

UT4 – Frameworks para el desarrollo de aplicaciones web

Laravel



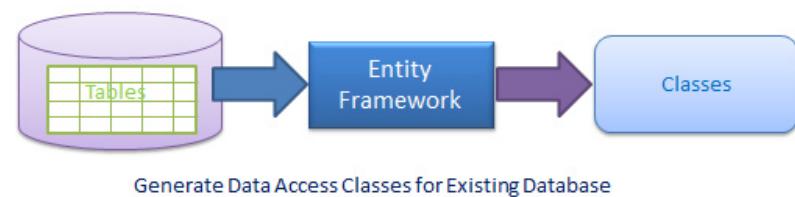
<https://laravel.com/>

Frameworks

- Herramienta que permite **simplificar el desarrollo de aplicaciones**, reduciendo el trabajo necesario.
- Ofrece una **arquitectura que sirve de base** para comenzar la aplicación a desarrollar.
- Proporciona al programador **funciones útiles** para realizar las tareas más comunes del desarrollo:
 - Plantillas para la presentación (vistas/forms)
 - Gestión de formularios y validación
 - Cookies y sesiones
 - Acceso a BBDD, ORM
- Están disponibles para la mayoría de los lenguajes de programación.



Un marco para comenzar el desarrollo

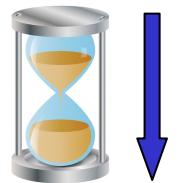


Frameworks

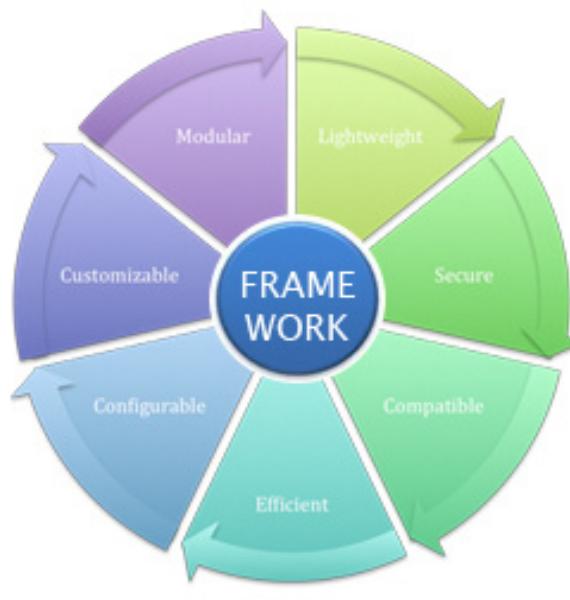
- El framework sirve de **estructura de soporte** para la aplicación a desarrollar.
- Ofrece **funcionalidades genéricas** (generación de código automatizado) que el programador puede personalizar con sus propias líneas de código → **facilita y reduce el desarrollo.**



Los frameworks evitan escribir aplicaciones desde cero



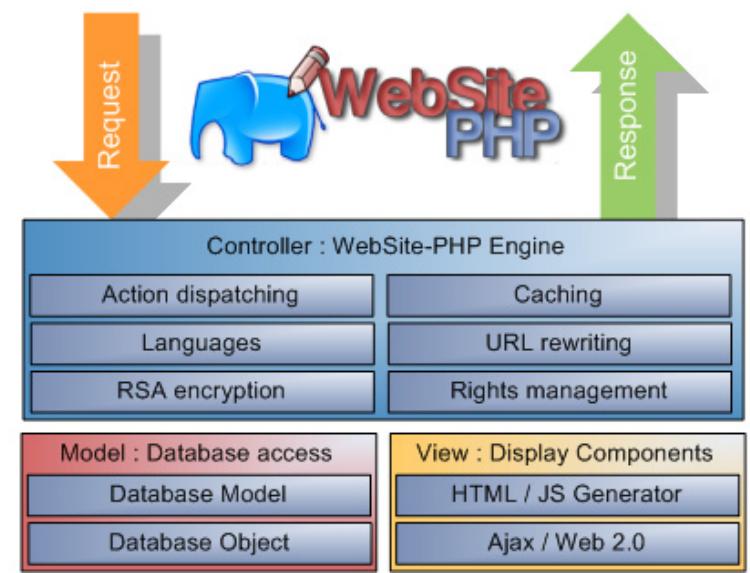
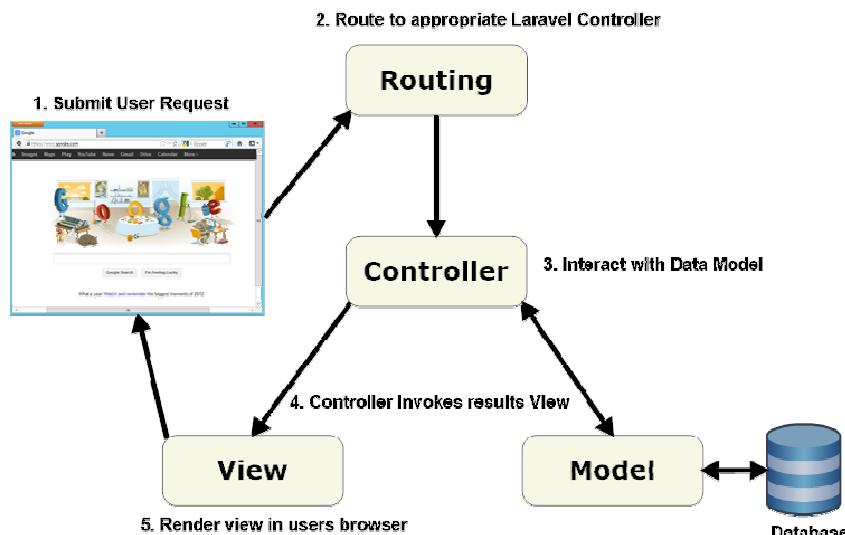
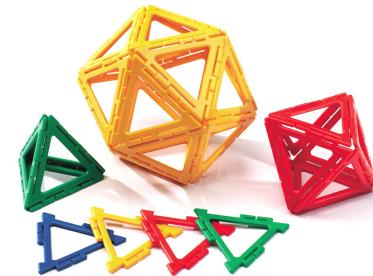
CMS vs Frameworks:



Todos queremos escribir menos código

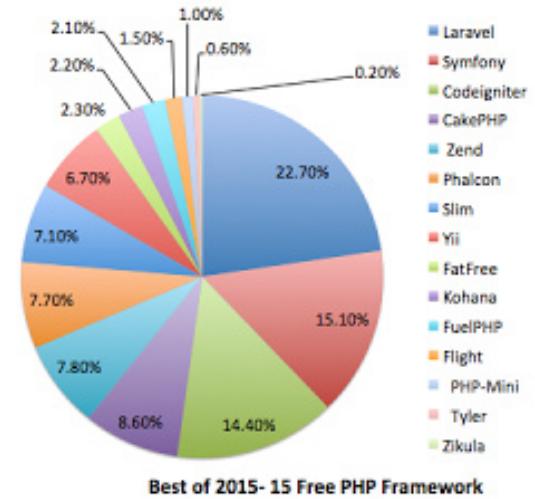
Frameworks

- Favorecen el **diseño modular**.
- Facilitan la **reutilización de código**.
- Reducen el **tiempo de desarrollo**.
- Emplean patrones de diseño (Poo) **MVC**:



Frameworks

- Existen numerosos frameworks para el **desarrollo de aplicaciones web** que usan el lenguaje de lado servidor **PHP**:

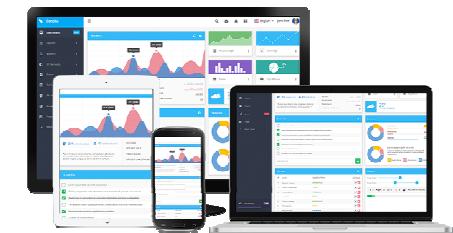


¿Cuál es el mejor framework?

<http://hotframeworks.com/>

<https://opensource.com/business/16/6/which-php-framework-right-you>

- También hay frameworks disponibles para otros lenguajes:
django (python), Ruby on RAILS (Ruby), ASP.NET (C#),
HIBERNATE (Java), Bootstrap (JavaScript), Catalyst (Perl).



Laravel

- Es un **framework MVC** para el desarrollo de **aplicaciones web** dinámicas y seguras con el lenguaje PHP.

- Es open source, cross-platform y sencillo.

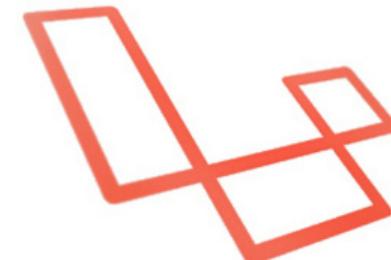
- Ofrece una **arquitectura base** para la aplicación a desarrollar.



- Cuenta con **herramientas** para generar código y acelerar el desarrollo:

- Plantillas	- Gestión de formularios y validaciones	- Gestión de cookies y sesiones
- ORM	- Autentificación	- AJAX
		- Subida de ficheros

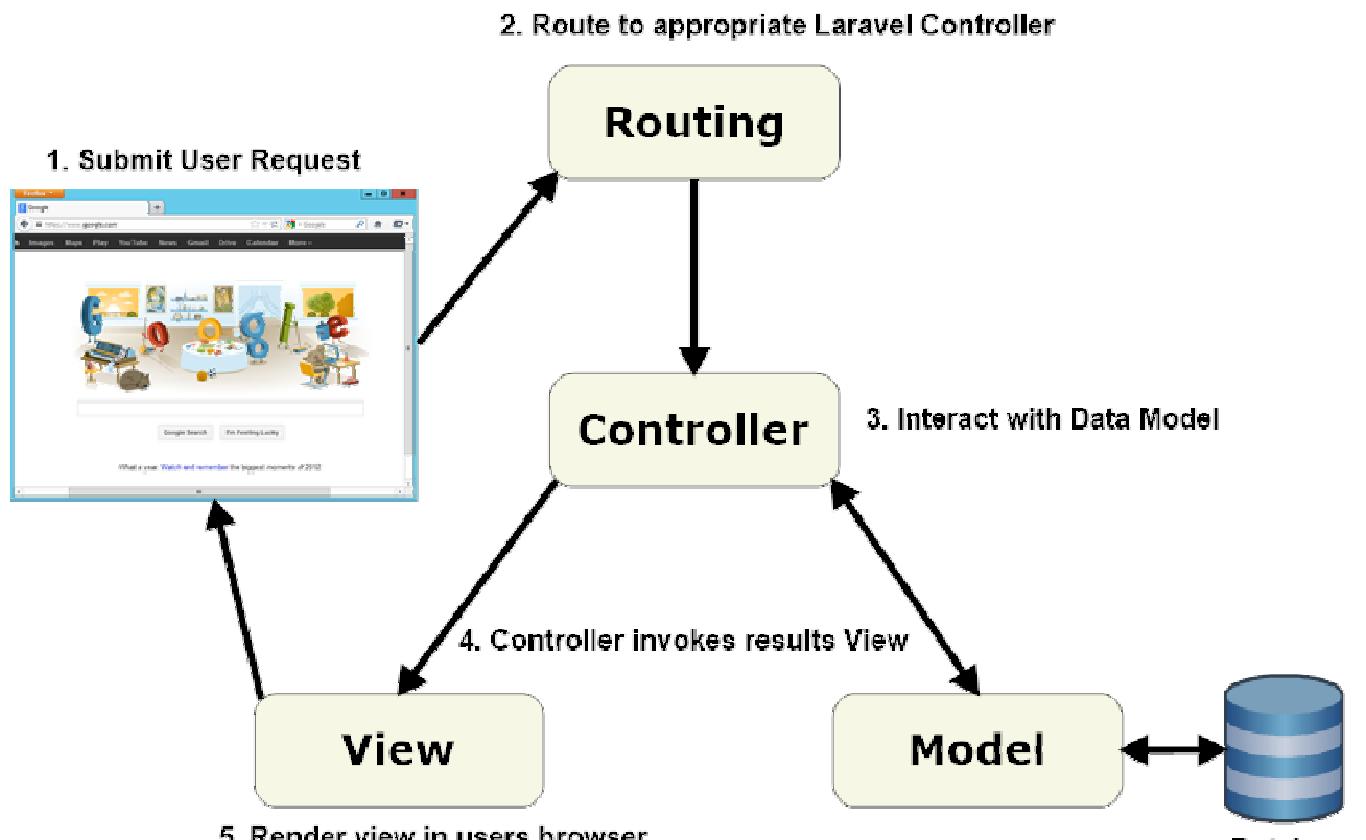
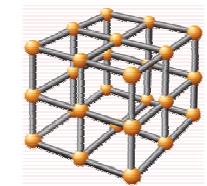
- Incorpora muchas de las mejores características de otros frameworks como CodeIgniter, Yii, ASP.NET, Ruby on RAILS o Symfony (puede usar sus componentes).



laravel

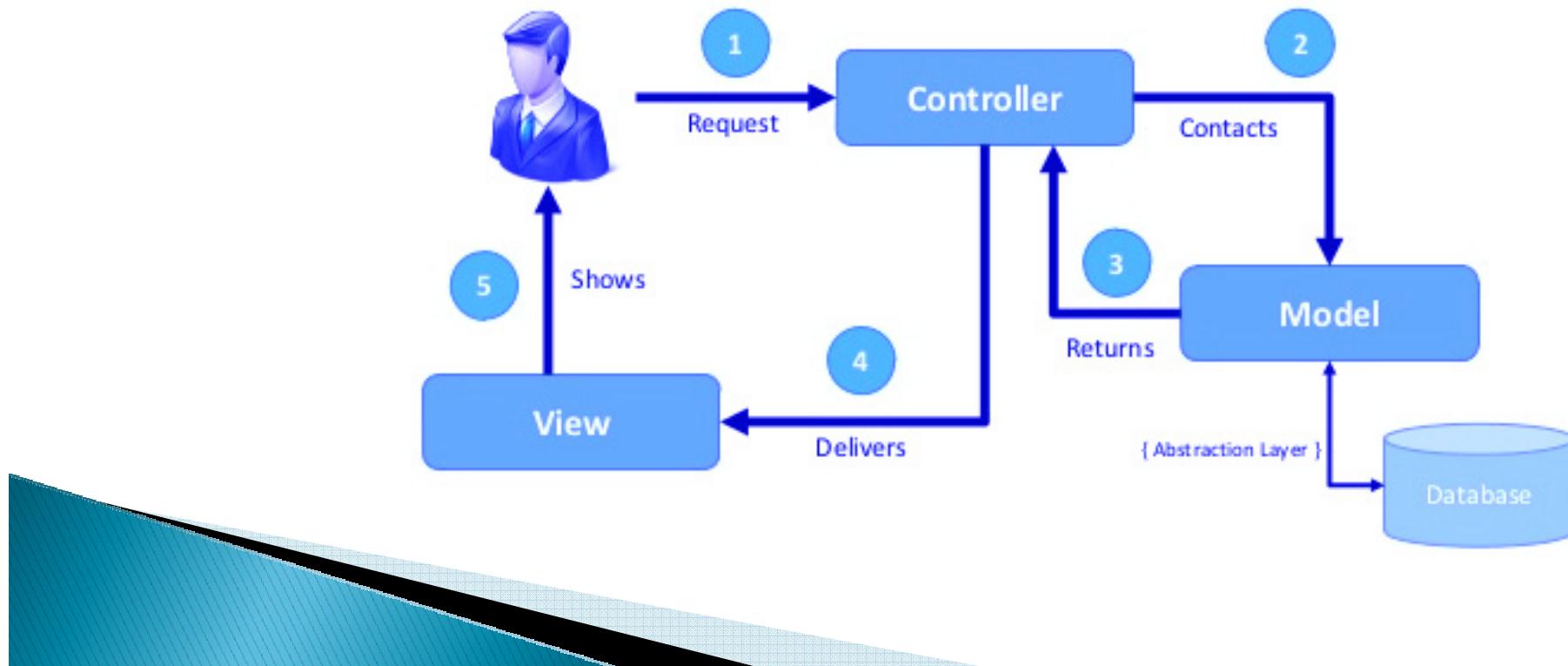
Laravel: arquitectura MVC

- Laravel crea aplicaciones web basadas en el **patrón de diseño arquitectónico MVC** (Modelo-Vista-Controlador):



Laravel: funcionamiento MVC

- El **controlador** es la clase cuyos métodos implementan la lógica de la aplicación.
- El **modelo** es el conjunto de datos (entidades) manejados por la aplicación: clases, ficheros, BBDDs.
- Las **vistas** son todas las representaciones de los datos que muestra la aplicación: plantillas para mostrar resultados, formularios de entrada, informes.



Laravel: Instalación del framework

1º Descargar Composer: el gestor de librerías y de paquetes para aplicaciones PHP en modo comando (parecido al apt-get de Linux):

<https://getcomposer.org/download/>

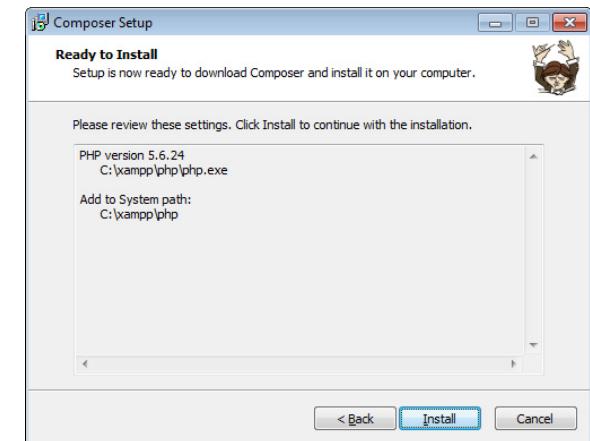
2º Instalar Composer. Pasos:

<https://getcomposer.org/doc/00-intro.md>

3º Probar la instalación en el terminal:

C:\>composer -v

Composer y Laravel están disponibles para desarrollar aplicaciones web (multiplataforma) desde un equipo Windows o Linux



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7600]
Copyright © 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\Juan>composer -v
[Composer logo]
Composer version 1.3.1 2017-01-07 18:08:51
Usage:
  command [options] [arguments]
Options:
  -h, --help                                Display this help message
  -q, --quiet                               Do not output any message
  -V, --version                             Display this application version
  --ansi                                  Force ANSI output
  --no-ansi                                Disable ANSI output
  --no-interaction                         Do not ask any interactive question
  --profile                                Display timing and memory usage information
  --no-plugins                            Whether to disable plugins.
  -d, --working-dir=WORKING-DIR           If specified, use the given directory as working directory
```

A screenshot of a Windows command prompt window titled 'C:\Windows\system32\cmd.exe'. The window shows the output of the command 'composer -v'. It includes the Composer logo, the version 'Composer version 1.3.1 2017-01-07 18:08:51', usage instructions, and a detailed list of command-line options with their descriptions.

Laravel: Instalación

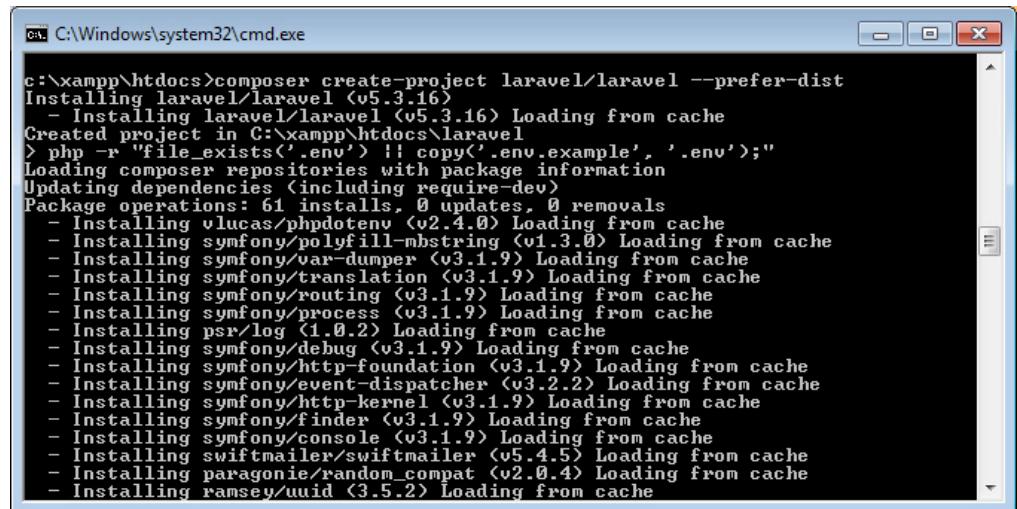
Para cada proyecto web que deseemos desarrollar, deberemos:

4º Descargar e Instalar mediante Composer el paquete con los archivos de Laravel en el directorio público del servidor web:

```
C:\cd c:\xampp\htdocs\  
C:\>composer create-project laravel/laravel --prefer-dist
```

Se creará el directorio *laravel* dentro de *htdocs*, que renombraremos con el nombre del proyecto (p.e. llamado *larawebapp*):

C:\xampp\htdocs\larawebapp



```
c:\Windows\system32\cmd.exe  
c:\xampp\htdocs>composer create-project laravel/laravel --prefer-dist  
Installing laravel/laravel (v5.3.16)  
- Installing laravel/laravel (v5.3.16) Loading from cache  
Created project in C:\xampp\htdocs\laravel  
> php -r "file_exists('.env') !! copy('.env.example', '.env');"  
Loading composer repositories with package information  
Updating dependencies (<including require-dev>  
Package operations: 61 installs, 0 updates, 0 removals  
- Installing vlucas/phpdotenv (v2.4.0) Loading from cache  
- Installing symfony/polyfill-mbstring (v1.3.0) Loading from cache  
- Installing symfony/var-dumper (v3.1.9) Loading from cache  
- Installing symfony/translation (v3.1.9) Loading from cache  
- Installing symfony/routing (v3.1.9) Loading from cache  
- Installing symfony/process (v3.1.9) Loading from cache  
- Installing psr/log (1.0.2) Loading from cache  
- Installing symfony/debug (v3.1.9) Loading from cache  
- Installing symfony/http-foundation (v3.1.9) Loading from cache  
- Installing symfony/event-dispatcher (v3.2.2) Loading from cache  
- Installing symfony/http-kernel (v3.1.9) Loading from cache  
- Installing symfony/finder (v3.1.9) Loading from cache  
- Installing symfony/console (v3.1.9) Loading from cache  
- Installing swiftmailer/swiftmailer (v5.4.5) Loading from cache  
- Installing paragonie/random_compat (v2.0.4) Loading from cache  
- Installing ramsey/uuid (3.5.2) Loading from cache
```

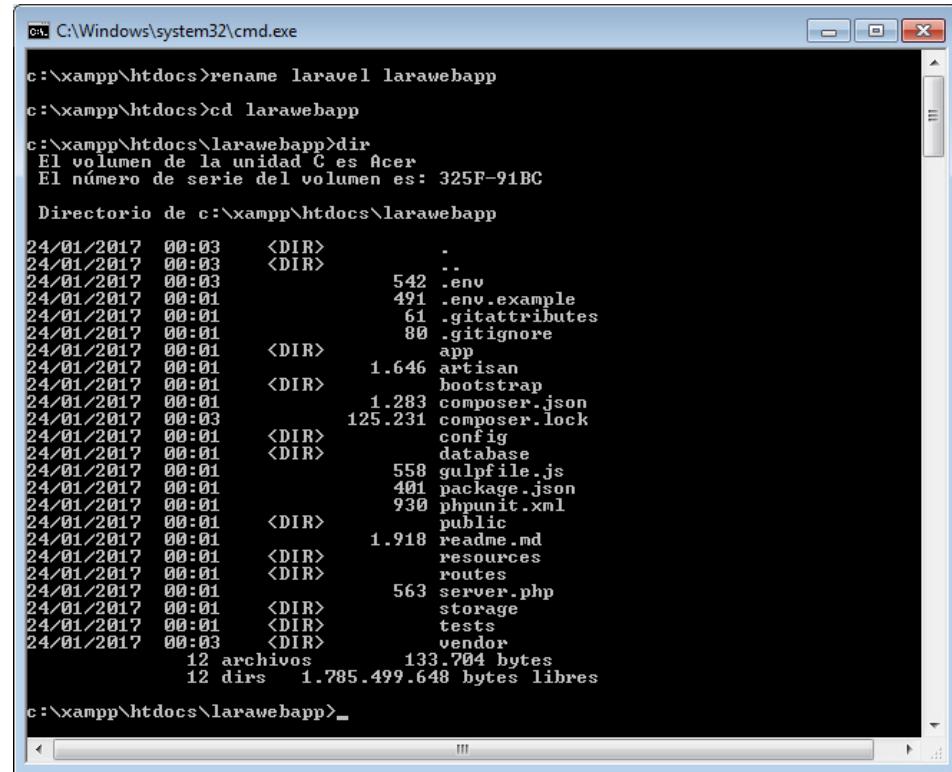
Laravel: Instalación

5º Listar los archivos creados por Laravel en el directorio del proyecto:

Dentro del subdirectorio **public** del proyecto (larawebapp) se encuentra el archivo **index.php** de la web en desarrollo.

6º Ejecutar el **servidor de desarrollo** mediante el intérprete de PHP:

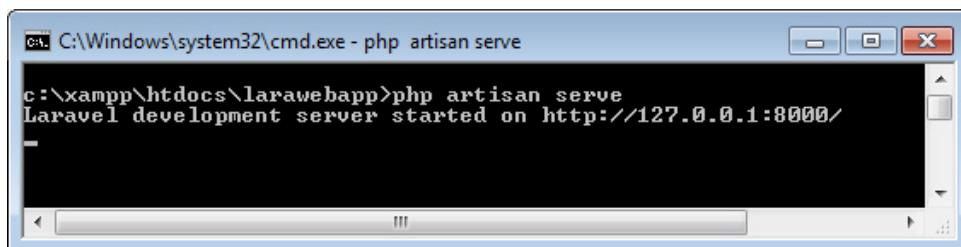
```
C:\xampp\htdocs\larawebapp>php artisan serve
```



```
c:\> C:\Windows\system32\cmd.exe
c:\> C:\xampp\htdocs>rename laravel larawebapp
c:\> C:\xampp\htdocs>cd larawebapp
c:\> C:\xampp\htdocs\larawebapp>dir
El volumen de la unidad C es Acer
El n mero de serie del volumen es: 325F-91BC
Directorio de c:\xampp\htdocs\larawebapp

24/01/2017  00:03    <DIR>      .
24/01/2017  00:03    <DIR>      ..
24/01/2017  00:03          542 .env
24/01/2017  00:01          491 .env.example
24/01/2017  00:01          61 .gitattributes
24/01/2017  00:01          80 .gitignore
24/01/2017  00:01    <DIR>      app
24/01/2017  00:01    <DIR>      1.646 artisan
24/01/2017  00:01    <DIR>      bootstrap
24/01/2017  00:01    <DIR>      1.283 composer.json
24/01/2017  00:03          125.231 composer.lock
24/01/2017  00:01    <DIR>      config
24/01/2017  00:01    <DIR>      database
24/01/2017  00:01          558 gulpfile.js
24/01/2017  00:01          401 package.json
24/01/2017  00:01          930 phpuunit.xml
24/01/2017  00:01    <DIR>      public
24/01/2017  00:01          1.918 readme.md
24/01/2017  00:01    <DIR>      resources
24/01/2017  00:01    <DIR>      routes
24/01/2017  00:01          563 server.php
24/01/2017  00:01    <DIR>      storage
24/01/2017  00:01    <DIR>      tests
24/01/2017  00:03    <DIR>      vendor
                           12 archivos       133.704 bytes
                           12 dirs        1.785.499.648 bytes libres

c:\> C:\xampp\htdocs\larawebapp>_
```

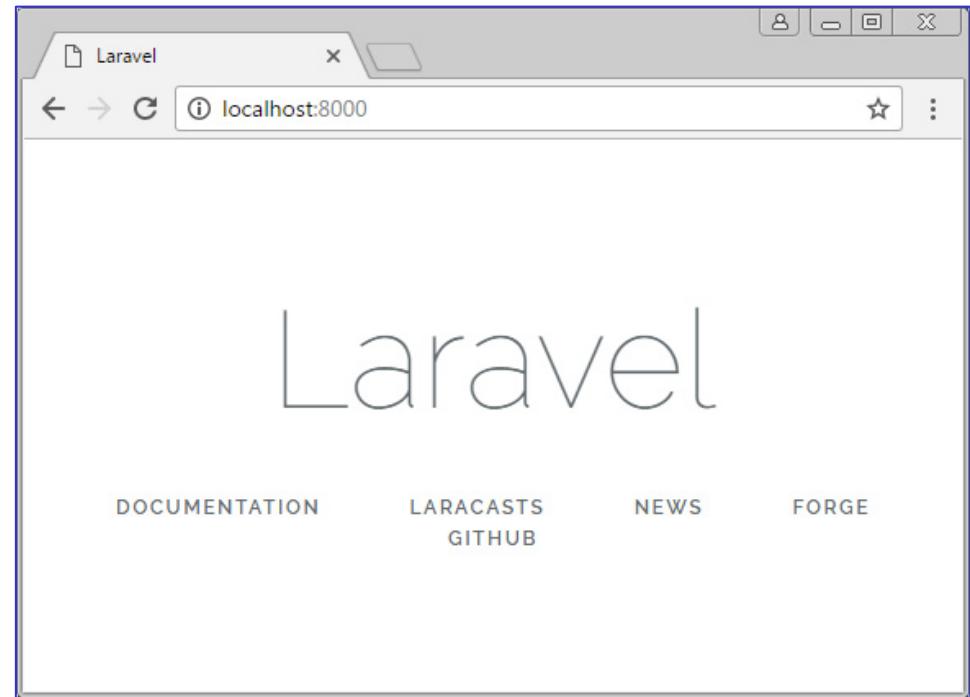


```
c:\> C:\Windows\system32\cmd.exe - php artisan serve
c:\> C:\xampp\htdocs\larawebapp>php artisan serve
Laravel development server started on http://127.0.0.1:8000/
```

Artisan es el CLI de Laravel

Laravel: Instalación

7º Visitar la URL que indica el servidor de desarrollo para visualizar la web que se ha creado y proseguir con el desarrollo:

A screenshot of a code editor window titled "server.php". The code is written in PHP and includes comments explaining its purpose. It handles URL decoding, Apache-style mod_rewrite emulation, and requires the index.php file from the public directory.

```
<?php  
/*  
 * Laravel - A PHP Framework For Web Artisans  
 *  
 * @package Laravel  
 * @author Taylor Otwell <taylor@laravel.com>  
 */  
  
$uri = urldecode(  
    parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH)  
);  
  
// This file allows us to emulate Apache's "mod_rewrite" functionality from the  
// built-in PHP web server. This provides a convenient way to test a Laravel  
// application without having installed a "real" web server software here.  
if ($uri !== '/' && file_exists(__DIR__.'/public'.$uri)) {  
    return false;  
}  
  
require_once __DIR__.'/public/index.php';
```

Laravel: Estructura de la aplicación web

Estructura de directorios de la aplicación web *base* creada:

app	Código de la aplicación
bootstrap	Script de configuración
config	Configuración de la aplicación
database	Bases de datos a usar por la aplicación
public	DocumentRoot de la aplicación. También contiene los assets (JS, CSS)
resources	Otros recursos de la aplicación (plantillas HTML)
storage	Archivos subidos al server, caché, logs
tests	Casos/test de prueba con PHPUnit
vendor	Dependencias de Composer
.env	
.env.example	
.gitattributes	
.gitignore	
artisan	
composer.json	
composer.lock	
gulpfile.js	
package.json	
phpspec.yml	
phpunit.xml	
readme.md	
server.php	

Contenido del subdirectorio **public**
(DocumentRoot):



<https://laravel.com/docs/5.3/structure>

Laravel: Configuración de la aplicación web

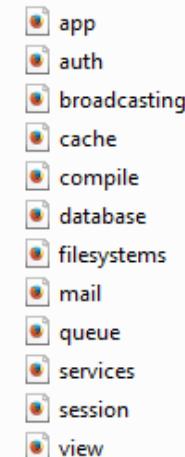
El subdirectorio **config** contiene los archivos de configuración de la aplicación: sesiones, bbdds, correo...

Configuración básica de la aplicación:

- Establecer permisos de escritura (chmod a+w) a los subdirectorios del proyecto: *storage* y *bootstrap/cache*
- Generar una clave para el envío cifrado de las variables de sesión:

```
C:\>php artisan key:generate
```

Que debe generar un archivo llamado **.env** en la carpeta del proyecto (laraweb).



```
C:\Windows\system32\cmd.exe
c:\xampp\htdocs\larawebapp>php artisan key:generate
Application key [base64:fZpaU58GoJGA59iEvF6NFeyuuyKx83xzayPank0vX70=] set successfully.

c:\xampp\htdocs\larawebapp>
```

Consulta el contenido del archivo .nev :
C:\>type .env

Laravel: Configuración de la aplicación web

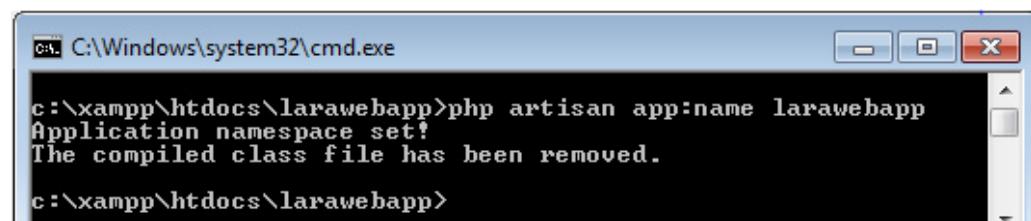
Configuración básica de la aplicación:

- Establecer la zona horaria en el archivo *config/app.php*
- Configurar los parámetros de conexión a la/s base/s de datos que usará la aplicación mediante el archivo:
config/database.php

Debemos indicar una BBDD por defecto.

- Indicar un **nombre** para la aplicación:

```
'connections' => [  
    'sqlite' => [  
        'driver' => 'sqlite',  
        'database' => env('DB_DATABASE', database_path('database.sqlite')),  
        'prefix' => '',  
    ],  
  
    'mysql' => [  
        'driver' => 'mysql',  
        'host' => env('DB_HOST', 'localhost'),  
        'port' => env('DB_PORT', '3306'),  
        'database' => env('DB_DATABASE', 'forge'),  
        'username' => env('DB_USERNAME', 'forge'),  
        'password' => env('DB_PASSWORD', ''),  
        'charset' => 'utf8',  
        'collation' => 'utf8_unicode_ci',  
        'prefix' => '',  
        'strict' => true,  
        'engine' => null,  
    ],
```

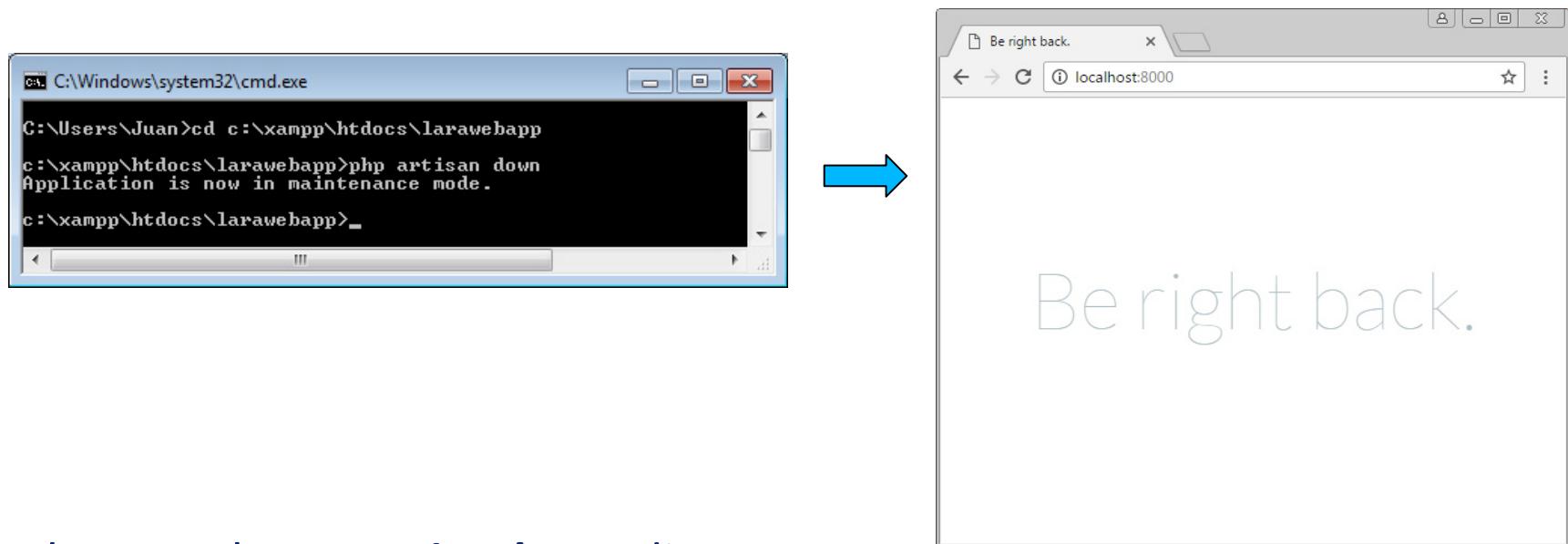


Laravel cuenta con APIs para conectar con BBDD de tipo: MySQL, PostgreSQL, SQLite y MS SQL Server.

Laravel: Mantenimiento de la aplicación web

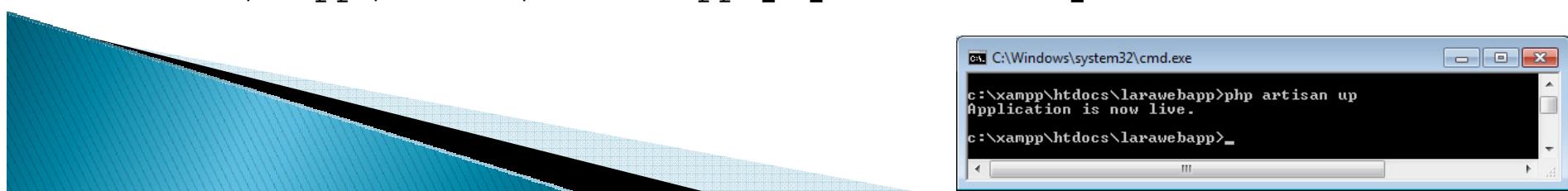
- Podemos **desactivar** (no despublicar) temporalmente la web durante el mantenimiento del código mediante:

```
C:\xampp\htdocs\larawebapp>php artisan down
```



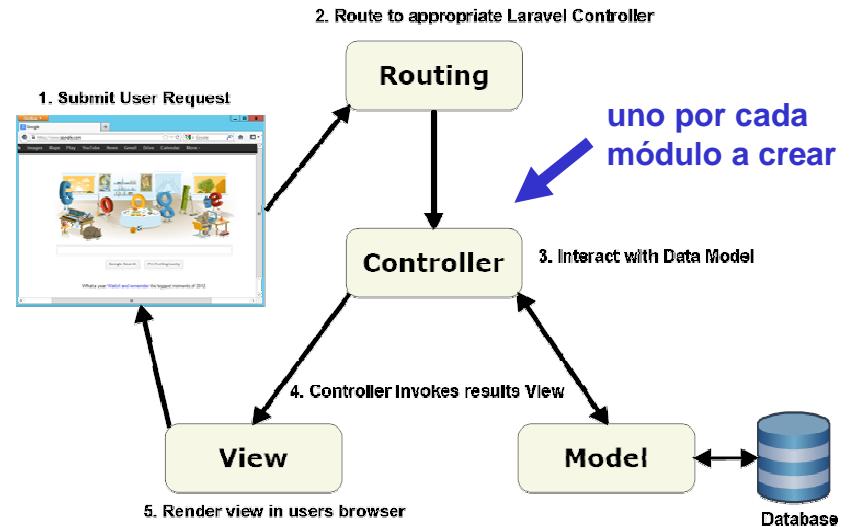
- Y luego volver a **activarla** mediante:

```
C:\xampp\htdocs\larawebapp>php artisan up
```



Laravel: Routing

- Laravel permite que la aplicación tenga **más de un Controlador** para modularizar la lógica del programa:
- El Routing es el mecanismo que permite a la aplicación "enrutar" cada petición del usuario hacia el **Controlador adecuado** → **invocar el método del controlador deseado.**
- Debemos crear **una ruta para cada Controlador o Módulo de la aplicación.**
- Las rutas a los distintos Controladores de la aplicación se definen en el archivo **routes/web.php** → **ESQUELETO DE LA APLICACIÓN**



A su vez, cada módulo tendrá asociado vistas distintas → en la aplicación habrá tantos patrones MVC implementados como módulos tenga.

Laravel: Routing

Para acceder a cada módulo de la aplicación empleamos en la URL la ruta creada hacia el controlador asociado a dicho módulo:

`http://www.larawebapp.com/nombreControlador`

- **Ejemplo:** archivo routes/web.php que define el esqueleto de una aplicación para la gestión de las notas de varias asignaturas:

The image shows the Laravel routes/web.php file on the left and four browser windows on the right.

routes/web.php:

```
<?php  
/*  
| Web Routes  
|  
| This file is where you may define all of the routes that are handled  
| by your application. Just tell Laravel the URIs it should respond  
| to using a Closure or controller method. Build something great!  
|  
|  
Route::get('/', function () {  
    echo "<h1>Página Principal</h1>";  
    // return view('welcome'); // Plantilla HTML de Blade ubicada en resources/views/welcome.blade  
});  
  
Route::get('/usuarios', function () { // URL: http://localhost:8000/usuarios  
    echo "<h1>Módulo usuarios</h1>";  
});  
  
Route::get('/asignaturas', function () { // URL: http://localhost:8000/asignaturas  
    echo "<h1>Módulo asignaturas</h1>";  
});  
  
Route::get('/notas', function () { // URL: http://localhost:8000/notas  
    echo "<h1>Módulo notas</h1>";  
});
```

Browser Windows (localhost:8000):

- Página Principal
- Módulo usuarios
- Módulo asignaturas
- Módulo notas

Route tb cuenta con el método post

Laravel: Routing

• Paso de parámetros al Controlador:

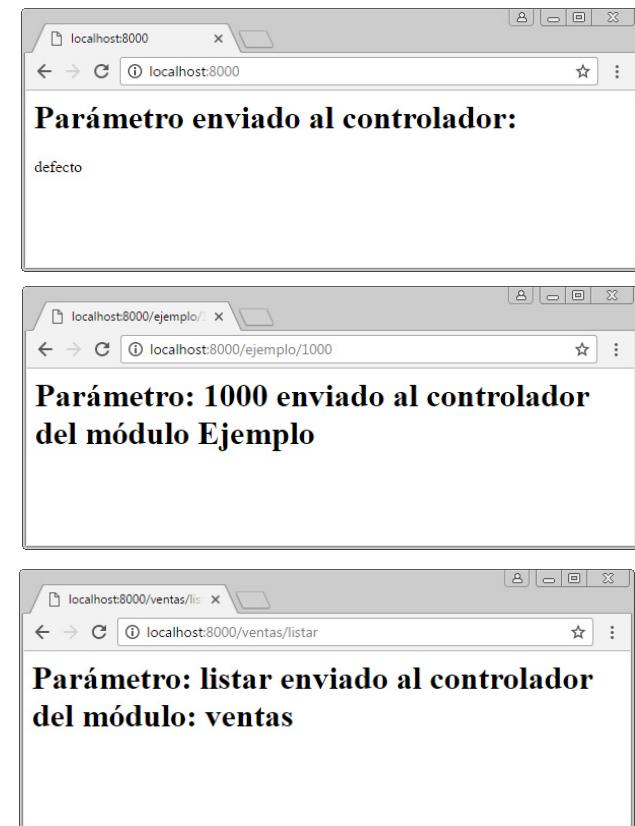
Un Controlador puede ser invocado pasándole parámetros a través de la ruta o URL (parecido al método GET):

Archivo routes/web.php

```
Route::get('/{parametro?}', function ($parametro = "defecto") {
    // URL: http://localhost:8000/valorDelParametro
    echo "<h1>Parámetro enviado al controlador:</h1>";
    echo $parametro;
});

Route::get('ejemplo/{parametro}', function ($parametro) {
    // URL: http://localhost:8000/ejemplo/valorDelParametro
    echo "<h1>Parámetro: ".$parametro." enviado al
controlador del módulo Ejemplo</h1>";
});

Route::get('{modulo}/{parametro}', function ($modulo, $parametro) {
    // URL: http://localhost:8000/nombreDelModulo/valorDelParametro
    echo "<h1>Parámetro: ".$parametro.
    " enviado al controlador del módulo: ".$modulo."</h1>";
});
```



Si añadimos ? al nombre de un parámetro y le asignamos un valor por defecto → se convierte en un **parámetro opcional**

Laravel: Routing

- **Paso de parámetros al Controlador:**

Podemos aplicar restricciones a los parámetros recibidos en la ruta.

Se usa el método *where* para aplicarle un patrón a la ruta solicitada a la aplicación:

```
Route::get('user/{name}', function ($name) {
    //
})->where('name', '[A-Za-z]+');

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

Route::get('user/{id}/{name}', function ($id, $name) {
    //
})->where(['id' => '[0-9]+', 'name' => '[a-z]+']);
```

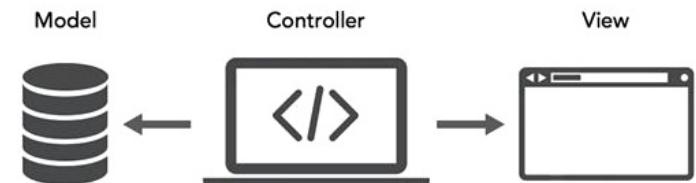
Las rutas que estarán disponibles a través de solicitudes realizadas por el método **GET** se especifican mediante métodos **Route::get()**

Las rutas que estarán disponibles a través de solicitudes por **POST** se especifican mediante métodos **Route::post()**

Laravel: MVC

- **Enlazar el Controlador con el Modelo/s y la Vista/s:**

El Controlador maneja los Modelos y las Vistas definidas en la aplicación. Puede asociar cualquier Modelo a la Vista apropiada:



Archivo **routes/web.php**

```
CONTROLADOR y PARÁMETROS           MODELO  
Route::get('ventas/{cliente}', function (App\Cliente $cliente) {  
    return view('fichaCliente', ['cliente' => $cliente]);  
});
```

VISTA y MODELO ASOCIADO

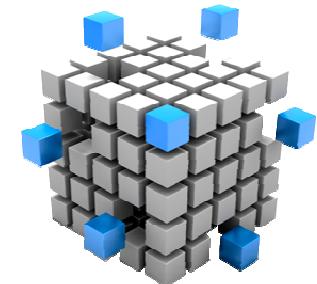
The code snippet shows a route definition in the `routes/web.php` file. It uses the `Route::get` method to map the URL `'ventas/{cliente}'` to a controller action. The controller action is defined using a closure, which returns a view named `'fichaCliente'` with a parameter `'cliente'` set to the value passed from the route. The code is annotated with boxes: 'CONTROLADOR y PARÁMETROS' covers the route and controller part, 'MODELO' covers the model part (the closure), and 'VISTA y MODELO ASOCIADO' covers the view and its association with the model.

El atributo **action** de los `<form>` de la aplicación apuntará hacia la **ruta del Controlador** creado para procesar dicho formulario:

```
<form method="get" action="/ventas">  
    ...  
</form>
```

Laravel: MVC

- **Ejemplo:** el controlador Ventas recibe como parámetro el nombre del Cliente y muestra su ficha mediante una vista.



Archivo **routes/web.php**

```
Route::get('ventas/{cliente}', function ($cliente) {
    return view('fichacliente', ['cliente' => $cliente]);
});
```

La **vista** se define creando en el directorio **resources/views** una plantilla Blade llamada **fichacliente** que acepta parámetros:

Archivo **resources/views/fichacliente.blade.php**

```
<!DOCTYPE html>
<html>
    <head> <title>Cliente</title> </head>
    <body>
        <p class="content"> Ficha del cliente: </p>
        <p> {{ $cliente }} </p>
    </body>
</html>
```



Los caracteres **{{ }}** permiten acceder al valor de los parámetros enviados a la plantilla Blade

Laravel: MVC

- **Actividad:** define en el archivo **routes/web.php** las rutas para los Controladores de los 3 módulos de una aplicación:
alumnos, asignaturas y notas.

Cada Controlador aceptará los siguientes parámetros al ser invocado mediante la URL solicitada:

alumnos/{nombre}
asignaturas/{nombre}
notas/{asignatura}/{alumno}/{nota}

Crea las vistas mediante 3 plantillas Blade en el directorio **resources/views** que reciban los parámetros recogidos por su Controlador y los muestre:

alumno.blade.php ← fondo rojo
asignatura.blade.php ← fondo verde
notas.blade.php ← fondo azul

Finalmente pruébalo visitando las siguientes URLs:

<http://localhost:8000/alumnos/ana>
<http://localhost:8000/asignaturas/DSW>
<http://localhost:8000/notas/DSW/ana/7>

Laravel: Middleware

- **Middleware:** *intermediario* entre una petición HTTP y su respuesta.
- Permite **filtrar** las peticiones que se soliciten a la aplicación al pasárselas al Controlador adecuado mediante Routing.
- Laravel incluye algunos middlewares para verificar si un usuario se ha autenticado, para cifrar las cookies, para evitar ataques CSRF.
- Para crear un middleware propio ejecutamos la orden:

```
php artisan make:middleware EjemploMiddleware
```

Que genera en un archivo `EjemploMiddleware.php` en el directorio:

`app/http/Middleware`

Con la siguiente estructura:

```
<?php
    namespace App\Http\Middleware;
    use Closure;
    class EjemploMiddleware {
        public function handle($request, Closure $next) {
            // LÓGICA
            return $next($request);
        }
    }
}
```

Laravel: Middleware

- Una vez creado hay que **Registrarlo** en el archivo **app/http/kernel.php**; añadiendo el nuevo middleware a alguno de los dos arrays:

\$middleware si queremos que el middleware se le aplique a cualquier ruta solicitada.

\$routeMiddleware si queremos que el middleware se ejecute solo para ciertas rutas que lo especifiquen.

...

```
protected $routeMiddleware = [
    'auth' => \Illuminate\Auth\Middleware\Authenticate::class,
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
    'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings::class,
    'can' => \Illuminate\Auth\Middleware\Authorize::class,
    'guest' => \larawebapp\Http\Middleware\RedirectIfAuthenticated::class,
    'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
    'EjemploMiddleware' => \larawebapp\Http\Middleware\EjemploMiddleware::class,
];
```

...

Laravel: Controladores

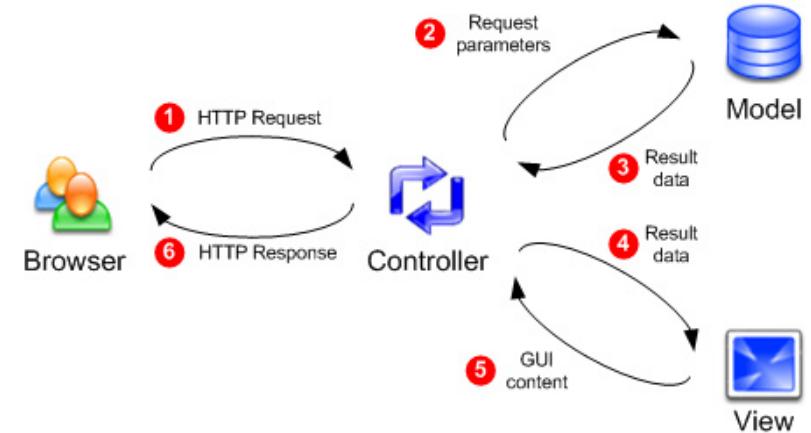
- Los controladores son quienes encapsulan la **lógica de la aplicación**.
- Para **crear un controlador** lanzamos:

```
php artisan make:controller <nombre>
```

que crea el archivo `<nombre>.php` en:

`app/Http/Controllers`

- El archivo creado contiene un **esqueleto** para la clase Controlador creada (que hereda de la clase Controller de Laravel).
- Debemos **implementar los métodos** necesarios para el controlador. Se suele implementar los métodos **CRUD** (Create, Read, Update & Delete).
- Cada método del controlador será **invocado a través de una ruta** en el archivo `routes/web.php`, pudiéndole pasar parámetros desde de la URL.



Laravel: Controladores

- **Ejemplo:** controlador con métodos para mostrar la ruta de la URL.

Ejecutar: `php artisan make:controller controlador`

Archivo app/Http/Controllers/controlador.php generado:

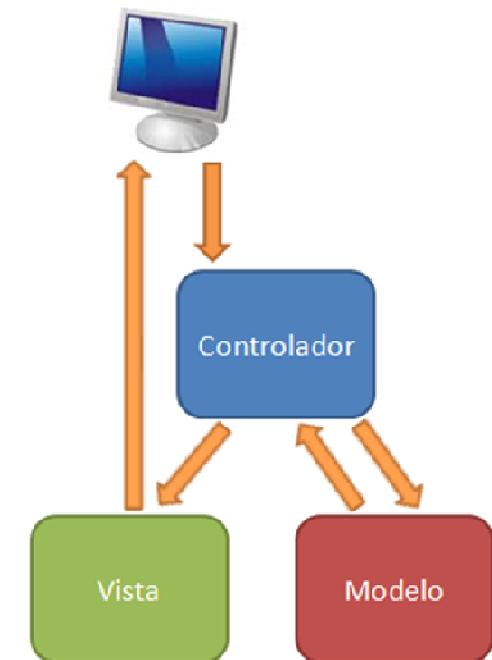
```
<?php
namespace larawebapp\Http\Controllers;
use Illuminate\Http\Request;

class controlador extends Controller {

    public function mostrarTodo (Request $request){
        echo '<br>MOSTRANDO EL CONTENIDO DE LA SOLICITUD:<br>';
        echo '<br>Request: '.$request;
        echo '<br>URL: '.$request->path();
        echo '<br>Method: ' . $request->method();
    }

    public function mostrarCliente ($n){
        return view("fichacliente", ['cliente' => $n]);
    }
}
```

↑
Es en los métodos del controlador donde
debemos llamar a las Vistas



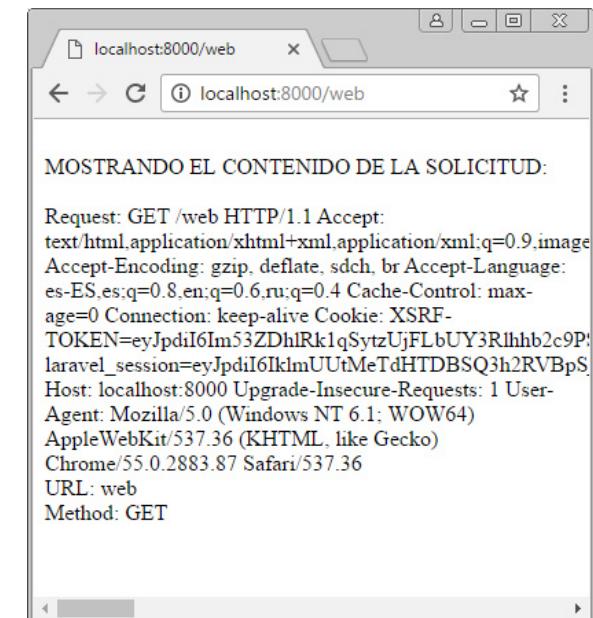
Laravel: Controladores

- Finalmente, hay que invocar a los distintos métodos (acciones) del controlador dependiendo de la ruta indicada en la URL:

['uses' => 'nombreControlador@método']

Archivo routes/web.php :

```
...          RUTA          CONTROLADOR y  
Route::get('/controlador', ['uses' => 'controlador@mostrarTodo']);  
// URL: http://localhost:8000/controlador  
  
Route::get('/controlador/cliente/{nombre}', ['uses' => 'controlador@mostrarCliente']);  
// URL: http://localhost:8000/controlador/cliente/<cualquiervalor>  
  
Route::get('/web', ['uses' => 'controlador@mostrarTodo']);  
// URL: http://localhost:8000/web  
  
Route::get('/ventas/{cliente}', function ($cliente) {  
    // URL: http://localhost:8000/ventas/nombreCliente  
    return view("fichacliente", ['cliente' => $cliente]);  
});  
...
```



Laravel: Ejemplo MVC

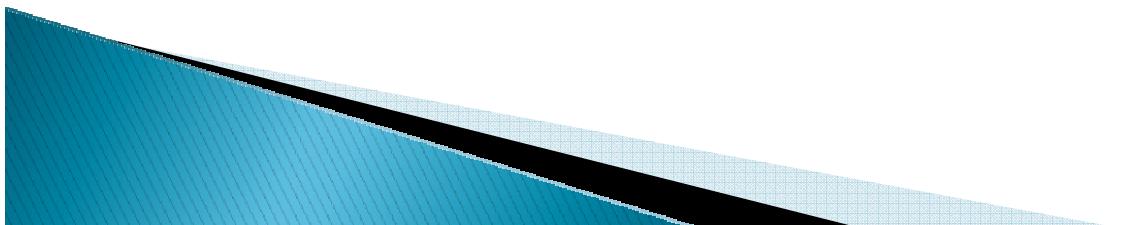
- **Actividad:** crear un controlador llamado *numeros*. El controlador deberá tener un método llamado *mayor* que acepte tres números y calcule cuál es el mayor de los tres.

Crear una ruta en el archivo routes/web.php para que cuando la aplicación reciba una URL del tipo:

`http://localhost/numeros/14/27/9`

, se invoque al método *mayor* del controlador, y éste muestre al usuario las siguientes vistas:

- una plantilla llamada `numero.blade.php` (crear) que muestra el valor del número mayor (pasárselo como parámetro).
- o una plantilla llamada `error.blade.php` (crear) que muestre un mensaje de error en caso de que alguno de los tres parámetros recibidos no sea numérico.



Laravel: Formularios

- Los formularios son **vistas** que se almacenan (junto con las plantillas) en el directorio **resources/views**.
- El atributo **action** de los forms debe referenciar a una de las rutas definidas en **routes/web.php** → se ejecutará el método del controlador asociado a dicha ruta.

A screenshot of a web browser window displaying a form. The form contains:

- A text input field labeled "Username" with the value "example@gmail.com".
- A file input field with the placeholder "No file selected."
- A radio button input field.
- A "Browse..." button.
- A "Large" dropdown menu.
- A "Click Me!" button.

- **Ejemplo:** formulario que al ser enviado permite listar las ventas:

Archivo resources/views/listarVentas.blade.php :

```
<form method="get" action="/ventas/listar"> ← .php  
    <input type="text" name="nombre">  
    ...  
    <input type="submit" name="listar" value="listar">  
</form>
```



Esta ruta hará que se ejecute un método de un controlador para procesar el formulario.
La relación entre la **ruta**, el **controlador** y su **método** se define en el archivo routes/web.php:

```
Route::get('/ventas/listar', [ 'uses' => 'controladorVentas@listar' ]);
```

Laravel: Formularios

- En los métodos del controlador podemos **recuperar los valores de los campos enviados** por el formulario mediante el objeto **\$request** (tanto para GET como para POST).
- Existen **dos formas** para recoger el valor introducido/elegido en un control de un formulario:

P.e., dada la caja de texto: <input type="text" name="nombre">

Podemos recuperar el valor introducido mediante alguna de estas formas:

```
$nombre = $request->nombre;           // coincide con el atributo name del input  
$nombre = $request->input('nombre');
```

- Estos valores de los campos recuperados pueden usarse por los métodos del controlador para: validarlos, operar con ellos, pasárselos como parámetro a una vista...



El objeto \$request equivale al array asociativo superglobal
\$_REQUEST[] = \$_GET[] || \$_POST[]

Laravel: Formularios

- Los valores recuperados de los campos pueden usarse para:
Validarlos, Operar con ellos, Pasárselos como parámetro a una vista

Archivo app/http/Controllers/controladorVentas.php :

```
<?php  
namespace larawebapp\Http\Controllers;  
use Illuminate\Http\Request;  
  
class controladorVentas extends Controller {  
  
    // Los métodos del controlador se diseñan pensados  
    // para realizar las operaciones que permite la interfaz  
    // de la aplicación. Se encargan de llamar a las Vistas:  
  
    public function listar (Request $request) { ← Debemos incluir el objeto $request  
        if (empty($request->nombre)) {  
            return view('fichaError');  
        } else {  
            $nombre = $request->nombre;  
            return view('tablaVentas', ['cliente' => $nombre]);  
        }  
    }  
}
```

El objeto \$request equivale al array asociativo superglobal
\$_REQUEST[] = \$_GET[] || \$_POST[]

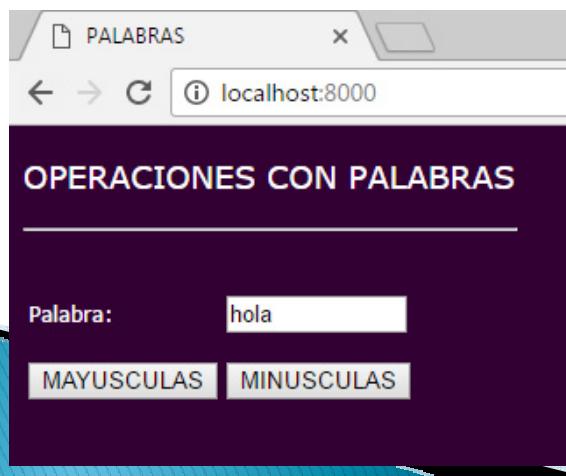
Laravel: Ejemplo Formulario

- **Ejemplo:** aplicación que convierte a mayúsculas/minúsculas una palabra.

Elementos necesarios:

- **1 Controlador:** contará con un método llamado *convertirPalabra*; que recibirá los datos del formulario (*\$request*) y convertirá la palabra → archivo **app/http/Controllers/controladorPalabra.php**
- **2 Vistas:**

a) Un formulario para introducir la palabra con dos botones: MAYÚSCULAS y MINÚSCULAS → **resources/views/formularioPalabra.blade.php**



PALABRAS

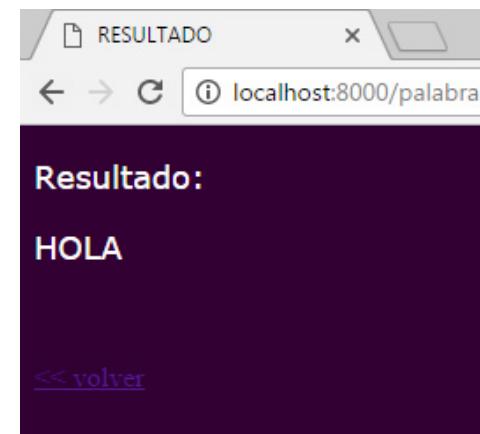
localhost:8000

OPERACIONES CON PALABRAS

Palabra: hola

MAYUSCULAS MINUSCULAS

b) Una plantilla para mostrar el resultado (la palabra convertida) → archivo **resources/views/resultado.blade.php**



RESULTADO

localhost:8000/palabra?

Resultado:

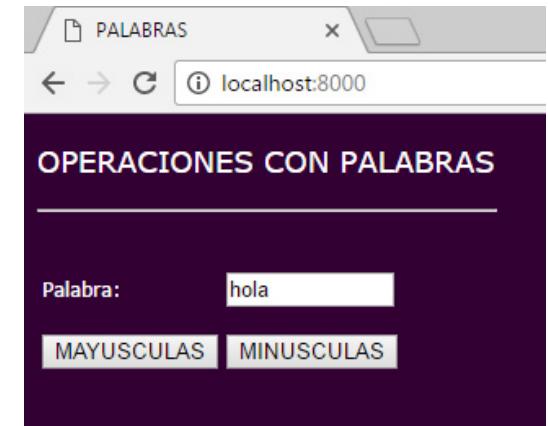
HOLA

<< volver

Laravel: Ejemplo Formulario

Archivo resources/views/formularioPalabra.blade.php

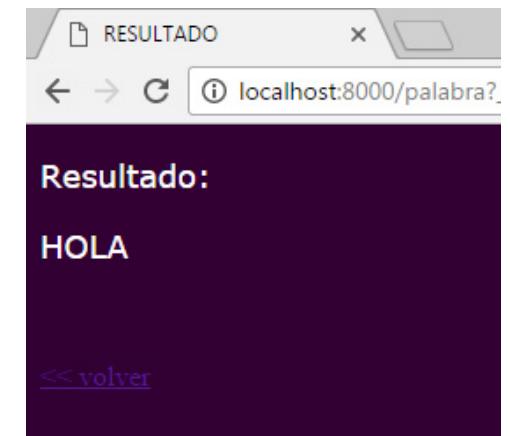
```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" >
        <meta http-equiv="X-UA-Compatible" content="IE=edge" >
        <meta name="viewport" content="width=device-width, initial-scale=1" >
            <title>PALABRAS</title>
            <style>
                .titulo { color:white; font-family:verdana; font-size:18px; }
                td { color:white; font-family:verdana; font-size:12px; }
            </style>
    </head>
    <body bgcolor="#330033">
        <p class="titulo">OPERACIONES CON PALABRAS</p>
        <hr align="left" width="280" >
        <p style="color:white;"> {{ $mensaje }} </p>
        <br>
        <form method="get" action="/palabra">
            {{ csrf_field() }}
            <table width="260" border="0" >
                <tr>
                    <td width="50"> Palabra: </td>
                    <td> <input type="text" name="palabra" size="10" > </td>
                </tr>
                <tr height="50" >
                    <td> <input type="submit" name="mayusculas" value="MAYUSCULAS" > </td>
                    <td> <input type="submit" name="minusculas" value="MINUSCULAS" > </td>
                </tr>
            </table>
        </form>
    </body>
</html>
```



Laravel: Ejemplo Formulario

Archivo **resources/views/resultado.blade.php**

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <style>
            .titulo { color:white;
                      font-family:verdana;
                      font-size:18px; }
        </style>
        <title>RESULTADO</title>
    </head>
    <body bgcolor="#330033">
        <p class="titulo"> Resultado: </p>
        <p class="titulo"> {{ $resultado }} </p>
        <br>
        <br>
        <a href="{{ url('/') }}"> << volver </a>
    </body>
</html>
```



↑ Archivo **resources/views/listadoIncidencias.blade.php**

Laravel: Ejemplo Formulario

Archivo app/Http/Controllers/controladorPalabra.blade.php

```
<?php
namespace larawebapp\Http\Controllers;
use Illuminate\Http\Request;

class controladorPalabra extends Controller{
    public function convertirPalabra (Request $request){
        // validación de los datos recibidos:
        if (empty($request->palabra)) {
            return view('formularioPalabra',
                ['mensaje' => "Debe indicar una palabra para convertir"]);
        } else {
            if (isset($request->mayusculas))
                $resultado = strtoupper($request->palabra);
            if (isset($request->minusculas))
                $resultado = strtolower($request->palabra);
            // Le pasamos el resultado a la Vista:
            return view('resultado', ['resultado' => $resultado]);
        }
    }
}
```

Archivo routes/web.php

```
<?php
Route::get('/',function() { // Inicio de la aplicación;
    return view('formularioPalabra', ['mensaje' => ""]);
});

Route::get('/palabra', ['uses' => 'controladorPalabra@convertirPalabra']);
```

Laravel: Ejemplo Varios Formularios

- **Ejemplo:** aplicación para el registro y consulta de incidencias on-line.

Las incidencias se almacenan en un archivo en el server (public/incidencias.dat)

La aplicación permite al usuario:

- **LISTAR** una tabla que muestre todas las incidencias registradas:

The screenshot shows a web browser window titled 'LISTADO DE INCIDENCIAS'. The URL is 'localhost:8000/incidencias/listar?_token=ICQgtVrD8JC8qshtEAMhmLnVEAIyHJu5ET8LV1ga&submit=LISTAR+INCIDENCIAS'. The page displays a table titled 'INCIDENCIAS REGISTRADAS' with 4 entries. The columns are: Asunto, Urgente, Tipo, Fecha, Id, E-mail, and Descripción. The data is as follows:

Asunto	Urgente	Tipo	Fecha	Id	E-mail	Descripción
asunto sin determinar		hardware		2	asad@asd.com	esto es la descripción de la incidencia
otra incidencia	on	redes	2020-02-01	5	asdad@asadas.esc	asd asada
equipo desconfigurado	on	software	2022-06-06	8	jorgqr@hotmail.com	No arranca el navegador
Sin conexión a la red		redes	2017-01-20	14	klllk@oop.com	No navega ni hace ping

[<< VOLVER](#)

↑ Archivo resources/views/listadoIncidencias.blade.php

- **INSERTAR** (registrar) una nueva incidencia mediante un form:

The screenshot shows a web browser window titled 'INCIDENCIAS'. The URL is 'localhost:8000'. The page is titled 'REGISTRO DE INCIDENCIAS ON-LINE'. It contains a form with the following fields:

- Asunto: (text input field)
- Urgente: (checkbox)
- Tipo: (radio buttons: Hardware, Redes, Software)
- Fecha: (date input field: dd/mm/aaaa)
- Identificador: (text input field)
- E-mail de contacto: (text input field)
- Descripción: (text area)

At the bottom are two buttons: 'INSERTAR INCIDENCIA' and 'LISTAR INCIDENCIAS'.

Archivo resources/views/formularioRegistrarIncidencia.blade.php →

Laravel: Formularios

- Dado que la aplicación permite realizar **dos operaciones**, cuenta con dos formularios que se enlazarán mediante las siguientes rutas definidas por el programador:

OPERACIÓN / FORMULARIO	ACTION / RUTA
INSERTAR INCIDENCIA	" <code>/incidencias/insertar</code> "
LISTAR INCIDENCIAS	" <code>/incidencias/listar</code> "

```
<form method="get" action="/incidencias/insertar">
...
</form>

<form method="get" action="/incidencias/listar">
...
</form>
```

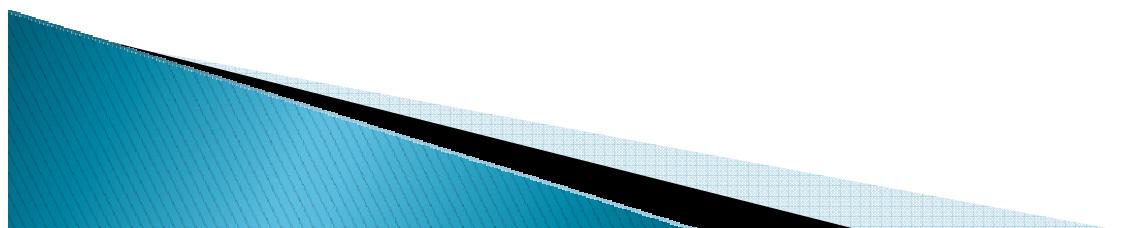
Laravel: Formularios

- Para la aplicación se definirá un único módulo/sección/controlador → **controladorIncidencias.php**, a crear en el directorio app/http/Controllers mediante:

```
C:\xampp\htdocs\proyectolara>php artisan make:controller controladorIncidencias
```

Al controlador se le añadirán **dos métodos** para poder realizar las dos operaciones que permite la aplicación web:

OPERACIÓN / FORMULARIO	ACCION/RUTA	MÉTODO DEL CONTROLADOR
INSERTAR INCIDENCIA	"/incidencias/insertar"	insertarIncidencia (Request \$request)
LISTAR INCIDENCIAS	"/incidencias/listar"	listarIncidencias ()



Laravel: Formularios

- Archivo app/http/Controllers/**controladorIncidencias.php**

```
<?php
namespace larawebapp\Http\Controllers;
use Illuminate\Http\Request;

class controladorIncidencias extends Controller {

    public function insertarIncidencia (Request $request){
        if (empty($request->asunto)) { // validación de los datos recibidos:
            return view('formularioRegistrarIncidencia',
                ['mensaje' => "Debe llenar los campos obligatorios de la incidencia"]);
        } else {
            // Escribe la incidencia en una sola línea con los valores separados por ":".
            $incidencia=$request->asunto.":". $request->urgente.":". $request->tipo.":". $request->fecha.":";
            $incidencia .= $request->identificador.":". $request->email.":". $request->descripcion;
            if (!file_exists("incidencias.dat")){
                // Crear el fichero con (w) porque no existe
                $ficheroIncidencias = fopen("incidencias.dat", "w");
                fputs($ficheroIncidencias, $incidencia.PHP_EOL);
                fclose($ficheroIncidencias);
            } else {
                $ficheroIncidencias = fopen("incidencias.dat", "a"); // escritura para añadir al final
                fputs($ficheroIncidencias, $incidencia.PHP_EOL);
                fclose($ficheroIncidencias);
            }
            return view('formularioRegistrarIncidencia',
                ['mensaje' => "Incidencia insertada correctamente"]);
        }
    }
}
```

Archivo public/incidencias.dat

```
otra incidencia:on:redes:2020-02-01:5:asdad@asadas.esc:descripcion
desconfigurado:on:software:2022-06-06:8:jorgqr@hotmail.com:No arranca
sin conexión::redes:2017-01-20:14:klllk@oop.com:No navega ni hace ping
```

Laravel: Formularios

- Continuación del archivo app/http/Controllers/**controladorIncidencias.php**

```
public function listarIncidencias () { // abre el archivo de incidencias y crea un array para la Vista
    if (!file_exists("incidencias.dat")){
        return view('formularioRegistrarIncidencia',[ 'mensaje'=>"No hay incidencias registradas"]);
    } else { // Cargamos el Modelo:
        $arrayIncidencias = array();
        $ficheroIncidencias = fopen("incidencias.dat", "r");
        while (!feof($ficheroIncidencias)) {
            $incidencia = fgets($ficheroIncidencias);
            if (strlen($incidencia) > 0)
                array_push($arrayIncidencias, $incidencia);
        }
        fclose($ficheroIncidencias);
        // Le pasamos el Modelo a la Vista:
        return view('listadoIncidencias', [ 'incidencias' => $arrayIncidencias]);
    }
}
```

- Archivo **routes/web.php**

```
Route::get('/',function(){ // Inicio de la aplicación; p.e.: http://www.larawebapp.com
    return view('formularioRegistrarIncidencia', [ 'mensaje' => ""]);});

Route::get('/incidencias/insertar', [ 'uses' => 'controladorIncidencias@insertarIncidencia' ]);
// URL: http://localhost:8000/incidencias/insertar

Route::get('/incidencias/listar', [ 'uses' => 'controladorIncidencias@listarIncidencias' ]);
// URL: http://localhost:8000/incidencias/listar
```

Laravel: Formularios

- La aplicación cuenta con dos Vistas (guardadas en resources/views):
 - Archivo **formularioRegistrarIncidencia.blade.php**

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>INCIDENCIAS</title>
        <style>
            .titulo { color:white; font-family:verdana; font-size:18px; }
            td { color:white; font-family:verdana; font-size:12px; }
        </style>
    </head>
    <body bgcolor="#5591BB">
        <p class="titulo">REGISTRO DE INCIDENCIAS ON-LINE</p>
        <hr align="left" width="360">
        <span> {{ $mensaje }} </span>
        <br><br>
        <form method="get" action="/incidencias/insertar">
            {{ csrf_field() }}
            <table width="360" border="0">
                <tr>
                    <td>Asunto:</td>
                    <td><input type="text" name="asunto"> * </td>
                </tr>
                <tr>
                    <td>Urgente:</td>
                    <td><input type="checkbox" name="urgente"></td>
                </tr>
```

The screenshot shows a web application window titled "INCIDENCIAS" at "localhost:8000". The main title is "REGISTRO DE INCIDENCIAS ON-LINE". The form has several fields:

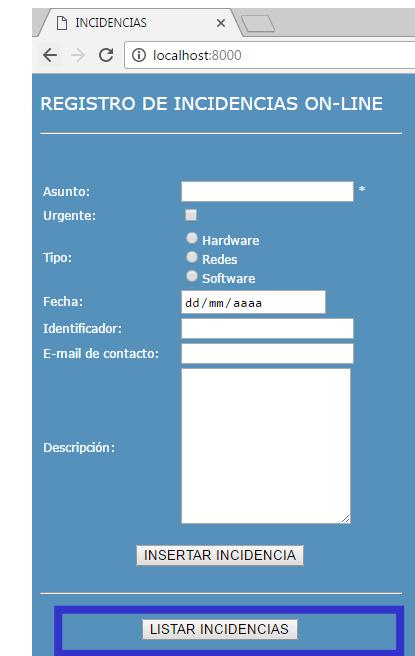
- "Asunto": A text input field with a red asterisk (*) indicating it is required. It is highlighted with a green box.
- "Urgente": A checkbox group with three options: "Hardware", "Redes", and "Software".
- "Tipo": A dropdown menu showing "dd/mm/aaaa".
- "Identificador": A text input field.
- "E-mail de contacto": A text input field.
- "Descripción": A large text area for description.

At the bottom of the form are two buttons: "INSERTAR INCIDENCIA" and "LISTAR INCIDENCIAS".

Laravel: Formularios

- Continuación del archivo **formularioRegistrarIncidencia.blade.php**

```
<tr>
    <td>Tipo:</td>
    <td> <input type="radio" name="tipo" value="hardware">Hardware<br>
        <input type="radio" name="tipo" value="redes">Redes<br>
        <input type="radio" name="tipo" value="software">Software<br>
    </td>
</tr>
<tr> <td>Fecha:</td> <td> <input type="date" name="fecha"> </td> </tr>
<tr> <td>Identificador:</td> <td><input type="number" name="identificador" min="1"></td> </tr>
<tr> <td>E-mail de contacto:</td><td> <input type="email" name="email"> </td> </tr>
<tr> <td>Descripción:</td>
    <td> <textarea name="descripcion" rows="10" cols="21"></textarea> </td>
</tr>
<tr> <td colspan="2" align="center" height="50">
    <input type="submit" name="submit" value="INSERTAR INCIDENCIA"> </td>
</tr>
</table>
</form>
<hr align="left" width="360">
<form method="get" action="/incidencias/listar">
    {{ csrf_field() }}
    <table width="360" border="0">
        <tr> <td align="center" height="50">
            <input type="submit" name="submit" value="LISTAR INCIDENCIAS">
        </td> </tr>
    </table>
</form>
</body>
</html>
```



Laravel: Formularios

- Archivo **listadoIncidentes.blade.php**

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>LISTADO DE INCIDENCIAS</title>

        <style>
            .titulo { color:white; font-family:verdana; font-size:18px; }
            .tabla { border-collapse:collapse; border: 1px solid black;
                      background:#77B3DD; color:white; font-family:verdana; font-size:12px; }
            td { background:#88C4EE; }
        </style>

        <script language="javascript" type="text/javascript">
            function filaMouseOver (fila) {
                fila.style.color = 'red';
            }

            function filaMouseOut (fila) {
                fila.style.color = 'white';
            }
        </script>
    </head>
```

La plantilla recibe como parámetro un array de incidentes extraídas de las líneas del archivo **public/incidentes.dat**

Asunto	Urgente	Tipo	Fecha	Id	E-mail	Descripción
asunto sin determinar		hardware		2	asad@asd.com	esto es la descripción de la incidencia
otra incidencia	on	redes	2020-02-01	5	asdad@asadas.esc	asd asada
equipo desconfigurado	on	software	2022-06-06	8	jorgqr@hotmail.com	No arranca el navegador
Sin conexión a la red		redes	2017-01-20	14	klllk@oop.com	No navega ni hace ping

[<< VOLVER](#)

Laravel: Formularios

- Continuación del archivo **listadoIncidencias.blade.php**

```
<body bgcolor="#5591BB>
    <p class="titulo">INCIDENCIAS REGISTRADAS </p>
    <hr align="left" width="800">

    <span> <?php echo count($incidencias)." incidencias registradas"; ?> </span>
    <br><br>

    <table width="800" border="1" class="tabla">
        <tr height="30">
            <th width="150">Asunto</th> <th width="65">Urgente</th> <th width="65">Tipo</th>
            <th width="85">Fecha</th> <th width="35">Id</th> <th width="130">E-mail</th>
            <th>Descripción</th>
        </tr>
        <?php
            foreach ($incidencias as $incidencia) {
                $arrayIncidencia = explode(":", $incidencia);
                echo "<tr height=\"25\""
                    onMouseOver=\"filaMouseOver(this);\" onMouseOut=\"filaMouseOut(this);\"";
                foreach ($arrayIncidencia as $campo) {
                    echo "<td>".$campo."</td>";
                }
                echo "</tr>";
            }
        ?>
    </table>

    <br>
    <a href="{{ url('/') }}> << VOLVER</a>

</body>
</html>
```

Laravel: Formularios

- **Actividad 1:** completa la aplicación anterior para añadir un **nuevo formulario** en la vista **formularioRegistrarIncidencia.blade.php** que permita **buscar** las incidencias que contengan un determinado asunto:

Debes asociar la acción del nuevo formulario a la ruta "**/incidencias/buscar**" en el archivo `routes/web.php`

Debes añadir a la clase controladora `controladorIncidencias.php` el método **buscarIncidencias (Request \$request)**

El método llamará a la vista **listadoIncidencias** pasándole como parámetro el array de incidencias halladas en el archivo `incidencias.dat`.

The screenshot shows a web browser window with the title "INCIDENCIAS" and the URL "localhost:8000". The page is titled "REGISTRO DE INCIDENCIAS ON-LINE". It contains a form with the following fields:

- Asunto: (text input field with asterisk)
- Urgente: (checkbox)
- Tipo: (radio buttons for Hardware, Redes, Software)
- Fecha: (text input field for date dd/mm/aaaa)
- Identificador: (text input field)
- E-mail de contacto: (text input field)
- Descripción: (text area)

Below the form are two buttons: "INSERTAR INCIDENCIA" and "LISTAR INCIDENCIAS". At the bottom, there is a search bar with the label "Asunto:" and a button "BUSCAR INCIDENCIAS". A green box highlights the "Asunto:" input field.

Laravel: Formularios

- **Actividad 2:** completa la aplicación anterior para añadir un **nuevo formulario** en la vista **formularioRegistrarIncidencia.blade.php** que permita **eliminar** una incidencia indicando su Identificador.

Debes asociar la acción del nuevo formulario a la ruta "**/incidencias/eliminar**" en el archivo **routes/web.php**

Debes añadir a la clase controladora **controladorIncidencias.php** el método **eliminarIncidencia (Request \$request)**

Recuerda añadir las rutas al archivo **routes/web.php**:

```
Route::get('/incidencias/buscar',
    ['uses' => 'controladorIncidencias@buscarIncidencias']);

Route::get('/incidencias/eliminar',
    ['uses' => 'controladorIncidencias@eliminarIncidencia']);
```

INCIDENCIAS

localhost:8000

REGISTRO DE INCIDENCIAS ON-LINE

Asunto: *

Urgente:

Tipo:

Hardware

Redes

Software

Fecha:

dd / mm / aaaa

Identificador:

E-mail de contacto:

Descripción:

INSERTAR INCIDENCIA

LISTAR INCIDENCIAS

BUSCAR INCIDENCIAS

Identificador:

ELIMINAR INCIDENCIA

Laravel: Formularios

- **Actividad 3:** modifica la aplicación anterior para mejorar su modularidad y navegabilidad. Divide la vista **formularioRegistrarIncidencia.blade.php** (que contiene 4 formularios) en 4 vistas separadas:

formularioRegistrarIncidencia.blade.php (1)

formularioListarIncidencias.blade.php (2)

formularioBuscarIncidencias.blade.php (3)

formularioEliminarIncidencia.blade.php (4)



INSERTAR INCIDENCIA

localhost:8000/incidencias/insertar

REGISTRO DE INCIDENCIAS ON-LINE

Debe llenar los campos obligatorios de la incidencia

Asunto: _____ *

Urgente:

Tipo: Hardware Redes Software

Fecha: dd / mm / aaaa

Identificador: _____

E-mail de contacto: _____

Descripción: _____

INSERTAR INCIDENCIA

<< VOLVER

(1)



LISTADO DE INCIDENCIAS

localhost:8000/incidencias/listar

INCIDENCIAS REGISTRADAS

5 incidencias registradas

Asunto	Urgente	Tipo	Fecha	Id	E-mail	Descripción
asunto sin determinar		hardware		2	asad@asd.com	esto es la descripción de la incidencia
otra incidencia	on	redes	2020-02-01	5	asdad@asadas.esc	asd asada
equipo desconfigurado	on	software	2022-06-06	8	jorgqr@hotmail.com	No arranca el navegador
Sin conexión a la red		redes	2017-01-20	14	klllk@oop.com	No navega ni hace ping
ejemplo 3	on	hardware	2019-01-02	32	asdad@asadas.esc	detalle

<< VOLVER

(2)



BUSCAR INCIDENCIAS

localhost:8000/incidencias/buscar

REGISTRO DE INCIDENCIAS ON-LINE

Debe indicar el asunto por el que buscar

Asunto: _____ BUSCAR INCIDENCIAS

<< VOLVER

(3)



ELIMINAR INCIDENCIA

localhost:8000/incidencias/eliminar

REGISTRO DE INCIDENCIAS ON-LINE

Identificador: _____ ELIMINAR INCIDENCIA

<< VOLVER

(4)

Laravel: Formularios

- Por último, crea una nueva **vista** llamada **inicioIncidencias.blade.php** que asociarás a la ruta '/' para que se muestre como página inicial de la aplicación. Esta página tendrá enlaces a las secciones de la web (formularios y otras vistas):

Archivo routes/web.php

```
Route::get('/',function(){ // Inicio de la aplicación
    return view('inicioIncidencias', ['mensaje' => ""]);});

Route::get('/incidencias/insertar',
    ['uses' => 'controladorIncidencias@insertarIncidencia']);

Route::get('/incidencias/listar',
    ['uses' => 'controladorIncidencias@listarIncidencias']);

Route::get('/incidencias/buscar',
    ['uses' => 'controladorIncidencias@buscarIncidencias']);

Route::get('/incidencias/eliminar',
    ['uses' => 'controladorIncidencias@eliminarIncidencia']);
```



Utiliza la función **url('ruta')** para generar un enlace que llame a cada vista-sección por medio del controlador:

```
<a href="{{ url('/incidencias/insertar') }}>
    INSERTAR INCIDENCIA </a>
```

Laravel: Formularios

- **Actividad 4:** modifica la aplicación anterior para que el **formularioRegistrarIncidencia.blade.php** permita tanto **insertar** como **actualizar** (modificar) una incidencia:

- Si el *Identificador* indicado en el formulario **no existe** en el archivo public/incidencias.dat → **insertar** la nueva incidencia al final del archivo

- Si el *Identificador* indicado **sí existe** → **modificar** en el archivo la incidencia existente.

INSERTAR INCIDENCIA

localhost:8000/incidencias/insertar

REGISTRO DE INCIDENCIAS ON-LINE

Debe llenar los campos obligatorios de la incidencia

Asunto: _____ *

Urgente:

Tipo:

Hardware

Redes

Software

Fecha: dd / mm / aaaa

Identificador: _____

E-mail de contacto: _____

Descripción:

INSERTAR - MODIFICAR INCIDENCIA

<< VOLVER