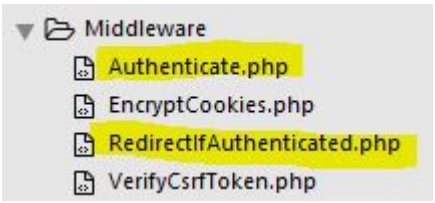
Créer une application : l'authentification

Article mis à jour le 26/10/2015

Laravel 5 arrive avec une infrastructure pour l'authentification. On trouve dans l'installation de base 2 contrôleurs :

Et deux middlewares :



Et c'est tout ! Il faut donc créer tout le reste.

Les routes

Avec Laravel 5.2 on peut obtenir toutes les routes de l'authentification avec **php artisan make:auth**, mais on peut aussi les écrire , ce que j'ai préféré faire :

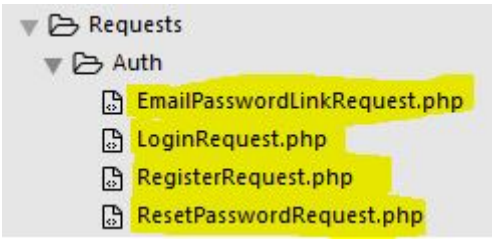
```
// Authentication routes...
Route::get('auth/login', 'Auth\AuthController@getLogin');
Route::post('auth/login', 'Auth\AuthController@postLogin');
Route::get('auth/logout', 'Auth\AuthController@getLogout');
Route::get('auth/confirm/{token}',
'Auth\AuthController@getConfirm');

// Resend routes...
Route::get('auth/resend', 'Auth\AuthController@getResend');

// Registration routes...
Route::get('auth/register', 'Auth\AuthController@getRegister');
```

```
Route::post('auth/register', 'Auth\AuthController@postRegister');

// Password reset link request routes...
Route::get('password/email', 'Auth\PasswordController@getEmail');
Route::post('password/email',
'Auth\PasswordController@postEmail');

// Password reset routes...
Route::get('password/reset/{token}',
'Auth\PasswordController@getReset');
Route::post('password/reset',
'Auth\PasswordController@postReset');
```

Notez que la route pour a confirmation de l'email est forcément à créer.

La validation

Comme j'ai dû surcharger les méthodes correspondantes des contrôleurs j'ai créé des requêtes de formulaires :



Ces requêtes ne présentent aucune difficulté ou particularité, je vous laisse donc les consulter si besoin.

Les contrôleurs

AuthController

Lorsqu'on regarde ce contrôleur dans l'installation de base on y trouve pas grand chose :

```
<?php

namespace App\Http\Controllers\Auth;
```

```
use App\User;
use Validator;
use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\ThrottlesLogins;
use Illuminate\Foundation\Auth\AuthenticatesAndRegistersUsers;

class AuthController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Registration & Login Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles the registration of new users, as well as the
    | authentication of existing users. By default, this controller uses
    | a simple trait to add these behaviors. Why don't you explore it?
    |
    */

    use AuthenticatesAndRegistersUsers, ThrottlesLogins;

    /**
     * Create a new authentication controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest', ['except' => 'getLogout']);
    }

    /**
     * Get a validator for an incoming registration request.
     *
     * @param  array  $data
     * @return \Illuminate\Contracts\Validation\Validator
     */
    protected function validator(array $data)
```

```
    {
        return Validator::make($data, [
            'name' => 'required|max:255',
            'email' => 'required|email|max:255|unique:users',
            'password' => 'required|confirmed|min:6',
        ]);
    }

    /**
     * Create a new user instance after a valid registration.
     *
     * @param  array  $data
     * @return User
     */
    protected function create(array $data)
    {
        return User::create([
            'name' => $data['name'],
            'email' => $data['email'],
            'password' => bcrypt($data['password']),
        ]);
    }
}
```

En fait juste un constructeur, un validateur et du code pour la création d'un utilisateur. L'essentiel du traitement est effectué dans le trait **AuthenticatesAndRegistersUsers**. Comme ce trait est dans le core il est évidemment déconseillé d'aller modifier son code. Il est prévu quelques fonctionnalités concernant les urls mais au delà de ces possibilités la seule solution pour adapter le code consiste à surcharger des méthodes du trait.

postLogin

J'ai choisi de surcharger cette méthode pour plusieurs raisons :

- j'ai donné la possibilité de se connecter à partir du pseudo OU de l'email
- je voulais vérifier que l'utilisateur a confirmé son adresse email
- je voulais gérer facilement la redirection et le message d'erreur

Ce qui donne finalement ce code :

```
public function postLogin(
    LoginRequest $request,
    Guard $auth)
{
    $logValue = $request->input('log');

     $logAccess = filter_var($logValue, FILTER_VALIDATE_EMAIL) ?
'email' : 'username';

    $throttles = in_array(
                                    ThrottlesLogins::class,
class_uses_recursive(get_class($this))
    );

    if ($throttles && $this->hasTooManyLoginAttempts($request)) {
        return redirect('/auth/login')
            ->with('error', trans('front/login.maxattempt'))
            ->withInput($request->only('log'));
    }

    $credentials = [
        $logAccess  => $logValue,
        'password'  => $request->input('password')
    ];

    if(!$auth->validate($credentials)) {
        if ($throttles) {
            $this->incrementLoginAttempts($request);
        }

        return redirect('/auth/login')
            ->with('error', trans('front/login.credentials'))
            ->withInput($request->only('log'));
    }

    $user = $auth->getLastAttempted();

    if($user->confirmed) {
        if ($throttles) {
            $this->clearLoginAttempts($request);
        }
```

```
        $auth->login($user, $request->has('memory'));

        if($request->session()->has('user_id'))    {
            $request->session()->forget('user_id');
        }

        return redirect('/');
    }

    $request->session()->put('user_id', $user->id);

          return  redirect('/auth/login')->with('error',
trans('front/verify.again'));
}
```

La principale particularité réside dans la double possibilité de connexion. Pour distinguer les deux je me suis fondé sur le type d'information.

postRegister

J'ai surchargé cette méthode pour utiliser une requête de formulaire, créer un code de confirmation pour l'adresse email, flasher un message pour l'utilisateur :

```
public function postRegister(
    RegisterRequest $request,
    UserRepository $user_gestion)
{
    $user = $user_gestion->store(
        $request->all(),
        $confirmation_code = str_random(30)
    );

    $this->dispatch(new SendMail($user));

                    return     redirect('/')->with('ok',
trans('front/verify.message'));
}
```

PasswordController

Ce contrôleur est très maigre dans l’installation de base :

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\ResetsPasswords;

class PasswordController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Password Reset Controller
    |--------------------------------------------------------------------------
    |
    | This controller is responsible for handling password reset requests
    | and uses a simple trait to include this behavior. You're free to
    | explore this trait and override any methods you wish to tweak.
    |
    */

    use ResetsPasswords;

    /**
     * Create a new password controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest');
    }
}
```

L’essentiel du code est également dans un trait : **ResetsPasswords**.

Le raisonnement est encore le même qu’avec le précédent contrôleur.

Pour ce contrôleur j’ai donc surchargé les principales méthodes.

postEmail

Là je devais gérer le contenu de l’email selon la langue, pour y parvenir je suis passé par un composeur de vues. D’autre part j’ai utilisé une requête de formulaire pour la validation :

```
public function postEmail(
    EmailPasswordLinkRequest $request,
    Factory $view)
{
    $view->composer('emails.auth.password', function($view) {
        $view->with([
            'title'   => trans('front/password.email-title'),
            'intro'   => trans('front/password.email-intro'),
            'link'    => trans('front/password.email-link'),
            'expire'  => trans('front/password.email-expire'),
            'minutes' => trans('front/password.minutes'),
        ]);
    });

     $response = Password::sendResetLink($request->only('email'),
function (Message $message) {
        $message->subject(trans('front/password.reset'));
    });

    switch ($response) {
        case Password::RESET_LINK_SENT:
                return redirect()->back()->with('status',
trans($response));

        case Password::INVALID_USER:
                return redirect()->back()->with('error',
trans($response));
    }
}
```

postReset

Ici ma seule justification est d’utiliser une requête de formulaire.

```
public function postReset(ResetPasswordRequest $request)
{
    $credentials = $request->only(
        'email', 'password', 'password_confirmation', 'token'
    );

     $response = Password::reset($credentials, function($user, $password) {
        $this->resetPassword($user, $password);
    });

    switch ($response) {
        case Password::PASSWORD_RESET:
                    return redirect()->to('/')->with('ok', trans('passwords.reset'));

        default:
            return redirect()->back()
                    ->with('error', trans($response))
                    ->withInput($request->only('email'));
    }
}
```

Les vues

C’est au niveau des vues qu’à porté le plus gros des adaptation pour obtenir un visuel bien intégré au site.

Le login

Voici le visuel :

On trouve un champ de saisie pour le pseudo ou l'email (au choix) et un autre pour le mot de passe. Il est prévu aussi une case à cocher pour la création d'un cookie pour mémoriser la connexion. D'autre part un lien est prévu pour l'oubli du mot de passe. Dans la partie inférieure on propose une inscription.

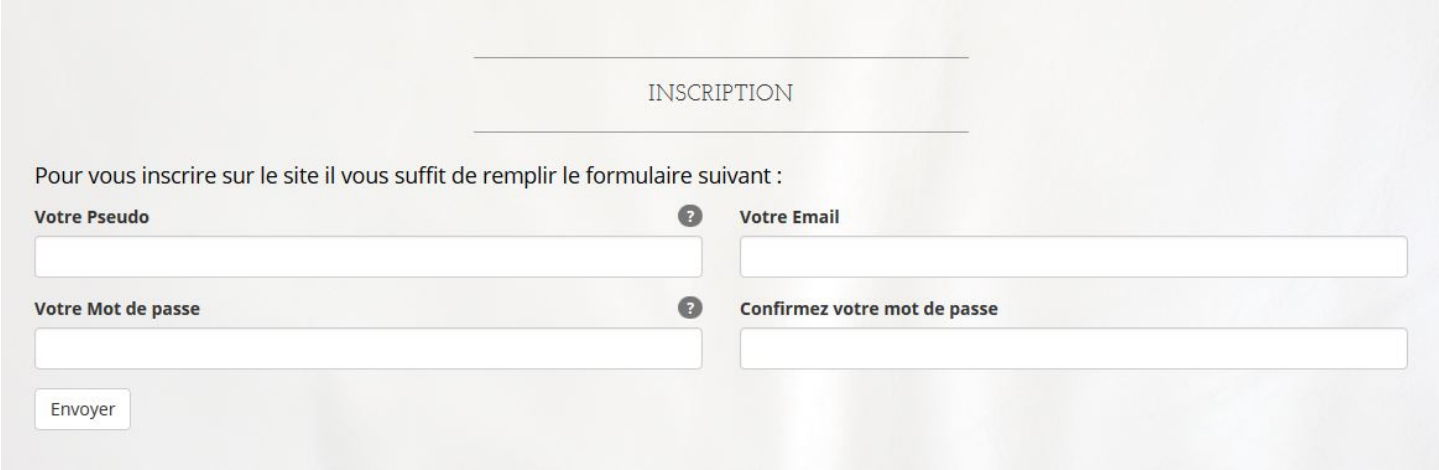
Le code de la vue (**login.blade.php**) est optimisé grâce à des méthodes ajoutées au **FormBuilder** (je détaillerai cet aspect dans un article ultérieur) :

```
@extends('front.template')

@section('main')
    <div class="row">
        <div class="box">
            <div class="col-lg-12">
                @if(session()->has('error'))
                        @include('partials/error', ['type' =>
'danger', 'message' => session('error')])
                @endif
                <hr>
                    <h2 class="intro-text text-center">{{
trans('front/login.connection') }}</h2>
                <hr>
                 <p>{{ trans('front/login.text') }}</p>


                {!! Form::open(['url' => 'auth/login', 'method' =>
```

```
'post', 'role' => 'form']) !!}

                <div class="row">

                    {!! Form::control('text', 6, 'log', $errors,
trans('front/login.log')) !!}
                    {!! Form::control('password', 6, 'password',
$errors, trans('front/login.password')) !!}
                    {!! Form::submit(trans('front/form.send'),
['col-lg-12']) !!}
                                {!!  Form::check('memory',
trans('front/login.remind')) !!}
                    {!! Form::text('address', '', ['class' =>
'hpet']) !!}
                <div class="col-lg-12">
                            {!! link_to('password/email',
trans('front/login.forget')) !!}
                </div>

                </div>

                {!! Form::close() !!}

                <div class="text-center">
                    <hr>
                         <h2 class="intro-text text-center">{{
trans('front/login.register') }}</h2>
                    <hr>
                      <p>{{ trans('front/login.register-info')
}}</p>
                              {!! link_to('auth/register',
trans('front/login.registering'), ['class' => 'btn btn-default'])
!!}
                </div>

            </div>
        </div>
    </div>
@stop
```

L’inscription

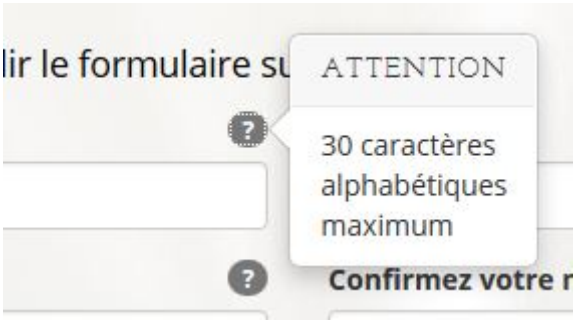
Voici le visuel :



Les champs de saisie sont prévus pour :

- le pseudo
- l’email
- le mot de passe
- la confirmation du mot de passe

J’ai aussi prévu des bulles d’information pour faciliter la saisie :



Ici aussi le code du formulaire est simplifié avec les méthodes ajoutées (**register.blade.php**) :

```
@extends('front.template')

@section('main')
    <div class="row">
        <div class="box">
            <div class="col-lg-12">
                <hr>
```

```
                    <h2 class="intro-text text-center">{{
trans('front/register.title') }}</h2>
                <hr>
                <p>{{ trans('front/register.infos') }}</p>

                {!! Form::open(['url' => 'auth/register', 'method'
=> 'post', 'role' => 'form']) !!}

                    <div class="row">
                        {!! Form::control('text', 6, 'username',
$errors,     trans('front/register.pseudo'),     null,
[trans('front/register.warning'),  trans('front/register.warning-
name')]) !!}
                         {!! Form::control('email', 6, 'email',
$errors, trans('front/register.email')) !!}
                    </div>
                    <div class="row">
                           {!! Form::control('password', 6,
'password',  $errors,  trans('front/register.password'),  null,
[trans('front/register.warning'),  trans('front/register.warning-
password')]) !!}
                           {!! Form::control('password', 6,
'password_confirmation', $errors, trans('front/register.confirm-
password')) !!}
                    </div>
                      {!! Form::text('address', '', ['class' =>
'hpet']) !!}

                    <div class="row">
                       {!! Form::submit(trans('front/form.send'),
['col-lg-12']) !!}
                    </div>

                {!! Form::close() !!}

            </div>
        </div>
    </div>
@stop

@section('scripts')

    <script>
```

```
        $(function() { $('.badge').popover();    });
    </script>

@stop
```

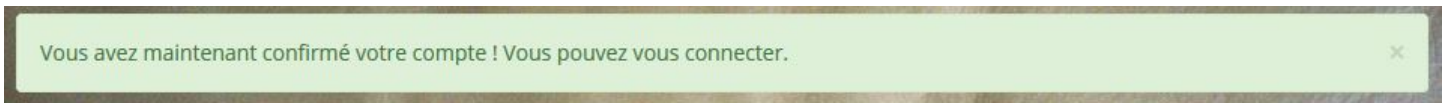
Lorsque quelqu’un s’inscrit il reçoit ce message :

En effet on lui a envoyé un email pour la confirmation de l’adresse :

Et quand on clique l’action est gérée par cette méthode du contrôleur :

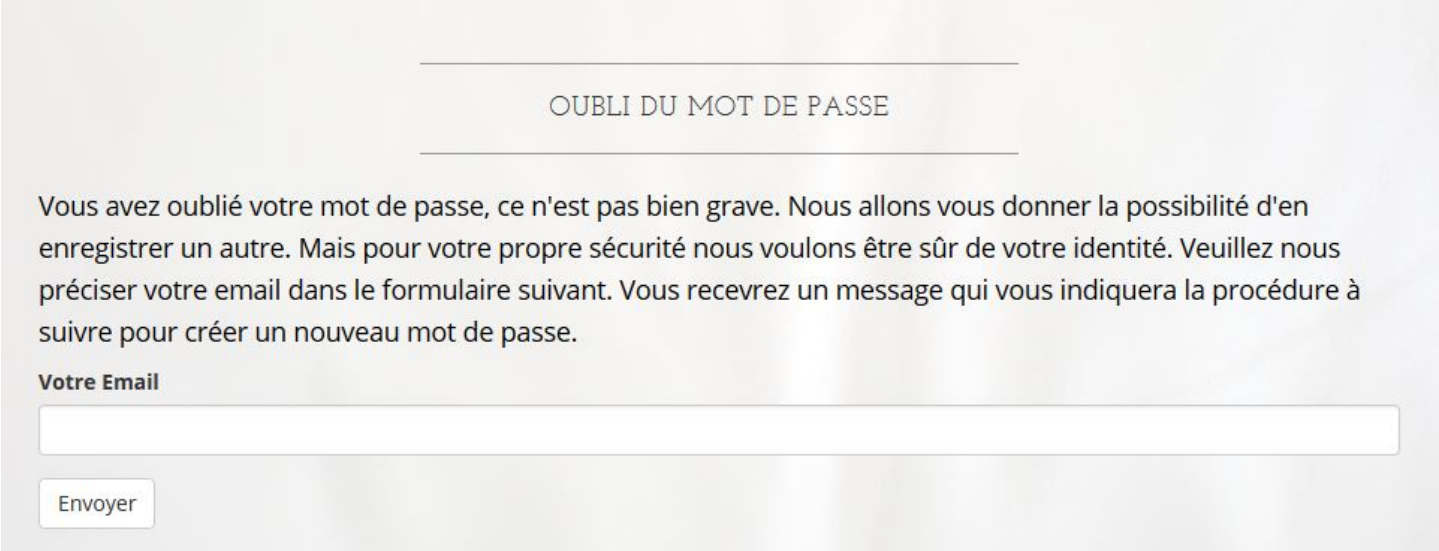
```
public function getConfirm(
    UserRepository $user_gestion,
    $confirmation_code)
{
    $user = $user_gestion->confirm($confirmation_code);

                    return     redirect('/')->with('ok',
trans('front/verify.success'));
}
```

Si la confirmation est bonne on obtient ce message :



L’oubli du mot de passe

Voici le visuel :



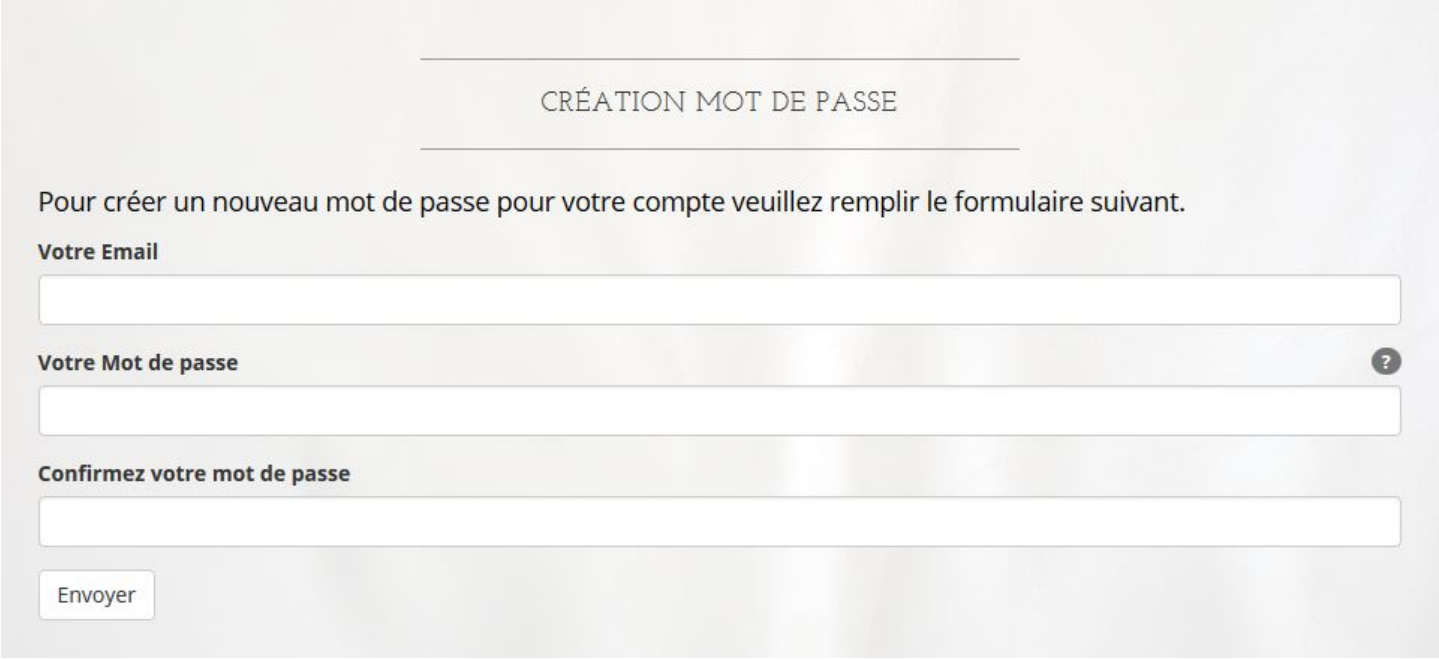
Un petit texte explicatif et un champ de saisie pour l'email. Le code est encore plus simple (**password.blade.php**) :

```
@extends('front.template')

@section('main')
        <div class="row">
                <div class="box">
                        <div class="col-lg-12">
                                @if(session()->has('status'))
                                        @include('partials/error',
['type' => 'success', 'message' => session('status')])
                                @endif
                                @if(session()->has('error'))
                                        @include('partials/error',
['type' => 'danger', 'message' => session('error')])
                                @endif
                                <hr>
                                   <h2 class="intro-text text-
center">{{ trans('front/password.title') }}</h2>
                                <hr>
                                <p>{{ trans('front/password.info')
}}</p>
                                    {!! Form::open(['url' =>
'password/email', 'method' => 'post', 'role' => 'form']) !!}

                                        <div class="row">

                                                {!!
Form::control('email',     6,     'email',     $errors,
```

```
trans('front/password.email')) !!}
                                                    {!!
Form::submit(trans('front/form.send'), ['col-lg-12']) !!}
                                                    {!!
Form::text('address', '', ['class' => 'hpet']) !!}
                                        </div>

                                {!! Form::close() !!}

                        </div>
                </div>
        </div>
@stop
```

Le nouveau mot de passe

Voici le visuel :



Trois champs de saisie :

- l'email
- le mot de passe
- la confirmation du mot de passe

Avec la même organisation du code (**reset.blade.php**) :

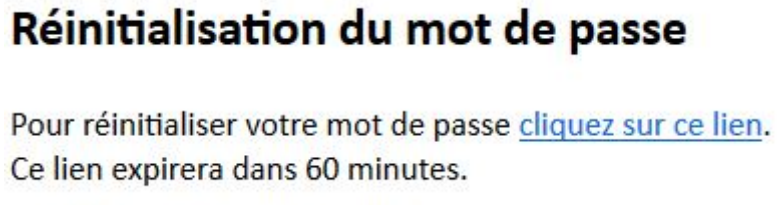
```
@extends('front.template')
```

```
@section('main')
        <div class="row">
                <div class="box">
                        <div class="col-lg-12">
                                @if(session()->has('error'))
                                        @include('partials/error',
['type' => 'danger', 'message' => session('error')])
                                @endif
                                <hr>
                                   <h2 class="intro-text text-
center">{{ trans('front/password.title-reset') }}</h2>
                                <hr>
                                <p>{{ trans('front/password.reset-
info') }}</p>

                                        {!! Form::open(['url' =>
'password/reset', 'method' => 'post', 'role' => 'form']) !!}

                                                <div class="row">

                                                                {!!
Form::hidden('token', $token) !!}
                                                                {!!
Form::control('email',     6,     'email',     $errors,
trans('front/password.email')) !!}
                                                                {!!
Form::control('password',   6,   'password',   $errors,
trans('front/password.password'),                   null,
[trans('front/password.warning'),  trans('front/password.warning-
password')]) !!}
                                                                {!!
Form::control('password', 6, 'password_confirmation', $errors,
trans('front/password.confirm-password')) !!}
                                                                {!!
Form::submit(trans('front/form.send'), ['col-lg-12']) !!}

                                                </div>

                                {!! Form::close() !!}

                        </div>
                </div>
        </div>
```

```
@stop

@section('scripts')

        <script>
                $(function() { $('.badge').popover();        });
        </script>

@stop
```

L’email

L’email reçu par l’utilisateur qui a oublié son mot de passe se présente ainsi :



Avec ce code (**emails/auth/password.blade.php**) :

```
<!DOCTYPE html>
<html lang="fr">
        <head>
                <meta charset="utf-8">
        </head>
        <body>
                <h2>{{ $title }}</h2>

                <div>
                        {!! $intro . link_to('password/reset/' .
$token, $link) !!}.<br>
                                                {{  $expire  .
config('auth.reminder.expire', 60)  .  $minutes}}.
                </div>
        </body>
</html>
```

On a ainsi fait le tour de tout ce qui concerne l’authentification pour cette application.