Laravel 5.7 par la pratique – Le profil

Maintenant notre galerie est globalement opérationnelle. Dans ce chapitre nous allons mettre en place une page de profil pour les utilisateurs pour leur permettre de modifier leur adresse courriel, la pagination et leur statut adulte. Pour respecter le RGPD on va aussi leur donner la possibilité de supprimer leur compte et de télécharger toutes les informations qui les concernent.

Un oubli…

Dans un chapitre précédent j'ai rendu le nom d'inscription unique mais j'ai oublié de mettre à jour la validation dans **RegisterController**, on va donc remédier à cette situation maintenant :

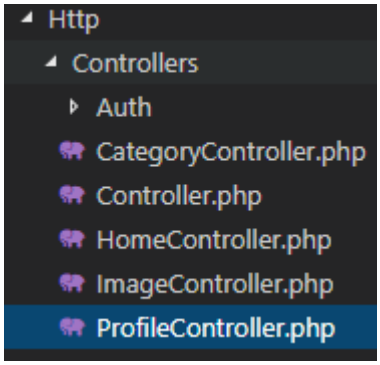
```
protected function validator(array $data)
{
    return Validator::make($data, [
        'name' => 'required|string|max:255|unique:users',
        'email' => 'required|string|email|max:255|unique:users',
        'password' => 'required|string|min:6|confirmed',
    ]);
}
```

Je pense qu'il doit trainer encore quelques bugs, si vous les trouvez…

Le contrôleur et les routes

On crée un nouveau contrôleur :

```
php artisan make:controller ProfileController --resource
```



On va conserver seulement les fonctions **edit**, **update** et **destroy** et ajouter la référence du modèle :

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\User;

class ProfileController extends Controller
```

Et on complète les routes :

```
Route::middleware ('auth', 'verified')->group (function () {
    Route::resource ('profile', 'ProfileController', [
        'only' => ['edit', 'update', 'destroy', 'show'],
        'parameters' => ['profile' => 'user']
    ]);
    ...

});
```

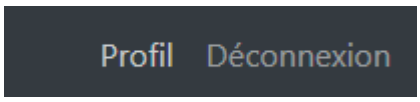
Lorsqu'on crée ainsi une route pour une ressource le paramètre prend le nom de la ressource, donc ici **profile**. Pour profiter de la liaison automatique avec le modèle on change ce nom pour **user** avec l'option **parameters**.

```
GET|HEAD  | profile/{user}  | profile.show    | App\Http\Controllers\ProfileController@show
PUT|PATCH | profile/{user}  | profile.update  | App\Http\Controllers\ProfileController@update
DELETE    | profile/{user}  | profile.destroy | App\Http\Controllers\ProfileController@destroy
```

Le menu

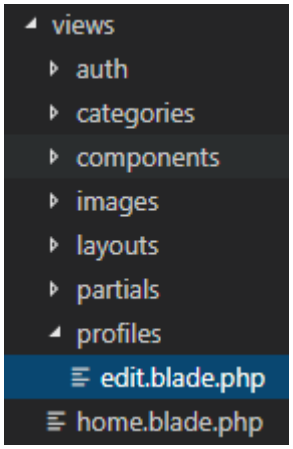
Il nous faut encore ajouter un item dans le menu (**views/layouts/app**) réservé aux utilisateurs connectés :

```
<ul class="navbar-nav ml-auto">
    @guest

        ...

    @else
        <li class="nav-item{{ currentRoute(
                    route('profile.edit', auth()->id()),
                    route('profile.show', auth()->id())
                )}}">
             <a class="nav-link" href="{{ route('profile.edit',
auth()->id()) }}">@lang('Profil')</a>
        </li>

        ...

    @endguest
</ul>
```



La vue

On crée un dossier et une vue pour le formulaire :



Avec ce code pour la vue :

```
@extends('layouts.form')

@section('css')

    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-slider/10.0
.0/css/bootstrap-slider.min.css">

@endsection

@section('card')

    @component('components.card')

        @slot('title')
            @lang('Modifer le profil')
             <a href="{{ route('profile.destroy', $user->id) }}"
class="btn btn-danger btn-sm pull-right invisible" role="button"
aria-disabled="true"><i  class="fas  fa-angry  fa-lg"></i>
@lang('Supprimer mon compte')</a>
        @endslot

         <form method="POST" action="{{ route('profile.update',
$user->id) }}">
            @csrf
            @method('PUT')

            @include('partials.form-group', [
                'title' => __('Adresse email'),
                'type' => 'email',
                'name' => 'email',
                'required' => true,
                'value' => $user->email,
            ])

            <div id="slider" class="form-group invisible">
                    @lang('Pagination : ')<span id="nbr">{{
$user->settings->pagination }}</span> @lang('images par page')<br>
                    <input id="pagination" name="pagination"
type="number" data-slider-min="3" data-slider-max="20"
                        data-slider-step="1" data-slider-value="{{
```

```
$user->settings->pagination }}"/><br>
            </div>

            <div class="form-group">
                <div class="custom-control custom-checkbox">
                    <input type="checkbox" class="custom-control-
input" id="adult" name="adult" {{ $user->settings->adult ?
'checked' : '' }}>
                        <label class="custom-control-label"
for="adult"> @lang('Je déclare être adulte')</label>
                </div>
            </div>

              <a href="{{ route('profile.show', $user->id) }}"
class="btn btn-warning invisible" role="button" aria-
disabled="true"><i class="fas fa-dolly fa-lg"></i> @lang('Exporter
mes données personnelles')</a>

            @component('components.button')
                @lang('Envoyer')
            @endcomponent

        </form>

    @endcomponent

@endsection

@section('script')

                                                    <script
src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-slider/10.2.
0/bootstrap-slider.min.js"></script>

    <script>
        $(() => {
            $("#pagination")
                .slider()
                .on("slide", (e) => {
                    $("#nbr").text(e.value)
                })
                .on("change", (e) => {
                    $("#nbr").text(e.value.newValue)
```

```
                })
            $('#slider, a').removeClass('invisible');
        })
    </script>

     @include('partials.script-delete', ['text' => __('Vraiment supprimer votre compte ?'), 'return' => 'home'])

@endsection
```

Boostrap 4 n’est pas équipé d’un slider alors on utilise [celui-ci](#) :

Il y a plusieurs exemples de mise en œuvre sur le site.

Comme on ne s’en sert que sur cette page on ne va pas l’ajouter dans les assets mais juste le charger par un CDN.

On va ajouter un peu de code dans la vue partielle **views/partials/script-delete** :

```
.done(() => {
    @switch($return)

        ...

        @case('home')
            location.replace('/')
            @break
    @endswitch
})
```

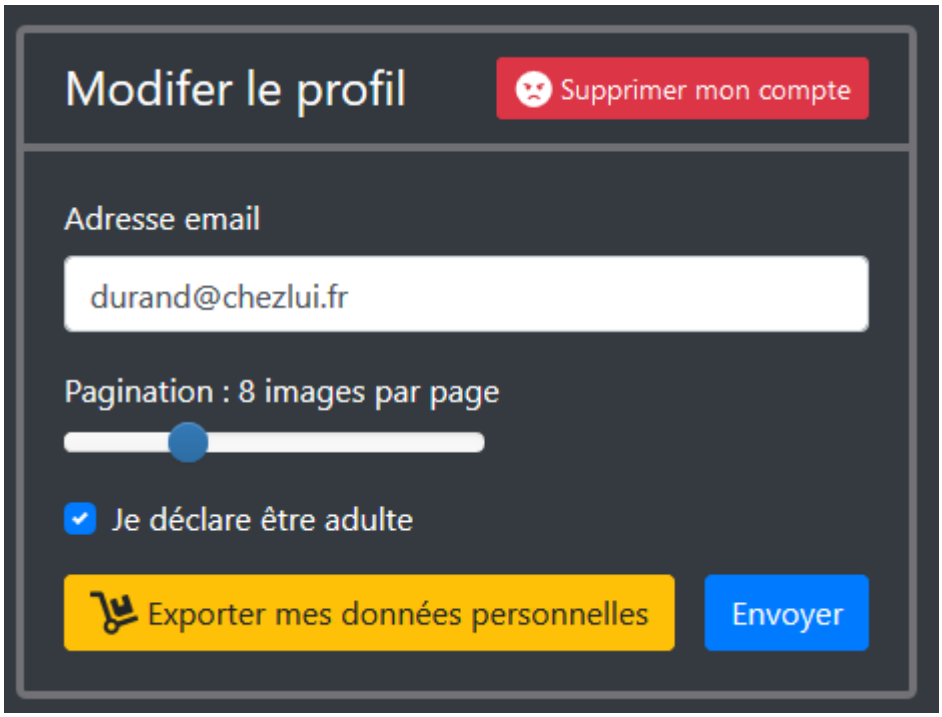
Le formulaire et sa soumission

L’affichage du formulaire

On complète **ProfileController** :

```
public function edit(User $user)
{
    return view ('profiles.edit', compact ('user'));
}
```

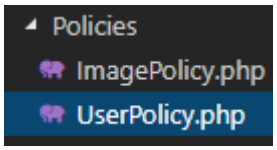
Et le formulaire peut maintenant s'afficher :



L'autorisation

Mais on va quand même protéger cet accès et le limiter à l'utilisateur concerné. On crée donc une nouvelle autorisation :

```
php artisan make:policy UserPolicy --model=User
```



Et on change ainsi le code :

```
<?php

namespace App\Policies;

use App\Models\User;
use Illuminate\Auth\Access\HandlesAuthorization;

class UserPolicy
```

```
{
    use HandlesAuthorization;

    public function manage(User $user, User $userprofile)
    {
        return $user->id === $userprofile->id;
    }
}
```

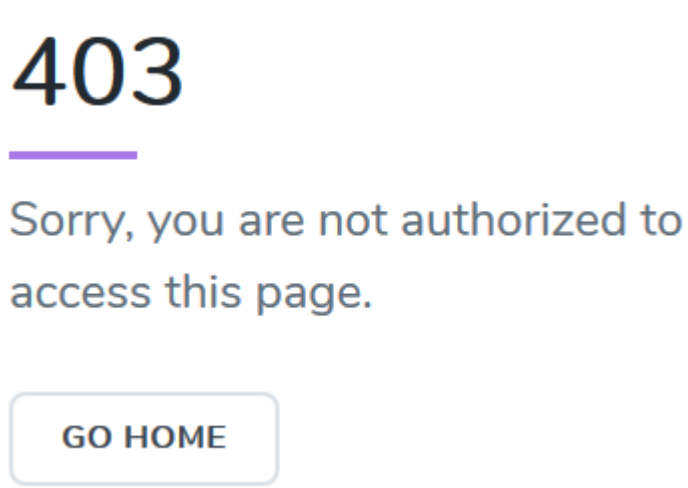
On le référence dans **AuthServiceProvider** :

```
use App\Models\ { Image, User };
use App\Policies\ { ImagePolicy, UserPolicy };
    ...

protected $policies = [
    Image::class => ImagePolicy::class,
    User::class => UserPolicy::class,
];
```

Et on l'ajoute dans le contrôleur :

```
public function edit(User $user)
{
    $this->authorize ('manage', $user);
    return view ('profiles.edit', compact ('user'));
}
```

Maintenant un accès à un autre profil que le sien est interdit :

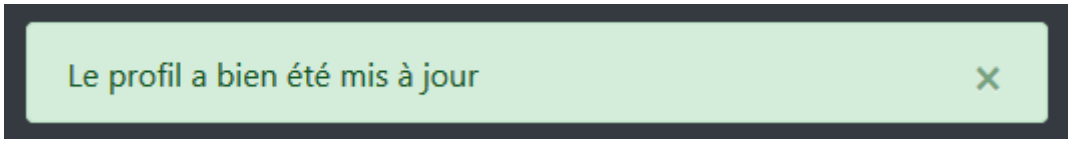


La soumission

On complète **ProfileController** pour traiter la soumission :

```
public function update(Request $request, User $user)
{
    $this->authorize ('manage', $user);

    $request->validate ([
                                                'email'      =>
'required|string|email|max:255|unique:users,email,' . $user->id,
        'pagination' => 'required',
    ]);

    $user->update ([
        'email' => $request->email,
        'settings' => json_encode ([
            'pagination' => (integer)$request->pagination,
            'adult' => $request->filled('adult'),
        ]),
    ]);

    return back ()->with ('ok', __ ('Le profil a bien été mis à
jour'));
}
```

Et on vérifie que ça fonctionne !

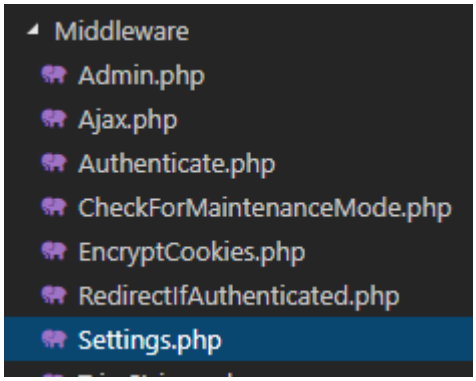


Un middleware

Ce n’est pas parce que maintenant la valeur de la pagination est fixée dans la table que ça va magiquement la changer dans la galerie !

On crée un nouveau middleware :

```
php artisan make:middleware Settings
```



Et on code ainsi :

```
<?php

namespace App\Http\Middleware;

use Closure;

class Settings
{
    public function handle($request, Closure $next)
    {
        if (auth ()->check ()) {
                config (['app.pagination' => auth ()->user
()->pagination]);
        }

        return $next($request);
    }
}
```

On voit qu'on fait appel à l'attribut **pagination** du modèle **User**. Cet attribut n'existe pas, il faut le créer à partir des données du setting de la même manière qu'on avait créé **adult**. Donc dans le modèle **User** on ajoute cet accesseur :

```
public function getPaginationAttribute()
{
    return $this->settings->pagination;
}
```

Si le visiteur est authentifié on récupère son réglage de pagination et on actualise la configuration.

On déclare ce middleware dans **app/Http/Kernel** :

```
protected $middlewareGroups = [
    'web' => [

        ...

        \App\Http\Middleware\Settings::class,
    ],

    ...
];
```

Et maintenant la pagination personnalisée doit fonctionner !

La suppression du compte

On a prévu un bouton pour supprimer le compte :

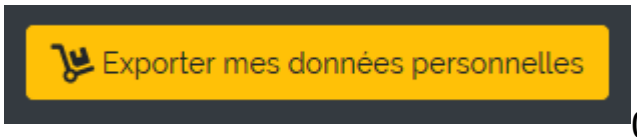
On va ajouter dans le contrôleur **ProfileController** le code pour que ça se réalise :

```
public function __construct()
{
    $this->middleware('ajax')->only('destroy');
}

public function destroy(User $user)
{
    $this->authorize ('manage', $user);
    $user->delete();
    return response ()->json ();
}
```

On a quand même prévu dans la vue une alerte pour éviter la suppression accidentelle !

Les données personnelles

On va terminer avec l'export des données personnelles (encore le RGPD !) :

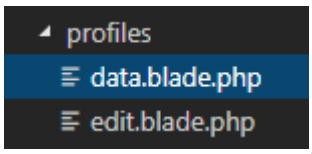


On va ajouter ce code dans **ProfileController** :

```
use App\Repositories\ImageRepository;

    ...

public function show(ImageRepository $imageRepository, User $user)
{
    $this->authorize ('manage', $user);
    $images = $imageRepository->getImagesForUser ($user->id);
    return view ('profiles.data', compact ('user', 'images'));
}
```

On vérifie que l'utilisateur à le droit d'accéder à cette information, on récupère ses images grâce au repository, puis on envoie la vue.

On a déjà la bonne fonction dans le repository, par contre il nous manque la vue, on la crée :



Avec ce code :

```
@extends('layouts.app')

@section('content')
    <main class="container-fluid">
        <h1>@lang('Export des données personnelles')</h1>
        <div class="card">
            <div class="card-body">
                    <h5 class="card-title">@lang('A propos')</h5>
                <table class="table">
                    <tr>
```

```
                    <td>@lang('Rapport généré pour : ')</td>
                    <td>{{ $user->email }}</td>
                </tr>
                <tr>
                    <td>@lang('Pour le site :')</td>
                    <td>{{ env('APP_NAME') }}</td>
                </tr>
                <tr>
                    <td>@lang("A l'url :")</td>
                    <td>{{ env('APP_URL') }}</td>
                </tr>
                 <tr>
                     <td>@lang('Le :')</td>
                       <td>{{ now()->formatLocalized('%x') }}</td>
                 </tr>
            </table>
            <em>@lang('Vous pouvez enregistrer cette page pour conserver vos données en utilisant le menu de votre navigateur.')</em>
        </div>
    </div>
    <br>
    <div class="card">
        <div class="card-body">
            <h5 class="card-title">@lang('Utilisateur')</h5>
            <table class="table">
                <tr>
                    <td>@lang("ID : ")</td>
                    <td>{{ $user->id }}</td>
                </tr>
                <tr>
                    <td>@lang("Nom de connexion : ")</td>
                    <td>{{ $user->name }}</td>
                </tr>
                <tr>
                    <td>@lang("Email :")</td>
                    <td>{{ $user->email }}</td>
                </tr>
                <tr>
                    <td>@lang("Date d'inscription :")</td>
                                                <td>{{ $user->created_at->formatLocalized('%x') }}</td>
```

```
                    </tr>
                </table>
            </div>
        </div>
        <br>
        @unless($images->isEmpty())
            <div class="card">
                <div class="card-body">
                    <h5 class="card-title">@lang('Medias')</h5>
                     <table class="table" style="margin-bottom:
140px">
                        @foreach($images as $image)
                            <tr>
                                <td>@lang('Adresse web :')</td>
                                <td>
                                    <div class="hover_img">
                                        <a href="{{ url('images/'
.  $image->name)  }}"  target="_blank">{{  url('images/'  .
$image->name) }}<span><img src="{{ url('thumbs/' . $image->name)
}}" alt="image" height="150" /></span></a>
                                    </div>
                                </td>
                            </tr>
                        @endforeach
                    </table>
                </div>
            </div>
        @endunless
    </main>
@endsection
```

L’utilisateur dispose ainsi de quelques généralités :

Export des données personnelles

A propos

Rapport généré pour :	dupont@chezlui.fr
Pour le site :	Album
A l'url :	album-plus.oo
Le :	20/09/2018

Vous pouvez enregistrer cette page pour conserver vos données en utilisant le menu de votre navigateur.

Ses identifiants :

Utilisateur

ID :	2
Nom de connexion :	Dupont
Email :	dupont@chezlui.fr
Date d'inscription :	12/09/2018

Les liens de ses images avec miniature au survol :

Medias

Adresse web :	http://album-plus.oo/images/RIPqCFrTB8dZ9KEv401egPQsgnbCHSbmhZhNVIVQ.jpeg
Adresse web :	55hgNZs0cW2SuDeJWUDUxhLLlhEwPiK6XgewnA.jpeg
Adresse web :	VyC69sLeu5ZPV1isNLvUzY5jAYoEsfjTpVPPK.jpeg
Adresse web :	sdZqwNw6fIWoQCsb13uFw1W4DiDRHuU4tZONT.jpeg
Adresse web :	http://album-plus.oo/images/hVCKABCaltIPhop9nQZBoZb7CFFwgGCYYTLgQEvE.jpeg

Remarque : les dates sont encore au format américain (mois/jour), on arrangera ça dans un chapitre ultérieur !

Conclusion

Dans ce chapitre on a :

- ajouté un item dans le menu pour le profil
- créé un contrôleur et les routes pour le profil
- créé le formulaire de modification du profil
- codé le traitement du formulaire
- ajouté un middleware pour rendre effective la pagination personnalisée
- ajouté une autorisation pour la modification du profil
- ajouté la suppression du compte
- ajouté l'export des données personnelles

Pour vous simplifier la vie vous pouvez charger le projet dans son état à l'issue de ce chapitre.