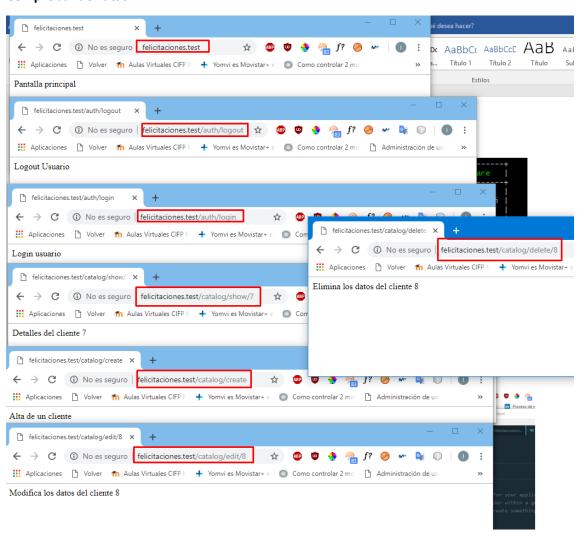
#### Ejercicio 1

Domain   Method	URI	Name	Action	Middleware
GET HEAD   GET HEAD	auth/logout   catalog   catalog/create   catalog/delete/{id}   catalog/edit/{id}		Closure Closure Closure Closure Closure Closure Closure Closure	web api,auth:api web web web web web web web web web
+++++++				

#### Comprobando rutas



#### Ejercicio 2

1. Crea una ruta que tenga un parámetro que sea opcional y comprueba que funciona.

```
← → C ① No es seguro | pruebas.test/user
Aplicaciones D Volver 🐧 Aulas Virtuales CIFP I 🛨 Yomvi es Movistar+ e 🔘 Como controlar 2 mc D Administración de us D Home D FuerteTienda > Panel 🗓 Proceso de n
                                  @ web.php — pruebas_rutas\routes — C:\Users\ivan\Google Drive\DAW\SEGUNDO DE DAW\Desarrollo web en entorno servidor\Tema 8\a36 — Atom
Parámetro opcional
                                  <u>File Edit View Selection Find Packages Help Debugger Nuclide</u>
                                    = 192.168.10.10
                                           > in felicitaciones
                                           > iii proyects

▼ impruebas_rutas

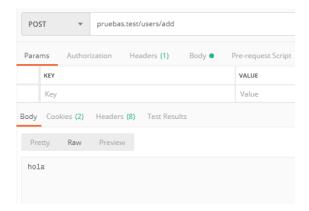
                                             > 🛅 app
                                              > iii bootstrap
                                              > 🛅 config
                                              > 🛅 database
                                              > 🛅 public
                                              > resources

▼ im routes
```

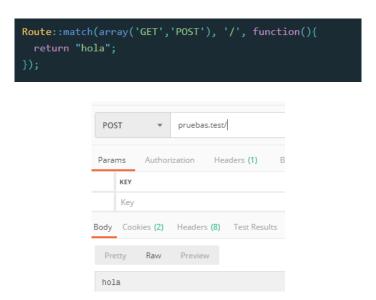
2. Crea una ruta que tenga un parámetro que sea opcional y tenga un valor por defecto en caso de que no se especifique.

 Crea una ruta que atienda por POST y compruébala con Postman. Si obtienes un error de tipo VerifyCsrfToken comenta la línea correspondiente en el fichero kernel.php (carpeta Http). Esto es un filtro para evitar ataques XSS (Cross-Site Scripting)

```
Route::post('/users/add', function(){
  return 'hola ';
});
```



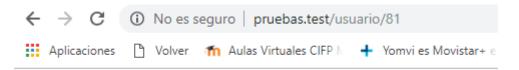
4. Crea una ruta que atienda por GET y por POST (en un único método) y compruébalas.



5. Crea una ruta que compruebe que un parámetro está formado sólo por números.

```
Route::get('usuario/{id}', function($id){
})->where('id', '[0-9]+');
```

## Prueba con parámetros numéricos



## Prueba con parámetros tipo texto.



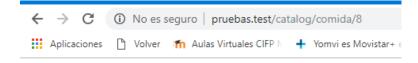
# 404

6. Crea una ruta con dos parámetros que compruebe que el primero está formado sólo por letras y el segundo sólo por números.

```
Route::get('catalog/{name}/{id}', function($id,$name){

})->where(array('name' => '[A-Za-z]+', 'id' => '[0-9]+'));
```

## Prueba con parámetros correctos



#### Prueba con un parámetro incorrecto



404

7. Ejecuta el siguiente comando y explica qué es lo que hace: artisan make:controller MiControlador

Crea la estructura para un controlador

```
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class MiControlador extends Controller
{
    //
}
```