DOCUMENTACIÓN DE LA PLANTILLA BASE PARA MENÚS

INTRODUCCIÓN

Se trata de un proyecto que sirve como base para crear aplicaciones Laravel. El proyecto consiste en un sistema de menús personalizables, y unas áreas para que el desarrollador ponga sus propios contenidos.

El proyecto está desarrollado sobre la base de Laravel 5.8. La razón de usar esta versión es que sea adaptable a cualquier proyecto en curso. Sin embargo, no hay problemas de incompatibilidades para adaptarla a una versión posterior, si es necesario (6, 7 u 8), con las excepciones que se mencionan en el apartado ACTUALIZACIONES DE LARAVEL, en este mismo documento.

El sistema permite gestionar una aplicación multi-idioma. En principio, está pensado para diez idiomas: Alemán, Catalán, Danés, Español, Euskera, Francés, Inglés, Italiano, Portugués y Rumano. Sin embargo, se pueden añadir otros, si fuera necesario. Además, por configuración, se podrán usar sólo parte de los que se ofrecen con la plantilla básica. Se incluyen las traducciones de la plantilla, aunque el desarrollador que la use, al crear su aplicación, deberá incluir las traducciones de sus propios contenidos.

El proyecto está concebido para cuatro ámbitos de usuario:

- **Visitante.** Solo tiene acceso a la página principal (página de acceso) que, normalmente, será una página corporativa, o con un resumen o presentación de la aplicación. En la página corporativa se ofrecerá un enlace de Acceso (login). Además, estableciéndolo por configuración, se podrá incluir un enlace de registro. Esto está pensado por si los usuarios deben poder registrarse por ellos mismos, o deben ser registrados por personal autorizado para ello.
- **Usuario registrado.** Una vez que un usuario se ha registrado (bien por sí mismo, o por haber sido registrado por un administrador) podrá acceder a esta zona, donde tendrá, a su disposición, las opciones de menú que un administrador, o persona autorizada, le asigne.
- **Administrador (Admin).** Es un nivel de autorización superior al del usuario registrado. Podrá crear y desactivar usuarios, así como suplantarlos, si es necesario. También tendrá acceso a las opciones de menú que se le asignen.
- **Master.** Es el nivel más alto de usuario. Este nivel está reservado para crear, editar o desactivar administradores, así como suplantarlos. Por configuración se puede determinar si también puede gestionar y suplantar a usuarios registrados.

La plantilla está pensada de tal modo que las opciones de los distintos menús se crean en base de datos, para poder asignárselas a los distintos usuarios, en una relación **`m-n`**. La plantilla está concebida, por tanto, para proyectos que admitan registro de usuarios en distintos niveles.

Los usuarios, en caso de que se determine que puedan registrarse a sí mismos, lo harán con el nivel más bajo (usuario registrado). Los usuarios administradores podrán crear y/o editar, mediante los permisos oportunos, a otros administradores o a usuarios registrados. Por

último, serán los usuarios de ámbito máster los que puedan crear y/o editar a otros usuarios de cualquier nivel de alcance.

La plantilla está pensada para poder establecer independientemente las opciones a las que pueda acceder cada usuario. En el momento de autenticarse se ofrecerá un menú lateral con las opciones que el usuario que se autentique tenga disponibles.

La plantilla también tiene posibilidad de cambiar su aspecto mediante temas CSS que se pueden establecer para cada para cada proyecto.

Se han previsto medidas de seguridad para evitar el acceso de un visitante o usuario a áreas o funcionalidades que no tenga asignadas.

En este documento se detallan todas las prestaciones de la plantilla, así como su uso con algunos datos de ejemplo que se incorporan en la misma, y que serán sustituidos por los pertinentes en cada proyecto.

ACTUALIZACIÓN A LA VERSIÓN 1.1

En esta versión se ha mejorado sensiblemente el proyecto original, desde el punto de vista técnico, para facilitarle al desarrollador que use la plantilla la construcción de su propio proyecto. En concreto, entre las novedades más relevantes, están las siguientes:

- Reparación de los bugs detectados en la versión inicial.
- Desacoplamiento de las rutas en distintos archivos. El objetivo de esto es doble: por una parte, una mayor organización a la hora de gestionar las rutas; por otro lado, se prepara la aplicación para nuevas funcionalidades que se desarrollarán en futuras versiones.
- Mayor desacoplamiento de los archivos de traducciones de la plantilla, a efectos organizativos.
- Implementación de pantallas de error personalizables, tanto para los errores típicos de expiración de sesión o página no encontrada, como para errores en el envío de SMS's, u otros errores genéricos.
- Implementación de funcionalidades básicas, como edición de perfil, listado de usuarios, impersonación, etc.
- A nivel de codificación, a los elementos que son propios de la plantilla (traducciones, rutas, controladores, etc.) se les ha cambiado el nombre, precediéndolo con el prefijo **BM_**, para evitar colisiones de nombres con elementos que incorpore cada desarrollador a la hora de crear sus propios proyectos basados en esta plantilla.
- Se han añadido esquemas de colores para el layout (temas), para dar mayor flexibilidad al diseño de proyectos específicos.
- Se ha añadido la posibilidad de usar autenticación de doble factor, lo que obligará a los usuarios a una doble autenticación, que se podrá hacer mediante correo electrónico o SMS enviado al móvil. En caso de que el proyecto a desarrollar no requiera este nivel de seguridad, esta opción es desactivable.

REQUISITOS TÉCNICOS

Los requisitos técnicos para usar esta plantilla en un proyecto son:

- Laravel versión 5.8 o posterior. Se ha realizado en esta versión para que se pueda usar, prácticamente, en cualquier proyecto. Nada impide usarlo en un proyecto con una versión superior (6, 7, 8, o las siguientes que surjan), pero no está limitado, salvo lo especificado en la sección ACTUALIZACIONES DE LARAVEL, en este mismo documento.
- PHP versión 7.3 o posterior.
- Servidor web Apache o Nginx.
- Motor de base de datos MySQL.
- Y, por supuesto, navegador web, editor de código y el gestor de dependencias Composer.

Además, y tal como nos indica la documentación oficial de Laravel, debemos asegurarnos de que PHP tenga instaladas las siguientes extensiones:

- BCMath
- Ctype
- JSON
- MBString
- OpenSSL
- PDO
- Tokenizer
- XML

Podemos utilizar la función `phpinfo()`, nativa del lenguaje, para ver cuáles de estas extensiones están disponibles en nuestra instalación.

OPCIONES DE CONFIGURACIÓN

En este apartado se recogen las opciones de configuración de la plantilla, que deben establecerse en el archivo `.env`, así como en el directorio `/config`, y que determinarán distintos aspectos de comportamiento de la misma. De esta forma se logra que la plantilla sea muy personalizable, para usarla en distintos proyectos. En esta sección vamos a ver las distintas opciones de personalización de la plantilla.

EL REGISTRO DE USUARIOS

Esta es la parte que más fácil resulta de configurar, y más práctica puede resultar a la hora de usar la plantilla. Las opciones se gestionan en el archivo `.env`, es decir, como variables superglobales de entorno, de modo que los parámetros que aquí establezcamos estarán disponibles para todos los usuarios.

Las opciones son las siguientes:

- `SPC_REGISTER`. Se emplea para determinar si los usuarios pueden registrarse a sí mismos. Si tiene el valor `true`, en la vista de inicio habrá un enlace de registro. Si se le asigna el valor `false`, los usuarios deberán solicitar el registro, y será un usuario autorizado quien deba registrarlos. En la plantilla se ha incluido el mecanismo de registro para cuando los usuarios pueden registrarse a sí mismos. En caso de que deban (o puedan) ser registrados por un usuario ya existente, el mecanismo, al ser personalizable, deberá añadirse como parte de cada proyecto específico.
- `MAIL_MUST_BE_VERIFIED`. Determina si los nuevos usuarios deben verificar su correo electrónico. Si se establece en `true`, cuando un usuario se registre recibirá en su bandeja de email un correo con un enlace de verificación. No podrá hacer uso de la aplicación hasta no haber verificado su correo electrónico. En caso de que esta variable tenga el valor `false`, una vez registrado un usuario podrá acceder inmediatamente a su espacio personal en la aplicación.
- `PHONE_MUST_BE_VERIFIED`. Determina si se debe verificar el número de móvil, mediante un SMS, para completar el registro. Esta opción está sujeta a que estén activadas las opciones `PHONE_IN_REGISTER` y `PHONE_MANDATORY`.
- `PHONE_IN_REGISTER`. Se determina si al registrarse un usuario puede indicar el código telefónico de su país y el número de su móvil. Los posibles valores son `true` o `false`. Esto se prevé por si se va a dotar al proyecto de la funcionalidad de envío de SMS's mediante el servicio **Vonage** (antes **Nexmo**), o algún otro servicio similar.
- `PHONE_MANDATORY`. Si en la variable anterior hemos establecido el valor `true`, podemos hacer, mediante esta variable, que el código de país y número de móvil sean obligatorios en el registro (valor `true`), o no (valor `false`). Si en la variable anterior hemos establecido el valor false esta variable no tendrá efecto alguno.
- `PASSWORDS_SHOWING`. Mediante esta variable damos la opción de que el usuario vea, de forma legible, sus contraseñas al teclearlas, tanto para registro como para

autenticación. Si el valor es **N**, los campos de contraseña nunca mostrarán lo que se teclee en ellos. Si establecemos el valor **B**, estos campos tendrán un icono que se podrá pulsar para mostrar legible el campo de contraseña. Esto se emplea para que el usuario se asegure de lo que está tecleando. Por último, el valor **S** mostrará siempre los campos de contraseña como texto en claro. Este último valor no es recomendable, salvo en las fases de desarrollo y pruebas.

- **2FA_CODE**. Sirve para establecer si se emplea doble factor de autenticación. La opción **none** indica que no se emplea; la opción **mail** indica que se emplea 2FA por correo electrónico. Por último, la opción **phone** se refiere al uso de 2FA por SMS.

Además, para el caso de que el desarrollo que hagamos sobre esta plantilla emplee mensajes SMS, se han reservado las claves **NEXMO_KEY** y **NEXMO_SECRET**, que deberán recibir los valores de la cuenta de Nexmo del desarrollo.

I18N (LAS TRADUCCIONES)

En esta sección vamos a describir cómo funciona el sistema de traducciones, así como los archivos que será necesario conocer para agregar o suprimir idiomas. También conoceremos los puntos dónde es necesario incluir los archivos de traducciones de cada proyecto que desarrollemos a partir de esta plantilla.

MIDDLEWARE

Para la gestión de idiomas, en el caso de que tu proyecto los emplee, se ha incluido un middleware, llamado **`Languages.php`**, cuya finalidad es recuperar el valor de sesión del idioma seleccionado, o utilizar el español (idioma por defecto) si no se ha seleccionado otro. Este middleware no necesita ser modificado para los proyectos que se desarrollen a partir de la plantilla. Está registrado en **`app/Http/Kernel.php`**, dentro de la matriz **`$middlewareGroups`**, con la siguiente línea:

```
\App\Http\Middleware\Languages::class,
```

CONFIGURACIÓN

Dentro del directorio **`/config`** tenemos varios puntos que debemos considerar a la hora de establecer el idioma o los idiomas que estarán disponibles para un proyecto específico. En primer lugar, dentro del archivo **`app.php`** estableceremos los valores de **`locale`** y **`fallback_locale`** en **`es`**, ya que el español será el idioma por defecto, que siempre estará disponible para cualquier proyecto basado en esta plantilla. Si deseamos que el idioma por defecto sea el inglés (algo muy habitual en aplicaciones web), estableceremos el valor de las variables mencionadas en **`en`**. Lo mismo se puede aplicar si tu proyecto va destinado, de forma mayoritaria, a un mercado local en el que el idioma principal sea otro.

También dentro del directorio **`/config`** encontramos el archivo **`available_languages.php`**, que contiene una matriz con los idiomas disponibles. El idioma elegido por defecto siempre deberá estar en esta matriz (en la plantilla es el español, pero puede ser el que decidamos), por lo que aparece en primer lugar. En el resto de la matriz podremos retirar los idiomas que no vayamos a emplear, o añadir los que necesitemos. Cada idioma en la matriz está representado por una submatriz, con la siguiente estructura:

```
'código' => [
    'code' => 'código',
    'variable_name' => 'nombre',
    'image_name' => 'código.jpg',
],
```

Los códigos de idioma, en todo el proyecto, se representan con una secuencia de dos letras, según el estándar **ISO 639-1**, que es el empleado por Laravel. Por ejemplo, la submatriz que se refiere al español tiene el siguiente formato:

```
'es' => [
    'code' => 'es',
    'variable_name' => 'Español',
    'image_name' => 'es.jpg',
],
```

Lo mismo se aplica al resto de idiomas.

Como hemos mencionado, si alguno de los idiomas de la plantilla no va a figurar en el proyecto, eliminaremos la correspondiente submatriz. Si vamos a añadir algún idioma nuevo, deberemos incluir aquí la submatriz con los datos de dicho idioma.

Dentro del directorio **config/** también encontramos el archivo **paths_to_files.php**. Este contiene las rutas a las imágenes de las banderas relativas a los distintos idiomas que pueda usar la aplicación dentro de **public/**. Cualesquiera otros archivos que vayamos a incluir jerarquizados en subdirectorios dentro de **public/** deberemos definir su ruta en este archivo de configuración.

LOS ARCHIVOS DE IDIOMAS

Dentro del directorio **resources/lang** encontramos los archivos de las traducciones en los distintos idiomas que soporta la plantilla. La lista de estos se detalló en la introducción de este documento.

Aquí entra la reorganización de los archivos de traducciones que se ha establecido en esta nueva versión. Dentro de la carpeta de cada idioma, las traducciones propias de la plantilla están en un directorio llamado **BM**. Todas las traducciones propias de un desarrollo específico deberán ponerse fuera del directorio **BM**, para evitar colisiones.

Lo primero que vemos son los archivos **.json**, que contienen los textos de los correos electrónicos de verificación de email, y recuperación de contraseña perdida. A continuación se reproduce, a modo de ejemplo, el archivo en español:

```
{
      "Regards" : "Saludos",
      "Hello!" : "¡Hola!",

      "Verify Your Email Address" : "Verifique su dirección de e-mail",
      "Please click the button below to verify your email address." : "Pulse el botón que aparece a continuación para verificar su correo.",
      "Verify Email Address" : "Verificar correo electrónico",
```

```
    "If you did not create an account, no further action is
required." : "Si usted no ha creado una cuenta, puede borrar
este correo sin más.",

    "Reset Password Notification" : "Restablecimiento de
contraseña",
    "You are receiving this email because we received a
password reset request for your account." : "Ha recibido este
correo porque ha solicitado restablececer la contraseña de su
cuenta.",
    "Reset Password" : "Restablecer contraseña",
    "This password reset link will expire in :count minutes." :
"Este enlace expirará en :count minutos.",
    "If you did not request a password reset, no further action
is required." : "Si no solicitó el restablecimiento de su
contraseña, simplemente ignore este correo.",

    "If you're having trouble clicking the \":actionText\"
button, copy and paste the URL below\ninto your web browser:
[:actionURL](:actionURL)" : "Si tiene problemas usando el botón
\":actionText\", copie y pegue la siguiente dirección en su
navegador: [:actionURL](:actionURL)"
}
```

Los archivos de los demás idiomas siguen la misma estructura. Si alguno de los idiomas previstos no va a utilizarse se puede eliminar el correspondiente archivo (aunque no es necesario hacerlo, ya que el proyecto, simplemente, lo ignorará). Si se va a utilizar algún idioma no previsto, habrá que añadir el archivo con los correspondientes valores. El nombre de los archivos corresponde al código ISO 639-1 (con dos letras) de cada idioma.

Además, encontrarás los archivos con las traducciones de las vistas que incluye la plantilla, en directorios nombrados con los correspondientes códigos de cada idioma. Si se van a emplear otros idiomas no previstos, será necesario añadir los directorios correspondientes. Dentro de cada directorio encontramos, a su vez, los siguientes directorios y archivos:

- **Directorio `BM/admin/`**: Contiene las traducciones de todas las vistas de los usuarios de nivel admin. En la plantilla sólo aparece el archivo **`index.php`**, con el único texto que hay en esta plantilla. Cuando desarrollemos nuestra aplicación, habrá que cambiar este texto y añadir los que sean necesarios. Además, habrá que añadir, en este directorio, los archivos de traducciones de las distintas vistas que preparemos para el usuario de nivel admin.
- **Directorio `BM/auth/`**: Contiene las traducciones de los textos que aparecen, por defecto, en las vistas de autenticación y de registro. Si vamos a modificar estas vistas, deberemos añadir o modificar las traducciones de los dos archivos (**`auth.php`** y **`register.php`**) que se encuentran en este directorio.
- **Directorio `BM/general/`**: Aquí se encuentran los archivos **`accesos_principales.php`**, **`idiomas.php`** y **`menu_lateral.php`**. En el

primero se encuentran los rótulos de los links de la barra de navegación superior, con excepción del selector de idioma. Será necesario modificarlo si se van a añadir más opciones a la barra superior. En el segundo se encuentran los textos de los enlaces del selector de idioma de la barra superior. Será necesario modificarlo si se van a añadir o suprimir idiomas en la barra superior. Por último, en el tercer archivo se encuentran los rótulos de opciones y grupos de opciones del menú lateral. Este menú sólo está disponible para usuarios registrados, no para visitantes. Será necesario modificarlo con las opciones y grupos que vaya a tener nuestro proyecto. Por defecto, se incluyen algunos rótulos que se emplean a modo de ejemplo, pero sólo son para la plantilla "en bruto", es decir, antes de su uso para un proyecto.

- **Directorio `BM/master/`**: Contiene las traducciones de todas las vistas de los usuarios de nivel master. En la plantilla sólo aparece el archivo **`index.php`**, con el único texto que hay en esta plantilla. Cuando desarrollemos nuestra aplicación, habrá que cambiar este texto y añadir los que sean necesarios. Además, habrá que añadir, en este directorio, los archivos de traducciones de las distintas vistas que preparemos para el usuario de nivel master.
- **Directorio `BM/user/`**: Contiene las traducciones de todas las vistas de los usuarios de nivel user. En la plantilla sólo aparece el archivo **`index.php`**, con el único texto que hay en esta plantilla. Cuando desarrollemos nuestra aplicación, habrá que cambiar este texto y añadir los que sean necesarios. Además, habrá que añadir, en este directorio, los archivos de traducciones de las distintas vistas que preparemos para el usuario de nivel user.
- **Archivo `auth.php`**: Contiene las traducciones de los errores de validación para la autenticación de un usuario.
- **Archivo `pagination.php`**: Contiene los textos de los enlaces **Previo** y **Siguiente** para cuando se emplea paginación de resultados en una vista.
- **Archivo `passwords.php`**: Contiene las traducciones de los errores de validación para el restablecimiento de contraseña.
- **Archivo `validation.php`**: Contiene las traducciones de todos los posibles errores de validación de un proyecto Laravel.

Dentro del directorio **`public/images/flags/`** se encuentran diferentes versiones, en tamaño y forma, de las banderas representativas de los idiomas que se pueden usar en la plantilla. Si vamos a crear un proyecto con idiomas que no están disponibles por defecto, deberemos incorporar imágenes de las correspondientes banderas.

LAS VISTAS

En las vistas se emplea el helper **`__()`** de Laravel, para referenciar los distintos textos que deben ser traducidos según los archivos de idiomas.

OTRAS CONSIDERACIONES DE ARCHIVOS

En este capítulo vamos a ver las consideraciones generales que se aplican a distintos archivos de la plantilla, de forma específica para la misma.

MODELOS

Los modelos se encuentran en el directorio **app/Models/**. Esta era una forma de operar en versiones anteriores de Laravel, y la versión 8 del framework la ha recuperado. Aunque, por defecto, en la versión 5.8 los modelos están, directamente, en la carpeta **app/**, en esta plantilla hemos creado la carpeta **Models/**, para facilitar la organización de la estructura de archivos. Por lo tanto, en los modelos hay que redefinir los namespaces, para que sean reconocidos correctamente por la aplicación.

MIDDLEWARES

En la aplicación se emplean varios middlewares. Estos están referenciados en el archivo **config/register.php**. Además de los middlewares inherentes a cualquier aplicación web, contamos con **app/Http/Middleware/Scope.php**. Este se encuentra referenciado en **app/Http/Kernel.php** con la siguiente línea:

```
'scope' => \App\Http\Middleware\Scope::class,
```

El objetivo del mismo es evitar que un usuario pueda acceder a las partes de la aplicación que corresponden a otro usuario con un mayor rango.

La plantilla también incluye el middleware **app/Http/Middleware/CheckOption.php**. Su finalidad es que un usuario no pueda acceder a una parte de la aplicación que no tenga asignada (ver el apartado BASE DE DATOS), ni siquiera tecleando la URL directamente en el navegador. Este middleware se incluirá en las rutas a las que se accede mediante un enlace autorizado por asignaciones (normalmente, rutas de tipo GET). No se incluirá en rutas de tipo POST (las que, por ejemplo, reciben formularios), ya que comprueba una ruta por las asignaciones de opciones que existen en base de datos.

RUTAS

Las rutas determinadas en **routes/web.php** se pueden considerar en las siguientes cinco secciones, según se detalla.

- **Rutas generales.** Incluye tres rutas: la de la página principal de la aplicación (a la que se accede como visitante no registrado), la destinada a cambiar el idioma de la

aplicación, y la destinada a cambiar el tema CSS del layout. Estas rutas están al principio del archivo, y son accesibles directamente mediante los enlaces adecuados.

- **Rutas de autenticación.** Están dentro de un grupo, y son las rutas de autenticación habituales de cualquier aplicación Laravel.
- **Rutas del Master.** Son las rutas que se emplean para acceder a las secciones disponibles para los usuarios de rango Master.
- **Rutas del Admin.** Son las rutas que se emplean para acceder a las secciones disponibles para los usuarios de rango Admin.
- **Rutas del User.** Son las rutas que se emplean para acceder a las secciones disponibles para los usuarios de rango User.

En las tres últimas secciones se deben definir las rutas adecuadas para las opciones de menú lateral de los distintos rangos de usuarios registrados, como se ve en la plantilla, a modo de ejemplo, para los usuarios de rango Master. Las opciones se definen en la base de datos.

Las rutas del Master, del Admin y del User se dividen, a su vez, en tres achivos: uno para las rutas permanentes, es decir, aquellas que están para todos los usuarios, como pueden ser las que llevan a la página principal de la aplicación o del usuario; otro para las rutas linkadas, es decir, aquellas a las que se accede directamente desde un enlace de menú, y que corresponden a opciones que se deben grabar en la base de datos; por último, están las rutas para acceder a controladores no linkados, como son las respuestas a formularios, peticiones Ajax, etc.

SERVICIOS

La plantilla cuenta con un servicio destinado a ordenar jerárquicamente los grupos y opciones disponibles para el menú lateral de cada usuario. Este servicio define un método específico para las colecciones de datos en las que determinados elementos dependen de otro. El servicio está definido en **app/Providers/appServiceProvider.php**, dentro del método **register()**, y su código es el siguiente:

```
Collection::macro('hierarchy', function() {
    $grupos = $this->filter(function($cat) {
        if ($cat->type == 'G') {
            return $cat;
        }
    });

    foreach ($grupos as $grupo) {
        $grupo->opciones = $this->filter(function ($cat) use
($grupo) {
            return ($cat->parent_id == $grupo->id && $cat->type
== "O");
        });
    }

    $grupos = $grupos->reject(function($grupo) {
```

```
            return $grupo->opciones->count() == 0;
        });
        return $grupos;
});
```

Esta plantilla también tiene otro servicio, en **app/Services/ReadPermissions.php**. En este script se declara la clase **ReadPermissions**, cuyo objetivo es obtener la lista de grupos y opciones del menú lateral del usuario en curso. El objetivo de esta clase es, por tanto, leer los permisos y, mediante el método **hierarchy()**, que hemos declarado en el proveedor de servicios de Laravel, ordenarlos adecuadamente para construir dinámicamente el menú de la barra lateral, como se ve en la vista que hay en el fichero **resources/views/commons/side_menu_bar.blade.php**.

El código del servicio es el siguiente:

```
<?php

namespace App\Services;

class ReadPermissions {
    public function permissions()
    {
        $permissions = null;
        if (auth()->user()) {
            $permissions = auth()->user()->permissions;
            $permissions = $permissions->hierarchy();
        }

        return $permissions;
    }
}
```

Este código se inyecta en **resources/views/layouts/app.php**, mediante la línea que se muestra a continuación:

```
@inject('side_menu', 'App\Services\ReadPermissions')
```

COMPONENTES

Los componentes de Blade son pequeños módulos que se pueden incluir en una vista. De esta forma se puede modularizar aún más el código de las vistas. En la plantilla tenemos un componente en resources/views/components/password.blade.php. La finalidad de este componente es mostrar los campos de contraseña del modo adecuado, según el valor de la variable **PASSWORDS_SHOWING**, del archivo **.env**. Su código es el siguiente:

```
@if (env('PASSWORDS_SHOWING') == 'N')
    <input type="password" id="{{ $id }}" name="{{ $name }}"
class="form-control {{ $class }}">
```

```
@elseif (env('PASSWORDS_SHOWING') == 'B')
    <div class="input-group">
        <input type="password" id="{{ $id }}" name="{{ $name }}"
class="form-control {{ $class }}">
        <div class="input-group-append">
            <span class="link-pointer pw-icon input-group-text"
data-id="{{ $id }}">
                <i class="fa fa-eye"></i>
            </span>
        </div>
    </div>
@else
    <input type="text" id="{{ $id }}" name="{{ $name }}"
class="form-control {{ $class }}">
@endif
```

Este componente, por lo tanto, se inyecta en las vistas de autenticación y registro (y en cualesquiera otras que le añadamos al proyecto que necesiten tener este formato para las contraseñas), del siguiente modo:

```
@component('components.password')
    @slot('id') password @endslot
    @slot('name') password @endslot
    @slot('class') @error('password') is-invalid @enderror
@endslot
@endcomponent
```

TEMAS CSS

Los temas del layout CSS que están disponibles en la plantilla se encuentran en **public/css/themes**, con los nombres **blacksky.css**, **standard.css**, **cherry.css** y **silver.css**. Cualesquiera otros que diseñemos para un proyecto específico se alojarán también en el directorio indicado, aunque la versión 1.1 de la plantilla sólo incluye, por defecto, los cuatro mencionados.

LA BASE DE DATOS

Como en cualquier aplicativo web, la base de datos es uno de los pilares fundamentales. La estructura básica de la plantilla, con independencia de las tablas que se añadan para cada proyecto en particular, incluye tres tablas fundamentales:

- **Tabla `users`**. Contiene los datos típicos de esta tabla en aplicaciones web. Además, incluye los siguientes datos:
  - **`scope`**. El rango del usuario, que podrá ser (**`M`**)aster, (**`A`**)dmin o (**`U`**)ser.
  - **`first_name`**. El nombre de pila del usuario.
  - **`surname`**. El (los) apellido(s) del usuario.
  - **`avatar`**. El nombre del fichero con el avatar del usuario.
  - **`theme`**. El tema CSS del layout.
  - **`country_code`**. El prefijo telefónico del país del usuario.
  - **`phone_number`**. El número de teléfono móvil del usuario.
  - **`phone_verified_at`**. La fecha y hora en que se verifica el teléfono del usuario.
  - **`menu_desplegado`**. Para indicar el grupo de opciones del menú lateral que se encuentra desplegado (sin uso en la versión 1.0 de la plantilla).
  - **`opcion_seleccionada`**. Para indicar la opción que está seleccionada en el menú lateral del usuario (sin uso en la versión 1.0 de la plantilla).
  - **`menu_bar_deployed`**. Para indicar si el menú lateral está desplegado o no (sin uso en la versión 1.0 de la plantilla).
- **Tabla `permissions`**. Contiene la lista de grupos y opciones que estarán disponibles para asignar a los usuarios. Sus campos son los siguientes:
  - **`id`**. El identificativo del grupo u opción.
  - **`scope`**. El rango de usuario parra el que es válida cada opción. Podrá ser **`M`**, **`A`** o **`U`**.
  - **`icono`**. El nombre de la clase de Font Awesome que se mostrará a la izquierda de cada opción (sin uso en la versión 1.0 de la plantilla).
  - **`rotulo`**. La clave de rótulo textual de cada grupo u opción, para ser traducida según los archivos de idiomas.
  - **`parent_id`**. El código del grupo al que pertenece cada opción. En la versión 1.0 de la plantilla sólo habrá un nivel de opciones dependientes de cada grupo, es decir, no habrá subgrupos.
  - **`created_at`**. Fecha y hora en que se crea el grupo o la opción, si se añade la posibilidad de crearlas desde la aplicación (en la versión 1.0 esta opción no está disponible).
  - **`updated_at`**. Fecha y hora en que se editó por última vez el grupo o la opción, si se añade la posibilidad de editarlas desde la aplicación (en la versión 1.0 esta opción no está disponible).

    - **deleted_at**. Fecha y hora en que se marca como borrado el grupo o la opción, si se añade la posibilidad de marcarlas desde la aplicación (en la versión 1.0 esta opción no está disponible).
- **Tabla permission_user**. Es la tabla pivote que se usa para relacionar los grupos y opciones con cada usuario concreto. Sus campos son los siguientes:
    - **id**. El identificador autoincrementable de la relación.
    - **permission_id**. El identificador de la opción o grupo que vamos a relacionar, coincidiendo con el campo id de la tabla permissions.
    - **user_id**. El identificador de usuario que vamos a relacionar, coincidiendo con el campo id de la tabla users.
    - **created_at**. Fecha y hora en que se crea la relación, si se añade la posibilidad de crearlas desde la aplicación (en la versión 1.0 esta opción no está disponible).
    - **updated_at**. Fecha y hora en que se editó por última vez la relación, si se añade la posibilidad de editarlas desde la aplicación (en la versión 1.0 esta opción no está disponible).

USANDO LA PLANTILLA

En este capítulo vamos a aprender como usar la plantilla para desarrollar, a partir de ella, nuestros propios proyectos. Ahora que ya tenemos una visión general, podemos empezar a trabajar desde aquí.

En primer lugar deberemos descargar la plantilla desde su alojamiento en GitHub. La almacenaremos y descomprimiremos en un directorio dentro de nuestro localhost. Después, en la terminal de mandatos (y, siempre dentro del directorio donde tengamos la plantilla), teclearemos la instrucción siguiente:

```
composer install
```

Con esto logramos tener disponible el núcleo de Laravel que, por razones de optimización, no se descarga con la plantilla.

LA CONFIGURACIÓN

El siguiente paso es configurar el proyecto en el archivo `.env`. Aparte de las opciones que hemos visto en este documento, deberemos configurar dos aspectos esenciales:

- El primero es la base de datos. En la plantilla es SQLite, para que quede como un fichero propio del proyecto. Sin embargo, lo más habitual es usar un motor como MySQL. Para esto hay que crear la base de datos y ejecutar las migraciones. Yo recomiendo usar SQLite durante el desarrollo, y pasar a MySQL para producción.
- También hay que configurar el servidor de correo. Para el desarrollo y pruebas yo uso Mailtrap (mailtrap.io). Sin embargo, para la puesta en producción de un proyecto hay que configurar los datos adecuados a este servicio.

Ahora vamos a configurar los idiomas a emplear. Empezaremos por establecer la matriz, en `config/available_languages.php`.

A continuación modificaremos los archivos de traducciones, creando los que necesitemos, en `resources/lang/`. Para ello, dentro de la carpeta de cada idioma, tendremos que poner la lista de idiomas en `general/idiomas.php`, y la lista de opciones que vayamos a crear para el menú lateral en `general/menu_lateral.php`. En este archivo hay que limpiar primero el contenido por defecto, ya que sólo se usa a efectos de la plantilla “en bruto”, antes de crear ningún proyecto concreto.

El siguiente paso es crear, en la tabla `permissions`, los accesos a rutas que se vayan a utilizar a través del menú lateral. Para ver como se crean, la plantilla incluye algunos enlaces predeterminados que, desde luego, habrán de borrarse. Hay que prestar especial atención a los siguientes puntos:

- **El campo** `rotulo`. Deberá contener la clave que se usará en las matrices de idiomas para cada opción disponible. Aunque en la versión 1.0 esto hay que hacerlo

manualmente, está previsto, para posteriores versiones que el archivo **general/menu_lateral.php** de cada idioma empleado se genere semiautomáticamente, de forma que sólo habrá que poner los valores traducidos.

- **El campo ruta**. Se pondrá el valor del prefijo, fijado con el método **name()** de cada grupo de rutas, seguido del nombre, fijado con el método **name()** de cada ruta dentro de un grupo (observa los ejemplos que vienen con la plantilla). Está previsto que en versiones posteriores de la plantilla el archivo de rutas se genere de forma automática, aunque en esta versión todavía es manual.
- **El campo scope**. Se pondrá una **U**, una **A** o una **M**, según el rango de usuarios a que se destine el enlace.
- **El campo type**. Se pondrá una **G**, si lo que estamos incluyendo es un grupo de opciones, o una **O**, si es una opción. Llegados a este punto, debemos aclarar que el valor de **scope** para una opción deberá ser el mismo que para el grupo al que pertenece dicha opción. Está previsto que, en próximas versiones de la plantilla, esto se controle de forma automática.
- **El campo parent_id**. Para cada grupo de opciones se pondrá en este campo el mismo valor que tenga el grupo en el campo **id**, es decir, cada grupo depende de sí mismo. Para las opciones de pondrá en este campo el valor de **id** del grupo al que pertenezca la opción. Es mediante este campo que se establece, por tanto, la jerarquía de grupos y opciones.

Ahora debemos crear las relaciones entre las opciones de **permissions**, y los usuarios de **users**, lo que hacemos en la tabla pivote **permission_user**. Es importante tener en cuenta que si relacionamos opciones de un determinado grupo con un usuario específico, este también deberá estar relacionado con el grupo al que pertenecen dichas opciones. Está prevista la automatización de este detalle en futuras versiones de la plantilla.

Para probar las funcionalidades de la plantilla, esta incluye tres usuarios, llamados **admin01**, **user01** y **master01**, todos con la contraseña **password**. Una vez desarrollado el proyecto, estod deberán eliminarse.

ACTUALIZACIONES DE LARAVEL

Se ha desarrollado esta plantilla sobre la versión 5.8 de Laravel, previendo la compatibilidad con versiones posteriores. Sin embargo, en las últimas versiones de Laravel hay un par de aspectos que debemos tener en cuenta.

A partir de la versión 6, cambió la forma de crear el scaffolding de autenticación. Esto no es importante, ya que dicho scaffolding ya va creado en la plantilla, por lo que, usemos la versión que usemos (siempre 5.8 o posterior), no habrá problemas.

En la versión 8 de Laravel los componentes de Blade ya no se crean de la forma tradicional, sino mediante clases que se almacenan en el directorio **`app/Components`**. Por lo tanto, si vamos a usar la versión 8, u otra posterior que salga en su momento, deberemos recrear el componente que hay en **`resources/views/components/password.blade.php`**. También deberemos cambiar la forma de usarlo enla vista de registro de usuarios, sita en **`resources/views/auth/register.blade.php`** y enla de autenticación, que se encuentra en **`resources/views/auth/login.blade.php`**.