ISA: Rapport - Iteration 1

Encadrant: Sébastien Mosser & Anne-Marie Déry
Client: Société des Cimes

Groupe AD

MARIN Alicia DA COSTA FERNANDES Islame Felipe DIAS DOS SANTOS JUNIOR Ronaldo

5 mars 2016

Table des matières

1 Introduction

Suite à la situation d'enneigement de ces dernières années, et d'un souci de modernisation des systèmes d'informations ; la société d'exploitation de remontés mécanique, la "Société des Cimes" a lancé un appel d'offre concernant le développement d'une application, permettant de proposer des services plus innovants aux usagers. Ce rapport vise à fournir une modélisation architecturale du système exigé par la société, en prenant en compte ses fonctionnalités et les différentes vues de sa modélisation, telles que la vue fonctionnelle, la vue développement et la vue déploiement. Dans ce contexte, seront présentés les diagrammes de : cas d'utilisation, de composants, de classes, modèle relationnel et déploiement. Bien sûr, une explication sera aussi donnée pour les transitions entre le diagramme de classes et le diagramme relationnel.

2 Vue Fonctionnelle

Afin d'avoir une vision du comportement fonctionnel de notre application, nous avons décrit un diagramme de cas d'utilisation, ainsi que des scénarios Cockburn qui viennent détailler et clarifier certains comportements du diagramme.

2.1 Diagramme de cas d'utilisation

Les cas d'utilisation sont des outils formels qui permettent de décrire les comportements de chaque sous-système. Les besoins des utilisateurs sont exprimés sous forme de fonctionnalités (ou cas d'utilisation) ainsi que d'interactions avec ces utilisateurs. Vous trouverez ci-dessous les 3 systèmes que nous avons décidé de réaliser.

2.1.1 Système client

Le premier système comporte l'ensemble des fonctionnalités concernant le skieur.
Un skieur a la possibilité « *d'acheter un forfait* ». Il a deux façon de faire, soit par le site web, soit directement à la caisse. De plus, un usager peut retirer une carte dans une caisse choisie lors de l'achat.
Il peut également demander un remboursement du forfait en cas de fermeture de la station pour raison météorologique.
En ce qui concerne la validation de la carte pour avoir accès aux remontés, il y a le cas d'utilisation « *Valider la carte* ». Celui-ci représente la lecture de la carte par le lecteur NFC et sa validation, ce qui pourra ou non activer les portillons d'accès. La relation « extend » signifie que le cas d'utilisation « *Activer les portillons d'accès* » n'est pas toujours obligatoire lorsque le client « *Valide la carte* ». Les portillons s'ouvriront si et seulement si la carte est bien validée. Les détails de ce cas d'utilisation est donné au Scénario Cockburn de la Figure 4.

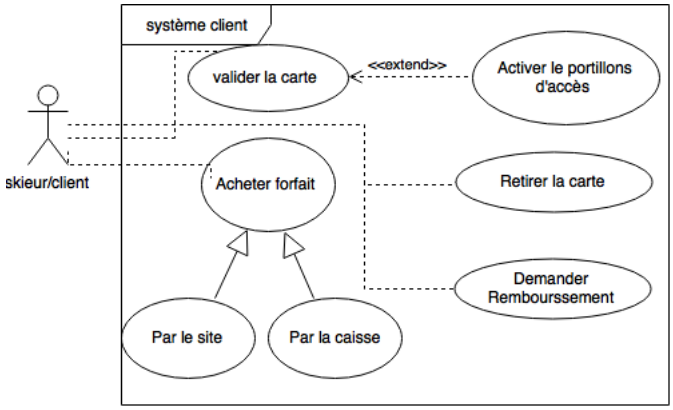

FIGURE 1 – Diagramme d'utilisation pour le système client

2.1.2 Système web de la société

Le système web pour la société est un peu plus complexe que le précédent. En effet, pour celui-ci, nous proposons trois acteurs. Le premier est un employé de la société, qui aura l'autorisation de manipuler le système web. Le second est un agent de caisse, qui aura la fonctionnalité « *Vente de forfait* » en plus des autorisations d'un employé de la société. Enfin, le dernier est un commerçant. il aura la possibilité de s'abonner aux statistiques afin d'avoir le droit de les visualiser.

Nous distinguons « *Visualiser les statistiques de vente de forfait* » et « *Visualiser les statistiques par abonnement* » car les abonnés n'auront pas forcément accès à l'ensemble des statistiques de la société (la période de temps peut être réduite). Ces deux cas d'utilisation sont détaillés dans la section suivant. La relation « extend » entre « *Visualiser les statistiques par abonnement* » et « S'abonner aux statistiques » signifie que la visualisation n'est pas une fonctionnalité indispensable à l'abonnement.

Pour les deux autres acteurs, nous proposons un ensemble de cas d'utilisation dont le fonctionnalités nécessitent toujours l'authentification de l'employé de la société (voir la flèche « include »). Cette authentification est importante pour bien identifier l'acteur qui utilise le système web. De la même manière, pour « *Contacter des usagers* » il faut obligatoirement « *Choisir le profil* » de celui que nous souhaitons contacter.

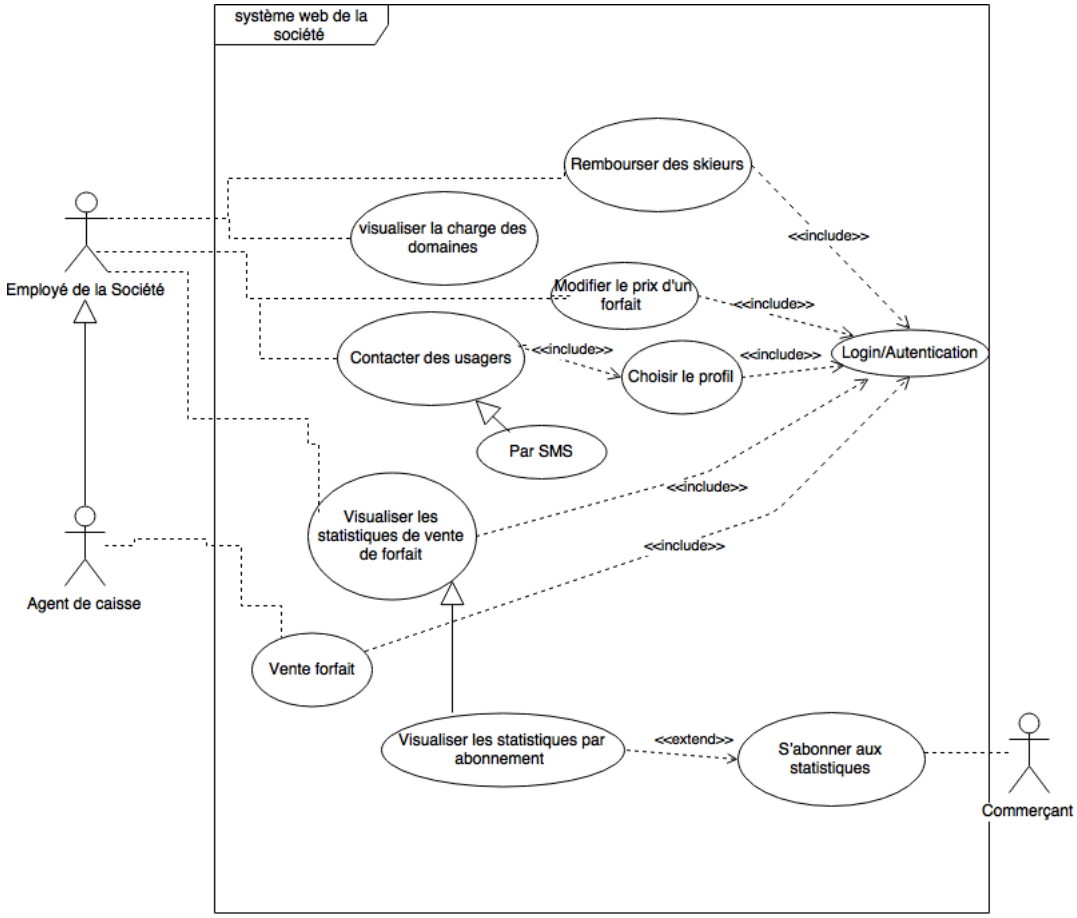

FIGURE 2 – Diagramme d'utilisation pour le système web de la société

2.1.3 Système d'affichage

Pour le système d'affichage, un pisteur est capable de mettre à jour les panneaux présents sur les pistes.

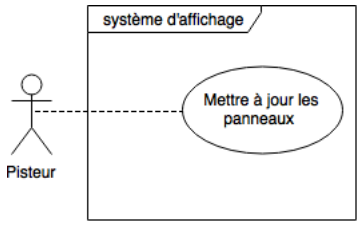

FIGURE 3 – Diagramme d'utilisation pour le Système d'affichage

2.2 Scénario Cockburn

Dans cette section, quelques cas d'utilisation seront détaillés au format des Scénarios Cockburn. L'objectif est de détailler de la manière la plus claire possible les cas d'utilisation qui ne sont pas intuitifs.

2.2.1 Usager

Cas d'utilisation : Valider la carte **Acteur Primaire :** Skieur	
Précondition : Être en possession d'une carte valable	
Scénario primaire : 1. Le skieur arrive près du lecteur NFC 2. Il positionne sa carte face au lecteur afin d'enclencher la lecture 3. Le lecteur lit la carte 4. La carte est validée 5. Les portillons s'ouvrent, donnant accès aux remontés	**Variantes :** — 3.1 Il y a eu une erreur de lecture. Un message d'erreur est retourné. — 4.1 La carte n'a pas été validée. Une message d'erreur est retournée. L'action 5. n'est pas exécutée.
Postcondition : — Les portillons se sont ouverts — Les portillons sont restés fermés	
Données nécessaires : La liste des forfaits ainsi que leurs droits d'accès	

FIGURE 4 – Scénario Cockburn du cas d'utilisation « Valider la Carte »

2.2.2 Société

Cas d'utilisation : Rembourser un usager **Acteur Primaire :** Un agent de la société	
Précondition : Un usager a fait une demande de remboursement d'un forfait et l'agent de la société s'est bien authentifié au système.	
Scénario primaire : 1. Dans l'interface « Remboursement » du système, l'agent de la société va rechercher le numéro de la carte. 2. Le forfait est trouvé 3. Le forfait est non-activé ou la station est fermé 4. La société clique sur le bouton « remboursement de l'usager ».	**Variantes :** — 2.1 Le forfait n'est pas repertorié — 3.1 La station est ouverte et le forfait est activé Il n'y a pas de remboursement
Postcondition : — Le remboursement a été accepté — Le remboursement a été refusé — Le remboursement n'a pas pu être traité	
Données nécessaires : La liste des forfaits, l'état météorologie ainsi que l'état de la station	

FIGURE 5 – Scénario Cockburn du cas d'utilisation « Rembourser un usager »

Cas d'utilisation : Visualiser les statistiques des ventes de forfait **Acteur Primaire :** Un agent de la société	
Précondition : Un agent de la société s'est authentifié au système	
Scénario primaire : 1. Sur l'interface « Statitisques » du système, une période de temps doit être choisie. 2. L'agent clique sur le bouton « Voir des statistiques » 3. Le système cherche la requête dans la base de données. 4. Le système affiche un graphe montrant la quantité vendue pour chaque forfait en fonction du temps	**Variantes :** — 1.1 Le système affiche un message d'erreur si la période choisie n'est pas cohérente — 3.1 Le système affiche un message d'erreur s'il y a eu des problèmes pour accéder à la base de données. — 4.1 Il n'y a pas des ventes de forfaits dans la période choisie.
Données nécessaires : Les statistiques de ventes	

FIGURE 6 – Scénario Cockburn du cas d'utilisation « Visualiser les statistiques »

Cas d'utilisation : Visualiser les statistiques par abonnement **Acteur Primaire :** Commerçant	
Précondition : Le commerçant doit être abonné aux statistiques de la société	
1. Le commerçant s'authentifie à l'aide de son numéro d'abonné 2. Le système valide du numéro du commerçant 3. Une période de temps doit être choisie. 4. Le commerçant clique sur le bouton « Voir des statistiques » 5. Le système cherche la requête dans la base de données, prenant en compte quelles sont les statistiques qui le commerçant peut voir 6. Le système affiche un graphe montrant la quantité vendue pour chaque forfait en fonction du temps	**Variantes :** — 2.1 Le système affiche un message d'erreur si le numéro du commercant n'est pas valide — 3.1 Le système affiche une message d'erreur si la période choisie n'est pas cohérente — 5.1 Le système affiche un message d'erreur s'il y a eu des problèmes pour accéder à la base de données. — 6.1 Il n'y a pas des ventes de forfaits dans la période choisie.
Données nécessaires : Les statistiques de ventes et la liste d'abonnés	

FIGURE 7 – Scénario Cockburn du cas d'utilisation « Visualiser les statistiques par abonnement »

2.3 Diagramme de composants

A la différence du diagramme de cas d'utilisation, le diagramme de composant ne décrit pas les fonctionnalité du système mais décrit les composants nécessaires pour mettre en place ces fonctionnalités. Sur ce diagramme, nous avons illustrés différents composants ainsi que les relations entre chacun que nous allons expliquer ci-dessous.

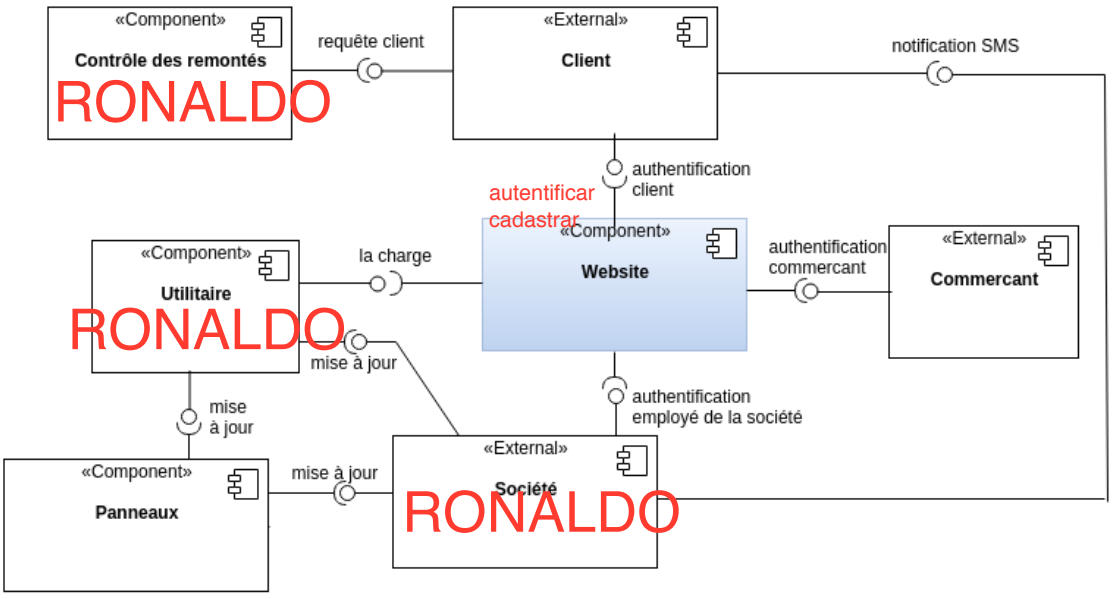

FIGURE 8 – Diagramme de composant global

Le composant « Contrôle des remontés » représente ce qui va gérer les autorisations d'accès aux remontés lorsque le client présentera sa carte NFC. Le composant « Utilitaire » décrit les remontés et ainsi que les pistes. Il communiquera sa charge au site web et leurs états (fermé/ouvert/damée) pourront être mis à jour par la société. Le composant « Panneaux » illustre les différents panneaux d'affichages présents sur les pistes. Ils seront mis à jour grâce aux pisteurs, des employé des la société particuliers. Le composant « Website » constitue l'application web de la société. Les composants internes sont représentés en Figure 9. La gestions des ventes de forfaits, les requêtes de remboursement ainsi que la visualisation des statistiques sera gérée sur cette plateforme. Nous avons représenté le client, la société ainsi que le commerçant en tant que composant externe.

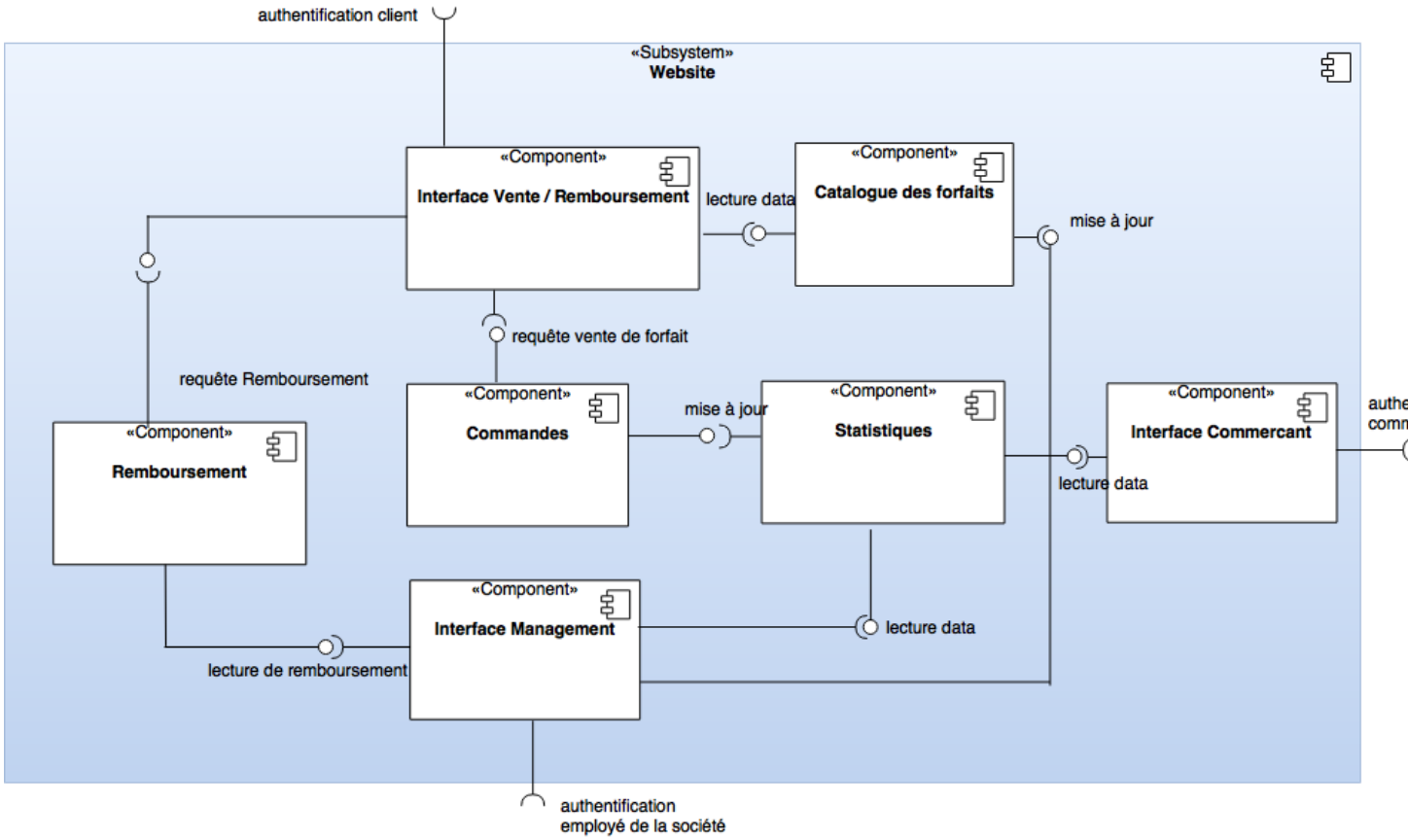

FIGURE 9 – Diagramme de Composants du Website

Toute personne souhaitant interagir avec le site web devra obligatoirement s'authentifier. Suivant la catégorie de personne, l'authentification ne mènera pas à la même interface de gestion. Un client aura accès au catalogue des forfaits, et aura la possibilité de réclamer un remboursement. Chaque commande de forfait mettra à jour les statistiques de vente, qui seront visibles par les commercants abonnés ainsi que la société. La société pourra de plus lire les charges des remontés, communiqué par le composant « Utilitaire »

3 Vue Developpement

3.1 Diagramme de classe des métiers

Au niveau de la modélisation orientée objet, il est très important de conceptualiser le diagramme de classe pour que l'on puisse avoir une vision de la structure interne du système, grâce à une représentation de ses objets et des relations entre eux. Dans cette section, nous proposons un diagramme de classe qui vise à comprendre au maximum les objets du système, tels que les forfaits, les états des utilitaires (remontés/pistes) d'un domaine et les différents affichages possibles. Dans les sections suivantes, nous détaillerons chaque partie du diagramme dont la représentation globale est donnée par la Figure 15.

3.1.1 Forfait

L'objectif principal de cet ensemble de classes est de modéliser les différents forfaits et ses relations spécifiques avec la station. Nous proposons deux grands groupes de forfaits : ceux qui donnent accès à un

nombre réduit de remontés et ceux qui donnent accès à des domaines entiers. Ces deux spécialisations sont représentées par deux classes abstraites, c'est-à-dire, qui ne pourront pas être instanciées. En effet, les instances ne seront que des sous-classes avec une particularité pour chacune. Elles devront toutes implémenter la méthode « donnerAcces » qui reçoit soit un Domaine soit une Remontée en paramètre, selon son type. Les classes « ForfaitSaison » et « Fidélicimes » devront aussi implémenter d'autres méthodes spécifiques à ces types de forfait. L'objectif principal de cette approche est de bien représenter les différents forfaits ainsi que leurs caractéristiques distinctes. Les catégories, c'est-à-dire les tranches d'âge sont données par une énumération. Nous pourrons combiner ces catégories avec chaque type de forfait. Par exemple un forfait Saison Adulte ou un forfait Fidélicimes Age d'Or. Si le forfait n'a aucune particularité, nous utiliserons la classe « ForfaitClassique ».
Encore dans cette partie du diagramme de classe, nous avons illustré les classes concernant les achats et les ventes de forfait. La classe mère Achat/Vente a deux classes filles qui sont « AchatOnline » et « AchatCaisse ».
Nous modèlisons aussi des objets pour les clients, les cartes et les agents de caisse.

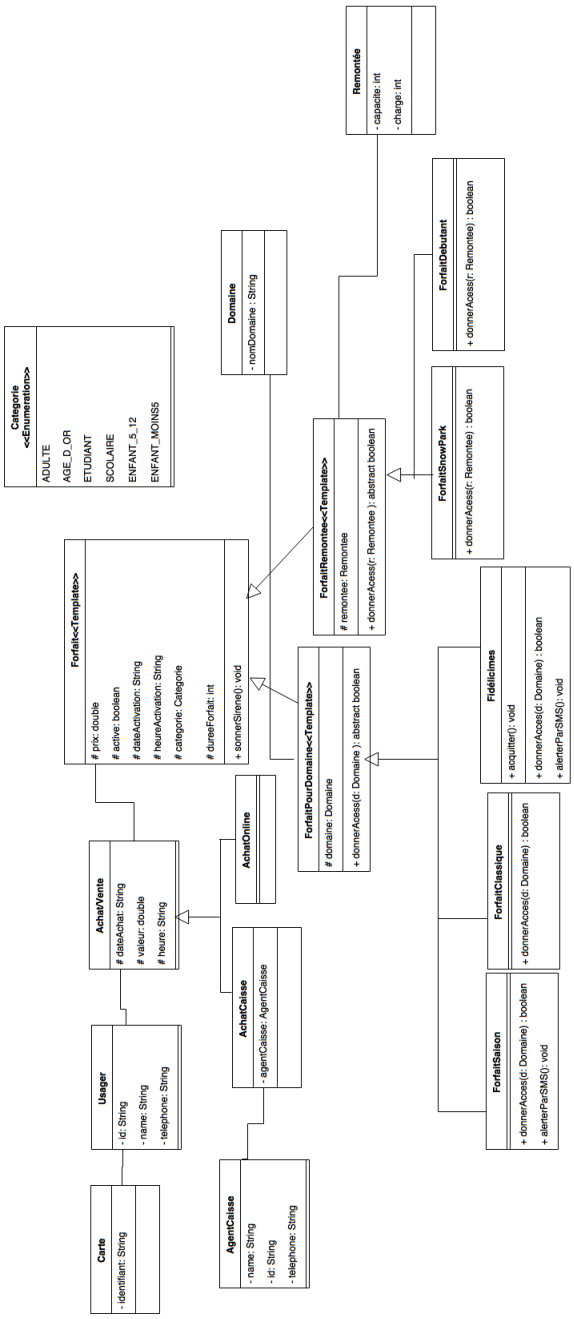

FIGURE 10 – Diagramme de classe - Forfait

3.1.2 Domaine

Le but de la partie du diagramme de classe représenté en Figure 11 est de modéliser les utilitaires du domaine (remontés et pistes) ainsi que leurs états. Le schéma de conception qui a été mis en oeuvre est le schéma de conception État. Pour comprendre comment cela fonctionne, nous allons prendre l'exemple de la classe « Utilitaire ». Elle possède un attribut du type « EtatUtilitaire » qui décrira son état à chaque instant : soit « Fermé », soit « Ouvert ». Celles-ci sont des classes Singleton, c'est-à-dire, que l'on n'aura qu'une seule instance de chacunes d'elles. Par défaut, les méthodes « ouvrir() » et « fermer() » de la classe EtatUtilitaire doivent retourner une exception, mais ces deux méthodes seront ré-écrites dans les singletons. Par exemple, la classe Ouvert doit implémenter la méthode « fermer() », pour que, lorsque l'état du utilitaire

sera « Ouvert », il puisse changer pour passer à « Fermé ». En revanche, la classe « Ouvert » ne ré-écrit pas la méthode « ouvrir() », donc utilise celle de la classe mère. Elle lèvera donc une exception, parce qu'un utilitaire « Ouvert » ne peut pas appeler la méthode « ouvrir() ». Le fonctionnement est respectivement le même pour la classe « Fermé » et pour les états de « Piste » (damée). Avec cette approche, nous pouvons avoir un meilleur contrôle des états et lancer des actions spécifiques à chaque fois que l'on change d'état, comme mettre à jour l'affichage des panneaux (voir la section suivant). C'est pourquoi nous ne l'avons pas modélisé avec une simple énumération.

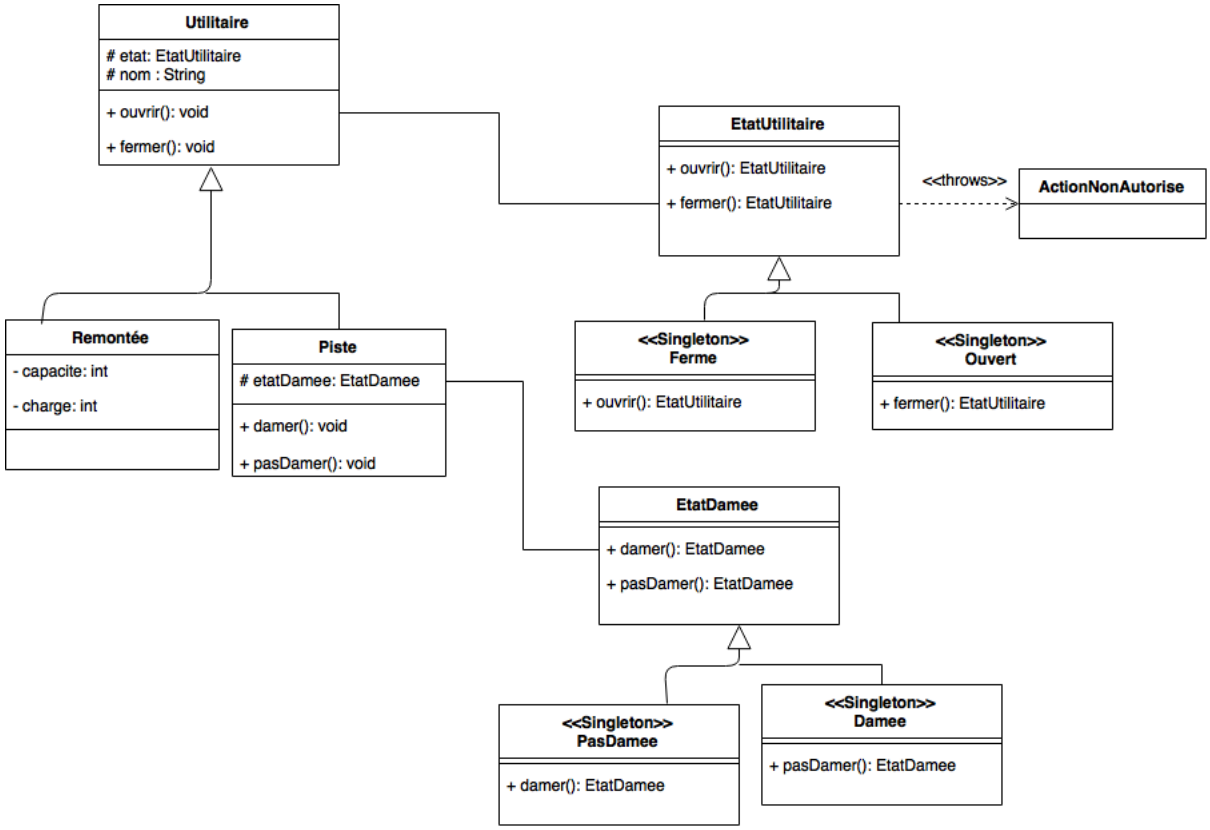

FIGURE 11 – Diagramme de classe - Domaine et état

3.1.3 Affichage

Le schéma de conception Observer a été mis en oeuvre pour modéliser le système d'affichage d'état des utilitaires du domaine. Le plus grand avantage de ce schéma est la facilité d'ajouter de nouveaux afficheurs pour un utilitaire ainsi que la flexibilité de les manipuler. Comme vous pouvez voir sur la Figure 12, chaque utilitaire aura deux afficheurs : le « Panneaux » et le « SiteWeb ». Ils implémentent tous les deux la méthode « afficher() » pour afficher, chacun à sa manière, l'état des utilitaires qui leur sont assignés. De plus, chaque afficheur peut gérer plusieurs utilitaires et un utilitaire peut être affiché par plusieurs afficheurs. Lorsqu'un afficheur est attaché à un utilitaire, il faut l'insérer au vector « _listeAfficheurs » de cet utilitaire. La méthode « notifier() », permet d'appeler les méthodes « afficher() » de chaque afficheur rattaché à cet utilitaire. Les méthodes qui changent l'état du utilitaire doivent ainsi appeler la méthode « notifier() » pour mettre à jour les afficheurs.

3.1.4 Global

Le diagramme de classe qui comprend les trois parties décrites ci-dessus sont représentés en Figure 15.

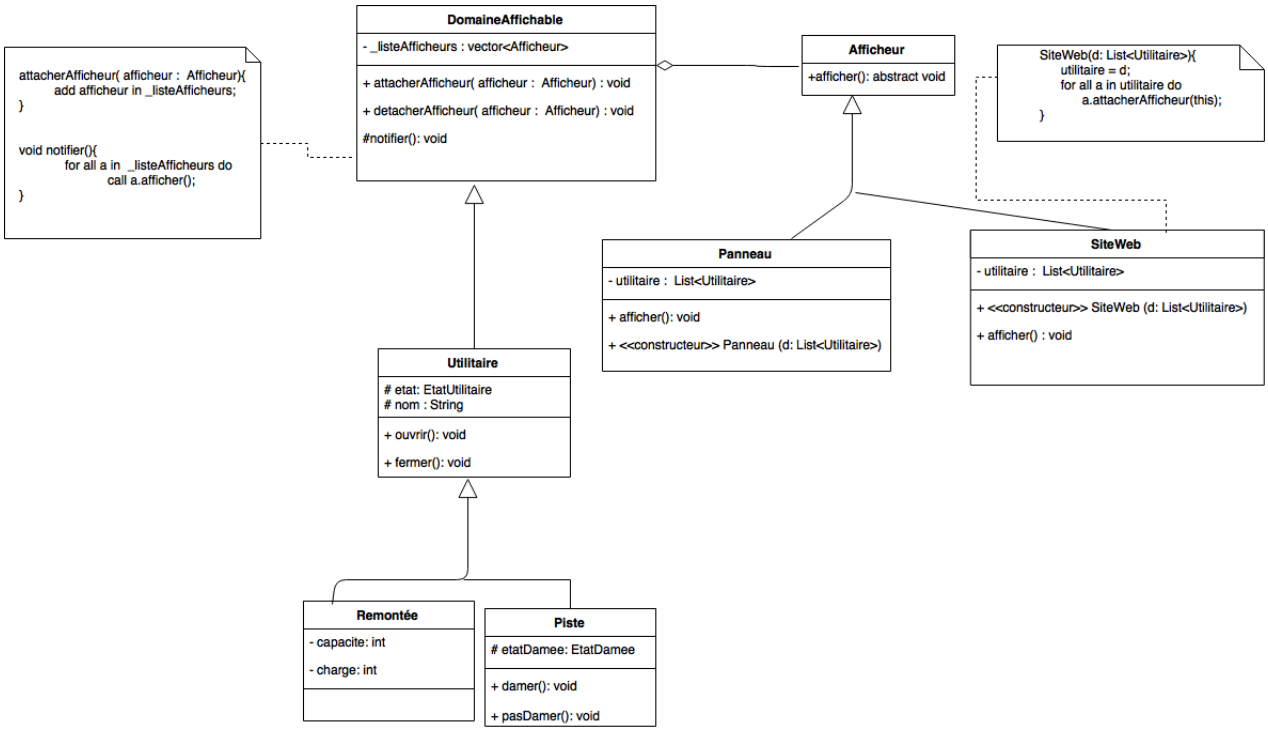

FIGURE 12 – Diagramme de classe - Affichage

3.2 Modèle relationnel de stockage

Du point de vue du stockage, le plus important est de savoir si les principales requêtes peuvent être exécutées de façon logique et rapide. Afin de réaliser une modèle relationnel, nous devons en premier mettre en place les objets principaux (forfait, skieur, Agent_Caisse, ..) du système, qui seront représentés sous forme de tableaux. Et seulement après, réfléchir aux relations entre chacun. Par exemple, un client peut acheter un ou plusieurs forfaits et chaque forfait peut être acheté par un ou plusieurs clients. Si on interprète le client et le forfait comme tableaux, nous devons associer au travers de jointure un unique client à un forfait dans une relation d'achat à une certaine date et horaire. Pour ce faire, nous pouvons créer une nouvelle table qui va contenir les clés primaires des deux tables primitives. En plus de cela, il y a d'autres techniques qui dépendent du type de relation entre les deux entités du système, mais le raisonnement est similaire à cette approche deux à deux. En utilisant cela, nous avons mis en place le diagramme suivant :

3.3 Explicitation du mapping objet - relationnel

Cependant, un autre point important dans le processus de construction du diagramme relationnel a été de regarder le diagramme de classe, notamment les interactions entre les objets principaux du système. Voyons quelques exemples : La relation Usager et Carte dans le diagramme de classe est typiquement une relation de type 1 pour 1. Cette situation demande une fusion des attributs des tables vers une table unique, que l'on a appelé Skieur. (De plus, l'union entre les clès primaires est aussi meilleur ;) La relation d'héritage entre les classes Achat/Vente, AchatCaisse et AchatOnline n'a pas beaucoup d'importance au niveau de la vision des données, donc nous avons effectué une fusion entre toutes les tables correspondants à ces classes dans un même table, que l'on a appelé Achat. Enfin, la modélisation hiérarchique de Forfait dans le diagramme de classes n'est pas essentielle du point de vue des requêtes. Afin de concevoir ce modèle au niveau des tables, il a été nécessaire de créer une table pour le type de forfait. Ainsi, la solution devient simple, pratique et elle ne surcharge pas les requêtes de base de données. Pour toutes les autres transformations, nous avons utilisé la principe de table d'association.

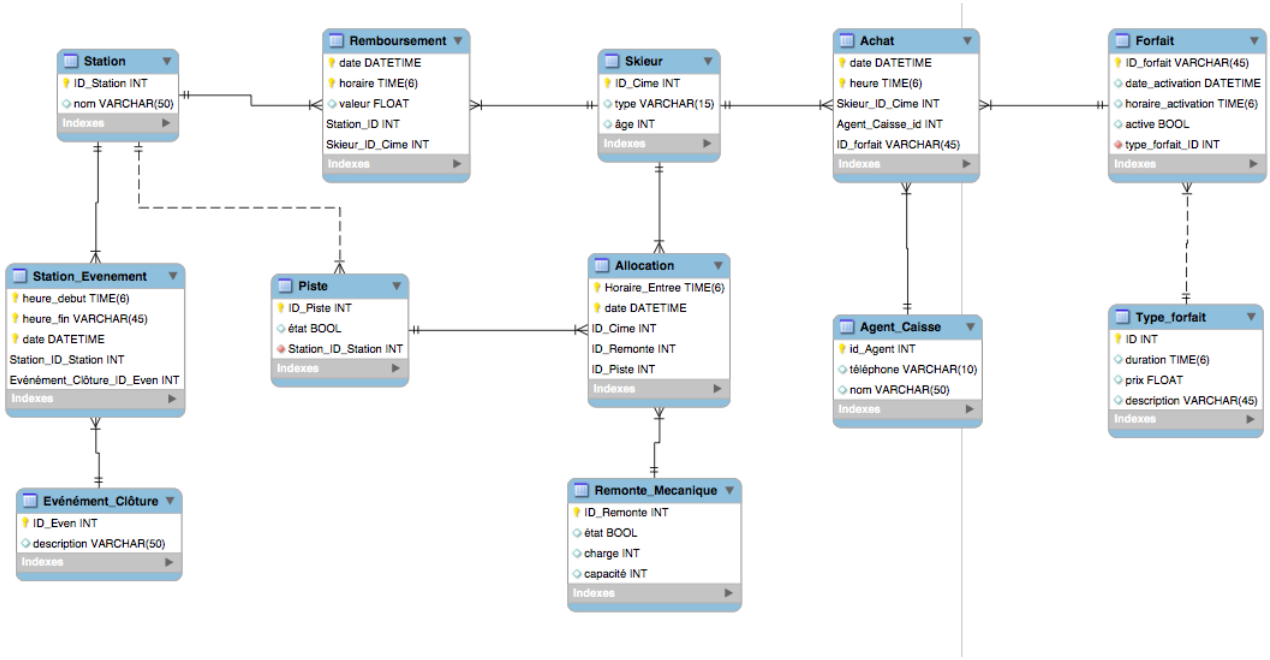

FIGURE 13 – Diagramme Relationnel

4 Vue Déploiement

4.1 Diagramme de déploiement

Du point de vue de déploiement, le projet a été conçu comme un site hébergé sur un serveur Web qui fournit des services aux agents de caisse et aux skieurs. Avec cette architecture client-serveur, les services seraient délimités par trois voies : dans la première voie, le skieur pourrait accéder via internet aux principales caractéristiques du système. Dans la seconde, l'agent de caisse (et le client indirectement) pourrait utiliser les services par le biais d'un ordinateur dans la station. Et enfin, le système serait d'accéder directement aux panneaux en temps réel sur la base des notifications des remontés. Notez que le serveur web est capable de distinguer les différents type d'utilisateurs afin de leur donner accès aux interfaces qui les concernent et les autorisations necessaires pour interagir avec la base de données.

L'accès aux données est géré par un serveur de base de données, ce serveur-ci recevra des messages (requêtes sql) effectuées par le serveur Web et les messages retournés seront les informations requises. La communication entre le serveur Web et le serveur de base de données est directe. Par ailleurs ils peuvent, être placés dans le même host. Ainsi, nous avons créé le diagramme de déploiement suivant :

5 Conclusion

Du point de vue fonctionnelle, au déploiement, nous avons pu ici, modéliser le problème à l'aide différents outils. Ce qui nous donne une idée claire sur la façon dont nous allons implémenter notre application, de manière à respecter au mieux les spécificités. Dans la prochaine itération, nous implémenterons en JavaEE un produit minimal viable.

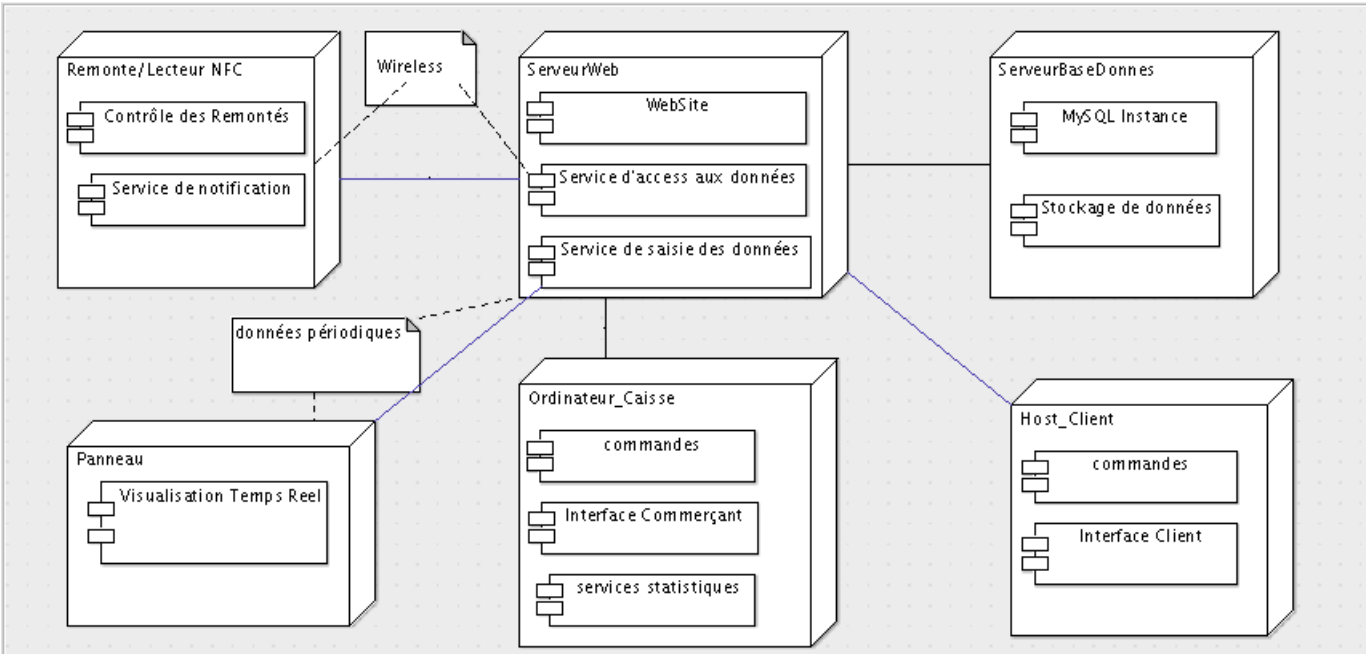

FIGURE 14 – Diagramme de déploiement

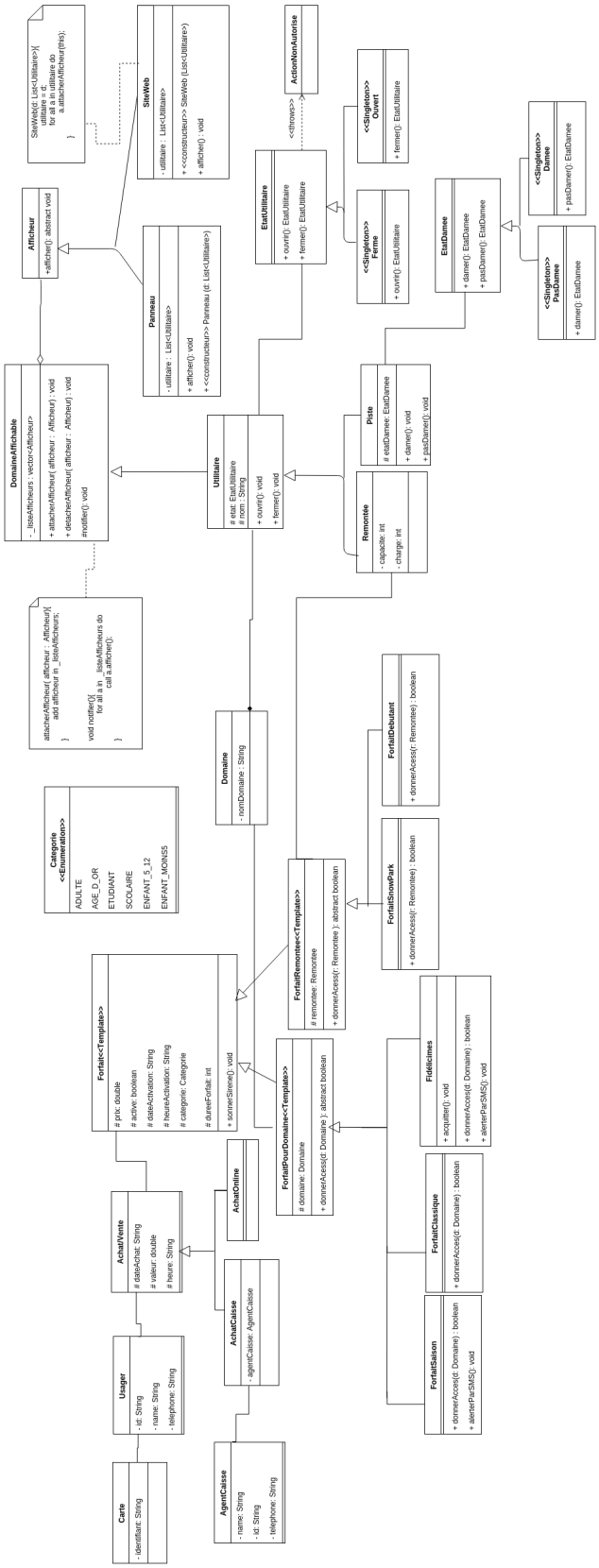

FIGURE 15 – Diagramme de classe global