Architecture technique

Le projet est composé de deux applications : e-doc pour la gestion des fichiers et une application pour la gestion des utilisateurs. Pour l’application de gestion des utilisateurs, nous avons décidé d’utiliser Apache Directory Studio qui est une plateforme complète de répertoire d’outillage destiné à être utilisé avec n’importe quel serveur LDAP. LDAP (*Lightweight Directory Access Protocol*) est un [protocole](http://www.commentcamarche.net/contents/531-protocoles) standard permettant de gérer des annuaires, c'est-à-dire d'accéder à des bases d'informations sur les utilisateurs d'un réseau par l'intermédiaire de protocoles TCP/IP. Généralement, les bases d’informations concernent des utilisateurs mais ces bases peuvent être utilisées pour la gestion des stocks de matériel dans une entreprise. Dans notre cas, les informations concerneront les utilisateurs de chaque service qui utiliseront l’application e-doc.

LDAP présente les informations sous forme d'une arborescence d'informations hiérarchique appelée **DIT** (*Directory Information Tree*), dans laquelle les informations, appelées **entrées** (ou encore *DSE, Directory Service Entry*), sont représentées sous forme de branches. Une branche située à la racine d'une ramification est appelée racine ou suffixe (en anglais *root entry*).   
  
Chaque entrée de l'annuaire LDAP correspond à un objet abstrait ou réel (par exemple une personne, un objet matériel, des paramètres, ...).   
  
Chaque entrée est constituée d'un ensemble de paires clés/valeurs appelées **attributs**. 

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| (**DC**) | projetsi | | | | | | |
|  |  |  | | | | |  |
| (**OU**) | Services | | | | users | | |
|  |  | |  |  |  |  |  |
| (**CN**) | L3App | | M1App | M2App |  |  |  |
| (**CN**) | Responsable | | Responsable | Responsable |  |  |  |
|  |  | |  |  |  |  |  |
| (**UID**) |  | |  |  | EmJavay | NoElMoja | SiAydin |

Dans LDAP, chaque utilisateur est composé de deux types d’attributs :

* Les attributs normaux : nom, prénom, l’adresse mail, adresse postale, …
* Les attributs opérationnels : Ils sont reconnus et peuvent être accédé que par le serveur LDAP afin de manipuler des données de l’annuaire (**CN**, **UID, …**).

Analysons en détails les attributs opérationnels utilisés dans l’application :

* **DC** (Domain Component) : Nom distinct permettant d’identifier de manière unique un élément de l’arborescence
* **OU** (Organizational Unit) : Permet d’organiser les services et les utilisateurs de chaque service
* **CN** (Common Name) : Nom attribué à chaque objet (Exemple : CN L3App ou CN El Mojahid Nordine)
* **SN** (SurName) : Surnom d’un utilisateur, renseignant utilisé pour effectuer des recherches plus rapide (Exemple : Maxime Decalf => SN madecalf)
* **UID** (User ID) : Il s’agit de l’identifiant unique d’un utilisateur, il peut être comparé à une clé primaire dans un table d’une base de donnée SQL

Une librairie LDAP en PHP a été intégré dans le projet afin d’accéder aux attributs normaux et opérationnels des utilisateurs.

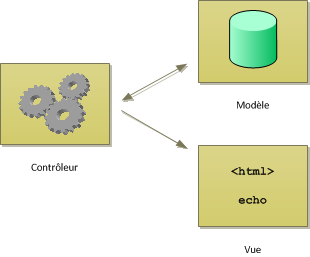
Après avoir configuré l’application de gestion des utilisateurs, nous avons mis en place une base de données ORACLE afin de gérer l’application e-doc. L’avantage d’Apache Directory Studio et d’ORACLE, c’est qu’ils peuvent être installés facilement sur l’ensemble des postes de mes collaborateurs. Un fichier SQL a été édité pour commencer une programmation répartie sur la même base. Le fichier SQL a permis à chacun tout au long du projet de remettre sa base à « neuve » après chaque jeu de test qu’on pouvait effectuer.

N’étant pas toute la semaine ensemble, il était important d’utiliser un outil qui nous permettrait à l’intégrateur de récupéré distinctement les dernières versions de code de chacun. C’est pourquoi, nous avons travaillé sur Trello qui permet à chacun de visualiser l’avancement des tâches et de publier leur dernière modification de code avec date et heure pour faciliter la compréhension de l’intégrateur.

Parfois nous avions des fichiers trop volumineux, par conséquent il fallait trouver un autre outil pour transférer nos fichiers volumineux, c’est ainsi qu’on a tous créé un compte sur GitHub. Trello nous permet de mettre à jour les tâches et d’avertir lors d’un envoie volumineux de fichier. La liaison entre Trello et GitHub a permis d’être toujours à jour dans nos codes et de considérablement simplifié la lourde tâche d l’intégrateur.

Il est très facile de perturber le travail de chacun et de vite tomber dans une confusion générale. C’est pourquoi dans un souci de lisibilité et de coopération, nous avons adopté le modèle MVC (Model View Controller). MVC est un modèle destiné à répondre aux besoins des applications interactives en séparant les problématiques liées aux différents composants au sein de leur architecture respective :

* **Modèle** : Il gère les données du site. Son rôle est d'aller récupérer les informations « brutes » dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur. Nous avons un modèle navigation qui contient une fonction de suppression de fichier ou encore qui permet de récupérer les droits d’un utilisateur. Il se peut que l’information qu’on cherche ne se trouve pas dans la base de données donc le rôle du modèle sera d’ouvrir, chercher et fermer le fichier mais dans notre cas, nous avons développé une base de données qui contient toutes les informations nécessaire aux recherches.
* **Vue** : Elle se concentre sur l'affichage. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve essentiellement du code HTML mais aussi quelques boucles et conditions PHP très simples, pour afficher par exemple la liste des fichiers auxquels un utilisateur possède un droit (Download ou Upload).
* **Contrôleur** : Il gère la logique du code qui prend des décisions. C'est en quelque sorte l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue. Le contrôleur contient exclusivement du PHP. C'est notamment lui qui détermine si le visiteur a le droit de voir la page ou non (gestion des droits d'accès).



MVC a permis d’améliorer un peu plus l’intégration du code et il nous a permis de travailler chacun de notre côté sans dépendre de l’avancement de chacun.

Afin de faciliter la compréhension du code, il était essentiel d’instaurer une norme de codage. Dans notre cas, elle est très adaptative, on le sait, la programmation de chacun est unique. C’est pourquoi, la seule norme imposé à tous est le nom des éléments du projet, par exemple, une fonction dans une vue doit avoir un nom qui détermine clairement son utilisation comme la fonction **uploadFile()** ou le controller **connexion.php** ou encore la vue **navigation**.