mediasmart

Rapport de stage

Clément MOYSAN

Licence Professionnelle Réseaux et Télécommunications

Option Web Développeur

Année universitaire : 2008/2009

Tuteur en entreprise :

Laurent DUBOIS

Tuteur universitaire :

Harold TRANOIS

Sommaire

I. Introduction

Dans le cadre de la validation de la licence professionnelle « réseaux et télécommunications » option Web Développeur développement de site intranet extranet, un stage de 18 semaines (Du 9 mars au 10 juillet) dans une entreprise m'a été demandé. Ce stage a été réalisé dans l'entreprise MEDIASMART, qui est une entreprise de Web développement qui réalise ou édite des sites.

Cette entreprise est basée sur Paris mais j'ai effectué mon stage sur Saint-Quentin tout en étant en communication avec l'agence.

Ce stage m'a permis de mettre en œuvre mes connaissances vues tout au long de l'année mais aussi celles des années précédentes. Il m'a permis aussi d'apprendre tout en mettant en application un nouveau Framework basée sur le langage PHP, le Framework s'appelle Symfony. Pendant cette période, j'ai commencé par développer un site dit « classique » (statique et dynamique avec gestion d'un back-office) et j'ai fini le stage par un deuxième projet plus compliqué avec la gestion des utilisateurs ou encore avec des fonctionnalités plus complexe.

Pour commencer, dans ce rapport vous aurez une synthèse du contexte professionnel, mais aussi un petit résumé sur les projets réalisés durant le stage.

Ensuite vous aurez une partie plus technique. Je commencerai par vous présenter la partie analyse des projets, ensuite le développement des projets avec l'explication des différents problèmes rencontrés, et je terminerai par une petite conclusion de chaque projet.

Enfin pour terminer ce rapport, je vous ferai une petite conclusion de ce stage, avec un bilan personnel sûr comment j'ai ressenti le stage.

II. Présentation du contexte de stage

A. Présentation de l'entreprise MEDIASMART

1. Présentation de l'entreprise

a) *Présentation générale*

MEDIASMART (http://www.mediasmart.fr/) est une entreprise qui réalise plusieurs métiers :

- Audit et Conseil
- Web Design
- Ingénierie Web
- E marketing et référencement

Cette entreprise est basée dans le onzième arrondissement de Paris.

b) *Historique et organigramme de l'entreprise*

MEDIASMART a été crée début 2004. C'est une SSII (**société de services en ingénierie informatique**) qui avait comme activité principale le développement d'application web. De 2004 à 2008 l'entreprise a eu une croissance importante de son activité principale. Et en 2009, elle a donné un coup d'accélérateur en ouvrant son capital à un partenaire financier (Laurent DUBOIS) et en faisant appel à des opérations de croissance externe en amont et en aval (Société de e-marketing et un studio graphique).

Au jour d'aujourd'hui, MEDIASMART a cet organigramme suivant :

- Deux directeurs conseils qui délivrent des préconisations ergonomiques et fonctionnelles concourants à l'atteinte des objectifs fixés.
- Un directeur technique qui étudie les objectifs et les contraintes techniques et budgétaires demandés par le client.
- Trois chefs de projets qui sont là pour faire la passerelle entre le bon déroulement du développement des applications web (Réalisation du cahier des charges, Répartition des tâches que ca soit dans le développement, en graphisme ou en marketing, Vérification du site avant déploiement sur un serveur, …) et le client qui a demandé cette application. Un seul chef de projet gère un projet d'un client.

c) *Les activités de MEDIASMART*

Audit et conseil :

MEDIASMART est à l'écoute de ce que souhaite le client. Le directeur conseil n'hésite pas à apporter ses idées dans le projet, et donc conseille le client pour une meilleure réalisation du projet dans un meilleur investissement. Suite à cela ils élaborent ensemble le cahier des charges, le document des spécifications fonctionnelles et techniques, puis les gabarits fonctionnels (navigation et étude ergonomique). Après la réalisation du site, MEDIASMART font des tests fonctionnels et de performances avant de le proposer au client.

Ingénierie web :

MEDIASMART est avant tout une entreprise qui développe des sites. Avant de commencer à développer elle fait l'analyse du cahier des charges. Après lecture de ce document, elle énumère les spécifications techniques du site, elle en fait une architecture pour après faire une modélisation d'une base de données. Après cela sous un environnement LAMP, elle commence le projet en faisant l'intégration HTML/CSS du site, avec des interfaces riches (FLASH, AJAX, …) et avec une ergonomie riche. Leur développement se fait à l'aide de PHP5 orienté objet. Pour cela les développeurs utilisent principalement des Framework. Ils utilisent aussi des solutions open-sources de référence comme Magento (utile pour les sites d'e-commerce) ou encore Typo3 (système de gestion de contenu libre). Pour stocker les données, MEDIASMART maîtrise les principales bases de données (MySQL, Oracle, PostGreSQL). Après avoir fait les tests unitaires et fonctionnels (débogage du site), ils mettent en ligne le site tout en gérant l'hébergement.

Webdesign :

MEDIASMART s'occupe de faire la création de charte graphique web, et de création de pages types. Elle peut faire aussi les animations sur les sites, comme faire un bandeau flash, des animations 3D. Et enfin elle peut aussi faire de l'E Marketing comme des bannières ou encarts publicitaires, réalisation de newsletter, …

E marketing / référencement :

MEDIASMART détermine des expressions de recherche, réalise et intègre des préconisations dans les pages du site, suit le positionnement du site, mais aussi des concurrents. Tout cela pour un référencement naturel. Elle fait aussi du référencement commercial en mettant en place des liens sponsorisés tout en faisant un suivi. Pour ce qui est du E marketing, elle réalise des campagnes de mailing, elle fait du marketing viral qui est une forme de publicité à la diffusion de laquelle le consommateur contribue.

Maintenance :

MEDIASMART réalise trois types de maintenance :

- Maintenance préventive : elle fait des supervisions serveurs, des supervisions applicatives (permet de connaître la disponibilité des machines en termes de services) et fait des tests sur l'environnement de production.
- Maintenance corrective : Elle fait des corrections de bug.
- Maintenance évolutive : Elle réalise une évolution des interfaces et fonctionnelles. Et peut aussi réaliser une mise à jour de versions.

Formation :

MEDIASMART propose ses services dans la formation de développeur. Elle propose différentes formations :

- Sur le développement en environnement LAMP (ensemble de logiciels libres permettant de construire des serveurs de sites Web).
- Sur le développement en PHP5 orienté objet avec l'aide du Framework Symfony.
- Sur la solution e-commerce de Magento.
- Sur la solution de gestion de contenu de Typo3.
- Sur le référencement naturel.

Elle peut aussi former des webmasters sur Magento et Typo3. Et enfin elle propose une formation a destinations des services marketing et commerciaux.

2. Solutions techniques

MEDIASMART utilise pour réaliser à bien les projets différents outils, différents systèmes et différentes solutions de développement.

Pour ce qui des outils utilisés en développement, MEDISAMART utilise deux Frameworks qui sont développés en PHP5, qui utilise le modèle MVC (Modèle Vue Contrôleur) :

- Symfony : C'est un Framework développé par une web Agency (appelée Sensio). Pour avoir plus de détails sur ce Framework, je fais une description de cet outil dans la partie « Présentation du Framework utilisé : Symfony ».
- Zend Framework : De même que Symfony, c'est un Framework développé par la société Zend, il a été réalisé dans le but de simplifier le développement tout en respectant les bonnes pratiques.

Pour ce qui est des systèmes utilisés, il y a Mac et linux pour ce qui des systèmes d'exploitation. Pour permettre de réaliser le site en local, MEDIASMART utilise l'environnement LAMP/XAMP, enfin il y a des serveurs mis en place pour permettre de tester les sites développés sur les différents navigateurs. Pour ce qui est de la gestion des données, les développeurs utilisent généralement MySQL comme système de gestion de base de données.

Enfin, ils utilisent différentes solutions d'Open Source, comme Magento ou Os-commerce pour le développement et la gestion d'un site e-commerce. Et ils utilisent Joomla et Typo3 comme système de gestion de contenu (appelé aussi CMS).

3. Les références

Enfin pour finir sur la présentation de l'entreprise, MEDISAMART a beaucoup de références en termes de clients. Ils ont tous types de client. Les demandes peuvent venir d'une ville/région, ou d'une association (exemple : Association à l'enlèvement parentale international => ALEPI) en passant par des petites entreprises, style PME (entreprise *votresommelier.com*), et en finissant par des grands noms, comme des grands centres commerciaux.

Voici quelques clients de MEDIASMART :

B. Enoncé du sujet de stage

Etude et réalisation de sites dynamique dites vitrine, d'un site communautaires et si assez de temps d'un site E-Commerce. Les sites seront réalisés en PHP/HTML à l'aide du Framework Symfony. Ces sites seront généralement accompagnés d'une base de données, et le système de gestion de base de données choisis et MySQL.

Si le site E-Commerce est développé, il sera développé à l'aide du CMS Magento.

III. Présentation technique des différents travaux

A. Présentation du Framework utilisé : Symfony

1. Présentation générale

Symfony est un Framework MVC libre écrit en PHP 5. Le projet a été lancé en 2005 par une web agency française, Sensio. Au début c'était pour ces propres besoins et ensuite elle a souhaité le faire partager à toute la communauté de développeur PHP. En tant que Framework, il facilite et accélère le développement de sites et d'applications Internet et Intranet. Symfony propose de nombreuses fonctionnalités :

- Avec le modèle MVC, cela permet de programmer en trois couches, et donc cela facilite la maintenance et l'évolution du code.
- Performances optimisées et un système de cache pour garantir des temps de réponse optimums.
- Une gestion d'URL rewriting performant, qui permet de formater l'URL d'une page indépendamment de sa position dans l'arborescence fonctionnelle.
- Un générateur de back office, qui permet aux clients de faire la gestion de son site sans rentrer dans le code directement. Un démarreur de module, c'est à dire de faire le lien entre le programme à développer et la base de données. Avec l'aide d'une ligne commande, un module ou le back office peut être généré.
- Symfony est multi langue, donc il y a un support pour la mise en place de l'internationalisation de son application web.
- Le support du JavaScript, ainsi que de l'AJAX. Par exemple des fonctions en PHP on été crée pour pouvoir développer en PHP et que cela doit être interpréter en AJAX.
- Une architecture facilitée, qui permet d'insérer facilement des plug-ins, mais aussi de créer son propre plug-in.

Le Framework a de nombreux avantages par rapport à d'autre. Le code est découplé, la configuration en cascade de ces modules permet de personnaliser facilement de nombreux paramètres. La documentation est complète et elle est mis à jour assez régulièrement, seul inconvénient c'est qu'elle est complète en anglais, et commence juste à s'internationaliser. L'apprentissage du Framework est facilité, avec deux bons tutoriaux en anglais, des livres, une API riche…

2. Présentation technique du Framework

a) Installation et structure d'un projet développé avec Symfony

Symfony peut s'installer de différentes façons, il y a l'installation via la bibliothèque PEAR, mais je vais vous parler de l'installation de Symfony avec l'archive directement téléchargés sur le site officiel de Symfony (http://www.**symfony**-project.org/).

Pour commencer il faut créer un répertoire pour enregistrer les fichiers de ton application web. Dans ce répertoire il faut ensuite créer un répertoire pour stocker les fichiers de la librairie du Framework Symfony. Exemple :

```
$ mkdir -p lib/vendor
```

Ensuite on récupère les fichiers de l'archive téléchargés précédemment, et on les met dans le dossier crée (dans cet exemple, c'est le dossier vendor). Suite à cela, les développeurs du Framework, on crée un fichier en PHP, qui permet de tester si la configuration du PC ou du serveur est correct. Ce fichier est exécutable en ligne de commande, et il se nomme : ***check_configuration.php***.

Si après l'exécution du script, aucune erreur n'est retournée, on peut vérifier que Symfony est bien installé en lançant cette commande :

```
$ php lib/vendor/symfony/data/bin/symfony -V
```

Enfin, on peut commencer à créer un projet. Dans Symfony les différentes applications partagent le même modèle de données et sont regroupées par projet. En général, on a besoin d'un frontend, pour le front office, et d'un backend, pour le back office. Alors pour créer son projet, il y a la commande Symfony : ***generate:project*** qui génère la structure par défaut des répertoires et crée les fichiers nécessaires.

La structure d'un projet créé à l'aide de Symfony :

Répertoire	Description
apps/	Contient les applications du projet
cache/	Les fichiers en cache
config/	Les fichiers de configuration du projet
lib/	Les librairies et classes du projet
log/	Les fichiers de log du projet
plugins/	Les plugins installés
test/	Les tests unitaires et fonctionnels
web/	Le répertoire racine web (fichier css, js, …)

Pour continuer dans son projet, il faut créer ces applications, la commande qui permet de faire cela est ***generate:app***. Voici un exemple de ligne de commande que l'on peut écrire :

```
$ php symfony generate:app --escaping-strategy=on --csrf-secret=Unique$ecret frontend
```

Comme pour la commande pour créer un projet, cette commande génère une structure par défaut dans le répertoire *apps/frontend* :

Répertoire	Description
config/	Les fichiers de configuration de l'application
lib/	Les librairies et classes de l'application
modules/	Le code de l'application
templates/	Les gabarits principaux

Pour que le projet soit accessible à partir d'un navigateur web, il ne faut pas oublié de configurer soit Apache directement, ou un VirtualHost. Pour la configuration, il faut mettre :

```
<VirtualHost *:80>
  DocumentRoot "/home/sfprojects/jobeet/web"
  DirectoryIndex index.php
  <Directory "/home/sfprojects/jobeet/web">
    AllowOverride All
    Allow from All
  </Directory>

  Alias /sf /home/sfprojects/jobeet/lib/vendor/symfony/data/web/sf
  <Directory "/home/sfprojects/jobeet/lib/vendor/symfony/data/web/sf">
    AllowOverride All
    Allow from All
  </Directory>
</VirtualHost>
```

b) *Construction d'une base de données dans un projet*

Donc tout d'abord, on peut utiliser soit MySQL, PostGreSQL, ou encore Oracle. Pour le rapport, on va utiliser le système de gestion de base de données MySQL. Alors avant d'utiliser les commandes Symfony, il faut créer une base de données en ligne de commande ou à l'aide d'une interface qui gère la base. Ensuite il nous faut indiquer à Symfony quelle base utiliser pour le projet :

```
$ php symfony configure:database --name=doctrine --class=sfDoctrineDatabase "mysql:host=localhost;dbname=jobeet" root mYsEcret
```

Le premier argument correspond au DSN PDO *(voir définition dans le glossaire),* le deuxième est le nom d'utilisateur et le troisième le mot de passe pour la base de données.

Après la liaison entre le projet et la base, il faut maintenant créer les tables, et les contraintes pour la base de données. Pour cela on va utiliser des fichiers écrit en YAML qui est un langage utilisé par l'ORM doctrine. Voici un extrait d'un fichier pour créer les tables, écrit en YAML :

```
JobeetCategory:
  actAs: { Timestampable: ~ }
  columns:
    name: { type: string(255), notnull: true, unique: true }
```

Après la création du fichier schema.yml, La commande ***doctrine:build-sql*** génère les déclarations SQL dans le répertoire data/sql, optimisées pour le moteur de base de données. Pour maintenant insérer les tables, il faut exécuter cette ligne de commande :

```
$ php symfony doctrine:insert-sql
```

Pour finir, la commande Symfony ***doctrine:build-model*** génère les fichiers PHP qui seront utilisés pour interagir avec la base de données dans le répertoire *lib/model.*

On peut gérer aussi l'insertion de données initiaux ou de données pour tester son application, on crée un fichier YAML pour chaque table utilisé :

```
JobeetCategory:
  design:
    name: Design
  programming:
    name: Programming
  manager:
    name: Manager
  administrator:
    name: Administrator
```

Puis pour les insérer dans la base de données, il faut utiliser : ***doctrine:data-load.***

c) Création d'un module pour une application

Depuis le début de l'explication du Framework, on a beaucoup utilisé les lignes de commande mais maintenant nous allons voir comment faire pour créer des pages web qui vont interagir avec la base de données. Pour cela, un projet est constitué de plusieurs applications, qui chacune d'entre elle est faite de module. Un module contient différents fichiers PHP qui représente une fonctionnalité d'une application du projet. Symfony est capable de générer automatiquement pour un modèle un module qui fournit des fonctionnalités basiques (créer, modifier, supprimer, …).

```
$ php symfony doctrine:generate-module --with-show --non-verbose-templates frontend job JobeetJob
```

La commande ***doctrine:generate-module*** permet de générer le module *job* dans l'application *frontend* sur le modèle *JobeetJob*. Voici la structure d'un module :

Répertoire	Description
actions/	Les actions du module
templates/	Les gabarits du module

d) Quelques commandes utiles pour un développeur en Symfony

Pour générer un back office par défaut pour la gestion de son site web :

```
$ php symfony generate:app --escaping-strategy=on --csrf-secret=UniqueSecret1 backend
```

Pour gérer l'administration d'un module créé dans une autre application :

```
$ php symfony doctrine:generate-admin backend JobeetJob --module=job
```

Pour vider le cache du projet :

```
$ php symfony cc
```

Pour réinstaller une base de données en une ligne de commande :

```
$ php symfony doctrine:build-all-reload
```

Pour installer un plug-in dans un projet :

```
$ php symfony plugin:install sfDoctrineGuardPlugin
```

Pour déployer le projet sur un serveur mutualisé :

```
$ php symfony project:deploy production --go
```

B. Site web à l'accompagnement international : TRUCLUC

1. Analyse du projet

Ce projet est le premier à avoir été fait pendant le stage après avoir passé un bon mois à apprendre à faire un site avec le Framework Symfony. Ce projet nous a permis de mettre en application ce qu'on a vu dans le tutoriel tout en personnalisant en fonction du cahier des charges que l'on a eu. Sur ce projet on était trois stagiaires à avoir travaillé dessus. Les deux autres étaient Thomas DELATTRE et Charles-Antoine LIVET. Pour commencer ce projet on a mis 4 semaines à le réaliser, à corriger les bugs d'affichage avec les différents navigateurs, et corriger les petits soucis lors de sa mise en ligne.

Ce projet consiste à réaliser un site d'aide à l'accompagnement à l'international, c'est-à-dire aider des personnes qui souhaitent aller à l'étranger pour proposer ou vendre ces services. Le but de ce site est de permettre, aux personnes qui sont intéressées, d'aller à l'étranger, d'avoir les meilleures informations possibles pour bien organiser leur voyage sans faire appel à une entreprise qui propose la même chose. Ce site comporte de nombreuses fonctionnalités comme un agenda mis en ligne, ou encore des nombreuses pages d'informations sur l'accompagnement à l'international. Ce site ne demande pas beaucoup de compétences en Symfony, mais demande surtout de l'organisation dans la réalisation du modèle conceptuel de données, car notre tuteur nous a demandé d'administrer pratiquement toutes les pages du site web.

Pour commencer, n'ayant peu de connaissance dans le domaine du graphisme des gabarits des pages, on nous a demandé de rechercher des Template sur un le site de Template Monster (**http://www.templatemonster.com**). Après avoir cherché chacun de son coté, on a trouvé un Template qui correspondait à ce que voulait notre tuteur, et qui correspondait bien au thème du site web.

Après avoir lu le cahier des charges, sur les différentes fonctionnalités à avoir sur le site, on a commencé à établir le schéma pour la base de données. Comme dit précédemment, le premier souhait était que pratiquement tous les textes des pages soient administrables. Pour tout ce qui des informations pour l'accompagnement, il devait avoir 6 catégories. Le titre et le résumé de chacune des catégories devaient être modifiables, et dans chacune des catégories on avait des sous catégories qui devaient être modifiables aussi. Chaque sous catégorie avait son titre et son contenu pour le détail. Pour cela on a fait une table « catégorie » et une table « sous catégorie » qui avait l'id de la catégorie comme clé étrangère.

Ensuite le second souhait est d'avoir un mini agenda mis en place sur le site, avec différents types d'événements à afficher. Pour chaque événement, il y avait la période, un titre, un résumé et le détail de l'événement. Si l'événement était encore d'actualité, il y avait la possibilité de se préinscrire à cet événement.

Enfin le dernier souhait était d'avoir la possibilité de gérer des publicités avec plusieurs formats possible (horizontale, verticale par exemple), sur différentes pages.

Pour finir, il y a aussi des fonctionnalités dites basiques dans un site web qui a été demandé, comme la gestion d'un formulaire de contact, ou encore la gestion de l'inscription à la newsletter.

2. Développement du site

a) *La mise en place des catégories et des sous catégories*

Comme dit dans l'analyse du projet, le site porte essentiellement sur l'information à l'accompagnement. Donc pour que cela soit plus clair sur le site, on a fait cela sur plusieurs pages, donc on a divisé cela en catégorie et en sous catégorie.

En sachant qu'on avait six catégories, on a décidé de faire cela dans un menu horizontale, et pour les sous catégories, en ne sachant pas le nombre de sous catégories possible, on a décidé de mettre en sous menus des catégories. Cela permet d'avoir la possibilité d'accéder aux deux sur les différentes pages du site. Pour gérer les sous catégories avec les catégories, on a décidé de stocker les deux séparément dans des tableaux, ensuite en parcourant les deux tableaux on teste si l'id de la catégorie dans les deux tableaux est identique, si oui on affiche le sous menu.

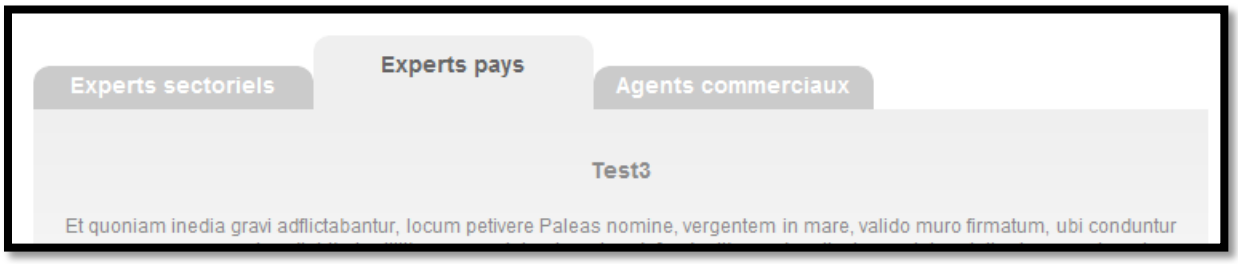
Voici un extrait du code pour afficher le menu avec un sous-menu qui se trouve dans ***apps/frontend/layout/_menu.php*** :

```
<?php foreach ($tab_cat as $cat): ?>
    <?php foreach ($tab_sscat as $sscat) : ?>
        <?php if ($sscat["idCategorie"]==$cat["id"]) : ?>
            <li colspan=3>
                <?php echo link_to($sscat["name"],
                "@show_news?idCategorie=".$sscat["idCategorie"].
                    "&nameCategorie=".$cat["url"].
                     "&id=".$sscat["id"].
                     "&name=".$sscat["name"]) ?>
            </li>
        <?php endif; ?>
    <?php endforeach; ?>
<?php endforeach; ?>
```

Pour ce qui est de l'appel des informations nécessaires pour le menu dans la base de données, les appels sont simples il suffit juste de faire une requête du style : ***SELECT * FROM sousCategorie ;***

Pour ce qui est du contenu des pages avec les informations demandées, on a utilisé un système d'affichage en onglet. C'est-à-dire que lorsque l'on clique sur une catégorie ou une sous catégorie, cela affiche les informations avec au dessus un menu en onglet avec les sous catégories correspondantes à la catégorie choisie. Quand on clique directement sur la catégorie, et non sur une sous catégorie, on affiche la sous catégorie choisi par défaut par l'administrateur du site.

Exemple d'affichage d'un menu en onglet :



L'interprétation du menu surélevé par rapport aux autres est en faite la récupération du nom de l'onglet courant dans l'url que l'on compare à l'onglet parcouru et auquel on applique la modification par rapport aux autres.

Pour finir cette partie, l'administration des catégories et des sous catégories est simple. On ne peut pas ajouter de catégorie ni en supprimer, à l'inverse des sous catégories. Par contre on peut modifier les catégories ou les sous catégories. Enfin pour rendre par défaut une sous catégorie dans une catégorie, il suffit de cliquer sur un lien pour rendre l'onglet par défaut au contraire des autres.

b) La mise en place de l'agenda

Pour ce qui est de l'agenda, on a décidé, avec l'accord du tuteur, de faire un agenda le plus simple possible, avec la possibilité de faire un tri avec différents critères. Ensuite pour l'affichage des différents résultats, il devait avoir 3 événements par page, n'afficher que les informations les plus importantes (période, type d'événement, titre de l'événement, le lieu, ainsi que le résumé de l'événement). Puis il avait la possibilité d'en savoir plus, avec le détail de l'événement. Pour cela il devait cliquer sur un bouton et cela l'ouvrait, à l'aide de la librairie JS Mootools et du plug-in Shadowbox, dans une fenêtre pop-up sans recharger la page. Pour la mis en place des différents programmes pour le pop-up, j'ai dû télécharger sur le site du plug-in Shadowbox (**http://www.shadowbox-js.com**) tous les fichiers nécessaires, puis mettre les fichiers JS dans le répertoire *web/js*, et le fichier CSS dans *web/css*. Le plus gros souci qu'on a rencontré avec cette librairie, c'est qu'à un autre endroit du site, la librairie JQuery est appelée. Entre ces deux librairies il y a un conflit, donc pour le résoudre, il fallait faire :

```
var $j = jQuery.noConflict();
```

Ensuite pour ce qui est du tri par critère, on a dû faire deux types de tri :

- Le tri par type d'événement, c'est-à-dire l'utilisateur a le choix de cocher entre plusieurs types d'événement.
- Le tri par événement à venir ou événement passé.

Quand l'utilisateur va valider ces choix, les résultats sont stockés dans des sessions. On a choisi les sessions car comme dit précédemment, on affiche trois résultats par page, et donc pour gérer entre les différentes pages le tri par critère c'était la manière la plus simple à gérer. Quand l'utilisateur choisit ces critères, on supprime tout ce qu'il y a dans les sessions utilisés pour l'agenda, ensuite on met le(s) choix des types d'événements dans une session, et le choix pour l'événement à venir ou pas dans une autre session. Puis après avoir fait tout cela, on fait les requêtes nécessaires pour mettre en place les résultats dans une pagination :

```
$this->pager = new sfDoctrinePager('InternationalisationAgenda', sfConfig::get('app_max_events_on_agenda'));
    if(is_null($events))
    {
    $this->pager->getQuery()
    ->from('InternationalisationAgenda a')
    ->whereIn('a.type', $types)
    ->orderBy('a.date_debut DESC');
}       else
    {
        if($events == '0')
        {
            $this->pager->getQuery()
        ->from('InternationalisationAgenda a')
        ->whereIn('a.type', $types)
        ->andWhere('a.date_debut > ?', date('Y-m-d h:i:s', time()))
        ->orderBy('a.date_debut ASC');
        }
        else
        {
            $this->pager->getQuery()
        ->from('InternationalisationAgenda a')
        ->whereIn('a.type', $types)
        ->andWhere('a.date_debut <= ?', date('Y-m-d h:i:s', time()));
        }
    }
$this->pager->setPage($this->getRequestParameter('page',1));
$this->pager->init();
```

Enfin, comme dit dans l'analyse du projet, pour chaque événement à venir, l'utilisateur peut en savoir plus sur l'événement, et en plus il a la possibilité de se préinscrire. Dans le pop-up qui va s'ouvrir, il y a un bouton qui permet d'envoyer sur une page avec un formulaire à remplir. En haut de la page pour la préinscription, on peut trouver des informations importantes concernant l'événement, comme la période et le titre. Après avoir bien rempli le formulaire, cela envoie un message à l'utilisateur qui se préinscrit et cela rajoute une ligne dans la base de données.

Dans la partie back-office, il y a la gestion des préinscriptions. C'est-à-dire il affiche toutes les préinscriptions faites pour chaque événement présent. Et pour chaque préinscrit, l'administrateur peut l'inscrire définitivement à l'événement, ou le supprimer de la liste. Pour la gestion de l'inscription il y a un champ booléen de la base de données, qui se met à « true » quand l'administrateur clique sur le lien pour inscrire l'utilisateur préinscrit. Pour terminer, il y a la possibilité de faire un export CSV de toutes les préinscriptions faites.

c) La gestion de la publicité du site

Pour le site, il était demandé de faire une gestion de la publicité pour différentes pages. Les pages concernées sont :

- La page d'accueil (publicité horizontale)
- La page des catégories et sous catégories (publicité verticale)
- La page contact (publicité verticale)
- L'agenda (publicité horizontale)

Il était aussi demandé que pour chaque page il y ait plusieurs publicités, et que lorsque l'utilisateur va cliquer sur la publicité cela doit l'emmener vers un autre site.

Pour faire la gestion de la publicité, on a décidé de stocker dans la base de données, le titre de la publicité, sur quelle page il faut l'afficher, le nom du fichier, et le lien du site de la publicité.

Pour la gestion de l'affichage, c'est-à-dire le changement de publicité sans rechargement de page, on a récupéré un plug-in de la librairie Mootools qui s'appelle smoothGallery. Pour l'installer c'est la même façon que la shadowbox, et en plus c'est configurable, comme pour le temps entre les images.

```
function startGallery() {
    var myGallery = new gallery($('myGallery'), {
        timed: true,
        showArrows: false,
        showInfopane: false,
        showCarousel: false,
        delay: 5000,
        fadeDuration: 1400

    });
}
window.addEvent('domready',startGallery);
```

Pour ce qui est de la gestion de la publicité dans la partie administrable, on affiche par page de site les publicités déjà mis en place avec le lien du site de la publicité et son titre. Comme on a pu le voir précédemment, il y a différents formats de publicité pour différente page, donc quand l'administrateur va vouloir ajouter ou modifier sa publicité il faut vérifier la hauteur et la largeur de la publicité :

```
    $files = $this->getRequest()->getFiles();
    $page = $request->getParameter('internationalisation_publicite[page]');
$infos_img = getimagesize($files['internationalisation_publicite']['tmp_name']['image']);
    if($page=='categorie' || $page=='contact' || $page=='agenda')
    {
        if($infos_img[0] <= sfConfig::get('app_largeur_pub_verticale') && $infos_img[1] <= sfConfig::get('app_hauteur_pub_verticale'))
            $this->processForm($request, $this->form);
        else
            $this->internationalisation_publicite = 'L\'image ne doit pas dépasser '.sfConfig::get('app_largeur_pub_verticale').
            'px en largeur et '.sfConfig::get('app_hauteur_pub_verticale'). 'px en hauteur';
    }
```

d) Autres fonctionnalités du site

Pour finir sur ce projet, je vais vous parler des fonctionnalités dites basiques présent sur le site.

Pour commencer il y a un formulaire qui permet de contacter l'administrateur du site, c'est à dire la personne qui s'occupe de gérer l'accompagnement à l'international. Ce formulaire demande pas mal d'information, et donc il y a un script en JavaScript qui permet de vérifier sir les champs obligatoires sont bien remplis et corrects, après cette vérification cela envoi un mail à l'administrateur.

Ensuite, il y a la gestion des inscriptions et désinscriptions à la newsletter. L'utilisateur rajoute son Email, et choisi s'il souhaite s'inscrire ou se désinscrire. Toutes ces actions sont faites sans rechargement de la page, donc à l'aide de l'AJAX. Après dans le back office, il y a la possibilité d'exporter en CSV tous les inscrits.

Enfin pour finir, tous les textes, les coordonnées, l'édito est modifiable. Pour que l'administrateur puisse mettre en place sont texte comme il souhaite sans taper une ligne d'HTML, on a utilisé un éditeur de texte avancé déjà créé. Le nom de cet éditeur est le **fckEditor**. Pour le mettre en place il y a un plug-in déjà réalisé qui était en ligne sur le site de Symfony. Donc on télécharge le plug-in, on le met en place dans le projet, et on appelle l'éditeur avancé dans le fichier *lib/form/doctrine/myForm.class.php* :

```
$this->widgetSchema['my_editor'] = new sfWidgetFormFCKEditor();
```

C. Site web de l'association contre l'enlèvement parental international : ALEPI

1. Analyse du projet

Pour le deuxième projet, il m'a pris un plus de temps car plus de fonctionnalité demandée et il fallait vraiment se perfectionner sur certaine chose dans le Framework Symfony, du genre il fallait que j'étudie en particulier un plug-in qui faisait la gestion d'utilisateur avec des groupes d'utilisateur. Mon projet consistait à faire une partie extranet sur un site vitrine d'une association. Cette association travail contre l'enlèvement parental international. Le but du projet est d'avoir un extranet qui permet à des parents victimes qui ont subi un enlèvement de leur enfant par l'autre parent, de remplir un dossier directement en ligne, de remplir les différents profils utiles pour la justice, de se faire aider pour les différentes démarches, par des personnes qualifiées qui sont consultants pour l'association.

Pour commencer, il fallait faire une étude détaillée des différents groupes d'utilisateur possible pour le site. Les différents groupes sont :

- **Les parents victime :** Ce groupe d'utilisateur concerne les utilisateurs qui ont été victime d'un enlèvement de leur enfant par leur conjoint ou Ex-conjoint. Pour appartenir à ce groupe, il faut faire une demande dans un formulaire de contact, suite a cela l'administrateur (détaillé plus loin dans le rapport) lui envoie un lien qui va lui permettre de remplir un formulaire avec différentes informations à donner. Après avoir envoyé le formulaire, l'administrateur peut voir un dossier qui est en cours de validation et s'il le valide, le parent reçoit ces identifiants pour se connecter. Après connexion, il peut remplir son profil, le profil des enfants enlevés, le profil du parent qui a enlevé l'enfant, le profil des avocats, la situation maritale et dans quel contexte s'est passé l'enlèvement. Il peut aussi remplir un dossier que je vous détaillerai dans la partie suivante.

- **Les consultants :** Ce groupe d'utilisateur est fait pour les personnes qui travaillent pour l'association. C'est-à-dire des personnes qui travaillent dans la justice, et qui pourrait aider en termes de loi les parents victimes. Les consultants sont en faites des chargés de dossiers. Leur inscription c'est l'administrateur qui le gère, il doit rentrer son nom, prénom et son adresse Email. cela va créer un compte et envoi un mail au consultant concerné avec les identifiants. Pour qu'il s'occupe d'un dossier d'un parent victime, il faut que l'administrateur lui attribue un dossier. Après il aura les mêmes droits qu'un parent victime pour remplir le dossier. Il peut aussi voir les autres dossiers en cours mais ne peut rien faire dessus. Il a son profil aussi à remplir, et il peut voir tous les utilisateurs inscrits sur le site.

- **Les administrateurs :** Ce groupe d'utilisateur est composé des fondateurs de l'association. C'est-à-dire du président, des associés mais aussi d'une secrétaire. Le rôle de l'administrateur est de tout gérer dans la partie extranet, pour ce qui est de la gestion du site vitrine il y a le back office de mis en place et donc ce n'est pas à lui de le faire. Il gère toutes les inscriptions, que ce soit d'un parent victime ou d'un consultant, il gère tous les dossiers, il peut les clôturer, les suspendre, les ouvrir. Il gère tous les utilisateurs, il peut les supprimer, il peut modifier leur profil, il peut mettre un dossier à un consultant. Enfin il a accès à toutes les messageries des différents dossiers.

Suite à cela, il fallait faire différentes fonctionnalités pour les utilisateurs. Pour commencer j'ai étudié les dossiers (avec mon tuteur de stage) pour les parents victimes. Donc on a décidé, de séparer le dossier en quatre :

- La partie historique : Cette partie du dossier concerne tous les événements qui ont pu se dérouler lors de l'enlèvement de l'enfant comme une discussion avec le parent qui a enlevé l'enfant.
- La partie action : Cette partie du dossier concerne toutes les actions réalisées pour le dossier, les actions peuvent être envoyées aux consultants par mail, comme par exemple appel avec l'avocat du parent qui a enlevé l'enfant.
- La partie profil : Comme détaillé précédemment, c'est dans cette partie qu'on remplie tous les profils utiles pour les papiers.
- La partie messagerie : Permet aux utilisateurs concernés par le dossier d'échanger des messages entre eux.

Pour les administrateurs et les consultants, il y a la possibilité de remplir une sorte de bloc note pour indiquer qu'est ce qu'ils ont à faire. Et quand ils ont réalisés leur tâche ils peuvent indiquer qu'elle a été faite.

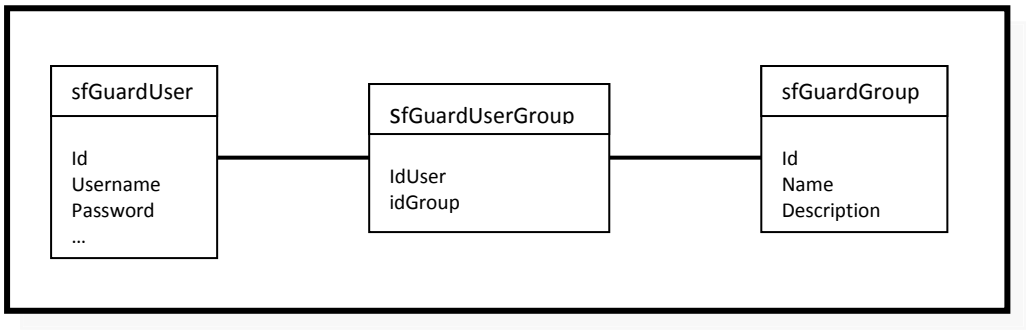
Pour finir, dans l'analyse du projet, pour ce qui est de la partie site vitrine du site, et de la partie back office ce n'est pas moi mais les autres stagiaires qui s'en sont occupés.

2. Développement du site

a) La gestion des utilisateurs

Comme dis dans l'analyse du projet, cette partie m'a pris pas mal de temps. Car j'ai dû étudier à fond un plug-in qui gère les utilisateurs. J'ai dû faire attention que certains utilisateurs n'ont pas les droits par rapports à d'autres utilisateurs. J'ai dû faire aussi la gestion du changement de mot de passe et lorsque que l'utilisateur perdait son mot de passe. Puis faire aussi la gestion des inscriptions et suppressions des utilisateurs.

Pour commencer, la gestion des différents groupes d'utilisateur a été particulièrement complexe. Car en utilisant le plug-in sfDoctrineGuardPlugin, j'ai dû utiliser trois tables pour réaliser la gestion des groupes. Voici un bref schéma des trois tables :



Alors comme on peut voir sur le schéma, dans la table ***sfGuardUserGroup*** elle a comme clé étrangère l'id de la table ***sfGuardUser*** et l'id de la table ***sfGuardGroup***. Et les deux champs id sont des clés primaires de la table.

Ensuite pour gérer les droits en fonction en fonction du groupe de l'utilisateur, j'ai dû réaliser des tests dans chaque action des modules pour savoir déjà s'il était bien connecter avec :

```
<?php if($sf_user->isAuthenticated() ): ?>
```

Cette fonction permet de tester si l'utilisateur est connecté, elle va en faite regardé si la session de l'utilisateur n'est pas vide. Ensuite je regarde dans quel groupe l'utilisateur appartient, le nom du groupe est comme la fonction **isAuhtenticated()** c'est stocké dans la session de l'utilisateur :

```
<?php if($sf_user->hasGroup('chargés de dossier') ||
```

Pour ce qui est du changement de mot passe, j'ai dû faire plusieurs vérifications avant que l'utilisateur puisse changer. La première vérification est l'ancien mot de passe, ensuite le nouveau mot de passe doit faire plus de 6 caractères, puis la confirmation du nouveau mot de passe et enfin l'adresse mail rentrée dans le profil de l'utilisateur. Pour toutes ces vérifications j'ai utilisé l'AJAX avec la librairie JQuery. Voici un exemple d'appel pour utiliser l'Ajax en JQuery :

```
//Appel de la fonction pour utiliser l'ajax
$j.ajax({

    type: "POST", //Indique quel méthode utiliser dans le formulaire
    url: "/utilisateur/getPassword", // La fonction PHP utilisé pour vérifier l'ancien mot de passe
    data : "exPassword="+value, // Les paramètres pour la fonction PHP
    success:function(data){ // Quand la fonction PHP retourne un résultat
                if(data==1) //Si c'est le bon mot de passe
                {
                    info_exPassword.empty();
                    info_exPassword.append("<img src='/images/tick.png' />");
                    alfa=true;
                }
                else // Si c'est le mauvais mot de passe
                {
                    info_exPassword.empty();
                    info_exPassword.append("<img src='/images/delete.png' /> Le mot de passe n'est pas bon");
                    alfa=false;
                }
    }
```

Enfin pour ce qui est de l'inscription des utilisateurs c'est l'administrateur qui gère tout. Pour un parent victime, quand il reçoit un mail avec une demande d'ouverture de dossier pour un nouvel enlèvement d'enfant il rentre l'email et ça envoie aux parents victime le lien pour remplir le formulaire de préinscription, après validation du formulaire l'utilisateur sera créé.

Pour ce qui est des consultants, cela se passe de la même façon, le consultant qui souhaite intégrer l'association doit remplir un formulaire après cela, l'administrateur doit remplir le nom, le prénom et l'adresse mail. Après avoir rempli tout ca, cela va créer le pseudo avec la première lettre du prénom et le nom du consultant et faire un mot de passe aléatoire avec la fonction **rand()** de PHP et l'adresse email du consultant. Après cela envoi des identifiants par mail au consultant concerné. Pour ce qui est des administrateurs cela ce passe de la même manière qu'avec les consultants.

Pour finir tous les utilisateurs ont un profil à remplir. Dans chaque profil on peut ajouter des photos d'identité (200x200). Les administrateurs et les consultants peuvent voir tous les profils mais seuls les administrateurs peuvent modifier les profils de tout le monde.

b) La gestion des dossiers

Pour les dossiers, on va diviser cela en quatre parties. Je vais commencer par vous parler par la demande d'ouverture de dossier, ensuite je vous expliquerai comment sont faits les historiques du dossier, les actions du dossier et des profils des différentes personnes qui interviennent pour le dossier.

Pour la préinscription, le parent fait la demande auprès d'un formulaire de contact, après avoir bien rempli le formulaire cela va envoyer un mail à l'administrateur pour, comme dit précédemment, qu'ensuite lui génère un lien pour remplir un formulaire détaillé avec de nombreuses informations sur le parent victime, mais aussi sur le ou les enfants enlevés. Le lien que cela va générer est un lien avec un token. C'est-à-dire le token est une chaine de caractère généré aléatoirement pour éviter que tout le monde ait accès à ce formulaire et donc éviter des préinscriptions intempestive. Voici un exemple de lien qui peut être envoyé à l'utilisateur :

http://www.alepi.com/preInscription/5cd02cc38d726469e01bdcdb9805455424fbb4a6

Quand le parent va cliquer sur le lien ca va l'envoyer directement sur le formulaire de préinscription. Je vérifie que le token qui a dans l'url soit le même qui est dans la base de données. Après que le parent est bien rempli le formulaire, tous les champs rentrés vont être stockés dans la base de données, cela va créer un utilisateur qui appartient au groupe « parent victime ». Pour ce qui est de l'username cela va faire comme avec les consultants et les administrateurs (***voire plus haut***), mais comme l'username doit être unique pour éviter les doublons, j'ai crée un petit programme que temps le pseudo existe je rajoute deux chiffres aléatoire derrière le pseudo (Exemple : cmoysan76).

```
$information = Doctrine::getTable('sfGuardUser')->getInfoParUsername($username);
//Regarde si il y a déjà un utilisateur avec ce pseudo

while($information != '') //Le temps qu'il ya le même pseudo
{
    $username = $username.rand(0,99); // Concaténation de deux chiffes au hasars après le username
    $information = Doctrine::getTable('sfGuardUser')->getInfoParUsername($username);
}
```

Par contre les identifiants de connexion ne sont pas donnés au parent. C'est crée mais il les recevra que quand le dossier sera validé par un administrateur. Concernant les dossiers en attente de validation, l'administrateur a trois possibilités :

- Accepter le dossier : C'est mettre le dossier en statut « En cours » et envoyer les paramètres de connexion au parent.
- Demander plus d'information : Le dossier reste en statut « En attente de validation » et cela va ouvrir un champ texte pour que l'administrateur, par mail, puisse demander aux parents les informations souhaitées.
- Refuser le dossier : Le dossier va être supprimé, l'utilisateur aussi, et envoyer un mail à l'utilisateur.

Ensuite dans le dossier il y a les historiques à remplir. Les historiques correspondent en faite à tous les événements qui ont pu se passer pendant l'ouverture du dossier. Sur le site, dans un historique on retrouve la date ou la période, le titre de l'historique, le résumé et si l'utilisateur le souhaite mettre un commentaire à cet événement. Tout historique est modifiable et supprimable. Pour l'affichage de tous les historiques, l'utilisateur peut choisir entre 3, 5 ou 10 résultats par page. Il peut aussi les trier par du plus récent au plus ancien ou l'inverse. Pour ces deux fonctionnalités, j'ai utilisé la méthode qu'on a utilisé dans le projet précédent pour afficher les événements de l'agenda (Voire partie ***III / B) / 2. / b***)) c'est-à-dire de stocker tous les choix de l'utilisateur dans une session. Enfin il y a la possibilité d'exporter tous les historiques dans un document PDF. Pour cela j'ai dû utiliser Zend Framework pour Zend-Pdf. Pour le mettre dans Symfony c'est simple il suffit de le mettre dans le même dossier que la librairie de Symfony, et mettre dans le fichier de configuration ***projectConfiguration.class.php*** qui se trouve dans le dossier *config/* :

```php
static public function registerZend()
{
  if (self::$zendLoaded)
  {
    return;
  }

  set_include_path(sfConfig::get('sf_lib_dir').'/vendor'.PATH_SEPARATOR.get_include_path());
  require_once sfConfig::get('sf_lib_dir').'/vendor/Zend/Loader.php';
  Zend_Loader::registerAutoload();
  self::$zendLoaded = true;
}
```

Puis pour l'appeler dans la fonction où l'on en a besoin, il faut mettre :

```php
// Appel à la librairie Zend, pour la création du PDF
ProjectConfiguration::registerZend();
$documentPdf = new Zend_Pdf();
```

Donc après pour ce qui l'écriture du document PDF, j'ai dû utiliser la documentation de Zend, car en faite il faut tout placer par soi-même. Mais après étude ceci s'est fait assez facilement.

Pour ce qui est des actions sur le dossier, c'est en faite tout ce qui doit être ou ce qui a été fait par le parent ou les personnes qui s'occupent du dossier. Pour la gestion et l'affichage c'est la même chose qu'avec les historiques sauf qu'il n'y a pas le champ commentaire, et en faite il peut choisir un destinataire. Quand il veut ajouter ou modifier une action il a la possibilité d'envoyer le détail de l'action aux destinataires par mail, et lui reçoit une copie.

Enfin il y a les différents profils dans le dossier, c'est-à-dire pour que le dossier soit complet il faut des informations sur différentes personnes concernées par l'enlèvement. Donc on demande au parent de remplir des informations sur lui, le(s) enfant(s) qui ont été enlevé, le parent qui a enlevé l'enfant, l'avocat du parent, l'avocat du parent « rapteur », la situation maritale et dans quel contexte l'enfant a été enlevé. Toutes ces informations sont ensuite stockées en base et affichés sur différentes pages où le parent peut naviguer avec un menu horizontal mis en place sous chaque profil. Enfin le consultant ou un administrateur peut l'exporter en document PDF.

Pour faciliter la navigation pour les consultants et les administrateurs, sur une page du site on peut voir des informations sur le dossier sur le parent victime, et deux menus sont mis en place :

- Un menu avec la messagerie, les historiques, les actions et les profils. Quand il va naviguer entre ces différentes pages, il pourra changer de pages sans revenir à la page d'accueil du dossier avec la mis en place du même menu sur les différentes pages.
- Un menu pour exporter les historiques, les actions, et les profils en document PDF.

c) *La gestion de la messagerie*

Pour terminer sur ce projet, je vais vous parler de la mise en place d'une « mini » messagerie. Cette messagerie a été mise en place dans le but de discuter entre personne qui s'occupe d'un dossier. Pour chaque dossier, il y a une messagerie mis en place. Il y a aucune liaison entre les différentes messageries.

Quand un utilisateur va vouloir poster un message, il doit remplir l'objet du message, le contenu du message et il doit choisir le destinataire du message. Il a aussi la possibilité de rajouter son message en intégralité dans les actions du dossier en question. Quand il valide son formulaire, le message est stocké dans la base de données avec l'id du dossier en question, l'id de l'utilisateur et l'id du destinataire. Juste après, cela va créer un nouveau message vide avec le même objet, l'auteur devient le destinataire, et le destinataire devient l'auteur. C'est fait pour que l'autre personne puisse répondre au premier message. Pour avertir l'autre personne ca envoi par mail le message de la première personne et en même temps un lien du même style que la préinscription c'est-à-dire avec un token. Quand il va cliquer sur le lien cela va lui renvoyer soit sur le formulaire pour répondre et donc mettre à jour le message vide s'il est connecté, sinon il doit ce connecter et après connexion cela lui renvoie directement sur la bonne page et non sur la page par défaut lorsque l'on se connecte autrement.

Pour gérer cela j'ai dû modifier le plug-in ***sfGuardDoctrinePlugin*** de Symfony, et donc j'ai créé une fonction quand on se connecte pour la messagerie et une autre autrement. Alors pour tester si c'est la bonne personne je récupère le token et l'id du message qu'il y a dans l'URL. Avec l'id du message je regarde si c'est le bon auteur, si oui j'accepte qu'il réponde aux messages, sinon je renvoie le formulaire de connexion.

```php
$interlocuteur = Doctrine::getTable('AlepiMessagerie')->getAuteur($user->getAttribute('id')); // Récupere l'auteur

$remember = isset($values['remember']) ? $values['remember'] : false; // Sauvegarde en session de la connexion
 // Si c'est le bon auteur, on le connecte pour qu'il puisse répondre
if($values['user']['id'] == $interlocuteur['auteur'])
{
    $user->signin($values['user'], $remember);
}
//Redirection vers la page pour la réponse, si connecter, sinon vers le formulaire pour la connexion
$this->redirect('messagerie/nouvelleReponse?id='.$user->getAttribute('id').'&token='.$request->getParameter('token'));
```

Quand la personne va répondre après avoir rentrée ces paramètres d'identification, cela refait la même chose que lorsqu' on poste un nouveau message.

Pour chaque messagerie, j'affiche sur la page tous les messages qui ont été postés, triés par objet et par date. Chaque message est modifiable ou supprimable par son propriétaire, l'administrateur peut modifier ou supprimer tous les messages.

Pour l'administrateur et les consultants, quand ils vont cliquer dans le menu sur « Messagerie », cela va afficher une page avec les différents dossiers avec la date et l'auteur du dernier message posté.

IV. Conclusion

Pour finir sur le rapport, les deux projets étudiés et développés ont permis de mettre en application ce qu'on a pû voir au début du stage. Le site TRUCLUC est un site plus vitrine que le site ALEPI, qui a demandé plus d'approfondissement, plus de réflexion dans la réalisation de ce projet. L'avantage des deux sites est qu'ils peuvent être améliorés, il y a la possibilité de rajouter de nouvelles fonctionnalités, d'optimiser le code, … Mais le plus gros problème rencontré dans les deux c'est lors de la mise en production il y a eu des soucis que ca soit graphique (problème avec différents navigateurs) ou technique (correction de code en fonction du serveur), donc pas mal de perte de temps pour des corrections de bugs.

Pour conclure, je vais vous faire mon bilan personnel, le stage m'a beaucoup apporté. C'est-à-dire par rapport à mon dernier stage réalisé pour la fin de mon cursus scolaire en DUT informatique pour l'IUT de Lens, j'ai pu travailler en équipe avec deux autres stagiaires. Cette expérience m'a permis à apprendre à être solidaire, à aller aider les personnes qui sont en difficultés, et à me faire aider quand moi j'étais bloqué sur quelque chose. Cela m'a permis aussi d'améliorer mon autonomie, car durant le stage je n'ai fait que d'apprendre de nouvelle chose, et dès le début j'ai dû apprendre par moi-même pour pouvoir continuer et développer pendant mon stage.

Sur le plan technique, j'ai appris de nouvelle chose mais j'ai pu aussi mettre en application ce que l'on avait vu pendant toute l'année. Pour mes nouvelles connaissances, j'ai pu apprendre une nouvelle façon de développer, même si cela ressemble beaucoup au Framework vu pendant l'année, Symfony est encore différent dans son organisation que le Zend Framework. La ressemblance est que c'est développé à l'aide du modèle MVC qui est très utile pour la maintenance ou une mis à jour des sites. J'ai pu aussi utiliser Zend Framework car il est possible de l'intégrer dans le Framework de Symfony. Autrement j'ai pu aussi revoir les bases du développement de site web, et me perfectionner dans le design d'un site web (CSS). J'ai ensuite retravailler à l'aide du dépôt SVN, cela m'a permis de me perfectionner dans ce domaine comme avec la configuration d'Apache. Pour finir j'ai approfondi mes compétences dans l'utilisation de JavaScript et de l'AJAX en utilisant différentes librairies.

V. Glossaire

DSN PDO : Le DSN contient les informations nécessaires pour se connecter à la base. Et PDO est un driver qui permet de faire la liaison entre le PHP et le système de gestion de base de données.

MVC : Le **Modèle-Vue-Contrôleur** (en abrégé **MVC**, de l'anglais *Model-View-Controller*) est une architecture et une méthode de conception qui organise l'interface homme-machine (IHM) d'une application logicielle.

Framework : un Framework est un ensemble de bibliothèques, d'outils et de conventions permettant le développement d'applications.

PEAR : (pour *PHP Extension and Application Repository*) est une collection de bibliothèques PHP. C'est aussi une application qui permet de gérer les bibliothèques (installer ou mettre à jour une bibliothèque).

CMS : Un système de gestion de contenu ou SGC ((en) *Content Management Systems* ou CMS) est une famille de logiciels destinés à la conception et à la mise à jour dynamique de site web ou d'application multimédia.

VI. Liens utiles

- http://www.symfony-project.org/ : Site officiel du Framework de Symfony, on y retrouve des tutoriels, documentation, plug-in, … (site en anglais).
- http://www.developpez.net/forums/f663/php/bibliotheques-frameworks/symfony/ : Communauté de développeurs en Symfony, très utile lorsque qu'on est bloqué sur quelque chose.
- http://www.php.net/manual/fr/ : Documentation sur le langage PHP.