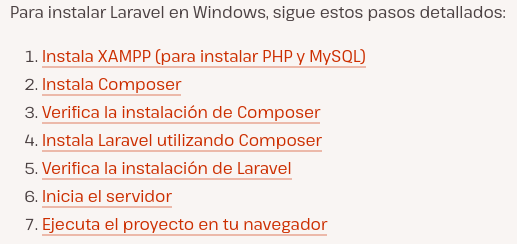
Objetivo: Construir una apiRest con Laravel 11

¿Qué es Laravel?

[Laravel](https://laravel.com/) es un potente y elegante framework de aplicaciones web PHP que agiliza el desarrollo de aplicaciones web mediante una arquitectura MVC (Modelo-Vista-Controlador).

Cómo Instalar Laravel en Windows



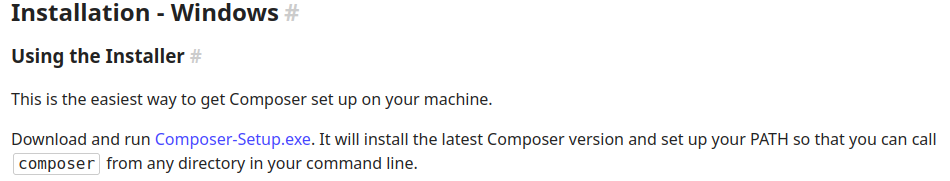
XAMPP

[XAMPP](https://kinsta.com/es/secciones/xampp/) es una solución de servidor web gratuita y de código abierto que incluye PHP, [MySQL](https://kinsta.com/es/base-de-conocimiento/que-es-mysql/) y Apache. Para instalar nos dirigimos a esta dirección https://www.apachefriends.org/download.html

Composer

Composer es una herramienta de gestión de dependencias para PHP que es necesaria para instalar Laravel. Para descargar el instalador nos dirigimos a

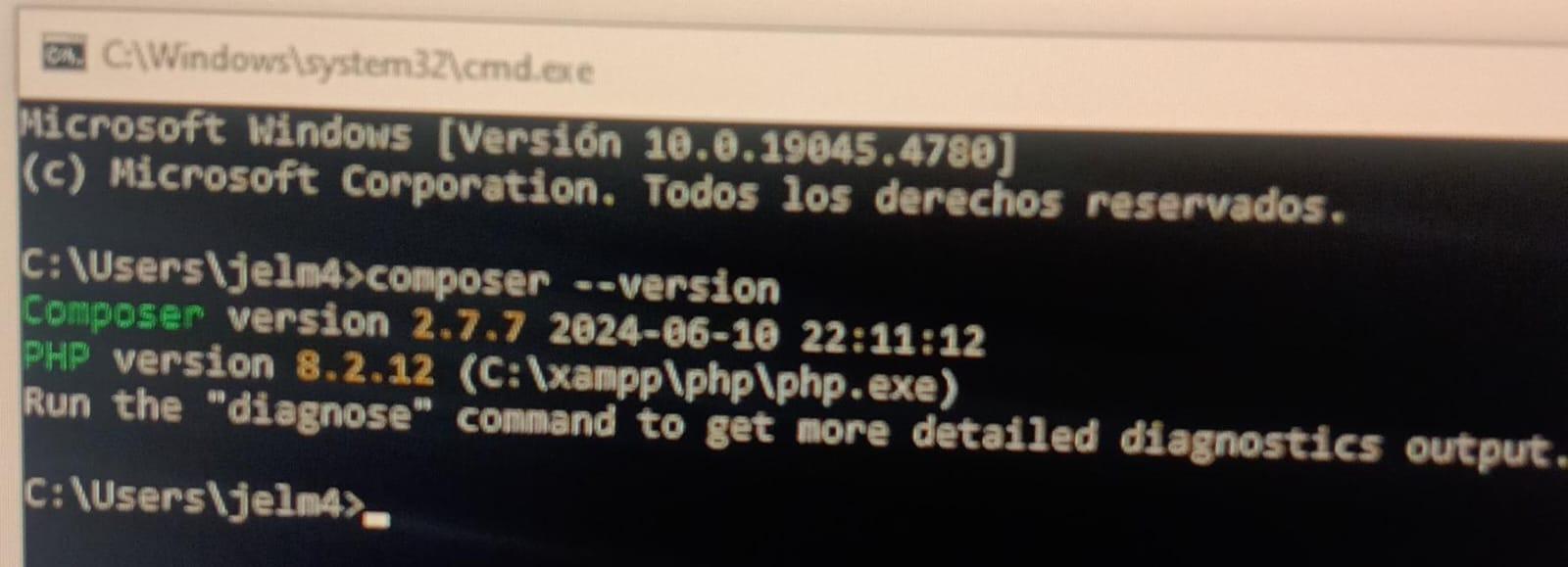
[https://getcomposer.org/doc/00-intro.md#installation-windows](https://getcomposer.org/doc/00-intro.md" \l "installation-windows) y buscamos el instalador para windows. Lo descargamos y ejecutamos.



Una vez instalado Composer verificamos mediante composer –version

Para ello abrimos una terminal y ejecutamos composer --version

Si la instalación se ha realizado correctamente, deberías ver la versión de Composer mostrada.



### 

### Instalar Laravel Utilizando Composer

Puedes utilizar Composer, que ya está instalado, para instalar Laravel globalmente en tu sistema. Para ello, abre el símbolo del sistema y ejecuta el comando indicado:

***composer create-project laravel/laravel app-name***

También podemos instalar Laravel mediante

***composer global require laravel/installer***

Si empleamos este último comando entonces creamos el proyecto mediante ***laravel new example-app***

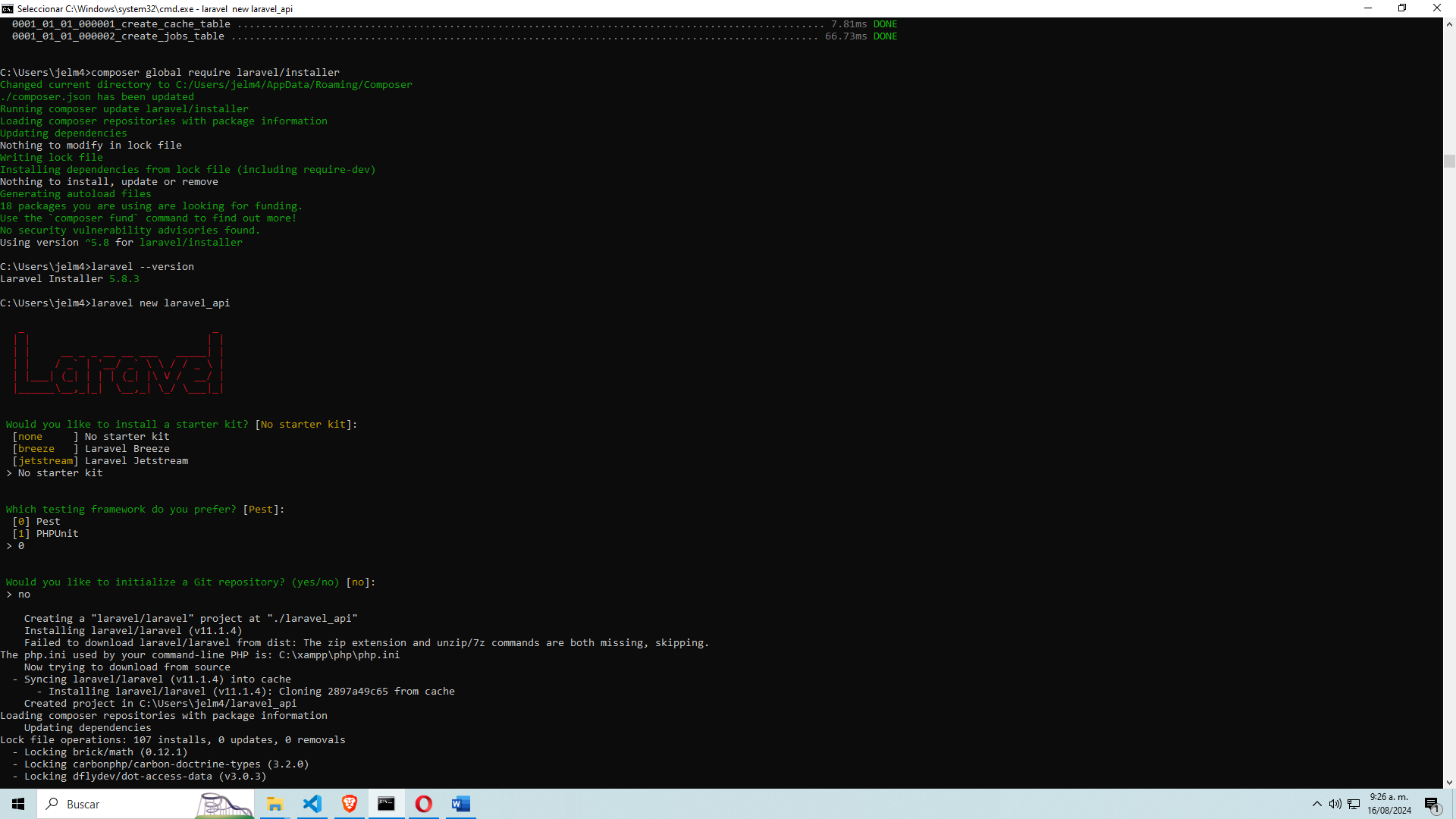
Para verificar que Laravel fue instalado correctamente escribimos

***laravel --version***

Creando un proyecto

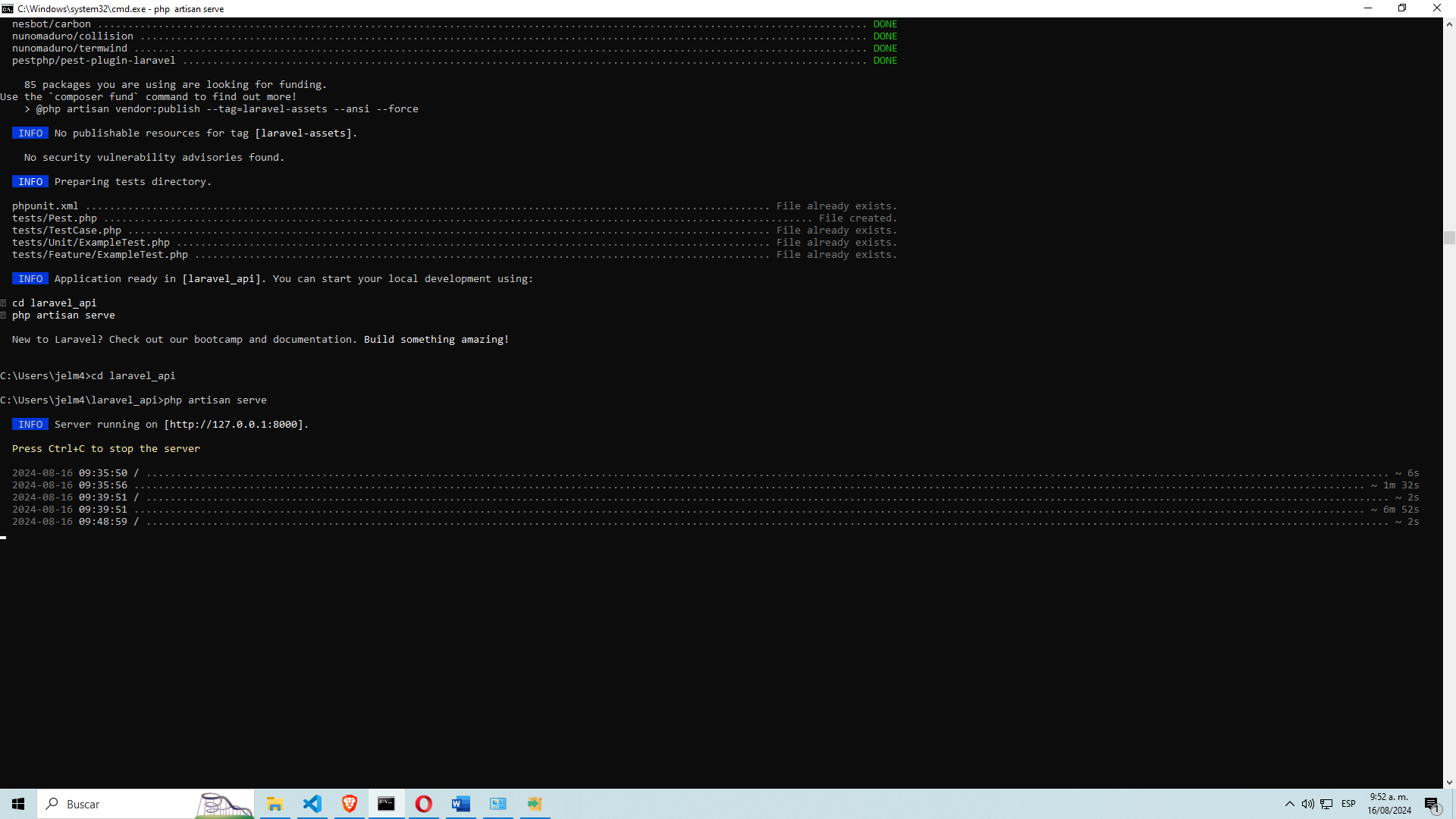
En una terminal digitamos ***laravel new laravel\_api*** para crear un proyecto.

Aparecerá el siguiente mensaje solicitando ciertas opciones, las cuales serán las mismas que la imagen.



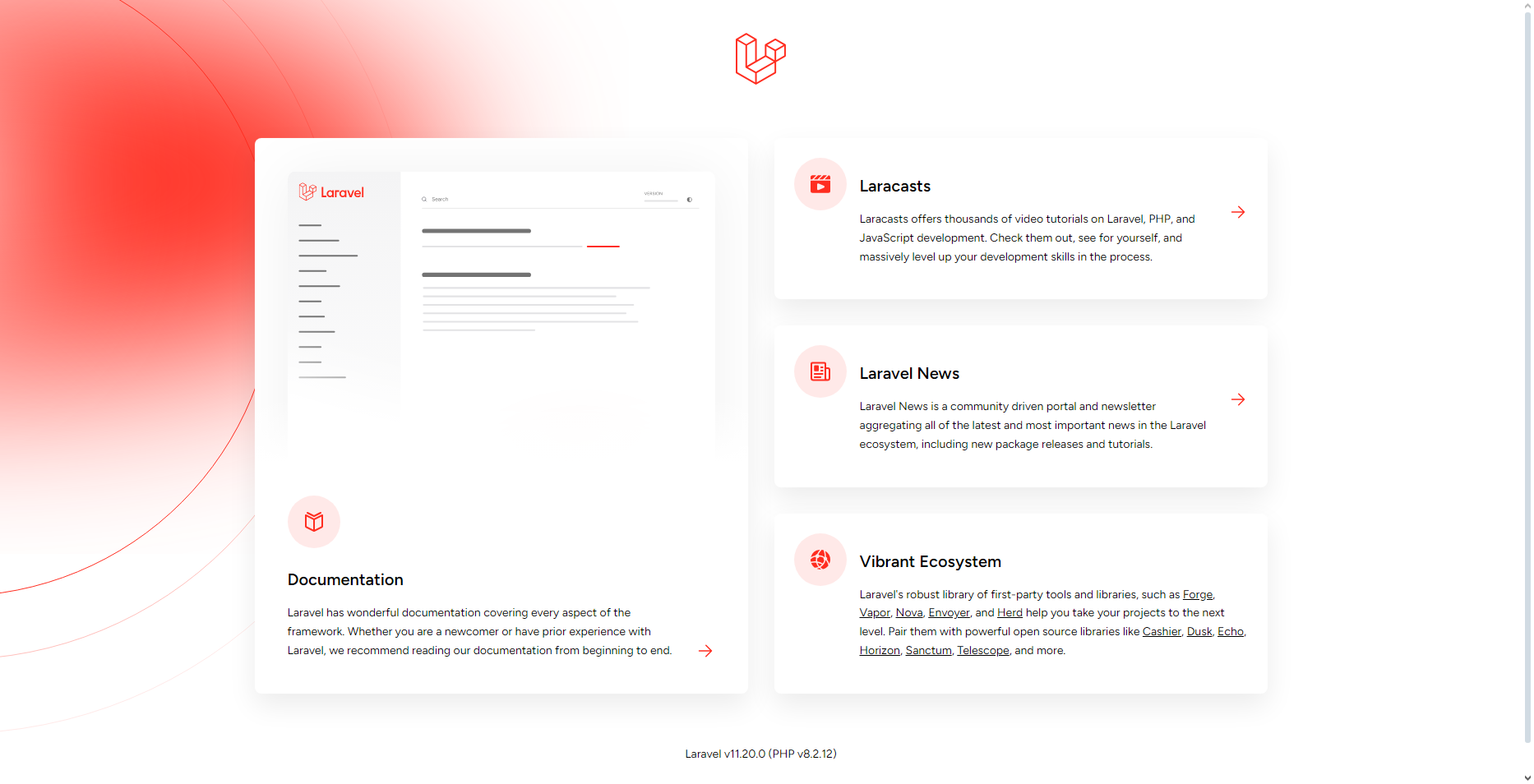
Una vez creado el Proyecto nos pedirá la base de datos a seleccionar. Escogemos MySQL y le decimos que **sí** a la migración de la Base de Datos.

Finalizado el proceso, nos vamos al proyecto creado con ***cd nombre\_del\_proyecto*** y arrancamos el servidor con **php artisan serve**

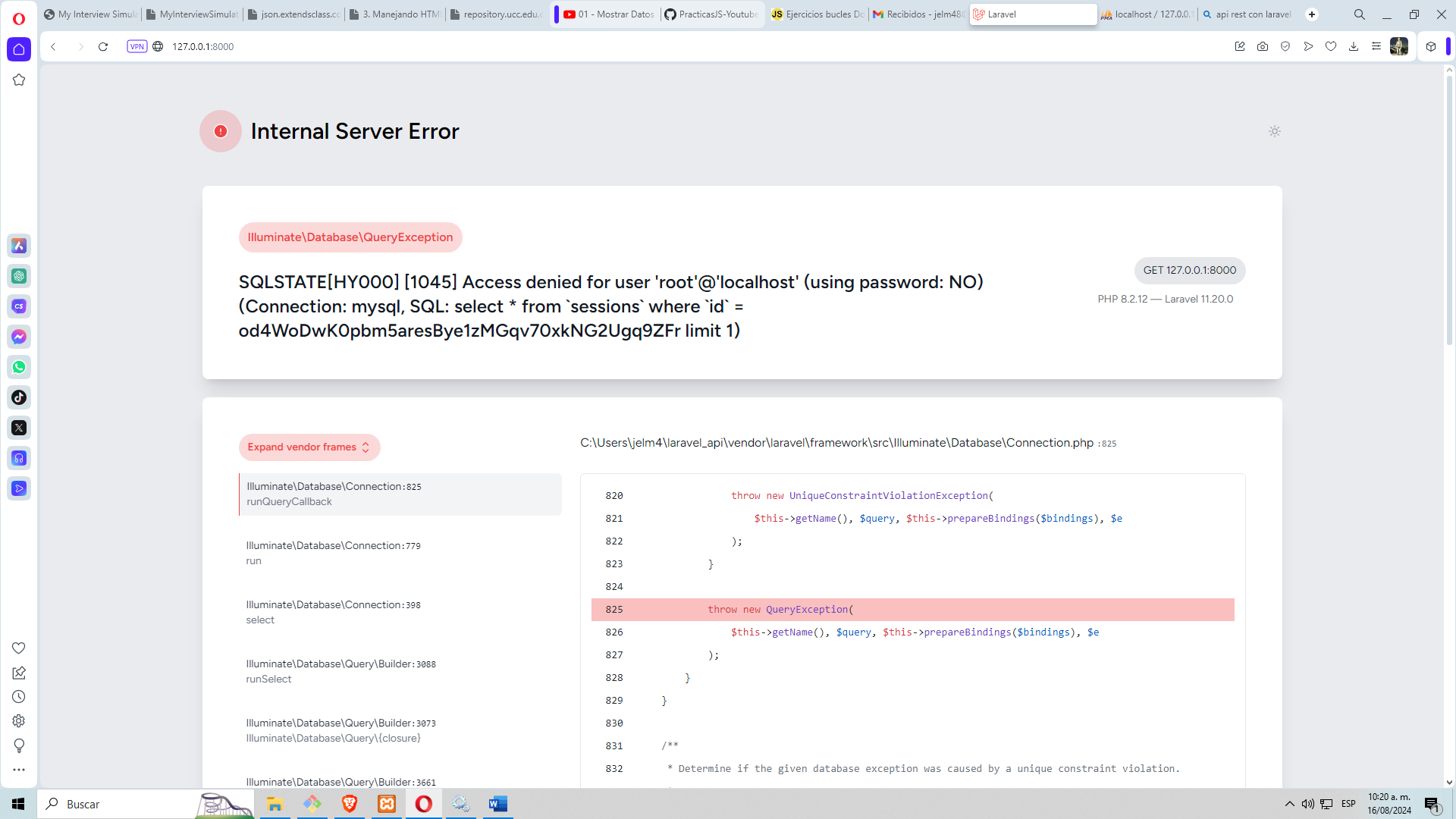


Nos vamos a un navegador y digitamos ***localhost:8000***

Si todo sale bien debe aparecer un mensaje de bienvenida como el siguiente.

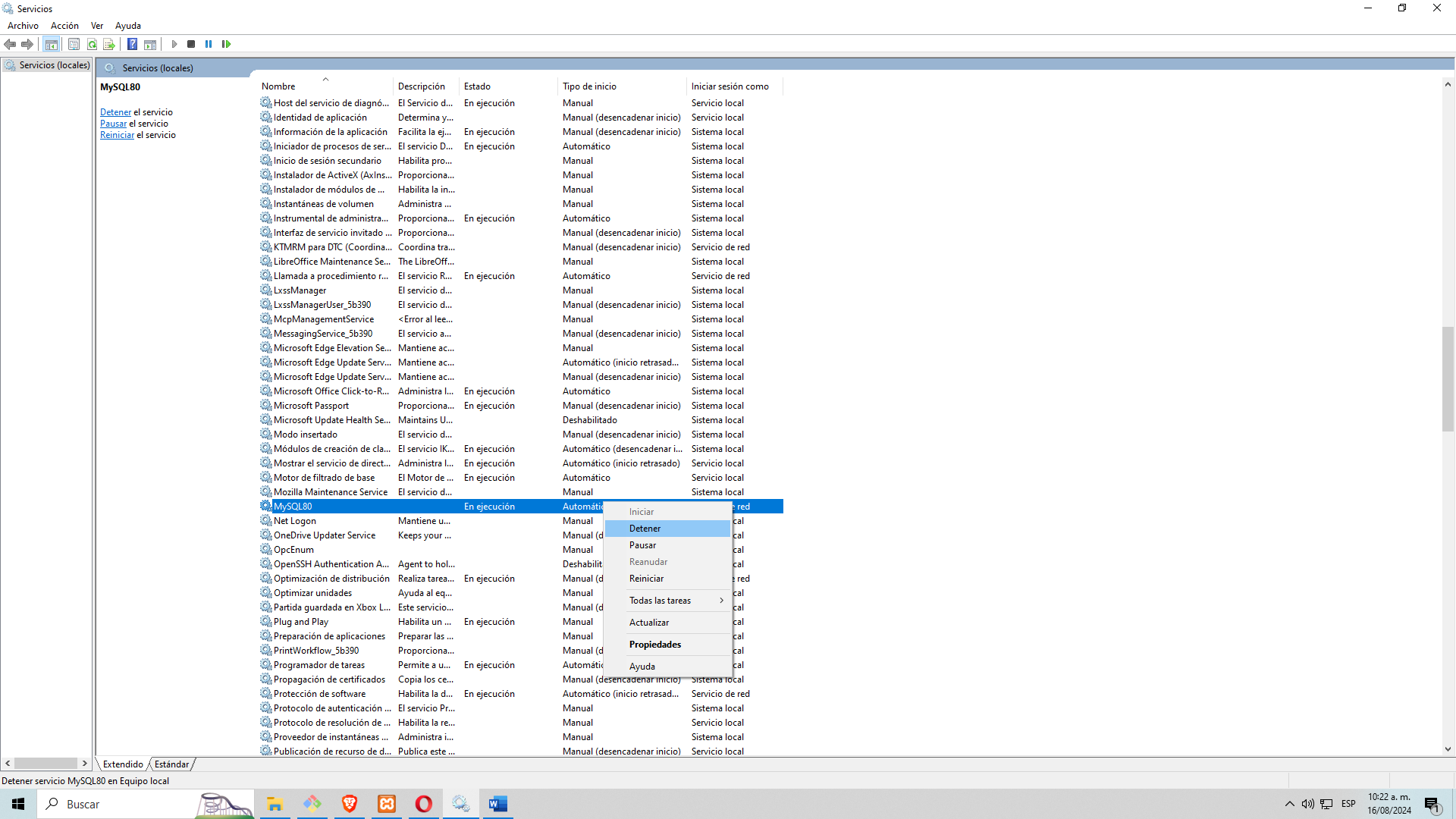


**Posibles errores al levantar el servidor**

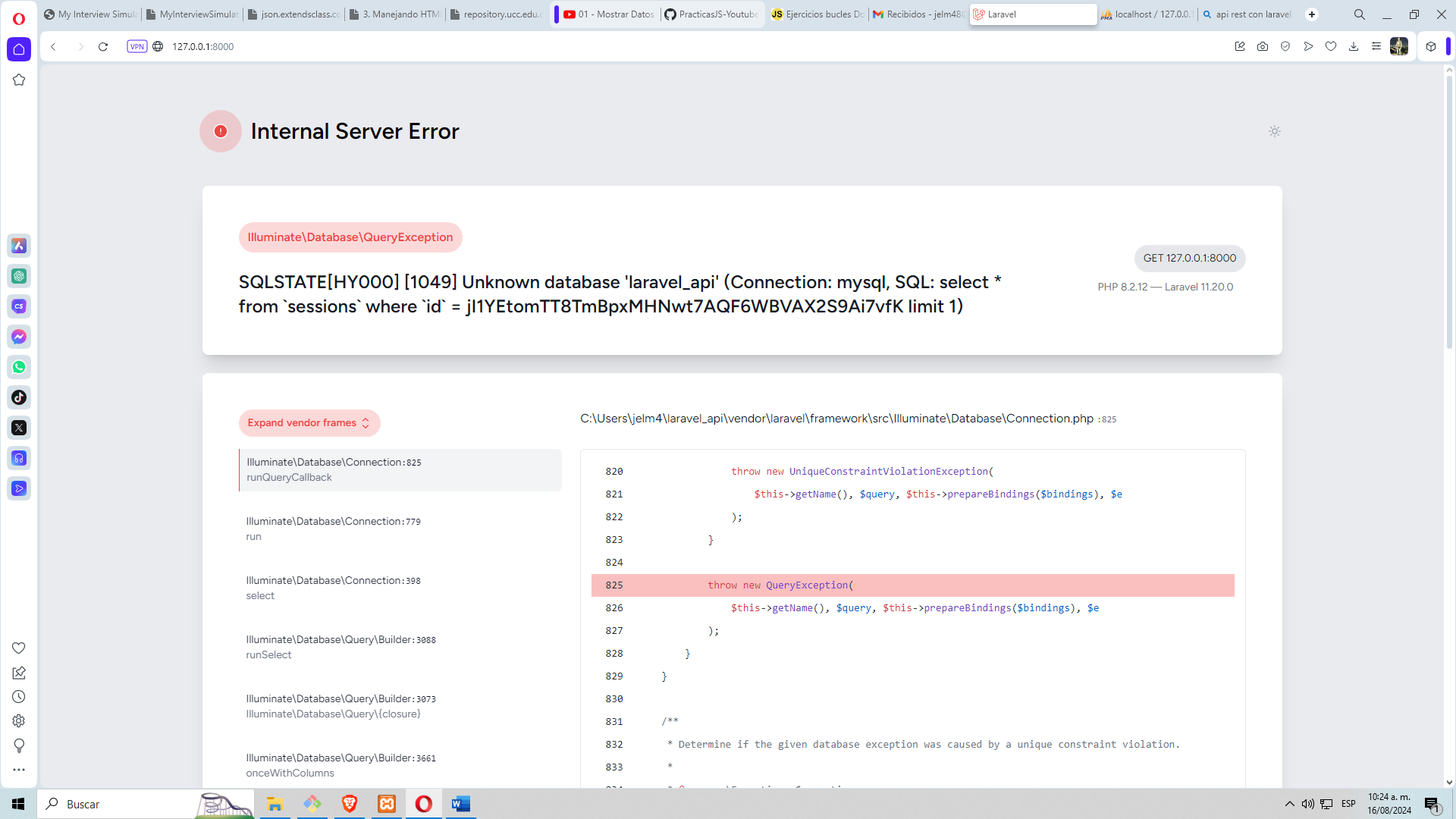


Este error aparece cuando MYSQL ya está corriendo en el Puerto 3306 o no está corriendo.

Para el primer caso, la solución es abrir los servicios de windows y detener ese proceso. Luego iniciamos MySQL dese XAMPP.

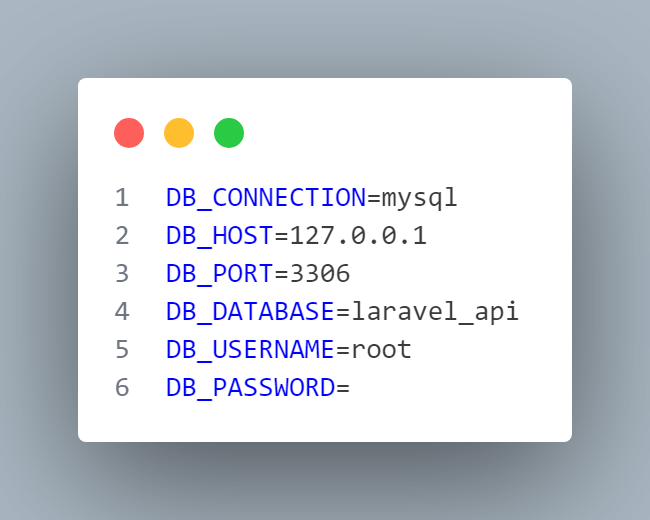




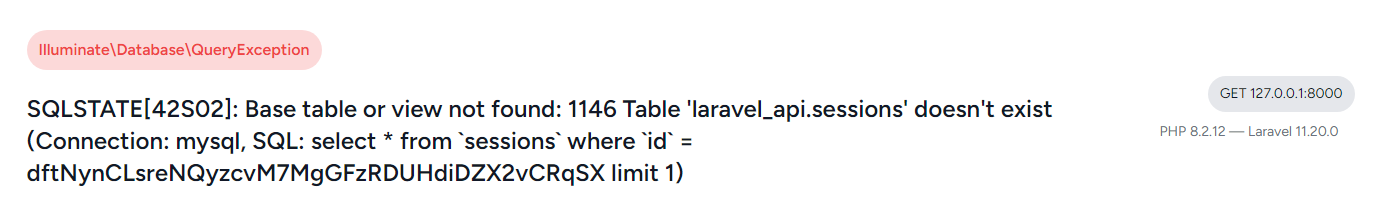


Este error aparece cuando no se ha creado la BD.

Para solucionarlo debemos ir a phpMyAdmin y crearla con el nombre que se nos indica en el mensaje. Una vez creada la BD nos vamos al archivo .env y verificamos las credenciales



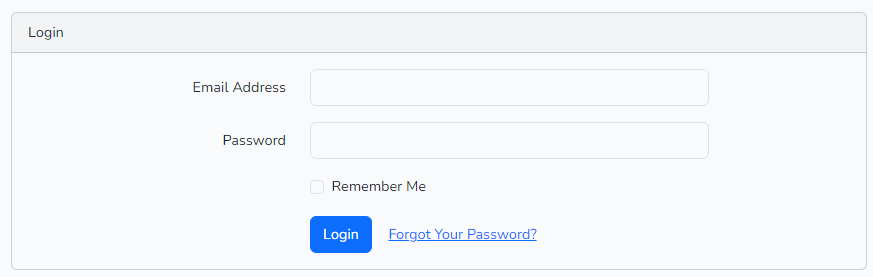
3.

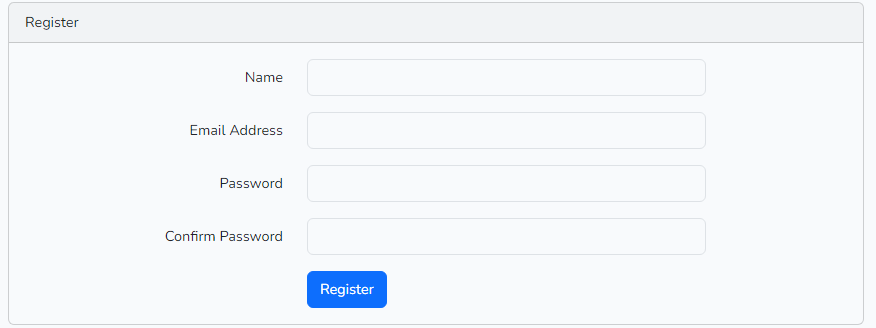


Para solucionar este error es necesario ejecutar las migraciones mediante ***php artisan migrate***

**Creando la integración del login**

En este punto vamos a crear y llamar la interfaz gráfica (ui) que nos permitirá configurar el login y el register.





1. En nuestro proyecto en la consola escribimos y ejecutamos

***composer require laravel/ui***

1. Integramos con bootstrap mediante ***php artisan ui bootstrap --auth***

Si aparece el mensaje **The [Controller.php] file already exists. Do you want to replace it? (yes/no) [yes]** le damos yes.

1. El sistema nos pedirá instalar NODE js. Luego de ser instalado ejecutamos ***npm install*** y después ejecutamos ***npm run dev***

Este último debería crear una carpeta js y css en la carpeta public.

1. Iniciamos de nuevo el servidor mediante ***php artisan serve***

Debemos ver en la parte superior derecha el enlace del Login y Register.

En caso de arrojar error similar a [**Vite manifest not found**](https://stackoverflow.com/questions/73045616/vite-manifest-not-found)entonces ejecutamos ***npm run build***

**Creando un CRUD**

1. Vamos a nuestro proyecto y detenemos el servidor mediante ctrl + c.

Ejecutamos ***php*** ***artisan install:api***

Este comando creará en la carpeta routes un archivo llamado api.php, y en la carpeta bootstrap configura el archivo app.php para arrancar las rutas definidas en la api.php

En este punto es posible que aparezca el mensaje

*One new database migration has been published. Would you like to run all pending database migrations? (yes/no) [yes]:*

Damos “yes” y enter.

1. Vamos al archivo api.php y lo modificamos de tal manera que quede así:

<?php

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Route;

Route::get('/usuarios', function (Request $request) {

return "Bienvenidos a mi API REST en LARAVEL 11";

});

Luego levantamos el servidor mediante ***php artisan serve***

En la URL de cualquier navegador escribimos [**http://127.0.0.1:8000/api/usuarios**](http://127.0.0.1:8000/api/usuarios)

Debe aparecer el mensaje **Bienvenidos a mi API REST en LARAVEL 11**

1. Creamos los diferentes EndPoints con sus respectivas peticiones

Route::get('/usuarios/{id}', function (Request $request) {

return "Consultando un solo usuario";

});

Route::post('/usuarios', function (Request $request) {

return "Creando usuarios";

});

Route::put('/usuarios/{id}', function (Request $request) {

return "Actualizando usuarios";

});

Route::delete('/usuarios/{id}', function (Request $request) {

return "Borrando usuarios";

});

Nótese que para las peticiones put y delete es necesario pasar como parámetro el id entre llaves del recurso a procesar. También para la get pero cuando se desea traer un solo recurso.

1. Probamos las peticiones mediante Thunder Client.
2. Creamos una tabla en nuestra BD. Para ello escribimos

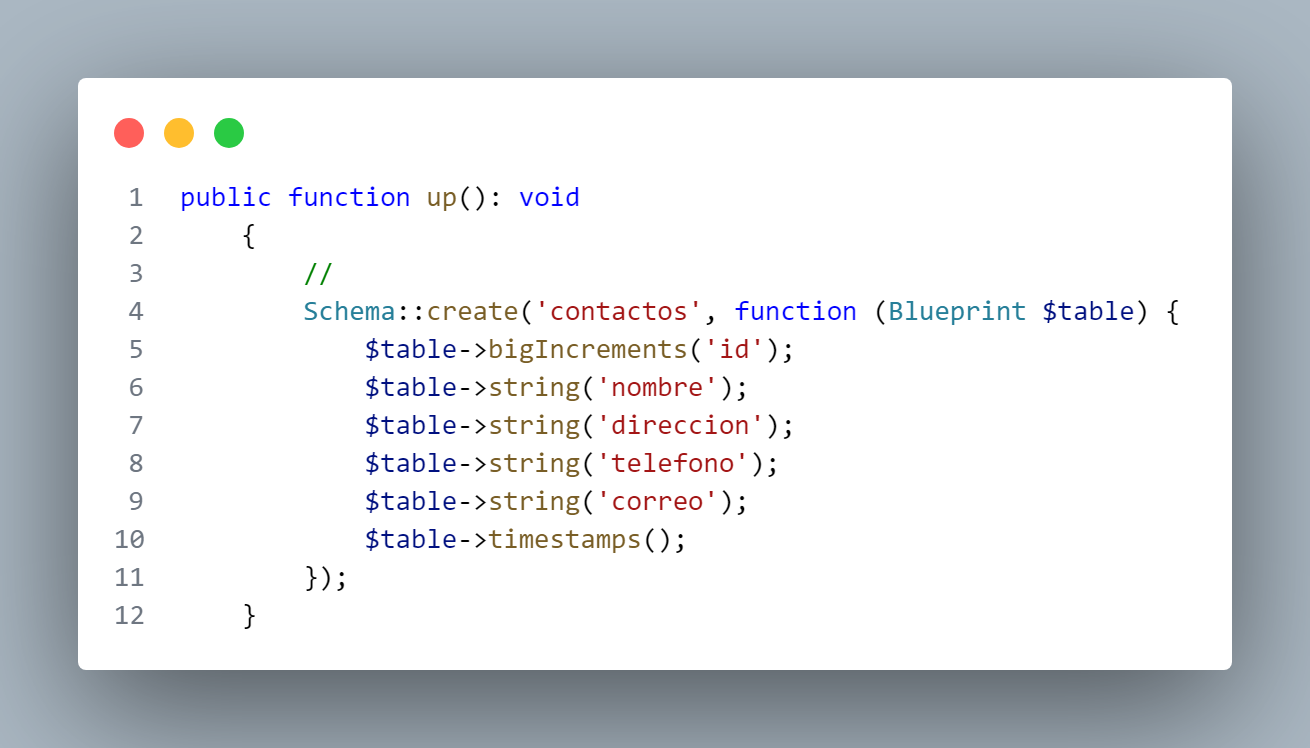
***php artisan make:migration contactos***

Esto nos creará una archivo en la ruta database/migration

Debemos agregar unos campos a la tabla recién creada. Para ello vamos a utilizar una extensión que nos ayudará en el proceso. Dicha extensión se llama **Laravel Snippets.**

Nos vamos para el archivo contactos.php y dentro de la function up escribimos schema::createtable

Deberá quedar así

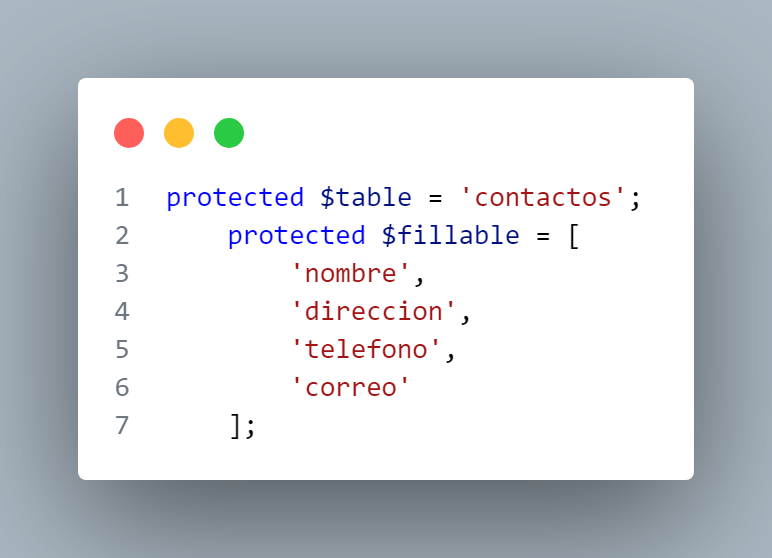


1. Una vez creada la tabla con sus campos ejecutamos la migración mediante ***php artisan migrate***

Ahora nos dirigimos a phpmyadmin y verificamos que en la BD se haya creado la tabla con los campos.

1. Siguiendo el patrón MVC, creamos un modelo para la tabla creada en el punto 5(contactos). Para ello escribimos ***php artisan make:model Contactos*** El nombre debe iniciar con la letra mayúscula. Esto creará en la ruta app/models un archivo llamado Contactos.php

En dicho archivo agregamos el siguiente código



Nótese que los campos deben ser los mismos que se establecieron en la tabla contactos.

1. Ahora creamos un controlador mediante ***php artisan make:controller contactosController***

Este controlador tendrá los métodos y/o funciones que se deben ejecutar cada vez cuando se visite una url. Por ejemplo, en el archivo contactosController podemos agregar el siguiente código

public function index()

{

return "Obteniendo lista de contactos";

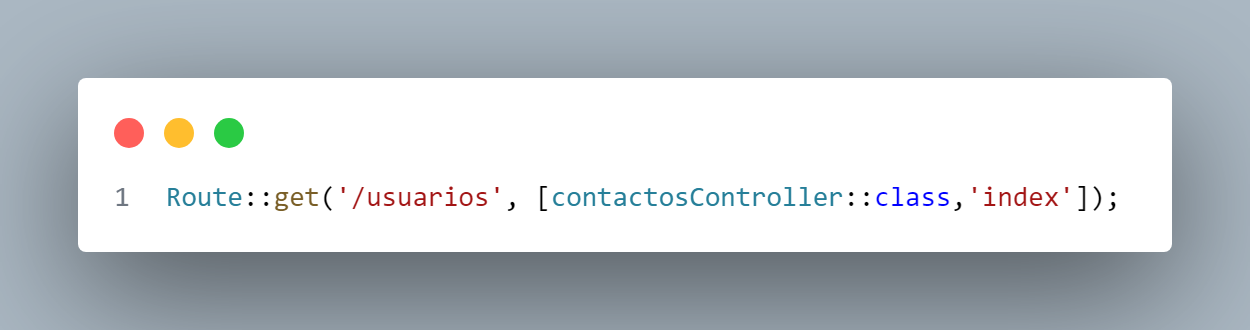
}



Luego nos vamos para la carpeta routes e invocamos el controller mediante

use App\Http\Controllers\contactosController;

Luego modificamos la petición get para traer todos los contactos quedando asi

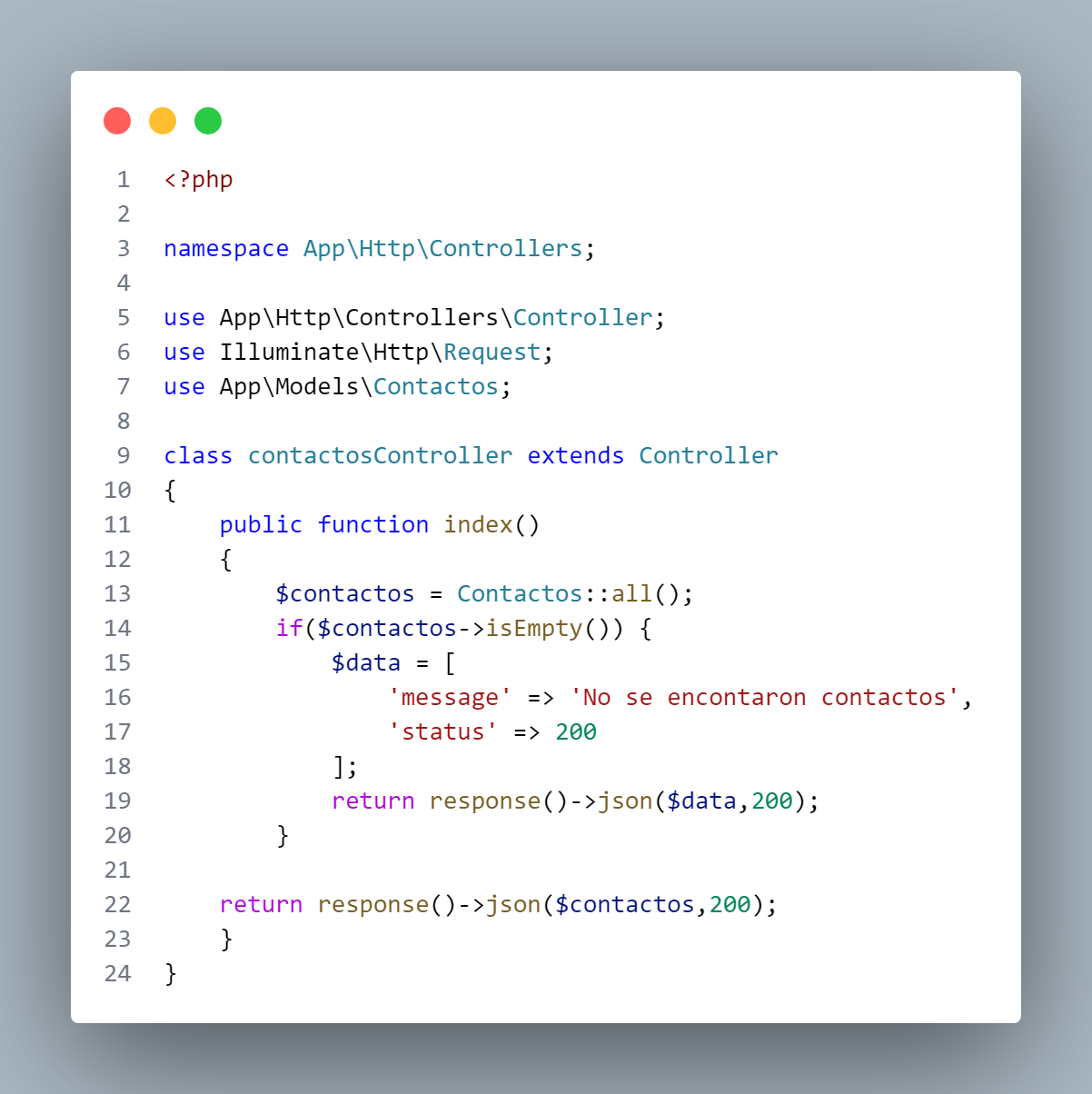


Levantamos el servidor de nuevo mediante ***php artisan serve*** y en la url del navegador escribimos localhost:8000/api/usuarios

Ahora debemos ver un mensaje pero enviado desde el controlador.

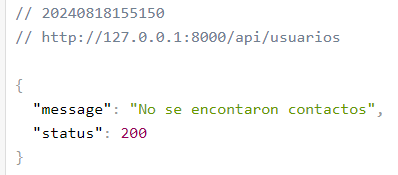
1. Consultamos la BD(GET)

Para ello modificamos la función index en el controller quedando así



En la url escribimos <http://127.0.0.1:8000/api/usuarios>

Debe aparecer el mensaje



1. Creamos un método(POST) para almacenamiento de datos con validación.

use Illuminate\Support\Facades\Validator;

En el archivo contactosController.php escribimos el siguiente código



En Thunder Client creamos la petición tipo POST. Si se superan las validaciones entonces la información enviada en el body deberá guardarse en la BD.

1. Creamos una petición tipo GET para consultar un solo registro mediante su identificador. Para ello escribimos el siguiente código.



Y en el enrutador modificamos de manera tal que quede asi

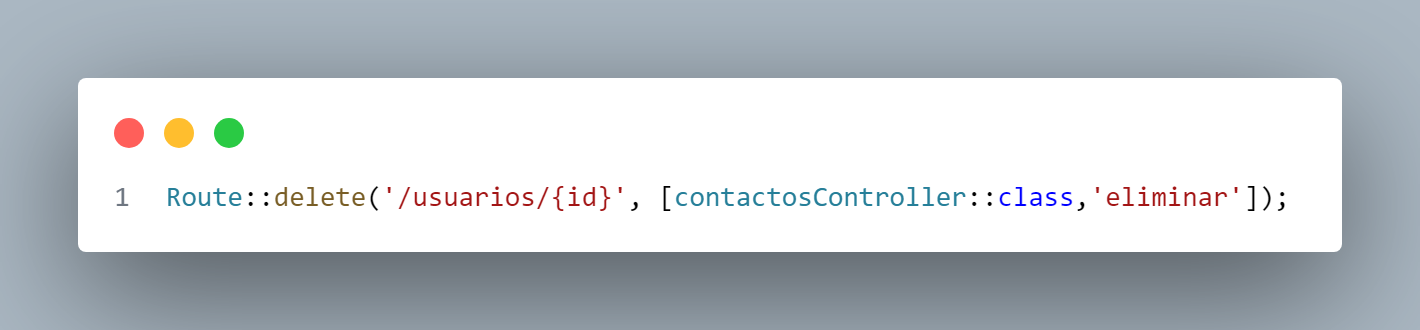


Al realizar la petición en Thunder Client o en el navegador debe mostrar el recurso solicitado, si es que existe.

1. Creamos el procedimiento para eliminar un recurso(DELETE). Para ello escribimos en el controller



Modificamos el endPoint en el archivo api.php



Realizamos la petición en Thunder Client y comprobamos que el recurso solicitado se haya eliminado de la BD.

1. Creamos el procedimiento para modificar un recurso específico(PUT).

En nuestro controller escribimos



Realizamos la petición en Thunder Client y verificamos que el recurso solicitado se haya actualizado.

https://www.youtube.com/watch?v=eLI8c\_NtkBk

<https://www.youtube.com/watch?v=fxifyqalyBc>