

# **CRUD con Laravel (usando plantillas)**

Resumen de pasos basado en:

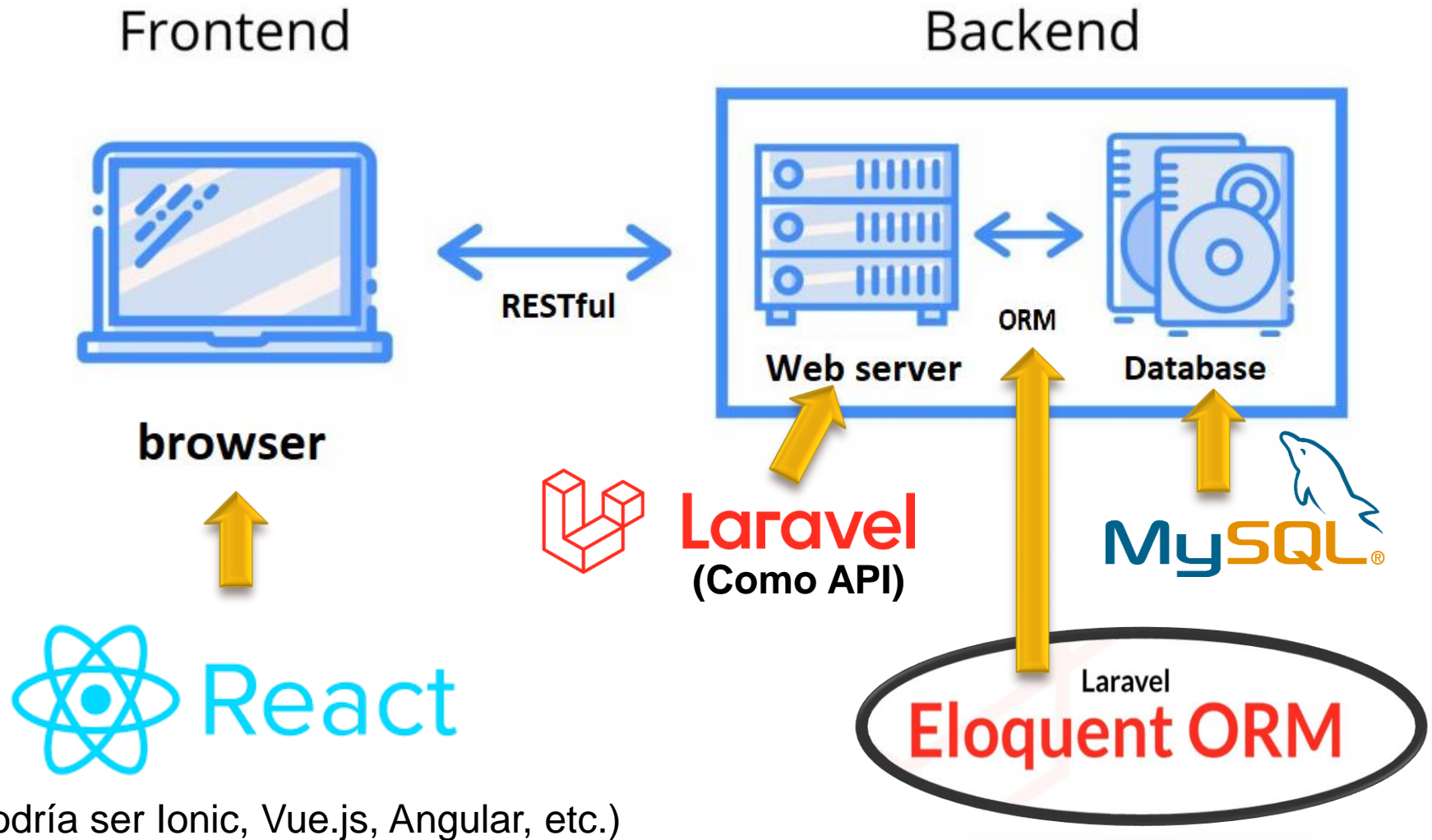
<https://www.sitepoint.com/laravel-project-setup-beginners-guide/>

**Repasemos los  
2 enfoques que  
hemos visto**

# ¿Dónde corre Laravel?

## (Enfoque 1: Creando una API)

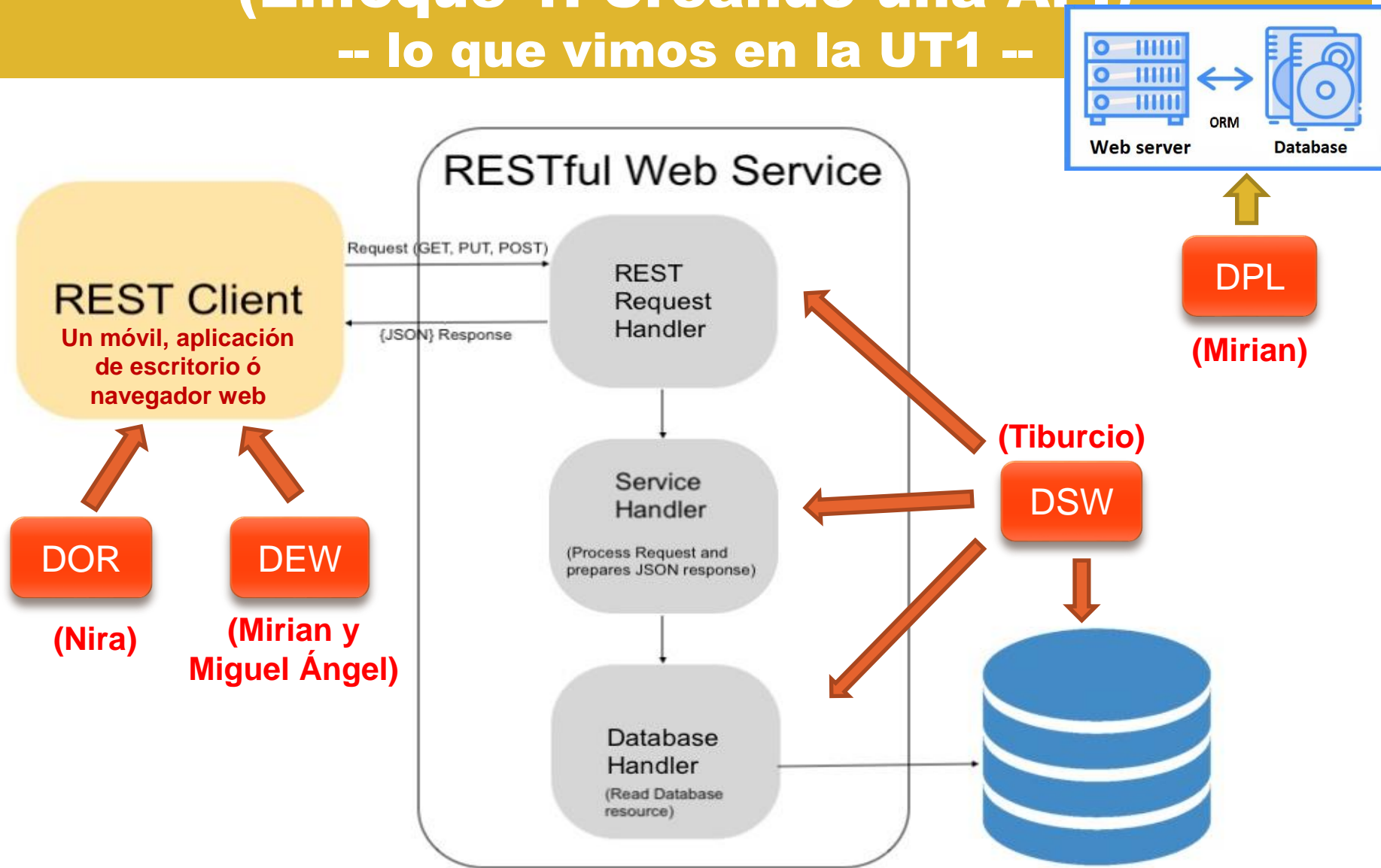
-- lo que vimos en la UT1 --



# Nuestra visión...

## (Enfoque 1: Creando una API)

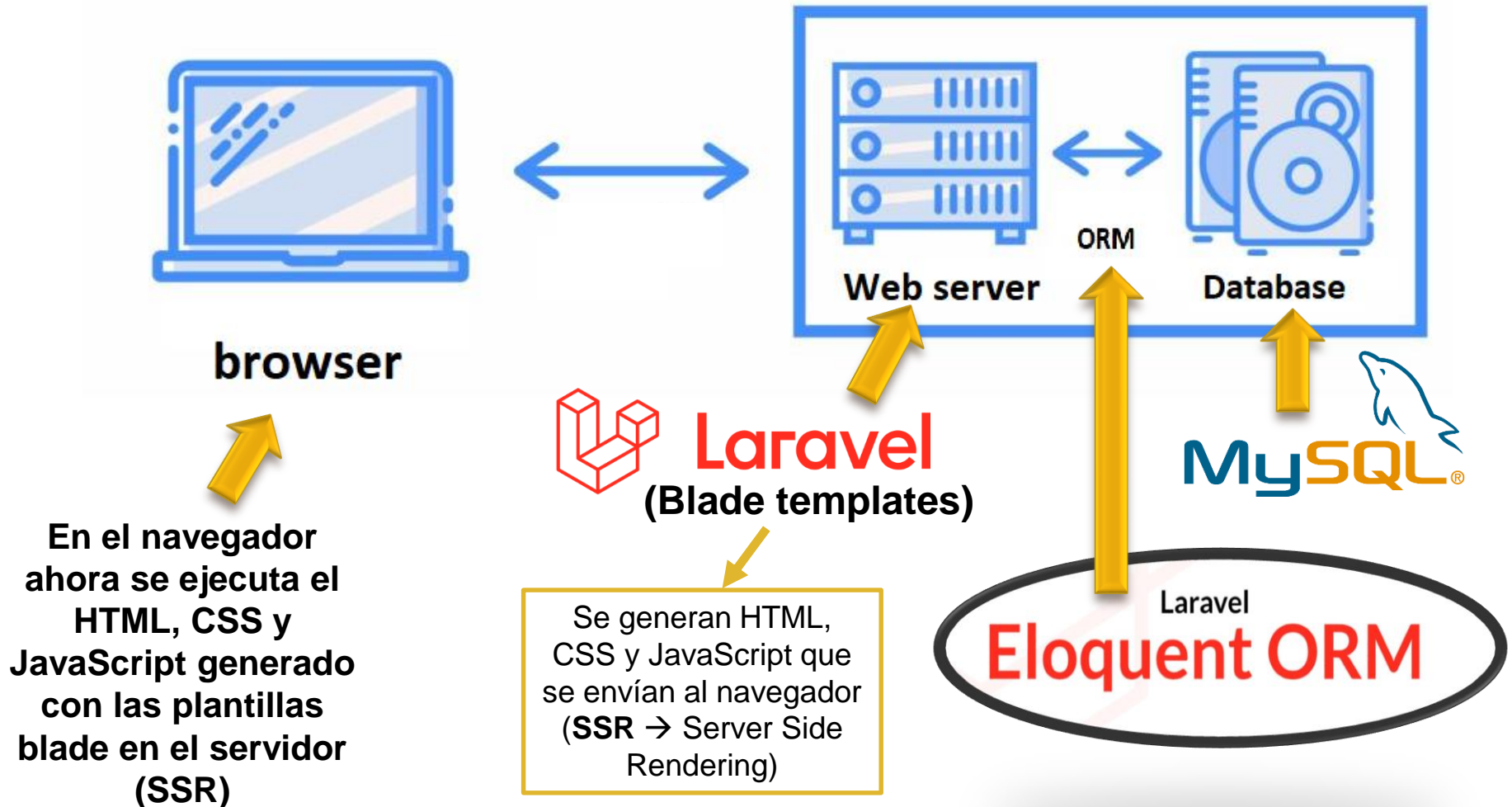
-- lo que vimos en la UT1 --



# ¿Dónde corre Laravel?

## (Enfoque 2: usando plantillas)

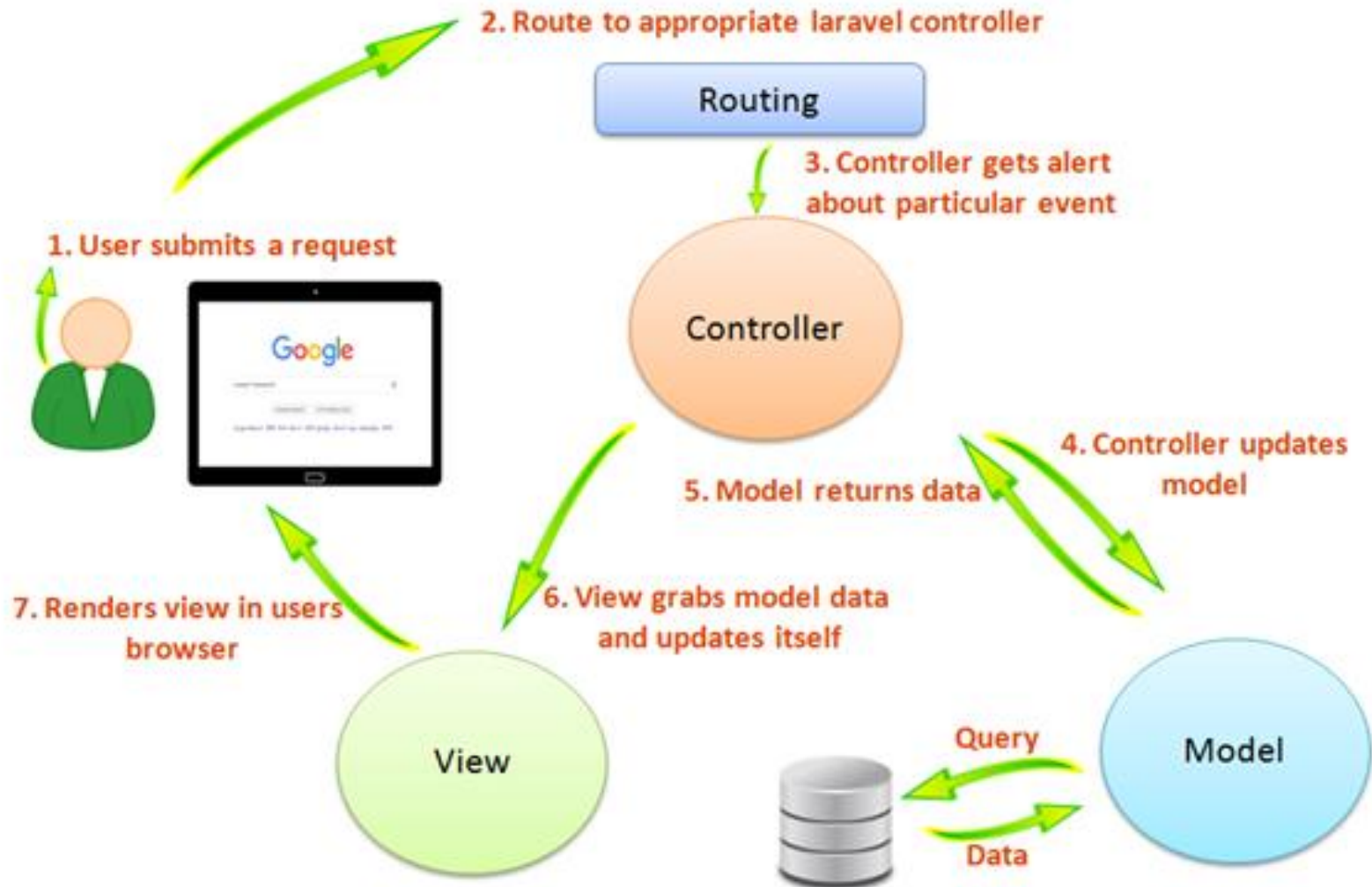
-- lo nuevo en la UT2 --



# ¿Dónde corre Laravel?

## (Enfoque 2: usando plantillas)

-- lo nuevo en la UT2 --





# Punto de partida de este tutorial

- Partimos de que ya tienes un entorno de trabajo configurado.
- En este tutorial aparecerán algunas referencias a Laragon pero en realidad es válido para cualquier otro tipo de instalación que tengas como por ejemplo XAMPP, Homestead, Herd o cualquier otra.





**¿Tenemos todo  
lo que  
necesitamos?**

# Comprobemos que tenemos todo instalado...

Pasos previos

Cmder

C:\laragon\www

λ php --version

PHP 8.3.12 (cli) (built: Sep 24 2024 20:22:14) (NTS Visual C++ 2019 x64)  
Copyright (c) The PHP Group  
Zend Engine v4.3.12, Copyright (c) Zend Technologies

C:\laragon\www

λ mysql --version

mysql Ver 8.0.30 for Win64 on x86\_64 (MySQL Community Server - GPL)

C:\laragon\www

λ composer --version

Composer version 2.4.1 2022-08-20 11:44:50

C:\laragon\www

λ node --version

v18.8.0

C:\laragon\www

λ npm --version

npm warn cli npm v10.8.2 does not support Node.js v18.8.0. This version of npm supports the following node versions: `^18.17.0 || >=20.5.0`. You can find the latest version at <https://nodejs.org/>.  
10.8.2

C:\laragon\www

λ laravel --version

Podemos comprobar las versiones de php, mysql, composer, node, npm y laravel dentro de la terminal de laragon.

# Si no habías creado anteriormente tu proyecto “bicycles”...

**Creamos nuestro proyecto con Laravel**

```
C:\laragon\www  
λ laravel new bicycles
```

Con este comando creamos el proyecto llamado “bicycles”.

```
Would you like to install a starter kit? [No starter kit]:
```

```
[none] No starter kit  
[breeze] Laravel Breeze  
[jetstream] Laravel Jetstream
```

```
> none
```

Podemos elegir diferentes kits de partida. Nosotros vamos a usar uno vacío

```
Which testing framework do you prefer? [Pest]:
```

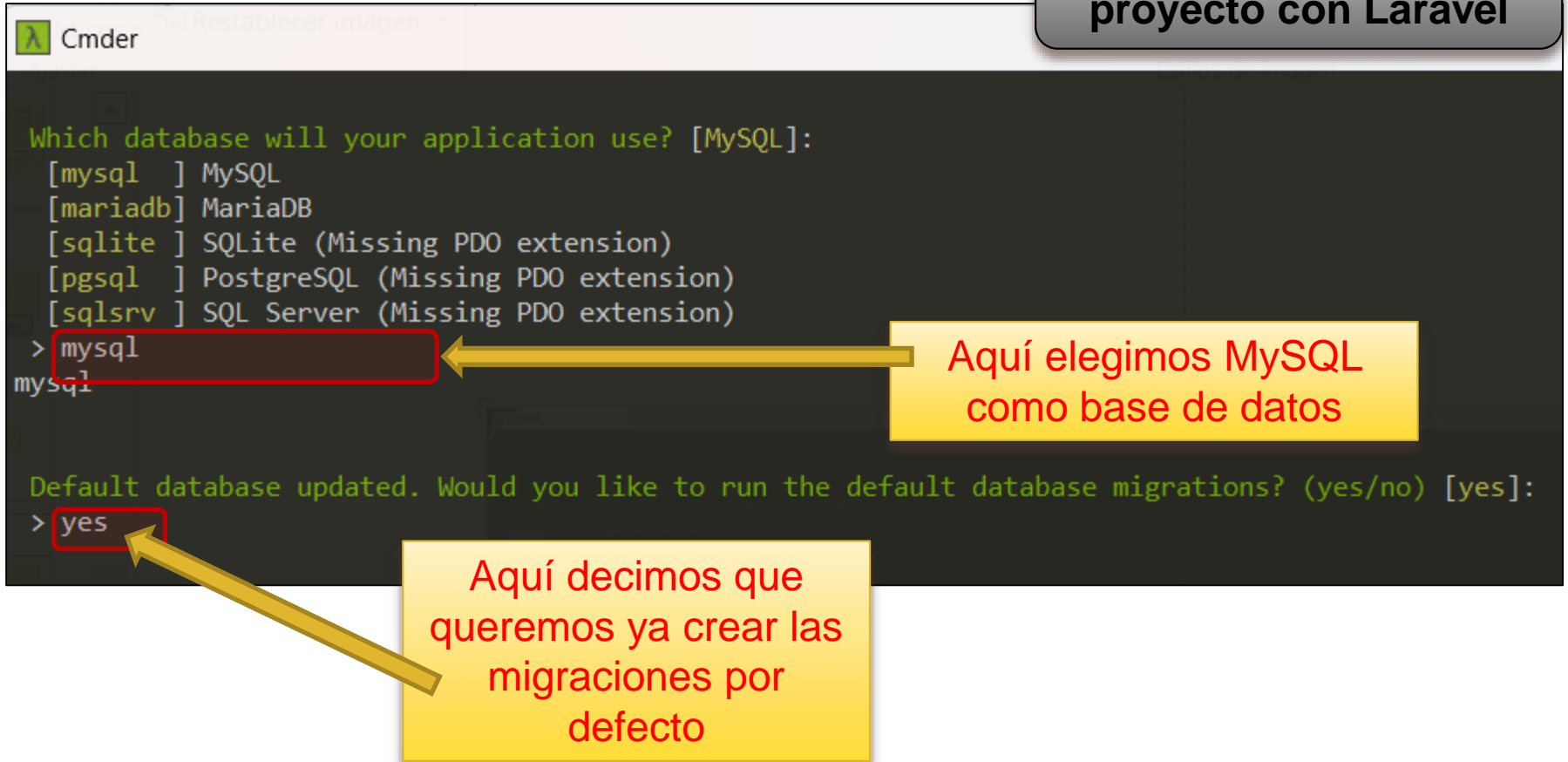
```
[0] Pest  
[1] PHPUnit
```

```
> 1
```

Los tests no los vamos a ver ahora, pero yo elegí PHPUnit

# Si no habías creado anteriormente tu proyecto “bicycles”...

**Creamos nuestro  
proyecto con Laravel**



```
Cmder
Restablecer imagen

Which database will your application use? [MySQL]:
[mysql  ] MySQL
[mariadb] MariaDB
[sqlite ] SQLite (Missing PDO extension)
[pgsql  ] PostgreSQL (Missing PDO extension)
[sqlsrv ] SQL Server (Missing PDO extension)
> mysql
mysql

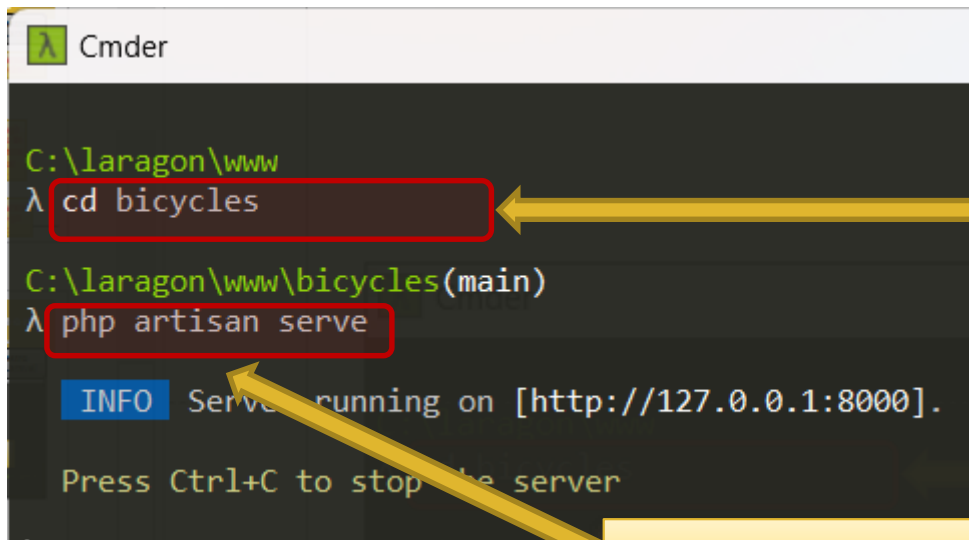
Default database updated. Would you like to run the default database migrations? (yes/no) [yes]:
> yes
```

Aquí elegimos MySQL como base de datos

Aquí decimos que queremos ya crear las migraciones por defecto

# Si no habías creado anteriormente tu proyecto “bicycles”...

**Creamos nuestro proyecto con Laravel**



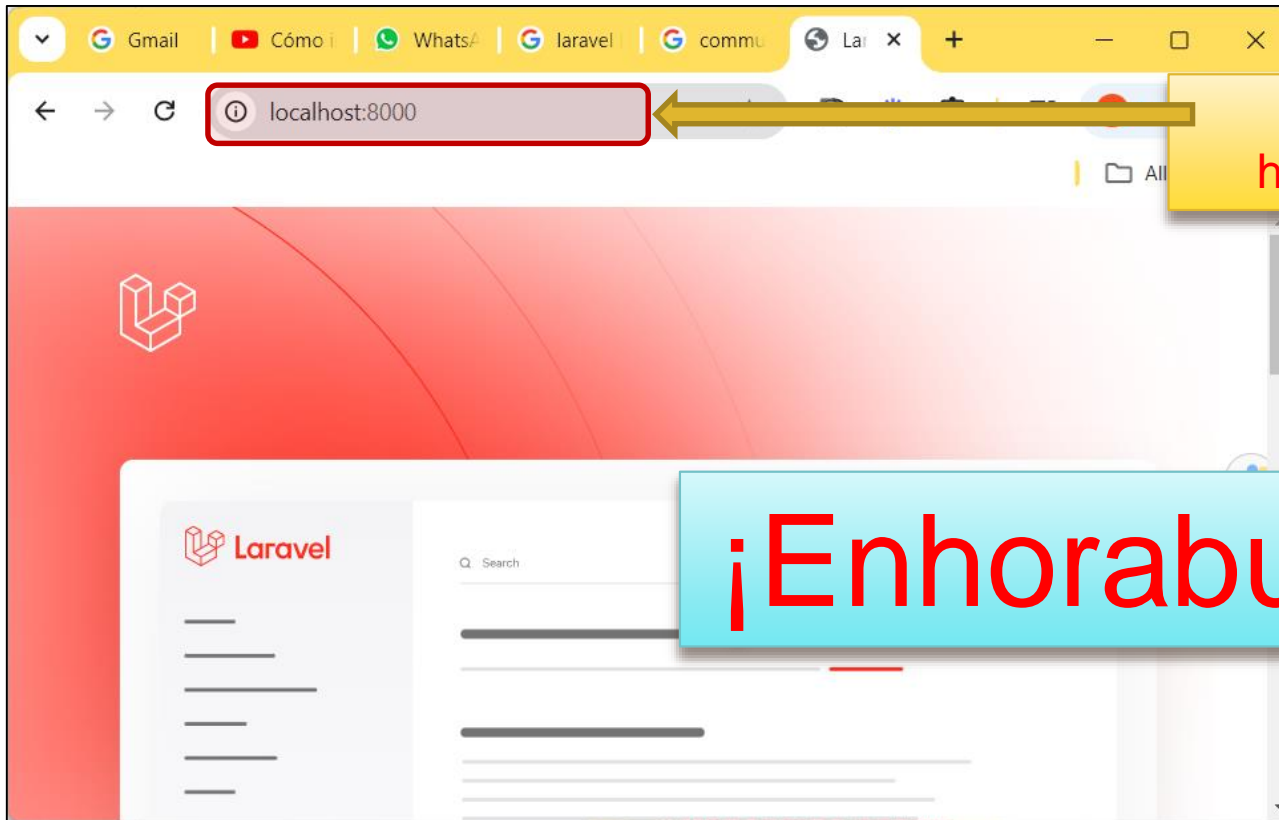
```
C:\laragon\www
λ cd bicycles
C:\laragon\www\bicycles(main)
λ php artisan serve
INFO Server running on [http://127.0.0.1:8000].
Press Ctrl+C to stop the server
```

**Cambiamos al directorio con nuestro proyecto**

**Arrancamos por fin nuestro proyecto**

# Si no habías creado anteriormente tu proyecto “bicycles”...

**Creamos nuestro  
proyecto con Laravel**



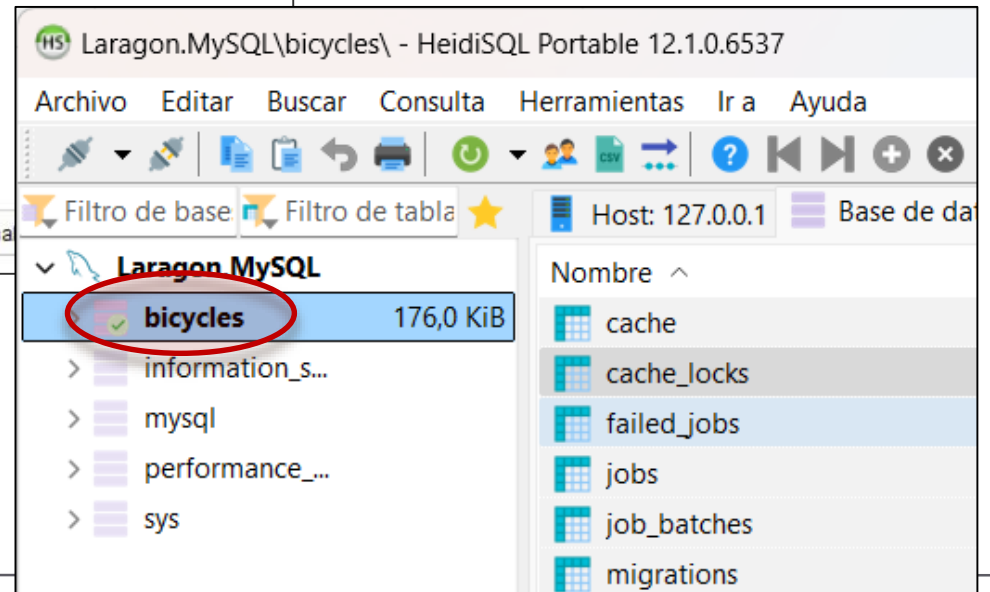
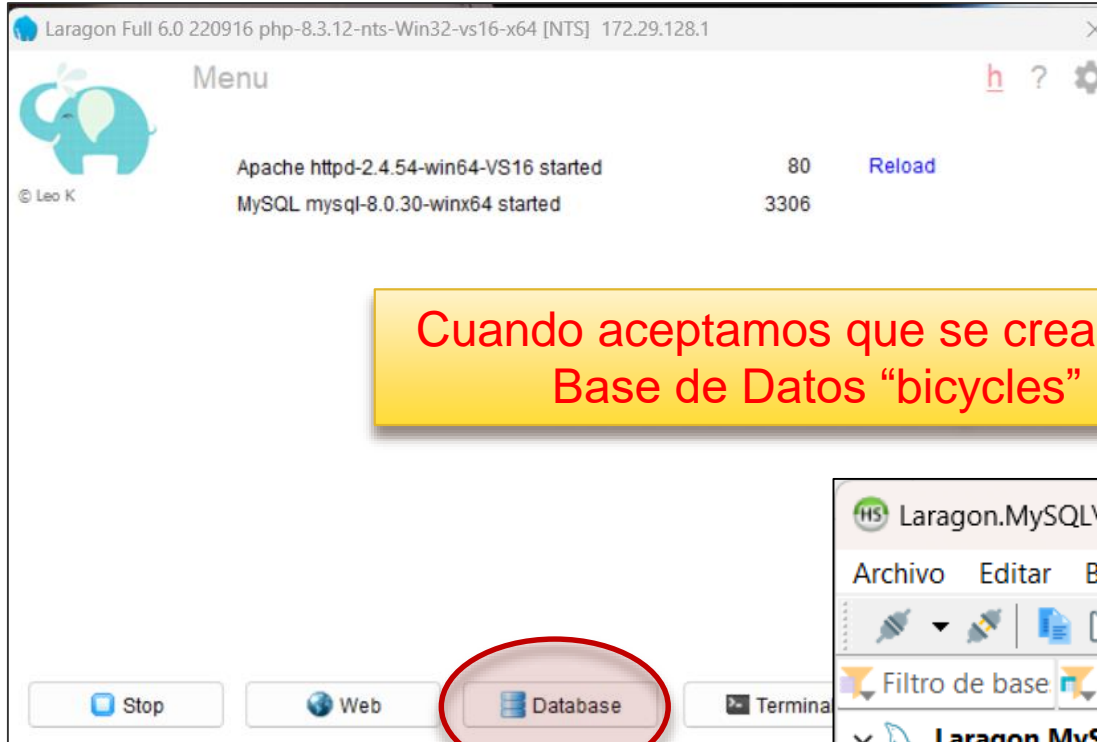
**Vamos a la URL  
<http://localhost:8000>**

## ¡Enhorabuena!

# Observa además que...

En MySQL ya hay una base de datos "bicycles"

Cuando aceptamos que se crearan las migraciones se creó en la Base de Datos "bicycles" un buen montón de tablas.



# Veamos dónde están algunas cosas en un proyecto de Laravel

(si quieres ir directamente al grano sáltate esta sección y vete a “[Empecemos a picar código ya por favor](#)”)

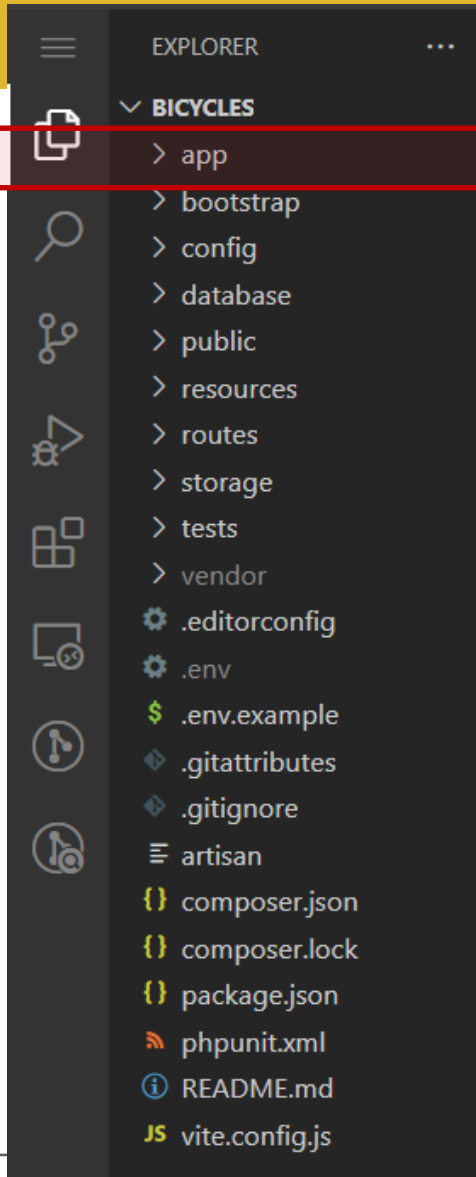


# Estructura de un proyecto Laravel...

**Estructura de un proyecto Laravel**

Este directorio alberga la lógica central de nuestra aplicación, incluidos controladores, modelos y proveedores de servicios

**Es donde estaremos tocando código la mayor parte del tiempo**



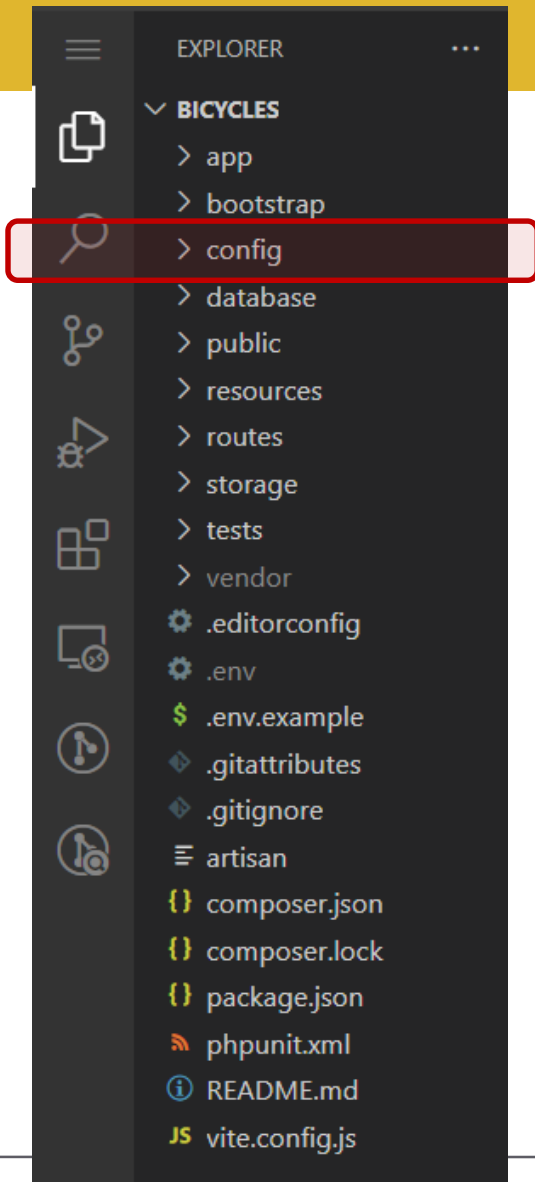
# Estructura de un proyecto Laravel...

## Estructura de un proyecto Laravel

Los archivos de configuración para varios componentes de nuestra aplicación se pueden encontrar aquí, lo que nos permite buscar y personalizar configuraciones como conexiones y servicios de bases de datos desde un único punto del proyecto.

**Por ejemplo aquí configuraremos la conexión a la Base de datos.**

**¿Te acuerdas cómo es esto en un proyecto de Express?**

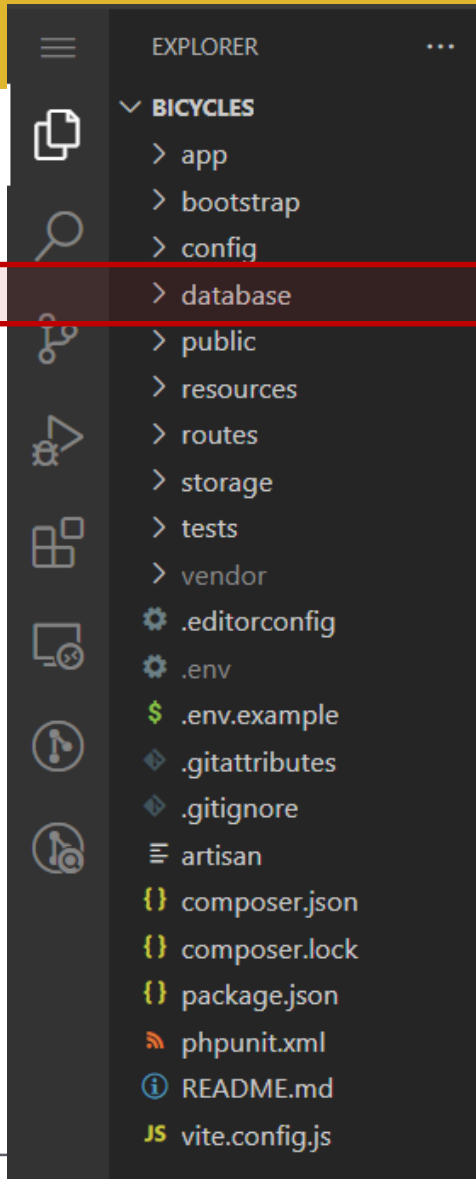


# Estructura de un proyecto Laravel...

## Estructura de un proyecto Laravel

En este directorio, definiremos las migraciones y semillas de nuestra base de datos que utilizará Eloquent ORM de Laravel. Eloquent simplifica la gestión de bases de datos.

**Cada vez que vayamos a crear, borrar o modificar la estructura de una tabla trabajaremos aquí gestionando las migraciones.**

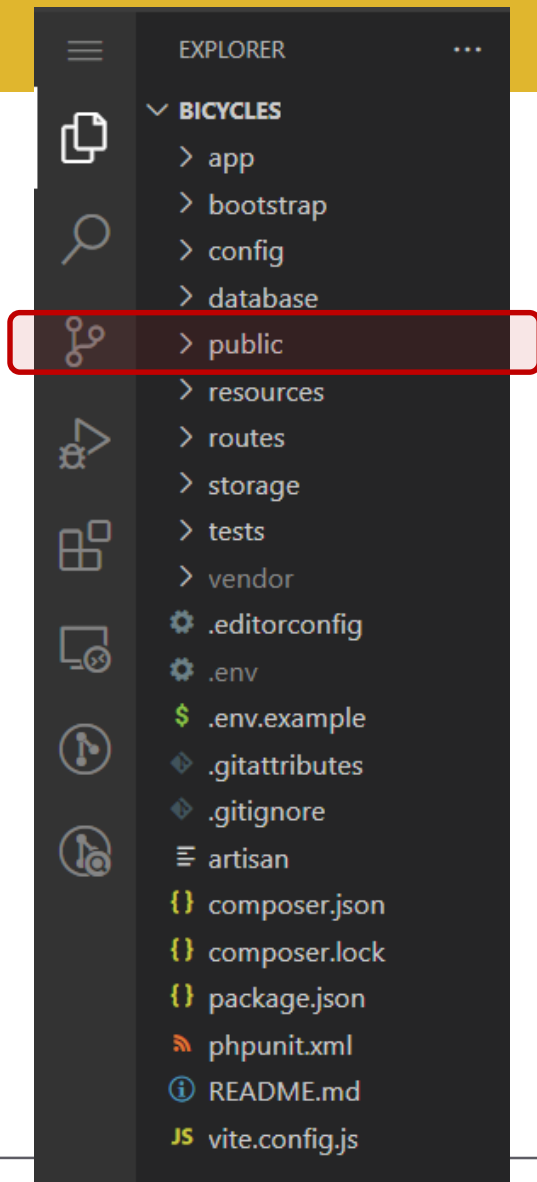


# Estructura de un proyecto Laravel...

## Estructura de un proyecto Laravel

Los archivos de acceso público como CSS, JavaScript e imágenes están aquí. Este directorio también contiene el punto de entrada para nuestra aplicación, el archivo index.php.

**Aquí añadiremos los recursos estáticos como por ejemplo las imágenes que son accesibles públicamente.**

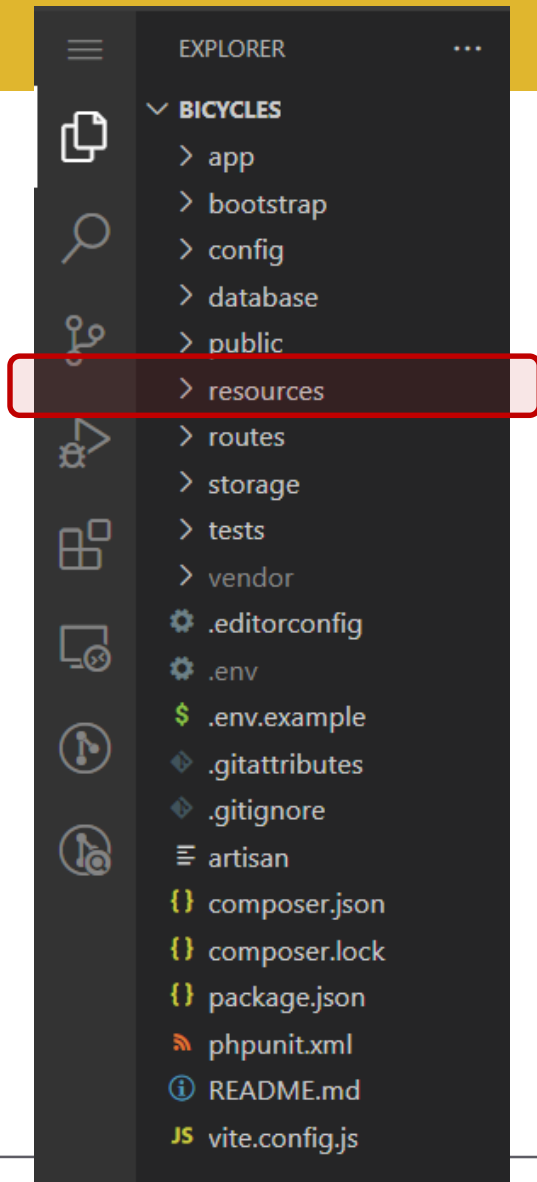


# Estructura de un proyecto Laravel...

## Estructura de un proyecto Laravel

Los archivos sin compilar y sin procesar de nuestra aplicación, como las plantillas Blade, Sass y JavaScript, se almacenan aquí.

**Se diferencia del directorio “public” en que aquí no es accesible directamente desde el sitio web. Las plantillas Blade, por ejemplo, se colocan aquí pero no son accesibles públicamente, sino que se procesan en el servidor para generar las páginas cuando el cliente lo solicita. Eso es lo que conocemos como SSR ó Server Side Rendering.**



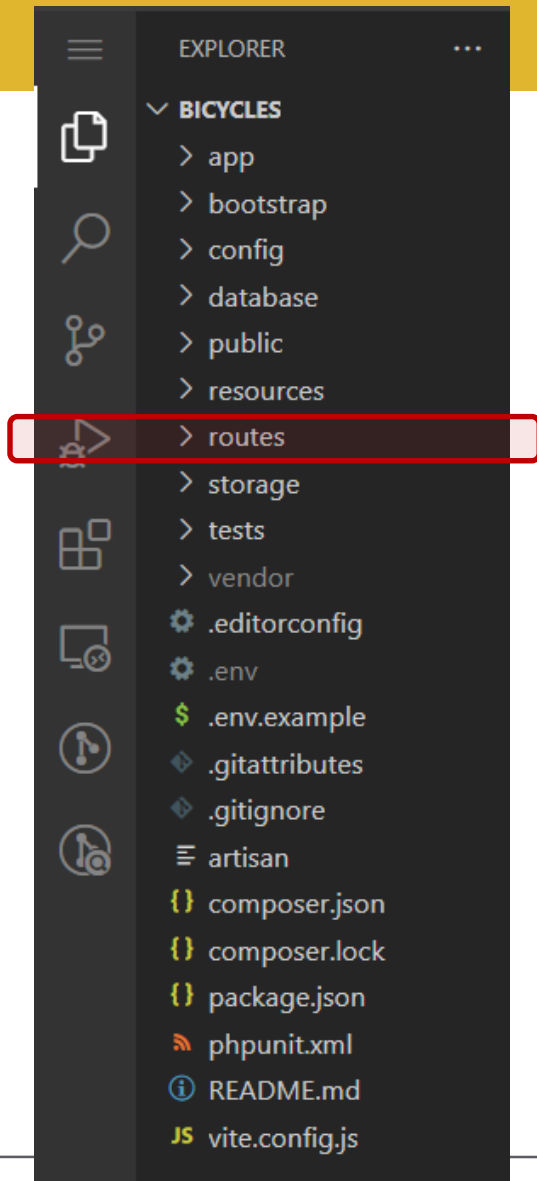
# Estructura de un proyecto Laravel...

## Estructura de un proyecto Laravel

La configuración de enrutamiento de Laravel se administra en este directorio.

**Aquí están las rutas indicando cada ruta a que controlador se dirige.**

**¿Te acuerdas cómo es esto en un proyecto de Express?**

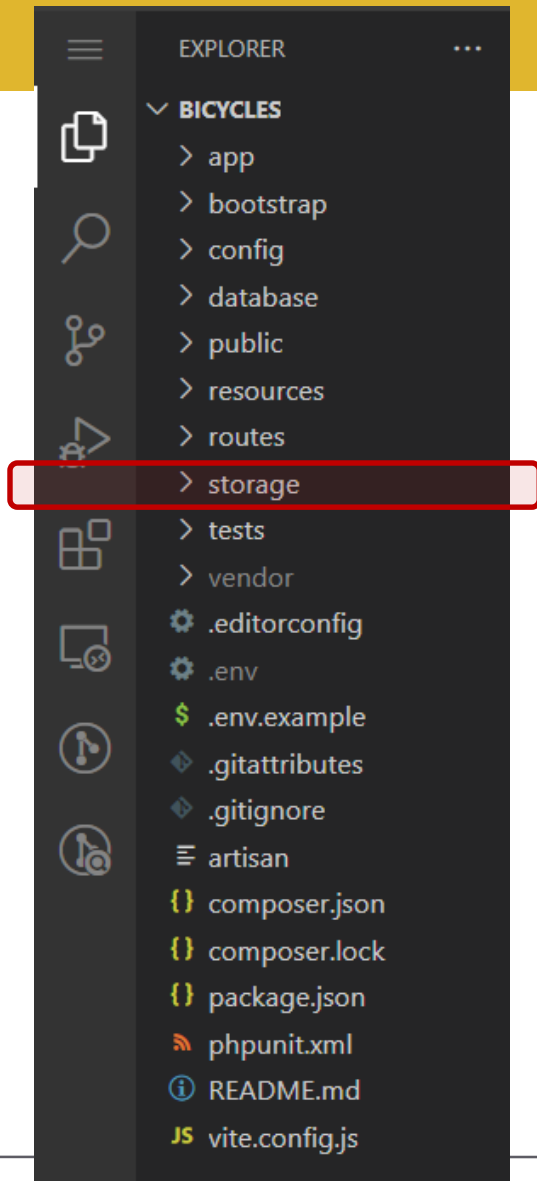


# Estructura de un proyecto Laravel...

## Estructura de un proyecto Laravel

Los archivos temporales y de caché, así como los registros, se almacenan aquí

**Mira los registros de log aquí para ver que ha pasado.**



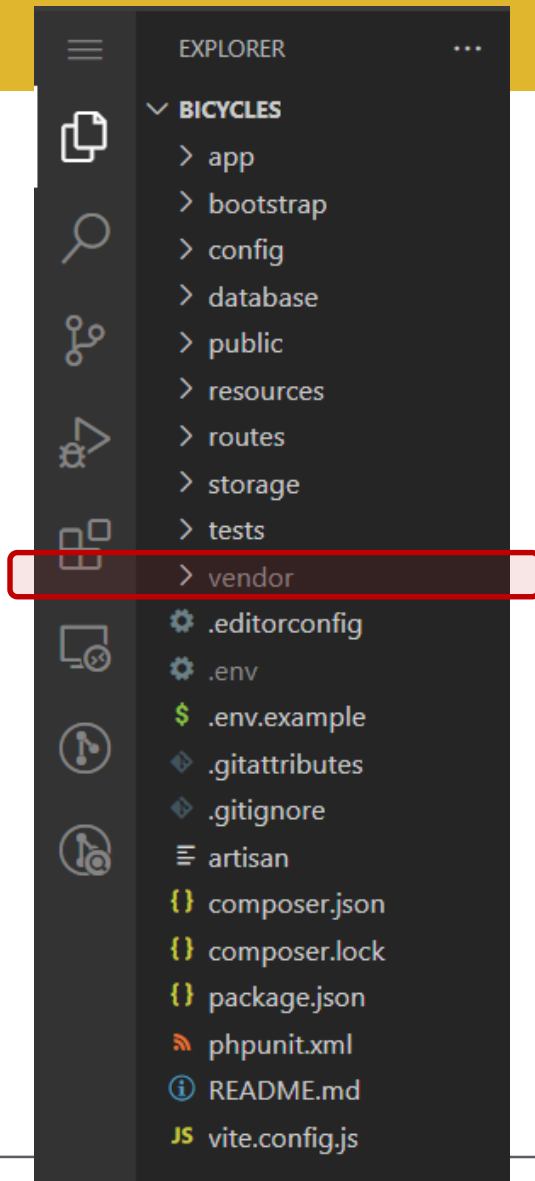
# Estructura de un proyecto Laravel...

## Estructura de un proyecto Laravel

Composer manages our project's dependencies in this directory. All downloaded libraries will be in this directory

Es como el “node\_modules” de un proyecto con node.

**NO SE TOCA normalmente.**

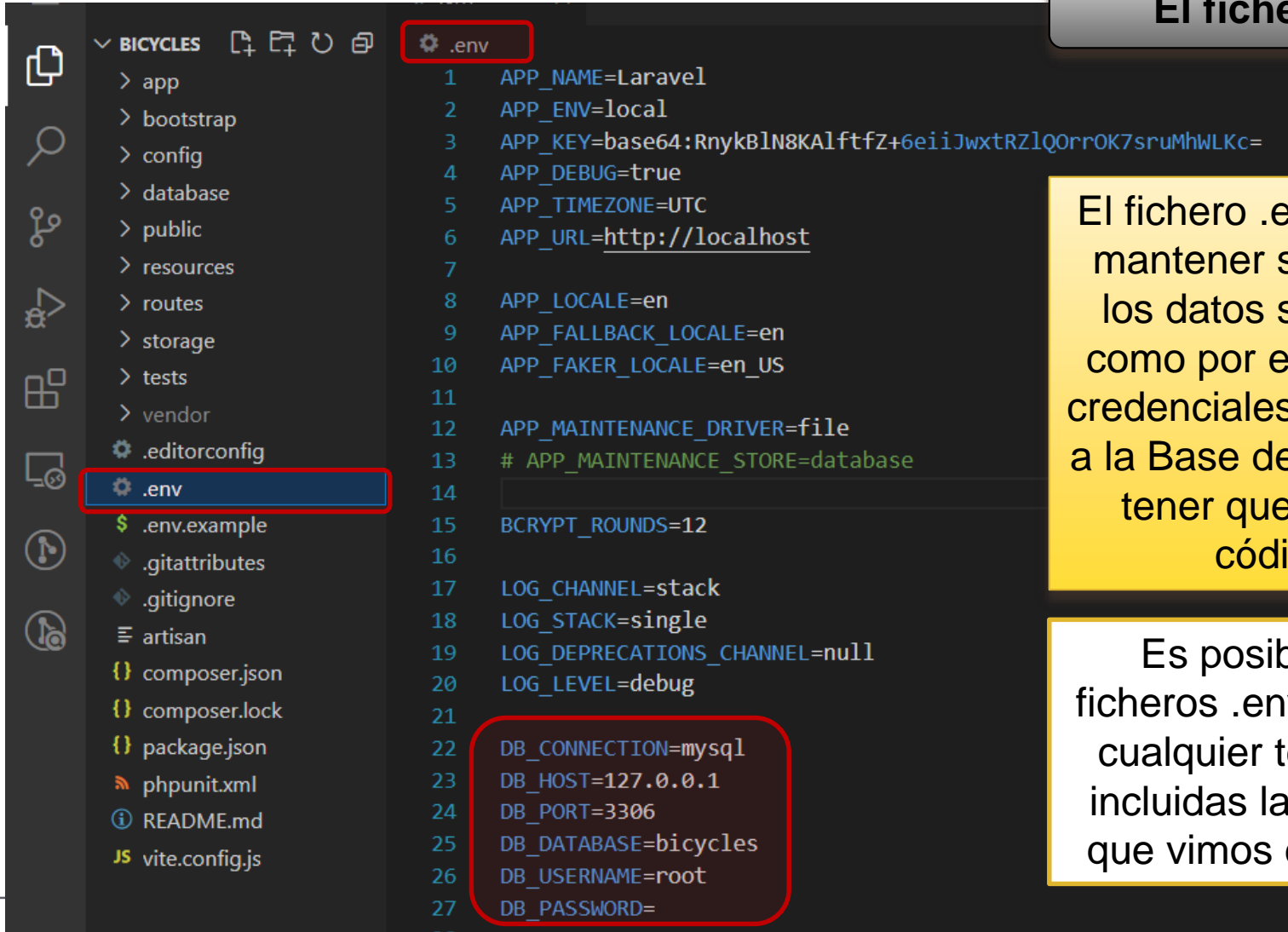




# El fichero .env

## (Mantiene a salvo los datos sensibles)

### El fichero .env



```
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:RnykBlN8KAlftfZ+6eiiJwxtRZlQ0rrOK7sruMhWLKc=
4 APP_DEBUG=true
5 APP_TIMEZONE=UTC
6 APP_URL=http://localhost
7
8 APP_LOCALE=en
9 APP_FALLBACK_LOCALE=en
10 APP_FAKER_LOCALE=en_US
11
12 APP_MAINTENANCE_DRIVER=file
13 # APP_MAINTENANCE_STORE=database
14
15 BCRYPT_ROUNDS=12
16
17 LOG_CHANNEL=stack
18 LOG_STACK=single
19 LOG_DEPRECATED_CHANNEL=null
20 LOG_LEVEL=debug
21
22 DB_CONNECTION=mysql
23 DB_HOST=127.0.0.1
24 DB_PORT=3306
25 DB_DATABASE=bicycles
26 DB_USERNAME=root
27 DB_PASSWORD=
```

El fichero .env permite mantener separados los datos sensibles, como por ejemplo las credenciales de acceso a la Base de Datos, sin tener que tocar el código.

Es posible usar ficheros .env para casi cualquier tecnología incluidas las de node que vimos en la UT1.

# Veamos dónde se usa una variable de entorno del fichero .env

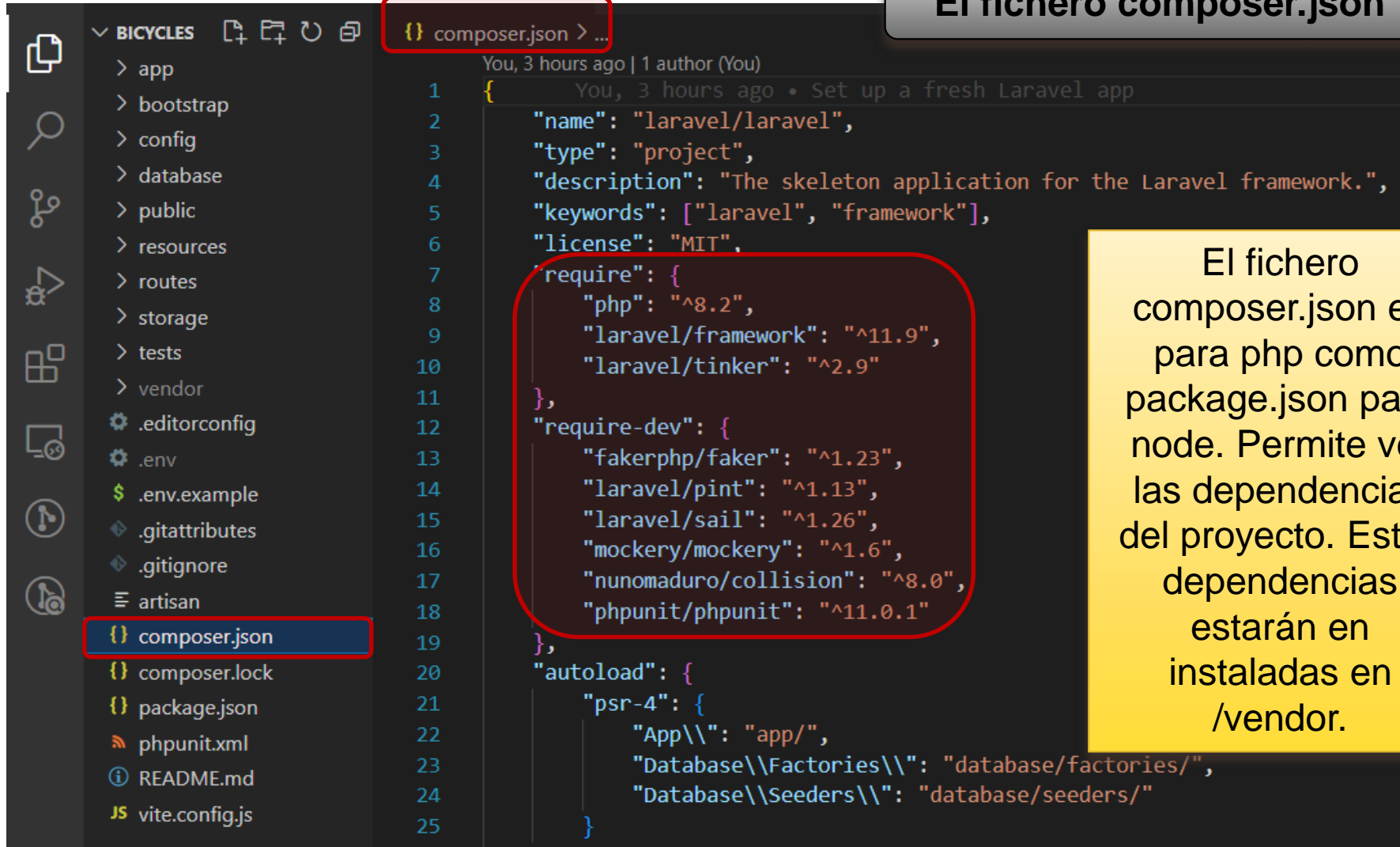
Un ejemplo de dónde se pueden usar las variables de entorno .env

```
config > database.php
5 return [
32     'connections' => [
34         'sqlite' => [
43     ],
44
45     'mysql' => [
46         'driver' => 'mysql',
47         'url' => env('DB_URL'),
48         'host' => env('DB_HOST', '127.0.0.1'),
49         'port' => env('DB_PORT', '3306'),
50         'database' => env('DB_DATABASE', 'laravel'),
51         'username' => env('DB_USERNAME', 'root'),
52         'password' => env('DB_PASSWORD', ''),
53         'unix_socket' => env('DB_SOCKET', ''),
54         'charset' => env('DB_CHARSET', 'utf8mb4'),
55         'collation' => env('DB_COLLATION', 'utf8mb4_unicode_ci'),
56         'prefix' => '',
57         'prefix_indexes' => true,
61         PDO::MYSQL_ATTR_SSL_CA => env('MYSQL_ATTR_SSL_CA'),
62     ] : [],
63 ],
64
```

Observa que en **config/database.php** se usan algunas de las variables de entorno de connexion que habíamos visto en el fichero .env

# El fichero composer.json (Contiene las dependencias del proyecto)

El fichero composer.json



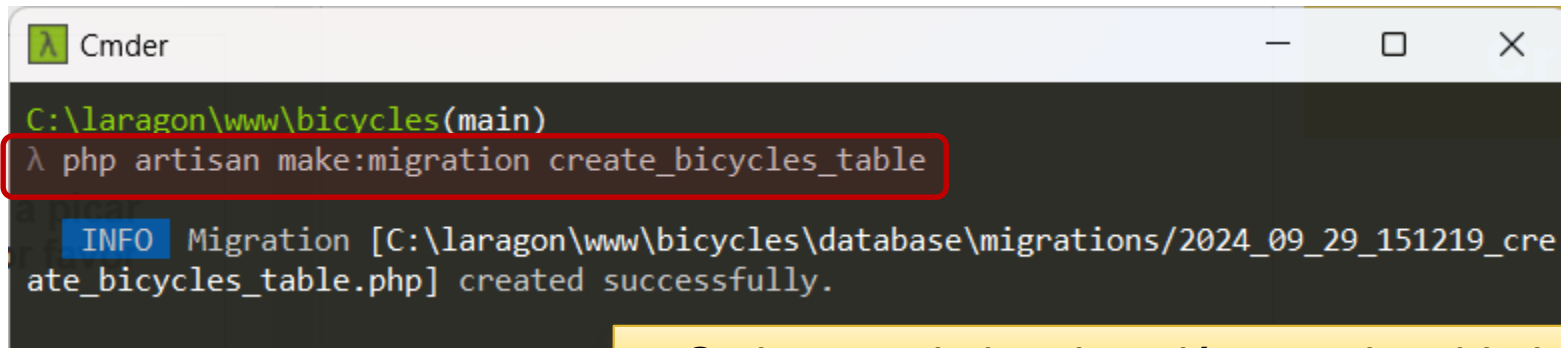
```
{
  "name": "laravel/laravel",
  "type": "project",
  "description": "The skeleton application for the Laravel framework.",
  "keywords": ["laravel", "framework"],
  "license": "MIT",
  "require": {
    "php": "^8.2",
    "laravel/framework": "^11.9",
    "laravel/tinker": "^2.9"
  },
  "require-dev": {
    "fakerphp/faker": "^1.23",
    "laravel/pint": "^1.13",
    "laravel/sail": "^1.26",
    "mockery/mockery": "^1.6",
    "nunomaduro/collision": "^8.0",
    "phpunit/phpunit": "^11.0.1"
  },
  "autoload": {
    "psr-4": {
      "App\\": "app/",
      "Database\\Factories\\": "database/factories/",
      "Database\\Seeders\\": "database/seeders/"
    }
  }
}
```

El fichero composer.json es para php como package.json para node. Permite ver las dependencias del proyecto. Estas dependencias estarán en instaladas en /vendor.

**Empecemos a picar  
código ya por favor**

# Creemos una migración

## Creación de una migración



```
C:\laragon\www\bicycles(main)
λ php artisan make:migration create_bicycles_table

INFO Migration [C:\laragon\www\bicycles\database\Migrations\2024_09_29_151219_create_bicycles_table.php] created successfully.
```

Se ha creado la migración para la tabla bicycles.

**Crear una migración consiste en crear un fichero que contiene la definición de lo que luego va a ser una tabla en la Base de Datos**

# Veamos la migración creada...

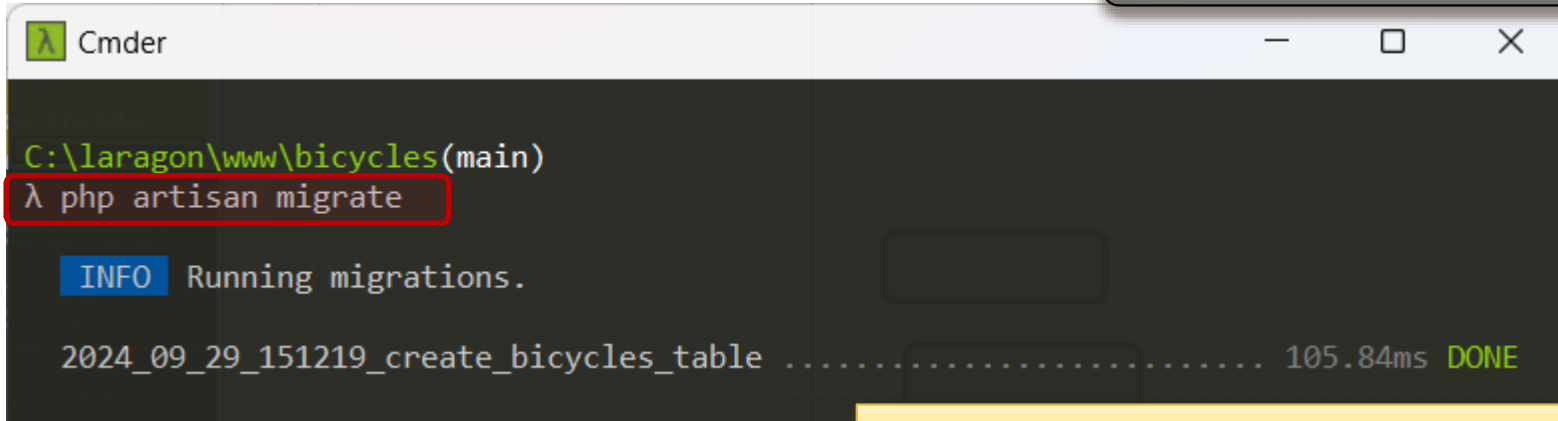
El fichero año\_mes\_día\_hora\_create\_bicycles\_table.php

```
database > migrations > 2024_09_29_151219_create_bicycles_table.php
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12     public function up(): void
13     {
14         Schema::create('bicycles', function (Blueprint $table) {
15             $table->id();
16             $table->string('brand');
17             $table->string('model');
18             $table->timestamps();
19         });
20     }
21 }
```

Fijate como he creado los campos “brand” y “model” en la migración “bicycles”.

# Después de crear la migración, ahora vamos a ejecutarla...

## Ejecución de una migración



```
C:\laragon\www\bicycles(main)  
λ php artisan migrate  
  
INFO Running migrations.  
  
2024_09_29_151219_create_bicycles_table ..... 105.84ms DONE
```

La migración se ha ejecutado

El resultado de ejecutar una migración es que, en este caso, la tabla “bicycles” se crea en la base de datos “bicycles”.

Las migraciones existen para la mayoría de las tecnologías de ORM. Para Sequelize también existen.

# El resultado es que se ha creado la tabla “bicycles” en la BD “bicycles” con los campos indicados en la migración...

Resultado de la ejecución de una migración

Laragon.MySQL\bicycles\bicycles\ - HeidiSQL Portable 12.1.0.6537

Archivo Editar Buscar Consulta Herramientas Ir a Ayuda

Filtro de base Filtro de tabla

**Laragon.MySQL**

- bicycles** 192,0 KiB
  - bicycles** 16,0 KiB
  - cache 16,0 KiB
  - cache\_locks 16,0 KiB
  - failed\_jobs 16,0 KiB
  - jobs 16,0 KiB
  - job\_batches 16,0 KiB
  - migrations 16,0 KiB
  - password\_r... 16,0 KiB
  - sessions 48,0 KiB
  - users 16,0 KiB
- information\_s...
- mysql
- performance\_...

Host: 127.0.0.1 Base de datos: bicycles Tabla: bicycles Datos Consulta

Básico Opciones Índices (1) Llaves foráneas (0) Comprobar restricciones

Nombre: bicycles

Comentario:

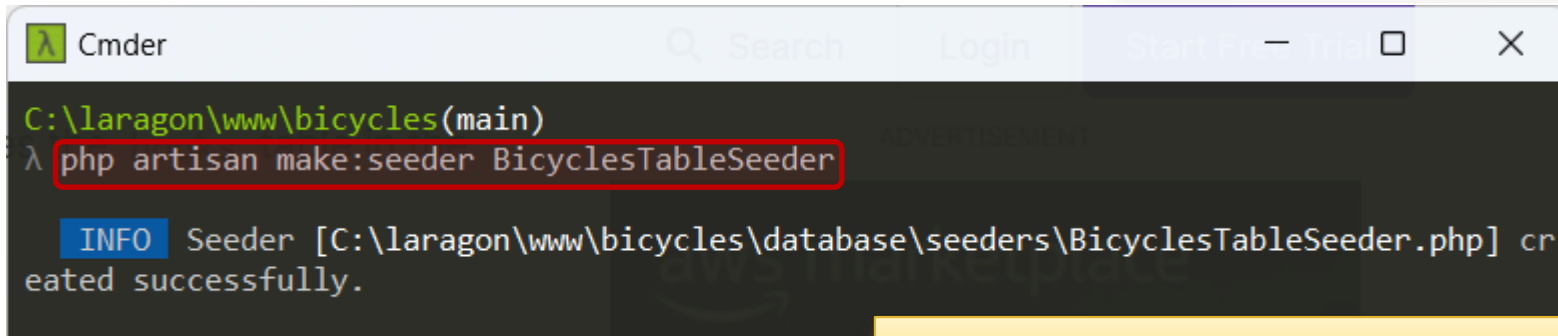
Columnas: + Agregar × Borrar ▲ Subir ▼ Bajar

#	Nombre	Tipo de datos	Longitud/Con...	Sin signo	Permitir ...
1	id	BIGINT	20	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2	brand	VARCHAR	255	<input type="checkbox"/>	<input type="checkbox"/>
3	model	VARCHAR	255	<input type="checkbox"/>	<input type="checkbox"/>
4	created_at	TIMESTAMP		<input type="checkbox"/>	<input checked="" type="checkbox"/>
5	updated_at	TIMESTAMP		<input type="checkbox"/>	<input checked="" type="checkbox"/>



# Crea un seeder para poblar la tabla con datos de prueba...

Creación de un seeder



```
Cmdr
C:\laragon\www\bicycles(main)
λ php artisan make:seeder BicyclesTableSeeder

INFO Seeder [C:\laragon\www\bicycles\database\seeders\BicyclesTableSeeder.php] created successfully.
```

Se ha creado el Seeder para la table bicycles

**Crear un seeder consiste en crear un fichero que contiene datos de prueba que luego se insertarán en la tabla correspondiente en la Base de Datos, que en este caso es en la tabla “bicycles”.**

Sequelize también tiene seeders

# Veamos el seeder creado...

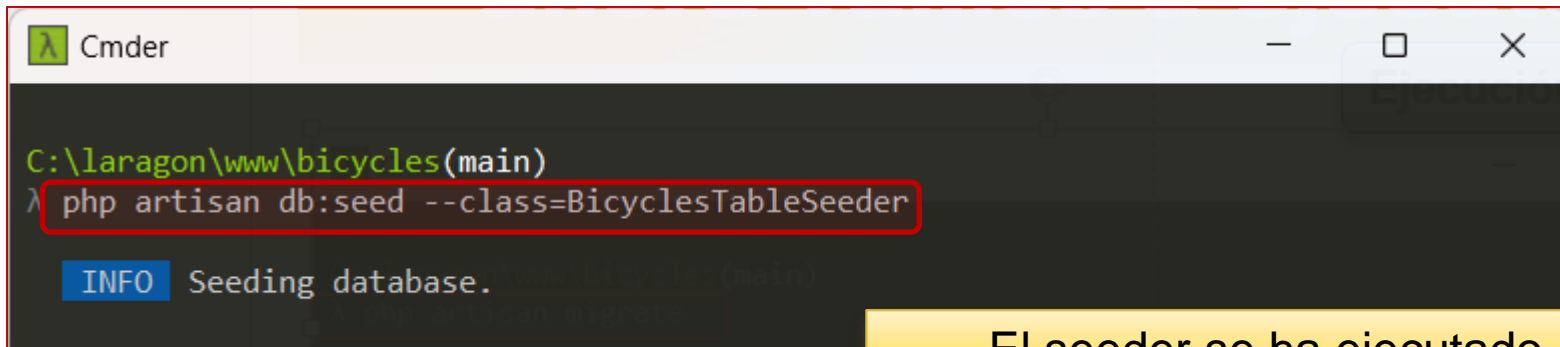
El fichero database/seeder/BicyclesTableSeeder.php

```
database > seeders > BicyclesTableSeeder.php
1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6  use Illuminate\Database\Seeder;
7
8  use DB;
9
10 class BicyclesTableSeeder extends Seeder
11 {
12     /**
13      * Run the database seeds.
14      */
15     public function run(): void
16     {
17         DB::table('bicycles')->insert([
18             ['brand' => 'Orbea', 'model' => 'Sky'],
19             ['brand' => 'BH', 'model' => 'Gacela'],
20         ]);
21     }
22 }
```

Fijate como he creado los campos "brand" y "model" en el seeder con 2 bicicletas.

# Después de crear el seeder, ahora vamos a ejecutarlo...

Ejecución de un seeder



```
C:\laragon\www\bicycles(main)
λ php artisan db:seed --class=BicyclesTableSeeder

INFO Seeding database.
```

El seeder se ha ejecutado

El resultado de ejecutar un seeder es que, en este caso, la tabla “bicycles” de la base de datos “bicycles” se puebla con los 2 registros del seeder.

**El resultado es que se ha poblado la tabla “bicycles” en la BD “bicycles” con los 2 registros del seeder...**

**Resultado de la ejecución de un seeder**

Laragon.MySQL\bicycles\bicycles\ - HeidiSQL Portable 12.1.0.6537

Archivo Editar Buscar Consulta Herramientas Ir a Ayuda

Filtro de bases de datos Filtro de tablas Host: 127.0.0.1 Base de datos: bicycles Tabla: bicycles

**Laragon.MySQL**

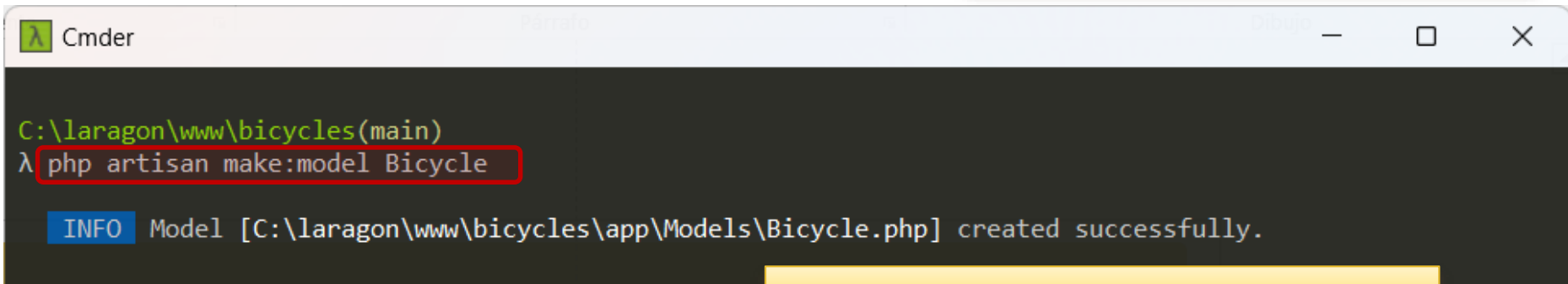
- bicycles** 192,0 KiB
  - bicycles** 16,0 KiB
  - cache 16,0 KiB
  - cache\_locks 16,0 KiB
  - failed\_jobs 16,0 KiB

bicycles.bicycles: 2 filas en total

id	brand	model	created_at	updated_at
1	Orbea	Sky	(NULL)	(NULL)
2	BH	Gacela	(NULL)	(NULL)

# Creamos el modelo Bicycle

Creación de un modelo



```
C:\laragon\www\bicycles(main)  
λ php artisan make:model Bicycle  
  
INFO Model [C:\laragon\www\bicycles\app\Models\Bicycle.php] created successfully.
```

Se ha creado el modelo Bicycle

Con el modelo creado podremos programar con orientación a objetos sin preocuparnos de queries SQL porque Eloquent ORM hará ese trabajo por nosotros.

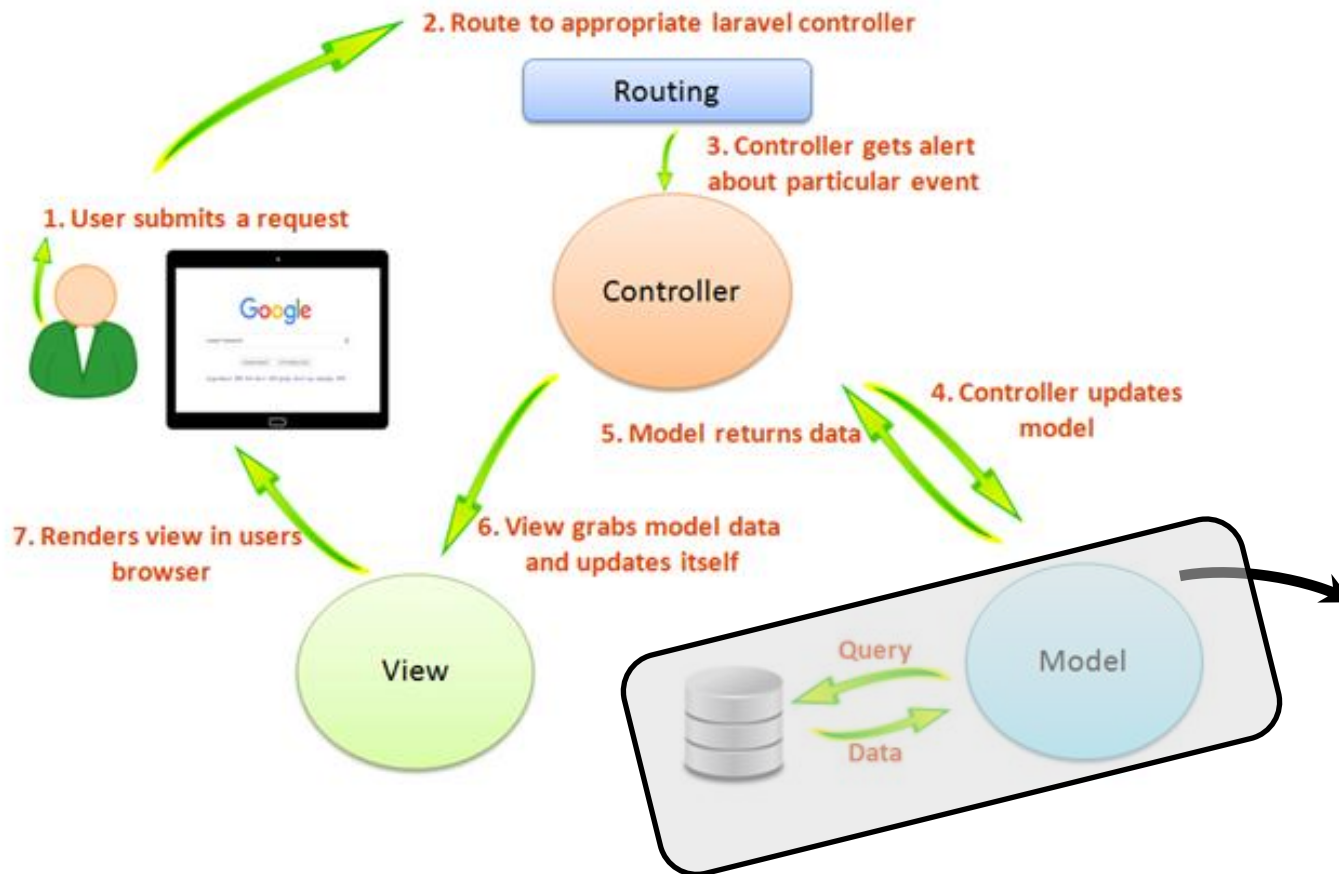
# Veamos el modelo creado...

El fichero  
app\Models\Bicycle.php

```
app > Models > Bicycle.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Bicycle extends Model
9  {
10     use HasFactory;
11
12     protected $fillable = [
13         'brand',
14         'model',
15     ];
16 }
17
```

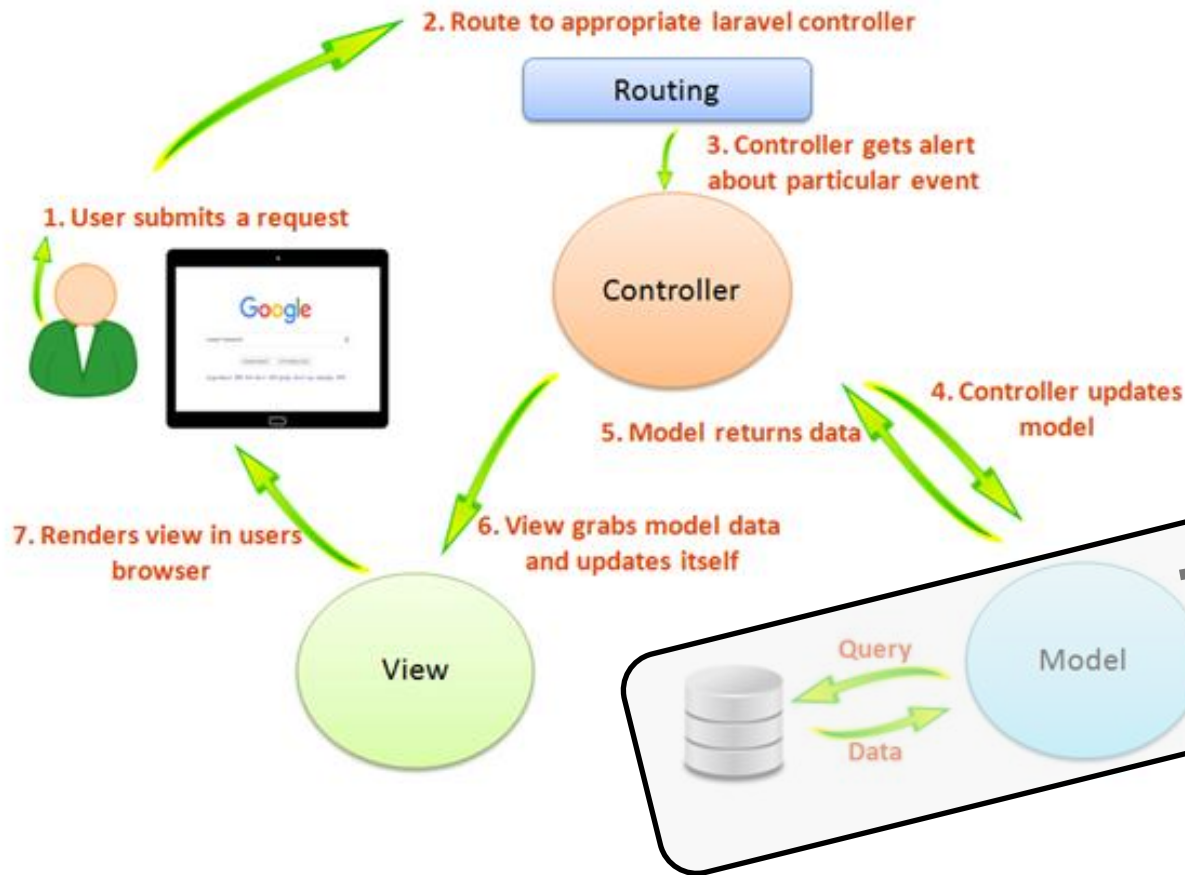
En el modelo  
se indican los  
campos.

# Enfoque 2: usando plantillas (lo que hemos hecho hasta el momento)



**Paso 1:**  
Creamos una migración y un seeder, y los hemos ejecutado con lo que ahora MySQL tiene la tabla bicycles con 2 registros

# Enfoque 2: usando plantillas (lo que hemos hecho hasta el momento)



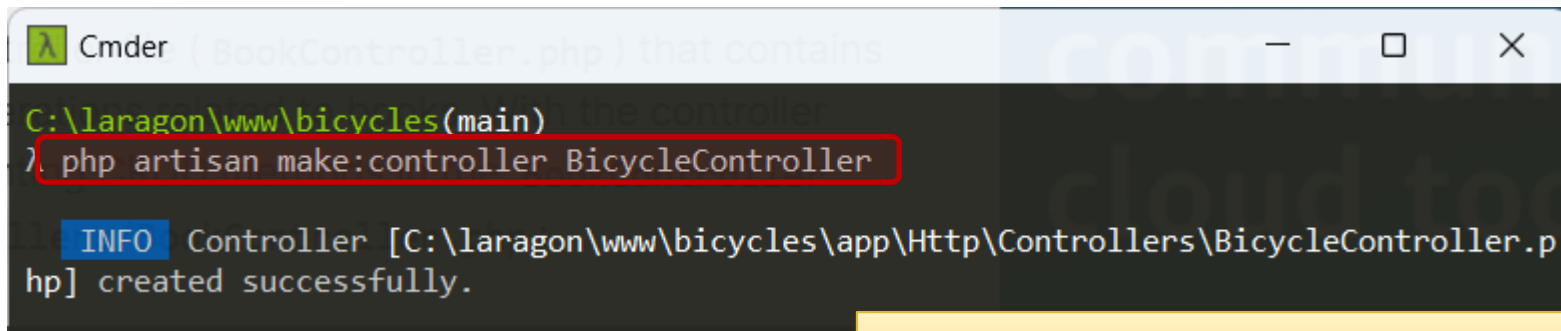
***Y ahora vamos  
a crear el  
controlador***

**Paso 1:**  
Creamos una migración y un seeder, y los hemos ejecutado con lo que ahora MySQL tiene la tabla bicycles con 2 registros



# Creamos el controlador BicycleController...

Creación de un controlador



```
C:\laragon\www\bicycles(main)  
λ php artisan make:controller BicycleController  
  
INFO Controller [C:\laragon\www\bicycles\app\Http\Controllers\BicycleController.php] created successfully.
```

Se ha creado el controlador  
BicycleController

Crear un controlador consiste en crear un fichero que contiene las funciones que forman la lógica de negocio, en este caso de “bicycles” que básicamente va a ser las funciones del CRUD de momento.

# Veamos el controlador creado...

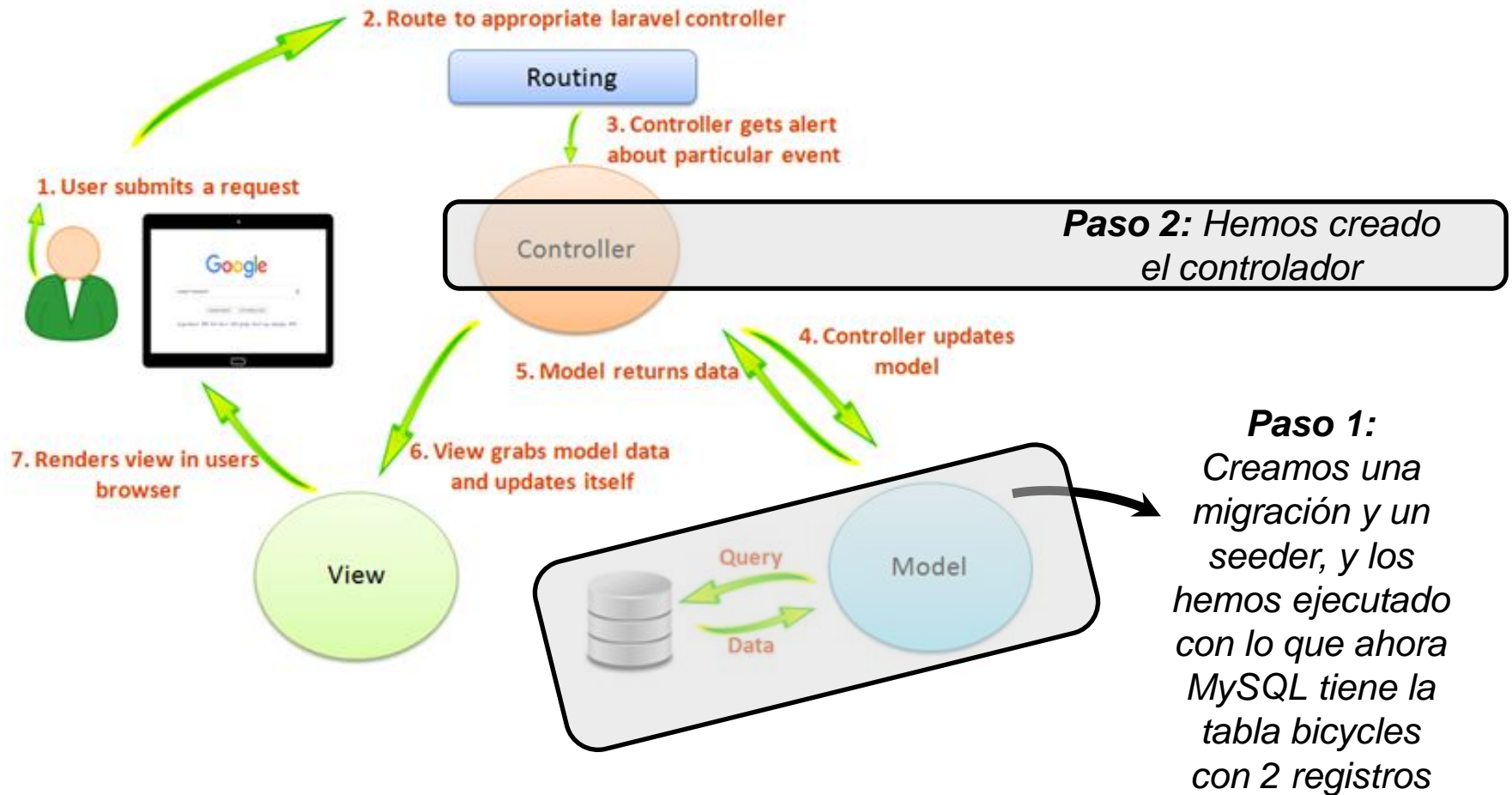
El fichero  
app\Http\Controllers\  
BicycleController.php

```
app > Http > Controllers > BicycleController.php
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  use App\Models\Bicycle;
8
9  class BicycleController extends Controller
10 {
11     public function index()
12     {
13         $bicycles = Bicycle::all();
14         return view('bicycles.index', compact('bicycles'));
15     }
16
17     public function create()
18     {
19         return view('bicycles.create');
20     }
21
22     public function store(Request $request)
23     {
24         $bicycle = new Bicycle;
25         $bicycle->brand = $request->input('brand');
26         $bicycle->model = $request->input('model');
27         $bicycle->save();
28
29         return redirect()->route('bicycles.index');
30     }
31 }
32
```

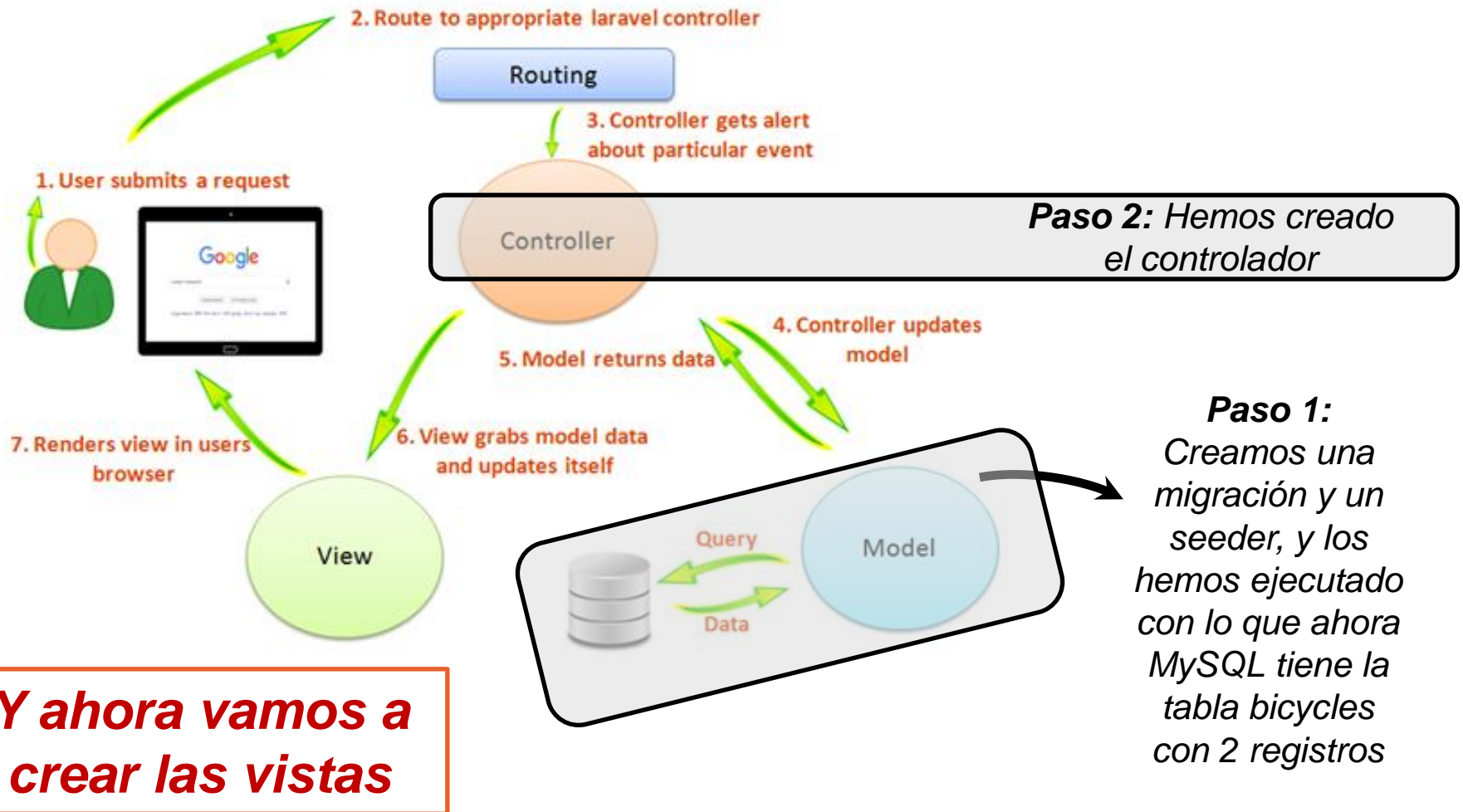
Fijate como he creado las funciones para mostrar un listado de bicicletas, crear una bicicleta con un formulario y almacenar una bicicleta.

Faltarían aún las funciones edit, update, show y delete.

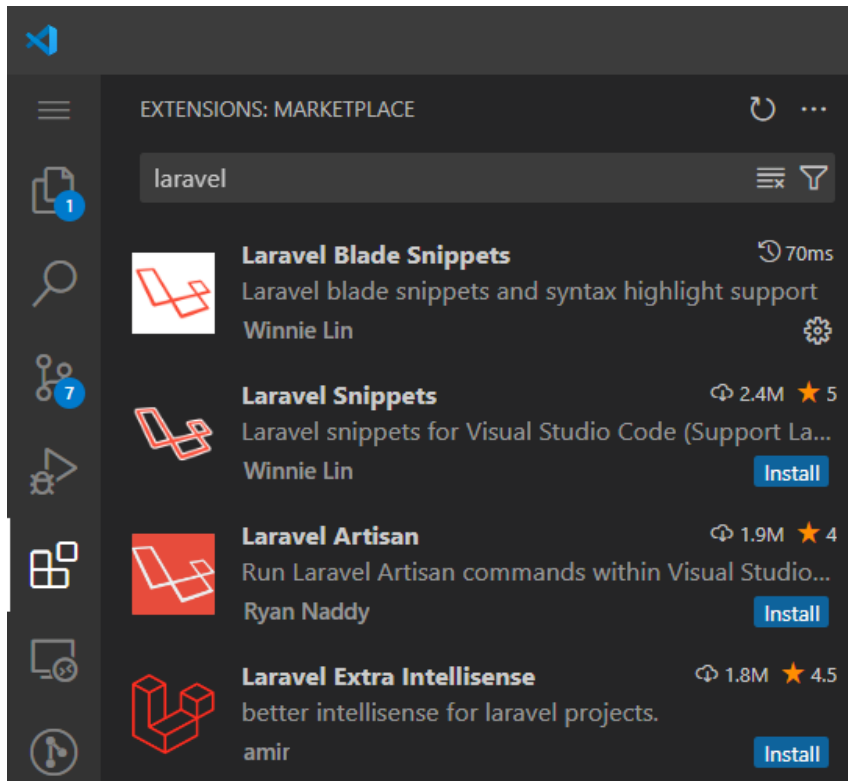
# Enfoque 2: usando plantillas (lo que hemos hecho hasta el momento)



# Enfoque 2: usando plantillas (lo que hemos hecho hasta el momento)



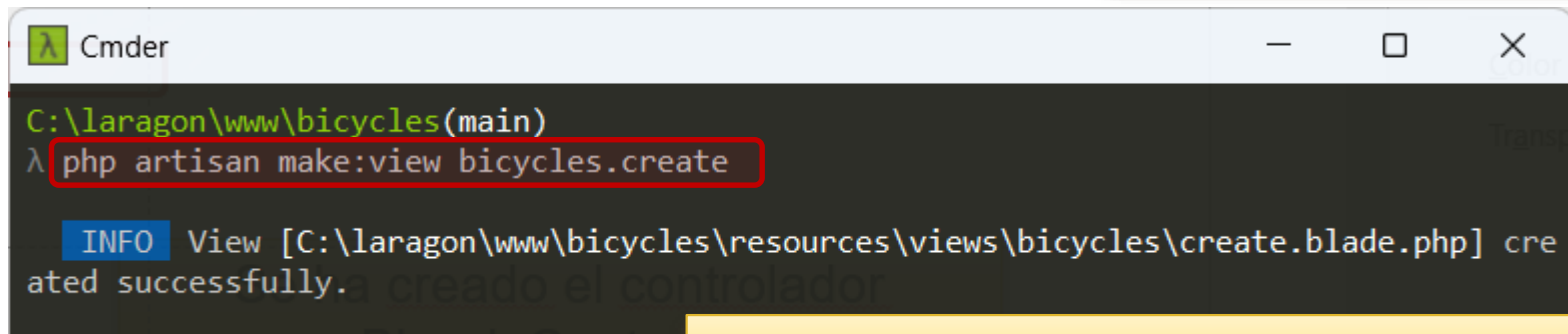
# Antes de empezar a crear vistas es interesante que instales la extensión para vistas de laravel



**Ayuda mucho a la hora de crear vistas con plantillas blade de laravel**

# Creamos la vista bicycles\create.blade.php

Creación de una vista



```
C:\laragon\www\bicycles(main)
λ php artisan make:view bicycles.create

INFO View [C:\laragon\www\bicycles\resources\views\bicycles\create.blade.php] created successfully.
```

Se ha creado la vista  
resources\views\bicycles\create.blade.php

**Crear una vista con el comando anterior realmente sólo crea el fichero en su sitio. Podríamos crearlo a mano también.**

En realidad podríamos crearlo todo a mano, pero...  
usar los comandos de laravel es más productivo.

# Veamos la vista...

El fichero resources\views\bicycles\create.blade.php

resources > views > bicycles > create.blade.php

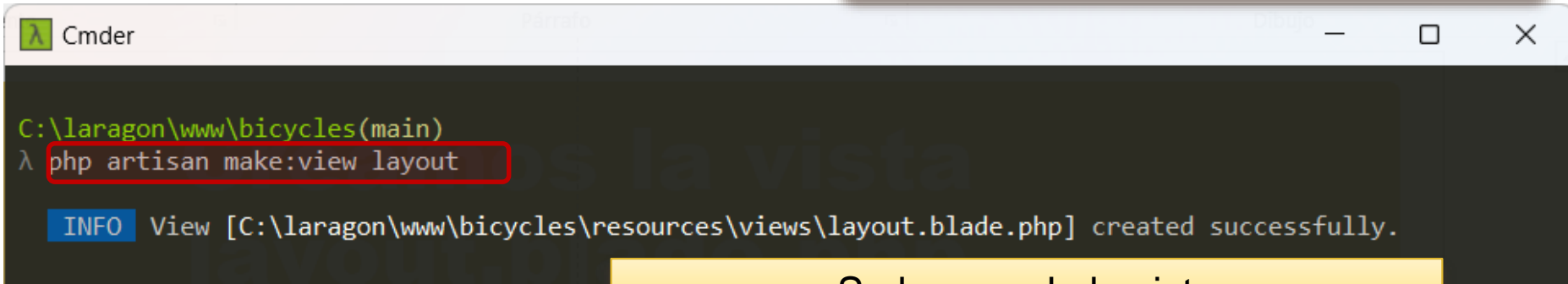
```
1 @extends('layout')
2
3 @section('content')
4     <h1>Create a New Bicycle</h1>
5     <form method="POST" action="{{ route('bicycles.store') }}">
6         @csrf
7         <div class="form-group">
8             <label for="brand">Brand</label>
9             <input type="text" name="brand" class="form-control" id="brand" placeholder="Enter bicycle brand">
10        </div>
11        <div class="form-group">
12            <label for="model">Model</label>
13            <input type="text" name="model" class="form-control" id="model" placeholder="Enter bicycle model">
14        </div>
15        <button type="submit" class="btn btn-primary">Submit</button>
16    </form>
17    <a href="{{ route('bicycles.index') }}">Back to the list</a>
18 @endsection
19
```

Realmente se trata de un formulario.

*La directiva @csrf genera un token CSRF para proteger contra la falsificación de solicitudes entre sitios.*

# Creamos la vista `layout.blade.php` que se repetiría en todas las vistas

Creación de una vista de layout



```
C:\laragon\www\bicycles(main)  
λ php artisan make:view layout  
  
INFO View [C:\laragon\www\bicycles\resources\views\layout.blade.php] created successfully.
```


Se ha creado la vista  
`resources\views\layout.blade.php`

Crear una vista con el comando anterior realmente sólo crea el fichero en su sitio. Podríamos crearlo a mano también.



# Veamos la vista layout...

El fichero resources\views\layout.blade.php



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'BICYCLES' with a directory structure: app, bootstrap, config, database, public, resources, and views. The 'resources' directory is expanded, showing 'css', 'js', 'views', and 'bicycles'. The 'views' directory is expanded, showing 'create.blade.php', 'layout.blade.php', and 'welcome.blade.php'. The 'layout.blade.php' file is selected and highlighted with a red box. The code editor on the right shows the content of 'resources > views > layout.blade.php'.

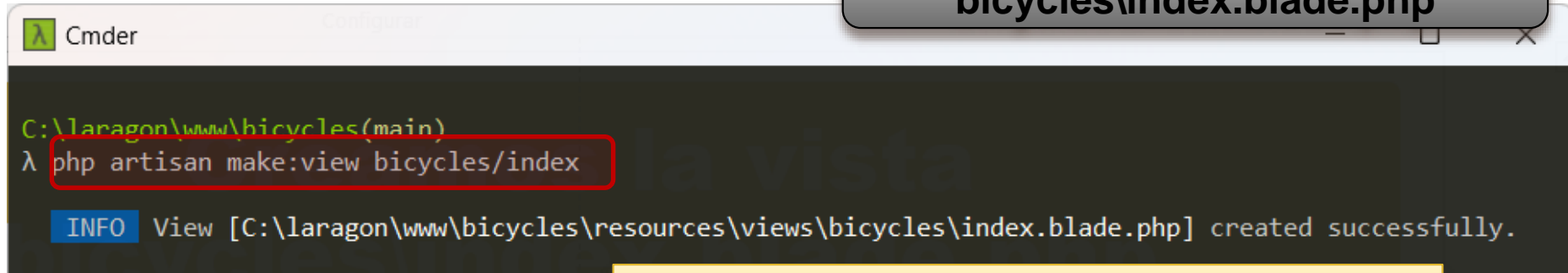
```
resources > views > layout.blade.php
1  <html>
2      <head>
3          <title>Bicycles</title>
4      </head>
5      <body>
6          <h1>Una App de Bicicletas</h1>
7
8          <div class="container">
9              @yield('content')
10         </div>
11     </body>
12 </html>
```

La intención es que este layout se repita en todas las vistas que heredan de esta, es decir, dónde se ponga la directiva `@extends('layout')`.

*Fíjate que la vista “resources/view/bicycles/create.blade.php” hereda de esta vista layout.blade.php.*

# Creamos la vista bicycles\index.blade.php

Creación de una vista de  
bicycles\index.blade.php



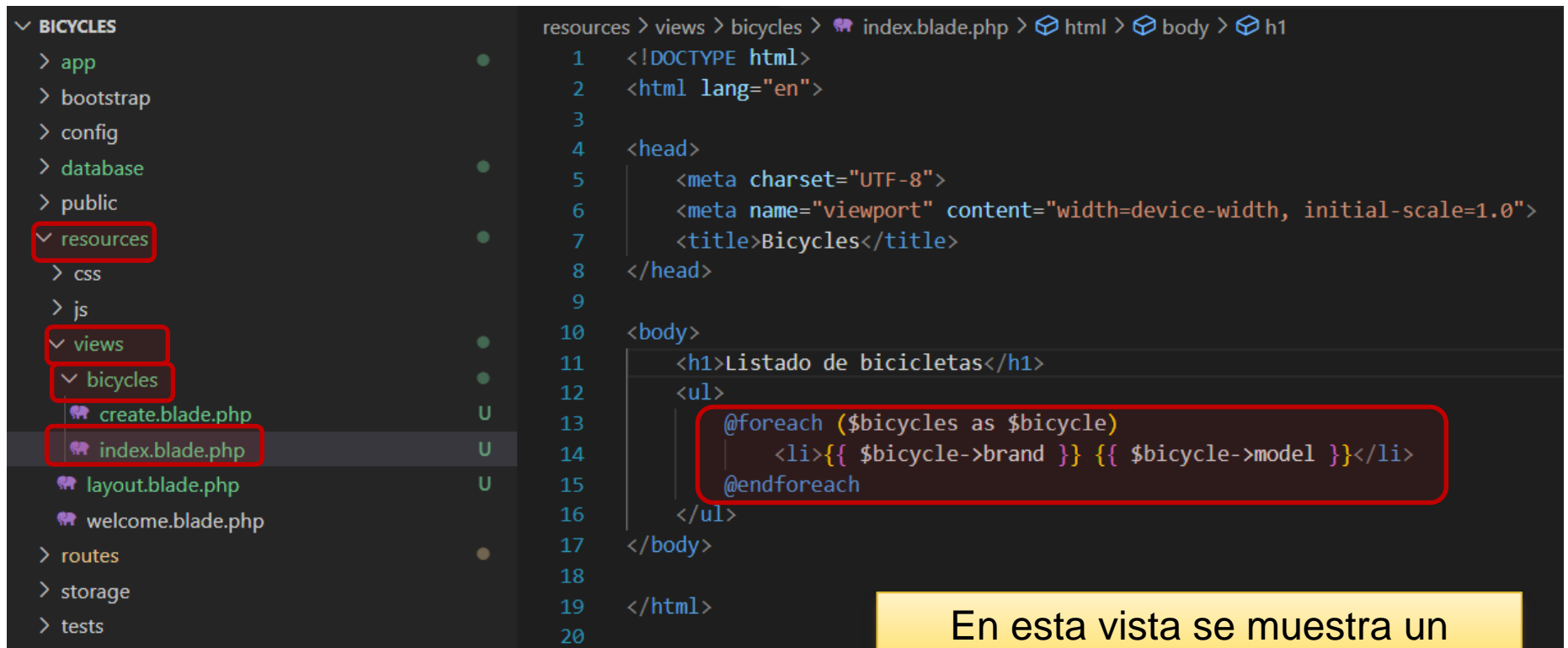
```
C:\laragon\www\bicycles(main)  
λ php artisan make:view bicycles/index  
  
INFO View [C:\laragon\www\bicycles\resources\views\bicycles\index.blade.php] created successfully.
```

Se ha creado la vista  
resources\views\bicycles\index.blade.php

Crear una vista con el comando anterior realmente sólo crea el fichero en su sitio. Podríamos crearlo a mano también.

# Veamos la vista index.blade.php

El fichero resources\views\bicycles\index.blade.php

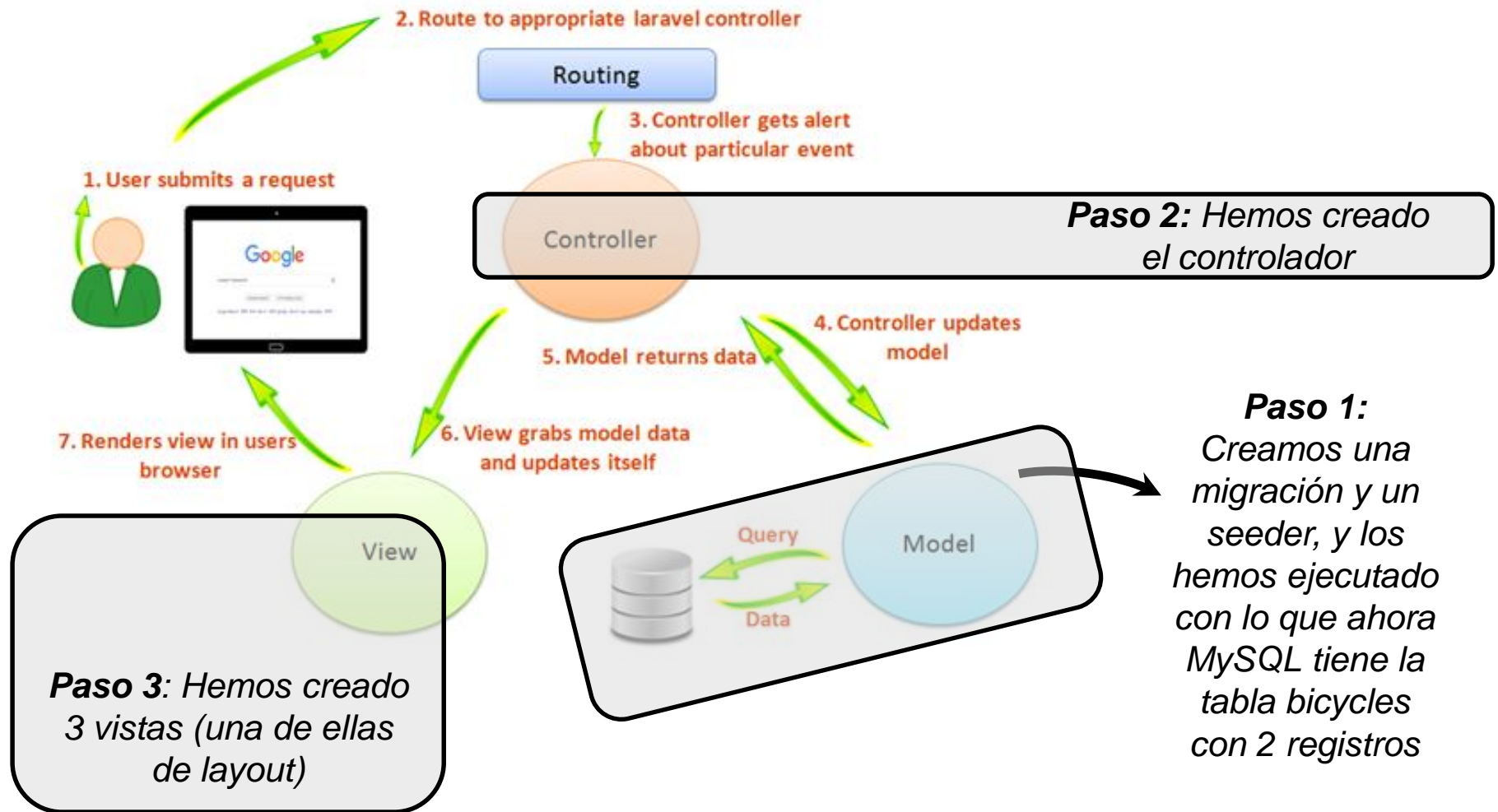


```
resources > views > bicycles > index.blade.php > html > body > h1
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Bicycles</title>
8 </head>
9
10 <body>
11     <h1>Listado de bicicletas</h1>
12     <ul>
13         @foreach ($bicycles as $bicycle)
14             <li>{{ $bicycle->brand }} {{ $bicycle->model }}</li>
15         @endforeach
16     </ul>
17 </body>
18
19 </html>
20
```

En esta vista se muestra un listado de todas las bicicletas.

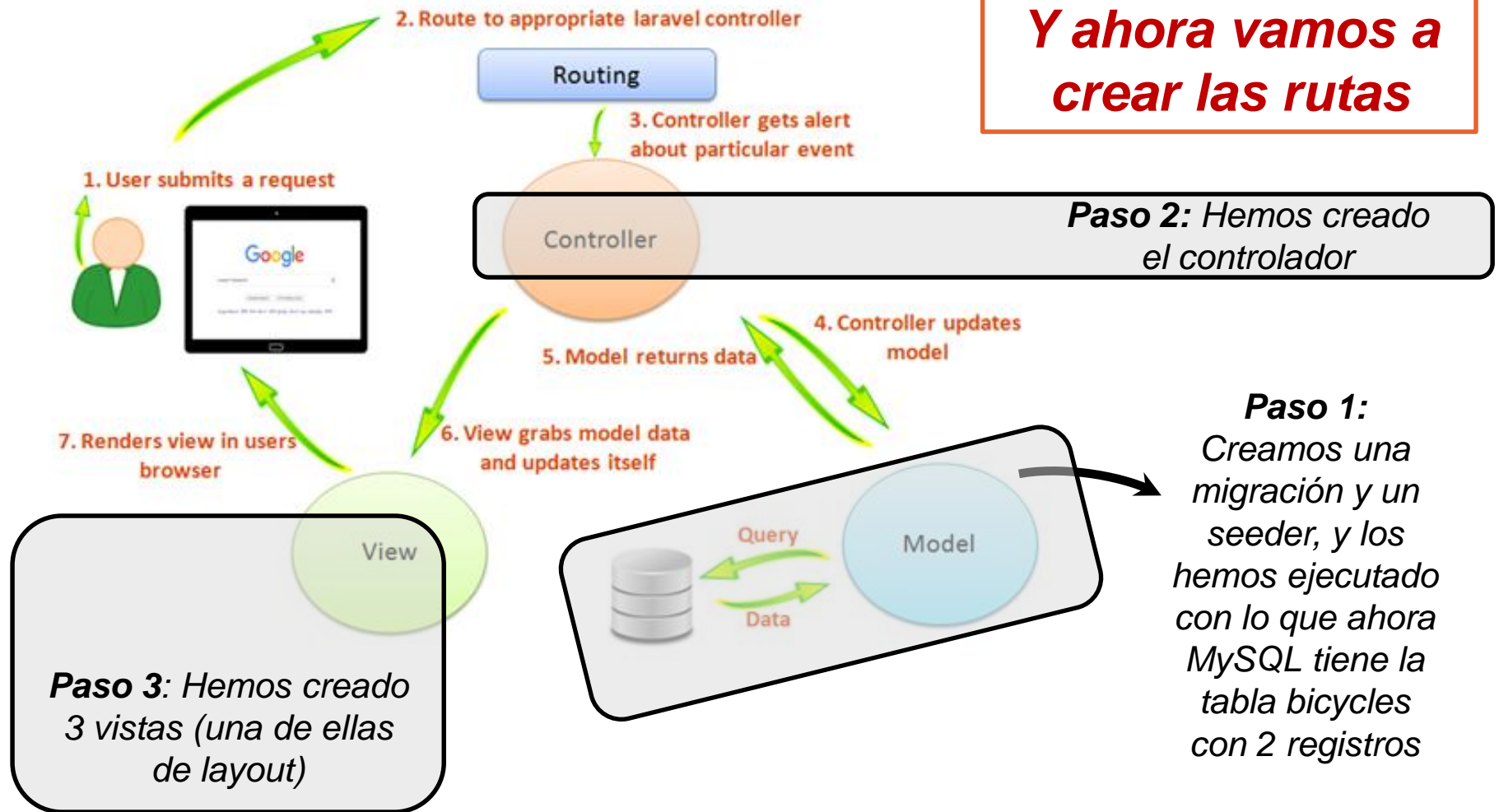
Lo más interesante aquí es la directiva @foreach

# Enfoque 2: usando plantillas (lo que hemos hecho hasta el momento)



# Enfoque 2: usando plantillas (lo que hemos hecho hasta el momento)

***Y ahora vamos a crear las rutas***



# Creamos las rutas para el controlador BicycleController

Creación de las rutas para un controlador

```
routes > web.php
You, 9 minutes ago | 1 author (You)
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4
5  Route::get('/', function () {
6      return view('welcome');
7  });
8
9  use App\Http\Controllers\BicycleController;
10
11 Route::resource('bicycles', BicycleController::class);
12
```

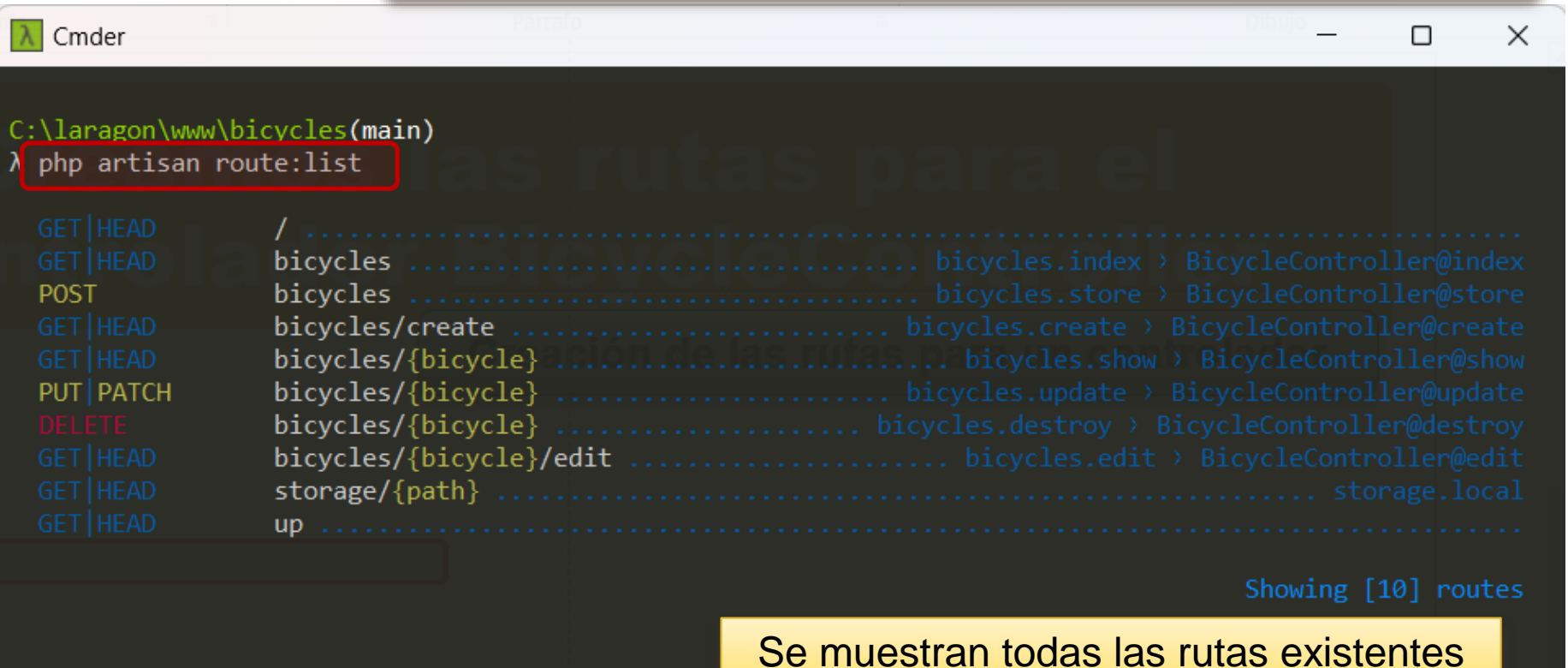
En api.php puedes implementar las rutas para una API (enfoque 1), pero ahora estamos con el enfoque 2.

Se ha creado con una sola línea todas las rutas correspondientes al CRUD correspondiente al controlador BicycleController

Lo comprobamos en la siguiente transparencia.

# Veamos el listado con todas las rutas de nuestro proyecto...

El comando para ver el listado de todas las rutas



```
C:\laragon\www\bicycles(main)
λ php artisan route:list

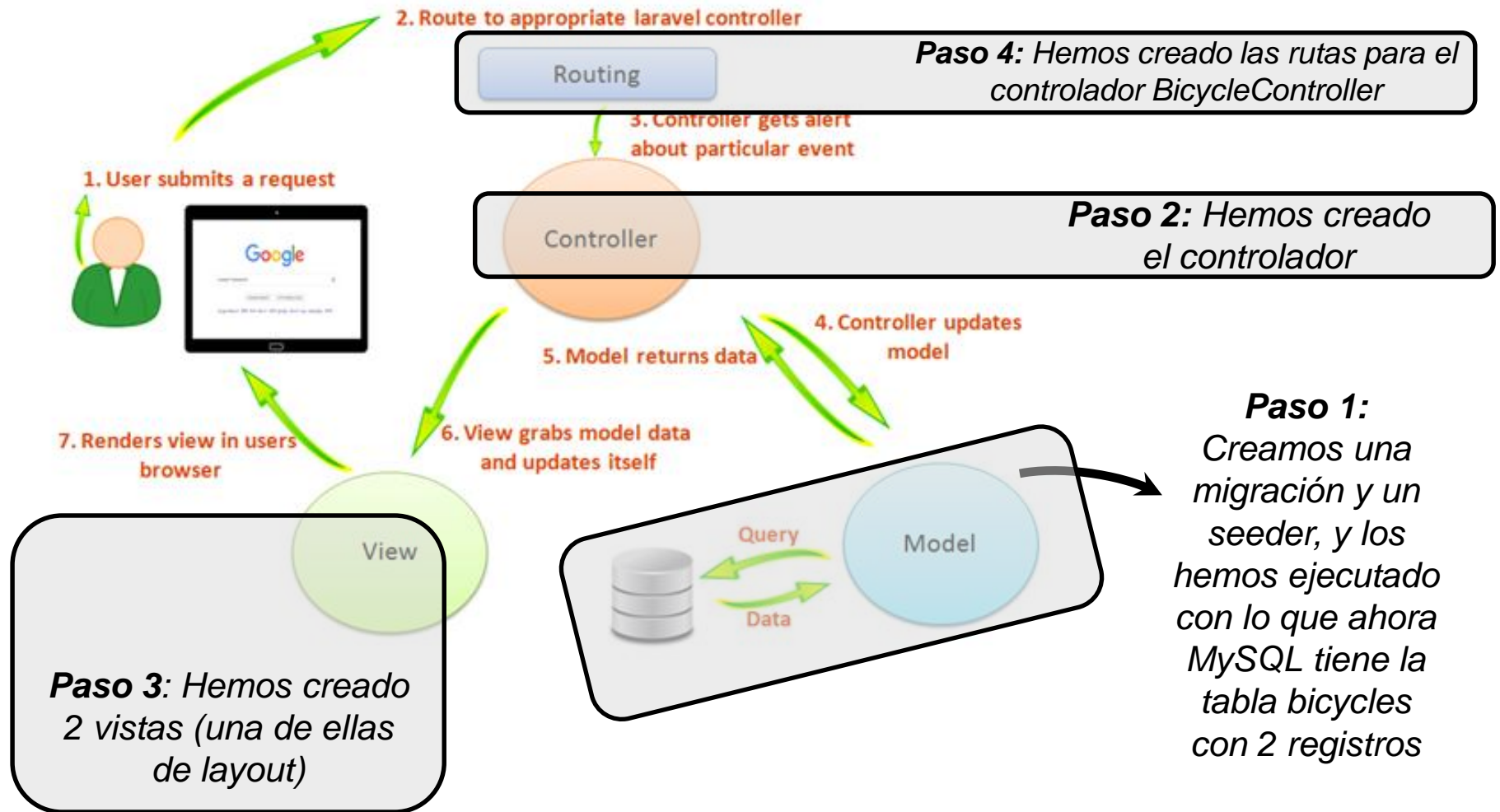
GET|HEAD / ..... bicycles.index > BicycleController@index
GET|HEAD bicycles ..... bicycles.store > BicycleController@store
POST bicycles ..... bicycles.create > BicycleController@create
GET|HEAD bicycles/create ..... bicycles.show > BicycleController@show
GET|HEAD bicycles/{bicycle} ..... bicycles.update > BicycleController@update
PUT|PATCH bicycles/{bicycle} ..... bicycles.destroy > BicycleController@destroy
DELETE bicycles/{bicycle} ..... bicycles.edit > BicycleController@edit
GET|HEAD bicycles/{bicycle}/edit ..... storage.local
GET|HEAD storage/{path} ..... up
GET|HEAD up .....

Showing [10] routes
```

Se muestran todas las rutas existentes ahora mismo en routes/web.php.

*La que nos interesa ahora mismo es GET bicycles/create*

# Enfoque 2: usando plantillas (lo que hemos hecho hasta el momento)





# ¡FUNCIONA!

La ruta para la vista  
bicycles/create.blade.php es  
`http://localhost:8000/bicycles/create`

A screenshot of a web browser window. The address bar shows 'localhost:8000/bicycles/create' highlighted with a red box. The browser's bookmark bar contains 'Import bookmarks...', 'Comenzar a usar Firefox', 'React App', and 'Cómo crear las accion...'. The page content includes the heading 'Una App de Bicicletas', the subheading 'Create a New Bicycle', a form with 'Brand' and 'Model' input fields, a 'Submit' button, and a 'Back to the list' link. Yellow arrows point from text boxes to specific elements on the page: one from 'Esto es por la vista layout.blade.php' to the heading, one from 'Prueba a crear una nueva bici. ¡Funciona!' to the 'Submit' button, and one from 'Si haces click aquí irás a la vista bicycles/index.blade.php' to the 'Back to the list' link.

← → ↻ 🛡️ 📄 localhost:8000/bicycles/create ☆ 🛡️ 👤 🏠 ⚙️ 📌 ≡

🔗 Import bookmarks... 🦊 Comenzar a usar Firefox ⚙️ React App ➤ \_ Cómo crear las accion... >> 📁 Other Bookmarks

## Una App de Bicicletas

### Create a New Bicycle

Brand

Model

[Back to the list](#)

Esto es por la vista  
layout.blade.php

Prueba a crear una nueva bici. ¡Funciona!

Si haces click aquí irás a la  
vista bicycles/index.blade.php

# Conclusiones

## ¿Qué hemos aprendido?

- Simplemente hemos comenzado a hacer un CRUD usando plantillas de blade.

## Próximos pasos...

- A continuación deberías terminar el CRUD.
  - La serie de vídeos en youtube de Aprendible (<https://www.youtube.com/channel/UC-R0zZjpkeoLHPxVTHoIVHw/videos>) está muy bien.
  - Échale un ojo al siguiente repositorio: <https://github.com/savanihd/Laravel-11-CRUD-Operation>
  - Aprende más sobre blade: <https://laravel.com/docs/11.x/blade#building-layouts>
-