

CICLO: DESARROLLO APLICACIONES WEB-DISTANCIA
MÓDULO: DESARROLLO DE APLICACIONES WEB EN ENTORNO SERVIDOR
TAREA: DWES-07
ALUMNO: DAVID MEDINA GARCIA

-Realizar una instalación de Laravel (versión 10 u 11), creando un proyecto llamado 'tarea7' mostrando y explicando en un documento *.pdf las capturas de pantalla en las que se detalle el proceso.

Vamos a instalar **Laravel 11**.

Para instalar **Laravel** en nuestro equipo debemos tener instalados ciertos programas y recursos antes:

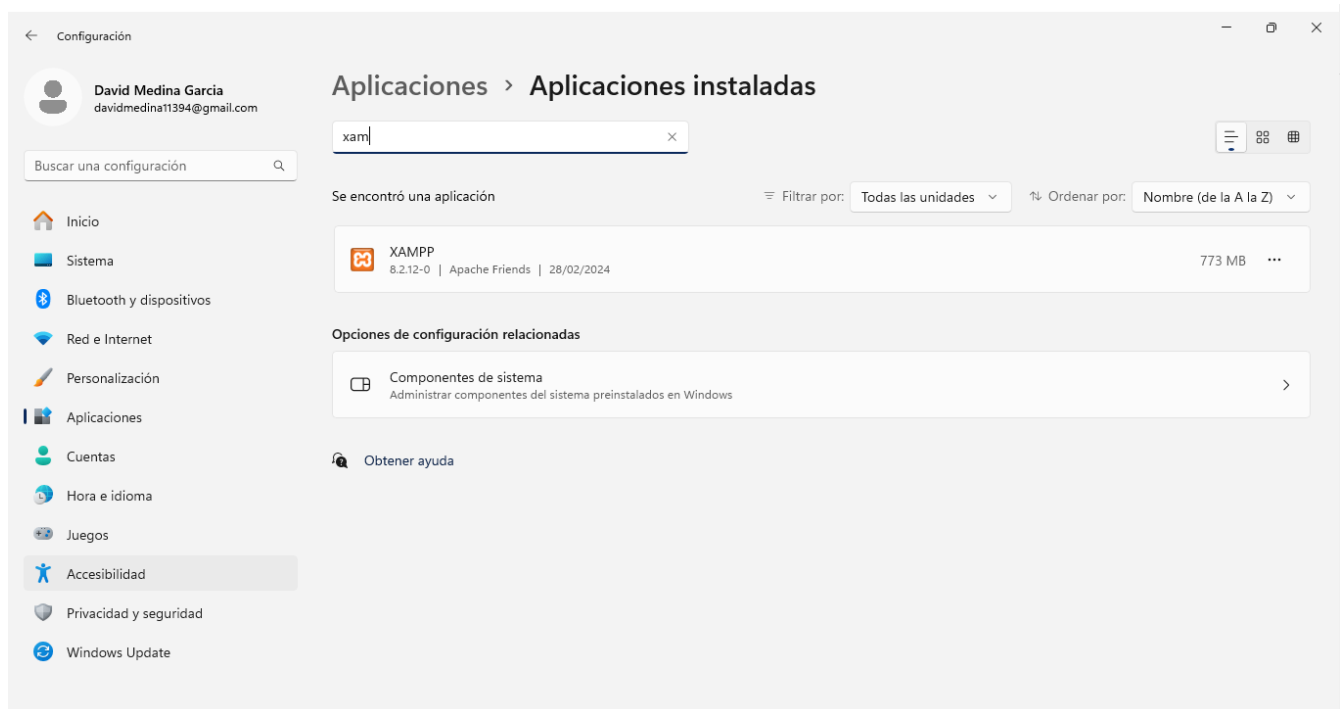
- PHP (Versiones compatibles con **Laravel 11**: 8.2/8.3)
- Composer
- Git (opcional)
- Node.js (opcional)

Aunque **Git** y **Node** son recomendables e incluso necesarios dependiendo las características del proyecto, para una tarea “simple” como esta no será necesario instalarlos.

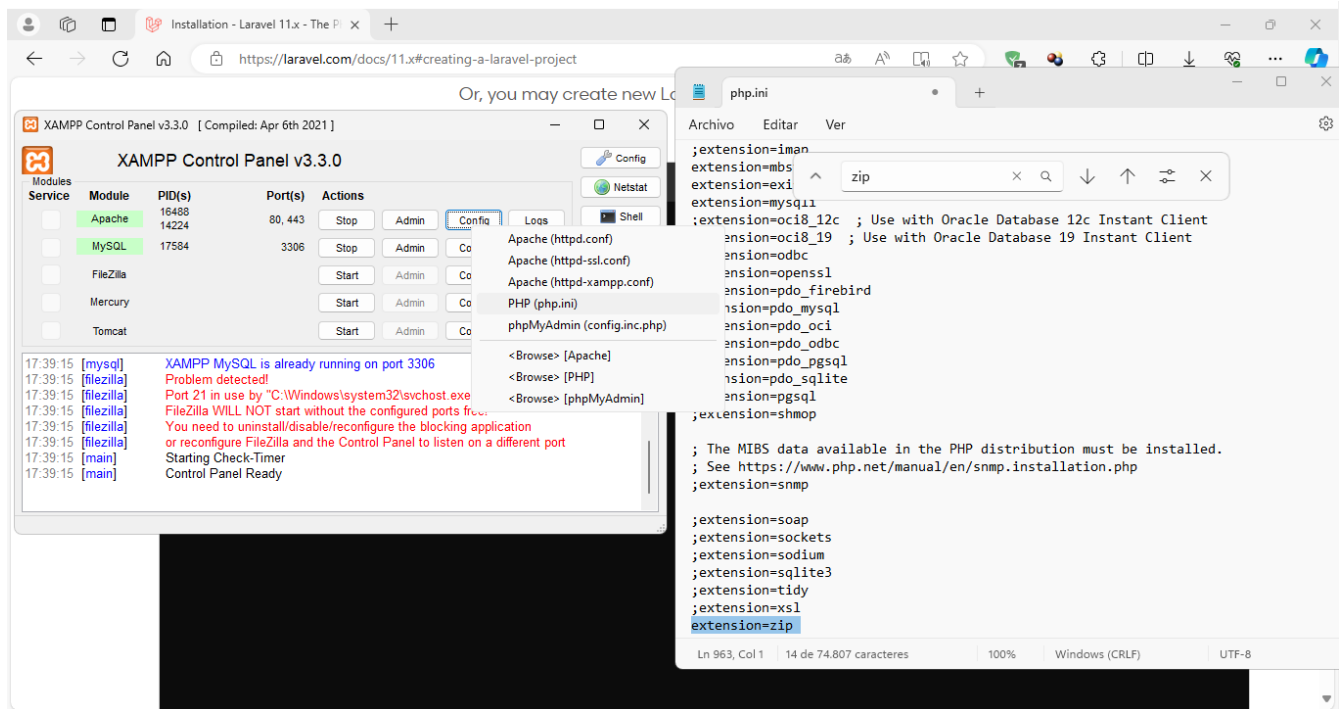
Además como conectaremos el proyecto a una base de datos necesitaremos un **gestor de bases de datos**.

Para la instalación de **PHP** y el **gestor de base de datos** usaremos **XAMPP** que ya tenemos previamente instalado de las practicas anteriores.

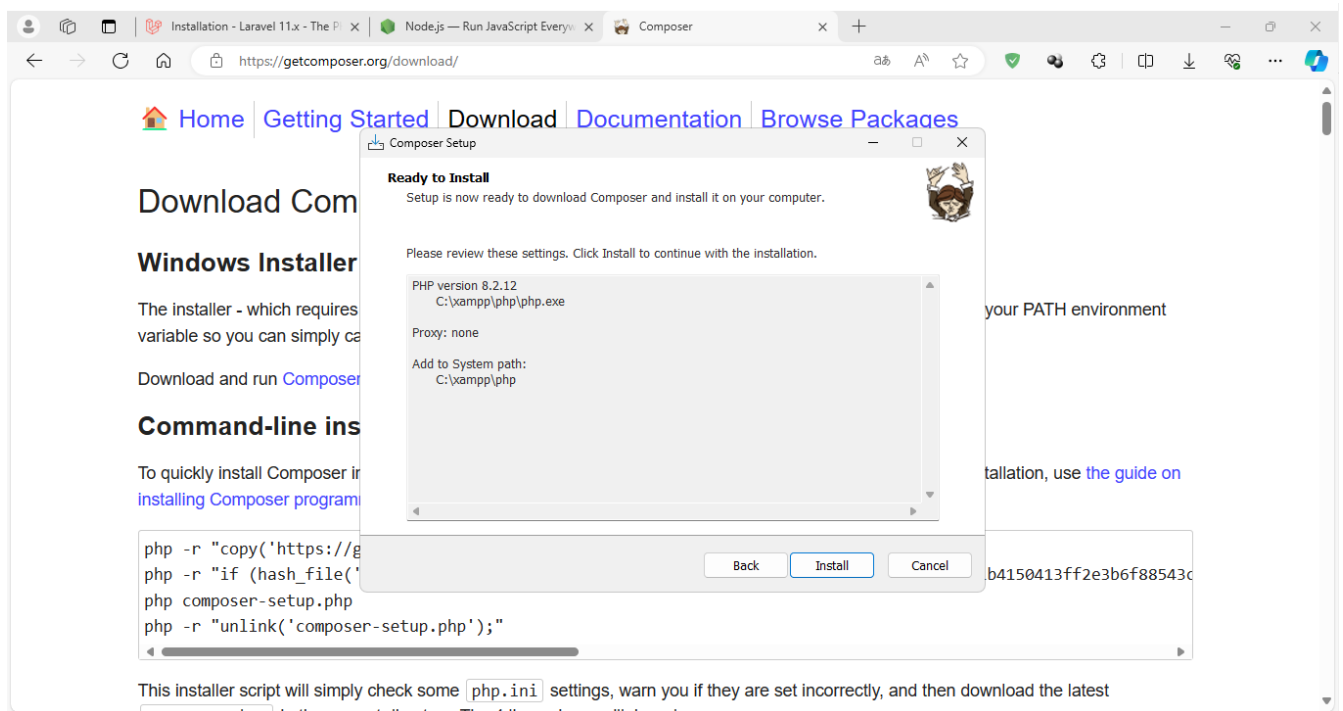
Si no está instalado, desde la página de **xampp** nos descargamos el instalador y lo ejecutamos.



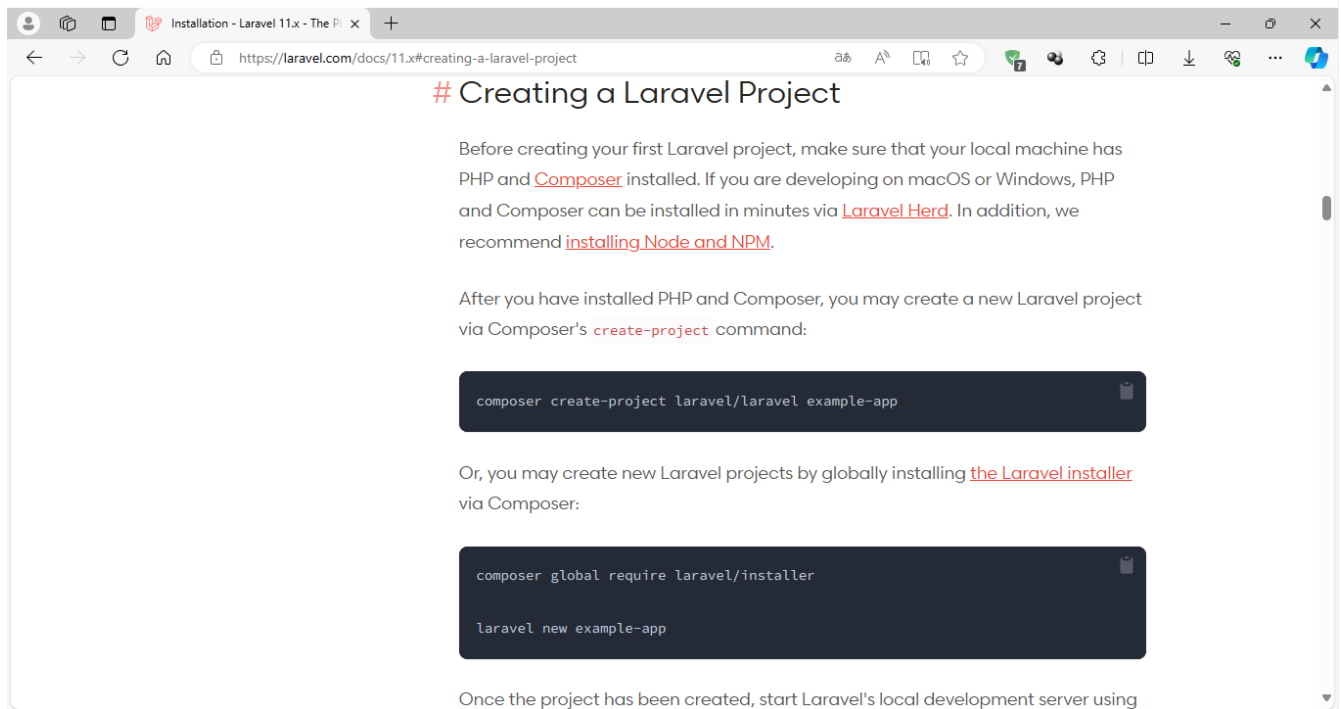
Abrimos **XAMPP** y arrancamos **Apache**. Abrimos el archivo de configuración de **php (php.ini)**, y descomentamos la **extensión=zip** para que pueda tratar con estos archivos. Reiniciamos **Apache** para aplicar los cambios.



A continuación instalamos **Composer**. Lo descargamos desde su página, iniciamos el instalador y seguimos los pasos (darle a next).



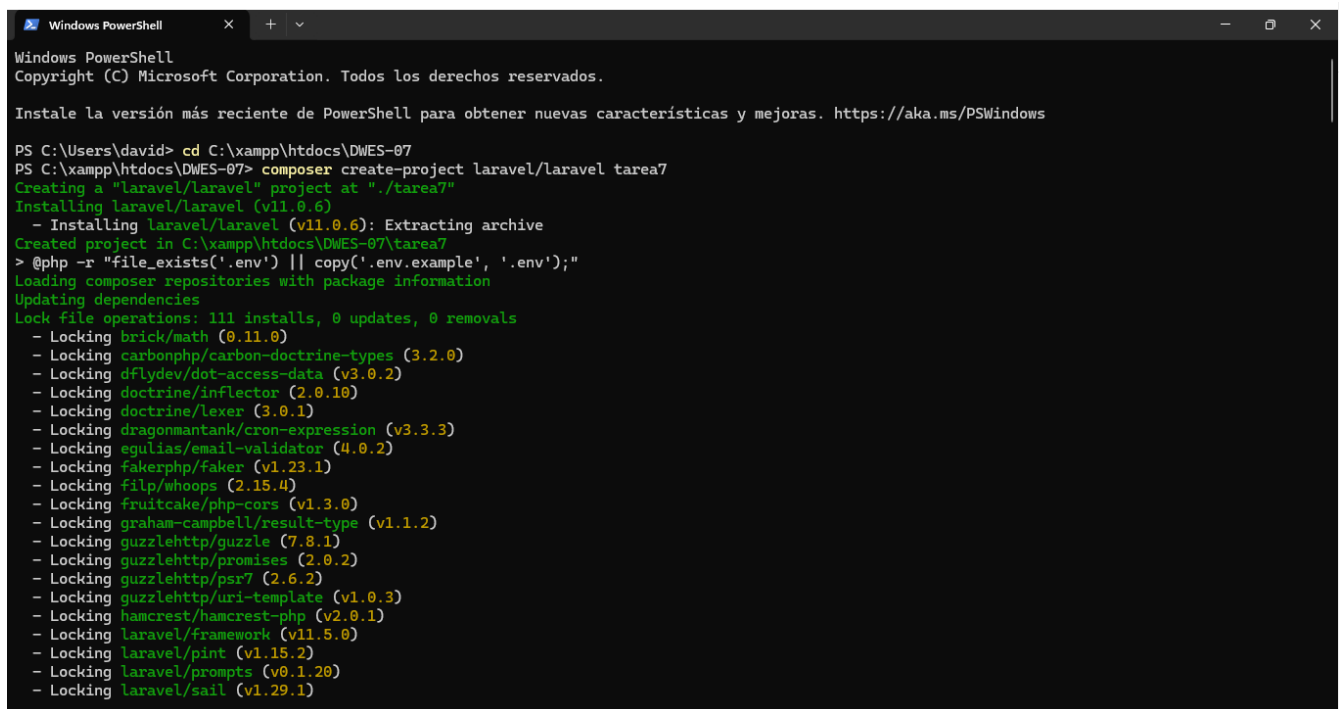
A continuación nos vamos a la página de instalación de **Laravel** al apartado #Creating a Laravel Project.



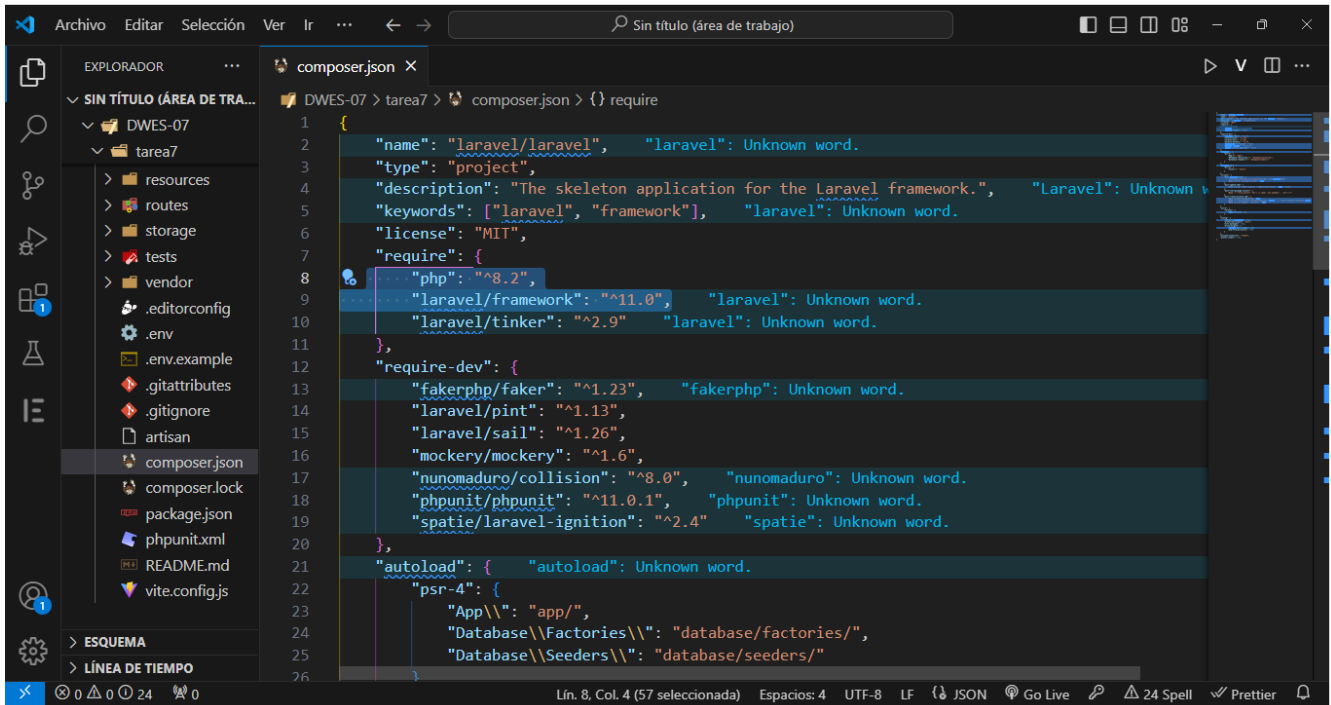
Ahora abrimos un terminal en el equipo y nos situamos en el directorio donde irá nuestro proyecto con **Laravel**.

En mi caso es: **C:\xampp\htdocs\DWES-07**

Una vez situados tenemos que introducir el siguiente comando en el terminal (“tarea7” es el nombre del proyecto): **composer create-project laravel/laravel tarea7**

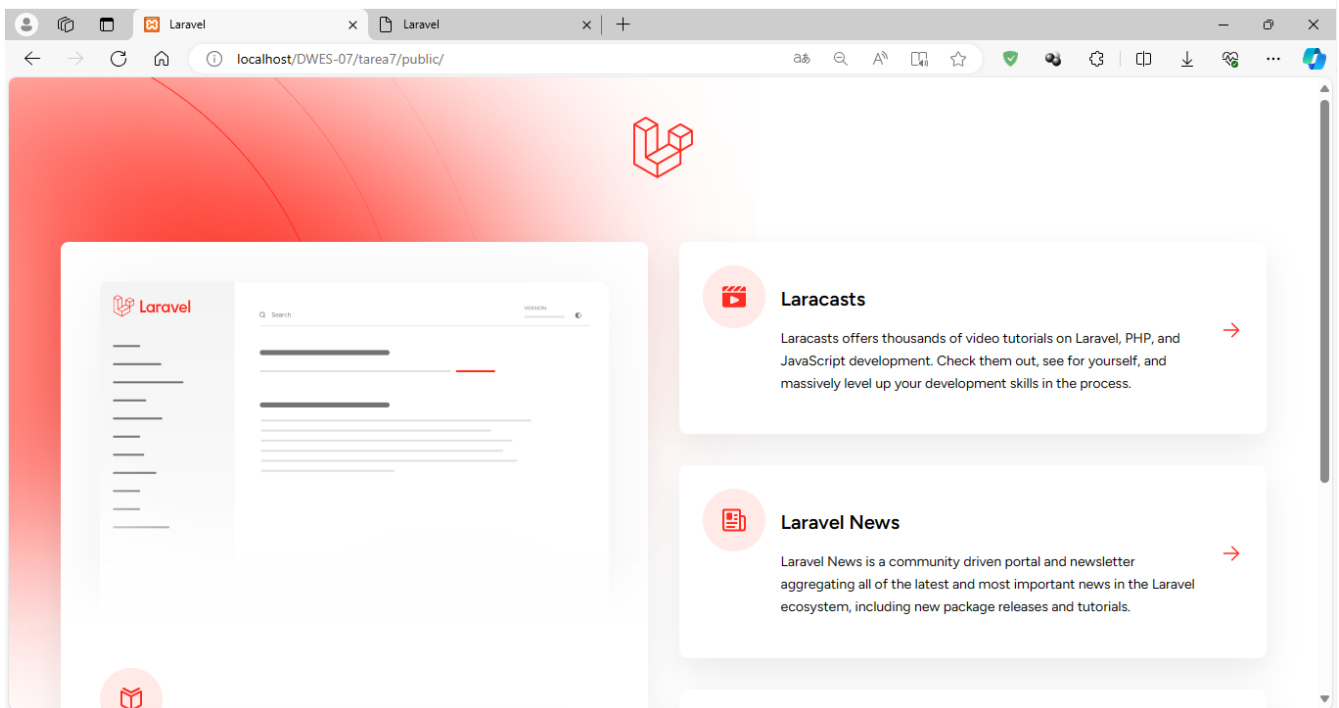


Si abrimos el archivo **composer.json** en un IDE, podemos ver las versiones de **PHP** y **Laravel** que tenemos instalados.



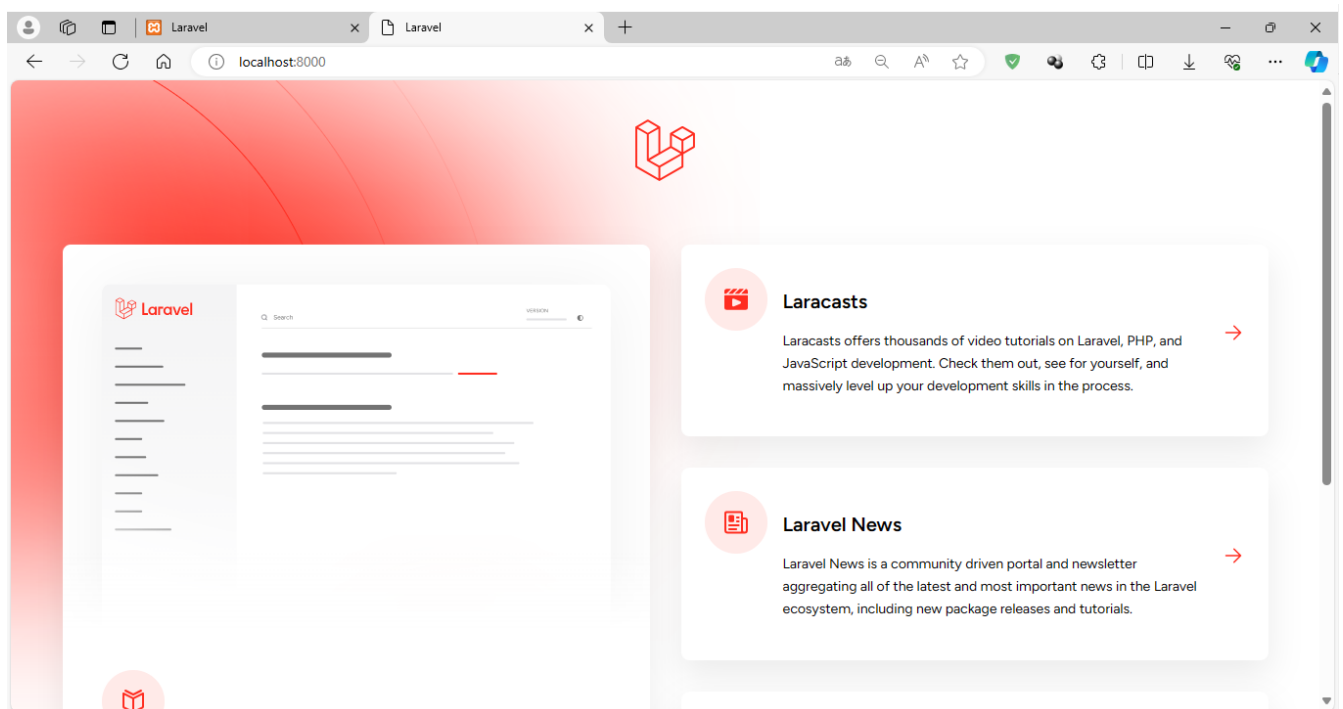
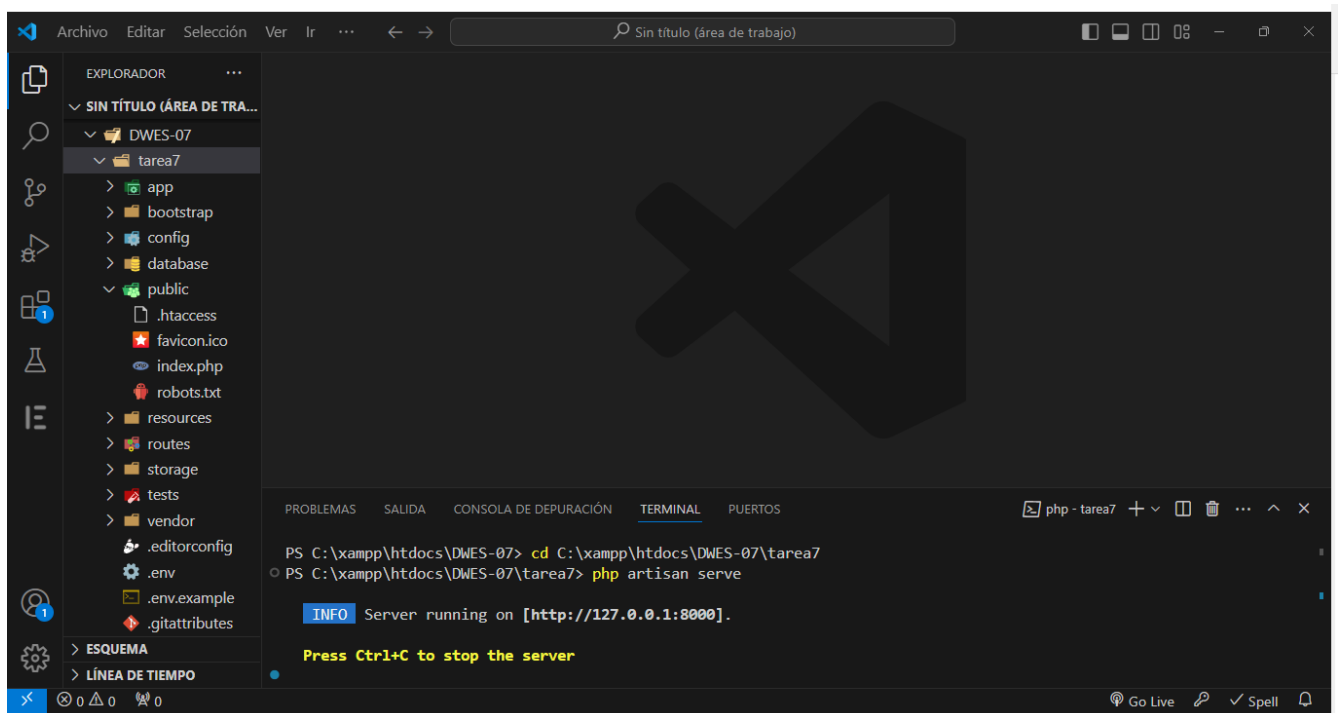
```
1 {
2     "name": "laravel/laravel", "laravel": Unknown word.
3     "type": "project",
4     "description": "The skeleton application for the Laravel framework.", "laravel": Unknown word.
5     "keywords": ["laravel", "framework"], "laravel": Unknown word.
6     "license": "MIT",
7     "require": {
8         "php": "^8.2",
9         "laravel/framework": "^11.0", "laravel": Unknown word.
10        "laravel/tinker": "^2.9" "laravel": Unknown word.
11    },
12    "require-dev": {
13        "fakerphp/faker": "^1.23", "fakerphp": Unknown word.
14        "laravel/pint": "^1.13",
15        "laravel/sail": "^1.26",
16        "mockery/mockery": "^1.6",
17        "nunomaduro/collision": "^8.0", "nunomaduro": Unknown word.
18        "phpunit/phpunit": "^11.0.1", "phpunit": Unknown word.
19        "spatie/laravel-ignition": "^2.4" "spatie": Unknown word.
20    },
21    "autoload": { "autoload": Unknown word.
22        "psr-4": {
23            "App\\": "app/",
24            "Database\\Factories\\": "database/factories/",
25            "Database\\Seeders\\": "database/seeders/"
26        }
27    }
```

Ahora si abrimos un navegador y vamos a la dirección **http://localhost/DWES-07/tarea7/public/** o a la dirección **http://localhost/DWES-07/tarea7/public/index.php** abriremos la página por defecto de **Laravel**.



Otra manera de abrir esa página es situarnos en la carpeta del proyecto (tarea7) e introducir el siguiente comando: **php artisan serve**.

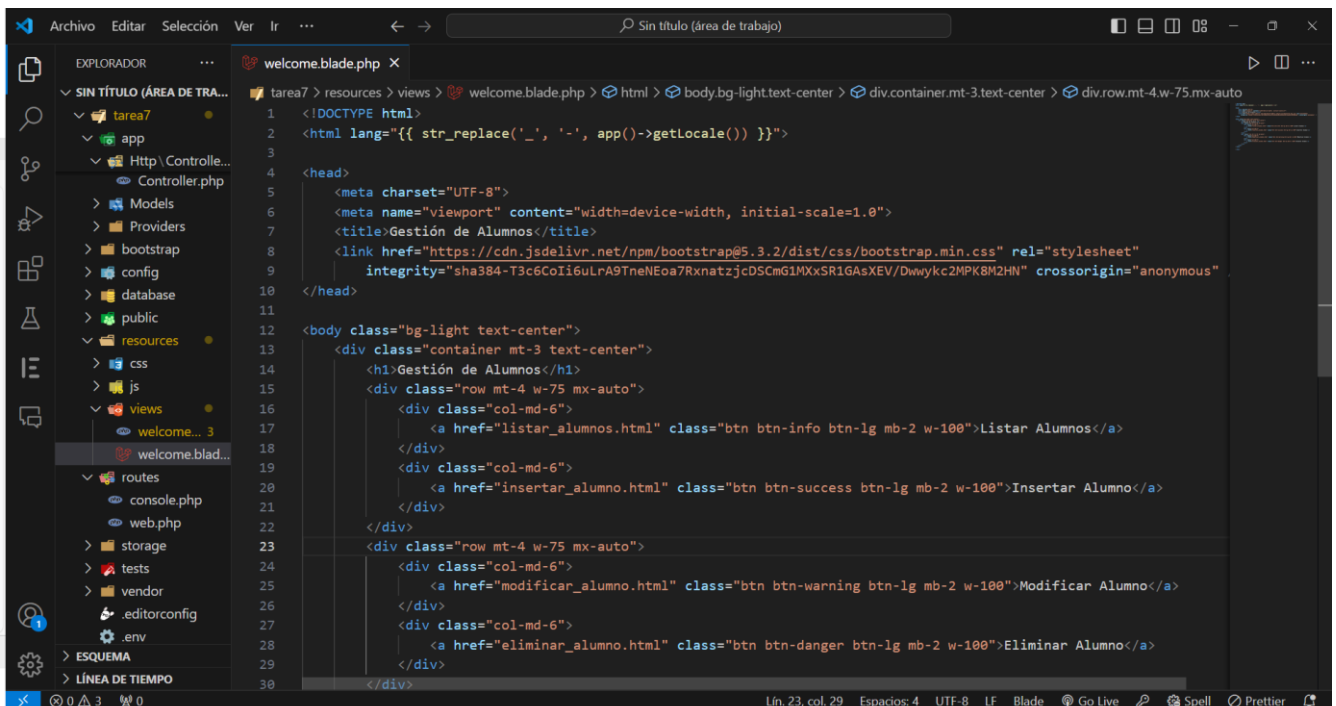
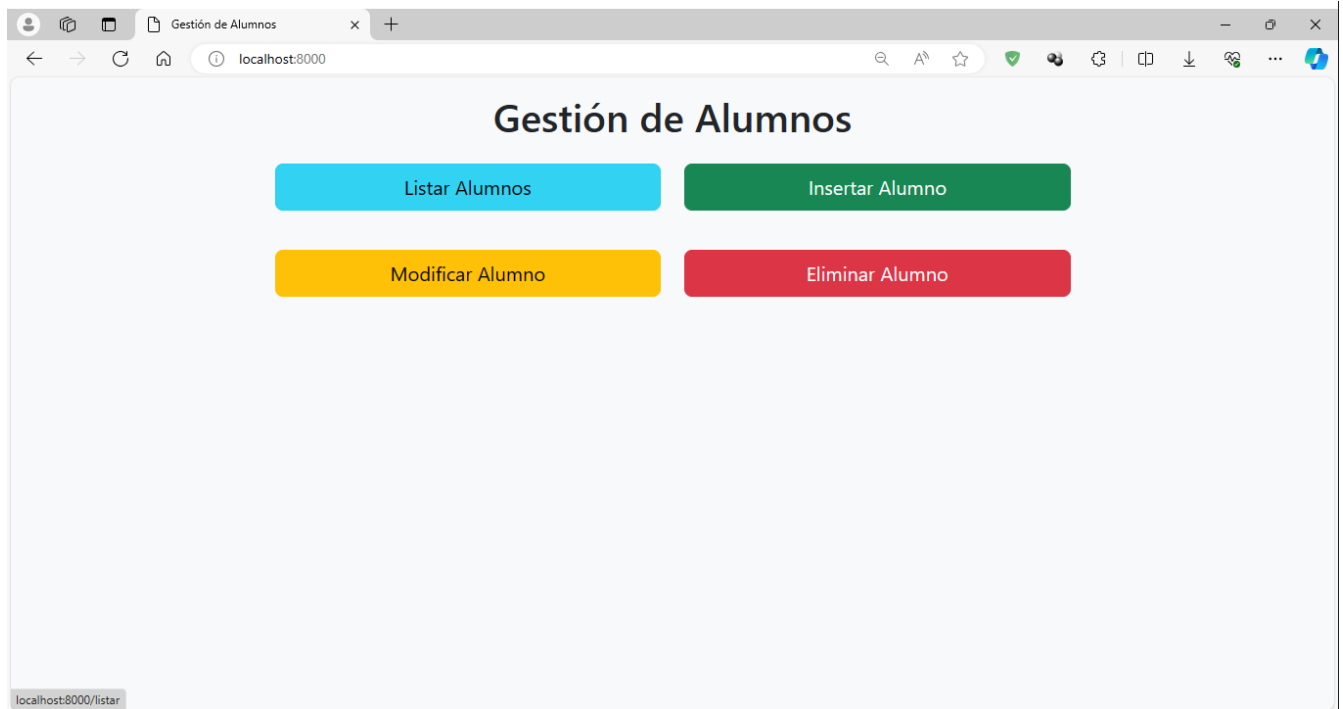
Con esto activamos un servidor local y si en un navegador nos vamos a **localhost:8000** también abriremos la página por defecto.



-Modificar la vista **welcome** que viene por defecto con un mensaje personalizado.

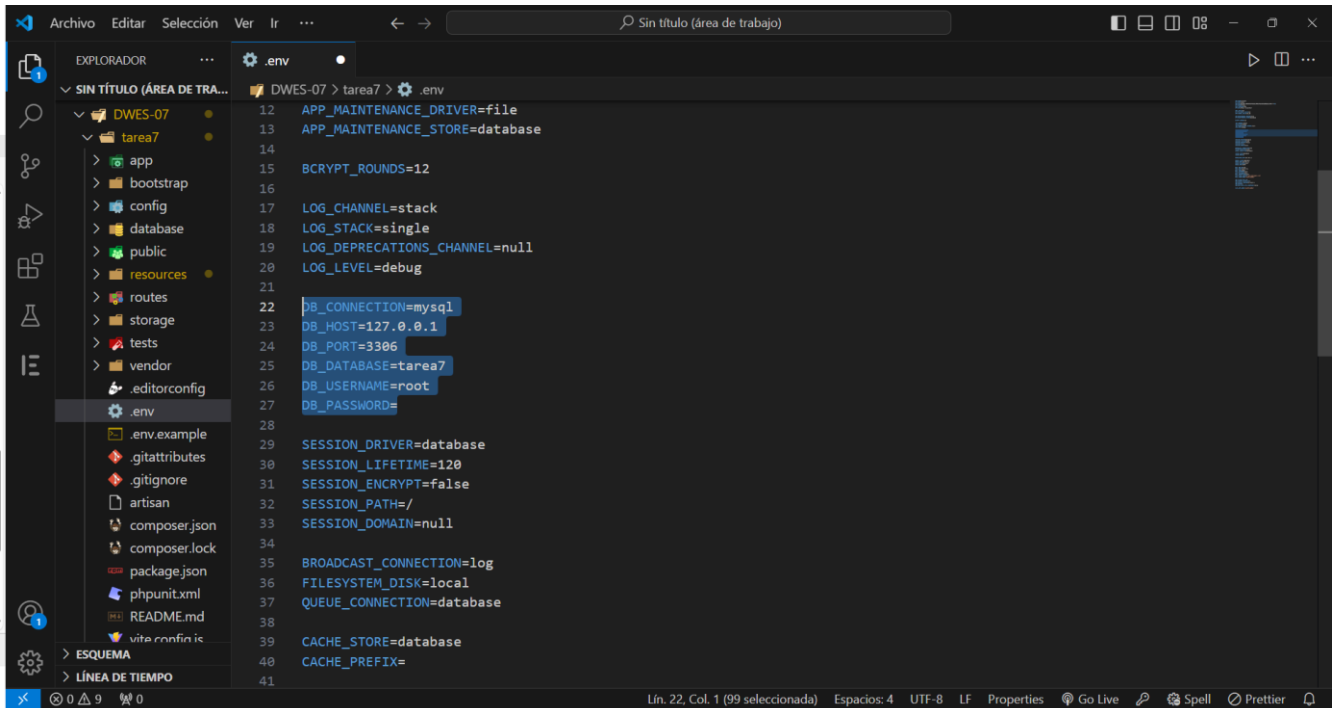
Para modificar la vista **welcome** se debe modificar el archivo **views/welcome.blade.php**

La modificación que he hecho ha sido sustituir al código que venía por una página a través de la cual podemos hacer diversas acciones sobre una base de datos con la que vamos a trabajar en esta unidad.



-Conectar el proyecto con la base de datos de la tarea anterior e indicar como se ha realizado

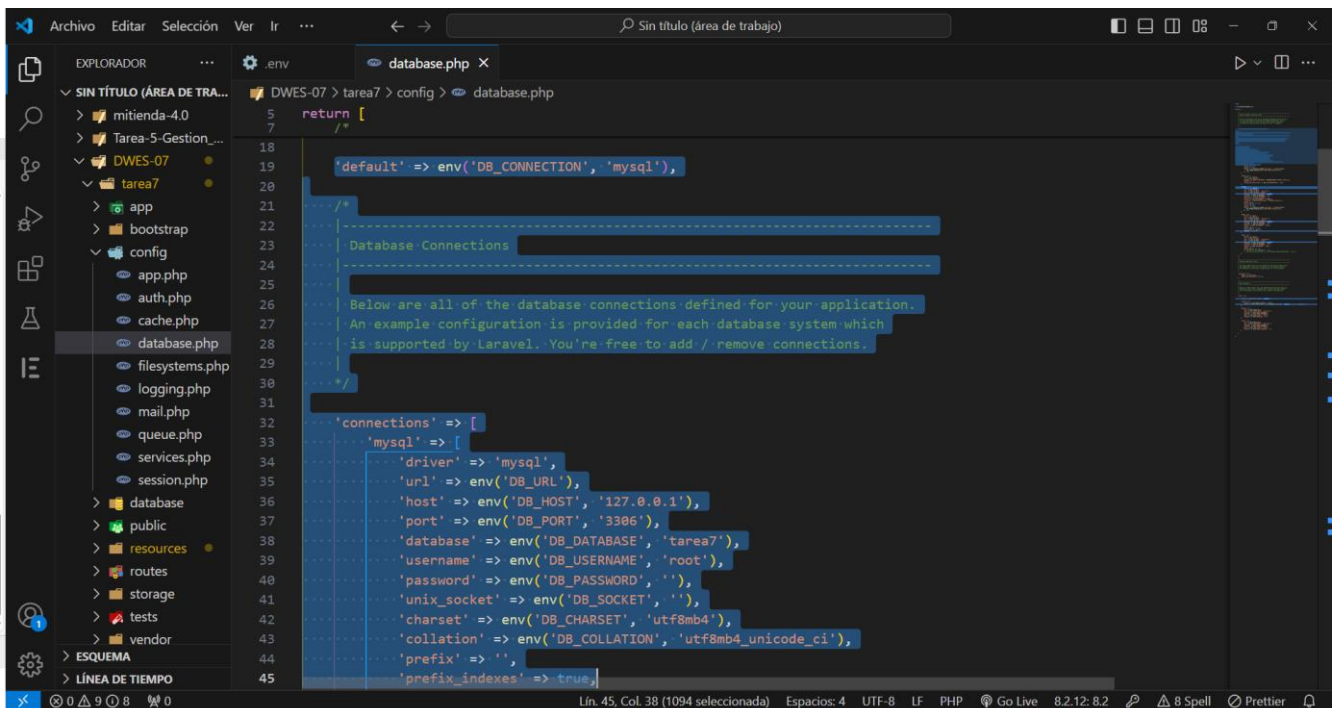
Para conectar el proyecto con nuestra base de datos tenemos que abrir el archivo `.env` y modificar las siguientes líneas con los datos que necesitamos.



The screenshot shows the Visual Studio Code editor with the `.env` file open. The file contains configuration for Laravel, including database connection details. The following lines are highlighted in blue:

```
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_DEPRECATIONS_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=tarea7
26 DB_USERNAME=root
27 DB_PASSWORD=
28
29 SESSION_DRIVER=database
30 SESSION_LIFETIME=120
31 SESSION_ENCRYPT=false
32 SESSION_PATH=/
33 SESSION_DOMAIN=null
34
35 BROADCAST_CONNECTION=log
36 FILESYSTEM_DISK=local
37 QUEUE_CONNECTION=database
38
39 CACHE_STORE=database
40 CACHE_PREFIX=
```

También hay que modificar los datos del archivo `config/database.php`



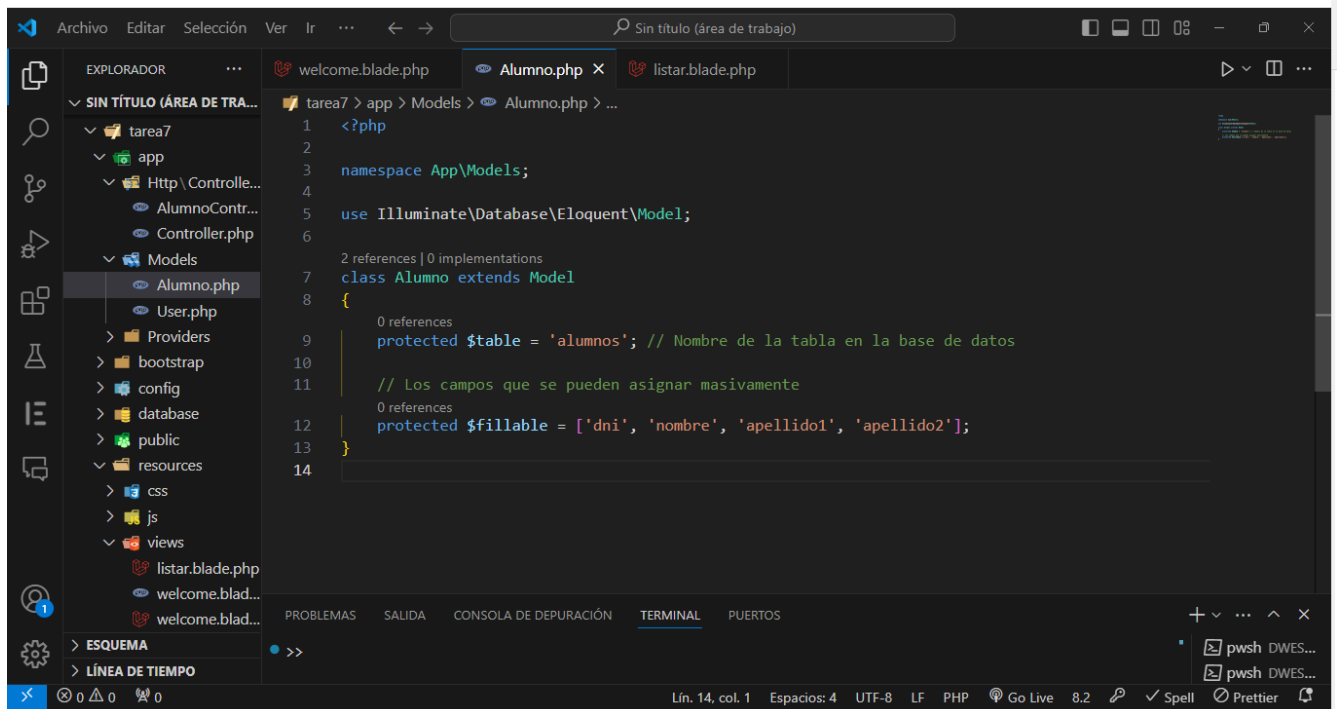
The screenshot shows the Visual Studio Code editor with the `config/database.php` file open. The file contains configuration for Laravel, including database connection details. The following lines are highlighted in blue:

```
5 return [
6     /*
7      * Here you may specify one of the following configurations for your application.
8      * Of course, you may also make custom configuration but you need to put the
9      * connections in the $connections array.
10     */
11
12     'default' => env('DB_CONNECTION', 'mysql'),
13
14     /*
15      * The connections array for the application.
16      */
17     'connections' => [
18         'mysql' => [
19             'driver' => 'mysql',
20             'url' => env('DB_URL'),
21             'host' => env('DB_HOST', '127.0.0.1'),
22             'port' => env('DB_PORT', '3306'),
23             'database' => env('DB_DATABASE', 'tarea7'),
24             'username' => env('DB_USERNAME', 'root'),
25             'password' => env('DB_PASSWORD', ''),
26             'unix_socket' => env('DB_SOCKET', ''),
27             'charset' => env('DB_CHARSET', 'utf8mb4'),
28             'collation' => env('DB_COLLATION', 'utf8mb4_unicode_ci'),
29             'prefix' => '',
30             'prefix_indexes' => true,
```

En mi caso, indico que use **mysql** y la base de datos **tarea7**. Como vamos a trabajar con el usuario **root**, no tocamos nada más.

-Crear una vista llamada /listar que muestre en una tabla html todos los alumnos de la tabla alumnos.sql.

Primero tenemos que crear el modelo alumno.



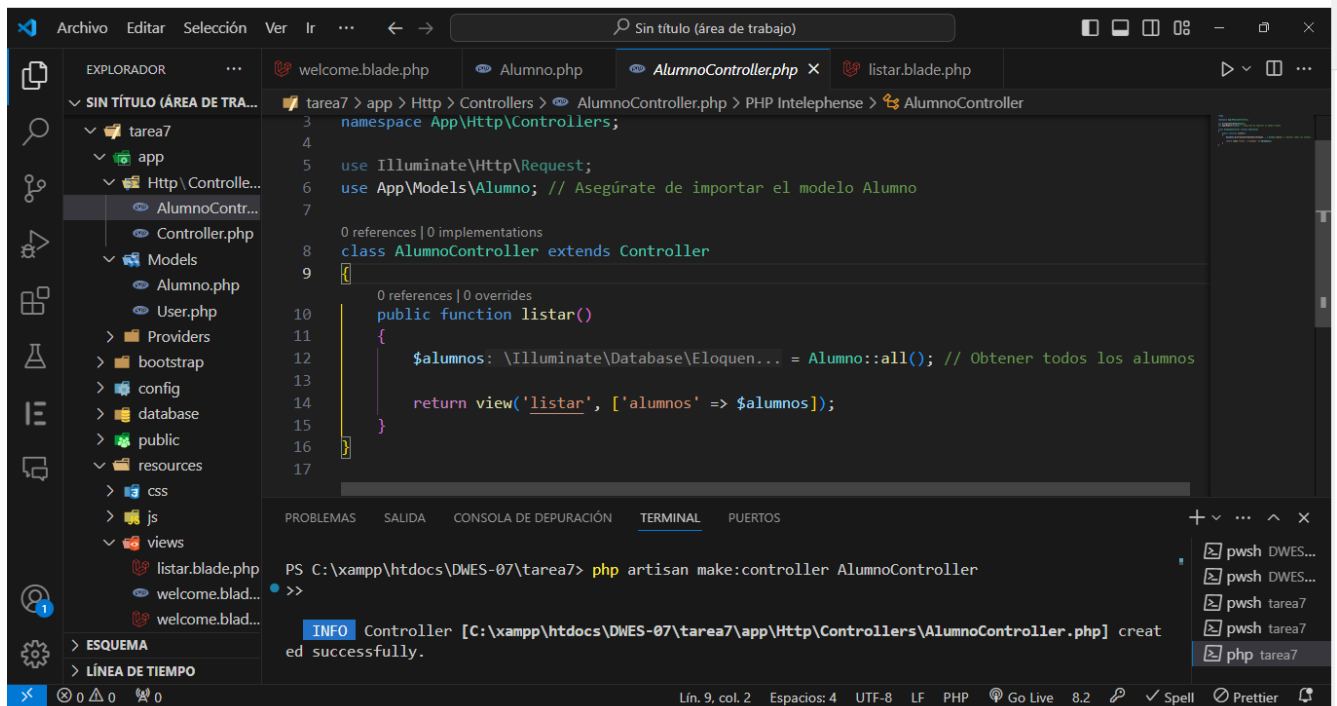
The screenshot shows the Visual Studio Code editor with the file explorer on the left. The project structure includes a 'Models' directory. The 'Alumno.php' file is open in the editor, showing the following code:

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 2 references | 0 implementations
8 class Alumno extends Model
9 {
10     0 references
11     protected $table = 'alumnos'; // Nombre de la tabla en la base de datos
12
13     // Los campos que se pueden asignar masivamente
14     0 references
15     protected $fillable = ['dni', 'nombre', 'apellido1', 'apellido2'];
16 }
```

The bottom panel shows the 'TERMINAL' tab with the command prompt.

Después creamos el controlador del alumno. Ponemos en la terminal lo siguiente (AlumnoController es el nombre): **php artisan make:controller AlumnoController**

La función listar usa el método **All()** para obtener todos los alumnos y después carga la vista “listar”.



The screenshot shows the Visual Studio Code editor with the file explorer on the left. The project structure includes a 'Http' directory. The 'AlumnoController.php' file is open in the editor, showing the following code:

```
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use App\Models\Alumno; // Asegúrate de importar el modelo Alumno
7
8 0 references | 0 implementations
9 class AlumnoController extends Controller
10 {
11     0 references | 0 overrides
12     public function listar()
13     {
14         $alumnos: \Illuminate\Database\Eloquent... = Alumno::all(); // Obtener todos los alumnos
15
16         return view('listar', ['alumnos' => $alumnos]);
17     }
18 }
```

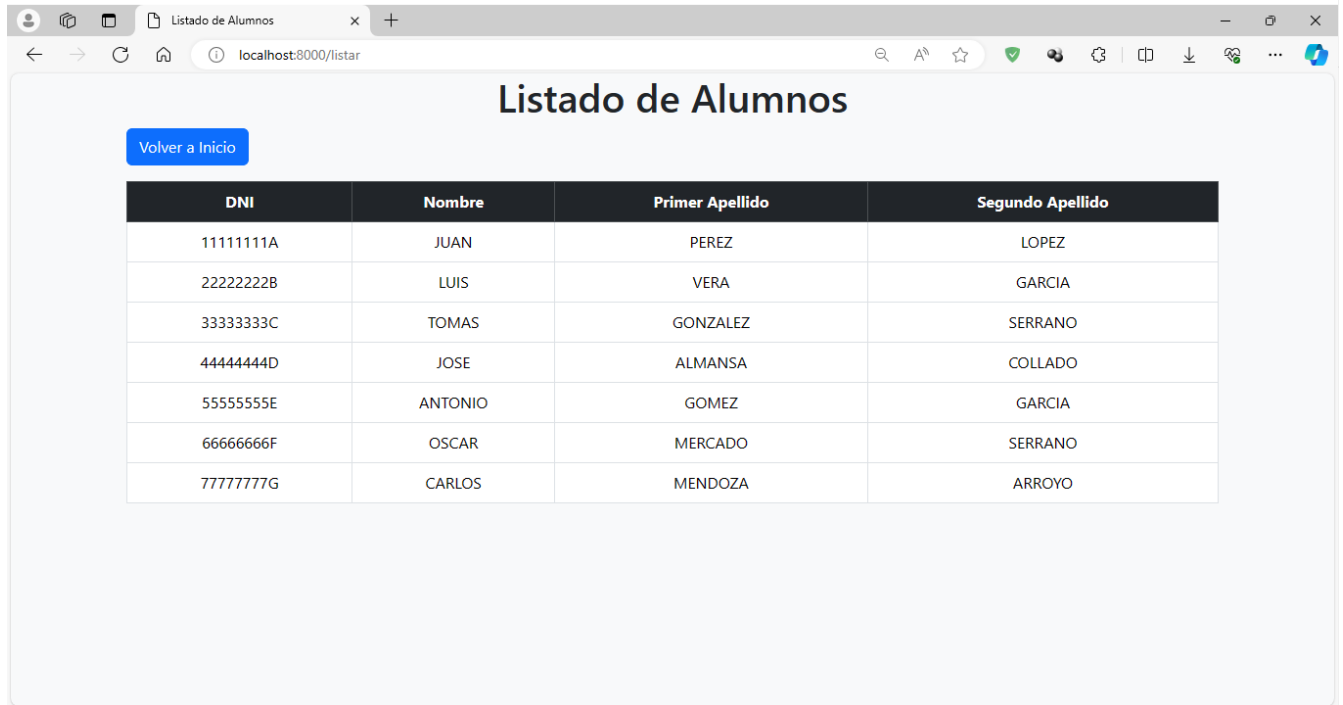
The bottom panel shows the 'TERMINAL' tab with the command prompt. The command **php artisan make:controller AlumnoController** has been executed, and the output shows that the controller was created successfully.

A continuación en el archivo **routes/web.php** añadimos una ruta:

Route::get('/listar', 'App\Http\Controllers\AlumnoController@listar');

Cuando un usuario accede a esta URL (listar), se llama al método listar del controlador **AlumnoController**.

Finalmente creamos nuestra vista “listar” (**listar.blade.php**).



DNI	Nombre	Primer Apellido	Segundo Apellido
11111111A	JUAN	PEREZ	LOPEZ
22222222B	LUIS	VERA	GARCIA
33333333C	TOMAS	GONZALEZ	SERRANO
44444444D	JOSE	ALMANSA	COLLADO
55555555E	ANTONIO	GOMEZ	GARCIA
66666666F	OSCAR	MERCADO	SERRANO
77777777G	CARLOS	MENDOZA	ARROYO

-Crear una vista llamada **/alta** que permita insertar alumnos en la tabla **alumnos.sql**.

Para este apartado debemos añadir dos funciones a **AlumnoController**.

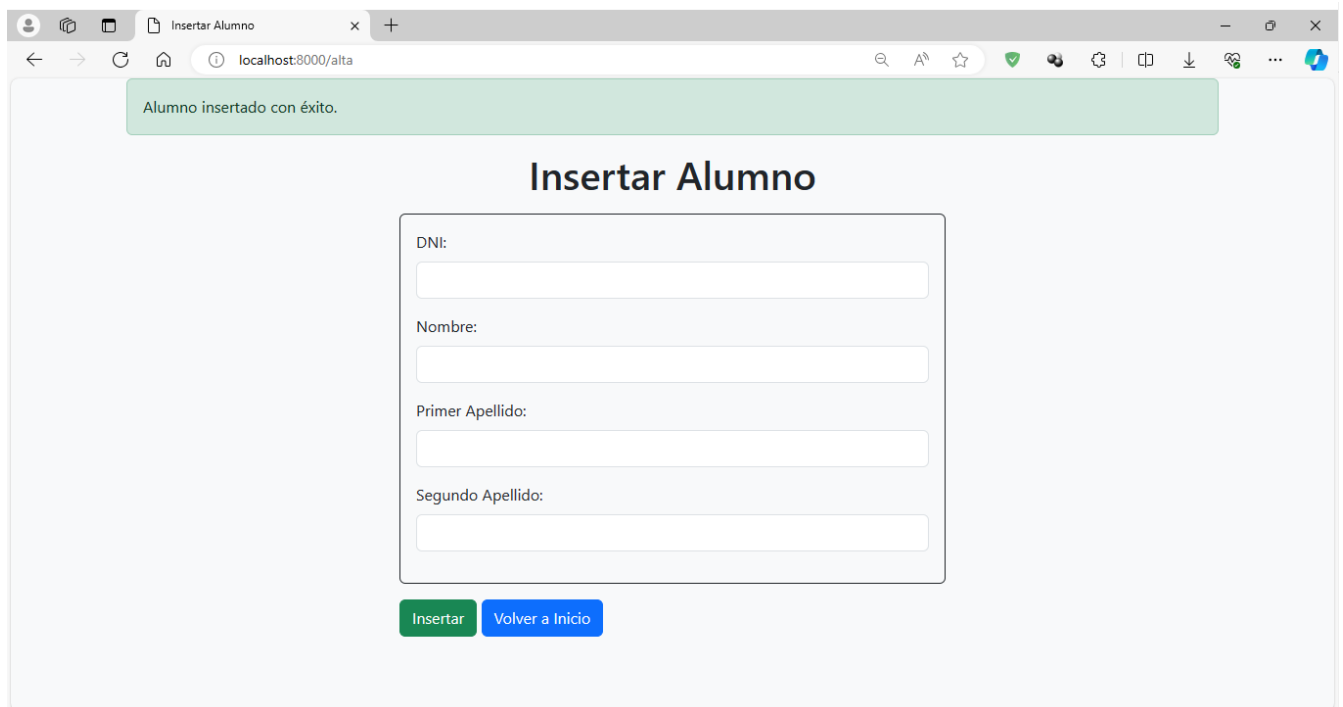
Una llamada **showAlta()** para mostrar la vista del formulario de inserción y otra llamada **store()** que procesa la solicitud para insertar un nuevo alumno y muestra un mensaje de éxito en caso de hacerlo.

Luego tenemos que añadir dos rutas nuevas a nuestro archivo **web.php**. Una llama al método **showAlta** y otra al método **store**.

```
Route::get('/alta', 'App\Http\Controllers\AlumnoController@showAlta');
```

```
Route::post('/alta', 'App\Http\Controllers\AlumnoController@store')->name('alta.alumno');
```

Finalmente creamos nuestra vista “alta” (**alta.blade.php**).



Alumno insertado con éxito.

Insertar Alumno

DNI:

Nombre:

Primer Apellido:

Segundo Apellido:

-Crear una vista llamada **/actualizar** que permita actualizar datos de un alumno en la tabla **alumnos.sql**.

Para este apartado debemos añadir tres funciones a **AlumnoController**.

Una llamada **showUpdate()** para mostrar la vista de actualización, otra llamada **buscarAlumnoActualizar()** que buscare al alumno con el DNI introducido y mostrara los datos en un formulario editable, y otra llamada **actualizar()** que recoge los datos del formulario y actualiza la base de datos.

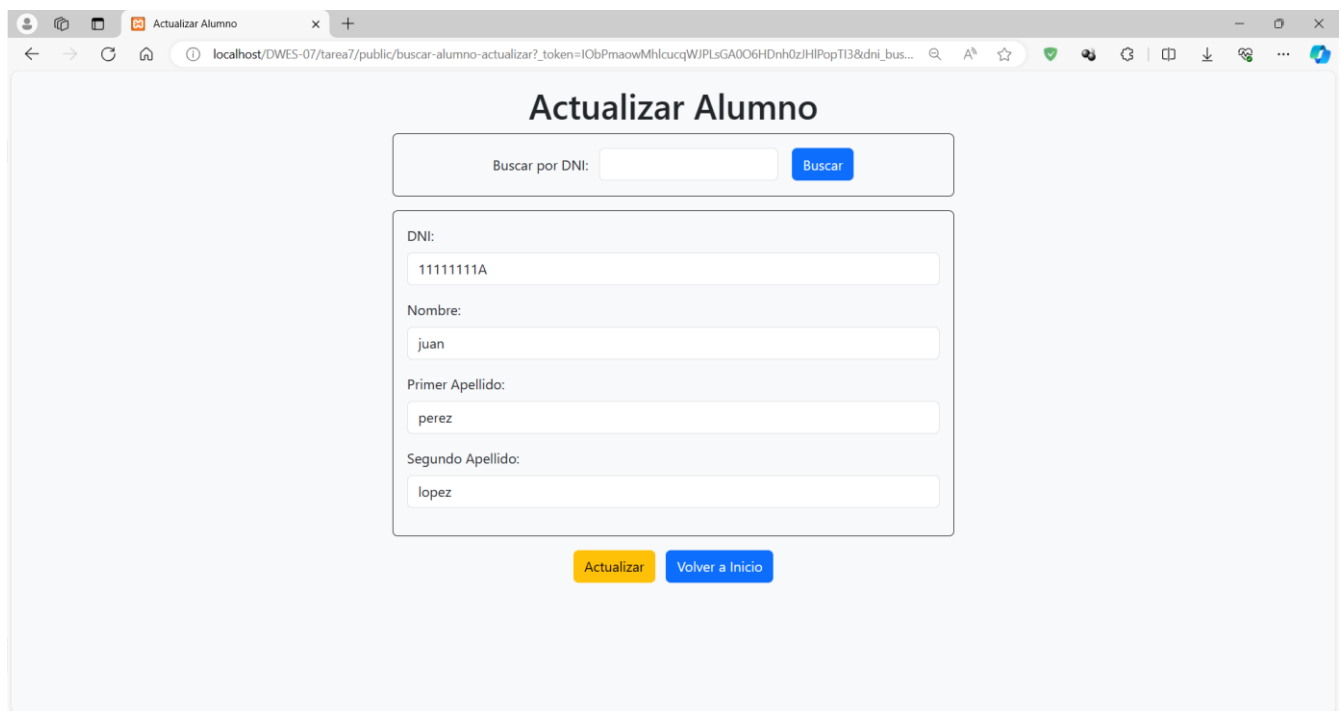
Luego tenemos que añadir dos rutas nuevas a nuestro archivo **web.php**. Una llama al método **showUpdate**, otra al método **buscarAlumnoActualizar** y otra al método **actualizar**.

```
Route::get('/actualizar', 'App\Http\Controllers\AlumnoController@showUpdate');
```

```
Route::post('/buscar-alumno-actualizar', 'App\Http\Controllers\AlumnoController@  
buscarAlumnoActualizar')->name('buscar.alumno.actualizar');
```

```
Route::post('/actualizar-alumno', 'App\Http\Controllers\AlumnoController@actualizar')-  
>name('actualizar.alumno');
```

Finalmente creamos nuestra vista “actualizar” (**actualizar.blade.php**).



Actualizar Alumno

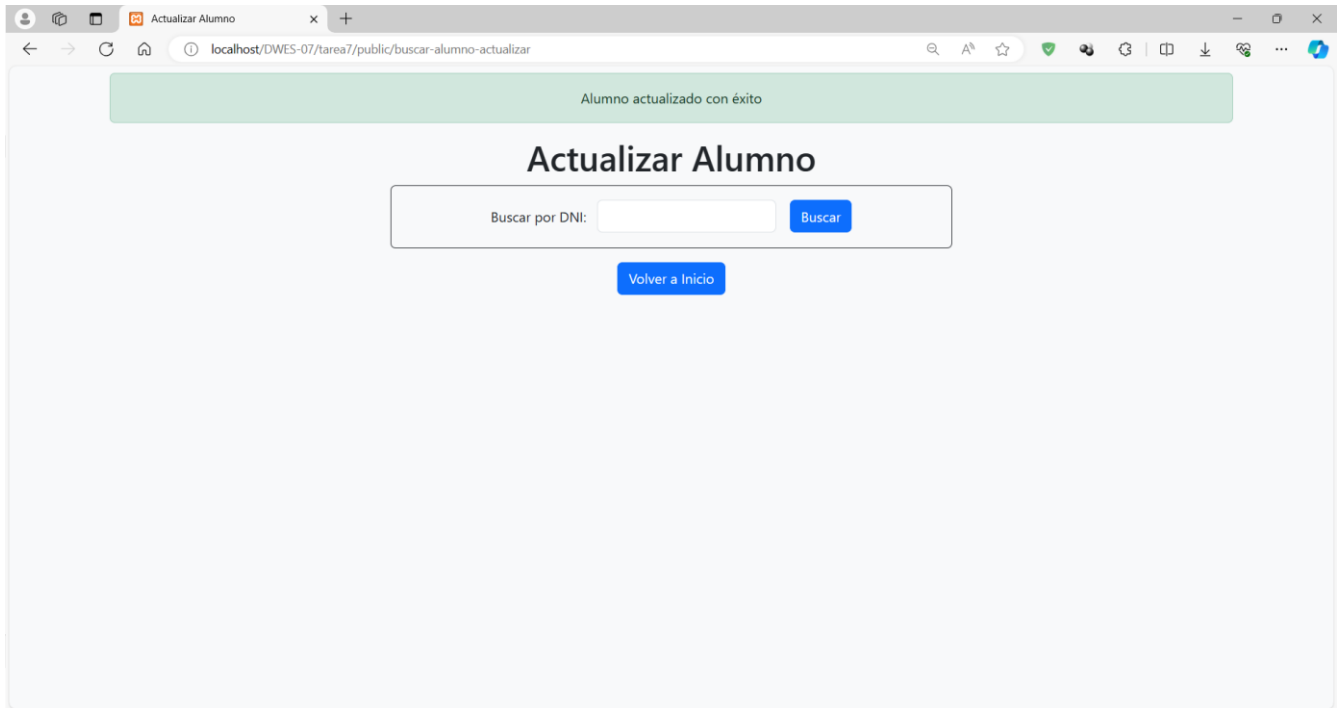
Buscar por DNI:

DNI:

Nombre:

Primer Apellido:

Segundo Apellido:



-He añadido una vista llamada **/eliminar** que permita eliminar datos de un alumno en la tabla **alumnos.sql**.

Para este apartado debemos añadir tres funciones a **AlumnoController**.

Una llamada **showDelete()** para mostrar la vista de actualización, otra llamada **buscarAlumnoEliminar()** que buscare al alumno con el dni introducido y mostrara una tabla con sus datos, y otra llamada **eliminar()** que elimina al alumno de la base de datos.

Luego tenemos que añadir dos rutas nuevas a nuestro archivo **web.php**. Una llama al método **showDelete**, otra al método **buscarAlumnoEliminar** y otra al método **eliminar**.

```
Route::get('/eliminar', 'App\Http\Controllers\AlumnoController@showDelete');
```

```
Route::post('/buscar-alumno-eliminar', 'App\Http\Controllers\AlumnoController@buscarAlumnoEliminar')->name('buscar.alumno.eliminar');
```

```
Route::post('/eliminar -alumno', 'App\Http\Controllers\AlumnoController@eliminar')->name('eliminar.alumno');
```

Finalmente creamos nuestra vista “eliminar” (**eliminar.blade.php**).

Eliminar Alumno

localhost:8000/buscar-alumno-eliminar?_token=c3WPXtNV0UXDR1TDaWtLfVpgiPpB2Ma5SHsryd&...

Eliminar Alumno

Buscar por DNI: Buscar

DNI	Nombre	Primer Apellido	Segundo Apellido
33333333C	TOMAS	GONZALEZ	SERRANO

Eliminar AlumnoVolver a Inicio