- Creamos el proyecto en Laravel 6:
    - composer create-project --prefer-dist laravel/laravel <name>
- Establecemos la base de datos en .env
- Instalamos el paquete laravel/ui usando composer
    - comopser require laravel/ui
- Creamos el scaffolding usando php artisan ui vue --auth
- Ejecutamos npm install && npm run dev, para generar los assets del scaffolding de autorización
- Migramos la base de datos
    - php artisan migrate

      la base de datos 'users' tiene un campo 'email_verified_at' donde se almacena la fecha y la hora cuando el usuario verificó su email.

- Establecemos el .env los datos para el envio de emails

```
MAIL_DRIVER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=<YOUR_MAILTRAP_USERNAME>
MAIL_PASSWORD=<YOUR_MAILTRAP_PASSWORD>
MAIL_ENCRYPTION=tls
```

- Necesitamos implementar la interfaz MustVerifyEmail en el modelo App/User

```
<?php
namespace App;

use Illuminate\Notifications\Notifiable;
use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable implements MustVerifyEmail
{
    use Notifiable;

    protected $fillable = [
        'name', 'email', 'password',
    ];

    protected $hidden = [
        'password', 'remember_token',
    ];

    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
}
```

- En routes/web.php, añadimos

```
Auth::routes(['verify' => true]);
```

  - Esto añade las rutas email/verify y email/resend a la aplicación

- Siguiente, en app/Http/Controllers/HomeController.php añadimos el middleware verified

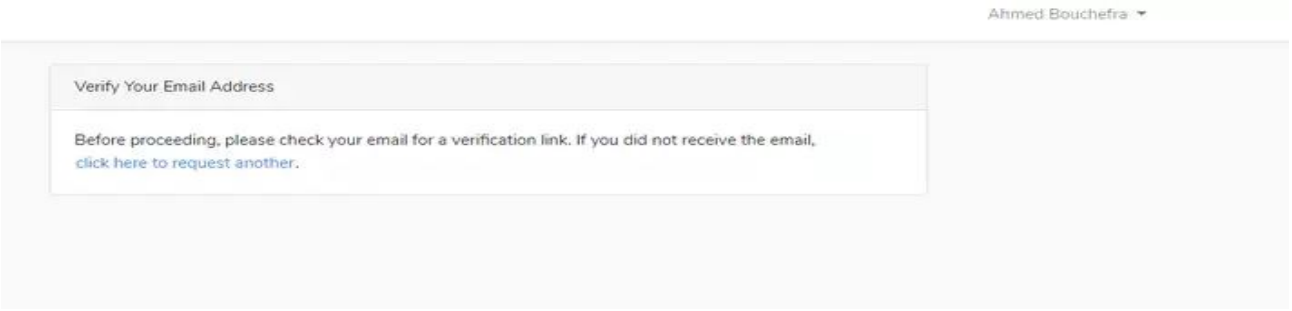
```
class HomeController extends Controller
{

   public function __construct()
   {
      $this->middleware('verified');
   }
```

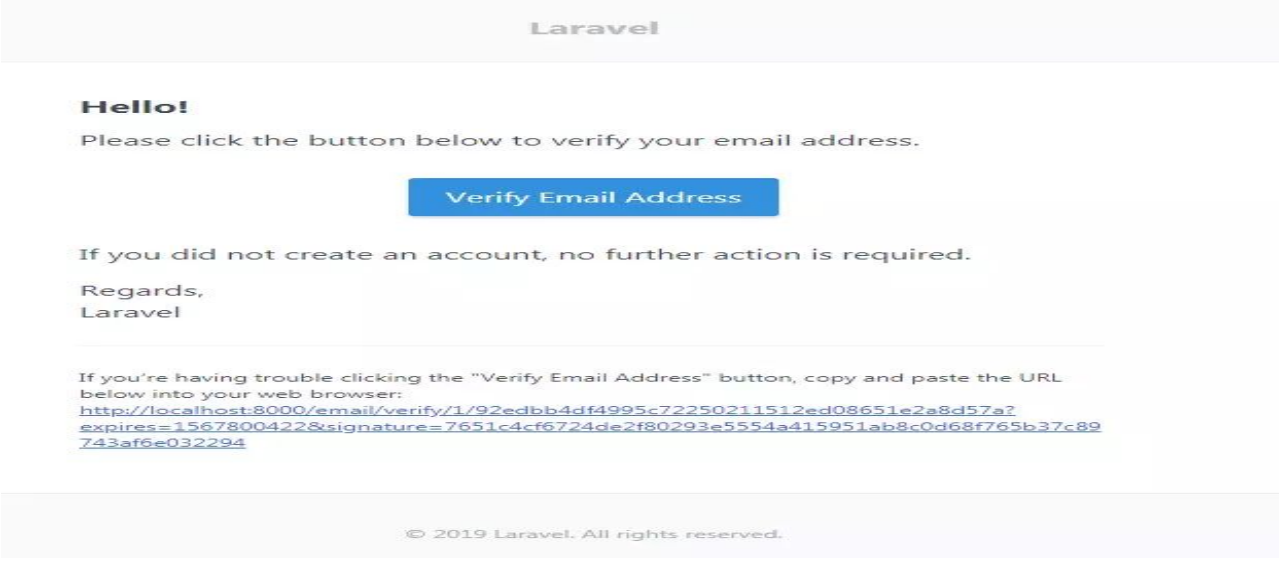
  - Si ya teníamos el middleware auth , añadimos el middleware anterior como un array

```
public function __construct()
   {
      $this->middleware(['auth', 'verified']);
   }
```

A partir de ahora, cuando nos registremos en nuestra aplicación, Laravel nos va a enviar un correo para que confirmemos nuestro registro



Y en nuestro buzón de correo debemos tener este email



El email se ve profesional, pero si queremos podemos cambiar el título, para que no ponga Laravel, en el archivo .env podemos cambiar esto.

```
APP_NAME=CRMApp
```

- Para cambiar la url de redirección una vez autentificado y verificado el email, debemos ir al controlador app/Http/Controllers/Auth/VerificationController.php

  Cambiamos la url de redirección en :

  ```
  protected $redirectTo = '/home'
  ```

  por la url que queramos se redireccione.

- La vista que se nos muestra para la verificación del email se encuentra en resources/views/auth/verify.blade.php, Ahi podemos modificar lo que queramos para personalizar la plantilla.

- Si queremos hacer la app multilenguaje, podemos poner en el fichero es.json o en el que definamos en lang/es las claves para traducir.

- El email de verificación que se envia , se hace a través de una notificación llamada VerifyEmail, que está en vendor/laravel/framework/src/Illuminate/Auth/Notifications/VerifyEmail.php

  Para traducir a distintos idiomas esta notificación, no debemos hacerla aqui, en el archivo del framework, hay que hacerlo con las traducciones del archivo lang/es.json

- En el archivo de rutas web, estamos usando un middleware que está definido en Kernel.php

  ```
  'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
  ```

- Para sobreescribir la notificación de envio y si queremos enviar la misma notificación, que se encuentra en vendor/laravel/framework/src/Illuminate/Auth/MustVerifyEmail.php
  - copiamos el método sendEmailVerificationNotification()
  - lo pegamos en el modelo User.php, pero añadiendo la ruta completa para el envio, que sería

    ```
    /**
    * Send the email verification notification.
    *
    * @return void
    */
    public function sendEmailVerificationNotification()
    {
       $this->notify(new \Illuminate\Auth\Notifications\VerifyEmail);
    }
    ```

- Para crear nuestra propia notificación (Podemos llamarla de la misma forma)
    - php artisan make:notification VerifiEmail

  Ahí copiamos todo el cuerpo de la clase del framework VerifyEmail y lo pegamos.

- Podemos añadir al mensaje
    - ->greeting (‘Hola ‘ . $notifiable->name) // título del mensaje
    - ->salutation((‘Saludos’) // Saludo final

- Si queremos cambiar en final del email, donde dice que si presenta problemas y más, tenemos que seguir con otro paso adicional.
    - Ejecutamos: php artisan vendor:publish --tag=laravel-notifications
    - Con esto se nos crea una carpeta en views/vendor/notifications, donde estaría la parte del pie del mensaje en la parte del component mail::subcopy, y se puede modificar de acuerdo a sus requerimientos:

      ```
      @component('mail::subcopy') Si usted está teniendo problemas para hacer clic en el botón "{{ $actionText }}" copia y pega el siguiente URL en su navegador web: [{{ $actionUrl }}]({{ $actionUrl }})
      @endcomponent
      ```