# Remerciements :

*Pour ce stage, je souhaiterai remercier l’association MEIM France qui travaille avec Soamada 28 de m’avoir accueilli. Et en particulier Ruben ANDRIAFEHIVOLARISOA le président de l’association et celui qui se charge de la partie administrative, son papa Renaud trésorier de l’association et son autre fils Elie qui est en charge de la partie technique informatique.*

*Je tenais à remercier aussi Lior CHAMLA, Maryam BOUCHAMI, Arthur JANSSENS, Emmanuel RAVRAT, GRAFIKART, ROAD TO DEV et OPENCLASSROOM pour leurs supports précieux sur le web.*

*Alexandre BACCO pour son livre sur le Framework Symfony 3 aux éditions Eyrolles.*

*Je tenais à remercier aussi Claire QUESNEL ma formatrice de communication à la formation « Concepteur(rice) Développeur(euse) Informatique » de m’avoir accompagné pendant ce stage.*

# Introduction :

Je m’appelle Honoré RASAMOELINA et je suis stagiaire de la formation continue appelée « Concepteur (rice) Développeur (euse) Informatique ». Cette formation a pour but, selon le site du RNCP, de « prend en charge la conception et le développement d'applications informatiques ». En d’autres termes, cette formation a pour ambition de nous préparer à des métiers ayant comme thématique l’étude et le développement informatique. Ces métiers peuvent être orienté MOE (maitrise d’oeuvre) c’est-à-dire « Développeur (euse) » quel que soit la thématique ou le langage, « Analyste Programmeur (euse) » qui ressemble au métier de Développeur, « Ingénieur(e) de développement informatique » qui est aussi un métier orienté technique. Ce sont essentiellement des métiers d’exécutants et des métiers techniques. Et il y a aussi les métiers MOA (maitrise d’ouvrage), comme « Chef de projet étude et développement informatique » qui établit le besoin d’un produit d’une application qui est décrit dans un cahier des charges, « Testeur / Testeuse informatique » comme son nom l’indique teste les différentes applications du développeur avec l’aide d’outils spécifiques. A la fin de la formation, je valide un Titre Professionnel de niveau bac+4 fourni par le Ministère du Travail et la Direction régionale des Entreprises, de la Concurrence, de la Consommation, du Travail et de l'Emploi (DIRECCTE). Ce certificat est délivré lors d’une soutenance d’une durée 1h45 au cours duquel je dois présenter un ou plusieurs projets couvrant l’ensemble des compétences pour valider le titre. Ces compétences sont listées dans le Référenciel d’Activités Compétences (REAC) de la formation présent sur le site de la DIRECCTE. La formation dure au total 1032 heures soit 752 heures au centre de formation et 280 heures destinés à la période en milieu professionnel. Et pour valider cette formation, j’ai décidé de présenté le projet que j’ai fais pendant la période en milieu professionnel. Ce stage je l’ai effectué dans des associations appelées Soamada 28 et MEIM France. MEIM France est l’association qui m’a reçu et Soamada 28 est l’association naissante pour qui j’ai fait un site internet. Ce site vitrine est un site effectué avec le framework Symfony 3 qui possède un espace membre où les internautes peuvent adhérer à l’association en s’inscrivant par l’intermédiaire des formulaires ou des réseaux sociaux. Ensuite inscrit, il peut se connecter et modifier son profil. Il y a aussi une partie actualité de l’association où l’administrateur du site peut créer, éditer et supprimer des articles. Et il y a une partie promesse de don où les internautes membre de l’association ou non peuvent effectué des dons en ligne par le biais de formulaire et d’une API appelé Stripe. On peut aussi contacter la future association grâce à un formulaire de contact. Je vais donc dans ce rapport dans un premier temps, vous présentez les deux associations avec qui j’ai travaillé à savoir MEIM France et Soamada 28, ensuite je vais vous détaillé les différents étapes de conception et de développement du futur site de Soamada 28 et enfin je vous analyserai le stage en faisant un bilan de celui-ci.

# Présentation des associations

## Q**ui est l’association MEIM France ?**

### Présentation de MEIM France

L’association Mission Evangélique Indépendante de Madagascar (MEIM) est une association humanitaire qui intervient auprès des populations en précarité à Madagascar. De par son histoire, c’est un pays qui vit depuis longtemps dans la précarité et dans la pauvreté. Et ne serait dans le cadre de l’éducation, tous les enfants de ce pays ne peuvent malheureusement pas accéder à l’école qui est payante. C’est une triste réalité mais Madagascar figure au 134ème rang pour le PIB, et figure ainsi parmi les pays les plus pauvres du monde. C’est pour cela que des associations comme MEIM France existent. C’est pour permettre à Madagascar de sortir de sa précarité et de pouvoir par exemple construire des écoles pour permettre à un plus grand nombre d’enfants de bénéficier de l’éducation, d’obtenir par exemple des établissements de santé pour pouvoir se soigner décemment dans le pays. Mais la principale mission de MEIM France c’est d’aider Madagascar dans le domaine humanitaire et spirituel. L’association est principalement présidée par Ruben ADRIAFEHIVOLARISOA. L’association se présente de la manière suivante :

(Organigramme)

Les principales actions que MEIM France a déjà effectuées à Madagascar sont les suivantes :

* La construction d’un dispensaire dans un quartier dans la ville d’Ambositra situé au sud de la capitale.
* L’envoi de médicaments, d’habits, de livres, du matériel médical et informatique.
* L’organisation de voyages touristiques pour visiter de beau pays noble qu’est Madagascar.

A l’avenir MEIM France souhaite organiser d’autres projets comme par exemple :

* La construction d’écoles notamment où se trouvent les églises CEIM. Car grâce aux églises CEIM, les enfants des villages Malgaches peuvent bénéficier d’enseignements dispensé dans les églises.
* La mise en place de micro-crédit qui permettra aux membres fréquentant les églises CEIM de bénéficier de finances remboursables pour réaliser divers projets (construction de silo pour préserver la récolte de riz, achats de pirogues et de filets pour la pêche…).

L’association MEIM France est située à Montreuil en région parisienne.

L’association est constitué de bénévoles réparti à travers la France. Ensemble, ils établissent des relations avec les églises CEIM et l’association MEIM Madagascar, ces principaux partenaires.

Pour adhérer, il suffit d’aller sur le site internet de l’association où tout ce que j’ai pu écrire dans ses lignes sont présentes et ces actions en plus détaillé : <https://www.meim.fr/index.php>

### L’organisation technique de MEIM France.

Au point de vue architecture informatique, MEIM France propose une infrastructure suivante :

(Infrastructure informatique)

### MEIM France et mon stage

MEIM France est l’association avec qui j’ai signé la convention de stage. C’est donc cette association qui m’a accueilli dans ce stage. Et Ruben ANDRIAFEHIVOLARISOA ainsi que son papa Renaud m’ont proposé de réaliser un site internet pour une association qui n’existe pas et qui existera avec le site internet qui sera créer et qui s’appelle Soamada 28.

## Qui est l’association Soamada 28 ?

### Présentation de Soamada 28

Au même titre que MEIM France, Soamada 28 sera une association humanitaire qui interviendra aussi auprès de populations désœuvrées de Madagascar. J’en parle au futur car cette association juridiquement parlant n’existe pas encore. C’est d’ailleurs pour cela que je suis là. C’est pour tenter de la faire exister. Comment ? En développant un site internet afin de présenter d’une part l’association, et d’autre part de pouvoir adhérer et la soutenir également. Soamada 28 a aussi des projets pour Madagascar comme par exemple :

* La mise en place de cantine qui permettra aux populations de déjeuner sainement.
* La création d’une école pour permettre aux petits enfants malgaches de suivre un enseignement.
* La création d’une fontaine à eau permettant aux malgaches de pouvoir de désaltérer. D’autant qu’en plein été, les températures peuvent régulièrement dépasser les 30°
* Ou alors la distribution de fournitures pour les enfants scolarisés.

L’association Soamada 28 est également présidée par Ruben ANDRIAFEHIVOLARISOA et son papa Renaud sera aussi le trésorier de l’association. Donc l’organigramme est le même que MEIM France. Et pour la partie technique c’est un de mes amis rencontré dans un BTS Informatique à Cachan qui m’encadrera et qui est aussi un des fils de Renaud à savoir Elie ANDRIAFEHIVOLARISOA.

→ voir la partie qui évoque l’organigramme de MEIM France.

### L’organisation informatique de Soamada 28.

Enoncé précédemment, l’association Soamada 28 n’est pas créée. Elle n’a pas encore de statut, ni de locaux. Elle n’a donc pas d’infrastructure informatique. Mais néanmoins, Soamada 28 sera partenaire de MEIM France. Donc, elle utilise l’architecture matérielle de l’association.

→ voir la partie qui évoque l’infrastructure informatique de MEIM France.

### Soamada 28 et mon stage

Et pour tenter de faire exister l’association ainsi que les différents projets de Soamada 28, la famille ANDRIAFEHIVOLARISOA m’a demandé de concevoir et de développer un site internet afin de communiquer sur ces différentes actions et de pouvoir adhérer aux projets de celle-ci. Ce site doit premettre d’avoir un espace membre où on peut s’enregistrer, se connecter et mettre à jour ses informations par le biais des formulaires mais aussi par le biais des réseaux sociaux. De pouvoir aussi contacter l’association par le biais d’un formulaire, de s’abonner à une lettre d’information où l’administrateur du site peut créer, modifier ou supprimer des articles. Il peut évidemment les afficher pour que tous les internautes puissent les lire. Et aussi un formulaire de don permettant aux adhérents de soutenir l’association.

# Présentation du projet

## Le besoin de l’association

L’association Soamada 28 n’existant pas encore, il fallait trouver un moyen de faire d’abord parler d’elle. Afin de trouver des donateurs pour financer les futurs projets, il y fallu trouver un tuyau. Et ce tuyau c’est le site Internet que je dois développer. Ce site a doit donc :

* Permettre de proposer une adhésion à celle-ci par le biais d’un formulaire d’inscription ou par le biais directement des réseaux sociaux à savoir Facebook, Google, Twitter et compagnie. Et n’importe quel internaute peut s’inscrire et adhérer à celle-ci.
* On doit pouvoir se connecter au site de l’association, regarder ses informations et pouvoir les mettre à jour quand on nous semble.
* Le site doit présenter ses différents projets à venir. C’est-à-dire la création d’une cantine, la distribution de fournitures scolaires ou la construction d’une fontaine à eau accompagné de photographie et de petits commentaires.
* Le site doit permettre de contacter l’association par le biais d’un formulaire de contact. Et n’importe quel internaute qu’ils soient membre de l’association ou non peut envoyer un ou plusieurs messages à l’association. Mais c’est uniquement un seul internaute qui envoie un message.
* Les internautes membre ou non de l’association peuvent consulter les actualités de l’association Soamada 28. Les articles sont présents dans l’onglet « Journal ». Les articles sont préalablement créer par l’administrateur du site Soamada 28 qui est aussi membre de l’association. Ce dernier peut donc créer, éditer ou supprimer un ou plusieurs articles. Les internautes peuvent lire aucun article ou plusieurs articles créer par l’administrateur et peuvent télécharger en format PDF un ou plusieurs articles proposées. Et l’administrateur peut envoyer par le biais d’un bouton une lettre d’information de l’association qui recense les derniers articles crée.
* Le site doit prévoir un moyen de récolter des dons. Donc les internautes membre ou non de l’association peut effectuer aucun ou plusieurs dons consécutifs pour soutenir l’association. Cela se fait par un formulaire de don accessible à tous.

## La démarche projet du site Soamada 28

Suite à cette demande, il a fallu établir une démarche de gestion de projet. J’ai décidé d’un commun accord d’utiliser la méthode AGILE Scrum pour mener à bien cette application. La méthode AGILE est une méthode qui consiste à satisfaire les besoins réelles du client en temps réel. Ce qui implique une certaine réactivité dans le développement de l’application et dans les demandes du client. Et dans la méthode AGILE j’ai utilisé la méthode SCRUM qui est un framework de la méthode AGILE dédié au management de projet et qui améliore la productivité d’une équipe. Dans la méthode AGILE SCRUM, il y a différents personnes :

* Le Scrum Master : Qui est comme son l’indique le maitre de la méthode Scrum. Il assure la bonne cohésion de l’équipe et fait en sorte que la méthode SCRUM soit bien appliquée au sein de l’équipe. Il assure aussi que la productivité et le savoir-faire de l’équipe soit à son paroxysme dans un projet. Dans ce projet, c’est mon ami Elie ANDRIAFEHIVOLARISOA qui est le maitre SCRUM.
* La Team Member : c’est l’équipe projet qui applique la méthode SCRUM. Ils ont des rôles bien déterminés : testeur, développeur, architecte… Ils ont tous à savoir à montrer lors du projet. Ici dans le projet la Team Member est représenté par une seule personne à savoir moi Honoré RASAMOELINA
* Le Product Owner : c’est l’expert métier du projet. Il détermine les principales fonctions du projet. Il établit les priorités à développer, à tester ou à corriger. Il valide les fonctionnalités développés et joue ainsi le rôle du client en quelque sorte. Dans ce projet, il y a trois personnes : Elie ANDRIAFEHIVOLARISOA qui avait établi avec moi les principales fonctionnalités prioritaires à développer, son frère Ruben qui a établi les principales fonctionnalités du projet et aussi leur papa Renaud qui a apporté avec son fils des corrections à l’application.
* Pour ce projet, nous avons établi des sprints qui représentent le cycle de vie d’un projet marqué par des itérations de quelques semaines.
* Et à partir de cela, nous avons établi des « user stories » qui décrivent les fonctionnalités du projet initial appelé « product backlog »

En m’inspirant de cette méthode SCRUM, j’ai défini un tableau SCRUM avec un outil disponible sur internet qui s’appelle Trello. Sur le site de Trello, le tableau comporte trois colonnes : une colonne « A faire » qui liste l’ensemble des taches à effectuer pendant le projet, les « user stories » autrement dit. Il y a une colonne « A finir », qui contient les user stories développé actuellement et une colonne « Fini » qui recense les user stories développé et testé.

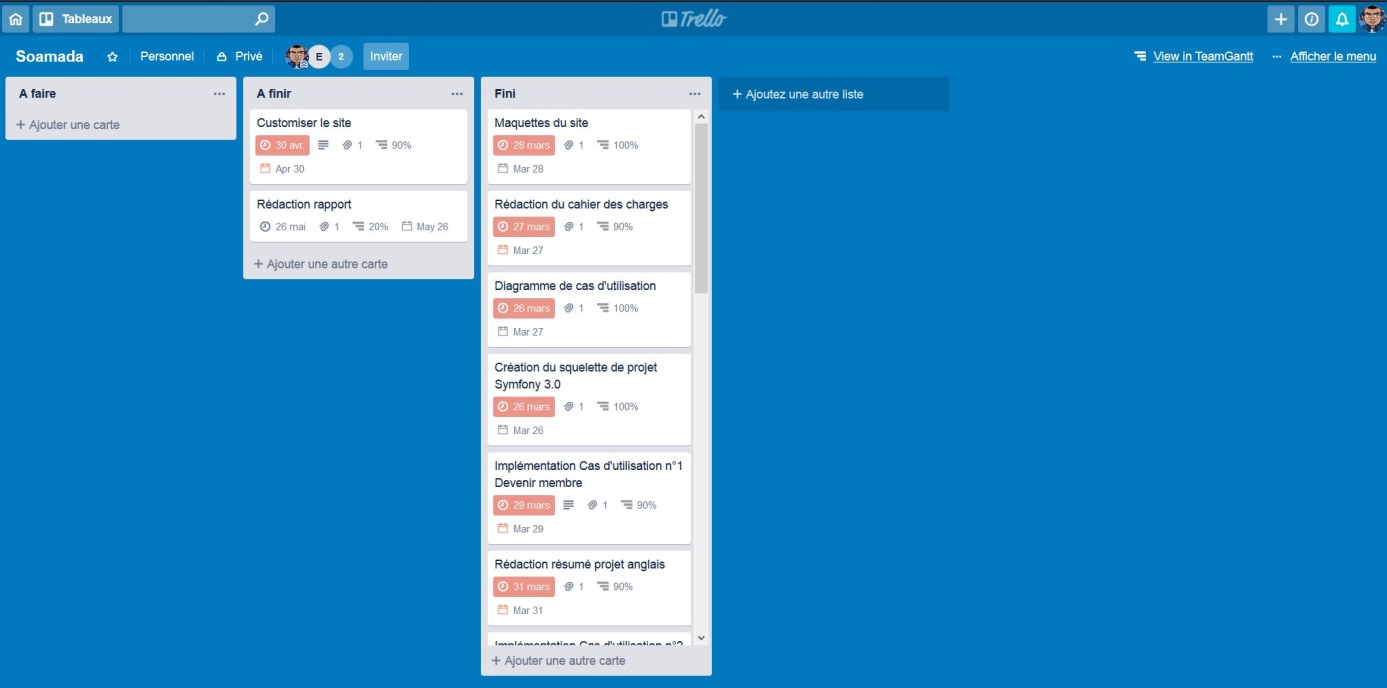


Figure 1 : Le tableau Scrum du projet Soamada 28 fait sur Trello

J’ai aussi fait pour ce projet un diagramme de GANTT. Un diagramme de GANTT est une représentation graphique permettant de visualiser la liste des tâches, leurs durées de début et de fin. Sur Trello, avec le plugin TEAMGANTT j’ai pu élaborer un diagramme de GANTT qui se trouve ci-dessous :

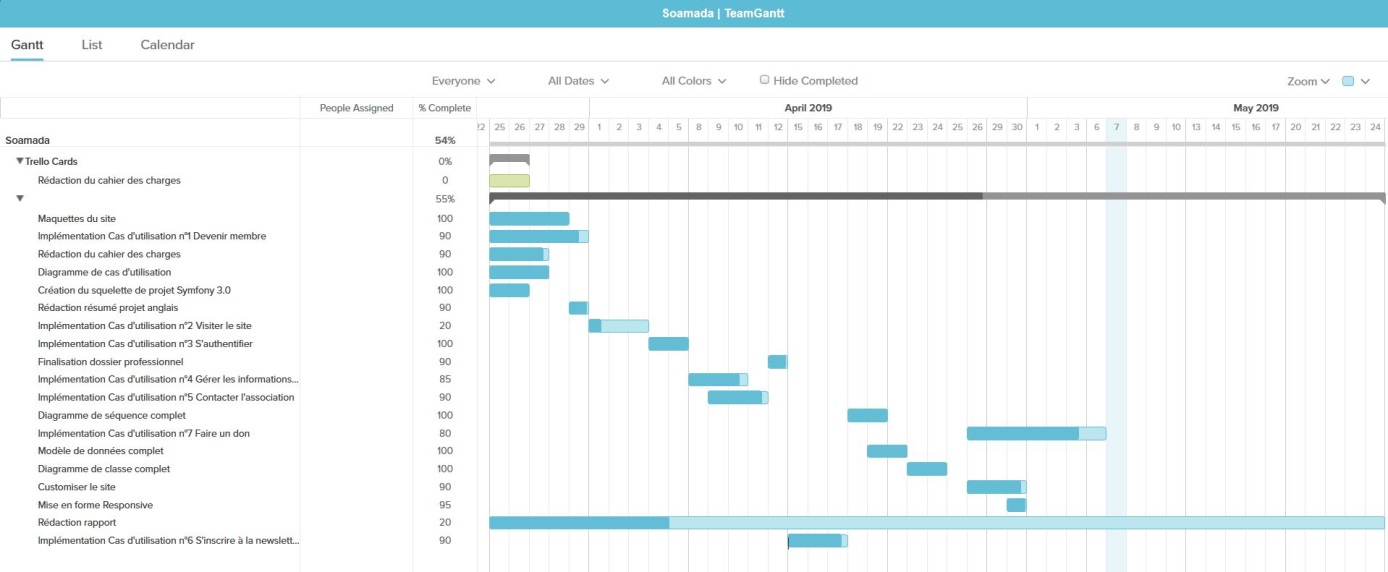


Figure 2 : Le diagramme de GANTT du projet Soamada 28 fait avec Trello

Le lien du projet Soamada sur Trello se trouve à l’adresse suivante : <https://trello.com/b/Fucvdpp9/soamada>

Et à partir de ces supports, j’ai mise en place les premiers sprints pour le projet. Et le premier sprint a été de rédiger le cahier des charges du projet Soamada 28. Ce document définit les besoin du projet dans tous les domaines (financier, calendaire, risques…). Dans ce cahier des charges il y a un diagramme UML définissant le besoin avec l’ensemble des fonctionnalités du site qui est le diagramme de cas d’utilisation. L’UML (Unified Modeling Language) est un langage qui permet de modéliser graphiquement toutes les étapes de développement orienté objet. C’est une vision uniquement d’utilisateur. Le voici :

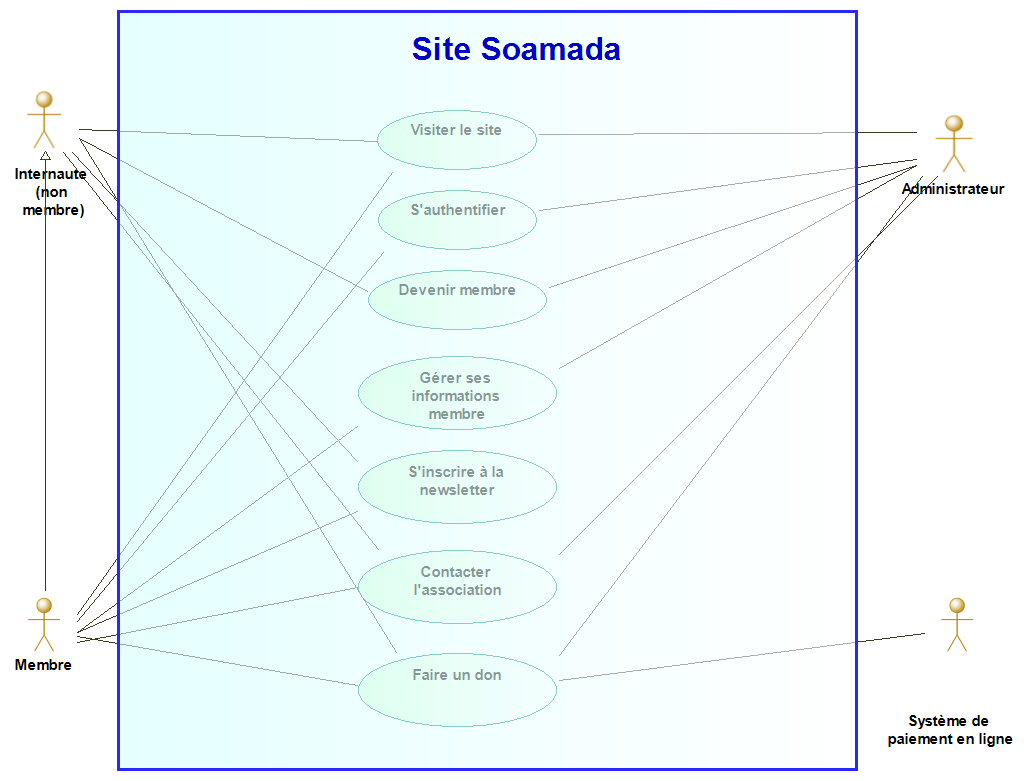


Figure 3 : Diagramme de cas d'utilisation du site Soamada 28

Les bonhommes dessinés sont les acteurs. Il y a les acteurs principaux, ceux qui sollicitent le cas d’utilisation et il y a les acteurs secondaires qui sont sollicité par le cas d’utilisation. Les ovales représentent les cas d’utilisation autrement dit les fonctionnalités du site Soamada. Et chaque acteur est relié par un lien avec le cas d’utilisation.

Et avec tous ses supports, j’ai pu établi une vraie méthode et démarche projet en bonne et due forme pour pouvoir développer le site. Après homologation et validation de ce cahier des charges, du diagramme de cas d’utilisation qui ont était mes premiers user stories que j’ai implémenté, il était présent, comme tous les users stories dans mon tableau SCRUM depuis la colonne « A faire » à la colonne « A finir ». Et ce fut comme ça fois à chaque sprint d’un user story.

Après homologation et validation de ces premières cartes, j’ai établi d’une manière très sommaire la maquette de départ du site que voici :

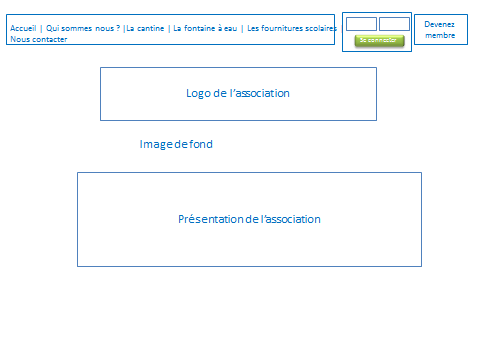


Figure 4 : Maquette de la page d'accueil du site Soamada

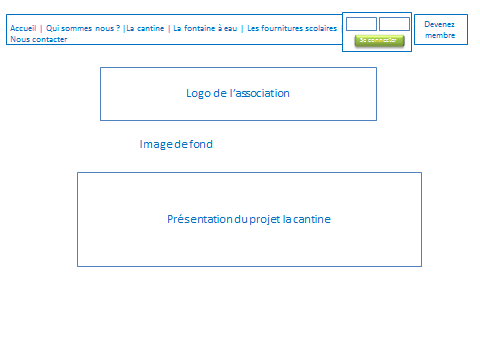


Figure 5 : Maquette du projet de la cantine du site Soamada

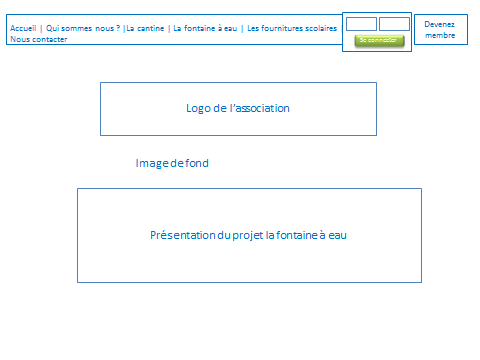


Figure 6 : Maquette de la page du projet de la fontaine à eau du site Soamada 28

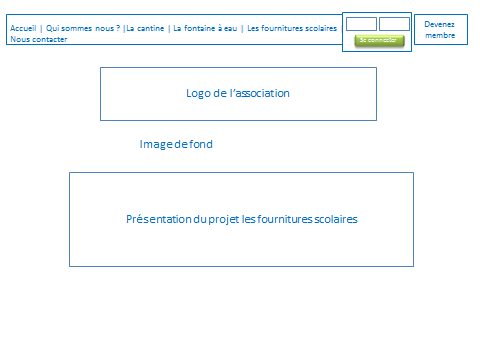


Figure 7 : Maquette de la page du projet des fournitures scolaires du site Soamada 28



Figure 8 : Maquette du formulaire de contact du site Soamada 28

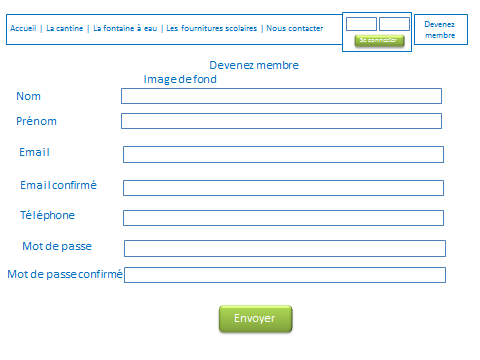


Figure 9 : Maquette du formulaire d'adhésion à l'association du site Soamada 28

Donc à partir de toutes ses ressources, j’ai commencé la partir développement qui est le gros du problème.

## Le projet web de Soamada 28

Pour développer ce site internet, j’ai décidé d’utiliser le langage PHP (HyPertext Processor). Le langage PHP en version 7 pour être plus précis. C’est un langage qui a vu le jour en 1994 par un ingénieur Rasmus Lerdorf. C’est un langage interprété côté serveur qui permet de faire de la persistance de données, de l’envoi de courriels, d’écriture de fichier…. Et dans l’écosystème de PHP, il y a pas mal d’outils dont notamment des frameworks. Appelé cadre d’applications en français dans le texte, c’est un ensemble de classe d’objets utilisables pour créer des applications informatiques. Le framework fournit ainsi des objets d’interfaces (boutons, menus, fenêtres, boites de dialogue...), des objets de service (collections, conteneurs…) et des objets de persistance (accès aux fichiers et aux bases de données) prêtent à l’emploi. Ce qui facilite le développement et qui permet au développeur de se concentrer sur l’aspect métier du projet. Et pour quoi j’ai décidé d’utilisé PHP ? Car c’est mon langage de prédilection. J’en fais depuis que j’étais en terminale (depuis 2004) et c’est ce langage qui m’a rendu accro au développement web. Et pour ce projet, je suis allé plus loin puisque j’ai aussi utilisé un framework PHP. Ce framework que j’ai choisi est Symfony. C’est la version 3 que j’ai utilisé. Ce framework a été développé par un français appelé Fabien Potencier et la première version a vu le jour en 2004. C’est le framework sur le sol Français le plus utilisé. Ce framework est détenu par une société française appelée Sensio Labs. Et j’ai décidé d’utilisé ce framework car c’est le framework le plus utilisé en France, il respecte l’organisation MVC (Modèle Vue Controleur) qui une manière d’architecturer son code dans un projet de développement, c’est une sorte de Design Pattern. Et puis lors d’une formation, j’ai été initié à ce framework et j’ai eu la aussi un véritable intérêt. D’où le choix de ces différents technologies pour développer le site de Soamada 28.

Et puis dans le cadre de la formation et pour valider celle-ci, un framework me parait plus approprier car il valide un grand nombre de compétence pour obtenir le Titre Professionnel.

Pour commencer il a fallu créer un projet Symfony 3. Cette étape constitue une étape cruciale dans le développement du site internet. Et puis, c’est aussi un user stories dans mon tableau Trello. Et pour effectuer cela j’ai dû ouvrir l’invite de commande de Windows. Ensuite j’ai dû me positionner dans le répertoire local de mes projets web à savoir : C:\wamp64\www. Et après pour générer un projet Symfony, j’ai dû savoir si Composer est installé. Composer est un gestionnaire de dépendances de librairie externe en PHP. Et il doit être installé pour générer notre projet Symfony mais aussi pour utiliser les bibliothèques externes à Symfony pour le projet plus tard.

Après avoir vu tout ça, j’ai exécuté la commande suivante dans la console de Windows :



Figure 10 : La commande Composer pour créer un projet vierge en Symfony 3

Si l’installation se passe bien il nous installe les principales librairies dont le framework a besoin. Après il nous demande de créer le fichier parameters.yml. Ce fichier répertorie l’ensemble des identifiants pour intéroger une base de données et aussi pour envoyer des courriels. La base de données qui sera utilisé pour le projet s’appellera Soamada. Donc après avoir renseigné dans la console Windows ce fichier, le projet a été correctement.

A partir de ce moment, nous avons notre projet fonctionnel et le user story a été implémenté sans problème.

A ce stade les user stories existant sur Trello et qui sont fini sont les suivants :

* Rédaction du cahier des charges
* Maquettes du site
* Diagramme de cas d’utilisation
* Création du projet squelette en Symfony 3.

Et à partir de ce moment arrive les user stories orienté Cas d’utilisation. C’est-à-dire chaque cas d’utilisation dessiné dans le diagramme de cas d’utilisation fait l’objet sur Trello d’un carte user story. C’est ce que nous verrons dans la partie suivante.

## Les cas d’utilisations du site Soamada 28

Lors de la rédaction de ce rapport, j’ai évoqué dans la partie besoin le diagramme de cas d’utilisation. Maintenant nous allons rentrer dans le détail de chaque cas d’utilisation évoqué leurs scénarii, leurs diagrammes de séquence système, leurs modèles entité association, leurs fonctionnalité développé avec le framework Symfony, leurs diagramme de classe en appliquant le stéréotype de Jacobson et leurs tests tout en restant dans la méthode AGILE SCRUM.

### Le cas d’utilisation n°1 « Devenir membre »

Le premier cas d’utilisation disponible sur Trello est le user story « Devenir membre » que voici :



Figure 11 : Le premier cas d'utilsation user story sous Trello

Ce cas d’utilisation doit permettre à un internaute lambda de devenir membre de l’association Soamada. Cette carte sous Trello était disponible comme toutes les autres dans la colonne « A faire » je l’ai déplacé dans la colonne « A finir »

Voici le diagramme de ce cas d’utilisation :

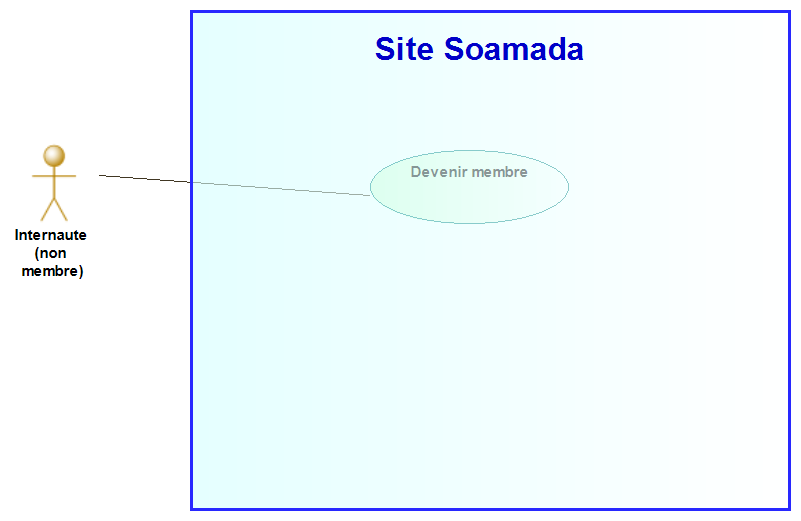


Figure 12 : Le diagramme du cas d'utilisation Devenir membre

Pour expliciter chaque cas d’utilisation, je décrierai textuellement chaque cas, voici pour le premier :

Titre : Devenir membre

Résumé : Ce cas d’utilisation permet à l’internaute de devenir membre de l’association

Acteurs principaux ou secondaire : L’internaute, le site, l’administrateur

Date de création et mise à jour : 25 mars 2019

Responsable : Moi Honoré Rasamoelina

Version : 1.0

Préconditions :

* Le site doit être fonctionnel
* Le formulaire d’inscription doit être fonctionnel
* L’internaute ne doit pas être inscrit et ne doit pas être connu du système

Scénario nominal :

1. L’internaute clique sur le bouton du site « Devenir membre »
2. Le site affiche à l’internaute le formulaire d’adhésion pour devenir membre de l’association
3. L’internaute saisie dans ce formulaire son login, son nom, son prénom, son adresse complète, son téléphone et son mot de passe
4. Le site vérifie les informations saisies par l’internaute
5. Le site informe à l’internaute de la véracité des informations saisies
6. Le site insère les informations saisies par l’internaute dans la base de données
7. L’internaute reçoit un courriel lui demandant de valider son compte par un lien
8. L’administrateur du site reçoit une notification d’inscription d’un internaute
9. L’administrateur vérifie les informations du nouveau membre
10. L’administrateur valide les informations saisies du nouveau membre
11. Le nouveau membre est informé qu’il fait bien partie de l’association

Scénarii d’exception :

SE1 : Informations saisies erronées

Le SE1 commence au point 3 du scénario nominal

Le site indique à l’internaute que les informations saisies sont incorrectes ou erronées

Le scénario reprend au point 4 du scénario nominal

SE2 : L’email existe déjà dans le système

Le SE2 commence au point 3 du scénario nominal

Le site indique que l’email saisie ou erroné ou existe déjà dans la base

Le scénario reprend au point 4 du scénario nominal

SE3 : Les mots de passe ne correspondent pas

Le SE3 commence au point 3 du scénario nominal

Le site indique que les mots de passe sont erronés ou existe déjà dans la base de données

Le scénario reprend au point 4 du scénario nominal

Scénarii alternatif :

* Le formulaire n’est pas fonctionnel
* L’insertion en base de données ne marche pas
* La mauvaise orthographe du nom et prénom pour s’inscrire

Post condition :

* L’internaute devient membre de l’association

Après avoir fait la description textuelle du cas d’utilisation, j’ai élaboré un diagramme de séquence système. Le diagramme de séquence système représente les différents échanges entre les différents objets et acteurs du système dans le temps.

Voici celui du cas d’utilisation n°1 « Devenez membre :

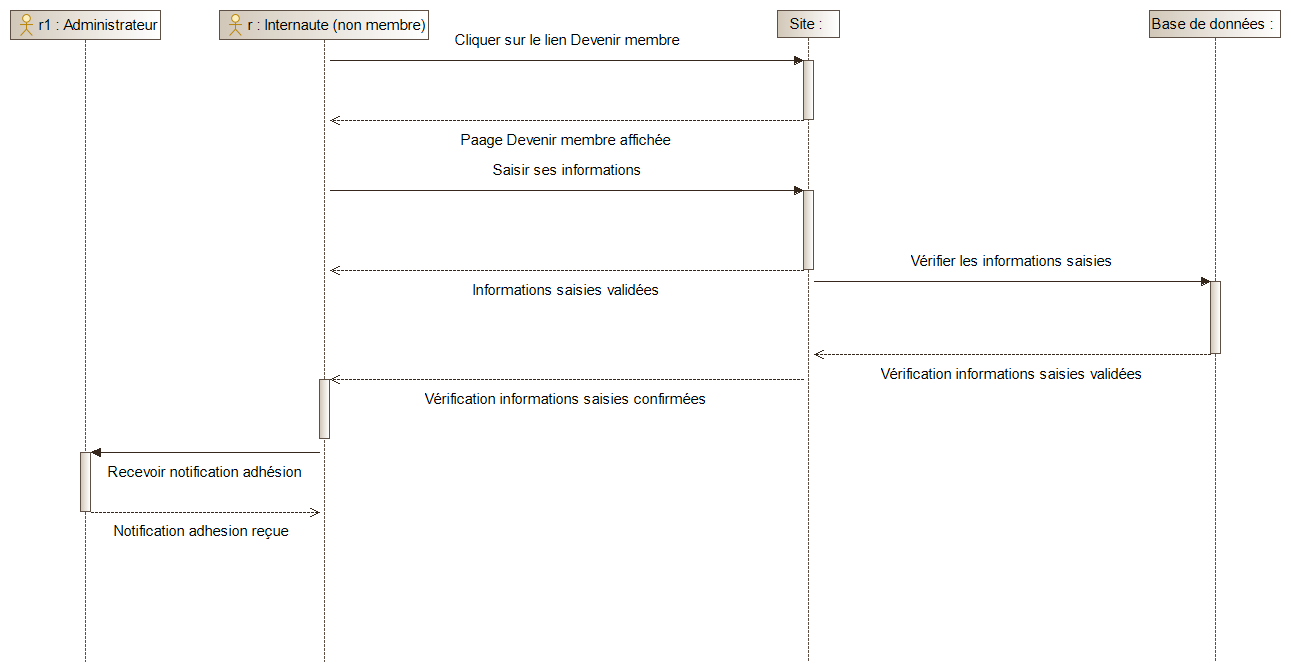


Figure 13 : Diagramme de séquence système du cas d'utilisation "Devenez membre"

Ensuite, j’ai élaboré des modèles entité association. Utilisé notamment dans la méthode Merise, cette représentation permet de donner une représentation de haut niveau des données de l’entreprise. Elle s’appelle communément sur Merise le Modèle Conceptuel de Données.

Voici la représentation pour le cas d’utilisation « Devenir membre » qui contient qu’une seule table :

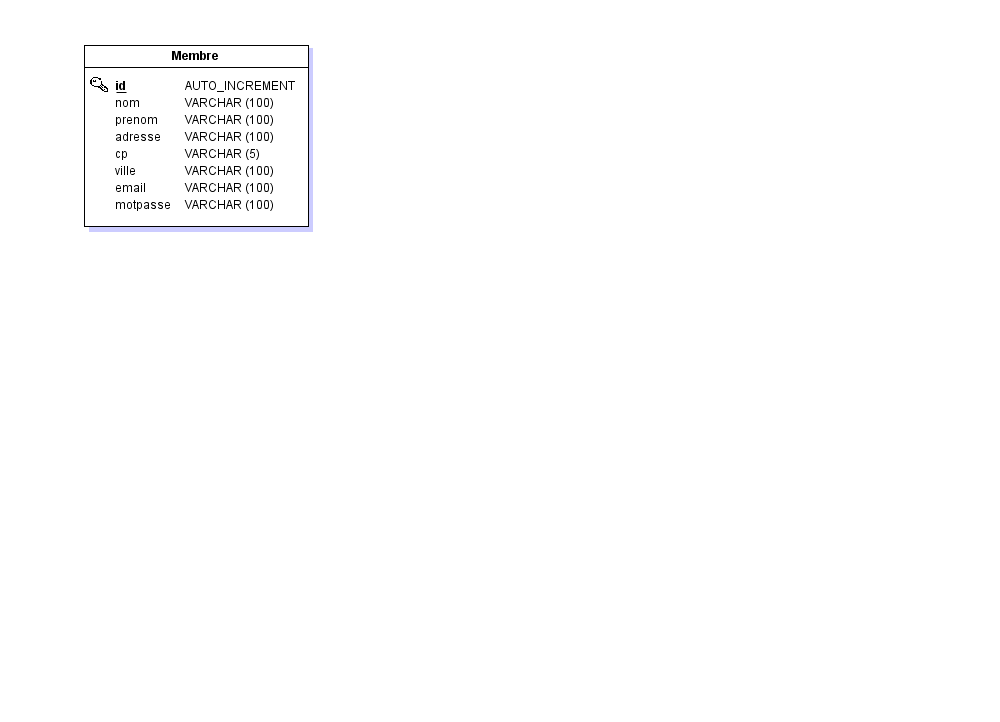


Figure 14 : Le Modèle Conceptuel de Données de Devenir Membre

Et à partir du modèle conceptuel de données, j’ai établi un modèle physique de données. Un modèle physique de données est une représentation graphique qui consiste à faciliter implantation de la base données dans le système de gestion de base de données.

Voici le modèle physique de données du cas d’utilisation Devenir membre :

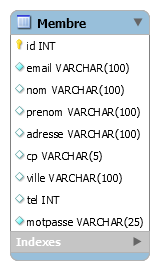


Figure 15 : Modèle physique de données Devenir membre

Ensuite après avoir conçu ces schémas, j’ai commencé l’implémentation du cas d’utilisation en développement. Sous Symfony le framework, un projet est architecturé de la manière suivante :

* App est un répertoire qui contient tout notre site web
* Bin est un répertoire qui contient les commandes PHP pour exécuter dans l’invite de commande.
* Src c’est notre répertoire de travail. C’est là où se trouve notre code source de notre site internet
* Tests c’est le répertoire des tests.
* Var est un répertoire qui indique l’activité de notre site (logs, caches…)
* Vendor est le répertoire qui possède toutes les bibliothèques externes dont on aura besoin
* Web est un répertoire accessible aux visiteurs. IL possède tous les fichiers de mise en forme CSS et JavaScript

Après pour développer ce cas d’utilisation, j’ai besoin d’un certains nombres d’outils que voici :

|  |  |
| --- | --- |
| **Thématique** | **Outils** |
| Serveur web | Apache 2.4.27 |
| Exécutable | PHP 7.0.23 |
| Système de gestion de base de données | MySQL 5.7.19 |
| Interface de gestion de base de données | PHPMyAdmin 4.8.4 |
| Environnement de développement intégré | NetBeans PHP 8.2 |

Lors de la génération de mon projet Symfony, celui-ci ma générer un bundle par défaut appelé AppBundle. Un bundle en Symfony est un composant d’un site web représenté par un répertoire de fichiers. Pour revenir à notre cas d’utilisation, j’ai décidé sous Symfony d’installer aussi un bundle. Et pour gérer tout ce qui touche à l’authentification, il y a un bundle déjà existant sous Symfony 3 que nous utiliserons pour la gestion des Membres c’est le bundle FriendOfSymfonyfonyUserBundle (FOSUserBundle).

Pour implémenter ce bundle, je me suis référé la documentation en anglais de Symfony disponible à cette adresse :

<https://symfony.com/doc/current/bundles/FOSUserBundle/index.html>

Donc dans un premier temps, j’ai configuré dans mon projet Symfony l’utilisation du bundle. Les configurations des bundles se trouve dans le répertoire app/config et c’est le fichier config.yml.

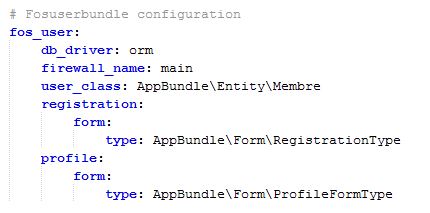


Figure 16 : Configuration du bundle FOSUserBundle dans le fichier config.yml

Ensuite, j’ai déclaré dans le fichier Appkernel.php que j’utiliserai le bundle FOSUserBundle :

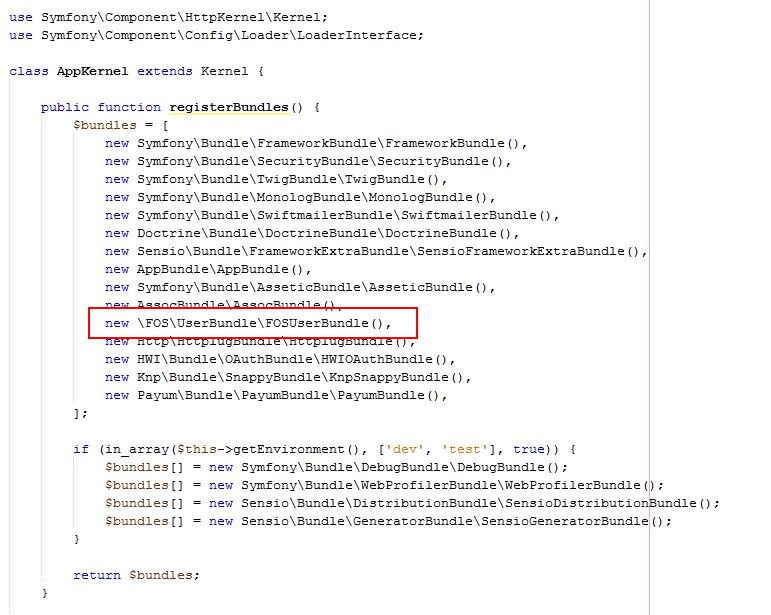


Figure 17 : Déclaration du bundle FOSUserBundle dans le fichier Appkernel.php

Ensuite j’ai du créer une entité pour pouvoir stocker mes nouveaux membres de l’association et qui s’appelle Membre. Donc dans mon environnement de développement intégré, j’ai fais un clic droit sur mon projet. Ensuite j’ai cliqué sur Symfony/Run command/ et en écrivant dans la barre de recherche entity j’ai cliqué sur console doctrine:generate:entity et il m’a exécuté dans la console de mon EDI de saisir les attributs de ma classe Membre dont j’ai besoin. Et enfin, j’ai effectué la commande doctrine:generate:entities toujours depuis mon EDI qui ma générer les assesseurs et les mutateurs de mon entité Membre. Et j’ai fais en sorte qu’il hérite de la classe Entité du Bundle FOSUserBundle qui s’appelle User :



Figure 18 : Un extrait de mon entité Membre

Après cela dans mon EDI, j’ai executé la commande doctrine :schema :update—dump-sql –force et ma table a été créé dans ma base de données. Ensuite, j’ai généré le classe formulaire de l’entité Membre. Car sous Symfony, pour construire un formulaire, il faut créer une classe de type FormType pour définir les champs, les boutons des formulaires. Je l’ai appelé RegistrationType et je l’ai fait pour le même nom que celui de la classe FormType du bundle pour qu’il puisse hériter des autres champs :



Figure 19 : La classe formulaire de l'entité Membre

Ensuite à partir de ce fichier, j’ai surchargé la vue qui doit afficher le formulaire d’adhésion fourni par le bundle que voici. Pour construire les vues, j’utiliserai le moteur de template de Symfony Twig. C’est un moteur de template qui permet d’amalgamer les balises HTML et le script PHP pour permettre aux designers de ne pas se tromper lorsque l’on met en forme le style du site :

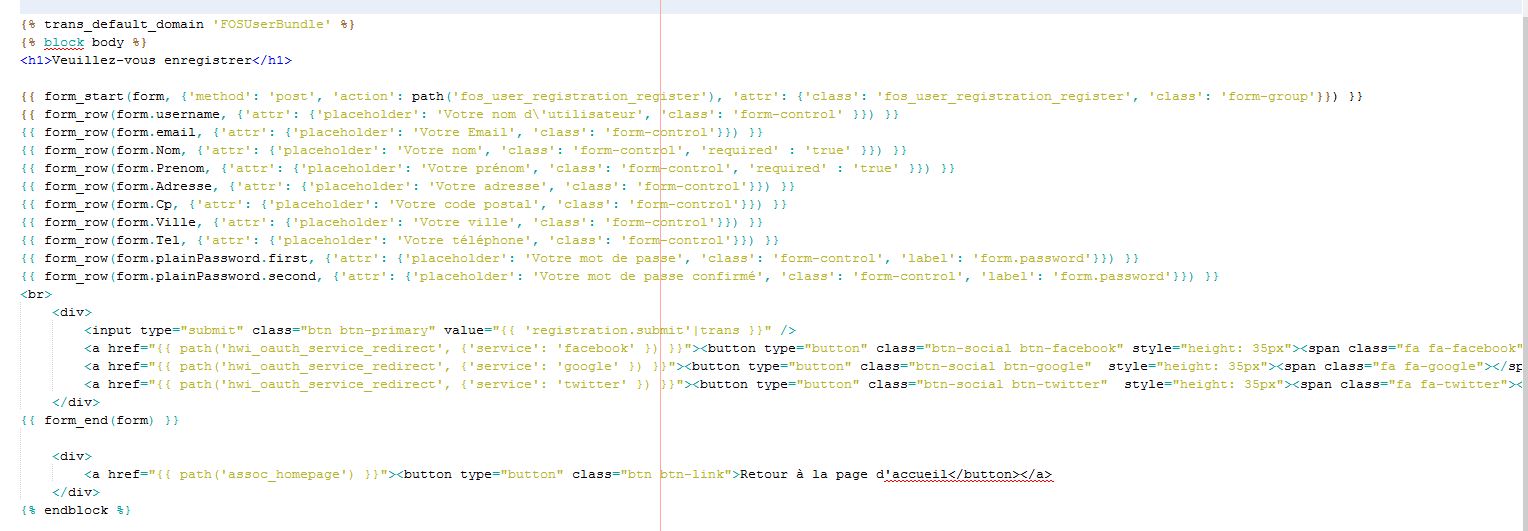


Figure 20 : Le formulaire d'adhésion en Twig

Après dans le fichier AppBundle.php de mon bundle AppBundle, j’ai déclaré l’utilisation du bundle FOSUserBundle ave la méthode getParent() que voici :

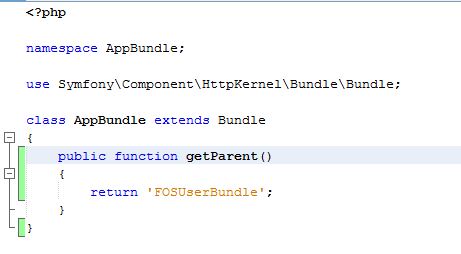


Figure 21 : Le fichier AppBundle.php où j'ai rajouté la méthode getParent()

Ensuite, j’ai copier coller le contrôleur fourni avec le bundle de FOSUserBUndle et je l’ai coller dans le répertoire contrôleur de mon bundle AppBundle. Ensuite, j’ai déclaré que j’utilisais le controller fournir par le bundle FOSUserBundle :



Figure 22 : Extrait du contrôleur RegistrationController

Après j’ai surchargé la méthode registrationAction() pour faire en sorte que l’administrateur du site reçoit un mail à chaque enregistrement de membre. Et cette méthode enregistre chaque membre inscrit dans la table de ma base de données :

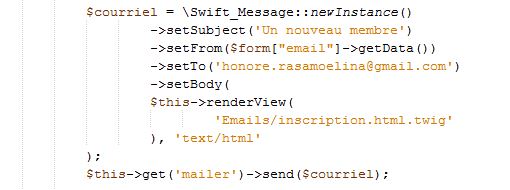


Figure 23 : L'envoi d'un courriel dans la méthode registrationAction()

Après en Symfony, il faut déterminer la route. Les routes permettent au projet Symfony le chemin à prendre pour effectuer le traitement d’une méthode. Pour cela, tout ce fait sous le fichier global routing.yml. Et comme on est dans le cadre d’un bundle installé qui est FOSUserBundle, il faut importer les routes existantes en les copiant collant dans le fichier. Les voici :



Figure 24 : Les routes du bundle FOSUserBundle

Pour l’enregistrement des membres la route du formulaire est : <http://localhost/Soamada/web/app_dev.php/register/>. Ce qui m’a donné l’affichage suivant :



Figure 25 : Le formulaire d'enregistrement des adhésions

Comme vous pouvez le constatez, j’ai rajouté au formulaire d’adhésion des boutons permettant de s’authentifier grâce aux réseaux sociaux. C’est le principe de l’authentification OAuth. Le principe est celui-ci :

* Le client c’est-à-dire moi, désigne une application tierce pour se connecter au site Soamada. L’application tierce est par exemple Facebook
* Il y a le propriétaire de la ressource c’est-à-dire moi aussi qui donne l’accès à une ressource grâce à des identifiants. Ces identifiants, il faut les déclarer dans la configuration du site de Soamada et sur le site de l’application tierce à savoir Facebook.
* Il y a le serveur de ressources protégée qui possède les ressources protégées du propriétaire de la ressource et du client
* Et il y a le serveur d’authentification qui autorise l’accès à la ressource protégée.

En appliquant ce principe du OAuth se schématise de la manière suivante :

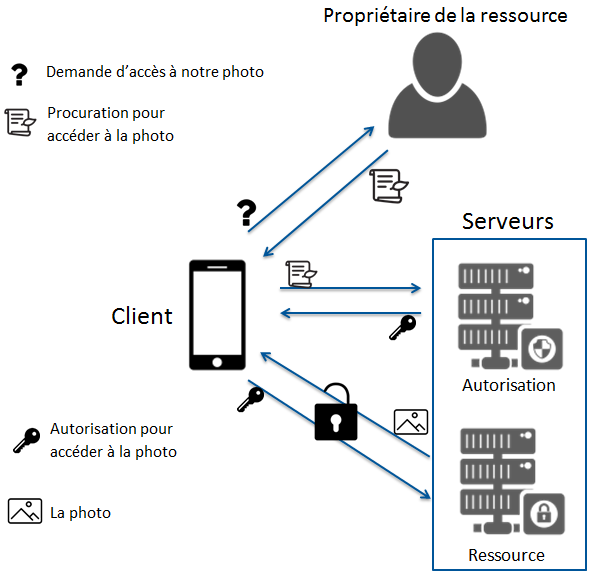


Figure 26 : Le principe du oAuth appliqué selon le site : <https://zestedesavoir.com/articles/1616/comprendre-oauth-2-0-par-lexemple/>

Concrètement dans mon application, j’ai utilisé un bundle qui s’appelle HWIOAuth. Après avoir déclaré dans mon fichier Appkernel.php le bundle, je suis allé sur l’api des développeurs de Facebook pour déclarer mon site, récupérer les identifiants de connexion :

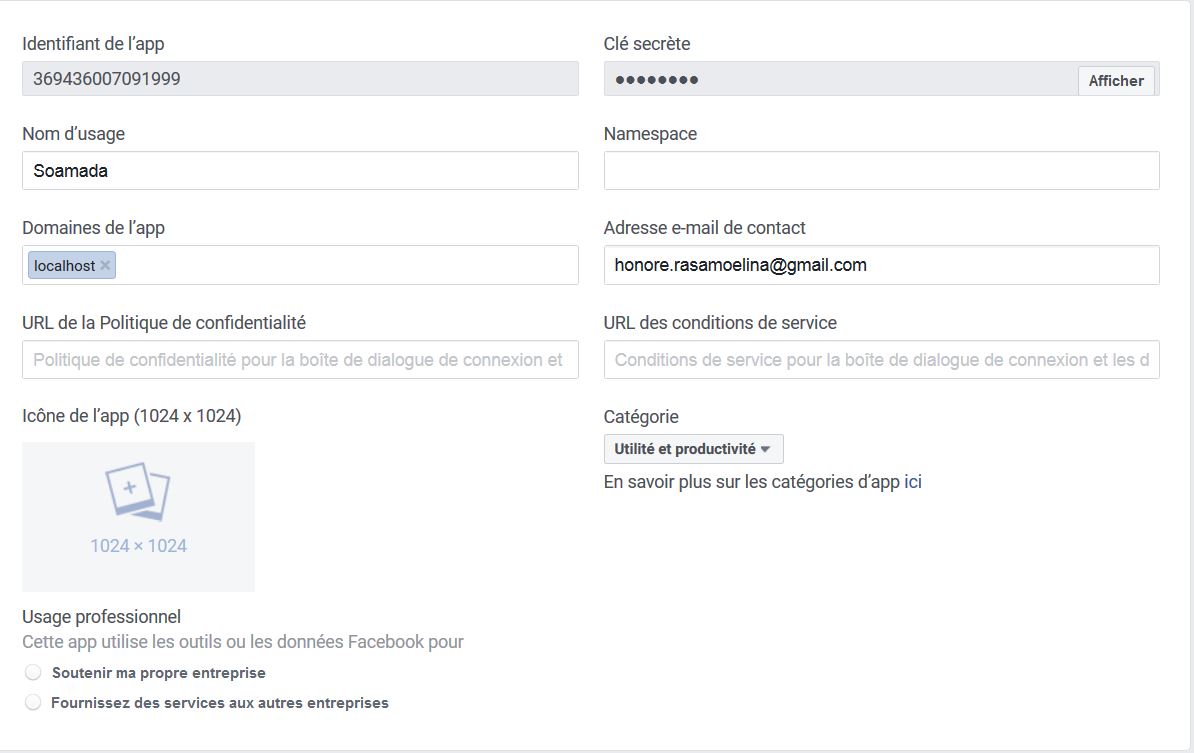
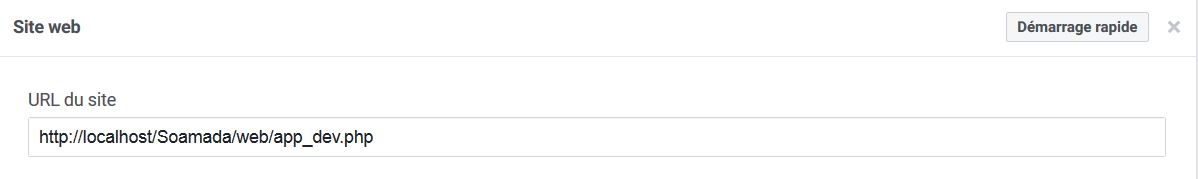


Figure 27 : L'interface de l’API des développeurs de Facebook

Après il a fallu déclarer une url qui autorise la redirection en cas de bonne connexion sur l’API de Facebook :



Ensuite dans mon fichier configuration config.yml, j’ai configuré les identifiants de connexion pour Facebook que voici :

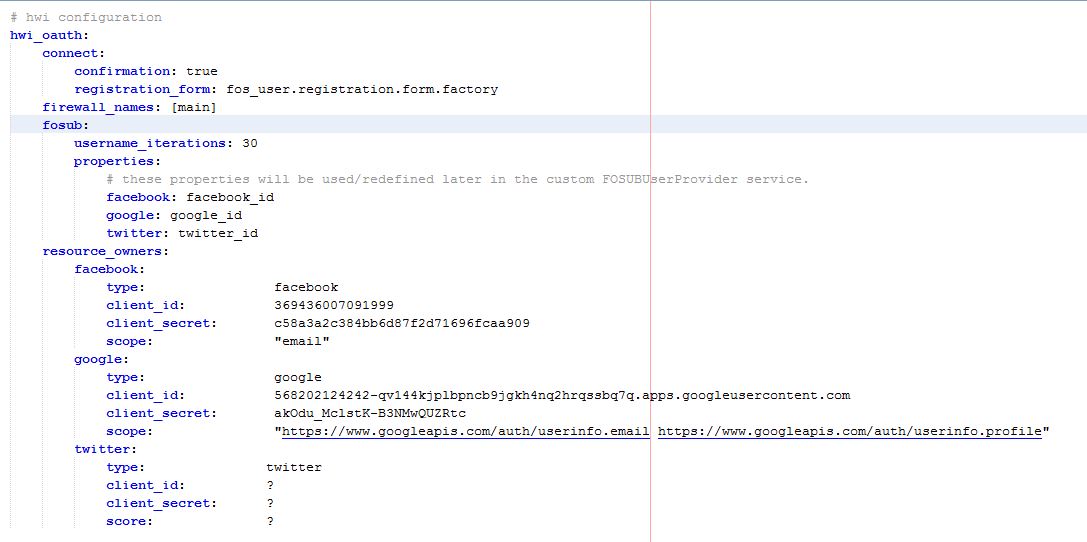


Figure 28 : La configuration du oAuth dans mon site Soamada

Après dans le fichier services.yml, j’ai déclaré le service que je dois utiliser pour faire matcher l’entité Membre et l’identifiant du fournisseur de ressource :



Figure 29 : La configuration du services.yml pour le principe oAuth

Enfin, j’ai rajouté les boutons de connexion aux réseaux sociaux avec le chemin de la route que voici :

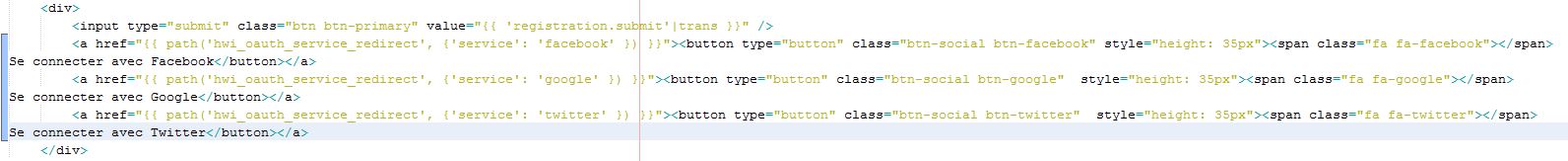


Figure 30 : Les boutons des réseaux sociaux dans le vue du formulaire d'adhésion

Sachez que j’ai appliqué le même principe pour Google. J’ai voulu le faire pour Twitter mais il faut remplir un certain nombre de critère que je n’ai pas pu avoir.

A noter j’ai appliqué aussi un style à mon site internet. Pour cela j’ai utilisé un bundle qui s’appelle AsseticBundle. Ce bundle permet de gérer les fichiers stockés et les pages CSS et JavaScript. Ensuit j’ai utilisé le framework d’interface Bootstrap pour avoir une mise en forme moderne et rapide. Tous cela je l’ai déclaré dans un template de base à savoir base.html.twig que voici :

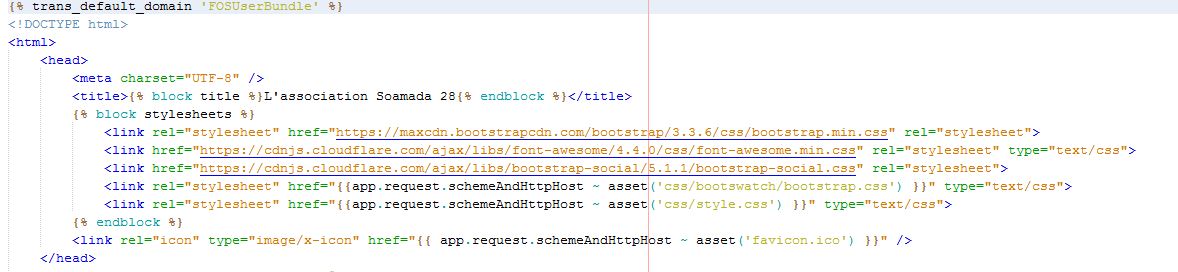


Figure 31 : Le style CSS bootstrap déclaré dans mon template de base



Figure 32 : Les feuilles JavaScript déclaré en fin de fichier dans mon template de base

Après tous cela j’ai effectué les tests du cas d’utilisation Devenir membre que voici :

1. L’internaute clique sur le bouton du site « Devenir membre » : Ok
2. Le site affiche à l’internaute le formulaire d’adhésion pour devenir membre de l’association : Ok
3. L’internaute saisie dans ce formulaire son login, son nom, son prénom, son adresse complète, son téléphone et son mot de passe : Ok
4. Le site vérifie les informations saisies par l’internaute : Ok
5. Le site informe à l’internaute de la véracité des informations saisies : Ok
6. Le site insère les informations saisies par l’internaute dans la base de données : Ok
7. L’internaute reçoit un courriel lui demandant de valider son compte par un lien : Ko
8. L’administrateur du site reçoit une notification d’inscription d’un internaute : Ok
9. L’administrateur vérifie les informations du nouveau membre : Ok
10. L’administrateur valide les informations saisies du nouveau membre : Ko
11. Le nouveau membre est informé qu’il fait bien partie de l’association : Ok

Et après tous les tests, j’ai mis en place le diagramme de classe en appliquant le stéréotype de Jacobson. C’est une représentation abstraite d’un ensemble d’objet et qui contient les informations nécessaires pour construire un objet. Et le principe de Jacobson consiste définir des rôles aux différentes classes de l’application que voici :

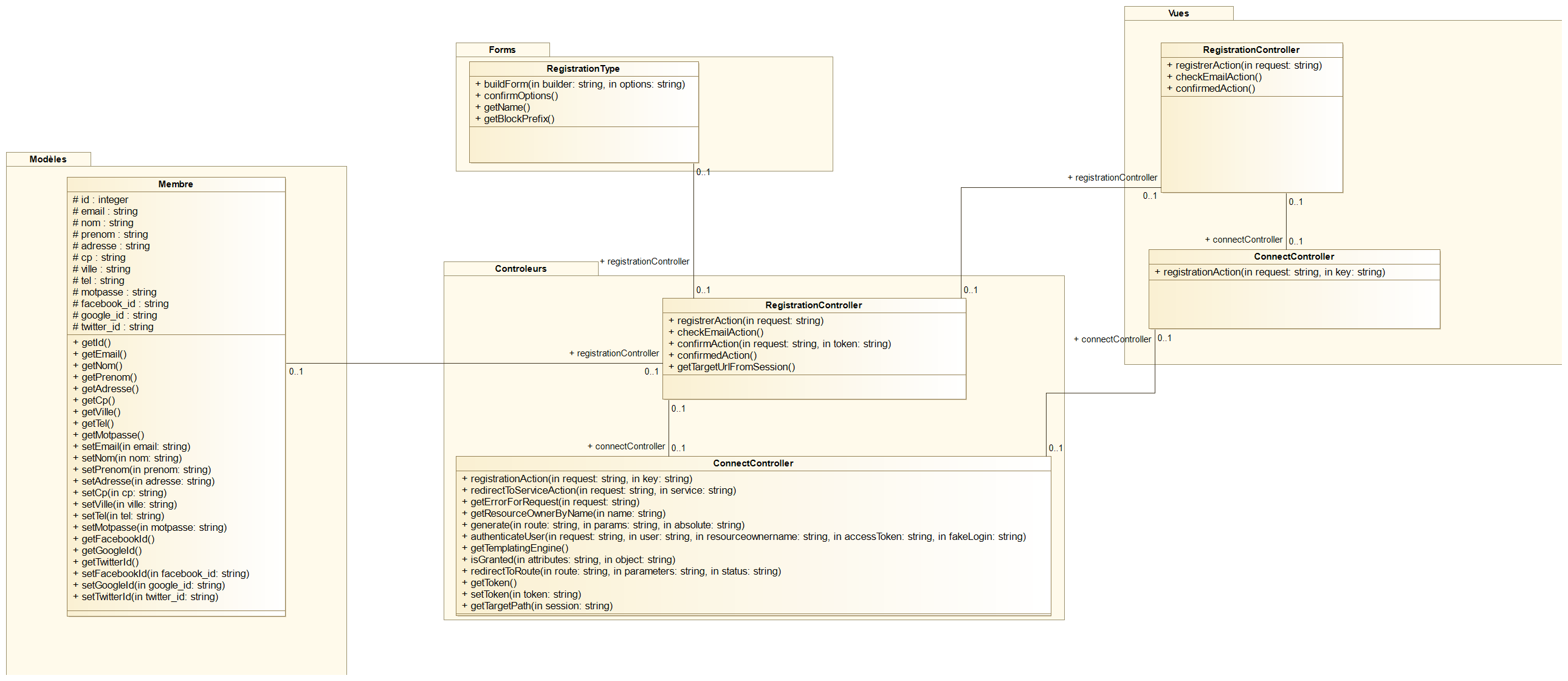


Figure 33 : Le diagramme de classe du cas d'utilisation Devenir membre

Il y a les classiques modificateurs accès :

* Public symbolisé par un signe + qui signifie que l’attribut ou la méthode est accessible dans tout le programme.
* Privé symbolisé par un signe – qui signifie que l’attribut ou la méthode est accessible uniquement dans la classe
* Protégé symbolisé par un # qui signifie que l’attribut ou la méthode est accessible uniquement dans la classe et dans ces classes dérivées.

Il y a des packages :

* Le package « Modèles » qui représentent les classes entités qui vont être persistés dans la base de données
* Le package « Forms » représente les classes qui construisent les formulaires
* Le package « Controleur » qui est le package qui possède tous les classes qui effectue le traitement de notre site. Il est en quelque sorte le trait d’union entre le package « Modèles », « Forms » et « Vues »
* Et le package « Vues » qui possède tous les classes qui affichent les vues renvoyé par les utilisateurs. Mais en Symfony, ce sont les méthodes du contrôleur qui

A l’intérieur des packages, il y a les classes avec leurs attributs typés et les méthodes symbolisés par un nom fini par une parenthèse ouvrante et fermante. Il y a aussi des classes où il y a seulement les méthodes notamment dans le packages « Controleurs ».

Voilà ce que j’ai fait pour le cas d’utilisation Devenir membre. Maintenant, je vais détailler brièvement, les autres cas d’utilisations.

### Le cas d’utilisation n°2 Visiter le site

Place au user story « Visiter le site » dont voici le diagramme de cas d’utilisation :

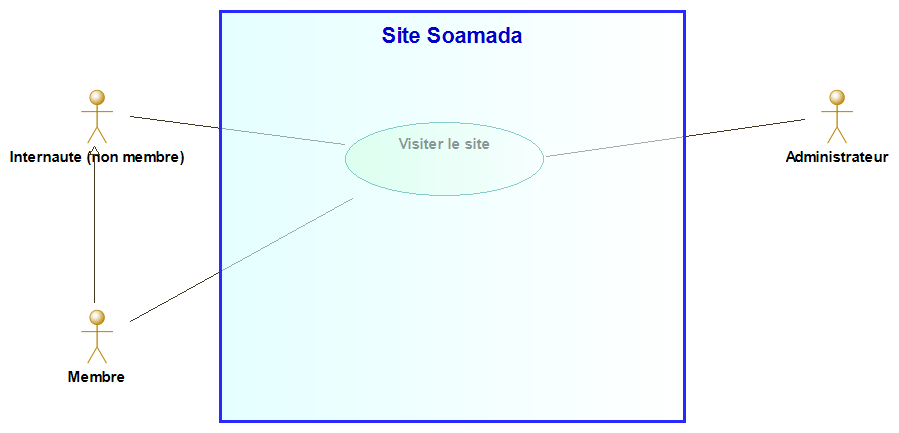


Figure 34 : Le diagramme du cas d'utilisation Visiter le site

Ensuite place aux scénarii de cas d’utilisation que voici :

|  |
| --- |
| Titre : Visiter le site  Résumé : Ce cas d’utilisation permet à l’internaute non membre de visiter le site  Acteurs principaux ou secondaire : L’internaute, le site, le moteur de recherche  Date de création et mise à jour : 25 mars 2019  Responsable : Moi Honoré Rasamoelina  Version : 1.0  Préconditions :   * Le site doit être fonctionnel   Scénario nominal :   1. L’internaute non membre saisie dans le moteur de recherche les mots clés pour accéder à l’association 2. Le moteur de recherche affiche à l’internaute les résultats référencés de sa recherche sur l’association 3. L’internaute trouve avec les résultats affichés l’association qu’il recherche 4. L’internaute clique sur le lien de l’association 5. Le moteur de recherche lui redirige vers le site de l’association 6. L’internaute visite le site notamment la page d’accueil où il y a la présentation de l’association, ses projets et le formulaire de contact.   Scénarii d’exception :  SE1 : La recherche de l’association n’a pas abouti  Le SE1 commence au point 1 du scénario nominal  Le moteur de recherche ne trouve pas les occurrences de la requête écrites par l’internaute  Le scénario reprend au point 2 du scénario nominal  SE2 : Le site est inaccessible (erreur 500)  Le SE2 commence au point 4 du scénario nominal  Le lien du site de l’association ne fonctionne pas et la page d’accueil ne s’affiche pas  Le scénario reprend au point 1 du scénario nominal    Scénarii alternatif :   * Le site n’est pas en ligne * Le moteur de recherche est hors service * La connexion internet est indispensable   Post condition :   * L’internaute accède au site internet de l’association |

Voici le diagramme de séquence système du cas d’utilisation :

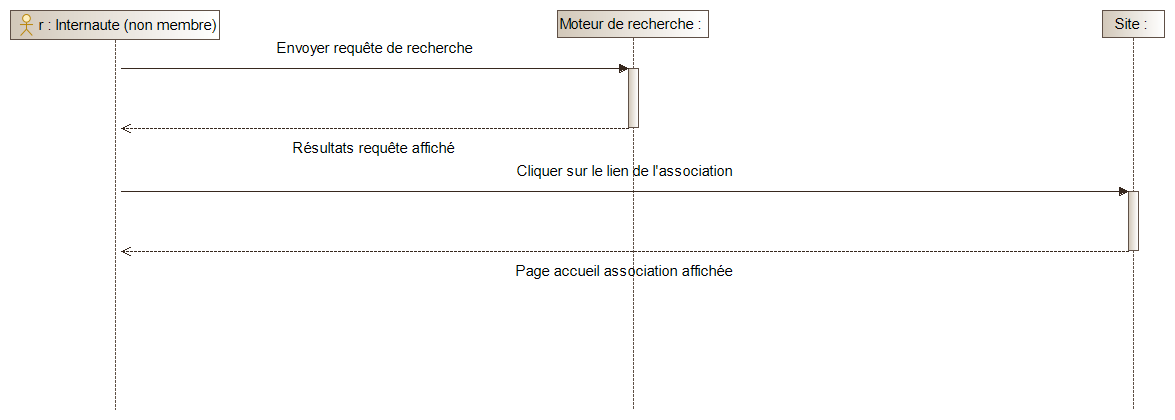


Figure 35 : Diagramme de séquence système du cas d'utilisation Visiter le site

Voici le modèle conceptuel de données du cas d’utilisation :

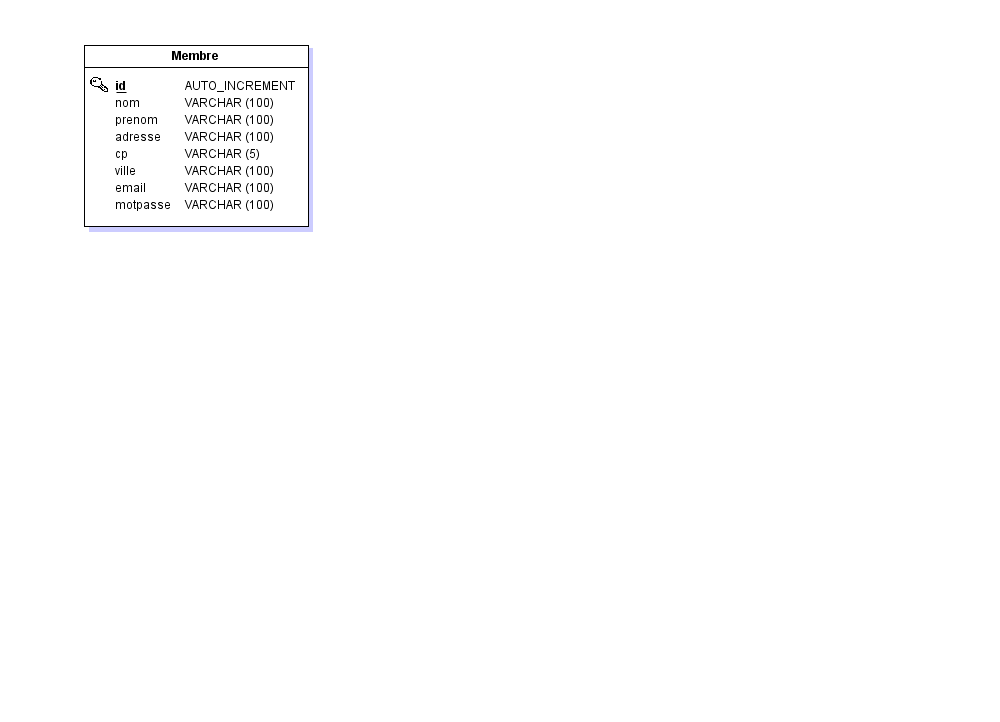


Figure 36 : MCD du cas d'utilisation Visiter le site

Et voici le modèle physique de données du cas d’utilisation :

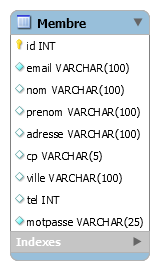


Figure 37 : MPD du cas d'utilisation Visiter le site

Voici un extrait du code la page d’accueil qui étend vers le template layout.html.twig et qui étend vers le template de base base.html.twig :

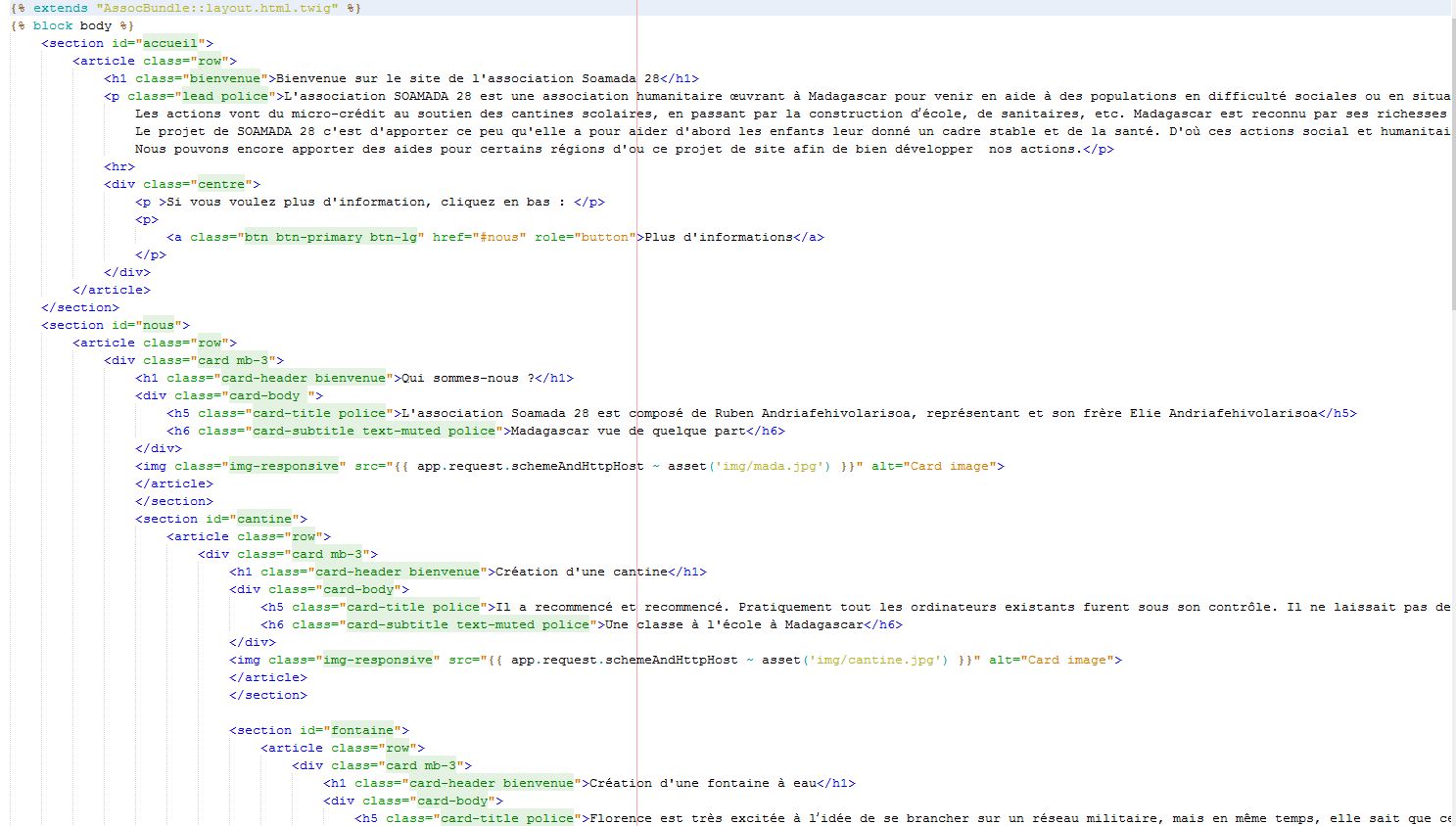


Figure 38 : La page d'accueil index.html.twig

Et voici le rendu de la page d’accueil :



Figure 39 : La page d'accueil du site

Et voici le contrôleur qui affiche la page d’accueil que j’ai intégralement codé. Car pour ce cas d’utilisation et pour les autres qui ne concernent pas l’espace membre, j’ai développé mon propre bundle appelé AssocBundle. Je l’ai intégralement développé notamment le contrôleur et la méthode indexAction() qui affiche la page d’accueil :

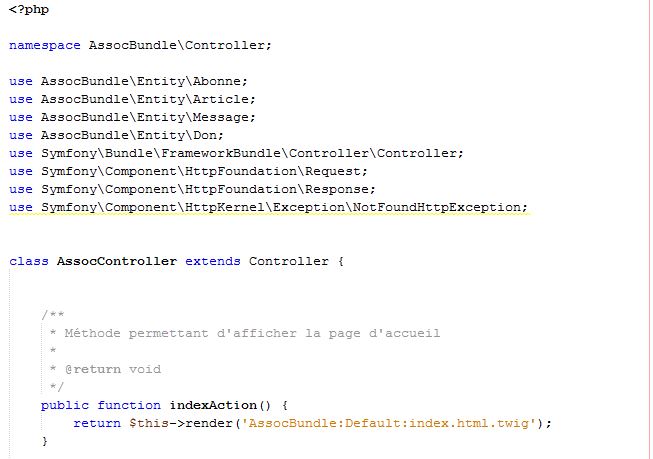


Figure 40 : Le contrôleur et sa méthode affichant la page d'accueil

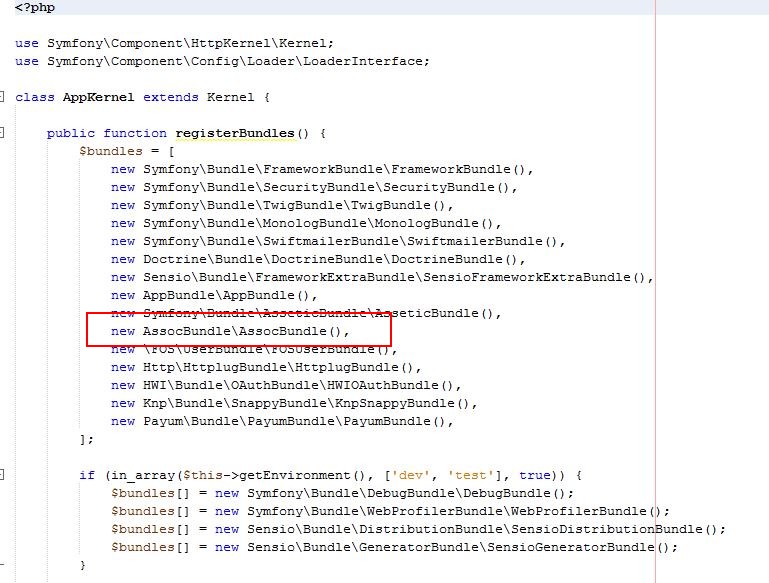


Figure 41 : Déclaration dans le fichier Appkernel.php mon bundle AssocBundle

Après j’ai effectué les tests du cas d’utilisation :

1. L’internaute non membre saisie dans le moteur de recherche les mots clés pour accéder à l’association : Ko (nécessite que le site soit déployé)
2. Le moteur de recherche affiche à l’internaute les résultats référencés de sa recherche sur l’association : Ko (nécessite que le site soit déployé)
3. L’internaute trouve avec les résultats affichés l’association qu’il recherche
4. L’internaute clique sur le lien de l’association : Ko (nécessite que le site soit déployé)
5. Le moteur de recherche lui redirige vers le site de l’association : Ko (nécessite que le site soit déployé)
6. L’internaute visite le site notamment la page d’accueil où il y a la présentation de l’association, ses projets et le formulaire de contact : Ok

Et pour conclure voici le diagramme de classe du cas d’utilisation :

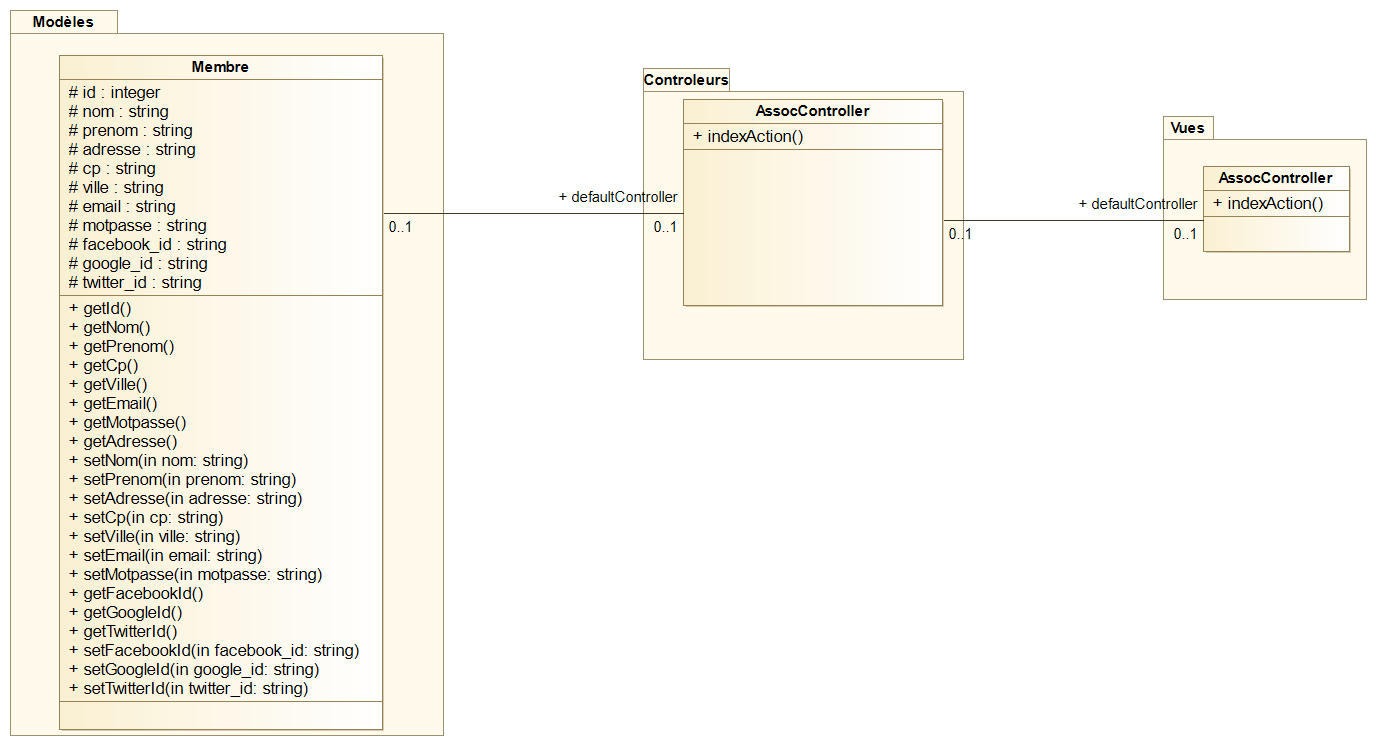


Figure 42 : Le diagramme de classe du cas d'utilisation Visiter le site

### Le cas d’utilisation n°3 S’authentifier

Maintenant place au user story « S’authentifier » dont vois le diagramme du cas d’utilisation :

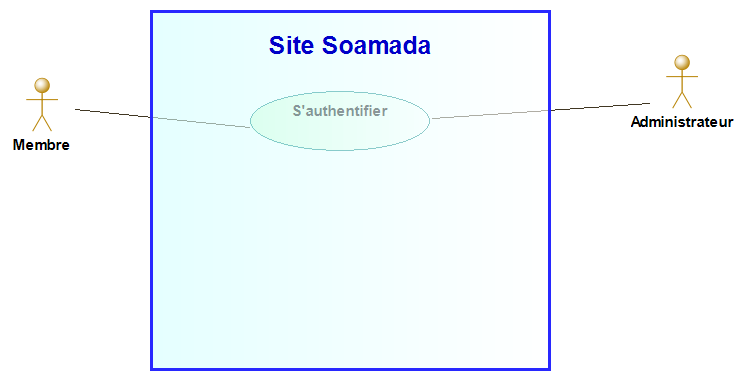


Figure 43 : Diagramme du cas d'utilisation S'authentifier

Ensuite place à la description textuelle du cas d’utilisation :

Titre : S’authentifier

Résumé : Ce cas d’utilisateur permet aux membres de s’authentifier

Acteurs principaux ou secondaires : le membre, le site

Date de création et de mise à jour : 29 mars 2019

Responsable : Moi Honoré Rasamoelina

Version : 1.0

Précondtions :

* Le site internet doit être fonctionnel
* Le formulaire de connexion doit être fonctionnel
* L’internaute doit être membre de l’association pour pouvoir se connecter

Scénario nominal :

1. Le membre clique sur le lien de connexion
2. Le site internet affiche le formulaire de connexion
3. Le membre saisit dans le formulaire son nom d’utilisateur et son mot de passe et valide ses choix
4. Le site vérifie les informations saisies par l’internaute
5. Le site informe à l’internaute de la véracité des informations saisies
6. Le site affiche un message au membre qu’il est connecté
7. Le site redirige le membre vers la page d’accueil

Scénario d’exception :

SE1 : Le membre ne connait plus ses identifiants de connexion

Le SE1 commence au point 2 du scénario nominal

Le membre ne se souvient plus de ses identifiants de connexion. Il demande alors un nouveau mot de passe en renvoyant son email dans un formulaire, il reçoit un nouveau mot de passe

Le scénario reprend au point 3 du scénario nominal

SE2 : Le membre saisit des informations invalides

Le SE2 commence au point 3 du scénario nominal

Les informations saisies par le membre sont invalides ou ne sont pas connu du site internet

Le scénario reprend au point 5 du scénario nominal

Scénarii alternatif :

* Le formulaire de connexion n’est pas fonctionnel
* La vérification pour la connexion n’a pas abouti
* La connexion internet est indisponible
* Le site internet est indisponible
* La base de données est défaillante

Post conditions :

* Le membre accède à sa page membre après avoir rempli le formulaire de connexion avec les bons identifiants

Voici le diagramme de séquence système correspondant à ce cas d’utilisation :

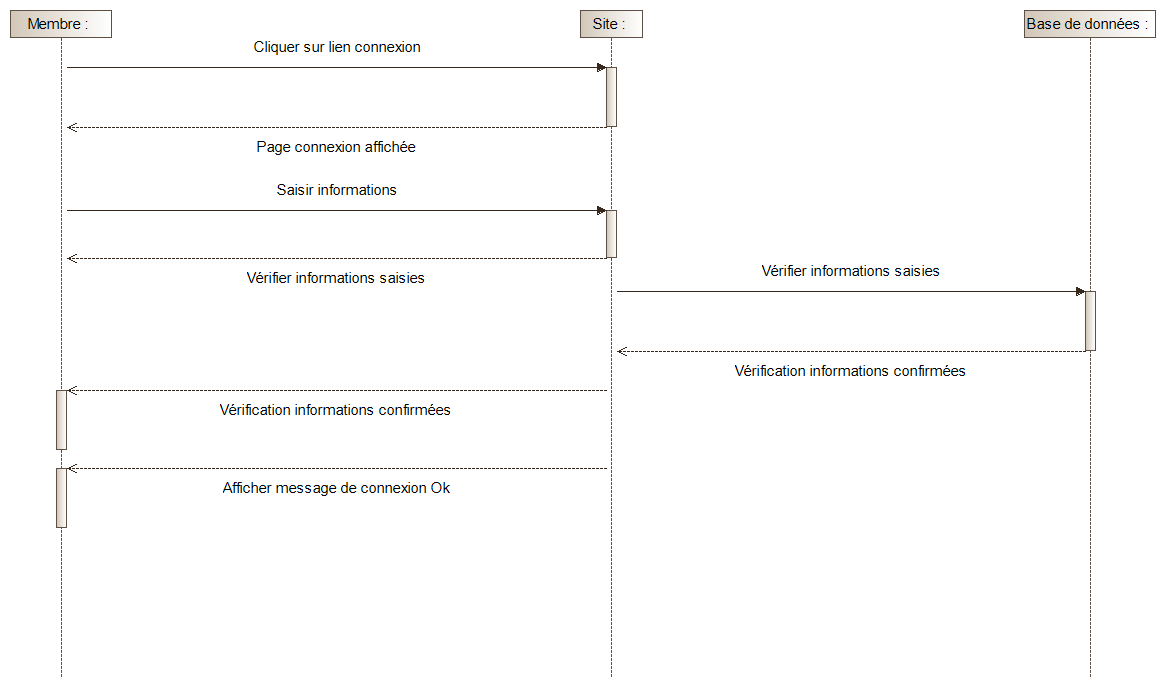


Figure 44 : Le diagramme de séquence système du cas d'utilisation S'authentifier

Ensuite, on évoque le modèle conceptuel des données de cas d’utilisation :

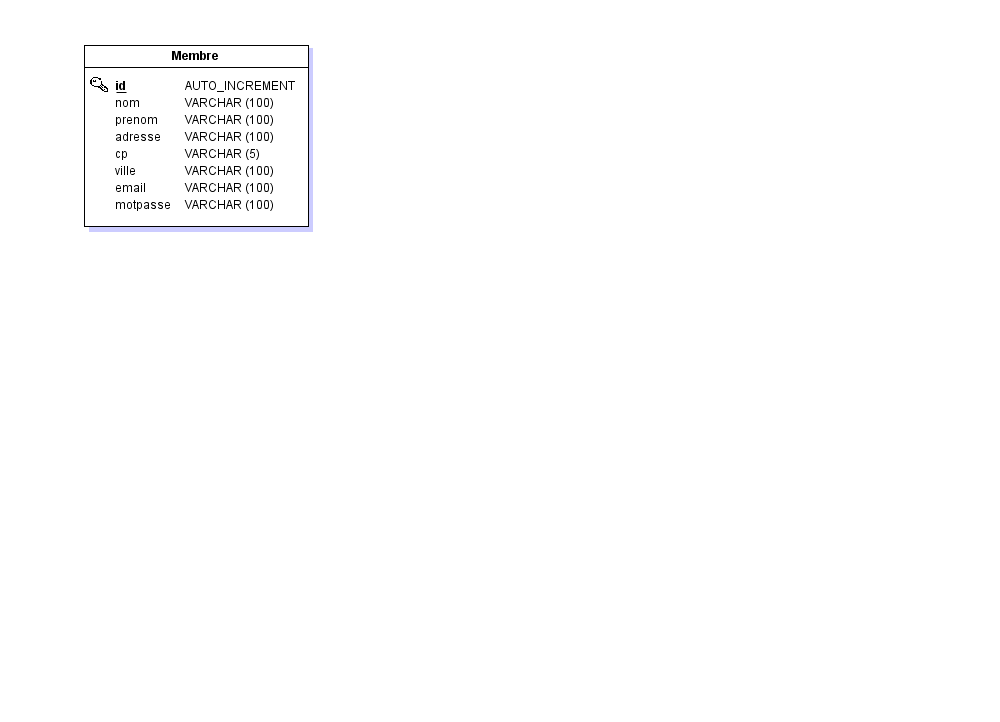


Figure 45 : MCD du cas d'utilisation S'authentifier

Et le fameux modèle physique de données de ce cas d’utilisation :

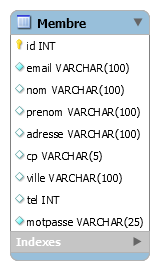


Figure 46 : MPD du cas d'utilisation S'authentifier

Voici un extrait script HTML TWIG du formulaire de connexion, ce formulaire est fourni pas le bundle FOSUserBundle mais je l’ai surchargé pour se connecter avec les réseaux sociaux et mettre en pratique le principe du oAuth :



Figure 47 : Formulaire de connexion mise en place grâce au bundle FOSUserBundle

Et voici la classe contrôleur qui effectue la connexion et qui est fourni par le bundle FOSUserBundle. Je ne l’ai pas surchargé :

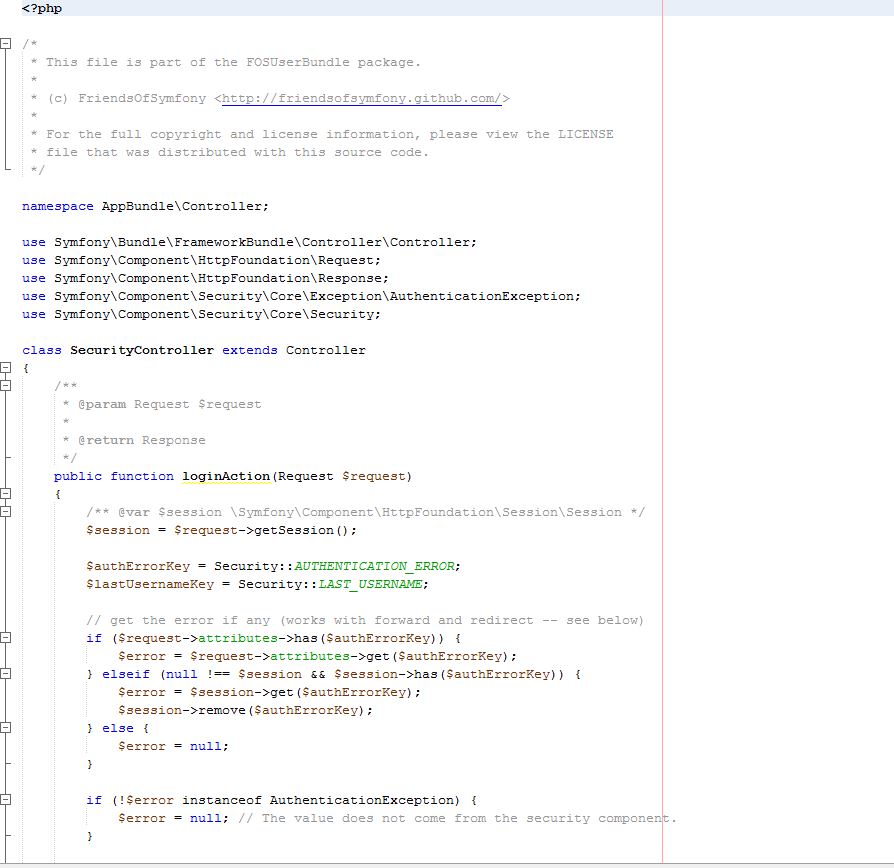


Figure 48 : SecurityController le contrôleur de FOSUserBUndle effectuant la connexion

Evidemment l’entité qui est vérifié est la table Membre crée précédemment.

Et voici le rendu du formulaire :

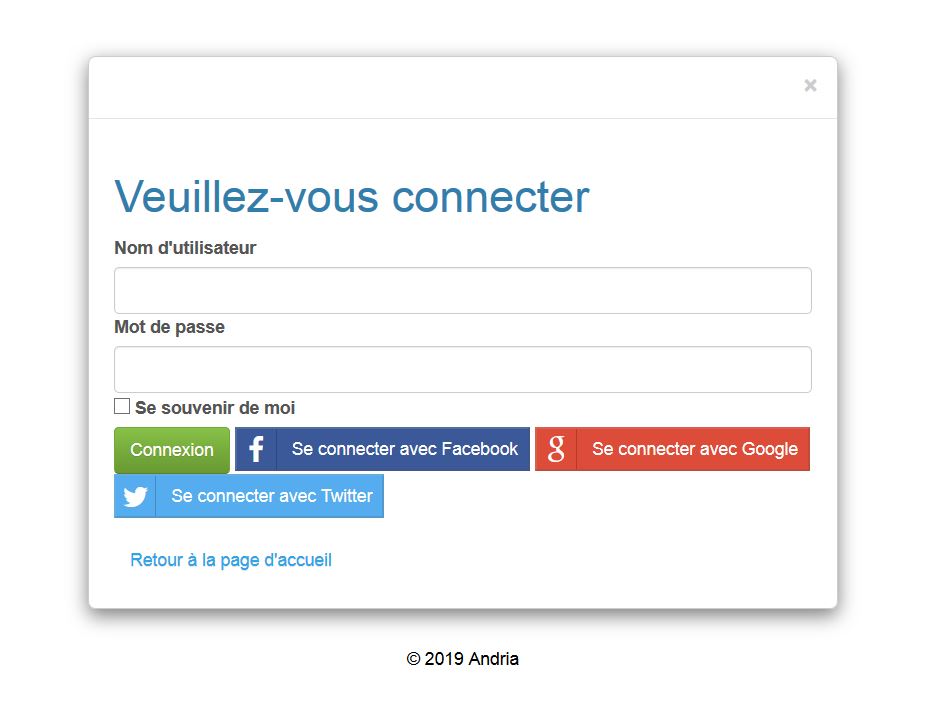


Figure 49 : Le formulaire de connexion du site

Le principe de l’oAuth sera également utilisé pour ce cas d’utilisation.

Voir le cas d’utilisation « Devenir Membre » pour plus d’explication sur le oAuth

Après j’ai effectué les tests de ce cas d’utilisation :

1. Le membre clique sur le lien de connexion : Ok
2. Le site internet affiche le formulaire de connexion : Ok
3. Le membre saisit dans le formulaire son nom d’utilisateur et son mot de passe et valide ses choix : Ok
4. Le site vérifie les informations saisies par l’internaute : Ok
5. Le site informe à l’internaute de la véracité des informations saisies : Ok
6. Le site affiche un message au membre qu’il est connecté : Ok
7. Le site redirige le membre vers la page d’accueil : Ok

Et voici le diagramme de classe de ce cas d’utilisation :

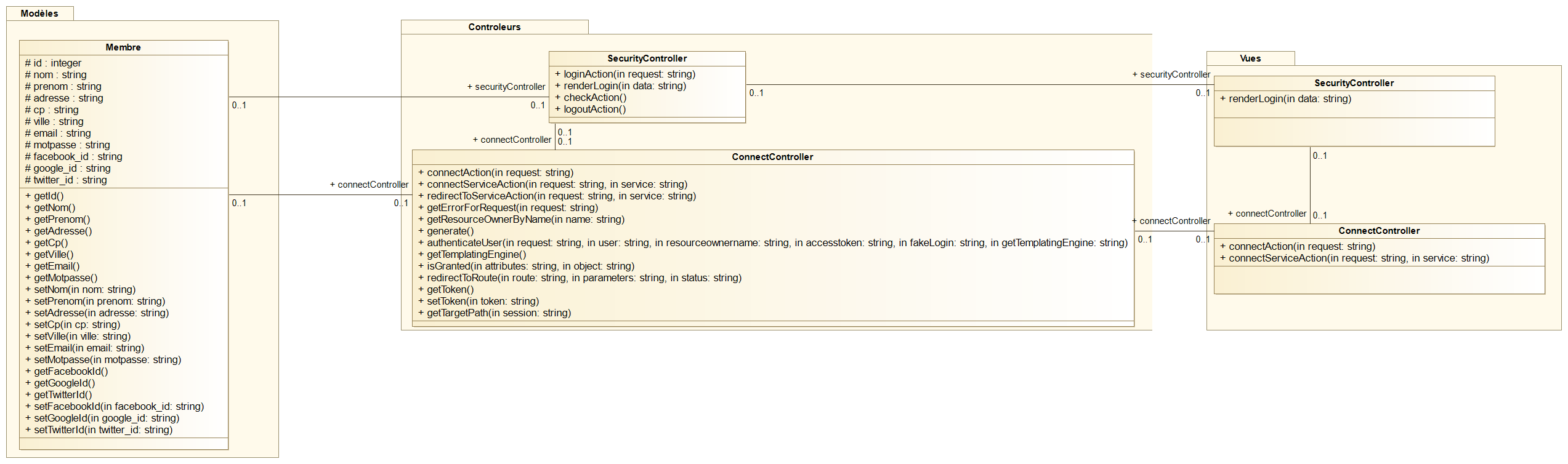


Figure 50 : Diagramme de classe du cas d'utilisation S'authentifier

Après on verra comment gérer les informations d’un membre en évoquant le cas d’utilisation « Gérer ses informations membre »

### Le cas d’utilisation n°4 « Gérer ses informations membre »

Voici le diagramme du cas d’utilisation « Gérer les informations d’un membre » :

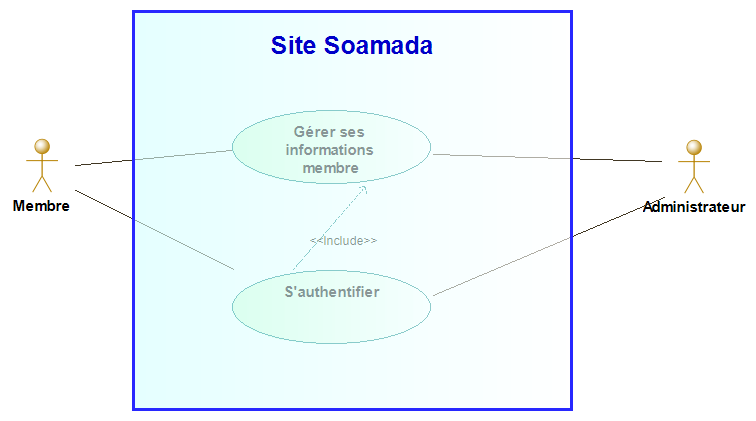


Figure 51 : Diagramme du cas d'utilisation "Gérer ses informations membre"

Comme vous le constatez, il y un lien « include » entre s’authentifier et « Gérer les informations des membres » car cela signifie que le cas « Gérer ses informations membre » inclut l’utilisation du cas d’utilisation « S’authentifier », et que l’on a besoin de ce dernier pour pouvoir utilisé le cas « Gérer ses informations membre ».

En nous allons décrire textuellement ce cas d’utilisation :

Titre : Gérer ses informations membre

Résumé : Ce cas d’utilisation permet aux membres de gérer ses informations saisies

Acteurs principaux ou secondaires : Le membre, le site

Date de création et de mise à jour : 29 mars 2019

Responsable : Moi Honoré Rasamoelina

Version : 1.0

Préconditions :

* Le site doit être fonctionnel
* Le formulaire de modification doit être fonctionnel
* Le membre doit être connecté pour accéder à sa page

Scénario nominal :

1. Le membre clique sur le lien de mise à jour de son profil
2. Le site affiche un formulaire où il y a ses informations saisies lors de son inscription
3. Le membre saisit dans ce formulaire les changements qu’il souhaite apporter en les saisissants les champs concernés. Il valide ses choix.
4. Le site vérifie les informations saisies par le membre
5. Le site informe le membre de la véracité des informations saisies
6. Le site affiche un message au membre que ces informations saisies sont bien modifiées.
7. Le site redirige vers la page d’accueil

Scénario d’exception :

SE1 : Le membre renseigne des informations erronées

Le SE1 commence au point 2 du scénario nominal

Le membre se trompe en modifiant ses informations. Il met à jour des informations qui ne sont pas en corrélation avec le type du champ (L’email dans le champ téléphone par exemple).

Le scénario reprend au point 6 du scénario nominal

Scénarii alternatif :

* Le formulaire de modification n’est pas fonctionnel
* La vérification pour la modification des informations n’a pas abouti
* La connexion à internet est indisponible
* Le site internet est indisponible
* La base de données est défaillante
* La base de données n’effectue pas la mise à jour des informations

Posts conditions :

* Le membre met à jour sa page personnel.
* La base de données a parallèlement mise à jour aussi ses informations membre.

Le diagramme de séquence système qui suit illustre le scénario nominal de ce cas d’utilisation :

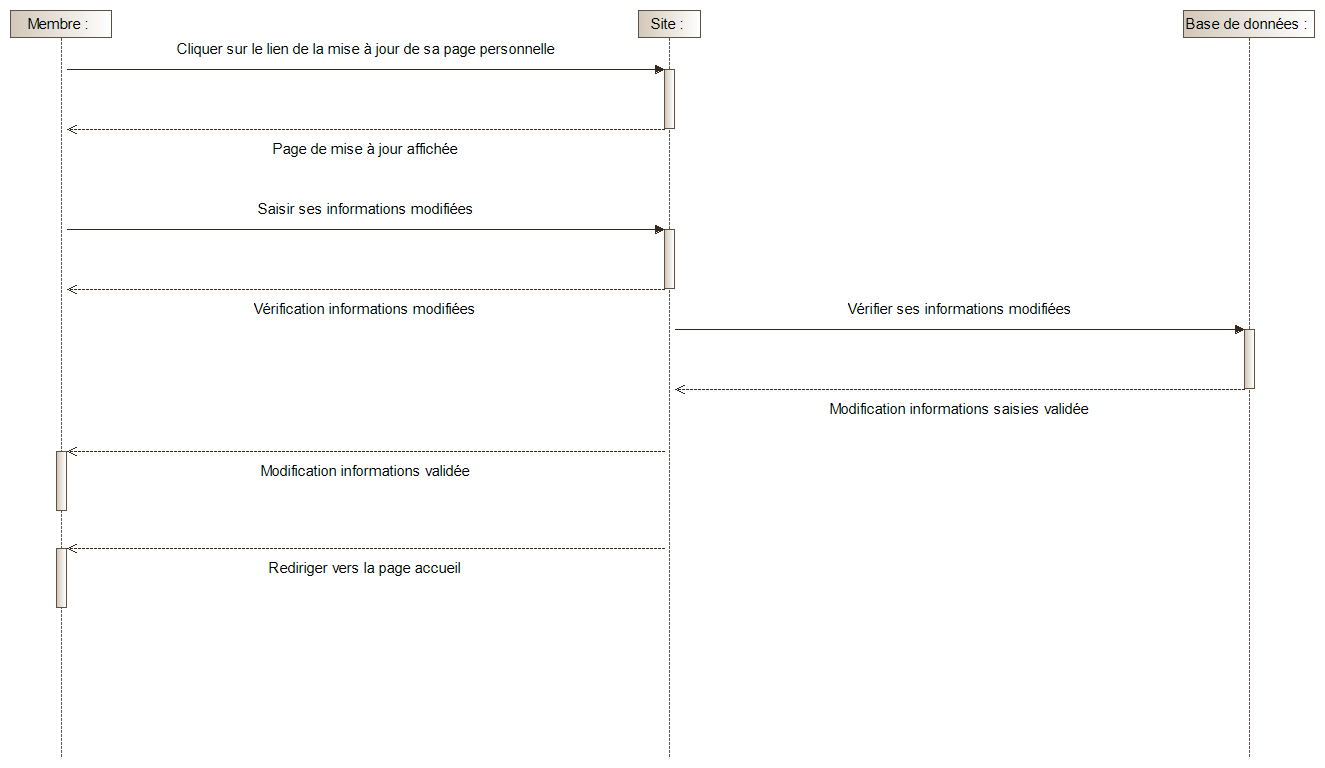


Figure 52 : Diagramme de séquence système du cas d'utilisation "Gérer ses informations membre"

Le modèle conceptuel ci-dessous illustre ce cas d’utilisation :

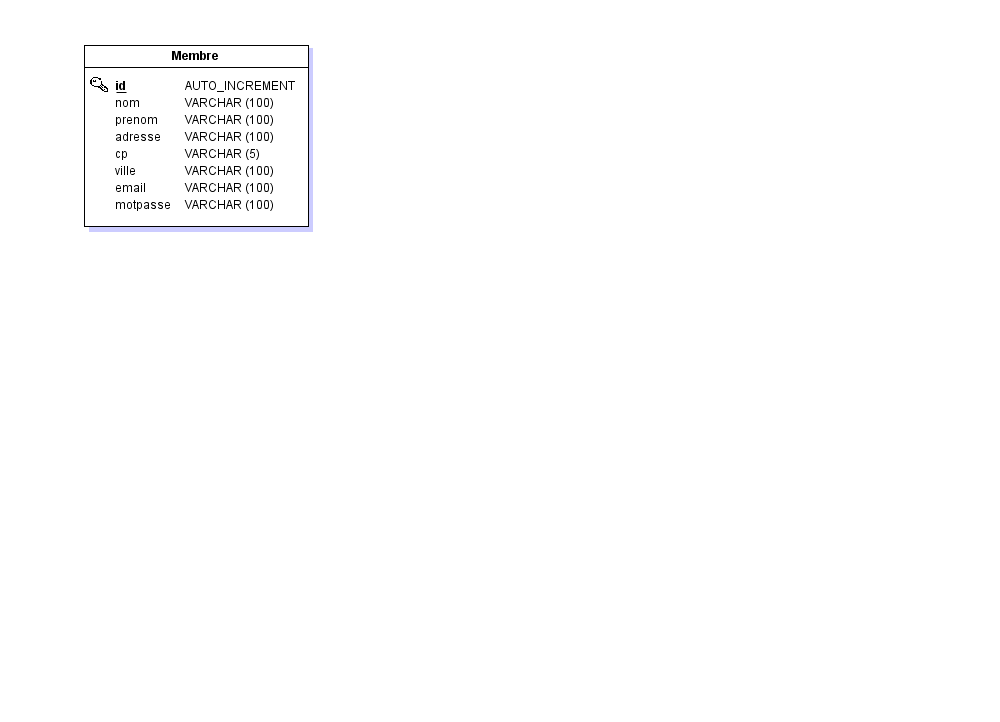


Figure 53 : MCD du cas d'utilisation "Gérer ses informations membre"

Et le modèle physique de données qui suit également :

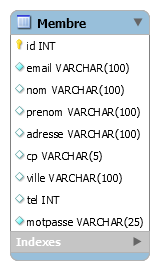


Figure 54 : MPD du cas d'utilisation "Gérer ses informations membre"

Voici en HTML TWIG la vue qui affiche le formulaire de mise à jour du profil membre, ce formulaire mixe les champs que j’ai créé moi-même et celui crée par le bundle FOSUserBundle :



Figure 55 : Le formulaire de mise à jour du profil membre

Et le rendu est le suivant :

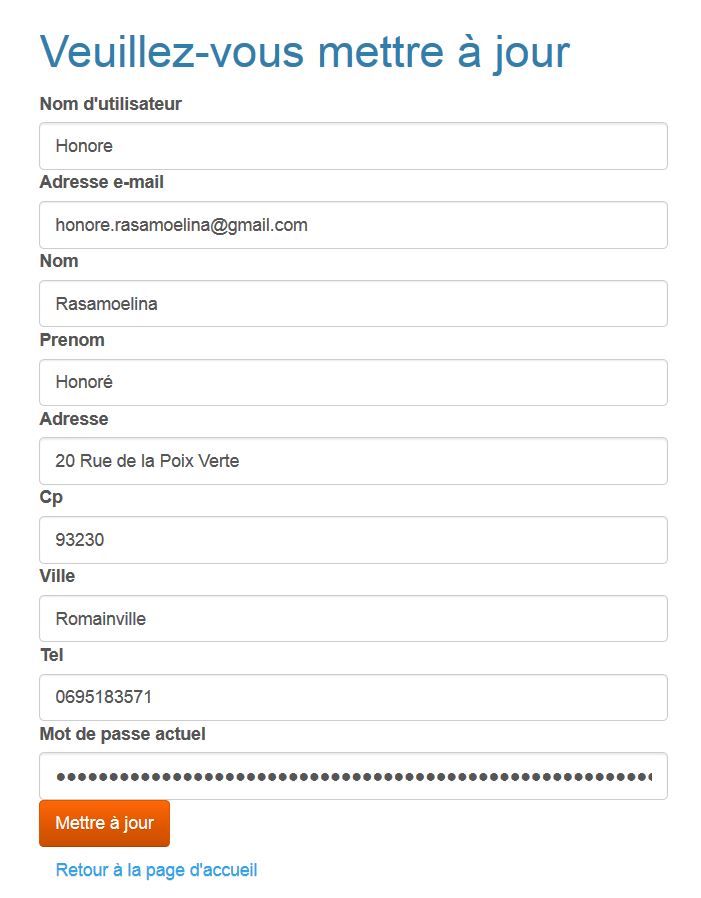


Figure 56 : Le formulaire de mise à jour d'un membre pour le cas d'utilisation "Gérer ses informations membre"