Conception et mise en œuvre d'une gestion de projet centralisée pour PME

MASTER THESIS

BORISLAVA KINAREVA-DUMONT

Novembre 2011

Thesis supervisors :

Prof. Dr. Jacques PASQUIER–ROCHA
Software Engineering Group

UNIVERSITÉ DE FRIBOURG SUISSE
UNIVERSITÄT FREIBURG SCHWEIZ

Software Engineering Group
Department of Informatics
University of Fribourg
(Switzerland)

"There are no good project managers – only lucky ones. And the more you plan the luckier you get!"

Résumé

Les objectifs d'une entreprise de services sont d'une part la satisfaction de ses clients à travers des services répondant à leurs attentes et réalisés dans les budgets et les délais impartis, et d'autre part l'optimisation des ressources afin que l'entreprise puisse réaliser ses travaux de manière rentable.

Dans une petite entreprise, de par le fait que les ressources sont faibles et les effectifs limités, les projets ne peuvent être gérés par une personne dédiée à ce rôle. Cela implique un risque élevé de dérive s'il n'y a pas de système de gestion bien rôdé qui facilite la communication entre des différents acteurs.

C'est dans ce cadre que ce travail de master a été rédigé. Dans un premier temps, un survol des différentes théories de gestion de projet a été réalisé. Il a permis de dégager les tendances actuelles de gestion et d'accompagner la direction dans la mise en place d'un processus de gestion simple mais efficace.

Toujours dans ce même objectif de simplicité, un recueil des exigences a été élaboré et modélisé à l'aide des diagrammes UML « entités relations ».

La version 1.0 du logiciel de gestion « Synaps » a été développée dans cet objectif. Cette application est actuellement utilisée dans l'entreprise BlueSystem.

Remerciements

J'adresse mes remerciements à toutes les personnes qui m'ont aidé et soutenu dans mon travail, en particulier au Prof. Pasquier-Rocha qui a supervisé ce travail de Master du début à la fin et donné de précieux conseils.

Un grand merci également à la société BlueSystem qui a participé à l'élaboration du logiciel, aux membres de sa direction qui ont été présents dans tous les brainstormings et réunions de projet.

Table des matières

Liste des figures

Liste des tables

1
Introduction

1.1 Motivation

Dans le cadre de mes expériences professionnelles en parallèle de mes études, j'ai eu l'opportunité de travailler dans différents secteurs d'entreprises et d'y exercer divers rôles. J'ai participé également dans la mise en place de plusieurs outils de gestion, que ce soit au niveau de la gestion de projet, des finances, du service après vente ou de la relation clientèle.

Ces étapes de ma vie professionnelle m'ont permis de repérer certains problèmes récurrents dans la gestion d'une entreprise, tout particulièrement dans la gestion d'une petite PME active dans le service.

1.2 Cadre du travail

Dans le cadre de mon travail de Master en « Informatique de gestion », mon but est de développer un outil de gestion d'entreprise permettant, d'une manière centralisée, d'aider les petites sociétés de service à améliorer leur productivité, leur suivi clientèle et leur planification. Ce travail se fera en collaboration avec la société BlueSystem, qui désire mettre en place une plateforme de gestion de projet au sein de son entreprise.

1.3 Objectifs

- Analyse des différentes problématiques de la gestion de projet.
- Définition du concept général.
- Développement de la version 1.0 du logiciel.
- Identification des évolutions futures possibles.

2

Notion de gestion de projet

2.1 Les concepts fondamentaux

2.1.1 Qu'est-ce qu'un projet

Un projet est un ensemble d'étapes et d'activités coordonnées ayant pour objectif de répondre à un besoin exprimé par un client dans un délai imparti et un coût estimé au préalable [Web1]. Le projet à :

- Un caractère unique, car le résultat final est propre au projet entrepris.
- Un caractère temporaire car il se termine à un moment déterminé.

Afin d'en assurer son bon déroulement, le projet est régulé par un plan. Ce plan guidera sa progression au travers de contraintes en le limitant à des objectifs et des paramètres déterminés.

Le projet est un objectif extraordinaire (au sens littéral du mot) qui combine cinq aspects :

- Fonctionnel (Répondre à un besoin)
- Technique (Respect des spécifications)
- Délai (Respect des échéances)

- Organisationnel (Respect d'un mode de fonctionnement)
- Coût (respect du budget) [Web2]

Généralement, la réalisation d'un projet nécessite la mise en place d'une équipe de collaborateurs qualifiés dirigée par un chef de projet.

2.1.2 Qu'est ce que la gestion de projet

La gestion de projet est l'utilisation de techniques et d'outils dans le but de satisfaire les exigences et les attentes des différentes parties prenantes.

Les différentes étapes de la gestion de projet sont les suivantes :

- **L'organisation**
  Organisation structurelle, des flux d'informations, des acteurs et des supports de communications.
- **La planification**
  Estimation des coûts et des délais
- **La coordination**
  Entre les différents acteurs du projet, responsables, exécutants, …
- **Le pilotage**
  Organisation du déroulement du projet, découpage en activités. Suivi du déroulement, gestion des ressources.
- **La surveillance**
  Contrôle des coûts, des délais et de la qualité.

2.1.3 Le chef de projet

Le chef de projet est responsable du bon déroulement du projet jusqu'à sa clôture. C'est un défi constant qui demande une vison globale du contexte du projet et la capacité de concilier des exigences contradictoires comme :

- Les ressources disponibles et les attentes.
- Les priorités divergentes des différents acteurs.
- La qualité et la quantité.

Tout au long du projet, il sera constamment tiraillé entre le respect des coûts, des délais et de la qualité exigée.

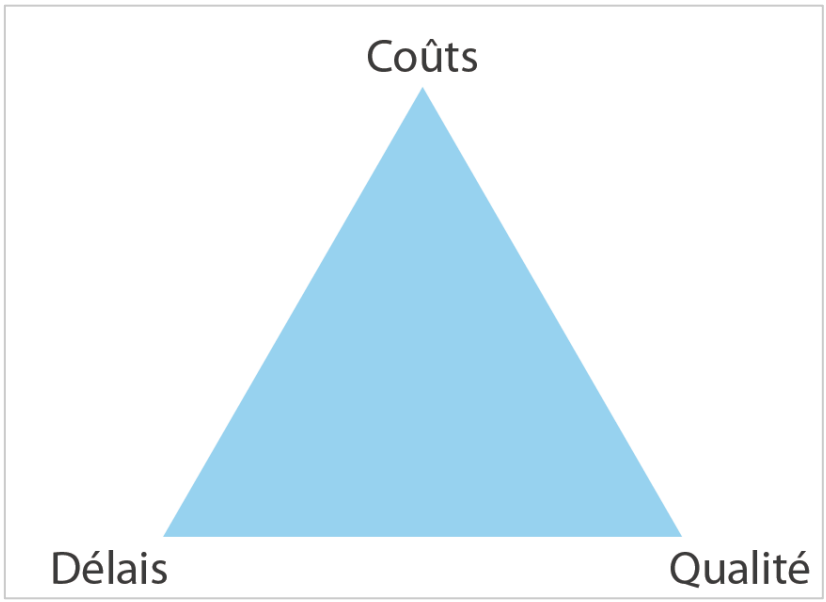

Figure 1 : Triangle magique de la gestion de projet

Le lecteur intéressé peut trouver plus d'informations sur le triangle magique de la gestion de projet sur le « Le blog d'un geek devenu directeur technique » [Web3]

2.2 La planification

Le projet peut être décomposé en phases, afin d'obtenir des sous-ensembles dont la complexité est plus facilement maîtrisable. Le découpage est essentiel à son aboutissement et à sa réussite.

L'approche par **jalons** permet de bien structurer le projet dans le temps. Les jalons permettent de faire le point sur le projet et de n'engager la phase suivante que si tout va bien. Le jalonnement préoccupe l'appréciation des résultats intermédiaires où le client est amené à se prononcer.

Le découpage d'un projet en sous-ensembles maîtrisables est essentiel à la conduite du projet et donc à son bon aboutissement et à sa réussite. Il permet également de procéder plus facilement à sa planification.

La planification a comme objectifs:

- De définir les travaux à réaliser
- De fixer les objectifs
- De coordonner les actions
- De suivre les actions en cours

- De maitriser l'avancement du projet.

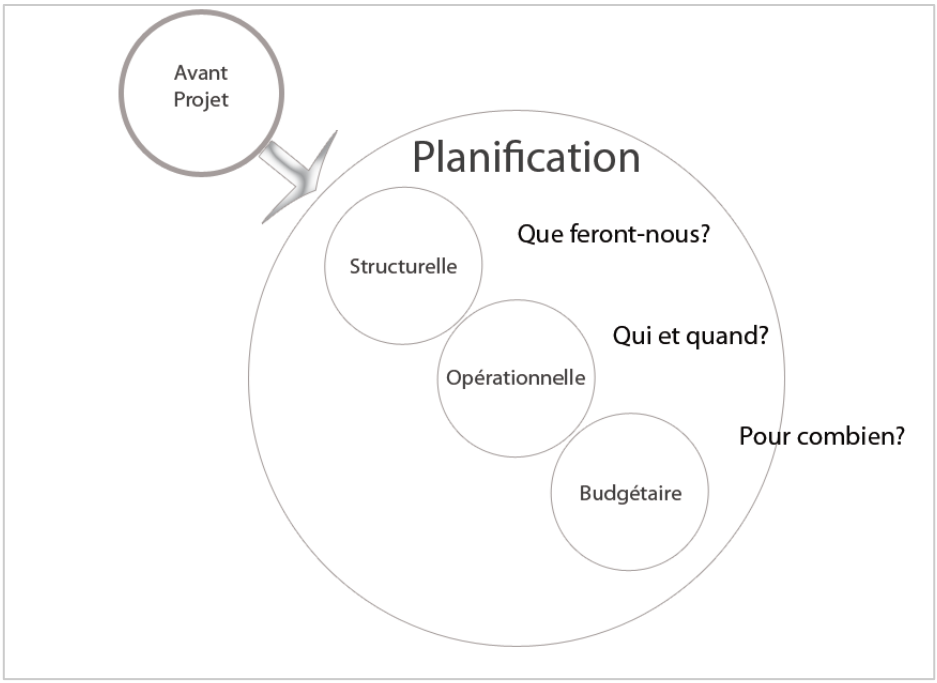

Figure 2 : Les différents types de planification

Le lecteur intéressé est renvoyé dans le cours INF3300 Planification structurelle [Des04].

2.2.1 La planification structurelle

Le rôle de la planification structurelle est d'identifier les travaux à compléter et de les décomposer en une liste d'activités à accomplir [Des04]. La constitution de cette base de données est essentielle aux étapes suivantes. L'équipe de projet doit avoir suffisamment confiance dans le caractère exhaustif de la liste des activités pour être assuré qu'une fois complétée, le projet réalisé sera conforme aux exigences initiales.

Cette planification peut être représentée à l'aide d'un organigramme de tâches « Work Breakdown Structure » (WBS) [Web4]. Le WBS permet de faciliter la planification du travail à effectuer, d'estimer la durée totale du projet et de déterminer les ressources nécessaires pour chaque étape. Ces différents objectifs sont atteints en divisant le travail à effectuer pour le projet dans des segments de plus en plus détaillés. La première étape consiste à identifier les principaux segments du travail à effectuer. Ensuite, il s'agit de diviser ces segments en des segments encore plus petits.

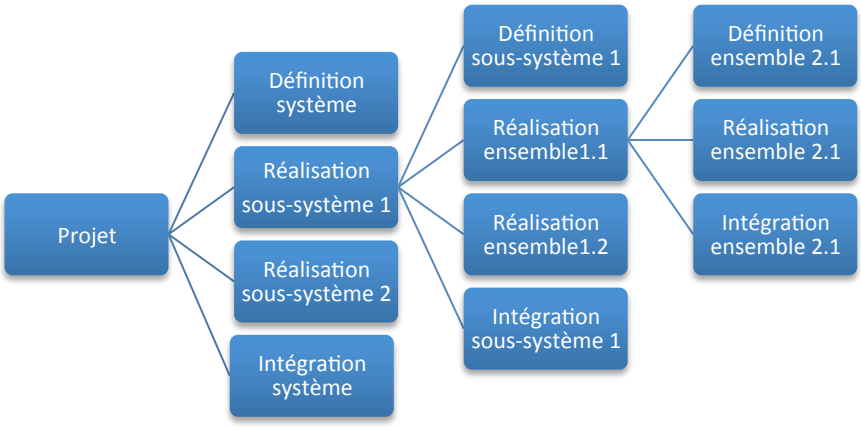

Figure 3 : Work Breakdown Structure

2.2.2 La planification opérationnelle

La planification opérationnelle a comme objectif de rendre le projet pilotable. Le plan est l'élément de base de cette planification. Il aide à finir dans les temps en respectant les coûts et en délivrant le produit attendu.

La planification opérationnelle place le plan au centre de la conduite du projet. Il rassemble les informations et permet d'en piloter les différentes dimensions ; les délais, les coûts, les ressources, la communication et les risques.

Les objectifs de la planification opérationnelle sont les suivants :

- La création d'un réseau ordonnancé d'activités à partir de la liste des activités identifiées dans la planification structurelle.
- L'estimation de la durée des activités et des ressources nécessaires.
- L'identification des dépendances entre les différentes activités.
- L'identification du chemin critique.
- L'assignation des activités aux différents participants au projet.

Deux outils sont principalement utilisés dans la planification organisationnelle. Il s'agit de la méthode du chemin critique et du diagramme de Gantt.

« Le **chemin critique** correspond à la séquence d'activités qui détermine la durée totale du projet. Ce chemin est continu depuis le début jusqu'à la fin du projet. Tout retard affectant une tâche du chemin critique est répercuté sur la durée du projet et donc sa date de fin. La tâche critique est une tâche du chemin critique. Toute modification sur la durée d'une de ces activités critiques impacte d'autant plus la durée totale du projet. » [Web5]

Le chemin critique est représenté à l'aide de la méthode PERT. PERT a été créé en 1956 à la demande de la marine américaine, qui veut planifier la durée de son programme de missiles balistiques nucléaires miniaturisés Polaris. L'enjeu principal est de rattraper le retard en matière de balistique par rapport à l'URSS, après le choc de la « crise de Spoutnik ». L'étude est réalisée par la société de conseil en stratégie Booz Allen Hamilton. Alors que le délai initial de ce programme – qui a fait intervenir 9000 sous-traitants et 250 fournisseurs – était de 7 ans, l'application de la technique du PERT a permis de réduire ce délai à 4 ans. [Web6]

La méthode PERT est une technique permettant de gérer l'ordonnancement des tâches dans un projet. Elle consiste à représenter sous forme de graphe, un réseau de tâches dont l'enchaînement permet d'aboutir à la réalisation d'un projet.

Un graphe de dépendances est utilisé. Pour chaque tâche, une date de début et de fin sont indiquées. Le diagramme permet de déterminer le chemin critique qui conditionne la durée minimale du projet.

Par exemple, la préparation d'un gâteau aux pommes peut être décomposée de la manière suivantes :

- A : acheter les pommes, la farine, le lait, le beurre, la levure (25 min)
- B : éplucher et couper les pommes en fines tranches (5 min)
- C : mélanger ensemble farine, lait, beurre, levure pour faire la pâte (5 min)
- D : demander à sa petite fille de préchauffer le four (2 min)
- E : étaler la pâte et la poser dans un moule à gâteau (5 min)
- F : ranger les tranches de pommes sur la pâte et placer le moule au four (5 min)
- G : attendre la fin de la cuisson pour sortir le plat du four et démouler le gâteau (5 min) .

La première étape consiste donc à faire un tableau qui indique, pour chaque tâche, la ou les tâches qui doivent absolument être terminées pour qu'elle puisse commencer. Par exemple ici:

Table 1 : Tableau préparatoite PERT

Tâche ou événement	Tâche précédente
B	**A**
C	**A**
E	**C, D**
F	**B, C, D, E**
G	**F**

On peut représenter les relations entre les tâches et les étapes par le diagramme suivant :

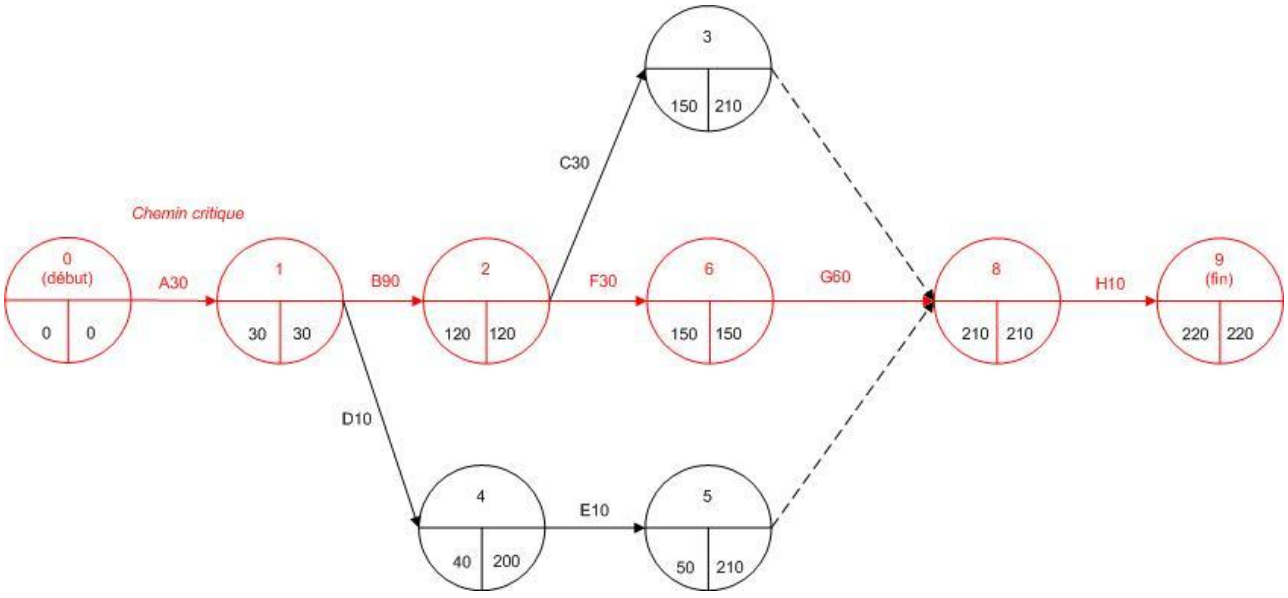

Figure 4 : Le chemin critique [Web7]

Pour déterminer la date au plus tôt d'une tâche, il faut parcourir le diagramme de gauche à droite et calculer le temps du plus long des chemins menant du début du projet à cette tâche. S'il y a plusieurs sous-chemins, on effectue le même calcul pour chacun et on choisit la date la plus grande. [Web7]

Pour déterminer la date au plus tard d'une tâche, il faut parcourir le diagramme de droite à gauche, et soustraire de la date au plus tard de la tâche suivante la durée de la tâche dont on calcule la date au plus tard. S'il y a plusieurs sous-chemins, on effectue le même calcul pour chacun et on choisit la date la plus petite. [Web7]

La différence entre la date au plus tard et la date au plus tôt d'une tâche s'appelle la marge totale. [Web7]

On dit qu'une tâche de A vers B est critique si la différence entre la date au plus tard de B et la date au plus tôt de A est égale à la durée de la tâche à accomplir. L'ensemble des tâches

critiques constitue le chemin critique, c'est-à-dire le chemin sur lequel aucune tâche ne doit avoir de retard pour ne pas retarder l'ensemble du projet (ici en rouge). [Web7]

Le **diagramme de Gantt** permet de visualiser dans le temps les diverses activités liées composant un projet. Il permet de représenter graphiquement l'avancement du projet. Le concept a été développé par Henry L. Gantt, ingénieur américain, vers 1910. [Web7]

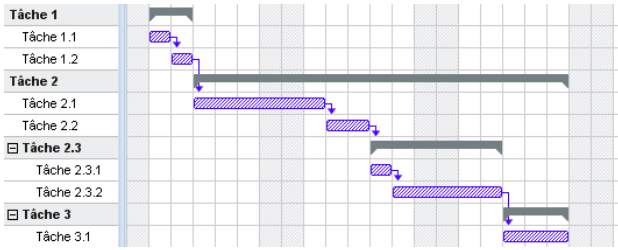

Figure 5 : Exemple de diagramme de Gantt

La durée d'exécution d'une tâche est matérialisée par une barre horizontale. Il est également fréquent de matérialiser par des flèches, les liens de dépendance entre les activités (la flèche relie la tâche précédente à la tâche suivante). Il est souvent complété en ligne par la liste des ressources affectées à chacune des activités ainsi que par divers indicateurs, fonction de la charge ou du délai, permettant d'en suivre l'avancement.

Ce diagramme permet :

- de déterminer les dates de réalisation d'un projet,
- d'identifier les marges existantes sur certaines activités,
- de visualiser d'un seul coup d'œil le retard ou l'avancement des travaux.

2.2.3 La planification budgétaire

La planification budgétaire se développe parallèlement à l'élaboration du calendrier du projet. [Web7] Son but est de déterminer les coûts liés aux activités que l'on définit, afin de comparer les coûts réels et les évaluer en fonction du budget.

Le budget est souvent un ensemble de paramètres équilibrés qui entrent dans la réalisation du projet. Des écarts sur les délais prévus affectent le coût. Quand les coûts du projet ont dépassé les prévisions, le chef de projet devrait revoir le plan afin de déterminer si le contenu, le budget ou le calendrier nécessitent un ajustement.

Elaborer les coûts d'un projet revient à additionner les coûts de chacune de ses activités. Si les activités ont été suffisamment décomposées et énoncées clairement, estimer les coûts devient un exercice relativement simple.

Pour une société de services, le coût d'une tâche est composé de main d'œuvre, de sous-traitance et des frais divers.

3

Le cas de l'entreprise BlueSystem

3.1 A propos de la société

BlueSystem est une société fribourgeoise spécialisée dans le développement de solutions basées sur les technologies Web. Ses principaux domaines d'activités sont les suivants :

- Création de sites Internet
- Développement de plateformes E-commerce
- Intégration d'Intranet Entreprise
- Développement d'applications

La société Bluesystem est active sur le marché depuis 2004. Elle compte aujourd'hui 7 collaborateurs et gère annuellement une vingtaine de projets.

3.1.1 Structure de la société

La structure de BlueSystem est composée de trois fonctions distinctes. L'*Administration* et le *Marketing* sont chacune assimilées à une personne. La *production* (développement informatique), quant à elle, est composée de 5 personnes organisées autour de projets.

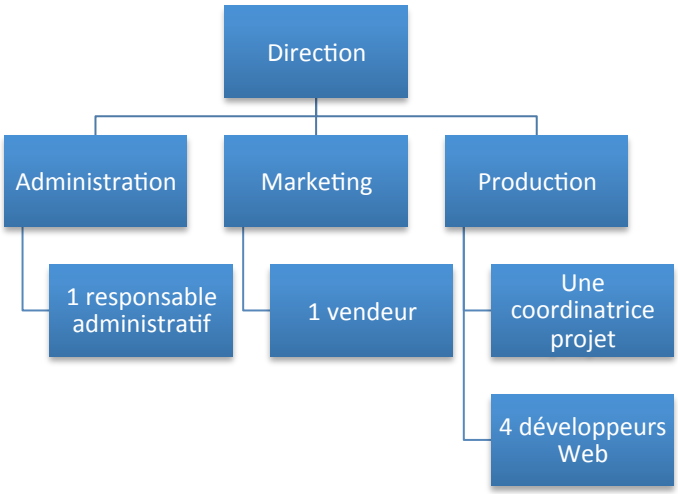

Figure 6 : Organigramme BlueSystem

3.1.2 Gestion par projet

Afin de simplifier la gestion et la délégation des différentes activités de l'entreprise, BlueSystem oriente depuis le début de son existence ses activités par projet. Elle différencie ses projets en trois catégories.

Les **projets externes** représentent les mandats réalisés pour ses clients. Ils sont touts gérés par le département *Production* et la coordinatrice de projet en assure le suivi.

Les projets **administratifs** regroupent différentes activités du secteur *administratif* et *marketing*. Ils sont gérés annuellement, mais le principal objectif est de mesurer le coût interne par groupe d'activités.

Les **projets internes** représentent les différents développements internes. Ces projets sont généralement de plus grande envergure et peuvent être planifiés sur plusieurs années.

3.2 Outils de gestion actuels

BlueSystem travaille actuellement avec différents outils informatiques, mais les informations ne sont pas centralisées dans un système de gestion globale. Le tableau ci-dessous donne un aperçu des fonctions de l'entreprise ainsi que les outils utilisés.

Table 2 : Outils informatiques actuellement en place

Fonctions	Outils utilisés
Gestion des clients	Feuille de calcul partagée en ligne
Gestion des contacts	Application Google Apps Professionnel (cf. [Web8])
Facturation	Base de données interne
Planification et suivi des activités	Gestionnaire de tâche en ligne toodledo (cf. [Web9])
Suivi des liquidités	Feuille Excel (cf. [Web10])

Même si ces différents outils apportent une solution basique, ils fonctionnent correctement et offrent chacun une solution acceptable. C'est principalement le fait qu'ils travaillent chacun de manière indépendante, sans offrir de possibilité de partage, qui les rend de plus en plus lourd à maintenir. Les activités de la société s'étant passablement développées ces deux dernières années, les flux d'informations sont devenus trop importants pour maintenir une gestion rigoureuse sans augmenter considérablement le temps passé à coordonner les activités.

Afin de garantir une gestion optimale, BlueSystem désire mettre en place un nouveau système de gestion centralisé. Elle désire profiter de cette nouvelle plateforme pour améliorer différents points et développer de nouveaux outils de gestion. La mise en place des éléments de la plateforme doit pouvoir se faire progressivement. Voici la liste des améliorations désirées :

- Centralisation de la base de données clients et contacts.
- Meilleure classification des contacts.
- Simplification de la planification pour le coordinateur projet.
- Meilleure visibilité pour les collaborateurs sur les activités en cours.
- Classification par projet de toutes les heures effectuées.
- Calcul des heures dues, heures supplémentaires et vacances.

3.3 Définitions et abréviations du vocabulaire commun

Le vocabulaire facilite la communication et évite les malentendus entre les utilisateurs et les réalisateurs du projet. Voici ci-dessous quelques termes essentiels régulièrement utilisés tout au long du projet :

- **Projet** : un mandat réalisé pour un client.
- **Phase** : une phase dans le projet informatique.

- **Activité** : composante d'un projet, généralement une tâche.
- **Heures effectuées** : les heures qu'un employé consacre pour la réalisation d'une activité.
- **Employé** : toute personne qui travaille pour la société Bluesystem.
- **Contact** : toute personne qui fait partie du carnet d'adresses de la société Bluesystem.
- **Compte** : une relation d'affaire de l'entreprise, généralement le client.

3.4 Objectifs de la nouvelle plateforme

Une série d'entretiens a permis de décomposer les besoins de la nouvelle plateforme. Le schéma heuristique présenté sur la page suivante résume l'expression de ces besoins. Ils ont été classés par degré d'urgence. Le développement sera planifié dans le temps en fonction des ces degrés.

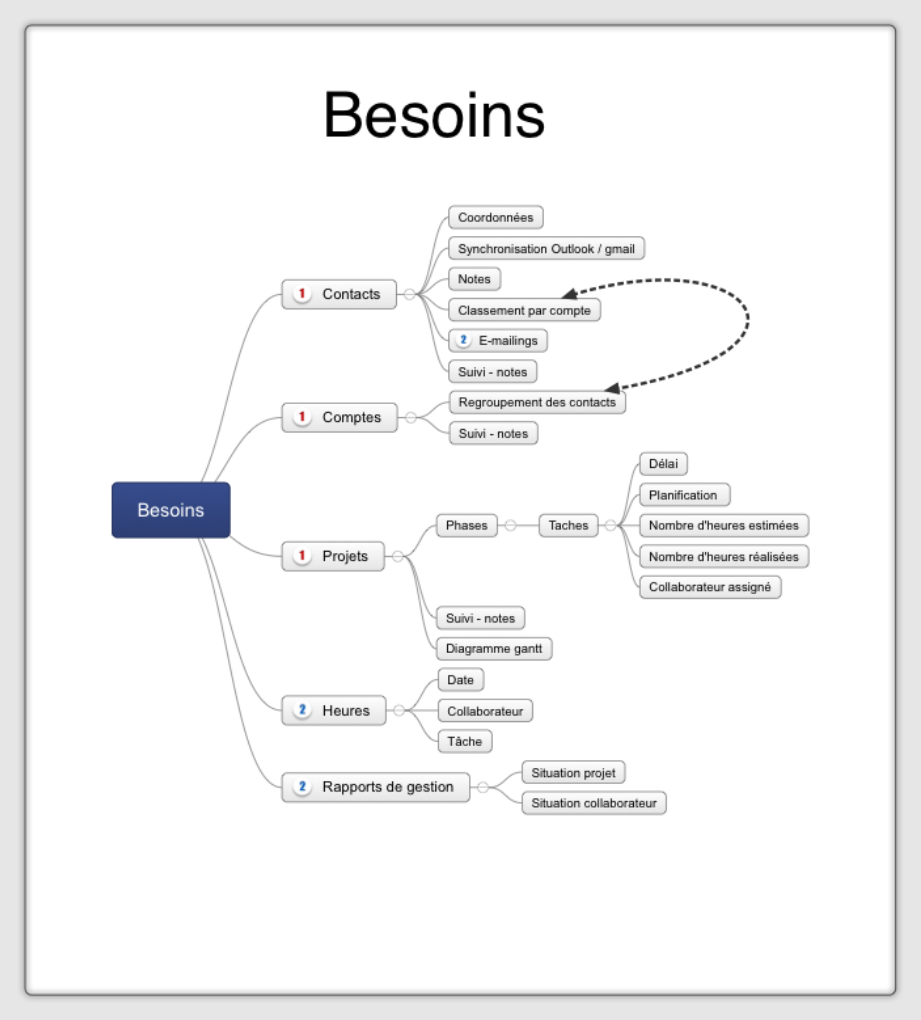

Figure 7 : Schéma heuristique des besoins

3.4.1 Objectifs fonctionnels

Contacts

- Chaque utilisateur doit pouvoir rechercher des contacts et visualiser ses informations détaillées.
- Chaque utilisateur doit pouvoir entrer des notes par rapport aux différentes relations avec le contact.
- Les contacts peuvent être classés dans des comptes.
- Les contacts peuvent être « taggés ».
- L'administrateur peut supprimer des contacts.
- Chaque utilisateur peut ajouter des contacts et les mettre à jour.
- Les contacts peuvent être exportés et synchronisés avec un compte exchange ou un compte Gmail.

Gestion des comptes

- Les comptes représentent les relations d'affaires avec l'entreprise. Ils sont généralement les clients, les partenaires ou les fournisseurs.
- Les chefs de projets et les administrateurs peuvent créer de nouveaux comptes.
- Tous les utilisateurs peuvent consulter les comptes, les contacts du compte et les notes relatives (compte et contacts du compte)

Planification des projets

- Le chef de projet (CP) peut créer et éditer des projets.
- Le CP peut créer l'équipe de projet.
- Le CP peut créer les différentes phases du projet.
- Le CP peut créer les différentes activités du projet, les planifier et les attribuer aux collaborateurs.
- Les collaborateurs peuvent à tout moment voir l'état du projet grâce à un diagramme GANTT.
- Chaque collaborateur peut créer des activités (tâches). Il peut les attribuer à un projet, pour autant que les paramètres du projet le lui permettent.
- Chaque collaborateur dispose dans son tableau de bord d'une vision globale de ses activités ainsi que leur état.
- L'administrateur et le chef de projet ont accès à un rapport de projet avec l'état de l'avancement et la situation budgétaire.

Gestion des heures

- Chaque employé doit entrer toutes ses heures quotidiennement.
- Chaque heure entrée correspond à une activité de projet.
- L'entrée des heures se fait le jour même ou le jour suivant.
- La saisie des vacances, maladies et autres congés se fait dans le système.
- Le collaborateur peut à tout moment accéder à son solde d'heures.

3.4.2 Objectifs non fonctionnels

- Plateforme multi-utilisateurs accessible depuis n'importe quel poste connecté à Internet
- Interface riche intuitive et rapide d'utilisation

4

Exigences fonctionnelles

4.1 Démarche pour décrire les besoins des utilisateurs

Ce chapitre décrit la majeure partie des besoins de l'utilisateur. Le processus unifié sera utilisé à cet effet [Web11]. C'est un processus de développement logiciel itératif et incrémental, centré sur l'architecture, conduit par des cas d'utilisation et piloté par des risques.

Un cas d'utilisation (use case) montre les interactions fonctionnelles entre les acteurs et le système. Il est utilisé dans la spécification des besoins des utilisateurs.

Selon Alistair Cockburn [Coc09] :

« Un cas d'utilisation établit, entre les différents intervenants, un contrat régissant le comportement d'un système. Il décrit ce comportement sous diverses conditions, lorsque le système répond à une requête émanant de l'un des intervenants, appelé acteur principal. L'acteur principal amorce une interaction avec le système en vue d'atteindre un objectif particulier. Le système répond, en veillant à protéger les intérêts de tous les intervenants. Diverses séquences de comportement, ou scénarios, peuvent se déployer, en fonction des requêtes effectuées et les conditions de leur réalisation. Le cas d'utilisation regroupe ces différents scénarios. »

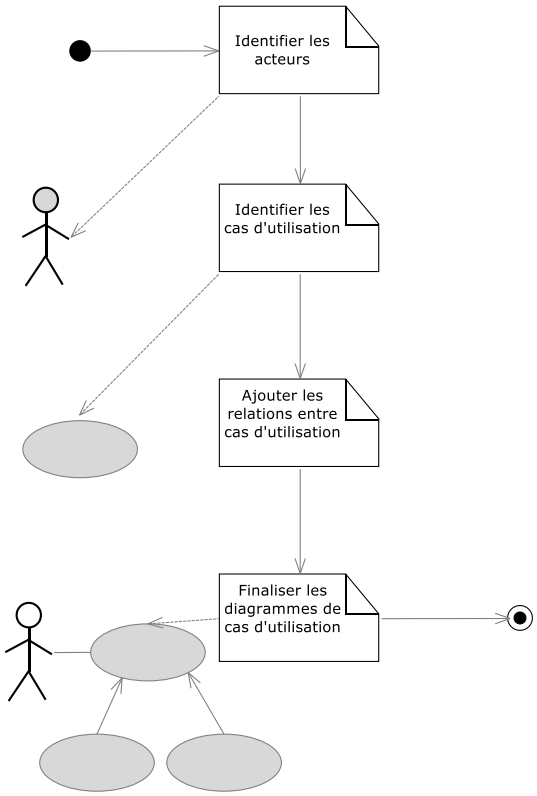

Figure 8 : Identification des Uses Cases

Une description textuelle sera faite pour décrire chaque cas d'utilisation. Cette description n'est pas normalisée par un diagramme UML. Pour la faire, nous nous sommes inspirés par la définition faite dans « Comment rédiger des cas d'utilisation efficaces » d'Alistair Cockburn [Coc09].

Dans sa forme textuelle un cas d'utilisation est une collection de scénarios de succès et d'échecs qui décrivent la façon dont un acteur particulier utilise le système pour atteindre un objectif.

Les termes suivants sont utilisés pour chaque cas d'utilisation:

- **Pré-condition**: définit ce qui doit être vrai en amont afin que le processus puisse démarrer.
- **Déclencheur:** action qui déclenche le scénario.
- **Scénario nominal**: scénario qui satisfait l'objectif des acteurs par le chemin le plus direct.
- **Extensions**: tous les autres scénarios ou branchements possibles aussi bien de succès que d'échec.

4.2 Identification des acteurs

D'après Alistair Cockburn [Coc09] :

« Un acteur représente un rôle joué par une entité externe qui interagit directement avec le système étudié. Il peut consulter et/ou modifier directement l'état du système, en émettant et/ou recevant des messages susceptibles d'être porteurs de données. »

Les acteurs suivants ont été identifiés :

- Employé de la société
- Chef de projet: employé responsable d'un ou plusieurs projets.
- Administrateur : employé qui s'occupe de l'administration de l'entreprise.

Il existe des relations d'héritage entre les différents acteurs. Les employés sont étendus en Chef de projet et en Administrateur, qui ont des privilèges supplémentaires comme on peut le voir sur le diagramme « Collection des cas d'utilisation ».

4.3 Identification des cas d'utilisation

Des cas d'utilisation représentent un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable intéressant pour un acteur principal.

Dans la collection des cas d'utilisation sont énumérés tous les processus principaux du système :

- Authentification
- Administration des employés
- Administration des contacts
- Administration des comptes
- Administration des projets
- Gestion des tâches
- Gestion des heures
- Suivi de projet
- Classement et planification des tâches
- Annotation

Chacun d'eux fera objet d'une analyse plus approfondie dans ce chapitre.

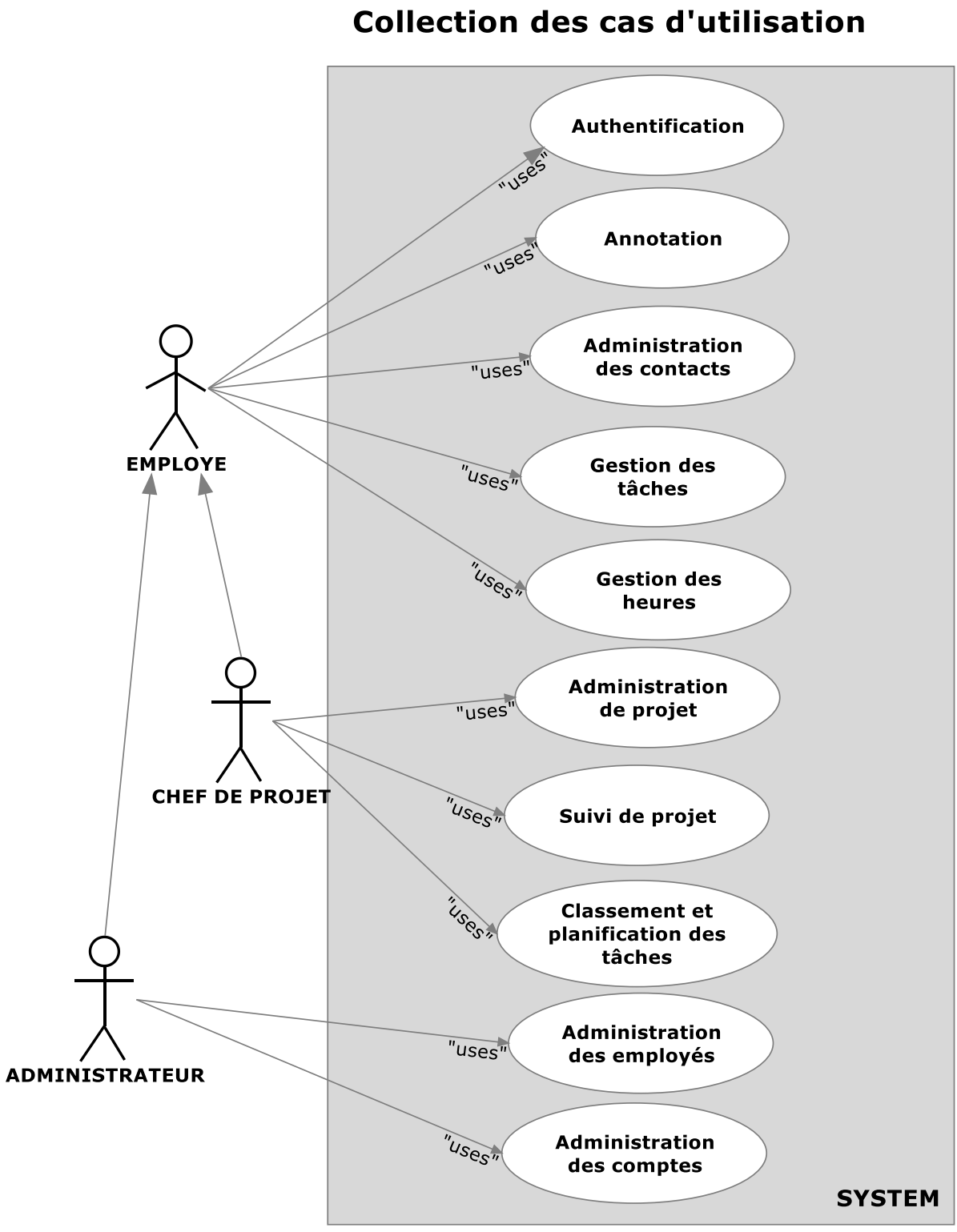

Figure 9 : Collecte des cas d'utilisation

4.4 Authentification

L'authentification est la condition préalable nécessaire à tous les autres processus décrits dans les cas d'utilisation. Elle doit être utilisée pour permettre aux acteurs d'exécuter leurs propres cas d'utilisation majeurs. Ce cas d'utilisation ne représente pas un objectif à part entière, mais un objectif de niveau intermédiaire. Elle est décrite ci-dessus (Identification des Uses Cases, figure 8) et n'est plus reprise dans les autres cas d'utilisation.

	Description
Acteur principal	Utilisateur
Objectif	L'utilisateur effectue son login dans le système.
Pré-conditions	Les données personnelles permettant l'authentification existent dans le système.
Déclencheur	L'utilisateur ouvre la page de login du système.
Scénario nominal	Le système affiche un formulaire de login avec les champs nom d'utilisateur et mot de passe. : L'utilisateur remplit les valeurs du formulaire et sélectionne « soumettre » une fois terminé. Le système valide les données. Le système permet à l'utilisateur de rentrer dans le système.
Extensions	Le login échoue si : Le couple de valeurs nom d'utilisateur + mot de passe ne sont pas trouvés dans le système L'utilisateur n'est plus actif

Figure 10 : UC1 Authentification

4.5 Administration des employés

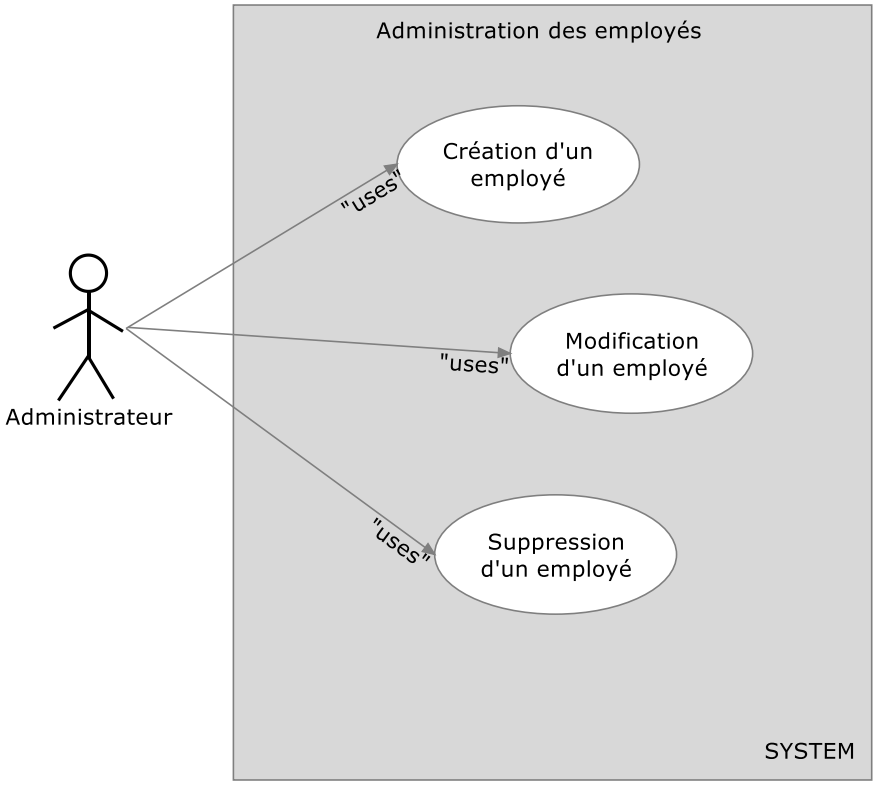

Figure 11 : Collecte des cas d'utilisation Administration des employés

4.5.1 Création d'un employé

	Description
Acteur principal	Administrateur
Objectif	L'administrateur doit pouvoir créer un utilisateur.
Pré-conditions	Doit être authentifié. Doit être administrateur.
Déclencheur	L'administrateur choisit « Création d'un employé »
Scénario nominal	Le système affiche un formulaire de création d'employé vide contenant les champs suivants : Nom d'utilisateur, mot de passé, nom, prénom, email, données synchronisation google, rôle (liste à choix avec employé, chef de projet, administrateur) L'administrateur remplit les valeurs du formulaire et sélectionne « soumettre » une fois terminé.

	Le système valide les données. Le système enregistre l'employé. Le système répond à l'administrateur en lui présentant le résultat.
Extensions	L'enregistrement échoue si : Les valeurs minimales ne sont pas remplies. Un employé avec ce nom existe déjà.

Table 3 : UC2, création d'un employé

4.5.2 Modification d'un employé

	Description
Acteur principal	Administrateur
Objectif	L'administrateur doit pouvoir modifier les données des employés.
Pré-conditions	Doit être authentifié. Doit être administrateur.
Déclencheur	L'administrateur choisit un employé de la liste des employés.
Scénario nominal	Le système affiche le formulaire de l'employé choisi contenant les champs suivants: Nom d'utilisateur, mot de passé, nom, prénom, rôle, données synchronisation google. L'administrateur change les valeurs du formulaire et sélectionne « soumettre » une fois terminé. Le système valide les données. Le système enregistre les modifications. Le système répond à l'administrateur en lui présentant le résultat.
Extensions	La modification échoue si: Les valeurs minimales ne sont pas remplies.

Table 4 : UC3, modification d'un employé

4.5.3 Suppression d'un employé

	Description
Acteur principal	Administrateur

Objectif	L'administrateur doit pouvoir supprimer les données des employés.
Pré-conditions	Doit être authentifié. Doit être administrateur.
Déclencheur	L'administrateur choisit un employé de la liste des employés .
Scénario nominal	Le système affiche le formulaire de l'employé choisi. L'administrateur choisit le bouton Supprimer. Le système demande confirmation. Le système valide la suppression et l'exécute. Le système répond à l'administrateur en lui présentant le résultat.
Extensions	La suppression échoue si: L'employé a fait des saisies des tâches ou des heures dans le système.

Table 5 : UC4, suppression d'un employé

4.6 Administration des contacts

L'administration des contacts est semblable à l'administration des employés. Pour cette raison elle est une décrite par la même collection de cas d'utilisation.

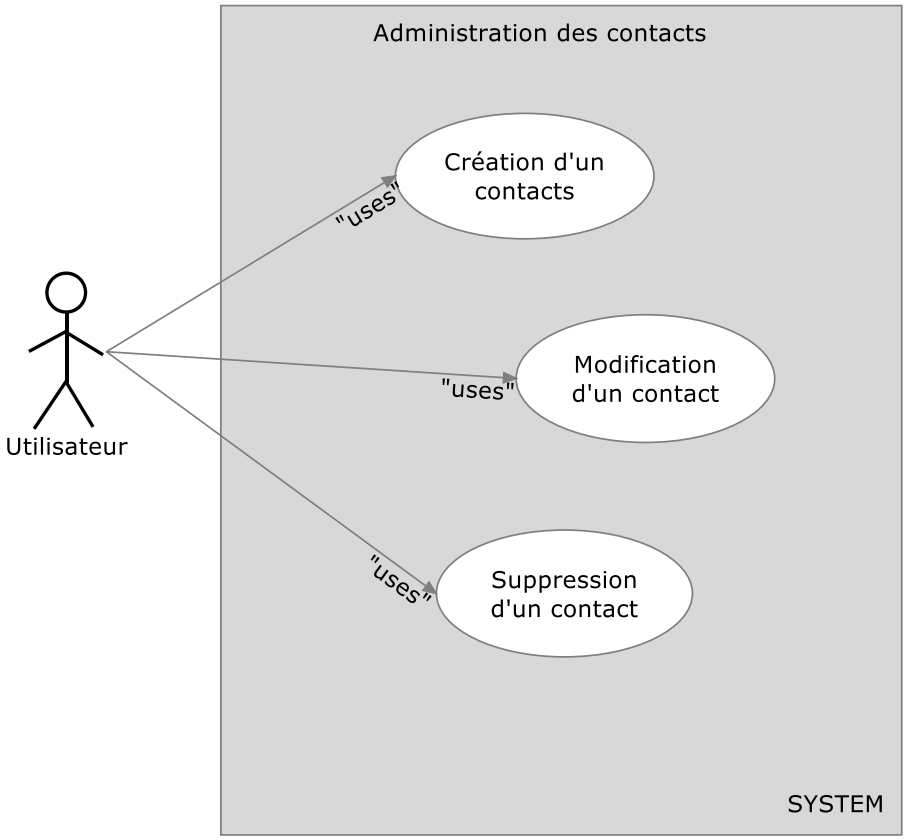

Figure 12 : Collecte des cas d'utilisation administration des contacts

4.7 Administration des comptes

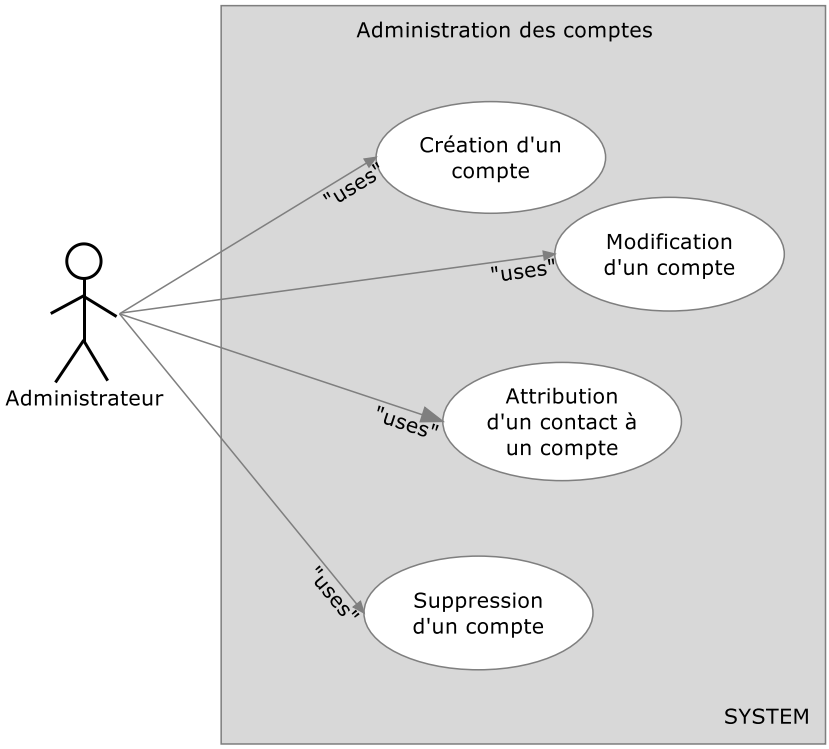

Figure 13 : Collecte des cas d'utilisation administration des comptes

Les trois cas d'utilisation suivants : création, modification et suppression d'un compte sont semblables à l'Administration des employés et ne sont pas développés. Le seul nouveau processus «Attribution d'un contacts à un compte » fait l'objet d'un cas d'utilisation décrit ci-dessus.

4.7.1 Attribution d'un contact à un compte

	Description
Acteur principal	Administrateur
Objectif	L'administrateur doit pouvoir rajouter un contact à un compte.
Pré-conditions	Doit être authentifié. Doit être administrateur.
Déclencheur	L'administrateur choisit la configuration des comptes.

Scénario nominal	Le système affiche la liste des comptes. L'administrateur choisit le compte qui l'intéresse. Le système affiche les données du compte qui contiennent la liste des contacts qui y sont engagés et qui n'y sont pas. L'utilisateur choisit un ou plusieurs de ces contacts et valide. Le système valide les données. Le système enregistre les modifications. Le système répond à l'administrateur en lui présentant le résultat.
Extensions	Le compte ou le contact n'existe pas préalablement et doit être créé.

Table 6 : UC5, attribution d'un contact à un comte

4.8 Gestion des tâches

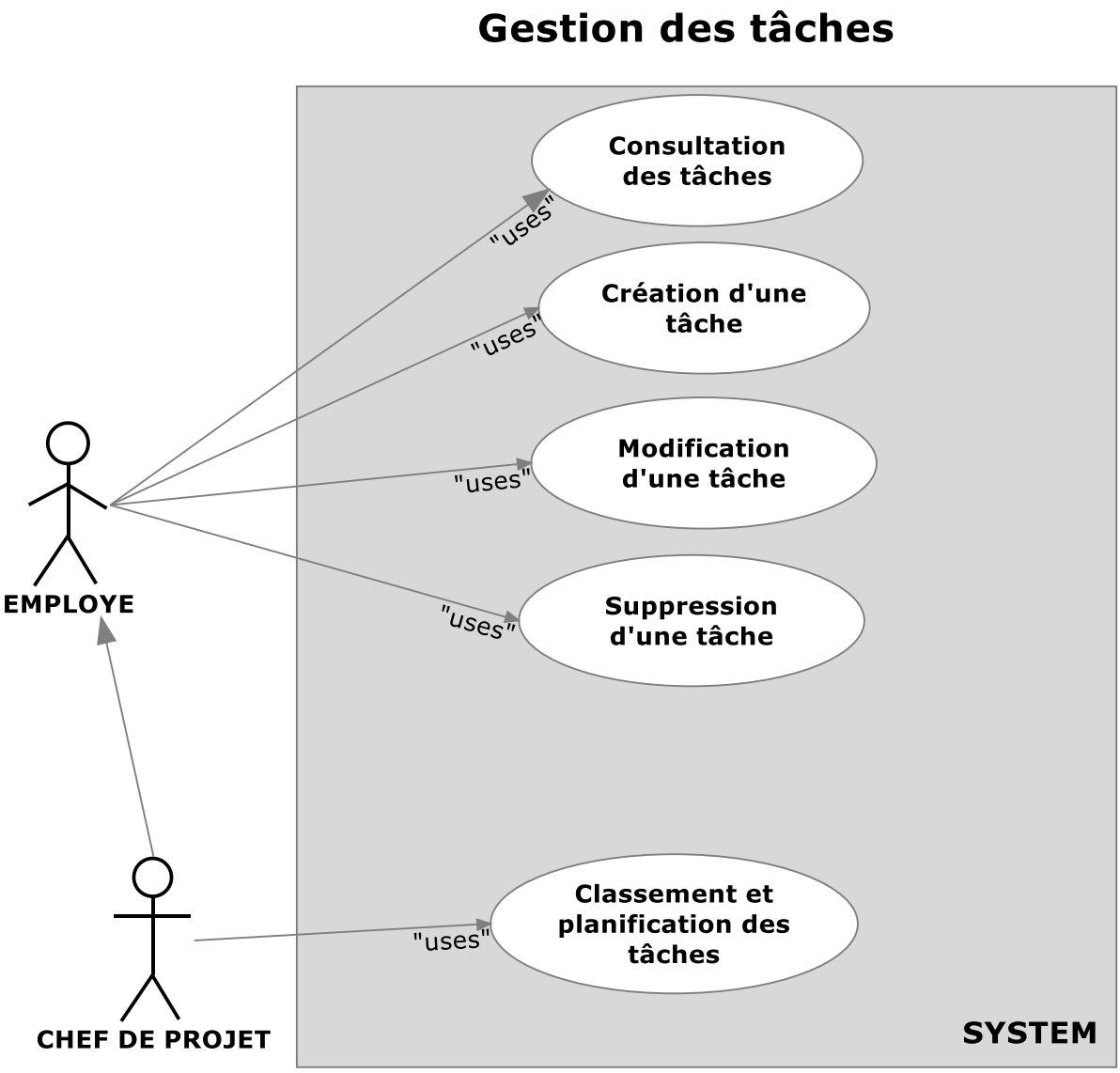

Figure 14 : Collecte des cas d'utilisation gestion des tâches

4.8.1 Consultation des tâches

	Description
Acteur principal	Employé, Chef de projet
Objectif	Les acteurs doivent pouvoir consulter et suivre l'état des tâches les concernant.
Pré-conditions	L'acteur doit être authentifié.

Déclencheur	L'acteur choisit « Liste des tâches »
Scénario nominal	Le système affiche la liste des tâches avec les filtres: Projet, utilisateur, état de l'activité (tous, ouverts, terminés) L'acteur choisit ses paramètres et lance la recherche. Le système génère une liste avec toutes les activités correspondant à la recherche.
Extensions	-

Table 7: UC6, consultation des tâches

4.8.2 Création d'une tâche

	Description
Acteur principal	Employé, Chef de projet
Objectif	Les acteurs doivent pouvoir créer une tâche
Pré-conditions	L'acteur doit être authentifié.
Déclencheur	L'acteur choisit « Nouvelle tâche »
Scénario nominal	Le système affiche un formulaire de nouvelle activité vide avec les champs suivants : sujet, détail, délai, planifié le, temps estimé, responsable L'acteur remplit le formulaire. Le système valide les données. Le système enregistre l'activité en ajoutant l'auteur de l'activité. Le système répond à l'acteur en lui présentant le résultat
Extension	La tâche ne peut pas être ajoutée si les valeurs minimum réqiuses ne sont pas remplies.

Table 8 : UC7, Création d'une tâche

4.8.3 Modification d'une tâche

	Description
Acteur principal	Employé, Chef de projet
Objectif	Les acteurs doivent pouvoir créer une activité
Pré-conditions	L'acteur doit être authentifié..
Déclencheur	L'acteur choisit « Modification d'une tâche» depuis « Consultation des tâches ».
Scénario nominal	Le système affiche le formulaire « Tâche » avec les champs suivants: sujet, détail, délai, planifié le, temps estimé, responsable L'acteur remplit le formulaire. Le système valide les données. Le système enregistre l'activité en ajoutant les heures dans le domaine de « Gestion des heures ». Le système répond à l'acteur en lui présentant le résultat
Extension	La tâche ne peut pas être modifiée si les valeurs minima requises ne sont pas remplies.

Table 9 : UC8, Modification d'une tâche

4.8.4 Affectation et planification des tâches

	Description
Acteur principal	Chef de projet
Objectif	Le Chef de projet doit pouvoir classer et planifier les tâches de ses projets et aussi celles qui ne sont pas encore attribuées.
Pré-conditions	L'acteur doit être authentifié. L'acteur doit avoir le rôle chef de projet.
Déclencheur	Le Chef de projet choisit « Consultation des tâche»
Scénario nominal	Le système affiche les paramètres de recherche suivants: Projet, utilisateur, état de l'activité (dépassée, du jour, de la semaine, toutes). Le Chef de projet choisit ses paramètres et lance la recherche. Le système génère une « view » avec toutes les activités correspondant à la recherche. Le Chef de projet choisit celle qui l'intéresse, la classe et la planifie en répondant aux questions suivantes : Qui, quand, combien de temps, phase ? ou contacts ?, ou

	compte ? Le système valide les données. Le système enregistre l'activité Le système répond au Chef de projet en lui présentant le résultat
Extensions	-

Table 10 : UC9, Affectation et planification d'une tâche

4.9 Administration de projet

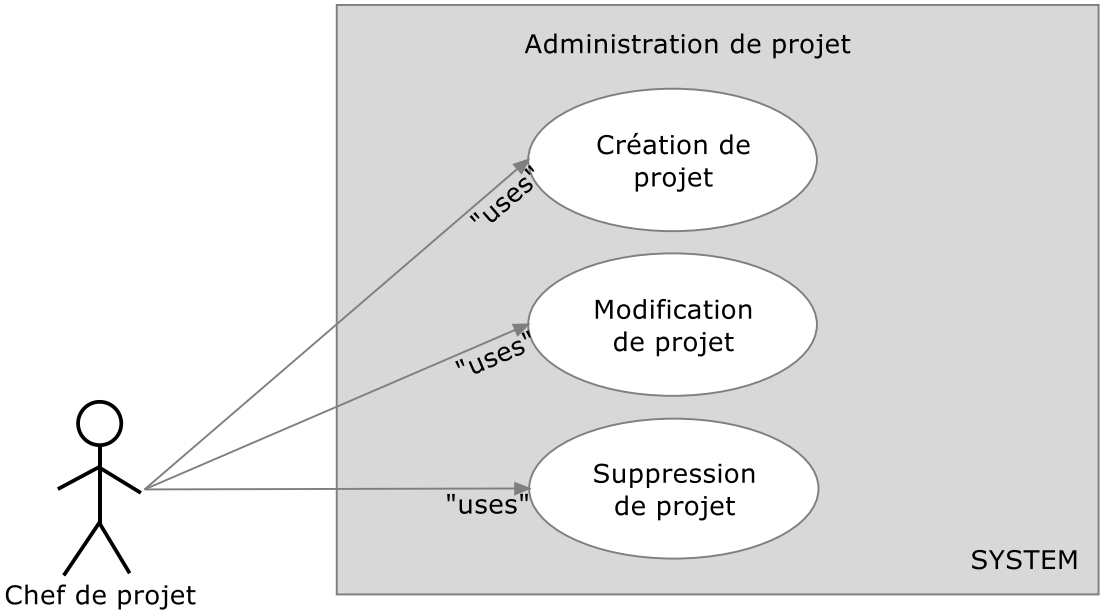

Figure 15 : Collecte de uses cases administration du projet

4.9.1 Création de projet

	Description
Acteur principal	Chef de projet
Objectif	Le chef de projet doit pouvoir créer un projet.
Pré-conditions	Doit être authentifié. Doit être chef de projet.
Déclencheur	L'administrateur choisit « Création de projet »
Scénario nominal	Le système affiche un formulaire de création de projet vide contenant les champs suivants :

	code, nom, date début, date fin, compte (liste à choix), chef de projet (liste à choix des employés) L'administrateur remplit les valeurs du formulaire et sélectionne « soumettre » une fois terminé. Le système valide les données. Le système enregistre le projet. Le système répond à l'administrateur en lui présentant le résultat.
Extensions	L'enregistrement échoue si : Les valeurs minimales ne sont pas remplies. Un projet avec ce nom ou code existe déjà.

Table 11 : UC10, Création d'un projet

4.10 Suivi de projet

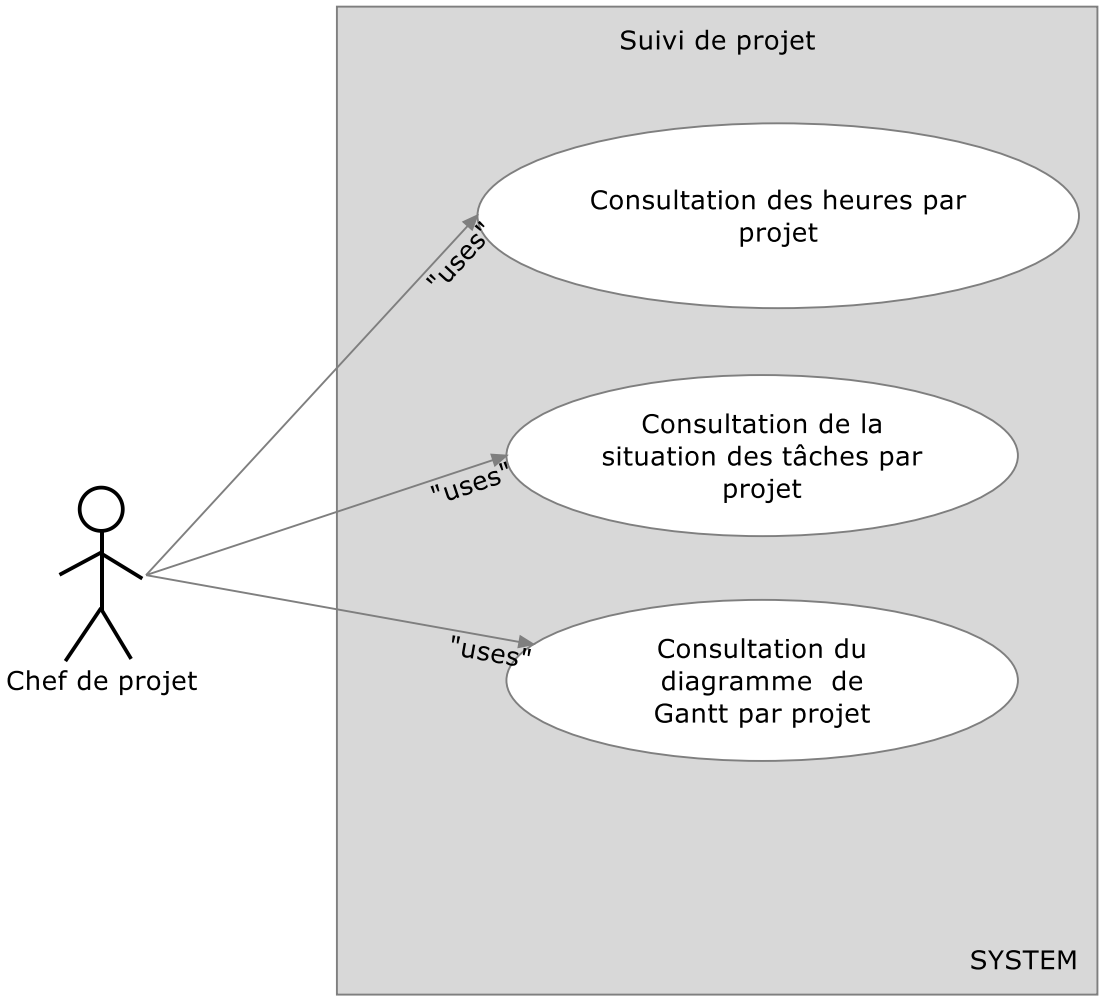

Figure 16 : Collecte de cas d'utilisations suivi de projet

4.10.1 Consultation des heures par projet/phase

	Description
Acteur principal	Chef de projet
Objectif	Le chef de projet doit pouvoir consulter les heures pour un projet/ phase d'un projet.
Préconditions	Doit être authentifié. Doit être chef de projet.
Déclencheur	L'administrateur choisit « Heures par projet »
Scénario nominal	Le système affiche un formulaire de recherche avec tous les projets (tous, actif, terminés). Le chef de projet remplit les valeurs du formulaire et sélectionne « soumettre » une fois terminé. Le système effectue la recherche. Le système retourne les données recherche sous forme de tableau dont toutes les colonnes peuvent être triées et filtrées. Les groupements doivent aussi être possibles pour permettre plus d'options d'analyse
Extensions	-

Table 12 : UC12, Consultation des heures par projet/phase

4.10.2 Consultation de la situation des tâches par projet

	Description
Acteur principal	Chef de projet
Objectif	Le chef de projet doit pouvoir consulter la situation des activités par projet/phase
Pré-conditions	Doit être authentifié. Doit être chef de projet.
Déclencheur	L'administrateur choisit « Consultation de la situation des activités par projet/phase »
Scénario nominal	Le système affiche un formulaire de recherche avec tous les projets et l'état des activités (ouvertes, réglées, délai dépassé). Le chef de projet remplit les valeurs du formulaire et sélectionne « soumettre » une fois terminé. Le système effectue la recherche. Le système retourne les données recherchées sous forme de tableau dont toutes les colonnes peuvent être triées et filtrées. Les groupements doivent aussi être possibles pour permettre plus d'options d'analyse.

Extensions	-

Table 14 : UC14, Consultation de la situation des activités par projet/phase

4.10.3 Consultation du diagramme de Gantt par projet/phase

	Description
Acteur principal	Chef de projet
Objectif	Le chef de projet doit pouvoir consulter le diagramme de Gantt par projet/phase
Pré-conditions	Doit être authentifié. Doit être chef de projet.
Déclencheur	L'administrateur choisit « Consultation du diagramme de Gantt par projet/phase »
Scénario nominal	Le système affiche un formulaire de recherche avec tous les projets. Le chef de projet remplit les valeurs du formulaire et sélectionne « soumettre » une fois terminé. Le système effectue la recherche. Le système retourne le diagramme Gantt qui correspond aux critères. Depuis ce diagramme, le chef de projet doit pouvoir accéder les tâches et les modifier.
Extensions	-

Table 15 : UC15, Consultation du diagramme de Gantt par projet

4.11 Gestion des heures

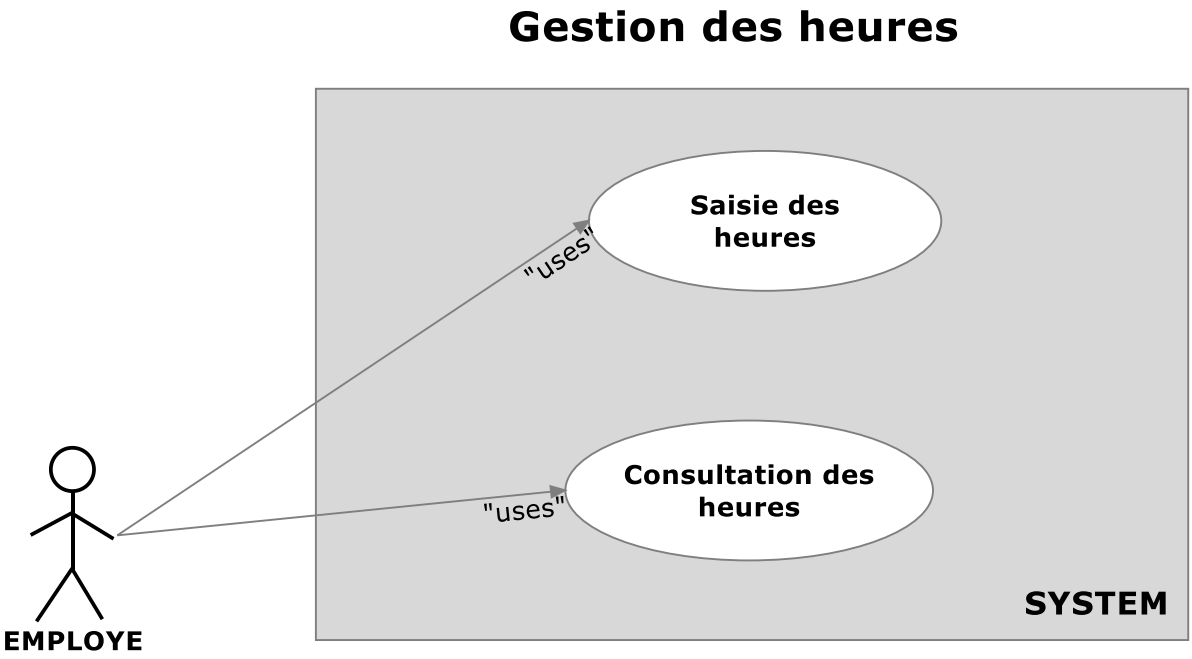

Figure 18 : Collecte de cas d'utilisations gestion des heures

4.11.1 Saisie des heures

	Description
Acteur principal	Employé
Objectif	L'employé doit pouvoir saisir ses heures
Pré-conditions	Doit être authentifié. Doit être employé.
Déclencheur	L'employé choisit « Saisies des heures ».
Scénario nominal	L'utilisateur choisit le projet, la tâche, la date, la description et le nombre d'heures. L'utilisateur soumet les valeurs au serveur. Serveur valide les données, les enregistre et retourne le résultat à l'utilisateur.
Extensions	L'enregistrement échoue si : Les valeurs minimales ne sont pas remplies

Table 16 : UC16, Saisie des heures

4.11.2 Consultation des heures

	Description
Acteur principal	Employé
Objectif	L'employé doit pouvoir consulter ses heures entre deux dates
Pré-conditions	Doit être authentifié.
Déclencheur	L'employé choisit « Mes heures »
Scénario nominal	Le système affiche un formulaire de recherche avec date début et date fin. L'employé remplit les valeurs du formulaire et sélectionne « soumettre » une fois terminé. Le système effectue la recherche. Le système retourne les données recherchées sous forme de tableau dont toutes les colonnes peuvent être triées et filtrées. Les groupements doivent aussi être possibles pour permettre plus d'options d'analyse.

Table 17 : UC17, Consultation des heures par projet/phase

4.12 Annotation

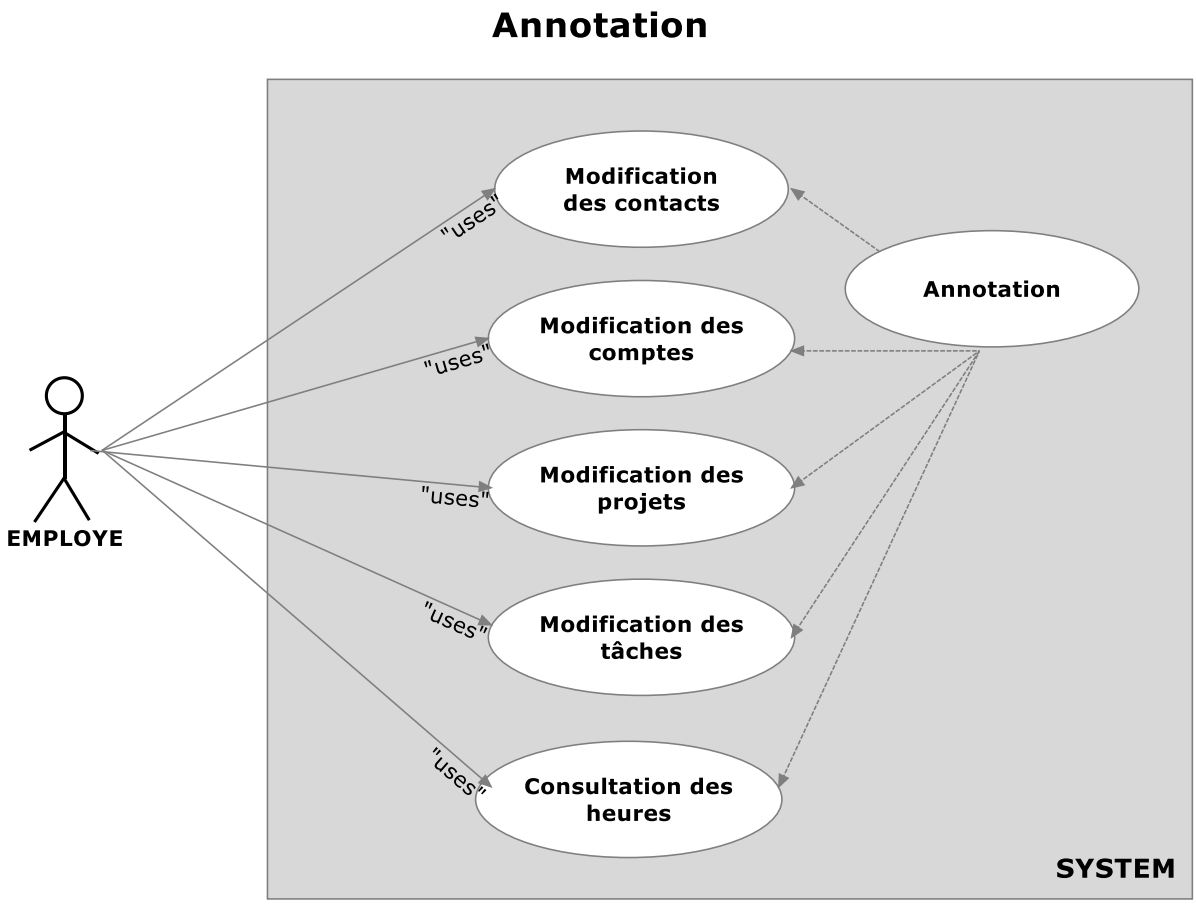

Figure 19 : Collecte de cas d'utilisations annotation

4.12.1 Annotation

	Description
Acteur principal	Employé
Objectif	L'employé doit pouvoir annoter n'importe quelle entité du système
Préconditions	Doit être authentifié.
Déclencheur	L'employé choisit modification d'une entité et après annotation.
Scénario nominal	En partant de la modification d'une entité, l'utilisateur ouvre le masque de l'annotation. L'utilisateur remplit les valeurs et soumet les données dans le

	système. Le système valide les données et les enregistre.
Extensions	

5

Analyse et conception

5.1 Architecture du système

Afin de simplifier la mise à jour de l'application et d'en faciliter l'accès, le futur système va être développé sur la base d'une application Web. Une application Web se trouve sur un serveur et se manipule à l'aide d'un navigateur Web (IE, Firefox, Chrome, …), via un réseau informatique (Internet, intranet, réseau local, etc.). Dans la technologie client-serveur, utilisée pour le World Wide Web, la communication est organisée par l'intermédiaire d'un réseau et *d'un interface Web entre plusieurs ordinateurs." cela signifie que des machines clientes (machines faisant partie du réseau) contactent un serveur, une machine généralement très puissante en terme de capacités d'entrées-sorties, qui leur fournit des services. Lesquels services sont exploités par des programmes, appelés programmes clients, s'exécutant sur les machines clientes."[Pil06]*

L'application Web s'oriente autour d'un serveur Web sur lequel est branché le logiciel applicatif, le tout accompagné d'un serveur de base de données.

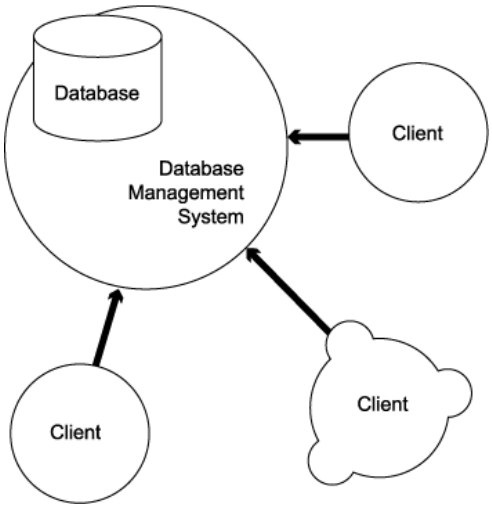

Figure 17 : Modèle client-serveur

Un des principaux avantages d'une application Web est que son utilisation est totalement indépendante du matériel de l'utilisateur. Qu'il utilise un Mac, un PC ou un autre environnement, l'utilisateur n'a besoin que d'un explorateur Web pour accéder au système.

5.1.1 Design pattern MVC

Dans le souci de développer une application stable et évolutive, l'utilisation d'une architecture standardisée est primordiale. Le Design Pattern Model-View-Controller est une des techniques de codage les plus répandues, principalement dans le développement Web. Les nouvelles possibilités d'interfaçages riches apportent de nouveaux défis. Le pattern MVC répond à ces challenges en offrant la possibilité de créer des applications plus flexibles.

Rassembler les vues et les données occasionne plusieurs désavantages :

- La difficulté de gérer les données de l'extérieur de l'objet.
- La difficulté de créer différentes interfaces utilisateurs avec la même logique
- La difficulté de faire évoluer les interfaces

Le concept de base du pattern MVC repose sur la séparation des données et des vues dans des logiques différentes. Le pattern MVC est basé sur trois couches:

- **Le modèle**
  Il décrit ou contient les données manipulées par l'application. Dans le cas typique d'une base de données le modèle offre des méthodes pour mettre à jour ces données (insertion, suppression, changement de valeur). Il offre aussi des méthodes pour récupérer ces

données. Les résultats renvoyés par le modèle sont dénués de toute présentation. C'est le modèle qui contient toute la logique métier de l'application. [Web12]

- **La vue**
  La vue correspond à l'interface avec laquelle l'utilisateur interagit. Sa première tâche est de présenter les résultats renvoyés par le modèle qui lui sont passés par le contrôleur. Sa seconde tâche est de recevoir toutes les actions de l'utilisateur (clic de souris, sélection d'une entrée, boutons, soumission de formulaires). Ces différents événements sont envoyés au contrôleur. La vue n'effectue aucun traitement, elle se contente d'afficher les résultats des traitements effectués par le modèle. [Web12]
- **Le contrôleur**
  Le contrôleur prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle et les synchroniser. Il reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer.
  Si une action nécessite un changement des données, le contrôleur demande la modification des données au modèle et ensuite avertit la vue que les données ont changé pour qu'elle se mette à jour. [Web12]

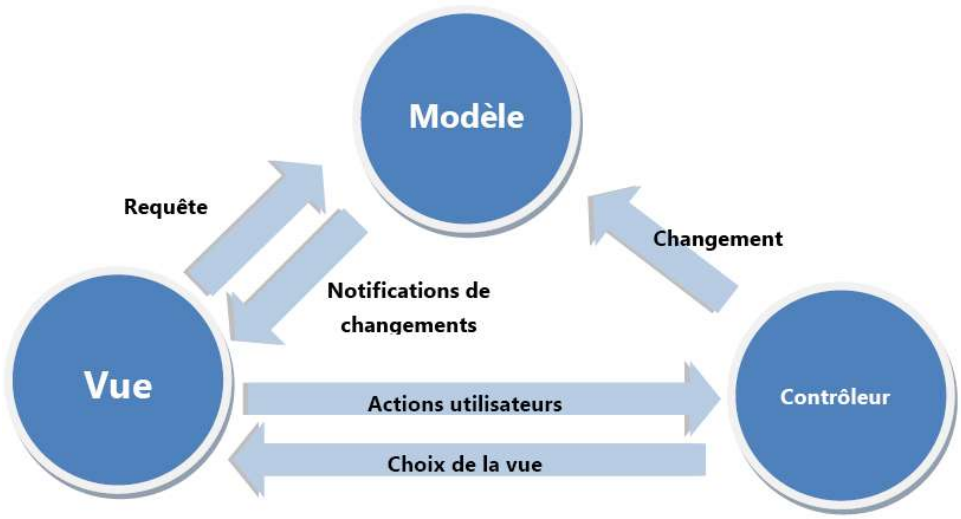

Figure 18 : Model-View-Controler

En résumé, lorsqu'un client envoie une requête à l'application :

1. la requête est analysée par le contrôleur
2. le contrôleur demande au modèle approprié d'effectuer les traitements
3. le contrôleur renvoie la vue adaptée.

5.1.2 Le framework « Codeigniter »

Afin d'assurer un développement rapide en respectant le modèle MVC, le nouveau logiciel va être développé sur la base d'un framework de développement.

Un framework PHP est un ensemble de codes qui fournit une organisation ainsi qu'un grand nombre de fonctionnalités. Il fournit un socle solide avec une organisation bien définie ansi qu'une multitude de classes et fonctions.

Il nous permet de ne nous occuper que de notre site et de laisser le reste à d'autres développeurs, c'est-à-dire sa base, son socle, mais aussi tout ce qui s'articule autour : les classes, les fonctions, etc.

Un certain nombre de framework « Open source sont disponibles. Les principaux du marchés sont les suivants : Symphony, cakePHP, Yii, Zend framework. Notre choix s'est porté sur « Codeigniter » pour les raisons suivantes :

- Légerté, simplicité et rapidité
- Respet strict du modèle MVC
- Compatibilité avec PHP5
- Grand nombre de bibliothèques développées
- Grande communauté d'utilisateurs et grande documentation

Voici le schéma de fonctionnement de Codeigniter extrait de la documentation. Il illustre bien son fonctionnement.

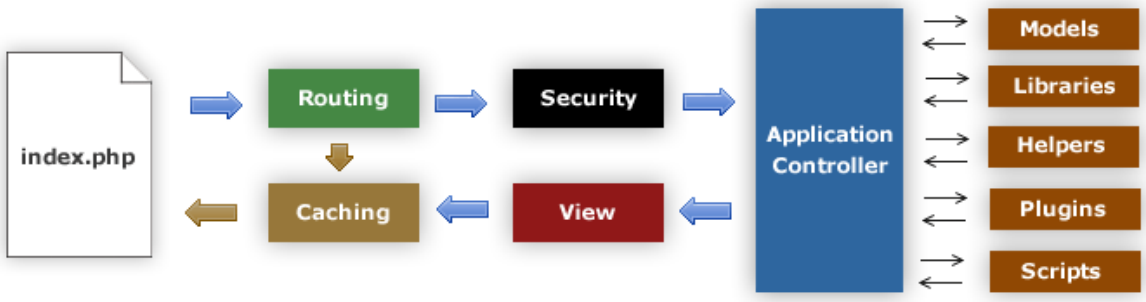

Figure 18 : Fonctionnement Codeigniter

- index.php – C'est toujours le fichier index.php, situé à la racine du répertoire, qui sera appelé en premier.
- Routing - C'est le routeur. C'est lui qui récupérera l'URL et la décomposera en actions. Il a pour rôle de trouver le contrôleur à appeler.
- Caching - Le module de cache contrôle s'il existe des fichiers mis en cache pour cette action. Dans le cas d'une réponse positive, ces fichiers vont être renvoyés au navigateur.
- Security - Cette partie sécurise toutes les données entrantes : cookies, variables get, post, etc. C'est la dernière étape avant le contrôleur.

- Application Controller – C'est ici que le développement PHP commence.
- Models, Libraries, Helpers… - Le contrôleur fait appel à différents éléments qui vont lui renvoyer des données.
- View - Les vues sont les fichiers qui contiennent le code HTML [Web14].

5.1.3 Un exemple MVC avec codeigniter

L'exemple ci-dessous illustre l'utilisation du framework Codeigniter pour l'affichage du listing des contacts.

Configuration des accès à la base de données

Pour configurer la base de données, il suffit de modifier le fichier *database.php* du dossier *config*.

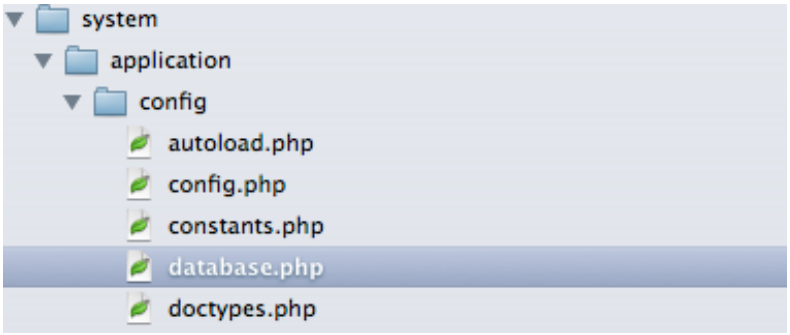

Figure 19: Fichier de configuration de la base de données

La configuration est très simple, il suffit de renseigner les différents paramètres d'accès à notre base de données mysql.

```
39
40  $db['default']['hostname'] = "localhost";
41  $db['default']['username'] = "dbase1";
42  $db['default']['password'] = "********";
43  $db['default']['database'] = "dbase1";
44  $db['default']['dbdriver'] = "mysql";
45  $db['default']['dbprefix'] = "";
46  $db['default']['pconnect'] = TRUE;
47  $db['default']['db_debug'] = TRUE;
48  $db['default']['cache_on'] = FALSE;
49  $db['default']['cachedir'] = "";
50  $db['default']['char_set'] = "utf8";
51  $db['default']['dbcollat'] = "utf8_general_ci";
52
```

Figure 20: Configuration de la connexion à la base de données

Chargement de la librairie *database*

Une fois les paramètres de connexion définis, il suffit de charger la libraire afin de pouvoir l'utiliser pour exécuter nos requêtes. Les librairies tels que *database*, qui seront utilisées dans pratiquement tous les contrôleurs, peuvent se charger de manière automatique en renseignant la variable de configuration *$autolaod* du fichier *autoload.php.*

```
41
42  $autoload['libraries'] = array('session','autoload');
43
44
```

Figure 21: Variable $autoload

Exemple d'affichage de contacts

Dans un environnement MVC, toute la partie traitant des données est centralisée dans les modèles. Dans *codeigniter*, chaque modèle correspond à une classe enregistrée dans un fichier. Ces fichiers sont réunis dans le dossier *models.*

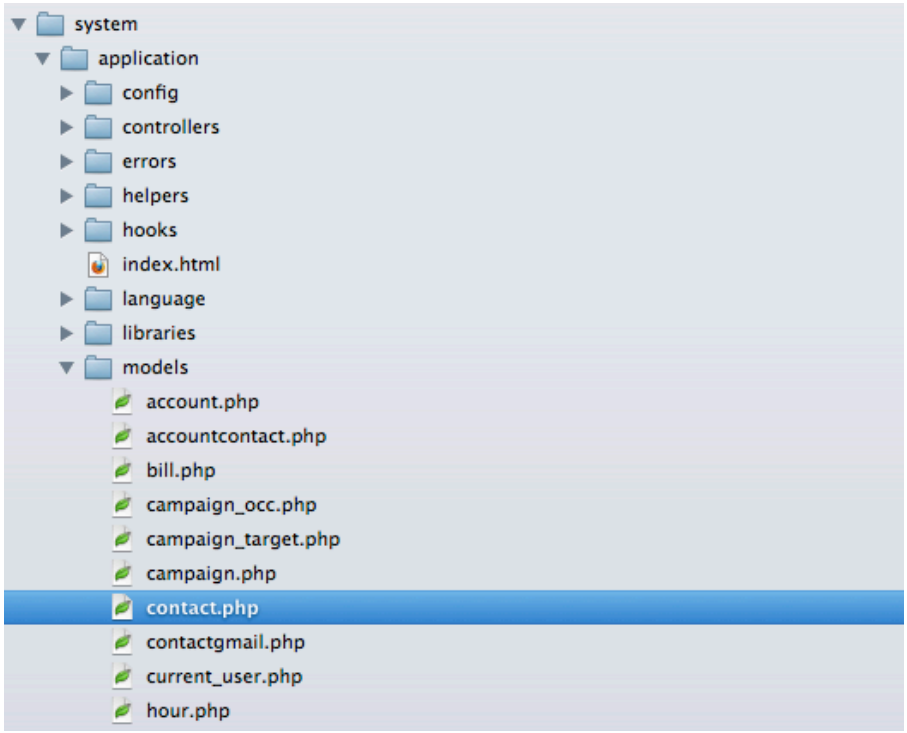

Figure 22 : Le fichier contact.php

Codeigniter offre la possibilité, d'une manière très simple, de travailler avec l'ORM (Object Relational Mapper) "Doctrine", un des DBAL (Database Abstration Layer) les plus puissants écrit en PHP.

Voici le modèle *Contact* simplifié pour notre exemple:

```
1     class Contact extends Doctrine_Record
2     {
3         public function setTableDefinition()
4         {
5             $this->hasColumn('id', 'integer', null);
6             $this->hasColumn('firstname', 'string', 255);
7             $this->hasColumn('lastname', 'string', 255);
8             $this->hasColumn('street', 'string', 255);
9             $this->hasColumn('zip', 'string', 255);
10            $this->hasColumn('city', 'string', null);
11        }
12
13    }
```

La classe « contact » définit le modèle « Contact ». La méthode « setTableDefinition » permet de définir les différents attributs du modèle.

Une fois le modèle correctement défini, il est très simple d'accéder aux données du modèle :

```
1     $q = Doctrine_Query::create()
2         ->select('c.*')
3         ->from('contact c')
4
5     $contacts = $q->execute()->toArray( true );
```

Ou d'enregistrer des nouveaux éléments :

```
1     $m = Doctrine::getTable('contact')->find( 5 ) ;
2     $m->firstname = 'Victor' ;
3     $m->save() ;
```

Du *modèle* à la *vue* en passant par le *contrôleur*

Les modèles permettent de sauvegarder et d'accéder aux données. Les contrôleurs, à l'aide des vues, vont permettre à l'utilisateur du système d'interagir avec ces dernières. Observons le contrôleur contacts, plus précisément sa méthode listing. Celui-ci est enregistré dans le fichier contacts.php dans le dossier *controlleurs.*

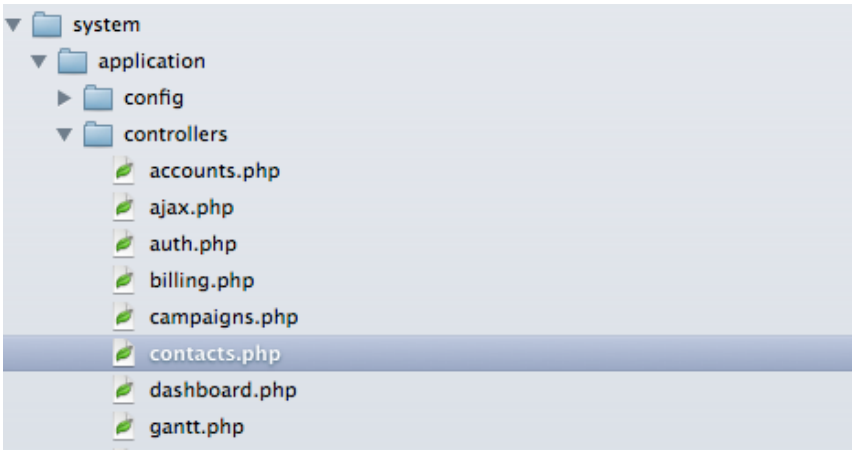

Figure 23 : Fichier contacts.php

Le contenu de la méthode listing, en simplifié :

```
1     class Contacts extends Controller
2     {
3         public function listing()
4         {
5             $q = new Doctrine_Pager(Doctrine_Query::create()
6                         ->select('c.*')
7                         ->from('contact c')
8                         ->orderBy("lastname, firstname");
9
10              $data = $q->execute()->toArray( true );
11              $vars['data'] = $data;
12              $this->load->view('contacts/listing', $vars );
13        }
14
15    }
```

Dans un premier temps. Une requête est effectuée sur le modèle *contact* afin d'en récupérer les données. Ces données sont enregistrées dans la variable *$vars* et seront transmises à la vue *contacts/listing*. La vue est alors affichée. Voici le code de la vue :

```
1     <html>
2     <head>
3     <body>
4     <h1>Liste des contacts</h1>
5     <table>
6     <?php foreach( $data as $row ):?>
7         <tr>
8         <td><?=$row['firstname']?></td>
9         <td><?=$row['lastname']?></td>
10        <td><?=$row['zip']?></td>
11        <td><?=$row['city']?></td>
12        </tr>
13    <?php endForeach ; ?>
14    </table>
```

5.1.4 Synchronisation mobile

Afin de pouvoir accéder à certaines données avec les téléphones mobiles, l'application devra d'une part être accessible sur un smart phone, et d'autre part mettre à disposition du téléphone la base de données "contact". A ce niveau du développement, seul la base de données "Contacts" sera synchronisée. Dans le future, une application mobile permettra d'accéder aux principales fonctions grâce à une interface web mobile, ou d'une application Ihpone et Android.

La synchronisation des contacts proposera dans un premier temps une Interface avec Google. Google propose une API très riche permettant de travailler en directe avec la base de données des contacts, de l'agenda, des mails, … Une fois les données synchronisées avec Google, elle sont très facilement accessibles via I-Phone ou Android grâce à Google Sync.

La librairie Zend Gdata est utilisée pour connecter les services API de Google. Il s'agit d'une série de classes permettant de travailler de manière aisée avec les différents services web de l'API Google Data.

Voici un exemple de synchronisation utilisé dans l'application. Les quelques lignes ci-dessous permettent de rajouter un contact dans « Google Contact ». La classe « Zend_Gdata » du framework « Zend Framework » permet de communiquer avec l'API de Google à l'aide de requêtes HTTP. Les requêtes contiennent différents paramètres encodés en XML. Les données sont retournées par les méthodes de la classe sous forme d' «Atom Feed ». Ces différents « feeds » peuvent ensuite être traités et repassés par après. Ces différentes méthodes offrent la possibilité d'ajouter, modifier ou supprimer des éléments de la base de données

Toutes les opérations effectuées sont sécurisées. L'utilisation des différentes fonctions proposées réussira seulement après une authentification avec le nom d'utilisateur et le mot de passe du propriétaire du compte « Google » à l'aide de la méthode « ClientLogin ». Cette opération est utilisée à la ligne 5 de l'exemple.

Cette classe utilise l'objet « Zend_HTTP_Client » à qui nous fournissions les paramètres de login. Elle permet l'ouverture d'une connexion authentifiée avec le service « Google Contact ».

Une fois connecté, l'utilisation de la méthode « createEntry » permet l'ajout d'un nouveau contact « Google ». Toutes les données du contact tels que le nom, prénom, e-mail, téléphones doivent être enregistrés dans un fichier XML. Les lignes 9 à 28 offrent un aperçu de la création d'un contact XML avec l'aide de la classe PHP « DOMDocument ».

```
15    try{
16      $client  =  Zend_Gdata_ClientLogin::getHttpClient($uname,  $pwd,
      'cp');
17
18      $gdata = new Zend_Gdata($client);
19      $gdata->setMajorProtocolVersion(3);
20
21      $doc  = new DOMDocument();
22
23     ...
24
25      // Insert Organization informations
26      $org = $doc->createElement('gd:organization');
27      $org->setAttribute('rel'
28                ,'http://schemas.google.com/g/2005#work');
29      $entry->appendChild($org);
30      $orgName = $doc->createElement('gd:orgName', $data->company);
31      $org->appendChild($orgName);
32
33      // Insert Contact Informations
34      $name = $doc->createElement('gd:name');
35      $entry->appendChild($name);
36      $fullName = $doc->createElement('gd:fullName', $data->firstname
      . ' ' . $data->lastname);
37      $name->appendChild($fullName);
38
39      $title = $doc->createElement('gd:title', $data->title );
40      $entry->appendChild($title);
41
42      ...
43
44
45      // Create google Contact
46      $entryResult = $gdata->createEntry( $doc->saveXML,
                null, $extra_header);
47    }
48    catch{
49       …
50    }
```

5.2 Modélisation du point de vue logique

5.2.1 Identification des entités

Une entité est un objet spécifique […] dans le monde réel ou dans notre pensée. Elle peut désigner une personne, un objet, un concept abstrait ou un événement. Les entités de même type forment un ensemble d'entités caractérisées par un certain nombre d'attributs. […]Pour chaque ensemble d'entités, nous définissons une clé d'identification, formé d'un attribut ou d'une combinaison d'attributs, qui permet de distinguer chaque entité de manière unique dans l'ensemble considéré. […] [Mei06]

Les entités du système avec leurs principaux attributs sont :

- **EMPLOYE (ID**, username, mot de passe, nom, prénom, taux d'occupation, rôle, coût par heure)
- **CONTACT (ID**, raison sociale, prénom, nom, fonction, adresse, code postal, ville, pays, département, tél. privé, tél. prof., email privé, email prof., site web, date de naissance, date de création, date de modification)
- **COMPTE (ID**, numéro, nom, adresse, état, date de création, date de modification)
- **PROJET (ID**, compte, numéro, nom, description, date début, date fin, chef de projet, statut)
- **TACHE (ID**, phase, responsable, description, délai)
- **HEURES (ID**, activité, employé, date, nombre d'heures, description)
- **NOTE (ID**, cible, date, note)

5.2.2 Identification des associations

Pour décrire le modèle logique, il est important de découvrir les liaisons entre les ensembles d'entités. Les associations sont des liens orientés d'une entité à une autre.

Selon Andreas Meier [Mei06], il existe 4 types d'association :

- Association simple (Type 1) : A chaque entité dans un ensemble d'entités correspond « une et une seule » entité dans un deuxième ensemble.
- Association conditionnelle (Type C) : A chaque entité dans un ensemble d'entités correspond « zéro ou une » entité dans un deuxième ensemble.
- Association multiple (Type m) : A chaque entité dans un ensemble d'entités correspondent « une ou plusieurs » entités dans un deuxième ensemble.
- Association multiple conditionnelle (Type mc) : A chaque entité dans un ensemble d'entités correspondent « aucune, une ou plusieurs » entités dans un deuxième ensemble.

Après le dénombrement des entités du système, les associations suivantes ont été identifiées :

- **CONTACT_ ENGAGEMENT** (contact): un CONTACT est engagé sur aucun, un ou plusieurs COMPTES, un COMPTE correspond à plusieurs CONTACTS.
- **PROJET_APPARTENANCE** (projet): un PROJET appartient à un seul COMPTE, un COMPTE correspond à aucun, un ou plusieurs PROJETS.
- **PROJET_COORDINATION** (projet): Un PROJET est coordonné par un EMPLOYE (chef de projet), EMPLOYE (chef de projet) coordonne aucun, un ou plusieurs projets.
- **EMPLOYE_PARTICIPATION** (employé): Un EMPLOYE participe à aucun, un ou plusieurs PROJETS, sur un PROJET travaillent plusieurs EMPLOYES.
- **TACHE_ATTRIBUTION** (activité): Une TACHE est attribuée à un PROJET, un PROJET a aucune, une ou plusieurs TACHE.
- **TACHE_REALISATION** (activité): Une TACHE est réalisée par un EMPLOYE, un employé peut réaliser aucune, une ou plusieurs tâches.
- **HEURE_APPARTENANCE** (heure): Une HEURE appartient à une TACHE, une TACHE est exécutée en aucune, une ou plusieurs HEURES.
- **HEURE_REALISATION** (heure): Une HEURE est réalisée par un employé, un EMPLOYE peut réaliser aucune, une ou plusieurs HEURES.
- **ANNOTATION** (projet, phase, activité, heure, employé, contact, compte,) : Une NOTE peut annoter aucune, une ou plusieurs de toutes les autres entités repérées, chacune de ces entités peut avoir aucune, une ou plusieurs NOTES.

5.2.3 Modèle Entité – Association

Le diagramme Entité-Association global est décomposé en 2 diagrammes. Le premier contient tout sauf l'annotation. Les notes sont traitées à part à cause de leur caractère complémentaire et indépendant du système. Cette séparation nous permet également d'alléger le diagramme de base.

Diagramme de base

En premier, notre intérêt est porté sur les éléments indispensables du système.