Projet Genie Logiciel

OTrain

Par

Allemand Adrien, Amrani Kamil, Guidoux Vincent, Krug Loyse

Rapport

Version 9 : 08.06.2018

Table des versions

|  |  |
| --- | --- |
| Version | Date |
| 1 | 13.04.2018 |
| 2 | 27.04.2018 |
| 3 | 30.04.2018 |
| 4 | 04.05.2018 |
| 5 | 11.05.2018 |
| 6 | 18.05.2018 |
| 7 | 25.05.2018 |
| 8 | 01.06.2018 |
| 9 | 08.06.2018 |

Table des matières

[Introduction 5](#_Toc515620834)

[Descriptif du jeu 5](#_Toc515620835)

[Fonctionnement général 5](#_Toc515620836)

[Déplacement de train 5](#_Toc515620837)

[Extraction de ressources 5](#_Toc515620838)

[Fabrication d’objets 5](#_Toc515620839)

[Commerce 5](#_Toc515620840)

[Un jeu sur la durée 6](#_Toc515620841)

[Les acteurs 6](#_Toc515620842)

[Fonctionnalités secondaires 6](#_Toc515620843)

[Mockup de l’interface utilisateur 7](#_Toc515620844)

[Descriptions des scénarios possibles 8](#_Toc515620845)

[Se connecter 8](#_Toc515620846)

[S’inscrire 8](#_Toc515620847)

[Changer de gare 8](#_Toc515620848)

[Placer une offre 8](#_Toc515620849)

[Achat 8](#_Toc515620850)

[Miner 8](#_Toc515620851)

[Fabriquer 8](#_Toc515620852)

[Améliorer train 9](#_Toc515620853)

[Donner ressources 9](#_Toc515620854)

[Bannir un joueur 9](#_Toc515620855)

[Ajouter une gare 9](#_Toc515620856)

[Supprimer une gare 9](#_Toc515620857)

[Schema des cas d’utilisation 10](#_Toc515620858)

[Client-Serveur 10](#_Toc515620859)

[Responsabilités client-serveur 10](#_Toc515620860)

[Responsabilité client 10](#_Toc515620861)

[Responsabilité serveur 10](#_Toc515620862)

[Lancement et arrêt du jeu 10](#_Toc515620863)

[Protocole d’échange client serveur 11](#_Toc515620864)

[Modèle de domaine 13](#_Toc515620865)

[Base de données 13](#_Toc515620866)

[Distribution des rôles 14](#_Toc515620867)

[Backlogs de produit 15](#_Toc515620868)

[Stories IceScrum 15](#_Toc515620869)

[Plan d’itérations 16](#_Toc515620870)

[1er Sprint 16](#_Toc515620871)

[Objectif 16](#_Toc515620872)

[Stories à faire 16](#_Toc515620873)

[Tasks 16](#_Toc515620874)

[Tests d’acceptation 16](#_Toc515620875)

[Bilan d’itération 17](#_Toc515620876)

[Sprint 2 18](#_Toc515620877)

[Objectifs 18](#_Toc515620878)

[Stories à faire 18](#_Toc515620879)

[Tasks 18](#_Toc515620880)

[Tests d’acceptation 19](#_Toc515620881)

[Bilan d’itération 20](#_Toc515620882)

[Sprint 3 21](#_Toc515620883)

[Objectifs 21](#_Toc515620884)

[Stories à faire 21](#_Toc515620885)

[Tasks 21](#_Toc515620886)

[Tests d’acceptation 23](#_Toc515620887)

[Bilan d’itération 25](#_Toc515620888)

[Sprint 4 26](#_Toc515620889)

[Objectifs 26](#_Toc515620890)

[Stories à faire 26](#_Toc515620891)

[Tasks 26](#_Toc515620892)

[Test d’acceptation 27](#_Toc515620893)

[Bilan d’itération 28](#_Toc515620894)

[Sprint 5 29](#_Toc515620895)

[Objectifs 29](#_Toc515620896)

[Stories à faire 29](#_Toc515620897)

[Tasks 29](#_Toc515620898)

[Test d’acceptation 29](#_Toc515620899)

[Bilan d’itération 31](#_Toc515620900)

[Sprint 6 31](#_Toc515620901)

[Objectifs 31](#_Toc515620902)

[Stories à faire 32](#_Toc515620903)

[Tasks 32](#_Toc515620904)

[Test d’acceptation 33](#_Toc515620905)

[Bilan d’itération 35](#_Toc515620906)

[Sprint 7 35](#_Toc515620907)

[Objectifs 35](#_Toc515620908)

[Stories à faire 35](#_Toc515620909)

[Tasks 35](#_Toc515620910)

[Test d’acceptation 35](#_Toc515620911)

[Bilan d’itération 35](#_Toc515620912)

[Sprints à venir 36](#_Toc515620913)

[Conclusion 36](#_Toc515620914)

# 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 peut lancer des commandes qui ont une influence sur ses ressources et son usine. Ses actions peuvent 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. Les joueurs s’y trouvant peuvent l’extraire pour récupérer de la matière première destinée à alimenter les machines, fabriquer de nouveaux objets ou améliorer leur train. Une ressource est minée sur une certaine durée et le joueur peut voir dans la durée sa quantité de minerais augmenter en parallèle de mine qui se vide. Chaque mine se régénère dans le temps, beaucoup plus lentement que la vitesse de minage.

### 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

En se créant un compte, un joueur choisit un type de wagon de départ. Il peut alors soit miner des ressources, soit couper du bois, soit pomper de l’eau et du pétrole. Seulement, pour fabriquer des objets, il a très vite besoin de tous les types de ressources. Ainsi, il doit collaborer avec d’autres joueurs pour 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. Les joueurs y déposent des offres en proposant une certaine quantité d’une ressource à échanger contre une autre ressource.

## 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**

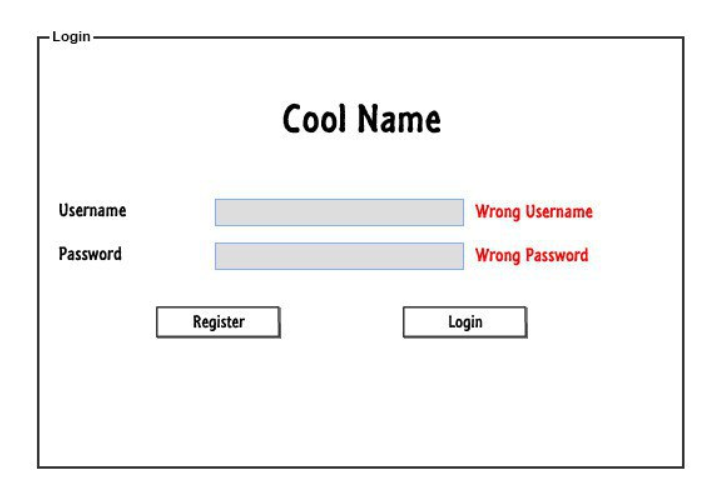
Un admin doit se connecter au compte administrateur du serveur. Il peut gérer les différents éléments du jeu : les gares, les ressources et les joueurs.

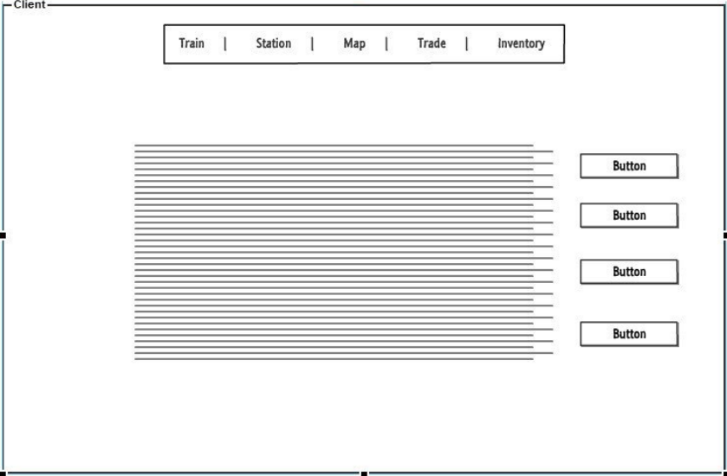
## Fonctionnalités secondaires

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





TEXTE

# 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

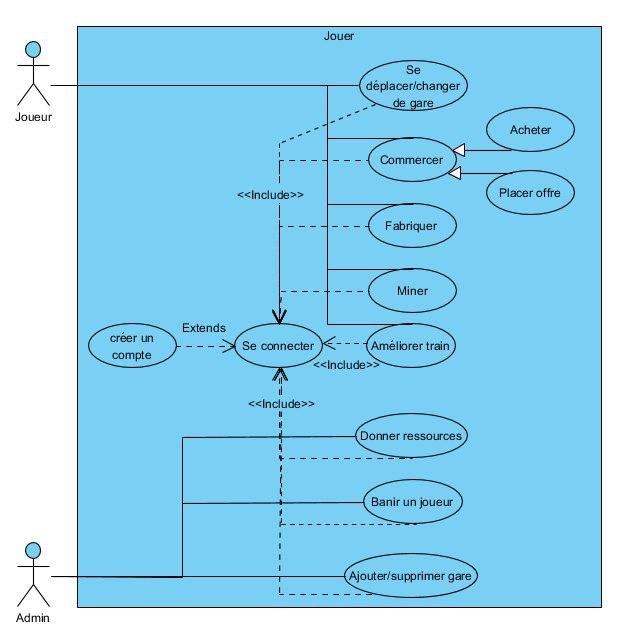
## Ajouter une gare

* 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 arrêt du jeu

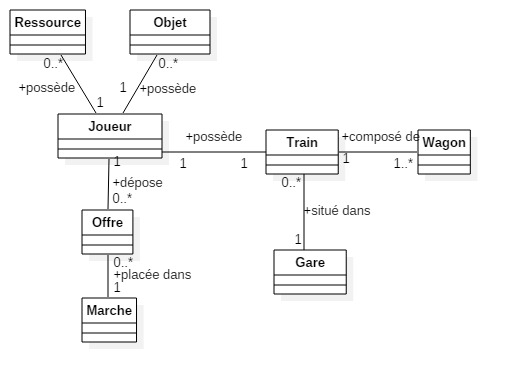
Un client démarre le jeu côté client en se connectant, et le quitte en se déconnectant. Du coté serveur le jeu tourne en continu, tant que le serveur est lancé.

## 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 | Si le train est a une gare sans être partis |
| JSON <la gare actuelle> | la gare ou il est |
| ON\_THE\_WAY | si le train est en route |
| <temps restant en secondes> | le temps avant l'arrivée |
| ARRIVED | Si le train est arrivé a destination après un déplacement |
| JSON<Evenement> | Le résultat du voyage |
|  | **Placer une offre** | | | |
| 1 | Demander de placer une offre | SET\_OFFRE |  |  |
| l'offre en question | 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 | MINE |  |  |
| la ressource qu'on désire miner | <id ressource> |
| 2 |  |  | SUCCESS ||FAILURE | Réponse que le minage a pu être lancé |
| 3 | demande d'arrêter le minage | STOP\_MINE |  |  |
| 4 |  |  | SUCCESS ||FAILURE | Réponse 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 connexion authentifiée | CONNECT <nom> <mdp> |  |  |
| 2 |  |  | SUCCESS ||FAILURE | Retourne si la connexion a fonctionné |

# Modèle de domaine

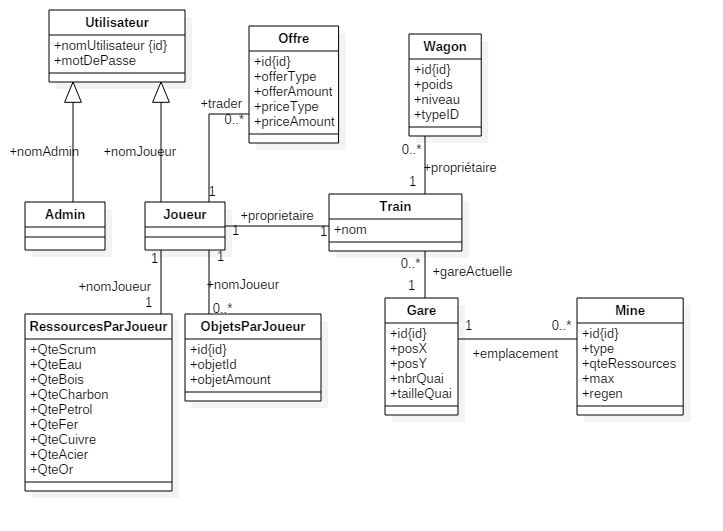
## Modèle de domaine côté client



## Modèle de domaine côté serveur

# 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



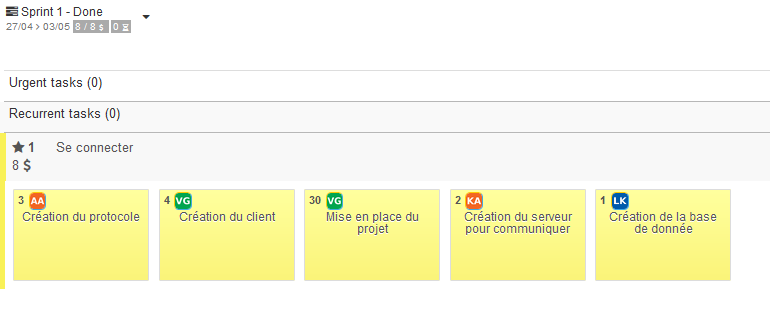
### Tests d’acceptation



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

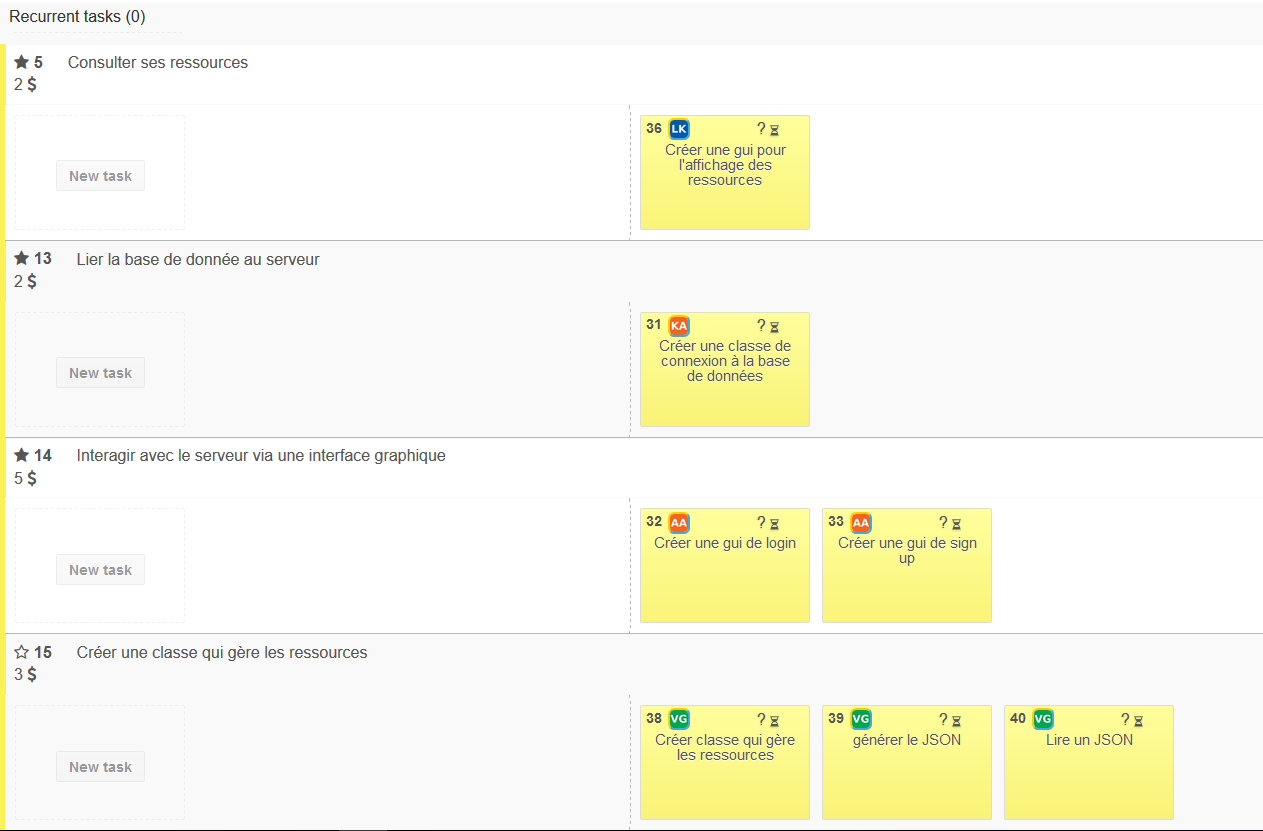
Pouvoir changer de gare et consulter ses ressources

### Stories à faire

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

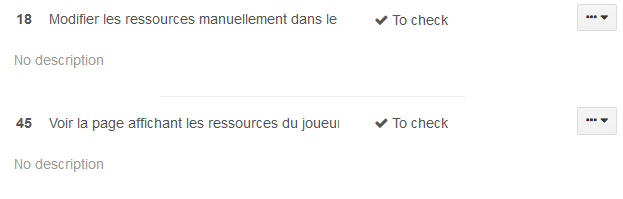
### Tasks

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

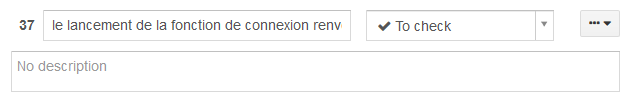


### Tests d’acceptation

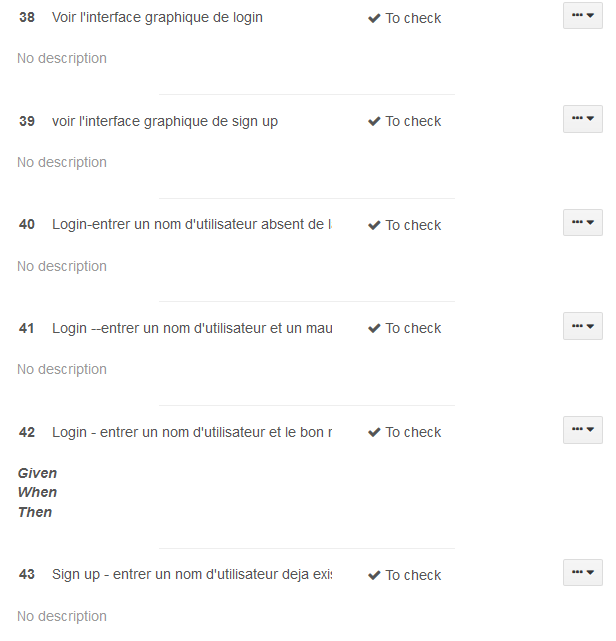
Consulter ses ressources :

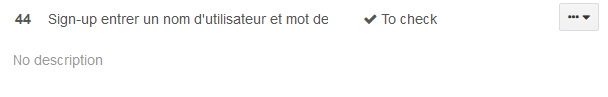


Lier la base de données au serveur :

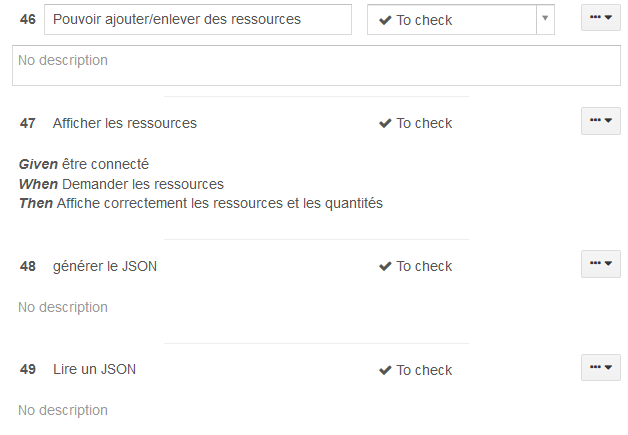


Interagir avec le serveur via une interface graphique :





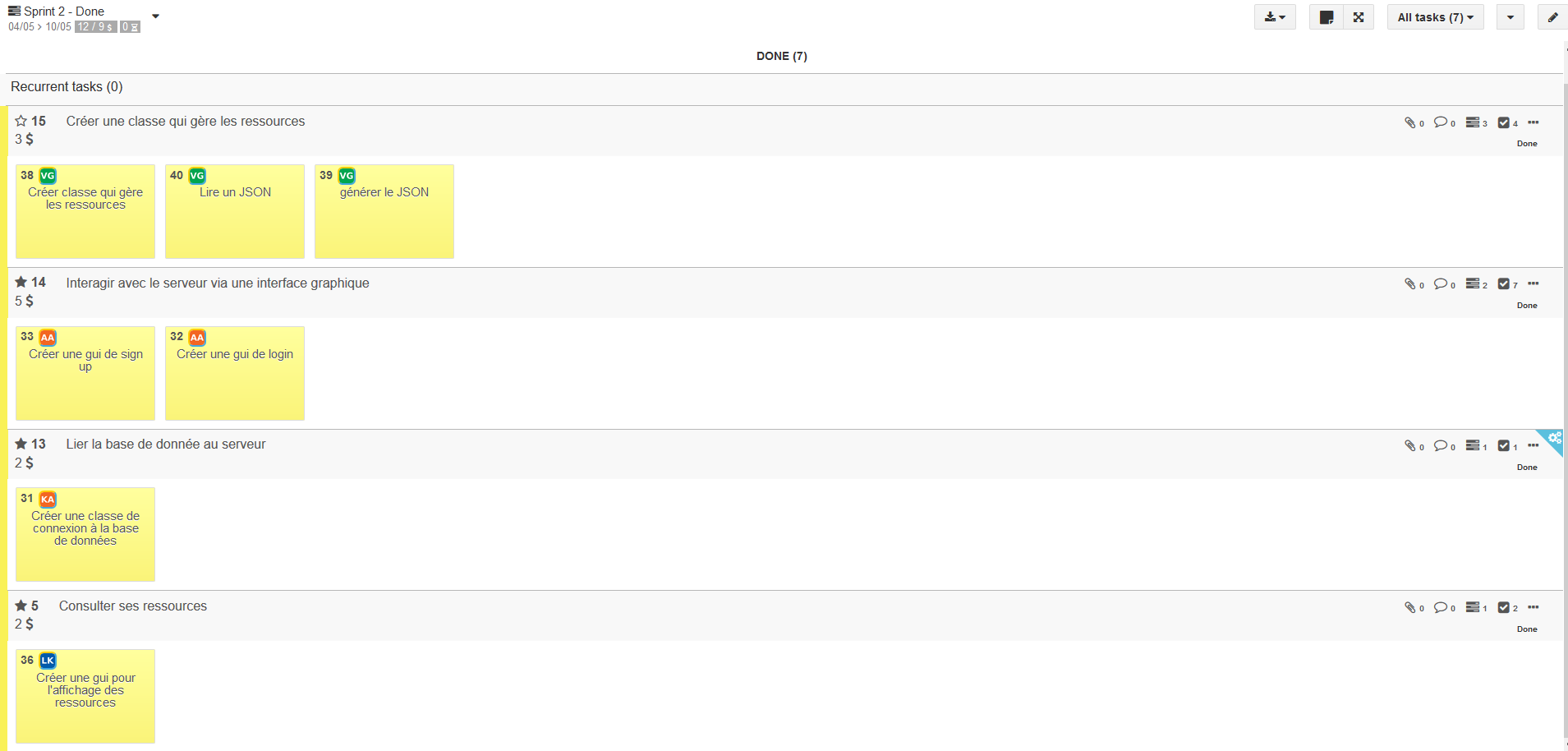
Créer une classe qui gère les ressources :



Estimation par tâche [heure/personne] :

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

### Bilan d’itération



#### Bilan Scrum master

##### Bilan de terminaison des histoires

Toutes les histoires prévues ont été terminées

##### Vélocité du sprint

12 points d’histoire

##### Replanification

Pour le 3ème sprint nous avons ajouté une story pour la finalisation de la GUI

##### Commentaire général

On a bien travaillé pour ce sprint, on a fini légèrement en avance.

##### Autocritique

-

#### Bilan personnels

**Adrien Allemand**

Temps prévu: 3h pour la gui du login et du sign up

Temps réalisé : 3

Commentaire : -

**Kamil Amrani**

Temps prévu: 3h pour créer une classe de connexion à la base de données

Temps réalisé : 3h

Commentaire :

**Vincent Guidoux**

Temps prévu:3h pour la classe qui gère les ressources, lire un JSON et générer un GSON

Temps réalisé : envrion 3h

Commentaire :

**Loyse Krug**

Temps prévu: 3h pour la création de la GUI de l’affichage des ressources

Temps réalisé : 3h

Commentaire : -

## Sprint 3

### Objectifs

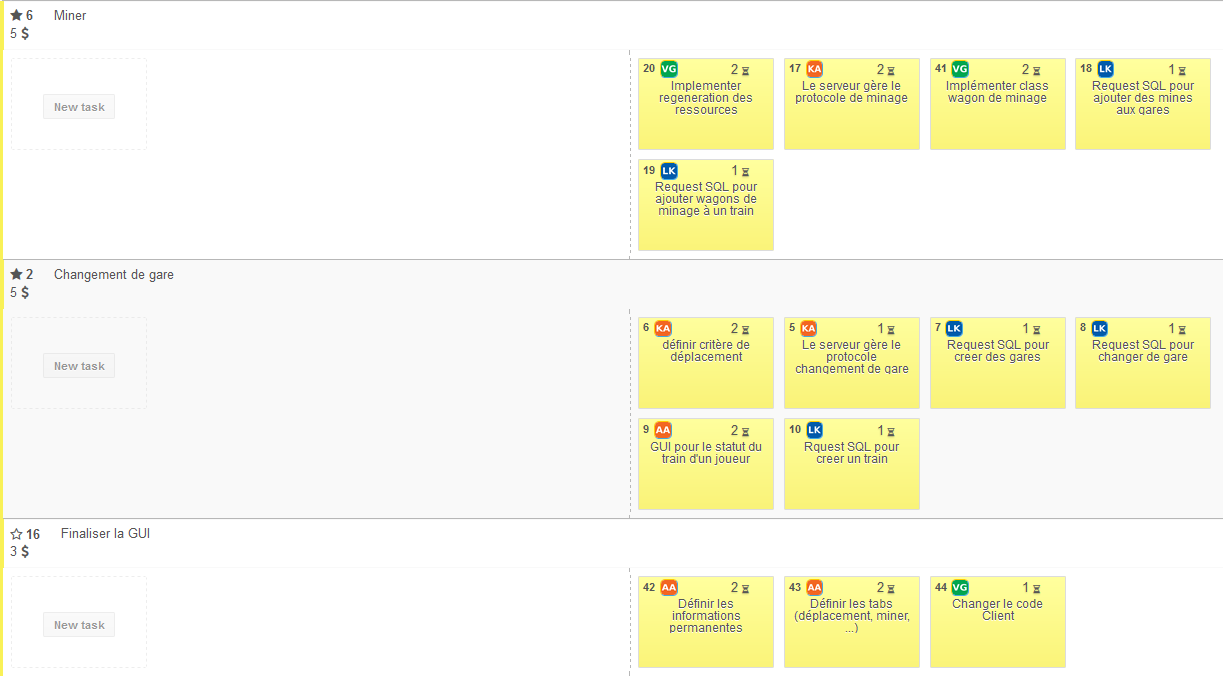
Pouvoir miner/récolter des ressources

### Stories à faire

Miner, Changement de gare, Finaliser la GUI

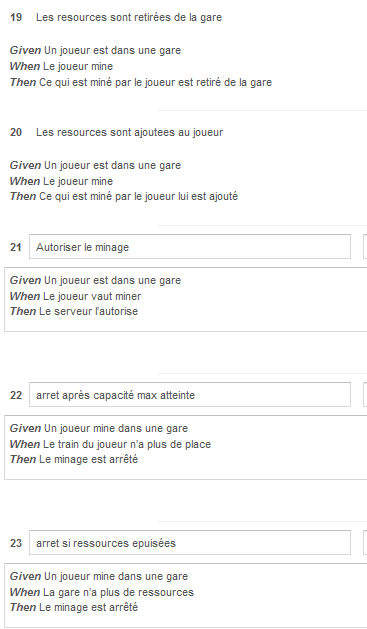
### Tasks

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

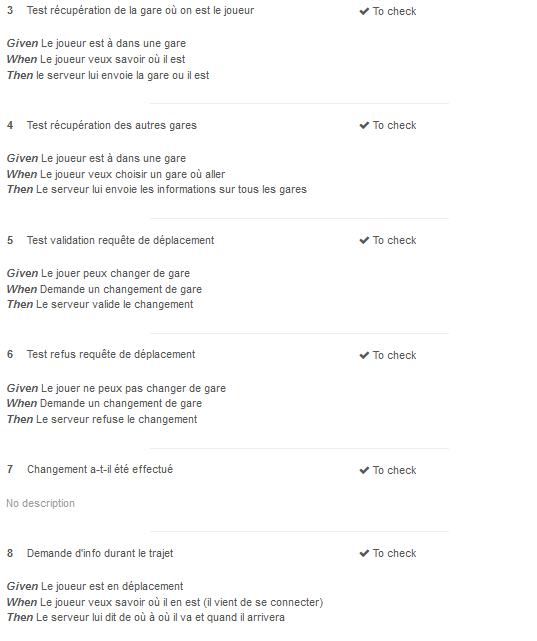


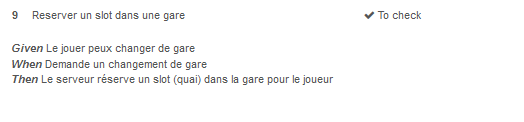
### Tests d’acceptation

Miner

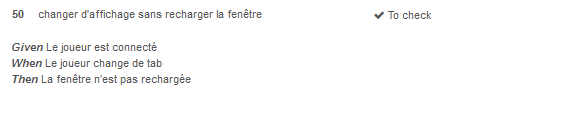


Changement de gare





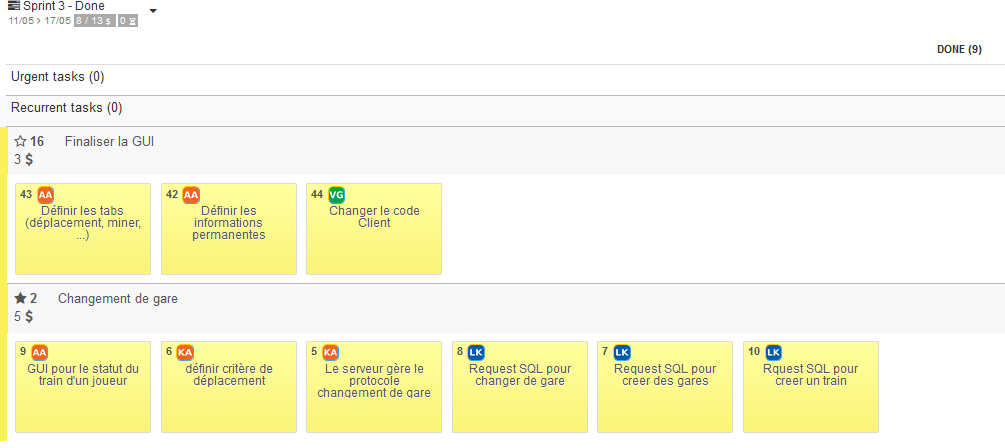
Finaliser la GUI



Estimation par tâche [heure/personne]  + bilan personnel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Numéro de tache | Personne assignée | Durée prévue [heure/homme] | Durée effective [heure/homme] | Commentaire |
| 17 | Kamil | 2 | 0 | Pas effectuée |
| 18 | Loyse | 1 | 1 | Requête écrite, mais pas encore de tests |
| 19 | Loyse | 1 | 1 | Requête écrite, mais pas encore de test |
| 20 | Vincent | 2 | 0 | Pas effectué |
| 41 | Vincent | 2 | 0 | Pas effectué |
| 5 | Kamil | 1 | 3 | Gestion du JSON en plus |
| 6 | Kamil | 2 | 2 | Ajout d’un contrôler |
| 7 | Loyse | 1 | 1 | Modification de la DB pour ajouter les gares |
| 8 | Loyse | 1 | 1 | Passé comme prévu |
| 9 | Adrien | 2 | 1 | Moins de travail que prévu |
| 10 | Loyse | 1 | 1 | Passé comme prévu |
| 42 | Adrien | 2 | 1 | - |
| 43 | Adrien | 2 | 1 | - |
| 44 | Vincent | 1 | 3 | Plus de temps que prévu (apprentissage swing) |

### Bilan d’itération



##### Bilan de terminaison des histoires

Histoires terminées :

* 16 : Finaliser GUI
* 2 : Changement de gare

En cours :

* 6 : Miner

##### Vélocité du sprint

8 points d’histoire

##### Replanification

L’histoire 6 (Miner) est déplacée au sprint suivant. L’histoire 7 (Fabriquer) est maintenue.

##### Commentaire général

A cause du travail des autres branches, nous n’avons pas pu terminer toutes les histoires. Nous avons enrichi la structure de la base de données. Nous n’avons pas implémenté suffisamment de tests.

##### Autocritique

Même si nous avions eu du temps supplémentaire, le planning était un peu ambitieux. Nous avons chacun pu travailler sur nos parties et avons réussi à les intégrer au projet sans trop de problèmes.

## Sprint 4

### Objectifs

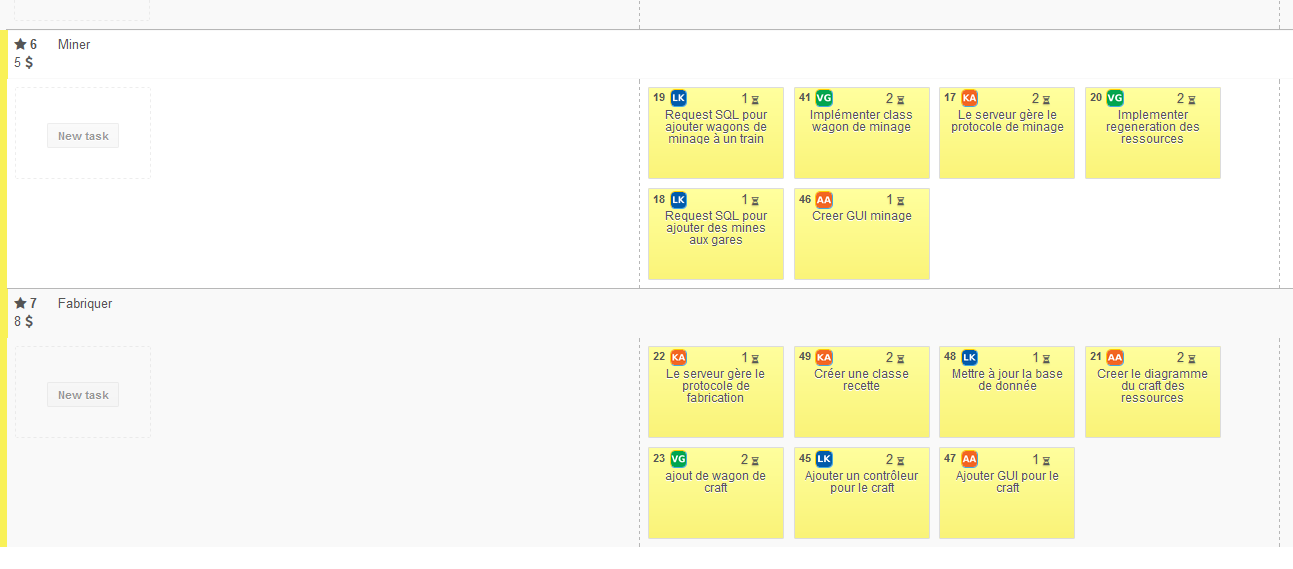
Implémenter le minage et la fabrication d’objets

### Stories à faire

6 :Miner , 7 : Fabriquer

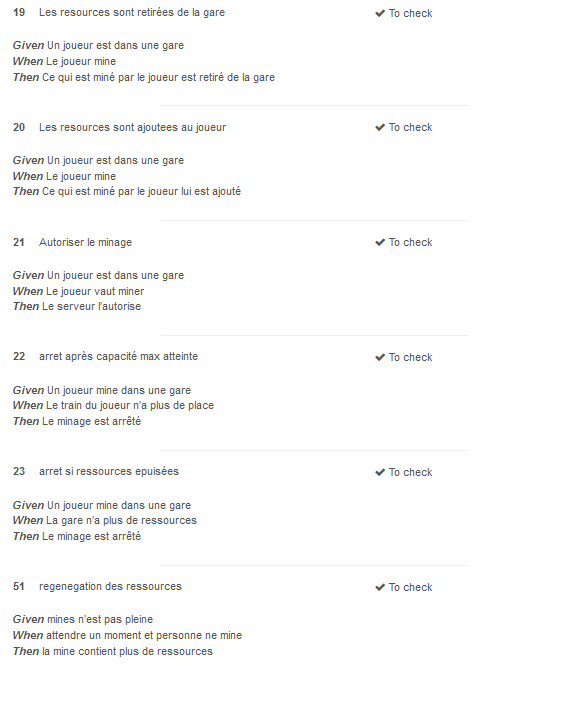
### Tasks

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



### Test d’acceptation

Miner :



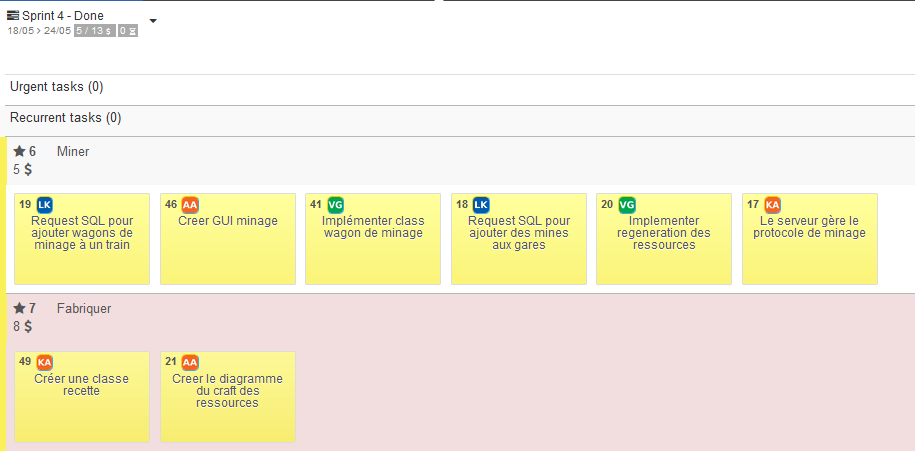
Fabriquer :



Estimation par tâche [heure/personne]  + bilan personnel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Numéro de tache | Personne assignée | Durée prévue [heure/homme] | Durée effective [heure/homme] | Commentaire |
| 17 | Kamil | 2 | 1 | On a l’habitude de faire évoluer la gestion du protocole par le serveur |
| 18 | Loyse | 1 | 1 | Mise en place des tests pour valider a mis du temps |
| 19 | Loyse | 1 | 1 | Mise en place des tests pour valider a mis du temps |
| 20 | Vincent | 2 | 2 | Les contrôleurs utilisent maintenant des timers |
| 21 | Adrien | 2 | 2 | Fait deux fois à cause d’un programme inadapté |
| 22 | Kamil | 1 | 1 | - |
| 23 | Vincent | 2 | 0 | Pas commencé, remplacé par des tests |
| 41 | Vincent | 2 | 4 | On n’utilise pas l’héritage pour les wagon, mais on stocke les infos dans WagonStats (nouvelle classe) |
| 45 | Loyse | 2 | 1 | Commencé mais pas eu le temps de terminer |
| 46 | Adrien | 1 | 1 | - |
| 47 | Adrien | 1 | 1 | - |
| 48 | Loyse | 1 | 0 | Pas eu le temps de commencer la tâche |
| 49 | Kamil | 2 | 1 | Plus facile que prévu |

### Bilan d’itération



#### Bilan de terminaison des histoires

Histoire terminée :

* 6 : Miner

Histoire en cours :

* 7 : fabriquer : faite à 20%

#### Vélocité du sprint

5 points d’histoire

#### Replanification

Nous avons dû déplacer l’histoire 7(Fabriquer) au sprint suivant. L’histoire 5 : Améliorer son train est maintenue pour le sprint 5, l’histoire 2 ajouter une gare, elle, est repoussée au sprint 6.

#### Commentaire général

Dû au rendu de PRO nous n’avons pas pu consacrer autant de temps que voulu au projet. La sérialisation globale dans le projet a été revue.

#### Autocritique

Chacun a travaillé dans son domaine (GUI, protocole, contrôleur, base de données) ce qui nous permet de bien paralléliser le travail.

## Sprint 5

### Objectifs

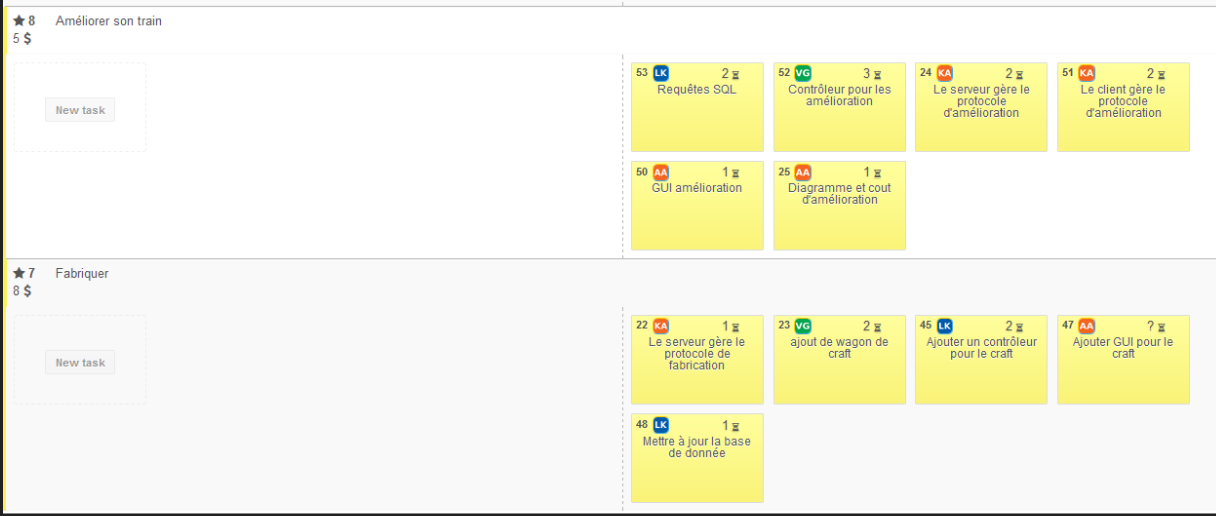
L’utilisateur doit pouvoir fabriquer des objets et améliorer les wagons de son train.

### Stories à faire

7 : Fabriquer, 8 : Améliorer son train

### Tasks

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

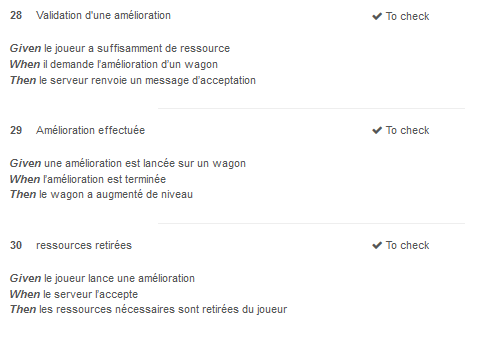


### Test d’acceptation

Fabriquer :



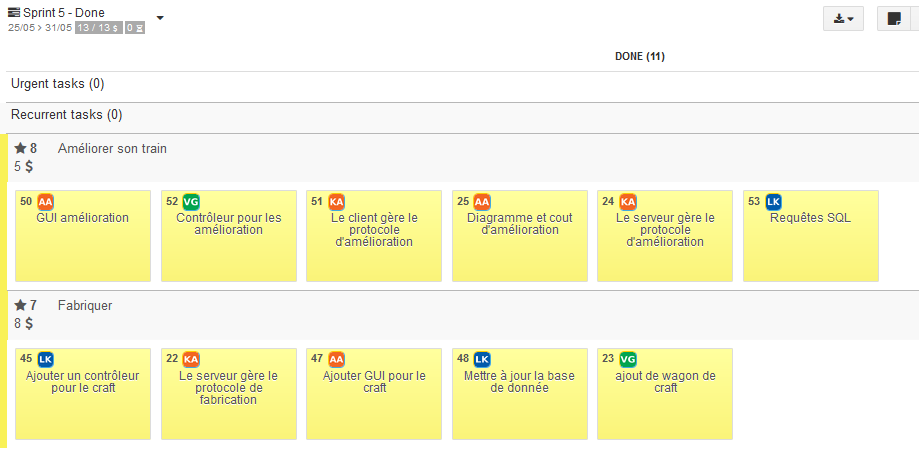
Améliorer son train :



Estimation par tâche [heure/personne]  + bilan personnel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Numéro de tâche | Personne assignée | Durée prévue [heure/homme] | Durée effective [heure/homme] | Commentaire |
| 22 | Kamil | 1 | 1 | - |
| 23 | Vincent | 2 | 1 | Plus simple que prévu |
| 24 | Kamil | 2 | 2 | - |
| 25 | Adrien | 1 | 3 | Equilibrage des ressources a pris beaucoup de temps |
| 45 | Loyse | 2 | 2 | - |
| 47 | Adrien | 1 | 1 | - |
| 48 | Loyse | 1 | 1 | - |
| 50 | Adrien | 1 | 1 | - |
| 51 | Kamil | 2 | 2 | - |
| 52 | Vincent | 3 | 2 | On a déjà beaucoup de contrôleurs, on commence à avoir l’habitude |
| 53 | Loyse | 2 | 1 | Beaucoup de requêtes déjà faites, on commence à avoir l’habitude |

### Bilan d’itération



#### Bilan de terminaison des histoires

Histoires terminées :

* 7 : Fabriquer
* 8 : Améliorer son train

#### Vélocité du sprint

13 points d’histoire

#### Replanification

Pas de replanification prévue

#### Commentaire général

On a rattrapé le retard. On commence à pouvoir jouer au jeu. Les lambda expression c’est la vie.

#### Autocritique

Chacun se sent de plus en plus investi, et les choses avance plus facilement.

## Sprint 6

### Objectifs

Ajout du pan admin au jeu. Un admin doit pouvoir ajouter et supprimer des éléments du serveur

### Stories à faire

9 : Ajout/Suppression des ressources

10 : Bannir un joueur

11 : Ajouter/Supprimer une mine

12 : Ajouter/Supprimer une gare

### Tasks

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

Ajout/Suppression des ressources



Bannir un joueur



Ajouter/Supprimer une mine

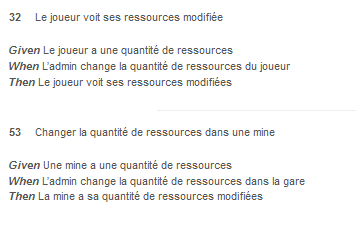


Ajouter/Supprimer une gare

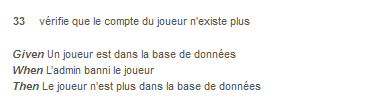


### Test d’acceptation

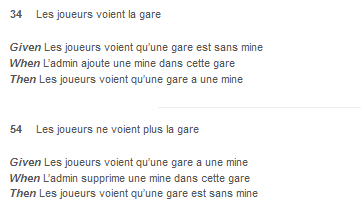
Ajout/Suppression des ressources



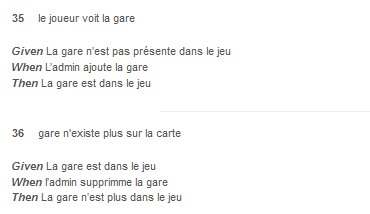
Bannir un joueur



Ajouter/Supprimer une mine



Ajouter/Supprimer une gare

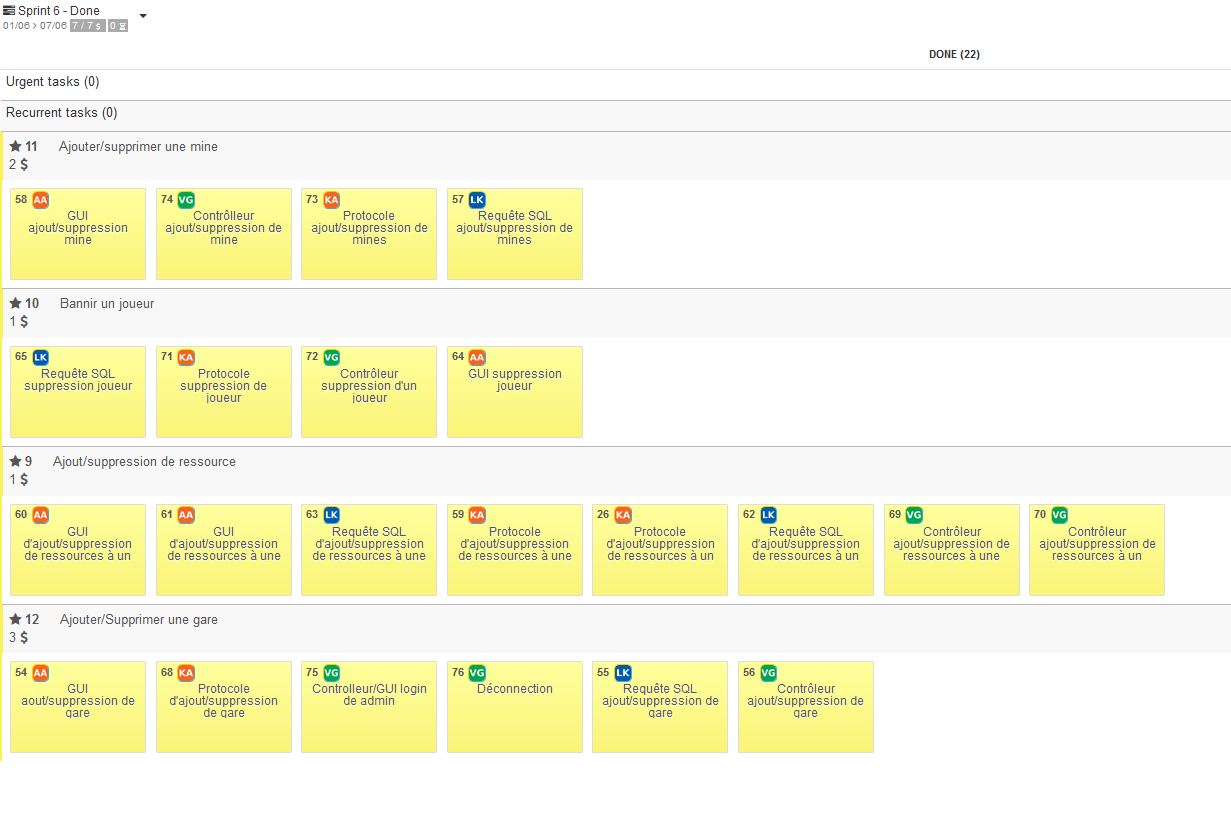


Estimation par tâche [heure/personne]  + bilan personnel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Numéro de tâche | Personne assignée | Durée prévue [heure/homme] | Durée effective [heure/homme] | Commentaire |
| 60 | Adrien | 1 | 0.5 | On a l’habitude de faire ça |
| 61 | Adrien | 1 | 0.5 | On a l’habitude de faire ça |
| 62 | Loyse | 1 | 1 | - |
| 63 | Loyse | 0.5 | 0.5 | - |
| 26 | Kamil | 0.5 | 0.5 | - |
| 59 | Kamil | 0.5 | 0.5 | - |
| 69 | Vincent | 1 | 0.5 | On a l’habitude de faire ça |
| 70 | Vincent | 1 | 0.5 | On a l’habitude de faire ça |
| 64 | Adrien | 1 | 0.5 | On a l’habitude de faire ça |
| 65 | Loyse | 0.5 | 0.5 | - |
| 71 | Kamil | 0.5 | 0.5 | - |
| 72 | Vincent | 1 | 0.5 |  |
| 58 | Adrien | 1 | 0.5 | On a l’habitude de faire ça |
| 57 | Loyse | 1 | 0.5 | - |
| 73 | Kamil | 0.5 | 0.5 | - |
| 74 | Vincent | 1 | 0.5 |  |
| 54 | Adrien | 1 | 0.5 | On a l’habitude de faire ça |
| 55 | Loyse | 1 | 0.5 | - |
| 68 | Kamil | 1 | 0.5 | On a l’habitude de faire ça |
| 75 | Vincent | 0.5 | 0.5 |  |
| 56 | Vincent | 1 | 0.5 | On a l’habitude de faire ça |
| 76 | Vincent | 0.75 | 0.5 | On a l’habitude de faire ça |

Remarque générale : Au vu de temps gagné sur ce sprint, nous avons pu mettre en place des améliorations générales du projet.

### Bilan d’itération



#### Bilan de terminaison des histoires

Histoires terminées :

* 9 : Ajout/Suppression des ressources
* 11 : Ajouter/supprimer une mine
* 12 : Ajouter/supprimer une gare

Histoire en cours :

* 10 : Bannir le joueur

#### Vélocité du sprint

6 points d’histoire

#### Replanification

L’histoire 10 (Bannir un joueur) est reportée au sprint suivant (Elle est quasiment terminée, il s’agira juste d’ajouter un bouton dans la GUI)

#### Commentaire général

Tout s’est bien passé. En plus du sprint, nous avons ajouté la mise à jour local automatique du client et fait un refactoring général de l’application.

#### Autocritique

Nous avons attribué plus de temps au projet GEN pour pouvoir refactorer le projet et améliorer son rendu, étapes qui n’étaient initialement pas prévues dans le projet , tout en remplissant les obligations du sprint 6.

## Sprint 7

### Objectifs

Mettre en place un système de commerce entre joueurs

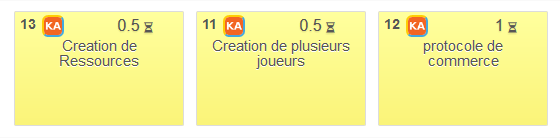
### Stories à faire

10 : Bannir un joueur

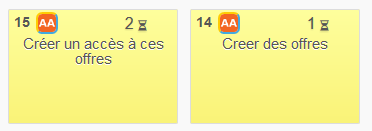
### Tasks

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

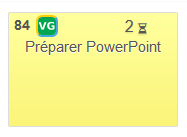
Placer une offre



Acheter des ressources



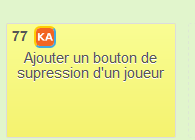
Préparation présentation



Rédaction rapport final

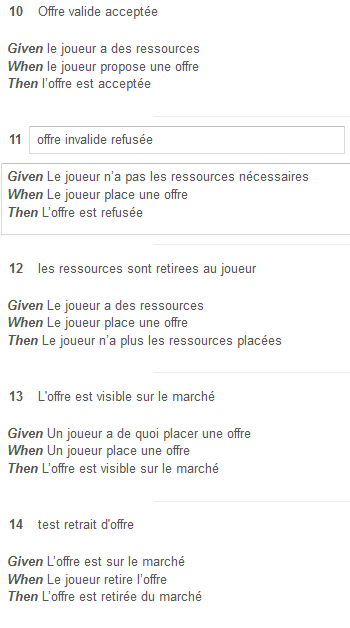


Bannir un joueur

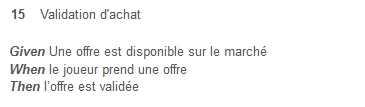


### Test d’acceptation

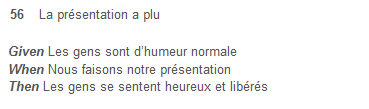
Placer une offre



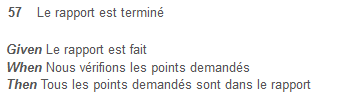
Acheter des ressources



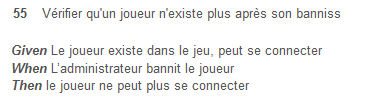
Présentation



Rédaction du rapport



Bannir un joueur



Estimation par tâche [heure/personne]  + bilan personnel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Numéro de tâche | Personne assignée | Durée prévue [heure/homme] | Durée effective [heure/homme] | Commentaire |
| 13 | Kamil | 0.5 |  |  |
| 11 | Kamil | 0.5 |  |  |
| 12 | Kamil | 2.5 |  |  |
| 15 | Adrien | 3 |  |  |
| 14 | Adrien | 1 |  |  |
| 84 | Vincent | 2 |  |  |
| 79 | Loyse | 0.5 |  |  |
| 78 | Loyse | 1 |  |  |
| 81 | Loyse | 1 |  |  |
| 83 | Loyse | 2 |  |  |
| 82 | Vincent | 1 |  |  |
| 80 | Vincent | 1 |  |  |
| 77 | Kamil | 0.5 |  |  |

### Bilan d’itération

#### Bilan de terminaison des histoires

#### Vélocité du sprint

#### Replanification

#### Commentaire général

#### Autocritique

## Sprints à venir



# Etat des lieux

## Ce qui fonctionne

Toutes les fonctionnalités principales de notre programme sont fonctionnelles en l’état.

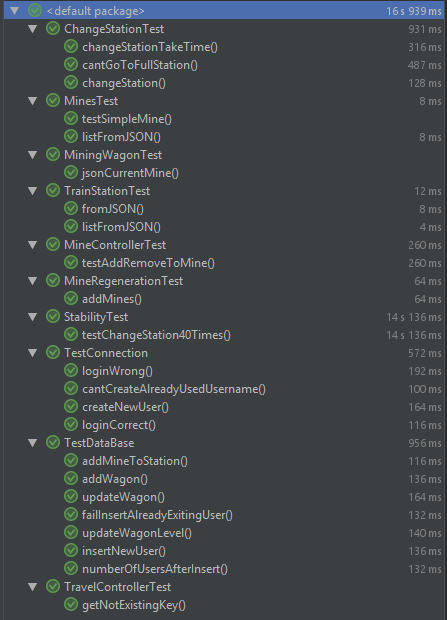
Un joueur peut se connecter au serveur et réaliser toutes les actions proposées initialement, à savoir :

* Se déplacer d’une gare à une autre
* Miner les ressources présentes dans une gare
* Fabriquer des objets à l’aide de ces ressources et éventuellement d’autres objets déjà craftés
* Améliorer les performances des wagons de son train , ou en créer de nouveau
* Commercer avec d’autres joueurs présents sur le serveur au travers d’une interface de commerce où il est possible de déposer, de visualiser et d’acheter des offres.

Du côté administrateur du serveur, les actions possibles sont :

* Créer une gare ou en détruire une existante (si aucun joueur ne se trouve dans la gare)
* Ajouter une mine à une gare en fixant son type de ressources et sa quantité de ressources maximales
* Modifier les ressources d’un joueur ou d’une gare
* Bannir un joueur en supprimant son compte de la base de données

Pour vérifier que toutes les fonctionnalités marchent bien, nous avons écrit 23 tests JUnit qui sont tous fonctionnels



En plus des fonctionnalités de base, nous avons ajouté :

* Un tableau de classement de tous les joueurs du serveur, affichant leur niveau global, le niveau de leur train, leur nombre d’objets.
* Un algorithme de génération déterministe des gares et des mines qui les composent

Nous avons bla bla

## Ce qu’il resterait à développer

# Auto-critique

# 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.

# Annexe : manuel d’utilisation

## Installation :

Nécessite :

* Un gestionnaire de base de données SQL dans lequel il est possible de faire tourner un script pour créer une base de données (Par exemple MySQL)
* Une machine virtuelle qui fonctionne avec java8. ( avec au minimum une installation de JRE (Java Runtime Environnement) qui tourne Java8 ) pour la lecture des .jar

Pour installer le serveur de OTrain

* Dans le gestionnaire SQL, lancer le script « gen\_otrain » qui crée la base de données du jeu
* Executer le fichier « server.jar »
* Le serveur est lancé, il reste à récupérer l’adresse ip liée au serveur pour pouvoir la donner aux clients.

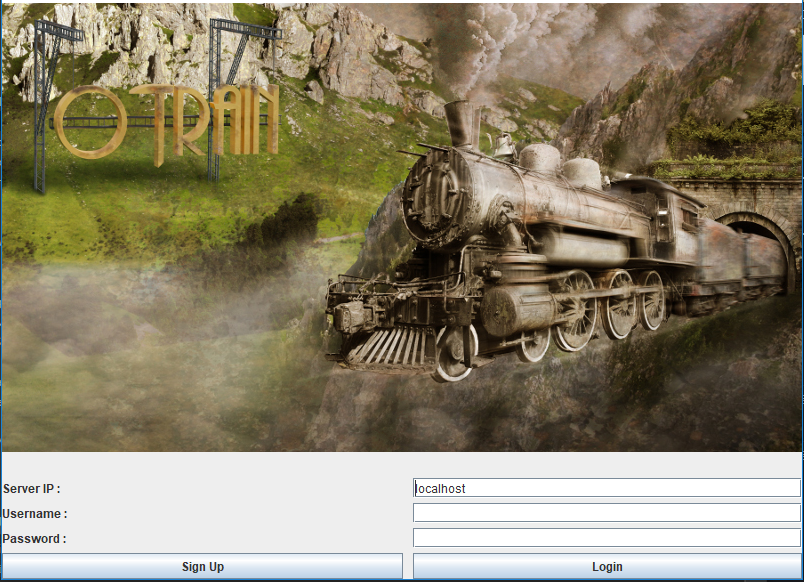
Pour installer le client de OTrain

* Executer le fichier « client.jar »
* Lui donner l’adresse ip du serveur auquel on veut se connecter.

## Utilisation

Lorsqu’ un client OTrain est lancé,

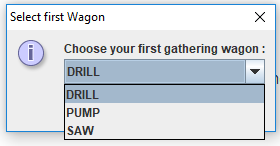
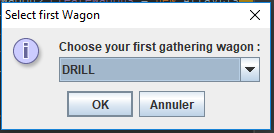
On tombe sur une interface de login sign/up



Choisir le server en fournissant son adresse IP.

Sign Up

* Entrer un nom d’utilisateur et un mot de passe. Chaque nom d’utilisateur doit être unique dans le serveur
* Cliquer sur le bouton Sign Up
* Dans la fenêtre qui s’ouvre, choisir son type de wagon de base

### Login

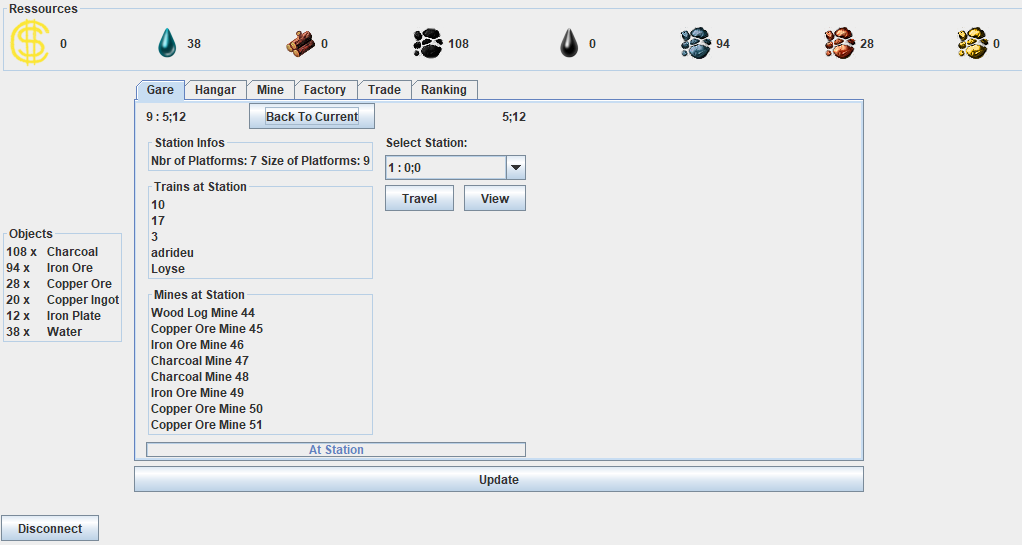
* Entrer votre nom d’utilisateur et votre mot-de-passe
* Cliquer sur le bouton Login

### Déconnexion

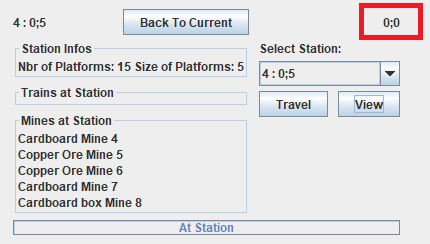
* Cliquer sur le bouton « Déconnexion » en bas à gauche du tableau de bord

### Changer de gare

Une fois connecté, la page d’accueil s’affiche :

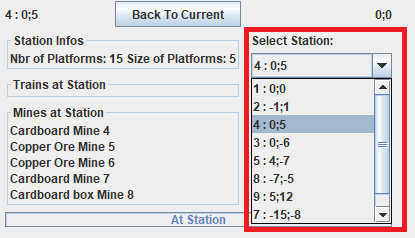
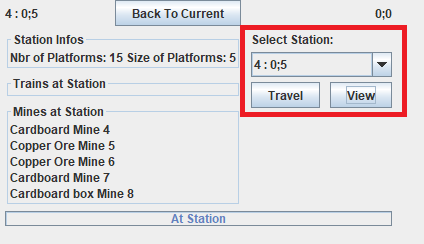


La gare à laquelle se trouve le train est indiquée par ses coordonnées



Il est possible de récupérer la liste des gares aux alentours et de :

* Soit s’y déplacer en cliquant sur le bouton Travel
* Soit s’observer ce qui s’y trouve en cliquant sur le bouton view

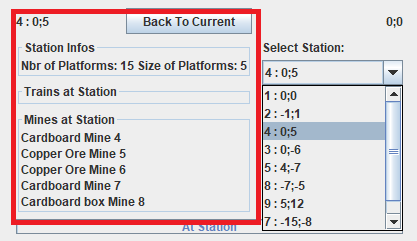
 

#### Affichage des infos d’une gare

L’affichage de la gare sélectionnée donne des informations sur

* Les coordonnées de cette gare
* Le nombre de plateformes et la taille des plateformes de la gare
* La liste des trains se trouvant déjà à cette station
* La liste des ressources trouvables à cette station

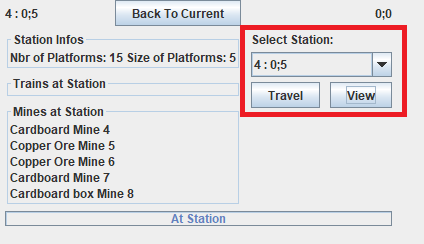
Le bouton « Back to current » permet de retourner à l’affichage de la station à laquelle le train se trouve



#### Déplacement vers une autre gare

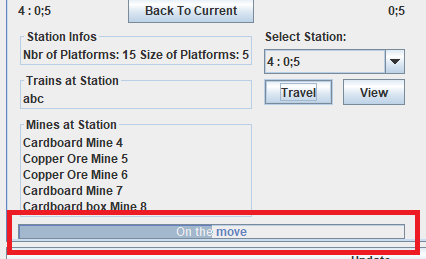
Le déplacement vers une autre gare se fait à deux conditions :

* La gare a assez d’espace pour accueillir le nouveau train. On voit s’il reste de la place en comparant le nombre de plateformes au nombre de train déjà présent dans la gare.
* La gare a une taille de plateforme qui permet d’accueillir tous les wagons du train



On lance un déplacement vers une nouvelle gare en cliquant sur le bouton Travel, après l’avoir sélectionnée.

Le déplacement prend un certain temps et il est possible de voir la progression grâce à la barre de progression en bas de la fenêtre.

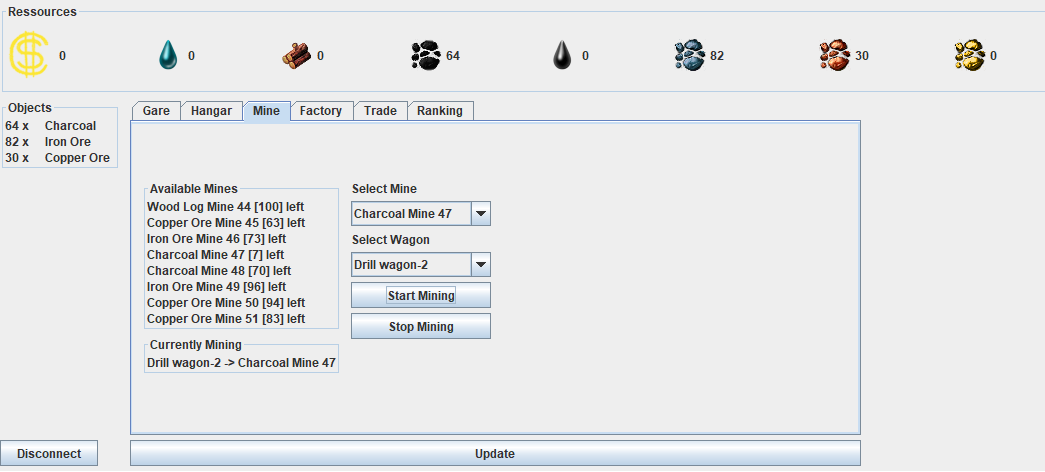


Une fois un trajet lancé , il n’est pas possible de l’annuler.

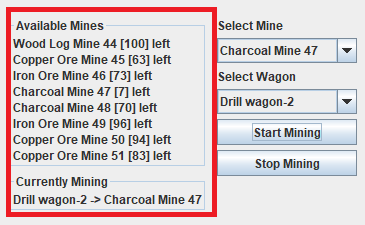
Un train en déplacement ne peut pas récolter de ressources, mais il peut fabriquer des objets, améliorer son train, ou encore commercer avec d’autres joueurs.

### Miner des ressources

Pour récolter des ressources, aller sur l’onglet Miner du tableau de bord

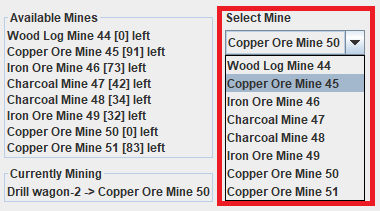
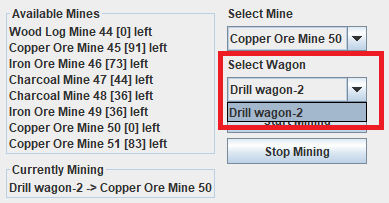


Les tableaux sur la gauche indiquent les mines disponibles à la gare actuelle, ainsi que le ressource actuellement en cours de récolte :



#### Démarrer le minage

Il est possible de sélectionner une mine dans la liste des mines présentes et de sélectionner un type de wagon de minage pour récolter la ressource.

3 types de wagon sont disponibles pour récolter les ressources :

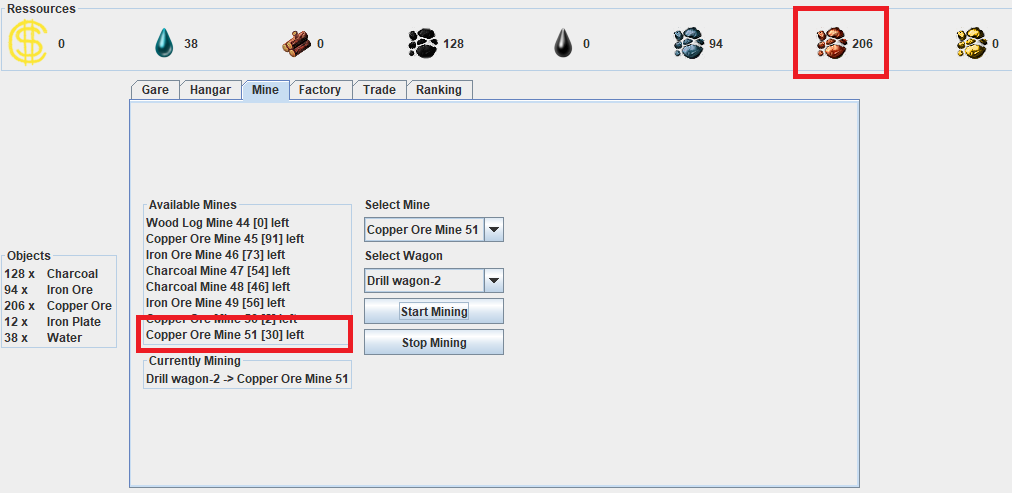
* Les wagon DRILL, qui permettent de récolter les ressources : Iron, Copper, Charcoal, Gold
* Les wagon PUMP, qui permettent de récolter les ressources : Water, Oil
* Les wagon SAW, qui permettent de récolter les ressources : Wood Log

Remarque : dans l’illustration ci-dessus, seul le wagon Drill est disponible, les seules mines qui pourront être minées sont celles de : Iron, Copper, Charcoal, Gold.

Une fois la mine et le wagon sélectionnés, cliquer sur « Start Mining »



Le minage lancé, les ressources sont petit à petit ôtées du côté de la mine pour être ajoutées du côtés des ressources du joueur



Les mines ont un temps de régénération complète aléatoirement choisi entre 30 min et deux heures. Ainsi, si un joueur arrive à l’épuisement de la mine, il continuera à miner la ressource à sa vitesse de régénération.

#### Arrêter de miner

Pour arrêter de miner, cliquer sur le bouton « Stop mining »



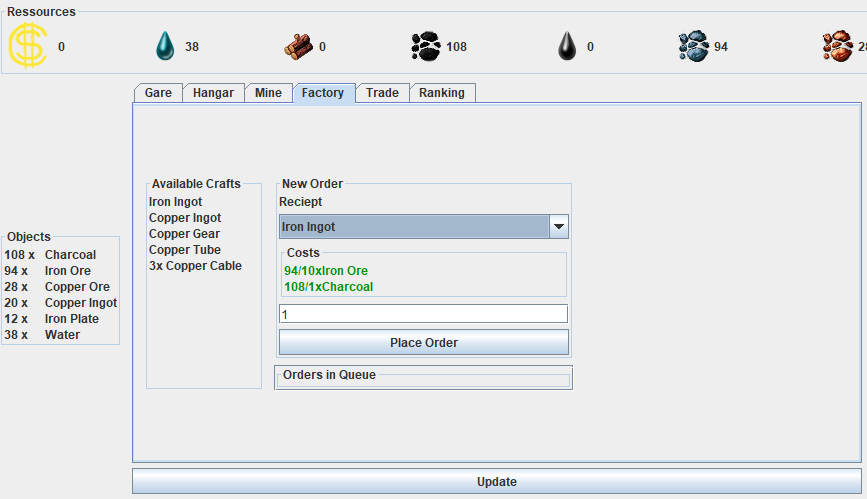
Il est nécessaire d’arrêter de miner lorsqu’on veut :

* Changer de ressource (mine) à récolter
* Entamer un déplacement vers une autre gare

**Attention, le minage ne s’arrête pas quand la mine est vidée**

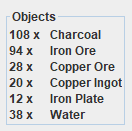
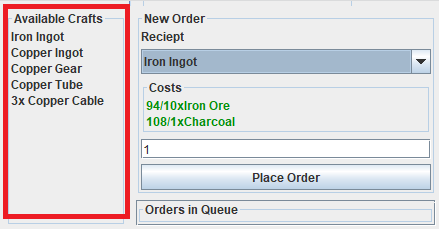
### Fabriquer des objets

On peut fabriquer des objets à partir des ressources du jeu, sous l’onglet Factory

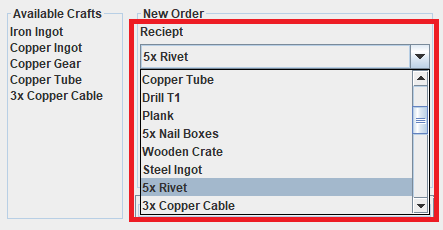
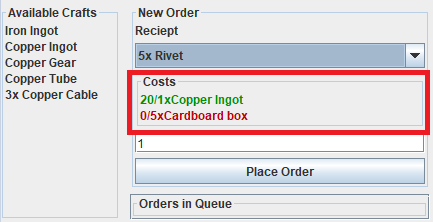


Le joueur peut voir l’ensemble de ses ressources et objets dans le tableau de gauche sur le tableau de bord

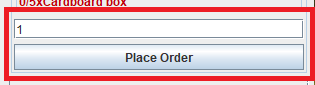
Les objets fabricables avec les ressources actuelles sont disponibles dans un tableau « Available Crafts »

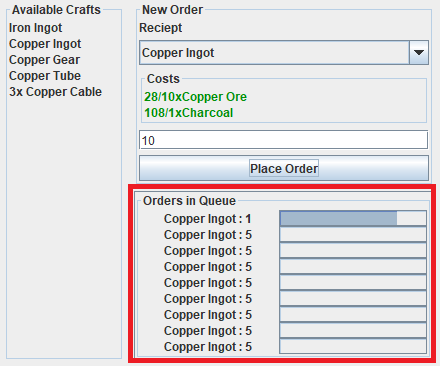
Pour lancer un nouveau craft, il s’agit de choisir un craft dans la liste des recettes. Le coût en ressources nécessaires à la création de l’objet sont indiquées en-dessous, en vert si l’utilisateur les possède déjà, en rouge sinon.

Une fois la recette sélectionnée, la quantité de objet à crafté est encore choisi, puis le craft est lancé en cliquant sur « Place Order ».

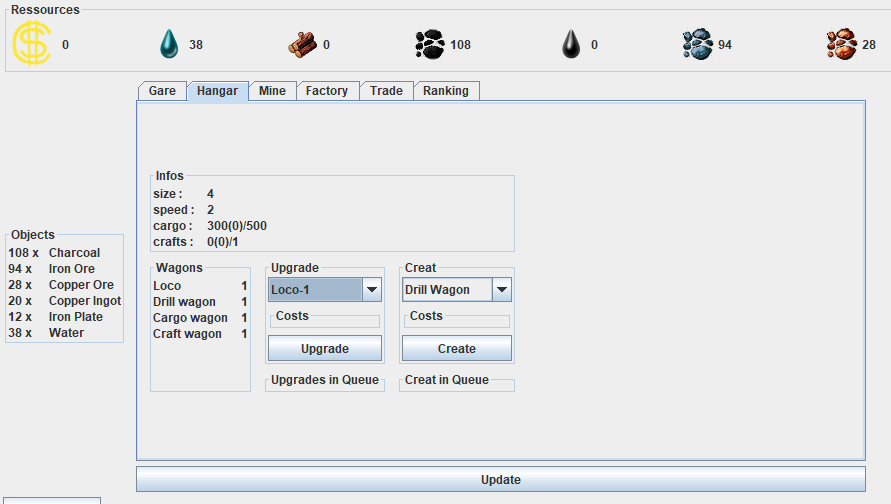


Une fois une série de craft lancée, une barre de progression est démarrée pour montrer le temps restant avant la création de l’objet. Les ressources nécessaires la fabrication de toute la série sont retirées des ressources du joueur et une fois le craft terminé, celui-ci est ajouté à la liste des objets du joueur.

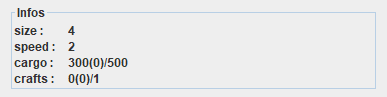
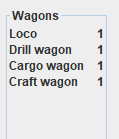


### Amélioration et construction d’un wagon

Pour améliorer ou construire un wagon, l’onglet Hangar est disponible sur le tableau de bord



Le tableau info donne des informations sur l’état courant du train

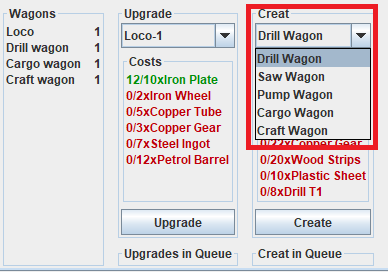
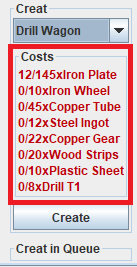
* Size  correspond au nombre de wagon du train
* Speed correspond à la vitesse de la locomotive
* Cargo à la quantité de ressources actuelles (ressource prévues) par rapport à la quantité maximale d’objets accumulables.
* Crafts indique le nombre de crafts lancé (crafts prévus) par rapport au nombre de crafts possible à lancer en parallèle

Le tableau des wagons indique le nombre de wagon que le train possède par type de wagon.

#### Créer un wagon

Pour construire un wagon, choisir le wagon à construire et cliquer sur le bouton « Create »

La liste des ressources nécessaires à la construction sont disponibles dans un tableau « Cost »

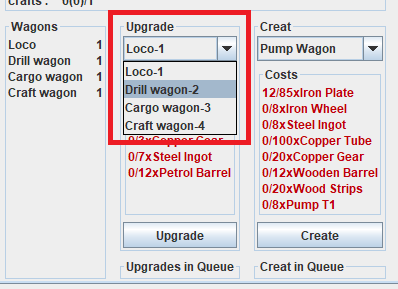
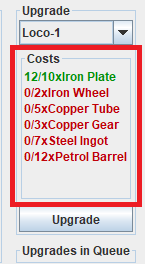
 

Une barre de progression se lance pour indiquer le temps restant avant que le wagon soit construit

#### *Upgrader un wagon*

Pour upgrader un wagon, choisir un wagon à améliorer et cliquer sur le bouton « Upgrade ».

La liste des ressources nécessaires à l’amélioration sont disponibles dans un tableau « Cost »

Une barre de progression se lance pour indiquer le temps restant avant que le wagon soit upgradé

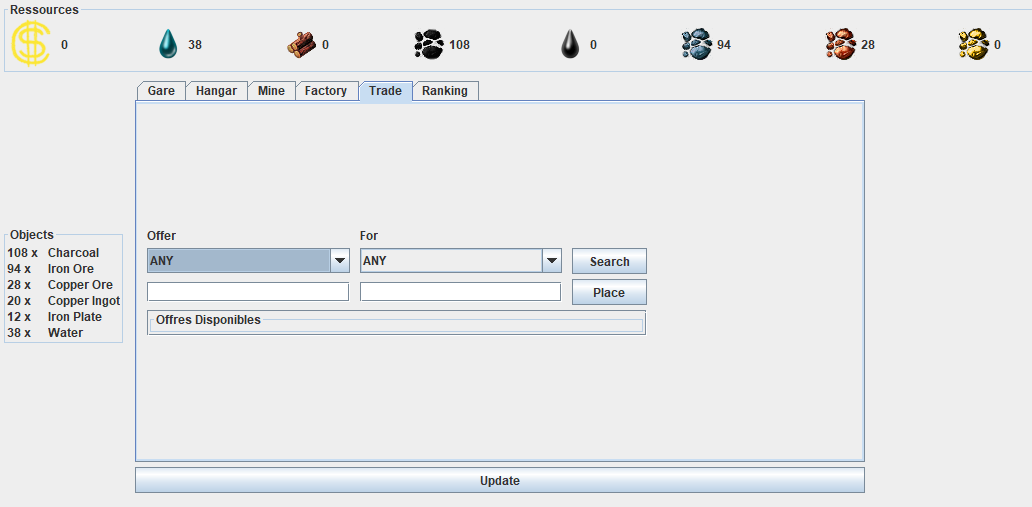
Les wagons upgradés deviennent plus efficaces dans leur rôle :

* Wagon drill, saw, pump : augmentation du nombre de ressource récoltées par secondes
* Wagon Cargo : augmenter de la capacité de stockage du train
* Wagon Loco : augmentation de la vitesse du train

### Trading

Un objectif du jeu est de pouvoir partager des ressources avec les autres joueurs du serveur.

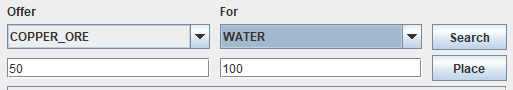
Pour ce faire, un onglet « Trade » est disponible sur le tableau de bord.



#### Placer une offre

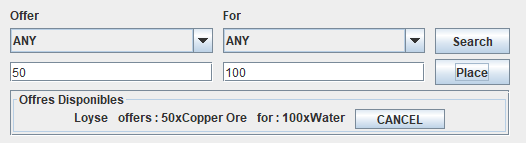
Pour déposer une offre :

* Choisir la ressource à offrir
* Déterminer sa quantité
* Choisir la ressource demandée en échange
* Déterminer son montant
* Cliquer sur le bouton « Place »



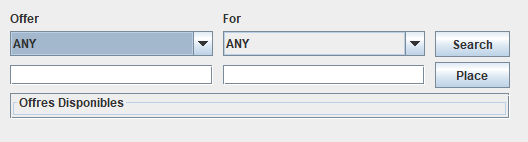
L’offre se place alors dans le tableau des offres disponibles

Une offre déposée sur le marcher peut être annulée à tout moment en cliquant sur le bouton « Cancel » à côté de l’offre.



#### Voir les offres disponibles / acheter une offre

Pour voir les offres disponibles, cliquer sur le bouton « Search »



Les offres disponibles sur le serveur s’affichent dans le tableau « Offres disponibles »

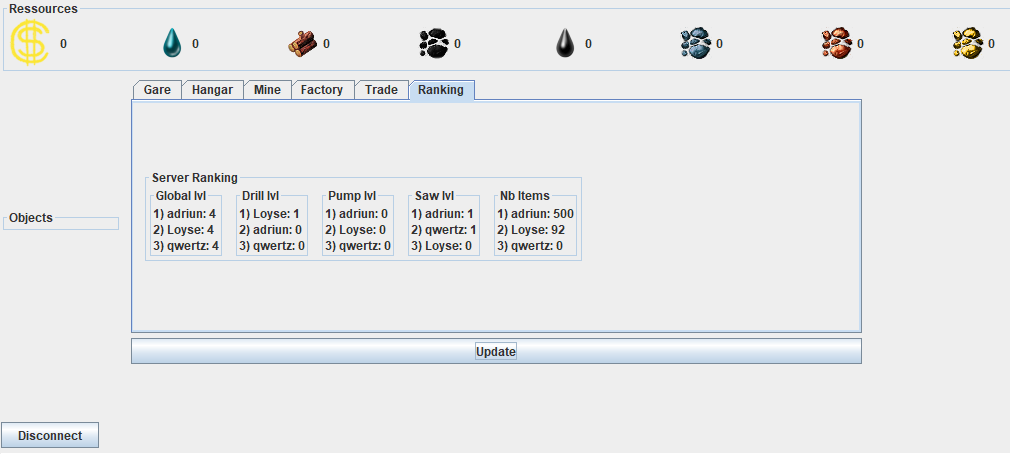


* Choisir une offre intéressante
* Cliquer sur le bouton « BUY »

Si les ressources disponibles sont suffisantes pour acheter l’offre, celle-ci est validée et le résultat de l’échange se retrouve dans les inventaires des deux joueurs.

### Classement

Il est possible de connaître le classement général des joueurs dans l’onglet « Ranking »



### Update

Pour forcer une synchronisation de l’affichage client et du serveur, cliquer sur le bouton « Update » en bas du tableau de bord.

