

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: | 7 |
| Edit shop: | 7 |
| Borrar shop: | 7 |
| Listar cuadros: | 7 |
| LogOut: | 7 |

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 tarea.

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:

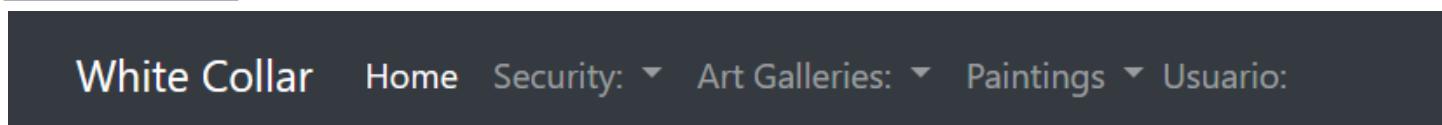
Se crean todos los formularios para realizar el proyecto **White Collar**:

Pagina de Home:

Página de home de nuestra aplicación.



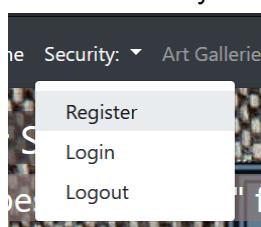
Menu de navbar:



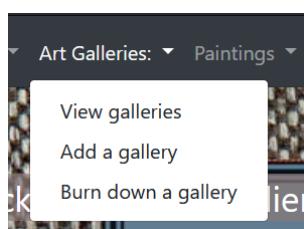
Menu superior:

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

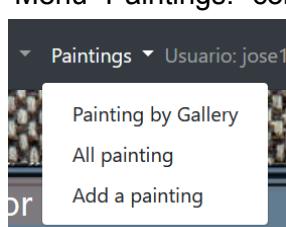
- Nombre de la aplicación “White Collar”,
- Botón “home”, para regresar al home,
- Menú “Security:” con las opciones:registerse, login, y logout.



-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

| Gallery list: | | | | | | |
|---------------|--------------------------|----------------------|---------------|---------|-------|---------|
| # | Name: | Address: | Max capacity: | Detail: | Edit: | Delete: |
| 1 | MoMA | Direccion | 22 | Show | Edit | Delete |
| 2 | 3 Punts Galería | Barcelona, España | 10 | Show | Edit | Delete |
| 3 | Galería Juana de Aizpuru | Madrid, España_65_32 | 12 | Show | Edit | Delete |
| 4 | Opera Gallery | Paris, Francia_12 | 11 | Show | Edit | Delete |
| 5 | Galería Joan Gaspar | Barcelona, España | 12 | Show | Edit | Delete |

Vista de la aplicacion de listar todas las galerias.

White Collar Home Security ▾ Art Galleries ▾ Paintings ▾ Usuario: Maria1 ▾

Create shop

Gallery name:

Address:

Max capacity:

0

Vista de crear una nueva galeria shop

White Collar Home Security ▾ Art Galleries ▾ Paintings ▾ Usuario: Maria1 ▾

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

Seleccionado:

Vista de la accion de quemar la galeria.

Pictures/Cuadros

White Collar Home Security ▾ Art Galleries ▾ Paintings ▾ Usuario: Maria1 ▾

select a gallery

Select a gallery to show pictures

| # | Name: | Address: | Detail: |
|---|--------------------------|----------------------|--|
| 1 | MoMA | Direccion | <input type="button" value="Painting by Gallery"/> |
| 2 | 3 Punts Galería | Barcelona, España | <input type="button" value="Painting by Gallery"/> |
| 3 | Galería Juana de Aizpuru | Madrid, España_65_32 | <input type="button" value="Painting by Gallery"/> |
| 4 | Opera Gallery | Paris, Francia_12 | <input type="button" value="Painting by Gallery"/> |
| 5 | Galería Joan Gaspar | Barcelona, España | <input type="button" value="Painting by Gallery"/> |

White Collar Home Security ▾ Art Galleries ▾ Paintings ▾ Usuario: Maria1 ▾

Paining by gallery, Gallery Number: 3

PICTURES:

Total picture number: 10



Title:

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

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

all painting

PICTURES:

Total picture number: 25

Title: La creacion de Adan
Author: Miguel Angel

Title: The Starry Night
Author: Vincent van Gogh

Title: The Raft of the Medusa
Author: Théodore Géricault

Vista general en la que vemos todos los cuadros y en la card podemos ver la informacion detallada.

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

Create picture

Author:

Picture name:

Shop id:

Price:

Entry date:
dd / mm / aaaa

Select a Gallery to Image url: *(list select)
Piet Mondriaan - Composición en rojo, amarillo, marrón y negro

Comment:

Vista para poder añadir un cuadro y asignarlo a una galeria.

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.4vF36Q3mtzIR41Scz36YwLcBjiEhhW7LaeUcsz.noasrp2

Token de login generado srv.:
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdWQiOiIxIiwianRpIjoicNDIzMjM0MjQ3YzVjNMF10tC5
wVyyTE1NTHhKG01CjIpX010jE2h2z50TA1MjkuMjIzMzxtLCjUvMv10jE2Mz50TA1MjkuMjIzMz0LCj1ehAi
n8vLBKRspPMZTQrlq8ht_UET5j3Qgdxj7kFBAP1FylsgUpY6GwiDyxBlxa24V23-Uy5Krn5BqA1880gnHsQ
IIYDxygbp-E3cJmrmR0r-n7v-dGnm1jmufTM1g1XLAuBZD0tx-Q0NTLn-Vtiq0VaiQ0sJfHnrQ0Pj5x1Jf
K3udGBPwmygd1-qCZK0ubfbyg9ybcb1BKHATcmQMe-J9EBF_3fkkrlytdl07zPBElvgCmCh_58UDLS3fb7nAk4g
2gnKeuC98wVWizkr8FVx_jy7mZmcLHF-ay_BAOsf-QTRvAVzhAfw1l95rcdq190tASBkq-8khyy4rV6cbn3dq5
hcvE74f14tfw8ztQw7zRDIGYUK7EC3CuswP9Rq4SD8o3s3jb-B8nQ74brotK72Mn34

-----
ANADIR SHOP:
Name: Gallery Merl Gleichner
Address: 8719 Ryan Mill Apt. 459
Kuhlcfurt, 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.