Kern Louis

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 SLAM

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

Contexte de la situation professionnelle

FM Logistics cherche à améliorer son système de tickets d'incidents, ils cherchent a pouvoir passer d'un stockage en format Excel à une vérifications du formats de données via un formulaire php. Avec un affichage sous forme d'interface graphique de ses tickets

Intitulé de la situation professionnelle

Développement d'un système de traitements de données en PHP

Période de réalisation : 3^{ème} semestre 2021 Lieu : Phalsbourg

Modalité: Equipe Individuelle

Conditions de réalisation

Finalisation du programme de gestion des demandes liées au Master Data de notre outil de Transport Mettre en place une restitution graphique des logs traitement jobs Talend

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 logitic.

Modalité d'accès aux productions :

Kern Louis

Introduction:

La Master Data Manager cherche à optimiser et faciliter son travail au sein de l'entreprise, en effet, elle reçoit tous les jours des requêtes et diverses demandes venant des clients, mais le problème est que souvent, les informations sont incomplètes, erronées ou tout simplement au mauvais format, cet outil à donc pour but de remédier à ce problème en normalisant le format des données à envoyer et à permettre au MDM de pouvoir gérer plus facilement les demandes qui lui sont adressé.

Mon objectif est de finir un projet de formulation de demandes envers un Master Data Manager (MDM) et de d'en améliorer l'ergonomie ainsi que d'ajouter des fonctionnalités supplémentaires.

Le but étant de faire régulièrement des points avec la Master Data Manager afin de s'assurer du bon format des demandes, que les données demandées soient claires et ordonnées afin de faciliter le travail du Master Data Manager, en effet ce site ne sert qu'à formuler at à afficher de manière lisible des demandes, le traitement est effectué ensuite par la Master Data Manager.

Un autre défi du projet et de devoir gérer les accès et possibilités d'action du côté client mais aussi du côté MDM, par exemples, un client peut formuler une demande et performer quelques actions dessus, en revanche, le MDM ne peut que les consulter, mais peut performer plus d'actions dessus.

Beaucoup de fonctionnalités du site sont affecté par rapport à quel type d'utilisateur est connecté (client ou MDM).

Condition de réalisation :

Environnement de développement

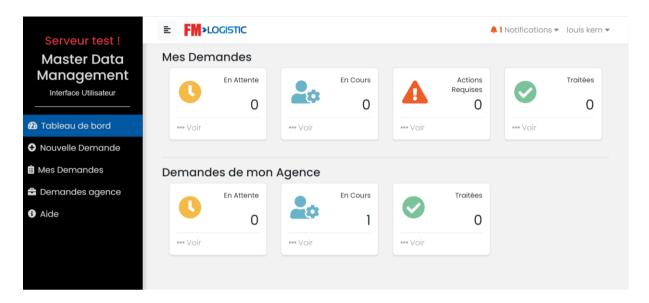
Logiciels: Visual Studio Code, MySQL Workbench

Langages: PHP, HTML, CSS, SQL

Kern Louis

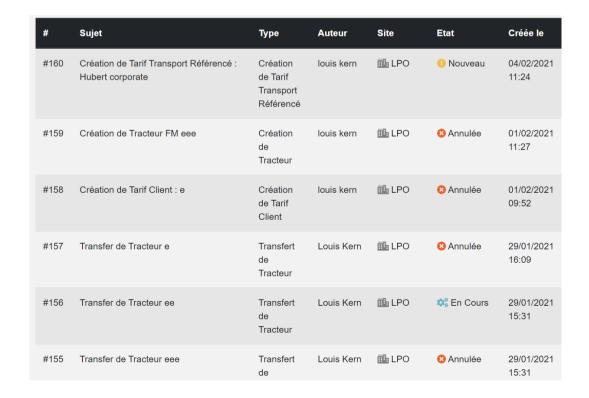
Formulaires de demandes :

Le travail le plus important de ce projet est la présentation, le contenu, le format et l'affichage des différentes demandes envers la Master Data Manager, pour illustrer le processus, nous allons nous connecter en tant que client :



Nous sommes sur la page tableau de bords, contenant une vue globale des différentes demandes déjà faite par l'utilisateur et son agence et leur état actuel.

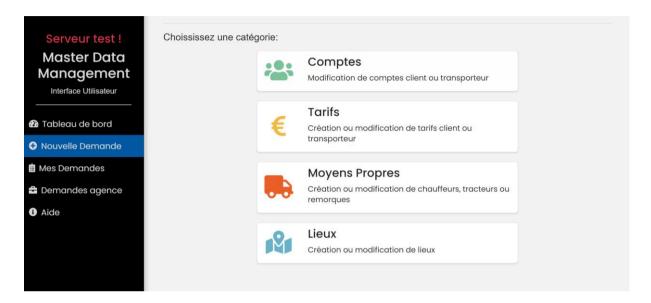
En cliquant soit sur « mes demandes », soit « Demande agence » l'utilisateur arrive sur cette fenêtre :



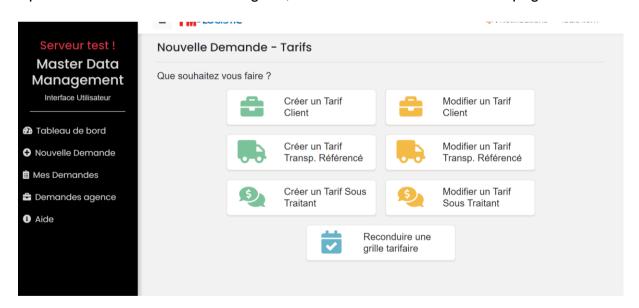
Kern Louis

Ceci affiche le listing de toutes les demandes avec leur numéro, leur sujet, type, auteur, etc...

Quand l'utilisateur veut formuler une nouvelle demande, il clique sur « Nouvelle demande » et se retrouve sur cette page :



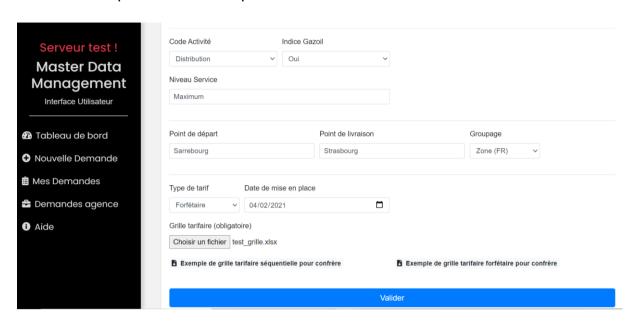
Après avoir sélectionné une catégorie, il se retrouve ensuite sur cette page :



Depuis là il peut choisir quel formulaire de demande l'utilisateur veut remplir. Il est important de noter que les modifications de demandes créées de nouvelles demandes au même titre que les demandes de créations, la différence étant que le sujet est différent de leur homologue « création », et que seule une information permettant d'identifier quel élément sera affecté par le MDM et les informations à modifier sont demandés (le formulaire peut être envoyé incomplet).

Kern Louis

Voici un exemple de formulaire pour une création de Tarif Client référencé :



Une fois le formulaire remplis, lors de l'envois on récupère le nom du formulaire (ex creaTC) et on construit le sujet de la demande avant d'envoyer en base de données le formulaire avec ses données.

Nom du formulaire dans un input nommé « type » caché, le but est de pouvoir récupérer le nom du « type » de formulaire (dans cet exemple « t_creaTC ») avec t_ qui indique que c'est un formulaire de la catégorie « tarif », « crea » qui indique que c'est un formulaire de création et, enfin, « TC » qui est les initiales de « Tarif Client », Avec cette nomenclature, nous pouvons facilement identifier de quel formulaire est utilisé et pouvoir construire le sujet grâce à cette information.

Kern Louis

On récupère ensuite la valeur des inputs (cette image montre un code venant d'un autre formulaire) avec chacun leur tag propre.

```
$req->c_searchname = $request->input('f_searchname');
$req->c typeclient = $request->input('f typeclient');
$req->c chez = $request->input('f chez');
$req->c_pour = $request->input('f_pour');
                                                                  use Illuminate\Database\Eloquent\Model;
$req->c_codegestion = $request->input('f_codegestion');
                                                                  class Request extends Model
$req->c_adresse = $request->input('f_adresse');
$req->c_codepostal = $request->input('f_codepostal');
                                                                    protected $fillable = [
$req->c_ville = $request->input('f_ville');
$req->c_pays = $request->input('f_pays');
$req->c_tva = $request->input('f_tva');
$req->c_telephone = $request->input('f_tel');
                                                                     'c_searchname',
$req->c_fax = $request->input('f_fax');
                                                                      'c_typeclient',
$req->c_mail = $request->input('f_mail');
                                                                      'c_chez',
$req->t_typemodif = $request->input('f_typemodif');
                                                                      'c_codegestion',
$req->t_codeactivite = $request->input('f_codeactivite');
$req->t_niveauservice = $request->input('f_niveauservice');
                                                                      'c_codepostal',
$req->t_indicegazole = $request->input('f_indicegazole');
                                                                      'c_pays',
$req->t_depart = $request->input('f_depart');
$req->t livraison = $request->input('f livraison');
                                                                      'c_telephone',
$req->t_groupage = $request->input('f_groupage');
                                                                      'c_mail',
$req->t_type = $request->input('f_type');
                                                                      't_codeactivite',
$req->t miseenplace = $request->input('f miseenplace');
                                                                      't_indicegazole',
```

Dans un fichier RequestController (à gauche), qui s'occupe de formater les données avant l'envoie dans la base de données, on récupère toutes données possibles venant d'un formulaire, le programme ne fait pas de distinction, les données manquantes seront simplement *null*. On envoie les variables dans le fillable du RequestBuilder. Puis ils les postent en base de données.

Kern Louis

Mais avant ça on doit encore construire le sujet de la demande :

```
public static function typeName($type)
   switch ($type)
       case 'l_creaL':
          return 'Création de Lieu';
       case 'l_modifL':
          return 'Modification de Lieu';
       case 'c_modifCT':
          return 'Modification de Client Transporteur';
       case 'c_modifRC':
          return 'Modification ou validation de relation client'
       case 'c_modifT':
          return 'Modification de Transporteur';
       case 't creaTC':
          return 'Création de Tarif Client';
       case 't_creaTST':
          return 'Création de Tarif Sous-Traitant';
       case 't_creaTTR':
          return 'Création de Tarif Transport Référencé';
       case 't_modifTC':
          return 'Modification de Tarif Client';
       case 't_modifTST':
          return 'Modification de Sous-Traitant';
```

La fonction typeName permet de renvoyer des messages prédéfinis en fonction du nom du type de formulaire passé en paramètre, on appelle cette fonction dans save pour pouvoir construire le sujet de la demande.

En fonction du type de formulaire on construit dans la fonction save le sujet par rapport au type de formulaire utilisé et on récupère la valeur qu'a retourné typeName pour le formulaire passé en paramètre ainsi que les valeurs désirées des formulaires.

```
protected function save(Request $request)
   $rea = new Rea:
   $req->etat = 'En Attente';
   $req->auteur = $request->input('auteur');
   $req->type = $request->input('type');
   switch ($request->input('type'))
       case "c_modifT":
          $req->sujet = $this->typeName($request->input('type')) . " : " . $request->input('f_searchname');
       break;
           $req->sujet = $this->typeName($request->input('type')) . " : " . $request->input('f_searchname');
       break;
       case "c_modifRC":
           if ($request->input('f_typemodif') == "Modification")
               $req->sujet = "Modification de relation client" . " : " . $request->input('f_searchname');;
               $req->sujet = "Validation de relation client" . " : " . $request->input('f_searchname');;
       break;
```

Kern Louis

Ensuite on ajoute les données spécifiques à la demande comme le nom de l'élément affecté puis on l'ajoute dans fillable. Si une grille tarifaire est présente on l'enregistre dans les fichiers du site, dans le cas d'un renvoi de grille via l'historique on lui attribue un nom spécial et un ld propre pour éviter les doublons de grilles sur la même demande et pouvoir tracer quelle grille correspond à quelle action :

```
protected function downloadGrille($id)
{
    $req = Req::findOrFail($id);
    $path = 'app/' . $req->t_grille;
    return response()
        ->download(storage_path($path));
}

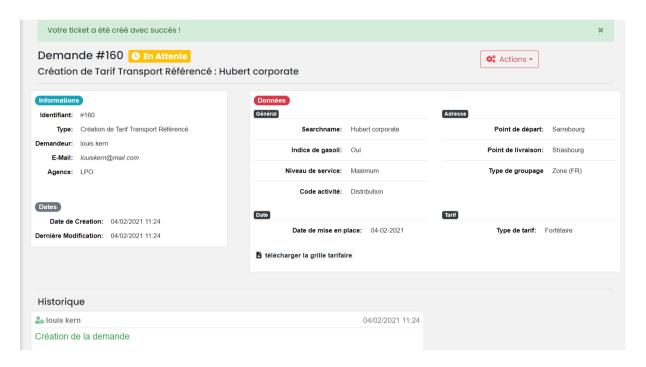
protected function downloadHistfile($id, $hist_id)
{
    $hist = Hist::where('req_id', $id)->where('hist_id', $hist_id)->first();
    $name = preg_replace('(.*\/(\d+_\d+_\)?)', '', $hist->fichier);
    $path = 'app/' . $hist->fichier;
    return response()
        ->download(storage_path($path) , $name);
}

protected function downloadExempleGrille($name_file)
{
    $path = 'app/public/exemples_grilles/' . $name_file . '.xlsx';
    echo $path;
    return response()->download(storage_path($path));
}
```

Puis on récupère le nom du fichier et on y ajoute le numéro de la demande associé pour éviter les doublons si jamais une grille correspondant à une autre demande ayant le même nom serait amené à être uploadé sur le site :

Kern Louis

Une fois la demande créée on l'affiche dans une nouvelle fenêtre :



Les informations sont rangées en fonction du type de données, avec les données propres à l'utilisateur ayant fait la demande étant affiché à droite et les données relatives à la demande en elle-même sont dans leur catégorie à droite. On inclut aussi un historique permettant de tracer les différentes actions et/ou commentaires mené sur la demande. Il sert aussi en quelque sorte d'espace de chat.

Ceci conclus le processus de création de demande.

Conclusion:

Ce PP, m'a permis d'approfondir mes connaissances en PHP, et surtout de s'initier à l'utilisation d'un Framework, spécialement du Framework Laravel. Mais aussi d'avoir un projet concret avec un suivi du « client » de manière régulière en planifiant des réunions, poser des questions, répondre de manière pertinente aux demandes.