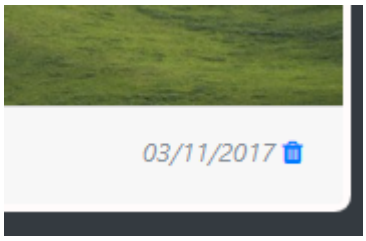
Créer une application avec Laravel 5.5 – Ajouter le changement de catégorie

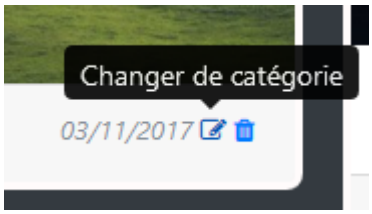
Dans les commentaires pour cette application j'ai eu la demande de pouvoir changer la catégorie des photos. Comme c'est quelque chose qui touche à pas mal de points et qui peut s'avérer didactique je détaille dans cet article tout le processus.

Le visuel

On ne doit permettre la modification de la catégorie d'une photo que pour le propriétaire de la photo et pour l'administrateur. Pour le moment on dispose juste d'une icône pour la suppression de la photo :



On va ajouter une nouvelle icône pour le changement de la catégorie :



Ça se passe dans la vue **home** :

```
@adminOrOwner($image->user_id)
    <a class="category-edit" id="{{$image->category_id}}" href="#" data-toggle="tooltip" title="@lang('Changer de catégorie')"><i class="fa fa-edit"></i></a>
     <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>
    ...
@endadminOrOwner
```

Il faut à présent décider comment on va présenter les choses, il faut pouvoir sélectionner une catégorie dans une liste.

Il me semble que le plus simple est d'utiliser une feuille modale, d'autant qu'on dispose de Bootstrap pour le faire.

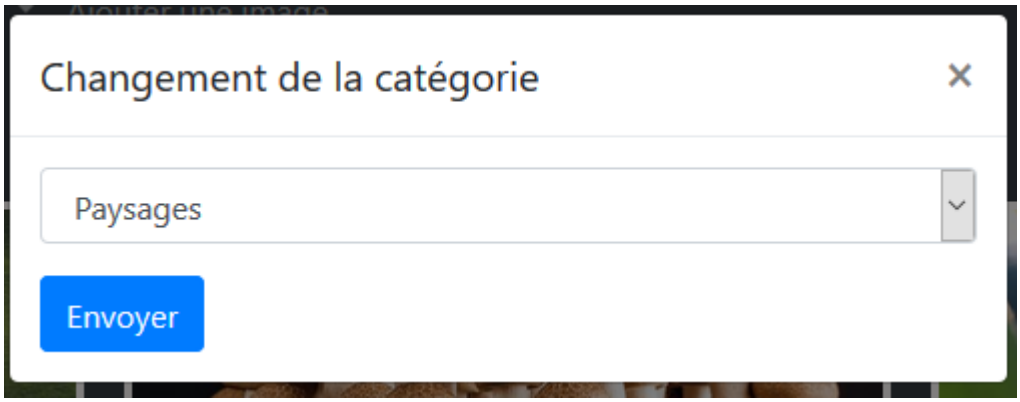
Toujours dans ma vue **home** j'ajoute le code pour la feuille modale :

```
<div class="modal fade" id="changeCategory" tabindex="-1" role="dialog" aria-labelledby="categoryLabel" aria-hidden="true">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                                    <h5 class="modal-title" id="categoryLabel">@lang('Changement de la catégorie')</h5>
                  <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                   <span aria-hidden="true">&times;</span>
                </button>
            </div>
            <div class="modal-body">
                <form action="" method="POST">
                    <div class="form-group">
                              <select class="form-control" name="category_id">
                        @foreach($categories as $cat)
                             <option value="{{ $cat->id }}">{{ $cat->name }}</option>
                        @endforeach
                        </select>
                    </div>
                       <button type="submit" class="btn btn-primary">@lang('Envoyer')</button>
                </form>
            </div>
        </div>
    </div>
</div>
```

Et le déclencheur dans le Javascript :

```
$('.category-edit').click(function (e) {
    e.preventDefault()
    $('#changeCategory').modal('show')
})
```

En cliquant sur l'icône j'ouvre bien la feuille modale et je dispose de la liste des catégories :



J'ai juste un petit souci : je n'ai pas la catégorie actuelle sélectionnée. Comment faire ?

On va régler ça en ajoutant un petite ligne de script :

```
$('.category-edit').click(function (e) {
    e.preventDefault()
    $('select').val($(this).attr('id'))
    $('#changeCategory').modal('show')
})
```

Et maintenant tout va bien ! Passons maintenant à la partie serveur…

Routes et contrôleur

Routes

Au niveau des routes on en a besoin d'une seule, on va ajouter **update** :

```
Route::middleware('auth')->group(function () {
```

```
    ...

    Route::resource('image', 'ImageController', [
        'only' => ['create', 'store', 'destroy', 'update']
    ]);

})
```

```
PUT|PATCH | image/{image}                | image.update          | App\Http\Controllers\ImageController@update
```

Contrôleur

Dans le contrôleur **ImageController** on ajoute la méthode **update** :

```
/**
 * Update the specified resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @param \App\Models\Image
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, Image $image)
{
 $image->category_id = $request->category_id;
 $image->save();

 return redirect()->back();
}
```

Le formulaire

Il nous faut revenir à notre formulaire pour le compléter :

```
<form action="" method="POST">
    <input type="hidden" name="_method" value="PUT">
    <input type="hidden" name="_token" value="{{ csrf_token() }}">
```

J’ai laissé l’action vierge parce qu’au départ on ne sait pas quelle image va être concernée. Il va donc falloir renseigner cette information en ajoutant un peu de Javascript :

```
$('a.category-edit').click(function (e) {
    e.preventDefault()
```

```
    $('select').val($(this).attr('id'))
    $('form').attr('action', $(this).next().attr('href'))
    $('#changeCategory').modal('show')
})
```

Pour l'url l'astuce est d'aller récupérer celle prévue dans le formulaire de suppression parce qu'elle est exactement celle dont on a besoin !

Et là ça doit fonctionner !

Un peu de sécurité

On ne veut pas que n'importe qui change la catégorie et un petit malin pourrait encore y parvenir là.

On va ajouter changer la méthode **delete** dans **ImagePolicy** pour la nommer **manage** (parce qu'elle ne va plus servir seulement à la suppression) :

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

On va actualiser la méthode **destroy** de **ImageController** :

```
public function destroy(Image $image)
{
    $this->authorize('manage', $image);

    $image->delete();

    return back();
}
```

Et ajouter l'autorisation dans **update** :

```
public function update(Request $request, Image $image)
{
    $this->authorize('manage', $image);
    $image->category_id = $request->category_id;
    $image->save();
```

```
    return redirect()->back();
}
```

Maintenant on est tranquilles !

Un alerte

On pourrait un peu améliorer en ajoutant une alerte pour prévenir l'utilisateur que le changement a bien eu lieu.

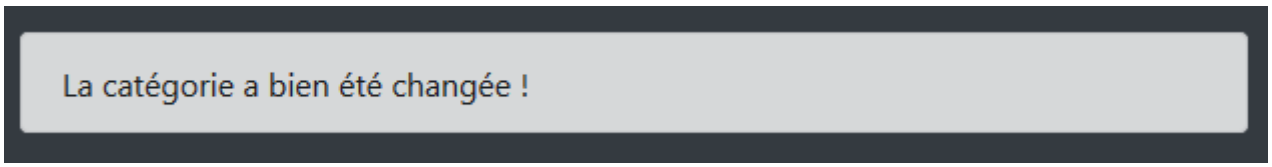
Dans la vue **home** on ajoute l'alerte :

```
<main class="container-fluid">
    @if(session('updated'))
        <div class="alert alert-dark" role="alert">
            {{ session('updated') }}
        </div>
    @endif
```

Et dans la méthode **update** du contrôleur on flashe la session :

```
return redirect()->back()->with('updated', __('La catégorie a bien été changée !'));
```

Et maintenant au changement on a :



Les langues

Comme l'application est prévue aussi en anglais il faut ajouter les textes. On va utiliser mon package des langues qui est normalement prévu avec l'application :

```
λ php artisan language:diff en
----------
Missing strings for this locale
----------
Changement de la catégorie
Changer de catégorie
La catégorie a bien été changée !
----------
No Further strings for this locale
----------
```

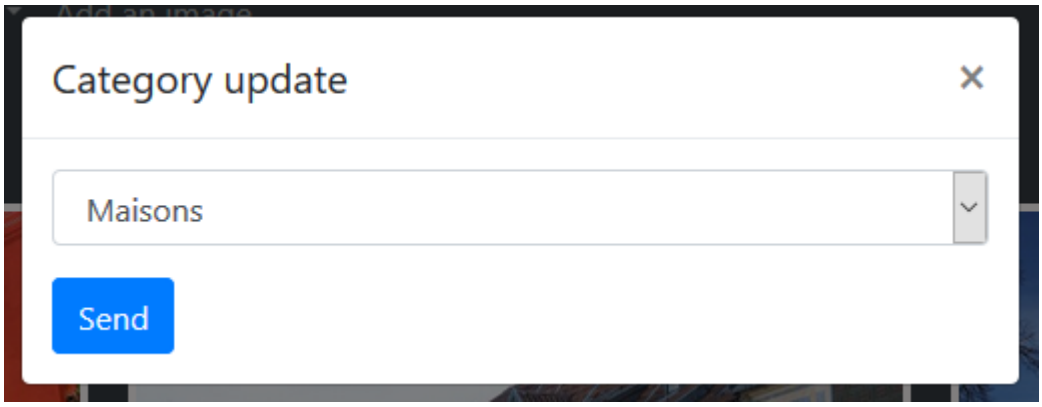
On voit qu'il nous manque 3 textes. On va les ajouter :

```
λ php artisan language:sync en
File successfully synchronised for missing strings
No further strings for this locale.
```

Il n'y a plus qu'à les entrer dans les emplacements prévus dans **en.json** :

```
{
    ...
    "Changement de la catégorie": "Category update",
    "Changer de catégorie": "Update category",
    ...
        "La catégorie a bien été changée !": "Category updated successfully !",
    ...
}
```

Et après vérifiez que vous avez bien travaillé :



Conclusion

On voit que pour ajouter une fonctionnalité il faut bien prendre en compte tous les éléments et être méthodiques. Ce n'est

évidemment pas la seule façon d’arriver au résultat mais ça me paraît faire partie des plus simples et rapides.