Projet BackLine

Dungeon Crawler

- Rapport de projet tuteuré -

- Cédric CANESTRARI
- Ghislain DUGAT
- Florian FIOL
- Charbel FOUREL
- Florian MEI

Remerciements :

La licence Pro SIL NTI et toute son équipe pédagogique

Les autres rangées de la classe

La faculté de St Charles pour l'accès a ses locaux

Les copines/parents/frères/sœurs et autres qui auront supportés les longues journées de développement, de prises de tête, et les WE a la maison

Sommaire :

I – Présentation du projet :

1. Sujet de départ :

Comme convenu avec M Ostrowski, l'objectif était de réaliser un jeu de plateau ou un puzzle logique pour un environnement Android.

2. Sujet choisi :

Dans un but principalement pédagogique, il a été choisi de développer un jeu de plateau de type Dungeon crawler (exploration de donjon) multijoueur en tachant de respecter le modèle MVC et en s'interdisant l'utilisation d'un framework de développement de jeu vidéo.

3. Cahier des charges :

- ***Capture des besoins fonctionnels :***

  - L'application permettra à un joueur de participer à une aventure en lui laissant la possibilité de créer des parties et de retrouver ses compagnons en ligne.
  - L'application sera accompagnée d'une application serveur dédiée pour héberger les parties
  - un maximum de 4 joueurs pourront participer à une même aventure
  - Le joueur sauvegardera son héros dans une base de données centralisée, en même temps que ses paramètres d'identification.
  - Une IHM claire et intuitive sera développée pour l'application client afin que le joueur puisse efficacement gérer sa connexion, création de partie, et son activité en jeu, ainsi que pour l'application serveur dédié pour permettre une administration basique (consultation logs, démarrage, arrêt, redémarrage)
  - il sera nécessaire de développer un site internet pour permettre aux joueurs de créer un compte et gérer leur héros.
  - le donjon sera généré aléatoirement à chaque nouvelle partie.
  - les joueurs pourront : se déplacer, attaquer des ennemis, fouiller, se soigner eux même ou leurs alliés.
  - Les joueurs pourront : créer des lobbies (salon pour héberger les joueurs d'une partie)
  - Les joueurs pourront rejoindre des lobbies existants
  - les parties seront lancées lorsque tous les joueurs seront prêt dans le lobby
  - le serveur pourra simultanément héberger plusieurs lobbies et donc plusieurs parties
  - le héros sera sauvegardé sur un serveur central, aucune sauvegarde locale.
  - le joueur devra posséder un compte pour pouvoir jouer, ainsi qu'un héros valide
  - le donjon aura une difficulté croissante
  - le donjon aura une difficulté adaptée au niveau moyen des héros présents dans la partie
  - les joueurs pourront choisir une taille de donjon prédéfinie
  - les joueurs pourront choisir une taille de lobby prédéfinie entre 2 et 4 joueurs
  - un système de chat (discussions instantanée) sera mis en place au niveau

serveur pour que tous les joueurs connectés puissent dialoguer entre eux
- Un système de chat sera disponible dans les lobbies pour que les joueurs puissent discuter entre eux au niveau du lobby
- Un système de chat sera mis en place pour permettre aux joueurs de dialoguer au sein d'une partie engagée.
- Un joueur ne possède qu'un héros (la possibilité au joueur de posséder plusieurs héros, bien que réalisable, ne sera pas totalement développée pour le moment pour une question de temps)
- Pour une question d'économie de ressources, les éléments relatifs au jeu (objets, équipement, butin, consommables) auront un modèle stocké en local dans un fichier externe mais importé uniquement au démarrage du jeu et stocké sous forme abstraite dans un modèle interne au programme. Il en sera de même pour le héros, stocké dans la base de donnée centralisée, mais chargé au démarrage dans un modèle abstrait au sein du programme pour limiter les requêtes.

- ***Détails de l'application/Game Design :***

L'objectif de ce projet sera donc de réaliser un jeu vidéo de type jeu de plateau d'exploration de donjon/grotte en équipe, ou il faudra trouver la sortie du donjon en emmenant un maximum de butin et bien sur en ayant survécu.

Les joueurs pourront expérimenter des combats contre des monstres gérés par l'ordinateur, et fouiller des coffres et autres objets pour récupérer du butin.
Le poids transporté étant limité les joueurs devront rechercher les butins les plus rares car ils ont le meilleur rapport valeur/poids.

Le donjon sera généré aléatoirement à chaque nouvelle partie à l'exception de la case départ qui sera toujours placée en bas du plateau mais sur un axe de X variable, et la case d'arrivée qui sera toujours placée en haut mais aussi sur un axe X variable.

Les joueurs agiront tour à tour, l'un après l'autre selon une séquence toujours identique :

a) on se déplace
b) on agit (attaquer, fouiller, soigner soi-même ou un allié ; une seule possibilité par tour)

La partie arrivera à sa fin quand tous les héros encore en vie auront rejoint la salle de sortie du donjon. A ce moment-la, le joueur peut convertir les butins et équipements qu'ils possèdent en pièce d'or qui représentent son score de la partie.

Pour pouvoir progresser dans ses aventures, le joueur pourra s'équiper d'armure et armes variées, il aura la possibilité de stocker des consommables comme des potions de vie. Il pourra aussi gagner de l'expérience et choisir des professions plus spécifiques que simple aventurier comme mage, scout ou encore templier.

Chacune de ces classes aura un rôle précis dans le triangle des classes classique des MMO classiques (jeux multijoueurs en ligne), nous entendons par là, la classe du Tank (classe qui attire les coups et les encaisse efficacement),

du Soigneur (classe qui soutient les autres joueurs principalement en les soignant) et enfin du Damage Dealer (littéralement distributeur de dégât, classe qui se charge d'infliger des dégâts importants aux ennemis).

Comme à l'accoutumé dans ce genre de jeu, le héros possédera un ensemble de caractéristiques qu'il pourra faire évoluer au fur et à mesure qu'il passera des niveaux d'expérience. Ces caractéristiques pourront profiter de bonus du au choix de classe avancée (comme mage qui améliorera l'intelligence) et le héros profitera bien évidement de l'ensemble de ces caractéristiques et bonus dans le cadre de ses aventures.

II – Développement:

1. Méthodes de gestion de l'équipe :

La Gestion de l'équipe s'est faite en deux temps :
Dans un Premier temps, du fait du confort du tableau noir pour la phase de conceptualisation, nous avons régulièrement eu des réunions physiques sur le site de l'université dans la salle de cours. Ces réunions étaient complétées par des vidéo-conférences régulières sur Skype.

Un partage d'écran était toujours actif lors de ces conversations sur le poste d'un des développeurs, tandis que deux autres au minimum assistaient le développeur en partage pendant son codage, lui donnant un recul supplémentaire pendant l'élaboration des premières classes. Les rôles changeaient, afin que ce ne soit pas toujours le même qui code.

Dans un deuxième temps, une fois la phase de conceptualisation terminée, nous avons mis fin aux réunions physiques pour favoriser les vidéo-conférences, cela avait pour principal intérêt de réduire les temps de transport importants pour certains, nuisant a la productivité.

Ces vidéo-conférences commençaient en général vers 14h pour se finir vers 18h. Durant ces réunions les différentes équipes travaillaient sur leurs modules tout en étant disponibles pour les autres en cas de question relatives à l'intégration des modules entre eux. Il est évident qu'entre les réunions, chacun tachait d'avancer de son coté sur son module, c'est pour cela qu'à chaque début de réunion un point était fait sur l'avancée de chacun pour garder au mieux la synchronisation entre les modules du projet. Le rythme de ces réunions était de deux jours de réunions pour un jour de relâche et de travail personnel.

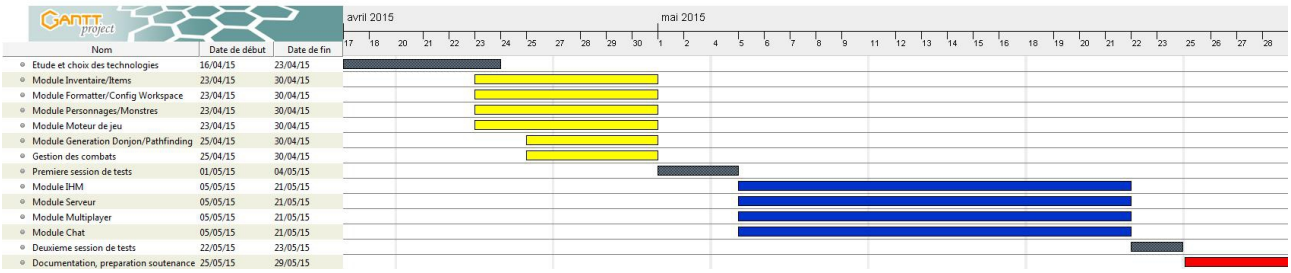


Illustration 1: Diagramme de Gantt Prévisionnel

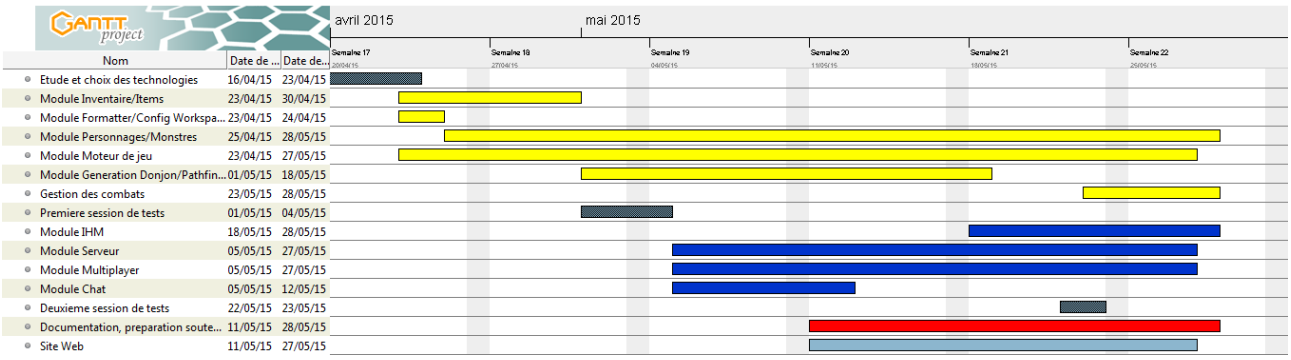


Illustration 2: Diagramme final

2. Méthode de conceptualisation et de gestion de projet :

Nous avons taché de nous inspirer de la méthode de conception 2TUP pour ce qui concerne la conceptualisation de l'application. Concernant la méthode de conception, nous avons adopté une méthode que nous avons appelé « modulaire ». Cette méthode sous entendait le découpage de l'application en différents modules, et la distribution du développement de ces modules entre les différents membres de l'équipe. Cela a permis de mettre en place un développement en parallèle avec plusieurs modules sur lesquels nous pouvions nous consacrer quasiment en simultanée, permettant à chacun d'avoir une idée assez précise du progrès général de la conception.

Distribution des rôles :

– Cédric CANESTRARI : Lead Serveur | Assistant IHM Java, intégration.
– Ghislain DUGAT : Lead Site internet | Assistant personnage.
– Charbel FOUREL : Lead Génération de donjon, projet | Assistant IHM Android, personnage.
– Florian MEI : Lead IHM Android | Assistant serveur
– Florian FIOL : Lead IHM Java, Personnage, Intégration.

3. Technologies choisies :

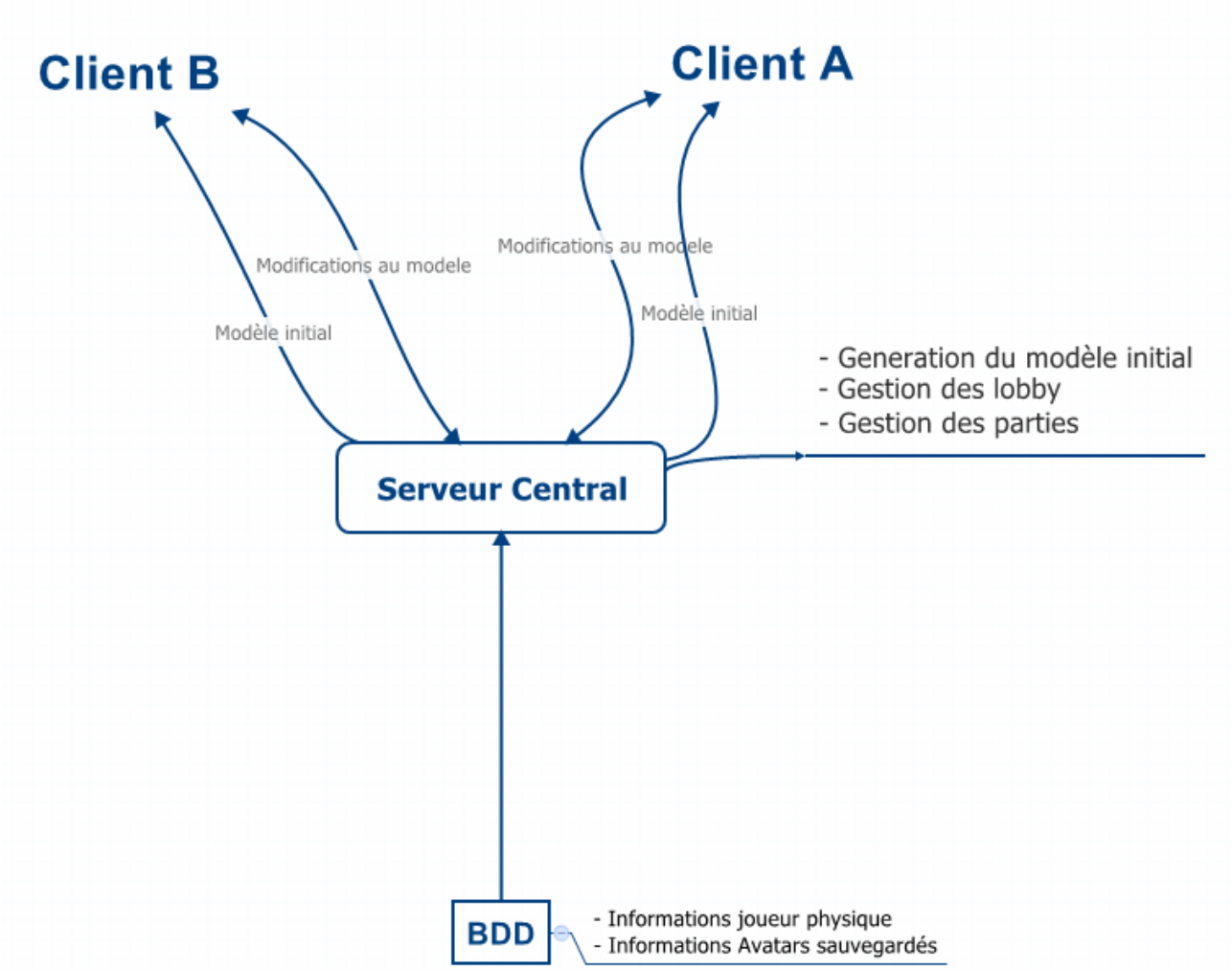
Nous nous permettons de rappeler que le principal objectif était de réaliser ce projet sans utiliser de framework de développement de jeux vidéos, voilà pourquoi dans ce qui va suivre nous ne trouverons principalement que des design patterns applicables au java et que nous avons estimés pertinents pour répondre à nos besoins :

– Langage Orienté Objet Java
– Design pattern Observer
– Design pattern Composite
– Design pattern Mediator
– Design pattern Prototype
– Modèle Vue Contrôleur
– Algorithme de recherche de chemin A*
– Sockets
– MySQL

- Connexion SSL
- SWING
- Framework de développement Android (il était incontournable pour l'IHM)

L'ensemble de ces technologies sera détaillé au sein des différents chapitres qui vont suivre, consacrés au développement propre de chaque module.

4. Plan de développement de l'application, idée directrice :



Vu qu'il nous a été nécessaire de maîtriser un certains nombre de technologies pendant le développement, un premier plan d'application a été mis en place, tout en tachant de le garder le plus simple possible. Le principe était d'avoir un serveur hébergeant les parties et générant un modèle de donjon et de partie pour les joueurs, modèle transmis aux joueurs par le serveur et mettant à jour le modèle local du donjon chez le client. Les joueurs modifient leur modèle client qui transmets les éléments modifié au serveur qui met a jour son modèle.Les échange entre le serveur et le client seraient des chaînes de caractères enfermées dans des messages. C'est en gardant en permanence cette idée directrice que nous avons avancé dans l'élaboration du projet, tout en nous permettant une interface simple entre nos différents modèles locaux communiquant avec des chaînes de caractères du fait du modèle vue contrôleur.

5. Etapes de conception :

Dans l'objectif de ne pas s'éloigner de la livraison d'une application la plus achevée possible, nous avons décider de quatre niveaux de réalisation :

- Premier niveau : Création d'un modèle d'application complet et fonctionnel avec un inventaire, des caractéristiques pour les personnages, un générateur aléatoire de donjon, et la gestion des déplacements dans le donjon.

- Deuxième niveau : Ajout des monstres et de la gestion des combats, ajout du mode multijoueurs, ajout de la possibilité de modifier les caractéristiques en jeu.

- Troisième niveau : Création d'une IHM tant pour l'application dédiée serveur, que pour l'application client en Java SWING (ce niveau est nécessaire en soit pour la mise en prototype et pour la mise en place d'une série de test de messages entres les différentes couches des modèles)

- Quatrième niveau : Implémentation du système de caractéristiques avec la mécanique de gain d'expérience. Implémentation de la mécanique de victoire du donjon. Portage de l'IHM sur Android

Nous sommes parvenus au niveau trois concernant l'IHM En java et SWING qui nous a servit de prototype, mais avons rencontrés de nombreuses problématiques au moment de son portage sur Android, nous retardant et nous empêchant d'atteindre le niveau 4.

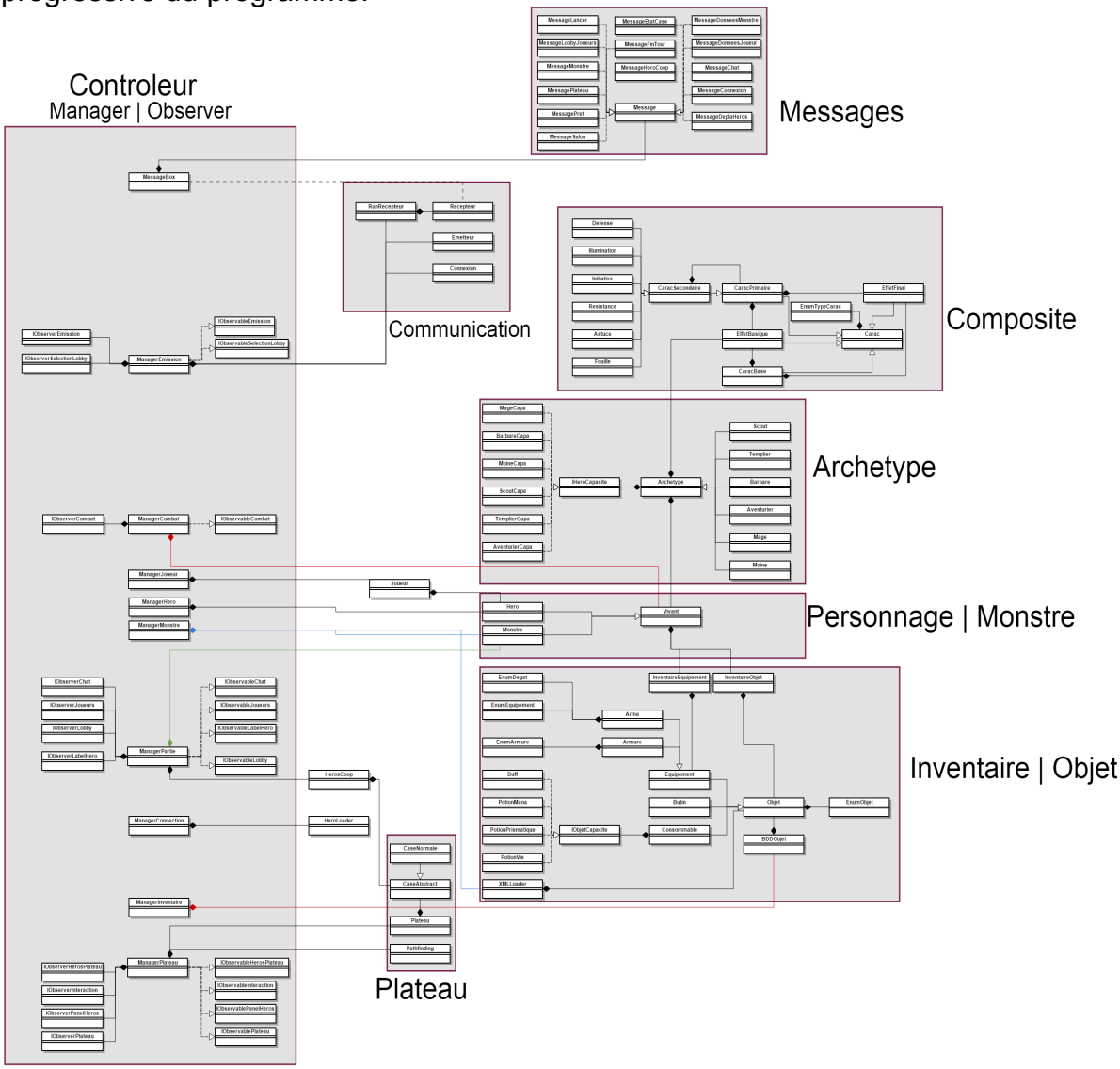
Ces problématiques seront développées en conclusion de ce rapport. A noter que nous avons taché de développer cette application tout en gardant une volonté de faire un programme facile a entretenir et a faire évoluer.

Ainsi tel quel il sera possible d'intégrer de nombreuses autres fonctionnalité que le temps manquant nous a empêché d'implémenter. Nous avions pensé entre autre a des cases pièges ou encore a vitesse de déplacement variable, une IA complète pour les monstres, une implémentation complets dans le pathfinder de la possibilité de donner un poids a chaque nœud d'un chemin pour permettre une gestion plus pointue de la distance maximum de déplacement... Ces éléments et d'autres sont prévues dans une future implémentation si le temps et la passion nous le permet.

III – Modules:

1. Introduction :

La méthode de conception adoptée, nous a orienté vers une conception modulaire et progressive du programme.

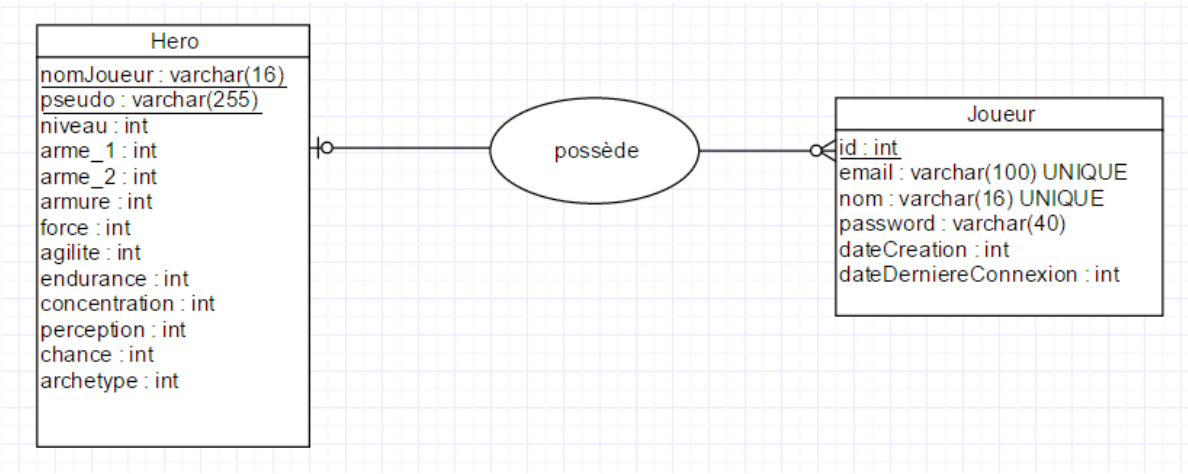


Encore une fois l'objectif était de garantir à la fois souplesse et facilité d'entretien du programme comme le présente l'image précédente où l'on a matérialisé les différents modules et leur placement et relations dans un résumé UML. Un aperçu plus grand de cet UML est disponible en **annexe 1** de ce rapport. L'IHM étant encore à l'état de prototype, elle n'apparaît pas sur ce diagramme.
Un UML du programme avec les détails des classes du prototype IHM est disponible en **annexe 2**

2. Module Base de données :

Une base de donnée simple a été mise en place pour répondre au besoin de

stocker a la fois les informations de login des utilisateurs et certaines informations relatives a des éléments de jeu comme le statut du héros après la dernière sauvegarde, ou encore son équipement. Bien que pour le moment un joueur ne possède qu'un seul héros, le format de la base de données telle que conçue devrait permettre le passage a plusieurs héros sauvegardés par joueurs sans trop de contraintes.



Du fait de la taille réduite de ce module il ne fait pas l'objet d'un rapport indépendant.
Le choix de la technologie s'est porté sur MySQL pour le moteur et PHPmyAdmin pour l'interface d'administration.

3. Rapports indépendants de chaque module :

Les modules étant mal grès tout assez nombreux nous avons choisi de ne présenter que les plus pertinents. Ils seront développés en détails par une partie de ses auteurs dans plusieurs rapports indépendants disponibles en **annexe 3** dans l'ordre qui suit :

- Rapport Serveur Client
- Rapport Site internet
- Rapport Composite / Caractéristiques
- Rapport Inventaire
- Rapport IHM Java SWING
- Rapport Génération de donjon
- Rapport Personnages

IV – Conclusions:

1. Retour sur la méthode de développement :

Nous avons put expérimenter certains avantages et certains inconvénient propres a nos choix en matière de méthode de travail et de gestion de projet :

Le travail collectif avec partage d'écran a permis un recul instantané sur ce qui était en cours de développement, un confort non négligeable vu les temps de production disponibles au regard de l'aspect ambitieux du projet. En contre partie certains modules ont étaient développés par petit groupes indépendants, ce qui a mis certaines personnes un peu en dehors des détails de développement concernant certains modules sur lesquels elles n'étaient pas intervenues car concentrées sur une autre branche du projet.
Le 2TUP nous a servi au début de la conceptualisation mais aussi durant la réalisation moins comme une méthode de développement en tant que tel mais plutôt comme un des principes directeur sur lequel nous devions nous appuyer des qu'il fallait ressoude une problématique structurelle.

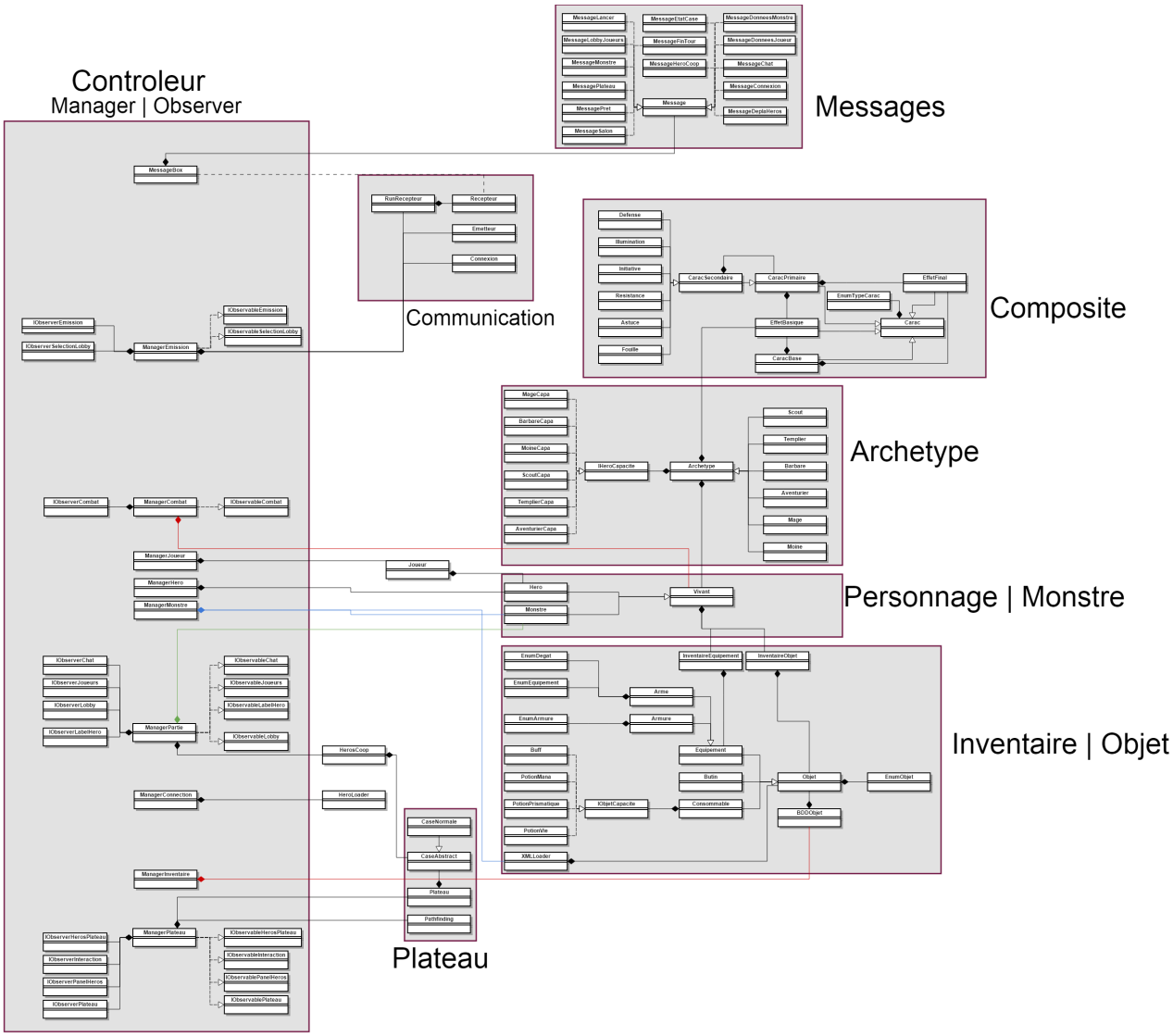
2. Problématique non résolu :

Au moment ou ce rapport est rédigé dans sa forme finale, nous n'avons pas été en mesure de fournir un prototype d'IHM Android fonctionnel et ceux du aux nombreux problèmes rencontrés durant le travail de portage de l'interface Java SWING.

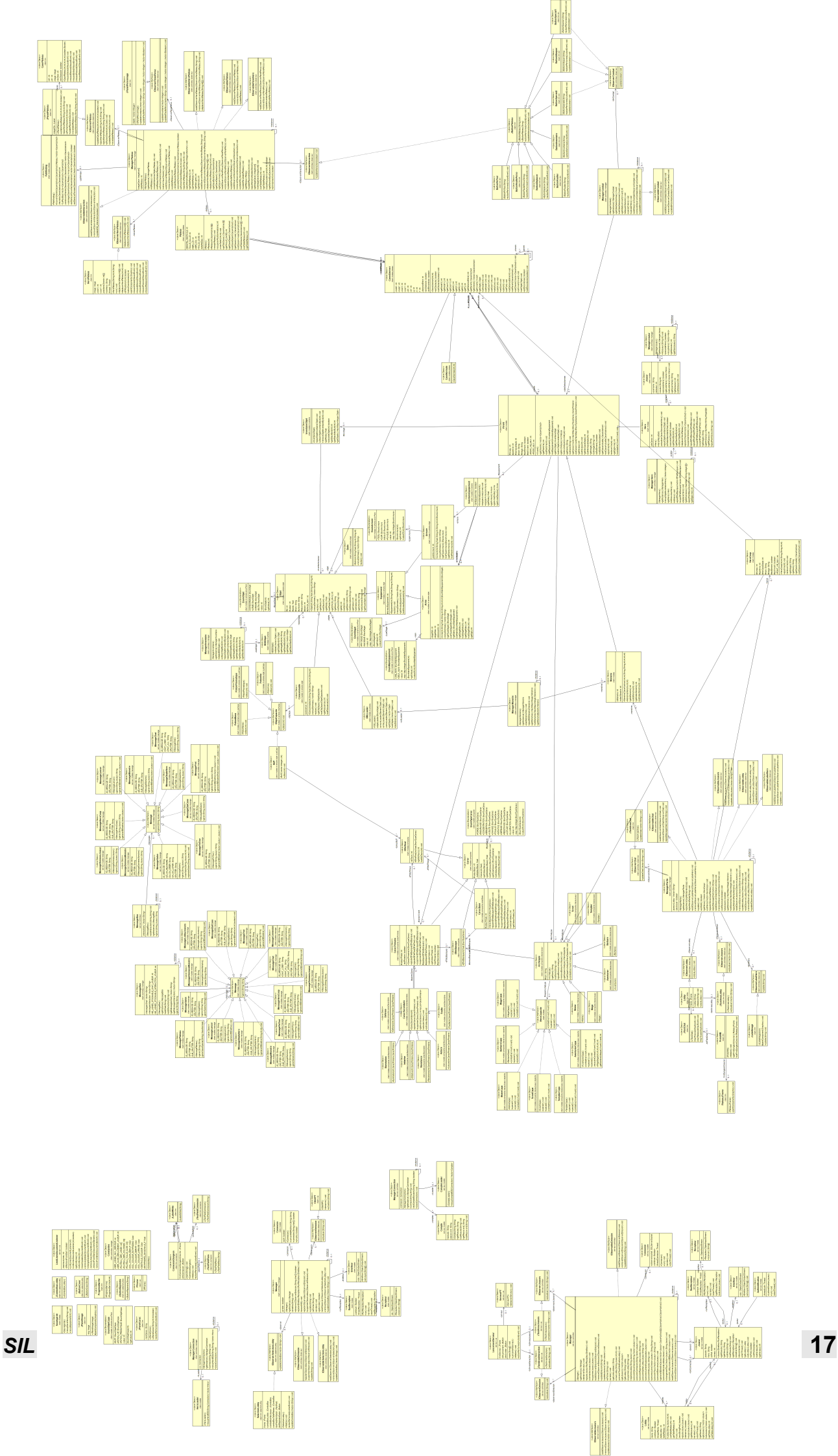
Voici un retour sur les différentes étapes de réflexion et certaines solutions trouvées :
- Tout d'abord les objets d'interface Android ne sont pas exactement les même que en SWING, il a fallut réécrire certains de ces objets pour les rendre compatibles avec l'environnement Android. On pourra citer les Jbutton traduits en Button, les TextField traduits en EditText.
- La connexion à la source de donnée se fait de manière singulière sur Android puisque apparemment il est chaudement conseillé de ne pas se connecter à une base de données directement via un terminal Android mais plutôt de passer par des scripts PHP hébergés sur le serveur central qui génèrent des fichiers JSON à la demande contenant les données nécessaires aux traitements. L'application récupère ces fichiers JSON au lieu des résultats de requêtes sous leur forme brut.
- Les XML doivent être stockés dans le dossier des Assets.
- Les clefs SSL sont stockées coté client dans le dossier Assets, et se doivent d'être au format BKS (JKS non compatible)..

Nous sommes restés bloqués à l'étape de connexion au serveur pour l'accès au gestionnaire des lobbies, la dernière piste que nous parcourions était la possibilité d'un autre problème relatif a la reconnaissance de la clef SLL.

ANNEXE 1


Controleur
Manager | Observer
Messages
Communication
Composite
Archetype
Personnage | Monstre
Inventaire | Objet
Plateau

ANNEXE 2



ANNEXE 3