

# *Rapport de stage*

*Création d'un site WEB d'association  
avec le Framework PHP  
SYMFONY2*

L'Internet, un enjeu pour les associations...

*Formation  
CONCEPTEUR & DEVELOPPEUR  
INFORMATIQUE  
effectuée du 12 Nov. 2014 au 29 Mai 2015*

*LAHMADI Sadok  
28/29 Mai 2015*

# *Avant-propos*

---

L'Internet est une formidable plate-forme de communication fondée sur la technologie du réseau égalitaire au financement équitable. Ce réseau mondial de grande envergure permet d'établir des échanges entre n'importe quels ordinateurs dans le monde. Cet échange se concrétise par la mise à disposition de l'un des nombreux services de l'Internet : Courrier électronique, publication de documents, d'images, de films et de sons sur le WEB, forum de discussion, partage de logiciels...

C'est certainement pour ces raisons évoquées, que les Associations ont choisi l'Internet en complément des outils classiques de communication (revues de presse, affiches, brochures...) pour faire connaître leur projet, recruter des adhérents et des bénévoles, organiser des événements ou diffuser des informations...

Un site web permettra sans aucun doute d'être l'outil de communication privilégié de l'association...

# *Abstract*

---

The Internet is a great communication platform based on the technology of egalitarian network equitable funding. This global network enables large scale to establish exchanges between any computers in the world. This exchange will result in the provision of one of the many Internet services: Email, publication of documents, pictures, film and sound on the Web, discussion forums, sharing software...

This is certainly for these reasons mentioned , that the Associations chose the Internet to complement traditional communication tools ( press releases, posters, brochures ...) to make known their project, recruit members and volunteers organize events or disseminating information ...

A website will undoubtedly be the communication tool of the Association...

# *Remerciements*

---

Je tiens, tout particulièrement, à remercier Mr **GASSAMA Ba Seckou**, ancien stagiaire du centre M2I et titulaire du même diplôme que j'envisage d'obtenir, pour m'avoir accordé sa confiance et permis d'entreprendre avec beaucoup d'intérêts, la réalisation de son site.

Je voudrais également remercier, Monsieur **BUGUET Pascal**, mon tuteur de stage, pour son soutien et sa disponibilité, ainsi que l'ensemble du personnel enseignant pour nous avoir dispensés d'une formation de grande qualité et m'avoir permis d'acquérir voire d'approfondir mes connaissances.

Sans oublier l'ensemble du personnel administratif, pour leur amabilité et disponibilité.

Et surtout, adresser mes remerciements à l'ensemble de mes collègues avec lesquels j'ai partagé des moments forts et inoubliables, qui ont contribué à enrichir mon expérience tant sur le plan professionnel que personnel.

# Sommaire

<i>Avant-propos</i>	1
<i>Abstract</i>	2
<i>Remerciements</i>	3
<i>I. INTRODUCTION</i>	7
1. <i>Présentation des acteurs du projet</i>	7
1.1 <i>Maître d'ouvrage : le client</i>	7
1.2 <i>Maître d'œuvre : le stagiaire</i>	7
2. <i>Le projet en question</i>	8
2.1 <i>Les critiques de l'existant</i>	8
2.2 <i>Le déroulement du projet</i>	9
2.3 <i>Le cahier des charges fonctionnel</i>	9
2.3.1 <i>L'espace public</i>	9
2.3.2 <i>L'espace privé</i>	10
<i>II. ANALYSE ET CONCEPTION : UML</i>	13
1. <i>Le diagramme des cas d'utilisation</i>	13
1.1 <i>Le recensement des acteurs</i>	13
1.2 <i>Le recensement des cas d'utilisations</i>	14
2. <i>La maquette ou les différentes IHM</i>	20

2.1	<i>L'interface « Accueil »</i>	20
2.2	<i>L'interface « Accueil Utilisateur »</i>	23
2.3	<i>L'Interface « Forum »</i>	24
2.3.1	<i>Ajout d'une « catégorie »</i>	25
2.3.2	<i>Ajout d'un « topic »</i>	26
2.3.3	<i>Ajout d'un « post »</i>	26
3.	<i>Spécification détaillée des exigences</i>	27
3.1	<i>Le scénario « d'Inscription »</i>	27
3.2	<i>Le scénario « Authentification »</i>	28
4.	<i>Le diagramme de Navigation</i>	31
5.	<i>Le diagramme de séquence détaillé</i>	33
5.1	<i>« L'inscription »</i>	33
5.2	<i>« L'Authentification »</i>	34
6.	<i>Le diagramme de Classe</i>	36
7.	<i>Le diagramme de déploiement</i>	38
III.	<i>CONCEPTION DE LA BASE DE DONNEES</i>	39
1.	<i>La modélisation conceptuelle des données</i>	39
2.	<i>La modélisation physique des données</i>	41
IV.	<i>CONCEPTION ET DEVELOPPEMENT : SYMFONY II</i>	42
1.	<i>Introduction à SYMFONY II</i>	42
1.1	<i>L'architecture des fichiers</i>	43

1.2 L'architecture conceptuelle -----	44
1.2.1 L'architecture MVC-----	44
1.2.2 Le parcours d'une requête http -----	45
1.3 Les Bundles -----	46
2. Les bases de SYMFONY II -----	47
2.1 Le routeur-----	47
2.1.1 Son fonctionnement -----	48
2.2 Le contrôleur -----	50
2.3 La vue : le moteur de Template TWIG -----	50
3. La Base de Données avec DOCTRINE II -----	52
3.1 La couche métier ou Model : Les Entités -----	52
3.2 La gestion des Entités -----	55
3.3 Les relations entre les Entités -----	56
Conclusion -----	59
Annexe -----	60
Bibliographie et Webographie -----	64
Tableau des illustrations -----	65

# I. INTRODUCTION

## 1. Présentation des acteurs du PROJET

### 1.1 Maitre d'ouvrage : Le client

C'est durant ma formation de concepteur développeur informatique au sein du centre M2I Formation, que j'ai eu le plaisir de rencontrer et faire la connaissance de Mr GASSAMA, ancien stagiaire ayant débuté la même formation quelques semaines auparavant. Nous avons pu nous entraider mutuellement à la réussite de notre formation. Lui étant en stage et moi à la recherche d'un stage, il me présenta et proposa comme projet professionnel la réalisation ou plus exactement la refonte du site WEB de son Association intitulée **AIKECS**, pour *Association Internationale Karamokhoba pour l'Education, la Culture et la Solidarité*, autant dire que cela suscitait en moi beaucoup d'intérêt à réaliser cette œuvre qui pourrait accomplir de bonnes choses. Je n'ai donc pas hésité à donner mon accord et me suis mis à la tâche.

Concernant la méthodologie de travail suivie et compte tenu de son indisponibilité physique, car domicilié en région Centre, nous sommes restés en contact permanent par téléphone et par messagerie, voire par SKYPE, afin de collaborer ensemble à la réalisation de ce projet.

### 1.2. Maitre d'œuvre : Le stagiaire

Je suis actuellement en poste chez ECONOCOM-OSIATIS comme Technicien informatique en CDI, en charge du traitement des demandes de moyens informatiques tant matériels que logiciels et d'assurer également leur maintien en condition opérationnel. J'ai bénéficié d'un financement en congé individuel de formation pour me consacrer à l'obtention du titre de Concepteur-Développeur Informatique au sein du centre de formation M2I FORMATION.

Passionné par le développement logiciel, j'ai entrepris, il y a quelques années, des études au CNAM de Paris, en vue d'obtenir le diplôme d'ingénieur en systèmes d'information. Je suis actuellement titulaire d'une Licence d'Informatique Générale et décidé à poursuivre mes études supérieures en vue d'obtenir le diplôme d'Ingénieur en Systèmes d'Information et afin de pouvoir, je l'espère, exercer le métier qui me tient à cœur, celui de développeur WEB.



## 2. Le projet en question

### 2.1 Les critiques de l'existant

Mr GASSAMA est l'auteur d'un site WEB associatif référencé à l'adresse suivante : [www.association-karamokhoba.com](http://www.association-karamokhoba.com) qui ne répond plus aux besoins de l'association. Voici les quelques raisons ou motivations qui ont poussés son auteur à vouloir entreprendre une refonte de son site :

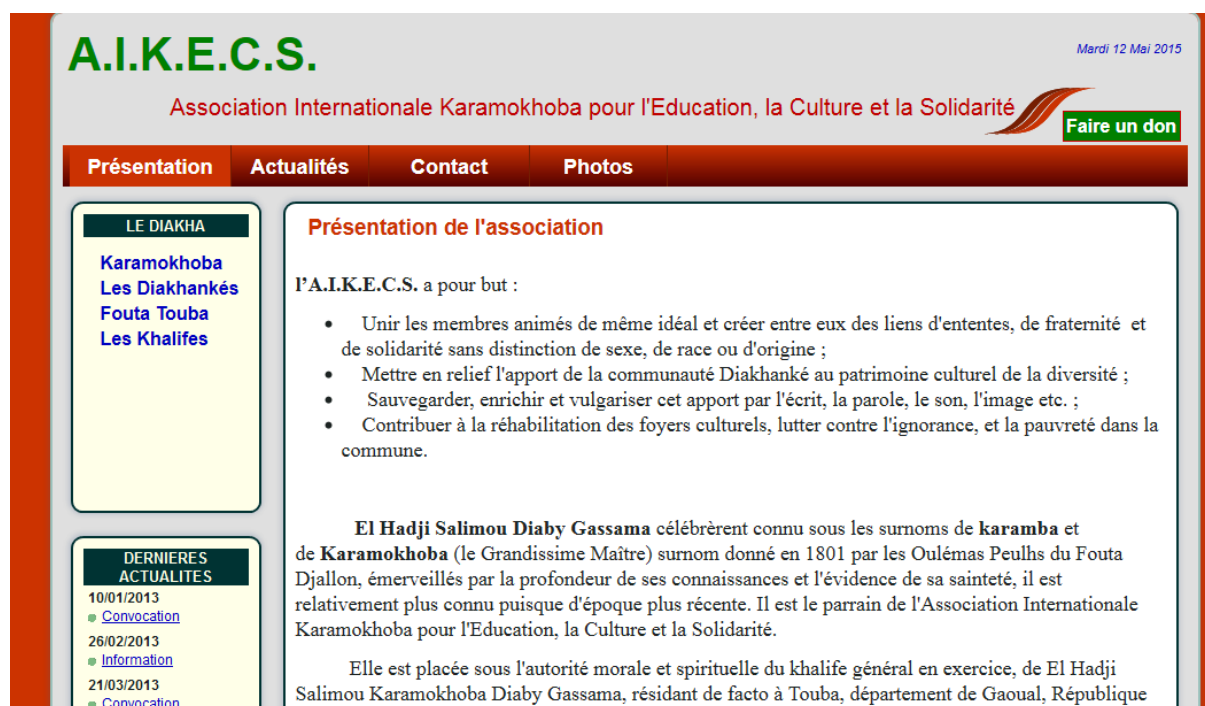


Figure 1 : Page d'accueil du site

- Améliorer la présentation du site. Il est impératif voire indispensable d'apporter une attention particulière au design, à l'ergonomie. En effet, à l'image d'un résumé ou d'une introduction d'une œuvre, une interface graphique soignée est un atout primordial afin d'attirer les visiteurs voire les convertir en adhérents ; d'où le deuxième axe d'amélioration.
- Permettre à un visiteur d'adhérer à l'association. En plus de lui apporter de l'information, consultable en libre accès, le visiteur doit pouvoir remplir un formulaire

d'inscription voire de contacter l'association pour obtenir de plus amples informations. Le site ne possédant pas ces fonctionnalités, nous les implémenteront.

- Permettre à toute personne de consulter le site aussi bien à partir d'un pc que d'un smartphone, ou tablette d'où la nécessité d'un design responsive, qui s'adapte à tous formats.
- Garantir facilement la maintenabilité et assurer l'évolutivité du site en intégrant les technologies modernes. Ce site développé en WINDEV, il le sera ultérieurement en PHP à l'aide du puissant Framework **SYMFONY 2**, en plus d'intégrer HTML5, CSS3 et JQUERY.

## 2.2 Déroulement du projet

*ECHEANCIER PREVISIONNEL DU 20 MARS 2015 AU 18 MAI 2015*

---

<sup>35</sup><sub>17</sub> Elaboration du cahier des charges fonctionnel [2 jours]

<sup>35</sup><sub>17</sub> Analyse et conception en UML [10 jours]

<sup>35</sup><sub>17</sub> Conception de la base de données [2 jours]

<sup>35</sup><sub>17</sub> Conception et développement sous SYMFONY2 [20 jours]

<sup>35</sup><sub>17</sub> Rédaction du rapport de stage [5 jours]

Nous sommes restés, Mr GASSAMA et moi-même en contact téléphonique et par SKYPE tout au long du projet et voir plus, et nous avons échangé par mail lorsque nous avons besoin d'informations et dans le même temps pour étudier et valider l'état d'avancement du projet.

Beaucoup de temps ont été consacrés à l'étude de SYMFONY2.

Après avoir abordés ces quelques points de discussion, nous avons pu rédiger un cahier des charges fonctionnel, qui est le suivant :

## 2.3 Le cahier des charges fonctionnel

Le site proposera 2 espaces d'échanges et d'informations : un espace public, ouvert à tous et un espace privé, exclusivement réservé aux adhérents de l'association.

### 2.3.1 L'Espace public

Tout visiteur aura la possibilité de :

- S'informer sur l'Association :
  - Le visiteur pourra se tenir informé des actualités de l'association, des activités ...
  - Il pourra accéder à une galerie de supports audio vidéo.

- De contacter les membres de l'association, via un formulaire de contact.

Le visiteur fournira à cette occasion son nom, prénom et adresse de messagerie. Ces informations identiques seront à fournir lorsqu'il souhaitera s'abonner aux newsletters, ou s'inscrire.

- De devenir adhérent à part entière, via un formulaire d'adhésion, et ainsi collaborer et participer à la vie associative: En plus des informations mentionnées ci-dessus, il fournira son Nom, Prénom et date de naissance. Un mail contenant un lien d'activation fera suite à l'envoi du formulaire d'inscription.
- Faire un don : le visiteur sera libre de fixer le montant qu'il souhaite mettre à disposition de l'association, qu'elle mettra à profit pour financer des projets socio-culturels. Le paiement pourra être effectué par virement bancaire ou par chèque.

### 2.3.2 L'Espace privé

Le site mettra à leur disposition des services permettant aux membres de l'Association de communiquer et collaborer mutuellement :

- L'adhérent aura la possibilité de **consulter** des événements inscrits dans un agenda. Les événements seront caractérisés par un titre, une date de parution, de début de l'évènement ainsi qu'un contenu.

Seuls les adhérents administrateurs principaux et secondaires auront l'autorisation de **créer** ou **supprimer** les évènements.

- L'adhérent disposera d'une messagerie interne. Il pourra communiquer en interne avec d'autres adhérents. Le message sera caractérisé par un titre (l'objet), une date et un contenu.
- L'adhérent pourra réagir aux publications de l'association, par des commentaires.
- L'adhérent pourra mettre à jour les informations relatives à son profil.

Seul l'administrateur principal aura l'autorisation de gérer les profils utilisateurs :

- Supprimer ou résilier l'adhésion d'un adhérent qui en ferait la demande, ou pas.
- Lire les informations d'un compte sans pouvoir les modifier.
- Les adhérents pourront disposer d'un forum de discussion. Compte-tenu de l'importance de son utilité au sein d'une communauté d'adhérents, on apportera à cette application WEB, un intérêt tout particulier, dont voici le mini-cahier des charges.

### **Cahier des charges fonctionnel du FORUM.**

Un forum de discussion est une application permettant à chacun de participer à des sujets (ou topics) variés.

Les droits (ou pouvoirs) seront définis comme suit :

- Un adhérent peut créer un topic, et participer (rédiger un post) dans n'importe quel topic du forum. Il peut également supprimer **ses posts**.
- Un modérateur ou l'administrateur secondaire, est un utilisateur et a donc les mêmes pouvoirs. Mais il a aussi la charge d'une ou plusieurs catégories.

Il peut supprimer des post ou des topics entiers. A cet effet, il a donc à disposition des liens, présents sur chaque topic et post de ses catégories permettant ces suppressions.

- Un administrateur principal a tous les droits cités précédemment. Il peut en plus créer des catégories, et les gère toutes. Il peut donc créer ou supprimer ce qu'il souhaite.

L'administrateur pourra aussi gérer les pouvoirs des modérateurs ; c'est-à-dire désigner quel modérateur s'occupe de quelle(s) catégorie(s).

#### Hiérarchie des types de contenus :

- Une catégorie a un titre, une description, est gérée par un modérateur ou administrateur secondaire, et possède des topics.
- Un topic a un titre, un auteur, et contient des post.
- Un post a un auteur, une date et une heure de rédaction, et un contenu.

Après avoir rédigé et analysé le cahier des charges fonctionnel de notre futur site WEB, correspondant aux besoins utilisateurs, nous abordons la deuxième partie de notre projet à savoir, la modélisation textuelle et graphique de notre futur système, en proposant les différents diagrammes mises à notre disposition par UML : Nous commencerons par établir le diagramme des cas d'utilisations, pour ensuite parvenir aux diagrammes de séquence détaillés et de classes, qui nous permettront d'aboutir à la représentation d'un diagramme d'interaction. En parallèle nous présenterons une maquette IHM, qui permettra d'établir un diagramme de navigation, afin que tout cet ensemble de représentations puissent nous amener, à la dernière partie de notre étude, à savoir la conception et le développement sous SYMFONY 2, avant de conclure.

## II. ANALYSE ET CONCEPTION : UML

### 1. Le Diagramme des cas d'utilisation

Après avoir rédigé le cahier des charges fonctionnel, conformément aux exigences et aux contraintes du client, nous abordons la deuxième étape de notre étude d'analyse, à savoir modéliser les grandes fonctionnalités de notre système, par un diagramme UML de cas d'utilisation, un des diagrammes fonctionnels.

#### 1.1 Le recensement des acteurs

L'étude du cahier des charges conduit à identifier rapidement 4 acteurs principaux, qui représentent un ensemble cohérent de rôles joués vis-à-vis du système et qui interagissent directement avec le système étudié. On a donc<sup>2</sup> 4 profils différents ou rôles :

<sup>35</sup><sub>17</sub> Visiteur (non-adhérent)

<sup>35</sup><sub>17</sub> Adhérent

<sup>35</sup><sub>17</sub> Administrateur secondaire

<sup>35</sup><sub>17</sub> Administrateur principal

Ces rôles seront gérés en interne grâce à des constantes sous SYMFONY 2 : on distinguera les rôles suivants : `ROLE_USER`, `ROLE_ADMIN_SECONDAIRE`, `ROLE_ADMIN_PRINCIPAL` respectivement pour un adhérent sans aucun droit administrateur, un administrateur ayant certains droits et un administrateur ayant tous les droits.

Le visiteur pouvant être considéré aussi bien comme un adhérent, UML nous permet de représenter cette relation par une relation de généralisation/spécialisation, et ceci graphiquement : nous le verrons plus loin.

Le même raisonnement est à appliquer à un adhérent pouvant avoir des droits administrateurs plus ou moins avancés.

On peut également mentionner au second plan, comme acteurs secondaires l'administrateur du site et l'opérateur de maintenance, chargés tout simplement d'assurer le maintien en condition opérationnel.

Nous avons répertorié simplement les acteurs textuellement, on pourrait toutefois réaliser un premier diagramme : le diagramme de contexte statique. Ce qui permet de spécifier le nombre d'instances d'acteurs connectés au système à un instant donné.

Le site est un système fondamentalement multi-utilisateur : à tout instant, il y a plusieurs instances de chaque acteur connectés au système d'où la représentation suivante :

En cas de maintenance, aucun utilisateur n'est connecté au site.

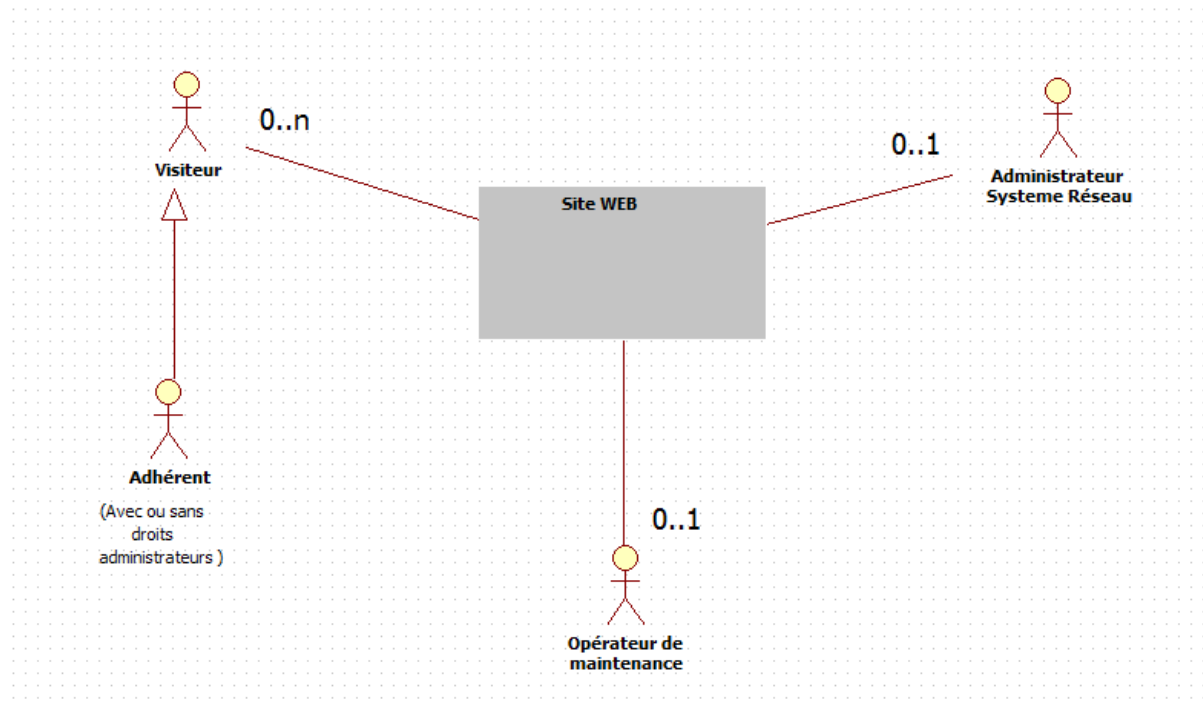


Figure 2: Diagramme de contexte statique

## 1.2 Le recensement des cas d'utilisations (ou use case)

L'ensemble des cas d'utilisation doit couvrir de manière exhaustive tous les besoins fonctionnels du système.

L'élaboration d'un diagramme de cas d'utilisation permet de représenter, avec la plus clarté, tous les besoins utilisateurs en recentrant le système sur les besoins fonctionnels.

Il décrit les grandes fonctions du système seulement du point de vue des acteurs.

L'existence de 2 espaces publics et privés nous impose de considérer le système en 2 sous-systèmes.

Pour plus de clarté, nous procéderons à un regroupement des différents cas relevés dans des packages. Le regroupement se fera par acteur :

<sup>35</sup><sub>17</sub> Visiteur non-adhérent

1. Faire un don
2. Consulter les informations relatives à l'Association
3. Remplir un formulaire de contact
4. Remplir un formulaire d'adhésion : à l'association
5. S'inscrire/Désinscrire aux Newsletters

Une précision importante me semble s'imposer, concernant le cas 'Faire un don'. Il se déclinera en 2 autres cas à savoir, 'Effectuer le paiement par virement bancaire' ou 'Effectuer le paiement par chèque'. UML propose de formaliser cette unité fonctionnelle par une relation de généralisation/ spécialisation comme évoqué plus haut, 'Faire un don' étant un cas d'utilisation généralisé.

En ce qui concerne le cas 2, « Consulter.. », il s'agit de manière générale, de mettre à disposition, et ceci en libre accès, toute information pertinente permettant de présenter l'Association et susceptible d'inciter le visiteur à y adhérer. Les informations seront présentées sous forme d'articles accompagnés de vidéos, d'images, d'actualités...

Présentons tout d'abord un premier diagramme de ce sous-ensemble. Ce diagramme ainsi que les suivants ont été réalisés à l'aide de l'outil **StarUML**.

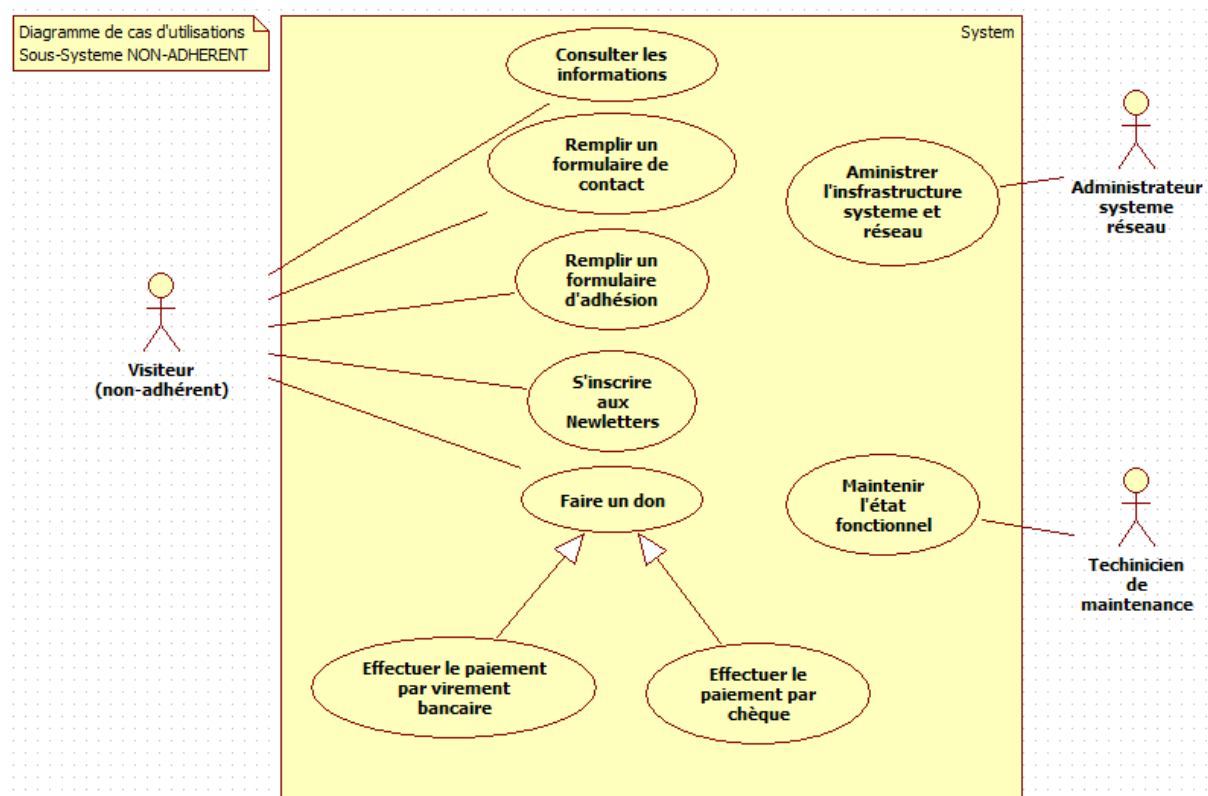


Figure 3: Diagramme de Cas d'Utilisation : NON-ADHERENT



Nous traitons dès à présent du cas, où l'utilisateur est un adhérent de l'association, c'est-à-dire après que l'utilisateur se soit **authentifié**. Aux cas d'utilisations énumérés ci-dessus, s'ajoutent d'autres fonctions métiers du système, mises à sa disposition.

#### <sup>35</sup><sub>17</sub> Adhérent simple Utilisateur

1. Consulter des actualités privées
2. Commenter ces informations
3. Accéder au forum de discussion
4. Gérer son profil

En considérant un administrateur secondaire comme un acteur spécialisé de l'acteur simple adhérent, on récupère l'ensemble des cas d'utilisation, à cela s'ajoutent les cas suivants :

#### <sup>35</sup><sub>17</sub> Adhérent administrateur secondaire

1. Créer des actualités
2. Supprimer des actualités

En appliquant le même raisonnement, on obtient pour l'administrateur principal le schéma suivant : Ce dernier à la charge essentielle de la gestion des comptes adhérents.

1. Créer un compte adhérent
2. Supprimer ou résilier une adhésion

Nous présentons à la figure suivante un diagramme global sans tenir compte de celui concernant le Forum, qui fera l'objet du diagramme suivant :

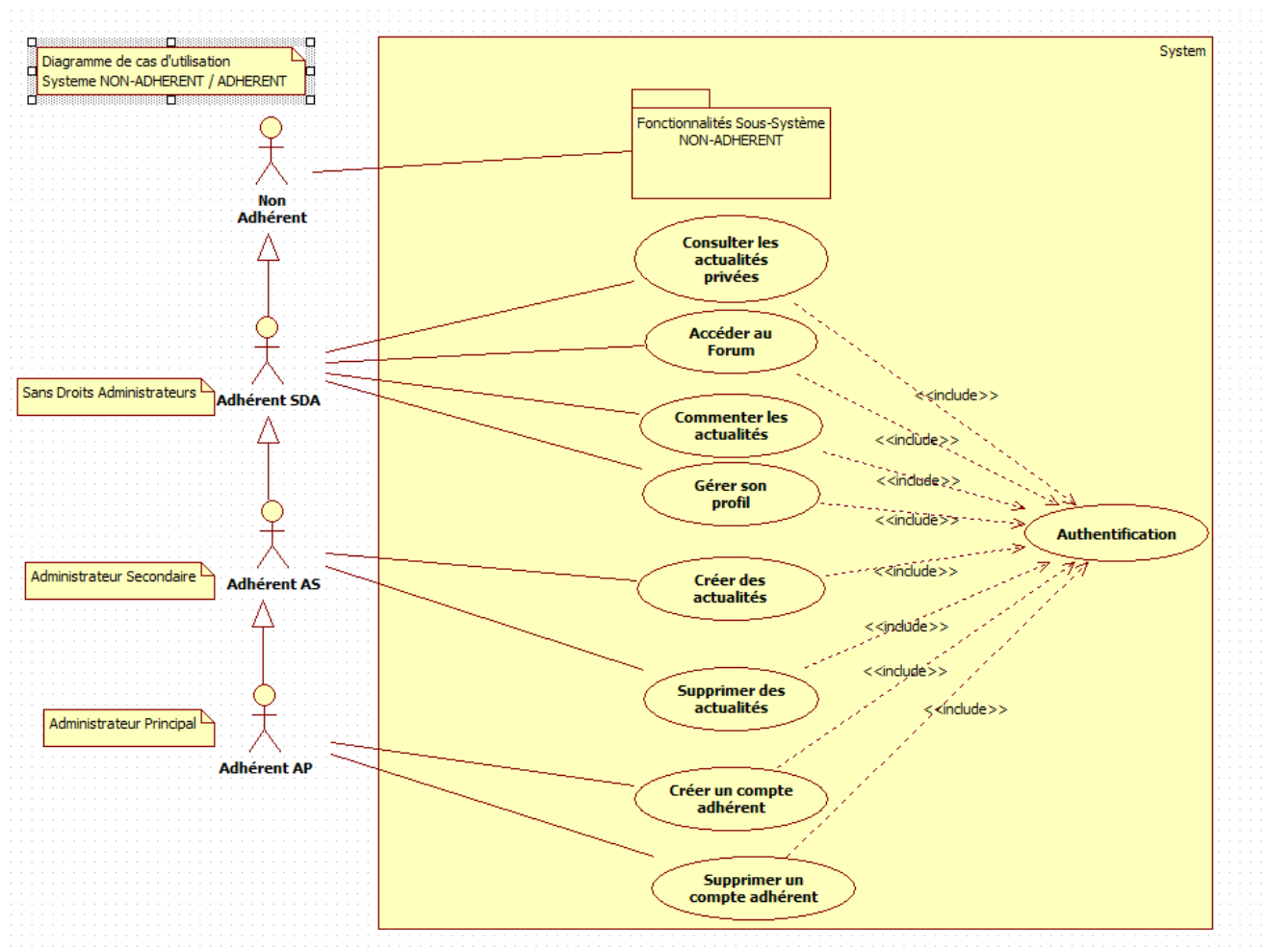


Figure 4: Diagramme de Cas d'Utilisation : ADHERENT

Le livret fonctionnel du Forum fait intervenir d'autres cas d'utilisations que nous allons recenser et représenter graphiquement : on considère le sous-système Forum suivant :

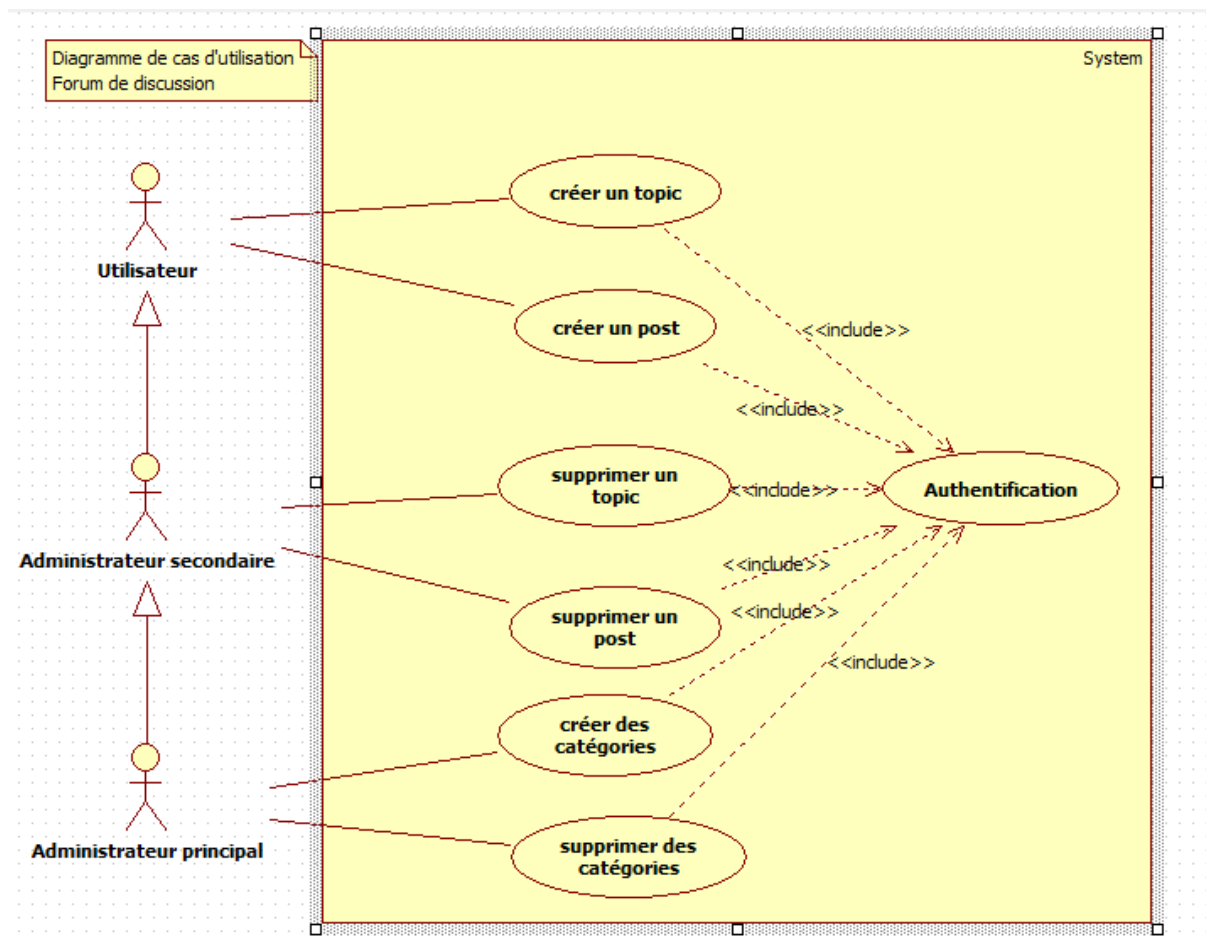


Figure 5: Diagramme de Cas d'Utilisation : FORUM

Pour conclure, un tableau récapitulatif des ayants droits au système :

	VISITEUR	ADHERENT (SD)	ADHERENT(AS)	ADHERENT(AP)
Consulter les informations	X	X	X	X
Remplir un formulaire de contact	X	X	X	X
Remplir un formulaire d'adhésion	X	X	X	X
S'inscrire aux newsletters	X	X	X	X
Faire un don	X	X	X	X
Consulter les actualités privées		X	X	X
Accéder au forum		X	X	X
Gérer son profil		X	X	X
Gestion des actualités (+/-)			X	X

<b>Supprimer un compte</b>				<b>X</b>
<b>Gestion des Posts (+/-)</b>		<b>X</b>	<b>X</b>	<b>X</b>
<b>Gestion des topics (+/-)</b>		<b>(+)</b>	<b>X</b>	<b>X</b>
<b>Gestion des catégories (+/-)</b>				<b>X</b>

(SD) sans Droit d'Administration (ROLE\_USER)

(AS) Administrateur Secondaire (ROLE\_ADMIN\_SECONDAIRE)

(AP) Administrateur Principal (ROLE\_ADMIN\_PRINCIPAL)

(+/-) Créer / Supprimer

## 2. La Maquette ou les différentes IHM

L'étude des fonctions métiers du système réalisée et les cas d'utilisations identifiés, on peut dès à présent commencer à réaliser les interfaces graphiques correspondantes, qui sont les suivantes :

### 2.1 L'interface « Accueil »

La première concerne tous les besoins exprimés par le visiteur (adhérent ou non).



Figure 6: IHM du menu de Navigation



Figure 7: Présentation de l'Association

Accueil > Contact

**ADRESSE POSTALE**

**AIKECS**

Chez Alagie GASSAMA  
34, Boulevard NEY Hall 23  
75018 Paris

**CONTACT TELEPHONIQUE**

GASSAMA Salim: 06 66 88 41 51  
DIABY Fanssoumane: 06 31 83 50 35

Formulaire de contact

**Name**

**Email**

**Subject**

**Body**


**Envoyer**

Figure 8 : Formulaire de contact

Accueil > Don

**Je souhaite soutenir l'association et faire un don:**

**Par chèque bancaire**



**A l'adresse suivante:**

AIKECS  
Chez Alagie GASSAMA  
34, boulevard NEY Hall 23  
75018 Paris

**Par virement bancaire**

Figure 9: Formulaire de don

**AIKECS**  
*Association Internationale Karamokhoba pour l'Education, la Culture et la Solidarité*

Accueil A Propos Galerie Contact Faire un don Authentification Inscription

Accueil > Authentification

Formulaire de connexion

Nom d'utilisateur :

Mot de passe :

☐ Se souvenir de moi

[Mot de passe oublié](#)

© Copyright par Sadek - 2015

[Nous contacter](#) || [Newsletters](#) || [Nous rejoindre sur Facebook](#)

Figure 10: Formulaire d'authentification

Accueil A Propos Galerie Contact Faire un don Authentification Inscription

Accueil > Inscription

Formulaire d'inscription

Nom

Prenom

Date naissance

Nom d'utilisateur :

Adresse e-mail :

Mot de passe :

Vérification :

Figure 11: Formulaire d'inscription

Après une authentification réussie, l'utilisateur sera connecté au système, et disposera de l'affichage suivant :



Figure 12: Affichage du menu après authentification

## 2.2 L'interface « Accueil Utilisateur »

Nous remarquons, la présence de 2 menus en remplacement des 2 menus « Authentification » et « Inscription ». Le menu « Home » est le lien vers un nouveau menu dédié à l'adhérent, donc voici une illustration graphique :



Figure 13: Formulaire de modification de profil



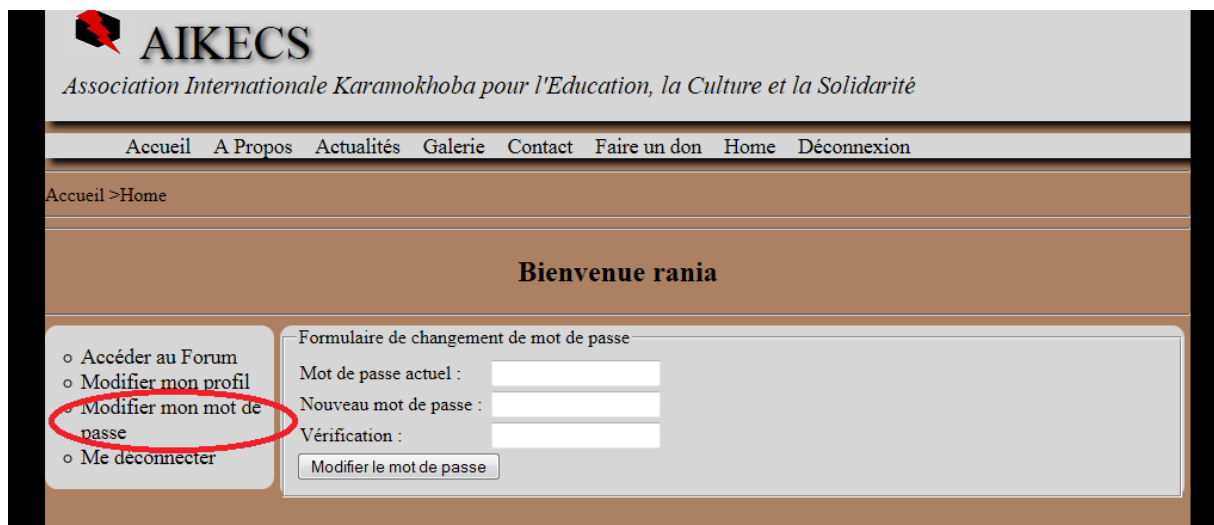


Figure 14: Formulaire de modification de mot de passe

## 2.3 L'interface « Forum »

Nous présentons ici quelques interfaces utilisateurs en prenant bien en compte le statut de l'adhérent (**les différents rôles gérés au niveau de la sécurité sous SYMFONY**). En effet, selon les rôles attribués à chacun, les liens correspondants aux actions réalisées (Ajout, Lecture, Suppression, Mise à jour) sur des entités (catégorie, topic, ...) seront visibles ou masqués.

L'absence ou la présence des liens sont le reflet des cas d'utilisations recensés précédemment. Par ex :

Un adhérent administrateur principal aura l'affichage suivant :

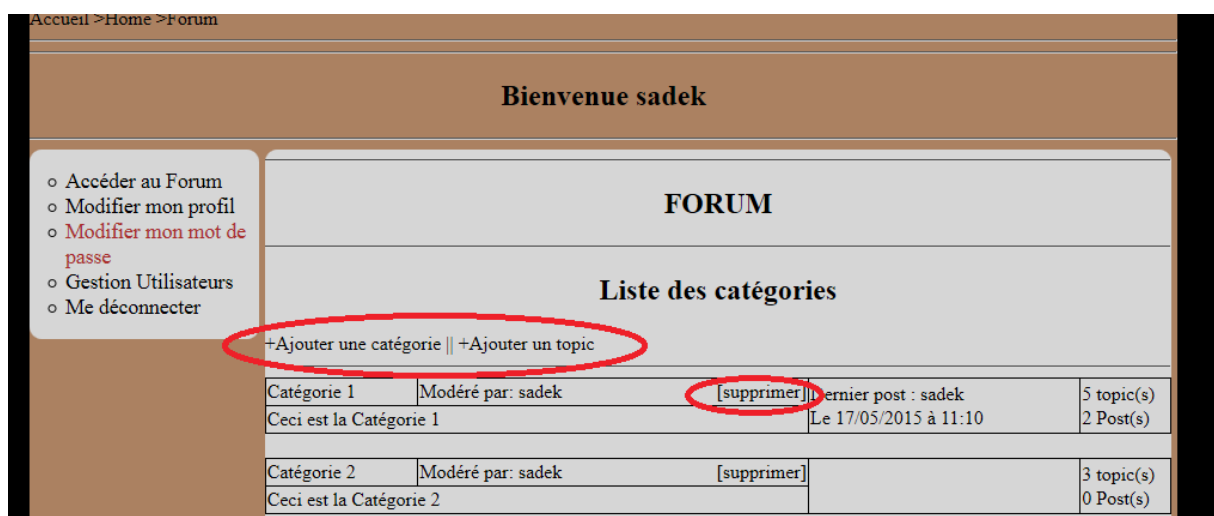


Figure 15: Interface FORUM du point de vue administrateur

Quant à l'adhérent, sans droits d'administration

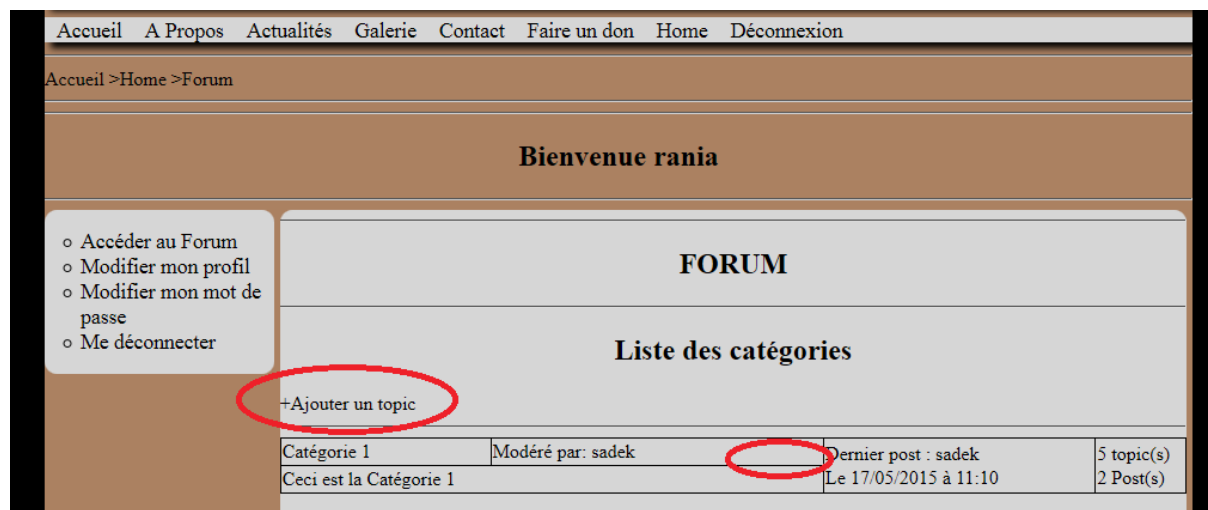


Figure 16: Interface FORUM du point de vue Adhérent sans droits

### 2.3.1 Ajout d'une « Catégorie »

The screenshot shows a web interface for adding a new category. At the top is a navigation bar with links: Accueil, A Propos, Actualités, Galerie, Contact, Faire un don, Home, and Déconnexion. Below this is a breadcrumb trail: Accueil > Forum > Ajout Catégorie. The main heading is "Formulaire d'ajout de Catégorie". The form contains two input fields: "Intitulé de la catégorie" and "Description de la catégorie". Below these fields is a "Valider" button.

Figure 17: Formulaire d'ajout d'une catégorie

### 2.3.2 Ajout d'un « Topic »

Figure 18: Formulaire d'ajout d'un topic

### 2.3.3 Ajout d'un « Post »

Figure 19: Formulaire d'ajout d'un post

## 3. Spécification détaillée des exigences

Après avoir identifié les acteurs et recensé de manière exhaustive l'ensemble des fonctions métiers du système, nous allons maintenant décrire de façon détaillée quelques-uns de ces cas, ceux qui présentent un intérêt particulier. Nous compléterons cette description textuelle par une représentation graphique UML très utile : le diagramme de séquence «système».

### 3.1 Le scénario d' « Inscription »

Avant de pouvoir devenir membre et ainsi accéder aux services dédiés à ce dernier, le visiteur doit s'inscrire.

Pour détailler la dynamique du cas d'utilisation, la procédure la plus évidente consiste à recenser de façon textuelle toutes les interactions entre l'acteur et le système. Le cas d'utilisation doit avoir un début et une fin clairement identifiés. Il faut aussi préciser les variantes possibles tout en essayant d'ordonner séquentiellement les descriptions afin d'améliorer leur lisibilité.

Le scénario nominal est celui qui satisfait les objectifs des acteurs par le chemin le plus direct : en voici une illustration :

## **Procéder à l'inscription**

### **Acteur Principal**

Le visiteur.

### **Objectif**

Etre adhérent et disposer, à l'issue de ce processus, d'un ensemble d'informations de connexion : login et mot de passe.

### **Préconditions**

Le visiteur dispose des informations : telles sont nom , prénom ...

### **Post conditions**

<sup>35</sup><sub>17</sub> Les informations ont été saisies et envoyées

<sup>35</sup><sub>17</sub> Le compte utilisateur a été créé et sauvegardé dans la base de données.

### **Scénario nominal**

1. le visiteur clique sur le lien « inscription » du menu.
2. le système lui affiche un formulaire d'inscription.
3. le visiteur saisie les informations demandées et valide.
4. le système vérifie que les données saisies sont correctes.
5. Après vérification, le système envoie un lien d'activation par mail.
6. L'utilisateur clique sur le lien et active son compte automatiquement.

Après validation des informations :



Après la création de son compte utilisateur et l'activation de ce dernier, l'utilisateur est en possession de ses identifiants de connexion et peut désormais se connecter. Voyons dès à présent le scénario d'authentification.

### 3.2 Le scénario « Authentification »

#### L'Authentification

##### Acteur Principal

L'adhérent

##### Objectif

S'authentifier afin de pouvoir accéder aux services liés

##### Préconditions

L'adhérent dispose de ses informations de connexion

##### Post conditions

### Scénario nominal

1. Le visiteur clique sur le lien « authentification » du menu.
2. Le système lui affiche un formulaire d'authentification.
3. Le visiteur saisie son Nom d'utilisateur et son mot de passe et valide.
4. Le système vérifie que les données saisies sont correctes par rapport à celles de la BD.
5. Après vérification, le système affiche un nouveau menu.

### Scénario alternative

- 4.a Le nom d'utilisateur et/ou le mot de passe sont incorrects. La connexion au système n'a pas pu être établie.
- 4.b L'utilisateur clique sur le lien « oublie son mot de passe » à sa disposition puis est invité à saisir son Nom d'utilisateur ou son e-mail (si Nom oublié).
- 4.c le Système d'authentification retourne à l'utilisateur, par mail, un lien lui permettant de réinitialiser son mot de passe. L'utilisateur en possession de ses nouveaux identifiants, reprend à l'étape 3.

AIKECS  
*Association Internationale Karamokhoba pour l'Education, la Culture et la Solidarité*

Accueil A Propos Actualités Galerie Contact Faire un don Authentification Inscription

Accueil > Authentification

Formulaire de connexion

Nom d'utilisateur : rania

Mot de passe : .....

☐ Se souvenir de moi

Connexion

Mot de passe oublié

© Copyright par xxx - 2015

Nous contacter || Newsletters || Nous rejoindre sur Facebook

L'affichage suivant correspond à une authentification réussie :



Le cas échéant :



On souhaite pouvoir récupérer ses identifiants de connexion. Pour cela on a cliqué sur le lien « Mot de passe oublié ».

**AIKECS**  
Association Internationale Karamokhoba pour l'Education, la Culture et la Solidarité

Accueil A Propos Actualités Galerie Contact Faire un don Authentification Inscription

Accueil > Authentification > Réinitialisation

Réinitialisation de mot de passe

Nom d'utilisateur ou adresse e-mail :

**Réinitialiser**

© Copyright par xxx - 2015

Nous contacter || Newsletters || Nous rejoindre sur Facebook

**AIKECS**  
Association Internationale Karamokhoba pour l'Education, la Culture et la Solidarité

Accueil A Propos Actualités Galerie Contact Faire un don Authentification Inscription

Accueil

Un e-mail a été envoyé à l'adresse ...@gmail.com. Il contient un lien sur lequel il vous faudra cliquer afin de réinitialiser votre mot de passe.

© Copyright par xxx - 2015

Nous contacter || Newsletters || Nous rejoindre sur Facebook

Nous avons intentionnellement décidés de traiter en détail, 2 cas d'utilisation à savoir **l'inscription** et **l'authentification**, et mis de côté les autres que nous étudierons à travers des diagrammes d'activité ou de séquences.

Les cas d'utilisation décrivent les interactions des acteurs avec le site Web que nous voulons spécifier et concevoir. Lors de ces interactions, les acteurs produisent des messages qui affectent le système informatique et appellent généralement une réponse de celui-ci. Nous allons isoler ces messages et les représenter sur des diagrammes de séquence UML

#### 4. Le diagramme de Navigation.

Le Diagramme de Navigation va nous permettre de représenter le cheminement de notre application, entre les différentes interfaces graphiques de l'application. C'est un diagramme dynamique.

Nous l'illustrerons, aux prochains paragraphes, par 2 diagrammes d'activité et séquence.



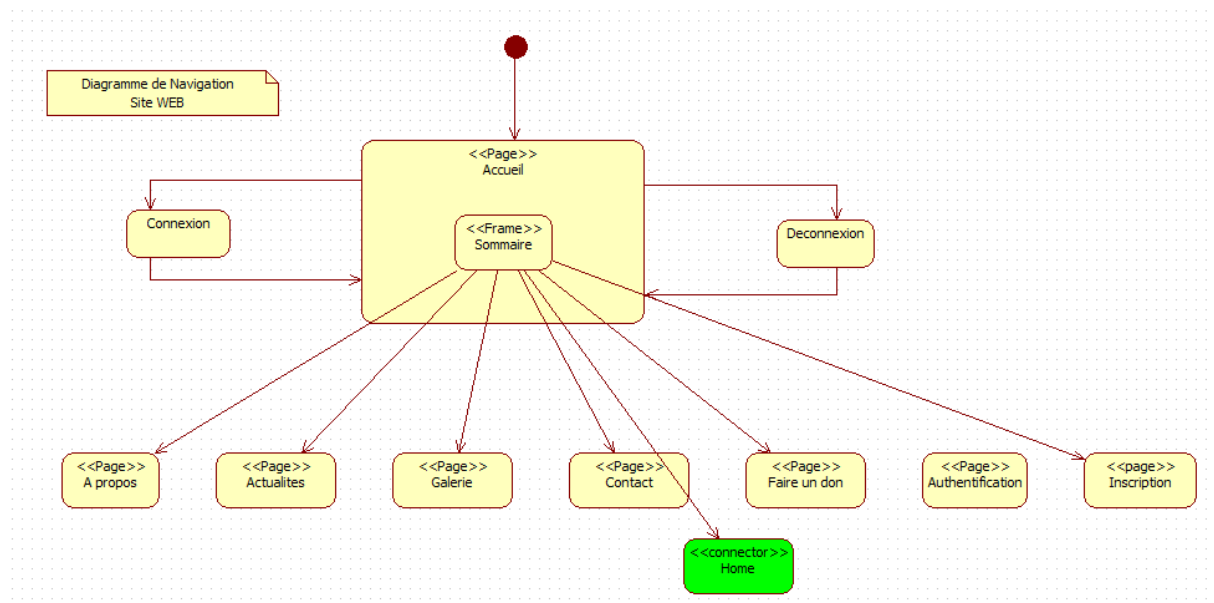


Figure 20: Diagramme de Navigation Front-Office

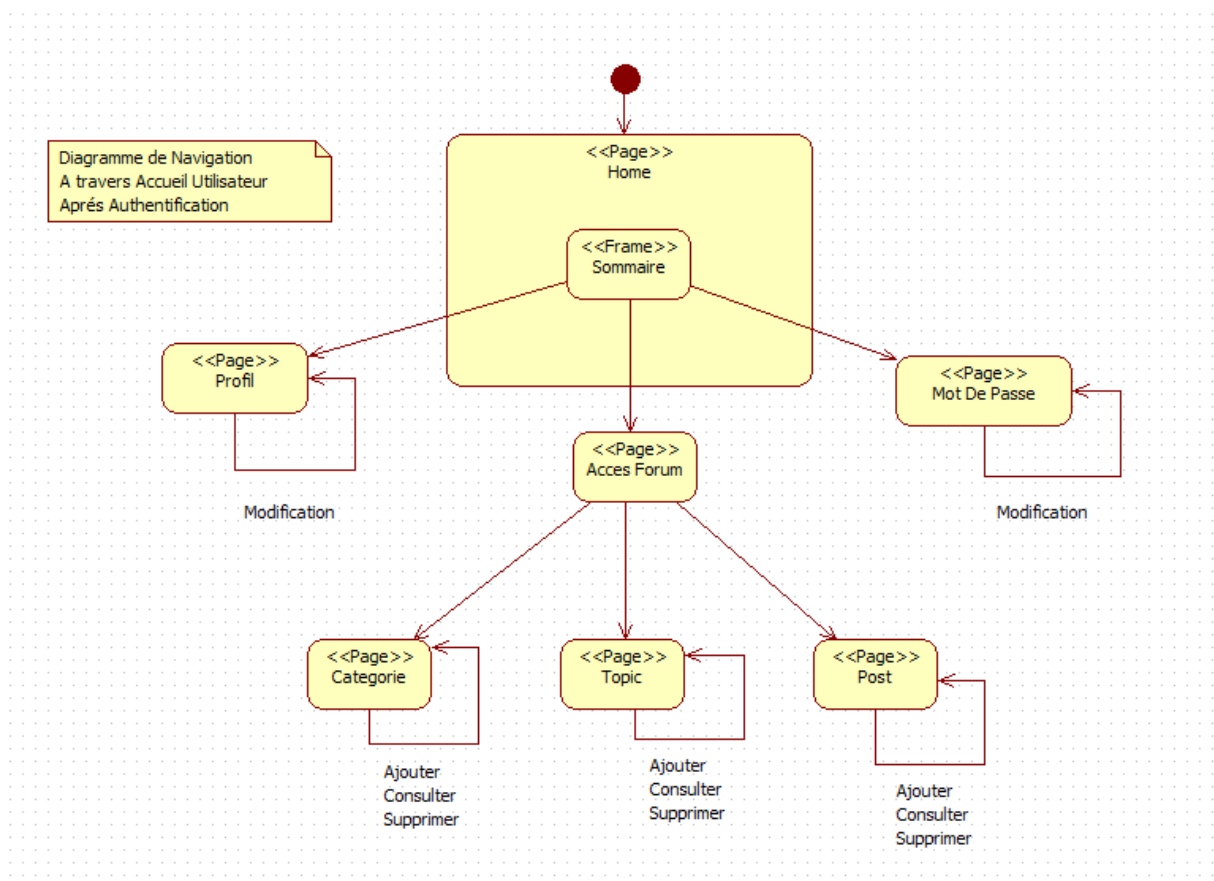


Figure 21: Diagramme de Navigation Back-Office

## 5. Le diagramme de séquence système

Pour documenter les cas d'utilisation, la description textuelle est indispensable, car elle seule permet de communiquer facilement avec les utilisateurs et de s'entendre sur la terminologie métier employée.

Néanmoins le texte nous permet pas de discerner la succession des enchainements, c'est pour la raison que nous compléterons la description textuelle par un ou plusieurs diagrammes dynamiques UML.

Certains cas d'utilisations ne présentent aucun intérêt à être représentés, nous ne les traiterons pas, c'est le cas de consulter la présentation du site, les articles, ...

**Les diagrammes de séquence système sont volontairement simplifiés, nous verrons plus détail leur implémentation interne dans une architecture MVC, lorsque nous aborderons le FRAMEWORK PHP, SYMFONY 2.**

### 5.1 « L'inscription »

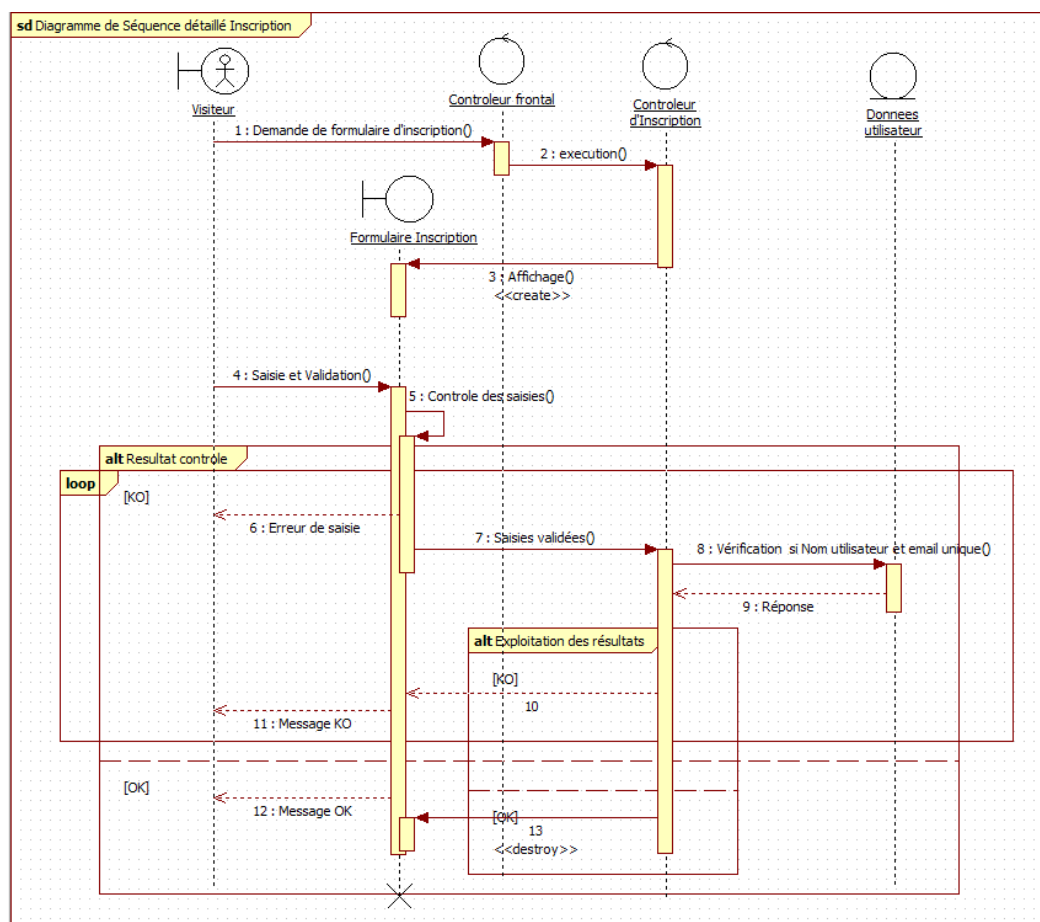


Figure 22: Diagramme de séquence détaillé « Inscription »

Le diagramme d'activité correspondant

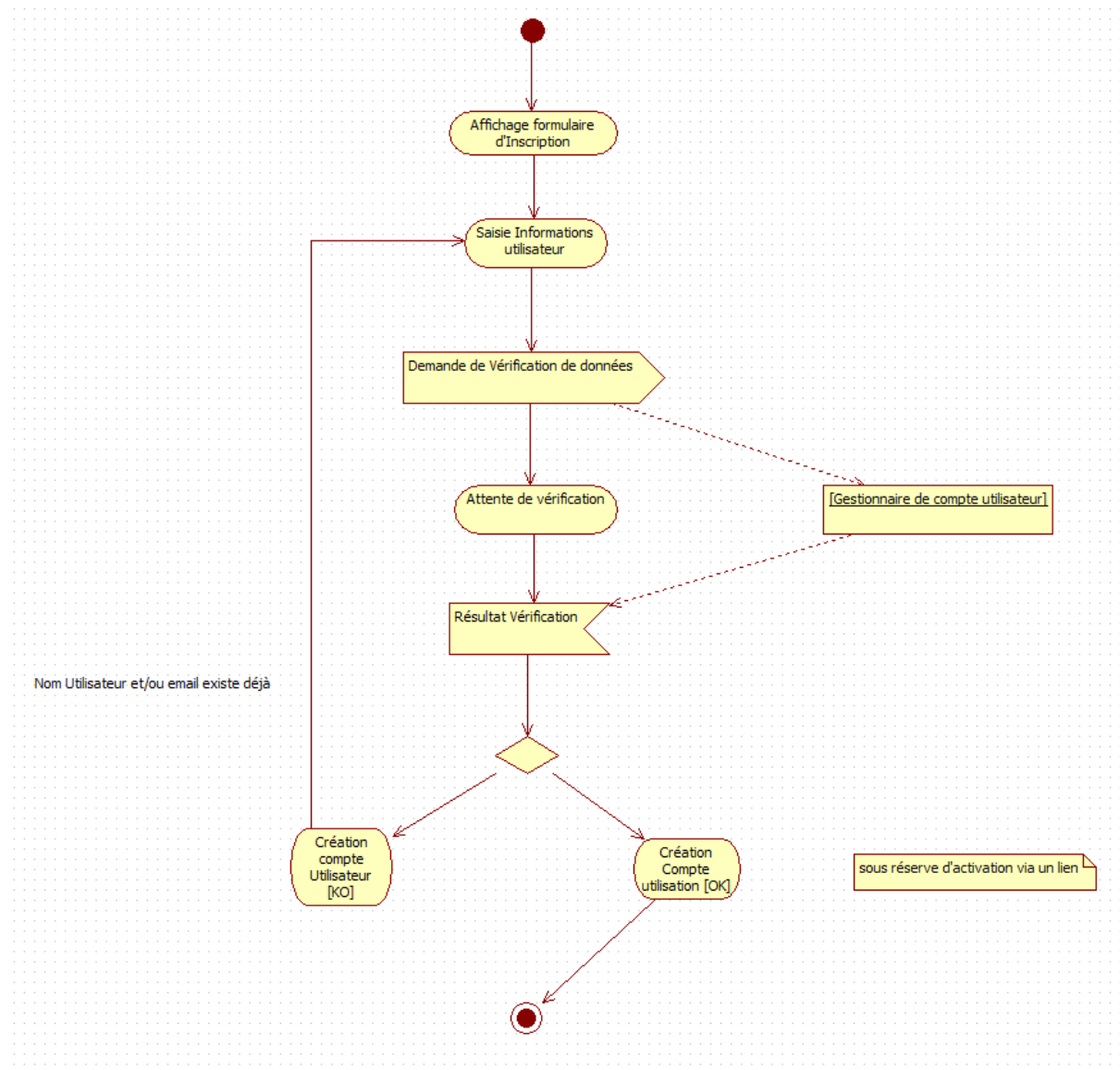


Figure 23: Diagramme d'activité « Inscription »

## 5.2 « L'authentification »

A peu de choses près, le diagramme de séquence correspondant est quasi identique à celui du précédent. Le contrôleur sollicité est différent :

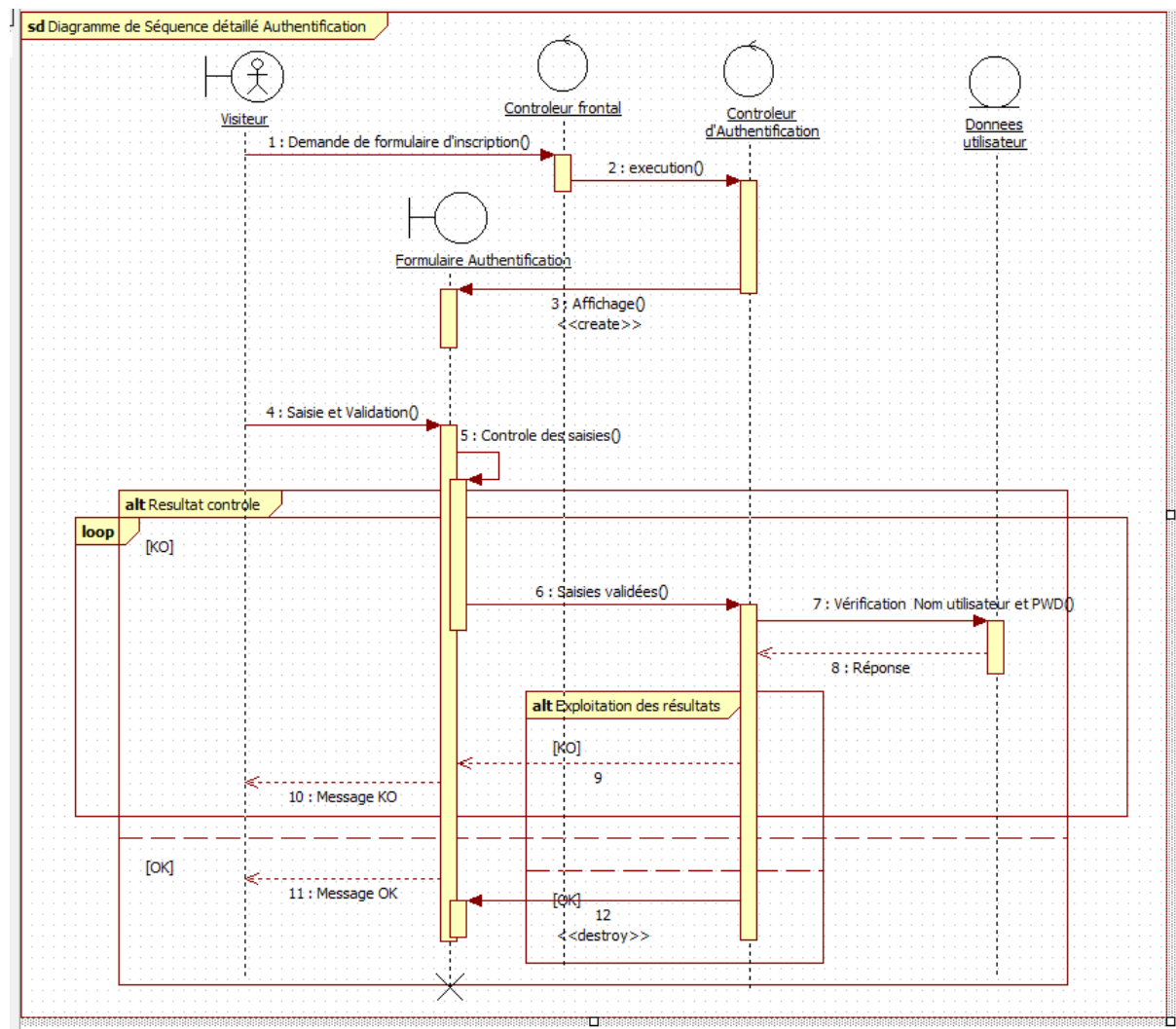


Figure 24: Diagramme de Séquence détaillé « Authentification »

Nous ajoutons au passage le diagramme d'activité correspondant à une procédure d'authentification :

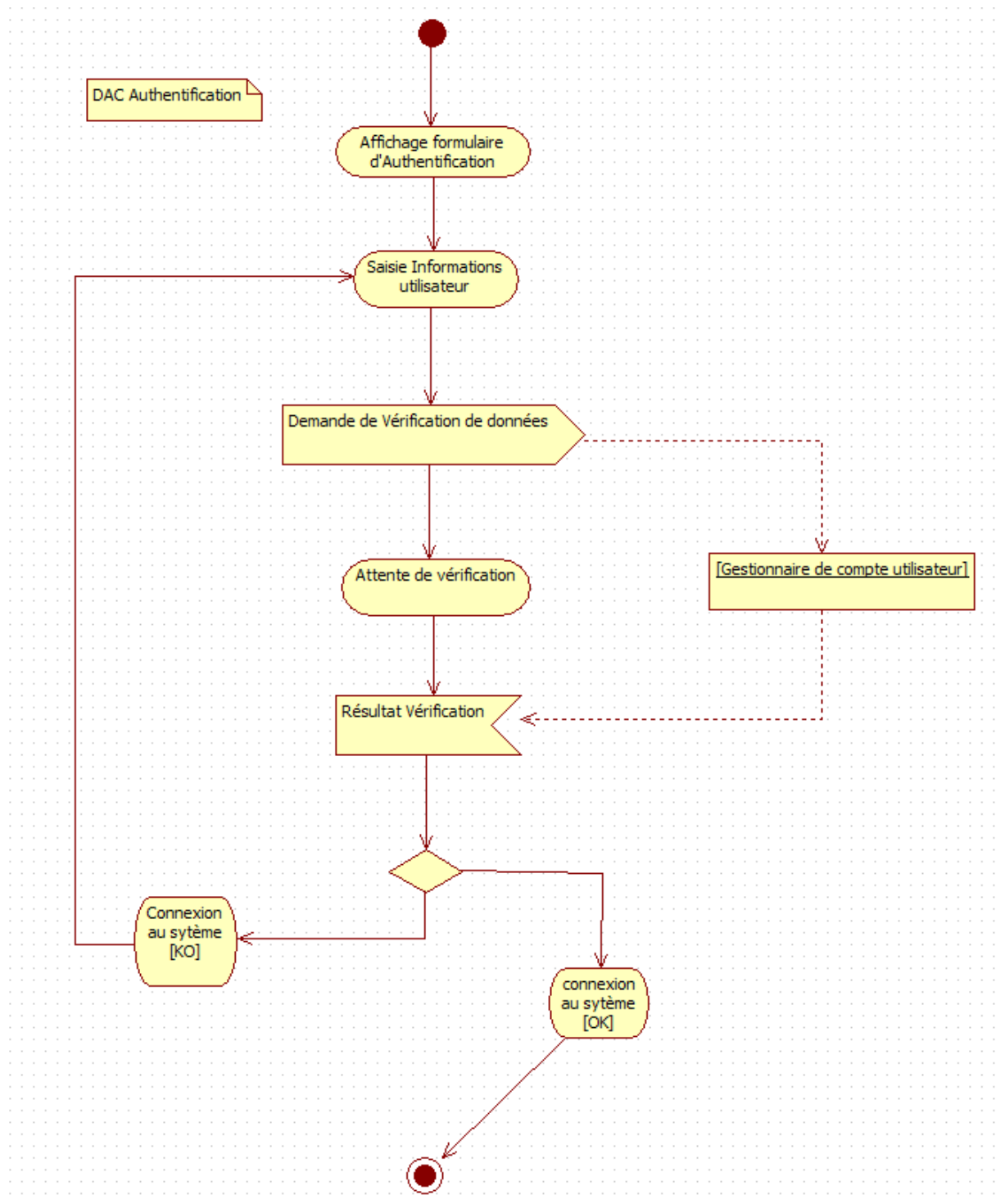


Figure 25: Diagramme d'activité « Authentification »

## 6. Le Diagramme de classe.

Le diagramme de classe est un diagramme statique. Il permet de fournir une représentation abstraite des objets du système étudié.

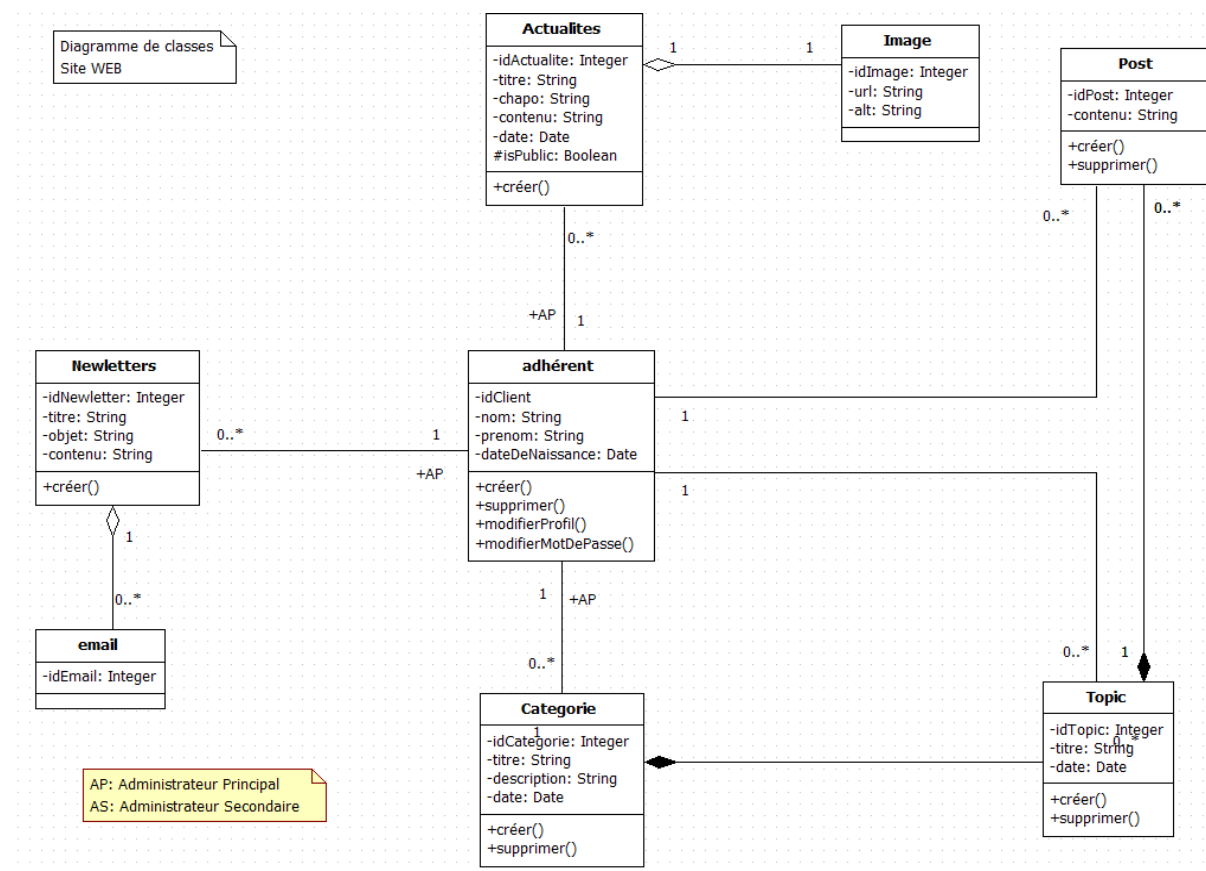


Figure 26: Diagramme de classes du site

## 7. Le Diagramme de déploiement.

Le diagramme de déploiement montre la disposition physique des matériels qui composent le système et la répartition des composants sur ces matériels.

Les ressources matérielles sont représentées sous forme de nœuds.

Les nœuds sont connectés entre eux à l'aide d'une voie communication.

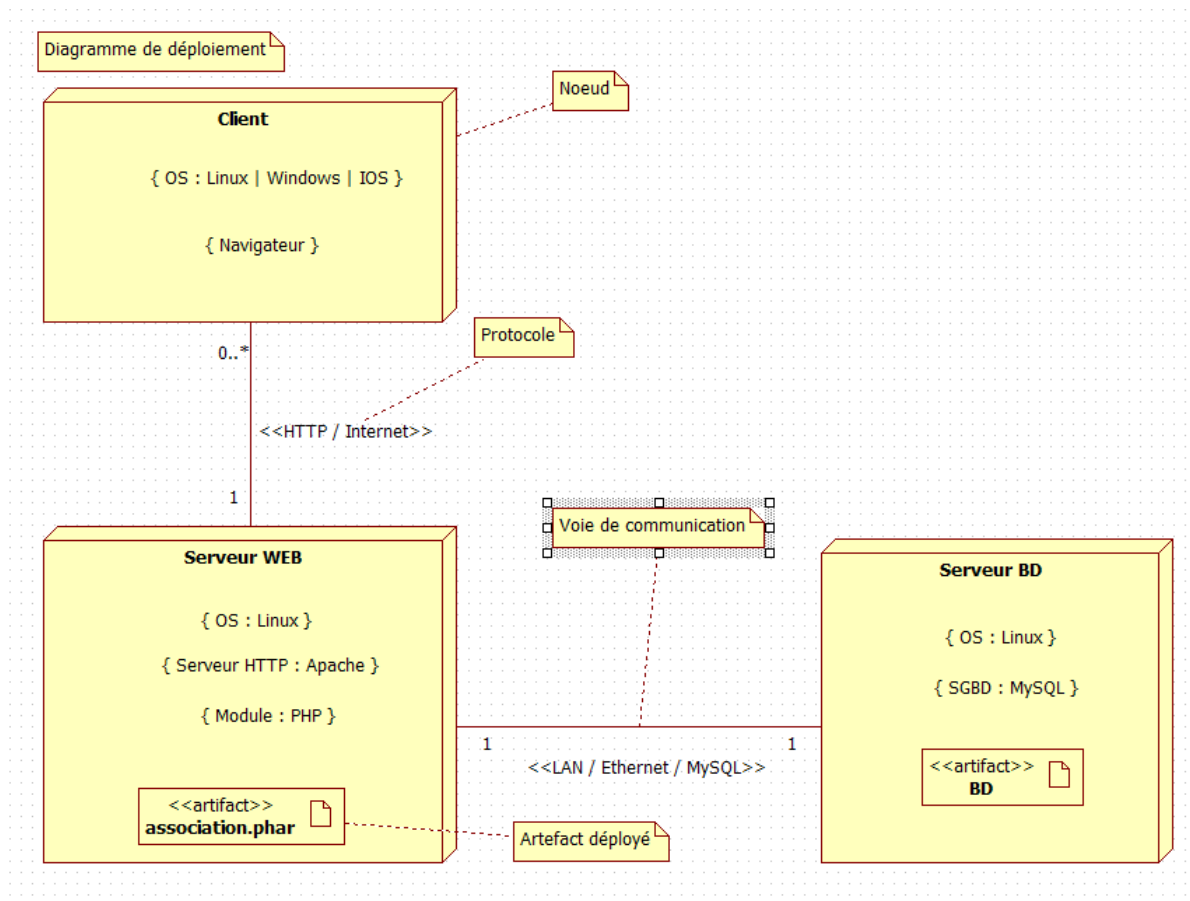


Figure 27 : Diagramme de Déploiement

Ici, nous avons une architecture d'application WEB à 3 niveaux.

Ce diagramme représente le client et son navigateur, le serveur WEB et le serveur de BD avec un SGBDR.

Le client accède aux données de la BD MySQL située sur un serveur Linux via http Apache.

# III CONCEPTION DE LA BASE DE DONNEES

## 1. La modélisation conceptuelle des données

La modélisation est une étape fondamentale de la conception de la BD dans la mesure où, d'une part, on y détermine le contenu de la BD et, d'autre part, on y définit la nature des relations entre les concepts principaux.

Les éléments de base du modèle Entité-Relation ou Entité Association sont les suivants :

<sup>35</sup><sub>17</sub> Les Entités : objet pouvant être identifié distinctement

<sup>35</sup><sub>17</sub> Les attributs : caractéristiques ou propriétés des entités

<sup>35</sup><sub>17</sub> Type de relation : cardinalités ou le nombre de participation d'une entité à une relation

Nous réalisons le recensement des éléments cités ci-dessus à partir du cahier des charges fonctionnel et des différentes IHM, et les présentons sous forme d'un schéma conceptuel réalisé avec JMERISE.



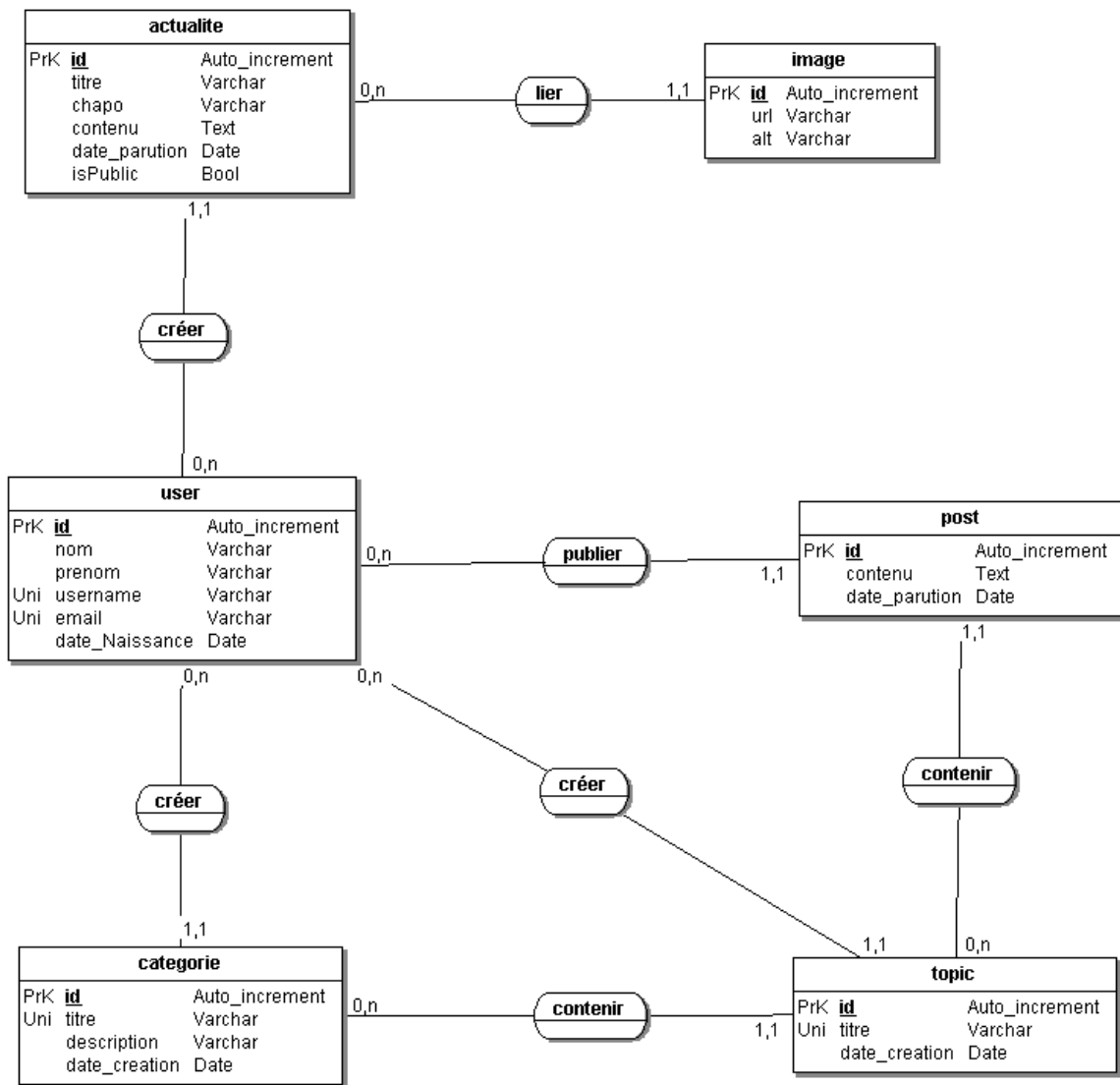


Figure 28 : MCD du site

1 **user** ayant le rôle adéquat peut créer plusieurs actualités, plusieurs catégories, plusieurs topics et plusieurs post. En revanche chacun de ces derniers ne peut être rattaché qu'à un seul user, donc relations de type **1..n**

La Relation [**catégorie – topic**] est une relation de type **1..n** : 1 catégorie peut contenir plusieurs topic, en revanche 1 topic est rattaché à une seule catégorie.

La Relation [**topic – post**] est une relation de type **1..n** : 1 topic peut contenir plusieurs post, en revanche 1 post est rattaché à une seule catégorie.

La Relation [**actualité – image**] est une relation de type **1..n** : 1 actualité peut être lié plusieurs images, en revanche 1 image est rattaché à une seule actualité.

## 2. La modélisation physique des données

A ce stade nous pourrions, générer le modèle physique à partir du modèle précédent. Nous disposerions d'un script de génération de notre future base de données, que nous pourrions importer et le voir s'exécuter sous phpMyAdmin mais en utilisant un FRAMEWORK PHP comme SYMFONY, nous disposons d'un ORM DOCTRINE2, qui se charge d'absolument tout.

En mode console, DOCTRINE2 générera la création de la base ainsi qu'il se chargera de la conversion des Entités en Tables.

Sous SYMFONY2, on crée les Entités puis on met à jour la Base de données, nous le verrons en détail au prochain chapitre,

Voilà ce que nous obtenons, au final :

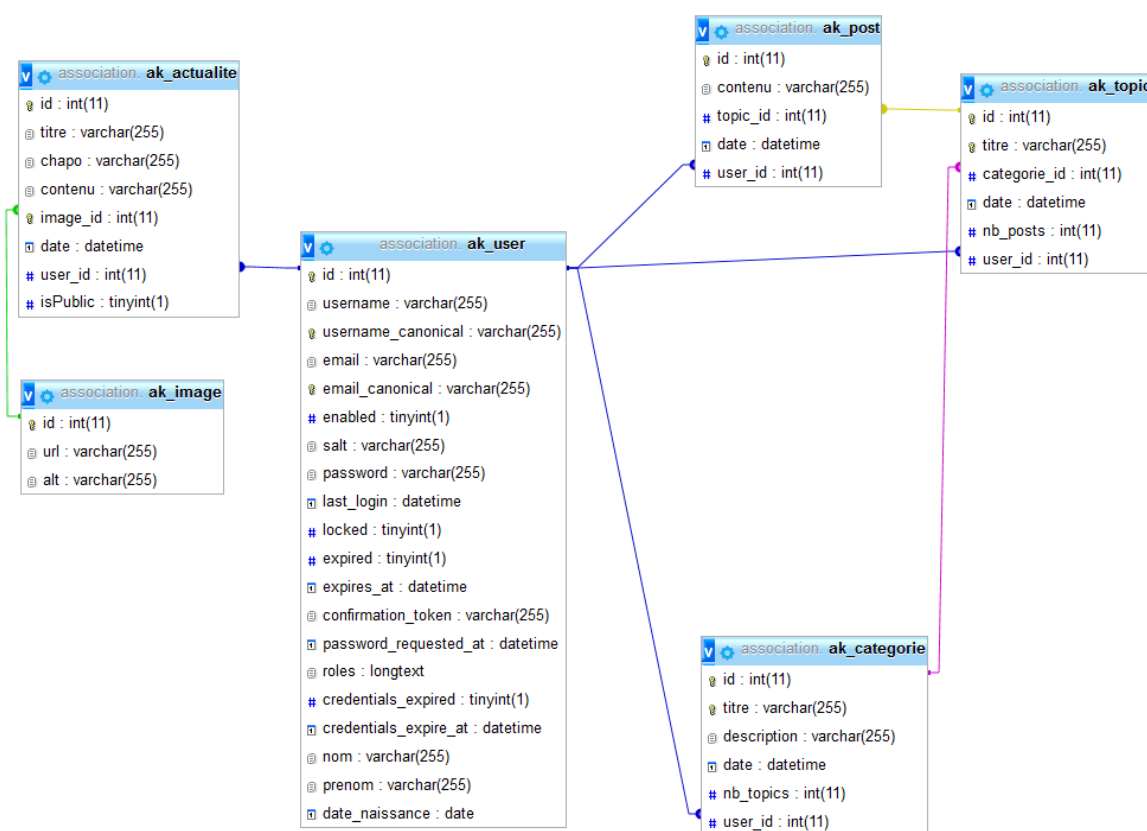


Figure 29 : Schéma de la Base de données

D'autres attributs de l'entité **user**, ont été ajoutés. Cela vient du fait que nous avons utilisé une entité user dans un bundle externe, **FOSUserBundle**, et notre entité user a hérité des attributs de l'entité mère.

# IV CONCEPTION ET DEVELOPPEMENT SOUS SYMFONY 2

Après avoir établi et analysé le cahier des charges fonctionnel, ce qui nous a permis de concevoir les différents diagrammes UML pour modéliser notre application, nous abordons dès à présent la partie Implémentation, ou la conception et le développement en PHP sous SYMFONY 2.

Les outils employés dans le développement sont les suivants :

- L'environnement de développement PHP : XAMPP, distribution Apache contenant entre autres un serveur HTTP Apache, MySQL, PHP
- Editeur de code NOTEPAD ++

## 1. Introduction à SYMFONY 2

Un Framework est un ensemble de composants permettant d'accompagner et de simplifier au maximum le travail du développeur, tout en l'obligeant à adopter les bonnes pratiques du développement de projet.

Les raisons pour lesquelles nous avons portés notre choix sur l'utilisation d'un Framework sont les suivantes :

- Il permet d'augmenter l'efficacité en faisant abstraction des traitements de base
- Il permet d'optimiser la sécurité en nous obligeant d'adopter des bonnes pratiques, comme l'utilisation du design pattern MVC et le concept Objet
- Il embarque tout un ensemble d'outils permettant d'automatiser les tâches qui peuvent souvent fastidieuses

SYMFONY 2 est l'un des meilleurs Framework PHP au monde, développé par SensioLabs, une entreprise française. Il est utilisé par de nombreux site, DAILYMOTION, OPENCLASSROOM...

Il embarque, comme outils :

- L'ORM DOCTRINE 2, pour la gestion de la BD
- Un composant de génération de formulaires, FORMBUILDER
- L'outil COMPOSER pour l'utilisation de BUNDLES, modules logiciels configurables et réutilisables

L'architecture SYMFONY 2 repose sur le concept MVC et sur le paradigme OBJET, pour une meilleure organisation du code en séparant le code PHP du code HTML.

Enfin, SYMFONY 2 est livré de base avec un moteur de Template TWIG pour l'affichage des données.

Après le téléchargement et l'installation de SYMFONY 2, nous commençons par créer notre projet « association » à l'aide du fichier **symfony.phar** placé à la racine de notre serveur local **/htdocs**, en ligne de commande :

**> php symfony new association**

Puis installer les bundles de base :

**C:/xampp/htdocs/association > php app/console assets : install**

### 1.1 L'architecture des fichiers

4 répertoires sont créés, dont 3 présentent un intérêt majeur dans le cadre de notre étude :

- **/app** : il contiendra tout la configuration de notre application en matière de sécurité, de services, et de routage excepté les codes sources de l'application.

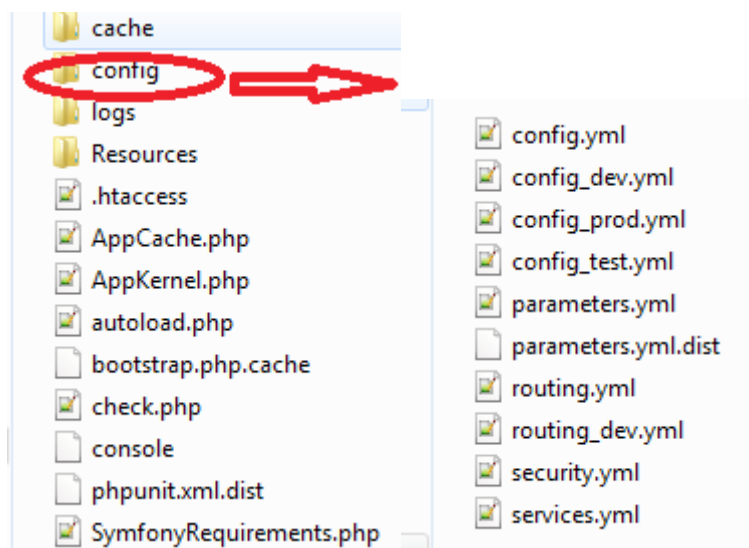


Figure 30 : Contenu du dossier **/app** et **/app/config**

- **/src** : il contiendra tout le code source. Nous organiserons notre code en BUNDLES, les composants ou fonctionnalités de notre application.

- /web : il contiendra tous les éléments destinés à nos visiteurs : images, fichiers CSS et JavaScript...et surtout le contrôleur frontal **app\_dev.php** (dans un environnement de développement), le point d'accès à notre application.

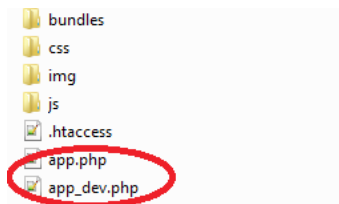


Figure 31: Contenu du répertoire /web

Le contrôleur frontal est chargé d'analyser la requête http (request) du client (le navigateur) afin de nous retourner une réponse (response). Le détail de ce processus est donné au prochain paragraphe.

## 1.2 L'architecture conceptuelle

Après avoir vu l'organisation des fichiers, voyons dès à présent comment s'organise l'exécution du code au sein de SYMFONY 2.

### 1.2.1 L'Architecture MVC

SYMFONY 2 respecte cette architecture MVC, pour **Modèle/Vue/Contrôleur** : découpage logiciel très répandu pour développer des sites internet car il sépare les couches selon leur logique :

- Le **Contrôleur** (Controller) : Son rôle est de générer la réponse à la requête HTTP demandée par notre visiteur. Il est la couche qui se charge d'analyser et de traiter la requête de l'utilisateur. Le contrôleur contient la logique de notre site Internet et va se contenter « d'utiliser » les autres composants : les modèles et les vues. Concrètement, un contrôleur va récupérer, par exemple, les informations sur l'utilisateur courant qui souhaite s'inscrire et demander la page du formulaire d'inscription.
- Le **Modèle** (Model) : Son rôle est de gérer les données et le contenu. Il permet au contrôleur de manipuler les données sans savoir comment elles sont stockées, gérées. Le modèle est une couche d'abstraction.
- La **Vue** (View) : son rôle est d'afficher les pages, il est chargé du rendu visuel. Dans l'exemple de l'affichage de notre Formulaire d'inscription, ce n'est pas le contrôleur qui affiche le formulaire, en somme ne contient pas le code d'affichage, mais sollicite la vue sans savoir ce qu'elle contient, il lui peut tout de même lui transmettre des données, récupérés par le modèle, à afficher. Nous le verrons en détail prochainement.

### 1.2.2 Le parcours d'une requête http sous SYMFONY 2

Voici une illustration graphique et textuelle du parcours d'une requête à travers l'exemple d'une demande d'Inscription.

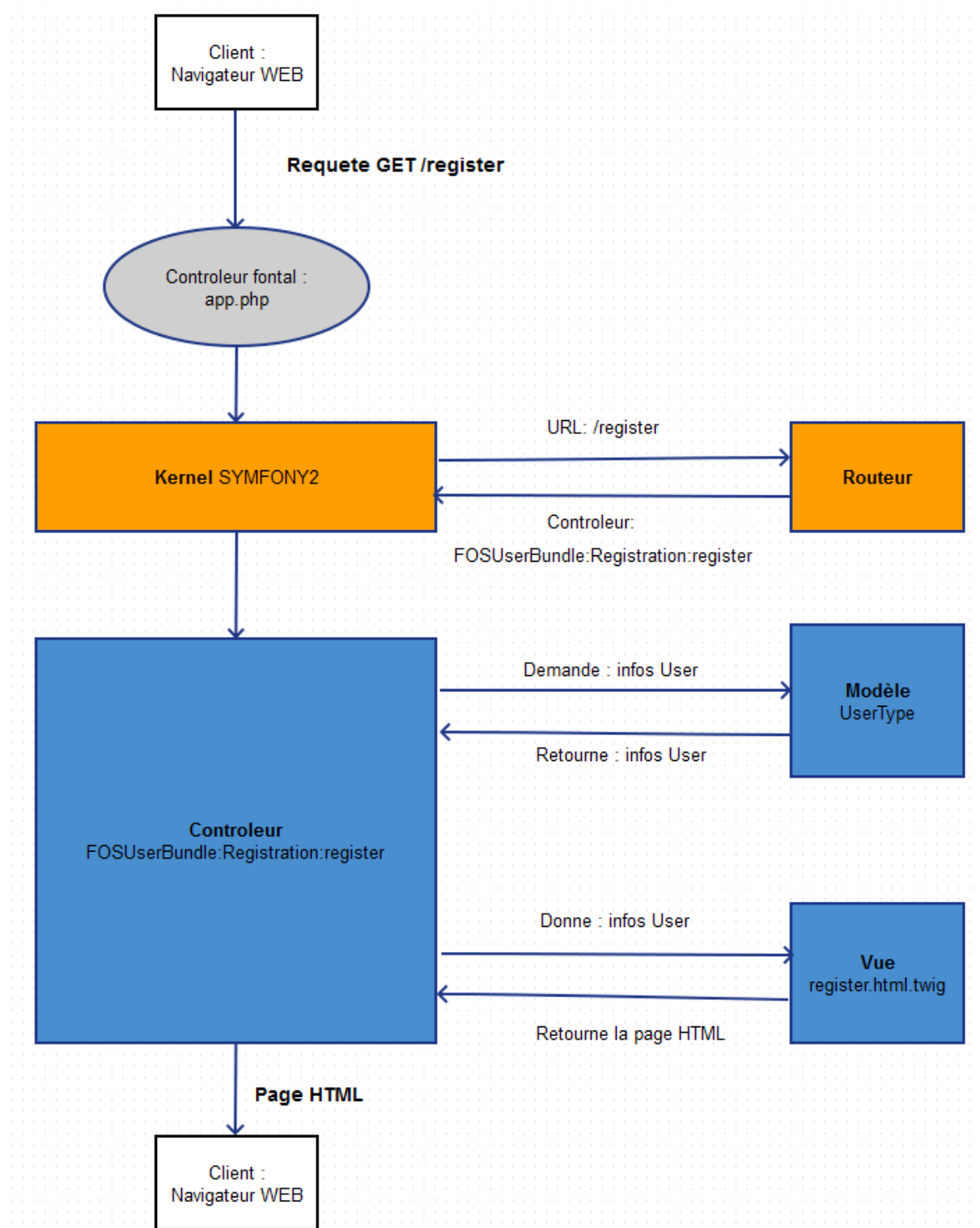


Figure 32: Parcours d'une requête Ex. « Authentification »

### Description textuelle :

- 1 Le Visiteur, en cliquant sur le lien Inscription, demande la ressource /register.
- 2 Le contrôleur frontal app.php reçoit la requête, charge le KERNEL et la lui transmet.

3 le KERNEL sollicite le Routeur, qui consulte sa table de routage, pour savoir quel contrôleur exécuter. Le Routeur (qui sera étudié au prochain chapitre) est chargé d'assurer la correspondance entre l'URL (ou chemin d'accès à la ressource) et le contrôleur : dans notre exemple, il s'agit du contrôleur **FOSUserBundle : Registration : register**.

4 Le KERNEL exécute ce contrôleur. Le contrôleur demande au modèle **UserType** (généré par l'outil FormBuilder) les informations concernant un utilisateur, puis les transmet à la vue afin qu'elle construise sa page HTML.

5 La Vue transmet sa page HTML au contrôleur qui l'affiche à l'utilisateur.

### 1.3 Les Bundles

Un bundle est une brique ou une unité fonctionnelle de notre application. SYMFONY2 utilise ce concept qui consiste à regrouper dans un même ensemble, le bundle, tout ce qui concerne une même **fonctionnalité**, à savoir les contrôleurs, les modèles, les vues, les fichiers CSS et JavaScript, ...

Notre site est composé, quant à lui, de 4 bundles :

<sup>35</sup><sub>17</sub> Un bundle **UserBundle**, hérité d'un bundle externe FOSUserBundle, qui est en charge de la gestion des utilisateurs ainsi que des groupes, intégrer des pages d'administrations de ces utilisateur, des formulaires d'inscription, de récupération de mot de passe....

<sup>35</sup><sub>17</sub> Un bundle **PortailBundle**, chargé de toute la présentation du site, affichage du menu, des items ...

<sup>35</sup><sub>17</sub> Un bundle **ForumBundle**, chargé de la gestion du forum, des Entités Catégorie, Topic, Post ..

<sup>35</sup><sub>17</sub> Un bundle **PlatformBundle**, pour toute la gestion des actualités.



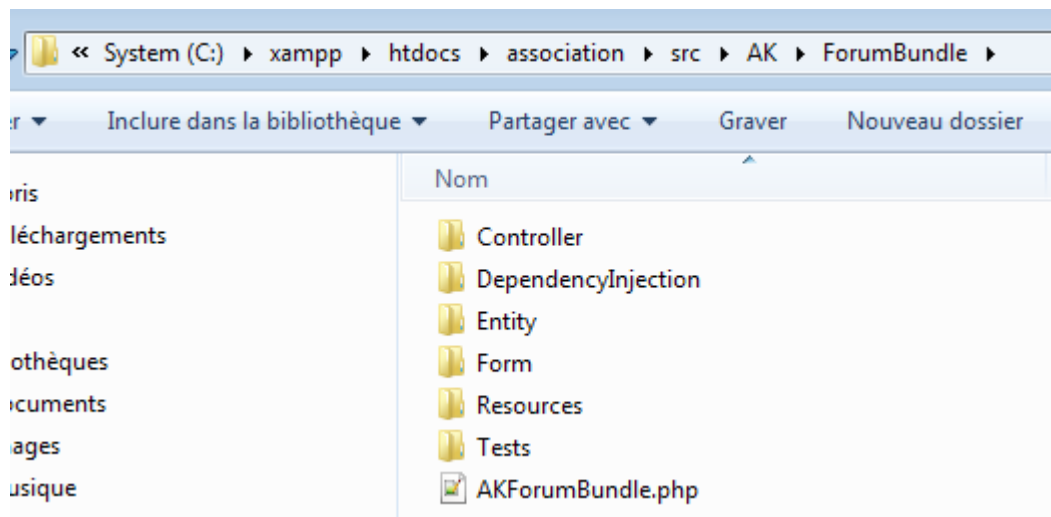
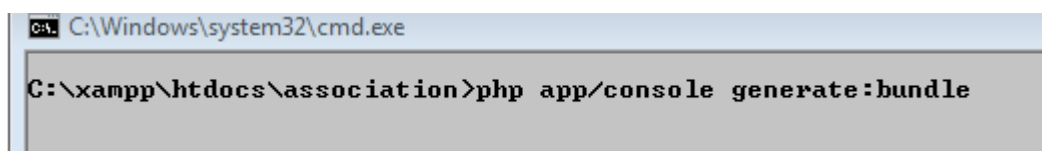


Figure 33: Les composants d'un bundle

SYMFONY assurera l'interopérabilité de ces bundles.

Pour générer un bundle, rien de plus simple avec l'outil en ligne de commande



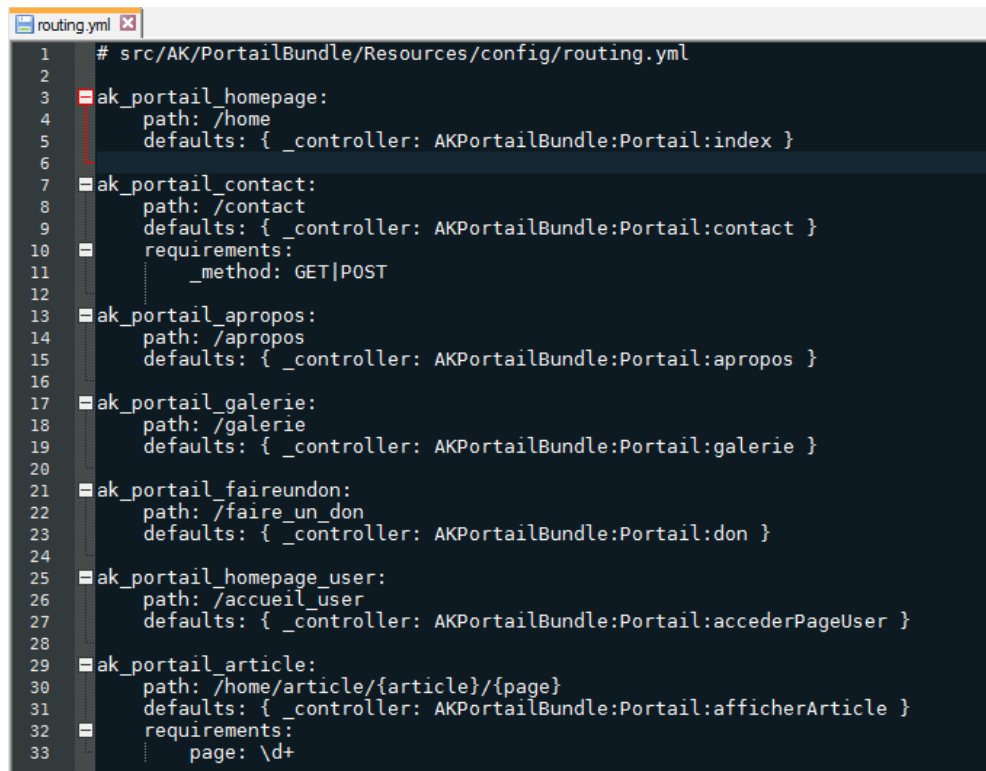
## 2. Les bases de SYMFONY2

Après avoir donné une vue d'ensemble des acteurs intervenants, nous allons maintenant nous intéresser à leur configuration et à leur fonctionnement.

### 2.1. Le Routeur

Le rôle du routeur est de déterminer quelle route empruntée pour la requête courante. En voici sa structure, dans un fichier de mapping situé dans *NomBundle/Ressources/config/routing.yml* :

Voici un exemple, l'ensemble des routes est présenté en annexe



```
1 # src/AK/PortailBundle/Resources/config/routing.yml
2
3 ak_portail_homepage:
4   path: /home
5   defaults: { _controller: AKPortailBundle:Portail:index }
6
7 ak_portail_contact:
8   path: /contact
9   defaults: { _controller: AKPortailBundle:Portail:contact }
10  requirements:
11    _method: GET|POST
12
13 ak_portail_apropos:
14   path: /apropos
15   defaults: { _controller: AKPortailBundle:Portail:apropos }
16
17 ak_portail_galerie:
18   path: /galerie
19   defaults: { _controller: AKPortailBundle:Portail:galerie }
20
21 ak_portail_faireundon:
22   path: /faire_un_don
23   defaults: { _controller: AKPortailBundle:Portail:don }
24
25 ak_portail_homepage_user:
26   path: /accueil_user
27   defaults: { _controller: AKPortailBundle:Portail:accéderPageUser }
28
29 ak_portail_article:
30   path: /home/article/{article}/{page}
31   defaults: { _controller: AKPortailBundle:Portail:afficherArticle }
32   requirements:
33     page: \d+
```

Figure 34 : Fichier de mapping du bundle « PortailBundle »

### 2.1.1 Son fonctionnement

Sur la figure précédente, on distingue 7 blocs. Chacun de ces blocs correspond à une route :

<sup>35</sup><sub>17</sub> Une entrée « path » : c'est l'URL à capturer.

<sup>35</sup><sub>17</sub> Une sortie « defaults » : ce sont les paramètres de la route, dont `_controller` qui indique le nom du contrôleur : ici, dans le deuxième bloc, il s'agit de « Portail », « contact » étant l'action du Controller à exécuter

Le but du routeur est donc, à partir d'une URL, de trouver la route correspondante et de retourner les paramètres de sortie que définit cette route, dont le contrôleur. Pour trouver la bonne route, le routeur va les parcourir une par une, dans l'ordre du fichier, et s'arrêter à la première route qui fonctionne.

Voici une illustration de ce fonctionnement :

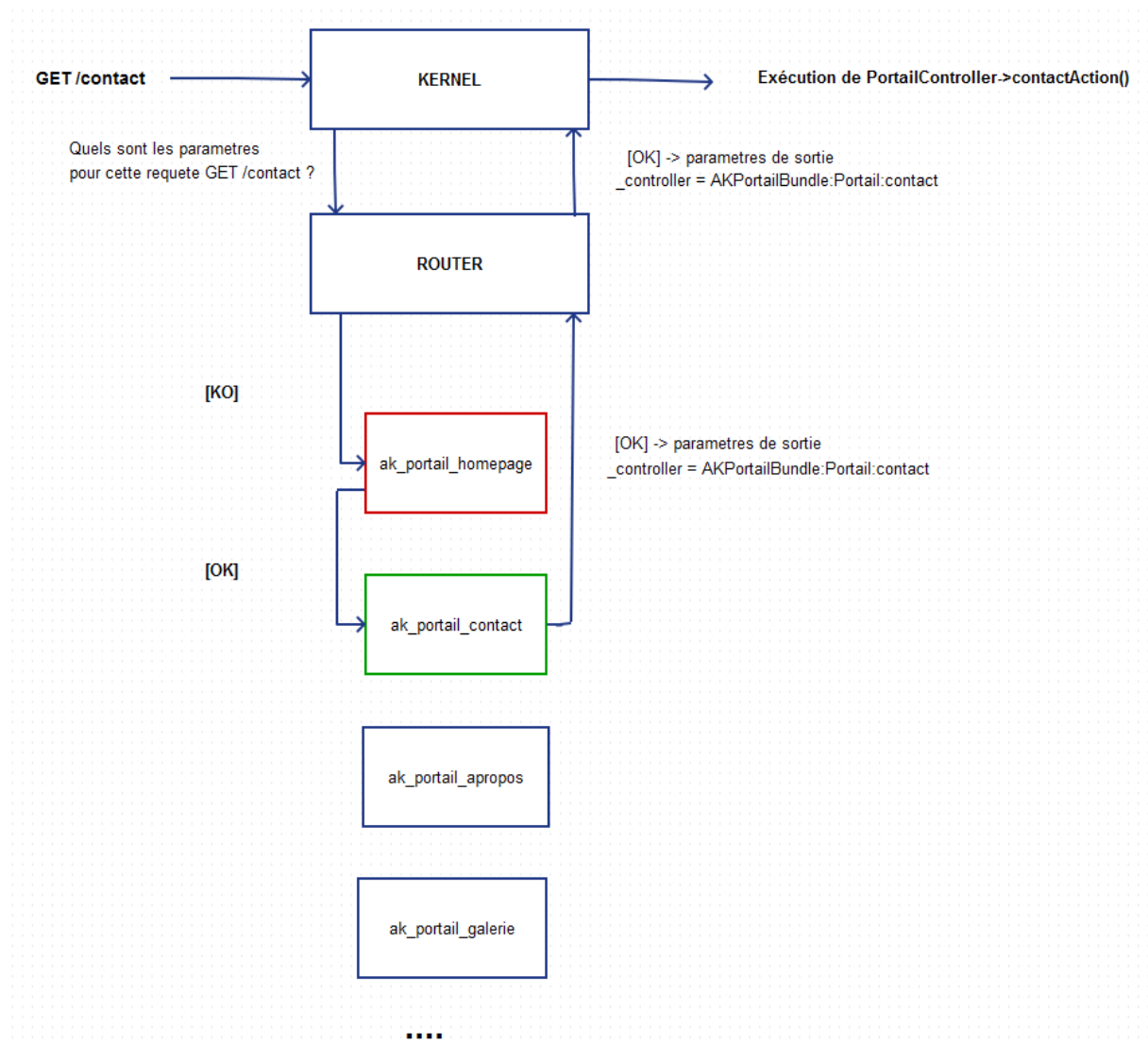


Figure 35 : Cheminement du routeur

Description textuelle :

- 1 On appelle l'URL /contact
- 2 Le routeur essaie de faire correspondre cette URL avec le path de la première route. Ici, /contact ne correspond pas du tout à /home (ligne path de la première route).
- 3 Le routeur passe donc à la route suivante qui correspond à l'url que l'on cherche à atteindre.
- 4 Le routeur s'arrête donc, il a trouvé sa route.
- 5 Il récupère les paramètres de sortie, c'est-à-dire « AKPortailBundle :Portail :contact »
- 6 Le routeur renvoie donc ces informations au KERNEL.
- 7 le noyau va exécuter le bon contrôleur avec les bons paramètres : « PortailController »

## 2.2. Le contrôleur

Le rôle du contrôleur est de retourner une réponse. Il construit la réponse en fonction des données qu'il a récupérées en entrée : paramètre de route. Il se sert de tout ce dont il a besoin pour construire la réponse : la base de données, les vues, ...il est appelé et une des fonctions est exécutée, ici `contactAction` [`AKPortailBundle :Portail :contact`]

```
<?php
namespace AK\PortailBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use AK\PortailBundle\Entity\Enquiry;
use AK\PortailBundle\Form\EnquiryType;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Security\Core\Exception\AccessDeniedException;

class PortailController extends Controller
{
    public function indexAction() {
        // On retourne la vue index.html.twig
        return $this->render('AKPortailBundle:Portail:index.html.twig');
    }

    public function aproposAction() {
        // On retourne la vue apropos.html.twig
        return $this->render('AKPortailBundle:Presentation:apropos.html.twig');
    }

    public function contactAction(Request $request)
    {
        // On construit un formulaire de contact à partir de l'objet Enquiry
        $enquiry = new Enquiry();
        $form = $this->createForm(new EnquiryType(), $enquiry);

        // Si le formulaire de contact est valide, on procède à son envoi
        if($form->handleRequest($request)->isValid())
        {
            // On construit l'objet avec ses champs à remplir
            $message = \Swift_Message::newInstance()
                ->setSubject('Contact enquiry from symblog')
                ->setFrom('sadlah75@gmail.com')
                ->setTo('sadlah75@gmail.com')
                ->setBody($this->renderView('AKPortailBundle:Portail:email.txt.twig', array('enquiry'=>$enquiry)));

            // on envoie le message après validation
            $this->get('mailer')->send($message);

            // Après validation on est redirigé vers la page contact
            return $this->redirect($this->generateUrl('ak_portail_contact'));
        }

        // si formulaire non validée, on charge la page contact
        return $this->render('AKPortailBundle:Contact:contact.html.twig', array(
            'form'=>$form->createView()
        ));
    }
}
```

Figure 36: Code PHP du contrôleur « Portail »

## 2.3. La Vue : le moteur de Templates TWIG

Un moteur de templates tel que Twig permet de bien séparer le code PHP du code HTML, dans le cadre de l'architecture MVC. Il nous permet de générer du code dynamique grâce à son pseudo-langage, plus adapté que PHP et l'héritage, proche de l'héritage objet, offre une grande souplesse de définition de templates.

Il est possible de définir des blocks dans le template parent, puis de redéfinir ces blocks dans le template enfants, tout en ayant accès à la définition parente. Tout block non redéfini sera hérité directement. On retrouve les concepts d'héritage du paradigme objet.

Nous mettons en œuvre un héritage sur trois niveaux :

- <sup>35</sup><sub>17</sub> Un template global, représentant la mise en page de tout le site
- <sup>35</sup><sub>17</sub> Un template spécifique à chaque bundle, mas sans contenu de pages
- <sup>35</sup><sub>17</sub> Un template spécifique à chaque page

Template global : [src/AK/PortailBundle/Resources/views/base.html.twig]

```
<title>{% block title %} Default title {% endblock %}</title>
</head>

<body>

  <div id="block_page">

    <header>
      <div id="titre_principal">
        
        <h1>AIKECS</h1>
        <div id="date_heure"></div>
        <h2>Association Internationale Karamokhoba pour l'Education, la Culture et la Solidarité</h2>

        <!-- <script type="text/javascript">window.onload = date_heure("date_heure");</script> -->
      </div>
    </header>

    <div id="menu_principal">
      <nav>
        <!-- Affichage du menu --></!-->
        <li><a href="{{ path('ak_portail_homepage') }}">'>Accueil</a></li>
        <li><a href="{{ path('ak_portail_apropos') }}">'>A Propos</a></li>
        <li><a href="{{ path('ak_platform_actualite_editAll') }}">'>Actualités</a></li>
        <li><a href="{{ path('ak_portail_galerie') }}">'>Galerie</a></li>
        <li><a href="{{ path('ak_portail_contact') }}">'>Contact</a></li>
        <li><a href="{{ path('ak_portail_faireundon') }}">'>Faire un don</a></li>

        {% if not is_granted('IS_AUTHENTICATED_REMEMBERED') %}
        <!-- si l'utilisateur n'est pas authentifié alors on affiche "Authentification" et "Inscription" -->
        <li><a href="{{ path('fos_user_security_login') }}">'>Authentification</a></li>
        <li><a href="{{ path('fos_user_registration_register') }}">'>Inscription</a></li>

        {% else %}
        <!-- sinon on affiche "Home" et "Deconnexion" -->
        <li><a href="{{ path('ak_portail_homepage_user') }}">'>Home</a></li>
        <li><a href="{{ path('fos_user_security_logout') }}">'>Déconnexion</a></li>

        {% endif %}
      </nav>
    </div>
    <hr>
    {% block ariane %}
      <a href="{{ path('ak_portail_homepage') }}">'>Accueil</a>
    {% endblock %}
    <hr>
    <!-- block a redéfinir -->
    {% block content %}
    {% endblock %}
  </div>
</body>
```

Figure 37 : Code source TWIG de la page ACCUEIL du site

On redéfinit les blocks [title] et [content]

```

#src/AK/PortailBundle/Resources/views/Contact/contact.html.twig
{% extends 'AKPortailBundle::base.html.twig' %}

{% block stylesheets %}
    {{ parent() }}
    <link rel="stylesheet" type="text/css" href="{{ asset('css/contact_style.css') }}" />
{% endblock %}

{% block javascripts %}
    {{ parent() }}
{% endblock %}

{% block title %} Contact {% endblock %}

{% block ariane%}
    {{ parent() }}>
    <a href="{{ path('ak_portail_contact') }}">Contact</a>
{% endblock %}

{% block content %}
    <div id="bloc_content" >
    <hr>
    <div id="infos_contact">
        <div id="infos_adresse">
            <ol>ADRESSE POSTALE</h1>
            <ol>AIKECS</h2>
            <p>Chez Alagie GASSAMA<br>
            34, Boulevard NEY Hall 23<br>
            75018 Paris</p>
        </div>
        <div id="infos_tel">
            <ol>CONTACT TELEPHONIQUE</h1>
            <p>GASSAMA Salim: 06 66 88 41 51<br>
            DABY Fanssoumane:06 31 83 50 35</p>
        </div>
    </div>
    <hr>
    <div id="formulaire">
        <form action="{{ path('ak_portail_contact') }}" method='POST' {{ form_etype(form)}} class="formu">
        <fieldset>
        <legend>Formulaire de contact</legend>
        {{ form_errors(form) }}

        {{ form_row(form.name) }}
        {{ form_row(form.email) }}
        {{ form_row(form.subject) }}

        {{ form_rest(form) }}

        <input type="submit" value="Envoyer" />
        </fieldset>
        </form>
    </div>
    </div>
{% endblock %}

```

Figure 38: Code source Twig de la page « Contact »

### 3. La Base de Données avec DOCTRINE 2

#### 3.1 La couche métier ou Model : les Entités

SYMFONY2 embarque un ORM, pour Object-Relational mapping, par défaut : DOCTRINE2.

L'objectif d'un ORM est d'assurer la persistance des données. Pour cela, on manipule des Objets, DOCTRINE se charge de les enregistrer en base de données.

Du point de vue DOCTRINE, une entité est un objet complété avec des informations de mapping, l'ANNOTATION, qui lui permettent d'enregistrer correctement l'objet en base de données, en faisant le lien entre les attributs et les colonnes.

Pour créer toutes les Entités de notre application, rien de plus simple, nous utilisons l'outil DOCTRINE en ligne de commande.

```
C:\xampp\htdocs\association>php app/console generate:doctrine:entity
```

A l'issue de cette opération, effectuée sur chacune de nos entités, nous obtenons des Entités qui comportent des attributs et des méthodes, getters/setters

Par exemple le code généré par DOCTRINE, pour l'entité « Catégorie »

```

<?php
namespace AK\ForumBundle\Entity;
use Doctrine\ORM\Mapping as ORM;

/**
 * Categorie
 *
 * @ORM\Table(name="ak_categorie")
 * @ORM\Entity(repositoryClass="AK\ForumBundle\Entity\CategorieRepository")
 * @ORM\HasLifecycleCallbacks()
 */
class Categorie
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="titre", type="string", length=255,unique=true)
     */
    private $titre;

    /**
     * @var string
     *
     * @ORM\Column(name="description", type="string", length=255)
     */
    private $description;

    /**
     * @var datetime
     *
     * @ORM\Column(name="date", type="datetime")
     */
    private $date;

    /**
     *
     *
     * @ORM\OneToMany(targetEntity="AK\ForumBundle\Entity\Topic", mappedBy="categorie",cascade={"persist"})
     * @ORM\JoinColumn(nullable=false)
     */
    private $topics;

    /**
     *
     *
     * @ORM\ManyToOne(targetEntity="AK\UserBundle\Entity\User", inversedBy="categories")
     * @ORM\JoinColumn(nullable=false)
     */
    private $user;

    /**
     *
     *
     * @ORM\Column(name="nb_topics", type="integer")
     */
    private $nbTopics=0;
}

```

Figure 39 : Code PHP de l'entité « Categorie »

En base de donnée, cet entité sera traduit en une table nommée « ak\_categorie » et aura entre autres un champ intitulé titre, de type VARCHAR, de longueur 255 et unique.



← Serveur: 127.0.0.1 » Base de données: association » Table: ak\_categorie

Afficher

Structure

SQL

Rechercher

Insérer

Exporter

	#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra
<input type="checkbox"/>	1	<b>id</b>	int(11)			Non	Aucune	AUTO_INCREMENT
<input type="checkbox"/>	2	<b>titre</b>	varchar(255)	utf8_unicode_ci		Non	Aucune	
<input type="checkbox"/>	3	<b>description</b>	varchar(255)	utf8_unicode_ci		Non	Aucune	
<input type="checkbox"/>	4	<b>date</b>	datetime			Non	Aucune	
<input type="checkbox"/>	5	<b>nb_topics</b>	int(11)			Non	Aucune	
<input type="checkbox"/>	6	<b>user id</b>	int(11)			Non	Aucune	

Figure 40: Structure de la table « Categorie »

Sous DOCTRINE le développeur est orienté Objet.

## 3.2 La gestion des Entités avec DOCTRINE2

Après avoir construit nos entités grâce à l'ORM DOCTRINE et avant de pouvoir les manipuler, il faut d'abord créer la table correspondante dans la BD.

On crée tout d'abord la Base de données, encore avec l'outil DOCTRINE en mode console après avoir configuré le fichier :

```
# app/config/parameters.yml

parameters:
    database_driver: pdo_mysql
    database_host: 127.0.0.1
    database_port: null
    database_name: association
    database_user: root
    database_password: null
    mailer_transport: null
    mailer_host: null
    mailer_user: null
    mailer_password: null
    locale: fr
    secret: f1022aa69c1fca99d4e009edaf91e0fe8acac89d
```

Figure 41: Paramètre de configuration de la création DB

Puis en ligne de commande :

```
C:\xampp\htdocs\association>php app/console doctrine:database:generate
```

Puis on génère les tables à l'intérieur de cette base de données, en exécutant la commande suivante :

```
C:\xampp\htdocs\association>php app/console doctrine:schema:update --force-sql
```

### 3.3 Les relations entre les Entités

Après avoir créé un ensemble d'entités indépendantes les unes des autres, et créé leur représentation sous forme de tables, on va devoir construire un ensemble d'entités en relation les unes avec les autres.

DOCTRINE nous permet de représenter les relations entre entités de manière simple et efficace, en ajoutant quelques annotations supplémentaires.

Toute relation [1..1] est traduite en annotation par `OneToOne`

Toute relation [1..n] est traduite par une relation `OneToMany`

Et enfin toute relation [n..m] est traduite par `ManyToMany`.

Prenons l'exemple des entités `Catégorie` et `Topic` :

1 **catégorie** comporte plusieurs **topics**, mais à un **topic** correspond une **catégorie**. On traduirait cette relation par une relation bidirectionnelle : `OneToMany` coté catégorie, et `ManyToOne` coté Topic.

```

<?php
namespace AK\ForumBundle\Entity;
use Doctrine\ORM\Mapping as ORM;

/**
 * Categorie
 *
 * @ORM\Table(name="ak_categorie")
 * @ORM\Entity(repositoryClass="AK\ForumBundle\Entity\CategorieRepository")
 * @ORM\HasLifecycleCallbacks()
 */
class Categorie
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="titre", type="string", length=255, unique=true)
     */
    private $titre;

    /**
     * @var string
     *
     * @ORM\Column(name="description", type="string", length=255)
     */
    private $description;

    /**
     * @var datetime
     *
     * @ORM\Column(name="date", type="datetime")
     */
    private $date;

    /**
     *
     *
     * @ORM\OneToMany(targetEntity="AK\ForumBundle\Entity\Topic", mappedBy="categorie", cascade={"persist"})
     * @ORM\JoinColumn(nullable=false)
     */
    private $topics;

    /**
     *
     *
     * @ORM\ManyToOne(targetEntity="AK\UserBundle\Entity\User", inversedBy="categories")
     * @ORM\JoinColumn(nullable=false)
     */
    private $user;
}

```

Figure 42 : Code source PHP de l'entité CATEGORIE

```

<:php
namespace AK\ForumBundle\Entity;

use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\ORM\Mapping as ORM;

/**
 * Topic
 *
 * @ORM\Table(name="ak_topic")
 * @ORM\Entity(repositoryClass="AK\ForumBundle\Entity\TopicRepository")
 * @ORM\HasLifecycleCallbacks()
 */
class Topic
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="titre", type="string", length=255,unique=true)
     */
    private $titre;

    /**
     * @var datetime
     *
     * @ORM\Column(name="date", type="datetime")
     */
    private $date;

    /**
     *
     * @ORM\ManyToOne(targetEntity="AK\ForumBundle\Entity\Categorie", inversedBy="topics")
     * @ORM\JoinColumn(nullable=false)
     */
    private $categorie;

    /**
     *
     * @ORM\OneToMany(targetEntity="AK\ForumBundle\Entity\Post", mappedBy="topic")
     * @ORM\JoinColumn(nullable=false)
     */
    private $posts;
}

```

Figure 43 : Code source de l'entité TOPIC

# *Conclusion*

---

Le site est fonctionnel dans son ensemble. Il nous reste cependant à retravailler tout de même le design à certains endroits de l'application, avant de pouvoir le déployer sur un serveur de production.

Ces 2 mois de stage ont été pour moi une expérience très enrichissante, dans la mesure où j'ai pu mettre en application les connaissances acquises durant mes 4 mois de formation, et surtout je me suis beaucoup investi dans l'apprentissage du Framework PHP, SYMFONY2.

Je suis conscient de l'exigence, de la rigueur et du savoir en matière de développement, que je dois davantage travailler afin de devenir je l'espère un bon développeur web.

Je remercie une fois de plus mon collègue Mr GASSAMA pour m'avoir confié la tâche de réaliser ce projet, sans oublié bien évidemment de remercier tout le personnel administratif et pédagogique.

# Annexe

L'ensemble des routes lues par le routeur :

ak_platform_actualite_add	ANY	ANY	ANY	/platform/actualite/add
ak_platform_actualite_editAll	ANY	ANY	ANY	/platform/actualite/editAll
ak_forum_categorie_editAll	ANY	ANY	ANY	/forum/home
ak_forum_categorie_add	ANY	ANY	ANY	/forum/categorie/add
ak_forum_categorie_delete	ANY	ANY	ANY	/forum/categorie/delete/{id}
ak_forum_topic_add	ANY	ANY	ANY	/forum/topic/add
ak_forum_topic_editAll	ANY	ANY	ANY	/forum/topic/editAll/{id}
ak_forum_topic_delete	ANY	ANY	ANY	/forum/topic/delete/{id}
ak_forum_post_add	ANY	ANY	ANY	/forum/post/add
ak_forum_post_editAll	ANY	ANY	ANY	/forum/post/editAll/{id}
ak_forum_post_delete	ANY	ANY	ANY	/forum/post/delete/{id}
ak_forum_topic_view	ANY	ANY	ANY	/forum/topic/view/{id}
ak_portail_homepage	ANY	ANY	ANY	/home
ak_portail_contact	GET:POST	ANY	ANY	/contact
ak_portail_apropos	ANY	ANY	ANY	/apropos
ak_portail_galerie	ANY	ANY	ANY	/galerie
ak_portail_faireundon	ANY	ANY	ANY	/faire_un_don
ak_portail_homepage_user	ANY	ANY	ANY	/accueil_user
ak_portail_article	ANY	ANY	ANY	/home/article/{article}/{page}
fos_user_security_login	ANY	ANY	ANY	/login
fos_user_security_check	POST	ANY	ANY	/login_check
fos_user_security_logout	ANY	ANY	ANY	/logout
fos_user_profile_show	GET	ANY	ANY	/profile/
fos_user_profile_edit	ANY	ANY	ANY	/profile/edit
fos_user_registration_register	ANY	ANY	ANY	/register/
fos_user_registration_check_email	GET	ANY	ANY	/register/check-email
fos_user_registration_confirm	GET	ANY	ANY	/register/confirm/{token}
fos_user_registration_confirmed	GET	ANY	ANY	/register/confirmed
fos_user_resetting_request	GET	ANY	ANY	/resetting/request
fos_user_resetting_send_email	POST	ANY	ANY	/resetting/send-email
fos_user_resetting_check_email	GET	ANY	ANY	/resetting/check-email
fos_user_resetting_reset	GET:POST	ANY	ANY	/resetting/reset/{token}
fos_user_change_password	GET:POST	ANY	ANY	/profile/change-password

Figure 44: Routes du site

## Code PHP d'ajout d'une CATEGORIE

```
<?php
namespace AK\ForumBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Request;
use AK\ForumBundle\Entity\Categorie;
use AK\ForumBundle\Form\CategorieType;
use AK\ForumBundle\Entity\Post;

class CategorieController extends Controller
{
    public function addAction(Request $request) {
        // 1.On crée un objet Categorie
        $categorie = new Categorie();

        //2.Methode raccourcie sans passer par le service factory
        $form = $this->createForm(new CategorieType(), $categorie);

        //3. On vérifie la validité du formulaire
        //si Ok on persiste l'objet, sinon on retourne à nouveau le formulaire
        if($form->handleRequest($request)->isValid()) {
            $em = $this->getDoctrine()->getManager();
            $user = $this->container->get('security.context')->getToken()->getUser();

            $categorie->setUser($user);
            $em->persist($categorie);

            $em->flush();

            $request->getSession()->getFlashBag()->add('notice','Catégorie bien enregistrée.');
```

/\*
4.On applique à \$form la méthode createView() de FormBuilder
puis on transmet à la vue en parametre l'objet généré.
\*/

```
            return $this->redirect($this->generateUrl('ak_forum_categorie_editAll'));
        }

        return $this->render('AKForumBundle:Categorie:add.html.twig',array(
            'form'=>$form->createView(),
        ));
    }

    public function editAllAction() {
        $em = $this->getDoctrine()->getManager();

        $listCategories = $em->getRepository('AKForumBundle:Categorie')
            ->findAll();

        return $this->render('AKForumBundle:Categorie:edit.html.twig',array(
            'listCategories'=>$listCategories,
        ));
    }
}
```

Figure 45 : Code PHP contrôleur « Catégorie »

## Code PHP d'ajout d'un TOPIC

```
<?php
namespace AK\ForumBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Request;
use AK\ForumBundle\Entity\Categorie;
use AK\ForumBundle\Entity\Topic;
use AK\ForumBundle\Form\TopicType;

class TopicController extends Controller
{
    public function addAction(Request $request) {
        // 1.On crée un objet Topic
        $topic = new Topic();

        //2.On crée le FormBuilder via le service form factory
        $form = $this->createForm(new TopicType(),$topic);

        //3.On établit le lien Requete <-> Formulaire
        if($form->handleRequest($request)->isValid()) {
            $em = $this->getDoctrine()->getManager();
            $user = $this->container->get('security.context')->getToken()->getUser();

            $topic->setUser($user);
            $em->persist($topic);
            $em->flush();

            $request->getSession()->getFlashBag()->add('notice','topic bien enregistré');
            return $this->redirect($this->generateUrl('ak_forum_categorie_editAll'));
        }

        /*
        4.On applique à $form la méthode createView() de FormBuilder
        puis on transmet à la vue en parametre l'objet généré.
        */

        return $this->render('AKForumBundle:Topic:add.html.twig',array(
            'form'=>$form->createView(),
        ));
    }

    public function editAllByCategorieAction($id) {
        $em = $this->getDoctrine()->getManager();
        $categorie = $em->getRepository('AKForumBundle:Categorie')
            ->find($id);

        $listTopics = $em->getRepository('AKForumBundle:Topic')
            ->findByCategorie($categorie);
        ;

        return $this -> render('AKForumBundle:Topic:editAllByCategorie.html.twig', array(
            'listTopics' => $listTopics,
        ));
    }
}
```

Figure 46 : Code PHP du contrôleur « Topic »



## Code PHP d'ajout d'un POST

```
<?php

namespace AK\ForumBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Request;
use AK\ForumBundle\Entity\Post;
use AK\ForumBundle\Form\PostType;

class PostController extends Controller
{
    public function addAction(Request $request) {
        //On crée un objet Post
        $post = new Post();

        // On crée le formulaire
        $form = $this->createForm(new PostType(), $post);

        if($form->handleRequest($request)->isValid()) {
            $em=$this->getDoctrine()->getManager();

            //On récupère l'objet USER
            $user = $this->container->get('security.context')->getToken()->getUser();

            //On persiste l'objet
            $post->setUser($user);
            $em->persist($post);
            $em->flush();

            //On redirige l'utilisateur apres validation du foormulaire
            $request->getSession()->getFlashBag()->add('notice','post bien enregistré');
            return $this->redirect($this->generateUrl('ak_forum_categorie_editAll'));
        }

        return $this->render('AKForumBundle:Post:add.html.twig',array(
            'form'=>$form->createView(),
        ));
    }

    public function editAllByTopicAction($id) {
        $em = $this->getDoctrine()->getManager();
        $topic = $em->getRepository('AKForumBundle:Topic')
            ->find($id);

        $listPosts = $em->getRepository('AKForumBundle:Post')
            ->findByTopic($topic);
        ;

        return $this -> render('AKForumBundle:Post:editAllByTopic.html.twig', array(
            'listPosts' => $listPosts,
        ));
    }
}
```

Figure 47 : Code PHP du Contrôleur « Post »

# *Bibliographie*

---

- UML2, Modéliser une application Web, Pascal ROQUES Ed. EYROLLES
- UML2 par la pratique, Pascal ROQUES, Ed. EYROLLES
- UML 2 en action, Pascal ROQUES, Ed. EYROLLES
- UML 2, Pratique de la modélisation, Benoit CHARROUX...Ed. Pearson Education

# *Webographie*

---

- <http://www.w3schools.com/>
- <http://php.net/>
- <http://www.elephorm.com/>
- <http://openclassrooms.com/>
- <https://symfony.com/>

# Tableau des illustrations

---

Figure 1 : Page d'accueil du site .....	8
Figure 2 : Diagramme de contexte statique .....	14
Figure 3 : Diagramme de cas d'utilisation : non-adhérent....	15
Figure 4 : Diagramme de cas d'utilisation : adhérent .....	17
Figure 5 : Diagramme de cas d'utilisation : Forum .....	18
Figure 6 : IHM du menu de navigation .....	20
Figure 7 : Présentation de l'Association .....	21
Figure 8 : Formulaire de contact .....	21
Figure 9 : Formulaire de don .....	22
Figure 10 : Formulaire d'authentification .....	20
Figure 11 : Formulaire d'inscription .....	23
Figure 12 : Affichage du menu après authentification .....	23
Figure 13 : Formulaire de modification de profil .....	24
Figure 14 : Formulaire de modification de mot de passe .....	24
Figure 15 : Interface FORUM du point de vue administrateur .....	25

<i>Figure 16 : Interface FORUM du point de vue Adh�rent sans droits .....</i>	<i>25</i>
<i>Figure 17 : Formulaire d'ajout d'une cat�gorie .....</i>	<i>26</i>
<i>Figure 18 : Formulaire d'ajout d'un topic .....</i>	<i>26</i>
<i>Figure 19 : Formulaire d'ajout d'un post .....</i>	<i>26</i>
<i>Figure 20 : Diagramme de Navigation Front-Office .....</i>	<i>32</i>
<i>Figure 21 : Diagramme de Navigation Back-Office .....</i>	<i>32</i>
<i>Figure 22 : Diagramme de s�quence « Inscription » .....</i>	<i>33</i>
<i>Figure 23 : Diagramme d'activit� « Inscription » .....</i>	<i>34</i>
<i>Figure 24 : Diagramme de s�quence « Authentification »...</i>	<i>35</i>
<i>Figure 25 : Diagramme d'activit� « Authentification » .....</i>	<i>36</i>
<i>Figure 26 : Diagramme de classes du site .....</i>	<i>37</i>
<i>Figure 27 : Diagramme de d�ploiement .....</i>	<i>38</i>
<i>Figure 28 : MCD du site .....</i>	<i>40</i>
<i>Figure 29 : Sch�ma de la base de donn�es .....</i>	<i>41</i>
<i>Figure 30 : Contenu su dossier /app et /app/config .....</i>	<i>43</i>
<i>Figure 31 : Contenu du r�pertoire /web .....</i>	<i>44</i>
<i>Figure 32 : Parcours d'une requ�te : « Authentification »</i>	<i>45</i>
<i>Figure 33 : Les composants d'un bundle .....</i>	<i>47</i>
<i>Fichier 34 : Fichier de mapping du bundle «PortailBundle ».....</i>	<i>48</i>
<i>Figure 35 : Cheminement du routeur .....</i>	<i>49</i>
<i>Figure 36 : Code PHP du contr�leur « Portail » .....</i>	<i>50</i>
<i>Figure 37 : Code source TWIG de la page d'accueil .....</i>	<i>51</i>

<i>Figure 38 : Code source TWIG de la page « Contact »</i>	<i>.....52</i>
<i>Figure 39 : Code PHP de l'entité « Catégorie »</i>	<i>.....54</i>
<i>Figure 41 : Structure de la table « Catégorie »</i>	<i>.....54</i>
<i>Figure 41 : Paramètre de configuration de la création de la BD</i>	<i>.....54</i>
<i>Figure 42 : Code source PHP de l'entité « Catégorie »</i>	<i>.....57</i>
<i>Figure 43 : Code source PHP de l'entité « Topic »</i>	<i>.....58</i>
<i>Figure 44 : Routes du site</i>	<i>.....60</i>
<i>Figure 45 : Code PHP du Contrôleur « Catégorie »</i>	<i>.....61</i>
<i>Figure 46 : Code PHP du contrôleur « Topic »</i>	<i>.....62</i>
<i>Figure 47 : Code PHP du contrôleur « Post »</i>	<i>.....63</i>