**PREPARAR ENTORNO LARAVEL**

1. **INSTALAR XAMPP**

[**https://www.apachefriends.org/es/index.html**](https://www.apachefriends.org/es/index.html)

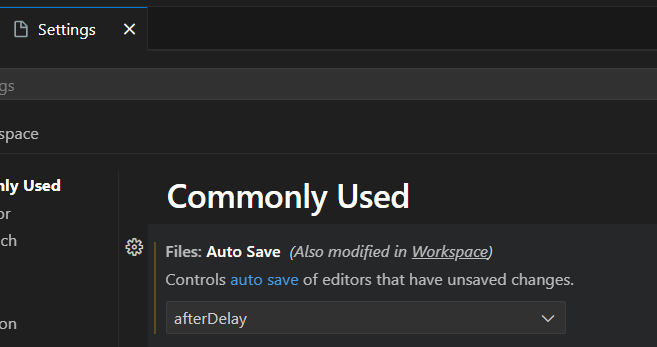
1. **Visual Studio**

[**https://code.visualstudio.com**](https://code.visualstudio.com)

1. **Activar guardado automatic**

**A screenshot of a black screen

Description automatically generated**

****

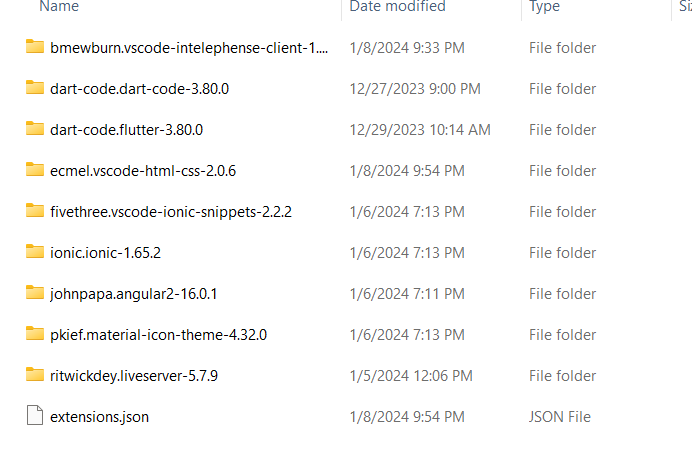
1. **Reset Visual Studio**

Entrar al usuario de la PC y entrar a AppData\Roaming\Code\User eliminar el archivo json

A screenshot of a computer

Description automatically generated

Entrar a la carpeta usuarios y entrar al link .vscode\extensions y eliminar todo



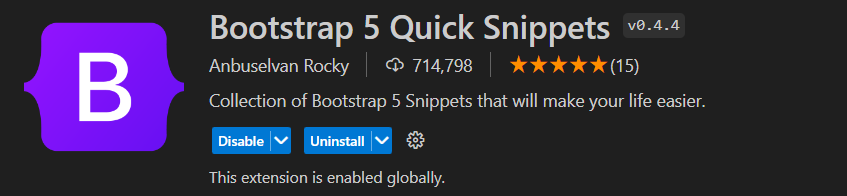
1. **Instalar extensiones**

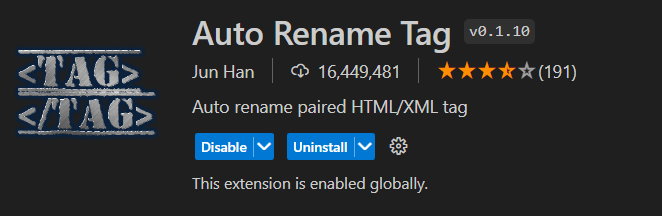
**A screenshot of a computer

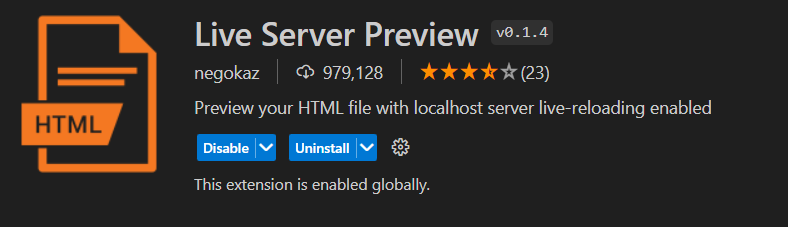
Description automatically generated**

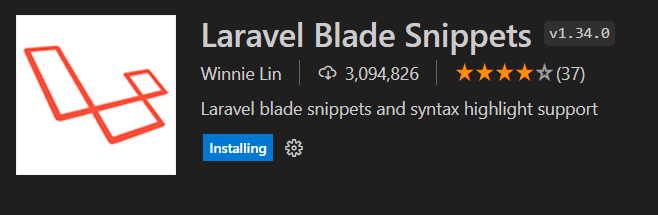
**A screenshot of a computer

Description automatically generated**

****

****

****

****

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

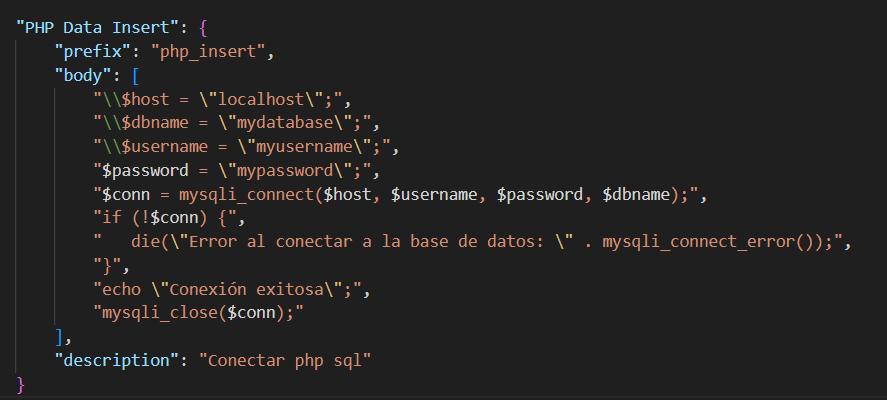
Description automatically generated**

1. **Configuracion de snippets**

Presionar shift+ctrl+p

escojer snippets configure user

escojer el lenguaje

Estructura: ****

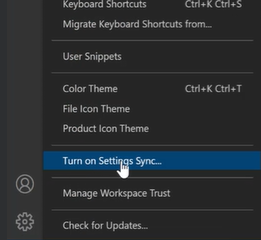
El “prefix” es como se va activar el snipet

El uso de comillas al principio y al final es obligatorio

Si se usa comillas se van a confundir con las comillas del inicio y final por eso se debe usar \

1. **Configuracion de snippets**

Sincronizar vs code



A screenshot of a computer

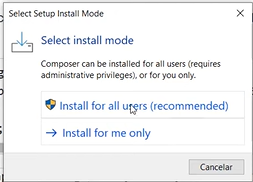
Description automatically generated

1. **Instalar Composer**

Descargar el .exe

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

Recomendable todos los usuarios



Verificar que la dirección este en el xampp/php

A screenshot of a computer

Description automatically generated

Todo next

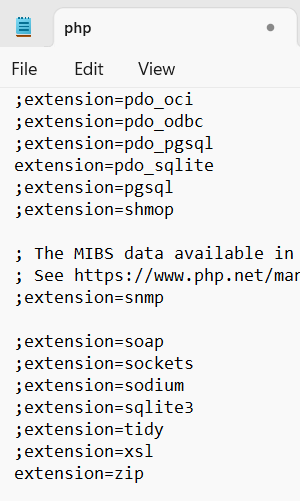
Vertificar que está instalado



1. **Instalación de Laravel**

Ir al config de php 

Y verificar que este descomentado la extensión = zip

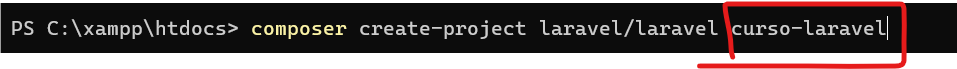


<https://laravel.com/docs/10.x>



Escribir el commando anterior en la ruta donde se encuentre el htdocs de xampp





Al ultimo cuadro rojo se pone el nombre del proyecto

1. **Encender el servidor**

****

1. **Información**

Se utiliza para realizar una gran variedad de tareas relacionadas con el desarrollo y la administración de aplicaciones Laravel.

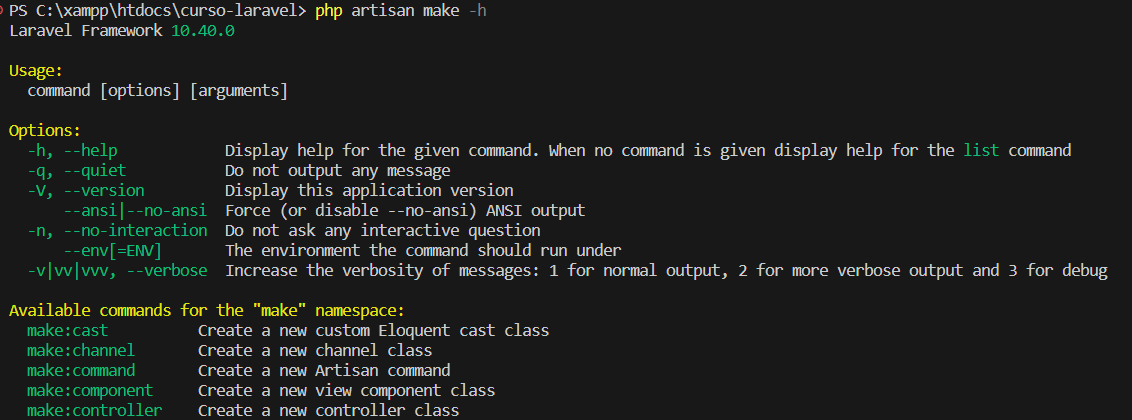
Generar archivos de código

Ejecutar tareas de depuración

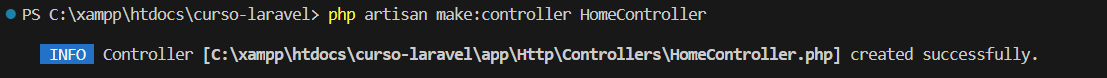
**A screen shot of a computer screen

Description automatically generated**

1. **Comando help**

****

1. **Comando para crear controlador**

****

1. **Comando php artisan tinker**

Es una herramienta de línea de comandos que viene incluida en el framework Laravel. Se utiliza para acceder a una consola interactiva de PHP en la que se pueden ejecutar comandos PHP y manipular los datos de una aplicación.

Strlen()

Encrypt()

Decrypt()

1. **Routes**

A screenshot of a computer

Description automatically generated

Se pueden cambiar las rutas

A black screen with text

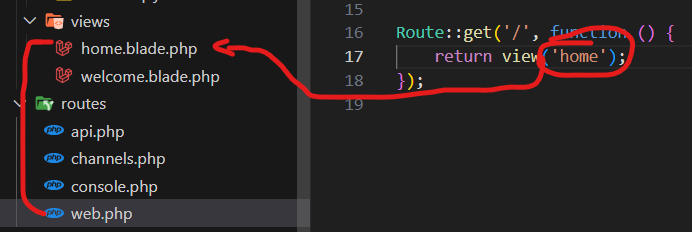
Description automatically generated

En views es lo que presenta al usuario

A screen shot of a computer

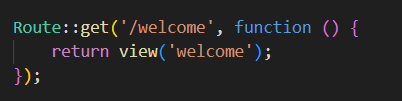
Description automatically generated

Se pueden crear más views y cambiarles las rutas

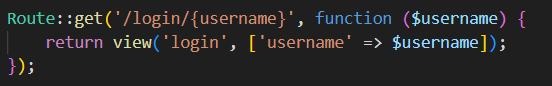


1. **Routas**

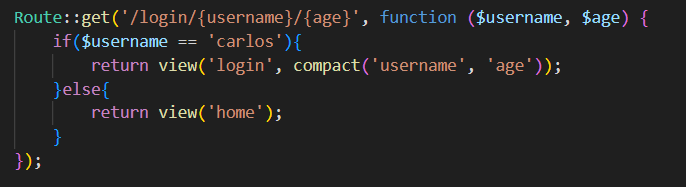
Normal

****

Rutas con variables

****

También se puede usar return view('login', **compact**('username');



A screen shot of a computer screen

Description automatically generated

Ruta para pagina off

A black background with colorful text

Description automatically generated

A computer screen with colorful text

Description automatically generated

Formas de llamar rutas

A computer screen with text

Description automatically generated

A screen shot of a computer program

Description automatically generated

Restricciones en los parametros

A screen shot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

Establecer reglas para las rutas

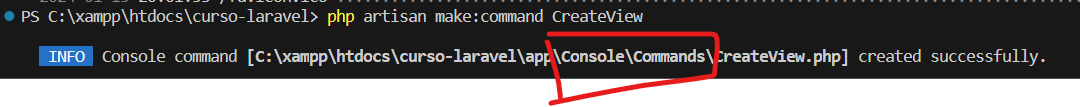
A screenshot of a computer program

Description automatically generated

A screen shot of a computer screen

Description automatically generated

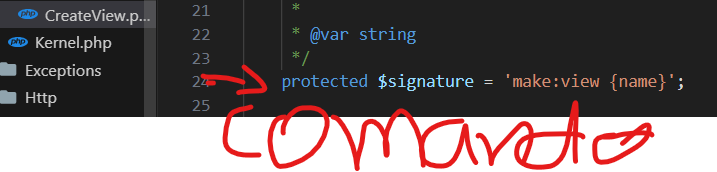
**VIEW**

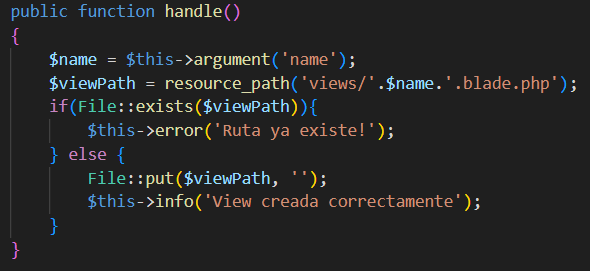
****

* **Establecer un comando correcto para la creación de views**

**A screen shot of a computer

Description automatically generated**

****

****

**A screen shot of a computer program

Description automatically generated**

****

Problema con el view

A screen shot of a computer

Description automatically generated

Solución de crear carpetas

A screenshot of a computer

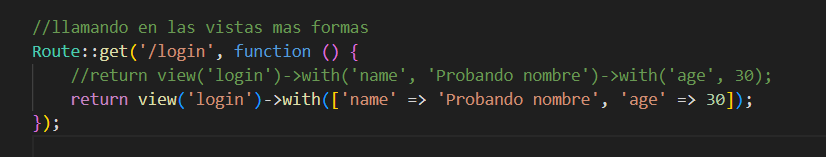
Description automatically generated

A black background with white text

Description automatically generated

Si adentro existen mas carpetas admin.sub.home etc

Pasar datos entre vistas



Compartir datos entre vistas

A screen shot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

Otra forma de poner variables

A computer screen with text

Description automatically generated

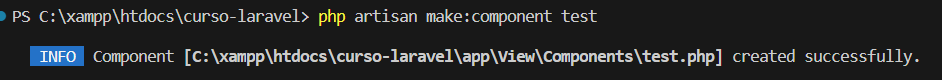
A screen shot of a computer code

Description automatically generated

A screen shot of a computer code

Description automatically generated

Mirar componentes



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Se puede llamar una o muchas veces

A screenshot of a computer

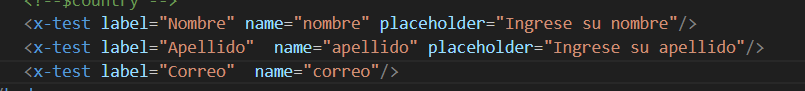
Description automatically generated

Hacer labels dinámicos

A screenshot of a computer program

Description automatically generated

Dinámicos inputs



* Ver components

Se declaran mas variables en la clase para poner más información en los inputs

A screen shot of a computer program

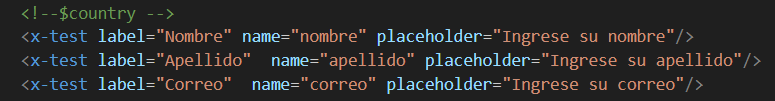
Description automatically generated

Debido a que en la clase en el constructor declaramos 3 variables entonces en los inputs deben a ver el uso de las 3 variables si no se usa 1 va a salir un error

A screen shot of a computer

Description automatically generated

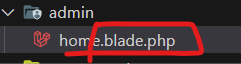
Ya no sale error



**BLADE**

* Visualización de datos

El Blade significa que se puede representar fácilmente la información o representar los datos frente al archivo



Para el ejemplo se declara una variable $name y se puede pasar los datos a la vista raíz

A black background with white text

Description automatically generated

A black background with white text

Description automatically generated

A black background with white text

Description automatically generated

Para evistar el XSS



Dentro de las llaves también se puede poner cualquier función php



* Datos sin escape

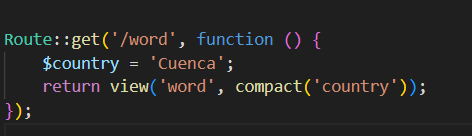
Para ponerlo en la forma de llaves y no en código php se lo hace de la siguiente manera:

Las dos formas evitan XSS



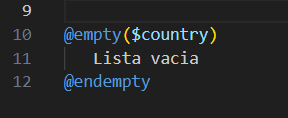
* Blade Directive IF

Las Blade directives son una forma de escribir código PHP en los archivos de plantilla de Laravel.



A screen shot of a computer program

Description automatically generated



A screen shot of a computer screen

Description automatically generated

* Blade directive switch

A screen shot of a computer program

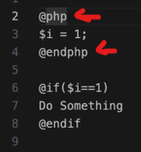
Description automatically generated

A computer screen shot of a program code

Description automatically generated

* PHP necesarios

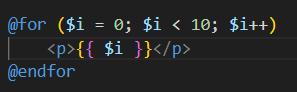
Si bien el escribir condigo php en Blade se lo hace con @ que pasa cuando queremos declarar una variable en Blade debido a que faltan <?php ?> se lo hace:



También se puede escribir código php en comenzando con esas etiquetas

* Bucles

Para imprimir en cualquier caso se puede usar ‘{{ }}’ o @php @phpend



A screen shot of a computer

Description automatically generated

Utilidad traer datos del backend

A screen shot of a computer code

Description automatically generated

A screen shot of a computer program

Description automatically generated

A black background with white text

Description automatically generated

* Subvistas

A black background with white text

Description automatically generated

Ejercicio: Incluir una vista que se encuentra en una carpeta

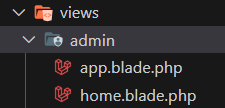
* Comentarios en Blade

A black background with white text

Description automatically generated

* Herencia secciones dinámicas Blade

En la carpeta de admin tengo 2 vistas, el ejercicio es que en la vista de app Blade tengo supuestamente el menú footer de una página web común por lo tanto para no estar repitiendo ese menú se puede heredar.



Vista **app.blade.php:**

Dentro del main\_content vamos a poner información de otras vistas por lo que se usa @yield para identificar el nombre de la sección de la otra vista

A screen shot of a computer program

Description automatically generated

Vista [**home.blade.php**](http://home.blade.php)**:**

En esta vista heredamos la vista app de la carpeta admin y en la sección va el mismo nombre que en @yield A screen shot of a computer program

Description automatically generated

**Resultado:**

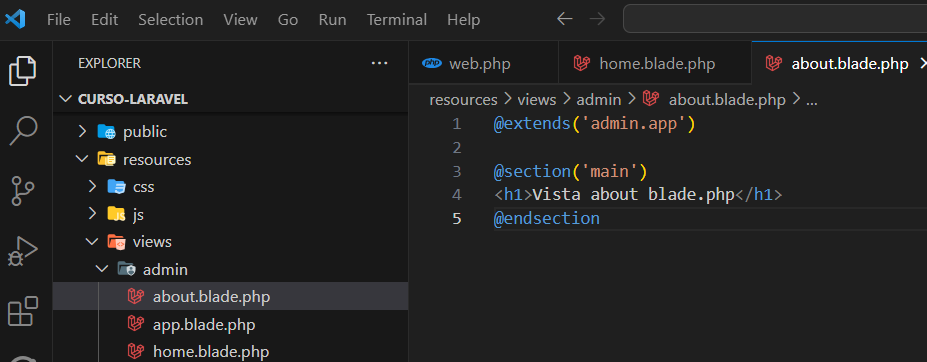
**A white background with black text

Description automatically generated**

Se obtiene la información tanto del app.blade y del home.blade

Ejercicio: Verificar el inspector

Podemos seguir creando vistas y heredando por ejemplo aquí creamos un archivo about y funciono



Ejercicio: Crear y heredar una vista afuera

**CONTROLLER**

Los controladores en Laravel son clases que manejan la lógica de la aplicación y se encargan de procesar las solicitudes del usuario. Son una parte fundamental de la arquitectura MVC de Laravel, lo que te permitirá desarrollar aplicaciones de manera estructurada y escalable.

Las principales funciones de los controladores en Laravel son:

* **Recibir y procesar las solicitudes del usuario**. Los controladores reciben las solicitudes del usuario a través de las rutas definidas en el archivo routes.php. Una vez recibida la solicitud, el controlador se encarga de procesarla y generar una respuesta.
* **Comunicar la vista con el modelo**. Los controladores se encargan de comunicar la vista con el modelo. La vista es la responsable de mostrar la información al usuario, mientras que el modelo es la responsable de gestionar los datos. El controlador es el intermediario entre ambos, proporcionando la información que necesita la vista y recibiendo los datos que necesita el modelo.
* **Manejar la lógica de la aplicación**. Los controladores pueden manejar cualquier tipo de lógica de la aplicación, como la validación de datos, la autenticación, la autorización, etc.

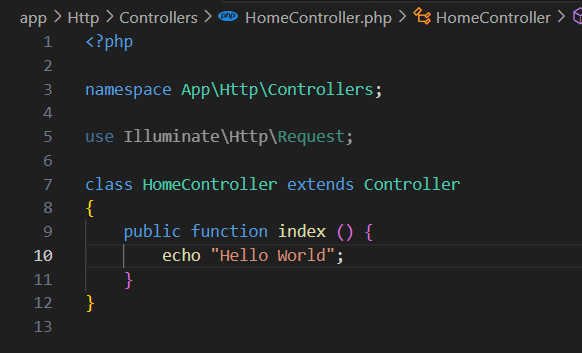


Carpeta de ubicación:

A screenshot of a computer

Description automatically generated

Creamos un método para probar si estamos llamando correctamente el controlador



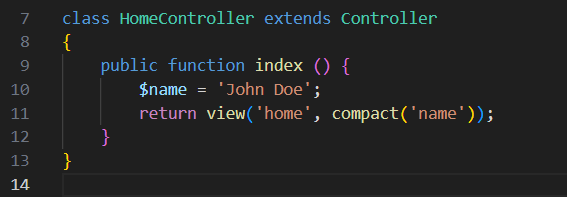
Como se llama el controlador por la ruta

A screenshot of a computer program

Description automatically generated

Ejecutar el servidor

Ahora en ves de devolver un mensaje vamos a devolver una vista



A screen shot of a computer

Description automatically generated

Se pueden devolver mas vistas de un mismo controlador

A screen shot of a computer program

Description automatically generated



Ejercicio pasar una variable por el path de la ruta y agregarlo a la vista

Para crear un controlador con método predefinidos llamado “Controlador de recursos”



A screenshot of a computer

Description automatically generated

* El método index del controlador se utiliza para mostrar la información de un tabla.
* Créate para crear formulario
* Show mostrar
* Lo demás de CRUD

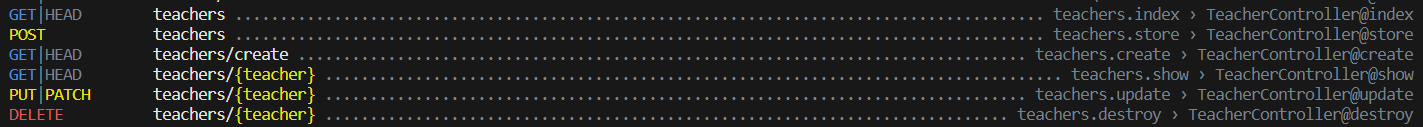
En el caso de llamarlo por la ruta no se definen los métodos que queremos llamar debido al poner ::resource tenemos disponible todos los métodos



En este caso vemos que el controller tiene 1 ruta pero no es así debido a que si ponemos el comando



Nos daremos cuenta que laravel crea todas las rutas de los métodos que tenemos disponibles



Se pueden crear controladores invocables lo que quiere decir que solo tienen un método se lo hace con el siguiente código



A screen shot of a computer screen

Description automatically generated

Como se puede observar para llamarlo en la ruta no se define el método porque solo tiene un método



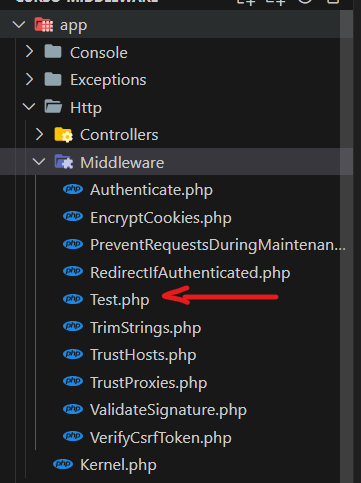
**MIDDLEWARE**

Un middleware es una clase que se ejecuta antes de que una solicitud HTTP llegue a un controlador. Los middlewares se pueden usar para realizar una variedad de tareas, como:

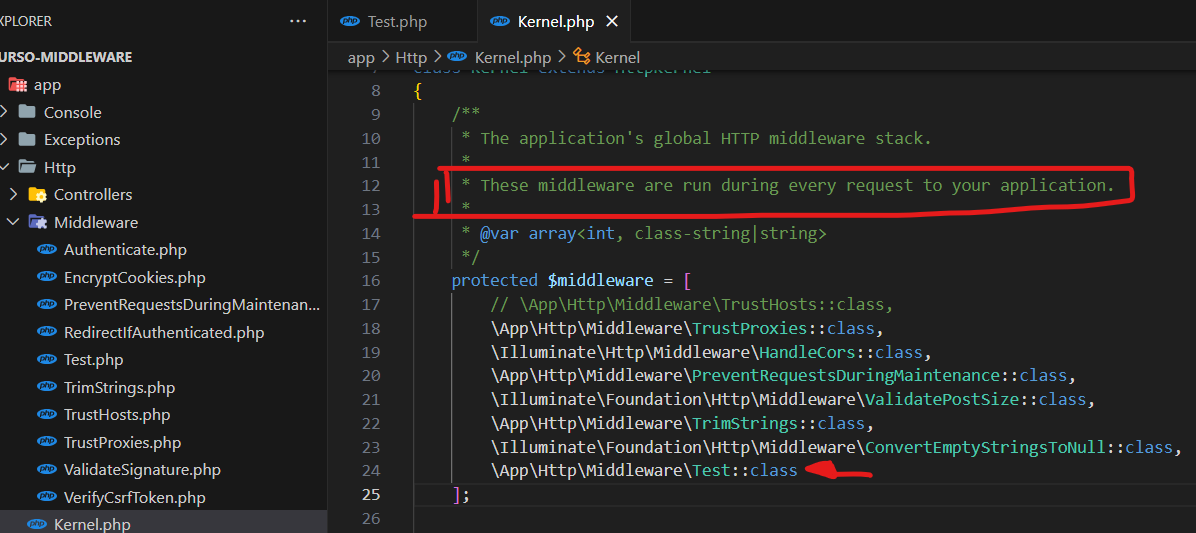
* Autenticación: verificar que el usuario que realiza la solicitud está autenticado.
* Autorización: verificar que el usuario que realiza la solicitud tiene los permisos necesarios para realizar la acción solicitada.
* Validación: verificar que los datos de la solicitud son válidos.
* Registro: registrar la solicitud para fines de análisis o auditoría.
* **Middleware Global**

****

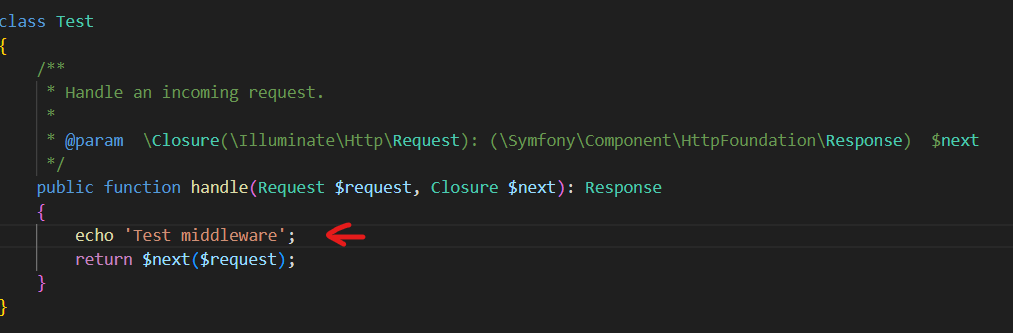
Carpeta donde se encuentra:



Los middleware globales se van a ejecutar durante la ejecución del programa por lo tanto importamos el middleware creado



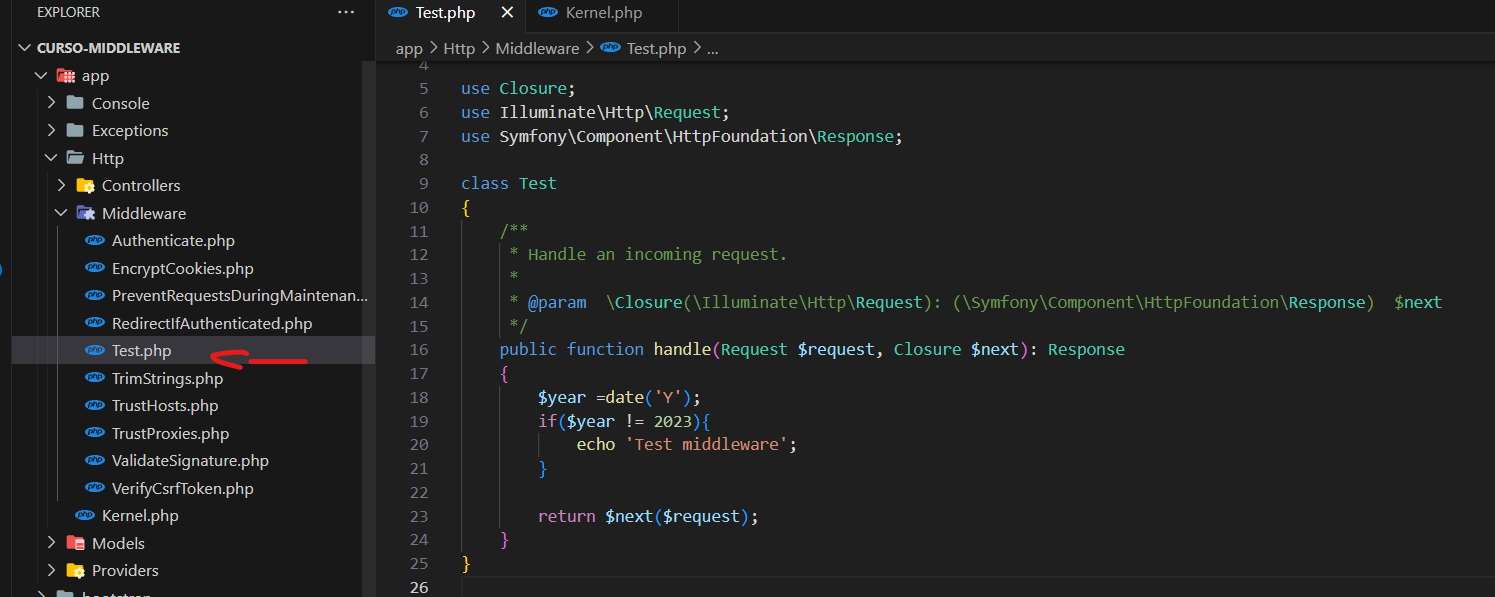
Como el middleware es global va aparecer rápidamente el mensaje



**Ejercicio prender servidor**

**Ejercicio crear otra vista y observar**

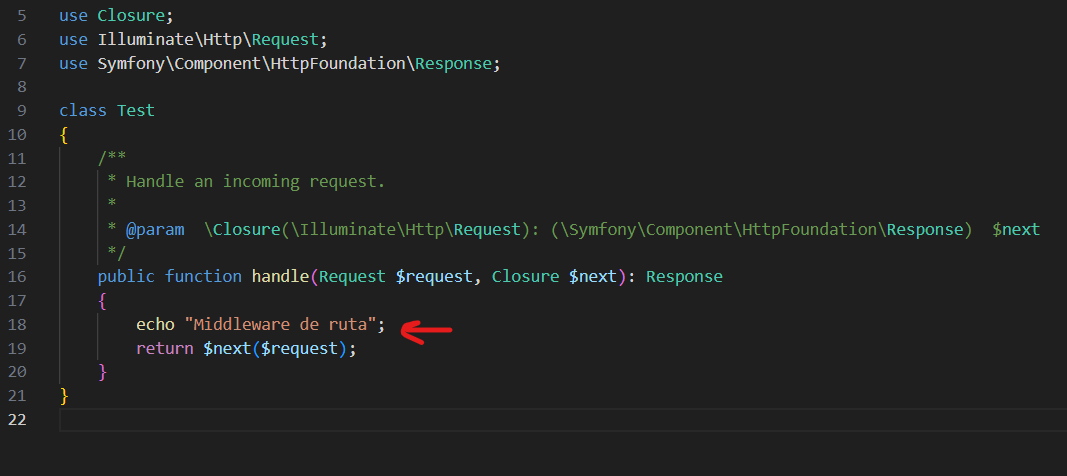
También se pueden agregar condicionales



**Ejercicio retornar una vista**

* **Middleware de ruta**

Ayuda a trabajar con rutas especificas en lugar de todas como el global

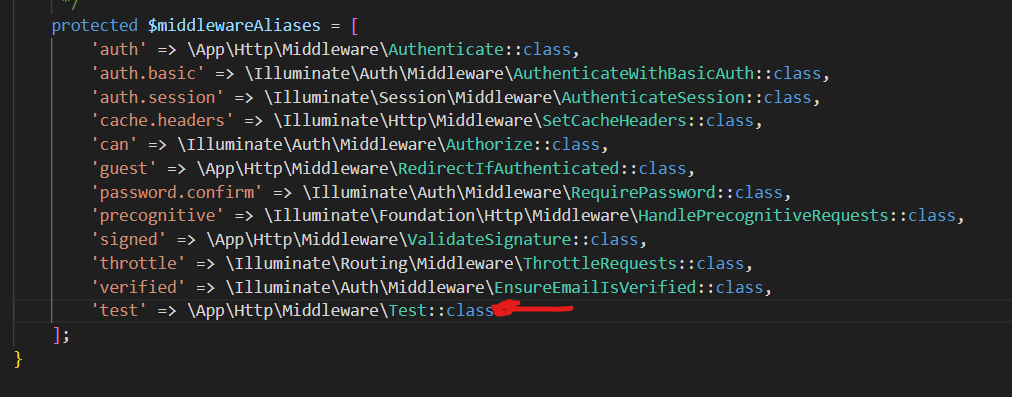
****

Eliminamos:

**A screen shot of a computer code

Description automatically generated**

Lo agregamos en la parte del ultimo método del mismo archivo kernel



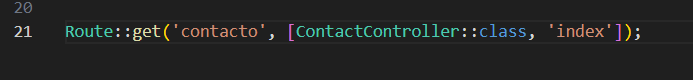


**VALIDACION DE FORMUARLIOS**

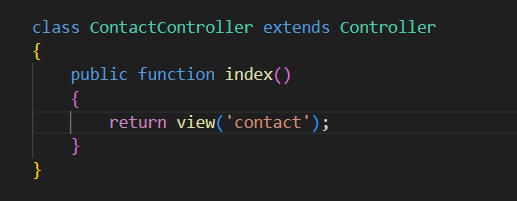
Creación del controlador



Lllamar a la ruta



Creamos el método index la vista contact para llamarla



**Ejercicio: Probar si funciona**

Creamos un formulario en contact.blade

A computer screen shot of text

Description automatically generated

* **CSRF TOKEN**

A continuación, vamos a definir la ruta donde el formulario se ira tras ser enviado el name es el mismo que se pone en la ruta del formulario.



Creamos el método store en el controlador contacto

A black screen with white text

Description automatically generated

En el forms dirigimos la ruta y el método de envio



Antes de usar el token y enviamos el formulario sale:

A screenshot of a computer

Description automatically generated

Ponemos el token en el formulario

A black background with white text

Description automatically generated

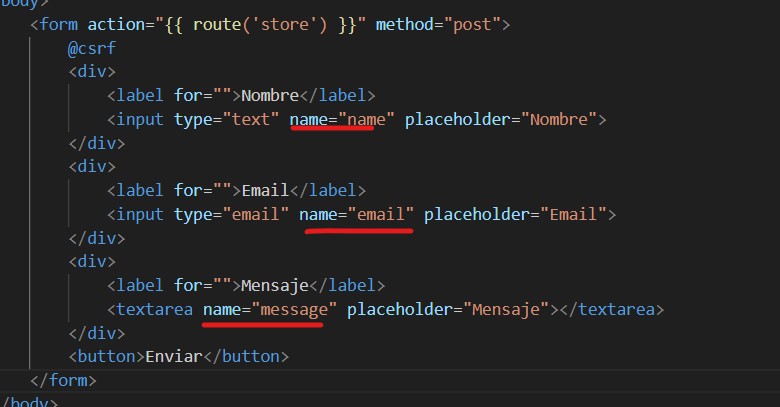
Activamos el inspector y vemos como funciona así cada usuario tendrá un único token y podrán realizar ataques

A computer screen with text and numbers

Description automatically generated

* **Validación de formularios**

Se lo hace mediante la propiedad name de la etiqueta



A computer screen shot of a program

Description automatically generated

Probamos a enviar el formulario y vemos el mensaje

A screen shot of a computer

Description automatically generated

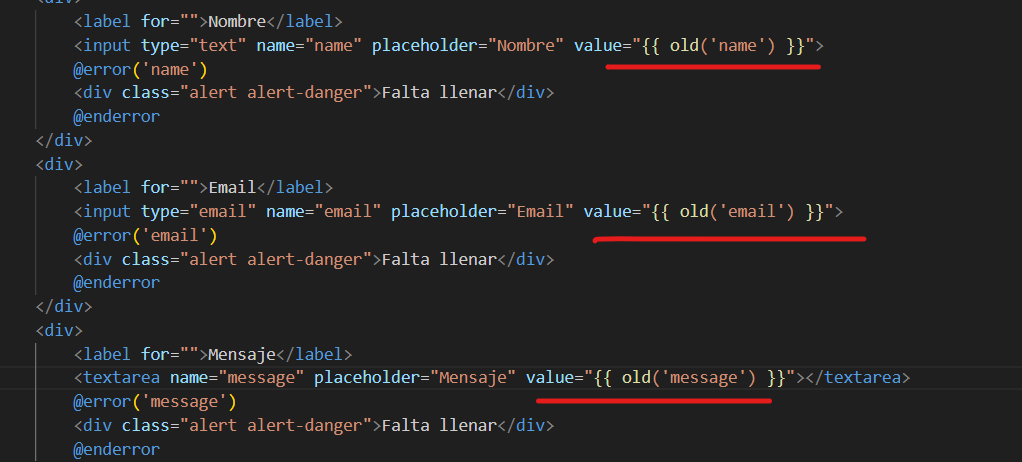
Ahora agregamos las validaciones



**Ejercicio: Cambiar la ruta para que sea contacto/store al momento de enviar y probar**

* **Guardar datos al envio incorrecto**

Lo que va a suceder es que si se envia mal el formulario los datos se van a guardar para no estar llenando todo de nuevo.



**Ejercicio poner estilos a todo el formulario y a los mensajes de error**

* Valaciones por inputs

A computer screen shot of a program

Description automatically generated

También se puede escribir en forma de array

A computer screen shot of a code

Description automatically generated

**Ejercicio agregar inputs para agregar fecha y edad y validarlos son (date, numeric)**

* Validacion expresiones regulares