

# Fiche PP 5 : Envoi de notifications aux utilisateurs en PHP

BTS Services informatiques aux organisations  
Session 2020-2021

E6 – Parcours de professionnalisation  
*Coefficient*

## Description d'une situation professionnelle

Épreuve ponctuelle      ~~Contrôle en cours de formation~~

~~Parcours SISR~~      Parcours SLAM

**NOM et prénom du candidat :** KERN Louis      **N° candidat :** 0319811520

### Contexte de la situation professionnelle

Dans le cadre du projet d'amélioration du système d'envoi de demandes au Master Data Manager. Un des objectifs est de pouvoir envoyer une notification au Master Data Manager lorsque l'utilisateur créé ou effectue une action sur une demande, et inversement.

### Intitulé de la situation professionnelle

Envoi de notifications aux utilisateurs en PHP

**Période de réalisation :** 3<sup>ème</sup> semestre 2021      **Lieu :** Phalsbourg

**Modalité :** ~~Equipe~~ Individuelle

### Conditions de réalisation

Implémentation d'un système de notifications.

### Productions associées

Le travail sera suivi par la Master data Manager et ses associés pour s'assurer qu'il respecte le cahier des charges, et une fois validé, il sera ensuite mis en production sur les serveurs du réseau FM Logistic.

**Compétences mises en œuvre :** A.1.1.1 et 3, A.1.2.2 à 5, A.1.3.1, A.1.4.2, A.2.1.2, A.2.2.1, A.2.3.2, A.3.2.2, A.4.1.1, A.4.1.3 et 4, A.4.1.6 et 8, A.4.2.1 et 3, A.5.2.1 et 3 et 4.

# Fiche PP 5 : Envoi de notifications aux utilisateurs en PHP

## Introduction :

Dans le cadre du projet d'amélioration du système d'envoi de demandes au Master Data Manager. Il a été demandé à ce que les utilisateurs et le Master Data Manager, puissent recevoir une notification lorsqu'une action est effectuée sur une demande les concernant (l'utilisateur en est le créateur ou bien le Master Data Manager est notifié qu'une demande a été créée par un utilisateur).

## Condition de réalisation :

Environnement de développement

Logiciels : Visual Studio Code, MySQL Workbench

Langages : PHP, HTML, CSS, SQL

## Envoi de notifications :

Lorsqu'une demande est créée ou qu'une action sur cette demande via le menu « action », lorsqu'on clique sur une demande :



The image shows a user interface with two main components. On the left is a dropdown menu titled 'Actions' with a gear icon. It contains five items: 'En cours' (with a gear icon), 'Demander une action' (with an exclamation mark icon), 'Traiter' (with a checkmark icon), 'Nouveau commentaire' (with a speech bubble icon), and 'Annuler' (with an 'x' icon). On the right is a notification panel. At the top, it says '5 Notifications' with a bell icon and 'louis kern' with a dropdown arrow. Below this is a list of five notifications, each with an icon and text: 1. Green checkmark: 'Votre demande #160 a été traitée'. 2. Red exclamation mark: 'La demande #160 requiert une action supplémentaire'. 3. Blue gear: 'Votre demande #160 est en cours de traitement'. 4. Blue speech bubble: 'La demande #160 a été commentée'. 5. Red 'x': 'La demande #159 a été annulée'. At the bottom of the notification panel is a link that says 'Marquer comme lu' with an eye icon.

Une notification est automatiquement envoyée à la personne concernée (exemple ci-dessous) :

L'utilisateur se voit indiqué le nombre de notifications non lues et, en cliquant sur le menu déroulant, voit plus précisément quelle action et sur quelle demande cette action concerne. Si l'utilisateur clique sur l'une de ces notifications, il sera redirigé vers la

## Fiche PP 5 : Envoi de notifications aux utilisateurs en PHP

demande en question. Enfin, le bouton « marquer comme lu » permet, comme son nom l'indique, de marquer comme lu, toutes les notifications présentes dans le menu.

La méthode permettant de marquer comme lu toutes les notifications se présente comme ceci :

```
protected function readall()
{
    foreach (Auth::user()->unreadNotifications as $notif)
    {
        $notif->markAsRead();
    }
    return Redirect::back()
        ->with('success', 'Notifications marquées comme lues !');
}
```

On identifie toutes les notifications non lues concernant l'utilisateur connecté actuellement, c'est-à-dire, les notifications ayant, en base de données, le champ read\_at null comme ceci :

type	notifiable_type	notifiable_id	data	read_at
App\Notifications\Cree	App\User	1	{"demande":155,"icone":"fas fa-plus-circle icon-...	2021-01-29 15:35:59
App\Notifications\annulee	App\User	13	{"demande":155,"icone":"fas fa-times-circle ic...	2021-01-29 16:08:46
App\Notifications\nouveauComm	App\User	13	{"demande":155,"icone":"fas fa-comment-dot...	2021-01-29 15:45:23
App\Notifications\nouveauComm	App\User	18	{"demande":160,"icone":"fas fa-comment-dot...	NULL
App\Notifications\demandeAction	App\User	13	{"demande":155,"icone":"fas fa-exclamation-...	2021-01-29 15:45:23
App\Notifications\nouveauComm	App\User	13	{"demande":155,"icone":"fas fa-comment-dot...	2021-01-29 15:47:06
App\Notifications\enCoursTraitement	App\User	13	{"demande":156,"icone":"fas fa-cogs icon-inf...	NULL
App\Notifications\nouveauComm	App\User	13	{"demande":155,"icone":"fas fa-comment-dot...	2021-01-29 15:51:09
App\Notifications\demandeAction	App\User	18	{"demande":160,"icone":"fas fa-exclamation-...	NULL
App\Notifications\Cree	App\User	1	{"demande":159,"icone":"fas fa-plus-circle icon-...	2021-02-01 11:27:43
App\Notifications\annulee	App\User	1	{"demande":158,"icone":"fas fa-times-circle ic...	2021-02-01 11:27:14
App\Notifications\nouveauComm	App\User	1	{"demande":158,"icone":"fas fa-comment-dot...	2021-02-01 11:23:08
App\Notifications\demandeAction	App\User	13	{"demande":155,"icone":"fas fa-exclamation-...	2021-01-29 15:45:23
App\Notifications\enCoursTraitement	App\User	18	{"demande":160,"icone":"fas fa-cogs icon-inf...	NULL
App\Notifications\demandeTraitee	App\User	18	{"demande":160,"icone":"fas fa-check-circle ic...	NULL

On remarque qu'à droite, certaines notifications n'ont, en effet pas été lues (read\_at espectral étant null). Elles vont donc être récupérées dans « unreadNotifications » et donc pour chaque notification non lue, on va les marquer comme lues (en entrant la date de lecture dans le champ read\_at).

Enfin, on affiche un petit message indiquant le bon déroulement de l'opération.

Pour ce qui est du traitement des notifications, nous utilisons la méthode changeState du contrôleur RequestController qui prend en paramètre d'entrée « request » qui nous permet d'accéder aux éléments de la requête SQL avant qu'elle ne soit exécutée en base de données et « id » qui est l'identifiant de la demande.

## Fiche PP 5 : Envoi de notifications aux utilisateurs en PHP

```
$user = App\User::find($request->demandeur);
$acteur = App\User::find($request->acteurid);
switch ($request->selEtat)
{
    case 'Traitée':
        $user->notify(new demandeTraitee($id, $request->commentaire));
        break;

    case 'Commentaire':
        if($acteur->isMDM()){
            $user->notify(new nouveauComm($id, $request->commentaire));
        } else {
            foreach (User::where('role', 1)
                ->get() as $MDM)
            {
                $MDM->notify(new nouveauComm($id, $request->commentaire));
            }
        }

        break;

    case 'En Attente':
        foreach (User::where('role', 1)
            ->get() as $MDM)
        {
            $MDM->notify(new reponseApportee($id, $request->commentaire));
        }
        break;

    case 'Annulée':
        if($acteur->isMDM()){
            $user->notify(new annulee($id, $request->commentaire));
        } else {
```

On récupère l'utilisateur via request -> demandeur, qui va récupérer dans la requête, le nom du demandeur dans le champ du même nom, et ensuite, on récupère son identifiant dans « acteur ».

En utilisant un switch on peut gérer toutes les actions possibles en fonction du type d'acteur, par exemple pour le cas « commentaire », on vérifie d'abord si l'acteur est un Master Data Manager (MDM), si c'est le cas, on notifie l'utilisateur que le MDM a commenté sa demande (en créant une nouvelle instance de « nouveauComm » comprenant l'id de la demande, et le contenu du commentaire). Si l'acteur est un

## Fiche PP 5 : Envoi de notifications aux utilisateurs en PHP

utilisateur, à ce moment-là, on envoie une notification à tous les MDM (où le rôle en base de données est à 1, 0 étant un utilisateur lambda).

Chaque type d'action a son objet correspondant, par exemple l'objet « demandeAction » :

```
public function toArray($notifiable)
{
    return [
        'demande' => $this->demande,
        'icone' => "fas fa-exclamation-circle icon-danger",
        'texte' => "La demande #" . $this->demande . " requiert une action supplémentaire"
    ];
}
```

Dans la méthode toArray, on construit la forme finale de la notification (ce que l'on voit dans le menu « notifications », avec, comme icône, la référence de l'icône venant de la librairie d'images gratuite FontAwesome et avec les indications de couleurs de Bootstrap (icon-danger dans notre exemple qui correspond à du rouge). Ensuite, nous ajoutons le message qui l'accompagne avec le numéro de la demande spécifiée.

Enfin, sur chaque demande, il y a un affichage de l'historique des actions effectués sur la demande :

```
<h5 class="FirstFont">Historique</h5>
@foreach($request->histories as $hist)
<div class="row">
    <div class="col-12 col-md-8 card cardSq p-2 mb-2">
        <div class="row">
            <div class="col" style="color: grey">
                @if($hist->acteur == Auth::user()->prenom." ".Auth::user()->nom )
                <i class="fas fa-user-clock mr-1 icon-success"></i><b>{{ $hist->acteur }}</b>
                @else
                <i class="fas fa-user-clock mr-1 icon-info"></i><b>{{ $hist->acteur }}</b>
                @endif
            </div>
            <div class="col text-right" style="color: grey">
                {{ $hist->created_at->format('d/m/Y H:i')}}
            </div>
        </div>
        <div class="row">
            <div class="col-12">
                <hr class="mt-1 mb-1" />
                @if($hist->hist_id == 0)
                <span class="text-success" style="width: 100%; font-size: 120%">Création de la demande </span>
                @else
                @switch($hist->etat)
                @case("En Cours")
                <span class="text-warning" style="width: 100%; font-size: 120%"> Demande en cours de traitement</b> </span>
                @break
                @case("Action Requise")
                <span class="text-danger" style="width: 100%; font-size: 120%"> Action du demandeur requise</b> </span>
                @break
                @case("Traitée")
                <span class="text-success" style="width: 100%; font-size: 120%"> Demande traitée</b> </span>
                @break
                @case("Annulée")
                <span class="text-danger" style="width: 100%; font-size: 120%"> Demande annulée</b> </span>
                @break
            </div>
        </div>
    </div>
</div>
```




## Fiche PP 5 : Envoi de notifications aux utilisateurs en PHP

On récupère le nom de la personne ayant effectué le changement, et, en fonction du type d'action, on affiche le message dans la bonne couleur ainsi que le message correspondant à l'action.

```
<p class="text-muted">{{$hist->commentaire}}</p>
@if(is_null($hist->fichier)==FALSE)
<p style="font-size:100%"><a class="badge badge-light" href="{{$request->id}}/{{$hist->hist_id}}/fichier">
  <i class="fas fa-file-download mr-1"></i>
  télécharger {{preg_replace('(.*/(\d+_)\d+)', '', $hist->fichier)}}
</a></p>
@endif
```

On affiche aussi les commentaires s'il y en a (première ligne de l'image), et, si il y a un fichier qui l'accompagne, on récupère son nom et son chemin dans le stockage interne, et on propose la possibilité de le télécharger.

Le rendu final ressemble à ceci :

 MDM Admin	04/02/2021 11:29
Demande traitée	
 MDM Admin	04/02/2021 11:29
Action du demandeur requise besoin de précisions	
 MDM Admin	04/02/2021 11:28
Demande en cours de traitement	
 MDM Admin	04/02/2021 11:28
Nouveau commentaire bonjour	

### Conclusion :

Ce stage a été très enrichissant de par l'utilisation d'éléments nouveau pour moi, dont le traitement des notifications en PHP et l'utilisation du Framework Laravel. Aussi, le fait de travailler sur un outil répondant aux besoins d'un « client », qui au fur et à mesure de l'avancement du projet, affine ses besoins, est très intéressant d'un point de vue relationnel et de conduite de projet.