Université ou Entreprise

Nom du Projet
Sujet du Projet

Auteur :
Premier Auteur
Deuxième Auteur
Troisième Auteur
Quatrième Auteur

Client :
Prénom Nom
Référent :
Prénom Nom

4 avril 2016

Résumé

Ce projet porte sur l'analyse de parties d'un jeu de cartes appelé Dominion sous formes de logs. L'objectif principal est d'obtenir des données utilisables facilement afin de pouvoir effectuer des analyses sur celles-ci. Pour cela une première partie du projet porte sur la lecture de ces logs puis dans un second temps le stockage des données extraites. La seconde partie du programme porte sur une bibliothèque capable de récupérer ces données , de les analyser et de les modifier.

Table des matières

Chapitre 1

Présentation du projet

Le jeu de cartes *Dominion* a été mis à disposition sur un serveur de jeu d'octobre 2010 à mars 2013. Les parties effectuées via ce serveur ont été enregistrées dans des *logs* qui mémorisaient toutes les actions des joueurs ; ces *logs* ont été mis à disposition du public. Mais ces enregistrements n'ont pas conservé un certain nombre d'informations ayant un rapport avec les décisions prises par les joueurs.
Un wiki relatif au jeu a été élaboré, et propose des avis d'experts comme aide à la décision pour les joueurs : il propose des conseils, notamment au niveau stratégique. Le but de ce projet, proposé par Yvan Le Borgne, chercheur au Labri, est de traiter ces *logs* (soit plus de 12 millions de parties) afin de répondre à plusieurs interrogations. Il s'agira principalement de comparer Les données recueillies aux préconisations de ce wiki,afin de valider, à travers les contenus des parties, l'éventuelle efficacité des avis et suggestions fournies par les experts. A tout le moins, on cherchera à élaborer des outils de validation présentant une efficacité suffisante.
A cet effet, le projet a pour but de créer une base de données recensant les parties puis un travail d'analyse sera effectuée à partir de celle-ci. Pour cela notre programme devra tout d'abord extraire les données présentes dans les logs, car ceux-ci sont trop volumineux et trop compliqués à utiliser directement (structure non standardisée). Il s'agira de mettre au point des structures de données permettant de stocker les données de manière plus efficace. Une manière simple de représenter les données extraites et analysées sera proposée à l'utilisateur. Par ailleurs, il manque des informations dans les enregistrements (*logs*). On cherchera à trouver par quelle méthode on peut les reconstituer et de les mettre dans un format contenant toutes les informations. En outre, on cherchera à optimiser les opérations menées dans l'analyse des donnnées, en recherchant le meilleur équilibre possible entre la mémorisation des recherches et le re-calcul. Enfin, on esssaiera de déterminer quelle stratégie a été utilisée dans chaque partie. Voire de faire émerger les changements de stratégie en cours de partie. Si on prend par exemple une des stratégies (la *Penultimate Province Rule*) peut on détecter à quel moment cette règle a été appliquée ou contournée ? ou qui a mis au point cette stratégie ? (découverte qui pourrait permettre d'organiser un classement des joueurs les plus performants) (une sorte de ELO demandé par le client). Dans la mesure où on parviendrait à mettre au point un nombre suffisant de ces démarches stratégiques, il s'agirait de déterminer le processus habituel d'apprentissage des joueurs.

1.1 Règles du jeu

Chaque joueur possède un deck de cartes et a accès à un «marché» ou différentes cartes d'action sont disponible. Il y a également une pile de cartes de victoires.
Les joueurs commencent avec un deck contenant uniquement des cartes de monnaie permettant d'acheter les autres cartes mises à la disposition des joueurs. Les joueurs commencent leur tour avec 5 cartes en main et le tour d'un joueur se déroule en deux phases, premièrement la phase d'action, le joueur peut jouer une carte d'action, puis une phase d'achat ou le joueur peut acheter des cartes du marché, des cartes de monnaie ou des cartes de victoires.
La partie se termine dans deux cas, si la réserve de cartes «Province» (carte de victoire au score le plus élevé) est vide ou bien si 3 piles du marché sont vides. Quand la partie est terminée, les points de victoires (découlant des cartes de victoire) de chaque deck sont comptés et le vainqueur est celui qui à la meilleur score.

1.2 Introduction au datamining

1.3 Sujet

Bla(cf. fig. 1.1)

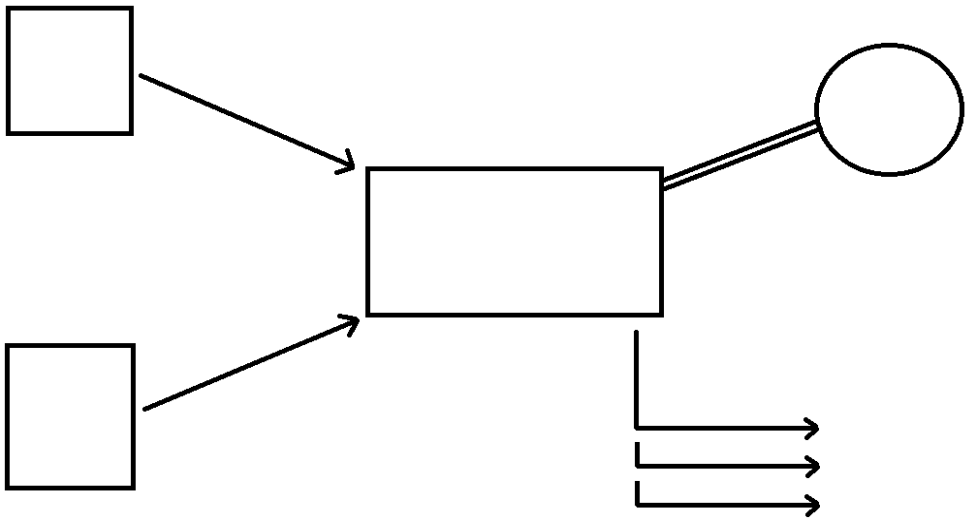


FIGURE 1.1 – Schéma descriptif

Bla
Bla

Bla

0. Note bas de page "intro"

1.4 Problématique soulevée

Bla

Problématique du sujet

1.5 Hypothèse de solution

Bla

Voici une liste :
— item 1 ;
— item 2 ;
— item 3 ;
— item 4.
Bla

Bla
Bla [1]

Bla(cf. ref. [**?**]).

1. Note bas de page "bla"

Chapitre 2

Analyse de l'existant

2.1 Analyse de l'existant

goko-dominion-tools est le principal outil disponible pour exploiter les logs, mais cette version ne fonctionne plus car les serveurs goko ont été fermés. Les fonctionnalités offertes par cet outil sont très proches de ce que nous voulons proposer, notamment un parsing de logs, une recherche dans ces logs et un leaderboard. Le programme est écrit en python et est disponible sur github.

2.2 Bilan récapitulatif

Voici un tableau (cf. fig. 2.1) récapitulatif de notre analyse de l'existant...

Solution	Critère 1	Critère 2	Critère 3	Critère 4
Solution 1(cf. ref. [**?**])	Oui	Oui	Oui	Oui
Solution 2(cf. ref. [**?**])	Oui	Oui	Oui	Non
Solution 3(cf. ref. [**?**])	Oui (sauf telle chose)	Non	Non	Oui
Solution 4(cf. ref. [**?**])	Oui	Non	Oui	Non
Solution 5(cf. ref. [**?**])	Oui (uniquement ceux-ci)	Non	Oui	Non

FIGURE 2.1 – Tableau récapitulatif des solutions

Chapitre 3

Analyse des besoins

Intro

3.1 Besoins fonctionnels

3.1.1 Modélisation des données

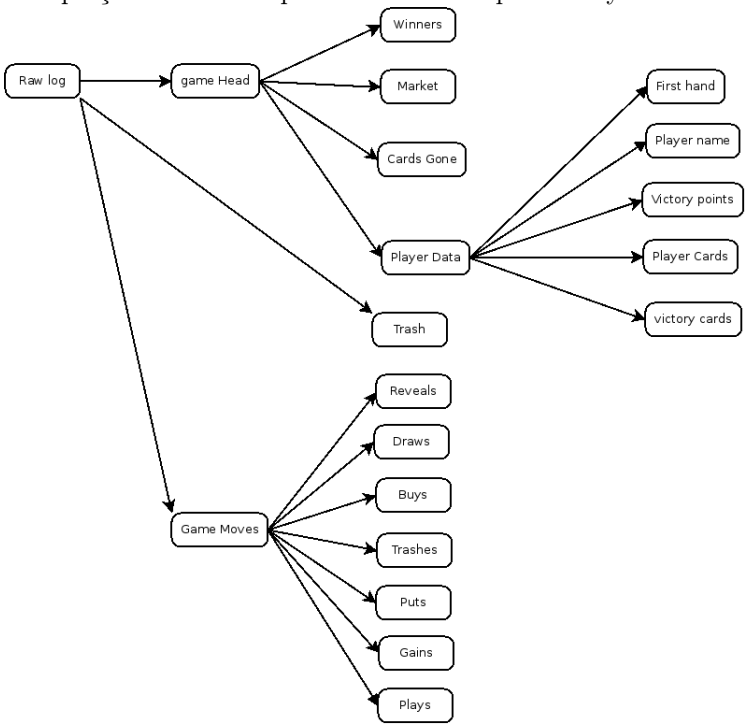
3.1.1.1 Décompression des logs

Niveau de priorité = haute

— Les fichiers tar.bz2 seront stockés dans un dossier.
— Le programme va décompresser un fichier spécifié à partir de ce dossier dans un dossier temporaire.
— Le programme va supprimer les logs décompressés à la demande du parser.

3.1.1.2 Parser

Niveau de priorité = haute
L'analyseur va prendre les logs sous format *HTML* et va identifier les mots clés présents sur ces logs.
L'analyseur est chargé de lire les logs et collecter l'information importante sans perdre la structure du log.
Un aperçu des données pour être reconnu par l'analyseur est illustré dans ce diagramme :



— Winners : liste des gagnants du jeu, il peut y avoir plusieurs gagnants en cas d'égalité.
— Market : liste des 10 cartes disponibles pour être acheté par les joueurs.
— Cards Gone : liste des cartes qui ont été entièrement acquises à la fin de la partie.
— Player Name : Nom du joueur.

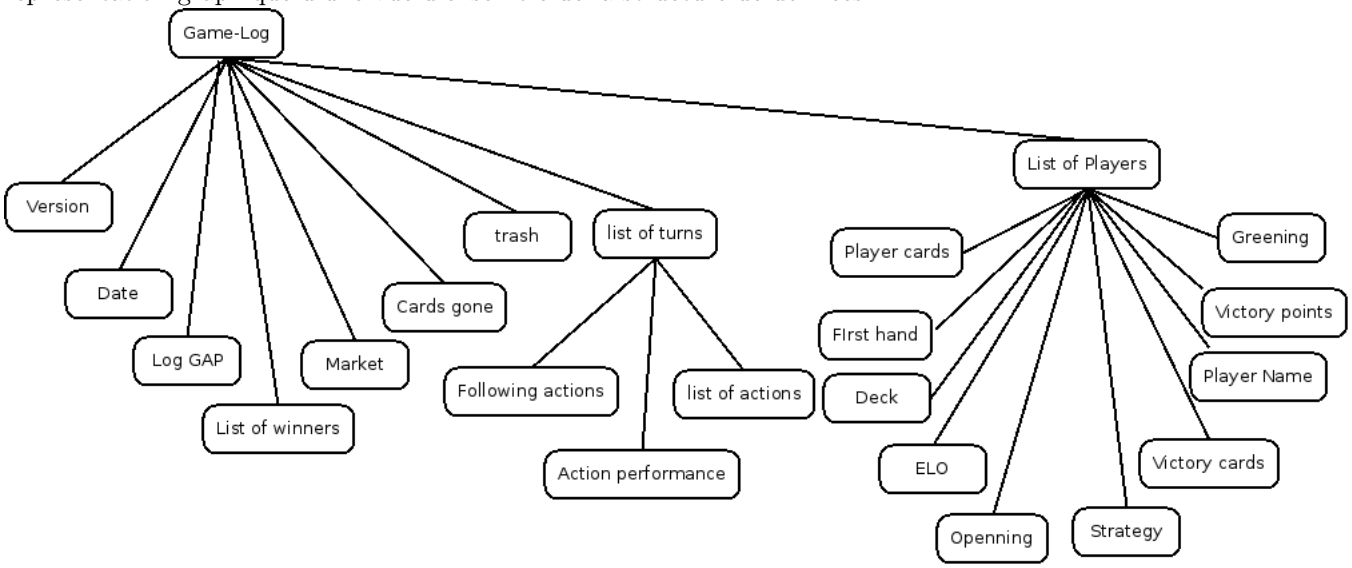
— Victory points : nombre de points à la fin de la partie.
— Player cards : liste des cartes obtenues à la fin de jeu avec des noms et de quantités differents.
— Victory cards : liste des cartes de victoire le joueur avec le acheté.
— Trash : liste des cartes qui sont jetés à la fin de la partie.
— Game Moves : liste des actions effectuées pendant chaque tour d'un joueur.

Le parser doit lire chaque élément montré dans le graphique précédent en tant que partie des données lorsque l'utilisateur exécute le programme de parsing

3.1.1.3 Créer game-log

Niveau de priorité = haute

Cette fonctionnalité est la création d'une structure de données où les données du log seront écrites. Voici une représentation graphique d'une vue d'ensemble de la structure de données :



3.1.1.4 Créer une base de données orientée document

Niveau de priorité = haute

Ce module sera responsable pour créer la base de données orientée document, contenant des données au format JSON.

3.1.1.5 Se connecter à la base de données orientée document

Niveau de priorité = haute

— La base de données orientée document sera en charge des échanges à travers un socket spécifique.
— La base de données orientée document recevra les logs de jeu au format JSON.
— La base de données orientée document recevra les demandes du programme relatif à chaque élément de données qu'il contient.
— La base de données orientée document enverra les résultats des demandes effectuées par le programme.

3.1.1.6 Compression

Niveau de priorité = moyen

La base de données orientée document contiendra la plupart des données et un certain niveau de compression devra être appliquée. La base de données que nous allons utiliser est focalisé sur mongodb, et il offre 2 niveaux de compression *Zlib* et *Snappy*. Au cours du développement *Snappy* est utilisé car il offre une meilleure performance. Mais le programme final devrait donner à l'utilisateur la possibilité de choisir le niveau de compression lors du démarrage de l'analyse du log.

3.1.1.7 Sauvegarder game-log

Niveau de priorité = haute

— La base de données orientée document permet de convertir la structure log de jeu dans un document au format JSON.
— Les données fournies au format JSON seront stockées dans la base de données.

3.1.2 Analyse des données

3.1.2.1 Recueillir des données

Niveau de priorité = haute

— L'analyseur de données va convertir les résultats de la requête dans un format utilisable par le module de traçage.
— Si les données sont simples à lire (par exemple : un numéro, un nom, une phrase), l'analyseur de données sera tout simplement envoyer le résultat au format texte.
— La dernière requête sera mémorisée avec son résultat afin de gagner du temps si la prochaine requête faite par l'utilisateur est le même.
— Si l'utilisateur souhaite appliquer un script python de logs de jeu, le programme demandera les logs de jeu et les mettre dans un fichier texte, ce fichier contiendra chaque action faite par chaque joueur et le script sera appliqué à ce fichier.

3.1.2.2 restorer game-log

Niveau de priorité = moyen

Cette fonction est responsable pour restaurer l'information manquante sur les logs analysés, en utilisant la déduction basique sur la base des données obtenues par le log.
En cas d'un en-tête du jeu manquant ou partie manquante :
— Gagnants / cartes de victoire / points de victoire manquant : le programme peut garder la trace des cartes détenues par les joueurs pendant le jeu afin de savoir combien de cartes victoire qu'ils ont à la fin de la partie.
— Market manquant : Le programme peut garder la trace des cartes achetées au cours du jeu afin de reconstituer partiellement ou totalement les cartes disponibles sur Market.
— Cartes Gone : Comme avec les données manquantes de Market, le programme permet de garder une trace de cartes achetées et compter la quantité achetée pour chaque carte, si le montant maximum de cartes est acheté, la carte a disparu à la fin de la partie.
— Nom du joueur manquant : le jeu se déplace partie d'un log assure également le suivi des noms des joueurs, le programme sera tout simplement de les restaurer dans l'en-tête.
— Cartes de joueur : le programme permet de garder une trace des actions effectuées au cours du jeu par rapport aux cartes de joueurs et les ajouter dans cette liste.

3.1.2.3 Plotting

Niveau de priorité = haute

Le module de traçage va obtenir les données converties obtenues après le processus de recueillir des données et de les tracer dans une interface graphique. Ce module va fonctionner d'une manière similaire de *GnuPlot*.

3.1.2.4 Calculer l'*ELO*

3.1.2.5 Plotting

Niveau de priorité = haute

L'analyseur devra calculer la finale ELO pour chaque joueur et ce qui était un joueur ELO sur un jeu donné. Pour chaque jeu dans l'ordre chronologique, ce module va demander au elo initiale des joueurs de la base de données relationnelle et les gagnants du jeu, puis il va calculer la nouvelle elo de chaque joueur de l'issue du jeu. Le module sera ensuite définir le elo calculés dans la base de données orientée documents au jeu en cours d'analyse et mettra également à jour l'elo globale pour chacun des joueurs qui participent au jeu sur la base de données relationnelle.

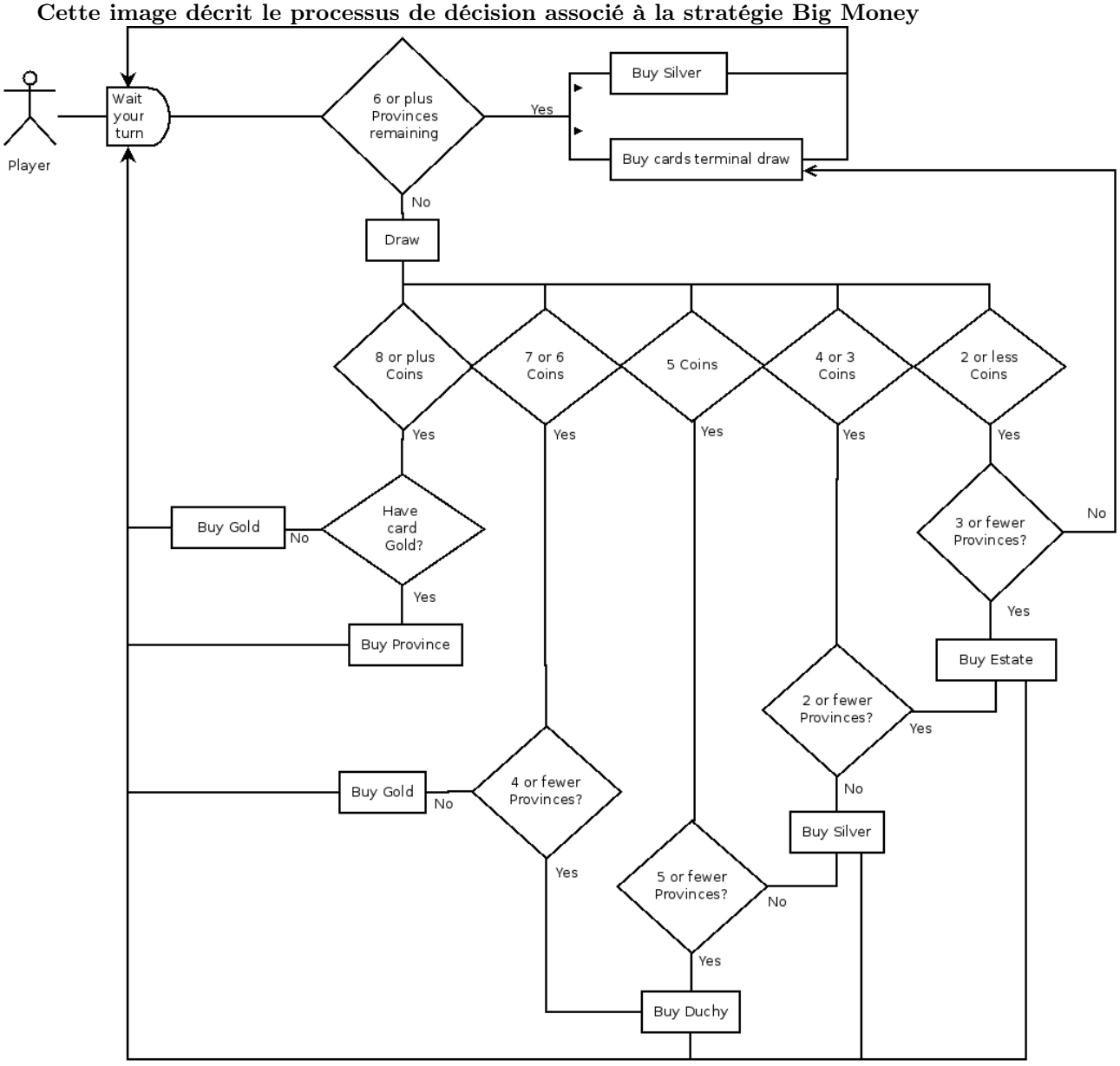
Plus d'informations sur ELO peuvent être consultés sur `https://en.wikipedia.org/wiki/Elo_rating_system`

3.1.2.6 Reconnaître les stratégies

Niveau de priorité = moyen

Le wiki de dominion décrit quelques stratégies qui peuvent être utilisés dans le jeu.
Par exemple :
Big Money
Beyond Silver
Penultimate Province Rule
Pour plus de détails sur `http://wiki.dominionstrategy.com/index.php/Strategy`

Cette image décrit le processus de décision associé à la stratégie Big Money



Et l'analyseur devra être en mesure de reconnaître quelle stratégie a été utilisée sur un match donné (si une stratégie a été utilisée) et générer des statistiques.
Plus de détails `http://wiki.dominionstrategy.com/index.php/Big_Money`

Afin de reconnaître la stratégie inventé Beyond Silver, l'analyseur doit reconnaître lorsque certains types de cartes sont achetés, la liste de ces cartes peuvent être trouvés sur : `http://wiki.dominionstrategy.com/index.php/Silver#Beyond_Silver`

Afin de reconnaître que Penultimate Province Rule est respecté (comme expliqué à `http://dominionstrategy.com/2011/03/28/the-penultimate-province-rule/`), l'analyseur doit garder une trace de la quantité de Points de Victoire de chaque joueur à chaque tour.

3.1.2.7 Reconnaître le Greening

Niveau de priorité = haute

L'analyseur doit être capable de reconnaître le moment de ***greening*** sur chaque match. En savoir plus à propos de ***greening*** sur `http://wiki.dominionstrategy.com/index.php/Greening`.
Le programme va reconnaître quand greening arrive en détectant lorsque les cartes de victoire commencent à être achetées.

3.2 Besoins non-fonctionnels

3.2.1 Besoins de performance

Aucun besoin de performance spécifique n'a été faite par le client. Mais pour l'analyseur le temps d'exécution doit être inférieur à une heure.
L'utilisateur doit être capable de parcourir le processus d'analyse sur un ensemble de données précis sans devoir attendre que la même requête à refaire.

3.2.2 Fiabilité

Le client demande que le nombre maximum de log doivent être analysé, que les données fournies ont des incohérences et quelques logs ne seront pas possibles d'analyser.
L'analyseur devra analyser 100% des données présentes sur le log et restaure toute information manquante si possible.

3.2.3 Attributs de qualité de logiciels

Même si seulement une ligne de commande est prévue pour le programme final, elle devrait avoir une courbe facile à apprendre et être facile à utiliser.
Les données générées par l'analyseur n'ont pas besoin à être lisible par l'homme.
Les game log générés devront être analysés par l'utilisateur afin de permettre la plus grande souplesse possible dans le traitement des logs.

3.3 Besoins organisationnels

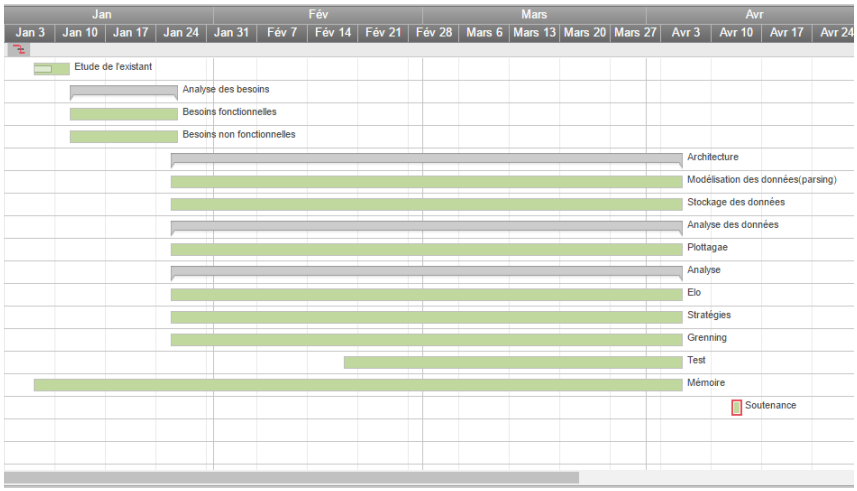
La répartition des tâches du projet et l'estimation de la durée de chacune d'elle sont présentées sur le planning suivant :

3.3.1 Planning prévisionnel

Tâches

Nom	Date de début	Date de fin
Formulation de groupe	30/11/2015	30/11/2015
Réunion1 avec le Client	02/12/2015	02/12/2015
Réunion1 entre les membres du Groupe	04/12/2015	04/12/2015
Création de dépôt svn + GitHub	04/12/2015	04/12/2015
Réunion2 avec le Client	18/12/2015	18/12/2015
Réunion2 entre les membres du Groupe	06/01/2016	06/01/2016
Etude de l'existant	07/01/2016	11/01/2016
Réunion1 avec le chargé de TD	25/01/2016	25/01/2016
Réunion3 entre les membres du Groupe	29/01/2016	29/01/2016
Réunion3 avec le Client	05/02/2016	05/02/2016
Réunion4 entre les membres du Groupe	18/02/2016	18/02/2016
Réunion2 avec le chargé de TD	04/03/2016	04/03/2016
Réunion5 entre les membres du Groupe	07/03/2016	07/03/2016
Réunion3 avec le chargé de TD	18/03/2016	18/03/2016
Réunion6entre les membres du Groupe	19/03/2016	19/03/2016
Réunion7 entre les membres du Groupe	21/03/2016	21/03/2016
Réunion4 avec le chargé de TD	31/03/2016	31/03/2016

3.3.2 Diagramme de Gantt



3.4 Développement

Intro

3.4.1 Tâches

Bla

Priorité	Nom	Raison
1	Tache 1	Doit être vérifié en premier car sinon [...]
2	Tache 2	On doit pouvoir [...]
3	Tache 3	Comme les principales fonctionnalités permettant de tester sont opérationnelles, nous pouvons passer à cette tâche.
4	Tache 4	Parce que [...]
5	Tache 5	La tache 5 fait partie des principales [...].
6	Tache 6	Dernière fonctionnalité essentielle à mettre en place.
7	Tache 7	Non-essentiel, mais apporterait un plus au projet.
8	Tache 8	Non-essentiel, mais apporterait un plus au projet.

FIGURE 3.1 – Tableau récapitulatif des tâches

3.4.2 Tests

Bla

Fonctionnalité	Test
Fonction 1	Quand [...], vérifier [...].
	Et quand [...], vérifier [...].
Fonction 2	Vérifier [...].
Fonction 3	Vérifier [...].
Fonction 4	Avoir [...].
Fonction 5	Accéder à [...].
	Vérifier que [...].
Fonction 6	Accéder à [...].
	Et vérifier [...].
Fonction 7	Installer [...].
	Vérifier [...].
Fonction 8	Compter [...].

FIGURE 3.2 – Tableau récapitulatif des tests

Chapitre 4

Architecture et description du logiciel

Dans cette partie, nous allons d'abord vous parler de la partie concernant le parsing puis passer à la partie d'analyse des données et de rendu graphique.

4.1 Partie parser

Cette partie est écrite en java, l'autre possibilité que nous avions était de coder le parser en python mais après quelques petits tests nous nous sommes rendus compte que le parser en python serait beaucoup trop lent. Voici l'architecture de la partie Java de notre projet :

En raison de logs au format variable, il est nécessaire de produire plusieurs versions de parser afin de garder une robustesse au niveau du parsing et d'éviter une classe trop grosse. Pour cela, à l'ouverture d'un log, le LogHandler demande la création d'un parser à une classe utilisant le design pattern factory. Cette classe va examiner le log en question et choisir la bonne version du parser a utiliser puis renvoyer ce parser au log handler. Afin d'éviter des duplications de code inutiles, le parser en lui même est implémenté à l'aide d'une classe abstraite regroupant toutes les similitudes entre les parsers puis les différences sont implémentées dans la classe du parser en question à proprement parler. Le LogHandler va ensuite envoyer sur une base de données les données obtenues à l'aide du parser.

4.2 Partie Analyse

Cette partie est écrite en python à la demande du client. Nous avons choisi de proposer plusieurs librairies contenant des outils besiques permettant de travailler sur les données acquises lors du parsing. Voici un schéma regroupant les différentes librairies et leurs liens : Tout d'abord, le module MongoInterface permet de communiquer avec la base de données.
Les objets Matchs et Player sont utilisés lors de la récupération de données afin de le mettre en forme pour une utilisation plus aisée par le client. Ce sont des structures de données regroupant toutes les information relatives au match ou joueur en question.
La bilbiothèque Tools regroupe les fonctions d'analyse basique pouvant être utilisées sur les logs, pour appliquer ces fonctions, il faut utiliser la fonction *apply_function_to_query*. Si dans le futur, le client ou bien des personnes souhaitant travailler sur ce projet veulent étoffer les fonctions proposées, il suffira de les ajouter dans cette libraire.
La solution proposée pour afficher les resultats consiste en un module Plotter et un module DataSet. Le module DataSet permet de mettre en forme les données à représenter de manière simple pour l'utilisateur, et le module Plotter propose une solution simple pour afficher des résultats de base (lignes cassées, nuage de points ou barres).

4.3 Extensions possibles

Sur la partie parser, il est très facile d'ajouter une nouvelle version de parser permettant de lire par exemple des logs d'un nouvelle version du serveur de jeu, cela permettrait de regouper des données sur une plus grande période et donc d'obtenir des statistiques plus intéressantes.
Sur la partie Python, il est possible d'ajouter de nouvelles fonctions d'analyse des données, des données peuvent également être rajoutées pour les logs mais il faudra faire attention à la lecture de ces données car les objets ne permettrons pas un accès facile à celles-ci (sauf si l'utilisateur modifie également les structures de données).

Dans cette partie nous cherchons à décrire dans un premier temps [...], puis, c[...].

4.4 Partie 1

Intro

4.4.1 Sous-partie 1

FIGURE 4.1 – autre partie image 1 [1]

1. Schéma d'après : *Auteur 1 & Propriétaire image*, LICENCE (cf. ref. [**?**])

4.4.2 Sous-partie 2

FIGURE 4.2 – autre partie globale de notre quelque chose

Nous retrouvons ici, blabla [2] [...].

4.4.2.1 Sous-sous-partie 1

Le bla (cf. ref. [**?**]) est [...] :
— item1 ;
— item2 ;
— item3 ;
— item4 ;
— item5.

2. Application bla - Interface blabla

4.4.2.2 Sous-sous-partie 2

4.4.2.2.1 Paragraphe 1 (agissant comme titre niveau 5)

FIGURE 4.3 – Structure d'une autre chose [3]

Ce schéma représente bla.

4.4.2.2.2 Paragraphe 2

Bla

Sous-paragraphe 1

Bla

FIGURE 4.4 – Diagramme de truc

3. Schéma et explication d'après le wiki bla (cf. ref. [**?**])

Sous-paragraphe 2

Bla

Bla

Sous-paragraphe 3

Bla

4.4.2.3 Sous-sous-partie 3

Bla

4.5 Partie 2

Bla
Bla

4.5.1 Sous-partie 1

Bla

4.5.2 Sous-partie 2

Bla

Paragraphe 1 (n'apparaitra pas dans l'index) Bla

Paragraphe 2 Bla

Paragraphe 3 Bla

4.5.3 Sous-partie 3

Bla

3. D'après le schéma disponible sur la documentfation officielle disponible sur le site blalbla

Chapitre 5

Autre partie

Dans cette partie nous cherchons à décrire dans un premier temps [...], puis, c[...].

5.1 Partie 1

Intro

5.1.1 Sous-partie 1

FIGURE 5.1 – autre partie image 1 [1]

1. Schéma d'après : *Auteur 1 & Propriétaire image*, LICENCE (cf. ref. [**?**])

5.1.2 Sous-partie 2

FIGURE 5.2 – autre partie globale de notre quelque chose

Nous retrouvons ici, blabla [2] [...].

5.1.2.1 Sous-sous-partie 1

Le bla (cf. ref. [?]) est [...] :
— item1 ;
— item2 ;
— item3 ;
— item4 ;
— item5.

2. Application bla - Interface blabla

5.1.2.2 Sous-sous-partie 2

5.1.2.2.1 Paragraphe 1 (agissant comme titre niveau 5)

FIGURE 5.3 – Structure d'une autre chose [3]

Ce schéma représente bla.

5.1.2.2.2 Paragraphe 2

Bla

Sous-paragraphe 1

Bla

FIGURE 5.4 – Diagramme de truc

3. Schéma et explication d'après le wiki bla (cf. ref. [**?**])

Sous-paragraphe 2

Bla

Bla

Sous-paragraphe 3

Bla

5.1.2.3 Sous-sous-partie 3

Bla

5.2 Partie 2

Bla
Bla

5.2.1 Sous-partie 1

Bla

5.2.2 Sous-partie 2

Bla

Paragraphe 1 (n'apparaitra pas dans l'index) Bla

Paragraphe 2 Bla

Paragraphe 3 Bla

5.2.3 Sous-partie 3

Bla

3. D'après le schéma disponible sur la documentfation officielle disponible sur le site blalbla

Chapitre 6

Fonctionnement et Tests

6.1 Partie 1

Intro

6.1.1 Sous-partie 1

Paragraphe 1 (n'apparaitra pas dans l'index) Bla

Paragraphe 2 Bla

Paragraphe 3 Bla

6.1.2 Sous-partie 2

Bla

6.1.3 Sous-partie 3

Bla

6.2 Partie 2

Intro

Sous-partie 1 ('apparaitra pas dans l'index)

Bla

Paragraphe 1 ('apparaitra pas dans l'index) Bla

Paragraphe 2 Bla

Paragraphe 3 Bla

Sous-partie 2

Bla

(a) Première image (b) Deuxième image (c) Troisième image

(d) Quatrième image (e) Cinquième image

FIGURE 6.1 – Différents screenshots quelque chose, en gallerie

Chapitre 7

Résultats

7.1 Partie 1

Intro

7.1.1 Sous-partie 1

Paragraphe 1 (n'apparaitra pas dans l'index) Bla

Paragraphe 2 Bla

Paragraphe 3 Bla

7.1.2 Sous-partie 2

Bla

7.1.3 Sous-partie 3

Bla

7.2 Partie 2

Intro

Sous-partie 1 ('apparaitra pas dans l'index)

Bla

Paragraphe 1 ('apparaitra pas dans l'index) Bla

Paragraphe 2 Bla

Paragraphe 3 Bla

Sous-partie 2

Bla

(a) Première image (b) Deuxième image (c) Troisième image

(d) Quatrième image (e) Cinquième image

FIGURE 7.1 – Différents screenshots quelque chose, en gallerie

Chapitre 8

Bilan

Intro / Rappel Contexte
Nous avons donc pu en tirer la problématique suivante :

Problématique du sujet

Bla
Bla

Bla

Bla
Bla

Bla
Bla

Bla
Bla

Bla

Annexes

Annexe 1

Intro

1 Partie 1

Bla

1.1 Sous-partie 1

Bla

1.2 Sous-partie 2

Bla

1.3 Sous-partie 3

Bla

2 Partie 2

Bla

2.1 Sous-partie 1

Bla

2.2 Sous-partie 2

Bla

2.3 Sous-partie 3

Bla

Annexe 2

Intro

Prérequis

Bla
— item1 ;
— item2 ;
— item3 ;
— item4.
Bla

1 Partie 1

Bla

1.1 Sous-parie 1

Bla

1.2 Sous-parie 2

Bla

2 Partie 2

Attention !
Texte d'avertissement

Bla

3 Partie 3

Bla

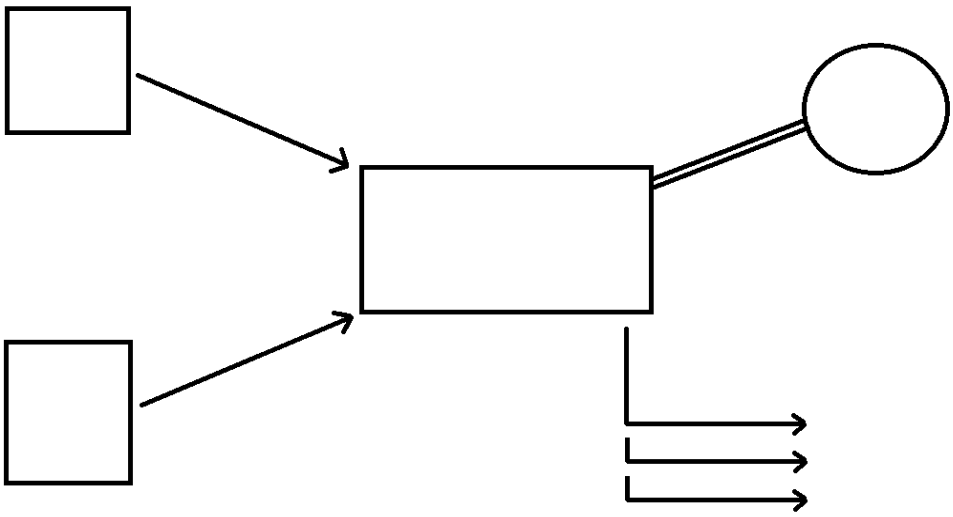


FIGURE 8.1 – Presentation schema

Paragraphe 1

Bla

Paragraphe 2

Bla

Paragraphe 3

Bla