

PROJET INDUSTRIEL

AlloTools-FAQ



allotools s.a

38, ZAE Le Triangle Vert

L5691 – MONDORF LES BAINS

Luxembourg



**BEN SAID – CORDARY – SONN – SPIGIEL**



# SOMMAIRE

[I. Introduction 3](#_Toc385685873)

[II. AlloTools : Présentation générale 3](#_Toc385685874)

[A. Contexte 3](#_Toc385685875)

[B. Public ciblé 4](#_Toc385685876)

[C. Objectifs 4](#_Toc385685877)

[III. Organisation du projet 5](#_Toc385685878)

[A. Equipe du projet 5](#_Toc385685879)

[B. Réunions périodiques 5](#_Toc385685880)

[IV. Outils utilisés 5](#_Toc385685881)

[A. Symfony2 5](#_Toc385685882)

[B. FosUserBundle 10](#_Toc385685883)

[C. Bootstrap 11](#_Toc385685884)

[V. Analyse 12](#_Toc385685885)

[A. Maquette de l’application: 12](#_Toc385685886)

[B. Design de l’application :](#_Toc385685888) 12

[C. Règles de gestion :](#_Toc385685889) 12

[VI. L’application 13](#_Toc385685890)

[A. La partie client 13](#_Toc385685891)

[B. La partie administration 15](#_Toc385685892)

[a. Connexion 15](#_Toc385685893)

[b. Utilisateurs 15](#_Toc385685894)

[c. Profils 16](#_Toc385685895)

[d. Serveurs 18](#_Toc385685896)

[e. Articles 21](#_Toc385685897)

[f. Catégories 21](#_Toc385685898)

[C. Arbre décisionnel 23](#_Toc385685899)

[a) Fonctionnement 23](#_Toc385685900)

[b) Les questions, au sein d’un arbre 24](#_Toc385685901)

[c) Gestion des arbres 26](#_Toc385685902)

[d) La partie Statistiques 32](#_Toc385685903)

[D. Propositions d’évolution 36](#_Toc385685904)

[VII. Développement 36](#_Toc385685905)

[A. Répartition des tâches 36](#_Toc385685906)

[B. Mise en commun 36](#_Toc385685907)

[VIII. Tests et livraison 37](#_Toc385685908)

[A. Tests 37](#_Toc385685909)

[B. Livraison 37](#_Toc385685910)

[IX. Conclusion 38](#_Toc385685911)

[X. Annexes 39](#_Toc385685912)

# Introduction

Dans le cadre de notre année de Licence professionnelle, un projet de type industriel nous a été confié. Celui-ci a été réalisé en groupes de 4 étudiants. Il constitue une étape importante dans notre formation d’une part il nous permet de mettre en application les connaissances que nous avons acquises au cours de l’année et d’autre part il nous permet d’apprendre à travailler en groupes.

# AlloTools : Présentation générale

## Contexte

Spécialisée dans le développement et la diffusion de Solutions Web et d'Outils d'Aide à la Vente dédiés aux métiers du Bâtiment et forte d'une expérience de plus de 10 ans dans le domaine qu'elle occupe, la société AlloTools S.A. a développé des outils performants et conviviaux utilisés par plus de 6500 utilisateurs en France.

AlloTools met en avant les nouvelles technologies de l'internet au profit de la vente de produits de menuiserie. Leader sur le Web, elle propose une technologie innovante pour la représentation de produits de menuiserie avec plans à l'échelle, visualisation 3D temps réel et tarification instantanée, le tout fonctionnant sur internet !

D'importants fabricants industriels de menuiserie utilisent quotidiennement les Outils d'AlloTools pour leur Gestion Commerciale et CRM (gestion de la relation client), ainsi que de nombreux revendeurs clefs sur le marché, tels que Point P, Réseau Pro, Leroy Merlin, etc...

AlloTools souhaite mettre en place sur son site un système de FAQ performant (foire aux questions) afin que ses clients puissent trouver des réponses à leurs questions sans avoir à appeler l’entreprise.

## Public ciblé

L’outil est donc essentiellement destiné aux clients de la société AlloTools, pour qu’ils puissent en quelques clics trouver des réponses à leurs problèmes et ainsi alléger le travail de maintenance de la société.

## Objectifs

* Réaliser un système de FAQ, afin que les utilisateurs puissent trouver réponse à leurs questions. Un utilisateur pourra afficher des catégories, des questions et des articles en fonction de ses droits. Un champ de recherche, ainsi qu’un système d’arbre décisionnel permet d’aider l’utilisateur à trouver plus facilement ce qu’il souhaite.
* Une partie administration permettra de regrouper des articles venant de plusieurs bases de données dans une seule et pouvoir leurs associer une catégorie, de donner des droits aux utilisateurs et d’accéder à des statistiques.

L’application a été développée avec l’aide du Framework Symfony2. Une base de données MySQL a été utilisée.

Cette application permet donc aux utilisateurs d’accéder à un certain nombre d’articles afin de répondre à leurs problèmes.

Un champ de recherche permet de lister les articles et les catégories qui peuvent intéresser l’utilisateur. Cliquer sur un article permet de l’afficher et cliquer sur une catégorie donne accès à la liste des articles de cette catégorie. Il est également possible de lister des ‘’arbres décisionnels’’. Lorsqu’un arbre est sélectionné par l’utilisateur, une série de questions lui sont posées afin de lui afficher l’article le plus pertinent pour répondre à sa question.

Une solution a également été trouvée afin que l’on puisse récupérer des articles venant de plusieurs bases de données différentes liées à un service de FAQ tel que phpMyFaq ou autre.

Un système de droits a aussi été mis en place afin de sécuriser l’application. A chaque utilisateur est associé un profil qui lui donne accès à une partie des articles. Une personne non connectée n’a donc accès qu’aux articles des catégories publiques.

Enfin nous avons mis en place une partie administration permettant de charger dans l’application les articles venants des différents serveurs et de leurs affecter un profil. Cette partie permet également d’ajouter, modifier et supprimer un profil, un utilisateur ou un arbre décisionnel, et d’afficher des statistiques.

Cette dernière permet de voir pour chaque article l’évolution des avis des utilisateurs par mois sur une année (utile ou non) ainsi que de voir le nombre de consultations d’un article. Cette partie permet également de voir les catégories dont les articles sont le plus affiché et les plus utiles.

# Organisation du projet

Dans ce chapitre, nous allons voir comment nous nous sommes organisés, ainsi que les outils que nous avons choisi d’utiliser pour mener à bien le projet.

## Equipe du projet

Notre équipe est composée de 4 personnes. Nous nous sommes répartis les tâches afin que le projet avance correctement.

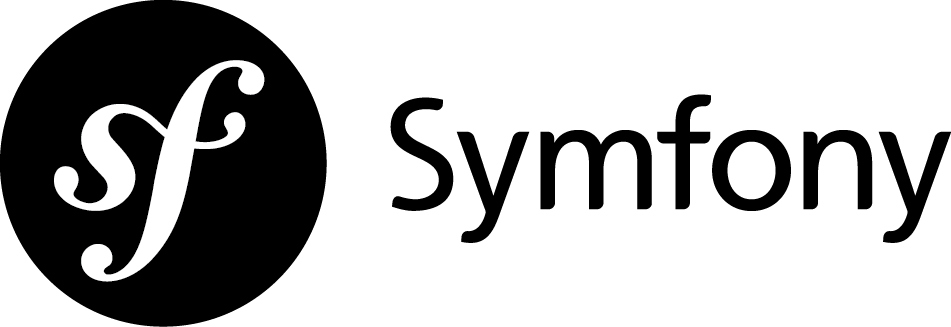
Nous étions encadrés par M.USAI en tant que représentant de la société AlloTools. Son rôle était de représenter le ‘’client’’, de vérifier que le projet était en accord avec ce que souhaitais réellement l’entreprise, et de valider nos choix ou de veiller à ce que le projet entrait les délais fixés.

## Réunions périodiques

Nous nous sommes réunis environ toutes les deux semaines depuis le début du projet afin de voir ensemble l’état d’avancement, de fixer des tâches pour les semaines qui suivaient et de voir si le tout était conforme à ses attentes.

# Outils utilisés

## Symfony2



Symfony est un framework MVC libre écrit en PHP 5. En tant que framework, il facilite et accélère le développement de sites et d'applications Internet et Intranet.

Notre choix s’est porté vers cet outil par soucis de réutilisabilité de la part des développeurs de la société AlloTools. En effet, l’architecture imposée par ce Framework permet beaucoup plus de reprendre le code et de continuer le développement par d’autres personnes, ce qui serait loin d’être évident avec une application PHP réalisée « from scratch ». De plus, c’était une bonne opportunité pour s’auto-former sur un Framework de plus en plus présent et demandé dans le monde du développement web.

Symfony 2.x requiert une version de PHP supérieure ou égale à 5.3.3.

La version que nous avons utilisée est la version 2.4.2 Standard Edition.

Nous avons également utilisé Composer, un outil permettant de gérer les dépendances d’une application PHP, afin d’installer et de mettre à jour Symfony et ses différents bundles.

Symfony propose notamment :

* Une séparation du code en trois couches, selon le modèle MVC, pour une plus grande maintenabilité et évolutivité. Symfony2 utilise le moteur de templates « twig » par défaut.
* Des performances optimisées et un système de cache afin d'assurer des temps de réponse optimaux.
* Une gestion des url parlante, permettant à une page d'avoir une URL distincte de sa position dans l'arborescence.
* Un système de configuration en cascade utilisant pleinement le langage YAML.
* Un générateur de back-office et un installateur de modules.
* L'internationalisation native.
* Une couche de mapping objet-relationnel (ORM) et une d'abstraction de données via Doctrine, une bibliothèque qui fournit des outils puissants afin de plus facilement persister les données.
* Une architecture extensible permettant la création et l’utilisation de plugins.

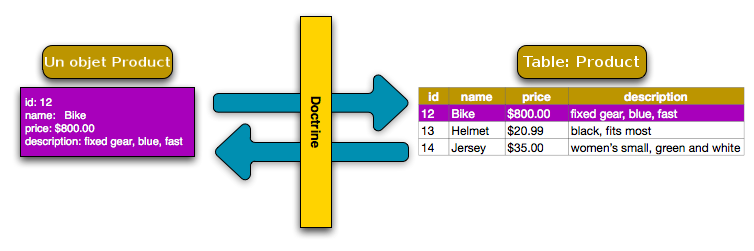
Le Framework fournit également une interface en ligne de commande pour améliorer la productivité en créant un code de base modifiable à volonté.

De nombreux modules déjà créés par la communauté sont facilement récupérables en ligne ou via les lignes de commande et permettent de ne pas avoir à réinventer la roue et permettent donc un gain de productivité. Ils nécessitent cependant de prendre le temps de suivre la documentation fournie afin de comprendre comment les implémenter correctement et efficacement.

Les lignes de commande que nous avons utilisées pour le projet ont permis par exemple de :

* *Générer un bundle :* php app/console generate:bundle
* *Créer rapidement des entités :* php app/console generate:doctrine:entity
* *Vider le cache :* php app/console cache:clear
* *Générer les getters et setters :* php app/console doctrine:generate:entities
* *Créer la base de données :* php app/console doctrine:database:create
* *Générer les tables et champs de la base :* php app/console doctrine:schema:update --force
* Etc…
* ***Doctrine :***

Symfony utilise efficacement Doctrine en récupérant les « métadonnées » liées à chaque entité, ou plus simplement les configurations qui expliquent à Doctrine exactement comment chaque classe et ses propriétés doivent être mappées avec la base de données.



Ces métadonnées peuvent être spécifiées dans de nombreux formats incluant le YAML, XML ou directement dans les classes avec les annotations :

*// src/Acme/StoreBundle/Entity/Product.php*

namespace Acme\StoreBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

*/\*\**

*\* @ORM\Entity*

*\* @ORM\Table(name="product")*

*\*/*

class Product

{

*/\*\**

*\* @ORM\Id*

*\* @ORM\Column(type="integer")*

*\* @ORM\GeneratedValue(strategy="AUTO")*

*\*/*

protected $id;

*/\*\**

*\* @ORM\Column(type="string", length=100)*

*\*/*

protected $name;

*/\*\**

*\* @ORM\Column(type="decimal", scale=2)*

*\*/*

protected $price;

*/\*\**

*\* @ORM\Column(type="text")*

*\*/*

protected $description;

}

A noter qu’un bundle ne peut utiliser qu’un seul format de définition des métadonnées, et que Doctrine est totalement découplée de Symfony. De plus, son utilisation est optionnelle. On peut par exemple aussi persister les données à l'aide de MongoDB en utilisant la bibliothèque ODM de Doctrine, ou utiliser des requêtes SQL brutes avec la couche DBAL de Doctrine.

Afin de récupérer les objets persistés dans la base, il est nécessaire de faire appel au repository de Symfony :

$repository = $this->getDoctrine()

->getRepository('AcmeStoreBundle:Product');

La chaîne « AcmeStoreBundle:Product » est un raccourci qu’on peut utiliser n'importe où dans Doctrine au lieu du nom complet de la classe de l'entité (c.à.d. Acme\StoreBundle\Entity\Product), tant que nos entités sont disponibles sous l'espace de nom Entity de notre bundle.

Une fois que l’on dispose de notre dépôt, nous pouvons accéder à toute sorte de méthodes utiles:

*// requête par clé primaire (souvent "id")*

$product = $repository->find($id);

*// Noms de méthodes dynamiques en se basant sur un nom de colonne*

$product = $repository->findOneById($id);

$product = $repository->findOneByName('foo');

*// trouver \*tous\* les produits*

$products = $repository->findAll();

*// trouver un groupe de produits en se basant sur une valeur de colonne*

$products = $repository->findByPrice(19.99);

Nous pouvons aussi personnaliser le repository pour créer nos propres méthodes de récupération.

* ***Le Routage :***

Symfony2 utilise un système de routage très pratique.

Le routeur nous laisse définir des URLs créatives que nous faisons correspondre à différents points de notre application. Cela permet d’éviter des URL du style « index.php?article\_id=57 » et de favoriser des URLs comme « /article/2 ». Cela permet d’éviter de devoir modifier les chemins dans de nombreux fichiers en cas de besoin par exemple, et de faciliter la gestion des accès à différentes rubriques par exemple.

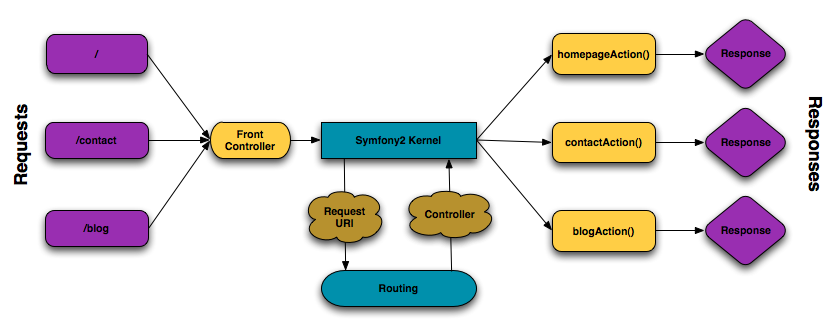
Le routage nous permet notamment de :

* Créer des routes complexes qui correspondent à des contrôleurs.
* Générer des URLs à l'intérieur des templates et des contrôleurs.
* Charger des ressources de routage depuis des bundles (ou depuis ailleurs).
* Débugger les routes.

Une route est une correspondance entre un pattern d'URL et un contrôleur.

Le but du système de routage de Symfony2 est d'analyser cette URL et de déterminer quel contrôleur devrait être exécuté. Le déroulement complet ressemble à ça :

* La requête est gérée par le contrôleur frontal de Symfony2 (par exemple : app.php) ;
* Le cœur de Symfony2 (c.-à-d. Kernel) demande au routeur d'inspecter la requête ;
* Le routeur fait correspondre l'URL entrante à une route spécifique et retourne l'information à propos de la route, incluant le contrôleur qui devrait être exécuté ;
* Le Kernel Symfony2 exécute le contrôleur, qui finalement retourne un objet Response.



*La partie routage est un outil qui traduit l'URL entrante en un contrôleur spécifique à exécuter.*

* ***Twig :***

Enfin, Symfony utilise Twig, un moteur de template PHP flexible, rapide et sécurisé. Il rend les templates plus concis et lisibles; il les rend également plus accessibles aux web designers.

Un template Twig est un fichier texte qui peut générer n'importe quel type de contenu (HTML, XML, CSV, LaTeX, ...). Twig définit deux sortes de délimiteurs :

* {{ ... }}: Affiche une variable ou le résultat d'une expression.
* {% ... %}: Contrôle la logique du template; il est utilisé pour exécuter des boucles for et des conditions if, entre autres.

Pour rendre un template dans Symfony, il faut utiliser la méthode render depuis un contrôleur et passer toutes les variables requises par le template :

$this->render('AcmeDemoBundle:Demo:hello.html.twig', array(

'name' => $name,

));

Les variables passées à un template peuvent être des chaînes de caractères, des tableaux ou même des objets. Twig les gère de la même manière et vous permet d'accéder aux « attributs » d'une variable grâce à la notation (.) :

*{# array('user' => array('name' => 'Fabien')) #}*

{{ user.name }}

## FosUserBundle

Le FosUserBundle ajoute un système de base de données qui permet de gérer les inscriptions, les connexions ou encore les droits d’accès des utilisateurs pour Symfony2.

Il possède un système d’authentification performant et sécurisé, la sécurité qui se compose de deux couches :

* L’authentification qui définit qui est le visiteur,
* L’autorisation, qui définit si le visiteur a accès à la ressource demandée.

Il fournit un cadre flexible pour la gestion des utilisateurs, qui vise à gérer les tâches courantes telles que l’enregistrement de l’utilisateur et la récupération de mot de passe.

Les caractéristiques comprennent :

* Les utilisateurs peuvent être stockés via ORM Doctrine, MongoDB/CoucheDB ODM ou Propel,
* Support d’enregistrement, avec une confirmation par e-mail en option,
* Support de réinitialisation de mot de passe

Pour ajouter un user avec FosUserBundle, il suffit d’ouvrir une fenêtre de commande et rentrer la ligne de commande suivante :

* + *php app/console fos:user:create*

Ensuite il suffit de renseigner un nom, un e-mail et un mot de passe où celui-ci sera encodé en SHA512.

Le fichier security.yml permet de configurer finement chaque acteur de la sécurité :

* La configuration de l'authentification passe surtout par le paramétrage d'un ou plusieurs pare-feu.
* La configuration de l'autorisation se fait au cas par cas suivant les ressources : on peut sécuriser une méthode de contrôleur, un affichage ou une URL.

## Bootstrap



Twitter Bootstrap est un puissant framework CSS. C’est une collection d'outils utile à la création de sites web et applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option. C'est l'un des projets les plus populaires sur la plate-forme de gestion de développement GitHub.

Bootstrap est compatible avec les dernières versions des navigateurs majeurs, mais peut fonctionner de manière dégradée sur des navigateurs plus anciens.

Depuis la version 2, le framework adopte la conception de sites web adaptatifs, permettant aux projets développés de s'adapter dynamiquement au format des supports sur lesquels ils sont consultés (PC, tablette, smartphone).

Bootstrap propose :

* Une mise en page basée sur une grille de 12 colonnes.
* Un reset CSS basé sur Normalize.css
* Du code fondé sur HTML 5 et CSS 3
* Une bibliothèque totalement open source sous licence Apache
* Du code qui tient compte du format d'affichage des principaux outils de navigation (responsive design) : smartphones, tablettes...
* Des plugins jQuery de qualité
* Un résultat cross-browser, en prenant même en compte IE7, donc une garantie de compatibilité maximale
* Une bonne documentation
* La garantie d'une évolution permanente
* Une architecture basée sur LESS, un outil bien pratique qui étend les possibilités de CSS

Tout le design de l’application a été entièrement réalisé à l’aide de cet outil, car facile à implanter et aussi par soucis de réutilisabilité et facilité de modification pour la société AlloTools, qui devra réadapter le design à sa convenance, à défaut de n’avoir pas pu nous fournir une charte graphique à temps.

# Analyse

## Maquette de l’application

## 

## Design de l’application

Une charte graphique aurait dû nous être fait parvenir de la part de la société AlloTools mais cela n’a malheureusement pas été le cas. Nous avons donc pris les devants et avons réalisé notre application avec l’aide de Bootstrap, ce qui facilitera les modifications futures.

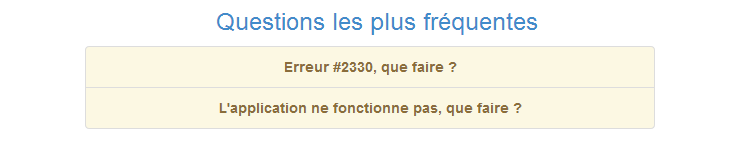
## Règles de gestion

Seuls les admins ont accès à l’administration. En effet, il est nécessaire de posséder le « ROLE\_ADMIN » pour y accéder. Les utilisateurs lambda possèdent le « ROLE\_CLIENT ».

# L’application

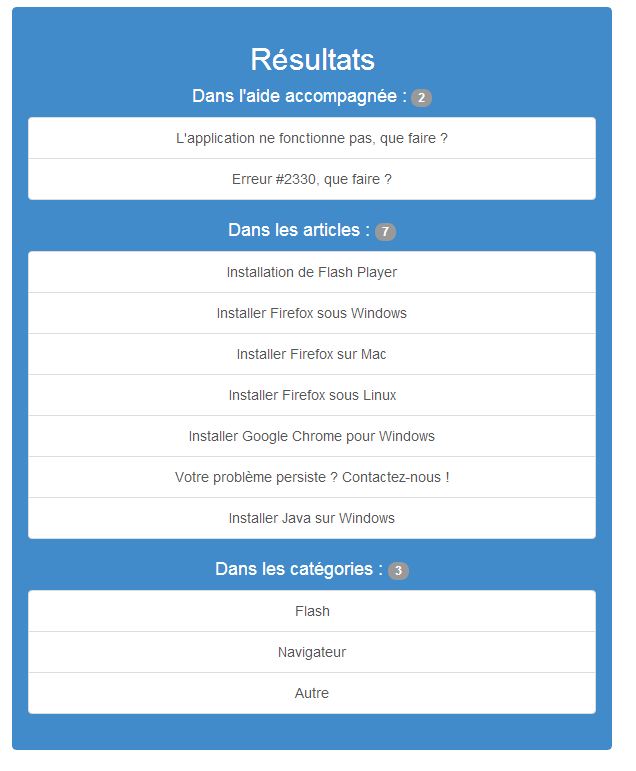
## La partie client

Un utilisateur, qu’il soit connecté ou non, va pouvoir accéder aux cinq  « arbres décisionnels » les plus consultés sur la page d’accueil.



Un utilisateur non connecté ne pourra par contre que voir les catégories publiques ainsi que les arbres et les articles appartenant à ces catégories. Un utilisateur connecté quant à lui possède un profil. Chaque profil donne accès à un certain nombre de catégories ainsi qu’à tous les arbres et les articles de ces catégories. Il a bien entendu également accès, tout comme une personne non connectée, aux catégories publiques.

L’onglet de recherche en haut à droite du menu permet de rechercher et de lister des articles, des arbres ainsi que des catégories en fonction de ce qu’il a écrit dans la recherche. Si rien n’a été écrit la liste de tous les articles, catégories et arbres va apparaitre. Sinon une recherche va être faite sur le libellé des arbres et des catégories ainsi que sur les mots clés de chaque article. Voici les résultats affichés lors d’une recherche :



Si l’utilisateur sélectionne un article, il va pouvoir directement le consulter. Il pourra également définir s’il pense que l’article lui a été utile ou non. Cela permettra par la suite de faire des statistiques :

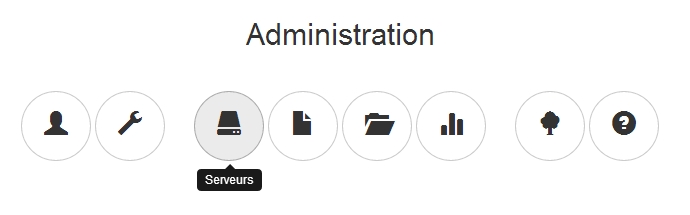


Si l’utilisateur clique sur une catégorie, il va alors voir la liste des articles de cette catégorie. Il pourra alors consulter ces différents articles.



Pour finir, s’il sélectionne une aide accompagnée, c’est-à-dire un arbre, une série de questions va lui être posées, jusqu’à l’amener à l’article le plus pertinent pour répondre à son problème.

## La partie administration



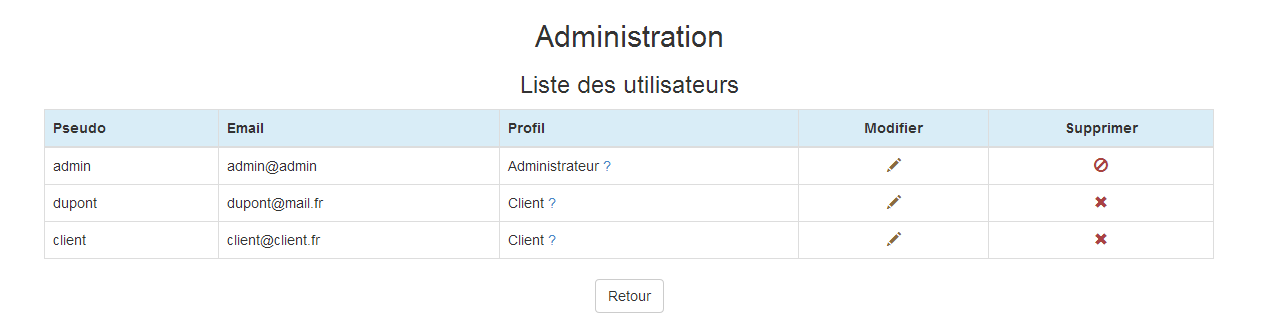
### Connexion

Dans chaque Bundle, on retrouve un .PHP qui est une classe qui hérite de la classe Bundle et qui permet dans notre cas de surcharger le FosUserBundle. Une fois celui-ci surchargé, nous pouvons utiliser nos propres vues plutôt que celles du Bundle en question, permettant ainsi notre propre personnalisation.

### [Utilisateurs](http://localhost/symfony/web/app_dev.php/admin/gestionUtilisateurs/)

Actuellement la partie utilisateur permet de modifier ou supprimer des utilisateurs. Le « super-admin » (qui devra conserver l’id 1), ne peut pas être supprimé. De plus pour des raisons de sécurité quel que soit son profil il aura toujours un rôle super admin. Il est possible de modifier le login, l’email et le profil des différents utilisateurs. L’application ne permet pas (volontairement) de créer de nouveaux utilisateurs.

Néanmoins, il sera possible aux développeurs de la société AlloTools, par la suite, d’importer leurs clients dans l’application.



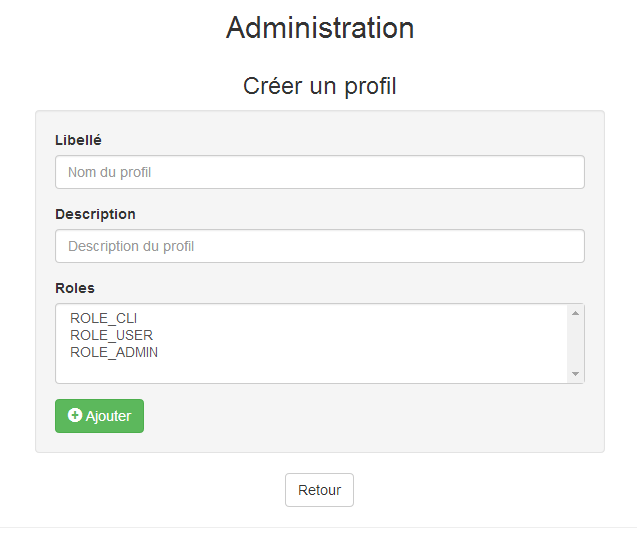
### [Profils](http://localhost/symfony/web/app_dev.php/admin/gestionProfils/)

Nous avons donné la possibilité de gérer les profils associés à chaque utilisateur. Il est donc possible d’en créer, modifier et supprimer. Chaque profil peut avoir plusieurs rôles que l’on indique à côté du menu déroulant et une description a été rajoutée pour chacun d’eux.



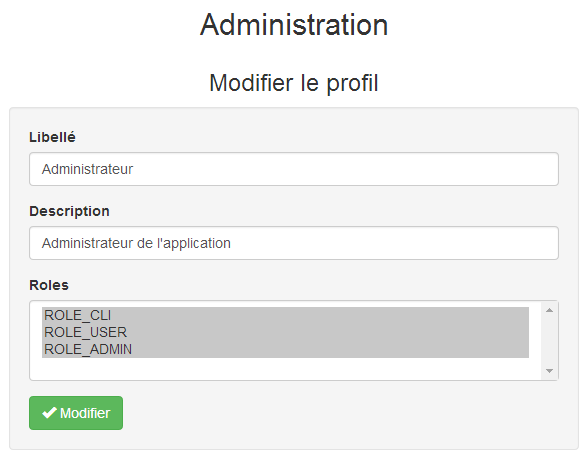
* **Création du profil :**

Lors de la création du profil, l’utilisateur définit un nom et une description de celui-ci et choisit les différents rôles qu’il souhaite lui attribuer.



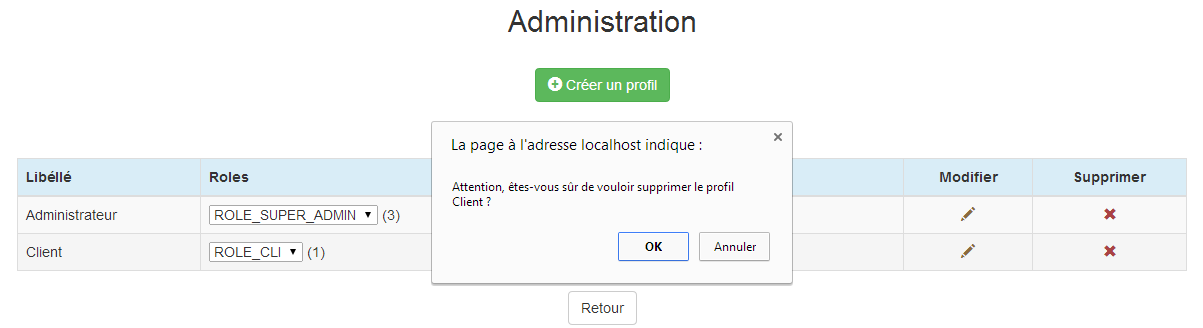
* **Modification du profil :**

Lors de la modification, l’utilisateur peut redéfinir le nom, la description et les rôles du profil.



* **Supprimer un profil :**

Lors de la suppression, un message d’alerte s’affiche et demande à l’utilisateur de confirmer sa suppression.

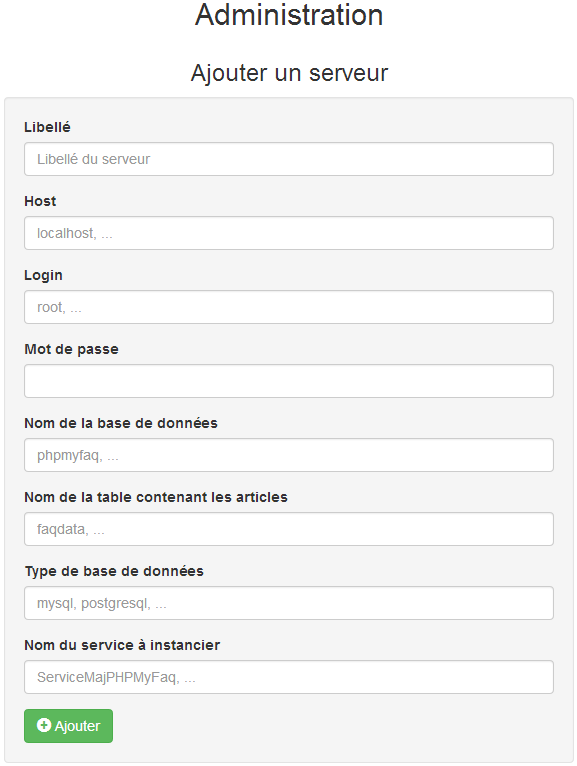


### [Serveurs](http://localhost/symfony/web/app_dev.php/admin/gestionServeurs/)

Voici la vue correspondant à la partie d’administration des serveurs :



Un service a été créé afin de récupérer les articles des différentes bases de données contenant les articles issus des applications de FAQ. La société AlloTools utilisait un service de FAQ nommé phpMyFaq. Afin de rendre ce service de mise à jour compatible avec d’autres types de FAQ ou de base de données (autres que MySQL par exemple), il est nécessaire de préciser toutes les informations requises au bon fonctionnement de l’application :



* **Les informations nécessaires sont :**
* Un libellé (pour le distinguer plus facilement)
* L’hôte (l’adresse du serveur)
* Le login
* Le mot de passe
* Le nom de la base
* Le nom de la table contenant les articles
* Le type de base de données (MySQL, PostgreSQL, etc…)
* Le nom du service à instancier (voir plus bas)
* **Comment fonctionne le service de mise à jour des articles :**

Ce service, nommé ServiceMaj dans notre application, contient 2 méthodes abstraites de mise à jour des articles. L’une permet de mettre à jour les articles sur tous les serveurs présents dans la base de données de l’application, l’autre sur celui de son choix.  
Ces méthodes abstraites vont à leur tour instancier le bon service correspondant à celui préciser dans la base ci-dessus. Ce service en question va s’occuper à son tour de récupérer les articles désirés avec les informations que l’on souhaite, en fonction du service, sachant que chaque type de service correspond à un type de FAQ.

Le seul service qui a été développé pour l’instant correspond à celui de phpMyFaq, utilisé principalement par la société AlloTools. Celui-ci se nomme « ServiceMajPHPMyFaq ».

Afin de faciliter le développement futur de nouveaux services correspondants à d’autres types de FAQ, chaque service hérite de la classe abstraite « ServiceCo » qui contient les fonctions de connexion avec PDO et l’ObjectManager de Symfony2. Enfin, cette classe abstraite hérite d’une interface qui force donc l’utilisation de certaines méthodes au service nouvellement créé, afin de guider le développement de celui-ci et donc de faciliter sa création.

(Voir annexe)

Il est également possible de modifier ou supprimer (après confirmation) les données concernant un serveur.

Enfin un simple clic sur le bouton « Mettre tous les serveurs à jour » permet, comme son libellé l’indique, d’appeler le service de mise à jour de tous les serveurs. Si un article n’est plus présent sur un serveur distant, celui-ci sera supprimé de la base de données. Il est également possible de mettre un seul serveur à jour en cliquant sur le symbole suivant :

Le nombre d’articles qui ont été mis à jour ou supprimés, ainsi que le nombre d’arbres désactivés par une éventuelle suppression d’articles sont affichés :



Si un article présent sur un serveur distant passe d’inactif à actif et que celui-ci est utilisé dans un arbre qui repasse complet en conséquences, alors celui-ci sera réactivé.

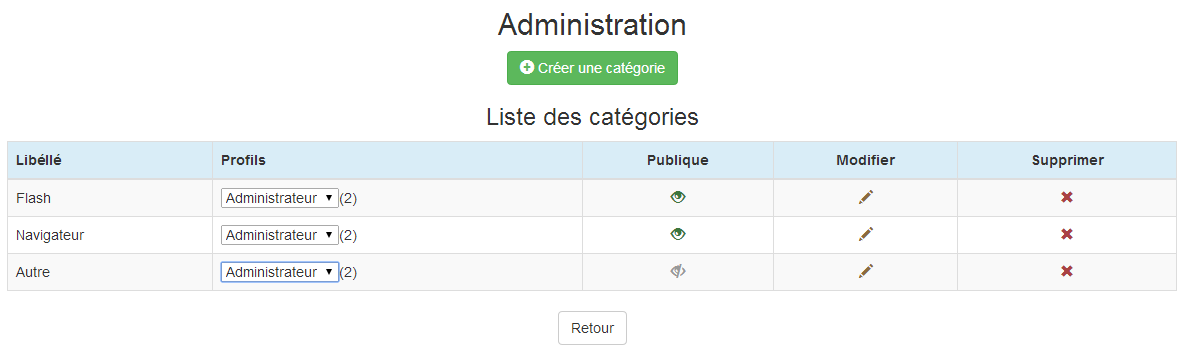
### [Articles](http://localhost/symfony/web/app_dev.php/admin/gestionArticles/)

Actuellement la partie articles permet uniquement d’associer des catégories aux articles. En effet l’ajout, la modification et la suppression d’articles se fait uniquement (volontairement) dans la partie serveur, en récupérant des articles d’autres bases de données. Un tableau affiche la liste des articles avec certaines informations. On y retrouve son titre, l’auteur, la date à laquelle il a été posté (sur le serveur distant), les catégories associées sous forme de liste, le libellé du serveur d’où il provient, et un œil vert si il est actif ou un œil gris barré s’il est inactif. Il suffit de cliquer sur le « crayon » de la colonne « modifier » pour assigner des catégories à un article.



### [Catégories](http://localhost/symfony/web/app_dev.php/admin/gestionCategories/)

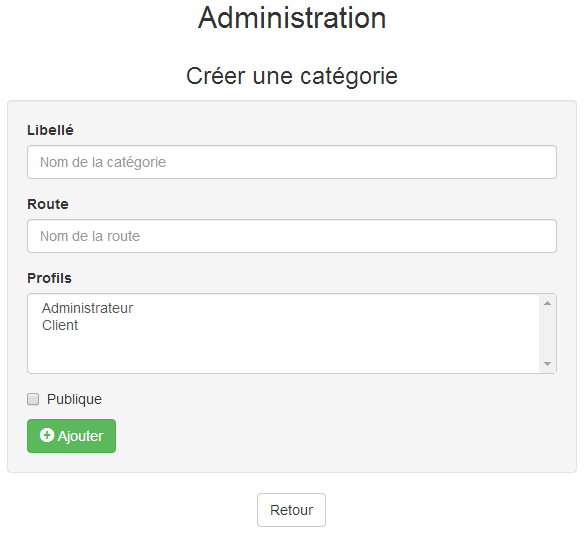
Chaque article est « rangé » dans une catégorie que nous définissons ci-dessous :



Nous avons associé chaque catégorie à des profils pour définir quels types d’utilisateurs peuvent les visualiser. On indique le nom, quels profils y ont accès, si la catégorie est publique ou non, ainsi que la possibilité de modifier et supprimer la catégorie voulue.

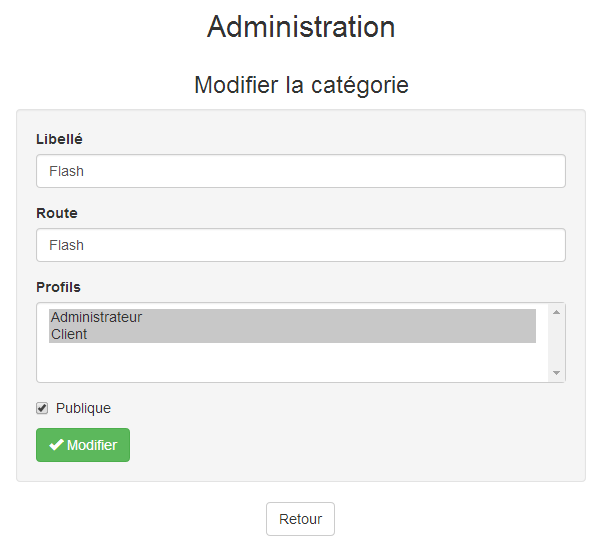
* **Créer une catégorie :**

Lors de la création d’une catégorie, on définit son nom, sa route, les profils qui ont accès à cette catégorie et si l’on souhaite rendre la catégorie publique ou non.



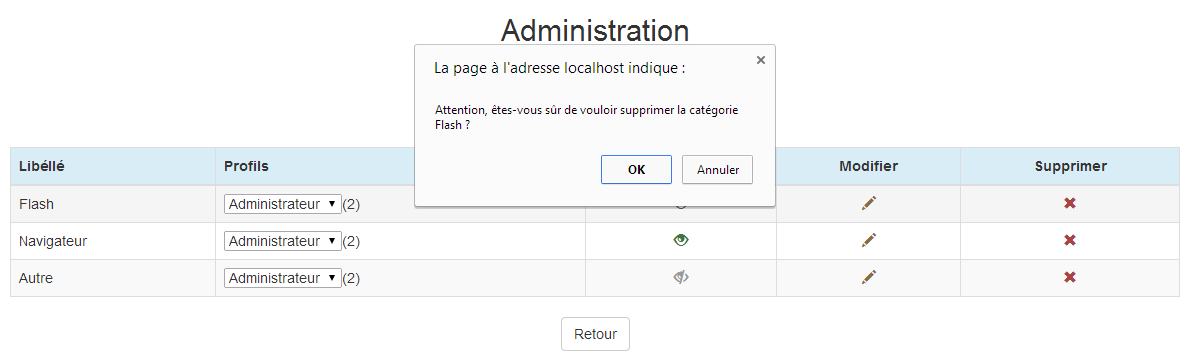
* **Modifier une catégorie :**

Lors de la modification, l’utilisateur peut redéfinir le nom, la route, l’accès des profils et modifier l’accès publique de la catégorie.



* **Supprimer une catégorie :**

Lors de la suppression, on demande encore une fois la confirmation de l’utilisateur.



## Arbre décisionnel

### Fonctionnement

Un arbre décisionnel est composé de feuilles, ou dans notre cas de «  fils » et appartient à une catégorie.

Une feuille peut être un lien vers une question ou un article.

Dans le cas d’une question il faut que l’utilisateur puisse répondre par Oui ou par Non à cette dernière. Ainsi une feuille se compose de deux liens supplémentaires vers d’autres feuilles correspondant à la réponse Oui et Non.

Une feuille de type « question » ne représentera jamais un état fini. En effet l’utilisateur pourra toujours répondre Oui ou Non. Alors il faudra, pour une question, un lien vers une feuille Oui et Non.

Dans le cas d’un article, il représente la fin d’un état de l’arbre. Ce dernier est le résultat des réponses aux questions. Alors il n’existera jamais de liens vers une réponse Oui et Non.

En résumé :

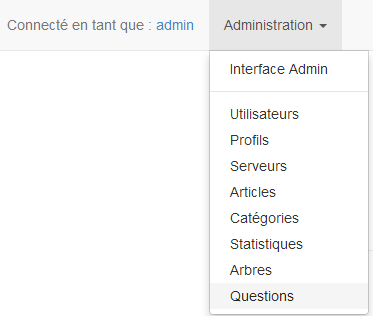
* Un arbre est rattaché à une catégorie
* Un arbre contient des feuilles
* Une feuille contient 2 feuilles si celle-ci est une question, ou aucune s’il s’agit d’un article
* Une feuille de type question est un état non fini
* Une feuille de type article est un état fini de l’arbre dans cette branche

Il a fallu mettre en place un système récursif qui permette de détecter la réponse du client à la première question. Puis en fonction de cette dernière, récupérer la feuille correspondant au Oui si la réponse est un Oui. Puis de boucler sur elle-même, affichant la feuille suivante, etc…

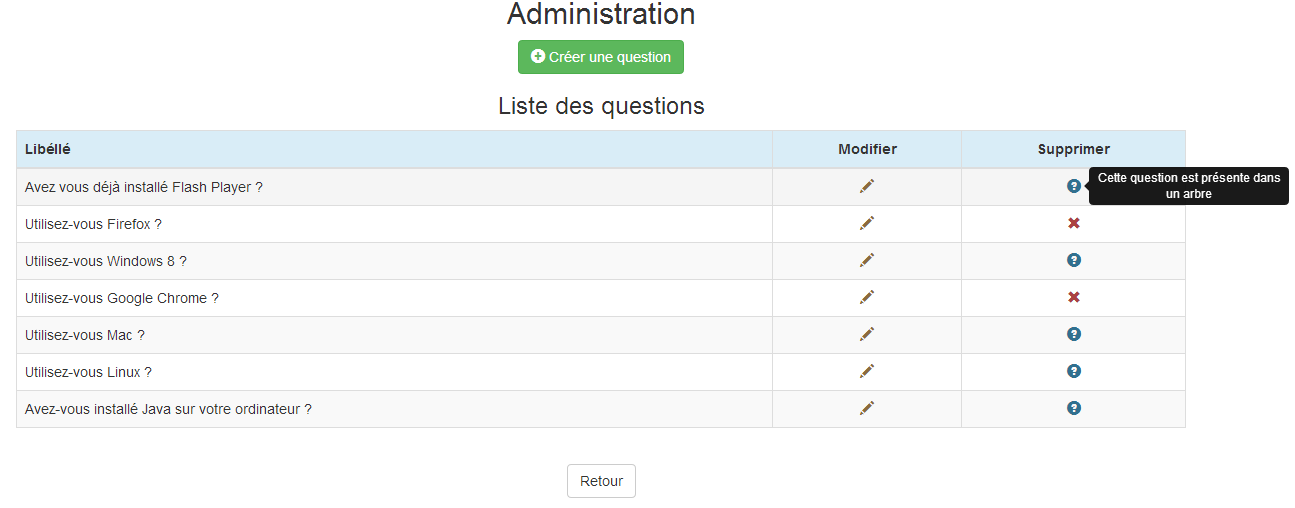
### Les questions, au sein d’un arbre

Un arbre étant indirectement composé de questions, une gestion autour de ces dernières a dû se faire.

Les opérations : Ajout, Modification et suppression ont été intégrées dans la partie Administrative comme nous pouvons le constater sur cet écran :

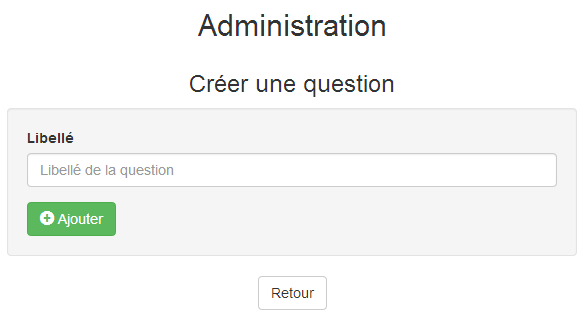


Lorsque l’on clique à cet endroit du menu, l’interface suivante s’affiche :



Les questions sont listées proposant pour chacune la modification et la suppression.  
On observe par ailleurs que certaines questions ne peuvent être supprimées et l’option est remplacée par un point d’interrogation sur fond bleu, ce qui signifie que cette question est présente dans au moins un arbre lorsqu’on place le curseur sur ce dernier. Ainsi on n’autorisera pas sa suppression. Il faudra que l’administrateur supprime la question des arbres dans lesquels elle se trouve pour pouvoir la supprimer.

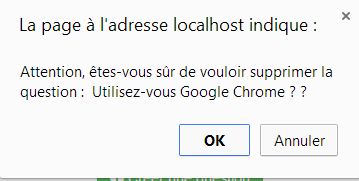
Plus haut, on y voit un bouton vert qui permet la création d’une question.



On remarque ici qu’une question est composée seulement d’un libellé. Il suffit à l’utilisateur de cliquer sur « Ajouter » en vert pour que l’insertion se fasse.

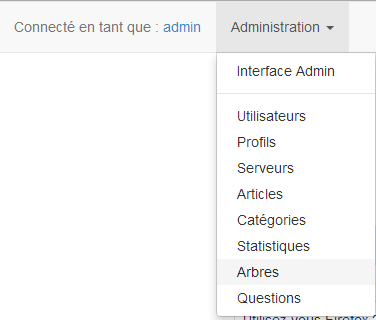
La modification est basée sur le même formulaire à la différence que le libellé sera déjà pré-remplit pour faciliter la tâche à l’administrateur.

La suppression est munie d’un avertissement qui permet de confirmer l’action effectuée :



### Gestion des arbres

Les opérations possibles pour un arbre sont elles aussi : L’ajout, la modification et la suppression. Il est possible d’arriver à l’interface de gestion en cliquant sur « Arbres » dans le menu Administration :



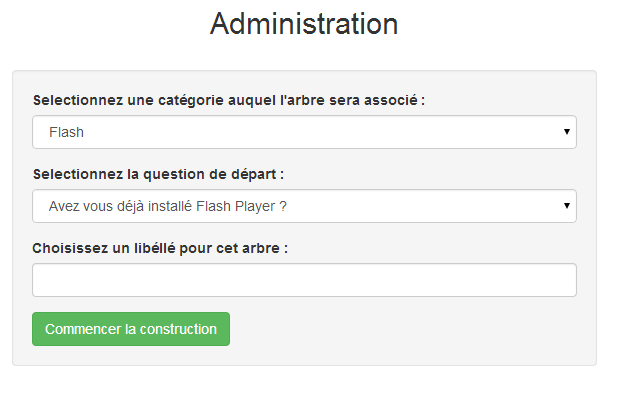
Pour alors obtenir cet écran :



Nous remarquons effectivement qu’un arbre est rattaché à une catégorie, et possède un libellé qui est le problème que les utilisateurs rencontrent. De plus on observe un champ « Complet » qui permet de s’avoir à l’administrateur si cet arbre possède des articles qui ne sont plus existants ou inactifs dans la base de données dû à une mise à jour des articles.

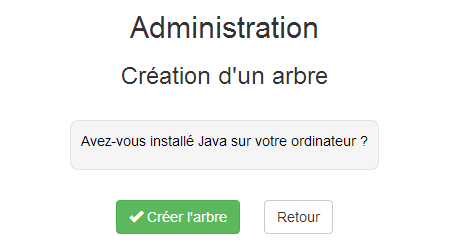
Si tel était le cas, alors le pouce vert se transformerait en pouce rouge, et l’arbre ne serait pas visible dans la recherche.

Lorsque l’administrateur désire créer un arbre, il lui suffit de cliquer sur « Créer un arbre » :

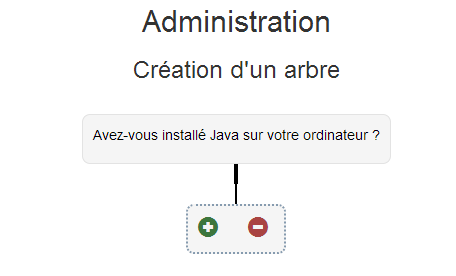


Ici, l’utilisateur devra sélectionner une catégorie auquel l’arbre sera rattaché, puis sélectionner la première question (état initial) qui sera présentée à l’utilisateur lorsqu’il consultera l’arbre, et pour finir, renseigner un libellé.

Lorsque ces étapes sont faites, la construction de l’arbre peut s’effectuer.

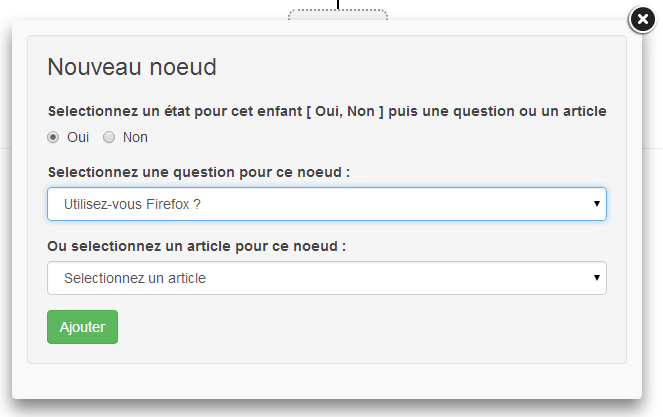


On remarque que la question initiale se retrouve en tête de la construction, il suffit ensuite à l’administrateur de survoler avec son curseur le parent, ici « Avez-vous installé Java sur votre ordinateur ? », pour qu’une feuille temporaire se présente en dessous du parent.



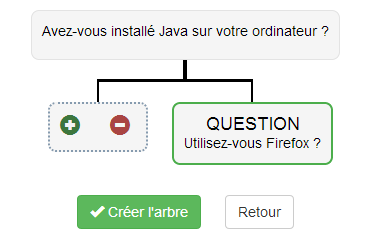
Cette feuille temporaire, possède une option d’ajout. Cela signifie : « Je veux créer une réponse à la question qui se trouve au-dessus de moi. »

Lorsque l’on clique sur cette dernière, un pop-up fait son apparition qui permet la création de la réponse ou encore d’un « nœud ».



Ce formulaire propose d’indiquer s’il s’agit de la réponse oui ou non de la question parente, puis de choisir quel sera la nouvelle question à poser à l’utilisateur OU de choisir l’article qui pourra aider son problème. On observera ici un OU EXCLUSIF, qui permet d’avoir seulement une question ou un article et non pas les deux.

Une fois les informations remplient il suffit de cliquer sur « Ajouter » :



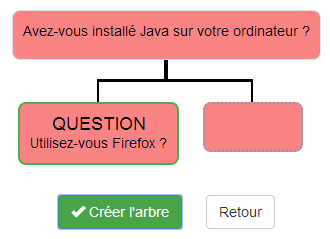
On remarque que l’arbre commence à prendre forme dû à son système de hiérarchie visuelle.

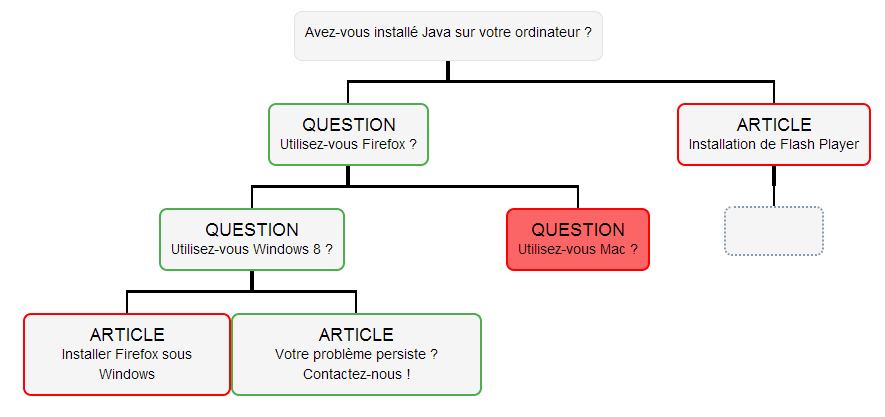
En effet nous avons utilisés JQuery ainsi que le plugin JOrgChart qui permet de visualisé sous forme de hiérarchie une suite de balises <ul> et <li> qui donne cet effet d’arbre.

Rappel : Dans notre cas, chaque nœud représente une feuille. Si la feuille est une question, il faut forcément 2 feuilles enfants au contraire d’un article qui ne doit posséder aucun enfant.

Il ne sera pas possible à l’administrateur de « Créer l’arbre » dans ce cas, puisqu’il manque dans la question principale une réponse « Non ». Et dans la question « Utilisez-vous Firefox » les réponses « Oui » et « Non ».

L’administrateur sera alors averti par un flash rouge sur les nœuds incomplets :

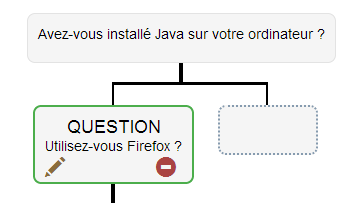


Autre exemple : 

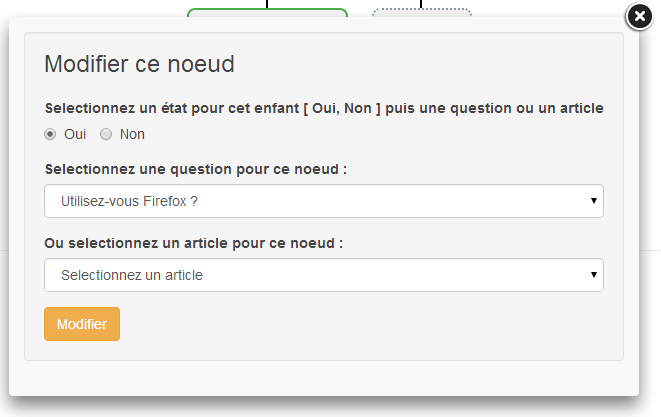
Ici chaque nœud a son état fini (possède un article en fin de branche) sauf celui qui est ici en rouge.

Par ailleurs, les cadres rouges informent l’administrateur qu’il s’agit de la réponse Non du parent, et ceux en verts, considérés comme une réponse Oui du parent.

Lorsque l’administrateur survole une feuille non temporaire, une nouvelle option s’ajoute.



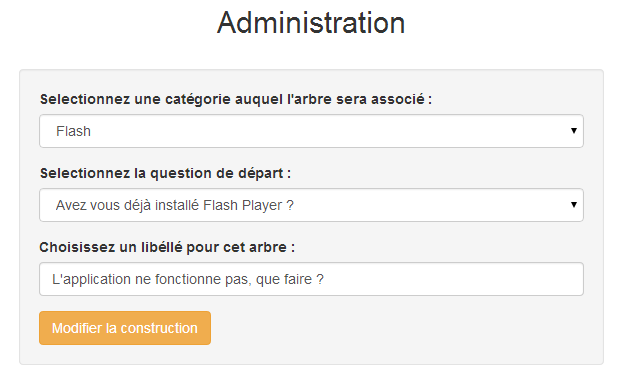
Celle de pouvoir modifier la feuille que l’on survole. Lorsque l’on désire faire cette opération, le pop-up suivant fait son apparition :



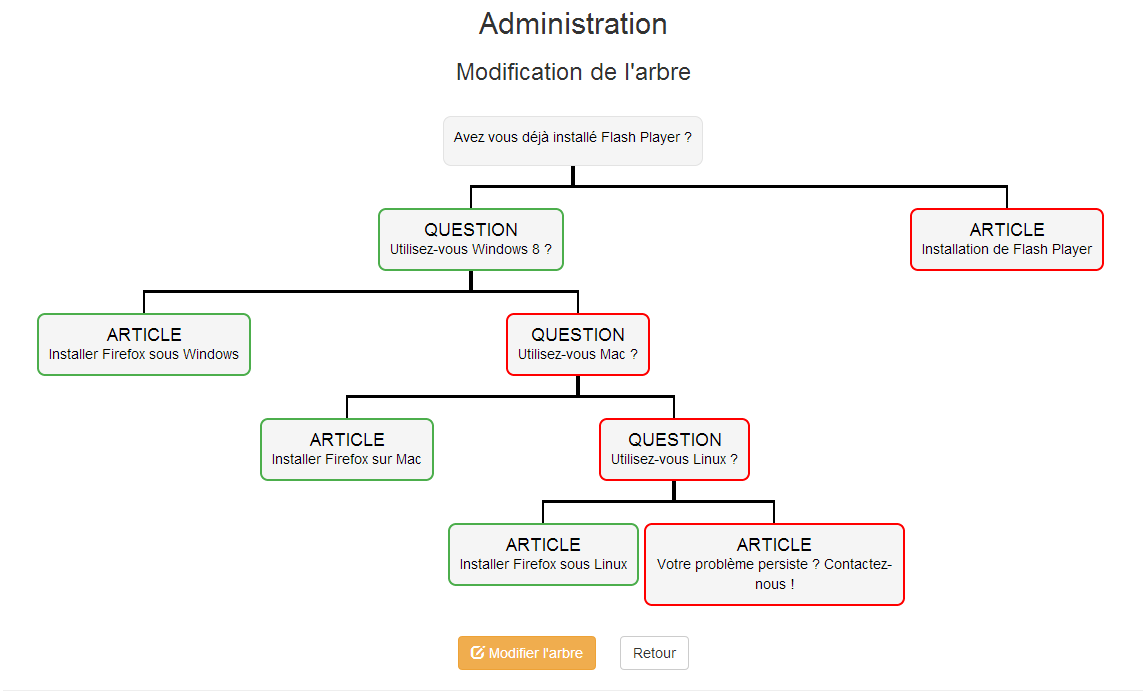
Dans ce-dernier, les champs sont déjà pré-remplis avec les informations contenues dans la feuille que l’on veut modifier. Il suffira de cliquer sur « Modifier » pour que l’édition fasse effet.

Il est aussi possible de pouvoir supprimer une branche en cliquant sur le logo rouge de droite. Cependant, comme un arbre, lorsque l’on scie une branche on perd les branches accrochées à cette dernière. Ainsi tous les rattachements à ce nœud s’effaceront.

Lorsque l’administrateur désire modifier tout un arbre grâce à l’interface de gestion, il est amené à cet écran :



Permettant d’éditer le premier nœud de l’arbre sur ce formulaire et par la suite, toute sa construction :



### La partie Statistiques

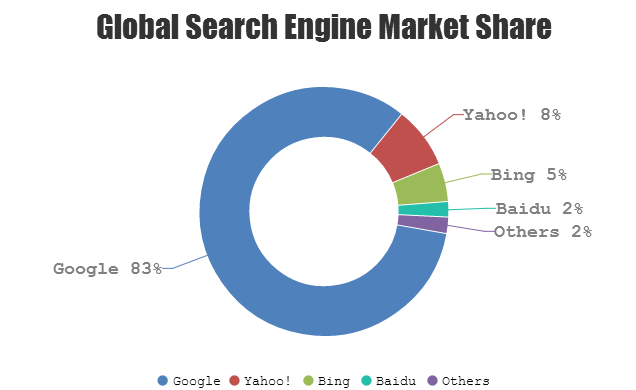
Cette partie n’est accessible qu’aux administrateurs. Elle permet de fournir des données statistiques sur les résultats obtenus par l’application. Tous les graphiques de cette partie ont été produits à l’aide de CanvasJS.

CanvasJS est une librairie JavaScript et HTML5 qui permet de construire des graphiques facilement.

Ces graphiques sont compatibles avec tous les navigateurs Web ainsi que les téléphones portables, tablettes…

Il est possible de produire une grande variété de graphiques différents comme par exemple des graphiques en camembert…

Voici un exemple de graphique produit à l’aide de CanvasJS :



Pour utiliser la bibliothèque il suffit de mettre le fichier canvasjs.js dans le dossier web/js et de faire appel à ce fichier <script type="text/javascript" src="{{ asset('js/canvasjs.min.js') }}"></script>.

Pour finir il suffit dans la vue de construire les différents graphiques et de leurs fournir les données dont ils ont besoins.

Pour que ces statistiques soient réalisées, il a été nécessaire d’enregistrer dans la base de données pour chaque article et pour chaque mois le nombre de fois qu’il a été consulté, ainsi que le nombre de fois qu’on lui a donné un avis positif ou négatif.

Tout d’abord il est possible de voir un résultat global. On peut voir le nombre total d’articles vus, d’avis positifs et négatifs. Il également possible de choisir l’année sur laquelle on veut voir les résultats. Si on sélectionne « Toutes les années », on fait une recherche sur l’ensemble des données.

Ces données sont représentées sous la forme de donuts :

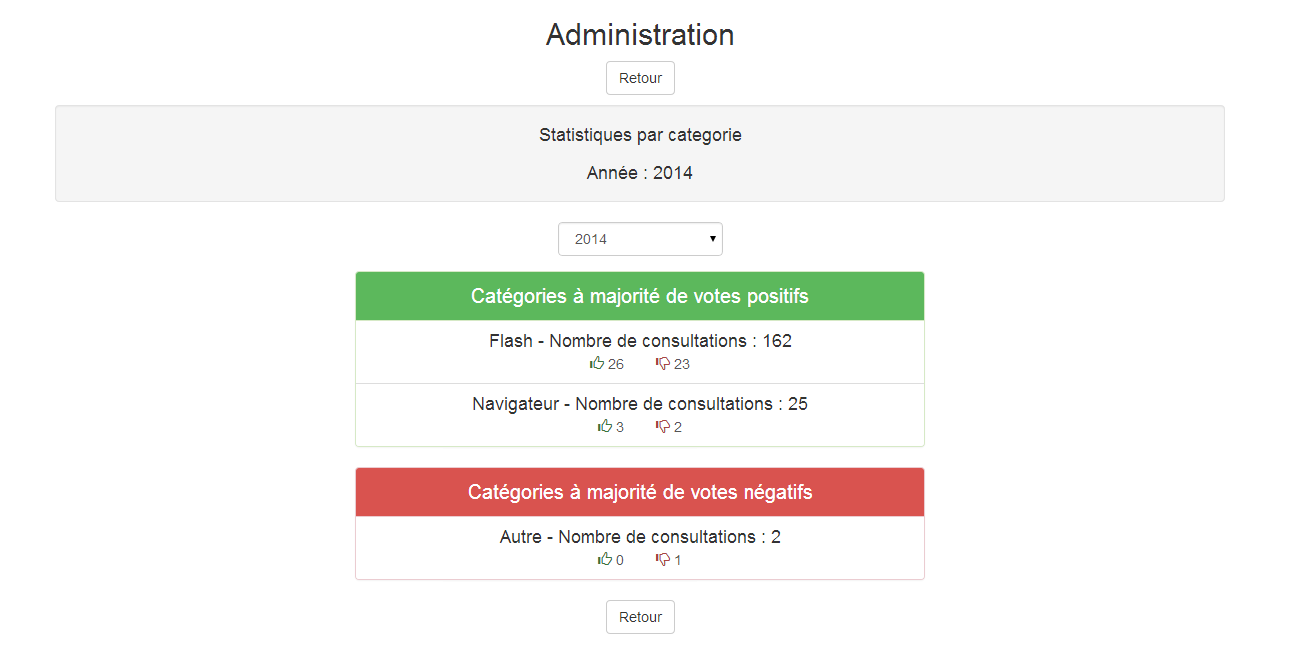


Ensuite en cliquant sur « avis sur les catégories » on accède aux différentes catégories. Pour chaque catégorie, il est possible de voir le nombre d’avis positifs, négatifs et le nombre de consultations. Les catégories sont divisées en 2 groupes afin d’avoir plus facilement une vision sur l’utilité des différentes catégories :

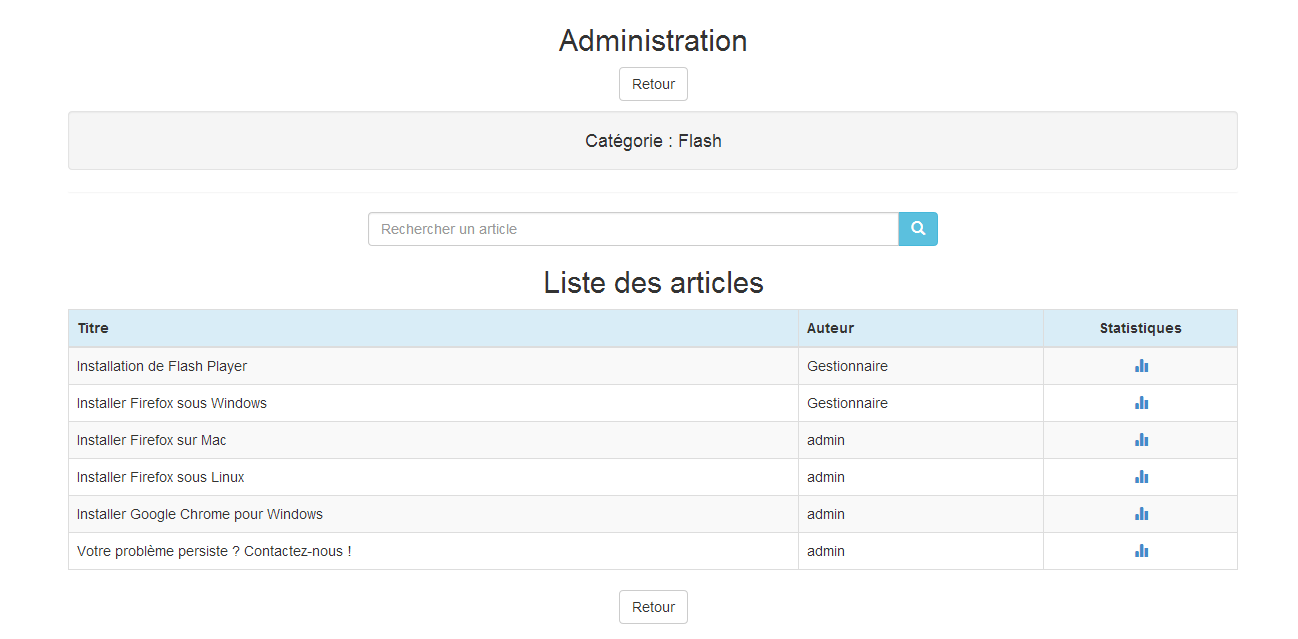
* les catégories où le nombre de votes positifs est supérieur ou égal au nombre de votes négatifs.
* les catégories où le nombre de votes négatifs est supérieur au nombre de votes positifs.

Les catégories sont triées par nombre de consultations.

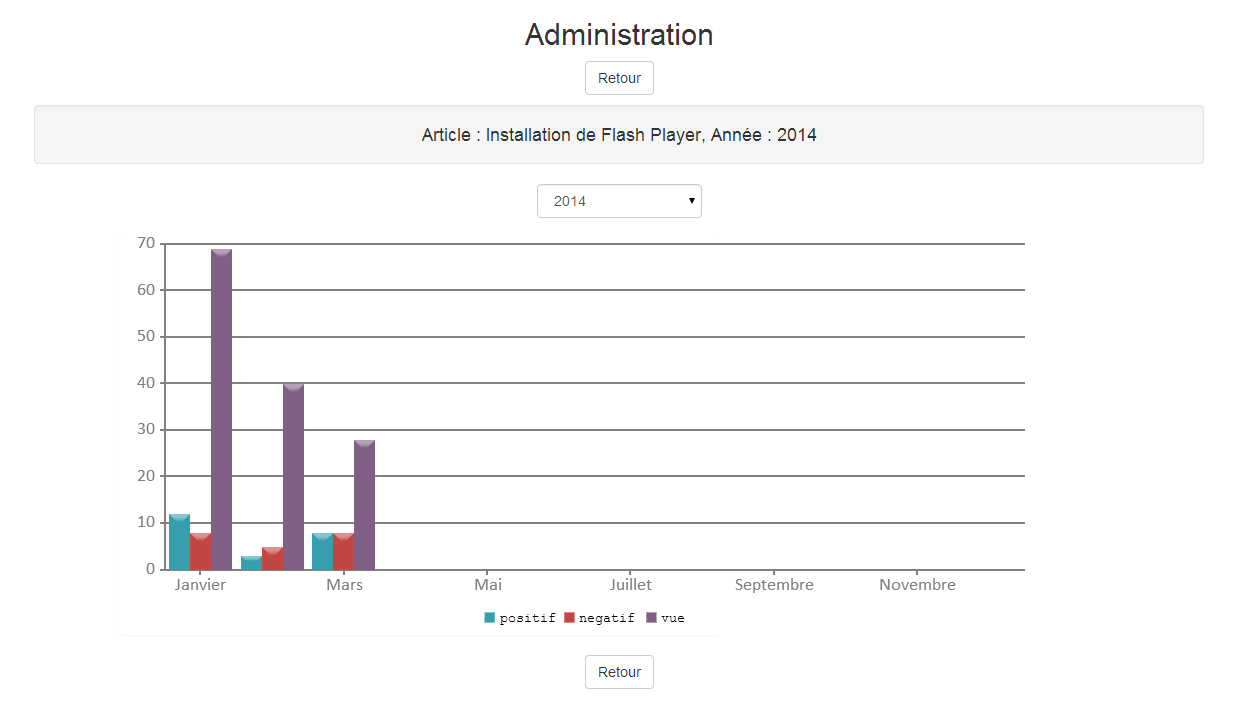
Comme dans la partie précédente, il est possible de sélectionner l’année sur laquelle on veut faire la recherche.



Si on clique sur une catégorie, on accède à tous les articles de cette catégorie. Il est possible de faire une recherche sur le titre des articles :



Si on clique sur les statistiques d’un article, on obtient pour l’année en cours, pour chaque mois de l’année, le nombre d’avis positif, négatifs et le nombre de consultations sous la forme d’un histogramme. Il est possible de sélectionner l’année sur laquelle on veut les données. Cela permet de voir l’évolution des résultats qu’obtient un article au cours d’une année :



## Propositions d’évolution

Voici quelques propositions d’évolutions possibles de l’application :

* Mettre en place une pagination dans les différentes parties de l’application (ex : lorsque l’on affiche la liste des articles)
* Permettre de pouvoir modifier des articles avec l’application et de mettre à jour les données sur les serveurs distants en conséquence
* Mettre en place un chat afin de pouvoir communiquer avec le personnel de la société lorsqu’un article n’a pas été utile

# Développement

## Répartition des tâches

Les tâches à effectuer pour mener à bien le projet ont été divisées et réparties entre les différents membres de l’équipe afin que le développement soit plus rapide et efficace :

**Jean-Michel Sonn** **:** Design de l’application, création du service de mise à jour des articles depuis un ou plusieurs serveurs distants avec PDO, affichage des questions les plus fréquentes.

**Bruno Cordary :** Statistiques à l’aide de CanvasJS, moteur de recherche et consultation d’un arbre, mise à jour des arbres lors de la mise à jour des articles.

**Vladimir Spigiel :** Ajout, modification et suppression d’articles, utilisateurs, profils, serveurs, catégories, questions, implémentation du bundle FosUser, gestion des arbres.

**Nasym Ben-Saïd :** Ajout, modification et suppression d’articles, utilisateurs, profils, serveurs, catégories et questions, surcharge du bundle FosUser et personnalisation des vues en conséquence.

## Mise en commun

Une fois les différentes parties achevées, la mise en commun a été facilitée grâce à l’utilisation de Symfony2. En effet, il n’y avait en grande partie qu’à modifier le contrôleur et les routes pour que les différentes parties soient réunies. Ainsi, le développement a pu être effectué sans qu'il soit contraint par la nécessité que le groupe soit au complet. Chacun a donc pu développer indépendamment tout en permettant ainsi une progression plus rapide du projet.

# Tests et livraison

## Tests

Les différents écrans ont tout d'abord été testés lors de leur développement par les personnes qui en étaient en charge. Ainsi, nous nous sonne assurés que chaque module puisse fonctionner indépendamment du reste de l'application pour qu'il s'y intègre facilement.

Lors des réunions, de nouveaux tests ont été effectués en présence du responsable du projet afin de vérifier le bon fonctionnement de chaque partie de manière plus approfondie et de recevoir un avis externe aux membres du groupe. De plus, tester en présence de l'équipe au complet a permis d'effectuer des tests plus précis et rigoureux.

Une fois l'application réalisée dans sa globalité, l'équipe a testé celle-ci « comme un tout » et non plus en ciblant un module précisément. Pour cela, un jeu de données a été mis en place. De cette manière, nous avons pu nous assurer qu'elle répondait aux spécifications précisées dans le cahier des charges et qu’elle soit approuvée par M. Usai avant d'en effectuer la livraison.

## Livraison

L’application livrée aux utilisateurs de la société AlloTools marchera de manière indépendante. Il sera nécessaire d’intégrer notre application au site de la société, et tout a été mis en œuvre pour que cette intégration se fasse le plus facilement possible. Néanmoins il sera nécessaire aux développeurs de la société d’importer leurs clients dans l’application, de la même manière que nous avons importé les articles.

Enfin, ils pourront mettre en place une tâche automatisée afin que les articles dans l’application se mettent régulièrement à jour afin de ne pas avoir à le faire manuellement.

Malheureusement, nous n’avons pas pu recevoir la charte graphique de l’entreprise, il sera donc peut être nécessaire de modifier la nôtre afin qu’elle soit le plus proche possible du site dans lequel l’application va être intégrée.

# Conclusion

Au cours de ce projet, nous avons développé un système de FAQ performant afin que les clients de la société AlloTools puissent trouver des réponses à leurs questions sans avoir forcément à les appeler.

Avant de développer l’application, nous avons d’abord dû questionner le responsable de la société afin d’avoir des compléments d’information sur l’application que nous devions développer. Nous avons ensuite effectué une phase d’analyse pour préparer le développement de l’application.

Lors de cette phase, nous avons :

* Etudié le cahier des charges fourni par le client, nécessaire au fonctionnement de l’application
* Défini les différents droits d’accès aux différentes parties
* Elaboré la maquette de l’application
* Et essentiellement appris d’avantage sur les outils nécessaires au développement de l’application en s’auto-formant (Symfony2 ou Bootstrap)

Après cette phase, nous avons débuté la phase de développement. Durant cette phase, nous avons développé les différentes parties de l’application une à une. Nous les avons ensuite testées afin de s’assurer que tout soit le plus fiable possible.

Une fois cette phase achevée et les tests sur chaque partie terminés, de nouveaux tests ont été effectués sur l’application dans sa globalité afin de voir si elle fonctionnait correctement et si elle respectait bien le cahier des charges, et certaines modifications ont été apportées petit à petit suite aux différentes réunions avec notre encadrant.

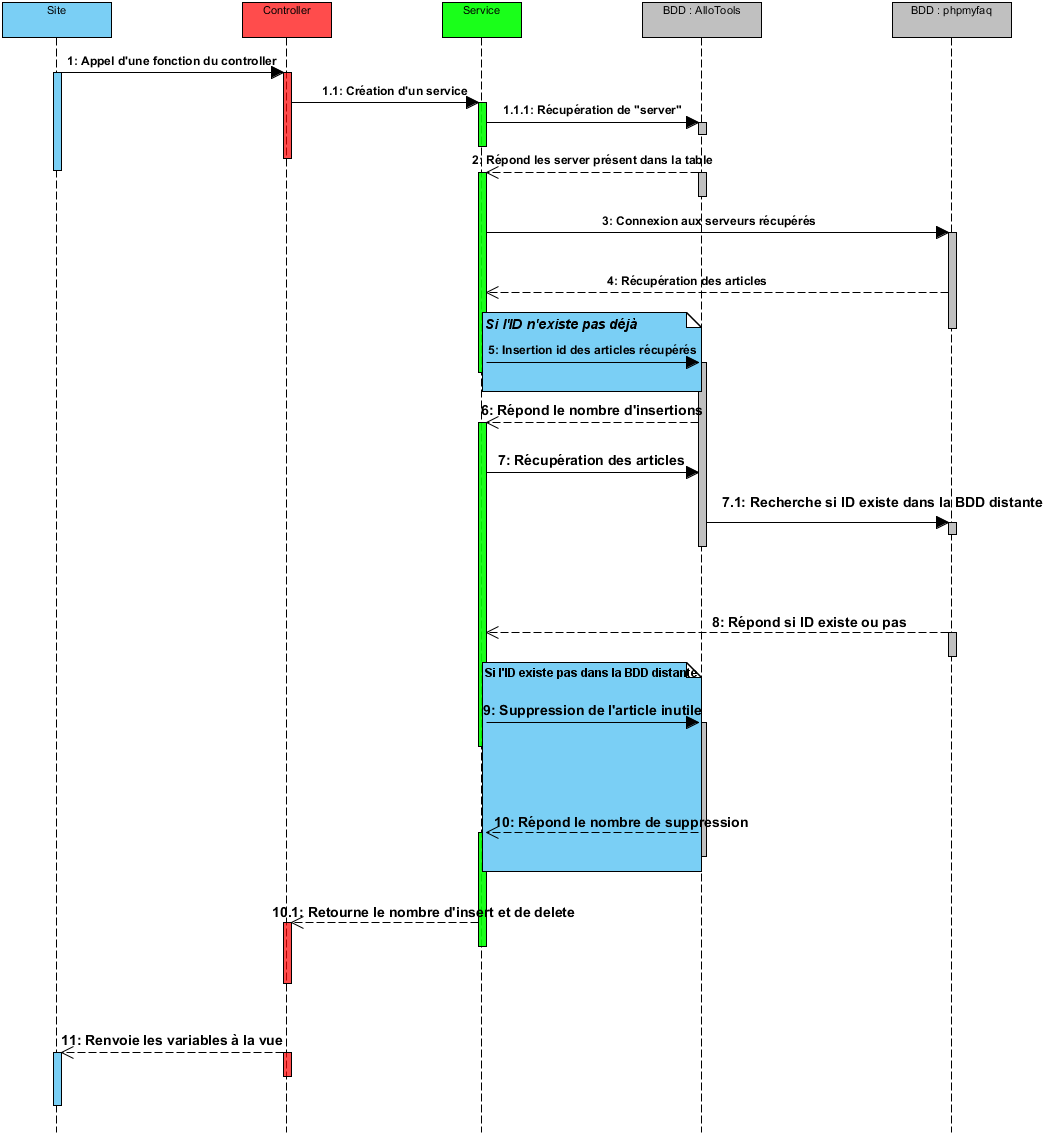
Enfin, ce que nous retiendrons essentiellement de ce projet sera les apports qui ont été nombreux et très positifs. En effet, il nous a permis :

* d’enrichir nos connaissances en développement Web, et surtout nous familiariser avec le framework Symfony2 qui est l’un des plus utilisés en entreprise
* de développer notre sens de l’organisation
* de prendre des initiatives, de travailler en équipe et de faire preuve d’autonomie

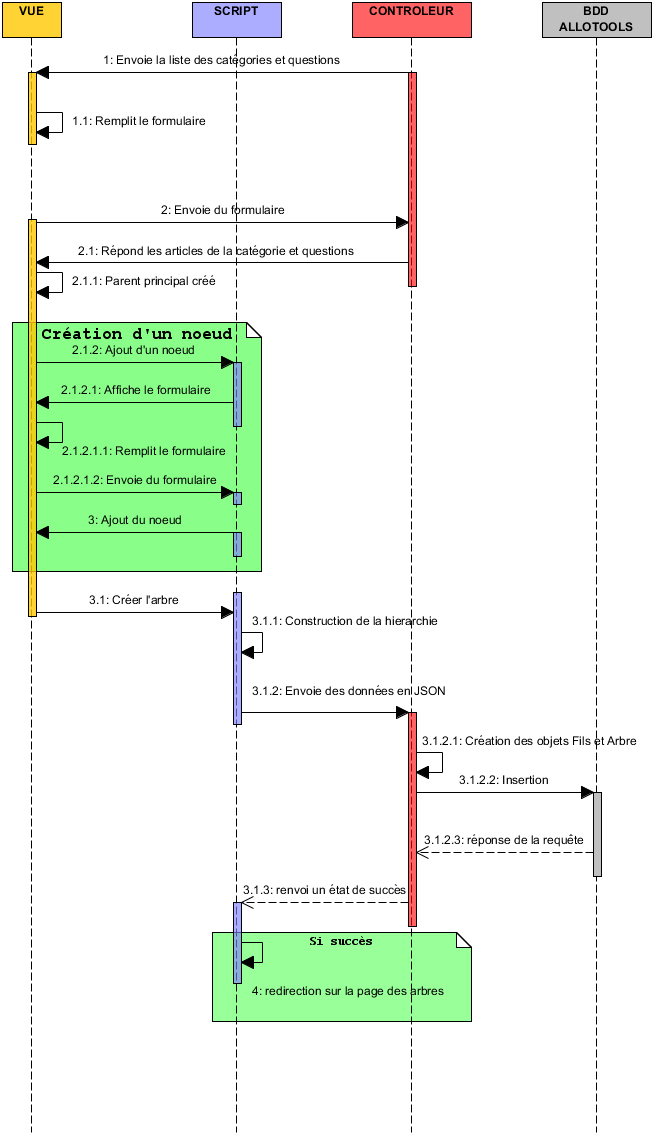
Pour toutes ces raisons, ce projet restera une expérience très enrichissante et extrêmement positive à tous les points de vue.

# Annexes

* **Diagramme séquentiel du Service de mise à jour des articles :**



* **Diagramme séquentiel de la création d’un arbre :**



* **Schéma E.A :**

