

# Laravel Telescope

Telescope est un nouvel assistant pour déboguer une application Laravel. IL nous donne accès à une foule d'informations sur les requêtes qui entrent dans l'application, sur les exceptions, les requêtes à la base de données, les files d'attente (queues), les mails, les notifications, le cache...

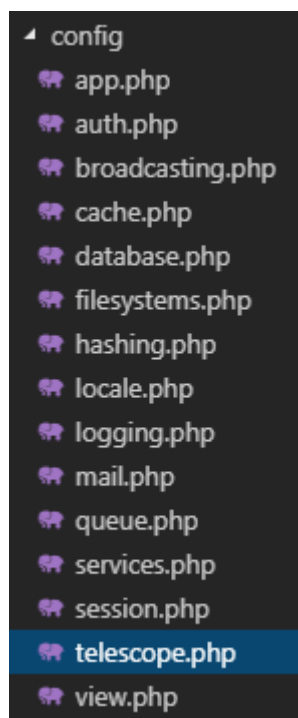
Telescope en est encore au stade beta mais il est déjà largement utilisable. Je vous propose dans cet article de regarder un peu ses possibilités. On va ainsi enfin disposer pour Laravel d'un outil digne de ce nom !

## Installation

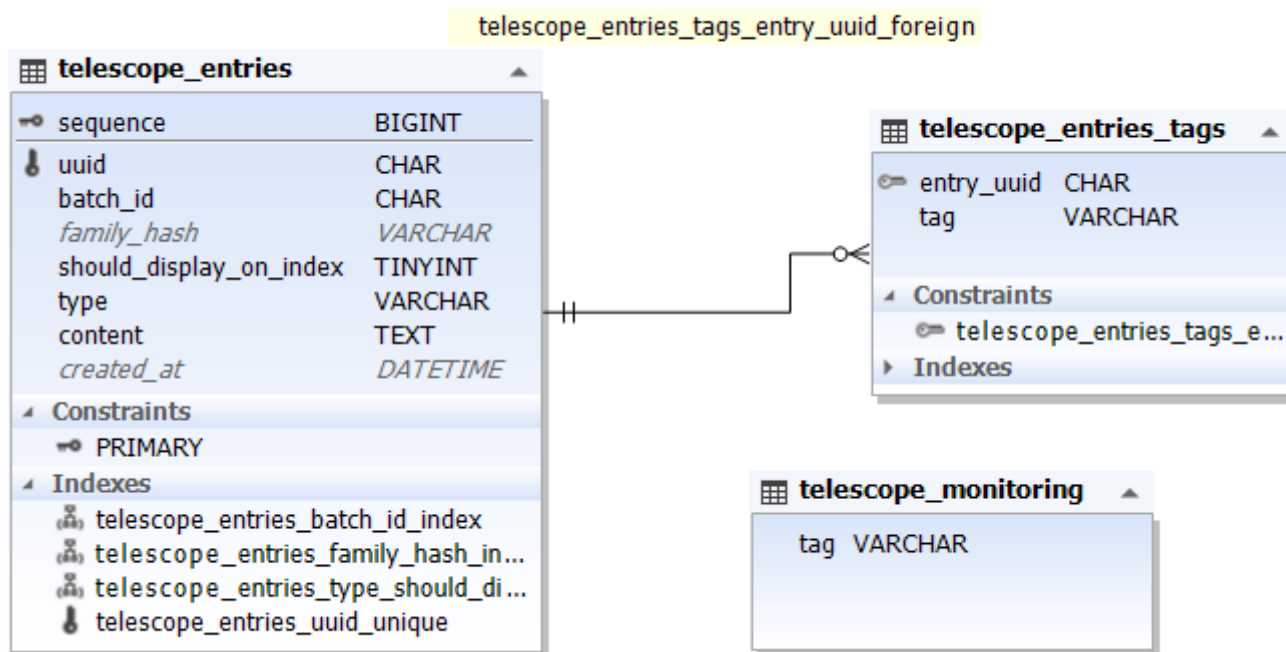
Pour installer Telescope c'est tout simple, mais il faut au minimum la version 5.7.7 de Laravel :

```
composer require laravel/telescope --dev
php artisan telescope:install
php artisan migrate
```

On se retrouve avec un fichier de configuration :



Dans ce fichier on trouve par défaut comme driver **database** avec une connexion **MySQL**. Avec la migration qu'on a faite on se retrouve avec ces 3 tables :



On a tous ces observateurs par défaut :

```

'watchers' => [
    Watchers\CacheWatcher::class => env('TELESCOPE_CACHE_WATCHER',
true),
        Watchers\CommandWatcher::class =>
env('TELESCOPE_COMMAND_WATCHER', true),
    Watchers\DumpWatcher::class => env('TELESCOPE_DUMP_WATCHER',
true),
    Watchers\EventWatcher::class => env('TELESCOPE_EVENT_WATCHER',
true),
        Watchers\ExceptionWatcher::class =>
env('TELESCOPE_EXCEPTION_WATCHER', true),
    Watchers\JobWatcher::class => env('TELESCOPE_JOB_WATCHER',
true),
    Watchers\LogWatcher::class => env('TELESCOPE_LOG_WATCHER',
true),
    Watchers\MailWatcher::class => env('TELESCOPE_MAIL_WATCHER',
true),
    Watchers\ModelWatcher::class => env('TELESCOPE_MODEL_WATCHER',
true),
        Watchers\NotificationWatcher::class =>
env('TELESCOPE_NOTIFICATION_WATCHER', true),
  
```

```
Watchers\QueryWatcher::class => [  
    'enabled' => env('TELESCOPE_QUERY_WATCHER', true),  
    'slow' => 100,  
],
```

```
Watchers\RedisWatcher::class => env('TELESCOPE_REDIS_WATCHER',  
true),  
    Watchers\RequestWatcher::class =>  
env('TELESCOPE_REQUEST_WATCHER', true),  
    Watchers\ScheduleWatcher::class =>  
env('TELESCOPE_SCHEDULE_WATCHER', true),  
],
```

On peut donc enlever ceux qui ne nous sont pas utiles.

Un réglage important est le nombre maximum d'enregistrements par observateur :

```
'limit' => env('TELESCOPE_LIMIT', 100),
```

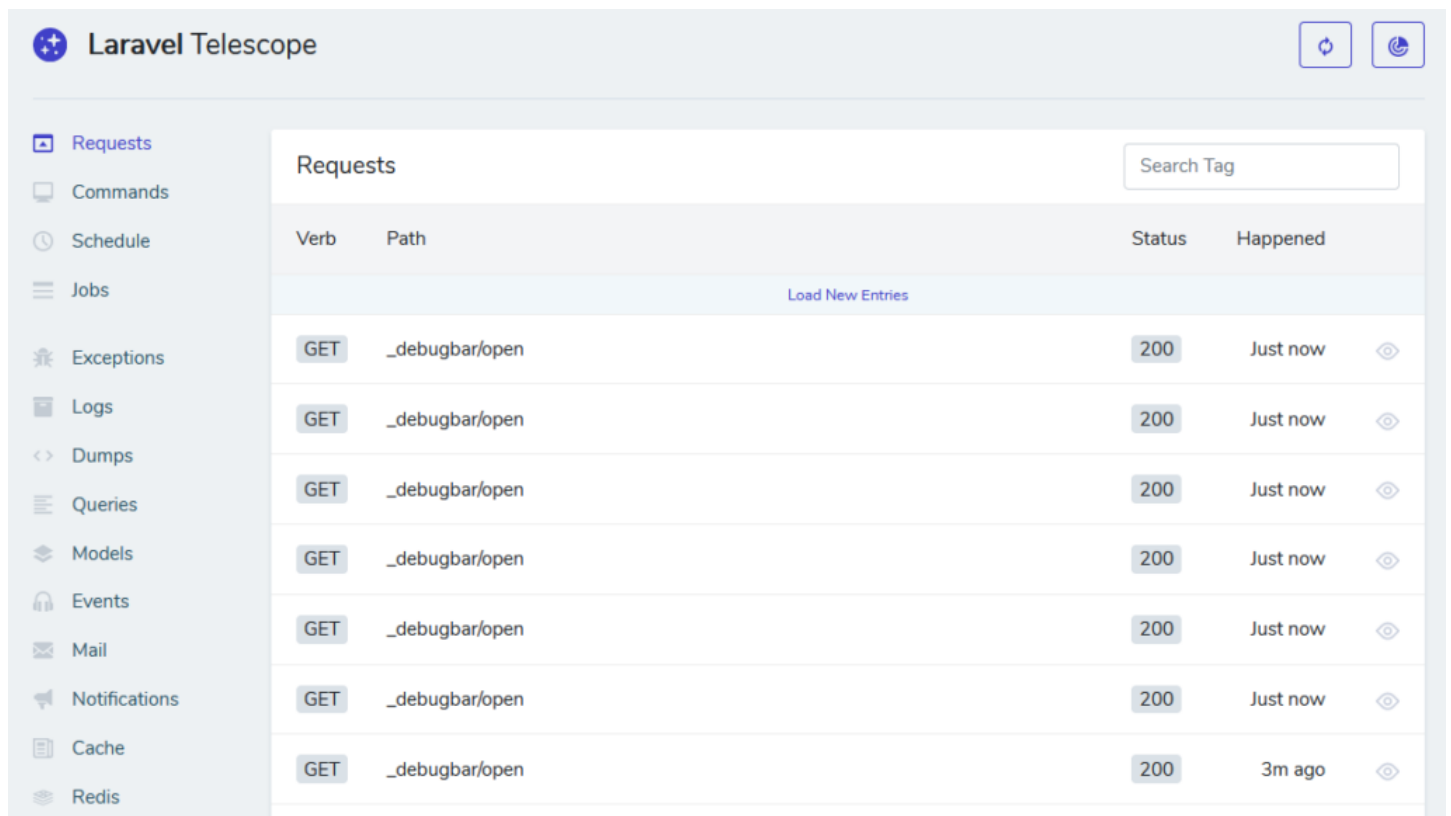
Par défaut on a 100, évidemment si on augmente cette valeur il y aura un impact sur les données stockées dans la base.

Pour accéder à Telescope il faut utiliser cette url :

<http://mondomaine/telescope>

Cette url est aussi réglable dans la configuration :

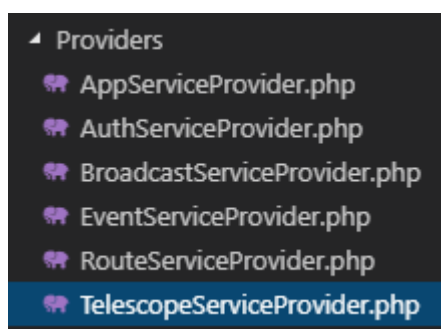
```
'path' => 'telescope',
```



Par défaut Telescope n'est accessible que dans l'environnement local :

`APP_ENV=local`

C'est logique puisque que c'est comme ça qu'on développe mais il peut arriver d'en avoir besoin dans l'environnement de production. Telescope crée un provider :



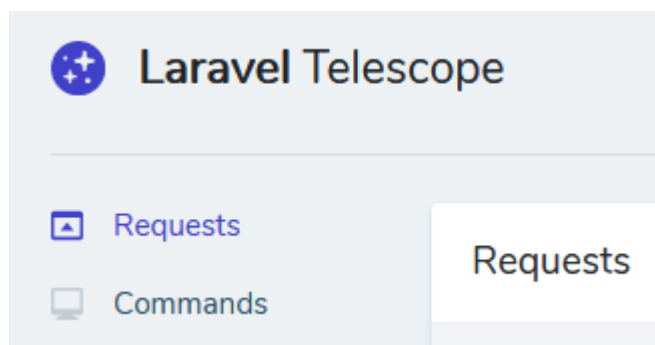
Dans ce provider on trouve une autorisation où vous pouvez ajouter les mails des personnes autorisées dans l'environnement de production :

```
protected function gate()
{
    Gate::define('viewTelescope', function ($user) {
        return in_array($user->email, [
```

```
        'bestmomo@chezlui.net'  
    });  
}
```

Maintenant qu'on a fait le tour pour l'installation et les réglage ouvrons un peu la boîte...

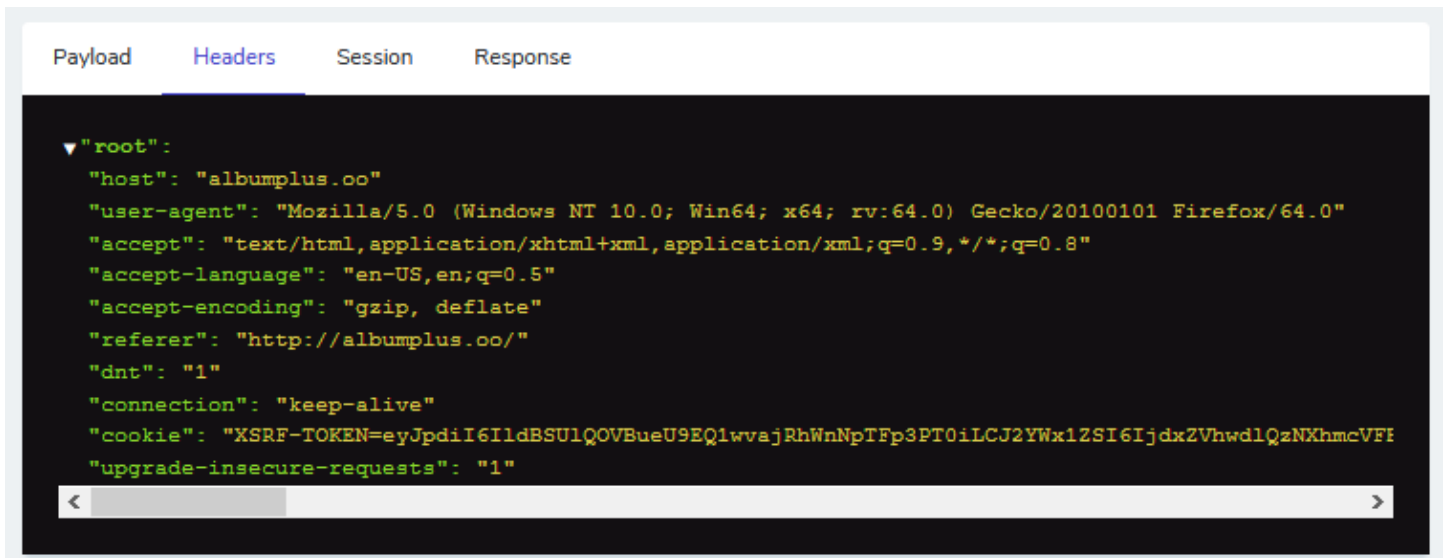
## Les requêtes



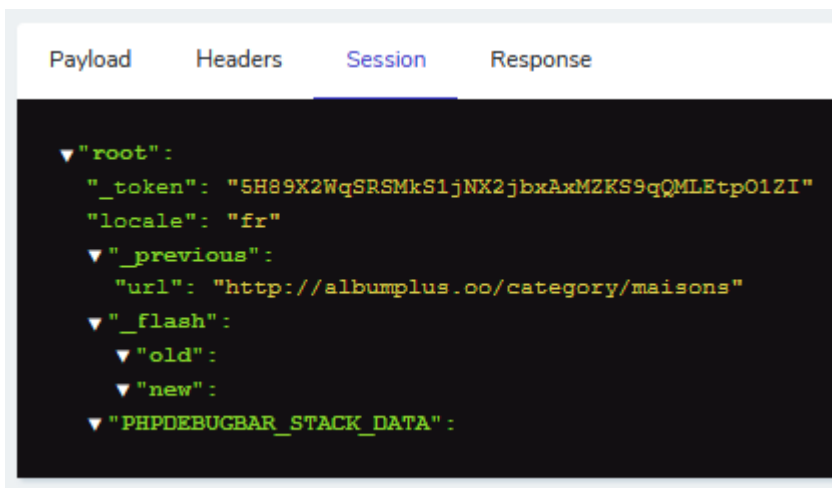
On a la liste des requêtes avec le verbe, l'url, le code HTTP et quand elle est passée. On dispose aussi de l'icône d'un œil pour obtenir plus de renseignements concernant une requête :

GET	category/maisons	200	Just now	👁
GET	/	200	Just now	👁
GET	language/fr	302	Just now	👁
GET	/	200	Just now	👁

On dispose alors de tous les détails comme les **headers** :



La session :



Les requêtes générées à la base :

Queries (12)		
Query	Duration	
select `users`.*, `image_user`.`image_id` as `pivot_image_id`, `image_user`.`user_id` as `pivot_user_id`...	0.45ms	🔍
select `users`.*, `image_user`.`image_id` as `pivot_image_id`, `image_user`.`user_id` as `pivot_user_id`...	0.44ms	🔍
select `users`.*, `image_user`.`image_id` as `pivot_image_id`, `image_user`.`user_id` as `pivot_user_id`...	0.63ms	🔍
select `users`.*, `image_user`.`image_id` as `pivot_image_id`, `image_user`.`user_id` as `pivot_user_id`...	0.68ms	🔍

Avec pour chacune le détail en cliquant sur l'icône :

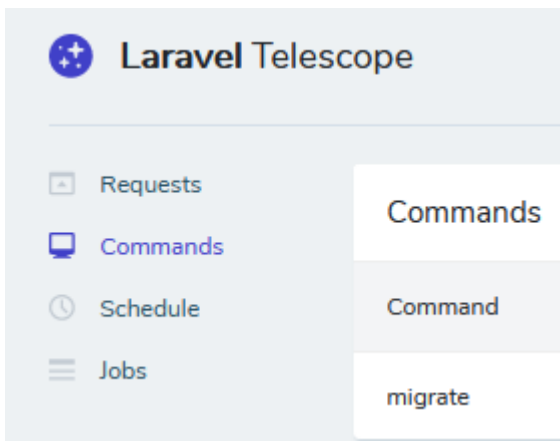
## Query

```
select
  `users`.*,
  `image_user`.`image_id` as `pivot_image_id`,
  `image_user`.`user_id` as `pivot_user_id`,
  `image_user`.`rating` as `pivot_rating`
from
  `users`
  inner join `image_user` on `users`.`id` = `image_user`.`user_id`
where
  `image_user`.`image_id` = 10
```

Vraiment pratique tout ça

!

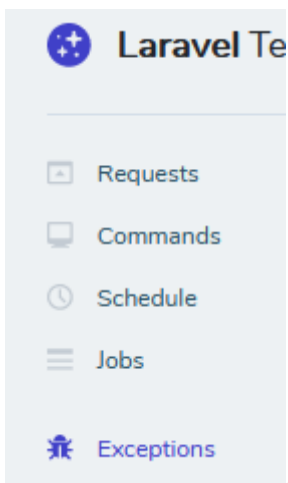
# Les commandes



Pour les commande Artisan c'est le même principe : on a la liste des commandes passées et on peut obtenir le détail :

# Les exceptions

```
Arguments  Options
▼ "root" :
  "command": "migrate"
```



On a la liste des exceptions :

Type	Occurrences	Happened	
ErrorException Call to undefined function App\Providers\auth() (View:...	1	1m ago	
list	0	Just now	
migrate	0	1h ago	

Et le détail pour chacune :



The screenshot shows the 'Exception Details' section of a Laravel application. It lists the following information:

- Time:** October 28th 2018, 2:26:48 PM (2m ago)
- Hostname:** pc-papa
- Type:** ErrorException
- Location:** E:\laragon\www\albumplus\app\Providers\AppServiceProvider.php:19
- Occurrences:** [View Other Occurrences](#)
- Request:** [View Request](#)

Below this, there are three tabs: 'Message', 'Location', and 'Stacktrace'. The 'Message' tab is active, showing the error message: 'Call to undefined function App\Providers\auth()' (View: E:\laragon\www\albumplus\resources\views\layouts\ap...'). A scrollbar is visible at the bottom of the message area.

On a un lien pour accéder à la requête concernée et aux autres occurrences.






## Les requêtes à la base

The screenshot shows the sidebar menu of the Laravel Tinker interface. The menu items are:

- Requests
- Commands
- Schedule
- Jobs
- Exceptions
- Logs
- Dumps
- Queries
- Models

Là aussi on a la liste avec un formulaire de recherche

:


Queries			<input type="text" value="Search Tag"/>
Query	Duration	Happened	
<a href="#">Load New Entries</a>			
select * from `categories`	24.17ms	Just now 	
select * from `categories`	17.05ms	Just now 	
select * from `categories`	3.89ms	Just now 	
select * from `categories`	3.68ms	Just now 	
select * from `categories`	3.78ms	Just now 	

Et pour chaque requête le détail :

Query Details

Time	October 28th 2018, 2:31:51 PM (Just now)
Hostname	pc-papa
Connection	mysql
Duration	0.99ms
Request	<a href="#">View Request</a>
Tags	<span>Auth:1</span>

Authenticated User

ID	1
Name	 Durand
Email Address	durand@chezlui.fr

Query

```
select
  count(*) as aggregate
from
  `images`
```

On dispose de l'utilisateur connecté et du détail de la requête.

## Les événements

- Queries
- Models
- Events
- Mail
- Notifications

On a aussi la liste :

Events			
Name	Listeners	Happened	
App\Events\UserCreated	1	Just now	

Et le détail :

<b>Event</b>	App\Events\UserCreated
<b>Job</b>	<a href="#">View Job</a>
<b>Request</b>	<a href="#">View Request</a>
<b>Tags</b>	App\Models\User:4


Event Data	Listeners
<pre> "root":   "user": "App\Models\User:4" </pre>	

On a un lien pour accéder au job associé (ici pour l'envoi du mail de confirmation) et à la requête.

## Les jobs

- Commands
- Schedule
- Jobs**
- Exceptions
- Logs

Pour les jobs on a aussi la liste avec le statut pour chacun :

Jobs		Search Tag	
Job		Status	Happened
App\Notifications\UserCreated Connection: redis   Queue: queues:default		pending	3m ago 

Là on voit que c'est en attente, après exécution on a cet aspect :

Status

processed

On peut aussi accéder au détail :









Status	processed
Job	App\Notifications\UserCreated
Connection	redis
Queue	queues:default
Tries	-
Timeout	-
Tags	App\Models\User:4

#### Data

```
▼ "root":
  "notifiables": "App\Models\User:1"
  ▼ "notification":
    "class": "App\Notifications\UserCreated"
    ▼ "properties":
      "id": "50fda5ba-1a15-43e5-8127-5161d110adf2"
      "locale": null
      "connection": null
      "queue": null
      "chainConnection": null
      "chainQueue": null
      "delay": null
      ▶ "chained": 0 items
    ▼ "channels":
      0: "mail"
      "connection": null
      "queue": null
      "chainConnection": null
      "chainQueue": null
      "delay": null
    ▼ "chained":
```

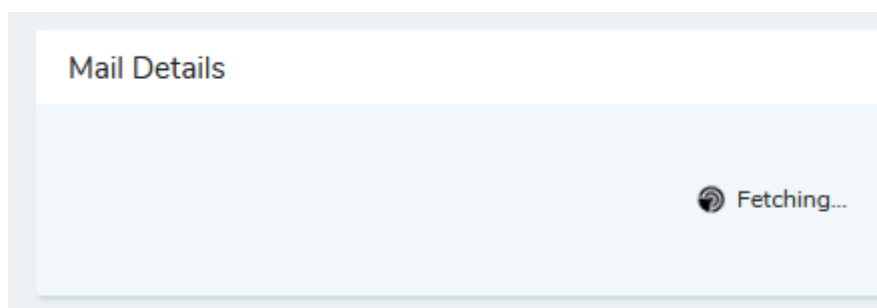
## Les mails

- Models
- Events
- Mail
- Notifications

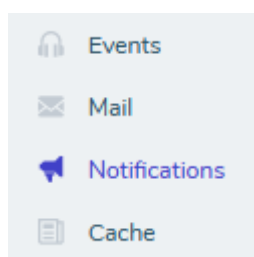
On a la liste des mails envoyés :

Mail		Search Tag	
Mailable		Recipients	Happened
App\Notifications\UserCreated	Queued	1	3m ago
Subject: Nouvel utilisateur			
App\Notifications\UserCreated	Queued	1	3m ago
Subject: Nouvel utilisateur			




Il est précisé s'ils sont dans une file d'attente. Par contre je n'ai pas réussi à accéder au détail des mails... On verra dans la prochaine version... Je suis resté coincé ici :



## Les notifications



On a la liste des notifications :


Notifications		Search Tag	
Notification		Channel	Happened
App\Notifications\ImageRated Recipient: App\Models\User:2		mail	1m ago 
App\Notifications\ImageRated Recipient: App\Models\User:2		mail	1m ago 
App\Notifications\UserCreated Recipient: App\Models\User:1	Queued	mail	15m ago 

On peut accéder aux détails :

<b>Channel</b>	mail
<b>Notification</b>	App\Notifications\ImageRated
<b>Notifiable</b>	App\Models\User:2
<b>Request</b>	<a href="#">View Request</a>
<b>Tags</b>	<span>App\Models\User:2</span> <span>Auth:1</span>

---

Authenticated User

<b>ID</b>	1
<b>Name</b>	 Durand
<b>Email Address</b>	durand@chezlui.fr

## Conclusion

Je n'ai pas passé en revue toutes les possibilités, je vous laisse les découvrir !