Institut National des Sciences Appliquées et de Technologie

UNIVERSITÉ DE CARTHAGE

Projet de Fin d'Études

Filière : **GL ou RT**

Titre du projet

Présenté par

Flen FOULENI

Encadrant INSAT : **Mr FOULENI Flen**
Encadrant ENTREPRISE : **Mme FALTEN Flena**

Présenté le : –/–/**2017**

JURY

M. President FLEN (Président)
Mme. Rapporteur FLENA (Rapporteur)

Année Universitaire : 2016/2017

Dédicaces

À ma chère et tendre mère,
à celle qui m'a accueillie dans ce monde,
à celle qui a toujours veillé sur moi,
pleine d'amour et si attentionnée,
à celle qui n'a cessé de m'épauler dans mes moments les plus durs,
je te serais toujours reconnaissant pour tes sacrifices et tes efforts.

À mon cher père,
à celui que j'aime, que j'aimais,
à celui qui m'a donné de lui,
à celui qui n'a cessé de croire en moi,
à celui qui ne m'a jamais rien dénié,
à celui qui remplissait mon monde,
tu resteras à jamais dans mes pensées, repose en paix.

À ma chère famille,

À ma chère sœur ainée particulièrement, à qui j'exprime ma gratitude la plus profonde pour son support continu et son investissement dans mon assistance tout au long de la période de réalisation de ce travail.

Remerciements

Je tiens tout d'abord à remercier l'équipe de l'entreprise qui a accepté de m'accueillir et m'a procuré l'opportunité de réaliser mon projet de fin d'études dans une ambiance conviviale.

J'exprime mes sincères gratitudes à mon encadrant Omar HAOUES, responsable de mon encadrement à l'entreprise, pour sa disponibilité, sa coopération continue, ses efforts, et ses conseils précieux qui m'ont guidé dans la réalisation de ce travail.

Mes remerciements d'adressent autant à Mme Ghada GASMI, mon encadrante à l'INSAT, pour sa pédagogie, son amabilité, ses remarques pertinentes, son temps précieux et son encouragement continu qui m'a permis de mener à bien ce projet.

Je ne saurais oublier de remercier l'ensemble de mes professeurs au sein de l'INSAT pour leur apport inestimable dans ma formation au cours de ces cinq dernières années. Je remercie en particulier les membres du jury pour m'avoir accordé l'honneur d'accepter de juger la qualité de mon travail.

Je finis par adresser mes remerciements à toute personne qui m'a assisté de près ou de loin au cours de ces quatre mois de travail.

Table des Matières

Liste des Figures

Liste des Tableaux

Introduction Générale

La gestion de projet est l'une des briques de base de gestion pour toute entreprise. Pour réussir son développement et son évolution de manière correcte, toute entreprise se doit d'adopter une méthode de gestion efficace pour ses projets. Avec les changements continus auxquels celles-ci font face, à la compression remarquable du cycle de vie de déploiement des produits et de services, la réalisation de missions sous haute contraintes est devenue monnaie courante dans le train de vie quotidien d'une entreprise.

Pour relever les défis auxquels sont confrontées ces entreprises, l'intégration d'un outil d'assistance à la décision et d'aide à la gestion de projet est souvent la première étape vers l'optimisation des processus de gestion de l'entreprise. Le challenge réside dans la capacité de voir clair dans l'ensemble des projets entrepris, de s'assurer du bon établissement des projets et du respect des normes de gestion, et de réaliser efficacement ces derniers en utilisant le minimum de ressources pour accomplir les objectifs dans les délais imposés.

C'est dans le but de répondre à ce besoin que l'entreprise IT SERV a décidé de développer un outil d'aide à la gestion de projets, basé sur son propre Cloud. En effet, l'adoption d'un applicatif en mode Cloud présente plusieurs avantages pour les entreprises, dont le principal apport est une installation immédiate et la délégation de la gestion des rouages du système, de l'entreprise, vers le fournisseur de la solution.

Notre projet porte ainsi sur le développement d'un outil de gestion de projets, aspirant à s'aligner aux normes de gestion de projet existantes et à offrir une base générique pour la gestion de n'importe quel type de projet. L'outil est destiné tout d'abord à améliorer les services de consulting de projets offert par l'entreprise d'accueil à ses clients, et rentre d'une autre part dans une optique de fournissement de la solution en tant que service à part entière, destiné à une nouvelle catégorie de clients qui l'exploiteront pour leurs propres besoins de gestion de projet, indépendamment de leur supervision ou non par l'entreprise.

Le présent rapport est structuré en six chapitres brièvement décrits :

- **Mise en situation** : ce chapitre est consacré à l'introduction au cadre général du projet.
- **Étude préliminaire** : cette partie représente une étude théorique préliminaire à la phase de développement.
- **Préparation au lancement** : cette section porte sur la spécification des besoins et la présentation des choix techniques généraux.
- **Gestion de projet : API** : c'est le point d'entrée concret dans la phase de développement, il traite de la majeure partie du développement effectué en back-end.
- **Gestion de projet : Couche présentation** : la partie de développement en Front-end est abordée dans cette section.
- **Gestion des utilisateurs et des clients** : cette partie concerne le développement de fonctionnalités de gestion d'utilisateurs et de clients et traite de particulièrement de l'intégration de l'aspect SaaS au projet.

Nous clôturons ce rapport par une conclusion, dans laquelle nous évaluerons les résultats atteints et nous exposerons les perspectives éventuelles du présent projet.

Chapitre I

MISE EN SITUATION

Plan

Introduction

Avant de commencer à détailler le travail réalisé, il est indispensable d'introduire l'organisme d'accueil afin de se familiariser avec son domaine d'activité, la situation actuelle qui l'a motivé à entreprendre le projet traité au cours de ce rapport, ainsi que son orientation pour le futur. Une fois entamée, nous enchaînerons avec la présentation de la problématique, suivie d'une description sommaire du sujet du projet et des concepts clés associés. Nous conclurons avec la présentation de la méthodologie de travail adoptée tout au long de la mise en œuvre du projet.

1 Présentation de l'organisme d'accueil

1.1 L'entreprise " IT SERV "

IT SERV est une société de services et de conseil exerçant dans le domaine des Technologies de l'Information et de la Communication. Fondée en 2008, elle a très tôt opté pour un

Figure I.1 – Logo de IT SERV

positionnement stratégique sur les activités de conseil et de services dans le secteur des TIC en Tunisie. Elle a pour but de fournir des services à forte valeur ajoutée à ses clients. Dans ce cadre, elle vise essentiellement à les orienter vers l'adoption des technologies modernes, à maximiser l'efficacité et l'efficience de leur organisation et à optimiser leurs processus.

Ses valeurs fondamentales s'articulent autour de la satisfaction des clients d'une part et l'épanouissement et l'évolution de ses ingénieurs et consultants d'autre part. L'organigramme à la figure I.2 présente la hiérarchie générale de l'entreprise. C'est au sein du département Consulting que notre travail s'est déroulé.

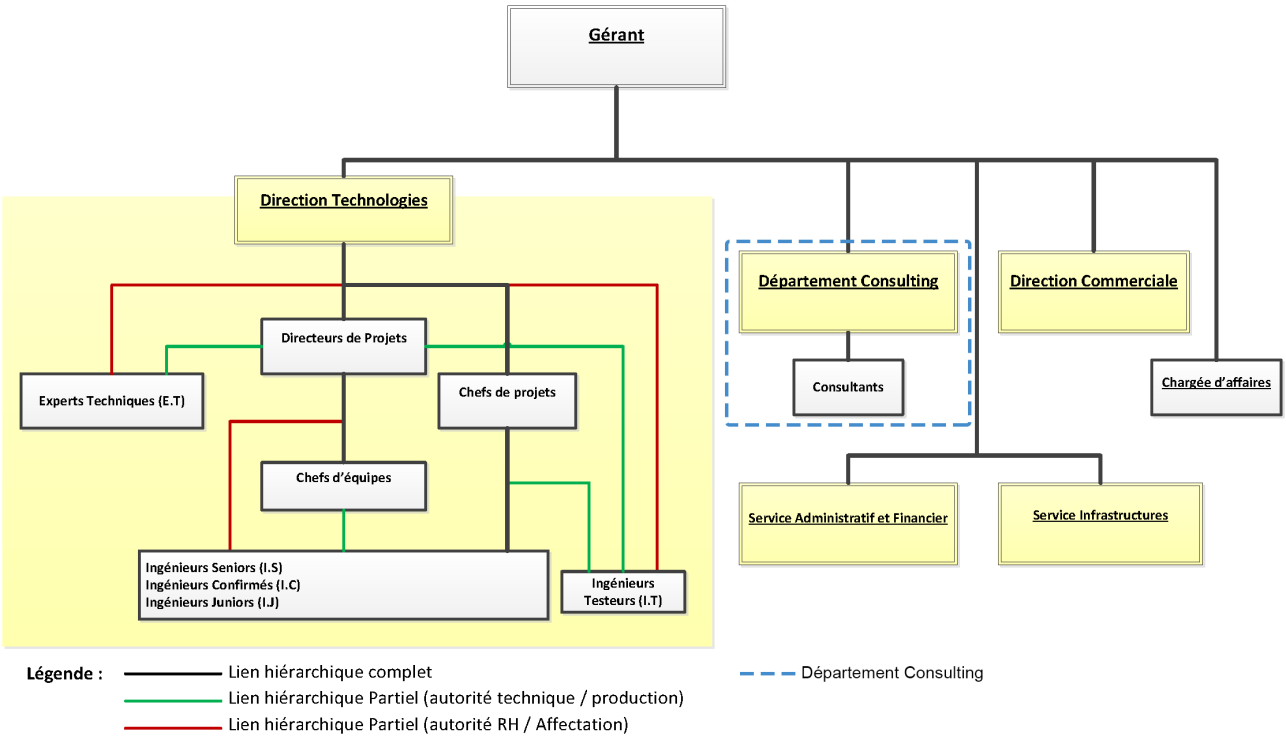

Figure I.2 – Structure organisationnelle de l'entreprise

L'entreprise, en plein essor, jouit d'une forte croissance sur tous les plans : chiffre d'affaires, résultats, effectifs, périmètre de compétence, portefeuille clients, etc. Sa réussite, à la fois en local et à l'international constitue un facteur de stabilité et un facteur de confiance en soi pour ses clients et ses collaborateurs.

IT SERV favorise les partenariats s'inscrivant dans la continuité et la durée. En effet, elle établit avec la majorité de ses clients des accords leur permettant de profiter de conditions avantageuses en matière de disponibilité, de priorité et d'engagement au plus haut niveau. Cette démarche stratégique lui permet d'avoir en contrepartie une visibilité long terme sur les commandes et les projets de ses clients, de dimensionner ses équipes en conséquence et d'optimiser son investissement en formation et mise à niveau de ses cadres en recherche & développement. La figure I.3 présente quelques une de ses références.

Figure I.3 – Références de IT SERV

1.2 Domaines d'expertise

L'entreprise vise la réalisation de missions complexes ayant un apport conséquent pour ses clients. La réussite de toutes ses missions et l'atteinte des objectifs fixés ont instauré des relations de confiance très durables et un élargissement progressif du périmètre d'intervention.

Les prestations d'IT SERV s'articulent autour de trois offres :

Consulting IT

L'entreprise propose une gamme de prestations de conseil qui se concentre sur trois axes :

— PMO : Gestion efficiente de projets clients. Le pilotage de projets peut porter sur l'intégralité d'un projet ou bien se focaliser sur une composante spécifique (l'initialisation du projet, la planification et le suivi de l'avancement, le suivi budgétaire et financier, etc).

— MOA : Aide dans l'accomplissement des activités de maîtrise d'ouvrage en systèmes d'information (rédaction de cahiers des charges, assistance à la sélection de fournisseurs, cadrage et assistance dans l'expression de besoins, etc).

— Audit : Audit informatique des composantes organisationnelles et techniques, incluant l'audit de systèmes d'information et d'applications.

Développement & Intégration

L'entreprise propose une offre complète allant du développement spécifique, à l'intégration de progiciels standards. Les prestations sont classifiées selon les deux catégories suivantes :

— Offres de services : conception et mise en œuvre de solutions spécifiques, intégration et interfaçage de systèmes, mise en application de divers outils de management et de supervision, etc.

— Expertise technologique : approches méthodologiques agile, expertise technologique (.Net, JavaEE, Oracle, PHP, ...), mise à disposition d'architectes techniques, expertise en solutions Workflows, B.I, etc.

Expertise Télécoms

L'entreprise propose ses services dans les différentes activités métiers et techniques des opérateurs, notamment : la facturation, la gestion de la relation client, le portail (Selfcare), les programmes de fidélisation, les services à valeur ajoutée (SMS/MMS Messaging, ...), etc.

La stratégie de développement de IT SERV se repose grandement sur l'établissement de partenariats stratégiques et durables avec les acteurs locaux ou internationaux dans ses domaines d'activités. Les partenariats établis permettent d'une part à IT SERV de disposer d'une base importante d'experts et consultants afin de répondre aux besoins actuels et futurs de ses clients, et lui permettent d'autre part d'accéder à des marchés nouveaux et de réaliser en collaboration avec ses partenaires des projets et missions nécessitant des expertises complémentaires ou des effectifs importants. Plusieurs missions ont été menées dans le cadre de ces accords de partenariat.

Figure I.4 – Partenaires de IT SERV

Plus récemment, l'entreprise aspire à élargir son éventail d'offres et entrevoit une extension de ses activités vers l'édition de logiciels.

2 Problématique et motivations

De nos jours, la gestion de projet s'impose dans les structures de toutes tailles comme le mode d'organisation par excellence. Un projet désigne un ensemble finalisé d'activités et d'actions entreprises dans le but de répondre à un besoin, soumis à des contraintes bien définies, généralement dans des délais fixés avec une allocation budgétaire plafonnée. La gestion de projet quant à elle, représente la démarche visant à organiser de bout en bout le bon déroulement d'un projet.
Celle-ci revient généralement à la gestion de multiples facettes, notamment :

— La planification
— La gestion budgétaire
— Le pilotage des risques
— La gestion des modifications
— La gestion des ressources

Avec les projets de plus en plus complexes auxquels les entreprises font face, gérer tous les flux d'information en perpétuel changement d'un projet se révèle être une tâche des plus ardues pour les chefs de projets. Sans un système efficace de traitement des données et d'acheminement de l'information, ces derniers finissent par être submergés par les requêtes, et se retrouvent le plus souvent dépassés par les événements. Cela se reflète en conséquence par un manquement au niveau de la réponse aux attentes des parties prenantes du projet, qui, souvent dû à des contraintes de disponibilité, restent trop longtemps en déphasage avec les dernières fluctuations et les mises à jour du projet pour pouvoir prendre les bonnes décisions aux moments opportuns.
C'est là que réside le cœur de la problématique : réussir à synchroniser de manière efficace toutes les parties prenantes d'un projet avec l'état actuel de ce dernier, en distinguant le chef de projet en tant que pièce maîtresse de sa supervision.

Le présent projet a pour but de répondre à ce problème en fournissant une solution logicielle

dédiée à la gestion de projets. La solution se doit de répondre adéquatement aux attentes des chefs de projets, en termes de fonctionnalités, mais aussi fournir un point pivotal d'aide à la décision pour les parties prenantes d'un projet. Pour répondre à leurs besoins, la solution devra offrir à ses usagers un niveau de flexibilité élevé par rapport aux exigences de son exploitation. En effet, les chefs de projets et autres parties prenantes doivent être en mesure de prendre avantage de cette dernière à tout moment, et à partir de n'importe quel lieu pour réussir à effectuer le suivi continu de leurs projets. Ceci requiert de la solution qu'elle soit accessible à partir du web, en plus d'offrir un support adapté à une utilisation via les dispositifs mobiles.

Au sein de l'environnement extrêmement concurrentiel dans lequel exerce l'entreprise d'accueil, et au vu du poids dont relève sa prestation de consulting, en l'occurrence la gestion de bout en bout de projets informatiques, il est clair qu'une solution élaborée, dédiée à la gestion du portefeuille de projets de ses clients, constituerais un point décisif face à la concurrence particulièrement grâce à la mise en avant de l'aspect décisionnel offert à ces derniers. Ceci aurait pour effet d'influencer grandement le niveau de satisfaction des clients et impacterais positivement l'image de la société.

Outre l'apport de l'application pour la gestion de ses propres projets en interne, dans l'optique de diversification de son domaine d'activités vers l'édition de logiciels, IT SERV entrevoit de développer une offre autour d'une version SaaS de l'application, disponible à partir de son propre Cloud. Dans ce cadre, le développement d'une interface de gestion des clients de cette offre s'impose en tant que partie à part entière du projet. Elle se verras traitée plus en détail au cours du chapitre VI.

3 Méthodologie de travail

Pour la réalisation d'un projet informatique d'envergure, il est indispensable de suivre une méthodologie de travail bien établie, prenant en compte les spécificités du projet, dans le but de garantir un niveau optimal de rendement tout au long de sa réalisation.

3.1 Choix de la méthodologie

Les avancées majeures dans l'univers de l'informatique ont été accompagnées d'une révolution dans la manière de faire et de penser pour produire des logiciels. Celle-ci a éventuellement donnée naissance au processus unifié [3] et au mouvement de développement agile [4], qui se caractérisent par le développement itératif et incrémental.
Les deux méthodologies initialement considérées pour la mise en œuvre du projet sont RUP [5] et Scrum [6]. Chacune met en avant les principaux avantages du processus unifié et des méthodes agiles, respectivement.

À première vue, le processus unifié semble assez similaire à l'approche agile. Néanmoins, il s'en distingue par certaines caractéristiques :

- Piloté par les cas d'utilisation et les risques
- Centré sur l'architecture
- Prescrit un niveau élevé de formalisme
- Préconise un travail considérable sur les spécifications en amont

Plus particulièrement, RUP s'impose en tant qu'implémentation standard du processus unifié. Son attrait majeur réside dans sa minimisation des risques très tôt dans le développement et l'adoption d'une approche rationnelle pour le cycle de développement de logiciel, scindée en quatre phases (pouvant elles-mêmes être subdivisées en sous-itérations) :

- Inception : essentiellement, cerner le système de façon adéquate pour établir la vision, déterminer les risques et établir des prévisions (coûts, charges, ...). Si le projet ne passe pas ce jalon, il peut être annulé ou répété après avoir été réajusté.
- Élaboration : l'objectif principal est d'atténuer les éléments de risque identifiés. C'est ici que le projet commence à prendre forme (analyse du domaine, architecture de base, ...).
- Construction : l'accent est mis sur le développement de composants et d'autres fonctionnalités du système. La majeure partie du codage prend place ici.
- Transition : transiter le système du développement à la production (bêta testing, contrôle qualité, etc).

Cette approche offre une description assez élaborée du processus de développement, ce qui se révèle être très utile pour le guider le d'une manière disciplinée et cohérente, tout en facilitant le travail de planification en aval. Cependant, cette discipline n'est pas sans prix. Le niveau de formalisme requis et la rigidité des processus prescrits pour la transition d'une phase à l'autre ainsi que pour la prise en compte du changement (planification lourde, principalement en amont, dates d'échéance, ...), nécessitant constamment la génération d'artéfacts divers (spécifications formelles, document d'architecture, ...) la rend inadaptée à l'application au projet présent, qui nécessite de prioriser le développement rapide de fonctionnalités et la validation fréquente du rendu.

Les méthodes Agiles pour leur part, partent du principe que spécifier et planifier dans l'intégralité les détails d'un produit (approche prédictive) avant de le développer est contre-productif. Le terme "agile" définit une approche de gestion de projet qui prend le contre-pied des approches traditionnelles. La notion même de "gestion de projet" est remise en question au profit de "gestion de produit". De façon à raisonner davantage "produit" que "projet". Après tout l'objectif d'un projet informatique consiste bien à donner naissance à un produit [7]. L'idée de l'approche consiste à se fixer un objectif à court terme et à se mettre en route sans tarder. Une fois ce premier objectif atteint, on effectue une brève revue de l'état actuel, et on adapte son itinéraire en fonction de la situation du moment, ainsi de suite, jusqu'à la destination finale. La méthode Scrum est la plus utilisée des méthodologies agiles existantes. Elle est caractérisée par :

- L'existence d'une auto-organisation
- Un nombre minimal de rôles définis (Scrum Master, Product Owner et équipe de développement)
- Un ensemble de cérémoniaux (réunion quotidienne, revue de sprint, . . .)
- La présence constante du client
- La mise en place de mécanismes favorisant les livraisons fréquentes (sprint, release)
- Une acceptation du changement (via un backlog dynamique)

Cette méthode est très attractive de par son approche du développement incrémental de produit

sous forme de releases (versions systèmes opérationnelles). Néanmoins, travaillant uniquement avec l'encadrant côté entreprise sur le projet présent, l'équipe de développement se résume simplement en ma personne. La mise de l'accent sur la collaboration entre les membres de l'équipe et la prescription de nombreux cérémoniaux à cet effet rend la méthode plus adaptée aux équipes beaucoup plus élaborées.

Dans le but de tirer profit des points forts de RUP tout en suivant une approche agile, s'inspirant notamment de Scrum, nous avons choisi de suivre une méthode hybride, flexible, permettant de tirer parti des principaux attraits des approches précédemment citées, à savoir : "Disciplined Agile Delivery" (ou l'agile discipliné) [8]. Celle-ci est présentée en détail dans la prochaine section.

3.2 Présentation de la méthodologie [1]

Disciplined Agile Delivery (DAD) est un cadre hybride qui s'appuie sur la base solide d'autres méthodes et cadres de processus logiciels. Le framework adopte des pratiques et des stratégies à partir de sources existantes et fournit des conseils pour savoir quand et comment les appliquer ensemble. Des méthodes telles que Scrum, Extreme Programming (XP) [9], Kanban [10] et Agile Modeling [11] fournissent les briques de processus et DAD le mortier pour les ajuster efficacement. L'un des avantages du développement de logiciel Agile et Lean réside dans la richesse des pratiques, des techniques et des stratégies à disposition. C'est aussi l'un de ses plus grands défis car, sans un cadre comme celui de la DAD, il est difficile de savoir vers lesquelles s'orienter et comment les intégrer.

Le diagramme de la figure I.5 montre une vue haut niveau du cycle de vie de la DAD. C'est un cycle de vie triphasé où l'on crée progressivement une solution consommable au fil du temps. Évidemment, il y a plus à la méthode que montre le diagramme précédent. DAD vise à être le moins prescriptif possible et s'efforce à s'adapter et à refléter la réalité du mieux possible. Quatre propositions du cycle de vie s'en suivent :

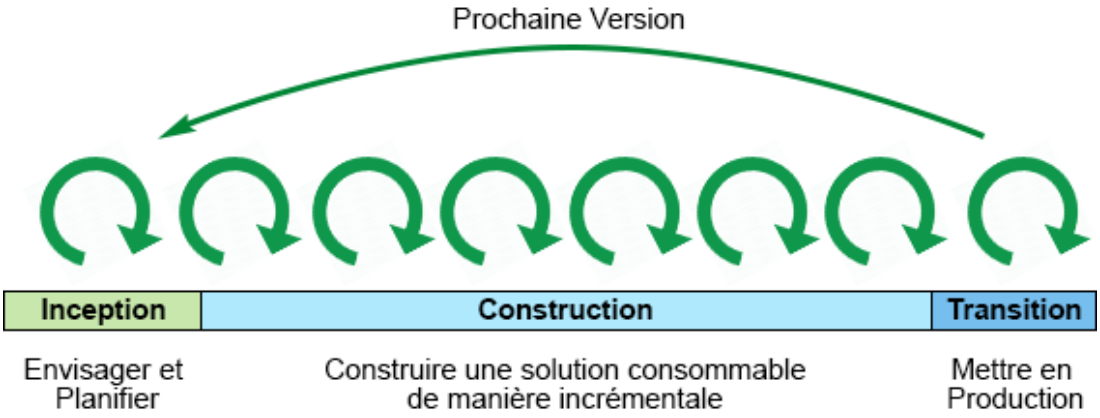

Figure I.5 – Cycle de vie haut niveau de la DAD

— Une version agile / basique qui prolonge le cycle de vie Scrum avec des idées éprouvées de RUP

— Un cycle de vie avancé / Lean

— Un cycle de vie de livraison continue Lean

— Un cycle de vie exploratoire “Lean Startup”

Notre choix s’est porté sur le cycle hybride, entre RUP & Scrum, illustré par la figure I.6.

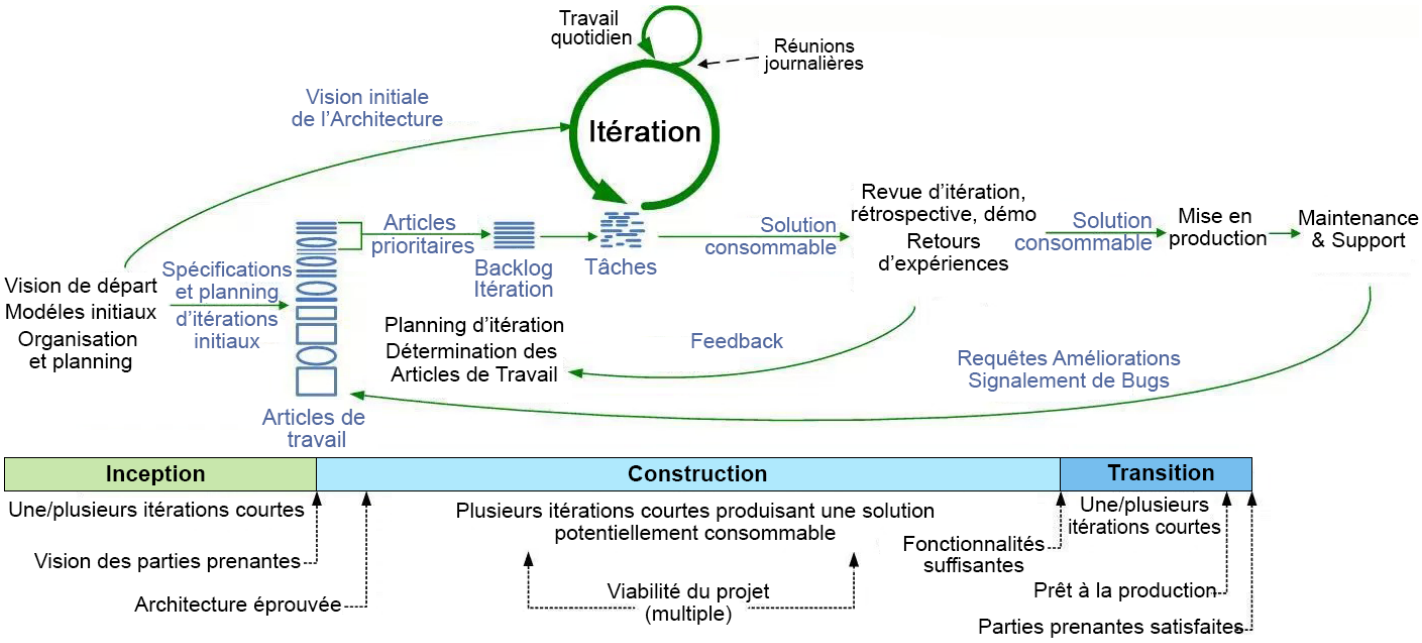

Figure I.6 – Cycle de vie basique de la DAD

Avec DAD, nous adoptons une approche axée sur les buts. Cette approche lui permet d'éviter d'être prescriptif et donc plus souple et plus adapté à notre situation. DAD indique simplement qu'il y a plusieurs problèmes entourant un but, dont l'on doit tenir compte, et propose plusieurs techniques pratiques à envisager pour le réaliser. DAD va plus loin encore et décrit les avantages et les inconvénients de chaque technique et dans quelles situations elle convient le mieux. Il soutient que certaines options conviennent pour la plupart des cas, et les recommande généralement, tout en fournissant des alternatives pour les scénarios atypiques. Le diagramme de la figure I.7 indique l'ensemble des buts de haut niveau associés à chaque phase de l'approche.

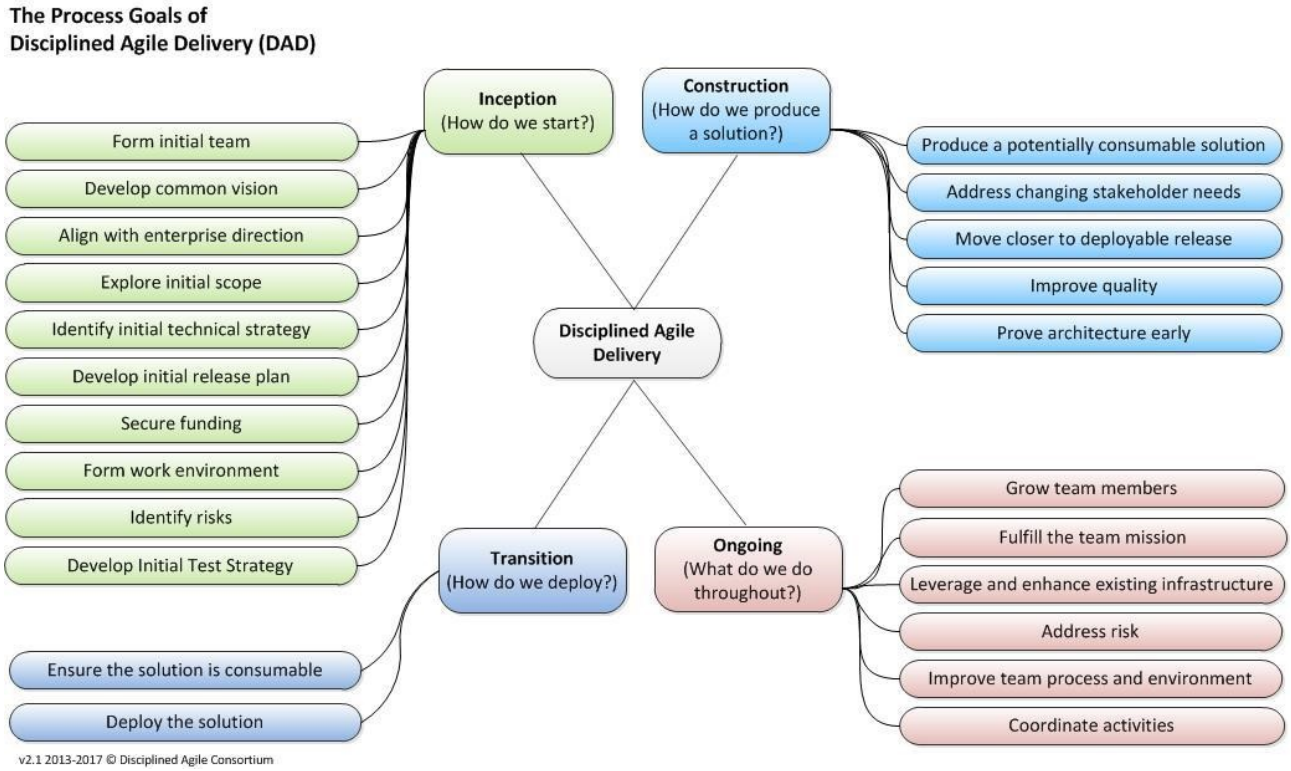

Figure I.7 – Buts associés aux phases de la DAD (en anglais)

L'illustration I.6 introduit le concept d'articles de travail ou "Work Items", assimilable au concept de Backlog de Scrum. Le Backlog fait office de carnet de fonctions, priorisées, contenant de courtes descriptions de toutes les fonctionnalités souhaitées dans le produit. La différence réside dans l'extensibilité du concept pour inclure des éléments outre fonctionnels (tel que l'élaboration de réunions, de formations, etc).

DAD intègre aussi un certain nombre de jalons légers ordonnés dans le temps dans ses cycles de vie, afin de guider et de mesurer les progrès et la santé d'un projet. Ils sont illustrés en bas de la figure I.6 :

- **Vision des parties prenantes** : entente générale sur l'objectif du projet entre les différentes parties prenantes.
- **Architecture éprouvée** : bonne pratique pour l'atténuation précoce des risques liés au bon fonctionnement global du système.
- **Viabilité du projet** (à multiples occurrences, optionnel) : à certains moments pendant le projet, les parties prenantes peuvent demander un point de contrôle pour s'assurer que le projet progresse conformément à la vision convenue, et choisir de la mettre à jour si nécessaire.
- **Fonctionnalité suffisante** : généralement vers la fin de la phase de construction.
- **Prêt à la production** : à un moment donné, il faut décider que la solution est prête à être produite. Pour les situations non triviales, une phase de transition formelle est souvent nécessaire pour faire les préparatifs finaux dans le cadre de la livraison de la solution aux parties prenantes.
- **Parties prenantes satisfaites** : les organismes de gouvernance et les autres parties sont informés quand l'initiative est officiellement terminée, afin qu'elles puissent initier une prochaine version ou diriger leurs fonds ailleurs.

Le journal de travail inclut les différents jalons encontrés tout au long de l'exécution du projet.

3.3 Environnement de travail

Pour les besoins de gestion de versions de notre travail, nous nous sommes reposés sur la solution Git [12]. C'est est un logiciel libre créé par l'auteur du noyau Linux. En 2016, il s'agissait du logiciel de gestion de versions le plus populaire utilisé [13]. Il se distingue par un système de gestion de versions décentralisé. Le choix de ce système de gestion de versions est judicieux dans notre cas, car il permet de garder une version complète de l'historique d'un projet en local, ce qui s'avère très utile pour le développement continue au sein et hors de

l'entreprise.

La gestion de la liste des articles de travail ou du "Backlog" a été réalisée à l'aide de l'outil Trello [14]. Celui-ci consiste à la base en un outil de gestion de projet en ligne, lancé en septembre 2011, et inspiré par la méthode Kanban. Ce dernier point le rend très adapté à la gestion agile dont nous avons besoin.

3.4 Journal de travail

Le diagramme de la figure I.8 permet de visualiser l'avancement du travail effectué au fil du temps :

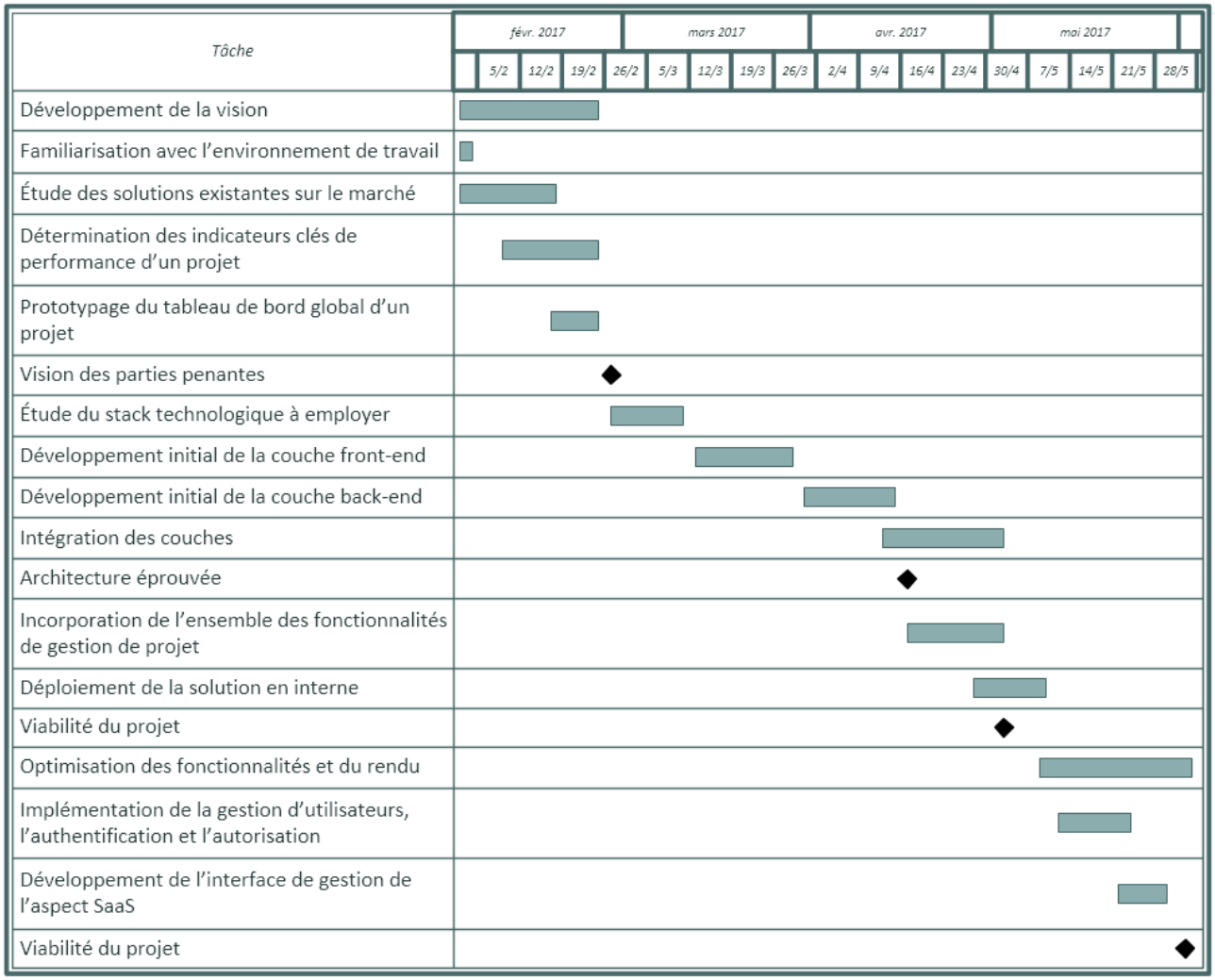

Figure I.8 – Diagramme de Gantt de l'avancement du projet

Conclusion

Dans ce chapitre, nous avons progressivement présenté le cadre général de travail, en commençant par introduire l'organisme d'accueil, pour ensuite exposer la problématique et les motivations qui ont donné naissance au projet. Pour finir, nous avons clarifié les choix méthodologiques effectués pour nous guider tout au long de sa mise en œuvre, en présentant les concepts fondamentaux associés à la méthode de travail ainsi que les étapes clés de sa réalisation.

Chapitre II

ÉTUDE PRÉLIMINAIRE

Plan

Introduction

Après avoir présenté le cadre général de notre projet, nous entamons par le biais de ce chapitre une étude théorique, préface à sa réalisation. Nous commencerons par présenter les notions de base associées à la partie métier du projet. Nous enchaînerons avec l'étude de l'existant, qui sera l'opportunité de présenter plus en détail la problématique rencontrée en interne par l'entreprise. Enfin, une vue d'ensemble des solutions similaires présentes sur le marché permettra de se faire une idée globale sur les fonctionnalités à attendre du produit.

Cette étude, partie intégrale de l'élaboration de la vision initiale du produit, s'inscrit dans la phase d'inception de la méthodologie de travail adoptée.

1 La Gestion de Projet

Les entreprises se confrontent à de nombreux challenges au cours de leur évolution (adaptation à des contraintes externes, lancement de nouveaux services ou produits, intégration de nouveaux outils, mises à jour des processus, ...). Chaque défi est relevé sous forme de projet. Un projet est défini sous forme d'une suite d'actions délimitée dans le temps, en vue de produire un résultat spécifique, un produit, un service ou une nouvelle organisation.
La gestion de projet a pour vocation de relever ces défis en mettant en place une organisation et une planification de l'ensemble des activités visant à assurer l'atteinte des objectifs du projet (qualité, coûts, délais, ...), à effectuer leur suivi, anticiper les risques et les changements à mettre en œuvre ainsi qu'à gérer la communication et la prise de décision.

On peut regrouper les activités de gestion de projet en deux phases :

- La phase d'organisation
- La phase de pilotage

Ces activités sont précédées par une phase de collecte d'informations et d'étude de faisabilité. Cette phase d'avant-projet permet de valider le projet et de décider des ressources à mobiliser pour sa mise en œuvre.

1.1 Phase d'organisation

La phase d'organisation comprend le développement des concepts définis dans la phase d'avant-projet et la définition du référentiel du projet, en fournissant suffisamment de détails pour pouvoir entamer la phase de réalisation. Les principaux éléments du référentiel d'un projet incluent :

- **La charte du projet** : Elle est rédigée par le chef de projet dès son lancement. Elle est destinée à toutes les personnes appelées à travailler sur le projet, ainsi qu'aux tiers concernés (la hiérarchie, les chefs de service, ...). C'est un document de référence à usage strictement interne dont le but est de fournir les informations de base du projet : son objet, ses parties prenantes, ses enjeux, ses contraintes, etc.

— **La structuration du projet** : La structure de découpage du projet organise et définit la totalité du contenu d'un projet. La structuration du projet dépend de nombreux paramètres (la complexité, la structure de l'organisation, le type de gestion, ...) et a pour objectif de découper le projet en plus petits éléments, plus faciles à gérer, afin de pouvoir définir des coûts et des durées pour chaque élément, ainsi que des résultats tangibles et mesurables.

— **Le planning de référence** : La planification d'un projet est l'activité qui consiste à déterminer et à ordonnancer les tâches du projet, à estimer leurs charges et à déterminer les profils nécessaires à leur réalisation. Le planning résultant inclut la liste des tâches et leurs antériorités, la définition des lots de travaux ainsi que les ressources associées.

— **Le plan de communication** : C'est un document qui présente les méthodes et les modalités adoptées pour collecter, conserver et diffuser l'information, les destinataires de chaque type d'information (rapport, document technique, ...), ainsi que les responsabilités associées.

— **Le budget de référence** : Résultat de l'analyse des coûts rattachés au projet, il est établi à partir de la liste des activités, des ressources associées et du planning de référence.

— **L'analyse des risques** : Elle comporte un recensement des aléas majeurs associés au projet ainsi que les plans d'action et les mesures préventives à mettre en place.

1.2 Phase de pilotage

Une fois budgétisé, organisé et planifié, le projet démarre. Le pilotage du projet permet de comparer le réalisé avec le prévisionnel, et de réviser éventuellement les plannings et les charges. Le chef de projet et son équipe s'efforcent de respecter le référentiel de gestion du projet, tout en mesurant les écarts constatés. Cela inclut la maîtrise des aspects suivants :

— **Les délais** : Le chef de projet recueille l'information sur l'avancement réel du projet à partir de l'avancement de chaque tâche, et identifie les retards actuels et potentiels. Dès lors, des actions correctives peuvent être mises en place afin de corriger les écarts (modification du planning, réallocation de ressources, ...).

— **Les coûts** : Une évaluation de la consommation budget est fréquemment entreprise, suivie d'une réestimation du budget qui reste à consommer. Le coût total prévu comparé au budget de référence permet d'analyser et d'anticiper des écarts ainsi que d'identifier les causes de dépassement.
— **Les modifications** : Elles sont le résultat d'événements extérieurs (changement de réglementation, changement demandé par le client, erreurs ou oublis dans la définition initiale, ...). Les demandes de changements peuvent apparaître sous plusieurs formes et conduire à un élargissement ou à une limitation du contenu et doivent être traitées grâce un ensemble de procédures formelles.
— **Les risques** : La maîtrise des risques permet d'identifier, de quantifier, de réduire ainsi que de suivre les risques encourus tout au long de la vie du projet.

Le chef de projet se focalise essentiellement sur la maîtrise des coûts et des délais.

L'usage d'indicateurs de pilotage est crucial pour le pilotage efficace d'un projet. Ces outils d'assistance à la décision permettent de mesurer de manière pratique une situation ou un risque, d'alerter ou au contraire de signifier l'avancement correct d'un projet.

2 Étude de l'existant

L'étude de l'existant permet d'identifier les points forts ainsi que les points faibles de la solution existante. Cette étude permettra de bien cibler les besoins de l'entreprise, en vue d'en tenir compte lors de la conception et du développement de la solution.

2.1 Description de l'existant

La solution de gestion de projet actuelle se base sur l'utilisation du logiciel Excel pour générer des documents associés aux différentes activités d'un projet. En effet, le chef de projet répertorie chaque catégorie d'artéfact associée à la gestion d'un projet sous forme de tableur Excel. Il procède ainsi à l'ajout et l'édition d'entrées pour chaque catégorie d'artéfact et y indique l'ensemble des informations requises. Cette procédure est facilitée par la duplication de

fiches Excel de base, vierges, préparées à l'avance pour chaque catégorie. Ainsi, pour chaque catégorie d'artéfact, l'ensemble des entêtes et des valeurs prédéfinies pour chaque champ y est déjà déclaré. La table d'entrées d'une fiche est souvent accompagnée d'un ensemble d'indicateurs, statiques et dynamiques, pertinents à la catégorie en question. Des exemples de ces fiches sont exposés aux figures II.2 et II.1, traitant respectivement des artéfacts actions et risques pour un projet.

Figure II.1 – Exemple d'une fiche de suivi des actions

Figure II.2 – Exemple d'une fiche de maîtrise des risques

Le chef de projet est régulièrement chargé de diffuser les mises à jour relatives à un projet, périodiquement ou à la demande, au travers de l'envoi de messages électroniques directement aux parties concernées.

Cette méthode de travail possède certains atouts, mais dissimule dans l'ensemble plusieurs inconvenances qui nuisent à son niveau d'efficacité et à la performance de la gestion de projets par l'entreprise. Ces points sont exposés dans la section suivante.

2.2 Critique de l'existant

La présentation de la méthode de gestion actuelle nous permet de déceler des limitations importantes, inhérentes à la démarche générale.

Reconnaissons tout d'abord les mérites non négligeables de celle-ci, à savoir :

- Disponibilité de fiches de base, contraignant les champs à choix restreints aux options prédéfinies et assistant à leur remplissage
- Présence de champs autocalculés ainsi que de supports visuels utiles pour la compréhension de certains champs, dont la signification peut s'avérer obscure
- Génération automatique d'indicateurs riches (statistiques numériques, graphiques en camembert, ...)

Outre ces quelques bénéfices, cette méthode de gestion présente les nombreuses limitations suivantes :

- **Perte en productivité** : induite d'un déficit d'automatisation de la démarche de gestion des fichiers
- **Fragmentation des données** : les données relatives à chaque projet sont réparties, sous forme d'une multitude de documents, souvent désorganisés
- **Journalisation non fiable des mises à jour** : la journalisation des mises à jours relève entièrement de l'organisation choisie par le chef de projet, et requiert un effort

manuel, exclusivement dédié à la tâche

- **Traçabilité impossible** : l'origine des modifications des fichiers n'est pas connue
- **Stockage inefficace** : problème de centralisation pour l'ensemble des données des projets menés par l'entreprise, redondance de données, organisation manuelle, etc
- **Sécurité non garantie** : la sécurisation des données (documents) est entièrement à la charge des parties prenantes y ayant accès
- **Confidentialité difficile à assurer** : elle implique en pratique l'édition soigneuse, au cas par cas, des fichiers partagés
- **Synthèse des données impossible** : impossible de se reposer fiablement sur les données des fichiers pour les besoins d'aide à la décision de l'entreprise (à cause des points susmentionnés)
- **Collaboration fastidieuse** : contribution d'intervenants non gérée par la solution actuelle. Le chef de projet transforme individuellement tous les flux d'information en données, avant de les indiquer dans le fichier approprié

Toutes les limitations susmentionnées imposent pour l'entreprise une révolution dans sa méthode actuelle de gestion de projets. Dans le cadre de notre projet de fin d'études, la solution à concevoir aspire ainsi à combiner les points forts de la démarche existante tout en apportant une solution efficace à ses problèmes inhérents.

3 Étude des solutions existantes sur le marché

Le problème de gestion de projet est aussi vieux que le concept de projet lui-même. En conséquence, un nombre important de solutions logicielles d'aide à la gestion de projet est disponible aujourd'hui sur le marché. Dans le but de produire une solution de qualité, une étude de l'existant s'impose afin de prendre en considération les points forts et les points faibles des solutions existantes. Celle-ci servira surtout à nous guider dans l'élaboration de la vision de notre produit.

3.1 Présentation des solutions existantes

Les solutions présentes sur le marché se distinguent les unes des autres essentiellement par le niveau de fonctionnalités offert et le domaine d'application cible. En effet, des solutions populaires, telle que JIRA [15], se limitent à un domaine très particulier, en l'occurrence la gestion de projets agiles pour cette dernière. Les solutions étudiées aspirent à être le plus générique possible, capable d'être employées dans n'importe quel contexte de projet. L'échantillon exposé ci-dessous vise à présenter une vue d'ensemble des caractéristiques communes et distinctives des solutions les plus populaires sur le marché qui partagent cette vision.

MS Projects [16]

Microsoft Projects a su s'imposer pendant longtemps en tant que solution de facto pour les besoins de gestion de projet en entreprise. Faisant partie de la suite logicielle de l'éditeur, MS Projects ne fait pas exception et s'intègre pleinement à sa plateforme collaborative, ce qui représente un réel bénéfice pour les clients ayant déjà opté pour l'intégration de leurs données sur cette dernière. Le logiciel souffre cependant d'un manque de flexibilité par rapport à l'approche de gestion de projet offerte ainsi que d'une courbe d'apprentissage pénible pour les nouveaux usagers.

ProjectManager.com [17]

ProjectManager.com couvre très bien les fonctionnalités de base de gestion de projet et, parmi toutes les solutions étudiées, s'apparente le plus à la vision initiale élaborée. Cette solution, disponible sous une offre SaaS, s'adapte parfaitement aux besoins des petites et moyennes entreprises, qui retrouvent une simplicité d'usage associée à une couverture plus que décente des piliers de la gestion professionnelle de projet.

Zoho Projects [18]

Zoho propose une suite logicielle qui répond aux besoins des particuliers et des entreprises. Dans cette optique, sa solution de gestion de projet s'intègre parfaitement avec certains de ses

autres produits, afin d'étendre ses fonctionnalités. La solution se veut simple, mais complète, en offrant un éventail étoffé de fonctionnalités pour la gestion des divers facettes d'un projet. Elle se caractérise par une attention particulière apportée à l'ergonomie et au confort général d'utilisation.

EasyRedmine [19]

Redmine est la solution open source gratuite la plus complète disponible sur le marché. Cependant, son manque d'intuitivité et le faible niveau d'ergonomie offert, associés à son interface datée, nuisent à son potentiel et en font une solution peu courante d'usage dans la pratique. C'est tout d'abord dans le but d'accroître son utilisabilité, tout en prenant avantage de la base de fonctionnalités existante, que la solution EasyRedmine offre une version commerciale, basée sur cette solution. Depuis, l'application s'est vu évoluer vers l'une des solutions les plus abouties du marché, s'alignant gracieusement avec les standards de gestion de projets IPMA et PMI.

Clarizen [20]

Due à l'amélioration continue de son produit et à la complétude de sa vision, l'entreprise Clarizen a vu son produit se hisser à la position de Leader dans les classements Gartner [21] et Forrester de 2016 [22], dans la catégorie des solutions SaaS de gestion de projets. De ce fait, ce produit a sans nul doute sa place dans notre étude et impose un bilan des fonctionnalités majeures qui lui valent son succès.

Dans le but de présenter un échantillon pertinent, d'autres solutions ont été omises, notamment pour des raisons de redondance ou de pauvreté en termes de fonctionnalités. Nous poursuivons avec la mise en relief des points communs et distinctifs de chacune dans la partie qui suit.

3.2 Synthèse des solutions existantes

Le tableau II.1 présente une comparaison haut niveau des solutions étudiées. On y observe que les solutions retenues partagent un tronc commun de fonctionnalités destiné à la gestion des aspects primordiaux d'un projet. Elles se différencient néanmoins sur d'autres points, tel que la nature de l'offre et le support de l'accès mobile.

Tableau II.1 – Comparatif des solutions de gestion de projet présentées

Spécificités	**MS Proj.**	**P.M .com**	**Zoho Proj.**	**EasyProject**	**Clarizen**
Nature d'offre	Bureau	SaaS	SaaS	SaaS & Local	SaaS
Planification	✓	✓	✓	✓	✓
Gestion Budget	✓	✓	✓	✓	✓
Structuration	✓	✓	✓	✓	✓
Gestion Ressources	✓	✓	✓	✓	✓
Gestion Problèmes	✓	✓	✓	✓	✓
Gestion Modifications	✗	✓	✓	✓	✓
Reporting	✓	✓	✓	✓	✓
GED	✓	✓	✓	✓	✓
Pilotage Risques	✓	✓	✓	✓	✓
Support Mobile	✗	Web	App	App	App
Collaboration	✗	✓	✓	✓	✓
Open Source	✗	✗	✗	✓	✗

Vu son expertise en développement d'applications, IT SERV a choisi de démarrer le développement de sa propre solution d'aide à la gestion de projet. Dans le cadre de notre projet de fin d'études, nous visons à concevoir une solution s'inspirant d'une sélection des traits jugés primordiaux des solutions étudiées. Ceux-ci seront établis au cours du chapitre suivant.

Conclusion

Afin de mieux cerner la problématique et de proposer une solution de qualité, tenant compte des spécificités de notre projet, nous avons établi l'état des lieux de l'existant, que ce soient les méthodes de gestion courantes de l'entreprise, ou les solutions similaires disponibles sur le marché.

Avec ces informations en main, le chapitre suivant nous permettra de dégager précisément les exigences de notre produit.

Chapitre III

PRÉPARATION AU LANCEMENT

Plan

Introduction

Avant d'appréhender le développement du système, il est primordial d'acquérir une compréhension claire des besoins des parties prenantes de notre projet et des fonctionnalités escomptées du système. Ce chapitre couronne l'étape d'élaboration de la vision de notre produit, par la spécification des besoins. La phase d'inception se poursuit avec l'édification de l'architecture globale du système et le choix de l'environnement technique. Ces deux parties concluront le chapitre et annoncent l'achèvement de la phase d'inception.

1 Analyse des besoins

L'analyse des besoins a pour objectif l'identification des acteurs du système et de leurs rôles, ainsi que la spécification des besoins et des contraintes contre lesquelles le produit final sera validé. Il existe deux types de besoins :

— Les besoins fonctionnels, qui présentent ce que l'utilisateur attend en termes de service
— Les besoins non fonctionnels, qui présentent les contraintes sous lesquelles l'application doit être opérationnelle

1.1 Objet global du projet

L'objectif du projet consiste à la conception, au développement, ainsi qu'au déploiement d'une application web de gestion de projets, compatible avec tous les terminaux, de format large ou réduit, et disponible d'usage principalement en mode SaaS.
Pour toute organisation cliente, l'application permettra essentiellement aux responsables, chefs de projets, de gérer les différents aspects des projets entrepris par leur organisation, ainsi que d'en monitorer l'état.

Le produit final comportera ainsi deux parties distinctes :
— Une application web de gestion de projets en mode SaaS
— Une solution complémentaire pour la gestion de l'aspect SaaS

1.2 Identification des acteurs

L'application est destinée à être acquise par une organisation de petite ou de grande envergure (entreprise, équipe, ...). Au sein de celle-ci, nous pouvons distinguer différents types d'acteurs :

— **L'administrateur** : C'est l'utilisateur associé au compte existant par défaut lors de l'acquisition de la solution. Il possède les pleins pouvoirs sur le reste des comptes et a la charge de créer le reste des comptes au tout début de la mise en route de l'application.
— **Un dirigeant** : Cet utilisateur s'occupe de la gestion du portefeuille global de projets. Il est chargé d'affecter les chefs de projets aux projets. Ne pouvant lui-même gérer un projet, il garde la capacité de consulter tous les projets en lecture seule. Son compte est créé et géré par l'administrateur.
— **Un chef de projet** : C'est l'utilisateur fondamental du système. Il s'intéresse essentiel-

lement à l'aspect de gestion de projets. Il peut néanmoins créer des comptes pour les parties prenantes d'un projet qu'il supervise, afin de leur offrir un moyen de monitorer en continu l'état de leurs projets.

— **Une partie prenante** : Cet utilisateur se limite à la consultation de projets en lecture seule. Son compte est créé et géré par un chef de projet.

Pour la solution complémentaire de gestion de l'offre SaaS, on reconnait un seul type acteur, à savoir : L'administrateur SaaS. C'est lui qui gère les clients de l'offre SaaS. Il a la charge de monitorer l'usage de l'application en mode SaaS et de remédier aux requêtes des clients.

1.3 Spécifications fonctionnelles

On énumère ici les différentes contraintes et besoins fonctionnels requis de la part du livrable final.

L'application doit être en mesure d'offrir les fonctionnalités suivantes :

— **Gestion du portefeuille de projets** : L'ensemble des projets du client doit être regroupé sous un portefeuille, qui offre des fonctionnalités d'ajout et d'édition de projets.

— **Gestion de la charte d'un projet** : Il s'agit d'offrir à l'utilisateur la possibilité de fournir les détails relatifs à la charte d'un projet dès sa création, tout en gardant la possibilité d'en éditer le contenu à postériori.

— **Reporting au niveau projet** : Une vue d'ensemble de l'état global d'un projet doit être accessible au travers d'un tableau de bord, contenant un agencement d'indicateurs clés pour le pilotage d'un projet.

— **Gestion de l'intégration** : Un projet doit pouvoir être structuré sous les différents niveaux suivants : projet, sous projet et chantier. Un projet peut inclure un ensemble de sous projets ou de chantiers lui étant directement rattachés. Un sous projet existe uniquement sous un projet et peut lui-même contenir une multitude de chantiers lui étant directement rattachés.

— **Gestion du plan d'action** : Une action est une tâche à réaliser, disposant essentielle-

ment d'une description, d'un responsable et d'une date de clôture planifiée. Une action doit pouvoir être rajoutée sous n'importe quel niveau d'un projet, et peut être mise à jour ou bien supprimée. Un indicateur sur les actions en retard doit être mis à disposition.

— **Gestion des coûts** : Le budget d'un projet ou d'un sous projet peut être renseigné et mis à jour tout au long de son existence. La gestion du budget inclut la spécification du budget initial, du budget consommé ainsi que d'une estimation du budget qui reste à consommer. Elle offre par ailleurs un indicateur sur le budget total prévisionnel.

— **Gestion du reste à faire** : Un reste-à-faire se distingue par une description et une charge associée, en homme / jour. Les restes à faire peuvent être créés et mis à jour, et sont rattachables à n'importe quel niveau d'un projet.

— **Planification** : La planification d'un projet ou d'un sous projet sera possible au travers de la création et de la consultation ultérieure des principaux jalons du niveau de projet. En essence, un jalon est défini par une brève description associée à une date d'échéance.

— **Gestion du "Scope"** : La gestion du "scope" ou du périmètre d'un niveau de projet s'accomplit principalement via la fourniture de documents en pièces jointes au niveau en question. D'autre part, la gestion du périmètre couvre la gestion du changement et des points en suspens. Un point en suspens est un point relatif au niveau de projet, en attente de résolution, auquel un responsable est assigné. La gestion du changement quant à elle est formalisée au travers de la soumission de demandes de changement. Chaque demande inclut essentiellement le demandeur ainsi que la décision prise la concernant.

— **Gestion des risques** : Le définition des risques peut être réalisée à n'importe quel niveau d'un projet. Un risque est défini par une description du risque encouru, associée à une probabilité et un impact. Ces derniers sont utilisés pour générer un indicateur sur le niveau de criticité (KRI - Key Risk Indicator) du risque. Leur suivi est réalisé principalement au travers de la spécification du statut, du plan d'action et des dispositions à prendre pour chaque risque, ainsi que de dates de qualification et de clôture.

— **Gestion des ressources humaines** : La définition des ressources humaines peut être réalisée au niveau du portefeuille global de projets ou bien au niveau d'un projet ou d'un sous projet. Les ressources globales peuvent plus tard être assignées à un projet

en particulier. Au sein d'un projet ou d'un sous projet, les ressources humaines peuvent être renseignées, mises à jour, et éventuellement assignées à une action ou un point en suspens. Une ressource humaine peut également être assignée en tant que responsable d'un plan de communication au niveau d'un projet.

— **Gestion de la communication** : La gestion de la communication se déroule au sein d'un projet. Elle inclut la définition de plans de communication, se distinguant par une description et un responsable communication, de même que la planification de réunions, disposant chacune d'un nom, d'une date et d'un lieu, accompagnés d'un statut, ainsi que des comptes rendus de réunions.

— **Gestion des ressources non humaines** : Toute ressource matérielle ou immatérielle associée à un niveau de projet doit pouvoir être renseignée et mise à jour.

— **Publication de mises à jour** : L'état d'un projet doit pouvoir être archivé sous forme d'une mise à jour.

D'autre part, la solution adjointe de gestion du service SaaS devra quant à elle permettre de gérer les clients de l'offre SaaS, à savoir l'ajout et l'édition de clients et des détails associés.

1.4 Spécifications non fonctionnelles

En plus d'apporter les fonctionnalités citées précédemment, le produit final devra être en mesure d'assurer les aspects suivants :

— **Adaptabilité** : Supporter la majorité des dispositifs des utilisateurs (smartphones, tablettes, ordinateurs, ...) de manière adaptée à chacun.

— **Sécurité** :

- Isoler proprement les données des clients, et limiter efficacement leurs accès d'un client à un autre.
- Mettre en place un système d'authentification et d'autorisation robuste.

— **Ergonomie** : Offrir une interface conviviale, intuitive et facile d'usage pour tous les types de dispositifs supportés.

1.5 Diagramme de cas d'utilisations

La figure III.1 présente le diagramme de cas d'utilisation global de l'application. Il offre une représentation des principaux services offerts par le système, en fonction du type d'utilisateur.

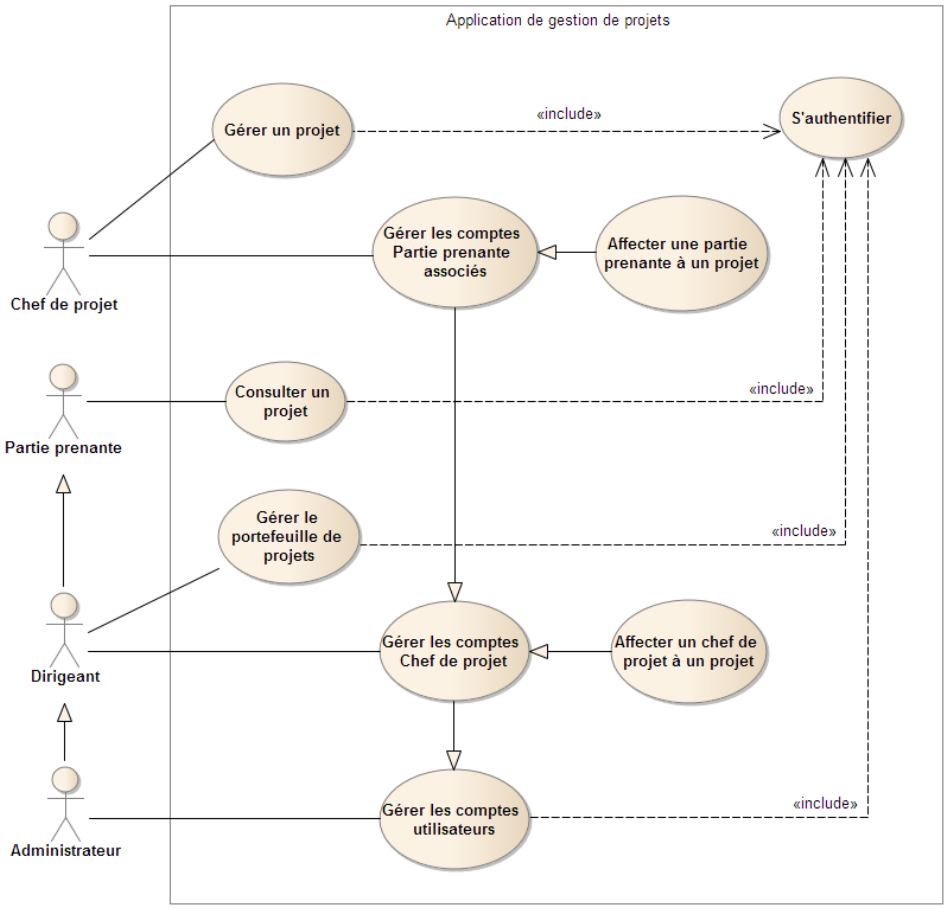

Figure III.1 – Diagramme de cas d'utilisation global de l'application de gestion de projets

Nous pouvons catégoriser les fonctionnalités à offrir aux utilisateurs en deux types :

— **Gestion de projets** : Gestion du portefeuille de projets global et des projets en particulier. Le portefeuille de projets représente simplement une collection regroupant les projets existants. La consultation d'un projet est incluse dans cette catégorie. Ces aspects seront adressés tout le long des chapitres IV et V.

— **Gestion d'utilisateurs** : Gestion de comptes utilisateurs. Elle sera traitée au cours du chapitre VI.

2 Étude technique

Avant d'entamer la phase de développement, il est nécessaire de prendre du recul par rapport aux exigences et d'établir une vue d'ensemble du système d'un point de vue architectural. Dans notre cas en particulier, l'aspect SaaS de l'application impose une étude approfondie de l'infrastructure sous-jacente à mettre en place et des technologies à employer.

2.1 Architecture générale

Une vue d'ensemble de l'architecture permet de mettre en évidence les différentes couches du système à développer.
Au vu du caractère Web de l'application, nous nous sommes d'abord orientés vers une architecture traditionnelle en 3-tiers [23], pour finalement opter pour une architecture en 4-tiers, se démarquant par l'ajout d'un serveur statique séparé du serveur d'application, dédié à la récupération du front-end. Le schéma de la figure III.2 illustre la relation entre les différents composants de cette architecture.

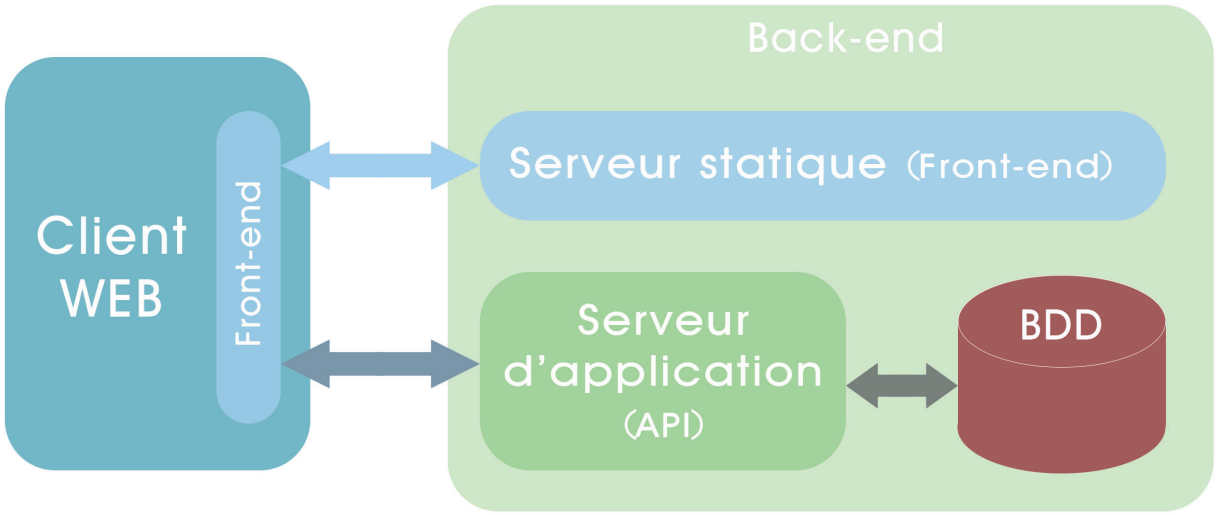

Figure III.2 – Architecture physique du système

Notre système modélisé par le Back-end, chaque tiers représente l'une de ses couches. L'architecture logique du système se retrouve divisée en trois couches :

- La couche d'accès aux données : Cette couche a pour charge la gestion des données du système (stockage, persistance, gestion de l'accès, ...). Elle est représentée par notre base de données.
- La couche métier : C'est la partie fonctionnelle du système. Elle implémente la logique opératoire métier et se repose sur les données fournies par la couche inférieure. Elle offre des services à destination de la couche de présentation. Elle est représentée par notre serveur d'application, qui fournit ces services sous forme d'API.
- La couche de présentation : Elle encapsule l'interface utilisateur et correspond à la partie visible de l'application. Cette couche se repose sur les services offerts par la couche métier pour traiter les requêtes de l'utilisateur et s'occupe de la restitution et de la mise en forme des informations récupérées. Elle est représentée par notre serveur statique, qui fournit l'application front-end au client.

Dans notre contexte, cette architecture en couches est la plus appropriée. L'architecture 3-tiers traditionnelle permet de bénéficier de :

- Une répartition claire des responsabilités entre les couches
- Un couplage faible entre les différents niveaux
- Une simplicité lors de la réalisation de tests (la logique métier est isolée du reste des couches)
- Sécurité accrue, due à la séparation entre le client et les données
- La possibilité d'utiliser des clients légers (adaptés aux dispositifs mobiles)
- Une évolutivité élevée, résultant de l'indépendance des couches

À partir de là, l'étape suivante logique pour une séparation plus aboutie des préoccupations, et donc une maintenance facilitée, a été l'orientation vers l'architecture en 4-tiers. Au travers de son adoption, nous avons clairement séparé entre l'application Front-end et l'application en charge de l'exposition de notre API. L'administration de ces deux applications sous la forme de deux projets distincts nous permet en plus de bénéficier de :

— Un découplage net entre les rôles des deux applications
— Une évolution indépendante du Front-end
— Un déploiement séparé pour les deux applications
— Une montée en charge optimisée pour chaque entité
— Un versionnage à part pour les deux projets

Les détails techniques de l'architecture seront exposés au cours des sections suivantes.

2.2 Technologies de base

Pour traiter efficacement l'aspect SaaS du produit, un choix judicieux des technologies a été effectué préalablement au démarrage de la phase de développement. Nous procéderons ici à la présentation des technologies de base relatives à chacune des couches de notre architecture ainsi qu'à notre environnement de développement.

Couche de présentation

Pour réussir à développer un produit SaaS de qualité, on s'est orienté vers la mise en place d'une SPA (Single Page Application) pour la partie Front-end de notre application.
Dû à sa notoriété et à notre familiarité avec lui, le framework AngularJS [24] a été notre choix de référence. L'ampleur de la communauté active développée autour de lui ainsi que l'abondance des ressources associées et le succès notoire en soi du framework nous a convaincu à l'adopter en tant que brique de base pour le développement de la couche Front-end.

Couche métier

La couche métier de notre application nécessite le développement d'un API, dont les services seront exposés à la couche supérieure.
Vu les fonctionnalités axées données (ressources) de notre application, nous nous sommes tournés vers la mise en place d'un API REST. En effet, dans notre cas, REST (REpresentational State Transfer [25] présente les avantages suivants :

— L'adoption du standard HTTP en tant que protocole de communication : son support universel assure une indépendance des technologies avec lesquelles interagit l'API et ne restreint aucunement l'évolutivité du reste des composants.
— L'aspect sans état du protocole HTTP facilite considérablement la montée en charge, considération importante à prendre en compte pour toute application de type SaaS.
— HTTP est optimisé pour l'interopérabilité avec les navigateurs (mise en cache, codes d'erreurs standards, ...) : REST représente en elle-même l'architecture du web.
— Les Web services REST sont légers, ce qui les rend adaptés à la consommation via dispositifs mobiles à ressources restreintes, exigence non fonctionnelle de notre application.
— Le support du format JSON le rend particulièrement adapté à l'interopérabilité avec le langage JavaScript, qui représente le langage de développement du Front-end.
— L'abondance des technologies permettant de développer des API REST de qualité, de manière très productive.

Vu le succès notable des projets de l'entreprise d'accueil se reposant sur le langage Java, ainsi que notre familiarité avec le langage, il a été convenu dès le départ de l'adopter pour le développement en Back-end de notre produit.
Pour l'édification de l'API REST, nous avons favorisé l'usage de technologies connues. C'est ainsi que nous avons opté pour l'emploi de la suite Spring. Véritable framework pour la conception de bout en bout de solutions logicielles complètes en Java, Spring offre une panoplie d'outils pour assister à la mise en place d'un API REST. Les technologies Spring employées seront présentées en détail au cours du chapitre IV.

Couche d'accès aux données

Pour répondre aux besoins de gestion de données de notre application, nous avons procédé à une étude approfondie des systèmes de gestion de base de données disponibles sur le marché répondant aux critères d'exigence définis par l'entreprise d'accueil. Cette étude sera exposée dans la section qui suit et aboutira à la détermination de la base de données la mieux adaptée à notre situation.

2.3 Choix du SGBD

Le choix du système de gestion de base de données est une étape cruciale pour le succès d'une application de type SaaS. L'étude suivante vise à mettre en exergue les fondations sur lesquelles s'est basé le choix de la solution retenue.

2.3.1 Exigences

Commençons tout d'abord par spécifier les critères majeurs auxquels notre système de gestion de données devra se plier :

- **Disponibilité en open-source** : la licence associée à son exploitation devra permettre un usage commercial, sans imposer de frais relatifs à son obtention ou le dévoilement du code source propre à notre application
- **Maturité** : être stable, éprouvé en milieu réel et prêt à la mise en production
- **Structuration de données** : permettre de modéliser convenablement les données à persister
- **Requêtage multidimensionnel** :être adapté au besoin principal d'aide à la décision à offrir via l'application
- **Isolation de données** : permettre la séparation efficace des données relatives à chaque organisation, en conservant une certaine aisance d'utilisation
- **Consistance de données** : garantir pour chaque organisation, à tout moment, la consistance de ses données (support des transactions intra-organisation)
- **Durabilité de données** : offrir une persistance de données sure et infaillible (mécanisme de résistance aux pannes offert, pas de perte de donnée admise)
- **Performance** : être optimisé pour la lecture tout en assurant un temps d'écriture raisonnable
- **Productivité** : offrir une documentation de qualité, plus ou moins exhaustive, et posséder des ressources communautaires utiles

Un autre aspect, relatif généralement aux applications de type SaaS, serait la :

- **Scalabilité** : support de mécanismes de montée en charge

Cependant, celui-ci n'a pas été estimé prioritaire pour le moment dû au nombre restreint d'utilisateurs prévu pour le proche futur. Le problème devrait-t-il se manifester ultérieurement, une décision plus optimale pourra être prise à ce moment pour le régler (fondamentalement, scalabilité verticale ou horizontale), compte tenu de l'envergure plus importante qu'aurait pris le projet et du nouveau contexte auquel il sera soumis.

2.3.2 Présentation des solutions étudiées

Parmi les systèmes de gestion de base de données disponibles sur le marché, quelques options ont retenu notre attention lors du choix des solutions à considérer. Étant donné le caractère open-source recherché, des solutions telles que OracleDB ou SQL Server n'ont pas été prises en compte. Les solutions étudiées se classifient essentiellement en deux catégories :

— Les bases de données relationnelles

— Les bases de données non-relationnelles

Les bases de données relationnelles ayant été considérées pendant plusieurs décennies comme le de facto pour le développement de solutions logicielles, deux solutions open-source se sont distinguées par rapport aux autres :

- **MySQL** [26] **/ MariaDB** [27] :
  C'est un système de gestion de bases de données relationnelles (SGBDR). Il fait partie des SGBDR les plus utilisés au monde, autant par le grand public (applications web principalement) que par des professionnels. Une partie de sa popularité provient de son inclusion dans les packages LAMP (Linux, Apache, MySQL/MariaDB, PHP) et ses variantes. MariaDB est une branche de MySQL maintenue par son créateur original. Celle-ci a vu le jour suite à l'acquisition de Sun (alors propriétaire de MySQL) par Oracle. À ce jour, leurs fonctionnalités restent très similaires.
- **PostgreSQL** [28] :
  PostgreSQL est un système de gestion de base de données relationnelle et objet (SGBDRO). Ce système, concurrent d'autres systèmes de gestion de base de données libres ou propriétaires, est plus avancé que ses concurrents dans la conformité aux standards

SQL et est pratiquement conforme aux diverses normes ANSI SQL. Il est largement reconnu pour son comportement stable, proche d'Oracle, mais aussi pour ses possibilités de programmation étendues, directement dans le moteur de la base de données.

D'une autre part, les bases de données non-relationnelles étant très populaires lors de la mise en place d'applications de type SaaS, les solutions suivantes ont aussi été considérées :

- **MongoDB** [29] :
  C'est un système de gestion de base de données orientée documents, répartissable sur un nombre quelconque d'ordinateurs et ne nécessitant pas de schéma prédéfini des données. Il est développé depuis 2007 par 10gen, devenue MongoDB. Le serveur et les outils sont distribués sous licence AGPL. Il fait partie de la mouvance NoSQL.
- **Cassandra** [30] :
  C'est un système de gestion de base de données (SGBD) adhérant au mouvement NoSQL, conçu pour gérer des quantités massives de données sur un grand nombre de serveurs, assurant une haute disponibilité en éliminant les points individuels de défaillance. Initialement développée par Facebook, l'application a été libérée dans l'espace open source en juillet 2008. Le projet est maintenant porté par la fondation Apache.
- **CouchDB** [31] :
  Apache CouchDB est un système de gestion de base de données orienté documents, écrit en langage Erlang, donc optimisé pour la concurrence et les environnements faibles en ressources. Il est distribué sous licence Apache. Conçu pour le Web, il fait partie de la mouvance NoSQL, et a été construit de manière à pouvoir être réparti sur de multiples serveurs.

Il est à noter que les solutions, dites "NoSQL", orientées clé-valeur n'ont pas été considérées, vu le manque de fonctionnalités (intentionnel) généralement offert. De même, la complexité des solutions orientées graphes ne parait pas justifiée face aux besoins de l'application.

2.3.3 Comparatif

Le Tableau III.1 résume globalement la compatibilité des solutions considérées avec les spécifications établies précédemment. On emploiera l'échelle illustrée par la figureIII.3 pour comparer la richesse du support relativement à la fonctionnalité examinée : de l'absence de support ou d'incompatibilité (couleur la plus froide), jusqu'à un support optimal ou une richesse élevée en termes de granularité du support (couleur la plus saturée).

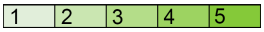

Figure III.3 – Échelle associée au tableau III.1

Tableau III.1 – Comparatif des SGBD considérés

		MySQL	MariaDB	PostgreSQL	MongoDB	Cassandra	CouchDB
Maturité	Ancienneté	1995		1996	2009	2008	2005
	Dernière version stable	5.7.17 - 12/2016	10.1.21 - 01/2017	9.6.2 - 02/2017	3.4.2 - 02/2017	3.10 - 02/2017	2.0.0 - 09/2016
	Popularité [1]	5		5	5	5	4
Stockage	Modèle de données	Entité-Relation		Entité-Relation	Document	Table Partitionnée	Document
	Hiérarchie	BDD, Table, Ligne, Colonne		BDD, Schéma, Table, Ligne, Colonne	BDD, Collection, Document	Keyspace, Table, Partition, Ligne, Colonne	BDD, Document
	Support schéma prédéfini	Niveau Table - imposé		Niveau Table - imposé	Niveau Collection (via validation) - optionnel	Niveau Table - imposé [2]	Niveau Document (via validation) - optionnel
	Support schéma dynamique	Type de donnée JSON		Type de donnée JSON	Document JSON	Ø	Document JSON
	Support d'index secondaire	Niveau Table - multi colonne		Niveau Table - multi colonne	Niveau Collection ou Document - multi clé	Niveau Table - uni colonne	Niveau BDD (via vue) - multi valué
Granularité des permissions		Colonne, Vue		Colonne / Rangée, Vue	Collection	Table	Base de données
Requêtage	Support jointure	inter-table		inter-table	inter-collection [3]	Ø [4]	Ø [4]
	Aggrégation	Groupement (fonctions d'aggrégation)		Groupement (fonctions d'aggrégation)	Groupement (fonctions d'aggrégation), Map-Reduce sur collection	Groupement (fonctions d'aggrégation)	Map-Reduce sur BDD (via vue)
	Insertion/Màj par lot	Transactionnables		Transactionnables	Insertions niveau Collection, Non transactionnable	Atomiques, Màj Isolée, Insertion isolée niveau Partition	Transactionnables
Conformance ACID	Atomicité	Transactions supportées		Transactions supportées	Niveau Document (niveau 1)	Niveau Partition	Niveau Document (niveau 1)
	Consistance	Transactions supportées		Transactions supportées	Transactions non supportées	Transactions non supportées	Transactions non supportées
	Isolation	Niveau BDD		Niveau BDD	Niveau Collection [5]	Niveau Ligne	Niveau Document (niveau 1)
	Durabilité	Tolérance aux pannes		Tolérance aux pannes	Tolérance aux pannes	Tolérance aux pannes	Tolérance aux pannes
Productivité	Ressources communautaires	4		4	4	4	3
	Documentation	5		5	5	2	5
Scalabilité	Disponibilité [6]	NBD Cluster	Galera	Postgres-XL	Natif	Natif	Natif
	Topologie	Multi	master-master	master-master	master-slave	master-master	master-master
	Réplication	Oui	Oui	Oui	Oui	Oui	Oui
	Partitionnement (Sharding)	Oui	Oui	Oui	Oui	Oui	Oui
	Consistance	Forte ou Éventuelle [7]	Forte ou Éventuelle [7]	Forte ou Éventuelle [7]	Forte ou Éventuelle	Forte ou Éventuelle (granulairement configurable)	Éventuelle

Notes :

1. Nous nous reposons sur la classification effectuée par le site de référence db-engines.com [32] pour l'analyse de la côte de popularité. La note maximale (5) est attribuée au SGBD du Top 10, secondée par une note inférieure (4) pour ceux du Top 20.
2. Cassandra ne supporte plus les colonnes dynamiques (rendu obsolète par le remplacement de l'interface Thrift par CQL) [33].
3. Les jointures sous MongoDB ne sont pas supportées pour les données partitionnées.
4. Cassandra et CouchDB ne supportent pas nativement les jointures. Cette fonctionnalité est cependant atteignable avec SparkSQL ou le connecteur ODBC DataStax pour Cassandra. Le concept de vue sous CouchDB permet de réaliser un résultat comparable aux jointures, via des fonctions map-reduce.
5. L'isolation sous MongoDB n'est pas supportée pour les données partitionnées.
6. Pour les SGBDR, des solutions de montées en charge populaires ont été mis en avant à titre indicatif pour le comparatif.
7. La consistance (forte ou éventuelle) est gérée par le niveau d'isolation employé pour les SGBDR.

Les performances ne peuvent être concrètement évaluées que sous un benchmarking élaboré, propre aux besoins de notre application. Ce critère n'a en conséquent pas été adressé dans le comparatif. Il reste à noter que toutes les solutions offrent des performances raisonnables sous des conditions normales d'utilisation d'après les retours de leurs communautés respectives.

2.3.4 Le SGBD retenu

On peut déduire du tableau que les solutions relationnelles supportent davantage les fonctionnalités à offrir à nos clients, étant hautement compatibles avec les besoins d'aide à la décision et de requêtage multidimensionnel. C'est surtout le support robuste des transactions et des jointures qui les favorisent par rapport aux autres solutions.
S'étant décidés à adopter une solution relationnelle, le choix final s'est porté sur PostgreSQL.

Sa richesse en fonctionnalités et son support concret des schémas au sens SQL du terme, pour l'isolation des données, le place en tête par rapport à MySQL/MariaDB.
De plus, l'avenir de MySQL et de son alter-ego MariaDB, restant incertain quant à leur divergence éventuelle en fonctionnalités pour les versions futures, les pénalisent par rapport à PostgreSQL.
Notons toutefois que le développement de la couche d'accès aux données sera réalisé à l'aide d'un middleware (ORM) qui permettra d'abstraire la base de données cible pour les déploiements internes, prévus de par la mise à disposition d'une offre alternative à l'offre SaaS (i.e on-premises). Dans un tel cas, une solution telle que MySQL pourra faire l'affaire malgré son manque de support concret des schémas, vu que le déploiement se fera pour une seule base de données et que ce dernier traite les schémas en tant que bases de données de manière sous-jacente.

Conclusion

Ce chapitre a été l'occasion de prendre du recul par rapport au travail d'implémentation à réaliser. Les besoins de l'entreprise ont été formalisés et exprimés de manière explicite. Les fonctionnalités requises du produit final identifiées, une étude technique a été entreprise en vue de poser les bases pour le travail de développement à entreprendre.
Le dénouement de la phase d'inception annonce le début de la phase de construction. Cette phase sera détaillée en long et en large au cours des prochains chapitres.

Chapitre IV

GESTION DE PROJET : API

Plan

Introduction

Ce chapitre représente une entrée en matière dans la phase de construction de notre produit. Après avoir spécifié la couche d'accès aux données du système, nous poursuivons la présentation du Back-end en nous attaquant à la couche supérieure, à savoir la couche métier. Le développement de la couche métier consiste à concevoir un API REST fournissant un accès supervisé aux données recueillies par notre application. Nous commencerons par aborder l'aspect conceptuel avant de rentrer dans les détails techniques.

1 Conception

1.1 Analyse fonctionnelle

La figure IV.1 présente le diagramme de cas d'utilisation raffiné du cas d'utilisation "Gérer un projet" et du reste des cas d'utilisations liés à la gestion de projet, présentés dans le chapitre

II - 1.5 (p.34). Nous procéderons dans la partie qui suit à la clarification des cas d'utilisation élémentaires modélisés.

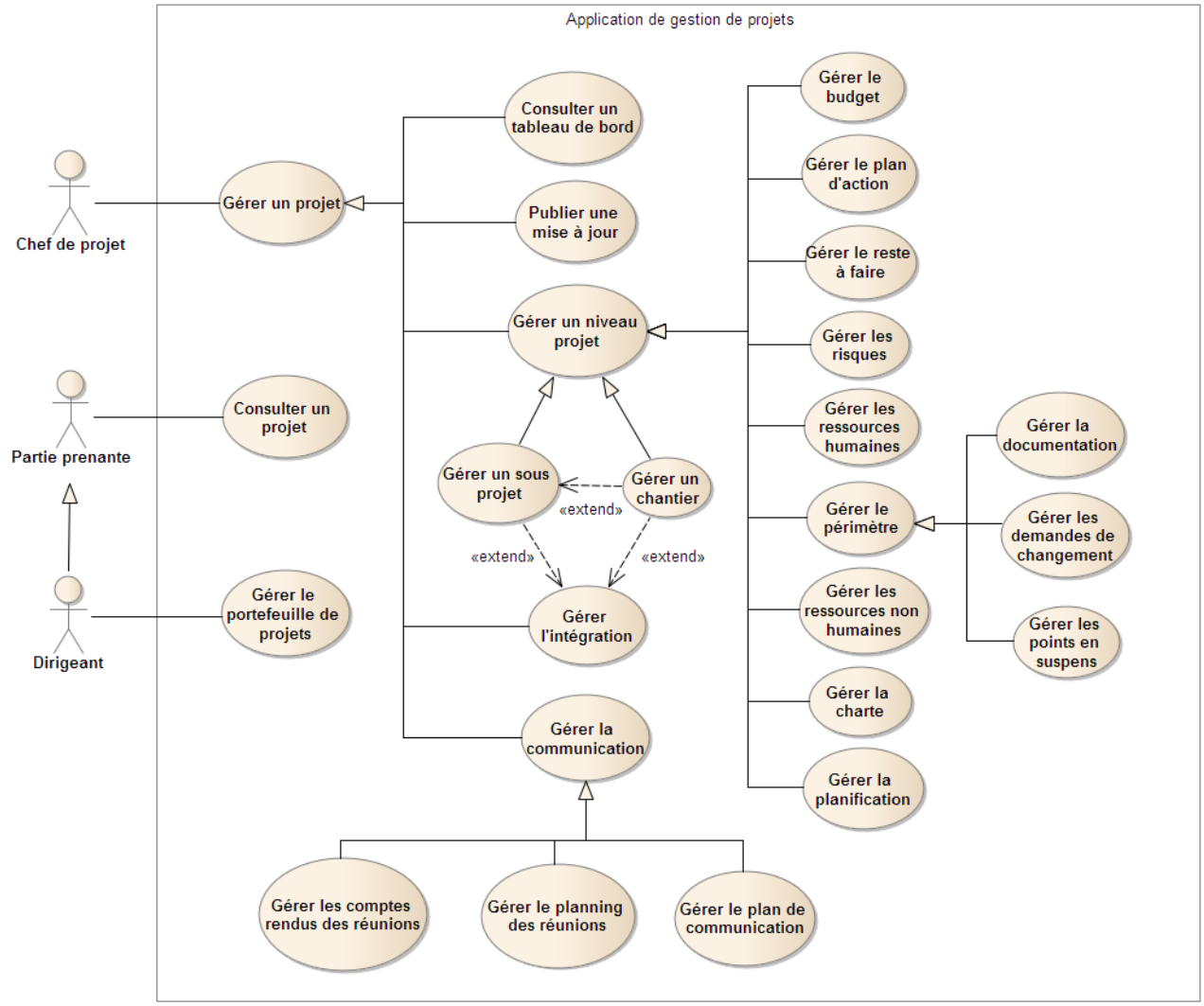

Figure IV.1 – Diagramme de cas d'utilisation de l'aspect gestion de projets de l'application

La gestion du portefeuille de projets consiste à modifier la liste des projets crées par le détenteur de l'application. La gestion est réalisée via l'application d'opérations CRUD (Create-Read-Update-Delete ou Ajout-Consultation-Édition-Suppression) sur les projets existants.

La gestion d'un projet est un cas abstrait qui peut se concrétiser grâce à la gestion de différentes parties d'un projet. Au sein de l'application, un projet est défini par un ensemble de ressources permettant de gérer différents aspects de la gestion professionnelle de projet :

— **Un plan budget** : celui-ci inclut la définition du budget initial, du budget consommé et d'une estimation du budget qui reste à consommer pour le projet.
— **Une charte** : la charte d'un projet rassemble l'ensemble des informations de base du projet (voir chapitre II - 1.1 p.19).
— **Des actions** : les actions représentent les tâches à accomplir pour réaliser le projet.
— **Des reste-à-faire** : un reste-à-faire correspond à une définition haut niveau de tâches à accomplir pour réaliser les objectifs du projet. Il est généralement concrétisé par la définition d'une ou plusieurs actions.
— **Des ressources humaines** : une ressource humaine correspond à un employé interne à l'entreprise cliente ou à un intervenant externe. La définition de ces ressources permet de déterminer précisément les ressources humaines en relation avec le projet en cours. Celles-ci sont souvent assignées en tant que superviseurs à d'autres éléments du projet.
— **Des ressources non humaines** : ces dernières peuvent consister en des ressources matérielles ou immatérielles, nécessaires à l'exécution du projet. Leur recensement permet d'obtenir une vue d'ensemble des ressources déployées pour la mise en œuvre du projet et fournit également une indication sur la répartition de l'investissement relatif au projet.
— **Des jalons** : la planification est essentiellement réalisée par la définition de jalons, décrivant les étapes majeures d'implémentation du projet.
— **Des documents** : la documentation du projet est achevée grâce au téléchargement de documents divers.
— **Des demandes de changement** : une demande de changement représente une requête formelle de modification d'un ou plusieurs éléments du projet.
— **Des points en suspens** : un point en suspens décrit un problème non résolu, rencontré lors de l'exécution du projet.
— *Des sous niveaux* : la gestion de l'intégration doit répondre aux besoins de structuration du projet (voir chapitre II - 1.1 p.19). Par suite, les deux niveaux suivants sont définis :
  - **Des chantiers** : un chantier représente une coalition consistante d'activités du projet. L'exécution d'un chantier est dans l'ensemble très similaire à celle d'un projet. Exception faite de la charte, un chantier peut contenir à lui-même chacun des élé-

ments susmentionnés précédemment.

- **Des sous-projet** : un sous-projet représente un chantier de plus grande envergure. Conséquemment, il peut lui-même contenir un ou plusieurs chantiers pour sa propre structuration. Il est géré de manière très similaire à un projet et comprend une charte, assimilable à celle d'un projet.

— *Des ressources utiles à la gestion de la communication* : pour gérer la communication projet (voir chapitre II - 1.1 p.19) les ressources suivantes sont mises à disposition :

- **Un plan de communication** : il permet de définir les modalités de communication par l'ajout d'éléments au plan. Chaque élément est défini par une description ainsi que le responsable de la supervision de ce dernier.
- **Un planning des réunions** : le planning est constitué d'une planification formelle de chaque réunion du projet, caractérisée par un nom, une date et un lieu ainsi que le statut de la réunion en question (annulée, clôturée, ...).
- **Des comptes rendus de réunion** : il s'agit de documents téléchargeables, chacun résumant les points pertinents d'une réunion ayant pris place dans le cadre du projet (P.V.).

Concrètement, chacune de ces ressources est gérée par l'exécution d'opérations CRUD sur la ressource en question.

Outre ces dernières, la supervision d'un projet est également assurée grâce à la consultation de tableaux de bord. Les niveaux projet et sous-projet possèdent chacun un tableau de bord très similaire. Afin d'éviter d'introduire des incohérences visibles au reste des parties prenantes du projet, dû à sa gestion en temps réel, le chef de projet peut publier des mises à jour à destination des parties prenantes. Celles-ci sont ainsi restreintes à ne consulter que les mises à jour logiques, validées par le chef de projet.

1.2 Modélisation dynamique

Le comportement général souhaité de notre API est modélisé à l'aide d'un diagramme de séquence système, illustré à la figure IV.2. Celui-ci représente la sollicitation par l'extérieur, d'une fonctionnalité CRUD sur une ressource exposée par ce dernier.

En effet, les fonctionnalités de gestion de projets à offrir à nos utilisateurs peuvent se résumer dans l'ensemble à un requêtage de ressources telles des projets, des sous-projets, la charte d'un projet, des risques, des actions, des demandes de changement, etc.

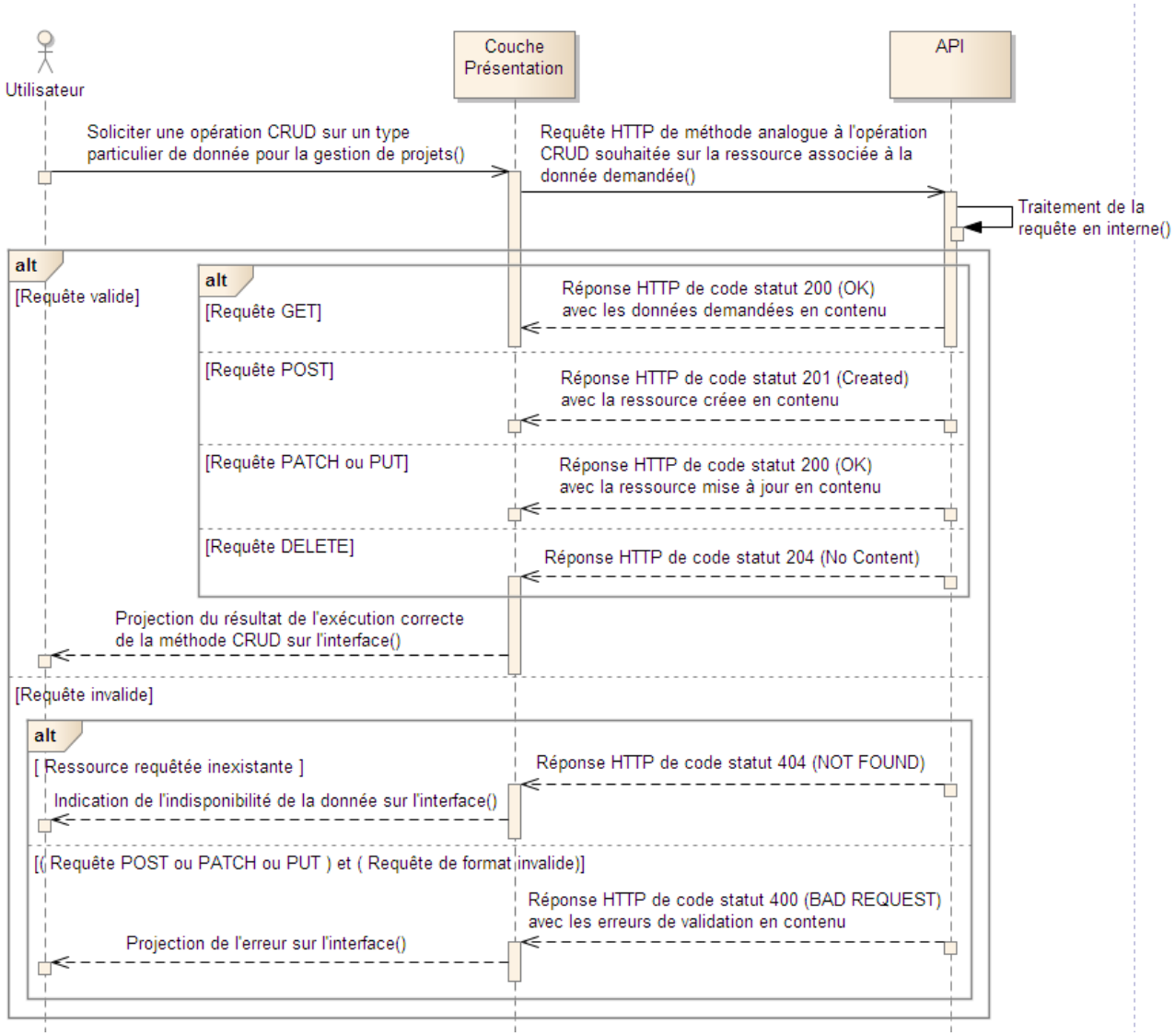

Figure IV.2 – Diagramme de séquence système de l'application de gestion de projets niveau API

1.3 Modélisation statique

Pour la conception d'un API REST, il est essentiel de déterminer au préalable le modèle de données à adopter afin de réussir le mapping objet-ressource caractéristique d'un modèle REST. Le modèle du domaine établi est illustré à la figure IV.3.

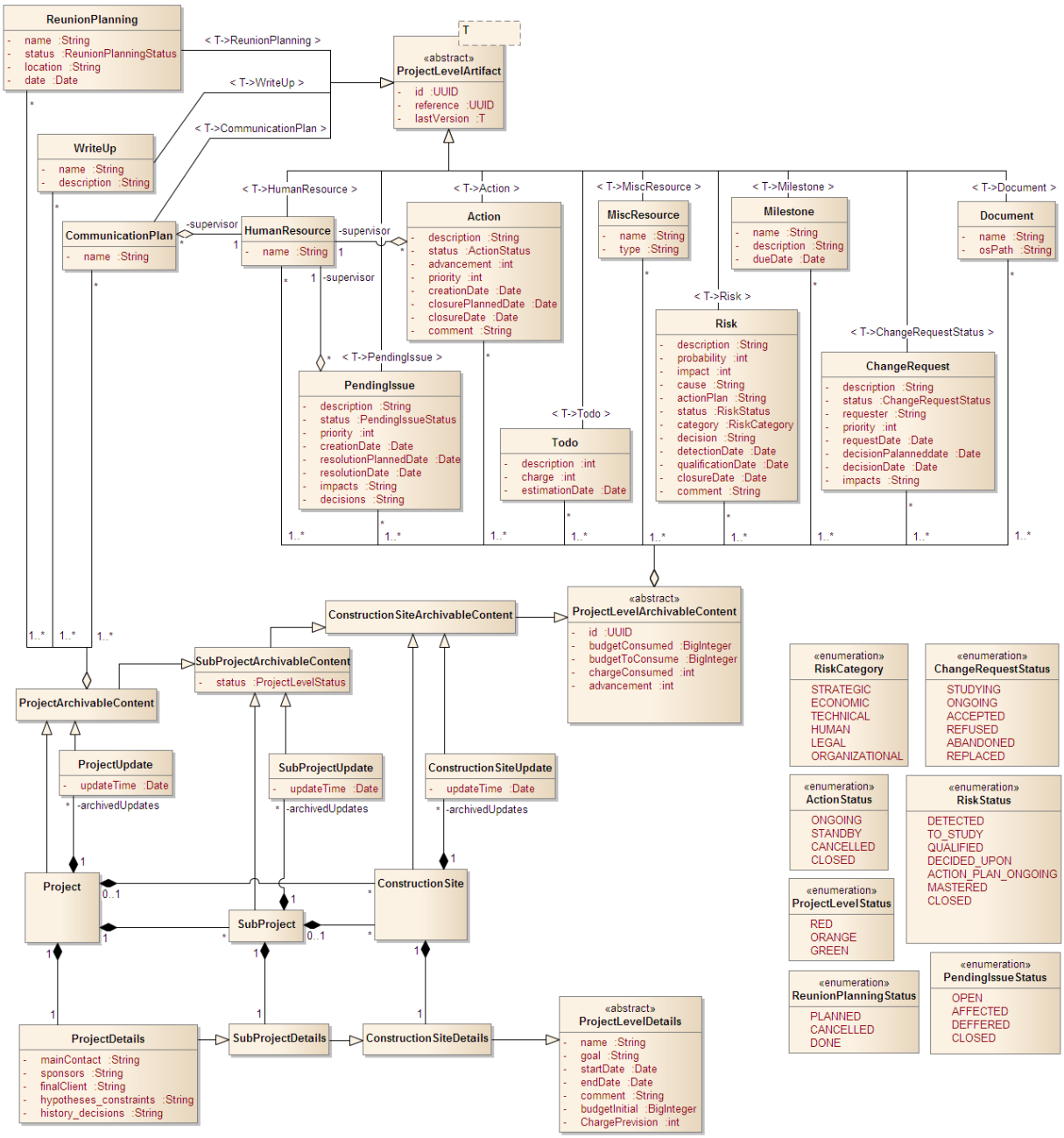

Figure IV.3 – Modèle du domaine de l'application de gestion de projets niveau API

Notre modèle s'articule autour de trois niveaux de projet distincts (ProjectLevels) :

— Projet (Project)

— Sous-projet (SubProject)

— Chantier (ConstructionSite)

Une relation d'héritage logique est déductible : un projet hérite des propriétés d'un sous-projet, qui lui-même hérite des propriétés d'un chantier.

Chaque niveau de projet possède un contenu catégorisé en deux types : archivable et non archivable, le contenu archivable étant destiné à être archivé sous forme de mise à jour (Update), identifiable par l'instant précis de celle-ci.
Cette catégorisation impose d'externaliser le contenu archivable en une classe à part, extensible, de manière à en dériver les entités des mises à jour de chaque niveau. Conséquemment, dû à la distinction entre le contenu des mises à jour pour chaque niveau de projet ainsi qu'au rattachement direct de ces mises à jour aux entités niveaux de projet, ces niveaux ne peuvent hériter directement les uns des autres. En résumé, chaque niveau est contraint d'hériter du contenu archivable d'une part, partagé avec les entités mise à jour, et du contenu non archivable d'une autre part. Vu l'indisponibilité de l'héritage multiple dans notre cas (langage d'implémentation Java) nous ne pouvons définir d'entité qui hérite des deux contenus directement. Pour pallier à cette limitation, en plus de l'héritage, nous avons recours à la composition :
En effet, en plus d'hériter d'un contenu archivable via les classes suffixées par "ArchivableContent", chaque niveau de projet se compose explicitement d'un contenu non archivable, représenté par les classes suffixées par "Detail" (une classe pour chaque niveau de projet). Nous réussissons ainsi à contourner les limites imposées et à modéliser l'héritage multiple alternativement via l'héritage des composés, en dérivant chaque entité du contenu non archivable (Detail) de l'entité similaire, respective au niveau projet inférieur.

Le contenu des niveaux de projet en soi peut être directement associé aux spécifications fonctionnelles telles que définies dans le chapitre précédent (section 1.3 p.31). Le contenu archivable se compose essentiellement d'artéfacts projet (ProjectLevelArtifact). Énumérons-les :

— Les documents (Document)
— Les demandes de changement (changeRequest)
— Les jalons (Milestone)
— Les risques (Risk)
— Les ressources humaines (HumanResource)
— Les autres ressources (MiscResources)
— Le reste à faire (Todo)
— Les actions (Action)
— Les points en suspens (PendingIssue)
— Les plans de communication (CommunicationPlan)
— Les comptes-rendus de réunions (WriteUp)
— Les planifications de réunions (ReunionPlanning)

Voici quelques remarques complémentaires sur le modèle :

— Le champs lastVersion de ProjectLevelArtifact est utilisé pour l'optimisation de l'espace occupé par les données archivées. En effet, si un artéfact n'a reçu aucune modification depuis la dernière mise à jour (vérifiable grâce à ce champs), il est tout simplement réutilisé tel quel lors du référencement des artéfacts de la nouvelle mise à jour, ce qui évite les duplications et nous rapporte un gain en espace de stockage.
— Le champs reference de ProjectLevelArtifact est utilisé pour garder une trace de l'historique de mises à jour d'un artéfact. Outre l'id, qui identifie la version de l'artéfact, la référence permet d'identifier de manière unique toutes les versions archivées d'un même artéfact.
— L'existence de classes dérivées ne spécifiant aucune propriété supplémentaire par rapport à leur classe mère au moment présent est justifié par la préservation de la logique générale du modèle (niveaux de projets distincts) et le garantissement d'une évolutivité aisée de la solution quant à cet aspect.
— La définition des classes abstraites ProjectLevelArchivableContent et ProjectLevelDetails, qui ne se différencient pas en contenu des classes ConstrcutionSiteArchivable-

Content et ConstructionSiteDetail, respectivement, est utile pour instaurer une certaine logique lors du maniement du code pour le désignement d'un niveau de projet quelconque. Se reposer sur les classes relatives à un chantier peut prêter à confusion lors de la compréhension et a en conséquence été évité.

2 Réalisation

Nous aspirons ici à fournir une vue détaillée du travail de réalisation effectué pour implémenter notre API. Nous présenterons brièvement les technologies et les concepts exploités dans le processus de développement avant de spécifier les étapes majeures de son déroulement.

2.1 Technologies et Patterns employés

Pour réaliser l'API REST avec notre langage de prédilection, Java, nous nous sommes reposés sur un nombre important de technologies.

Spring

La suite Spring nous a été indispensable tout au long du développement de l'API, en particulier nous avons utilisé les éléments suivants :

— **Spring Framework** [34] : C'est un conteneur léger pour l'injection de dépendance, la gestion de transactions, le développement web, la gestion de l'accès aux sources de données, et bien d'autres. Il se repose grandement sur l'usage de la programmation orientée aspect. Le composant Spring Web MVC nous sera particulièrement utile dans notre cas au vu de son support natif du développement d'API REST.

— **Spring Data** [35] : Ce projet a pour mission de fournir une manière consistante et familière à Spring pour l'accès à diverses sources de données. Il se repose sur l'utilisation extensive du pattern Repository.

— **Spring Data REST** [36] : il se repose sur Spring Data pour fournir une démarche simplifiée du développement d'API REST, en nous dispensant des tâches rébarbatives

relatives à l'écriture du code d'implémentation de base, nécessaire pour l'exposition de ressources, et en simplifiant l'extension.

— **Spring Boot** [37] : Cet outil nous permet d'adopter son approche dogmatique au développement d'applications Spring. Le gain en productivité associé à son usage provient de l'adoption de conventions (configurations par défaut) au lieu de configuration manuelle explicite.

Nous nous reposerons ainsi essentiellement sur la programmation orientée aspect, POA (ou AOP, pour Aspect Oriented Programming) et le pattern Repository pour la réalisation de notre API. Nous utiliserons aussi intensivement l'injection de dépendance (ou DI, pour Dependency Injection) mise à disposition.
Présentons brièvement ces concepts :

Pattern Repository [38]

Le pattern Repository est l'un des pattern définis par Martin Fowler dans son livre sur les patterns d'architecture d'applications d'entreprise [39]. Il consiste en la mise en place d'un médiateur entre le modèle du domaine et la couche d'accès aux données, à l'aide d'une interface abstractant l'accès aux objets du domaine, sous forme de collections.

Dependency Injection [40]

L'injection de dépendances est un mécanisme qui permet d'implémenter le principe de l'inversion de contrôle. Ce dernier est un pattern architectural générique, commun aux frameworks, qui préconise de transférer le contrôle du flot d'exécution d'un logiciel, de l'application elle-même vers le framework (ou la couche logicielle sous-jacente). L'injection de dépendance en soi vise à réduire les dépendances explicites entre les composants d'un logiciel. Ainsi les dépendances entre composants logiciels ne sont plus exprimées dans le code de manière statique, mais déterminées dynamiquement à l'exécution.

Programmation Orientée Aspects [41]

La Programmation Orientée Aspect est un paradigme de programmation qui introduit une nouvelle manière de structurer, et donc de programmer, les applications. Elle peut être appliquée sur les langages de programmation procéduraux et orientés objet. La POA appliquée à la Programmation Orientée Objet (POO, ou Object Oriented Programming, OOP) vise à pallier principalement à deux limitations de l'usage individuel de la POO : l'entremêlement de la logique métier à la logique des fonctionnalités transversales, ainsi que la dispersion de code. La figure IV.4 illustre la différence et met en relief l'apport de l'AOP à la POO.

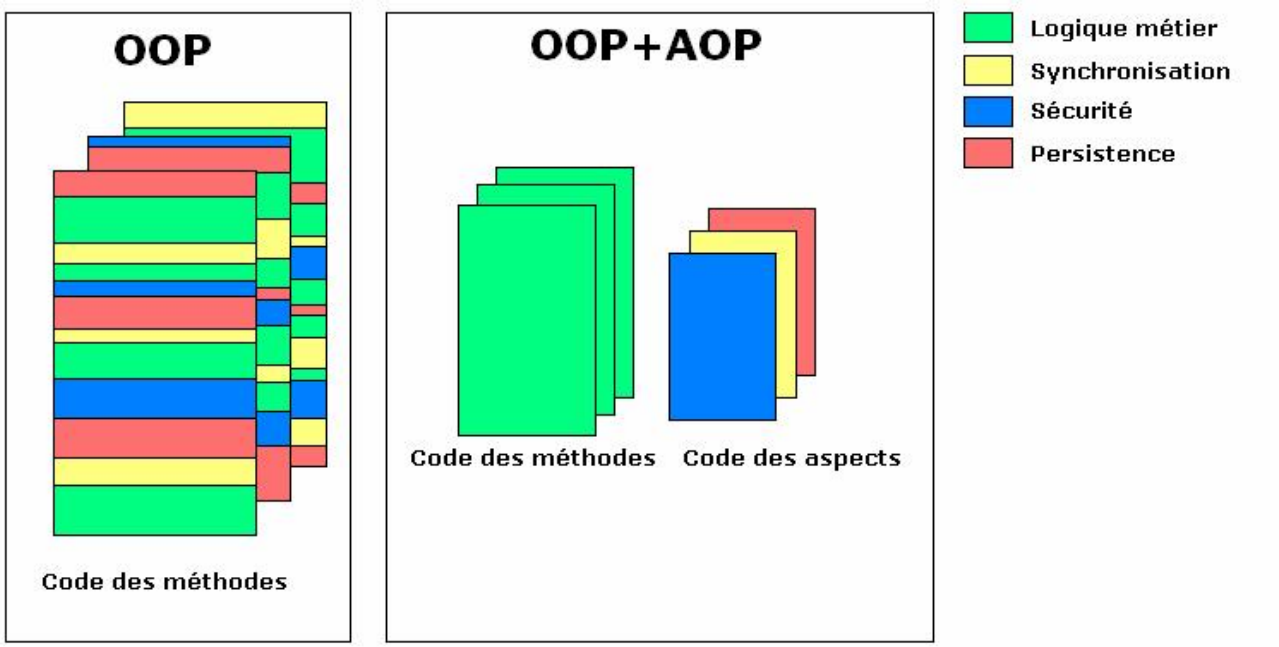

Figure IV.4 – [2] Comparaison entre la structuration du code avec la POO et la POA+POO

Il en résulte clairement un découplage de la logique fonctionnelle de notre application de la logique transversale relative aux aspects tels que la sécurité, la persistance, etc. Le code résultant est découpé en classes à proprement dites et en aspects ; les classes n'ayant aucune conscience de l'existence des aspects. C'est à la charge d'un tisseur d'aspects de greffer les aspects au reste des classes d'un programme.

D'autres technologies ont été utiles pour la mise en place de l'environnement de développement. En particulier, nous avons fait usage des outils suivants :

ORM

Pour l'interaction avec notre couche d'accès aux données, nous avons opté pour l'utilisation du mapping objet-relationnel ou **ORM** (Object Relational Mapping) via l'usage des interface fournies à cet effet par **JPA** (Java Persistance API [42]). Une fois définie sur notre modèle, la correspondance entre le monde objet et relationnel nous aide à rester focalisés sur la logique métier sans nous soucier des rouages propres internes de la base de données.
Spring Boot nous aide sur ce point en nous offrant par défaut d'utiliser l'outil ORM **Hibernate** [43], implémentation de JPA. Nous n'utiliserons aucune fonctionnalité spécifique au framework (i.e. hors JPA) pour le développement de la présente partie.

Serveur d'application

Il reste indispensable de choisir un serveur d'application pour exposer notre application via le web. Une fois encore, Spring Boot vient à notre secours en nous offrant par défaut d'exécuter notre application en sein d'un serveur **Tomcat** intégré. Tomcat [44] est en soi un conteneur web JEE [45], libre, issu de la fondation Apache. Son intégration par défaut signifie qu'une instance de ce serveur est lancée conjointement à chaque exécution de notre application. Cette disposition par défaut se révèle particulièrement utile dans notre cas, à savoir la mise en place d'une application SaaS, vu l'aspect sans état souhaité de notre application. Il suffit ainsi simplement de lancer une nouvelle instance de notre application pour réaliser la montée en charge de ce tiers (en plus d'un routage trivial entre ces instances).

Outil de build

Nous avons opté pour **Gradle** pour les besoins de build de notre application. Gradle [46] est un système open source d'automatisation de build qui s'appuie sur les concepts d'Apache Ant [47]et Apache Maven [48]. Il utilise un langage spécifique au domaine (Domain Specific

Langage) basée sur Groovy au lieu de la forme XML utilisée par Apache Maven pour déclarer la configuration du projet, ce qui le rend beaucoup moins verbeux. Il prend en charge les build incrémentaux en déterminant de manière intelligente quelles parties du processus de build sont à jour, évitant ainsi de réexécuter les tâches dépendantes de ces parties. Il a été conçu pour les build multiprojets. Ce dernier point se révélera particulièrement utile, plus tard, pour la réutilisation de code lors de la modularisation de notre projet.

2.2 Implémentation

Le processus d'implémentation s'est déroulé en un ensemble d'étapes successives :

— **Définition des POJO** (Plain Old Java Objects) : celles-ci représentent le modèle de notre domaine via la définition de classes simples à accesseurs publiques pour les attributs.

— **Mappage objet-relationnel** : à l'aide d'annotations JPA, nous spécifions nos entités, qui représenteront les données persistées via la couche d'accès aux données, et nous fournissons par la même les lignes directrices du schéma à adopter pour leur persistence.

— **Mise en place des repositories** : grâce à Spring Data, nous définissons simplement les interfaces qui donneront naissance à nos repositories (implémentations du pattern Repository) plus tard, via l'injection de dépendances offerte Spring. Ce travail est facilité par la dérivation des ces interfaces d'interfaces de base fournies automatiquement par Spring Data.

— **Intégration** : Nous fournissons ici quelques informations de base à Spring Boot (source d'accès aux données, port d'écoute serveur, ...). Celui-ci se charge de configurer automatiquement le reste de l'application et de nous fournir notre serveur d'application. À ce point, notre API de base est déjà opérationnel.

— **Extension de l'API de base** : les opérations complexes comme l'archivage de mise à jour et la préparation des données de reporting nécessitent un travail supplémentaire pour l'implémentation. Le résultat désiré a été atteint grâce à l'exploitation de deux types de composants :

- *Contrôleurs* : les contrôleurs représentent le C du paradigme MVC (Modèle-Vue-Contrôleur), qui préconise de découpler la logique métier de la présentation. Dans notre cas (API REST), la gestion de la présentation ne se déroule pas à ce niveau. C'est par le biais du composant Spring Web MVC que nous prenons avantage de la POA pour définir nos contrôleurs personnalisés. Ainsi, notre modèle déjà en place, nous définissons notre logique d'archivage dans une classe dédiée et nous étendons les contrôleurs natifs, fournis par Spring Data REST lors de l'exécution, dans le but d'assurer l'aspect transactionnel de modification de ressources de notre API, indisponible par défaut pour les ressources complexes (toutes les opérations doivent être accomplies en une seule requête).
- *Projections* : Spring Data REST met en avant le concept de projection, qui permet pour l'une des ressources de l'API REST engendré par ses soins, d'en restreindre le contenu à certaines propriétés, ou bien de l'enrichir par des propriétés virtuelles. Ceci est achevé grâce à la définition d'une interface pour chaque type de projection. C'est le langage SpEL (Spring Expression Langage) qui permet de spécifier un comportement particulier aux méthodes de l'interface. Ceci va nous permettre d'offrir des données personnalisées logiquement associées à nos ressources, accessibles via un paramètre de requête pour la ressource en question. Nous en verrons un exemple concret dans la section Tests et Validation.

— **Validation des données en entrée** : Nous prenons encore une fois avantage de la POA ici, en définissant des validateurs pour les ressources de l'API. Ceci est réalisé à l'aide d'auditeurs d'événements (Event Listeners) fournis par Spring data REST, dédiés à la tâche. Pour chaque ressource, nous spécifions les conditions d'échec de sa validation et les messages d'erreurs associés. Nous spécifions en plus le formatage global souhaité pour les réponses d'échec de validation.

Le diagramme de packages de la figure IV.5 permet d'acquérir une vue d'ensemble du travail réalisé jusqu'à présent :

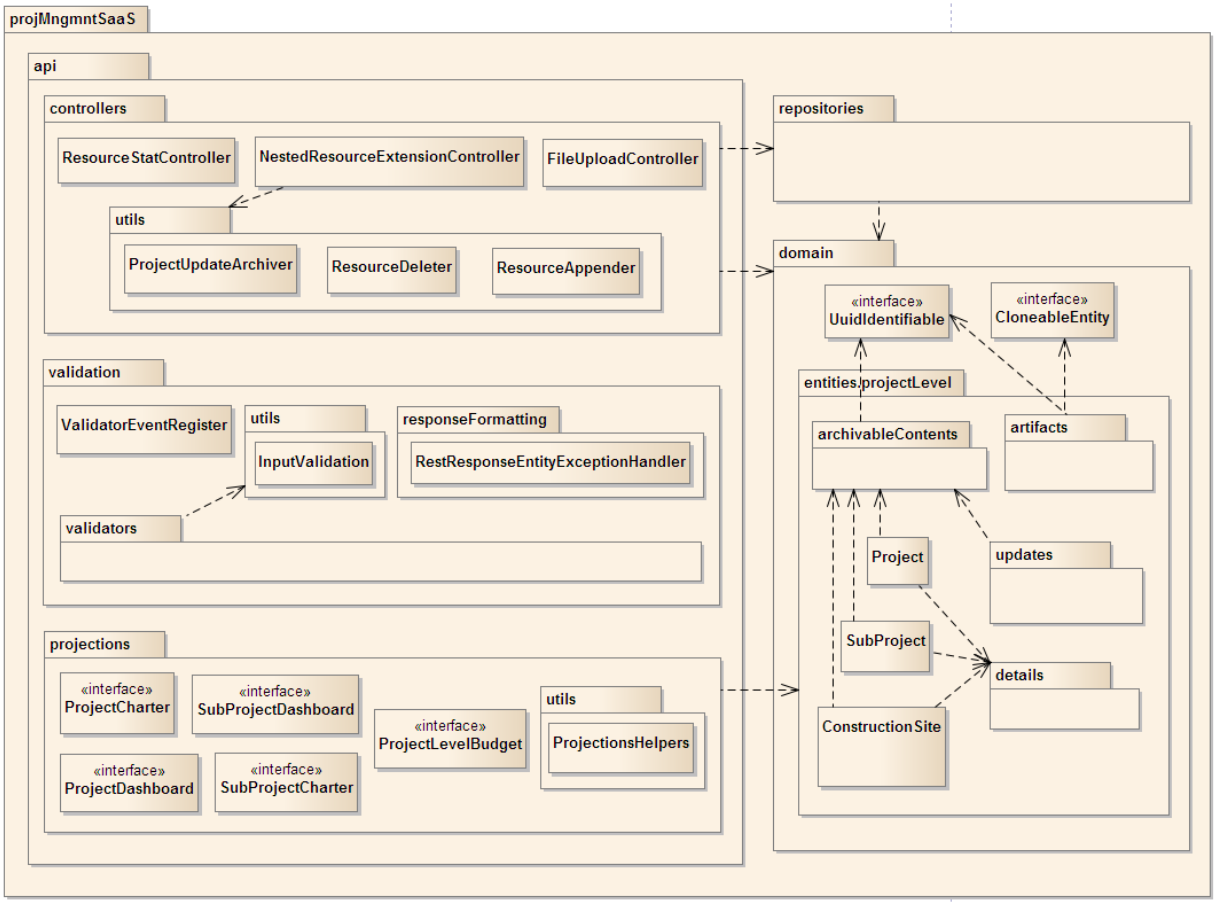

Figure IV.5 – Diagramme de packages de l'application de gestion de projets niveau API

Dans un soucis de clarté, le contenu des packages repositories, validators, artifacts, details, archivableContents et updates a été omis du diagramme. Concrètement, le package repositories contient une interface pour chacune des entités exposées via l'API (ProjectRepository, SubProjectRepository, RiskRepository, ...). Elles héritent toutes d'une interface générale pour l'implémentation du pattern Repository. Le package validators contient pareillement une classe de validation pour chacune des entités exposées via l'API. Le contenu du reste des packages omis consiste pour chacun d'entre eux d'une classe pour chaque niveau de projet, ainsi que d'une classe abstraite générale dont elles héritent. Le diagramme du modèle du domaine à la figure IV.3 les présentent plus en détail. Le rôle du contrôleur FileUploadController est d'étendre les fonctionnalités du contrôleur natif offert par Spring Data REST pour le support du téléchargement de documents.

3 Tests & Validation

Pour valider le bon fonctionnement de notre API, nous avons pris le soin de réaliser un ensemble de tests fonctionnels visant pour chaque ressource de l'API REST à :

— vérifier son accessibilité par le requêtage de l'URI associé
— vérifier la validité du contenu de la réponse après requêtage
— vérifier le rejet d'une requête invalide et le bon formatage de la réponse d'erreur
— vérifier le bon fonctionnement des requêtes paramétrées s'il existe une projection pour la ressource en question

Nous avons utilisé l'outil **Postman** [49] pour réaliser l'ensemble de nos tests. Postman est un client HTTP avancé pour le test d'API REST. Il facilite le test, le développement et le traitement des API en permettant de mettre en place rapidement des requêtes HTTP simples ou complexes. Ses fonctionnalités incluent le choix de la méthode HTTP de requêtage, l'édition du contenu de la requête (utile pour les requêtes POST), la manipulation des entêtes des requêtes HTTP, la préservation de Cookies, la sauvegarde de requêtes et leur regroupement en collections ainsi que la mise à disposition d'une interface intuitive fournissant la configuration simple des requêtes (champs clés-valeur pour les entêtes et les paramètres de requête, liste déroulante pour le choix de la méthode HTTP et le type du contenu de la requête, ...).

Les scénarios de tests suivants sont fournis en annexe (Annexe A) :

— le requêtage simple de ressources (via la méthode GET)
— le requêtage paramétré de ressources (supportant les projections)
— l'ajout de ressources (via la méthode POST)
— la redéfinition de ressources (via la méthode PUT)
— l'édition partielle de ressources (via la méthode PATCH)
— la suppression de ressources (via la méthode DELETE)
— l'archivage de l'état d'un projet (création d'une mise à jour)

Conclusion

La mise en place de l'API tel que présenté dans ce chapitre nous a permis de valider les services de gestion de projets à fournir à la couche de présentation. Il est indispensable cependant de compléter ce travail par la supervision de l'accès aux données de cet API. Nous poursuivrons la validation de l'aspect métier avec la construction de la couche de présentation et la réalisation d'une application opérationnelle minimale couvrant l'ensemble des fonctionnalités de gestion de projet dans le chapitre qui suit.

Chapitre V

GESTION DE PROJET : COUCHE PRÉSENTATION

Plan

Introduction

La phase de construction se poursuit avec le développement de la couche supérieure : la couche de présentation. Celle-ci représente la partie visible de notre application. C'est uniquement avec elle qu'interagiront nos utilisateurs. Par conséquent, l'ensemble des fonctionnalités de l'API précédemment introduit doit être concrétisé par des interfaces opérationnelles permettant d'implémenter les spécifications fonctionnelles de notre application. Ce chapitre débutera avec une modélisation conceptuelle du comportement de l'interface utilisateur dédiée à la gestion de projets. Nous passerons ensuite à la partie technique, avec une introduction à l'environnement de développement et des technologies exploitées pour sa réalisation. Ce chapitre sera conclu par la validation des spécifications fonctionnelles avec la présentation d'interfaces graphiques répondant à l'ensemble des besoins de gestion de projet exprimés.

1 Conception

Nous procédons ici à la description du comportement global de l'application Front-end grâce à des diagrammes d'états-transitions. La figure V.1 présente le diagramme d'états-transitions global de notre application. Il présente principalement les types de pages mises à disposition pour la gestion de projets.

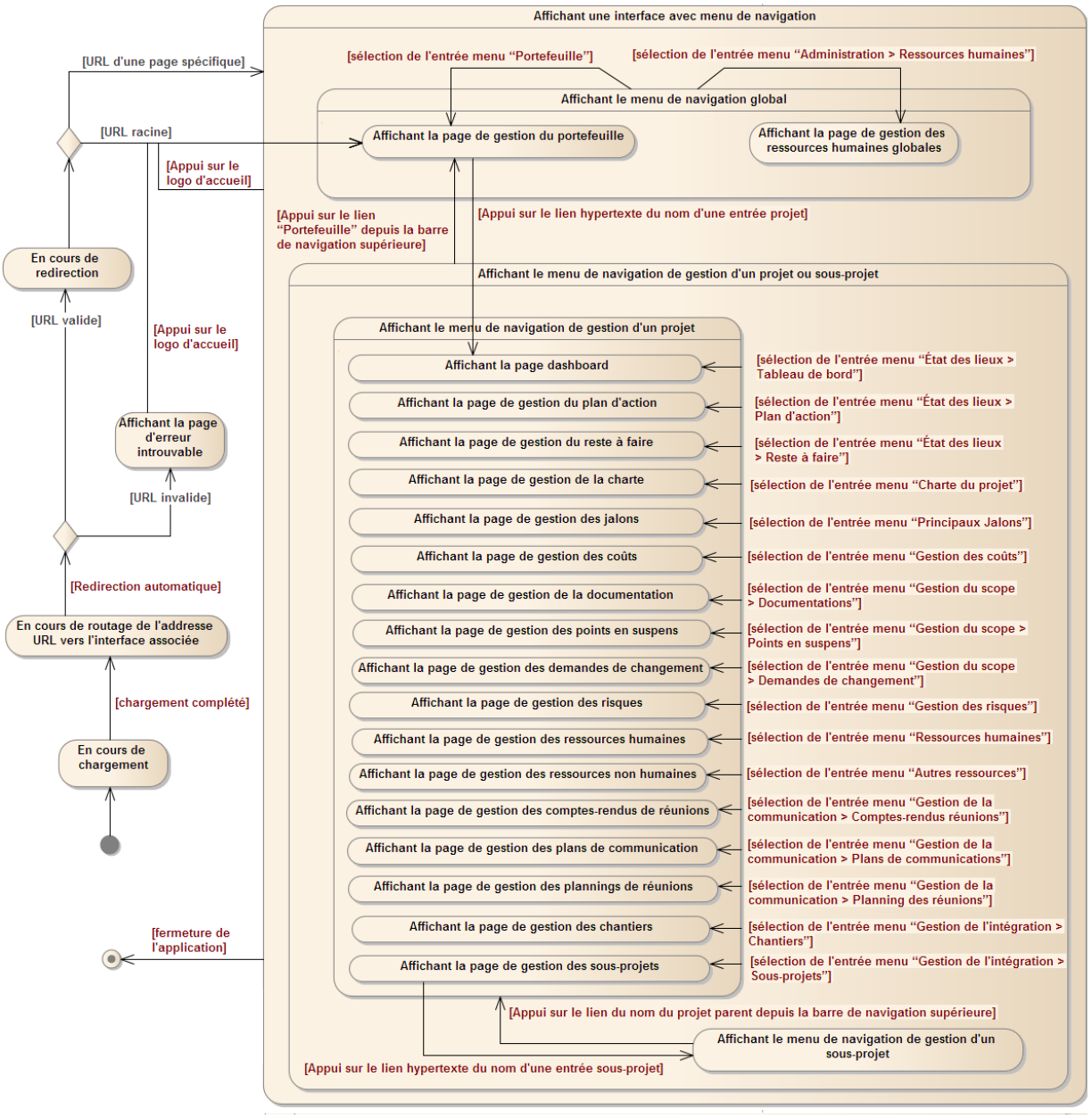

Figure V.1 – Diagramme d'états-transitions de l'application Front-end de gestion de projets

L'interface générale est caractérisée par un menu latéral offrant la navigation vers d'autres pages, directement reliées à la page en cours de visualisation. Le contenu du menu peut prendre trois formes :

— un menu spécifique à une gestion générale, indépendante de tout projet
— un menu spécifique à la gestion d'un projet
— un menu spécifique à la gestion d'un sous-projet

Les menu de gestion d'un projet se distingue par la disponibilité de gestion de l'intégration de sous-projets (un sous-projet ne peut en contenir un autre) et de gestion de la communication (pages planning de réunions, plans de communication et comptes rendus de réunions).

C'est par le biais d'une page spécifique à chaque aspect de gestion d'un projet que ce dernier peut être efficacement piloté et supervisé. En particulier, la page dashboard offre un tableau de bord rassemblant un ensemble d'indicateurs pertinents, destinés à assister les chefs de projet ou les parties prenantes dans leur prise de décisions. Pour le reste des pages, l'aspect de gestion de projet couvert peut être directement relié à un type de ressource spécifique.
Le comportement de la page de tableau de bord se résume dans le chargement des données provenant de l'API (une seule requête paramétrée sur la ressource projet ou sous-projet) ainsi que leur affichage sous forme d'indicateurs graphiques. Pour le reste des pages, nous spécifions leur comportement grâce à un diagramme complémentaire détaillant l'état associé à chaque page dans le diagramme précédent, illustré par la figure V.2.

Il est à noter que certaines de ces pages, en plus de charger les entrées du type de ressource associé, chargent des données statistiques dans une vue supplémentaire à celle de la table, sous forme assimilable à un mini tableau de bord. Il s'agit en l'occurrence des pages du plan d'action et de la gestion des risques. Celles-ci seront présentées plus tard dans la partie de validation des interfaces (section 4 p.73).

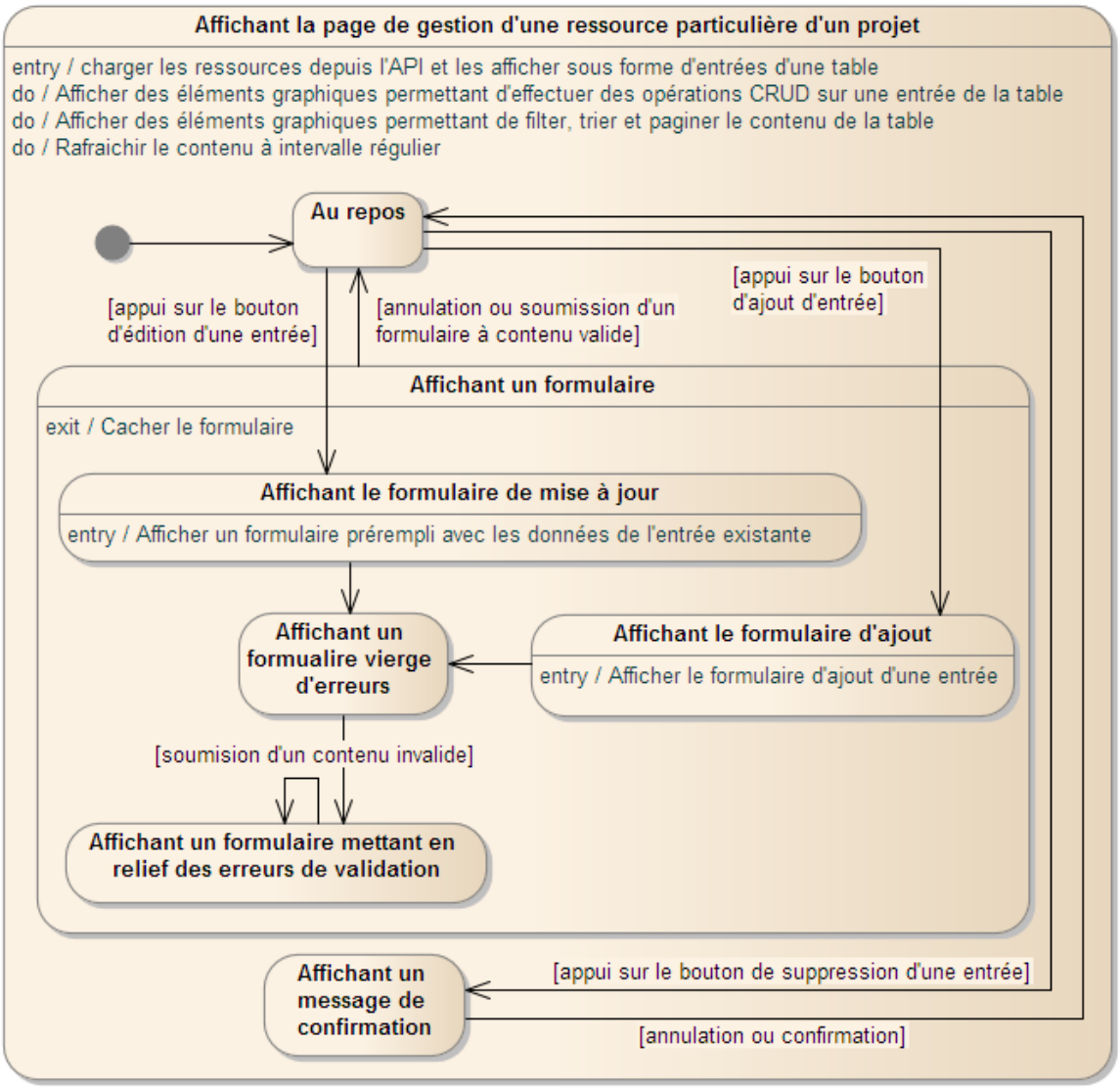

Figure V.2 – Diagramme d'états-transitions des pages caractérisées par une ressource

2 Technologies

Le produit à délivrer consiste avant tout en une application web. Dans ce cadre, le développement de la couche de présentation est désormais plus communément connu en tant que développement Front-end. En effet, avec la vulgarisation de la navigation web via dispositifs mobiles et l'émergence du langage JavaScript en tant que langage de programmation côté serveur, un nombre exorbitant de nouveaux d'outils et de concepts a vu le jour, dont le seul but est de perfectionner l'art du développement Front-end, d'accroître la qualité des interfaces web

modernes et d'instaurer la discipline nécessaire pour s'adapter au niveau élevé de complexité introduit par cet écosystème, toujours en pleine évolution.
C'est dans cet esprit que nous avons procédé à la séparation du Front-end et du Back-end de notre application en deux projets séparés. Le développement du Front-end peut ainsi se libérer des chaînes imposées par le Back-end et évoluer aisément de manière isolée et logique.

2.1 Environnement de développement

Un environnement de développement à part entière a été mis en place pour assister au développement de l'application Front-end. Parmi les nombreuses alternatives disponibles, nous nous sommes reposées sur les technologies suivantes :

— **Npm** [50] : Node.js [51] est un projet open-source visant à fournir un environnement d'exécution JavaScript côté serveur. Npm en soi en est le gestionnaire de paquets officiel. Entièrement écrit en JavaScript, il ne peut donc pas s'en dépasser pour son exécution. Npm permet, de manière assimilable à Gradle pour notre développement en Back-end, de gérer les dépendances de notre application Front-end et de télécharger automatiquement les ressources spécifiques dont nous avons besoin. Il intègre également des fonctions de build par l'exécution de commandes en terminal. Cette fonctionnalité est principalement utilisée conjointement à Node.js pour exécuter des programmes JavaScript souvent offerts par les librairies téléchargées.

— **GULP** [52] : Gulp est un outil d'automatisation écrit en JavaScript tournant sous Node.js. Il adopte un système à base de traitement de flux pour la mise en place facile de workflow complexes. C'est cet esprit de simplicité qui nous a convaincu à l'adopter en tant qu'outil de build pour notre application Front-end.

— **SASS** [53] : SASS est un langage de script dédié à la génération de feuilles de style CSS. Il permet d'étendre le langage CSS pour en simplifier le développement. À cet effet, il y intègre d'une part de nouvelles fonctionnalités utiles telles que l'imbrication de styles, l'héritage de styles, etc. Il y introduit d'autres part des concepts de programmation tels que les variables, les boucles, les mixins (assimilables aux fonctions) et bien d'autres.

Ceci permet de gérer efficacement les feuilles de style complexes pour les applications de grande envergure.

— **Autoprefixer** [54] : C'est avant tout un plugin PostCSS [55] qui répond au problème de préfixage de styles en CSS. En effet, les navigateurs se concurrencent pour l'implémentation des dernières mises à jour au langage CSS, et décident souvent d'implémenter de nouvelles fonctionnalités avant leur standardisation officielle. Chaque moteur web sur lesquels ces navigateurs se reposent adopte un préfixage particulier aux styles non standardisés. Autoprefixer permet d'analyser les feuilles de style CSS et de les traiter afin d'en mettre à jour le contenu et de couvrir correctement l'ensemble des styles spécifiques des navigateurs web à supporter, ce qui résout le problème susmentionné.

— **UglifyJS** [56] : C'est un analyseur de code JavaScript. Il permet de minifier les scripts ou en revanche de les embellir pour une compréhension facilitée à l'œuil nue. Nous nous intéressons ici à la fonctionnalité de minification de code, qui nous permettra d'alléger la taille des scripts engendrés pour une transmission web optimisée.

— **Live-server** [57] : C'est un serveur HTTP minimal. Il est impossible de développer en Front-end en local sans un serveur HTTP dédié au développement si l'on utilise les requêtes AJAX [58] dans nos pages web (dû à des mesures restrictives imposées par les navigateurs). Son attrait majeur réside dans sa capacité de mise à jour automatique de pages web en cours de développement. Se voulant minimal, il offre néanmoins de puissantes fonctionnalités de routage et de mapping.

— **Apache Server** [59] : Apache est un serveur web HTTP open-source, multiplateforme, développé sous les auspices de la fondation Apache. C'est le serveur web le plus populaire du marché (en 2016 [60]). Il offre une panoplie de fonctionnalités avancées comme le support des protocoles IPv6 et HTTP/2, la résistance et le recouvrement automatique aux pannes, la répartition de charges, etc. Ce sera notre serveur de base pour le déploiement de l'application en milieu de production.

2.2 Frameworks et Librairies côté client

Pour choisir adéquatement les composants Front-end à utiliser pour l'interface graphique, nous avons procédé à la définition de certaines exigences :

- Le Front-end devra consister en une SPA (Single Page Application) afin de rivaliser avec l'ergonomie offerte par les interfaces d'applications de bureau.
- Les interfaces devront être en mesure de s'adapter à différentes tailles d'écran pour offrir une navigation optimisée pour les dispositifs mobiles.
- les interfaces dédiées au reporting et à la visualisation de statistiques devront utiliser des graphiques familiers (cercles en camemberts, diagrammes à barres, ...) pour la consultation de données, afin de répondre convenablement aux besoins d'aide à la décision escomptées de l'application.
- les composants visuels doivent répondre aux attentes des utilisateurs en termes de modernité (animations fluides, calendrier de sélection pour les dates, ...).

Au sein de la multitude de choix à notre disposition, nous avons finalement décidés d'employer les libraries suivantes pour parvenir à réaliser chacune de nos exigences :

- **AngularJS** [24] : présélectionné dans l'étude technique effectuée préalablement à la phase de construction, le framework a été développé essentiellement dans le but d'étendre le langage HTML afin d'en dynamiser le contenu. Ceci est réalisé intuitivement via les éléments de base du langage (attributs et balises) et l'usage d'expressions simples (évaluées via JavaScript) pour la définition de leur contenu. Il se base sur le pattern de conception MVVM (Model-View-ViewModel) ou MVB (Model-View-Binder) qui dérive directement du paradigme MVC. Il aspire à garantir un certain niveau de qualité dans la conception d'applications web complexes, en séparant d'emblée la présentation (View) du modèle (Model) et assurant leur mise en relation au travers du concept de data-binding. Ce concept est implémenté via l'utilisation d'objets JavaScript pour la communication entre les deux couches. AngularJS est aussi populaire pour son modèle de développement à base d'injection de dépendances. En somme, il répond parfaitement aux besoins de

conception de SPA.

— **Bootstrap** [61] : c'est un framework pour le design de sites et d'applications web. Il fournit des composants de base HTML et CSS visant à s'aligner avec les principes d'ergonomie généraux d'interfaces comme l'adoption de modèles typographiques modernes, de styles par défaut populaires pour les boutons, la navigation, les éléments de formulaires, etc. En particulier, le framework offre sa propre solution pour la réalisation d'interfaces web adaptives aux différentes dimensions d'écran, ce qui se révèlera très utile pour la conception de l'interface utilisateur mobile.

— **NVD3** [62] : c'est une librairie offrant un ensemble de composants graphiques réutilisables pour la création de graphiques statistiques. Il s'agit d'une abstraction au-dessus de la librairie d3.js [63]. Là où cette dernière offre une liberté complète de manipulation des plus fins détails des graphiques réalisés, NVD3 se veut plus simple d'utilisation en mettant à disposition un nombre limité de graphiques familiers légèrement configurables. Ceci ne nous empêche cependant en rien d'utiliser d3.js pour configurer plus en détail ces composants si le besoin se présente.

— **AngularUI** [64] : cette librairie se repose sur AngularJS et son modèle de fonctionnement pour intégrer des composants avancés, facilement configurables depuis ce dernier. En particulier, nous avons utilisé les modules AngularUI Bootstrap pour l'intégration la configuration simple des éléments de Bootstrap depuis notre application, ainsi qu'AngularUI Router pour la gestion de la navigation entre les vues de notre application.

— **NgTable** [65] : cette librairie offre des fonctionnalités puissantes de dynamisation de tables HTML : tri, pagination, filtrage, etc. Elle se repose sur l'intégration avec AngularJS.

3 Réalisation

Le modèle de développement en AngularJS se repose sur le pattern d'injection de dépendance. Le framework distingue entre deux types de composants injectables :

— **Les services** : un service est une implémentation du pattern Facade, qui préconise de cacher la complexité inhérente au fonctionnement d'un composant, du reste des composants avec lesquels il interagit. Un service fournit simplement un ensemble de fonctionnalités sous forme d'interface, exploitable par le reste des composants. Les services sont mis à disposition des autres services et des objets spécialisés au travers de l'injection de dépendances.

— **Les objets spécialisés** : ce sont des objets définis de manière conforme à une spécification d'AngularJS. Il s'agit soit de contrôleurs, de directives, de filtres ou d'animations. Très brièvement :

  — Le *contrôleur* est le composant responsable de la communication avec l'interface utilisateur.

  — Une *directive* consiste en une extension d'un composant HTML de base en un composant à comportement dynamique.

  — Un *filtre* applique une logique prédéfinie pour la mise en forme d'une donnée en entrée.

  — Une *animation* permet de lier dynamiquement un style spécifique à un composant HTML.

  Nous ne procéderons qu'à la conception de contrôleurs au cours de notre développement. Pour le reste, nous nous reposerons sur les objets spécialisés mis à disposition par défaut par le framework.

Les composants non injectables quant à eux consistent principalement en des constantes, qui représentent des objets JavaScript supposés immuables, ainsi qu'en des blocks de configuration, exécutés au démarrage de l'application.

La figure V.3 présente le diagramme de packages de l'application Front-end réalisée. Celui-ci nous permet d'identifier précisément les différents composants mis en place. Ceux-ci seront détaillés dans la partie qui suit.

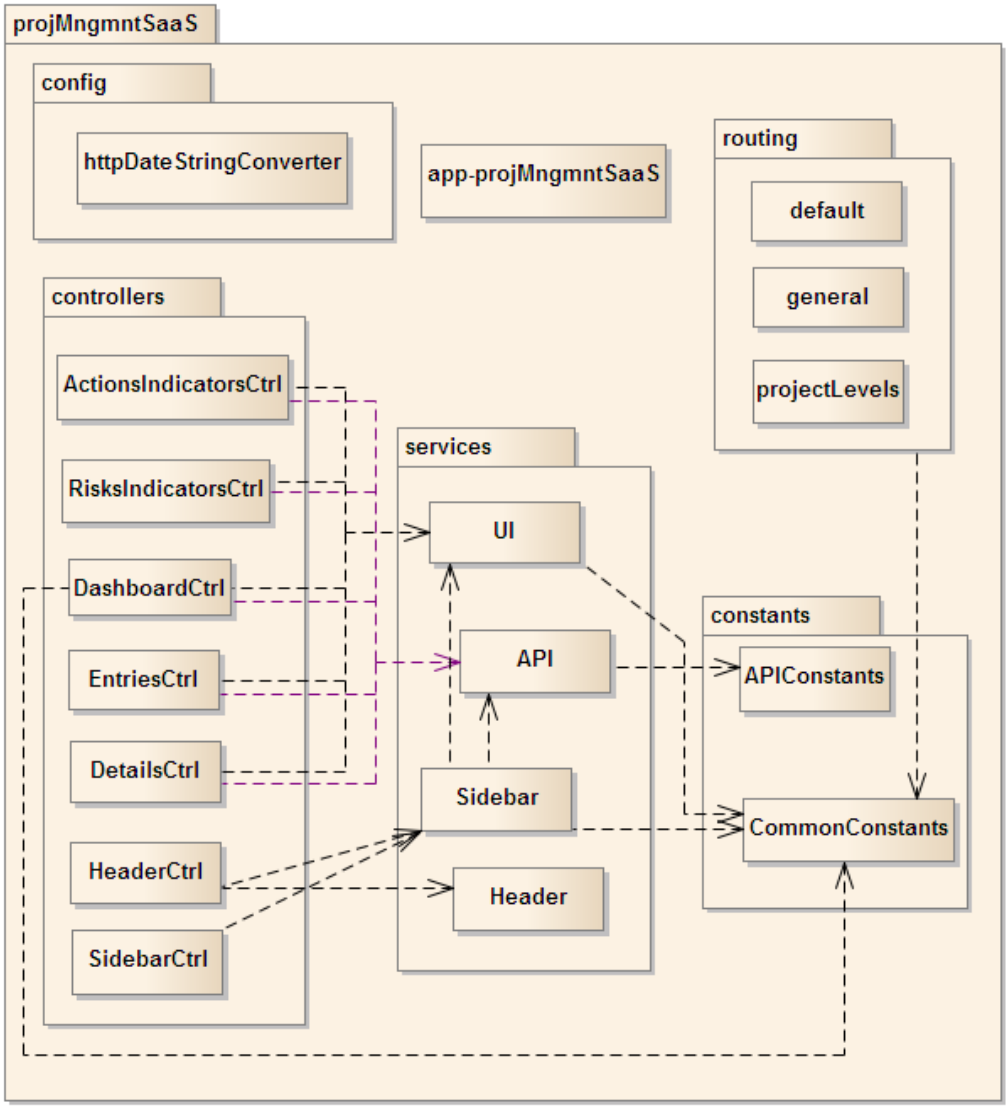

Figure V.3 – Diagramme de packages de l'application Front-end

Le package config inclut des blocks de code à caractère de configuration, exécutés au démarrage de l'application. *httpDateStringConverter* en l'occurrence inscrit un intercepteur pour messages HTTP entrants, afin de convertir les champs date textuels (format JSON) en des objets JavaScript de type Date. Le package routing inclut de même des blocks de code exécutés au démarrage de notre application, spécifiquement responsables de la gestion du routage et de la navigation en bas niveau (grâce à AngularUI Router). Les fichiers default, general et projectLevel définissent respectivement le routage par défault, celui associé aux pages générales (telles

celle de gestion du portefeuille de projets) ainsi qu'au routage associé aux pages de gestion d'un projet ou sous-projet.

Le package services inclut les services déclarés, à savoir :

- *UI* : Le service UI permet de centraliser la récupération des données des interfaces utilisateurs (champs formulaires, menus, ...).
- *API* : Ce service fournit un repository générique pour la récupération des données statistiques et des ressources requises par les différentes pages de l'application. C'est le seul composant qui communique directement avec l'API en Back-end.
- *Sidebar* : Ce service permet de consulter et de modifier les données du menu latéral de notre interface.
- *Header* : Ce service permet de consulter et de modifier les données de la barre de navigation supérieure de notre interface.

Nos contrôleurs sont définis au sein du package de même nom. Ils incluent :

- *EntriesCtrl* : C'est le responsable des vues associées à nos collections de ressources pour la gestion de projets. Ces vues présentent ces dernières sous forme d'entrées dans une table, acompagnée d'un formulaire déroulant pour l'ajout et la mise à jour de ces ressources.
- *DetailCtrl* : Il est responsable des vues associées à une ressource unique (en contraste avec les collections de ressources). Il s'agit en l'occurrence des pages dédiées à la gestion de la charte d'un projet ou d'un sous-projet, et à la gestion des coûts.
- *ActionsIndicatorsCtrl* et *RisksIndicatorsCtrl* : Ce sont les contrôleurs associés aux vues des minis tableaux de bord des pages du plan d'action et de la gestion des risques.
- *HeaderCtrl* et *SidebarCtrl* : Ces contrôleurs permettent d'abstraire la logique complexe associée à la gestion des vues du menu latéral et de la barre de navigation supérieure en centralisant le code en question sous forme d'une interface simple.

Les applications AngularJS consistent en concrètement d'un ensemble de modules, chacun déclaré en association avec une page web. Notre application est définie grâce au module déclaré au sein du fichier app-projMngmntSaaS. Elle est associée essentiellement à la page d'index de notre application. Elle représente le seul point d'entrée de notre application. Il a ainsi été nécessaire de configurer notre serveur statique afin de fournir notre page d'entrée, indépendam-

ment de l'URL demandée. La redirection vers l'URL requêtée est intrinsèquement traitée par l'application Front-end grâce aux éléments de routage définis.

4 Tests & Validation

L'application Front-end a été déployée au sein des serveurs de l'entreprise d'accueil afin de valider le fonctionnement correct de l'application en milieu de production. En particulier, le déploiement a représenté l'occasion d'effectuer des tests fonctionnels en milieu réel, par ma personne et par l'ensemble du personnel de l'entreprise impliqué dans le projet. Ceci a permis l'ajout ou la redéfinition de spécifications tout au long du déploiement continu de l'application, de même que l'identification de bugs ou d'anomalies, techniques ou conceptuelles, par rapport à la release précise les ayant introduits.

Cette section vise d'une part à présenter l'interface graphique réalisée et l'ergonomie générale de l'application, et d'autre part, à valider les spécifications fonctionnelles relatives à la gestion de projets. C'est dans cette optique que nous procéderons à la présentation et au test du rendu de l'interface sous différents contextes.

Barre supérieure

L'interface se caractérise par une barre supérieure offrant une navigation simplifiée pour l'aspect hiérarchique de gestion de projets.

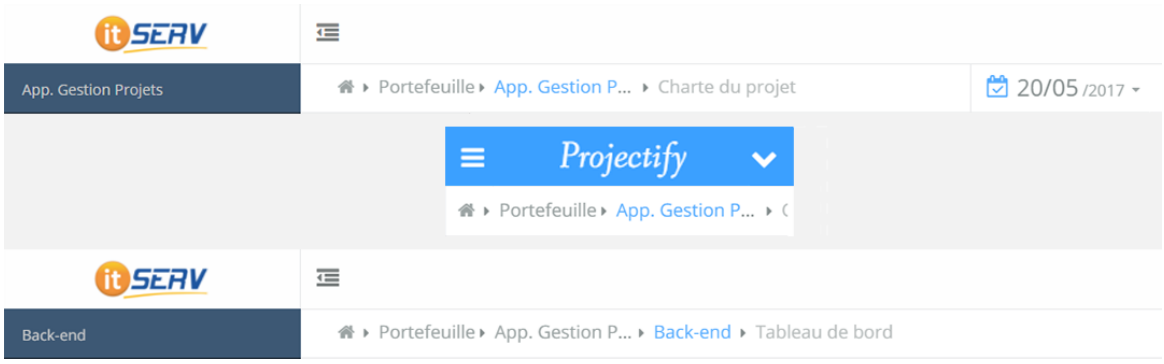

Figure V.4 – Rendu de la barre supérieure

La figure V.4 présente, de haut en bas : le rendu dans la page de gestion d'un projet, la version réduite, et le rendu dans la page de gestion d'un sous-projet. Notons la présence d'un bouton dans la version élargie, permettant de transitionner entre la version réduite et complète du menu latéral. Un bouton est aussi présent dans les pages de contenu archivable d'un projet, présentant tout d'abord la date de la mise à jour du contenu en cours de visualisation, et offrant de changer celle-ci via un sélecteur de date (son rendu peut être visualisé plus tard dans les formulaires d'autres pages).

Menu latéral

Le menu général de navigation consiste en un menu latéral de contenu logiquement relatif à la nature de la page en cours de visualisation : Général (global) ou Projet (projet ou sous-projet). La figure V.5 présente de gauche à droite dans l'ordre : le menu général en vue mobile, le menu de gestion de projet, le menu de gestion de sous-projet ainsi qu'un exemple du menu en format réduit (niveau sous-projet).

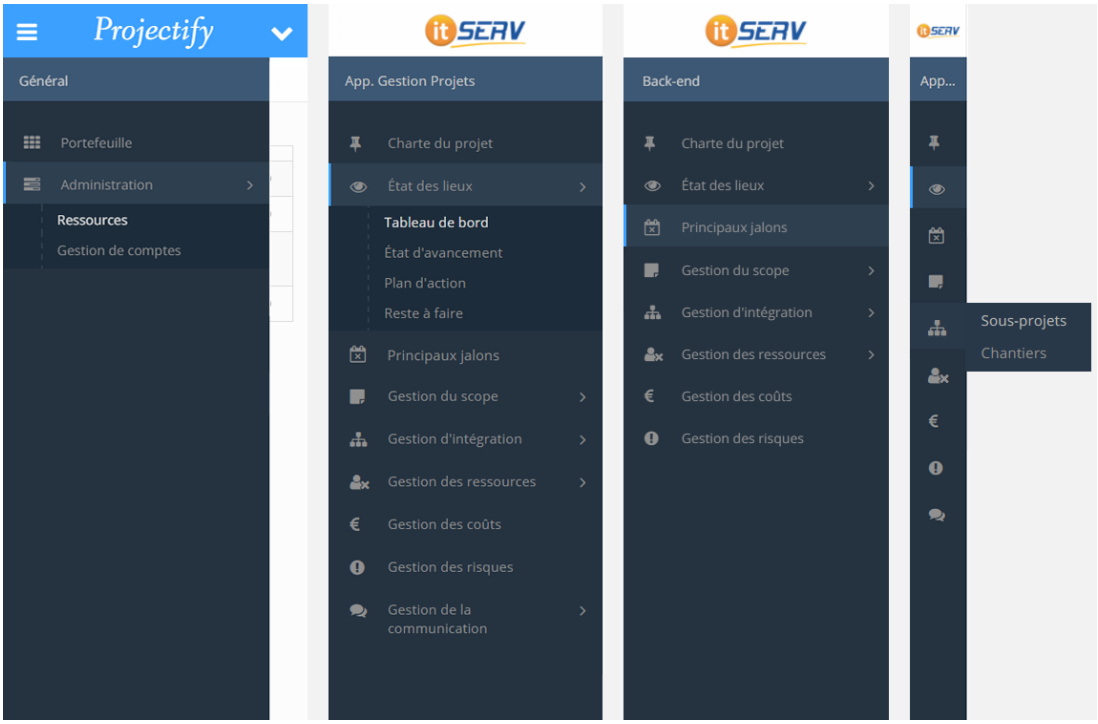

Figure V.5 – Rendu du menu latéral

Tableau de bord

La conception du tableau de bord a fait le sujet d'une étude préliminaire sur les indicateurs de performance généraux pour le pilotage d'un projet. Son interface assure une adaptabilité élevée aux différentes tailles d'écrans.

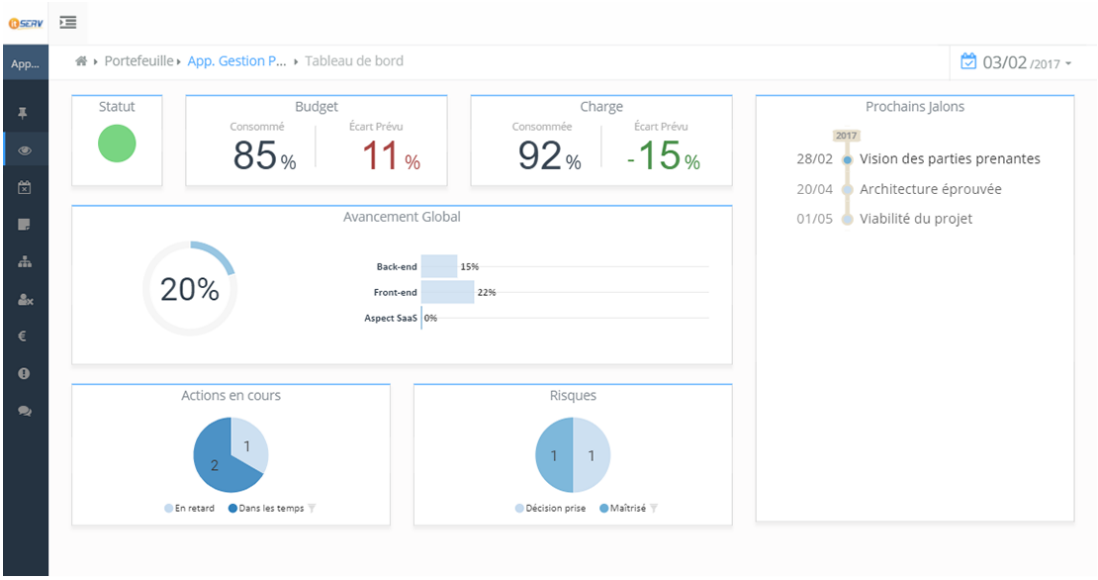

Figure V.6 – Rendu du tableau de bord sous écran large - menu réduit

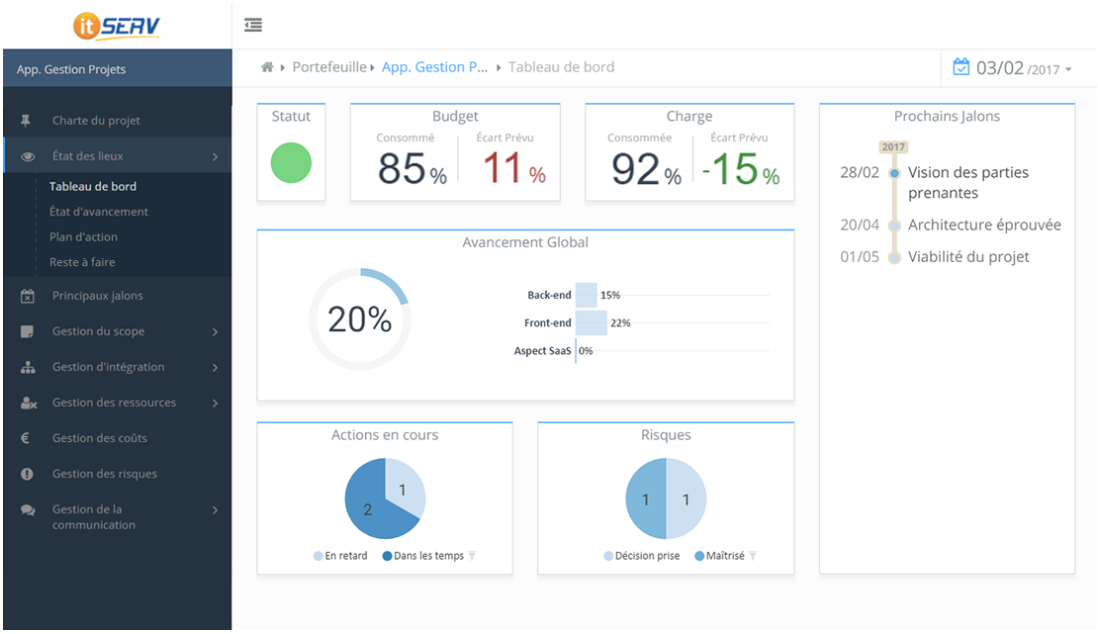

Figure V.7 – Rendu du tableau de bord sous écran large - menu ouvert

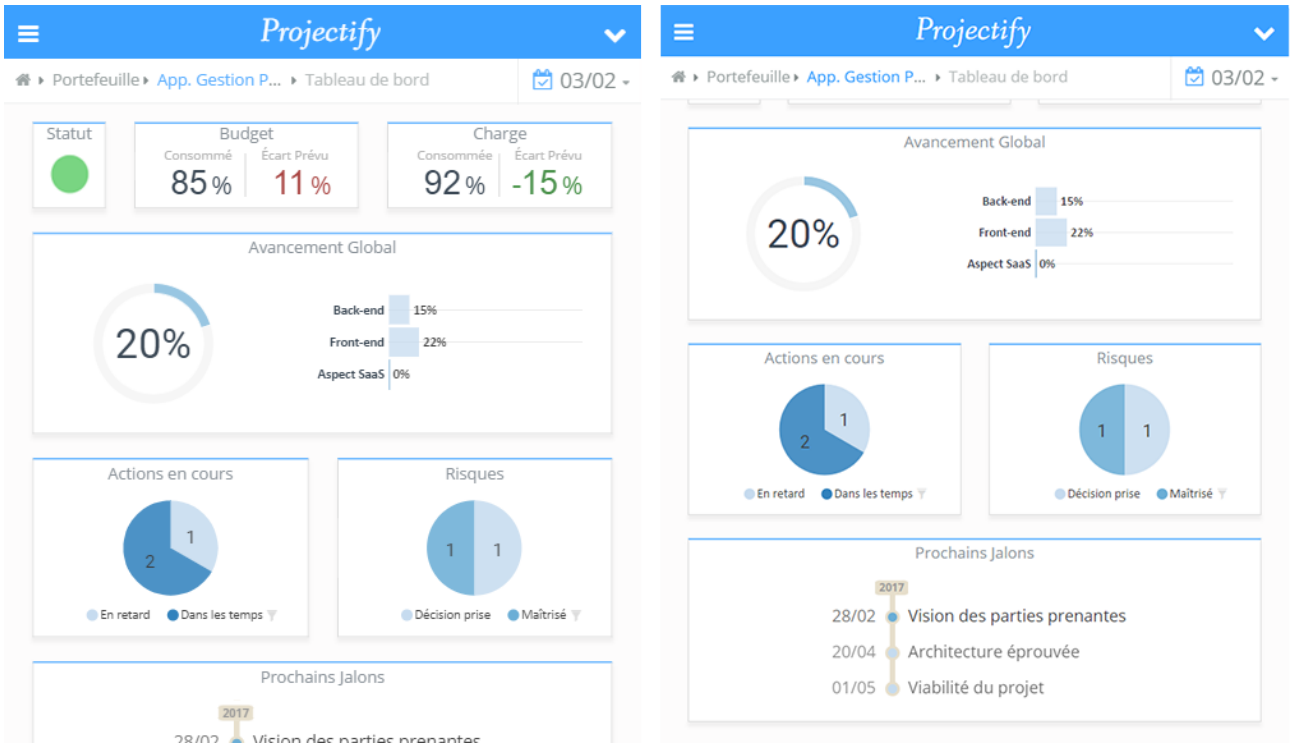

Figure V.8 – Rendu du tableau de bord sous écran réduit (tablette)

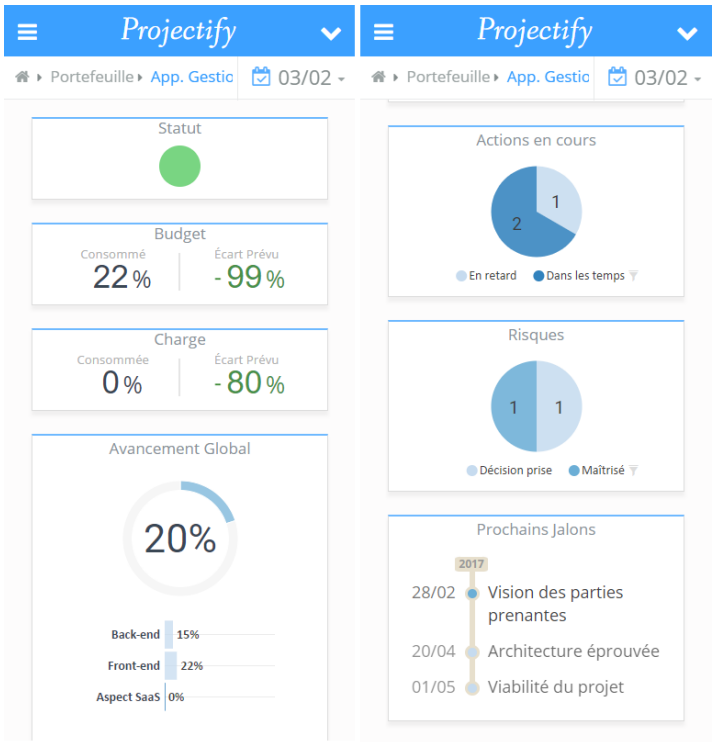

Figure V.9 – Rendu du tableau de bord sous écran réduit (smartphone)

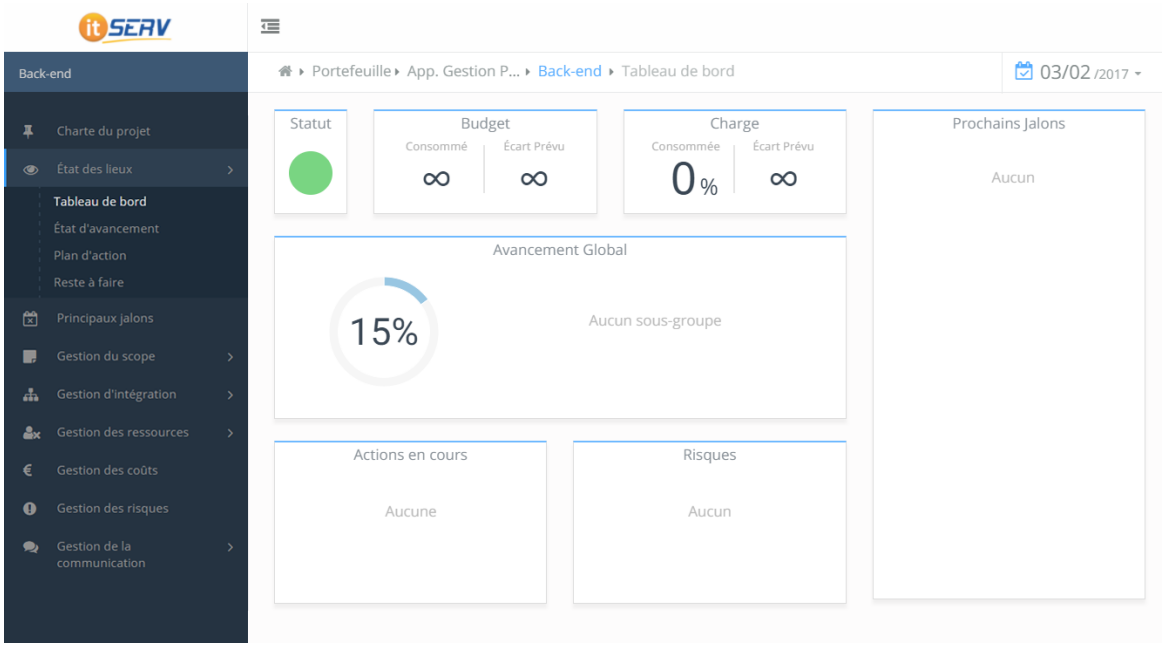

Figure V.10 – Visualisation du tableau de bord d'un sous-projet (similaire)

Pages de gestion de collection de ressources

Un projet est principalement gérable sous forme de ressources (actions, risques, etc), se distinguant uniquement par l'objet de leur usage et les propriétés propres à chacune. Ceci nous a poussé à adopter une interface générique pour la gestion de chacune de ces collections de ressources. Précisons aussi que la page de gestion du portefeuille de projets global ne se différencie aucunement de ces dernières. Nous présenterons ici l'exemple de la page de gestion des principaux jalons.

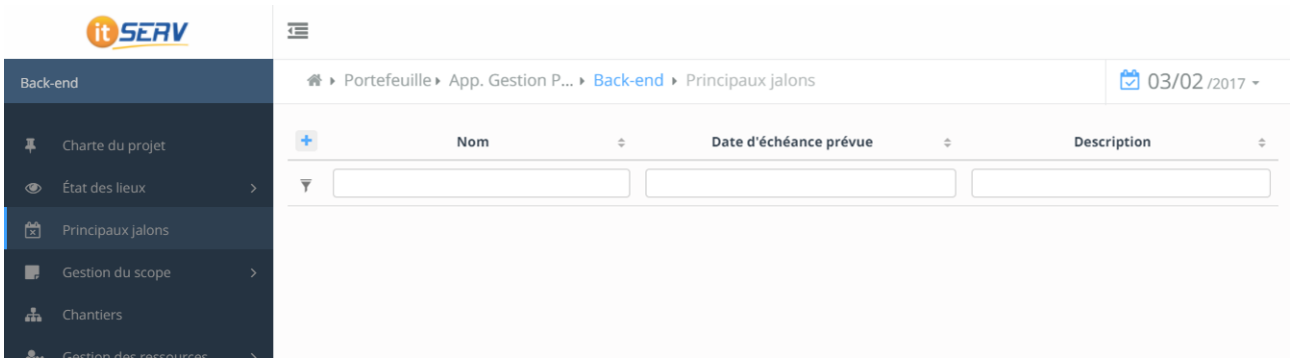

Figure V.11 – Page de gestion des jalons - état initial

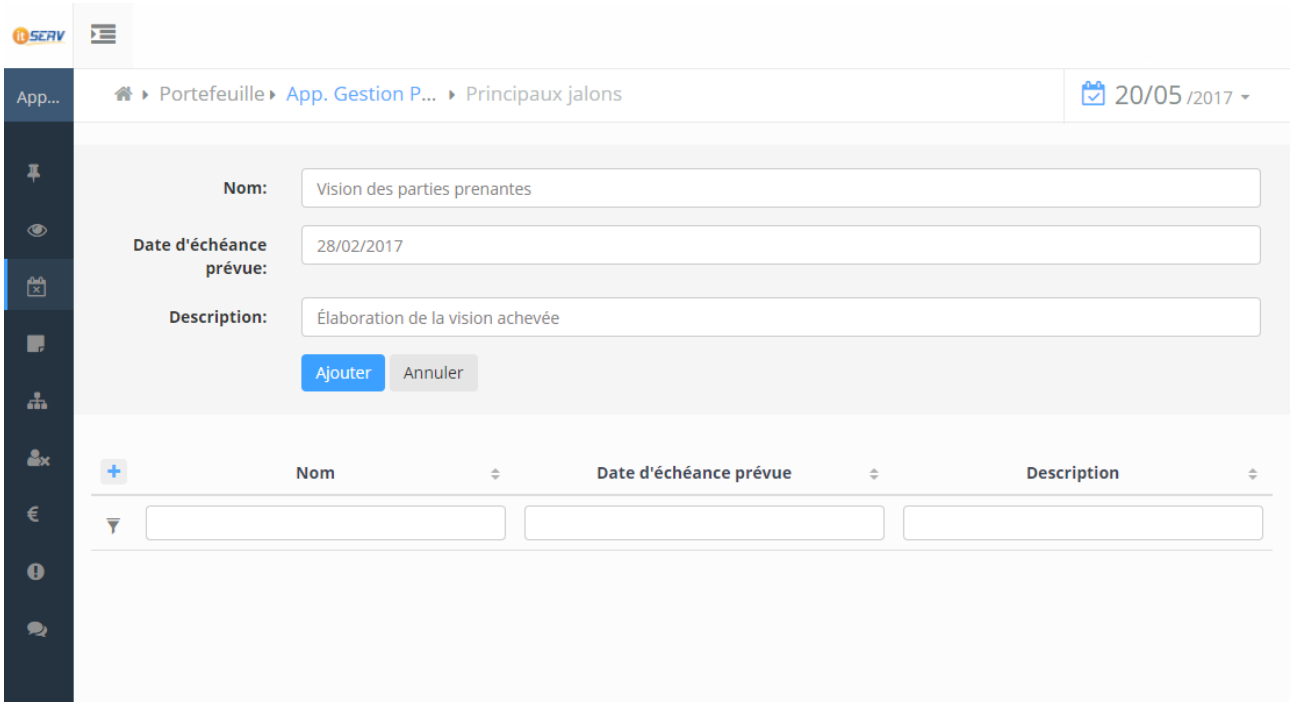

Figure V.12 – Page de gestion des jalons - formulaire valide

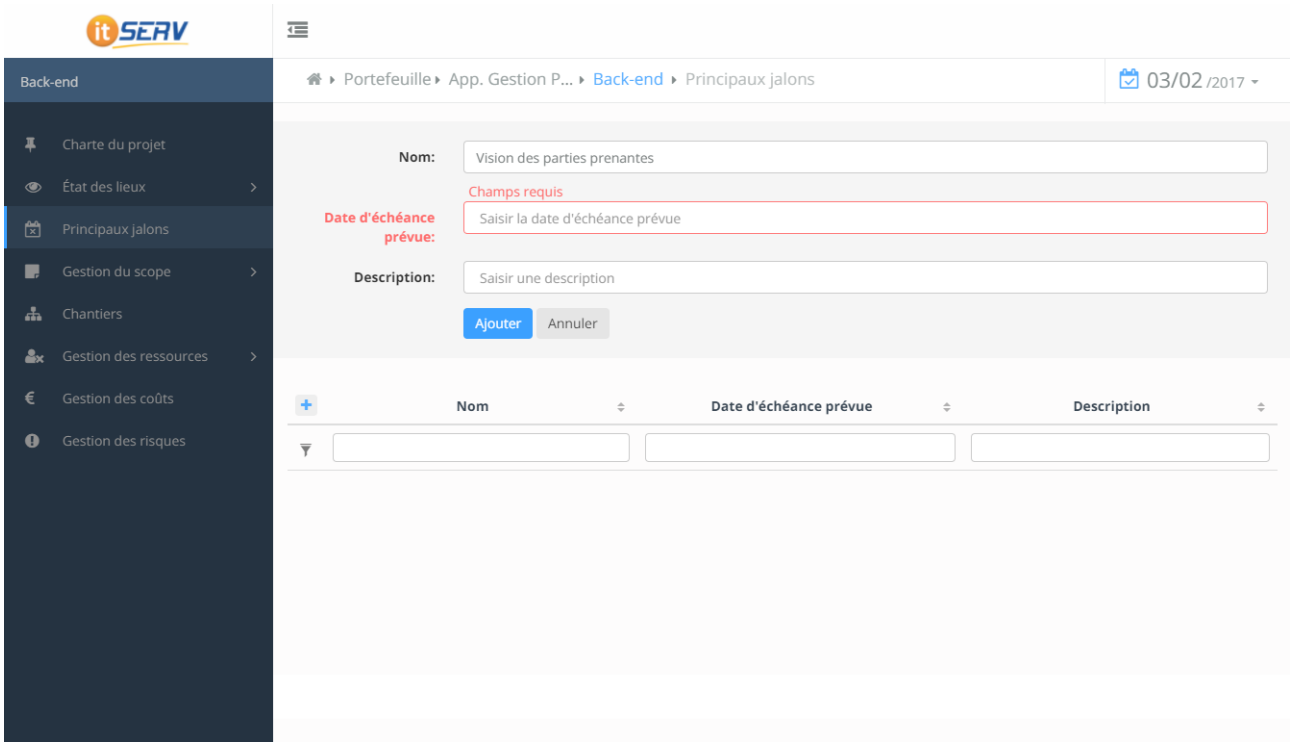

Figure V.13 – Page de gestion des jalons - soumission invalide d'un formulaire

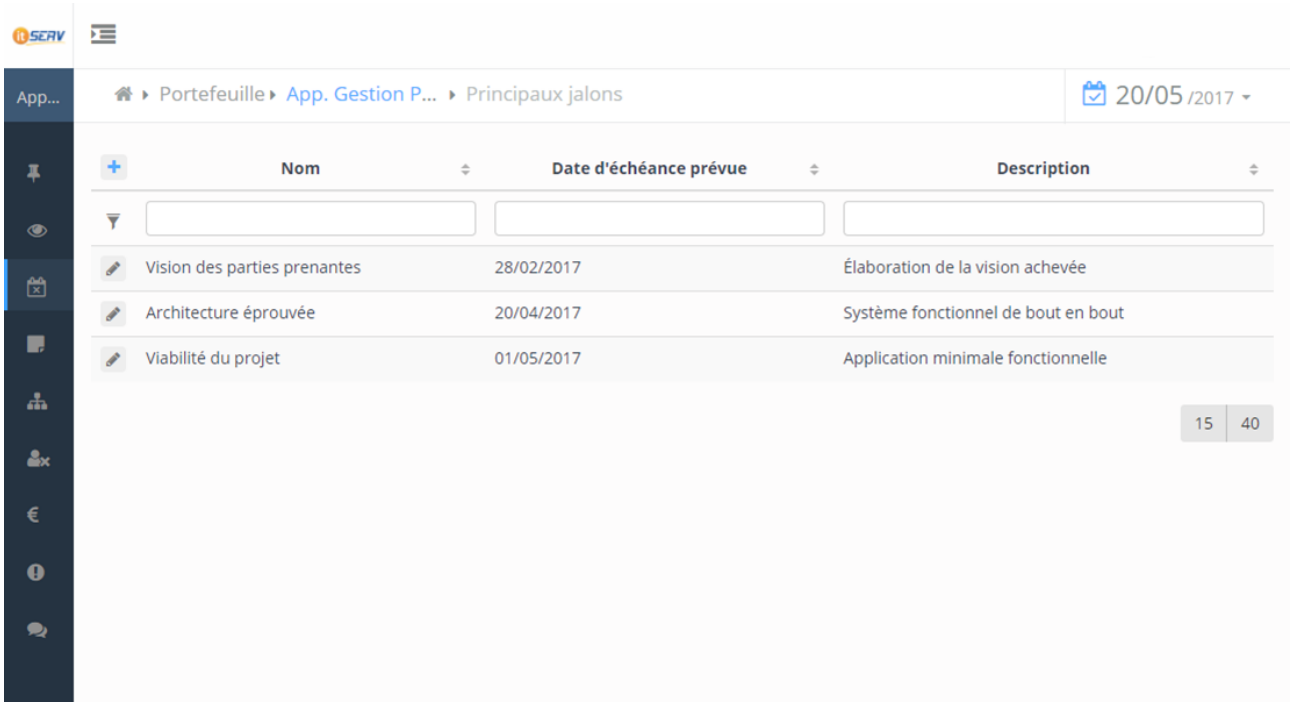

Figure V.14 – Page de gestion des jalons - table d'entrées remplie

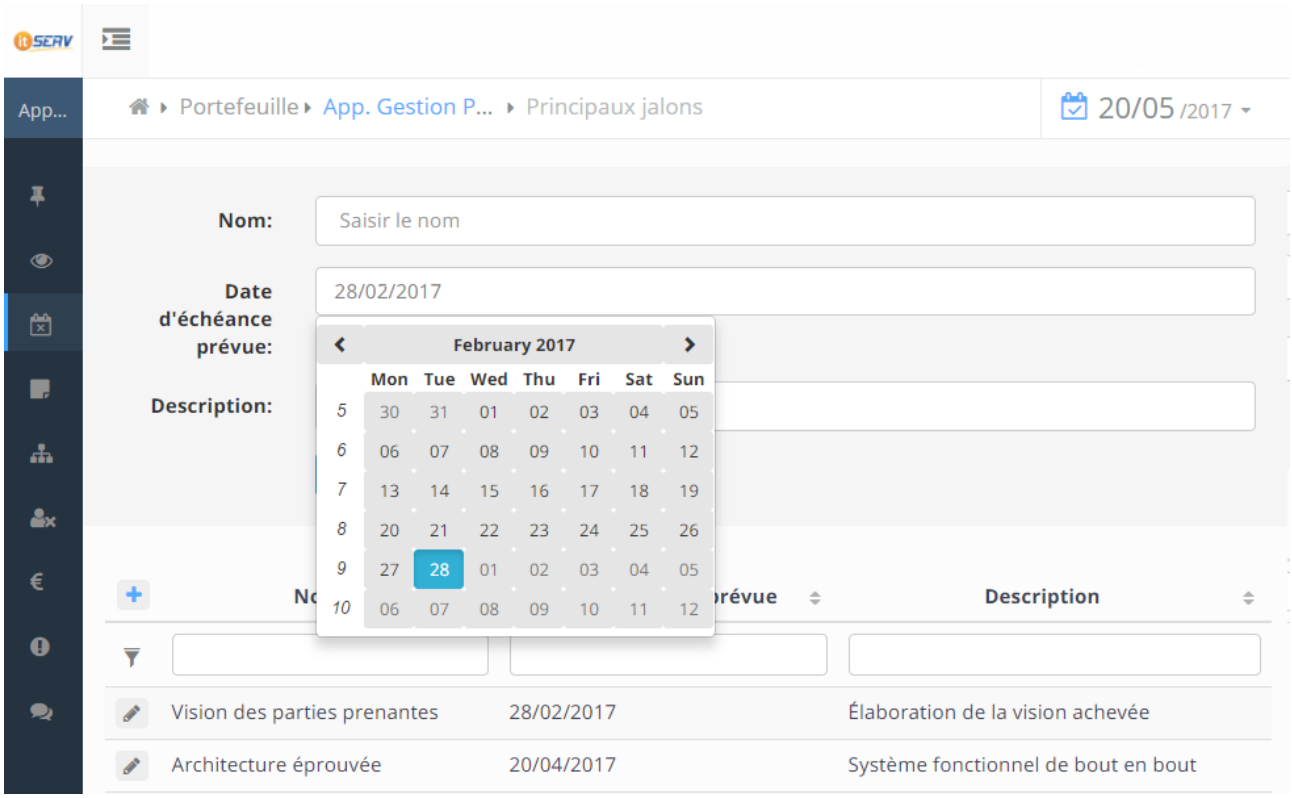

Figure V.15 – Page de gestion des jalons - composant d'entrée de date

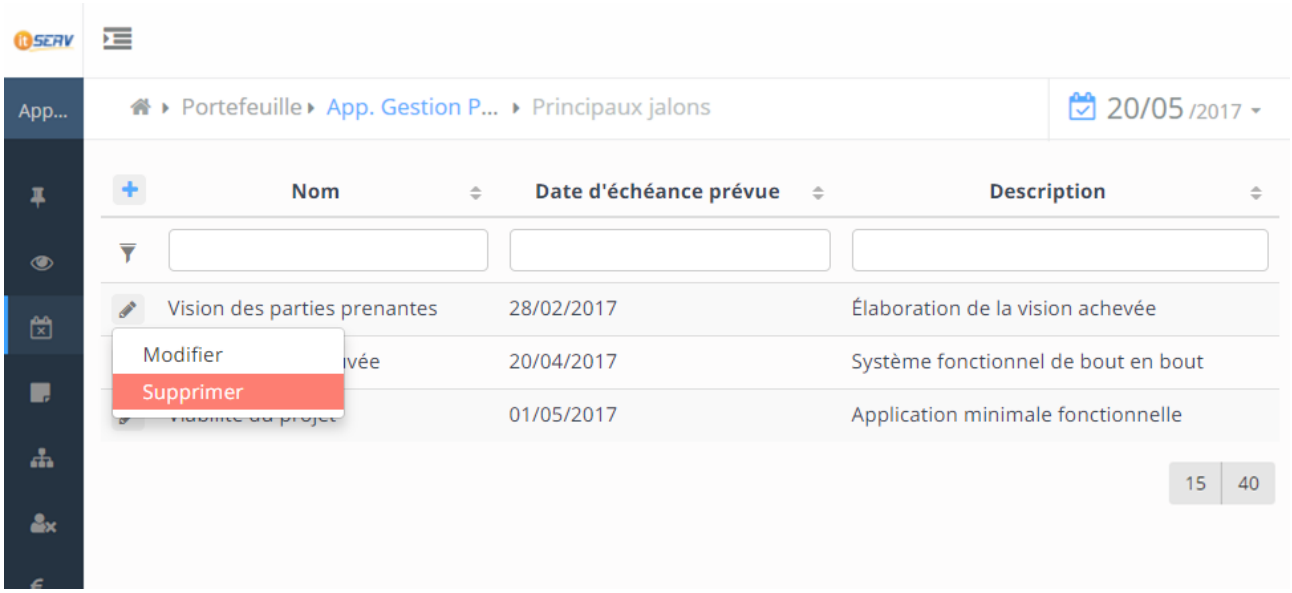

Figure V.16 – Page de gestion des jalons - options d'édition d'entrées

Les pages de gestion de l'intégration consistent en la gestion des chantiers et des sous-projets.

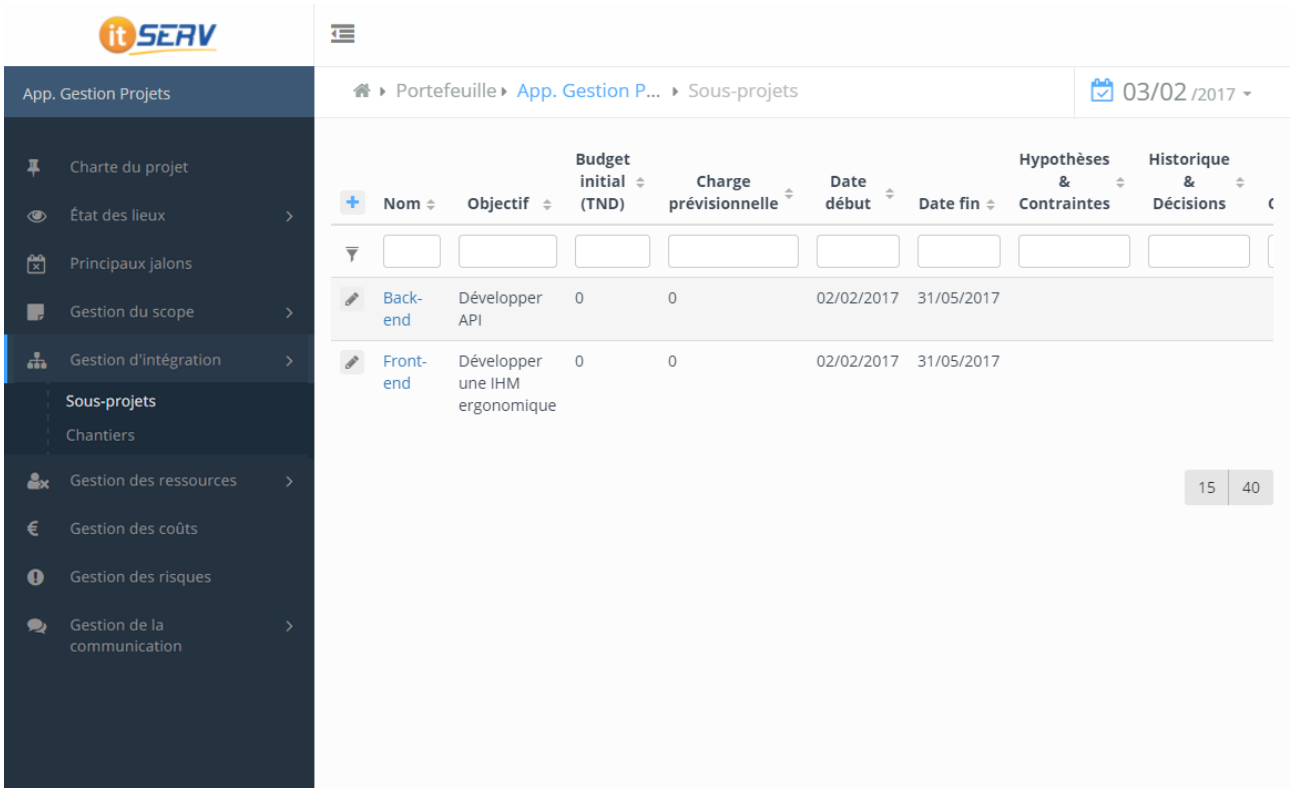

Figure V.17 – Rendu de la page de gestion des sous projets sous écran large

Ces pages sont identiques aux précédentes pages de gestion de ressources, cependant, un sous-projet se voulant très similaire dans sa gestion à un projet, son envergure nous a convaincu à lui dédier un menu complet pour sa gestion, se distinguant très peu de celui d'un projet (voir figure V.5). Celui-ci est accessible au travers d'un lien hypertexte, représenté par le nom du sous-projet, qui plonge l'utilisateur dans l'interface consacrée à sa gestion (visualisable à la figure V.10). Nous saisissons l'occasion pour présenter la visualisation sous écrans à format réduit des pages des pages génériques de gestion de collection de ressources.

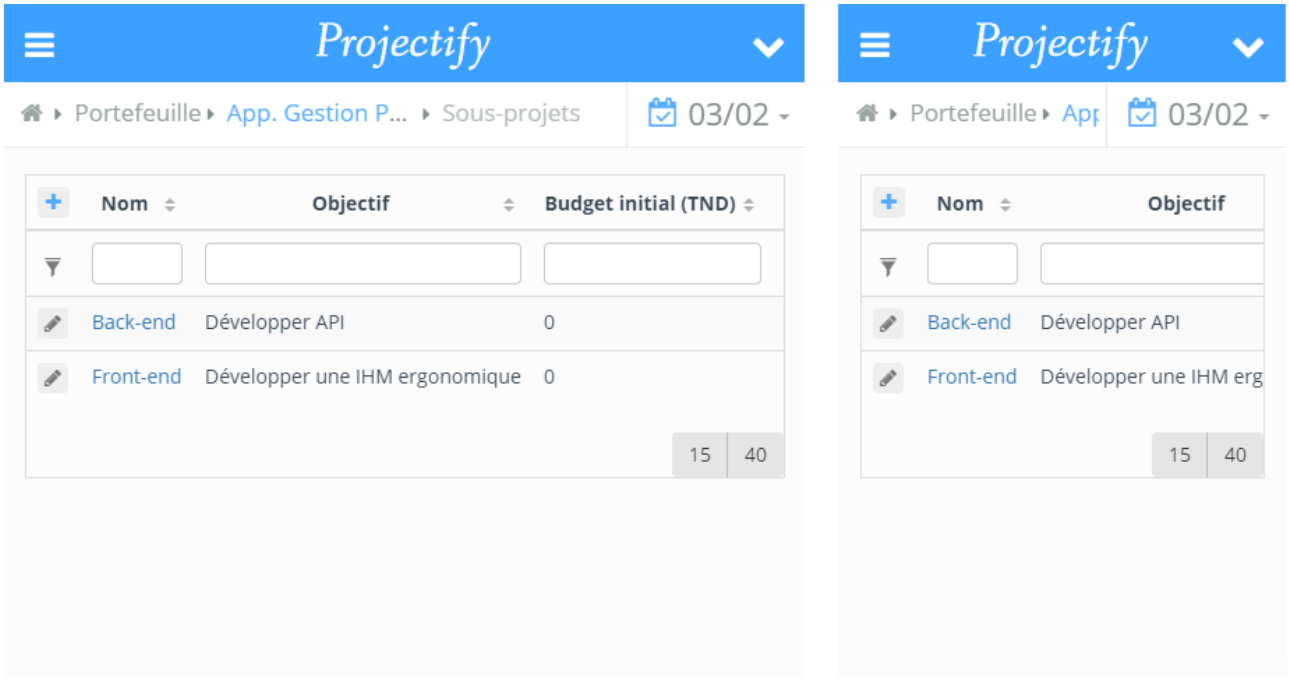

Figure V.18 – Rendu de la page de gestion des sous projets sous écrans réduits

Page de gestion des actions

Pour un suivi efficace des actions, la page dédiée à leur gestion a été enrichie d'un mini tableau de bord résumant en quelques indicateurs pertinents le statut de l'avancement global de celles-ci.

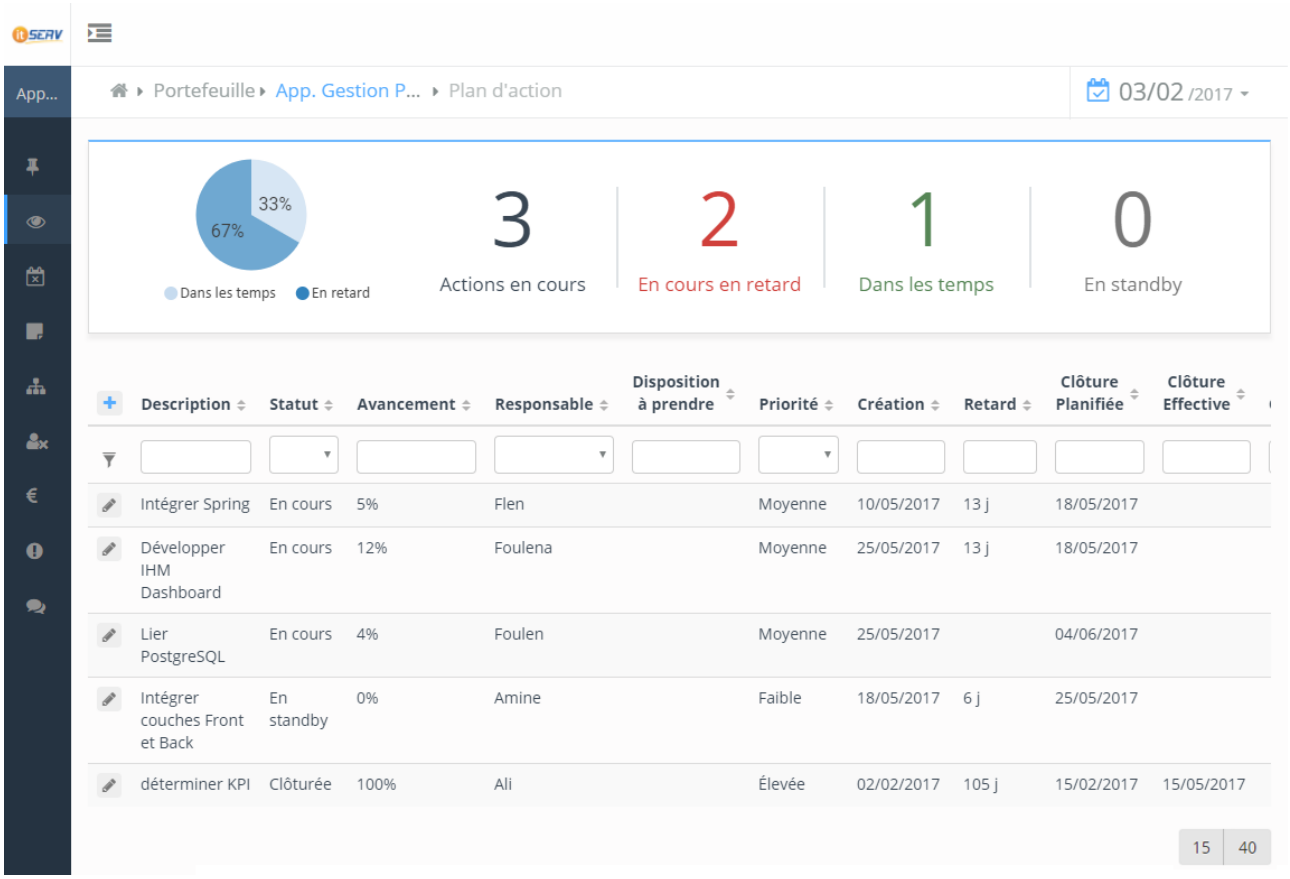

Figure V.19 – Rendu de la page de gestion des actions

Pages de ressource individuelle

La page de gestion de la charte d'un projet ainsi que la page de gestion des coûts représentent chacune un ensemble de données associés à une ressource (le projet lui-même). Nous présentons ici le rendu de ces pages sous les modes de consultation et de mise à jour.

Figure V.20 – Rendu de la page de gestion des coûts en mode de consultation sur écran large

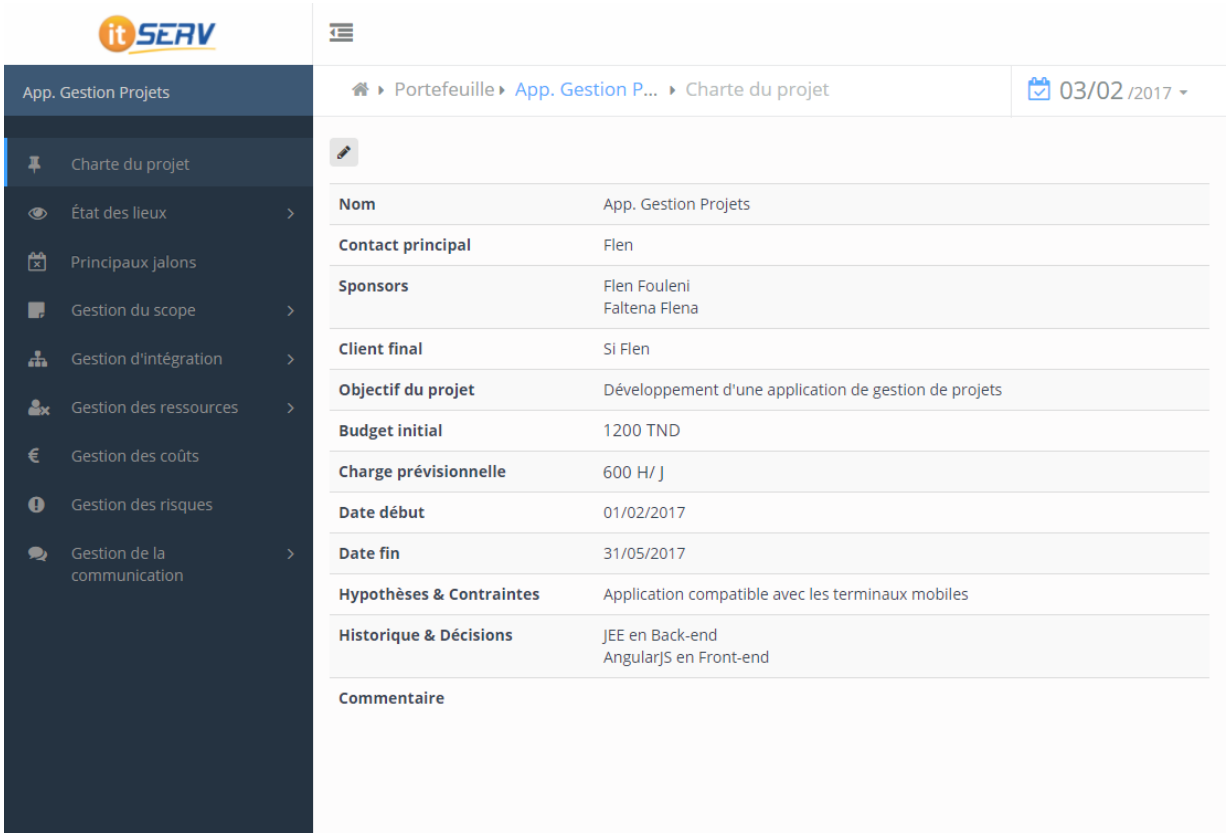

Figure V.21 – Rendu de la page de gestion de la charte projet en mode de consultation sur écran large

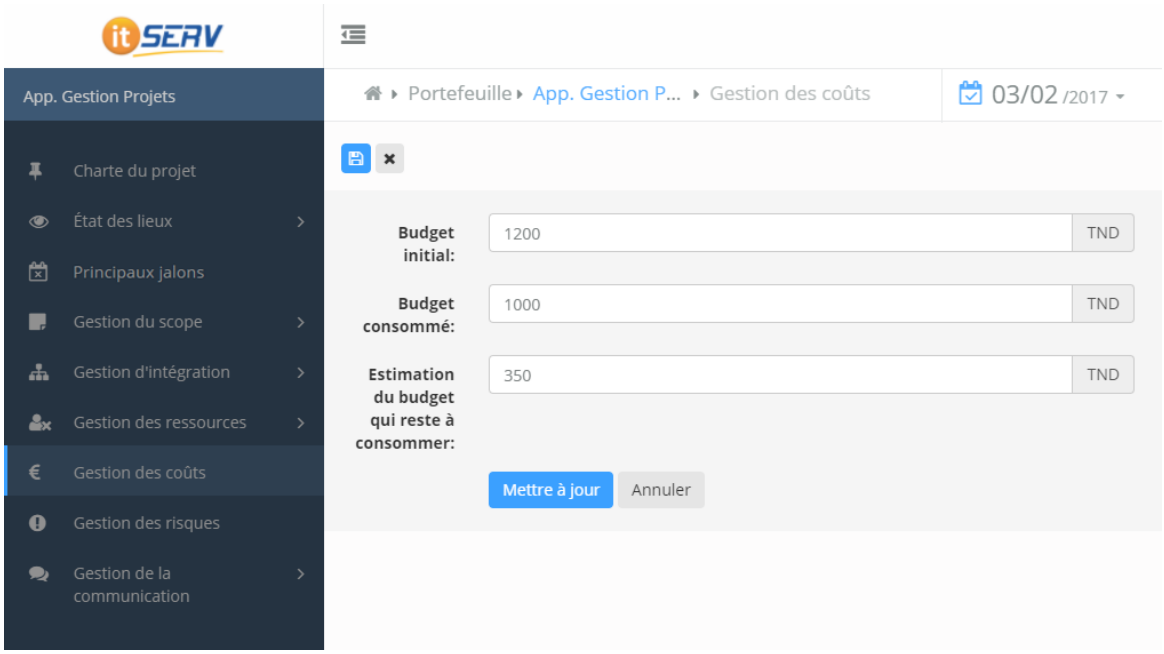

Figure V.22 – Rendu de la page de gestion des coûts en mode d'édition sur écran large

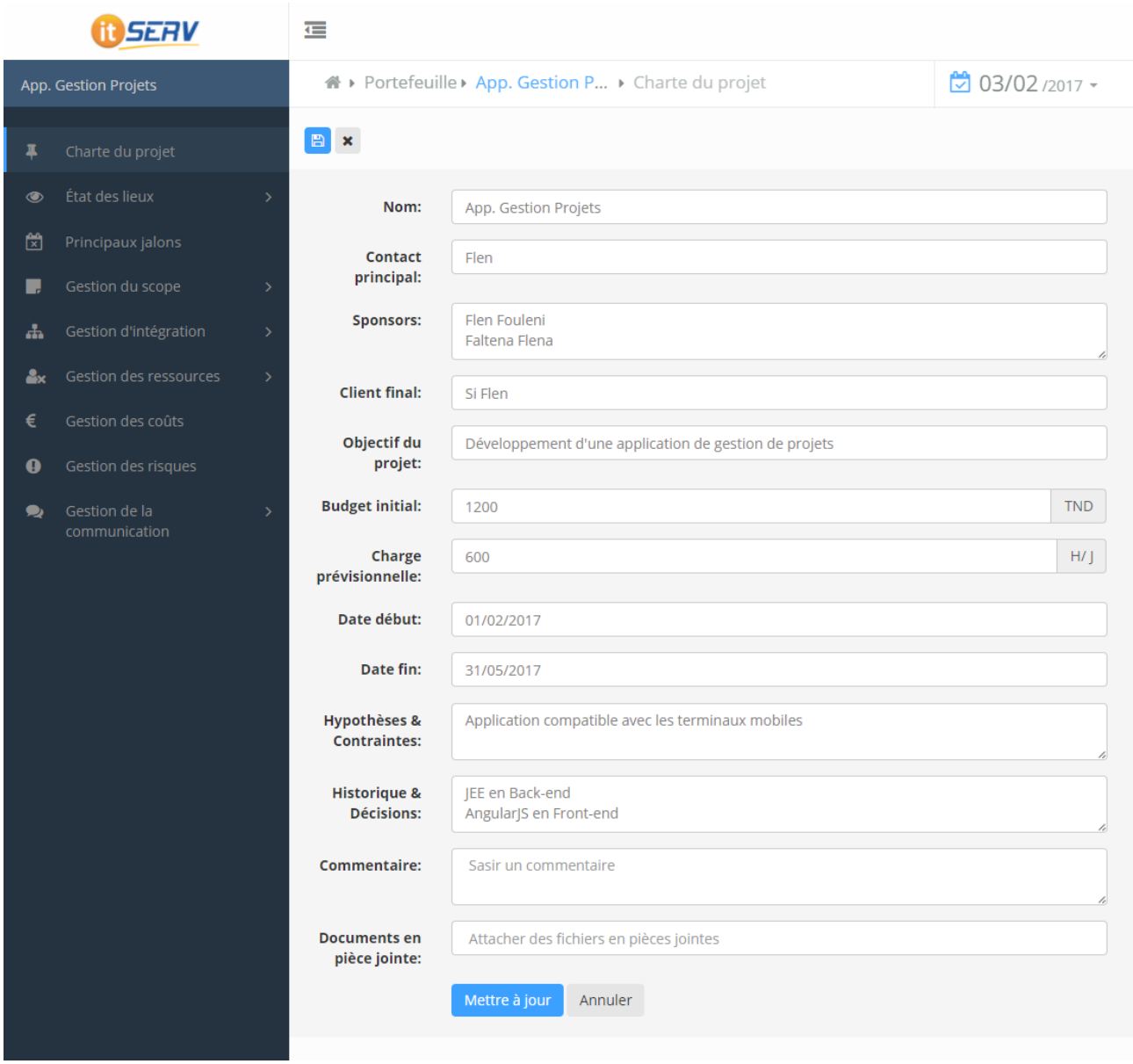

Figure V.23 – Rendu de la page de gestion de la charte projet en mode d'édition sur écran large

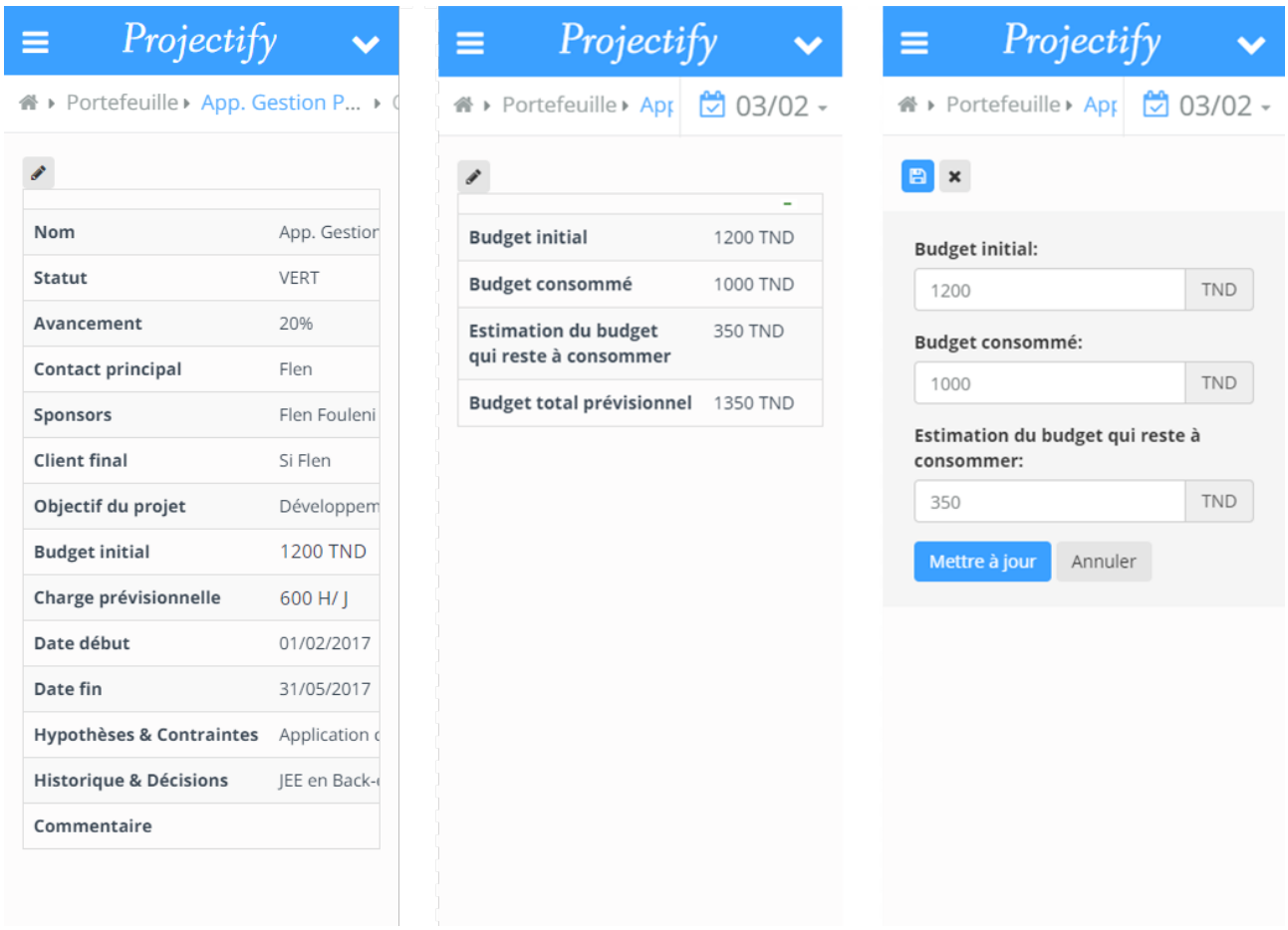

Figure V.24 – Rendu des pages de gestion des coûts et de la charte projet sur écran réduit

Bilan

Les interfaces précédentes valident la majorité des spécifications non fonctionnelles de notre application :

— **Ergonomie** : outre les choix typographiques et les composants graphiques de référence utilisés, c'est surtout grâce au menu latéral de navigation entre les pages logiquement reliées, à la navigation simplifiée entre les niveaux de gestion qu'assure la barre de navigation supérieure, et à l'intuitivité et la familiarité introduites par la généricité adoptée pour les pages de notre application que nous pouvons prétendre d'offrir une interface conviviale et facile d'utilisation.

— **Adaptabilité** : l'adaptabilité des interfaces développées aux écrans de format réduit a constitué un aspect primordial pour nous lors de la conception de l'interface utilisateurs. Celui-ci a été exhaustivement couvert par la visualisation du rendu sur différents format d'écrans réduits (smartphone et tablette) de l'ensemble des types de pages conçues.

L'aspect non fonctionnel de sécurité sera finalement concrétisé dans le chapitre restant.

Conclusion

À la complétion de ce chapitre, notre application de gestion de projets est déjà prête à l'emploi en mode mono-utilisateur. Avec le développement en Front-end achevé, et les fonctionnalités de gestion de projet couvertes, nous pouvons désormais nous focaliser sur l'aspect de gestion des utilisateurs. De même, les spécificités du processus de développement des caractéristiques de produit SaaS de notre applicatif seront présentées de manière exhaustive au cours du chapitre final.

Chapitre VI

GESTION DES UTILISATEURS ET DES CLIENTS

Plan

Introduction

Ce dernier chapitre sera consacré à la présentation des fonctionnalités de gestion d'utilisateurs des des clients de l'offre SaaS. Le développement de ces deux aspects sera étroitement lié dans le processus de développement suivi. Finalement, c'est la présentation de l'application complémentaire de gestion des clients de l'offre SaaS qui attestera de la validation de l'ensemble des spécifications fonctionnelles et non fonctionnelles de notre projet.

1 Gestion des utilisateurs

La gestion des utilisateurs de notre application en mode SaaS impose d'associer chaque compte utilisateur à l'un des clients de l'offre SaaS au travers de l'implémentation d'un mécanisme d'authentification élaboré. Pour un même client, la distinction entre les utilisateurs de l'application et leurs rôles sollicitera l'implémentation d'un mécanisme d'autorisation complémentaire.

1.1 Conception

Nous procéderons en premier lieu à l'expression des besoins en termes de fonctionnalités, relativement à la gestion de comptes utilisateurs, grâce à une analyse fonctionnelle basée sur les cas d'utilisation. Nous enchaînerons ensuite avec une modélisation dynamique mettant en relief les aspects d'autorisation et d'authentification mis en place.

1.1.1 Analyse fonctionnelle

La figure VI.1 présente le diagramme de cas d'utilisation associés à l'aspect de gestion d'utilisateurs au sein de notre application. Précisions notamment que chaque acteur illustré est assimilable à un utilisateur de type de compte de même nom. Quelques remarques complémenteront le diagramme.

À l'acquisition de l'application, il n'existe qu'un seul compte : celui de l'administrateur par défaut. C'est au moyen de ce compte que l'ensemble initial du reste des comptes utilisateurs est créé. Notons de plus qu'il peut exister plusieurs comptes administrateurs, principalement dans le but d'éviter une éventuelle rigidité lors de l'exploitation de l'application.
Il est utile de créer un compte pour une partie prenante d'un projet afin de lui fournir un suivi facilité de l'état de son projet. Un compte partie prenante est créé par un chef de projet et est affecté à un projet. Tout compte chef de projet supervisant le projet en question peut gérer les comptes de l'ensemble de ses parties prenantes.
Un chef de projet n'a accès qu'aux projets auxquels il a été affecté. Cette tâche d'affectation revient à un dirigeant, qui se réserve le droit de l'affecter à un ou plusieurs projets. Le dirigeant n'a pas accès aux comptes parties prenantes puisqu'il ne s'occupe pas de l'aspect de gestion du projet, mais se limite uniquement à une consultation en lecture seule pour un projet quelconque.

La gestion de chaque compte inclut sa création, la mise à jour du nom, du login ou du mot de passe de ce dernier, ou bien sa suppression. La gestion des comptes par un chef de projet ou un dirigeant, respectivement ceux de parties prenantes et de chefs de projets, incluent de plus l'affectation du compte à un projet. Notons par ailleurs que les administrateurs gardent la

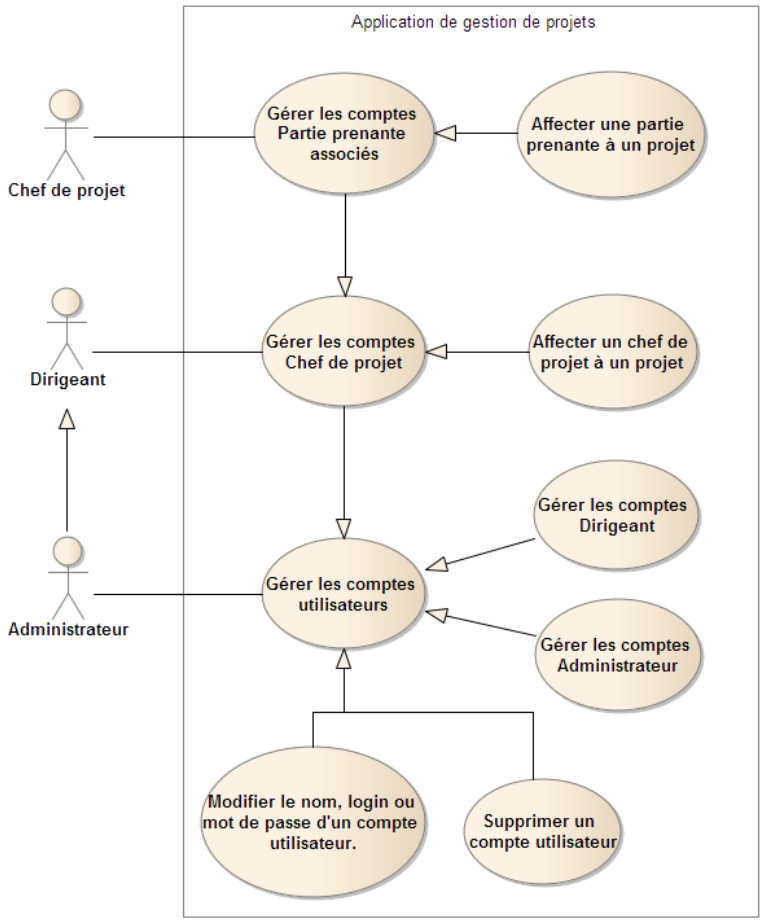

Figure VI.1 – Diagramme de cas d'utilisation de l'aspect gestion d'utilisateurs de l'application

possibilité de gérer tous les types de comptes, néanmoins sans possibilité de création de comptes parties prenantes ou d'affectation de projets.

1.1.2 Modélisation statique

Le modèle de classes relatif à l'aspect de gestion des utilisateurs, de l'authentification et de l'autorisation est présenté à la figure VI.2. Les accesseurs par défaut ont été omis du diagramme. Dans cette partie, nous utilisons des fonctionnalités spécifiques à Hibernate pour gérer l'isolation des données en schéma au niveau de la base de données, grâce aux classes du package config.hibernate. La classe TenantContext est utilisée pour chaque requête pour garder une trace sur le schéma en base de données à utiliser lors du requêtage de celle-ci (détermination

du client). Ce sont les intercepteurs (package config.interceptors) qui interceptent les requêtes avant leur arrivée aux contrôleurs pour configurer le schéma à utiliser au moyen d'analyse des entêtes HTTP dédiées à cet effet. Pour l'authentification, nous concevons que nos clients se connecteront à l'application via une URL contenant un sous domaine l'identifiant uniquement (champ pseudo de Tenant). Ici, on procède à l'introduction d'une classe associée aux comptes utilisateurs (User). Pour inscrire l'utilisateur couramment authentifié au contexte global, nous recourons à l'implémentation de UserDetails, interface fournie par SpringSecurity. Précisons que le comportement des classes préexistantes a lui aussi été sujet à certaines extensions. Les détails de réalisation sont abordés plus en profondeur dans la section 1.2.1.

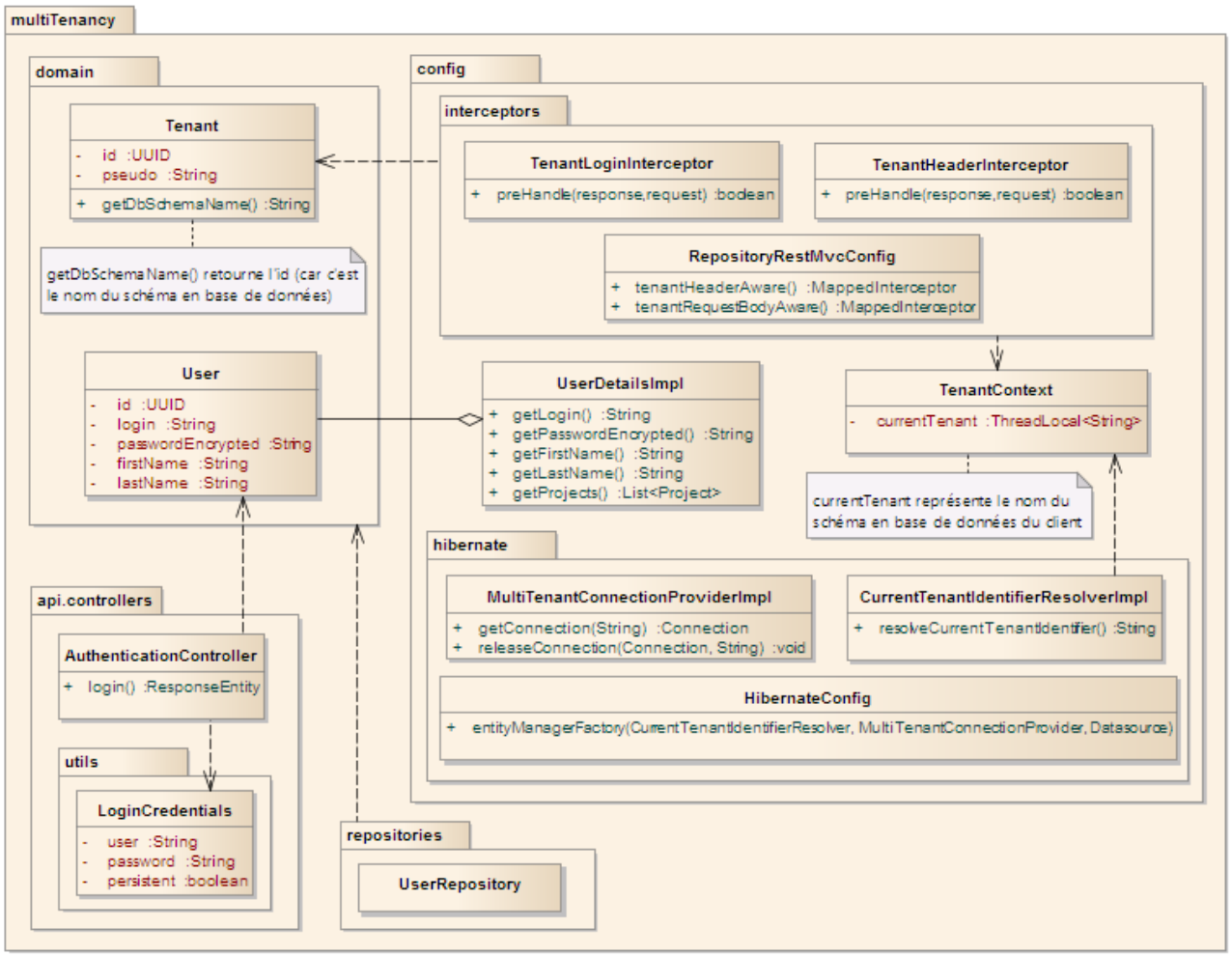

Figure VI.2 – Diagramme de classes des nouvelles entités relatives à la gestion des utilisateurs

1.1.3 Modélisation dynamique

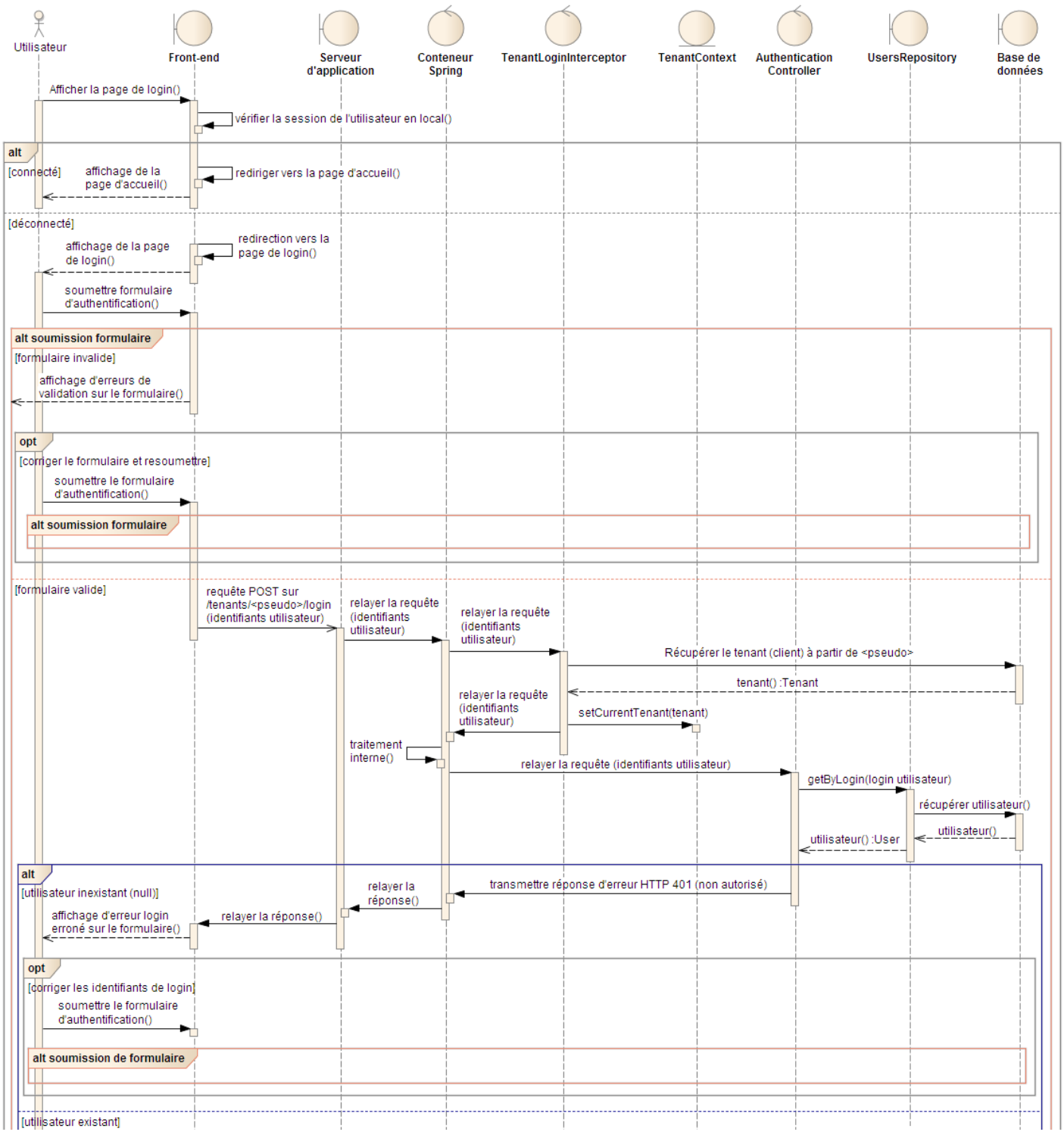

Figure VI.3 – Diagramme de séquence : S'authentifier (partie 1)

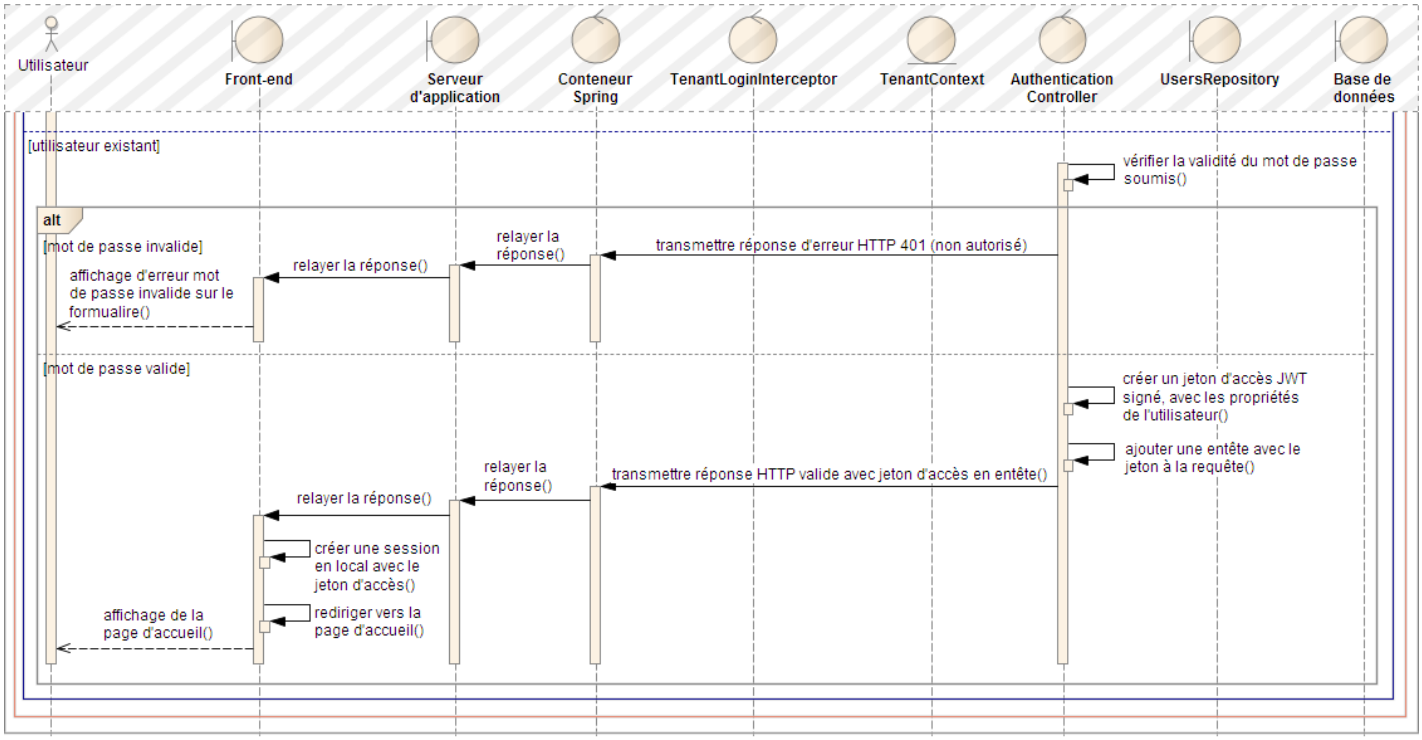

Figure VI.4 – Diagramme de séquence : S'authentifier (partie 2)

Le diagramme illustré par les figures VI.3 et VI.4 démontre le fonctionnement du mécanisme d'authentification mis en place. Il est basé sur l'utilisation d'un jeton d'accès, de contenu signé (de confiance), à persister en local et transmettre au serveur avec chaque requête une fois authentifié. Ceci permet effectivement de déplacer la gestion de la session, traditionnellement côté serveur, du côté du client. Le serveur ne procède plus qu'à la lecture du contenu de confiance pour connaître l'état de la session, avant de le modifier et de transmettre l'état mis à jour avec un nouveau jeton à la réponse. Notre application réussit ainsi à conserver son aspect sans état.

Le contrôleur AuthenticationController offre un nouveau point d'entrée à notre API pour l'authentification d'un utilisateur. Une requête d'authentification, avant sa réception par celui-ci, passe par l'intercepteur TenantLoginInterceptor qui définit le schéma à utiliser en base de données lors de son traitement. C'est ce qui nous permet de déterminer le client de notre application SaaS à servir relativement à la requête, de manière transparente au reste de l'application déjà en place. Effectivement, nous nous sommes décidés à adopter un modèle se reposant sur

les schémas pour la gestion des clients de notre applicatif SaaS. Chaque client possède par conséquent un schéma qui lui est propre, isolé du reste des schémas. Tous ces derniers restent similaires en structure, néanmoins un schéma en particulier déroge à la règle, il s'agit du schéma dédié à la gestion des clients eux-mêmes (tenants). Il sera exploité lors de la réalisation de l'application complémentaire de gestion des clients de l'offre.

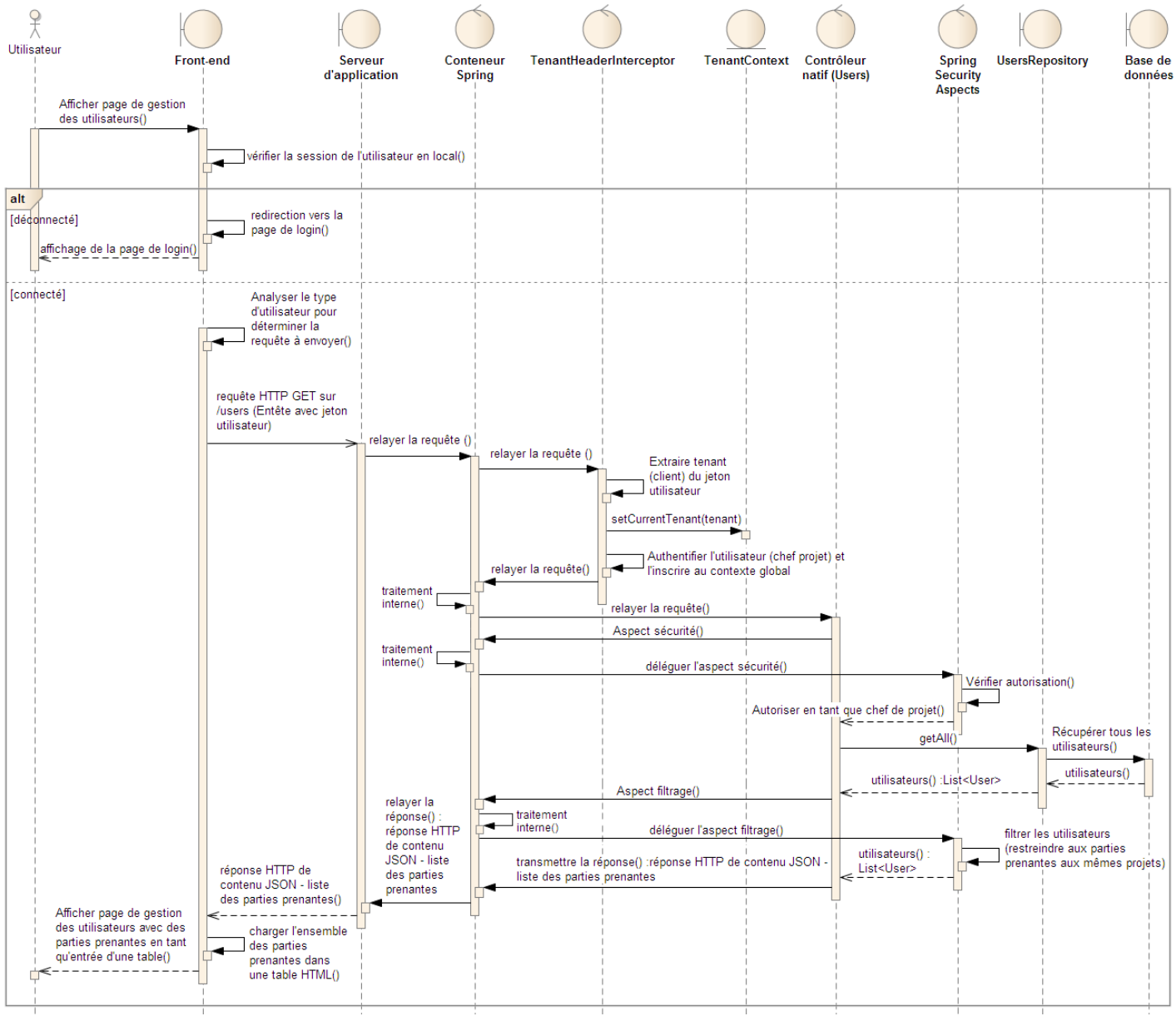

Figure VI.5 – Diagramme de séquence : Solliciter l'affichage de la page de gestion des utilisateurs

Le diagramme de séquence à la figure VI.5 quant à lui présente le mécanisme d'autorisation. Nous saisissons l'occasion pour présenter plus particulièrement le cas de gestion des utilisateurs, nouvellement introduit. On remarque tout d'abord l'introduction d'éléments nouveaux par rapport à l'existant, à savoir TenantHeaderInterceptor, TenantContext et SpringSecurity Aspects. Le reste des éléments font partie du flot d'exécution déjà mis en place préalablement. Quelque soit l'utilisateur qui requête une ressource de notre API, le comportement de ce dernier reste très similaire.
En premier lieu, c'est cette fois-ci l'intercepteur TenantHeaderInterceptor qui se charge de spécifier le client global à servir avant de laisser le système s'intéresser à l'utilisateur en particulier. L'identifiant du client est fourni par le jeton d'accès envoyé avec la requête en Front-end. Ceci est toujours le cas, car le Front-end redirigerait un utilisateur automatiquement vers la page de login si celui venait à demander l'accès à une autre page sans être authentifié.
La deuxième étape consiste à autoriser l'accès à la ressource pour l'utilisateur en question. La gestion de cet aspect est déléguée à SpringSecurity au travers de l'utilisation de la POA, de manière à enrichir le comportement de notre API sans pour autant toucher à l'infrastructure déjà en place. Dans ce cas en particulier, le seul utilisateur qui serait dénié d'accès serait celui de type de compte partie prenante, puisque ce dernier n'a aucun utilisateur à gérer (communiquerait-t-il avec notre API autrement que par notre Front-end, qui quant à lui, ne lui fournit même pas la fonctionnalité).
Le dernier rouage du mécanisme d'autorisation se charge du filtrage du contenu à retourner. En effet, ici le chef de projet ne peut gérer que les utilisateurs de type de compte partie prenante, qui sont de plus affectés aux mêmes projets que lui. Cet aspect est aussi géré par SpringSecurity qui se charge de filtrer le contenu en retour relativement au type d'utilisateur autorisé.

Ce comportement est très générique et est ajusté à chaque contexte autre que celui de la gestion de comptes utilisateurs (projets du portefeuille filtrés pour un chef de projet, droits de requêtage uniquement via la méthode GET pour une partie prenante, etc), logiquement déductible à partir des descriptions faites précédemment de chaque rôle.

1.2 Réalisation & Rendu

1.2.1 Réalisation

Back-end : Nous nous sommes reposés sur l'infrastructure déjà en place pour y intégrer les mécanismes d'authentification et d'autorisation de manière anonyme aux composants existants. Ceci a été réalisé au travers de l'usage de Spring et de son implémentation du paradigme AOP. Pour nous faciliter davantage la tâche, nous avons de plus eu recours à l'intégration du module Spring Security [66] de la suite Spring. Celui-ci offre en effet une manière pratique pour gérer les aspects d'authentification et d'autorisation au travers des annotations PreAuthorize et PreFilter qui permettent respectivement d'autoriser ou dénier l'accès à une méthode (ici responsable de l'une de nos ressources), et de filtrer le résultat en retour d'une méthode (ici relativement à la ressource en question). Ces dernières enrichissent ainsi les repositories déjà mis en place et auto-exposés en tant que ressources pour notre API (via Spring Data REST). Une fois authentifié (voir TenantHeaderInterceptor à la figure VI.5), l'utilisateur est constamment accessible via le contexte global partagé avec Spring Security, ce qui nous permet ainsi qu'à cette dernière de le prendre en compte lors des traitements ultérieurs.
Pour respecter les bonnes pratiques de sécurité et protéger nos comptes utilisateurs, nous nous sommes efforcés de ne persister aucun mot de passe au sein de notre base de données. En effet, nous nous reposons entièrement sur le couplage de hashage avec salt de mots de passes (via la classe utilitaire BCrypt de SpringSecurity) au niveau de la création d'un utilisateur ou de la validation de ses identifiants lors de son authentification au système. Ceci n'a à être exécuté qu'aux deux étapes mentionnées uniquement, car pour l'accès ultérieur l'utilisateur communique toujours avec un jeton d'accès de confiance, signé par nos soins. On a veillé ici à respecter le standard JWT [67] pour l'élaboration et la transmission du jeton.
Bien évidemment, nous avons également réitéré les étapes habituelles encontrées au chapitre IV pour l'implémentation de la ressource utilisateur.

Front-end : Nous avons d'une part ajouté une page de login et de gestion de profil aux pages existantes, et d'autre part concrétisé la page de gestion des comptes utilisateurs du menu global par la réutilisation du modèle de pages génériques de gestion de ressources, qui les représentent

sous formes d'entrées organisées dans une table accompagnées d'un formulaire déroulant pour leur édition. Ce formulaire en particulier se caractérise ici par un champ de mot de passe qui n'est jamais affiché dans la table (il ne peut qu'être mis à jour). Cette page diffère d'un type d'utilisateur à un autre par la possibilité d'affectation de projets aux comptes ainsi que les types d'utilisateurs gérés en eux-mêmes.

La session est stockée en local via la persistence du jeton JWT reçu dans le localStorage ([68] technologie HTML5), qui contrairement aux cookies, n'est pas automatiquement envoyé à chaque requête au serveur. Ceci nous oblige à gérer son envoi explicitement dans notre code, mais nous protège en contrepartie d'attaques de type CSRF (XSRF [69]) car seule une requête AJAX envoyée depuis un script exécuté sur notre domaine peut y accéder (restriction implémentée par les navigateurs populaires). Dès lors, la déconnexion prend place simplement à travers la suppression du jeton de session en local. Le contenu des jetons étant consultable, cet état de session en local nous permet de dynamiser davantage nos interfaces utilisateurs, en ajustant leurs détails relativement à chaque utilisateur (menu latéral global sans gestion de comptes ni d'options d'édition du portefeuille de projets pour les parties prenantes, pas d'options d'éditions pour les dirigeants ou les parties prenantes au sein de l'interface de gestion d'un projet, pages restreintes aux pages de gestion de comptes et de ressources humaines globales pour un administrateur, consultation uniquement des mises à jour du tableau de bord pour les parties prenantes d'un projet, etc).

1.2.2 Rendu

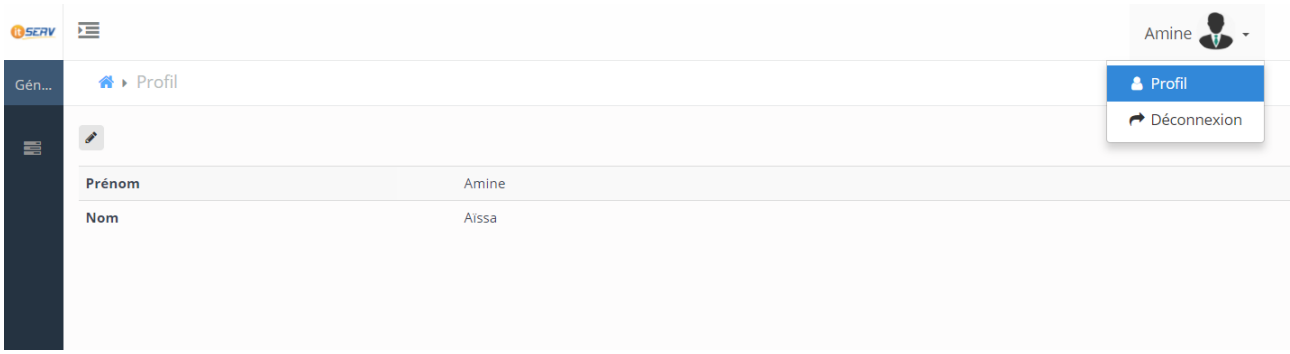

Figure VI.6 – Rendu de la page de gestion du profil - mode consultation

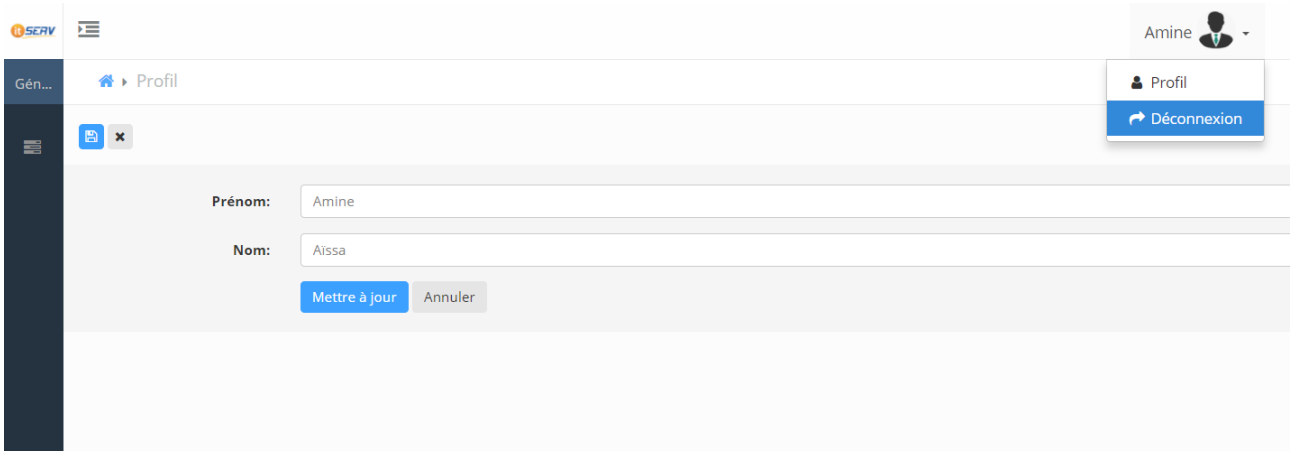

Figure VI.7 – Rendu de la page de gestion du profil - mode d'édition

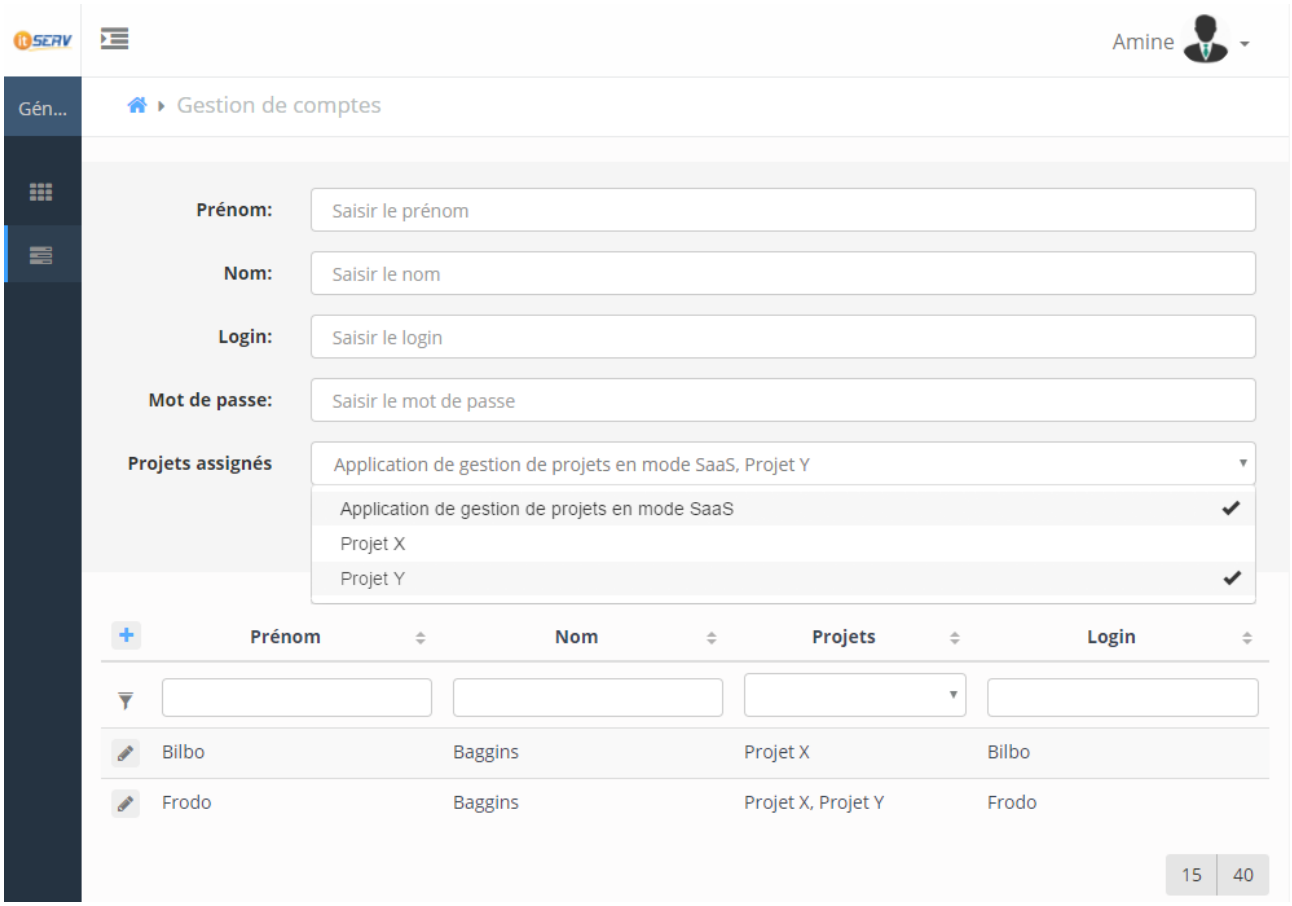

Figure VI.8 – Rendu de la page de gestion d'utilisateurs en tant que dirigeant : assignation de projets

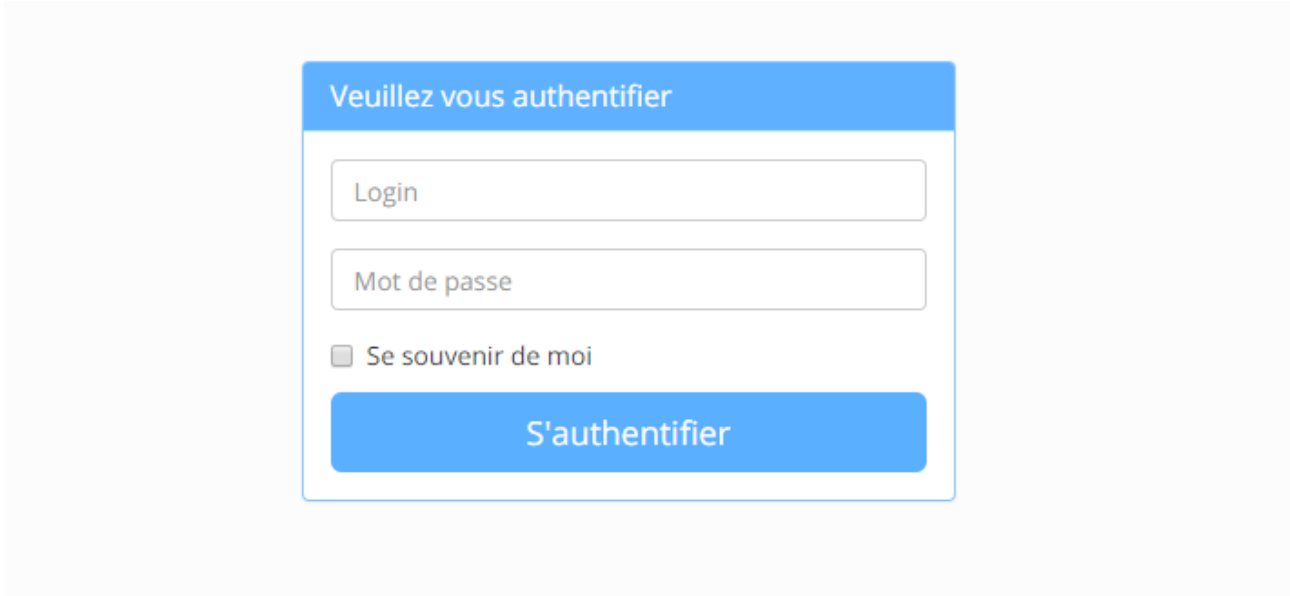

Figure VI.9 – Rendu de la page d'authentification

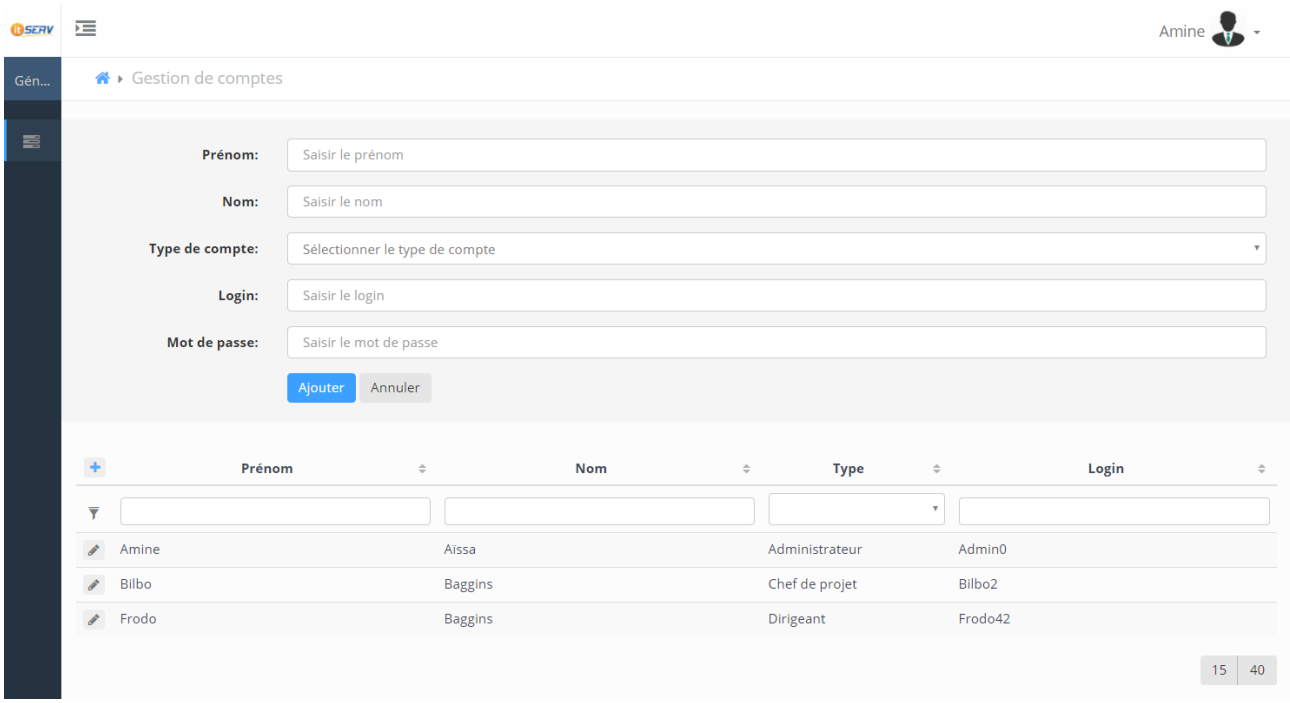

Figure VI.10 – Rendu de la page de gestion d'utilisateurs en tant qu'administrateur

2 Applicatif de gestion des clients de l'offre SaaS

2.1 Conception

Le diagramme de cas d'utilisation suivant (figure VI.11) permet de recenser les besoins de base nous étant exprimés pour la gestion de la clientèle de l'offre SaaS. Il définit des exigences simples, assimilables à des opérations CRUD basiques.

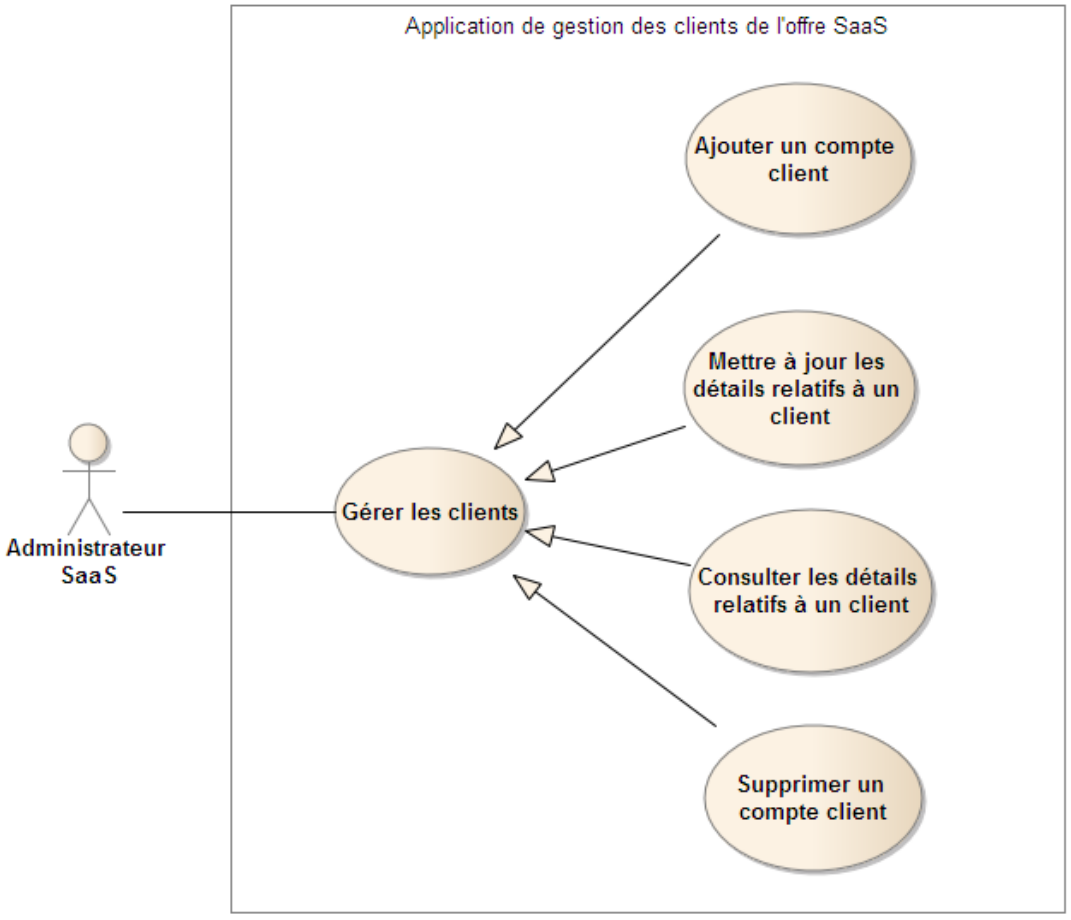

Figure VI.11 – Diagramme de cas d'utilisation de l'application complémentaire

Le diagramme de composants présenté par la figure VI.12 présente l'infrastructure globale établie vers l'achèvement de notre projet.

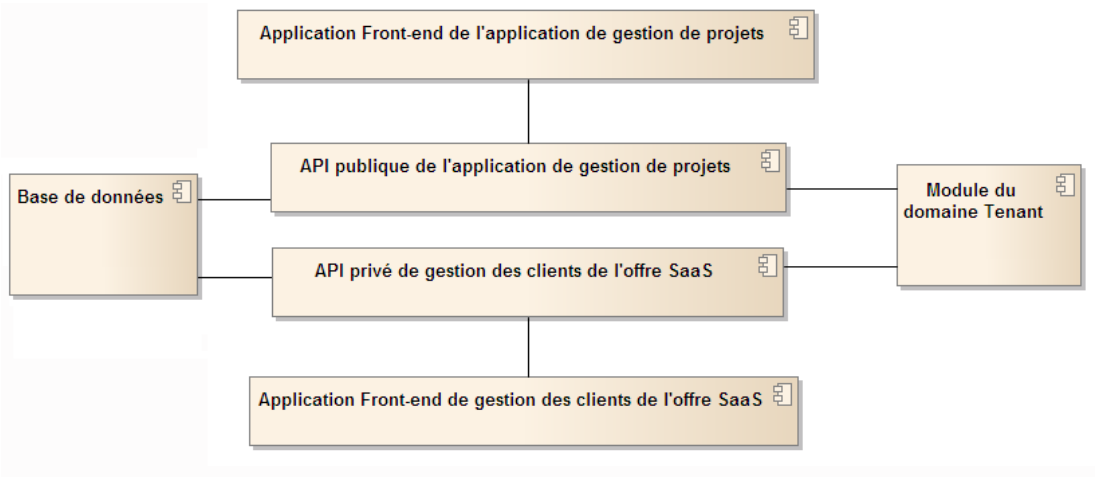

Figure VI.12 – Diagramme de composants du travail global réalisé

2.2 Réalisation & Rendu

Pour la réalisation, nous avons externalisé la classe Tenant vers un module (Gradle) séparé, pour l'intégrer au sein du nouvel API privé à développer pour les besoins de l'application de gestion des clients de l'offre SaaS. Celle-ci a été enrichie de quelques propriétés jugées pertinentes pour l'identification de nos clients (voir champs formulaire à la figure VI.13). L'application en elle-même est un simple service bâti au-dessus d'un repository (TenantRepository) sur la classe Tenant. Le comportement du repository a légèrement été étendu pour exécuter un script SQL de création du schéma client lors de l'appel de la méthode d'ajout de client (création des mêmes tables et relations au sein d'un nouveau schéma), et un script SQL de suppression (drop) du schéma et de son contenu (en cascade) lors de la suppression de ce dernier. Ceci a été réalisé grâce à l'implémentation d'une classe concrète, implémentant simplement l'interface repository traditionnelle créee via Spring Data. Spring procède à l'auto-injection de celle-ci de manière automatique et transparent lors de l'appel habituel de l'interface repository que nous utilisons usuellement. Ici, le travail est vraiment banalisé puisque nous nous connectons directement au schéma global de gestion des clients (uniquement une table Tenant est présente). Comme déductible de la section précédente, l'id de nos entrées client seront reflétées par un schéma du même nom en base de données, contenant la structure complète établie dans le travail précédent.

Côté Front-end, l'interface a été reprise de l'application Front-end déjà réalisée pour en exploiter les fonctionnalités de gestion générique de collection de ressources. Cette application Front-end sera dédiée à la communication avec l'API privé de gestion des clients de l'offre SaaS, et ne sera déployé que localement à l'entreprise d'accueil, pour une utilisation strictement en interne.

La figure VI.13 présente un aperçu de l'interface finale adoptée.

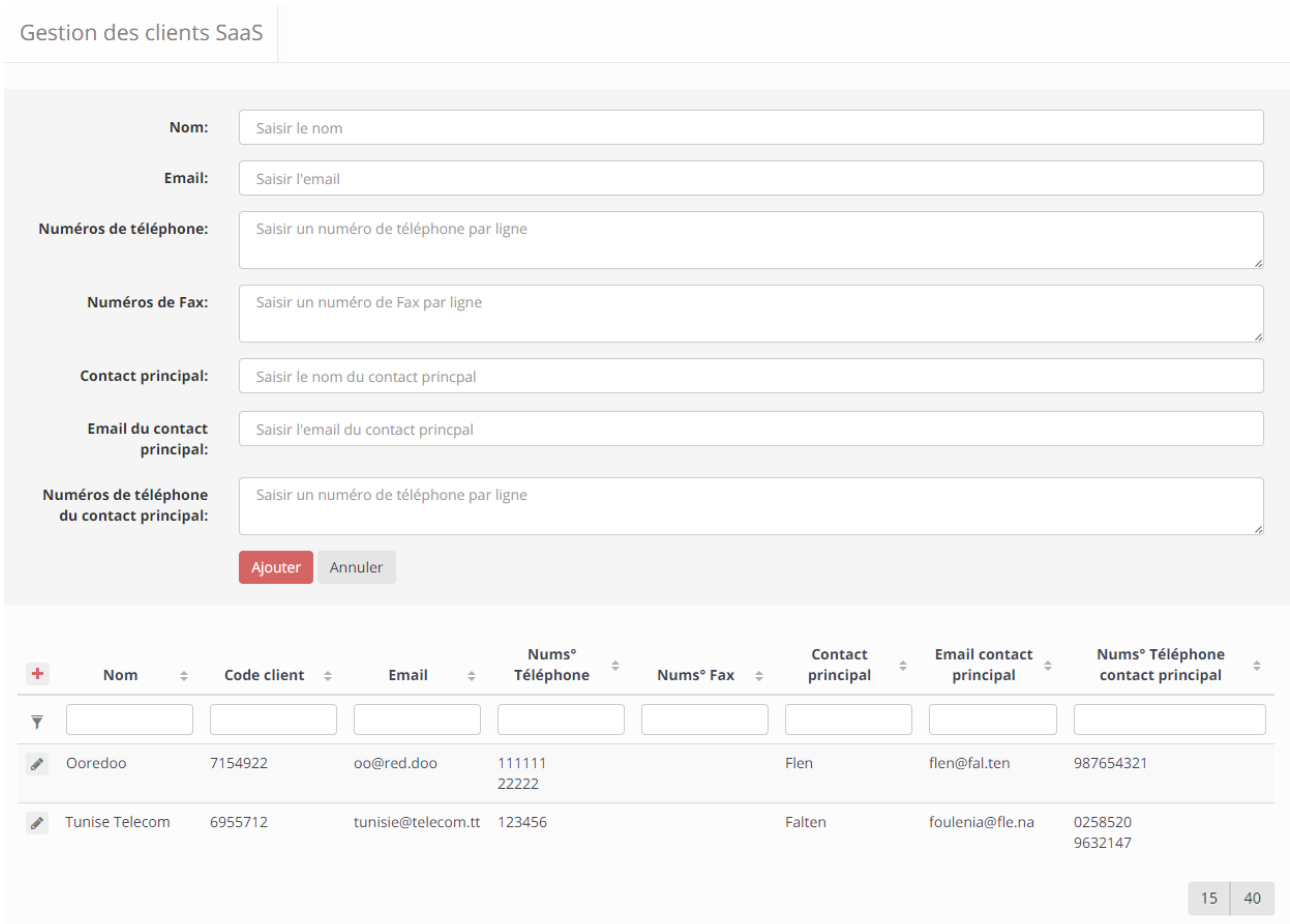

Figure VI.13 – Interface de l'application de gestion des clients de l'offre SaaS

Conclusion

L'aspect SaaS de notre application a été concrétisé au cours de ce dernier chapitre. En soi, toute la difficulté du développement d'une application de type SaaS réside dans le choix initial du mécanisme d'isolation des données des clients. Grâce à l'étude préliminaire effectuée, nous avons bien cerné les exigences de notre application et avons décidé d'opter pour l'utilisation d'une seule base de données pour notre clientèle, l'isolation se déroulant au niveau des schémas qui regroupent pour un client l'ensemble des tables associées à ses données et à ses données uniquement.

Au cours de la réalisation des objectifs de ce chapitre, nous avons résolu le problème de manière à découpler l'aspect de multiplicité des clients de l'aspect de gestion de projets. Grâce à la combinaison des technologies Spring et Hibernate nous avons réussi en quelques étapes à faire évoluer notre application mono-utilisateur en une application, avant tout multi-client, au vu de l'utilisation efficace du concept de schéma précédemment mentionné, mais aussi en une application multi-utilisateur à part entière du point de vue du client. En conclusion, le développement de l'application de gestion des clients en elle-même n'a représenté qu'une formalité, visant à abstraire la gestion directe de la base de données du point de vue de l'administrateur SaaS.

Conclusion Générale et Perspectives

Dans leur évolution naturelle, les entreprises sont continuellement à la recherche de solutions et d'outils capables de les aider dans la concrétisation de leurs ambitions. Celles-ci sont le plus souvent définies formellement sous forme de projets, un projet étant caractérisé par des objectifs clairs à atteindre et le plus souvent soumis à un ensemble de contraintes statiques et dynamiques. Par conséquent, le perfectionnement de leurs processus de gestion de projets constitue une préoccupation majeure pour toutes les entreprises.

Pour optimiser son processus de gestion de projets au sein du département consulting, IT SERV a entrepris de développer une solution personnalisée capable d'assister ses chefs de projets dans l'accompagnement de ses clients à la réalisation de leurs projets. Dans une optique d'extension de ses services vers l'édition de logiciels, elle s'est proposé de fournir la solution à concevoir en mode Cloud SaaS vers sa clientèle existante ainsi que pour une nouvelle catégorie de clients, uniquement intéressés par les avantages que lui apportent l'adoption d'une telle application pour ses besoins de gestion de projets en interne.

Pour notre projet de fin d'études, nous avons à cet effet entrepris de développer une application d'aide à la gestion de projets adaptée à une utilisation en mode SaaS. Le projet a démarré avec une analyse de l'état des lieux de la gestion de projets en local, suivie d'une étude théorique sur les notions générales de gestion de projet et des traits fonctionnels caractéristiques de solutions similaires existantes. Nous avons établi la vision générale des objectifs à atteindre pour notre projet par la spécification formelle des besoins. Celle-ci a tout de suite été suivie d'une étude technique visant à spécifier les grandes lignes de l'architecture générale à adopter pour le développement de la solution proposée.
Avec toutes ces données en main, nous avons entamé la phase de développement de notre projet par l'implémentation des fonctionnalités de gestion de projets désirées, sous forme du développement itératif et incrémental des couches du système. La phase d'intégration de l'aspect SaaS à l'application a représenté la phase finale de développement et a marqué l'atteinte de l'ensemble des objectifs définis pour notre système.

Cette démarche d'inscrit dans le suivi d'une méthodologie de développement disciplinée tout au long de la réalisation de l'applicatif. Elle représenta pour nous une démarche logique dans la conduite de notre projet.

Au travers de l'effectuation de ce stage, nous avons non seulement pu nous introduire efficacement au monde professionnel, mais aussi acquérir par la même une connaissance approfondie des concepts clés liés au développement d'applications SaaS ainsi qu'aux notions avancées de gestion professionnelle de projets.

Plusieurs perspectives sont envisageables pour enrichir et améliorer la solution développée. Citons l'intégration d'un diagramme de Gantt de gestion combinant la gestion de ressources humaines et de tâches, l'ajout de fonctionnalités de collaboration avancées pour la gestion d'un projet, ou encore le développement de tableaux de bord additionnels et détaillés de facettes spécifiques d'un projet. L'enrichissement de l'interface graphique est aussi envisageable par l'utilisation par exemple de cartes Kanban à gestion via glisser-déposer ou encore le développement d'une application mobile à part entière se reposant sur l'API mis en place.

Bibliographie

[1] http://http://www.disciplinedagiledelivery.com. [En ligne ; consulté le 10-Mars-2017]. i, 3, 12

[2] http://www-igm.univ-mlv.fr/~dr/XPOSE2005/gmasquelier/images/comparaison.png. [En ligne ; consulté le 21-05-2017]. iv, 55

[3] James Rumbaugh Ivar Jacobson, Grady Booch. *The Unified Software Development Process*. Addison-Wesley Professional Février (1999). 10

[4] http://www.agilemanifesto.org/iso/en/principles.html. [En ligne ; consulté le 21-Avril-2017]. 10

[5] Phillipe Kruchten. *The Rational Unified Process : An Introduction*. Addison-Wesley Professional Décembre (2003). 10

[6] J.J. Sutherland Jeff Sutherland. *Scrum : The Art of Doing Twice the Work in Half the Time*. Crown Business Septembre (2014). 10

[7] http://www.agiliste.fr/introduction-methodes-agiles. [En ligne ; consulté le 21-Avril-2017]. 11

[8] Mark Lines Scott W.Ambler. *Disciplined Agile Delivery : A Practioner's Guide to Agile Software Delivery in the Enterprise*. IBM Press Juin (2012). 12

[9] Kent Beck. *Extreme Programming Explained : Embrace change*. Addison-Wesley Professional Septembre (2004). 12

[10] Joakim Sunden Marcus Hammarberg. *Kanban in Action*. Manning Publications Mars (2014). 12

[11] Scott W.Ambler. *Agile Modeling : Effective Practices for eXtreme Programming and the Unified Process*. Wiley Avril (2002). 12

[12] https://git-scm.com/. [En ligne ; consulté le 15-Mai-2017]. 15

[13] https://rhodecode.com/insights/version-control-systems-2016. [En ligne ; consulté le 15-Mai-2017]. 15

[14] https://trello.com. [En ligne ; consulté le 15-Mai-2017]. 16

[15] https://www.atlassian.com/software/jira. [En ligne ; consulté le 14-Mai-2017]. 25

[16] https://products.office.com/en/project/project-management. [En ligne ; consulté le 14-Mai-2017]. 25

[17] https://www.projectmanager.com. [En ligne ; consulté le 14-Mai-2017]. 25

[18] https://www.zoho.com/projects. [En ligne ; consulté le 14-Mai-2017]. 25

[19] https://www.easyredmine.com. [En ligne ; consulté le 14-Mai-2017]. 26

[20] https://www.clarizen.com. [En ligne ; consulté le 14-Mai-2017]. 26

[21] https://mkt.clarizen.com/gartner-mq.html. [En ligne ; consulté le 14-Mai-2017]. 26

[22] https://mkt.clarizen.com/report-forrester-wave.html. [En ligne ; consulté le 14-Mai-2017]. 26

[23] https://fr.wikipedia.org/wiki/Architecture_trois_tiers. [En ligne ; consulté le 20-05-2017]. 35

[24] https://angularjs.org. [En ligne ; consulté le 21-05-2017]. 37, 68

[25] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. Thèse de Doctorat, UNIVERSITY OF CALIFORNIA, IRVINE (2000). 37

[26] https://www.mysql.com. [En ligne ; consulté le 20-05-2017]. 40

[27] https://mariadb.org. [En ligne ; consulté le 20-05-2017]. 40

[28] https://www.postgresql.org. [En ligne ; consulté le 20-05-2017]. 40

[29] https://www.mongodb.com. [En ligne ; consulté le 20-05-2017]. 41

[30] http://cassandra.apache.org. [En ligne ; consulté le 20-05-2017]. 41

[31] https://couchdb.apache.org. [En ligne ; consulté le 20-05-2017]. 41

[32] https://db-engines.com/en/ranking. [En ligne ; consulté le 20-05-2017]. 43

[33] https://www.datastax.com/dev/blog/does-cql-support-dynamic-columns-wide
[En ligne ; consulté le 20-05-2017]. 43

[34] http://docs.spring.io/spring-framework/docs/current/spring-framework-reference/html. [En ligne ; consulté le 21-05-2017]. 53

[35] https://docs.spring.io/spring-data/rest/docs/current/reference/html. [En ligne ; consulté le 21-05-2017]. 53

[36] https://docs.spring.io/spring-data/rest/docs/current/reference/html. [En ligne ; consulté le 21-05-2017]. 53

[37] http://docs.spring.io/spring-boot/docs/current/reference/html. [En ligne ; consulté le 21-05-2017]. 54

[38] https://martinfowler.com/eaaCatalog/repository.html. [En ligne ; consulté le 21-05-2017]. 54

[39] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Novembre (2002). 54

[40] https://martinfowler.com/articles/injection.html. [En ligne ; consulté le 21-05-2017]. 54

[41] http://www-igm.univ-mlv.fr/~dr/XPOSE2005/gmasquelier/images/comparaison.png. [En ligne ; consulté le 21-05-2017]. 55

[42] http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html. [En ligne ; consulté le 21-05-2017]. 56

[43] http://hibernate.org. [En ligne ; consulté le 21-05-2017]. 56

[44] http://tomcat.apache.org/. [En ligne ; consulté le 21-05-2017]. 56

[45] http://www.oracle.com/technetwork/java/javaee. [En ligne ; consulté le 21-05-2017]. 56

[46] https://gradle.org. [En ligne ; consulté le 21-05-2017]. 56

[47] https://ant.apache.org. [En ligne ; consulté le 21-05-2017]. 56

[48] https://maven.apache.org. [En ligne ; consulté le 21-05-2017]. 56

[49] https://www.getpostman.com. [En ligne ; consulté le 21-05-2017]. 60

[50] https://www.npmjs.com/. [En ligne ; consulté le 21-05-2017]. 66

[51] https://nodejs.org. [En ligne ; consulté le 21-05-2017]. 66

[52] https://gulpjs.com. [En ligne ; consulté le 21-05-2017]. 66

[53] https://sass-lang.com. [En ligne ; consulté le 21-05-2017]. 66

[54] https://github.com/postcss/autoprefixer. [En ligne ; consulté le 21-05-2017]. 67

[55] https://postcss.org. [En ligne ; consulté le 21-05-2017]. 67

[56] https://github.com/mishoo/UglifyJS. [En ligne ; consulté le 21-05-2017]. 67

[57] https://github.com/tapio/live-server. [En ligne ; consulté le 21-05-2017]. 67

[58] https://en.wikipedia.org/wiki/Ajax_(programming). [En ligne ; consulté le 21-05-2017]. 67

[59] https://httpd.apache.org/. [En ligne ; consulté le 21-05-2017]. 67

[60] https://news.netcraft.com/archives/2016/07/19/july-2016-web-server-survey.html. [En ligne ; consulté le 21-05-2017]. 67

[61] getbootstrap.com/. [En ligne ; consulté le 21-05-2017]. 69

[62] https://nvd3.org. [En ligne ; consulté le 21-05-2017]. 69

[63] https://d3js.org. [En ligne ; consulté le 21-05-2017]. 69

[64] https://angular-ui.github.io. [En ligne ; consulté le 21-05-2017]. 69

[65] https://github.com/esvit/ng-table. [En ligne ; consulté le 21-05-2017]. 69

[66] https://projects.spring.io/spring-security. [En ligne ; consulté le 26-05-2017]. 95

[67] https://tools.ietf.org/html/rfc7519. [En ligne ; consulté le 26-05-2017]. 95

[68] https://www.w3.org/TR/webstorage/#the-localstorage-attribute. [En ligne ; consulté le 26-05-2017]. 96

[69] https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF). [En ligne ; consulté le 26-05-2017]. 96

Annexe A : Tests fonctionnels - API de gestion de projets

Requêtage simple de ressources

Notre API permet de requêter spécifiquement une instance d'un type de ressource, identifée par son id, ou bien l'ensemble de la collection des ressources de ce type.

Ressource unique

URI : /projects/88b5cf95-0ccc-4b4f-8493-12d2342ae01e
Requête : Méthode - GET
Réponse : Code Statut - 200 (OK)


```
{
  "budgetToConsume": 7,
  "budgetConsumed": 2,
  "chargeConsumed": 0,
  "advancement": 0,
  "status": "GREEN",
  "name": "Projet X",
  "comment": null,
  "sponsors": "Flen Fouleni\nFaltena Flena",
  "finalClient": "Flen",
  "budgetInitial": 1000,
  "endDate": "2017-05-12T00:00:00.000+0000",
  "goal": "Faire le travail demande",
  "mainContact": "Flen",
  "startDate": "2017-05-31T00:00:00.000+0000",
  "chargePrevision": 50,
  "hypotheses_constraints": null,
  "history_decisions": null,
  "_links": {
    "self": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
          d2342ae01e"
    },
    "project": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
          d2342ae01e"
```

```
},
"constructionSites": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/constructionSites"
},
"milestones": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/milestones"
},
"humanResources": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/humanResources"
},
"actions": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/actions"
},
"risks": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/risks"
},
"resources": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/resources"
},
"subProjects": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/subProjects"
},
"documents": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/documents"
},
"archivedUpdates": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/archivedUpdates"
},
"todos": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/todos"
},
"communicationPlans": {
```

```
        "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
            d2342ae01e/communicationPlans"
      },
      "reunionPlannings": {
        "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
            d2342ae01e/reunionPlannings"
      },
      "writeups": {
        "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
            d2342ae01e/writeups"
      },
      "pendingIssues": {
        "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
            d2342ae01e/pendingIssues"
      },
      "changeRequests": {
        "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
            d2342ae01e/changeRequests"
      }
    }
  }
```


Collection de ressources

URI : /projects
Requête : Méthode - GET
Réponse : Code Statut - 200 (OK)


```
  {
    "_embedded": {
      "projects": [
        {
          "budgetToConsume": 7,
          "budgetConsumed": 2,
          "chargeConsumed": 0,
          "advancement": 0,
          "status": "GREEN",
          "name": "Projet X",
          "comment": null,
          "sponsors": "Flen Fouleni\nFaltena Flena",
```

```
"finalClient": "Flen",
"budgetInitial": 1000,
"endDate": "2017-05-12T00:00:00.000+0000",
"goal": "Faire le travail demande",
"mainContact": "Flen",
"startDate": "2017-05-31T00:00:00.000+0000",
"chargePrevision": 50,
"hypotheses_constraints": null,
"history_decisions": null,
"_links": {
  "self": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
       -8493-12d2342ae01e"
  },
  "project": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
       -8493-12d2342ae01e"
  },
  "constructionSites": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
       -8493-12d2342ae01e/constructionSites"
  },
  "milestones": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
       -8493-12d2342ae01e/milestones"
  },
  "humanResources": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
       -8493-12d2342ae01e/humanResources"
  },
  "actions": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
       -8493-12d2342ae01e/actions"
  },
  "risks": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
       -8493-12d2342ae01e/risks"
  },
  "resources": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
       -8493-12d2342ae01e/resources"
  },
```

```
    "subProjects": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
          -8493-12d2342ae01e/subProjects"
    },
    "actions": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
          -8493-12d2342ae01e/actions"
    },
    "documents": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
          -8493-12d2342ae01e/documents"
    },
    "archivedUpdates": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
          -8493-12d2342ae01e/archivedUpdates"
    },
    "todos": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
          -8493-12d2342ae01e/todos"
    },
    "communicationPlans": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
          -8493-12d2342ae01e/communicationPlans"
    },
    "reunionPlannings": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
          -8493-12d2342ae01e/reunionPlannings"
    },
    "writeups": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
          -8493-12d2342ae01e/writeups"
    },
    "pendingIssues": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
          -8493-12d2342ae01e/pendingIssues"
    },
    "changeRequests": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
          -8493-12d2342ae01e/changeRequests"
    }
  }
},
```

```
{
  "budgetToConsume": 7,
  "budgetConsumed": 2,
  "chargeConsumed": 0,
  "advancement": 0,
  "status": "RED",
  "name": "Projet Y",
  "comment": null,
  "sponsors": "Flen Fouleni",
  "finalClient": "Flena",
  "budgetInitial": 3500,
  "endDate": "2017-05-12T00:00:00.000+0000",
  "goal": "Realiser le travail demande",
  "mainContact": "Flena",
  "startDate": "2017-05-31T00:00:00.000+0000",
  "chargePrevision": 180,
  "hypotheses_constraints": null,
  "history_decisions": null,
  "_links": {
    "self": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
          -8493-12d2342ae01e"
    },
    "project": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
          -8493-12d2342ae01e"
    },
    "constructionSites": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
          -8493-12d2342ae01e/constructionSites"
    },
    "milestones": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
          -8493-12d2342ae01e/milestones"
    },
    "humanResources": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
          -8493-12d2342ae01e/humanResources"
    },
    "actions": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
          -8493-12d2342ae01e/actions"
```

```
},
"risks": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
     -8493-12d2342ae01e/risks"
},
"resources": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
     -8493-12d2342ae01e/resources"
},
"subProjects": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
     -8493-12d2342ae01e/subProjects"
},
"documents": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
     -8493-12d2342ae01e/documents"
},
"archivedUpdates": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
     -8493-12d2342ae01e/archivedUpdates"
},
"todos": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
     -8493-12d2342ae01e/todos"
},
"communicationPlans": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
     -8493-12d2342ae01e/communicationPlans"
},
"reunionPlannings": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
     -8493-12d2342ae01e/reunionPlannings"
},
"writeups": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
     -8493-12d2342ae01e/writeups"
},
"pendingIssues": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
     -8493-12d2342ae01e/pendingIssues"
},
"changeRequests": {
```

```
            "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f
               -8493-12d2342ae01e/changeRequests"
          }
        }
      }
    ]
  },
  "_links": {
    "self": {
      "href": "http://127.0.0.1:9000/projects"
    },
    "profile": {
      "href": "http://127.0.0.1:9000/profile/projects"
    }
  },
  "page": {
    "size": 20,
    "totalElements": 1,
    "totalPages": 1,
    "number": 0
  }
}
```

Requêtage paramétré de ressources

Notre API supporte pour certaines de nos ressources des requêtes paramétrées. Nous prendrons l'exemple du dashboard et des détails budget pour la ressource projet.

Dashboard

URI : /projects/88b5cf95-0ccc-4b4f-8493-12d2342ae01e ?projection=dashboard
Requête : Méthode - GET
Réponse : Code Statut - 200 (OK)


```
{
  "subProjectsAdvancement": [
    {
      "advancement": "15",
      "name": "Back-end",
      "id": "a18d034a-a235-4a6c-9046-9a9d88fc9b36"
    },
    {
      "advancement": "22",
      "name": "Front-end",
      "id": "c1386f16-ae37-4d69-b9f8-3d552973e5ca"
    }
  ],
  "name": "App. Gestion Projets",
  "milestones": [
    {
      "reference": "9fdcdece-9825-4fc6-8ef6-afda98ae0fc8",
      "name": "Vision des parties prenantes",
      "dueDate": "2017-02-28T00:00:00.000+0000",
      "description": "Elaboration de la vision achevee"
    },
    {
      "reference": "3d2f2c35-9d94-43f1-a0aa-bddb0dc1a7b9",
      "name": "Architecture eprouvee",
      "dueDate": "2017-04-20T00:00:00.000+0000",
      "description": "Systeme fonctionnel de bout en bout"
    },
    {
      "reference": "96f97407-deb9-4e02-9584-74fec3f895ed",
```

```
    "name": "Viabilite du projet",
    "dueDate": "2017-05-01T00:00:00.000+0000",
    "description": "Application minimale fonctionnelle"
  }
],
"status": "GREEN",
"advancement": 20,
"actionsOngoingInTimeOrNotCount": {
  "false": 1,
  "true": 2
},
"budgetConsumed": 74,
"chargeConsumed": 0,
"chargePrevisionGap": -80,
"riskStatusesCount": {
  "DECIDED_UPON": 1,
  "MASTERED": 1
},
"budgetPrevisionGap": 12,
"constructionSitesAdvancement": [
  {
    "advancement": "0",
    "name": "Aspect SaaS",
    "id": "e430ebaf-ecb4-4931-86b6-cd59274795cd"
  }
],
"_links": {
  "self": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
       d2342ae01e"
  },
  "project": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
       d2342ae01e{?projection}",
    "templated": true
  },
  "writeups": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
       d2342ae01e/writeups"
  },
  "resources": {
```

```
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
        d2342ae01e/resources"
  },
  "documents": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
        d2342ae01e/documents"
  },
  "communicationPlans": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
        d2342ae01e/communicationPlans"
  },
  "subProjects": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
        d2342ae01e/subProjects"
  },
  "todos": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
        d2342ae01e/todos"
  },
  "actions": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
        d2342ae01e/actions"
  },
  "archivedUpdates": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
        d2342ae01e/archivedUpdates"
  },
  "changeRequests": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
        d2342ae01e/changeRequests"
  },
  "risks": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
        d2342ae01e/risks"
  },
  "milestones": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
        d2342ae01e/milestones"
  },
  "reunionPlannings": {
    "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
        d2342ae01e/reunionPlannings"
```

```
    },
    "humanResources": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
          d2342ae01e/humanResources"
    },
    "constructionSites": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
          d2342ae01e/constructionSites"
    },
    "pendingIssues": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
          d2342ae01e/pendingIssues"
    }
  }
}
```


Détails budget

URI : /projects/88b5cf95-0ccc-4b4f-8493-12d2342ae01e ?projection=budget
Requête : Méthode - GET
Réponse : Code Statut - 200 (OK)


```
{
  "budgetInitial": 1200,
  "budgetConsumed": 1000,
  "budgetToConsume": 350,
  "_links": {
    "self": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
          d2342ae01e"
    },
    "project": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
          d2342ae01e{?projection}",
      "templated": true
    },
    "writeups": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
          d2342ae01e/writeups"
    },
```

```
"resources": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/resources"
},
"documents": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/documents"
},
"communicationPlans": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/communicationPlans"
},
"subProjects": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/subProjects"
},
"todos": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/todos"
},
"actions": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/actions"
},
"archivedUpdates": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/archivedUpdates"
},
"changeRequests": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/changeRequests"
},
"risks": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/risks"
},
"milestones": {
  "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
     d2342ae01e/milestones"
},
"reunionPlannings": {
```

```
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
          d2342ae01e/reunionPlannings"
    },
    "humanResources": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
          d2342ae01e/humanResources"
    },
    "constructionSites": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
          d2342ae01e/constructionSites"
    },
    "pendingIssues": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
          d2342ae01e/pendingIssues"
    }
  }
}
```

Ajout de ressources

Notre API permet via une seule opération de créer une ressource donnée et de la lier à sa ressource parente (préfixe à son URI). Nous prendrons l'exemple de ressources non humaines.

Ressources non humaines : état a priori

URI : /projects/88b5cf95-0ccc-4b4f-8493-12d2342ae01e/resources
Requête : Méthode - GET
Réponse : Code Statut - 200 (OK)

```
{
  "_embedded": {
    "resources": []
  },
  "_links": {
    "self": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
          d2342ae01e/resources"
    }
  }
}
```

Ressources non humaines : ajout

URI : /projects/88b5cf95-0ccc-4b4f-8493-12d2342ae01e/resources
Requête : Méthode - POST

```
[
  {
    "name": "License logicielle X",
    "type": "logiciel"
  },
  {
    "name": "Objet materiel U",
    "type": "materiel"
  }
]
```

Réponse : Code Statut - 201 (CREATED)


```
[
  "http://127.0.0.1:9000/resources/11a86239-6213-4fac-a603-bb2d43a8fb9a",
  "http://127.0.0.1:9000/resources/6ca176d9-9a6f-493d-8986-666daa6a3dd4"
]
```


Ressources non humaines : état a posteriori

URI : /projects/88b5cf95-0ccc-4b4f-8493-12d2342ae01e/resources
Requête : Méthode - GET
Réponse : Code Statut - 200 (OK)


```
{
  "_embedded": {
    "resources": [
      {
        "reference": "22e9ada4-9381-4415-a62f-ce8bff620463",
        "name": "License logicielle X",
        "type": "logiciel",
        "_links": {
          "self": {
            "href": "http://127.0.0.1:9000/resources/11a86239-6213-4fac-a603-
                bb2d43a8fb9a"
          },
          "resource": {
            "href": "http://127.0.0.1:9000/resources/11a86239-6213-4fac-a603-
                bb2d43a8fb9a"
          },
          "lastVersion": {
            "href": "http://127.0.0.1:9000/resources/11a86239-6213-4fac-a603-
                bb2d43a8fb9a/lastVersion"
          }
        }
      },
      {
        "reference": "3a4c518c-7be9-4367-89ce-2b6643acb1fe",
        "name": "Objet materiel U",
        "type": "matrriel",
```

```
        "_links": {
          "self": {
            "href": "http://127.0.0.1:9000/resources/6ca176d9-9a6f-493d
                -8986-666daa6a3dd4"
          },
          "resource": {
            "href": "http://127.0.0.1:9000/resources/6ca176d9-9a6f-493d
                -8986-666daa6a3dd4"
          },
          "lastVersion": {
            "href": "http://127.0.0.1:9000/resources/6ca176d9-9a6f-493d
                -8986-666daa6a3dd4/lastVersion"
          }
        }
      }
    ]
  },
  "_links": {
    "self": {
      "href": "http://127.0.0.1:9000/projects/88b5cf95-0ccc-4b4f-8493-12
          d2342ae01e/resources"
    }
  }
}
```

Mise à jour de ressources

Notre API permet modifier partiellement le contenu propre à une ressource. Examinons le cas d'un reste-à-faire.

Reste-à-faire : état a priori

URI : /projects/88b5cf95-0ccc-4b4f-8493-12d2342ae01e/todos/4c0198d6-e503-491b-99f5-e3ac7017ce53
Requête : Méthode - GET
Réponse : Code Statut - 200 (OK)


```
{
  "reference": "8e701801-23ee-4e86-9c0f-3bd722173fcd",
  "description": "Accomplir une tache importante",
  "charge": 10,
  "estimationDate": "2017-05-02T00:00:00.000+0000",
  "_links": {
    "self": {
      "href": "http://127.0.0.1:9000/todos/4c0198d6-e503-491b-99f5-
          e3ac7017ce53"
    },
    "todo": {
      "href": "http://127.0.0.1:9000/todos/4c0198d6-e503-491b-99f5-
          e3ac7017ce53"
    },
    "lastVersion": {
      "href": "http://127.0.0.1:9000/todos/4c0198d6-e503-491b-99f5-
          e3ac7017ce53/lastVersion"
    }
  }
}
```


Reste-à-faire : ajout

URI : /projects/88b5cf95-0ccc-4b4f-8493-12d2342ae01e/todos/4c0198d6-e503-491b-99f5-e3ac7017ce53
Requête : Méthode - PATCH

```
{
  "charge": 20
}
```

Réponse : Code Statut - 200 (OK)

```
{
  "reference": "8e701801-23ee-4e86-9c0f-3bd722173fcd",
  "description": "Accomplir une tache importante",
  "charge": 20,
  "estimationDate": "2017-05-02T00:00:00.000+0000",
  "_links": {
    "self": {
      "href": "http://127.0.0.1:9000/todos/4c0198d6-e503-491b-99f5-
          e3ac7017ce53"
    },
    "todo": {
      "href": "http://127.0.0.1:9000/todos/4c0198d6-e503-491b-99f5-
          e3ac7017ce53"
    },
    "lastVersion": {
      "href": "http://127.0.0.1:9000/todos/4c0198d6-e503-491b-99f5-
          e3ac7017ce53/lastVersion"
    }
  }
}
```

Reste-à-faire : état a posteriori

URI : /projects/88b5cf95-0ccc-4b4f-8493-12d2342ae01e/todos/4c0198d6-e503-491b-99f5-e3ac7017ce53
Requête : Méthode - GET
Réponse : Code Statut - 200 (OK)

```
{
  "reference": "8e701801-23ee-4e86-9c0f-3bd722173fcd",
  "description": "Accomplir une tache importante",
  "charge": 20,
  "estimationDate": "2017-05-02T00:00:00.000+0000",
  "_links": {
```

```
    "self": {
      "href": "http://127.0.0.1:9000/todos/4c0198d6-e503-491b-99f5-
          e3ac7017ce53"
    },
    "todo": {
      "href": "http://127.0.0.1:9000/todos/4c0198d6-e503-491b-99f5-
          e3ac7017ce53"
    },
    "lastVersion": {
      "href": "http://127.0.0.1:9000/todos/4c0198d6-e503-491b-99f5-
          e3ac7017ce53/lastVersion"
    }
  }
}
```

Suppression de ressources

Notre API permet via une seule opération de détacher une ressource de ses ressources parentes, et de la supprimer proprement de manière transactionnelle.

Reste-à-faire : état a priori

URI : /projects/88b5cf95-0ccc-4b4f-8493-12d2342ae01e/todos/4c0198d6-e503-491b-99f5-e3ac7017ce53
Requête : Méthode - GET
Réponse : Code Statut - 200 (OK)


```
{
  "reference": "8e701801-23ee-4e86-9c0f-3bd722173fcd",
  "description": "Accomplir une tache importante",
  "charge": 10,
  "estimationDate": "2017-05-02T00:00:00.000+0000",
  "_links": {
    "self": {
      "href": "http://127.0.0.1:9000/todos/4c0198d6-e503-491b-99f5-
          e3ac7017ce53"
    },
    "todo": {
      "href": "http://127.0.0.1:9000/todos/4c0198d6-e503-491b-99f5-
          e3ac7017ce53"
    },
    "lastVersion": {
      "href": "http://127.0.0.1:9000/todos/4c0198d6-e503-491b-99f5-
          e3ac7017ce53/lastVersion"
    }
  }
}
```


Reste-à-faire : suppression

URI : /projects/88b5cf95-0ccc-4b4f-8493-12d2342ae01e/todos/4c0198d6-e503-491b-99f5-e3ac7017ce53
Requête : Méthode - DELETE
Réponse : Code Statut - 204 (NO CONTENT)

Reste-à-faire : état a posteriori

URI : /projects/88b5cf95-0ccc-4b4f-8493-12d2342ae01e/todos/4c0198d6-e503-491b-99f5-e3ac7017ce53
Requête : Méthode - GET
Réponse : Code Statut - 404 (NOT FOUND)

Archivage de l'état d'un projet

Notre API permet d'archiver l'état d'un projet par la création de sauvegardes des ressources archivables actuelles d'un projet.

Archivage : état a priori

URI : /projects/88b5cf95-0ccc-4b4f-8493-12d2342ae01e/archivedUpdates
Requête : Méthode - GET
Réponse : Code Statut - 200 (OK)


```
{
  "_embedded": {
    "projectUpdates": []
  },
  "_links": {
    "self": {
      "href": "http://127.0.0.1:9000/projects/c561b4d4-f221-4852-9419-6
          bb2d09e3078/archivedUpdates"
    }
  }
}
```


Archivage

URI : /projects/88b5cf95-0ccc-4b4f-8493-12d2342ae01e/archivedUpdates
Requête : Méthode - POST
Réponse : Code Statut - 204 (NO CONTENT)

Archivage : état a posteriori

URI : /projects/88b5cf95-0ccc-4b4f-8493-12d2342ae01e/archivedUpdates
Requête : Méthode - GET
Réponse : Code Statut - 200 (OK)


```
{
  "_embedded": {
```

```
"projectUpdates": [
  {
    "budgetToConsume": 0,
    "budgetConsumed": 0,
    "chargeConsumed": 0,
    "advancement": 0,
    "status": "GREEN",
    "updateTime": "2017-06-01T15:51:44.332+0000",
    "_links": {
      "self": {
        "href": "http://127.0.0.1:9000/projectUpdates/8334fdb3-59f5-473f-
            a782-2fa2118e86c4"
      },
      "projectUpdate": {
        "href": "http://127.0.0.1:9000/projectUpdates/8334fdb3-59f5-473f-
            a782-2fa2118e86c4"
      },
      "actions": {
        "href": "http://127.0.0.1:9000/projectUpdates/8334fdb3-59f5-473f-
            a782-2fa2118e86c4/actions"
      },
      "reunionPlannings": {
        "href": "http://127.0.0.1:9000/projectUpdates/8334fdb3-59f5-473f-
            a782-2fa2118e86c4/reunionPlannings"
      },
      "humanResources": {
        "href": "http://127.0.0.1:9000/projectUpdates/8334fdb3-59f5-473f-
            a782-2fa2118e86c4/humanResources"
      },
      "pendingIssues": {
        "href": "http://127.0.0.1:9000/projectUpdates/8334fdb3-59f5-473f-
            a782-2fa2118e86c4/pendingIssues"
      },
      "changeRequests": {
        "href": "http://127.0.0.1:9000/projectUpdates/8334fdb3-59f5-473f-
            a782-2fa2118e86c4/changeRequests"
      },
      "documents": {
        "href": "http://127.0.0.1:9000/projectUpdates/8334fdb3-59f5-473f-
            a782-2fa2118e86c4/documents"
      },
      "writeups": {
```

```
            "href": "http://127.0.0.1:9000/projectUpdates/8334fdb3-59f5-473f-
                a782-2fa2118e86c4/writeups"
          },
          "resources": {
            "href": "http://127.0.0.1:9000/projectUpdates/8334fdb3-59f5-473f-
                a782-2fa2118e86c4/resources"
          },
          "milestones": {
            "href": "http://127.0.0.1:9000/projectUpdates/8334fdb3-59f5-473f-
                a782-2fa2118e86c4/milestones"
          },
          "todos": {
            "href": "http://127.0.0.1:9000/projectUpdates/8334fdb3-59f5-473f-
                a782-2fa2118e86c4/todos"
          },
          "communicationPlans": {
            "href": "http://127.0.0.1:9000/projectUpdates/8334fdb3-59f5-473f-
                a782-2fa2118e86c4/communicationPlans"
          },
          "risks": {
            "href": "http://127.0.0.1:9000/projectUpdates/8334fdb3-59f5-473f-
                a782-2fa2118e86c4/risks"
          }
        }
      }
    ]
  },
  "_links": {
    "self": {
      "href": "http://127.0.0.1:9000/projects/c561b4d4-f221-4852-9419-6
          bb2d09e3078/archivedUpdates"
    }
  }
}
```