GEN – Mini Projet

Version number 1.1.4

Drunk&Smart – Rapport Intermédiaire

Table of Contents

I. Fonctionnement général

1) Synopsis du jeu

Le jeu est un jeu de question-réponse (quizz), joueur contre joueur ou joueur contre ordinateur. Les joueurs incarnent toujours un étudiant et l’ordinateur sera soit assistant ou un professeur de la HEIG-VD.

Il existe donc deux mode de jeu : le mode Aventure (joueur vs ordinateur) et le mode Challenge (joueur contre joueur). Les joueurs progressent en répondants correctement, parmi les quatre choix de réponse possible, aux diverses questions qui lui sont posées. Il récupère également les questions auxquelles il a répondu correctement.

Quand un joueur possède un certain nombre de questions, il peut alors commencer à défier d’autres joueurs grâce au mode Challenge.

2) Modes de jeu

Il existe plusieurs modes de jeu disponibles :

- Aventure
- Challenge
- Plateforme (Priorité très faible)

a) Aventure

Ce mode permet de lancer une aventure ou scénario. Chaque scénario porte le nom d’une filière de la HEIG-VD.

Scénarios :
- TIC

- TIN (Priorité très faible)
- EC+G (Priorité très faible)
- COMEM (Priorité très faible)
- HEG (Priorité très faible)

Une fois le scénario choisi l'aventure débute. Un scénario consiste en une succession de niveaux, jusqu'au niveau final. Lors de chacun de ces niveaux, l'étudiant devra affronter un certains nombre d'assistants et les vaincre avant de pouvoir affronter un professeur puis une fois celui-ci vaincu il passe à l'examen final qu'il devra passé dans un temps limité.

Une fois l'ultime professeur (légendaire) et son examen final réussi, le scénario est alors terminé.

Après avoir remporté un certain nombre de victoire dans ce mode, et donc obtenu un certains nombre de question, il est alors possible d'accéder au mode Challenge.

b) Challenge

Ce mode consiste à défier un autre joueur. Chaque joueur pose tour à tour une question qu'il a gagnée lors de l'un de ses modes Aventure à son adversaire. Le temps pour poser la question est pour y répondre est limité afin d'éviter qu'un joueur puisse prendre en otage l'affrontement en laissant tourner l'application jusqu'à ce que l'autre joueur quitte l'application.

(Optionnel) Le joueur qui gagne le Challenge remporte des points de classements lui donnant un classement général parmi tous les joueurs inscrits. Le joueur qui perd, perd alors des points de classement.

c) (Très optionnel) Le mode plateforme

Il s'agit d'un mode alternatif permettant de remporter des objets utiles pour le mode Aventure ou Versus. Il s'agirait d'un mini-jeu de plateforme ou le joueur manipulerait son personnage à travers différents niveau pour récupéré les dit objets.

3) Ressources

a) Point de vie

b) Point d'expérience

c) Niveau

d) Objets

e) État d’avancement dans les scénarios

f) Pool de question à poser

a) Points de vie

Chaque entité joueur (ordinateur ou joueur « humain ») possède un nombre de point de vie initial (défini par sa classe).

Les points de vie sont perdu soit parce que le joueur a répondu incorrectement à une question posée, soit parce que son adversaire à répondu correctement à une question qu’il lui à posé.

Les points de vie ne peuvent être gagné que grâce à certains objets.

b) Points d’expérience

Seul les joueurs « humain » possèdent des points d’expérience. Ces points sont acquis par la joueur lorsque celui-ci remporte un affrontement. Il ne sont jamais perdu. Une fois un certains palier atteint, le joueur gagne un niveau mais ces points d’expérience retourne à 0. Les paliers à atteindre s’incrémente de niveau en niveau.

c) Niveaux

Les niveaux sont un indicateur de la menace que représente un adversaire (ordinateur ou humain). Ils sont gagné une fois un certain palier de point d’expérience atteint. Il ne sont JAMAIS perdu.

d) Objets

Chaque joueur démarre le jeu avec un ou plusieurs objets initiaux (en fonction de sa classe). Les objets sont des “joker” qui permettent aux joueurs d’obtenir un regain de point de vie, une aide pour une question ou autre allègement de la difficulté.

Les objets peuvent être gagné :

- En récompense à la fin d’un affrontement remporté (pourcentage de chance)
- Lors du mode plateforme (Très optionnel)

Les objets sont perdu après leur utilisation.

Exemple d’objets :

- Une bière : redonne des points de vie
- Une anti-sèche : enlève une mauvaise réponse des propositions de réponses
- Un certificat médical : permet de passer à une autre question sans dommage aucun.

Ces exemples ne sont ni exhaustifs, ni définitifs.

e) État d’avancement dans les scénarios

Chaque joueur possède un état d’avancement pour chaque scénario qu’il a démarré. Chaque scénario possède des « checkpoints » et lorsqu’un joueur atteint l’un de ces « checkpoints » son état d’avancement est mis à jour. Les endroits et l’espacement des « checkpoints » ne sont pas encore défini. Si un joueur quitte un scénario (ou l’application), pendant un affrontement ou n’importe quand, il reprend sont périple au derniers « checkpoint » atteint.

f) Pool de question à poser

Il s’agit d’une base de question dans laquelle joueur comme ordinateur peuvent aller chercher une question à poser à son adversaire. Les questions sont toujours liées à leurs propositions de réponses.

Le joueur possède

- Un nombre de point de vie
- Un nombre de point d’expérience
- Un niveau
- Zéro à plusieurs objets pouvant l’aider
- Un état d’avancement dans le jeu
- Un pool de questions durement acquises

Les professeurs/assistants ont

- Un nombre de point de vie
- Un niveau
- Un pool de questions sur le serveur, définies par les administrateurs

4) Classe de joueur

Il s’agit d’une précision concernant le type d’étudiant que le joueur incarne. Par exemple, l’intello, le fêtard, le glandeur, le tricheur etc. *(il s’agit d’exemple et ce n’est pas définitif)*

La classe définira

- Les points de vie initial du joueur
- Les « dégâts » du joueur *(optionnel)*
- Les objets de départ du joueur
- Les objets qu’il a le plus de chance de gagner *(optionnel)*
- Autres avantages à voir (*optionnel)*

5) Objectifs

- Mode aventure

  Le but de ce mode est de terminer un à plusieurs des scénarios qui sont proposés. Lors d'un scénario, le but est de réussir tous les niveaux qui le compose. Pour réussir un niveau, il faut remporter tous les affrontements proposés dans celui-ci.

- Mode Challenge

  Ce mode propose des affrontements contre d'autre joueurs. Le but est donc de remporter cet affrontement contre le joueur adverse. Le joueur remportant l'affrontement est déclaré vainqueur du Challenge.

- Remporter un affrontement

  Pour remporter un affrontement, il faut que les points de vie de son adversaire (joueur ou ordinateur) tombe à zéro.

6) Déroulement du Jeu

1. Le joueur lance le jeu

2. **Si le joueur ne possède pas déjà un compte**
   - Le joueur est invité à le faire
   - Il choisit alors sa classe

   Sinon
   - passe à l'étape 3

3. Le joueur arrive sur son menu principal (dashboard) ou il peut voir:
   - Son niveau
   - Son nombre de point d'expérience
   - Ses objets éventuels

   A ce niveau le joueur peut (via des boutons):
   - Démarrer le mode Aventure (joueur contre ordinateur) et passe à l'étape 4
   - **Si il possède suffisamment de question**, démarrer le mode Challenge (joueur contre joueur) et passe à l'étape 8
   - Accéder à la liste de ses haut-faits, ses réussites **(optionnel – sera peut être intégré)**
   - Accéder à un tableau des scores **(optionnel – sera peut être intégré)**

4. Le joueur accéder à sa prochaine confrontation, selon son avancement dans le jeu. Il affronte alors le prochain assistant ou professeur. **(voir déroulement d'un affrontement)**
5. **Si le joueur gagne**
    - il gagne de l'expérience
    - il gagne les questions posées par le professeur
    - il gagne Zéro à plusieurs objets
    - il peut passer à la suite du mode (avancement dans le jeu)

    Si le joueur perd
    - il ne gagne rien
    - les objets utilisé lors de l'affrontement sont perdu
    - il reste au même point dans le mode (il n'avance pas plus dans le jeu)

6. **Si le joueur possède suffisamment de point d'expérience**
    - joueur gagne un niveau

    Sinon
    - passer à l'étape 7

7. Recommencer à l'étape 4 jusqu'à ce que le mode Aventure soit terminé.

8. Le joueur choisi son adversaire (autre joueur)

9. Le joueur lance alors un affrontement **(voir déroulement d'un affrontement)**

10. **Si le joueur gagne**
    - il gagne de l'expérience
    - il gagne Zéro à plusieurs objets
    - il gagne des point de classement **(optionnel – sera peut être intégré)**

    Si le joueur perd
    - il ne gagne rien
    - les objets utilisé lors de l'affrontement sont perdu
    - il perd des points de classement **(optionnel – sera peut être intégré)**

7) Déroulement d'un affrontement (joueur contre ordinateur)

1. Le professeur/assistant pose une question sur son domaine d'enseignement
2. Le joueur se voit proposer quatre proposition de réponses
3. **Le joueur peut alors**
    - choisir l'une des quatre options et passe à l'étape 4

- utiliser un objet s'il en possède un **(voir exemple d'objet)** et reste à l'étape 3

4. **Si le joueur choisi la réponse correct**
   - le professeur/assistant perd des points de vie

   Sinon
   - Le joueur perd des points de vie
5. Si le joueur n'a plus de points de vie → le joueur perd **(fin de l'affrontement)**
6. Si le professeur n'a plus de points de vie → le joueur gagne **(fin de l'affrontement)**
7. Retour à l'étape 1.

8) Déroulement d'un affrontement (joueur contre joueur)

1. On tire au hasard le joueur qui commence
2. Le premier joueur pose alors une question qu'il possède au joueur adverse
3. **Ce dernier peut alors**
   - choisir l'une des quatres options de réponse et passe à l'étape 4
   - utiliser un objet s'il en possède un **(voir exemple d'objet)** et reste à l'étape 3
4. **Si il choisi la réponse correct**
   - le joueur ayant posé la question perd des points de vie

   Sinon
   - Il perd des points de vie
5. Si il n'a plus de points de vie → Il perd **(fin de l'affrontement)**
6. Si le joueur ayant posé la question n'a plus de points de vie → il gagne **(fin de l'affrontement)**
7. Retour à l'étape 1 et le joueur posant la question change (et ainsi de suite)

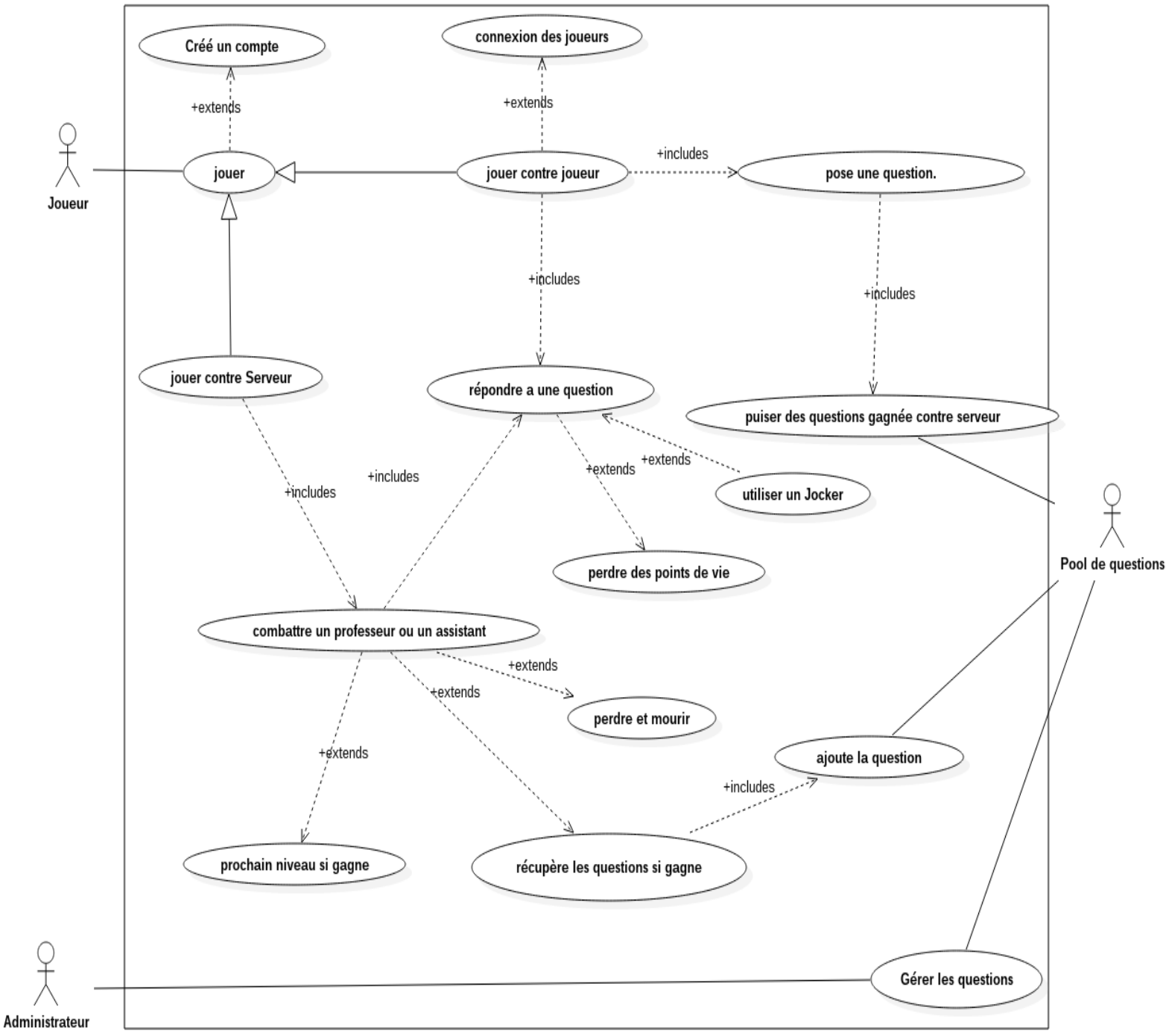
N'importe quand si un joueur quitte l'application pendant l'affrontement, la victoire est concédée à son adversaire (par abandon).

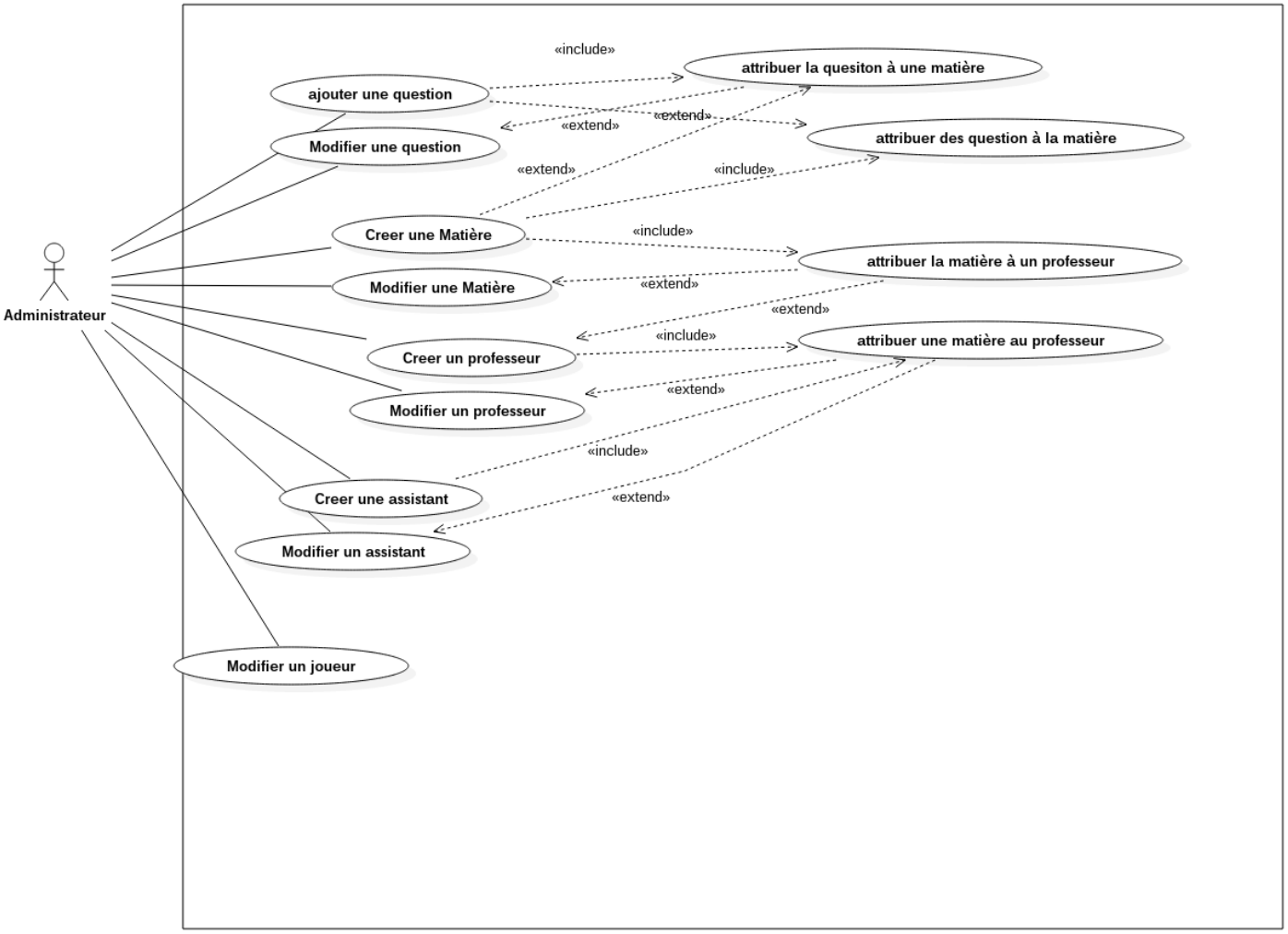
9) Interface graphique

Lors des affrontements, l'interface sera à peu près le même que celui des jeux Pokemon (voir brouillon interfaces utilisateurs - **Point VI**).

II. Cas d'utilisation

1) Diagramme général




Administrateur
ajouter une question
Modifier une question
Creer une Matière
Modifier une Matière
Creer un professeur
Modifier un professeur
Creer une assistant
Modifier un assistant
Modifier un joueur
attribuer la quesiton à une matière
attribuer des question à la matière
attribuer la matière à un professeur
attribuer une matière au professeur
«include»
«extend»

2) Acteurs

a) Administrateur

- Ajouter/Créer/Modifier

  L'administrateur a le pouvoir d'ajouter, créer, modifier des éléments de la base de donnée tel que des matières définissant le type de question existantes, des questions, des professeurs, des assistants.

- Modifier un joueur

  l'administrateur a la possibilité d’intervenir sur le compte utilisateur d'un joueur et d'en modifier des caractéristiques.

- Attribuer

  l'administrateur a la possibilité d'attribuer les questions aux matières correspondantes et vice-versa. Il peut aussi attribuer des matières a un professeur ou l'inverse.

b) Joueur

- Jouer

  Le joueur peut lancer l'application et choisir soit de jouer contre un serveur, soit de jouer contre un autre joueur.

  Pour cela le joueur devra avoir créé un compte via l'interface de création de compte.

- Créer un compte

  Cela inclus de passer par l'interface de création de compte afin de se connecter ou de s'ajouter en tant qu'utilisateur de l'application et donc de se créer un compte.

3) Scénarios

a) Joueur contre Joueur

- Si le joueur souhaite jouer contre un autre joueur, il devra attendre la connexion de son adversaire.
- Le mode de jeu "joueur contre joueur" inclus le fait de poser des questions à l’adversaire puis de répondre aux siennes. A travers ce jeu de questions réponses plusieurs événement peuvent se passer tel que perdre et donc perdre des points de classements.

- **gagne objet si gagne**

  Si le joueur gagne le "combat" contre son adversaire, celui-ci peut récupérer certains objet pouvant l'aider a avancer dans le jeu.

- **gagne points de classement si gagne**

  Si le joueur gagne le "combat" contre son adversaire, il va alors gagner des points de classement.

- **perd points de classement si perd**

  Si le joueur perd le "combat" contre son adversaire, il va alors perdre des points de classement.

- **pose une question**

  Durant un "combat" contre un adversaire, chaque joueur pose a tour de rôle une question qu'il puise dans la liste de question a disposition.

- **sélectionner une question parmi celles gagnées**

Lorsque un joueur pose une question, il en choisit une parmi celle gagnée dans le mode histoire.

b) joueur contre serveur

- le mode de jeu "joueur contre serveur" inclus le fait d'avancer dans l'aventure qui inclus de combattre des professeurs/assistants.
- un combat contre un professeur/assistant inclus de répondre aux questions de "l'ennemi" si on répond juste on passe à la question suivante.
- Si un joueur "tue" un ennemi, il gagne un niveau et passe à l'ennemi suivant.
- Si un joueur tombe à zéro points de vie, il perd et meurt. Il doit donc recommencer le combat.
- Si un joueur réussit à combattre un professeur/assistant il gagne toutes les questions répondues justes et les ajoutes à son "pool" de questions. Ces questions lui seront utiles en mode joueur contre joueur.

- **combattre une professeur ou un assistant**

  Durant le mode histoire "contre serveur" le joueur va devoir d'attaquer à des assistants puis les professeur. Celui-ci devra les battre afin de passer a l'étape suivante.

- **prochain niveau si gagne**

  Si un joueur combat un assistant ou un professeur il peut alors passer au niveau suivant lui proposant alors un nouvel adversaire.

- **gagne expérience si gagne**

  Si un joueur gagne dans un combat en mode histoire, il va alors gagner en retour des points d'expérience.

- **récupère question si gagne**

Si un joueur gagne dans un combat en mode histoire, Les questions répondues juste sont alors ajoutées a son pool de questions qu'il peut poser dans le mode joueur contre joueur.

c) Général

- **perdre et mourir**

  Lorsque que le joueur tombe a zéro point de vie, alors il meurt et devra recommencer le niveau.

- **Répondre a une question**

  Répondre au question inclus que si on répond faux, on perd des points de vie. Si on tombe à zéro on perd le combat.

  Si un joueur est en difficulté pour répondre à une question il peut choisir d'utiliser un Objet qui va l'aider en modifiant de comportement du jeux et en lui facilitant la tâche (dépend du joker utilisé).

- **perdre des points de vie**

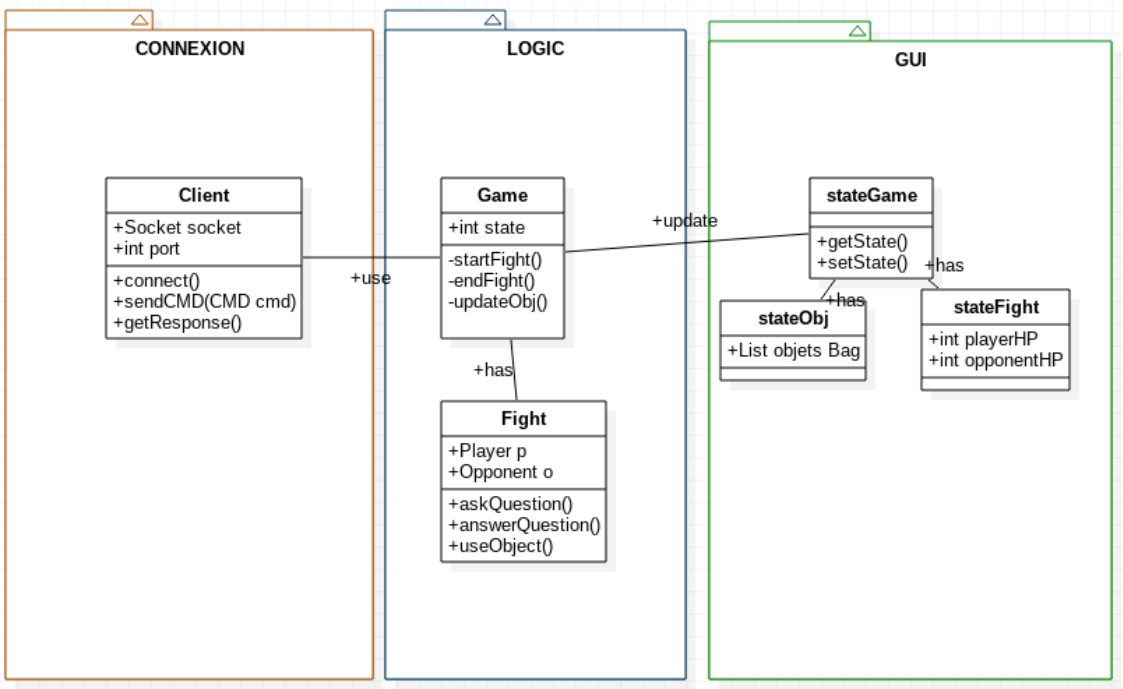
  Lorsque le joueur répond faux a une question il perd alors des points de vie.

- **utiliser un objet**

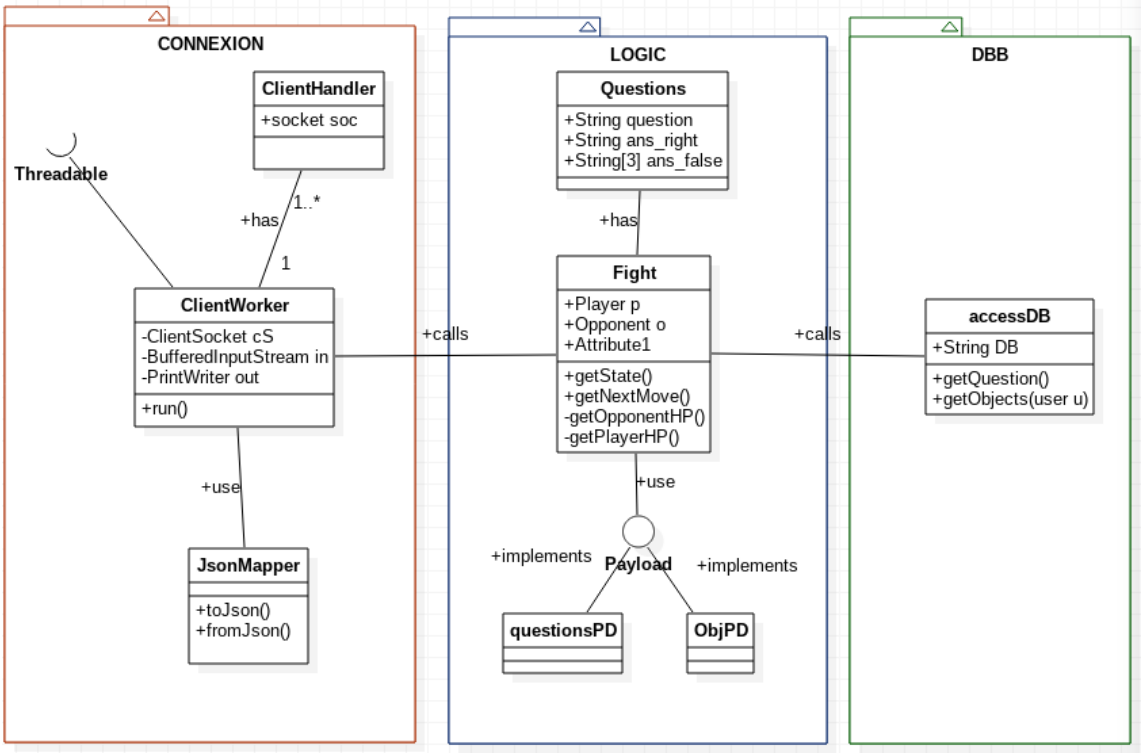
  Durant les combats, un joueur peut utiliser un objet se trouvant a ça disposition dans son inventaire. Cet objet peut être de type différent l'aidant alors a répondre a la question ou a re-gagner des points de vie.

III. Ébauches du modèle de domaine

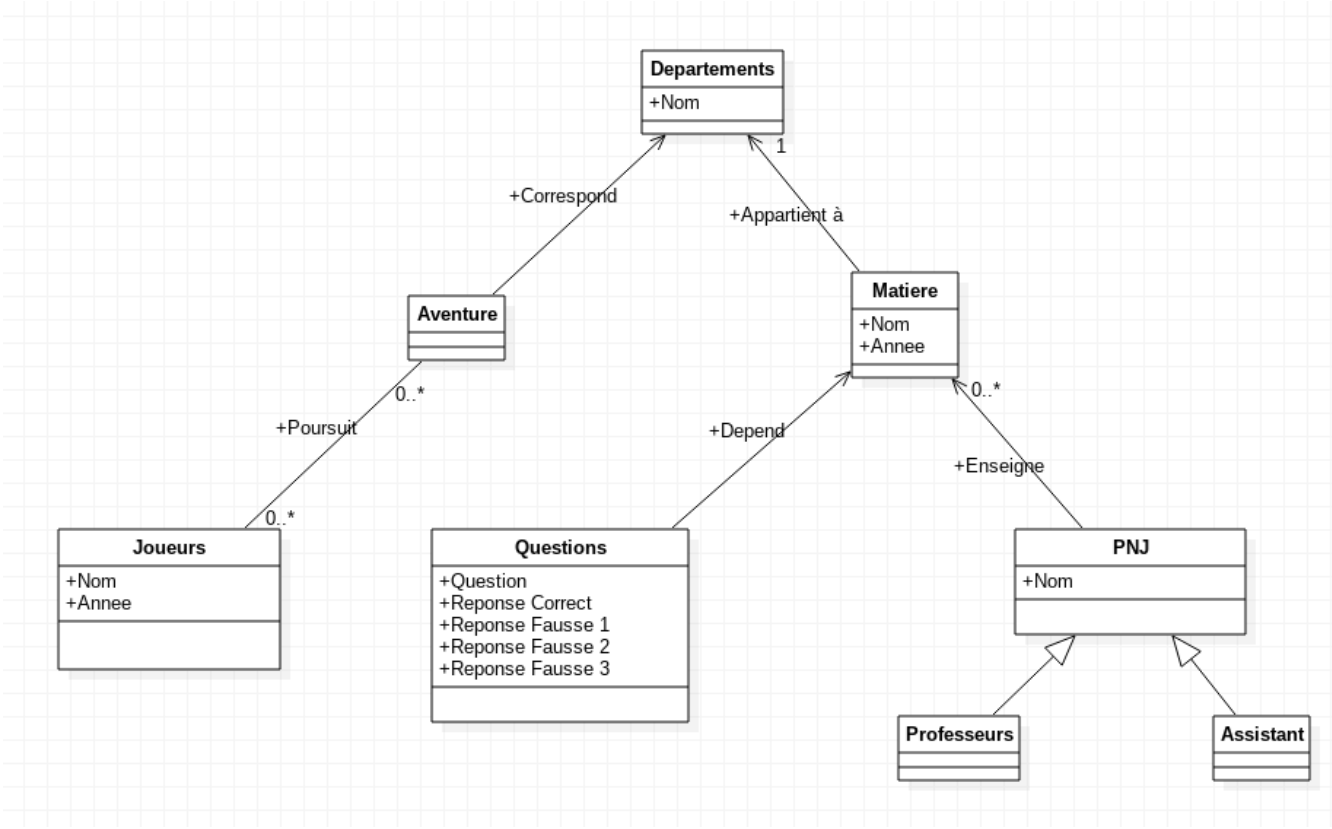
1) Modèle client



2) Modèle serveur



IV. Base de données

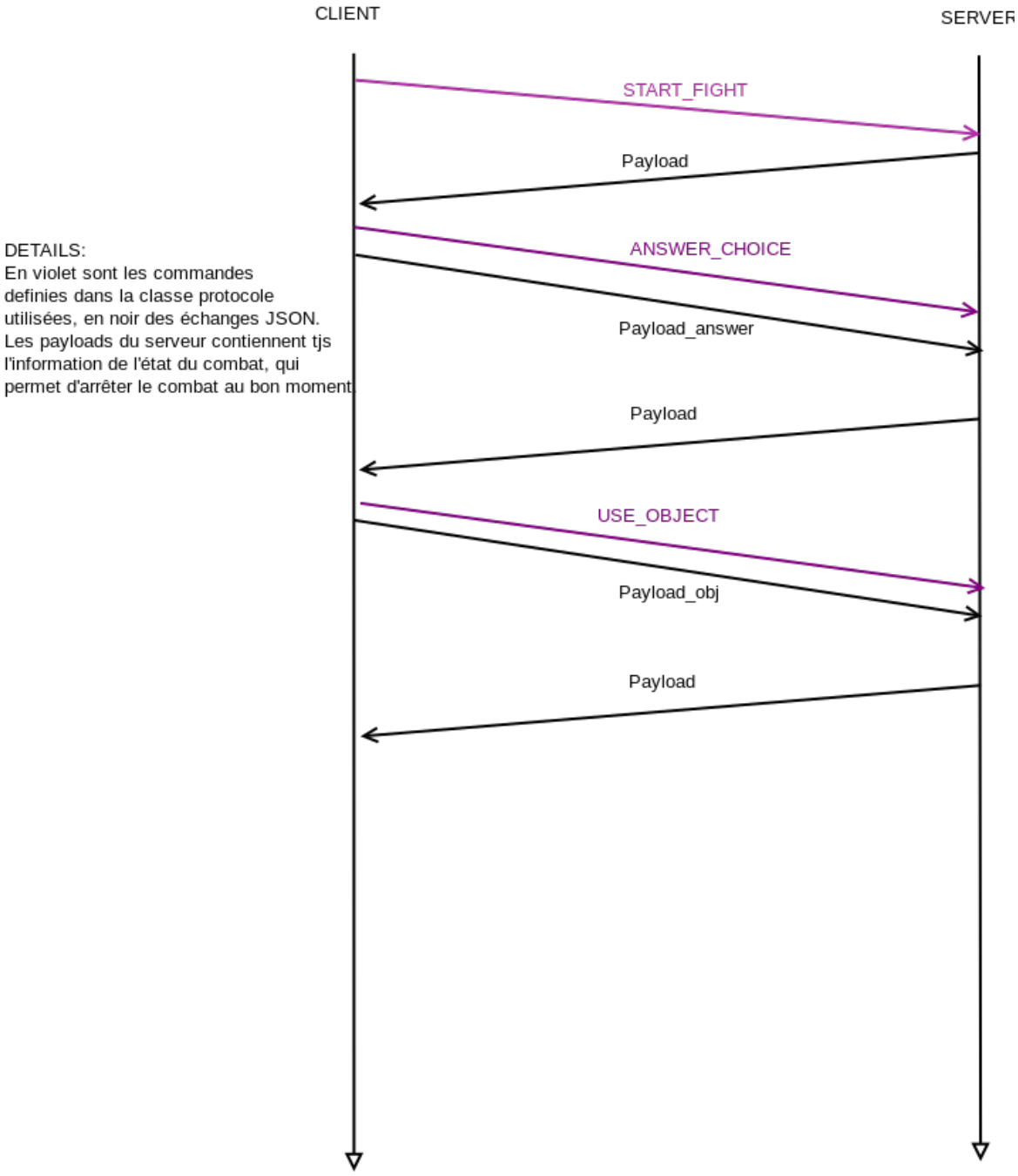


V. Serveur et Protocole d’échange

1) Serveur

- stocke l'avancement de chaque joueur dans le jeu
- gère les interactions des joueurs qui jouent contre l'ordinateur
- gère les échanges entre joueurs qui jouent contre d'autres joueurs
- stocke les questions dans une base de donnée des questions disponibles
- gère les ordinateurs (profs / assistants), choix des questions de chaque

2) Protocole



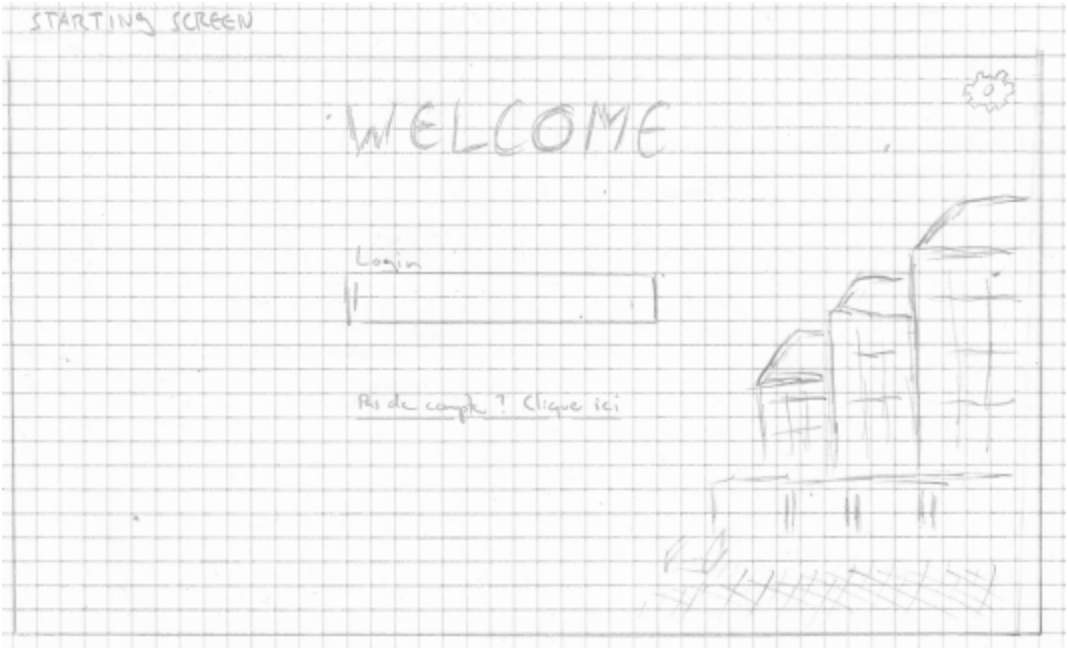
DETAILS:
En violet sont les commandes definies dans la classe protocole utilisées, en noir des échanges JSON.
Les payloads du serveur contiennent tjs l'information de l'état du combat, qui permet d'arrêter le combat au bon moment

Exemple de payload

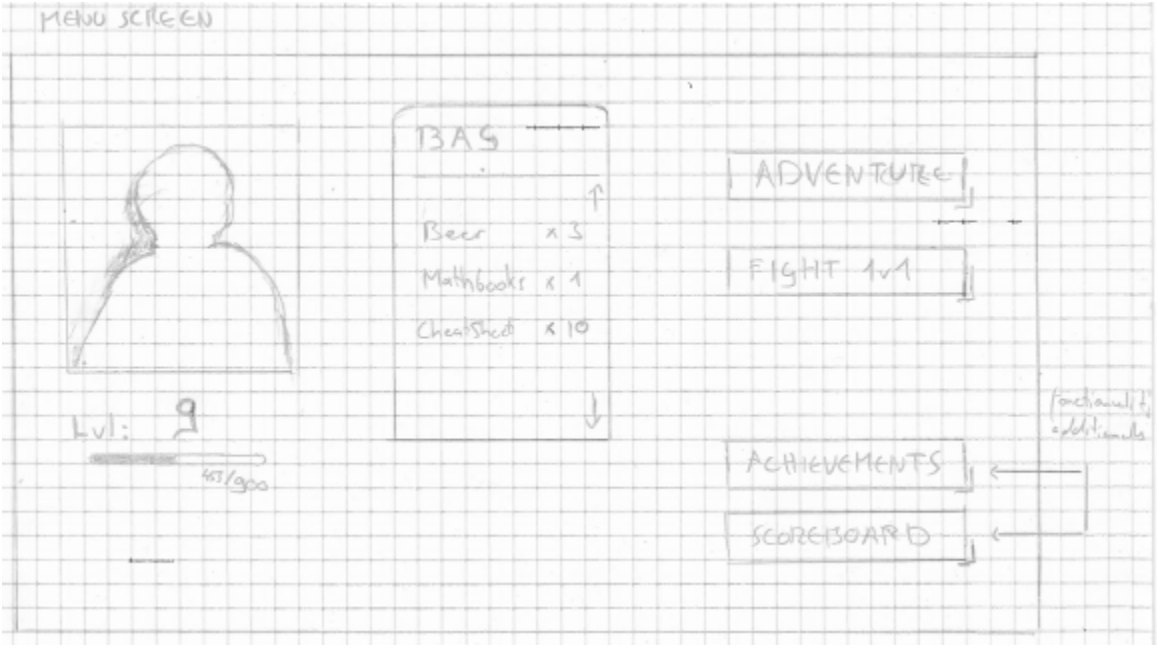
```
{"payload":
    {"question":"..",
    "ans_right":"..",
    "ans_false":["1":"..","2":"..","3":".."],
    "opponentHP":45,
    "playerHP":80
  }
 "state":"WIN|LOSE|FIGHTING"
}
```

VI. Ébauches interfaces utilisateur

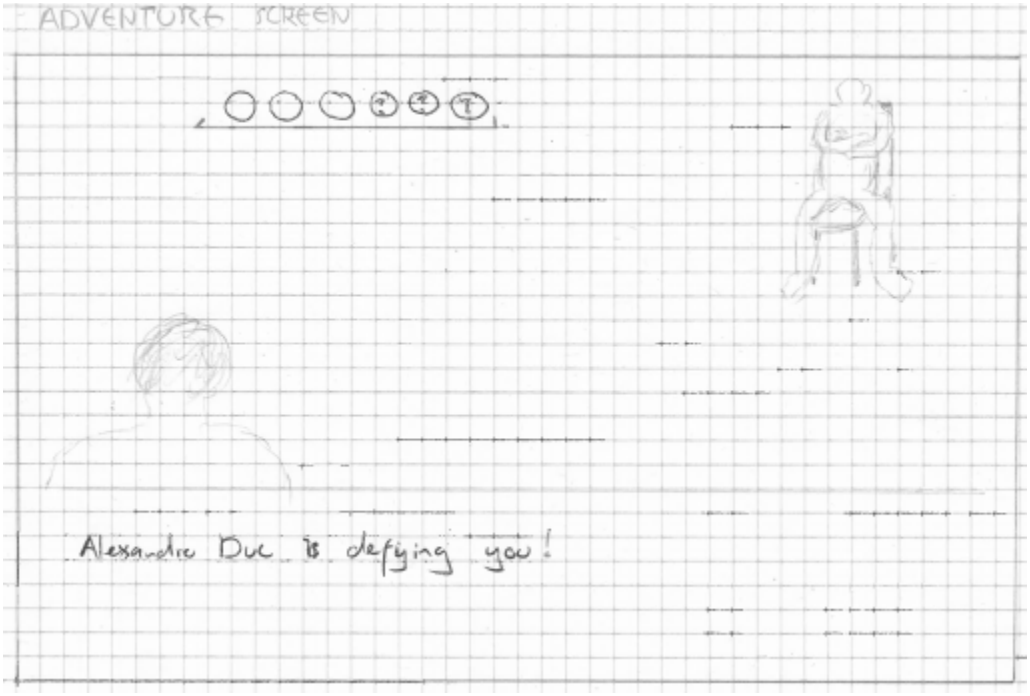
1) Starting Screen



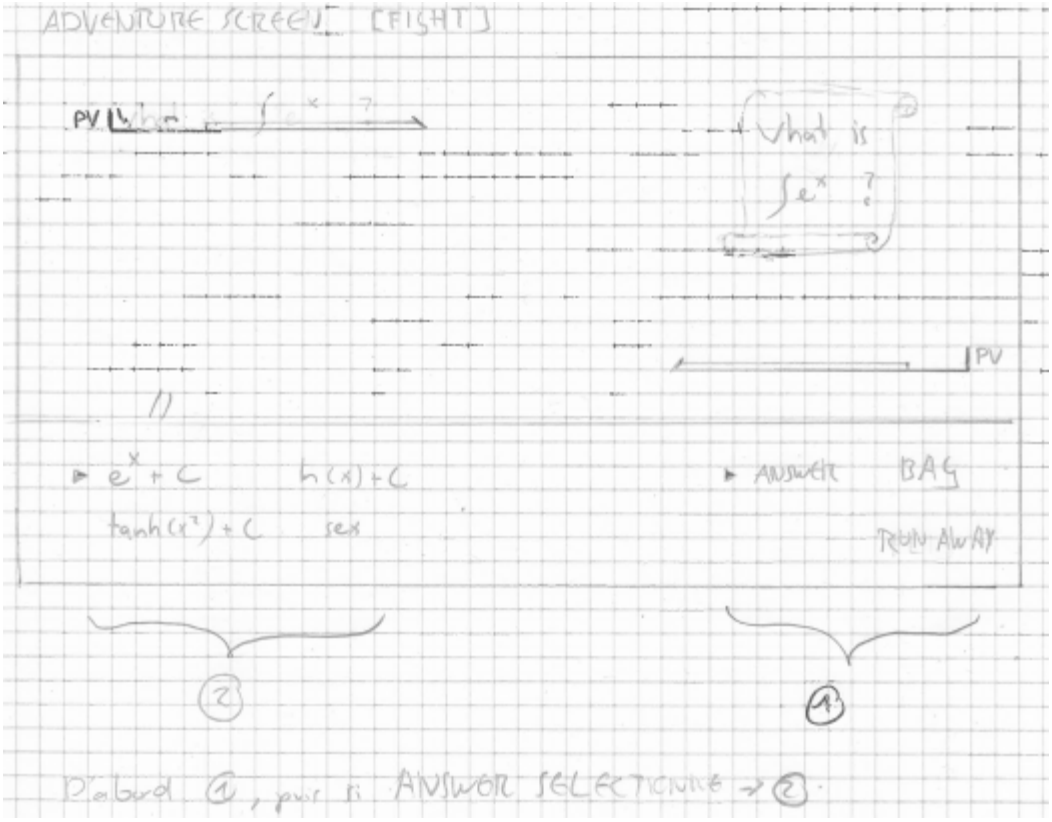
2) Menu Screen



3) Adventure Screen



4) Adventure Screen [Fight]



VII. Gestion de projet

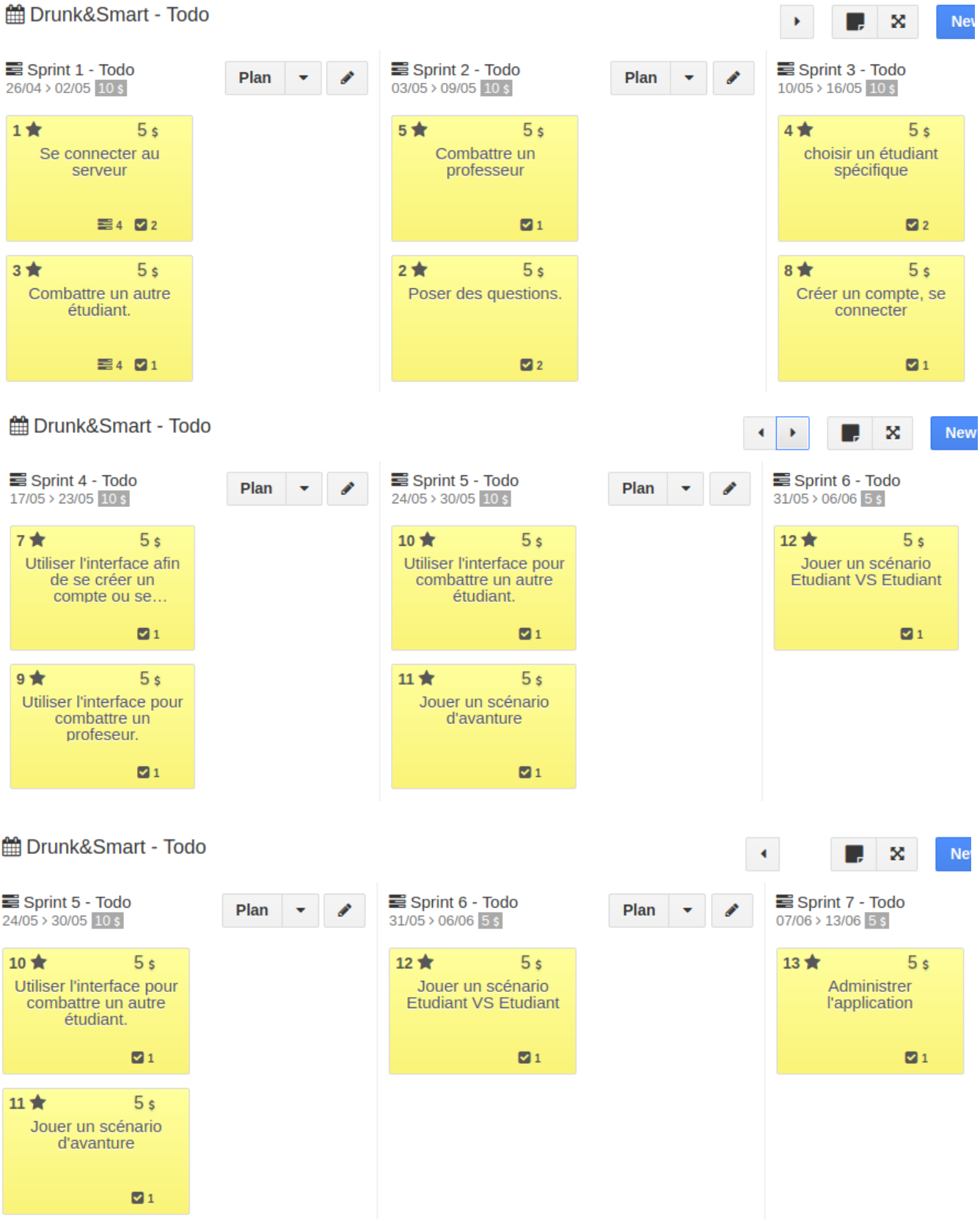
1) Rôle

- Srcrum master : Yann
- Product Owner : Joël
- Developper: Yohann, Loïc

2) Backlog

3) Plan d’itération

Tous nos sprints durent 1 semaine.

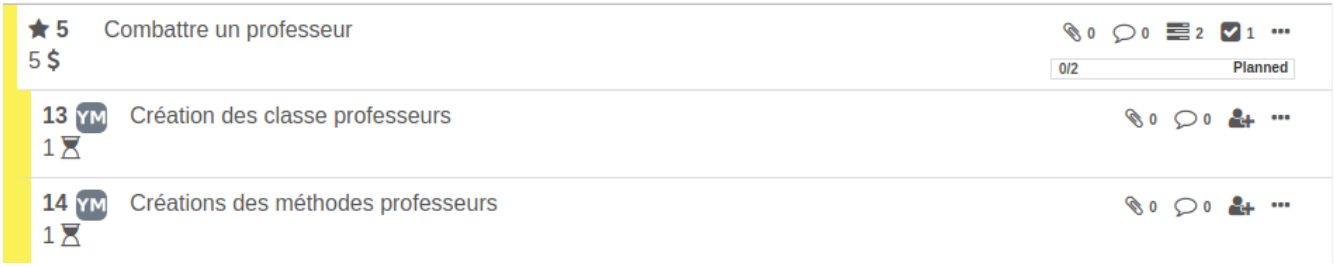

Drunk&Smart - Todo
Sprint 1 - Todo
26/04 › 02/05 10 $
Plan
1 5 $
Se connecter au serveur
4 2
3 5 $
Combattre un autre étudiant.
4 1
Sprint 2 - Todo
03/05 › 09/05 10 $
Plan
5 5 $
Combattre un professeur
1
2 5 $
Poser des questions.
2
Sprint 3 - Todo
10/05 › 16/05 10 $
4 5 $
choisir un étudiant spécifique
2
8 5 $
Créer un compte, se connecter
1
Drunk&Smart - Todo
New
Sprint 4 - Todo
17/05 › 23/05 10 $
Plan
7 5 $
Utiliser l'interface afin de se créer un compte ou se…
1
9 5 $
Utiliser l'interface pour combattre un profeseur.
1
Sprint 5 - Todo
24/05 › 30/05 10 $
Plan
10 5 $
Utiliser l'interface pour combattre un autre étudiant.
1
11 5 $
Jouer un scénario d'avanture
1
Sprint 6 - Todo
31/05 › 06/06 5 $
12 5 $
Jouer un scénario Etudiant VS Etudiant
1
Drunk&Smart - Todo
Sprint 5 - Todo
24/05 › 30/05 10 $
Plan
10 5 $
Utiliser l'interface pour combattre un autre étudiant.
1
11 5 $
Jouer un scénario d'avanture
1
Sprint 6 - Todo
31/05 › 06/06 5 $
Plan
12 5 $
Jouer un scénario Etudiant VS Etudiant
1
Sprint 7 - Todo
07/06 › 13/06 5 $
13 5 $
Administrer l'application
1

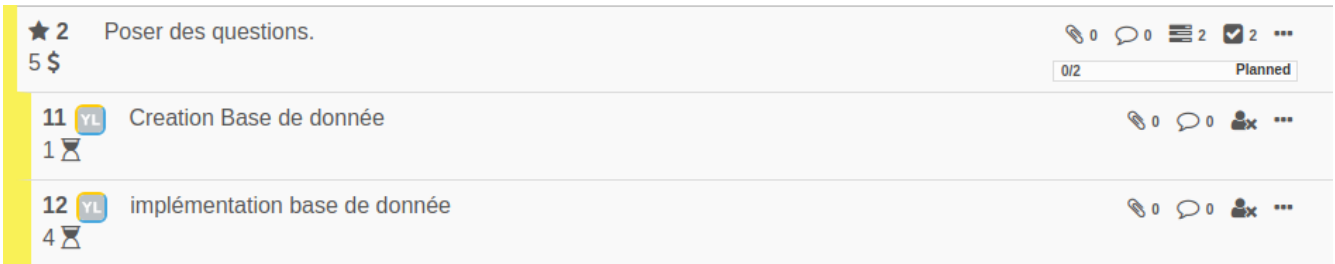
VIII. Planification détaillée des sprints

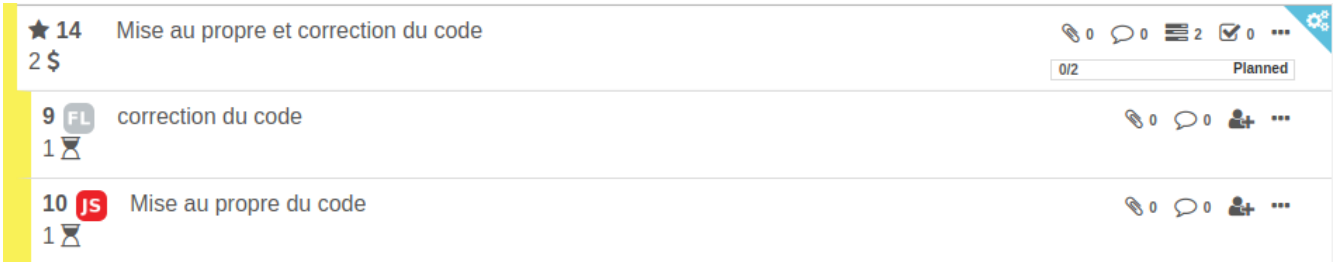
1) Sprint numéro 1

s

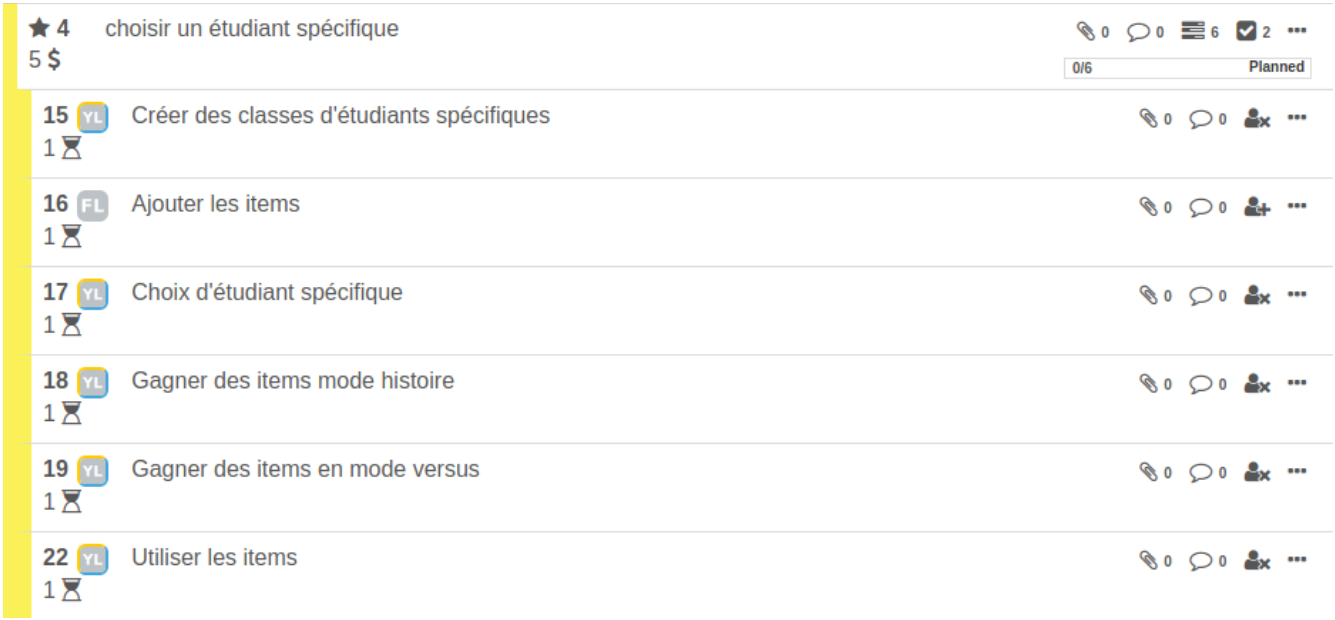
2) Sprint numéro 2



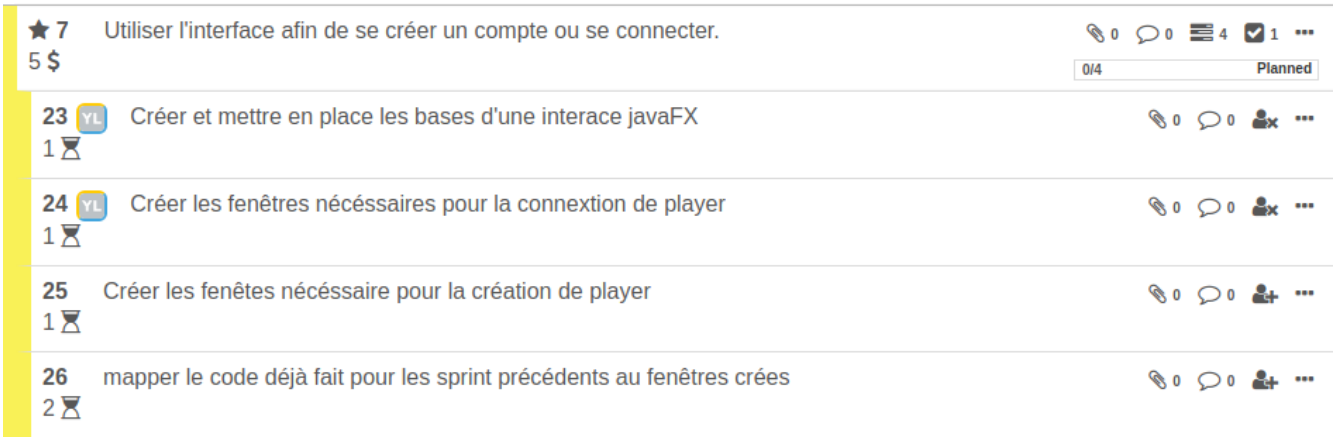


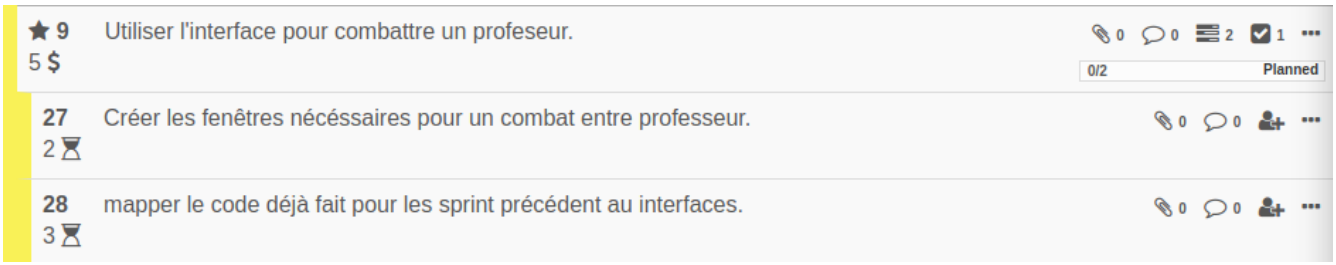


Sprint numéro 3



Sprint numéro 4





IX. Bilan Sprint

1) Sprint numéro 1

a) Histoire id(1) et id(3) planifiées, Les deux histoire ont été terminées

b) velocité: on avait prévu 10 pts d'histoire partagés en différentes tâches. Tous les histoire ont été effectuées.

c) Aucun Replanification des tâches effectuées n'est a prévoir. Mais nous ajoutons une histoire pour la réorganisation du code et corrections.

1. Nous avons pas prévus la compléxité du code pour réussir le sprint, du coup nous n'avons pas programmé de la manière la plus propre. Notre code nécéssite un remaniement et des corrections.
2. Le sprint concerné par cela est le 2.

3.

d) commantaires générals: Le sprint a été compliqué, nous avions mal estimé la difficulté de la communication server-client. Cela nous a donc posé plusieurs problèmes et prit plus de temps que prévu.

e) Nous devrions mieux gérer la difficulté des tâches et travailler sur l'enticipation du travail ainsi que les tests.

2) Bilans personnels Sprint numéro 1

- Yann Lederrey
  - temps prévu versus temps réalisé : le temps prévu était de 2 heures environ pour des tâches prévus simple qui se sont avérées plus compliquées. Au final j'ai pris plutot 3-4 heures
  - commentaire personnel : Nous avons mal anticipé la conceptualisation du code. Nous aurions du passer plus de temps sur une feuille et sur des schémas au lieu de partir dans le code, ce qui nous aurait fait gagner du temps.
- Joel Schär
  - temps prévu versus temps réalisé : heures passées : 6 heures, heures prévues 4 heures
  - commentaire personnel : La base n'étant pas encore faite, il a fallu faire tout le travail de conceptualisation du code en très peu de temp.Nous avons donc pris une première direction, que nous avons du corriger et repenser, afin d'obtenir une structure utilisable. Ces revirements nous on fait perdre du temps et nous ont compliquer le lancement du projet.
    Dans la même idée nous, n'avions pas pris le temps avant le début du premier sprint de mettre sur papier l'entiertié des protocols de communication entre les clients et le serveur. Nous avons donc implémenté quelquechose d'un peu bancale pour commencer. Du fait que cette structure n'a pas été choisie et validée il est possible qu'elle change encore dans le future et cela va encore nous faire perdre du temps.
    Je pense que nous avons pour le moment réussi à fonder une base, mais que celle-ci

nécessite encore un peu de travail de consolidation, afin d'avoir une bonne base de travail sur laquelle construire la suite des fonctionnalités.

- Yohann Mayer
  - temps prévu versus temps réalisé : Imprévu du à beleinev.
  - commentaire personnel : A cause de baleinev je n'ais pas pu avancer sur le projet durant ce sprint. Ceci a été accépté par les autres.
- Loic Frueh
  - temps prévu versus temps réalisé : Heures prévue: 3h (je pensais facile)
    Heures passées: 12h (Ah en faite non...)
  - commentaire personnel : Ce premier sprint m'a un peu surpris car une tache qui me semblait simple en apparence cachait
    en faite une complexité non soupçonnée. La Gestion de la connexion au serveur était relativement facile mais la gestion de la mise en communication d'un client vers un autre était
    quelque chose de totalement nouveau pour moi et on a du à plusieurs reprises réviser le code
    voir remanier sa structure entière. Mais sinon, j'ai beaucoup appris de ce sprint. Dans l'ensemble je dirais intense mais instructif.

3) Sprint numéro 2

a) Histoire id(5), id(2) et id(14) planifiées, Les histoires ont été terminées

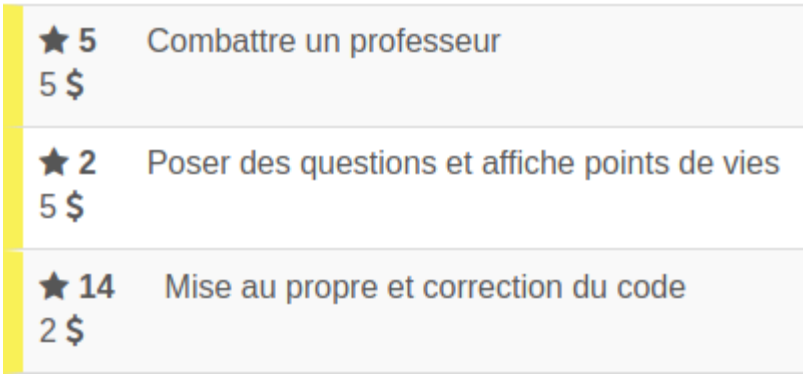
b) velocité: on avait prévu 12 pts d'histoire partagés en différentes tâches. Tous les histoire ont été effectuées.

c) Aucune Replanification des tâches effectuées n'est a prévoir. Par contre nous avons ajouté au programme l'affichage des points de vies des joueur et adversaires. Ceci n'était prévu dans aucune histoire et a été ajouté pour des raisons de confort de jeu.

1. Afin que cela soit cohérent j'ai modifé l'histoire d'ID 2 afin que cela soit y spécifié.

2. Le sprint concerné est le 2.

3.



d) Commantaires générals: Ce sprint nous a demandé énormement de temps de debug afin de corriger, améliorer et ajouter le système de question durant un combat multi-étudiants. Le mode histoire a été programmé mais mérite des amélioration durant l'histoire "Jouer un scénario d'avanture". Il nous reste potentiellement un problème de concurrence à corriger.

e) Notre démarche de travail était bonne. C'est la partie "debug" du code qui nous a demandé énormément de temps afin de comprendre ou se trouvaient nos soucis à travers les différents Threads.

4) Bilans personnels Sprint numéro 2

- Yann Lederrey : Temps prévu 5 heures : temps réalisé ~6h30.

  J'ai passé beaucoup d'heures sur le debug des soucis dans la communication du mode multiPlayer. Ceci est notre problème récurrent. Je pense qu'il faut que l'on essaye de rendre notre code plus clair/simple afin de diminuer le nombre d'heures de débug. Sinon la communication du groupe est toujours bonne et nous avançons correctement.

- Yohann Meyer :Temps prévu 5heures: temps réalisé 4 heures.

  J'ai découvert dans ce sprint l'essentiel du travail réalisé par mes collègues lors du sprint présent et me suis familiarisé avec toutes les idées implémentées.

  Après avoir développé une version du combat contre professeur, je me suis orienté sur la refactorisation du code.

- Loic Frueh : Temps prévu 4 heures, temps réalisé 12 heures.

  Une fois n'est pas coutume, ce deuxième sprint à denouveau été l'occasion de constater qu'une tache que l'on pense rapide et relativement simple peut prendre des proportions immenses dès qu'un petit bug entre dans l'équation... Le code à été vite écrit, le débuggage par contre lui à pris une eternité ! Je pense sans exagéré que sur les 12h effective de travail sur la tache, 9h ont été consacré à débugger le programme. Ce sprint m'a montré qu'une application client->serveur<-client est très difficle à débugger.

- Joel Schär : Temps prévu 2-3 heures, temps réalisé 2 heures.

  Lors de ce sprint je me suis concentré principalement sur la remise au propre et la relecture du code.
  Je commence avec ce second sprint a voir l'efficacité d'avancement d'un projet avec la méthode scrum. Avec des petits objectifs a court terme on perd rapidement de vue l'objectif final et on se concentre sur la tache en court. Chaque fin de sprint avec une solution fonctionnel est comme une petite fin de projet. et une petite victoire.

5) Sprint numéro 3

f) Histoire id(7), id(9) planifiées, Les histoires ont été terminées

g) velocité: on avait prévu 10 pts d'histoire partagés en différentes tâches. Tous les histoire ont été effectuées.

h) Aucune Replanification des tâches effectuées n'est a prévoir. Le sprint prochain sera assez conséquent et nous n'auront pas beaucoup de temps. Nous avons malgré cela fait le choix que garder le sprint tel quel afin de nous motiver. Nous replanifierons après coup si nous n'avons pas réussis à le terminer.

i) Commantaires générals: Ce sprint c'est assez bien passé, Nous avons travaillé de manière coordonné. Aucun bug bloquant ont étés rencontrés (pour une fois…). Nous avons par contre des difficultés dans le partage correcte du travail. Sans pour autant impacter sur le projet car les personnes qui sont moins impliquées dans ce projet durant un sprint, nous soulage sur le travail d'autre projets.

j) Nous avons fait une discution pour revoir la manière dont nous partagons le travail. Par exemple Yohann Meyer, ayant eu des empêchements au premier sprint. Il a prit un retard sur la comprehensions du code préalablement écrit et ce retard c'est augmanté au fur et à mesure. Il nous a aidé indirectement malgré tout.

   Nous avons décidé de laisser le prochain sprint comme il est à cause de la surcharge de travail de cette semaine. Pour le sprint 5, nous allons discuter avec Yohann Meyer et d'autres si besoin , afin de rattraper le niveau du projet pour que tout le monde puisse être correctement impliqué.

   De même que pour les tâches que nous respectons rarement. Le plus souvent on travail de la manière suivante : Si on a le temps, on avance. Même si ça signifie faire des tâches encore incomplètes et appartenant à un autre membre du groupe.

6) Bilans personnels Sprint numéro 3

- Yann Lederrey : Temps prévu 5 heures : temps réalisé 9h00.

  Ce sprint c'est plutôt bien passé, j'ai rapidement et "facilement" avancé sur le travail à faire. J'ai soulagé mes collègue pris par d'autre travaux et avancant sur ce sprint. J'ai surtout travaillé sur la strucutre général du programme, la création de compte et de player ainsi que les bases pour les items.

- Yohann Meyer :Temps prévu 0 heures: temps réalisé 1 heures.

Après une profonde inspection du code et une introspection de même, je me suis rendu compte que l'apport possible sur le code de ma part ne serait que marginal au mieux et négatif au pire. J'ai donc pris le parti de commenter le sprint et de me concentrer sur l'obtention d'une vision générale pointue.

- Loic Frueh : Temps prévu 4 heures, temps réalisé 0 heures.

  Pour ce sprint, je n'ai pas fait la moindre ligne de code.
  J'ai pris une petite semaine de vacances en ce qui concerne le projet en tout cas.

- Joel Schär : 3 heures, temps réalisé 7 heures.

  Cette semaine j'ai travaillé sur l'implémentation des items pour les joueurs.
  Je me suis rendu compte que le code que j'avais écrit il y a que quelques jours, je ne savais plus vraiment ce qu'il faisait et comment il fonctionnait.
  On remaque alors que sur une architecture un peu complexe, il est très facile de se perdre. La raison pour laquel nous arrivons dans cette situation, est que nous
  devons travaller rapidement pour arriver à terminer les sprints et ne prenons donc pas beaucoup de temps à rendre le code claire. Les étapes de refactorings du code
  qui devrait être fait après l'implémentation d'une fonctionnalité se révèleraient très utilie au final.
  Pour le reste je pense que nos sprints sont efficaces et que nous arrivons au resultat escompté à chaque fois, ce qui prouve que notre méthode de travail reste efficace.