INVIARE UNA MAIL DI RISPOSTA

Utilizziamo *mailtrap.io* per la creazione di una casella di posta per i nostri esercizi.
Nell'inbox cerchiamo i setting di Laravel e li inseriamo nei nei *setting corrispondenti* nel file **.env**

NB: Se utilizziamo una versione di Laravel inferiore alle 8, dovremo manualmente riavviare il server per permettere alle modifiche di essere salvate. *CMD+C + php artisan serve*

Generiamo una ***Mailable*** , una classe di Laravel che mi permette di inviare una mail.
- *php artisan make:mail ContactMail* nella Bash crea il in *app/Mail/ContactMail.php*

```
class ContactMail extends Mailable
{
	public $contact;

	public function __construct($contact){
		$this->contact = $contact;
	}

	public function build()
	{
		return $this->from('amministrazione@aulab.it')
				->view('mails.contact-mail');
	}
}
```

from **.** La mail dal quale parte la mail.
view . La pagina nelle *view* che avrà il corpo della Mail. Creiamo una cartella Mails in view e un file *contact-mail.blade.php*

Ora dobbiamo ***istruire il controller*** (nel caso del nostro esempio il precedente *ContactController*) ad inviare una mail.
Lo facciamo attraverso una rotta, ma prima di tutto separiamo per ogni key l'array che riceviamo dal form con tutti i nostri dati.

```php
class ContactController extends Controller{
	public function submit(ContactRequest $request){

		$username = $request-input('username');
		$email = $request-input('email');
		$message = $request-input('message');

		$contact = compact('username', 'message');

//dd($username, $email, #message); verifichiamo di ottenere i dati separati

		Mail::to($email)->send(new ContactMail($contact));

		return redirect(route('thankyou'));
	}

	public function thankyou(){
		return view('thankyou');
	}
}
```

compact() . Assegniamo ad una variabile due valori che abbiamo spacchettato.
to() . Indichiamo la mail che riceverà il messaggio sempre dall'array che abbiamo spacchettato.
send() . Facciamo una *composition* creando una nuova Mailable, con la variabile che vogliamo trasportare.
redirect() **.** Il return di una rotta di tipo ***post*** e' sempre un redirect per farlo uscire dalla rotta, altrimenti rimarrebbe incastrato dentro. Il redirect deve essere una rotta di tipo ***get***. Creiamo quindi una ***route*** e una ***view*** ad essa collegata (es *thankyou*), e richiamiamo la funzione.

NB: Ricordiamoci ***sempre*** di importare le classi quando le richiamiamo.

Creiamo ora il ***corpo*** della mail (con il solo HTML e Style in linea) in *contact-mail.blade.php* che abbiamo creato prima utilizzando come reference alle variabili la seguente modalità.

```php
<h1>Grazie per esserti iscritto {{$contact['username']}}</h1>
```

Tools Utili:
Fake Filler (Chrome Extension) per riempire con testo i form in automatico. ***plugin***
mailtrap.io . Simulare una cartella di posta per invio e ricezione mail di risposta. ***sito***