

# Cybersécurité

---

Consigne :

- 1- Installer une VM Debian
- 2- Installer framework php sur la VM (au choix symfony, laravel...)
- 3- Configurer un écran de login basé sur JWT
- 4- Configurer une restriction du nombre tentative de login échoué (au bout de 3 tentatives ratés on inscrit dans un fichier de log)

Pour la première partie, une machine virtuelle a été installé suivant l'utilisation de Debian 10 :

<https://www.debian.org/releases/stable/>

Pour la deuxième partie, j'ai décidé d'installer Laravel qui est le plus rapide à mettre en place et dont je maîtrise plus facilement. Pour ce faire, j'ai rajouté des packages pour faire de l'administration plus facilement :

Préconfig (1) :

<https://wiki.debian.org/fr/SSH> (pour la connexion distante)

<https://linuxize.com/post/how-to-install-and-use-composer-on-debian-10/>

Tuto pour la mise en place de laravel sous nginx

<https://blog.here-host.com/install-laravel-debian-9/>

<https://laravel.com/docs/8.x>

Mise en place d'une bdd pour permettre de faire les tests d'authentification

[https://linuxhint.com/install\\_phpmyadmin\\_debian\\_10/](https://linuxhint.com/install_phpmyadmin_debian_10/)

Ressource code :

[https://github.com/Darylaborador/cybersecurite\\_projets](https://github.com/Darylaborador/cybersecurite_projets)

## Mise en place de laravel sur la VM

---

### Sources :

<https://www.fosslinux.com/9284/how-to-install-laravel-on-debian-9.htm>

<https://www.howtoforge.com/tutorial/install-laravel-on-ubuntu-for-apache/> (pour le fichier config)

On passe en tant qu'utilisateur root pour toute les commandes qui suivent : su –

### Installation des dépendances php et apache2 :

```
apt install apache2 libapache2-mod-php php php-common php-xml php-gd php-opcache php-mbstring  
php-tokenizer php-json php-bcmath php-zip unzip php-mysql
```

```
apt install curl git unzip
```

### Initialisation d'un projet laravel

```
apt update  
apt upgrade  
cd /var/www/  
composer create-project --prefer-dist laravel/laravel cybersecurityite
```

### Configuration server web apache

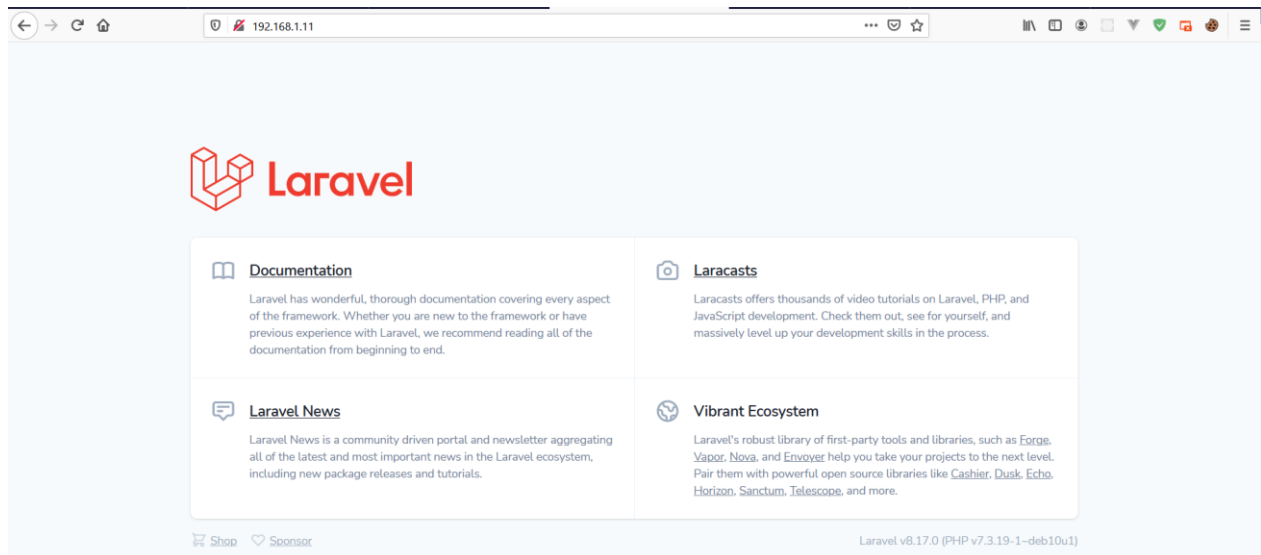
```
chgrp -R www-data /var/www/cybersecurityite  
chown -R secu /var/www/cybersecurityite  
chmod -R 775 /var/www/cybersecurityite/storage
```

```
vim /etc/apache2/sites-available/laravel.conf
```

```
GNU nano 3.2                                laravel.conf  
  
<VirtualHost *:80>  
    ServerName cybersecurityite.tld  
    ServerAdmin admin@localhost  
    DocumentRoot /var/www/cybersecurityite/public  
  
    <Directory /var/www/cybersecurityite/public>  
        Options Indexes FollowSymLinks MultiViews  
        AllowOverride All  
        Order allow,deny  
        allow from all  
        Require all granted  
    </Directory>  
  
    ErrorLog ${APACHE_LOG_DIR}/error.log  
    CustomLog ${APACHE_LOG_DIR}/access.log combined  
</VirtualHost>
```

```
a2dissite 000-default.conf
a2ensite laravel.conf
a2enmod rewrite
service apache2 restart
```

Pour on accède à notre projet en entrant l'adresse ip dans notre navigateur hôte :



Pour connaitre l'adresse ip on fait les commandes :

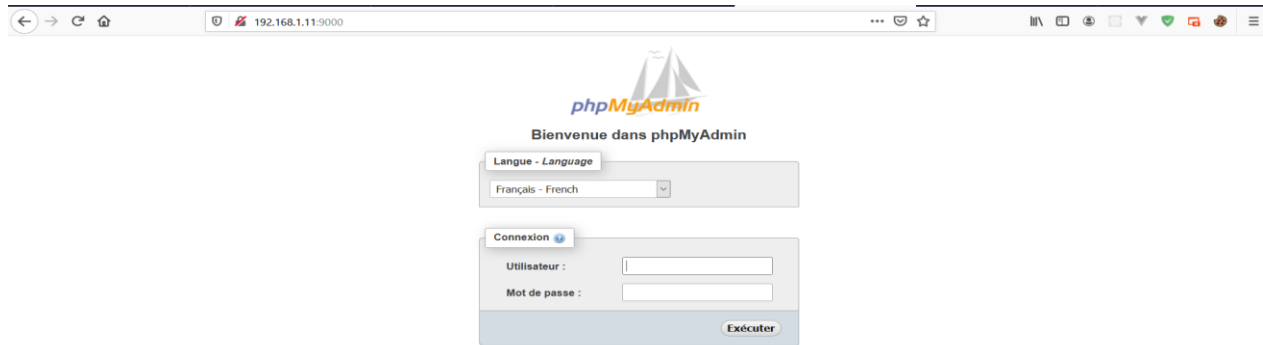
```
Su -
Ip a
```

```
secu@debian:~$ su -
Mot de passe :
root@debian:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:99:20:22 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.11/24 brd 192.168.1.255 scope global dynamic noprefixroute enp0s3
        valid_lft 82689sec preferred_lft 82689sec
    inet6 fe80::a00:27ff:fe99:2022/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
root@debian:~#
```

# Configuration d'une BDD

**Mise en place d'une bdd pour permettre de faire les tests d'authentification :**

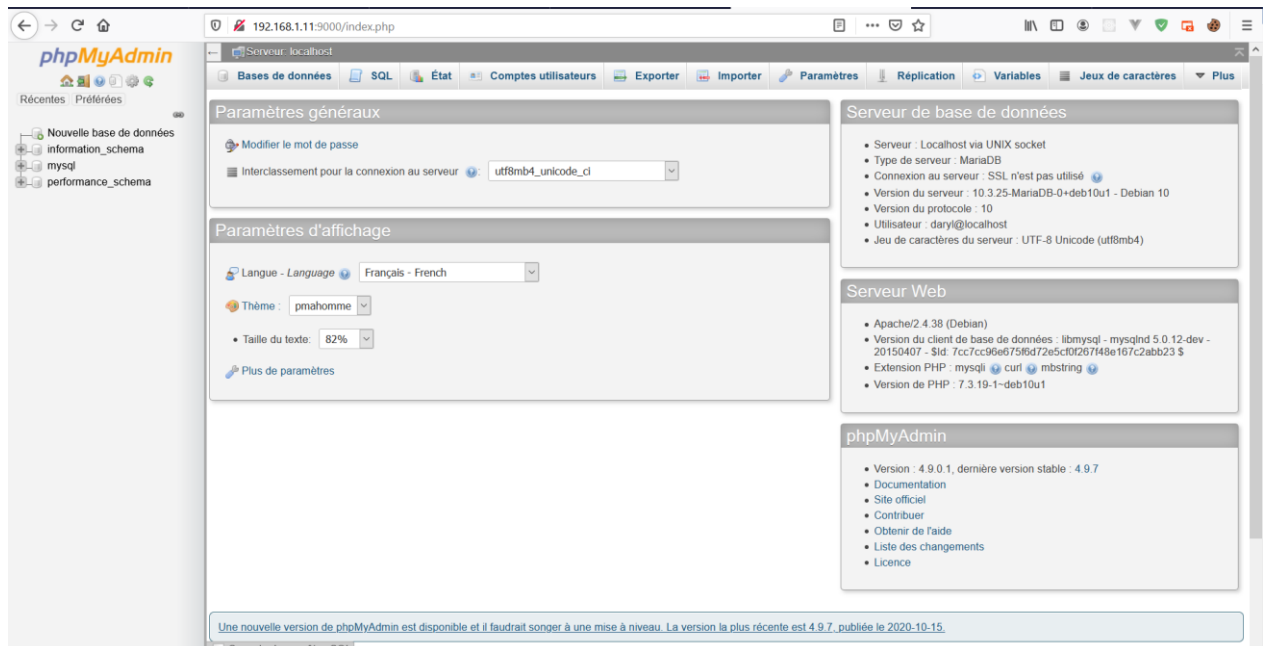
[https://linuxhint.com/install\\_phpmyadmin\\_debian\\_10/](https://linuxhint.com/install_phpmyadmin_debian_10/)



**Je crée un utilisateur pour pouvoir me connecter à l'interface :**

```
Su -  
mysql -u root -p  
GRANT ALL ON *.* TO 'daryl'@'localhost' IDENTIFIED BY '123456';  
FLUSH PRIVILEGES;  
\\q
```

**Je me connecte avec le compte utilisateur précédemment créer :**



## - Création d'une bdd -

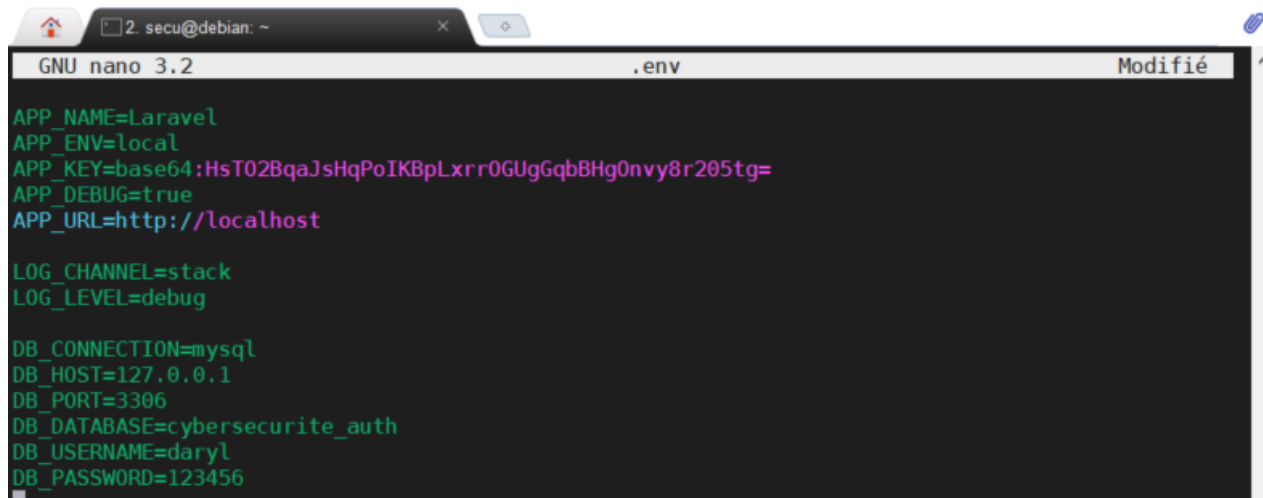
### Configuration projet pour la connexion bdd

---

Création de la BDD :

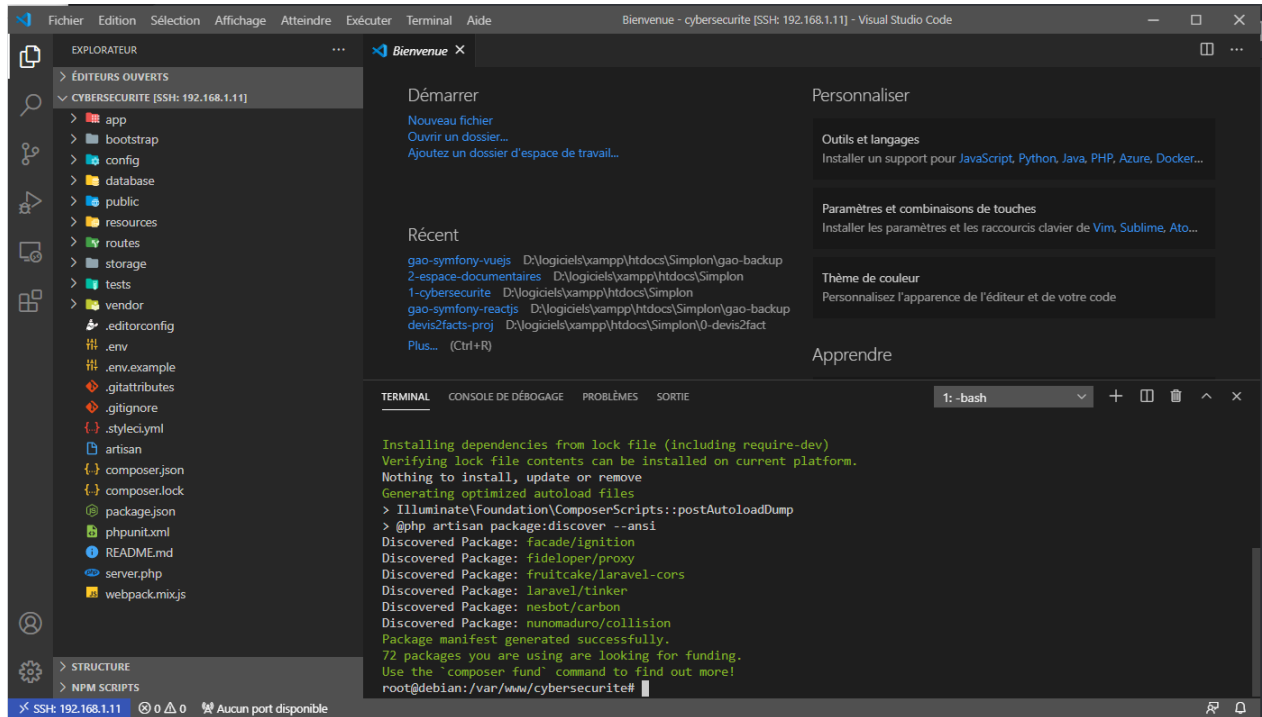


Configuration du .env laravel :

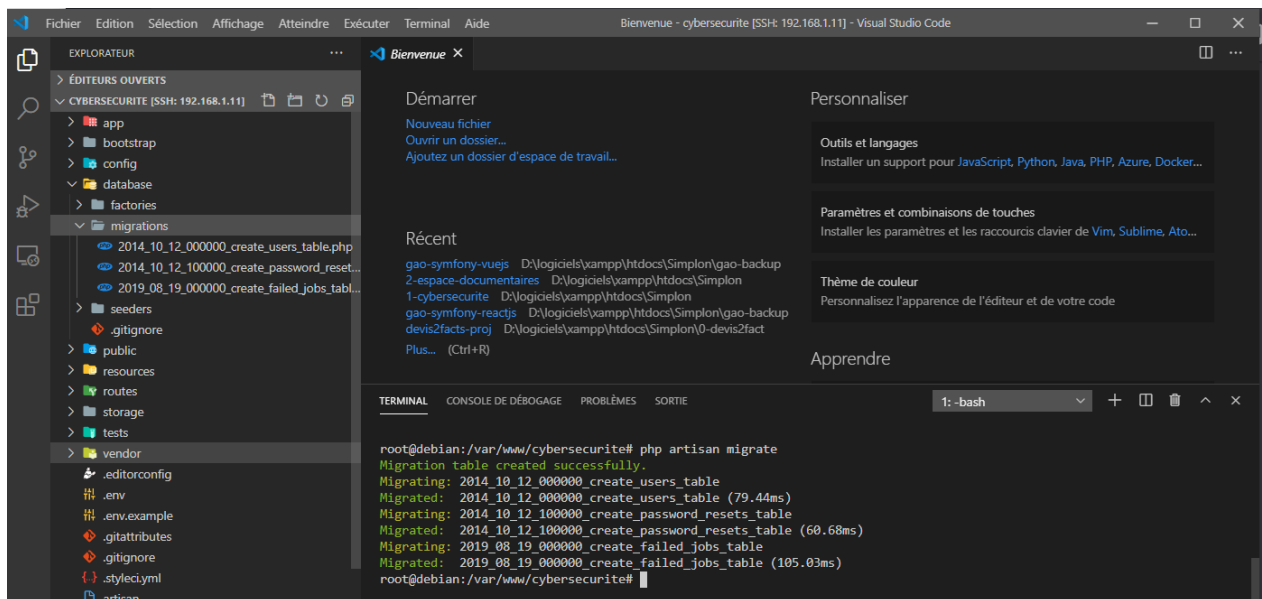


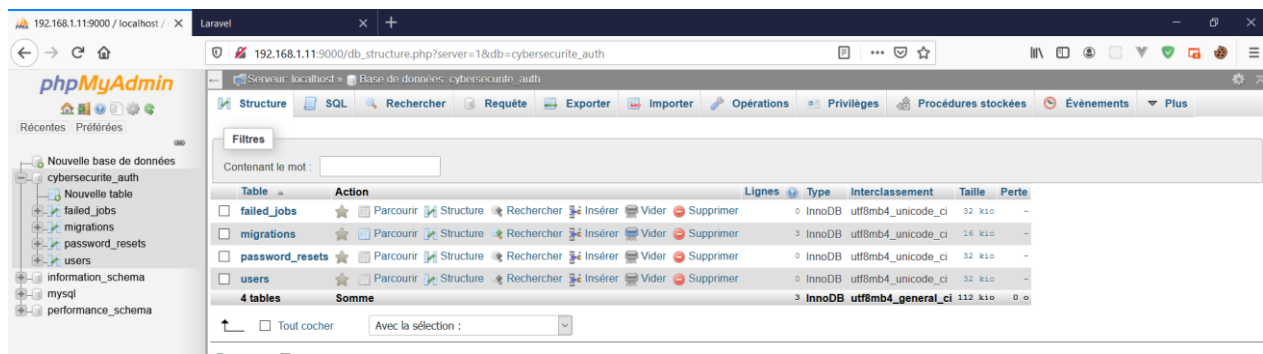
# Initialisation des fichiers config

## Composer install

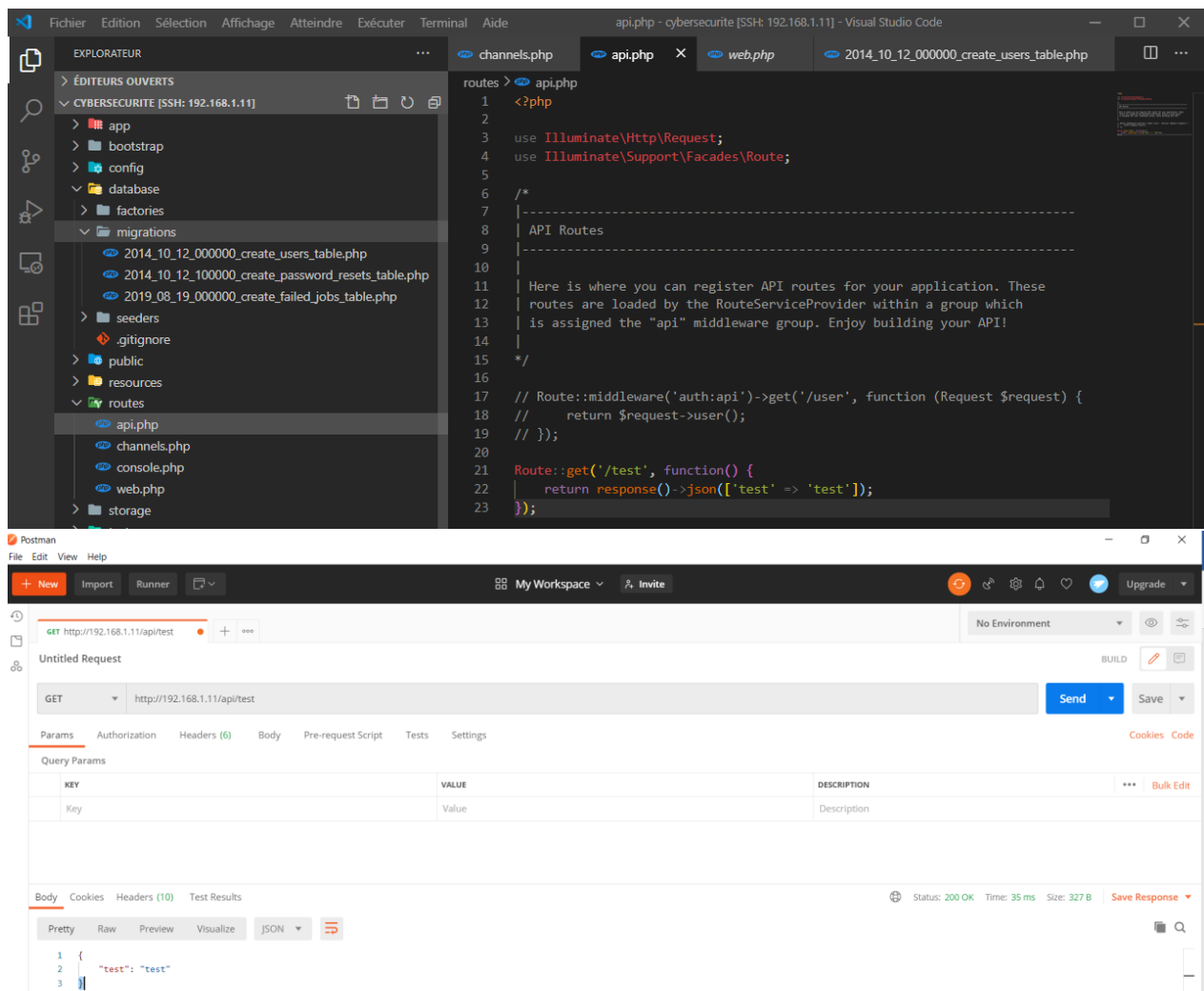


## Test d'une migration pour vérifier la connexion à la BDD :





## Test configuration API :

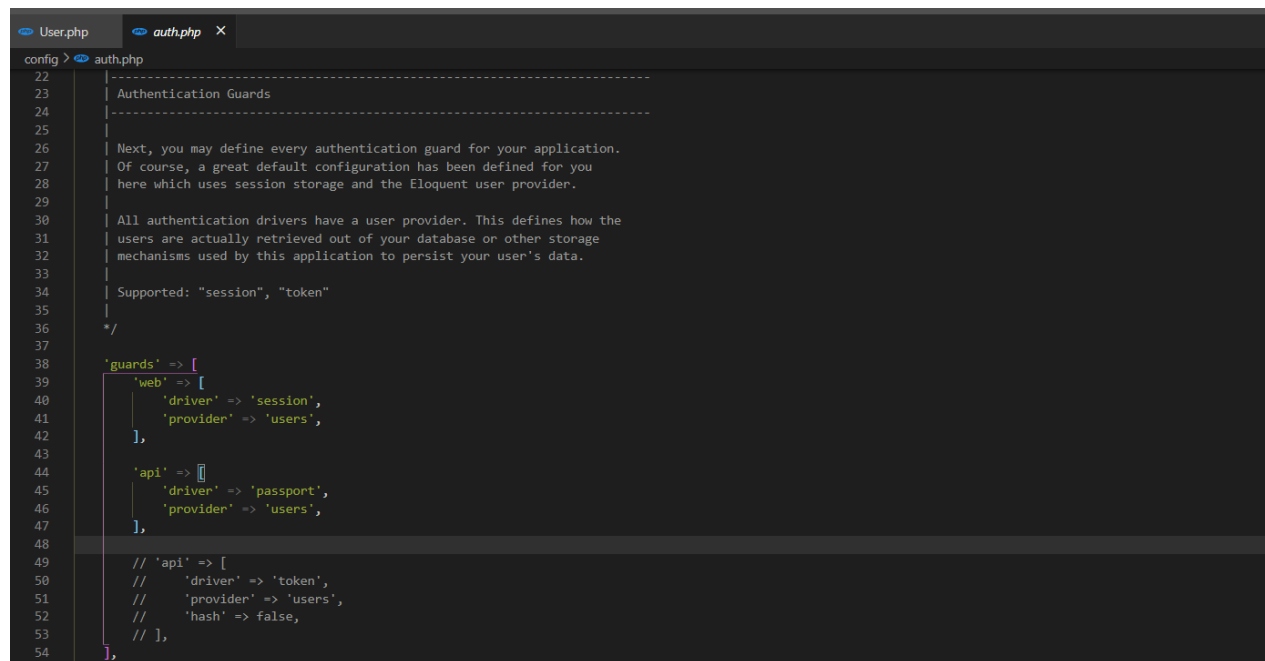


## Configuration JWT par le biais de passport

<https://laravel.com/docs/8.x/passport>

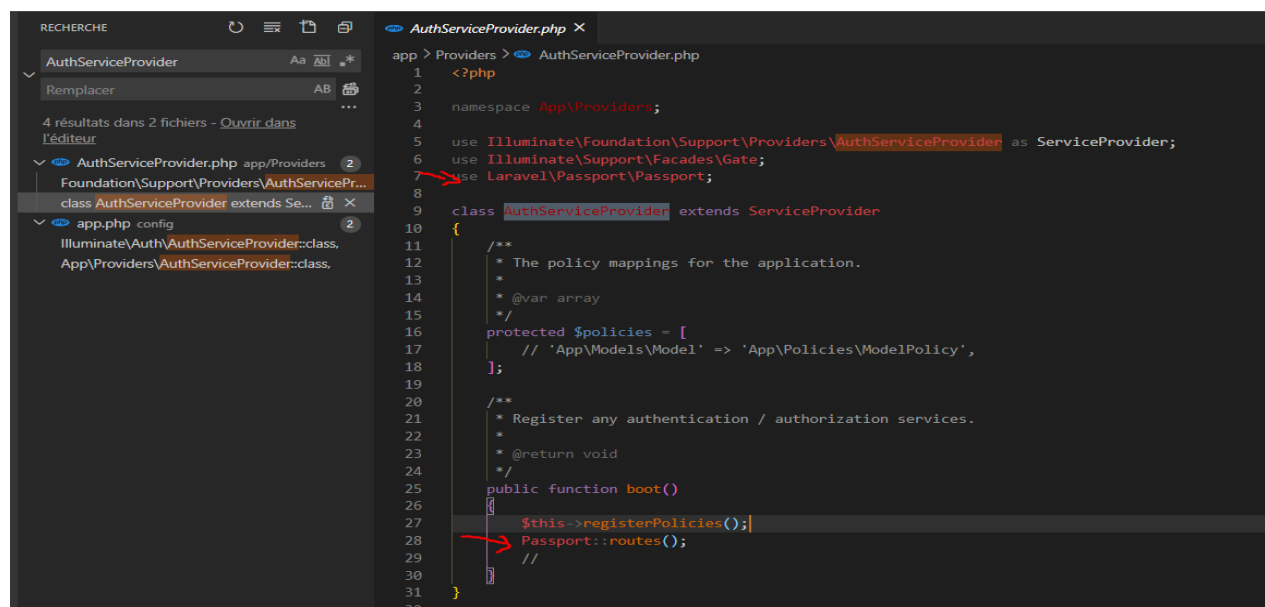
Puis tout ce passe dans le code pour le reste de la configuration

```
composer require laravel/passport
php artisan migrate
php artisan passport:install
```



The screenshot shows the `auth.php` file in the `config` directory. It defines the authentication guards for the application. The `'web'` guard uses the `'session'` driver and the `'users'` provider. The `'api'` guard uses the `'passport'` driver and the `'users'` provider. The `'api'` guard also has a `'hash'` property set to `false`.

```
22 | -----
23 | Authentication Guards
24 | -----
25 |
26 | Next, you may define every authentication guard for your application.
27 | Of course, a great default configuration has been defined for you
28 | here which uses session storage and the Eloquent user provider.
29 |
30 | All authentication drivers have a user provider. This defines how the
31 | users are actually retrieved out of your database or other storage
32 | mechanisms used by this application to persist your user's data.
33 |
34 | Supported: "session", "token"
35 |
36 | */
37 |
38 | 'guards' => [
39 |     'web' => [
40 |         'driver' => 'session',
41 |         'provider' => 'users',
42 |     ],
43 |
44 |     'api' => [
45 |         'driver' => 'passport',
46 |         'provider' => 'users',
47 |     ],
48 |
49 |     // 'api' => [
50 |     //     'driver' => 'token',
51 |     //     'provider' => 'users',
52 |     //     'hash' => false,
53 |     // ],
54 | ],
```



The screenshot shows the `AuthServiceProvider.php` file in the `app/Providers` directory. It defines the `AuthServiceProvider` class, which extends the `ServiceProvider` class. The class has a `boot` method that registers the policies and the passport routes.

```
1 | <?php
2 |
3 | namespace App\Providers;
4 |
5 | use Illuminate\Foundation\Support\Providers\AuthServiceProvider as ServiceProvider;
6 | use Illuminate\Support\Facades\Gate;
7 | use Laravel\Passport\Passport;
8 |
9 | class AuthServiceProvider extends ServiceProvider
10 | {
11 |     /**
12 |      * The policy mappings for the application.
13 |      *
14 |      * @var array
15 |      */
16 |     protected $policies = [
17 |         // 'App\Models\Model' => 'App\Policies\ModelPolicy',
18 |     ];
19 |
20 |     /**
21 |      * Register any authentication / authorization services.
22 |      *
23 |      * @return void
24 |      */
25 |     public function boot()
26 |     {
27 |         $this->registerPolicies();
28 |         Passport::routes();
29 |         //
30 |     }
31 | }
32 |
```



```
User.php x
app > Models > User.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Contracts\Auth\MustVerifyEmail;
6  use Illuminate\Database\Eloquent\Factories\HasFactory;
7  use Illuminate\Foundation\Auth\User as Authenticatable;
8  use Illuminate\Notifications\Notifiable;
9  use Laravel\Passport\HasApiTokens;
10
11 class User extends Authenticatable
12 {
13     use HasApiTokens, HasFactory, Notifiable;
14
15     /**
16      * The attributes that are mass assignable.
17      *
18      * @var array
19      */
20     protected $fillable = [
21         'name',
22         'email',
23         'password',
24         'tentatives'
25     ];
26 }
```

## Restriction du nombre de tentative

Migration user :

```
database > migrations > 2014_10_12_000000_create_users_table.php
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreateUsersTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('users', function (Blueprint $table) {
17             $table->id();
18             $table->string('email')->unique();
19             $table->string('password');
20             $table->integer('tentatives')->default(0);
21             $table->timestamps();
22         });
23     }
24
25     /**
26      * Reverse the migrations.
27      *
28      * @return void
29      */
30     public function down()
31     {
32         Schema::dropIfExists('users');
33     }
34 }
```

Factory / seeders :

```
database > factories > UserFactory.php
1/
18
19     /**
20      * Define the model's default state.
21      *
22      * @return array
23      */
24     public function definition()
25     {
26         return [
27             'email' => "admin@gmail.com",
28             'password' => '$2y$10$92IXUNpkj00r0Q5byMi.Ye4oKoEa3Ro9llC/.og/at2.uhelWG/ig1', // password
29         ];
30     }
31 }
```

```
database > seeders > DatabaseSeeder.php
1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Seeder;
6
7  class DatabaseSeeder extends Seeder
8  {
9      /**
10       * Seed the application's database.
11       *
12       * @return void
13       */
14     public function run()
15     {
16         \App\Models\User::factory(1)->create();
17     }
18 }
```

Model user :

```
User.php X
app > Models > User.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Contracts\Auth\MustVerifyEmail;
6  use Illuminate\Database\Eloquent\Factories\HasFactory;
7  use Illuminate\Foundation\Auth\User as Authenticatable;
8  use Illuminate\Notifications\Notifiable;
9  use Laravel\Passport\HasApiTokens;
10
11 class User extends Authenticatable
12 {
13     use HasApiTokens, HasFactory, Notifiable;
14
15     /**
16      * The attributes that are mass assignable.
17      *
18      * @var array
19      */
20     protected $fillable = [
21         'name',
22         'email',
23         'password',
24         'tentatives' ←
25     ];
26 }
```

AuthController :

```
namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\Log;

class AuthController extends Controller
{
    /**
     * Handle user connection
     * @param \Illuminate\Http\Request $request
     */
    public function postLogin(Request $request){
        $validator = Validator::make(
            $request->all(),
            [
                'email' => 'required',
                'password' => 'required',
            ],
            [
                'required' => 'Le champ :attribute est requis',
            ]
        );
    }
}
```

```

    );

    $errors = $validator->errors();
    if (count($errors) != 0) {
        return response()->json([
            'success' => false,
            'message' => $errors->first()
        ]);
    }

    $user = User::where('email', $request->email)->first();
    if(!$user || !Hash::check($request->password, $user->password)){
        $user->tentatives = $user->tentatives + 1;
        $user->save();

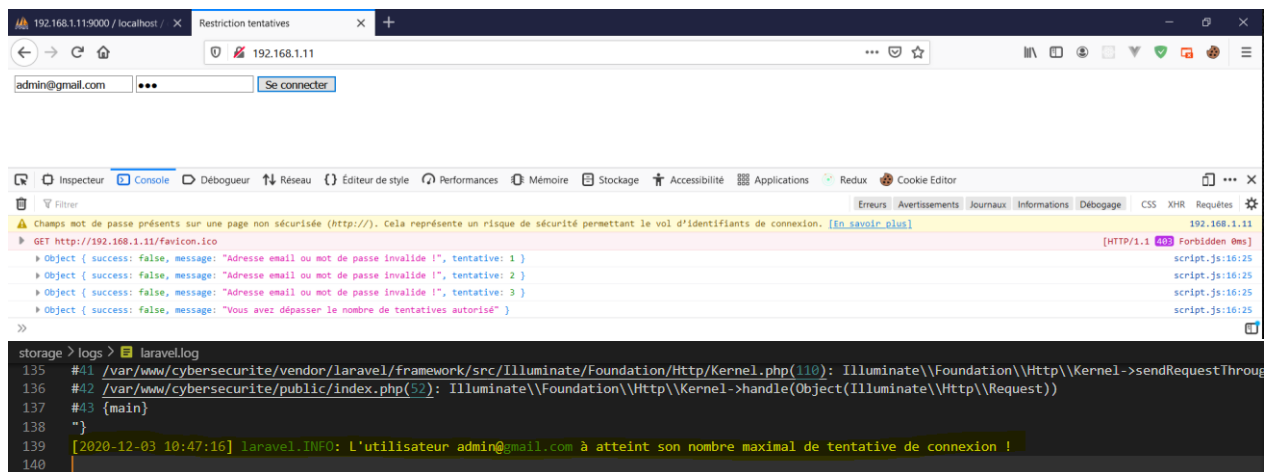
        if($user->tentatives > 3) {
            $user->tentatives = 3;
            $user->save();
            Log::channel('abuse')->info("L'utilisateur {$user->email} à atteint son nombre maximal de tentative de connexion ! ");
            return response()->json([
                'success' => false,
                'message' => "Vous avez dépasser le nombre de tentatives autorisé",
            ]);
        }
        return response()->json([
            'success' => false,
            'message' => "Adresse email ou mot de passe invalide !",
            'tentative' => $user->tentatives
        ]);
    }

    $token = $user->createToken('Auth token')->accessToken;
    return response()->json([
        'success' => true,
        'token' => $token
    ]);
}
}

```

```
User.php x AuthController.php api.php x
routes > api.php
1  <?php
2
3  use Illuminate\Http\Request;
4  use Illuminate\Support\Facades\Route;
5  use App\Http\Controllers\AuthController;
6
7
8  /*
9  |-----
10 | API Routes
11 |-----
12 |
13 | Here is where you can register API routes for your application. These
14 | routes are loaded by the RouteServiceProvider within a group which
15 | is assigned the "api" middleware group. Enjoy building your API!
16 |
17 */
18
19 Route::post('/login', [AuthController::class, 'postLogin']);
```

Résultat (inscription dans le fichier global) :



The screenshot shows a web browser window with the address bar at 192.168.1.11:5000. The page title is "Restriction tentatives". The login form contains the email "admin@gmail.com" and a password field with three dots. A "Se connecter" button is visible. The browser's developer console shows several error messages: "Champs mot de passe présents sur une page non sécurisée (http://). Cela représente un risque de sécurité permettant le vol d'identifiants de connexion." and "GET http://192.168.1.11/favicon.ico [HTTP/1.1 403 Forbidden 0ms]". Below these, there are four log entries for failed login attempts with messages like "Adresse email ou mot de passe invalide !" and "Vous avez dépassé le nombre de tentatives autorisé". At the bottom, the Laravel log shows a message: "[2020-12-03 10:47:16] laravel.INFO: L'utilisateur admin@gmail.com à atteint son nombre maximal de tentative de connexion !".

```
storage > logs > laravel.log
135 #41 /var/www/cybersecurite/vendor/laravel/framework/src/Illuminate/Foundation/Http/Kernel.php(110): Illuminate\Foundation\Http\Kernel->sendRequestThrough
136 #42 /var/www/cybersecurite/public/index.php(52): Illuminate\Foundation\Http\Kernel->handle(Object(Illuminate\Http\Request))
137 #43 {main}
138 ""
139 [2020-12-03 10:47:16] laravel.INFO: L'utilisateur admin@gmail.com à atteint son nombre maximal de tentative de connexion !
140
```

## Pour un fichier de log personnalisé

Résultat dans un fichier personnalisé :

The image displays a web application interface and its underlying code and logs. The top section shows the `logging.php` configuration file, which sets up logging for different levels of messages. The middle section shows the `AuthController.php` file, which handles login attempts. A red arrow points to a log statement in the controller that logs the user email and the number of failed attempts. The bottom section shows the browser's developer console with a warning about an insecure connection and a list of failed login attempts. Finally, the bottom-most section shows the `abuse.log` file, which contains the log message generated by the application.

```
config > logging.php
87
88 ],
89
90 'errorlog' => [
91     'driver' => 'errorlog',
92     'level' => env('LOG_LEVEL', 'debug'),
93 ],
94
95 'null' => [
96     'driver' => 'monolog',
97     'handler' => NullHandler::class,
98 ],
99
100 'emergency' => [
101     'path' => storage_path('logs/laravel.log'),
102 ],
103
104 'abuse' => [
105     'driver' => 'single',
106     'path' => storage_path('logs/abuse.log'),
107     'level' => 'debug'
108 ],
109
110 ];
```

```
app > Http > Controllers > AuthController.php
40 $user->save();
41
42 if($user->tentatives > 3) {
43     $user->tentatives = 3;
44     $user->save();
45     log::channel('abuse')->info("L'utilisateur { $user->email } à atteint son nombre maximal de tentative de connexion ! ");
46     return response()->json([
47         'success' => false,
48         'message' => "Vous avez dépasser le nombre de tentatives autorisé",
49     ]);
50 }
51 return response()->json([
52     'success' => false,
53     'message' => "Adresse email ou mot de passe invalide !",
54     'tentative' => $user->tentatives
55 ]);
56 }
57
58 $token = $user->createToken('Auth token')->accessToken;
59 return response()->json([
60     'success' => true,
61     'token' => $token
62 ]);
63 }
64 }
65 }
```

admin@gmail.com  Se connecter

192.168.1.11

Inspecteur Console Débugueur Réseau Éditeur de style Performances Mémoire Stockage Accessibilité Applications Redux Cookie Editor

Champs mot de passe présents sur une page non sécurisée (http://). Cela représente un risque de sécurité permettant le vol d'identifiants de connexion. [En savoir plus](#)

GET http://192.168.1.11/favicon.ico 192.168.1.11 [HTTP/1.1 403 Forbidden 0ms]

Object { success: false, message: "Adresse email ou mot de passe invalide !", tentative: 1 } script.js:16:25

Object { success: false, message: "Adresse email ou mot de passe invalide !", tentative: 2 } script.js:16:25

Object { success: false, message: "Adresse email ou mot de passe invalide !", tentative: 3 } script.js:16:25

Object { success: false, message: "Vous avez dépasser le nombre de tentatives autorisé" } script.js:16:25

```
storage > logs > abuse.log
1 [2020-12-03 11:28:42] local.INFO: L'utilisateur admin@gmail.com à atteint son nombre maximal de tentative de connexion !
2
```

## Ajustement pour un blocage temporaire

Tâche cron pour laravel :

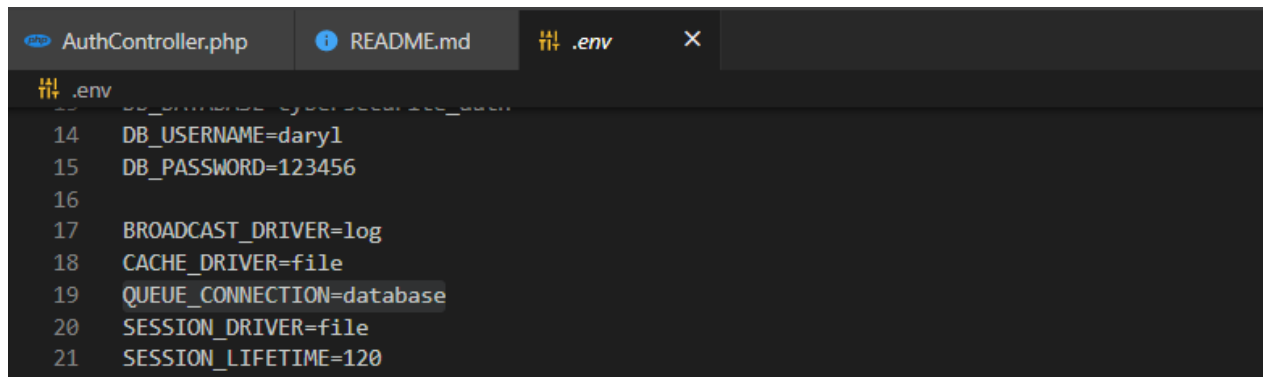
<https://laravel.com/docs/8.x/scheduling>

<https://appdividend.com/2017/12/21/laravel-queues-tutorial-example-scratch/>

php artisan queue:table

php artisan migrate

php artisan make:job ResetTentatives

A screenshot of a code editor with a dark theme. The top bar shows three tabs: 'AuthController.php', 'README.md', and '.env'. The '.env' tab is active, showing a list of environment variables. Line 14 is 'DB\_USERNAME=daryl', line 15 is 'DB\_PASSWORD=123456', line 17 is 'BROADCAST\_DRIVER=log', line 18 is 'CACHE\_DRIVER=file', line 19 is 'QUEUE\_CONNECTION=database', line 20 is 'SESSION\_DRIVER=file', and line 21 is 'SESSION\_LIFETIME=120'.

```
.env
14 DB_USERNAME=daryl
15 DB_PASSWORD=123456
16
17 BROADCAST_DRIVER=log
18 CACHE_DRIVER=file
19 QUEUE_CONNECTION=database
20 SESSION_DRIVER=file
21 SESSION_LIFETIME=120
```

Fichier : App\Jobs\ResetTentatives

```
<?php
namespace App\Jobs;

use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldBeUnique;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Foundation\Bus\Dispatchable;
use Illuminate\Queue\InteractsWithQueue;
use Illuminate\Queue\SerializesModels;
use App\Models\User;

class resetTentatives implements ShouldQueue{
    use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;
    public $userId;

    /**
     * Create a new job instance.
     * @return void
     */
    public function __construct($userId)
    {
        $this->userId = $userId;
    }
}
```

```

    }

    /**
     * Execute the job.
     * @return void
     */
    public function handle(){
        $user = User::whereId($this->userId)->first();
        $user->tentatives = 0;
        $user->save();
    }
}

```

AuthController.php X

app > Http > Controllers > AuthController.php

```

1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\User;
6  use Illuminate\Http\Request;
7  use Illuminate\Support\Facades\Hash;
8  use Illuminate\Support\Facades\Validator;
9  use Illuminate\Support\Facades\Log;
10 use App\Jobs\ResetTentatives;
11 use Carbon\Carbon;
12

```

AuthController.php X

app > Http > Controllers > AuthController.php

```

40 $user = User::where('email', $request->email)->first();
41 if(!$user || !Hash::check($request->password, $user->password)){
42
43     if($user) {
44         $user->tentatives = $user->tentatives + 1;
45         $user->save();
46         if($user->tentatives > 3) {
47             $user->tentatives = 3;
48             $user->save();
49
50             $resetJob = (new ResetTentatives($user->id))->delay(Carbon::now()->addSeconds(30));
51             dispatch($resetJob);
52
53             Log::channel('abuse')->info("L'utilisateur { $user->email } à atteint son nombre maximal de tentative de connexion ! ");
54             return response()->json([
55                 'success' => false,
56                 'type' => 'info',
57                 'message' => "Veuillez réessayer dans 30 secondes",
58             ]);
59         }
60     }
61
62     return response()->json([
63         'success' => false,
64         'type' => 'danger',
65         'message' => "Adresse email ou mot de passe invalide !",
66     ]);

```

TERMINAL

CONSOLE DE DÉBOGAGE

PROBLÈMES

SORTIE

1: -bash, php

pour régler l'identité par défaut de votre compte.  
Éliminez --global pour ne faire les réglages que dans ce dépôt.

fatal: impossible de détecter automatiquement l'adresse ('secu@debian.(none)') trouvé

```

secu@debian:/var/www/cybersecurite$ su -
Mot de passe :
root@debian:~# cd /var/www/cybersecurite/
root@debian:/var/www/cybersecurite# ls
app      composer.json  database      public        routes        tests
artisan  composer.lock  package.json  README.md    server.php    vendor
bootstrap config          phpunit.xml  resources     storage       webpack.mix.js
root@debian:/var/www/cybersecurite# git add .
root@debian:/var/www/cybersecurite# git commit -m "Jobs pour le reset tentative"
[main 27b25f5] Jobs pour le reset tentative
4 files changed, 85 insertions(+), 4 deletions(-)
create mode 180644 app/Jobs/ResetTentatives.php
create mode 180644 database/migrations/2020_12_04_095108_create_jobs_table.php
root@debian:/var/www/cybersecurite#

```

```

secu@debian:/var/www/cybersecurite$ php artisan queue:work
^C
secu@debian:/var/www/cybersecurite$ php artisan queue:work
^C
secu@debian:/var/www/cybersecurite$ php artisan queue:work
^C
secu@debian:/var/www/cybersecurite$ php artisan queue:work
^C
secu@debian:/var/www/cybersecurite$ php artisan queue:work
[2020-12-04 10:25:34][1] Processing: App\Jobs\ResetTentatives
[2020-12-04 10:25:35][1] Processed: App\Jobs\ResetTentatives
[2020-12-04 10:25:38][2] Processing: App\Jobs\ResetTentatives
[2020-12-04 10:25:38][2] Processed: App\Jobs\ResetTentatives
[2020-12-04 10:25:38][3] Processing: App\Jobs\ResetTentatives
[2020-12-04 10:25:38][3] Processed: App\Jobs\ResetTentatives
[2020-12-04 10:25:38][4] Processing: App\Jobs\ResetTentatives
[2020-12-04 10:25:38][4] Processed: App\Jobs\ResetTentatives

```



## Imposer le mot de passe fort

Mettre en place une jauge pour forcer un mot de passe fort

Exemple :

Lorsque vous vous inscrivez sur un site, vous avez une jauge qui regarde si votre mot de passe est fort ou pas. Il faut que votre mot de passe soit obligatoirement fort (moins facile à casser en gros)

<https://github.com/dropbox/zxcvbn>

<https://regexr.com/3bfsi>

<https://www.thepolyglotdeveloper.com/2015/05/use-regex-to-test-password-strength-in-javascript/>

Moving beyond the basic AngularJS stuff, lets get down to the good stuff. We're going to make use of two different RegEx validation strings. One string will represent what is required for creating a strong password and the other will represent what is necessary for creating a medium strength password. If neither expressions are satisfied, we'll assume it is poor strength.

```
var strongRegex = new RegExp("^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#$%^&*])(?={8,})");
```

Above you can see the expression for validating a strong password. Let's break down what it is doing.

RegEx	Description
^	The password string will start this way
(?=.*[a-z])	The string must contain at least 1 lowercase alphabetical character
(?=.*[A-Z])	The string must contain at least 1 uppercase alphabetical character
(?=.*[0-9])	The string must contain at least 1 numeric character
(?=.*[!@#\$%^&*])	The string must contain at least one special character, but we are escaping reserved RegEx characters to avoid conflict
(?={8,})	The string must be eight characters or longer

That wasn't so bad! So how about our medium strength validator? There is less precision in this one, thus making it necessary to create a more complex regular expression.

```
var mediumRegex = new RegExp("^(?((?=.*[a-z])(?=.*[A-Z]))|(?((?=.*[a-z])(?=.*[0-9]))|(?((?=.*[A-Z])(?=.*[0-9]))< >
```

The expression is nearly the same as the strong condition, except this time we're including an or condition. We essentially want to label a password as having medium strength if it contains six characters or more and has at least one lowercase and one uppercase alphabetical character or has at least one lowercase and one numeric character or has at least one uppercase and one numeric character. We've chosen to leave special characters out of this one.

Finally we need a way to track input on the input field. This will be handled through AngularJS by using the `ngChange` directive. By placing `ng-change="analyze(password)"` in the input tag it will call the `analyze(value)` method every time a key-stroke happens. That's when we call our RegEx validators.

```
var strongRegex =  
new RegExp("((?=.*[a-z]{2,})(?=.*[A-Z]{2,})(?=.*[0-9]{2,})(?=.*[!@#\\$%^&\\ *]))(?.{8,})");
```

Il faut avoir :

Au moins 2 minuscules consecutif

Au moins 2 majuscules consecutif

Au moins 2 chiffres consecutif

Au moins 1 caractère spécial

La totalité de la chaine de caractères doit faire 8 caractères ou plus

```
var mediumRegex =  
new RegExp("^(?=.*[a-z])(?=.*[A-Z])|((?=.*[a-z])(?=.*[0-9]))|((?=.*[A-Z])(?=.*[0-9]))(?.{6,})");
```

Il faut avoir une des conditions suivantes :

Condition 1 : contenir 1 caractère minuscule et 1 majuscule

Condition 2 : contenir 1 caractère minuscule et 1 chiffre

Condition 3 : contenir 1 caractère majuscule et 1 chiffre

La totalité de la chaine de caractères doit faire 6 caractères ou plus