Laravel 5.7 par la pratique – Les notifications

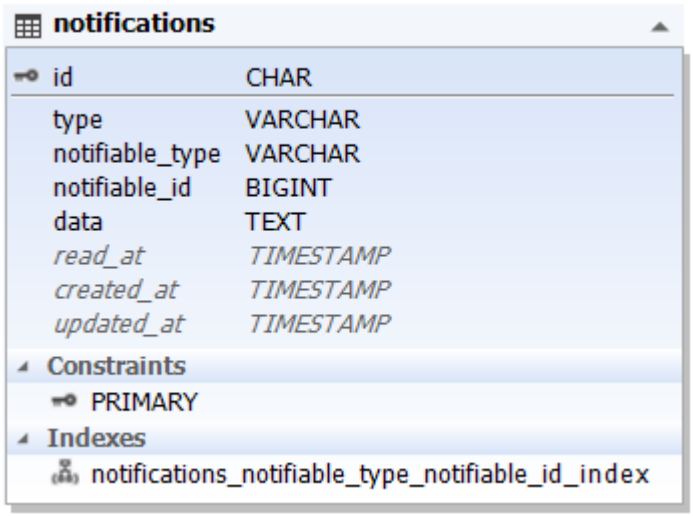
Notre galerie est désormais bien équipée avec ses catégories, ses albums, son administration, sa gestion du profil utilisateur, la notation des photos… Dans ce chapitre nous allons voir les notifications. Les utilisateurs seront prévenus si on a noté leurs photos. L'administrateur sera aussi prévenu lors de l'inscription d'un nouvel utilisateur. Dans le premier cas ce sera avec la présence d'une icône avec le nombre de notifications dans la barre de menu. Dans le second cas avec l'envoi d'un mail.

Notification par la base de données

La table des notifications

Lorsqu'on veut stocker les notifications dans la base de données en attendant que l'utilisateur en ait connaissance il faut commencer par créer une table spécifique :

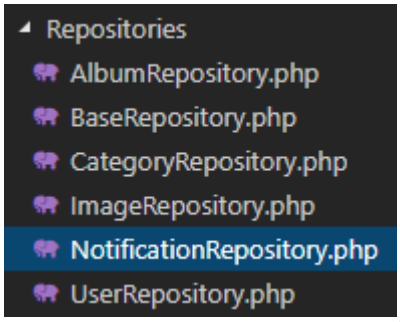
```
php artisan notifications:table
php artisan migrate
```



Voyons les champs de cette table :

- **id** : un identifiant unique
- **notifiable_type** : le modèle en relation (dans notre cas de notation d'image ça sera **User**)
- **notifiable_id** : l'id du modèle en relation (donc l'id de l'utilisateur notifié)
- **data** : les données qui seront sous forme JSON (là on met ce qu'on veut)
- **read_at** : le moment où la notification est marquée comme étant lue
- **created_at** : le moment où la notification est créée
- **updated_at** : le moment où la notification est modifiée

Histoire d'avoir déjà des données on va créer ce seeder :



Avec ce code :

```
<?php

use Illuminate\Database\Seeder;

class NotificationsTableSeeder extends Seeder
{
    public function run()
    {
        \DB::table('notifications')->insert([
            0 => [
                'id' => '6bd79182-0d88-48b7-8e4e-59dbf3371763',
                'type' => 'App\Notifications\ImageRated',
                'notifiable_type' => 'App\Models\User',
                'notifiable_id' => '2',
                                                    'data'   =>
'{"image":"hVCKABCaItIPhop9nQZBoZb7CFFwgGCYYTLgQEvE.jpeg","image_i
d":31,"rate":3,"user":3}'
            ],
            1 => [
                'id' => '6c7b833c-4a12-44d5-8fbe-f542e688b865',
```

```
                'type' => 'App\Notifications\ImageRated',
                'notifiable_type' => 'App\Models\User',
                'notifiable_id' => '2',
                                                    'data'    =>
'{"image":"RvlsdZqwNw6fIWoQCsb13uFw1W4DiDRHuU4tZONT.jpeg","image_i
d":32,"rate":5,"user":3}'
            ],
        ]);
    }
}
```

Mettez à jour **DatabaseSeeder** :

```
public function run()
{
    ...

    $this->call(NotificationsTableSeeder::class);
}
```

Puis rafraichissez la base :

```
php artisan migrate:fresh --seed
```

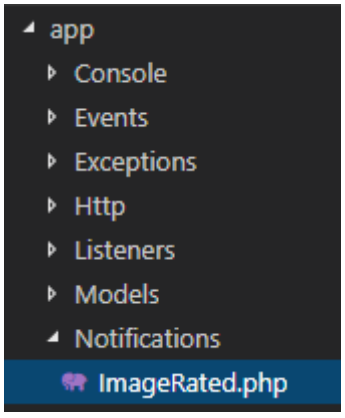
On a maintenant deux notifications :

id CHAR(36)	type VARCHAR(255)	notifiable_type VARCHAR(255)	notifiable_id UNSIGNED BIGINT(20)	data TEXT
6bd79182-0d88-48b7-8e4e-59dbf3371763	App\Notifications\ImageRated	App\Models\User	2	{"image":"hVCKABCaItIPhop9nQZBoZb7CFFwgGCYYTLgQEvE.jpeg","image_id":31,"rate":3,"user":3}
6c7b833c-4a12-44d5-8fbe-f542e688b865	App\Notifications\ImageRated	App\Models\User	2	{"image":"RvlsdZqwNw6fIWoQCsb13uFw1W4DiDRHuU4tZONT.jpeg","image_id":32,"rate":5,"user":3}

La notification

On crée maintenant la notification :

```
php artisan make:notification ImageRated
```



On change ainsi le code :

```
<?php

namespace App\Notifications;
use Illuminate\Notifications\Notification;

class ImageRated extends Notification
{
    protected $image;
    protected $rate;
    protected $user_id;

    public function __construct($image, $rate, $user_id)
    {
        $this->image = $image;
        $this->rate = $rate;
        $this->user_id = $user_id;
    }

    public function via()
    {
        return ['database'];
    }

    public function toArray()
    {
        return [
            'image' => $this->image->name,
            'image_id' => (integer)$this->image->id,
            'rate' => (integer)$this->rate,
            'user' => (integer)$this->user_id
        ];
    }
}
```

On va transmettre à la notification 3 valeurs :

- l’image concernée (**$image**)
- la note donnée (**$rate**)
- le propriétaire de l’image notée (**$user_id**)

Dans la fonction **via** on définit le canal, donc dans notre cas la base de données (**database**).

Dans la fonction **toArray** on définit les données transmises dans la base.

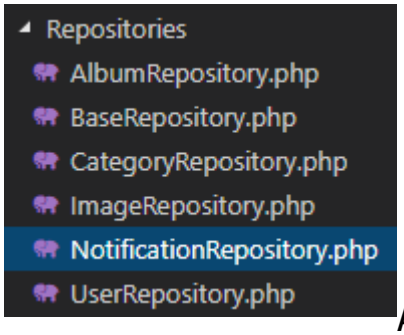
L’envoi de la notification

Dans le contrôleur **ImageController**, lorsqu’on reçoit une note pour une photo on doit créer la notification :

```
use App\Repositories\ { ImageRepository, NotificationRepository, AlbumRepository, Category Repository };
use App\Notifications\ImageRated;

    ...

public function rate(Request $request, Image $image)
{
    ...
    $this->imageRepository->setImageRate ($image);

    // Notification
    $notificationRepository->deleteDuplicate($user, $image);
     $image->user->notify(new ImageRated($image, $request->value, $user->id));

    return ...
}
```

On envoie la notification avec la méthode **notify** appliquée sur l’instance de **User**. On sait que **User** est déjà équipé du trait **Notifiable** sinon il aurait fallu l’ajouter.

On fait appel à un repository (**NotificationRepository**) qui n’existe pas encore pour accomplir une action : si on reçoit une note pour une photo déjà notifiée et non lue on supprime le doublon. On crée ce repository :



Avec ce code :

```
<?php

namespace App\Repositories;

use Illuminate\Support\Facades\DB;

class NotificationRepository
{
    public function deleteDuplicate($user, $image)
    {
        DB::table('notifications')
            ->whereNotifiableId($image->user->id)
            ->whereNull('read_at')
            ->where('data->image_id', $image->id)
            ->where('data->user', $user->id)
            ->delete();
    }
}
```

L’affichage des notifications

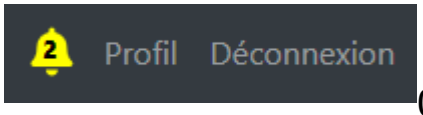
On va ajouter deux routes :

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

    ...

                                                    Route::name
('notification.')->prefix('notification')->group(function () {
                        Route::name   ('index')->get   ('/',
'NotificationController@index');
            Route::name ('update')->patch ('{notification}',
'NotificationController@update');
    });
});
```

Dans la vue **layouts.app** on ajoute ce code :

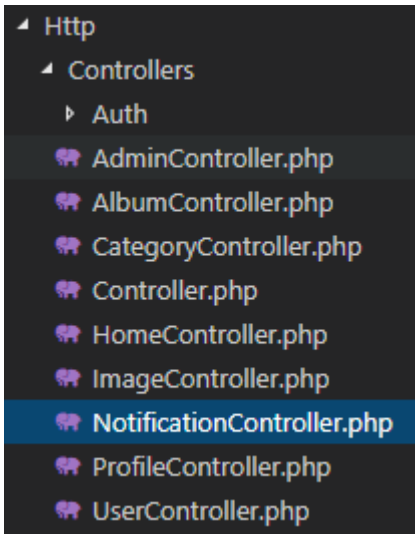
```
@endmaintenance
@unless(auth()->user()->unreadNotifications->isEmpty())
    <li class="nav-item">
        <a class="nav-link" href="{{ route('notification.index')
}}">
            <span class="fa-layers fa-fw">
                <span style="color: yellow" class="fas fa-bell fa-
lg" data-fa-transform="grow-2"></span>
                 <span class="fa-layers-text fa-inverse" data-fa-
transform="shrink-4 up-2 left-1" style="color: black; font-
weight:900">{{    auth()->user()->unreadNotifications->count()
}}</span>
            </span>
        </a>
    </li>
@endunless
```

Maintenant si on se connecte avec Dupont on a la petite icône et le nombre 2 associé :



On crée le nouveau contrôleur :

```
php artisan make:controller NotificationController
```

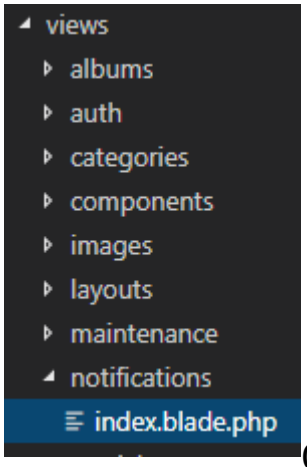


Avec ces deux méthodes :

```
<?php

namespace App\Http\Controllers;
```

```
use Illuminate\ {
    Http\Request,
    Notifications\DatabaseNotification
};

class NotificationController extends Controller
{
    public function index(Request $request)
    {
        $user = $request->user();

        return view('notifications.index', compact('user'));
    }

    public function update(Request $request, DatabaseNotification
$notification)
    {
        $notification->markAsRead();

        if($request->user()->unreadNotifications->isEmpty()) {
            return redirect()->route('home');
        }

        return back();
    }
}
```

La première (**index**) sert à afficher la page des notifications. Créons la vue :

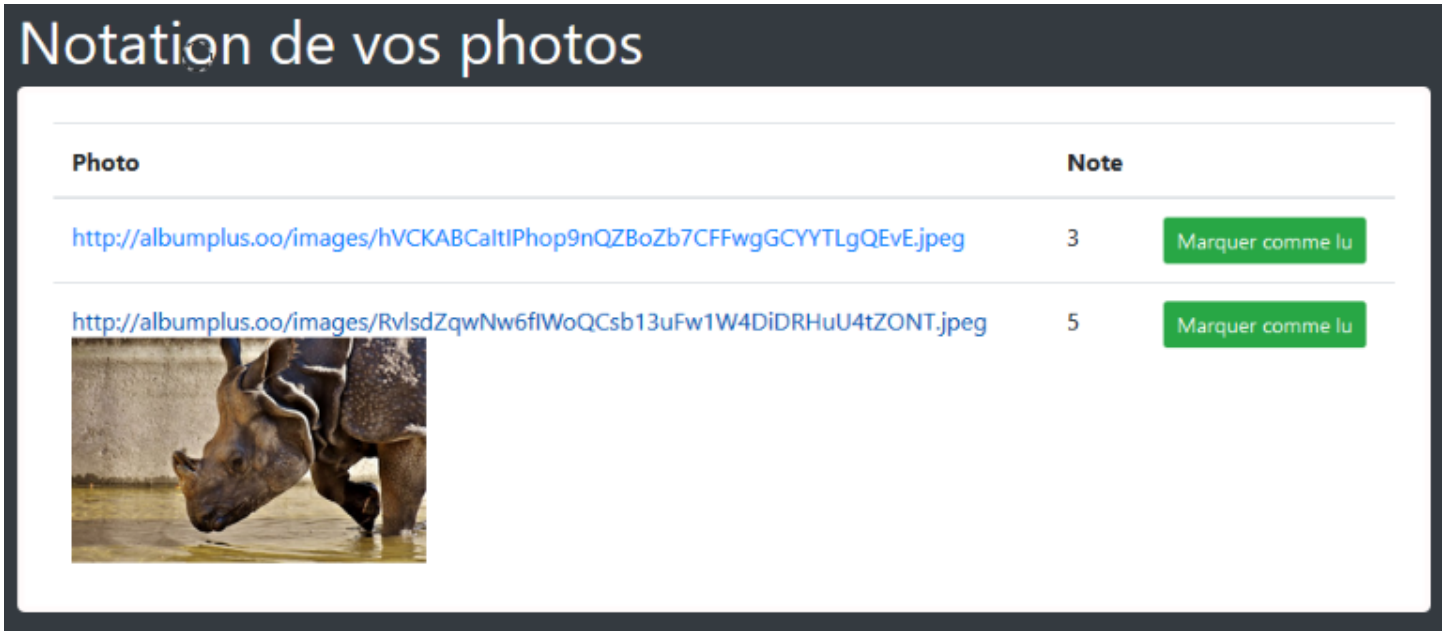


Codée ainsi :

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

@section('content')
```

```
    <main class="container-fluid">
        <h1>@lang('Notation de vos photos')</h1>
        <div class="card">
            <div class="card-body">
                <div class="table-responsive">
                     <table class="table" style="margin-bottom:
140px">
                        <thead>
                            <tr>
                                <th>@lang('Photo')</th>
                                <th>@lang('Note')</th>
                                <th></th>
                            </tr>
                        </thead>
                        <tbody>
                             @foreach ($user->unreadNotifications
as $notification)
                                <tr>
                                    <td>
                                        <div class="hover_img">
                                                <a href="{{
url('images/'   .   $notification->data['image'])   }}"
target="_blank">{{ url('images/' . $notification->data['image'])
}}<span><img src="{{ url('thumbs/' . $notification->data['image'])
}}" alt="image" height="150" /></span></a>
                                        </div>
                                    </td>
                                                <td>{{
$notification->data['rate'] }}</td>
                                    <td>
                                            <form action="{{
route('notification.update', $notification->id) }}" method="POST">
                                            @csrf
                                            @method('PATCH')
                                             <input type="submit"
class="btn btn-success btn-sm" value="@lang('Marquer comme lu')">
                                            </form>
                                    </td>
                                </tr>
                            @endforeach
                        </tbody>
                    </table>
                </div>
```

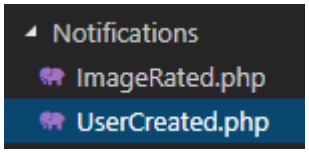
```
            </div>
        </div>
        <br>
    </main>
@endsection
```



On a le lien des photos notées ,la note, et un bouton pour dire qu'on a bien lu la notification. Le survol du lien fait apparaître une miniature de l'image.

Notification par mail

L'administrateur est averti de l'inscription d'un nouvel utilisateur par mail. On crée une nouvelle notification :



On va compléter le code :

```
<?php

namespace App\Notifications;

use App\Models\User;
use Illuminate\Bus\Queueable;
```

```
use Illuminate\Notifications\Notification;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Notifications\Messages\MailMessage;

class UserCreated extends Notification
{
    protected $user;

    public function __construct(User $user)
    {
        $this->user = $user;
    }

    public function via($notifiable)
    {
        return ['mail'];
    }

    public function toMail($notifiable)
    {
        return (new MailMessage)
                    ->subject(__('Nouvel utilisateur'))
                       ->line(__("Un nouvel utilisateur s'est
enregistré."))
                    ->line(__('Nom : ') . $this->user->name)
                    ->line(__('Email : ') . $this->user->email);
    }
}
```

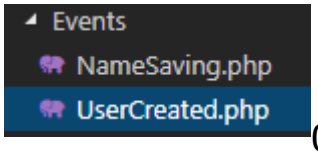
On transmet juste l'utilisateur à notifier, donc l'administrateur.

Cette fois dans la méthode **via** on précise par **mail**.

On a une méthode **toMail** pour définir ce que doit contenir le mail.

Comment savoir quand un utilisateur est créé ? Laravel va nous en informer. On commence par créer un événement :

```
php artisan make:event UserCreated
```

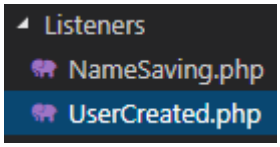


On le code ainsi :

```php
<?php

namespace App\Events;

use Illuminate\Queue\SerializesModels;
use App\Models\User;

class UserCreated
{
    use  SerializesModels;

    public $user;

    public function __construct(User $user)
    {
        $this->user = $user;
    }
}
```

On crée ensuite un listener :

```
php artisan make:listener UserCreated
```



Avec ce code :

```php
<?php

namespace App\Listeners;

use App\ {
    Events\UserCreated as UserCreatedEvent,
    Notifications\UserCreated as SendNotificationUserCreated,
    Models\User
};
use Illuminate\Support\Facades\Notification;

class UserCreated
{
    public function handle(UserCreatedEvent $event)
    {
        Notification::send(User::whereRole('admin')->first(), new
SendNotificationUserCreated($event->user));
```

```
    }
}
```

C’est ici qu’on envoie la notification, donc le mail.

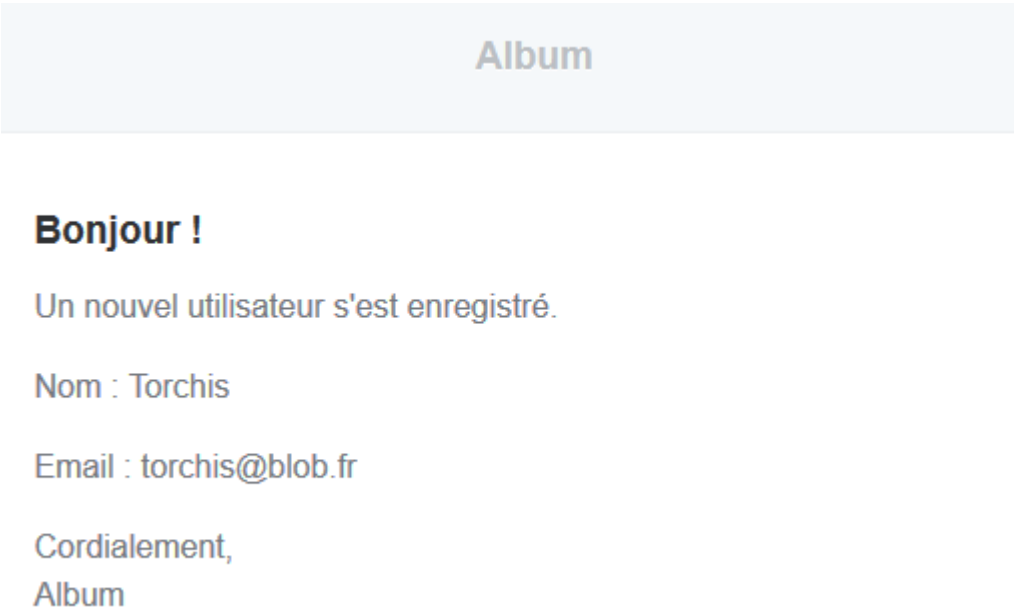
On établit la liaison entre les deux dans **EventServiceProvider** :

```
protected $listen = [
    ...

    'App\Events\UserCreated' => ['App\Listeners\UserCreated',],
];
```

Il ne reste plus qu’à déclencher l’événement, on va le faire dans le modèle **User** :

```
use App\Events\UserCreated;

    ...

protected $dispatchesEvents = [
    'created' => UserCreated::class,
];
```

Et maintenant quand un nouvel utilisateur est créé l’administrateur (ou les administrateurs) reçoit un mail :



Pour gagner en efficacité on peut établir une file d’attente (queue) parce que l’envoi d’un mail prend du temps. On ajoute ce trait dans la notification (et on implémente **ShouldQueue**) :

```
class UserCreated extends Notification implements ShouldQueue
{
    use Queueable;
```

Dans le fichier .env on choisit un driver, par exemple **redis** :

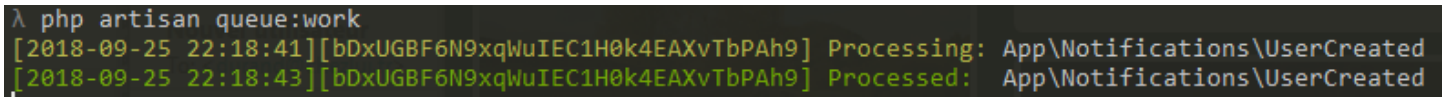
```
QUEUE_CONNECTION=redis
```

Il faut également ajouter ce package :

```
composer require predis/predis
```

Plus qu'à lancer :

```
php artisan queue:work
```



En résumé

Dans ce chapitre on a créé deux sortes de notification :

- avec la base de données pour mémoriser la notation des photos et en informer le propriétaire
- avec un mail pour avertir l'administrateur de l'inscription des nouveaux utilisateurs

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