# Projet Genie Logiciel

# OTrain

Par

Allemand Adrien, Amrani Kamil, Guidoux Vincent, Krug Loyse

Rapport

Version 4 : 04.05.2018

# Table des versions

| Version | Date |
|---|---|
| **1** | 13.04.2018 |
| **2** | 27.04.2018 |
| **3** | 30.04.2018 |
| **4** | 04.05.2018 |

# Table des matières

Introduction ..... 4
Descriptif du jeu ..... 4
Fonctionnement général ..... 4
Déplacement de train ..... 4
Extraction de ressources ..... 4
Fabrication d'objets ..... 4
Commerce ..... 4
Un jeu sur la durée ..... 5
Les acteurs ..... 5
Fonctionnalités secondaires ..... 5
Mockup de l'interface utilisateur ..... 6
Descriptions des scénarios possibles ..... 7
Se connecter ..... 7
S'inscrire ..... 7
Changer de gare ..... 7
Placer une offre ..... 7
Achat ..... 7
Miner ..... 7
Fabriquer ..... 7
Améliorer train ..... 8
Donner ressources ..... 8
Bannir un joueur ..... 8
Ajouter une gare ..... 8
Supprimer une gare ..... 8
Schema des cas d'utilisation ..... 9
Client-Serveur ..... 9
Responsabilités client-serveur ..... 9
Responsabilité client ..... 9
Responsabilité serveur ..... 9

Lancement et arret du jeu ....9
Protocole d'échange client serveur ....10
Modèle de domaine ....12
Base de données ....12
Distribution des rôles ....13
Backlogs de produit ....14
Stories IceScrum ....14
Plan d'itérations ....15
1er Sprint ....15
Objectif ....15
Stories à faire ....15
Tasks ....15
Tests d'acceptation ....15
Sprints à venir .... **Erreur ! Signet non défini.**
Sprint 2 ....17
Objectifs ....17
Sprint 3 ....19
Objectifs ....19
Sprint 4 ....19
Objectifs ....19
Sprint 5 ....19
Objectifs ....19
Sprint 6 ....19
Objectifs ....19
Sprint 7 ....20
Objectifs ....20
Conclusion ....20

# Introduction

Dans le cadre du cours de Génie Logiciel, il nous est demandé de réaliser un projet de programmation par groupe de 4 personnes. Il doit proposer un moyen de se connecter et aussi être assez important pour pouvoir utiliser les techniques de gestion de groupe et de travail vues en cours. Le logiciel doit avoir une base de données, deux programmes, un serveur et un client communiquant par Internet – sockets. Nous devons utiliser nos connaissances GIT et JUnit comme ciment à la maison qu'est notre projet. Nous avons décidé de faire un jeu de gestion avec une interface graphique textuelle.

# Descriptif du jeu

OTrain consiste en un jeu de gestion de trains-usine dans un univers post-apocalyptique où la collaboration est de mise pour survivre et atteindre l'apogée des technologies de manufacture et d'extraction de matière première.

## Fonctionnement général

OTrain se veut un jeu à interface graphique simple (O Game like), genre tableau de bord, avec des boutons d'action, du texte et d'éventuelles images illustratives.

le joueur pourra lancer des commandes qui auront une influence sur ses ressources et son usine. Ses actions pourront faire partie d'une des catégories suivantes : déplacement du train vers une autre gare, extraction de ressource, fabrication d'objets, commerce avec d'autres joueurs.

### Déplacement de train

Un joueur peut demander le déplacement d'une gare à une autre après avoir récupéré la liste des gares alentours. Son déplacement une fois validé, un compte à rebours est lancé pour lui indiquer sa progression vers sa destination. Un joueur se déplace de gare en gare dans l'objectif d'y trouver toujours de meilleures ressources.

### Extraction de ressources

A chaque gare, une certaine quantité de ressources est disponible et les trains s'y trouvant peuvent l'extraire. Pour récupérer de la matière première destinée à alimenter les machines, fabriquer de nouveaux objets ou améliorer son train. Une ressource est minée sur un certaine durée et le joueur peut voir dans la durée sa quantité de minerais augmenter.

### Fabrication d'objets

Avec ses ressources disponibles, le joueur peut décider de fabriquer des objets plus complexes. Cette fabrication consume des ressources et prends un certain temps pendant lequel le joueur peut lancer d'autres processus. L'objectif de fabrication d'objet est de pouvoir utiliser dans la durée des technologies de plus en plus puissantes pour faire évoluer son train.

### Commerce

Les ressources n'étant pas disponibles partout, un joueur seul pourra avoir de la peine à atteindre les technologies les plus évoluées. Pour cette raison, un espace de commerce central regroupe toutes les offres ou demandes qu'un joueur pourrait formuler. La monnaie du jeu est le Scrum

## Un jeu sur la durée

Au cours du temps et avec l'évolution des technologies, les actions du joueur voient leur durée augmenter drastiquement. Ainsi, il s'agit d'un jeu se déroulant sur la durée, demandant une connexion pour lancer une commande ou observer l'état de ses ressources. Une fois le joueur déconnecté, le serveur continue à traiter les requêtes qui ont été lancées, jusqu'à leur terminaison.

Le jeu se passe dans un univers pseudo-infini et ainsi ne connait ni niveaux, ni fin, l' « objectif » étant d'enrichir son train et l'améliorer sans cesses.

L'idée de OTrain est de favoriser la collaboration, ainsi un joueur avancera plus vite s'il se concentre principalement sur une activité (miner ou manufacturer) et qu'il utilise les mécanismes de commerce pour recevoir des ressources qu'il n'extrait ou ne fabrique pas lui-même.

## Les acteurs

**Joueur**

Un joueur est un utilisateur de l'application pour y avoir un train-usine à gérer. Ses actions sont limitées par les règles du jeu. Pour joueur, il doit posséder un compte. Il a la possibilité s'inscrire en tout temps.

**Admin**

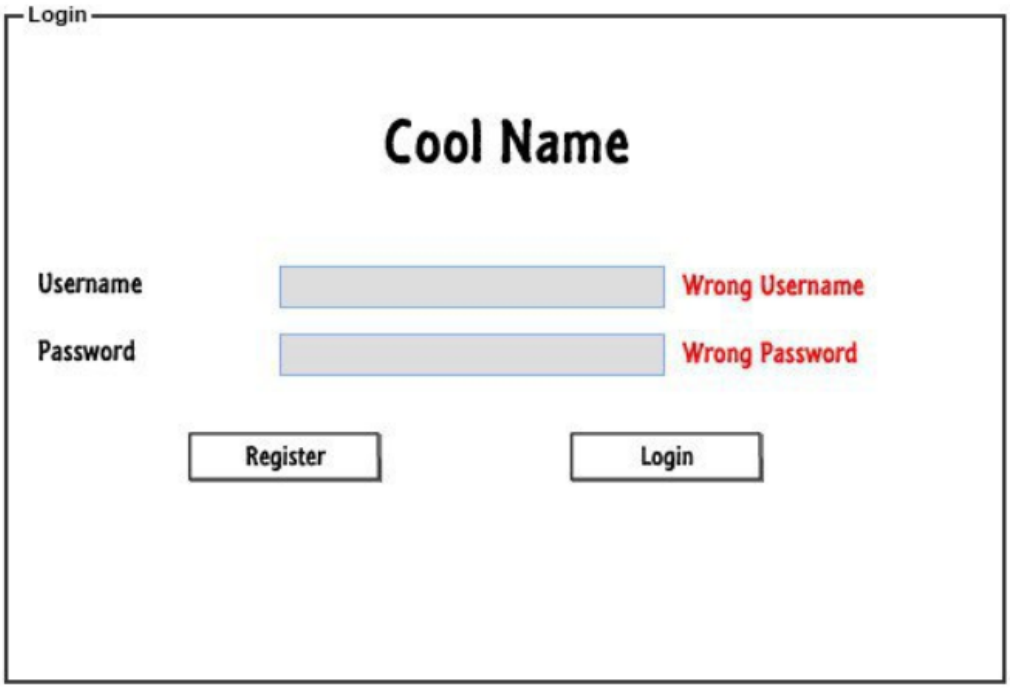
Un admin doit se connecter au compte administrateur du serveur. Il peut gérer les différents éléments du jeu : les gares, les ressources et les joueurs.

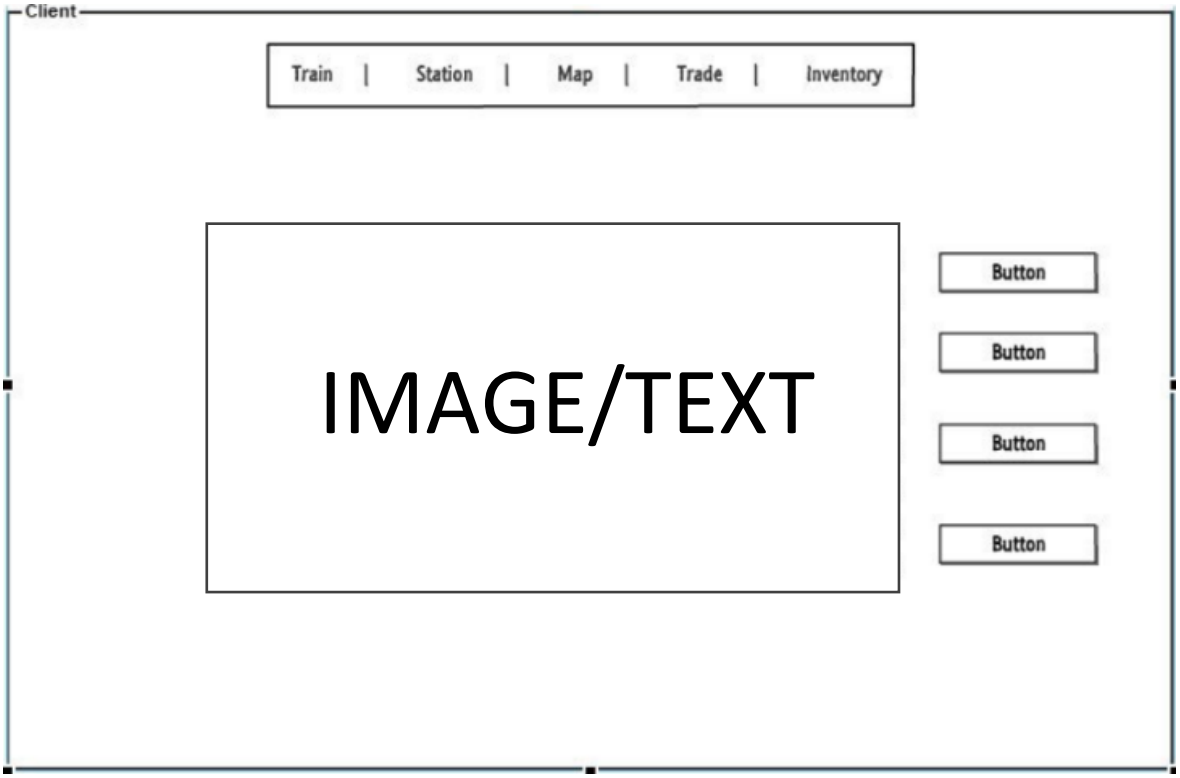
## Fonctionnalités secondaires

Ce jeu comporte des fonctionnalités de plusieurs degré d'importance. Les fonctionnalités principales sont celles décrites dans le fonctionnement général du jeu. Il sera néanmoins possible d'ajouter des éléments pour le diversifier, une fois sa première version terminée.

- Le joueur a une vue restreinte des gares autour de lui, plus elles sont proches de lui, plus il en connaît les détails
- Ajout d'éléments aléatoires ayant lieu lors des déplacements en train
- Création d'un marché propre à chaque gare, pour inviter au déplacement
- Les gares sont des points d'intérêts proposant de temps à autre des « missions » aux joueurs
- Des fabrications d'objets en plusieurs étapes. Un objet complexe demande la fabrication de petits objets au préalable. Si un joueur lance la production d'un objet complexe, c'est tout son processus de production qui est lancé (du clou à la loco). Le joueur pourrait alors demander un arrêt de fabrication et conserve les objets déjà créés et les ressources pas encore utilisées. (Système Factorio like)
- Permettre aux serveurs de se multiplier

## Mockup de l'interface utilisateur



# Descriptions des scénarios possibles

## Se connecter

- L'utilisateur va sur la page de login
- Il entre son nom d'utilisateur et son mot-de-passe
- Si son authentification fonctionne (il a un compte et son mot de passe est correct), il se retrouve envoyé sur son tableau de bord.

## S'inscrire

- Le joueur va sur la page de sign up
- Il entre ses données (nom d'utilisateur et mot de passe)
- Si ses informations sont acceptées, son compte est créé et il est envoyé sur la page de login pour se connecter.

## Changer de gare

- Un joueur demande la liste des gares au serveur
- Il choisit la gare à laquelle il veut se déplacer
- Si son mouvement est validé (une place dans la nouvelle gare lui est réservée) il entame son trajet.
- Au bout d'un certain temps, il arrive à destination.

## Placer une offre

- Un joueur consulte la liste de ses ressources (sur son tableau de bord)
- Il place une offre sur le marché central du jeu (un objet ou une ressource qu'il voudrait vendre)
- Si son offre est validée, le contenu de l'offre est retiré des ressources du joueur pour être placé dans une zone tampon.
- Si le joueur retire son offre avant sa vente, ses ressources lui sont rendues

## Achat

- Un joueur consulte la liste des offres sur le marché central du jeu
- Il fait une demande d'achat sur une offre
- Si sa demande est validée le contenu de l'offre est ajouté aux ressources du joueur
- Le prix de l'offre est retiré du budget du joueur
- Le prix de l'offre est ajouté au joueur qui a placé l'offre

## Miner

- Un joueur consulte la liste des ressources disponibles dans sa gare courante
- Il lance une demande de minage d'une ressource
- Si sa demande est acceptée, il commence à extraire la matière première
- Le serveur indique combien de ressources ont été minées après un certain laps de temps (cette information est renouvelée régulièrement pendant le minage)
- Les ressources extraites sont ajoutées aux ressources du joueur
- Les ressources extraites sont retirées de la gare

## Fabriquer

- Un joueur consulte la liste des objets qu'il peut fabriquer
- Il demande la fabrication d'un objet
- Si sa demande est validée, le processus de fabrication est enclanché
- Les ressources nécessaires à la fabrication sont retirées des ressources du joueur
- Après un certain temps l'objet est terminé et est ajouté aux ressources du joueur

### Améliorer train

- Un joueur consulte la liste des améliorations possibles de son train
- Il demande une amélioration d'une partie de son train.
- Si la demande est acceptée Les ressources nécessaires à l'amélioration sont retirées des ressources du joueur
- Après un certain temps, l'amélioration est terminée et est ajoutée au train

### Donner ressources

- Un admin consulte la liste des joueurs/gares
- Il sélectionne un joueur ou une gare
- Il indique la ressource à donner et sa quantité
- Les ressources sont ajoutées au joueur/à la gare

### Bannir un joueur

- Un admin consulte la liste des joueurs
- Il sélectionne le joueur à bannir
- Le compte du joueur est supprimé

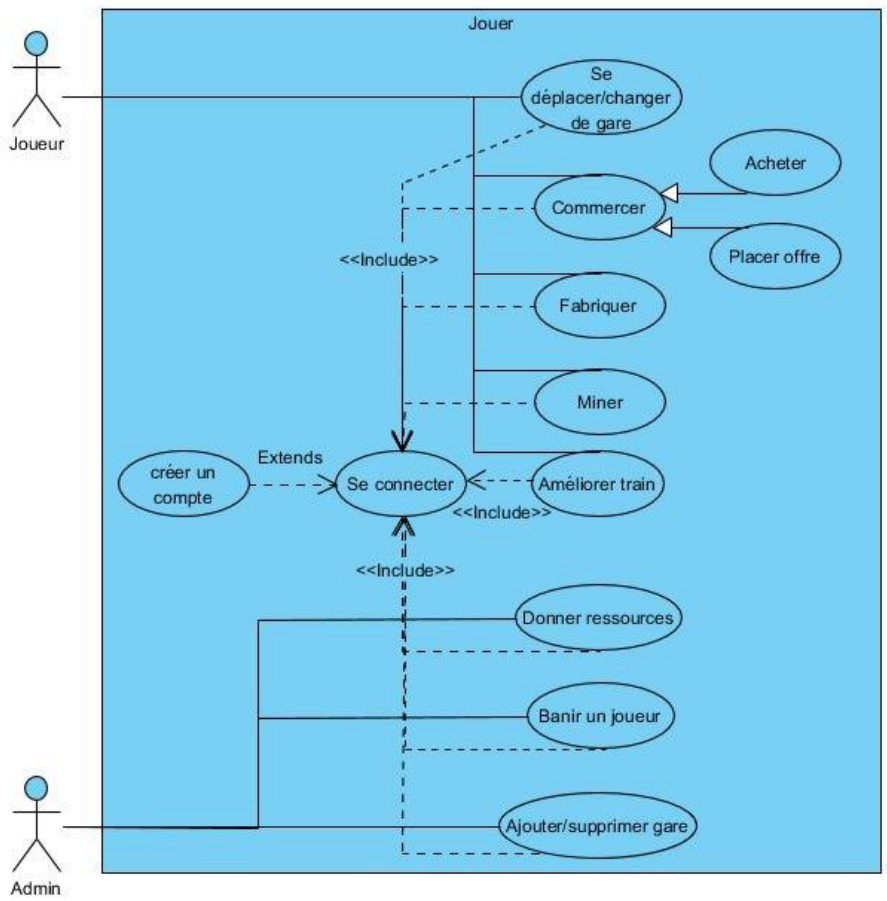
### Ajouter une gare

- Un admin va sur la page de création de gare
- Il entre les infos de la nouvelle gare
- La nouvelle gare est ajoutée à la liste des gares

### Supprimer une gare

- Un admin consulte la liste des gares du jeu
- Il sélectionne la gare à supprimer
- La gare est retirée de la liste des gares existantes

# Schema des cas d'utilisation




# Client-Serveur

## Responsabilités client-serveur

### Responsabilité client

Le client est un observateur contrôleur du serveur. Il va récupérer des informations et envoyer des requêtes pour effectuer des actions avec son train.

### Responsabilité serveur

Le serveur récupère et valide les requêtes du client. Il s'occupe de faire tous les calculs et contrôles. Il est le seul à avoir accès à la base de données.

### Lancement et arret du jeu

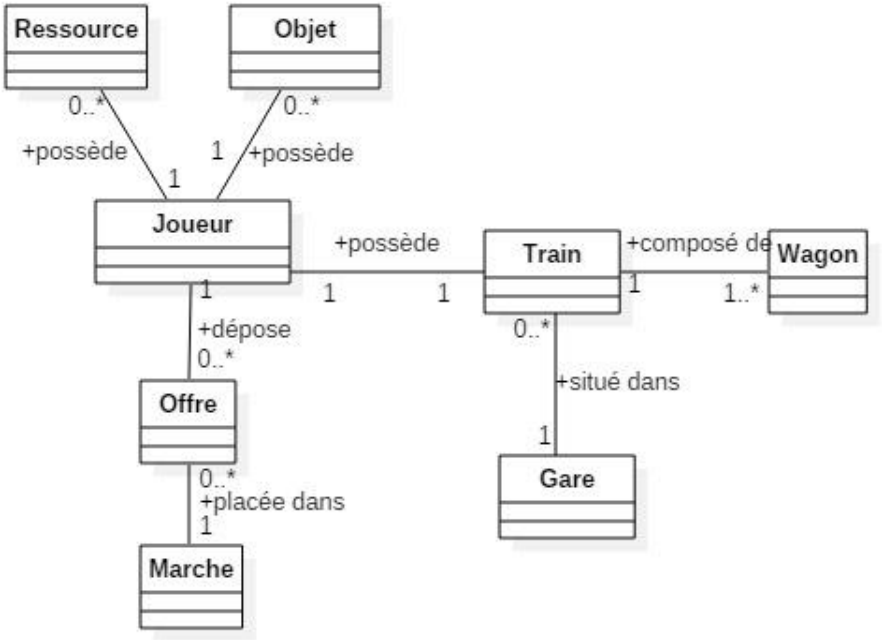
Un client démarre le jeu côté client en se connectant, et le quitte en se déconnectant. Du coté serveur le jeu tourne en continu, tant que le serveur est lancé.

## Protocole d'échange client serveur

| No | Description CLI | Client Command | Server Answer | Description SRV |
|---|---|---|---|---|
| | | **Changer de Gare** | | |
| 1 | Demander les gares disponibles | GET_GARES | | |
| 2 | | | JSON<Gares> | Une liste de gare Jsonifiées |
| 3 | La gare de destination | GO_TO <id gare> | | |
| 4 | | | SUCCESS \|\|FAILURE | Si la demande est valide, le serveur initie le déplacement et réponds SUCCESS, sinon il réponds FAILURE |
| | | **Etat du train** | | |
| 1 | Demander l'état du train | GET_TRAIN_STATUS | | |
| 2 | | | AT_STATION<br>JSON <la gare actuelle> | Si le train est a une gare sans etre partis<br>la gare ou il est |
| | | | ON_THE_WAY<br><temps restant en secondes> | si le train est en route<br>le temps avant l'arrivée |
| | | | ARRIVED<br><br>JSON<Evenement> | Si le train est arrivé a destination aprés un déplacement<br><br>Le résultat du voyage |
| | | **Placer une offre** | | |
| 1 | Demander de placer une offre<br>l'offre en question | SET_OFFRE<br>JSON<Offre> | | |
| 2 | | | SUCCESS \|\|FAILURE | Si la demande est valide, le serveur initie le placement de l'offre et réponds SUCCESS, sinon il réponds FAILURE |
| | | **Acheter une ressource** | | |
| 1 | Demander la liste des offres | GET_OFFRES | | |
| 2 | | | JSON<Offres> | retourne la liste des offres |
| 3 | Choisir une offre a acheter | BUY_OFFRE <id offre> | | |

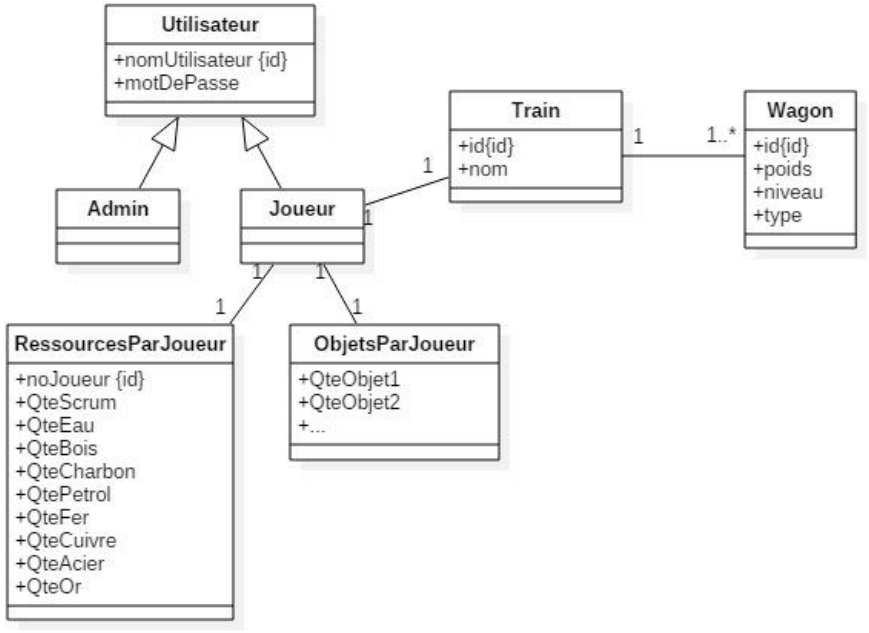
| | | | | |
|---|---|---|---|---|
| 4 | | | SUCCESS \|\|FAILURE | Si la demande est valide, le serveur transfert les ressources et réponds SUCCESS, sinon il réponds FAILURE |
| **Consulter ses ressources** | | | | |
| 1 | Demande l'état de ses ressources | GET_RESSOURCES | | |
| 2 | | | JSON<Ressources> | retourne la liste des offres |
| **Miner** | | | | |
| 1 | Demande de démarrer le minage d'une ressource<br>la ressource qu'on désire miner | MINE<br><id ressource> | | |
| 2 | | | SUCCESS \|\|FAILURE | Reponse que le minage a pu être lancé |
| 3 | demande d'arrêter le minage | STOP_MINE | | |
| 4 | | | SUCCESS \|\|FAILURE | Reponse que le minage a pu être arrêté |
| **Fabrication** | | | | |
| 1 | Demande l'état de la queue de production | | | |
| 2 | | | JSON<Queue de production> | Retourne la queue de production |
| 3 | Demande l'ajout d'un craft | CRAFT <id objet> <quantité> | | |
| 4 | | | SUCCESS \|\|FAILURE | Retourne si le craft a pu être ajouté à la queue |
| 5 | Demande l'annulation d'un craft | CANCEL_CRAFT <id groupe de prod> | | |
| 6 | | | SUCCESS \|\|FAILURE | Retourne si le craft a pu être annulé |
| **Connexion** | | | | |
| 1 | Demande de connection authentifiée | CONNECT <nom> <mdp> | | |
| 2 | | | SUCCESS \|\|FAILURE | Retourne si la connection a fonctionné |

## Modèle de domaine




## Base de données

La base de données du jeu devra contenir les informations sur les joueurs, les ressources et les objets contenus dans le jeu. Elle doit permettre au serveur d'aller vérifier l'état d'un joueur avant de lui autoriser une action. C'est également dans la base de données que seront contenu les informations destinées à l'authentification du joueur.



# Distribution des rôles

**Membres de l'équipe :** Allemand Adrien, Amrani Kamil, Guidoux Vincent, Krug Loyse

**Scrum master :** Kamil Amrani

**Project Owner :** Vincent Guidoux

**Equipe de développement :** Allemand Adrien, Amrani Kamil, Guidoux Vincent, Krug Loyse

**Testeurs :** Allemand Adrien, Amrani Kamil, Guidoux Vincent, Krug Loyse

# Backlogs de produit

## Stories IceScrum

| Rank | Name | description | type |
|---|---|---|---|
| 1 | Se connecter | Le joueur va sur la page de login, entre son nom d'utilisateur et son mot de passe pour se connecter. Si ses identifiants sont corrects il est envoyé sur son tableau de bord | Technique |
| 2 | Changement de gare | Le joueur demande à se déplacer jusqu'à une gare. Si c'est possible, le déplacement est lancé, sinon, il reste où il est. | Utilisateur |
| 3 | Consulter ses ressources | Le joueur peut voir ses ressources sur un tableau de bord | Utilisateur |
| 4 | Miner | Le joueur déploie son materiel de minage et commence à extraire une ressource dans la gare où il se trouve. Son minage s'arrête quand la ressource est épuisée ou s'il a donné l'ordre d'arrêt | Utilisateur |
| 5 | Fabriquer | Le joueur crée de nouvelles ressources à partir de ses ressources actuelles | Utilisateur |
| 6 | Améliorer son train | Le joueur peut utiliser des ressources pour améliorer des parties de son train. | Utilisateur |
| 7 | Ajouter une gare | L'administrateur peut ajouter une gare dans la partie | Utilisateur |
| 8 | Supprimer une gare | l'admin peut supprimer un gare du jeu | Utilisateur |
| 9 | Donner des ressources | l'administrateur donne des ressources à un joueur ou à une gare | Utilisateur |
| 10 | Bannir un joueur | L'administrateur peut bannir un  joueur du jeu | Utilisateur |
| 11 | Placer une offre | Le joueur pose une offre sur le marché | Utilisateur |
| 12 | Acheter des ressources | Le joueur peut voir la liste des offres et en acheter, s'il en a les moyens. | Utilisateur |

# Plan d'itérations

## 1er Sprint

### Objectif

Pouvoir se connecter au serveur

### Stories à faire

Se connecter

### Tasks

VG : Vincent Guidoux, KA : Kamil Amrani, AA : Adrien Allemand, LK : Loyse Krug


### Tests d'acceptation


Estimation par tâche [heure/personne] :

| **No tâche** | 30 | 2 | 1 | 3 | 4 |
|---|---|---|---|---|---|
| **Durée[heure/personne]** | 0.5 | 4 | 3 | 3 | 4 |

### Bilan d'itération

*Bilan Scrum master*

### Bilan de terminaison des histoires

Toutes les histoires prévues ont été terminées

### Vélocité du sprint

8 points d'histoire

### Replanification

Pour le 2ème sprint, il faudra lier la base de données à l'application. De plus on ajoute une story pour la création d'une interface graphique permettant de s'inscrire dans la base de données et de se connecter.

La story de changement de gare est déplacée au sprint 3 pour donner le temps de finaliser la connexion avec la base de données.

### Commentaire général

Le projet avance bien.

### Autocritique

-

## *Bilan personnels*

**Adrien Allemand**

Temps prévu: 4h pour réaliser le protocole

Temps réalisé : 1h30

Commentaire : -

**Kamil Amrani**

Temps prévu: 4h pour réaliser l'implémentation du serveur

Temps réalisé : environ 4h

Commentaire : Beaucoup de debug à la fin du sprint

**Vincent Guidoux**

Temps prévu: 4h pour réaliser l'implémentation du client - 0.5h pour la mise en place du projet

Temps réalisé : envrion 2h

Commentaire : Merci à Kamil pour l'aide

**Loyse Krug**

Temps prévu: 3h pour réaliser la base de donnée

Temps réalisé : 2h30

Commentaire : -

## Sprint 2

### Objectifs

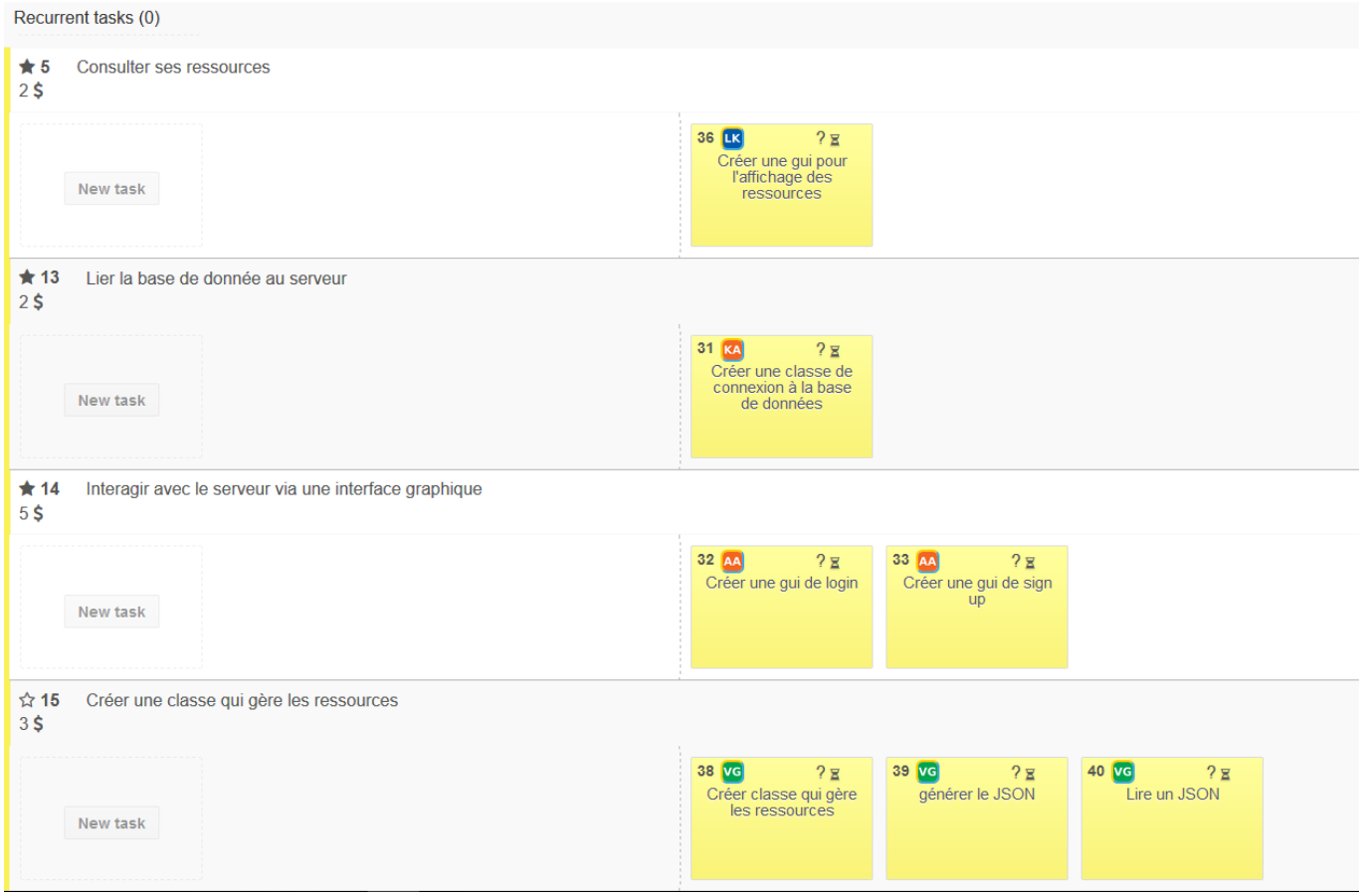
Pouvoir changer de gare et consulter ses ressources

### Stories à faire

Consulter ses ressources, lier la base de données avec le serveur, interagir avec le serveur via une interface graphique

### Tasks

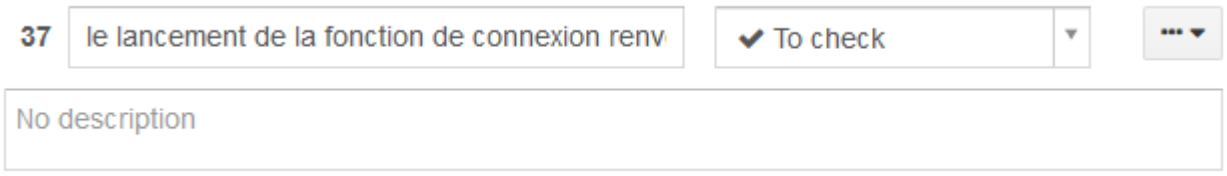
VG : Vincent Guidoux, KA : Kamil Amrani, AA : Adrien Allemand, LK : Loyse Krug



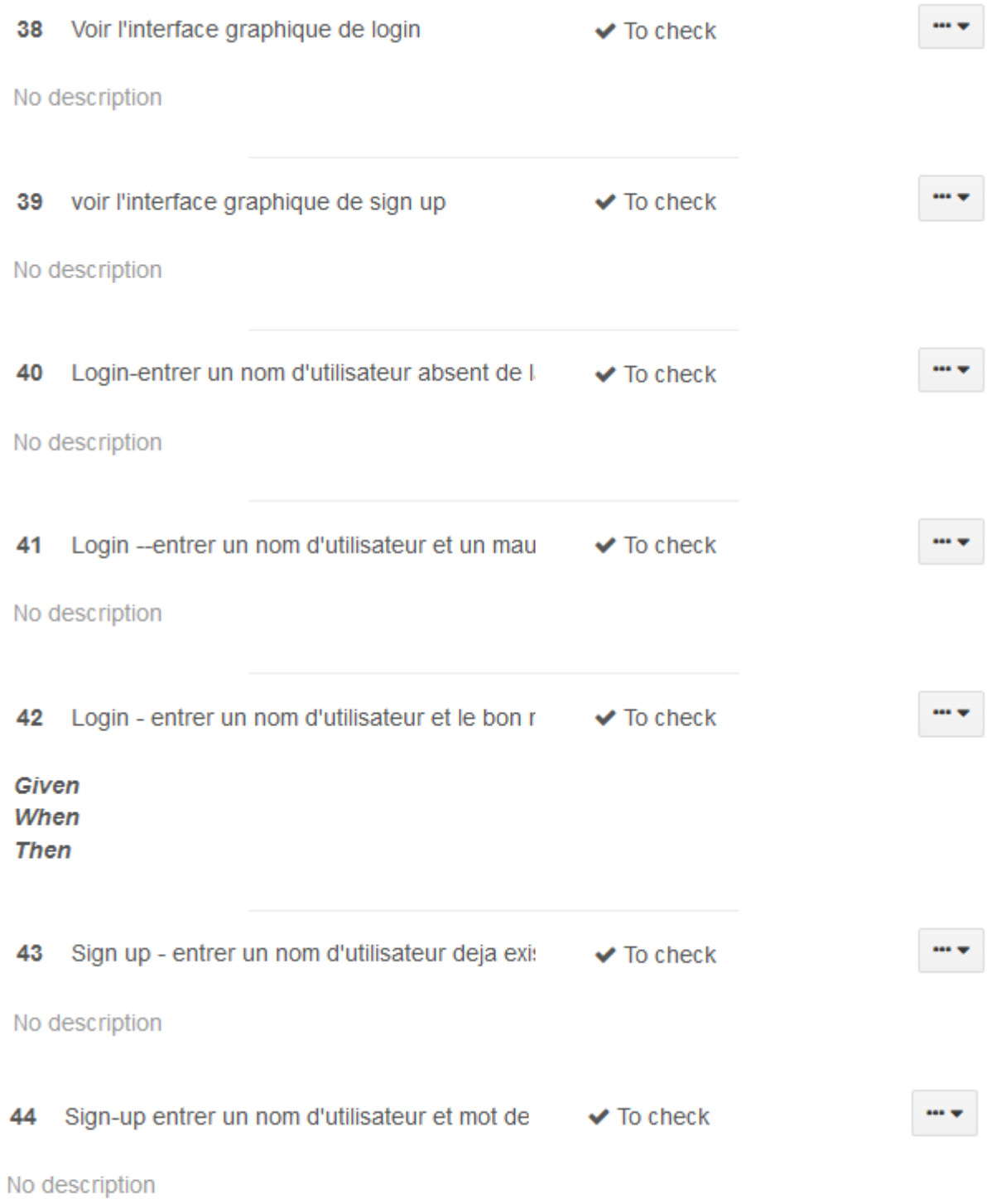

### Tests d'acceptation

Consulter ses ressources :

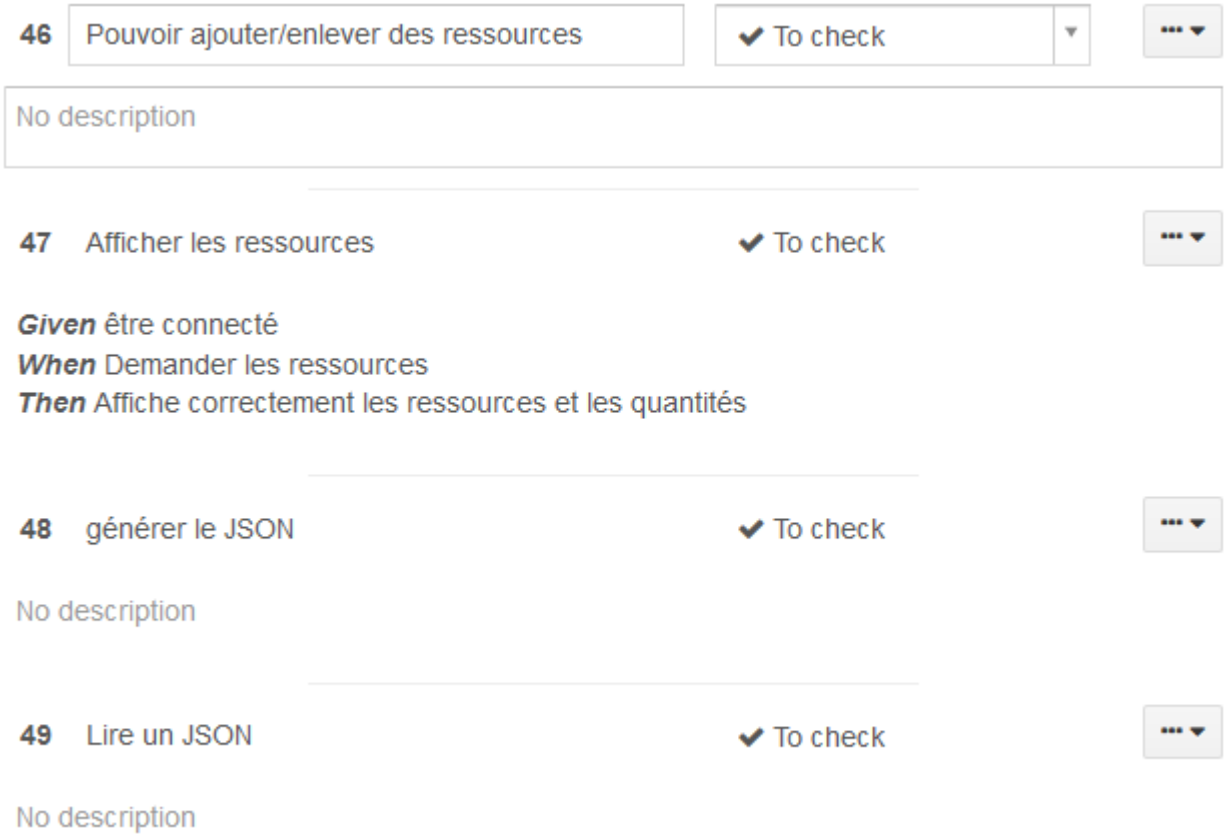
Lier la base de données au serveur :




Interagir avec ke serveur via une interface graphique :




Créer une classe qui gère les ressources :




Estimation par tâche [heure/personne] :

| **No tâche** | 36 | 31 | 32 | 33 | 38 | 39 | 40 |
|---|---|---|---|---|---|---|---|
| **Durée[h/p]** | 3 | 3 | 2 | 1 | 1 | 1 | 1 |

## Sprint 3

### Objectifs

Pouvoir miner/récolter des ressources

## Sprint 4

### Objectifs

pouvoir crafter des ressources, pour en créer de nouvelles

## Sprint 5

### Objectifs

Pouvoir améliorer le niveau de son train et l'administrateur peut créer une gare

## Sprint 6

### Objectifs

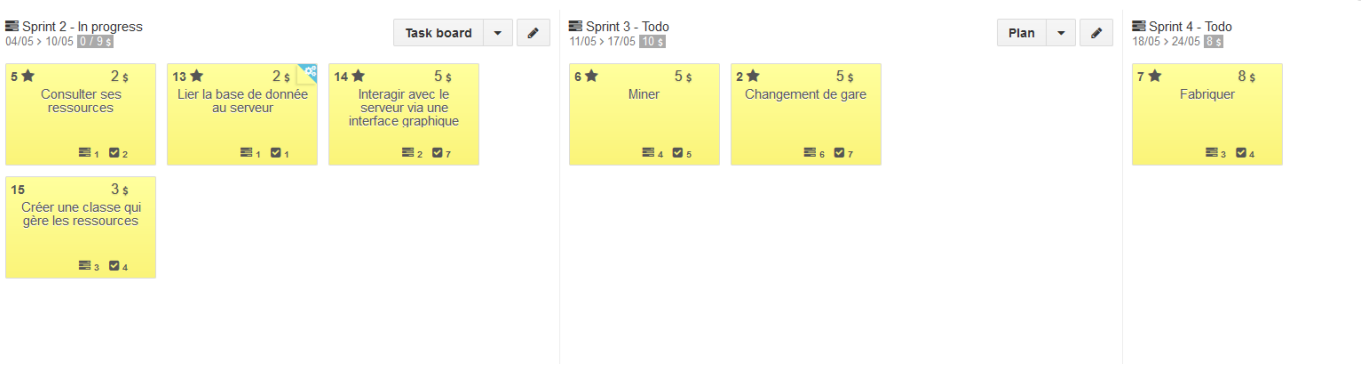
Finir actions à disposition des administrateurs

## Sprint 7

### Objectifs

Mettre en place le marché entre les joueurs

## Sprints à venir




# Conclusion

Cette étape nous a permis de mettre en place les fondations de notre petite maison que sera le logiciel final, mettre d'accord tout le monde sur le fil rouge principal et surtout, effacer le mirage que nous avions en tête pour la fin du projet par une image plus claire mais qui va sûrement changer en cours de route. Les rôles se forment gentiment dans le groupe, nous nous connaissions déjà alors la moitié du travail de collaboration est déjà fait. Nous avons fait cette étape assez efficacement en utilisant les techniques agiles vues en cours. Tout c'est bien déroulé et dans un temps raisonnable.