

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 installée 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/

Ressource code :

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

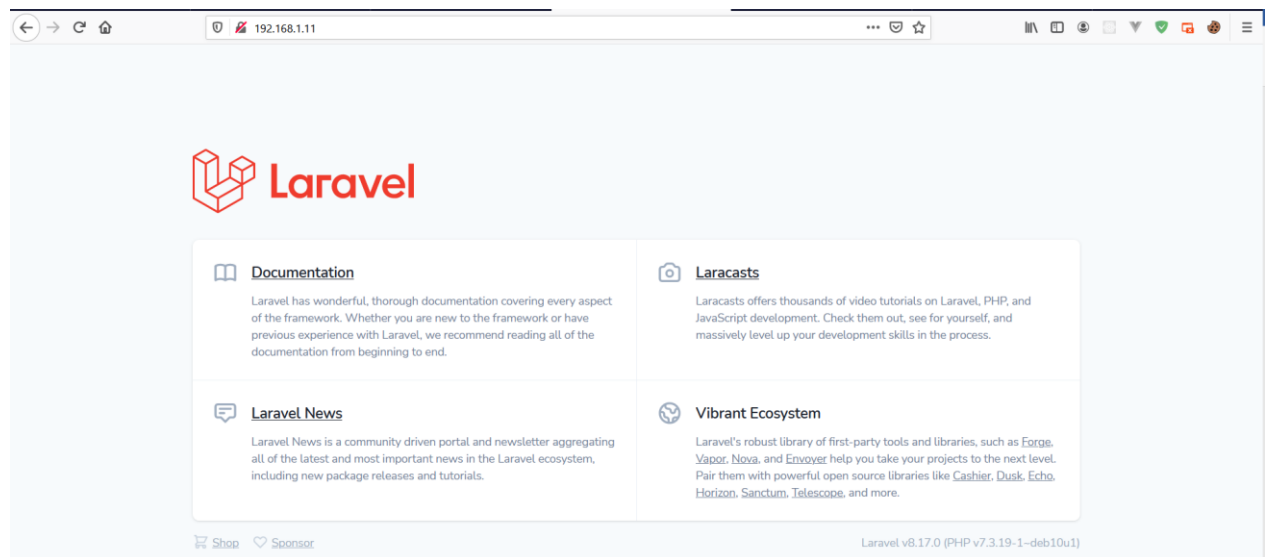
```
chgrp -R www-data /var/www/cybersecurityite  
chown -R secu /var/www/cybersecurityite  
chmod -R 775 /var/www/cybersecurityite/storage
```

```
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
```

```

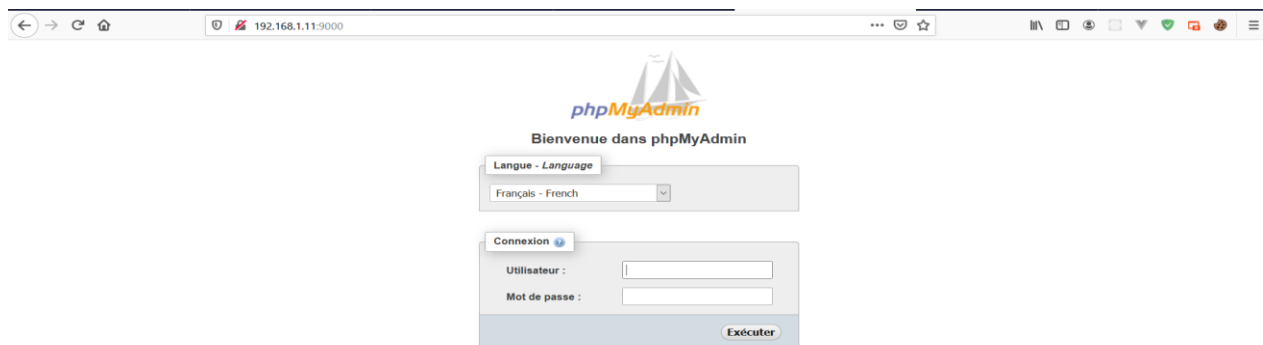
secu@debian:~$ su -
Mot de passe :
root@debian:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 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/



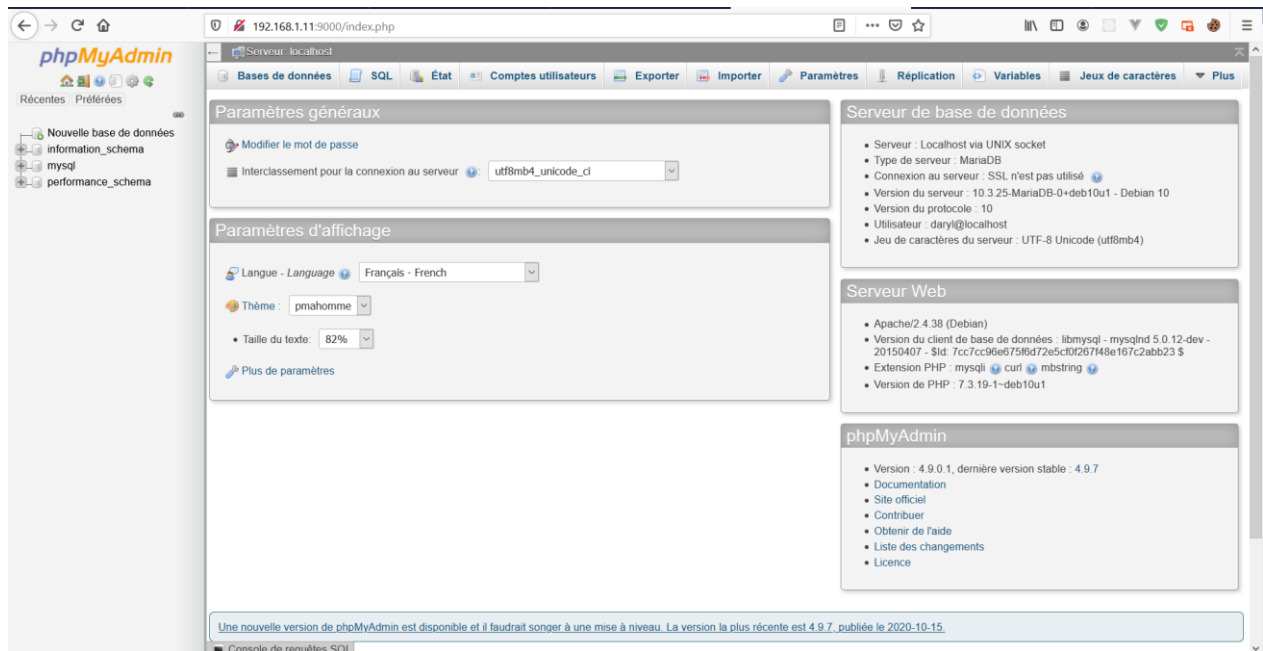
Je crée un utilisateur pour pouvoir me connecter à l'interface :

```

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 :

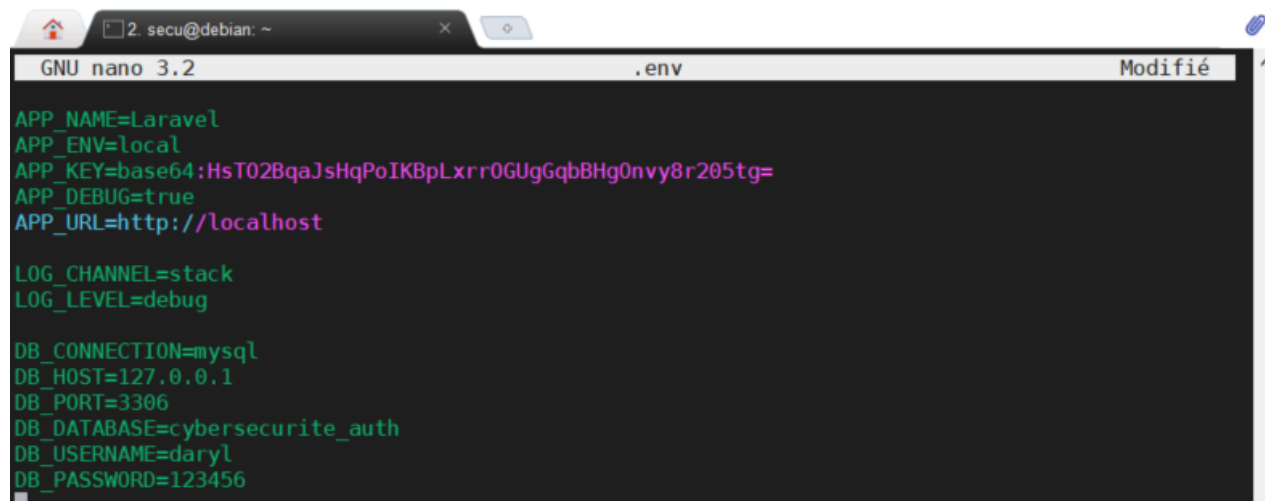


- Création d'une bdd - Configuration projet pour la connexion bdd

Création de la BDD :



Configuration du .env laravel :



The image shows a terminal window with a dark background. At the top, the window title bar indicates the user is '2. secu@debian' in the home directory. The terminal itself shows the 'GNU nano 3.2' editor interface editing a file named '.env'. The status bar at the top right of the editor says 'Modifié'. The content of the file is as follows:

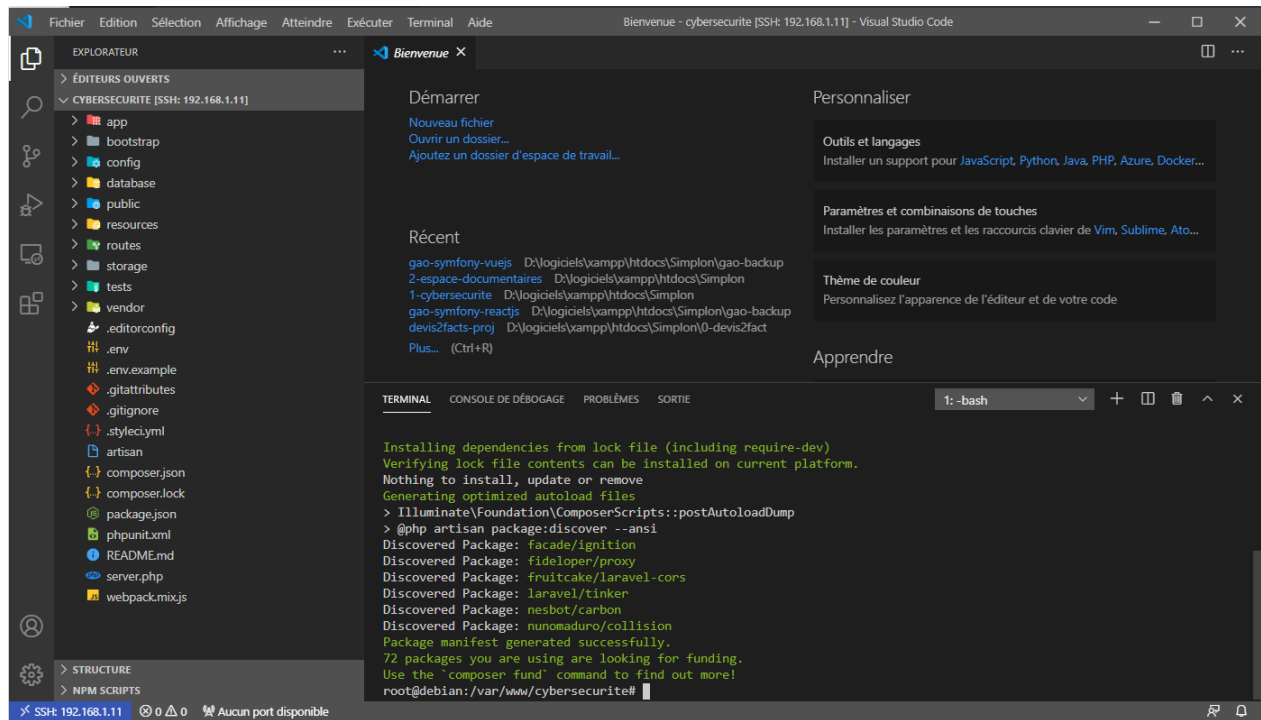
```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:HsT02BqaJsHqPoIKBpLxrr0GUgGqbBHg0nvy8r205tg=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_LEVEL=debug

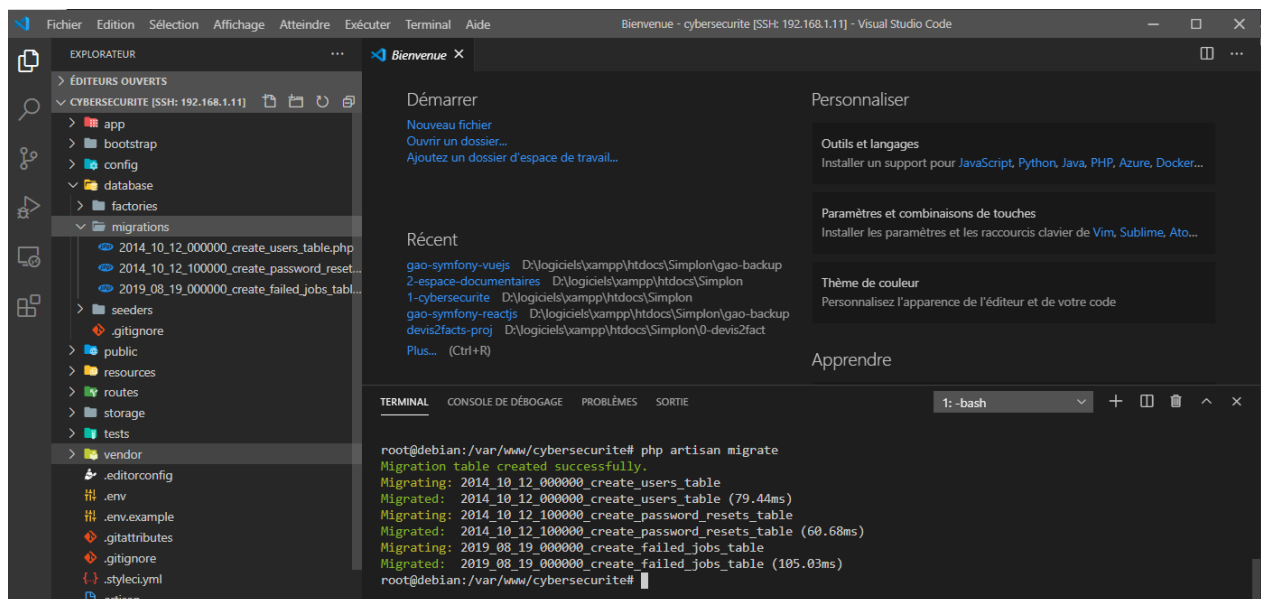
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=cybersecurite_auth
DB_USERNAME=daryl
DB_PASSWORD=123456
```

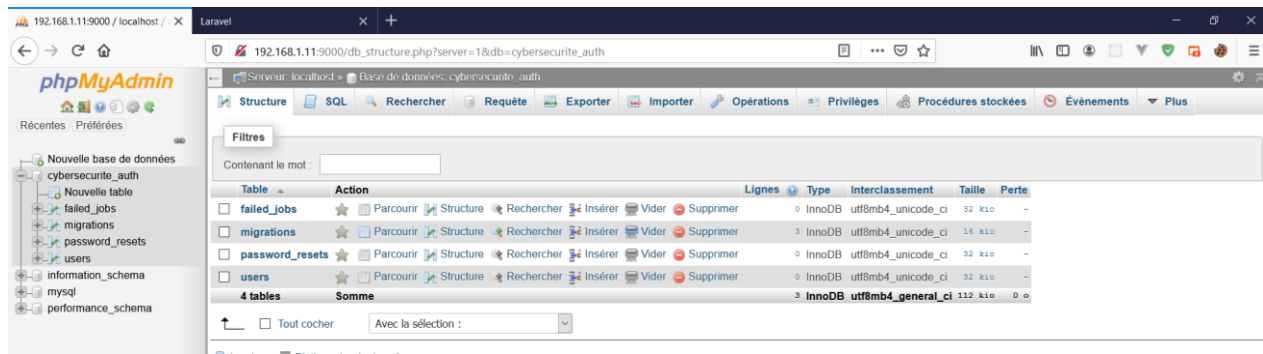
Initialisation des fichiers config

Composer install

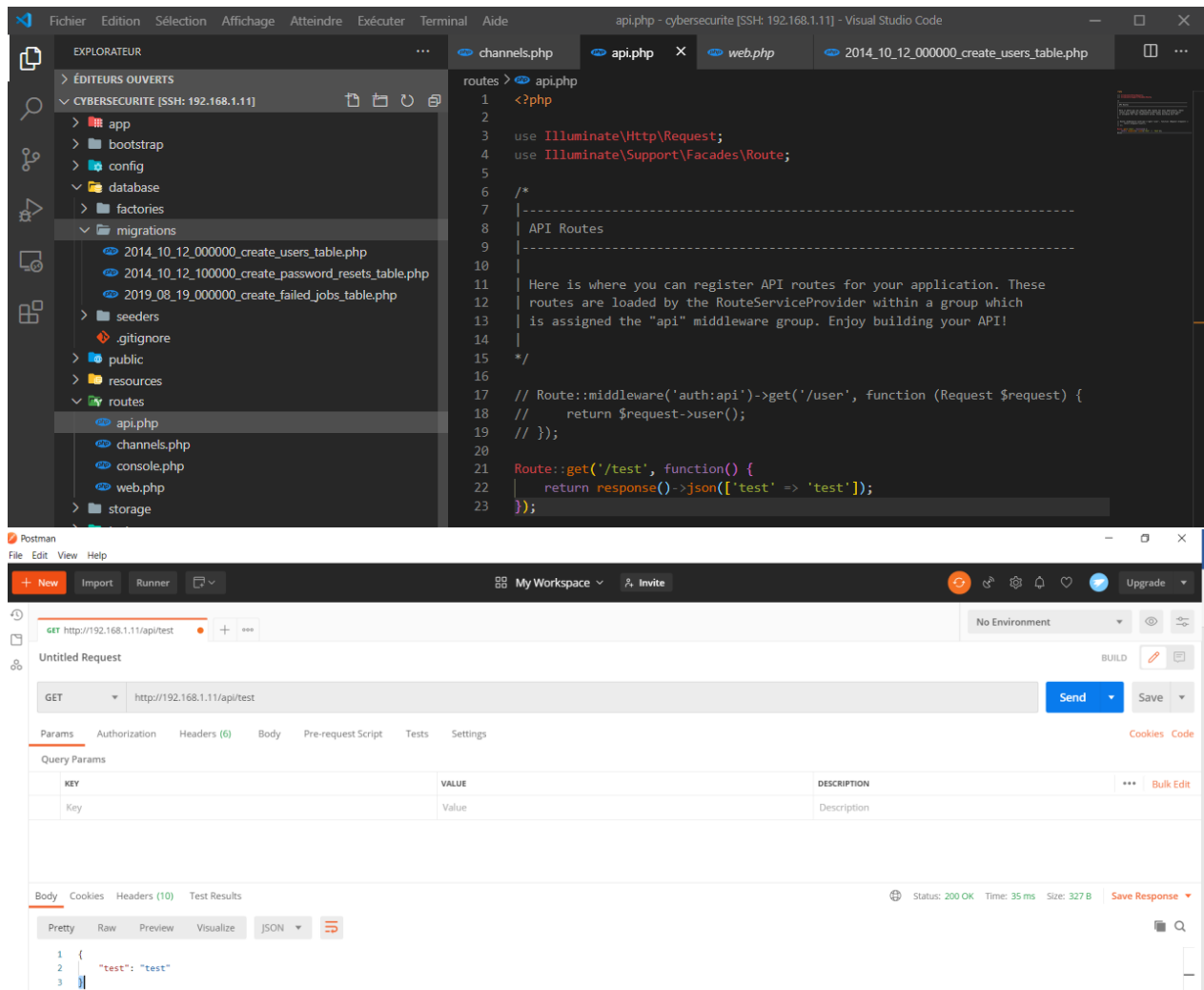


Test d'une migration pour vérifier la connexion à la BDD :





Test configuration API :

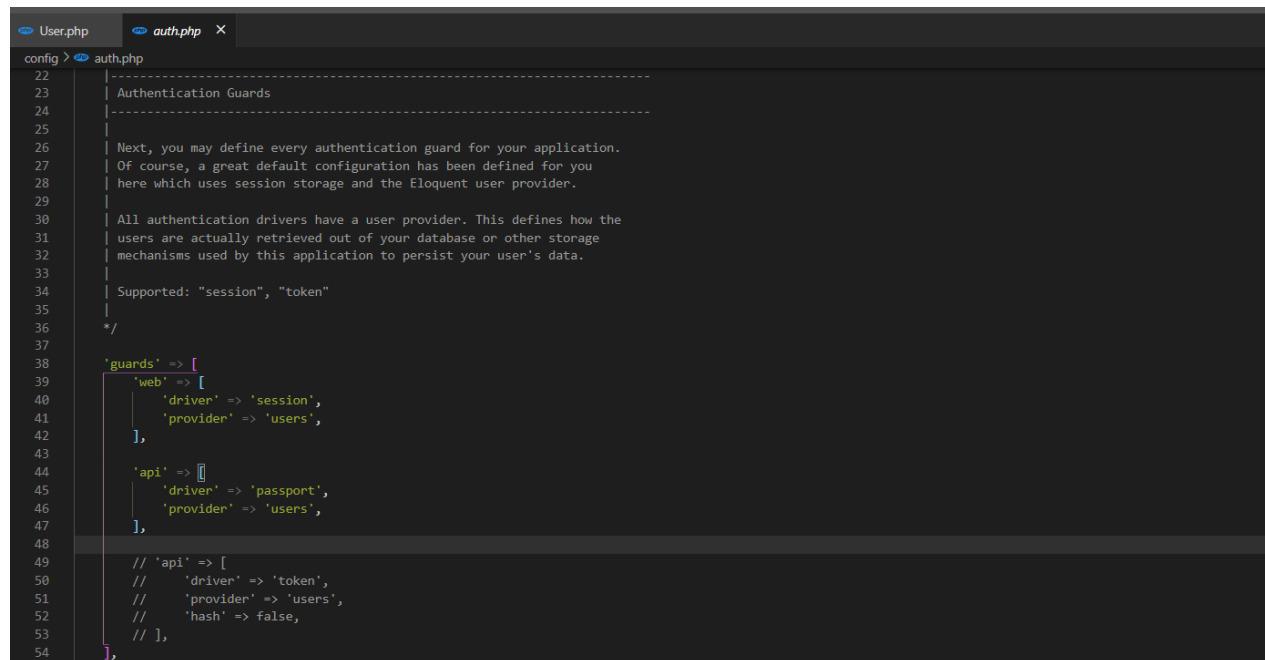


Configuration JWT par le biais de passport

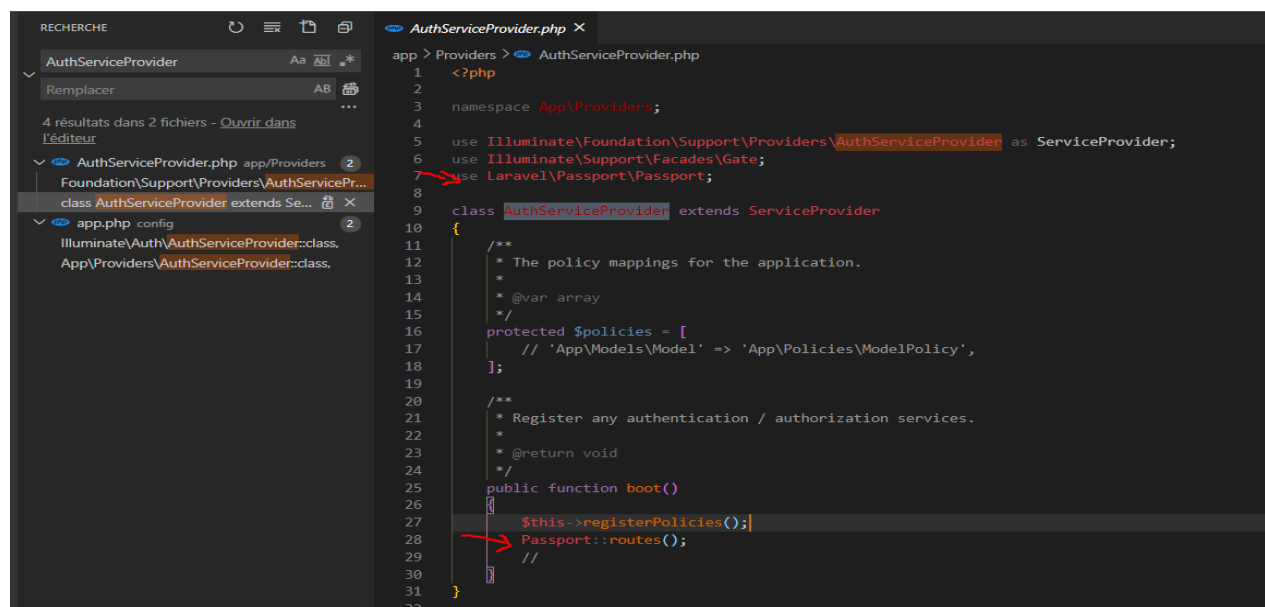
<https://laravel.com/docs/8.x/passport>

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

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



```
config > auth.php
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 |     'api' => [
44 |         'driver' => 'passport',
45 |         'provider' => 'users',
46 |     ],
47 | ],
48 |
49 | // 'api' => [
50 | //     'driver' => 'token',
51 | //     'provider' => 'users',
52 | //     'hash' => false,
53 | // ],
54 | ],
```



```
RECHERCHE
AuthServiceProvider
Remplacer
4 résultats dans 2 fichiers - Ouvrir dans l'éditeur
AuthServiceProvider.php app/Providers
Foundation\Support\Providers\AuthServicePr...
class AuthServiceProvider extends Se...
app.php config
Illuminate\Auth\AuthServiceProvider::class,
App\Providers\AuthServiceProvider::class,

app > Providers > AuthServiceProvider.php
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 :

```
User.php x  AuthController.php  2014_10_12_000000_create_users_table.php x
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 :

```
User.php AuthController.php UserFactory.php X
database > factories > UserFactory.php
17
18 /**
19  * Define the model's default state.
20  *
21  * @return array
22  */
23 public function definition()
24 {
25     return [
26         'email' => "admin@gmail.com",
27         'password' => '$2y$10$92IXUNpkj00r0Q5byMi.Ye4oKoEa3Ro9llC/.og/at2.uhWG/igi', // password
28     ];
29 }
30
31
DatabaseSeeder.php X
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. 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 developer console shows the following messages:

- A warning: "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]"
- A GET request to "http://192.168.1.11/favicon.ico" resulting in a "Forbidden 403" error.
- Four failed login attempts with the message: "Adresse email ou mot de passe invalide !", tentative: 1, 2, 3, and 4.
- A final message: "Vous avez dépassé le nombre de tentatives autorisé"

The bottom of the image shows a terminal window with the following log output:

```
storage > logs > laravel.log
135 #41 /var/www/cybersecurite/vendor/laravel/framework/src/Illuminate/Foundation/Http/Kernel.php(110): Illuminate\Foundation\Http\Kernel->sendRequestThroughMiddleware()
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 development environment with two main components: a code editor and a web browser.

Code Editor (Top):

- logging.php:** Shows the configuration for Laravel's logging system. It defines three log channels: 'errorlog' (using the 'errorlog' driver), 'null' (using the 'monolog' driver with a 'NullHandler' class), and 'emergency' (using the 'monolog' driver with a custom path 'logs/laravel.log').
- AuthController.php:** Shows the authentication logic. It includes a check for the number of login attempts. If the number of attempts exceeds 3, it logs the event using the 'abuse' channel and returns a response indicating failure. Otherwise, it creates an authentication token and returns it.

Web Browser (Bottom):

- The browser shows a login form with the email 'admin@gmail.com' and a 'Se connecter' button.
- The console shows the following log messages:
 - Object { success: false, message: "Adresse email ou mot de passe invalide !", tentative: 1 }
 - Object { success: false, message: "Adresse email ou mot de passe invalide !", tentative: 2 }
 - Object { success: false, message: "Adresse email ou mot de passe invalide !", tentative: 3 }
 - Object { success: false, message: "Vous avez dépasser le nombre de tentatives autorisé" }

Custom Log File (Bottom):

- The file 'abuse.log' in the 'storage/logs' directory contains the following log entry:

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