**Informe técnico creación de modulo docentes**

Julian David Anturi Duque

Carlos Andrés Mora Agudelo

CENTRODE DISEÑO E INNOVACION TECNOLOGICA INDUSTRIAL

DOSQUEBRADAS

SENA

11 y 12 DE AGOSTO DE 2022

Senita soft solicita elaborar un nuevo módulo para la los docentes de la academia, para tener el registro de los datos personales de cada instructor del curso.

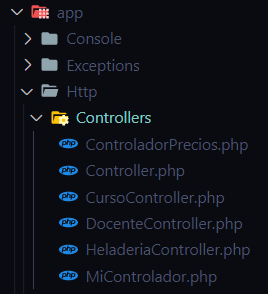
Para tener control de cada registro, se solicita que se muestre dentro de una tabla los instructores y se puedan hacer las respectivas operaciones del CRUD.

Para su elaboración se siguieron los pasos:

1. Para crear el controlador de recursos requirió el siguiente comando:

**Php artisan make:controller DocenteController –resource**

1. En la ruta App/Http/Controllers se pueden encontrar los controladores



1. Se procedió a crear la ruta en el archivo web.php para conectar este nuevo controlador a la aplicación, como se ve a continuación:

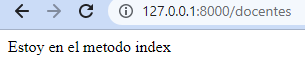
Se hizo el llamado:



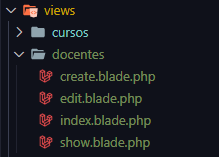
Se creó la ruta:



1. Se verifico la conexión realizada



1. Posteriormente se crearon los archivos index, create, edit y show para las vistas en una nueva carpeta denominada docentes



1. Se realizó la modificación al navbar creado en el archivo app.blade para agregar el direccionamiento a la vista index

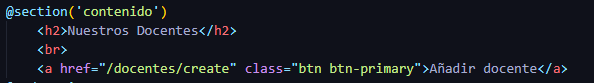


1. Se modificaron las vistas index y create heredando los estilos de Bootstrap desde el archivo app.blade que se encuentra en la carpeta layouts, como se muestra a continuación:

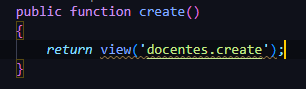
* En la vista create se agregó el formulario solicitado por el cliente, con los campos nombre, apellido, título universitario, edad, etc:



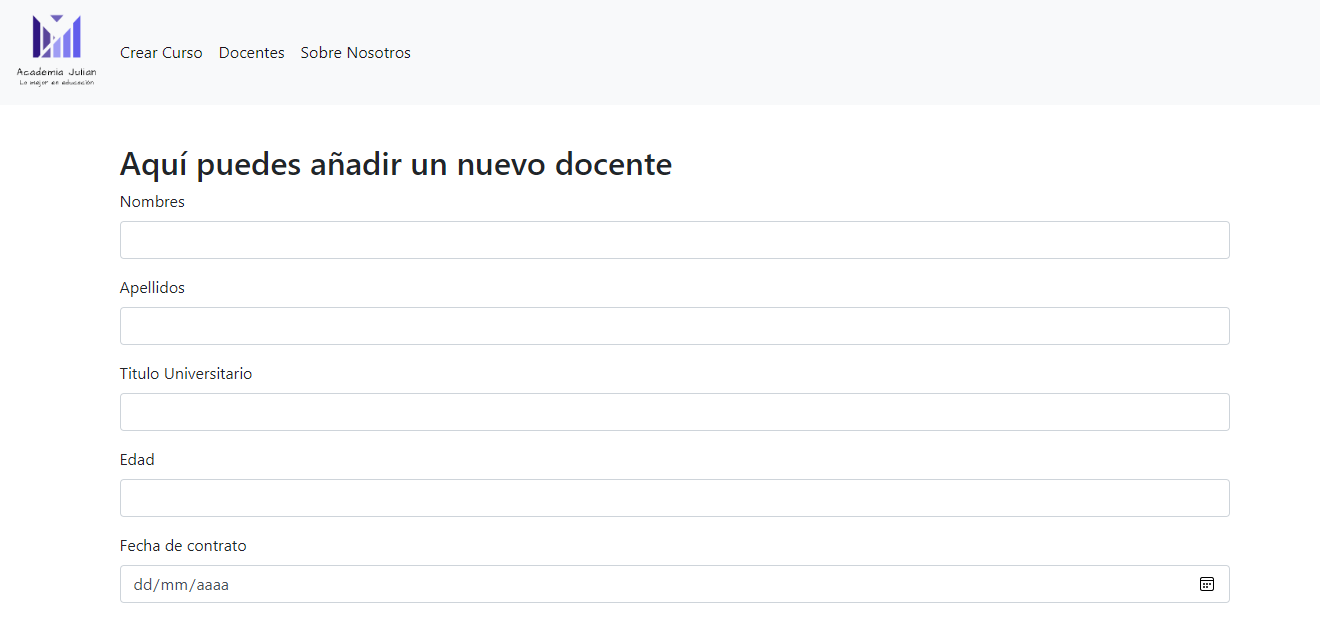
* En la vista index se agregó un botón para dirigirse al formulario



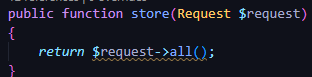
1. Posteriormente se ingresó al método create del controlador para retornar la vista create



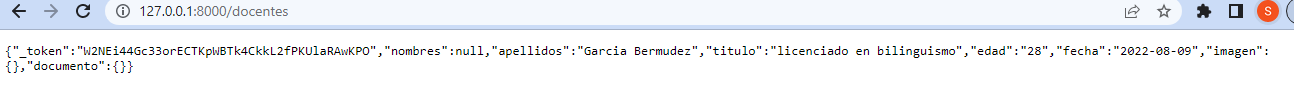
Obtenido el siguiente resultado:



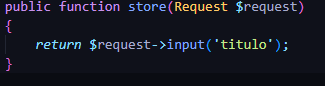
1. Se ingresó al método store del controlador y se agregó el request



Obteniendo el siguiente resultado al llenar los campos del formulario:



1. Posteriormente se realizó la validación, para determinar si el request si recibe los datos del formulario, de la siguiente forma:

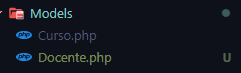


Obteniendo el siguiente resultado:

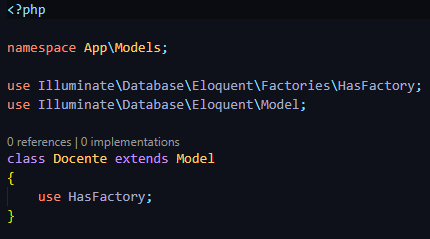


**Creación del modelo**

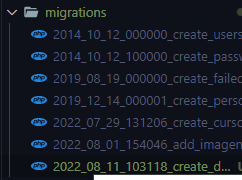
1. El modelo fue creado usando el comando php artisan make:model nombre –m, a la cual se le dio el nombre de Docente y se aloja en la carpeta models.

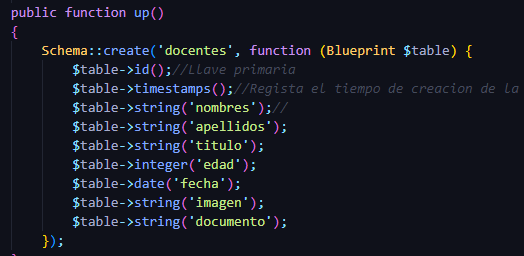


Y el archivo contenía lo siguiente:

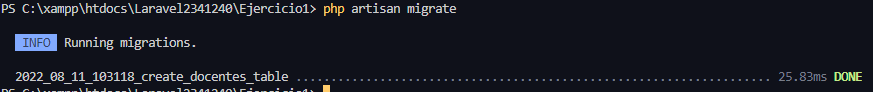


1. Automáticamente se crea el archivo de migración en el cual agregamos los campos de la nueva tabla, como se aprecia a continuación:

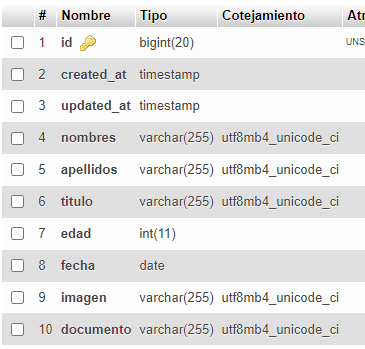


Se agregaron los campos con los nombres del formulario creado en la vista create:

1. Se hace la migración de la tabla la base de datos por medio del comando php artisan migrate

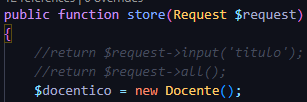


1. Una vez realizada la migración entramos a la base de datos y revisamos la estructura de la tabla docentes

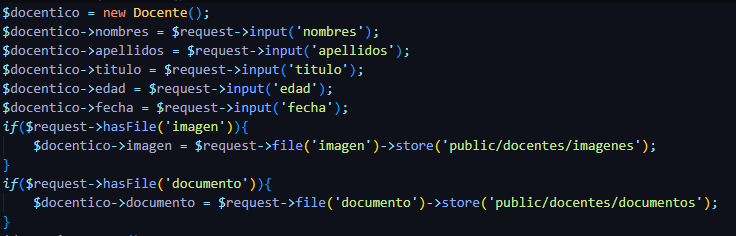


**Trabajo con store()**

1. Se ingresó al método store del archivo DocenteController, y en el se creo la instancia de la clase Docente llamada docentico.



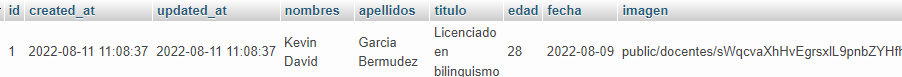
1. La instancia actúa como un array, por lo cual se agregan los nombres de posición los cuales coinciden con las columnas de la tabla Docentes.
2. Cada nombre del arreglo se almacena los campos que vienen desde el formulario. Se igualo con el input como se ve en la captura:



1. Finalmente se almacena en la tabla Docente los datos almacenados en el array docentico, por medio del comando save(). Y retornamos el mensaje.



1. Obteniendo el siguiente resultado al cargar los datos del formulario en la base de datos:

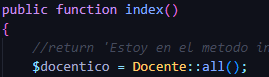


**Programación de la vista index**

La vista index permite ver todos los registros que se encuentran en la tabla en cuestión, para este cado Docentes

Se siguieron los siguientes pasos para su modificación:

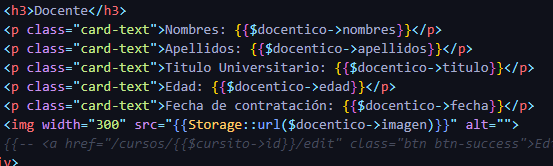
1. Se ingresó al controlador de recursos, posteriormente se ubicó el método index y se creó la instancia docentico.



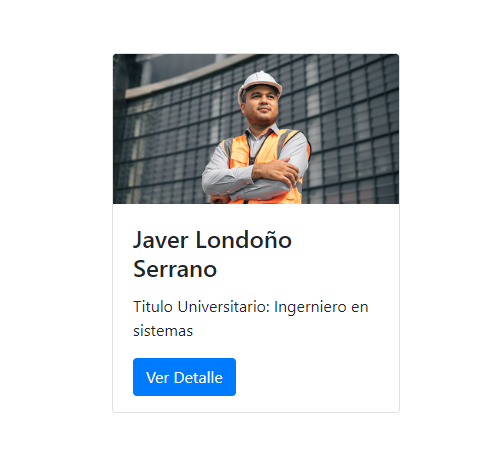
1. Posteriormente se adjuntó la instancia a la vista con el método compact



1. Luego en el index se realizó el llamado al array, iterándolo mediante un foreach y asignando un alias. Y dentro del mismo se agregó la información que queremos llamar de la tabla



Obtenido el siguiente resultado en la vista index:

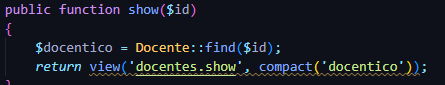


**Programación de la vista show**

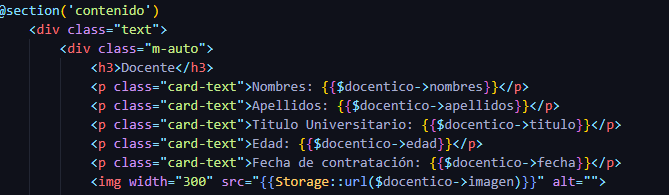
Esta vista nos permite ver toda la información sobre os docentes de la academia, por medio de la conexión a la base de datos

Se siguieron los siguientes pasos:

1. Se ingresó al controlador de recursos, posteriormente se ubicó el método show y se creó la instancia docentico.
2. Se añadió la instancia a la vista por medio del método compact. Como se aprecia a continuación:



1. Posteriormente se ingresó al archivo show.blade y se modificó la vista, haciendo el llamado a los registros almacenados en la tabla



**Visualizador de PDF**

En este punto era necesario que la vista retornara el archivo pdf para esto gue necesario usar la etiqueta (<iframe>) que permite incrustar, y por medio de la conexión a la tabla hacer el llamado al documento almacenado. Como se puede ver a continuación:



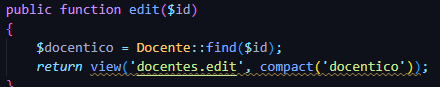
Obteniendo que la vista retorne todos los registros almacenados y se pueda visualizar el pdf.

**Programación de la vista edit**

Esta vista nos permite editar los registros almacenados en la tabla, por lo general contiene los mismo que la vista create

Se siguieron los siguientes pasos:

1. Se ingresó al controlador de recursos, posteriormente se ubicó el método edit y se creó la instancia docentico.
2. Se añadió la instancia a la vista por medio del método compact como se ha hecho con las otras vistas creadas. Como se aprecia a continuación:

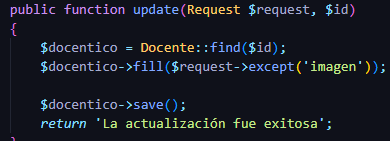


1. Posteriormente se ingresó ala vista edit.blade y se realizaron sus respectivas modificaciones

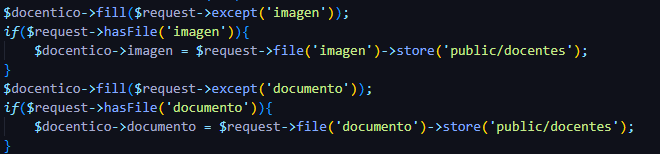


**Actualización de datos**

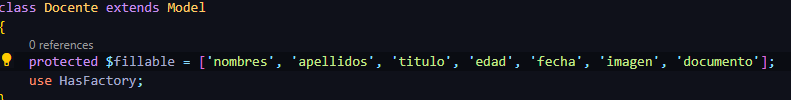
1. Se ingresó al controlador, se ubicó el método update, y se especificó la información a actualizar, por medio de la creación de la instancia



1. Posteriormente se exceptuaron los campos de imagen y documento, por medio del método fill y validar si venia una imagen o un documento en el request como se hizo en el método store:

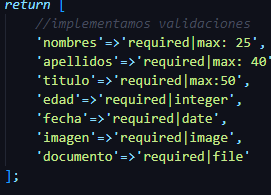


1. Se agregaron permisos de manipulación de campos en el modelo de Docente que representa a la tabla Docentes

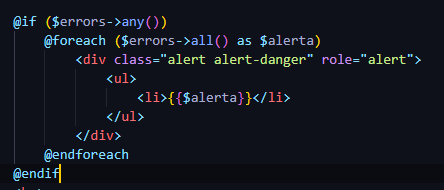


**Creación de clase para el request**

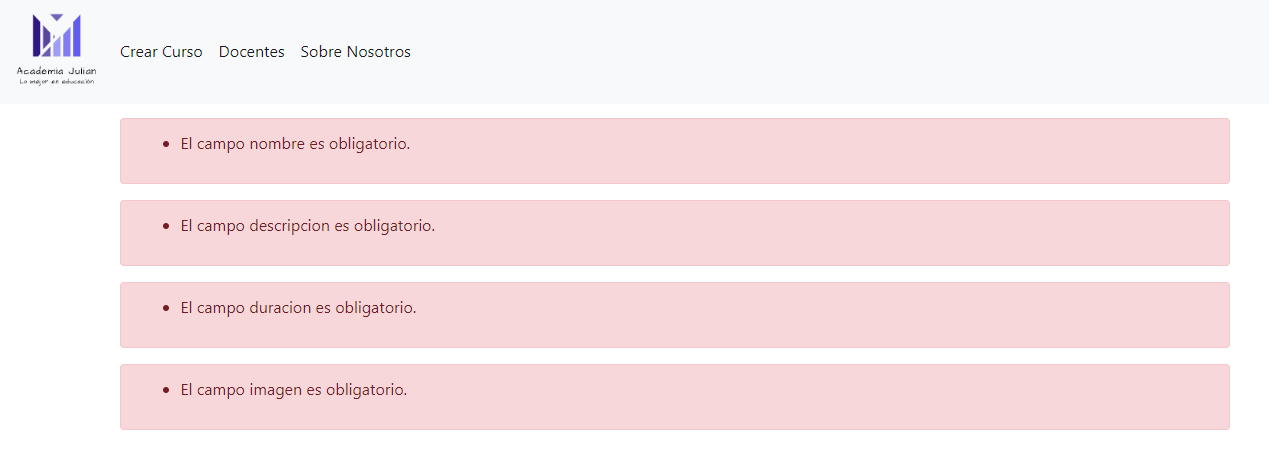
1. Se creó el archivo para la request por medio del comando php artisan make:request storeDocenteRequest
2. Dentro del método authorize se cambió el false por un true.
3. En el cual se ubicó el método rules y se agregaron las restricciones para los campos del formulario



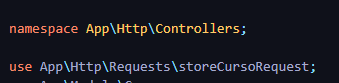
1. Se agregaron los ciclos dentro de la vista create.blade, para que al ingresar los datos indebidamente salga su respectiva alerta



Obteniendo el siguiente resultado:



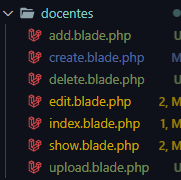
1. En el controlador de recursos se ubica el método store y se cambia request por el nombre de la nueva clase storeCursoRequest 
2. Y automaticamente despues del punto anterior se crea la conexión en la parte superior del controlador



**Mensajes de alerta**

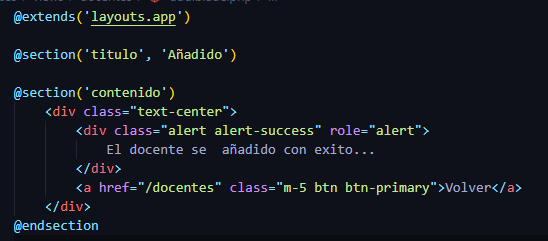
Se solicitó que para las acciones crear, actualizar y eliminar, se imprimiera una alerta visualmente agradable para el usuario, por lo cual se siguieron los siguientes pasos:

1. Se crearon tres vistas nuevas, llamas add, upload y delete.

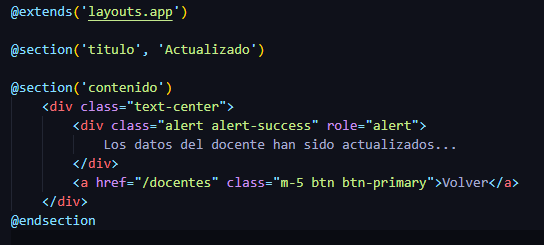


1. En cada vista se heredó la plantilla del archivo app.blade
2. Posteriormente se agregaron las alertas tipo Bootstrap para cada vista, además de un botón que retorne a la vista index, como se puede ver a continuación:

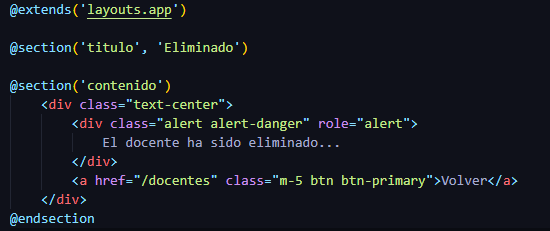
**Vista add.blade:**



**Vista upload.blade**



**Vista delete.blade:**

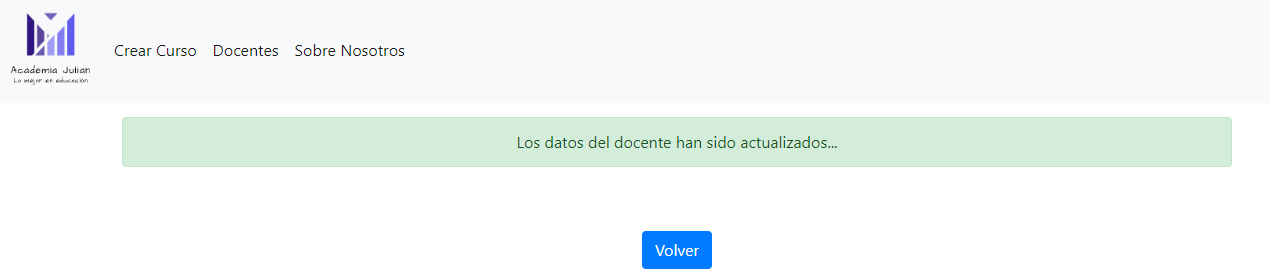


**Resultado de alertas**

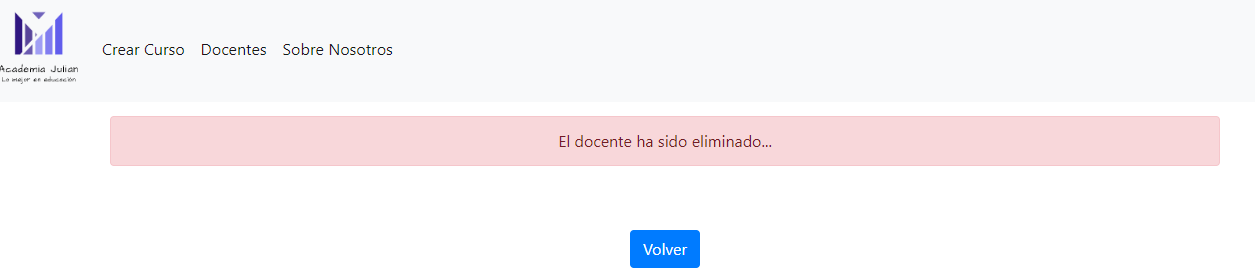
**Vista add.blade:**

****

**Vista upload.blade:**

****

**Vista delete.blade:**

****