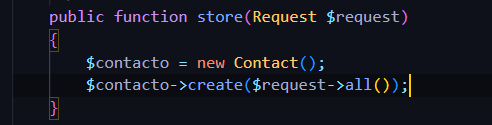
**Guía 3 – Usando el componente de Vue**

Modificando el método Store del controlador

1. Agregamos el siguiente código:



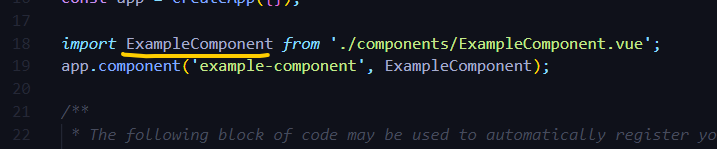
Aquí lo que se hizo fue usar el método create() dentro de store() para almacenar todos los datos que vengan desde la vista de formulario respectiva en la tabla contacts. Es una forma abreviada de trabajar.

|  |
| --- |
| Para una mejor comprensión de lo anterior, observe el siguiente ejemplo antes de continuar con el Ejercicio actual:  Se tiene una tabla llamada movies (o películas) donde se quiere guardar entre otras cosas el nombre y la descripción de una nueva película. El modelo respectivo es Movie. Normalmente lo haríamos así:    Pero se puede hacer usando menos código, así:    Fuente: <https://styde.net/aprende-a-usar-eloquent-el-orm-de-laravel/>  De todas maneras, si desde el formulario de creación vienen archivos como imágenes u otros, puede presentarse un conflicto por lo cual se puede usar en vez de all() el método except(), para que se guarde todo excepto el archivo y así poder procesar el guardado del archivo de otra manera. |

**Codificación del componente Vue**

Si vamos a resources/js, encontraremos el archivo principal llamado app.js.

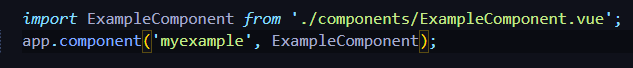
Dentro de ese archivo existe el siguiente código:



Como puede observarse, existe un componente llamado, entre comilla simple, ‘example-component’, cuya programación está dentro de un archivo llamado ExampleComponent.vue. Claro, podrían crearse otros componentes, todos los que se deseen, pero este se puede reutilizar.

Siga estos pasos:

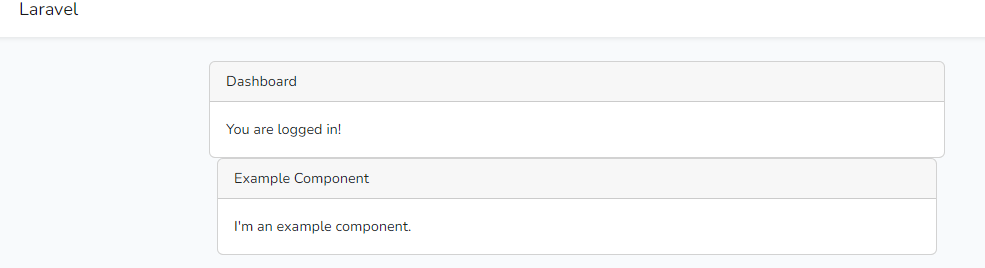
1. Cambie lo que está entre comillas por el siguiente nombre ‘myexample’



1. Luego vaya resources / views / homeblade.php e incluya este componente. Se incluye similar a como se haría con una etiqueta html. Se ha hecho la inserción antes de cerrar el div inicial. Observe:



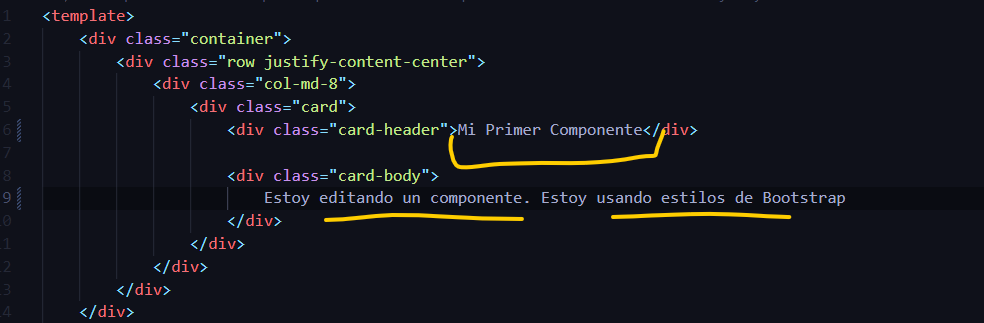
1. Vaya a la aplicación en el navegador web, registre un nuevo usuario y luego inicie sesión. Es la única manera de acceder a la vista home. Recuerde tener corriendo en dos terminales los servidores de backend y frontend, con php artisan serve y npm run dev. Verá algo como:



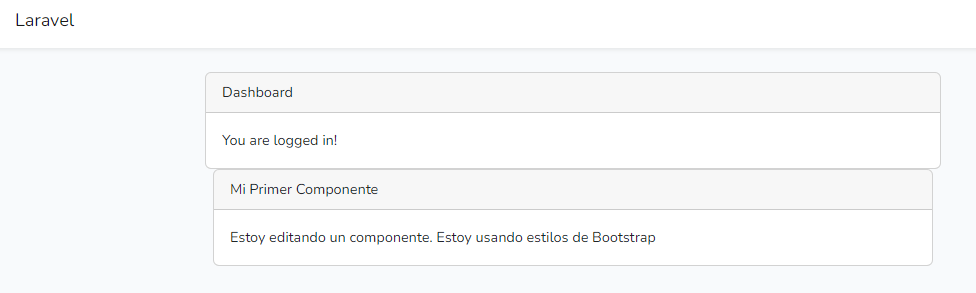
1. Realice alguna modificación en el componente original. Vaya a resources / js / components y ubique el componente:



Y modifique



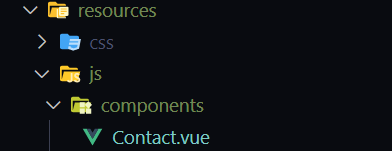
1. Guarde y revise cómo se ve en la página web. Los cambios deben verse reflejados automáticamente si el sevidor de frontend está corriendo.



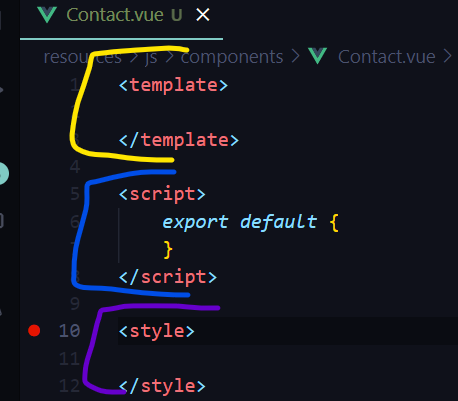
POR AHORA ES UN COMPONENTE QUE NO IMPLEMENTA ABSOLUTAMENTE NADA DE JAVASCRIPT, LO CUAL ES NUESTRO OBJETIVO FINAL.

**Creando un Componente de cero y su respectivo llamado**

1. Se puede borrar el componente existente y crear otro llamado Contact.vue o cambiarle al nombre al existente pero borrar prácticamente casi todo su contenido. Por estándar el nombre empieza con letra mayúscula:



1. El componente debe tener dos partes importantes que son el template (que tiene los elementos html), y el script (que contiene la parte lógica). Opcional, a menos que se vaya a utilizar Bootstrap, llevará un Style. Tener en cuenta esto para la competencia. Digítelo así en el archivo Contact.vue:



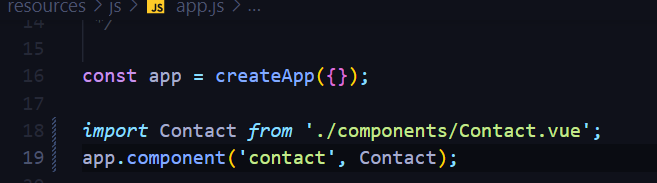
1. En el template digitamos lo siguiente:

Una captura de pantalla de un celular con texto e imagen

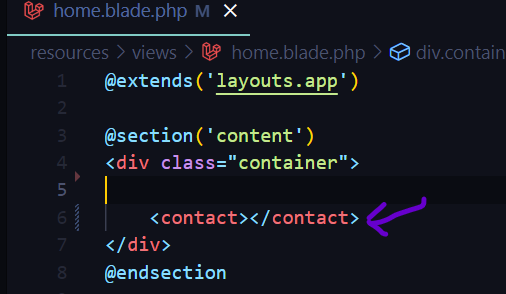
Descripción generada automáticamente con confianza media

En este caso se invocó una clase de Bootstrap para centrar el texto.

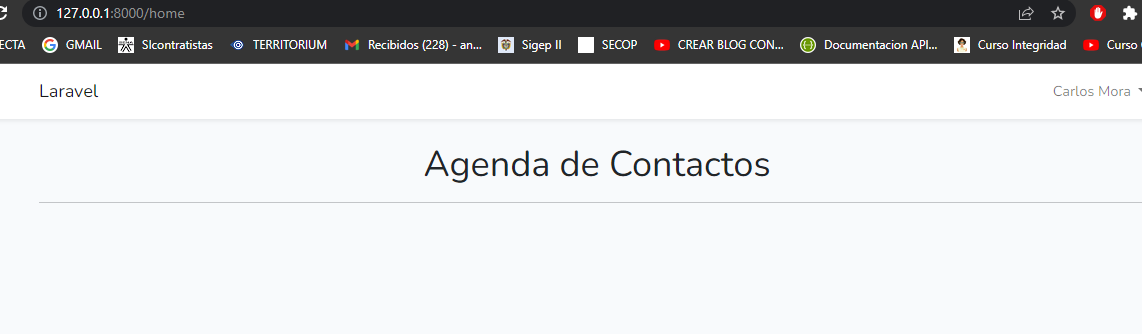
1. Hacemos el cambio respectivo en app.js que está en la ruta resources/js de la siguiente manera:

Como puede apreciarse, cambiamos el nombre del componente cuando se está invocando, es decir, lo que está en comilla simple (‘contact’). En cuanto al import también lo cambiamos, pero en realidad, de momento, no afectaría en nada sino cambiáramos ese nombre. Este mismo import, se coloca tal cual después de la coma, en el paréntesis. Debe corresponderse con el nombre del archivo, por eso empieza con mayúscula (Contact).

1. Lo llamamos desde la vista [home.blade.php](http://home.blade.php), pero antes borramos prácticamente todo el contenido de ese archivo. Debe quedar así (no olvidar guardar):

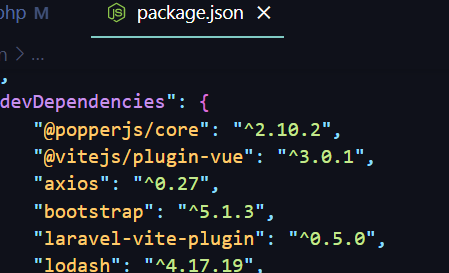


Nuestra aplicación básicamente se verá como sigue:



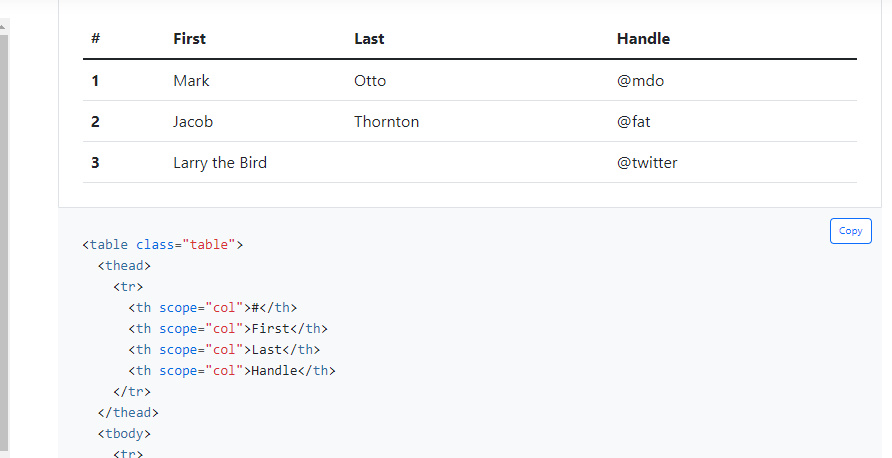
**Añadiendo la tabla para mostrar los contactos**

1. **Vamos a la página oficial de Bootstrap.** Debemos ubicarnos en la versión indicada en nuestro archivo package.json de Laravel, que para nuestro caso a agosto de 2022 es 5.1.3

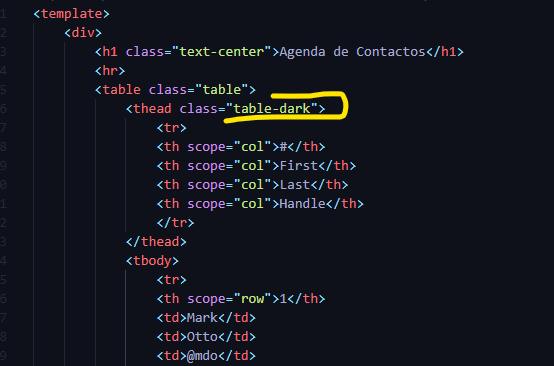
****

Captura archivo package.json 1

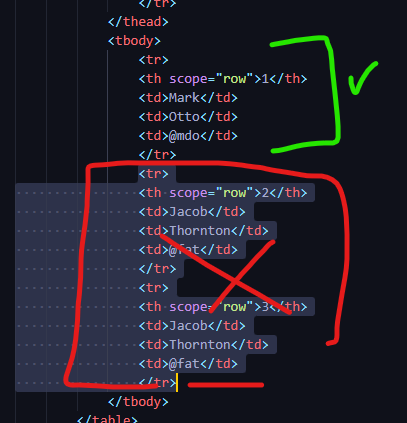
1. Ubicamos content->Tables en la página de Bootstrap y copiamos el código de la primera tabla:

****

1. Pegamos en el archivo Contact.vue y hacemos algunas modificaciones al código:

****

**Aparte del thead, solo necesitaremos una fila más. Por eso borramos los otros <tr> con sus respectivos <td>:**

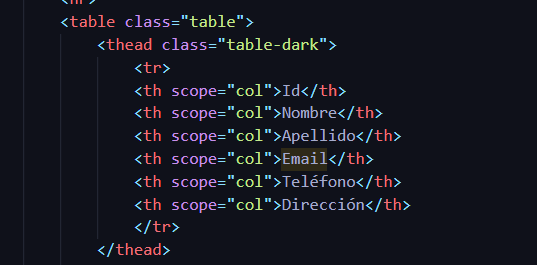


1. Observamos el resultado en pantalla:

Interfaz de usuario gráfica, Texto, Aplicación

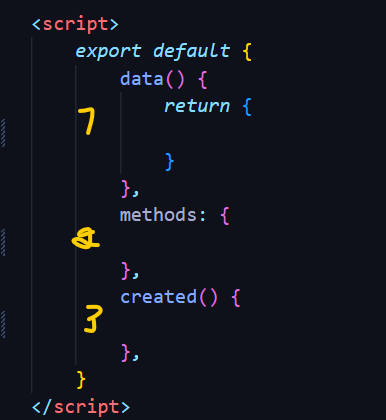
Descripción generada automáticamente

1. Ahora vamos a colocar los encabezados que necesitamos. Si se requieren más <th> se agregan:



**Trabajando el Script del Componente**

1. En este caso nos dirigimos a la sección del script, y creamos el esqueleto con algunas de las muchas partes que puede tener un script de esta clase:

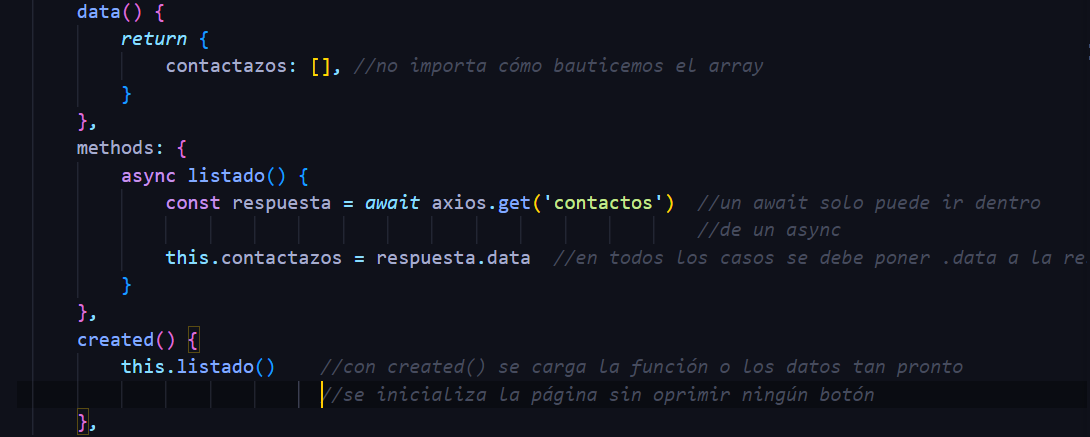


En el data() deben ir los atributos o variables. En vez de igual (=) se usa dos puntos (:).

En methods() deben ir todos los métodos a utilizar. Ya no se necesitará la palabra reservada function.

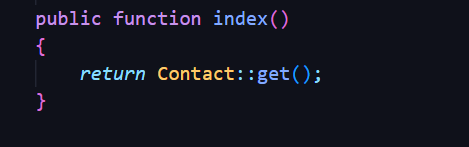
Luego pueden ir otros elementos, como por ejemplo ciclos de vida (consultar la documentación oficial en <https://vuejs.org/guide/essentials/lifecycle.html>). En este caso se ha colocado un ciclo de vida que se llama created() de los varios que pueden usarse. Otro tipo de elementos que puede usarse es computed.

1. Digite las líneas respectivas dentro de cada elemento como se ve a continuación:



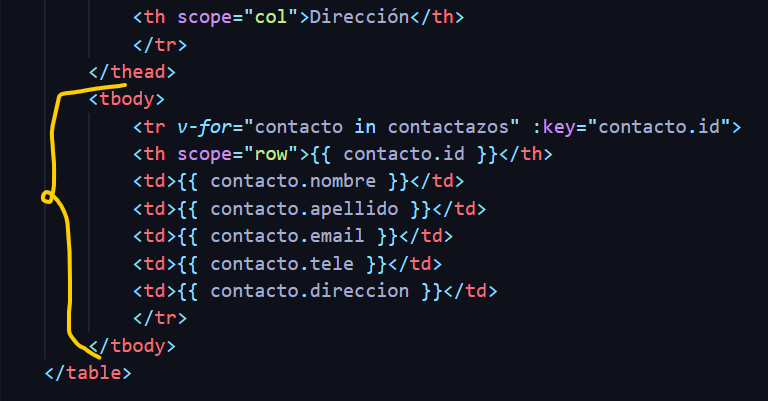
Con la librería axios podemos hacer una petición, en este caso a la ruta contactos, que hace parte del mismo proyecto y que retornará todos los contactos de la tabla contactos. Esto se logra gracias a la rutian programa en el index del controlador ContactController.

1. En el index, del mencionado controlador, debe ir lo siguiente:



Sin esta petición desde el index será imposible que la ruta ‘contactos’ llamada con axios desde el componente retorne dato alguno.

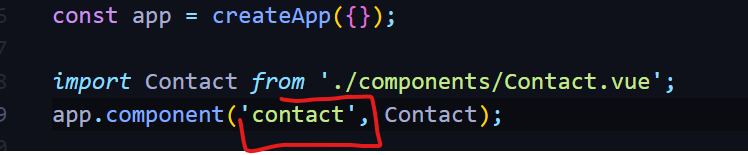
1. Volviendo al componente de Vue, vemos que uno de los elementos digitados es created() el cual es un ciclo de vida de Vue, el cual permite cargar la información sin necesidad de que el usuario oprima algún botón o link. En este caso, se hizo así para que la información cargue inmediatamente se ingrese a la ruta.
2. Finalmente, en el mismo componente, subimos un poco nuevamente a la sección de template. Allí utilizaremos un v-for para iterar el array contactazos[] previamente creado en el script. Además, este v-for debe llevar un :key el cual corresponderá a la llave primaria de la tabla que se quiere iterar.



**Llamar el componente desde alguna de nuestras vistas.**

Esto es bastante sencillo.

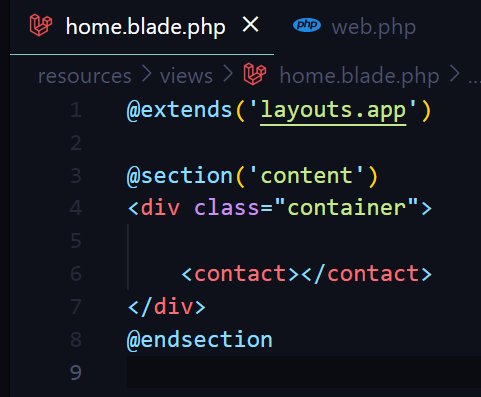
Primero, recordemos cuál es el nombre del componente. En app.js había hecho anteriormente la siguiente modificación:



Lo encerrado en rojo es el nombre con que se puede exportar el componente.

En este caso queremos ver el listado de contactos, desde la vista home. Esta vista solo puede accederse si se está logueado. De todos modos, podríamos llamar el componente desde otro lugar si quisiéramos.

1. En resources / views buscamos [home.blade.php](http://home.blade.php) y borramos prácticamente todo el contenido de allí. Luego llamamos el componente como si fuera una etiqueta más, así:



1. Guardamos todo. Encendemos los servidores de backend y de frontend (este último con npm run dev), si es que los habíamos apagado. Registramos un usuario y luego iniciamos sesión con él.
2. Ya deberíamos ver en una tabla todos los contactos:

