# Web social pour monnaie libre

Fabien Brisset - fabien.brisset@etud.univ-montp2.fr
Florian Galinier - florian.galinier@etud.univ-montp2.fr
Thin-Hinen Hedli - thin-hinen.hedli@etud.univ-montp2.fr
Rime Lamrani - rime.lamrani@etud.univ-montp2.fr
Oualid Manai - oualid.manai@etud.univ-montp2.fr
Quentin Philbert - quentin.philbert@etud.univ-montp2.fr
Adrien Plazas - adrien.plazas@etud.univ-montp2.fr
Clément Simon - clement.simon@etud.univ-montp2.fr

**Résumé**

Ce rapport est le compte rendu du projet *Web social pour monnaie libre* exécuté par les auteurs et proposé par Jacques Ferber pour l'unité d'enseignement *GLIN601 Projet* du sixième semestre du parcours Licence Informatique de la Faculté de Sciences de Montpellier en 2013-2014.

# Remerciements

Nous remercions tout particulièrement Jacques Ferber pour nous avoir brillamment encadré et soutenu tout le long de la réalisation de ce projet.

# Table des matières

# Chapitre 1

# Introduction

## 1.1 Le sujet

### 1.1.1 Objectif de ce projet

L'objectif de ce projet est la réalisation d'un site Web social permettant à des personnes de proposer des services, d'obtenir des services et d'échanger en utilisant des monnaies libres.

### 1.1.2 Ce qu'est une monnaie libre

Une monnaie libre est un concept économique inspiré des règles fondamentales du logiciel libre (voir [7]). Pour être considérée comme libre, une monnaie propose :

- la liberté du choix du système monétaire (la monnaie ne s'impose pas) ;
- la liberté d'utilisation des ressources (économiques et monétaires) ;
- la liberté d'estimation et de production de toute valeur ;
- la liberté d'échanger dans la monnaie (afficher, comptabiliser, échanger dans l'unité monétaire choisie).

### 1.1.3 Les SELs (réseaux sociaux autour d'une monnaie libre)

Un SEL (Système d'Échange Local) (voir [15]) est un système de partage de biens et services entre membres d'une même communauté et fondé autour d'une monnaie libre. Ces systèmes fonctionnent considérablement sur la confiance des membres de cette communauté envers la monnaie et envers leurs pairs, la localité de ces communautés aidant.

Il est fréquent que les comptes y soient tenus dans des carnets, ne soient pas à jour ou que les échanges finissent par se faire sans paiement entre connaissances.

## 1.2 Cahier des charges

### 1.2.1 Fonctionnalités attendues

Un certain nombre de fonctionnalités étaient à l'origine prévues par le cahier des charges, ainsi qu'un certain nombre de contraintes. Bien que toutes n'aient pas été implémentées, elles ont été la base de notre travail :

- intégration à un groupe, définition d'un profil (photos, etc...) ;
- définition de ses intérêts ;
- définir les services que l'on propose, leur donner un prix dans la monnaie libre associée au groupe (ou dans une autre monnaie) ;
- regroupement par intérêts ;
- évaluation des services proposés ;

— communication par sujets ou petits forums avec soutien (et remontée des informations les plus appréciées avec des techniques semblables à Reddit) ;
— partage d'informations...

### 1.2.2 Contraintes

Les contraintes étant :
— utiliser un framework ou CMS existant ;
— l'évaluation du logiciel de base devra être réalisée ;
— la possibilité de tester ce logiciel dans un groupe de personnes fonctionnant dans un JEU (Jardin d'Echange Universel), un système de monnaie libre et d'échange qui existe dans plusieurs régions, dont Montpellier.

# Chapitre 2

# Organisation du projet

## 2.1 Organisation du travail

Nous avons rapidement opté pour une méthodologie SCRUM, alternant régulièrement réunions, phases d'analyse et phases de développement, avec une version à rendre à chaque réunion.

### 2.1.1 Réunions de travail

Des réunions de travail regroupant les membres du groupe ainsi que notre encadrant Jacques Ferber se sont tenues régulièrement au LIRMM. et ont vu se dérouler de nombreuses présentations d'outils et de propositions de directions à prendre. Durant la phase de développement ces réunions ont été espacées de façon bimensuelle, afin de laisser le temps au groupe de réaliser une nouvelle version, et avaient pour but principal de définir l'avancement du projet et quelle direction il devait prendre aussi bien à court terme qu'à long terme.

D'autres réunions plus informelles se sont déroulées entre tout ou partie des membres du groupe en de divers lieux comme la Faculté Des Sciences ou le LIRMM, chacune répondant à un besoin spécifique.

### 2.1.2 Répartition des tâches

À la fin de chaque réunion, les membres choisissaient les tâches dont ils souhaitaient s'acquitter, ou prenaient les tâches restantes ainsi le travail était réparti et tous les membres du projet avaient des objectifs à réaliser pour la réunion suivante.

### 2.1.3 Planification du développement

Le développement a été réalisé intelligemment, en effet, la méthode agile (fortement réputée dans le monde de l'entreprise) semblait être la plus propice pour la réalisation de ce projet. C'est pourquoi nous l'avons utilisée. Ainsi, environ toutes les deux semaines (après un certain nombre de changements majeurs) une nouvelle version opérationnelle naissait. Le site est désormais hébergé, il est donc accessible en ligne (voir [8]).

### 2.1.4 Élection d'un chef de projet

L'élection d'un chef de projet a eu lieu lors de la première réunion. Deux volontaires se sont exprimés et, après discution, il a été décidé de confier ce rôle à Florian et Adrien. Ils sont ainsi devenus les chefs de projet et ont eu pour objectifs de prendre en main la gestion des différents acteurs de ce projet mais aussi de le mener à bien.

### 2.1.5 Gestion du groupe d'étudiants

La communication entre membres du groupe s'est faite via de divers moyens :

— de vive voix lors des réunions, de rencontres informelles, ou par téléphone ;

— par messagerie écrite : par SMS, échanges d'e-mails groupés ;
— par sites web : par Facebook, le gestionnaire de tâches Producteev (voir [9]) ou lors de pull requests sur Github.

## 2.2 Choix des outils de développement

### 2.2.1 Analyse des outils disponibles

Le premier élément sur lequel nous avons dû nous pencher fut le choix d'utilisation d'un CMS (WordPress/Drupal) ou d'un Framework (Symfony2/CakePHP).

Ainsi nous avons dû définir les avantages et les inconvénients de chacun de ces outils de programmation.

**Les systèmes de gestion de contenu (CMS)**

WordPress

WordPress (voir [14]) est un CMS libre et gratuit et également l'un des plus connus. Bien qu'il soit principalement utilisé comme moteur de blog, les fonctionnalités de Wordpress lui permettent de supporter n'importe quel site Web. Il permet à plusieurs auteurs de publier des billets, lesquels sont classés par date et par catégories.

**Avantages** L'installation de WordPress est aisée, et sa paramétrabilité offre aux utilisateurs avertis de multiples possibilités pour transformer leur blog en une boutique e-commerce, un portfolio, un site plaquette, etc... De plus, des thèmes sont prêts à l'emploi et des plugins divers sont disponibles depuis l'interface WordPress.

**Inconvénients** Du fait de ses nombreuses fonctionnalités, WordPress est un logiciel de blog plutôt destiné à des utilisateurs avancés, ayant un minimum de connaissances des systèmes de gestion de contenu. Malgré la clarté de son interface, la profusion de menus et ses possibilités en matière de configuration peuvent rebuter des utilisateurs débutants. Cet outil ne semble pas utilisable pour un projet d'une telle envergure, à 8 collaborateurs.

Drupal

Drupal (voir [11]), lui aussi est un CMS libre et gratuit.

Il s'organise autour d'unités de contenus minimales, appelées « noeuds », qui correspondent à différents éléments : article, blog, commentaire, formulaire de saisie, image ou galerie d'images, sondage, page de wiki, etc...

**Avantages** La structure modulaire et évolutive de Drupal permet d'ajouter de nombreuses fonctionnalités, rendant possible la réalisation de nombreux projets de tailles différentes, notamment dans les variantes suivantes :

— publication Web (création de plateformes et sites communautaires sur Internet) ;
— création de systèmes de gestion des connaissances (notamment via une classification par catégories des contenus) ;
— création de groupes de travail (intranet).

Drupal contient un peu plus de 6 000 « modules communautaires » utilisables.

**Inconvénients** Contrairement à d'autres CMS (comme Wordpress), Drupal n'est pas un outil « clé-en-main » et, du fait de sa structure modulaire et hautement adaptable, son utilisation nécessite l'intervention d'un développeur expérimenté.

Comme pour WordPress, cet outil ne semble pas utilisable pour un projet d'une telle envergure, à 8 collaborateurs.

### Les frameworks


CakePHP (voir [10]) est un framework de développement rapide pour PHP gratuit et open-source. C'est un ensemble de briques élémentaires pour les programmeurs qui créent des applications Web. L'objectif principal d'un framework comme CakePHP est de permettre de travailler de manière rapide et structurée, sans toutefois perdre en flexibilité.

**Avantages** Disposant d'une communauté active, sa documentation se trouve en français, de même que de nombreux tutoriels et notamment des tutoriels vidéo.

Il est compatible avec les versions 4 et 5 de PHP, dispose également de fonctions CRUD (Create, Read, Update, Delete) intégrées pour les interactions avec la base de données (Scaffolding) et de fonctions de génération de code via sa console.

Comme la plupart des frameworks PHP, CakePHP utilise une architecture MVC (Model, View, Controller), et des URLs propres et personnalisables sont créées grâce à un système de routes.

Le core de CakePHP réalise une première validation des données mais une seconde validation de celles-ci reste indispensable afin d'assurer la sécurité du site ainsi que de ses données.

Aussi, CakePHP au niveau « Vue » de la structure MVC présentée antérieurement utilise un système de template rapide et souple (syntaxe PHP avec des « Helpers »).

Ce framework fonctionne sur n'importe quelle arborescence de site web, avec peu de configuration.

**Inconvénients** ORM peu véloce, remplaçable avantageusement par des procédures stockées. Bien qu'il soit en plein essor, il est encore peu utilisé par les entreprises.

Symfony2 (voir [13]) est un framework MVC libre écrit en PHP5. En tant que framework, il facilite et accélère le développement de sites et d'applications internet et intranet.

**Avantages** Symfony2 est ambitieux, actuellement, ce framework est en relation avec le CMS Drupal afin de sortir Drupal8, un mélange de Symfony2 et de Drupal (de Framework et de CMS).

Au niveau des « Vues », Symfony2 utilise le moteur de template Twig qui est réputé pour sa simplicité tant dans l'accès à ces vues que dans la rédaction de celles-ci.

Symfony2, lui aussi, utilise le CRUD, celui-ci est réalisé par Doctrine qui est un outil très puissant et très simple d'utilisation. Ainsi, les interactions avec la base de données sont aisées et très rapides.

De très bons tutoriels se trouvent partout sur le web, notamment sur OpenClassroom.

À ce jour, Symfony est le framework le plus utilisé au sein des entreprises françaises.

Symfony2 utilise lui aussi le système de routes, cette fonctionnalité est des plus pratiques car elle permet d'associer un nom de route à une URL et une action du contrôleur, ainsi, cette route est accessible très facilement dans les contrôleurs ou encore dans les vues Twig. Ces URLs sont définies par le programmeur.

La configuration de Symfony2 est très simple et la mise en ligne du site achevée l'est également, après avoir trouvé un hébergeur compatible avec Symfony2.

**Inconvénients** Ce framework est d'après de nombreux sondages plus difficile à prendre en main que CakePHP et nécessite plus de temps pour l'apprentissage des fonctionnalités du framework.


FuelPHP (voir [12]) est un framework PHP très récent car il est apparu en 2011 dans sa version 1.0.

**Avantages** Il est très utilisé en Amérique du Nord (surtout dans les entreprises Canadiennes) et se substitue à l'un des anciens leaders Nord-Américain : Kohana.

FuelPHP utilise une structure HMVC (Hierarchical Model View Controller) ce qui inclut une arborescence de fichiers en cascade (inspirée du framework Kohana). Son principe est d'organiser l'arborescence des répertoires partiellement à l'image des espaces de noms dédiés aux classes.

FuelPHP utilise lui aussi un système de routes.

FuelPHP comprend des moteurs de template, à savoir Stags (moteur de template spécifique à FuelPHP) et Mustache, de plus, FuelPHP fournit les pilotes pour les moteurs de template Markdown, Smarty, Twig, Haml, Jade et Dwoo16.

Au niveau sécurité, FuelPHP encode les caractères non alphanumériques lors de la génération des pages web, fournit les protections contre les attaques des types CRSF et cross-site scripting, fournit une fonction de filtrage des variables super-globales, et protège des attaques de type injection SQL20.

FuelPHP implémente aussi un ORM (Object Relationnal Mapping), un Scaffolding pour l'interaction avec la base de données ainsi que UnitPHP qui est un outil permettant de réaliser des tests unitaires afin de vérifier que les fonctions réalisent bien leur travail.

**Inconvénients** Ce framework est récent et est peu connu ou du moins peu utilisé par les programmeurs français. De plus, sa complexité demanderais une plus longue phase d'apprentissage que ses concurrents.

**Framework CSS/Javascript** Ensuite, la question du style à utiliser s'est posée et l'utilisation d'un Framework CSS/Javascript a été choisie (Bootstrap Twitter).


Bootstrap, de nom officiel Twitter Bootstrap, est un framework CSS/JS sous licence Apache qui est une bibliothèque d'outils facilitant la création de sites et d'applications web. La version 3.1.1 a été choisie pour notre site.

Voici un exemple d'utilisation de Bootstrap avec la page d'accueil du site. Elle dispose du modèle « carousel » de Bootstrap qui utilise à la fois des éléments CSS pour, par exemple, l'affichage de boutons, et de fonctions Javascript pour le défilement d'un slider à un autre.

### 2.2.2 Choix du langage

Le choix du langage ou plutôt des langages est apparu naturellement lors de la présentation des différents outils de programmation.

Le PHP est indispensable ainsi que le CSS pour la mise en forme.

Quant au HTML, il sera utilisé au sein des templates Twig ce qui modifie légèrement la syntaxe car Twig ajoute des outils intéressants au niveau des vues.

Du JavaScript et du JQuery semblent utiles pour donner de la vie au site, notamment pour réaliser une page d'accueil dynamique, réactive et attrayante et pour la gestion des droits au sein des groupes ou du site d'une façon plus générale.

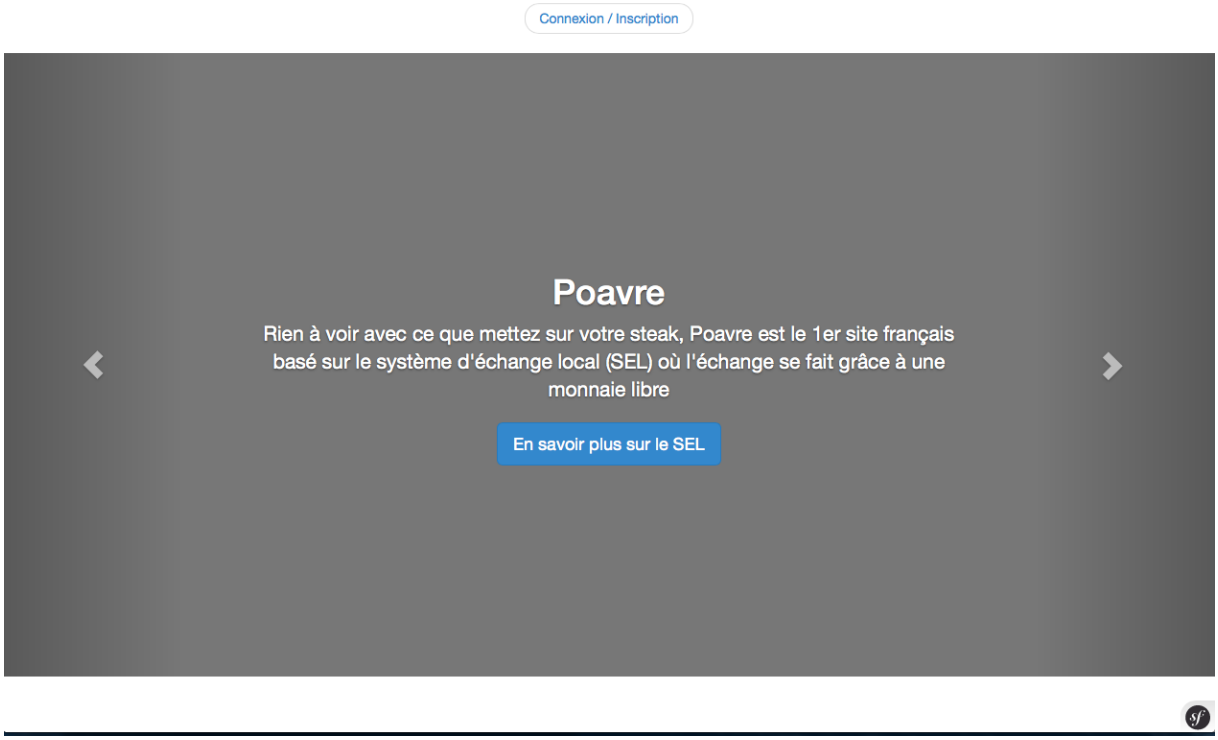



FIGURE 2.1 – Page d'accueil du site

Au niveau de la langue, nous avions décidés initialement de réaliser un site cent pour cent anglophone (programmation, commentaires et affichage en anglais), puis finalement, après modification du code réalisé, nous sommes venus à une programmation en anglais, des commentaires en anglais mais un affichage en français car les utilisateurs finaux sont français.

### 2.2.3 Choix du framework

Tout d'abord, nous avons établis qu'un framework serait plus judicieux pour la réalisation de ce projet c'est pourquoi les CMS présentés n'ont pas été retenus.

Aussi, nous voulions un framework puissant, mettant à notre disposition des fonctions utilisables pour notre projet afin de nous faciliter la programmation. Nous recherchions également un framework que nous pourrions utiliser à court terme, dans notre vie professionnelle.

CakePHP étant peu utilisé dans les entreprises françaises il n'a pas retenu notre attention. Il en va de même pour FuelPHP qui n'est quasiment pas utilisé en France. Finalement et avec enthousiasme Symfony2 a été choisi à l'unanimité pour ses performances, ses tutoriels mis à disposition (voir [16]) et par sa forte utilisation au sein des entreprises françaises.

### 2.2.4 Choix du serveur

Au niveau de l'hébergement du site web il nous fallait un hébergeur acceptant une version récente de PHP, qui intégrait facilement des projets Symfony2 et qui donnait les pleins pouvoirs sur le projet de l'extérieur.

Nous avons tout d'abord pensé que nous pourrions héberger le site sur le serveur possédé par notre tuteur mais ce dernier ne s'est pas révélé compatible avec notre projet. Aussi, l'un des membres du projet ayant réalisé un projet avec Symfony2 antérieurement, nous avons décidé d'héberger notre site sur le même serveur à savoir celui de l'hébergeur Hostinger (voir [5]).

Ainsi, nous disposons d'un accès FTP (File Transfert Protocol) pour le transfert des fichiers vers le serveur, d'un accès à PHPMyAdmin qui stocke la base de données, d'une adresse mail propre au projet associée à l'hébergeur et de divers outils pratiques mais pas toujours utiles.

Ce serveur n'est pas dédié pour des raisons pécuniaires, il est donc partagé, mais les performances sont tout de même très bonnes et le site est accessible en tout temps.

### 2.2.5 Choix du gestionnaire de projet et du gestionnaire de versions

#### Le gestionnaire de projet

Un outil de gestion de projet s'est révélé indispensable pour une réalisation efficace de ce projet. Il semblait important que celui-ci permette de gérer les différentes tâches de façon lisible et ordonnée.

Notre tuteur nous a présenté Producteev (voir figure 2.2), un outil de gestion de tâches en ligne et gratuit nous permettant de créer des tâches en spécifiant les collaborateurs leurs étant assignés ainsi que des « suiveurs » , afin que ces derniers soient informés par mail de leur avancée. Une échéance et une priorité peuvent être apposées sur les tâches et des sous-tâches qui peuvent également être commentés.

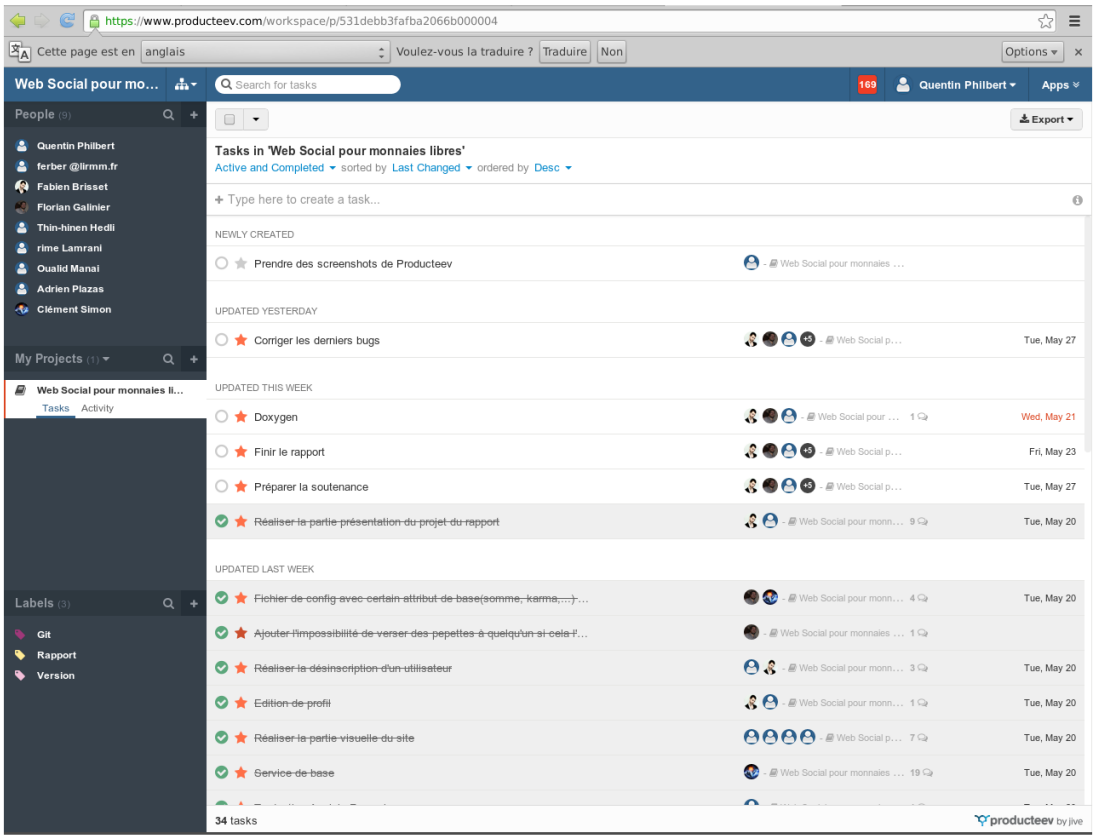



FIGURE 2.2 – Capture d'écran de Producteev

Le nombre de collaborateurs sur ce gestionnaire de tâche étant illimité, cet outil avec ses fonctionnalités est celui que nous avons choisi pour ce projet. En effet, nous ne nécessitions pas plus de fonctionnalités que celles-ci, nous n'avons donc pas cherchés d'autres outils pour la gestion des tâches du projet.

#### Le gestionnaire de versions

Un gestionnaire de versions semblait nécessaire afin de collaborer efficacement. Le gestionnaire de version distribué Git a rapidement été proposé, en association à Github (voir [4]) pour l'hébergement du code du projet (voir [1]). Après analyse des avantages et inconvénients par rapport à un système de

gestion de versions centralisé comme SVN, notamment sur l'utilisation des branches, Git et Github ont été retenus.

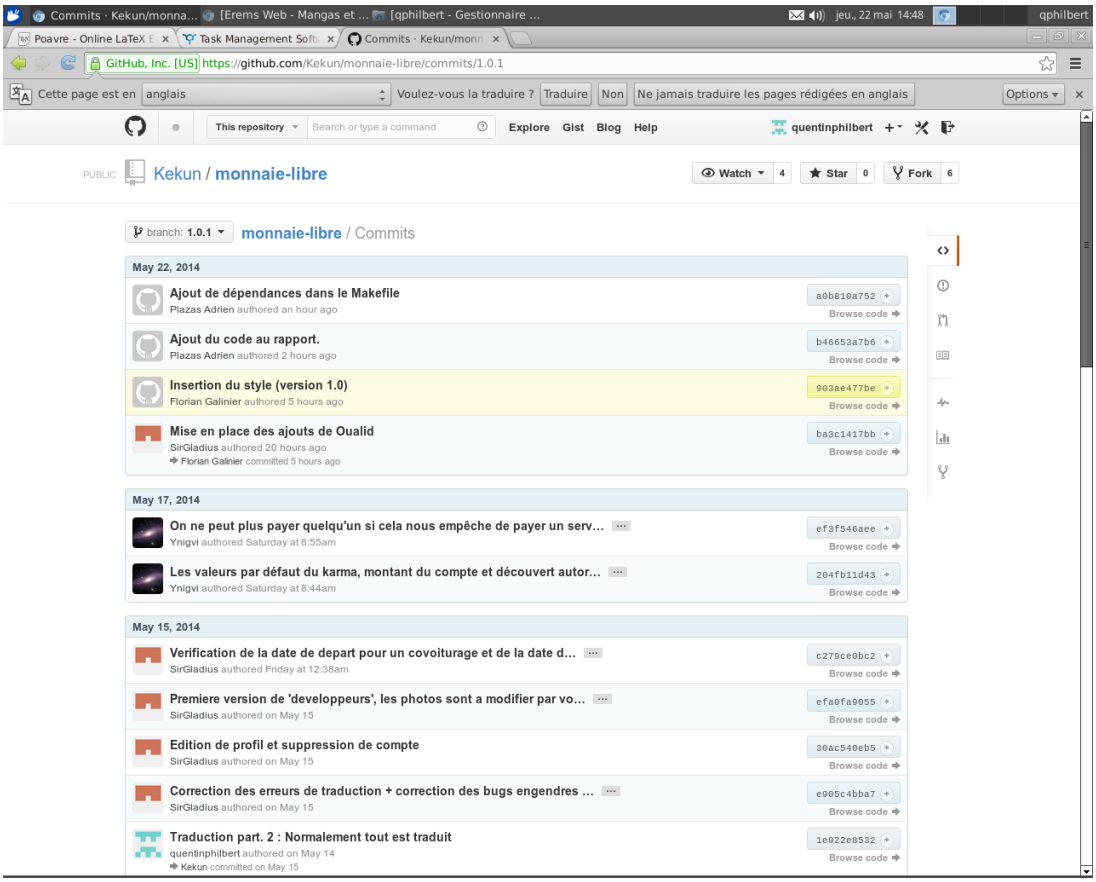



FIGURE 2.3 – Capture d'écran de GitHub

**Avantages** Git est un gestionnaire de version très utilisé et il est apprécié pour sa robustesse, ses performances, sa praticité et sa capacité de mise à l'échelle (Git gère les versions du noyau Linux, une énorme base de code avec des milliers de collaborateurs). De plus, cet outil est utilisable en lignes de commande sur de très nombreux systèmes d'exploitation (de Windows à Haiku en passant par Android).

**Inconvénients** Il demande à l'équipe d'apprendre de nombreux nouveaux concepts et son utilisation sur Windows peut être laborieuse, de par son interface en ligne de commande.

### 2.2.6 Choix des outils de documentation

Doxygen (voir [2]) est apparu comme un choix évident pour la documentation :
- c'est un logiciel libre, solide et puissant ;
- il n'est pas spécifique à un langage et fonctionne bien avec PHP ;
- il est assez simple à appréhender ;
- sa syntaxe est simple et lisible telle quelle comme documentation dans le code, le rendant assez peu intrusif.

Il répond parfaitement à nos besoins de documentation et il n'est pas étonnant qu'il soit devenu un leader dans son domaine.

# Chapitre 3

# Analyse

## 3.1 Analyse des besoins

L'objectif initial était de réaliser un site communautaire, permettant à ses utilisateurs de proposer des services entre eux par l'échange d'une monnaie libre a travers un site épuré et agréable. Il a été nécessaire que l'on se réunisse plusieurs fois, afin de donner forme a ce projet, d'en dessiner les contours, de définir les tenants et les aboutissants. En effet, il était primordial de savoir exactement ce que ce site doit apporter. En quoi est-il intéressant ? A qui s'adresse-t-il en particulier ? Que propose-t-il de plus ou de différent par rapport aux sites existants ? Pourquoi, in fine, l'utilisation d'une monnaie libre ?

Ce n'est que par cette analyse préalable, que nous avons pu identifier correctement les tâches à réaliser, les limites de notre projet et les éventuelles améliorations futures, dans une perspective d'évolution.

La première étape, s'est portée plus sur le fond que sur la forme ou la technicité. Il s'agissait de se donner un maximum d'idées sur ce que le site devait proposer. Vis à vis de l'objectif initial, il paraissait important de spécifier au mieux les services auxquels se rattache notre projet. C'est à dire, ce qu'il permettait de faire pour ses utilisateurs. Pour nous il était intéressant qu'un utilisateur puisse à la fois bénéficier de services et/ou en proposer. L'utilisateur de notre site devait donc avoir la possibilité, à sa guise, d'être un producteur et/ou un consommateur de services. Il devait être maître, non seulement, du choix du service qu'il désire (évident) mais aussi du type et de la hauteur de sa rétribution pour le service qu'il proposerait.

Il apparu assez évident que les types d'échanges se porteraient essentiellement sur des services à la personne, d'aide et de partage (type covoiturage, couchsurfing et troc). À priori, un site comme le Bon coin (voir [6]) le permettait déjà. Cependant, c'est là que l'idée de monnaie libre nous distinguait de l'existant. Jusqu'à présent, le principal modèle de rétribution d'un service était celui de l'argent. À ce modèle encore largement prédominant, apparaissait de plus en plus des réponses de substitution. La monnaie libre permettrait de générer des échanges de services entre individus sans argent réel. Ce projet s'inscrit donc dans une perspective d'utilité sociale par la proposition d'un modèle alternatif d'échange de services.

## 3.2 Analyse de l'existant

Des SELs (Système d'Echange Local) existent dans plusieurs régions de France sous forme de « réunions » où les différents usagers peuvent communiquer et échanger leurs services. Ce type d'échanges ne semble pas au goût du jour, en effet, l'Informatique a pris une place prépondérante dans notre système, dans notre mode de vie. L'idée de proposer ces échanges par le biais de l'outil informatique est donc d'un enjeu important car elle en permet une adaptation moderne et pratique. Un SEL en ligne existait déjà, « Flowplace » (voir figure 3.1), mais il ne semblait pas tout à fait effectif, il a, semble-t-il, d'ailleurs fermé à ce jour. Nous étions donc libre d'imaginer comme nous le souhaitions notre site.

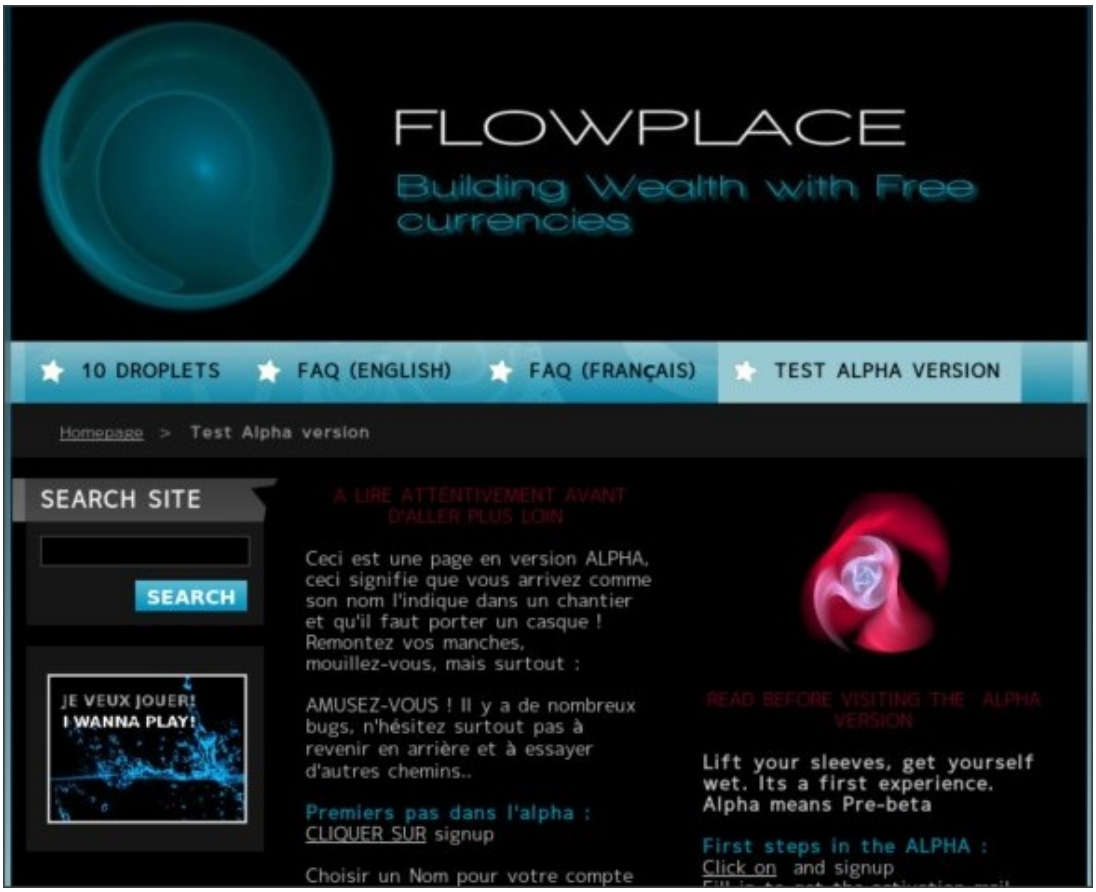



FIGURE 3.1 – Flowplace

### 3.2.1 Analyse préalable

Lors de l'analyse, le référentiel utilisé fut le point de vue de l'utilisateur de notre site. Il fut décidé d'identifier un utilisateur par son nom, son prénom et son pseudo. Il fut également choisi d'attribué à l'utilisateur une grandeur quantifiable nommée Karma, en référence à l'ensemble des bonnes et mauvaise actions qu'il aura réalisé au cours de son activité sur le site. Cette grandeur peut s'apparenter à une note de satisfaction, attribuée par les autres utilisateurs auxquels il aurait rendu service. L'idée de proposer des « missions » aux utilisateurs fut aussi abordée, mais ne fut finalement pas retenue pour le projet final. Ces « missions » devaient être un ensemble des services qu'un utilisateur devait effectuer ou proposer, recevant en échange des récompenses monétaires et/ou en karma. Nous abordions également le thème de la « monnaie ». Un utilisateur devait pouvoir choisir de « payer » ou de se faire « payer » dans la monnaie libre du site, complètement décorrélée de quelconque monnaie existante. Cela devait s'apparenter à une quantité de « points » qui permettrait l'échange de prestations entre utilisateurs.

Au départ, un nouvel utilisateur devait posséder un Karma neutre et une quantité de monnaie qui pourraient fluctuer au cours de son activité intra-site.

Chaque utilisateur devait également être attaché à un compte unique. Ce compte enregistrerait les diverses transactions et calculerait dynamiquement le solde restant, à l'instar d'un compte bancaire.

L'idée de groupe fut également abordé. Chaque membre pouvaient faire partie d'un ou plusieurs groupes. À ces groupes devait également être rattachée une liste de services.

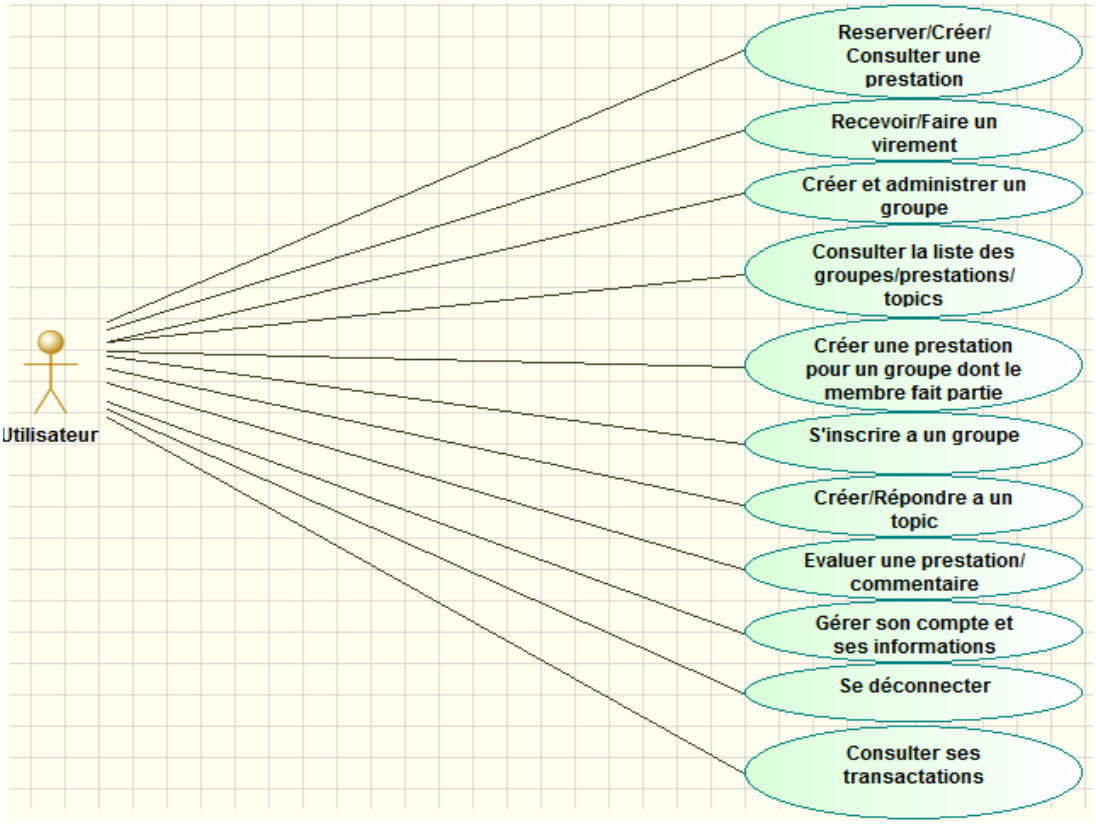



FIGURE 3.2 – Diagramme de cas d'utilisation d'un utilisateur

## 3.3 Diagrammes de cas d'utilisation

### 3.3.1 Cas d'utilisation : Membre

A la figure 3.2 nous pouvons constater les actions qui sont disponibles pour un membre. Un membre peut donc réserver, créer ou consulter une prestation, faire ou recevoir un virement, créer et administrer un groupe de membres, consulter la liste des groupes, prestations et topics, créer une prestation pour un groupe dont le membre fait partie, s'inscrire à un groupe, créer et répondre à un topic, évaluer un topic ou une prestation, gérer son compte et ses informations et, enfin, se déconnecter.

### 3.3.2 Cas d'utilisation : Modérateur

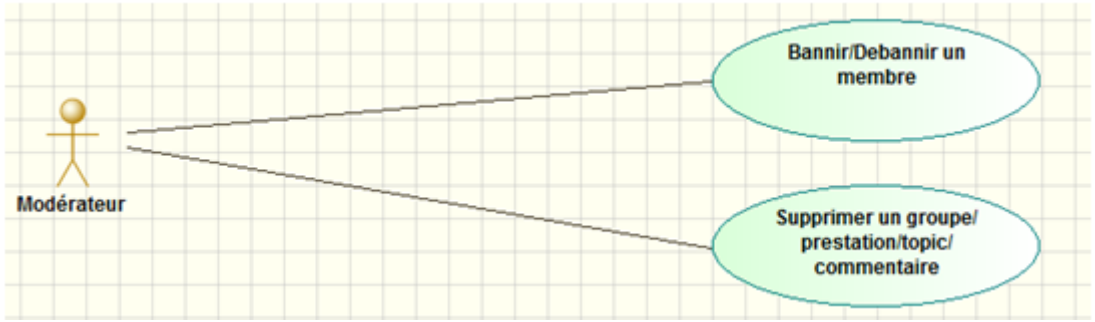



FIGURE 3.3 – Diagramme de cas d'utilisation d'un modérateur

A la figure 3.3 nous pouvons constater les actions qui sont disponibles pour un modérateur. Un

modérateur a pour responsabilité de veiller à ce que les services soient corrects, qu'il n'y ait pas d'abus au niveau des interactions entre les utilisateurs. Pour mener cette tâche à bien, un modérateur peut bannir ou débannir un membre, supprimer un groupe, une prestation ou encore un topic.

## 3.4 Diagrammes de classes

### 3.4.1 Diagramme de classe prévisionnel

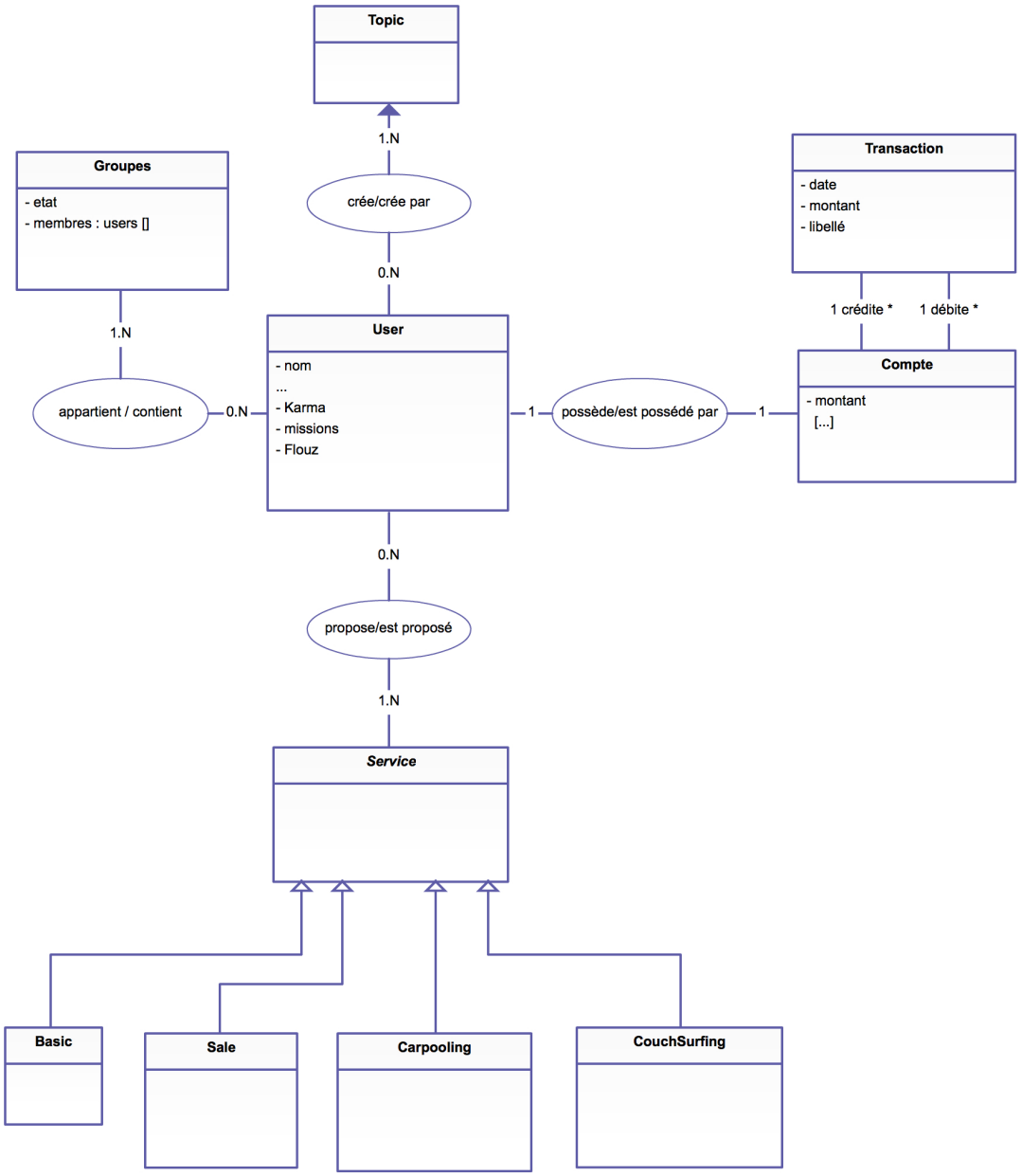



FIGURE 3.4 – Diagramme de classe prévisionnel

La figure 3.4 représente la modélisation du système au niveau des classes que nous avions prévues avant de commencer à coder. Il a ensuite bien évolué...

### 3.4.2 Diagramme de classe final

Le diagramme qui apparaît à la figure 3.5 représente nos classes d'un point de vue conceptuel. C'est le diagramme de classe final, il implémente actuellement notre système.

Notre diagramme prévisionnel a évolué vers son état final en ce diagramme. Cette fois-ci les services se divisent en quatre grandes catégories :

**Couchsurfing** qui correspond à trouver/proposer un logement temporaire ;

**Covoiturage** qui permet un déplacement à plusieurs partageant ainsi les coûts inhérents au voyage ;

**Ventes** qui, à l'instar de sites comme Le Bon Coin ou encore Ebay, permet des échanges d'objets ;

**Basique** qui correspond à tout ce qui est autre que les trois catégories pré-citées.

La possibilité pour chaque utilisateur d'utiliser un forum afin de favoriser le lien social et d'expliciter éventuellement au mieux sa demande et/ou proposition. Le karma quant à lui est toujours une quantité qui fluctue en fonction des évaluations des autres utilisateurs ayant bénéficiés du service fourni. De plus, chaque classe a été définie par un certain nombre d'attributs qui sont plus précis que ceux du diagramme prévisionnel. Ceux-ci sont commentés plus spécifiquement dans la section en rapport avec les bundles.

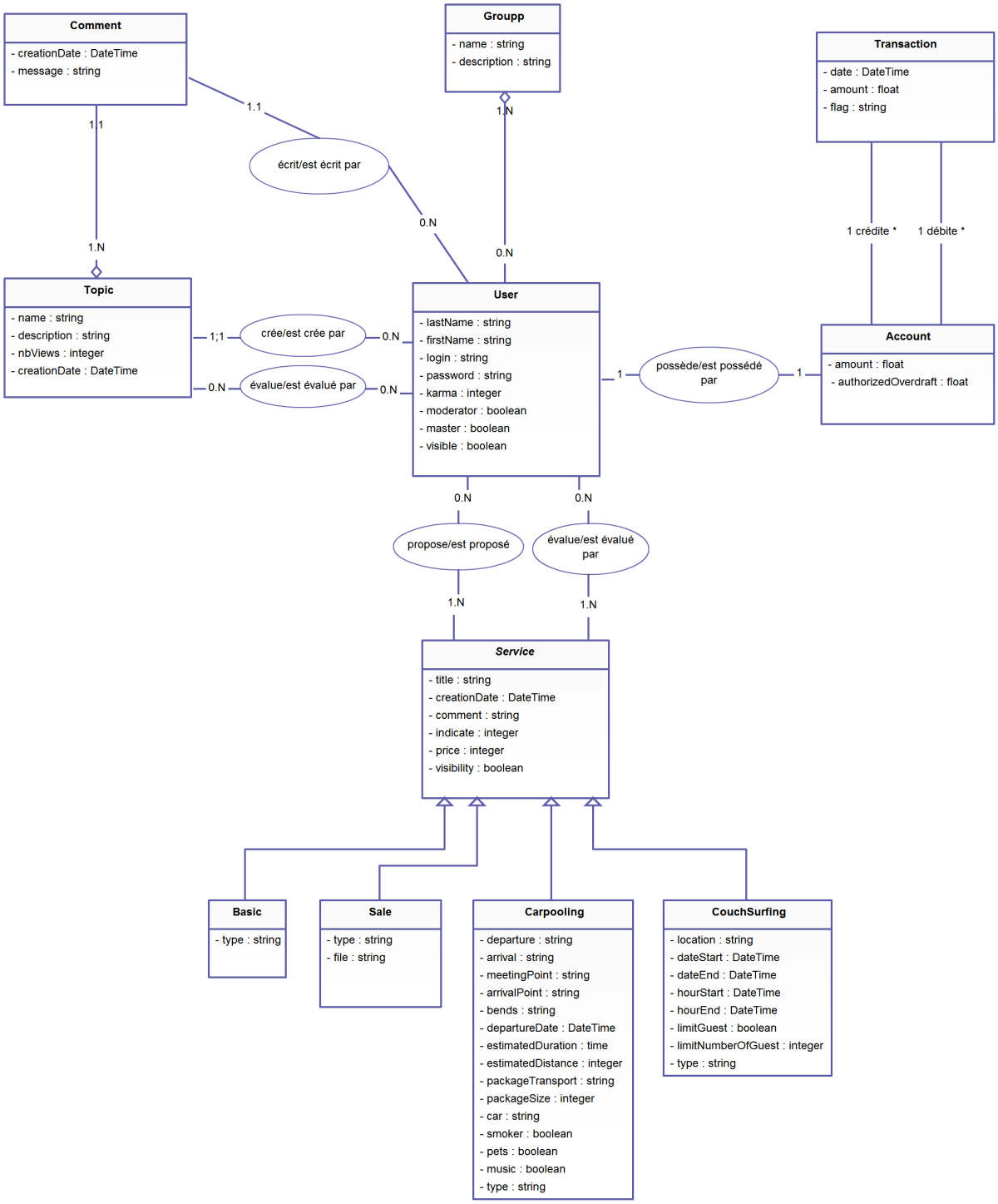



FIGURE 3.5 – Diagramme de classe final

# Chapitre 4

# Réalisation du projet

## 4.1 Les bundles

Afin de réaliser le projet, nous avons décidé de découper le travail en bundles s'occupant chacun d'une partie précise du site, afin de simplifier la naviguation entre ces parties. Chaque bundle gère une partie du modèle, de la vue et du contrôleur de l'application. Un bundle possède son propre nom de la manière suivante, suffixé par `Bundle` (par exemple, le bundle `User` est nommé `UserBundle`).

Le modèle est géré dans le dossier `Entity`, où toutes les classes seront implémentées.

Le contrôleur se trouve dans le dossier `Controller`, un contrôleur effectue des contrôles sur le modèle avant d'envoyer les données à la partie vue. Dans Symfony, les opérations d'un contrôleur sont sous forme de méthodes suffixées par `Action`.

Les vues sont répertoriées dans le dossier `Ressources/views`, on y retrouve donc les différentes vues en lien avec le contrôleur et les modèles donnés.

### 4.1.1 Le bundle User

Le bundle `UserBundle` implémente l'entité utilisateur qui comporte les attributs types d'un utilisateur : pseudo, mot de passe (encodé en MD5), nom, prénom, date de naissance, karma, ...

Le contrôleur permet d'effectuer différentes opérations liées à un utilisateur. En voici les différentes fonctionnalités :

**indexAction** si un utilisateur vient de se connecter la méthode `seeAction` est appelée, sinon la page d'inscription/connexion est affichée ;

**seeAction** affiche le profil d'un utilisateur ;

**addAction** vérifie si les données d'inscription ne sont pas redondantes avec des données déjà contenues dans la base de données : si elles ne le sont pas un nouvel utilisateur est créé et la méthode `seeAction` est appelée, sinon le formulaire est généré à nouveau dans le but d'être rempli correctement ;

**deleteAction** supprime l'utilisateur et tous les services qui y sont associés ;

**connectionAction** connecte un utilisateur en fonction de son login et de son mot de passe ;

**deconnectionAction** déconnecte un utilisateur du site et le redirige vers la page de connexion ;

**editAction** un utilisateur, une fois connecté, peut modifier son profil.

### 4.1.2 Le bundle Service

Ce bundle catégorise les différents types de service existants sur le site, implémentés par des entités séparées, que sont : service de base, covoiturage, vente et couchsurfing.

Chaque service contient :

— un identificateur ;

— un titre ;

— une date de création ;
— une description ;
— un prix ;
— une visibilité ;
— un groupe associé (ou non) ;
— une liaison avec un utilisateur à travers l'entité `ServiceUser` et ses dérivés ;
— un type de service qui permet de définir s'il s'agit d'un service de base, d'une vente, d'un covoiturage ou d'un couchsurfing.

Le contrôleur quant à lui, fonctionne via les méthodes suivantes :

**indexAction** énumère tous les services existants selon un type de classement choisi (par date de création par défaut, par créateur, par type de service, par prix croissant ou décroissant) ;

**addServiceAction** appelle la méthode de création du service que l'on veut créer ;

**seeBasicAction et ses dérivés** permet de voir son service ;

**addBasicAction et ses dérivés** permet d'ajouter son service, à un groupe si on le désire ;

**deleteBasicAction et ses dérivés** permet de supprimer son service ;

**seeMyServicesAction** répertorie tous les services que l'on propose, qu'ils soient réservés ou non ;

**serviceDoneAction** est appelé pour spécifier qu'un service a été effectué.

### 4.1.3 Le bundle Transaction

Ce bundle avait pour objectif premier la gestion des comptes « monétaires » des utilisateurs. Par la suite, la partie évaluation s'est ajoutée à ce bundle, afin de faire le lien entre un service et un paiement.

Les différentes entitées de ce bundle sont :

**Account** cette entité contient un identifiant de compte, un montant de compte ainsi qu'un découvert autorisé ;

**Transaction** cette entité est chargée de stocker une transaction entre deux comptes, c'est à dire les deux comptes concernés, le montant de la transaction ainsi qu'un libellé optionnel ;

**Evaluation et ses dérivés** cette entité fait le lien entre un service souscrit par un utilisateur ainsi qu'une note, précisant si le service a été payé ou non ; l'évaluation sera considérée comme « à faire » tant qu'elle n'aura pas été payée.

Il existe ainsi dans ce bundle deux contrôleurs :

**TransactionController** ce contrôleur gère les actions effectuées entre deux comptes :

- **indexAction** appelle la vue qui affiche, si l'utilisateur est connecté, la liste de ses transactions ;
- **paymentAction** appelle la vue qui affiche un formulaire permettant de faire un virement vers un autre utilisateur et traite le formulaire lors de la soumission ;

**EvaluationController** ce contrôleur gère l'évaluation et le paiement lorsqu'un service est effectué :

- **indexAction** appelle la vue qui affiche, si l'utilisateur est connecté, la liste de tous les services en attente d'évaluation ;
- **evaluationAction** permet à l'utilisateur d'attribuer une note à un service effectué et de le payer, en gérant automatiquement la soumission du formulaire d'évaluation, effectuant le paiement et évaluant le karma (via la méthode `evalKarma`) ;
- **evalKarma** qui va recalculer le karma attribuées à l'utilisateur passé en paramètre en faisant une moyenne arithmétique de toutes les notes ramenées sur 100 (les notes sont sur 10, le karma est sur 100).

### 4.1.4 Le bundle Group

Ce bundle est utilisé pour regrouper les utilisateurs autour de services.

**Groupp** est une entité qui permet le groupement d'utilisateurs par un identifiant unique, un nom et une description ainsi qu'un administrateur, cette entité se nomme `Groupp` car `Group` est un terme réservé par Symfony2 ;

**associatedGroup** est un attribut de l'entité `Service` qui permet d'associer un service à un groupe ; ce service figurera dans la vue `see_group.html.twig\verb` du groupe associé.

`GroupController` est le contrôleur qui permet de gérer les groupes, à savoir, les afficher, afficher un groupe en détails, ajouter un utilisateur, bannir un utilisateur, demander à rejoindre un groupe, supprimer une demande pour rejoindre un groupe et supprimer un groupe.

### 4.1.5 Le bundle Forum

Ce bundle est utilisé pour la communication entre les membres de Poavre sur divers sujets, cela peut être de la communication pure (chat), de la communication sur les services proposés ou encore débattre sur des sujets et proposer des nouveautés à apporter au site.

**Topic** (ou sujet) est une entité qui possède un identifiant unique, un ratio d'approbation (likes/dislikes), un nom, une description, une date de création, un nombre de vues et un auteur ;

**Comment** est une entité qui possède un identifiant unique, un message, une date de création et un auteur ;

**TopicComment** est une entité associative (une association) entre les entités `Topic` et `Comment`, ainsi un commentaire est propre à un sujet qui dispose de plusieurs commentaires ; cette entité possède deux attributs, l'un référençant le sujet et l'autre référençant le commentaire ;

**TopicUser** est une entité associative (une association) entre les entités `Topic` et `User`, ainsi, un utilisateur peut commenter plusieurs sujets et même plusieurs fois le même sujet et chaque sujet est commenté par plusieurs utilisateurs ; cette entité possède deux attributs, l'un référençant le sujet et l'autre référençant l'utilisateur.

`ForumController` est un contrôleur qui permet de gérer les sujets et commentaires, à savoir, afficher la liste des sujets, afficher, créer, commenter, « liker » et « disliker » un sujet.

### 4.1.6 Le bundle Home

Ce bundle est utilisé pour l'affichage de la page d'accueil du site ainsi que pour l'accès à l'onglet « Développeurs ». Ainsi la majeure partie des redirections se font soit vers la création de compte utilisateur ou de connexion, soit vers la page d'accueil.

`HomeController` est un contrôleur qui permet l'affichage de la page d'accueil du site ainsi que l'accès à l'onglet « Développeurs ».

### 4.1.7 Le bundle Administration

Ce bundle est utilisé pour la gestion de Poavre par les modérateurs du site.

`AdministrationController` est un contrôleur qui permet à un administrateur de bannir des membres, les nommer modérateur ou « master » (utilisateur suprême), supprimer des groupes, des services, des sujets du forum ou encore des commentaires sur les différents sujets du forum.

## 4.2 Les services Symfony

Symfony2 permet d'implémenter des services qui sont des fonctions appelables depuis un contrôleur quelconque. Ceci est réalisable grâce à l'« importation » du service dans le fichier `service.yml` présent dans tous bundle généré par Symfony2.

Ici dans chaque bundle et avant chaque méthode d'un contrôleur, un service est appelé. Ce service, `ml.session`, permet de vérifier au début de chaque méthode si la session d'un utilisateur est activée. Si c'est le cas la suite de la méthode se déroule sans problème, sinon l'utilisateur est redirigé vers la page d'inscription.

## 4.3 L'implémentation du modèle

La partie modèle de Symfony2 contient 2 parties, les classes d'entités et les `entityRepository` contenant notamment des requêtes personnalisées avec la base de donnée et les formulaires d'entités.

### 4.3.1 Les classes d'entités

Un bundle contient les différentes entités concernées. Une entité représente une classe du modèle de classe.

Après la création d'un bundle, Symfony2 propose la génération d'une entité grâce à une simple ligne de commande dans la console intégrée de Symfony2.

Les attributs sont alors demandés ainsi que leurs types et quelques spécialisations. Les getters et setters sont ensuite automatiquement générés. Une entité peut être modifiée, suite à cela la base de donnée doit être mise à jour. Cette base de donnée est générée grâce à des commentaires (annotations) et à l'ORM Doctrine.

### 4.3.2 Les « Repositories »

Les `repositories` sont des « dépôts » qui sont interrogés par les contrôleurs. Ces dépôts requêtent la base de données et créent les objets demandés. Cela dit, les requêtes héritées de `EntityRepository` ne permettent que des requêtes simples. Des requêtes plus complexes (personnalisées), peuvent être créés grâce au DQL (Doctrine Query Language), notamment dans les sujets pour calculer les ratios (likes/dislikes) de chaque sujet (voir figure 4.1).

```
$likes = $em->createQuery(
"SELECT COUNT(tu.avis) as nb_likes
FROM M1ForumBundle:TopicUser tu
WHERE tu.avis = 1
AND tu.topic = :value")
->setParameter('value', $value);

$count_likes = (int)$likes->getResult()[0]['nb_likes'];
```

FIGURE 4.1 – Exemple d'utilisation du DQL

Les formulaires peuvent être implémentés grâce à une méthode `buildForm` de Symfony2 (voir 4.2) qui permet de construire un formulaire ne contenant que les attributs désirés. La classe implémentant ce formulaire est alors stockée dans le dossier `Form` d'un bundle et elle doit hériter de la classe `AbstractType`, qui est la superclasse de tous les formulaires. Cette classe peut être générée automatiquement grâce à une commande dans la console de Symfony2.

```
class UserType extends AbstractType {
    public function buildForm(FormBuilderInterface $builder, array $options) {
        $builder
            ->add('lastName','text', array('label' => "Nom"))
            ->add('firstName','text', array('label' => "Prenom"))
            ->add('login','text', array('label' => "Login"))
            ->add('password','password', array('label' => "Mot de passe"));
    }
    ...
}
```

FIGURE 4.2 – Exemple d'utilisation d'un formulaire

## 4.4 Le fonctionnement du contrôleur

Sous Symfony2, le contrôleur est une classe présente dans chaque bundle. Lorsqu'un visiteur du site cherchera à visiter une page, le routeur va analyser l'adresse entrée et utiliser les fichiers de routage pour déterminer quelle méthode (action) de quel contrôleur doit être appelée.

Ainsi, avec la ligne du fichier présentée dans la figure 4.3, c'est la méthode `evaluationAction` du contrôleur `EvaluationController` qui sera appelée. De plus, `{serviceType}` et `{id}` indiquent que ces éléments de l'adresse seront passés en tant que paramètres `$serviceType` et `id` à la méthode

`evaluationAction`. Ainsi, si un visiteur se rend à l'adresse `/evaluation/b/1`, c'est `EvaluationController::evaluati` `('b',1)` qui sera exécutée.

```
ml_transaction_evaluation:
    pattern: /evaluation/{serviceType}/{id}
    defaults: { _controller: MlTransactionBundle:Evaluation:evaluation }
```

FIGURE 4.3 – Extrait du fichier routing.yml du bundle

## 4.5 L'organisation des vues

Les vues sous Symfony sont gérées par Twig qui ajoute de nombreuses fonctionnalités à HTML :
— variables passées en paramètre par le contrôleur ;
— conditions ;
— boucles ;
— nommage de sections de document ;
— hiérarchie de documents Twig : héritage d'un document pour en modifier les sections.

### 4.5.1 Le layout

Le découpage de la vue s'est fait en différents layouts s'occupant d'une partie non-métier de la vue :

**layout.html.twig** est la racine de l'arbre Twig, il déclare le corps du document (header, body) sans le définir et déclare les sections de titre, de shell (interface utilisateur), de style et de scripts ;

**simple.html.twig** définit un shell simple en spécialisant `layout.html.twig`, définissant un pied de page et déclarant le corps de la page ;

**headerbar.html.twig** définit un shell complexe en spécialisant `layout.html.twig`, définissant une barre d'en-tête servant de menu principal, et déclarant une barre latérale et un corps de page à spécialiser.

### 4.5.2 Les vues spécifiques

Chaque bundle définit un lot de vues spécifiques, spécialisant selon le besoin soit `simple.html.twig`, soit `headerbar.html.twig`. La spécialisation de `simple.html.twig` est réalisée en définissant la section de corps du document, le terminant ainsi.

La spécialisation de `headerbar.html.twig` est réalisée en deux étapes, un fichier Twig spécifique au bundle redéfinit la barre latérale, puis est hérité par les vues du bundle qui définissent le corps du document, le terminant ainsi. Ceci permet d'avoir une barre d'en-tête commune à toutes les vues d'un bundle.

## 4.6 Les versions

### 4.6.1 Version 0.1

La première version du projet a été programmée début mars après trois à quatre semaines d'analyse. Cette première version faisant office de socle pour la suite de l'avancement de notre projet, celle-ci fut centrée sur les fondements du projet et réalisée sur une période de trois semaines.

L'objectif de cette version était donc l'implémentation des modèles Utilisateur, Prestation et Transaction. Pour cela, nous avons dû créer un bundle correspondant à ces 3 modèles ; les bundles `User`, `Prestation` et `Transaction`. Le travail fut réparti en fonction des préférences, des curiosités ou encore selon les connaissances de chacun et fut facilitée par le biais de la structuration de Symfony, en effet un certain nombre de personnes étaient rattachées à un bundle ce qui permis d'éviter les conflits.

Concernant la partie modèle, le modèle `Prestation` implémenté par Oualid, Rime et Quentin fut enrichi par des services comme le covoiturage, le bricolage ou les cours particuliers, seul le service covoiturage

a été retenu. Le modèle `User` fut implémenté par Clément et Thin-Hinen et le modèle `Transaction` par Florian.

La vue a été prise en main d'une part par Fabien en structurant les layouts principaux et secondaires et d'autre part par Adrien qui avait pour rôle de coder les vues relatives aux modèles implémentés.

Enfin le contrôleur du bundle utilisateur a été implémenté par Clément pour permettre d'inscrire un utilisateur et par la suite d'être complété pour répondre aux fonctionnalités définies.

Dans un premier temps la version 0.1 a permis la structuration de base du site et l'implémentation des éléments de base mais aussi la prise en main des nouveaux outils (pour la plupart des membres du groupe) que sont Symfony2, Git (et GitHub) et Producteev. De plus la version 0.1 a permis d'établir les conventions nécessaires à l'homogénéisation du projet ainsi il a été décidé que les commentaires et la documentation se feraient en français quant au site il serait en anglais mais disponible en français.

Dans un second temps, lors de l'élaboration de la version 0.1, le but fut son intégration, guidée par Adrien et Fabien, tout le groupe était alors concerné par la résolution des bugs mais aussi par l'ajout de certaines fonctionnalités (formulaire de prestation...), l'ajout de la base du site (vue des prestations) et la vérification de la validité des données récupérées au niveau du contrôleur.

### 4.6.2 Version 0.2

Début Avril l'avancement du projet et l'aboutissement de la version 0.2 du site amena la question du nom ; « Poavre » a alors été proposé par Adrien, qui est un jeu de mots lié au mot « SEL ». La « faute » a été choisie pour permettre une meilleure identification du projet.

La 0.2 fut l'occasion d'une amélioration de la vue, proposée par Adrien, tandis qu'en collaboration avec Clément et Thin-Hinen le site était réorganisé. Fabien, Quentin, Rime et Oualid ont étoffé le bundle Service en proposant de généraliser la proposition de service et en ne conservant que deux services types, le covoiturage et le couchsurfing, tout en adaptant le contrôleur et les vues associées. La partie transaction, encore en cours d'évolution, était toujours en développement par Florian.

Ainsi, cette partie fut principalement un renforcement de la version précédente, seules quelques légères améliorations ont été apportées.

Elle fut réalisée en deux semaines.

### 4.6.3 Version 0.3

La version 0.3 a été l'occasion d'un enrichissement au niveau de la vue, où les prestations (renommées services à partir de cette version) sont alors complètes tant au niveau de la vue que du contrôleur. L'implémentation des transactions est également terminée. Arrivé à cette version, les outils sont bien compris rendant le travail plus efficace et dynamique.

Une épuration du code s'est faite lors de cette version : la factorisation du code est alors réalisée (mise en place du service `ml_session` qui teste si un utilisateur est connecté). De plus, les pages `Twig` reçoivent désormais des entités complètes et non plus chaque attribut d'une entité.

### 4.6.4 Version 0.4

Lors de cette version, Adrien et Thin-Hinen se sont chargés de peaufiner l'aspect visuel ; la barre d'en-tête a été remaniée ainsi que la navigation, pour donner un aspect plus épuré, simple et facile de prise en main pour l'utilisateur.

Au niveau du modèle, les bundles `Group` et `Forum` ont vu le jour grâce à Fabien et Quentin ; Clément et Florian ont effectué l'implémentation de l'évaluation et du paiement des services réalisés ; ces évaluations influent également sur le karma du prestataire, calculé grâce aux notes attribuées par les utilisateurs ayant eu recours à un service.

Oualid et Quentin ont factorisé les services en types de service pour faciliter la gestion aux utilisateurs. Quant à la réalisation des interfaces des services, elle fut réalisée par Rime.

Cette version fut également l'occasion de la mise en ligne du site. Adrien devait à l'origine la réaliser, mais suite à un problème de compatibilité de la version de PHP présente sur le serveur qui nous a été prêté, ce fut Fabien qui s'en chargea finalement sur un autre serveur auquel il avait accès.

### 4.6.5 Version 1.0

Cette version étant une finalisation du projet, certains points on été améliorés pour plus de clarté et de sécurité à l'interieur du site.

Une vérification de la capacité de paiement des utilisateurs fut implémentée par Florian. En effet, un utilisateur ne peut pas réserver de nouveaux services si ses moyens ne sont pas suffisants ; les transactions encore en cours mais non-payées sont maintenant prises en compte pour calculer le solde disponible de l'utilisateur lors d'une nouvelle demande de service.

La désactivation du compte (si l'utilisateur n'a pas de transactions en cours) a également été ajoutée. Nous n'avons pas implémentés la suppression totale du compte (pour garder une trace des transactions en cas de litige), mais avons cependant laissé cette possibilité à l'utilisateur via un mail de contact, afin de rester en conformité avec les règles de la CNIL.

Oualid et Fabien ont été chargés de la transition vers la dernière version de `Bootstrap` pour un meilleur rendu. La `side-bar` a été allégée et la page d'accueil fut également modifiée pour être plus attractive.

### 4.6.6 Version 1.0.1

La version 1.0.1 fut une version de correction de bug. Rien n'a réellement été ajouté, seuls quelques bugs encore présents dans la version 1.0 ont été corrigés.

# Chapitre 5

# Manuel

## 5.1 Manuel d'installation

Concernant l'hébergement du site chez un hébergeur web (Hostinger par exemple car il possède une version récente de PHP ainsi que tous les outils nécessaires à la mise en place du projet avec Symfony2) vous devez lui transmettre via FTP (par exemple FileZilla (voir [3])) l'ensemble des fichiers en conservant la structure actuelle des dossiers, à savoir :

- app ;
- bin ;
- public_html (et non web) ;
- src ;
- vendor ;

Ainsi que les différents fichiers situés à la racine du projet.

En cas de page blanche lors de l'accès au site en ligne vous devez vérifier qu'il n'y ait pas d'erreurs dans le code du projet car ceci arrive régulièrement et aucune erreur n'est décelée (la correction de ces problèmes peut se faire en local ou en distant via app_dev.php et non app.php car l'erreur apparaît clairement par ce biais et devient facilement solvable).

Aussi, pour vérifier que tout fonctionne chez l'hébergeur vous devez pouvoir accéder au fichier `config.php` situé à la racine de votre site et il ne doit y avoir aucune erreur sur cette page, si tel n'est pas le cas alors vérifiez le fichier `config.php` du dossier public_html. Si vous ne disposez pas des droits d'accès aux fichiers vous devez ajouter votre adresse ip, dans `app_dev.php` à la ligne `!in_array (@$_SERVER['REMOTE_ADDR'], array('127.0.0.1', 'fe80::1', '::1'))` et dans `config.php` aux lignes :

```
if (!in_array(@$_SERVER['REMOTE_ADDR'], array(
    '127.0.0.1',
    '::1',
))) {
    header('HTTP/1.0 403 Forbidden');
    exit('This script is only accessible from localhost.');
}
```

Ainsi, vous disposerez de votre site en production. Encore une chose, il faut faire attention car chez certains hébergeurs le Système de Gestion de Bases de Données (SGBD) respecte la casse, dans ce cas, vous devrez modifier à la main le nom des tables dans ce même SGBD en les mettant sous la forme « UpperCamelCase » autrement les tables ne seront pas reconnues lors de l'exécution des requêtes utilisées dans le programme.

## 5.2 Manuel d'utilisation

Un visiteur peut accéder aux différentes informations sur le Système d'Echange Local, la monnaie libre. Il peut aussi s'inscrire par le biais de son prénom, de son nom, d'un login ainsi que d'un mot de passe. Avec ces deux derniers il peut se connecter (voir figure 5.1).

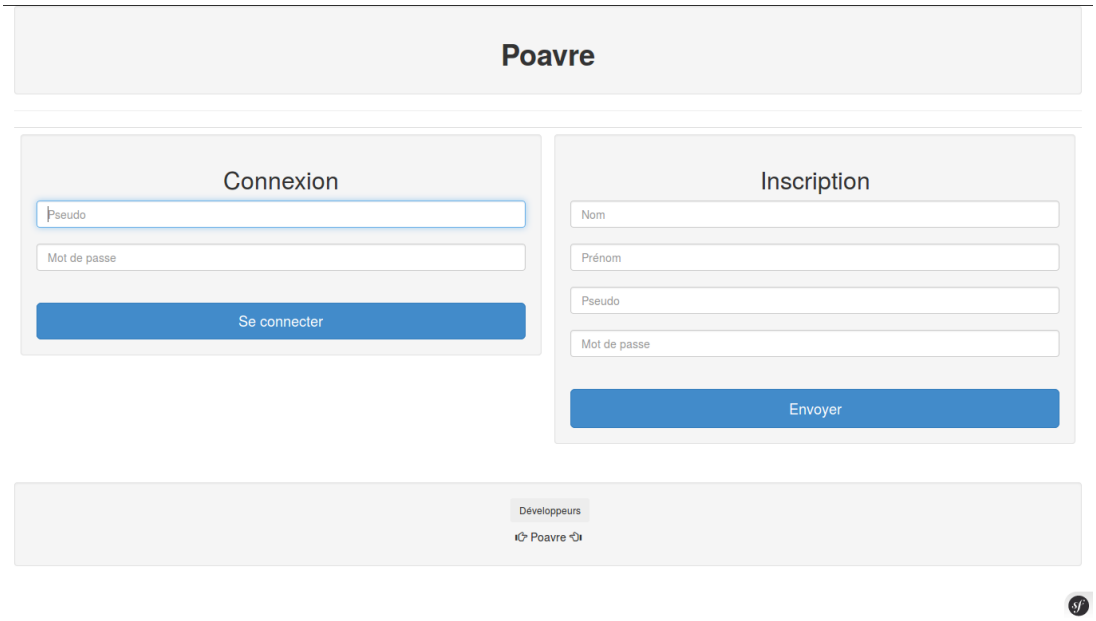



FIGURE 5.1 – Capture d'écran d'une inscription/connexion

Lorsque le visiteur désormais nommé utilisateur est connecté (voir figure 5.2) il peut à tout moment se déconnecter, éditer son profil (nom, prénom et/ou mot de passe).

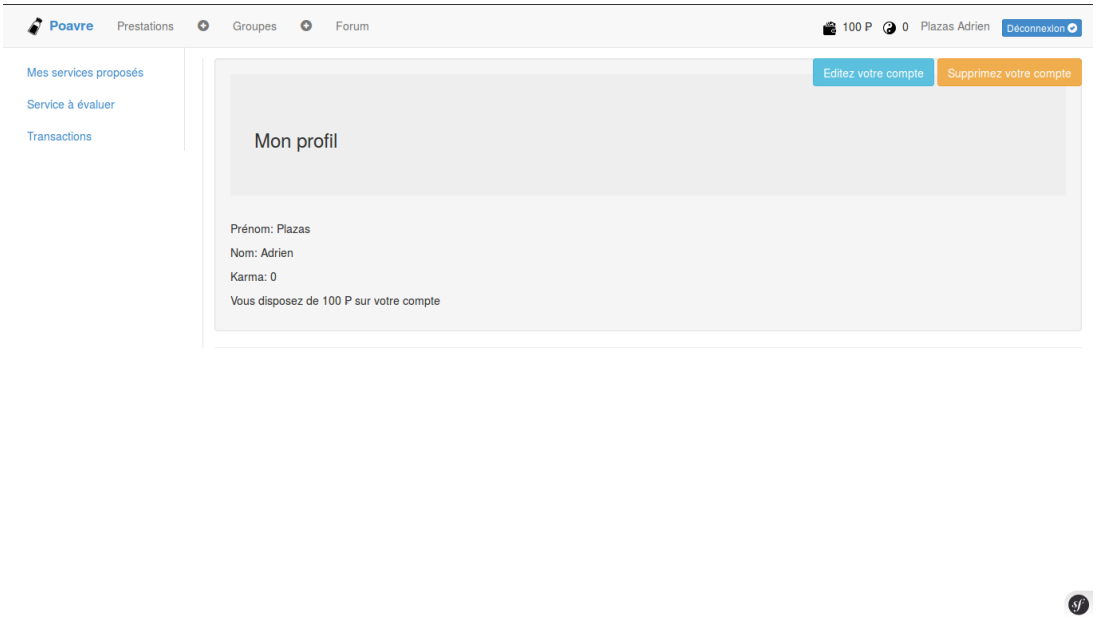



FIGURE 5.2 – Capture d'écran du profil

Il peut accéder aux différents services proposés sur le site. Ces derniers peuvent être des services de covoiturage, de couchsurfing, de vente ou un autre type de service nommé « service de base ».

Ainsi, il peut réserver un service, dans ce cas, le créateur du service se doit de le réaliser ce qui entraînera par la suite une obligation de transaction entre l'utilisateur et le créateur du service ainsi qu'une évaluation par la personne ayant bénéficiée du service (voir figure 5.3).

Chaque utilisateur peut également créer son propre service, il peut à tout moment l'annuler, il disparaît ainsi de la liste des services proposés sur le site (voir figure 5.4).

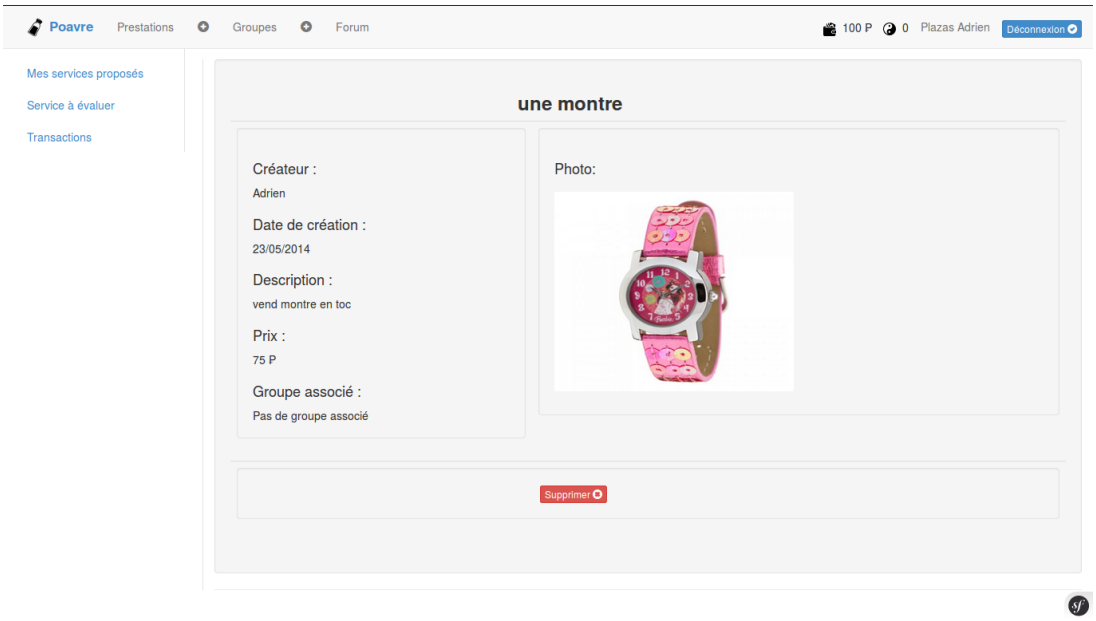



FIGURE 5.3 – Capture d'écran d'une réservation

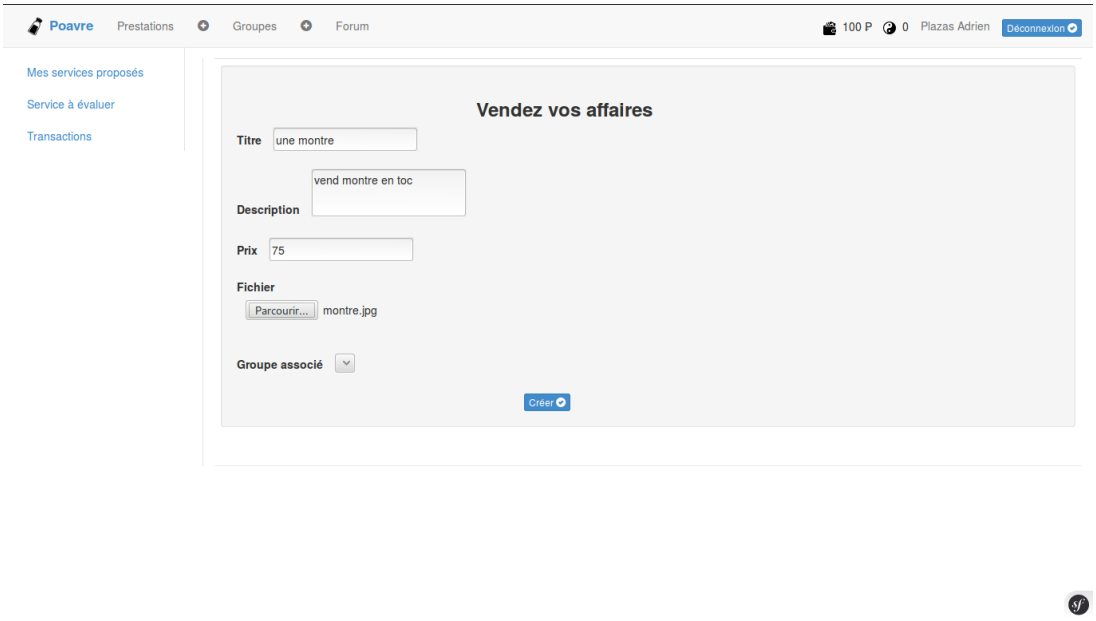



FIGURE 5.4 – Capture d'écran d'une vente

L'utilisateur ne peut quitter le site que s'il a clôturé tous les services qu'il a réservé (un mail de contact est cependant disponible pour une suppression complète). Par ailleurs, un utilisateur peut voir la liste des groupes présents sur le site, il peut candidater pour rejoindre ces différents groupes et peut à tout moment annuler sa demande. Il peut aussi visualiser pour chaque groupe la liste de ses membres ainsi que les services qui lui sont associés. Lorsque l'utilisateur fait parti d'un groupe il peut quitter ce dernier s'il le souhaite. Un utilisateur peut aussi créer son propre groupe en ajoutant des utilisateurs à ce dernier. Seuls un nom de groupe et une description sont demandés pour la création d'un groupe. Un administrateur de groupe peut bannir les membres du groupe ou en ajouter de nouveaux et il peut voir la liste des candidatures pour rejoindre le groupe et les accepter ou refuser. Il peut à tout moment dissoudre le groupe, dans ce cas, tous les services associés au groupe ne le sont plus, ils ne disparaissent cependant

pas du site (voir figure 5.5).

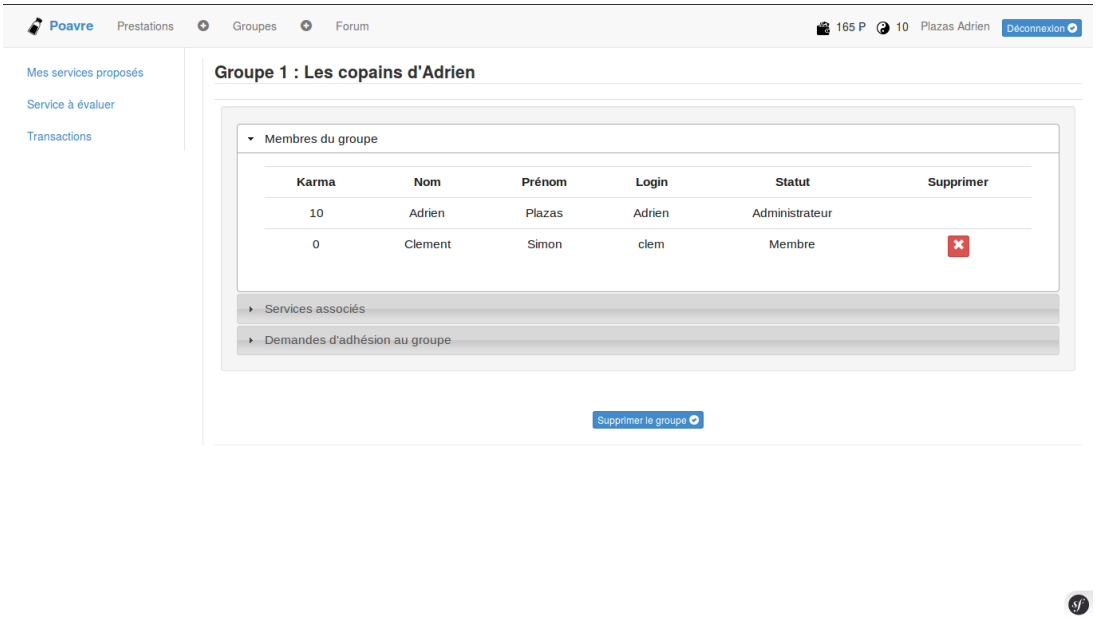



FIGURE 5.5 – Capture d'écran d'un groupe

Un membre du site peut accéder aux différents services qu'il propose ainsi qu'aux services qu'on lui a réservé. Par ce dernier cas, il peut confirmer qu'un service a bien été effectué et après cela il peut évaluer le membre qui a réalisé le service. Cette évaluation permet à ce dernier d'obtenir des points de karma et déclenche la transaction entre les deux utilisateurs (voir figure 5.6).


FIGURE 5.6 – Capture d'écran d'une évaluation

A tout moment l'utilisateur peut visualiser le nombre de points de karma dont il dispose ainsi que l'argent qu'il possède. Un utilisateur ne peut réserver un service que s'il dispose des fonds nécessaires pour payer le service (l'argent qu'il doit à un autre utilisateur est pris en compte dans ce calcul). L'utilisateur peut visualiser la liste de ses transactions avec les autres utilisateurs et peut réaliser des paiements

qualifiés de « dons » avec les autres utilisateurs (voir figure 5.7).

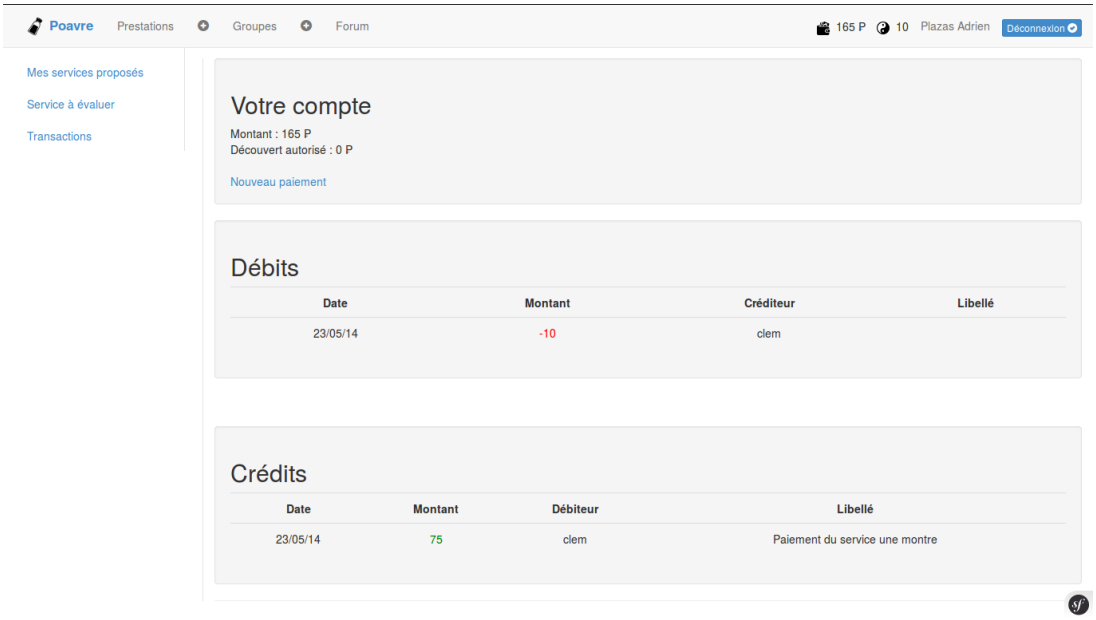



FIGURE 5.7 – Capture d'écran d'une transaction

Un membre du site peut accéder au Forum, par celui-ci il peut visualiser la liste des différents « topics » (ou « sujets ») classés par ratios (« likes »/« dislikes ») décroissants (voir figure 5.8).

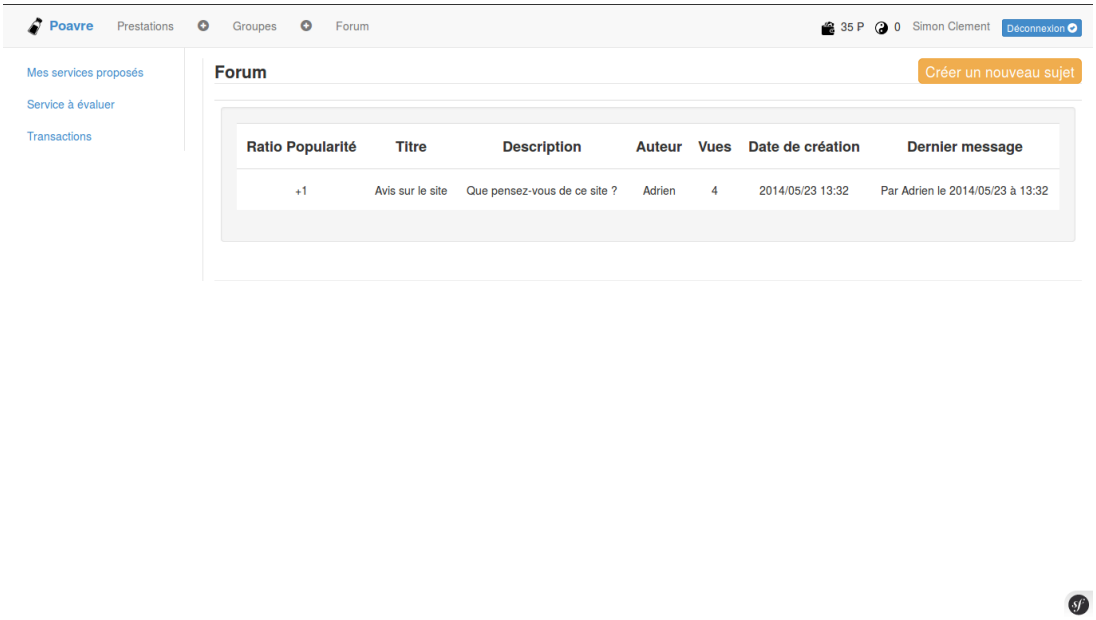



FIGURE 5.8 – Capture d'écran de la liste des topics

Il peut ainsi accéder à un topic, commenter celui-ci ou encore l'évaluer par un « like » ou un « dislike ». Bien évidemment l'utilisateur peut lui-même créer un topic (voir figure 5.9).

De plus, un utilisateur bénéficiant des droits d'administration peut accéder à l'espace d'administration du site. Depuis ce dernier, il peut bannir des utilisateurs, leur accorder les droits d'administration ou les droits « suprêmes », il peut aussi supprimer des services qu'il juge non-conformes, des groupes, des topics ou encore des commentaires sur les topics. Les droits « suprêmes » fournissent à un utilisateur un rôle

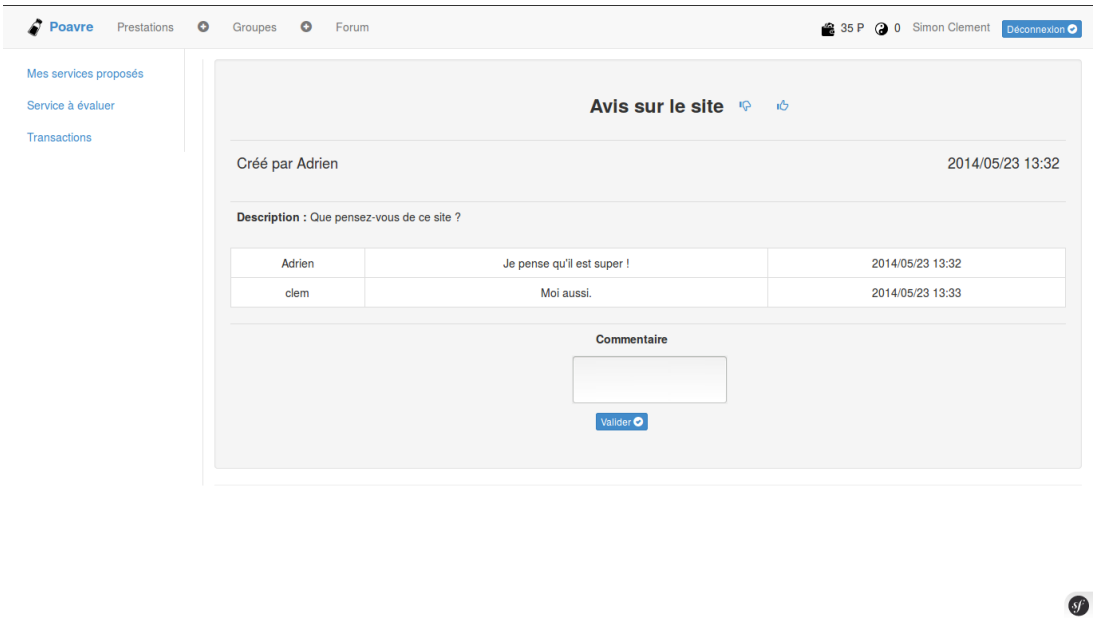



FIGURE 5.9 – Capture d'écran du forum

de dirigeant du site. Les dirigeants ne peuvent pas se bannir entre eux.

# Chapitre 6

# Bilan de ce projet

Le travail collaboratif nécessite beaucoup d'organisation, notamment lorsqu'il s'agit d'un travail de groupe à 8. En effet, la coordination entre les membres s'est avérée difficile, bien que chacun ait travaillé sur des parties indépendantes. Des outils simples et efficaces se sont révélés indispensables, de ce fait, Producteev et Github ont été d'une grande utilité et Symfony2 fut un très bon choix du fait de ses performances. Cela a cependant également été source de ralentissement en début de semestre, certains membres du groupe devant apprendre le fonctionnement de git, ou le temps de comprendre le fonctionnement de Symfony2.

Les réunions de projet qu'elles aient été formelles ou informelles ont apportées beaucoup au projet, elles en ont été le fondement. Elles ont permis à de nombreuses reprises de discuter et d'évacuer les tensions qui avaient pu apparaître entre les membres du groupe. De façon globale, nous avons appréciés travailler ensemble, malgré les difficultés. En effet, nous avons avec satisfaction produit un site fonctionnel, en trouvant au fur et à mesure de l'avancée du semestre une certaine synergie qui nous a permis de travailler de façon agréable et efficace.

# Chapitre 7

# Perspectives et conclusions

Des perspectives de développement demeurent, à savoir :

— un système de chat privé (discussion instantanée et/ou message avec sujet) entre les utilisateurs afin de discuter des services qu'ils proposent ;

— l'implémentation du compte premium imaginé n'a pas été utilisé finalement dans le projet (système de gain en monnaie libre contre des euros, par exemple) ;

— la mise en place d'avatars qui pourront être utiles notamment pour le forum et le chat privé.

Les kiwis (mini-avatars représentatifs de l'échelon social de l'utilisateur sous forme de poivriers ou de grains de poivre par exemple), initialement prévus n'ont pas vus le jour et sont donc implémentables dans un futur proche. Aussi, une barre latérale présentant les services qui sont proposés par les utilisateurs possédant des hauts karmas pourrait être implémentée sur les vues liées aux services.

Finalement, la majeure partie des idées énoncées lors des premières réunions de projet ont été réalisées, le diagramme des cas d'utilisation ainsi que le diagramme de classe initiaux ont globalement été bien suivis. En effet, un utilisateur peut créer des services en échange de la monnaie libre choisie, il peut gérer ses services et en réserver d'autres. Il peut aussi évaluer un service réalisé par un prestataire ce qui donne des points de karma au prestataire, de ce fait, il gagne en « confiance » et ses services en sont valorisés et deviennent attrayants. Un utilisateur peut aussi créer un groupe, rejoindre un groupe et créer des services associés à un groupe, ainsi, les services internes au groupe sont accessibles plus facilement par les membres du groupes et peuvent être catégorisés selon le désir du fondateur du groupe. Aussi, plusieurs types de services sont disponibles, à savoir, les ventes, le couchsurfing, le covoiturage ainsi que d'autres services que nous considérons comme des services de base (cours d'anglais par exemple). Un forum est également disponible afin de communiquer entre membres de Poavre, le forum fonctionne par topics et par commentaires de topics. Les topics sont classés par ratio (likes/dislikes) afin de faire remonter les plus intéressants. Un espace administrateur est disponible afin de gérer le site et ses diverses fonctionnalités.

# Annexe A

# Code

## A.1 Exemple de modèle

### A.1.1 src/Ml/TransactionBundle/Entity/Transaction.php

```
<?php

namespace Ml\TransactionBundle\Entity;

use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Validator\Constraints as Assert;

/**
 * Transaction
 *
 * @ORM\Table()
 * @ORM\Entity(repositoryClass="Ml\TransactionBundle\Entity\TransactionRepository")
 */
class Transaction
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

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

    /**
     * @var float
     *
     * @Assert\GreaterThan(value=0,
  *         message="Amount's transaction must be positive")
     * @ORM\Column(name="amount", type="float")
     */
    private $amount;

    /**
     * @var string
     *
     * @ORM\Column(name="flag", type="string", length=255)
```

```
     */
    private $flag;

    /**
     * @ORM\ManyToOne(targetEntity="Ml\TransactionBundle\Entity\Account",cascade={"
        persist"})
     * @ORM\JoinColumn(nullable=false)
     */
     private $debitedAccount;

    /**
     * @ORM\ManyToOne(targetEntity="Ml\TransactionBundle\Entity\Account",cascade={"
        persist"})
     * @ORM\JoinColumn(nullable=false)
     */
     private $creditedAccount;

    /**
     * Create a new transaction with parameters.
     */
    public function __construct($debited,$credited,$amount,$flag) {
        $this->setDebitedAccount($debited);
        $this->setCreditedAccount($credited);
        $this->setAmount($amount);
        $this->setFlag($flag);
        $this->setDate(new \DateTime());
    }

    /**
     * Get id
     *
     * @return integer
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set date
     *
     * @param \DateTime $date
     * @return Transaction
     */
    public function setDate($date)
    {
        $this->date = $date;

        return $this;
    }

    /**
     * Get date
     *
     * @return \DateTime
     */
    public function getDate()
    {
        return $this->date;
    }

    /**
     * Set amount
     *
     * @param float $amount
     * @return Transaction
     */
    public function setAmount($amount)
    {
```

```
        $this->amount = $amount;

        return $this;
    }

    /**
     * Get amount
     *
     * @return float
     */
    public function getAmount()
    {
        return $this->amount;
    }

    /**
     * Set flag
     *
     * @param string $flag
     * @return Transaction
     */
    public function setFlag($flag)
    {
        $this->flag = $flag;

        return $this;
    }

    /**
     * Get flag
     *
     * @return string
     */
    public function getFlag()
    {
        return $this->flag;
    }

    /**
     * Set DebitedAccount
     *
     * @param \Ml\TransactionBundle\Entity\Account $DebitedAccount
     * @return Transaction
     */
    public function setDebitedAccount(\Ml\TransactionBundle\Entity\Account
        $debitedAccount)
    {
        $this->debitedAccount = $debitedAccount;

        return $this;
    }

    /**
     * Get DebitedAccount
     *
     * @return \Ml\TransactionBundle\Entity\Account
     */
    public function getDebitedAccount()
    {
        return $this->debitedAccount;
    }

    /**
     * Set CreditedAccount
     *
     * @param \Ml\TransactionBundle\Entity\Account $CreditedAccount
     * @return Transaction
     */
```

```
    public function setCreditedAccount(\Ml\TransactionBundle\Entity\Account
        $creditedAccount)
    {
        $this->creditedAccount = $creditedAccount;

        return $this;
    }

    /**
     * Get CreditedAccount
     *
     * @return \Ml\TransactionBundle\Entity\Account
     */
    public function getCreditedAccount()
    {
        return $this->creditedAccount;
    }
}
```

### A.1.2 src/Ml/TransactionBundle/Entity/Account.php

```
<?php

namespace Ml\TransactionBundle\Entity;

use Doctrine\ORM\Mapping as ORM;
use Ml\TransactionBundle\Exception\TransactionException as TransactionException;
use Ml\TransactionBundle\Exception\RefusedTransactionException as
    RefusedTransactionException;
use Symfony\Component\Validator\Constraints as Assert;
use Ml\UserBundle\Entity\User;

/**
 * Account
 *
 * @ORM\Table()
 * @ORM\Entity(repositoryClass="Ml\TransactionBundle\Entity\AccountRepository")
 */
class Account
{

 /**
     * @ORM\OneToOne(targetEntity="Ml\UserBundle\Entity\User",mappedBy="account",cascade
         ={"persist"})
  */
 private $owner;

    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

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

    /**
     * @var float
     *
     * @Assert\Range(max=0)
     * @ORM\Column(name="authorizedOverdraft", type="float")
     */
```

```
private $authorizedOverdraft;

/**********************/
/***** ACCESSORS *****/
/**********************/

/**
 * Get owner
 *
 * @return User
 */
public function getOwner()
{
    return $this->owner;
}

/**
 * Get id
 *
 * @return integer
 */
public function getId()
{
    return $this->id;
}

/**
 * Set amount
 *
 * @param float $amount
 * @return Account
 */
public function setAmount($amount)
{
    $this->amount = $amount;

    return $this;
}

/**
 * Get amount
 *
 * @return float
 */
public function getAmount()
{
    return $this->amount;
}

/**
 * Set authorizedOverdraft
 *
 * @param float $authorizedOverdraft
 * @return Account
 */
public function setAuthorizedOverdraft($authorizedOverdraft)
{
    $this->authorizedOverdraft = $authorizedOverdraft;

    return $this;
}

/**
 * Get authorizedOverdraft
 *
 * @return float
 */
public function getAuthorizedOverdraft()
{
```

```
        return $this->authorizedOverdraft;
    }

    /**********************/
    /***** CONSTRUCTOR *****/
    /**********************/

    public function __construct($amount=0.0,$authorizedOverdraft=0.0) {
        /* Par défaut, aucun découvert n'est autorisé */
        $this->amount = $amount;
        $this->authorizedOverdraft = $authorizedOverdraft;
    }

    /*******************/
    /***** METHODS *****/
    /*******************/

    /**
     * Credit the amount
     *
     * @return float The new amount
     */
     private function pay($amount) {
        $this->setAmount($this->getAmount()+$amount);
     }

     /**
     * Debit the amount
     *
     * @return float The new amount
     */
     private function withdraw($amount) {
        $this->setAmount($this->getAmount()-$amount);
     }

    /**
     * Pay another account
     *
     * @return The transaction done
     */
    public function payment(&$target,$amount,$flag="") {
        /* Test des préconditions : la cible existe, le montant est positif et le compte
            peut effectuer le paiement. */
        if($amount < 0) throw new TransactionException("Amount has to be positive.");
        if($this->getAmount()-$amount < $this->authorizedOverdraft) throw new
            RefusedTransactionException("Insufficient fund.");
        if($this === $target) throw new RefusedTransactionException("Cannot send money to
            yourself.");

        $this->withdraw($amount);
        $target->pay($amount);

        return new Transaction($this,$target,$amount,$flag);
    }


    /**
     * Set owner
     *
     * @param \Ml\UserBundle\Entity\User $owner
     * @return Account
     */
    public function setOwner(\Ml\UserBundle\Entity\User $owner = null)
    {
        $this->owner = $owner;

        return $this;
    }
}
```

### A.1.3 src/Ml/TransactionBundle/Entity/Evaluation.php

```php
<?php

namespace Ml\TransactionBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * Evaluation
 *
 * @ORM\MappedSuperclass
 *
 */
abstract class Evaluation
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;

    /**
    * @ORM\ManyToOne(targetEntity="Ml\UserBundle\Entity\User",inversedBy="service")
    * @ORM\JoinColumn(nullable=false)
     */
    protected $subscriber;

    /**
    * @ORM\ManyToOne(targetEntity="Ml\UserBundle\Entity\User",inversedBy="service")
    * @ORM\JoinColumn(nullable=false)
     */
    protected $owner;

    /**
     * @var boolean
     *
     * @ORM\Column(name="payed", type="boolean")
     */
    protected $payed;

    /**
     * @var integer
     *
     * @ORM\Column(name="eval", type="integer")
     */
    protected $eval;


    /**
     * Get id
     *
     * @return integer
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set payed
     *
     * @param boolean $payed
     * @return Eval
     */
    public function setPayed($payed)
```

```
    {
        $this->payed = $payed;

        return $this;
    }

    /**
     * Get payed
     *
     * @return boolean
     */
    public function getPayed()
    {
        return $this->payed;
    }

    /**
     * Set eval
     *
     * @param integer $eval
     * @return Eval
     */
    public function setEval($eval)
    {
        $this->eval = $eval;

        return $this;
    }

    /**
     * Get eval
     *
     * @return integer
     */
    public function getEval()
    {
        return $this->eval;
    }


    public function setSubscriber($sub)
    {
        $this->subscriber = $sub;

        return $this;
    }

    public function getSubscriber()
    {
        return $this->subscriber;
    }

    public function setOwner($owner)
    {
        $this->owner = $owner;

        return $this;
    }

    public function getOwner()
    {
        return $this->owner;
    }
}
```

### A.1.4 src/Ml/TransactionBundle/Entity/BasicEval.php

```
<?php
```

```php
namespace Ml\TransactionBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * BasicEval
 *
 * @ORM\Table()
 * @ORM\Entity(repositoryClass="Ml\TransactionBundle\Entity\BasicEvalRepository")
 */
class BasicEval extends Evaluation
{

    /**
  * @ORM\ManyToOne(targetEntity="Ml\ServiceBundle\Entity\Basic")
  */
    private $service;


    /**
     * Get id
     *
     * @return integer
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set service
     *
     * @param string $service
     * @return BasicEval
     */
    public function setService($service)
    {
        $this->service = $service;

        return $this;
    }

    /**
     * Get service
     *
     * @return string
     */
    public function getService()
    {
        return $this->service;
    }
}
```

## A.2 Exemple de formulaire

### A.2.1 src/Ml/UserBundle/Form/UserType.php

```php
<?php

namespace Ml\UserBundle\Form;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolverInterface;

/**
 * User form
 */
```

```
class UserType extends AbstractType
{
        /**
     * @param FormBuilderInterface $builder
     * @param array $options
     */
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            //->add('premium','boolean')
            ->add('lastName','text', array(
            'label' => "Nom"))
            ->add('firstName','text', array(
            'label' => "Prénom"))
            ->add('login','text', array(
            'label' => "Login"))
            ->add('password','password', array(
            'label' => "Mot de passe"))
        ;
    }

    /**
  * Set Default options
     * @param OptionsResolverInterface $resolver
     */
    public function setDefaultOptions(OptionsResolverInterface $resolver)
    {
        $resolver->setDefaults(array(
            'data_class' => 'Ml\UserBundle\Entity\User'
        ));
    }

    /**
  * Get name
     * @return string
     */
    public function getName()
    {
        return 'ml_userbundle_user';
    }
}
```

## A.3 Exemple d'exception

### A.3.1 src/Ml/TransactionBundle/Exception/TransactionException.php

```
<?php

namespace Ml\TransactionBundle\Exception;

interface TransactionExceptionInterface {}

class TransactionException extends \Exception implements TransactionExceptionInterface {
    public function __construct($message=null, $code=0) {
        parent::__construct($message,$code);
    }
}
```

### A.3.2 src/Ml/TransactionBundle/Exception/RefusedTransactionException.php

```
<?php

namespace Ml\TransactionBundle\Exception;

class RefusedTransactionException extends TransactionException implements
    TransactionExceptionInterface {
    public function __construct($message=null, $code=0) {
        parent::__construct("Transaction refused : ".$message,$code);
```

```
    }
}
```

## A.4 Exemple de contôleur

### A.4.1 src/Ml/TransactionBundle/Controller/TransactionController.php

```php
<?php

namespace Ml\TransactionBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpFoundation\Session\Session;
use Ml\TransactionBundle\Entity\Account;
use Ml\TransactionBundle\Entity\Transaction;
use Ml\TransactionBundle\Exception\TransactionException;
use Ml\UserBundle\Entity\User;

/**
 * Transaction Controller extending Controller
 * This one is used for transactions between current user and another users
 */
class TransactionController extends Controller
{
 /**
  * Display all transactions and able current user to pay someone
  * @return Twig template MlTransactionBundle:Transaction:index.html.twig
  */
    public function indexAction() {
  /* Test connexion */
  $req = $this->get('request');

        try {
      $login = $this->container->get('ml.session')->sessionExist($req);
  }
  catch (\Exception $e) {
      return $this->redirect($this->generateUrl('ml_user_add'));
  }

        $user = $this->getDoctrine()
   ->getRepository('MlUserBundle:User')
   ->findOneByLogin($login);

  // On récupère les transactions sortantes
  $outTransactions = $this->getDoctrine()
       ->getManager()
       ->getRepository('MlTransactionBundle:Transaction')
       ->findBy(array("debitedAccount" => $user->getAccount()),array('date' => 'DESC'));

  // Puis les entrantes
  $inTransactions = $this->getDoctrine()
       ->getManager()
       ->getRepository('MlTransactionBundle:Transaction')
       ->findBy(array("creditedAccount" => $user->getAccount()),array('date' => 'DESC'));

  if ($outTransactions == NULL) {
   $outTransactions = NULL;
  }

  if ($inTransactions == NULL) {
   $inTransactions = NULL;
  }

  return $this->render('MlTransactionBundle:Transaction:index.html.twig', array(
    'outTransactions' => $outTransactions,
    'inTransactions' => $inTransactions,
    'user' => $user));
```

```
    }

/**
 * Pay someone (from current user to another one)
 * @return Twig template MlTransactionBundle:Transaction:payment.html.twig
 */
    public function paymentAction() {
        /* Test connexion */
$req = $this->get('request');

        try {
     $login = $this->container->get('ml.session')->sessionExist($req);
}
catch (\Exception $e) {
     return $this->redirect($this->generateUrl('ml_user_add'));
}

$all_users = $this->getDoctrine()
 ->getRepository('MlUserBundle:User')
 ->findAll();

if ($all_users != NULL) {
 for ($i = 0; $i<sizeof($all_users); $i++) {
  // Add all users except current user
  if ($all_users[$i]->getLogin() != $login) {
   $users[$all_users[$i]->getLogin()] = $all_users[$i]->getLogin();
  }
 }
}

        $user = $this->getDoctrine()
 ->getRepository('MlUserBundle:User')
 ->findOneByLogin($login);

 // Si un paiement est effectué
 if($req->getMethod() == 'POST'){
     $recipient = $this->getDoctrine()
    ->getRepository('MlUserBundle:User')
    ->findOneBy(array('login' => $req->request->get('recipient')));

  $amount = $req->request->get('amount');
  $flag = $req->request->get('flag');

      if($this->container->get('ml.reservation')->canPay($user,$amount,$this->
          getDoctrine()->getManager())) {

       if($recipient != null) {
        try {
         $account = &$recipient->getAccount();
         $ret = $user->getAccount()->payment($account,$amount,$flag);
         $this->getDoctrine()->getManager()->persist($ret);
         $this->getDoctrine()->getManager()->persist($user->getAccount());
         $this->getDoctrine()->getManager()->persist($account);
         $this->getDoctrine()->getManager()->flush();

             return $this->redirect($this->generateUrl('ml_transaction_homepage'));
        }
        catch(\Exception $e) {
         return $this->render('MlTransactionBundle:Transaction:payment.html.twig', array
             ('user'=>$user,'users'=>$users,'error'=>$e->getMessage()));
        }
       }
       else {
           return $this->render('MlTransactionBundle:Transaction:payment.html.twig',
               array('user'=>$user,'users'=>$users,'error'=>'L\'utilisateur '.$req->
               request->get('recipient').' n\'existe pas.'));
       }

      }
```

```
        else {
            return $this->render('MlTransactionBundle:Transaction:payment.html.twig',
                array('user'=>$user,'users'=>$users,'error'=>'Vous n\'avez pas les moyens
                d\'effectuer cette transaction.'));
        }
    }

  return $this->render('MlTransactionBundle:Transaction:payment.html.twig', array('user'
      =>$user,'users'=>$users));
    }
}
```

## A.5 Exemple de routes

### A.5.1 src/Ml/TransactionBundle/Resources/config/routing.yml

```
ml_transaction_homepage:
    pattern:  /transaction
    defaults: { _controller: MlTransactionBundle:Transaction:index }

ml_transaction_payment:
    pattern:  /transaction/pay
    defaults: { _controller: MlTransactionBundle:Transaction:payment }

ml_transaction_eval_index:
    pattern:  /evaluation
    defaults: { _controller: MlTransactionBundle:Evaluation:index }

ml_transaction_evaluation:
    pattern:  /evaluation/{serviceType}/{id}
    defaults: { _controller: MlTransactionBundle:Evaluation:evaluation }
```

## A.6 Exemple de vues

### A.6.1 app/Resources/views/layout.html.twig

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

    <title>{% block title %}Poavre{% endblock %}</title>

    {% block stylesheets %}
      <link rel="stylesheet" href="{{ asset('css/bootstrap.css') }}" type="text/css" />
   <link href='http://fonts.googleapis.com/css?family=Audiowide' rel='stylesheet' type='
       text/css'>
   <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.
       css">
    {% endblock %}
  </head>

  <body>
    {% block shell %}
    {% endblock %}
  </body>

  {% block javasripts %}
  <script src="//code.jquery.com/jquery-1.9.1.js"></script>
  <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>

   <script>
    $(function() {
   $( "#accordion" ).accordion({
     heightStyle: "content",
     collapsible: true,
     active: false
```

```
    });
     });
    </script>
  {% endblock %}
</html>
```

### A.6.2 app/Resources/views/headerbar.html.twig

```
{% extends "::layout.html.twig" %}

{% block shell %}
 {% block header %}
  <header>
  <div class="navbar navbar-default navbar-fixed-top" role="navigation">
   <div class="container">
    <div class="navbar-header">
     <div id='homepage' class='navbar-brand' title="Poavre, le compagnon de votre SEL">
      <a href="{{ path('ml_service_add') }}"><img src="{{ asset('img/poavr-24.png') }}"
          /></a>
      <a href="{{ path('ml_service_add') }}">Poavre</a>
     </div>
    </div>
    <div class="collapse navbar-collapse">
     <ul class="nav navbar-nav">
      <li>
       <a href="{{ path('ml_service_homepage') }}" title="Les prestations disponibles">
           Prestations</a>
      </li>
      <li>
       <a href="{{ path('ml_service_add') }}" title="Proposer une prestation !"><i class=
           "glyphicon glyphicon-plus-sign"></i></a>
      </li>
      <li>
       <a href="{{ path('ml_group_homepage') }}" title="Regroupez-vous et proposez vos
           prestations !">Groupes</a>
      </li>
      <li>
       <a href="{{ path('ml_group_creation_group') }}" title="Créez votre propre groupe !
           "><i class="glyphicon glyphicon-plus-sign"></i></a>
      </li>
      <li>
       <a href="{{ path('ml_forum_homepage') }}" title="Discutez de tout et n'importe
           quoi !">Forum</a>
      </li>
     </ul>
     <ul class="nav navbar-nav navbar-right">
      <li style = "margin-right: 10px;">
       <img src="{{ asset('img/portefeuille.png') }}"/> {{user.account.amount}}
      </li>
      <li>
       <img src="{{ asset('img/yinyang.png') }}"/> {{ user.karma }}
      </li>
      <li>
       <a href="{{ path('ml_user_homepage') }}">{{ user.firstName }} {{ user.lastName
           }}</a>
      </li>
      <li>
       <div id = "logout">
        <form action="{{ path('ml_user_deconnection') }}" method="post">
         <button class="btn btn-primary btn-xs" type="submit">Déconnexion <i class="icon-
             white glyphicon glyphicon-ok-sign"></i> </button>
        </form>
       </div>
      </li>
      {% if user.master or user.moderator %}
       <li>
        <div id = 'admin' class = 'pack-end'>
         <form action="{{ path('ml_administration_homepage') }}" method="post">
```

```
            <button class="btn btn-primary btn-xs" type="submit">Administration <i class="
                icon-white glyphicon glyphicon-ok-sign"></i> </button>
           </form>
          </div>
         </li>
        {% endif %}
       </ul>
      </div>
     </div>
    </header>
{% endblock %}

<div class="container-fluid">
 <div class="row">
  <div class="col-sm-3 col-md-2 sidebar">
   <ul class="nav nav-sidebar">
    <li class="active"> <p> </p></li>
    <li><a href="{{ path('ml_service_see_mine') }}">Mes services proposés</a></li>
    <li><a href="{{ path('ml_transaction_eval_index') }}">Service à évaluer</a></li>
    <li><a href="{{path('ml_transaction_homepage')}}">Transactions</a></li>
   </ul>
   {% block sidebar %}
   {% endblock %}
  </div>
  <div class="sidebar-right">
   <div class="sidebar-container">
    {% block body %}
    {% endblock %}
   </div>
  </div>
 </div>
</div>
{% endblock %}
```

### A.6.3 src/Ml/TransactionBundle/Resources/views/Transaction/index.html.twig

```
{% extends "MlTransactionBundle::layout.html.twig" %}

{% block title %}
 {{ parent() }} - Transaction - Liste
{% endblock %}

{% block transaction_body %}
<div class="well">
    <h2>Votre compte</h2>
    Montant : {{user.account.amount}} <br />
    Découvert autorisé : {{user.account.authorizedOverdraft}}  <br /><br>
    <a href="{{path('ml_transaction_payment')}}">Nouveau paiement</a>
</div>

<div class="well">
    <h2>Débits</h2>
    <table align = "center" class = "table">
        <tr>
            <td align = "center"><b>Date</th>
            <td align = "center"><b>Montant</th>
            <td align = "center"><b>Créditeur</th>
            <td align = "center"><b>Libellé</th>
        </tr>
        {% if outTransactions is defined%}
            {% if outTransactions is not null%}
                {% for transaction in outTransactions%}
                <tr>
                    <td align = "center">{{transaction.date|date("d/m/y")}}</td>
                    <td align = "center" style="color:red;">-{{transaction.amount}}</td>
                    <td align = "center">{{transaction.creditedAccount.owner.login}}</td
                        >
                    <td align = "center">{{transaction.flag}}</td>
                </tr>
```

```
                {% endfor %}
    {% else %}
     <tr>
      <td align = "center" colspan = "4">
       Aucun débit
      </td>
     </tr>
            {% endif %}
   {% else %}
    <tr>
     <td align = "center" colspan = "4">
      Aucun débit
     </td>
    </tr>
        {% endif %}
     </table>
 </div>
 <br><br>
 <div class="well">
     <h2>Crédits</h2>
     <table align = "center" class = "table">
         <tr>
             <td align = "center"><b>Date</b></td>
             <td align = "center"><b>Montant</b></td>
             <td align = "center"><b>Débiteur</b></td>
             <td align = "center"><b>Libellé</b></td>
         </tr>
         {% if inTransactions is defined%}
             {% if inTransactions is not null%}
                 {% for transaction in inTransactions%}
                 <tr>
                     <td align = "center">{{transaction.date|date("d/m/y")}}</td>
                     <td align = "center" style="color:green;">{{transaction.amount}}</td
                         >
                     <td align = "center">{{transaction.debitedAccount.owner.login}}</td>
                     <td align = "center">{{transaction.flag}}</td>
                 </tr>
                 {% endfor %}
    {% else %}
     <tr>
      <td align = "center" colspan = "4">
       Aucun crédit
      </td>
     </tr>
            {% endif %}
         {% else %}
    <tr>
     <td align = "center" colspan = "4">
      Aucun crédit
     </td>
    </tr>
   {% endif %}
     </table>
 </div>

{% endblock %}
```

### A.6.4 src/Ml/UserBundle/Resources/views/User/edit.html.twig

```
{% extends "MlUserBundle::layout.html.twig" %}

{% block title %}
 {{ parent() }} - Edition du compte membre
{% endblock %}

{% block user_body %}
 <div class="jumbotron">
  <h3 style="text-align:center">Editez votre profil </h3>
 </div>
```

```
 <div class="well" align = "center">
  <form action="{{ path('ml_user_edit', {'id': user.id}) }}" method="post" {{
      form_enctype(form) }}>
    {{ form_widget(form) }}
    <button type = "submit" class = "btn btn-success" id = "validate"> Valider</button>
    <button type  = "reset" class = "btn btn-danger" id = "zero">Remettre a zéro</button>
  </form>
 </div>
{% endblock %}
```

# Annexe B

# Documentation des espaces de nommage

## B.1 Référence de l'espace de nommage Ml

### Espaces de nommage

— AdministrationBundle
— ForumBundle
— GroupBundle
— HomeBundle
— libServices
— ServiceBundle
— TransactionBundle
— UserBundle

## B.2 Référence de l'espace de nommage Ml\AdministrationBundle

### Espaces de nommage

— Controller

### Classes

— class MlAdministrationBundle

## B.3 Référence de l'espace de nommage Ml\AdministrationBundle\Contro

### Classes

— class AdministrationController

## B.4 Référence de l'espace de nommage Ml\ForumBundle

### Espaces de nommage

— Controller
— Form

### Classes

— class MlForumBundle

## B.5 Référence de l'espace de nommage Ml\ForumBundle\Controller

### Classes

— class ForumController

## B.6 Référence de l'espace de nommage Ml\ForumBundle\Form

### Classes

— class CommentType
— class TopicType

## B.7 Référence de l'espace de nommage Ml\GroupBundle

### Espaces de nommage

— Controller

### Classes

— class MlGroupBundle

## B.8 Référence de l'espace de nommage Ml\GroupBundle\Controller

### Classes

— class GroupController

## B.9 Référence de l'espace de nommage Ml\HomeBundle

### Espaces de nommage

— Controller

### Classes

— class MlHomeBundle

## B.10 Référence de l'espace de nommage Ml\HomeBundle\Controller

### Classes

— class HomeController

## B.11 Référence de l'espace de nommage Ml\libServices

### Classes

— class ReservationTested
— class SessionTested

## B.12 Référence de l'espace de nommage Ml\ServiceBundle

### Espaces de nommage

— Controller
— Form

### Classes

— class MlServiceBundle

## B.13 Référence de l'espace de nommage Ml\ServiceBundle\Controller

### Classes

— class ServiceController

## B.14 Référence de l'espace de nommage Ml\ServiceBundle\Form

### Classes

— class BasicType
— class CarpoolingType
— class CouchSurfingType
— class SaleType

## B.15 Référence de l'espace de nommage Ml\TransactionBundle

### Espaces de nommage

— Controller
— Exception

### Classes

— class MlTransactionBundle

## B.16 Référence de l'espace de nommage Ml\TransactionBundle\Controlle

### Classes

— class EvaluationController
— class TransactionController

## B.17 Référence de l'espace de nommage Ml\TransactionBundle\Exceptio

### Classes

— class RefusedTransactionException
— interface TransactionExceptionInterface
— class TransactionException

## B.18 Référence de l'espace de nommage Ml\UserBundle

### Espaces de nommage

— Controller
— Form

### Classes

— class MlUserBundle

## B.19 Référence de l'espace de nommage Ml\UserBundle\Controller

### Classes

— class UserController

## B.20 Référence de l'espace de nommage Ml\UserBundle\Form

### Classes

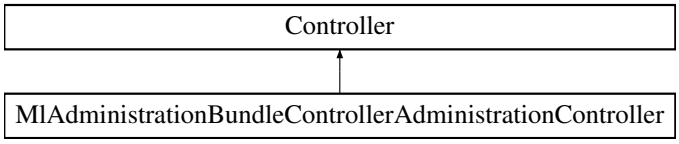
— class UserType

# Annexe C

# Documentation des classes

## C.1 Référence de la classe Ml\AdministrationBundle\Controller\Adminis Controller

Graphe d'héritage de Ml\AdministrationBundle\Controller\AdministrationController :




### Fonctions membres publiques

— indexAction ()
— banAction ()
— grantAction ()
— grantMasterAction ()
— deleteGroupAction ()
— deleteServiceAction ()
— deleteCommentAction ()
— deleteTopicAction ()

### C.1.1 Description détaillée

Administration Controller extending Controller This one is used to manage exchanges between users (services, groups, topics, comments, ...) and users themselves

### C.1.2 Documentation des fonctions membres

**Ml\AdministrationBundle\Controller\AdministrationController : :banAction ( )**

Ban an user (set visible attribute to false)

Renvoie

Redirection to ml_administration_homepage

**Ml\AdministrationBundle\Controller\AdministrationController : :deleteCommentAction ( )**

Delete a comment from database

Renvoie

Redirection to ml_administration_homepage

**Ml\AdministrationBundle\Controller\AdministrationController : :deleteGroupAction ( )**

Delete a group from database Set associatedGroup attribute from Services to NULL

Renvoie

Redirection to ml_administration_homepage

**Ml\AdministrationBundle\Controller\AdministrationController : :deleteServiceAction ( )**

Delete a service from database

Renvoie

Redirection to ml_administration_homepage

**Ml\AdministrationBundle\Controller\AdministrationController : :deleteTopicAction ( )**

Delete a topic from database

Renvoie

Redirection to ml_administration_homepage

**Ml\AdministrationBundle\Controller\AdministrationController : :grantAction ( )**

Grant an user to moderator

Renvoie

Redirection to ml_administration_homepage

**Ml\AdministrationBundle\Controller\AdministrationController : :grantMasterAction ( )**

Grant an user to Master

Renvoie

Redirection to ml_administration_homepage

**Ml\AdministrationBundle\Controller\AdministrationController : :indexAction ( )**

Display all data (services, groups, topics, comments and users) and allow administrators to manage them

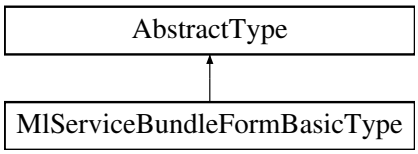
Renvoie

Twig template MlAdministrationBundle :Administration :index.html.twig

La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/AdministrationBundle/Controller/AdministrationController.php

## C.2 Référence de la classe Ml\ServiceBundle\Form\BasicType

Graphe d'héritage de Ml\ServiceBundle\Form\BasicType :




### Fonctions membres publiques

— buildForm (FormBuilderInterface $builder, array $options)
— setDefaultOptions (OptionsResolverInterface $resolver)
— getName ()

### C.2.1 Description détaillée

Basic form

### C.2.2 Documentation des fonctions membres

**Ml\ServiceBundle\Form\BasicType : :buildForm ( FormBuilderInterface *$builder,* array *$options* )**

Paramètres

| | | |
|---|---|---|
| Form-Builder-Interface | *$builder* | |
| array | *$options* | |

**Ml\ServiceBundle\Form\BasicType : :getName ( )**

Get name

Renvoie

string

**Ml\ServiceBundle\Form\BasicType : :setDefaultOptions ( OptionsResolverInterface *$resolver* )**

Set Default options

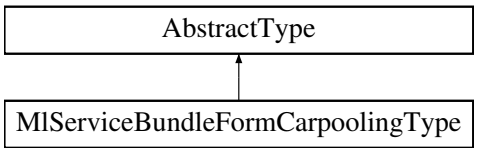
Paramètres

| | | |
|---|---|---|
| Options-Resolver-Interface | *$resolver* | |

La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/ServiceBundle/Form/BasicType.php

## C.3 Référence de la classe Ml\ServiceBundle\Form\Carpooling-Type

Graphe d'héritage de Ml\ServiceBundle\Form\CarpoolingType :




### Fonctions membres publiques

— buildForm (FormBuilderInterface $builder, array $options)
— setDefaultOptions (OptionsResolverInterface $resolver)
— getName ()

### C.3.1 Description détaillée

Carpooling form

### C.3.2 Documentation des fonctions membres

**Ml\ServiceBundle\Form\CarpoolingType : :buildForm ( FormBuilderInterface *$builder,* array *$options* )**

Paramètres

| | | |
|---|---|---|
| Form-Builder-Interface | *$builder* | |
| array | *$options* | |

**Ml\ServiceBundle\Form\CarpoolingType : :getName ( )**

Get name

Renvoie

string

**Ml\ServiceBundle\Form\CarpoolingType : :setDefaultOptions ( OptionsResolverInterface *$resolver* )**

Set Default options

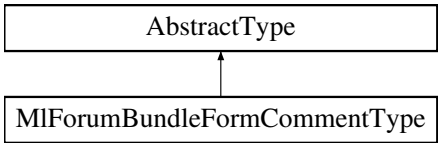
Paramètres

| | | |
|---|---|---|
| Options-Resolver-Interface | *$resolver* | |

La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/ServiceBundle/Form/CarpoolingType.php

## C.4 Référence de la classe Ml\ForumBundle\Form\CommentType

Graphe d'héritage de Ml\ForumBundle\Form\CommentType :




### Fonctions membres publiques

— buildForm (FormBuilderInterface $builder, array $options)
— setDefaultOptions (OptionsResolverInterface $resolver)
— getName ()

### C.4.1 Description détaillée

Comment form

### C.4.2 Documentation des fonctions membres

**Ml\ForumBundle\Form\CommentType : :buildForm ( FormBuilderInterface *$builder,* array *$options* )**

Paramètres

| | | |
|---|---|---|
| Form-Builder-Interface | *$builder* | |
| array | *$options* | |

**Ml\ForumBundle\Form\CommentType : :getName ( )**

Get name

Renvoie

string

**Ml\ForumBundle\Form\CommentType : :setDefaultOptions ( OptionsResolverInterface *$resolver* )**

Set Default options

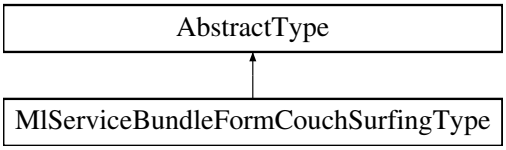
Paramètres

| | | |
|---|---|---|
| Options-Resolver-Interface | *$resolver* | |

La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/ForumBundle/Form/CommentType.php

## C.5 Référence de la classe Ml\ServiceBundle\Form\CouchSurfing-Type

Graphe d'héritage de Ml\ServiceBundle\Form\CouchSurfingType :




### Fonctions membres publiques

— buildForm (FormBuilderInterface $builder, array $options)
— setDefaultOptions (OptionsResolverInterface $resolver)
— getName ()

### C.5.1 Description détaillée

CouchSurfing form

### C.5.2 Documentation des fonctions membres

**Ml\ServiceBundle\Form\CouchSurfingType : :buildForm ( FormBuilderInterface *$builder,* array *$options* )**

Paramètres

| | | |
|---|---|---|
| Form-Builder-Interface | *$builder* | |
| array | *$options* | |

**Ml\ServiceBundle\Form\CouchSurfingType : :getName ( )**

Get name

Renvoie

string

**Ml\ServiceBundle\Form\CouchSurfingType : :setDefaultOptions ( OptionsResolverInterface *$resolver* )**

Set Default options

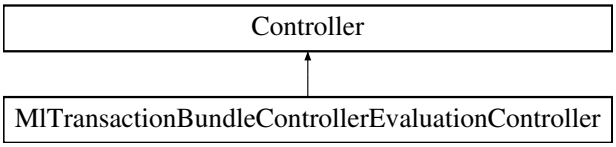
Paramètres

| | | |
|---|---|---|
| Options-Resolver-Interface | *$resolver* | |

La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/ServiceBundle/Form/CouchSurfingType.php

## C.6 Référence de la classe Ml\TransactionBundle\Controller\Evaluation-Controller

Graphe d'héritage de Ml\TransactionBundle\Controller\EvaluationController :




### Fonctions membres publiques

— indexAction ()
— evaluationAction ($serviceType, $id)
— evalKarma ($user)

### C.6.1 Description détaillée

Evaluation Controller extending Controller This one is used to evaluate a service and to display all evaluations now requested from current user

### C.6.2 Documentation des fonctions membres

**Ml\TransactionBundle\Controller\EvaluationController : :evalKarma ( *$user* )**

Give karma from an eval to $user

Paramètres

| | | |
|---|---|---|
| User | *$user* | |

**Ml\TransactionBundle\Controller\EvaluationController : :evaluationAction ( *$serviceType, $id* )**

Allow current user to evaluate a service that another user made for him

Paramètres

| | | |
|---|---|---|
| string | *$serviceType* | |
| int | *$id* | |

Renvoie

Twig template MlTransactionBundle :Transaction :evaluation.html.twig

**Ml\TransactionBundle\Controller\EvaluationController : :indexAction ( )**

Display all evaluations now requested from current user

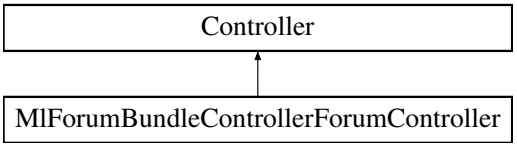
Renvoie

Twig template MlTransactionBundle :Transaction :index_eval.html.twig

La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/TransactionBundle/Controller/EvaluationController.php

## C.7 Référence de la classe Ml\ForumBundle\Controller\ForumController

Graphe d'héritage de Ml\ForumBundle\Controller\ForumController :




### Fonctions membres publiques

— indexAction ()
— newTopicAction ()
— seeTopicAction ($topic=NULL)

### C.7.1 Description détaillée

Forum Controller extending Controller This one is used to display the forum and its components (topics, comments, likes, dislikes, ...) and to manage it.

### C.7.2 Documentation des fonctions membres

**Ml\ForumBundle\Controller\ForumController : :indexAction ( )**

Display all topics ordered by ratio(likes/dislikers) and allow a user to access a topic or to create a new one

Renvoie

Twig template MlForumBundle :Forum :index.html.twig

**Ml\ForumBundle\Controller\ForumController : :newTopicAction ( )**

Display the form allowing a user to create a new topic

Renvoie

Twig template MlForumBundle :Forum :new_topic.html.twig

**Ml\ForumBundle\Controller\ForumController : :seeTopicAction ( *$topic* = `NULL` )**

Display a topic with its details and comments and allow a user to comment the topic or to like/dislike it

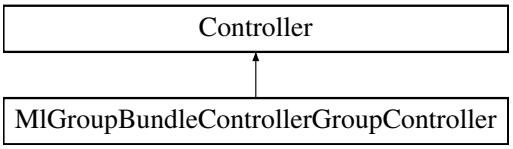
Renvoie

Twig template MlForumBundle :Forum :see_topic.html.twig

La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/ForumBundle/Controller/ForumController.php

## C.8 Référence de la classe Ml\GroupBundle\Controller\GroupController

Graphe d'héritage de Ml\GroupBundle\Controller\GroupController :



### Fonctions membres publiques

— indexAction ()
— creationGroupAction ()
— displayGroupAction ($group_id=null)
— addUserAction ()
— joinGroupAction ($group_id=NULL)
— deleteUserAction ()
— refuseUserAction ()
— leaveGroupAction ($group_id=null)
— deleteGroupAction ($group_id=NULL)

## C.8.1 Description détaillée

Group Controller extending Controller This one is used to display groups and all its data and to manage groups for groups' creators or to join a group

## C.8.2 Documentation des fonctions membres

**Ml\GroupBundle\Controller\GroupController : :addUserAction ( )**

Add a user to the group

Renvoie

Redirection to ml_group_display_group

**Ml\GroupBundle\Controller\GroupController : :creationGroupAction ( )**

Allow a user to create a group

Renvoie

Redirection to ml_group_display_group

**Ml\GroupBundle\Controller\GroupController : :deleteGroupAction ( *$group_id = NULL* )**

Delete a group and all its data (set associatedGroup of Services to NULL)

Renvoie

Redirection to ml_home_homepage

**Ml\GroupBundle\Controller\GroupController : :deleteUserAction ( )**

Remove a user from the group

Renvoie

Redirection to ml_group_display_group

**Ml\GroupBundle\Controller\GroupController : :displayGroupAction ( *$group_id = null* )**

Display a group with all its data

Renvoie

Twig template MlGroupBundle :Group :display_group.html.twig

**Ml\GroupBundle\Controller\GroupController : :indexAction ( )**

Display all groups with data

Renvoie

Twig template MlGroupBundle :Group :groups.html.twig

**Ml\GroupBundle\Controller\GroupController : :joinGroupAction ( *$group_id = NULL* )**

Send a request to join the group

Renvoie

Redirection to ml_group_display_group

**Ml\GroupBundle\Controller\GroupController : :leaveGroupAction ( *$group_id = null* )**

Delete a user from the group because he decided to leave the group

Renvoie

Redirection to ml_home_homepage

**Ml\GroupBundle\Controller\GroupController : :refuseUserAction ( )**

Refuse a request to join the group

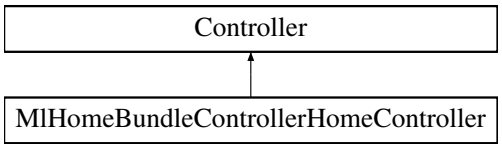
Renvoie

Redirection to ml_group_display_group

La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/GroupBundle/Controller/GroupController.php

## C.9 Référence de la classe Ml\HomeBundle\Controller\HomeController

Graphe d'héritage de Ml\HomeBundle\Controller\HomeController :




### Fonctions membres publiques

— indexAction ()
— developersAction ()

### C.9.1 Description détaillée

Home Controller extending Controller This one is used for redirections and to access homepage

### C.9.2 Documentation des fonctions membres

**Ml\HomeBundle\Controller\HomeController : :developersAction ( )**

Display MlHomeBundle :Home :developers.html.twig

Renvoie

Twig template MlHomeBundle :Home :developers.html.twig

**Ml\HomeBundle\Controller\HomeController : :indexAction ( )**

Redirect to ml_user_see if connected, else, display MlHomeBundle :Home :index.html.twig

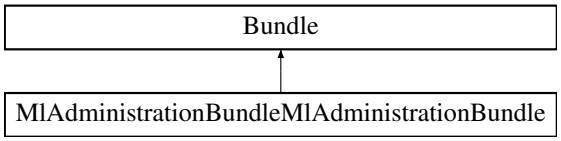
Renvoie

Twig template MlHomeBundle :Home :index.html.twig

La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/HomeBundle/Controller/HomeController.php

## C.10 Référence de la classe Ml\AdministrationBundle\MlAdministrationBundle

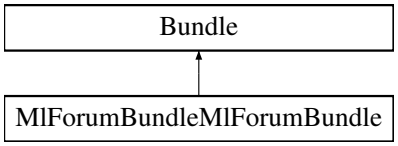
Graphe d'héritage de Ml\AdministrationBundle\MlAdministrationBundle :




La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/AdministrationBundle/MlAdministrationBundle.php

## C.11 Référence de la classe Ml\ForumBundle\MlForumBundle

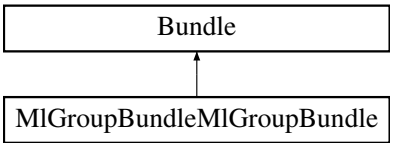
Graphe d'héritage de Ml\ForumBundle\MlForumBundle :




La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/ForumBundle/MlForumBundle.php

## C.12 Référence de la classe Ml\GroupBundle\MlGroupBundle

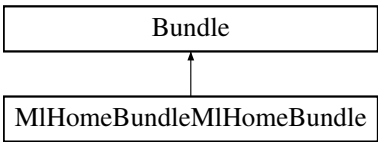
Graphe d'héritage de Ml\GroupBundle\MlGroupBundle :




La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/GroupBundle/MlGroupBundle.php

## C.13 Référence de la classe Ml\HomeBundle\MlHomeBundle

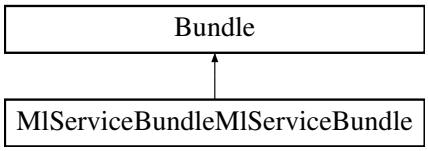
Graphe d'héritage de Ml\HomeBundle\MlHomeBundle :




La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/HomeBundle/MlHomeBundle.php

## C.14 Référence de la classe Ml\ServiceBundle\MlServiceBundle

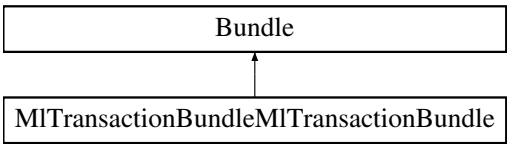
Graphe d'héritage de Ml\ServiceBundle\MlServiceBundle :




La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/ServiceBundle/MlServiceBundle.php

## C.15 Référence de la classe Ml\TransactionBundle\MlTransaction-Bundle

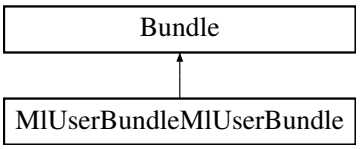
Graphe d'héritage de Ml\TransactionBundle\MlTransactionBundle :




La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/TransactionBundle/MlTransactionBundle.php

## C.16 Référence de la classe Ml\UserBundle\MlUserBundle

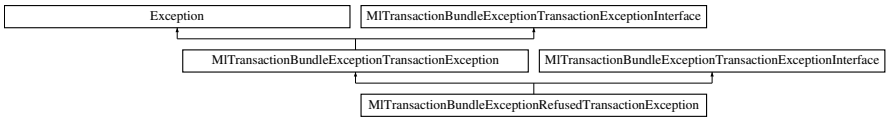
Graphe d'héritage de Ml\UserBundle\MlUserBundle :




La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/UserBundle/MlUserBundle.php

## C.17 Référence de la classe Ml\TransactionBundle\Exception\RefusedTransactionException

Graphe d'héritage de Ml\TransactionBundle\Exception\RefusedTransactionException :




### Fonctions membres publiques

— __construct ($message=null, $code=0)

### C.17.1 Documentation des constructeurs et destructeur

**Ml\TransactionBundle\Exception\RefusedTransactionException : :__construct ( *$message = null, $code = 0* )**

La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/TransactionBundle/Exception/RefusedTransactionException.php

## C.18 Référence de la classe Ml\libServices\ReservationTested

### Fonctions membres publiques

— canReserve ($user, $service, $em)
— canPay ($user, $amount, $em)

### C.18.1 Description détaillée

Check if user can still reserve a service

### C.18.2 Documentation des fonctions membres

**Ml\libServices\ReservationTested : :canPay ( *$user, $amount, $em* )**

Check if user can pay

Paramètres

| | | |
|---|---|---|
| User | *$user* | |
| int | *$amount* | |
| Entity-Manager | *$em* | |

Renvoie

bool

**Ml\libServices\ReservationTested : :canReserve ( *$user,* *$service,* *$em* )**

Check if user can still reserve a service

Paramètres

| | | |
|---|---|---|
| User | *$user* | |
| Service | *$service* | |
| Entity-Manager | *$em* | |

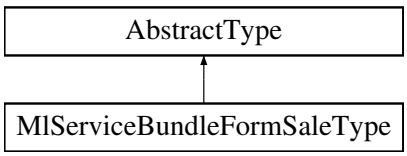
Renvoie

bool

La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/libServices/ReservationTested.php

## C.19 Référence de la classe Ml\ServiceBundle\Form\SaleType

Graphe d'héritage de Ml\ServiceBundle\Form\SaleType :




### Fonctions membres publiques

— buildForm (FormBuilderInterface $builder, array $options)
— setDefaultOptions (OptionsResolverInterface $resolver)
— getName ()

### C.19.1 Description détaillée

Sale form

### C.19.2 Documentation des fonctions membres

**Ml\ServiceBundle\Form\SaleType : :buildForm ( FormBuilderInterface *$builder,* array *$options* )**

Paramètres

| | | |
|---|---|---|
| Form-Builder-Interface | *$builder* | |
| array | *$options* | |

**Ml\ServiceBundle\Form\SaleType : :getName ( )**

Get name

Renvoie

string

**Ml\ServiceBundle\Form\SaleType : :setDefaultOptions ( OptionsResolverInterface *$resolver* )**

Set Default options

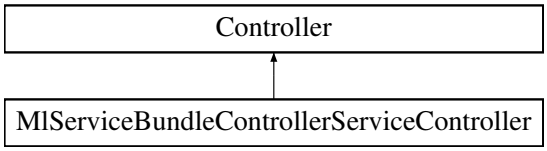
Paramètres

| | | |
|---|---|---|
| Options-Resolver-Interface | *$resolver* | |

La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/ServiceBundle/Form/SaleType.php

## C.20 Référence de la classe Ml\ServiceBundle\Controller\ServiceController

Graphe d'héritage de Ml\ServiceBundle\Controller\ServiceController :




### Fonctions membres publiques

— indexAction ()
— addServiceAction ()
— seeBasicAction ($basic=null)
— addBasicAction ()
— deleteBasicAction ()
— seeCarpoolingAction ($carpooling=null)
— addCarpoolingAction ()
— deleteCarpoolingAction ()
— addCouchSurfingAction ()
— seeCouchSurfingAction ($couchsurfing=null)
— deleteCouchsurfingAction ()
— addSaleAction ()
— seeSaleAction ($sale=null)
— deleteSaleAction ()
— seeMyServicesAction ()
— serviceDoneAction ()

### C.20.1 Description détaillée

Service Controller extending Controller This one is used to manage service (create, delete, reserve a service) and to display them

### C.20.2 Documentation des fonctions membres

**Ml\ServiceBundle\Controller\ServiceController : :addBasicAction ( )**

Add a basic service in database

Renvoie

Twig MlServiceBundle :Service :add_basic.html.twig

**Ml\ServiceBundle\Controller\ServiceController : :addCarpoolingAction ( )**

Add a carpooling service in database

Renvoie

Twig MlServiceBundle :Service :add_carpooling.html.twig

**Ml\ServiceBundle\Controller\ServiceController : :addCouchSurfingAction ( )**

Add a couchsurfing service in database

Renvoie

Twig MlServiceBundle :Service :add_couchsurfing.html.twig

**Ml\ServiceBundle\Controller\ServiceController : :addSaleAction ( )**

Add a sale service in database

Renvoie

Twig MlServiceBundle :Service :add_sale.html.twig

**Ml\ServiceBundle\Controller\ServiceController : :addServiceAction ( )**

Add a service in database (user can choose the kind of service he wants to create)

Renvoie

Twig MlServiceBundle :Service :add_service.html.twig

**Ml\ServiceBundle\Controller\ServiceController : :deleteBasicAction ( )**

Delete a basic service from database

Renvoie

Redirection to ml_service_homepage

**Ml\ServiceBundle\Controller\ServiceController : :deleteCarpoolingAction ( )**

Delete a carpooling service from database

Renvoie

Redirection to ml_service_homepage

**Ml\ServiceBundle\Controller\ServiceController : :deleteCouchsurfingAction ( )**

Delete a couchsurfing service from database

Renvoie

Redirection ml_service_homepage

**Ml\ServiceBundle\Controller\ServiceController : :deleteSaleAction ( )**

Delete a sale service from database

Renvoie

Redirection ml_service_homepage

**Ml\ServiceBundle\Controller\ServiceController : :indexAction ( )**

Display all services and data

Renvoie

Twig MlServiceBundle :Service :index.html.twig

**Ml\ServiceBundle\Controller\ServiceController : :seeBasicAction ( *$basic* = *null* )**

Display a service and its data

Paramètres

| | | |
|---|---|---|
| int | *$basic* | |

Renvoie

Twig MlServiceBundle :Service :see_basic.html.twig

**Ml\ServiceBundle\Controller\ServiceController : :seeCarpoolingAction ( *$carpooling* = *null* )**

Display a carpooling service and its data

Paramètres

| | | |
|---|---|---|
| int | *$carpooling* | |

Renvoie

Twig MlServiceBundle :Service :see_carpooling.html.twig

**Ml\ServiceBundle\Controller\ServiceController : :seeCouchSurfingAction ( *$couchsurfing* = *null* )**

Display a couchsurfing service and its data

Paramètres

| | | |
|---|---|---|
| int | *$couchsurfing* | |

Renvoie

Twig MlServiceBundle :Service :see_couchsurfing.html.twig

**Ml\ServiceBundle\Controller\ServiceController : :seeMyServicesAction ( )**

Display all services of current user

Renvoie

Twig MlServiceBundle :Service :index_my_services.html.twig

**Ml\ServiceBundle\Controller\ServiceController : :seeSaleAction ( *$sale* = *null* )**

Display a sale service and its data

Paramètres

| | | |
|---|---|---|
| int | *$sale* | |

Renvoie

Twig MlServiceBundle :Service :see_sale.html.twig

**Ml\ServiceBundle\Controller\ServiceController : :serviceDoneAction ( )**

Set a service as done

Renvoie

Redirection ml_service_see_mine

La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/ServiceBundle/Controller/ServiceController.php

## C.21 Référence de la classe Ml\libServices\SessionTested

### Fonctions membres publiques

— sessionExist ($req)

### C.21.1 Description détaillée

Check if a session is set (user is authenticated)

### C.21.2 Documentation des fonctions membres

**Ml\libServices\SessionTested : :sessionExist ( *$req* )**

Check if a session is set (user is authenticated)

Paramètres

| | | |
|---|---|---|
| array | *$req* | |

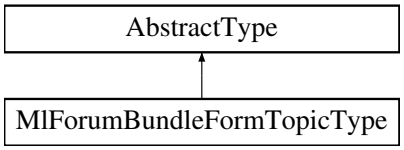
Renvoie

string $login

La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/libServices/SessionTested.php

## C.22 Référence de la classe Ml\ForumBundle\Form\TopicType

Graphe d'héritage de Ml\ForumBundle\Form\TopicType :




### Fonctions membres publiques

— buildForm (FormBuilderInterface $builder, array $options)
— setDefaultOptions (OptionsResolverInterface $resolver)
— getName ()

### C.22.1 Description détaillée

Topic form

### C.22.2 Documentation des fonctions membres

**Ml\ForumBundle\Form\TopicType : :buildForm ( FormBuilderInterface *$builder,* array *$options* )**

Paramètres

| | | |
|---|---|---|
| Form-Builder-Interface | *$builder* | |
| array | *$options* | |

**Ml\ForumBundle\Form\TopicType : :getName ( )**

Get name

Renvoie

string

**Ml\ForumBundle\Form\TopicType : :setDefaultOptions ( OptionsResolverInterface *$resolver* )**

Set Default options

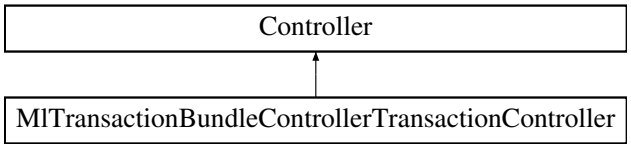
Paramètres

| | | |
|---|---|---|
| Options-Resolver-Interface | *$resolver* | |

La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/ForumBundle/Form/TopicType.php

## C.23 Référence de la classe Ml\TransactionBundle\Controller\Transactio Controller

Graphe d'héritage de Ml\TransactionBundle\Controller\TransactionController :




### Fonctions membres publiques

— indexAction ()
— paymentAction ()

### C.23.1 Description détaillée

Transaction Controller extending Controller This one is used for transactions between current user and another users

### C.23.2 Documentation des fonctions membres

**Ml\TransactionBundle\Controller\TransactionController : :indexAction ( )**

Display all transactions and able current user to pay someone

Renvoie

Twig template MlTransactionBundle :Transaction :index.html.twig

**Ml\TransactionBundle\Controller\TransactionController : :paymentAction ( )**

Pay someone (from current user to another one)

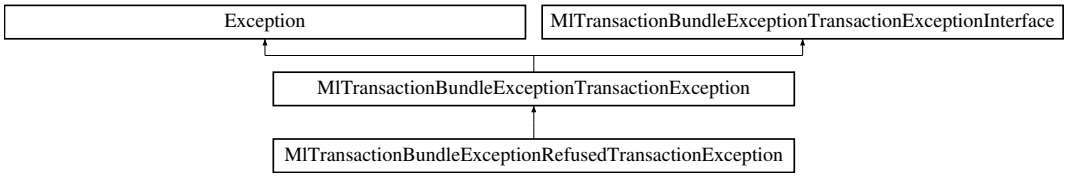
Renvoie

Twig template MlTransactionBundle :Transaction :payment.html.twig

La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/TransactionBundle/Controller/TransactionController.php

## C.24 Référence de la classe Ml\TransactionBundle\Exception\Transactio Exception

Graphe d'héritage de Ml\TransactionBundle\Exception\TransactionException :




### Fonctions membres publiques

— __construct ($message=null, $code=0)

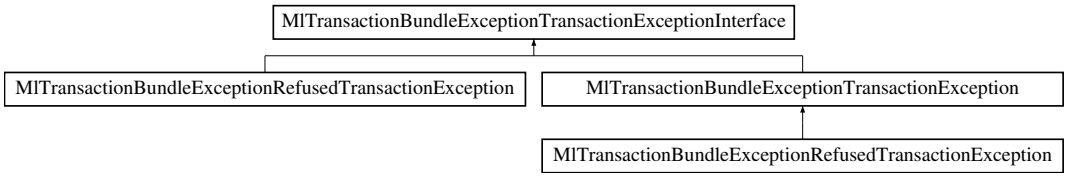
### C.24.1 Documentation des constructeurs et destructeur

**Ml\TransactionBundle\Exception\TransactionException : :__construct (** ***$message*** *= null,* ***$code*** *= 0* **)**

La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/TransactionBundle/Exception/TransactionException.php

## C.25 Référence de l'interface Ml\TransactionBundle\Exception\Transact ExceptionInterface

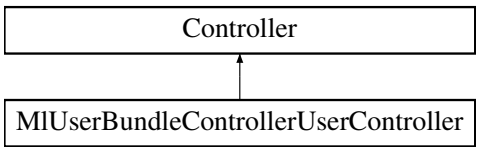
Graphe d'héritage de Ml\TransactionBundle\Exception\TransactionExceptionInterface :




La documentation de cette interface a été générée à partir du fichier suivant :
— Symfony/src/Ml/TransactionBundle/Exception/TransactionException.php

## C.26 Référence de la classe Ml\UserBundle\Controller\UserController

Graphe d'héritage de Ml\UserBundle\Controller\UserController :




### Fonctions membres publiques

— indexAction ()
— seeAction ()
— addAction ()
— deleteAction ()
— connectionAction ()
— deconnectionAction ()
— editAction ()

### C.26.1 Description détaillée

User Controller extending Controller This one is used to access all data about current user, to log him in, to log him out, to manage his account

### C.26.2 Documentation des fonctions membres

**Ml\UserBundle\Controller\UserController : :addAction ( )**

Add a user in database

Renvoie

Twig template MlUserBundle :User :add_user.html.twig or Redirection to ml_user_see

**Ml\UserBundle\Controller\UserController : :connectionAction ( )**

Connect a user (set session)

Renvoie

Twig template MlUserBundle :User :see.html.twig

**Ml\UserBundle\Controller\UserController : :deconnectionAction ( )**

Disconnect a user (destroy session)

Renvoie

Redirection to ml_user_add

**Ml\UserBundle\Controller\UserController : :deleteAction ( )**

Delete a user from database

Renvoie

Twig template MlUserBundle :User :delete.html.twig if user can't leave Poavre or Redirection to ml_user_deconnection

**Ml\UserBundle\Controller\UserController : :editAction ( )**

Edit current user's profile

Renvoie

Twig Template MlUserBundle :User :edit.html.twig

**Ml\UserBundle\Controller\UserController : :indexAction ( )**

If connected display all data about current user and allow him to manage his account and to access all benefits of Poavre (Services, Groups, Forum, ...) else allows current user to connect or to create an account

Renvoie

Twig template MlUserBundle :User :see.html.twig or Redirection to ml_user_add

**Ml\UserBundle\Controller\UserController : :seeAction ( )**

Display all data about current user and allow him to manage his account and to access all benefits of Poavre (Services, Groups, Forum, ...)

Renvoie

Twig template MlUserBundle :User :see.html.twig

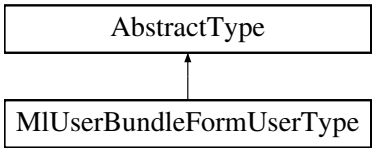
S'il existe, il est envoyé à la vue

La documentation de cette classe a été générée à partir du fichier suivant :

— Symfony/src/Ml/UserBundle/Controller/UserController.php

## C.27 Référence de la classe Ml\UserBundle\Form\UserType

Graphe d'héritage de Ml\UserBundle\Form\UserType :




### Fonctions membres publiques

— buildForm (FormBuilderInterface $builder, array $options)
— setDefaultOptions (OptionsResolverInterface $resolver)
— getName ()

### C.27.1 Description détaillée

User form

### C.27.2 Documentation des fonctions membres

**Ml\UserBundle\Form\UserType : :buildForm ( FormBuilderInterface *$builder,* array *$options* )**

Paramètres

| | | |
|---|---|---|
| Form-Builder-Interface | *$builder* | |
| array | *$options* | |

**Ml\UserBundle\Form\UserType : :getName ( )**

Get name

Renvoie

string

**Ml\UserBundle\Form\UserType : :setDefaultOptions ( OptionsResolverInterface *$resolver* )**

Set Default options

Paramètres

| | | |
|---|---|---|
| Options-Resolver-Interface | *$resolver* | |

La documentation de cette classe a été générée à partir du fichier suivant :
— Symfony/src/Ml/UserBundle/Form/UserType.php

# Annexe D

# Documentation des fichiers

## D.1 Référence du fichier Symfony/src/Ml/AdministrationBundle/-Controller/AdministrationController.php

### Classes

— class Ml\AdministrationBundle\Controller\AdministrationController

### Espaces de nommage

— Ml\AdministrationBundle\Controller

## D.2 Référence du fichier Symfony/src/Ml/AdministrationBundle/-MlAdministrationBundle.php

### Classes

— class Ml\AdministrationBundle\MlAdministrationBundle

### Espaces de nommage

— Ml\AdministrationBundle

## D.3 Référence du fichier Symfony/src/Ml/ForumBundle/Controller/-ForumController.php

### Classes

— class Ml\ForumBundle\Controller\ForumController

### Espaces de nommage

— Ml\ForumBundle\Controller

## D.4 Référence du fichier Symfony/src/Ml/ForumBundle/Form/-CommentType.php

### Classes

— class Ml\ForumBundle\Form\CommentType

### Espaces de nommage

— Ml\ForumBundle\Form

## D.5 Référence du fichier Symfony/src/Ml/ForumBundle/Form/-TopicType.php

### Classes

— class Ml\ForumBundle\Form\TopicType

### Espaces de nommage

— Ml\ForumBundle\Form

## D.6 Référence du fichier Symfony/src/Ml/ForumBundle/MlForum-Bundle.php

### Classes

— class Ml\ForumBundle\MlForumBundle

### Espaces de nommage

— Ml\ForumBundle

## D.7 Référence du fichier Symfony/src/Ml/GroupBundle/Controller/-GroupController.php

### Classes

— class Ml\GroupBundle\Controller\GroupController

### Espaces de nommage

— Ml\GroupBundle\Controller

## D.8 Référence du fichier Symfony/src/Ml/GroupBundle/MlGroup-Bundle.php

### Classes

— class Ml\GroupBundle\MlGroupBundle

### Espaces de nommage

— Ml\GroupBundle

## D.9 Référence du fichier Symfony/src/Ml/HomeBundle/Controller/-HomeController.php

### Classes

— class Ml\HomeBundle\Controller\HomeController

### Espaces de nommage

— Ml\HomeBundle\Controller

## D.10 Référence du fichier Symfony/src/Ml/HomeBundle/MlHome-Bundle.php

### Classes

— class Ml\HomeBundle\MlHomeBundle

### Espaces de nommage

— Ml\HomeBundle

## D.11 Référence du fichier Symfony/src/Ml/libServices/Reservation-Tested.php

### Classes

— class Ml\libServices\ReservationTested

### Espaces de nommage

— Ml\libServices

## D.12 Référence du fichier Symfony/src/Ml/libServices/Session-Tested.php

### Classes

— class Ml\libServices\SessionTested

### Espaces de nommage

— Ml\libServices

## D.13 Référence du fichier Symfony/src/Ml/ServiceBundle/Controller/-ServiceController.php

### Classes

— class Ml\ServiceBundle\Controller\ServiceController

### Espaces de nommage

— Ml\ServiceBundle\Controller

## D.14 Référence du fichier Symfony/src/Ml/ServiceBundle/Form/-BasicType.php

### Classes

— class Ml\ServiceBundle\Form\BasicType

### Espaces de nommage

— Ml\ServiceBundle\Form

## D.15 Référence du fichier Symfony/src/Ml/ServiceBundle/Form/-CarpoolingType.php

### Classes

— class Ml\ServiceBundle\Form\CarpoolingType

### Espaces de nommage

— Ml\ServiceBundle\Form

## D.16 Référence du fichier Symfony/src/Ml/ServiceBundle/Form/-CouchSurfingType.php

### Classes

— class Ml\ServiceBundle\Form\CouchSurfingType

### Espaces de nommage

— Ml\ServiceBundle\Form

## D.17 Référence du fichier Symfony/src/Ml/ServiceBundle/Form/-SaleType.php

### Classes

— class Ml\ServiceBundle\Form\SaleType

### Espaces de nommage

— Ml\ServiceBundle\Form

## D.18 Référence du fichier Symfony/src/Ml/ServiceBundle/MlServiceBundle.php

### Classes

— class Ml\ServiceBundle\MlServiceBundle

### Espaces de nommage

— Ml\ServiceBundle

## D.19 Référence du fichier Symfony/src/Ml/TransactionBundle/Controller/EvaluationController.php

### Classes

— class Ml\TransactionBundle\Controller\EvaluationController

### Espaces de nommage

— Ml\TransactionBundle\Controller

## D.20 Référence du fichier Symfony/src/Ml/TransactionBundle/Controller/TransactionController.php

### Classes

— class Ml\TransactionBundle\Controller\TransactionController

### Espaces de nommage

— Ml\TransactionBundle\Controller

## D.21 Référence du fichier Symfony/src/Ml/TransactionBundle/Exception/RefusedTransactionException.php

### Classes

— class Ml\TransactionBundle\Exception\RefusedTransactionException

### Espaces de nommage

— Ml\TransactionBundle\Exception

## D.22 Référence du fichier Symfony/src/Ml/TransactionBundle/-Exception/TransactionException.php

### Classes

— interface Ml\TransactionBundle\Exception\TransactionExceptionInterface
— class Ml\TransactionBundle\Exception\TransactionException

### Espaces de nommage

— Ml\TransactionBundle\Exception

## D.23 Référence du fichier Symfony/src/Ml/TransactionBundle/-MlTransactionBundle.php

### Classes

— class Ml\TransactionBundle\MlTransactionBundle

### Espaces de nommage

— Ml\TransactionBundle

## D.24 Référence du fichier Symfony/src/Ml/UserBundle/Controller/-UserController.php

### Classes

— class Ml\UserBundle\Controller\UserController

### Espaces de nommage

— Ml\UserBundle\Controller

## D.25 Référence du fichier Symfony/src/Ml/UserBundle/Form/-UserType.php

### Classes

— class Ml\UserBundle\Form\UserType

### Espaces de nommage

— Ml\UserBundle\Form

## D.26 Référence du fichier Symfony/src/Ml/UserBundle/MlUser-Bundle.php

### Classes

— class Ml\UserBundle\MlUserBundle

### Espaces de nommage

— Ml\UserBundle

# Bibliographie


[1] Scott Chacon. *Pro Git*. URL : http://git-scm.com/book.

[2] *Doxygen*. URL : http://www.stack.nl/~dimitri/doxygen/.

[3] *FileZilla*. URL : http://filezilla.fr/.

[4] *Github*. URL : https://github.com/.

[5] *Hostinger*. URL : http://www.hostinger.fr/.

[6] *Le Bon Coin*. URL : http://www.leboncoin.fr/.

[7] *Monnaie libre*. URL : http://fr.wikipedia.org/wiki/Monnaie_libre.

[8] *Poavre*. URL : http://poavre.besaba.com/.

[9] *Producteev*. URL : https://www.producteev.com/.

[10] *Site officiel de CakePHP*. URL : http://cakephp.org/.

[11] *Site officiel de Drupal*. URL : http://drupalfr.org/.

[12] *Site officiel de FuelPHP*. URL : http://fuelphp.com/.

[13] *Site officiel de Symfony*. URL : http://symfony.com/.

[14] *Site officiel de WordPress*. URL : http://fr.wordpress.org/.

[15] *Système d'Echange Local*. URL : http://www.selidaire.org/spip/.

[16] *Tutoriel Openclassroom sur Symfony2*. URL : http://fr.openclassrooms.com/informatique/cours/developpez-votre-site-web-avec-le-framework-symfony2.