Utilisation d'un framework de développement :A piece of Cake(Php) ?

Florent Charuel
DiSI -Pôle Urbanisation du SI
Domaine universitaire, 2061, Rue de la Piscine, 38401 SAINT MARTIN D'HERES CEDEX

Christian Lenne
DiSI -Pôle Urbanisation du SI
Domaine universitaire, 2061, Rue de la Piscine, 38401 SAINT MARTIN D'HERES CEDEX

Romain Sury
DiSI -Pôle Urbanisation du SI
Domaine universitaire, 2061, Rue de la Piscine, 38401 SAINT MARTIN D'HERES CEDEX

Résumé

L'université Joseph Fourier est amenée à développer des services en complément des produits et applications qui s'appuient majoritairement sur les logiciels de l'AMUE. Ces développements permettent de fluidifier ou dématérialiser l'ensemble des processus constituant les tâches les plus courantes de l'établissement.

Pour ces développements, il est nécessaire de faire des choix techniques qui doivent intégrer différents paramètres structurels.

Au niveau du langage de programmation, les langages sont divers (JAVA, PHP, C#, …). Du coté des interfaces Homme-Machine, les choses sont suffisamment mûres et le modèle MVC (Model-View-Controller) est désormais incontournable.

La prise en compte de tous les paramètres (structure de l'équipe, typologie des applications, …), nous a amené à nous interroger en premier lieu sur le choix du langage (JAVA vs PHP). Nous avons choisi de partir sur PHP 5. S'est alors posé le choix de la méthode de développement pour répondre aux différentes demandes nécessitant de la réactivité. Le choix d'un environnement de développement a été fait et nous en avons envisagé plusieurs pour choisir CakePHP (1).

De l'expérience acquise par les développements effectués, nous présentons les points forts de l'utilisation du framework CakePHP ainsi que ses faiblesses. Nous faisons également une analyse du retour sur investissement lié à la prise en main d'outils de ce type. Enfin, la dernière perspective liée au choix par la DSI de Drupal 7 (2) comme CMS nous interpelle sur son utilisation pour le développement de plusieurs demandes. Au delà de ses capacités de CMS, la richesse de ses modules et son architecture le positionne comme un "framework" potentiel de développement.

Mots clefs

Développement, Framework, CakePhp, Modèle MVC, Applications périphériques

1 Contexte

L'université Joseph Fourier, comme bon nombre de ses équivalentes françaises, dispose d'une équipe de développeurs au sein de sa DSI. Bien entendu, les produits et applications déployés en son sein s'appuient tout naturellement sur les offres nationales (AMUE), mutualisées (inter Universitaires) ainsi qu'au maximum sur des développements du monde libre.

Ces logiciels intégrés ne couvrent pas tous les besoins fonctionnels et il est nécessaire de maintenir une offre interne pour notamment fluidifier ou dématérialiser l'ensemble des processus constituant les tâches les plus courantes de l'établissement.

L'organisation de l'université évolue, poussée en partie par la mise en place de la loi sur les Libertés et Responsabilités des Universités (loi LRU) et le passage aux Responsabilités et Compétences Élargies (RCE) entraine une augmentation des missions à prendre en charge. En parallèle, la perspective de l'application, pour notre ministère, de la Révision Générale des Politiques Publiques (RGPP) ne peut qu'au mieux voir notre nombre de postes de titulaires stagner et l'on constate, dès à présent, un recours de plus en plus fréquent à l'embauche de personnels contractuels.

On peut également constater que les demandes fonctionnelles sont peu finalisées, que les demandeurs sont également surchargés et ne peuvent pas définir un cahier des charges précis. Il faut les assister dans cette tâche, les aider à préciser leurs besoins, souvent flous. Autrement dit, l'établissement du cahier des charges peut être long et fait appel à une certaine connaissance du fonctionnement interne de l'université. Dans ces conditions, le recours à une prestation externe, basée sur un prestataire peu au fait de nos environnements, travaillant sur un cahier des charges peu précis est souvent inenvisageable.

Dans ce contexte, il est nécessaire de faire des choix pour le développement d'applications qui vont bien au delà de considérations purement techniques, et prennent en compte les différentes facettes de la gestion des ressources humaines.

2 Choix du langage et de son environnement de développement

Nous nous attachons ici à ne prendre en compte que les langages utilisés pour les développements Web. Les langages de développement sont divers, mais offrent généralement des concepts objets plus ou moins poussés. Pour le Web, dans un environnement non Microsoft, JAVA et PHP se démarquent d'autres langages. JAVA fait encore figure de langage noble par sa structuration en couche, mais surtout parce qu'il est fortement typé. A contrario, l'absence de typage fort dans PHP, donne de ce dernier, une figure plus populaire. De nombreux forums sur Internet traitent de la question technique « Vaut-il mieux prendre JAVA que PHP ? ». Nous n'allons pas rentrer dans ce débat qui, comme le montrent les forums, ne permet pas d'avoir un avis tranché sur la question, tout au moins pour le développement d'applications non orientées métier. On note cependant que, pour ces deux langages, différents environnements de développement existent. Se posent donc les questions : d'une part de choisir son langage, d'autre part de choisir l'environnement de développement qui va avec.

2.1 Choix du langage

Comme nous l'avons mentionné, nous nous plaçons dans un environnement non Microsoft au niveau des serveurs qui hébergeront les applications qui seront développées. Notre environnement d'exécution est assez classique dans notre milieu, à savoir une souche Unix (FreeBSD ou Linux), Apache et Tomcat. Côté SGBD, même si les applications nationales sont construites sur Oracle, les bases de données complémentaires sont supportées actuellement par MySQL.

De cet ensemble de faits, nous excluons d'emblée C#. Nous considérons dans un premier temps quatre candidats potentiels : Python, PHP, JAVA et Ruby. Dans cette « short list », nous éliminons Ruby et Python car ils sont peu répandus au regard des deux autres langages. De fait, nous n'avons aucun développeur dans notre environnement qui connaisse ces langages et de plus nous pressentons des difficultés pour recruter des candidats ayant ces compétences.

Il nous restait à privilégier un des deux langages : JAVA ou PHP. Pendant de nombreuses années, les développements locaux se faisaient en JAVA, mais nous rencontrions des difficultés pour recruter des ingénieurs rompus à ce langage qui acceptent les conditions salariales proposées par l'établissement. Nous avons donc opté pour PHP qui nous permet de recruter plus facilement des ingénieurs ou assistants ingénieurs contractuels.

2.2 Critères d'analyse des environnement de développement.

Les critères d'analyse des environnements de développement peuvent être définis, entre autres, à partir de la taille et de la complexité des projets ainsi qu'en fonction des compétences à l'intérieur de l'équipe. Le choix technologique n'est donc pas figé et varie selon le contexte. Ces dernières années, l'offre de solutions adressées aux développements web s'est largement étoffée et chacune présente des qualités et des défauts face aux diverses exigences. Nous décrivons ici les critères qui nous ont guidés dans la sélection d'un framework de développement.

(1) Cohérence et richesse des classes du cœur : afin de répondre à un large spectre de demandes on attend d'un framework qu'il permette de couvrir l'ensemble des problématiques récurrentes dans le développement web telles que les interactions avec la base de données, l'authentification, la gestion des interfaces graphiques.

(2) Facilité d'apprentissage : ce facteur est à évaluer selon les compétences des développeurs et la complexité des projets à mener. La prise en main d'un framework peut être plus ou moins longue avant d'atteindre le cap où le gain en temps de développement est atteint. Le temps d'apprentissage peut aller jusqu'à six mois, voire un an, suivant la complexité et la richesse du framework. La courbe d'apprentissage sera considérablement améliorée si les concepts de la programmation orientée objet sont maitrisés ainsi que les « designs patterns », abondamment utilisés au sein des frameworks. En ce qui nous concerne, nous recherchions un compromis entre facilité d'accès et puissance de l'outil. Nous avons également considéré que la phase d'apprentissage pouvait être raccourcie en nous appuyant sur une prestation de formation. Nous avons donc regardé si ce genre de formation était disponible facilement.

(3) Extensibilité : cette notion désigne la possibilité d'enrichir les classes du framework. Dans le cadre du Système d'Information universitaire, certains cas de figure sont récurrents dans les applicatifs. Par exemple, il est très fréquent que nos applications nécessitent une authentification via le SSO CAS ou que la vérification se situe au niveau de l'annuaire LDAP de l'université. Par conséquent, si le framework propose déjà une prise en charge de l'authentification basée uniquement sur un SGBD, il est appréciable de pouvoir étendre facilement la librairie de manière à ce que plusieurs modes d'authentification soient disponibles. Nous étions donc attentifs à la souplesse et à l'ouverture de code dans un souci d'intégrer nos propres composants logiciels.

(4) Une documentation suffisamment riche et bien organisée : une mauvaise documentation empêchera inévitablement d'utiliser toutes les possibilités de l'outil. Cela peut même conduire à mal utiliser le système, ce qui génèrerait par là-même une perte de temps pour rechercher des contournements éventuels. Une bonne documentation est donc essentielle.

(5) Stabilité : l'ensemble des fonctionnalités proposées par le framework doivent être fiables et testées. Si un bug contenu dans le framework est rencontré, il est toujours difficile d'apporter un correctif en interne à cause des nombreuses couches qui constituent le moteur de l'outil. Nous avons donc préféré nous orienter vers un framework ayant atteint une bonne maturité.

2.3 Choix effectué

Au moment de faire le choix d'un framework, Symfony (3) et Zend (4) se présentent alors comme les solutions les plus complètes, mais dont la prise en main semble assez lourde. A l'autre extrémité, on trouve des solutions très légères telles que codeIgniter (5) ou Yii (6) mais qui souffrent de certaines lacunes et ne permettent pas de couvrir tous les aspects cités précédemment. A mi-chemin, on trouve cakePHP qui propose un panel assez complet de ce que l'on peut attendre d'un framework moderne tout en restant accessible aux non initiés. De plus, il bénéficie d'une communauté active. Au regard de nos critères, Symfony et CakePHP se démarquaient, avec des évaluations différentes sur les différents critères. Si Symfony nous permettait d'envisager d'avoir recours à des prestations externes de formation, CakePHP quant à lui, impliquait de nous appuyer sur une communauté locale aux universités grenobloises. En fin de compte, pour répondre avec efficacité à des demandes d'applications de complexité simple à moyenne nous nous sommes donc orientés vers CakePHP privilégiant ainsi le partage d'expérience et le développement commun avec nos collègues d'établissements voisins.

3 Le choix à l'épreuve du temps

3.1 Utilisation de PHP

Cela fait maintenant plusieurs années que toutes les applications demandées sont développées en PHP. Les besoins émanant des personnels administratifs, des étudiants ou des enseignants-chercheurs ont de nombreux points communs : saisir de l'information, la traiter et la stocker, la restituer ; le tout depuis n'importe quel endroit. Les sites dynamiques en PHP répondent parfaitement à ces besoins. Le langage est facilement maitrisable et de très nombreuses bibliothèques, plug-ins et exemples sont disponibles. Même si l'équipe de développement se renouvelle régulièrement (trois changements en trois ans), la mise à niveau des nouveaux membres est assez rapide. Toutes les applications fonctionnant sur un serveur web, l'équipe premier niveau/hotline n'est jamais sollicitée : les modifications sont faciles à faire et sont disponibles immédiatement.

Actuellement nous mettons l'accent sur les interfaces client : le code PHP est fonctionnel mais n'est pas suffisant pour répondre aux standards actuels. L'utilisation de Javascript, AJAX par Jquery principalement (cf points forts des assistants de Cake décrits plus bas) est de plus en plus nécessaire.

3.2 Utilisation de Cake

En se basant sur notre expérience depuis presque deux ans de cet outil, nous proposons ici un tour d'horizon non exhaustif de ce qu'il nous paraît être les principales forces et faiblesses de CakePHP.

3.2.1 Les points forts

Cet environnement de développement bénéficie d'une communauté active comme en atteste la sortie récente d'une nouvelle version majeure (cakePHP v2 stable sortie le 16/10/11) et d'un forum d'aide très réactif.

Architecture MVC : comme de nombreux framework cakePHP implémente le modèle MVC (Modèle-Vue-Contrôleur) qui permet de séparer le code associé aux interactions avec la base de données du code associé à la vue. Cela permet d'avoir un code plus clair, mieux structuré et plus souple. Ainsi, comme on peut le voir sur la figure 1, tout projet Cake se voit attribué la même arborescence de répertoires et de fichiers. Outre les répertoires view/controller/model on trouve un dossier config dans lequel se trouve les paramètres de connexion à la base de données ainsi qu'un dossier webroot qui contient les fichiers publics tels que les images, le Javascript, les CSS.

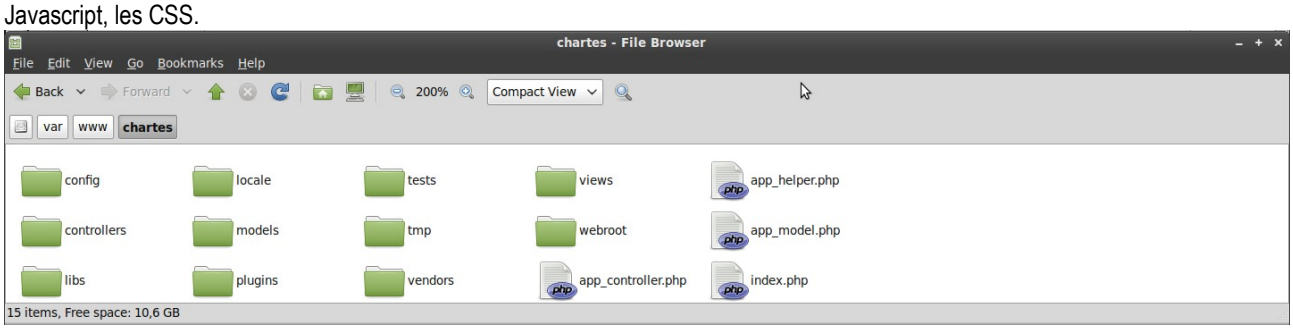

Figure 1 – Arborescence d'une application Cake.

« Privilégier les conventions sur la configuration » . ce principe a été repris de RubyOnRails dont CakePHP s'inspire fortement. Plus on suit ces conventions et moins il est nécessaire d'écrire de code PHP. Par exemple, lors du nommage des tables de la base de données, on est invité à mettre le nom des tables au pluriel (selon les règles anglophones) et à passer les noms des clés étrangères au format nomtable_au_singulier_id. Un champ nommé id pour la clé primaire est également requis. Si ces contraintes sont satisfaites Cake pourra alors automatiquement détecter les différentes relations entre les tables. Ces efforts sont donc récompensés lors des requêtes SQL. Par exemple lors de la récupération de tous les enregistrements d'une table les entrées des autres tables jointes associées à chaque enregistrement seront également récupérées. Finalement on obtient un gain important en termes de temps et de volume de lignes de code. Les conventions sont également utilisées dans le nommage des noms de fichiers et des classes.

Abstraction de la base de données : ici nous faisons référence aux fonctions grâce auxquelles on va pouvoir consulter,mettre à jour, ajouter et supprimer des entrées dans la base de données, ce sont les fonctions CRUD (7). Ces fonctions, associées aux modèles, sont abstraites en ce qu'elles sont indépendantes du SGBD utilisé. On pourra donc changer de SGBD sans avoir à modifier le code des requêtes. Par ailleurs le code des requêtes se révèle plus concis et rapide à écrire et le découpage des différentes portions (conditions, champs sélectionnés) de la requête dans un tableau permet de mieux s'y retrouver dans le cas des requêtes longues. A noter que dans la version 2 le mapping objet/relationnel (ORM) basé sur Active Record (8) a été abandonné au profit de PDO, l'ORM natif de PHP qui assure donc de meilleures performances à l'exécution. L'exemple de la Figure 2 présente le code nécessaire pour l'activation d'une requête en consultation grâce à la méthode find appliquée au modèle Person qui correspond à la table people dans la base de données.

```php
<?php
   // Requête SQL avec l'ORM de CakePHP

   $fields = array('Person.nom', 'Person.prenom','Person.age', 'Person.email');
   $conditions = array('Person.age' => 25);
   $this->Person->find('all',array('fields' => $fields,'conditions' => $conditions,'order' => 'Person.nom'));


  // code équivalent sans le framework CakePHP

  $query = ' SELECT Person.nom, Person.prenom,Person.age, Person.email FROM
             Person WHERE Person.age = 25 ORDER BY Person.nom;

   $base->query($quey);
        if ($base->errno) {
                printf("Sauvegarde du nouvel enregistrement impossible",
                $base->error);
                exit();
        }
?>
```

Figure 2 - *Récupération d'informations sur un modèle*.

« Bake » l'utilitaire en lignes de commandes : à la création d'un projet cake, cet outil s'avère très efficace. En effet avec des commandes simples on va générer tout d'abord le répertoire de l'application avec l'arborescence de base citée plus haut. A l'étape suivante, on renseignera les éléments nécessaires pour la connexion avec la base de données. Ceci fait, cake pourra générer automatiquement les trios modèle/vue/contrôleur associés à chaque table avec pour chacune d'elle les fonctions CRUD de base dans les contrôleurs et vues. Si on dispose du modèle de données il est donc possible en quelques minutes d'obtenir une application minimale avec les fonctions d'administration de base. Une fois encore cela permet un gain de temps appréciable et permet de se concentrer rapidement sur les spécificités de l'application.

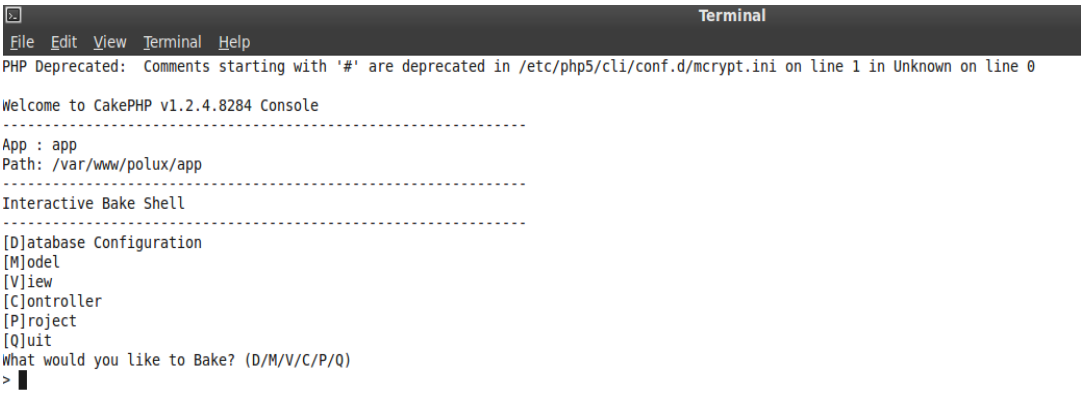

Figure 3 - Accueil de Bake.

La figure ci-dessus présente l'interface de « bake » l'outil en ligne de commandes de génération de code.

Assistants intégrés . Les assistants sont des librairies qui ont vocation à être utilisées au sein des pages de vues. Dans l'implémentation de l'interface graphique la génération de champs de formulaire est une tâche que l'on peut qualifier de répétitive et parfois ennuyeuse. C'est là qu'intervient l'assistant Form. Cette classe est composée de méthodes générant des champs de formulaires de tous types (texte, liste déroulante, boutons radio, cases à cocher,etc) et ce à moindre coût car une seule ligne de code suffit pour générer les balises HTML associées à un champ. De plus, on a la garantie d'avoir un code HTML bien formé, conforme W3C. Dans la pratique cet assistant très complet nous épargne du temps de travail fastidieux et est incontournable dans tous nos projets.

```
  <div class="people form">
<?php echo $form->create('Person');?>
        <fieldset>
                <legend><?php __('Ajouter un utilisateur');?></legend>
        <?php
                echo $form->input('login');
                echo $form->input('nom');
                echo $form->input('prenom');
                echo $form->input('email');
                echo $form->input('statuts_id');
        ?>
        </fieldset>
<?php echo $form->end('Envoyer');?>
</div>


   // code HTML généré par le code ci-dessus
<form id="PersonAddForm" method="post" action="/polux/people/add" accept-charset="utf-8">
<fieldset>
                <legend>Ajouter un utilisateur</legend>
        <div class="input text"><label for="PersonLogin">Login</label><input name="data[Person][login]" type="text" id="PersonLogin" /></div>
        <div class="input text"><label for="PersonNom">Nom</label><input name="data[Person][nom]" type="text" id="PersonNom" /></div>
        <div class="input text"><label for="PersonPrenom">Prenom</label><input name="data[Person][prenom]" type="text" id="PersonPrenom" />
        </div><div class="input text"><label for="PersonEmail">Email</label><input name="data[Person][email]" type="text" id="PersonEmail" /><
        /div><div class="input select"><label for="PersonStatutsId">Statuts</label><select name="data[Person][statuts_id]" id="PersonStatutsId">

<option value="1">Demandeur</option>
<option value="2">Admin</option>
<option value="3">Commission</option>
</select></div> </fieldset>
<div class="submit"><input type="submit" value="Envoyer" />
</div>
</form>
</div>
```

Figure 4 - Assistant Form.

Ci-dessus, un exemple montre la génération d'un formulaire au moyen de l'assistant Form. En haut de la page, se trouve le code PHP et en bas le code HTML produit après interprétation du code PHP. Ceci constitue un bon aperçu du travail qu'effectue CakePHP à la place du développeur.

Concernant le code javascript, l'assistant Js se montre efficace dans la prise en charge d'AJAX. Cet assistant se base sur un framework javascript qui peut être au choix du développeur Jquery ou Mootools. Tout comme les requêtes SQL, il suffit d'appeler une méthode sur l'objet Js et de renseigner les différents paramètres au moyen d'un tableau indexé. Toujours en une seule ligne de code il est très facile de lier une requête AJAX à une balise HTML. Le moteur Js génère automatiquement la requête AJAX ce qui épargne au développeur toute la gestion de l'objet *xmlHttpRequest.* Comme précédemment, cela permet de gagner en vitesse de développement et d'avoir un code moins propice aux erreurs.

Extensibilité et réutilisabilité : de manière récurrente, nous sommes confrontés à chaque nouveau projet à des problématiques déjà traitées dans des projets antérieurs. On est dans ce cas de figure, par exemple, avec l'authentification. On s'aperçoit alors que de nombreux projets partagent la même logique, des parties de code identique. De la même façon, il est souvent intéressant de pouvoir récupérer un même code à différents endroits d'une application. Par exemple, lorsqu'on veut afficher un champ ayant un format particulier (avec calendrier qui s'affiche dans plusieurs pages). Pour cela, Cake propose pour chaque constituant du triptyque MVC des classes permettant d'encapsuler du code dans des objets qui seront réutilisés à plusieurs endroits d'une application ou pour d'autres projets : les composants (contrôleur), les comportements (modèle) et les assistants (vue). Il est également possible de développer des extensions logicielles grâce aux plugins.

Support de qualité : pour finir, CakePHP jouit d'une communauté importante et dynamique. On trouve de nombreux contributeurs ainsi que différents groupes d'utilisateurs. Par ailleurs, le forum français affiche une très bonne réactivité (cakephp-fr.org).

3.2.2 Les points faibles

La documentation n'est pas toujours bien structurée à l'intérieur des chapitres. Pour les débutants, cela peut avoir pour conséquence de rendre la recherche difficile surtout lorsqu'on s'éloigne du code basique illustré par les exemples de la documentation (blogs, livres). Il existe de nombreux cas où les exemples ne sont pas complets et il faut passer beaucoup, beaucoup de temps à tester et chercher pour faire fonctionner le code. En version 1.x, de nombreuses fonctions Cake sont paramétrables avec des tableaux d'options, de profondeur parfois insondable. S'y retrouver et jongler avec objets et tableaux demande un temps important. La documentation la plus à jour est uniquement disponible en langue anglaise.

L'offre de formation est très réduite et n'aborde que les principes simples, identiques aux tutoriels. Il n'existe pas à l'heure actuelle de programme avancé pour comprendre toutes les subtilités et la richesse de ce framework.

On est vite dépassé quand on cherche à sortir du cadre figé des exemples où une vue correspond à une table. Il existe de bons tutoriels pour prendre en main le framework, mais se l'approprier et l'adapter à ses besoins peut être très laborieux au début. Il existe de nombreux sites d'aide et il faut souvent les visiter tous pour avoir une réponse à nos questions.
La convention de nommage peut se révéler contraignante dans certains cas. Par exemple si on reprend une base de données qui n'a pas été conçue selon le modèle de cake : il y aura de nombreux paramètres à changer une fois la génération automatique faite pour écraser les conventions de nommage (relations entre fichiers, BD et routage).

La prise en charge de PHP 4 pour la version 1.3 bride les performances du framework. Dans la version 2 c'est PHP 5.2 qui a été choisi et cela fait l'objet de quelques critiques car cette version n'est plus supportée par la communauté PHP et de plus on ne retrouve pas les nouveaux concepts introduits dans PHP 5.3 que sont les espaces de noms et les fonctions lambda. Par rapport à ses deux principaux concurrents (Zend et Symfony), un certain retard technologique peut être observé : sorties plus tardives et moins complètes en fonctions.

Pour conclure ce retour d'expérience sur l'utilisation de Cake, voici une vue synthétique avant/après :

Grille d'évaluation selon nos critères et à l'épreuve du temps

	Evaluation au moment du choix	Evaluation de nos jours	Commentaires
Cohérence et richesse des classes du cœur	Abstraction SGBD, assistants pour l'interface graphique, I10n et i18n, modules d'authentification.	Toutes les tâches courantes sont très bien prises en charge.	Bake facilite grandement le démarrage d'un projet.
Facilité d'apprentissage	Compromis entre facilité d'accès et puissance de l'outil.	Séparation claire en MVC, fonctions simples aisées, mais toujours une perte de temps quand on cherche à mettre en place une nouvelle fonctionnalité pour la première fois. Le code est suffisamment bien architecturée pour apporter des évolutions lorsque cela est nécessaire.	Bonne ré-utilisabilité du code entre les différents projets.
Extensibilité	Souplesse et ouverture du code.	Un site recense toutes les extensions communautaires. Les fonctions de base du cœur sont assez développées pour ne pas avoir eu à utiliser des extensions.	
Une documentation suffisamment riche et bien organisée	Documentation en français.	La doc française n'est pas à jour et il faut jongler avec plusieurs sites pour avoir une vue exhaustive. Le forum d'aide français est très réactif.	La doc de l'API est la plus complète mais manque d'exemples. Après quelques recherches sur le net et une demande sur le forum, on trouve une solution ou un contournement.
Stabilité	Mises à jour avec des versions mineures régulières.	Très bonne stabilité, évolutions régulières, mais la màj entre 2 versions (1.x et 1.y) demande du travail.	Toutes nos applications ne sont pas au même niveau (1.2 et 1.3, bientôt 2.0).

4 Conclusion

Au sein du pôle urbanisation de la DISI de l'UJF, la suite constituée de PHP, MySQL et de CakePHP utilisée quotidiennement depuis maintenant plus d'un an permet de satisfaire avec efficacité les différentes demandes d'applications sur mesure. Ces demandes se caractérisent, en général, par des dématérialisations de processus administratifs telles que des procédures de recrutement. Dans ce contexte, cakePHP, en automatisant les tâches répétitives du développement d'applications web dynamiques, nous a aidé à être plus réactifs. Après élaboration du cahier des charges, l'utilisation d'un tel outil rend possible la production rapide d'une première maquette. En outre l'architecture orientée objet du framework permet d'apporter facilement et rapidement des modifications ou des évolutions, à la suite, notamment, d'un changement du cahier des charges. En somme, l'adoption d'un framework nous a naturellement amené vers une conduite de projet plus itérative, où les fonctionnalités sont affinées progressivement grâce à des points plus fréquents avec l'utilisateur final. L'intérêt est manifeste quand on sait que, par expérience, le cahier des charges n'est pas figé dès le départ et qu'il est amené à évoluer au cours du cycle de développement.

Récemment notre veille technologique nous a conduit à nous intéresser de près à Drupal, un CMS open-source dont la puissance et la souplesse se sont accrues avec la sortie de la version 7 et la présence de plus de 7000 modules téléchargeables sur le dépôt Drupal. Personnalisable et extensible, il propose de base des modules d'administration très riches qui permettent de générer graphiquement le front et le back-office d'une application. Toujours dans un souci d'améliorer la productivité, mais pas au détriment de la qualité logicielle, il est possible que Drupal puisse constituer une bonne alternative à Cake surtout lorsqu'on constate que les processus à dématérialiser font souvent appel à une gestion de « workflow ». L'appellation CMS de Drupal, s'il venait à le remplacer complètement, deviendrait alors, pour cet outil, mal représentative de ses fonctionnalités.

5 Bibliographie

[1] Mark Story (Cake software foundation, inc.), cakePHP : framework libre de développement rapide sous licence MIT, http://cakephp.org/

[2] Dries Buytaert, Drupal est un système de gestion de contenu sous licence GPL, http://drupal.org/

[3] Fabien Potencié, Symfony : framework open-source de développemnt web sous licence MIT, www.symfony.com

[4] Zend technologies, zend framework est distribué sous licence New BSD, http://framework.zend.com/

[5] Rick Ellis (EllisLab), codeIgniter : framework libre de développement web, http://codeigniter.com/

[6] Qiang Xue, Yii : framework pour le développement d'applications web 2.0, http://www.yiiframework.com/

[7] James Martin, create, read, update, delete (CRUD) dans *Managing the database Environment*, Prentice-Hall, 1983

[8] Martin Fowler, *Patterns of enterprise application architecture*, Addison-Wesley, 2003.