



## Projet Art-Manager

Rapport de stage

-

Jean-Baptiste PRIAM MASSAT

-

Du

08 avril 2013

au

14 juin 2013

-

Développeur Logiciel

-

Bordeaux

-

Tuteur de stage :

Mr. Alain Andre

-

# Remerciements

Étant donné que ma période de stage s'est déroulée dans le cadre d'un projet libre et autonome mené par Mr. Alain Andre, mes remerciements convergent principalement vers ce dernier.

Je tiens à lui adresser une grande gratitude quant à sa patience, sa disponibilité ,sa générosité ainsi que son partage d'expérience . J'ai aussi eu la chance d'avoir un accueil exemplaire et bénéficier d'une certaine pédagogie de sa part. Travailler à ses coté fut réel un plaisir et j'espère que l'on restera en contact même après ce stage.

## Liste des auteurs du projet art-manager.

---

### Dev

- Alain ANDRÉ [wordsbybird@gmail.com](mailto:wordsbybird@gmail.com) [github](#)
- Jean-Baptiste PRIAM-MASSAT [jean.massat@gmail.com](mailto:jean.massat@gmail.com) [github](#)

### Images

- Jean-François ANDRÉ

### Traduction et marketing

- Anne Tess GUIMARÃES ARAÚJO DE SOUZA

### Concepte et présentation

- Alexandre RACCOUSSOT

Il m'est capital de remercier encore une fois M.Andre de m'avoir ajouté en tant qu'auteur du projet .Ce geste me touche d'autant plus qu'un tel titre n'était pas de vigueur du fait de ma position durant le projet art-manager.

Pour conclure, je remercie également Hilzonde Tauxe pour son aide littéraire dans la rédaction de ce rapport de stage.

# Sommaire

<b><u>Remerciements</u></b>	<b>2</b>
<b><u>1.Introduction</u></b>	<b>4</b>
1.1. Présentation du projet	4
1.2 Situation du projet à mon arrivée en stage	5
1.3 Cahier des charges	6
<b><u>2.Présentation des outils</u></b>	<b>7</b>
2.1. CMS Joomla	7
2.2. GitHub/GIT	8
2.3 Notepad++	9
2.4. Gantry	9
2.5. Console de debug Firefox	9
2.6. MooTools	9
<b><u>3.Base de données</u></b>	<b>10</b>
<b><u>4.Développement</u></b>	<b>11</b>
4.1. Gestions des erreurs	11
4.2. Mise en place d'objets représentants des œuvres et des albums	14
4.3. Ajout et aperçu d'images personnelles	17
<b><u>5.Conclusion</u></b>	<b>22</b>
5.1. Apports personnel	22
5.2. Apports au projet	23
<b><u>6.Annexe</u></b>	<b>24</b>
6.1. Nouvelle ergonomie	24
6.2. Panel document	25
6.3. Class JavaScript gérant la requête Ajax de l'image	26
6.4. Extrait de la Class gérant l'ajout d'image	27
6.5. Recherche Add-on	28

## 1.1.Le projet Art-manager

**art-manager.org** est un site internet qui permet à un artiste inscrit sur le site d'informatiser et créer un suivi commercial de ses œuvres via des bons, des factures en relation avec ses contacts.

L'utilisateur peut modifier le statut marchand d'une œuvre en la déplaçant avec la souris vers un lien ou un objet parent (factures, contacts). Ces actions peuvent lui permettre de générer des documents utiles à sa profession et surtout ses activités d'un point de vue lucratif (Bon de commande, facture etc..).

Les différentes œuvres pourront donc être classées dans des albums créés et personnalisés (image, titre, description) par l'internaute.

Sur chacune des pages, le client dispose de filtres sur les parents d'œuvre lui permettant une commodité certaine quant à ses recherches. Le but du site étant de proposer une interface simple aérée.

A terme, le site proposera une offre d'abonnement pour ses utilisateurs, ce qui leur permettra de lever la limitation du nombre d'œuvres qu'ils pourront gérer sur le site.

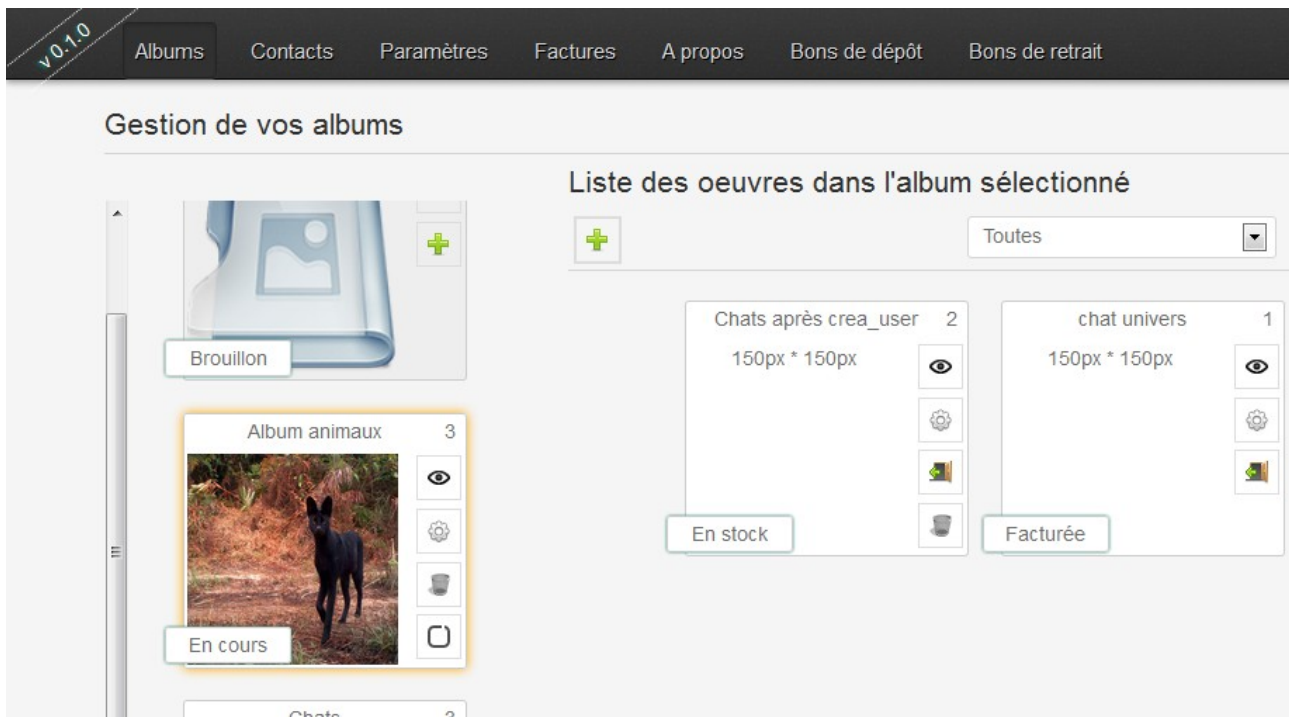
D'autres fonctionnalités sont aussi prévues mais mises en attente pour le moment, en attendant que le projet soit en ligne, stable, et fonctionnel. On retrouve entre autres l'idée d'ajouter au projet une dimension communautaire, à la même manière qu'un réseau social.

Le projet a un *repository* sur **Github.com** à l'adresse suivante :

<https://github.com/organizations/art-manager> rendant le code source accessible à tous et pouvant amener des contributeurs.

## 1.2. Situation du projet à mon arrivée en stage :

Lorsque j'ai commencé mon stage, le site était en version 0.1.0. L'interface graphique n'était pas finalisée. Au niveau des interactions et des échanges entre le serveur et le client, une base de données permettait de recueillir les informations envoyées par le client. La plupart de ces flux se faisaient en Ajax au format JSON. Pour chaque objet parent (album, facture, bon de dépôt etc..) , une page web différente était disponible, amenant l'utilisateur à se déplacer de page en page pour retrouver son contenu.



*- Art Manager dans sa version 0.1.0 pré stage -*

A ce stade du développement, mon tuteur de stage (Alain Andre) avait commencé une réévaluation des besoins des utilisateurs ainsi que de l'interface graphique du composant. De cette réflexion sont ressortis plusieurs points capitaux :

Tout d'abord, une interface plus ergonomique que celle proposée dans la version initiale serait un réel avantage puisqu'en effet, la consultation des données sur les objets parents se complexifie au fil de la navigation sur le site (duplication des vues). C'est pourquoi les caractéristiques d'interface du projet ont dû être repensées. Une esquisse d'une nouvelle idée d'interface a alors été dessinée (voir annexe « Nouvelle ergonomie »).

L'abandon des pages donnant accès aux objets parents tel que les bons de dépôt ou les factures fut réalisé. De fait, une nouvelle manière de penser la gestion et la création de ces documents a donc été adoptée (voir annexe « **Panel document** »+ « **Nouvelle Ergonomie** »).

### 1.3. Cahier des charges complet via github.com:

#### Phase I [Panels, Artworks, Albums, Contacts]

13 closed — 4 open

[Browse issues →](#)

76%

Due in 5 days

Cette phase du projet met en place ses bases:

- [JS, GIT] Installation de l'environnement de travail
- [PHP] Création des divers vues de la partie gestion des flux
- [JS] Création des classes Artwork, Album, Contact
- [PHP] Création des classes Artwork, Album, Contact
- [JS] Création du div de gestion des images
- [JS] Mise en place des limites d'écriture dans les inputs
- [PHP] Création des ajax d'enregistrement, maj, suppression
- [JS] Ajouter l'option "retirer de l'album".
- [PHP] Collecte des erreurs (debug)

#### Phase II [flux, documents]

0 closed — 8 open

[Browse issues →](#)

0%

Due in about 1 month

Mise en place des actions de déplacement d'œuvres : historisation et création de documents.

- [PHP] Vue du panel Document
- [JS] Selection du type de Document (dépôt, retrait, facture)
- [JS] Préremplissage du Panel Document (Date, numéro, données)
- [JS] Remplissage du Customer. Modification des champs possibles
- [JS] Remplissage du Contact choisi. Sélection en auto-complete; si existe pas, il sera créé
- [JS] Zone de drop d'artwork affiche les informations sous forme de tableau
- [PHP] Enregistrement du Document.
- [PHP] Ajax de déplacement d'œuvre.

#### Phase III [impressions, benchmarks]

1 closed — 13 open

[Browse issues →](#)

7%

Due in 2 months

Tests utilisateurs Alpha et benchmarking.

- [JS, PHP] Mise en place d'une prise de traces des actions utilisateurs
- [Utilisation] Test de l'application par une personne étrangère à sa dev.
- [PHP] Ajax d'impressions d'un Document de type dépôt
- [PHP] Ajax d'impressions d'un Document de type retrait
- [PHP] Ajax d'impressions d'un Document de type facture
- [PHP] Ajax d'impressions d'une Œuvre
- [JS] Mise en place d'un benchmarking des actions gourmandes

#### Phase IV [Traductions, finalisations]

0 closed — 10 open

[Browse issues →](#)

0%

Due in 3 months

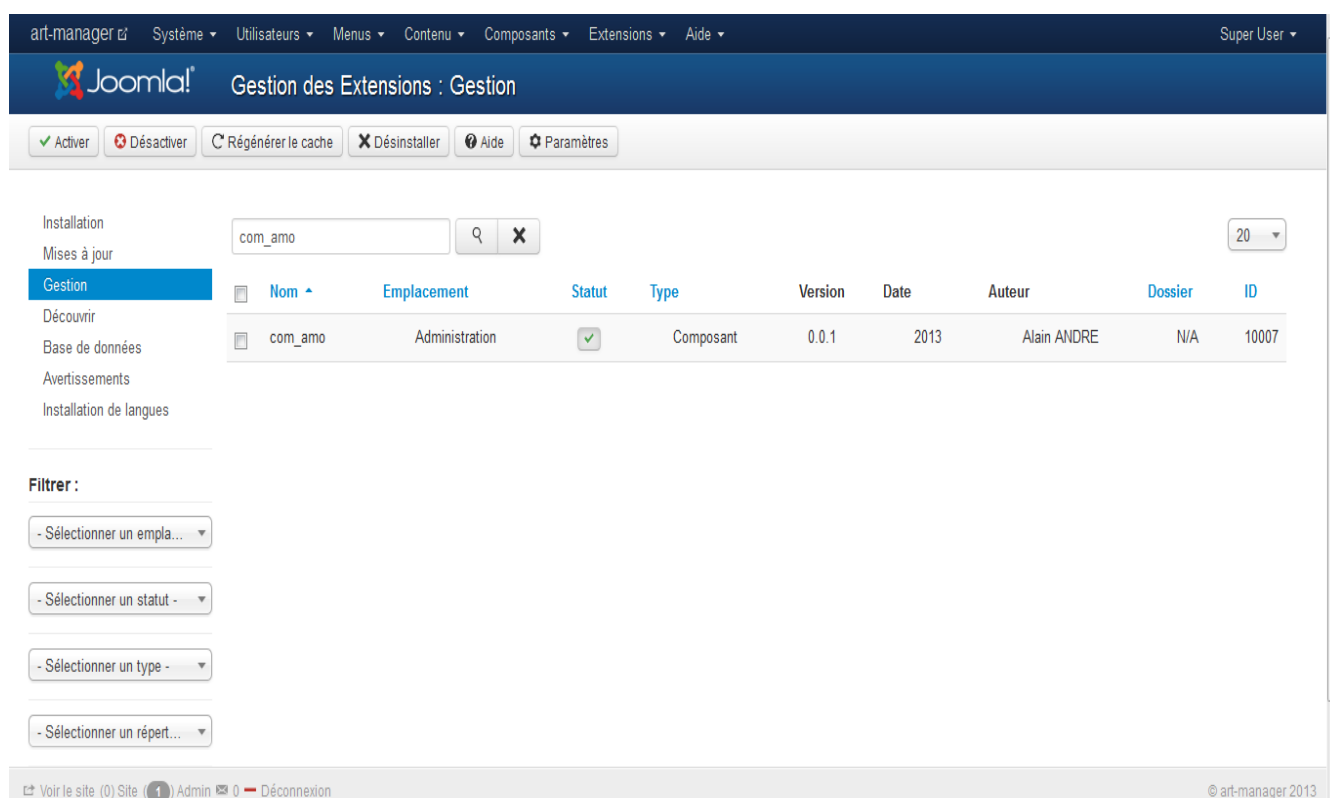
Mise en place des fichiers de langues et finalisation des détails restants.

- [XML] Traductions des fichiers de langue en Portugais
- [XML] Traductions des fichiers de langue en Anglais
- [PHP] Mise en place d'un moyen de créer un nouveau compte directement lié au composant
- [BETA] Mise en pré-production pour finalisation des détails
- [DROIT] Termes et conditions d'utilisation
- [JS] Confirmer la sortie utilisateur

## 2. Présentation des outils de développement :

### 2.1. -Joomla ! 3 :

Joomla 3 est un CMS « *Content Management System* », libre, *open source* et gratuit. Il constitue également un cadre de développement ayant des composantes *Back Office/Front Office* et offre une possibilité d'intégration de plugins et de composants ouvrant ainsi une très large gamme de possibilités à ses adhérents.



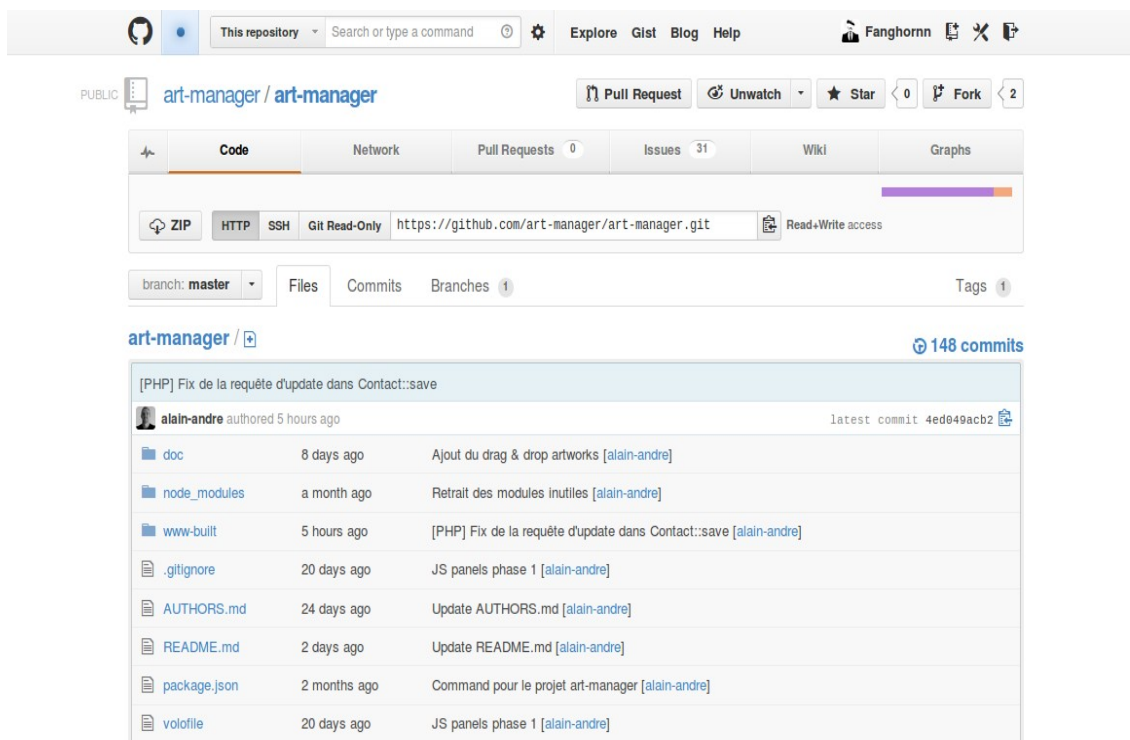
- Illustration de la partie Back Office de Joomla ! 3 -

« Pourquoi avoir choisis *Joomla* ? » fut ma première interrogation par rapport au projet. En effet, il présente de réels avantages pour la création d'un site web. Il met à disposition une multitude de fonctions PHP (avec une API associée) et octroie une gestion de la sécurité du code cruciale, notamment sur la problématique du « *Hacking* », en plus de ça, celles-ci sont mises à jour dès qu'une nouvelle version de Joomla est publiée. De tels avantages permettent de gagner en temps et en sécurité. On peut aussi retrouver des outils permettant de faciliter la traduction du site en différentes langues à l'aide de fonctions PHP.

Le projet Art-Manager est construit comme un composant *Joomla*, une archive « *com\_amo.zip* » contenant le MVC du site suivi d'un fichier *.XML* qui communique avec Joomla et des fichiers *Langues* en ce qui concerne les traductions.

## 2.2. - *github.com/GIT* :

Github est un site qui héberge un système de versionning décentralisé (Tous les contributeurs ont une copie du projet sur leur poste de travail) communautaire basé sur le fonctionnement du logiciel *GIT*, développé en 2005 par Linus Torvalds.



- Repository du projet Art-Manager sur *github.com* -

Il se trouve que l'utilisation de *GIT* se fait surtout avec une console prompt n'étant pas habitué à l'utilisation d'une console, j'ai dû tout au long de ce stage, me familiariser avec cette pratique. Github représente un vrai atout pour le développement d'un projet, en cas de complication, les fichiers ne sont presque jamais perdus dans la nature. Il offre également une certaine souplesse de développement car ils permettent d'avoir un environnement de travail propre et garantissent une clarté quant aux modifications et ajouts sur un projet. Chaque développeur travaille sur son repository (copie locale du projet) ainsi que sur sa tâche affectée.

Un nombre incroyable de projets et logiciels open source tels que Linux ou Ruby on Rails, se trouvent sur *github.com* et donc ouverts à tous contributeurs.



### **2.3. - Notepad++ :**

Notepad++ est un éditeur de code source qui prend en charge plusieurs langages. Il a pour fonction de fournir un éditeur de code source très performant. Je l'ai utilisé durant toute la période du stage et contrairement à l'utilisation d'un IDE (Environnement de développement intégré), et il se trouve que Notepad++ présente une qualité non négligeable qui est celle de forcer le développeur à porter une attention particulière aux lignes de code qu'il écrit.

### **2.4. - Gantry :**

Gantry est un framework graphique open source directement intégré en tant qu'extension de Joomla. Il offre de nombreuses fonctionnalités et aide dans la conceptions graphique d'un site web.

### **2.5. - Console de debug Firefox :**

Le stage m'a permis, de découvrir ce qu'était la console de debug Firefox. c'est un outil mis à disposition au sein du navigateur par ses créateurs et pour les développeurs, il se trouve que cette dernière se révèle être un outil apportant une grande quantité d'informations utiles au debug. Elle permet notamment d'inspecter ou d'éditer le code source et les éléments présents sur la page web, surveiller ainsi que contrôler les flux HTTP durant la navigation, et de signaler la présence d'erreurs et avertissements de compilation éventuelle.

### **2.6. - MooTools:**

Le MooTools est un framework JavaScript libre, compact, modulable et orienté objet. Il permet d'écrire un code puissant, flexible et compatible avec tout les navigateurs. Il est important de donner une attention particulière à ce framework car cet outil nous a permis de faciliter les démarches de renouvellement du site . Une majeure partie de mon travail repose sur l'utilisation du MooTools.

Il possède une API gigantesque et sa communauté a pu nous offrir de nombreuses possibilités et d'aides durant tout le long du stage, ce framwork nous donne le moyen de faire des Class, des sous Class avec les notions d'héritage.

Sa principale caractéristique étant de pouvoir exercer une vraie programmation orientée objet avec du JavaScript, Comme pour le PHP .

Il permet aussi de manipuler avec souplesse des éléments du DOM(Document Object Model), ou de styles. De plus, ses innombrables fonctions nous habilitent à appliquer une multitude de traitement ou d'événements.

### 3. Base de données

La base de données du projet est liée à celle de Joomla, lors de l'installation du CMS, un ensemble de tables sont créées pour assister le développeur dans sa manipulation des données. Dans notre cas échéant, c'est seulement la table #\_users de Joomla qui nous sera utile, elle permet d'enregistrer les informations de connexions de l'utilisateur et c'est donc elle qui sera sollicitée lorsque l'un d'entre eux souhaite se connecter à son compte.

La clé primaire de cet utilisateur est alors récupérée dans la table art-manager « #\_amo\_customers » (amo pour « art-manager »), dans le champ « uid » pour « user ID » et c'est lui qui sera utilisé comme clé étrangère ou Foreign Key pour toutes les opérations liées au site et à son utilisateur.



uid	adresse1	adresse2	telephone	contract	currency	contractValid
826				1		0

« #\_amo\_customers – Mon « user ID » et mon contrat pour le développement »

Notamment pour donner les différents privilèges à l'utilisateur enregistré selon son contrat, c'est notre table #\_amo\_contracts, qui va nous indiquer à quel type de contrat il appartient (abonnement) et donc combien d'œuvres il peut posséder au maximum par exemple.



id	price	label	description	externalConnection	artworkLimit
1	0.00	amo_CONTRACTS_FREE	amo_CONTRACTS_FREE_DESCRIPTION	0	100
2	20.00	amo_CONTRACTS_BASE	amo_CONTRACTS_BASE_DESCRIPTION	0	100
3	00.00	amo_CONTRACTS_BEST	amo_CONTRACTS_BEST_DESCRIPTION	1	0

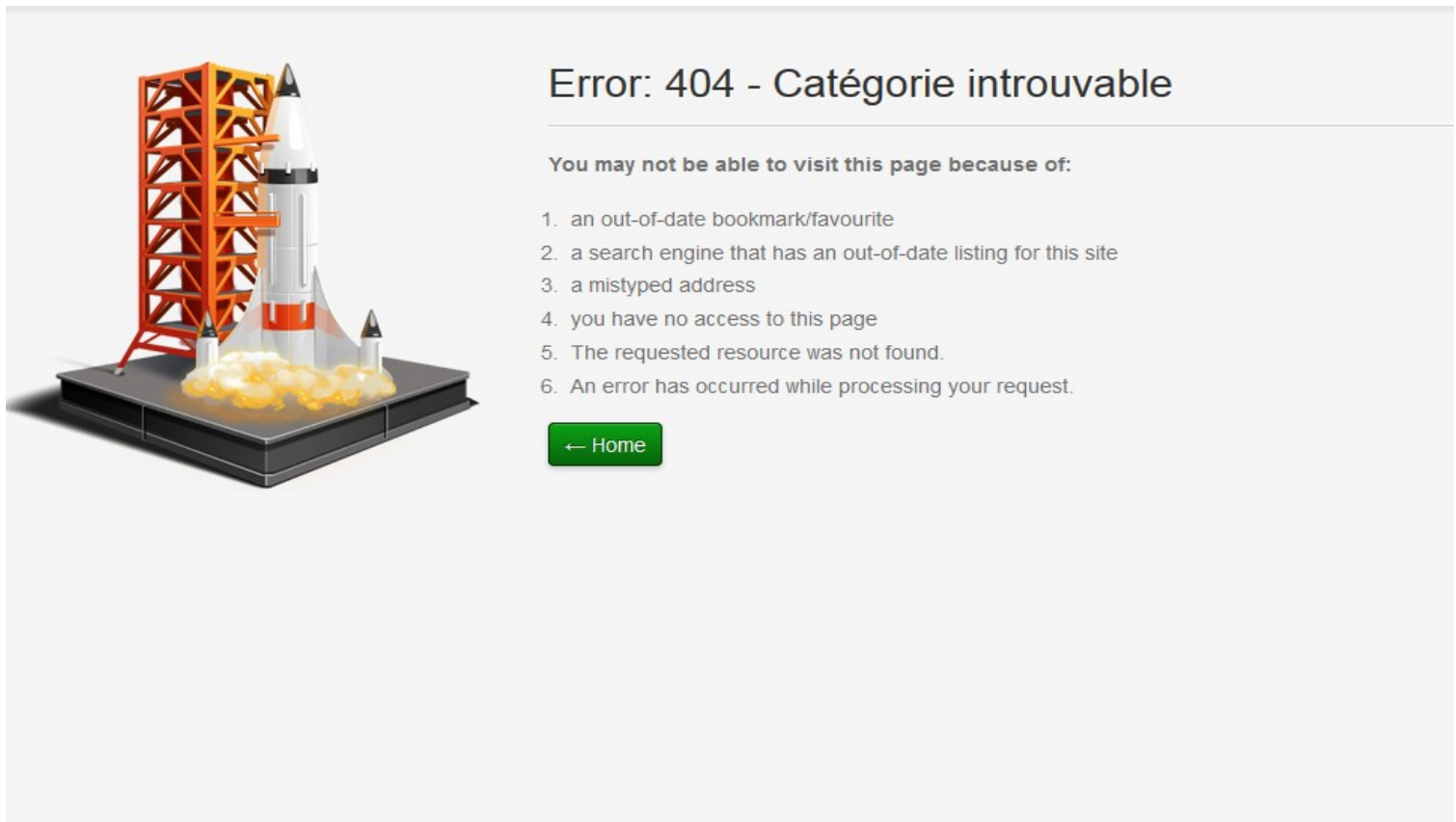
« #\_amo\_contracts - Les différents contrats possibles »

Concernant toutes les opérations de création d'objets parents ou d'œuvre, une table est prévue pour chaque entités. Notez aussi qu'une table est aussi prévue pour garder en mémoire l'emplacement des objets dans un parent. Par exemple la table #\_amo\_albums\_artworks crée un lien entre un identifiant d'album et tout les identifiants d'œuvres « artworks » qu'il contient.

## 4. Développement

### 4.1. Collecte et gestion des erreurs de navigation :

Lors de ma deuxième semaine de stage, une problématique s'est imposée d'elle même. En effet, *Gantry* contient dans son framework graphique, un fichier nommé ***error.php*** permettant l'affichage d'une page d'erreur pré-conçue. Cette page est affichée dès l'apparition d'une erreur durant la navigation de l'utilisateur. Voici un exemple d'affichage d'une erreur.



- Exemple de la page d'erreur de Gantry -

Après réflexions avec Mr Andre, nous avons déterminé deux points importants quant à la gestion des erreurs de navigation. Le premier point est que nous ne voulions pas que l'utilisateur soit interrompu d'une manière si abrupte. En plus de ça, l'internaute n'est pas concerné pas les erreurs du serveur et n'a pas forcément besoin de savoir quel est le code et le message retournés par l'exception.

En revanche, ce sont des informations nécessaires pour les administrateurs, de ce fait, la solution est de cacher la page d'erreur lors de la navigation.

Néanmoins, il nous fallait également trouver une parade permettant de récupérer ces informations pour les mettre à disposition des administrateurs pour le *debug*.

En premier lieu, pour venir à bout de cette tâche, mon réflexe a été de retrouver le code source de la page d'erreur estimant qu'il y ai de grandes chances pour que la majeure partie des modifications se fassent au sein de ce fichier.

Pour procéder aux modifications du script, la solution s'est donc présentée d'elle même et se définira en deux partie :

#### **-Récupération des erreurs :**

Il est plus intéressant pour le *debug* de connaître quand, quel utilisateur et avec quel navigateur l' erreur s'est produite, avec des recherches sur internet, j'ai pu trouver une fonction PHP récupérant une multitude d'informations sur le navigateur (nom, version etc..).

Création d'une table dans la base de données pour récupérer les informations :

**amo\_error :**

- id (int, auto increment)
- uid (int)
- errorCode (int)
- errorText (text)
- browserInfo (text)
- date (varchar)

Mise en place d'une fonction PHP récupérant les erreurs et les insérant dans la dite table.

#### **-Affichage :**

Redirection de l'utilisateur vers l'accueil :

La victime de l'erreur verra apparaître un pop-up en haut de la page l'avertissant qu'une erreur se serai produite et de fait, qu'il aura été redirigé vers l'accueil.

## Suppression de tout affichage :

Cette page ne sera de toute façon jamais visualisée à proprement dit étant donné que l'internaute sera redirigé dès la fin de l'extraction des erreurs dans la base, il est donc inutile d'alourdir le code de lignes inutiles.

```
25 if (!isset($this->error)) {
26     $this->error = JError::raiseWarning(404, JText::_('JERROR_ALERTNOAUTHOR'));
27     $this->debug = false;
28 }
29
30 //Insertion dans la base des informations de l'erreur.
31 $errorCode=$this->error->getCode();
32 $errorText=$this->error->getMessage();
33 InsertError($errorCode, $errorText);
34
35 function InsertError($errorCode,$errorText) {
36     ...
37     $uid = JFactory::getUser()->get('id');
38     $browserInfo=getenv("HTTP_USER_AGENT");
39     $errorCode = (!get_magic_quotes_gpc()) ? addslashes($errorCode):$errorCode;
40     $errorText = (!get_magic_quotes_gpc()) ? addslashes($errorText):$errorText;
41     $date=date('Y-m-d H:i:s');
42
43     $db=JFactory::getDBO();
44     $query = "INSERT INTO `#__amo_error`(`uid`, `errorCode`, `errorText`, `browserInfo`, `date`)
45             VALUES ('$uid','$errorCode','$errorText','$browserInfo', '$date')";
46     $db->setQuery($query);
47     $db->query();
48 }
49
50 //popup à l'accueil
51 JText::_('ERROR_DISPLAY');
52
53
```

- Extrait du code source error.php -

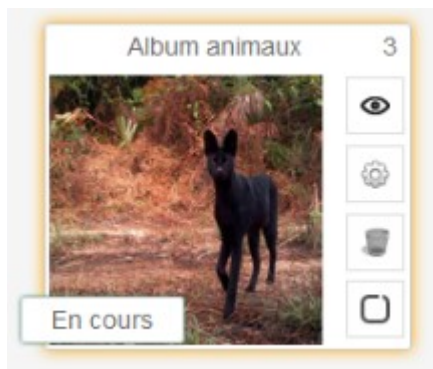
Comme on peut le remarquer sur cet extrait, l'ouverture de la connexion à la base de données est faite par le biais d'une fonction PHP issue de l'API de Joomla « `$db=JFactory::getDBO()` ».

Ce genre de fonctions permettent de mettre en place toutes les mesures de sécurité lors des manipulations de bases.

Après différents test de mon travail, toutes les informations utiles au debug sont correctement récupérées et insérées dans la base de données et c'est ainsi que mon travail fut rajouté au repository **Github** du projet.

## **4.2. -Mise en place d'objets JavaScript représentant des œuvres et des albums**

Comme vous pouvez le constater sur l'impression écran de la première version du site, des blocs HTML permettent l'affichage d'objets . Les Objets sont dotés de fonctionnalités selon leur type (album , œuvre , galerie , transporteur...). En exemple nous pourrions prendre un album avec un bouton permettant la visualisation de toutes les œuvres qu'il contient tandis qu'une œuvre elle , ne peut contenir d'objet et n'a donc pas besoin d'une telle fonction).



*« un objet album de la première version »*

Suite à la fameuse restructuration complète du site, le fonctionnement et la construction de ces objets devaient être entièrement revus.

Pour parvenir à cet objectif, je me suis inspiré de la conception des objets ayant principalement une composante JS et PHP qui étaient déjà en place sur l'ancienne version. Voici donc les objectifs de développement :

### **JavaScript:**

- Création d'une Class « WIDGET » nous proposant de mettre en place la structure HTML, que celle-ci soit un contact ou une œuvre, l'aspect graphique général de l'entité sera la même selon la charte graphique. Un ID permet d'identifier de manière unique chaque instances d'un WIDGET à sa création.

- Création d'une sous Class héritant de WIDGET pour chaque type d'objets. Des éléments HTML sont créés selon leur nature. Par exemple, le statut d'un album affichera le nombre d'œuvres qu'il contient tandis que les œuvres elles auront un statut mercantile en affichage. Ces sous objets constituent donc l'aspect final des styles et des données.

Des données au format JSON leurs seront passées en paramètre à leur création et donc prévoir tous les « getters » et « setters » qui donneront lieu aux assignations des valeurs aux éléments adéquats.

Lorsqu'une autre partie JavaScript du site aura besoin d'afficher des objets, ces Class seront instanciées, autant de fois que la base de données retourne d'objets par un Ajax.

## PHP :

Certes, nous avons structuré graphiquement nos objets et ces derniers sont donc prêts à recevoir des données en JavaScript, mais nous avons besoin du PHP pour aller chercher nos données dans la base.

De fait , la meilleure solution fut celle de prévoir dans le contrôleur principal du composant, un accès à d'autres contrôleurs PHP (des Class en réalité) pour chaque types d'objets.

Dans le controlleur albums par exemple, nous retrouverons **toutes** les opérations de base de données concernant les albums du client (enregistrements, suppressions, modifications).

Pour detecter les incohérences de réponse de base données, des vérifications sont faites en php au retour de la base :

J'ai choisi de reprendre l'exemple du controlleur albums pour illustrer mes propos, une fonction listAlbums() à pour tâche de retourner tous les albums de l'utilisateur à l'Ajax appelant, si le nombre d'album est inférieur ou égal à 0, c'est qu'il n'y aucun albums ou que le retour de la base de donnée comporte une anomalie, alors on assigne un code et un message à deux variables qui seront présentes dans la réponse de l'Ajax.

```
/** Albums */
foreach($rows as $row){
    $query = "select count(`artwork`) as number from `#__amo_albums_artworks` where `uid`
    $db->setQuery($query); $r = $db->loadObject();
    array_push($objects, array("type"=>"album", "id"=>$row->id, "label"=>$row->label, "n
}
if(count($rows) > 0){ $result=1; $message=JText::_('AMO_LIST_ALBUMS_OK'); }
else{ $result=2; $message=JText::_('AMO_LIST_ALBUMS_EMPTY'); }
```

*« une partie de la fonction listAlbums du controlleur Albums »*

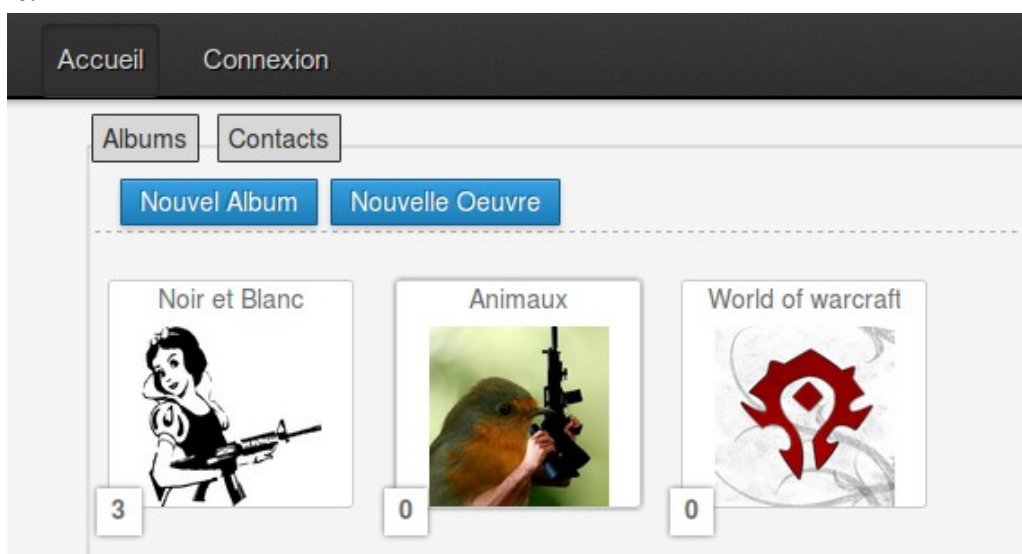
Selon le code et le message retournés, le fonction callback de l'ajax va prendre les mesures appropriées. En fait, la gestion des erreurs réside dans l'anticipation d'une erreur possible et la communication entre les différents acteurs des opérations.

Une fois les données récupérées, elles sont encodés au format JSON et c'est ce qui va constituer la réponse de la requête ajax. Cette réponse va servir à instancier les WIDGETS dont le constructeur se contentera de placer et organiser le JSON passé en paramètre grâce au getters et setters définis dans sa Class.

## CSS :

L'utilisation du CSS pour la mise en forme des éléments HTML construits en JavaScript a bien évidemment été de rigueur. Il me semble important de souligner ce point car l'emploi du CSS a été un exercice fructueux dans mes recherches quant à la compréhension des comportements de certaines propriétés CSS pendant la création de ces objets, notamment sur les notions de marges et de positions.

Bien que l'utilisation du CSS ne soit pas primordiale dans le cadre de cette présentation de rapport de stage , son aspect graphique me semble être un bon véhicule pour conclure le paragraphe sur cette tâche.



« Liste des Albums »



« Liste des œuvres dans un albums »



### **4.3. -Ajout et aperçu d'images personnelles :**

Art-Manager est à proprement dit un site de gestion d'œuvres personnelles . De fait , la partie centrale du site repose sur l'interaction de l'internaute entre ses différents widgets. Quoi de plus normal donc de lui proposer l'envoi d'une image au serveur pour la création d'un widget, ce qui lui permettra de différencier d'un simple coup d'œil toutes ses œuvres, albums etc...

Ma dernière mission consistait à proposer à l'internaute un système simple et intuitif lui permettant de choisir son image avant l'enregistrement de son objet.

#### **Par où commencer ?**

En effet, concevoir une telle chose nécessite une définition précise des étapes mais nous commencerons par énumérer les différentes consignes préconisées par mon tuteur dont voici les grandes lignes :

- Proposer le choix et un aperçu de l'image voulue à la création/modification de l'objet.
- Cet enregistrement se fera avant la soumission du formulaire de création de l'œuvre et donc avant l'envoi de l'image dans la base de données.
- Elle sera enregistrée dans un dossier temporaire sur le serveur lié à au compte de l'utilisateur en attendant l'enregistrement du formulaire.
- Une balise image vide activera le gestionnaire lors d'un clic.
- Appliquer des traitements et des vérifications à l'image avant de l'accepter sur le serveur.
- Proposer à l'internaute une solution de Drag&Drop(Glisser&Déposer) depuis une image provenant sa machine.

## **Recherches :**

Les bases étant posées, j'ai du commencer par entreprendre des recherches afin de pouvoir mener à terme le développement de cette composante.

### **-Envoi d'image avec Ajax :**

Il me fallait trouver le moyen d'envoyer une image en JavaScript par un Ajax. C'est sur github qu'une solution s'est proposée d'elle même via un contributeur du Mootools, il proposait aux sympathisants du Framework une Class octroyant cette possibilité (**voir annexe 3**).

Néanmoins, cette dite Class proposait des fonctions inutiles à mon projet comme par exemple, la possibilité d'envoyer plusieurs images d'un coup et nous n'avions aucun intérêt à alourdir et encombrer le code par ces biais. C'est par cette problématique qu'une nouvelle facette du développement s'est faite découvrir.

## **Drag&Drop:**

Le dernier objet de mes recherches concerne l'intégration du Drag&Drop. Je me suis d'abord tourné vers la documentation officiel du Mootools tout en connaissant déjà des notions de drag&drop grâce à mes précédentes consultations de l'API.

Malheureusement, cette documentation se révéla être une fausse piste. En effet, elle était très complète mais n'autorisait que l'action du drag&drop entre éléments du DOM(Document Object Model), tandis que je cherchais à capturer un fichier provenant du système d'exploitation de l'internaute.

Le succès de mon travail s'est fait après de fructueuses recherches auprès de l'un des concepteur du Mootools nommé « Arian Stolwijk ». C'est d'abord sur son profil github que j'ai pu trouver l'ensemble des Class permettant de déposer plusieurs fichiers locaux dans une zone prévue à cet effet. Arian étant un acteur assidu du Framework, son travail a été publié en tant qu'add-on sur le site officiel (**voir annexe 5**).

Le Mootools est un framework écrit en JavaScript et la plupart des add-on proposés par ses différents adeptes sont écrit en pur JS. Ainsi, si mon but était d'obtenir un résultat qui me correspond, je n'avais guère le choix que d'aller au sein du code et modifier le comportement de la Class à ma guise. Cela fut l'occasion pour moi d'apprendre à lire un code source étranger, le comprendre, mais surtout de parvenir à le modifier ou l'optimiser dans mon propre intérêt.

### **Difficultés rencontrés :**

C'est surtout lors de l'édition de l'add-on du « Drag&Drop » que j'ai pu rencontrer le plus de difficultés. Ne concernant que la récupération des données après un événement de glisser/déposer il m'aura aussi fallu ruser pour faire cohabiter cet add-on avec l'envoi des données en Ajax.

### **Note :**

En ce qui concerne les codes sources, ces derniers étaient sous licence MIT-style, nous autorisant donc à les utiliser et modifier selon notre bon vouloir du moment que les auteurs sont cités.

### **Enregistrement temporaire de l'image :**

C'est au rôle du PHP de récupérer cette dernière, envoyée par l'Ajax et d'employer des procédures de vérification et enregistrement sur le serveur. Nous avons donc créé un contrôleur nommé « Controller.Images.php » ,en reprenant la même logique citée dans l'enregistrement des « Widgets ».

A sa réception, l'image est sous forme d'un tableau contenant diverses informations :

- Le nom du fichier.
- Son emplacement temporaire sur le serveur apache.
- Un code d'erreur retourné par le serveur en cas d'exceptions (ex :Fichier plus lourd que la limitation autorisée du PHP.ini).
- Les données brut de l'image.

Le contrôleur récupère les erreurs s'il s'en présentent, et effectue ainsi différents traitements notamment quant au nom du fichiers pour le nettoyer de tout caractères non désirés afin d'éviter des tentatives de piratage.

Après toutes vérifications, le script effectue un upload du fichier vers le nouvel emplacement temporaire. Celui ci se trouve sur le serveur qui se situe dans un dossier créé pour l'utilisateur concerné et dont le nom sera généré par la combinaison de l'identifiant du compte et d'un « hash »(suite de caractères générés aléatoirement) ,toujours dans une dynamique sécuritaire.

Le PHP finit donc par générer un JSON encodé contenant :

- Nouvelle url de l'image après upload
- Message d'erreur (si erreur il y a)
- Code retour

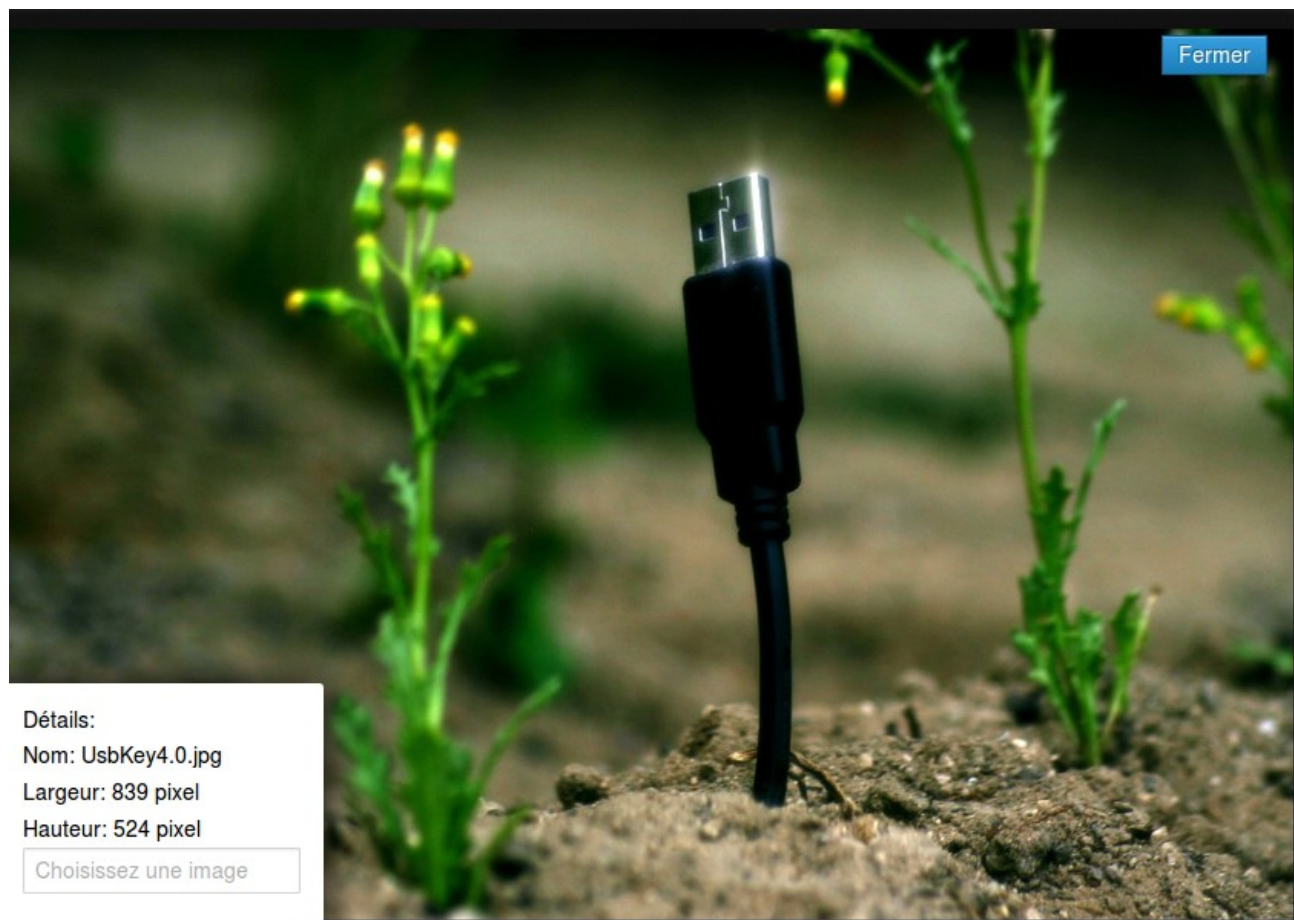
Une fonction attendant le retour de l'Ajax se prépare à recevoir les données et agir en conséquence.

- Ouverture de l'interface graphique si Drag&Drop
- Affichage de l'image et redimensionnement
- Affichage d'informations
- Affichage d'une erreur dans le cas d'une présence d'erreur

### **Mise en place d'une interface graphique :**

Pour concevoir un affichage des plus limpides à lire, j'ai du entreprendre l'utilisation de ce que l'on appelle un « overlay »(cadre s'affichant au premier plan et bloquant l'arrière plan) associé à un bouton de fermeture. Au retour de l'Ajax, un panneau sera placé à l'intérieur pour donner diverses informations calculées par une autre fonction.

Une balise « input » de type « file » offre la sélection d'une image dans l'ordinateur.




« Interface du gestionnaire après sélection »

Albums

Contacts

Retour

2



Nom

Nom de l'oeuvre

Description

Decrivez votre oeuvre

Date de création

aaaa-mm-jj

Dimensions

Définissez les dimension de l'oeuvre

Hauteur

Définissez une hauteur

Prix

Prix HT

Liste des matériaux

Imprimer

Enregistrer

« A la fermeture de l'interface, l'image reste affichée dans la balise initiale au sein du formulaire »

### **Difficultés :**

Le plus complexe a été de pouvoir redimensionner correctement l'image selon sa taille et celle de l'écran. Il faut imaginer tout les cas de figure possibles, l'outil de la console firefox permettant modifier la résolution de la page me fut d'une grande aide. La résolution de cette problématique permet en outre d'obtenir une bonne adaptation du site sur des terminaux mobile ou de type tablette.

Le debug fut aussi long. De plus, de longues recherches sur l'API du Mootools ont été nécessaires afin de trouver des fonctions qui on pu m'accorder de réaliser un travail propre, fluide et sans bugs. Le plus grand problème était que le script du retour de l'image s'exécutait beaucoup trop vite pour que la fonction de redimensionnement d'image puisse récupérer les largeurs et hauteurs .

La solution fut la découverte d'une fonction Mootools permettant d'attendre que l'image s'affiche.

## **5. Conclusion**

### **5.1. Les apports personnels:**

Ce stage aura été pour moi l'occasion de me perfectionner dans la pratique du développement. J'ai eu grand intérêt à œuvrer dans l'évolution du projet , car ce dernier m'a permis de m'investir au maximum et donner le meilleur de moi même.

En conséquence, j'ai pu grandement accroître mon expérience personnelle dans le monde du développement et satisfaire mes attentes quant à ce stage.

Malgré les bases acquises au cours de la formation AFPA, les débuts du stage se révélèrent être particulièrement difficiles étant donné la quantité importante de nouvelles notions et concepts à acquérir.

Finalement après un bon mois d'adaptation j'ai su me sentir à l'aise avec les outils de développement et les objectifs de développement.

Malheureusement, mon stage avait un caractère non conventionnel puisqu'il ne se déroulait pas au sein d'une entreprise et donc je n'ai pu apprendre que peu de chose sur le monde professionnel. Cependant , je ne considère pas avoir perdu en apprentissage étant donné l'usage de différentes méthodes de travail (télétravail , visioconférences..) et m'auront permis d'avoir une certaine proximité avec mon tuteur de stage qui a su être entièrement disponible et pédagogue lors des difficultés rencontrés.

Grâce à cette expérience et au diverses conversations avec M.Andre, ce dernier aura su me faire découvrir le monde du logiciel libre et son avancé incroyable dans les technologies de développement comme le MooTools , m'ayant habilité à appréhender toutes les subtilités de la programmation orientée objet (POO). C'est d'ailleurs pourquoi j'y porte un grand intérêt aujourd'hui et souhaite m'impliquer et contribuer à cette dimension.

Pour conclure sur mes apports personnels, le fait d'avoir repris un projet depuis le début à pu m'apprendre à en concevoir un en tant que tel.En effet ,les étapes de la création d'une application dans son intégralité étant encore un peu floues à mes yeux lors de mon arrivée en stage.

Je suis particulièrement satisfait d'avoir pu prêter main forte à Art-Manager et espère que le site aura tout le succès qu'il mérite lors de sa mise en ligne . Cette expérience sera clairement pour moi un facteur de motivation et de détermination pour rentrer le plus rapidement possible dans le monde du travail.

## **5.2. Les apports au projet :**

Mon tuteur travaillant pratiquement seul sur Art-Manager avant mon arrivée, je considère que l'impact de mon stage a pu porter une évolution et un déroulement du projet plus rapide qu'initialement . Les résultats qu'ont pu porter mes recherches et nos travaux mutuels sont les auteurs de ma satisfaction aujourd'hui.

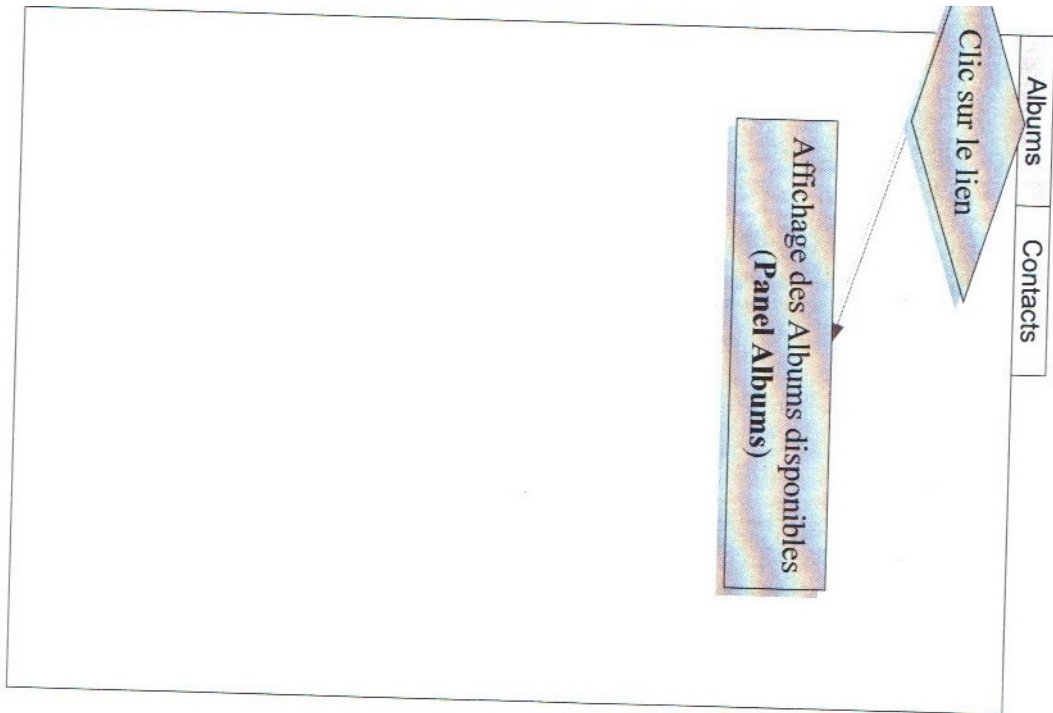
J'ai pu aussi soumettre des idées de développement ou de fonctionnement qui ont été écoutées et même quelques fois approuvées et intégrées au projet comme par exemple des idées quant à l'aspect graphique des « Widget », mais également dans les écritures du codes.

Monsieur Andre m'a donné la chance d'avoir une certaine liberté dans la conception des applications et j'ai donc fais de mon mieux pour qu'elles soient optimisées et surtout propres dans l'écriture.

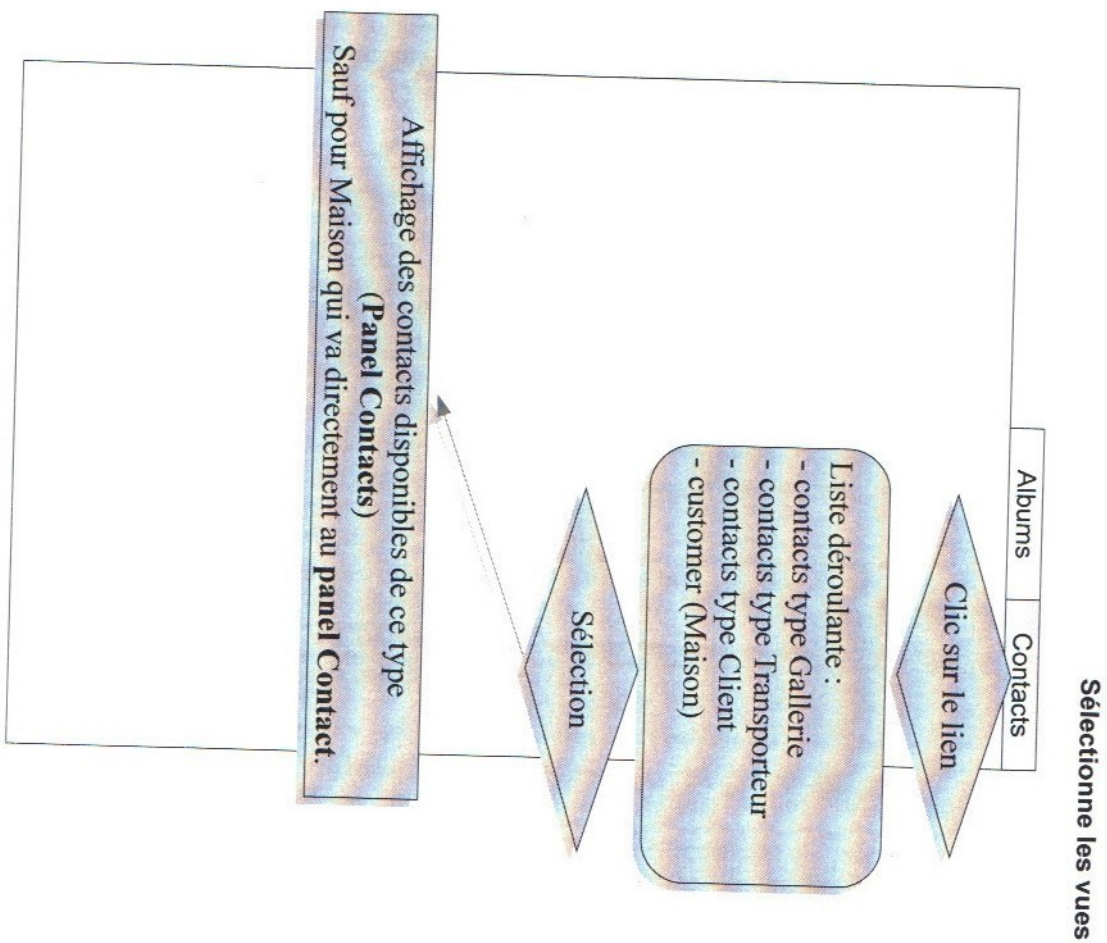
Lors de la fin de mon stage, tout le développement des composantes les plus lourdes et longues à concevoir du projet étaient terminé. Mr.Andre m'a d'ailleurs fait savoir que la phase de test BETA, donc la mise en ligne du site n'était plus très loin et donc que les choses allaient vite s'accélérer après mon départ.

## 6. Annexe

### 6.1. Nouvelle ergonomie :

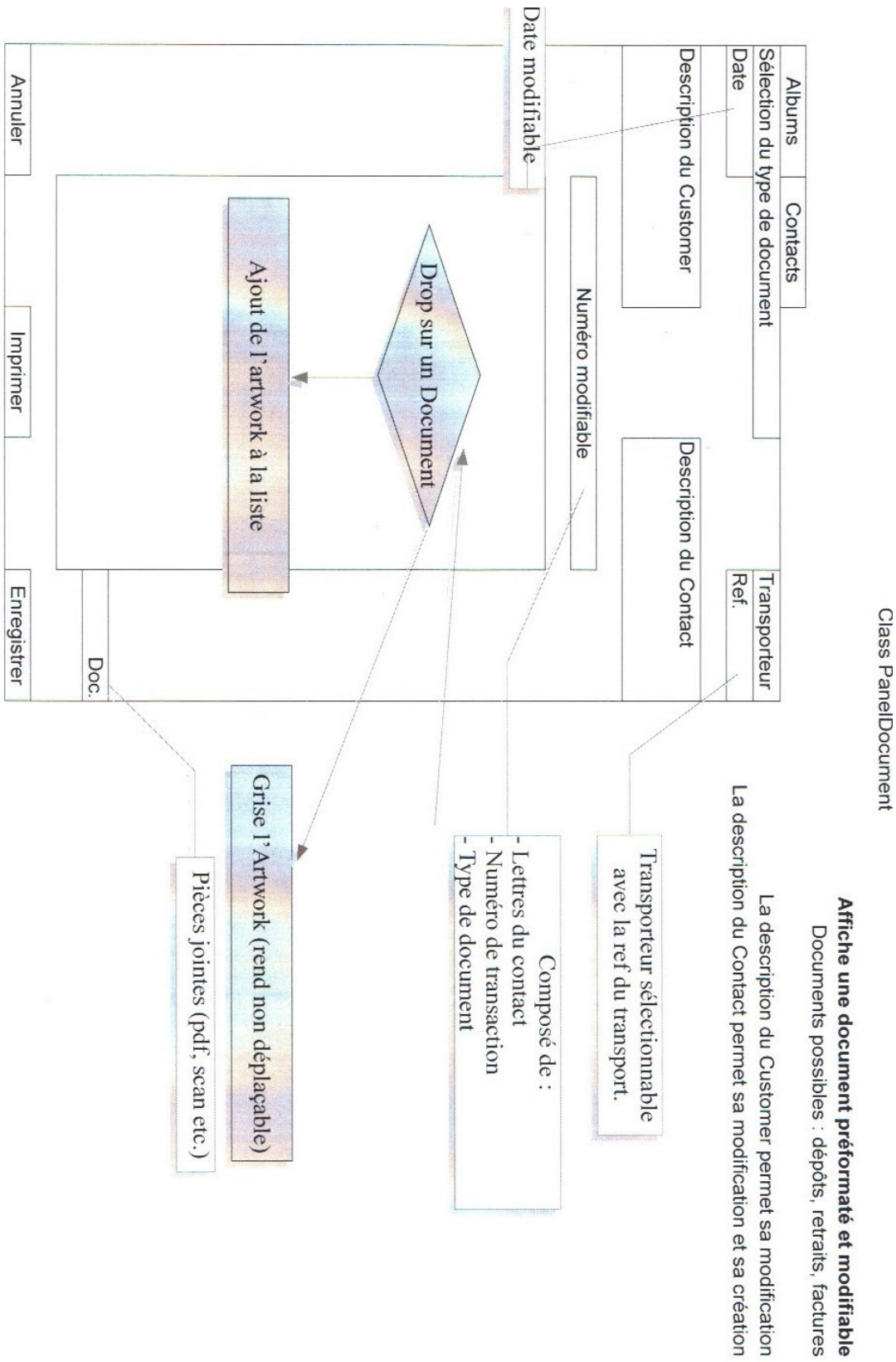


Class Panel





6.2. Panel document :




### 6.3. Class JavaScript gérant la requête Ajax de l'image :


```
22 File.Upload = new Class({
23
24   Implements: [Options, Events],
25   options: {
26     onComplete: function(){}
27   },
28   initialize: function(el, type, drop, fileDrop, options){
29     var self = this;
30     this.el=el;
31     this.type=type;
32     this.drop=drop;
33     if(drop) this.file=fileDrop;
34     this.setOptions(options);
35     this.uploadReq = new Request.File({
36       onComplete: function(){
37         self.fireEvent('complete', arguments);
38         this.reset();
39       }
40     });
41     if(this.options.data) this.data(this.options.data);
42     if(this.options.images) this.addMultiple(this.options.images);
43   },
44   data: function(data){
45     var self = this;
46     if(this.options.url.indexOf('?') < 0) this.options.url += '?';
47     Object.each(data, function(value, key){
48       if(self.options.url.charAt(self.options.url.length - 1) != '?') self.options.url += '&';
49       self.options.url += encodeURIComponent(key) + '=' + encodeURIComponent(value);
50     });
51   },
52   ,addMultiple: function(inputs){
53     var self = this;
54     inputs.each(function(input){
55       self.add(input);
56     });
57   },
58   ,add: function(input){
59     var self = this;
60     if(self.drop==false){
61       var input = self.el.retrieve('hiddenInputFile'),
62       name = input.get('name'),
63       file = input.files[0];
64     }else{
65       var input = self.el.getElement('.amo-input-drop'),
66       name = input.get('name'),
67       file=self.file;
68     }
69     this.uploadReq.append(name, file);
70   },
71   ,send: function(input){
72     if(input) this.add(input);
```

#### 6.4. Extrait de la Class gérant l'ajout d'image

```
9
10 var DivImage = new Class({
11   initialize: function(el, type){
12     var self = this; this.attachClick(el);
13     var hiddenInputFile = new Element('input', {type:'file' ,class:'amo-input-hidden amo-hidden', name:'image'});
14     var formDropImage = new Element ('form', { method:'post',class:'amo-drag-form', action:'files.php', enctype:'mul
15     var divFormRow = new Element('div', {class:'formRow'});
16     var inputDropImage = new Element('input', {type:'file', class:'amo-input-drop amo-hidden', name:'imageDrop'});
17     el.store('divImage', this);
18     el.grab(formDropImage);
19     el.store('formDropImage', formDropImage);
20     formDropImage.grab(divFormRow);
21     divFormRow.grab(inputDropImage);
22     el.store('hiddenInputFile', hiddenInputFile);
23     document.body.grab(hiddenInputFile);
24
25     hiddenInputFile.onChange = function (){
26       var upload = new File.Upload(el, type, false, null, {
27         data: { type:type }
28         ,url: 'index.php?option=com_amo&format=raw&task=uploadImage'
29         ,onRequest: function(){
30           waiter(el, true);
31         }
32         ,async : false
33         ,images: ['image']
34         ,onComplete: function(response){
35           response=eval('(' + response + ')');
36           //Recuperation des données provenant de image.controller et appel fonction affichage du nouveau src
37           if(response.result!=0){ message(response.result, response.message); }
38           else{
39             var newSrc = response.url, divImage = self.showDisplay(el, newSrc);
40             divImage.addEvent('load', function(){
41               self.calculate(divImage, type);
42             });
43           }
44         }
45       });
46       upload.send();
47     }
48     // Gestion du Drag&Drop
49     var uploadImage = new Form.Upload(el, '.amo-input-drop', type, {
50       dropMsg: "Lacher image ici"
51       ,onComplete: function(){}
52     });
53     //Fin du Drag&drop
54   }
55
56   /**
57    * @param el : Le DivImage
58    */
59   ,attachClick : function(el){
```

## 6.5. Résultat de mes recherches pour la gestion de Drag&Drop

  
a compact javascript framework

IN PARTNERSHIP WITH 

Home | Download | Docs | Forge | Blog | Demos

---

### Form.Upload 0.3

Upload your files the HTML5 way, drag and drop from your computer, multiple files the ajaxy way!

It works about the same as uploading attachments to gmail.

- HTML5 Drag and Drop files from your computer
- HTML5 multiple attribute files for selecting multiple files
- Uploading files through XMLHttpRequest with FormData
- Progress bar
- Graceful degradation for IE / Opera

[DEMO](#) [DOWNLOAD](#)

#### Details

**Author** [Arian Stolwijk](#)

**Current version** 0.3

**GitHub** [arian/mootools-form-upload](#)

**Downloads** 3502

**Category** [Utilities](#)

**Tags** [upload](#) [html5](#) [file](#) [drag and drop](#)

**Report** [GitHub Issues](#)

#### Releases

[0.3](#) [0.2](#) [0.1](#)

#### Dependencies

- `_self_/_current_:`
  - Element.Event
  - Class
  - Options
  - Events
  - Form.MultipleFileInput
  - Request.File
  - Request

#### How to use

Include the JavaScript files in your page, for example with the `<script>` tag, or with a Script Loader with an extra

[GO](#)

#### Developer Login

Email

Password

[LOGIN](#)

[Sign in with Twitter](#)

[Normal Signup](#)  
[I forgot my password](#)

#### Browse (all)

[Effects](#)  
[Forms](#)  
[Interface](#)  
[Media](#)  
[Native](#)  
[Realtime](#)  
[Request](#)  
[Utilities](#)  
[Widgets](#)

#### Resources

[How to Add a Plugin](#)  
[Plugin Writing Guidelines](#)  
[Send Feedback](#)

#### People

[Plugin Authors](#)