# Dossier de Soutenance - Projet Elyne

## Introduction

Le Projet Elyne est destiné à organiser le transport de marchandises. Initialement, le module avait été conçu pour l'enlèvement de déchets d'un chantier A à un chantier B. Lors de la prise en main du projet, l'utilisateur principal du module Elyne, intégré à Dolibarr, était le transporteur.

Une évolution est envisagée afin que l'utilisateur du module devienne le donneur d'ordre, c'est-à-dire le tiers demandant un transport.

## I. Diagramme d'activité

Un diagramme d'activité a été présenté à Antoine Nguyen afin de vérifier la compréhension de la mécanique globale du module. Certaines étapes ont été simplifiées.

Suite à ce diagramme, nous avons décidé que :

* Un lieu est toujours un projet.
* Le lieu d’enlèvement correspond à un projet distinct du lieu de destination.
* Il est donc nécessaire de créer deux projets par demande de transport.
* Chaque projet est relié à un tiers, qui peut être le même ou un tiers différent.

### Exemple :

J.G Maçonnerie demande l’enlèvement de marchandise en lieu A vers un lieu B.

* Projet A : Tiers = J.G Maçonnerie, Lieu = J.G Maçonnerie Laval.
* Projet B : Tiers = Déchetteries de France, Lieu = Déchetterie de Laval.

## II. Simplification du module

Un second diagramme a été établi pour visualiser la simplification du processus.

## III. Réunion client et réorganisation du développement

Suite à la réunion avec le client, les délais ont été clarifiés.  
Nous avons décidé de répartir le projet entre deux développeurs, avec un délai de 2 mois pour finaliser l’application. Ce délai sera bien allongé par la suite.

Ma mission consiste dans les deux premières semaines à préparer l’application tablette destinée aux chauffeurs.

### 1. Maquette de l’application

J’ai conçu la maquette afin de visualiser les composants nécessaires. Le workflow d’une lettre de voiture comprend 4 étapes principales :

1. Commande existante mais non assignée à un chauffeur.
2. Assignée à un chauffeur.
3. Marchandise chargée.
4. Marchandise en transport.
5. Marchandise déchargée, commande clôturée.

Certains composants de la lettre de voiture sont récurrents, parfois éditables et parfois en lecture seule. Il est donc nécessaire de créer des composants modulables selon le contexte.

De plus, un besoin d’héritage se fait ressentir, car le destinataire et l’expéditeur de la marchandise auront exactement les mêmes interventions :

* Inscription des coordonnées.
* Signature pour officialiser la lettre de voiture.

A l’observation des éléments composants une voiture, un véhicule porteur et une remorque possèdent exactement les mêmes attributs.

Ce document tripartite implique trois intervenants ayant des rôles différents mais agissant ensemble pour produire un document final unique.

### 2- **Établissement d’un diagramme de classe et définition de l’architecture du projet**

Une fois les composants schématisés et les éléments qu’ils doivent contenir définis, j’ai créé un diagramme de classe afin de mettre en place une architecture orientée composants, comprenant les couches suivantes : Model, Vue, Repository, Controller et Composants. Le dossier Composants fait appel aux repositories pour générer le template final nécessaire à l'affichage.

* Model (Entités) : Représente les objets métier et la structure des données.
* Vue (Pages HTML injectant les composants) : Gère l'affichage des informations en utilisant les composants générés.
* Composants (Construction du template final) :
  + Lists : Permet de boucler et d'afficher des cartes ou éléments sous forme de liste.
  + Card : Représente une carte ou un élément individuel dans la liste.
  + Boutons : Gère les boutons et interactions utilisateurs.
  + Header et Footer : Gère l'entête et le pied de page des templates.
* Controllers (Utilisation du repository) : Prépare et passe les données aux vues en interagissant avec les repositories.
* Repository (Requêtes SQL utilisant les accesseurs des modèles) : Gère les accès aux données, interroge la base de données et retourne les résultats nécessaires aux controllers.

La nécessité d’utiliser des composants modulaires (comme en React) et la volonté de bien structurer le code m’ont conduit à choisir cette architecture. Travaillant dans un environnement Dolibarr, qui limite l'implémentation de frameworks et de bibliothèques modernes, j’ai opté pour cette approche afin de faciliter la prise en main par le développeur qui continuera le projet une fois ma formation terminée.Ne sachant pas encore comment l’application pour les chauffeurs allait être « branchée » à Dolibarr, comment les Classes allaient être représentées et manipulées dans les tables Dolibarr dont l’environement est plutôt restrictif.

L’implémentation du diagramme de classe a mené à quelques **questions** notamment la désignation des classes pour les éléments constitutifs de la lettre de voiture. En effet ces différents éléments sont visibles dans l’application initiale parfois sous le nom de « commande » , parfois sous le nom de « missions » pour le point de vue chauffeur et parfois sous le nom de « chantier ».

Cela a également mené à réfléxion sur la relation entre la « mission » et l’entité « location » qui est nécessaire tant pour le lieu de chargement que le lieu de déchargement.

3-Maquette de l’appli côté Dolibarr

a) Worfklow côté Dolibarr natif

Le client a validé son souhait de signer le contrat avec PandaERP. Les besoins commencent à se préciser. Il souhaite un style épuré pour l’interface Dolibarr et certains mécanismes doivent changer par rapport à ce qui avait été pensé au début.  
Un Workflow Dolibarr Standard est :

1. Devis

2. Commande

3. Facture

Le besoin final du client est d’éditer des lettres de voitures. Les transports qu’il planifie sont très souvent un « lot » de transports tous similaires, qu’il duplique autant de fois nécessaire et planifie par la suite, camion par camion.

Ce « lot » sera appelé un « Puits de commandes ».

Le donneur d’ordre commande donc X transports, d’un point A à un point B.  
Cette première étape correspond à une Commande Dolibarr .

La commande contiendra notamment le « Tiers » expéditeur et le « Tiers » destinataire.  
Une commande qui possède une ligne avec le tarif d’un transport et la quantité de transports commandés.  
Après validation de la commande, les étapes non-natives à Dolibarr vont être insérées dans le Workflow.

b) Worfklow **de** Dolibarr **custom**  
Ce qui a été décidé :

1. Devis

2. Commande

3. Puits de Commande

4. Lettres de voiture

Après validation de la commande, le puits de commande va contenir les lignes de produits transportés. Un puits de commande récapitulera le point d’enlèvement, le point de déchargement, la quantité de transports a effectuer qui se décrémentera au fur et à mesure de l’avancement de ceux-ci et surtout le détail de chaque produit, y compris leur spécificités (Produits dangereux par exemple).

C’est à partir de ce puit de commande que l’attribution des transports au trio chauffeur-remorque-véhicule porteur se fera.  
C’est cet objet qui est le dernier relié aux lettres de voitures.  
L’objet puit de commande est branché par des « objets liés » à toutes les lettres de voitures qui découleront des transports dans le puits.

L’objet lettre de voiture :

L’interface Dolibarr va recevoir en temps direct toutes les informations sur la vie de la lettre de voiture, à tous ses stades (non attribué, attribué mais non commencé, en cours de chargement, chargé, en cours de déchargement, terminé)  
La « Card » Lettre de voiture de Dolibarr représentera toutes les informations contenues par la Lettre de voiture y compris les fichiers joints (signatures et photos).  
Le PDF de l’objet sera, comme dans chaque Objet Dolibarr, inclus dans la Card.

A partir des informations la maquette du Workflow est entammée afin de pouvoir revenir vers le client et faire une proposition visuelle de sa future application.