

---

# Remerciements

J'aimerais avant tout adresser mes remerciements aux personnes qui m'ont aidé et soutenu durant tout le déroulement de mon stage.

Je tiens à remercier d'abord Monsieur Ahmed Trabelsi, mon encadrant de stage à Logicale, et je lui exprime toute ma gratitude pour l'attention qu'il m'a portée, ses directives et ses conseils qui m'ont servi pour atteindre les objectifs du stage dans les délais convenus.

Je voudrais remercier ensuite Mme Faten Chaieb, mon encadrante à l'INSAT pour la qualité de son enseignement, son encadrement et pour sa participation à l'élaboration de ce rapport.

Je voudrais exprimer aussi ma profonde gratitude à ma famille et tout mes amis pour leur apport moral tout au long de mon parcours dans la vie.

Mon dernier mot s'adresse à tous les membres du jury pour l'honneur qu'ils me font de participer à l'examen de mon travail.

---

# Table des Matières

<b>Liste des Figures</b>	<b>iv</b>
<b>Liste des Tableaux</b>	<b>vi</b>
<b>Introduction Générale</b>	<b>1</b>
<b>I Étude préalable et Cadre du projet</b>	<b>3</b>
1 Présentation de l'entreprise d'accueil . . . . .	3
2 Industrie de la Mode . . . . .	4
3 Réseaux sociaux professionnels . . . . .	6
4 Solutions web de Mode existantes . . . . .	8
4.1 FashionUnited . . . . .	8
4.2 Fashionista . . . . .	8
4.3 ManyMucho . . . . .	9
4.4 Discussion . . . . .	10
5 Projet Thimble Fashion . . . . .	10
6 Méthodologies Agiles . . . . .	12
6.1 Méthodologie SCRUM . . . . .	13
6.2 Scrum appliqué à notre projet . . . . .	14
<b>II Analyse et Modélisation des besoins</b>	<b>17</b>
1 Utilisateurs cibles de la solution . . . . .	17
2 Analyse des besoins . . . . .	18
2.1 Besoins fonctionnels . . . . .	18
2.2 Besoins non-fonctionnels . . . . .	19
3 Modélisation des besoins . . . . .	20
3.1 Diagramme de cas d'utilisation d'un visiteur . . . . .	20
3.2 Diagramme de cas d'utilisation d'un membre amateur . . . . .	22
3.3 Diagramme de cas d'utilisation d'un membre professionnel . . . . .	24
3.4 Diagramme de cas d'utilisation des acteurs système . . . . .	26
<b>III Architecture et Choix Techniques et Conceptuels</b>	<b>28</b>
1 Architecture cible de la solution . . . . .	28
2 Choix Techniques . . . . .	29
2.1 Framework d'implémentation de la plateforme Thimble . . . . .	29

2.2	API de web services . . . . .	32
3	Conception de la solution . . . . .	36
3.1	Diagramme de Packages . . . . .	37
3.2	Diagramme de classes des modèles de domaine . . . . .	39
3.3	Diagramme de Classes : Package API Controllers . . . . .	42
3.4	Diagramme de séquences : Création d'un projet . . . . .	45
3.5	Conception de la base de données . . . . .	47
4	Design patterns employés . . . . .	47
4.1	Design pattern : Repository . . . . .	47
4.2	Design pattern : Factory Method . . . . .	49
<b>IV</b>	<b>Mise en œuvre et Tests</b>	<b>51</b>
1	Architecture finale de la solution . . . . .	51
2	Environnement de travail . . . . .	52
2.1	Environnement matériel . . . . .	52
2.2	Environnement logiciel . . . . .	52
3	Mise en œuvre de l'application Front-office . . . . .	53
4	Mise en œuvre de l'API REST . . . . .	57
4.1	Exemples de web services REST réalisés . . . . .	57
4.2	Sécurisation de l'API . . . . .	59
4.3	Documentation de l'API . . . . .	62
5	Tests de la solution . . . . .	62
5.1	Tests unitaires . . . . .	62
5.2	Tests de l'API REST . . . . .	63
<b>Conclusion Générale et Perspectives</b>		<b>69</b>
<b>Références</b>		<b>70</b>
<b>Annexe 1</b>		<b>72</b>
<b>Annexe 2</b>		<b>77</b>
<b>Annexe 3</b>		<b>79</b>

---

# Liste des Figures

I.1	Logo de Logicale Développement Inc. [3]	3
I.2	Page d'accueil de FashionUnited [10]	8
I.3	Page d'accueil de Fashionista [11]	9
I.4	Page d'accueil de ManyMucho [12]	10
I.5	Vue d'ensemble de Scrum [14]	14
II.1	Les utilisateurs cibles de la solution	18
II.2	Diagramme UseCase d'un visiteur	21
II.3	Diagramme UseCase d'un membre amateur	23
II.4	Diagramme UseCase d'un membre professionnel	25
II.5	Diagramme UseCase des acteurs système	27
III.1	Architecture cible de la solution	28
III.2	Logo du framework Laravel [17]	30
III.3	Échange Client-Serveur REST	33
III.4	Structure d'un message SOAP	35
III.5	Analyse des Protocoles et des Styles d'API 2012 [23]	36
III.6	Diagramme de packages Web et API REST	37
III.7	Diagramme de classes des modèles de domaine	40
III.8	Diagramme de Classes : Package API Controllers Partie 1	43
III.9	Diagramme de Classes : Package API Controllers Partie 2	44
III.10	Diagramme de séquence de création d'un projet	46
III.11	Diagramme de classes Design Pattern Repository	48
III.12	Structure du design pattern Factory Method	49
IV.1	Architecture finale de la solution	51
IV.2	Page d'accueil de l'application front-office	53
IV.3	Page d'édition du profil de l'application front-office	54
IV.4	Page de création de projets de l'application front-office	55
IV.5	Vue d'un projet publié sur le réseau Thimble Fashion	55
IV.6	Page de statistiques d'un membre du réseau Thimble Fashion	56
IV.7	Interface d'appel rejeté par l'API	60
IV.8	Demande d'autorisation API REST	61
IV.9	Appel réussi du service de récupération des projets	61

IV.10 Assertion de type Script . . . . .	64
IV.11 Test Fonctionnel List Events Valide . . . . .	65
IV.12 Appel réussie du service de recherche de projets par identifiant . . . . .	66
IV.13 Réponse erronée du service de recherche de projets par identifiant . . . . .	66
IV.14 Interface de configuration du test de charge . . . . .	67
IV.15 Courbe de montée en charge du service de recherche de projet . . . . .	67
IV.16 Résultat du Test Load avec la stratégie Rafale . . . . .	68
A.1 Diagramme de Classes : Package Web Controllers Partie 1 . . . . .	72
A.2 Diagramme de Classes : Package Web Controllers Partie 2 . . . . .	73
A.3 Diagramme de Classes : Package Repositories . . . . .	74
A.4 Diagramme de Classes : Package Requests . . . . .	75
A.5 Diagramme de Classes : Package Middlewares . . . . .	75
A.6 Diagramme de Classes : Package Transformers . . . . .	76

---

# Liste des Tableaux

I.1	Les acteurs Scrum dans notre projet . . . . .	15
I.2	Les sprints du projet et leurs descriptions . . . . .	16
II.1	Description textuelle du UseCase«Créer un projet» . . . . .	26
III.1	Schéma de Routage REST . . . . .	45
IV.1	Contrat du web service d'inscription utilisateur . . . . .	57
IV.2	Contrat du web service de découverte les projets . . . . .	58
IV.3	Contrat du web service de récupération des Images . . . . .	59

---

# Introduction Générale

La Mode vestimentaire désigne la manière de se vêtir selon les goûts émergeant dans un cadre spatio-temporel donné. C'est un phénomène influencé par la culture d'un peuple et les codes qu'elle impose, mais aussi par le goût individuel. Historiquement, l'apparence vestimentaire était un moyen de se différencier par rapport aux autres et d'exprimer ses orientations et ses goûts. Elle est considérée comme l'un des moyens pour annoncer sa classe sociale. La Mode a vu le jour comme étant une pratique élitiste adoptée par la classe bourgeoise. Ce phénomène a connu par la suite une démocratisation qui n'a commencé réellement qu'avec la naissance du prêt-à-porter à la fin des années 50, puis de la Mode des créateurs dans les années 70 [1]. Cette démocratisation reste encore insuffisante de nos jours car la Mode pointue et celle des designers renommés est généralement exposée et accessible au niveau des pôles suivants : Paris, Milan, New York et London. Ce qui semble un peu ironique sachant que la Mode et ses tendances ont été toujours de nature virale et tirent leurs succès de l'atteinte du plus grand nombre du public.

La Mode est aussi l'une des grandes industries qui font tourner l'économie mondiale. Contrairement aux autres secteurs tels que le transport, la médecine ou l'éducation, cette industrie n'a pas vu de grands changements durant les trois dernières décennies et elle est toujours considérée en retard technologique. Les Technologies d'Information et de Communication (TIC) s'avèrent essentielles de nos jours pour améliorer et booster un secteur économique. A titre d'exemple, Uber qui est une entreprise ayant révolutionné le transport en ville. Elle offre un service de location de véhicules de tourisme avec chauffeur (VTC), pensé comme une alternative aux taxis classiques. Ce service passe par une application mobile, qui par un système basé sur la géolocalisation déniche en quelques minutes un véhicule (propriété d'une société de transport en contrat avec Uber). Le prix varie en fonction de la disponibilité et de la distance, selon la loi de l'offre et de la demande. Aujourd'hui, Uber offre ce service dans 22 pays et 62 villes dans le monde[2].

Une des façons pour améliorer un secteur économique c'est de rendre la communication et les échanges professionnels plus simples et efficaces dans ce secteur. Parmi les solutions technologiques qui ont fait leurs preuves dans la réalisation d'un tel objectif, on trouve les réseaux sociaux professionnels et plus exactement ceux qui se spécialisent dans une industrie bien définie. Ce type de réseaux permet de regrouper les différents acteurs d'un secteur économique et leur fournir la possibilité d'exposer leurs compé-

tences, d'interagir plus facilement entre eux et d'être plus proche de l'actualité et des nouvelles pratiques du métier. L'intégration des TICs aujourd'hui dans le domaine de la Mode se résume principalement par l'incorporation du commerce électronique à travers le web et les applications mobiles avec un aspect social et interactif insuffisant. Aucune des solutions existantes ne propose un véritable réseau social professionnel. Dans ce cadre, le projet « Thimble Fashion » intervient pour combler ce besoin à travers une plateforme professionnelle et sociale regroupant le contenu et l'actualités de la Mode de façon à démocratiser encore plus ce domaine et accélérer la diffusion des nouvelles tendances.

Le présent rapport, qui résume nos travaux pour la réalisation du réseau Thimble Fashion, est structuré en quatre chapitres principaux. Le premier chapitre commencera par la présentation de l'organisme d'accueil Logicale et enchaînera avec une étude préalable qui présentera l'industrie de la Mode et les réseaux sociaux professionnels. Nous donnerons aussi des exemples de solutions web de Mode existantes. Ceci va nous permettre d'introduire le cadre du projet et de présenter la méthodologie de travail utilisée. L'analyse des besoins fonctionnels et non-fonctionnels sera l'objet du deuxième chapitre. Dans le troisième chapitre nous allons détailler l'architecture et les choix conceptuels et techniques de notre solution. Dans le quatrième chapitre nous mettrons l'accent sur la phase de réalisation en décrivant l'environnement du travail et la mise en œuvre des différentes parties de la solution. Nous allons clôturer ce chapitre avec description tests effectués et leurs résultats.

---

# Chapitre I

---

## Étude préalable et Cadre du projet

### Introduction

Ce chapitre va présenter en premier l'entreprise d'accueil Logicale et ses activités. Nous allons exposer l'état de l'art de notre projet en introduisant l'industrie de la Mode et ses différents acteurs. De plus, nous allons définir les réseaux sociaux professionnels et leurs caractéristiques. Nous allons enchainer avec l'étude de quelques solutions de Mode existantes tout en indiquant l'apport de notre solution. La dernière partie sera focalisée sur la délimitation du cadre du projet tout en indiquant ses objectifs, sa planification et la méthodologie adoptée pour son élaboration.

### 1 Présentation de l'entreprise d'accueil

Logicale est une boîte de développement logiciel Canadienne implantée à Tunis dont la société mère réside au Québec. Logicale est spécialisée dans le développement des solutions web et e-commerce à base des technologies libres et de pointe. Cette boîte a aussi de l'expertise dans le développement des applications mobiles Android et iOS conçue dans le but d'offrir aux utilisateurs une expérience intuitive et personnalisée. Ces applications vont permettre aux entreprises d'augmenter considérablement leurs productivités, d'accéder instantanément à leurs données et de rejoindre rapidement un grand nombre d'utilisateurs.

En outre, Logicale offre le programme START-UP aux porteurs d'idées qui leurs aide à concrétiser leurs projets en collaboration avec l'équipe d'experts de Logicale et en passant par les différentes phases qui mènent au lancement du projet jugé viable [3].



**Figure I.1** – Logo de Logicale Développement Inc. [3]

Logicale est une entreprise qui mise sur les jeunes talents que se soit en développement, en design ou en management. Elle leurs offre des conditions favorables pour se

donner à fond, mettre en œuvre leurs créativités et collaborer harmonieusement dans la réalisation d'excellentes applications. En effet, grâce au partenariat fait entre Logicale et Orange Developer Centre - Tunisie, nous avons été sélectionnés parmi les finalistes d'Orange Summer Challenge 2014 pour faire partie du projet Thimble Fashion. Logicale réserve à ce projet une grande importance grâce à son potentiel et son unicité dans une industrie de Mode qui manque d'initiative qui mise sur les nouvelles technologies afin d'améliorer la collaboration, l'interactivité et la rentabilité de ce secteur.

## 2 Industrie de la Mode

L'industrie de la Mode est le secteur économique dédié à la fabrication et la vente des vêtements, des chaussures et des produits de luxe. On distingue parfois entre la haute couture (« High Fashion ») et l'industrie des vêtements et textiles caractérisée par la production de masse (« Mass Fashion »). La frontière entre ces deux catégories a commencé à disparaître depuis le début des années 1970. La Mode est de nos jours définie comme les styles d'habillements et d'accessoires portés par un groupe de personnes à une période de temps donnée. Cependant, Il existe toujours des différences entre les produits des stylistes exposés dans les défilés de Mode à Paris ou à New York et les vêtements produits en masse et vendus dans les boutiques et les centres commerciaux autour du monde. Cette industrie peut être divisée en quatre niveaux séparés mais interdépendants. Le premier niveau c'est celui de la production de la matière première qui est principalement les fibres, les textiles mais aussi les cuirs et les fourrures. Le niveau suivant est la production des biens faite par les créateurs de Modes et les fabricants. Dans le troisième niveau on trouve la vente en gros ou au détail. On trouve en dernier niveau tout ce qui est en relations avec la publicité et la promotion des produits de Mode. Cette industrie vaut en chiffre plus de 1 778 milliards de dollars de revenues en 2011. Une accélération de cette performance est prévue durant les 5 années entre 2011 et 2016 avec une augmentation de 4.2% selon ReportLinker 2012 [1].

Ce secteur emploie des dizaines de millions de personnes dans le monde directement ou indirectement. Il existe une grande variété de professions intervenant dans ce domaine parmi lesquelles on peut citer [4] :

- **Créateur ou Designer Textile**

Le designer textile est celui responsable de la conception des maquettes et des plans de collections d'une part et la création des échantillons et des prototypes d'autre part. Ce professionnel surveille et identifie également les nouvelles ten-

dances, les modes de consommation, les nouveautés en termes de gammes et de collections de la concurrence.

- **Styliste**

Le styliste conseille le créateur mais aussi les personnes qui l'emploient. De nombreuses personnalités font appel à ses services et savoir-faire pour des sorties officielles par exemple. Il est amené à donner son avis sur les nouvelles collections au niveau du choix des matières et des coloris. Il doit être à l'affût des nouvelles tendances de la Mode mais aussi celles proposées par les entreprises concurrentes. En plus de jouer un rôle de conseiller, ce spécialiste assure le stylisme des défilés tant dans l'aspect organisationnel qu'artistique. De nombreux stylistes travaillent aussi pour des magazines.

- **Modéliste**

Le modéliste donne vie aux créations du styliste. Il matérialise les dessins de ce dernier. À l'aide du croquis, il crée le patron et réalise le prototype. Il doit alors allier technique et créativité. Ayant réalisé le prototype, il peut ensuite procéder aux ajustements nécessaires, qui peuvent être faits par ordinateur, à l'aide des logiciels adéquats.

- **Chef de Produit**

Le chef de produit est responsable d'un type d'articles ou d'une gamme entière depuis sa conception jusqu'à sa commercialisation. Pour lancer un produit, il s'appuie sur différentes études. Il analyse les attentes des clients et les initiatives de la concurrence. Il fait preuve d'anticipation pour devancer les besoins des consommateurs. Sa mission principale : imaginer le futur produit en collaboration avec le service de production et les commerciaux. Dans le secteur de la Mode, il doit prendre en compte les tendances de style et tout ce que font ses concurrents. Il analyse l'historique des ventes et établit le plan de collection en tenant compte des informations obtenues auparavant.

- **Journaliste de Mode**

Le journaliste de Mode a pour mission de faire parler des collections de haute couture et de prêt à porter le plus rapidement possible. Le rôle des journalistes est de déceler les nouvelles tendances, d'identifier les talents prometteurs, de défendre les valeurs sûres. Leurs publications dans la presse papier et audio-visuelle ont un impact certain sur l'industrie de la Mode, car elles peuvent faire et défaire

une réputation.

Comme toute autre industrie, l'activité du secteur de la Mode naît de l'interaction et la collaboration entre ses différents acteurs dont les principaux ont été définis précédemment. Cette activité est basée sur la communication et l'échange d'idées concernant l'aspect créatif, commercial ou managérial de ce secteur. Il est clair aussi que les tendances et produits de Mode émergent de l'analyse et l'étude de la variation des goûts et de la consommation du public. Parmi les canaux de communication utilisés de nos jours, les TICs s'avèrent les plus rapide et efficace pour atteindre une plus grande communauté et offrir un espace de discussion et de collaboration. Les TICs permettent aussi d'avoir les informations nécessaires pour analyser le comportement d'une communauté envers un produit ou un phénomène quelconque. Il sera possible par la suite de dévoiler les tendances, les produits les plus appréciés ou encore les individus les plus influents. Les réseaux sociaux sont aujourd'hui à la tête de la liste des outils technologiques et communicationnels utilisés dans les différents secteurs économiques.

## 3 Réseaux sociaux professionnels

Par définition, un réseau social est une plateforme en ligne qui permet aux utilisateurs de créer un profil public, partager du contenu et interagir avec les autres membres. Les utilisateurs peuvent aussi découvrir les cercles des relations de leurs amis et par la suite élargir leurs réseaux de contacts. Depuis leur apparition au début des années 2000, les réseaux sociaux ont attiré des millions d'utilisateurs qui les ont intégrés dans leur routine quotidienne [5].

Les réseaux sociaux possèdent un ensemble de caractéristiques dont les deux principales sont les suivantes [6] :

- **Basés sur l'utilisateur**

Avant l'apparition des réseaux sociaux, les sites web sont à base du contenu qui est géré par le webmaster ou l'équipe éditoriale et consulté par les internautes. Le flux d'information était dans un seul sens. Cependant, les réseaux sociaux tournent autour des utilisateurs. Le contenu est dynamique et il est créé par n'importe quel utilisateur sur le réseau.

- **Interactifs**

C'est une caractéristique des réseaux sociaux modernes. Ces derniers ne sont plus

juste une simple collection de chatrooms ou de forums. Des sites tels que Facebook proposent par exemple plusieurs jeux multi-joueurs où on peut défier ses amis.

Pour avoir une vocation professionnelle, un réseau social doit proposer essentiellement des services et des fonctionnalités professionnels. Il doit se focaliser sur la mise en valeur du parcours académique et professionnel des membres d'une part et les relations entre eux d'une autre part. Ce type de réseaux se divise en deux grandes catégories [7] :

- **Les réseaux sociaux professionnels internes (RSPI)**

Les réseaux sociaux professionnels internes sont des réseaux appartenant à une organisation (entreprise, administration, organisation non-gouvernementale, etc.) et dont les membres sont les individus appartenant à cette organisation. Chaque nouveau membre est amené à enrichir son réseau de contacts qui va l'aider à progresser dans l'organisation.

- **Les réseaux sociaux professionnels externes (RSPE)**

Contrairement aux RSPI, Les réseaux sociaux professionnels externes sont des réseaux constitués en dehors de l'organisation professionnelle de rattachement. Pour un chef d'entreprise, les RSPE facilitent le recrutement de nouveaux collaborateurs et la recherche de nouveaux clients et partenaires. Pour les employés, ces réseaux permettent de décrocher l'emploi le plus adapté aux compétences et aux attentes de chacun. Parmi les RSPE existants, on peut citer :

- **LinkedIn** : c'est le plus grand RSPE généraliste avec plus de 300 millions d'utilisateurs provenant de près de 120 secteurs d'activité dans plus de 200 pays autour du monde. Il a été lancé officiellement en 2003 et il a comme objectif de connecter les professionnels et améliorer leur productivité. Ce réseau offre la possibilité aux membres d'agrandir leurs relations professionnelles et de consulter les offres de travaux et l'actualité des industries [8].
- **Behance** : c'est une plateforme de portfolio en ligne appartenant à Adobe. Elle est destinée aux professionnels du web, des arts graphiques et aux photographes. Cette plateforme est sous forme d'un véritable réseau social puisqu'elle donne la possibilité de suivre l'actualité des autres utilisateurs, d'apprécier et de commenter leurs projets. Behance propose aussi

la possibilité de créer son CV complet en ligne pour avoir plus de visibilité par les recruteurs et mettre en avant ses créations [9].

## 4 Solutions web de Mode existantes

Actuellement, il n'existe pas un véritable réseau social professionnel spécialisé dans le domaine de la Mode. Dans ce qui suit, nous allons exposer une sélection de solutions web actives dans ce secteur tout en indiquant les fonctionnalités qu'elles proposent.

### 4.1 FashionUnited

FashionUnited est une plateforme web internationale consacrée exclusivement à la Mode. Active dans le monde entier, cette plateforme attire plus de 1.6 millions de professionnels de l'industrie par mois. Les visiteurs peuvent parcourir leur site web local traduit dans leur langue maternelle afin de consulter : les dernières actualités de la Mode, des statistiques commerciales, un annuaire, des campagnes publicitaires, un espace consacré à l'emploi, un calendrier complet des événements de la Mode, et le réseau communautaire FashionUnited sur lequel les professionnels de la Mode peuvent se connecter et discuter [10]. La figure I.2 présente la page d'accueil de la plateforme FashionUnited.

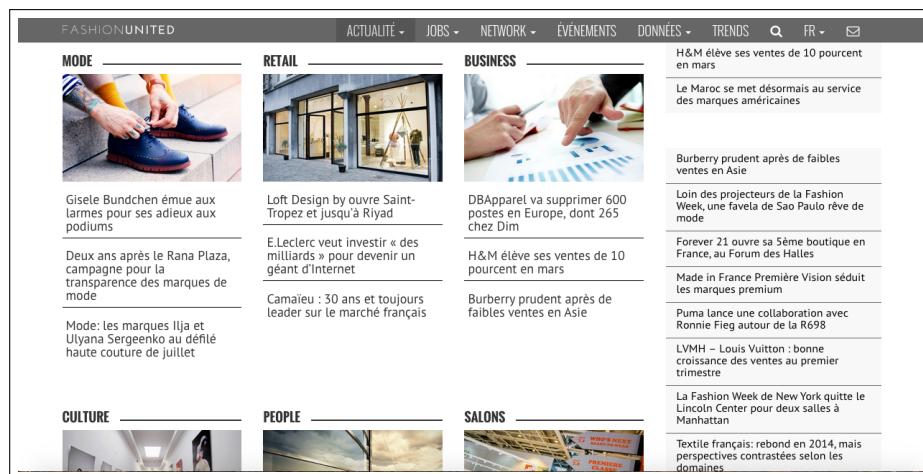
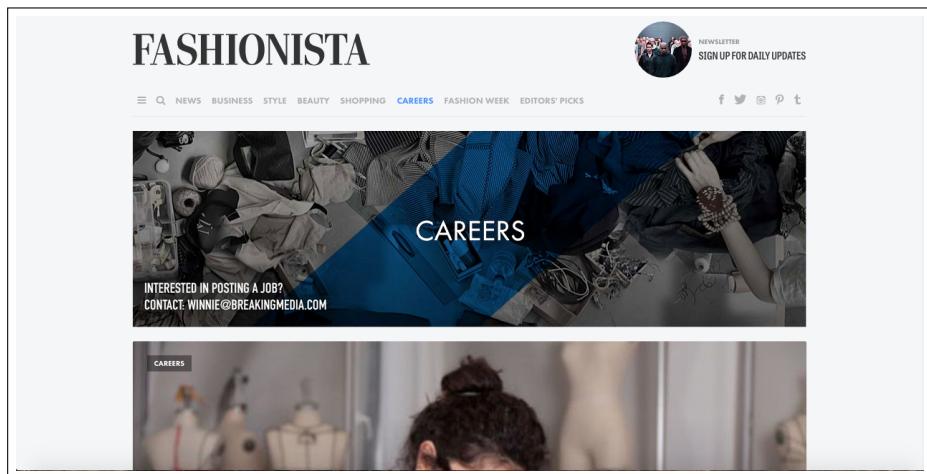


Figure I.2 – Page d'accueil de FashionUnited [10]

### 4.2 Fashionista

C'est l'un des plus grands sites indépendant dédié à l'actualité de la Mode. Il présente une collection raffinée des nouvelles, des critiques et conseils professionnelles en

relation avec la Mode. Fashionista produit quotidiennement des business stories autour des plus grandes marques et des interviews avec les industriels pour exposer les clés du succès dans cette industrie [11]. La figure I.3 présente la page d'accueil du site web Fashionista.



**Figure I.3 – Page d'accueil de Fashionista [11]**

### 4.3 ManyMucho

Le site web « manymucho.com » est un espace d'échange d'informations réservée aux professionnels du textile et du prêt à porter. Les fabricants grossistes présentent leurs collections. Les clients choisissent les produits et contactent les exposants. Les services proposés par ce site web se divisent en deux catégories [12] :

- Services pour les détaillants :
  - Les appels d'offres
  - Les demandes de devis
  - Le contact direct à travers les messages
- Services pour les grossistes :
  - Les showrooms pour exposer ses produits aux détaillants
  - La gestion de son réseau de détaillants
  - Redirection des détaillants vers son site web

La figure I.4 présente la page d'accueil du site web ManyMucho.

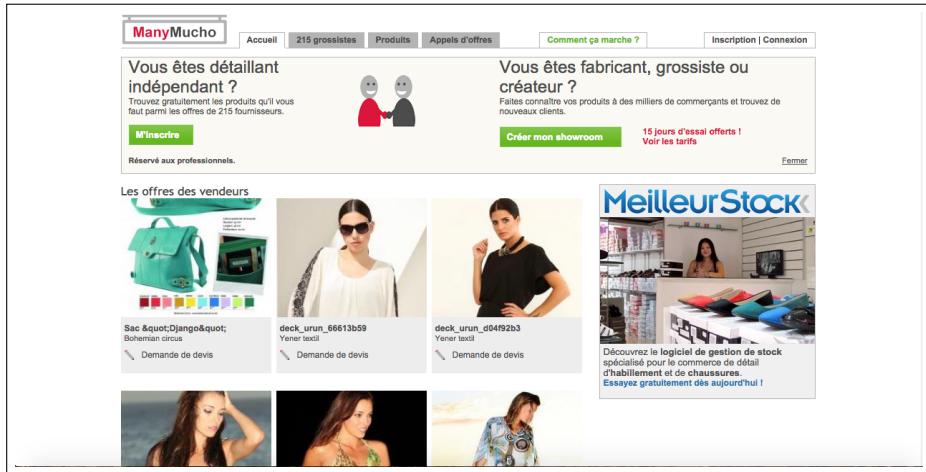


Figure I.4 – Page d'accueil de ManyMucho [12]

### 4.4 Discussion

Les trois solutions présentées précédemment se concentrent soit sur l'aspect informatif et l'actualité de la Mode soit sur des services commerciaux et d'exhibition tels que les showrooms. Ils ne prennent pas en compte les besoins créatifs du modéliste ou du créateur textile par exemple. En outre, aucune de ces dernières ne présente un vrai réseau social permettant de regrouper les différents acteurs du secteur de la Mode dans un espace adéquat facilitant l'interaction et la communication entre eux. C'est là où intervient le projet Thimble Fashion afin d'adapter le concept des réseaux sociaux professionnels externes à l'industrie de la Mode tout gardant un équilibre entre les services professionnels proposés et l'aspect créatif et social de la plateforme. Nous allons présenter ce projet avec plus de détails dans le point suivant en délimitant le périmètre de notre projet de fin d'études.

## 5 Projet Thimble Fashion

Thimble Fashion est considéré comme l'un des premiers projets internes à Logicale. Ce projet vise à mettre en place une plateforme sociale dédiée aux professionnels et aux amateurs de la Mode. Cette plateforme sera un espace pour partager et publier des créations et du contenu en relations avec le modélisme, le stylisme et les nouveautés du monde du textile. De cette manière, les gens peuvent découvrir aisément les nouvelles tendances et interagir avec des designers prometteurs. Les designers de leurs coté

peuvent suivre de plus proche la réaction du public envers leurs nouveaux modèles. Ce réseau permet aussi aux entreprises de la Mode de poster leurs offres de travaux et dénicher les nouveaux talents. Finalement, Thimble Fashion a pour but aussi de redéfinir en partie la chaîne de création de valeur dans cette industrie. En effet, cette plateforme va incorporer une boutique en ligne qui produit et vend les produits conçus par les designers et que les utilisateurs jugent intéressants et veulent acheter. Donc la production sera faite selon la demande et les tendances. Ce concept va éliminer le besoin de recourir à la promotion par exemple pour écouter les surplus d'inventaire.

Cette plateforme va être disponible à travers une application web et des applications mobiles natives (iOS/Android). Elle comportera aussi un back-office qui va servir pour :

- L'administration de la plateforme.
- Le suivi des statistiques concernant les projets publiés et le trafic interne.
- Gestion des offres de travaux.
- Lancement des opérations Fashion Kickstater.

Le projet est divisé en trois grandes phases pour diminuer le risque d'exécution :

### – Phase 1

Cette phase est dédiée à l'élaboration de la base du projet qui est le réseau social professionnel. A la fin de cette dernière, les différents utilisateurs de la plateforme seront capables de créer du contenu, le partager et interagir entre eux. Notre projet de fin d'étude intervient pour concrétiser cette première phase. Son objectif est de concevoir et développer la plateforme web du réseau et l'API de web services qui va être consommée par les applications mobiles natives et le back-office englobant les fonctionnalités mentionnées précédemment.

### – Phase 2

C'est à cette phase que l'opération Fashion Kickstater va être déclenchée. Cette opération est gérée par Logicale et consiste en la sélection des meilleurs designers selon les statistiques et les analytiques du back-office. Suite au choix du designer ayant les modèles les plus appréciés du public, un de ses modèles va être sélectionné pour la production. Après avoir une entente avec un fournisseur de livraison comme Fedex et un accord avec un fabriquant, une offre de vente d'une quantité limitée d'exemplaires sera mise sur la plateforme pour tester le concept. Il faut noter que la production ne sera lancée que si un nombre prédéfini

d'acheteurs a été atteint sinon l'argent déboursé sera renvoyé à l'acheteur. De plus, 5% des ventes de toute les campagnes Fashion Kickstater sera donnée à une œuvre de bienfaisance que les membres du réseau auront choisi lors de l'achat du vêtement. Cette fonctionnalité va permettre à Logicale d'avoir une idée claire sur les attentes et les goûts des potentiels acheteurs. Ce concept permet de résoudre un des problèmes actuels de l'industrie qui fait que les compagnies produisent des vêtements et ensuite elles investissent dans le marketing et la commercialisation. Ces dernières sont susceptibles d'encourir encore plus de pertes pour écouler probablement le surplus d'inventaire. Le but de ce modèle économique est de produire ce que les gens veulent uniquement.

### – Phase 3

A ce niveau, Logicale va incruster un nouveau module technologique à la plate-forme qui est la prise automatique de mensuration. En effet, les visiteurs auront la possibilité d'avoir précisément les différentes tailles de leurs corps à travers le webcam d'un ordinateur ou la caméra d'un smartphone. Ce module va leur suggérer par la suite leurs tailles de vêtements selon les standards internationaux. Cette fonctionnalité va être exposée ensuite à travers une API aux autres boutiques en lignes.

## 6 Méthodologies Agiles

Actuellement, il existe plusieurs méthodologies employées dans le développement logiciel. On les appelle les méthodes agiles. Cette partie a pour but de présenter ces méthodes.

Les méthodes de développement dites « méthodes agiles » visent à réduire le cycle de vie du logiciel et accélérer son élaboration en développant une version minimale, puis en intégrant les fonctionnalités par un processus itératif. L'origine des méthodes agiles est liée à l'instabilité de l'environnement technologique et au fait que le client est souvent dans l'incapacité de définir ses besoins de manière exhaustive dès le début du projet. Le terme « agile » fait ainsi référence à la capacité d'adaptation aux changements de contexte et aux modifications de spécifications intervenant pendant le processus de développement. En 2001, le manifeste agile a vu le jour et il se focalise sur les points suivants [13] :

- Individus et interactions plutôt que processus et outils.

- Développement logiciel plutôt que documentation exhaustive.
- Collaboration avec le client plutôt que suivi d'un plan rigide.

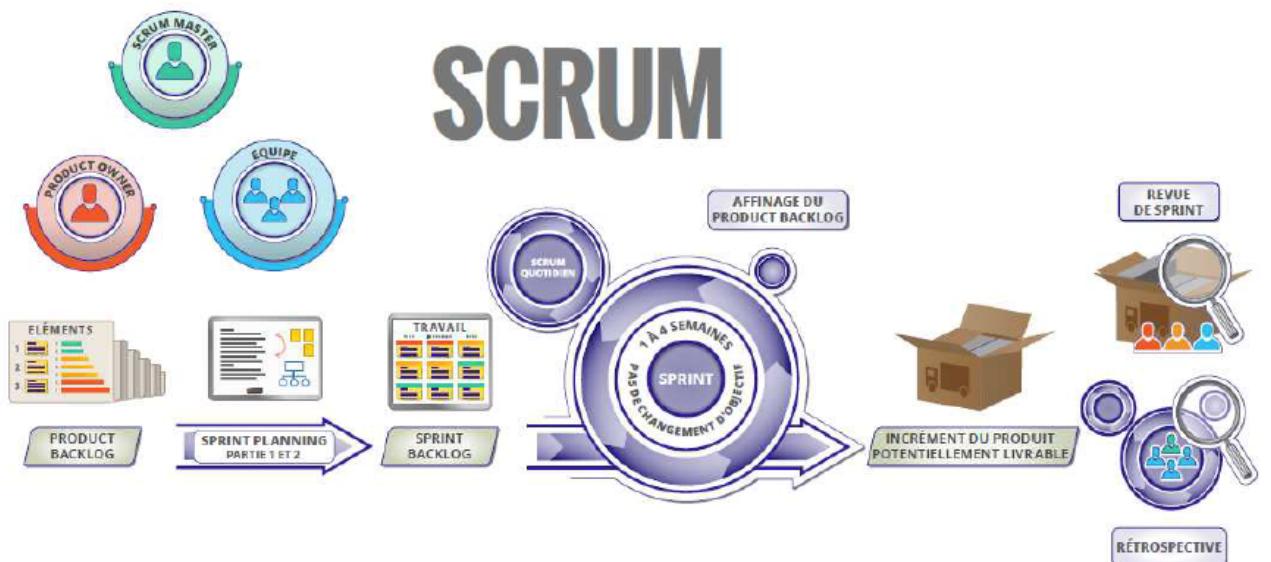
Grâce aux méthodes agiles, le client est pilote à part entière de son projet et obtient très vite une première mise en production de son logiciel. Nous allons détailler dans le point suivant la méthode SCRUM étant l'une des méthodologies les plus utilisées.

### 6.1 Méthodologie SCRUM

Scrum est un cadre de développement dans lequel des équipes plurifonctionnelles réalisent des produits de manière itérative et incrémentale. Cette méthode décompose le développement en cycles appelés Sprints. Les Sprints sont d'une durée limitée qui est de quartes semaines au maximum. Au début de chaque Sprint, l'équipe plurifonctionnelle sélectionne des éléments (exigences du client) dans une liste priorisée. L'équipe se met d'accord sur un objectif constitué de ce qu'elle pense pouvoir accomplir durant le Sprint. Chaque jour, l'équipe se réunit brièvement afin de contrôler sa progression et ajuster les prochaines étapes. A la fin de chaque Sprint, une revue est organisée avec les parties prenantes durant laquelle l'équipe montre ce qu'elle a réalisé. Le feedback obtenu peut être pris en compte sur le Sprint suivant. Scrum insiste sur la nécessité de livrer un produit opérationnel à la fin de chaque Sprint. La méthodologie Scrum fait intervenir trois rôles principaux qui sont [14] :

- **Product Owner** : Dans la majorité des projets, le responsable produit (Product owner) est le responsable de l'équipe projet client. C'est lui qui va définir et prioriser la liste des fonctionnalités du produit et choisir la date et le contenu de chaque sprint sur la base des valeurs (charges) qui lui sont communiquées par l'équipe.
- **Scrum Master** : Il veille à ce que chacun puisse travailler au maximum de ses capacités en éliminant les obstacles et en protégeant l'équipe des perturbations extérieures. Il porte également une grande importance au respect de l'approche Scrum.
- **Equipe** : Elle regroupe tous les rôles habituellement nécessaires à un projet, à savoir l'architecte, le concepteur, le développeur, le testeur, etc. L'équipe s'organise elle-même et sa composition reste inchangée pendant toute la durée d'un sprint.

Les rôles, les artefacts et les événements clés de la méthodologie Scrum sont présentés dans la figure I.5.



**Figure I.5** – Vue d'ensemble de Scrum [14]

Notre choix de la méthodologie Scrum se justifie par la variété de ses avantages et parmi lesquels on peut citer :

- L'adoption de cette méthodologie par Logicale dans ses divers projets.
- Cette méthodologie est plus adaptée dans un contexte où les spécifications sont fréquemment rectifiées.
- L'augmentation de la productivité.

## 6.2 Scrum appliqué à notre projet

Durant notre projet, nous avons affiné la méthodologie Scrum au contexte de notre travail. L'équipe réduite du projet est présentée dans le tableau I.1 :

Nom de la personne	Poste de la personne dans l'entreprise	Rôle de la personne
David Dupont	Directeur de Logicale	Product Owner
Selmi Chouheib	Architecte Système	Scrum Master
Pepin Yoann	Infographiste	Scrum team
Ghabi Amin	Stagiaire Développeur Mobile	
Ben Ammar Fares	Stagiaire Développeur Mobile	
Mahjoubi Mehdi	Stagiaire Traitement D'image	
Satouri Nidhal	Stagiaire Développeur Web	

Tableau I.1 – Les acteurs Scrum dans notre projet

En tenant compte des priorités des exigences communiquées par le Product Owner, le projet a été décomposé en quatre sprints principaux décrits dans le tableau I.2 :

Sprint	Date Début	Date Fin	Description
<b>Sprint #1</b> (Analyse, Conception et Choix Technologiques)	01/02/2015	16/02/2015	<ul style="list-style-type: none"> <li>– Initiation au projet</li> <li>– Analyse des besoins</li> <li>– Architecture et conception</li> <li>– Choix et initiation aux technologies d'implémentation</li> </ul>
<b>Sprint #2</b> (Mise en place de l'API)	17/02/2015	15/03/2015	<ul style="list-style-type: none"> <li>– Choix du schéma et ressources de l'API de web services</li> <li>– Gestion des autorisations</li> <li>– Développement des méthodes de l'API de web services</li> <li>– Tests et documentation</li> </ul>

Sprint	Date Début	Date Fin	Description
<b>Sprint #3</b> (Mise en place du Front-office)	16/03/2015	14/04/2015	<ul style="list-style-type: none"> <li>– Authentification Thimble/Facebook</li> <li>– Gestion d'un compte utilisateur (Amateur/Professionnel)</li> <li>– Gestions des projets, collections et évènements</li> <li>– Gestions des équipes</li> <li>– Tests et documentation</li> </ul>
<b>Sprint #4</b> (Mise en place du Front-office)	15/04/2015	20/05/2015	<ul style="list-style-type: none"> <li>– Gestion de l'aspect social (Follow, Like, Comment, Share)</li> <li>– Gestion du filtrage et recherche du contenu</li> <li>– Gestions des offres de travail</li> <li>– Tests et documentation</li> </ul>

**Tableau I.2** – Les sprints du projet et leurs descriptions

## Conclusion

Nous avons entamé ce chapitre avec la présentation l'entreprise d'accueil Logicale. Nous avons mis l'accent ensuite sur l'industrie de Mode et ses acteurs ainsi que les réseaux sociaux professionnels qui constituent l'état de l'art de notre projet. Nous avons présenté par la suite trois solutions web destinées à la mode en finissant par l'explication de l'insuffisance des services qu'elles proposent. Nous avons passé à la définition du cadre général de notre projet dans la partie suivante en décrivant le projet Thimble Fashion dans sa globalité et en délimitant le périmètre de notre projet de fin d'études. Dans la dernière partie, nous avons défini l'approche Scrum en justifiant son adoption et en décrivant son application dans notre projet.

Le chapitre suivant va se focaliser sur l'analyse et la modélisation des besoins fonctionnels et non-fonctionnels et des utilisateurs de notre solution.

---

# Chapitre II

---

## Analyse et Modélisation des besoins

### Introduction

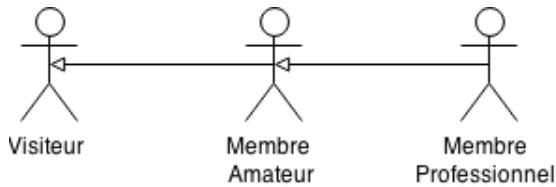
La phase de spécification est une phase vitale dans le cycle de vie de tout logiciel. Elle permet de dégager les exigences du projet et d'en définir les attentes et les spécifications. Dans ce qui suit nous allons procéder à la définition des acteurs utilisant notre solution ainsi que les différents besoins fonctionnels et non-fonctionnels. Nous allons modéliser ces besoins dans la dernière partie de ce chapitre en utilisant le langage UML.

### 1 Utilisateurs cibles de la solution

Nous avons décidé de classer les éventuels utilisateurs de notre solution sous forme des trois types d'acteurs suivants :

- **Visiteur** : C'est n'importe quel internaute visitant notre réseau Thimble Fashion dans le but de découvrir le contenu créé par les autres membres ainsi que les différents événements publiés. Il peut être aussi un recruteur de l'industrie de Mode venant dénicher les talents actifs sur notre réseau.
- **Membre Amateur** : Un compte amateur est destiné à ceux qui ne veulent pas se limiter à la découverte du contenu. Ce compte donne la possibilité à l'utilisateur d'exprimer son avis envers le contenu créé par les autres membres et de communiquer avec eux.
- **Membre Professionnel** : C'est un membre ayant migré vers un compte professionnel. En conséquent, il peut profiter d'autant plus de fonctionnalités telle que la création d'un portfolio de projets, la création d'évènements, la création d'une équipe de travail ainsi que la découverte des offres de travail. Ce type de compte est adéquat aux besoins des différents professionnels de la Mode qui peuvent être des designers textile, des stylistes ou des journalistes de Mode par exemple.

Nous avons résumé l'ensemble des acteurs cités précédemment ainsi que les relations de d'héritage entre eux dans la figure II.1.



**Figure II.1** – Les utilisateurs cibles de la solution

## 2 Analyse des besoins

L'analyse de la solution cible et de ses objectifs a permis de dégager les fonctionnalités à mettre à la disposition des acteurs cités précédemment. Les besoins dégagés sont classés en besoins fonctionnels et besoins non-fonctionnels.

### 2.1 Besoins fonctionnels

Selon les exigences fonctionnelles décrites par le Product Owner du projet, nous avons pu extraire les besoins fonctionnels de notre solution. Nous avons classé ces besoins en deux catégories : les besoins liés au Front-office<sup>1</sup> et les besoins liés à l'API de web services.

#### 2.1.1 Besoins fonctionnels liés au Front-office

La liste suivante décrit brièvement l'ensemble des fonctionnalités à prévoir au niveau du l'application web vitrine :

- S'authentifier.
- Afficher les membres, les projets, les évènements, les collections de projets, les équipes de créateurs et les offres de travail.
- Permettre d'effectuer une recherche et le raffinement de l'affichage avec des paramètres de filtrage.
- Créer un compte amateur et configurer le profil utilisateur.

---

1. Front-office : C'est l'application web de base du réseau social professionnel. Contrairement à l'application web back-office dédié uniquement à l'administration, le front-office présente la vitrine web accessible par le public.

- Permettre l’interaction sociale (Aimer (Like), Commenter (Comment), Partager sur Facebook (Share), Suivre un membre (Follow), Participer à un événement (Participate)).
- Envoyer et recevoir des messages privés.
- Afficher à l’utilisateur l’activité des membres qu’il suit et les notifications qui lui concernent.
- Créer et gérer une équipe de travail, des projets, des évènements, des collections personnelles de projets.
- Migrer gratuitement vers un compte professionnel et compléter son CV.
- Afficher à l’utilisateur les graphs et les statistiques concernant ses projets et la visibilité de son profil.

### 2.1.2 Besoins fonctionnels liés à l’API de web services

Le réseau Thimble Fashion sera disponible aussi sous forme d’applications mobiles natives. Il faut noter aussi que le back-office va être sous forme d’une application web séparée de celle dédiée au réseau Thimble. Par conséquent, nous avons jugé nécessaire l’élaboration préalable d’une API (Application Programming Interface) de web services. Cette API va garantir l’interopérabilité et l’échange de données entre les applications mobiles et le back-office d’une part et le serveur de l’application front-office d’une autre part. Elle doit fournir les fonctionnalités suivantes :

- Obtenir une autorisation pour consommer les web services.
- Invoquer les web services appropriés.

## 2.2 Besoins non-fonctionnels

A travers les besoins non-fonctionnels, nous décrivons les exigences en terme de performance, de sécurité, d’extensibilité et de maintenance. Nous avons veillé durant le cycle de conception et de réalisation à bien gérer ces préoccupations qui sont transversales à la logique métier de la solution. Nous avons classifié ces besoins non-fonctionnels avec la même approche utilisée pour les besoins fonctionnels.

### 2.2.1 Besoins non-fonctionnels liés au Front-office

- **Interface utilisateur** : Le front-office doit intégrer une interface simple et ergonomique conforme aux maquettes fournies par l’infographiste. Cette interface doit refléter les spécificités et le jargon utilisé dans l’industrie de la Mode.

- **Sécurité** : L’authentification se fait soit avec une adresse email et mot de passe soit avec la connexion Facebook. Le front-office doit bien contrôler les permissions des utilisateurs authentifiés avant d’exposer et d’exécuter les différentes fonctionnalités.
- **Maintenabilité** : Le front-office doit être implémenté suivant une architecture robuste et en tenant compte des bonnes pratiques conceptuelles. L’application doit être modulaire avec un code bien lisible et commenté afin de faciliter sa compréhension, l’identification d’éventuelles bugs et leurs corrections.

### 2.2.2 Besoins non-fonctionnels liés l’API de web services

- **Schéma et Extensibilité** : Le schéma de l’API doit être conçu selon les standards des web services. Ce schéma doit être clair, flexible et simple à évoluer.
- **Sécurité** : L’API ne doit autoriser que les requêtes provenant des applications mobiles ou du back-office. Elle doit bien veiller à contrôler les sessions d’utilisateurs et la validation des données incluses dans les requêtes.
- **Temps de réponse** : Le temps d’exécution des requêtes doit être raisonnable tout en gardant un bon niveau de scalabilité face à un nombre important de requêtes simultanées.

## 3 Modélisation des besoins

Après avoir dégagé l’ensemble des besoins fonctionnels et non-fonctionnels, nous allons procéder à la modélisation des besoins fonctionnels en détaillant ceux qui nécessitent une description textuelle approfondie. Nous avons utilisé le langage de modélisation UML (Unified Modeling Language) et plus précisément les diagrammes de cas d’utilisation (UseCase) pour représenter les fonctionnalités proposées pour chaque type d’utilisateur.

### 3.1 Diagramme de cas d’utilisation d’un visiteur

La figure II.2 présente l’ensemble des cas d’utilisation accessibles par un simple visiteur sans qu’il soit obligé de faire une inscription. Il s’agit principalement des cas

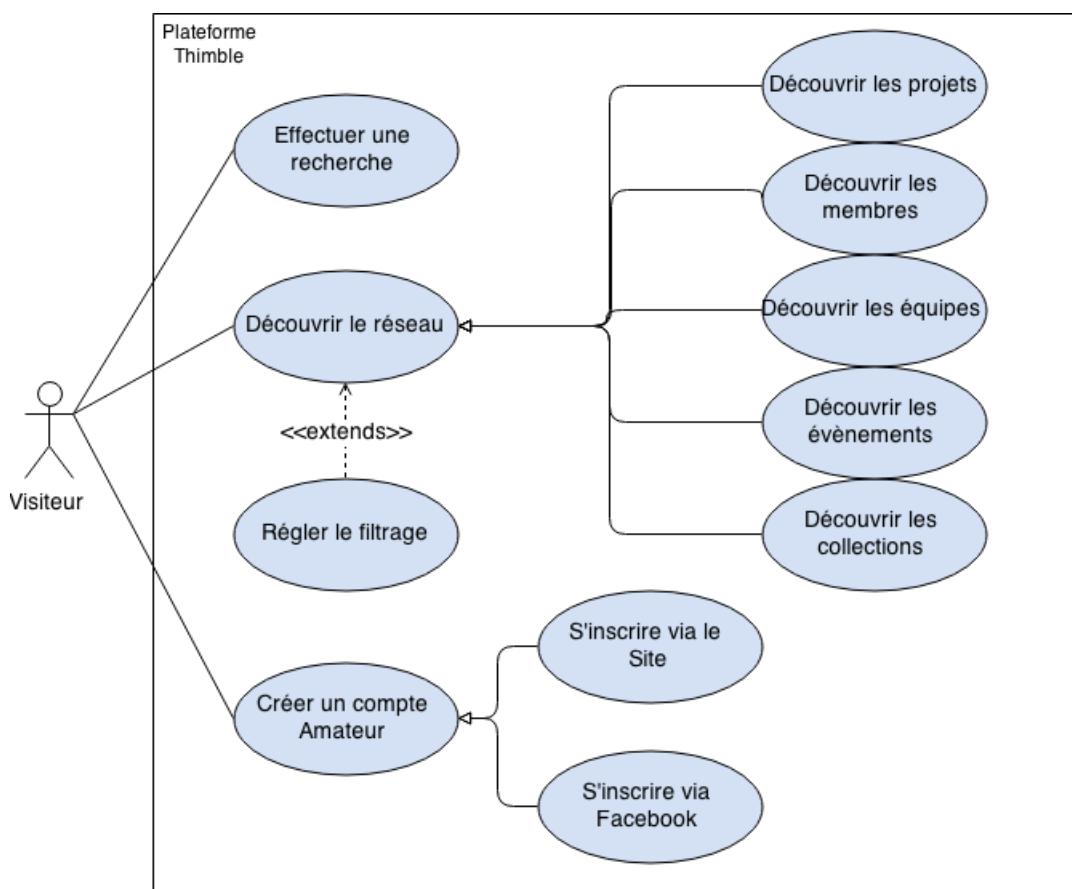
d'utilisation suivants :

- **Cas d'utilisation : Découvrir le réseau**

Chaque visiteur est capable de découvrir le contenu de la plateforme (les membres, les projets, les évènements, les collections de projets et les équipes de créateurs). Il aura la possibilité d'affiner sa découverte à travers la barre de recherche et le menu de filtrage qui lui permettra par exemple d'afficher les projets par catégorie ou par sous-catégorie et de choisir l'ordre d'affichage.

- **Cas d'utilisation : Créeer un compte Amateur**

Pour devenir membre du réseau Thimble Fashion, un visiteur peut créer un compte amateur à travers la page d'inscription ou en utilisant l'inscription Facebook. La vérification de l'email est demandée à l'utilisateur en cas d'inscription standard puisqu'elle va servir ultérieurement pour la réinitialisation du mot de passe ou pour l'envoi des lettres d'information par exemple.



**Figure II.2 – Diagramme UseCase d'un visiteur**

### **3.2 Diagramme de cas d'utilisation d'un membre amateur**

Outre les cas d'utilisation qu'il hérite de l'acteur « visiteur », un membre amateur dispose des cas d'utilisation présentés dans le diagramme UseCase de la figure II.3. Nous avons résumé ces cas d'utilisation dans les points suivants :

- Cas d'utilisation : Gérer son profil**

Une fois inscrit, l'utilisateur dispose d'un compte amateur. Il a la possibilité d'éditer et accomplir son profil amateur avec ses données personnelles, une photo de profil et une photo de couverture. Ce dernier est en mesure de migrer vers un compte professionnel gratuitement à partir de la page de configuration de son compte.

- Cas d'utilisation : Gérer les collections**

Les membres ont la possibilité de regrouper les projets/créations qu'ils apprécient sous forme de collections. Ils peuvent à tout moment retirer des projets de l'une des collections, la modifier ou la supprimer.

- Cas d'utilisation : Interagir socialement**

La plateforme permet aux membres d'évaluer le contenu qu'ils découvrent et donner leurs avis grâce à l'interaction sociale (Aimer, Commenter, Partager sur Facebook).

- Cas d'utilisation : Découvrir le réseau**

Nous avons repris ce cas d'utilisation qui a été décrit au niveau de diagramme UseCase de la figure II.2 pour indiquer les fonctionnalités sociales qui ont été rajoutées à la découverte. Étant connecté, le membre peut maintenant interagir socialement lors de la découverte du contenu du réseau. Il peut également participer aux événements qui l'intéressent. Pour bien organiser les projets qu'il apprécie, le membre peut les ajouter à l'une de ses collections. De plus, ce dernier a la possibilité de suivre les utilisateurs qu'il juge intéressant et communiquer avec eux via les messages privés.

- Cas d'utilisation : Afficher fils d'activité des Following / Afficher ses notifications**

Pour suivre l'actualité des membres qu'il suit à savoir leurs nouveaux projets, équipe ou évènements, l'utilisateur peut consulter le fils l'activité des Following. Le membre a la possibilité de lister des notifications qui lui concerne tels que les likes et les commentaires sur l'un de ses projets.

- Cas d'utilisation : Gérer ses messages**

Nous avons cité précédemment que les membres peuvent communiquer à travers

les messages privés. En effet, les utilisateurs peuvent consulter la liste des messages reçus, envoyer des nouveaux messages, les éditer ou vérifier si un message a été lu par le destinataire ou pas.

#### - Cas d'utilisation : S'authentifier

L'authentification d'un utilisateur se fait soit par une combinaison d'une adresse email et d'un mot de passe ou en utilisant l'authentification Facebook.

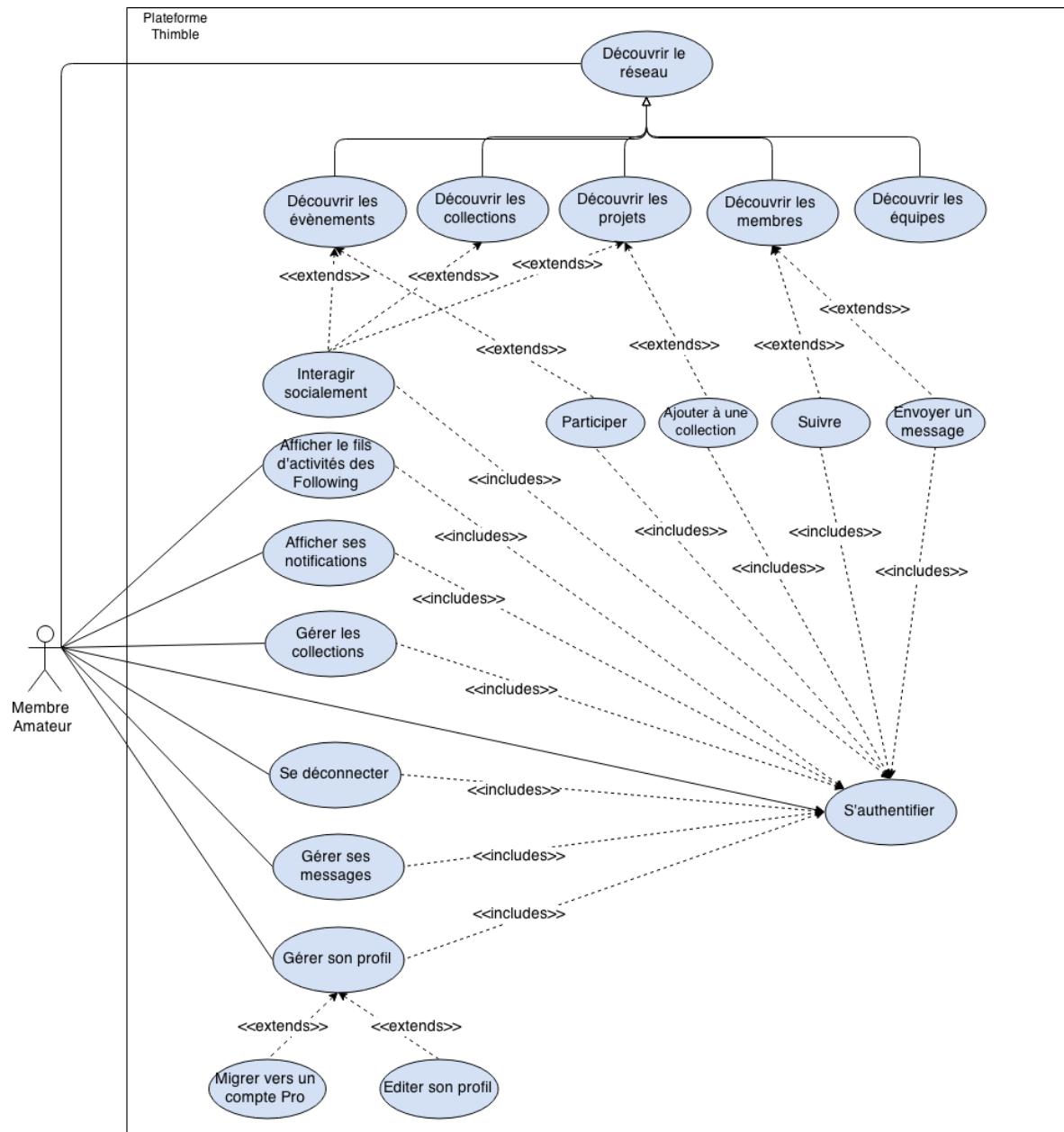


Figure II.3 – Diagramme UseCase d'un membre amateur

### 3.3 Diagramme de cas d'utilisation d'un membre professionnel

Un membre professionnel bénéficie d'encore plus de fonctionnalités lui permettant de créer du contenu, collaborer avec d'autre membre au sein d'une équipe et être plus visible. Ces fonctionnalités se rajoutent à celle qu'il hérite de l'acteur « membre amateur » et elles sont indiquées dans le diagramme UseCase de la figure II.4. Nous avons résumé ces nouvelles fonctionnalités dans les points suivants :

- **Cas d'utilisation : Compléter son CV** Un membre professionnel peut compléter son profil avec les données de son parcours professionnel et académique, ses compétences et les prix qu'il a décroché.
- **Cas d'utilisation : Gérer son équipe / Ajouter à son équipe** Pour améliorer ses projets et profiter d'un travail collaboratif avec d'autres utilisateurs, un membre professionnel peut créer une équipe de créateurs ou rejoindre l'une des équipes existantes. Ayant créé une équipe, ce dernier peut éditer les informations correspondantes, ajouter de nouveaux membres ou supprimer l'équipe.
- **Cas d'utilisation : Consulter statistiques et graphs** Les membres professionnels peuvent consulter les graphs et statistiques concernant leurs projets et la visibilité de leurs profils.
- **Cas d'utilisation : Découvrir les offres de travail** Un membre professionnel bénéficie de la possibilité de consulter les offres de travail postées sur la plate-forme et venant des recruteurs de l'industrie de la Mode.
- **Cas d'utilisation : Gérer les projets** La gestion des projets désigne la possibilité de créer un nouveau projet, l'éditer, accorder le droit de modification à son équipe ou le supprimer. Notre solution se base sur le contenu bâtit par les différents membres. La création d'un projet est avant tout un moyen d'expression pour les membres mais aussi un moyen pour enrichir le contenu du réseau. Un projet peut varier d'un simple article décrivant une nouvelle tendance à la description d'un dessin de patron d'une pièce de vêtement. Le tableau II.1 offre une description détaillée de ce cas d'utilisation.



Figure II.4 – Diagramme UseCase d'un membre professionnel

<b>Acteur</b>	Membre Professionnel
<b>Pré-conditions</b>	Le membre doit être authentifié et possédant un compte professionnel. Il doit accéder à la page de création de projet.
<b>Scénario Nominal</b>	<ul style="list-style-type: none"> <li>– Entrer le titre du projet</li> <li>– Choisir une catégorie de Mode</li> <li>– Choisir une sous-catégorie</li> <li>– Indiquer l'état du projet (Finished / In Progress)</li> <li>– Indiquer la visibilité du projet (Privé / Publique)</li> <li>– Ajouter une description</li> <li>– Ajouter des balises (Tags)</li> <li>– Ajouter une photo de couverture</li> <li>– Rédiger le contenu du projet (Insérer du texte, des images, des vidéos et des liens, effectuer la mise en page)</li> <li>– Soumettre le nouveau projet</li> </ul>
<b>Enchainements d'erreur</b>	Une exception est déclenchée si la validation des paramètres de la requête de création a échoué. On reprend au cas d'utilisation « Créer un projet ».

Tableau II.1 – Description textuelle du UseCase «Créer un projet»

### 3.4 Diagramme de cas d'utilisation des acteurs système

Notre API de web services est sollicitée par l'application web du back-office et par les applications mobiles qui vont être développées par nos coéquipiers. Le diagramme UseCase de la figure II.5 représente ces acteurs systèmes ainsi que les fonctionnalités offertes par l'API. Ces fonctionnalités se résument en trois cas d'utilisation principaux :

- **Cas d'utilisation : Obtenir une autorisation** Ce cas d'utilisation est primordiale pour protéger les ressources gérées par l'API de web services contre les intrusions. En effet, la consommation d'un web services nécessite l'obtention d'une autorisation d'accès préalable sous d'un clé d'accès générée par un web service dédié.
- **Cas d'utilisation : Invoquer les web services (Mobile)** Ce cas d'utilisation nécessite une autorisation. Il permet aux deux applications mobiles iOS et

Android d'invoquer les web services fournissant les différentes fonctionnalités proposées aux visiteurs, membres amateurs et membres professionnels.

- **Cas d'utilisation : Invoquer les web services (Back-office)** Ce cas d'utilisation permet à l'application web back-office d'invoquer les web services fournissant les fonctionnalités proposées au future administrateur du réseau Thimble Fashion. Ce cas d'utilisation nécessite une autorisation préalable.

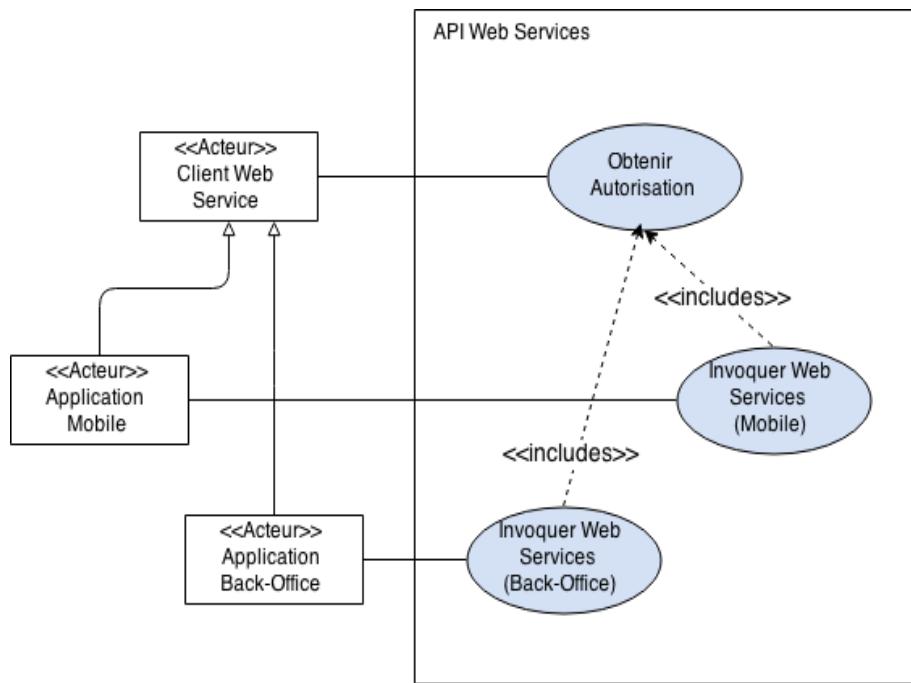


Figure II.5 – Diagramme UseCase des acteurs système

## Conclusion

Dans la première partie de ce chapitre, nous avons arrêté la liste des utilisateurs cibles et défini les spécifications fonctionnelles et non-fonctionnelles du projet. Nous avons par la suite décrit les fonctionnalités accessibles par chacun des utilisateurs à travers les diagrammes de cas d'utilisation. Nous avons décrit les acteurs systèmes de notre solution à la fin de ce chapitre.

Ce travail nous permet d'aborder dans le chapitre suivant les choix architecturaux et techniques pris lors de ce projet et d'entamer la phase de conception.

# Chapitre III

## Architecture et Choix Techniques et Conceptuels

### Introduction

Dans ce chapitre nous allons commencer par présenter l'architecture cible de notre projet. Ensuite, nous allons expliquer les choix techniques pris pour la mise en place de cette architecture. En outre, la dernière partie de ce chapitre sera dédié non seulement aux choix conceptuels et leur concrétisation dans notre solution mais aussi aux patrons de conception employés.

## 1 Architecture cible de la solution

Au vu des spécifications fonctionnelles et non-fonctionnelles, nous proposons l'architecture cible présentée dans la figure III.1 :

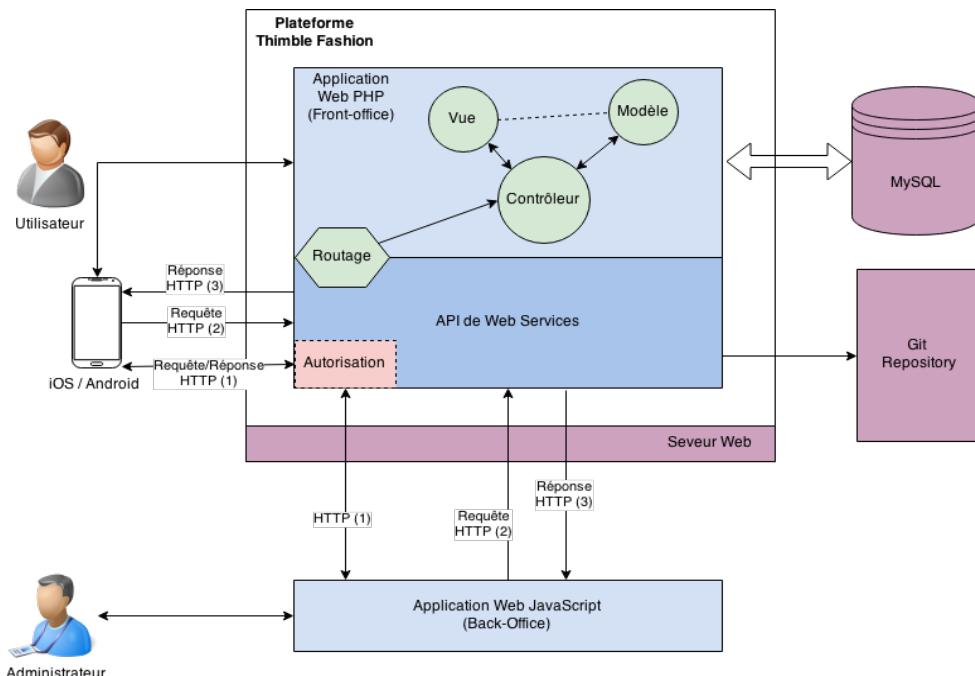


Figure III.1 – Architecture cible de la solution

Dans notre architecture cible, l'application web PHP représente le noyau de la plateforme Thimble Fashion. Cette application web est structurée selon le patron d'architecture logicielle MVC (Modèle-Vue-Contrôleur) qui favorise la séparation entre la couche de présentations, la couche métier et la couche d'accès aux données. Cette application propose une vitrine web pour les utilisateurs opérant à travers un navigateur web. Elle expose aussi une API de web services proposant les fonctionnalités requises par les applications mobiles et l'application web du back-office. Une autorisation préalable est obligatoire pour consommer ces web services en échangeant des requêtes/réponse HTTP.

## 2 Choix Techniques

Afin de mettre en place l'architecture citée précédemment, nous avons étudié et sélectionné un ensemble de technologies à employer tout en tenant compte des choix technologiques spécifiés par le Product Owner de notre projet. Nous allons dans ce qui suit présenter ces technologies en comparant quelques unes avec ce qui existe sur le marché.

### 2.1 Framework d'implémentation de la plateforme Thimble

La réseaux Thimble Fashion est basée sur une application web qui va exposer une vitrine web ainsi qu'une API de web services qui va fournir les fonctionnalités proposées aux visiteurs et membres amateurs et professionnels que nous avons mentionné dans le chapitre précédent. Pour mettre en place cette application web, notre Product Owner a choisi les framework PHP et plus précisément le Framework Laravel.

#### 2.1.1 Un Framework : c'est quoi ?

Un Framework peut être définie comme un cadre logiciel qui facilite le développement des composantes applicatives d'une solution logicielle. Il s'agit généralement d'un ensemble d'API ou librairies qui masquent la complexité d'autres API sous-jacentes [15]. Il permet aussi de guider les développeurs dans le choix technologique et d'architecture afin de pouvoir ajouter des « briques » au projet facilement par la suite.

Un Framework web PHP répond aux besoins de base qui interviennent dans la majorité des projets de développement web en PHP. Il doit par exemple prendre en compte nativement les éléments suivants [16] :

- **Modèle MVC** : Model-View-Controller ou Modèle-Vue-Contrôleur qui est un patron d’architecture permettant de structurer une application en distinguant entre la partie présentation, la partie de gestion des données et la partie applicative.
- **Cache** : Il permet de stocker les pages en cache afin d’optimiser leur temps de chargement.
- **Gestion des SGBD** : Il doit pouvoir gérer plusieurs types de base de données à travers l’intégration des pilotes des Systèmes de Gestion de Base de Données correspondants.
- **ORM** : Object-Relational Mapping ou Mapping Objet-Relationnel qui est un module permettant l’accès et la manipulation des données d’une base de donnée relationnelle à travers des objets.
- **Conventions** : Il facilite aux développeurs d’utiliser les mêmes conventions de codage afin d’avoir un code uniforme.
- **URL conviviales** : Le Framework doit fournir la fonctionnalité de gestion des règles de routage et de redirection. Il doit pouvoir gérer les URLs (Uniform Resource Locator) facilement.

#### **2.1.2 Framework Laravel**



**Figure III.2** – Logo du framework Laravel [17]

Laravel est un Framework Web PHP respectant le modèle MVC et entièrement développé avec le paradigme de la Programmation Orientée Objet (POO). Il est devenu l’un des Frameworks PHP les plus utilisés et les plus reconnus au monde suite à la sortie de sa version 4. Ce Framework a été créé par Taylor Otwell en juin 2011 et il est

sous la licence MIT [17].

Laravel est basé sur une philosophie générale de développement qui priorise la qualité et la maintenabilité du code. Il suffit de suivre un ensemble de directives simple pour aboutir à un rythme de développement rapide et à un code facilement maintenable. Ceci est garanti grâce l'incorporation d'un ensemble de patrons de développement web prouvés et de bonnes pratiques parmi lesquelles on peut citer [18] :

- **Single Responsibility Pattern** : L'architecture MVC de Laravel est très utile du point de vue d'un développeur puisqu'elle sépare les différents composants d'une application web. Ceci rend la manipulation d'un composant plus facile sans se soucier de bouleverser le code des autres composants.
- **Convention over configuration** : Laravel défini une hiérarchie pour le dossier d'une application tout en respectant un ensemble de conventions de nommage des fichiers, des classes et des tables de la base de données. Laravel profite aussi de ces conventions pour interconnecter les différents éléments d'une application sans recourir à plusieurs configurations. Ces conventions facilitent le travail en équipe et permettent au développeur de se concentrer sur la logique métier de son application ainsi que de développer rapidement des applications puissantes.
- **Don't Repeat Yourself (DRY)** : Laravel facilite l'application du principe DRY qui consiste à écrire une fonctionnalité une et une seule fois et de favoriser la réutilisation. En effet, ce Framework fournit un environnement ayant des outils simplifiant la réutilisation d'un code dans différents composants de l'application.
- **Tests Unitaires** : Laravel donne assez d'importance aux tests. Les tests sont généralement effectués en parallèle avec la phase d'implémentation afin d'assurer que le code fonctionne comme prévu et continue à fonctionner même en changeant les données manipulées par exemple.

L'entreprise Logicale a adopté depuis quelques années le Framework PHP Laravel pour la mise en place de différents projets web grâce aux multiples avantages qu'il présente ainsi que la grande communauté active autour de lui. Laravel facilite énormément l'implémentation des web services scalables. Par conséquent, notre Product Owner a souligné que l'implémentation la partie web ainsi que l'API de web services sera faite avec la dernière version de Laravel 5.0.2.

## 2.2 API de web services

L'implémentation de l'API de web services est faite à travers le Framework Laravel en exposant les fonctionnalités appropriées aux applications mobiles et back-office tout en gérant les préoccupations transversales qui les concernent telles que la sécurité. L'utilisation des web services permet plus d'interopérabilité entre ces différents composants hétérogènes grâce aux protocoles et standards qu'ils mettent en œuvre. En effet, un web service peut être défini comme étant un composant logiciel permettant l'échange de données et la communication entre applications et systèmes hétérogènes dans des environnements distribués. Selon le livre d'Eben HEWITT intitulé Java SOA Cookbook [19], un web service est caractérisé par les points suivants :

- Il est disponible à travers un réseau.
- Il est défini par une interface qui est indépendante de la plateforme.
- Les opérations définies dans l'interface exercent des fonctions métier car elles opèrent sur des objets métier.

Actuellement, le choix du moyen d'implémentation des web services se limite aux deux approches utilisées couramment dans le développement de web services : l'architecture REST (REpresentational State Transfer) et le protocole SOAP (Simple Object Access Protocol). Nous tenons à noter que Laravel supporte ces deux approches.

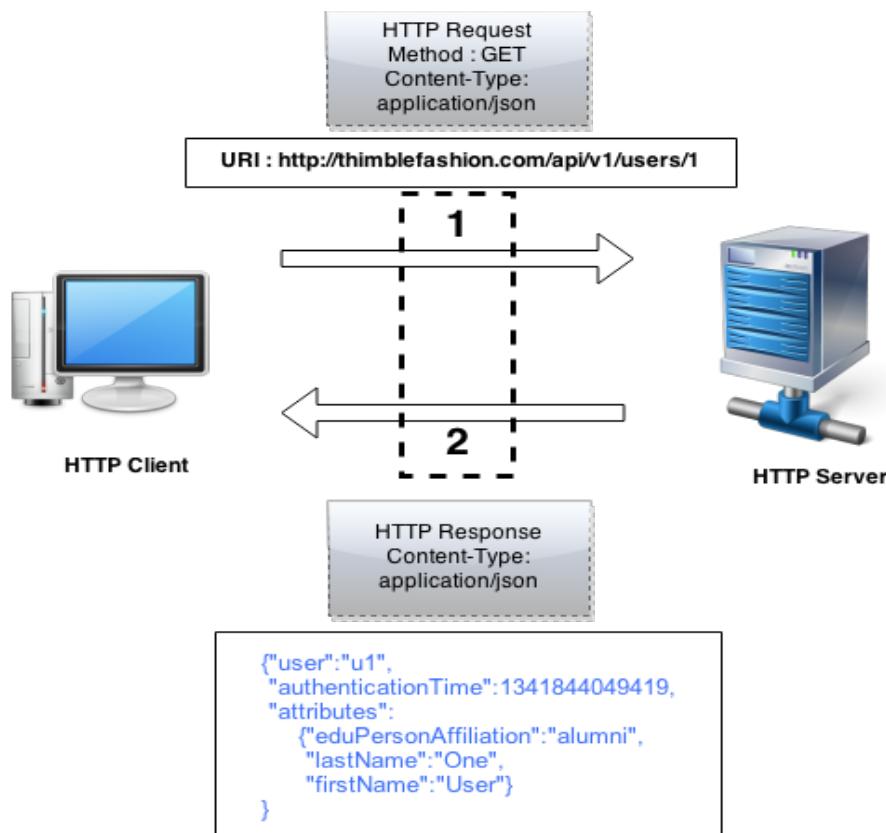
### 2.2.1 REpresentational State Transfer (REST)

REST est un style d'architecture qui utilise les standards du web et sur lequel on peut construire des services lors de la mise en place des systèmes distribués. C'est Roy Fielding qui a introduit ce style d'architecture dans sa thèse en 2000. REST repose sur un principe simple selon lequel la combinaison du protocole HTTP (HyperText Transfer Protocol) et URI (Uniform Resource Identifier) satisfait largement les besoins d'un web service en utilisant les différentes méthodes HTTP (GET, POST, PUT, DELETE, etc.). REST est considéré aussi comme une « Architecture Orientée Ressource » selon Leonard RICHARDSON et Sam RUBY qui ont expliqué la notion de « Ressource » et son importance dans les quatre concepts du REST énoncés dans leur livre RESTful Web Services [20] :

- **Les ressources** : Ce sont les objets manipulés par les services REST. Une ressource désigne un objet qu'on peut enregistrer numériquement et qui peut être un objet physique ou un concept abstrait.

- **Les URI des ressources** : C'est un moyen universel pour identifier une ressource. Cette dernière ne peut pas exister sans un URI.
- **Liens et connectivité** : Pour assurer la navigation entre les différentes ressources au sein d'une application REST, on utilise des liens sous forme d'URI et de lien hypermédias.
- **Les représentations** : Une ressource peut être représentée sous différentes formats (XML, JSON, etc.) dans la réponse HTTP. Le format souhaité par le consommateur du service peut être défini dans l'entête de la requête HTTP.

La figure III.3 illustre un exemple de consultation d'un web service RESTful par un client HTTP tout en précisant le format de réponse souhaité.



**Figure III.3 – Échange Client-Serveur REST**

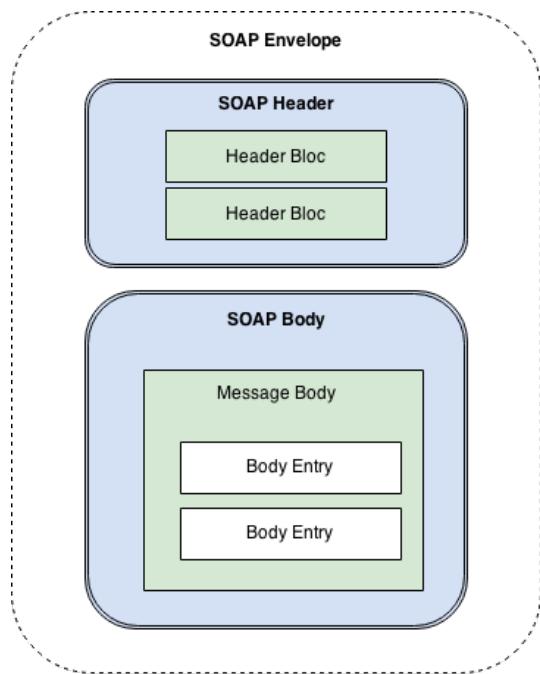
Le point fort du REST est qu'il est léger et simple de manière que les messages sont courts et faciles à décoder par le navigateur et par le serveur d'application. REST est

aussi auto-descriptif. En effet, il y a un URI intuitif unique pour chaque ressource. Il faut noter aussi que REST supporte la gestion en cache ce qui lui rend plus performant [21].

Le fait que REST est un moyen de communication sans état (Stateless) peut être vu à la fois un avantage et un inconvénient. En effet, REST est parfait pour les web services ayant des opérations simples telles que créer, lire, mettre à jour ou supprimer une ressource. En revanche, il n'est pas adapté aux transactions longues et complexes. Pour simuler la gestion d'état, le client ou le serveur est obligé à de persister les données nécessaires au bon déroulement d'une requête. Ceci peut conduire à une consommation de bande réseau plus importante.

#### 2.2.2 Simple Object Access Protocol (SOAP)

SOAP est un protocole de communication basé sur XML servant à transporter échangées entre applications via HTTP. Il permet ainsi l'accès aux services web et l'interopérabilité des applications à travers le web. SOAP est un protocole simple et léger et qui repose entièrement sur des standards établis comme le HTTP et XML. Il est portable et donc indépendant de toute plateforme. SOAP est une spécification non propriétaire. Il enrichie la spécification HTTP qu'il utilise avec une couche d'abstraction au dessus de ce protocole. Il utilise le format d'échange XML qui apporte son extensibilité permettant ainsi de définir de nouveaux types de données par exemple. Les messages SOAP sont structuré en un document XML et comporte deux éléments obligatoires : Une enveloppe et un corps (une entête facultative) [22]. La structure des messages SOAP est schématisée dans la figure III.4.



**Figure III.4** – Structure d'un message SOAP

Le schéma ci-dessus indique les éléments de base d'un message SOAP :

- Une enveloppe qui définit le contenu du message.
- Un en-tête optionnel qui contient les informations d'en-tête (autorisations et transactions par exemple)
- Un corps contenant les informations sur l'appel et la réponse.

Ce protocole étant standardisé par le W3C permet une adaptabilité à différents protocoles de transport et une meilleure extensibilité. Ce standard supporte aussi les transactions ACID de sorte qu'il peut gérer un commit à deux phases sur des ressources transactionnelles distribuées. En outre, ce protocole dispose des mécanismes pour garantir la fiabilité de la transmission et de la réception des données.

Toutefois, SOAP présente quelques inconvénients. En effet, SOAP est considéré trop complexe et verbeux pour les besoins ordinaires. SOAP nécessite une compréhension claire des APIs spécifiques au site. Il demande du temps et de la documentation. De plus, l'interopérabilité de SOAP a diminuée au fil des années, avec des normes différentes et les implémentations des fournisseurs de logiciels. Désormais il est certain que REST est de loin plus interopérable [21].

### 2.2.3 Choix de la technologie d'implémentation des web services

Pour l'implémentation de l'API de web services, nous avons décidé d'utiliser l'approche REST et de profiter des avantages qu'elle propose. Notre but était de développer rapidement une API qui soit légère, scalable et ayant un schéma intuitif et simple à comprendre. Nous avons aussi pris en considération compatibilité de ce style d'architecture avec les choix technologiques effectués auparavant. En effet, Laravel facilitent l'implémentation des API REST en offrant par exemple des contrôleur de ressources permettant de retourner les données provenant de son ORM Eloquent directement sous format JSON. Les web services RESTful sont beaucoup plus simples à consommer sous iOS, Android et JavaScript ce qui est bénéfique pour l'application web back-office et les applications mobiles du réseau Thimble Fashion. Finalement, REST est considéré en quelque sorte comme un artifice du « web 2.0 » et il domine clairement depuis des années le marché des API de web services comme le démontre la figure III.5. Cette figure présente le résultat d'analyse effectuée par le site ProgrammableWeb [23] sur une liste de 5100 APIs en Février 2012. Ce résultat confirme la domination du style d'architecture REST.

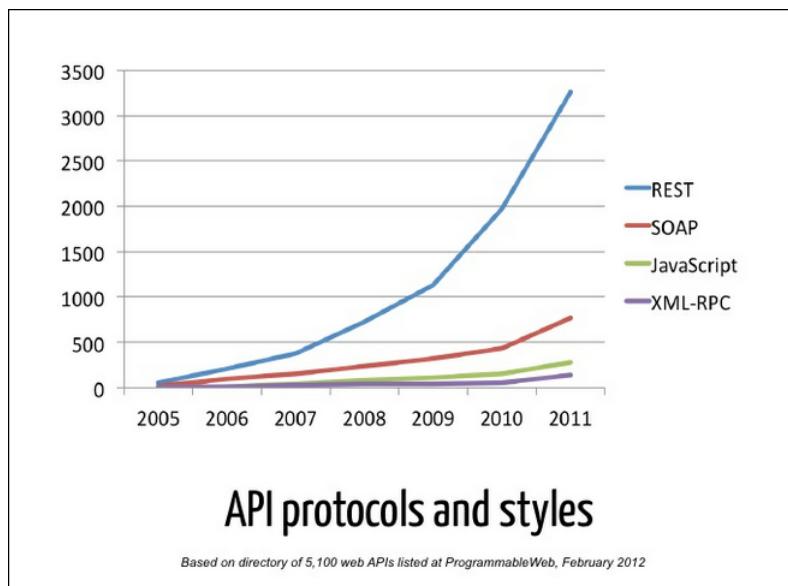


Figure III.5 – Analyse des Protocoles et des Styles d'API 2012 [23]

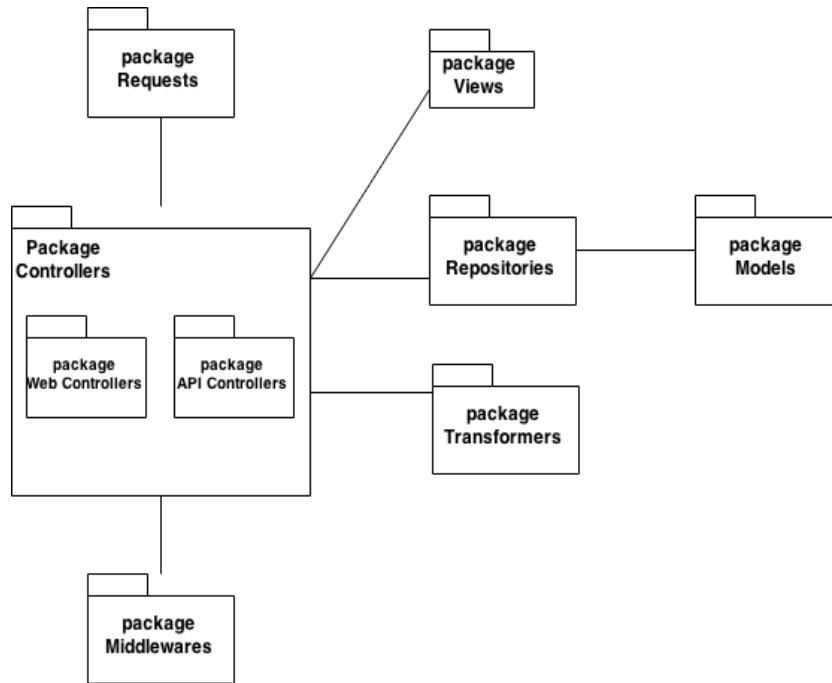
## 3 Conception de la solution

Nous allons traiter dans cette partie la conception notre solution à travers une modélisation UML. Nous allons commencer avec une conception générale expliquant la structuration des packages de notre solution. Nous allons ensuite apporter plus de

détails à travers les diagrammes de classes et de séquences.

### 3.1 Diagramme de Packages

La figure III.6 présente l'organisation et la répartition du code de notre application front-office ainsi que l'implémentation des web services de notre API REST.



**Figure III.6** – Diagramme de packages Web et API REST

Dans le diagramme de la figure III.6, nous distinguons sept packages. Cette répartition a été faite afin de garantir une meilleure modularité, lisibilité et maintenabilité du code. Cette décomposition nous permet aussi de mieux respecter le patron logiciel MVC et de gérer plus facilement les préoccupations transversales à savoir la sécurité. Nous avons répartie notre code tout en respectant la bonne pratique DRY (Don't Repeat Yourself) et en favorisant la réutilisabilité. Dans ce qui suit, nous allons décrire brièvement ces sept packages :

- **Package Models :**

Ce package est dédié aux classes Modèles de l'architecture MVC. Chaque classe de ce package va être représentée par une classe de type Model correspondant à l'implémentation ORM Eloquent du framework Laravel. Ces classes concrétisent les entités de notre solution ainsi que les relations qui les relient et elles sont

responsables de la manipulation des données et leur persistance. Nous allons apporter plus de détails à ce package avec son diagramme de classes dans le point suivant.

– **Package Controllers :**

Ce package représente la partie Contrôleur du modèle MVC. Il regroupe deux sous-packages : Package « Web Controllers » (ensemble des contrôleurs de l'application front-office) et Package « API Controllers » (ensemble des contrôleurs de l'API REST). Cette séparation a été faite afin de ne pas centraliser le traitement des requêtes provenant des utilisateurs web et des utilisateurs REST dans un seul type de contrôleur. Une telle centralisation peut éventuellement alourdir le traitement et rendre difficile le respect de spécifications fonctionnelles et non-fonctionnelles de l'API REST et de la vitrine web qui diffèrent en quelques points. Ces deux sous-packages partagent une partie importante de la logique métier. Par conséquent, nous avons réutilisé le traitement commun entre ces deux types de contrôleurs en l'externalisant sous forme de packages et en appliquant les patrons de conceptions adéquats. L'annexe 1 présente le diagramme de classes de ce package « Web Controllers ». On va détailler le package « API Controllers » dans les points suivants.

– **Package Repositories :**

Pour l'accès aux données, nous avons appliqué le patron de conception Repository que nous allons détailler dans la partie des designs patterns. Les classes de ce package fournissent une implémentation utilisant les classes du package « Models » permettant aux classes des contrôleurs la manipulation des données. Ceci garantie un découplage entre la couche métier et la couche d'accès aux données. Le diagramme de classes de ce package est présenté dans l'annexe 1.

– **Package Requests :**

Nous avons utilisé le package « Requests » afin de regrouper les classes représentant les principaux types de requêtes pouvant être reçus par nos contrôleurs. Chaque classe de requête effectue la validation des données tout en respectant la suite de règles de validations ayant été définies dans cette classe. L'annexe 1 présente le diagramme de classes de ce package.

– **Package Middlewares :**

Concernant le package « Middlewares », c'est un package regroupant les classes

responsables des préoccupations transversales telle que la sécurité. Les classes de packages respectent le patron de conception « Chain of Responsibility ». En effet, toutes les requêtes provenant aux contrôleurs sont traitées préalablement en chaîne par les classes « Middlewares ». Elles passent en premier par un middleware vérifiant l'autorisation d'accès à l'API REST. Si la requête est bien autorisée, ce dernier passe la requête au middleware responsable de la gestion des sessions et d'authentification des membres et ainsi de suite. Le diagramme de classes de ce package est présenté dans l'annexe 1.

– **Package Transformers :**

Le package « Transformers » est utilisé uniquement par les contrôleurs de l'API REST. En effet, les classes de ce package sont responsables de l'adaptation et la décoration des réponses JSON provenant des contrôleurs RESTful. Ils permettent en fait de modifier les réponses JSON retournées en sélectionnant les attributs à afficher, les renommer ou les convertir. Ce package joue le rôle d'une couche de présentation pour les contrôleurs de l'API REST. L'annexe 1 présente le diagramme de classes de ce package.

– **Package Views :**

Ce package regroupe l'ensemble des vues du modèle MVC. Elles constituent l'interface utilisateur du front-office du réseau Thimble Fashion ainsi que les vues formatée des corps des emails qui peuvent être envoyés aux utilisateurs. Elles récupèrent les données et les actions de l'utilisateur sous forme de requêtes qui vont être routées vers le contrôleur web adéquat. Ce contrôleur traite la requête et suivant le résultat obtenu il redirige l'utilisateur vers la bonne vue. Nous avons opté pour la réutilisation lors de la détermination des vues en profitant du Template Engin (Engin de Gabarits) Blade. Nous avons décortiqué les vues en appliquant une hiérarchie de gabarits et en externalisant les interfaces réutilisables sous formes de vues partielles.

## 3.2 Diagramme de classes des modèles de domaine

La figure III.7 détaille le package « Models » à travers le diagramme de classes de domaines qui montre l'ensemble des entités de notre solution ainsi que les relations entre eux. C'est en se basant sur cette représentation qu'on a créé le modèle physique de la base de données.

### III.3 Conception de la solution

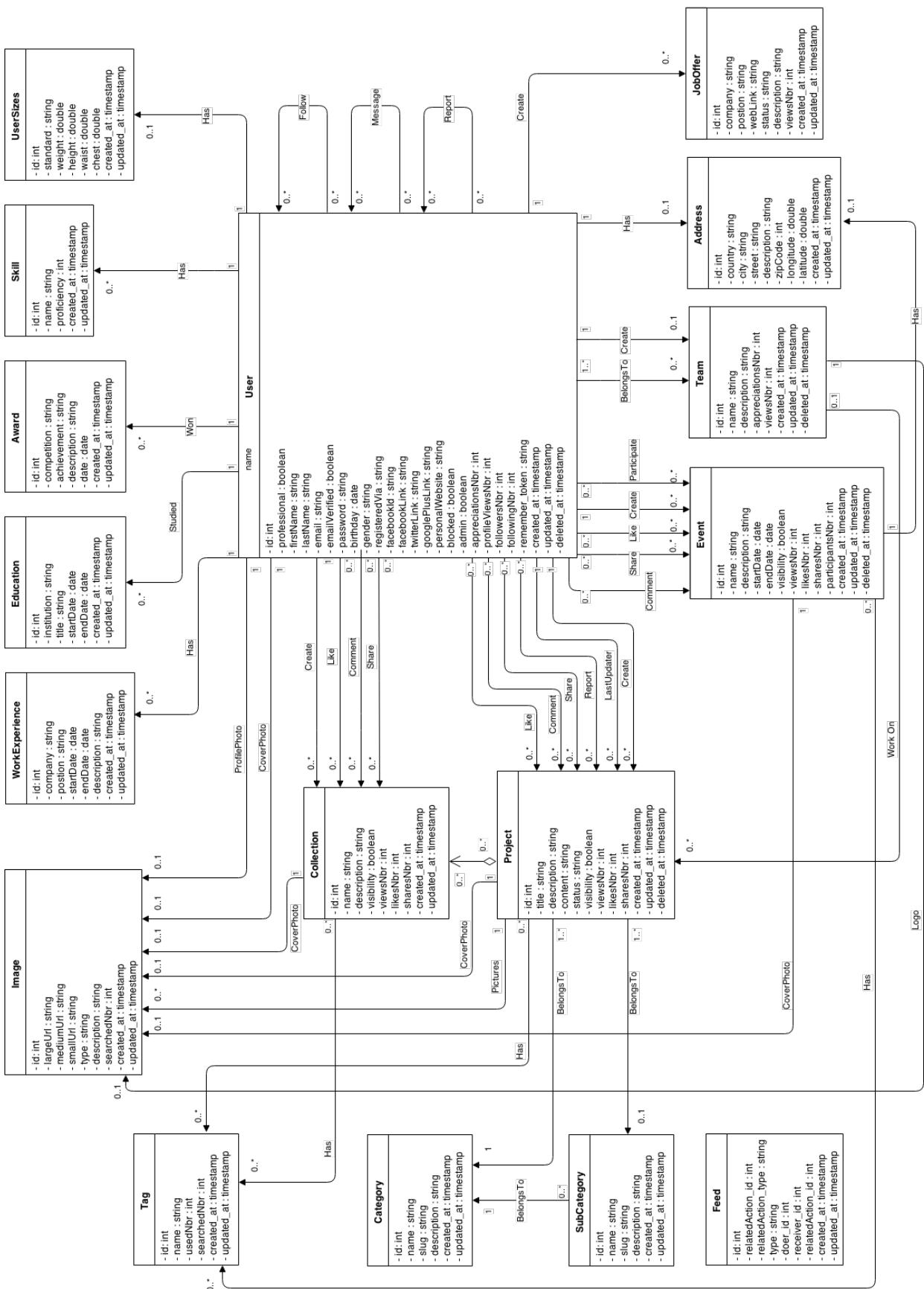


Figure III.7 – Diagramme de classes des modèles de domaine

Dans le modèle présenté par la figure III.7, nous distinguons quatre classes principales dans notre solution : Utilisateur « User », Projet « Project », Équipe « Team » et Evènement « Event ».

L'entité « User » regroupe les informations de base d'un utilisateur inscrit dans notre plateforme telles que sont email et mot de passe crypté, ses informations personnelles et son statut (Membre Amateur/Professionnel, Administrateur). L'attribut « remember\_token » va servir pour empêcher l'expiration de la session d'un utilisateur s'il l'indique au moment de son authentification. Un utilisateur possède aussi une entité de la classe « Address ». Un Utilisateur peut éventuellement suivre (follow) un ou plusieurs utilisateurs, leurs envoyer des messages (message) ou signaler (report) un utilisateur ou un projet. Afin d'exposer son CV, une entité User peut posséder des entités « WorkExperience » (expérience professionnelle), « Education » (formation académique), « Skill » (compétence) et « award » (prix). Une entité « User » peut posséder aussi une entité « UserSizes » (les tailles d'un utilisateur) qui va servir pour le lancement de la boutique de vêtements en ligne dans les prochaines phases du projet global Thimble. Un utilisateur peut avoir une photo de profile et une photo de couverture représentée chacune par une entité de la classe « Image ». Ayant le statut d'un administrateur, un utilisateur peut créer des offres de travail.

L'entité « Project» regroupe les informations de base des projets créés par les utilisateurs professionnels. Ces informations sont principalement son titre, sa description, sa visibilité (publique ou privé), son statut (fini ou en cours), son contenu qui contient le corps du projet sous forme d'un code HTML riche. Les projets sont classés par sous-catégories dans notre solution. En effet, une catégorie peut posséder une ou plusieurs sous-catégories. Un projet peut appartenir directement à une catégorie si cette dernière ne possède pas de sous-catégories. Un projet possède éventuellement une photo de couverture et un ensemble de photos inclue dans son corps. Chacune des photos mentionnées correspond à une entité de la classe « Image ». Afin de faciliter l'annotation et la recherche des projets, une entité « Project » peut être marquée par une ou plusieurs entités de la classe « Tag » (balise). Le lien d'agrégation entre la classe « Project » et la classe « Collection » concrétise le fait qu'un utilisateur peut regrouper un ensemble de projets dans l'une des collections qu'il a créé.

La classe « Team » représente les équipes de travail qui peuvent être fondées par les utilisateurs professionnels. Une équipe peut contenir un ou plusieurs membres. Une entité de la classe « Team » peut posséder un logo qui va être concrétisé par une entité

de la classe « Image ».

Concernant la classe « Event », elle correspond à l'ensemble des évènements de la Mode créés par les utilisateurs professionnels. Cette classe décrit les informations de base d'un événement qui sont son nom, sa description, sa date de début et de fin ainsi que sa visibilité. Un événement peut posséder éventuellement une entité de la classe « Image » qui représentera sa photo de couverture.

Finalement, les activités d'interaction sociale telles que les « Follows », les « comments » et les « Likes » ainsi que la création d'un projet, événement, collection ou équipe engendre la création d'une entité de la classe « Feed » (actualité). Ceci va servir comme trace pour l'analyse de l'activité des utilisateurs sur la plateforme, l'élaboration des statistiques et la constitution des fils d'actualités pour chacun des utilisateurs.

### 3.3 Diagramme de Classes : Package API Controllers

Le diagramme de classes suivant détaille l'ensemble des classes du package « API Controllers » et leur hiérarchie. Nous avons divisé ce diagramme sur deux parties ([A.1](#) et [A.2](#)) par souci de clarté. Dans ce package, nous avons dégagé l'ensemble des attributs et des méthodes qu'un ensemble de contrôleur de ressources REST peuvent avoir en commun. Nous avons regroupé ces éléments communs dans une classe mère qu'on a appelé « APIController ». Cette classe contient un ensemble de constantes représentant les différents codes de statuts d'une réponse HTTP qu'on a besoin. Cette classe possède aussi une variable « statusCode » qui contient le code de statut courant. Les méthodes fournies par la classe « APIController » sont l'accesseur et le mutateur de l'attribut « statusCode » ainsi qu'une suite de méthode de réponse qui retourne une réponse HTTP avec le bon code de statut. Cette classe propose aussi la méthode « respondWithPagination » qui permet d'inclure les différents données de pagination dans la réponse HTTP (exemple : Nombre d'éléments par page, Nombre totale des pages, l'URL de la page suivante/précédente, etc.). Les classes héritant de « APIController » exploitent les méthodes héritées afin d'implémenter les web services de chacune des ressources de notre API.

### III.3 Conception de la solution

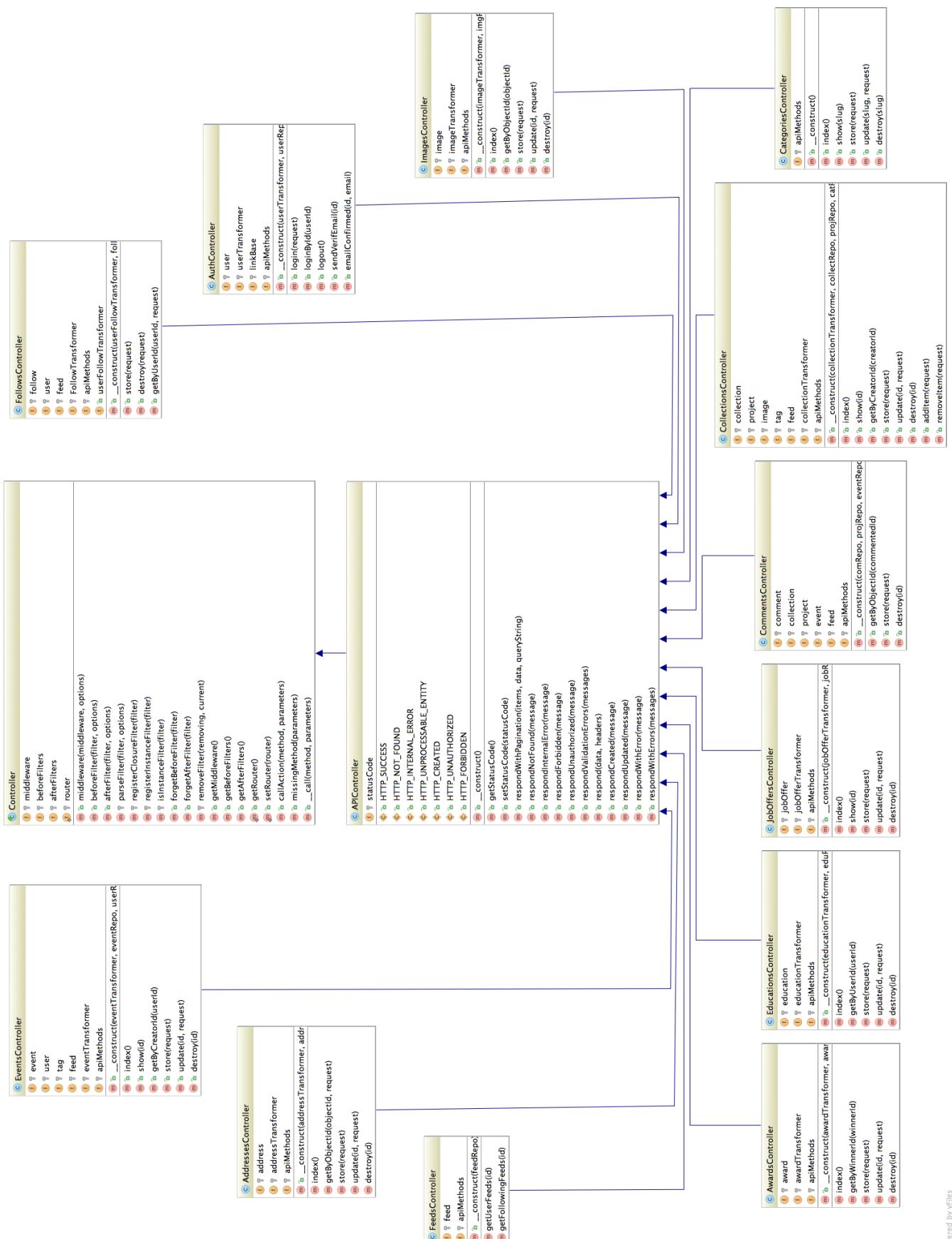


Figure III.8 – Diagramme de Classes : Package API Controllers Partie 1

Powered by rFMS

### III.3 Conception de la solution

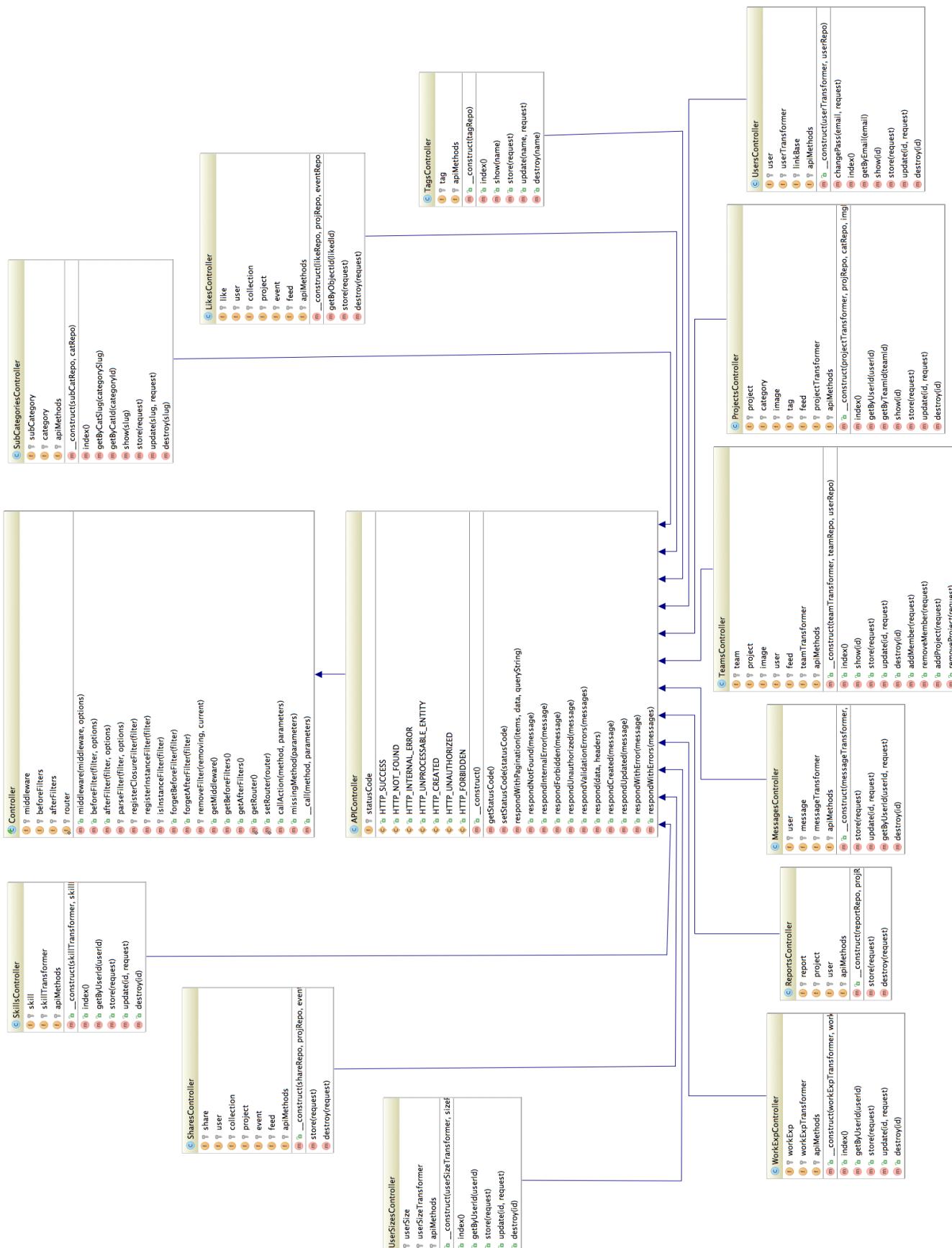


Figure III.9 – Diagramme de Classes : Package API Controllers Partie 2

Powered by yFiles

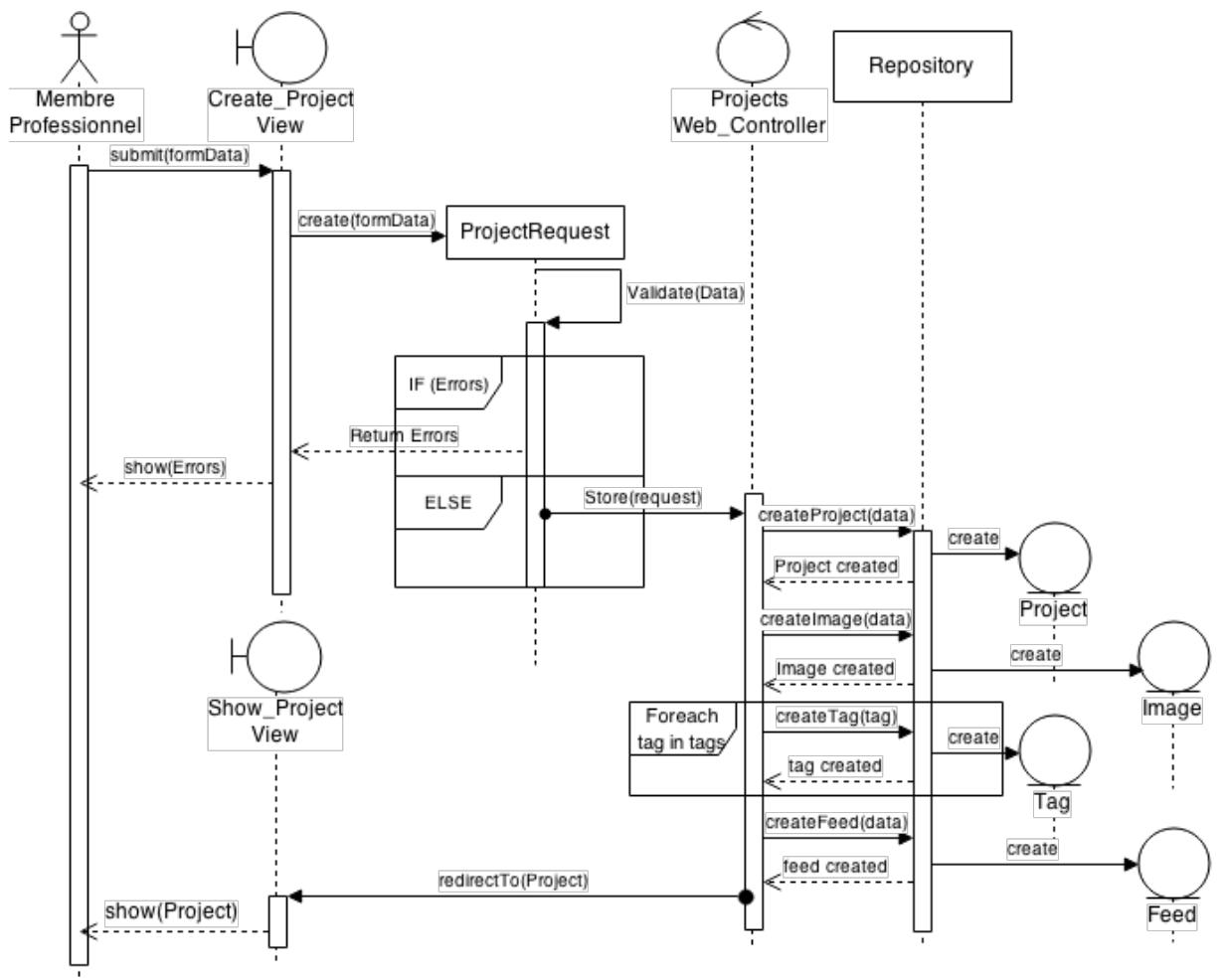
Pour acheminer les requêtes HTTP vers le bon web service RESTful, ce dernier doit être identifié par une URI unique et doit correspondre à une méthode HTTP. Nous avons respecté le schéma décrit dans le tableau III.1 pour le routage des requêtes. L'utilisation de ce schéma nous a permis d'avoir des routes lisibles et intuitives.

Méthode HTTP	URI	Service REST
GET	/api/v1/ressource	Index : Lister toute les instances d'une ressource
GET	/api/v1/ressource/ :id	Find : Trouver l'instance de la ressource ayant l'id spécifié comme paramètre fragment.
POST	/api/v1/ressource	Store : Sauvegarder une nouvelle instance de la ressource avec les données fournies dans la requête POST
PUT	/api/v1/ressource/ :id	Update : Mettre à jour l'instance ayant l'id spécifié avec les données fournies dans la requête PUT
DELETE	/api/v1/ressource/ :id	Destroy : Supprimer l'instance ayant l'id spécifié.

Tableau III.1 – Schéma de Routage REST

### 3.4 Diagramme de séquences : Crédation d'un projet

Dans le diagramme de séquence de la figure III.10, nous allons donner une modélisation dynamique du scénario de création d'un projet par un utilisateur professionnel à travers l'application front-office. Nous allons indiquer les éléments participant dans l'exécution de cette opération ainsi que les échanges qui ont eu lieu entre eux.



**Figure III.10** – Diagramme de séquence de création d'un projet

Dans le diagramme de séquence précédent, un membre professionnel accède à la page de création de projet après avoir été authentifié. Une fois qu'il a fini l'entrée des informations du projet, la rédaction et la mise en page du contenu et la sélection de la photo de couverture, il procède à la soumission du formulaire qui va engendrer la création d'un requête HTTP Post de type « ProjectRequest » et héritant de la classe « FormRequest ». L'instance de cette requête va se charger de la validation des données selon les règles prédéfinies. Si la validation échoue, l'instance « ProjectRequest » retourne les messages d'erreurs à la vue « Create\_Project » pour les afficher à l'utilisateur. En cas de succès de la validation, l'instance de cette requête va être injectée dans la méthode « Store » du contrôleur web des projets. Ce dernier va passer par les classes Repository adéquates pour créer une nouvelle entité « Project » dans la base de données ainsi que les entités en relation avec le projet à savoir la photo de couverture, les tags et le feed. Par souci de clarté, les classes Repository utilisées par le contrôleur web des projets ont été regroupées en une seule classe sachant qu'elles effectuent la

même action de création d'une instance du modèle correspondant et de sa sauvegarde dans la base de données.

#### 3.5 Conception de la base de données

Afin de concevoir notre base de données relationnelle, nous nous sommes basé sur le modèle de domaine expliqué précédemment. Notre solution va sauvegarder les données des utilisateurs et le contenu qu'ils créent, leurs sessions, les clés d'autorisation de l'API REST. La trace de l'activité des utilisateurs va être persistée tout en étant classifiée et répartie dans différentes tables de la base de données selon le type de l'action générant cette trace. Nous avons aussi toléré l'emploi des champs calculés dans différentes tables de notre base de données (exemple : la persistance du nombre de followers et de following dans la table « users » ou la persistance du nombre de participants dans la table « events »). En outre, nous avons décidé d'attribuer des Index aux colonnes qui vont être utilisées fréquemment pour la recherche des enregistrements dans la base de données. Nous avons jugé que l'emploi de ces pratiques va nous permettre de gagner en temps de recherche et de récupération des données. Finalement, nous avons défini un identifiant technique dans chaque table. Cette identifiant n'a aucune signification métier et il va servir juste pour assurer l'unicité et l'intégrité des données persistées.

L'annexe 2 présente un extrait du modèle physique correspondant à la base de données conçue.

### 4 Design patterns employés

Les Design Patterns (Patrons de conception) sont un ensemble de bonnes pratiques et solutions fiables et standardisées pour des problèmes de programmations récurrents suivant le paradigme orienté objet. De plus, l'utilisation correcte de ces patrons permet d'obtenir des applications robustes, extensibles et simples à maintenir.

#### 4.1 Design pattern : Repository

Le patron de conception Repository est employé afin de découpler entre la logique d'accès aux données et la couche métier. La communication entre ces deux couches se fait à travers des interfaces. En d'autres mots, ce design pattern permet à la logique métier d'accéder aux modèles de données sans avoir à connaître l'architecture d'accès aux données utilisée [24].

La séparation entre la logique métier et l'accès aux données présente différents avantages parmi lesquels on peut citer :

- La centralisation de la logique d'accès aux données ce qui facilite la maintenance du code correspondant.
- La logique métier et la logique d'accès aux données peuvent être testées séparément.
- Possibilité de mise en œuvre de différente implémentation d'accès aux données respectant le même contrat reconnu par la logique métier.

Nous avons décidé de profiter des avantages cités précédemment en adoptant ce design pattern pour le découplage entre le logique métier assuré par les contrôleurs web et API et la couche d'accès aux données qui se base essentiellement sur les modèles de données. Dans le diagramme de classes de la figure III.11, nous allons présenter un exemple d'utilisation de ce patron de conception par le contrôleur web « UsersController ».

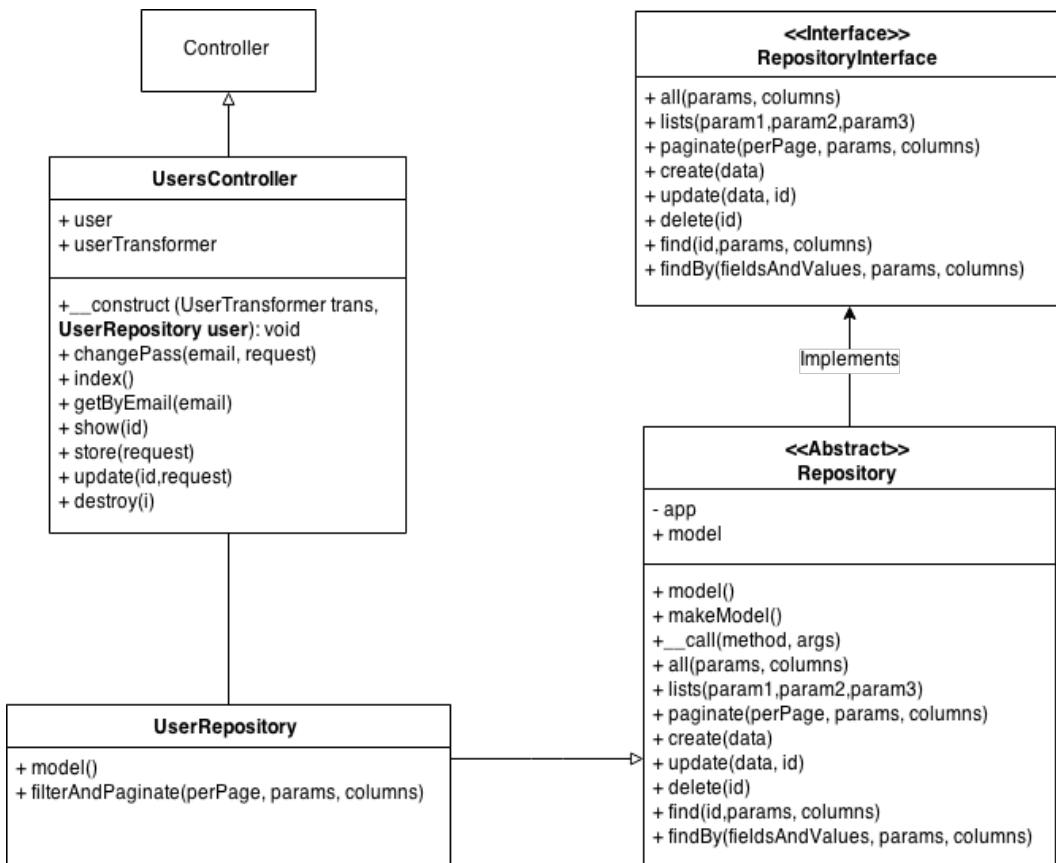


Figure III.11 – Diagramme de classes Design Pattern Repository

Dans le diagramme de classes précédent, la mise en place du patron de conception Repository est faite à travers l'interface « RepositoryInterface » qui représente le contrat à satisfaire par toute implémentation de la couche d'accès aux données. Dans notre solution, nous avons une seule implémentation Repository représentée par la classe abstraite « Repository » et ses différentes classes filles. Cette implémentation est basée sur l'ORM Eloquent de Laravel. Parmi les classes filles de « Repository » on peut trouver « UserRepository » qui assure l'accès au modèle « User ». La classe du contrôleur web « UserController » utilise une instance de la classe « UserRepository » qui est injectée dans son constructeur afin de pouvoir créer de nouveaux utilisateurs et récupérer les données de ceux déjà existants.

## 4.2 Design pattern : Factory Method

Le design pattern « FactoryMethod » ou Fabrique en français est l'un des patrons de conception les plus simple à comprendre et à mettre en œuvre. Il est un design pattern de création faisant partie des 23 Design Patterns de GoF (Gang of Four). Son objectif principal est de construire et fournir un objet configuré correctement et prêt à l'emploi. Il permet ainsi de libérer le code client du choix de l'implémentation, la configuration et l'instanciation de l'objet [25].

Nous avons tenu à profiter des avantages de ce patron de conception dans notre projet. À titre d'exemple, nous avons appliqué ce patron au niveau des classes de test unitaire de la couche d'accès aux données et plus précisément les tests du package « Repository ». La figure III.12 présente notre exemple d'application de ce design pattern.

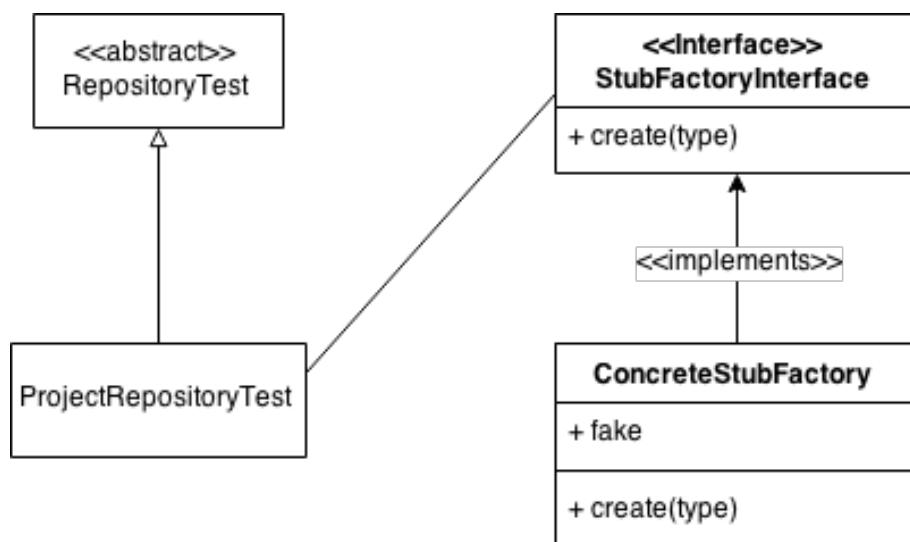


Figure III.12 – Structure du design pattern Factory Method

Dans le diagramme de classes précédent, la classe « ProjectRepositoryTest » implémente le cas de test (Test Case) des méthodes de la classe « ProjectRepositorry ». Nous avons besoin de créer des objets fictifs (Stub) pour les différents type de modèles durant les tests. Avec le patron de conception Fabrique, nous avons délégué linstanciation et la configuration des objets fictifs à la classe implementant linterface « StubFactoryInterface » et plus précisément la méthode « create ». Cette méthode instancie un objet ayant comme type celui qu'elle prend en paramètre et le configure avant de le retourner au code client.

## **Conclusion**

Nous avons décortiqué progressivement dans ce chapitre la solution à réaliser. Nous avons commencé par présenter le choix larchitecture cible et des technologies à employer, pour passer ensuite à la conception de lapplication front-office et lAPI REST. Nous avons fini avec la description dune sélection de patrons de conceptions qui ont été mis en œuvre dans notre projet.

Dans le chapitre suivant, nous exposons la réalisation de notre travail ainsi que les résultats des tests obtenus.

# Chapitre IV

## Mise en Œuvre et Tests

### Introduction

Dans ce chapitre nous allons exposer les réalisations faites au cours de ce projet tout en décrivant l'utilisation des technologies mises à notre disposition. La dernière partie de ce chapitre contiendra une présentation des tests effectués sur la solution, notamment les tests de performance de l'API REST.

### 1 Architecture finale de la solution

Notre solution a été structurée selon le schéma d'architecture présenté dans la figure IV.1 :

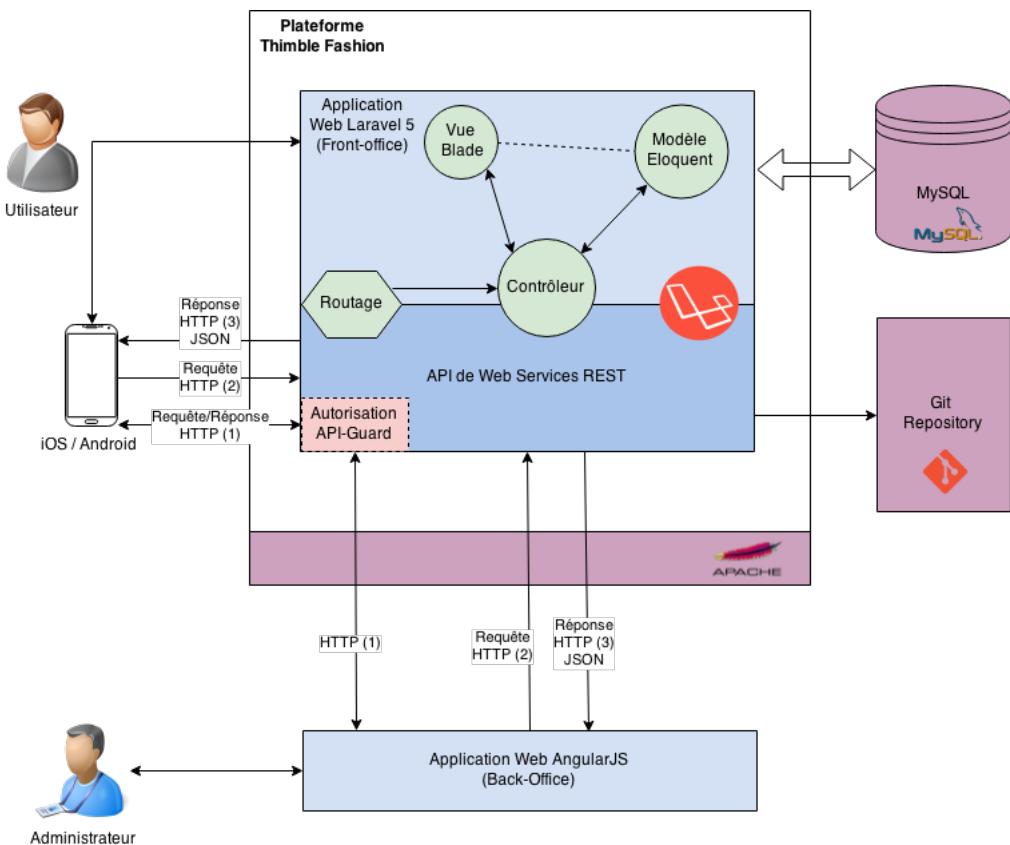


Figure IV.1 – Architecture finale de la solution

L'architecture précédente expose l'ensemble des composants de notre solution ainsi que les technologies qui les sous-tendent. L'application web Laravel 5 représente le noyau de notre solution puisqu'elle englobe la front-office du réseau social professionnel Thimble Fashion et l'API de web services REST. Cette application gère aussi la persistance des données au niveau d'un serveur SGBD MySQL. Concernant l'API REST, elle va être sécurisée par le module de contrôle d'autorisation API-Guard. L'API de web services va être consommée par les applications mobiles (iOS/Android) et par l'application back-office. La communication avec l'API se fait à travers les requêtes HTTP et les données des réponses HTTP vont être retournées sous format JSON. Il faut noter que la gestion des versions est faite au niveau d'une repository Git.

## 2 Environnement de travail

Dans cette partie, nous allons décrire l'environnement de travail matériel et logiciel dans lequel nous avons implémenté notre solution.

### 2.1 Environnement matériel

Le développement de notre solution a été fait dans un environnement matériel composé d'un serveur Mac OS Yosemite 10.10 mené d'un serveur de base de données MySQL ainsi qu'un serveur web local Apache ayant la version 5.5.14 de PHP. La gestion des versions a été faite sur une repository Git propre à Logicale et qui est déployée sur un serveur distant. Nous avons utilisé les outils Slack et Asana qui facilitent le travail collaboratif et le suivi du projet.

Nous avons utilisé un serveur web distant pour le déploiement de la solution. Ce serveur possède aussi un SGBD MySQL et opère aussi avec la version 5.5.14 de PHP.

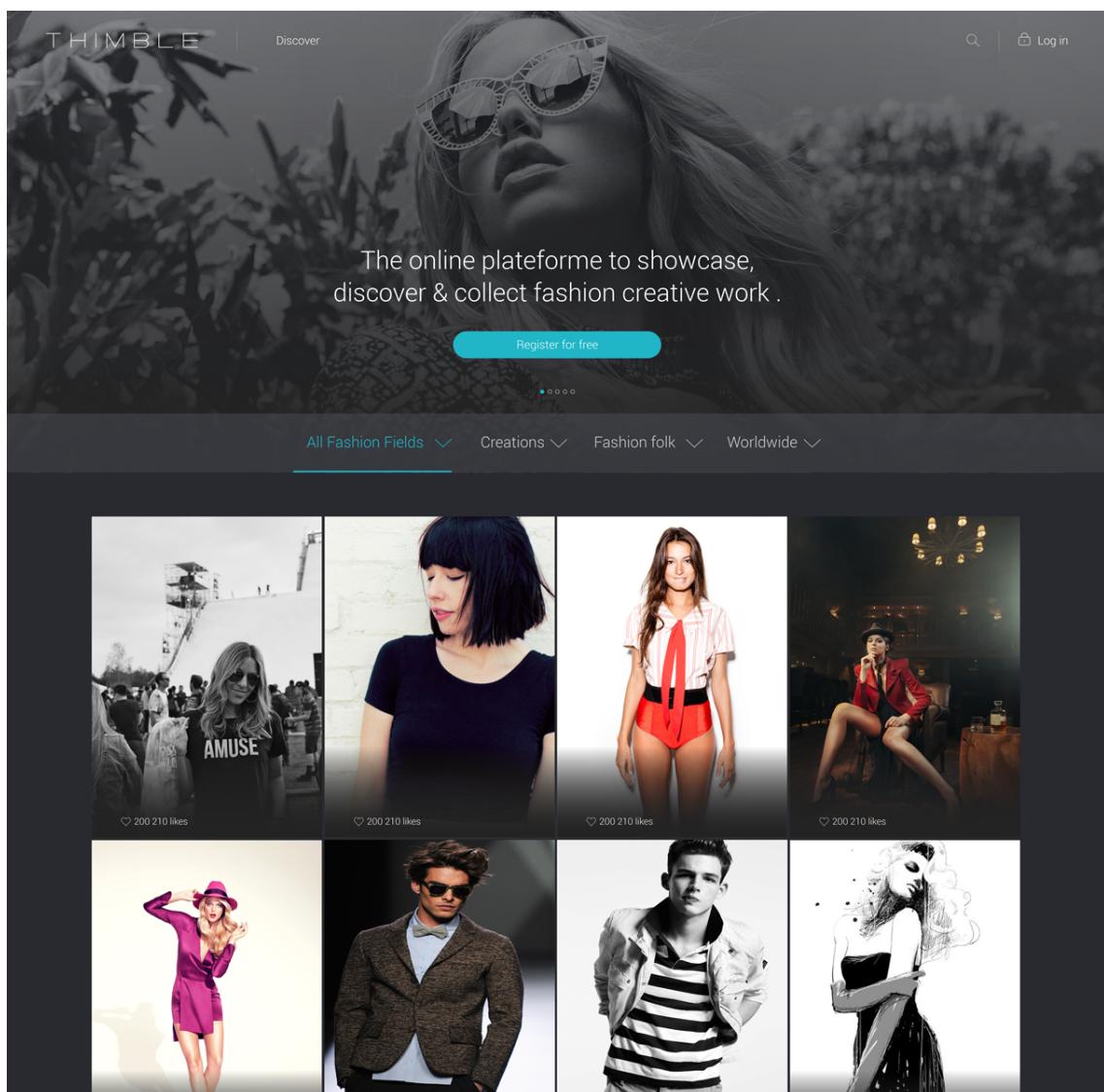
### 2.2 Environnement logiciel

Nous avons eu recours à plusieurs logiciels pour le développement et la conception de notre projet. En effet, La modélisation UML a été faite essentiellement par l'outil Draw.io. Nous avons utilisé l'IDE PhpStorm 8.0.3 pour le développement PHP de l'application Laravel. Pour gérer les versions de notre solution, nous avons utilisé le client du système de gestion de version Git « GitLab ». Les tests unitaires ont été assurés par PHPUnit 4.0. Quant aux tests des web services REST, ils ont été faits avec l'outil SoapUI 5.1.3.

### 3 Mise en œuvre de l'application Front-office

Dans cette partie, nous allons exposer la réalisation de l'application front-office en détaillant l'un des principaux scénarios d'utilisation de cette dernière. Ce scénario décrit l'inscription d'un nouvel utilisateur, sa migration vers un compte professionnel et ensuite la création d'un nouveau projet.

La figure IV.2 présente la page d'accueil de réseau Thimble Fashion. L'utilisateur a la possibilité de découvrir le contenu du réseau sans être obligé de créer un compte. Pour passer à la page d'inscription, il suffit de cliquer sur le bouton "Register for free".



**Figure IV.2** – Page d'accueil de l'application front-office

#### **IV.3 Mise en œuvre de l'application Front-office**

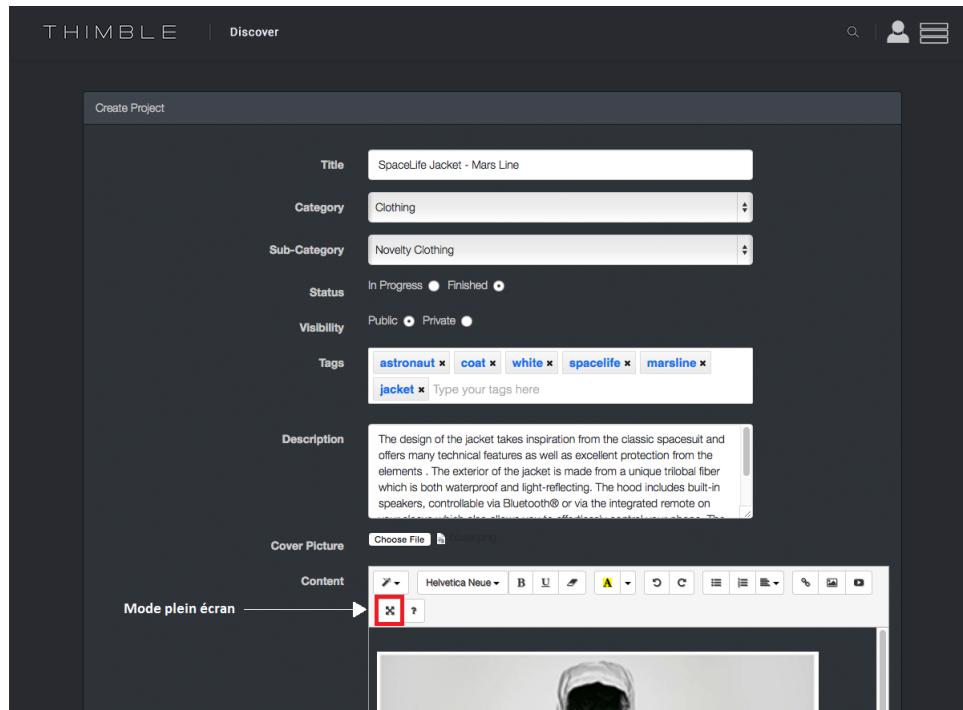
Une fois inscrit, l'utilisateur dispose automatiquement d'un compte amateur. Afin de pouvoir compléter son CV et créer des projets, ce membre doit migrer vers un compte professionnel. La figure IV.3 montre la page d'édition du profil où le membre peut changer le type de son compte.

The screenshot shows the 'Edit Profile' form within the Thimble application. The form is titled 'Edit Profile' at the top left. It contains fields for personal information: First Name (Jessica), Last Name (Halpin), E-mail (halpin.jessica@outlook.com), Birthday (02/07/1990), Gender (Female selected), and social media links for Facebook, Twitter, Google+, and Personal Website. At the bottom of the form is a green button labeled 'Become Professional' with a checkbox next to it, and a grey 'Submit' button below it. The background of the application interface is dark, with a navigation bar at the top featuring the Thimble logo, a search icon, and user profile icons.

**Figure IV.3 –** Page d'édition du profil de l'application front-office

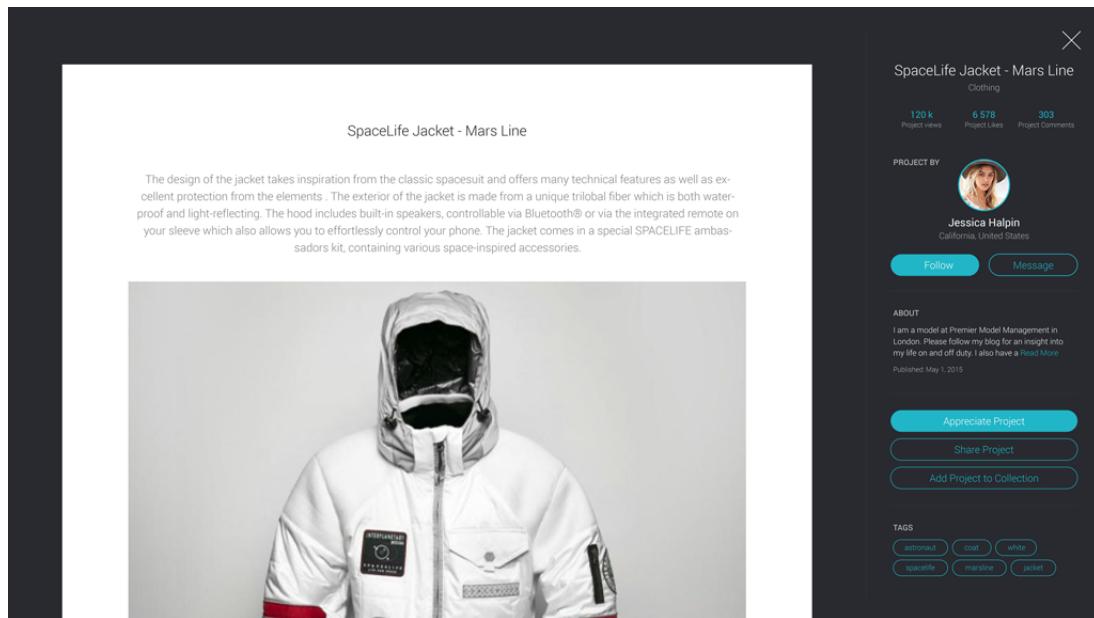
L'utilisateur peut passer maintenant à la création d'un nouveau projet et le publier sur le réseau Thimble Fashion. La figure IV.4 présente la page de création de projet. Il faut noter que le membre peut ouvrir l'éditeur du contenu du projet en mode plein écran ce qui lui donnera plus d'espace pour la rédaction avec une vue plus ergonomique.

### IV.3 Mise en œuvre de l'application Front-office



**Figure IV.4** – Page de création de projets de l'application front-office

La figure IV.5 montre la vue du projet publié.

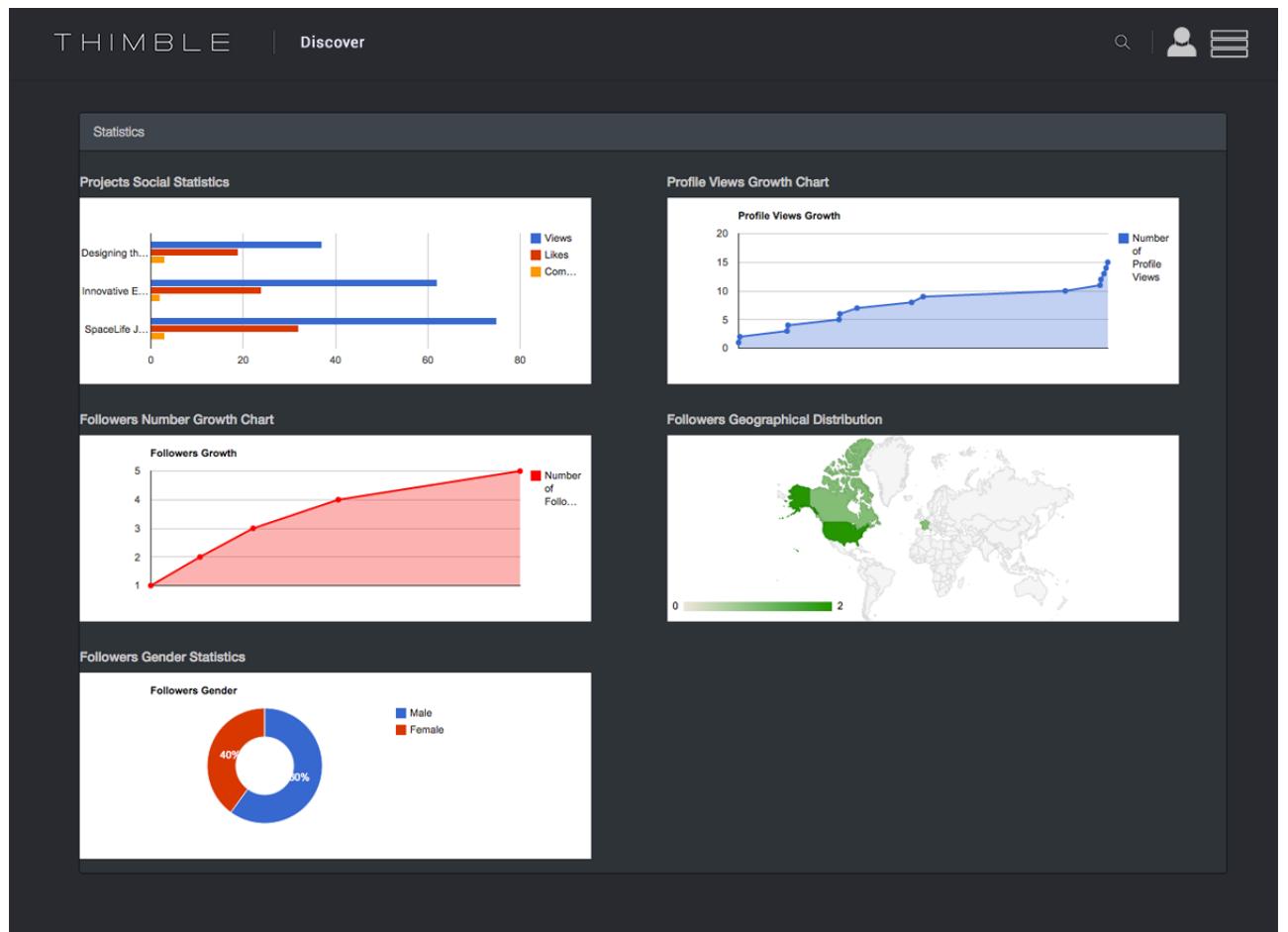


**Figure IV.5** – Vue d'un projet publié sur le réseau Thimble Fashion

Finalement, ce membre professionnel peut suivre les statistiques de son profil, ses projets et ses followers dans la page "My Statistics". Cette page est exposée dans la figure

**IV.6** et elle est composée de cinq graphes :

- Diagrammes en bâtons superposés des statistiques sociales des projets (Likes, Views et Comments).
- Courbe d'évolution du nombre de vue du profil.
- Courbe d'évolution du nombre de followers.
- Mappe de distribution géographique des followers.
- Diagramme en camembert du sexe des followers.



**Figure IV.6** – Page de statistiques d'un membre du réseau Thimble Fashion

Pour obtenir l'ensemble des graphes et statistiques présentés par la figure **IV.6**, nous avons intégré le module PHP Lavacharts qui une library permettant d'éviter de coder manuellement la construction des graphs avec JavaScript. Cette library est basée sur l'API Google Chart.

## 4 Mise en œuvre de l'API REST

Pour exposer la réalisation de notre API de web services, nous allons présenter dans cette partie des exemples de web services mis en place, la sécurisation de ces web services ainsi que leur documentation.

### 4.1 Exemples de web services REST réalisés

Dans ce qui suit, nous allons présenter trois exemples de contrats de web services REST implémentés et leurs descriptions.

#### Inscription Utilisateur

User Store Method {POST}	
<a href="http://thimblefashion.com/api/public/api/v1/users">http ://thimblefashion.com/api/public/api/v1/users</a>	
Header	
X-Authorization	Clé d'autorisation d'accès aux web services
Input	
firstName	Nom de l'utilisateur
lastName	Prénom de l'utilisateur
email	L'adresse email de l'utilisateur
birthday	La date de naissance
password	Le mot de passe
gender	Le sexe de l'utilisateur
registeredVia	La méthode d'inscription (iOS ou Android)
facebookLink (optionnel)	Le lien du profile Facebook
twitterLink (optionnel)	Le lien du profile Twitter
googlePlusLink (optionnel)	Le lien du profile Google+
personalWebsite (optionnel)	Le lien du site web personnel
Output	
message	Message indiquant le résultat de l'opération de création ainsi que le résultat de l'envoi de l'email de confirmation
object	Contient l'instance de l'utilisateur créé sous format d'un objet JSON
statusCode	Code de statut de la réponse HTTP

Tableau IV.1 – Contrat du web service d'inscription utilisateur

Les requêtes reçues par le web service défini par le contrat du tableau IV.1 font l'objet d'une vérification de l'autorisation d'accès. Cette vérification se base sur l'entête « X-Authorization » de la requête et elle va être détaillée dans la partie suivante. Ce service web permet la création d'un nouvel utilisateur à partir des informations fournis dans la requête POST. Le service retourne une réponse HTTP ayant le code 422 indiquant une éventuelle erreur de validation. Si l'opération de création a réussi, ce service web envoi un email de confirmation à l'adresse email du nouveau utilisateur et retourne un réponse HTTP avec un code de succès 201. Cette réponse contient un message informatif et l'objet JSON de l'utilisateur créé.

### Découverte des projets

Project Index Method {GET}	
http ://thimblefashion.com/api/public/api/v1/projects	
Header	
X-Authorization	Clé d'autorisation d'accès aux web services
Input	
limit	Nombre de projets par page souhaité
sort	L'attribut à utiliser pour le tri des projets
order	L'ordre de tri ascendant ou descendant
visibility	La visibilité des projets à récupérer (publique ou privé)
status	Le statut des projets à récupérer (Fini ou en cours)
categoryId	Pour filtrer les projets selon la catégorie
subCategoryId	Pour filtrer les projets selon la sous-catégorie
search	Pour chercher les projets ayant ce mot clé dans leurs titres, leurs descriptions ou leurs tags.
embed	Pour effectuer du EagerLoading, ce paramètre permet au consommateur d'indiquer qu'il veut récupérer aussi les entités en relations avec un projet telles que son créateur, sa photo de couverture ou l'ensemble de ses tags par exemple. Ces entités vont être incorporées dans le projet.
Output	
data	C'est un tableau contenant un ensemble de projets sous format d'objets JSON.
paginator	C'est un objet JSON contenant l'ensemble des informations nécessaire pour la pagination.
statusCode	Code de statut de la réponse HTTP

**Tableau IV.2** – Contrat du web service de découverte les projets

Le service web décrit par le contrat du tableau IV.2 permet la découverte des projets. Il est assez flexible proposant une variété de paramètres permettant aux consommateurs de filtrer les projets et raffiner leur recherche. Le consommateur peut également effectuer du EagerLoading s'il en a besoin. Il doit juste indiquer l'ensemble des entités en relations avec un projet dans le paramètre « embed ». Le service va se charger d'imbriquer ces données supplémentaires dans les projets retournés.

### Récupération des Images

Image GetByObjectId Method {GET}	
http://thimblefashion.com/api/public/api/v1/images/getByObjectId/ :objectId	
Header	
X-Authorization	Clé d'autorisation d'accès aux web services
Input	
relatedObject_type	Le type de l'objet spécifié par l'identifiant « objectId » et auquel appartient l'image
type	Le type de l'image (photo de profile, photo de couverture, logo)
Output	
data	C'est un tableau contenant une ou plusieurs images sous format d'objets JSON.
statusCode	Code de statut de la réponse HTTP

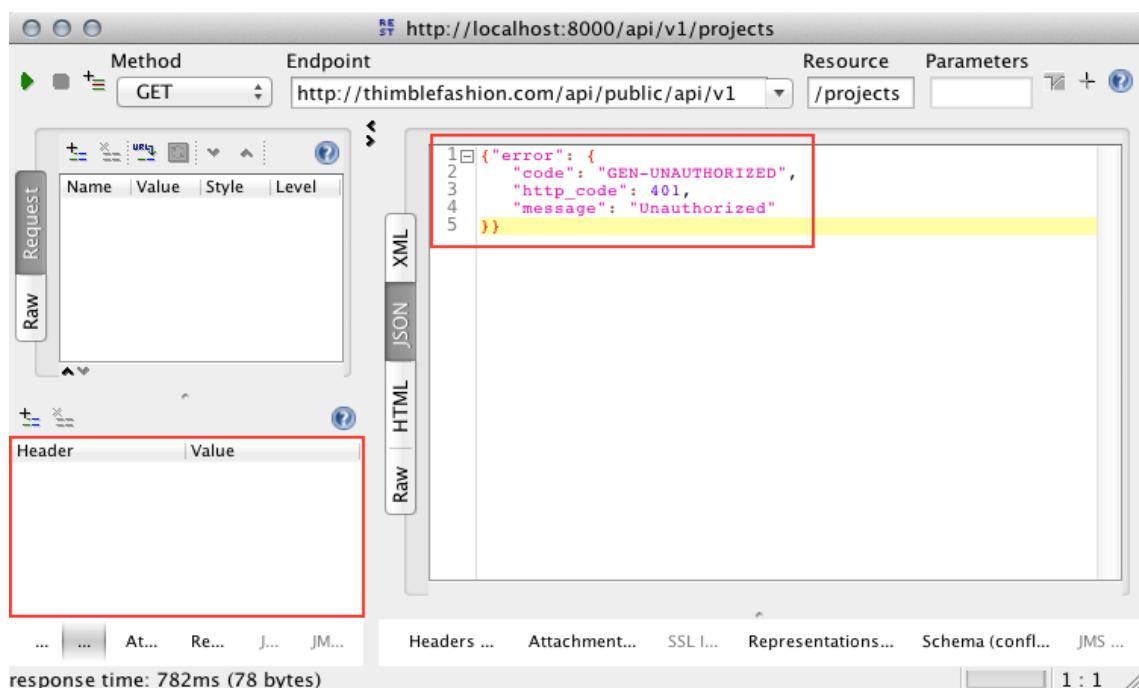
**Tableau IV.3** – Contrat du web service de récupération des Images

Le web service défini par le contrat du tableau IV.3 permet la récupération des images d'un utilisateur, d'un projet, d'une collection ou d'un événement. En effet, le consommateur indique l'identifiant de l'objet en question dans l'URI (Fragment Parameter). Il doit rajouter aussi le type de cet objet ainsi que le type des images à récupérer comme des paramètres URL. Le web services retourne l'ensemble des images correspondantes dans un tableau « data » d'objets JSON.

## 4.2 Sécurisation de l'API

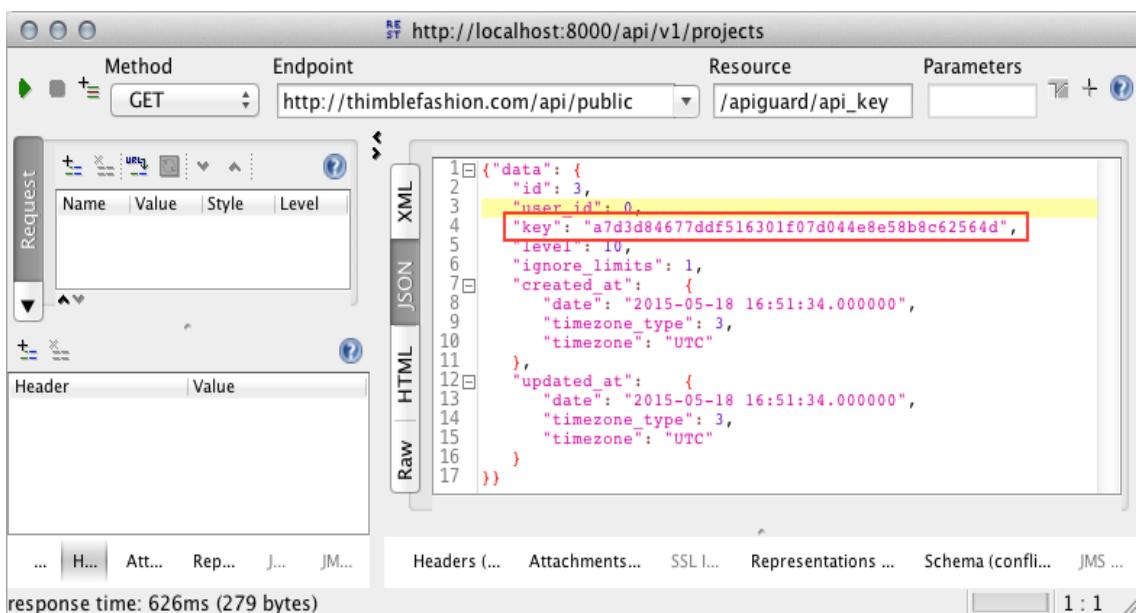
Afin de restreindre l'accès à l'API REST qu'aux applications mobiles et l'application du back-office, nous avons utilisé l'outil API-Guard. C'est un module PHP qui

s'intègre à l'application Laravel afin de gérer l'accès aux web services RESTFul via un système de génération et de vérification des clés d'autorisation. Ce module permet aussi de configurer le taux maximum de consultation d'un web service donné par clé d'autorisation. Il offre également la possibilité aussi de journaliser les requêtes REST reçues dans la base de données. Dans ce qui suit, nous allons présenter un exemple illustrant le mécanisme d'autorisation. Premièrement, la figure IV.7 montre une simulation d'un appel rejeté par le web service de récupération des projets avec un code d'erreur 401 faute d'autorisation. En effet, la requête HTTP GET envoyée ne contient pas l'entête « X-Authorization » qui doit avoir comme valeur la clé d'accès à notre API.



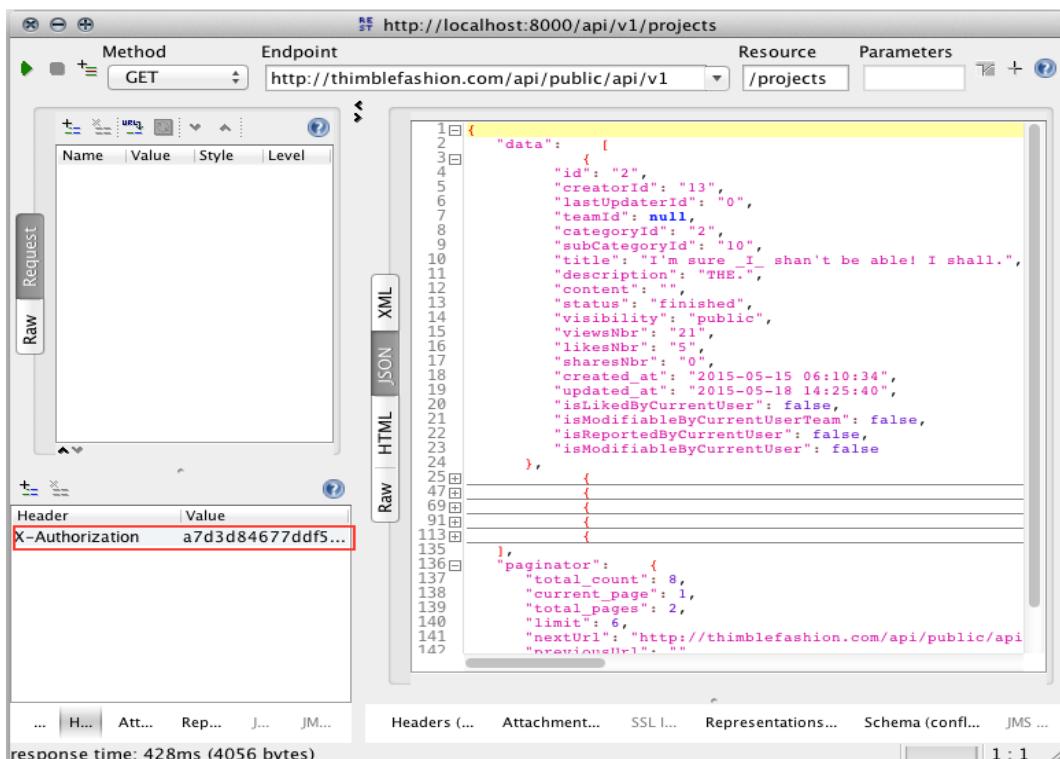
**Figure IV.7 – Interface d'appel rejeté par l'API**

Dans la figure IV.8, nous avons simulé une demande d'autorisation d'accès auprès du web service dédié. Nous allons récupérer et conserver la clé d'accès contenue dans la réponse JSON pour les prochains appels. Il faut noter que ce web service de génération des clés d'accès va être désactivé dès que toutes les applications consommant l'API ont eu leurs propres clés.



**Figure IV.8 – Demande d'autorisation API REST**

Par la suite, la simulation d'appel du web service de récupération des projets a réussi puisqu'on a rajouté la clé d'autorisation qu'on vient d'obtenir dans l'entête « X-Autorisation ».



**Figure IV.9 – Appel réussi du service de récupération des projets**

La figure IV.9 montre la réponse HTTP obtenue avec un code de succès 200 contenant l'ensemble des projets paginés sous format JSON.

### 4.3 Documentation de l'API

Afin de faciliter la compréhension et l'utilisation de notre API REST, nous avons utilisé l'outil « APIDoc » qui est une librairie PHP permettant de générer une documentation claire et complète d'une API RESTful. Après l'intégration et la configuration de l'outil dans notre solution, ce dernier se base sur les blocs de documentation que nous insérons dans les contrôleurs de l'API. La documentation générée à la fin est sous forme de Wiki html décrivant tout les web services, leurs contrats, les types d'erreurs qu'ils peuvent générer ainsi que quelques exemples des réponses JSON qu'ils retournent. L'annexe 3 présente un extrait de la documentation obtenue.

## 5 Tests de la solution

Nous allons dédier cette partie aux tests réalisés sur notre solution. Ces tests permettent de valider les exigences fonctionnelles et non-fonctionnelles citées dans le chapitre 2 d'analyse et de modélisation des besoins.

### 5.1 Tests unitaires

Les tests unitaires ont pour objectif d'assurer que des unités individuelles de code fonctionnent convenablement lorsqu'elles sont utilisées de manière isolée. Nous avons procédé par exemple aux tests unitaires de la couche d'accès aux données gérée par le package Repository. Nous avons commencé par cette couche puisqu'elle est la couche la plus primordiale et son bon fonctionnement est indispensable pour les autres couches.

Nous avons choisi le framework de test PHP « PHPUnit » pour rédiger et exécuter les tests unitaires. C'est un framework open-source qui offre [26] :

- Une syntaxe simple et facile à retenir.
- Un grand nombre de méthodes de test.
- Organisation et exécution flexibles des tests.
- Un utilitaire en ligne de commande complet.

En effet, nous avons rédigé un ensemble de cas de test (Test case) pour les classes Repository. À titre d'exemple, nous avons testé la classe « ProjectRepository » et plus

précisément la méthode de récupération d'un projet à partir de son Identifiant « find ». Dans ce test, nous avons commencé par créer un projet fictif à l'aide de la fabrique « StubFactory » et nous l'avons inséré dans la base de test. Ensuite, nous avons testé si la méthode « find » parvient à récupérer le projet qu'on a inséré par son identifiant.

## 5.2 Tests de l'API REST

Nous allons nous concentrer dans cette partie sur les tests effectuées sur les web services de notre API REST. Nous allons présenter trois types de tests qui ont été réalisés avec l'outil SoapUI : Tests Fonctionnels, Tests de Performance et Tests de Charge.

### 5.2.1 Outil SoapUI

SoapUI qui est un outil JAVA open source et cross-plateforme de test de web services. Il offre la possibilité de créer et exécuter automatiquement différents types de tests à travers une interface graphique simple à manipuler. SoapUI est doté d'une panoplie de fonctionnalités à savoir [27] :

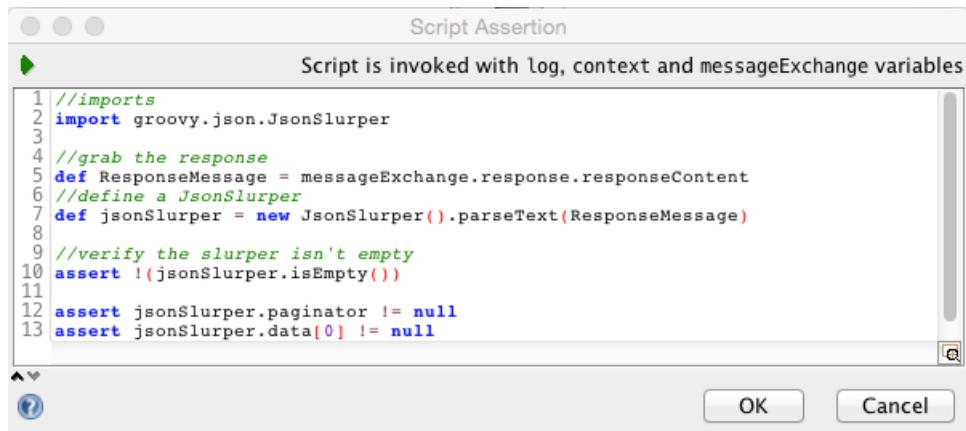
- Les tests unitaires.
- Les tests d'intégration.
- Les tests automatisés.
- Les tests de performance et de charge

Un autre avantage de cet outil est qu'il est compatible avec différents protocoles et technologies standards. Soap UI supporte le style d'architecture REST et il nous a permis de simuler la consommation de nos web services RESTful et d'effectuer différents types de tests qu'on va détailler dans les points suivants.

### 5.2.2 Tests fonctionnels

Les tests fonctionnels vérifient l'intégration de composants appartenant aux différentes couches de l'application en partant de la couche d'accès aux données jusqu'à la couche présentation. Les tests fonctionnels peuvent être assimilés aux tests manuels qu'on peut effectuer à travers le navigateur web. Les tests fonctionnels sont employés afin d'automatiser ce type de tests manuels. Nous avons procédé aux tests fonctionnels de l'API de web services en utilisant l'outil SoapUI. Dans ce qui suit nous allons illustrer ces tests avec l'exemple de récupération l'ensemble des évènements paginés.

Nous avons commencé par la création de la requête REST GET faisant appel au service web de récupération des évènements. Ensuite, nous avons créé un cas de test fonctionnel et nous l'avons associé à la requête qu'on vient de créer. Nous avons ajouté tout d'abord une première assertion de type « Valid HTTP Status Codes » vérifiant que le code de statut HTTP de la réponse soit un code de succès 200. Puis, nous avons inséré une deuxième assertion de type « Script » qui vérifie que la réponse JSON retournée contient les éléments « data » (un tableau contenant au moins un objet) et « paginator » (réponse paginée). Cette assertion est écrite en langage Groovy et elle est présentée dans la figure IV.10 .



```

1 //imports
2 import groovy.json.JsonSlurper
3
4 //grab the response
5 def ResponseMessage = messageExchange.response.responseContent
6 //define a JsonSlurper
7 def jsonSlurper = new JsonSlurper().parseText(ResponseMessage)
8
9 //verify the slurper isn't empty
10 assert !(jsonSlurper.isEmpty())
11
12 assert jsonSlurper.paginator != null
13 assert jsonSlurper.data[0] != null

```

**Figure IV.10** – Assertion de type Script

Finalement, nous avons exécuté ce cas de test fonctionnel et la figure IV.11 montre l'exemple d'un test validé.



Figure IV.11 – Test Fonctionnel List Events Valide

### 5.2.3 Tests de performance

Pour tester la performance des web services en termes de temps de taille et de données des réponses, nous avons créé une requête GET faisant appel au service web dédié à la recherche d'une ressource de type projet par identifiant. L'identifiant recherché doit être inclus en tant qu'un paramètre fragment dans l'URI de la requête. La figure

IV.12 présente la réussite d'une requête ayant comme réponse le projet correspondant à l'identifiant spécifié. Le temps de réponse de cette dernière est de 348ms et elle contient 669 bytes de données.

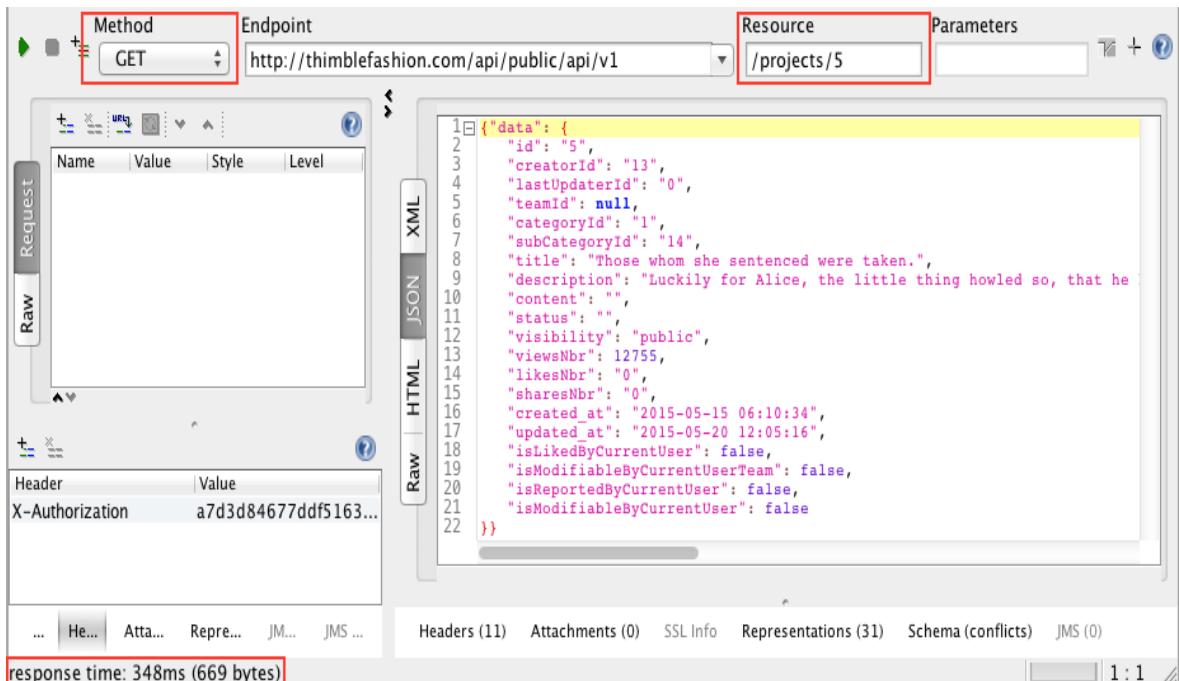


Figure IV.12 – Appel réussi du service de recherche de projets par identifiant

Nous lançons ensuite la même requête précédente mais avec un identifiant inexistant dans la base de données. Nous obtenons ainsi une réponse d'erreur (présentée dans la figure IV.13) de la part du service web en question ayant comme temps de réponse 567 ms et 31 bytes en données. Nous remarquons qu'une exception coute considérablement plus de temps (x 1,6) qu'une réponse valide. La taille de données réduite de la réponse erronée s'explique par son contenu limité à un seul message d'erreur.

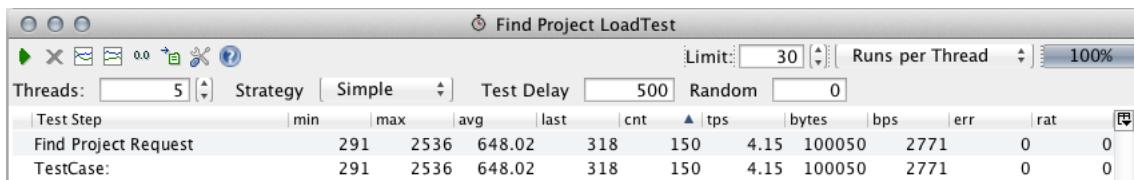


Figure IV.13 – Réponse erronée du service de recherche de projets par identifiant

### 5.2.4 Tests de charge

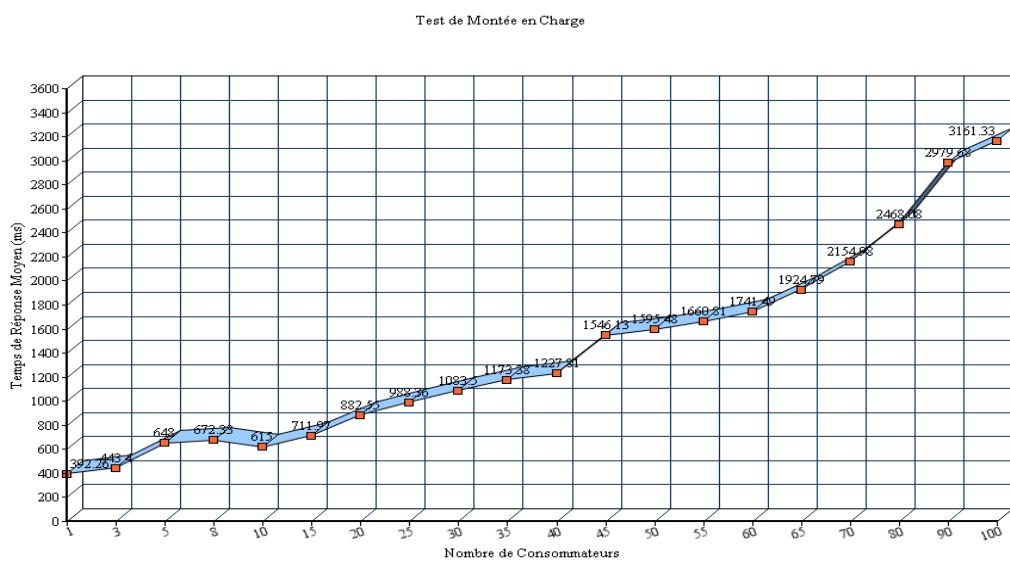
Nous allons nous concentrer dans ce paragraphe sur le test de la monté en charge de notre API REST face à différents nombres de consommateurs concurrents. Les tests de charge (Load Tests) ont été effectués avec SoapUI qui permet de varier le nombre de threads (consommateurs) concurrents et la stratégie de déclenchement de ces derniers.

A titre d'exemple, nous avons créé un Load Test pour le service de recherche de projet par identifiant. Ensuite, nous avons choisi la stratégie de test simple qui permet de déclencher les threads successivement avec une marge de délai qui peut être fixe (Random = 0) ou aléatoire. Dans notre cas, nous avons mis un délai fixe de 500 ms. Nous avons limité aussi le nombre d'exécution par thread à 30. La figure IV.14 montre l'interface de configuration de notre test de charge.



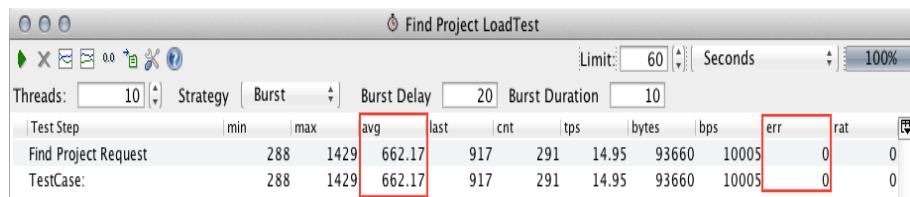
**Figure IV.14** – Interface de configuration du test de charge

Après avoir fini la configuration, nous avons exécuté ce Load Test une vingtaine de fois en variant le nombre de threads et en notant le temps de réponse moyen du web service. Nous avons obtenu ainsi un tableau de valeurs nous permettant de tracer la courbe de monté en charge du web service en question présentée dans la figure IV.15.



**Figure IV.15** – Courbe de montée en charge du service de recherche de projet

Nous avons repris ensuite le test avec dix threads mais en utilisant la stratégie Burst (Rafale) cette fois. Cette stratégie conçue spécialement pour les tests de récupération (Recovery Tests). L'idée de cette stratégie est qu'elle ne déclenche aucun thread pendant un délai bien déterminé. Une fois ce délai se termine, elle lance un nombre de threads prédéfini simultanément afin qu'ils « attaquent » le web services avec un nombre important de requêtes. La durée de l'attaque de trafic est désignée par le testeur. A la fin de l'exécution, un test de charge est déclenché afin de mesurer le temps de récupération du service web. La figure IV.16 expose le résultat obtenu par ce test.



**Figure IV.16** – Résultat du Test Load avec la stratégie Rafale

Nous pouvons constater à travers les mesures obtenues dans la figure IV.16 que le web service a bien passé ce test sans retourner des erreurs. Son temps de récupération est raisonnable puisque le temps de réponse moyen obtenu est de 662.17 ms, soit 7.6% de plus que le temps moyen dans les conditions normales (Stratégie Simple).

## Conclusion

Dans ce chapitre, nous avons exposé en premier lieu l'architecture finale de notre solution. Ensuite, nous avons décrit l'environnement de travail logiciel et matériel du projet. Nous avons passé par la suite à la description de la mise en œuvre des composants de notre architecture et plus précisément l'application front-office et son API de web services. Dans la dernière partie, nous nous sommes concentrés sur la phase de test de notre solution en exposant des exemples de tests qu'on a effectué et leurs résultats. Ces résultats ont été conformes aux attentes de notre Product Owner.

---

# Conclusion Générale et Perspectives

Ayant identifié l'absence d'un véritable espace social et professionnel en ligne dédié à une industrie si importante et si gigantesque que l'industrie de la mode, la startup Logicale a voulu profiter de cette opportunité et combler ce besoin en mettant en œuvre un grand projet intitulé Thimble Fashion. L'objectif général de ce projet est d'exploiter les technologies de pointe afin d'aboutir à une plateforme proposant de nouveaux services aux différents acteurs de l'industrie de la Mode. Notre projet de fin d'études fait partie de la première phase de Thimble Fashion et notre mission était de concevoir, réaliser et tester un réseau social professionnel web adapté au domaine de la Mode ainsi que l'API de web services qui va être utilisée pour le développement des applications mobiles et du back-office.

Au terme de ce rapport, nous résumons les grandes lignes de ce travail. Nous avons essayé de décrire tout ce qui s'avère indispensable pour donner une idée claire sur les étapes de ce projet de fin d'étude. Nous avons commencé par présenter une étude préliminaire autour de l'industrie de la Mode et des réseaux sociaux professionnels. Cette étude comprend la description de quelques solutions web dédié à la Mode et les services qu'elles proposent. Nous avons enchainé avec l'exposition du cadre générale du projet et la méthodologie de travail employée. Puis, nous avons spécifié et modélisé les besoins fonctionnels et non-fonctionnels auxquels devra répondre notre solution. Nous avons présenté ensuite l'architecture cible ainsi que l'étude technique et conceptuelle de cette solution. Finalement, nous avons dédié le dernier chapitre à la description de la mise en œuvre de l'application web Laravel du réseau et de l'API RESTful. Il faut noter qu'on a rencontré des problèmes de disfonctionnement au niveau du framework Laravel vue que sa cinquième et dernière version sortie en Février 2015 n'était pas encore complètement stable au moment où nous avons commencé le développement. Nous avons clôturé ce chapitre avec l'ensemble des cas de test effectués et leurs résultats.

Pour conclure, nous estimons avoir satisfait les objectifs initialement fixés, néanmoins, la solution réalisée reste ouverte à plusieurs apports et extensions. Tout d'abord, l'application peut être enrichie avec l'ajout d'un nouveau type de compte pour les entreprises leur permettant ainsi d'avoir une page de présentation et de gérer eux même leurs offres de travail. Il serait judicieux aussi d'améliorer les fonctionnalités de recherche et filtrage du contenu et plus précisément la recherche des membres professionnels en ajoutant par exemple l'option de recherche par niveau d'expérience professionnelle ou niveau académique.

---

## Références

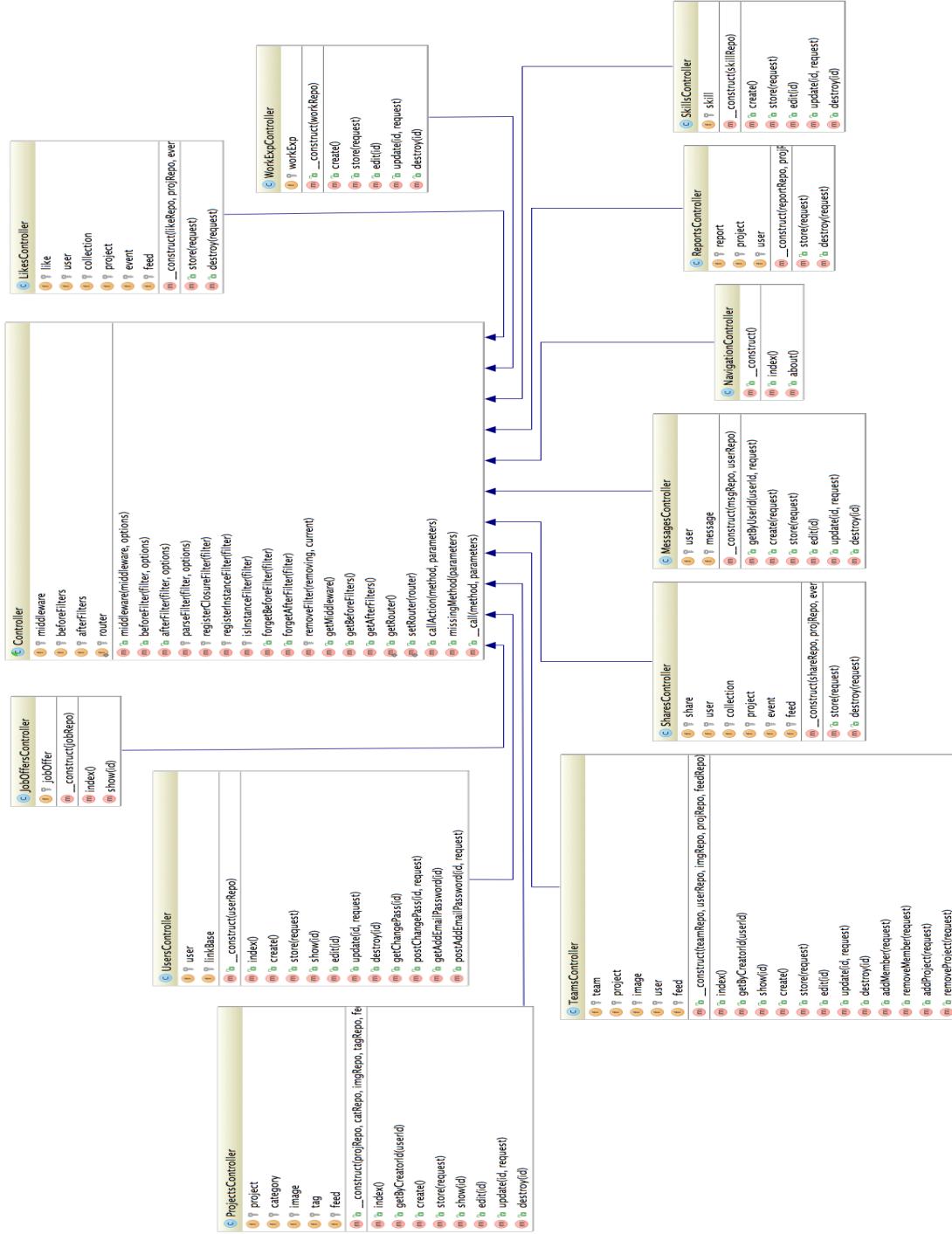
- [1] V. STEELE. Fashion Industry. <http://www.britannica.com/EBchecked/topic/1706624/fashion-industry>. [En ligne ; consulté le 05-Avril-2015]. 1, 4
- [2] J. LEFILLIATRE. Uber la start-up qui veut révolutionner le transport en ville, à Paris comme à San Francisco. <http://www.challenges.fr/high-tech/20131210.CHA8145/uber.html>. [En ligne ; consulté le 07-Avril-2015]. 1
- [3] Logicale. <http://logicale.ca>. [En ligne ; consulté le 07-Avril-2015]. iv, 3
- [4] Les métiers : Mode - textile. , <http://www.studyrama.com/formations/fiches-metiers/Mode-textile>. [En ligne ; consulté le 08-Avril-2015]. 4
- [5] N. B.ELLISON D. M.BOYD. *Social Network Sites : Definition, History, and Scholarship*. University of California-Berkeley/Michigan State University. 1,2p. 6
- [6] R. DUBE. Characteristics of Social Networks. [http://socialnetworking.lovetoknow.com/Characteristics\\_of\\_Social\\_Networks](http://socialnetworking.lovetoknow.com/Characteristics_of_Social_Networks). [En ligne ; consulté le 10-Avril-2015]. 6
- [7] Réseaux sociaux professionnels : Introduction. <http://www.job2-0.com/article-les-50-reseaux-sociaux-pros-a-connaître-106828611.html>. [En ligne ; consulté le 10-Avril-2015]. 7
- [8] About LinkedIn. <https://press.linkedin.com/about-linkedin>. [En ligne ; consulté le 10-Avril-2015]. 7
- [9] M. STÉPHANIE. Crétifs, réalisez votre portfolio en ligne avec Behance. <http://blog.digitives.com/2012/05/creatifs-realisez-votre-portfolio-en.html>. [En ligne ; consulté le 10-Avril-2015]. 8
- [10] FashionUnited. <https://fashionunited.info/about-us>. [En ligne ; consulté le 10-Avril-2015]. iv, 8
- [11] Fashionista. <http://www.fashionista.com/page/about>. [En ligne ; consulté le 10-Avril-2015]. iv, 9
- [12] Manymucho. <http://www.manymucho.com/presentation>. [En ligne ; consulté le 10-Avril-2015]. iv, 9, 10
- [13] Methodes Agiles. [http://web-serv.univ-angers.fr/docs/etudquassi/Methodes\\_agiles.pdf](http://web-serv.univ-angers.fr/docs/etudquassi/Methodes_agiles.pdf). [En ligne ; consulté le 05-Avril-2015]. 12

- 
- [14] C. LARMAN B.VODDE P. DEEMER, G. BENEFIELD. *Scrum Primer, version 2.0*. Traduction : Cyril Stock(2012) p 1,2. iv, 13, 14
  - [15] J. EHRET. Qu'est ce qu'un Framework. <http://www.elephorm.com/java-spring/framework>. [En ligne ; consulté le 11-Avril-2015]. 29
  - [16] <http://www.inmediaveritas.com/comparatif-framework-php>. [En ligne ; consulté le 11-Avril-2015]. 29
  - [17] <http://laravel.com>. [En ligne ; consulté le 14-Avril-2015]. iv, 30, 31
  - [18] A. JAIN. An Introduction about Laravel Framework. <http://mrbool.com/an-introduction-about-laravel-framework/26410>. [En ligne ; consulté le 15-Avril-2015]. 31
  - [19] E. HEWITT. *Java SOA Cookbook*. O'Reilly, California (2009). 740p. 32
  - [20] S. RUBY L. RICHARDSON. *RESTful Web Services*. O'Reilly, California (2007). 448p. 32
  - [21] C. MANOLE. Services Web : REST face à SOAP. <http://blog.beleringenierie.com/2011/10/services-web-rest-face-a-soap>. [En ligne ; consulté le 17-Avril-2015]. 34, 35
  - [22] Le protocole SOAP. [http://igm.univ-mlv.fr/~dr/XPOSE2003/axis\\_seng/soap.html](http://igm.univ-mlv.fr/~dr/XPOSE2003/axis_seng/soap.html). [En ligne ; consulté le 17-Avril-2015]. 34
  - [23] [http://www.programmableweb.com/category/all/news?search\\_id=136478&articletypes=analysis](http://www.programmableweb.com/category/all/news?search_id=136478&articletypes=analysis). [En ligne ; consulté le 18-Avril-2015]. iv, 36
  - [24] The Repository Pattern. <https://msdn.microsoft.com/en-us/library/ff649690.aspx>. [En ligne ; consulté le 25-Avril-2015]. 47
  - [25] Factory Method. <https://www.binpress.com/tutorial/the-factory-design-pattern-explained-by-example/142>. [En ligne ; consulté le 26-Avril-2015]. 49
  - [26] PHPUnit. <http://jp-grossglauser.developpez.com/tutoriels/langages/php/phpunit>. [En ligne ; consulté le 10-Mai-2015]. 62
  - [27] What is SoapUI. <http://www.soapui.org/about-soapui/what-is-soapui-.html>. [En ligne ; consulté le 15-Mai-2015]. 63

## Annexe 1

## Diagramme de classes du package : Web Controllers

Par souci de clarté, nous avons divisé le diagramme de classes du package Web Controllers en deux parties ( [A.1](#) et [A.2](#) )



**Figure A.1** – Diagramme de Classes : Package Web Controllers Partie 1

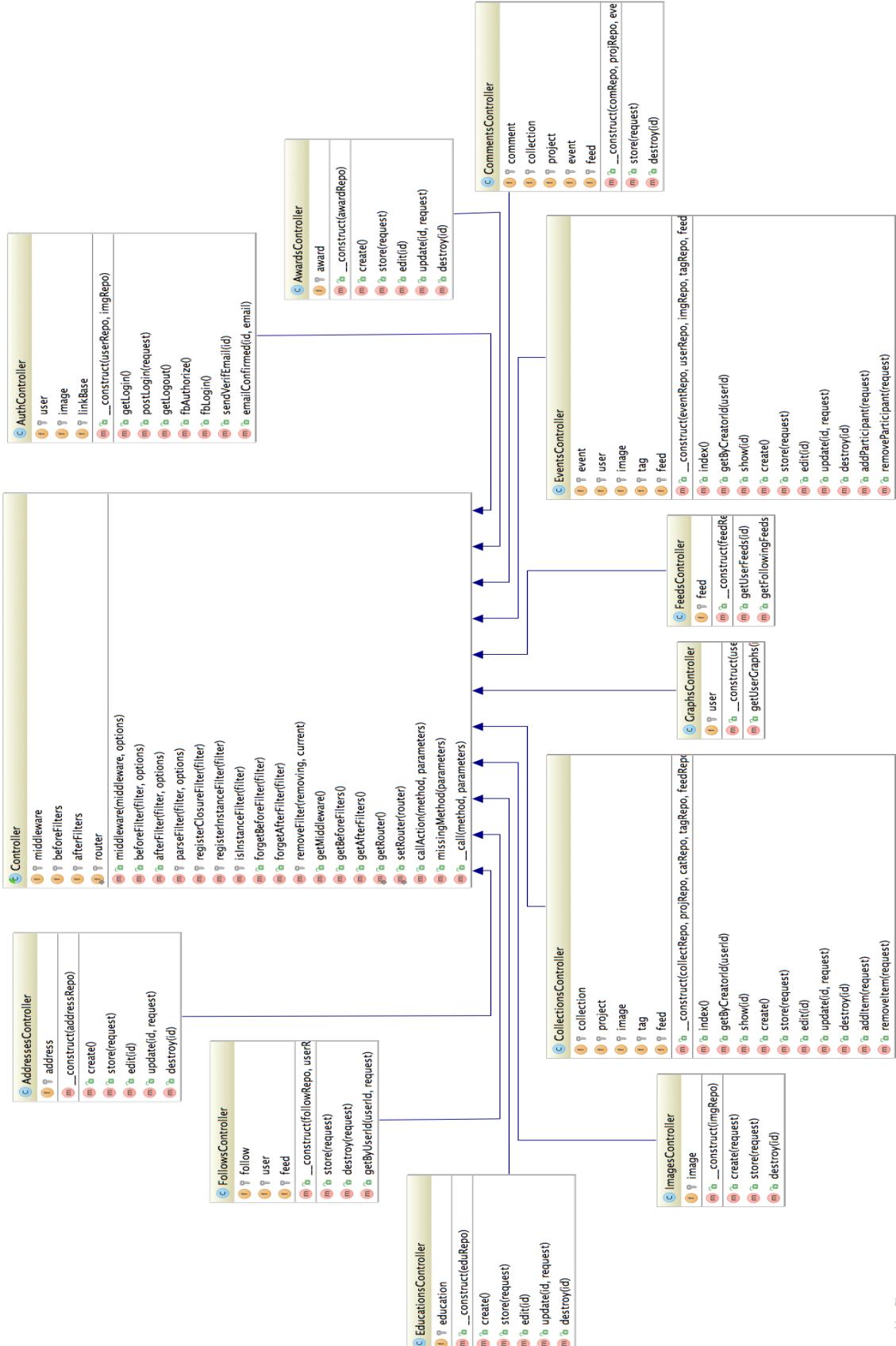


Figure A.2 – Diagramme de Classes : Package Web Controllers Partie 2

# Diagramme de classes du package : Repositories

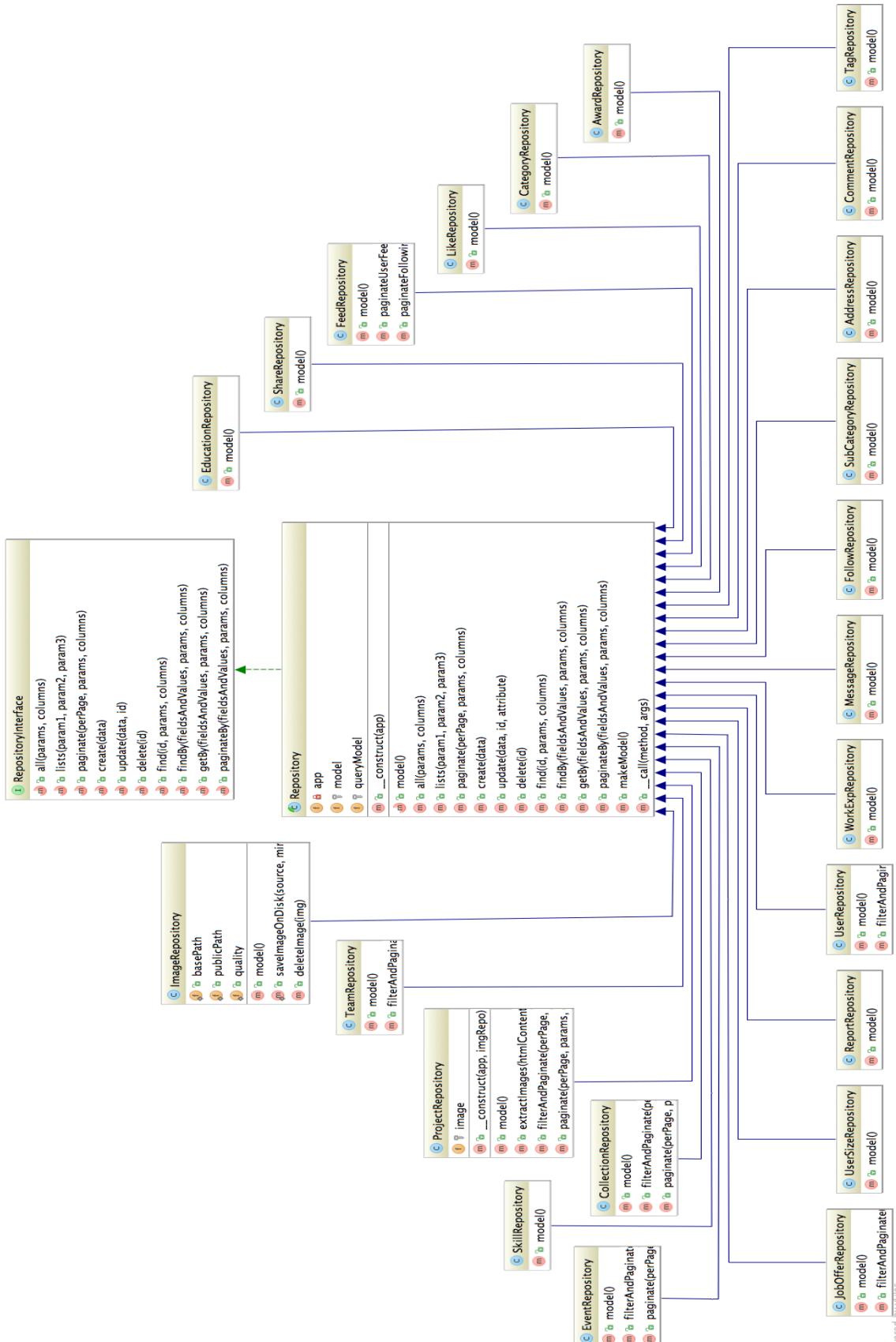


Figure A.3 – Diagramme de Classes : Package Repositories

## Diagramme de classes du package : Requests

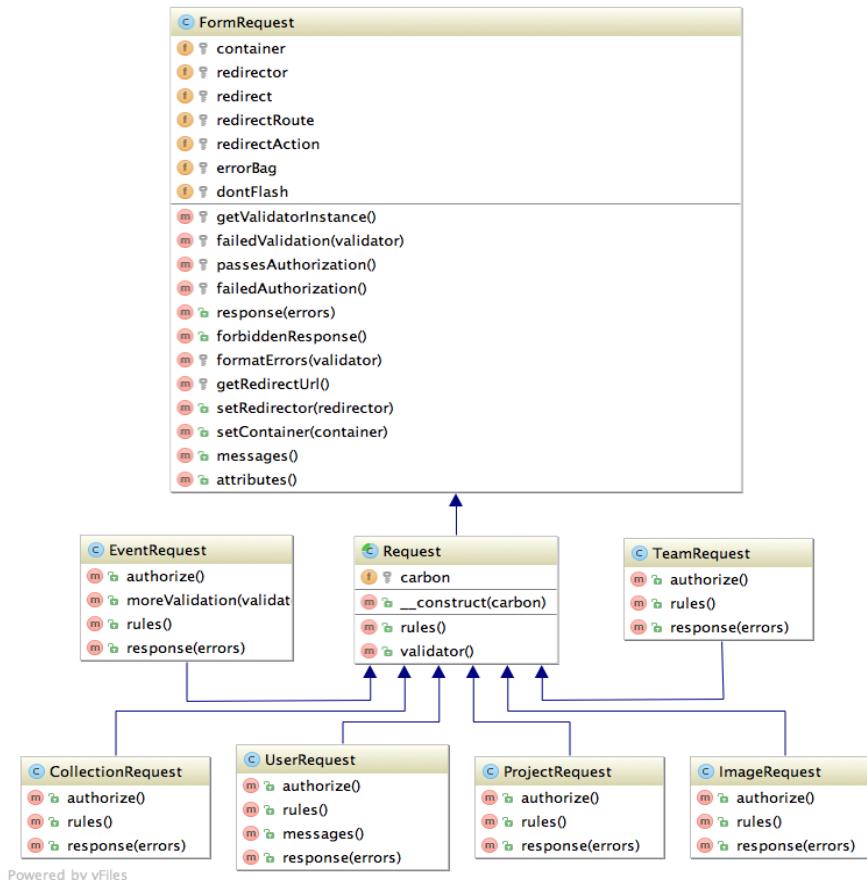


Figure A.4 – Diagramme de Classes : Package Requests

## Diagramme de classes du package : Middlewares

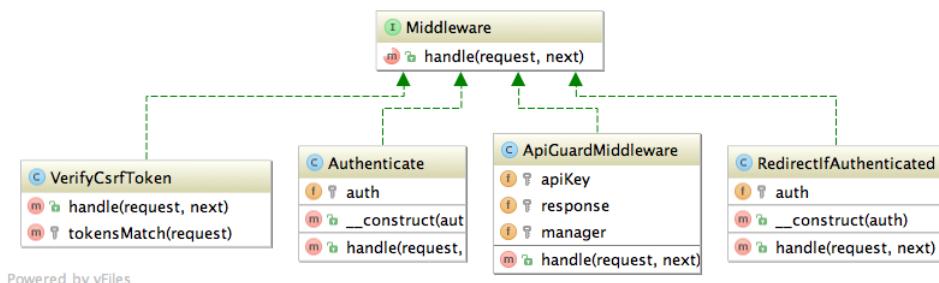
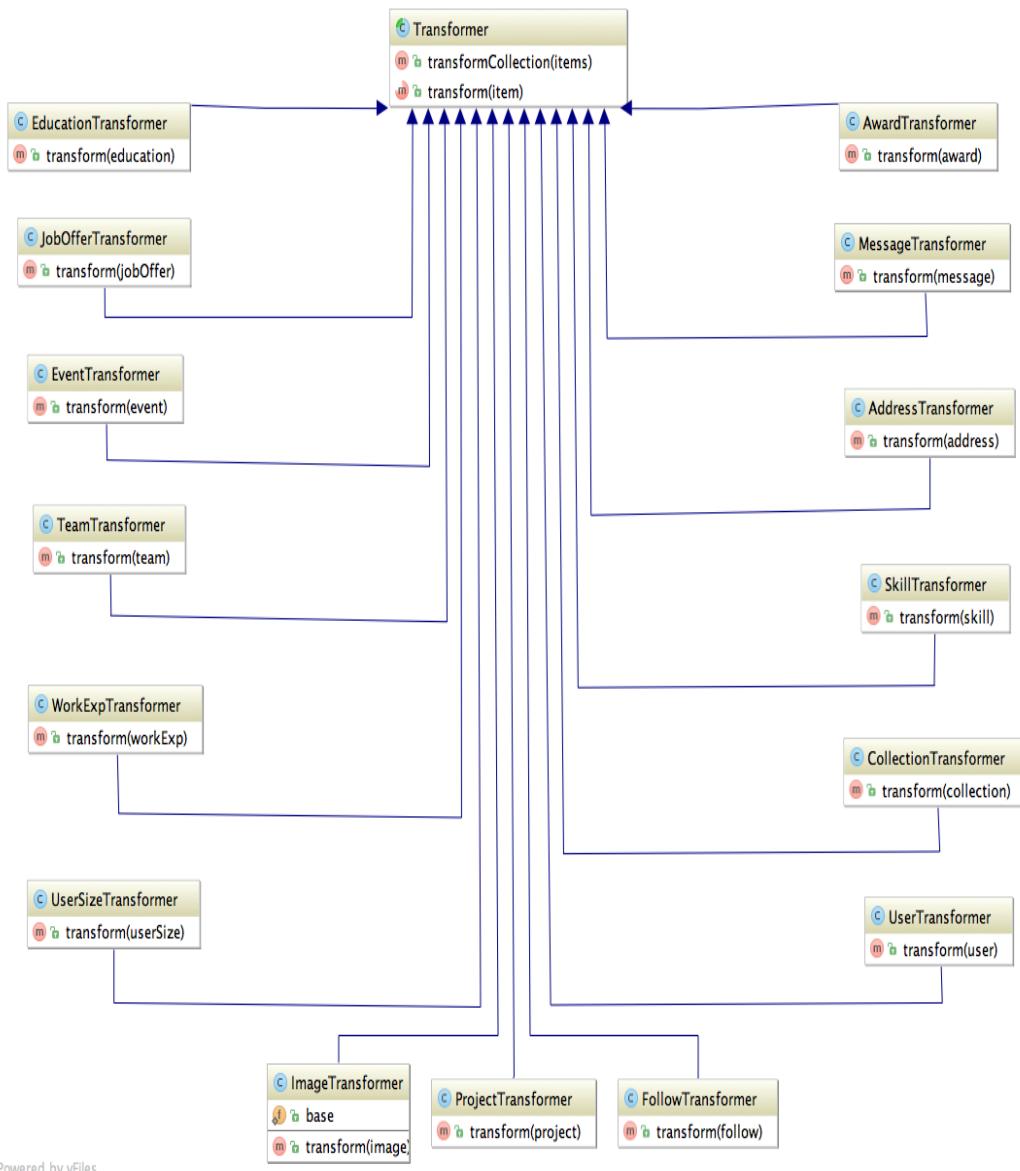


Figure A.5 – Diagramme de Classes : Package Middlewares

## Diagramme de classes du package : Transformers



**Figure A.6** – Diagramme de Classes : Package Transformers

---

## Annexe 2

### Extrait du modèle physique de données

#### Script SQL : Table th\_users

```
CREATE TABLE `th_users` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `professional` tinyint(1) NOT NULL DEFAULT '0',
  `admin` tinyint(1) NOT NULL DEFAULT '0',
  `firstName` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `lastName` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `email` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `emailVerified` tinyint(1) NOT NULL DEFAULT '0',
  `password` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `birthday` date DEFAULT NULL,
  `gender` enum('male','female') COLLATE utf8_unicode_ci NOT NULL,
  `registeredVia` enum('web','ios','android') COLLATE utf8_unicode_ci NOT NULL,
  `facebookId` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `facebookLink` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `twitterLink` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `googlePlusLink` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `personalWebsite` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `blocked` tinyint(1) NOT NULL DEFAULT '0',
  `appreciationsNbr` bigint(20) unsigned NOT NULL DEFAULT '0',
  `followersNbr` bigint(20) unsigned NOT NULL DEFAULT '0',
  `followingNbr` bigint(20) unsigned NOT NULL DEFAULT '0',
  `profileViewsNbr` bigint(20) unsigned NOT NULL DEFAULT '0',
  `remember_token` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  `created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  `updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  `deleted_at` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `users_email_unique` (`email`),
  UNIQUE KEY `Email_Index` (`email`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

#### Script SQL : Table th\_projects

```
CREATE TABLE `th_projects` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
```

---

```

`creatorId` bigint(20) unsigned NOT NULL,
`lastUpdaterId` bigint(20) unsigned NOT NULL,
`teamId` int(10) unsigned DEFAULT NULL,
`categoryId` int(10) unsigned DEFAULT NULL,
`subCategoryId` int(10) unsigned DEFAULT NULL,
`title` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
`description` text COLLATE utf8_unicode_ci NOT NULL,
`content` longtext COLLATE utf8_unicode_ci NOT NULL,
`status` enum('finished','in_progress') COLLATE utf8_unicode_ci
NOT NULL DEFAULT 'in_progress',
`visibility` enum('public','private') COLLATE utf8_unicode_ci
NOT NULL DEFAULT 'public',
`viewsNbr` bigint(20) unsigned NOT NULL DEFAULT '0',
`likesNbr` bigint(20) unsigned NOT NULL DEFAULT '0',
`sharesNbr` bigint(20) unsigned NOT NULL DEFAULT '0',
`created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
`updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
`deleted_at` timestamp NULL DEFAULT NULL,
PRIMARY KEY (`id`),
KEY `projects_creatorid_foreign` (`creatorId`),
KEY `projects_lastupdaterid_foreign` (`lastUpdaterId`),
KEY `projects_categoryid_foreign` (`categoryId`),
KEY `projects_subcategoryid_foreign` (`subCategoryId`),
KEY `projects_teamid_foreign` (`teamId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

```

```

ALTER TABLE `th_projects`
ADD CONSTRAINT `th_projects_ibfk_4` FOREIGN KEY (`subCategoryId`)
REFERENCES `th_subCategories` (`id`),
ADD CONSTRAINT `th_projects_ibfk_1` FOREIGN KEY (`creatorId`)
REFERENCES `th_users` (`id`),
ADD CONSTRAINT `th_projects_ibfk_2` FOREIGN KEY (`teamId`)
REFERENCES `th_teams` (`id`),
ADD CONSTRAINT `th_projects_ibfk_3` FOREIGN KEY (`categoryId`)
REFERENCES `th_categories` (`id`);

```

---

# Annexe 3

## Extrait de la documentation de l'API REST

### Message - getById Method

1.0.0 ▾

GET

<http://thimblefashion.com/api/public/api/v1/messages/getById/:userId>

Permission: User\_Auth

#### Header

Field	Type	Description
X-Authorization	String	API access-key.

Header-Example:

```
{  
  "X-Authorization": "8158f1047c2d68f29b2c35c21af6b44d8baff1e3"  
}
```

#### Request

Field	Type	Description
type	optional string	indicate the type of this user's messages to list (Sent or Received). Allowed values: "Sent", "Received"

#### Success 200

Field	Description
200	Operation Succeeded OK.

#### Error 4xx

Field	Description
404	the Item(s) was/were not found.

# Auth

## Auth - Login Method

1.0.0 ▾

POST

<http://thimblefashion.com/api/public/api/v1/auth/login>

Permission: None

### Header

Field	Type	Description
X-Authorization	String	API access-key.

Header-Example:

```
{  
    "X-Authorization": "8158f1047c2d68f29b2c35c21af6b44d8baff1e3"  
}
```

### UserLoginRequest

Field	Type	Description
email	string	User email address.
password	string	User password. Size range: 8..30

### Success 200

Field	Description
200	Operation Succeeded OK.

Login Success Example:

[User Already Logged In Example:](#)

```
{  
    "session_id": "8158f1047c2d68f29b2c35c21af6b44d8baff1e3",  
    "object": {  
        "id": 1,  
        "professional": false,  
        "firstName": "Marquis",  
        "lastName": "Nienow",  
        "email": "Jack.Stracke@Stracke.org",  
        "emailVerified": false,  
        "birthday": "2001-07-26",  
        "gender": "male",  
        "teamId": null,  
        "facebookId": null,  
        "facebookLink": null,  
        "twitterLink": null,  
        "googlePlusLink": null,  
        "appreciationsNbr": 0,  
        "followersNbr": 0,  
        "followingNbr": 0,  
        "reviewsNbr": 0,  
        "profileViewsNbr": 0  
    }  
}
```

### Error 4xx

Field	Description
422	Validation Error(s) while processing the Item(s).
403	Operation Forbidden.

---

## JobOffer

### JobOffer - Destroy Method

1.0.0 ▾

**DELETE****http://thimblefashion.com/api/public/api/v1/jobOffers/:id**

Permission: User\_Auth

**Header**

Field	Type	Description
X-Authorization	String	API access-key.

Header-Example:

```
{  
  "X-Authorization": "8158f1047c2d68f29b2c35c21af6b44d8baff1e3"  
}
```

**Success 200**

Field	Description
200	Operation Succeeded OK.

**Error 4xx**

Field	Description
404	the Item(s) was/were not found.

**Error 5xx**

Field	Description
500	Operation Failed, Server Internal Error.

---