**Les Notifications**

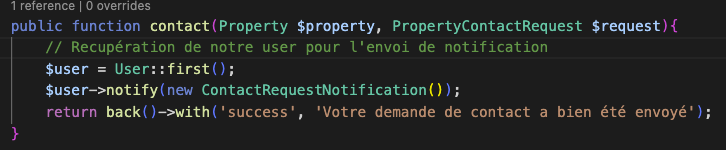
Dans cette partie nous verrons comment envoyer les **notifications** à travers plusieurs **canaux** (sms, slack, comme websockets), ce qui n’est pas possible avec le **maillable**.

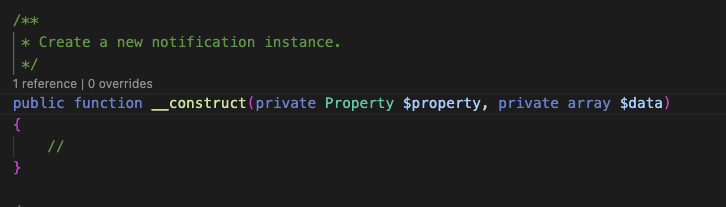
**Création:**

**php artisan make:notification ContactRequestNotification**

Nous avons la méthode **toArray()** qui va nous permettre de sérialiser notre notification pour d'autres canaux, par exemple la sauvegarde en base de données.

Dans notre cas, nous allons considérer le fait que nous voulions avertir l’utilisateur dès qu’un changement est fait au niveau du bien. Nous ferons cet essai avec notre premier utilisateur qui est l’admin.

Dans notre **PropertyController,** nous allons reprendre la logique de notre application pour l’adapter à notre système de notification.

Dans notre **ContactRequestNotification,**  nous allons tout comme dans le maillable implémenter notre constructeur de telle sorte qu’il reçoive non seulement le bien mais aussi le tableau d’informations qui seront entrées.

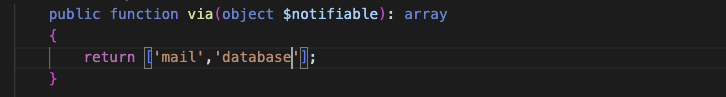
De ce fait dans notre **PropertyController**, nous aurons :



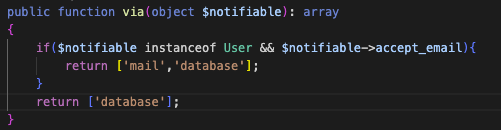
Autre précision, dans notre méthode **toMail(),** nous remarquons qu’il crée une instance de **MailMessage;**  On pourrait tout à fait changer cela par notre **PropertyContactMail** pour avoir le content selon notre class de mail en lui passant les bonnes informations, mais cela ne change pas grande chose.

**NB:** Quel que soit le canal, nous allons avoir une méthode qui va nous permettre de convertir notre notification en quelque chose d’exploitable.

Nous allons voir un exemple avec le canal **database.**

Dans notre **ContactRequestNotification,** nous allons rajouter **database** à la méthode **via()**

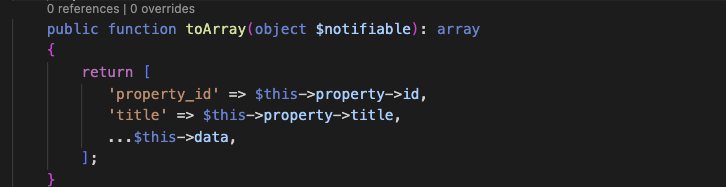
Nous pouvons également faire une vérification bien avant l’envoi.



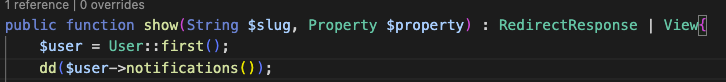
Pour les bases de données, nous allons générer une table qui va contenir les notifications.

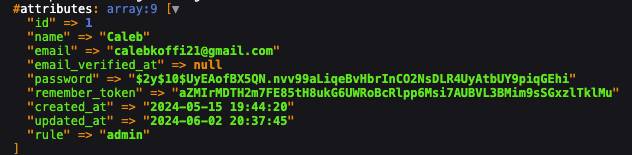
**php artisan notifications:table** puis **php artisan migrate**

Comme constaté dans notre **db** nous avons une partie **$data** qui va recevoir les données de notre notification qui sera la représentation sous forme de tableau c’est à dire le retour de notre function **toArray()** se trouvant dans notre **ContactRequestNotification;** on a de même la possibilité de créer une fonction **toDatabase,** si nous voulons un autre formatage des notifications lors de l’introduction dans la base de données.

Donc dans notre cas, nous aurrons besoin de l’**id**, du **title** et des **data.**

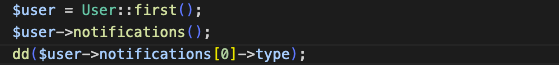
l’object $notifiable en paramètre de notre fonction **toArray()** concerne le sujet sur lequel nous envoyons la notification. nous typiquement, c’est l’utilisateur qui est censé recevoir la notification.

Cela enverra non seulement la notification mais aussi insérera les informations dans la base de données. Du coup quand nous voulons accéder à ces informations, dans notre **PropertyController,** dans la méthode show, nous pourrons récupérer de nouveau notre utilisateur et appeler la méthode **notification()** qui affichera toutes les notifications liées à l’utilisateur.



On peut également renvoyer les notifications non lues avec la fonction **unreadNotifications();** ou encore on peut marquer une notification comme lue comme suit:

Exemple: on recupère notre prémière notification pour la marquer comme lue: 

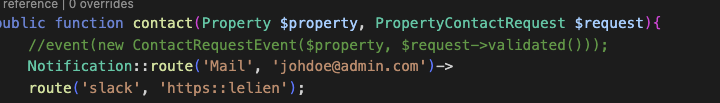
On peut également se dire que l’on va récupérer toutes les notifcations de l’utilisateur pour ensuite les afficher dans un système centralisé de notification. De ce fait on peut boucler là-dessus pour ensuite modifier le texte en fonction du type 



Cela est très intéressant dans le cas d’une sorte de réseaux sociaux dans lesquels on peut envoyer des notifications à plus étapes et ensuite faire un centre de notification pour l’utilisateur afin de voir tout ce qui s’est passé.

***info:*** *Il serait très intéressant de vérifier les implémentations pour chaque canal; on remarque également que ce système de notification à été beaucoup penser pour les utilisateurs, on peut le voir même dans la base de données, il y’a* ***notifiable\_id*** *qui est l’id de l’user appelé ou encore par le* ***notifiable\_type****. On pourrait parcontre utiliser ce système sur autre chose que l’utilisateur.*

**NB:** on pourrait également utiliser la façade **Notification** pour envoyer des emails de façon arbitraire en utilisant la méthode **route()** tout en lui précisant les différents canaux de diffusion

Exemple:

Et ainsi de suite en fonction des canaux.

Dans notre cas, vu que nous sommes avec les mails, nous allons le faire dessus.



De cette sorte nous faisons manuellement les choses car nous ne choisissons pas un utilisateur pour cela.

### **Pourquoi Utiliser Notification::route ?**

* **Cas d'usage spécifique** : Lorsque vous avez besoin d'envoyer une notification à une adresse email, un numéro de téléphone ou un identifiant de canal Slack spécifique qui n'est pas directement lié à un modèle utilisateur ou notifiable.
* **Flexibilité** : Vous pouvez envoyer des notifications sans avoir besoin de modifier ou de créer des relations supplémentaires dans vos modèles.
* **Simplicité** : Cela simplifie le processus d'envoi de notifications dans des cas spécifiques et exceptionnels.

Ainsi, Notification::route est une méthode très puissante et flexible pour gérer des cas particuliers d'envoi de notifications dans Laravel.

**Cas Particulier:**

Au cas où nous voulons envoyer les informations au propriétaire d’un bien par exemple pour l’aviser d’une quelconque information concernant son bien, nous pouvons le faire en établissant une relation entre l’article et l’utilisateur en question dans le model **Article**:

// Relation belongsTo entre Article et User

**public function user() {**

**return $this->belongsTo(User::class);**

**}**

Ensuite, dans notre **ArticleController** nous envoyons la notification comme suit et pas besoin de recupérer l’utilisateur.

**// Notifier l'utilisateur propriétaire de l'article**

**$article->user->notify(new ArticleContactNotification($article, $message));**

### **Explication Finale**

* **Expéditeur de l'email** : Le champ MAIL\_USERNAME dans votre .env (par exemple, your-email@yourdomain.com) est l'adresse email utilisée pour envoyer l'email via le service de messagerie configuré.
* **Destinataire de l'email** : Le destinataire est l'email de l'utilisateur propriétaire de l'article, récupéré dynamiquement grâce à la relation définie dans le modèle Article. Laravel utilise automatiquement l'email du notifiable(l'utilisateur dans ce cas) pour envoyer l'email.