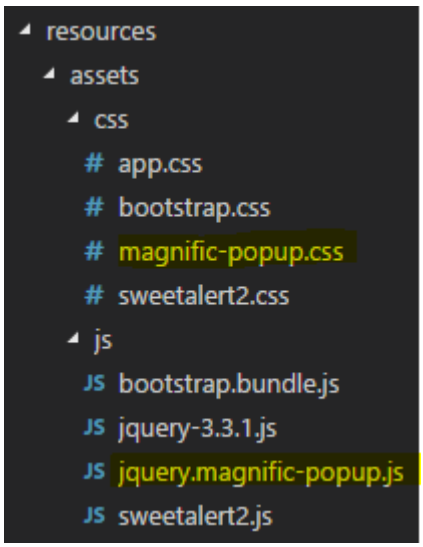
# Créer une application avec Laravel 5.5 – La galerie 1/2

Maintenant qu'on a créé l'essentiel de la gestion des catégories et des images on va enfin passer à la réalisation de la galerie elle-même pour visualiser les images sur la page d'accueil. On va aussi donner la possibilité de zoomer les images. On va prévoir une pagination, un affichage par catégorie..

# Une light box

Quand on fait une recherche sur Internet pour une light box on a surtout l'embarras du choix. je me suis décidé pour Magnific Popup qui est simple, rapide et esthétique. Vous pouvez récupérer les fichiers sur Github.

On ajoute ça dans les assets :




On complète **webpack.mix.js** :

```
mix.styles([
    'resources/assets/css/bootstrap.css',
    'resources/assets/css/sweetalert2.css',
    'resources/assets/css/magnific-popup.css',
    'resources/assets/css/app.css',
], 'public/css/app.css')
.scripts([
```

```
    'resources/assets/js/jquery-3.3.1.js',
    'resources/assets/js/bootstrap.bundle.js',
    'resources/assets/js/sweetalert2.js',
    'resources/assets/js/jquery.magnific-popup.js',
], 'public/js/app.js');
```

Et on relance npm…

Pour l'utilisation on trouve une documentation assez détaillée sur le site.

# Une nouvelle directive Blade

On a déjà créé une directive Blade pour filtrer des administrateurs. On va en ajouter une pour filtrer « owner ou admin ». On ajoute ce code dans **AppServiceProvider** :

```
public function boot()
{
    ...

    Blade::if('adminOrOwner', function ($id) {
          return auth()->check() && (auth()->id() === $id ||
auth()->user()->role === 'admin');
    });

    ...
}
```

On pourra ainsi écrire **@adminOrOwner** dans les vues ! Ça va nous servir pour autoriser la suppression des images.

# La vue pour la galerie

Pour l'affichage des vignettes de photos on va utiliser encore le composant card de Bootstrap 4 avec l'intéressante possibilité de les afficher automatiquement en colonnes sans effort !

Voilà le nouveau code de la vue **views/home** :

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

```
@section('content')

    <main class="container-fluid">
        @isset($category)
            <h2 class="text-title mb-3">{{ $category->name }}</h2>
        @endif
        @isset($user)
             <h2 class="text-title mb-3">{{ __('Photos de ') .
$user->name }}</h2>
        @endif
        <div class="card-columns">
            @foreach($images as $image)
                <div class="card">
                     <a href="{{ url('images/' . $image->name) }}"
class="image-link"><img class="card-img-top" src="{{ url('thumbs/'
. $image->name) }}" alt="image"></a>
                    @isset($image->description)
                        <div class="card-body">
                                    <p class="card-text">{{
$image->description }}</p>
                        </div>
                    @endisset
                    <div class="card-footer text-muted">
                        <small><em>
                                 <a href="#" data-toggle="tooltip"
title="{{ __('Voir les photos de ') . $image->user->name }}">{{
$image->user->name }}</a>
                            </em></small>
                        <small class="pull-right">
                            <em>
                                {{ $image->created_at }}
                                @adminOrOwner($image->user_id)
                                 <a class="form-delete" href="{{
route('image.destroy', $image->id) }}" data-toggle="tooltip"
title="@lang('Supprimer cette photo')"><i class="fa fa-
trash"></i></a>
                                            <form action="{{
route('image.destroy', $image->id) }}" method="POST" class="hide">
                                    {{ csrf_field() }}
                                    {{ method_field('DELETE') }}
                                </form>
                                @endadminOrOwner
                            </em>
```

```
                        </small>
                    </div>
                </div>
            @endforeach
        </div>
        <div class="d-flex justify-content-center">
            {{ $images->links() }}
        </div>
    </main>
@endsection

@section('script')
    <script>
        $(function() {
            $('[data-toggle="tooltip"]').tooltip()

            $('.card-columns').magnificPopup({
                delegate: 'a.image-link',
                type: 'image',
                gallery: { enabled: true }
            });

            $('a.form-delete').click(function(e) {
                e.preventDefault();
                let href = $(this).attr('href')
                $("form[action='" + href + "']").submit()
            })
        })
    </script>
@endsection
```

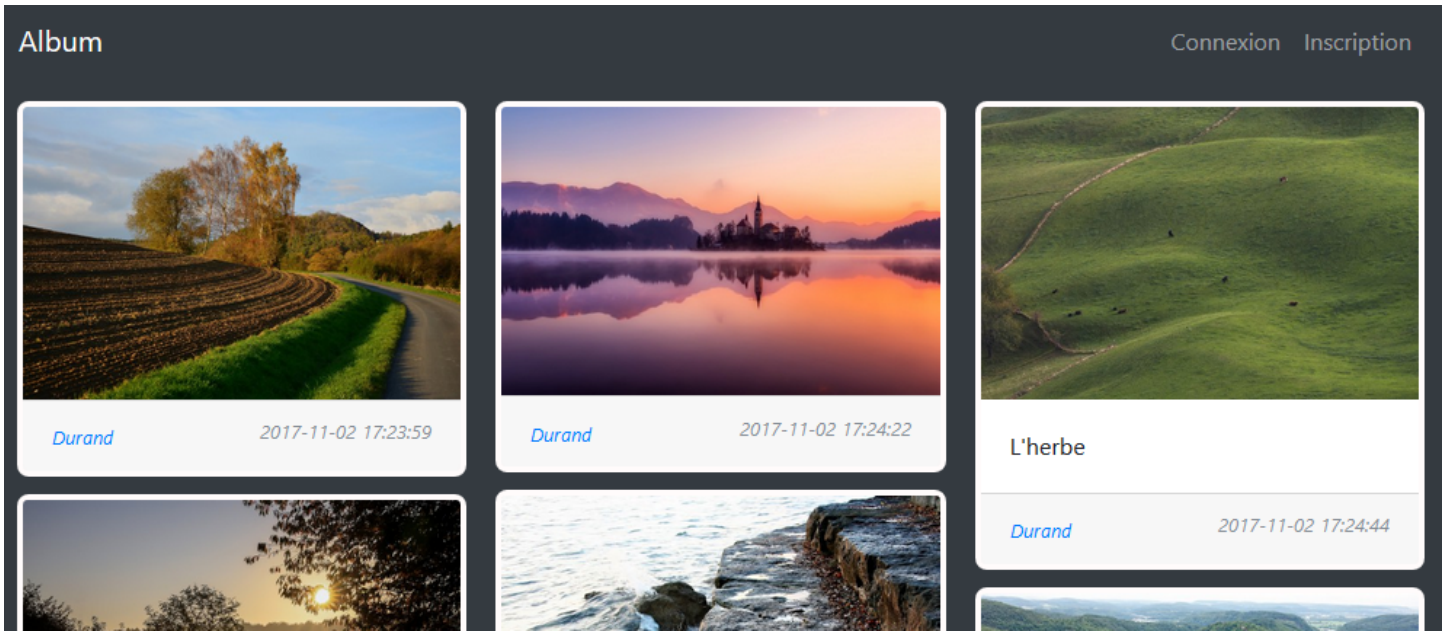
# Le contrôleur HomeController

On va compléter le code du contrôleur **HomeController** pour envoyer les images dans la vue :

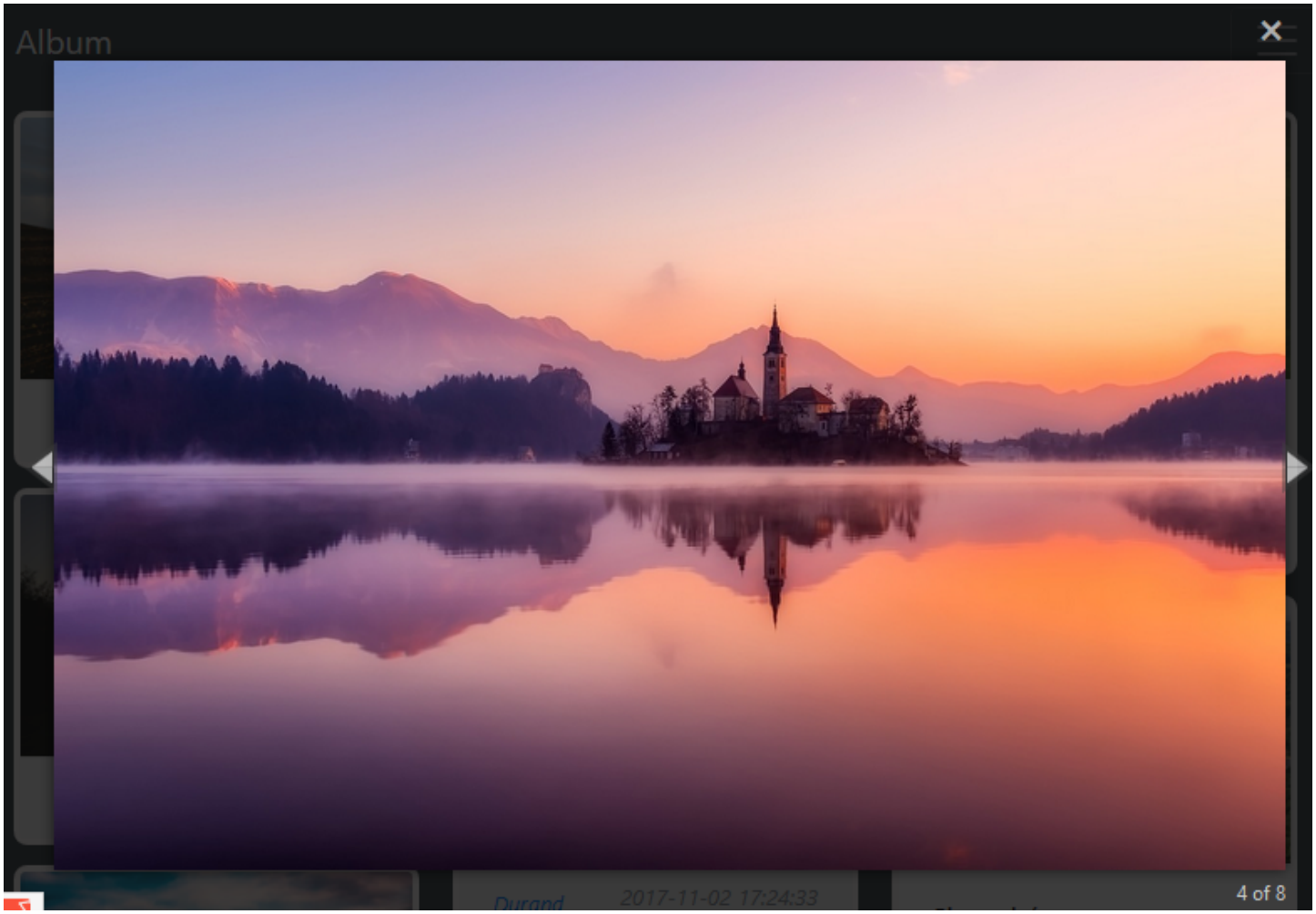
```
use App\Models\Image;

...

public function index()
{
```

```
    $images = Image::paginate(8);

    return view('home', compact('images'));
}
```

On peut enfin voir la galerie :




Vérifiez que la lightbox fonctionne :




Les dates ne sont pas formatées mais on s'en occupera quand on traitera les langues.

Le nom de celui qui a envoyé la photo apparaît mais on a pas prévu l'eager loading et ce n'est pas vraiment propre. D'autre part on devrait afficher les photos de la plus récente à la plus ancienne. On va créer un scope dans le modèle **Image** :

```
public function scopeLatestWithUser($query)
{
    return $query->with('user')->latest();
}
```

Et changer le code du contrôleur :

```
$images                                                    =
Image::latestWithUser()->paginate(config('app.pagination'));
```

Maintenant c'est beaucoup mieux !

# La pagination

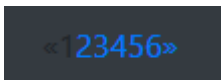
Pour faire quelque chose de plus correct on va mettre la valeur de la pagination dans la configuration (**config/app.php**) :

```
'pagination' => 8,
```

Et on modifie le code dans le contrôleur :

```
$images = Image::paginate(config('app.pagination'));
```

Mais il ne vous a sans doute pas échappé que la pagination n'est pas très jolie :




c'est parce qu'elle est prévue par défaut pour Bootstrap 3 et nous on utilise Bootstrap 4. Pour arranger ça ajoutez ce code dans **AppServiceProvider** :

```
use Illuminate\Pagination\AbstractPaginator;

...


public function boot()
{
    ...

    AbstractPaginator::defaultView("pagination::bootstrap-4");
}
```

Rafraichissez la page, normalement ça doit aller mieux :

# L’affichage par catégorie

Pour l’affichage par catégorie on va commencer par ajouter la route :

```
Route::name('category')->get('category/{slug}', 'ImageController@category');
```

Ajouter cette fonction dans le contrôleur **ImageController** :

```
use App\Models\Category;

...

public function category($slug)
{
    $category = Category::whereSlug($slug)->firstorFail();

    $images = $this->repository->getImagesForCategory($slug);

    return view('home', compact('category', 'images'));
}
```

Et celle ci dans le repository **ImageRepository** :

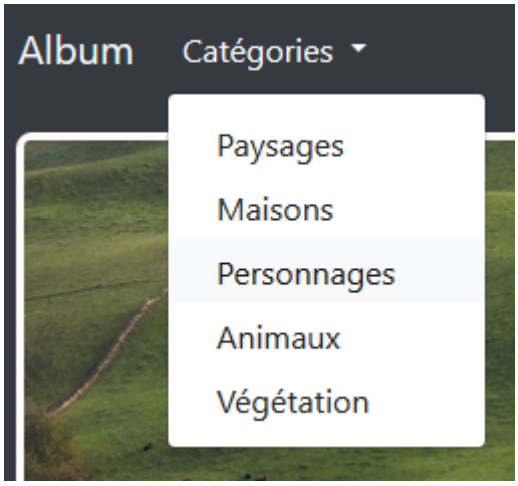
```
public function getImagesForCategory($slug)
{
    return Image::latestWithUser()->whereHas('category', function ($query) use ($slug) {
        $query->whereSlug($slug);
    })->paginate(config('app.pagination'));
}
```

On ajoute un menu déroulant dans la barre de navigation :

```
<li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle
        @isset($category)
            {{ currentRoute(route('category', $category->slug)) }}
        @endisset
            " href="#" id="navbarDropdownCat" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
        @lang('Catégories')
    </a>
```

```
                <div    class="dropdown-menu"    aria-
labelledby="navbarDropdownCat">
        @foreach($categories as $category)
             <a class="dropdown-item" href="{{ route('category',
$category->slug) }}">{{ $category->name }}</a>
        @endforeach
    </div>
</li>
@admin
```

Et normalement :



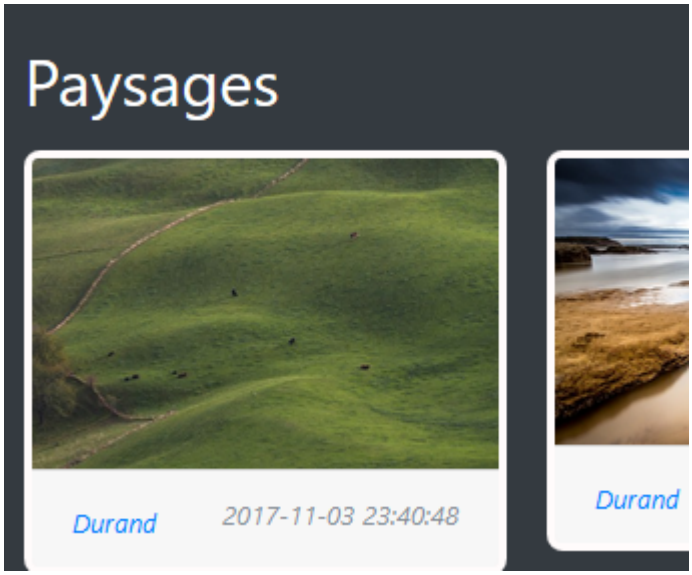

On affiche bien les images par catégorie mais le nom de la catégorie n’est pas visible…

Ajoutez ce code dans **resources/assets/css/app.css** :

```
.text-title, h1 {
    color: white;
}
```

Et relancez npm.

Le nom de la catégorie doit être maintenant visible :




On verra dans le prochain chapitre l'affichage par utilisateur et la suppression des images.

# Conclusion

Dans ce chapitre on a :

- ajouté une lighbox pour la visualisation des images
- ajouté une directive Blade
- créé la route et la vue pour la galerie
- adapté la pagination à Bootstrap 4
- ajouté un menu pour les catégorie
- écrit le code pour l'affichage par catégorie

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