascuna amb un nom i una capacitat màxima de quadres (o collars^^). Hi ha quadres que tenen un nom d'autor i d'altres que són anònims. Això sí, tots tenen un nom, un preu i una data d'entrada (és la data del moment en el que entra a la botiga). El client ens demana implementar una [API REST](#) amb Laravel. Aquesta API ha de cumplir les següents funcionalitats:

- Crear botiga: li direm el nom i la capacitat (POST /shops/).
- Llistar botigues: retorna la llista de botigues amb el seu nom i la capacitat (GET /shops/).
- Afegir quadre: li donarem el nom del quadre i el del autor (POST /shops/{ID}/pictures)
- Llistar els quadres de la botiga (GET /shops/{ID}/pictures).
- Incendiar quadres: per si ve la policia, es poden eliminar tots els quadres de la botiga sense deixar rastre (DELETE /shops/{ID}/pictures).

NOTES

Has de tindre en compte els següents detalls de construcció:

- Dissenya la base de dades com a primer pas. Utilitza ELOQUENT per accedir-hi.
- Inclou en un directori del projecte la col·lecció [Postman](#) per a les invocacions http
- Per a consultes a mysql, utilitzeu MysqlWorkbench. Si ho prefereixes, també pots utilitzar DataGrip, Dbeaver o altres.

Nivell 2

- Exercici 2

Afegeix el sistema de control d'accés de Laravel Passport. Defineix el login, registre i recuperació de contrasenya que l'usuari necessita per accedir a l'aplicació. Fes us de la configuració per defecte.

- Exercici 3

Defineix sistema de rols i bloqueja el accés a les diferents rutes segons el seu nivell de privilegis.

Nivell 3

- Exercici 4

Crea una aplicació frontal amb AJAX per a consumir les dades de les diferents rutes. Si vols pots fer-la per mitjà d'un framework: Angular, Vue, React...

- Exercici 5

Soluciona el CORS.

Solucion:

Hemos implementado este proyecto según las especificaciones de la tasca.

Adjuntamos una carpeta de “postman” con el fichero de los links de las pruebas realizadas en el back desde Postman.

1.- Rutas del proyecto:

Estas son las rutas que se han generado para el proyecto.

Method	URI	Name	Action	Middleware
GET	api/home	home.spa	App\Http\Controllers\WC ApiController@home	api
POST	api/register	PassportRegister	App\Http\Controllers\PassportController@register	api
POST	api/login	PassportLogin	App\Http\Controllers\PassportController@login	api
POST	api/logout	PassportLogout	App\Http\Controllers\PassportController@logout	auth:api
POST	api/shops	shops.create	App\Http\Controllers\WC ApiController@createShops	auth:api
GET	api/shops	shops.view	App\Http\Controllers\WC ApiController@viewShops	auth:api
POST	api/shops/delete/{id}	shops.delete	App\Http\Controllers\WC ApiController@deleteShops	auth:api
POST	api/shops/edit/{id}	shops.edit	App\Http\Controllers\WC ApiController@updateShops	auth:api
POST	api/shops/{id}/pictures	picture.create	App\Http\Controllers\WC ApiController@addPicture	
GET	api/shops/{id}/pictures	shop.pictureViewByShop	App\Http\Controllers\WC ApiController@viewPictureByShop	auth:api
DELETE	api/shops/{id}/pictures	shop.destroy	App\Http\Controllers\WC ApiController@destroy	auth:api
GET	api/show/picture	shop.pictureView	App\Http\Controllers\WC ApiController@viewPictures	auth:api
GET	api/pruebas	PassportPruebas	App\Http\Controllers\PassportController@pruebas	auth:api

2.- Formularios:

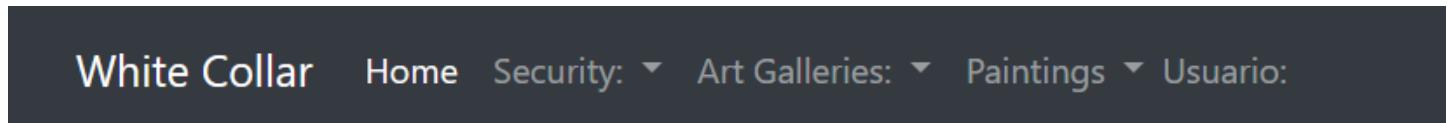
Se crean todos los formularios para realizar el proyecto **White Collar**:

Pagina de Home:

Página de home de nuestra aplicación.



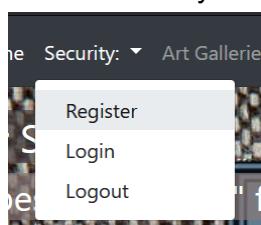
Menu de navbar:



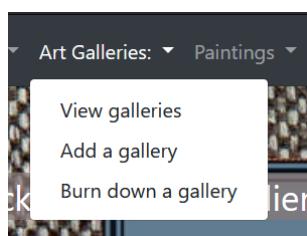
Menu superior:

Nos muestra las opciones de menu. De Izquierda a derecha:

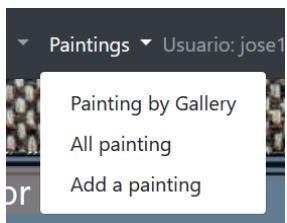
- Nombre de la aplicación “White Collar”,
- Botón “home”, para regresar al home,
- Menú “Security:” con las opciones: registrarse, login, y logout.



- Menú “Art Galleries:” con las opciones: View galleries, add a gallery, Burn down a gallery.



- Menú “Paintings:” con las opciones: painting by Gallery, All painting, Add a painting.



-Usuario:, nos muestra la información del usuario logado en la aplicación

Formularios de seguridad:

Register:

User name:

Email:

Password:

Submit

Formulario de registro:

Login:

Email:

Password:

Submit

Formulario de Login

Logout:

Push button to logout...

LOGOUT...

Vista de logout

Shops/Galerias

White Collar Home Security: Art Galleries: Paintings: = Usuario: Maria1 - .

#	Name:	Address:	Max capacity:	Detail:	Edit:	Delete:
1	Galería Helga de Alvear	Madrid, España	11	Detail	Edit	Delete
2	Galería Heinrich Ehrhardt	Madrid, España	13	Detail	Edit	Delete
3	Galería Joan Gaspar	Barcelona, España	11	Detail	Edit	Delete
4	Galería Casado Santapau	Madrid, España	11	Detail	Edit	Delete
5	Gagosian Gallery	Nueva York, Estados Unidos	10	Detail	Edit	Delete

Vista de la aplicación de listar todas las galerías. (cambiado!! añadido los botones detail, edit, delete)

White Collar Home Security: Art Galleries: Paintings: = Usuario: Maria1 - .

Create gallery:

Gallery name:

Address:

Max capacity: 0

[Submit](#)

Vista de crear una nueva galería shop. (cambiado!! solo a nivel estético)

White Collar Home Security: Art Galleries: Paintings: = Usuario: Maria1 - .

Select a Gallery to burn down: *(list select)

Seleccionado: [Burn Down...](#)

Vista de la acción de quemar la galería.

Pictures/Cuadros

Select a gallery to show pictures:

Select...

#	Name:	Address:	Action:
1	Galería Helga de Alvear	Madrid, España	Painting by Gallery
2	Galería Heinrich Ehrhardt	Madrid, España	Painting by Gallery
3	Galería Joan Gaspar	Barcelona, España	Painting by Gallery
4	Galería Casado Santapau	Madrid, España	Painting by Gallery
5	Gagosian Gallery	Nueva York, Estados Unidos	Painting by Gallery

(cambiado!! solo a nivel estético)

White Collar Home Security: Art Galleries: Paintings: = Usuario: Maria1 - .

Paining by gallery:

Gallery Number: 1

PICTURES:

Total picture number: 5



Acceso en 2 pasos para ver los cuadros de una galería. Primero seleccionamos la galería y nos aparecen los cuadros que tiene asignado esa galería.



Title:

La ultima cena

Author:

Leonardo da Vince

Quisquam asperiores doloribus asperiores
error laudantium delectus.

Price: 478

Entry date: 1979-07-21

Delete

id: 8 Gallery: 1

Estan son las cards de cada cuadro. Ahora las cards tienen la opción de borrado de un cuadro. (añadido!! botón delete)

A screenshot of a web application interface. At the top, there is a navigation bar with links for 'White Collar', 'Home', 'Security' (with a dropdown menu), 'Art Galleries' (with a dropdown menu), 'Paintings' (with a dropdown menu), and a user account for 'Maria1'. Below the navigation bar, there are two sections: 'All paintings:' and 'PICTURES:'. The 'All paintings:' section contains a heading and three small thumbnail images of different artworks. The 'PICTURES:' section has a heading and a note indicating a total of 28 pictures. Below these sections, there are three larger thumbnail images of artworks.

Vista general en la que vemos todos los cuadros y en la card podemos ver la informacion detallada. (cambiado!! solo a nivel estético)

A screenshot of a form titled 'Add a picture:'. The form includes fields for 'Author' (with a text input field), 'Picture name' (with a text input field), 'Select a existing Gallery: *(list select)' (with a dropdown menu), 'Price' (with a text input field), 'Entry date' (with a date input field), 'Select a Image url: *(list select)' (with a dropdown menu), 'Comment' (with a text input field), and a 'Submit' button at the bottom.

Vista para poder añadir un cuadro y asignarlo a una galeria. (cambiado!! solo a nivel estético)

3 .- Laravel Test

Hemos creado una serie de **test** para verificar que nuestro back funciona correctamente. Se puede verificar que la aplicación funciona bien sin necesidad de usar navegador o postman.

Se han creado los tests personalizados en el fichero `.\test\feature\WhiteCollarTest`. Hemo incluido todos los test solicitados, para comprobar las rutas y un CRUD completo.

Con el comando **php artisan test** pasamos los test a nuestra aplicación.

```
PASS Tests\Feature\WhiteCollarTest
✓ example
✓ home
✓ register
✓ login
✓ listar shops
✓ añadir shops
✓ edit shops
✓ listar pictures
✓ logout

Tests: 11 passed
Time: 0.74s
```

Podemos comprobar que pasa correctamente todos los test.

Analisis de los test generados:

Antes de empezar los test generamos unas variables estáticas para su posterior utilización:

```
// === Variables que utilizamos para cargar valores por defecto.
static $tokenLogin;
static $userName = "maria";
static $email = "@gmail.com";
static $password = "12345678";
```

La más necesaria es la variable estática **\$tokenLogin**, ya que necesitaremos reutilizar el token, que nos genera el srv, para utilizarlo como muestra de que estamos logados. Tambien mostramos información del test a medida que se pasa. Mostramos ciertas variables, como el token, el numero de elementos del array, y el json retornado, etc.

```
REGISTER NEW USER:
User generated: maria971
email generated: maria971@gmail.com
-----
LOGIN USER:
usuario: Maria1
email: maria1@gmail.com
password: 12345678 - Cript: $2y$10$RadP.4vf36Q3mtz1R415CeZ36WlcBjiEhhWv7LaeUcsz.noasrP2
Token de login generado srv.:
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIiNiJ9.eyJhdWQiOiIxLiwianRpIjoiNDIzMGFjMDU4MjQ3YzVjNmFiOTc5
WjYyYTE1NTlhMGQ1LCJpXQ1OjE2m2y50TA1Mjk0IjzMzhxLCJuYmY1OjE2MzY50TA1Mjk0IjzMz0LCj1eHai
n8vLBkRsPmWzTgxrlQohf_UETsJ3bgDXj7kfBAP1FylsgtpY6GwAiDyB0xa24V23-Uy5SknsBqA180DgNhsQ
I1YExygubp-E33cJmrnORor-n7v-d0Gm1juuETMiqixXLAuBZDtqx-09QNTLn-VtiqDVAuQjsfhRKOpJ5x1Jf
K3uugBPWmyvd1-qCZKCuPbPyg9ybclBK4ATcmGMTe-J9Ebf_3fkhrNlytdI07zBElvgCmCh_58UDL53FB7nAk4g
2gnXeuC98MyVizkr8fvX_jy/mZmcLhfF-aty_BAOsf-QERvAVzhAwIL95rcdq190IsABWq-8khyy4rV6cbn3dq5
hcvBE74f14tfw8ztQw7zRDK3YuLk/EC3cuswP9tq4SD8o3s3jb-B8nQ74brotk72MYn34
-----
ANADIR SHOP:
Name: Gallery Merl Gleichner
Address: 8719 Ryan Mill Apt. 459
Kuhlfurt, MD 98499
Max_capacity: 18
-----
EDIT SHOP:
Arr:
Num total de elementos:27
Pos rnd: 2
Valor a editar id:3
Id:3
ruta:/api/shops/edit/3
Data Old value:---
Name: Galeria Juana de Aizpuru_65
```

La imagen anterior es un ejemplo de la salida que tenemos al pasar los test.

Funciones de test generadas:

Home:

-El primer test **test_home** es una llamada simple a la página de home.

Register:

-En el test de **test_register** generamos un usuario y un email y lo registramos en el sistema. Guardamos estos datos para poderlo logar en los siguientes test.

Login:

-En el test **test_login** logamos al usuario con las credenciales de email y password. El password será igual para todos los test “12345678”. En este test obtenemos el token y lo guardamos en la variable estática **\$tokenLogin**, para reutilizarlo en los demás test. Obtenemos la info de login del usuario de la bbdd.

Listar Shops:

-En el test **test_listarShops** realizamos una petición de todas las shop y confirmamos que funcionan bien.

Añadir shops:

-En el test **test_añadirShops** generamos con faker un "name", una "address" , y un num rnd entre 15 y 30. Con estos datos realizamos el request de insercion de una nueva shop.

Edit shop:

-En el test **test_editShops**, obtenemos los id de las shops de la bbdd, generamos un número rnd con el num de registros obtenidos, y cargamos este registro para poder editarlo. Modificamos los campos obtenidos añadiendo una cadena _num a cada valor de txt, e incrementando en 1 el valor numérico. Por último lo añadimos en la bbdd. En el log podemos ver el valor "old" y el "new", para poderlos comparar.

Borrar shop:

-En el test **test_borrarShop** esta comentado, ya que borra un registro de la bbdd y solo lo utilizaremos, cuando se hayan realizado correctamente todo los otros test.

Listar cuadros:

-En el test **test_listarPictures** solamente realizamos la petición a la Api, y luego realizamos un **print_r** de json retornado para comprobar algunos valores.

LogOut:

-El ultimo test **test_logout** reliza el logout de la apicacion.

Tambien podemos verificar mediante la bbdd que todos nuestros test han realizado los cambios persistentes en la bbdd. Se ha añadido un registro, se ha actualizado, editado, y borrado.

Correciones:

He añadido indicaciones en cada punto que me comentabas, para detallar la corrección que he realizado.
Hola Jorge,

He revisado minuciosamente la última entrega, he de decir que esta bastante bien. Me gusta mucho que incursionaste con vue para poder tener los tres niveles de la tarea original y que además incluiste un manual con los componentes y secciones de tu aplicación. Gran trabajo!.

Ahora bien, he visto algunos detalles que quizá entre los más importantes estan los que tienen que ver con postman, las creaciones de registros y el gestor de roles, sin embargo, te dejo la anotación general de lo que encontré.

1.-*La redirección cuando creas una shop de la galería no esta funcionando correctamente

-Ya estaba puesta la redirección pero estaba mal escrito: route(mal) cambiado por router(bien). Se corrige y se realizan pruebas, y ya redirecciona correctamente, después de logarte.

2.-* Aparece un alert que imprime la ruta del backend a la hora de "quemar" una shop (quizá algo que se quedo mientras desarrollabas).

-He quitado los alert y los he cambiado por console.log, para verlo a nivel interno.

3.-* Una vez "quemada" una shop esta debería desaparecer de las opciones del desplegable. De hecho al llevar acabo esta acción el usuario no recibe feedback de si sea ejecuto correctamente o no la funcionalidad.

-He realizado cambios ahora el desplegable/select es dinámico y borra el elemento de la lista del select.

-El Json retornado, creía que era un array, pero resulta que es un objeto, por lo que he añadido un pequeño código que recorre todos los nombres de las propiedades, para obtener la posición/nombre del objeto, del id que te retorna la propiedad vue-selected. Por último elimino por su nombre la propiedad en vue dinamicamente, y se actualiza la lista des desplegable. He dejado que al borrar la galería solo se actualice la lista del select/list, para que se pueda, apreciar en la lista del desplegable. He dejado comentado por código de reenvio a otra pagina.

4.-* El usuario no debería de ver las opciones de registro o login si esta autenticado en el sistema.

* Si el usuario no esta autenticado quizá no debería de ver módulos como la creación de pinturas o galerias en la navegación.

-He cambiado el nav-bar. He creado 3 variables **lnk_reg**, **lnk_log**, **lnk_logOut**. Se ponen a true y false en función de si estas logeado o deslogado. Intenté utilizar la propiedad de Vue v-bind:disabled, pero no he conseguido que funcione. He leído en un foro que si utilizas bootstrap es necesario cambiar la clase. Ya que bootstrap tiene la propiedad class="disabled", entonces desde vue hay que utilizar la clase v-bind:class(propiedad : variable).

5.- * En la creación de una pintura, para seleccionar la shop, lo ideal sería mostrar en el desplegable las shops registradas en el sistema

-He añadido en el list select desplegable del que se puede seleccionar las galerías que ya están dadas de alta en el sistema.

6.-* ¿Puede ser que no este la colección de postman en la carpeta?

-En la carpeta de postman, estaba el pdf de documentación por error. He añadido el fichero de exportación de Postman.

7.-* No puedo editar o visualizar una shop ¿Necesito algún rol o permiso?

-He cambiado el formulario para listar las galerías y ahora puedes realizar 3 opciones: ver el detalles de la galería, editar la galería, y borrar la galería.

8.-* Probe los test y no me pasa el test de login con los datos que tienes ¿tengo que tener en cuenta algo? agradecería puedas probar porque al no realizar correctamente el login las siguientes pruebas fallan por no tener el token.

-Los test dependen del usuario con el que se realizan, ya que los test esperan un password genérico "12345678" y al cargar el usuario María1+maría1@gmail.com+1234578 ya funcionan correctamente.

Quedo al pendiente en caso de alguna duda.

Saludos!

He modificado la estética de algunos formularios. También he incluido la primera y la segunda versión de este pdf y la exportación de postman, para realizar comparaciones. Ante cualquier duda o problema que dudes en consultarme.