

Drupal, le couteau suisse de la publication d'informations

Florian TURC, Romain SURY, Isabelle VAN DER ZYPPE

Direction du Système d'Information – Université Joseph Fourier – Grenoble I
Domaine Universitaire
2061, rue de la Piscine
38 400 SAINT-MARTIN-D'HÈRES

Université Joseph Fourier
Direction du Système d'Information
B.P. 53
38 041 GRENoble CEDEX 9

Résumé

Drupal est un CMS libre. L'université Joseph Fourier – Grenoble I travaille avec cet outil (dans sa version 7) dans le cadre de la mise en place de sa nouvelle solution technique pour sa communication numérique.

L'UJF développe des applications web en lien avec des applications tierces en utilisant l'interface graphique de Drupal. Des applications de gestion de processus simples sont également créées avec Drupal 7.

L'UJF a également industrialisé le processus de création et de maintenance des sites Web créés sur ses instances Drupal 7.

Mots-clefs

Drupal 7, CMS, Internet, application, framework, cadriciel, architecture, mallette, infrastructure, processus, modules, formulaire, système d'information, UJF.

Introduction

Pour construire les briques applicatives du système d'information et répondre aux différentes demandes des structures de l'établissement, la volonté de la DSI de l'université Joseph Fourier – Grenoble I s'oriente vers la prise en charge de bout en bout de la plupart des développements informatiques. Toutefois ce choix nous contraint d'intégrer des outils de développement en fonction du contexte suivant : demandes de plus en plus nombreuses et de nature diversifiée, équipe opérationnelle réduite et budget limité.

Ainsi depuis trois ans la mise en place de notre environnement technique s'articule autour des technologies libres et open source. Côté serveur Unix/Linux, Apache, MySQL et PHP pour le langage de programmation web. Dans le cas de projets de développement spécifique qui concernent essentiellement la dématérialisation de procédures administratives, nous nous sommes dotés d'un cadriciel de développement, CakePHP, afin de concilier réactivité et qualité logicielle. Si cet outil a montré ses preuves, force est de constater que cette solution n'est pas la plus optimale dès lors que l'on aborde des problématiques de gestion de contenu et de gestion fine des droits. Les développements de sites web institutionnels ouverts sur l'extérieur, de portails de services ou d'un intranet ont en commun deux caractéristiques essentielles : la nécessité pour les services gestionnaires d'être autonome dans la mise à jour du contenu de leur site web ; le besoin de gérer simplement et efficacement les droits d'accès et de diffusion du contenu. Ces caractéristiques nous ont naturellement guidés vers l'adoption de ce qu'on appelle un système de gestion de contenu (CMS).

Au moment où nous comparions les nombreuses forces en présence dans ce domaine, Drupal se démarqua largement par rapport à nos exigences. Nous recherchions alors un outil souple et ouvert pour l'adapter à des besoins spécifiques, avec une couverture fonctionnelle importante et ayant une communauté importante et active. Drupal 7 affiche de solides arguments sur l'ensemble de ces points.

Ces trois années d'utilisation nous ont permis de profiter de la puissance de cet outil et de constater qu'il dépasse le cadre du CMS. Mais cette souplesse ne va pas sans une certaine complexité et son appropriation a été un processus long. Dans cet article nous montrerons l'implantation finale de cet outil dans notre environnement technique. Dans un premier temps nous présenterons la mise en place d'une mallette Drupal, sorte de « package » prêt à l'emploi contenant des modules préinstallés pour déployer un site rapidement. Nous expliquerons ensuite comment nous avons modelé notre architecture système et décliné différentes mallettes par rapport à des familles de cibles fonctionnelles. Enfin, nous dresserons un bilan des leviers et des freins à l'adoption de Drupal dans une équipe de développement.

1 Notre utilisation de Drupal 7

La DSI de l'UJF a commencé à utiliser la technologie Drupal 7 à partir d'avril 2011. Au cours des huit premiers mois, quelques projets nous ont permis de nous familiariser et de découvrir les fonctionnalités de l'application. Les demandes étaient simples tout en étant assez évoluées pour permettre d'élargir notre champ de connaissance de la technologie.

Après ce temps de découverte, face au flux important de nouvelles demandes de projets, une analyse de nos pratiques a permis de mettre au point un modèle simple et structuré de gestion des sites Drupal.

1.1 Élaboration d'une mallette

Notre premier constat fut que les modules de cœur de Drupal 7, bien que nombreux, n'étaient généralement pas suffisants pour nos projets. Nous avons donc décidé de créer un ensemble, appelé « mallette », qui contiendrait les modules du cœur Drupal ainsi que les modules issus de la communauté testés et choisis par nos soins en fonction des besoins que nous avons identifiés.

Nous avons également mis au point un site référent de cette mallette, permettant ainsi d'avoir toujours une copie du projet type pour nos développements. Cela nous permet, à chaque début de projet, de copier rapidement et facilement les fichiers et la base de données de ce site vers le nouveau projet.

1.1.1 Fonctionnalités

La liste ci-dessous développe sommairement la liste des fonctionnalités principales ajoutées au cœur Drupal 7 dans notre mallette.

1. Éditeur de texte et gestionnaire de fichiers

Drupal 7 ne possède pas d'éditeur de texte WYSIWYG dans ses modules de base. Nous avons fait le choix d'utiliser le module *CKEditor*. Le paramétrage des options de l'éditeur est possible en définissant des formats de texte. Un rôle peut se voir attribuer un ou plusieurs formats de texte.

Pour gérer les fichiers déposés sur le serveur via l'éditeur de texte, nous utilisons le module *IMCE* (et ses relatifs, notamment le module *IMCE FileField* qui permet d'utiliser ce module pour les dépôts de fichiers et de médias via les champs au lieu de l'interface Drupal initiale qui n'offre pas la possibilité de déposer dans un sous-dossier).

2. Compléments d'administration

Nous utilisons quelques modules supplémentaires pour améliorer l'interface initiale de Drupal : *Module Filter* pour la gestion des modules ; *Admin Views* pour la gestion des contenus et des utilisateurs ; *Administration Toolbar* pour la barre d'administration.

3. Authentification via un annuaire ou une fédération d'identité

Différents modules existent pour déléguer la phase d'authentification à des dispositifs spécialisés. Le module *LDAP* permet de gérer la connexion vers un serveur LDAP, la récupération d'attributs utilisateur, ainsi que les droits d'accès en proposant l'édition graphique d'un mappage entre des rôles Drupal et des groupes LDAP. Le module *CAS* peut également être intégré. Le module *Shibboleth* intervient dans l'authentification via une fédération d'identité en automatisant l'interaction avec un fournisseur de service. Nous l'avons mis en place en s'appuyant sur la fédération éducation-recherche de RENATER.

4. Formulaire, type de contenus et types de champs

Drupal permet de créer des types de contenus en collectant un ensemble de données dans des champs. Nous avons ajouté un certain nombre de modules¹ afin d'étendre les possibilités de saisie d'informations.

Le module *Webform* (et ses relatifs) permet aussi de créer des formulaires spécifiques sans passer par le principe de type de contenus. Il est plus souple mais ses utilisations sont différentes.

Enfin, nous avons également installé le module *CAPTCHA* pour la prévention de saisie des formulaires par des robots.

5. Vues et requêtes

Le module *Views*, pratiquement incontournable, est un outil puissant de filtrage de données qui se présente de manière graphique. De plus, l'agencement et l'aspect visuel du contenu peut également être paramétré.

6. Fichiers de données

Les modules *Nodequeue* et *Smartqueue* (pour la compatibilité avec le module *Domain Access*) permettent de créer des listes de contenus. Il est possible de les utiliser dans des vues.

7. Filtres et masques

Le module *Custom Filter* permet d'ajouter des filtres (ou masques) au sein d'une zone de texte via l'outil des formats de texte. Il est ainsi possible d'afficher le résultat d'un code PHP sans utiliser ce format de texte généralement utilisable uniquement par les administrateurs (si c'est le format choisi, les autres utilisateurs ne peuvent alors rien modifier dans la zone de texte en question). Cela permet de laisser une autonomie plus grande aux utilisateurs non administrateurs tout en permettant l'affichage de contenus complexes.

La même fonctionnalité existe spécifiquement pour l'affichage du contenu d'un bloc avec le module *Insert block*.

8. Internalisation et multilinguisme

Un ensemble de modules (*Internationalization*, *Localization* et leurs relatifs) fournissent les interfaces nécessaires à la traduction de tout ou partie d'un site. Chaque élément pourra se voir doté d'un onglet de traduction : du type de contenu au champ, les menus, les vues, ou encore les taxonomies.

9. Variables

Le module *Variable* permet d'ajouter des variables au niveau système. Cela permet, dans le cas de notre mallette préconfigurée, de définir rapidement certaines informations utilisées fréquemment dans le site.

10. Processus et règles

Le module *Rules* ajoute la notion de déclencheurs conditionnels et permet de modifier des données en fonction des paramètres.

Le module *Workflow* permet de configurer un processus et de gérer le passage entre les différentes étapes d'un contenu. Il permet aussi d'ajouter des fonctions au module *Rules*.

11. Webservices

Le module *Wsclient* permet d'ajouter des fonctions au module *Rules* pour interagir avec un Web Service. Il est ainsi possible de récupérer des informations et de les enregistrer dans les données des types de contenus.

1.1.2 Mise en commun des données

Drupal permet de mutualiser les données (modules, thèmes, librairies) grâce à sa prise en charge du multi-site. Les fonctionnalités sont donc partagées entre les sites d'une même instance et les mises à jour côté fichiers sont simplifiées.

De plus, les thèmes ont été construits de manière à pouvoir utiliser la notion d'héritage entre eux. Par défaut, un sous-thème est un thème qui reprend l'ensemble des éléments de son thème parent. Il peut ensuite en supplanter ou en ajouter d'autres.

¹ Notamment *BigInt*, *Date*, *Double Field*, *Email*, *Field Collection*, *Field Collection*, *Field Group*, *Field Group Multiple*, *Field Group Views*, *Field Permissions*, *Google Map Field*, *Invisimail*, *Link*, *Phone*, *Serial*, *Unique Field*.

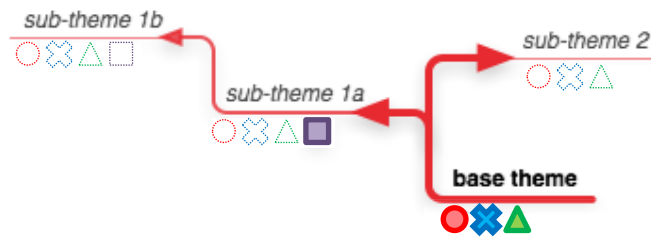


Figure 1 - Héritage entre thèmes²

Nous avons construit nos thèmes avec cette philosophie, partant du modèle *Zen* très utilisé, générateur de HTML5 et web adaptatif. Nous avons quatre couches de thèmes : le thème *Zen* non modifié, évolutif en fonction de ses propres mises à jour ; le thème *Zen to UJF*, thème tampon entre *Zen* et le thème *UJF* ; le thème *UJF*, référence commune pour nos thèmes (en conformité avec la charte graphique de l'établissement définie par le service communication) ; et enfin les différentes déclinaisons de ce thème pour les différents projets (couleurs différentes et parfois quelques détails supplémentaires).

La mise en commun des fichiers permet, outre une diminution de l'espace utilisé, de simplifier les mises à jour et de réduire le temps consacré à cette tâche, même s'il faut passer sur l'ensemble des instances sur les différents serveurs pour la répercuter une fois validée (pas d'incompatibilité avec un autre module ou de perte de fonctionnalités).

Il est aussi possible de partager les contenus entre différents sites en utilisant une seule base de données pour plusieurs sites. La première option est la définition de tables communes et de tables spécifiques. Nous n'utilisons pas cette fonctionnalité, ne la maîtrisant pas suffisamment.

La deuxième option est l'installation du module *Domain Access* qui permet de gérer l'affichage d'un contenu en fonction du domaine d'accès au contenu. Les utilisateurs et leurs rôles sont également mis en commun et les fonctionnalités développées pour un site sont répercutées automatiquement sur l'ensemble des autres domaines. Le module n'est toutefois pas compatible facilement avec nombre de modules de gestion d'accès utilisés dans les autres mallettes (gestion des droits au nœud ou à la section). Nous utilisons donc ce module uniquement lorsque du contenu doit être partagé entre plusieurs sites.

1.2 Mise en place d'une structure dédiée

En complément de l'élaboration de la mallette décrite dans la section précédente, nous avons mis en place une organisation compatible en termes d'infrastructures et de gestion des sites.

Trois serveurs dédiés ont été configurés (un pour les sites en test, un pour les sites en préproduction et un pour les sites en production). Sur chaque serveur, nous avons créé cinq instances Drupal, chacune étant orientée en fonction des différents types de besoins que nous avons analysés (site Internet et intranet de composantes ou de services, site Internet ou intranet de laboratoires, applications web, sites Internet pour le service Grenoble Sciences³, sites Internet autres). Les sites de laboratoires sont séparés des autres sites de composantes ou de services car ces derniers doivent pouvoir répondre à un besoin supplémentaire, à savoir le partage de contenus entre eux. La solution technique est apportée par le module spécifique *Domain Access*.

Un filtrage est effectué sur adresse IP pour chacun des serveurs : uniquement celles de l'équipe de développement pour le serveur de test, l'ensemble du réseau de l'établissement pour le serveur de préproduction et au monde entier sur le serveur de production une fois le site ouvert. Cela complète la gestion des droits opérée par Drupal.

Une nomenclature des *virtual host* a aussi été définie et un fichier de configuration a été créé pour chaque instance dans un souci de symétrie avec leur organisation.

Il est possible de réaliser différentes opérations d'administration des sites via un terminal du serveur. Le module *Drush* (logiciel tiers spécifique à Drupal) fournit cette interface avec des commandes spécifiques à Drupal. L'automatisation de certaines actions via la création et l'exécution de scripts est réalisable avec *Drush*.

² Source : <https://drupal.org/node/225125>.

³ Service d'édition de livres de l'établissement.

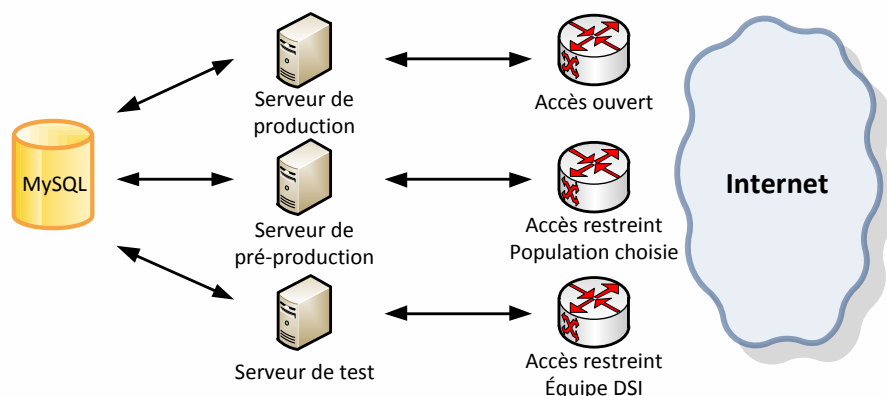


Figure 2 - Organisation de l'infrastructure

1.3 Impact de cette structuration sur l'organisation et la vie du projet

Cette structuration de Drupal a permis de simplifier la partie technique dans l'élaboration de projets informatiques et de communication internes à l'établissement.

Pour chaque nouvelle demande, une analyse du cahier des charges permet de définir les acteurs impliqués (service communication ou DSI).

Dans le cas où nous participons au projet, l'analyse précédente du cahier des charges nous permet également de savoir quelles fonctionnalités sont désirées et, au final, sur quelle instance le site sera hébergée. Cela nous permet également de définir rapidement quels éléments sont à demander au graphiste en fonctions des patrons de thèmes déjà prédéfinis.

Si un développement technique supplémentaire est nécessaire, nous envisageons toujours de l'intégrer dans les fonctions de base de la mallette (évaluation de l'apport).

Pour tout besoin de conseil éditorial, le service communication est sollicité et ce point lui est attribué.

2 Développement dans Drupal 7

Drupal est assez souple pour permettre l'ajout de fonctionnalités personnalisées. Plusieurs méthodes sont envisageables, chacune pouvant être utilisée dans des cas précis.

2.1 Création de modules

La première, la plus évidente ou la plus élégante, est la création d'un module. Un module se compose d'au moins deux fichiers (un `.info` pour que Drupal le repère comme tel, et un `.module` qui contient les premières lignes de code).

Passer par un module a un intérêt si et seulement si l'action souhaitée n'est pas faisable directement dans l'interface ou si elle personnalise les fonctions `hook` qui permettent d'ajouter, supplanter ou supprimer des fonctionnalités élémentaires de Drupal. Par exemple, l'ensemble des `hook_node_access` est vérifié au chargement d'un nœud par un utilisateur. De même, pour altérer les données d'un formulaire à son chargement, on pourra utiliser la fonction `hook_form_alter`.

Un module peut aussi servir à apporter des fonctions de dialogue avec d'autres applications. Il peut alors ressembler à une petite librairie spécifique et réutilisable par un autre module.

Enfin, par défaut, l'AJAX n'est pas utilisé dans l'interface (hormis pour les vues). Toutefois, il est possible de le mettre en place lors de l'altération de la saisie des données dans un formulaire, l'API de Drupal gérant bien l'AJAX.

2.2 Personnalisation de thèmes

Drupal permet de spécifier les thèmes avec des gabarits allant du nœud, à l'élément (champ, bloc...) et au type de contenu. Cette souplesse offre des possibilités de développements spécifiques à ces différentes granularités.

Il est également possible de paramétrer certaines fonctions pour certains thèmes uniquement.

2.3 Intégration de code PHP via l'interface et filtres personnalisés

Drupal permet aussi d'intégrer du code PHP via l'interface directement dans les zones de texte. Ceci est loin d'être recommandé puisque l'ensemble du contenu du champ n'est alors plus modifiable par les utilisateurs qui n'ont pas les droits d'accès à ce format de texte. De plus, insérer des fonctions mal paramétrées risque de rendre le site inaccessible.

Toutefois, il apparaît parfois impossible d'intégrer des éléments à certains endroits du site, sans pour autant vouloir insérer des blocs (blocs souvent mal pris en compte dans les impressions PDF par exemple). C'est notamment le cas des blocs de vues avec plusieurs arguments passés en paramètres car non contextuels. Pour résoudre ce problème, nous utilisons la fonctionnalité de filtres personnalisables, qui permet de remplacer un contenu spécifique par un autre sans pour autant bloquer l'accès au contenu du champ, car l'appel du code PHP a lieu dans le paramétrage du filtre.



Figure 3 - Fonctionnement des filtres personnalisés.

2.4 Temps moyen de développement

Suite à notre utilisation de Drupal, nous avons fait le constat suivant : dans le cas d'un développement de site nécessitant uniquement le paramétrage de modules connus et intégrés dans la mallette, le déploiement peut être effectué en moins d'une semaine par une seule personne à temps plein.

S'il est nécessaire d'ajouter des fonctionnalités supplémentaires (nouveau module ou fonctions de dialogue avec une autre application par exemple), le temps de développement excède généralement ce délai et varie selon la complexité des éléments à implémenter.

Enfin, pour le cas des sites Internet et intranet de l'établissement, de ses composantes et de ses services, avec la mise au point de notre mallette, il nous est maintenant possible de livrer un site en une demi-journée (hors conception graphisme).

3 Exemples d'applications

Pour illustrer les points évoqués dans les deux précédentes parties, nous avons jugé intéressant de vous décrire brièvement quatre projets différents et complémentaires.

3.1 Sites Internet et intranet (établissement, composantes et services)

L'UJF a engagé un projet d'uniformisation des sites Internet et intranet de l'établissement, de ses composantes et de ses services. L'ensemble de ces sites doivent pouvoir partager du contenu et des utilisateurs entre eux tout en ayant une cohérence technique et éditoriale.

Nous avons fait le choix d'utiliser une instance spécifique pour ces demandes (modules, thèmes et librairies centralisés). Nous utilisons aussi le module *Domain Access* pour permettre le partage des contenus et des utilisateurs.

De nombreux utilisateurs participant à ce projet, un travail important d'uniformisation des demandes est effectué constamment afin d'éviter de multiplier les développements à chaque expression des besoins. Les files de données permettent de gérer par domaine l'affichage (dont l'ordre) des ressources partagées (actualités, liens...). Les filtres et masques sont aussi utiles pour la simplification de cet affichage dans les champs disponibles pour les utilisateurs.

3.2 Application Emplois Étudiants

Le suivi et la gestion des emplois étudiants au sein d'un établissement est un processus long sur la durée et par le nombre d'étapes. L'UJF travaille à leur simplification en passant notamment par l'outil informatique, de la demande d'emplois, l'édition du contrat de travail au paiement des payes puis au suivi politique du processus (édition de statistiques).

Les deux ensembles de données principaux de l'application sont le contrat et la fiche emploi. Chacun a son processus et ils se recoupent à différents moments dans l'application. Les données sont saisies dans les champs de base de Drupal. Le processus de chaque type de contenus est géré par le module *Workflow*, complété par le module *Rules*. Pour les contrats, les données sont en partie importées du logiciel de gestion des étudiants APOGEE au moyen d'un *Web Service*. L'affichage des informations (liste des emplois, statistiques...) est généré grâce au système de vues.

3.3 Application Marchés Publics

L'université Joseph Fourier développe une solution interne de publication de ses appels d'offres pour ses marchés publics inférieurs à un certain montant. La donnée principale de l'application est le lot du marché. Ce lot poursuit un pseudo processus : il est créé, affecté à une annonce de publication, indiqué comme en cours d'étude une fois celle-ci terminée, complété par un avis de conclusion une fois la décision prise.

Un lot pouvant avoir un temps de vie différent par rapport à celle de son annonce de publication initiale, nous avons fait le choix de ne pas utiliser le module *Workflow* pour gérer le processus mais de travailler sur la possibilité d'affectation du lot à différents objets pour faire évoluer sa visibilité dans l'application.

De plus, durant cette application, nous avons travaillé sur l'intégration d'AJAX dans l'apparition des différents champs afin de mieux guider l'utilisateur à travers la création d'un lot. Nous avons pour cela créé un module qui complète le formulaire d'édition et de création via les *hooks* en définissant les différents attributs AJAX pour les champs en question.

3.4 Formulaire ADE

Les composantes ont exprimé le besoin d'avoir un canevas complet pour toutes les demandes de création, modification ou annulation de réservation de salles. L'UJF utilise l'application ADE pour gérer l'utilisation de ses locaux (emplois du temps). Il était également souhaité que l'ensemble des utilisateurs concernés soit informé de cette demande (gestionnaire, demandeur, intervenant) par courriel.

Le cycle des informations dans l'application étant très court (saisie dans le formulaire, envoi d'un courriel) et les champs demandés étant simples (champ texte, champ nombre, listes, courriel), le choix a été fait d'utiliser le module *Webform* pour la création du formulaire.

Le point intéressant du projet réside en la création de deux modules : un pour modifier les données proposées par le formulaire au fur et à mesure de son remplissage ; un autre pour récupérer ces données à partir de la base de données ADE dupliquée. Le choix a été fait dès le départ de passer par une manipulation humaine pour modifier le contenu de la base de données du produit tiers (via l'interface de ce produit). De plus, afin de ne pas trop encombrer le serveur inter-universitaire utilisé par le logiciel tiers par des requêtes supplémentaires, nous effectuons une copie quotidienne des tables liées à nos demandes sur un de nos serveurs. C'est cette base qui est utilisée pour notre application, diminuant au passage le temps de réponse.

4 Avantages et inconvénients

Nous avons choisi Drupal pour ses nombreux atouts. Cependant ceux-ci sont parfois contrebalancés par des faiblesses qui peuvent dans certains cas constituer des freins à son adoption. Nous faisons ci-dessous un rapide bilan objectif de notre expérience d'utilisation.

4.1 Inconvénients

4.1.1 Les performances

Avoir une plateforme contrôlable et paramétrable à souhait à partir de l'interface graphique comme Drupal se fait au détriment des performances. On peut citer l'exemple très significatif de la création automatique d'une nouvelle table lors de l'ajout d'un champ à un type de contenu. Ceci produit une structure non optimisée et génère de nombreuses requêtes en base de données. Finalement ce sont les performances à l'exécution qui s'en trouvent dégradées. À l'heure du big data et des nouvelles applications type réseaux sociaux cette ombre au tableau ne doit pas être négligée.

4.1.2 Gestion de la complexité

Certains projets nécessitent d'installer plusieurs modules en quantité plus ou moins importante. Il arrive alors dans certains cas que deux modules soient incompatibles entre eux. Plusieurs voies sont alors possibles : remplacer les modules mais au risque de perdre des fonctionnalités ; développer un module pour adapter les modules initiaux mais cela peut s'avérer fastidieux ; utiliser des modules supplémentaires et les détourner de leur utilisation première. La multiplication de modules qui ont des interactions fortes entre eux peut déboucher sur une instance dont le comportement est difficile à prédire et à maîtriser. Ainsi, le choix d'intégrer Drupal doit se faire en fonction des compétences en place au sein de l'équipe de développement.

4.1.3 Gestion des mises à jour

Lors de la simple mise à jour d'un module sur une instance, plusieurs problèmes peuvent survenir : la nouvelle version n'intègre plus certaines fonctionnalités ; elle fait apparaître des bugs sur l'instance ou une incompatibilité. Chaque mise à jour requiert donc une attention particulière avant bascule en production. Dans le cas d'une mise à jour générale, il n'existe pas de fonctions natives pour automatiser la propagation de l'opération sur plusieurs modules ou plusieurs instances. On doit alors recourir à l'utilitaire *Drush* et à du bricolage de scripts système. Si l'on gère des grosses architectures, c'est une partie du travail qui peut se révéler pénible et chronophage.

4.2 Avantages

4.2.1 Structure modulaire

Le cœur de Drupal présente une structure de base abstraite où tout élément dans sa définition fondamentale est un nœud. Ce concept consiste à tirer parti des mécanismes orientés objets et donc à rajouter des composants facilement et les faire communiquer entre eux. Au final, on obtient un outil modulaire qui pourra prendre des formes très diverses d'un projet à l'autre, adapté au plus près de ses besoins.

4.2.2 Nombreux modules fournis par la communauté

L'architecture modulaire, mise au point par les concepteurs de Drupal, ainsi que le système de contribution publique sur les dépôts du site, qui suit un cycle de développement à plusieurs états (version test, préproduction, production), ont pris une ampleur considérable. Plusieurs milliers de modules sont actuellement disponibles. Lors de l'identification d'une fonctionnalité non présente dans le cœur de Drupal la première chose à faire est de rechercher si un module développé par les contributeurs peut couvrir le besoin. C'est le cas la plupart du temps et finalement le développement de nouveaux modules reste une activité marginale pour un utilisateur de Drupal.

4.2.3 API riche, structurée et bien documentée

L'environnement de développement s'appuie sur la programmation événementielle. En effet, pour agir sur le fonctionnement de Drupal, il est nécessaire de surcharger des fonctions de *callback* (appelées *hooks*) qui se déclenchent lors d'événements particuliers (à la création d'un type de contenu par exemple). De plus de très nombreuses fonctions sont déjà disponibles (quatre milles) et sont très bien documentées. Lorsqu'on développe un module, nul besoin de réinventer la roue, le programme s'insère dans un environnement riche et réutilisable.

4.2.4 Une communauté importante, dynamique et réactive

Outre l'ensemble des modules disponibles, il existe également de nombreux forums sur lesquels les membres sont très dynamiques et réactifs en termes de support. De plus, le CMS est bien répandu au sein de l'enseignement supérieur et de la recherche et cela favorise les collaborations dans les réseaux universitaires. Par exemple, une liste de diffusion dédiée regroupant les établissements Grenoblois a été créée⁴.

⁴ sari-gt-drupal@services.cnrs.fr

Conclusion

Nous avons décrit dans cet article le résultat de notre expérience de l'utilisation du CMS Drupal dans notre environnement de développement. Il est le fruit d'une intégration progressive des différentes demandes de fonctionnalités effectuées au cours des projets traités.

D'un point de vue technique nous avons largement exploité les possibilités du CMS Drupal, que cela soit en termes d'architecture, d'utilisation de modules externes ou de développement par le biais de l'API. Nous disposons de différentes plateformes qui répondent aux spécificités des besoins que nous avons identifiés.

Le spectre d'intervention inclut : le développement de sites publics à destination des composantes, des laboratoires ; le développement de sites intranet pour les services généraux et composantes ; et enfin le développement de projets d'applications simples à complexes.

Ce constat pose la question de la pertinence de conserver deux outils de programmation différents : Drupal et CakePHP. Cette interrogation est d'autant plus légitime lorsqu'on évoque les interactions fortes annoncées entre Drupal 8 et le cadriciel de développement Symfony.

Bibliographie

1. **BYRON A., BERRY A., HAUG N. et al.**, *Composez avec les modules de Drupal*. Paris, France : Pearson Education France, 2009, 485 p. ISBN 978-2-7440-2364-4.
2. **MERCER D.**, *Drupal 7, créer et administrer son site de manière rapide et efficace*. 2^{ème} Ed. Paris, France : Pearson Education France, 2011, 416 p. ISBN 978-2-7440-2475-7.
3. **MELANÇON B., LUISI J., NÉGYESI K. et al.**, *The Definitive Guide to Drupal 7*. New York, NY États-Unis : Apress, 2011, 1047 p. ISBN 978-1-4302-3135-6.
4. **AUBRY C.**, *Drupal 7, concevoir et administrer vos sites web*. Saint-Herblain, France : Éditions ENI, 2011, 494 p. ISBN 978-2-7460-6461-4.

Sitographie

1. <http://drupal.org> – Site officiel de la technologie Drupal.
2. <http://drupalfr.org> – Communauté Drupal France et francophonie.
3. <http://www.drupalfacile.org> – Apprenez Drupal en vidéo et en français.