

Se puede ver el funcionamiento de la web en <http://51.254.116.159:8004>  
(Los datos de sesión se encuentran en los comentarios de la práctica)

## Ejercicio 1. Añadir y editar clientes

Tenemos que añadir las rutas que nos van a hacer falta para recoger los datos al enviar los formularios. Para esto editamos el fichero de rutas y **añadimos** dos rutas (que deben estar protegidas por el filtro `auth`):

```
24
25     Route::get('/catalog/create', 'CatalogController@getCreate');
26     Route::post('/catalog/create', 'CatalogController@postCreate');
27
28     Route::get('/catalog/edit/{id}', 'CatalogController@getEdit');
29     Route::put('/catalog/edit/{id}', 'CatalogController@putEdit');
30
```

A continuación vamos a editar la vista `catalog/edit.blade.php` con los siguientes cambios:

- Tenemos que modificar los *inputs* para que en el formulario se carguen los valores correspondientes al cliente que vamos a editar. Por ejemplo, en el primer *input* tendríamos que añadir `value="{{ $cliente->nombre }}"`.

```
@csrf
<div class="form-group">
    <label for="title">Nombre</label>
    <input type="text" name="title" id="title" class="form-control" value="{{ $cliente->nombre }}">
</div>

<div class="form-group">
    <label for="url">Url de la imagen</label>
    <input type="url" name="url" class="form-control" value="{{ $cliente->imagen }}">
</div>

<div class="form-group">
    <label for="url">Fecha Nacimiento</label>
    <input type="text" name="fecha_nacimiento" class="form-control" value="<?php echo $newDate?>">
</div>
```

Por último tenemos que actualizar el controlador `CatalogController` con los dos nuevos métodos. En ambos casos tenemos que usar la [inyección de dependencias](#) para añadir la clase `Request` como parámetro de entrada (revisa la sección "[Datos de entrada](#)" de GitBook) que nos permitirá acceder a los datos del formulario:

```
38
39     public function postCreate(Request $request){
40         $c = new Cliente;
41         $c->nombre = $request->input('title');
42
43         if ($request->hasFile('file')) {
44
45             //obtenemos el campo file definido en el formulario
46             $fichero = $request->file('file');
47             $c->imagen = $fichero->getClientOriginalName();
48             $fichero->storeAs('img',$c->imagen);
49         }
50         $c->fecha_nacimiento = $request->input('fecha_nacimiento');
51         $c->correo = $request->input('email');
52         $c->save();
53
54         return redirect('/catalog');
55     }
56 }
```

## Realizamos la prueba

Crearemos a trinity

Añadir cliente

Nombre

Trinity

Url de la imagen

Seleccionar archivo

trinity.jpg

Fecha Nacimiento

04/07/2006

Email

trinity@gmail.es

Añadir cliente

Vemos como se ha creado.



Creamos el método putEdit()

```
public function putEdit(Request $request, $id){
    $c = Cliente::findOrFail($id);
    // $c = new Cliente;
    $c->nombre = $request->input('title');

    if ($request->hasFile('file')) {

        //obtenemos el campo file definido en el formulario
        $fichero = $request->file('file');
        $c->imagen = $fichero->getClientOriginalName();
        $fichero->storeAs('img', $c->imagen);
    }
    $c->fecha_nacimiento = $request->input('fecha_nacimiento');
    $c->correo = $request->input('email');
    $c->save();

    return redirect('/catalog');
}
```

## Modificamos a Trinity

pruebas.test/catalog/edit/14


Virtuales CIFP | + Yomvi es Movistar+ e Como controlar 2 mo Administración de us Home FuerteTienda > Panel

### Modificar cliente

Nombre

Trinityssss

Url de la imagen



Seleccionar archivo Ningún archivo seleccionado

Fecha Nacimiento

22/04/2006

Email

trinity@gmail.es

Modificar

## Consultamos a trinity

No es seguro | pruebas.test/catalog/show/14

Volver Aulas Virtuales CIFP + Yomvi es Movistar+ e Como controlar 2 mo Ad

Felicitaciones APP | Listado de clientes + Nuevo cliente



## Trinityssss

Correo-e: trinity@gmail.es  
Fecha de nacimiento: 2006-04-22

Editar Volver

## Ejercicio 2

Para finalizar nuestro prototipo de aplicación vamos a añadir un botón de borrar cuando estemos en la vista de detalle de un cliente.

Tendremos que:

1. Crear una nueva ruta,
2. Crear un nuevo método en el controlador
3. Actualizar el botón en la vista y redireccionar a la vista catálogo.

En la siguiente tabla se muestra un resumen de la ruta:

Ruta	Tipo	Controlador / Acción
/catalog/delete/{id}	DELETE	CatalogController@putDelete

En el método `putDelete` obtendremos el registro del cliente en cuestión y llamaremos al método `delete()`.

Tenemos que editar la vista detalle de clientes para añadir el botón de borrado. Dado que el borrado se realiza usando una petición HTTP tipo DELETE no podemos poner un enlace normal (ya que sería tipo GET). Para solucionarlo tenemos que crear un formulario alrededor del botón y asignar al formulario el método correspondiente, por ejemplo:

```
<form action="{action('CatalogController@putDelete', $cliente->id)}"
    method="POST" style="display:inline">
    {{ method_field('DELETE') }}
    {!! csrf_field() !!}
    <button type="submit" class="btn btn-danger" style="display:inline">
        Borrar
    </button>
</form>
```

### Creamos el método putDelete

Ojo! para usar `Storage::delete` hay que añadir : use `Illuminate\Support\Facades\Storage`;

```
public function putDelete($id){
    $cliente = Cliente::find($id);
    $img = $cliente["imagen"];
    Storage::delete('img/'.$img);
    $cliente->delete();

    return redirect('/catalog');
}
```