El objetivo de este material es mostrar el uso del Framework Larval en su versión 4.2, aunque para versiones posteriores puede que no hay una cambio tan notorio, primeramente Laravel es un Framework PHP, el cual facilita el desarrollo de aplicaciones web, ofreciendo grandes ventajas en comparación con la programación PHP puro.

Antes de comenzar es importante conocer el concepto de Framework, el cual consiste en una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software.

Un Framework en la mayoría de los caso pude incluir soporte de programas, bibliotecas y un lenguajes interpretado, entre otras herramientas que ayudan a desarrollar y unir diferentes componentes de un proyecto.

Entre las ventajas de la utilización del Framework Laravel de PHP está en la agilidad de desarrollo comparado, el uso del patrón MVC (Modelo Vista Controlador), lo cual permite tener una estructura de desarrollo más ordenada evitando mezcla un conjunto de tecnologías y como hojas de estilo, lenguaje del lado del cliente y servidor en un mismo archivo separando la lógica de las vistas, además de que facilita la extensibilidad o que el software siga creciendo sin tener impedimentos. También Laravel permite una mayor seguridad en nuestra aplicación ya que sus sentencias SQL se encuentran protegidas por defecto ante ataques, y es muy fácil utilizar protección contra CSRF en los formularios al igual que escapar caracteres inválidos.

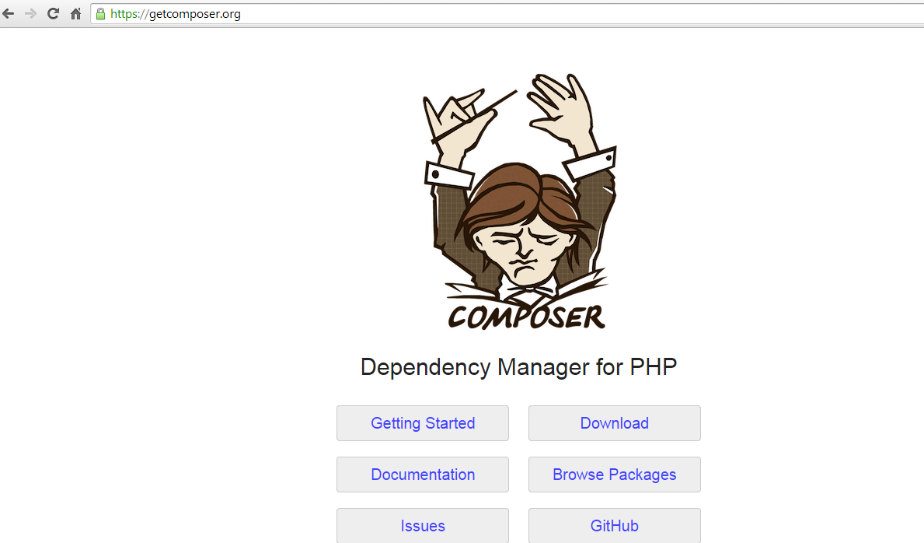
Otra característica muy interesante que cuenta Laravel es la utilización de modelos ORM los cuales tienen la finalidad de tratar las tablas de la base de datos como objetos lo que permite realizar las sentencias CRUD (CREATE READ UPDATE DELETE) de una manera muy sencilla y sin tener la necesidad de escribir las sentencias SQL, además otra ventaja es que el modelo ORM puede utilizar distintas bases de datos sin necesidad de modificar todo el código fuente algo sería imposible en una programación tradicional.

INSTALACIÓN

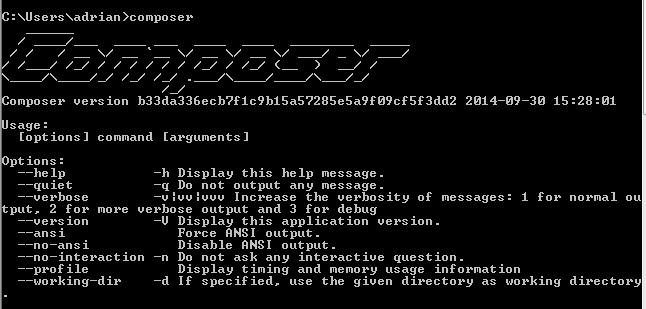
Para comenzar a utilizar el Framework Laravel de PHP, solamente debe de asegurarse que cuenta con los siguientes requerimientos:

* Tener una versión de PHP igual o superior a 5.3.
* Poseer el gestor de paquetes composer.

Composer es un gestor de paquetes de PHP que permite el manejo de dependencias PHP (librerías adicionales de PHP), similar al utilizado en otros Frameworks como PIP de Django, o para que se den una idea como el gestor de paquetes de distribuciones Linux donde escriben un comando en la terminal y este se encarga de localizar el paquete y descárgalo e instalarlo. Pueden descargarlo de la página http://www.getcomposer.org. Y los llevara a una página como esta.



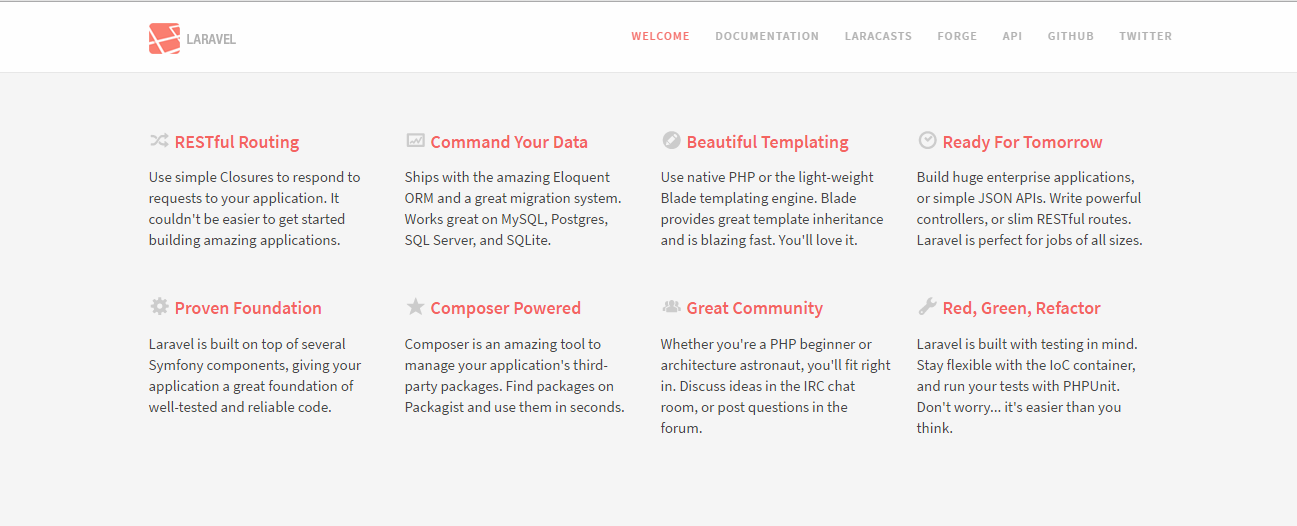
El proceso de instalación de composer en Windows es sumamente fácil solo es necesario definir la ubicación del archivo php.exe y dar un par de veces siguiente has que concluya la instalación. Una vez concluida la instalación es necesario comprobar que es posible utilizar el comando composer, para hacer esto habrá una terminal o consola y escriba la el comando composer si aparece un error deberá definrlo en variables del entorno, dirigiéndose a propiedades del sistema y configuración avanzada.



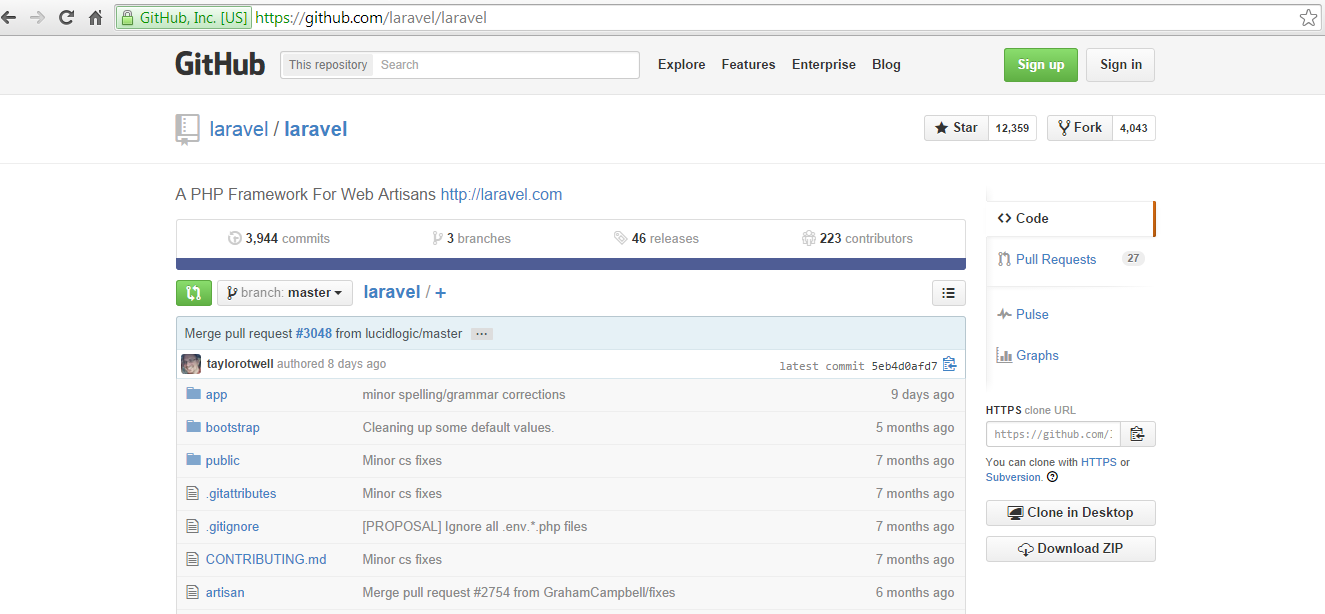
INSTALACIÓN DE LARAVEL

OPCIÓN 1

Para instalar Laravel existen dos maneras la primera es dirigiéndose a la página web de laravel en htpp://laravel.com y seleccionar Github. Esta acción lo dirigirá al repositorio público de Laravel el Github donde se encontrara la última versión de Laravel.



Una vez en Github solamente es necesario presionar el botón Download Zip y comenzarla la descarga de la última versión de Laravel.

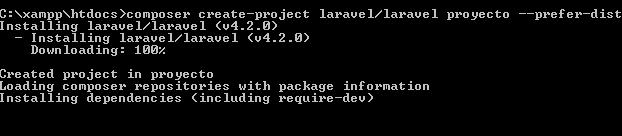


Una vez descargado en archivo Zip solamente falta descargarlo en la raíz de su sitio web de manera local el cual puede ser htdocs/ o www/ de acuerdo al paquete de servidor web que hayan instalado.

OPCIÓN 2

Para la instalación en esta opción usaremos el gestor de paquetes composer que instalamos anteriormente, el primer paso a realizar es abrir una consola y dirigirse a la carpeta htdocs/ o www/ de acuerdo a su máquina utilizando el comando cd y la carpeta.

Una vez en la carpeta es necesario ingresar el siguiente comando **composer create-project laravel/laravel nombre-del-proyecto –prefer-dist,** al igual que la instalación anterior descargara la última versión de composer en su sistema. Al presionar Enter comenzará la instalación.

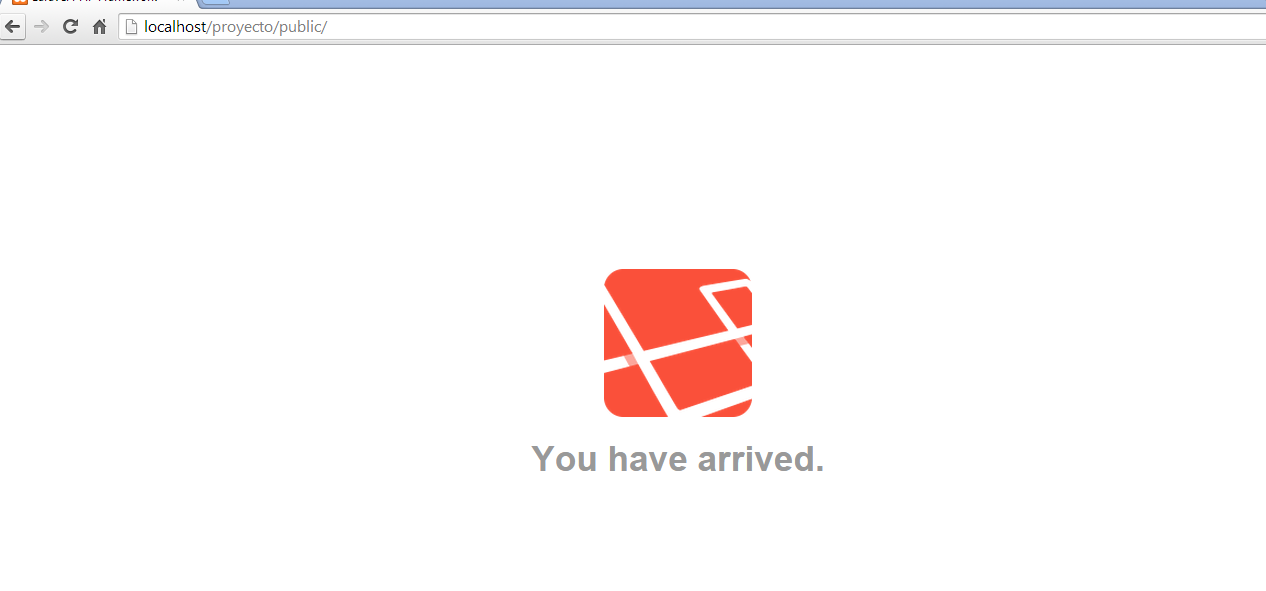


Una vez instalado Laravel por alguno de los pasos anteriores ya tienes instalado y utilizable el Framework Laravel.

ESTRUCTURA DE CARPETAS

COMPROBAR QUE LARAVEL ESTA FUNCIONANDO

Para comprobar que Laravel está corriendo en tu maquina solo basta con abrir algún navegador e ir a **localhost/nombre-de-tu-proyecto/public** y se mostrará una página como la siguiente, si aparece indica que todo está funcionado adecuadamente.

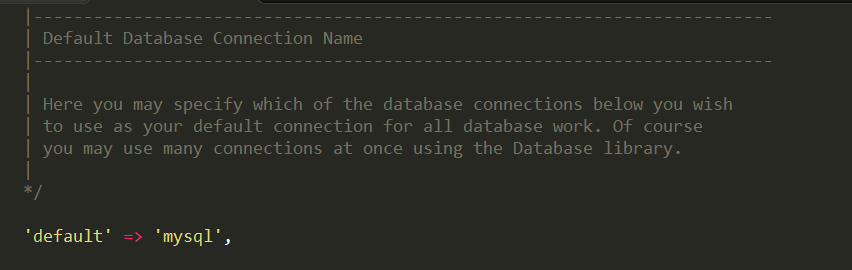


PRIMEROS PASOS CON LARAVEL

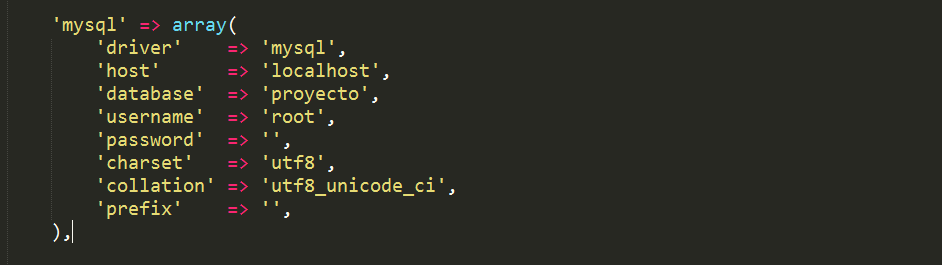
En este apartado se demostrará algunas de las funcionalidades más importantes que se pueden utilizar con el Framework Laravel, puede que parezca un poco complejo pero cuando profundice más en este documento lo encontrara más comprensible. En este ejemplo se planea tener una base de datos y obtener el primer registro obtenido, pero servirá para tener una mejor idea del proyecto y en que sección debería ir cada fragmento de código.

Para comenzar se necesita crear una base de datos, puede elegir entre varias opciones y en todas ellas el código seguirá siendo el mismo, en este caso usaremos mysql y la base de datos se llamara proyecto. Nada muy complejo.

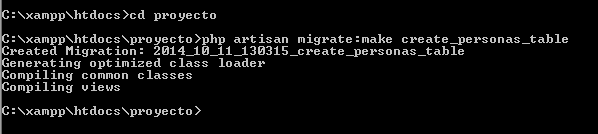
Ahora es necesario definir la base de datos a usar en nuestro proyecto para eso nos dirigimos a la carpeta **app/config/database.php**,por defecto esta seleccionado mysql como la base de datos a usar, si desea usar una distinta solo cámbielo por alguna de las demás opciones.



Además es necesario definir los parámetros de conexión a usar, entonces nos dirigimos un poco más abajo y encontraremos, las posibles conexiones a usar, y cambiamos la contraseña, usuario y base de datos de acuerdo a nuestra configuración, como se muestra. Y presionamos guardar.



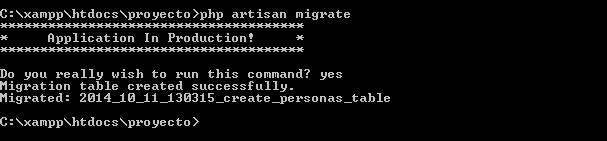
Una vez definida la base de datos, utilizaremos una herramienta que cuenta Laravel llamada migraciones lo que permite crear tablas en nuestra base datos y regresar de nuevo a su estado anterior, puede ser muy útil cuando no se tiene una base de datos bien definida o cuando el proyecto está siendo desarrollado por un grupo de personas siendo útil para tener un mejor control de los cambios. Para comenzar debemos utilizar nuevamente nuestra terminal y dirigirnos a la raíz de nuestro proyecto y teclear el siguiente comando **php artisan migrate:make create\_nombre\_de\_tu\_tabla\_create**, para tener un mejor control se recomienda utilizar un nombre en plural a la tabla a crear.



Ya que termino nos dirigimos a la siguiente estructura de carpetas **app/database/migrations/**  y dentro de ella encontraremos un archivo con la fecha actual vamos a abrirlo e ingresar el siguiente código.

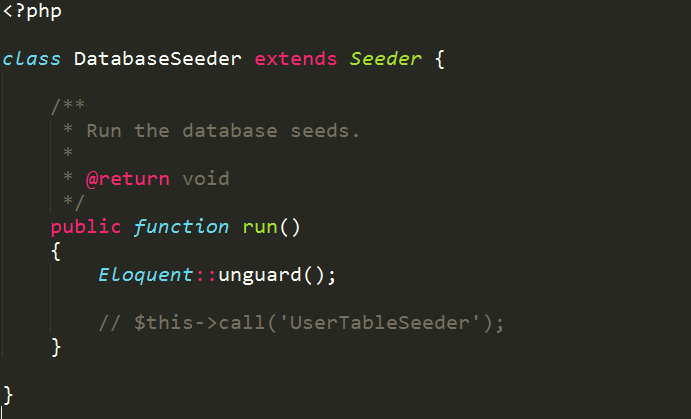


La función up es llamada cuando se ingresa el comando migrate y por consiguiente crear una tabla llamada personas, por defecto es necesario ingresar $table->timestamps() que Laravel utiliza aunque no se puede omitir, más adelante se describirá como. Y la función down será llamada cuando se quiera regresar a un estado anterior (borrar la tabla). Vamos a ver cómo usar esto. Volvemos a nuestra terminal e ingresamos el comando **php artisan migrate.**

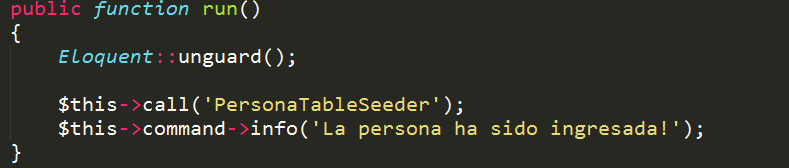
****

Ahora si comprueba podrá ver que ya cuenta con dos tablas la tabla que deseaba crear y la tabla migrations que se crear por defecto y es la encargada de volver las migraciones a un estado anterior o siguiente.

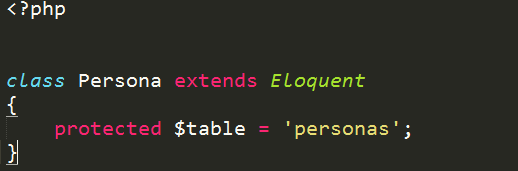
También es posible ingresar datos a nuestra tabla llamados seeds que es lo que realizaremos a continuación, para hacer esto es necesario ubicar el archivo **DatabaseSeeder.php** que se encuentra en **app/config/database/seeds/** este archivo tendrá un código como el siguiente:

cuenta

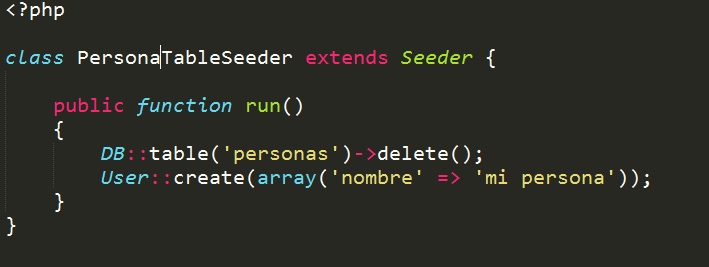
Como se muestra en la imagen anterior cuenta con una función llamada run, dentro de este método se debe ingresar la clase seed que se desea utilizar como se muestra en el código comentado en este caso escribiremos nuestra clase a utilizar y en la línea siguiente mostraremos un mensaje indicando que el ingreso de los datos fue exitoso. El código a usar quedaría de la siguiente manera.



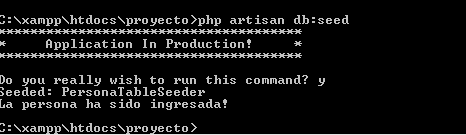
El siguiente paso será definir nuestro modelo el cual en una representación de nuestra clase (los modelos serán explicados detalladamente más adelante), para ello nos dirigiremos al siguiente directorio **app/models/** y crearemos el siguiente archivo **Persona.php** con el siguiente codigo.



Ahora solo bastara crear el archivo del seeder a utilizar cuyo nombre será el mismo que definimos en la función **call** en este caso **PersonaTableSeeder.php** que se ubicara en el directorio **app/config/database/seeds/**, en este archivo se ingresara el siguiente código.

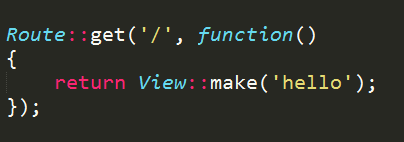


Una vez ingresado el código anterior volveremos a nuestra consola y escribiremos el siguiente comando **php artisan db:seed** .



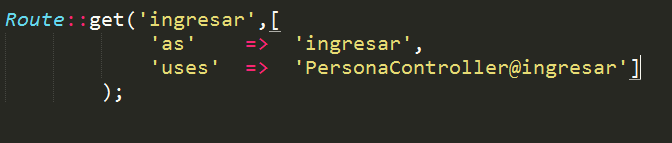
Si todo ocurrió de la manera correcta debería de mostrarse al final el texto que describimos dentro del método run de la clase DatabaseSeeder, ahora puede comprobar en su base de datos y vera que ahora cuenta con una nueva tabla con un registro en ella, los concepto de migrations y seeder serán retomados más adelante.

Ahora que ya contamos con nuestra tabla en la base de datos podremos realizar consultas y mostrar registro en una página web, el primer paso para realizar cualquier tipo de acción en nuestro servidor tiene que existir una ruta que apunte a la acción que se realizara, para hacer esto es necesario definirlas en el archivo **routes.php** ubicado en la carpeta **app/** si lo abrimos solamente encontraremos la ruta **/**  que indica el home o la raíz similar al archivo index habitual que se ejecuta al ingresar a la página principal.

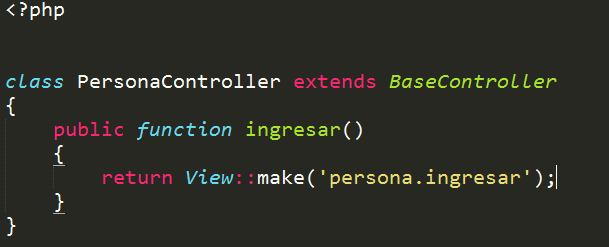


Como se muestra en la imagen la ruta cuenta con una función como parámetro llamada función anónima y dentro de ella se encuentra la lógica de la aplicación en este caso simplemente regresa una vista que la pantalla de bienvenida de la aplicación.

Además de las rutas get existen otro tipo de verbos http como post, put y delete, también la ruta puede recibir en vez de una función un arreglo donde en la llave uses se define el nombre del controlador a usar y la función indicada separado con el símbolo @, también lleva una llave con el nombre as que servirá como el nombre a utilizar al direccionar alguna ruta hacia ella, a continuación se muestra un ejemplo.

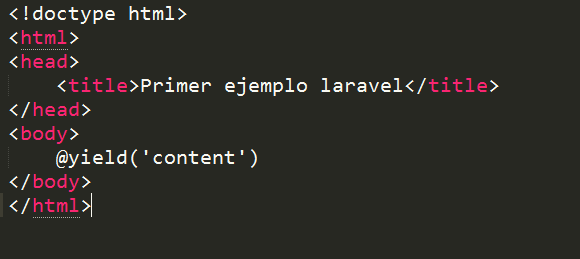


Una vez agregada la ruta será necesario crear el archivo y el controlador a utilizar para ello nos situamos en el directorio **app/controllers/** y crearemos en archivo **PersonaController.php** tal como se ingresó en la ruta y dentro del escribiremos el siguiente código, como ven se creó la función ingresar dentro de la clase PersonaController y dentro de ella retorna una vista a utilizar, se debe tomar él cuenta que dentro la función **View::make()**  recibe el nombre de la vista a utilizar y el punto significa un directorio a diferencia de otros lenguajes que se utilizaría el símbolo /.

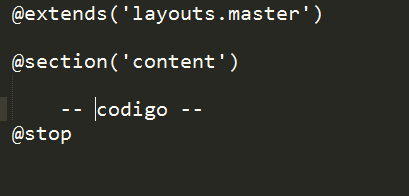


Ahora solamente será necesario crear la vista ingresar dentro de un directorio llamado persona, Laravel utiliza un motor de plantillas llamado blade que permite nuestras vistas tengan un mejor orden, sean más expresivas y se reutilice el código. Para utilizar el sistema de vistas blade los archivos deben tener una extensión de la siguiente forma **ejemplo.blade.php** cabe mencionar que no es estrictamente necesario utilizar el sistemas blade pero en este caso lo utilizaremos.

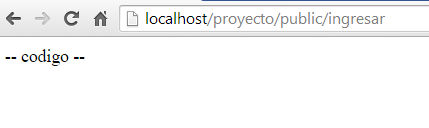
Para comenzar nos situaremos en el directorio **app/views/** y dentro de él crearemos dos carpetas una llamada **layouts** y la otra **persona**, dentro de la capeta layouts crearemos un archivo sencillo con una estructura html sencilla, crearemos un archivo llamado **master.blade.php**  con el siguiete código.

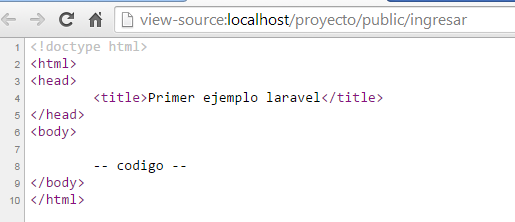


EL archivo **master.blade.php** puede ser utilizado como la base para las vistas a usar por ejemplo en las páginas web que todas cuentan con el mismo footer, cabecera, logos, etc. La sentencia **@yield(‘content’)** tiene la función de ser el espacio de la página cambiante el cual será sobre escrito, para realizar esto dentro de la carpeta **persona**  crearemos el archivo **ingresar.blade.php** con el siguiente código, la sentencia **@extends(‘’)** indica que esta página hereda de la página master y **@section(‘content’)** indica el código que será sobre escrito en **@yield(‘content’)** del archivo padre.Como se muestra a continuación.

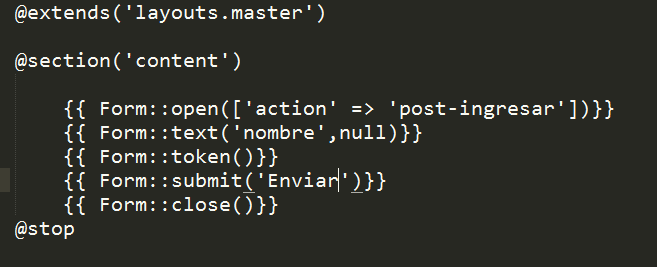


Con los pasos realizados hasta ahora nuestra ruta debería de estar funcionando, sin nos dirigimos a la ruta **localhost/projecto/public/ingresar** nos debería de mostrar – código – en la página como se muestra en la siguiente imagen y si inspeccionamos el código fuente de la página verán que se agregó en el sitio adecuado.

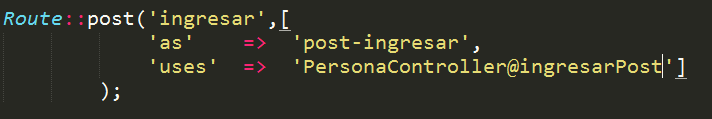




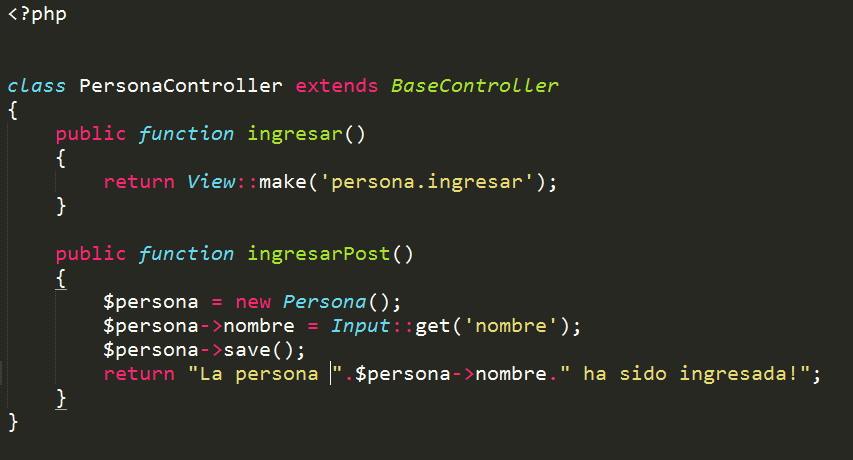
Pero en nuestra ruta ingresar como su nombre lo indica debemos de poder ingresar una nueva persona para hacer esto ingresaremos el siguiente código en la sección content.



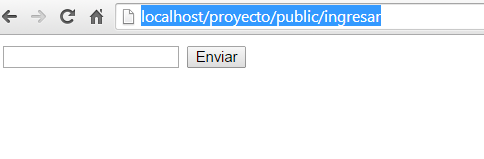
Ahora ya contamos con un formulario con el campo nombre que nos permitirá ingresar una nueva persona, pero para ingresar una nueva persona debemos definir una ruta hacia donde se enviar el nombre de la nueva persona y este tipo de ruta debe ser de tipo post, aunque no definimos el método post Laravel lo utiliza por defecto, para hacer esto volvemos a nuestro archivo **routes.php** y crearemos la siguiente ruta.

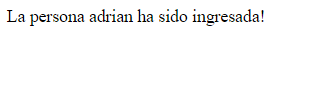


Como se puede notar esta ruta será de tipo post y utilizar la función **ingresarPost()**  que se encuentra en el controlador **PersonaController** y con se imaginan necesitamos crear esa función. El código en el archivo PersonaController.php quedara de la siguiente manera.

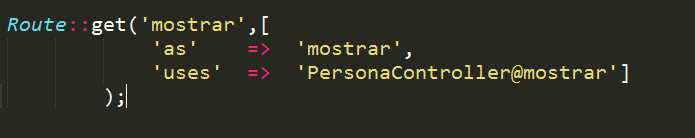


Ahora con el código que tenemos has este momento ya debería de ser posible ingresar a una nueva persona, como ven no se necesitó realizar ninguna sentencia SQL gracias al objeto Persona que es el modelo **Persona.php** definido en la carpeta **app/models/** anteriormente, una vez que se registre la nueva persona nos aparecerá un mensaje indicando. Lo puede probar volviendo a ingresar a la ruta **localhost/proyecto/public/ingresar** lo cual ahora les mostrara un formulario de envió y un mensaje indicando que se realizó de manera exitosa, también pueden revisar en su base de datos la cual tendrá un nuevo registro.

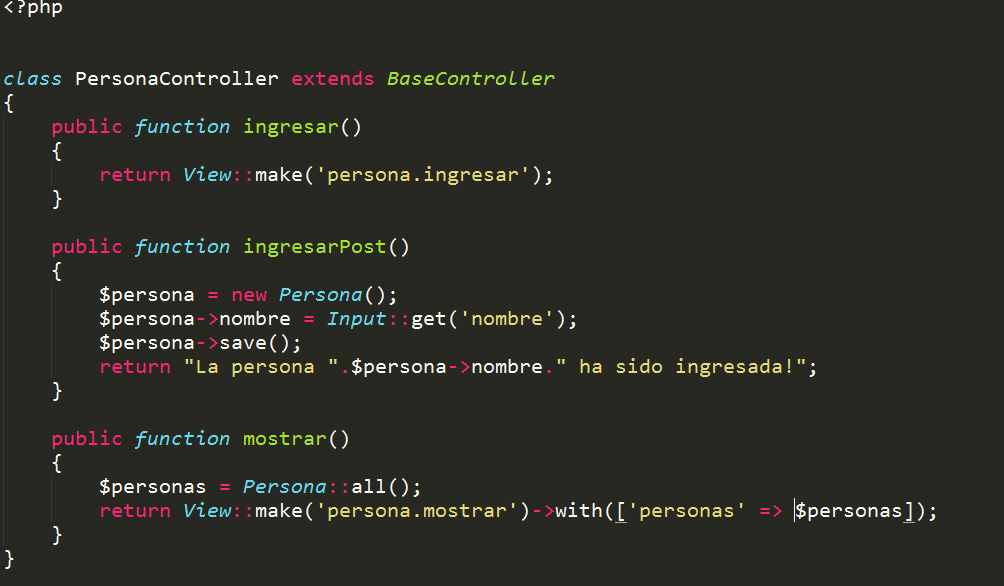




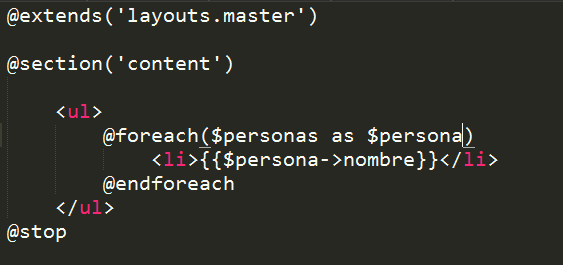
Nuestro programa ya puede ingresar registro, ahora si queremos mostrar las personas ingresadas, como hicimos anteriormente crearemos una ruta, que realizara una acción en un controlador la cual será obtener los registros de un modelo y mostrarlos en una vista, para hacer esto el primer paso es crear nuestra ruta, como la siguiente imagen.



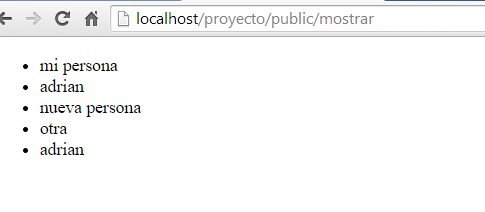
Ahora tenemos que definir nuestra función en el controlador PersonaController el cual quedara de la siguiente manera.



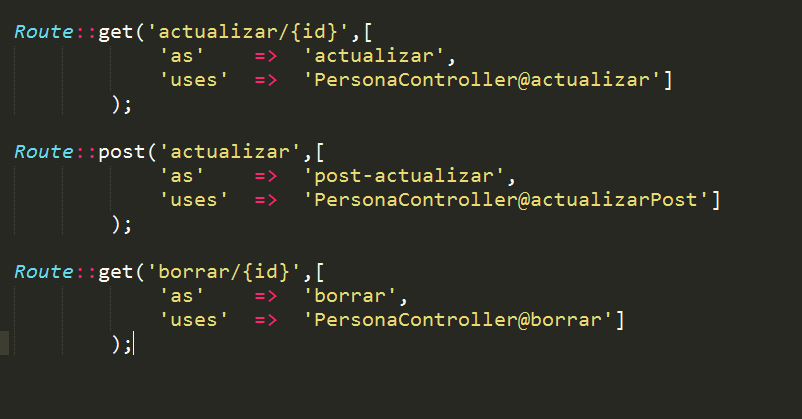
De nuevo gracias al modelo Persona solamente utilizamos el método **all()** el cual regresara todos los registros y los pasamos a la vista mediante el código **View::make()** como vimos anteriormente más el método **with()** que adjunta la variable personas. Ahora debemos de crear el archivo **mostrar.blade.php** dentro de la ruta **app/views/persona/** con el siguiente código.



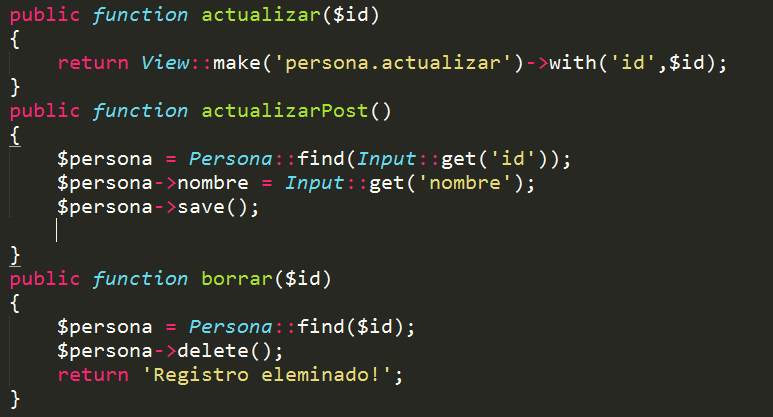
Una vez agregado el siguiente código se vamos al navegador e ingresamos la dirección web **localhost/proyecto/public/mostrar** aparecerá una lista similar a la siguiente con los nombres de las personas en la base de datos.



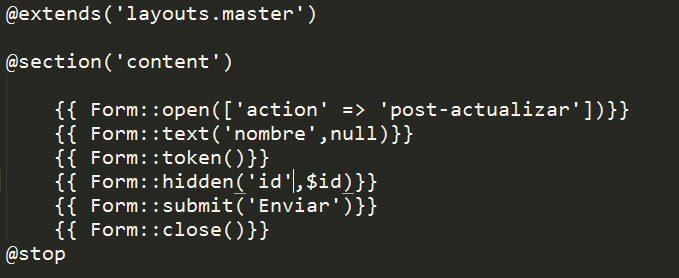
Para completar las operaciones básicas ahora se crearan las rutas para actualizar y borrar los registros, recuerde que deben de ir en el archivo **routes.php**, las rutas quedaran de la siguiente forma:



Las rutas ingresada son similares a las ingresadas anteriormente solo con una diferencia, ahora tienen el siguiente formato en el nombre de la ruta **actualizar/{id}** esto indica que esta ruta necesitara un parámetro en este caso el id, en la definición de la función a utilizar no es necesario agregarlo pero si es obligatorio en la función del controlador o si se utiliza una función anónima en lugar de utilizar un arreglo. El código en el controlador quedara de la siguiente forma.

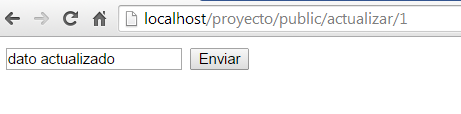


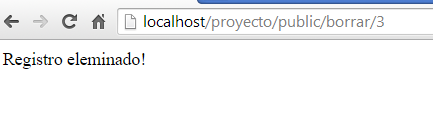
El nuevo código ingresado es similar a los anteriores solo tome en cuenta que se debe de agregar el parámetro o los paramentos necesarios a las funciones que apuntan las rutas, también se utiliza una nueva función llamada **find()** que como su nombre lo dice regresa un elemento con el id que recibe como parámetro, una vez asignado el registro a una variable ya es posible realizar las operación de actualizar y eliminar el registro, antes de probar las nuevas funcionalidades es necesario definir la vista actualizar que se encontrara dentro de la carpeta persona, con el siguiente código.



El formulario es muy similar al de ingresar, solamente con una diferencia que es un campo oculto que contendrá el id del registro a eliminar.

Para probar que los el código está funcionando será necesario ingresar a la dirección web **localhost/proyecto/public/actualizar/un \_numero\_de\_id**, de la misma forma la ruta para borrar, una vez hecho los cambios en la base de datos puede comprobarlo ingresando a la ruta mostrar creada anteriormente.

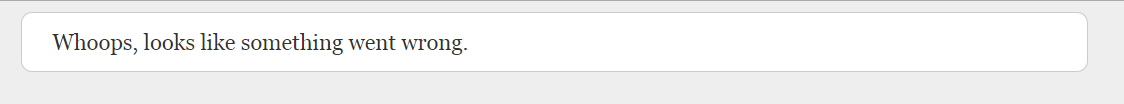




Con esto concluye la aplicación proyecto con las operaciones básicas de búsqueda, ingreso, borrado y actualizado de datos, utilizan las funciones principales de Laravel como son el sistema de vista, modelos, controles, migraciones, etc. Temas que serán reforzados en capítulos posteriores.

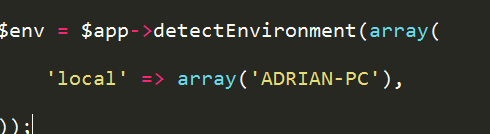
PRIMEROS PASOS AL DESARROLLAR UN PROYECTO

Antes de comenzar a desarrollar en Laravel es recomendable cambiar la configuración de modo debug o depuración a false en el archivo **app.php** ubicadon en **app/config/** lo que nos permitirá ya que somos humanos ver algún error de manera detallada que pueda haber ocurrido mediante se desarrolla la aplicación ya que si no se realiza solamente aparecerá un mensaje como el siguiente:

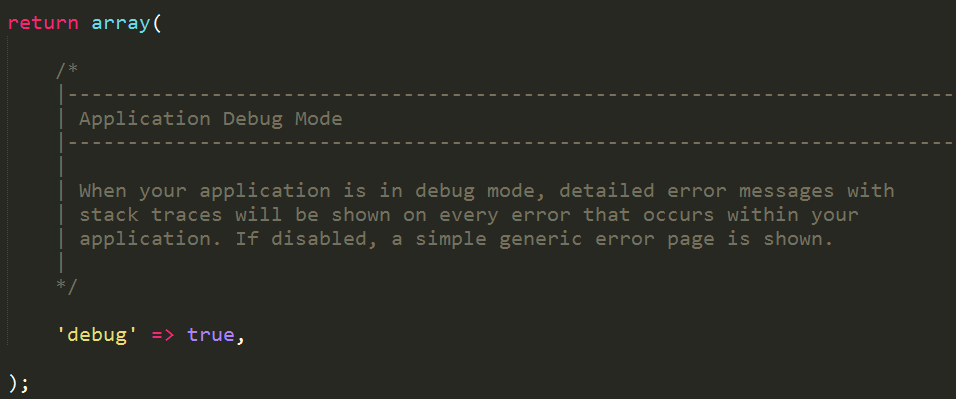


Pero existe una mejor forma de realizar esto que es devolviendo nuevamente la variable debug a false y definir nuestra maquina como servidor local lo cual nos permitirá sobrescribir esa variable a true, de esta forma en nuestra computadora se mostrara un resumen del error ocurrido mientras que en otra máquina o el servidor de producción solamente se mostrara la imagen anterior, entre las ventajas que cuenta realizar esta configuración son en el uso de bases de datos ya que puedes trabajar en tu computadora con un motor de bases de datos y en producción otro al igual que el usuario y contraseña ya que usualmente en el servidor de producción utiliza datos más complejos lo que nos ayuda a no tener que modificar estos datos cada vez que se realice un cambio en el código.

Para definir nuestro o nuestros equipos como un entorno local abrimos el archivo **start.php** ubicando en **/bootstrap/** y dentro de él solamente es necesario agregar el nombre de nuestra maquina similar a lo siguiente.



Con esto debería de ser suficiente ya la opción de depuración en modo local está habilitada por defecto, solamente para comprobarlo es necesario ingresar a **app/config/local/app.php** el cual tendrá el siguiente código.

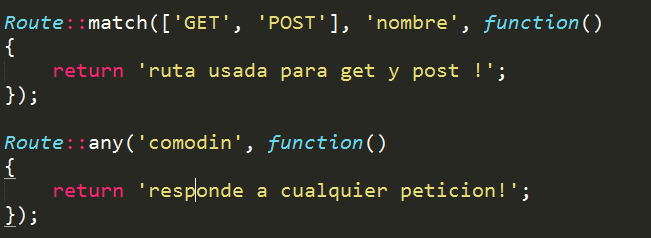


RUTAS

Como se mencionó anteriormente existen varios tipos de rutas como son get, post, put, delete y estas pueden llevar o no parámetros y pueden utilizar una función anónima o un controlador, pero además de estas existen varios tipos más cono los grupos de rutas, las rutas con filtros, las rutas con comprobación de parámetros o parámetros por defecto, las rutas a controladores conocidas como RESTFull, rutas a resources entre otras, las cuales serán descritas en este capítulo.

RUTAS QUE UTILIZAN VARIOS VERBOS HTTP

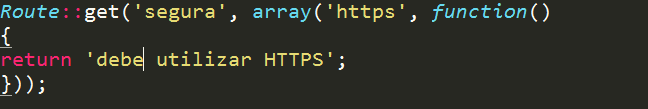
Estas rutas como su nombre lo dice pueden responder a varios tipos de peticiones, pueden ser utilizados mediante la función **match** en la cual se definen los verbos que se va a utilizar o la función **any** que sería como un comodín que puede funcionar con cualquier verbo, a continuación se muestra un ejemplo de cada una.



Como se muestra en el código la ruta match necesita tres parámetros en el primero un arreglo con los verbos disponibles, en el segundo el nombre de la ruta y en el tercero un arreglo o función anónima, mientras que la función any solamente se declara en vez del verbo http habitual.

RUTAS PROTEGIDAS CON HTTPS

Las rutas protegidas con https necesitan un arreglo en el cual se define que debe utilizar https como se muestra en la imagen.



RUTAS CON PARÁMETROS

Esta rutas reciben uno o varios parámetros para funcionar, los parámetros deben ir encerrados entre corchetes, también los parámetros pueden ser definidos por defecto esto se hace agregando el símbolo de pregunta ? antes del corchete de cierre, además de esto las rutas con parámetros pueden llevar condiciones o reglas que deben cumplir como que serán números o cadenas de caracteres. En la imagen de enseguida se muestra un ejemplo de cada una.

