

RAPPORT DE PROJET

GESTION DE DOSSIERS ADMINISTRATIFS

Groupe 13

Tuteur : Damien GENTHIAL

Cliente : Marion BERTHOZ

Mathie BERTHOLET

Thibault GRANADA

Mathis KLIMCZAK

Théo ZURCHER

2015-1016

RAPPORT DE PROJET

GESTION DE DOSSIERS
ADMINISTRATIFS

Table des matières

| | |
|---|-----------|
| Introduction..... | 5 |
| I) Présentation et objectifs principaux du projet..... | 7 |
| I.A) Contraintes..... | 7 |
| a. La base de données | 7 |
| b. L'intégration à l'intranet..... | 8 |
| c. Symfony..... | 8 |
| I.B) Le cahier des charges et les objectifs fixés..... | 8 |
| a. Le cahier des charges | 8 |
| b. Objectifs..... | 9 |
| I.C) Propositions et modifications du cahier des charges | 10 |
| a. Modifications..... | 10 |
| b. Propositions..... | 10 |
| II) Elaboration et mise en œuvre du projet..... | 11 |
| II.A) Les différentes phases..... | 11 |
| a. Phase de conception | 11 |
| b. Phase de développement..... | 12 |
| c. Phase de rédaction et modification..... | 12 |
| II.B) Les choix techniques | 13 |
| a. Un choix d'intégration à l'intranet..... | 13 |
| b. Les technologies qui en découlent | 13 |
| c. Les modules de Symfony2 | 14 |
| II.C) Répartition du travail au sein du groupe..... | 17 |
| II.D) Problèmes et solutions envisagées..... | 18 |
| a. Difficultés d'installation de Symfony2 | 18 |
| b. Le développement collaboratif..... | 19 |
| III) Résultats et perspectives..... | 20 |
| III.A) Etat final et problèmes non résolus..... | 20 |
| III.B) Développement futur..... | 23 |
| Conclusion | 26 |
| Glossaire | 27 |
| Table des illustrations | 28 |

Introduction

Nous avons été amené dans le cadre du projet de deuxième année de DUT informatique à réaliser une application web pour Mme Marion Berthoz, travaillant au sein du service administratif de l'IUT.

L'administration de l'IUT de Valence gère tous les dossiers administratifs de chaque département et de chaque vacataire qui travaille dans l'IUT. Ces vacataires ayant des situations professionnelles diverses, doivent fournir différentes pièces et ainsi constituer un dossier administratif.

Or le système de gestion n'est pas automatisé ce qui a pour conséquence une perte de temps non négligeable et peut aussi entraîner de nombreuses erreurs. Mme Marion Berthoz a pour charge de recontacter par mail les vacataires n'ayant pas fourni toutes les pièces demandées afin de constituer leurs dossiers.

Ce projet a pour but de simplifier et d'automatiser les envois de mails envers ces vacataires. Pour cela, nous avons donc créé une application web intégrée à l'intranet afin d'avoir un aperçu des dossiers des vacataires et ainsi automatiquement les recontacter en cas de nécessité.

Dans le cadre de ce projet, nous avons travaillé pour Mme Marion Berthoz dans le but de répondre à ses besoins vis-à-vis des relances de mails. Notre équipe, composée de Mathie Bertholet, Thibault Granada, Mathis Klimczak et Théo Zurcher, a travaillé sous la tutelle de M. Damien Genthial, professeur d'informatique et responsable du CRI à l'IUT de Valence, pour la conception de ce projet.

Dans ce rapport nous nous poserons les questions suivantes :

Quelles ont été les solutions proposées pour répondre aux besoins d'automatisation des relances de mails, et comment avons-nous mis en place cette solution ?

Nous aborderons tout d'abord les objectifs généraux du projet et le cadre initial dans lequel le projet a été pensé afin de montrer comment nous sommes passés du premier cahier des charges au développement de notre application de gestion de dossiers administratifs.

Nous poursuivrons sur la mise en œuvre du projet avec la présentation des outils utilisés, la répartition des tâches au sein du groupe, les problèmes rencontrés lors de la conception ainsi que les différentes phases de développement.

Enfin, nous verrons quels sont les résultats de notre projet, les problèmes non résolus et les éventuels prolongements possibles au développement de l'application.

I) Présentation et objectifs principaux du projet

Cette partie présentera le projet dans sa globalité et exposera les contraintes instaurées et les solutions proposées.

Nous verrons tout d'abord quelles ont été les contraintes liées à ce projet puis nous résumerons notre cahier des charges et les modifications qui lui ont été apportées.

Finalement, nous aborderons les objectifs fixés et les propositions de solutions faites.

I.A) Contraintes

Dans cette partie nous traiterons des différentes contraintes qui ont été imposées au début du projet.

Nous verrons notamment les contraintes imposées par la base de données stockée sous forme de fichiers excel, de l'utilisation de Symfony pour le développement et l'intégration de notre application à l'intranet.

a. La base de données

La première contrainte que nous avons eue a été de récupérer les données stockées dans les fichiers excel servant de base de données.

Nous avons dû décider si nous réutilisions ces fichiers excel ou si nous faisons une application vierge de toutes données.

La deuxième décision associée à la base de données a été de choisir quelles sont les données à extraire, nous avons choisi les noms, les formations, les adresses mail et la civilité de chaque vacataire. Nous avons autorisé que certains vacataires puissent être intégrés à la nouvelle base même sans adresse mail pour les contacter ou sans prénom.

Nous avons dû trouver un moyen pour extraire ces données.

b. L'intégration à l'intranet

La deuxième contrainte a été l'intégration à l'intranet de l'IUT pour des raisons de simplicité et d'accessibilité. L'utilisateur n'a besoin de s'authentifier qu'une seule fois afin de pouvoir utiliser l'application et de l'intranet lui-même.

c. Symfony

La dernière contrainte imposée a été l'utilisation du framework Symfony pour développer notre application web. Symfony a été utilisé car nous voulions intégrer notre application à l'intranet lui-même conçu grâce à ce framework. Symfony a permis aussi de simplifier de nombreuses manipulations et de structurer le code créé.

I.B) Le cahier des charges et les objectifs fixés

Nous exposerons dans cette partie quel a été le cahier des charges établi avant la partie de développement de l'application avec le framework Symfony.

a. Le cahier des charges

L'application dispose de diverses fonctionnalités, les principales fonctions sont l'envoi de mail de relance et la visualisation des différents éléments contenus dans la base de données.

L'ajout d'éléments à la base de données

Tout d'abord, nous pouvons créer un vacataire et lui attribuer un nom, un prénom, une adresse mail ainsi qu'une civilité. Nous pouvons ajouter des formations et des pièces utiles pour compléter un dossier de vacataire.

La création d'un dossier

L'utilisateur peut aussi créer un dossier qui correspond à l'association entre un vacataire et une formation spécifique, il pourra alors choisir les pièces que le vacataire doit encore rendre. Les dossiers peuvent être modifiés quand un vacataire a finalement rendu les pièces qui lui manquaient.

Le fonctionnement des mails et des modèles de mails

Des modèles de mails peuvent être créés et être complétés lors de l'envoi d'un mail, un historique des mails envoyés est aussi disponible.

La visualisation de la base de données

L'ensemble des vacataires, des pièces, des dossiers, des modèles de mails et des formations est visible sous forme de tableaux.

L'authentification de l'utilisateur

L'utilisateur devait aussi avoir la possibilité de se connecter à l'application avec un identifiant et un mot de passe, mais cette fonctionnalité a été mise de côté car l'intégration à l'intranet effectuait déjà cette tâche.

b. Objectifs

Les principaux objectifs du projet étaient de découvrir comment utiliser un framework tel que Symfony à travers un projet ainsi que de réussir à comprendre les besoins d'un client et de réussir à concevoir une application capable de répondre aux attentes de notre cliente Mme Marion Berthoz.

I.C) Propositions et modifications du cahier des charges

Dans cette partie nous aborderons les différentes modifications ayant été apportées au cahier des charges ainsi que les différentes propositions faites.

a. Modifications

Les principales modifications du cahier des charges ont été d'empêcher la suppression de certains éléments de la base de donnée relatifs aux dossiers déjà créés. En effet, si une formation ou un vacataire venait à être supprimer alors des dossiers risqueraient de se retrouver liés à des informations inexistantes.

b. Propositions

Nous avons fait plusieurs propositions à notre cliente afin de répondre à sa demande qui était de pouvoir automatiser les relances des mails.

Dans la première proposition, les fichiers formant la base de données seraient encore remplis par Mme Marion Berthoz et automatiquement chargés dans l'application. L'application n'aurait plus eue qu'à récupérer les données contenues dans les fichiers et s'en servir pour envoyer les mails.

Cette proposition ne pouvait pas fonctionner car les fichiers excel n'étaient pas uniformes dans leur contenu.

La deuxième proposition faite en alternative avec la première, était de ne récupérer des anciens fichiers excel uniquement les données utilisables et récupérables afin de les intégrer à une nouvelle base de données gérée par l'application elle même. La base de données liées aux relances de vacataires et sous forme de fichiers excel était alors abandonnée. C'est cette solution qui a été finalement retenue.

II) Elaboration et mise en œuvre du projet

II.A) Les différentes phases

Nous avons structuré le travail à effectuer en trois grandes phases. La première phase consiste à l'analyse du projet, la conception ainsi qu'à l'étude de l'existant et enfin la réalisation du cahier des charges. La seconde, la plus importante, se caractérise par toute la partie développement de l'application web, c'est-à-dire le codage à proprement parlé. Cette partie représente les deux tiers du temps consacrés au projet. Enfin la troisième phase, consiste à la rédaction du rapport de projet et aux modifications et aux corrections de dernières minutes.

La première et la troisième phase équivalent au tiers restant du temps imparti sur le projet.

a. Phase de conception

La phase de conception est déterminante pour la suite de notre projet. Dès le début de cette phase nous avons pris de nombreux rendez-vous avec notre cliente Mme Berthoz et notre tuteur M. Genthial afin de prendre connaissance de l'existant et de mettre en commun nos idées sur l'élaboration de l'application web. Une fois l'étude de l'existant effectuée, nous nous sommes penchés sur la conception du projet en réalisant plusieurs diagrammes (classe, cas d'utilisation, séquence) et en élaborant le cahier des charges du projet. Dans ce dernier, nous avons défini le projet et les objectifs, puis, nous avons déterminé nos futurs choix de conceptions (les différentes fonctions principales de l'application), ainsi que leur description.

Nous avons effectué par la suite, le choix des différents logiciels pour le codage de l'application. Nos choix se sont portés sur le framework* PHP Symfony2* ainsi que sur Bootstrap3* pour le thème de notre application web.

b. Phase de développement

La phase de développement est la phase la plus longue de notre projet. Nous avons, chacun de notre côté utilisé un serveur local Apache ainsi qu'une base de données locale avec PhpMyAdmin* pour faciliter le développement en groupe. A l'aide du diagramme de Gantt, réalisé lors du cahier des charges dans la première partie, nous avons choisi de diviser le codage en plusieurs parties. Dans un premier temps, nous avons installé Symfony et mis en place les modules nécessaires au projet. Ensuite, nous avons codé toute la partie liée aux vacataires (c'est-à-dire l'ajout et la modification de vacataire) et toute la partie liée aux dossiers et à leur affichage (c'est-à-dire la création et la modification des dossiers). Dans un second temps, nous avons développé la partie sur les formations et les pièces manquantes, c'est-à-dire toutes les informations à utiliser lors de la création d'un dossier. Enfin, dans une troisième et dernière partie, nous avons programmé les mails et les relances de mails afin que notre client puisse relancer les dossiers incomplets ou prévenir les vacataires qu'ils leur manquent des pièces.

c. Phase de rédaction et modification

Cette ultime phase est aussi très importante, elle consiste à réaliser le rapport de projet de notre application web. Elle permet aussi de faire le point sur tout ce qui a déjà été fait au préalable. C'est aussi dans cette phase, peu avant la rédaction du rapport que nous avons effectué les ultimes retouches et modifications sur notre projet. Ces modifications ont aussi bien été de l'ordre visuel avec des changements de titrage ou réordonnancement de certains onglets mais aussi d'ordre technique avec plusieurs changements du code des fonctions principales à la demande ou non du client. Une fois ces modifications effectuées, nous avons par la suite réalisé le rapport de projet.

II.B) Les choix techniques

a. Un choix d'intégration à l'intranet

Les contraintes qui ont été définies dans notre cahier des charges nous ont amenés à un choix de technologies assez restreint. En effet, notre choix s'est porté sur une application intégrable à l'intranet de l'IUT afin de faciliter l'accès à l'application. En effet, procurer une application accessible de partout depuis un navigateur internet au sein de l'IUT peut s'avérer très utile pour une évolution possible de l'application. Cela peut permettre un accès multi utilisateur par exemple. De plus intégrer l'application sur les serveurs de l'IUT permet d'effectuer des sauvegardes régulières de la base de données ce qui est un atout important pour une application qui interagit avec de nombreuses informations. Enfin, cela permet d'utiliser une seule phase d'authentification, celle de l'intranet, pour accéder à l'interface de gestion des dossiers et permet donc de ne pas multiplier les identifiants et mots de passes pour un utilisateur.

b. Les technologies qui en découlent

La contrainte d'intégration à l'intranet nous a dirigés vers l'utilisation des technologies du Web et principalement l'utilisation des langages HTML5, CSS3, JavaScript et PHP. Pour ce qui est de l'interface graphique, nous avons utilisé une bibliothèque nommée Bootstrap3 couplée à un thème fourni par AlmsaeedStudio et qui a permis d'avoir une interface claire et ergonomique sans peu de configurations. De plus ce thème est essentiellement basé sur le JavaScript et plus précisément sur la bibliothèque jQuery. Celle-ci est utilisée pour gérer les éléments dynamiques de l'application, comme par exemple les menus.

En ce qui concerne le langage PHP, nous nous sommes servis d'un framework, Symfony2, qui est actuellement utilisé par l'intranet. Ce

framework bien que complexe à prendre en main s'est avéré extrêmement puissant et nous a permis d'accélérer et de fiabiliser les phases de codage grâce aux différents modules présents par défaut et ceux ajoutés manuellement en fonction des besoins.

c. Les modules de Symfony2

Symfony2 comporte de nombreux modules permettant de faciliter la vie des développeurs et apporter un ensemble de fonctionnalités qui ne sont pas présentes par défaut dans le framework. Il permet aussi de simplifier certaines actions en proposant du code plus évolué que ce que l'on aurait pu faire.

Le module Doctrine

Parmi ces modules, on peut citer Doctrine2, qui permet de manipuler une base de données relationnelle avec une notion d'objets qui est propre aux langages orientés objets. Celui-ci nous a permis d'interagir avec notre base de données MySQL de façon très simple, de faciliter la rédaction de requêtes SQL et de réaliser des jointures entre les tables par le biais d'associations propres à Doctrine. Le code ci-dessous permet de se rendre compte de la puissance de Doctrine.

```
public function findClosedDossier(){  
    return $this->createQueryBuilder("d")  
        ->leftJoin("d.etat", "e")  
        ->where("e.libelle = 'Complet'")  
        ->addSelect("d")  
        ->getQuery()->getResult();  
}
```

Figure 1:Exemple de requête avec Doctrine

Comme on peut le constater, l'utilisation de jointure n'aura jamais été aussi facile. L'appel à la méthode `leftJoin()` avec les paramètres adéquats permet de réaliser une requête fonctionnelle. Dans le cas de cet exemple, la requête va simplement récupérer les dossiers complétés.

Le moteur Twig

En plus de Doctrine2 qui nous a été fortement utile, nous avons utilisé le moteur de template Twig, permettant grâce à une certaine structure de code, d'interagir avec un code PHP dans une page HTML très facilement et ce d'une façon plus élégante qu'en PHP pur. Vous trouverez ci-dessous un exemple de code exploitant la puissance de ce moteur de template.

```
{% for mail in dossier.mails %}
    <tr id="mail-{{ mail.id }}">
        <td class="mail-id" style="display: none">{{ mail.id }}</td>
        <td>
            <a href="#" class="rowlink"
              data-toggle="modal"
              data-target=".modal">
                {{ mail.date | date("d/m/y") }}
            </a>
        </td>
        <td>{{ mail.titre }}</td>
    </tr>
{% endfor %}
```

Figure 2: Exemple d'utilisation de Twig

Comme on peut le constater, la structure se rapproche de celle du PHP, sans l'inconvénient d'alourdir le code.

En plus de cette fonctionnalité, Twig nous a permis d'organiser le code de nos pages d'une façon optimisée, en séparant les pages en modules. En effet, ces modules peuvent être réutilisés ou intégrés dans d'autres pages pour diverses raisons. Cela peut être très utile si par exemple nous avons une structure de page globalement identique de page en page, mais dont seulement certains éléments doivent changer.

Le module Doctrine Fixtures

Lors de tests, il est souvent pratique de repartir d'une base de données vierge. Les fixtures permettent, à la création de la base de données, d'ajouter des données préalablement définies. Cela s'avère très utile dans la phase de développement pour créer des jeux d'essais que l'on peut générer, ou régénérer pour effectuer des tests. De plus les fixtures peuvent être utilisées pour partager des informations entre les différents développeurs afin de travailler sur les mêmes données.

Le module JsRouting

Par défaut, il n'est pas possible d'utiliser les routes* de Symfony dans un fichier Javascript. Pour remédier à ce problème, on peut ajouter un nouveau module, FOSJsRoutingBundle, qui offre la possibilité de les utiliser. L'exemple ci-dessous montre un exemple d'utilisation.

```
$.ajax({
  url: Routing.generate('mail-relevance_afficher', { id: mailId }),
  data: {id: mailId},
  beforeSend: function(){
    $("#loading").removeClass('hide');
  },
  success: function(data){
    $('.modal-title time').html(data.date);
    $(".modal-body p").html(data.message);
  },
  complete: function(){
    $("#loading").addClass('hide');
  }
});
```

Figure 3:Exemple d'utilisation du module FOSJsRoutingBundle

```
mail-relevance_afficher:
  path: /dossiers/relevance/{id}
  defaults: { _controller: IutDossiersBundle:Mail:afficherRelevance }
  requirements:
    id: \d+
  options:
    expose: true
```

Figure 4 : Exemple de route utilisable avec le module FOSJsRoutingBundle

Le code ci-dessus permet au travers d'une requête AJAX le chargement de courriels présents dans la base de données. On peut noter l'utilisation d'une méthode (`Routing.generate()`) permettant de créer une URL d'accès à partir d'une route. Cette méthode va donc générer le chemin correspondant à la route `mail-re lance_afficher` en lui transmettant un identifiant comme paramètre par le biais d'une méthode `GET*`.

Afin que la génération d'URL fonctionne, il est nécessaire d'indiquer à notre route qu'elle peut être utilisée par le module. L'option « expose » présente à la fin de la route est utilisée dans ce but.

II.C) Répartition du travail au sein du groupe

Pour chacune des phases, le travail a été réparti entre nous après concertation. Chaque décision importante a été réfléchie et a été prise par tout le groupe de projet.

Pour la première phase, chacun a choisi une partie de la conception qu'il souhaitait effectuer. Nous nous sommes donc répartis les différents diagrammes UML* à réaliser dès le départ. Par la suite, tout le monde a travaillé sur le cahier des charges. Toutes les fonctions principales de notre application web et chacune des parties du cahier des charges ont été divisées en quatre. Tous les membres du groupe ont aussi été présents pour chaque rendez-vous importants avec notre client ou avec notre tuteur.

Un brainstorming pré-développement a ensuite eu lieu avec les quatre membres du groupe pour essayer de se répartir équitablement le code de notre application web. Nous nous sommes donc répartis une fois de plus les tâches clefs et fonctions principales mentionnées dans notre cahier des charges à réaliser. L'implémentation de notre framework `Symfony2` sur notre `GitHub*` (branche principale) a été réalisée par Thibault. S'étant documenté plus tôt, il a pu nous expliquer le fonctionnement de beaucoup de modules de `Symfony`. Une fois la prise en

main effectuée par l'ensemble des membres du groupe, chacun a pu effectuer le travail qui lui avait été confié. Théo a travaillé sur les pièces, les formations et certaines fonctionnalités de l'accueil. Mathis a travaillé sur les vacataires, les dossiers ainsi que certaines parties des pièces. Thibault a réalisé la partie sur les mails, l'accueil, les dossiers et a mis en place le thème avec bootstrap3. Mathie a travaillé sur la partie vacataire et formation ainsi que sur l'accueil. Bien évidemment, si un membre rencontre un problème au niveau de son code, il peut le mettre en commun pour que chacun puisse l'aider. L'entraide fut le maître mot de tout notre projet. En ultime recours pour certains problèmes liés à la programmation, nous avons été obligés de demander à notre tuteur certaines explications.

Enfin, pour la dernière phase de notre projet, le processus reste le même que pour les deux phases précédentes. Toutes les parties du rapport de projet ont été réparties entre les membres du groupe. Toutefois, chacun a pu revoir ou modifier certains morceaux du rapport, après l'avoir informé auprès du groupe.

Au final, chaque membre du groupe a pu apporter sa pierre à l'édifice dans toutes les phases du projet. Chaque répartition du travail a fait l'objet d'une réunion des membres du groupe afin que chacun puisse donner son avis sur le travail à fournir.

II.D) Problèmes et solutions envisagées

a. Difficultés d'installation de Symfony2

Bien que l'installation du framework Symfony2 ait été assez simple et rapide pour un seul développeur, les choses ont été un peu plus compliquées dès le passage du projet sur la plateforme collaborative GitHub. En effet, pour des raisons de sécurités et de performances, certains fichiers et dossiers n'ont pas été téléversés sur GitHub. En effet, celui-ci

possède plusieurs fichiers de configurations extrêmement importants avec des informations qui ne doivent absolument pas être divulguées à des tiers. On parle notamment du fichier `parameters.yml` qui contient entre autres les identifiants d'accès à la base de données, un jeton permettant de chiffrer des informations liées à l'application, notamment les cookies*, et pour finir les identifiants d'accès au serveur d'envoi de courriels. Le problème qui s'est donc posé était d'avoir un fichier relativement identique afin de pouvoir utiliser nos divers environnements de travail de la même façon.

De plus, lors du téléversement, l'intégralité du dossier `vendors` de l'application n'a pas été transférée. Ce dossier renferme le cœur du framework, et par conséquent un nombre de fichiers assez important. Pour « corriger » ce problème, nous avons dû utiliser `Composer*`, un gestionnaire de dépendances, afin de télécharger le dossier manquant et par l'occasion mettre à jour les différents modules.

b. Le développement collaboratif

Comme énoncé précédemment, nous avons utilisé Git au travers de GitHub pour partager notre projet, afin de faciliter la distribution du code entre les développeurs, et pouvoir versionner nos fichiers. Cela a dans l'ensemble plutôt bien fonctionné, mais nous avons quand même rencontré un problème. En effet, travailler à plusieurs sur un même code peut engendrer des conflits si l'on a par exemple édité les mêmes lignes d'un même fichier. Pour résoudre en partie ce problème, nous avons décidé de développer chacun sur une branche différente et surtout de rester dans le cadre de ce que l'on avait à développer. C'est-à-dire de ne pas empiéter sur le travail d'un autre pour par exemple l'aider dans l'immédiat, mais attendre, ou reprendre sa branche afin de corriger les problèmes.

III) Résultats et perspectives

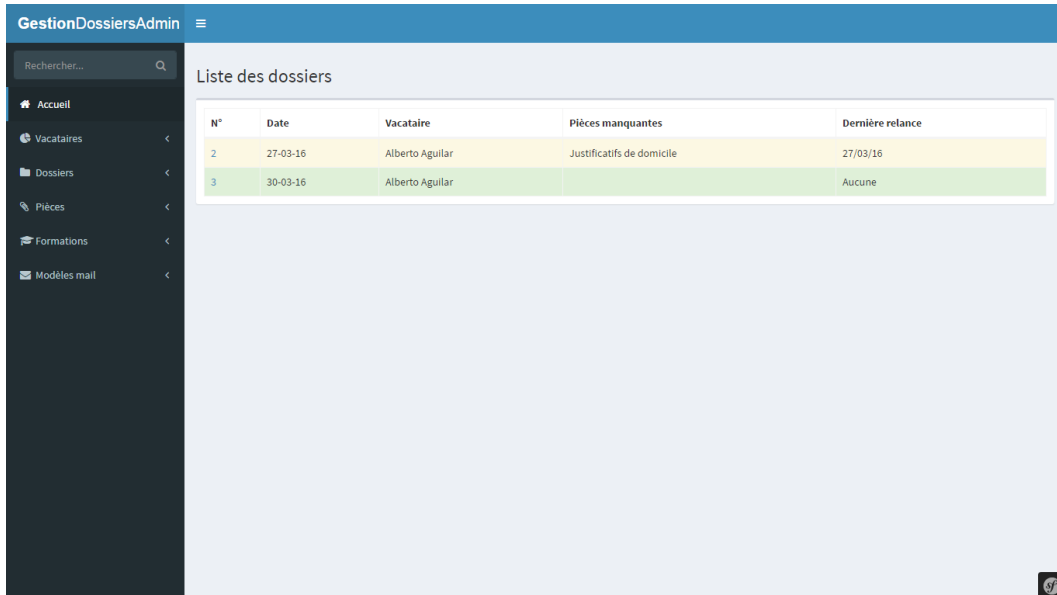
III.A) Etat final et problèmes non résolus

Actuellement, notre application Web est entièrement fonctionnelle.

Nous avons réussi à respecter les exigences primaires du client. En effet, notre application peut envoyer des mails de relance à partir d'un dossier qui a été créé auparavant. Lorsque l'on souhaite envoyer un mail, l'application propose de sélectionner soit un modèle de mail, soit un mail vierge. Ce premier permet de générer directement un mail pré-formaté avec les pièces manquantes, le nom, et la civilité du vacataire, pouvant être modifiés avant l'envoi, ce qui fait gagner du temps lors de l'envoi. Ceci est possible grâce aux balises, de type « {{ vacataire }} » par exemple, qui permettent de placer les données correspondantes à leurs places lorsqu'on appelle l'affichage de ce modèle de mail. Le second permet tout simplement de générer un modèle de mail vide. Chaque modèle de mails est enregistré dans une base de données et a un identifiant associé. Par défaut, chaque donnée qui sera rajoutée à la base, dans n'importe quelle table, se verra attribuer un identifiant, ce qui nous permet plus facilement de la récupérer. Une page HTML est dédiée pour l'ajout de modèle de mails, où on choisit un titre et on rédige un texte balisé. Ainsi, le modèle sera sélectionnable lorsque l'on souhaite faire une relance. On pourra également modifier ou supprimer un modèle en y accédant via la liste des modèles de mails disponible en cliquant sur le sous onglet correspondant. Pour continuer, lorsqu'on crée un dossier, il faut sélectionner le nom du vacataire, l'état du dossier, les pièces manquantes (on peut en sélectionner plusieurs), et la formation. Dans la base de données, ce sera les identifiants correspondant à ces éléments qui seront inscrits ainsi que la date et l'heure de la création. L'enregistrement de la formation se fait dans une autre table spécialement dédiée, car un vacataire peut avoir plusieurs formations. Lorsqu'on y

accède via la page HTML correspondante, on peut voir le tableau de l'historique des relances, relancer ce dossier et voir les informations associées. En sélectionnant une ligne de l'historique des relances, on peut voir le mail envoyé associé à cette ligne. Un dossier ne peut voir que ces pièces manquantes et son état être modifiés. Par contre, on ne peut pas supprimer un dossier. Lorsqu'on décide de faire une relance, la page correspondante s'ouvre alors. Lors du clic sur le bouton « Envoyer », l'email sera envoyé à l'adresse mail du dossier à relancer avec l'adresse du client comme expéditeur. De plus, l'email, la date et l'heure d'envoi sont enregistrés et rajoutés au tableau historique des relances correspondant au dossier relancé. Aussi, une page « Historique des dossiers » permet de voir tous les dossiers qui ne sont plus à traiter. Dès qu'un dossier sera complet, il apparaîtra sur cette page. On a intégré une pagination ne faisant apparaître que 10 dossiers par page pour un souci d'ergonomie et de performance. Pour chaque liste d'objets, on met en place une pagination qui permet de respecter ces soucis derniers. Par défaut, on a mis en formation GEA, TC, Info, RT et « aucune formation », mais on peut en rajouter via l'application. La suppression ici n'est également pas possible. La liste des formations est visible grâce à l'onglet « Liste formations ». De plus, pour créer un vacataire sur la page correspondante, il faut renseigner son nom, son prénom, son email, et sa civilité. Une fois ces renseignements fournis et la validation effectuée, le vacataire est rajouté à la base de données avec un identifiant attribué qui est généré à ce moment-là. La liste des vacataires est accessible depuis l'application. Enfin, chaque pièce peut être modifiée ou supprimée. On peut aussi en rajouter grâce à la page HTML qui lui est associée. La liste des pièces est également accessible en cliquant sur l'onglet correspondant.

Nous avons choisi d'utiliser le thème AdminLTE et d'utiliser Bootstrap pour rendre notre interface le plus ergonomique possible. Ainsi, lorsqu'on lance l'application, on arrive sur cette page d'accueil (figure 5):



| N° | Date | Vacataire | Pièces manquantes | Dernière relance |
|----|----------|-----------------|---------------------------|------------------|
| 2 | 27-03-16 | Alberto Aguilar | Justificatifs de domicile | 27/03/16 |
| 3 | 30-03-16 | Alberto Aguilar | | Aucune |

Figure 5: Page d'accueil de l'application de gestion de dossiers administratifs

Sur la page d'accueil se trouvent tous les dossiers qui sont traités, complets ou incomplets. On a mis en place une pagination, pour les mêmes raisons que précédemment, de 10 dossiers. A chaque fois que l'on aura terminé une action, de type relance d'un dossier, création d'un vacataire, etc., on tombera sur cette page. De plus, pour montrer qu'une action s'est bien ou mal passée, on affichera un message flash qui disparaîtra lors de l'actualisation de la page. L'arborescence du menu est choisie de manière à faciliter la navigation et la compréhension du fonctionnement de l'application pour le client. On peut choisir de l'ouvrir ou de le fermer. L'accès aux pages est toujours possible lorsqu'on clique sur l'onglet correspondant qui reste toujours visible mais moins imposant que lorsque le menu est ouvert.

Grâce aux nombreuses fonctionnalités intégrées, notre application Web est donc fonctionnelle et ergonomique.

Parmi tous nos problèmes, certains n'ont pas pu être résolus.

Pour les exigences primaires, nous n'avons pas de problèmes non résolus. Par contre, pour celles qui sont secondaires, il y a une fonctionnalité qui nous a posé problème, c'est la recherche de vacataires. Tout d'abord, notre souci majeur est le temps. Il nous en aurait fallu plus, cela nous aurait peut-être permis de trouver une solution ou un script qui soit fonctionnel. Ensuite, nous avons rencontrés un ensemble de petits problèmes. En effet, lorsqu'on a voulu utiliser une requête AJAX pour récupérer les vacataires, on a eu un souci avec l'implémentation d'url de Symfony2. Cette implémentation ne marche pas dans les requêtes AJAX. Pour le résoudre, nous avons dû installer le bundle FOSJsRoutingBundle et mettre le code correspondant. Ainsi, nous avons pu accéder à l'url. Ensuite, nous avons eu un souci avec le script PHP permettant d'exécuter la requête AJAX. Il a fallu l'adapter avec Symfony2. Pour continuer, nous n'avons pas réussi à afficher une liste déroulante lorsque le client tape dans la barre de recherche. En effet, nous aurions aimé que lorsque le client écrit un caractère dans la barre de recherche, il y ait une actualisation de la recherche et qu'une liste déroulante s'affiche en dessous de la barre de recherche, avec tous les vacataires possédant la chaîne de caractères écrite dans leur nom. Hélas, nous n'avons pas réussi à trouver le code HTML correspondant qui permet d'exécuter cela. De plus, nous ne sommes pas parvenus à réparer les problèmes d'actualisation, qui intervenaient à certains moments et seulement pour cette partie du code.

On a donc eu plusieurs problèmes, nombreux ont été résolus, mais d'autres nous ont posé plus de soucis.

III.B) Développement futur

A ce stade, notre application est entièrement fonctionnelle, ergonomique et fluide, mais il reste quelques améliorations à apporter.

Tout d'abord, ces améliorations sont peu nombreuses. En effet, nous avons fait en sorte de maximiser l'efficacité de notre application Web pour qu'elle soit le plus efficace possible. Mais on peut encore optimiser notre code. Effectivement, en essayant de modifier certaines lignes de notre code, on pourrait réduire le temps de chargement de chaque page. Ce gain serait minime, de l'ordre de quelques millisecondes, mais il améliorerait encore les performances de l'interface. De plus, il y aura sans doute des améliorations à faire dans le futur, que nous n'avons pas vues. Enfin, nous avons effectué des tests d'utilisation sur notre application, dont les résultats sont satisfaisants. Toutefois, elle pourra réagir différemment lorsqu'il y aura plus de données enregistrées dans la base de données et il est possible que des améliorations soient nécessaires. Les langages informatiques évoluent sans cesse, et en révisant le code, on pourra sans doute l'améliorer.

Actuellement, très peu d'améliorations sont à prévoir, car l'application est déjà très performante et offre de nombreuses possibilités.

Malgré tout, on peut envisager des développements futurs qui seraient utiles.

En priorité, il faut résoudre le problème lié à la barre de recherches. En réussissant à la coder, cela apporterait une nouvelle fonctionnalité non négligeable. En effet, elle permet de trouver un vacataire très rapidement d'une part, sans avoir à aller sur la page des vacataires, et de chercher le vacataire en question. C'est un gain de temps qui peut être précieux lorsque l'on a beaucoup de relance à faire pendant une journée par exemple. De plus, elle affiche une liste de vacataires à partir d'une chaîne de caractères. Ensuite, on peut intégrer une relance automatique dont la fréquence serait déterminée par le client lors de la première relance. Par exemple, le client choisit que tous les 30 jours, le vacataire reçoit une relance tant qu'il n'a pas envoyé tous les papiers demandés. Alors, tous les 30 jours, un mail sera

généré automatiquement, le nom, la civilité et les papiers manquants (qui peuvent être différents de ceux de la première relance) seront récupérés et l'envoi de rappel sera effectué. De plus, cette option pourra être décochée si le client ne veut pas effectuer de relance automatique pour un vacataire.

Enfin, on peut développer des fonctionnalités supplémentaires permettant ainsi une gestion administrative plus élaborée depuis notre application. Par exemple, l'application aurait une partie « Mail » qui analyserait automatiquement les mails et déterminerait si le nom de l'expéditeur correspond à un vacataire, on aurait alors plusieurs options permettant de changer l'état d'un dossier si le mail reçu contient toutes les pièces manquantes.

L'interface Web dispose donc de nombreuses possibilités de développements futurs.

Conclusion

Notre objectif fut de réaliser une application web de gestion de dossiers administratifs pour notre client Mme Berthoz. Cette application devait être implantée sur l'intranet afin de faciliter son utilisation future. Nous devions réaliser un système de création de dossiers pour les nouveaux vacataires ainsi qu'un système de relance de mails pour pouvoir relancer ces derniers. Nous devions aussi ajouter à cette application chaque formation de l'IUT et chaque pièce demandée à la création d'un dossier administratif, ainsi que plusieurs modèles de mail de relance. Nous avons réussi à finir le projet dans les délais accordés. Notre application web fonctionne parfaitement, il n'y a aucun problème à signaler au niveau des fonctions principales.

Ce projet nous a permis de développer notre esprit d'équipe et d'enrichir nos connaissances avec de nouvelles technologies comme le framework Symfony2 très utilisé en milieu professionnel et qui peut être un atout pour notre travail futur au sein d'une entreprise. Ce projet nous a aussi permis de perfectionner notre autonomie face à la prise en main de nouveaux outils et face aux diverses difficultés rencontrées durant le projet. Enfin, nous avons pu constater le déroulement complet d'un projet informatique, de la conception jusqu'à la réalisation, avec ses contraintes et délais, ce qui est aussi un plus pour notre avenir dans le monde de l'entreprise.

Glossaire

Bootstrap : Bootstrap est un framework CSS/JS

Composer : Composer est un gestionnaire de dépendances open source écrit en PHP. Il permet à ses utilisateurs de déclarer et d'installer les bibliothèques dont le projet principal a besoin.

Cookie : Le cookie est l'équivalent d'un petit fichier texte stocké sur le terminal de l'internaute. Existants depuis plus de 20 ans, ils permettent aux développeurs de sites internet de conserver des données utilisateur afin de faciliter leur navigation et de permettre certaines fonctionnalités.

Framework : Un framework ou structure logicielle est un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel

GET : la méthode GET est un moyen de passer des paramètres d'une requête HTTP depuis le navigateur au serveur. Cette méthode place les paramètres, généralement séparés par un caractère spécial tel que l'esperluette (&), dans l'URL même, qui est visible pour la personne qui utilise le navigateur.

GitHub : GitHub est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git.

PHPMyAdmin : phpMyAdmin (PMA) est une application Web de gestion pour les systèmes de gestion de base de données MySQL

Route : une route permet, à partir d'une URL, de déterminer quel contrôleur appeler et avec quels arguments. Cela permet de configurer son application pour avoir des URL propres et explicites, ce qui est important pour le référencement et pour le confort des visiteurs.

Symfony : Symfony est un framework MVC libre écrit en PHP 5

UML : UML est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée pour visualiser la conception d'un système. Il est couramment utilisé en développement logiciel et en conception orientée objet.

Table des illustrations

| | |
|---|----|
| Figure 1:Exemple de requête avec Doctrine..... | 14 |
| Figure 2: Exemple d'utilisation de Twig..... | 15 |
| Figure 3:Exemple d'utilisation du module FOSJsRoutingBundle | 16 |
| Figure 4 : Exemple de route utilisable avec le module FOSJsRoutingBundle | 16 |
| Figure 1:Page d'accueil de l'application de gestion de dossiers administratifs..... | 22 |